DOI 10.12694/scpe.v17i4.1204
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 17, Number 4, pp. 323–330. http://www.scpe.org

EXPOSING HPC SERVICES IN THE CLOUD: THE CLOUDLIGHTNING APPROACH

IOAN DRĂGAN*, TEODOR-FLORIN FORTIŞ† AND MARIAN NEAGUL‡

Abstract. Nowadays we are noticing important changes in the way High Performance Computing (HPC) providers are dealing with the demand. The growing requirements of modern data- and compute-intensive applications ask for new models for their development, deployment and execution. New approaches related with Big Data, peta- and exa-scale computing are going to dramatically change the design, development and exploitation of highly demanding applications, such as the HPC ones. Due to the increased complexity of these applications and their outstanding requirements which cannot be supported by the classical centralized cloud models, novel approaches, inspired by autonomic computing, are investigated as an alternative. In this paper, we offer an overview of such an approach, undertaken by the CloudLightning initiative [9]. In this context, a novel cloud delivery model that offers the capabilities to describe and deliver dynamic and tailored services is being considered. This new delivery model, based on a self-organizing and self-managing approach, will allow provisioning and delivery of coalitions of heterogeneous cloud resources, built on top of the resources hosted by a cloud service provider.

Key words: autonomic computing; cloud computing; HPC as a service; resource modeling; service description language

AMS subject classifications. 68M14, 68Q10

1. Introduction. With the continuous development of cloud computing, and the integration of new resource types, novel methods are considered for application development. The diversity of available resources, coupled with the convergence of the different approaches (like fog and edge computing, Internet of Things and cloud of things, bare-metal clouds, and others), are moving the focus towards building of more complex and demanding applications, for which the classical centralized cloud approach – where a full set of information is made available to support decisions – is not enough anymore.

A different approach can be built in relation with the generic principles of autonomic computing [7] and which is centered around self-organization and management [10] to enable the construction of systems which are tailored to customers' needs. With such an approach one can equally serve the outstanding requirements of modern, large-scale distributed applications, as well as the needs of the specialized ones.

Inspired from the set of four major self-* principles (self-configuration, self-healing, self-optimization and self-protection [7]), the approach considered in the context of the CloudLightning project is primarily focused on self-configuration and self-optimization in order to organize and deliver coalitions of heterogeneous resources. The implied self-organization of resources imposes extensions to the notion of resource, by using the mechanisms of organization for 'basic' heterogeneous resources in order to offer so called 'mega-resources' (*i.e.*, CloudLightning coalitions), which are then able to respond to the management requirements of a diversity of large scale applications – like commissioning, deployment or execution –, tailored for high-performance computing.

The remainder of this paper is organized as follows: section 2 offers an overview of the CloudLightning initiative, its approach, objectives and considered use cases. Section 3 offers a description of the major components of the CloudLightning architecture. Conclusions and future work description are formulated in Section 4.

2. The CloudLightning initiative.

2.1. Concept. The CloudLightning initiative intends to offer a new delivery model that will go beyond the currently used centralized management approach. An Intersect360 report identifies "provider capability to meet performance and capacity requirements" as being one of the barriers faced by High Performance Computing (HPC) to adopting cloud computing [18]. The new delivery model developed by CloudLightning will allow to make important steps in dealing with this barrier, by allowing to provision and deliver heterogeneous cloud resources in order to be consumed by HPC applications, by the means of a specialized, but still extensible description language.

CloudLightning service descriptions enables services and resources discovery – represented by the resources which are hosted by the cloud service provider –, and decomposition mechanisms which are triggering, in

*"Victor Babes" University of Medicine and Pharmacy and Institute e-Austria Timisoara Romania, (idragan@ieat.ro).
†West University of Timisoara and Institute e-Austria Timisoara Romania, (florin.fortis@e-uvt.ro).
‡West University of Timisoara and Institute e-Austria Timisoara Romania, (marian.neagul@ieat.ro).

turn, a response from the infrastructure services resulting in a proposition of some resource gatherings (the CloudLightning coalitions) capable to cover the requirements of customers' services. Heterogeneous by their nature, the CloudLightning coalitions are composed of heterogeneous components chosen to offer a better quality of service.

Self-organization and self-management are key techniques used to discover, build and propose coalitions with minimal effort related to low-level service provisioning, and to support the automation of the cloud service lifecycle. Their successful adoption limit the over-provisioning of resources [5], currently used for high demanding services, while avoiding under-provisioning of resources and enabling important savings, both on cloud provider and cloud service consumer sides.

2.2. The main components of the solution. The CloudLightning approach is based on three major components, which are further used to build the supporting use-cases [9]. Although most of these components have been extensively researched in previous projects, it is still an outstanding challenge to implement all of them in close relation to each other.

2.2.1. The declarative approach to service provisioning. The first component of the project's approach is built in relation to some concepts coming from cloud service brokers (defined as entities which "concentrate on the negotiation of relationships between consumers and providers without owning or managing the whole cloud infrastructure" [3] and can build and offer some services created over a PaaS or IaaS support). Still, the CloudLightning approach is different, as it is based on the idea that one cannot make any assumptions about the number, type and availability of resources when self-management is in action.

With this assumption, the necessity of a CloudLightning-specific service description language (CL-SDL) was identified, such that a clear separation between the declaration on what services are required and how these services are provided can be clearly realized. This clear separation allows a high level description of requested services while the resources hosted by the cloud service provider have the capability to organize themselves and respond with several offerings, if any, based on current availabilities. The CL-SDL, which is compatible with OASIS TOSCA [14], allows users to create complex solutions that are further deployed in the CloudLightning system, and provides a detailed description of the identified SDL [19].

2.2.2. Decentralized self-management. Traditionally cloud computing was based on a centralized approach, where full information about the system and its components were made available for further decisions. Once we get more diversity and increased complexity in the cloud computing landscape, new approaches to cloud management must be considered: for example, in the case of edge computing, it "is pushing computing applications, data, and services away from centralized cloud data centre architectures to the edges of the underlying network" [4, 13].

Such an approach, which is highly relevant for the case of CloudLightning self-management, will also expose an increased complexity of the problem whilst offering support for cloud management at a greater scale than in the case of the traditional, centralized approach. In order to deal with this complexity, self-organizing is considered together with the self-management component, as similarly with the "biological self-organizing system, the global goals of the system are expected to emerge from local actions" [2, 10].

2.2.3. Exposure of heterogeneous resources. HPC relevant resource types, such as GPU and GPGPU, MIC, FPGA, programmable network routers, and others, are offering promising options for cloud computing deployments whenever compute- or data-intensive applications are expected to be deployed in a cloud environment [4, 12]. While the heterogeneity of resources may be relevant for both the loosely coupled cloud deployments (e.g., the Nebula system [4]) and the tightly coupled ones (like in the case of the HARNESS project uses-cases [6]), the self-organizing and self-management approach from CloudLightning builds a distinct approach, in order to get improved efficiency and simplified implementations regardless of the range of locally available resources or their distribution. The *declarative approach to service provisioning* is thus enriched with resource discovery mechanisms allowing easier incorporation, identification and consumption of a variety of heterogeneous resources.

2.3. Use-cases. Several use cases applications, data- or compute-intensive, are considered to demonstrate the benefits of the previously mentioned self-organizing and self-management approach and the usability of

its components. The considered use cases are HPC applications, which expose outstanding requirements in compute processing power and/or data processing capabilities. They are inspired by 1. *genomics*, where the computational cost is still an important barrier; 2. *fluid dynamics* (applied in oil and gas explorations), where large scale simulation currently require dedicated and highly expensive clusters; 3. *ray tracing*, where important steps are being made towards interactive visualizations; 4. *self-optimized scientific libraries*, where increased performance for several compute-intensive libraries of scientific functions (such as BLAS/MKL, cudaFFT, or FFTW) will be supported by the specificities of the underlying heterogeneous resources. A detailed description of the currently tackled use cases can be found in [1].

2.3.1. Genome processing. The genome processing problem that CloudLightning intends to exploit in this first use-case arises from the human gene sequencing. Set in the context of the central dogma of molecular biology ("DNA makes RNA makes proteins"), and considering the huge amount of data involved in the encoding of human DNA (approximately 3 billions base pairs, a total amount of data of up to 750MB), the problem of gene sequencing becomes highly demanding in what regards computational and storage resources. In fact, as specified in [17], "genomics is a Big Data science which is going to get much bigger".

CloudLightning intends to support this topic by demonstrating how to enable some genomics-oriented systems in a CloudLightning-enabled platform, and to benchmark their performance in the context of our approaches.

2.3.2. Fluid simulations. Sophisticated models for modeling the underground basins that store natural resources of oil and gas were developed for the industry of oil & gas exploration and exploitation. These models are based on readings coming from both seismic surveys and other means of survey. A recent interest exists in multiphase flow of fluids in porous media, typically rock formations. However, as current simulations are based on in-house solutions (workstations/clusters) that are usually used under their computing capacity, their total cost is growing.

Cost reduction is an important issue for this type of applications, and it can be addressed by parallel computing approaches. According to the study from [16], only parallelization cannot offer an alternative as I/O operations become a bottleneck, due to the huge amount of processed data and the dynamics of complex fluid simulations. An approach based on heterogeneous resources and cloud-specific developments becomes thus a promising alternative. By considering this use-case, in the context of the CloudLightning project will be identified best practices required for migrating such a demanding application type to a tailored cloud environment, while keeping costs under control.

2.3.3. Ray tracing. Ray tracing is heavily used in various industries or research areas where a photorealistic preview may offer valuable information. Once the realism of the rendered scene increases, the incurred computational costs of the ray tracing process becomes extremely large. Moreover, as the process is also time consuming, it is rather performed offline, leading to important issues related to under- and over-utilization of dedicated equipments.

There are promising experiments in using heterogeneous resources in order to deliver ray tracing in real-time [8, 15]. Thus, by migrating such a problem to a self-organizing, self-management cloud environment, it will be possible to create an environment where a tailored approach, exploiting heterogeneous resources, is being offered to fully support its timely execution.

3. The architecture of the CloudLightning solution. As identified in Section 2, CloudLightning is intended to provide customers with the possibility of deploying complex applications on a coalition of *heterogeneous resources* by using a *declarative approach to service provisioning*. In order to achieve this, the principles of self-organization and self-management are applied in a decentralized way, which allows achieving the goal of optimizing resource utilization and, as a consequence, energy consumption.

Based on the declarative approach to service provisioning, a set of design requirements were identified, such as: 1. lower service cost; 2. optimize utilization/multi-tenancy and reduce operational overhead by automation; 3. provide faster service provisioning and better performance; 4. supporting dynamic workload and resource management; 5. offering service placement optimization [11]. Based on these requirements, a succinct presentation of the core components of the CloudLightning architecture is realized in what follows.

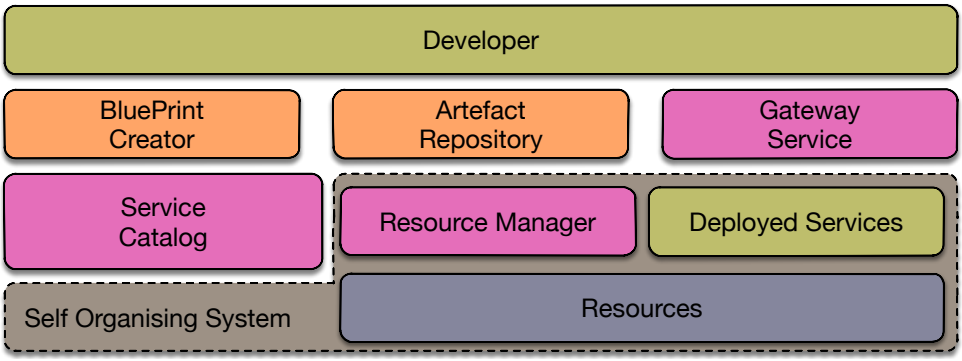

FIG. 3.1. *General overview of the CloudLightning components grouped as layers*

3.1. The layered architecture. The architecture of CloudLightning can be viewed as a three layered architecture, as depicted in Figure 3.1. Its first layer, which is represented by the *Developer*, is in charge of creating and designing application requirements that can be understood by the CloudLightning system by the means of the *CL-Blueprints*, as a representation of the CloudLightning *declarative approach to service provisioning*. Individualized blueprints are required for the deployment of different applications, and the *developer* will have to interact with components from the second layer of the architecture in order to provide such an individualization of blueprints.

The second layer of the architecture is constituted from a set of major components, namely the *Blueprint Creator*, *Gateway Service* and *Artefact Repository*, each having specific functionalities. The interactions from first level happens via the *Artefact Repository*, when a blueprint was already defined and its reuse is considered, or via the *Blueprint Creator*, when a new valid blueprint is required. In the latter case, the blueprint will be added to the repository prior to resource acquisition and application deployment. When an application is fully specified by its blueprint, it will be submitted to the *Gateway Service*, which acts as an entry point to the third layer of the architecture.

The *Self-organizing and Self-managing* level is at the core of the third level of CloudLightning architecture. At this level the system will trigger the self-organizing mechanisms to the adequate resources based on applications' requirements. Additionally, the system will provide some deployment options and will keep track of resource utilization and deployed applications, eventually by using additional repositories.

3.2. CloudLightning's architectural components. The *Blueprint* is at the heart of the CloudLightning architecture. It represents a formal description of the intended application, with some relevant annotations. Conceptually, a blueprint is usually a meaningful composition of several atomic services, a service being atomic if it cannot be decomposed into smaller services. However, a blueprint can also be imagined as a composition of both complex and atomic services.

From a technical point of view, inside CloudLightning a *Blueprint* is fully compatible OASIS TOSCA and fully supports Apache Brooklyn[1] blueprints, and includes information regarding topology, deployment and even scaling. Moreover, the *Blueprint* enables interactions between the different CloudLightning components, as they were identified in Figure 3.1, and drafted in Figure 3.2.

A sample blueprint description. Typical descriptions of CloudLightning blueprints are realized in YAML[2]. As a sample blueprint, we can consider the YAML description of a deployment scenario for a simple ray tracing application (see Section 2.3.3), which is composed of three basic components (as described in Listing 1):

- The ray tracing compute service, which will provide the required computing services;
- The controller for managing ray tracing services;
- A web portal, supporting client's access to computing services.

[1] `https://brooklyn.incubator.apache.org`
[2] `http://www.yaml.org/spec/1.2/spec.html`

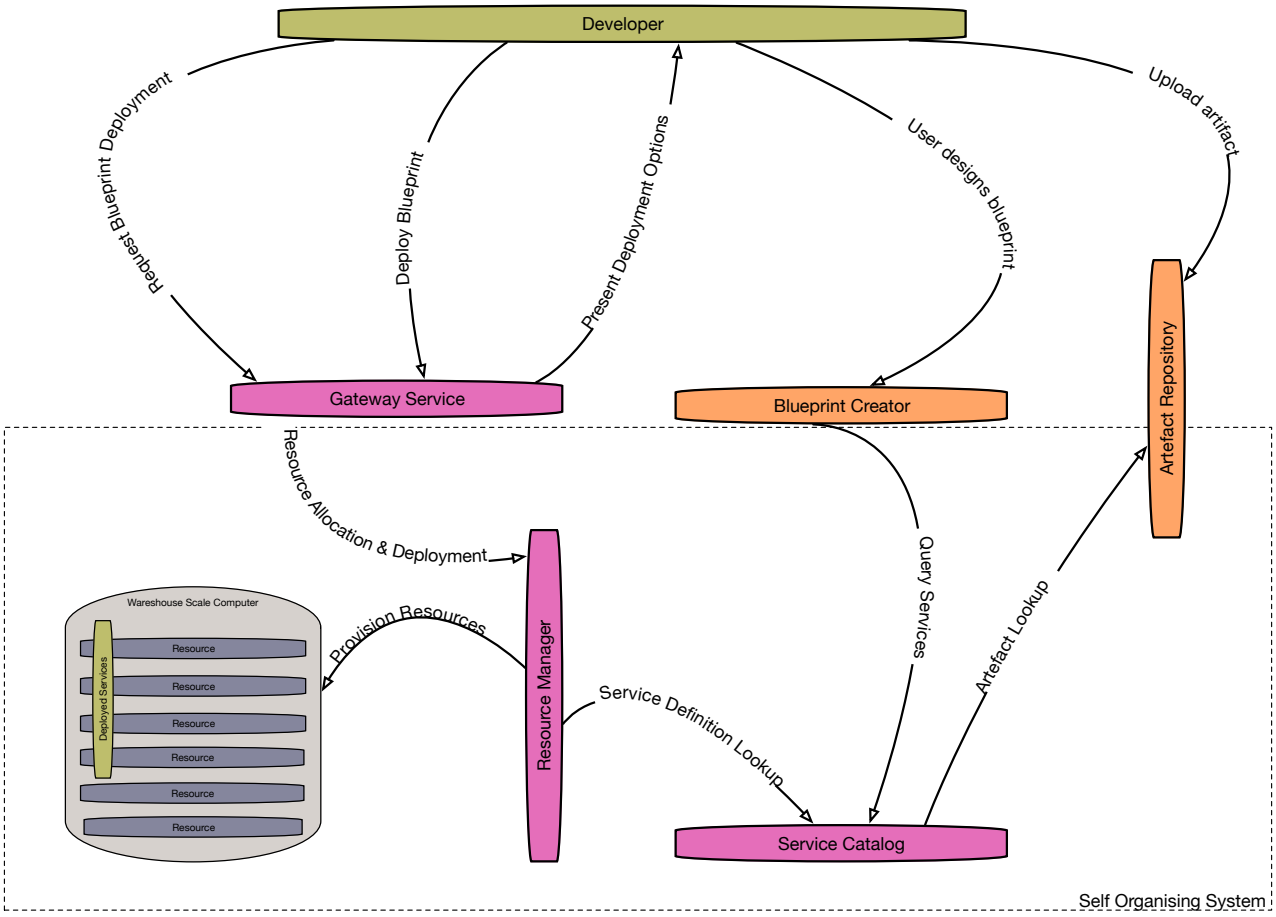

FIG. 3.2. *General overview of the CloudLightning components interactions*

For each of these components an identification, some Brooklyn configuration details, and a service type description is offered. The service type description will offer to the CloudLightning platform some additional information necessary for self-* activities, such as: the exact type of virtual machine or container that should be instantiated, and the configuration management rules to be applied.

Additionally, the blueprint specification offers the required details on components' relationships. In our sample blueprint, the *jetty_server* references the *rt_ctr* component which, in turn, references the *rt_cs* component. The latter component has some specific information: it has an abstract service type, which will serve as input for creating requests for tenders that are submitted by the service gateway to the self organizing meta-component of the CloudLightning system, thus triggering the autonomic capacities of the platform.

3.2.1. The Blueprint Creator. Acts as the component that empowers developers to easily create complex blueprints. Individual service description and definition is retrieved from a *Service Catalog* where they are defined. When some of the services needed are not defined, it provides the user with a concrete way of adding them to the *Service Catalog*. This component will also allow developers to load and reuse already defined blueprints from the previously identified *Artefact Repository*. One can view this component as being the liaison between the CloudLightning service description language (CL-SDL) and the internal components of the CloudLightning system.

3.2.2. The Service Catalog. Represents a core component of the CloudLightning architecture which is in charge of storing service definitions that can be eventually used for blueprint creation. This component is tightly connected with most of other major components of the CloudLightning architecture. Its major goal is

```
location: ieat-compute:timisoara
name: Sample Raytracing Service
services:
  - type: io.cl.entity.meta.RayTracingComputeService
    id: rt_cs
    brooklyn.config:
      cl.soso.min_performance: "100gflops"
      cl.config:
        hints:
          prefer: ['GPU', "FPGA", "CPU"]
  - type: io.cl.entity.meta.raytracing.controller
    id: rt_ctr
    brooklyn.config:
      lsf_cluster.head_node: $brooklyn:component("rt_cs")
  - type: io.cl.entity.java.jetty.web-portal-1
    id: jetty_server
    brooklyn.config:
      raytracing.controller: $brooklyn:component("rt_ctr")
```

Listing 1
Sample YAML description of a blueprint

to provide an easy to use component that keeps track of previously defined services that are made available for composing complex blueprints tailored for different applications. As previously specified, one can view the services as atomic components that can be interchanged between various users of the CloudLightning system. By doing so the process of creating custom blueprints for individual applications becomes a much easier task.

More precisely, in order to use a new service, the developer must add the atomic service(s) to the catalog such that they can be later used for blueprint creation. Additionally, the service catalog is used by the *Gateway Service* in order to decompose a blueprint in atomic services. The *Self Organizing System* will also interact with the *Service Catalog* in order to provide the *Resource Manager* with the formal description of the individual services.

3.2.3. The Resource Manager. This component is in charge with the management of the available underlying resources. Its responsibilities are related to resource reservation, provisioning, discovery and monitoring. The process of resource reservation is one of the core functionalities of the resource management system. Through this functionality, as soon as the *Self Organizing System* provides the user with a suitable set of resources to deploy its application the intended resources are reserved for this purpose and marked as being busy. An initial approach to the process of resource reservation is to reserve the resources as soon as they are presented to the user as a solution to the blueprint request. Besides reservation of resources, it is also in charge of discovering newly added resources to the infrastructure. The information about available and used resources are then aggregated and sent to the *Self Organizing System* so that they can be further used in the process of coalition formation for individual blueprints. During the deployment step, the *Resource Management* plays a crucial role by providing the means of interaction between physical resources and the CloudLightning system.

3.2.4. The Deployment Service. Represents the core component in charge of deploying an application on the reserved resources. Besides deployment of applications on readily provided coalitions the deployment service also takes care of monitoring the status of both the deployed application and of the underlying architecture. The collected monitoring information is aggregated and sent to the *Self Organizing System* where together with the information received from the *Resource Management* assists the underlying decision making of the Self Organizing System.

3.2.5. The Gateway Service. This component is in charge of creating the link between the outside world and the *Self Organizing System* from behind the scenes of CloudLightning. One of the core features that this component exposes is resource brokering and reservation. For achieving this the *Gateway Service* is exposing

a set of application programming interfaces (APIs) allowing the interaction between the outside world and the inner components of CloudLightning.

As a primary input the *Gateway Service* will receive a user defined Application Blueprint, based on which the Gateway service will call the *decomposition engine* in order to transform the user defined blueprint into a blueprint containing only atomic services. This step is required, as in the application Blueprint the developer may compose complex services in order to describe its application. The newly created blueprint is then sent to the Self Organizing System in order to provide with a coalition of resources meeting the requirements described in the decomposed blueprint. Once the *Self Organizing System* identifies the needed resources it will transmit them back to the Gateway Service which, in turn, provides them to the developer. Also, the Gateway Service provides a set of API's intended for controlling the deployed services and managing the commissioned resources.

3.2.6. The Self Organizing System. Finally, this system can be viewed as a meta-component which deals with the creation and management of coalitions where different applications run, and running at the core of the entire system. Being one of the core components of CloudLightning architecture, it is tightly coupled with other components in order to provide with adequate solutions for different application requests and it enables the decentralized self management approach and the self-organization mechanisms. The *Self Organizing System* communicates by the *Resource Manager* with individual resources in order to provide with answers to the independent application requests that arrive as blueprints in the system. It keeps a tight connection with the *Deployment* component so that it can integrate the information about already deployed services as respond to individual requests.

4. Conclusion and future work. We have shortly presented the approach and architecture of the CloudLightning system under development. The CloudLightning approach is built over a set of three central principles: (a) a declarative approach to service provisioning; (b) a decentralized, self-management approach; (c) integration of heterogeneous resources. The major components that are exposed by the CloudLightning architecture are constructed around the *Self Organizing System* and the CloudLightning service description language (CL-SDL), which offer important links with the other components of the system.

While the requirements and initial development of the CL-SDL were already delivered, the language is continually evolving to reflect the development of the *Gateway Service*. Additionally, it will support the core information required to build the various catalogues, as they are required by the CloudLightning system.

Acknowledgment. This work was partially funded by the European Union's Horizon 2020 Research and Innovation Programme through the CloudLightning action (`http://www.cloudlightning.eu`) under Grant Agreement Number 644869.

REFERENCES

[1] T. Becker, G. Gaydadjiev, P. Kuppuudaiyar, A.C. Elster, M.M. Khan, G. Gravvanis, C. Papadopoulos, T. Lynn, and D. Kenny. *D2.1.1: Use case requirements report.* Deliverable, CloudLightning Project Consortium, 2015.

[2] B.A. Caprarescu, N.M. Calcavecchia, E. Di Nitto, and D. J. Dubois. *SOS cloud: Self-organizing services in the cloud.* In Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pages 48–55. Springer Science + Business Media, 2012.

[3] J. Cave, N. Robinson, S. Kobzar, and H. R. Schindler. *Regulating the cloud: more, less or different regulation and competing agendas.* Less or Different Regulation and Competing Agendas (March 30, 2012), 2012.

[4] A. Chandra, J. Weissman, and B. Heintz. *Decentralized edge clouds.* IEEE Internet Computing, 17(5):70–73, September 2013.

[5] C. Delimitrou and C. Kozyrakis. *Hcloud: Resource-efficient provisioning in shared cloud systems.* SIGPLAN Not., 51(4):473–488, March 2016.

[6] O. Pell (editor). *D2.3: Validation plan.* Deliverable, HARNESS project, 2013.

[7] M.C. Huebscher and J.A. McCann. *A survey of autonomic computing: Degrees, models, and applications.* ACM Comput. Surv., 40(3):7:1–7:28, August 2008.

[8] A. Keller, T. Karras, I. Wald, T. Aila, S. Laine, J. Bikker, C. Gribble, W.-J. Lee, and J. McCombe. *Ray tracing is the future and ever will be... ACM SIGGRAPH 2013 Courses on - SIGGRAPH 13, 2013.*

[9] T. Lynn, H. Xiong, D. Dong, B. Momani, G. Gravvanis, C.F. Papadopoulos, A.C. Elster, M.M. Zaki Murtaza Khan, D. Tzovaras, K. Giannoutakis, D. Petcu, M. Neagul, I. Dragan, P. Kuppudayar, S. Natarajan, M. McGrath, G. Gaydadjiev, T. Becker, A. Gourinovitch, D. Kenny, and J. Morrison. *CLOUDLIGHTNING: A framework*

for a self-organising and self-managing heterogeneous cloud. In 6th International Conference on Cloud Computing and Services Science, CLOSER 2016, 2016.

[10] D.C. Marinescu, J.P. Morrison, and A. Paya. *Is cloud self-organization feasible?* In Adaptive Resource Management and Scheduling for Cloud Computing: Second International Workshop, ARMS-CC 2015, pages 119–127. Springer International Publishing, 2015.

[11] J. Morrison, H. Xiong, D. Dong, and B. Momani. *D3.1.2: Architecture.* Deliverable, CloudLightning Project Consortium, 2016.

[12] K. C. Nunna, F. Mehdipour, A. Trouvé, and K. J. Murakami. *A survey on big data processing infrastructure: evolving role of FPGA.* International Journal of Big Data Intelligence, 2(3):145, 2015.

[13] C. Pahl and B. Lee. *Containers and clusters for edge cloud architectures – a technology review.* In Proc. 3rd Int Future Internet of Things and Cloud (FiCloud) Conf, pages 379–386, August 2015.

[14] D. Palma and T. Spatzier. *Topology and orchestration specification for cloud applications* version 1.0, 2013.

[15] D. Pohl. *Experimental cloud-based ray tracing using intel mic architecture for highly parallel visual processing.* Intel Software Network Article, 21, 2011.

[16] E Rodrigues, JM Segura, P Vargas Mendoza, R Ausas, K Das, U Mello, MR Lakshmikantha, et al. *Exploring efficient alternatives for high performance computing requirements in coupled fluid-flow and stress simulations for the oil & gas industry.* In SPE Large Scale Computing and Big Data Challenges in Reservoir Simulation Conference and Exhibition. Society of Petroleum Engineers, 2014.

[17] Z.D. Stephens, S.Y. Lee, F. Faghri, R.H. Campbell, C. Zhai, M.J. Efron, R. Iyer, M.C. Schatz, S. Sinha, and G.E. Robinson. *Big data: Astronomical or genomical?* PLoS Biol, 13(7):e1002195, Jul 2015.

[18] Christopher G. Willard, Addison Snell, Sue Gouws Korn, and Laura Segervall. *Cloud computing in HPC: Barriers to adoption.* Research report, Intersect360 Research, 2011.

[19] H. Xiong, D. Dong, J. Morrison, I. Antoniadis, M. Neagul, K. Giannoutakis, T.-F. Fortiş, and I. Drăgan. *D5.1.1: Service description format specification.* Deliverable, CloudLightning Project Consortium, 2016.

Edited by: Viorel Negru
Received: June 15, 2016
Accepted: October 5, 2016