
White Rose Research Online

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk
Universities of Leeds, Sheffield and York

Deposited via The University of Leeds.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/92900/

Version: Accepted Version

Proceedings Paper:
Armstrong, D, Kavanagh, R and Djemame, K (2015) Towards an interoperable energy efficient Cloud computing architecture-practice & experience. In: 2015 IEEE International Conference on Communication Workshop, ICCW 2015. 2015 IEEE International Conference on Communication Workshop (ICCW),, 08-12 Jun 2015, London, United Kingdom. Institute of Electrical and Electronics Engineers, 1807 - 1812. ISBN: 9781467363051.

https://doi.org/10.1109/ICCW.2015.7247443

Reuse
Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.


UNIVERSITY OF LEEDS
University of Sheffield
UNIVERSITY of York

Towards An Interoperable Energy Efficient Cloud Computing Architecture - Practice & Experience

Django Armstrong, Richard Kavanagh & Karim Djemame*,
*School of Computing, University of Leeds, UK
D.J.Armstrong,R.E.Kavanagh,K.Djemame@leeds.ac.uk

***Abstract*—The energy consumption of Cloud computing continues to be an area of significant concern as data center growth continues to increase. This paper reports on an energy efficient interoperable Cloud architecture realized as a Cloud toolbox that focuses on reducing the energy consumption of Cloud applications holistically across all deployments models. The architecture supports energy efficiency at service construction, deployment, and operation and interoperability through the use of the Open Virtualization Format (OVF) standard. We discuss our practical experience during implementation and present an initial performance evaluation of the architecture. The results show that the implementing Cloud provider interoperability is feasible and incurs minimal overhead during application deployment in comparison to the time taken to instantiate Virtual Machines.**

I. Introduction

Energy efficiency is at the heart of the EUs Europe 2020 Strategy for smart, sustainable and inclusive growth as part of a transition to a resource efficient economy. Current trends in industry show continuous growth in the adoption and market value of Cloud computing with many companies changing their business models and products to adapt to a service orientated outlook (e.g. Microsoft's planned Windows 10 give-away, Office 365). With this growth, predictions have been made on an unsustainable quadrupling in the energy consumption and carbon emissions of data centres used to operate Cloud services by 2020 [23] with comparable emissions to the aeronautical industry. Thus considering and improving the energy efficiency of cloud computing is important. This paper is concerned with this topical issue and specifically focuses on the design, construction, deployment, and operation of cloud services through the implementation of a reference energy-aware architecture that supports provider interoperability through the use of OVF [20]. OVF if an open standard for defining, packaging and distributing virtual appliances that can run virtualized on a cloud.

The paper's main contributions are: 1) An energy efficiency aware cloud architecture, which is discussed in the context of the cloud service life cycle: construction, deployment, and operation; and 2) An implementation of the OVF standard enabling energy-awareness and interoperability through a standardised cloud application descriptor. The remainder of the paper is structured as follows: Section II reviews the literature on energy-aware Cloud computing. Section III describes the proposed architecture to support energy-awareness. Section IV explains how the standard OVF is used to specify resources for an initial deployment on an energy-aware cloud. Section V presents the experimental design, and Section VI discusses the evaluation results of the architecture. In conclusion, Section VII provides a summary of the research.

II. Related Work

Research effort has targeted energy efficiency support at various stages of the cloud service lifecycle. In the *service development stage*, *requirements elicitation* includes techniques for capturing, modelling and reasoning on energy requirements as well as product line oriented techniques to model and reason about system configuration [6], [7]. In terms of *software design* in relation to energy consumption, some research efforts relate energy awareness and optimization at the application and system level [24], focus on profiling the applications energy consumption at runtime to iteratively narrow down on energy hot spots [10], or considers Cloud architecture patterns to achieve greener business processes [1]. Energy efficiency has also been the subject of investigation in *Software development*, e.g. by studying the energy consumption of the application prior to deployment [8]. In the service *deployment stage*, research effort has focused on *Service Level Agreement* (SLA) deployment strategies especially with regard to SLAs that are energy-aware, e.g. by implementing specific policies to save energy [12], [11], as well as service deployment technologies which play a critical role in the management of the cloud infrastructure and thus have an effect on its overall energy consumption [3]. In the service *operation* stage, energy efficiency has been extensively studied and has focused for example on approaches towards energy management for distributed management of Virtual Machines (VMs) in cloud infrastructures, where the goal is to improve the utilization of computing resources and reduce energy consumption under workload independent quality of service constraints [2]. Other research effort has focused on *scheduling techniques, data management, virtualisation, networks, and operating systems*. For a full state of the art on the subject of energy efficiency in clouds see [4]. The use of OVF [20] as part of a service descriptor to define the requirements of an applications is not new and has been implemented within the OPTIMIS Toolkit [21], where an OVF fragment resides in a non-standard XML based service manifest schema. One issue with this approach is the impact on interoperability with Cloud Providers that need to support this schema to enable application deployment. This compared to the solution that is presented in this paper where a pure OVF document is used, extended and implemented according to the capabilities of the OVF Specification version 1.1.1 (last updated 22/08/2013) makes our solution 100% compliant with Cloud providers and technologies that already support OVF.

III. Energy Efficient Cloud Architecture

To reduce the energy consumption of a cloud system, a reference architecture is needed to first enable energy aware-

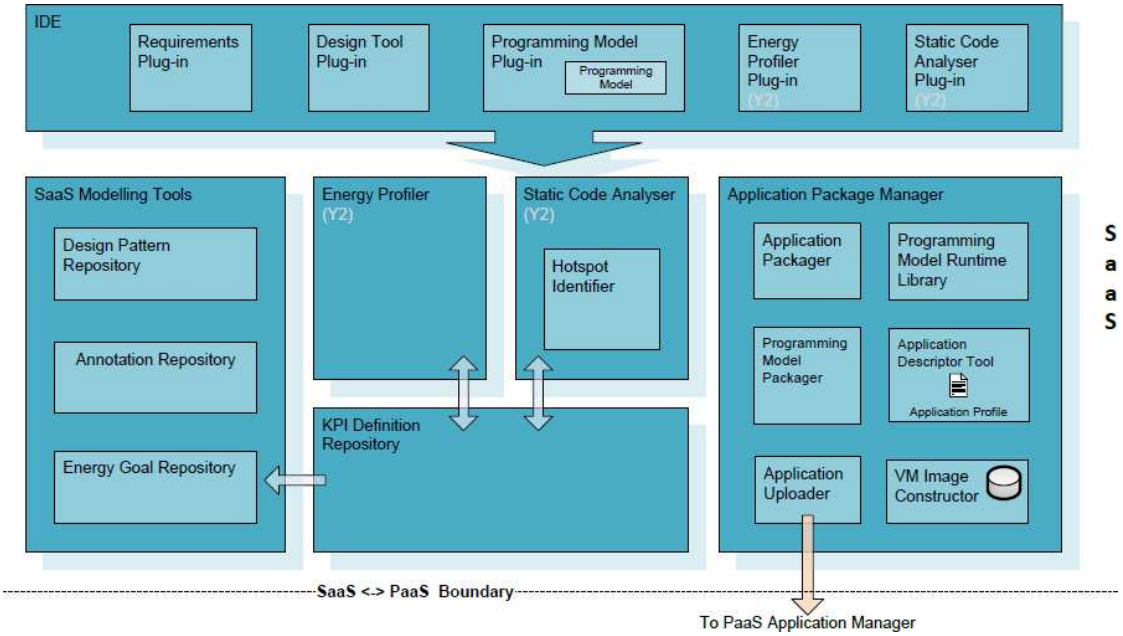

Fig. 1: ASCETiC Architecture - SaaS Layer

ness of all phases of an application's life-cycle and secondly provide actuators to reduce and optimizes energy efficiency. To this end we have realised such a reference architecture through the implementation of a toolbox, details of which can be found in [9]. Figures 1,2,3 provide an overview of the proposed architecture. It includes the high-level interactions of all components, is separated into three distinct layers and follows the standard Cloud deployment model.

In the SaaS layer, illustrated by Figure 1, a set of components interact to facilitate the modelling, design and construction of a Cloud application. The components aid in evaluating energy consumption of a Cloud application during its construction. A number of plug-ins are provided for a frontend *Integrated Development Environment* (IDE) as a means for developers to interact with components within this layer. A number of packaging components are also made available to enable provider agnostic deployment of the constructed cloud application, while also maintaining energy awareness.

The PaaS layer, illustrated by Figure 2, provides middleware functionality for a Cloud application and facilitates the deployment and operation of the application as a whole. Components within this layer are responsible for selecting the most energy appropriate provider for a given set of energy requirements and tailoring the application to the selected providers hardware environment. Application level monitoring is also accommodated for here, in addition to support for Service Level Agreement (SLA) negotiation.

In the IaaS layer, illustrated by Figure 3, the admission, allocation and management of virtual resource are performed through the orchestration of a number of components. Energy consumption is monitored, estimated and optimized using translated PaaS level metrics. These metrics are gathered via a monitoring infrastructure and a number of software probes. The *Energy Awareness* provision is an important step in the architecture implementation plan as it concentrates on delivering energy awareness in all system components. Monitoring and metrics information will be measured at IaaS level and propagated through the various layers of the Cloud stack (PaaS, SaaS) considering static energy profiles. The *Cloud Stack Adaptation* with regard to energy efficiency will focus on the addition of capabilities required to achieve dynamic energy management per each of the Cloud layers, in other words *intra* and *inter* layer adaptation and is the subject of future work.

IV. Life-cycle, Energy-Awareness and OVF

This section discusses the life-cycle of Cloud applications in the context of energy-awareness and the metrics this requires, in addition to the ASCETiC implementation of the OVF standard used to describe applications. There are many phases to the life-cycle of an application run on a Cloud. The phases recognised by the ASCETiC architecture are:

1) **Construction** - An application is developed, integrated, tested, packaged and uploaded on a developer's local machine via an IDE.
2) **Submission** - The application and its description is submitted for deployment on to a Cloud provider.
3) **Negotiation** - A number of rounds of Service Level Agreement (SLA) negotiation are performed with suitable Cloud providers that meet the requirements of the application contained within its descriptor.
4) **Contextualization** - The application is configured in the context of the Cloud provider's environment for which it is about to be executed (see [3] for details).
5) **Deployment** - The application is distributed to and scheduled on the physical infrastructure of the selected Cloud provider.
6) **Initialization** - The application is initializing its virtual resources, i.e. the VM's operating system is booting and the application is starting up.
7) **Operation** - The application is ready and able to service end-user requests.

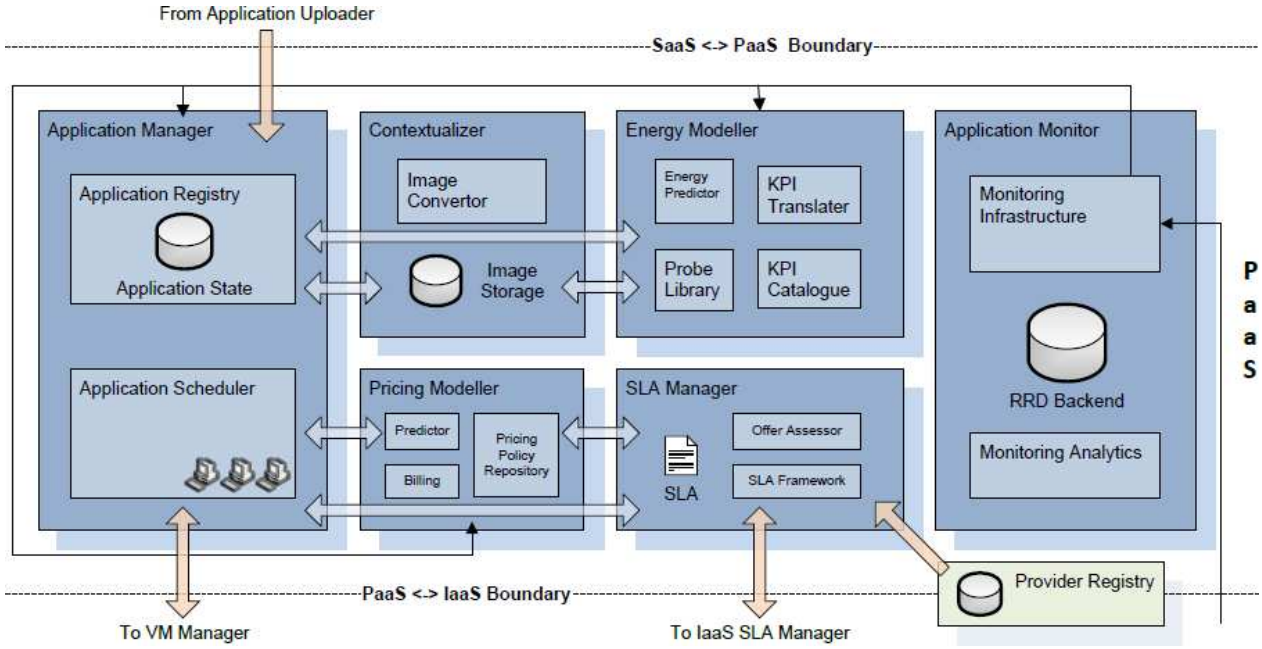

Fig. 2: ASCETiC Architecture - PaaS Layer

8) **Undeployment** - The application execution is stopped and is removed from the selected Cloud provider.

In the context of this paper the following phases have been covered in experimentation: 1) Submission; 2) Negotiation; 3) Contextualization; 4) Deployment; 5) Initialization and 6) Undeployment. This choice has been made to provide perspective on the performance and energy consumption of the ASCETiC architecture and its OVF implementation. Construction and Operation have been omitted from this scope as these phases are highly dependent on the application. This is also true of Initialization but this phase enables comparisons to be drawn on the energy consumption of the ASCETiC architecture against that of operating a Cloud application, where by in normal usecases of a Cloud, Operation energy consumption far out-ways that used in Initialization. Energy metrics at various levels (host, VM, tasks) are targeted in the ASCETiC Architecture in order to measure the energy consumed by application's events, operations or components defined at development time. These metrics have different contexts: software, platform, infrastructure and architectural component level. Examples of such metrics include: 1) host energy consumption from the socket; 2) host resource usage (e.g. CPU Utilization); 3) VM energy consumption; 4) VM resource usage (e.g. I/O Utilization); 5) application energy consumption; and 6) application throughput (e.g. sessions per time unit).

The metrics used in this paper's evaluation are discussed in further detail within the next section of the paper and are used within the ASCETiC OVF implementation and as a data model, starting from the SaaS layer. A developer at the SaaS layer can use the integrated plug-ins within their IDE to create an OVF description of the application and it's potential workload profile they wish to construct, deploy and execute containing metrics and KPIs within the extensible *<ProductSection>* key-value store. In addition to this, the section is used to specify the constraints on an applications virtual resource allocation. When a developer is ready to deploy their application the images and associated OVF description is uploaded to the PaaS layer and processed by the Application Manager component where contextulization adds configuration to the OVF in the form of a virtual CDROM device and a unique deployment ID. Additionally, energy and price modelling alongside SLA negotiation is performed using the OVF virtual resource and workload definitions, to evaluate the suitability of cloud providers to run the application. Finally the altered OVF is passed to the IaaS layer where the (Virtual Machine Manager) VMM component breaks up the OVF into a number of VM instances and instantiates them via an API call to the virtual infrastructure manager. The freely available open source ASCETiC implementation of the OVF specification [13] is implemented in Java using XMLBeans [22] and to the best of our knowledge at the time of writing, there are currently no complete Java implementations available.

V. Experimental Design

This section presents the experimental design. The objectives of the experiments are to ascertain the viability of using OVF as a means to enable interoperate between cloud service providers and capture/share energy efficiency requirements while having a minimal impact on the energy consumption of the Cloud system as a whole.

A. Cloud Testbed

To perform the experiments outlined in this section, a cloud testbed was used. The cloud testbed is located at the *Technische Universität Berlin* (see Figure 4). The computing cluster consists of sixteen nodes. Each of these nodes is equipped with two quad-core processors with $2,66$ GHz, 32 GB of RAM,

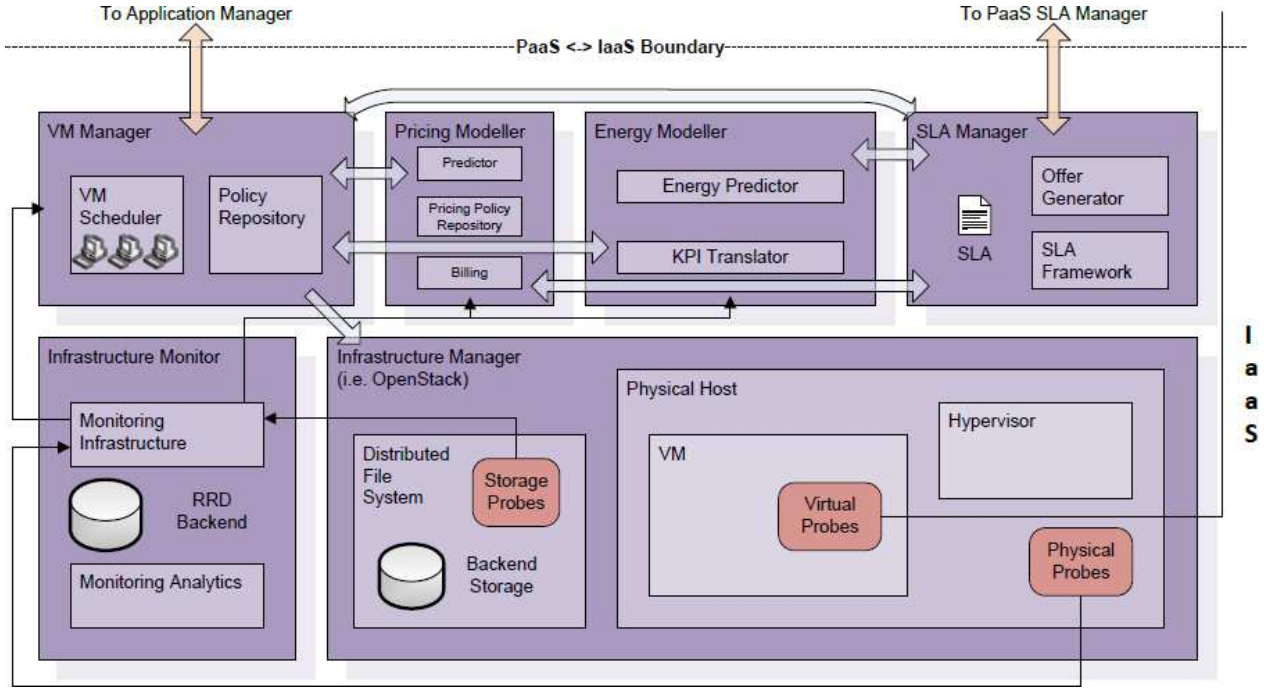

Fig. 3: ASCETiC Architecture - IaaS Layer

750 GB of local hard disk capacity and an IPMI card for administration. Each node is connected to two different networks and able to transfer full speed with one Gbit/s synchronously. The first network is dedicated for infrastructure management as well as regular data exchange between the nodes. The second network is available for storage area network usage only where storage nodes are accessible through a distributed file systems. While some hardware information is obtainable through the IPMI we measure the energy-consumption of each node just before the power supply unit. Each energy-meter can measure voltage, current and power consumption. We use identical energy-meters to guarantee comparative measurements. The actual devices are *Gembird EnerGenie Energy Meters* [5] that share their measurements over the local network. These devices can measure power up to 2500 watts with an accuracy of $\pm 2\%$ and are able to deliver two measurements per second. A dedicated node collects all measurements regularly and can share the aggregated information with monitoring components. Furthermore, this node shares visualized real-time information in a web front-end with the local cluster administrators. Additionally, the Cloud Testbed deploys OpenStack[17] to manage virtual infrastructure and Zabbix[19] to store monitored data. The VMM component within the ASCETiC Architecture was configured to use an energy aware scheduling algorithm that tries to minimise energy consumption of newly provisioned VMS.

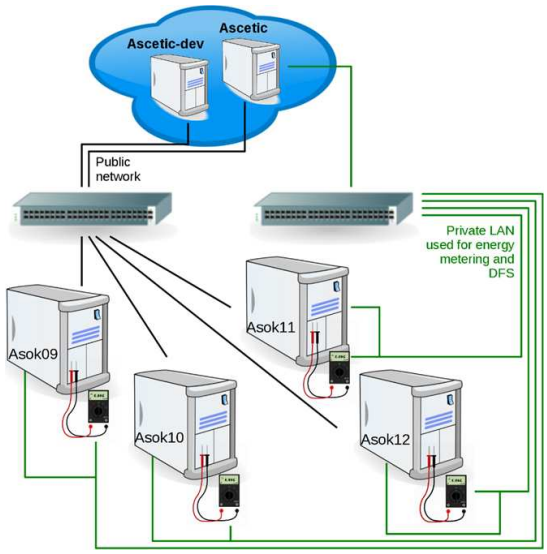

Fig. 4: Cloud Testbed

B. Cloud Application & Experimental Set-up

The application we have chosen to fulfil the objectives of the experiments is a generic three tier web application. The three tier web application is composed of a set of VM images as illustrated in Figure 5. The load balancer image implemented via HAProxy [14] distributes load between application servers. These application servers are comprised of a single JBoss [15] web container running within a Java VM and have pre-installed a photo album application. The photo album application stores and retrieves data within a single MySQL [16] image.

To ascertain the feasibility of using OVF and showcase the energy awareness of the architecture in the life-cycle (as described in Section IV) of a Cloud application, the experimental results in the subsequent section measure the time taken to complete each life-cycle phase. Additionally the energy consumption and resource usage of the VMs deployed are recorded. The experiments record the time taken to run the Cloud application through its life-cycle in five different VM configurations each performed over ten iterations to reduce

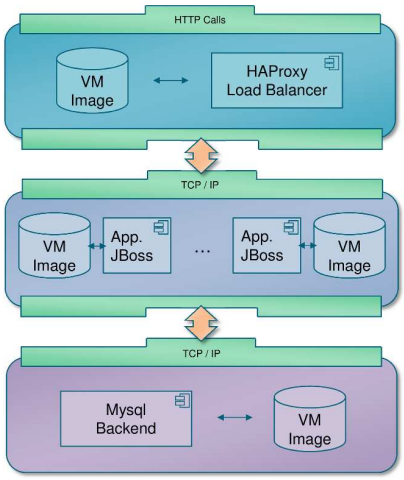

Fig. 5: Three Tier Web Application Architecture.

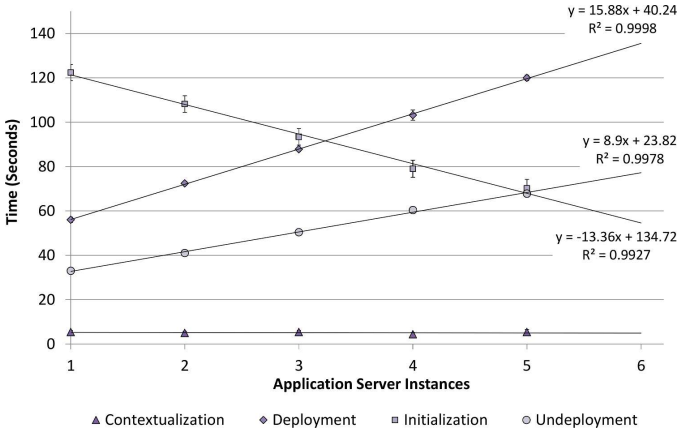

Fig. 7: Linear relationships of application life-cycle phases against the number of application server instances.

variance. Each configuration of VMs requires the use of both the load balancer and MySQL backend after which the number of JBoss application server instances is incremented from one to five VMs. In addition to this, the per-VM attributed energy consumption and CPU utilization are recorded as part of the standard functionality in the ASCETiC architecture amongst many other metrics and KPIs that are readily available for retrieval.

VI. Results

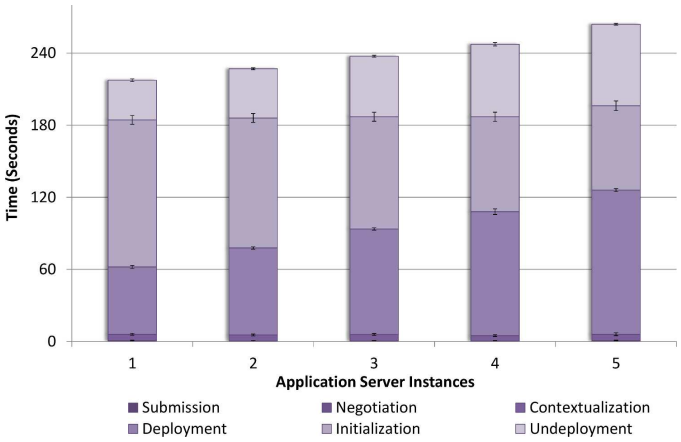

Fig. 6: Application life-cycle phases against the number of application server instances.

This section discusses the performance results obtained from the execution of the experiment defined in the previous section. Figure 6 shows the time associated with each phase of the life-cycle of an application run on the ASCETiC architecture. The general performance message from the experimental results show that increasing the number of VMs impacts different life-cycle phases in different ways. From the graph it can be seen that as the number of application server instances increases from 1 to 5 VMs that the Submission and Negotiation phases using the OVF application descriptor remains negligible. The time to contextualize is constant at around 5 seconds. Deployment and Undeployment of the VMs via OpenStack increases as the number of application server instances increases and oddly the initialization time of the application decreases. This decrease in the initialization phase time of the experiment is a artefact that can be attributed to the specific application and additionally the mechanism used to detect when initilaization of the application is complete. It can be attributed to the non-deterministic nature in which the application server instances register with the HAProxy load balancer, where by only a single instance is required for the application stack to be functional. As the number of instances increases there is a higher probability that one of these instances will be online and available to service requests.

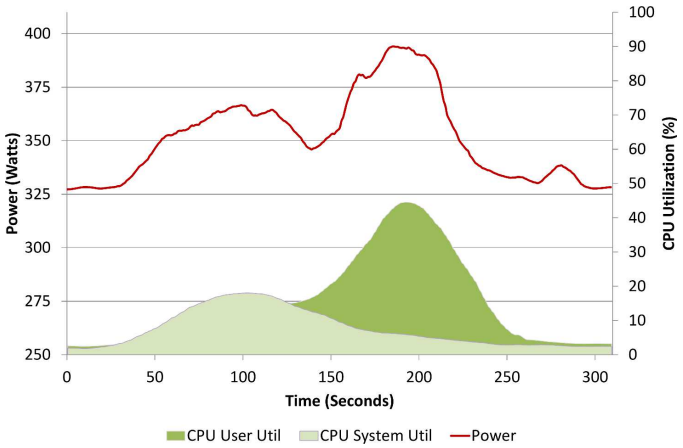

Fig. 8: Power and CPU Utilization during deployment

From Figure 7 the linear relationships between the time associated with the completion of each life-cycle phase and the number of application server Instances can be seen. A full set of the data from the experiment, including standard deviation of phase times, can been seen in the Table I. From the standard deviation it can be seen that the variance between experimental iterations is minimal. Finally Figure 8 shows the aggregated power consumption and CPU utilization as a moving average of a single deployment of the three tier web application using 2 application server instances and its other associated VMs. From the graph it can be seen that deployment and initialization of the virtual machines accounts for the majority of energy consumed through the time period

TABLE I: Application life-cycle phases against number of application server instances.

Phase	Application Server Instances				
	1	2	3	4	5
Submission	0.4s (+/- 0.59)	0.4s (+/- 0.40)	0.2s (+/- 0.40)	0.2s (+/- 0.40)	0.2s (+/- 0.40)
Negotiation	0.2s (+/- 0.40)	0.0s (+/- 0.40)	0.2s (+/- 0.40)	0.2s (+/- 0.40)	0.4s (+/- 0.49)
Contextualization	5.4s (+/- 0.80)	5.0s (+/- 0.80)	5.4s (+/- 0.80)	4.4s (+/- 0.80)	5.4s (+/- 0.80)
Deployment	56.0s (+/- 1.26)	72.4s (+/- 0.98)	87.8s (+/- 0.98)	103.2s (+/- 2.32)	120.0s (+/- 1.26)
Initialization	122.4s (+/- 3.61)	108.2s (+/- 3.72)	93.4s (+/- 3.72)	79.0s (+/- 2.85)	70.2s (+/- 4.02)
Undeployment	33.0s (+/- 1.10)	41.0s (+/- 0.80)	50.4s (+/- 0.80)	60.4s (+/- 1.50)	67.8s (+/- 0.75)

5-250 seconds. The initial phases of submission, negotiation and contextualization that utilize an OVF description and are directly attributable to the energy consumption of ASCETiC architecture is minimal.

VII. Conclusion

This paper has highlighted the importance of providing novel methods and tools to support software developers aiming to optimise energy efficiency and minimise the carbon footprint resulting from designing, developing, deploying and running software at the different layers of Cloud stack while maintaining other quality aspects of software to adequate and agreed levels. In the architecture, energy efficiency is addressed at all layers of the Cloud software stack and during the complete life-cycle of a Cloud application. OVF is used to enable interoperability between Cloud providers and as a data model across the architectural layers. Its implementation, feasibility and performance has been showcased via the deployment of a three tier web application illustration, where by the introduction of OVF and the ASCETiC architecture has minimal impact on life-cycle phase times and host energy consumption. Future work on the architecture will include support for the Topology and Orchestration Specification for Cloud Applications (TOSCA) [18] future proofing further Cloud provider interoperability.

Acknowledgments

This work is partly supported by the European Commission under FP7-ICT-2013.1.2 contract 610874 - Adapting Service lifeCycle towards EfficienT Clouds (ASCETiC) project.

References

[1] Alexander Nowak and Frank Leymann. Green Business Process Patterns - Part II (Short Paper). In *6th IEEE International Conference on Service-Oriented Computing and Applications*, pages 168–173, Koloa, Hi, 2013.

[2] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Future Generation Computer Systems*, 28(5):755 – 768, 2012.

[3] Django Armstrong, Daniel Espling, Johan Tordsson, Karim Djemame, and Erik Elmroth. Runtime Virtual Machine Recontextualization for Clouds. In *Euro-Par 2012: Parallel Processing Workshops*, volume 7640 of *Lecture Notes in Computer Science*, pages 567–576. Springer Berlin Heidelberg, 2013.

[4] ASCETiC. Requirements Specification and State of the Art. Deliverable D2.1.1, February 2014. http://www.ascetic.eu/content/state-art.

[5] GEMBIRD Deutschland GmbH. EGM-PWM-LAN data sheet. http://gmb.nl/Repository/6736/EGM-PWM-LAN_manual—7f3db9f9-65f1-4508-a986-90915709e544.pdf, 2013.

[6] Sebastian Götz, Claas Wilke, Sebastian Cech, and Uwe Aßmann. Runtime variability management for energy-efficient software by contract negotiation. *Proceedings of the International Workshop on Models@ run. time*, 2011.

[7] Wolfgang Hilty, Lorenz M.; Lohmann. The Five Most Neglected Issues in "Green IT". *CEPIS UPGRADE*, 12(4):11–15, 2011.

[8] Timo Hönig, Christopher Eibel, Rüdiger Kapitza, and Wolfgang Schröder Preikschat. SEEP: Exploiting Symbolic Execution for Energy-aware Programming. In *Proceedings of the 4th Workshop on Power-Aware Computing and Systems*, HotPower '11, pages 4:1–4:5, New York, NY, USA, 2011. ACM.

[9] Karim Djemame, Django Armstrong, Richard E. Kavanagh, Ana Juan Ferrer, D. G. Perez, D. R. Antona, Jean-Christophe Deprez, Christophe Ponsard, D. Ortiz, M. Macías, Jordi Guitart, Francesc Lordan, Jorge Ejarque, Raul Sirvent, Rosa M. Badia, M. Kammer, Odej Kao, Eleni Agiatzidou, Antonis Dimakis, Costas Courcoubetis, and L. Blasi. Energy Efficiency Embedded Service Lifecycle: Towards an Energy Efficient Cloud Computing Architecture. In *Proceedings of the Energy Efficient Systems (EES'2014) Workshop, 2nd International Conference on ICT for Sustainability 2014*, volume 1203, page 1–6, Stockholm, Sweden, Aug 2014. CEUR Workshop Proceedings. http://ceur-ws.org/Vol-1203/EES-paper1.pdf.

[10] Kay Grosskop and Joost Visser. Identification of Application-level Energy Optimizations. In Lorenz M. Hilty, editor, *Proceedings of the First International Conference on Information and Communication Technologies for Sustainability (ICT4S'2013)*, Zurich, Switzerland, February 2013.

[11] Sonja Klingert, Andreas Berl, Michael Beck, Radu Serban, Marco Girolamo, Giovanni Giuliani, Hermann Meer, and Alfons Salden. Sustainable Energy Management in Data Centers through Collaboration. In *Energy Efficient Data Centers*, volume 7396 of *Lecture Notes in Computer Science*, pages 13–24. Springer Berlin Heidelberg, 2012.

[12] Olli Mammela, Mikko Majanen, Robert Basmadjian, Hermann Meer, Andre Giesler, and Willi Homberg. Energy-aware job scheduler for high-performance computing. *Computer Science - Research and Development*, 27(4):265–275, 2012.

[13] ASCETiC OVF Java Implementation - Available on SVN. https://ascetic-dev.cit.tu-berlin.de/svn/trunk/utils/ovf-xmlbeans-api, Jan 2015.

[14] HAProxy - A Reliable, High Performance TCP/HTTP Load Balancer. http://www.haproxy.org/, Jan 2015.

[15] JBoss Application Server. http://jbossas.jboss.org/, Jan 2015.

[16] MySQL - Open Source Database. http://www.mysql.com/, Jan 2015.

[17] OpenStack: Open source software for building private and public clouds. http://www.openstack.org/, Jan 2015.

[18] Topology and Orchestration Specification for Cloud Applications - Version 1.0. http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html, Jan 2015.

[19] Zabbix - An Enterprise-class Monitoring Solution. http://www.zabbix.com/, Jan 2015.

[20] Open Virtualization Format (OVF) - A standard from the Distributed Management Task Force. http://www.dmtf.org/standards/ovf, Jan 2015.

[21] OPTIMIS Toolkit. http://optimistoolkit.com, Jan 2015.

[22] XMLBeans. http://xmlbeans.apache.org/, Jan 2015.

[23] Michael Pawlish, Aparna S. Varde, and Stefan A. Robila. Cloud Computing for Environment-friendly Data Centers. In *Proceedings of the Fourth International Workshop on Cloud Data Management*, CloudDB '12, pages 43–48, New York, NY, USA, 2012. ACM.

[24] Steven te Brinke, Somayeh Malakuti, Christoph Bockisch, Lodewijk Bergmans, and Mehmet Aksit. A design method for modular energy-aware software. In Sung Y. Shin and Jose Carlos Maldonado, editors, *Procedings of the 28th Annual ACM Symposium on Applied Computing (SAC'2013*, pages 1180–1182. ACM, 2013.