Proceedings Paper:
Pasupuleti, Suryachandra Prasad and Wright, Steven A. (2026) The Performance-Power Frontier: A Model-Driven Approach to Energy-Aware Application Optimisation. In: ACM International Conference on Supercomputing. ACM International Conference on Supercomputing, 06-09 Jul 2026 ACM, GBR.

https://doi.org/10.1145/3797905.3807859

The Performance-Power Frontier: A Model-Driven Approach to Energy-Aware Application Optimisation

Suryachandra Pasupuleti
University of York
York, United Kingdom
sp1822@york.ac.uk

Steven A. Wright
University of York
York, United Kingdom
steven.wright@york.ac.uk

Abstract

Energy consumption in high-performance computing (HPC) has emerged as a primary design constraint, with Tier-0 systems now demanding tens of megawatts. While users are increasingly being encouraged to minimise the environmental and operational costs of their workloads, they frequently lack the analytical frameworks required to navigate the complex trade-offs between execution time and energy efficiency. In this paper, we extend the Power-Optimised Software Envelope (POSE) model and introduce a visualisation tool designed to help performance engineers explore the energy-time design space. By aligning multi-objective optimisation metrics with institutional operational priorities, our framework enables policy-aware decision-making at the application level. We demonstrate this approach by analysing the NAS Parallel Benchmarks. Our results reveal a significant variance in energy-aware optimisation potential: while the Embarrassingly Parallel (EP) kernel shows negligible opportunity for gains (up to 0.37% improvement in Energy-Delay Summation), the Conjugate Gradient (CG) kernel offers greater scope (up to a 12.96% improvement). Furthermore, we study the benchmarks under frequency and power constraints and identify critical efficiency thresholds; for the Block Tri-Diagonal (BT) solver, configurations operating below 2.00 GHz or under a 140 W power-cap fail to outperform the baseline efficiency, highlighting the risks of aggressive over-throttling.

CCS Concepts

• **Computer systems organization** → *Parallel architectures*; • **Computing methodologies** → **Parallel computing methodologies**; • **Hardware** → **Power estimation and optimization**.

ACM Reference Format:
Suryachandra Pasupuleti and Steven A. Wright. 2026. The Performance-Power Frontier: A Model-Driven Approach to Energy-Aware Application Optimisation. In *2026 International Conference on Supercomputing (ICS '26), July 06–09, 2026, Belfast, United Kingdom.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3797905.3807859

1 Introduction

HPC systems are among the most energy-intensive scientific facilities. Even if every new system generation is more energy efficient than the previous one, the number and size of supercomputers are continuously increasing, driven by the growing demand for computational capacity across all scientific disciplines, and artificial intelligence (AI) workloads [19]; in turn, this further drives up the energy demand. The sustainability of HPC systems has become a major concern. Traditionally, increasing the performance used to be the primary goal in supercomputing, but as we are reaching (and surpassing) the Exascale (i.e., 10^{18} floating-point operations per second) computing era, minimising power consumption is becoming a primary concern. Analysis of jobs running on thousands of nodes show that significant power swings can be observed, which can lead to system instability [22]. Additionally, large fluctuations in power can impose challenges on operational power management resulting in overall system performance loss. The HPC community are actively investigating power-efficient computing techniques.

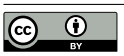
The simplest approach is to allow users to decide appropriate frequency settings, as they are best placed to understand the requirements of their applications, but Suarez et al. [19] suggest that this may not to be the case; users are often not well informed about the performance and energy properties of their own workloads. The scenario could change if HPC centres started to account for energy instead of CPU and GPU hours, giving users a strong incentive to care about resource efficiency.

Solórzano et al. present an interesting approach to improve the power efficiency and sustainability of supercomputers [18]. Users of Fugaku, the largest HPC system in Japan at the time, were provided with "knobs" to apply power control functions to their workloads, with users awarded through an incentive-based program for reducing their energy consumption. As a result, they found that there is a critical need for tools to guide users to select the power knobs that are more suitable for their workloads. As power consumption changes from workload to workload, there is a need for power consumption tools and models to analyse the characteristics of the workload, and their impact on power consumption.

With power as a critical limiting factor in future system designs, application developers need to consider both runtime and power efficiency [22]. Existing optimisation practices must be re-evaluated to optimise both runtime and power, and existing tools aimed at runtime optimisation need to be augmented to additionally support power optimisation.

To guide the users of HPC systems in selecting appropriate power controller knobs suitable for their applications in order to contribute to the sustainability of HPC systems, this work introduces the following contributions:

- We introduce the Cache-Aware Power-Optimised Software Envelope (CA-POSE), an extension of the existing Power-Optimised Software Envelope (POSE) model [13, 14]. Our extended model uses an application's arithmetic intensity to



ICS '26, Belfast, United Kingdom

ACM ISBN 979-8-4007-2522-7/2026/07
https://doi.org/10.1145/3797905.3807859

better capture minimum and maximum power draws, and therefore provides more realistic bounds for energy-aware optimisation;
- We outline the development of a POSE toolkit that produces POSE and CA-POSE visual heuristic models for application developers to use to analyse the scope for power-aware optimisation of their own applications;
- We demonstrate our model and our visualisation toolkit on applications from the NASA Parallel Benchmarks, highlighting opportunities for improving the energy-efficiency of core scientific algorithms.

The remainder of this paper is structured as follows: Section 2 provides an overview of existing work in energy-aware performance engineering; Section 3 introduces our Cache-Aware POSE model; Section 4 outlines a tool to plot POSE and CA-POSE models for analysis; Section 5 applies our extended model and our visualisation toolkit to the NASA Parallel Benchmarks, highlighting opportunities for energy-aware optimisation; and finally, Section 6 concludes this paper.

2 Background and Related Work

Power budgets are becoming a major limitation, potentially preventing HPC systems from achieving significant performance improvements [4]. The demand for computing power has increased over time, this in turn increases the energy costs and the carbon footprint. The monetary cost of running HPC systems is often greater than the cost of purchasing and maintaining them [1]. HPC centres must be prepared for power grid emergencies, such as power cuts during wildfires in the USA and power outages during storms in Japan [6]. The carbon emissions of data and HPC centres exceed that of today's airline industry. Emerging AI models are estimated to draw up to 21% of the world's electricity supply by 2030 [20].

The number and size of HPC centres are increasing due to a rising demand for computational facilities for scientific computing and AI workloads, increasing the energy demand. This is a significant challenge for HPC centres in countries like Germany, due to its stringent national policies, high energy costs, and its commitment to environmental sustainability [19]. Saurez et al. note that the German Energy Efficiency Act mandates that commercial data centres and supercomputing centres observe stricter energy consumption limits. They advise that HPC centres should balance the need for advanced computing infrastructure with sustainability efforts by exploring cutting-edge solutions such as energy-efficient hardware, system monitoring and management, and optimised cooling systems.

Several power-efficient computing techniques have been investigated by the HPC community. Karimi et al. present a telemetry data-driven approach to derive energy saving projections on the Frontier supercomputer [4]. They report that for certain resource-constrained jobs, energy savings of up to 8.5% can be achieved without compromising the performance. They evaluate the impact of software-driven power management techniques like Dynamic Voltage and Frequency Scaling (DVFS) and Power Capping. Karimi et al. argue that even though these techniques have been developed over the past few decades, there is a need to understand the effect of software-driven power management techniques on newer hardware architectures designed based on chiplets.

Power efficiency was considered an important design factor in the development of the Fugaku supercomputer in Japan, and various power control mechanisms were implemented and made available to users of the supercomputer [5]. Users can set the "power knobs" by using parameters in the job script submitted to the job scheduler. To incentivise the usage of power knobs, the system administrators reward the groups consuming less power than expected with "Fugaku Points", which can be used as node-hours in a reserved priority queue [18]. They find that users engage with the power control knobs but have little idea which settings work best for their application codes. Solórzano et al. imply that a "one-size-fits-all" power-control mechanism for saving power may not be optimally effective in practice. As a result, they find that there is a critical need for tools to guide users to select the power knobs that are more suitable for their workloads.

2.1 Energy-Aware Metrics

Metrics enable performance engineers to evaluate HPC systems and software by focussing on key properties, facilitating comparisons across platforms, and measuring the impact of code modifications. The Energy-Delay Product metrics (EDP, Equation (1)) have been commonly used in the hardware community; however, Roberts et al. show that these metrics are not suitable for measuring software performance [12], and propose two new dimensionless metrics, the Energy-Delay Summation (EDS, Equation (2)) and the Energy-Delay Distance (EDD, Equation (3)):

$$M(\theta) = E_\theta t_\theta^n \quad (1)$$

$$M(\theta) = \alpha E_\theta + \beta t_\theta \quad (2)$$

$$M(\theta) = \sqrt{(\alpha E_\theta)^2 + (\beta t_\theta)^2} \quad (3)$$

where $M(\theta)$ is the calculated metric of application θ, E_θ is the energy consumption, and t_θ is the run time of the application θ. Both metrics can be parametrised with α and β to emphasise the importance of either energy or runtime.

Conversely, Shin et al. propose a weighted EDP metric to communicate data centre constraints and energy efficiency goals, allowing performance engineers to make policy-aware optimisations at application-level [17]. The proposed weighted EDP extends the traditional EDP metric by placing weights on both energy and time:

$$M(\theta) = E_\theta^\alpha t_\theta^\beta \quad (4)$$

where the values α and β are set according to HPC or data centre priorities. Shin et al. suggest that α can be set to 1.0 and β to 2.0 if the data centre prioritises performance. In power-constrained environments, α can be set to 2.0 and β to 0.2, favouring energy reduction. Under high electricity prices, α can be set to 1.3 and β to 0.7 to balance energy consumption and performance degradation.

The work presented in this paper is metric agnostic, but our results are presented using the EDS metric, parametrised to approximately reflect the dollar cost of execution.

2.2 Energy-Aware Performance Engineering

As HPC centres become power-constrained and over-provisioned, performance engineers must now consider both runtime and energy consumption in their optimisation efforts. A potential approach to

understanding application performance on specific hardware involves using analytical, white-box performance models [19]. These models simplify the complexities of hardware, software, and their interactions. The Roofline model is a performance visualisation model that offers insights with respect to the arithmetic intensity of the kernel and the available bandwidth [3, 21]. Performance modelling techniques enable exploration of large hardware and software design spaces.

The Power-Optimised Software Envelope (POSE) is a mathematical and visual modelling tool that captures the trade-off between software power consumption and runtime [12, 13]. POSE is metric agnostic and works with all members of the $E^m t^n$ group with $m > 0$ and $n \geq 0$, EDD and EDS metrics, and indeed any metric that increases with runtime and energy consumption. POSE enables engineers to make decisions that can influence the energy consumption of their codes.

Shin et al. propose an energy-aware performance engineering workflow that is a systematic and iterative approach to optimise applications using traditional profiling tools along with multi-objective optimisation tools and models [17]. POSE models and the tool presented in this paper can be used as part of the systematic and iterative approach, as POSE models complement the traditional profiling tools by providing insights that guide performance engineers in answering whether there is any scope for energy optimisation and in finding optimal trade-offs points in the large energy-runtime design space.

In this paper, we extend the original POSE model by further constraining the energy–time design space through the incorporation of both code characteristics (arithmetic intensity) and hardware characteristics (memory hierarchy and instruction set architecture).

3 The Cache-Aware POSE Model

The POSE model, introduced by Roberts et al. [13, 14], is built around the concept of a *feasible performance envelope*. It is constructed by plotting P_{min} and P_{max} as shown in Figure 1. P_{min} and P_{max} represent the minimum and maximum power that the target platform can draw during normal operation. The energy and runtime costs of running a code θ under similar conditions must be within the feasible region. P_{min} and P_{max} are empirically determined using benchmarks and are defined using (S,A,C) tuples, where S denotes the P-state, A the activity factor, and C the number of active cores [14]. The benchmarks must capture the range of values these parameters can assume for a given code θ. This concept is captured by Equation 5.

$$\begin{aligned} P_{\text{min}} &= (S_{\text{min}}, A_{\text{min}}, C_{\text{min}} \mid \theta) \\ P_{\text{max}} &= (S_{\text{max}}, A_{\text{max}}, C_{\text{max}} \mid \theta) \end{aligned} \tag{5}$$

For a given code θ three bounds can be defined using a metric of choice. The *Optimisation Bound* links all points with the same metric value as θ; all optimised version of θ should fall below the optimisation bound. The *Contribution Bound* subdivides this region into runtime and power optimisations. The bound links all points for which power and runtime contribute to the metric in the same ratio as the unoptimised θ; optimised versions of θ that lie to the left of the bound stem primarily from a reduction in runtime, while optimised versions of θ that lie to the right of the bound stem primarily from a reduction in power. Finally, the *Optimisation Limit* links all points with the same metric value as point C (where the contribution bound meets P_{min}, which corresponds to the best possible outcome from power optimisation); any optimised version of θ below this limit will outperform any possible power optimisation.

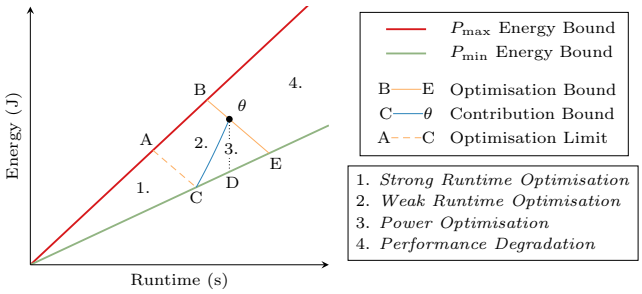


Figure 1: The POSE model with EDS metric [13]

These bounds partition the feasible performance envelope of Figure 1 into four regions with differing performance characteristics. Area 1 comprises runtime optimisations that strongly dominate the best-case power-optimised version in terms of metric (strong runtime optimisation). Area 2 includes runtime optimisations that outperform the baseline code θ in terms of metric, but may yet be outperformed by certain power-optimised variants of θ (weak runtime optimisation). Area 3 consists of optimisations where improvements in metric are driven primarily by reducing power consumption (power optimisation). Area 4 represents versions that exhibit performance loss compared to θ (performance degradation).

Roberts et al. provide derivations for these bounds using EDP, EDS, and EDD [13]; in this paper, we use EDS as this metric can be parametrised to represent the financial cost of parallel computation.

3.1 Model Extension

The POSE model captures the characteristics of the code, such as runtime, total energy consumed, and average power. It also captures characteristics of the hardware, such as maximum power and minimum power drawn by the system, and encodes the information as P_{min} and P_{max} as shown in Figure 1. In the original formulation, P_{min} and P_{max} were determined empirically based on the minimum possibly power draw without allowing the CPU to sleep, and the maximum possible power drawn when running the FIRESTARTER [15] application. As stated by Roberts at al. "POSE works best when the power bounds are as tight as possible" [13]. In this paper, we improve upon the POSE model, proposing a tighter and more realistic feasible performance envelope, that is further constrained by incorporating the characteristics of the software code and hardware, such as the memory hierarchy of the system and the instruction set architecture supported by the processor. This improves the applicability of the POSE model to investigating the scope for energy-aware optimisation for scientific applications.

Our **Cache-Aware POSE Model** (CA-POSE) refines the P_{min} and P_{max} bounds, using values empirically captured, but further constrained by considering the inherent arithmetic intensity of an application, and its performance. This places an application within a specific region of the Cache-Aware Roofline Model, and thus allows us to measure tighter power bounds that are related to attainable

application performance, and better inform performance engineers on possible pathways to more energy-efficient execution of their applications.

3.2 CA-POSE Construction

To generate a CA-POSE model, we follow the same process as for the POSE model, but with custom benchmarks for calculating the P_{min} and P_{max} values, based on a specific arithmetic intensity.

Maximum power consumption occurs when both the compute units and the memory load/store units are heavily utilised. Under such conditions, the processor simultaneously sustains high arithmetic throughput and high memory activity, increasing dynamic power across multiple micro-architectural components. Consequently, peak power consumption is attained by micro-benchmarks that drive high activity in both execution units and memory access units. For building a CA-POSE model, we leverage the customised mixed benchmarks feature supported by the Cache-Aware Roofline Model (CARM) tool [10]. The CARM tool provides python scripts to automatically generate assembly micro-benchmarks tailored to specific user-selected options and the targeted underlying hardware. Users can control the ratio of floating point operations to load/store memory accesses targeting a specific memory level. We generate micro-benchmarks at each memory level L1, L2, L3, and DRAM with arithmetic intensity as close as possible to the application in question, and then we profile the micro-benchmarks to collect runtime and energy consumption to construct the P_{max} line of our CA-POSE model.

For the P_{min} line of the CA-POSE model, our provided script modifies the L1 cache mixed micro-benchmarks to introduce a dependency chain, preventing micro-architectural optimisations such as register renaming, out-of-order and speculative execution, and to enforce the use of a single register and repeated loads and stores to the same memory location in order to reduce power as much as possible. We can further decrease power consumption by introducing NOP (no-operation) instructions into the modified micro-benchmarks. When the number of NOP instructions exceeds that of load/store and floating-point instructions, the P_{min} of the CA-POSE model becomes the same as the P_{min} of the original POSE model.

The iterative energy-aware performance engineering workflow involving the POSE/CA-POSE model is shown in Figure 2.

In this paper we focus our study on the OpenMP-enabled C++ implementations of the NAS Parallel Benchmarks [8] running on an EPYC 7313P 64-core processor. The NAS Parallel Benchmark suite contains several benchmarks derived from computational fluid dynamics (CFD) applications.

The arithmetic intensity of each application can be obtained by profiling relevant performance counters to capture FLOP/s and memory traffic, or by using tools such as the Intel Software Development Emulator (Intel SDE)[1]. We collect relevant metrics such as the number of floating point operations performed by the kernel and memory loads/stores to calculate the arithmetic intensity of the kernels. We parse the output files using python scripts provided as part of the CARM Tool [10] to obtain the arithmetic intensity and other

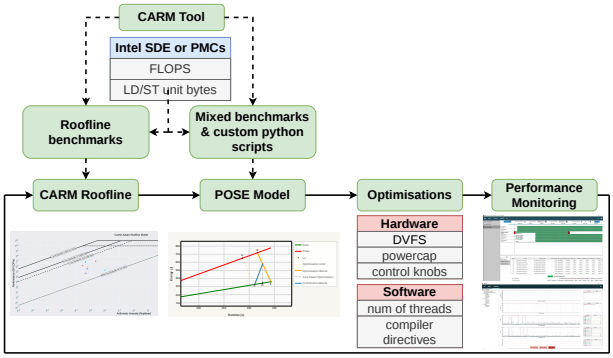


Figure 2: Workflow using POSE models for Energy-Aware Performance Engineering

metrics in readable format. We also construct a CARM model [3] for our target CPU to analyse the performance of the kernels as a traditional application optimisation approach (see Figure 3).

CARM and the Original Roofline Model (ORM) differ in their approach to memory traffic, which in turn leads to different performance characterisation capabilities and insights [21]. CARM observes the memory traffic from the core perspective, i.e., it captures all the loads and store operations, and complete hierarchy in a single plot leading to a single arithmetic intensity value for all the memory levels [10]. In contrast, ORM measures the traffic between two subsequent memory levels in the memory hierarchy, leading to a different operational intensity for each memory level. Applications and kernels are represented as single points in the CARM plot. Depending on the location of the point relative to the modelled rooflines, an application is characterised as memory-bound, mixed-bound, or compute-bound, which guides performance engineers in exploring optimisations tailored to the application's characteristics [9]. The POSE model can complement CARM in iterative energy-aware application optimisation.

We compile the OpenMP implementation of the NAS Parallel Benchmarks using the AMD Optimising C/C++ and Fortran Compiler (AOCC) version 5.0 with OpenMP support enabled[2]. AOCC uses Clang as its frontend and is optimised for high-performance computing on AMD processors, particularly the Zen architecture. All experiments are executed with 64 OpenMP threads, specified via `OMP_NUM_THREADS=64`. To ensure deterministic thread affinity and minimise scheduling variability, we configure the OpenMP runtime with `OMP_PLACES=cores` and `OMP_PROC_BIND=close`, binding threads to distinct physical cores and placing threads as close as possible to their parent thread. This configuration provides consistent core-level placement across runs and improves the reproducibility of performance and power measurements.

For energy measurements we rely on Running Average Power Limit (RAPL) counters, a low-level interface designed by Intel that provides energy consumption of a CPU without the need for additional hardware. AMD began implementing the RAPL interface in its Zen CPU Architecture. Energy measurements can be obtained from the CPU by reading the low-level RAPL MSR registers directly or by using high-level interfaces provided by operating

[1] https://www.intel.com/content/www/us/en/developer/articles/tool/software-development-emulator.html

[2] https://www.amd.com/en/developer/aocc.html

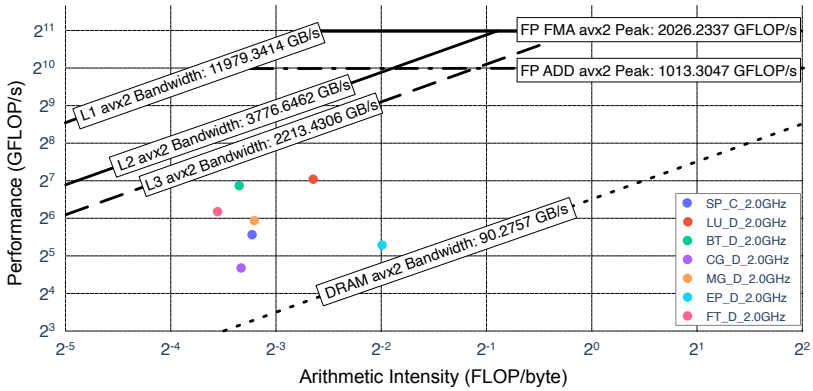


Figure 3: The Cache-Aware Roofline Model for our applications

Table 1: Arithmetic Intensity obtained from Intel SDE

Benchmark	Class	Arithmetic Intensity
Embarrassingly Parallel (EP)	D	0.2512
Multi Grid (MG)	D	0.1083
Conjugate Gradient (CG)	D	0.0993
Fast Fourier Transform (FT)	D	0.0850
Block Tri-diagonal solver (BT)	D	0.0980
Scalar Penta-diagonal solver (SP)	C	0.1067
Lower-Upper Gauss-Seidel (LU)	D	0.1597

systems. Linux provides the Power Capping Framework (powercap) and the Performance Counters Subsystem (perf-events) [11]. Raffin et al. found significant overhead differences on an idle server and in microbenchmarks; they recommend using the perf-events interface whenever possible and introduced a minimal RAPL-based measurement tool [11]. We use the tool introduced in the paper to make energy measurements of the benchmarks in our experiments. The AMD EPYC System Management Interface library (E-SMI) can be used to control and monitor the power consumption of AMD EPYC processors[3]. We use the E-SMI interface to set powercap in one of our case studies.

Table 1 presents the captured arithmetic intensity of our applications, measured using the Intel SDE tool on the EPYC 7313P processor. The arithmetic intensity of the four kernels (EP, MG, CG, FT) and three pseudo applications (BT, SP, LU) ranges from 0.0850 to 0.2512. Figure 3 shows the four kernels and three pseudo applications on a CARM plot, showing that all codes lie in between the L3 and DRAM bandwidth lines, with BT closest to the L3 bandwidth line.

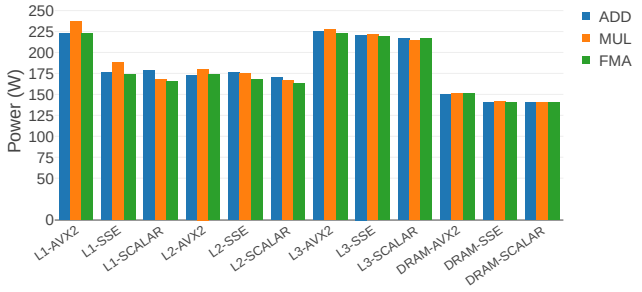
Table 1 shows that the MG, CG, BT, and SP benchmarks all have an arithmetic intensity close to 0.1. Figure 4(a) plots the power consumption for 36 generated micro-benchmarks, running at 64 threads, using an arithmetic intensity of 0.1 across all memory levels, and using every combination of ISA (scalar, SSE, AVX2) and instruction type (ADD, MUL, FMA). Our results show that for a fixed memory level, wider SIMD execution results in higher power consumption, with AVX2 consistently consuming more power than SSE and scalar execution. Variations among ADD, MUL, and FMA instructions are comparatively small, indicating that execution width and memory locality dominate power behaviour rather than the type of arithmetic operations at arithmetic intensity of 0.1. The maximum power consumption is observed for AVX2 MUL operations at the L1 cache level, with a power consumption of 237.3 W, followed by AVX2 MUL operations at L3 cache level, with a power consumption of 227.7 W.

The arithmetic intensity of FT (0.085) is close to an FP operations to load/store bytes ratio of 1/12. Our analysis shows that the power consumption profile of mixed micro-benchmarks at arithmetic intensity of 0.083 behaves similarly to the benchmark with an arithmetic intensity of 0.1, and so we omit the figure here. Similarly to our previous data, we observe that maximum power is obtained for AVX2 MUL operations at the L1 cache level, with a power consumption of 241.44 W, followed by ADD operations at the same L1 cache level, with a power consumption of 239.65 W, whereas FMA operations consume 222.43 W.

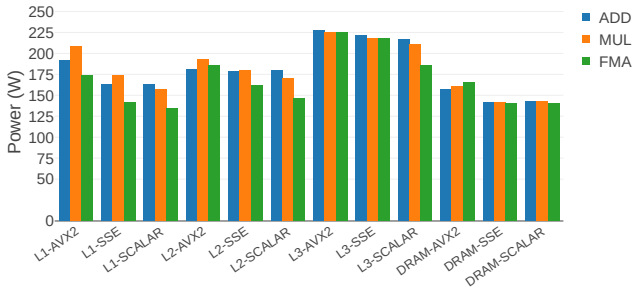
For finding $P_{\max}$ for the LU benchmark with an arithmetic intensity of 0.1597, we generate mixed micro-benchmarks with arithmetic intensity of 0.16. Figure 4(b) shows that the power consumption of L3 cache level micro-benchmarks dominate compared to other memory levels. Among the evaluated configurations, the AVX2 ADD operation exhibits the highest power consumption at 227.46 W slightly higher than the MUL and FMA versions which consume 225.51 W and 225.58 W respectively. At the AVX2 L1 cache level, variations in power consumption among the ADD, MUL, and FMA instructions become visible, with the MUL version consuming the highest power at 207.87 W, whereas the FMA version consumes less power at 174.22 W.

EP has the highest arithmetic intensity in our benchmark set, at 0.25. The power consumption of mixed micro-benchmarks with an arithmetic intensity of 0.25 is shown in Figure 4(c). At the L3 cache level, the AVX2 and SSE versions dominate other configurations in terms of power consumption, with the SSE MUL operation version consuming slightly higher power (227.58 W) than the AVX2 ADD, MUL, and FMA versions, which draw 226.03 W, 225.04 W, and 223.53 W, respectively. A shift in power consumption can be observed among the ADD, MUL, and FMA instructions, with FMA consuming higher power than ADD and MUL; however, at arithmetic intensities of 0.083, 0.1, and 0.16, FMA exhibits lower power consumption.

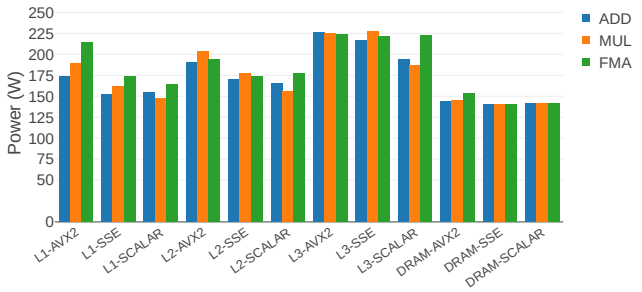
[3]https://www.amd.com/en/developer/e-sms/e-smi-in-band-library.html



(a) Arithmetic Intensity = 0.1



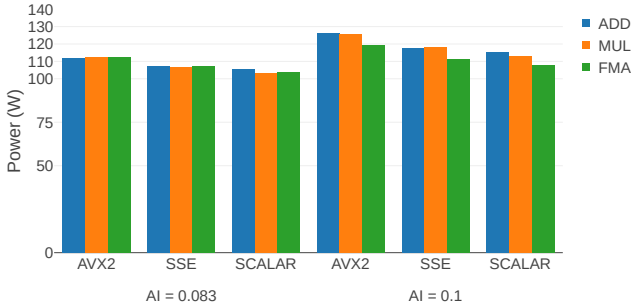
(b) Arithmetic Intensity = 0.16



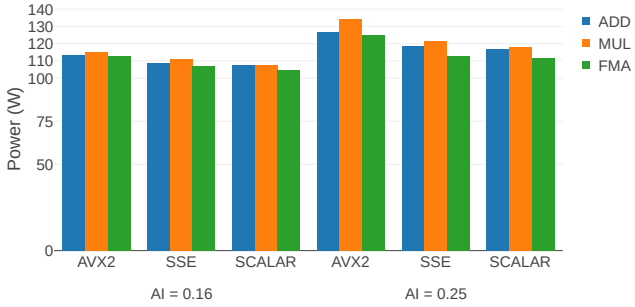
(c) Arithmetic Intensity = 0.25

Figure 4: Power comparison of mixed benchmarks at Arithmetic Intensity 0.1, 0.16 and 0.25

From the power profiles of mixed micro-benchmarks at arithmetic intensities of 0.083, 0.1, 0.16, and 0.25, we observe that micro-benchmarks at DRAM memory level consume significantly less power. The L3 cache level micro-benchmarks exhibit significantly high power throughout the experiments, but scalar versions seem to exhibit lower power consumption as we increase arithmetic intensity. The AVX2 versions at any memory level consume more power than the SSE and scalar versions. FMA instructions consume more power compared to ADD and MUL instructions at higher arithmetic intensity benchmarks, but consume less power in lower arithmetic intensity benchmarks.



(a) Arithmetic Intensity = 0.083 and 0.1



(b) Arithmetic Intensity = 0.16 and 0.25

Figure 5: Power comparison of $P_{\min}$ version of L1 mixed benchmarks at Arithmetic Intensity 0.83, 0.1, 0.16 and 0.25

Figure 5 shows the power consumption behaviour of the memory-reuse modified micro-benchmark at arithmetic intensities of 0.083, 0.1, 0.16, and 0.25, in order to obtain a bound for $P_{\min}$. The scalar version of the micro-benchmarks draws less power at all evaluated arithmetic intensities (0.083, 0.1, 0.16, and 0.25), consuming 103.08 W, 107.92 W, 104.87 W, and 111.65 W, respectively.

3.3 EDS Metric Tuning

The results presented above allow us to select the $P_{\min}$ and $P_{\max}$ bounds that are most relevant to our application of interest. All other bounds in the POSE and CA-POSE models are specified based on the chosen multi-objective optimisation metric.

In this paper, we present our results using the Energy-Delay Summation (Equation (2)) metric, parameterised to approximately correspond to the financial cost of application execution on our local HPC system.

We calculate α as the energy cost per Joule, i.e., price per kWh divided by Joules per kWh (3,600,000). β captures the fixed costs of the HPC system over its lifetime, including the hardware purchase price, the administration costs, software and support costs, and any rack/space rental costs. For the work presented in this paper, we use £0.15 per kWh for the cost of electricity, and we approximate the fixed costs for our system at £2.5 million over a 5 year expected life. The system contains both CPU and GPU nodes and has a combined peak performance (i.e., R_{peak}) around 1.0 PFLOP/s. A single CPU

Table 2: P_{min} and P_{max} values for Cache-Aware POSE model

Benchmark	P_{max} (W)	P_{min} (W)
EP	242.65	111.66
MG	237.31	107.93
CG	237.31	107.93
FT	241.45	103.09
BT	237.31	107.93
SP	237.31	107.93
LU	227.46	104.87

node within the system is capable of approximately 3500 GFLOP/s, and thus represents 0.35% of its total performance. We therefore choose the following values for α and β to represent the dollar per second cost of using a single CPU node within our system:

$$\begin{aligned}\alpha &= 0.15 \text{ (Price per kWh)} \div 3{,}600{,}600 \text{ (Joules per kWh)} \\ &= 0.0000000416\dot{}\end{aligned}$$

$$\begin{aligned}\beta &= 2.5 \times 10^6 \text{ (Total Fixed Costs)} \div 5 \text{ (Years)} \\ &\div 365 \text{ (Days)} \div \ 24 \text{ (Hours)} \div 60 \text{ (Minutes)} \\ &\div 60 \text{ (Seconds)} \times 0.0035 \text{ (Portion of machine)} \\ &= 0.000056\end{aligned}$$

Crucially, any improvement in the metric $M(\theta)$ through either power reduction or runtime reduction represents a real-terms financial saving.

3.4 Comparison of Models

We investigate the NAS Parallel Benchmarks using POSE and our CA-POSE model and compare the insights provided by the models. For all experiments, the CPU frequency governor was set to "performance", and the operating frequency was fixed at 2.00 GHz using `cpupower`, ensuring a fixed operating frequency. We configure OpenMP with `OMP_PLACES=cores` and `OMP_PROC_BIND=close`, binding threads to distinct physical cores and placing threads as close as possible to their parent thread. All applications are executed in parallel on 64 cores. Therefore, S and C for calculating $P_{\max}$ and $P_{\min}$ are set to 2.00 GHz and 64 respectively in Equation (5). The $P_{\max}$ value for the POSE model can be obtained by running FIRESTARTER [15], a stress test that generates near peak power consumption. The $P_{\min}$ value is obtained by executing an assembly micro-benchmark composed primarily of NOP instructions, with a jump instruction inserted to maintain control flow (see Roberts et al. [13]). The $P_{\min}$ and $P_{\max}$ for constructing the feasible performance envelope are given by Equation (6). The CA-POSE $P_{\min}$ and $P_{\max}$ of all benchmarks are shown in Table 2.

$$\begin{aligned}P_{\min} &= (2.00 \text{ GHz}, A_{\min}, 64) = 94.39W \\ P_{\max} &= (2.00 \text{ GHz}, A_{\max}, 64) = 242.78W\end{aligned} \tag{6}$$

The energy and runtime costs associated with running the NAS Parallel Benchmarks are presented in Table 3.

Figure 6 shows generated POSE and CA-POSE models for the EP and CG benchmarks. The insights offered by both POSE and CA-POSE are shown in Table 4. The insights can be represented in both absolute and relative terms; due to space constraints, we present only the absolute insights. The visualisation suite automatically generates insights in both absolute and relative terms.

Table 3: Code Metrics for S = 2.0 GHz, C = 64

Code	Runtime (s)	Energy (J)	Power (W)	EDS
EP	119.13	13619.34	114.32	0.00724
MG	51.23	8428.27	164.52	0.00320
CG	142.53	30259.00	212.30	0.00924
FT	122.77	22569.35	183.83	0.00782
BT	489.69	82758.46	169.00	0.03087
SP	29.06	4930.32	169.65	0.00183
LU	326.51	58165.23	178.14	0.02071

Table 4: POSE model Summaries

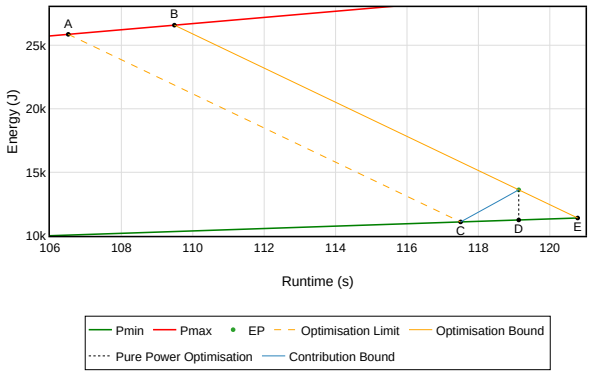
Embarrassingly Parallel	**POSE**	**CA-POSE**
Best Case Energy Saved by Reducing Power Consumption	2374.56 J	317.837 J
Worst Case Slowdown as a Result of Power Optimisation	1.65s	0.22s
Best Case Improvement in EDS from Power Optimisation	2.71%	0.37%
Minimum Speed Up Guaranteed to Outperform θ	9.64s	9.64s
Speed Up Guaranteed to Dominate Power Optimisation	12.62s	10.04s
Conjugate Gradient		
Best Case Energy Saved by Reducing Power Consumption	16805.52J	14876.38J
Worst Case Slowdown as a Result of Power Optimisation	11.68s	10.25s
Best Case Improvement in EDS from Power Optimisation	14.58%	12.96%
Minimum Speed Up Guaranteed to Outperform θ	2.74s	2.25s
Speed Up Guaranteed to Dominate Power Optimisation	23.12s	20.44s

Table 4 shows the differences in the insights produced by the two models, highlighting that CA-POSE provides a much more pessimistic view of the potential of power optimisations, which reflects the tightened feasible performance envelope region in the extended model. As shown in Figure 6(b), the energy and runtime costs of the EP code lie extremely close to the P_{min} energy bound in the Cache-Aware POSE model. The extended model shows a best case energy saved by pure power optimisation of 317.84 J which is 86.61% lower than the POSE model. The best-case improvement in the EDS metric due to power optimisation decreases from 2.71% to 0.37%, indicating that there is even less potential for power optimisation in the EP kernel. The CG kernel lies closer to the P_{max} energy bound as shown in Figure 6(d) and has an upper bound of 12.96% improvement in the EDS metric due to power optimisation, and it is the highest among the benchmarks we evaluated.

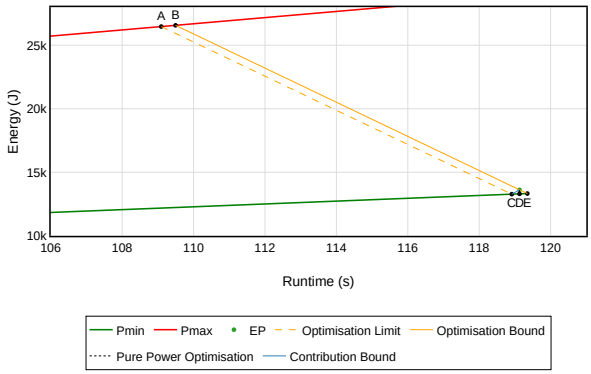
4 The POSE Visualisation Suite

To facilitate interactive exploration of the POSE and CA-POSE models, we introduce a web-based visualisation tool that enables users to analyse runtime–energy trade-offs. The interface is organised into multiple vertically stacked panels to accommodate the full analysis workflow. The upper section of the interface provides a set of global controls that allow users to configure the analysis. These include dataset selection, metric parameters α, β, m, and n, and the choice of metric EDP, EDS, or EDD. Below the parameter controls, the tool presents the POSE model visualisation panel. This panel allows users to select the minimum and maximum power consumption lines of a system ($P_{\min}$ and $P_{\max}$) and to generate an interactive runtime–energy plots representing the corresponding feasible performance envelope.

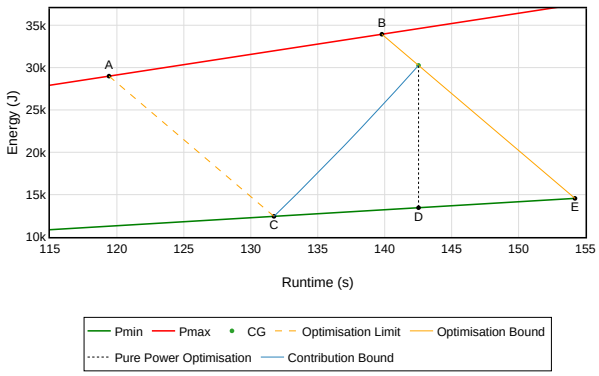
Below the POSE visualisation panel, the CA-POSE visualisation panel provides an analogous visualisation for cache-aware configurations. This panel follows the same layout and interaction model as the POSE panel, enabling direct comparison between the models. In addition to graphical output, each model panel includes dedicated controls for computing point metrics and generating insights. These features enable users to extract quantitative information –



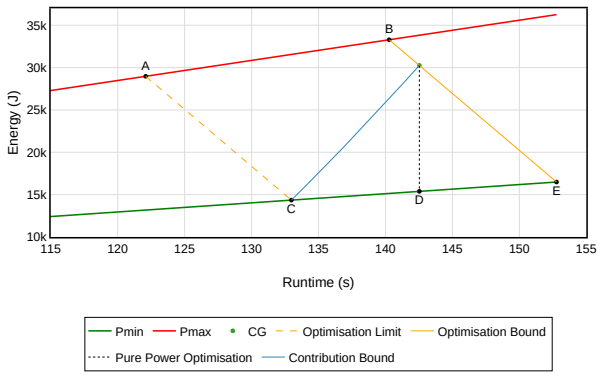
(a) POSE model for EP



(b) CA-POSE model for EP



(c) POSE model for CG



(d) CA-POSE model for CG

Figure 6: Generated POSE and CA-POSE models using our web-based POSE Visualisation Suite

such as bounds on optimisation potential and required speedups – directly from the visualised envelopes. All insights are generated automatically. The POSE visualisation panels support two modes of operation. The first is a normal mode, shown in Figure 6, which is used to construct the model for a selected version of the code θ of interest. The second is a batch mode, which enables the identification of code variants of θ whose power-optimised versions can outperform an optimised version of θ. Batch mode is used to study the combined effects of power capping and DVFS-based frequency scaling, as in the following case studies.

5 Case Studies

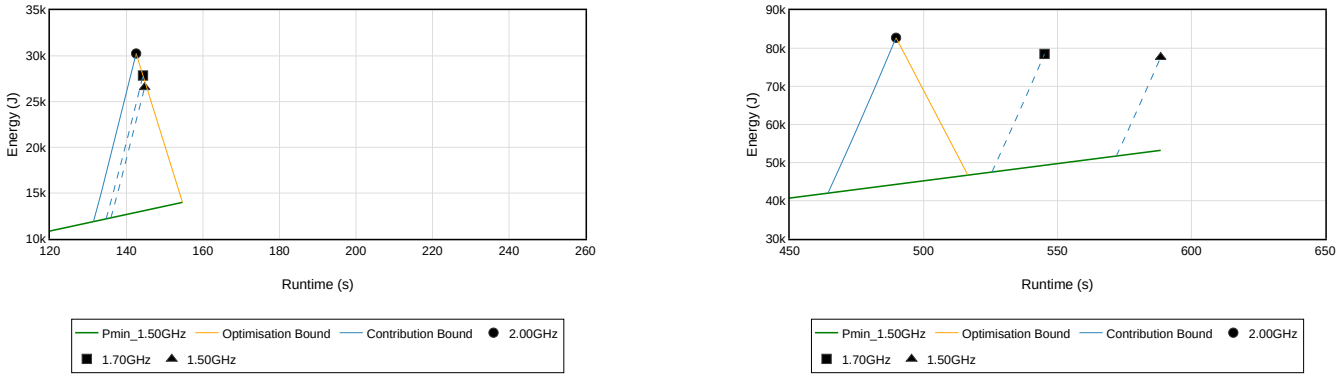
In this section, we apply our improved model and our visualisation toolkit to investigate the effects of frequency scaling and power capping on the Conjugate Gradient (CG) kernel and the Block Tridiagonal (BT) pseudo application.

DVFS is a firmware-level technique to dynamically change the voltage and/or frequency of processor cores to reduce overall system power consumption [7]. Three P-states are available on the AMD EPYC 7313P processor, operating at frequencies of 2.00 GHz, 1.70 GHz, and 1.50 GHz. In general, higher operating frequencies improve performance, but do not always result in energy-efficient execution and depend on the characteristics of the workload [19]. Memory-bound applications can be executed at lower frequencies with negligible impact on performance, whereas compute intensive applications may benefit from operating at higher frequencies [2]. Although the BT benchmark has an arithmetic intensity similar to CG (0.0980 vs. 0.0993), BT achieves significantly higher performance, reaching 116.98 GFLOP/s compared to 25.61 GFLOP/s for CG. The CG kernel places significant stress on data communication, memory locality, and cache behaviour [8].

Figure 7 shows a DVFS analysis for CG and BT, with $P_{\min}$ set according to Equation (7).

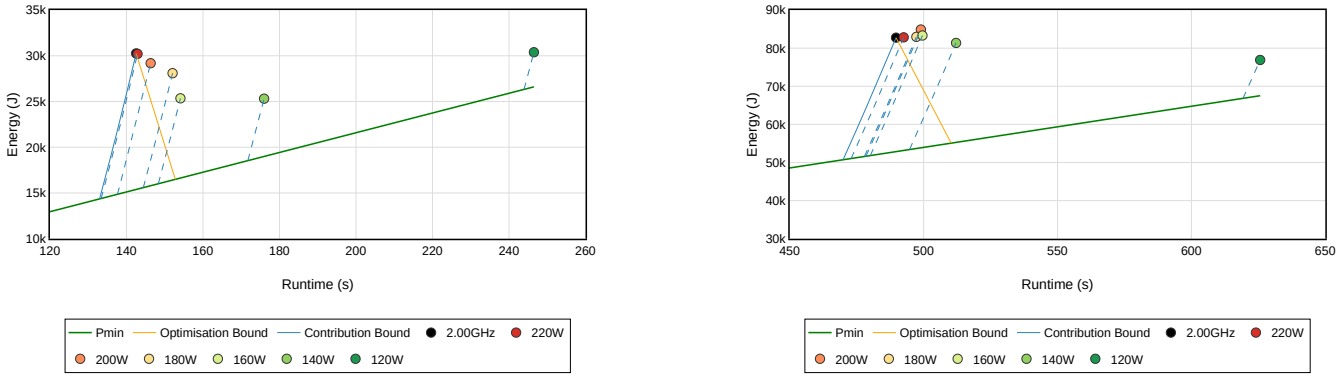
$$P_{\min} = (1.50\ \text{GHz}, A_{\min}, 64) = 90.42W \tag{7}$$

We choose 2.00 GHz as our baseline as that is the highest frequency among the available P-states. As seen in Figure 7(a), the 1.50 GHz version of the CG kernel exhibits a lower EDS metric than the baseline version at 2.00 GHz, indicating that an energy-to-solution strategy is optimal. Executing the CG kernel at 1.50 GHz results in a 12.21% reduction in energy consumption at the cost of a 1.56% performance degradation. In contrast, the baseline version of the BT benchmark at 2.00 GHz has a lower EDS metric than other P-states, suggesting race-to-halt is a better strategy. However, alternative power optimisations at other P-states may achieve optimal performance [14]. POSE models can be used to identify if



(a) CG **(b) BT**

Figure 7: DVFS-based frequency scaling analysis using CA-POSE model



(a) CG **(b) BT**

Figure 8: Power capping analysis using CA-POSE model

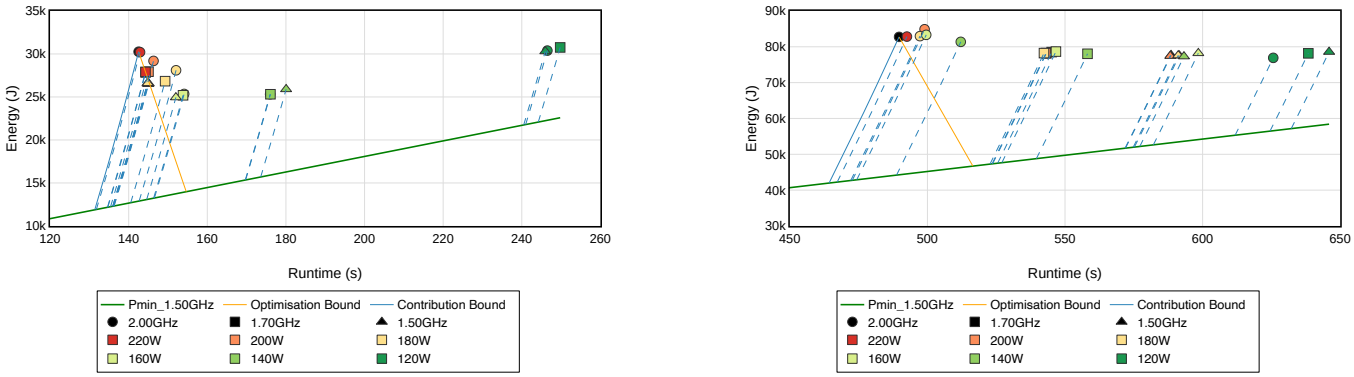
any power-optimised code running at lower frequency P-state can outperform an unoptimised code at a higher frequency. As shown in Figure 7(b), the contribution bound of the lower P-states do not intersect with the optimisation bound of the baseline version at 2.00 GHz, indicating that no power-optimised versions of the BT benchmark operating at lower-frequency P-states achieve a lower EDS metric than the baseline.

The AMD E-SMI library can be used to apply power capping on AMD EPYC processors[3]. The maximum power cap that can be set on the EPYC 7313P processor is 240 W, but we were able to achieve around 247 W when running FIRESTARTER [15]. Our power capping study is carried out by applying power limits in decrements of 20 W, starting at 220 W and ending at 120 W. The operating frequency is fixed at 2.00 GHz, with the uncapped 2.00 GHz configuration used as the baseline. The P_{min} energy bound is set to 107.93 W according to Table 2.

As seen in Figure 8, the EDS metric is lower (i.e. closer to the origin) for the uncapped baseline code at 2.00 GHz for both CG and BT benchmarks. The baseline CG and BT benchmarks exhibit average power consumptions of 212.30 W and 169 W, respectively. Moreover, all configurations for CG except the 120 W version in Figure 8(a) reduce energy consumption at the cost of longer runtime. This is not the case for BT, where energy consumption does not decrease significantly, while runtime increases. The CA-POSE model suggests that CG may have power-optimised configurations at power caps higher than 140 W that can outperform the baseline, while for BT, the baseline may be outperformed by power-optimised configurations at power caps higher than 120 W.

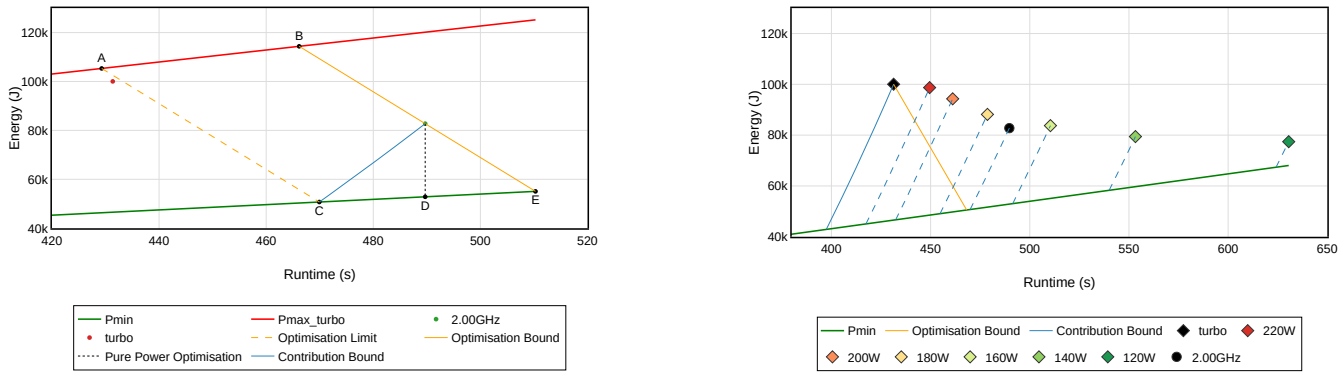
We also explore the combined effects of DVFS-based frequency scaling and power capping on CG and BT benchmarks as shown in Figure 9. For the BT benchmark, the uncapped baseline configuration operating at 2.00 GHz yields the lowest EDS metric among all combinations of power caps and P-states evaluated in this study. No power-optimised configuration operating at a frequency below 2.00 GHz can outperform the baseline in terms of the EDS metric. Additionally, no power-optimised configuration operating at 2.00 GHz under a power cap lower than 140 W can outperform the baseline EDS metric.

For the CG benchmark, the configuration operating at 1.50 GHz without a power cap outperforms all evaluated combinations of power caps and P-states in terms of the EDS metric. No power-optimised version below a power cap of 160 W can outperform the



(a) CG **(b) BT**

Figure 9: Combined effects of power capping and frequency scaling



(a) Frequency scaling **(b) Combined effects of power capping and frequency scaling**

Figure 10: Analysis of Turbo Boost effect on BT benchmark

baseline. Interestingly, the configuration operating at 1.50 GHz under a power cap of 180 W outperforms the baseline, while incurring a performance degradation of 0.1 s and exhibiting no significant change in energy consumption compared to the optimal uncapped configuration operating at 1.50 GHz. By setting a power cap of 180 W at the expense of a negligible increase to runtime, users can contribute to overall throughput in over-provisioned HPC centres.

Turbo mode allows the processor to operate above its base frequency when sufficient power and thermal headroom are available. As discussed above, the BT kernel exhibits improved performance at higher-frequency P-states. We therefore further investigate whether enabling turbo mode is beneficial for the BT kernel and whether any power-optimised configuration operating at 2.00 GHz can outperform the BT kernel executed with turbo mode enabled. It is evident from Figure 10(a) that enabling turbo boost outperforms the benchmark operating at 2.00 GHz, as well as all power-optimised configurations at that frequency. By further applying power capping while turbo boost is enabled, we show in Figure 10(b) that no power-optimised configuration operating under a power cap below 180 W can outperform uncapped, unoptimised code running with turbo boost enabled.

6 Conclusion

As we move beyond the Exascale era, HPC centres face growing challenges in achieving energy efficiency, as steady improvements in performance per Watt are slowing due to the fundamental physical limits of silicon technology [16]. As sustainability becomes a growing concern, HPC centres are exploring strategies to motivate users to consider energy optimisations in their applications by providing incentives [18] and by promoting application-level energy optimisation through normalisation of energy-aware tools in performance engineering workflows [17].

In this work, we introduced the Cache-Aware Power-Optimised Software Envelope, an extension of the Power-Optimised Software Envelope Model that incorporates application arithmetic intensity to further constrain the feasible energy-runtime design space. These refined bounds enable more accurate reasoning about energy-aware optimisation opportunities and provide insights to guide performance engineers to make energy-aware decisions. Our analysis of the Embarrassingly Parallel benchmark, from the NAS Parallel Benchmarks, shows that the maximum improvement in the EDS metric due to power optimisation is 0.37%, indicating a very limited scope for power optimisation; conversely the Conjugate Gradient benchmark could achieve a 12.96% improvement in the same metric.

To support users in achieving their sustainability-driven objectives in HPC, we developed a visualisation framework for the POSE and CA-POSE models. Our web-based tool allows users to plot the respective models and extract insights from datasets that include performance and power information.

Finally, we demonstrated both our improved CA-POSE model and our visualisation framework to examine the effects of DVFS-based frequency scaling, power capping, and their combined impact on the Conjugate Gradient and Block Tri-diagonal solver benchmarks. Our analysis reveals that the BT kernel is sensitive to operating frequency. The BT kernel exhibits a better EDS metric when turbo boost is enabled and can be further power-capped to 180 W to explore power-optimised configurations that outperform the uncapped configuration. In contrast, the CG kernel achieves a better EDS metric at lower operating frequencies.

Through the use of models and visualisation/analysis tools such as POSE and CA-POSE, performance engineers can make informed decisions about their applications to improve the performance in an energy-aware manner. All data and scripts used in this work, and our visualisation framework can be accessed at github.com/pose-model/cache-aware-pose.

6.1 Future Work

In this paper we have presented an extended POSE model that improves the applicability of its key insights. To date, the POSE model (and our extended model) have only been demonstrated on x86_64-based architectures, and these platforms have shown reasonably limited scope for pure *power* optimisations. Accelerator platforms, and in particular reconfigurable architectures, may show more promise for optimisation in this space, and so our intention is to further develop the models to account for GPU, FPGA, and CGRA architectures.

The study presented in this paper evaluates mini-applications, and we recognise that the model's predictive power is most potent when applied to single kernels. Consequently, we plan to extend our tools to full production applications, with runtime and power consumption measured on a kernel-by-kernel basis. This granular approach will allow the model to specifically identify which computational phases are most applicable to power or runtime-led optimisation.

Acknowledgements

The Viking cluster was used during this project, which is a high performance compute facility provided by the University of York. We are grateful for computational support from the University of York, IT Services and the Research IT team.

References

[1] Abdessalam Benhari, Denis Trystram, Fanny Dufossé, Yves Denneulin, and Frédéric Desprez. 2024. Green HPC: An analysis of the domain based on Top500. https://arxiv.org/abs/2403.17466

[2] Sridutt Bhalachandra, Allan Porterfield, Stephen L. Olivier, Jan F. Prins, and Robert J. Fowler. 2017. Improving Energy Efficiency in Memory-constrained Applications Using Core-specific Power Control. In *Proceedings of the 5th International Workshop on Energy Efficient Supercomputing (E2SC'17)*. Article 6.

[3] Aleksandar Ilic, Frederico Pratas, and Leonel Sousa. 2013. Cache-aware Roofline model: Upgrading the loft. *IEEE Computer Architecture Letters* 13, 1 (April 2013).

[4] Ahmad Maroof Karimi, Matthias Maiterth, Woong Shin, Naw Safrin Sattar, Hao Lu, and Feiyi Wang. 2024. Exploring the Frontiers of Energy Efficiency using Power Management at System Scale. In *Proceedings of the Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC-W'24)*.

[5] Yuetsu Kodama, Tetsuya Odajima, Eishi Arima, and Mitsuhisa Sato. 2020. Evaluation of Power Management Control on the Supercomputer Fugaku. In *Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER'20)*.

[6] Jacklin Kwan. 2022. Climate change threatens supercomputers. *Science* 378, 6616 (Oct. 2022).

[7] Etienne Le Sueur and Gernot Heiser. 2010. Dynamic voltage and frequency scaling: the laws of diminishing returns. In *Proceedings of the International Conference on Power Aware Computing and Systems (HotPower'10)*.

[8] Júnior Löff, Dalvan Griebler, Gabriele Mencagli, Gabriell Araujo, Massimo Torquati, Marco Danelutto, and Luiz Gustavo Fernandes. 2021. The NAS Parallel Benchmarks for evaluating C++ parallel programming frameworks on shared-memory architectures. *Future Generation Computer Systems* 125 (Dec. 2021).

[9] Diogo Marques, Helder Duarte, Aleksandar Ilic, Leonel Sousa, Roman Belenov, Philippe Thierry, and Zakhar A Matveev. 2017. Performance Analysis with Cache-Aware Roofline Model in Intel Advisor. In *Proceedings of the International Conference on High Performance Computing & Simulation (HPCS'17)*.

[10] José Morgado, Leonel Sousa, and Aleksandar Ilic. 2024. CARM Tool: Cache-Aware Roofline Model Automatic Benchmarking and Application Analysis. In *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC'24)*.

[11] Guillaume Raffin and Denis Trystram. 2024. Dissecting the Software-Based Measurement of CPU Energy Consumption: A Comparative Analysis. *IEEE Transactions on Parallel and Distributed Systems* 36 (June 2024). Issue 1.

[12] Stephen Roberts, Steven A. Wright, Suhaib Fahmy, and Stephen Andrew Jarvis. 2017. Metrics for Energy-Aware Software Optimisation. *Lecture Notes in Computer Science (LNCS)* 10266 (June 2017).

[13] Stephen I. Roberts, Steven A. Wright, Suhaib A. Fahmy, and Stephen A. Jarvis. 2019. The Power-Optimised Software Envelope. *ACM Transactions on Architecture and Code Optimization* 16, 3, Article 21 (June 2019).

[14] Stephen I. Roberts, Steven A. Wright, David Lecomber, Christopher January, Jonathan Byrd, Xavier Oro, and Stephen A. Jarvis. 2015. POSE: A Mathematical and Visual Modelling Tool to Guide Energy Aware Code Optimisation. In *Proceedings of the 6th International Green and Sustainable Computing Conference (IGSC'15)*.

[15] Robert Schöne, Markus Schmidl, Mario Bielert, and Daniel Hackenberg. 2021. FIRESTARTER 2: Dynamic Code Generation for Processor Stress Tests. In *Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER'21)*.

[16] Woong Shin et al. 2024. Towards Sustainable Post-Exascale Leadership Computing. In *Proceedings of the Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC-W'24)*.

[17] Woong Shin et al. 2025. Bridging the Gap: User-Centric Energy Monitoring for Policy-Driven Application Optimization in HPC Data Centers. In *Proceedings of the Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC-W'25)*.

[18] Ana Luisa Veroneze Solórzano et al. 2024. Toward Sustainable HPC: In-Production Deployment of Incentive-Based Power Efficiency Mechanism on the Fugaku Supercomputer. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'24)*.

[19] Estela Suarez et al. 2025. Energy-aware operation of HPC systems in Germany. *Frontiers in High Performance Computing* 3 (Feb. 2025).

[20] Michela Taufer, Catherine Gill, and Chandra Krintz. 2025. Computing is Eating the World: Will Saving the Planet Destroy It?. In *AAAS Annual Meeting*.

[21] Samuel Williams, Andrew Waterman, and David Patterson. 2009. Roofline: An Insightful Visual Performance Model for Multicore Architectures. *Commun. ACM* 52, 4 (April 2009).

[22] Zhengji Zhao, Ermal Rrapaj, Sridutt Bhalachandra, Brian Austin, Hai Ah Nam, and Nicholas Wright. 2023. Power Analysis of NERSC Production Workloads. In *Proceedings of the Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W'23)*.