
Article:
Aad, G. orcid.org/0000-0002-6665-4934, Aakvaag, E. orcid.org/0000-0001-7616-1554, Abbott, B. orcid.org/0000-0002-5888-2734 et al. (2976 more authors) (2025) Software and computing for Run 3 of the ATLAS experiment at the LHC. The European Physical Journal C, 85. 234. ISSN 1434-6044

https://doi.org/10.1140/epjc/s10052-024-13701-w

Eur. Phys. J. C (2025) 85:234
https://doi.org/10.1140/epjc/s10052-024-13701-w

THE EUROPEAN PHYSICAL JOURNAL C

Regular Article - Experimental Physics

Software and computing for Run 3 of the ATLAS experiment at the LHC

ATLAS Collaboration[*]

CERN, 1211 Geneva 23, Switzerland

Received: 11 April 2024 / Accepted: 6 December 2024

Abstract The ATLAS experiment has developed extensive software and distributed computing systems for Run 3 of the LHC. These systems are described in detail, including software infrastructure and workflows, distributed data and workload management, database infrastructure, and validation. The use of these systems to prepare the data for physics analysis and assess its quality are described, along with the software tools used for data analysis itself. An outlook for the development of these projects towards Run 4 is also provided.

Contents

[*] e-mail: atlas.publications@cern.ch

Published online: 06 March 2025

1 Introduction

ATLAS [1,2], one of the general-purpose experiments at the Large Hadron Collider (LHC), has recently begun its third collision data taking campaign (Run 3) with proton–proton and heavy-ion collisions. The data collected since its operational start in 2010 are used for a broad physics programme, from precision measurements of the Standard Model, including the discovery of the Higgs boson [3,4], to searches for various manifestations of new physics. This paper describes the Software and Computing infrastructure of the ATLAS experiment, in its current state in the midst of Run 3, along with some of the major upgrades that were done in preparation for Run 3.

An overview of the ATLAS detector, the operation of the experiment, and the software and computing resources necessary to support it, is given in Sect. 2. The experiment relies on resources provided by the Worldwide LHC Computing Grid (WLCG) [5,6], and has developed an extensive set of tools to capture additional resources. The scale of these resources is laid out in Sect. 2.1, along with a brief overview of the ATLAS operational data-taking processes and timescales. A description of the ATLAS experiment's detector is given in Sect. 2.2. From the moment the data leave the detector, they undergo a series of processing steps, calibrations, and recalibrations. An overview of the software chain necessary to support the processing of both real and simulated particle collision data is presented in Sect. 2.3. Following this processing the data are transferred around the world, their quality is examined carefully, and they are processed into many derived data formats in preparation for physics analysis. This process, including the chain of custody, is described in Sect. 2.4.

A detailed discussion of the ATLAS software that supports data production, processing, and simulation is undertaken in Sect. 3. The ATLAS software has evolved considerably since collision data were first recorded in 2009, growing to match the complexity of the data analysis programme. The software infrastructure, described in Sect. 3.1, has undergone major changes during this period, most recently to support multithreading. The software also includes a revised configuration layer, detailed in Sect. 3.2, and infrastructure for the handling of detector conditions information, which is described in Sect. 3.3. The Event Data Model (EDM), described further in Sect. 3.4, has also evolved to better support downstream data analysis. The modelling of the data requires a detailed detector description, which is described in Sect. 3.5. Many modern data analyses and portions of the upstream

data processing software rely on machine learning tools; the integration of and support for these in ATLAS is described in Sect. 3.6.

An extensive Monte Carlo (MC) simulation and data processing chain was developed to support the experiment, and is described in Sect. 4. This chain includes 'event generation', i.e. the modelling of the initial proton–proton, proton–ion, or ion–ion collision (described in Sect. 4.1), detector simulation (described in Sect. 4.2), and 'digitisation' (modelling of the detector electronics, described in Sect. 4.3). Both the real detector data and MC simulation are passed through a common reconstruction process (hereafter referred to as *the reconstruction*, described in Sect. 4.4) and first stage of processing for analysis known as 'derivation making' (described in Sect. 4.5). The forward systems are treated with special workflows through many of these steps, which are summarised in Sect. 4.6. Section 4.7 describes the support of all of this software for future configurations of the detector, which must be maintained to provide accurate projections of physics analyses and prepare for future data-taking runs.

The data and MC simulation are extensively vetted and validated before being declared good for analysis, and before the start of any new production campaign, as discussed in Sect. 5. The assessment of data quality is described in Sect. 5.1. The preparation for both new campaigns of MC simulation production at the beginning of a data-taking period, and the large-scale reprocessing of data at the end of a data-taking period, are complex processes that involve many different steps and require significant time. The software validation process for these campaigns is explained in Sect. 5.2, and the computing performance of the various steps of the Run 3 software chain is detailed in Sect. 5.3. The steps required to begin a new MC simulation or data reprocessing campaign are laid out in Sect. 5.4.

Section 6 presents the significant technical infrastructure required for the experiment to efficiently develop and deploy software and operate at scale. The infrastructure for development, building, and deployment of the complex software stack is described in Sect. 6.1. The database infrastructure, handling calibrations and evolving conditions information, is described in Sect. 6.2. There are several systems for handling of metadata at various stages of processing; these are described in Sect. 6.3.

The ATLAS distributed computing systems are described in Sect. 7. The management of the MC simulation and real detector data, as well as all subsequent data forms and formats, is described in Sect. 7.2. Section 7.3 describes the workflow management system that was developed to orchestrate the running of ATLAS software around the world, including the use of resources beyond those provided by the WLCG. These are all subject to extensive monitoring to ensure their efficient use; the monitoring and analytics applied are detailed in Sect. 7.4.

Provisions for downstream data analysis are discussed in Sect. 8. After being processed into derivations, physics data are manipulated using a large suite of data analysis tools. An overview of these tools is given in Sect. 8.1. Data visualisation techniques are used with both simulated and real data to validate the performance of the detector simulation, check for unusual detector conditions or reconstruction features, study interesting events, and educate the public about the physics programme of ATLAS. Images that show physics events and processes within the ATLAS detector, called *event displays*, are prepared for these purposes. The use of event displays and the software developed to produce them are introduced in Sect. 8.2. Instruction and support for ATLAS members making use of all of these tools is provided through documentation and regular tutorials, which are discussed in Sect. 8.3.

While the approaches developed to date are sufficient for the challenges of Run 3, plans for significant upgrades to prepare for the High-Luminosity LHC (HL-LHC), scheduled to begin operations in 2029, are already underway. Significant revisions to the software and computing itself are also in preparation. Some of these are described in Sect. 9, along with the outlook for the future of software and computing in the experiment.

2 Overview and resources

This section provides an introduction to the ATLAS experiment, including operational data-taking and resource requirements, and an overview of the detector itself. An executive summary of the software chain needed to process data collected by the detector (or generated by MC simulations) to a point where they are in a format appropriate for downstream physics analysis is also presented, along with a high-level description of how data are collected from LHC particle collisions.

2.1 Resource scope

The ATLAS experiment comprises almost 6000 members, about 3000 of whom are authors of ATLAS scientific papers. This section provides a brief overview of the data-taking environment for ATLAS. The scope of the computing and software resources required to support its operation, and the needs of the collaboration at large, is introduced.

About 140 full-time equivalents (FTE) of effort, divided among about 450 people, is spent on 'central' software development and distributed computing support,[1] and a further

[1] In addition to central software development and computing support, further efforts are needed to maintain and develop detector-sub-system-

150 FTEs of effort is spent on maintaining the around 100 WLCG computing centres constituting the distributed computing infrastructure supporting ATLAS. A small fraction of that effort, around 10–20 FTEs, includes real-time *shift* work for monitoring resources, tests, and software updates.

2.1.1 ATLAS operational overview

In 2022, the LHC began a third prolonged period of data-taking, called Run 3, following on from Run 1 (2009–2012) and Run 2 (2015–2018). The terms Run 1/2/3 refer to the multi-year operational data collection campaigns of the ATLAS experiment. Each *Run* is followed by a *Long Shutdown* (LS) during which significant upgrades and repairs to the detector can be made; the most recent of these was Long Shutdown 2 (LS2, 2019–2021). Each Run is divided into years, which are further subdivided into data-taking *periods*, wherein detector and collider conditions are generally stable. The upgrades of the detector are referred to in *phases*, where Phase-0 indicates the upgrade during LS1, Phase-I the upgrade during LS2 [7–10], and Phase-II the upgrade during the upcoming LS3 [11–17]. This last upgrade will be significant, in preparation for the high-luminosity LHC (HL-LHC) era wherein the instantaneous luminosity of the accelerator will be increased roughly 3-fold.

A period during which beams are circulating in the LHC ring uninterrupted is called a *fill*, and ATLAS will typically record collision data continuously during an LHC fill in an ATLAS *run*.[2] A run is divided into individual *luminosity blocks*, or lumi blocks, which are roughly one minute long and represent the smallest unit of stable conditions for data analysis.

During proton–proton collision data taking, protons collide in ATLAS about two billion times per second. The protons in the beams are grouped into bunches, and the collisions occur during *bunch crossings*, which happen about 30 million times per second. Each bunch crossing results in a physics *event*, and each crossing is labelled with a unique *bunch crossing identifier* (BCID). A single proton–proton collision from a given bunch crossing is chosen as the collision of interest. Other proton–proton interactions from within the same bunch crossing, and from neighbouring bunch crossings, can present a *background* to the chosen collision. These other proton–proton interactions are collectively referred to as *pile-up*, and the treatment of pile-up in the processing of data and MC is discussed in Sect. 4. The average number of pile-up collisions in a BCID is denoted by $\langle\mu\rangle$. Of these 30 million events per second, about 3500 complete events are written out in full by ATLAS, selected by a multi-stage trigger system [18]. Small parts of many more events are recorded as well for calibration purposes and specialised data analyses.

During Run 2, ATLAS recorded about 18 billion complete events. Generally, 2–3 times more simulated events are produced than real detector events are recorded. This volume of MC simulation is required to keep statistical uncertainties in predictions small, and to provide capacity for simulation of new physics signals.

2.1.2 Computing resources

ATLAS distributed computing resources comprise about one million cores of computing, 350 PB of disk, and almost 450 PB of tape storage. The computing is distributed over a variety of sites:

1. The CERN *Tier-0* site, where the initial data processing (known as *prompt* data processing) is done, and where software releases are built and tested every night.
2. WLCG sites, which include 11 ATLAS Tier-1 sites and around 70 ATLAS Tier-2 sites. The distinction between the two is described in Ref. [19]; generally speaking, Tier-1 sites are larger and include archival tape storage.
3. A variety of high-performance computing centres, some of which are provided via the standard WLCG pledge mechanism (i.e. they are accounted for as a part of the Tier-1 and Tier-2 resources), and some of which come independent of the WLCG (see Sect. 7.3.2).
4. The high-level trigger processing farm, located near the detector to minimize data transfer latency, which can be used for offline data processing when the experiment is not taking data (see Sect. 7.3.2).
5. BOINC [20–22] resources from volunteer computing provided by ATLAS@Home (see Sect. 7.3.2).
6. Commercial cloud resources, integrated into the experiment like standard WLCG sites where possible (see Sect. 7.2.6).

These distributed computing resources are used to process all of the real and simulated data produced by the ATLAS experiment. The resources pledged by the WLCG sites are tracked in the Computing Resource Information Catalogue (CRIC) [23]. Generally speaking, all of the disk and tape used for long-term storage is provided as pledge, and about half of the CPU used by the experiment is provided as a part of pledge. Because a significant portion of the CPU is unpledged, there are large variations ($+300k/-100k$ cores from the average) in the amount of CPU available at any given time.

Footnote 1 continued
specific software, or software to support dedicated physics analysis areas.

[2] This overlap of terminology is unfortunate but ubiquitous at the LHC. Throughout this paper, Run 1/2/3 refers to the multi-year running time, and 'run' refers to a single, several-hour data taking period when beams are continuously circulating in the LHC.

2.1.3 Software resources

An extensive software suite [24] is used in the reconstruction and analysis of real and simulated data, in detector operations, and in the trigger and data acquisition systems of the experiment.[3] The terms *online* and *offline* are often used to distinguish between software and services that are run during real-time data-taking for the trigger and data acquisition systems (*online software*), and those that are used for subsequent data processing (*offline software*).

The overall offline software framework, Athena [25], is described in Sect. 3.1. The collaboration's software is all open source, published in a publicly accessible GitLab repository [26]. It is available under the Apache 2.0 License [27], with Copyright held by CERN on behalf of the collaboration. Every ATLAS Collaboration member can contribute to the software through *merge requests* to the active branches of the software repository. These merge requests are reviewed by a team of review shifters and experts, to ensure that only high-quality code that conforms with ATLAS conventions and passes a comprehensive spectrum of Continuous Integration (CI) tests enters the repository. More detail on ATLAS software development and release processes can be found in Sect. 6.1.

2.2 ATLAS detector

The ATLAS detector [1,2] at the LHC covers nearly the entire solid angle around the collision point.[4] It is located at Point 1 of the LHC; 'Point 1' therefore appears in the name of several software projects. The ATLAS detector consists of an inner tracking detector surrounded by a thin superconducting solenoid, electromagnetic and hadron calorimeters, and a muon spectrometer incorporating three large superconducting air-core toroidal magnets.

The inner-detector system (ID) is immersed in a 2 T axial magnetic field and provides charged-particle tracking within $|\eta| = 2.5$. This system provides essential information for the reconstruction of *physics objects* such as electrons and photons [28], muons [29], τ-leptons [30], and jets [31], as well as for identification of jets containing b-hadrons [32], and for event-level quantities that use charged-particle tracks as input. The high-granularity silicon pixel detector covers the interaction region and typically provides four measurements (often called 'hits') per track, the first hit normally being in the insertable B-layer (IBL) installed before Run 2 [33,34]. It is followed by the silicon microstrip tracker (SCT), which usually provides eight measurements per track. These silicon detectors are complemented by the transition radiation tracker (TRT), which enables radially extended track reconstruction up to $|\eta| = 2.0$. The TRT also provides electron identification information based on the fraction of hits (typically 30 in total) above a higher energy-deposit threshold corresponding to transition radiation, which is absorbed by the gas mixture filling the TRT straws. During Run 1, several leaks in TRT active-gas exhaust pipes developed. With the number of leaks expected to increase with higher luminosity operation, continuing operation with the baseline xenon-based gas mixture in the TRT became unaffordable. Leaking modules were updated to operate with an argon-based gas mixture for Run 2, and for Run 3 the new argon-based gas mixture is used for the entire barrel and parts of the endcaps. While particle identification performance is largely preserved in the endcap regions, it is significantly reduced in the barrel region due to poor absorption of transition radiation photons by the argon gas.

The calorimeter system covers the pseudorapidity range $|\eta| < 4.9$. Within the region $|\eta| < 3.2$, electromagnetic calorimetry is provided by barrel and endcap high-granularity lead/liquid-argon (LAr) calorimeters, with an additional thin LAr presampler covering $|\eta| < 1.8$ to correct for energy loss in material upstream of the calorimeters. Hadronic calorimetry is provided by the steel/scintillator-tile calorimeter, segmented into three barrel structures within $|\eta| = 1.7$, and two copper/LAr hadron endcap calorimeters. The solid angle coverage is completed with forward copper/LAr and tungsten/LAr calorimeter modules optimised for electromagnetic and hadronic energy measurements respectively. The calorimeters play an important role in the reconstruction of physics objects such as photons, electrons, τ-leptons and jets, as well as event-level quantities such as missing transverse momentum (with magnitude denoted by $E_{\mathrm{T}}^{\mathrm{miss}}$) [35].

The muon spectrometer (MS) comprises separate trigger and high-precision tracking chambers measuring the deflection of muons in a magnetic field generated by the superconducting air-core toroidal magnets. The field integral of the toroids ranges between 2.0 and 6.0T m across most of the detector. Three layers of precision chambers, each consisting of layers of monitored drift tubes (MDTs), cover the region $|\eta| < 2.7$, except in the innermost layer of the endcap region, where the background is highest and layers of small-strip

[3] Reference [24] and the references therein also describe some of the firmware that is used in the ATLAS data acquisition system; that software is not described further here.

[4] ATLAS uses a right-handed coordinate system with its origin at the nominal interaction point (IP) in the centre of the detector and the z-axis along the beam pipe. The x-axis points from the IP to the centre of the LHC ring, and the y-axis points upwards. Cylindrical coordinates (r, ϕ) are used in the transverse plane, ϕ being the azimuthal angle around the z-axis. The pseudorapidity is defined in terms of the polar angle θ as $\eta = -\ln\tan(\theta/2)$ and is equal to the rapidity $y = \frac{1}{2}\ln\left(\frac{E+p_z c}{E-p_z c}\right)$ in the relativistic limit. Angular distance is measured in units of $\Delta R \equiv \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$ for geometric quantities, and in units of $\Delta R_y \equiv \sqrt{(\Delta y)^2 + (\Delta\phi)^2}$, where y is the rapidity, for physical quantities like momenta.

thin-gap chambers and Micromegas chambers both provide precision tracking in the region $1.3 < |\eta| < 2.7$. The muon trigger system covers $|\eta| < 2.4$ with resistive-plate chambers (RPCs) in the barrel, and thin-gap chambers (TGCs) in the endcap regions, and small-strip thin-gap chambers and Micromegas chambers in the innermost layer of the endcap. During Run 2 the innermost endcap stations, covering $1.3 < |\eta| < 1.8$, were cathode-strip chambers (CSCs). These were replaced by new small wheels (NSW) [7] during LS2, to improve tracking efficiency and resolution in the high rate environment of Run 3. The ATLAS software supports the simulation and reconstruction of data from both Run 2 and Run 3, and therefore both the detector technologies are available.

Four forward detector systems are installed around the interaction point. The luminosity is measured mainly by the LUCID-2 detector that records Cherenkov light produced in the quartz windows of photomultipliers located close to the beampipe. The Zero-Degree Calorimeter (ZDC), located about 140 m from the interaction point, measures neutral particles. It is used primarily during heavy ion data taking, both in the trigger and offline event selection. The ATLAS Forward Proton (AFP) detector, located 210 m from the interaction point, is used primarily to study diffractive physics at low instantaneous luminosity. The Absolute Luminosity For ATLAS (ALFA) forward proton spectrometer comprises Roman pot detectors placed about 240 m from the interaction point on both the sides of the detector. It is used both for luminosity information and to measure the total proton–proton scattering cross section.

Events are selected by the first-level trigger system implemented in custom hardware, followed by selections made by algorithms implemented in software in the high-level trigger (HLT) [18,36]. The hardware trigger accepts events from the 40MHz bunch crossings at a rate below 100kHz, which the high-level trigger further reduces to record complete events to disk at an average rate of about 3kHz.

The Run 3 detector configuration benefits from several upgrades compared with that of Run 2 to maintain high detector performance at the higher pile-up levels of Run 3. The improvements from the NSW provide higher redundancy and a large reduction in fake muon triggers. The trigger system also benefits from new LAr digital electronics with significantly increased granularity. Other updates and further details are provided in Ref. [2].

2.3 Software workflow

The software workflows for the MC simulation production and the main stages of the data processing are depicted in Fig. 1. The workflow for data involves other steps that are discussed in Sect. 2.4. A detailed description of the data and MC simulation production chain is then given in Sect. 4.

The first stage of MC simulation production is called *event generation*. Here, an event generator configuration is run to create HepMC output [37]. This is written to an output EVNT file, containing the particles output by the event generator together with metadata expressing the configuration of the generator job. The process of event generation is described in Sect. 4.1.

Following event generation, the resulting particles are passed through a detector *simulation* [38–42]. The output from this processing step is stored as a HITS file.[5] The HITS output contains records of energy deposits from each particle, with associated timing information, for each subsystem; *truth* information about the simulated behaviour of particles during the event, appended to the generator particle record; and additional metadata concerning the configuration of the simulation job. The process of detector simulation is described in Sect. 4.2.

The simulated energy deposits are then run through a *digitisation* step, in which the detector electronics are modelled, resulting in RDO (Raw Data Object) output, which is conceptually similar to the raw data collected from the detector. During this stage, pile-up is modelled by the addition of background events, using one of several technical options: the overlay of many simulated events, a single pre-digitised collection of simulated events, or a specially recorded data event. The RDO output files also pass along the truth records from the simulation and include further records of the correspondence between true particles and detector signals, as well as additional metadata. This digitisation process is detailed in Sect. 4.3.

The RDO file is then passed through a *trigger simulation* stage, where a *menu* of triggers specific to the MC simulation process is simulated and decisions and key trigger object collections are added to the output. Both the hardware and high-level triggers are reproduced. This step may be performed using an older software release to appropriately simulate a trigger that was run online some years ago and is no longer supported in the newest releases. The output of this step is written to an *RDO Trigger* file, which is an RDO with trigger information and metadata added to the file. The trigger is described in significantly more detail in Ref. [18].

The data coming from the detector, called RAW, are written in a custom bytestream format [9]. These data are most often processed directly. They can also be filtered to create derived RAW (DRAW) datasets for processing with special settings (e.g. lower momentum thresholds or modified reconstruction configurations for special analyses).

[5] In this paper, *HIT* will refer to an energy deposition during detector simulation, which is written to a HITS file, while *hit* will refer to a position measurement along a charged particle trajectory or other physical detector measurement.

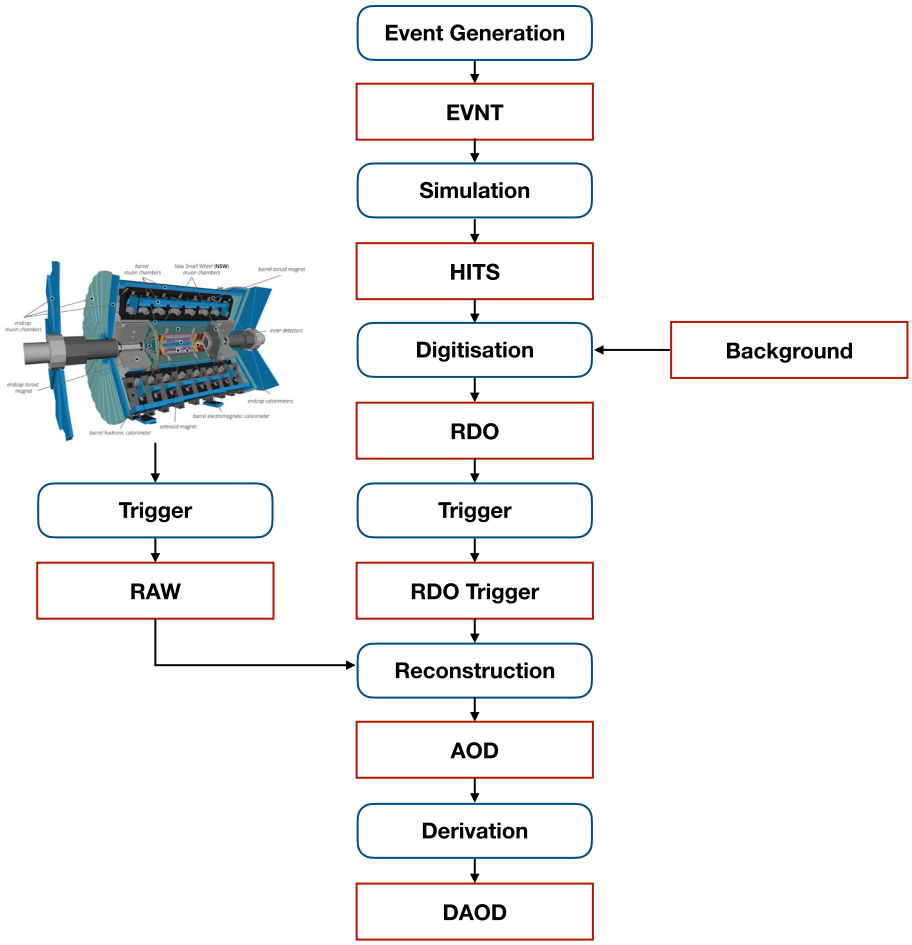


Fig. 1 The standard software workflow of the ATLAS experiment. Processing steps are represented by blue ovals, with output formats represented as red boxes. The various steps and data formats are described in the text. The background entering digitisation may be additional simulated HITS files, pre-digitised RDO files, or specially processed RAW detector data

From this point on, the workflows for data and MC simulation are the same, with the next step being *reconstruction*. The detector signals are converted into physics objects (electron candidates, muon candidates, and so on). The output is written to an AOD (Analysis Object Data) file; in special cases, a larger ESD (Event Summary Data) file or other custom format might be written. The reconstruction process is described in Sect. 4.4.

The next stage after this main reconstruction step is the *derivation* production. This is primarily a data-reduction operation, but it also includes the reconstruction of some secondary physics objects for which only the inputs were reconstructed and stored in the AOD. For example, jets are built during derivation-making from particle flow objects stored in the AOD; similarly, heavy flavour tagging is performed based on these jets and tracks stored in the AOD (see Sect. 4.4 for more about these reconstruction steps and objects). The step also achieves data reduction by the calculation of variables that consolidate or summarise several inputs. Many different derivations might be written out; these all share a common file format and are called DAODs, but they differ in event selection and amount of information written for each physics object. Derivations can also be made directly from the EVNT files if generator output is being analysed. A single derivation format might serve one or several analyses, or might be used for calibration purposes. Derivations are discussed in Sect. 4.5.

Depending on the number of events in the output files and to provide large, uniform files better suited to large storage systems, a *merge* step is run after some processing steps. This merging is typically optimised within the production system to provide the desired number of events per output file.

Generally speaking, these steps are all performed centrally using WLCG resources, called *the Grid*, and any subsequent steps of an analysis are under the control of the individual analyser or data analysis team. Teams might choose to create further-reduced data formats on the Grid, or to transfer their derivations to a local processing centre for reduction and further analysis. These later steps are described in Sect. 8.

2.4 Data flow

The flow of data from the ATLAS detector for prompt reconstruction, and the procedures employed to ensure these are data of sufficient quality for physics analysis, are described in the context of Run 2 in Ref. [43]. The ATLAS data flow in Run 3 is similar. Most data selected by the trigger are promptly reconstructed at the ATLAS Tier-0 computing

facility, before being distributed to the Grid for production of derivations (as described in Sect. 4.5) and further analysis. Data reprocessing, which involves repeating the data reconstruction process completely (for reasons discussed in Sect. 5.4.2), normally takes place directly on the Grid rather than at the Tier-0 site to take advantage of the larger pool of available resources.

The data are organised into *streams*, each stream being a set of defined trigger selections and prescales (fractions of events to accept for a given trigger) that contain all events recorded to disk after satisfying the selections.

- *Physics* streams contain data that are potentially interesting for physics analysis;
- *Calibration* streams exist for particular calibration purposes;
- the *Express* stream is a special stream that contains a representative subset of the physics streams;
- *Delayed* streams are a special class of physics streams that contain data that may be processed at a later time (e.g. at the end of the year), rather than promptly;
- the *Debug* stream includes a small fraction (typically of order $10^{-}7$) of events that are flagged for further investigation, for example because the trigger processing failed or timed out; and
- *TLA* (Trigger-Level Analysis) streams include events for which only a small subset of the event data (e.g. only the jets identified by the trigger) is preserved. This technique allows events to be written at a high rate without consuming significant resources.

Before processing, all these data are stored in the RAW format. To provide a backup, each run's RAW data are stored at the Tier-0 site and at one Tier-1 site.

The nominal flow of data from through the reconstruction chain is illustrated in Fig. 2. Before the relatively large physics streams are processed, a *calibration loop* (which typically lasts for 48 hours) begins. During this time the express stream and several calibration streams are processed at the Tier-0 site. The data from this rapid processing are used to update calibration constants and detector conditions in the conditions database (see Sect. 6.2.2) for items such as the alignment of the ATLAS ID system, the measured LHC beam spot position at ATLAS, and channel calibration, electronic noise, data corruption or disabled modules within detector sub-system components. These data are additionally used for operational data quality monitoring, as discussed in Sect. 5.1. The processes applied in the calibration loop are detailed in Ref. [43]. Once the calibration loop is completed, this improved conditions information is then applied in the prompt processing of the physics streams. Further conditions updates can be made following the bulk data processing, making use of the entire promptly processed data. Generally, high precision data analyses operate exclusively on reprocessed data, while some preliminary or lower-precision analyses might use data immediately after the first processing.

3 Core software components

The ATLAS software framework supports data production and processing, MC generation and simulation, and downstream analysis of the ATLAS detector data. The codebase is divided into approximately 2000 packages, within which groups place code with a common functionality or aim. These packages are gathered into several partially-overlapping subsets called *Projects*, which can be compiled together. The broadest selection form the basis for Athena, the general-purpose offline software project. Other projects contain more limited sets of packages and support specific use-cases such as Simulation, as discussed further in Sect. 6.1.1. ATLAS software has external software dependencies on roughly 200 high-energy physics, data science, and general Linux software packages. In addition, the offline software depends on approximately 200 packages from the ATLAS online detector software system (TDAQ common; see Sect. 6.1.1).

The Athena code, consisting of over 50,000 unique files, is mostly C++, configured using PYTHON and built using CMAKE [44] (see Sect. 6.1.1). The full breakdown of the code base by language from a snapshot in time of the repository (which is continuously evolving) is shown in Table 1 [45]. Most of the custom configuration code listed in the table is for data quality monitoring. The XML files hold a mixture of configuration information (e.g. for the trigger system), data (e.g. for the muon geometry), and configuration for dictionaries (class descriptions for I/O) in ROOT [46,47].

The core infrastructure of the software is presented in Sect. 3.1, and the configuration infrastructure is described in Sect. 3.2. During execution, the software often needs access to conditions data (e.g. alignments and calibrations), which is discussed in Sect. 3.3. The data processed by this software must be represented in a way that ensures common access interfaces, internal consistency and long-term maintainability. The ATLAS *event data model* was developed for this purpose, and is presented in Sect. 3.4. To ensure an accurate description of the ATLAS detector is available, a detector description system was developed, as described in Sect. 3.5. Machine learning tools are increasingly being used at various stages of the data processing chain. A summary of those most commonly in use within the ATLAS Collaboration is given in Sect. 3.6.

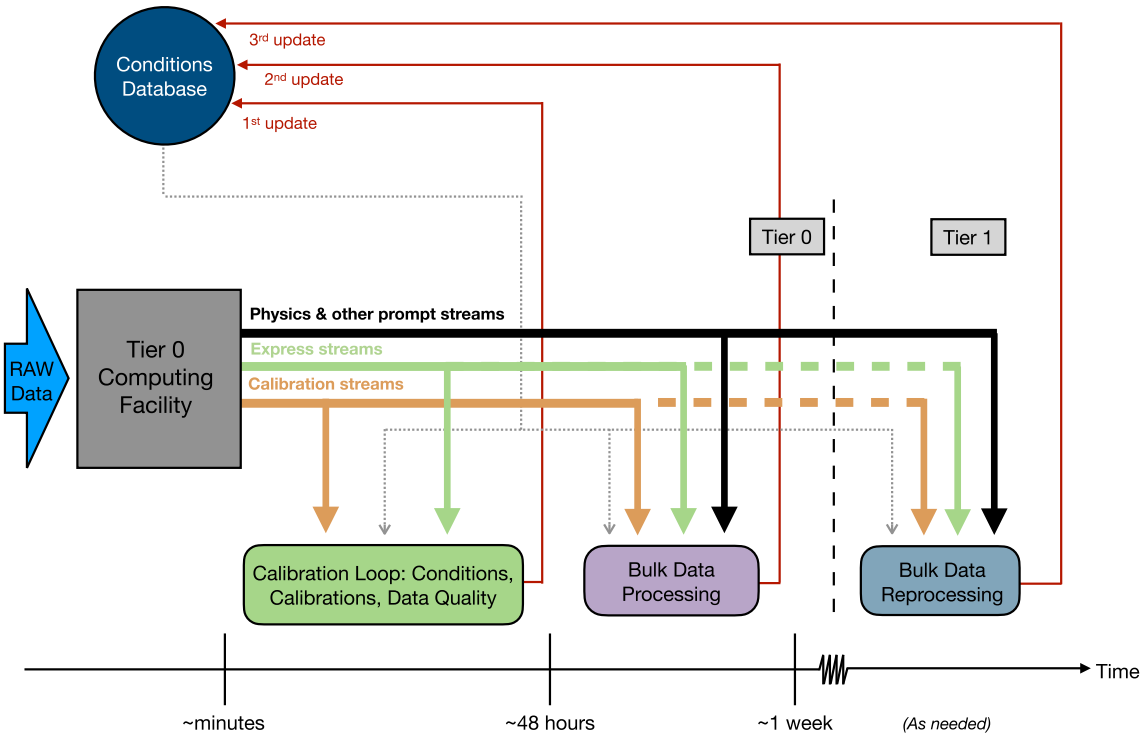


Fig. 2 Schematic diagram illustrating the nominal flow of ATLAS data from initial reconstruction of a subset of streams during the calibration loop and the bulk processing of promptly reconstructed streams at the Tier-0 site, to future bulk reprocessing of data at Tier-1 Grid sites

Table 1 Number of files, comment lines, and code lines in the Athena software repository, divided by programming language. This represents a specific snapshot of the repository, which is continuously evolving

Language	Files	Comment	Code
C++	17,273	457,373	2,608,231
PYTHON	9478	211,655	1,009,088
C/C++ Header	20,475	469,490	843,679
Custom Configuration	307	0	368,828
XML	954	12,800	204,169
Shell	1243	12,283	48,782
CMAKE / make	2070	11,021	35,751
Fortran	166	7674	24,024
Web (HTML, CSS, PHP)	44	289	7085
CUDA	28	648	5445
Other	171	3235	24,027
Total	52,191	1,186,288	5,178,472

3.1 Core software

Athena is based on the Gaudi project [48]. Gaudi itself is developed jointly with LHCb and other experiments. An Athena application consists of dynamically-loadable *components*, which implement the concepts of Algorithms, Services, and Tools; see Fig. 3. Algorithms process data which reside in a shared *event store*; they read objects (identified by type and a string key) from the store and write new objects back to the store. Ideally, an Algorithm itself does not contain any event data and does not communicate with any other Algorithm except via the event store (exceptions are mostly special-purpose algorithms that have not been fully migrated to multithreading). Services are objects used by multiple other components; examples include the event store itself, error logging, and random number generation. Tools serve as helpers for other components. They may be uniquely owned by Algorithms, Services, or other Tools. All three component types may declare *Properties* to be initialized during job configuration in a uniform manner (see also Sect. 3.2).

The framework may run in one of four modes:

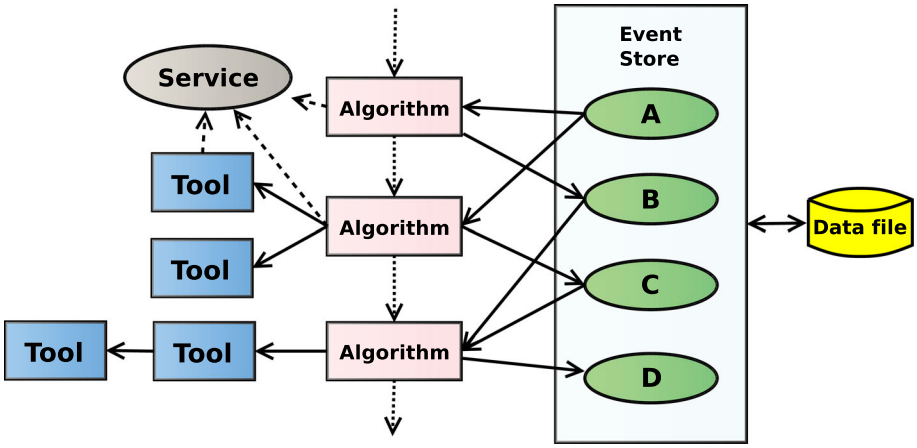


Fig. 3 General structure of an Athena application. The dotted line indicates the application control flow. The solid lines on the right indicate data flow; on the left they indicate ownership. Dashed lines indicate a non-owning reference between components

- A serial mode (Athena), in which one event is processed at a time and the order in which Algorithms run is fixed during job configuration;
- A multi-process mode (AthenaMP), in which several worker jobs are forked after initialization is completed, and memory is shared among the workers as long as it is not modified, significantly reducing the total memory footprint of the job while achieving throughput very similar to multiple serial jobs [49];
- A multithreaded mode (AthenaMT), in which multiple events may be processed concurrently and Algorithms may be executed in parallel in separate threads. In this latter mode, there is a separate instance of the event store or *slot* for each event that may be processed concurrently. Parallelism may be both *intra-event*, in which Algorithms processing data from the same event run in parallel, and *inter-event*, in which Algorithms process data from different events in parallel. This significantly reduces memory requirements beyond what is possible with AthenaMP, while providing comparable throughput for computationally-intensive workloads [50]; or
- A hybrid multi-process/multithreaded mode, which is used in the HLT in particular to maximize throughput for the memory capacity of the HLT farm configuration [18]. The choice of the number of threads per worker process is configurable.

For context, over a recent six-month period, about 18% of the CPU on the Grid was consumed by serial (single-core) jobs, of which 40% was user jobs and 50% was event generation. About 12% of CPU was consumed by multi-process jobs, and the remaining 70% was consumed by multi-threaded jobs. On the Grid during the same period, about 60% of job slots were 8-core, 13% were 64-core, 18% were single-core, and the remainder were of various other breadths; the number of cores per slot is largely determined by the site administrators.

Each Algorithm declares the set of data objects it reads from and writes to the event store, allowing the construction of a dependency graph for Algorithms during the initialization of the application. In addition to this 'data flow' information, 'control flow' rules may be used to explicitly state the sequence in which some collection of Algorithms must run. This can be used to implement filtering, allowing event processing to stop early if some condition is not satisfied. The 'Avalanche' scheduler [51,52] looks for Algorithms that have all their inputs available and control flow rules satisfied and queues them for execution, using the task facility of the Intel Threading Building Blocks (TBB) [53] library. The scheduler is designed to have a low latency and to scale well to many threads.

The data dependencies of Algorithms and Tools are declared via their access mechanism to the data [54]. To read or write an object of type `T` from the event store, an Algorithm declares a Property of one of the special types `SG::ReadHandleKey<T>` or `SG::WriteHandleKey<T>`. The value of the Property is the name of the data object in the event store. During execution of an Algorithm, these key objects may be combined with an `EventContext` object (see below) to form smart pointers that are used to access the data objects. These declarations then form the input that the scheduler uses to build the data flow graph. Tools may declare such dependencies as well as Algorithms; the Tool dependencies are then propagated up to the owning Algorithm.

To ease the adoption of multithreading, a given Algorithm instance by default cannot be scheduled simultaneously in more than one thread. When an Algorithm is migrated to run multithreaded, it is preferably declared *reentrant*. In that case, the Algorithm instance may execute concurrently in multiple threads; accordingly, most Algorithm execute methods are declared `const` to reduce the likelihood of multithreading-related issues. In a typical reconstruction job, only a few percent of the roughly 1000 scheduled algorithms are not reentrant. Some of the Algorithms that cannot be made reentrant are declared as *clonable*. In this case, multiple instances of the Algorithm can be made, and distinct instances may

be scheduled in multiple threads. Most Algorithms that are declared clonable are related to detector simulation.

A specific event among the ones currently being processed is identified via an instance of the `EventContext` class. This contains the identifying numbers of the event (a 'run' and 'event' number), the number of the event slot used by this event, and a direct reference to the event store implementing that slot. When the scheduler starts execution of an Algorithm, it passes to it the `EventContext` for the event to be processed. The `EventContext` is then used to access the event store. While it is preferred to pass the `EventContext` explicitly to functions that need it, the current `EventContext` is also stored in a thread-local global variable to ease the integration of older code.

Once an object is recorded in the event store, it should not be modified; besides the possibility of data races, this can lead to circular dependencies in the data flow graph that the scheduler cannot resolve. However, it is common to need to remake an object already existing in an input file from other data in the file (e.g. if a bug is discovered in a reconstructed physics object that can be re-computed using other information in the input file). To allow for this, any objects in the input file with names that match any declared `WriteHandleKeys` will be ignored, rather than read. One may also need to make a revision to an object existing in an input file, for example to correct a problem from an earlier version of the software. To support this, objects being read may be renamed. For example, an object called '`Electrons`' may be renamed to '`Electrons_in`' on input. An Algorithm can then read '`Electrons_in`', make a copy while applying a correction and save the copy as '`Electrons`'. Later algorithms that use '`Electrons`' will then retrieve the correct version without having to be modified.

Besides event data, reconstruction Algorithms may depend on *conditions* data [55], like calibrations, alignment, or maps of problematic detector readout channels. These data often require some special handling, as described in Sect. 3.3.

Persistency of data objects within the ATLAS offline software [56] uses ROOT I/O and was built on top of the LCG POOL [57] framework. POOL provided high performance and highly scalable object serialization to self-describing, schema-evolvable, random-access files. However, the intention of serving multiple experiments and use-cases with different software stacks caused the project to grow and be less efficient than desired. After the other experiments abandoned POOL, the software was streamlined and incorporated into the ATLAS repository. This software layer enables ATLAS I/O to support persistent navigation, by providing tokens that contain the storage address of data objects. Given a token for a particular data object, the infrastructure is able to directly access that object, read it and restore its state in the transient store. This mechanism works independent of file boundaries and even storage technologies, in principle allowing ATLAS jobs to navigate to upstream data or data augmentations on demand.[6] The software framework itself does not require overall event data organization in the transient store, but persistifies a lists of data objects managed by a `DataHeader`. This software also controls the placement of data objects in ROOT `TTrees` and `TBranches` and setting properties such as compression, managed by the ATLAS I/O framework.

In multithreaded operation, some special considerations apply to the I/O components [58]. Unlike other Athena Algorithms and Services, which independently process data for a single event, the I/O and storage infrastructure handles compressed buffers containing data for many events (10–1000) at a time. This limits concurrency when reading and writing event data as locking is required to assure a particular I/O buffer is accessed by a single thread only, even as many events may be processed simultaneously. There is one Service for reading event data, one for reading conditions data (see Sect. 3.3), and one for writing event data. When writing multiple streams/files, separate output services are used to gain concurrency. In addition, the column-wise storage of ATLAS data in ROOT [46] (see Sect. 3.4) allows separate branches to be processed by separate input services concurrently. All these services are each individually serialized but may run concurrently with each other.

As a result, I/O is not a bottleneck for most applications; typically I/O is below 5% of the total job time including compression and decompression. Read-ahead and caching of event data is provided by the ROOT `TTreeCache`. To prevent I/O thrashing when multiple events are being processed simultaneously, the maximum virtual `TTree` size is increased to hold trailing I/O buffers. Additional parallelism on writing is gained by using the implicit multithreading mode of ROOT.

Gaudi supports *incidents*, a form of structured callback; a component can at any time raise an incident of some type. These incidents are handled by the Gaudi IncidentSvc that in turn makes callbacks to components that have registered their interest in that particular type of incident. Incident types include starting and ending events, files, and runs. Incidents are problematic for multithreading because they could in principle be asynchronous relative to event processing, and do not respect event boundaries. However, as used in Athena, almost all incidents are raised by the event loop due to discrete state changes. Therefore, rather than having the IncidentSvc make callbacks directly, they are instead made from special Algorithms that run at the beginning and end of event processing. Incidents are now only sent to Services, not Algorithms; these Services may retain data separately for each

[6] In practice, the complexities of navigation to another file that might be at a different site have limited the application of this ability.

active event context. Algorithms may observe the effects of incidents by calling the Services that handle them.

Random number generation can be problematic in a multithreaded environment. Requiring locking or access to thread-local storage for every random number call can add a considerable performance overhead, but the generator state must somehow be protected against concurrent access. Further, to have reproducible results, the sequence of random numbers must be independent of the order in which events are processed or in which Algorithms are scheduled. In Athena, different streams of random numbers, with separate seeds and generators, are distinguished by the name of the Algorithm that uses them. For each type, Athena maintains an array of generator states, one for each event slot. Because no Algorithm can be executing on the same event slot in more than one thread, the generator state retrieved from this array can be used without further synchronization. For improved reproducibility, the generator states are reseeded for each event, based on the event and run numbers.

In addition to the offline reconstruction, the online software running in the ATLAS HLT uses the same multithreaded framework [59]. However, a key additional requirement for the HLT is the ability to limit reconstruction to within a set of geometric *regions of interest* (RoI) in the detector. This is implemented via an *EventView*. It provides the same interface as the event store, but provides only a subset of the detector data, corresponding to the RoI. In this way, Algorithms that access event data can be transparently restricted to the subset of event data provided by an EventView. EventViews are created by specialized Algorithms that fill them with region-specific data and request the scheduler to execute a sequence of Algorithms in the context of each EventView. At the completion of processing, the results from all EventViews are merged and saved to the primary event data store. In addition, the raw detector data are managed by a special thread-safe container such that data for different regions of the detector can be simultaneously unpacked in different threads as is required by the trigger Algorithms.

Although Algorithms execute independently of each other based on their data dependencies and thus can usually proceed without explicit synchronization, some special data structures used by the framework and event data model can be shared between threads and thus require some form of synchronization. This can be done using locks, but this may result in bottlenecks when many threads are used; locking may also involve significant overheads even in the absence of contention. In some cases, there are *lockless* methods for doing synchronization without explicit locking, but these can be quite complicated and involve their own significant overheads. However, many of the structures of interest in the framework are *read-mostly*; that is, reads are frequent and are important for performance, while modifications are infrequent. In this case, one can allow multiple lockless readers along with a single writer, which can be serialized with a lock. This is usually much simpler than the general lockless case while still providing good performance for read-mostly workloads. ATLAS uses several data structures developed using these principles [60] to improve the scalability of core components of the framework. These include a variable-length bitmap, a hash table, and a specialized container that maps from intervals of validity to conditions data objects (see Sect. 3.3), as well as helpers to manage lazy initialization of mutable class members without requiring locking.

3.1.1 Updates for multithreading

The adoption of a fully multithreaded framework, deployed for Run 3, required numerous changes to the code base. For example:

- Event and conditions data have to be accessed via handles; non-thread-safe data caching and back-channels for communication had to be removed.
- Conditions Algorithms must be used rather than caching derived conditions data in Algorithms or Tools (see Sect. 3.3).
- Algorithms that modified data objects existing in the event store had to be redesigned.
- Thread-unfriendly constructs, such as non-`const` static data and `const`-correctness violations, had to be avoided.
- Services must be explicitly thread-safe.
- Reentrant algorithms must avoid the use of mutable instance data.
- Uses of Gaudi incidents needed to be adapted to the multithreaded scheme.
- Normal ('private') Gaudi Tools are owned by another Algorithm, Tool, or Service. However, Gaudi also supports 'public' tools that act as singletons. As these mostly overlap with the functionality of Services, almost all public Tools in Athena were changed to either private Tools or Services.

To assist in finding thread-unfriendly code, ATLAS uses a static code checker [61], implemented as a GCC [62] plugin. As GCC is the primary compiler used by ATLAS, the plugin can be enabled for both the central software builds as well as for individual developers. The plugin can check for problems such as the use of non-`const` static data, `const`-correctness violations (including the use of mutable members), casting away `const`, returning non-`const` pointer members from a `const` member function, and calling non-`const` methods via a pointer member from a `const` member function. Diagnostic messages about such violations may be suppressed on a case-by-case basis by the use of macros that expand to custom C++ attributes. Code authors can tag specific packages

or source files to be checked; in addition, a configuration file may also be used to request checking of all files in a particular source subtree. The checker also enforces several other ATLAS coding rules, such as naming conventions (see also Sect. 6.1.4).

As failures in multithreaded programs can be rare and are often not consistently reproducible, it is essential to have good diagnostics in case of application crashes [63]. On a crash, the currently-executing Algorithm(s) for each slot are printed, and ROOT is used to generate a stack backtrace for each thread. This procedure can, however, fail if the program state is corrupt. Therefore, the handler for fatal signals first executes a 'fast' stack trace in the thread in which the error was detected. This starts by dumping the contents of the machine registers, which is often invaluable in understanding the details of the crash. This is followed by a stack backtrace that is carefully written to avoid any dynamic memory allocation. Addresses in this backtrace are written both in absolute form and as an offset in the containing shared library, allowing for easy location of the code in a disassembly of the shared library. For builds with gcc on x86_64 Linux systems, the system stack unwinder is also modified to allow it in many cases to proceed past corrupt stack frames (as could be caused, for example, by calling a virtual function on a deleted C++ object). A preallocated alternate stack is also declared for the signal handler. These measures ensure that a usable backtrace can be produced most of the time, even in the event of heap or stack corruption. Techniques used for diagnosing some of the more difficult threading-related failures included modifying the memory allocator (TCMALLOC [64]) to include extra checking and a custom VALGRIND [65] checker to watch for particular memory access patterns associated with a crash (e.g. identifying all cases where a specific value was written to memory, irrespective of the address [63]).

3.2 Athena configuration

A typical reconstruction job consists of hundreds of Algorithms, Services, and Tools, all of which must be properly and consistently configured in order for the job to run. As there can be complex dependencies between the various components, this is done using PYTHON 3.

For each component to be configured, a corresponding PYTHON object is created (a '`Configurable`') containing the values of the Properties for that component. The PYTHON `Configurable` classes are automatically generated during the software build based on the components and their Properties declared in the C++ code. Sets of components representing a consistent configuration are collected in an object called a `Component Accumulator` [66].

PYTHON functions set Properties of components and arrange their dependencies as needed. These functions take *Configuration Flags* as input and return `Component Accumulator` objects. The Configuration Flags describe global properties, like whether the input is simulated data or detector data, or if cosmics or collisions are to be processed. Generally, flags are used to ensure the configuration of individual components is done consistently for a particular situation. This approach allows the same PYTHON program ('RecoSteering') to process inputs from proton–proton or heavy-ion collisions, simulation, cosmic rays, and so forth.

The flags have auto-configuration capabilities: they can set themselves automatically based on information found in the input file. For this purpose, the (first) input file is opened and inspected at the configuration stage. For data reconstruction jobs, the run number and time stamp of the first event are used to query the conditions database to determine the appropriate data-taking conditions and configure the reconstruction job accordingly.

For each piece of a job (e.g. calorimeter clustering in reconstruction), there is a PYTHON function taking configuration flags and producing a `Component Accumulator`. With minimal additions, these small units are standalone-runnable as long as their inputs (in the case of calorimeter clustering, the calorimeter cells) are in the input file. This design allows unit-testing of individual components or relatively small sets of components.

The `Component Accumulator` objects created by the configuration functions can be merged together to assemble more complex jobs. In this merging process, duplicate components are reconciled into one instance, with an error reported if they are configured inconsistently. The process is recursive, so that each component can call functions to add any of its dependencies, add the component being configured, merge them all together, and return the complete `Component Accumulator` object. This process allows the configuration of the full job (e.g. reconstruction or simulation) to be built in a modular way. The PYTHON program producing the configuration of the full reconstruction calls functions configuring subsets (for example jet-finding), passing the configuration flags as a function parameter and merges the resulting `Component Accumulator` objects.

Once the final configuration is built, it can be used directly to create the C++ components for the job and run the application. Alternatively, it can be saved to a PYTHON pickle file for later use.

This `Component Accumulator`-based approach was developed during LS2 and put into production for the 2023 data-taking year. The configuration system used up to 2022 was also PYTHON-based but was not as modular, leading to difficulties in maintenance, extension, simplification, and debugging. The old system relied on fragments of PYTHON code, called *job options*, which could be stitched together to drive the configuration of an Athena job. This type of configuration persists in event generation (see Sect. 4.1), where hundreds of thousands of individual generator configurations

are defined in short PYTHON snippets that are executed as a part of the configuration of a job. Many of these snippets are programmatically generated. Each snippet corresponds to a separate physics process; these are described further in Sect. 4.1. The migration of the myriad configurations of event generators to use the `Component Accumulator` is ongoing.

For standard workflows like detector simulation or reconstruction, rather than writing a `Component Accumulator`-based PYTHON script for each job, *Job Transforms* are used to provide a convenient command-line configuration. The input type and output type must be specified on the command line, and almost all other settings are optional. The job transform then determines what software steps are to be run based on a graph from input to output file type and configures the job by passing command-line parameters to the appropriate `Component Accumulator` functions. In the production system (see Sect. 7.3), all jobs are run via these job transforms, so that only the command-line settings need to be stored to fully reproduce the configuration of the job. These production configurations are stored in the AMI metadata system (see Sect. 6.3). The job transform also includes some convenience features, including running a monitoring program that records memory and I/O usage of the job, running input and output file validation, automatic log file parsing (e.g. to search for indications of errors), and producing a job summary report that can be easily parsed in the production system.

3.3 Conditions data handling

Conditions data are valid over some range of events or time, called an *Interval of Validity* (IoV). Conditions data are presented to Athena algorithms in the form of *Conditions Objects*. In Athena, two types of Conditions Objects are distinguished: *Raw* and *Derived*. Raw objects are constructed using the data retrieved from the ATLAS COOL [57,67] Conditions Database by a specialized Athena service called `IOVDbSvc`. Each of these objects corresponds to one COOL *Folder* in the Conditions Database. Often it is necessary to apply some transformations to conditions data. The objects that are created by such transformations are called Derived Conditions Objects. Derived objects are constructed by taking one or several other conditions objects – either raw or derived – as input. Athena supports conditions objects of arbitrary type, although in most cases conditions objects are represented as a collection of key-value pairs where the keys are strings and the values are simple C++ types or vectors of them.

Since the multithreaded framework may be concurrently processing multiple events, it must be able to manage having potentially several versions of a conditions object active at any one time. To satisfy this requirement, conditions data are kept in a separate transient *conditions store* analogous to the event store; however, objects recorded in this store are containers (i.e. *Conditions Containers*) that can hold multiple versions of a conditions object of the same type. Elements in each Conditions Container are indexed by their IoV. IoVs within a single Conditions Container are non-overlapping, and hence one can uniquely identify an object within the container by providing an input time or (luminosity block, run number) pair.

Condition Objects are inserted into Condition Containers by *Conditions Algorithms*, which are a specialised set of Athena Algorithms that operate on conditions data. All COOL folders needed for a given Athena job are registered at the configuration stage with a special Conditions Algorithm called `CondInputLoader`. This algorithm is executed at the start of processing an event. It loops over all COOL folders registered to it and for each of them checks that the corresponding Condition Container has a conditions object that is valid for the given event. If this is not the case for some containers, then new raw conditions objects are retrieved from the `IOVDbSvc` (which means either fetching new data from the Conditions Database or retrieving it from the `IOVDbFolder` cache) and inserted into the container. After that, the framework schedules the execution of those Conditions Algorithms that are responsible for creating the derived conditions objects that depend on the newly constructed raw objects, using the same data flow mechanism as in regular Algorithms.

Algorithms, both regular and conditions, and the Tools they use access objects in the conditions store in a similar manner to that for event data. A Property of the `SG::ReadCondHandleKey<T>` declares a dependency on the conditions object, and may be used to initialize a smart pointer that can use the event identification information in the `EventContext` to look up the proper version of the Conditions Object in the store. Similarly, `SG::WriteCondHandleKey<T>` is used to record new objects in the conditions store that are created by a Conditions Algorithm. This provides the framework with the dependency information needed to ensure that Conditions Algorithms execute before any downstream Algorithms that require the data that they produce.

Since Conditions Algorithms can only insert new elements into Conditions Containers, the size of the conditions store grows as the job progresses, and this may result in a significant increase in the job's memory footprint over time. However, not all container elements are required at any one time, only the ones that are valid for the events that are currently being processed. To optimise the memory usage by the conditions store, a garbage collection mechanism removes objects from conditions containers when they are no longer needed [55]. This mechanism takes into consideration the fact that events are not necessarily guaranteed to be processed

in the same order that they were taken, and tries to avoid the need for multiple instantiation of the same conditions object during the job.

3.4 Event data model

The structure of ATLAS data is defined at various stages along the processing chain, which is discussed in detail in Sect. 4. The scale of the ATLAS experiment, the collaboration, and the data it collects and produces, is such that common data objects and interfaces are crucial to ensure maintainability and internal consistency. The ATLAS EDM is a collection of interfaces, classes and types that combine to provide a representation of an ATLAS event. It provides a commonality across detector sub-systems, allowing common tools to be factored out and shared. It also permits the use of common software between online and offline data processing environments.

The object stores used by Athena, such as the event store and the conditions store, are implemented by instances of the Service `StoreGateSvc`, which provides a type-safe mapping of string-based identifiers to arbitrary C++ objects. For event data, containers of objects are usually represented by the type `DataVector<T>` [68], which is much like `std::vector<T*>`, but with several additions:

Optional ownership A `DataVector` may own the elements to which it points, which are then deleted when they are erased from the vector. An optional argument to the `DataVector` constructor is used to control whether or not a given `DataVector` owns its elements. A `DataVector` that does not own its elements is sometimes called a *view vector*, as it may be used to create a 'view' of elements from other `DataVector` instances.

Container covariance In these declarations,

```
class FourVector {};
class Particle : public
            FourVector {};
DATAVECTOR_BASE
  (Particle, FourVector);
```

the use of the `DATAVECTOR_BASE` macro causes the class `DataVector<Particle>` to derive from `DataVector<FourVector>`. This makes it possible to write, for example, a generic algorithm operating on a container of `FourVector` objects. The event store is also aware of this, so that objects of type `DataVector<Particle>` may be retrieved as type `DataVector<FourVector>`.

Auxiliary data Named data of arbitrary type can be attached to elements of a `DataVector`. These data are stored as contiguous blocks of memory (resulting in improved locality of reference) and are accessed via an abstract interface of a separate *auxiliary store* object.

The design of the auxiliary data is summarised in Fig. 4. A `DataVector` object contains pointers to its elements, each of which contains a pointer back to the container and its index within the container. A `DataVector` also has a pointer to an object implementing the abstract interface `IAuxStore`. This auxiliary store object manages the variables, which are internally identified by small integers and which must be stored in a contiguous block of memory (such as a `std::vector`). When an auxiliary variable is accessed for a given element of the container, the `DataVector` retrieves the vector for that variable from the store, saving it in a cache indexed by the variable's identifier. The element's index is then used to find the proper entry in the variable vector.

For objects that are intended to form the input to analysis, almost all the object data are stored as auxiliary data rather than as class members. These are sometimes referred to as *analysis data objects*, or *xAOD* objects. Other object types, which usually represent intermediate or more detailed results of the reconstruction, generally do not use auxiliary data.

For a given xAOD object type, for example `xAOD::Electron`, the type `DataVector< xAOD::Electron>` is aliased to `xAOD::Electron Container`. An additional class `xAOD::ElectronAuxContainer` then implements the `IAuxStore` interface and contains the 'static' data always associated with an `xAOD::Electron`. Its members are `std::vector` instances, one for each variable. An `AuxContainer` class can forward requests for variables that it does not contain to another `IAuxStore` instance. In this case, this is usually an object of type `IAuxStoreInternal`, which manages a dynamic map of variables. Thus, an xAOD object has a certain set of variables that it should always contain, but additional variables, called *decorations*, may be added dynamically. This design is shown in Fig. 5. Like the objects themselves, the scheduler knows about decorations read and written by Algorithms and can take that into account in scheduling decisions. The auxiliary store object is saved separately in the event store: for an object named `Electrons`, the auxiliary store object is named `ElectronsAux`. The four-momenta are stored in the objects, while information about identification and classification is included in the auxiliary store. This design allows flexibility in the data that is provided to analyses, letting users select exactly those variables they require, thereby reducing disk space and confusion at the cost of some modest CPU overhead. The software workflow steps

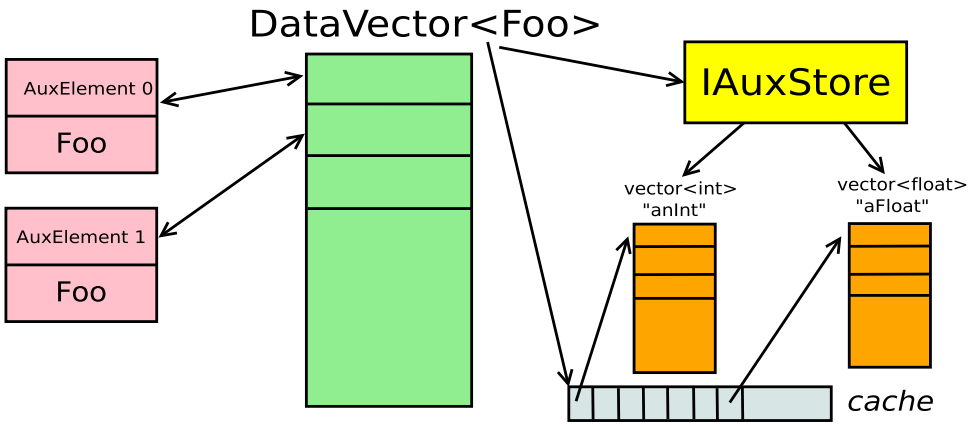


Fig. 4 A `DataVector<Foo>` contains pointers to individual `Foo` objects, which derive from `AuxElement`. Each `Foo` object contains a pointer back to the `DataVector` and its index within the vector. The auxiliary data are stored in a separate object that derives from `IAuxStore`; the `DataVector` has a pointer to this auxiliary store object. The auxiliary store contains contiguous blocks of memory for each available variable, here '`anInt`' and '`aFloat`'. These are usually managed using `std::vector` but this is not required by the design. For efficiency, the `DataVector` maintains a cache array of pointers to the start of the block for each variable; this array is indexed by the integer identifiers for the variables

preceding the analysis stage are much more standard and programmatic, and this flexibility is not normally needed.

A key feature of this design is that the data in the auxiliary store is accessed via an abstract interface. This feature allows using different implementations of the auxiliary store without having to make changes to the `DataVector` class. For example:

- Rather than having separate static and dynamic stores, an xAOD object can transparently use only a dynamic store, making all variables dynamic. This is useful when xAOD data are used as the input for analysis.
- Data objects produced during data-taking by the HLT can use an auxiliary store implementation that is specialised for storage in RAW data files.
- When objects are being read, their dynamic variables are managed by an auxiliary store implementation that defers the actual reading of the data until it is used for the first time.
- This mechanism is also used to implement *shallow copies*, which allow data to be shared between multiple objects. A shallow copy of a `DataVector` will produce a new `DataVector` that has an auxiliary store of type `ShallowAuxContainer`, which maintains a reference to the original store. Variables that are written or modified are managed by the `ShallowAuxContainer` using a copy-on-write mechanism, while attempts to read variables not in the `ShallowAuxContainer` are forwarded to the original store. This is useful for cases where one wants to make a copy of an object and change a few variables, allowing storage to be shared for the unchanged variables.

To represent references between objects in the event store, special link classes are used. `DataLink<Obj>` represents a link to an object of type `Obj` in the event store (e.g. a charged particle track might link to its component hits). The object is identified by an integer hash of the object name and type; this integer is what is written when a `DataLink` is saved. The mapping between hashes and (name, type) pairs is saved in the file metadata. Similarly, `ElementLink< DataVector<T> >` represents a reference to a particular element in a `DataVector`. It consists of a hash as for `DataLink` identifying the container, along with the index of the desired element in the vector.

When a container is being written to persistent storage, it is possible to select specific elements to be written. This process is called *thinning*[7] (see also Sect. 4.5.1). For each container that is to be thinned, there should be an Algorithm that writes to the event store a special object containing a bit map of the entries in the container that should be written. The name of this object encodes both the name of the container to be written and the output file to which it is to be written. When a container is written, the I/O code will test for the presence of this thinning object and implement thinning if it exists. The transient version of the container in the event store is not itself modified by thinning. It is also possible to write only a select subset of decorations, which is a process known as *slimming*.

The I/O process works differently for xAOD and non-xAOD objects. For a non-xAOD transient object of type `Obj`, there is a corresponding *persistent* class of type `Obj_p1`. In

[7] While thinning can in principle be applied to any container, in practice it has to date only been needed for and applied to `DataVector` containers.

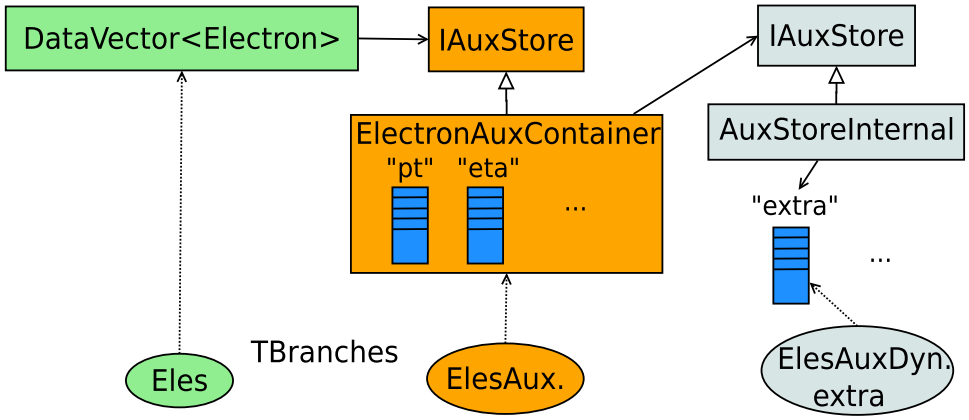


Fig. 5 Illustration of static and dynamic stores used for xAOD objects. In this figure, an object representing a container of electrons of type `DataVector<Electron>` is stored in the event data store with name '`Eles`', which is also the name used for the ROOT `TBranch` when these objects are saved to ROOT. This container references a static auxiliary store object of type `ElectronAuxContainer` (deriving from the abstract interface `IAuxStore`). This object has members `pt`, `eta`, and so forth that are usually `std::vector` instances holding the actual data. This static auxiliary store is also saved in the event data store with name '`ElesAux.`', which is again the name of the corresponding ROOT `TBranch` into which the `ElectronAuxContainer` object is saved. The static auxiliary store can also own a reference to a dynamic auxiliary store, here of type `AuxStoreInternal`, also deriving from `IAuxStore`. The dynamic store holds vectors representing additional variables that are attached to the container. In this example, there is an additional variable (a decoration) named '`extra`'. When written to ROOT, this appears as an additional `TBranch` named '`ElesAuxDyn.extra`'. Solid lines with open arrows indicate inheritance relationships. Solid lines show references between objects. Dotted lines indicate the mapping between names and objects in the event store

the case that the contents of the class change beyond what can be handled by ROOT's automatic schema evolution, additional versions `Obj_p2`, `Obj_p3`, and so on may be defined. Classes that rely on polymorphism use a more complicated scheme. When an object is to be written, a specialised converter class copies data from the `Obj` instance to the `Obj_pN` instance (where N is the latest version). During this copy, thinning requests are applied. If `Obj` is a container that is being thinned, the requested elements are omitted, and if `Obj` contains `ElementLinks` to a container being thinned, then the indices of those links are adjusted to preserve the references. The resulting persistent object is then written as a branch in a ROOT `TTree` representing the event data. When an object is read, the version of the persistent class present in the input file is used to select the proper converter class to copy data back from the persistent object to the transient object. This approach allows any version of Athena to read many old data formats, while it writes only the latest.

For xAOD objects, the `DataVector<xAOD::Obj>` itself is saved as if it were a `std::vector<xAOD::Obj>` with a custom ROOT collection proxy. Since for most xAOD objects the elements themselves do not contain any data, this effectively just records the length of the vector. As discussed above, object data are stored in the separate auxiliary store objects. These are versioned: for an object of type `DataVector<xAOD::Obj>` (aliased to `xAOD::ObjContainer`) there are auxiliary store classes `xAOD::ObjAuxContainer_v1`, `xAOD::ObjAuxContainer_v2`, and so on, with the most recent one aliased as `xAOD::ObjAuxContainer`. This object is saved separately to the ROOT event `TTree` as a single object. Dynamic variables are then saved to separate ROOT `TBranch` objects, one branch per object. Although there is no separate persistent form for xAOD objects, copies are still made of the objects before giving them to ROOT to be written. This design allows the implementation of thinning, as well as transformations such as making all variables dynamic. On input, if an older version of the auxiliary store object is found in the input file, a converter class is used to convert the data to an instance of the current version of the store.

3.5 Detector description

The ATLAS detector is described in Sect. 2.2. The software description of the detector underlies much of ATLAS software and the workflows that depend on it. In simulation, an accurate description is required to model the interactions between particles and detector components, both active (i.e. instrumented for readout) and inactive. In both the simulation and reconstruction, the detector description is used to translate detector hits into various relevant coordinate systems. Since this translation depends upon the time-dependent alignment of detector elements, such as silicon sensors, muon chambers and the like, the software description of the detector needs to follow conditions data, in addition to static data describing the default position of such elements. The implementation of the detector description for Runs 1–3 is pre-

sented in Sect. 3.5.1, and new tools developed to support the detector description in Run 4 are presented in Sect. 3.5.2.

3.5.1 Detector description in Runs 1, 2, and 3

For about two decades the software description of the ATLAS detector has relied on the GeoModel class library [69]. In brief, this library uses a scene-graph approach, building a hierarchical tree representing the geometry that permits a compact in-memory description of the detector geometry. Users construct a directed acyclic graph consisting of:

- Physical Volumes (volumes with a size and shape),
- Transformations (translations and rotations to place the physical volumes),
- Name tags (which assign character strings to physical volumes), and
- Identifier tags (which associate integers to physical volumes),

and special graph nodes that are built to allow repeated, systematic placement of multiple volumes following a set of rules:

- Serial Transformers (to programmatically translate and rotate a series of volumes),
- Serial Denominators (to programmatically name volumes), and
- Serial Identifiers (to programmatically identify volumes, e.g. by index number).

The GeoModel class library relies on polymorphism for flexibility and extensibility: other geometrical objects or properties thereof can be added according to need.

Serial Transformers allow the embedding of almost-arbitrary recipes for generating transformations to be encoded within the tree. Serial Denominators and Serial Identifiers specify policies for naming and identifying physical volumes. `FullPhysicalVolumes` are specializations of Physical Volumes used for active detector elements; they cache the position of the detector element in the world coordinate system, after computing it from the sequence of cascading local transformations. Alignable Transformations are specializations of Transformations that can be adjusted or tweaked according to evolving alignment conditions.

During its first two decades of use, very few modifications to the GeoModel were required. The most intricate change was the adaptation of the Alignable Transforms to the multithreading environment of today's offline software, which implies that detector elements simultaneously cache *multiple* global-to-local transformations, to enable the concurrent processing of multiple events that may have different alignments.

Primary numbers for the description of the ATLAS detector are stored as tabular data in a relational database (ORACLE®). This database consists of more than 1000 database tables to describe all geometric layouts used over the lifetime of the experiment, including both tables of geometry information populated by experts using a variety of tools and scripts and tables used for versioning of the geometries. A small fraction of these tables are typically updated when a new layout is constructed. Specific ranges in the data tables are assigned alphanumeric tags and the tags are finally collected into one overall tag for the whole detector configuration. These tags may be developed, locked and placed into production, and eventually obsoleted by indicating the last Athena release that supports them.[8] The contents of the database can be easily consulted through a dedicated web-based browser, and is replicated in a lightweight SQLite [70] file with a size of around 75 MB.

The same database holds tables corresponding to materials for all of the detector elements. A precise elemental composition of some elements is quite important for the simulation of radiation in the detector. For example, including layers of boronated polyethylene is critical to understanding low-energy neutron flux. The understanding of the detector is sufficiently precise that even element densities are important. For example, the evaporative cooling system in the inner detector produces visibly different numbers of hadronic interactions in different detector regions, owing to the transition from denser liquid-coolant to less-dense gas. In some cases, the material of the detector changes over the course of data taking; this is the case for the gas in some modules of the TRT. Which modules contain which gas is therefore encoded in the conditions database.

The detector geometry system must not only provide a best-knowledge geometry for each year of data taking, but must also support several alternative configurations and geometries that are important to the collaboration. For example, to support cosmic-ray data taking and MC simulation, layouts of the detector that include the concrete cavern, steel gangways and other infrastructure around the detector, and even the bedrock above the detector are supported. Several of the forward detector systems can be described using GeoModel, but they are not normally included when building a layout of the detector for simulation, for example. During commissioning of the detector, some detector elements might be significantly displaced from their nominal positions (e.g. the calorimeter endcap might be displaced to allow access to the inner detector). These layouts have also been simulated

[8] This obsoletion mechanism allows the removal of obsolete code and a clear sign of what detector layouts and geometries must be supported by all systems.

to provide early understanding of cosmic-ray data for detector groups. Elements of the ATLAS detector have also been placed in many different test beams, some of which are supported by the standard detector description system. These have proven useful to the GEANT4 Collaboration [38–40] towards the understanding of the tuning of physics models, and have therefore been exported in XML format for their use outside of ATLAS.

This detector description provides the reference geometry for the detector. However, for several applications it must be translated into a different format. For the detector simulation (see Sect. 4.2), the geometry is loaded into GEANT4 during the job initialization. This also means that the alignment of the detector during a simulation job does not change. For charged particle tracking (see Sect. 4.4.2) and fast track simulation (see Sect. 4.2.2), a simplified *tracking geometry* is created that maps the geometry into simplified concentric cylindrical shells for much faster navigation and transportation.

Because of its importance to the work of the experiment, constant efforts are underway to improve the geometry description of the detector. Some of these, like efforts around radiation simulation (see Ref. [2]), require better understanding of materials and details like the heavy metals inside on-detector electronics. In other cases, this includes going back to design drawings, discussing with engineers, or examining pictures taken during installation to build CAD representations of the detector that can then be compared with the existing geometry, or to simply identify missing material in the detector description and add it directly. This is both difficult and important in some regions of the detector like cable trays, where reality often deviates from design.

One final, critical element of the detector description is the magnetic field map. A link is provided in the conditions database to select the external field map file to be loaded during an Athena job. The map itself is built from a combination of probe measurements and simulation of the currents in the solenoid and toroidal magnets, including the distortion effects from material in the cavern. These maps are generated for several standard magnet configurations (e.g. nominal field, solenoid off, and toroidal magnets off). The field strength is scaled based on the currents recorded during operation. The field value is based on interpolation between values at points provided within the map; a thread-local cache holds the values at the eight map points closest to the position being queried, such that a re-evaluation of the field value nearby (as occurs frequently during detector simulation) is fast.

3.5.2 Detector description in Run 4

The GeoModel class library will continue to be used in Run 4. However, new tools were developed since about 2019 to greatly simplify the workflow of development and maintenance of the ATLAS detector model on the one hand, and its incorporation into offline workflows on the other. The 'GeoModel Toolkit' [71] constitutes an integrated development environment for detector modelling. It consists of:

- The same GeoModel class library as used today in ATLAS;
- Mechanisms to save and restore detector geometries to data files;
- A mechanism for dynamically loading plugins that build geometry;
- GMEX, the 'geometry explorer' for fast visual debugging of geometry;
- GMCAT, a tool for assembling geometry files from plugins or file input;
- GMSTATISTICS, for performance benchmarking;
- GMCLASH, GEANT4-based detection of geometry overlaps (clashes) which cause unpredictable behaviour during simulation;
- GMGEANTINO, a command line tool for generation of Geantino[9] maps;
- Tools for gdml-to-geomodel conversion (and vice versa); and
- FULLSIMLIGHT, a lightweight GEANT4-based simulator and FSL, its graphical user interface.

The introduction of these tools is the result of lessons learned in Runs 1, 2 and 3, as well as the availability of newer technologies. The main implication for the detector description workflow is that the geometry description can be implemented in a lightweight, portable, and modular environment, taking primary numbers from XML files in a git-managed database, following which an SQLite file is created and fed to ATLAS simulation, reconstruction, and other offline tasks. A transition to these new tools, and the new workflow, is now underway.

Already during LHC Run 3, the GeoModel toolkit is built and compiled outside of ATLAS and linked as an external package in Athena-based workflows (see Sect. 6.1.1). It does not depend on ATLAS and is available for use in other experiments as well.

3.6 Machine learning and software infrastructure

Machine learning (ML) is used in a wide range of applications in the ATLAS Collaboration, including physics analysis, simulation and physics object identification in the reconstruction and trigger system. Models trained for classification and regression, as well as generative models, are core

[9] A Geantino is a fictitious particle implemented in GEANT4 that has transportation processes but no physics properties. It is often used to integrate the material traversed by a particle traveling in a detector.

components of the software and analysis chain. This section describes the infrastructure and tools for ML used within the collaboration.

Classification models are the most commonly employed. For physics analyses, these models are used to categorise events into signal-like or background-like samples, efficiently distinguishing events of interest. Often multivariate event classifiers based on boosted decision trees (BDTs) are used [72–85]. BDTs have also been used for regression [86]. Other analyses have explored the use of neural networks (NNs) for classification, for example in top-quark physics analyses [87–94] to separate signal and background processes. Classification models are also used extensively to accurately identify physics objects, with applications in the identification of hadronically-decaying τ-leptons [95], boosted jets [96] and heavy-flavour jets [32]. More recently, recurrent neural networks (RNNs) [97] are exploited for their ability to harness sequential characteristics to take into account correlations between track impact parameters, thereby improving physics performance for jet flavour tagging [98] and τ-lepton reconstruction [99] (see Sect. 4.4).

Artificial neural networks have also been used for lower-level reconstruction tasks. For example, to identify merged clusters in the pixel detector, which are created by multiple charged particles traversing the silicon, a clustering algorithm is implemented to identify and split the clusters and provide improved position estimates for the individual cluster pieces [100].

Generative models have predominantly been explored for use in simulation tasks. Originally developed for image generation, generative adversarial networks (GANs) [101] have shown particular promise for calorimeter simulation [102, 103], where energy depositions from particles interacting with a calorimeter can be represented as images. GANs have recently been integrated into the ATLAS fast simulation chain (see Sect. 4.2.2 and Ref. [42]), where they are used to approximate the calorimeter response to pions.

In the past, most simple models used in the collaboration were trained with and supported by the TMVA ROOT library [104]. With the rise of more modern and complex ML methods, in particular with the advent of deep learning, the use of other libraries to train models has become more prevalent. The most popular libraries for training models are TensorFlow [105], using the Keras [106] frontend, and PyTorch [107]. However, XGBoost [108], LightGBM [109] and other frameworks are still commonly used for the training of BDTs. Many neural networks used in physics analysis rely on the NeuroBayes package [110]. Support for inference from many modern ML frameworks is provided in the ATLAS software environment. To avoid maintaining and supporting multiple competing ML toolkits for inference, the ATLAS Collaboration has moved to using general inference frameworks.

The first general purpose inference framework integrated into the ATLAS software environment was LWTNN [111], which uses the boost library [112] as a JSON parser and Eigen [113] for tensor operations. Users can convert their trained NN models into JSON format, defining the weights and layer operations. This is then handled by a lightweight class for running inference on single events. For the inference of BDTs, a custom wrapper was written that converts models trained with TMVA, LightGBM or XGBoost into a common ROOT `TTree` format for inference at runtime. However, as BDTs are mostly used in physics analyses, other libraries are compiled in standalone ntuple processing frameworks to provide inference. Due to the rise of more complex models, which use layers and operations that are not supported by LWTNN, the ONNX Runtime (ORT) library [114] was chosen as a general inference library. Most modern ML libraries can save models in an ORT format, or are supported with conversion tools. These include, but are not limited to, PyTorch, TensorFlow, LightGBM and XGBoost. Using only a single inference library to support many models trained in different frameworks significantly reduces the maintenance load. ORT is observed to provide faster inference than previous implementations and support for batched inference. Although deep NNs are reliant on GPUs for training, inference is performed almost exclusively on CPUs. In addition to supporting inference of modern ML models, training infrastructure is available through WLCG with support for job submissions using PanDA [115], including access to many GPUs (see Sect. 6.1). However, most models used by the collaboration are trained on private resources due to the modest resource requirements and satisfactory availability of institutional GPU resources.

4 Data and Monte Carlo production and processing

There are several steps in the standard ATLAS software workflow, as described in Sect. 2.3. This section goes into detail about each step of the workflow, including improvements that were made for Run 3. Here the focus is on those steps of the workflow that are run centrally, while steps that are normally run by individual users are described in Sect. 8. The forward detector systems are dealt with somewhat separately, often outside of the normal workflows, because they are often used for specific, special purposes. Their treatment is described in Sect. 4.6. The support within the current Run 3 software for future hardware upgrades to the detector is described in Sect. 4.7.

4.1 Event generation

Simulated detector events play a critical role in numerous areas of the ATLAS experiment, from understanding the

detector response to serving as the signal and background estimates for the large range of measurements and searches performed by ATLAS. MC event generators are at the core of these simulated datasets and are the first step needed to produce simulated datasets. All MC sample sets used in data analysis are centrally produced and managed in ATLAS. The first step of MC sample production is the generation of the event record from a theoretical framework implemented in an event generator program. Events produced at this level are written into files called EVNT files. While many of the MC samples are passed through the full detector simulation, digitisation, and reconstruction steps (described in the following sections), some samples with alternative configurations, such as samples with systematic variations of one or more theoretical or phenomenological parameters, are often not.

The ATLAS Collaboration makes use of a wide range of MC event generator programs: POWHEG [116], MADGRAPH5_aMC@NLO [117] (MADGRAPH), SHERPA [118], PYTHIA [119], and HERWIG [120] are used for a wide variety of processes. Some generators, including SHERPA, PYTHIA, and HERWIG, can produce complete events alone, including a calculation of the relevant process (e.g. W-boson production), parton showering, hadronisation, and particle decays. Several generators, particularly those that implement higher-order calculations, only produce final states with a few partons (often referred to as hard-scatter or matrix-element events) and require other programs to produce realistic events with stable particles. The physics details of such further processing can be quite involved (relating to resummation or avoidance of double-counting), but essentially they must all be interfaced to another program, such as PYTHIA or HERWIG, for the parton shower, hadronisation, and particle decays. A few generators run as secondary Athena algorithms after the main generator code, to refine specific aspects of the event generation. The main such generators, often referred to as *afterburners*, are PHOTOS++ [121] for refinement of QED final state radiation (FSR), TAUOLA++ [122] for precise τ-lepton decay modelling, and EVTGEN [123] for refinement of heavy-flavour hadron decays, in particular for B-physics and flavour tagging purposes.

The generators providing the most events and samples in ATLAS are SHERPA, POWHEG, and MADGRAPH, with showering performed using PYTHIA or HERWIG. However, a wide variety of bespoke, limited-use, or customised generators are used for special samples, including:

- A beam-halo generator that reads files provided by the LHC machine group detailing the expected result of the showering of LHC beams through upstream collimators, useful for background estimations.
- A cavern-background generator, used for re-simulation of low-momentum neutrons and photons that permeate the cavern during operation.
- A cosmic-ray generator based on Refs. [124,125], used for the simulation of cosmic rays, including their propagation through the bedrock above the detector.
- EPOS [126], a generator of perturbative QCD events that is most often used to model minimum bias interactions.
- HIJING [127,128], a generator specifically used for the modelling of heavy-ion collisions.
- HTO4L [129,130], a generator for events with the Standard Model Higgs boson decaying into four leptons.
- HYDJET [131], an alternative generator for heavy-ion collisions.
- A PYTHON-based particle gun, useful for single-particle event generation and various tests of detector performance.
- PROFECY4F [132], a generator for events with a Standard Model Higgs boson decaying into four fermions.
- PROTOS [133], a leading-order event generator for some new physics models involving the top quark.
- PYTHIA8B, a modified version of PYTHIA that re-decays heavy-flavour hadrons to produce samples enriched in specific hadrons and decays.
- QGSJET [134], a generator used for alternative modelling of minimum bias interactions.
- STARLIGHT [135], an event generator for ultra-peripheral heavy-ion collisions.
- SUPERCHIC [136], an event generator for exclusive and photon-induced processes.

The reading of events in Les Houches Event Format (LHEF) [137] is also supported as a means to aid the usage of additional generators like GG2VV [138], DYNNLO [139], and CHARYBDIS [140]. Many new physics models are generated using Universal FeynRules Output (UFO) models [141], in particular through integration with MADGRAPH. This plethora of event generators and their capabilities is shown schematically in Table 2, along with the output formats for each step.

Although the generation for the newest (Run 3) campaigns is ongoing, for illustrative purposes the approximate numbers of events and datasets (unique configurations) generated with various event generators thus far in Run 2 is shown in Table 3. Additionally, the number of simulated events with the standard Run 2 configuration is included in the table. In many cases, the number of events per generator includes multiple generator versions that were produced throughout the entire Run 2 period as improved theoretical predictions became available. Among the largest datasets are the single vector boson samples produced using the SHERPA, MADGRAPH and POWHEG event generators. These samples are

Table 2 Event generators supported in Athena and their general capabilities. At top, the output formats used at each step. Les Houches Event Format (LHEF) output is used for the transmission of matrix element events to generators that implement parton showers; this output format is not supported by all event generators

Output format	Les houches Events		HepMC / EVNT	HepMC / EVNT
Generator	Matrix element	Parton shower	Stable particles	Afterburner
BEAM HALO GENERATOR			✓	
CAVERN BACKGROUND GENERATOR			✓	
COSMIC RAY GENERATOR			✓	
EPOS	Only Minimum Bias	✓	✓	
EVTGEN				✓
HERWIG	$2 \rightarrow 2$ LO and NLO	✓	✓	
HIJING	Only Minimum Bias	✓	✓	
HTO4L	✓			
HYDJET	Only Minimum Bias	✓	✓	
MADGRAPH5_AMC@NLO	✓			
PARTICLEGUN			✓	
PHOTOS				✓
POWHEG BOX	✓			
PROPHECY4F	✓			
PROTOS	✓			
PYTHIA	Only $2 \rightarrow 2$	✓	✓	
PYTHIA 8B			✓	
QGSJET	Only $2 \rightarrow 2$ jets	✓	✓	
SHERPA	✓	✓	✓	
STARLIGHT		✓	✓	
SUPERCHIC	✓			
TAUOLA				✓
Other Generators (via LHEF)	✓			
Other Generators (via HepMC)			✓	

Table 3 Number of datasets (with unique configurations) and events (in billions) generated with various generators thus far during the MC simulation campaign of Run 2

Event generator	Datasets	Generated events ($\times 10^9$)	Simulated events ($\times 10^9$)
SHERPA	3887	89.7	27.6
POWHEG	6747	55.7	15.9
MADGRAPH	251,023	52.2	12.5
PYTHIA	6240	13.8	7.5
PYTHIA 8B	422	5.1	2.0
HERWIG	813	4.3	2.4
Others	9851	3.5	0.5
Total	280,935	224.4	68.4

used throughout ATLAS, all the way from detector calibrations to measurements and searches, and as such, large samples are needed across a wide phase-space. As can be seen from Table 3, most generated events are not passed through the detector simulation. These events are primarily used as inputs for low filter efficiency final states, including final states with extra heavy flavour jets or high missing transverse momentum, which are selected from the inclusively generated phase-space. The generator with the most datasets is MADGRAPH, because it is typically used for searches for new physics scanning a model parameter space. While this results in many datasets, the contribution to the total number of events is small compared with the high-rate Standard Model processes described above.

Many event generators can provide estimates of systematic uncertainties arising from variations of the renormalisation and factorisation scales, and variations in the Parton Distribution Function (PDF) sets, via modified event weights. Moreover, some generators such as SHERPA are now capable of providing cross-sections calculated at next-to-leading order (NLO) in the electroweak coupling constant, allowing analyses to assess the impact of NLO electroweak corrections in their phase-space by modifying an event weight [142]. While these extra *on-the-fly weights* incur extra CPU time during event generation, it is small – typically around 10% additional CPU for more than 100 event weights [143] – particularly when compared with generating each variation standalone. Additionally, the generation of systematic variations as event weights within the nominal sample allows these variations to be available in the simulated and reconstructed datasets with no additional CPU cost. They also significantly reduce statistical fluctuations when evaluating systematic uncertainties, because the kinematics of the nominal and varied events are identical. For standard samples, around 200–300 systematic weights are stored per event. For systematic variations that can theoretically be estimated by using on-the-fly weights, the weights are saved; for other variations dedicated samples need to be prepared.

Most of the MC event generators are provided by dedicated teams outside ATLAS. This software is built using the *Layered-Stack* that is used in GENSER (GENerator SERvice) installations (see also Sect. 6.1.1). One of the features of the layered stack is that there cannot be multiple generator versions in one LCG release. Whenever a new version of a generator appears that will be used in ATLAS (an almost weekly occurrence), a new layer is requested, in which the generator version is updated but all other externals remain fixed. This significantly reduces the overhead and build times for providing new generator versions to the collaboration. GENSER provides a new layer for all the platforms supported by a given LCG configuration. After installation and validation by GENSER the layer is included into a nightly build and validated by ATLAS. For a few generators, like POWHEG, builds are maintained by ATLAS directly.

Regular validation of event generation is performed as a part of the ATLAS Release Testing (see Sect. 6.1.3). In these tests, small samples of events are generated for select physics processes. The output of each test is compared automatically to the reference output from the previous test. In the case of a generator version change, a more thorough physics validation based on RIVET [144] analyses is performed [145]. The procedure of layer creation and validation usually takes a week. However, for very urgent installations, it can be shortened to one or two days.

The steering of the MC event generators is passed from the Athena framework to the event generators. The generators are instantiated and configured in job options (see Sect. 3.2). The job options contain relevant information about the physics process and define the generator type and its configurations. Major generator packages contain single job fragments to configure common aspects of the job to ensure consistency across different sample sets. The event generation is invoked by Athena as part of the standard algorithm event loop, and the generated events that are created are converted into HepMC format. In Run 3, the HepMC event record used is HepMC3 [146]. An attractive feature that motivated the migration from HepMC2 to HepMC3 is to enable event numbers with up to 64-bit precision. In HepMC2, event numbers were limited in size to 32 bits, requiring careful workaround solutions for the largest of samples the collaboration needed to produce.

After generation, standardised filter algorithms can be inserted into the Athena algorithm sequence. These are used to select only events that meet certain criteria, e.g. charged leptons, photons or jets with certain transverse momentum (p_{T}) and η properties, certain flavours of W-boson, Z-boson or top-quark decays or other event shape variables. A combination of several filters is also possible. Significant validation of the generated events is run during the Athena job, check-

ing for un-decayed partons, stable particles not known to GEANT4, highly-displaced particle decays (e.g. K^0_L decays), energy imbalance, and other features that might indicate incorrect physics output or create problems in downstream software.

Beyond the nominal workflow described above, events can also be filtered in a two-step approach if an inclusive sample must be split up, for example by the flavour of jets or number of leptons. In this workflow, first an inclusive dataset is generated and saved to disk. Then, a dedicated filtering job is run that takes as input the inclusive dataset, and an output dataset with events that satisfy the event filter are saved to disk. Finally, the output dataset is then passed through detector simulation and subsequent steps.

4.2 Detector simulation

The ATLAS simulation begins from events in EVNT files and simulates the propagation through the material of the ATLAS detector of all generated particles that escape the beam pipe, including any decays and interactions with detector material that may occur. All energy deposits in sensitive volumes of the ATLAS sub-detectors, along with additional truth information about particles created or destroyed during simulation, are written to a file format called HITS.

There are two main simulation approaches employed by ATLAS: 'Full Simulation', discussed in Sect. 4.2.1, and 'Fast Simulation' discussed in Sect. 4.2.2. About 50% of all MC simulations in Run 3 are fast simulations. Section 4.2.1 includes some discussions of the variation optimisations used to speed up 'Full Simulation'. Section 4.2.2 includes a brief discussion of further improvements to the ATLAS fast simulation that were developed for Run 3. Some of the basic concepts applied in the ATLAS simulation are described in Ref. [41]; this section focuses on improvements, and particularly those improvements deployed for Run 3.

Simulated data are divided into *MC Campaigns*, numbered based on the year in which the production began, and with each supporting conditions that match specific data-taking periods. The MC16 campaign supported Run 2 analyses in the older Release 21 version of the ATLAS software, MC20 supports analysis of reprocessed Run 2 data in Release 22, MC21 was the initial Run 3-like MC simulation campaign for testing and early validation, and MC23 supports detailed analysis of Run 3 data. Letters are used to distinguish the modelling of different data-taking years: MC20a corresponds to 2015 and 2016 data, MC20d corresponds to 2017 data, and MC20e corresponds to 2018 data, for example.

4.2.1 *Full simulation*

ATLAS uses the GEANT4 toolkit [38–40] to simulate in detail the interactions of particles with the ATLAS detector. In this section, the GEANT4 setups and optimisation used for Run 3 are described.

GEANT4 version The MC16 and MC20 campaigns (see Sect. 2.1.3) of Run 2 used an ATLAS-patched version of GEANT4, namely `10.1.patch03`. For MC21 and MC23, which are the current MC campaigns in Run 3, the more recent GEANT4 `10.6.patch03` is used. This version includes an improved hadronic physics model and several updates to the electromagnetic (EM) model. The gamma general process (described under 'G4GammaGeneralProcess' below) feature is enabled in this version. A custom stepper (described in Ref. [41]) with a configuration optimised for ATLAS is also enabled, but this was used already since Run 2. A retuning of Birk's law and a recalculation of the sampling fractions[10] were completed for this updated GEANT4 version.

Birks' law tuning Birks' Law [147] describes the relation between the energy deposited by particles and the signal of the calorimeters. A retuning of the parameters if this formalism became necessary after the updates of the physics models in GEANT4 10.6. The initial parameters of Birks' Law used in the LAr and tile calorimeters were taken from GEANT3 [148] at the point when ATLAS switched to using GEANT4. These values were obtained from experimental data assuming no delta-ray emissions. While correct for GEANT3, this is inconsistent with GEANT4 simulation, where delta-ray emissions do occur above reasonable production thresholds. More precisely, it leads to the quenching effect being underestimated, which implies an artificially higher energy response in the simulation.

The parameters were therefore re-tuned such that the ratio of the EM and hadronic response in data and MC simulation matched. This tuning was undertaken for the Tile calorimeter by comparing MC simulation to test beam data using the previously published analysis [149]. For the EM calorimeters, the value of E/p for charged pions measured in low pile-up collision data collected in 2017 was used [35]. The hadronic endcap calorimeter response was found to be unaffected by tuning Birks' Coefficient. No tuning was done for the forward calorimeter due to the lack of available test beam data. The values for Birks' Coefficient used in Run 2 and Run 3 simulation are shown in Table 4.

Simulation improvements Several improvements were made to the configuration of the simulation for Run 3 that improve the physics performance (i.e. the agreement with data) of the MC simulation. While many small improvements were implemented, the two most significant global improvements are described here.

[10] The sampling fraction is the ratio of the total energy deposit in a cell to the energy deposit in the active material. It depends on the calorimeter specifics and therefore varies between layers.

Table 4 Values of kB used in the Birks' law relation $dS/dr = (A \times dE/dr)/(1 + kB \times dE/dr)$ for the sensitive volumes of ATLAS calorimeters in Run 2 and Run 3 simulation

Calorimeter	Run 2 [MeV/(g × cm^2)]	Run 3 [MeV/(g × cm^2)]
LAr EM barrel and endcap	0.0486	0.05832
Tile	0.0130	0.02002

Beam spot modelling The longitudinal beam spot size varies over the course of a run as the LHC attempts to *level* the instantaneous luminosity during the run as the beams are depleted. In Run 3, the beam spot size within ATLAS shrinks from 43 mm to 34 mm during the first part of each run because of this luminosity levelling. This effect was included in the Run 3 MC simulation. Rather than modelling a continuous distribution, four discrete beam spot sizes (in the z direction) are used: 43, 40, 37 and 34 mm. These values are chosen to be representative of the continuous variation in beam spot sizes during the 2022–2023 Run 3 proton–proton collision data-taking period. The first three values represent the luminosity levelling regime. Levelling is stopped for the last part of the run and the beam spot is approximately constant as the instantaneous luminosity falls. The fourth value represents the beam spot size during this final period. In signal process simulation jobs, distinct lumi block ranges are used to mark off groups of events with different beam spot widths and thereby generate the correct fraction of events with each beam spot size. During the merging of HITS files in the production system, events are sorted according to luminosity block so that events with the same beam spot size are grouped in the merged HITS file.

Quasi-stable particle simulation In the MC16 campaign, and before, all particles decayed by the generator were ignored by GEANT4, even those that propagated outside the beam-pipe. If such particles propagate past sensitive detector layers before decaying, then there is the potential for missed energy deposits and hence tracking differences compared with data. This has implications for flavour tagging and τ-lepton reconstruction in particular at high p_{T}. For the MC20, MC21 and MC23 campaigns this was changed to pass such *quasi-stable* particles to GEANT4 along with their predefined decay chains. Missing particle definitions were added to GEANT4 and the ionisation process was added to all charged particles. This means that all charged quasi-stable particles can undergo energy loss, deposit energy, and be affected by the magnetic field.[11] This improves the agreement with data and allows better tuning of flavour tagging and τ-lepton reconstruction algorithms [150]. Quasi-stable particles are a separate category from long-lived exotic particles, which are not decayed by the generator, but which require extensions to the GEANT4 physics list to be propagated and decayed within GEANT4.

Full simulation optimisations The performance optimisation of the GEANT4-based simulation has been a continuous task since before the start of data taking [151]. The GEANT4 Optimisation Task Force was established in September 2020 and is responsible for optimising the performance of the ATLAS GEANT4 simulation software with the mandate of achieving $>$ 30% CPU performance speedup for Run 3 relative to the Run 2 simulation. Several optimisations were implemented, validated and put in production and are described in what follows.

EM range cuts Several physics processes have very high cross sections at low energies (e.g. bremsstrahlung, ionisation, and electron–positron pair production by muons) and it is therefore necessary to implement a production cut so that all particles below the cut are not generated [152]. GEANT4 offers a solution with the *range cuts*, where it is possible to specify a minimum propagation distance (range) for secondary particles. This distance is converted to an energy threshold in each material internally by GEANT4. Below this energy threshold secondary particles are not created and their energy is immediately deposited at the end of the production step. Further, it is possible to specify range cuts for each material–volume pair separately for photons, electrons, positrons, and protons. In the Run 2 simulation, the range cuts were off by default for Compton scattering, conversion and the photoelectric effect, and turning them on with the value of 0.1 mm already used for electron processes results in a significant decrease in the number of secondaries, as shown in Fig. 6 [153]. This decrease in the number of secondaries led to an 8% decrease in simulation time.

Neutron and photon Russian roulette Neutrons and photons take the most CPU time in the simulation of the electromagnetic calorimeters, which are usually the most resource intensive systems to simulate. This is illustrated in Fig. 7, which shows the number of steps in various volumes for several particle species [153]. The idea of the Neutron and Photon Russian Roulette (NRR/PRR) method is to randomly discard particles below an energy threshold and increase the energy deposits of remaining particles accordingly. This strongly reduces the number of secondary particles generated in the showers that are simulated by GEANT4. The final tuning used in production in Run 3 is a 2 MeV threshold with a weight of 10 for neutrons and a 0.5 MeV threshold with a weight of 10 for photons. These settings result in a speed-up of about 10% for $t\bar{t}$ MC events.

G4GammaGeneralProcess The `G4GammaGeneralProcess`, introduced in GEANT4 version 10.6, is a super-

[11] GEANT4 lacks hadronic interaction models for many of these particles, but once they are available, they would naturally be added.

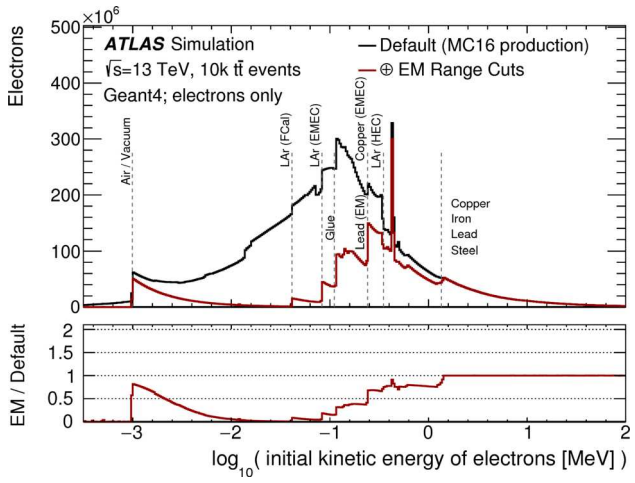


Fig. 6 Distribution of the initial kinetic energy of electrons in the ATLAS GEANT4 simulation. The black curve shows the distribution for the Run 2 setup (MC16 production) and the red curve shows the distribution after the addition of range cuts for electromagnetic GEANT4 processes ('conv', 'phot', and 'compt'). Vertical grey dashed lines indicate range cut values for several key materials and the right-most dashed line indicates an area with multiple range cuts in close proximity for various metals. Figure from Ref. [153]

process that hides all the physics processes involving photons (e.g. Rayleigh scattering, the photoelectric effect, the Compton effect, conversion to electron–positron pairs, and photo-nuclear scattering), providing a single access point to the `G4SteppingManager` that sees only one physics process. As a consequence, only one mean free path must be calculated for a photon, and therefore the number of instructions is reduced at the price of introducing extra physics tables shared between threads. Once validated and included in the production configuration, a speed up of 3% was observed for $t\bar{t}$ MC events.

Woodcock tracking Woodcock Tracking [154] is a tracking optimisation technique suited to be applied to highly segmented detectors where the geometry boundaries rather than the physical interactions limit the simulation steps. It is applied on top of the `G4GammaGeneralProcess`, and the idea is to track particles in a simplified geometry made of the densest material (i.e. without boundaries).

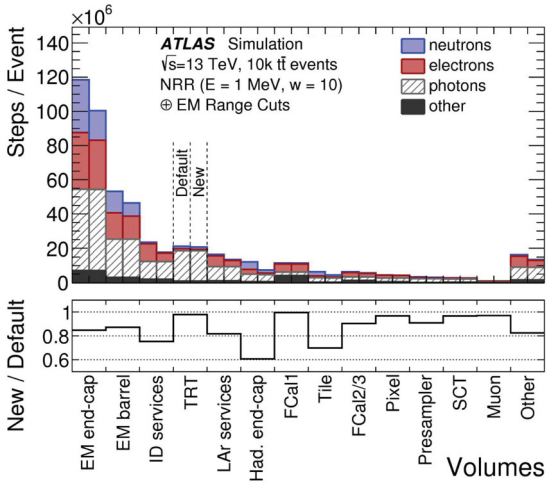
Woodcock Tracking introduces an additional, fictitious or δ-interaction, which does not alter the initial state, with a macroscopic cross section σ_δ that can be expressed as

$$\sigma_\delta(E, \text{material}) = \text{const.} - \sigma_\gamma(E, \text{material}), \quad (1)$$

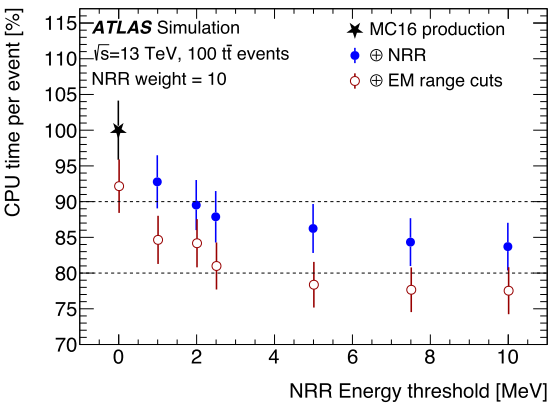
where $\sigma_\gamma = \sum_p \sigma_\gamma^p(E, \text{material})$ is the total macroscopic cross section, summing up the cross sections for all possible interactions p of a photon in material σ_γ^p, which itself is already simplified to one process within the `G4GammaGeneralProcess`. Using this fictitious interaction, the macroscopic cross section $\sigma(E)$ is now constant and can be written as

$$\sigma(E) = \sigma_\gamma(E, \text{material}) + \sigma_\delta(E, \text{material}) = \text{const.}, \quad (2)$$

which is constant also and especially if the material changes along a step. Using $\sigma(E)$ to sample the step length $s(E)$ until the next (real γ or a δ) interaction eliminates the need to stop at volume or material boundaries. Therefore, volume



(a)



(b)

Fig. 7 **a** Average number of GEANT4 steps per event as a function of the subsystem in the Run 2 setup (MC16 production). **b** CPU time per event for various thresholds of the NRR algorithm relative to the average Run 2 MC16 value (black dot) with or without the EM range cuts. Error bars indicate the root-mean-square of the CPU time for the simulated events. Figures from Ref. [153]

or material boundaries can be ignored and the probability of a real interaction P_γ can be calculated as:

$$P_\gamma(E, \text{material}) = \sigma_\gamma(E, \text{material})/\Sigma(E). \quad (3)$$

By applying this technique, the number of cross section evaluations as well the number of steps caused by the crossing of a geometric boundary is drastically reduced, without compromising the physics results. Woodcock Tracking was implemented and tested to be used in the EM endcap calorimeter and benchmarks have shown a speedup of 17.5% for $t\bar{t}$ MC events.

VecGeom VecGeom [155] is a geometry modelling library with hit-detection features designed to support CPU optimisations such as data-level parallelism. It provides optimised and fast geometry primitives and navigation algorithms that perform well especially for complex geometric shapes. The detector geometry used by Geant4 is built from classes inheriting from `G4Solid`, each representing a different shape (or set of shapes). It is possible to replace specific `G4Solid` classes with their VecGeom equivalents. The optimal set for the ATLAS geometry was found to be cones, tubes, and polycones. The use of VecGeom for these shapes gives a speed-up that varies depending on the CPU model from 2%–7% for $t\bar{t}$ MC events.

Geant4 static linking *Static linking* [156] is a purely technical optimisation that targets the way Geant4 is linked and used within Athena. By default, Athena builds many small shared libraries, one per package of code, which are dynamically loaded at runtime (see Sect. 6.1.1). Different build types were compared and tested for performance: dynamic (the default multi-library configuration used in Athena during Run 2), single dynamic library, and static linking.[12] Tests with the single dynamic library resulted in a slowdown in the execution time, which can be ascribed to the trampoline/lookup table mechanism of dynamic linking. Each call to a function in a dynamic library takes advantage of a trampoline that reads the memory address of the called method from a lookup table and passes it to the calling function. This results in an increased number of calls and jumps, which slows down the simulation execution. The static linking has proven to be the best performing build type. To enable it in Athena, all the packages that link to Geant4 were reorganised into one large library that then can be linked statically with Geant4. During execution, although the remainder of Athena continues to use dynamic libraries, function calls internal to this static library benefit from using the known function code locations within a statically linked library and avoid the overhead from lookups. Benchmarks have shown a gain of about 5%–7% for $t\bar{t}$ MC events.

[12] More information about shared libraries is available in Ref. [157].

EMEC geometry optimisation The EM endcap calorimeter (EMEC) is built in a complicated 'Spanish Fan' geometry, which could not be efficiently described with the geometric primitives available in early versions of Geant4. The geometry was therefore described with custom solids implementing the geometry algebraically. In Run 3, the code describing the EMEC detector was optimised and two new variants were introduced, besides the nominal one called the Wheel variant:

- Cones: this version reduces the use of `G4Polycones` by the introduction of an improved shape (`G4ShiftedCone`). In this configuration, the outer wheel is divided into two conical sections.
- Slices: this variant reduces the time needed for geometry navigation calls by dividing the inner (outer) wheel into 14 (21) thick slices along the z-axis.

The Slices variant was found to be the best performing, bringing a speedup of about 5%–6% for $t\bar{t}$ MC events.

Magnetic field tailored switch-off The solenoid field in the inner detector returns through the iron yoke supporting the tile calorimeter, resulting in a very small residual magnetic field within the bulk of the calorimeter volume. It is therefore possible to switch off the magnetic field in the LAr calorimeter for all particles except muons without significantly affecting the shower shapes. In addition to the small field value, the effect of the magnetic field on the electron trajectories is dwarfed by the effect of multiple scattering. However, the calculation of the showering electron and positron trajectories in the magnetic field still requires significant CPU time. This filter on the detector region and particle type was implemented and integrated to be used in production for Run 3. Tests with 200 $t\bar{t}$-production events have shown a speedup of about 3% for $t\bar{t}$ MC events. The same concept could be further exploited for other parts of the detector.

The frozen showers method The frozen showers method [158] speeds up the simulation of showers in the calorimeters by replacing low energy electrons, photons and neutrons with pre-simulated showers. This mostly affects secondaries that are produced in huge numbers in high-energetic showers. The showers are simulated with Geant4 until the energy of the particles produced in the shower falls below the energy threshold, at which point HITs are generated from the shower library. This approach was already used for the forward calorimeter in the Run 2 MC simulation production. The energy thresholds are 1 GeV for electrons, 10 MeV for photons, and 100 MeV kinetic energy for neutrons; below these values, showers from libraries are used. The pre-simulated showers are stored in ROOT libraries, binned in $|\eta|$ and

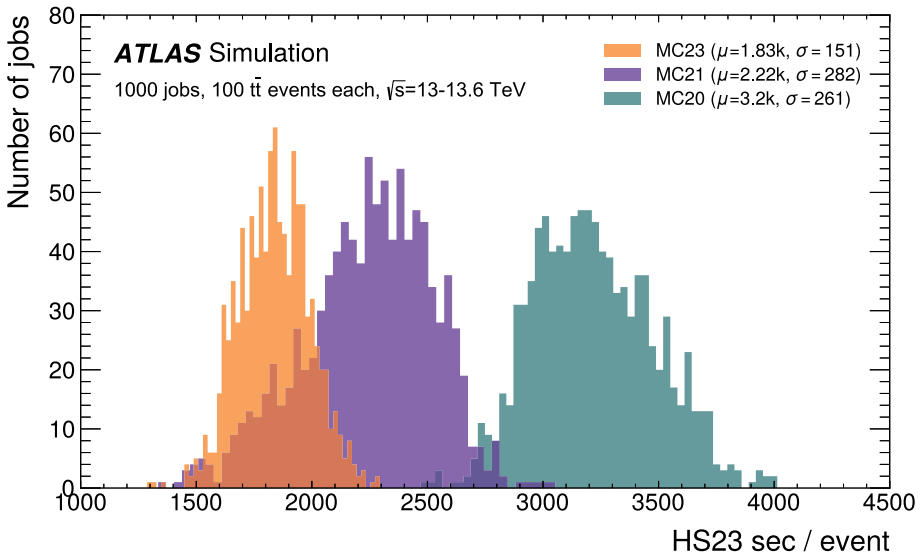


Fig. 8 Distributions of ATLAS GEANT4 detector simulation HS23 sec per event using the nominal and the optimised versions of the software by the GEANT4 Optimisation Task Force for the MC20, MC21 and MC23 campaigns, corresponding to a centre-of-mass energy of 13–13.6 TeV. The benchmarks comprise 1000 jobs simulating 100 $t\bar{t}$ events each at the Brookhaven National Lab Tier-1 cluster. The mean value (μ) and standard deviation (σ) of the distributions are also indicated

distance from the closest rod[13] centre; different libraries are also used for the first and second hadronic compartments of the forward calorimeter. The library is derived from a $t\bar{t}$-production event sample simulated with the GEANT4-based full simulation.

In the first Run 3 MC campaign (MC21) the frozen showers library was derived by scaling the energy of the showers in the Run 2 library to reproduce the energy scale observed in a Run 3 GEANT4-based full simulation sample. The energy thresholds and the bins were kept the same as in Run 2. A new Run 3 library was developed in 2022, with revised bins, providing a more accurate modelling of the GEANT4-based full simulation without the need for energy scale tuning. The improved library is used in MC23. In all campaigns, the use of frozen showers reduces the CPU required to simulate high-energy (several hundred GeV) electrons and photons in the forward calorimeter by a factor of three, and reduces the overall simulation time for $t\bar{t}$-production events by about 25%.

Summary of full simulation optimisations The MC21 simulation campaign included the optimisations described above in this section. The MC23 simulation campaign added the Woodcock Tracking optimisation. The CPU required to simulated one $t\bar{t}$-production event was reduced by 48% (36%) relative to Run 2 in the MC23 (MC21) campaign, as shown in Fig. 8. This reduction allows the simulation of 92% (55%) more events with the same CPU resources for MC23 (MC21).

Each of these changes to the simulation is put through a rigorous validation of physics performance (see Sect. 5.2). Only those changes that were found to not alter the physics performance, and therefore to not affect the agreement between data and MC simulation, are put into production.

[13] The forward calorimeter can be thought of as a large, solid tube with many holes in it. These holes are filled with small rods, and the gaps between the rods and the tube are filled with liquid argon. It is the distance to these rods that is of relevance for the frozen showers.

4.2.2 *Fast simulation* ATLFAST3

Overview Even after the optimisations discussed in the previous section, the full simulation of the detector requires considerable CPU resources. For this reason, ATLAS has developed tools to replace the calorimeter shower simulation, which is the most CPU intensive step, with faster simulation methods. In ATLFAST3 [42], the simulation of hadrons, photons and electrons in the calorimeters is handled by a combination of two fast simulations tools; `FastCaloSimV2`, which uses a parametric approach, and `FastCaloGAN`, which uses generative adversarial networks (GANs). `FastCaloGAN` is among the first tools based on generative models used for production in a large HEP experiment. `FastCaloGAN` was developed later than `FastCaloSim`, which struggled to reproduce lateral correlations in hadronic showers, and overcame that challenge using ML methods. Each tool is applied in the kinematic region in which it provides the best performance [42], after detailed validation and comparisons of physics observables. These tools replace the slow propagation and interactions of incident particles with the direct generation of energy deposits in the calorimeters. For the average ATLAS MC simulation event, ATLFAST3 requires only 20% of the CPU of the full simulation, where most time is spent on the simulation of the inner detector with GEANT4.

ATLFAST3 (as deployed in Run 2) was initially tuned to reproduce the output of the full simulation for Run 2 as closely as possible. The update of the GEANT4 version used in Run 3 simulation implies that the shape of EM and hadronic showers is sufficiently different between Run 2 and Run 3 full simulation samples that the parameterisation used by `FastCaloSimV2` and the training of the neural networks used by `FastCaloGAN` needed to be updated accordingly. For the Run 3 version of ATLFAST3, several qualita-

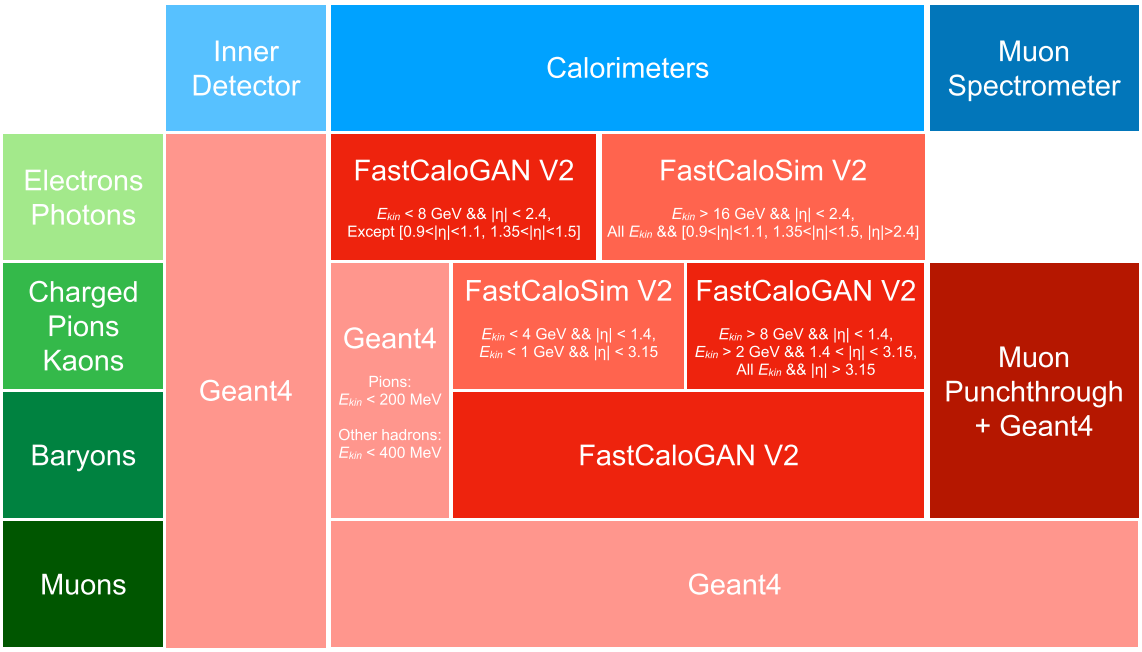


Fig. 9 The configuration of tools that together form ATLFAST3. The tools (GEANT4, `FastCaloSimV2` and `FastCaloGANV2`) are combined in a way that the physics performance of ATLFAST3 is closest to GEANT4, which depends on the particle type, η slice and kinetic energy (E_{kin}) range. The inner detector and muon spectrometer are simulated using only GEANT4

tive improvements have also been developed, in particular for `FastCaloGAN`, resulting in an upgraded version of the tool now called `FastCaloGANV2`. These improvements are listed below.

The training of the GANs and the parameterisation used for `FastCaloSimV2` are based on single particles that are simulated with GEANT4. This training is performed separately for various particle types and in fine bins of η, because the detector geometry and material changes strongly with η. The parameterisation is separately performed for 17 different energy values; the GANs in `FastCaloGANV2` are trained for two energy ranges. The inputs for the lateral shower shape modelling are *HITs* (point-like localized energy deposits) for `FastCaloSimV2` and *voxels* (small, regularly-sized volumes in which energy deposits are integrated) for the GANs. The granularity of voxels was optimised and is finer than that of the calorimeter cells, which improves the modelling.

For each particle type and energy, the tool that reproduces the GEANT4 simulation output with the best accuracy is used. The combination of the two tools was reoptimised in Run 3, and is illustrated in Fig. 9. GEANT4 is still used for the simulation of all particles in the inner detector, for muons and very low energy hadrons in the calorimeters, and in the muon spectrometer. `FastCaloGANV2` is used for simulating baryons (except at very low energies), low-energy photons and electrons, and higher-energy pions. `FastCaloSimV2` is used for pions with lower-energy and higher-energy electrons and photons.

High-energy hadrons may interact late – or even not at all – in the calorimeter. The resulting spray of hadrons into the muon spectrometer is known as *punch-through*. The particles produced in these showers are now modelled with a new tool based on deep neural networks (DNN). This tool can predict the probability of a punch through occurring better than was possible in the Run 2 version of ATLFAST3.

For most distributions of properties of objects used in physics analyses, ATLFAST3 and GEANT4 agree within a few percent. The agreement is much better than that achieved with the previous generation of fast simulation, ATLFAST2 [41], with key improvements in the modelling of the forward calorimeters and a better fluctuation model that enables in particular the simulation of substructure within jets [42]. Comparisons of energy distributions obtained with the Run 3 version of ATLFAST3 to those of GEANT4 are presented in Fig. 10 for single pions. Further comparisons for reconstructed single photons and pions are shown in Fig. 11. The comparisons for single pions display shower moments as defined in Ref. [159], where λ is the energy-weighted distance of a cell from the shower centre along the shower axis, and r is the energy-weighted distance of a cell from the shower centre perpendicular to the shower axis. Figure 11e in particular shows an example of a distribution with significantly better modelling in `FastCaloGANV2` compared with `FastCaloSimV2`; these distributions were among those that motivated the choice of the scheme shown in Fig. 9.

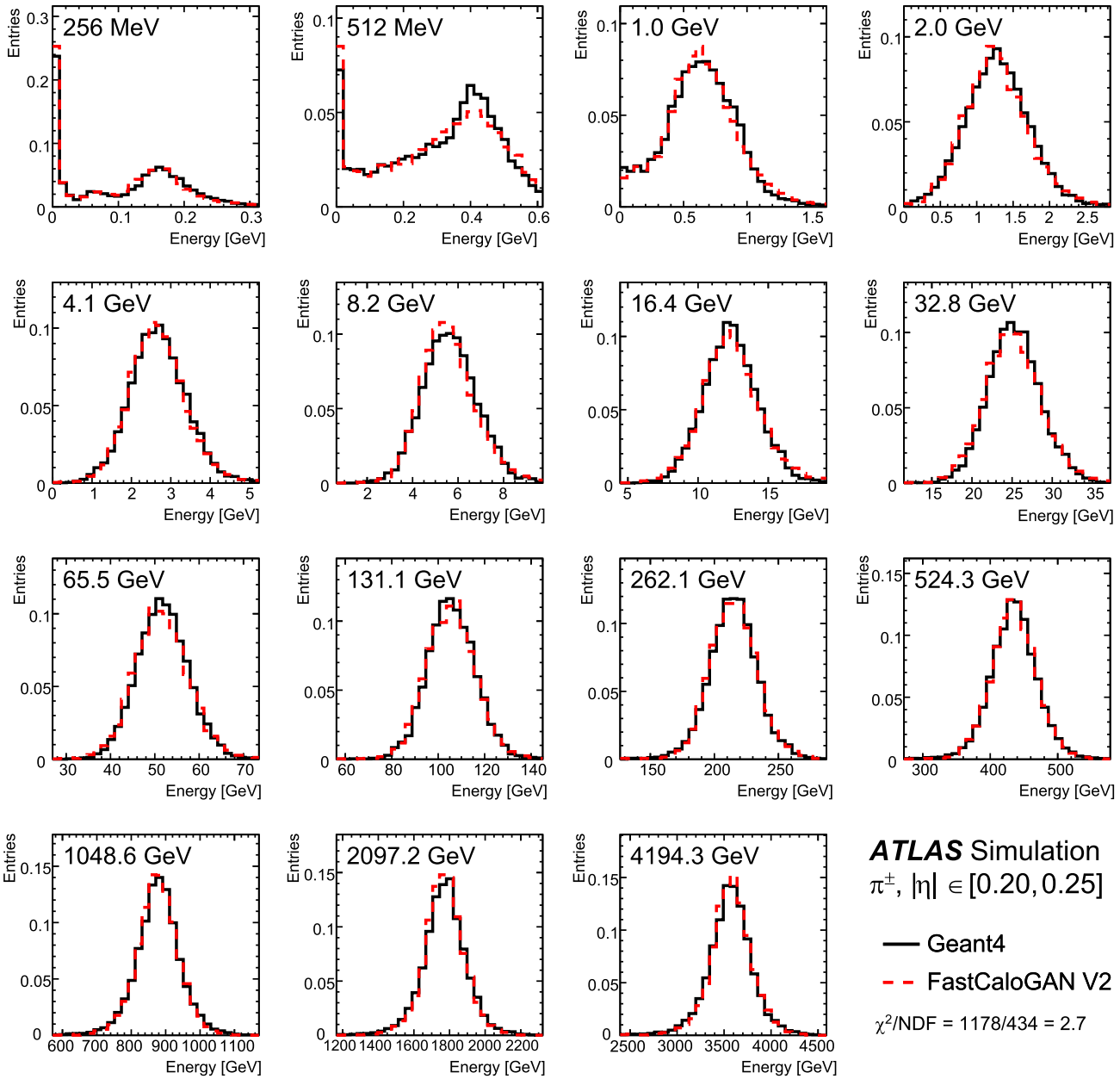
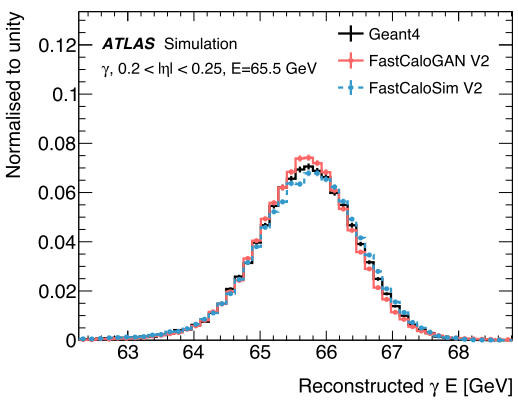


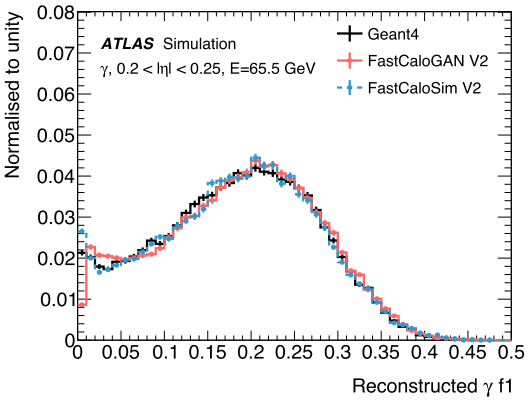
Fig. 10 The performance of `FastCaloGANV2` for single pions in an $|\eta|$ range of 0.2–0.25. The energy distributions from GEANT4 are well reproduced for all energy values. The single pion truth energy value is indicated in each figure

`FastCaloGANV2` improvements for Run 3 Several improvements were made to `FastCaloGAN` that are incorporated into the upgraded version called `FastCaloGANV2`, which is used in the Run 3 version of ATLFAST3:

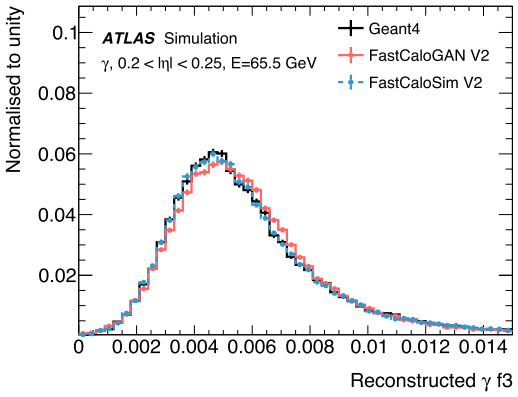
- The number of volumes in which GEANT4 energy deposits are grouped (voxels) was optimised, reducing extrapolation in voxel-to-cell energy assignment.
- The bias in the energy of HITS generated with a small simulation step, described in Ref. [42], was corrected, resolving a discrepancy in the reconstructed electron and photon total energy.
- A similar correction is applied to the hadrons but the energy at which the shower is rescaled is derived from the cell energies rather than the GEANT4 HITs.
- The ϕ-modulation correction [42] related to the accordion structure of the EM calorimeter is now corrected before training.
- The architecture of the networks and the hyperparameters were optimised for each particle, energy range, and detector region.



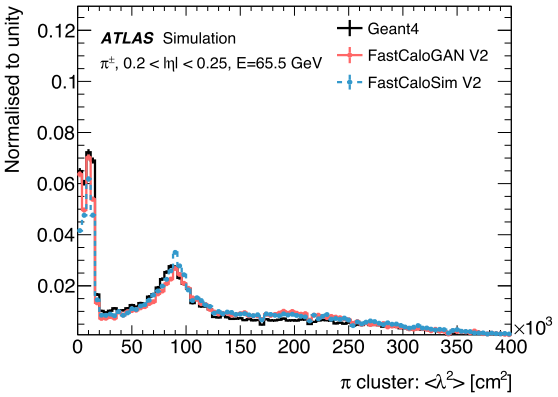
(a)



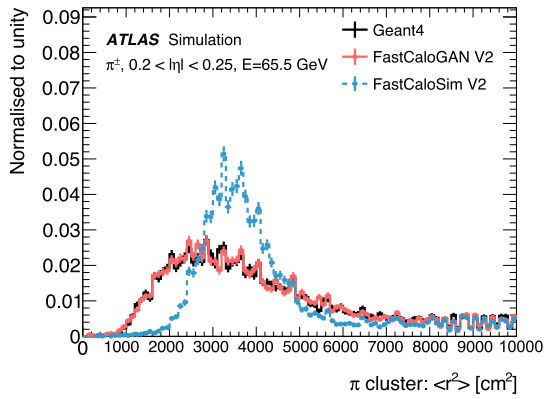
(b)



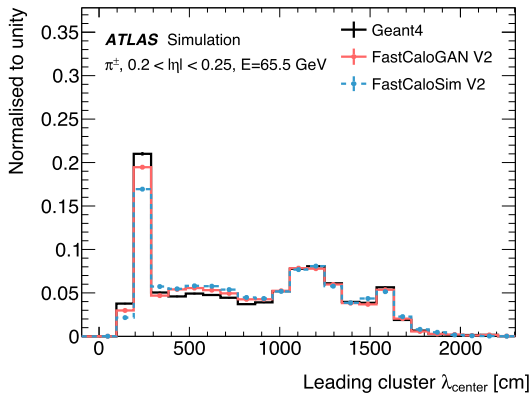
(c)



(d)



(e)



(f)

Fig. 11 **a**–**c** The energy of reconstructed central photons with a true energy of 65.5 GeV. **a** displays the total energy, **b** the energy fraction deposited in the first calorimeter layer (relative to the total energy), and **c** the energy fraction in the third calorimeter layer. **d**–**f** Shower moments for reconstructed central pions with a true energy of 65.5 GeV. **d** displays the second moment in λ, **e** the second moment in r, and **f** the λ_{center}. GEANT4 (black) is compared with `FastCaloGANV2` (red) and `FastCaloSimV2` (blue)

- Separate trainings are done for low- and high-energy electron and photon showers, and the minimum energy was lowered from 256 MeV to 64 MeV, matching the value used in `FastCaloSimV2`.
- The training strategy was updated to use the full set of initial particle energy ranges from the first iteration, rather than starting from a single range and slowly adding more energies, as was done previously.
- The HIT-to-cell assignment was improved by correcting for the average lateral energy distribution within each voxel. More energy is deposited closer to the centre of the shower as occurs in GEANT4.
- A minimum energy of 10 MeV is used for HITs in electron and photon showers as in `FastCaloSimV2`. No changes were introduced for pions, i.e. the total energy assigned to a voxel is split between the HITs without any minimum energy criteria.
- The GANs also benefit from a new correction for the longitudinal position of the HITs within a layer. This is implemented using a DNN extrapolation method trained on the energy deposited in each layer.

In addition to these improvements, 100 new GANs were trained using protons to parameterise their energy response in the calorimeter; these GANs are then used to simulate baryons. This method replaces the previous approach in which only the total energy was corrected and thus enables consideration of the different shape of the showers between pions and baryons. This new feature significantly improves the simulation of low energy (< 10 GeV) baryons. Kaons are still simulated based on pions, since they have similar shower shapes.

Outlook Several further improvements to the ATLFAST3 tools are being developed. These include:

- For fast calorimeter simulation, the use of variational autoencoders and other methods such as flow or diffusion models are investigated and the performance compared with the GANs.
- A new voxelisation is being investigated with the aim to improve the description of showers in the highly granular regions of the EM calorimeter where the relatively course voxelization is expected to play a more significant role.
- A low-energy parameterisation is planned to address the simulation of pions with kinetic energy below 200 MeV, currently done with GEANT4; this will further speed up the simulation.
- The memory footprint of `FastCaloSimV2` will be optimised, and the memory of the GANs will be reduced using ONNX instead of LWTNN for the inference of the models.
- In the ATLFAST3 infrastructure, the fast simulation will be integrated more tightly with GEANT4 using the `G4VFastSimulationModel` class. The reduction of custom code is expected to make the fast simulation code less brittle and easier to develop and maintain.

In addition, an even faster simulation will be delivered by the FastChain [160] project, which includes two research activities that are still planned for deployment in Run 3. FATRAS (Fast Tracker Simulation) [161,162] aims to replace the propagation of particles in the inner detector using GEANT4 with a simplified treatment of particle interactions with the simpler detector description model used in track reconstruction (see Sect. 3.5); preliminary results show that the CPU time needed to simulate $t\bar{t}$-production events can be reduced by a factor of 30 over that of ATLFAST3. Another speedup as a part of FastChain is the treatment of pile-up simulation, using a technique called Track Overlay, described in Sect. 4.3.2.

4.3 Digitisation

After simulation using the GEANT4 toolkit or ATLFAST3, all energy deposits in ATLAS sub-detectors sensitive volumes are stored in a file format called `HITS`. The digitisation code reads `HITS`[14] and emulates the detector response, producing an output (*digits*) that conceptually mirrors the real data detector response (typically voltages or times on pre-amplifier outputs), with the addition of some truth and metadata information that is then altogether stored in RDO output files. In addition, the inputs for the hardware trigger are produced during the digitisation step. This operation is strongly specific to the different sub-system technologies used in ATLAS. Section 4.3.1 describes the sub-system specific code, with an emphasis on the new features introduced in Run 3.

In digitisation, the treatment of the proton–proton collision of interest, often referred to as the signal or hard-scatter collision, is typically separate from the treatment of the additional proton–proton background collisions (pile-up). In Sect. 4.3.2 the treatment of pile-up during the LHC fill in Run 3 collisions and various techniques to merge it with the hard-scatter information are also described.

4.3.1 Sub-system digitisation code

Pixel digitisation The pixel digitisation takes the energy deposits from charged particles in the silicon wafers (as simulated by GEANT4) and converts them into time-over-threshold

[14] Time information is also included in `HITS` files. For all sub detectors, the time of flight at the speed of light from the interaction point is subtracted from the GEANT4 HITS' time during digitisation.

(ToT) values on pixel sensor electrodes. The first step of the pixel digitisation process is to divide the ionisation energy deposit from each GEANT4 HIT into a maximum of twenty ionisation energy deposits using the Bichsel model [163] to correct for possible straggling in thin silicon.

As the integrated luminosity delivered by LHC since the installation of the pixel detector increases, the effects of radiation damage in the silicon bulk on the detector response become increasingly important. The pixel detector digitisation for Run 3 includes for the first time in production the effects of radiation damage in the silicon sensors [164]. This represents a valuable tool to understand and predict radiation damage effects and their relation to the performance of physics object reconstruction.

The radiation damage digitiser computes the signals induced by the charge carriers produced by ionising particles in GEANT4 [38] by using precise electric field, Lorentz angle and weighting potential maps, taking into account carrier-trapping and diffusion effects. The electric field distribution for a given applied bias voltage after irradiation at a given fluence are taken from detailed TCAD (Technology Computer Aided Design) simulations. The predicted electric field is used to calculate the expected time spent by charge carriers to reach the collecting electrode, via the carrier-mobility relation. The induced charge on the pixels is calculated from the initial and trapped positions using the weighting potential map and including charge sharing effects. The total induced charge is converted into a time-over-threshold value used for clustering in the reconstruction.

The IBL charge collection efficiency variation as a function of the integrated luminosity measured in data and that predicted by the radiation damage simulation from the start of Run 2 is presented in Fig. 12 [165]. For Run 3, the track reconstruction algorithms were re-tuned to at least partially mitigate the performance degradation caused by the radiation damage [165].

The charge calibration of the reconstruction involves recalculating the charge from the ToT values. While ample data are available to calibrate this procedure at moderate ToT values, for very large and very small ToT values an extrapolation or fit function must be used. For Run 3, the pixel charge calibration procedure for the IBL was modified to avoid the fitting step of the charge and ToT values that may cause biases at low and high charge values. The new IBL calibration procedure implements look-up tables that use the average of the injected charge values on the pixels of a front end chip corresponding to each ToT calibration point. These look-up-tables are stored in the calibration database and applied to detector data. In simulation, the ToT is extracted by linear interpolation of the input charges and rounded to an integer value.

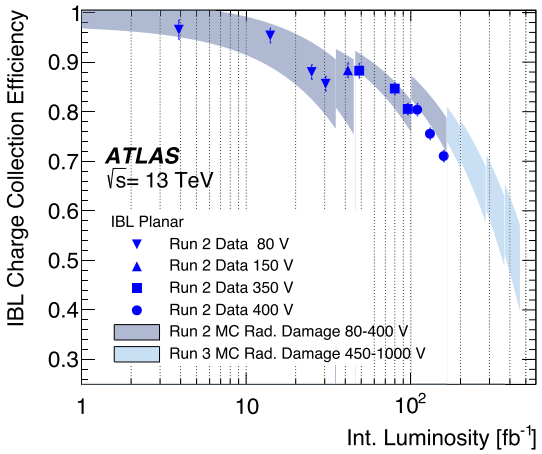


Fig. 12 Charge collection efficiency as a function of integrated luminosity for IBL planar sensors for Run 2 data and the ATLAS radiation damage simulation for Run 2 and Run 3. The points represent the data and the bands the simulation predictions. The parametric uncertainty in the simulation defining the width of the bands includes variations in the radiation damage model parameters and the uncertainty in the luminosity-to-fluence conversion. Horizontal error bars on the data points due to the luminosity uncertainty are smaller than the size of the markers. Figure from Ref. [165]

SCT digitisation The SCT digitisation process begins with the conversion of the energy deposited on the silicon wafer by each particle into a charge on the readout electrodes. The energy deposited in each GEANT4 tracking step is divided uniformly into 5 μm sub-steps and converted into an electron–hole pair for each 3.63 eV of deposited energy. The charges are drifted towards the electrodes taking into account the diffusion and the Lorentz angle, which are calculated assuming a uniform electric field in the sensor.

The final stage of digitisation is the simulation of the response of the readout electronics. The amplified signal of a charge arriving at a readout strip is calculated for three different readout time intervals, corresponding to three consecutive bunch crossings. The cross-talk between neighbouring strips is also taken into account and estimated from the shape of the main output pulse of the strips. Electronic readout noise is generated independently in each time interval from a Gaussian distribution based on the data measurements and added to the strips. The nominal readout threshold, above which the signal is recorded on a single strip, is set to 1 fC according to the data acquisition configuration. To reproduce the noise occupancy observed in the data, random strips are added to the readout list from among those with no deposited charge.

TRT digitisation The TRT digitisation software [166] converts HITs produced by the GEANT4 simulation into digits that correspond to the detector readout signal for each straw. Each digit consists of a timed bit-pattern spanning 2.5 bunch crossings. This bit-pattern contains 20 low-level threshold bits (3.125 ns/bit) that are set if the signal exceeds

a low threshold in the corresponding time interval. One high-threshold bit is set if the high threshold is exceeded at any time during the 25 to 50 ns time interval corresponding to the bunch crossing of interest.

The signals in each straw are simulated in detail. For charged particle tracks, energy clusters are created and placed randomly along the path of the track in the active gas volume. The number of clusters is calculated by sampling a Poisson distribution with the most probable value set to the mean free path of the particle. For photons from transition radiation and bremsstrahlung, a single energy cluster is created. The number of drift electrons in a cluster is determined from the energy of the cluster. Some electrons can be recaptured stochastically as they drift towards the anode, where the number of surviving electrons is determined by sampling a Binomial distribution with a survival probability of 0.4. The time taken for each surviving electron to arrive at the anode wire is determined from the electron drift length and includes a detailed mapping of the magnetic field that causes the electron trajectories to bend. Each electron that arrives at the anode wire cascades and induces a signal whose amplitude is simulated as a random sample of an exponential distribution. The propagation time for the signal to arrive at the front-end electronics and the attenuation in the wire are determined taking into account that half of the signal travels directly to the front-end while the other half travels in the opposite direction and reflects.

The signal from each drift electron is superimposed and convolved with signal shaping functions (separately for low and high threshold and for xenon and argon gas mixtures). Electronic noise is added and the signals are then discriminated against low and high thresholds in appropriate time slices and encoded into the output signal that is saved in the RDOs.

In earlier LHC operations where detector occupancy was relatively low, digits were written to storage in (digit, straw identifier) pairs only for hit straws. To reduce the size of the TRT RDO at higher LHC luminosity where most TRT straws are hit, this scheme was changed to sequentially write out digits for all straws omitting the straw identifier and use a detector map to assign the straw identifier in the reconstruction step. To further reduce RDO size, and CPU time, the simulation of noise in straws with no hits is removed. To account for inaccuracies in high threshold probability due to the overlay procedure, corrections are applied during overlay by randomly adding high-threshold bits with a probability that increases with occupancy.

LAr digitisation The energy deposited in the liquid-argon gaps of the LAr calorimeters [167] induces an electrical current that is proportional to the deposited energy. The signal is then amplified and shaped in the front-end readout electronics [168] using three different gains to cover the large dynamic range of the signals of interest. The output is then sampled at the LHC clock frequency and converted to ADC (Analog to Digital Converter) counts. The digitisation [169,170] emulates the detector readout, converting the deposited energy for each bunch crossing to ADC counts, taking into account the ADC gain, the LAr sampling fraction and the energy calibration of each cell. The ADC conversion also considers the time of each event relative to the hard scatter collision time (*cell-timing*). During MC overlay (see Sect. 4.3.2), the ADC counts in the presampled RDOs are converted back in raw energy, which is then added to the energy from the hard scatter, for each time sample and cell. The combined energy is then converted back into ADC counts and the electronics noise is added. Optimal filtering coefficients (OFCs) [171] are then used in the reconstruction to compute the energy per cell, as described in Ref. [167].

To save disk space, the identifiers of the channels are not stored; instead, a vector with a length equal to the number of readout channels is created, and for each channel two bits are used to store whether that channel has a signal and, if so, what readout gain was used.[15] Another vector stores the ADC values of the channels with a signal. The number of samples is assumed to be the same for all readout channels and so it is only stored once.

As part of the activities of the Phase-I (Run 3) trigger upgrade, it was suggested in Ref. [172] that the hardware trigger system for the calorimeter (L1Calo) could profit from introducing *Super Cells* [173]. Super Cells have higher granularity than trigger towers [8] (which are $\delta\eta \times \delta\phi = 0.1 \times 0.1$ in the detector central region) used during Runs 1 and 2, with dimensions further optimised to describe electron and photon showers. In the digitisation step of the electronics simulation, these are emulated by grouping together the HITs (energy–time pairs) that are related to all cells that form each of the Super Cells. Mapping tools allow such grouping based on the identifiers of the cells. The information is used to simulate pulses in all bunch crossings still prone to produce some signal in the event of interest (i.e. the present event at $t = 0$), given the length of the liquid argon ionisation pulse in the detector. The normalised expected pulse shape as recorded in the conditions database is multiplied by the estimated amplitude of the pulse, which is directly related to the energy content of the Super Cell and shifted by the HIT time, producing estimated samples. Samples from multiple HITs composing the same Super Cell are simply added together, forming the pile-up plus hard-scattering digits. Electronics noise, estimates of which are also saved in the conditions database, is added at this step. This way, digits with regular 25 ns sam-

[15] The detector readout uses three different gain settings to more precisely read out a wide range of energies. The appropriate gain is chosen per cell, based on the energy in that cell.

ples of the signal are formed and recorded as part of the RDO content.

Tile digitisation For the tile calorimeter (TileCal), HITs contain energy deposits in scintillator tiles. Since all normal TileCal cells are read out by two photomultiplier tubes (PMTs), every energy deposit is split into two parts according to the optical model and stored in two independent `TileHit` objects that correspond to the two PMTs. All energy deposits are accumulated in 0.5 ns time bins, and every `TileHit` contains a vector of time values that corresponds to non-empty time bins and a vector of total energy deposits in every time bin.

The first step in TileCal digitisation is the simulation of photo-statistics effects [174]. According to measurements from beam tests, at nominal PMT gain 70 photo-electrons per 1 GeV of deposited energy are created in a PMT. Energy stored in a `TileHit` is smeared according to a Poisson distribution with an average value equal to the number of photo-electrons created in a given PMT in a given event.

After that, every energy deposit is converted independently to a pulse using pulse shapes measured at the beam tests in 0.5 ns time steps [175]. Every channel is readout by two 10-bit ADCs, called low gain (LG) and high gain (HG), with a gain ratio of $1:64$ between them. Two pulses are created with amplitudes proportional to the value of the energy deposit, converted from MeV to ADC counts, taking into account the ADC gain and TileCal sampling fraction. The position of the pulse maximum is shifted according to the time value from the deposit. All individual pulses are summed up to construct final LG and HG pulses in every PMT, and these pulses are 'digitised' at seven fixed times: 0, ± 25, ± 50, and ± 75 ns, and two vectors of seven samples are created. After all the signals are summed up, a constant pedestal value of about 40–60 ADC counts (depending on the cell) is added to the resulting pulse.

To simulate electronics noise, Gaussian-distributed noise of about 1.5 (0.7) ADC counts is added to the HG (LG) samples. After the two final pulses are constructed, the LG ADC is dropped if the maximal amplitude in the HG ADC remains below 1023 ADC counts (which roughly corresponds to 10–12 GeV). If the HG ADC saturates, the LG ADC is selected. So, at the end of the digitisation step a single `TileDigit` object with seven samples is created for every readout channel. This `TileDigit` object represents the data that are sent from the on-detector electronics (located in the support drawers for the TileCal) to the off-detector electronics.

The next step is to simulate the behaviour of off-detector electronics, which, similarly to the case of the LAr calorimeter, reconstructs cell energies using OFCs. The results of the OFC application are stored in `TileRawChannel` objects and the `TileRawChannelContainer` with all 12,228 readout channels is written to the output RDO file [176].

Muon system digitisation The digitisation of the muon spectrometer sub-systems installed for Run 1 (MDT, RPC, TGC and CSC[16]) was documented in Ref. [177]. The RPC digitisation code has not changed substantially since then, while some noticeable improvements were made to the TGC digitisation code: a new timing calculation is used in the bunch crossing identification, considering the position dependence of the signal propagation to the front end; the bunch identification is done for a 4-bunch readout from the previous to the next-to-next bunch crossing; the channel cross-talk calculation was improved; and numbers previously read from text files were migrated to the conditions database.

The MDT digitisation code processes GEANT4 `MDTSimHits` to make MDT digits that are a simulation of MDT raw data. The `MDTSimHit` data has an identifier that identifies the specific MDT tube hit, the global time, and the impact parameter of the track relative to the wire in the MDT tube. The digitised HITs consist of an offline identifier indicating the specific MDT tube hit, the drift time emulating the output of the Time-To-Digital Converter (TDC) and pulse height data emulating that calculated with an ADC. The MDT digitisation code includes a simulation of ionisation clusters in the MDT tube that are propagated to the MDT wire using a time-to-space function. An amplifier response function simulates the signal generated at the wire from drift electrons. The first drift electron which creates a signal over threshold determines the drift time. The TDC signal is calculated by combining the drift time, time-of-flight from the interaction point to the MDT, propagation time of the signal along the MDT wire, and the simulated ATLAS beam clock time. The ADC value is calculated using a simulation of the Wilkinson ADC [178] used in the MDT electronics. The TDCs have a programmable dead time that is set to the maximum dead time of the tube [179]; this is accounted for in the digitisation when the HIT with the earliest time sets the beginning of the dead time window, during which additional signals are discarded.

The New Small Wheels (NSW) are composed of Micromegas (MM) and small-strip Thin Gas Chambers (sTGC). Similar to the other subsystems, the digitisation consists of two parts, the first one modelling the response of the detector to the passage of any ionising particle and the second one simulating the response of the readout electronics.

The MM chambers contain micro-mesh gaseous detectors, with a gas gap of a few mm where charges are created and drifted, and an amplification region of 120–130 μm between the metallic mesh and the readout electrode. The signal is read from the strips of the readout electrodes, with a 425–450 μm pitch. For each strip, if the collected charge exceeds a set threshold, the charge and the time are recorded. Since

[16] As explained in Sect. 2.2, the CSC chambers were removed at the end of Run 2.

a detailed simulation of the charge creation, transport, and avalanche for billions of events would be extremely expensive in terms of CPU usage, the MM digitisation instead relies on distributions obtained by simulating the passage of a smaller number of muons through a Micromegas detector using GARFIELD++ [180], software specialised for the simulation of gaseous detectors. These distributions, e.g. for the diffusion or the number of ionizations per distance, are sampled in the digitisation, leading to good modelling of the detector response. Sampling the distributions also allows tuning of the simulated detector response towards the response of the real detector based on the parameters of the distributions.

The second part of the Micromegas digitisation is performing the simulation of the electronics response, in particular the response of the charge amplification and digitisation process carried out in a custom integrated circuit called the VMM [181]. The actual transfer function of the VMM is used to get the response of the shaping amplifier. Afterward, the peak height and time measurement are performed in the same way as implemented in the VMM. For the modelling of the charge threshold, a linear dependence between the noise level and the strip length is implemented using noise data gathered during the detector commissioning to determine the noise of the shortest and longest strips. Other features of the VMM are also implemented in the digitisation, including the ability to read the signal of a strip below the threshold if its neighbour has a signal above threshold (the *neighbour logic*), and the *address in real time* signal, providing the information of the first VMM channel (out of 64 channels per VMM chip) above the threshold, which is used for the trigger.

The sTGC consists of a multi-wire proportional chamber with three independent and complimentary readout technologies: wires, strips, and pads. When a particle crosses the sTGC gaseous gap, an avalanche forms along the ionising track towards the nearest wire. If the induced charge on the strips and pads exceeds set thresholds, the charge and time are recorded. A detailed timing spectrum and avalanche process on the sTGC anode wires operated at 2.9 kV is simulated using the GARFIELD++ [180] and MAGBOLTZ [182] packages. The induced signals on the strips and pads follow the response of the resistive layer as described in Ref. [183]. The detailed simulation of the sTGC is then parameterised for fast HIT digitisation in Athena. The energy lost by particles traversing the sTGC gaseous gap is provided by GEANT4 in ATLAS. This energy is converted into an effective ionization charge in the gap. The time of arrival of the first electron cluster onto the wire is parameterised as a function of the distance of closest approach of the particle. The effects of ionization, noise, electronics threshold and avalanche gain fluctuations are modelled with a Polya distribution. The pad and strips closest to the wire with the avalanche fire. The charge on the strips is modelled by a Gaussian distribution of tunable width as a function of the polar angle of the incoming particle. The pad, wire and strip timing spectrum, as well as the strip hit multiplicity, were tuned on data collected during test-beam campaigns [184,185]. The sTGC digitisation parameterised model offers an accurate timing performance, an adequate simulation of the charge sharing among adjacent readout strips, and a good representation of the overall spatial resolution. The final time and charge of the hits associated with a global BCID are obtained using a VMM calibration curve that converts a time in nanoseconds and a charge in picocoulombs into a *Time Detector Output* and a *Peak Detector Output* object, respectively.

4.3.2 Treatment of pile-up and beam size effects

MC overlay In MC23, as in MC20, the technique of MC Overlay [170] is used to add the effect of additional minimum bias collisions to simulated events. In this approach, digitisation is carried out separately for the hard-scatter part. Digitised hard-scatter events are then combined with pre-digitised pile-up events. The libraries of these presampled RDO events are campaign-specific and each amount to about 500 million events per year, corresponding to a size of 1 PB. This method has reduced CPU, memory and I/O requirements relative to the pile-up digitisation technique used in previous campaigns (see, for example, Ref. [41]). The technique is similar to the premixing technique used in CMS [186].

Pile-up digitisation is still used to create the presampled pile-up RDO datasets, but this is only done once per MC simulation sub-campaign (e.g. MC20a). This step takes as input two equally large datasets of minimum bias events in HITS format: a high-p_{T} sample containing at least one jet with p_{T} larger than 35 GeV, photon with p_{T} larger than 8 GeV, or b-hadron that has p_{T} larger than 5 GeV that decays to a lepton with p_{T} larger than 4 GeV, and a low-p_{T} sample with the remaining events. The two datasets are sampled according to their relative cross sections, but avoiding the duplication of the high-p_{T} minimum bias events that might cause visible features during analysis [170]. The presampled RDO events can then be re-used for multiple hard-scatter samples without causing any issues in physics analysis. Each presampled RDO event has a unique hash that can be used to tag whether a presampled RDO event is selected multiple times within an analysis. This can be used to adjust the presampled RDO dataset size and hence the level of re-use for future MC simulation campaigns. The lumi block-ordered hard-scatter HITS and presampled pile-up RDO files are both merged into files with 10,000 events each. Having the same number of events in the two file types has the advantage that when the MC Overlay job reads the Nth event from each file and combines them, the hard scatter and pile-up events have the same beam spot size. The MC Overlay code contains a sanity check that will abort the job if this is not the case.

In addition to the overlay algorithms described for the Run 2 detector sub-systems in Ref. [170], for Run 3 new overlay algorithms were prepared for the sTGC and MM sub-systems. One further new overlay algorithm was implemented for LAr Super Cells used as input to the Phase-I Trigger.

For the presampled pile-up, the signals in the sTGC (MM strips) from the hard-scatter and pile-up events are combined as follows. In a given channel, if a signal is only present in either the hard-scatter event or the pile-up event, that signal is copied to the output RDO. If a signal is in both, the output RDO depends on the timing of the signals and the sTGC VMM pulse shaping time (MM VMM integration time). If the hard-scatter and pile-up signals are separated in time by more than the set shaping time (integration time), the earliest signal is copied to the output RDO. If the signals are separated by less than the shaping time (integration time, about 200 ns), the two signals are combined by adding the charges and taking the earliest signal time.

As with standard LAr Overlay, LAr Super Cell Overlay is done at the digit level. Hard-scatter HITS are digitised in the standard way, except that no noise is added (as this has already been applied to the background LAr Super Cell digits). `LArDigits` contain several time-samples of the signal in each channel. The algorithm loops over the Super Cells. Any hard-scatter contribution is added to the pile-up background contribution, if available, and otherwise it is added to the standard pedestal for that channel. If any of the samples are below zero or above the maximum ADC value, then they are constrained to lie in this range. The results are written into the output `LArDigit` container. The creation of LAr Super Cells from the combined `LArDigits` then proceeds in the standard way.

Beam size effects are also taken into account in the minimum bias simulation. Separate minimum bias background datasets are simulated with each beam spot size (43, 40, 37 and 34 mm, see also Sect. 4.2.1). During the pile-up presampling step [170] minbias files with all four beam spot sizes are used as input. Separate Athena jobs are run for each beam spot size with the appropriate $\langle\mu\rangle$ distribution and number of events per job for that part of the run. The RDO files created are then merged into lumiblock-ordered presampled RDO files.

Track overlay Track overlay is an evolution of the concept of MC overlay. In MC overlay, simulated and digitised pile-up data (or digitised real detector pile-up data) is overlayed onto the simulated hard-scatter event. In track overlay, the inner detector tracks of simulated pile-up data are first reconstructed, and then these events are overlayed onto the hard-scatter event. Repeated ID reconstruction for the same events is thus avoided, and CPU can be saved (up to 50% based on preliminary performance tests). Track overlay is not yet applied in the recent MC23 campaigns, but its deployment is still planned for Run 3.

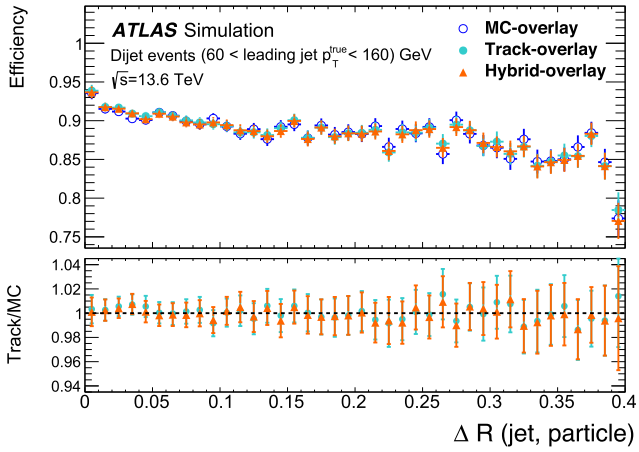
Tracks in the core of (often) high-p_{T} jets are affected by the presence of pile-up hits, meaning the individual reconstruction of pile-up and hard-scatter tracks leads to differences relative to the merged reconstruction. Track overlay is therefor not suitable for all events. To remedy that, a neural network is used to decide for individual hard-scatter events whether it is appropriate to use track overlay or MC overlay. This is called *hybrid overlay*. This neural network is trained on the tracks using truth features and variables relating to the track density, as well as pile-up conditions, with the aim to classify tracks into those that can be used for track overlay or not. The per-event ratio of these track populations is then used as a score, to decide whether the event is sent to track or MC overlay.

In Fig. 13, the efficiency of track reconstruction is shown as a function of the distance between the reconstructed jet axis and truth particles, for a sample with low or high-p_{T} jets. The reconstructed jets are particle flow jets with a radius parameter of 0.4 (see also Sect. 4.4.6). In Fig. 14, the track reconstruction efficiency for high-p_{T} jets is presented as a function of the jet p_{T}. Track overlay reproduces the efficiency of the MC overlay well for jets with moderate transverse momentum, while close to the centre of very high-p_{T} jets the efficiency is overestimated. The hybrid case results in a very good agreement with pure MC overlay even at high-p_{T}. In the studies presented, the ML score cut is 0.742, corresponding to a fraction of events used for track overlay of 93.5% for low-p_{T} dijets and 35.3% for high-p_{T} dijets.

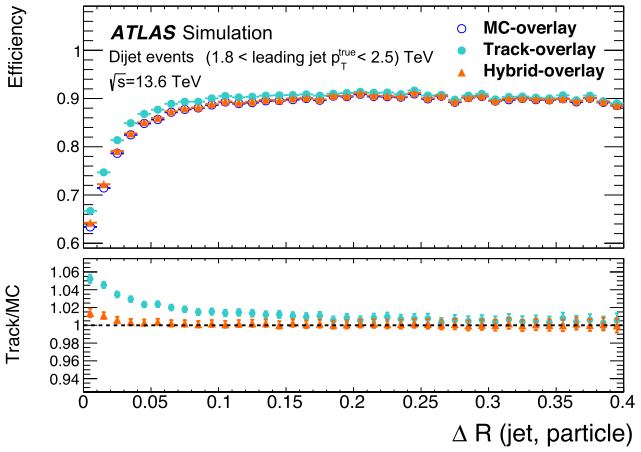
Data overlay Particularly for heavy ion data analysis, data overlay is used rather than MC overlay. In heavy ion collisions, rather than additional collisions, the 'background' is primarily the enormous underlying event from the collision itself. This background is extremely difficult to correctly model, owing to the complex particle correlations that arise from the dynamics of the collisions. For that reason, it is extracted directly from data events and overlaid onto 'signal' events.

The data are selected using special triggers based on the total energy in the event. Several different triggers are combined to produce a complete spectrum of total energy (which is strongly correlated with centrality in the heavy ion collisions). The events are then mixed together into batches that represent the entire heavy ion data-taking period, and the MC simulation is reweighted to reproduce the spectrum of total energy observed in the data. To ensure accuracy in the overlay step, the data must be read out without the normal suppression of calorimeter cells that are below threshold.

Simple proton–proton collision events are then generated for a variety of interesting processes. For each generated proton–proton signal event, a background heavy ion colli-



(a)



(b)

Fig. 13 The hard-scatter event track reconstruction efficiency for tracks in jets as a function of the distance between the jet axis and truth particles. Dijet events at 13.6 TeV containing a leading jet with **a** 60 GeV $< p_{\mathrm{T}}^{\mathrm{true}} <$ 160 GeV and **b** 1.8 TeV $< p_{\mathrm{T}}^{\mathrm{true}} <$ 2.5 TeV are reconstructed with MC overlay (empty circles), pure track overlay (filled circles) or the hybrid (filled triangles). For the hybrid case, the fraction of events that is sent to track overlay is 93.5% (35.3%) for the low-p_{T} (high-p_{T}) sample

sion is selected. The generated event is then placed exactly on top of the reconstructed vertex of the heavy ion data event, so that the particles from the proton–proton collision appear to also emanate from the same ion–ion collision as the background. The detector simulation then proceeds with the same detector geometry and conditions as are used for the reconstruction of the real detector data. At the digitisation stage, the signals from the simulated proton–proton collision and the data heavy ion event are combined, in much the same way as is done for MC overlay. The reconstruction then proceeds as normal. The result is an event that has a simulated signal, for example a Z boson decaying into two leptons, embedded in a heavy ion collision.

Because the data conditions are used, the heavy ion data overlay procedure perfectly emulates certain detector features, including noise, disabled detector channels, and non-collision backgrounds. However, because the simulation must use conditions from the real detector, misalignments can induce overlaps in the volumes simulated with GEANT4 – there is no requirement in the alignment that volumes not overlap. While for precision proton–proton collision measurements these overlaps are sufficiently concerning that they have prevented the up-take of data overlay, for heavy ion events where the events are much busier and most analysis is done using ratios of quantities to cancel out systematic effects, the risk is much reduced.

4.4 Reconstruction

Following the digitisation step, simulated events are passed through a trigger simulation [18]. The next step for both the

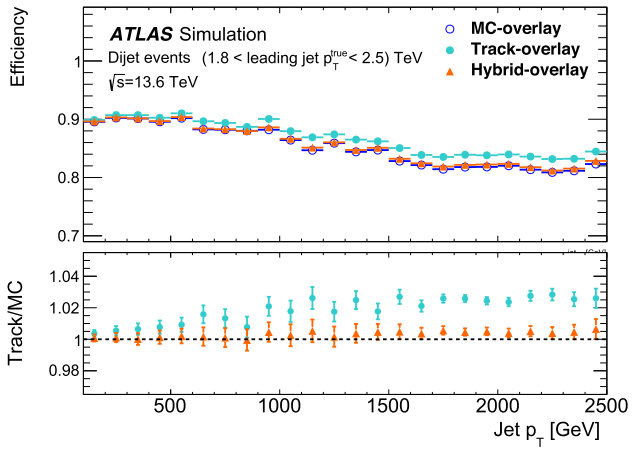


Fig. 14 The hard-scatter event track reconstruction efficiency for tracks in jets as a function of the jet transverse momentum. Dijet events at 13.6 TeV containing a leading jet with 1.8 TeV $< p_{\mathrm{T}}^{\mathrm{true}} <$ 2.5 TeV are reconstructed with MC overlay (empty circles), pure track overlay (filled circles) or the hybrid (filled triangles). For the hybrid case, the fraction of events that is sent to track overlay is 35.3%

simulated and real particle collision data is reconstruction. Both the simulated and real data pass through the same set of algorithms, except algorithms specifically treating truth information (e.g. event generator records). The reconstruction process provides analysis object data (AOD) outputs, representing a summary of the reconstructed event. These outputs are produced in an xAOD format (see Sect. 3.4), which is readable by both Athena and ROOT, easing manipulation for subsequent user analysis. In practice, xAODs are further processed to create *derived*-xAODs (DAODs), or

derivations, before being used as input to physics analysis, as described in Sect. 4.5.

The ATLAS reconstruction software was updated for the migration to a multithreaded framework[17] for Run 3.

This required numerous changes, as outlined in Sect. 3.1.1.

The ATLAS experiment has two main categories of detector sub-systems:

- Tracking detectors (the inner detector and muon spectrometer), which measure charged particle momentum; and
- Calorimeters (LAr and Tile calorimeters), which measure energy deposited by traversing particles.

Information from these systems are used to reconstruct physics objects such as charged particle tracks [187], primary vertices [188], calorimeter energy clusters [189], electrons, photons, muons, τ-leptons, jets, and event-level quantities such as $E_{\mathrm{T}}^{\mathrm{miss}}$. These objects and their associated quantities (for example the objects' energy) are often referred to as object *collections*. A single object collection will contain a set of consistently defined objects of a given type. For example, track objects are reconstructed using inputs from the ID with various thresholds and selection criteria applied. While a standard track collection is sufficient for many physics analyses, searches for long-lived particles may require a track collection containing large-radius tracks (see Sect. 4.4.2).

The reconstruction of physics objects and event-level quantities during proton–proton physics data-taking is described in Sects. 4.4.1–4.4.9. The LHC also delivers heavy-ion collisions to ATLAS during dedicated data-taking periods. The reconstruction process during these periods differs from 'standard' proton–proton collision reconstruction, and is described in Sect. 4.4.10. Special configurations of the muon and inner detector track reconstruction can also be used during the collection of cosmic ray data, particularly when no beams are circulating in the collider.

4.4.1 Calorimeter energy clusters

The calorimeter energy clustering runs first in the reconstruction of an event, as it can be used to 'seed' track reconstruction, in particular for back-tracking to recover photon conversions (see Sect. 4.4.2). Calorimeter reconstruction begins from individual cells, and is described in Ref. [167]. The digitisation process for simulated calorimeter data is described in Sect. 4.3. For real data, the energy deposited in each calorimeter cell is calculated by the online system through the application of OFCs to the time-sampled ionization pulse. The bytestream data written by the ATLAS data-acquisition system contains energies for each of the 191,720 calorimeter channels. For channels of the LAr calorimeter above a preset threshold, the cell-timing, a quality factor [190] and the raw ADC samples are also written out. The energy for these channels is then recalculated offline, allowing for update of calibration constants during offline processing.

Calibration constants required to derive the energy from the ADC samples include the *pedestal*,[18] OFCs and an ADC-to-MeV conversion factor.[19] In total, about 68 MB of calibration constants are used to calculate the energy deposit in MeV for all 182,468 LAr cells, each of which has four associated ADC values, one from each of the four samples readout from sampling the cell's ionisation pulse.

The next step of the reconstruction chain involves building a `CaloCell` container. This is a container of all cells in both the LAr and Tile calorimeters, ordered by a geometric identifier such that the same physical cell is always in the same container position. During the cell-building process, several higher-level corrections are applied. The most important of these is the correction of fluctuations (trips) of the high-voltage (HV). The actual voltage of each of the 4837 HV lines is measured by the Detector Control System (DCS) [191] and stored in a database. The offline software calculates energy-correction factors based on these voltages. The correction factors change every time the voltage changes, which can happen multiple times in a single luminosity block.[20] In serial reconstruction (as used in Run 2), IoV callbacks trigger the recalculation of correction factors at event boundaries. In multithreaded reconstruction (used in Run 3), multiple sets of sets of HV-correction factors have to be kept in memory simultaneously, because events belonging to different HV-periods are processed concurrently (see Sect. 3.3).

Each LAr cell is supplied by at least two HV lines, so there is redundancy in the case of a trip of one of the HV lines. If one line trips, the energy deposit is estimated by using an appropriate correction factor. The expected noise is then re-scaled accordingly, to account for the higher correction factor. When updating to run multithreaded reconstruction, the re-scaling of noise proved to be challenging owing to the dependence on both the conditions data and high voltage information. Input conditions data are indexed by run and luminosity block number, while the voltage is recorded by the DCS and indexed by time-stamp.

[17] The first piece of ATLAS software to work in a multithreaded environment was the calorimeter energy clustering, which was used as demonstrator to showcase the feasibility of the transition.

[18] The baseline signal level of each LAr channel and the noise characteristics of the associated readout electronics is called the *pedestal* and is measured in dedicated *pedestal runs*.

[19] The ADC-to-MeV conversion factor includes the amplification of the readout electronics, which is regularly re-measured in pulser runs for each channel.

[20] The high frequency of HV changes is partly due to imperfect smoothing by the Detector Control System.

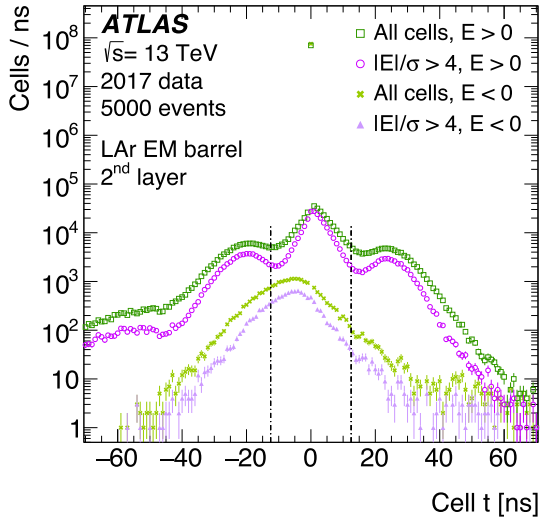


Fig. 15 The reconstructed signal time distribution for calorimeter cells in the second layer of the LAr EM barrel calorimeter. Both the inclusive cell time spectrum (green empty squares and solid crosses) and the one for seed candidates with signal-over-noise ratio greater than four (empty circles and solid triangles) are shown, separately for cells with positive and negative reconstructed energy. The vertical lines represent the cell time rejection limits of ±12.5 ns. Figure from Ref. [192]

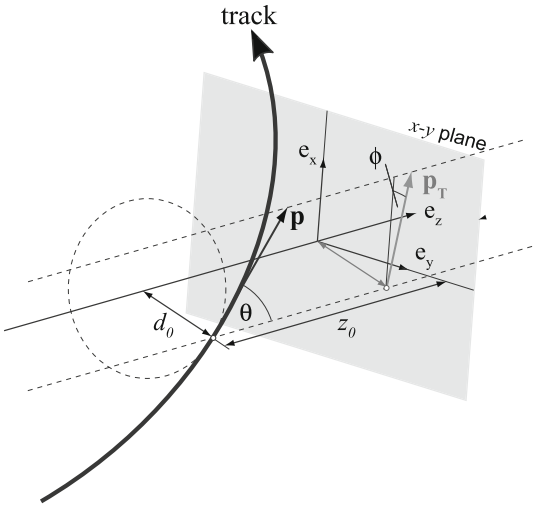


Fig. 16 Illustration of the five global track parameters, d_0, z_0, ϕ, θ, q/p [193]. These are defined relative to a reference point, the perigee, or point of closest approach to the beam line

Once built, `CaloCells` are gathered into clusters that contain the shower induced by a traversing particle. For Run 3 the topological clustering algorithm [189] is most commonly used to complete this task. Input cells are classified according to their signal-over-noise ratio, which depends on the high-voltage-corrected noise for each cell.

A reconstruction time requirement is applied to cells to reduce the effect of out-of-time pile-up [192]. The effect of this requirement is shown in Fig. 15 [192]. In the time distribution of positive energy cells, secondary contributions are visible at ±25 ns, one bunch crossing from the collision of interest. These are out-of-time pile-up contributions that are suppressed thanks to the calorimeter energy cluster time requirement.

The topological clusters, or *TopoClusters*, output by this algorithm serve as seeds for electron and photon reconstruction, and are provided as inputs for jet-building.

4.4.2 Inner detector tracks

Precise measurements of charged-particle trajectories are vital to a successful physics programme. Charged particles passing through the ID deposit energy through processes such as ionisation and radiative loss, and their trajectories can be reconstructed to form tracks. Energy deposits in neighbouring detector channels are clustered to form three-dimensional space points referred to as *hits*. Neural networks are used to refine the estimates of where the particles pass through active material, and to split individual clusters if they were produced by several coincident particles. During reconstruction, tracks are constructed as a set of hits identified as being consistent with coming from the same single charged particle, and then fit to determine the corresponding particle trajectory.

Particle trajectories are described by five parameters (d_0, z_0, ϕ, θ, q/p) defined relative to a reference point (the perigee). These parameters are depicted in Fig. 16 [193]. Here d_0 and z_0 are the transverse and longitudinal impact parameters, ϕ and θ are the azimuthal and polar angle, and q/p is the charge divided by the momentum. The default reference is the beam line centred at the beam spot [188].

Improvements were made to the ATLAS track reconstruction software ahead of Run 3 to prepare for proton collisions with an average pile-up of about 50 [194]. Such a busy environment demands highly performant software capable of efficiently exploiting available computing resources to provide prompt reconstruction of particle collision events. The main changes to the ATLAS track reconstruction software in preparation for Run 3 are documented in Ref. [194].

The track reconstruction strategy followed by the ATLAS experiment pivots on a single *inside-out* track reconstruction sequence requiring eight hits to form tracks with $p_{\mathrm{T}} > 500$ MeV. Additional inside-out and outside-in or *back-tracking* sequences targeting specific signatures such as long-lived particles (LLP) and photon conversions complement the main sequence.

An inside-out sequence begins with a *seeding* process that forms triplets from the individual silicon detectors. Then, a combinatorial Kalman Filter [195] extends the seeds along search roads through the detector elements creating multiple track candidates to maximise reconstruction efficiency. Limits on hit usage and sharing, which adapt to local conditions using a set of neural networks [187], are enforced through

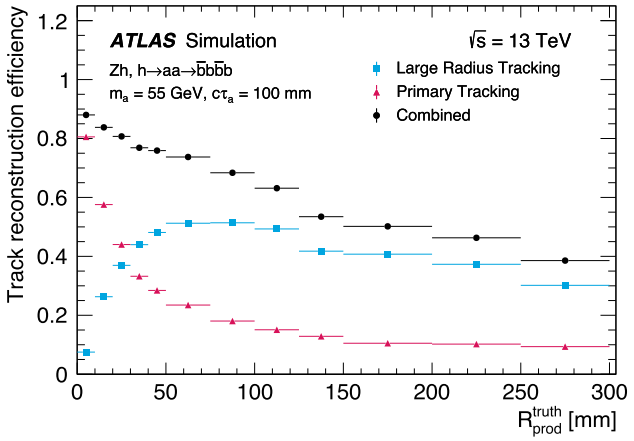


Fig. 17 The primary, LRT, and combined track reconstruction efficiencies for displaced charged particles produced by the decay of a LLP in a Higgs portal signal model. Efficiencies are shown as a function of true decay position (which is the charged particle production position) in $R^{\text{truth}}_{\text{prod}} = \sqrt{x^2 + y^2}$. Truth particles are required to have $p_{\text{T}} > 1.2$ GeV and $|\eta| < 2.5$. Figure from Ref. [197]

the ambiguity-solving stage to maintain a low rate of false positive track assignments (a low 'fake rate'). Next, surviving track candidates are passed to the global χ^2 method for a high-precision track parameter estimate. Finally, tracks are extended when possible, with TRT measurements improving momentum resolution and particle identification.

Subsequent inside-out and back-tracking sequences are tuned to particular signatures and do not consider hits already associated with reconstructed tracks. For example, large-radius tracking (LRT) considerably improves the efficiency for reconstructing the decays of LLPs displaced by more than 5 mm transversely from the interaction point by increasing the allowed range for the track candidate impact parameters [196]. Stricter selection criteria combat the increased combinatorics to ensure a fast processing time per event, while a back-tracking sequence maintains the secondary track and photon conversion efficiency. The back-tracking is seeded from TRT and SCT segments built in a narrow geometric cone (a *region of interest*) around a calorimeter energy cluster. During Run 2, LRT was only applied to a subset of events, but the performance improvements in both the standard and large-radius track reconstruction in preparation for Run 3 have resulted in the ability to run LRT on all events, thanks to computational speed-ups of a factor of 40–60, depending on $\langle\mu\rangle$ [197]. This development provides a potential boost in accessible phase-space for LLP searches, and increases the efficiency of LLP reconstruction and analysis workflows. The improvement in efficiency is shown for the case of a Higgs portal signal model in which long-lived pseudo-scalar bosons (a) from the Higgs boson decay into charged particles is shown in Fig. 17 [197].

Specific sequence configurations target particles with $p_{\text{T}} < 500$ MeV separately in low and high pile-up events. Heavy ion collisions warrant a specialised configuration as well (see Sect. 4.4.10).

The main inside-out track reconstruction iteration reconstructs primary tracks with an isolated track efficiency of approximately 70 to 85% depending on the track p_{T} and η [198]. Following high-quality selection criteria, the rate of reconstructed tracks remains extremely linear up to high $\langle\mu\rangle$, suggesting the rate of spuriously reconstructed tracks remains at a sub-percent level on average for typical operating conditions. Within a jet with p_{T} of 1 TeV, where very high measurement densities make track reconstruction challenging, reconstruction efficiencies are maintained at approximately 90% of that for isolated tracks. Photon-conversion reconstruction efficiency is around 70% over a broad range of E_{T}, with some degradation as a function of $\langle\mu\rangle$ [199]. Low pile-up reconstruction is discussed in Ref. [200]. The track-based alignment of the detector sensors is described in Ref. [201].

4.4.3 Primary vertices

Primary vertices (PV) represent the reconstructed position of an individual particle collision. For a given bunch crossing the PV with the largest sum of track p_{T}^2 is labelled the *hard-scatter vertex*. This identifies the location of what is considered to be the primary event and the interaction of interest for the given bunch crossing. The location of the hard-scatter vertex also serves as a reference point for downstream reconstruction algorithms.

PVs start as candidate vertex locations estimated by using a Gaussian-distributed resolution model for the track impact parameters. During the track fit, candidate vertices compete for tracks to reduce the chance of nearby proton–proton interactions being reconstructed as a single merged vertex. For Run 3, the primary offline vertex reconstruction in ATLAS is done using the open-source, experiment-independent ACTS toolkit [202,203]. The higher instantaneous luminosity in Run 3 means that primary vertex reconstruction is more challenging than ever before. To prepare for this, two new tools were developed to preserve primary vertex reconstruction efficiency [204]. A Gaussian track density seed finder (GS) and adaptive multi-vertex finder (AMVF) replace the iterative vertex finder (IVF) [188] used during Run 2. For the beam spot determination IVF is still used, and once the beam spot is known AMVF is used.

PV reconstruction performance depends on the event topology and is described in Ref. [198]. In Standard Model $t\bar{t}$ events, the reconstruction efficiency for the vertex including the $t\bar{t}$ production is close to 100% in Run 3 conditions. For the softer pile-up vertices, the reconstruction acceptance is around 70%, and the reconstruction efficiency is around 80% at $\langle\mu\rangle = 60$. The identification efficiency in $t\bar{t}$ events (i.e. efficiency for correctly identifying the $t\bar{t}$ production collision

as the hard scatter) is around 95% at $\langle\mu\rangle = 60$ [204]. The vertex position resolution along the beam line is estimated to be 18 μm in $t\bar{t}$ events and is somewhat degraded for lower multiplicity processes.

4.4.4 Electrons and photons

The process by which electrons and photons are reconstructed at ATLAS is described in Ref. [199]. Electron and photon reconstruction begins from calorimeter TopoClusters. In order for a TopoCluster to be considered, at least 50% of the total TopoCluster energy must be from cells in the EM calorimeter. The track parameter estimates for electron candidates are improved by a Gaussian Sum Filter (GSF) [205], which accounts for significant electron energy loss due to bremsstrahlung interactions with the detector material. The inner detector tracks with the GSF applied are also used to build photon-conversion vertices. Both the conversion vertices and the refitted tracks are then matched to the selected TopoClusters.

Following this track–cluster matching process, electron and photon *superclusters* are built. The electron and photon superclustering algorithms [199,206] replace previously used sliding-window algorithms [207] (which resulted in fixed-size clusters) in favour of the creation of dynamic superclusters of variable size. This improves the ability to account for radiation loss in the ID due to bremsstrahlung by enabling the recovery of low-energy photons, which are then paired with their associated electron or converted photon via the supercluster. First, seed clusters for electron and photon supercluster building are identified, and following this satellite cluster candidates are identified from nearby clusters. Positional corrections are applied to the resulting superclusters, and ID tracks are matched to electron superclusters, while conversion vertices are matched to photon superclusters. Track-matched superclusters form electrons, superclusters matched to conversion vertices form converted photons, and superclusters with no conversion vertex or track match form unconverted photons.

In addition to the improvements brought by switching to use dynamic clustering for electron and photon reconstruction, optimisations and simplifications to reconstruction software in preparation for Run 3 resulted in significant performance gains. For example, the use of a dedicated track extrapolation vastly reduced the CPU consumption of one of the most demanding algorithms in the electron reconstruction chain. The algorithm time was reduced by more than an order of magnitude, and the overall electron and photon reconstruction time was reduced by more than 25%. The GSF was also improved and sped up by about a factor of two. Optimisation and tuning of conversion identification was necessary for Run 3 to address the new gas configuration of the TRT, as discussed in Sect. 2.2. To improve support for analyses focusing on long-lived particles, an electron collection built using large-radius tracks was introduced for Run 3.

4.4.5 Muons

During LS2, the muon small wheel was replaced by the NSW. The NSW deploys two detector technologies, Micromegas and sTGCs, as noted in Sect. 2.2. Both the technologies provide excellent trigger and tracking performance in high-rate environments [7].

Muon reconstruction in Run 3 starts from inside the muon spectrometer, as described in Ref. [208]. MS tracks are created, extrapolated to the hard-scatter vertex, and refitted with a loose interaction point constraint taking into account the energy loss in the calorimeter.

The first step of muon reconstruction is segment finding. A single physical muon chamber might comprise several layers of one or more detector technologies, physically attached to one another. Rather than directly reconstructing muons from the hits from all of these detectors, first the hits within each chamber are gathered into segments. In standard muon reconstruction, these segments serve as a first cleaning of electronic noise and cavern background. In standard proton–proton collision reconstruction, they are required to roughly correspond to a feasible muon trajectory.

Alignment effects are implemented using an event data model object known as an *Alignment Effect on Track* (AEOT) [209]. This object holds the list of hits it affects and the standard deviations on the constraints (angular or sagitta-based) to apply to those hits. A refitting algorithm (the `MuonRefitTool`) constructs the appropriate AEOTs for the hits used in the track, based on misalignment information stored in the conditions database and processed by a specialized tool (the `AlignmentErrorTool`), and adds them to the vector of measurements that defines the track. After a muon is identified, a track refit is then done, with the AEOTs included in the derivative matrix that enters the computation of the fit χ^2, so that their effect is directly incorporated into the minimization process.

Several different algorithms are used to identify and reconstruct muons, with separate muon collections being available in xAOD files. *Combined* (CB) muon reconstruction relies on the successful combination of an MS track with an ID track [29]. Inner detector tracks are sought in a cone around the extrapolated MS track. This strategy follows an outside-in approach where the information from the MS is used to seed the muon reconstruction, looking for confirmation in the ID detector. The *inside-out* algorithm [210] starts from tracks reconstructed in the inner detector and extrapolates them to the MS. In this strategy, information from the ID is used to seed the muon reconstruction, with confirmation later being sought in the MS detector. The ID tracks with $p_{\mathrm{T}} > 2$ GeV are extrapolated outwards, with MS hits found

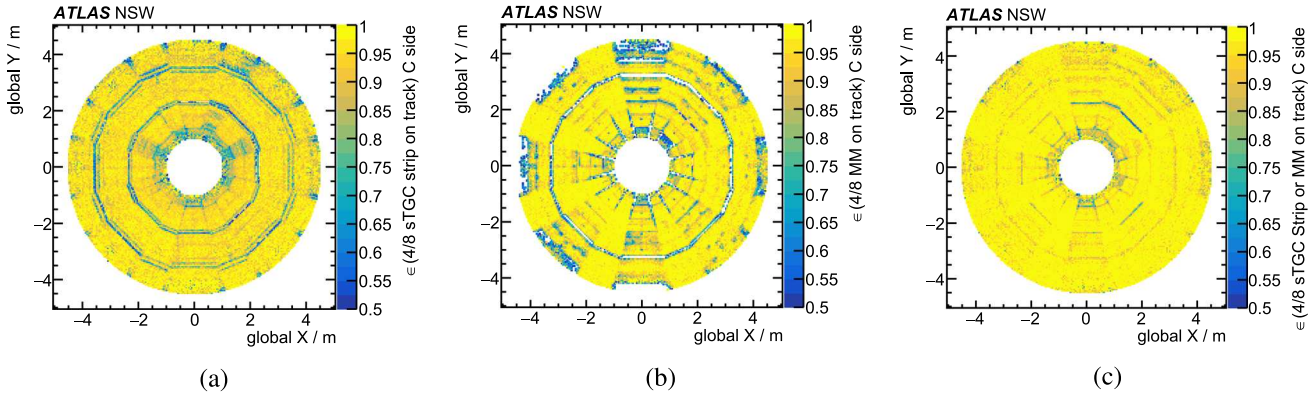


Fig. 18 Efficiency for having at least four out of eight layers of either **a** sTGC strips, **b** Micromegas strips and **c** sTGC or Micromegas strips associated with a combined (with the inner detector) or standalone (muon spectrometer only) track with $p_{\mathrm{T}} > 15$ GeV passing through the NSW on the C side during proton–proton collision data taking on May 14th 2023. Regions of low efficiency correspond to detector or readout issues during data taking. Figures from Ref. [211]

along the extrapolated trajectory being associated with the given track. For Run 3, the inside-out algorithm was significantly sped-up by removing ID tracks from consideration if they had already successfully created a CB muon. Lower-p_{T} muons that do not have enough energy to traverse the entire MS may be reconstructed as a *Segment-Tagged* (ST) muon, which does not require a fully reconstructed muon track, but only muon segments associated with an ID track. Finally, *Calo-Tagged* (CT) muons are identified selecting ID tracks matched with a calorimeter energy deposition compatible with a minimally ionizing particle. A new addition to the software with the start of Run 3 is the `CaloMuonScore` algorithm. This extrapolates particle tracks from the ID to the MS, forms three-dimensional representations of energy deposits in the calorimeter, and runs a convolutional neural network on these energy deposit patterns to assign a likelihood score corresponding to whether or not the particle is a muon.

The NSW is still being commissioned. It is fully integrated into the offline reconstruction, and standard data analysis using Run 3 data now includes muons with NSW hits. The efficiency for having a NSW segment (at least four of eight layers with either technology) associated with a combined (with the inner detector) or standalone (muon spectrometer only) track with $p_{\mathrm{T}} > 15$ GeV is shown for a 2023 data taking run in Fig. 18 [211].

4.4.6 *Particle flow and jets*

The reconstruction of jets, collimated groups of hadrons emerging from the proton–proton collision, makes use of iterative clustering algorithms, taking various detector signals as inputs. The most commonly used algorithm is the anti-k_t [212] algorithm. The jet-finding inputs, or constituents, may be TopoClusters, particle flow objects [213], particles from an event generator (for simulated samples) or any other object representable as a momentum four-vector. The clustering algorithms themselves use the FASTJET [214] implementation, meaning the constituents must be translated from ATLAS data types into FASTJET `PseudoJet` objects for interfacing.

The particle flow procedure was updated ahead of Run 3. To exploit the fact that particle flow objects are built from the same calorimeter TopoClusters and inner detector tracks as other objects, links – known as global particle flow (GPF) links – between the particle flow objects and electron, photon, muon and τ-lepton objects were introduced. By default, particle flow uses all tracks, satisfying some basic quality criteria and p_{T} requirement, and TopoClusters. The tracks in this context are assumed to be pions, and the energy in TopoClusters corresponding to extrapolated charged particle tracks is subtracted from the particle flow objects on that basis. The introduction of GPF links allows users to revisit this decision, since the list of TopoClusters matched to each track (along with their subtracted energies) is available from the charged particle flow objects.

The first step of jet reconstruction is to prepare the jet constituents, which may entail procedures such as calibration or filtering, e.g. to suppress the impact of pile-up. This is executed by a `JetConstituentModSequence`, which runs a configurable series of Tools that modify the collections of constituents. These Tools each implement the `IJetConstituentModifier` interface.

Next, a collection of `PseudoJet` objects is constructed from the constituents, such that they can be input into FASTJET for clustering. This is done by a `PseudoJetAlgorithm`, which reads in the constituent collection, creates a corresponding collection of `PseudoJets`, and records them to the event store. Any constituent type that implements the

IParticle interface (i.e. has a momentum four-vector) can be used.

The interfacing to FASTJET and translation of the resulting jets into the xAOD Jet type is handled by a JetClusterer. This is a Tool that reads in the collection of PseudoJets, runs the actual clustering algorithm, and builds a JetContainer from the output. The JetClusterer does not necessarily record these jets to the event store, as further modifications to the collection may be desired first. Such modifications (e.g. calibration, filtering, and calculation of additional variables beyond the four-momentum) are carried out by tools that implement the IJetModifier interface. Usually, these modifications must be done before the jet collection is recorded to the event store to preserve const-correctness for thread safety. An IJetModifier that only adds new variables to jets as decorations (as opposed to modifying existing quantities) implements the IJetDecorator interface, which is derived from IJetModifier but guarantees that the jets are treated as const. This permits safely running IJetDecorators on the jet collection after it is recorded to the event store, which is useful for downstream operations such as flavour tagging or analysis-specific jet variable calculations.

Jet grooming is a technique that involves selectively modifying the constituents of a jet, and can include procedures such as trimming [215] and pruning [216]. Jet grooming requires a more specialized treatment compared with other jet modifications, as it generally requires that the clustering procedure is repeated, with further interfacing with FASTJET. A dedicated JetGroomer class is provided to help this. This is a Tool that takes as input the ungroomed jet collection and its input PseudoJet collection, and outputs a PseudoJet collection representing the groomed jets (which are then translated into a new JetContainer). Because many different grooming techniques exist, a different class derived from JetGroomer is provided for each one that implements the specifics of that method.

To simplify the jet reconstruction procedure for the user and streamline configuration, an Algorithm (JetRecAlg) is provided that wraps some of the previous steps. It has three steps: first, creating a jet collection using an IJetProvider; second, running an arbitrary sequence of IJetModifiers on it; and third, recording the final collection to the event store. IJetProvider is a unified interface for all Tools that create a jet collection. This includes JetClusterer and JetGroomer as well as JetCopier, a Tool that provides a jet collection by copying an existing one.

The same jet reconstruction code runs as part of the reconstruction and derivation-making (see Sect. 4.5). In practice, the jets that are used for data analysis are built during the derivation step. Jet-building can also be done even further down the processing chain as a part of an analysis, assuming the input constituents are available. This might be done for analyses that do some jet reclustering, for example. The Algorithms and Tools used for jet reconstruction are sufficiently flexible to allow this.

4.4.7 Missing transverse momentum

Missing transverse momentum ($E_{\mathrm{T}}^{\mathrm{miss}}$) is a valuable observable that is used widely in particle physics analyses. This observable serves to approximate the transverse momentum carried by particles that remain undetected as they traverse the detector. It is essentially the negative vector sum of the transverse momenta of all objects in an event, but the calculation is sensitive to the definitions and calibrations of these objects, and the treatment of overlaps between them. This makes the reconstruction of $E_{\mathrm{T}}^{\mathrm{miss}}$ [217] particularly complex. Object definitions can vary considerably, depending on how individual analyses optimise and prioritise their object selections. For example, a set of tracks matched to calorimeter TopoClusters may be considered as an electron in one analysis, but as part of a jet in another analysis, meaning different calibrations would be applied in the two cases. To accommodate the wide spectrum of analysis needs, missing transverse momentum is only computed during the data analysis, according to its own event description. At the reconstruction step, therefore, the aim is to ensure that the objects required to permit this calculation during the analysis are provided with the necessary information. To make this possible, a compact representation of all possible overlaps between objects in the event is required to avoid double-counting in the $E_{\mathrm{T}}^{\mathrm{miss}}$ calculation. This is implemented in the MissingETAssociationMap [218], which provides all information needed to compute the $E_{\mathrm{T}}^{\mathrm{miss}}$ using any arbitrary object selection. Jets are used as the basis objects for this representation: the map consists of a MissingETAssociation object for each jet in the event, plus one 'miscellaneous' association to capture detector signals not associated with any jet. Each MissingETAssociation object contains information about which other objects share which detector signals with that jet, and with each other. The reconstruction of $E_{\mathrm{T}}^{\mathrm{miss}}$ is therefore divided into two steps: the construction of the association map, and the usage of this association map to compute the final $E_{\mathrm{T}}^{\mathrm{miss}}$ based on the object selection definitions of the given analysis.

Constructing the association map amounts to calculating the overlaps between all objects that could potentially go into the final event description. This is done for each type of reconstructed object sequentially using METAssociator Tools. Each type of object (jets, electrons, muons, photons, τ-leptons, and 'soft' detector signals – signals that too low-energy to be associated with a reconstructed object) has a class derived from METAssociator that extracts the corresponding detector signals and computes the appropriate

overlaps. These overlaps are recorded in the association map, and the map is recorded in the event store.

The second step of computing the final missing transverse momentum is carried out using the `METMaker` Tool. This takes as input the association map and reconstructed objects with analysis-specific selections applied. First, a 'term' of the overall vector sum is constructed for each type of reconstructed object by sequentially providing the corresponding reconstructed object collections. The order in which these are provided defines a priority ordering for overlap removal, and only objects satisfying this overlap removal are included in the $E_{\mathrm{T}}^{\mathrm{miss}}$ sum. Keeping track of the overlaps during this process is done using a `MissingETAssociationHelper` object, which implements a transient thread-local cache recording this information. Updates to ensure that this is a thread-safe process were made ahead of Run 3. Lastly, the `METMaker` adds each individual overlap-removed term to compute the final result.

This second step is applied by each analysis, downstream of the standard reconstruction, in order to account for analysis-specific selections. Analyses are thereby able to determine whether some object should be considered an electron, photon, or jet, for example, or whether a muon is sufficiently well reconstructed to be included, and to use exactly those objects with their corresponding calibrations in the $E_{\mathrm{T}}^{\mathrm{miss}}$ definition. The analysis tools described in Sect. 8.1.4 help ensure that uncertainties on these objects are correctly propagated into the $E_{\mathrm{T}}^{\mathrm{miss}}$, and the $E_{\mathrm{T}}^{\mathrm{miss}}$ tools provide an additional calibration and uncertainty on the soft detector signals described above.

4.4.8 Flavour tagging

Flavour tagging [98] is the process of distinguishing jets that contain b- and c-hadrons from those that do not. This procedure is applied in two stages. In the first stage, low-level algorithms are used to reconstruct the features of jets resulting from heavy-flavour decays. These features are then used as inputs to multivariate classifiers, which constitute the set of higher-level algorithms used in the second stage of the flavour-tagging process. During the flavour-tagging process, multiple jet collections can be tagged simultaneously either during reconstruction or derivation-making (see Sect. 4.5). The primary purpose of flavour tagging, when applied during reconstruction, is to enable the monitoring of flavour-tagging outputs for data quality monitoring purposes [43] (see Sect. 5.1). Flavour tagging is applied to jets built from both particle flow objects and calorimeter TopoClusters during reconstruction, but the results of this process are not saved to the output xAOD file. The flavour tagging process is then applied to particle flow and variable-radius jets during derivation-making, where outputs are saved in the output DAOD file.

Flavour tagging relies on simulated samples to enable the training of the tagging algorithms, or *flavour taggers*. In simulated samples, jet flavour *labels* are assigned according to the presence of a hadron from the event generator within $\Delta R_y(\text{hadron, jet}) = 0.3$ of the jet four-momentum. If a b-hadron is found, the jet is labelled a b-jet. Without a b-hadron, if a c-hadron is found, the jet is called a c-jet. If no b- or c-hadrons are found, but a τ-lepton is found in the jet, it is labelled a τ-jet. Otherwise the jet is labelled a *light*-jet. This information is used for training the flavour taggers to efficiently identify b-jets and sufficiently reject background jets.

The first step of the flavour tagging process is to associate tracks to the jets in an exclusive way. This is done by setting a maximum angular separation, ΔR_y, between the jet and track four-momenta. Given that b-hadrons with a higher p_{T} will decay to form more collimated jets than those with lower p_{T}, the maximum allowed angular separation between the tracks and the jets decreases as a function of jet p_{T}. The precise relation used is $\Delta R_y < 0.239 + \exp(-1.22 - 0.0164/\mathrm{GeV} \times p_{\mathrm{T}})$, with $\Delta R_y \sim 0.45$ for jets with $p_{\mathrm{T}} > 20$ GeV and $\Delta R_y \sim 0.25$ for jets with $p_{\mathrm{T}} > 200$ GeV. If a track is within the allowed distance from more than one jet, it is assigned to the jet with a smaller $\Delta R_y(\text{track, jet})$.

After track-jet association, flavour tagging is done on each jet independently. It is possible to use multiple taggers without the need to create separate containers for the results of each. Associated tracks are used to reconstruct secondary vertices, using two algorithms: `JetFitter` [219] and `SV1` [220]. The parameters describing the secondary vertices from heavy flavour decays, together with the hit content and impact parameters of the associated tracks relative to the primary vertex, are used by higher-level algorithms to compute the final flavour-tagging discriminant. These higher-level algorithms include binned-likelihood estimators `IPxD`, and several neural networks that are trained using a PYTHON software stack that runs outside of Athena [221] and is based primarily on NUMPY [222], HDF5 [223], KERAS [106] / TENSORFLOW [105], PYTORCH [107], PYTORCH LIGHTNING [224], and DEEP GRAPH LIBRARY [225]. HDF5 datasets to quantify systematic uncertainties and for training are written directly from DAODs, using Athena (or one of the lighter projects described in Sect. 6.1.1), and are used as input to the PYTHON-based training software. Trained networks are then loaded using the ONNX RUNTIME library or LWTNN for inference within Athena.

4.4.9 τ-leptons

The τ-lepton reconstruction algorithms focus on hadronic τ-lepton decays and the associated visible decay products, which form a $\tau_{\text{had-vis}}$ candidate. Several improvements were

made to τ-lepton identification, classification and background rejection algorithms for Run 3, largely driven by the development of neural-network based approaches. These approaches are discussed in Ref. [99], alongside a detailed description of hadronic τ-lepton reconstruction, identification and calibration for Run 3.

Hadronic τ-lepton decays display characteristic displacement, multiplicity and kinematic properties, with a branching ratio of approximately 65%. To reconstruct $\tau_{\text{had-vis}}$ objects corresponding to the visible decay products of a hadronically decaying τ-lepton, $\tau_{\text{had-vis}}$ candidates are seeded by jets, reconstructed from calorimeter TopoClusters, using the anti-k_t algorithm with a radius parameter $R = 0.4$. A τ-lepton will travel away from the interaction point before decaying. To identify the production vertex for each $\tau_{\text{had-vis}}$ candidate, a dedicated τ-vertex association algorithm is used [95]. The identified vertex serves as the basis of the coordinate system in which τ-lepton identification variables are calculated. The track parameters of tracks associated with the $\tau_{\text{had-vis}}$ candidate are recalculated relative to the τ-lepton production vertex.

The tracks associated with a $\tau_{\text{had-vis}}$ candidate are processed by a track classifier, and then categorised as either τ tracks (i.e. corresponding to charged τ-lepton decay products), conversion tracks (from electrons and positrons from photon conversions), isolation tracks (likely originating from quark- or gluon-initiated jets) or fake tracks (misreconstructed tracks or pile-up tracks). While a BDT-based method for τ-lepton track classification was used during Run 2, for Run 3 a novel method that uses a recurrent neural network (RNN) was developed [99].

A separate τ-lepton identification algorithm was developed during Run 2 to distinguish true $\tau_{\text{had-vis}}$ from misidentified $\tau_{\text{had-vis}}$ originating from quark- and gluon-initiated jets, resulting in significant improvement in the rejection of mis-identified $\tau_{\text{had-vis}}$. The algorithm uses an RNN architecture that is the same as that described in Ref. [226].

Electrons can be mis-identified as $\tau_{\text{had-vis}}$ objects, especially in the case of 1-prong $\tau_{\text{had-vis}}$ candidates. A new electron veto algorithm [99] for 1-prong and 3-prong $\tau_{\text{had-vis}}$, also based on an RNN, was developed for Run 3. This new algorithm improves electron rejection by approximately a factor of three for a given $\tau_{\text{had-vis}}$ efficiency compared with the BDT-based veto used during Run 2 [227].

A DeepSet neural network (DeepSet NN) [228] algorithm was developed for Run 3 to both classify the decay modes and calculate the visible four-momenta of reconstructed τ-lepton candidates [99]. This algorithm can significantly improve the energy calibration of the reconstructed $\tau_{\text{had-vis}}$ candidates relative to the BDT-based algorithm used during Run 2 [229], improving the resolution by almost 50%. The algorithm exploits the reconstruction of individual charged and neutral hadrons of the τ-lepton decays using decay kinematics, track impact parameters and calorimeter energy cluster properties for τ-lepton tracks, nearby conversion tracks, π^0 candidates and *photon shots* (local energy maxima in the first layer of the electromagnetic calorimeter, associated with photons from the candidate decay of a π^0).

4.4.10 Heavy ions

Heavy ion collision reconstruction differs from the standard proton–proton reconstruction due to the large number of underlying event (UE) particles produced in the ion collisions. The standard calorimeter energy clustering in heavy ion reconstruction begins from $0.1 \times \frac{\pi}{32}$ 'towers' assembled from geometric combinations of cells, rather than from the cells themselves [230]. The same algorithms as described above are used to reconstruct tracks in heavy ion events, but often different, somewhat more restrictive requirements are applied during physics analyses to reduce fake rates. For heavy ion data-taking the jet, electron and photon reconstruction sequences are run only after the UE is 'removed' from the calorimeter. During this process the energy deposition from the UE is subtracted from the energy of reconstructed calorimeter energy clusters, and containers for these subtracted clusters are created. The UE subtraction process and subsequent heavy ion jet reconstruction procedure is detailed in Refs. [230,231].

The average energy density associated with the UE is calculated as a function of η using all calorimeter layers for jets and per-layer for electrons and photons. It is modulated in ϕ to account for anisotropic flow present in heavy ion collisions and corrected for detector non-uniformities. This calculation excludes *seed jets* from the average, where these seed jets are jets reconstructed using the anti-k_t algorithm with energy above a given threshold. From this calculation the UE is subtracted from the seed jets. The energy calculation, exclusion and subtraction procedure is applied iteratively using different jet definitions. Each of these iterations proceeds according to the following steps:

1. Identification of seed jets.
2. Computation of UE parameters [230].
3. Application of the UE subtraction to the towers in each jet, and recalculation of the jet four-momenta.
4. Application of the energy scale calibration procedure.

The resulting jets are UE-subtracted and fully calibrated, permitting their use in the construction of seed jets in the next iteration. Three iterations are done in total for jet, electron and photon reconstruction. The criteria for seed jets built with anti-k_t radius parameter R at each iteration are as follows:

1. Seed jets are built using $R = 0.2$ and must have a maximum tower $E_{\mathrm{T}} > 3$ GeV and a ratio of maximum tower E_{T} to mean tower E_{T} greater than 4 (all without UE subtraction applied).
2. Seed jets are built using $R = 0.2$ and must have calibrated $p_{\mathrm{T}} > 25$ GeV after subtracting the UE, as determined during the first iteration. Track jets with $p_{\mathrm{T}} > 7$ GeV can also be included.
3. For the final iteration, seed jets are built using $R = 0.4$ and must have calibrated $p_{\mathrm{T}} > 25$ GeV after subtracting the UE, as determined during the second iteration.

4.5 Derivations

4.5.1 Definition and role of derived data

A feature common to many high energy physics analyses is the use of intermediate-sized data types at some stage of the analysis procedure. Typically, these files are derived from the output of the reconstruction (AOD in ATLAS) and may have the following features:

1. Their size is usually around a few percent to a few per mil of the input data;
2. They usually contain all of the information necessary for smearing, scaling, selection, calibration and other operations on reconstructed objects (collectively known in ATLAS as combined performance operations), and to determine the systematic uncertainties related to these operations;
3. They are used privately by physicists or groups of physicists to produce custom data files (usually ROOT ntuples);
4. They are typically modified and reproduced more than once for a given version of the input, and may be read several or many times by the analysis teams as they produce different private ntuples;
5. They may be aimed at one analysis or perhaps a group of related analyses (for example sharing the same final state).

The second point above is particularly important since an optimal understanding of the reconstruction, which feeds into the combined performance recommendations and calibrations, tends only to be achieved after many months of study of the data and MC simulation. Moreover, different domains of the reconstruction update their recommendations at different times, so it is usual practice to store all the information needed to allow the application of these recommendations during user analysis.

ATLAS defines four standard operations for building derived data files (see also Sect. 3.4):

- *Skimming* is the removal of whole events, based on some criteria related to the features of the event;
- *Thinning* is the removal of individual objects within an event, based on some criteria related to the features of the object;
- *Slimming* is the removal of variables within a given object type, uniformly across all objects of that type and all events (unlike the other operations, slimming does not depend on any event/object properties because the same variables are removed for every event and object);
- *Augmentation* involves adding information in the form of new variables (*decorations*) or new objects, which in some way summarise aspects of the reconstructed data, allowing much larger volumes of data to be dropped.

At the start of Run 2, ATLAS introduced centrally produced intermediate data files written in the xAOD format (see Sect. 3.4), formally called Derived AODs (DAODs) but widely known across the collaboration as *derivations*. These were defined and managed by the physics and combined performance groups and were usually tailored to specific analyses. In almost all cases the events were skimmed, with the skim rate varying depending on the physics analysis and whether the input events were real or simulated. At times during Run 2 there were more than 100 such formats in active use. These overlapped heavily for simulated events, leading to excessive disk space use, and managing such a large profusion of formats became increasingly difficult. Consequently, ATLAS has revised the model for Run 3, and this is described in the following sections.

Unskimmed derived data The main innovation in the Run 3 analysis model is the introduction of two new unskimmed data types written in the xAOD format known as `DAOD_PHYS` and `DAOD_PHYSLITE`. These are conceptually similar to the MiniAOD and NanoAOD of CMS [232].

`DAOD_PHYS` contains all of the object types and variables required by the combined performance tools to apply recommendations and calibrations (the *common slimming* content) and consequently is very similar to the existing DAODs, except that it is unskimmed and therefore can be used by a wide range of analyses. Despite containing all of the essential analysis variables, it is smaller than 50 (40) kB/event for simulated (data) events, which is less than 10% of the AOD size (see Sect. 5.3.5). Most of the size reduction is achieved by removing all tracks with p_{T} below 10 GeV that are not associated with leptons or jets. Further reductions are obtained by dropping most of the MC truth record in favour of summary information for the substantive truth particles (detectable leptons, gauge bosons, hypothetical new particles and b-hadrons), and also by removing trigger objects, leaving behind only the trigger decision and information indicating which offline objects were responsible for firing each trigger.

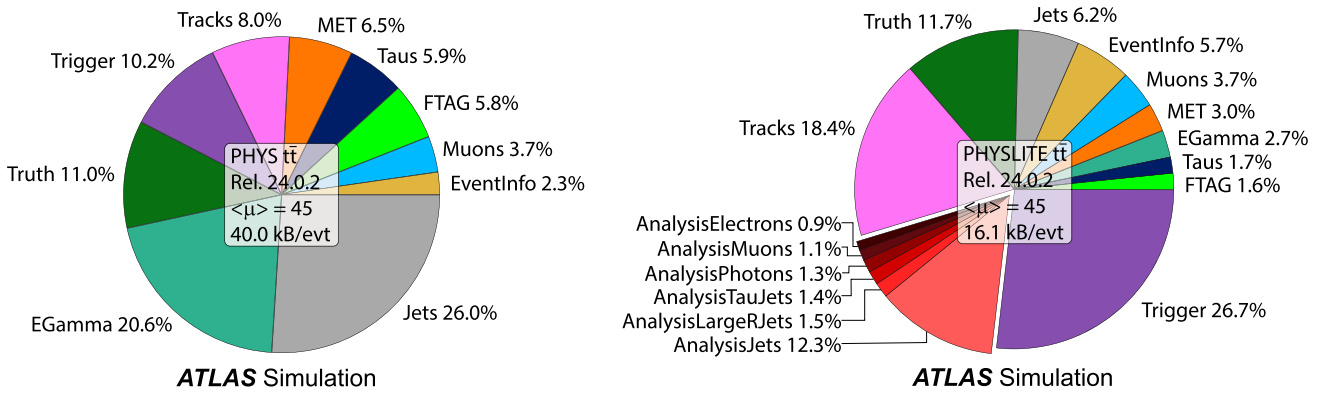


Fig. 19 The composition of a Run 3 $t\bar{t}$ sample in **a** DAOD_PHYS and **b** DAOD_PHYSLITE format. The disk size from DAOD_PHYS is dominated by the jet collections, while DAOD_PHYSLITE is dominated by the trigger information. The containers that hold the calibrated analysis objects in DAOD_PHYSLITE are depicted in red with an offset

Essentially any analysis that does not require low-p_{T} tracking information can use DAOD_PHYS, which only excludes B-physics and long-lived particle searches. Consequently, it is expected that 80% of the research output of ATLAS will be able to use the format.

DAOD_PHYSLITE is smaller still, with much of the common slimming content also dropped. Instead, the tools for applying recommendations and calibrations are run as the format is built, and the calibrated objects are written directly into the format in place of the uncalibrated objects produced by the reconstruction. The calibrations are subject to instrumental systematic uncertainties, so the central values are recorded, and the variables required to estimate the uncertainties are retained for use by analysts. DAOD_PHYSLITE is around 15(10) kB/event for simulated (data) events. Assuming that all of the calibrations are available during its production, DAOD_PHYSLITE has the same functionality as DAOD_PHYS, the only added value of the larger format being the ability to re-apply recommendations and calibrations at analysis level, which would be relevant in the event of a combined performance update. DAOD_PHYSLITE can also be produced from DAOD_PHYS, and this workflow, which is at least six times faster than the usual workflow from AOD due to the smaller size of the input and the absence of the jet and τ-lepton reconstruction (done as DAOD_PHYS is built), is expected to be highly significant in Run 4.

Figure 19 shows the composition of DAOD_PHYS and DAOD_PHYSLITE for a Run 3 $t\bar{t}$ MC simulation sample, which includes pile-up with, on average, 45 interactions. DAOD_PHYSLITE is not a fixed format; it will change over the course of Run 3, with the aim to make it as useful as possible for analysers. There are also ongoing efforts to reduce the size of DAOD_PHYSLITE further.

Skimmed derived data and event sample augmentation Many ATLAS activities require more information than is available in DAOD_PHYS(LITE). These include combined performance analyses, where the recommendations and calibrations referred to in the previous section are derived. Physics analyses requiring low-p_{T} tracks, such as B-physics and some long-lived particle searches, are also unable to use the two new data types due to the aggressive thinning of inner detector tracks. Such analyses will consequently continue to use the Run 2 analysis model; that is, they will define DAODs containing all of the variables and objects needed for their tasks, and will compensate for the bigger per-event size by removing events that are not relevant to their analyses. It is unlikely that the combined performance groups will need to process all of the data and MC simulation to determine the calibrations and recommendations. Due to these size reduction measures, and the far fewer DAOD formats, it is expected that production of these residual skimmed formats will be sustainable even during the HL-LHC era.

An inefficiency arises in this model when there is significant overlap between the additional derivations and the common DAOD_PHYS content, combined with a large skim fraction. To address this issue and reduce data duplication, ATLAS has developed a feature called *Event Sample Augmentation* [233]. This feature enables the extension of a base format (e.g. DAOD_PHYS) with supplementary data obtained from a secondary skimmed format, but only for the subset of events that meet the skimming criteria. In the Event Sample Augmentation configuration, the secondary format is incorporated as a ROOT friend tree into the base format and stored within the same file (storage in a separate file is also possible). This design allows seamless access to the supplemental data during downstream data processing, eliminating the need for redundant information to be stored to disk. Moreover, since

the augmentations are stored in separate trees, they do not compromise I/O performance for clients who only require the baseline DAOD content.

In one benchmark scenario, a long-lived particle analysis required the inclusion of extra containers on top of the baseline DAOD_PHYS content, leading to an average event size increase of approximately 40%. In the Run 2 model, this situation would cause a significant 56% rise in disk space usage due to significant redundant data. However, with the implementation of Event Sample Augmentation, incorporating the additional data for just 40% of the events results in a mere 16% increase in overall disk usage. This approach proves notably more efficient than either the Run 2 model or the inclusion of the extra content in DAOD_PHYS directly. The Event Sample Augmentation thus provides an effective solution that enables analyses to use customized data while mitigating resource impacts.

4.5.2 Derivation framework software

The software used to produce derived data products (the *derivation framework*) is built on the software used in Run 2. It is implemented within the Athena framework, and so shares many similarities with reconstruction. In fact, to save disk space in the AOD, some parts of reconstruction are normally run during derivation building. In particular, jet reconstruction (see Sect. 4.4.6) and the reconstruction of objects that depend on jets (flavour tagging, $E_{\mathrm{T}}^{\mathrm{miss}}$, and τ-leptons; see Sects. 4.4.7–4.4.9) are normally included in derivation-making. This ensures that any specialisation (e.g. for particular grooming techniques to be applied to large-radius jets) can be provided to an analysis without the need to save all possible combinations directly in the AOD. The derivation framework consists of an Athena Algorithm called a *kernel* that drives the event loop. Each derived data product implements such an Algorithm, and passes it a series of Tools for skimming, thinning and augmentation, which it calls in turn to produce the new data type. Multiple data products can be produced in the same job from the same input (*train production*), even with different skimming selections.

Alongside the other data production activities, the configuration layer of the derivation framework was recently updated to use the new Component Accumulator (see Sect. 3.2). The derivation framework can only be run as serial or multi-process (AthenaMP, see Sect. 3.1) jobs; multithreading support (AthenaMT) is in development to be deployed before Run 4.

To avoid having to merge separate output files from forked workers or processes of a derivation job, ATLAS developed a SharedWriter process that collects data objects from all workers and writes them to a single output file. In Run 2, only the SharedWriter itself would interact with ROOT for file writing, ensuring that the container entry numbers were synchronized. This means that when the SharedWriter received data from a worker, it needed to modify the data somewhat (e.g. to account for the number of events already written to the output file) before recording them. Similarly, because in Run 2 the container entry number was used as external reference, objects were not relocatable, for example by fast merging techniques. Since it eliminated the merging step (which had to be run serially), the SharedWriter approach significantly sped up derivation production. But as the single SharedWriter has to compress all data, the scalability with output data volume and the number of workers could become limiting. Therefore, a new version of the SharedWriter was developed for Run 3 to support unskimmed formats (i.e. a larger data volume) and potentially higher core counts. The new SharedWriter relies on a change to the ATLAS persistence navigational infrastructure: For Run 3, the persistent references were changed so that rather than relying on the container entry number they use a unique identifier that is stored within the container itself. This infrastructure allows fast merging without invalidating existing references. In this new design, the workers write their data to a ROOT memory file that is sent to a single SharedWriter. Thus, the worker processes can compress data in parallel, and for a moderate increase in memory consumption (due to compression buffers) a very significant speed up and much better scalability is achieved [234]. The performance improvement is also shown in Sect. 5.3.4.

4.6 Forward detectors

The four forward systems are treated specially throughout the software workflow, from simulation through to data analysis. These generally have stand-alone GEANT4-based simulation configurations for comparisons to test beam results and the understanding of detector calibrations.

LUCID-2 is used primarily for luminosity measurements [235]. Although the detector can be simulated and the signals digitised and reconstructed as a part of the standard software workflow, this is not done for most MC simulation samples. The analysis for luminosity proceeds from bespoke data formats, as individual photomultiplier tube signals are analysed. The simulation of LUCID-2 has been used most for studies of the impact of changes to the detector geometry (e.g. changes to the beampipe) on the ATLAS luminosity measurements.

Far-forward particles are normally not simulated to save CPU time. The far-forward detector systems (ZDC, AFP, and ALFA) also require simulating material and beam pipe regions not normally included in the ATLAS detector description, including some regions under the responsibility of the LHC machine group. For ALFA and AFP, forward-pointing particles are propagated through a simple simulation of the LHC optics before being injected into specialised

simulations of the forward detectors. For analysis purposes, a simple fast simulation is often used to approximately model the response of these detectors directly from the forward particle momenta. For events entering these analyses, beam divergence and crossing angles are included in the simulation.

The Zero-Degree Calorimeter (ZDC) is primarily used for triggering and analysis of heavy ion collisions. A detailed GEANT4-based simulation is used to model the response of the detector to single neutrons and neutral pions. These simulations can be computationally intensive, because the particles incident on the ZDC are often several TeV. The simulation software (including digitisation) is being reviewed to reduce its resource needs. Nevertheless, for some events the same forward-transport mechanisms as with AFP and ALFA will be used, particularly in the analysis of data from the new Reaction Plane Detector [236] (a part of the ZDC that is new in Run 3).

4.7 Phase-II upgrade support

The planned Phase-II upgrade of the ATLAS experiment for the HL-LHC requires robust, performant software several years before data-taking to study the impact of changes to the detector design, and to understand and estimate the physics performance of the new detector. The necessary developments, described in this section, were integrated into the existing Run 3 software; some major software developments are still planned before Run 4, and these are described in Sect. 9. Required features include the ability to simulate the Phase-II detector geometry, digitise the simulated HITs, and reconstruct physics objects, all including the new and improved detector systems. Once the concepts are proven and approved, the software must also support new developments and features targeting the HL-LHC to be ready for data-taking in Run 4. This is also important for the HL-LHC road map (see Sect. 9), where support for the Phase-II detector is fundamental to achieve the milestones defined on the set timescales.

There are several major changes foreseen as a part of the Phase-II upgrades [15]. The entire inner tracker will be replaced with a new system, ITk, with significantly expanded coverage in $|\eta|$. A new timing detector, the High-Granularity Timing Detector (HGTD), will be installed in the forward region. Significant improvements are planned for readout systems and trigger systems throughout the detector.

Historically, ATLAS chose to provide a dedicated 'upgrade release', branched from the main release that was used for Run 2 data-taking in early 2016. This release was decoupled from the main development branch, mainly to simplify conceptual testing and R&D projects without worrying about possible interference with on-going data-taking. Inevitably the two releases diverged to the extent that it was no longer possible to compare the current state-of-the-art data reconstruction performance and the future expected detector performance. Estimates had to rely on old algorithms no longer in use by ATLAS analysers at the end of Run 2. The infrastructure updates that took place after the upgrade release was defined also created difficulties, as the upgrade and main releases used different version control systems, build systems, for example. After the approval of all ATLAS Phase-II TDRs, the urgent need to provide simulated results ceased and significant effort could be put into integrating upgrade software into the main development branch again. The main release now supports both the present and upgrade detectors, providing a solid ground for future developments working towards Run 4 data-taking, while keeping up-to-date with advancements made during Run 3.

During the conceptual design process of the Phase-II detectors, the layouts rapidly changed, which meant the need to implement new or modified geometries quickly in simulation. The framework for building the ATLAS geometry (GeoModel, see Sect. 3.5) is robust but somewhat cumbersome, and not designed for quick iterations. A new tool to easily build the geometries via an XML format, called GeoModelXML, was developed for building the ITk Strip geometry. This tool proved useful and easier to work with, and was adopted by a large fraction of the sub-detectors of ATLAS to prepare for the HL-LHC.

Another major challenge was to incorporate into the software the new HGTD, which will replace the Minimum Bias Trigger Scintillators (MBTS) during Run 4. In the simulated geometry, the MBTS is situated and built within the calorimeter envelope/volume. Initially, the HGTD was added in the same place as the MBTS in the simulation framework. This came with few issues that were discovered during development. For instance, the calorimeter volume is treated differently in regards to treating the truth particle record: a significant fraction of information is not stored by default. This created issues for digitising the HGTD HITs and for performance studies. Moreover, there was no support in the EDM for the timing information to be propagated from the hits to the track particles. The time distribution of pile-up proton–proton collisions must also be modelled. Proper solutions to these problems are now in place in the main development release.

Digitisation for the Phase-II tracking detectors is heavily based on the software utilised for the present pixel and SCT detectors, with adaptations to the new granularity and expected operation conditions in terms of low and high voltage, and charge thresholds. This software was developed for the original detectors more than 10 years ago and was not optimised for the high pile-up conditions expected at HL-LHC, which brings with them several complications. The resource usage in terms of CPU, but especially in memory, is far from sustainable in the long-term. For example, the radia-

tion damage modelling used in the pixel detector digitisation (see Sect. 4.3.1) is satisfactory for current radiation levels, but for the radiation levels expected during the HL-LHC is far too resource-intensive. Efforts are ramping up to revamp the code and several milestones were defined in the HL-LHC roadmap to make pile-up digitisation fully multithreading compatible and to reduce the overall memory usage.

Similarly, as with digitisation, the upgrade reconstruction software is derived from the existing ATLAS algorithms for all physics object domains, although it was adapted to the larger η-coverage of the ITk. However, track reconstruction was studied in detail during several years and is the most advanced among the domains. A prototype [237] already meets the HL-LHC CPU resource requirements.

5 Software integration, evaluation, and validation

Before any real or simulated data is used within physics analyses, the quality of the software and configuration used must be thoroughly vetted. The validation proceeds through several steps, only some of which are shared between data and MC simulation. An extensive suite of data quality tools is applied to check the real data; these are described in Sect. 5.1. The same tools are often used both online (i.e. in real time as the data are taken) and offline (later, after the data have been recorded) to look for issues with the detector or the data itself, as well as reconstruction and conditions (e.g. calibration, noise masking, or alignment) problems that might arise. A separate procedure, with separate tools, is used to validate configurations of the MC simulation chain; this procedure is described in Sect. 5.2. The computing performance of the software, for example in terms of CPU time, memory consumption, and output file size, is constantly monitored. The performance is described in Sect. 5.3. When a new MC simulation or data (re)processing campaign is to be undertaken, all of these tools are used to carefully validate the software and configuration to be used. The steps required for these campaigns to be launched are laid out in Sect. 5.4.

5.1 Data quality monitoring of collision data

A detailed description of the ATLAS data quality operation and performance for Run 2 can be found in Ref. [43]. This section focuses on the software and computing support for data quality.

During data-taking the quality of recorded data may occasionally be compromised due to, for example, hardware failures, noisy detector elements, or configuration problems. To ensure only high-quality data are selected for use in downstream analysis, an extensive suite of data quality (DQ) monitoring and assessment tools are applied to each ATLAS physics run.

DQ checks begin in real time, on-site in the ATLAS Control Room (ACR). Here dedicated *shifters* are on duty around the clock to monitor the status of the detector and the data itself at various points in the data flow. Once a run has completed and the RAW data have been recorded, a two-stage offline data quality process begins [43]. The first stage takes place during the calibration loop, and the second stage makes use of the results from the prompt bulk reconstruction (which includes all promptly processed streams), as indicated in Sect. 2.4. The end product of the data quality assessment is a *good runs list* (GRL): a list containing, run-by-run, all luminosity blocks certified for physics analysis (see Sect. 8.1.3). Depending on their precise needs, different GRLs might be used by different analyses of the same data-taking period.

Time-dependent information about, for example, detector and trigger status or run configuration are stored in the ATLAS conditions database (see Sect. 6.2.2). DQ issues are flagged by the storing of *defects* in the defect database [238], a subcomponent of the conditions database. Defects are set for a given run according to the outcome of the data quality assessment, though they are sometimes uploaded automatically based on feedback from other supporting infrastructure. If a defect is *present* for a given luminosity block, and that defect is severe enough to prevent the inclusion of those data in physics analysis, the corresponding luminosity block is absent in the resulting GRL.

During data-taking several applications are used to monitor the status of conditions and to record those in the conditions database. The Detector Control System (DCS) [191] records the operational status of detector hardware components, including e.g. component temperatures. Another application, GNAM [239,240], monitors the detector status at several stages of the data flow and generates monitoring histograms. Events passing through the hardware trigger are monitored via the trigger system. This includes monitoring events not recorded to disk, thereby ensuring interesting events are not unexpectedly discarded. Histograms are produced at the high-level trigger computing farm to monitor HLT operation. A subset of collision events are also passed through the full reconstruction chain at this stage, to produce histograms that can be used to monitor high-level information and the quality of reconstructed physics objects. The Data Quality Monitoring Framework (DQMF) [241] provides algorithms to do automated checks on the histogram outputs. A key requirement of the DQMF is that it is lightweight enough to comply with strict time constraints for data processing. The DQMF is used in both the online and offline DQ chains, which means it must be compatible with both the online and offline software environments.

The Information Service (IS) retrieves monitoring data from the Trigger and Data Acquisition (TDAQ) [242] systems for temporary storage so that they can be shared between various applications. The IS can also temporarily

host histograms generated by GNAM and DQMF results. Histograms stored in the IS can be retrieved and rendered by dedicated display tools. The two main display tools used in the ACR are the Data Quality Monitoring Display (DQMD) [243] and Online Histogram Presenter (OHP) [244], with DQMD providing a global view of data and detector status in a structured hierarchy, and OHP allowing the display of any published histogram. Event displays (see Sect. 8.2) are also used for online DQ monitoring in the ACR, as discussed in Sect. 8.2.6.

A common framework is used to fill and manage DQ monitoring histograms in both the online software trigger and the offline reconstruction code. This framework provides an interface to users in which client code only provides the values to plot, deferring the detailed specification of the histograms themselves to the Athena runtime configuration. In this way, histograms can be added and modified easily without changing any compiled code. The user code is completely insulated from the representation of the histograms or filling, rebinning, and other operations on them. This layer of abstraction also allows the entire system to be safely multithreaded in a controlled manner as all access and changes proceed through tightly controlled APIs. This architecture will also permit straightforward migration should the histogramming backend be changed in the future. Several optimisations are provided by this code [245], such as the ability to 'vectorise' filling operations by providing many data points at once in a single library call, and providing several variables at the same time to trigger the filling of multiple histograms.

5.2 Validation

The Physics Validation Group (PVG) is tasked with investigating changes to the core code base, be it at the level of event generation, simulation or reconstruction. Since new MC or data reprocessing campaigns (described in Sect. 5.4) require significant resources (in terms of both computing and person power), validation is a crucial step. New derivation formats or major updates to the existing derivation formats also require extensive validations, but these are carried out by a different group within ATLAS, the Analysis Model group.

Specifically, PVG is tasked with understanding whether such changes affect the accuracy of the physics modelling in MC simulation and detector data events. Validation of detector data (as opposed to simulated data) is done only occasionally, for example to compare setups for reprocessings after changes to reconstruction algorithms or conditions. If changes to any physics observables are seen, PVG determines whether these changes are expected and acceptable. If the changes are unexpected then the group tracks down the causes of these issues and works with experts to understand and resolve the issue to restore a sensible physics description. The group consists of two conveners, and one or two validators for each physics object being studied: tracking, electrons, photons, muons, τ-leptons, TopoClusters, particle flow, jets, flavour tagging, and $E_{\mathrm{T}}^{\mathrm{miss}}$. Experts from each combined performance group also often take part in these validation efforts, with $\mathcal{O}(100)$ validation tasks being carried out each year. Validation itself therefore requires substantial CPU and human assets.

Validation tasks are defined starting with a request to validate a certain feature (e.g. a simulation code change). Nightly testing helps safeguard against changes from simple coding bugs or unintended consequences (see Sect. 6.1.3); validation tasks are normally created for more significant, intentional changes. A JIRA [246] ticket (see Sect. 6.1) is created with a brief description of the task, from which the PVG conveners work with the relevant experts/conveners to define the production configurations that will evaluate these changes. A reference is defined from a known and stable code base from which the changes can be validated. The test is then defined starting from the reference and changing the specific feature that is under investigation. It is common for multiple references and tests to be defined within a single validation task, to ensure that all aspects of the relevant changes are investigated (e.g. reconstruction might be tested both with and without pile-up). While normally the test are done in a stable, numbered release, occasionally physics validations are done using nightly releases (see Sect. 6.1.2) for the test configuration. For very special cases, nightlies can also be used for the reference configuration, for example when a release with a validated feature has not been built yet. This is particularly useful when development is proceeding rapidly, or a significant change must be deployed and then backed-out until it is fully validated.

Once the references and tests are defined, production configurations are created and the PVG conveners launch jobs to produce a set of physics validation samples (*Sample-A*). Sample-A is a list of about a dozen relevant MC simulation samples that effectively populate and test the different objects used to complete the validation task, including both complete physics events and single particles. For example, the jet validation usually uses both a $t\bar{t}$ sample and, separately, a high-p_{T} dijet sample. Similarly, flavour tagging will also use the $t\bar{t}$ sample, and additionally a $Z' \to q\bar{q}$ sample with a Z' mass of 4 TeV to test flavour tagging at high-p_{T} The electron (photon) validation includes samples of single electrons (photons). Each sample is produced with 100,000 events to avoid large statistical uncertainties that would make small changes hard to observe consistently.

Once the samples are ready, each group of physics validators runs specialised analysis code over the relevant samples to produce a standard set of $\mathcal{O}(100)$ histograms of quantities relevant to their respective physics object. These plots are uploaded to a webpage for easy viewing, where physics validators scrutinise the reference in comparison to the test,

together with the respective object experts. Interesting or discrepant results are presented in a weekly PVG meeting, where the conveners together with most validators and experts can collectively scrutinise and discuss the results.

Ultimately, physics validators mark the task as either green (all good), yellow (discrepancies, but may be understood or need further study), or red (clear issues). If all objects report the task to be green, then PVG signs off the task and the ticket is closed. If the task is marked by any object as yellow or red, then further follow-up is done with experts, and either the results are ultimately understood or a new reference and test are defined and the physics validation cycle repeats to look into these differences further.

To get an even broader view, every 6–12 months a round of Physics Analysis Validation (PAV) is done, in which analysis teams from the Higgs, Exotics, HDBS, Standard Model, SUSY and Top physics groups run their analysis code over a standard set of MC simulation and detector data samples, comparing a well-known reference to a recent test version which has the integrated changes from numerous validation tasks. This effort is vital as it brings even more scrutiny, crucially in numerous corners of kinematic and physics usage phase-space, using standard analysis tools. While standard physics validation covers common ATLAS data formats, PAV also ensures that validation is done for the entire workflow down to final analysis outputs.

Leading up to Run 3, there was a large and in-depth validation effort, starting from the Run 2 code base and sequentially checking both optimisations and changes due to the Phase-I upgrade. This validation effort started in October 2021 with the validation of the first Run 3 geometry tag and continued right up to July 2022 when the MC21a production campaign was launched on a large scale (see Sect. 5.4). The major milestones along the way were the validation of the various conditions updates that involved most of the sub-detectors, GEANT4 updates in terms of version and speed optimisation, MC Overlay against the standard pile-up, and the variable beam spot (see Sect. 4.2). Some of these tasks required multiple validation rounds as their outcomes were not initially in-line with expectations.

Some of the main issues and causes of delay that were experienced during this validation effort were unexpected changes between software releases and incorrect settings that led to crashes and unphysical behaviours (e.g. unreasonably large changes in performance) or other kinds of inaccuracies. In the case of the former a careful validation of the single releases was sometimes needed to find the specific changes that altered the validation chain, to restore the expected agreement.

Given the experience from the Run 3 validation effort, some improvements are explored for future large-scale validation efforts. Those include robust checks of any individual changes before validating them combinedly in a validation campaign, which can help catching problems at a low level and will reduce the number of validation iterations. Extensive low-level automated checks should be done to better track release-by-release issues that arise in production. It should also be ensured that production campaigns precisely match the configurations used for a detailed physics validation. Sufficient input from experts, who can disentangle smaller issues within large overall expected changes, will ensure nothing subtle is missed.

5.3 Software performance

5.3.1 Introduction and configuration

This section describes the computing performance of the Run 3 software. In all cases, tests are run on a bare-metal machine with two AMD EPYC™ 7302 16-core processors configured to be in the *Performance* mode with Simultaneous Multi-Threading (SMT) disabled. To disentangle its effects, clock frequency boosting is disabled and the clock frequency is capped at the base frequency of 3.0 GHz. The machine has 252 GB of available memory. In all results, memory is measured by the proportional set size, and time by the wall clock processing time unless stated otherwise.

Insofar as it is possible, the job configurations mirror what is used in the production system by ATLAS. The number of events to be processed concurrently is set to the number of available threads. The computing performance results for each step of the MC simulation processing workflow are provided in their respective subsections below. The MC simulation workloads use $t\bar{t}$-production events. In all cases, a constant number of events per thread or worker is processed (i.e. these are tests of weak scaling).

5.3.2 Monte Carlo processing chain

MC simulation As described in Sect. 4.2, during LS2 the GEANT4 Optimisation Task Force was launched with the aim of reducing the GEANT4 simulation CPU time in Run 3 by more than 30% without making any compromise on the physics accuracy. In this section, the simulation is benchmarked for the three latest ATLAS MC simulation campaigns: MC20 (the last ATLAS MC campaign for Run 2), MC21 (the first ATLAS MC Campaign for Run 3) and MC23 (the latest ATLAS MC Campaign for Run 3).

Figure 20a shows the memory usage, which scales well with the number of threads and similarly across the three campaigns. In these jobs, Athena release 22.0.92 is used, with 100 events per thread. The increase in the memory footprint is expected. It is explained partly by the different centre-of-mass energy of the input events in Run 2 (13 TeV) and Run 3 (13.6 TeV), some changes in the geometry and beam conditions and, most significantly, by the introduction of a

technical optimisation that concerns the way Athena code is linked with GEANT4 [156] (see Sect. 4.2.1, this optimisation reduces CPU in turn).

Figure 20b shows the event throughput scaling of these simulation configurations in the same benchmark jobs. The scaling is almost ideal; an extrapolation of the single-thread performance is provided for reference. In fact, ideal performance is not expected to be linear on modern processors because, for example, the core frequency is automatically changed to optimise instruction throughput against power (see, for example, Ref. [50]). The CPU hyper-threading is also expected to result in somewhat lower throughput. Overall, the three workflows scale in a similar way. The throughput increased from MC20 to MC21 by 27%–32%, and from MC21 to MC23 by 45%–47%. These improvements are the results from the CPU optimisations in MC21 and MC23 with the GEANT4 optimisations that are detailed in Sect. 4.2; a CPU speed-up of 33%–38% is observed in MC21 relative to the MC20 setup and the speed-up increases to 50% in MC23.

Overlay, trigger simulation and reconstruction for MC
Figure 21a shows the memory usage of several workloads normally run together in MC simulation production. The MC23 production configuration is used with Athena release 23.0.53. The specific MC campaign of this performance study is MC23c, which corresponds to an average pile-up of 56 interactions. The first step is MC Overlay, where the effects of pile-up are simulated by overlaying hard-scattering events with pre-mixed RDOs to model the desired $\langle\mu\rangle$ profile and other effects (see Sect. 4.3.2). The result of this step is fed into the second step, offline trigger simulation, labelled RDOtoRDOTrigger. The resulting file is then passed through the reconstruction, labelled RAWtoALL, where physics objects are created (see Sect. 4.4) and stored in an AOD. The memory usage increases by 200–400 MB per additional thread, and each 8-thread configuration stays well below the standard 2 GB per core available on the Grid.

Figure 21b shows the event throughput scaling of these MC simulation production workloads in the same benchmark job. Standard production configurations use eight threads, and thus optimisation efforts were focused on this regime. Indeed, up to eight threads the scaling is relatively linear in all of the three steps. At this point, MC Overlay reaches its Amdahl's law [247] plateau and the improvement due to additional threads is significantly reduced. This is a subject for future improvement. The same happens for offline trigger simulation above 24 threads, whereas reconstruction shows almost ideal improvement throughout (within the caveats discussed above). Nevertheless, the production setup is well-optimised for the most-used hardware configurations on standard Grid production queues that offer eight cores per job.

5.3.3 Reconstruction of collision data

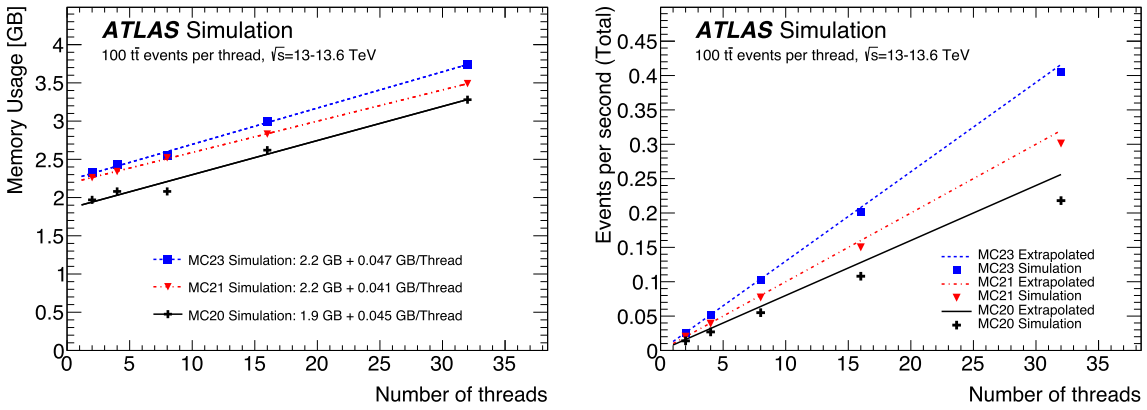
Figure 22a shows the memory usage as a function of the number of threads, using input proton–proton collision data collected in 2023. Athena release 23.0.53 is used for these tests, with 150 events per thread. The corresponding average numbers of interactions per bunch crossing, $\langle\mu\rangle = 65$. As expected, the memory usage grows linearly with the number of worker threads. However, it is always well below the standard 2 GB per core available on the Grid for 8-core configurations, meaning that data reconstruction jobs can successfully be executed on any Grid node. The memory is significantly larger than that for the reconstruction of MC (see Fig. 21a) because data reconstruction includes additional data quality monitoring tools that involves the creation of many histograms.

Figure 22b shows the event throughput as a function of the number of threads in the same jobs. For the production configuration with eight threads, the data throughput is measured to be 0.5 events per second. The scaling is close to ideal, with small deviations in the region with many threads, but the throughput continuously improves until all available cores on the node are utilised. Although there is room for improvement, the reconstruction job is no longer memory-bound in production configurations, a notorious problem in previous versions of the Athena software.

5.3.4 Derivation production

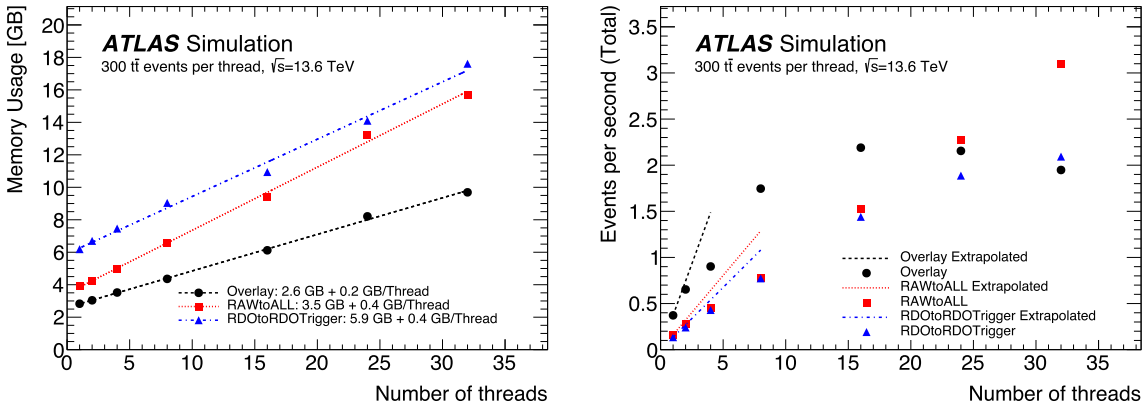
Figure 23a shows the memory usage as a function of the number of worker processes in three different multi-process derivation production configurations. These performance tests are all done with Athena release 24.0.12, mirroring production configurations as closely as possible. The data run was mostly in the range of $62 < \langle\mu\rangle < 67$. Derivation production based on simulated $t\bar{t}$-production events is shown with serial compression (see Sect. 4.5.2) and the Run 3-standard parallel compression. As expected from the additional compression buffers, the memory requirement for parallel compression is higher. The memory usage for derivation production using 2023 detector data as input is also shown. The memory usage is slightly lower for low numbers of workers because fewer algorithms run on data (e.g. those operating on generator records run only on MC simulation); the scaling with the number of workers is slightly worse owing to subtle differences in the configuration of the forking (i.e. which memory pages are fully shared between processes).

Figure 23b shows the event throughput as a function of the number of worker processes in the same jobs. 1000 events per worker are included for MC simulation, and 700 events per worker are included for detector data. Here the advantage of parallel compression is clearly visible: there is dramatically better scaling for high numbers of workers when par-



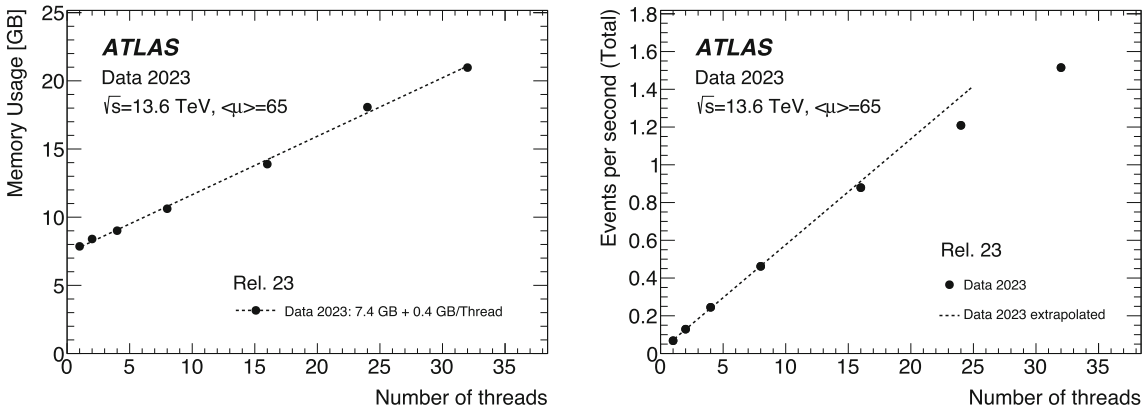
(a) (b)

Fig. 20 **a** Memory usage as a function of number of threads and **b** event throughput as a function of number of threads for three recent MC simulation campaigns. In the memory figure linear fits are superimposed, and in the throughput figure extrapolations of the single-thread results are shown to guide the eye



(a) (b)

Fig. 21 **a** Memory usage as a function of number of threads and **b** event throughput as a function of number of threads for several MC simulation workloads in MC23. In the memory figure linear fits are superimposed, and in the throughput figure extrapolations of the single-thread results are shown to guide the eye



(a) (b)

Fig. 22 **a** Memory usage as a function of number of threads and **b** event throughput as a function of number of threads for 2023 detector data reconstruction. The average pile-up $\langle\mu\rangle$ was 65. In the memory figure a linear fit is superimposed

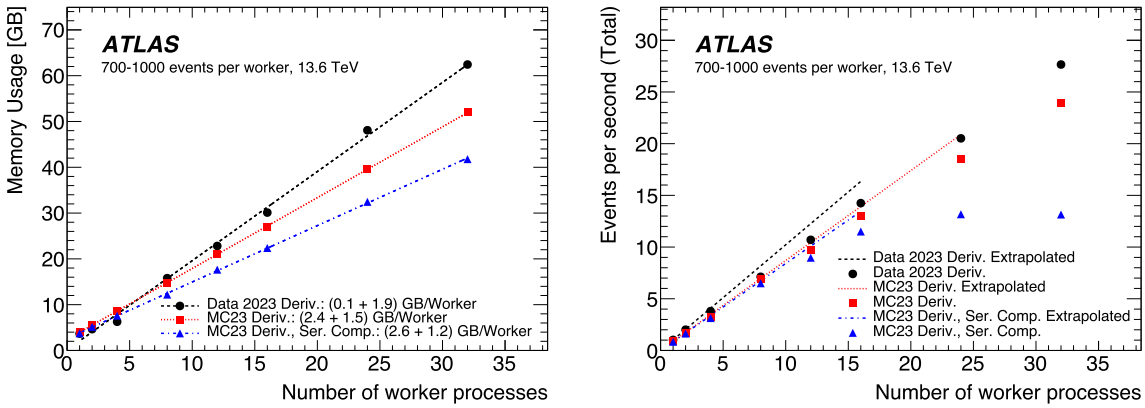


Fig. 23 **a** Memory usage as a function of number of workers and **b** event throughput as a function of number of workers for derivation production using 2023 data and MC simulation. The MC simulation workloads use $t\bar{t}$-production events. Also shown is derivation production for MC simulation with serial compression ('Ser. Comp.'). In the memory figure linear fits are superimposed, and in the throughput figure extrapolations of the single-worker results are shown to guide the eye

Table 5 Average size of an MC simulation $t\bar{t}$ event, for various MC production campaigns. The data taking year represented by each campaign and the average number of proton–proton interactions, $\langle\mu\rangle$, are reported. All the MC20 sub-campaigns use the same simulation configuration at $\sqrt{s} = 13$ TeV; the MC23 sub-campaign simulation configurations differ only by the beamspot, which has a negligible effect on the output file size. Both MC21 and MC23a represent the 2022 data conditions at $\sqrt{s} = 13.6$ TeV, but the former refers to production performed before and during data taking, while the latter represents production done in 2023 with an updated software release and conditions matching those of the data taking (rather than a prediction). The HITS files are the output of the GEANT4-based or ATLFAST3 simulation of the hard-scatter process; the RDO files are the output of digitisation and MC Overlay, and store the contribution of pile-up; and the AOD files are the output of reconstruction of objects needed for physics analysis. 'N.A.' indicates combinations that were not validated for physics analysis use. The Full and Fast Simulation HITS file sizes listed for the MC20d subcampaign are also applicable to the MC20a and MC20e sub-campaigns

Campaign	MC20a	MC20d	MC20e	MC21	MC23a	MC23c
Representing data taken in	2015–2016	2017	2018	2022	2022	2023
$\langle\mu\rangle$	23.9	37.8	36.1	44.3	42.8	55.6
Full Simulation HITS (kB/event)		906		711	620	
Fast Simulation HITS (kB/event)		801		N.A.	591	
RDO (kB/event)	1648	1897	1878	2073	2052	2240
AOD (kB/event)	271	364	354	375	348	430

allel compression is enabled. The derivation production for detector data runs slightly faster than that of MC simulation owing again to the lack of algorithms and tools running on generator records. With parallel compression in MC simulation, the scaling is very close to ideal beyond 16 workers.

5.3.5 Disk sizes of formats and containers

The average disk sizes of the standard formats for MC simulation datasets under several representative sets of pile-up conditions are listed in Table 5. From MC20 to MC21, the 'standard' $t\bar{t}$-production sample was changed; however, this has only a minor impact on the file size (at the 5%-level). The dominant part of the change in HIT file size between MC20 and MC21 is due to the full simulation optimisations described in Sect. 4.2. Changes like the range cut application to all EM processes and the neutron and photon Russian Roulette reduce the number of steps taken in the simulation, and particularly the number of steps taken at large times. This has a significant impact on the calorimeter output, reducing it by almost a factor of two. From MC21 to MC23 the compression algorithm was changed from ZLIB to LZMA, resulting in a 10%–15% reduction in file size after compression. Both these changes affected both the Full and Fast Simulation outputs. The RDO and AOD sizes increased over time due to the larger amount of pile-up. The composition of a background RDO (which stores pile-up information only, to be later overlayed on hard-scatter events) is displayed in Fig. 24, and the compositions of a $t\bar{t}$-production FullSim HITS file and AOD file are shown in Fig. 25. The RDO file is dominated by calorimeter and inner detector information. The largest fraction in the HIT file is that of Silicon HITs (recorded in

the pixel and SCT detectors). The AOD is later reduced to a DAOD (derived AOD, see Sect. 4.5), which is the format used for most physics analyses. The AOD does not contain any reconstructed jet collections, because jets are built only at the derivation step using the low-level inputs stored in the AOD. This approach helps to reduce the file size of the AOD.

5.4 Preparation for new campaigns

Major campaigns for improvements to reconstruction (of either data or MC simulation) or detector simulation require significant resources and are only undertaken after careful preparation and validation. The data and MC simulation must be kept consistent to minimise analysis corrections, and therefore often the reprocessing of one mandates the reprocessing of the other. Typically, two major data (re)processing and MC simulation campaigns accompany each LHC Run: one that is undertaken continuously as the data taking proceeds, and one at the end of the Run, gathering the best-known geometry and conditions and final optimisation of the reconstruction. The time required for these campaigns is dominated by preparation time; the time for the processing itself is typically dominated by fixing rare crashes and similar issues that affect a few events, to ensure that every data event is reprocessed and provided to analysis.

5.4.1 Monte Carlo simulation campaign preparations

The preparation for a new MC simulation campaign involves many inter-dependent steps that must be performed and checked before large-scale production is undertaken. The full sequency of changes described here is required for updates of the detector simulation and its reconstruction; some new MC simulation campaigns involve only changes to the reconstruction of the simulated events, and the first of these steps can be skipped.

First among these is the preparation of the simulation software, including the choice of GEANT4 version. Because changes to the GEANT4 version often result in changes to physics observables (even with the same configuration), the version must be fixed for the duration of an MC simulation campaign. Normally, some large (e.g. 10 million event) sample of $t\bar{t}$ events is simulated to detect any remaining, rare potential bugs, and several other samples are run with modest numbers of events to check for computing performance issues (speed, memory consumption, and so on). Small patches are normally allowed after this stage, built as part of an ATLAS-specific GEANT4 release, but the official GEANT4 patch version is fixed.

In parallel with the fixing of the GEANT4 version, the initial geometry and detector conditions must be prepared. Only coarse (large-scale) detector alignments, the beam spot position and distribution, and the gas mixtures used in detectors

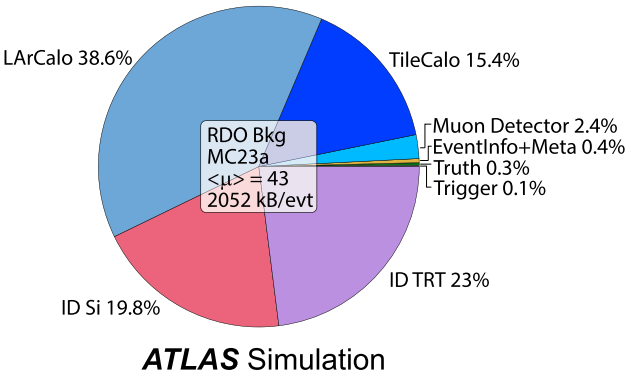


Fig. 24 The composition of a simulated Run 3 background RDO file that stores pile-up information, which is later overlayed on hard-scatter events. The pile-up corresponds to $\langle\mu\rangle = 43$, which approximates the run conditions of data taking in 2022. Here 'ID Si' includes both the silicon pixels and strips

(e.g. in the TRT the gas mixture may change during the run and is therefore recorded as a part of the conditions rather than as a part of the geometry) enter the simulation; other conditions like detailed maps of disabled channels only enter in the reconstruction and therefore can be finalized later on. Small improvements to the geometry might arrive close to the time of production, but most physics validation (see Sect. 5.2) is done with a close-to-final geometry.

With the geometry and GEANT4 versions in place, the calorimeter sampling fractions can be checked, which calibrate the electromagnetic-scale response of the calorimeters. These are normally calculated using bespoke single-particle samples, where the particles are simulated starting from within the uniform sampling region of the calorimeter. The sampling fractions are dependent on the GEANT4 version, owing primarily to the details of the treatment of range cuts in various processes (see Sect. 4.2). Although these are in principle purely geometric quantities, they have also been observed to have dependence on particle incident angle in some detector geometries.

The next step towards production is the preparation of OFCs for the calorimeter (see Sect. 4.3). The calorimeter OFCs are based on noise pedestals that are pile-up dependent, and therefore a $\langle\mu\rangle$ value must be selected for their calculation. This $\langle\mu\rangle$ value need not match the anticipated pile-up conditions as discrepancies can be re-calibrated and high values result in the suppression of some physics signal, but generally a close, round value is used (e.g. $\langle\mu\rangle = 40$). The OFCs (and sampling fractions) are calculated assuming perfect calorimeter detector conditions (e.g. without disabled channels and with nominal high-voltage conditions).

With these steps complete, frozen showers (see Sect. 4.2.1) can be prepared for the MC simulation campaign. Calibrations also begin at this stage with the preparation of some TopoCluster calibrations, calculated based on single particle

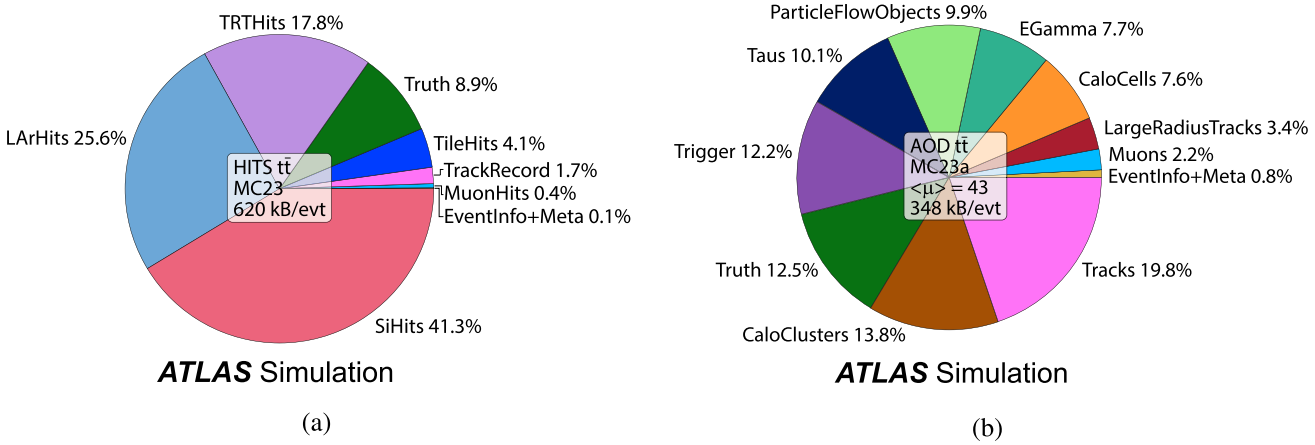


Fig. 25 The composition of a Run 3 **a** HITS file and **b** AOD file. The process simulated in both cases is $t\bar{t}$ with a single lepton from a W-boson decay. The AOD includes hard-scatter information and also simulated pile-up. The pile-up corresponds to $\langle\mu\rangle = 43$, which approximates the run conditions of data taking in 2022

simulation. These each depend only weakly on final detector conditions and non-calorimeter geometry.

At this stage, the geometry and those detector conditions important to the simulation can be finalized, and physics validation of the simulation itself can begin. Often the first validations are done using the reconstruction from the previous MC simulation campaign, since it is well-understood. This procedure allows the isolation of simulation changes, although changes that require new calibrations (e.g. changes in the hadronic energy scale in the calorimeter) must be inspected with care by experts.

With the simulation validated, the digitisation is the next workload to finalize. This includes updates to digitisation algorithms (e.g. for improved noise models or response treatments) and conditions that must be applied during digitisation (e.g. abnormal high-voltage in the calorimeter). These changes are normally validated with and without pile-up, and both the direct digitisation of hard-scatter and pile-up as well as the MC Overlay workflow must be validated. Normally these validations proceed with a preliminary version of the reconstruction to be used in the MC simulation campaign, because often in production MC Overlay and reconstruction are run in a single job on the Grid.

Finally, the geometry, conditions and configuration to be used for reconstruction must be finalised and validated. As a part of the Sample-A production run during physics validation (see Sect. 5.2), computing performance issues can also be checked on the Grid. The final validation and production normally begins once the last updates to the detector and accelerator conditions are included; this can include, for example, a final update to the $\langle\mu\rangle$ distribution used in the sample, which then requires an updated RDO production for MC Overlay before the reconstruction configuration is finalised. Often, only once the full chain is validated and the sample production begins can samples be produced for the training of the various types of fast simulation, which then requires its own update and validation process.

The production itself often begins from samples required for the derivation of calibrations and systematic uncertainties. These must be available before data analysis can begin. Sometimes, *transfer* uncertainties are used, wherein the uncertainty prescription from a previous MC simulation campaign is applied, along with extra systematic uncertainties related to the known or expected changes between the two campaigns. These temporary uncertainties allow faster data analysis uptake in new software releases.

Depending on the number and scale of changes expected, this entire procedure from start to end may take more than a year. This was the case in the lead-up to Run 3, where the thread-safety of the software was being regularly checked on increasingly large-scale samples, new detector elements from the Phase-I upgrades had to be introduced and the corresponding reconstruction software developed and commissioned, and significant uncertainty around the accelerator conditions remained until significant data were recorded. For this reason, often the initial MC simulation campaign in a Run is used as a prototype for preparatory work, validation, early data tests, and very early analysis. A subsequent campaign includes a more realistic picture of the data taking and is used for most data analyses. This was the case, for example, for MC15 and MC16 (the latter has run for seven years), and more recently for MC21 and MC23. For the same reason, a small sub-campaign is often undertaken based on estimates of the coming year's data-taking conditions, and a longer campaign follows the data-taking with conditions that are known to be representative of the year. This was the case, for example, for MC16c (before 2017 data-taking and meant to be representative of the 2017 data) and MC16d (after 2017).

5.4.2 Data reprocessing preparations

To begin a campaign to reprocess data, it is necessary first to identify what improvements will be made and then to validate each of the improvements via the process discussed in Sects. 5.1 and 5.2. The improvements broadly fall into three categories: updates to the detector conditions data, updates to the detector geometry, and updates to reconstruction algorithms.

Conditions data updates [248] incorporate improvements in the knowledge of the time-dependent status of the detector during periods of operation. Typical examples include improved knowledge of detector alignment, detector noise and instantaneous luminosity delivered by the LHC. This information is stored in the ATLAS conditions database [67], an ORACLE® database hosting a COOL technology schema [57] (see Sect. 6.2.2). Each detector system may provide updated conditions in advance of a reprocessing. Only coarse (large-scale) alignment is applied in the detector simulation; small alignment corrections, common in the inner detector and muon spectrometer, can be updated without modifying the simulated geometry. Similarly, updates to disabled or noisy channel maps do not require an update of the MC simulation.

Detector geometry updates incorporate improvements in knowledge of the detector geometry over a long period of stability. Typical examples are updates to maps of *dead material*, the inactive parts of the detector that are not read-out, such as supporting material of subdetectors or the cryostat walls. The updates may also include corrections to the positioning or internal geometry of sub-detectors assumed in event reconstruction. Significant updates to the geometry may require a new simulation campaign to maintain good agreement with the data.

Algorithm updates incorporate improvements to the algorithms used to reconstruct the events. New tracking and vertexing algorithms, as well as updates to lepton identification and algorithms are typical examples. The updates may also include fixes to deal with crashes or floating point exceptions observed in events that were previously problematic to reconstruct. Many of these updates require corresponding changes to the reconstruction of the MC simulation; some, however, might involve improvements to the rejection of backgrounds that only occur in real detector data, in which case only the detector data must be updated.

The validation of these changes uses the Data Quality tools described in Sect. 5.1. Most often, a small sub-set of data are first processed to test for technical issues. Later, a larger sub-set of the data are processed and passed through a Data Quality validation process to check for issues with various updates and ensure that the changes are as expected. Particularly because conditions are period-specific, this larger subset includes runs from several different conditions periods. Only after all of these checks are completed is the reprocessing of the remainder of the data undertaken. Afterwards there may be updates to the Good Runs List to include additional data that could be declared 'good' thanks to improvements in the conditions or reconstruction.

6 Infrastructure and databases

This section describes the significant central infrastructure required to support the services, systems, and developers of the collaboration. Section 6.1 introduces the infrastructure maintained to support the software development cycle. The database systems used throughout the experiment are described in Sect. 6.2. Finally, the development and use of metadata, much of which relies on these systems, is described in Sect. 6.3.

6.1 Software engineering process and infrastructure

The software developed for event simulation, reconstruction and analysis, as well as for detector calibration and alignment, has to run on the computing facilities that are available to the collaboration members; these facilities are not necessarily uniform in their hardware architecture, nor their operating systems and environments. To ensure that ATLAS software is robust and portable, it is necessary to provide a common process and infrastructure to integrate all new code with the existing code base, and then build and test changes. A central ticket and workflow system is provided using JIRA [246], which provides functionality for bug reports, improvements, and tasks; for the preparation of MC simulation requests before they enter the production system; for tracking physics validation (see Sect. 5.2); and for the tracking of long-term development. GitLab integration is used to ensure code updates and tickets are cross-linked.

6.1.1 Software structure and compilation system

The ATLAS software, shown in Fig. 26, is organized into a project for general use (Athena) and several projects for specific purposes like event generation (AthGeneration), detector simulation (AthSimulation), and data analysis (the AthAnalysis and AnalysisBase projects). These rely on several projects like 'TDAQ Common' (common trigger and data acquisition software) and several external packages (e.g. event generators, ROOT, or GEANT4). Each project is further sub-divided into packages with much more specific purposes (e.g. pixel detector digitisation, or muon momentum calibration); there are about 2000 packages in the ATLAS software.

ATLAS uses the CMAKE build system [44] for building all of its own code, in conjunction with most of the HEP software projects that the experiment uses. CMAKE is a very flexible

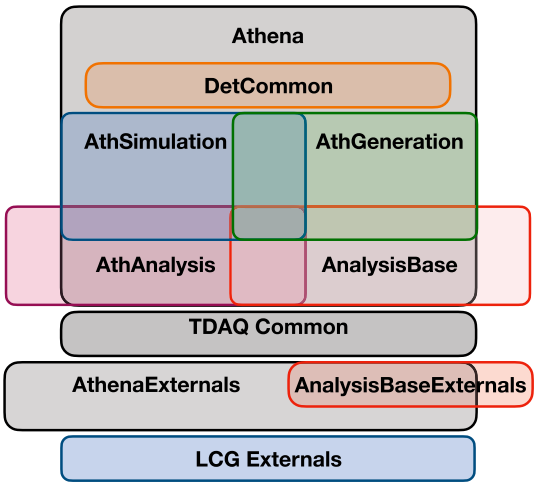


Fig. 26 Schematic overview of the ATLAS software projects and external dependencies. The overlaps between project areas indicate overlaps in code included in the projects. The widths of AnalysisBase-Externals and AthenaExternals are wider to indicate their support for the entirety of the AnalysisBase and AthAnalysis projects, respectively

system supporting Linux, MacOS and Windows hosts, x86 and ARM CPU (aarch64) architectures, GNU MAKE, and NINJA and additional low-level build systems, with a rich high-level language allowing support for the complex software building steps required for the ATLAS reconstruction, simulation, trigger, and analysis code.

Thanks to the flexibility of CMAKE and to harmonise how different parts of the code are built, ATLAS-specific functions and macros are used throughout the build configuration of the offline and trigger software for setting up the building of libraries and executables, and for dealing with the installation of scripts and data files. For the offline software, that basic CMAKE infrastructure is maintained in the `AtlasCMake` package [249].

One fundamental requirement for the build system is to allow the users to build small parts of the software against a pre-built full software release (a *base project*), as described in Sect. 6.1.5. This is achieved with imported targets in CMAKE. Using `AtlasCMake`'s facilities the user is able to build packages and libraries as part of a project that are already part of the base project. This is achieved by only importing the targets of the base project into the build of the current project that are not present in the current project already.

The HEP_OSLIBS meta package [250] provides the software package dependencies for the Athena build and runtime environment for the two currently-used operating systems (CentOS 7 and Alma Linux 9; see Sect. 6.1.2). The LCG software stack [251], maintained by the CERN EP-SFT group in collaboration with several large LHC experiments like ATLAS, provides several hundred external packages that include recent versions of commonly used packages as well as HEP-specific tools and MC event generators (see Sect. 4.1). There are usually two or three major LCG software releases per year and development builds are available every night. The major LCG versions are normally released with a new major stable release of ROOT [46]. Minor LCG version updates are done every few weeks and usually contain updates of MC event generators or bug-fixes of other external packages. The LCG software stack is built for x86_64 and aarch64 architectures and several recent versions of the GCC and Clang compilers and C++ standards. As of autumn 2023, LCG version 104b with ROOT 6.28/08 is used in Athena, compiled using GCC 13 with the C++20 language standard.

External software, software used by the ATLAS software that is maintained separately from the experimental software and is not provided by an LCG package, is built using CMAKE as part of an ATLAS external project. As shown in Fig. 26, every ATLAS offline software project (Athena, AthSimulation, etc.) has a corresponding external project (AthenaExternals, AthSimulationExternals, etc.), upon which it is built. This includes packages like GAUDI [48], and allows a more rapid update of external packages that receive rather frequent version updates or are not sufficiently common to be included in the LCG stack directly. This separation is not present in the TDAQ projects: TDAQ Common builds a combination of external and ATLAS packages as part of the same project. Some parts of this TDAQ software are also required in the ATLAS offline data processing, for example to read the ATLAS RAW data.

Any software build that uses CMAKE is separated into configuration, build, installation and packaging steps. Configuring the build of small projects like analysis projects, or the WorkDir project used for developing code in the Athena repository, takes just a few seconds, while configuring the build of the largest project, Athena, takes a few minutes.

Each package is typically built into one or more small dynamic libraries that are loaded on-demand at runtime. This helps ensure a minimal memory footprint from library loading despite the wide variety of jobs that can be run within Athena. It also allows the user to override specific libraries provided in the release or by the system, for example by preloading a different memory allocation library or math library. There is a small CPU overhead associated with dynamic library loading, discussed further in Sect. 4.2.1.

The packaging and distribution of the ATLAS software is achieved using RPM packages built using the CPACK tool within CMAKE [252]. The RPMs are configured to depend on each other such that the installation of a full Athena release can be done by requesting the installation of just the top-most Athena RPM package. This top-most package in turn ensures that all dependent RPMs, including those provided by LCG and packages providing various data files (e.g. for GEANT4) are installed in the correct place automatically.

6.1.2 Build system, CI, nightly and stable releases

The ATLAS Nightly and Continuous Integration (CI) Systems provide a modern software development workflow for developers, feature fast development cycles and assure confidence in new software deployments. Both the systems are Jenkins-based [253] and have a long evolution history [254]. Interconnected with the ATLAS GitLab code repository [26], the CI system performs up to 100 multi-project software builds daily, probing code changes proposed in GitLab merge requests. A comprehensive test suite of unit and short integration tests runs for each CI job. The Nightly System (separate from CI) performs daily builds of all ATLAS software projects from the code repository (called 'nightlies' because the largest projects are built overnight), often on several platforms. It maintains a multi-stream, parallel development environment with many simultaneous branches. The Nightly System probes how the changes from accepted merge requests work together. In addition, it helps to support migrations to new platforms and compilers and verify patches to external tools.

The CI and Nightly jobs run on the 1400-core build farm, including both real hardware and virtual nodes. Various *operational intelligence* techniques such as incremental compilations, selective testing, operations parallelization, and dimensionality reduction result in efficient resource use and faster delivery of results. The CI and Nightly monitoring system provides dynamic information about build and test results and installation status. It is based on the Nightlies Database residing in the ATLAS database production cluster dedicated to offline analysis (ATLR, see Sect. 6.2.1) [255]. The ATLAS Nightlies Database is the source of dynamic content for the nightlies dashboards hosted on the BigPanDA web application [256] (see Sect. 7.4.2).

The systems build software releases from the GitLab main branch and from dedicated branches for the online high-level trigger, reconstruction and simulation of Run 3 data, and legacy branches for Run 2 reconstruction, simulation and analysis. Regular 'sweeps' are used to copy changes made in various branches to the main branch. Special development nightlies are available that use development versions of ROOT, LCG builds, or other externals. These ensure that when preparing for significant changes (e.g. a new operating system, GCC version, or C++ language standard), the changes can be thoroughly vetted without putting at risk the main development stream.

When the compilation and test steps of the nightly build are successfully completed, CPack [252] is used to produce RPMs, which are then uploaded to EOS [257]. This acts as a staging area for the subsequent installation step that fetches them and, using AYUM [258], an ATLAS-specific wrapper around YUM, installs the nightly releases in the atlas-nightlies repository on the CernVM FileSystem (CVMFS) [259], and thereby makes them globally accessible to developers. CVMFS is the service by which ATLAS distributes all of its development and production software, and auxiliary datasets.

The installation software [260] is executed locally on a publishing node (officially, a Release Manager) that communicates with a backend Gateway node. Having multiple publisher nodes allows concurrent transactions, increasing the overall publication rate of the repository and improving the speed at which nightly builds become available.

Currently the total size of the atlas-nightlies repository is 10 TB, with approximately 4 TB in use. In 2020, the repository data was migrated to CEPH-based S3 object storage, resulting in a performance enhancement over the previous AUFS system (advanced multi-layered unification system). All nightly releases (currently about 35) built in ATLAS and using different architectures (binaries, operating systems and compilers) are deployed on CVMFS in the form of RPMs. The installation time ranges from 3 to 60 minutes, depending on the number of external packages needed and the format of the build (e.g. whether debugging symbols are included) and type of release (e.g. DetCommon/AnalysisBase compared with full Athena). The installations are kept for 30 days by default, with the possibility of extensions for developers to test newer code against older nightlies or if a physics validation is performed with a nightly release (see Sect. 5.2).

The nightly releases are rigorously tested in the ATLAS Release Tester (ART) Grid-based framework [261] described in Sect. 6.1.3. When the set development goals are achieved, a successful nightly release is transformed into a stable release by the team of ATLAS offline release shifters. Stable releases have unique numeric identifiers and indefinite lifetime. The numbering of stable releases follows the pattern A.B.C(.D), where A is the major release version that changes at most once a year, B generally indicates the purpose of the release (e.g. 2 for analysis, or 6 for generation; these numbers are often historical), C indicates the minor release version and is typically increased about once a week for a new stable release, and D is an optional patch version in case a stable release must be patched. Patching is most often used so that a stable release, for example one to be used for trigger simulation, which must therefore match the release used during data taking, must be patched to fix a rare bug, add a feature for MC simulation, or read a new file version.

All production releases until fall 2023 were built on the CentOS 7 Linux operating system [262] on the x86_64 and aarch64 architectures using the CMAKE build system described above. A migration to Alma Linux 9 [263] of all infrastructure was completed by the end of 2023, though several legacy releases continue to be based on CentOS 7.

The default compiler for all Run 3 releases is at present GCC version 11.2.0 [264] with the BINUTILS 2.37 [265] and the C++17 standard. In parallel there is a nightly release build from the GitLab main branch using the Clang 16.0.3 com-

piler [266] on the x86_64 architecture. Maintaining builds with at least two compilers has helped identify a variety of minor issues in the software and led to increased robustness. For the releases built on Alma Linux 9, the recent stable version of GCC, version 13.1, with the C++20 language standard and BINUTILS 2.40 are used. Nightly builds on the aarch64 architecture exist both on the CentOS 7 and Alma Linux 9 operating systems with the GCC 11.2 and 13.1 compiler versions respectively. Since summer 2023, the main branch releases are built for the microarchitecture x86-64-v2 and newer [267], which brings support among other things for vector instructions up to Streaming SIMD Extensions 4.2 (SSE4.2). Newer microarchitecture levels are not yet supported by a sufficiently large fraction of machines available on the WLCG Grid to make adoption beneficial.

6.1.3 Release testing

The ART system [261] is designed to run test jobs on the Grid after an ATLAS nightly release is built. The choice was made to exploit the Grid as a back-end as it offers a huge resource pool, suitable for a deep set of integration tests, and running the tests could be delegated to the highly scalable ATLAS production system (PANDA, see Sect. 7.3). The challenge of enabling the Grid as a test environment is met with the CVMFS file system for the software and input data files. Test jobs are submitted to the Grid by the GitLab-CI system, which itself is triggered at the end of a release build. Jobs can be adorned with special headers that direct how to run the specific test, allowing many options to be customised. These options might include which releases are to be tested, input- and output-file specifications, and the number of cores to be used for multi-process or multithreaded running, for example. The GitLab-CI waits for the exit status, and output files are copied back from the Grid to an EOS area accessible by the users. All GitLab-CI jobs run in ART virtual machines using DOCKER images [268] for their ATLAS setup. ART jobs can be tracked by using the PANDA monitoring system.

ART can also be used to run short test jobs locally. They use the same ART command-line interface, where the back-end is replaced to access a local machine for job submission rather than the Grid. This feature allows developers to ensure their tests work correctly before adding them to the system. In both the Grid and local machine options, running and result copying are completely parallelised.

6.1.4 Software quality

To assist in the development of high-quality code, ATLAS has maintained a Coding Standard [269]. The goal of the standard is to ensure some consistency in code style (e.g. naming conventions for variables and files) so that developers can quickly understand a piece of ATLAS code with which they are not familiar, while acknowledging that with thousands of developers working over tens of years, not all style issues must be rigorously enforced (e.g. spacing and bracket placement is not enforced). The standard is divided into requirements, which are checked using a static code checker implemented as a GCC plugin [61], and recommendations that are taught but not enforced.

A wide variety of static code-checking tools are used, including COVERITY [270], CPPCHECK [271], and LIZARD [272] for C++, and FLAKE8 [273] for PYTHON. Both FLAKE8 and CPPCHECK (when available) are integrated into the build system so that compilation warnings are raised if code problems are identified.

The primary mechanism used to enforce code quality and consistency is via multi-level reviews of merge requests. Because of the size of the code base, 'tidying' the entirety of ATLAS code would be impractical. Instead, when touching a particular piece of code, developers are encouraged to gently improve the code around their changes, help identify and improve missing documentation or code weakness, and only introduce new code that is of high quality. For most merge requests, a first-level reviewer looks over the changes to ensure that they are sensible and well explained in the merge request, and that all of the automated tests have succeeded. If the merge request is particularly complex, a second-level (expert) reviewer might be asked to examine the code further. Some domain-specific review is often performed, as the CI system automatically alerts some experts to changes to code within their domain. Finally, a release coordinator is responsible for a last check before the changes are merged into the central repository.

6.1.5 Development and run environment

As described in Sect. 6.1.2, the current default runtime environment is the CentOS 7 Linux operating system. The AFS [274] and EOS [257] file systems are used during the software nightly builds to access larger calibration files that are then integrated into the respective software releases.

Although primarily designed for clusters of Linux processors with open external connectivity that allows the software to be imported through CVMFS and the conditions data accessed through FRONTIER (see Sect. 6.2.2), ATLAS software must be able to run on different hardware architectures and different operating system versions. A notable example is High-Performance Computers (HPCs), which may have large numbers of processors but may lack external connectivity. In that case the software and conditions data need to be packaged into a container that is then uploaded through a gateway to the HPC; input and output data files also go through the same gateway.

The ATLASLOCALROOTBASE (ALRB) [275] suite of shell scripts, whether accessed from CVMFS or locally

installed on a laptop or HPC site, provides a uniform look and feel to configure many tools with their dependencies in the correct order by encapsulating the details. It is also a wrapper to start containers with the necessary environments and mount points for runtime DOCKER [268], SINGULARITY [276], APPTAINER [277] and SHIFTER [278]. It provides mechanisms to test tools and validate containers before deployment. Diagnostics are also available in ALRB for user support and for tutorials. It works for BASH/ZSH shells, x86_64 and aarch64 architectures, and RHEL-derived operating systems with versions 5–9; it is PYTHON 3 ready and supports containers on other Linux flavours and MacOS.

The ATLASSETUP (or simply ASETUP) script [275] provides users a convenient and fast way (a few seconds) to locate the required release and configure its environment for stable and nightly releases. For the latest nightly releases, ASETUP checks the release completeness because a nightly release is built in a few steps with a time gap of about an hour. It only requires users to specify a few short tags, like the project (e.g. Athena or AthGeneration) and stable version number or day of the month for a nightly release. There are a few layers of configuration for ASETUP, and ASETUP can print out the origin of a given configuration parameter. The original shell environment is saved before the first ASETUP run, so that ASETUP can configure another release environment in the same session, even in subshells. The details of the required release are saved into the working directory, such that the same release under that working directory can be restored easily and quickly the next time.

The ASETUP script works for BASH/ZSH shells, with PYTHON 2 (including old PYTHON 2 in SLC5) and PYTHON 3, and in both SINGULARITY/APPTAINER and DOCKER containers. It supports releases in architectures of i686, x86_64 and aarch64 for RHEL-derived operating systems from SLC 5 to Alma Linux 9. In the case of a conflict between the setup release environment and system commands, ASETUP helps provide a solution via wrappers and aliases. A suite of regression tests comes with ASETUP to validate new versions, as ASETUP can run in different RHEL-derived Linux operating systems through SINGULARITY/APPTAINER and DOCKER containers from CVMFS. The regression tests include more than 100 tests for the CentOS 7 platform.

Most contributors to the ATLAS software only need to work on a subset of packages (usually one or two packages) and do not require checking out the whole Athena GitLab repository. A wrapper command, GIT-ATLAS [279], handles the consistent checking out of the packages the developer needs using the sparse checkout functionality, as well as an interface to the developer's fork, which is necessary for submitting merge requests. A user needs only to specify the package name (e.g. `PixelDigitization`), and the code from the relevant package area (in this case, `InnerDetector/InDetDigitization/PixelDigitization`) is added to the existing sparse checkout. This configuration enables local partial builds against the nightlies, significantly speeding up development cycles. Since it aimed to minimise the disruption to (non-core) feature developers, the GIT-ATLAS command was very useful to many developers during the transition from SVN to GIT, as it did not expose the complex commands required for sparse checkouts to the end users.

6.1.6 Central services

The Central Services Operations (CSops) team is central to all ATLAS activities. Its role is to support a wide-ranging set of projects given the great variety of services needed by ATLAS computing. The team is responsible for the configuration, deployment, and operation of different services on the CERN IT infrastructure. They also act as the interface between CERN IT and the service managers of different projects and ensure that security and good computing practices are maintained.

CSops is responsible for the management of over 630 machines with more than 4700 CPU cores, spread across 60 projects including PANDA [115] (see Sect. 7.3.1), RUCIO [280] (see Sect. 7.2.1), the services described Sect. 6.1.2, and others. These machines are hosted in the CERN-IT OPENSTACK [281] infrastructure and the configurations are managed by PUPPET [282], which allows a significant reduction of complicated central operation tasks, while ensuring the high availability and scalability of all ATLAS central services.

These machines can be split up into three main groups: those for ADC, those for ATLAS nightly software building, and the miscellaneous nodes. The last group contains machines from many different detector and operations groups in ATLAS and include critical web services, logbooks, monitoring and other services that are used by both the offline and online teams in ATLAS.

More than half of the machines deployed in OPENSTACK are maintained with PUPPET manifests. This helps to ensure that security and configurations are reproducible on a large scale. All machines that are controlled with PUPPET can generally be destroyed and recreated within 45 minutes. There are 139 different configuration manifests in place that are used to define the major configurations. Specific changes to each configuration can be made with YAML [283] files that are used by the PUPPET Database to fine tune individual hosts. All of these manifests and configurations are stored in CERN GitLab repositories to aid with rapid deployment and keep track of changes to configurations over time.

Together with ADC and the ATLAS TDAQ system administrators, CSops also helps to maintain the Simulation at Point 1 (Sim@P1, see Sect. 7.3.2) project, which is an opportunistic cloud that makes use of the ATLAS TDAQ HLT computing farm for offline grid workflows. The Sim@P1 project

is able to spawn over 2000 virtual machines using in excess of 111,000 CPU cores that can be deployed in under 15 minutes; however, this is usually done over an hour for maintainability and to avoid unnecessary pressure on critical systems.

6.2 Databases

6.2.1 Database infrastructure

ATLAS relies on several services for the management of non-event data; for example data derived from the run configuration, detector calibration and alignment, slow control system of the experiment, metadata associated with events, raw data files or data and workload management systems. Most of these services profit from a database infrastructure that is maintained by the CERN IT department and is based on ORACLE® technology. Other applications include analytics clusters utilising the HADOOP [284] file system and ELASTICSEARCH [285]. The same infrastructure supports other database applications like those for ATLAS publications and membership information [286].

Some conditions data, including information about the detector state (e.g. temperature, enabled and disabled modules, and the trigger configuration) are recorded in real time and automatically inserted into the conditions database. In some cases these data may undergo pre-processing to reduce their volume and simplify subsequent use. Significant additional conditions data are inserted into the database by experts and calibration tools that process the detector data later on.

The original (Run 2) relational database infrastructure, as shown in Fig. 27, consisted of three production ORACLE® clusters deployed and maintained by CERN IT: ATONR for Point 1 (P1) usage, ATLR as a general purpose cluster and ADCR for data and workload management systems like RUCIO and PANDA (see Sect. 7). In this configuration, for data-processing purposes a subset of the ATLR data (which is limited to conditions, geometry and some trigger information) is regularly replicated to two Tier-1 sites, TRIUMF and CC-IN2P3. ATONR was and still is within the firewall at P1, protecting it from accidental external use.

During the long shutdown between Run 2 and Run 3, the relational database infrastructure was consolidated in view of changes to the ORACLE® licensing model at CERN. ATLAS therefore progressively migrated to a new infrastructure, now in place and illustrated in Fig. 28, completing the migration in early 2024. The cost and the complexity of the system was reduced by eliminating ORACLE® instances at the Tier-1 sites, and to consolidate the CERN resources of the P1 cluster ATONR and its standby ACTIVE DATA GUARD (ADG) instance that are accessible from outside of the P1 network in a read-only mode. An important step towards consolidation was the migration of the ATLAS Metadata Interface (AMI) database [287] (see Sect. 6.3), which was historically based at CC-IN2P3, to CERN by the end of 2021. Although the independent infrastructure had offered some advantages in terms of resilience, the advantages of the tighter integration, central support, and centralised authentication mechanisms available at CERN made the migration beneficial. The migration of the conditions data in the offline schemas into ATONR is now complete as well; to preserve isolation of Point 1 during data taking a special proxy server was developed to permit updates of conditions in the offline COOL schemas. Additional nodes were added into the ATONR cluster to accommodate additional workflows originating from the General Purpose Network, which were previously utilizing ATLR. This expansion ensures the preservation of the existing resources for data-taking workflows at Point 1. On the ADCR cluster, new more powerful hardware was deployed to support the heavy workflows coming from the data management system. The most delicate parts of this migration had to be performed during pauses in data taking (either short technical stops or end of year shutdowns), but the migration could be done during Run 3.

6.2.2 Conditions data distribution

The conditions data are distributed using several technologies. The main service in place for access to conditions is FRONTIER [288], a web application that redirects SQL requests to ORACLE® databases and provides a caching layer implemented using Squid proxies. The system is in use since Run 1, and the main difference in Run 3 is its deployment architecture as mentioned in the previous section; the FRONTIER launchpads are deployed only at CERN and provide access via the ATONR_ADG cluster. The network access to conditions via FroNTier may not be possible, especially for certain HPCs lacking outbound network connectivity (see Sect. 7.3.2). In such cases, an alternative approach is available; data can be extracted from ORACLE® and distributed through simple SQLite files. In this case there is the possibility to extract data from ORACLE® and distribute them via simple SQLite files. The tools for the easy generation of an SQLite file with a selection of a subset of conditions data are gathered in a GitLab repository [289]. The SQLite files can then be distributed in a file system or via a DOCKER container provided their volume are reasonable.

6.3 Metadata handling

The ATLAS experiment has developed three systems that build upon information from other ATLAS systems to provide unique metadata services to the experiment at the three fundamental levels of granularity of ATLAS data: services at the event, run, and dataset levels are provided by the EventIndex, COMA, and AMI, respectively. Each of these systems has evolved considerably since Run 1 in mutual cooperation

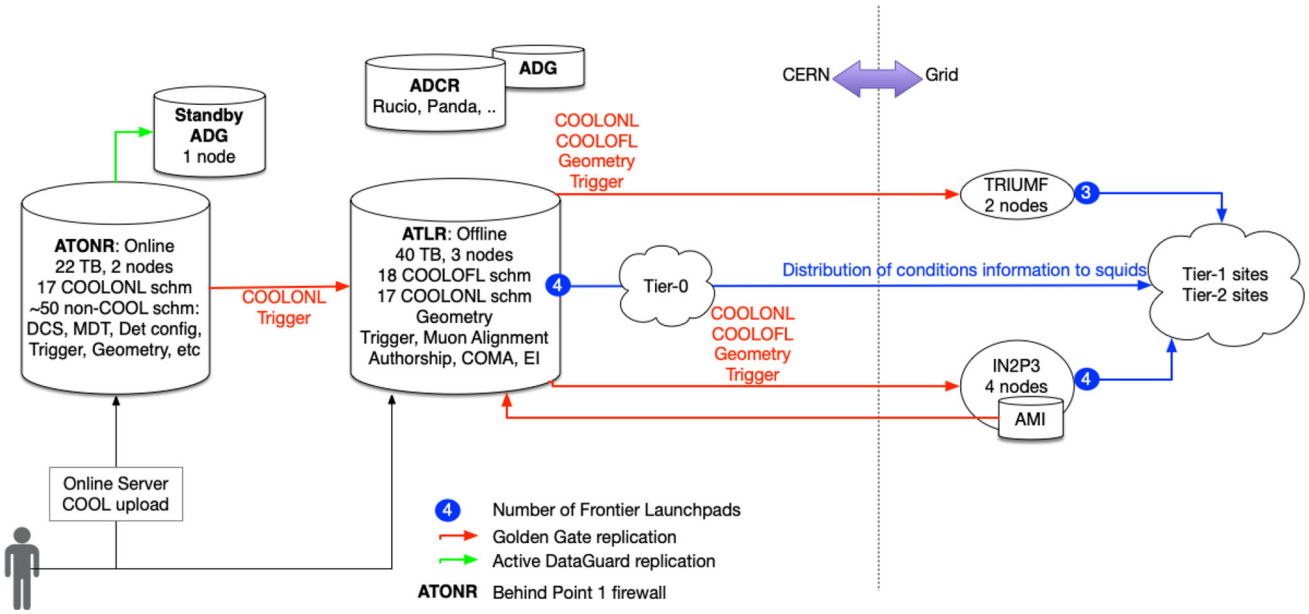


Fig. 27 The database infrastructure employed by ATLAS before Run 3. The person icon represents the injection of conditions information into the database by either an expert or an application

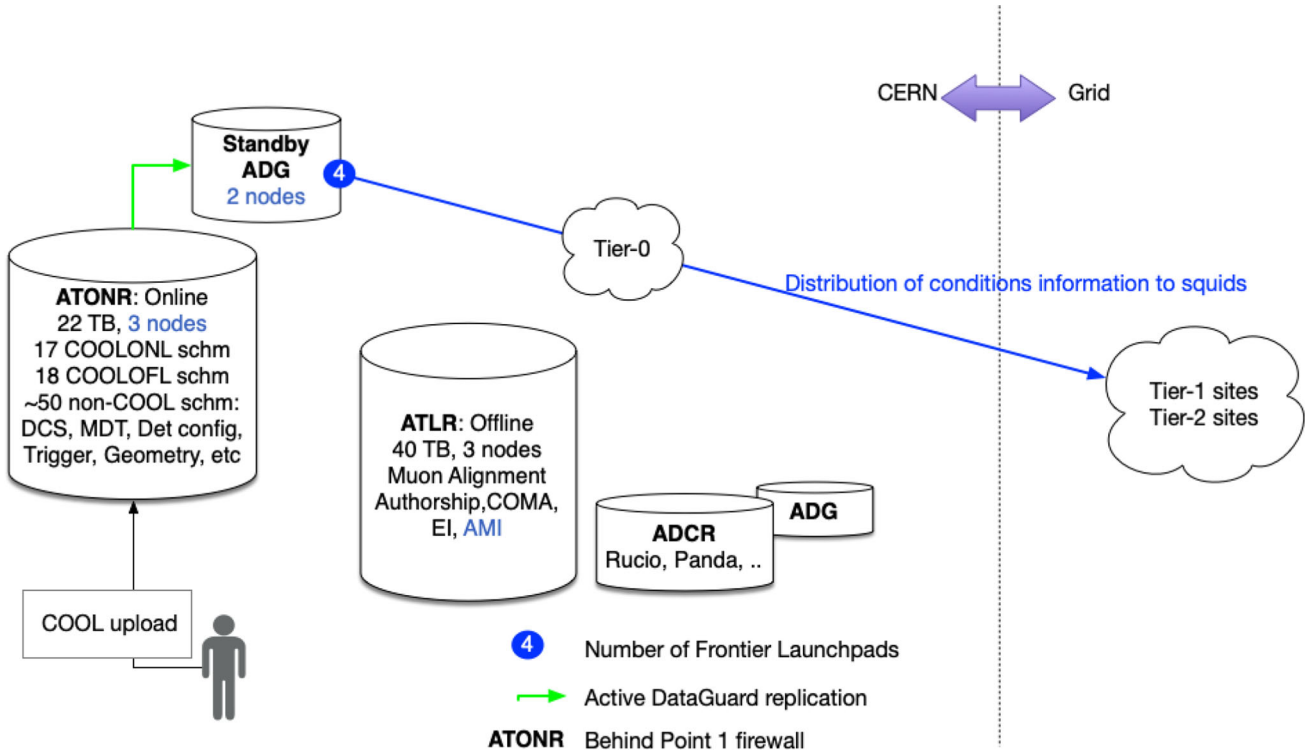


Fig. 28 The database infrastructure employed by ATLAS for Run 3. The person icon represents the injection of conditions information into the database by either an expert or an application

with many systems and services throughout ATLAS (having connections to aid data collection and service optimisation). Furthermore, they each have evolved into related areas to provide additional services as described in this section.

The EventIndex [290] is the metadata catalogue of all ATLAS events. For each real or simulated event in the primary processing formats (e.g. RAW and AOD), it stores the main event identification parameters (run and event number, trigger stream, luminosity block), the trigger record and the provenance information, i.e. the Globally Unique Identifiers (GUIDs) of the files that contain it. This information is used to check the completeness and correctness of the processing campaigns, as the number of output events must match the number of input events and each event must be found once and only once in the output streams, and to monitor the overlaps between trigger chains or between offline selection streams. The provenance information is used to extract one or several events that are selected during any analysis and produce event displays (see Sect. 8.2), to study details of the calibration, alignment and reconstruction algorithms, or for special analysis workflows. In this last instance, physics analyses that need specific reconstruction algorithms to be applied on specific event samples (up to one million events) can use EventIndex provenance information and the Event Picking Service [291] to extract those events from the bulk of the RAW data and process them separately. The EventIndex metadata are collected by Grid jobs that scan all newly produced data files. These metadata are then sent to a central storage system at CERN, implemented using HADOOP [284], HBASE [292], and PHOENIX [293], an interface layer on top of HBASE that allows SQL access. For Run 3, the system was extensively revised and re-implemented [294]. The EventIndex stores several hundred billion real and simulated event records and grows linearly with data production rates.

The COMA system (Conditions/Configuration Metadata for ATLAS) [295] collects run-level conditions and configuration metadata mainly from the Conditions database, as well as essential run-related information from other systems: the HLT farm, Tier-0 site, Trigger, AMI, and EventIndex data repositories and file systems such as good runs list (GRL) XML files (see Sect. 5.1). Early in Run 1, COMA became the main repository to store the ATLAS Data Periods, which are official sets of run numbers during physics data-taking with common detector or machine conditions. These ATLAS Periods form the basis to group sets of runs for many purposes for ATLAS data processing and reprocessing as well as data analysis. COMA provides both web-based and command line interfaces featuring collected data and unique related derived quantities, as well as aggregated information across many runs (e.g. runs in specific projects, periods and GRLs) such as event counts (by stream and trigger), data volumes, luminosity and LHC beam-related quantities. To ease the management of the ATLAS Conditions database, the COMA repository was expanded [296] to collect metadata about conditions data structure, payloads and related metrics, providing reports to help experts understand conditions database organisation, content, and usage. COMA systems have undergone continuous refinements in data collection and client utilities into Run 3.

The ATLAS Metadata Interface (AMI) [287] is a software ecosystem dedicated to scientific metadata that provides two critical services; management of metadata for all datasets produced by the experiment and bookkeeping of all the parameters defining the data processing workflows. The AMI task server aggregates metadata information from several sources (e.g. RUCIO and PANDA) via specific tasks and stores them in the AMI database. Metadata sets are recorded for each processing step a dataset undergoes and the 'filiation links' (provenance information) between processing steps of the same datasets are saved.

In parallel with ongoing operations, the AMI environment is constantly evolving: an internal rewrite of both the core server and Web interfaces were done for Run 3 to provide improved stability, scalability and flexibility when addressing the metadata needs of the users. A new administration interface for the AMI task server was written to easily isolate, monitor and manage tasks. In addition, the interface to make super containers (a set of datasets for a given data taking period or year, or any other logically-grouped set of datasets) was rewritten. A prototype was developed and tested for the revision of the AMI-tags that define all the parameters of a processing workflow; it will be finalised in coordination with the Tier-0 and database teams.

Users most often access AMI via Web applications to manage and display metadata. These are also available via command-line interfaces for easy use in scripts or data analyses. AMI also recently developed a whiteboard system that allows users to set arbitrary tags onto a dataset to be able to retrieve them in a more efficient manner. These tags are used to identify samples that are standard samples to be used in data analysis; they can also be used to identify all datasets in use by a single analysis, for example.

In addition to these external metadata systems, ATLAS employs *in-file* metadata to transfer information from one processing step to the next. These data include information about the configuration of the job that produced the data (e.g. the parton distribution function used during event generation, or the geometry used during simulation). They can be used for automatic configuration of subsequent workflow steps. For example, the reconstruction automatically initializes the correct geometry based on information in the input file. This automatic configuration can be overridden, or the information can be explicitly provided if it is not present. The in-file metadata can also be used for consistency checks, for example by checking that files to be combined in MC Overlay (see Sect. 4.3.2) were created with a consistent geometry. For real

data, the in-file metadata includes information about which luminosity blocks were processed, to help ensure that the entire set of events from each luminosity block is correctly processed in an analysis. During derivation production (see Sect. 4.5.1), some of the in-file metadata are converted into a simple ROOT-readable format. These can then be used in analyses, for example for the configuration of calibration and uncertainty tools. Including this in-file metadata helps reduce the necessity of database connections in analysis jobs and enhances the ability to run analysis with only simple ROOT or similar software.

7 Distributed computing

ATLAS Distributed Computing (ADC) comprises the hardware, software and operations needed to support distributed processing, simulation and analysis of ATLAS data and to support the evolving needs of the experiment. This can be broken down into two closely related areas, which operate side by side: Distributed Data Management (DDM), covering all aspects of storage, transfer, and access to the (collision and simulated) data, and the Workflow Management System (WFMS), which handles request, task and job definition, and manages the various workloads performed on the ATLAS data, for both large scale production tasks and user analysis. The first site through which all data move, the Tier-0 site, uses special configurations and software due to the stringent operational requirements of the site. This site is described in Sect. 7.1. The remaining Grid sites use common software for both the data and workflow management. The DDM model employed by ATLAS is presented in Sect. 7.2, followed by a description of the many aspects of the WFMS in Sect. 7.3. The data monitoring and analytics systems described in Sect. 7.4 naturally cover both of these areas.

7.1 The Tier 0 site

The ATLAS Tier-0 site comprises the hardware, software and operations needed to support the *prompt* processing of the data produced by the ATLAS detector. This prompt processing mainly consists of the first-pass reconstruction workflow and many calibration workflows (see Sect. 2.4). The Tier-0 site also provides support for ad-hoc reprocessing, and is frequently used for various commissioning tasks. Reprocessing and commissioning tasks may include the recall of data from tape storage if necessary.

The Tier-0 site uses the same conceptual models of *datasets* (i.e. collections of files) being processed by *tasks* (i.e. collections of jobs executed on some external batch service) as DDM (see Sect. 7.2) and WFMS (see Sect. 7.3). However, it uses different workflow management software from most Grid sites for the prompt processing of data.

7.1.1 Tier-0 resources

The Tier-0 site consists of about 40,000 cores in the CERN HTCondor-managed batch system. The cluster is shared with Grid processing, which takes over the resources if they are not needed for Tier-0 operations (see also Sect. 7.3.2). The Tier-0 site has about 2 PB of EOS disk space available to temporarily store intermediate datasets, and about 100 GB of shared AFS disk space to store all other (non-dataset) files needed for operation.

In addition, a single-core virtual machine is used to run the software that drives Tier-0 site operations. Another single-core virtual machine is used to host the web service allowing both the monitoring and operation of the system.

7.1.2 Tier-0 software

The Tier-0 software is designed to depend on as few external services as possible. This not only minimises the chance of service interruptions caused by them, but also the effort required to follow their evolution over time. The services the Tier-0 software does depend on tend to be stable and highly reliable. For the software, there is a similar strategy to depend on as little third-party software as possible, for the same reasons.

The Tier-0 software system is a modular, self-contained system comprising a workload management system (comparable to ProdSys2-PanDA, as described in Sect. 7.3), a data management system (comparable to Rucio, as described in Sect. 7.2), a monitoring system (probes, data collectors, alarms, etc.), and a comprehensive web interface for monitoring and operations.

The Tier-0 workload management system consists of two entities: one operating on the dataset/task level (the Tier-0 Manager, or TOM), and one operating on the file/job level (the Supervisor). Both follow a component-oriented software paradigm; each entity is composed of approximately ten components that collaborate to provide the desired functionality. This approach not only allows different implementations of the same component (e.g. interfacing to different batch system flavours, or to different database backend flavours), but also allows the sharing of components between the two entities (e.g. the logging component). In addition, some components provide a plug-in mechanism that allows loading of alternative implementations for particular key algorithms.

This combined component-oriented and plug-in approach has not only facilitated a smooth evolution of the system over the last 15 years, but also allowed another CERN experiment (NA62) to use the ATLAS Tier-0 software system with minimal effort [297].

Tier-0 manager Each Tier-0 Manager (TOM) instance/process runs a set of configured plug-in tasks, called *tomprocesses*, at regular intervals in an infinite loop. Each TOM is single-threaded, ensuring strict serialisation. The main task of the TOMs is to implement the Tier-0 reconstruction and calibration workflows. The paradigm followed here is that of a dataset blackboard: files created on EOS and published in a handshake catalogue by the online system are organised by a tomprocess into RAW datasets and put on a virtual blackboard. Other tomprocesses regularly scan the blackboard for new datasets fulfilling certain criteria and, if triggered, automatically define tasks that process these datasets into one or more new output datasets. For Run 3, about 100 such dataset-processing tomprocesses are defined, each configured with about 20 parameters.

Taking advantage of the generic architecture of the TOM system, most other Tier-0 activities are also run as tomprocesses. Examples include the registration of permanent output datasets and their files with `Rucio`, the management of the temporary EOS disk space containing the transient datasets, and the management of the temporary AFS disk space.

Supervisor As the name suggests, the task of a Supervisor process is to supervise the execution of the jobs defined by the T0 Manager(s). Each supervisor process is single-threaded, and multiple processes can be configured and deployed simultaneously. The standard configuration for Run 3, however, only needs to deploy a single instance, which manages the about 10,000 jobs running simultaneously on the HTCondor batch system.

Web control interface The Tier-0 web interface, called *conTZole*, is a classical JavaScript/Django/Apache single-page web application. It allows comprehensive monitoring of the state of the Tier-0 system, with both live and historical information. Subject to authentication and authorisation, it also allows the Tier-0 operators to dynamically change the Tier-0 configuration, and third-party users to request ad-hoc processing of selected datasets.

7.2 Distributed data management

ATLAS data are distributed over a worldwide network of data centres, also called sites, under the umbrella of the WLCG [298]. These sites are categorised into Tiers in a semi-rigid hierarchy with various capacities, duties, and responsibilities. CERN is the origin of detector data and thus the single Tier-0 centre. There are 11 ATLAS Tier-1 sites that are connected via dedicated national research and education networks (NRENs) with typically 10 to 100 GBit optical private networks [299]. These Tier-1 sites provide disk and tape storage and are charged with the perpetual archival of detector data. Around 70 ATLAS Tier-2 sites, typically hosted by national universities and laboratories, provide disk storage that is used for data processing and user analysis. Several additional computing centres complement the storage and computing available to ATLAS, including opportunistic sites (see Sect. 7.3.2) and sites dedicated to data analysis (see Sect. 8.1.7). As shown in Fig. 29, as of 2023 ATLAS has more than 800 PB of resident data. The total volume is split roughly equally between tape and disk. All these data are organised, managed, transferred, accessed, and accounted for via the Rucio distributed data management system (see Sect. 7.2.1).

7.2.1 Rucio

Rucio [280] is a flexible and modular software framework to build data management federations. It allows seamless integration of scientific and commercial storage, and their network systems. The data are stored in a single global namespace and can contain any payload. Facilities hosting the storage can be distributed at multiple locations, and belong to different administrative domains. Rucio was designed with more than a decade of operational experience in very large-scale data management, building from the work of its predecessor DQ2 [300].

As a technology, Rucio is location-aware and manages data in heterogeneous storage solutions and environments. It allows the creation, location, transfer, deletion, annotation, and access of and to the data. The major, standout feature is the orchestration of dataflows. These include both high-level dataflows, such as institutional or experiment policies, and low-level dataflows, such as specific user requests. Its main users are the PanDA and ProdSys2 workflow management systems, including the PanDA Pilot job processing framework, as described in Sect. 7.3. Individual users have command-line interface and web-based access through dedicated clients, both for access to data and for the manipulation of rules for data movement and storage.

At ATLAS mid-2022 scale, the interaction rate of operations with the Rucio system is typically beyond 200 Hz and often reaches 500 Hz, from distributed clients across the world interacting with the single Rucio instance at CERN. This includes diverse operations such as registering new files, searching for data, downloading files for processing jobs, scheduling files for deletion,[21] or modifying metadata. Rucio handles on average 1.5 million file transfers per day, with a peak rate of three million files per day. On average around 8 PB of data is moved per day from job and user upload/download, peaking at 12 PB per day. Transfers between data centres accounts for 2 PB per day on average, peaking at 4 PB per day. The high-availability deployment

[21] See Sect. 7.2.3 for the reason behind scheduling files for deletion, rather than deleting them immediately.

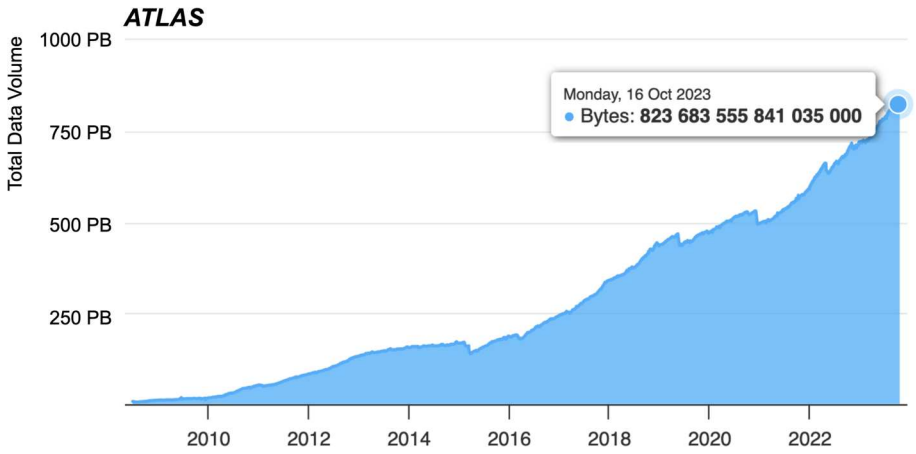


Fig. 29 The evolution of the ATLAS total data volume over the last 15 years, since before Run 1

of RUCIO is done via KUBERNETES [301] in the CERN data centre and is built atop free, open-source technologies. Additionally, extensive monitoring solutions were built and are the cornerstone of daily operations.

RUCIO was principally developed by and for the needs of the ATLAS experiment, with a view to eventually making it community open source software. It has since matured into the de-facto general solution for scientific data management. Now, it is developed, used, and supported by a multitude of different communities from diverse scientific fields, from neutrino experiments and dark matter searches, to astronomical observatories. As software, it is both free and open source, provided under the APACHE 2.0 License [27].

7.2.2 Dataset nomenclature

ATLAS datasets follow strict nomenclature rules to ease findability and ensure that dataset contents can be quickly identified. The names can be generated algorithmically, are unique, and are case-insensitive, although the original case is preserved throughout the system.

The dataset name is a series of fields separated with a '.'. For MC simulation, the fields are:

1. The project, a short indicator of the MC simulation campaign and, when relevant, centre-of-mass energy ($<$ 15 characters);
2. A numerical dataset identifier ($<$ 8 characters);
3. A short description of the physics described by the dataset ($<$ 50 characters);
4. The production step that generated the dataset (e.g. 'simul' for simulation or 'evgen' for event generation; $<$ 15 characters);
5. The data type (e.g. 'EVNT', 'HITS', or 'AOD'; $<$ 15 characters);
6. A series of processing tags, called AMI tags (see Sects. 6.3 and 3.2), indicating the configuration of the software that was used for each production step in the creation of the dataset, separated by underscores. Each step is represented by a letter, followed by the numerical index of the configuration, stored in the AMI database (e.g. 'e1234' for an event generation configuration; $<$ 32 characters in total); and
7. Optionally, a production task index and sub-task index.

An example of an MC simulation dataset name is 'mc15_13TeV.300402.Pythia8B_A14_CTEQ6L1_Bs_mu3p5mu3p5.recon.AOD.e4397_s2608_r6869'. In this case, 'mc15_13TeV' is the project; 300402 is the dataset identifier; the production step is 'recon' (reconstruction); the data type is 'AOD'; and the configuration tags are 'e4397', 's2608', and 'r6869' for event generation (e), simulation (s), and reconstruction (r). The description means that the sample was generated with the PYTHIA8B event generator (see Sect. 4.1) using the A14 tune of underlying event and hadronisation parameters in PYTHIA and the CTEQ6L1 parton distribution function set. The sample contains $B_s^0 \to \mu\mu$ decays, where both the muons are required to have $p_{\mathrm{T}} > 3.5$ GeV. Although not trivial to decode, with some experience these labels are enough to understand the contents of the sample.

For detector data, the fields are:

1. The project, indicating the year and, when relevant, the centre-of-mass energy for collision data;
2. The unique run number for the dataset, established by the data acquisition software;
3. The name of the data stream (e.g. 'physics_Main' for the most commonly analysed data stream [18]);
4. The production step that generated the data, similar to MC simulation;
5. The data type, similar to MC simulation;
6. A series of processing tags, similar to MC simulation; and

7. Optionally, a production task index and sub-task index, similar to MC simulation.

One example is 'data15_13TeV.00284484.physics_Main.merge.AOD.f644_m1518'. In this case, 'data15_13TeV' is the project; 284484 is the run number; the data are from the physics main stream; the data are merged (following the reconstruction, in this case); the data format is 'AOD'; and the production tags are 'f644' and 'm1518' for prompt reconstruction at the Tier-0 site (f) and merging (m).

User- or group-defined datasets are required to follow a less-strict set of rules; they must begin with 'user.' followed by the user name, or 'group.' followed by the group name.

RUCIO, used to organize the data, requires each dataset to be defined within a scope. For MC simulation and detector data, the scope is identical to the project in the dataset name. For group and user data, the scope is 'group.' followed by the group name, or 'user.' followed by the user name. Datasets can also be grouped into containers (groups of like datasets) within RUCIO; containers follow the same naming conventions where possible.[22]

These conventions were established before the start of data collection and were only carefully updated since. This ensures that any ATLAS dataset in the entire storage system can be quickly and easily identified by any users familiar with the naming conventions.

7.2.3 Data policies and life cycle

One key feature of ATLAS data is immutability. Datasets are locked once production is complete, and no additional files can be added, nor removed except in rare cases of data loss. Changes are not allowed; if a fix must be applied, an additional processing step is undertaken and the pre- and post-fix data can be retained in the system independently if necessary. This ensures that all data produced centrally (whether MC simulation datasets or real detector data, at any stage of processing) can be reproduced at any time, and the full chain of provenance is known.

Data on disk and tape are managed in slightly different ways. Disk-resident data are divided into three categories:

- *Persistent* data are pinned on disk for an indefinite period of time,
- *Temporary* data are pinned for a limited period of time, and
- *Cached* data are not pinned.

[22] For example, a container of multiple data runs follows the same general scheme, but does not specify a unique run number if it contains many runs. It may instead specify a Period (see Sect. 6.3) based on which the runs are grouped.

When non-user-analysis data are produced by Grid jobs they are copied to their final location and pinned there as persistent data. These data cannot be deleted except through one of the mechanisms described later in this section. Data that are moved around by the production system to be used as input for jobs (including data that are replicated from tape) are pinned on disk for a limited time and are thus labelled temporary. Once this time runs out, the data are considered cached and are eligible for deletion. The total volume of data stored on disk by ATLAS and the distribution within these different categories is shown in Fig. 30.

Each site hosting ATLAS data defines a quota of space available for use by ATLAS, and publishes frequently the capacity, used and free space. RUCIO sets a watermark just below the defined capacity and aims to keep the used space at this watermark. If the site reports used space over the watermark, RUCIO deletes cache data in order of least recent access until the watermark is reached. In this way, ATLAS uses as much disk space as is possible, unpopular data are automatically expunged from disk and popular data are likely to stay and be reused.

A site can be assigned as a *nucleus* when it can reliably aggregate output by *satellite* sites. The nucleus designation is made manually by operators in the Computer Resource Information Catalogue, CRIC [23], and can change over time. Job output data are transferred from the satellite sites and aggregated at the nucleus for the full task. The output is kept at the nucleus for archival or further processing. Brokering ensures that jobs are spread across satellites, while taking the available free storage space of the nuclei into account.

Data on tape are managed in a much less dynamic way, as the medium is not designed for rapid turnover. Obsolete data on tape are marked as cached, but they are only deleted in organised deletion campaigns coordinated with the sites, which take place at most once per year. This is because space freed from deleted data is usually only reclaimed when tape cassettes are *repacked* (i.e. the remaining data on each cassette are packed into a new one).

The policies for data replication and lifetime are designed to optimise the availability and redundancy of data for processing and analysis, whilst ensuring that the limited storage capacity is effectively used. The data replication factor and lifetime of the data are heavily dependent on how actively the data are used and their reproducibility in the case of loss. The policies defining whether to store data on disk or tape also depend on the access patterns and expected lifetime of the data. Raw data from the detector, for example, are always copied both to tape storage at CERN and one Tier-1 centre, because they are read infrequently and to ensure a negligible risk of data loss (only one raw data file was lost since the start of ATLAS operations). On the other hand, DAOD datasets used for analyses have at least two copies on disk so those data are always available, but are not replicated to

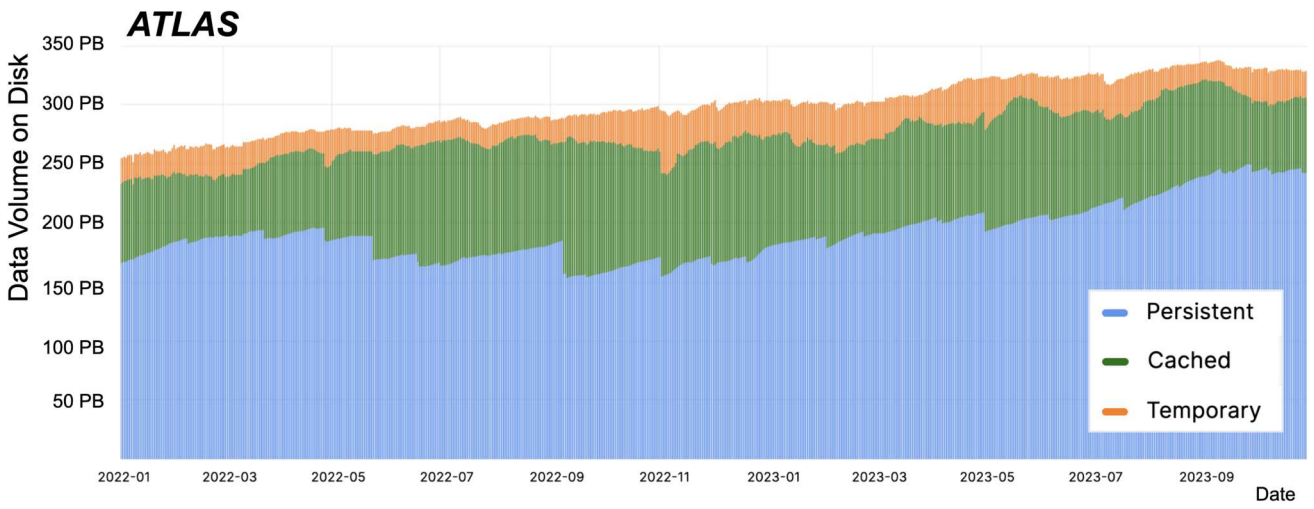


Fig. 30 The distribution among persistent, temporary, and cached of ATLAS data stored on disk in 2022–2023

tape because new versions are frequently produced and hence each version has a limited lifetime.

The replication of new data to match the policies is handled by RUCIO subscriptions, which automatically create copies of data in the required places. Archiving to tape of data produced on the Grid (HITS and AOD, and reprocessed data AOD) is done through a separate asynchronous process that delays the archival until one month after the data are produced, to avoid bad data being written to tape that are then immediately deleted.

An important shift in the ATLAS data model in recent years is from viewing tape as a pure archive or backup to be used in limited cases to a more active use of tape as the main data store, and disk as a cache to which data are staged from tape temporarily and then deleted. This change required detailed investigation into how sites' tape storage was organised and developments in ATLAS services for optimally reading data back from tape. This work was carried out under the auspices of the Data Carousel project [302] (see Sect. 7.3.3), which was successfully implemented during LS2. The amount of data staged from tape increased from 20 PB in 2018 to 130 PB in 2021, without disrupting writing to tape (which increased from 47 PB in 2018 to 75 PB in 2021).

Another important handle in controlling what data are cached on disk is the automatic deletion of unused data from disk when there is an archival copy on tape. This mechanism can be tuned to be more or less aggressive according to how much cache space is available globally, which processing campaigns are planned and so on.

The life cycle of data is controlled by the *lifetime model*, through which data are deleted from all centrally-managed storage a certain period of time after they are produced. The lifetime depends on the type of data and their expected use; following the examples above, raw data have an infinite lifetime and DAOD data have a lifetime of six months. The lifetime may be extended in two ways. When data are accessed (i.e. used as input to a PANDA job or manually downloaded using RUCIO), their lifetime is extended (usually by six or twelve months). Alternatively, an analysis team may ask for an extension at the point when data become eligible for deletion, for example if the team is midway through a publication procedure and the data may be needed to recreate results. The lifetime model has worked well to control the disk and tape space used by ATLAS, allowing obsolete data to be easily removed to free up space for more copies of actively used data. However, with the length of individual ATLAS analysis efforts often stretching to many years, the amount of data that is not active but was requested to be kept 'just in case' is steadily growing and is now roughly 10 PB. Recent studies have shown that only a fraction of these data are accessed after a lifetime extension request. Therefore, work is ongoing and will continue throughout Run 3 to identify alternatives to keeping such data pinned on disk, such as moving to 'cold' storage or streamlining procedures to allow recreation of data on demand.

This lifecycle is applied to data on general ATLAS resources; some resources exist outside of these rules. Analysis groups (e.g. the Standard Model or Higgs group) have dedicated *group disk* pools where data supporting ongoing analyses can be stored without regard to the lifetime model. Similarly, institutes have local group disk areas where data supporting the local data analysis can be stored. Users or group space managers are responsible for creating rules to transfer data to these disk areas and for deleting data from them.

7.2.4 WLCG data challenges

In anticipation of HL-LHC data rates, the WLCG has decided to run increasingly challenging stress tests of the data infrastructure, including disk, network, and the tape systems. The plan is to run these tests every two years until 100% of the expected HL-LHC data rate is reached by 2027. Two target rates are defined: the *minimal* target, which is the rate expected with a rigid hierarchical computing model, and the *flexible* target, which is the rate expected with flexible computing models allowing transfers across and between all Tiers dynamically. The latter model is similar to the current computing model.

In October 2021, the first disk and network challenge took place with the aim of achieving a sustained overall transfer throughput of 10% of the expected HL-LHC data rate, with the minimal target corresponding to the expected Run 3 rate [303]. Consequently, this data challenge also served as a commissioning test for Run 3. The results are shown in Fig. 31, showing that the minimal target was easily met throughout the duration of the challenge and at peak rate the 'flexible' target was achieved.

Dedicated tape challenges focused on the capability of the Tier-0 and Tier-1 tape systems to handle the expected rate of Run 3 RAW data export, which is up to 8 GB/s from all physics streams combined. These tape challenges were successful, and these rates were met and even exceeded without any operational incidents during Run 3 data taking. Different streams are distributed across different sites, but all data within each stream are sent to a single site for a single run. The largest stream is produced at a typical rate of 3.5 GB/s, and the final tape challenge in March 2022 showed that all Tier-1 sites were capable of handling this data rate.

In mid-2022, the planning for the follow-up data challenge has started, with a projected date in early 2024. The major features to be tested are the new token authentication mechanisms (see Sect. 7.2.5), software-defined networks, and much-improved monitoring. Potential new dataflows introduced by HPC centres (see Sect. 7.3.2) and dedicated analysis facilities (see Sect. 8.1.7) will be folded into the plan once their usage is more clearly articulated within the community.

7.2.5 Further WLCG infrastructure changes during Run 3

A virtual organisation-based (VO-based) security architecture, using X.509 certificates, has been a reliable solution for authentication and authorisation in ATLAS, but has also showed usability issues and required ad-hoc services and libraries in the Grid middleware. The need to move beyond the VO-based scheme was recognised as an important objective in WLCG Authentication and Authorisation Infrastructure (AAI) [304], to overcome the usability issues of the current AAI and embrace recent advancements in web technologies. A token-based AAI was implemented in ATLAS using the INDIGO Identity and Access Management service [305], fully compliant with OIDC/OAUTH2.0 and capable of identity federations among scientific and academic identity providers. The deployment of this new AAI infrastructure across services is ongoing.

Until 2023, all WLCG experiments utilised the HEPSPEC06 [306] benchmark for accounting and pledges. However, this 32-bit benchmark is no longer a reliable indicator of HEP workloads, whilst the underlying SPEC-CPU 2006 benchmark [307] is no longer supported since 2018. Moreover, there is a coming need for a benchmark that caters to non-x86 architectures such as ARM processors and GPUs. To address these issues, a task force was established by the WLCG in November 2020, to identify a new benchmark based on the current workloads of the LHC experiments and a transition plan. A new benchmark, HEPSCORE, which includes both x86 and ARM processors, was developed [308] and deployed in April 2023. The experiments now use this new benchmark for resource requests, and sites are expected to score new hardware purchases using this benchmark instead of HEPSPEC06. The benchmark is made up of a variety of workloads [309] from all four LHC experiments and Belle II; the initial version is HEPSCORE23. The transition to HEPSCORE23 will enable more accurate accounting and pledges, providing a more realistic representation of HEP workloads. Furthermore, HEPSCORE23 was normalised to the old HEPSPEC06 benchmark, so that sites do not have to re-benchmark existing systems, and only use it to measure newly purchased hardware.

7.2.6 DDM R&D projects

In addition to the ongoing development work and data challenges, which come with their own research and development communities, several distributed data management-specific R&D projects were started: most importantly, integration with commercial clouds, dynamic data handling, and distributed caching.

The first of these R&D projects, underway in Run 3, is the integration of RUCIO with commercial clouds, specifically Google Cloud [310], Amazon Web Services [311], and Seal Storage Technologies [312]. The integration of such cloud services is reaching a stage where these types of storage can be included in the distributed data management system beyond simple demonstrators. The main benefit is that funding agencies and institutes can achieve more flexible storage installations (including, potentially, the purchase of cloud-based resources), while RUCIO takes care of abstracting the peculiarities from the scientists.

A second R&D project is revamping the policies that drive ATLAS dataflows, mostly given by experiment data agreements, processing requirements, and MoUs, as well as oper-

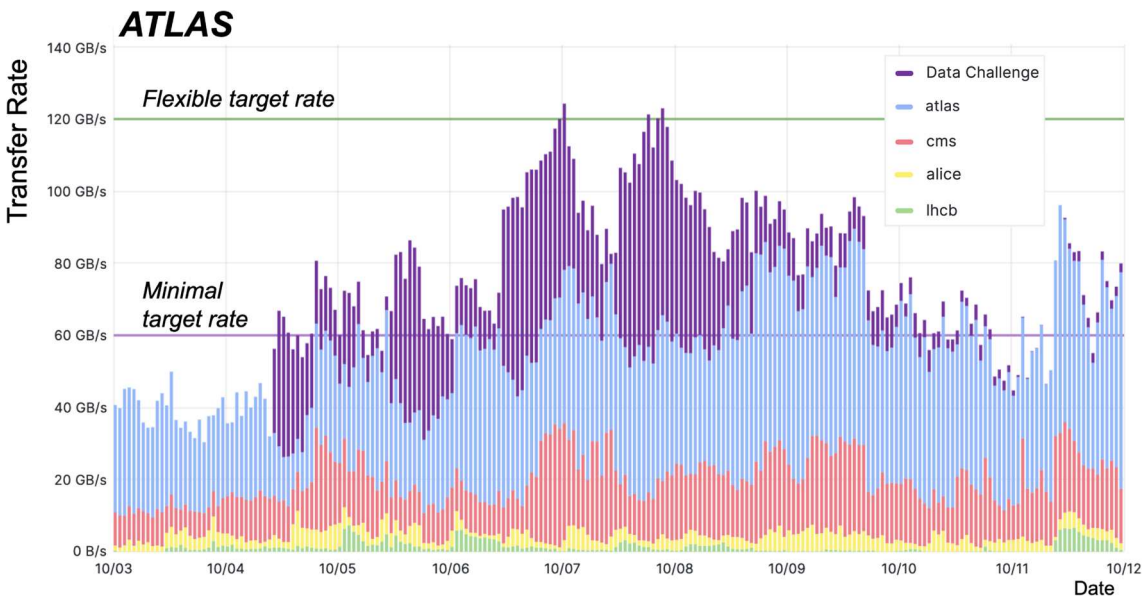


Fig. 31 In purple, additional 'Data Challenge' traffic injected to reach target rates during the October 2021 Data Challenge, along with the standard operational traffic from the four LHC experiments

ational and infrastructural needs. To address this, a working group was formed to address dynamic data handling in a coherent and consistent way. The working group will study and tune ATLAS dataflows, most importantly placement of new data, rebalancing between data centres, deletion of obsolete and unused data, and data replication both for production and analysis. The eventual goal is to reduce workload execution time, reduce data access time and make better use of available storage through improved placement, movement, addition, and deletion of replicas of ATLAS data, under the hard constraint of limited storage space.

Finally, a third R&D project deals with distributed caching to reduce wide-area-network (WAN, site-to-site) traffic, reduce latency to the analysis software, and eventually to increase CPU efficiency. This is an integration of RUCIO with the XCACHE system [313] to assign processing jobs not only with the common 'job to data' paradigm, but to allow a more flexible approach that includes significant staging of data to sites with extra computing capacity. During LS2 a prototype deployment was put in place. First observations have already provided insights, including increased cache use and reduced WAN traffic. During Run 3 this mechanism will be exercised with real analysis use cases and, if proven successful, will be integrated into RUCIO.

7.3 Workflow management

A large variety of workflows are required by the ATLAS experiment, including data processing (and reprocessing), MC event generation, simulation and reconstruction, and derivation data production for physics groups (see Sect. 4 for details). This is in addition to data analysis conducted by physics groups and individual users. Figure 32 shows the number of running ATLAS jobs since the beginning of 2022, for each of the different activities.

To fulfil this workload demand, the ATLAS experiment uses more than 250 computer centres worldwide, primarily made up of the WLCG Grid sites described in the previous section, but also including HPC centres and national, academic, and commercial cloud computing resources. Figure 33 again shows the number of running ATLAS jobs since the beginning of 2022, but this time grouped by resource type. It can be seen that although the contribution from the Grid dominates, as expected, there are many running jobs on opportunistic resources such as HPCs and cloud, the latter contribution mainly coming from Sim@P1[23] (see Sect. 7.3.2). Further contributions are present from assigning dedicated production tasks to both HPCs ('hpc_special') and clouds ('cloud_special'), where the latter is mainly made up of BOINC [21,22] job submissions and jobs running on the Tier-0 site. These additional resources are described in Sect. 7.3.2.

7.3.1 A universal workflow management system for ATLAS

The processing, simulation and analysis of data from modern HEP experiments requires the orchestration of multiple diverse computing facilities. This section describes

[23] Because originally Sim@P1 was managed using OpenStack, it was accounted for as part of the cloud resources. Although this is no longer the case, for historical consistency it remains in this accounting category.

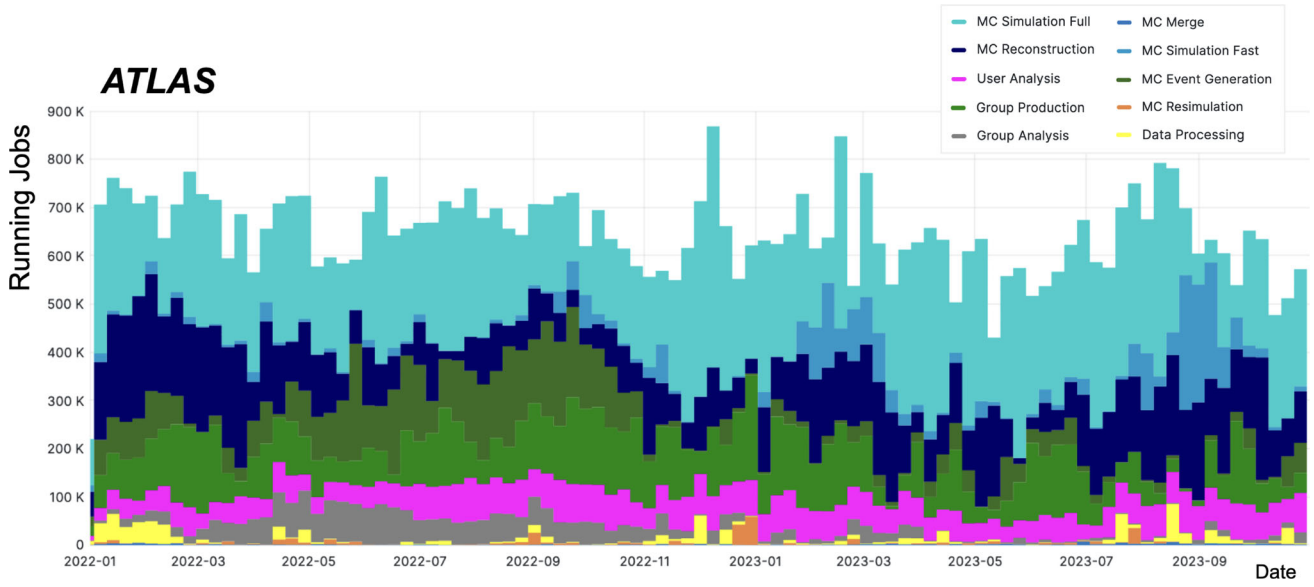


Fig. 32 Running ATLAS jobs since the beginning of 2022, grouped by activity

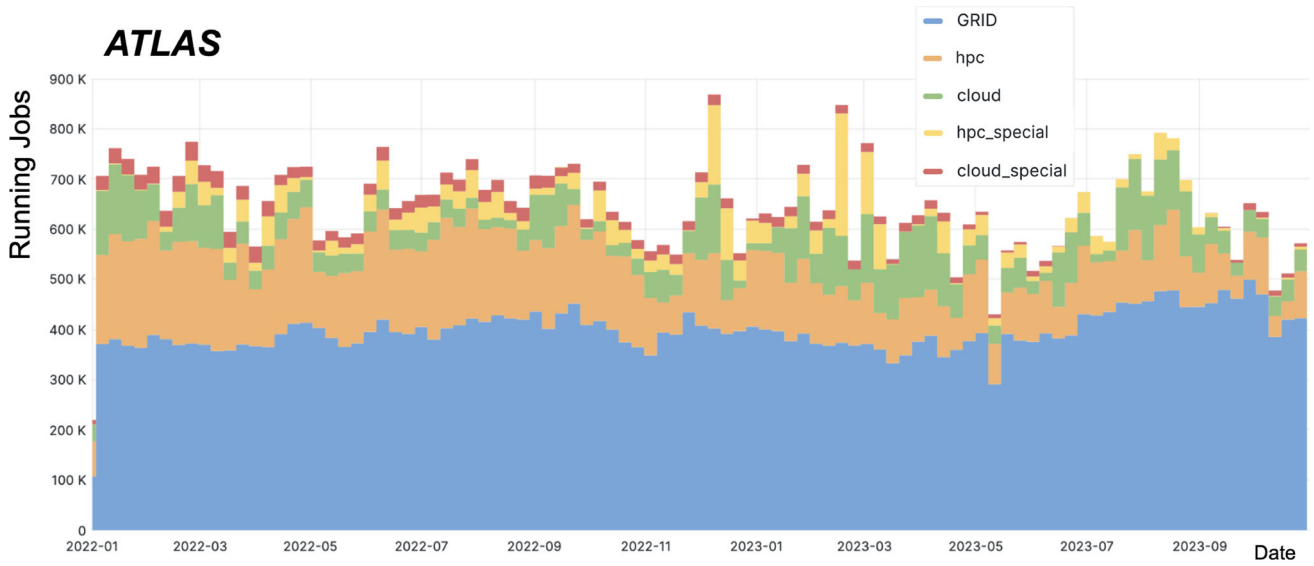


Fig. 33 Running ATLAS jobs since the beginning of 2022 grouped by resource type

the methods developed and the approaches taken to arrive at a universal workflow management system for ATLAS, ProdSys2-PanDA [115,314], which has been used since the beginning of 2014. New elements such as iDDS [315] and Harvester [316] have since been developed and brought into production before the beginning of LHC Run 3 to address the heterogeneity of resources and the introduction of new data processing scenarios such as the Data Carousel [302] (Sect. 7.3.3).

Design concept When designing a suitable workflow management system for a distributed and heterogeneous computing infrastructure, there are many aspects to consider. The primary role of such a system is the management of the various production and analysis workloads, as well as defining how the resources are to be used and the application of appropriate fair-share policies. The system should also work regardless of the type of infrastructure and level of heterogeneity and be dynamic enough for possible changes in the available resources. The various resources used and the associated access protocols, are described using CRIC [23]. User identification and security considerations are handled via the use of X.509 certificates, which are to be replaced by tokens during Run 3 as described in Sect. 7.2.5.

After analysing the various classes of workflows, which are described in Sects. 4 and 5, the essential components

of the production system were identified and a logical data model built. The following entities are defined:

- A *request* is the upper level of abstraction, and is made of tasks of one type. A typical example would be the (re)processing of all data for a certain period or a collection of MC simulations with common parameters, which are to be used together to compare to a given period of data. A request may comprise multiple steps, where event generation is one step, for example. Each step on each input dataset is represented as a separate task.
- A *task* is an entity for passing parameters to the payload submission system, and is composed of jobs. The result of the task is a set (or several sets) of files, organised in datasets, typically with one dataset per task per output file type.
- A *job* is a single executable workload, where each task consists of one or many jobs (up to ten thousand). A job is executed on a single Grid computing element, opportunistic CPU or worker node. A job may have input data and writes the result of the work to an output file or files.

The system architecture is designed to ensure continuous and optimal access of the scientific community to computing resources, which is achieved with an extensible layered architecture. Figure 34 shows the different levels of the workflow management system, where the relationship between the relevant entities is schematically described on the right [317]. The implementation of these levels as part of the actual system, shown in the left of Fig. 34, is described in the following.

ProdSys2 web user interface The ProdSys2 [314] Web User Interface (Web-UI) is used by the ATLAS production managers to interact with the task management system. A production manager represents a group activity in the ATLAS experiment and defines workflows for their group: Data (re)processing, MC production (event generation, simulation and reconstruction), and Derivation (DAOD) production. Each activity has its own workflow and specific task requirements.

The heart of a work process with the Web-UI is the request. The production manager creates a request using the Web-UI, and defines its parameters and input datasets. For each input dataset, a sequential set of steps to be executed is defined, which is translated by the system into a sequence of tasks to be executed. A single step might correspond to event generation alone, with detector simulation as a second step following it. The data model features the logical division of the request into horizontal *slices* defining all the steps beginning from a single input dataset. Each slice may comprise several sequential steps in one production workflow, each executed as a task, which itself is divided into jobs.

The Web-UI is also used to monitor the progress of running tasks and adjust them as required. Active bookkeeping is provided, metadata is stored, including user-defined tags, and the aggregation of task statistics is done, which may be used for the fine-tuning of running and historical task analysis. Advanced error handling and reporting is included to help quickly understand the root of any problem and to fix it by redefining and resubmitting the task. The chaining together of slices in separate requests is also possible and successfully used, for example when connecting MC event generation, simulation and reconstruction together with derivation production.

DEfT The Database Engine for Tasks (DEfT) is a top-level sub-system and is the engine beneath the Web-UI. DEfT accepts task requests via a dedicated user interface or from prepared lists (e.g. a text file, or Google or Excel document). DEfT processes requests and is responsible for the formation of processing steps, tasks, and input data and parameters.

ProdSys2 sets default parameters via DEfT, for example memory and CPU limits, which are often not defined by the user. ATLAS has many default parameters for complex workflows, which may be viewed and changed via a special ProdSys2 interface. After parameters are set, the system checks for their consistency and compatibility. For example, it is verified that the input dataset has enough events or that the parameters defined in the task are valid for the ATLAS software release to be used. A further check is performed to see if similar tasks were defined and successfully executed, to avoid event duplication. During task definition, if ProdSys2 detects that some of the events in an input dataset were already used for a given configuration, DEfT sets an offset (in files when input datasets are used, or in event numbers and random number seeds for steps without input data like event generation) and the new task uses only unique events.

JEDI The Job Execution and Definition Interface, or JEDI, is a middle-level sub-system that receives formalised job descriptions from DEfT. JEDI dynamically determines the number of jobs for each task and is responsible for launching and executing individual jobs. JEDI verifies information about the data via RUCIO and about the job queues via CRIC. A common database is used by JEDI and PANDA (see Sect. 7.3.1) to store information about the status of jobs and tasks.

PANDA PANDA [115] is the engine of the underlying system and the most complex layer. It determines which resources are used when for each of the jobs, receives information from Pilot jobs (see below) and the CRIC information system, and manages the progress of jobs. The various PANDA components are described below.

A database of jobs and tasks is used system-wide, storing comprehensive static and dynamic information and meta information about all jobs and tasks defined, running, and

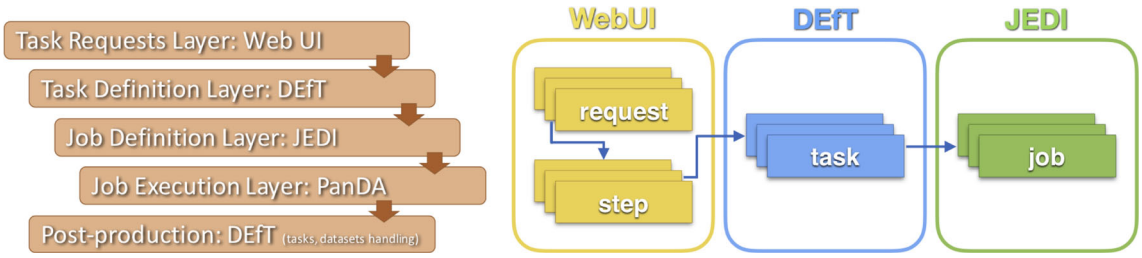


Fig. 34 The several levels of the ATLAS workflow management system. Figure from Ref. [317]

completed in the system, including the history of their execution and errors that have occurred in the process.

Pilot jobs are used to collect information about the state of computing resources and requirements of a task. Jobs are submitted to successfully activated and verified Pilots by a PANDA server based on resource selection criteria. Late binding of jobs to the execution location prevents delays and failures and maximises the flexibility of resource allocation for a job, based on the dynamic state of processing resources and job priorities. A Pilot is the main 'isolating layer' for WFMS, encapsulating complex heterogeneous environments and the Grid interfaces and tools with which the WFMS interacts. Pilot jobs also serve to identify the resource requirements of a task and adjust subsequent job definitions, for example to request additional memory. In the case of failures, pilots ensure that only a small fraction of the task runs, and the task is marked as failed before wasting significant resources.

The *Intelligent Data Delivery Service* (iDDS) [315] is an additional layer developed to orchestrate workflow management and distributed data management systems to optimise resource usage in various workflows. The input data to a task are dynamically transformed, so that the data preprocessing, delivery, and main processing in each workflow are decoupled, which allows them to run asynchronously. iDDS was introduced in 2020 to address inefficiencies in the Data Carousel (see Sect. 7.3.3), which previously required many input events due to constraints in the workflow, creating delays in bulk reprocessing campaigns. iDDS propagates the detailed information about the input data status from RUCIO to JEDI, allowing the incremental release of tasks so that processing can begin even if input data are only partially staged-in. iDDS is now also used in new workflows such as some AI/ML workflows [318,319], and supports several workflow definition languages.

Harvester [316] mediates the control and information flow between PANDA and the resources to enable more intelligent workload management and dynamic resource provisioning based on detailed knowledge of resource capabilities and their real-time state. Harvester was designed around a modular structure to separate core functions and resource-specific plugins, simplifying the operation with heterogeneous resources and providing a uniform monitoring view.

PANDA also includes functionality to automatically retry or re-broker jobs. Sometimes, site failures can be identified and jobs can be re-directed to an alternative site that has a copy of the input data. Additionally, a catalogue of known errors is kept, defining rules for the number of retry attempts that should be made when a specific error is encountered. With this functionality, jobs that are suffering from system or site failures can be retried and moved around the Grid, while jobs with clear failures that will not converge can be stopped before significant resources are consumed.

The concept of *global shares* is used to set the amount of resources available instantaneously for a certain activity, for example MC simulation, as a fraction of the total amount of resources available to ATLAS. Global shares have recently been updated to be measured in the new benchmark, HEP-SCORE23 (see Sect. 7.2.5); previously, they were measured in HEPSPEC06, the previous standard unit for core-power benchmarking used in HEP community. A nested structure of global shares is used, where siblings have the preference to occupy unused shares, before the unused share is taken by higher levels. For example, 75% of the resources might be assigned to Production (Level 1 share); of those, 25% might be assigned to Simulation (Level 2 share); and of those, 20% might be assigned to the MC16 simulation campaign (Level 3 share). If MC16 does not fully occupy its fraction of the resources, idle resources would be preferentially reallocated first to run other simulation tasks, then to run other production tasks, and then to run any available tasks. Within a share, jobs and tasks are assigned priorities, allowing urgent tasks to run quickly, ahead of large, low-priority tasks. Tasks and jobs are assigned a global share at creation time, and jobs are sorted by priority and creation time within a particular global share.

The *BigPanDA* monitoring system [256] is a DJANGO-based [320] web application consisting of a set of standard monitoring pages and separate modules that can be plugged in or removed on demand. Data are collected from a variety of different sources including the PANDA database, the CRIC information system, payload logs of jobs from RUCIO, PANDA/JEDI logs from the ELASTICSEARCH cluster, and the MONIT GRAFANA instance where accounting data are available (see Sect. 7.4.1). The data flow of these sources is illus-

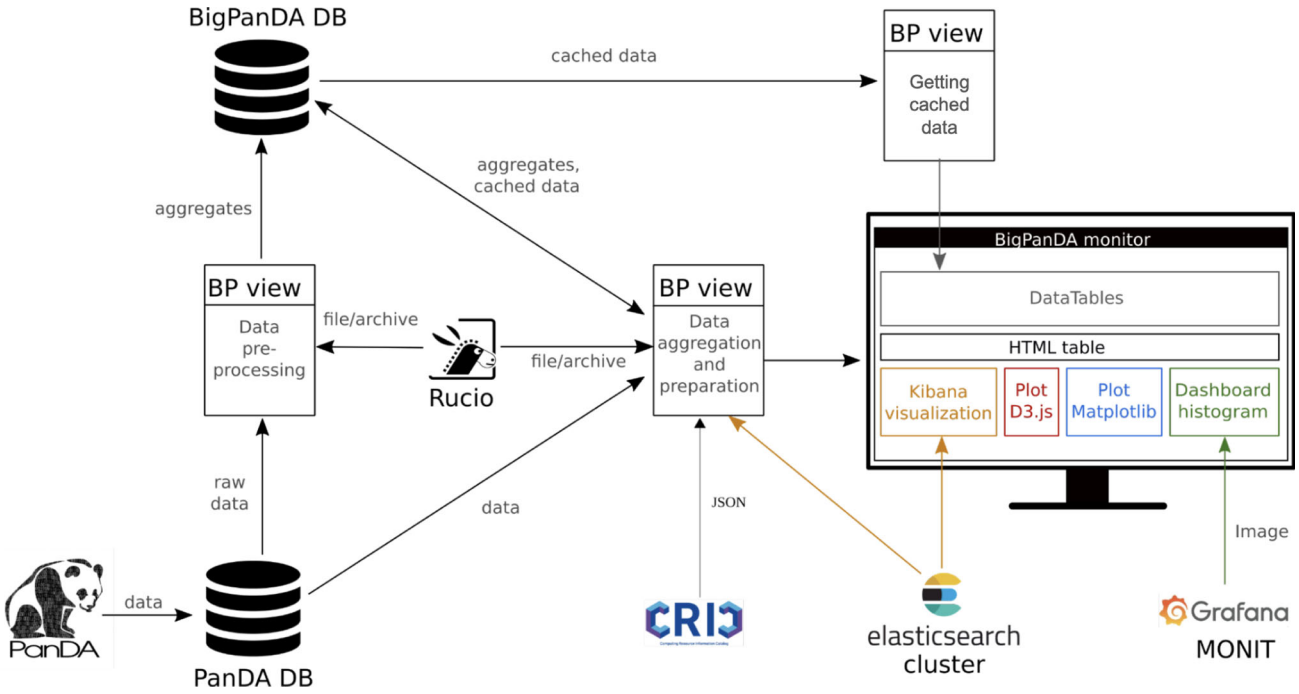


Fig. 35 Data flows in the BigPanDA monitoring system

trated in Fig. 35. More information about monitoring is provided in Sect. 7.4.

7.3.2 Opportunistic resources

One of the challenges of the ATLAS production system from the beginning has been the integration of a wide variety of computing resources whilst at the same time hiding the heterogeneity from the users. The initial computing model was based on the Grid [5], developed for the LHC experiments and realised via homogeneous resources situated in the computing centres distributed worldwide. This homogenisation was based on a list of software and hardware requirements to be met by the centres, where any remaining differences were absorbed by the associated Grid middleware.

More recently, new and often more exotic types of resources have become available to ATLAS, requiring adaptation and evolution of the production system to integrate them. Among these resources are the Tier-0 site when not in use for operations (see Sect. 7.1), HPCs, various cloud resource providers and other *opportunistic resources*,[24] such as those running volunteer computing initiatives like 'ATLAS@Home' [321,322].

Such resources may not be dedicated specifically to ATLAS and therefore miss critical component infrastructure, are often constrained in their use in that they are optimal only for some workflows, and may not be permanently available, depending on the allocated hours or budget. A measure of the success of the integration of these special resources is the production system's ability to effectively broker jobs to them, dispatching the appropriate job type, and to detect changes in resource availability in real time. These various additional resources are described in the following.

Use of high performance computing resources ATLAS has a long history of exploiting the potential of High Performance Computing (HPC) centres to provide additional computing cores for ATLAS workloads. Employing HPCs presents significant challenges, not least that access to such systems is usually strictly limited, so that connections to the nodes themselves are heavily restricted and installation of local software is tightly controlled. Furthermore, the CPU architecture may not be suitable for ATLAS software, the expected job structure and number may be atypical of an ATLAS workflow, and network issues may arise due to geographic location.

Nevertheless, ATLAS has successfully used a series of HPCs for more than a decade, many of which appeared or

[24] The name 'opportunistic' is meant to distinguish these resources from those upon which the experiment relies to deliver its primary physics programme. The WLCG Grid sites pledge resources annually to the experiment to satisfy that physics programme. Recently, because of the scale of these opportunistic resources and their consistent availability, some amount of these resources were accounted for in the core resource needs, to more fairly request the necessary resources from the Grid sites. Another term for these resources that is in use, therefore, is *unpledged*, to be distinguished from *pledged* resources at WLCG Grid sites and *beyond-pledge* resources that Grid sites deliver in addition to their pledged resources.

continue to appear near the top of the Top500 list of supercomputers [323]. These included Cori [324] at the National Energy Research Scientific Computing Center (NERSC), Titan [325] at Oak Ridge National Laboratory, both in the USA, SuperMUC-NG [326] at the Leibniz Supercomputing Centre in Germany, and Toubkal [327] at the African Supercomputing Centre in Morocco. Typically, these machines ran dedicated ATLAS Full Simulation tasks, delivered to the site with all required inputs and database information, allowing them to run in an isolated way via the use of edge services at each machine. In the case of Titan, this was mainly used in back-filling mode, where ATLAS exploited spare cores not used by existing tasks to run ATLAS Fast Simulation jobs.

More recently, Perlmutter [328] at NERSC, Karolina [329] at the IT4Innovations National Supercomputing Center in Czechia and in particular Vega [330] at the Institute of Information Science in Slovenia have provided significant additional computing resources to ATLAS. Karolina and Vega are part of the EuroHPC project [331]. The more widespread use and acceptance of native CVMFS and the adoption of containerised workflows by ATLAS has enabled a more generic use model of HPCs for ATLAS. All production workflows now run on most HPCs, allowing such machines to be fully integrated as additional resources, essentially no different to a standard Grid site. This has meant reduced operational load, more flexibility and periods where the number of available CPU cores available to ATLAS has more doubled relative to the WLCG pledged resources. Some HPC centres are sufficiently well integrated that they can run as part of standard WLCG sites, like MareNostrum [332] at the Barcelona Supercomputing Center in Spain. The large-scale use of these resources is expected to continue throughout Run 3 and beyond.

Opportunistic use of the HLT farm: Sim@P1 The ATLAS Trigger and Data Acquisition (TDAQ) high-level trigger (HLT) computing farm [333] is a critical part of online data taking for the ATLAS experiment. It facilitates the final selection of events to be stored for further physics analysis based on full event reconstruction [18]. It consists of 145,000 computing cores spread amongst three different flavours of hypervisors with RAM between 0.9 and 1.1 GB/core. The nodes are located within the private network of the ATLAS experiment, which is separated from the internet and are accessible to ATLAS Distributed Computing via a VLAN on a data link layer through two 100 Gbps connections. Networking and RAM per core are the two limiting factors that define the type of workflows that can be run efficiently on this resource.

Since 2013, ATLAS has been running MC full simulation in longer periods of inactivity of the LHC within the *Simulation at Point One* (Sim@P1) project on these resources [334]. Recently it was shown that they can also successfully be used for other workflows, such as MC reconstruction, whenever necessary, albeit with lower efficiency.

Current studies [335] show that thanks to improvements in the new ATLAS fast simulation, AtlFast3 (see Sect. 4.2.2), and optimisation of the job submission configuration, ATLAS is able to process many simulated events within one hour. This speed allows Sim@P1 to process simulation events between LHC Run 3 fills when the inter-fill break lasts longer than one hour. Furthermore, with the move of the project to Kubernetes, it may be possible to run Sim@P1 in parallel to the trigger whenever part of the compute resources are not needed by the ATLAS online system.

Volunteer computing: ATLAS@Home ATLAS@Home [321,322] is a volunteer computing project based on BOINC [21,22] for utilising the free CPU cycles of volunteer computers around the world for MC simulation of the ATLAS experiment. ATLAS@Home was one of the first volunteer computing initiatives in HEP.

This resource is fully integrated into the Grid computing infrastructure of the ATLAS experiment through PanDA. Detector simulation tasks assigned to this resource are assigned to a single PanDA queue that covers all the volunteer resources. The jobs from the tasks are pulled by an ARC Control Tower [336], and then the control tower submits the jobs to an ARC compute element, which forwards the jobs to the BOINC server. The ARC Control Tower and Compute Element handle all interactions with PanDA and Grid services and hence no credentials are required on the volunteer hosts. BOINC clients request jobs from the BOINC server and process them whenever they have idle CPU cycles.

In addition to members of the public, ATLAS@Home was used in a back-filling mode at several Grid sites to make full use of CPU not normally used due to job or scheduling inefficiencies [337]. BOINC has become a very reliable computing resource for the ATLAS experiment; major simulation tasks are run, and it contributes about 1% of the CPU available to ATLAS computing daily.

7.3.3 The data carousel

The evolution of the computing facilities and the way storage is organised and consolidated will play a key role in how the LHC experiments will address the possible shortage of resources in the HL-LHC era. In particular, to reduce storage costs to the experiments at a time when the data volume is expected to significantly increase, it is anticipated that the use of tape may be expanded. To address this data handling challenge, the Data Carousel project [302] was established to study the feasibility of directly receiving input data from tape for various ATLAS workflows.

The Data Carousel is the result of a successful orchestration between the workflow management system ProdSys2-

PanDA, the distributed data management system Rucio, and the tape services at the Tier-1 sites. It enables a bulk production campaign, with input data resident on tape, to be executed by staging and promptly processing a sliding window of a fraction of the input onto buffer disk such that only a percentage of the data is pinned on disk at any one time. The production system follows site-specific preferred staging profiles, provided by the Tier-0 and each Tier-1 site, which define the upper and lower limits of concurrent staging requests that can be handled. The typical staging pattern over a two-week period can be seen in Fig. 36, where the peak data staging performance reaches over 15 GB/s and the colours represent different Tier-1 sites.

Staging inputs from tape rather than having them resident on (more expensive) disk allows the dedication of significant disk space for more popular data such as DAODs. Without Data Carousel, a large fraction of HITS or AODs would need to be kept permanently on disk to run regular processing campaigns. In the case of data reprocessing campaigns, this would require complete (and large) RAW datasets to be pre-staged onto disk before they could begin. The Data Carousel model therefore brings significant disk space savings for ATLAS, where, for example, less than half of the AODs are on disk at any one time.

To promptly process the staged data and to improve turnaround time, iDDS (see Sect. 7.3.1), was developed and integrated with the existing system. The collaboration between the Data Carousel and iDDS R&D projects is an excellent example of early HL-LHC R&D delivery and commissioning for LHC Run 3.

7.3.4 WFMS R&D projects

In addition to the distributed data management research and development projects described in Sect. 7.2.6, several areas related to ATLAS workflow management are being investigated in further R&D initiatives.

The integration of additional HPC resources is a priority for ATLAS, so that the experiment can benefit from new machines in the US, in Europe via the EuroHPC project, or anywhere they become available. Whilst an easier integration than a decade ago is possible thanks to the evolution described in Sect. 7.3.2, each new machine still presents a different challenge. Discussions are continuously underway to identify possible new HPCs of interest, whether in production or in planning, to understand if they might provide significant resources to the experiment.

An evaluation of commercial cloud resources complements the HPC effort, and includes the commissioning of an ATLAS site at Google [338], together with a Total Cost of Ownership analysis [339]. Both HPC and commercial cloud resources provide an opportunity to investigate and integrate non-x86 resources before such technologies are provided by WLCG pledged resources. This includes both GPUs and ARM architectures, which are expected to be more prevalent in the future (see Sects. 7.2.5 and 9).

Related to the dynamic data handling R&D project described in Sect. 7.2.6, the future creation, storage and lifetime of DAODs is under investigation. The benefits and consequences of the adoption of a model where DAODs are recreated on demand will be evaluated.

7.4 Data monitoring and analytics

The distributed computing systems produce a wealth of data that can be used to monitor the health of the many servers and services, and at the same time investigate the ways collaboration members use these services and find ways to optimise the overall resource usage.

7.4.1 ADC monitoring

The monitoring infrastructure collects information from the workflow management system PanDA [115] and the data management system Rucio [280], complements it with static information about site configurations, aggregates it into time bins and stores it in appropriate data storage systems.

Figure 37 shows a schema of the data flow through the main monitoring system based on Kafka [340], using the infrastructure provided by the CERN IT Monit team [341, 342]. Several data sources are queried or send periodically information to the central monitoring system; here the information is processed and aggregated using Apache Spark [343] jobs, and finally the records are stored in ElasticSearch [285], InfluxDB [344] or HDFS [345] systems (depending on their type). These data stores can be used as sources of data for visualisation in Grafana [346] or Kibana [347] dashboards, or treated interactively using the CERN Swan suite [348].

7.4.2 Dashboards

The same data may be used to feed dashboards that have several purposes. For example, information about successful and failed jobs grouped in time bins of 10 minutes and displayed over a few hours can be useful to detect faults in software or site operations, or grouped in bins of one week and displayed over several years can be used to prepare accounting reports. Several dashboards [349] were developed to cover the main needs, from short-term operation monitoring to long-term accounting, for the main ATLAS systems, PanDA and Rucio, as well as for the operation of WLCG sites that support ATLAS and data transfers between them. Auxiliary dashboards cover specific needs and other smaller systems. All dashboards can be customised in real time, changing the time range, the time binning, the quantities to be displayed

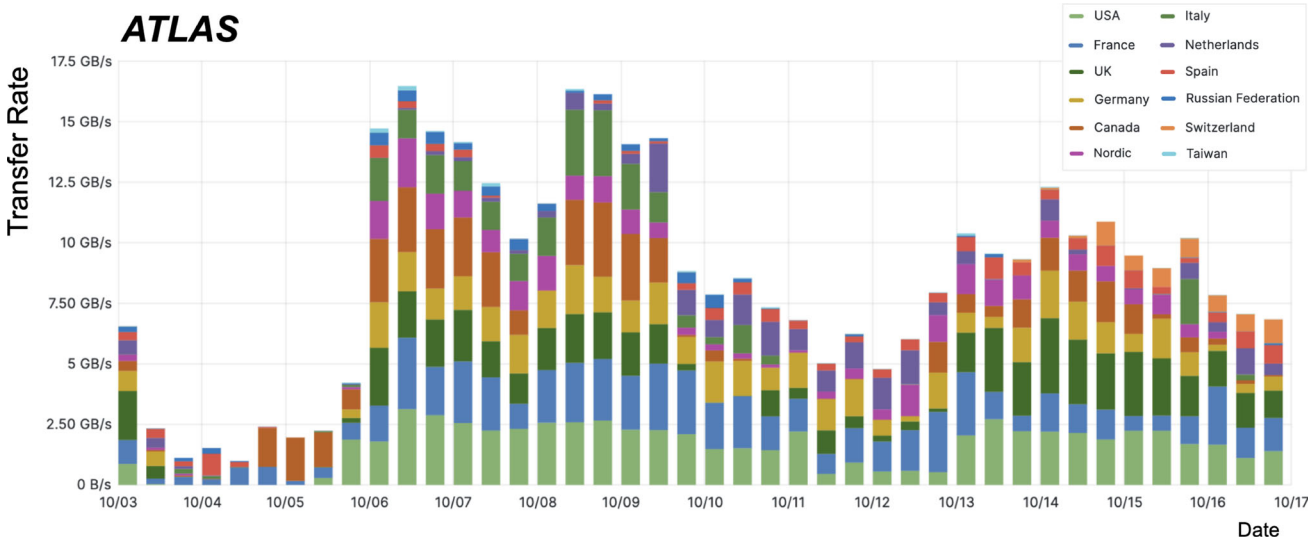


Fig. 36 The data-staging pattern over a two-week period corresponding to the Data Challenge (see Sect. 7.2.4) across many Tier-1 sites by the Data Carousel

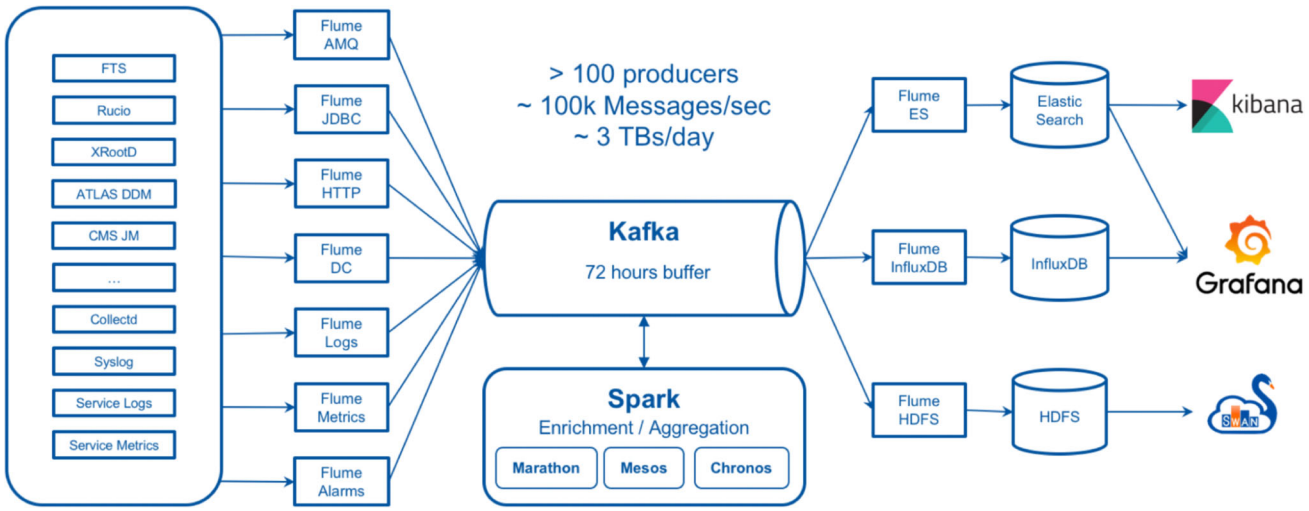


Fig. 37 The information flow through the monitoring infrastructure provided by the CERN IT MONIT team. Figure from Ref. [342]

and the data grouping. Figure 38 shows, as an example, a screenshot of the top part of the job accounting dashboard for a period of one month, in 1-hour bins, grouped by ATLAS activity; the pull-down menus at the top of the page allow filtering the data and customising the displays, and many other charts are displayed in the lower parts of the page.

The BigPanDA monitoring suite [256] is a web application that provides various processing capabilities and representations of the PANDA system state. Analysing hundreds of millions of computation entities, such as tasks or jobs, BigPanDA monitoring builds reports with several scales and levels of abstraction in real time. The reports allow users to understand specific failures or observe the broad picture by tracking the computation performance or the progress of a whole production campaign. BigPanDA is a core component of the PANDA system, commissioned in the middle of 2014, and is now the primary source of information for ATLAS users about the state of their computations and a key source of information for shifters, site operators and production managers.

7.4.3 ADC analytics

An ADC Analytics coordination activity was started to better link the diverse efforts in ADC, raise awareness of the existing projects, and share experience with tools and available data. In addition, the effort aims to improve the link between users (e.g. physicists doing data analysis) and developers of analytics solutions. Current activities cover a wide range of topics, including optimising the use of FRONTIER, anomaly

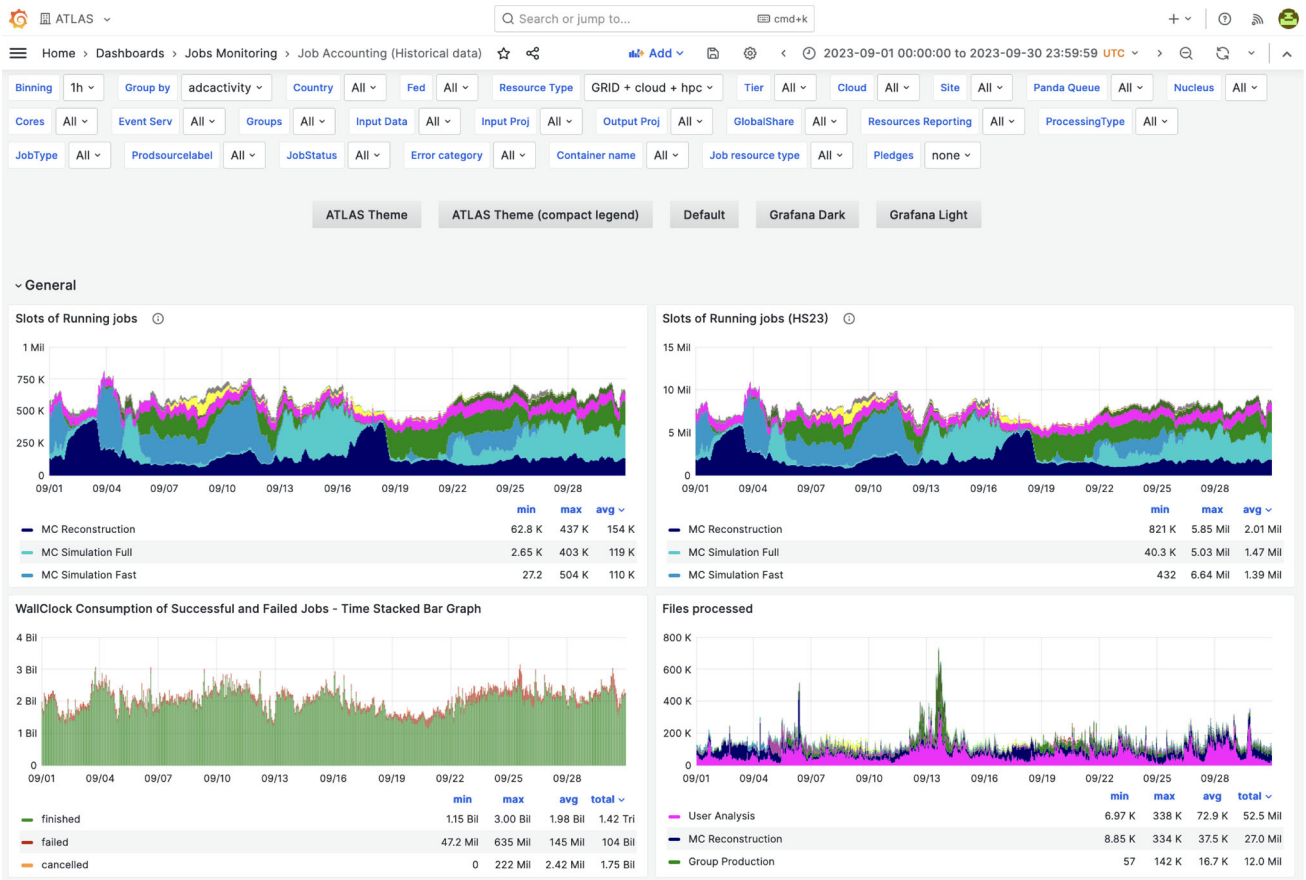


Fig. 38 An example of a customisable dashboard showing the number of jobs and CPU resources used during a set time period

detection in networks, data popularity, studies of access patterns at the file and tree level, prediction of the time to completion of tasks, development toward alerts for managers of operations, anomaly detection in data management, and the composition of dashboards that allow effective analysis of operational incidents and issues.

7.4.4 ElasticSearch *clusters for analytics*

The ElasticSearch infrastructures at CERN and at the University of Chicago (UC) form the backbone of many monitoring and analytics activities in ADC. At CERN, the cluster is a 20-core two-node OpenStack Kubernetes cluster that collects WFMS data and ships it to the ElasticSearch infrastructure at UC. The 29-node cluster at UC serves as the infrastructure for the 'ATLAS Alarm and Alert System' and provides resources to the new ADC analytics platform [350]. This platform will offer a common front-end to different ML solutions in analytics and monitoring and serve as a registry for analytics projects.

New data sources were added to ElasticSearch over the years. For example, xAOD data-access monitoring data enables the study of access patterns in great detail, with the potential to speed up data access by tuning the configuration of `TTreeCache` in real time, based on the profile of similar jobs. To keep Squid monitoring data (see Sect. 6.2.2) with high precision for a longer time period and allow analytics tasks to combine the data with other datasets, the monitoring data is now sent via Logstash to ElasticSearch. To discover unused files in disk storage, access information was exported from dCache [351] metadata databases to the ElasticSearch infrastructure. Two sites are currently part of the first implementation of this project.

8 Analysis tools, event displays, and tutorials

Performing physics analysis is the final step of the data and MC processing chain, with the aim to publish groundbreaking results using stringent scientific methods. A variety of tools are available that assist physicists with their analyses. Among these, event displays are special tools for visualizing recorded or simulated events and the detector for analysis purposes, and they can also be useful for visual inspection and troubleshooting. This section describes these tools and also the efforts to train physicists to use them.

8.1 Analysis tools

Analysis tools enable the data processing steps from centrally provided derivation formats for analysis to final results for publication and public dissemination. Many analyses are carried out in ATLAS, with several hundred ongoing at any one time, which cover a broad range of topics and focus on a wide range of signatures. As a result, analysis tools and workflows are necessarily diverse, flexible, and focused on analysis-specific needs. This section covers commonly used tools and methods and highlights examples of how some analysis-specific needs are accommodated.

Analysis workflows in ATLAS often entail the following steps, described in more detail below:

1. Processing from central derivations, which may include applying corrections to analysis objects (calibrations), steps towards data reduction, and systematic variations.
2. Analysis design and development, which may include the creation of additional intermediate data formats and processing steps.
3. Statistical analysis to quantify the correspondence between experimental observations and theoretical predictions.
4. Creation of data products (e.g. figures, tables, likelihood functions) for publications and other publicly available material, analysis preservation, and reinterpretation.

Carrying out an analysis typically relies on a combination of centrally provided code, community supported tools, and analysis-specific software. In addition, several computing and storage resources are available for processing of analysis data, ranging from the Grid to local machines and personal computers.

8.1.1 Analysis data formats and workflows

The starting point for an analysis workflow is the central derivation formats for analysis, described in Sect. 4.5. The model for Run 3 is characterised by the introduction of a new common central derivation format for analysis, `DAOD_PHYS`, whose goal is to support most analyses in Run 3 [352]. This format contains information for all physics objects (muons, electrons, photons, hadronically-decaying τ-leptons, jets including identification and flavour tagging, and missing transverse momentum), which makes it possible to do calibrations and study systematic variations. This information allows significant flexibility in object definitions, such that analysts can optimise the choice of objects for their specific analysis needs. The format also has content related to tracking and vertexing, the trigger, and information from the event generator record for simulated samples.

As was already done during Run 2, additional derivation formats are produced for a few specific needs:

Combined Performance (CP) studies: Specialized formats are used to complete detailed studies of object performance and to design new reconstruction and identification algorithms. These formats are developed by the CP groups whose task is to deliver the recommendations for the different analysis objects. These specialized derivation formats typically contain detailed information relevant to the object(s) in question and are produced for a few key samples. The recommendations are then propagated to the central recommendations and included in `DAOD_PHYS` or other formats for analysis.

Analyses using non-standard objects or methods: Analyses with specialized processing needs, such as searches for long-lived particles, often use dedicated derivation formats that include the relevant information required for the analysis. These derivation formats typically are subjected to heavy skimming, reducing the fraction of events to the level of a few percent to reduce the storage needs. As these methods advance, the goal is to integrate these skimmed events into the central format to minimize both the number of formats and the number of analyses relying on special formats (see Sect. 4.5.1). Having this two-stage setup allows for flexibility in pursuing new developments for analysis.

MC event generator information: Derivation formats containing more detailed information about the MC event generation process, called `TRUTH` formats, are used for specialized tasks including the validation of event generator configurations and MC simulation samples, detailed classification based on object origin, and the calculation of the acceptance for a selection. Standard analysis formats like `DAOD_PHYS` include significant truth information as well, including links between reconstructed objects and the truth particles to which they most closely correspond, as well as classification of the truth particles in terms of their origin (e.g. lepton from a hadron decay or from a Z boson; truth particle jet containing a B-hadron or not).

All derivation formats are typically further skimmed and slimmed by analysers using a set of analysis tools that comprise an analysis framework. In the case of formats for analysis workflows, objects are generally calibrated, after which common object selections are applied using a set of tools provided by the CP groups, referred to as 'CP Tools'. Derivation formats also retain additional in-file metadata (see also Sect. 6.3), describing, for example, the number of events before skimming is applied, to allow the correct normalisation of an MC simulation sample and to check for dataset completeness.

Another new development for Run 3 is the introduction of a smaller derivation format, `DAOD_PHYSLITE` (see Sect. 4.5.1), which contains calibrated physics objects, obtained after applying the CP Tools. It is intended to support most analyses in the future (i.e. for Run 4 and beyond). This format is already being deployed for development and early

adoption in Run 3, and was used for published data analyses [353]. The DAOD_PHYSLITE format is intended to be produced from DAOD_PHYS and can be centrally produced with frequent updates, typically every few weeks or months as needed.

8.1.2 Object calibrations and systematic uncertainties

ATLAS provides software releases that are dedicated for analysis and include all the relevant tools, including the tools from CP groups. These are the AnalysisBase releases and AthAnalysis. Conceptually, AthAnalysis was designed to provide an algorithm scheduling framework as close to that of Athena as possible, while reducing the amount of code that must be distributed. AnalysisBase was designed to be a light project external to Athena, containing only the bare bones necessary to run an analysis (e.g. the CP Tools and Algorithms themselves). Significant effort was invested to ensure that Tools are *dual-use*, meaning that they can be run either within Athena and AthAnalysis or within AnalysisBase releases; this avoids risks of code duplication between the two projects. The AnalysisBase software stack is somewhat simpler (e.g. it does not rely on LCG releases and instead explicitly tracks all its external dependencies), but several components must therefore be re-implemented within it (e.g. Algorithms, Tools, and messaging base-classes are all re-written in AnalysisBase compared with those in AthAnalysis). One of the key advantages of all these software releases is the wide availability of the software, which is available on all Grid computing sites and user machines via CVMFS [354]. The inputs required for CP Tools, such as conditions information for object calibration, are also stored on CVMFS. Software images of AthAnalysis and AnalysisBase releases are additionally available to download to local machines on a GitLab registry [26].

Analysis frameworks are used to run on derivations and can read xAOD objects and apply CP tools that are used to calibrate, correct simulation to match the expected performance in data, and select the physics objects used for most physics analyses. These tools also provide estimates of the systematic uncertainties in the performance of the objects (such as the efficiency and resolution). There are a few centrally available frameworks, including some that are ATLAS-specific. Most often, analyses running using AnalysisBase rely on the EventLoop framework, a framework widely used in ATLAS that processes single events at a time, helps with parallel processing of events by providing a job configuration that is sent out to worker nodes either in a batch or Grid system, and handles the merging of outputs after processing. Analyses running in AthAnalysis most often directly use the Athena framework and corresponding base-classes. Several physics groups have developed specific frameworks, often based on EventLoop. Analysis frameworks are typically based on ROOT [47], which is used for I/O and provides many other capabilities.

To improve the sharing of code and harmonize user analyses, common 'CP Algorithms' are increasingly being used in analysis frameworks in ATLAS. These CP Algorithms provide a wrapper around the CP Tools that configures and schedules them in such a way that analysts do not need to write additional code to use them, but rather can incorporate them into the rest of the analysis code. The common CP Algorithms are used to produce the DAOD_PHYSLITE format and are also a common layer in several analysis frameworks.

In addition, several analysis frameworks are designed to account for the computational infrastructure available and aid parallel processing, for example relying on TSelector in the PROOF environment [355]. There is also an increasing use of PYTHON for analysis, particularly for the user interface such as with PYROOT [356] and DASK [357] for parallel processing. There have also been several recent developments in functional and distributed approaches to data analysis in ROOT [358] and an increasing interest in using tools and methods from data science (see for example Refs. [359,360]).

8.1.3 Event selection and data reduction

Dedicated, common analysis tools are used to select the data for analysis, account for trigger selections and modelling, model the pile-up and so on.

Data quality checks: Soon after data-taking, each luminosity block (see Sect. 2.1.1) of data is certified for physics analysis. The quality of the data in each luminosity block is encoded with data quality flags that are set in the data quality monitoring and stored in a database. Depending on the specific needs of each analysis, the data in luminosity blocks with undesired or bad quality need to be filtered out. A good runs list (GRL) is an XML file that lists the ranges of good luminosity blocks in each run. A tool is provided for analyses to check if events are contained in a luminosity block within a desired range. Data quality processes at ATLAS are detailed in Ref. [43] and in Sect. 5.1.

Event cleaning: A few events are also not recommended for analysis due to specific detector, readout, or software issues, as described in Ref. [43]. These problematic events, which may be corrupted or incomplete, are flagged in the analysis data formats for removal. In addition, common tools were implemented for the removal of non-collision backgrounds (e.g. beam-halo passing through the detector) [361].

Luminosity: Another ingredient needed for an analysis is the integrated luminosity. A tool is available to calculate the luminosity corresponding to a specific analysis depending on two inputs: the luminosity blocks processed from the GRL, and the trigger(s) to be applied in the analysis. The trigger

is then used to calculate the fraction of the luminosity that ATLAS recorded, which is called the trigger *live-time*, and the *prescale* of the trigger used to select events. Prescaled triggers are those that do not accept all events (e.g. a trigger with a prescale of 50 only accepts one in 50 events) and the value of the prescale often changes with changing luminosity conditions. Prescales can only change along luminosity block boundaries, and the prescale of each trigger in each luminosity block is available in a database. Most analyses rely on triggers without a prescale applied (unprescaled triggers), but the luminosity tool checks and calculates the luminosity by adding all the selected luminosity blocks and scaling the luminosity each according to the prescale of the trigger of interest [235].

Pile-up modelling: Simulated samples are produced with an estimate for the pile-up in the corresponding data. It is difficult to predict the exact conditions of the LHC, including how much luminosity is delivered for each pile-up value; some trigger selections induce biases in the pile-up distribution (i.e. there is not a single pile-up distribution for all analyses); and systematic uncertainties in the luminosity calibration and other sources lead to uncertain modelling of pile-up that require the evaluation of systematic uncertainties in the distribution. Therefore, a reweighting procedure is used to match the distribution of pile-up in MC simulation to that of data. A tool is used to derive event weights that correct differences between MC simulation and data for the distribution of instantaneous luminosity and trigger prescale conditions. The modelling of pile-up affects several key variables such as the reconstruction efficiency and isolation of objects.

Trigger modelling: The data for analysis is collected with a suite of triggers. The trigger information in a sample is packed to save space; therefore, tools are provided to identify events that satisfied trigger selections based on the human-readable trigger names, and to describe the conditions under which the event satisfied the trigger selection. To match the MC simulation and the data, each analysis requires that both satisfy the requirements of the trigger logic for one or multiple triggers. While the MC simulation tries to describe the expected behaviour of the triggers, there might be differences between the logic implemented in the MC simulation and the one used for the actual data-taking. Sometimes, a trigger used online might not be available in the MC simulation, and a proxy must be used. Corrections to the efficiency for the trigger selection are derived as event weights that may depend on several inputs such as the kinematics of the reconstructed objects (e.g. where in the detector the object falls; see for example Ref. [362]). A tool is provided to determine the appropriate scaling factors that an analyst should apply to the MC simulation to reproduce the efficiency in data. The choice of trigger is typically associated with several event selection requirements for an analysis. Analysers might wish to associate a reconstructed object to one that resulted in the trigger decision (often called *trigger matching*, see for example Ref. [362]), and this information is also available.

Overlap removal: Several algorithms are used to identify objects of a particular type, and a single object in the detector can be identified as multiple types of objects (e.g., hadronic taus or electrons may also be identified as jets). To carry out an analysis, a choice must be made between the types of objects to interpret the content of the event. This choice is not universal: some analyses may wish to favour leptons over jets; others may wish to favour jets over leptons, for example. The removal of the duplication of an object is called *overlap removal*. Several schemes for overlap removal are supported centrally in ATLAS (see for example Ref. [363]) and the implementation for analysis is available in a tool.

8.1.4 Simulation modelling and uncertainties

Most analyses rely on MC simulation, which is often used to verify the understanding of the detector performance, to model the process of interest (signal) and several background processes, to validate data-driven methods, and to quantify systematic uncertainties by changing the settings used in the simulations or using alternative methods. As described previously, calibration and uncertainty recommendations are implemented in common tools in AnalysisBase and AthAnalysis software releases that are used by almost all analyses to improve the modelling of objects and event properties (see for example Ref. [218]). The use of MC simulation for analysis also heavily relies on metadata (see also Sect. 6.3), which includes information such as a unique integer MC *dataset identifier* that can be used to identify a sample (i.e. a specific event generator configuration) and the number of events processed in a file, which is needed to normalise the samples to the expected cross section and luminosity. Theoretical uncertainties from the knowledge of the parton distribution functions, scale variations, alternative generators, parton showers, tunes, etc., are available either through alternative samples or stored as event weights in the samples. Insofar as is possible, recommendations are made for specific sample combinations, most precise available cross-section calculations for normalisation, and weights to apply for the best possible estimate of a Standard Model process and its uncertainties. However, each event generator suffers from some (often unique) set of modelling inaccuracies, and some analyses are uniquely sensitive to the modelling of particular observables; therefore, sometimes alternative prescriptions must be identified.

In addition, many analyses rely on multivariate classifiers, such as boosted decision trees or neural networks, to complete tasks such as discriminating between signal and back-

ground. These classifiers or methods typically rely on MC simulation for the training and performance validation (see also Sect. 3.6).

8.1.5 End-stage analysis and statistical interpretation

Several tools are used for end-stage analysis to do additional data reduction steps, derive selections for analysis regions; compute derived variables, including multivariate discriminants; produce histograms; compare data and expected backgrounds; and prepare inputs for statistical tools. In addition to the many capabilities available in ROOT [46], there are several projects within SCIKIT-HEP [364] that provide similar functionality: file-handling tools like UPROOT(+COFFEA) [365], histogramming tools like BOOST- HISTOGRAM [366] and HIST [367], data manipulation tools like AWKWARD ARRAY [368], and tools for statistical analysis like IMINUIT [369].

Several tools are available to implement the most common statistical tests used at the LHC experiments. Experimental results are formulated in a statistical language, so a measurement is a parameter estimate, a discovery is a hypothesis test, and for a physics model parameterised by theoretical parameters (such as hypothetical masses and couplings for new particles), excluded and allowed regions are defined as confidence intervals. For most analyses, once the goal and statistical model are defined, the statistical procedures are encoded in the RooStats project [370], which is based on the RooFit modelling language [371]. Several of the statistical analysis frameworks that are widely used in ATLAS are based on HistFactory, a tool to build parameterised probability density functions in the RooFit/RooStats framework from ROOT histograms [372], such as HistFitter [373] and TRexFitter [374]. These frameworks provide a user-friendly interface and help with common tasks, such as profiling of nuisance parameters used to encode uncertainties, carry out validation checks, and provide an interpretation of the results. A more modern implementation of HistFactory in Python using tensors and automatic differentiation is available in pyHF [375].

8.1.6 Preparation of results, preservation and reinterpretations

The final key steps in an analysis are to prepare the results for publication and sharing with the broader scientific community. In addition to providing the figures and tables in public pages, many ATLAS results are available in HEPData [376], which is an open-source, publicly available repository for high-energy physics results. In addition to providing the results in figures and tables in digitised format, some ATLAS analyses have recently also provided the likelihood function in HEPdata [377]. Several other tools have recently become available to help with the reuse and preservation of analyses. Examples that have widespread use in ATLAS, particularly for searches, include Simple Analysis [378], RECAST [379], and Reana [380]. Another example, extensively used for measurements, is the Rivet framework [144], which is used by theorists and experimentalists for a range of studies including understanding and improving the modelling of event generators.

8.1.7 Analysis infrastructure

The specific resources used for analysis depend on the size of the derivation used, the specific workflow, and sometimes the analysis needs for example for a final event selection. Some analyses that are very selective can run on local computing clusters or even laptops. However, the most common analysis workflows involve running on derivations on the Grid or sometimes on a local computing cluster. The final selections, analysis optimisations, and the statistical interpretation are often done on a local cluster or sometimes on a personal computer. The final stages of analyses, such as preparing final figures, are often done on local resources such as a personal computer or interactive linux system like lxplus available at CERN. It is relatively straightforward to run the ATLAS software using CVMFS. There are several options for data access from several sites (see Sect. 7.2.3), typically with standard ROOT I/O [46], with or without XCache [313], or the Rucio redirector [280], both over the LAN and the WAN.

More recently, there is a renewed interest in developing dedicated infrastructure for analysis (often referred to as *analysis facilities*), particularly when considering future needs at the HL-LHC. There is also an interest in exploring the potential of technologies such as the cloud and partnerships with industry, for example with Google and Amazon, that enable rapid and large scaling-up and specialised resources for analysis. Already, existing national analysis facilities are being expanded with additional resources and capabilities (e.g. Jupyter [381] support and GPU resources) to support modern data analysis techniques and tools. Local computing clusters at individual institutes already represent significant computing resources; according to a recent estimate, together they are comparable to or larger than the largest Grid sites.[25] The availability and harmonisation of these resources is important not just for the optimisation of computing resources towards the HL-LHC, but also for the equity of the collaboration and its institutes.

8.2 Event displays

Interactive data visualisation is a key component in HEP experiments, where it is used at each step of the data pipeline

[25] These resources are not included among those discussed in Sect. 7.

(simulation, reconstruction, analysis, and so forth) to inspect and explore different types of data interactively: event data (such as tracks, hits, or energy deposits), detector geometry (such as passive and sensitive volumes), magnetic field (e.g. direction and magnitude) or conditions data. These visualization tools are often used offline (for example in physics analysis), and can also be used online for the detector operation. They can be utilised to prepare high-quality images to show the experimental signature of particular physics processes (used for example for outreach purposes) or to inspect the detector's response under given experimental conditions, such as beam splashes [382]. These images are collectively called *Event Displays*, and all public ATLAS event displays can be found at the ATLAS Event Display Public Results webpage [383].

The ATLAS Collaboration has developed several tools to visualise events [384], each of them addressing different needs and targeting different use cases and end users. For the Run 3 data-taking period, three visualisation tools were updated and are actively developed and maintained: VP1 (Virtual Point 1) [385], ATLANTIS [386,387] and PHOENIX-ATLAS [388,389], as well as the JIVEXML data exporter tool and the Online Event Displays machinery, all of which are briefly described below. A further tool was developed for Run 3, the GeoModel Explorer or GMEX, which is derived from VP1 and specialises in the visualisation of the detector geometry, as described in Sect. 3.5.

8.2.1 Software design principles

When developing visualisation tools, there are two main paradigms that applications can follow: full integration of the visualisation tools into the main experiment's software framework or standalone operation outside of the framework. The tools that follow the former paradigm can access all experimental data in a native way, directly from the experiment's software framework, but are limited by the technology and platform boundaries imposed by the framework itself. The applications following the standalone paradigm are free to choose any technology or tool that suit best their need, but have to design intermediate data exchange formats and develop and maintain tools to export data from the experiment's software framework.

8.2.2 VP1

VP1 [385] is the visualisation tool integrated into the ATLAS experiment's framework, Athena. As such, it can directly access experiment data, without the need for intermediate data formats, and can re-use the same tools as are used in reconstruction and simulation workflows. VP1 provides C++ based interactive 3D graphics, and can be extended with custom plugins to visualise ATLAS data. As it also reads the detector description information from Athena directly, it shows precisely the same detector geometry that is used in the simulation and reconstruction, and is able to show any geometry configuration of the detector. Furthermore, as VP1 is a module that can be run together with all other Athena applications, it can make use of all other `AlgTools` in Athena (see Sect. 4), and can also be used to visualise transient data, such as inner detector tracks or jets, while they are reconstructed by the dedicated algorithms.

VP1 is developed on top of the COIN [390] 3D graphics C++ framework and uses the QT [391] framework as the graphical user interface (GUI) layer, with the SOQT [392] library as the glue package between the 3D and GUI layers. One of the main tasks in preparation for Run 3 was the integration of VP1 in the ATLAS Control Room and the *Online Event Displays* machinery, as described in Sect. 8.2.6. Figure 39 shows the VP1 user interface.

8.2.3 ATLANTIS

ATLANTIS [386,387] is a stand-alone JAVA event display for the ATLAS experiment. It provides a collection of specialised, data-oriented projections in two or three dimensions to visualise physics processes and monitor the performance of all ATLAS sub-detectors. These projections and modifications to the projections (e.g. fish-eye or regional zoom) allow users to explore the data in a way that a strictly physical representation of the detector might not allow. For example, a fish-eye view allows hits in both the tracker and the calorimeter to be simultaneously visible. Figure 40 shows the user interface of the ATLANTIS event display. ATLANTIS runs independent of Athena, making it easier to install on different platforms. It uses a simplified geometry of the ATLAS detector in XML [393] format, and the event data are also read from XML files. Both the geometry XML and event data XML are produced by a dedicated algorithm, JIVEXML, which is detailed in Sect. 8.2.5. Customized versions of ATLANTIS are used in the MINERVA [394] and HYPATIA [395] projects as educational tools in master-classes for high school students.

8.2.4 PHOENIXATLAS

PHOENIXATLAS is the ATLAS web-based event display intended to easily visualise ATLAS events using a web browser. It is a TYPESCRIPT [396] application and uses the PHOENIX [397] event display library to read and process the events, with a dedicated JIVEXML converter providing access to the ATLAS XML data format. An Athena Algorithm to dump ATLAS event data to PHOENIX's native JSON data format is also available. PHOENIXATLAS can show all the main reconstructed analysis objects and provides an intuitive interface to add cuts, slice away geometry and change the colours and various properties of what is displayed. Addi-

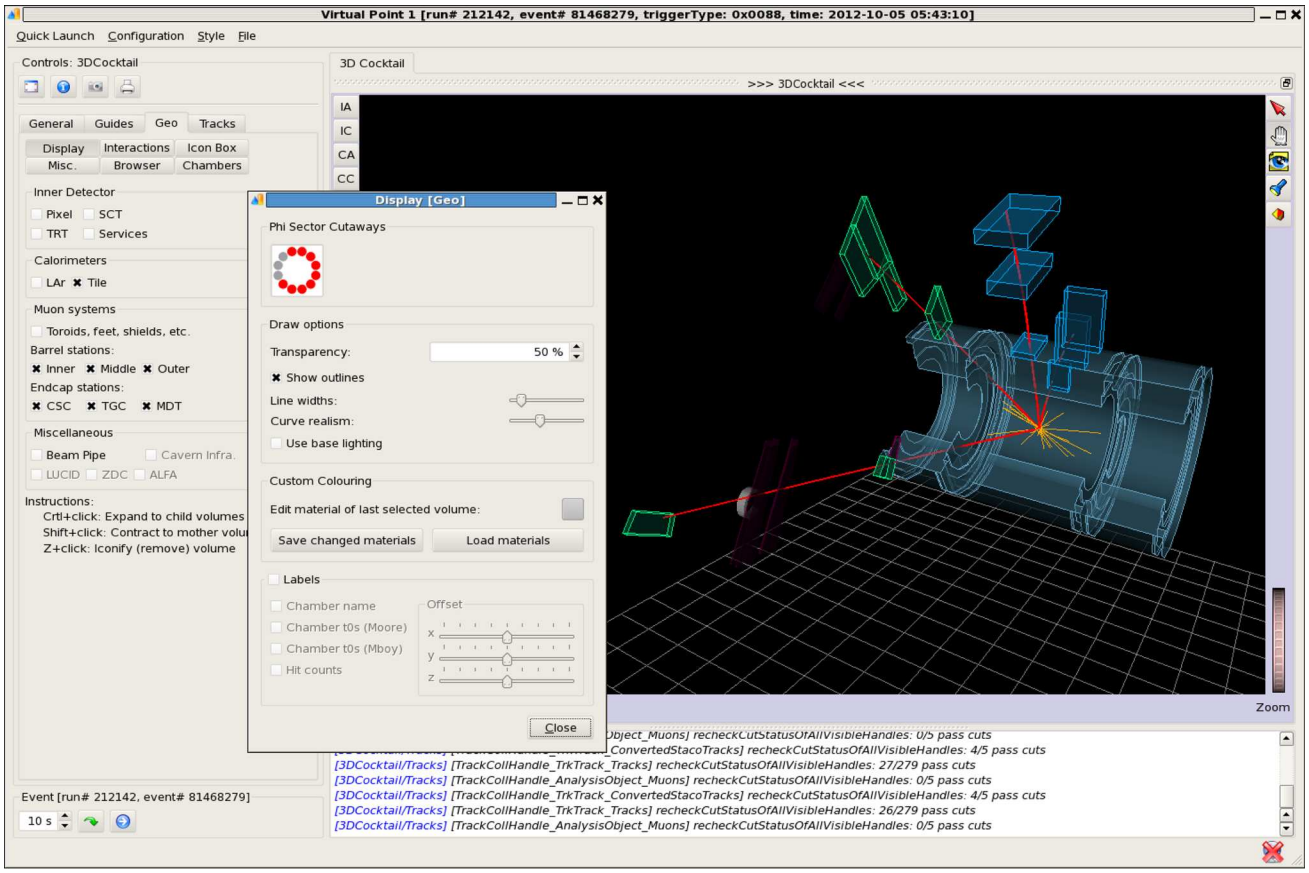


Fig. 39 The VP1 event display. The image shows the main window and one of the many widgets that let users change settings, set cuts, and customize the visualisation. In this example, the main window shows some of the physics objects from a candidate Higgs boson production event in real data: the tracks reconstructed in the inner detector (orange lines), as well as the four muons (red) and their associated muon chambers (blue and green boxes). The widget shows the settings to customize the visualisation of the detector geometry

tionally, there is support for virtual reality and augmented reality on appropriate display technology. Figure 41 shows a screenshot of PHOENIXATLAS.

Preparation for Run 3 has involved several tasks: adding the ability to show different geometries for different runs, validating PHOENIXATLAS output by ensuring it matches that of VP1 and ATLANTIS, improved cut (requirements like minimum p_{T} or η restrictions) functionality and support for more analysis object types. Another recent and significant project was preparing PHOENIXATLAS to show live events and adding links to these on the ATLAS live page [398].

8.2.5 JIVEXML

JIVEXML is a C++ event converter interface between the ATLAS event displays and the ATLAS reconstruction data. It consists of a series of retriever algorithms, running in Athena and converting fully reconstructed events to XML format. The event data XML files can be viewed in both ATLANTIS and PHOENIXATLAS. JIVEXML can extract the detector geometry to produce a geometry XML file that is then used by ATLANTIS to display a simplified ATLAS detector. JIVEXML can also act as an XML-RPC server to send files directly to ATLANTIS.

8.2.6 Online event displays

Event displays are used during ATLAS data taking to provide visual feedback to the operations team on the detector's performance. Figure 42 shows a chart of the online event display workflow. The core of the online event displays is hosted in an *Event Displays partition* [399], which controls a set of applications as a group that can function independently during data taking. Several Athena event-processor applications run the online reconstruction on a subset of events recorded by ATLAS in real time, producing XML and ESD files and writing them to disk. A JIVEXML server application runs in the partition to serve the reconstructed events in XML format to several instances of ATLANTIS, including two instances of ATLANTIS running as image producer

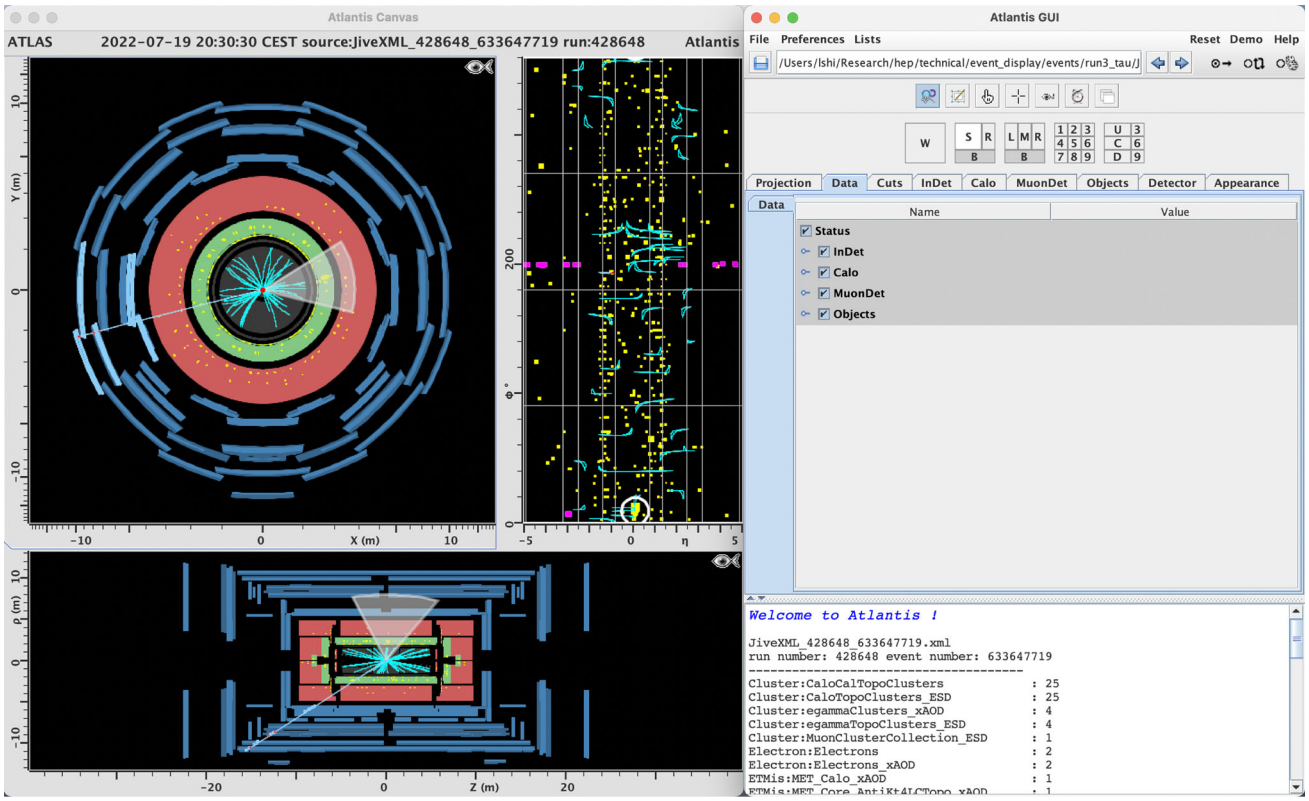


Fig. 40 The ATLANTIS event display. The main ATLANTIS canvas on the left allows multiple projections to be displayed simultaneously. In this example, a $Z \rightarrow \tau\tau$ candidate event in real data is displayed in the y–x, ϕ–η and ρ–z projections, with one τ-lepton decaying leptonically into a muon (the light blue track) and the other τ-lepton decaying hadronically, shown as a jet (the white cone). The ATLANTIS GUI on the right provides a tabbed panel of menus for display manipulation

applications in the partition that render a subset of events to share with the CERN Control Centre and the general public, and several instances of ATLANTIS running outside the partition in the control room: the one running on the Data Quality shifter's desk allowing the shift crew to assess the data online, and one projected on the control room's wall for quick feedback. These event displays have proven useful for quickly identifying issues like dysfunctional regions of the detector, even when low-level monitoring does not indicate a serious problem. The ESD files on disk can be read by instances of VP1 to produce interactive 3D event displays on the shifters' desks and the control room's wall.

The XML and ESD files on disk are also transferred to EOS, where different event displays can access the event data and render them on offline servers. The ATLAS Live web page [398] displays the images rendered by ATLANTIS with the corresponding XML files for downloading, as well as providing links to the same events on the PHOENIXATLAS web page, to allow a wider range of interested physicists to study these events.

8.3 Tutorials and education

8.3.1 Historical software tutorials

The ATLAS Collaboration has offered specialized training for various aspects of the available software and workflows since 2004. Beginning in 2008, ATLAS began holding regular centralized tutorial events that comprehensively cover the necessary tools for offline physics analysis. The target audience for the tutorials is early PhD students and other, more senior individuals who are new to the collaboration or who would benefit from learning the details of software analysis. Tutorial events are held three to four times per year, on average. Initially, the event consisted of 2.5 days of material, and it has since expanded to have four to five days of material. Since its inception, the curriculum evolved to be increasingly pedagogical with a focus on practical applications for physics analysis. These general tutorials, along with several subject-specific tutorials, are publicly available online [400].

The educational material was designed to showcase the latest available software using examples of usage in data analysis. Until 2020, tutorial events were held in-person at

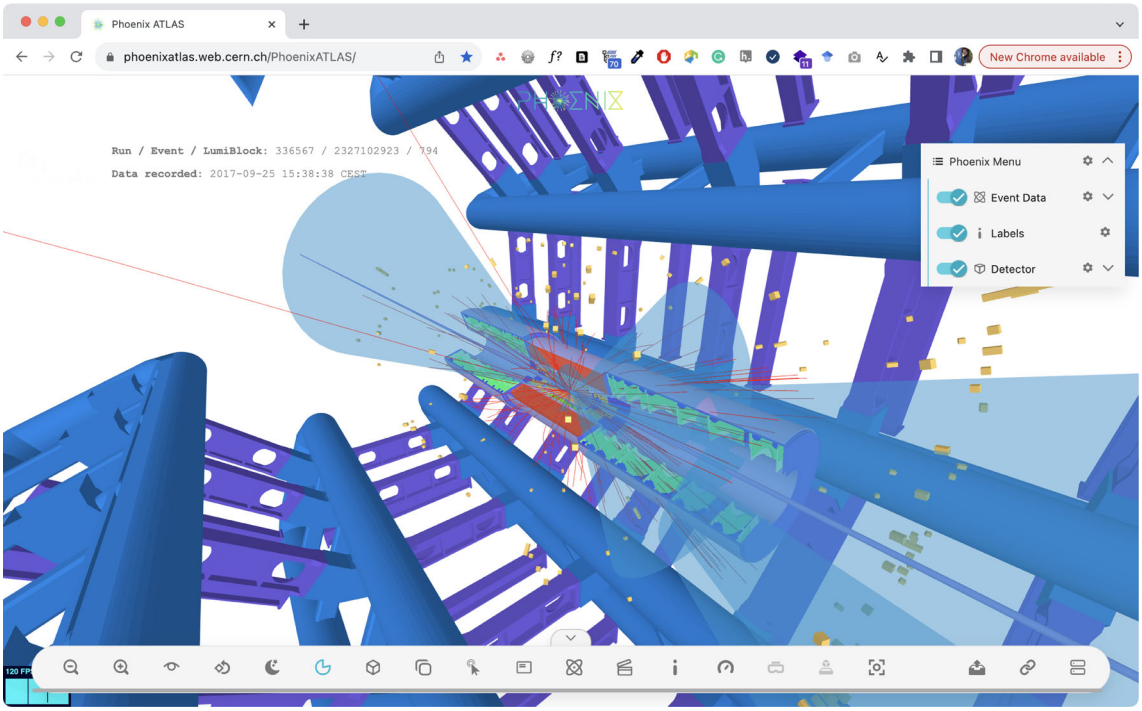


Fig. 41 The PHOENIXATLAS event display. The image shows the main user interface, with the buttons and the drop-down menus that let users change the settings and customize the visualisation. In the image, several physics objects (such as jets, tracks, and hits) are shown on top of a view of the barrel toroid magnet (blue/purple bars)

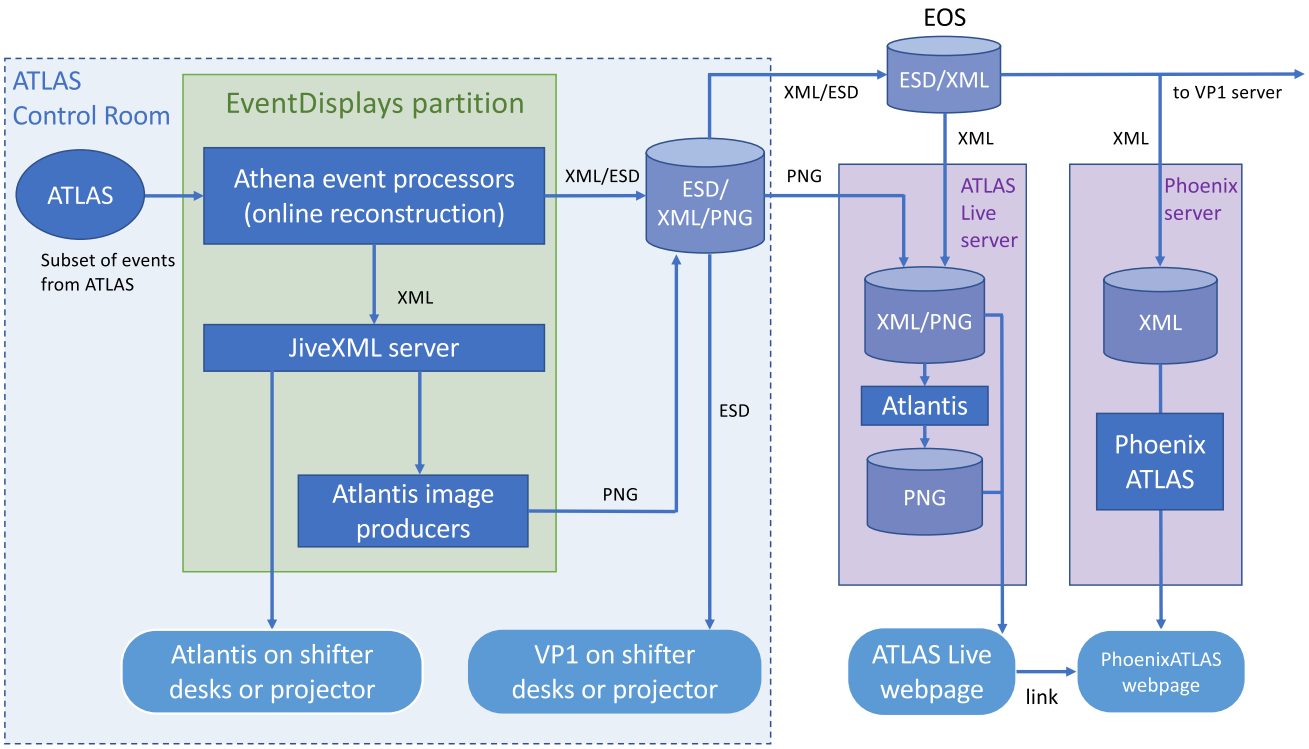


Fig. 42 The online event display workflow

CERN without a remote connection, consisting of lectures and hands-on sessions providing examples of the tools and methods introduced in the lectures. Starting in 2017, the software tutorial was coupled to Induction Day, a day-long event that introduces new members of ATLAS to critical aspects of the collaboration, its organization, and work in experimental high-energy physics. The Induction Day is held in person at CERN with a video connection for remote participants. Following the Induction Day sessions, experts are made available to assist participants in setting up their CERN and ATLAS computing accounts. The process of setting up the necessary accounts is complex and often requires the intervention of expert advice, beyond what is available from the tutorial organizers and tutors. This centralized help session makes the account setup process relatively simple for the participants. In a typical event, the Induction Day and account setup session are held on a Monday and the software training session is held during the remainder of the week. Holding the events in a single week makes it easier for participants who travel to CERN.

8.3.2 Tutorials during COVID-19

Due to the COVID-19 global pandemic and subsequent safety measures, the ATLAS software training transitioned to an asynchronous, remote format. The in-person lectures were replaced by pre-recorded lectures that participants could watch at their convenience. The recorded lectures were optimised for a remote audience with the lecturer speaking to the camera with slides. Closed captioning was professionally provided for the videos to improve accessibility. An approximately hour-long live question-and-answer session was scheduled each day during which participants could interact with experts for a discussion of any questions that arose while watching the recorded lectures. The question-and-answer session was scheduled to maximize accessibility for registered participants around the world. Throughout each day, tutors were available on a Discord server to provide in-time assistance for the hands-on exercises. It was found that Discord provides an optimal experience allowing group discussions as well as text- and voice-based breakout rooms for one-on-one help. This format was highly successful in making the training resources (material, exercises, and access to experts) available to participants around the world.

In 2022 as COVID-19 safety measures were loosened, in-person events at CERN were once again allowed. In June 2022, Induction Day and software tutorials were held in a hybrid format to provide the benefits of in-person attendance at CERN and remote asynchronous accessibility for participants not at CERN. For the software training, in-person participants were provided with live lectures (typically in-person, with some being remote) and access to in-person tutors. Remote participants could either remotely listen to the live lectures or watch the recorded lectures from the online version of the tutorial and could interact with tutors on the Discord server. It was found that those attending in-person had a high level of engagement, while those attending remotely, either synchronously or asynchronously, had a significantly lower level of engagement. This is possibly due to subconscious bias of the organizers, lecturers and tutors towards focusing primarily on in-person participants. It is also noteworthy that many individuals participated remotely. This experience with a hybrid format was crucial for making future tutorials as widely accessible as possible. In 2023, the collaboration transitioned to offering separate in-person and remote options, offset by a few weeks, to try to improve the tutorial experience for both the groups.

8.3.3 Run 3 training format

Since Fall 2023, the software training was revamped to pedagogically demonstrate the use of ATLAS software in the major steps of a physics analysis workflow. The format closely follows the full Run 2 same-flavour dilepton scalar leptoquark (LQ) analysis [401], from MC simulation production to statistical analysis and setting limits. To allow participants to experience the numerous steps in the analysis in a condensed period of four days, the procedure is significantly simplified and time-consuming computing steps are bypassed such that complete output files are provided to students after each step. Additionally, analysis optimisation procedures, background estimation methods, and systematic uncertainty evaluation steps are minimized to provide examples of each without being as rigorous and time consuming as they would be for a published analysis. The tutorial is designed assuming familiarity of participants with C++ and PYTHON.

The tutorial begins with MC simulation (see also Sect. 4.1). Participants generate LQ MC simulation samples using leading-order MADGRAPH [117] with PYTHIA [119] showering with inclusive decays. Generator filters are implemented to focus on the same-flavour dilepton final state and analysis code is provided to validate the simulated kinematics and decay modes. To minimize computation time, participants produce only a few hundred events. Next, participants are given information about the procedure to request centrally produced MC simulation samples and how to search for and access available samples.

The next step in the tutorial focuses on producing simple ROOT ntuples from derivations (see Sect. 4.5.1). Signal samples with $\mathcal{O}(10{,}000)$ events are provided in a DAOD format. First, the infrastructure to read DAODs and write information to ntuples is presented, with the option of using either `EventLoop` or Athena/AthAnalysis. Next, CP Algorithms are introduced with the example of using the good runs list (see Sect. 5.1) and pile-up reweighting. CP algo-

rithms are then used to access, calibrate, and store physics objects (e.g. electrons, jets, and missing transverse energy) for event reconstruction and analysis. Finally, CP algorithms are used to access trigger decision information and to select events satisfying a set of single lepton triggers. Participants are then introduced to using batch systems and the Grid to process the ntuple production jobs.

At this point, participants are provided with a complete set of ntuples for analysis. The ntuples contain significantly more variables than those produced in the previous part of the tutorial and consist of numerous signal MC simulation samples, a complete set of background MC simulation samples, and detector data. The first exercise using the ntuples is to plot dilepton masses in data to see the Z-boson and J/Ψ peaks, which is used to constrain the normalisation of the estimated background from Z-boson production with jets. This is done using NumPy, uproot, awkward array, coffea, and MatPlotLib [402,403] through a Jupyter [381] notebook interface. Next, the same interface is used to compare kinematic distributions between signal and background samples. A signal significance figure of merit is used to find an optimal threshold for a single kinematic variable. Following this cut-based optimisation, an alternative approach is also presented, using Boosted Decision Trees (BDTs) to discriminate signal from background. The BDT material is also presented through a Jupyter notebook interface. While the work done is not sufficient for a published analysis, it is designed to give participants some experience with common analysis steps and available software. Although the tutorial focuses heavily on Python-based tools (targetting mostly, but not exclusively, younger colleagues), ROOT is still widely used in practice for analysis in the collaboration.

Following the analysis design and optimisation section, systematic uncertainties are introduced. Participants return to the ntuple production step and make use of the CP Algorithms to evaluate and save systematic variations of the physics objects. From a technical perspective, this section would be better suited before the analysis optimisation step. However, in a typical analysis development, systematic uncertainties are neglected or greatly simplified until the analysis strategy is mature and the optimisations are well-advanced. Therefore, this section ordering is chosen to reflect actual analysis workflows.

Finally, participants complete a statistical analysis and set limits on the LQ signal. TRexFitter [374] is used for the fit, first using Asimov data and then with detector data. The fit is done with the reconstructed LQ mass and the BDT discriminant. Some background normalisations are allowed to float in the fit and a few systematic uncertainties are included. Finally, expected and observed upper limits on the cross section times branching ratio are set for a range of signal mass points.

A feedback survey is provided to the participants before the final session in the tutorial event. The participants are requested to submit their responses before the end of the session. This results in a much higher response rate than circulating a survey to the participants after the event has ended. The results of the surveys are taken into account when modifying the material and format for future events.

8.3.4 Retention rates and offsite events

Most of the tutorial events are held at CERN for reasons of cost and simple logistics. However, due to the nature of the daily routine at CERN, these sessions often have a low retention rate. Participants generally have various meetings and other obligations while on site, and therefore do not attend all sessions. The new format in which one step follows from the next to form a complete analysis was found to strongly encourage participants to attend all sessions, resulting in an improved retention rate. Throughout the years, several training events were held at other sites, often with financial assistance for participants. This isolation from the daily life at CERN results in nearly perfect retention.

8.3.5 Other available instructional resources

In addition to the analysis software tutorial events, ATLAS offers a variety of other training materials and events. Self-guided tutorials are provided for GitLab, the Grid, Athena development, specific tools such as Visual Studio Code and Docker, and various specific aspects of ATLAS software such as tracking and flavour-tagging tools. This material is provided via internal Twiki pages, AtlasSoftwareDocs [400], and dedicated websites such as Ref. [404]. The AtlasSoftwareDocs pages also host numerous other resources, including training guides and instruction manuals for code review and building releases. Furthermore, numerous mailing lists are available that allow individuals to quickly contact experts for technical support. In addition to these resources for asynchronous training, ATLAS also organizes a variety of other tutorial events, in-person at CERN, at offsite locations, or remotely. These cover topics ranging from Athena development to machine learning and introductions to newly adopted collaboration-wide tools such as GitLab.

Several other training initiatives not specific to ATLAS but useful for high-energy physicists were developed worldwide [405]. An attempt is made to avoid duplication when good learning modules are already available for specific tools (e.g. C++ and Python).

9 Outlook highlights

The ATLAS experiment is preparing for a major upgrade during the next long shutdown, LS3, when the accelerator complex at CERN will also be upgraded. The outcome of these upgrades together are known as the high-luminosity LHC (HL-LHC). The HL-LHC will deliver 3–4 times more proton–proton collisions to ATLAS per second, reaching $\langle\mu\rangle = 140$ in Run 4 and $\langle\mu\rangle = 200$ in Run 5. The detector will have several parts completely replaced and some new components added to deal with this higher rate, and the readout system will be upgraded to record about 10,000 events per second, with an event size about three times larger than that of today, resulting in overall data processing requirements that are about 10 times greater than today [15].

To prepare for the HL-LHC, ATLAS Software and Computing have developed a Conceptual Design Report [406] and a Roadmap [407] detailing the challenges towards the HL-LHC and the milestones and deliverables required to meet those challenges. Major development is expected throughout the software and computing systems, improving concurrency, adopting modern approaches to certain simulation and reconstruction challenges, integrating new tools for event generators, machine learning, and data analysis, and re-working some parts of the processing and data handling systems to cope with an order of magnitude more data (see also Sect. 4.7).

These upgrade research and development projects include the incorporation of accelerators (GPUs, FPGAs, and other types) into several different parts of the software. The online trigger system is investigating the use of hardware accelerators for a variety of purposes, and offline developments are significantly advanced in event generation, simulation, reconstruction, and data analysis. The Athena infrastructure itself already supports the use of accelerators, and limited applications including fast calorimeter simulation [408] and topological clustering of energy in the calorimeter [409] were already validated, with significant development around charged particle tracking ongoing [410,411]. The outstanding question is whether the savings in time and electrical power will be sufficient to merit the investment to deploy hardware accelerators on worldwide Grid sites. A firm decision about the use of accelerators both online and offline is expected around 2025–2026.

One key question is what will happen to the existing data from Runs 1, 2, and 3 during the HL-LHC era. The ATLAS Collaboration has committed to saving all the RAW data from the experiment and to releasing some of the data[26] to the public after a period of embargo: 25% of the data will be released five years after each Run, 50% will be released 10 years after each run, and all the data will be released by the end of the lifetime of the collaboration. These public releases of data will be in the `DAOD_PHYSLITE` format described in Sect. 4.5, and will be accompanied by a set of MC simulation datasets sufficient to complete real data analysis. Other, smaller datasets will be regularly released for specific purposes. Several were already released on the CERN Open Data Portal [412] for use in educational settings and for exploration of machine learning techniques.

An effort is now underway to understand how the collaboration should treat the Runs 1, 2, and 3 data during the HL-LHC internally, from relying on the open datasets and published data artifacts as the only connection to earlier Runs, to retaining the ability to fully reconstruct the older data in modern releases. The latter path presents significant challenges: not only must the software be able to reproduce older geometries and configurations, but conditions data must be brought forward into new database infrastructure, calibrations must be provided for old runs, the issue of how to run a static piece of trigger software (that which was run during the corresponding data-taking period) on top of new MC simulation must be addressed, and so on. The development is a major undertaking, almost equivalent to supporting two experiments simultaneously.

10 Summary

The ATLAS experiment is supported by a complex software and computing system that provides extensive functionality, flexibility, and performance in support of about 100 published data analyses every year. Many of these systems were re-developed in recent years, following 25 years of experience in the experiment and a variety of advances in modern computing, including multithreading, database systems, and open-source software solutions. This paper describes the modern production software at the beginning of Run 3 of the LHC.

The computing infrastructure of the experiment is broad, flexible, and highly distributed, with a hub at CERN where the real detector data are first processed and examined. The simulated and real data proceed through a series of well-defined transformation steps, where the simulation closely mirrors the behaviour of the detector, and all processing is built on the same core infrastructure. The core software of the experiment has evolved considerably over the years, now supporting multithreading and sporting an entirely new, more standard and more maintainable configuration system.

[26] The Run 1 data will not be released in this way without significant additional effort and attention. These data have not been reprocessed in almost 10 years, despite a significant effort at the beginning of Run 2. The challenge of ensuring that all of the Run 2 software was able to support the different geometry, conditions, and configuration of the Run 1 detector proved too onerous a task.

Robust support for machine learning techniques was built into the system, and the data model allows extreme flexibility in outputs.

The simulated and detector data processing steps integrate gold-standard tools for event generation and detailed detector simulation. At the same time, many bespoke steps like fast simulation, digitization, and reconstruction have developed complexity to deal with new detectors and conditions, to provide greater veracity with limited computing resources, and to make the best possible use of new software technologies like machine learning. The first common analysis step, derivation making, in which analysis teams have direct influence over the processing of the data is rapidly evolving in preparation for the HL-LHC. All these processing steps are integrated into a unified workflow system, with the flexibility to optionally include forward detector systems, to change the detector layout to account for upgrades, or to reconfigure the reconstruction for different operational conditions.

Significant validation is done in preparation of new productions of MC simulation or reprocessings of detector data, with monitoring and staged productions ensuring minimal waste in the case an issue is identified. Extensive infrastructure has also been built-up to support developers and users, from flexible builds, to nightly builds and testing, to scripts supporting consistent environments that ensure work is relocatable. Databases and metadata systems ensure that configuration, condition, and result histories are preserved and can be used both for monitoring and to automate some parts of subsequent software configurations and productions.

The simulated and real detector data, as well as the jobs processing the data, are spread over many distributed computing sites in a system that is sufficiently flexible to integrate many complex new resources. Automation and industry-standard tools were integrated to assist in and reduce operation efforts. Downstream, a suite of software was developed for user analysis that ensures robustness, useability, and flexibility to maximize the physics output of the experiment. New users are continuously introduced to the system and trained on the use of modern data analysis tools in processing the complex detector data.

Development within software and computing is continuing in preparation for the HL-LHC, and a 10-year plan was laid out in some detail. These plans provide for further modernization of some systems, explore the incorporation of new tools like more advanced machine learning techniques and hardware accelerators, and reinforce the infrastructure that will be ready to deal with an order-of-magnitude increase in data volume. Even as a mature experiment, there is constant, healthy pressure for improvements and optimisations, pushing ATLAS to meet the coming challenges.

The current schedule foresees the ATLAS experiment and the HL-LHC running through at least 2041. By that time, it is likely that the software and computing landscape will have changed significantly. With sustainable, robust, performant, and adaptable infrastructure, the experiment will be able to continue to deliver high-quality physics analyses for the decades to come.

Acknowledgements We thank CERN for the very successful operation of the LHC and its injectors, as well as the support staff at CERN and at our institutions worldwide without whom ATLAS could not be operated efficiently. The crucial computing support from all WLCG partners is acknowledged gratefully, in particular from CERN, the ATLAS Tier-1 facilities at TRIUMF/SFU (Canada), NDGF (Denmark, Norway, Sweden), CC-IN2P3 (France), KIT/GridKA (Germany), INFN-CNAF (Italy), NL-T1 (Netherlands), PIC (Spain), RAL (UK) and BNL (USA), the Tier-2 facilities worldwide and large non-WLCG resource providers. Major contributors of computing resources are listed in Ref. [413]. We gratefully acknowledge the support of ANPCyT, Argentina; YerPhI, Armenia; ARC, Australia; BMWFW and FWF, Austria; ANAS, Azerbaijan; CNPq and FAPESP, Brazil; NSERC, NRC and CFI, Canada; CERN; ANID, Chile; CAS, MOST and NSFC, China; Minciencias, Colombia; MEYS CR, Czech Republic; DNRF and DNSRC, Denmark; IN2P3-CNRS and CEA-DRF/IRFU, France; SRNSFG, Georgia; BMBF, HGF and MPG, Germany; GSRI, Greece; RGC and Hong Kong SAR, China; ISF and Benoziyo Center, Israel; INFN, Italy; MEXT and JSPS, Japan; CNRST, Morocco; NWO, Netherlands; RCN, Norway; MNiSW, Poland; FCT, Portugal; MNE/IFA, Romania; MESTD, Serbia; MSSR, Slovakia; ARIS and MVZI, Slovenia; DSI/NRF, South Africa; MICIU/AEI, Spain; SRC and Wallenberg Foundation, Sweden; SERI, SNSF and Cantons of Bern and Geneva, Switzerland; NSTC, Taipei; TENMAK, Türkiye; STFC/UKRI, United Kingdom; DOE and NSF, United States of America. Individual groups and members have received support from BCKDF, CANARIE, CRC and DRAC, Canada; PRIMUS 21/SCI/017, CERN-CZ and FORTE, Czech Republic; COST, ERC, ERDF, Horizon 2020, ICSC-NextGenerationEU and Marie Skłodowska-Curie Actions, European Union; Investissements d'Avenir Labex, Investissements d'Avenir Idex and ANR, France; DFG and AvH Foundation, Germany; Herakleitos, Thales and Aristeia programmes co-financed by EU-ESF and the Greek NSRF, Greece; BSF-NSF and MINERVA, Israel; Norwegian Financial Mechanism 2014–2021, Norway; NCN and NAWA, Poland; La Caixa Banking Foundation, CERCA Programme Generalitat de Catalunya and PROMETEO and GenT Programmes Generalitat Valenciana, Spain; Göran Gustafssons Stiftelse, Sweden; The Royal Society and Leverhulme Trust, United Kingdom. In addition, individual members wish to acknowledge support from CERN: European Organization for Nuclear Research (CERN PJAS); Chile: Agencia Nacional de Investigación y Desarrollo (FONDECYT 1190886, FONDECYT 1210400, FONDECYT 1230812, FONDECYT 1230987); China: Chinese Ministry of Science and Technology (MOST-2023YFA1605700, MOST-2023YFA1609300), National Natural Science Foundation of China (NSFC-12175119, NSFC 12275265, NSFC-12075060); Czech Republic: Czech Science Foundation (GACR-24-11373S), Ministry of Education Youth and Sports (FORTE CZ.02.01. 01/00/22_008/0004632), PRIMUS Research Programme (PRIMUS/21/SCI/017); EU: H2020 European Research Council (ERC-101002463); European Union: European Research Council (ERC-948254, ERC 101089007), Horizon 2020 Framework Programme (MUCCA-CHIST-ERA-19-XAI-00), European Union, Future Artificial Intelligence Research (FAIR-NextGenerationEU PE00000013), Italian Center for High Performance Computing, Big Data and Quantum Computing (ICSC, NextGenerationEU); France: Agence Nationale de la Recherche (ANR-20-CE31-0013, ANR-21-CE31-0013, ANR-21-CE31-0022, ANR-22-EDIR-0002), Investissements d'Avenir Labex (ANR-11-LABX-0012); Germany: Baden-Württemberg Stiftung (BW Stiftung-Postdoc Eliteprogramme), Deutsche Forschungsgemeinschaft (DFG-469666862, DFG-CR 312/5-2); Italy: Istituto Nazionale di Fisica Nucleare (ICSC, NextGenera-

tionEU), Ministero dell'Università e della Ricerca (PRIN-20223N7F8K-PNRR M4.C2.1.1); Japan: Japan Society for the Promotion of Science (JSPS KAKENHI JP21H05085, JSPS KAKENHI JP22H01227, JSPS KAKENHI JP22H04944, JSPS KAKENHI JP22KK0227); Netherlands: Netherlands Organisation for Scientific Research (NWO Veni 2020-VI.Veni.202.179); Norway: Research Council of Norway (RCN-314472); Poland: Polish National Agency for Academic Exchange (PPN/PPO/2020/1/00002/U/00001), Polish National Science Centre (NCN 2021/42/E/ST2/00350, NCN OPUS nr 2022/47/B/ST2/03059, NCN UMO-2019/34/E/ST2/00393, NCN and H2020 MSCA 945339, UMO-2020/37/B/ST2/01043, UMO-2021/40/C/ST2/00187, UMO-2022/47/O/ST2/00148); Slovenia: Slovenian Research Agency (ARIS grant J1-3010); Spain: Generalitat Valenciana (Artemisa, FEDER, IDIFEDER/2018/048), Ministry of Science and Innovation (MCIN and NextGenEU PCI2022-135018-2, MICIN and FEDER PID2021-125273NB, RYC2019-028510-I, RYC2020-030254-I, RYC2021-031273-I, RYC2022-038164-I), PROMETEO and GenT Programmes Generalitat Valenciana (CIDEGENT/2019/023, CIDEGENT/2019/027); Sweden: Swedish Research Council (Swedish Research Council 2023-04654, VR 2018-00482, VR 2022-03845, VR 2022-04683, VR 2023-03403, VR grant 2021-03651), Knut and Alice Wallenberg Foundation (KAW 2018.0157, KAW 2018.0458, KAW 2019.0447, KAW 2022.0358); Switzerland: Swiss National Science Foundation (SNSF-PCEFP2_194658); United Kingdom: Leverhulme Trust (Leverhulme Trust RPG-2020-004), Royal Society (NIF-R1-231091); United States of America: U.S. Department of Energy (ECA DE-AC02-76SF00515), Neubauer Family Foundation.

Data Availability Statement This manuscript has no associated data. [Authors' comment: Data sharing not applicable to this article as no datasets were generated or analysed during the current study.]

Code Availability Statement This manuscript has no associated code/software. [Authors' comment: All code discussed in this paper is available under the Apache 2.0 license on gitlab.cern.ch.]

Funded by SCOAP[3].

References

1. ATLAS Collaboration, The ATLAS experiment at the CERN large hadron collider. JINST **3**, S08003 (2008). https://doi.org/10.1088/1748-0221/3/08/S08003
2. ATLAS Collaboration, The ATLAS experiment at the CERN Large Hadron Collider: a description of the detector configuration for Run 3. JINST **19**(3), P05063 (2023). https://doi.org/10.1088/1748-0221/19/05/P05063. arXiv:2305.16623 [physics.ins-det]
3. ATLAS Collaboration, Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. Phys. Lett. B **716**, 1 (2012). https://doi.org/10.1016/j.physletb.2012.08.020. arXiv:1207.7214 [hep-ex]
4. CMS Collaboration, Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. Phys. Lett. B **716**, 30 (2012). https://doi.org/10.1016/j.physletb.2012.08.021. arXiv:1207.7235 [hep-ex]
5. I. Bird et al., LHC Computing Grid: Technical Design Report, CERN-LHCC-2005-024 (2005). https://cds.cern.ch/record/840543
6. I. Bird et al., Update of the Computing Models of the WLCG and the LHC Experiments, CERN-LHCC-2014-014 (2014). https://cds.cern.ch/record/1695401
7. ATLAS Collaboration, ATLAS New Small Wheel: Technical Design Report, ATLAS-TDR-020; CERN-LHCC-2013-006 (2013). https://cds.cern.ch/record/1552862
8. ATLAS Collaboration, ATLAS Liquid Argon Calorimeter Phase-I Upgrade: Technical Design Report, ATLAS-TDR-022; CERN-LHCC-2013-017 (2013). https://cds.cern.ch/record/1602230
9. ATLAS Collaboration, ATLAS TDAQ System Phase-I Upgrade: Technical Design Report, ATLAS-TDR-023; CERN-LHCC-2013-018 (2013). https://cds.cern.ch/record/1602235
10. ATLAS Collaboration, ATLAS Forward Proton Phase-I Upgrade: Technical Design Report, ATLAS-TDR-024; CERN-LHCC-2015-009 (2015). https://cds.cern.ch/record/2017378
11. ATLAS Collaboration, ATLAS Inner Tracker Strip Detector: Technical Design Report, ATLAS-TDR-025; CERN-LHCC-2017-005 (2017). https://cds.cern.ch/record/2257755
12. ATLAS Collaboration, ATLAS Muon Spectrometer Phase-II Upgrade: Technical Design Report, ATLAS-TDR-026; CERN-LHCC-2017-017 (2017). https://cds.cern.ch/record/2285580
13. ATLAS Collaboration, ATLAS LAr Calorimeter Phase-II Upgrade: Technical Design Report, ATLAS-TDR-027; CERN-LHCC-2017-018 (2017). https://cds.cern.ch/record/2285582
14. ATLAS Collaboration, ATLAS Tile Calorimeter Phase-II Upgrade: Technical Design Report, ATLAS-TDR-028; CERN-LHCC-2017-019 (2017). https://cds.cern.ch/record/2285583
15. ATLAS Collaboration, ATLAS TDAQ Phase-II Upgrade: Technical Design Report, ATLAS-TDR-029; CERN-LHCC-2017-020 (2017). https://cds.cern.ch/record/2285584
16. ATLAS Collaboration, ATLAS Inner Tracker Pixel Detector: Technical Design Report, ATLAS-TDR-030; CERN-LHCC-2017-021 (2017). https://cds.cern.ch/record/2285585
17. ATLAS Collaboration, A High-Granularity Timing Detector for the ATLAS Phase-II Upgrade: Technical Design Report, ATLAS-TDR-031; CERN-LHCC-2020-007 (2020). https://cds.cern.ch/record/2719855
18. ATLAS Collaboration, The ATLAS Trigger System for LHC Run 3 and Trigger performance in 2022 (2024). https://doi.org/10.48550/arxiv.2401.06630. arXiv:2401.06630 [hep-ex]
19. Worldwide LHC Computing Grid Collaboration, Sample MoU, CERN-C-RRB-2005-01 (2015). https://wlcg.web.cern.ch/organisation-mou/sample-mou
20. D.P. Anderson, BOINC: a platform for volunteer computing. J. Grid Comput. **18**, 99 (2019). https://doi.org/10.1007/s10723-019-09497-9. arXiv:1903.01699 [cs]
21. D.P. Anderson, BOINC: a system for public-resource computing and storage. In *Grid '04: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing 4* (2004). https://doi.org/10.1109/GRID.2004.14
22. D.S. Myers, A.L. Bazinet, M.P. Cummings, Expanding the reach of Grid computing: combining Globus- and BOINC-based systems. Grids for Bioinformatics and Computational Biology, Wiley Book Series on Bioinformatics: Computational Techniques and Engineering, Chapter 4, 71 (2008). https://doi.org/10.1002/9780470191637.ch4

23. A. Anisenkov, J. Andreeva, A. Di Girolamo, P. Paparrigopoulos, B. Vasilev, CRIC: Computing Resource Information Catalogue as a unified topology system for a large scale, heterogeneous and dynamic computing infrastructure. EPJ Web Conf. **245**, 03032 (2020). https://doi.org/10.1051/epjconf/202024503032
24. ATLAS Collaboration, The ATLAS Collaboration Software and Firmware, ATL-SOFT-PUB-2021-001 (2021). https://cds.cern.ch/record/2767187
25. Athena. https://doi.org/10.5281/zenodo.2641997
26. Athena gitlab repository. https://gitlab.cern.ch/atlas/athena
27. Apache Licence, Version 2.0, Apache Software Foundation (2004). https://www.apache.org/licenses/LICENSE-2.0
28. ATLAS Collaboration, Electron and photon efficiencies in LHC Run 2 with the ATLAS experiment. JHEP **05**, 162 (2024). https://doi.org/10.1007/JHEP05(2024)162. arXiv:2308.13362 [hep-ex]
29. ATLAS Collaboration, Studies of the muon momentum calibration and performance of the ATLAS detector with pp collisions at $\sqrt{s} = 13$ TeV. Eur. Phys. J. C **83**, 686 (2023). https://doi.org/10.1140/epjc/s10052-023-11584-x. arXiv:2212.07338 [hep-ex]
30. ATLAS Collaboration, Measurement of the tau lepton reconstruction and identification performance in the ATLAS experiment using pp collisions at $\sqrt{s} = 13$ TeV, ATLAS-CONF-2017-029 (2017). https://cds.cern.ch/record/2261772
31. ATLAS Collaboration, Jet energy scale and resolution measured in proton–proton collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector. Eur. Phys. J. C **81**, 689 (2021). https://doi.org/10.1140/epjc/s10052-021-09402-3. arXiv:2007.02645 [hep-ex]
32. ATLAS Collaboration, ATLAS b-jet identification performance and efficiency measurement with $t\bar{t}$ events in pp collisions at $\sqrt{s} = 13$ TeV. Eur. Phys. J. C **79**, 970 (2019). https://doi.org/10.1140/epjc/s10052-019-7450-8. arXiv:1907.05120 [hep-ex]
33. ATLAS Collaboration, ATLAS Insertable B-Layer: Technical Design Report, ATLAS-TDR-19; CERN-LHCC-2010-013 (2010). https://cds.cern.ch/record/1291633 [Addendum: ATLAS-TDR-19-ADD-1; CERN-LHCC-2012-009 (2012). https://cds.cern.ch/record/1451888]
34. B. Abbott et al., Production and integration of the ATLAS Insertable B-Layer. JINST **13**, T05008 (2018). https://doi.org/10.1088/1748-0221/13/05/T05008. arXiv:1803.00844 [physics.ins-det]
35. ATLAS Collaboration, E/p measurements and Geant4 physics list comparisons, JETM-2020-03. https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PLOTS/JETM-2020-03/
36. ATLAS Collaboration, Performance of the ATLAS trigger system in 2015. Eur. Phys. J. C **77**, 317 (2017). https://doi.org/10.1140/epjc/s10052-017-4852-3. arXiv:1611.09661 [hep-ex]
37. M. Dobbs, J.B. Hansen, The HepMC C++ Monte Carlo event record for High Energy Physics. Comput. Phys. Commun. **134**, 41 (2001). https://doi.org/10.1016/S0010-4655(00)00189-2
38. S. Agostinelli et al., Geant4—a simulation toolkit. Nucl. Instrum. Meth. A **506**, 250 (2003). https://doi.org/10.1016/S0168-9002(03)01368-8
39. J. Allison et al., Geant4 developments and applications. IEEE Trans. Nucl. Sci. **53**, 270 (2006). https://doi.org/10.1109/TNS.2006.869826
40. J. Allison et al., Recent developments in Geant4. Nucl. Instrum. Meth. A **835**, 186 (2016). https://doi.org/10.1016/j.nima.2016.06.125
41. ATLAS Collaboration, The ATLAS simulation infrastructure. Eur. Phys. J. C **70**, 823 (2010). https://doi.org/10.1140/epjc/s10052-010-1429-9. arXiv:1005.4568 [physics.ins-det]
42. ATLAS Collaboration, AtlFast3: the next generation of fast simulation in ATLAS. Comput. Softw. Big Sci. **6**, 7 (2022). https://doi.org/10.1007/s41781-021-00079-7. arXiv:2109.02551 [hep-ex]
43. ATLAS Collaboration, ATLAS data quality operations and performance for 2015–2018 data-taking. JINST **15**, P04003 (2020). https://doi.org/10.1088/1748-0221/15/04/P04003. arXiv:1911.04632 [physics.ins-det]
44. CMake. https://cmake.org/download/
45. A. Danial, CLOC: Count Lines of Code (2015). https://cloc.sourceforge.net
46. R. Brun, F. Rademakers, ROOT—an object oriented data analysis framework. Nucl. Instrum. Meth. A 389, 81 (1997). https://doi.org/10.1016/S0168-9002(97)00048-X. http://root.cern.ch
47. I. Antcheva et al., ROOT—a C++ framework for petabyte data storage, statistical analysis and visualization. Comput. Phys. Commun. **180**, 2499 (2009). https://doi.org/10.1016/j.cpc.2009.08.005. arXiv:1508.07749 [physics.data-an]
48. G. Barrand et al., GAUDI—a software architecture and framework for building HEP data processing applications (2001). https://gitlab.cern.ch/gaudi/Gaudi
49. P. Calafiura et al., Running ATLAS workloads within massively parallel distributed applications using Athena Multi-Process framework (AthenaMP). J. Phys. Conf. Ser. **664**, 072050 (2015). https://doi.org/10.1088/1742-6596/664/7/072050
50. ATLAS Collaboration, Performance of Multi-threaded Reconstruction in ATLAS, ATL-SOFT-PUB-2021-002 (2021). https://cds.cern.ch/record/2771777
51. I. Shapoval et al., Graph-based decision making for task scheduling in concurrent Gaudi. In *Proceedings of the 2015 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC 2015): San Diego, California* (2016). https://doi.org/10.1109/NSSMIC.2015.7581843
52. I. Shapoval, Adaptive Scheduling Applied to Non-Deterministic Networks of Heterogeneous Tasks for Peak Throughput in Concurrent Gaudi, PhD thesis: Kharkov, KIPT (2016). http://inspirehep.net/record/1503877
53. Intel Threading Building Blocks. https://github.com/oneapi-src/oneTBB
54. ATLAS Collaboration, Event data access in AthenaMT. https://twiki.cern.ch/twiki/bin/view/AtlasComputing/MultiThreadingEventDataAccess
55. C. Leggett, I. Shapoval, S. Snyder, V. Tsulaia, Conditions datahandling in the multithreaded ATLAS framework. EPJ Web Conf. **214**, 05031 (2019). https://doi.org/10.1051/epjconf/201921405031
56. P. van Gemmeren, D. Malon, The event data store and I/O framework for the ATLAS experiment at the Large Hadron Collider. In *2009 IEEE International Conference on Cluster Computing and Workshops 1* (2009). https://doi.org/10.1109/CLUSTR.2009.5289147
57. R. Trentadue et al., LCG persistency framework (CORAL, COOL, POOL): status and outlook in 2012. J. Phys. Conf. Ser **396**, 052067 (2012). https://doi.org/10.1088/1742-6596/396/5/052067
58. J. Cranshaw, D. Malon, M. Nowak, P.V. Gemmeren, I/O in the ATLAS multithreaded framework. EPJ Web Conf. **214**, 05017 (2019). https://doi.org/10.1051/epjconf/201921405017
59. S. Martin-Haugh, Implementation of the Atlas trigger within the multithreaded Athenamt framework. EPJ Web Conf. **214**, 01046 (2019). https://doi.org/10.1051/epjconf/201921401046
60. S. Snyder, Concurrent data structures in the ATLAS offline software. EPJ Web Conf. **245**, 05007 (2020). https://doi.org/10.1051/epjconf/202024505007
61. S. Kama, C. Leggett, S. Snyder, V. Tsulaia, The ATLAS multithreaded offline framework. EPJ Web Conf. **214**, 05018 (2019). https://doi.org/10.1051/epjconf/201921405018
62. The GNU Compiler Collection. https://gcc.gnu.org
63. A. Krasznahorkay et al., Multithreading ATLAS offline software: a retrospective. EPJ Web Conf. **295**, 03024 (2024). https://doi.org/10.1051/epjconf/202429503024
64. TCMalloc. https://github.com/google/tcmalloc
65. Valgrind. https://valgrind.org

66. W. Lampl, A new approach for ATLAS Athena job configuration. EPJ Web Conf. **214**, 05015 (2019). https://doi.org/10.1051/epjconf/201921405015
67. M. Verducci, ATLAS conditions database experience with the LCG COOL conditions database project. J. Phys. Conf. Ser. **119**, 042031 (2008). https://doi.org/10.1088/1742-6596/119/4/042031
68. A. Buckley et al., Implementation of the ATLAS Run 2 event data model. J. Phys. Conf. Ser. **664**, 072045 (2015). https://doi.org/10.1088/1742-6596/664/7/072045
69. J. Boudreau, V. Tsulaia, The GeoModel Toolkit for Detector Description, Computing in High Energy Physics and Nuclear Physics 2004 (2005). https://doi.org/10.5170/CERN-2005-002.353. https://cds.cern.ch/record/865601
70. S. Consortium, SQLite (2023). https://www.sqlite.org/index.html
71. GeoModel—A Detector Description Toolkit for HEP experiments. https://geomodel.web.cern.ch/home/
72. ATLAS Collaboration, Search for s-channel single top-quark production in proton–proton collisions at $\sqrt{s} = 8$ TeV with the ATLAS detector. Phys. Lett. B **740**, 118 (2015).https://doi.org/10.1016/j.physletb.2014.11.042. arXiv:1410.0647 [hep-ex]
73. ATLAS Collaboration, Measurement of the production cross-section of a single top quark in association with a W boson at 8 TeV with the ATLAS experiment. JHEP **01**, 064 (2016). https://doi.org/10.1007/JHEP01(2016)064. arXiv:1510.03752 [hep-ex]
74. ATLAS Collaboration, Measurement of the cross-section for producing a W boson in association with a single top quark in pp collisions at $\sqrt{s} = 13$ TeV with ATLAS. JHEP **01**, 063 (2018). https://doi.org/10.1007/JHEP01(2018)063. arXiv:1612.07231 [hep-ex]
75. ATLAS Collaboration, Measurement of differential cross-sections of a single top quark produced in association with a W boson at $\sqrt{s} = 13$ TeV with ATLAS. Eur. Phys. J. C **78**, 186 (2018). https://doi.org/10.1140/epjc/s10052-018-5649-8. arXiv:1712.01602 [hep-ex]
76. ATLAS Collaboration, Search for pair production of heavy vector-like quarks decaying into high-p_{T} W bosons and top quarks in the lepton-plus-jets final state in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector. JHEP **08**, 048 (2018). https://doi.org/10.1007/JHEP08(2018)048. arXiv:1806.01762 [hep-ex]
77. ATLAS Collaboration, Measurement of the top quark mass in the $t\bar{t} \to$ lepton+jets channel from $\sqrt{s} = 8$ TeV ATLAS data and combination with previous results. Eur. Phys. J. C **79**, 290 (2019). https://doi.org/10.1140/epjc/s10052-019-6757-9. arXiv:1810.01772 [hep-ex]
78. ATLAS Collaboration, Searches for scalar leptoquarks and differential cross-section measurements in dilepton–dijet events in proton–proton collisions at a centre-of-mass energy of $\sqrt{s} = 13$ TeV with the ATLAS experiment. Eur. Phys. J. C **79**, 733 (2019). https://doi.org/10.1140/epjc/s10052-019-7181-x. arXiv:1902.00377 [hep-ex]
79. ATLAS Collaboration, Search for long-lived neutral particles in pp collisions at $\sqrt{s} = 13$ TeV that decay into displaced hadronic jets in the ATLAS calorimeter. Eur. Phys. J. C **79**, 481 (2019). https://doi.org/10.1140/epjc/s10052-019-6962-6. arXiv:1902.03094 [hep-ex]
80. ATLAS Collaboration, Search for light long-lived neutral particles produced in pp collisions at $\sqrt{s} = 13$ TeV and decaying into collimated leptons or light hadrons with the ATLAS detector. Eur. Phys. J. C **80**, 450 (2020). https://doi.org/10.1140/epjc/s10052-020-7997-4. arXiv:1909.01246 [hep-ex]
81. ATLAS Collaboration, Evidence for $t\bar{t}t\bar{t}$ production in the multi-lepton final state in proton–proton collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector. Eur. Phys. J. C **80**, 1085 (2020). https://doi.org/10.1140/epjc/s10052-020-08509-3. arXiv:2007.14858 [hep-ex]
82. ATLAS Collaboration, Measurements of WH and ZH production in the $H \to b\bar{b}$ decay channel in pp collisions at 13 TeV with the ATLAS detector. Eur. Phys. J. C **81**, 178 (2021). https://doi.org/10.1140/epjc/s10052-020-08677-2. arXiv:2007.02873 [hep-ex]
83. ATLAS Collaboration, Measurement of the $t\bar{t}t\bar{t}$ production cross section in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector. JHEP **11**, 118 (2021). https://doi.org/10.1007/JHEP11(2021)118. arXiv:2106.11683 [hep-ex]
84. ATLAS Collaboration, Reconstruction and identification of boosted di-τ systems in a search for Higgs boson pairs using 13 TeV proton–proton collision data in ATLAS. JHEP **11**, 163 (2020). https://doi.org/10.1007/JHEP11(2020)163. arXiv:2007.14811 [hep-ex]
85. ATLAS Collaboration, Search for neutral long-lived particles in pp collisions at $\sqrt{s} = 13$ TeV that decay into displaced hadronic jets in the ATLAS calorimeter. JHEP **06**, 005 (2022). https://doi.org/10.1007/JHEP06(2022)005. arXiv:2203.01009 [hep-ex]
86. ATLAS Collaboration, Electron and photon energy calibration with the ATLAS detector using LHC Run 1 data. Eur. Phys. J. C **74**, 3071 (2014). https://doi.org/10.1140/epjc/s10052-014-3071-4. arXiv:1407.5063 [hep-ex]
87. ATLAS Collaboration, Search for single top-quark production via flavour-changing neutral currents at 8 TeV with the ATLAS detector. Eur. Phys. J. C **76**, 55 (2016). https://doi.org/10.1140/epjc/s10052-016-3876-4. arXiv:1509.00294 [hep-ex]
88. ATLAS Collaboration, Comprehensive measurements of t-channel single top-quark production cross sections at $\sqrt{s} = 7$ TeV with the ATLAS detector. Phys. Rev. D **90**, 112006 (2014). https://doi.org/10.1103/PhysRevD.90.112006. arXiv:1406.7844 [hep-ex]
89. ATLAS Collaboration, Measurement of the inclusive cross-sections of single top-quark and top-antiquark t-channel production in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector. JHEP **04**, 086 (2017). https://doi.org/10.1007/JHEP04(2017)086. arXiv:1609.03920 [hep-ex]
90. ATLAS Collaboration, Measurement of the production cross-section of a single top quark in association with a Z boson in proton–proton collisions at 13 TeV with the ATLAS detector. Phys. Lett. B **780**, 557 (2018). https://doi.org/10.1016/j.physletb.2018.03.023. arXiv:1710.03659 [hep-ex]
91. ATLAS Collaboration, Fiducial, total and differential cross-section measurements of t-channel single top-quark production in pp collisions at 8 TeV using data collected by the ATLAS detector. Eur. Phys. J. C **77**, 531 (2017). https://doi.org/10.1140/epjc/s10052-017-5061-9. arXiv:1702.02859 [hep-ex]
92. ATLAS Collaboration, Measurement of the inclusive and fiducial $t\bar{t}$ production cross-sections in the lepton+jets channel in pp collisions at $\sqrt{s} = 8$ TeV with the ATLAS detector. Eur. Phys. J. C **78**, 487 (2018). https://doi.org/10.1140/epjc/s10052-018-5904-z. arXiv:1712.06857 [hep-ex]
93. ATLAS Collaboration, Observation of the associated production of a top quark and a Z boson in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector. JHEP **07**, 124 (2020). https://doi.org/10.1007/JHEP07(2020)124. arXiv:2002.07546 [hep-ex]
94. ATLAS Collaboration, Measurement of single top-quark production in association with a W boson in the single-lepton channel at $\sqrt{s} = 8$ TeV with the ATLAS detector. Eur. Phys. J. C **81**, 720 (2021). https://doi.org/10.1140/epjc/s10052-021-09371-7. arXiv:2007.01554 [hep-ex]
95. ATLAS Collaboration, Identification and energy calibration of hadronically decaying tau leptons with the ATLAS experiment in pp collisions at $\sqrt{s} = 8$ TeV. Eur. Phys. J. C **75**, 303 (2015). https://doi.org/10.1140/epjc/s10052-015-3500-z. arXiv:1412.7086 [hep-ex]
96. ATLAS Collaboration, Performance of top-quark and W-boson tagging with ATLAS in Run 2 of the LHC. Eur. Phys. J. C **79**,

375 (2019). https://doi.org/10.1140/epjc/s10052-019-6847-8. arXiv:1808.07858 [hep-ex]
97. A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks, Studies in Computational Intelligence*, 1st edn. (Springer, Berlin, 2012)
98. ATLAS Collaboration, ATLAS flavour-tagging algorithms for the LHC Run 2 *pp* collision dataset. Eur. Phys. J. C **83**, 681 (2023). https://doi.org/10.1140/epjc/s10052-023-11699-1. arXiv:2211.16345 [physics.data-an]
99. ATLAS Collaboration, Reconstruction, Identification, and Calibration of hadronically decaying tau leptons with the ATLAS detector for the LHC Run 3 and reprocessed Run 2 data, ATL-PHYS-PUB-2022-044 (2022). https://cds.cern.ch/record/2827111
100. ATLAS Collaboration, A neural network clustering algorithm for the ATLAS silicon pixel detector. JINST **9**, P09009 (2014). https://doi.org/10.1088/1748-0221/9/09/P09009. arXiv:1406.7690 [hep-ex]
101. I.J. Goodfellow et al., Generative Adversarial Networks (2014). arXiv:1406.2661 [stat.ML]
102. L. de Oliveira, M. Paganini, B. Nachman, Learning particle physics by example: location-aware generative adversarial networks for physics synthesis. Comput. Softw. Big Sci. **1**, 4 (2017). https://doi.org/10.1007/s41781-017-0004-6
103. M. Paganini, L. de Oliveira, B. Nachman, Accelerating science with generative adversarial networks: an application to 3D particle showers in multilayer calorimeters. Phys. Rev. Lett. **120**, 042003 (2018). https://doi.org/10.1103/PhysRevLett.120.042003
104. A. Hoecker et al., TMVA—Toolkit for Multivariate Data Analysis (2009). arXiv:physics/0703039 [physics.data-an]
105. M. Abadi et al., TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, Software available from tensorflow.org (2015). https://www.tensorflow.org/
106. F. Chollet et al., Keras (2015). https://keras.io
107. A. Paszke et al., PyTorch: An Imperative Style, High-Performance Deep Learning Library (2019). arXiv:1912.01703
108. T. Chen, C. Guestrin, XGBoost: a scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16*, 785 (2016). https://doi.org/10.1145/2939672.2939785
109. G. Ke et al., Lightgbm: a highly efficient gradient boosting decision tree. Adv. Neural Inf. Process. Syst. **30**, 3146 (2017)
110. M. Feindt, U. Kerzel, The NeuroBayes neural network package, Nucl. Instrum. Meth. A **559**, 190 (2006). Proceedings of the X International Workshop on Advanced Computing and Analysis Techniques in Physics Research, ISSN: 0168-9002. https://doi.org/10.1016/j.nima.2005.11.166
111. D.H. Guest et al., lwtnn/lwtnn: version 2.13, version v2.13, Zenodo. https://doi.org/10.5281/zenodo.6467676 (2022)
112. Boost C++ Libraries. https://www.boost.org
113. Eigen. https://eigen.tuxfamily.org/index.php
114. O. R. developers, ONNX Runtime. https://onnxruntime.ai/ (2021)
115. F.H.B. Megino et al., PanDA for ATLAS distributed computing in the next decade. J. Phys. Conf. Ser. **898**, 052002 (2017). https://doi.org/10.1088/1742-6596/898/5/052002
116. S. Alioli, P. Nason, C. Oleari, E. Re, A general framework for implementing NLO calculations in shower Monte Carlo programs: the POWHEG BOX. JHEP **06**, 043 (2010). https://doi.org/10.1007/JHEP06(2010)043. arXiv:1002.2581 [hep-ph]
117. J. Alwall et al., The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations. JHEP **07**, 079 (2014). https://doi.org/10.1007/JHEP07(2014)079. arXiv:1405.0301 [hep-ph]
118. S. Höche, F. Krauss, M. Schönherr, F. Siegert, QCD matrix elements + parton showers. The NLO case. JHEP **04**, 027 (2013). https://doi.org/10.1007/JHEP04(2013)027. arXiv:1207.5030 [hep-ph]
119. T. Sjöstrand et al., An introduction to PYTHIA 8.2. Comput. Phys. Commun. **191**, 159 (2015). https://doi.org/10.1016/j.cpc.2015.01.024. arXiv:1410.3012 [hep-ph]
120. J. Bellm et al., Herwig 7.1 Release Note (2017). arXiv:1705.06919 [hep-ph]
121. P. Golonka, Z. Was, PHOTOS Monte Carlo: a precision tool for QED corrections in Z and W decays. Eur. Phys. J. C **45**, 97 (2006). https://doi.org/10.1140/epjc/s2005-02396-4. arXiv:hep-ph/0506026
122. S. Jadach, J.H. Kühn, Z. Was, TAUOLA—a library of Monte Carlo programs to simulate decays of polarized τ leptons. Comput. Phys. Commun. **64**, 275 (1991). https://doi.org/10.1016/0010-4655(91)90038-M
123. D.J. Lange, The EvtGen particle decay simulation package. Nucl. Instrum. Meth. A **462**, 152 (2001). https://doi.org/10.1016/S0168-9002(01)00089-4
124. O.C. Allkofer, K. Carstensen, W. Dau, The absolute cosmic ray muon spectrum at sea level. Phys. Lett. B **36**, 425 (1971). https://doi.org/10.1016/0370-2693(71)90741-6
125. A. Dar, Atmospheric neutrinos, astrophysical neutrons, and proton-decay experiments. Phys. Rev. Lett. **51**, 227 (1983). https://doi.org/10.1103/PhysRevLett.51.227
126. K. Werner, F.-M. Liu, T. Pierog, Parton ladder splitting and the rapidity dependence of transverse momentum spectra in deuteron-gold collisions at the BNL Relativistic Heavy Ion Collider. Phys. Rev. C **74**, 044902 (2006). https://doi.org/10.1103/PhysRevC.74.044902. arXiv:hep-ph/0506232 [hep-ph]
127. X.-N. Wang, M. Gyulassy, hijing: a Monte Carlo model for multiple jet production in pp, pA, and AA collisions. Phys. Rev. D **44**, 3501 (1991). https://doi.org/10.1103/PhysRevD.44.3501
128. X.-N. Wang, M. Gyulassy, Systematic study of particle production in $p + p(\bar{p})$ collisions via the HIJING model. Phys. Rev. D **45**, 844 (1992). https://doi.org/10.1103/PhysRevD.45.844
129. S. Boselli, C.M.C. Calame, G. Montagna, O. Nicrosini, F. Piccinini, Higgs boson decay into four leptons at NLOPS electroweak accuracy. JHEP **06**, 023 (2015). https://doi.org/10.1007/JHEP06(2015)023. arXiv:1503.07394 [hep-ph]
130. S. Boselli et al., Higgs decay into four charged leptons in the presence of dimension-six operators. JHEP **01**, 096 (2018). https://doi.org/10.1007/JHEP01(2018)096. arXiv:1703.06667 [hep-ph]
131. I.P. Lokhtin, A.M. Snigirev, A model of jet quenching in ultra-relativistic heavy ion collisions and high-pt hadron spectra at RHIC. Eur. Phys. J. C **45**, 211 (2006). https://doi.org/10.1140/epjc/s2005-02426-3. arXiv:hep-ph/0506189 [hep-ph]
132. A. Bredenstein, A. Denner, S. Dittmaier, M.M. Weber, Precise predictions for the Higgs-boson decay H $\to$ WW/ZZ $\to$ 4 leptons. Phys. Rev. D **74**, 013004 (2006). https://doi.org/10.1103/PhysRevD.74.013004. arXiv:hep-ph/0604011 [hep-ph]
133. J.A. Aguilar-Saavedra, Single top quark production at LHC with anomalous Wtb couplings. Nucl. Phys. B **804**, 160 (2008). https://doi.org/10.1016/j.nuclphysb.2008.06.013. arXiv:0803.3810 [hep-ph]
134. S. Ostapchenko, QGSJET-II: towards reliable description of very high energy hadronic interactions. Nucl. Phys. B Proc. Suppl. **151**, 143 (2006). https://doi.org/10.1016/j.nuclphysbps.2005.07.026. arXiv:hep-ph/0412332 [hep-ph]
135. S.R. Klein, J. Nystrand, J. Seger, Y. Gorbunov, J. Butterworth, STARlight: a Monte Carlo simulation program for ultra-peripheral collisions of relativistic ions. Comput. Phys. Commun. **212**, 258 (2017). https://doi.org/10.1016/j.cpc.2016.10.016. arXiv:1607.03838 [hep-ph]
136. L. Harland-Lang, SuperChic 4—A Monte Carlo for Central Exclusive and Photon-Initiated Production. https://superchic.hepforge.org

137. J. Alwall et al., A standard format for Les Houches Event Files. Comput. Phys. Commun. **176**, 300 (2007). https://doi.org/10.1016/j.cpc.2006.11.010. arXiv:hep-ph/0609017 [hep-ph]
138. N. Kauer, C. O'Brien, E. Vryonidou, Interference effects for $H \to WW \to \ell\nu q\bar{q}'$ and $H \to ZZ \to \ell\bar{\ell}q\bar{q}$ searches in gluon fusion at the LHC. JHEP **10**, 074 (2015). https://doi.org/10.1007/JHEP10(2015)074. arXiv:1506.01694 [hep-ph]
139. S. Catani, L. Cieri, G. Ferrera, D. de Florian, M. Grazzini, Vector boson production at hadron colliders: a fully exclusive QCD calculation at next-to-next-to-leading order. Phys. Rev. Lett. **103**, 082001 (2009). https://doi.org/10.1103/PhysRevLett.103.082001. arXiv:0903.2120 [hep-ph]
140. C.M. Harris, P. Richardson, B.R. Webber, CHARYBDIS: a black hole event generator. JHEP **08**, 033 (2003). https://doi.org/10.1088/1126-6708/2003/08/033. arXiv:hep-ph/0307305 [hep-ph]
141. C. Degrande et al., UFO—The Universal FeynRules Output. Comput. Phys. Commun. **183**, 1201 (2012). https://doi.org/10.1016/j.cpc.2012.01.022. arXiv:1108.2040 [hep-ph]
142. B. Biedermann et al., Automation of NLO QCD and EW corrections with Sherpa and Recola. Eur. Phys. J. C **77**, 492 (2017). https://doi.org/10.1140/epjc/s10052-017-5054-8. arXiv:1704.05783 [hep-ph]
143. E. Bothmann et al., Accelerating LHC event generation with simplified pilot runs and fast PDFs (2022). arXiv:2209.00843 [hep-ph]
144. C. Bierlich et al., Robust independent validation of experiment and theory: Rivet version 3. SciPost Phys. **8**, 026 (2020). https://doi.org/10.21468/SciPostPhys.8.2.026. arXiv:1912.05451 [hep-ph]
145. ATLAS Collaboration. Monte Carlo Validation in ATLAS with PAVER (2024). https://cds.cern.ch/record/2904943
146. A. Buckley et al., The HepMC3 event record library for Monte Carlo event generators. Comput. Phys. Commun. **260**, 107310 (2021). https://doi.org/10.1016/j.cpc.2020.107310
147. J.B. Birks, Scintillations from organic crystals: specific fluorescence and relative response to different radiations. Proc. Phys. Soc. A **64**, 874 (1951). https://doi.org/10.1088/0370-1298/64/10/303
148. R. Brun et al., Geant: detector description and simulation tool, CERN-W5013 (1994). https://cds.cern.ch/record/1073159
149. J. Abdallah et al., Study of energy response and resolution of the ATLAS Tile Calorimeter to hadrons of energies from 16 to 30 GeV. Eur. Phys. J. C **81**, 549 (2021). https://doi.org/10.1140/epjc/s10052-021-09292-5. arXiv:2102.04088 [physics.ins-det]
150. ATLAS Collaboration, Impact of ATLAS quasi-stable particle simulation on quantities related to flavour tagging and tau reconstruction (2020). http://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PLOTS/FTAG-2020-002/
151. A. Rimoldi et al., Final Report of the Simulation Optimization Task Force, ATL-SOFT-PUB-2008-004 (2009). https://cds.cern.ch/record/1151298
152. M. Muskinja, J.D. Chapman, H. Gray, Geant4 performance optimization in the ATLAS experiment. EPJ Web Conf. **245**, 02036 (2020). https://doi.org/10.1051/epjconf/202024502036
153. ATLAS Collaboration, ATLAS Geant4 Performance Optimization Plots (2019). https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PLOTS/SIM-2019-001/
154. E. Woodcock, Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry. In *Proceedings of the Conference on Applications of Computing Methods to Reactor Problems, 1965* 557 (1965). https://cir.nii.ac.jp/crid/1573668925582592640
155. S. Wenzel, J. Apostolakis, G. Cosmo, A VecGeom navigator plugin for Geant4. EPJ Web Conf. **245**, 02024 (2020). https://doi.org/10.1051/epjconf/202024502024
156. C. Marcon et al., Studies of GEANT4 performance for different ATLAS detector geometries and code compilation methods. EPJ Web Conf. **251**, 03005 (2021). https://doi.org/10.1051/epjconf/202125103005
157. U. Drepper, How to write shared libraries (2011). https://www.akkadia.org/drepper/dsohowto.pdf
158. E. Barberio et al., Fast simulation of electromagnetic showers in the ATLAS calorimeter: frozen showers. J. Phys. Conf. Ser. **160**, 012082 (2009). https://doi.org/10.1088/1742-6596/160/1/012082
159. A. Dotti, Simplified Calorimeter: Shower Moments (2023). https://ep-dep-sft.web.cern.ch/sites/default/files/documents/ShowerMoments.pdf
160. A. Basalaev, Z. Marshall on behalf of the ATLAS Collaboration, The Fast Simulation Chain for ATLAS. J. Phys. Conf. Ser. **898**, 042016 (2017). https://doi.org/10.1088/1742-6596/898/4/042016
161. K. Edmonds et al., The Fast ATLAS Track Simulation (FATRAS), ATL-SOFT-PUB-2008-001 (2008). https://cds.cern.ch/record/1091969
162. ATLAS Collaboration, Performance of the Fast ATLAS Tracking Simulation (FATRAS) and the ATLAS Fast Calorimeter Simulation (FastCaloSim) with single particles, ATL-SOFT-PUB-2014-001 (2014). https://cds.cern.ch/record/1669341
163. H. Bichsel, Straggling in thin silicon detectors. Rev. Mod. Phys. **60**, 663 (1988). https://doi.org/10.1103/RevModPhys.60.663
164. ATLAS Collaboration, Modelling radiation damage to pixel sensors in the ATLAS detector. JINST **14**, P06012 (2019). https://doi.org/10.1088/1748-0221/14/06/P06012. arXiv:1905.03739 [hep-ex]
165. ATLAS Collaboration, Performance of ATLAS Pixel Detector and Track Reconstruction at the start of Run 3 in LHC Collisions at $\sqrt{s} = 900$GeV, ATL-PHYS-PUB-2022-033 (2022). https://cds.cern.ch/record/2814766
166. T.H. Kittelmann, Slepton spin determination and simulation of the transition radiation tracker at the ATLAS experiment (2007). https://cds.cern.ch/record/2224292
167. ATLAS Collaboration, Readiness of the ATLAS liquid argon calorimeter for LHC collisions. Eur. Phys. J. C **70**, 723 (2010). https://doi.org/10.1140/epjc/s10052-010-1354-y. arXiv:0912.2642 [hep-ex]
168. H. Abreu et al., Performance of the electronic readout of the ATLAS liquid argon calorimeters. JINST **5**, P09003 (2010). https://doi.org/10.1088/1748-0221/5/09/P09003
169. W. Lampl et al., Digitization of LAr calorimeter for CSC simulations, ATL-LARG-PUB-2007-011 (2007). https://cds.cern.ch/record/1057879
170. ATLAS Collaboration, Emulating the impact of additional proton–proton interactions in the ATLAS simulation by presampling sets of inelastic Monte Carlo events. Comput. Softw. Big Sci. **6**, 3 (2022). https://doi.org/10.1007/s41781-021-00062-2. arXiv:2102.09495 [hep-ex]
171. W.E. Cleland, E.G. Stern, Signal processing considerations for liquid ionization calorimeters in a high rate environment. Nucl. Instrum. Meth. A **338**, 467 (1994). https://doi.org/10.1016/0168-9002(94)91332-3
172. Y. Enari, on behalf of the ATLAS Collaboration, the phase-1 trigger readout electronics upgrade of the ATLAS liquid argon calorimeter. J. Phys. Conf. Ser. **1162**, 012041 (2019). https://doi.org/10.1088/1742-6596/1162/1/012041
173. R. Schwienhorst, The phase-1 upgrade of the ATLAS first level calorimeter trigger. JINST **11**, C01018 (2016). https://doi.org/10.1088/1748-0221/11/01/C01018
174. P. Adragna et al., Testbeam studies of production modules of the ATLAS tile calorimeter. Nucl. Instrum. Meth. A **606**, 362 (2009). https://doi.org/10.1016/j.nima.2009.04.009
175. ATLAS Collaboration, Readiness of the ATLAS tile calorimeter for LHC collisions. Eur. Phys. J. C **70**, 1193 (2010). https://doi.

org/10.1140/epjc/s10052-010-1508-y. arXiv:1007.5423v2 [hep-ex]
176. E. Fullana et al., Optimal Filtering in the ATLAS Hadronic Tile Calorimeter, ATL-TILECAL-2005-001 (2005). https://cds.cern.ch/record/816152
177. D. Rebuzzi et al., Geant4 Muon Digitization in the ATHENA Framework, ATL-SOFT-PUB-2007-001 (2007). https://cds.cern.ch/record/1010495
178. C. Posch, S.P. Ahlen, E.S. Hazen, J. Oliver, CMOS front-end for the MDT sub-detector in the ATLAS Muon Spectrometer, development and performance (2001). https://doi.org/10.5170/CERN-2001-005.199. https://cds.cern.ch/record/529410
179. C. Posch, E. Hazen, J. Oliver, MDT-ASD, CMOS front-end for ATLAS MDT; rev. version 2.1, ATL-MUON-2002-003 (2007). https://cds.cern.ch/record/684217
180. R. Veenhof, Garfield—simulation of gaseous detectors. http://garfield.web.cern.ch/garfield/
181. T. Alexopoulos et al., The VMM readout system. Nucl. Instrum. Meth. A **955**, 163306 (2020). https://doi.org/10.1016/j.nima.2019.163306
182. S.F. Biagi, Magboltz 11. http://magboltz.web.cern.ch/magboltz
183. M. Dixit, A. Rankin, Simulating the charge dispersion phenomena in Micro Pattern Gas Detectors with a resistive anode. Nucl. Instrum. Meth. A **566**, 281 (2006). https://doi.org/10.1016/j.nima.2006.06.050 (**issn: 0168-9002**)
184. V. Smakhtin et al., Thin Gap Chamber upgrade for SLHC: position resolution in a test beam. Nucl. Instrum. Meth. A **598**, 196 (2009). https://doi.org/10.1016/j.nima.2008.08.098
185. A. Abusleme et al., Performance of a full-size small-strip thin gap chamber prototype for the ATLAS new small wheel muon upgrade. Nucl. Instrum. Meth. A **817**, 85 (2016). https://doi.org/10.1016/j.nima.2016.01.087. arXiv:1509.06329 [physics.ins-det]
186. M. Hildreth et al., Upgrades for the CMS simulation. J. Phys. Conf. Ser. **898**, 042040 (2017). https://doi.org/10.1088/1742-6596/898/4/042040
187. ATLAS Collaboration, Performance of the ATLAS track reconstruction algorithms in dense environments in LHC Run 2. Eur. Phys. J. C **77**, 673 (2017). https://doi.org/10.1140/epjc/s10052-017-5225-7. arXiv:1704.07983 [hep-ex]
188. ATLAS Collaboration, Reconstruction of primary vertices at the ATLAS experiment in Run 1 proton–proton collisions at the LHC. Eur. Phys. J. C **77**, 332 (2017). https://doi.org/10.1140/epjc/s10052-017-4887-5. arXiv:1611.10235 [hep-ex]
189. ATLAS Collaboration, Topological cell clustering in the ATLAS calorimeters and its performance in LHC Run 1. Eur. Phys. J. C **77**, 490 (2017). https://doi.org/10.1140/epjc/s10052-017-5004-5. arXiv:1603.02934 [hep-ex]
190. ATLAS Collaboration, Monitoring and data quality assessment of the ATLAS liquid argon calorimeter. JINST **9**, P07024 (2014). https://doi.org/10.1088/1748-0221/9/07/P07024. arXiv:1405.3768 [hep-ex]
191. A. Barriuso Poy et al., The detector control system of the ATLAS experiment. JINST **3**, P05006 (2008). https://doi.org/10.1088/1748-0221/3/05/p05006
192. ATLAS Collaboration, Improving topological cluster reconstruction using calorimeter cell timing in ATLAS, CERN-EP-2023-207 (2023). arXiv:2310.16497 [physics.ins-det]
193. T.G. Cornelissen et al., Updates of the ATLAS Tracking Event Data Model (Release 13), ATL-SOFT-PUB-2007-003 (2007). https://cds.cern.ch/record/1038095
194. ATLAS Collaboration, Software Performance of the ATLAS Track Reconstruction for LHC Run 3. Comput. Softw. Big Sci. **8**, 9 (2024). https://doi.org/10.1007/s41781-023-00111-y. arXiv:2308.09471 [hep-ex]
195. R. Frühwirth, Application of Kalman filtering to track and vertex fitting. Nucl. Instrum. Meth. A **262**, 444 (1987). https://doi.org/10.1016/0168-9002(87)90887-4
196. ATLAS Collaboration, Performance of the reconstruction of large impact parameter tracks in the inner detector of ATLAS, ATL-PHYS-PUB-2017-014 (2017). https://cds.cern.ch/record/2275635
197. ATLAS Collaboration, Performance of the reconstruction of large impact parameter tracks in the inner detector of ATLAS. Eur. Phys. J. C **83**, 1081 (2023). https://doi.org/10.1140/epjc/s10052-023-12024-6. arXiv:2304.12867 [hep-ex]
198. ATLAS Collaboration, ID Track and Vertex Reconstruction, IDTR-2021-01 (2023). https://cds.cern.ch/record/2778932
199. ATLAS Collaboration, Electron and photon performance measurements with the ATLAS detector using the 2015–2017 LHC proton–proton collision data. JINST **14**, P12006 (2019). https://doi.org/10.1088/1748-0221/14/12/P12006. arXiv:1908.00005 [hep-ex]
200. ATLAS Collaboration, Charged-particle distributions at low transverse momentum in $\sqrt{s} = 13$ TeV pp interactions measured with the ATLAS detector at the LHC. Eur. Phys. J. C **76**, 502 (2016). https://doi.org/10.1140/epjc/s10052-016-4335-y. arXiv:1606.01133 [hep-ex]
201. ATLAS Collaboration, Alignment of the ATLAS Inner Detector in Run 2. Eur. Phys. J. C **80**, 1194 (2020). https://doi.org/10.1140/epjc/s10052-020-08700-6. arXiv:2007.07624 [hep-ex]
202. X. Ai et al., A Common Tracking Software Project. Comput. Softw. Big Sci. **6**, 8 (2022). https://doi.org/10.1007/s41781-021-00078-8
203. A. Salzburger et al., acts-project/acts: v25.0.0, Zenodo (2023). https://doi.org/10.5281/zenodo.7853173. https://github.com/acts-project/acts
204. ATLAS Collaboration, Development of ATLAS Primary Vertex Reconstruction for LHC Run 3, ATL-PHYS-PUB-2019-015 (2019). https://cds.cern.ch/record/2670380
205. ATLAS Collaboration, Electron reconstruction and identification in the ATLAS experiment using the 2015 and 2016 LHC proton–proton collision data at $\sqrt{s} = 13$ TeV. Eur. Phys. J. C **79**, 639 (2019). https://doi.org/10.1140/epjc/s10052-019-7140-6. arXiv:1902.04655 [physics.ins-det]
206. ATLAS Collaboration, Electron and photon reconstruction and performance in ATLAS using a dynamical, topological cell clustering-based approach, ATL-PHYS-PUB-2017-022 (2017). https://cds.cern.ch/record/2298955
207. W. Lampl et al., Calorimeter Clustering Algorithms: Description and Performance, ATL-LARG-PUB-2008-002 (2008). https://cds.cern.ch/record/1099735
208. ATLAS Collaboration, Muon reconstruction performance of the ATLAS detector in proton–proton collision data at $\sqrt{s} = 13$ TeV. Eur. Phys. J. C **76**, 292 (2016). https://doi.org/10.1140/epjc/s10052-016-4120-y. arXiv:1603.05598 [hep-ex]
209. W. Leight et al., New fitting concept in ATLAS muon tracking for the LHC Run-2. EPJ Web Conf. **214**, 06006 (2019). https://doi.org/10.1051/epjconf/201921406006
210. ATLAS Collaboration, Muon reconstruction and identification efficiency in ATLAS using the full Run 2 pp collision data set at $\sqrt{s} = 13$ TeV. Eur. Phys. J. C **81**, 578 (2021). https://doi.org/10.1140/epjc/s10052-021-09233-2. arXiv:2012.00578 [hep-ex]
211. ATLAS Collaboration, ATLAS Muon New Small Wheel performance plots (2023). https://atlas.web.cern.ch/Atlas/GROUPS/MUON/PLOTS/MDET-2023-05
212. M. Cacciari, G.P. Salam, G. Soyez, The anti-k_t jet clustering algorithm. JHEP **04**, 063 (2008). https://doi.org/10.1088/1126-6708/2008/04/063. arXiv:0802.1189 [hep-ph]
213. ATLAS Collaboration, Jet reconstruction and performance using particle flow with the ATLAS Detector. Eur. Phys. J. C **77**,

466 (2017). https://doi.org/10.1140/epjc/s10052-017-5031-2. arXiv:1703.10485 [hep-ex]
214. M. Cacciari, G.P. Salam, G. Soyez, FastJet user manual. Eur. Phys. J. C **72**, 1896 (2012). https://doi.org/10.1140/epjc/s10052-012-1896-2. arXiv:1111.6097 [hep-ph]
215. D. Krohn, J. Thaler, L.-T. Wang, Jet trimming. JHEP **02**, 84 (2010). https://doi.org/10.1007/JHEP02(2010)084. arXiv:0912.1342 [hep-ph]
216. S.D. Ellis, C.K. Vermilion, J.R. Walsh, Recombination algorithms and jet substructure: pruning as a tool for heavy particle searches. Phys. Rev. D 81 (2010). https://doi.org/10.1103/physrevd.81.094023. arXiv:0912.0033 [hep-ph]
217. ATLAS Collaboration, Performance of missing transverse momentum reconstruction with the ATLAS detector using proton–proton collisions at $\sqrt{s} = 13$ TeV. Eur. Phys. J. C **78**, 903 (2018). https://doi.org/10.1140/epjc/s10052-018-6288-9. arXiv:1802.08168 [hep-ex]
218. W. Balunas, D. Cavalli, T. Khoo et al., A flexible and efficient approach to missing transverse momentum reconstruction. Comput. Softw. Big Sci. **8**, 2 (2024). https://doi.org/10.1007/s41781-023-00110-z. arXiv:2308.15290 [hep-ex]
219. ATLAS Collaboration, Topological b-hadron decay reconstruction and identification of b-jets with the JetFitter package in the ATLAS experiment at the LHC, ATL-PHYS-PUB-2018-025 (2018). https://cds.cern.ch/record/2645405
220. ATLAS Collaboration, Secondary vertex finding for jet flavour identification with the ATLAS detector, ATL-PHYS-PUB-2017-011 (2017). https://cds.cern.ch/record/2270366
221. A. Froch et al., Umami: A Python toolkit for jet flavour tagging in the ATLAS experiment (2023), Submitted to JOSS
222. C.R. Harris et al., Array programming with NumPy. Nature **585**, 357 (2020). https://doi.org/10.1038/s41586-020-2649-2
223. Library of Congress, HDF5, Hierarchical Data Format, Version 5 (2023). https://www.loc.gov/preservation/digital/formats/fdd/fdd000229.shtml
224. W. Falcon and The PyTorch Lightning team, PyTorch Lightning, version 1.4 (2019). https://github.com/Lightning-AI/lightning
225. M. Wang et al., Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks (2020). arXiv:1909.01315
226. ATLAS Collaboration, Identification of hadronic tau lepton decays using neural networks in the ATLAS experiment, ATL-PHYS-PUB-2019-033 (2019). https://cds.cern.ch/record/2688062
227. ATLAS Collaboration, Reconstruction, Energy Calibration, and Identification of Hadronically Decaying Tau Leptons in the ATLAS Experiment for Run-2 of the LHC, ATL-PHYS-PUB-2015-045 (2015). https://cds.cern.ch/record/2064383
228. M. Zaheer et al., Deep Sets (2017). arXiv:1703.06114
229. ATLAS Collaboration, Reconstruction of hadronic decay products of tau leptons with the ATLAS experiment. Eur. Phys. J. C **76**, 295 (2016). https://doi.org/10.1140/epjc/s10052-016-4110-0. arXiv:1512.05955 [hep-ex]
230. ATLAS Collaboration, Jet energy scale and its uncertainty for jets reconstructed using the ATLAS heavy ion jet algorithm, ATLAS-CONF-2015-016 (2015). https://cds.cern.ch/record/2008677
231. ATLAS Collaboration, Measurement of the jet radius and transverse momentum dependence of inclusive jet suppression in lead–lead collisions at $\sqrt{s_{\mathrm{NN}}} = 2.76$ TeV with the ATLAS detector. Phys. Lett. B **719**, 220 (2013). https://doi.org/10.1016/j.physletb.2013.01.024. arXiv:1208.1967 [hep-ex]
232. M. Peruzzi et al., The NanoAOD event data format in CMS. J. Phys. Conf. Ser. **1525**, 012038 (2020). https://doi.org/10.1088/1742-6596/1525/1/012038
233. P. van Gemmeren et al., Framework for custom event sample augmentations for ATLAS analysis data. EPJ Web of Conf. **295**, 03016. https://doi.org/10.1051/epjconf/202429503016 (2024)
234. A. Serhan Mete, P. van Gemmeren, Shared I/O Developments for Run 3 in the ATLAS Experiment. Proceedings of 41st International Conference on High Energy physics—PoS (ICHEP2022) **414**, 219 (2022). https://doi.org/10.22323/1.414.0219
235. ATLAS Collaboration, Luminosity determination in pp collisions at $\sqrt{s} = 13$ TeV using the ATLAS detector at the LHC. Eur. Phys. J. C **83**, 982 (2023). https://doi.org/10.1140/epjc/s10052-023-11747-w. arXiv:2212.09379 [hep-ex]
236. M. Phipps, Development of a novel reaction plane detector for event plane measurements in heavy ion collisions at the large hadron collider, PhD thesis: Illinois U., Urbana (2022). https://www.ideals.illinois.edu/items/124506
237. ATLAS Collaboration, Fast Track Reconstruction for HL-LHC, ATL-PHYS-PUB-2019-041 (2019). https://cds.cern.ch/record/2693670
238. T. Golling, H.S. Hayward, P.U.E. Onyisi, H.J. Stelzer, P. Waller, The ATLAS data quality defect database system. Eur. Phys. J. C **72**, 1960 (2012). https://doi.org/10.1140/epjc/s10052-012-1960-y. arXiv:1110.6119 [ins-det]
239. D. Salvatore et al., The GNAM system in the ATLAS online monitoring framework. Nucl. Phys. B Proc. Suppl. **172**, 317 (2007). https://doi.org/10.1016/j.nuclphysbps.2007.08.015
240. P. Adragna et al., GNAM: a low level monitoring program for the ATLAS experiment. IEEE Trans. Nucl. Sci. **2**, 1212 (2004). https://doi.org/10.1109/NSSMIC.2004.1462420
241. A. Corso-Radu et al., Data quality monitoring framework for the ATLAS experiment: performance achieved with colliding beams at the LHC. J. Phys. Conf. Ser. **331**, 022027 (2011). https://doi.org/10.1088/1742-6596/331/2/022027
242. ATLAS TDAQ Collaboration, The ATLAS Data Acquisition and High Level Trigger system. JINST **11**, P06008 (2016). https://doi.org/10.1088/1748-0221/11/06/p06008
243. Y. Ilchenko et al., Data Quality Monitoring Display for ATLAS experiment at the LHC. J. Phys. Conf. Ser. **219**, 022035 (2010). https://doi.org/10.1088/1742-6596/219/2/022035
244. A. Dotti, P. Adragna, R.A. Vitillo, The Online Histogram Presenter for the ATLAS experiment: a modular system for histogram visualization. J. Phys. Conf. Ser. **219**, 032037 (2010). https://doi.org/10.1088/1742-6596/219/3/032037
245. I. Bordulev et al., Recent Improvements to the ATLAS Offline Data Quality Monitoring System. EPJ Web Conf. **251**, 03066 (2021). https://doi.org/10.1051/epjconf/202125103066
246. Jira Software. https://www.atlassian.com/software/jira
247. G.M. Amdahl, Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18–20, 1967, Spring Joint Computer Conference 483* (1967). https://doi.org/10.1145/1465482.1465560
248. M. Böhler et al., Evolution of ATLAS conditions data and its management for LHC Run-2. J. Phys. Conf. Ser. **664**, 042005 (2015). https://doi.org/10.1088/1742-6596/664/4/042005
249. ATLAS CMake gitlab repository. https://gitlab.cern.ch/atlas/atlasexternals/-/tree/main/Build/AtlasCMake
250. HEP_OSlibs. https://gitlab.cern.ch/linuxsupport/rpms/HEP_OSlibs
251. LCG Releases. https://lcgdocs.web.cern.ch/lcgdocs/lcgreleases/introduction/
252. CPack. https://cmake.org/cmake/help/latest/module/CPack.html
253. Jenkins. https://www.jenkins.io
254. J. Elmsheuser, A. Krasznahorkay, E. Obreshkov, A. Undrus, and on behalf of the ATLAS Collaboration, A roadmap to continuous integration for ATLAS software development. J. Phys. Conf. Ser. **898**, 072009 (2017). https://doi.org/10.1088/1742-6596/898/7/072009

255. G. Dimitrov, L. Canali, M. Blaszczyk, R. Sorokoletov, ATLAS database application enhancements using Oracle 11g. J. Phys. Conf. Ser. **396**, 052027 (2012). https://doi.org/10.1088/1742-6596/396/5/052027
256. A. Alekseev et al., ATLAS BigPanDA monitoring. J. Phys. Conf. Ser. **1085**, 032043 (2018). https://doi.org/10.1088/1742-6596/1085/3/032043
257. EOS. https://eos-web.web.cern.ch/eos-web/
258. ayum. https://gitlab.cern.ch/atlas-sit/ayum/
259. CVMFS. https://cvmfs.readthedocs.io/en/stable/
260. Installation gitlab repository. https://gitlab.cern.ch/atlas-sit/code-distribution
261. Cuhadar Donszelmann, Tulay, Lampl, Walter and Stewart, Graeme A., ART ATLAS Release Tester using the Grid. EPJ Web Conf. **245**, 05015 (2020). https://doi.org/10.1051/epjconf/202024505015
262. CentOS 7. https://www.centos.org/about/
263. Alma Linux 9. https://almalinux.org/
264. GCC 11.2. https://gcc.gnu.org/gcc-11
265. GNU binutils. https://www.gnu.org/software/binutils/
266. Clang 14.0.0. https://releases.llvm.org/14.0.0/tools/clang/docs/ReleaseNotes.html
267. Building Red Hat Enterprise Linux 9 for the x86-64-v2 microarchitecture level. https://developers.redhat.com/blog/2021/01/05/building-red-hat-enterprise-linux-9-for-the-x86-64-v2-microarchitecture-level
268. Docker. https://www.docker.com/
269. S. Albrand et al., Atlas C++ Coding Standard Specification, ATL-SOFT-2002-001, Last updated in 2018 (2002). https://cds.cern.ch/record/685315/
270. Synopsys, Coverity SAST Software (2023). https://www.synopsys.com/software-integrity/security-testing/static-analysis-sast.html
271. Cppcheck: a tool for static C/C++ code analysis (2023). https://cppcheck.sourceforge.io
272. T. Yin, lizard, A simple code complexity analyser (2023). https://github.com/terryyin/lizard
273. T. Ziadé, A. Sottile, I. Cordasco, flake8 (2023). https://pypi.org/project/flake8/
274. OpenAFS. https://www.openafs.org/
275. ATLASLocalRootBase. https://gitlab.cern.ch/atlas-tier3sw/ATLASLocalRootBase
276. G.M. Kurtzer, V. Sochat, M.W. Bauer, Singularity: scientific containers for mobility of compute. PLoS One **12**, 1 (2017). https://doi.org/10.1371/journal.pone.0177459
277. Apptainer. https://apptainer.org/
278. Shifter. https://shifter.readthedocs.io/en/latest/
279. git-tools. https://gitlab.cern.ch/atlas-sit/git-tools
280. M. Barisits et al., Rucio: scientific data management. Comput. Softw. Big Sci. **3**, 11 (2019). https://doi.org/10.1007/s41781-019-0026-3 (**issn: 2510-2044**)
281. Openstack. http://opensource.com/resources/what-is-openstack
282. Puppet. https://www.puppet.com/
283. YAML. https://yaml.org/
284. The Hadoop ecosystem. https://hadoop.apache.org
285. ElasticSearch. https://www.elastic.co
286. J.P.A. Espinosa et al., A continuous integration and web framework in support of the ATLAS publication process. JINST **16**, T05006 (2021). https://doi.org/10.1088/1748-0221/16/05/T05006. arXiv:2005.06989 [cs.DL]
287. ATLAS Metadata Interface. https://ami.in2p3.fr
288. B. Blumenfeld, D. Dykstra, L. Lueking, E. Wicklund, CMS conditions data access using FroNTier. J. Phys. Conf. Ser. **119,** 072007 (2008). https://doi.org/10.1088/1742-6596/119/7/072007. http://frontier.cern.ch
289. Conditions database release. https://gitlab.cern.ch/atlcond/customdbrelease
290. D. Barberis et al., The ATLAS EventIndex: a BigData catalogue for all ATLAS experiment events. Comput. Softw. Big Sci. **7**, 2 (2023). https://doi.org/10.1007/s41781-023-00096-8. arXiv:2211.08293 [cs.DC]
291. E.I. Alexandrov et al., Development of the ATLAS Event Picking Server. In *9th International Conference on Distributed Computing and Grid Technologies in Science and Education 223* (2021). https://doi.org/10.54546/MLIT.2021.35.43.001
292. HBase database in Hadoop. https://hbase.apache.org
293. Phoenix interface to HBase. https://phoenix.apache.org
294. E.J. Gallas et al., Deployment and Operation of the ATLAS EventIndex for LHC Run 3, tech. rep., 01018 (2024)
295. E.J. Gallas et al., Conditions and configuration metadata for the ATLAS experiment. J. Phys. Conf. Ser. **396**, 052033 (2012). https://doi.org/10.1088/1742-6596/396/5/052033
296. E.J. Gallas, S. Albrand, M. Borodin, A. Formica, and the ATLAS Collaboration, Utility of collecting metadata to manage a large scale conditions database in ATLAS. J. Phys. Conf. Ser. **513**, 042020 (2014). https://doi.org/10.1088/1742-6596/513/4/042020
297. P. Laycock et al., Data preparation for NA62. EPJ Web Conf. **214**, 02017 (2019). https://doi.org/10.1051/epjconf/201921402017
298. Worldwide LHC Computing Grid (WLCG). https://wlcg.web.cern.ch/
299. E. Martelli, S. Stancu, LHCOPN and LHCONE: status and future evolution. J. Phys. Conf. Ser. **664**, 052025 (2015). https://doi.org/10.1088/1742-6596/664/5/052025
300. M. Branco et al., Managing ATLAS data on a petabyte-scale with DQ2. J. Phys. Conf. Ser. **119**, 062017 (2008). https://doi.org/10.1088/1742-6596/119/6/062017
301. Kubernetes. https://kubernetes.io/
302. Borodin Mikhail et al., The ATLAS Data Carousel Project Status. EPJ Web Conf. **251**, 02006 (2021). https://doi.org/10.1051/epjconf/202125102006
303. C. Simone, WLCG data challenges for HL-LHC-2021 planning. Zenodo (2021). https://doi.org/10.5281/zenodo.5532452
304. A. Ceccanti, E. Vianello, M. Caberletti, F. Giacomini, Beyond X.509: token-based authentication and authorization for HEP. EPJ Web Conf. **214**, 09002 (2019). https://doi.org/10.1051/epjconf/201921409002
305. Ceccanti Andrea, Vianello Enrico, Giacomini Francesco, Beyond X.509: token-based authentication and authorization in practice. EPJ Web Conf. **245**, 03021 (2020). https://doi.org/10.1051/epjconf/202024503021
306. HEPSPEC06. https://twiki.cern.ch/twiki/bin/view/FIOgroup/TsiBenchHEPSPEC
307. J.L. Henning, SPEC CPU2006 benchmark descriptions. SIGARCH Comput. Archit. News **34**, 1 (2006). https://doi.org/10.1145/1186736.1186737 (**issn: 0163-5964**)
308. D. Giordano, E. Santorinaiou, Next generation of HEP CPU benchmarks. J. Phys. Conf. Ser. **1525**, 012073 (2020). https://doi.org/10.1088/1742-6596/1525/1/012073
309. D. Giordano et al., HEPiX benchmarking solution for WLCG computing resources. Comput. Softw. Big Sci. **5**, 28 (2021). https://doi.org/10.1007/s41781-021-00074-y
310. Google Cloud Storage. https://cloud.google.com/storage
311. Cloud Computing Services—Amazon Web Services (AWS). https://aws.amazon.com/
312. Seal Storage Technologies. https://www.sealstorage.io/
313. A. Hanushevsky et al., Xcache in the ATLAS distributed computing environment. EPJ Web Conf. **214**, 04008 (2019). https://doi.org/10.1051/epjconf/201921404008
314. F.H. Barreiro et al., The ATLAS production system evolution: new data processing and analysis paradigm for the LHC Run2 and

high-luminosity. J. Phys. Conf. Ser. **898**, 052016 (2017). https://doi.org/10.1088/1742-6596/898/5/052016
315. Guan Wen et al., Towards an intelligent data delivery service. EPJ Web Conf. **245**, 04015 (2020). https://doi.org/10.1051/epjconf/202024504015
316. Barreiro Megino, Fernando Harald et al., Managing the ATLAS grid through harvester. EPJ Web Conf. **245**, 03010 (2020). https://doi.org/10.1051/epjconf/202024503010
317. M. Titov et al., Advanced analytics service to enhance workflow control at the ATLAS Production System. EPJ Web Conf. **214**, 03007 (2019). https://doi.org/10.1051/epjconf/201921403007
318. W. Guan et al., An intelligent Data Delivery Service for and beyond the ATLAS experiment. EPJ Web Conf. **251**, 02007 (2021). https://doi.org/10.1051/epjconf/202125102007. arXiv:2103.00523 [cs.DC]
319. W. Guan et al., An intelligent Data Delivery Service for and beyond the ATLAS experiment. PoS **ICHEP2022**, 218 (2022). https://doi.org/10.22323/1.414.0218
320. Django. https://docs.djangoproject.com/en/4.2/
321. C. Adam-Bourdarios et al., ATLAS@Home: harnessing volunteer computing for HEP. J. Phys. Conf. Ser. **664**, 022009 (2015). https://doi.org/10.1088/1742-6596/664/2/022009
322. D. Cameron, W. Wu, A. Bogdanchikov, R. Bianchi, Advances in ATLAS@Home towards a major ATLAS computing resource. EPJ Web Conf. **214**, 03011 (2019). https://doi.org/10.1051/epjconf/201921403011
323. Top 500 List of Supercomputers. https://www.top500.org
324. Cori Supercomputer. https://docs.nersc.gov/systems/cori/
325. Titan Supercomputer. https://www.olcf.ornl.gov/olcf-resources/compute-systems/titan/
326. SuperMUC-NG Supercomputer. https://doku.lrz.de/supermuc-ng-10745965.html
327. Toubkal Supercomputer (2023). https://www.top500.org/system/179908/
328. Perlmutter Supercomputer. https://docs.nersc.gov/systems/perlmutter/architecture/
329. Karolina Supercomputer. https://www.it4i.cz/en/infrastructure/karolina
330. Vega Supercomputer. https://doc.vega.izum.si
331. The European High Performance Computing Joint Undertaking (EuroHPC JU). https://eurohpc-ju.europa.eu/index_en
332. MareNostrum (2023). https://www.bsc.es/marenostrum/marenostrum
333. ATLAS Collaboration, Technical Design Report for the Phase-II Upgrade of the ATLAS TDAQ System, CERN-LHCC-2017-020 (2017). https://cds.cern.ch/record/2285584
334. Berghaus Frank et al., ATLAS Sim@P1 upgrades during long shutdown two. EPJ Web Conf. **245**, 07044 (2020). https://doi.org/10.1051/epjconf/202024507044
335. I. Glushkov et al., Optimization of opportunistic utilization of the ATLAS high-level trigger farm for LHC Run 3. EPJ Web Conf. **295**, 07035 (2024). https://doi.org/10.1051/epjconf/202429507035
336. A. Filipčič, arcControlTower: the system for Atlas production and analysis on ARC. J. Phys. Conf. Ser. **331**, 072013 (2011). https://doi.org/10.1088/1742-6596/331/7/072013
337. Wu. Wenjing, Cameron David, Backfilling the grid with containerized BOINC in the ATLAS computing. EPJ Web Conf. **214**, 07015 (2019). https://doi.org/10.1051/epjconf/201921407015
338. F.B. Megino et al., Operational experience and R&D results using the Google Cloud for High-Energy Physics in the ATLAS experiment. Int. J. Mod. Phys. A **39**, 2450054 (2024). https://doi.org/10.1142/S0217751X24500544. arXiv:2403.15873 [hep-ex]
339. ATLAS Collaboration, Total cost of ownership and evaluation of Google cloud resources for the ATLAS experiment at the LHC, CERN-EP-2024-124 (2024). arXiv:2405.13695 [cs.DC]
340. APACHE KAFKA (2023). https://kafka.apache.org
341. A. Aimar et al., Unified monitoring architecture for IT and grid services. J. Phys. Conf. Ser. **898**, 092033 (2017). https://doi.org/10.1088/1742-6596/898/9/092033
342. monit-docs Overview. https://monit-docs.web.cern.ch/overview/
343. Apache Spark. https://spark.apache.org/
344. InfluxDB. https://www.influxdata.com/
345. HDFS file system. https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
346. Grafana. https://grafana.com/
347. Kibana. https://www.elastic.co/kibana/
348. Bocchi Enrico et al., Facilitating collaborative analysis in SWAN. EPJ Web Conf. **214**, 07022 (2019). https://doi.org/10.1051/epjconf/201921407022
349. Beermann Thomas et al., Implementation of ATLAS distributed computing monitoring dashboards using InfluxDB and Grafana. EPJ Web Conf. **245**, 03031 (2020). https://doi.org/10.1051/epjconf/202024503031
350. ATLAS Alarm And Alert. https://aaas.atlas-ml.org/
351. T. Mkrtchyan et al., dCache: inter-disciplinary storage system. EPJ Web Conf. **251**, 02010 (2021). https://doi.org/10.1051/epjconf/202125102010
352. J. Elmsheuser et al., Evolution of the ATLAS analysis model for Run-3 and prospects for HL-LHC, ATL-SOFT-PROC-2020-002 (2020). https://cds.cern.ch/record/2708664
353. ATLAS Collaboration, Measurement of ZZ production cross-sections in the four-lepton final state in pp collisions at $\sqrt{s} =$ 13.6 TeV with the ATLAS experiment. Phys. Lett. B **855**, 138764 (2023). https://doi.org/10.1016/j.physletb.2024.138764. arXiv:2311.09715 [hep-ex]
354. A. De Salvo et al., Software installation and condition data distribution via CernVM FileSystem in ATLAS, ATL-SOFT-SLIDE-2012-206 (2012). https://cds.cern.ch/record/1448195
355. J. Iwaszkiewicz, G. Ganis, F. Rademakers, PROOF: the Parallel ROOT Facility. Scheduling and Load-balancing, PoS ACAT 022 (2009). https://doi.org/10.22323/1.050.0022
356. M. Galli, E. Tejedor, S. Wunsch, A new PyROOT: modern, interoperable and more pythonic. EPJ Web Conf. **245**, 06004 (2020). https://doi.org/10.1051/epjconf/202024506004
357. M. Rocklin, Dask: parallel computation with blocked algorithms and task scheduling. In *Proceedings of the 14th Python in Science Conference* (2015). https://doi.org/10.25080/Majora-7b98e3ed-013
358. G. Amadio et al., Novel functional and distributed approaches to data analysis available in ROOT. J. Phys. Conf. Ser. **1085**, 042008 (2018). https://doi.org/10.1088/1742-6596/1085/4/042008
359. N. Smith et al., Coffea columnar object framework for effective analysis. EPJ Web Conf. **245**, 06012 (2020). https://doi.org/10.1051/epjconf/202024506012. arXiv:2008.12712 [cs.DC]
360. M. Adamec et al., Coffea-casa: an analysis facility prototype. EPJ Web Conf. **251**, 02061 (2021). https://doi.org/10.1051/epjconf/202125102061. arXiv:2103.01871 [cs.DC]
361. ATLAS Collaboration, Characterisation and mitigation of beam-induced backgrounds observed in the ATLAS detector during the 2011 proton–proton run. JINST **8**, P07004 (2013). https://doi.org/10.1088/1748-0221/8/07/P07004. arXiv:1303.0223 [hep-ex]
362. ATLAS Collaboration, Performance of the ATLAS muon triggers in Run 2. JINST **15**, P09015 (2020). https://doi.org/10.1088/1748-0221/15/09/p09015. arXiv:2004.13447 [physics.ins-det]
363. ATLAS Collaboration, Tools for estimating fake/non-prompt lepton backgrounds with the ATLAS detector at the LHC. JINST **18**, T11004 (2023). https://doi.org/10.1088/1748-0221/18/11/T11004. arXiv:2211.16178 [hep-ex]
364. E. Rodrigues et al., The Scikit HEP Project overview and prospects. EPJ Web Conf. **245**, 06028 (2020). https://doi.org/10.

1051/epjconf/202024506028. arXiv:2007.03577 [physics.comp-ph]
365. J. Pivarski et al., Uproot, Zenodo (2017). https://doi.org/10.5281/zenodo.4340632
366. H. Schreiner, H. Dembinski, Welcome to boost-histogram's documentation! (2020). https://boost-histogram.readthedocs.io/en/latest/
367. H. Schreiner, S. Liu, A. Goel, hist, Zenodo (2020). https://doi.org/10.5281/zenodo.4057112. https://github.com/scikit-hep/hist
368. J. Pivarski et al., Awkward Array, Zenodo (2018). https://doi.org/10.5281/zenodo.4341376
369. H. Dembinski, P. Ongmongkolkul et al., scikit-hep/iminuit (2020). https://doi.org/10.5281/zenodo.3949207
370. L. Moneta et al., The RooStats Project, PoS ACAT2010 057 (2011). https://doi.org/10.22323/1.093.0057. arXiv:1009.1003 [physics.data-an]
371. W. Verkerke, D. Kirkby, The RooFit toolkit for data modeling (2003). arXiv:physics/0306116 [physics.data-an]
372. K. Cranmer, G. Lewis, L. Moneta, A. Shibata, W. Verkerke, HistFactory: a tool for creating statistical models for use with RooFit and RooStats, CERN-OPEN-2012-016 (2012). https://cds.cern.ch/record/1456844
373. M. Baak et al., HistFitter software framework for statistical data analysis. Eur. Phys. J. C **75**, 153 (2015). https://doi.org/10.1140/epjc/s10052-015-3327-7. arXiv:1410.1280 [hep-ex]
374. TRExFitter documentation. https://trexfitter-docs.web.cern.ch/trexfitter-docs/
375. L. Heinrich, M. Feickert, G. Stark, K. Cranmer, pyhf: pure-Python implementation of HistFactory statistical models. J. Open Source Softw. **6**, 2823 (2021). https://doi.org/10.21105/joss.02823
376. E. Maguire, L. Heinrich, G. Watt, HEPData: a repository for high energy physics data. J. Phys. Conf. Ser. **898**, 102006 (2017). https://doi.org/10.1088/1742-6596/898/10/102006. arXiv:1704.05473 [hep-ex]
377. ATLAS Collaboration, Reproducing searches for new physics with the ATLAS experiment through publication of full statistical likelihoods, ATL-PHYS-PUB-2019-029 (2019). https://cds.cern.ch/record/2684863
378. ATLAS Collaboration, SimpleAnalysis: Truth-level Analysis Framework, ATL-PHYS-PUB-2022-017 (2022). https://cds.cern.ch/record/2805991
379. K. Cranmer, I. Yavin, RECAST—extending the impact of existing analyses. JHEP **04**, 038 (2011). https://doi.org/10.1007/JHEP04(2011)038. arXiv:1010.2506 [hep-ex]
380. T. Šimko, L. Heinrich, H. Hirvonsalo, D. Kousidis, D. Rodríguez, REANA: a system for reusable research data analyses. EPJ Web Conf. **214**, 06034 (2019). https://doi.org/10.1051/epjconf/201921406034
381. Project Jupyter (2023). https://jupyter.org
382. R.M. Bianchi, Waiting for physics: splashing beams. https://atlas.cern/updates/blog/waiting-physics-splashing-beams
383. ATLAS Collaboration, ATLAS Public Event Displays. https://twiki.cern.ch/twiki/bin/view/AtlasPublic/EventDisplayPublicResults
384. R.M. Bianchi et al., Event visualization in ATLAS. J. Phys. Conf. Ser. **898**, 072014 (2017). https://doi.org/10.1088/1742-6596/898/7/072014
385. T. Kittelmann, V. Tsulaia, J. Boudreau, E. Moyse, The Virtual Point 1 event display for the ATLAS experiment. J. Phys. Conf. Ser. **219**, 032012 (2010). https://doi.org/10.1088/1742-6596/219/3/032012
386. Atlantis website. https://atlantis.web.cern.ch/
387. N.P. Konstantinidis et al., The Atlantis event visualisation program for the ATLAS experiment (2005). https://doi.org/10.5170/CERN-2005-002.361. http://cds.cern.ch/record/865603
388. PhoenixATLAS website. https://phoenixatlas.web.cern.ch/PhoenixATLAS/
389. PhoenixATLAS project on github. https://github.com/ATLAS-experiment/PhoenixATLAS
390. Coin, an OpenGL-based, 3D graphics library. https://github.com/coin3d/coin
391. Qt, the cross-platform software development framework. https://doc.qt.io/
392. SoQt, a Qt GUI toolkit library for Coin. https://github.com/coin3d/soqt
393. XML, Extensible Markup Language. https://www.w3.org/XML/
394. MINERVA project website. https://atlas-minerva.web.cern.ch/
395. HYPATIA project website. http://hypatia.phys.uoa.gr/
396. Microsoft, The TypeScript Programming Language. https://www.typescriptlang.org/
397. Phoenix project on github. https://github.com/HSF/phoenix
398. ATLAS Live website. https://atlas-live.cern.ch
399. ATLAS Collaboration, ATLAS High-Level Trigger, Data Acquisition and Controls: Technical Design Report, ATLAS-TDR-16; CERN-LHCC-2003-022 (2003). https://cds.cern.ch/record/616089
400. Atlas Software Documentation. https://atlassoftwaredocs.web.cern.ch/
401. ATLAS Collaboration, Search for pairs of scalar leptoquarks decaying into quarks and electrons or muons in $\sqrt{s} = 13$ TeV pp collisions with the ATLAS detector. JHEP **10**, 112 (2020). https://doi.org/10.1007/JHEP10(2020)112. arXiv:2006.05872 [hep-ex]
402. J.D. Hunter, Matplotlib: a 2D graphics environment. Comput. Sci. Eng. **9**, 90 (2007). https://doi.org/10.1109/MCSE.2007.55
403. The Matplotlib Development Team, Matplotlib: visualization with Python, Zenodo (2023). https://doi.org/10.5281/zenodo.8347255
404. VP1 Website. https://atlas-vp1.web.cern.ch/atlas-vp1/home/
405. S. Hageboeck et al., Training and onboarding initiatives in high energy physics experiments (2023). arXiv:2310.07342 [hep-ex]
406. ATLAS Collaboration, ATLAS HL-LHC Computing Conceptual Design Report, CERN-LHCC-2020-015 (2020). https://cds.cern.ch/record/2729668
407. ATLAS Collaboration, ATLAS Software and Computing HL-LHC Roadmap, CERN-LHCC-2022-005 (2022). http://cds.cern.ch/record/2802918
408. Z. Dong et al., Porting HEP parameterized calorimeter simulation code to GPUs. Front. Big Data **4**, 665783 (2021). https://doi.org/10.3389/fdata.2021.665783. arXiv:2103.14737 [hep-ex]
409. N. Fernandes, on behalf of the ATLAS Collaboration, GPU acceleration of the ATLAS calorimeter clustering algorithm. J. Phys. Conf. Ser **2438**, 012044 (2023). https://doi.org/10.1088/1742-6596/2438/1/012044
410. B. Yeo et al., CTD2022: traccc—GPU Track reconstruction demonstrator for HEP. Proceedings of CTD2022, Zenodo (2023). https://doi.org/10.5281/zenodo.8119769. https://zenodo.org/doi/10.5281/zenodo.8119863
411. S. Caillou et al., Physics Performance of the ATLAS GNN4ITk Track Reconstruction Chain, . EPJ Web of Conf. **295**, 03030. https://doi.org/10.1051/epjconf/202429503030 (2024)
412. CERN Open Data Portal (2023). https://opendata.cern.ch
413. ATLAS Collaboration, ATLAS Computing Acknowledgements, ATL-SOFT-PUB-2023-001 (2023). https://cds.cern.ch/record/2869272

ATLAS Collaboration[*]

G. Aad[104], E. Aakvaag[16], B. Abbott[122], K. Abeling[56], N. J. Abicht[49], S. H. Abidi[29], M. Aboelela[44], A. Aboulhorma[35e], H. Abramowicz[154], H. Abreu[153], Y. Abulaiti[119], E. Accion Garcia[53], B. S. Acharya[70a,70b,l], V. Acin Portella[53], A. Ackermann[64a], C. Acosta Silva[53], C. Adam Bourdarios[4], L. Adamczyk[87a], S. V. Addepalli[26], M. J. Addison[103], J. Adelman[117], A. Adiguzel[21c], T. Adye[136], A. A. Affolder[138], Y. Afik[39], M. N. Agaras[13], J. Agarwala[74a,74b], A. Aggarwal[102], C. Agheorghiesei[27c], A. Ahmad[36a], F. Ahmadov[38,z], W. S. Ahmed[106], S. Ahuja[97], X. Ai[63e], G. Aielli[77a,77b], A. Aikot[165], M. Ait Tamlihat[35e], B. Aitbenchikh[35a], M. Akbiyik[102], T. P. A. Åkesson[100], A. V. Akimov[37], D. Akiyama[170], N. N. Akolkar[24], S. Aktas[21a], K. Al Khoury[41], G. L. Alberghi[23b], J. Albert[167], P. Albicocco[54], G. L. Albouy[61], S. Alderweireldt[52], Z. L. Alegria[123], M. Aleksa[36a], I. N. Aleksandrov[38], A. Alekseev[8], C. Alexa[27b], E. Alexandrov[38], T. Alexopoulos[10], F. Alfonsi[23b], M. Algren[57], M. Alhroob[144], B. Ali[134], H. M. J. Ali[93], S. Ali[31], S. W. Alibocus[94], M. Aliev[33c], G. Alimonti[72a], W. Alkakhi[56], C. Allaire[67], B. M. M. Allbrooke[149], J. F. Allen[52], C. A. Allendes Flores[139f], P. P. Allport[20], A. Aloisio[73a,73b], F. Alonso[92], C. Alpigiani[141], M. Alvarez Estevez[101], A. Alvarez Fernandez[102], M. Alves Cardoso[57], M. G. Alviggi[73a,73b], M. Aly[103], Y. Amaral Coutinho[84b], A. Ambler[106], C. Amelung[36a], M. Amerl[103], C. G. Ames[111], D. Amidei[108], K. J. Amirie[157], S. P. Amor Dos Santos[132a], K. R. Amos[165], S. An[85], V. Ananiev[127], C. Anastopoulos[142], T. Andeen[11], J. K. Anders[36a], S. Y. Andrean[47a,47b], A. Andreazza[72a,72b], S. Angelidakis[9], A. Angerami[41,ab], A. V. Anisenkov[37], A. Annovi[75a], C. Antel[57], E. Antipov[148], M. Antonelli[54], F. Anulli[76a], M. Aoki[85], T. Aoki[156], M. A. Aparo[149], L. Aperio Bella[48], C. Appelt[18], A. Apyan[26], S. J. Arbiol Val[88], C. Arcangeletti[54], A. T. H. Arce[51], E. Arena[94], J-F. Arguin[110], S. Argyropoulos[55], J. -H. Arling[48], O. Arnaez[4], H. Arnold[116], G. Artoni[76a,76b], H. Asada[113], K. Asai[120], S. Asai[156], N. A. Asbah[36a], K. Assamagan[29], R. Astalos[28a], K. S. V. Astrand[100], S. Atashi[161], R. J. Atkin[33a], M. Atkinson[164], H. Atmani[35f], P. A. Atmasiddha[130], K. Augsten[134], S. Auricchio[73a,73b], A. D. Auriol[20], V. A. Austrup[103], G. Avolio[36a], K. Axiotis[57], G. Azuelos[110,af], D. Babal[28b], E. Bach[105], H. Bachacou[137], K. Bachas[155,p], A. Bachiu[34], F. Backman[47a,47b], A. Badea[39], T. M. Baer[108], P. Bagnaia[76a,76b], M. Bahmani[18], D. Bahner[55], K. Bai[125], J. T. Baines[136], L. Baines[96], O. K. Baker[174], E. Bakos[15], D. Bakshi Gupta[8], V. Balakrishnan[122], R. Balasubramanian[116], E. M. Baldin[37], P. Balek[87a], E. Ballabene[23a,23b], F. Balli[137], L. M. Baltes[64a], W. K. Balunas[32], J. Balz[102], E. Banas[88], M. Bandieramonte[131], A. Bandyopadhyay[24], S. Bansal[24], L. Barak[154], M. Barakat[48], E. L. Barberio[107], D. Barberis[58a,58b], M. Barbero[104], M. Z. Barel[116], K. N. Barends[33a], T. Barillari[112], M-S. Barisits[36a], T. Barklow[146], P. Baron[124], D. A. Baron Moreno[103], A. Baroncelli[63a], G. Barone[29], A. J. Barr[128], J. D. Barr[98], F. Barreiro[101], J. Barreiro Guimarães da Costa[14a], F. H. Barreiro Megino[8], U. Barron[154], M. G. Barros Teixeira[132a], S. Barsov[37], F. Bartels[64a], R. Bartoldus[146], A. E. Barton[93], P. Bartos[28a], A. Basan[102], M. Baselga[49], A. Bassalat[67,b], M. J. Basso[158a], R. Bate[166], R. L. Bates[60], S. Batlamous[101], B. Batool[144], M. Battaglia[138], D. Battulga[18], M. Bauce[76a,76b], M. Bauer[36a], P. Bauer[24], L. T. Bazzano Hurrell[30], J. B. Beacham[51], T. Beau[129], J. Y. Beaucamp[92], P. H. Beauchemin[160], P. Bechtle[24], H. P. Beck[19,o], K. Becker[169], A. J. Beddall[83], V. A. Bednyakov[38], C. P. Bee[148], L. J. Beemster[15], T. A. Beermann[36a], M. Begalli[84d], M. Begel[29], A. Behera[148], J. K. Behr[48], J. F. Beirer[36a], F. Beisiegel[24], M. Belfkir[118b], G. Bella[154], L. Bellagamba[23b], A. Bellerive[34], P. Bellos[20], K. Beloborodov[37], D. Benchekroun[35a], F. Bendebba[35a], Y. Benhammou[154], D. P. Benjamin[29], K. C. Benkendorfer[62], L. Beresford[48], M. Beretta[54], E. Bergeaas Kuutmann[163], N. Berger[4], B. Bergmann[134], J. Beringer[17a], G. Bernardi[5], C. Bernius[146], F. U. Bernlochner[24], F. Bernon[36a,104], A. Berrocal Guardia[13], T. Berry[97], P. Berta[135], A. Berthold[50], S. Bethke[112], A. Betti[76a,76b], A. J. Bevan[96], N. K. Bhalla[55], M. Bhamjee[33c], S. Bhatta[148], D. S. Bhattacharya[168], P. Bhattarai[146], K. D. Bhide[55], V. S. Bhopatkar[123], R. M. Bianchi[131], G. Bianco[23a,23b], O. Biebel[111], R. Bielski[125], M. Biglietti[78a], C. S. Billingsley[44], M. Bindi[56], A. Bingul[21b], C. Bini[76a,76b], A. Biondini[94], C. J. Birch-sykes[103], G. A. Bird[32], M. Birman[171], M. Biros[135], S. Biryukov[149], T. Bisanz[49], E. Bisceglie[43a,43b], J. P. Biswal[136], D. Biswas[144], I. Bloch[48], A. Blue[60], U. Blumenschein[96], J. Blumenthal[102], V. S. Bobrovnikov[37], M. Boehler[55], B. Boehm[168], D. Bogavac[36a], A. G. Bogdanchikov[37], C. Bohm[47a], V. Boisvert[97], P. Bokan[36a], T. Bold[87a], M. Bomben[5], M. Bona[96], M. Boonekamp[137], C. D. Booth[97], A. G. Borbély[60], I. S. Bordulev[37], H. M. Borecka-Bielska[110], G. Borissov[93], M. Borodin[81], D. Bortoletto[128], D. Boscherini[23b], M. Bosman[13], J. D. Bossio Sola[36a], K. Bouaouda[35a], N. Bouchhar[165], J. Boudreau[131], E. V. Bouhova-Thacker[93], D. Boumediene[40], R. Bouquet[58a,58b], A. Boveia[121], J. Boyd[36a], D. Boye[29], I. R. Boyko[38], L. Bozianu[57], J. Bracinik[20], N. Brahimi[4], G. Brandt[173], O. Brandt[32],

F. Braren[48], B. Brau[105], J. E. Brau[125], R. Brener[171], L. Brenner[116], R. Brenner[163], S. Bressler[171], D. Britton[60], D. Britzger[112], I. Brock[24], R. Brock[109], G. Brooijmans[41], E. Brost[29], L. M. Brown[167], L. E. Bruce[62], T. L. Bruckler[128], P. A. Bruckman de Renstrom[88], B. Brüers[48], A. Bruni[23b], G. Bruni[23b], M. Bruschi[23b], N. Bruscino[76a,76b], L. Bryant[39], T. Buanes[16], Q. Buat[141], D. Buchin[112], A. G. Buckley[60], O. Bulekov[37], B. A. Bullard[146], S. Burdin[94], C. D. Burgard[49], A. M. Burger[36a], B. Burghgrave[8], O. Burlayenko[55], J. T. P. Burr[32], C. D. Burton[11], J. C. Burzynski[145], E. L. Busch[41], V. Büscher[102], P. J. Bussey[60], J. M. Butler[25], C. M. Buttar[60], J. M. Butterworth[98], W. Buttinger[136], C. J. Buxo Vazquez[109], A. R. Buzykaev[37], S. Cabrera Urbán[165], L. Cadamuro[67], D. Caforio[59], H. Cai[131], Y. Cai[14a,14e], Y. Cai[14c], V. M. M. Cairo[36a], O. Cakir[3a], N. Calace[36a], P. Calafiura[17a], G. Calderini[129], P. Calfayan[69], G. Callea[60], L. P. Caloba[84b], D. Calvet[40], S. Calvet[40], M. Calvetti[75a,75b], R. Camacho Toro[129], S. Camarda[36a], D. Camarero Munoz[26], P. Camarri[77a,77b], M. T. Camerlingo[73a,73b], D. Cameron[36a], C. Camincher[167], M. Campanelli[98], A. Camplani[42], V. Canale[73a,73b], L. Canali[36b], A. C. Canbay[3a], E. Canonero[97], J. Cantero[165], Y. Cao[164], F. Capocasa[26], M. Capua[43a,43b], C. Caramarcu[29], A. Carbone[72a,72b], R. Cardarelli[77a], J. C. J. Cardenas[8], G. Carducci[43a,43b], T. Carli[36a], G. Carlino[73a], J. I. Carlotto[13], B. T. Carlson[131,q], E. M. Carlson[158a,167], J. Carmignani[94], L. Carminati[72a,72b], A. Carnelli[137], M. Carnesale[76a,76b], S. Caron[115], E. Carquin[139f], S. Carrá[72a], G. Carratta[23a,23b], A. M. Carroll[125], T. M. Carter[52], M. P. Casado[13,i], M. Caspar[48], F. L. Castillo[4], L. Castillo Garcia[13], V. Castillo Gimenez[165], N. F. Castro[132a,132e], A. Catinaccio[36a], J. R. Catmore[127], T. Cavaliere[4], V. Cavaliere[29], N. Cavalli[23a,23b], Y. C. Cekmecelioglu[48], E. Celebi[21a], S. Cella[36a], F. Celli[128], M. S. Centonze[71a,71b], V. Cepaitis[57], K. Cerny[124], A. S. Cerqueira[84a], A. Cerri[149], L. Cerrito[77a,77b], F. Cerutti[17a], B. Cervato[144], A. Cervelli[23b], G. Cesarini[54], S. A. Cetin[83], D. Chakraborty[117], J. Chan[17a], W. Y. Chan[156], J. D. Chapman[32], E. Chapon[137], B. Chargeishvili[152b], D. G. Charlton[20], M. Chatterjee[19], C. C. Chau[34], C. Chauhan[135], Y. Che[14c], S. Chekanov[6], S. V. Chekulaev[158a], G. A. Chelkov[38,a], S. Chelsky[145], A. Chen[108], B. Chen[154], B. Chen[167], H. Chen[14c], H. Chen[29], J. Chen[63c], J. Chen[145], M. Chen[128], S. Chen[156], S. J. Chen[14c], X. Chen[63c,137], X. Chen[14b,ae], Y. Chen[63a], C. L. Cheng[172], H. C. Cheng[65a], S. Cheong[146], A. Cheplakov[38], E. Cheremushkina[48], E. Cherepanova[116], R. Cherkaoui El Moursli[35e], E. Cheu[7], K. Cheung[66], L. Chevalier[137], V. Chiarella[54], G. Chiarelli[75a], N. Chiedde[104], G. Chiodini[71a], A. S. Chisholm[20], A. Chitan[27b], M. Chitishvili[165], M. V. Chizhov[38,r], K. Choi[11], T. Chou[29], Y. Chou[141], E. Y. S. Chow[115], D. Christidis[36a], K. L. Chu[171], M. C. Chu[65a], X. Chu[14a,14e], J. Chudoba[133], J. J. Chwastowski[88], D. Cieri[112], K. M. Ciesla[87a], V. Cindro[95], A. Ciocio[17a], F. Cirotto[73a,73b], Z. H. Citron[171], M. Citterio[72a], D. A. Ciubotaru[27b], A. Clark[57], P. J. Clark[52], C. Clarry[157], J. M. Clavijo Columbie[48], S. E. Clawson[48], C. Clement[47a,47b], J. Clercx[48], Y. Coadou[104], M. Cobal[70a,70c], A. Coccaro[58b], R. F. Coelho Barrue[132a], R. Coelho Lopes De Sa[105], S. Coelli[72a], B. Cole[41], J. Collot[61], P. Conde Muiño[132a,132g], M. P. Connell[33c], S. H. Connell[33c], E. I. Conroy[128], F. Conventi[73a,ag], H. G. Cooke[20], A. M. Cooper-Sarkar[128], F. A. Corchia[23a,23b], A. Cordeiro Oudot Choi[129], L. D. Corpe[40], M. Corradi[76a,76b], F. Corriveau[106,x], A. Cortes-Gonzalez[18], M. J. Costa[165], F. Costanza[4], D. Costanzo[142], B. M. Cote[121], J. Couthures[4], G. Cowan[97], K. Cranmer[172], D. Cremonini[23a,23b], S. Crépé-Renaudin[61], F. Crescioli[129], M. Cristinziani[144], M. Cristoforetti[79a,79b], V. Croft[116], J. E. Crosby[123], G. Crosetti[43a,43b], R. Cruz Josa[53], A. Cueto[101], Z. Cui[7], W. R. Cunningham[60], F. Curcio[165], J. R. Curran[52], P. Czodrowski[36a], M. M. Czurylo[36a], M. J. Da Cunha Sargedas De Sousa[58a,58b], J. V. Da Fonseca Pinto[84b], C. Da Via[103], W. Dabrowski[87a], T. Dado[49], S. Dahbi[151], T. Dai[108], D. Dal Santo[19], C. Dallapiccola[105], M. Dam[42], G. D'amen[29], V. D'Amico[111], J. Damp[102], J. R. Dandoy[34], D. Dannheim[36a], M. Danninger[145], V. Dao[36a], G. Darbo[58b], S. J. Das[29,ah], F. Dattola[48], S. D'Auria[72a,72b], A. D'Avanzo[73a,73b], C. David[33a], T. Davidek[135], I. Dawson[96], H. A. Day-hall[134], K. De[8], R. De Asmundis[73a], N. De Biase[48], S. De Castro[23a,23b], N. De Groot[115], P. de Jong[116], H. De la Torre[117], A. De Maria[14c], A. De Salvo[76a], U. De Sanctis[77a,77b], F. De Santis[71a,71b], A. De Santo[149], A. De Silva[158a], J. B. De Vivie De Regie[61], D. V. Dedovich[38], J. Degens[94], A. M. Deiana[44], F. Del Corso[23a,23b], J. Del Peso[101], F. Del Rio[64a], L. Delagrange[129], F. Deliot[137], C. M. Delitzsch[49], M. Della Pietra[73a,73b], D. Della Volpe[57], A. Dell'Acqua[36a], L. Dell'Asta[72a,72b], M. Delmastro[4], P. A. Delsart[61], S. Demers[174], M. Demichev[38], S. P. Denisov[37], L. D'Eramo[40], D. Derendarz[88], F. Derue[129], P. Dervan[94], K. Desch[24], J. S. DeStefano[29], C. Deutsch[24], R. Devbhandari[158a], A. Dewhurst[136], F. A. Di Bello[58a,58b], A. Di Ciaccio[77a,77b], L. Di Ciaccio[4], A. Di Domenico[76a,76b], C. Di Donato[73a,73b], A. Di Girolamo[36a], G. Di Gregorio[36a], A. Di Luca[79a,79b], B. Di Micco[78a,78b], R. Di Nardo[78a,78b], M. Diamantopoulou[34], F. A. Dias[116], T. Dias Do Vale[145], M. A. Diaz[139a,139b], F. G. Diaz Capriles[24], M. Didenko[165], E. B. Diehl[108],

S. Díez Cornell[48], C. Diez Pardos[144], C. Dimitriadi[24,163], A. Dimitrievska[20], J. Dingfelder[24], I-M. Dinu[27b], S. J. Dittmeier[64b], F. Dittus[36a], M. Divisek[135], F. Djama[104], T. Djobava[152b], C. Doglioni[100,103], A. Dohnalova[28a], J. Dolejsi[135], Z. Dolezal[135], K. M. Dona[39], M. Donadelli[84c], B. Dong[109], J. Donini[40], A. D'Onofrio[73a,73b], M. D'Onofrio[94], J. Dopke[136], A. Doria[73a], N. Dos Santos Fernandes[132a], P. Dougan[103], M. T. Dova[92], A. T. Doyle[60], M. A. Draguet[128], E. Dreyer[171], I. Drivas-koulouris[10], M. Drnevich[119], M. Drozdova[57], D. Du[63a], T. A. du Pree[116], F. Dubinin[37], M. Dubovsky[28a], E. Duchovni[171], G. Duckeck[111], O. A. Ducu[27b], D. Duda[52], A. Dudarev[36a], E. R. Duden[26], M. D'uffizi[103], L. Duflot[67], M. Dührssen[36a], I. Duminica[27g], A. E. Dumitriu[27b], M. Dunford[64a], S. Dungs[49], K. Dunne[47a,47b], A. Duperrin[104], H. Duran Yildiz[3a], M. Düren[59], A. Durglishvili[152b], B. L. Dwyer[117], G. I. Dyckes[17a], M. Dyndal[87a], B. S. Dziedzic[36a], Z. O. Earnshaw[149], G. H. Eberwein[128], B. Eckerova[28a], S. Eggebrecht[56], E. Egidio Purcino De Souza[129], L. F. Ehrke[57], G. Eigen[16], K. Einsweiler[17a], T. Ekelof[163], P. A. Ekman[100], S. El Farkh[35b], Y. El Ghazali[35b], H. El Jarrari[36a], A. El Moussaouy[110], V. Ellajosyula[163], M. Ellert[163], F. Ellinghaus[173], N. Ellis[36a], J. Elmsheuser[29], M. Elsawy[118a], M. Elsing[36a], D. Emeliyanov[136], Y. Enari[156], I. Ene[17a], S. Epari[13], P. A. Erland[88], M. Errenst[173], M. Escalier[67], C. Escobar[165], J. H. Esseiva[17a], E. Etzion[154], G. Evans[132a,132b], H. Evans[69], L. S. Evans[97], A. Ezhilov[37], S. Ezzarqtouni[35a], F. Fabbri[23a,23b], L. Fabbri[23a,23b], G. Facini[98], V. Fadeyev[138], R. M. Fakhrutdinov[37], D. Fakoudis[102], S. Falciano[76a], L. F. Falda Ulhoa Coelho[36a], F. Fallavollita[112], J. Faltova[135], C. Fan[164], Y. Fan[14a], Y. Fang[14a,14e], M. Fanti[72a,72b], M. Faraj[70a,70b], Z. Farazpay[99], A. Farbin[8], A. Farilla[78a], T. Farooque[109], S. M. Farrington[52], F. Fassi[35e], D. Fassouliotis[9], M. Faucci Giannelli[77a,77b], W. J. Fawcett[32], L. Fayard[67], P. Federic[135], P. Federicova[133], O. L. Fedin[37,a], M. Feickert[172], L. Feligioni[104], D. E. Fellers[125], C. Feng[63b], M. Feng[14b], Z. Feng[116], M. J. Fenton[161], L. Ferencz[48], R. A. M. Ferguson[93], A. F. Fernández Casani[165], F. Fernandez Galindo[158a], S. I. Fernandez Luengo[139f], P. Fernandez Martinez[13], M. J. V. Fernoux[104], J. Ferrando[93], A. Ferrari[163], P. Ferrari[115,116], R. Ferrari[74a], D. Ferrere[57], C. Ferretti[108], F. Fiedler[102], P. Fiedler[134], A. Filipčič[95], E. K. Filmer[1], F. Filthaut[115], M. C. N. Fiolhais[132a,132c,c], L. Fiorini[165], W. C. Fisher[109], T. Fitschen[103], P. M. Fitzhugh[137], I. Fleck[144], P. Fleischmann[108], T. Flick[173], M. Flores[33d,ac], L. R. Flores Castillo[65a], L. Flores Sanz De Acedo[36a], F. M. Follega[79a,79b], N. Fomin[16], J. H. Foo[157], A. Formica[137], A. C. Forti[103], E. Fortin[36a], A. W. Fortman[17a], M. G. Foti[17a], L. Fountas[9,j], D. Fournier[67], H. Fox[93], P. Francavilla[75a,75b], S. Francescato[62], S. Franchellucci[57], M. Franchini[23a,23b], S. Franchino[64a], D. Francis[36a], L. Franco[115], V. Franco Lima[36a], L. Franconi[48], M. Franklin[62], G. Frattari[26], Y. Y. Frid[154], J. Friend[60], N. Fritzsche[50], A. Froch[55], D. Froidevaux[36a], J. A. Frost[128], Y. Fu[63a], S. Fuenzalida Garrido[139f], M. Fujimoto[104], J. Fulachier[61], K. Y. Fung[65a], E. Furtado De Simas Filho[84e], M. Furukawa[156], J. Fuster[165], A. Gabrielli[23a,23b], A. Gabrielli[157], P. Gadow[36a], G. Gagliardi[58a,58b], L. G. Gagnon[17a], S. Gaid[162], S. Galantzan[154], E. J. Gallas[128], B. J. Gallop[136], C. F. Gamboa[29], K. K. Gan[121], S. Ganguly[156], Y. Gao[52], F. M. Garay Walls[139a,139b], B. Garcia[29], C. García[165], A. Garcia Alonso[116], A. G. Garcia Caffaro[174], C. Garcia Montoro[165], J. E. García Navarro[165], M. Garcia-Sciveres[17a], G. L. Gardner[130], R. W. Gardner[39], N. Garelli[160], D. Garg[81], R. B. Garg[146,m], J. M. Gargan[52], C. A. Garner[157], V. Garonne[29], C. M. Garvey[33a], V. K. Gassmann[160], G. Gaudio[74a], V. Gautam[13], P. Gauzzi[76a,76b], I. L. Gavrilenko[37], A. Gavrilyuk[37], C. Gay[166], G. Gaycken[48], E. N. Gazis[10], A. A. Geanta[27b], C. M. Gee[138], A. Gekow[121], C. Gemme[58b], M. H. Genest[61], A. D. Gentry[114], S. George[97], W. F. George[20], T. Geralis[46], P. Gessinger-Befurt[36a], M. E. Geyik[173], M. Ghani[169], K. Ghorbanian[96], A. Ghosal[144], A. Ghosh[161], A. Ghosh[7], B. Giacobbe[23b], S. Giagu[76a,76b], T. Giani[116], P. Giannetti[75a], A. Giannini[63a], S. M. Gibson[97], M. Gignac[138], D. T. Gil[87b], A. K. Gilbert[87a], B. J. Gilbert[41], D. Gillberg[34], G. Gilles[116], L. Ginabat[129], D. M. Gingrich[2,af], M. P. Giordani[70a,70c], P. F. Giraud[137], G. Giugliarelli[70a,70c], D. Giugni[72a], F. Giuli[36a], I. Gkialas[9,j], L. K. Gladilin[37], C. Glasman[101], A. Glazov[48], G. R. Gledhill[125], G. Glemža[48], M. Glisic[125], I. Glushkov[8], I. Gnesi[43b,f], Y. Go[29], M. Goblirsch-Kolb[36a], B. Gocke[49], D. Godin[110], B. Gokturk[21a], S. Goldfarb[107], T. Golling[57], F. Golnaraghi[39], M. G. D. Gololo[33g], D. Golubkov[37], J. P. Gombas[109], A. Gomes[132a,132b], G. Gomes Da Silva[144], A. J. Gomez Delegido[165], R. Gonçalo[132a], L. Gonella[20], A. Gongadze[152c], F. Gonnella[20], J. L. Gonski[146], R. Y. González Andana[52], S. González de la Hoz[165], R. Gonzalez Lopez[94], C. Gonzalez Renteria[17a], M. V. Gonzalez Rodrigues[48], R. Gonzalez Suarez[163], S. Gonzalez-Sevilla[57], L. Goossens[36a], B. Gorini[36a], E. Gorini[71a,71b], A. Gorišek[95], T. C. Gosart[130], A. T. Goshaw[51], M. I. Gostkin[38], S. Goswami[123], C. A. Gottardo[36a], S. A. Gotz[111], M. Gouighri[35b], V. Goumarre[48], A. G. Goussiou[141], N. Govender[33c], I. Grabowska-Bold[87a], K. Graham[34], E. Gramstad[127], S. Grancagnolo[71a,71b], C. M. Grant[1,137], P. M. Gravila[27f], F. G. Gravili[71a,71b], H. M. Gray[17a], M. Greco[71a,71b],

C. Grefe[24], I. M. Gregor[48], K. T. Greif[161], P. Grenier[146], S. G. Grewe[112], M. Grigoryeva[37], A. A. Grillo[138], K. Grimm[31], S. Grinstein[13,t], J. -F. Grivaz[67], L. S. Groer[157], E. Gross[171], J. Grosse-Knetter[56], J. C. Grundy[128], L. Guan[108], W. Guan[29], J. G. R. Guerrero Rojas[165], G. Guerrieri[70a,70c], F. Guescini[112], D. Guest[18], R. Gugel[102], J. A. M. Guhit[108], A. Guida[18], E. Guilloton[169], S. Guindon[36a], F. Guo[14a,14e], J. Guo[63c], L. Guo[48], Y. Guo[108], R. Gupta[131], S. Gurbuz[24], S. S. Gurdasani[55], S. Gurniak[8], G. Gustavino[36a], M. Guth[57], P. Gutierrez[122], L. F. Gutierrez Zagazeta[130], M. Gutsche[50], C. Gutschow[98], C. Gwenlan[128], C. B. Gwilliam[94], E. S. Haaland[127], A. Haas[119], M. Habedank[48], C. Haber[17a], H. K. Hadavand[8], A. Hadef[50], S. Hadzic[112], A. I. Hagan[93], J. J. Hahn[144], E. H. Haines[98], M. Haleem[168], J. Haley[123], J. J. Hall[142], G. D. Hallewell[104], L. Halser[19], K. Hamano[167], M. Hamer[24], G. N. Hamity[52], E. J. Hampshire[97], J. Han[63b], K. Han[63a], L. Han[14c], L. Han[63a], S. Han[17a], Y. F. Han[157], K. Hanagaki[85], M. Hance[138], D. A. Hangal[41], H. Hanif[145], M. D. Hank[130], J. B. Hansen[42], P. H. Hansen[42], A. Hanushevsky[146], K. Hara[159], D. Harada[57], T. Harenberg[173], S. Harkusha[37], M. L. Harris[105], Y. T. Harris[128], J. Harrison[13], N. M. Harrison[121], P. F. Harrison[169], N. M. Hartman[112], N. M. Hartmann[111], T. Hartmann[48], R. Z. Hasan[97,136], Y. Hasegawa[143], S. Hassan[16], R. Hauser[109], C. M. Hawkes[20], R. J. Hawkings[36a], Y. Hayashi[156], S. Hayashida[113], D. Hayden[109], C. Hayes[108], R. L. Hayes[116], C. P. Hays[128], J. M. Hays[96], H. S. Hayward[94], F. He[63a], M. He[14a,14e], Y. He[140], Y. He[48], Y. He[98], N. B. Heatley[96], V. Hedberg[100], A. L. Heggelund[127], N. D. Hehir[96,*], C. Heidegger[55], K. K. Heidegger[55], W. D. Heidorn[82], J. Heilman[34], S. Heim[48], T. Heim[17a], J. G. Heinlein[130], J. J. Heinrich[125], L. Heinrich[112,ad], J. Hejbal[133], A. Held[172], S. Hellesund[16], C. M. Helling[166], S. Hellman[47a,47b], R. C. W. Henderson[93], L. Henkelmann[32], A. M. Henriques Correia[36a], H. Herde[100], Y. Hernández Jiménez[148], L. M. Herrmann[24], T. Herrmann[50], G. Herten[55], R. Hertenberger[111], L. Hervas[36a], M. E. Hesping[102], N. P. Hessey[158a], M. Hidaoui[35b], E. Hill[157], S. J. Hillier[20], J. R. Hinds[109], F. Hinterkeuser[24], M. Hirose[126], S. Hirose[159], D. Hirschbuehl[173], T. G. Hitchings[103], B. Hiti[95], J. Hobbs[148], R. Hobincu[27e], N. Hod[171], M. C. Hodgkinson[142], B. H. Hodkinson[128], A. Hoecker[36a], D. D. Hofer[108], J. Hofer[48], T. Holm[24], M. Holzbock[112], L. B. A. H. Hommels[32], B. P. Honan[103], J. J. Hong[69], J. Hong[63c], T. M. Hong[131], B. H. Hooberman[164], W. H. Hopkins[6], Y. Horii[113], S. Hou[151], A. S. Howard[95], J. Howarth[60], J. Hoya[6], M. Hrabovsky[124], A. Hrynevich[48], T. Hryn'ova[4], P. J. Hsu[66], S. -C. Hsu[141], T. Hsu[67], F. Hu[39], M. Hu[17a], Q. Hu[63a], S. Huang[65b], X. Huang[14a,14e], Y. Huang[142], Y. Huang[102], Y. Huang[14a], Z. Huang[103], Z. Hubacek[134], M. Huebner[24], F. Huegging[24], T. B. Huffman[128], C. A. Hugli[48], M. Huhtinen[36a], S. K. Huiberts[16], R. Hulsken[106], N. Huseynov[12], J. Huston[109], J. Huth[62], R. Hyneman[146], G. Iacobucci[57], G. Iakovidis[29], L. Iconomidou-Fayard[67], J. P. Iddon[36a], P. Iengo[73a,73b], R. Iguchi[156], T. Iizawa[128], Y. Ikegami[85], N. Ilic[157], H. Imam[35a], M. Ince Lezki[57], T. Ingebretsen Carlson[47a,47b], G. Introzzi[74a,74b], M. Iodice[78a], V. Ippolito[76a,76b], R. K. Irwin[94], M. Ishino[156], W. Islam[172], C. Issever[18,48], S. Istin[21a,aj], H. Ito[170], H. Ito[29], R. Iuppa[79a,79b], A. Ivina[171], J. M. Izen[45], V. Izzo[73a], P. Jacka[133], P. Jackson[1], C. S. Jagfeld[111], G. Jain[158a], P. Jain[48], K. Jakobs[55], T. Jakoubek[171], J. Jamieson[60], M. Javurkova[105], L. Jeanty[125], J. Jejelava[152a,aa], P. Jenni[55,g], C. E. Jessiman[34], S. Jézéquel[4], C. Jia[63b], J. Jia[148], X. Jia[62], X. Jia[14a,14e], Z. Jia[14c], C. Jiang[52], S. Jiggins[48], J. Jimenez Pena[13], S. Jin[14c], A. Jinaru[27b], O. Jinnouchi[140], P. Johansson[142], K. A. Johns[7], J. W. Johnson[138], D. M. Jones[149], E. Jones[48], P. Jones[32], R. W. L. Jones[93], T. J. Jones[94], H. L. Joos[36a,56], D. A. Jordan[39], R. Joshi[121], J. Jovicevic[15], X. Ju[17a], J. J. Junggeburth[105], T. Junkermann[64a], A. Juste Rozas[13,t], M. K. Juzek[88], S. Kabana[139e], A. Kaczmarska[88], M. Kado[112], H. Kagan[121], M. Kagan[146], A. Kahn[130], C. Kahra[102], T. Kaji[156], E. Kajomovitz[153], N. Kakati[171], I. Kalaitzidou[55], C. W. Kalderon[29], S. Kandasamy[29], N. J. Kang[138], D. Kar[33g], K. Karava[128], E. Karavakis[29], M. J. Kareem[158b], E. Karentzos[55], O. Karkout[116], S. N. Karpov[38], Z. M. Karpova[38], V. Kartvelishvili[93], A. N. Karyukhin[37], E. Kasimi[155], J. Katzy[48], S. Kaur[34], K. Kawade[143], M. P. Kawale[122], C. Kawamoto[89], T. Kawamoto[63a], E. F. Kay[36a], F. I. Kaya[160], S. Kazakos[109], V. F. Kazanin[37], Y. Ke[148], J. M. Keaveney[33a], R. Keeler[167], G. V. Kehris[62], J. S. Keller[34], A. S. Kelly[98], J. J. Kempster[149], P. D. Kennedy[102], O. Kepka[133], B. P. Kerridge[136], S. Kersten[173], B. P. Kerševan[95], L. Keszeghova[28a], S. Ketabchi Haghighat[157], R. A. Khan[131], A. Khanov[123], A. G. Kharlamov[37], T. Kharlamova[37], E. E. Khoda[141], M. Kholodenko[37], T. J. Khoo[18], G. Khoriauli[168], J. Khubua[152b,*], Y. A. R. Khwaira[67], B. Kibirige[33g], D. W. Kim[47a,47b], Y. K. Kim[39], N. Kimura[98], M. K. Kingston[56], A. Kirchhoff[56], C. Kirfel[24], F. Kirfel[24], J. Kirk[136], A. E. Kiryunin[112], C. Kitsaki[10], O. Kivernyk[24], M. Klassen[160], C. Klein[34], L. Klein[168], M. H. Klein[44], S. B. Klein[57], U. Klein[94], P. Klimek[36a], A. Klimentov[29], T. Klioutchnikova[36a], P. Kluit[116], S. Kluth[112], E. Kneringer[80], T. M. Knight[157], A. Knue[49], R. Kobayashi[89], D. Kobylianskii[171], S. F. Koch[128], M. Kocian[146], P. Kodyš[135], D. M. Koeck[125], P. T. Koenig[24], T. Koffas[34],

O. Kolay[50], I. Koletsou[4], T. Komarek[124], V. Kondratenko[158a], K. Köneke[55], A. X. Y. Kong[1], T. Kono[120], N. Konstantinidis[98], P. Kontaxakis[57], B. Konya[100], R. Kopeliansky[41], S. Koperny[87a], T. Korchuganova[131], K. Korcyl[88], K. Kordas[155,e], A. Korn[98], S. Korn[56], I. Korolkov[13], N. Korotkova[37], B. Kortman[116], O. Kortner[112], S. Kortner[112], W. H. Kostecka[117], V. V. Kostyukhin[144], A. Kotsokechagia[137], A. Kotwal[51], A. Koulouris[36a], A. Kourkoumeli-Charalampidi[74a,74b], C. Kourkoumelis[9], E. Kourlitis[112,ad], O. Kovanda[125], R. Kowalewski[167], W. Kozanecki[137], A. S. Kozhin[37], V. A. Kramarenko[37], G. Kramberger[95], P. Kramer[102], M. W. Krasny[129], A. Krasznahorkay[36a], J. W. Kraus[173], J. A. Kremer[48], T. Kresse[50], J. Kretzschmar[94], K. Kreul[18], P. Krieger[157], S. Krishnamurthy[105], M. Krivos[135], K. Krizka[20], K. Kroeninger[49], H. Kroha[112], J. Kroll[133], J. Kroll[130], K. S. Krowpman[109], U. Kruchonak[38], H. Krüger[24], N. Krumnack[82], M. C. Kruse[51], O. Kuchinskaia[37], S. Kuday[3a], S. Kuehn[36a], R. Kuesters[55], T. Kuhl[48], V. Kukhtin[38], Y. Kulchitsky[37,a], S. Kuleshov[139b,139d], M. Kumar[33g], N. Kumari[48], P. Kumari[158b], A. Kupco[133], T. Kupfer[49], A. Kupich[37], O. Kuprash[55], H. Kurashige[86], L. L. Kurchaninov[158a], O. Kurdysh[67], Y. A. Kurochkin[37], A. Kurova[37], E. S. Kuwertz[36b], M. Kuze[140], A. K. Kvam[105], J. Kvita[124], T. Kwan[106], N. G. Kyriacou[108], L. A. O. Laatu[104], C. Lacasta[165], F. Lacava[76a,76b], H. Lacker[18], D. Lacour[129], N. N. Lad[98], E. Ladygin[38], A. Lafarge[40], B. Laforge[129], T. Lagouri[174], F. Z. Lahbabi[35a], S. Lai[56], F. Lambert[61], J. E. Lambert[167], S. Lammers[69], W. Lampl[7], C. Lampoudis[155,e], G. Lamprinoudis[102], A. N. Lancaster[117], E. Lançon[29], U. Landgraf[55], M. P. J. Landon[96], V. S. Lang[55], O. K. B. Langrekken[127], A. J. Lankford[161], F. Lanni[36a], K. Lantzsch[24], A. Lanza[74a], A. Lapertosa[58a,58b], J. F. Laporte[137], T. Lari[72a], F. Lasagni Manghi[23b], M. Lassnig[36a], V. Latonova[133], A. Laudrain[102], P. Laurens[109], A. Laurier[153], S. D. Lawlor[142], Z. Lawrence[103], R. Lazaridou[169], M. Lazzaroni[72a,72b], B. Le[103], E. M. Le Boulicaut[51], F. M. Le Flour Chollet[4], L. T. Le Pottier[17a], B. Leban[23a,23b], A. Lebedev[82], M. LeBlanc[103], F. Ledroit-Guillon[61], C. J. Lee[148], S. C. Lee[151], S. Lee[47a,47b], T. F. Lee[94], L. L. Leeuw[33c], H. P. Lefebvre[97], M. Lefebvre[167], C. Leggett[17a], G. Lehmann Miotto[36a], M. Leigh[57], W. A. Leight[105], W. Leinonen[115], A. Leisos[155,s], M. A. L. Leite[84c], C. E. Leitgeb[18], R. Leitner[135], K. J. C. Leney[44], T. Lenz[24], S. Leone[75a], C. Leonidopoulos[52], A. Leopold[147], C. Leroy[110], R. Les[109], C. G. Lester[32], M. Levchenko[37], J. Levêque[4], L. J. Levinson[171], G. Levrini[23a,23b], M. P. Lewicki[88], C. Lewis[141], D. J. Lewis[4], A. Li[5], B. Li[63b], C. Li[63a], C-Q. Li[112], H. Li[63a], H. Li[63b], H. Li[14c], H. Li[14b], H. Li[63b], J. Li[63c], K. Li[141], L. Li[63c], M. Li[14a,14e], S. Li[14a,14e], S. Li[63c,63d,d], T. Li[5], X. Li[106], Z. Li[128], Z. Li[156], Z. Li[14a,14e], S. Liang[14a,14e], Z. Liang[14a], M. Liberatore[137], B. Liberti[77a], K. Lie[65c], J. Lieber Marin[84e], H. Lien[69], F. Lin[8], K. Lin[109], R. E. Lindley[7], J. H. Lindon[2], E. Lipeles[130], A. Lipniacka[16], A. Lister[166], J. D. Little[4], B. Liu[14a], B. X. Liu[14d], D. Liu[63c,63d], E. H. L. Liu[20], J. B. Liu[63a], J. K. K. Liu[32], J. L. Liu[29], K. Liu[63d], K. Liu[63c,63d], M. Liu[63a], M. Y. Liu[63a], P. Liu[14a], Q. Liu[63c,63d,141], X. Liu[63a], X. Liu[63b], X. Liu[158a], Y. Liu[14d,14e], Y. L. Liu[63b], Y. W. Liu[63a], J. Llorente Merino[145], S. L. Lloyd[96], E. M. Lobodzinska[48], P. Loch[7], T. Lohse[18], K. Lohwasser[142], E. Loiacono[48], M. Lokajicek[133,*], J. D. Lomas[20], J. D. Long[164], I. Longarini[161], L. Longo[71a,71b], R. Longo[164], I. Lopez Paz[68], A. Lopez Solis[48], N. Lorenzo Martinez[4], A. M. Lory[111], G. Löschcke Centeno[149], O. Loseva[37], X. Lou[47a,47b], X. Lou[14a,14e], A. Lounis[67], P. A. Love[93], G. Lu[14a,14e], M. Lu[67], S. Lu[130], Y. J. Lu[66], H. J. Lubatti[141], F. Luchetti[8], C. Luci[76a,76b], F. L. Lucio Alves[14c], F. Luehring[69], I. Luise[148], O. Lukianchuk[67], O. Lundberg[147], B. Lund-Jensen[147,*], N. A. Luongo[6], M. S. Lutz[36a], A. B. Lux[25], D. Lynn[29], R. Lysak[133], E. Lytken[100], V. Lyubushkin[38], T. Lyubushkina[38], M. M. Lyukova[148], M. Firdaus M. Soberi[52], H. Ma[29], K. Ma[63a], L. L. Ma[63b], W. Ma[63a], Y. Ma[123], G. Maccarrone[54], J. C. MacDonald[102], P. C. Machado De Abreu Farias[84e], R. Madar[40], T. Madula[98], J. Maeda[86], T. Maeno[29], M. Maeno Kataoka[8], H. Maguire[142], V. Maiboroda[137], A. Maio[132a,132b,132d], K. Maj[87a], O. Majersky[48], S. Majewski[125], N. Makovec[67], V. Maksimovic[15], B. Malaescu[129], Pa. Malecki[88], V. P. Maleev[37], F. Malek[61,n], M. Mali[95], D. Malito[97], U. Mallik[81,*], S. Maltezos[10], S. Malyukov[38], J. Mamuzic[13], G. Mancini[54], M. N. Mancini[26], G. Manco[74a,74b], J. P. Mandalia[96], I. Mandić[95], L. Manhaes de Andrade Filho[84a], I. M. Maniatis[171], J. Manjarres Ramos[91], D. C. Mankad[171], A. Mann[111], S. Manzoni[36a], L. Mao[63c], X. Mapekula[33c], A. Marantis[155,s], G. Marchiori[5], M. Marcisovsky[133], C. Marcon[72a], M. Marinescu[20], S. Marium[48], M. Marjanovic[122], A. Markhoos[55], M. Markovitch[67], E. J. Marshall[93], Z. Marshall[17a], S. Marti-Garcia[165], T. A. Martin[136], V. J. Martin[52], B. Martin dit Latour[16], L. Martinelli[76a,76b], M. Martinez[13,t], P. Martinez Agullo[165], V. I. Martinez Outschoorn[105], P. Martinez Suarez[13], S. Martin-Haugh[136], G. Martinovicova[135], V. S. Martoiu[27b], A. C. Martyniuk[98], A. Marzin[36a], D. Mascione[79a,79b], L. Masetti[102], T. Mashimo[156], J. Masik[103], A. L. Maslennikov[37], P. Massarotti[73a,73b], P. Mastrandrea[75a,75b], A. Mastroberardino[43a,43b], T. Masubuchi[156], T. Mathisen[163], J. Matousek[135],

N. Matsuzawa[156], J. Maurer[27b], A. J. Maury[67], B. Maček[95], D. A. Maximov[37], A. E. May[103], R. Mazini[151], I. Maznas[117], M. Mazza[109], S. M. Mazza[138], E. Mazzeo[72a,72b], C. Mc Ginn[29], J. P. Mc Gowan[167], S. P. Mc Kee[108], C. C. McCracken[166], E. F. McDonald[107], A. E. McDougall[116], J. A. Mcfayden[149], R. P. McGovern[130], G. Mchedlidze[152b], R. P. Mckenzie[33g], T. C. Mclachlan[48], D. J. Mclaughlin[98], S. J. McMahon[136], C. M. Mcpartland[94], R. A. McPherson[167,x], S. Mehlhase[111], A. Mehta[94], D. Melini[165], B. R. Mellado Garcia[33g], A. H. Melo[56], F. Meloni[48], A. M. Mendes Jacques Da Costa[103], H. Y. Meng[157], L. Meng[93], S. Menke[112], M. Mentink[36a], E. Meoni[43a,43b], G. Mercado[117], S. Merianos[155], C. Merlassino[70a,70c], L. Merola[73a,73b], C. Meroni[72a,72b], J. Metcalfe[6], A. S. Mete[6], E. Meuser[102], C. Meyer[69], J-P. Meyer[137], R. P. Middleton[136], L. Mijović[52], G. Mikenberg[171], M. Mikestikova[133], M. Mikuž[95], H. Mildner[102], A. Milic[36a], D. W. Miller[39], E. H. Miller[146], L. S. Miller[34], A. Milov[171], D. A. Milstead[47a,47b], T. Min[14c], A. A. Minaenko[37], I. A. Minashvili[152b], L. Mince[60], A. I. Mincer[119], B. Mindur[87a], M. Mineev[38], Y. Mino[89], L. M. Mir[13], M. Miralles Lopez[60], M. Mironova[17a], S. Misawa[29], A. Mishima[156], M. C. Missio[115], A. Mitra[169], V. A. Mitsou[165], Y. Mitsumori[113], O. Miu[157], P. S. Miyagawa[96], T. Mkrtchyan[64a], M. Mlinarevic[98], T. Mlinarevic[98], M. Mlynarikova[36a], S. Mobius[19], P. Mogg[111], M. H. Mohamed Farook[114], A. F. Mohammed[14a,14e], S. Mohapatra[41], G. Mokgatitswane[33g], L. Moleri[171], B. Mondal[144], S. Mondal[134], K. Mönig[48], E. Monnier[104], L. Monsonis Romero[165], J. Montejo Berlingen[13], M. Montella[121], F. Montereali[78a,78b], F. Monticelli[92], S. Monzani[70a,70c], N. Morange[67], A. L. Moreira De Carvalho[48], M. Moreno Llácer[165], C. Moreno Martinez[57], P. Morettini[58b], B. Morgan[169], S. Morgenstern[36a], M. Morii[62], M. Morinaga[156], F. Morodei[76a,76b], L. Morvaj[36a], P. Moschovakos[36a], B. Moser[36a], M. Mosidze[152b], T. Moskalets[55], P. Moskvitina[115], J. Moss[31,k], P. Moszkowicz[87a], A. Moussa[35d], E. J. W. Moyse[105], O. Mtintsilana[33g], S. Muanza[104], J. Mueller[131], D. Muenstermann[93], R. Müller[19], G. A. Mullier[163], A. J. Mullin[32], J. J. Mullin[130], D. P. Mungo[157], D. Munoz Perez[165], F. J. Munoz Sanchez[103], M. Murin[103], D. T. Murnane[17a], W. J. Murray[136,169], H. Musheghyan[56], M. Muškinja[95], C. Mwewa[29], A. G. Myagkov[37,a], A. J. Myers[8], G. Myers[108], M. Myska[134], B. P. Nachman[17a], O. Nackenhorst[49], K. Nagai[128], K. Nagano[85], J. L. Nagle[29,ah], E. Nagy[104], A. M. Nairz[36a], Y. Nakahama[85], K. Nakamura[85], K. Nakkalil[5], H. Nanjo[126], E. A. Narayanan[114], I. Naryshkin[37], L. Nasella[72a,72b], M. Naseri[34], S. Nasri[118b], C. Nass[24], G. Navarro[22a], J. Navarro-Gonzalez[165], R. Nayak[154], A. Nayaz[18], P. Y. Nechaeva[37], S. Nechaeva[23a,23b], F. Nechansky[48], L. Nedic[128], T. J. Neep[20], A. Negri[74a,74b], M. Negrini[23b], C. Nellist[116], C. Nelson[106], K. Nelson[108], S. Nemecek[133], M. Nessi[36a,h], M. S. Neubauer[164], F. Neuhaus[102], J. Neundorf[48], P. R. Newman[20], C. W. Ng[131], Y. W. Y. Ng[48], B. Ngair[118a], H. D. N. Nguyen[110], R. B. Nickerson[128], R. Nicolaidou[137], J. Nielsen[138], M. Niemeyer[56], J. Niermann[56], N. Nikiforou[36a], V. Nikolaenko[37,a], I. Nikolic-Audit[129], K. Nikolopoulos[20], P. Nilsson[29], I. Ninca[48], G. Ninio[154], A. Nisati[76a], N. Nishu[2], R. Nisius[112], J-E. Nitschke[50], E. K. Nkadimeng[33g], T. Nobe[156], T. Nommensen[150], M. B. Norfolk[142], R. R. B. Norisam[98], B. J. Norman[34], M. Noury[35a], J. Novak[95], T. Novak[95], L. Novotny[134], R. Novotny[114], M. Nowak[29], L. Nozka[124], K. Ntekas[161], N. M. J. Nunes De Moura Junior[84b], J. Ocariz[129], A. Ochi[86], I. Ochoa[132a], J. Odier[61], S. Oerdek[48,u], J. T. Offermann[39], A. Ogrodnik[135], A. Oh[103], C. C. Ohm[147], H. Oide[85], R. Oishi[156], M. L. Ojeda[48], Y. Okumura[156], L. F. Oleiro Seabra[132a], S. A. Olivares Pino[139d], G. Oliveira Correa[13], D. Oliveira Damazio[29], D. Oliveira Goncalves[84a], J. L. Oliver[161], Ö. O. Öncel[55], A. P. O'Neill[19], A. Onofre[132a,132e], P. U. E. Onyisi[11], M. J. Oreglia[39], G. E. Orellana[92], D. Orestano[78a,78b], N. Orlando[13], R. S. Orr[157], V. O'Shea[60], L. M. Osojnak[130], R. Ospanov[63a], G. Otero y Garzon[30], H. Otono[90], P. S. Ott[64a], G. J. Ottino[17a], M. Ouchrif[35d], F. Ould-Saada[127], T. Ovsiannikova[141], M. Owen[60], R. E. Owen[136], V. E. Ozcan[21a], F. Ozturk[88], N. Ozturk[8], S. Ozturk[83], H. A. Pacey[128], A. Pacheco Pages[13], C. Padilla Aranda[13], G. Padovano[76a,76b], S. Pagan Griso[17a], G. Palacino[69], A. Palazzo[71a,71b], J. Pampel[24], J. Pan[174], T. Pan[65a], D. K. Panchal[11], C. E. Pandini[116], J. G. Panduro Vazquez[136], H. D. Pandya[1], H. Pang[14b], P. Pani[48], G. Panizzo[70a,70c], L. Panwar[129], L. Paolozzi[57], S. Parajuli[164], A. Paramonov[6], C. Paraskevopoulos[54], D. Paredes Hernandez[65b], A. Pareti[74a,74b], K. R. Park[41], T. H. Park[157], M. A. Parker[32], F. Parodi[58a,58b], E. W. Parrish[117], V. A. Parrish[52], J. A. Parsons[41], U. Parzefall[55], B. Pascual Dias[110], L. Pascual Dominguez[101], E. Pasqualucci[76a], S. Passaggio[58b], F. Pastore[97], P. Patel[88], U. M. Patel[51], J. R. Pater[103], T. Pauly[36a], C. I. Pazos[160], J. Pearkes[146], M. Pedersen[127], R. Pedro[132a], S. V. Peleganchuk[37], O. Penc[36a], E. A. Pender[52], G. D. Penn[174], K. E. Penski[111], M. Penzin[37], B. S. Peralva[84d], A. P. Pereira Peixoto[141], L. Pereira Sanchez[146], D. V. Perepelitsa[29,ah], E. Perez Codina[158a], M. Perganti[10], H. Pernegger[36a], O. Perrin[40], K. Peters[48], R. F. Y. Peters[103], B. A. Petersen[36a], T. C. Petersen[42],

E. Petit[104], V. Petousis[134], C. Petridou[155,e], T. Petru[135], A. Petrukhin[144], M. Pettee[17a], N. E. Pettersson[36a], A. Petukhov[37], K. Petukhova[135], R. Pezoa[139f], L. Pezzotti[36a], G. Pezzullo[174], T. M. Pham[172], T. Pham[107], P. W. Phillips[136], G. Piacquadio[148], E. Pianori[17a], F. Piazza[125], R. Piegaia[30], D. Pietreanu[27b], A. D. Pilkington[103], M. Pinamonti[70a,70c], J. L. Pinfold[2], B. C. Pinheiro Pereira[132a], A. E. Pinto Pinoargote[137], L. Pintucci[70a,70c], K. M. Piper[149], A. Pirttikoski[57], D. A. Pizzi[34], L. Pizzimento[65b], A. Pizzini[116], E. M. Planas Teruel[53], M. -A. Pleier[29], V. Pleskot[135], E. Plotnikova[38], G. Poddar[96], R. Poettgen[100], L. Poggioli[129], I. Pokharel[56], S. Polacek[135], G. Polesello[74a], A. Poley[145,158a], A. Polini[23b], C. S. Pollard[169], Z. B. Pollock[121], E. Pompa Pacchi[76a,76b], N. I. Pond[98], D. Ponomarenko[115], L. Pontecorvo[36a], S. Popa[27a], G. A. Popeneciu[27d], A. Poreba[36a], D. M. Portillo Quintero[158a], M. C. Porto Fernandez[53], S. Pospisil[134], M. A. Postill[142], P. Postolache[27c], K. Potamianos[169], P. A. Potepa[87a], I. N. Potrap[38], C. J. Potter[32], H. Potti[1], J. Poveda[165], M. E. Pozo Astigarraga[36a], A. Prades Ibanez[165], J. Pretel[55], D. Price[103], M. Primavera[71a], M. A. Principe Martin[101], R. Privara[124], T. Procter[60], M. L. Proffitt[141], N. Proklova[130], K. Prokofiev[65c], F. Prokoshin[38], G. Proto[112], J. Proudfoot[6], M. Przybycien[87a], W. W. Przygoda[87b], A. Psallidas[46], J. E. Puddefoot[142], D. Pudzha[37], D. Pyatiizbyantseva[37], J. Qian[108], D. Qichen[103], Y. Qin[13], D. Qing[158a], T. Qiu[52], A. Quadt[56], M. Queitsch-Maitland[103], G. Quetant[57], R. P. Quinn[166], G. Rabanal Bolanos[62], D. Rafanoharana[55], F. Raffaeli[77a,77b], F. Ragusa[72a,72b], J. L. Rainbolt[39], J. A. Raine[57], S. Rajagopalan[29], E. Ramakoti[37], I. A. Ramirez-Berend[34], K. Ran[14e,48], N. P. Rapheeha[33g], H. Rasheed[27b], V. Raskina[129], D. F. Rassloff[64a], A. Rastogi[17a], S. Rave[102], B. Ravina[56], I. Ravinovich[171], M. Raymond[36a], A. L. Read[127], N. P. Readioff[142], D. M. Rebuzzi[74a,74b], G. Redlinger[29], A. S. Reed[112], K. Reeves[26], J. A. Reidelsturz[173], D. Reikher[154], A. Rej[49], C. Rembser[36a], M. Renda[27b], M. B. Rendel[112], F. Renner[48], A. G. Rennie[161], A. L. Rescia[48], S. Resconi[72a], M. Ressegotti[58a,58b], S. Rettie[36a], J. G. Reyes Rivera[109], E. Reynolds[17a], O. L. Rezanova[37], P. Reznicek[135], H. Riani[35d], N. Ribaric[93], E. Ricci[79a,79b], R. Richter[112], S. Richter[47a,47b], E. Richter-Was[87b], M. Ridel[129], S. Ridouani[35d], P. Rieck[119], P. Riedler[36a], E. M. Riefel[47a,47b], J. O. Rieger[116], M. Rijssenbeek[148], M. Rimoldi[36a], L. Rinaldi[23a,23b], O. Rind[29], T. T. Rinn[29], M. P. Rinnagel[111], G. Ripellino[163], I. Riu[13], J. C. Rivera Vergara[167], F. Rizatdinova[123], E. Rizvi[96], B. R. Roberts[17a], S. H. Robertson[106,x], D. Robinson[32], C. M. Robles Gajardo[139f], M. Robles Manzano[102], A. Robson[60], A. Rocchi[77a,77b], C. Roda[75a,75b], S. Rodriguez Bosca[36a], Y. Rodriguez Garcia[22a], A. Rodriguez Rodriguez[55], A. M. Rodríguez Vera[117], S. Roe[36a], J. T. Roemer[161], A. R. Roepe-Gier[138], J. Roggel[173], O. Røhne[127], R. A. Rojas[105], C. P. A. Roland[129], J. Roloff[29], A. Romaniouk[37], E. Romano[74a,74b], M. Romano[23b], A. C. Romero Hernandez[164], N. Rompotis[94], L. Roos[129], S. Rosati[76a], B. J. Rosser[39], E. Rossi[128], E. Rossi[73a,73b], L. P. Rossi[62], L. Rossini[55], R. Rosten[121], M. Rotaru[27b], B. Rottler[55], C. Rougier[91], D. Rousseau[67], D. Rousso[48], A. Roy[164], S. Roy-Garand[157], A. Rozanov[104], Z. M. A. Rozario[60], Y. Rozen[153], A. Rubio Jimenez[165], A. J. Ruby[94], V. H. Ruelas Rivera[18], T. A. Ruggeri[1], A. Ruggiero[128], A. Ruiz-Martinez[165], A. Rummler[36a], Z. Rurikova[55], N. A. Rusakovich[38], H. L. Russell[167], G. Russo[76a,76b], J. P. Rutherfoord[7], S. Rutherford Colmenares[32], M. Rybar[135], G. Rybkin[67], E. B. Rye[127], A. Ryzhov[44], J. A. Sabater Iglesias[57], P. Sabatini[165], H. F-W. Sadrozinski[138], F. Safai Tehrani[76a], B. Safarzadeh Samani[136], S. Saha[1], M. Sahinsoy[112], A. Saibel[165], M. Saimpert[137], M. Saito[156], T. Saito[156], A. Sala[72a,72b], D. Salamani[36a], A. Salnikov[146], J. Salt[165], A. Salvador Salas[154], D. Salvatore[43a,43b], F. Salvatore[149], A. Salzburger[36a], D. Sammel[55], E. Sampson[93], D. Sampsonidis[155,e], D. Sampsonidou[125], J. Sánchez[165], V. Sanchez Sebastian[165], H. Sandaker[127], C. O. Sander[48], J. A. Sandesara[105], M. Sandhoff[173], C. Sandoval[22b], L. Sanfilippo[64a], D. P. C. Sankey[136], T. Sano[89], A. Sansoni[54], L. Santi[76a,76b], C. Santoni[40], H. Santos[132a,132b], A. Santra[171], E. Sanzani[23a,23b], K. A. Saoucha[162], A. Sapronov[38], J. G. Saraiva[132a,132d], J. Sardain[7], O. Sasaki[85], K. Sato[159], C. Sauer[64b], E. Sauvan[4], P. Savard[157,af], R. Sawada[156], C. Sawyer[136], L. Sawyer[99], C. Sbarra[23b], A. Sbrizzi[23a,23b], T. Scanlon[98], J. Schaarschmidt[141], U. Schäfer[102], A. C. Schaffer[44,67], D. Schaile[111], R. D. Schamberger[148], C. Scharf[18], M. M. Schefer[19], V. A. Schegelsky[37], D. Scheirich[135], M. Schernau[161], C. Scheulen[56], C. Schiavi[58a,58b], M. Schioppa[43a,43b], B. Schlag[146], K. E. Schleicher[55], S. Schlenker[36a], J. Schmeing[173], M. A. Schmidt[173], K. Schmieden[102], C. Schmitt[102], N. Schmitt[102], S. Schmitt[48], L. Schoeffel[137], A. Schoening[64b], P. G. Scholer[34], E. Schopf[128], M. Schott[24], J. Schovancova[36a], S. Schramm[57], T. Schroer[57], H-C. Schultz-Coulon[64a], M. Schulz[36b], M. Schumacher[55], B. A. Schumm[138], Ph. Schune[137], A. J. Schuy[141], H. R. Schwartz[138], A. Schwartzman[146], P. A. Schwarz[36a], T. A. Schwarz[108], Ph. Schwemling[137], R. Schwienhorst[109], F. G. Sciacca[19], A. Sciandra[29], G. Sciolla[26], F. Scuri[75a], C. D. Sebastiani[94], K. Sedlaczek[117], S. C. Seidel[114], A. Seiden[138], B. D. Seidlitz[41], C. Seitz[48], J. M. Seixas[84b], G. Sekhniaidze[73a], L. Selem[61], N. Semprini-Cesari[23a,23b], D. Sengupta[57],

V. Senthilkumar[165], C. Serfon[29], L. Serin[67], M. Sessa[77a,77b], H. Severini[122], F. Sforza[58a,58b], A. Sfyrla[57], Q. Sha[14a], E. Shabalina[56], A. H. Shah[32], R. Shaheen[147], J. D. Shahinian[130], D. Shaked Renous[171], L. Y. Shan[14a], M. Shapiro[17a], A. Sharma[36a], A. S. Sharma[166], M. Sharma[108], P. Sharma[81], P. B. Shatalov[37], K. Shaw[149], S. M. Shaw[103], Q. Shen[5,63c], D. J. Sheppard[145], P. Sherwood[98], L. Shi[98], X. Shi[14a], C. O. Shimmin[174], S. Y-H Shin[158a], J. D. Shinner[97], I. P. J. Shipsey[128,*], S. Shirabe[90], M. Shiyakova[38,v], M. J. Shochet[39], J. Shojaii[107], D. R. Shope[127], B. Shrestha[122], S. Shrestha[121,ai], M. J. Shroff[167], P. Sicho[133], A. M. Sickles[164], E. Sideras Haddad[33g], A. C. Sidley[116], A. Sidoti[23b], F. Siegert[50], Dj. Sijacki[15], F. Sili[92], J. M. Silva[52], E. Silva Junior[36b], M. V. Silva Oliveira[29], S. B. Silverstein[47a], S. Simion[67], B. Simmons[93], R. Simoniello[36a], E. L. Simpson[103], H. Simpson[149], L. R. Simpson[108], N. D. Simpson[100], S. Simsek[83], S. Sindhu[56], P. Sinervo[157], S. Singh[157], S. Sinha[48], S. Sinha[103], M. Sioli[23a,23b], I. Siral[36a], E. Sitnikova[48], J. Sjölin[47a,47b], A. Skaf[56], E. Skorda[20], P. Skubic[122], M. Slawinska[88], V. Smakhtin[171], B. H. Smart[136], S. Yu. Smirnov[37], Y. Smirnov[37], L. N. Smirnova[37,a], O. Smirnova[100], A. C. Smith[41], D. R. Smith[161], E. A. Smith[39], H. A. Smith[128], J. L. Smith[103], J. Smith[29], R. Smith[146], M. Smizanska[93], K. Smolek[134], A. A. Snesarev[37], S. R. Snider[157], H. L. Snoek[116], S. Snyder[29], R. Sobie[167,x], A. Soffer[154], C. A. Solans Sanchez[36a], E. Yu. Soldatov[37], U. Soldevila[165], A. A. Solodkov[37], S. Solomon[26], A. Soloshenko[38], I. Soloviev[161], K. Solovieva[55], O. V. Solovyanov[40], P. Sommer[36a], A. Sonay[13], W. Y. Song[158b], A. Sopczak[134], A. L. Sopio[98], F. Sopkova[28b], J. D. Sorenson[114], M. Sosebee[8], I. R. Sotarriva Alvarez[140], V. Sothilingam[64a], O. J. Soto Sandoval[139b,139c], S. Sottocornola[69], R. Soualah[162], Z. Soumaimi[35e], D. South[48], N. Soybelman[171], S. Spagnolo[71a,71b], M. Spalla[112], D. Sperlich[55], G. Spigo[36a], S. Spinali[93], D. P. Spiteri[60], M. Spousta[135], E. J. Staats[34], R. Stamen[64a], A. Stampekis[20], M. Standke[24], E. Stanecka[88], W. Stanek-Maslouska[48], M. V. Stange[50], B. Stanislaus[17a], M. M. Stanitzki[48], B. Stapf[48], E. A. Starchenko[37], G. H. Stark[138], J. Stark[91], P. Staroba[133], P. Starovoitov[64a], S. Stärz[106], R. Staszewski[88], G. Stavropoulos[46], J. Steentoft[163], P. Steinberg[29], B. Stelzer[145,158a], H. J. Stelzer[131], O. Stelzer-Chilton[158a], H. Stenzel[59], J. L. Stephen[39], T. J. Stevenson[149], G. A. Stewart[36a], J. R. Stewart[123], M. C. Stockton[36a], G. Stoicea[27b], M. Stolarski[132a], S. Stonjek[112], A. Straessner[50], J. Strandberg[147], S. Strandberg[47a,47b], M. Stratmann[173], M. Strauss[122], T. Strebler[104], P. Strizenec[28b], R. Ströhmer[168], D. M. Strom[125], R. Stroynowski[44], A. Strubig[47a,47b], S. A. Stucci[29], B. Stugu[16], J. Stupak[122], N. A. Styles[48], D. Su[146], S. Su[63a], W. Su[63d], X. Su[63a], D. Suchy[28a], K. Sugizaki[156], V. V. Sulin[37], M. J. Sullivan[94], D. M. S. Sultan[128], L. Sultanaliyeva[37], S. Sultansoy[3b], T. Sumida[89], S. Sun[108], S. Sun[172], O. Sunneborn Gudnadottir[163], N. Sur[104], M. R. Sutton[149], H. Suzuki[159], M. Svatos[133], M. Swiatlowski[158a], T. Swirski[168], I. Sykora[28a], M. Sykora[135], T. Sykora[135], M. P. Szymanski[6], D. Ta[102], K. Tackmann[48,u], A. Taffard[161], R. Tafirout[158a], J. S. Tafoya Vargas[67], Y. Takubo[85], M. Talby[104], A. A. Talyshev[37], K. C. Tam[65b], N. M. Tamir[154], A. Tanaka[156], J. Tanaka[156], R. Tanaka[67], M. Tanasini[148], Z. Tao[166], S. Tapia Araya[139f], S. Tapprogge[102], A. Tarek Abouelfadl Mohamed[109], S. Tarem[153], K. Tariq[14a], G. Tarna[27b], G. F. Tartarelli[72a], M. J. Tartarin[91], P. Tas[135], M. Tasevsky[133], E. Tassi[43a,43b], A. C. Tate[164], G. Tateno[156], Y. Tayalati[35e,w], G. N. Taylor[107], R. P. Taylor[167], W. Taylor[158b], A. S. Tee[172], R. Teixeira De Lima[146], P. Teixeira-Dias[97], J. J. Teoh[157], K. Terashi[156], J. Terron[101], S. Terzo[13], M. Testa[54], R. J. Teuscher[157,x], A. Thaler[80], O. Theiner[57], N. Themistokleous[52], T. Theveneaux-Pelzer[104], O. Thielmann[173], D. W. Thomas[97], J. P. Thomas[20], E. A. Thompson[17a], P. D. Thompson[20], E. Thomson[130], R. E. Thornberry[44], Y. Tian[56], V. Tikhomirov[37,a], Yu. A. Tikhonov[37], S. Timoshenko[37], D. Timoshyn[135], E. X. L. Ting[1], P. Tipton[174], S. H. Tlou[33g], K. Todome[140], S. Todorova-Nova[135], S. Todt[50], L. Toffolin[70a,70c], M. Togawa[85], J. Tojo[90], S. Tokár[28a], K. Tokushuku[85], O. Toldaiev[69], R. Tombs[32], M. Tomoto[85,113], L. Tompkins[146,m], K. W. Topolnicki[87b], E. Torrence[125], H. Torres[91], E. Torró Pastor[165], M. Toscani[30], C. Tosciri[39], M. Tost[11], D. R. Tovey[142], A. Traeet[16], I. S. Trandafir[27b], T. Trefzger[168], A. Tricoli[29], I. M. Trigger[158a], S. Trincaz-Duvoid[129], D. A. Trischuk[26], B. Trocmé[61], L. Truong[33c], M. Trzebinski[88], A. Trzupek[88], F. Tsai[148], M. Tsai[108], A. Tsiamis[155,e], P. V. Tsiareshka[37], S. Tsigaridas[158a], A. Tsirigotis[155,s], V. Tsiskaridze[157], E. G. Tskhadadze[152a], M. Tsopoulou[155], Y. Tsujikawa[89], I. I. Tsukerman[37], V. Tsulaia[17a], S. Tsuno[85], K. Tsuri[120], D. Tsybychev[148], Y. Tu[65b], A. Tudorache[27b], V. Tudorache[27b], A. N. Tuna[62], S. Turchikhin[58a,58b], I. Turk Cakir[3a], R. Turra[72a], T. Turtuvshin[38,y], P. M. Tuts[41], S. Tzamarias[155,e], E. Tzovara[102], F. Ukegawa[159], P. A. Ulloa Poblete[139b,139c], E. N. Umaka[29], G. Unal[36a], A. Undrus[29], G. Unel[161], J. Urban[28b], P. Urrejola[139a], G. Usai[8], R. Ushioda[140], M. Usman[110], Z. Uysal[83], V. Vacek[134], B. Vachon[106], T. Vafeiadis[36a], A. Vaitkus[98], C. Valderanis[111], E. Valdes Santurio[47a,47b], M. Valente[158a], S. Valentinetti[23a,23b], A. Valero[165], E. Valiente Moreno[165],

A. Vallier[91], J. A. Valls Ferrer[165], D. R. Van Arneman[116], T. R. Van Daalen[141], A. Van Der Graaf[49], P. Van Gemmeren[6], M. Van Rijnbach[36a], S. Van Stroud[98], I. Van Vulpen[116], P. Vana[135], M. Vanadia[77a,77b], W. Vandelli[36a], E. R. Vandewall[123], D. Vannicola[154], L. Vannoli[54], R. Vari[76a], E. W. Varnes[7], C. Varni[17b], T. Varol[151], D. Varouchas[67], L. Varriale[165], A. Vartapetian[8], K. E. Varvell[150], M. E. Vasile[27b], L. Vaslin[85], G. A. Vasquez[167], A. Vasyukov[38], R. Vavricka[102], T. Vazquez Schroeder[36a], J. Veatch[31], V. Vecchio[103], M. J. Veen[105], I. Veliscek[29], L. M. Veloce[157], F. Veloso[132a,132c], S. Veneziano[76a], A. Ventura[71a,71b], S. Ventura Gonzalez[137], A. Verbytskyi[112], M. Verducci[75a,75b], C. Vergis[96], M. Verissimo De Araujo[84b], W. Verkerke[116], J. C. Vermeulen[116], C. Vernieri[146], M. Vessella[105], M. C. Vetterli[145,af], A. Vgenopoulos[155,e], N. Viaux Maira[139f], T. Vickey[142], O. E. Vickey Boeriu[142], G. H. A. Viehhauser[128], L. Vigani[64b], M. Villa[23a,23b], M. Villaplana Perez[165], E. M. Villhauer[52], E. Vilucchi[54], M. G. Vincter[34], A. Visibile[116], C. Vittori[36a], I. Vivarelli[23a,23b], E. Voevodina[112], F. Vogel[111], M. Vogel[8], J. C. Voigt[50], P. Vokac[134], Yu. Volkotrub[87b], J. Von Ahnen[48], E. Von Toerne[24], B. Vormwald[36a], V. Vorobel[135], K. Vorobev[37], M. Vos[165], K. Voss[144], M. Vozak[116], L. Vozdecky[122], N. Vranjes[15], M. Vranjes Milosavljevic[15], M. Vreeswijk[116], N. K. Vu[63c,63d], R. Vuillermet[36a], O. Vujinovic[102], I. Vukotic[39], S. Wada[159], C. Wagner[105], J. M. Wagner[17a], W. Wagner[173], S. Wahdan[173], H. Wahlberg[92], M. Wakida[113], J. Walder[136], R. Walker[111], W. Walkowiak[144], A. Wall[130], E. J. Wallin[100], T. Wamorkar[6], A. Z. Wang[138], C. Wang[102], C. Wang[11], H. Wang[17a], J. Wang[65c], R. Wang[62], R. Wang[6], S. M. Wang[151], S. Wang[63b], S. Wang[14a], T. Wang[63a], W. T. Wang[81], W. Wang[14a], X. Wang[14c], X. Wang[164], X. Wang[63c], Y. Wang[63d], Y. Wang[14c], Z. Wang[108], Z. Wang[51,63c,63d], Z. Wang[108], A. Warburton[106], R. J. Ward[20], N. Warrack[60], S. Waterhouse[97], A. T. Watson[20], H. Watson[60], M. F. Watson[20], E. Watton[60,136], G. Watts[141], B. M. Waugh[98], J. M. Webb[55], C. Weber[29], H. A. Weber[18], M. S. Weber[19], S. M. Weber[64a], C. Wei[63a], Y. Wei[55], A. R. Weidberg[128], E. J. Weik[119], J. Weingarten[49], C. Weiser[55], C. J. Wells[48], T. Wenaus[29], B. Wendland[49], T. Wengler[36a], N. S. Wenke[112], N. Wermes[24], M. Wessels[64a], A. M. Wharton[93], A. S. White[62], A. White[8], M. J. White[1], D. Whiteson[161], L. Wickremasinghe[126], W. Wiedenmann[172], M. Wielers[136], C. Wiglesworth[42], D. J. Wilbern[122], H. G. Wilkens[36a], J. J. H. Wilkinson[32], D. M. Williams[41], H. H. Williams[130], S. Williams[32], S. Willocq[105], B. J. Wilson[103], P. J. Windischhofer[39], F. I. Winkel[30], F. Winklmeier[125], B. T. Winter[55], J. K. Winter[103], M. Wittgen[146], M. Wobisch[99], T. Wojtkowski[61], Z. Wolffs[116], J. Wollrath[161], M. W. Wolter[88], H. Wolters[132a,132c], A. Wong[29], A. Y. Wong[158a], M. C. Wong[138], E. L. Woodward[41], S. D. Worm[48], B. K. Wosiek[88], K. W. Woźniak[88], S. Wozniewski[56], K. Wraight[60], C. Wu[20], M. Wu[14d], M. Wu[115], S. L. Wu[172], W. Wu[108], X. Wu[57], Y. Wu[63a], Z. Wu[4], J. Wuerzinger[112,ad], T. R. Wyatt[103], B. M. Wynne[52], S. Xella[42], L. Xia[14c], M. Xia[14b], J. Xiang[65c], M. Xie[63a], S. Xin[14a,14e], A. Xiong[125], J. Xiong[17a], D. Xu[14a], H. Xu[63a], L. Xu[63a], R. Xu[130], T. Xu[108], Y. Xu[14b], Z. Xu[52], Z. Xu[14c], B. Yabsley[150], S. Yacoob[33a], Y. Yamaguchi[140], E. Yamashita[156], H. Yamauchi[159], T. Yamazaki[17a], Y. Yamazaki[86], J. Yan[63c], S. Yan[60], Z. Yan[105], H. J. Yang[63c,63d], H. T. Yang[63a], S. Yang[63a], T. Yang[65c], W. Yang[146], X. Yang[36a], X. Yang[14a], Y. Yang[44], Y. Yang[63a], Z. Yang[63a], W-M. Yao[17a], H. Ye[14c], H. Ye[56], J. Ye[14a], S. Ye[29], X. Ye[63a], Y. Yeh[98], I. Yeletskikh[38], B. Yeo[17b], M. R. Yexley[98], T. P. Yildirim[128], P. Yin[41], K. Yorita[170], S. Younas[27b], C. J. S. Young[36a], C. Young[146], C. Yu[14a,14e], Y. Yu[63a], M. Yuan[108], R. Yuan[63c,63d], L. Yue[98], M. Zaazoua[63a], B. Zabinski[88], E. Zaid[52], Z. K. Zak[88], T. Zakareishvili[165], N. Zakharchuk[34], S. Zambito[57], J. A. Zamora Saa[139b,139d], J. Zang[156], D. Zanzi[55], O. Zaplatilek[134], C. Zeitnitz[173], H. Zeng[14a], J. C. Zeng[164], D. T. Zenger Jr[26], O. Zenin[37], T. Ženiš[28a], S. Zenz[96], S. Zerradi[35a], D. Zerwas[67], M. Zhai[14a,14e], D. F. Zhang[142], J. Zhang[63b], J. Zhang[6], K. Zhang[14a,14e], L. Zhang[63a], L. Zhang[14c], P. Zhang[14a,14e], R. Zhang[172], S. Zhang[108], S. Zhang[91], T. Zhang[156], X. Zhang[63c], X. Zhang[63b], Y. Zhang[63c], Y. Zhang[98], Y. Zhang[14c], Z. Zhang[17a], Z. Zhang[67], H. Zhao[141], T. Zhao[63b], X. Zhao[29], Y. Zhao[138], Z. Zhao[63a], Z. Zhao[63a], A. Zhemchugov[38], J. Zheng[14c], K. Zheng[164], X. Zheng[63a], Z. Zheng[146], D. Zhong[164], B. Zhou[108], H. Zhou[7], N. Zhou[63c], Y. Zhou[14b], Y. Zhou[14c], Y. Zhou[7], C. G. Zhu[63b], J. Zhu[108], X. Zhu[63d], Y. Zhu[63c], Y. Zhu[63a], X. Zhuang[14a], K. Zhukov[37], N. I. Zimine[38], J. Zinsser[64b], M. Ziolkowski[144], L. Živković[15], A. Zoccoli[23a,23b], K. Zoch[62], T. G. Zorbas[142], O. Zormpa[46], W. Zou[41], L. Zwalinski[36a]

[1] Department of Physics, University of Adelaide, Adelaide, Australia
[2] Department of Physics, University of Alberta, Edmonton, AB, Canada
[3] [(a)]Department of Physics, Ankara University, Ankara, Türkiye; [(b)]Division of Physics, TOBB University of Economics and Technology, Ankara, Türkiye
[4] LAPP, Université Savoie Mont Blanc, CNRS/IN2P3, Annecy, France

[5] APC, Université Paris Cité, CNRS/IN2P3, Paris, France
[6] High Energy Physics Division, Argonne National Laboratory, Argonne, IL, USA
[7] Department of Physics, University of Arizona, Tucson, AZ, USA
[8] Department of Physics, University of Texas at Arlington, Arlington, TX, USA
[9] Physics Department, National and Kapodistrian University of Athens, Athens, Greece
[10] Physics Department, National Technical University of Athens, Zografou, Greece
[11] Department of Physics, University of Texas at Austin, Austin, TX, USA
[12] Institute of Physics, Azerbaijan Academy of Sciences, Baku, Azerbaijan
[13] Institut de Física d'Altes Energies (IFAE), Barcelona Institute of Science and Technology, Barcelona, Spain
[14] [(a)]Institute of High Energy Physics, Chinese Academy of Sciences, Beijing, China; [(b)]Physics Department, Tsinghua University, Beijing, China; [(c)]Department of Physics, Nanjing University, Nanjing, China; [(d)]School of Science, Shenzhen Campus of Sun Yat-sen University, Shenzhen, China; [(e)]University of Chinese Academy of Science (UCAS), Beijing, China
[15] Institute of Physics, University of Belgrade, Belgrade, Serbia
[16] Department for Physics and Technology, University of Bergen, Bergen, Norway
[17] [(a)]Physics Division, Lawrence Berkeley National Laboratory, Berkeley, CA, USA; [(b)]University of California, Berkeley, CA, USA
[18] Institut für Physik, Humboldt Universität zu Berlin, Berlin, Germany
[19] Albert Einstein Center for Fundamental Physics and Laboratory for High Energy Physics, University of Bern, Bern, Switzerland
[20] School of Physics and Astronomy, University of Birmingham, Birmingham, UK
[21] [(a)]Department of Physics, Bogazici University, Istanbul, Türkiye; [(b)]Department of Physics Engineering, Gaziantep University, Gaziantep, Türkiye; [(c)]Department of Physics, Istanbul University, Istanbul, Türkiye
[22] [(a)]Facultad de Ciencias y Centro de Investigaciónes, Universidad Antonio Nariño, Bogotá, Colombia; [(b)]Departamento de Física, Universidad Nacional de Colombia, Bogotá, Colombia
[23] [(a)]Dipartimento di Fisica e Astronomia A. Righi, Università di Bologna, Bologna, Italy; [(b)]INFN Sezione di Bologna, Bologna, Italy
[24] Physikalisches Institut, Universität Bonn, Bonn, Germany
[25] Department of Physics, Boston University, Boston, MA, USA
[26] Department of Physics, Brandeis University, Waltham, MA, USA
[27] [(a)]Transilvania University of Brasov, Brasov, Romania; [(b)]Horia Hulubei National Institute of Physics and Nuclear Engineering, Bucharest, Romania; [(c)]Department of Physics, Alexandru Ioan Cuza University of Iasi, Iasi, Romania; [(d)]Physics Department, National Institute for Research and Development of Isotopic and Molecular Technologies, Cluj-Napoca, Romania; [(e)]National University of Science and Technology Politechnica, Bucharest, Romania; [(f)]West University in Timisoara, Timisoara, Romania; [(g)]Faculty of Physics, University of Bucharest, Bucharest, Romania
[28] [(a)]Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava, Slovak Republic; [(b)]Department of Subnuclear Physics, Institute of Experimental Physics of the Slovak Academy of Sciences, Kosice, Slovak Republic
[29] Physics Department, Brookhaven National Laboratory, Upton, NY, USA
[30] Departamento de Física, y CONICET, Facultad de Ciencias Exactas y Naturales, Instituto de Física de Buenos Aires (IFIBA), Universidad de Buenos Aires, Buenos Aires, Argentina
[31] California State University, Los Angeles, CA, USA
[32] Cavendish Laboratory, University of Cambridge, Cambridge, UK
[33] [(a)]Department of Physics, University of Cape Town, Cape Town, South Africa; [(b)]iThemba Labs, Western Cape, South Africa; [(c)]Department of Mechanical Engineering Science, University of Johannesburg, Johannesburg, South Africa; [(d)]National Institute of Physics, University of the Philippines, Diliman, Philippines; [(e)]Department of Physics, University of South Africa, Pretoria, South Africa; [(f)]University of Zululand, KwaDlangezwa, South Africa; [(g)]School of Physics, University of the Witwatersrand, Johannesburg, South Africa
[34] Department of Physics, Carleton University, Ottawa, ON, Canada
[35] [(a)]Faculté des Sciences Ain Chock, Université Hassan II de Casablanca, Casablanca, Morocco; [(b)]Faculté des Sciences, Université Ibn-Tofail, Kénitra, Morocco; [(c)]Faculté des Sciences Semlalia, Université Cadi Ayyad, LPHEA-Marrakech, Morocco; [(d)]LPMR, Faculté des Sciences, Université Mohamed Premier, Oujda, Morocco; [(e)]Faculté des sciences,

Université Mohammed V, Rabat, Morocco; [f]Institute of Applied Physics, Mohammed VI Polytechnic University, Ben Guerir, Morocco
[36] [a]CERN, Geneva, Switzerland; [b]CERN Tier-0, Geneva, Switzerland
[37] Affiliated with an Institute Covered by a Cooperation Agreement with CERN, Geneva, Switzerland
[38] Affiliated with an International Laboratory Covered by a Cooperation Agreement with CERN, Geneva, Switzerland
[39] Enrico Fermi Institute, University of Chicago, Chicago, IL, USA
[40] LPC, Université Clermont Auvergne, CNRS/IN2P3, Clermont-Ferrand, France
[41] Nevis Laboratory, Columbia University, Irvington, NY, USA
[42] Niels Bohr Institute, University of Copenhagen, Copenhagen, Denmark
[43] [a]Dipartimento di Fisica, Università della Calabria, Rende, Italy; [b]INFN Gruppo Collegato di Cosenza, Laboratori Nazionali di Frascati, Frascati, Italy
[44] Physics Department, Southern Methodist University, Dallas, TX, USA
[45] Physics Department, University of Texas at Dallas, Richardson, TX, USA
[46] National Centre for Scientific Research "Demokritos", Agia Paraskevi, Greece
[47] [a]Department of Physics, Stockholm University, Stockholm, Sweden; [b]Oskar Klein Centre, Stockholm, Sweden
[48] Deutsches Elektronen-Synchrotron DESY, Hamburg and Zeuthen, Germany
[49] Fakultät Physik , Technische Universität Dortmund, Dortmund, Germany
[50] Institut für Kern- und Teilchenphysik, Technische Universität Dresden, Dresden, Germany
[51] Department of Physics, Duke University, Durham, NC, USA
[52] SUPA-School of Physics and Astronomy, University of Edinburgh, Edinburgh, UK
[53] Port d'Informacio Cientifica (PIC), Universitat Autónoma de Barcelona (UAB), Bellaterra, Spain
[54] INFN e Laboratori Nazionali di Frascati, Frascati, Italy
[55] Physikalisches Institut, Albert-Ludwigs-Universität Freiburg, Freiburg, Germany
[56] II. Physikalisches Institut, Georg-August-Universität Göttingen, Göttingen, Germany
[57] Département de Physique Nucléaire et Corpusculaire, Université de Genève, Geneva, Switzerland
[58] [a]Dipartimento di Fisica, Università di Genova, Genoa, Italy; [b]INFN Sezione di Genova, Genoa, Italy
[59] II. Physikalisches Institut, Justus-Liebig-Universität Giessen, Giessen, Germany
[60] SUPA-School of Physics and Astronomy, University of Glasgow, Glasgow, UK
[61] LPSC, Université Grenoble Alpes, CNRS/IN2P3, Grenoble INP, Grenoble, France
[62] Laboratory for Particle Physics and Cosmology, Harvard University, Cambridge, MA, USA
[63] [a]Department of Modern Physics and State Key Laboratory of Particle Detection and Electronics, University of Science and Technology of China, Hefei, China; [b]Institute of Frontier and Interdisciplinary Science and Key Laboratory of Particle Physics and Particle Irradiation (MOE), Shandong University, Qingdao, China; [c]School of Physics and Astronomy, Shanghai Jiao Tong University, Key Laboratory for Particle Astrophysics and Cosmology (MOE), SKLPPC, Shanghai, China; [d]Tsung-Dao Lee Institute, Shanghai, China; [e]School of Physics, Zhengzhou University, Zhengzhou, China
[64] [a]Kirchhoff-Institut für Physik, Ruprecht-Karls-Universität Heidelberg, Heidelberg, Germany; [b]Physikalisches Institut, Ruprecht-Karls-Universität Heidelberg, Heidelberg, Germany
[65] [a]Department of Physics, Chinese University of Hong Kong, Shatin, N.T., Hong Kong, China; [b]Department of Physics, University of Hong Kong, Hong Kong, China; [c]Department of Physics and Institute for Advanced Study, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, China
[66] Department of Physics, National Tsing Hua University, Hsinchu, Taiwan
[67] IJCLab, Université Paris-Saclay, CNRS/IN2P3, 91405 Orsay, France
[68] Centro Nacional de Microelectrónica (IMB-CNM-CSIC), Barcelona, Spain
[69] Department of Physics, Indiana University, Bloomington, IN, USA
[70] [a]INFN Gruppo Collegato di Udine, Sezione di Trieste, Udine, Italy; [b]ICTP, Trieste, Italy; [c]Dipartimento Politecnico di Ingegneria e Architettura, Università di Udine, Udine, Italy
[71] [a]INFN Sezione di Lecce, Lecce, Italy; [b]Dipartimento di Matematica e Fisica, Università del Salento, Lecce, Italy
[72] [a]INFN Sezione di Milano, Milan, Italy; [b]Dipartimento di Fisica, Università di Milano, Milan, Italy
[73] [a]INFN Sezione di Napoli, Naples, Italy; [b]Dipartimento di Fisica, Università di Napoli, Naples, Italy
[74] [a]INFN Sezione di Pavia, Pavia, Italy; [b]Dipartimento di Fisica, Università di Pavia, Pavia, Italy
[75] [a]INFN Sezione di Pisa, Pisa, Italy; [b]Dipartimento di Fisica E. Fermi, Università di Pisa, Pisa, Italy
[76] [a]INFN Sezione di Roma, Rome, Italy; [b]Dipartimento di Fisica, Sapienza Università di Roma, Rome, Italy

[77] [(a)]INFN Sezione di Roma Tor Vergata, Rome, Italy; [(b)]Dipartimento di Fisica, Università di Roma Tor Vergata, Rome, Italy
[78] [(a)]INFN Sezione di Roma Tre, Rome, Italy; [(b)]Dipartimento di Matematica e Fisica, Università Roma Tre, Rome, Italy
[79] [(a)]INFN-TIFPA, Povo, Italy; [(b)]Università degli Studi di Trento, Trento, Italy
[80] Department of Astro and Particle Physics, Universität Innsbruck, Innsbruck, Austria
[81] University of Iowa, Iowa City, IA, USA
[82] Department of Physics and Astronomy, Iowa State University, Ames, IA, USA
[83] Istinye University, Sariyer, Istanbul, Türkiye
[84] [(a)]Departamento de Engenharia Elétrica, Universidade Federal de Juiz de Fora (UFJF), Juiz de Fora, Brazil; [(b)]Universidade Federal do Rio De Janeiro COPPE/EE/IF, Rio de Janeiro, Brazil; [(c)]Instituto de Física, Universidade de São Paulo, São Paulo, Brazil; [(d)]Rio de Janeiro State University, Rio de Janeiro, Brazil; [(e)]Federal University of Bahia, Bahia, Brazil
[85] KEK, High Energy Accelerator Research Organization, Tsukuba, Japan
[86] Graduate School of Science, Kobe University, Kobe, Japan
[87] [(a)]AGH University of Krakow, Faculty of Physics and Applied Computer Science, Krakow, Poland; [(b)]Marian Smoluchowski Institute of Physics, Jagiellonian University, Krakow, Poland
[88] Institute of Nuclear Physics Polish Academy of Sciences, Krakow, Poland
[89] Faculty of Science, Kyoto University, Kyoto, Japan
[90] Research Center for Advanced Particle Physics and Department of Physics, Kyushu University, Fukuoka , Japan
[91] L2IT, Université de Toulouse, CNRS/IN2P3, UPS, Toulouse, France
[92] Instituto de Física La Plata, Universidad Nacional de La Plata and CONICET, La Plata, Argentina
[93] Physics Department, Lancaster University, Lancaster, UK
[94] Oliver Lodge Laboratory, University of Liverpool, Liverpool, UK
[95] Department of Experimental Particle Physics, Jožef Stefan Institute and Department of Physics, University of Ljubljana, Ljubljana, Slovenia
[96] School of Physics and Astronomy, Queen Mary University of London, London, UK
[97] Department of Physics, Royal Holloway University of London, Egham, UK
[98] Department of Physics and Astronomy, University College London, London, UK
[99] Louisiana Tech University, Ruston, LA, USA
[100] Fysiska Institutionen, Lunds Universitet, Lund, Sweden
[101] Departamento de Física Teorica C-15 and CIAFF, Universidad Autónoma de Madrid, Madrid, Spain
[102] Institut für Physik, Universität Mainz, Mainz, Germany
[103] School of Physics and Astronomy, University of Manchester, Manchester, UK
[104] CPPM, Aix-Marseille Université, CNRS/IN2P3, Marseille, France
[105] Department of Physics, University of Massachusetts, Amherst, MA, USA
[106] Department of Physics, McGill University, Montreal, QC, Canada
[107] School of Physics, University of Melbourne, Victoria, Australia
[108] Department of Physics, University of Michigan, Ann Arbor, MI, USA
[109] Department of Physics and Astronomy, Michigan State University, East Lansing, MI, USA
[110] Group of Particle Physics, University of Montreal, Montreal, QC, Canada
[111] Fakultät für Physik, Ludwig-Maximilians-Universität München, Munich, Germany
[112] Max-Planck-Institut für Physik (Werner-Heisenberg-Institut), Munich, Germany
[113] Graduate School of Science and Kobayashi-Maskawa Institute, Nagoya University, Nagoya, Japan
[114] Department of Physics and Astronomy, University of New Mexico, Albuquerque, NM, USA
[115] Institute for Mathematics, Astrophysics and Particle Physics, Radboud University/Nikhef, Nijmegen, The Netherlands
[116] Nikhef National Institute for Subatomic Physics and University of Amsterdam, Amsterdam, The Netherlands
[117] Department of Physics, Northern Illinois University, DeKalb, IL, USA
[118] [(a)]New York University Abu Dhabi, Abu Dhabi, United Arab Emirates; [(b)]United Arab Emirates University, Al Ain, United Arab Emirates
[119] Department of Physics, New York University, New York, NY, USA
[120] Ochanomizu University, Otsuka, Bunkyo-ku, Tokyo, Japan
[121] Ohio State University, Columbus, OH, USA
[122] Homer L. Dodge Department of Physics and Astronomy, University of Oklahoma, Norman, OK, USA

[123] Department of Physics, Oklahoma State University, Stillwater, OK, USA
[124] Joint Laboratory of Optics, Palacký University, Olomouc, Czech Republic
[125] Institute for Fundamental Science, University of Oregon, Eugene, OR, USA
[126] Graduate School of Science, Osaka University, Osaka, Japan
[127] Department of Physics, University of Oslo, Oslo, Norway
[128] Department of Physics, Oxford University, Oxford, UK
[129] LPNHE, Sorbonne Université, Université Paris Cité, CNRS/IN2P3, Paris, France
[130] Department of Physics, University of Pennsylvania, Philadelphia, PA, USA
[131] Department of Physics and Astronomy, University of Pittsburgh, Pittsburgh, PA, USA
[132] [(a)]Laboratório de Instrumentação e Física Experimental de Partículas-LIP, Lisbon, Portugal; [(b)]Departamento de Física, Faculdade de Ciências, Universidade de Lisboa, Lisbon, Portugal; [(c)]Departamento de Física, Universidade de Coimbra, Coimbra, Portugal; [(d)]Centro de Física Nuclear da Universidade de Lisboa, Lisbon, Portugal; [(e)]Departamento de Física, Universidade do Minho, Braga, Portugal; [(f)]Departamento de Física Teórica y del Cosmos, Universidad de Granada, Granada, Spain; [(g)]Departamento de Física, Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal
[133] Institute of Physics of the Czech Academy of Sciences, Prague, Czech Republic
[134] Czech Technical University in Prague, Prague, Czech Republic
[135] Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic
[136] Particle Physics Department, Rutherford Appleton Laboratory, Didcot, UK
[137] IRFU, CEA, Université Paris-Saclay, Gif-sur-Yvette, France
[138] Santa Cruz Institute for Particle Physics, University of California Santa Cruz, Santa Cruz, CA, USA
[139] [(a)]Departamento de Física, Pontificia Universidad Católica de Chile, Santiago, Chile; [(b)]Millennium Institute for Subatomic Physics at High Energy Frontier (SAPHIR), Santiago, Chile; [(c)]Instituto de Investigación Multidisciplinario en Ciencia y Tecnología, y Departamento de Física, Universidad de La Serena, La Serena, Chile; [(d)]Department of Physics, Universidad Andres Bello, Santiago, Chile; [(e)]Instituto de Alta Investigación, Universidad de Tarapacá, Arica, Chile; [(f)]Departamento de Física, Universidad Técnica Federico Santa María, Valparaíso, Chile
[140] Department of Physics, Institute of Science, Tokyo, Japan
[141] Department of Physics, University of Washington, Seattle, WA, USA
[142] Department of Physics and Astronomy, University of Sheffield, Sheffield, UK
[143] Department of Physics, Shinshu University, Nagano, Japan
[144] Department Physik, Universität Siegen, Siegen, Germany
[145] Department of Physics, Simon Fraser University, Burnaby, BC, Canada
[146] SLAC National Accelerator Laboratory, Stanford, CA, USA
[147] Department of Physics, Royal Institute of Technology, Stockholm, Sweden
[148] Departments of Physics and Astronomy, Stony Brook University, Stony Brook, NY, USA
[149] Department of Physics and Astronomy, University of Sussex, Brighton, UK
[150] School of Physics, University of Sydney, Sydney, Australia
[151] Institute of Physics, Academia Sinica, Taipei, Taiwan
[152] [(a)]E. Andronikashvili Institute of Physics, Iv. Javakhishvili Tbilisi State University, Tbilisi, Georgia; [(b)]High Energy Physics Institute, Tbilisi State University, Tbilisi, Georgia; [(c)]University of Georgia, Tbilisi, Georgia
[153] Department of Physics, Technion, Israel Institute of Technology, Haifa, Israel
[154] Raymond and Beverly Sackler School of Physics and Astronomy, Tel Aviv University, Tel Aviv, Israel
[155] Department of Physics, Aristotle University of Thessaloniki, Thessaloniki, Greece
[156] International Center for Elementary Particle Physics and Department of Physics, University of Tokyo, Tokyo, Japan
[157] Department of Physics, University of Toronto, Toronto, ON, Canada
[158] [(a)]TRIUMF, Vancouver, BC, Canada; [(b)]Department of Physics and Astronomy, York University, Toronto, ON, Canada
[159] Division of Physics and Tomonaga Center for the History of the Universe, Faculty of Pure and Applied Sciences, University of Tsukuba, Tsukuba, Japan
[160] Department of Physics and Astronomy, Tufts University, Medford, MA, USA
[161] Department of Physics and Astronomy, University of California Irvine, Irvine, CA, USA
[162] University of Sharjah, Sharjah, United Arab Emirates
[163] Department of Physics and Astronomy, University of Uppsala, Uppsala, Sweden
[164] Department of Physics, University of Illinois, Urbana, IL, USA
[165] Instituto de Física Corpuscular (IFIC), Centro Mixto Universidad de Valencia-CSIC, Valencia, Spain

[166] Department of Physics, University of British Columbia, Vancouver, BC, Canada
[167] Department of Physics and Astronomy, University of Victoria, Victoria, BC, Canada
[168] Fakultät für Physik und Astronomie, Julius-Maximilians-Universität Würzburg, Würzburg, Germany
[169] Department of Physics, University of Warwick, Coventry, UK
[170] Waseda University, Tokyo, Japan
[171] Department of Particle Physics and Astrophysics, Weizmann Institute of Science, Rehovot, Israel
[172] Department of Physics, University of Wisconsin, Madison, WI, USA
[173] Fakultät für Mathematik und Naturwissenschaften, Fachgruppe Physik, Bergische Universität Wuppertal, Wuppertal, Germany
[174] Department of Physics, Yale University, New Haven, CT, USA

[a] Also Affiliated with an institute covered by a cooperation agreement with CERN, Geneva, Switzerland
[b] Also at An-Najah National University, Nablus, Palestine
[c] Also at Borough of Manhattan Community College, City University of New York, New York, NY, USA
[d] Also at Center for High Energy Physics, Peking University, China
[e] Also at Center for Interdisciplinary Research and Innovation (CIRI-AUTH), Thessaloniki, Greece
[f] Also at Centro Studi e Ricerche Enrico Fermi, Rome, Italy
[g] Also at CERN, Geneva, Switzerland
[h] Also at Département de Physique Nucléaire et Corpusculaire, Université de Genève, Geneva, Switzerland
[i] Also at Departament de Fisica de la Universitat Autonoma de Barcelona, Barcelona, Spain
[j] Also at Department of Financial and Management Engineering, University of the Aegean, Chios, Greece
[k] Also at Department of Physics, California State University, Sacramento, USA
[l] Also at Department of Physics, King's College London, London, UK
[m] Also at Department of Physics, Stanford University, Stanford, CA, USA
[n] Also at Department of Physics, Stellenbosch University, Stellenbosch, South Africa
[o] Also at Department of Physics, University of Fribourg, Fribourg, Switzerland
[p] Also at Department of Physics, University of Thessaly, Thessaly, Greece
[q] Also at Department of Physics, Westmont College, Santa Barbara, USA
[r] Also at Faculty of Physics, Sofia University, 'St. Kliment Ohridski', Sofia, Bulgaria
[s] Also at Hellenic Open University, Patras, Greece
[t] Also at Institucio Catalana de Recerca i Estudis Avancats, ICREA, Barcelona, Spain
[u] Also at Institut für Experimentalphysik, Universität Hamburg, Hamburg, Germany
[v] Also at Institute for Nuclear Research and Nuclear Energy (INRNE) of the Bulgarian Academy of Sciences, Sofia, Bulgaria
[w] Also at Institute of Applied Physics, Mohammed VI Polytechnic University, Ben Guerir, Morocco
[x] Also at Institute of Particle Physics (IPP), Toronto, Canada
[y] Also at Institute of Physics and Technology, Mongolian Academy of Sciences, Ulaanbaatar, Mongolia
[z] Also at Institute of Physics, Azerbaijan Academy of Sciences, Baku, Azerbaijan
[aa] Also at Institute of Theoretical Physics, Ilia State University, Tbilisi, Georgia
[ab] Also at Lawrence Livermore National Laboratory, Livermore, USA
[ac] Also at National Institute of Physics, University of the Philippines, Diliman, Philippines
[ad] Also at Technical University of Munich, Munich, Germany
[ae] Also at The Collaborative Innovation Center of Quantum Matter (CICQM), Beijing, China
[af] Also at TRIUMF, Vancouver, BC, Canada
[ag] Also at Università di Napoli Parthenope, Naples, Italy
[ah] Also at Department of Physics, University of Colorado Boulder, Boulder, CO, USA
[ai] Also at Washington College, Chestertown, MD, USA
[aj] Also at Physics Department, Yeditepe University, Istanbul, Türkiye
* Deceased