Energy Efficiency Embedded Service Lifecycle: Towards an Energy Efficient Cloud Computing Architecture

Karim Djemame, Django Armstrong, Richard Kavanagh[*],
Ana Juan Ferrer, David Garcia Perez, David Antona[§], Jean-Christophe Deprez, Christophe Ponsard[||],
David Ortiz, Mario Macias, Jordi Guitart, Francesc Lordan, Jorge Ejarque, Raul Sirvent, Rosa Badia[†],
Michael Kammer, Odej Kao[¶], Eleni Agiatzidou, Antonis Dimakis, Costas Courcoubetis[††], Lorenzo Blasi[**]
[*]School of Computing, University of Leeds, UK
[†]Barcelona Supercomputing Center, Spain
[§]Atos Research, Barcelona, Spain
[¶]Technische Universitat Berlin, Germany
[||]Centre d'Excellence en Technologies de l'Information et de la Communication, Charleroi, Belgium
[**]HP Italiana SRL, Italy
[††]Athens University of Economic and Business - Research Center, Greece

Abstract—**The paper argues the need to provide 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 in Clouds, while maintaining other quality aspects of software to adequate and agreed levels. A cloud architecture to support energy efficiency at service construction, deployment, and operation is discussed, as well as its implementation and evaluation plans.**

I. Introduction

Cloud Computing aims to streamline the on-demand provisioning of software, hardware, and data to provide flexibility and agility, and economies of scale in IT resource management. Although building, deploying and operating applications on a cloud can help to achieve speed, scalability, and maintain a flexible infrastructure, it brings about a variety of challenges due to its massive scalability, complexity, as well as dynamic and evolving environments.

There is currently an acceleration of adoption of cloud applications and services by enterprises. Consequently, experts warn of a dramatic increase in energy consumption for cloud computing. It is predicted that the global energy consumption for cloud computing will increase from 632 billion kWh in 2007 to 1,963 billion kWh by 2020 and the associated CO_2 equivalent emissions will reach 1,034 megatonnes [9].

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 [5], [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 [14], focus on profiling the applications energy consumption at runtime to iteratively narrow down on energy hot spots [6], or considers Cloud architecture patterns to achieve greener business processes [13]. 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], [10], 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 [1]. 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 [3]. 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 [2].

This paper is concerned with the topical issue of energy efficient computing, specifically focusing on the design, construction, deployment, and operation of cloud services. It argues that research is needed to propose novel methods and develop tools to support software developers in monitoring, minimising the carbon footprint and optimising energy efficiency resulting from developing and deploying software in cloud environments. Such research addresses the need for continued development of infrastructure support for clouds in order to optimise, monitor and reduce carbon footprint and costs for cloud providers and end-users. The major contribution to the carbon footprint of IT software in general is energy consumed in its operation, thus the primary aim of this paper is to relate software design and energy use. Although energy use is of relevance across all software design and implementation, this paper makes specific reference to cloud-based service operations: the emergence of cloud computing with its emphasis on shared software components which are likely to be used and reused many times in many different applications makes it imperative that the software to be developed is as energy efficient as it possibly can be.

Therefore, the paper's primary goal is to characterise the factors which affect energy efficiency in software development, deployment and operations. The approach focuses firstly on the identification of the missing functionalities to support energy efficiency across all cloud layers, and secondly on the definition and integration of explicit measures of energy requirements into the design and development process for software to be executed on a cloud platform.

The paper's main contributions are: 1) the incorporation of a novel approach that combines energy-awareness related to cloud environments with the principles of requirements engineering and design modelling for self-adaptive software-intensive systems. This way, the energy efficiency of both cloud infrastructure and software are considered in the cloud service development and operation lifecycle, and 2) a proposed energy efficiency aware cloud architecture, its components, and their roles. This architecture is discussed in the context of the cloud service life cycle: construction, deployment, and operation.

The remainder of the paper is structured as follows: Section II describes the proposed architecture to support energy-awareness. Section III discusses the implementation and evaluation plans of the architecture. In conclusion, Section IV provides a summary of the research.

II. Energy Efficient Cloud Architecture

As argued in Section I it is clear that methods and tools that consider energy efficiency are needed to manage the life cycle of cloud services from requirements to run-time through construction, deployment, operation, and their adaptive evolution over time. Their availability will result in an implementation of a software stack for energy efficient-aware Clouds. Thus, an architecture supporting energy efficiency and capable of self-adaptation while at the same time aware of the impact on other quality characteristics of the overall cloud system such as performance is proposed.

Figure 1 illustrates the vision of how to deal with energy requirements together with other types or requirements, and how to manage them across the design and run-time stages. Consequently, the research questions that need to be addressed are the normalisation of energy measurements, the mapping between hardware, VM and software level, the management of Key Performance Indicators (KPIs) of contributing/conflicting goals as well as the identification of variability points available for (self)-adaptation.

Figures 2,3,4 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. Next, details on the interactions of the architectural components are discussed.

A. Layer 1 - SaaS

In the SaaS layer a set of components interact to facilitate the modelling, design and construction of a Cloud application.

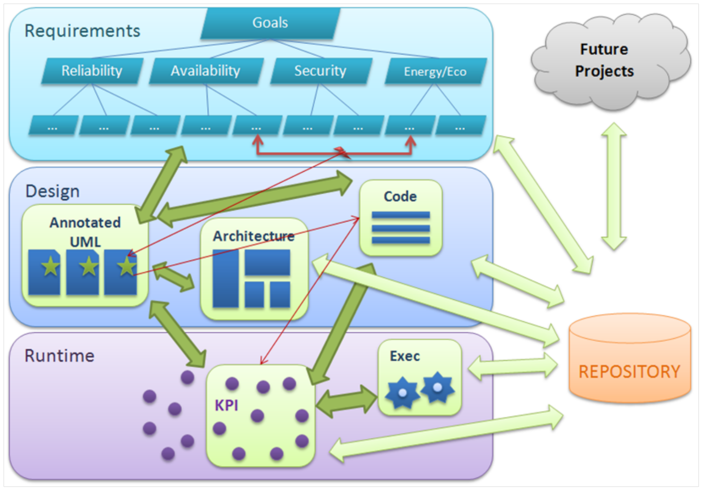

Fig. 1: Energy traceability framework across requirements, design and run-time levels

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 IDE is intended to be the main entry point to the infrastructure for service designers and developers. The idea is that the IDE integrates the graphical interfaces to the different tools available in the SaaS layer, thus offering a unified and integrated view to users. The *Programming Model Plug-in* (PM plug-in) provides a graphical interface to use the Programming Model and supporting tools to enable the development, analysis and profiling of an application in order to improve energy efficiency. On the other hand, the *Programming Model* provides the service developers with a way to implement services composed of source code, legacy applications executions and external Web services [11]. Although these complex services are written in a sequential fashion without APIs, the applications are instrumented so they call the Programming Model Runtime to be executed in parallel. The *Modelling Tools* component extends the requirements engineering and design phase with the capability to capture and reason on energy as well as other quality aspects in the early steps of the application. This will allow the SaaS architect to take the right design decision either statically or by enabling adequate run-time behaviour when deployed (e.g. through annotation propagation, SLA definitions). The goal is also to provide the KPI that will be monitored to check if the platform is behaving properly and take corrective actions in case of deviation. This component has a number of features that facilitate requirements gathering and modelling and includes: 1) *Goal energy repository*: supporting energy patterns in requirements; 2) *Design pattern repository*: supporting energy-aware cloud-patterns, and 3) *Annotation repository*: supporting the association of energy KPIs across the whole refinement structure of the (requirements) goal model. The *Application*

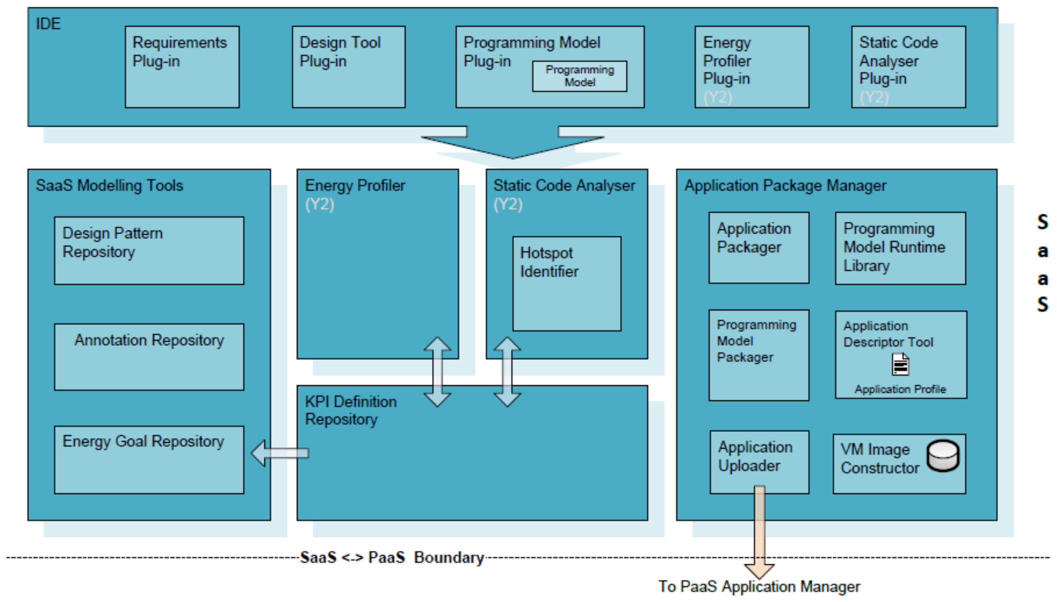
IDE
Requirements Plug-in
Design Tool Plug-in
Programming Model Plug-in
Programming Model
Energy Profiler Plug-in (Y2)
Static Code Analyser Plug-in (Y2)
SaaS Modelling Tools
Design Pattern Repository
Annotation Repository
Energy Goal Repository
Energy Profiler (Y2)
Static Code Analyser (Y2)
Hotspot Identifier
KPI Definition Repository
Application Package Manager
Application Packager
Programming Model Runtime Library
Programming Model Packager
Application Descriptor Tool
Application Profile
Application Uploader
VM Image Constructor
SaaS
SaaS <-> PaaS Boundary
To PaaS Application Manager

Fig. 2: Architecture - SaaS

Package Manager is a collection of components providing functionality to describe and bundle appropriate software, including the selection of the most suitable energy efficient runtime environment (e.g. JVM), into one or more VM images ready for deployment. It is a conceptual component in the architecture with a set of sub-components that package the application and make it ready for deployment. The *Programming Model Runtime* (PMR) deals with the orchestration of the task executions. The PMR component is in charge of detecting the dependencies among the task invocations and managing their proper execution in the remote resources. The *Programming Model Packager* (PMP) component creates the bundles of a Programming Model application. More precisely, it will pack method calls and tasks taking into account their requirements, or any other constraints pointed out by the developer. It will also generate the Service Manifest of the PM application. The *Application Descriptor Tool* is a graphical tool that assists developers in creating a Service Manifest. It helps the user to build an Open Virtualization Format (OVF) description document that describes the relation between different VMs and the software installed in them to later be used by the Application Packager to build the Service Manifest to submit to the PaaS layer. The *Application Packager* component is in charge of packaging non-PM applications. This component will take into account the template filled by the user in OVF format to package the software with the different requirements. It also generates a Service Manifest to submit to the PaaS layer. The *VM Image Constructor* (VMIC) uses the application packages and the service manifest or application descriptor to create VM images that can be deployed in the PaaS layer. The *Application Uploader* interacts with the Application Manager to register the final VMs ready for deployment. It essentially serves the PM plug-in and the Application Packager once images have been completed.

B. *Layer 2 - PaaS*

The PaaS layer 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.

The *Application Manager* (AM) component manages the user applications that are described as virtual appliances, formed by a set of VMs that are interconnected between them. The role of the *Virtual Machine Contextualizer* (VMC) is to embed software dependencies of a service into a VM image and configure these dependencies at runtime via an infrastructure agnostic contextualization mechanism. Additionally, the VMC enables the use of energy probes for the gathering of VM level energy performance metrics. The *Application Monitor* (APPM) is able to monitor the resources (CPU, memory, network ...) that are being consumed by a given application, by providing historical statistics for host and VM metrics. The monitoring of an application must be performed in terms of performance (e.g. CPU that an application is consuming during a given period of their execution) and energy (e.g. watts that an application requires during a given moment of their execution). The goal of the *Energy Modeller* is to gather and manage energy related information throughout the whole Cloud Service lifecycle and Cloud layers: from requirement level KPIs to programming model annotations down to PaaS and IaaS level measurements made through the monitoring agents present at those levels. The energy modeller provides an interface to estimate the energy cost of a PaaSs KPIs, and the

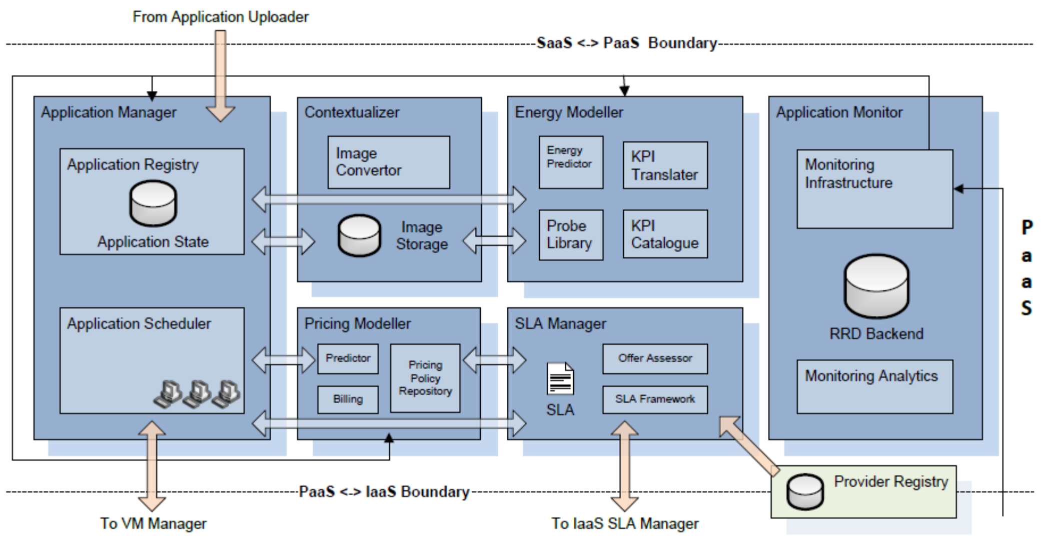

Fig. 3: Architecture - PaaS

provided estimations assist in the selection of the appropriate IaaS provider for running the application. The *SLA Manager* is responsible for managing SLAs at PaaS level. This requires interacting with the Application Manager, the Pricing Modeller and the IaaS SLA Manager. The Application Manager provides to the PaaS SLA Manager information to establish which terms need to be scheduled and then negotiated with the IaaS Providers. Once negotiation between PaaS SLA Manager and IaaS SLA Manager is done, the PaaS SLA Manager will request the price of the build offer to the Pricing Modeller. The goal of the *Pricing Modeller* is to provide energy-aware cost estimation related to the operation of applications on top of VMs on a specific IaaS provider. In addition, the component provides billing information.

C. Layer 3 - IaaS

In the IaaS layer 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 *Virtual Machine Manager* (VMM) component is responsible for managing the complete life cycle of the virtual machines that are deployed in a specific infrastructure provider. The goal of the *Energy Modeller* is to gather and manage energy related information throughout the whole Cloud Service lifecycle and Cloud layers. This components core responsibility is to provide energy usage estimates by presenting the relevant KPIs for a virtual machine deployment on the infrastructure provided. This will include cost trade off analysis based on sources such as prior experience, the application profile as defined in the SLA, which is subsequently translated into infrastructure level KPIs, and finally from current up to date monitoring information from the deployment environment. The *SLA Manager* is responsible to manage SLA negotiation requests at IaaS level. This is required in order to allow the PaaS SLA Manager to provide offering for User/SaaS that want to negotiate SLA with the platform. The IaaS SLA Manager is also responsible to contact the VM Manager to get the status of the available resources in order to determinate the offer and it contacts the Pricing Modeller to assign a price to the offered terms. The goal of the *Pricing Modeller* is to provide energy-aware cost estimation related to the operation of the physical resources that belong to the IaaS provider and are used by specific VMs. In addition, it will provide billing information. The *Infrastructure Manager* (IM) manages the physical infrastructure and redirects requests to hardware components. It maintains lists of hardware energy-meters, physical cluster nodes, network components and storage devices. External components can obtain and manipulate the state of the infrastructure through a common API that is independent of the actual hardware. The IM in the duty to provide power consumption information for each cluster node. Furthermore, the IM requires an authentication for all operations which ensures protection against attacks as well as a sufficient separation of different parties.

D. Adaptation

Recall that the paper addresses energy-efficient management of cloud resources across the entire cloud software stack. Therefore, the proposed cloud architecture needs to support self-adaptation regarding energy and eco-efficiency while at the same time being aware of the impact on other quality characteristics of the overall cloud system such as space

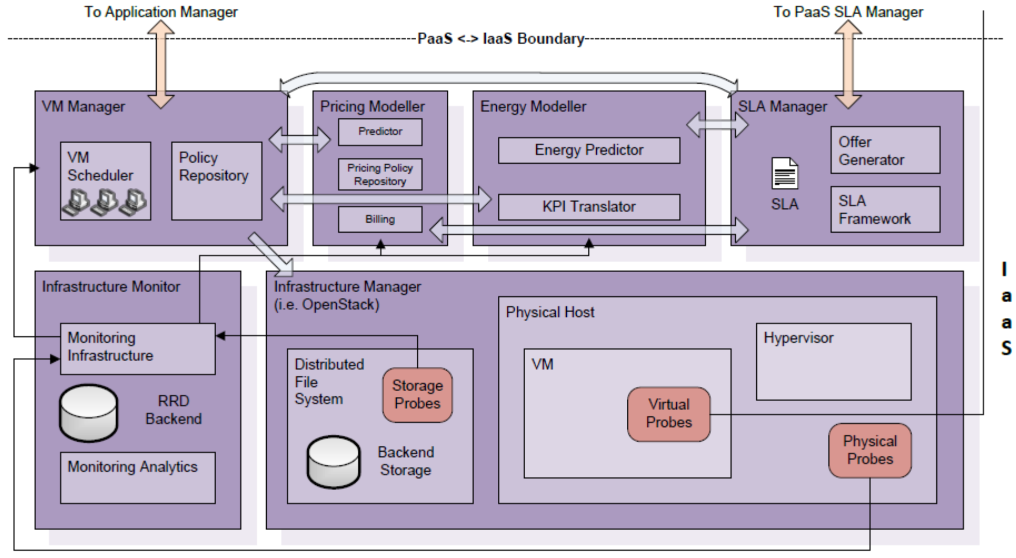

Fig. 4: Architecture - IaaS

and time performance. Three steps are foreseen: 1) *Energy Awareness*: 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; 2) *Intra-Layer Cloud Stack Adaptation*: adaptation with regard to energy efficiency focuses on the addition of capabilities required to achieve dynamic energy management per each of the Cloud layers, in other words local layer adaptation. It considers the extensions of the runtime environment in order to be able to orchestrate the invocation of different application components with advanced scheduling techniques that take into account energy efficiency parameters. It will provide the means to assess the services compliance during their operation to the terms of a negotiated SLA (and thus their QoS and their energy efficiency) as well as the overall energy efficiency from the Cloud provider perspective, and 3) *Inter-Layer Cloud Stack Adaptation*: builds on top of intra-layer Energy Efficiency, in order to achieve steering information and decisions among Cloud layers for triggering other layers to adapt their energy mode. The aim is to enable the entire Cloud stack to actively adapt to changing situations. From the SaaS perspective the main focus will be on enriching the runtime energy profiles with further PaaS and SaaS energy related attributes so that an application can be better tailored to the environment in which it is to run. At the PaaS and IaaS layers, the main focus will be on information sharing and decision making.

III. Implementation and Evaluation

The proposed architecture is currently being implemented in a cloud testbed. The evaluation plan consists in the consideration of two industrial use cases that will be used to exploit and illustrate the architecture, and the possible energy gains it will provide to the applications it operates.

Use cases: The aim of the first use case is to adapt an existing computationally intensive software and make it suitable for an energy aware cloud environment by considering a product life-cycle management system for the building Industry. This is to facilitate the design and construction of a new generation of prefabricated buildings by means of collaborative software and industrial production. The aim of the second use case is to adapt an existing data intensive software and make it suitable for an energy aware cloud environment. This software has been built to assist news agencies in creating, managing and distributing breaking news quickly and efficiently to various customers through a variety of delivery methods. Its core functionality includes: news gathering, editing, archiving, managing the production process and delivering services through multiple channels.

Metrics and KPIs: Two types of information for a running application can be collected, namely its energy consumption and application events. The application, developed with SaaS tools (see section II-A), defines when an event or a specific operation occurs during the application run time (e.g. when a new request arrives to the application, when a search operation is performed etc). Moreover the application (or some architectural component) provides an OVF description of its deployment (in terms of virtual systems and their connections). Energy consumption at host physical systems is monitored and the energy consumption of guest virtual systems is estimated. Power consumption at a given sampling frequency is monitored, as well as other metrics at a similar sampling frequency to avoid inconsistencies. Applications are generally interested in specific measurements that indicate how the application behaves (e.g. response time, user served, number of times an operation has been performed). Some of these measures can be related to energy consumption, e.g. throughput-like measures (*number of operations in a given time*), or an application-level energy metric (*energy consumption of a conceptual software architecture component*).

Energy metrics at various levels (host, VM, tasks) will be targeted in order to measure the energy consumed by applications 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 the *Execution Plan Energy Efficiency* to measure how efficient a service is in its energy use or the *Disk Energy metric* to measure the energy consumed by the disk over time duration for a single VM dedicated to a given SaaS application.

Testbed: The testbed is located at the *Technische Universität Berlin*. 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, 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 several hardware information are 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* [4] that share their measurements in 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.

IV. 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.

The specification of a proposed architecture has been presented, which includes the architectural roles and scope of the components. This architecture complies with a standard Cloud architecture, considers the classical SaaS, PaaS and IaaS layers and supports components such as the Energy modeller, the VM manager, the infrastructure monitor etc. The design of the various architectural components was described, with emphasis on the extension requirements in order to support energy efficiency management. In this architecture, energy efficiency is addressed at all layers of the cloud software stack and during the complete lifecycle of a cloud application. Future work includes its implementation and evaluation, which will be showcased considering two industrial application deployment illustrations.

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] 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.

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

[3] 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.

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

[5] 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.

[6] 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.

[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] GreenPeace International. Make it green - cloud computing and its contribution to climate change, 2010. http://www.greenpeace.org/international/Global/international/planet-2/report/2010/3/make-it-green-cloud-computing.pdf.

[10] 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.

[11] Francesc Lordan, Enric Tejedor, Jorge Ejarque, Roger Rafanell, Javier Alvarez, Fabrizio Marozzo, Daniele Lezzi, Raul Sirvent, Domenico Talia, and Rosa Badia. Servicess: An interoperable programming framework for the cloud. *Journal of Grid Computing*, pages 1–25, Sep. 2013.

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

[13] 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.

[14] 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 Jos Carlos Maldonado, editors, *Procedings of the 28th Annual ACM Symposium on Applied Computing (SAC'2013*, pages 1180–1182. ACM, 2013.