CHAPTER 5

Simulating Heterogeneous Clouds at Scale

Christos K. Filelis-Papadopoulos, Konstantinos M. Giannoutakis, George A. Gravvanis, Charalampos S. Kouzinopoulos, Antonios T. Makaratzis, and Dimitrios Tzovaras

Abstract In this chapter, a review of existing cloud simulation frameworks is given along with an overview of the recently proposed CloudLightning simulation framework. Moreover, the parallel architecture and parallel implementation details of the CloudLightning simulator are presented along with the characteristics of the supported cloud architectures. These architectures include the traditional centralised approach as well as the Self-Organised and Self-Managed CloudLightning approach. The supported memory, network, and application execution models are

C. K. Filelis-Papadopoulos (✉) • G. A. Gravvanis
Democritus University of Thrace, Komotini, Greece
e-mail: cpapad@ee.duth.gr; ggravvan@ee.duth.gr

K. M. Giannoutakis • C. S. Kouzinopoulos • A. T. Makaratzis • D. Tzovaras
Centre for Research and Technology Hellas, Thessaloniki, Greece
e-mail: kgiannou@iti.gr; kouzinopoulos@iti.gr; antomaka@iti.gr; Dimitrios.Tzovaras@iti.gr

T. Lynn et al. (eds.), *Heterogeneity, High Performance Computing, Self-Organization and the Cloud*, Palgrave Studies in Digital Business & Enabling Technologies,
https://doi.org/10.1007/978-3-319-76038-4_5

reviewed. Furthermore, a recently proposed class of power models for heterogeneous CPU-Accelerator-based hardware is discussed. Finally, large-scale simulations for traditional and Self-Organised and Self-Managed cloud environments are presented and compared.

Keywords CloudLightning simulator • Self-organisation • Self-management • Scalability • Large-scale simulations

5.1 Introduction

Cloud simulation tools have been extensively used for the analysis of cloud data centres, since the cost of experimentation using various scenarios is low. A number of different aspects, related to cloud environments, can be studied through simulation including resource allocation strategies, live migration of running applications to more efficient data centre resources, energy consumption, and hardware resource utilisation. Several cloud simulation tools have been developed during the past few years focusing on different aspects of cloud environments. These tools can be categorised into:

- *Discrete Event Simulators (DES)*: These examine macro-scale phenomena, such as application events that take place in certain moments in time while completely disregarding micro-scale phenomena, including network packet communication. DES are used to examine large-scale simulations, while focus is given among others in the study of cloud environments behaviour in terms of service delivery, Virtual Machine (VM) allocation policies, utilisation of resources, and the energy consumption of data centres.
- *Packet-Level Simulators (PLS)*: These examine micro-scale phenomena related to cloud environments, including packet loss and network communication protocols. PLS offer high levels of accuracy at the cost of performance though, since large-scale data centres cannot be studied due to the restricting resolution of the simulations.

Cloud infrastructures continue to grow in both size and diversity to cater for demand in terms of both user and data volumes and the variety of hardware resources. As a result, existing cloud simulation tools cannot be used to efficiently simulate these heterogeneous environments at scales several orders of magnitude greater than traditional data centres. By 2020, hyperscale data centres will account for a substantial portion of all cloud workloads and data (Cisco 2016). Furthermore, as hyperscale data centres consist of servers in distinct geographical locations, the efficient management of such infrastructures is made more difficult resulting in network congestion and underutilisation of resources. Resource heterogeneity further exacerbates these challenges. While hyperscale data centre operators increasingly offer specialised hardware, such as Graphical Processing Units (GPUs), Many Integrated Cores (MICs), and Field-Programmable Gate Arrays (FPGAs), existing cloud simulation tools do not support them. The efficient exploitation of the hardware infrastructure of heterogeneous hyperscale cloud environments is a topic of great importance during the last few years; thus, cloud simulation tools for studying heterogeneous cloud environments that can cater for hyperscale need to be developed.

The remainder of this chapter is organised as follows. Section 5.2 provides a summary review of common cloud simulation frameworks used by the scientific community and their limitations. A new simulation framework, the CloudLightning Simulator, designed to simulate hyperscale cloud environments composed of heterogeneous resources is presented in Sect. 5.3. This is followed by a discussion of initial experimentation using the CloudLightning Simulator to compare service delivery of three application scenarios: oil and gas exploration, ray tracing, and genomics, using (i) conventional cloud service delivery and (ii) cloud service delivery using a self-organising self-managing (SOSM) approach.

5.2 Cloud Simulation Frameworks

During the last decade, various cloud simulation frameworks have been proposed, such as CloudSim (Calheiros et al. 2011), DCSim (Tighe et al. 2012), GDCSim (Gupta et al. 2011), GreenCloud (Kliazovich et al. 2012), iCanCloud (Nunez et al. 2012), and CloudSched (Tian et al. 2015). However, no existing cloud simulation framework is designed for hyperscale simulations.

One of the main limitations of existing cloud simulation tools is the lack of scalability. Most existing cloud simulation tools do not support parallelism; thus, the simulation of very large data centres is not possible (Byrne et al. 2017). Parallelism is of great importance for the simulation of hyperscale cloud environments since both computational work and memory requirements can be distributed among multiple nodes, reducing the execution time significantly and enabling the simulation of large-scale data centres.

An important factor influencing scalability of the extant simulation tools is memory requirements. In DES a large number of events should be created and retained. The number of these events is closely related to the number of resources simulated as well as the input tasks. Discrete Event based simulators initialise the task list that will be executed for the whole simulation and augment it gradually with new events according to time. This process requires retaining a very large list in memory, its augmentation with new events, and its sorting in order to perform events in the correct order. Thus, memory requirements increase significantly with the number of resources or the simulation time. Memory restrictions also occur due to the high level of detail of the simulated components, such as in the case of the iCanCloud and GreenCloud frameworks, which becomes prohibiting in very large-scale executions.

The effective management of resources is a significant challenge as their number increases. More specifically, strategies which require the detection of specific hardware cannot be applied or require significant computational cost when hyperscale systems are considered. Also, status information corresponding to the underlying hardware resources is becoming outdated, and thus efficient management of the system becomes more challenging. Specialised strategies are required in hyperscale cloud environments for the efficient and up-to-date management of the system. Such strategies are not supported in existing simulation frameworks, and thus the simulation of hyperscale systems is difficult to perform.

Finally, the inclusion of heterogeneous resources is not supported by existing cloud simulation tools. Simple generic models are required for the simulation of heterogeneous resources in order to be integrated in cloud simulation environments (Makaratzis et al. 2017; Giannoutakis et al. 2017).

5.3 CloudLightning Simulator

Unlike existing frameworks, the CloudLightning Simulator has been designed from the ground up as a massively scalable solution, able to simulate hyperscale data centres consisting of millions of cloud nodes/servers.

The framework is written in C++ and is parallelised using Message Passing Interface (MPI) (Gropp et al. 1996) and OpenMP (Dagum and Menon 1998) to enable the efficient handling of hyperscale simulations. CloudLightning supports the simulation of heterogeneous infrastructures (including GPUs, MICs, and FPGAs/DFEs) that are commonly used for the acceleration of High Performance Computing applications. One important characteristic of the developed framework is the use of a time-advancing loop, a technique that removes the need for pre-computation and storage of future events, resulting in a significant reduction of its memory requirements. This allows the integration of dynamic resource allocation policies, such as SOSM, enabling the efficient management of computer resources for simulating hyperscale environments. Moreover, the CloudLightning Simulator places an emphasis on the simplicity of the models it uses, focusing on models that require reduced number of computations for producing the results of the simulations without loss of accuracy. Finally, all inputs and outputs of the simulator are represented graphically.

The remainder of this section presents the generalised and extensible CloudLightning simulation framework for simulating heterogeneous resources using an SOSM approach.

5.3.1 Architecture and Basic Characteristics of the Parallel CloudLightning Simulation Framework

The CloudLightning Simulator was designed to simulate clouds relying on the Warehouse Scale Computer (WSC) architecture (Barroso et al. 2013). WSC has been adopted by a multitude of companies including Google, Amazon, Yahoo, Microsoft, and Apple, and has been widely used in the design of cloud environments (Mars 2012). In the WSC architecture, interconnected cloud computing nodes are grouped into cells that are centrally managed (Fig. 5.1).

In this architecture, the Gateway service is responsible for redirecting end user requests to the appropriate Cells. The Gateway service is the entry point of the system and is a cloud entity that receives resource requests from the end users and redirects them to the Cells. A conceptual cloud architecture with multiple Cells is presented in Fig. 5.2. The resources are organised and monitored by the Cell manager's broker that is responsible for the provision of appropriate resources to end user

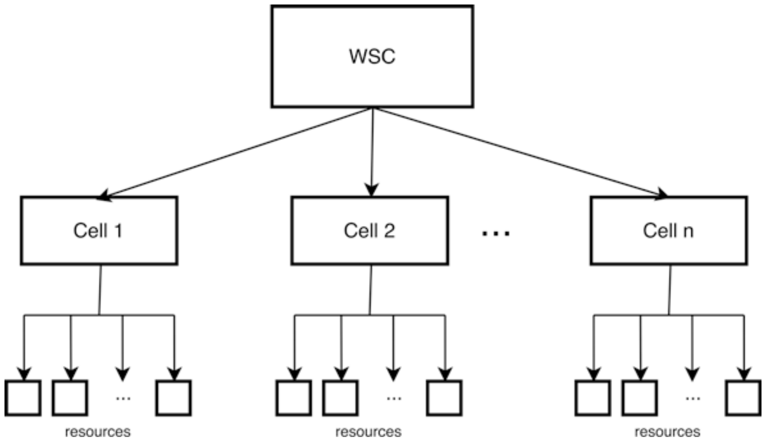

Fig. 5.1 Warehouse Scale Computer abstract architecture

requests and for the deployment of incoming tasks to the available resources. The broker component is composed of multiple services, including orchestration, telemetry, and identity service. Hyperscale cloud environments consist of a considerably large number of Cells.

In the CloudLightning simulation framework, each Cell is hosted on a different computing node of a distributed system, while the Gateway service is hosted on the master computing node. The communication between the Gateway service and the Cells is performed using the MPI framework. The following operations are performed by each Cell (Filelis-Papadopoulos et al. 2017, b):

- Receiving simulation parameters
- Initialisation of different components, including hardware resources, the broker, network, telemetry, and the SOSM engine
- Receiving the task queue in each time-step
- Searching for available resources for the execution of the tasks, using the SOSM engine
- Updating the state of the resources and controlling the execution of the tasks
- Communicating status information to the Gateway Service

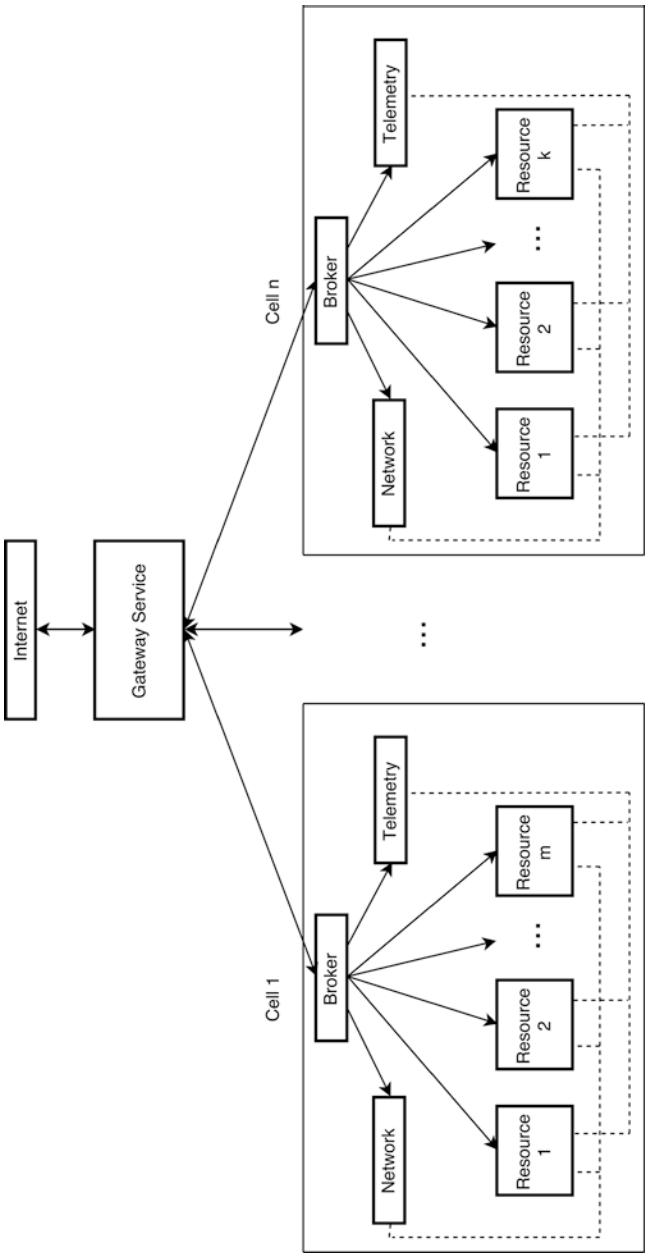

Fig. 5.2 Abstract cloud architecture with multiple cells

The operations performed by the Gateway service are the following (Filelis-Papadopoulos et al. 2017, b):

- Retaining simulation inputs and communicating data to the Cells for the initialisation of the simulation components
- Creation of the task queue in each time moment, fragmentation of the task queue into subqueues, and communication of the subqueues to the Cells, by maintaining load balance through all Cells
- Receiving status information from the Cells
- Processing and storing historical statistics and metrics

The parallelisation of the CloudLightning Simulator in distributed systems is of great importance, since simulating hyperscale infrastructures is a computationally and memory-intensive process. For this reason, various components of the CloudLightning Simulator use the OpenMP framework in different ways to accelerate their computations on shared memory multiprocessors. The Gateway Service processes statistics in parallel—the Cells perform resource discovery and task deployment as well as the update of the resources' state on different multiprocessor cores. The SOSM techniques are also performed in parallel.

Figure 5.3 presents the software architecture of the CloudLightning Simulator (Filelis-Papadopoulos et al. 2017):

5.3.2 SOSM Engine

One of the most important characteristics of the CloudLightning Simulator is the use of SOSM techniques to control the underlying resources of the Cells in a more efficient manner (Filelis-Papadopoulos et al. 2017).

In traditional cloud architectures, the resources are managed by the broker, a central entity that is responsible for the search and deployment of the available resources with respect to incoming task requests, the collection of data for the state of all underlying resources, and the management of all underlying resources of the data centre. This centralised approach has limitations due to the computational complexity involved in locating specific hardware, especially when the number of resources increases. Locating the most appropriate server for the execution of a task is a computationally expensive operation in large data centres, and it is generally avoided in favour of strategies such as the "first-fit approach," where a task is deployed on the first available server or coalition of servers.

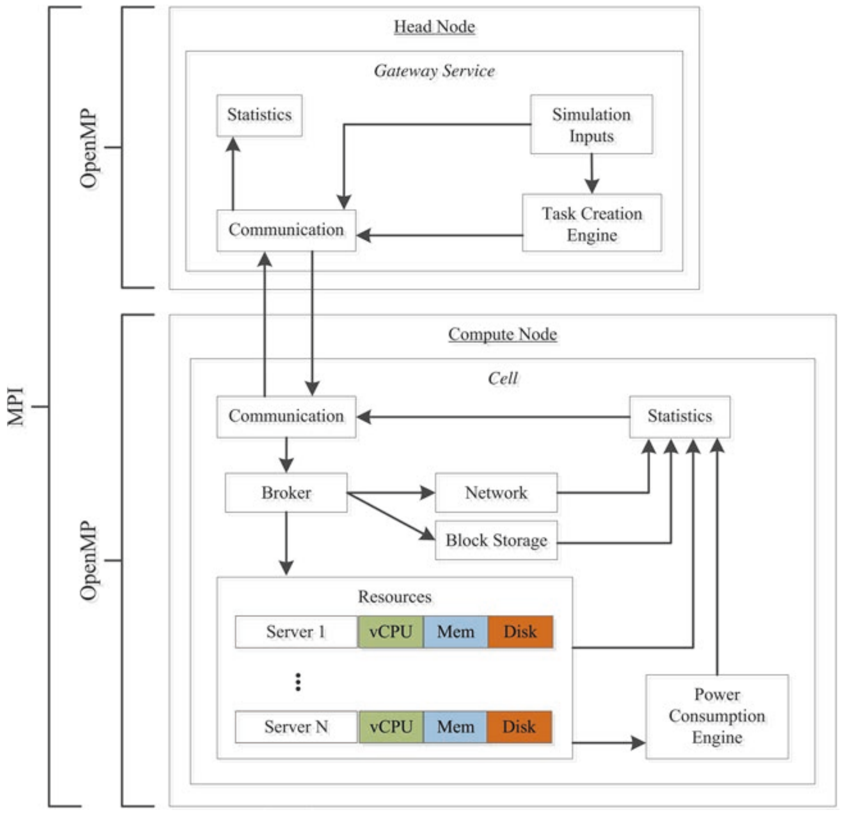

Fig. 5.3 Software architecture of the parallel CloudLightning simulation framework

This type of strategy is not effective though in terms of both computational and energy efficiency, resulting largely in the underutilisation of the available resources (Filelis-Papadopoulos et al. 2017). More effective strategies, such as SOSM, need to be applied to achieve high levels of resource utilisation and thus computational and energy efficiency.

In the CloudLightning architecture, each Cell is organised in a hierarchical tree structure. As discussed earlier, the tree contains different entities, including prescription Routers (pRouters), prescription Switches (pSwitches), and virtual Rack Managers (vRMs). Figure 5.4 presents an example of the CloudLightning tree structure. In this structure, the

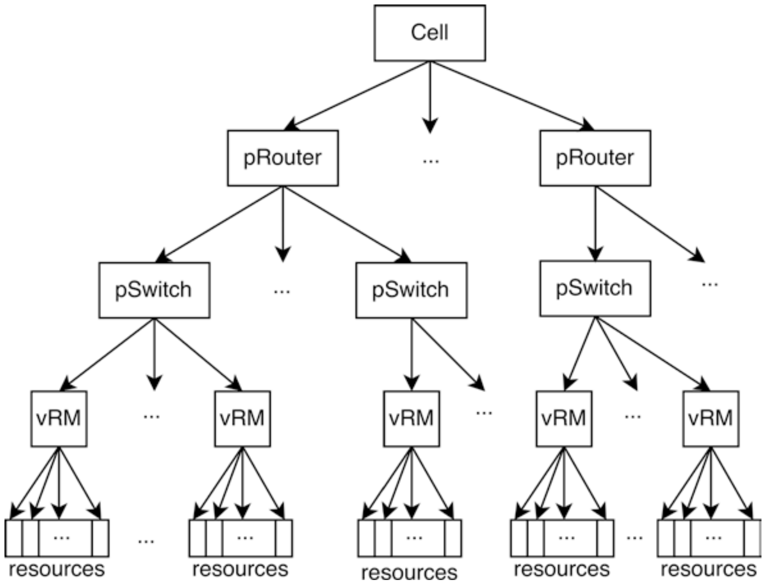

Fig. 5.4 Hierarchical structure of the SOSM engine

resources are locally managed by the vRMs which in turn are locally managed by the pSwitches, while the pSwitches are locally managed by the pRouters. The local management of the architectural components allows the efficient collection and analysis of data that can lead to an improved decision-making process. Each component can describe the state of its underlying resources since metrics describing the state of the resources are collected with respect to an interval and averaged by each component to form its own state. Also, weights describing the desired state of the system are communicated from the Gateway Service to the underlying components. By using these metrics and weights, each component's Suitability Index is computed. The Suitability Index expresses how appropriate is a component to receive an incoming task. By using the Suitability Index, each incoming task can be subsequently directed to the most efficient resources.

The exchange of metrics and weights between the components is part of the Self-Management actions and is performed by all the components of the SOSM engine. The Self-Organisation techniques, on the other hand, are solely performed by the vRMs and the pSwitches. In the case of vRMs, there can be an exchange of resources between vRMs that are hosted by the same pSwitch, in order to maximise the efficiency of the system and to host tasks that require more resources than available on a vRM. New vRMs can also be created, while vRMs that do not contain any resources to manage can be destroyed. Similarly, pSwitches that are hosted by the same pRouter can exchange vRMs; new pSwitches can be created, while existing pSwitches can be dismissed when they have no vRMs to manage.

Each pRouter of a Cell is homogeneous, as it contains resources of the same type. In order to maintain the homogeneity, Self-Organising actions are not performed at the pRouter level; thus, pSwitches cannot be exchanged between pRouters. For this reason, pRouters are the entry point for the selection of a specific type of resource inside a Cell (Filelis-Papadopoulos et al. 2017).

The SOSM system improves significantly the scalability of cloud environments since the most appropriate hardware for the execution of a task can be located fast and with low computational cost, even in data centres with a very large number of resources. In the CloudLightning Simulator, the SOSM engine is implemented in parallel using the OpenMP framework.

5.3.2.1 Power Consumption Modelling

To estimate the power consumption of large-scale heterogeneous data centres, a number of different power models for both Central Processing Unit (CPU) servers and combined CPU-accelerator pairs were developed. The power models are generic with low computational cost (Filelis-Papadopoulos et al. 2017; Giannoutakis et al. 2017). For this reason, the CloudLightning Simulator is capable of computing the power consumption of very large heterogeneous data centres without a significant impact on its scalability. The following subsection gives a detailed presentation of the integrated power consumption models.

CPU Power Models

Piecewise interpolation methods between recorded CPU power consumption levels, and generic models that estimate the trend of the power-utilisation diagram of CPUs by using the idle and maximum power consumption of the CPU servers, have been integrated.

The interpolation methods are performed between recorded CPU power consumption levels that are available mainly as part of the

Standard Performance Evaluation Corporation (SPEC) benchmark (SPEC 2008). Existing simulators, such as CloudSim, use linear interpolation between power measurements on rounded utilisation intervals (i.e. 0%, 10%, 20%, etc.) (Beloglazov and Buyya 2012). In order to achieve improved accuracy, the interpolation methods in the CloudLightning Simulator are applied on the exact utilisation intervals of the power measurements (i.e. 0%, 10.2%, 19.7%, etc.) as the error of the rounded interpolation intervals increases when simulating very large data centres (Giannoutakis et al. 2017). Two different interpolation methods were used, the linear and the “not-a-knot” cubic spline interpolation.

Generic models were also integrated, since they require less computational cost and power measurements compared to the interpolation methods. The models estimate the power consumption of CPU servers by using the utilisation of the CPU server and its power consumption in idle and max states. The linear, square, cubic, and square root models that have been used in existing cloud simulators (i.e. CloudSim) were integrated (Beloglazov and Buyya 2012). For the CloudLightning Simulator, a generic CPU power model was used based on a third-degree polynomial, which estimates more accurately the trend of the power-utilisation diagram of CPU servers (Filelis-Papadopoulos et al. 2017). The trend of the generic models compared with the actual CPU measurements provided by SPEC (SPEC 2008) for an HP Proliant DL560 Gen 9[1] is presented in Fig. 5.5.

Existing cloud simulators (i.e. GreenCloud and CloudSim) support the use of real application traces in order to compute the power consumption of the simulated applications in each time-step. This approach would negatively affect the scalability of the simulator in large-scale simulations, and for this reason, mean values of real application traces were computed and integrated. More specifically, the mean value of the CPU utilisation for each application is used to compute the mean power consumption of the application. Then, the energy consumption of the application is computed by multiplying the mean power consumption of the application with its execution time. This approach provides a lower computational cost, while the result of the energy consumption of the application is computed with approximately the same accuracy that would have been obtained if all the power traces were used. This methodology has been tested, achieving high levels of accuracy in the estimation of the energy consumption of applications (Makaratzis et al. 2017).

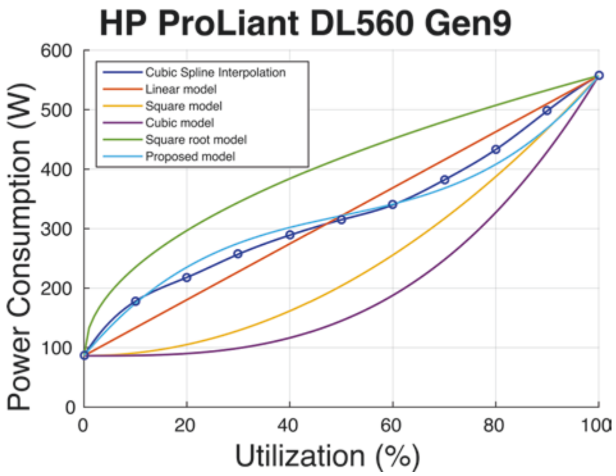

Fig. 5.5 Generic CPU power models compared to the power-utilisation diagram of an HP Proliant DL560 Gen 9 server

Combined CPU-Accelerator Power Models

A generic power consumption model was used for the estimation of the power consumption of accelerators such as GPUs, MICs, and DFE (Giannoutakis et al. 2017). This model was built around the idea that the maximum power consumption of an accelerator is consumed when an application is executed on the accelerator, while the idle power consumption is consumed when the application is executed only on the CPU. This binary model provides simplicity and increased accuracy (Makaratzis et al. 2017). The model for the power consumption of hardware accelerators is described as follows:

$$P_{acc}(\rho) = (1-\rho)P_{acc-\min} + \rho P_{acc-\max}$$

where $P_{acc\text{ - }min}$ and $P_{acc\text{ – }max}$ are the minimum and maximum power consumption values, respectively, that the application can consume on the accelerator, while ρ is the percentage of the application that is parallelised on the accelerator, thus in each time moment. Similarly, with the utilisation parameters of the CPU power model, the mean value of parameter is computed based on real application traces, thus the mean value of the power that is consumed on the accelerator is computed for the total execution time of the application. The combined CPU-accelerator mean power consumption of the application is computed as the sum of the mean power consumption of the CPU server and the mean power consumption of the accelerator. The energy consumption of an application that is executed on a heterogeneous node is computed by multiplying the combined CPU-accelerator mean power consumption with the execution time of the application.

To conclude, in order to keep the computational cost low, generic CPU and accelerator power models were integrated in the CloudLightning Simulator. The simplicity of the models is of great importance since models that are based on architectural details of the hardware resources require a substantial number of computations, considering the heterogeneity and the very large number of resources in the simulations. These models were validated on heterogeneous testbeds and a good accuracy level was achieved (Makaratzis et al. 2017).

5.3.2.2 Memory, Storage, and Network Modelling

Detailed modelling of memory would negatively affect the scalability of the simulator, especially in large-scale simulations, since it would require an increased amount of computations. Memory was implemented as a resource, measured in GBytes, that is used in the allocation of VMs to physical servers. Memory overcommitment was also implemented; thus, the total available memory was computed as the product of the total physical memory and the overcommitment ratio. The power consumption of memory was included in the power consumption of the CPU servers, eliminating the need for a separate memory power consumption calculation.

The modelling of storage was also implemented with simplicity in order to keep the computational cost in low levels. The storage was implemented as a resource measured in TBytes. Global storage was not implemented, though its impact can be added directly to the time span of tasks. Detailed modelling of the power consumption of storage was not implemented since it would require substantially large number of computations, which would negatively affect the scalability of the simulator. The energy consumption of storage is considered to be included in the energy consumption of the CPU servers, similar to memory modelling.

The network was implemented as a global component, visible from all the underlying resources, with the network bandwidth being shared among the arriving tasks of the system. When the requested network bandwidth exceeds the available capacity, the execution of applications is affected negatively (in terms of the execution time). It should be noted that the network model of the CloudLightning Simulator was implemented through a catalogue of tasks, retaining all tasks executing at a given time-step. A linear model for computing the time required to transfer initial data and output data was implemented with a function of the following form:

$$NT(t) = fileSize / bandwidth$$

where *fileSize* is the size of the file to be transferred and *bandwidth* is the available physical bandwidth.

5.3.2.3 Application Models

In the design of the CloudLightning Simulator, the execution of VMs is part of a given task and their life cycle is directly connected to it. Each task is defined based on the following characteristics (Filelis-Papadopoulos et al. 2017):

- Type of application (Genomics, Oil and Gas, Ray Tracing)
- Available implementations (CPU-only, CPU+GPU, CPU+DFE, CPU+MIC)
- Number of instructions (in Millions of Instructions [MIs])
- Required number of VMs
- Required number of processing units per VM
- Required memory per VM (in GBytes)
- Required storage per VM (in TBytes)
- Required accelerators per VM
- Required network bandwidth

The minimum and maximum values are defined for the actual utilisation of the CPU, the memory, and the network. The actual resources used by an application (utilisation) are computed based on application traces as a percentage of the requested resources over a number of predefined intervals. These utilisation parameters are considered as mean values with respect to the total execution time of the application. This approach maintains the computational cost low, while the desired metrics are obtained with the same accuracy that would have been obtained if all the application traces were used.

All task parameters, including the number of instructions, the required number of VMs, and memory size, are randomly generated using a uniform random number generator with respect to predefined intervals. The intervals are computed based on real application characteristics.

This approach of application modelling reduces computational cost, allowing for large-scale simulations, while also providing realistic results during the simulations.

5.3.2.4 Execution Models

Existing cloud simulators generally create *a priori* task lists for the whole duration of the simulation, augment, and sort that list with respect to events triggered by inputs and so on. However, this has the disadvantage of simulation data storage, not only for the current event but also for future ones, restricting the execution of large-scale simulations over long time periods. In contrast, the CloudLightning Simulator is based on a time-advancing loop, where incoming tasks are created dynamically in each time-step and where each time-step is independent from any previous or future ones (Filelis-Papadopoulos et al. 2017). A task list is then created at the beginning of each time-step, removing the need for data storage of future tasks of the simulation. Creating task lists per time-step reduces significantly the memory requirements of the simulation and offers the ability to simulate dynamical components that change their state according to dynamic strategies, including pRouters, pSwitches, and vRMs while allowing for the simulation over extended time periods.

In the execution of tasks, the time-step is used as the control mechanism of the execution. The performance of applications is measured in MIs while the computational capability of the physical servers is measured in Millions of Instructions per Second (MIPS). In each time-step, the number of instructions that can be executed by the available resources is subtracted from the total number of instructions of the application. This time-step-controlled execution model offers significant capabilities since the impact of various phenomena can be modelled by applying penalties on the execution of tasks. For example, phenomena such as performance degradation due to cache sharing or "noisy-neighbours" can be modelled by reducing the computational capability, meaning that fewer of the application's instructions will be executed on the current time-step. Similarly, the usage of hardware with a higher computational capability, that is, accelerators, can be modelled by increasing the computational capability of the current time-step. Service-level Agreement violations concerning memory, storage, or network limitations can be modelled by applying similar penalties in the execution of tasks.

This approach of execution modelling allows the integration of possible extensions on the simulator, since any phenomenon can be modelled during a simulation by applying penalties or gains in the execution of the applications. Also, this execution model allows the simulation of very large time periods and millions of cloud servers, since the memory requirements of the execution model are very low.

5.4 Experimental Results

This section presents the experimentation framework and the numerical results occurred after simulating the traditional cloud delivery system and the SOSM framework.

The experiments were performed on a cluster consisting of four Dell PowerEdge C4130 nodes, each containing two 10-core Intel Xeon E5-2630 v4 CPUs running at 2.20 GHz (3.10 GHz Max Turbo frequency) with 128GB of Random Access Memory (RAM), and a Dell PowerEdge R730 node containing two 8-core Intel Xeon E5-2609 v4 CPUs running at 1.70 GHz. During the simulation, the Dell PowerEdge R730 node was used to host the Gateway service, while the 4 Dell PowerEdge C4130 nodes were used to host the Cells.

The time period of the simulation was set to one week (604,800 seconds), with a time-step of 1 second. The update interval of the Gateway Service was chosen to be 200 seconds, while the update interval of the pRouters, pSwitches, and vRMs was 20 seconds. The cloud nodes of the simulated data centre were selected to use an Intel Xeon E5-2699 v4 2.20 GHz-based node with 44 cores and 385,063.42 MIPS, 128 GBytes of RAM, and 40 TBytes of storage.

Each Cell consisted of four different types of hardware, that is, CPUs+GPUs, CPUs+MICs, CPUs+DFEs, or CPU servers with no accelerators. Each heterogeneous node consisted of a CPU and four accelerators. The characteristics of the CPUs and the accelerators are presented in Table 5.1. It is noted that the linear interpolation method on uneven utilisation intervals was used for the estimation of the power consumption of the CPU servers, where the power values for the various utilisation intervals were obtained[2] from SPEC (SPEC 2008).

Table 5.1 Selected simulation framework characteristics (adapted from Byrne et al. 2017)

Simulation platform	*Licence*	*Language(s)*	*Platform portability*	*Distributed architecture*	*Model persistence type*	*Web API availability*	*GUI availability*	*Headless execution*	*Result output format*
CloudSched	–	Java	Yes	No	Text	No	Yes	No	XLS, Text
CloudSim	Apache 2	Java	Yes	No	YAML	No	No	Yes	Text
DCSim	GPL 3	Java	Yes	No	Java classes	No	No	Yes	Text
GDCSim	GPL 2	C/C++, Shell	No	No	C code	No	No	Yes	Text
GreenCloud	GPL	C++, TCL, JS, CSS, Shell	No	No	TCL	Yes	Yes	Yes	Dashboard plots
iCanCloud	GPL 3, GNU, Academic	C/C++, Shell	Yes	No	NED	No	Yes	Yes	Text

During the simulations, three different types of applications were considered. The characteristics of the applications are presented in Tables 5.2 and 5.3.

The CloudLightning Simulator was executed for different number of resources, Cells, and submitted tasks. Each Cell was hosted on a Dell PowerEdge C4130 node, while in the experiments with eight Cells, each computing node was hosting two Cells. Three different configurations were tested. In the first configuration, 11,000 resources per Cell were utilised, while the experiment was performed for different number of Cells. Similarly, in the second configuration, 110,000 resources per Cell were used, and in the third configuration, 1,100,000 resources per Cell were considered. The maximum number of submitted tasks was set equal to four per second when one Cell was used, while this number was multiplied with the number of Cells when additional Cells were used. The VM allocation policy used was the "first-fit approach," according to which tasks are placed on the first available server found.

Table 5.2 Hardware characteristics

Hardware	*MIPS*	*Idle power consumption (Watts)*	*Max power consumption (Watts)*
CPU	385,063.4268	44.9	269.0
MIC	1,347,721.9938	30.0	350.0
DFE	2,310,380.5608	70.0	100.0
GPU	1,155,190.2804	50.0	400.0

Table 5.3 Application characteristics

Application type:	*1*	*2*	*3*
Millions of instructions	1386.23–5544.91	462.08–2772.46	693.11–4158.69
Number of VMs	1–16	1–8	1–4
Number of vCPUs	4–8	8–16	4–8
Memory (GBytes)	4–8	4–8	4–8
Storage (TBytes)	0.02–0.04	0.01–0.02	0.04–0.08
Network bandwidth (MBps)	2.5–5	0.5–1	2.5–5
Network storage (GB)	0–0	0–0	0–0
Implementations	1, 2, 3	1, 2, 3	1, 4
ρ	0, 0.7, 0.5	0, 0.8, 0.9	0, 0.9

Table 5.4 presents the outputs, in terms of the number of accepted tasks, the average processor and accelerator utilisation, the average network utilisation, the energy consumption of the data centre, and the execution time of the CloudLightning Simulator, simulating a traditional centralised cloud service delivery system.

For all different configurations, it can be observed that the total number of rejected tasks was high, with an ~86% task rejection on average. The task rejection was caused mainly by the network congestion appearing early in the simulated cloud (Fig. 5.6). Despite the fact that the selection of applications and their corresponding implementations (Table 5.3) were performed randomly using a uniform random generator, accelerator implementations were starting to be rejected after a period of simulated time, since the network resources are shared between the resources hosted across a Cell. This yields the acceptance of additional CPU tasks that in general require more computational time for execution and consequently overload the network.

The energy consumption estimation of the cloud infrastructure increased with the number of resources per Cell and the number of Cells. It is expected that, except from the idle servers that consume the minimum power, when the utilisation of the cloud increases, the energy consumption will proportionally increase.

The CloudLightning Simulator was also tested using the SOSM resource allocation framework, for 100 resources per vRM, 10 vRMs per pSwitch, and 5 pSwitches per pRouter. The VM allocation policy was "Task Compaction," where the system is provisioning as many VMs as possible on each physical server. Table 5.5 presents the outputs of the CloudLightning Simulator, in terms of the number of accepted tasks, the average processor and accelerator utilisation, the average network utilisation, the energy consumption of the data centre, and the execution time of the simulator, when using the SOSM engine.

During the SOSM resource allocation simulation, it can be observed that there was a more balanced utilisation between CPUs and accelerators. More specifically, accelerators tended to be utilised at the same levels as CPUs, while in many cases, their utilisation percentages overcame the corresponding CPU ones. This was due to the fact that the system (SOSM framework) decides the resources (and types of implementations) to be allocated for a task, according to the predefined assessment functions, that targets on (a) improved service delivery, (b) computational efficiency, (c) improved energy consumption, and (d) efficient management of

Table 5.4 Execution of the CloudLightning simulator for different number of resources, different number of Cells, and different number of tasks for the traditional centralised cloud service delivery system

Configuration	*Cells*	*Submitted tasks*	*Accepted tasks*	*Average processor utilisation (over active servers)*	*Average accelerator utilisation (over active servers)*	*Average network utilisation (%)*	*Energy consumption (GWh)*	*Execution time (sec)*
Configuration 1: 11,000 resources per Cell	1	2,419,200	136,734	34.15% (89.94%)	1.51% (72.83%)	99.46	0.41434	2037.57
	2	4,838,400	770,219	46.25% (94.23%)	4.78% (64.93%)	96.95	1.25828	2565.49
	3	7,257,600	1,363,382	43.19% (90.71%)	4.48% (56.68%)	98.78	1.72955	2691.67
	4	9,676,800	2,446,410	45.82% (91.59%)	7.19% 77.86%	96.7	2.55623	3004.94
	8	19,353,600	3,301,122	44.05% (91.2%)	4.48% (70.72%)	98.18	4.40216	2833.70
Configuration 2: 110,000 resources per Cell	1	2,419,200	136,578	3.42% (90.03%)	0.15% (71.53%)	99.47	2.15033	5102.79
	2	4,838,400	420,451	5.29% (86.86%)	0.16% (62.15%)	99.39	4.5139	6197.07
	3	7,257,600	1,096,134	4.36% (89.58%)	0.35% (55.5%)	99.26	6.83921	7560.64
	4	9,676,800	1,990,111	7.03% (90.06%)	0.45% (74.8%)	98.85	9.63639	7599.38
	8	19,353,600	2,583,260	4.67% (87.11%)	0.31% (69.21%)	99.07	18.03786	7465.97

(*continued*)

Table 5.4 (continued)

Configuration	*Cells*	*Submitted tasks*	*Accepted tasks*	*Average processor utilisation (over active servers)*	*Average accelerator utilisation (over active servers)*	*Average network utilisation (%)*	*Energy consumption (GWh)*	*Execution time (sec)*
Configuration 3: 1,100,000 resources per Cell	1	2,419,200	136,462	0.34% (89.96%)	0.02% (70.99%)	99.46	19.50804	41,319.31
	2	4,838,400	419,313	0.53% (86.61%)	0.02% (61.93%)	99.39	39.22957	40,754.94
	3	7,257,600	1,095,999	0.44% (89.09%)	0.03% (55.4%)	99.26	58.91502	40,747.46
	4	9,676,800	1,989,893	0.71% (90.05%)	0.05% (75.46%)	98.85	79.07197	44,777.33
	8	19,353,600	2,585,701	0.47% (87.06%)	0.03% (69.36%)	99.07	156.90569	43,971.90

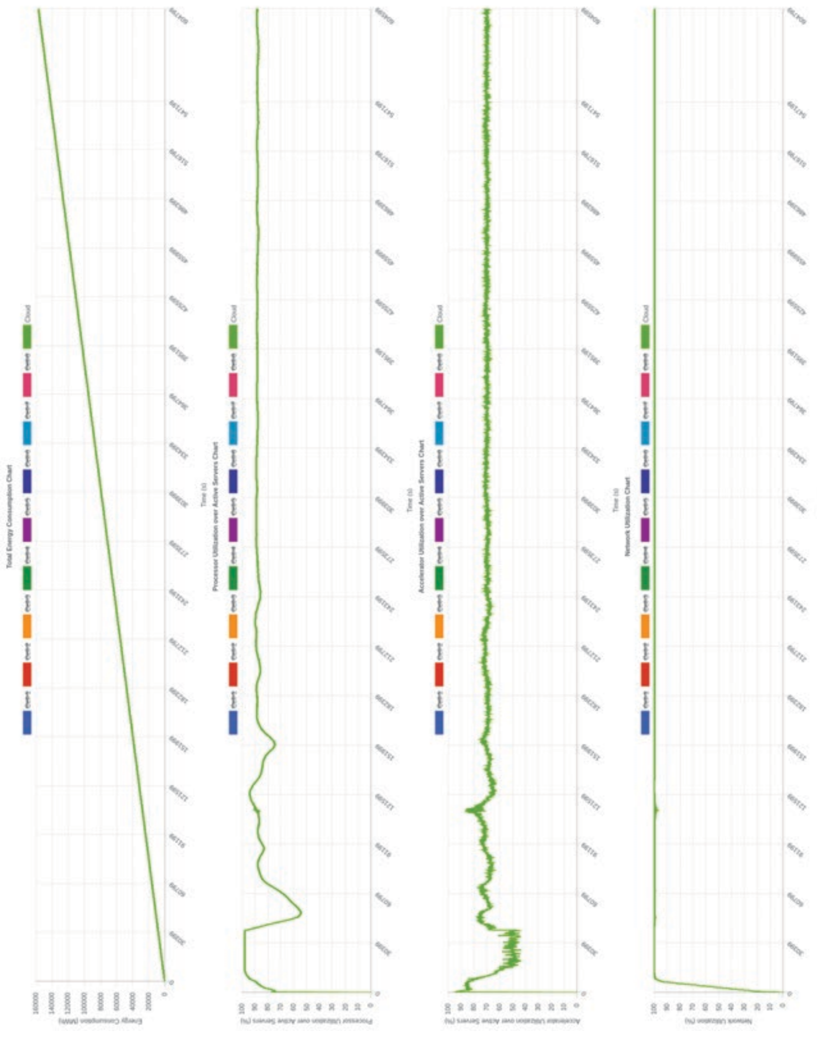

Fig. 5.6 Energy consumption, processor, accelerator, and network utilisation of the cloud over time for the traditional cloud delivery simulation

Table 5.5 Execution of the CloudLightning simulator for different number of resources, different number of Cells, and different number of tasks by using SOSM engine

Configuration	*Cells*	*Submitted tasks*	*Accepted tasks*	*Average processor utilisation (over active servers)*	*Average accelerator utilisation (over active servers)*	*Average network utilisation (%)*	*Energy consumption (GWh)*	*Execution time (sec)*
Configuration 1: 11,000 resources per Cell	1	2,419,200	2,407,079	33.86% (78.38%)	34.19% (89.86%)	90.52	1.25502	1870.03
	2	4,838,400	4,838,400	33.23% (79.61%)	30.68% (79.19%)	50.51	2.60836	1704.74
	3	7,257,600	7,257,600	23.64% (72.77%)	26.3% (78.33%)	37.4	3.26585	1776.24
	4	9,676,800	9,676,800	35.19% (77.15%)	34.24% (86.47)	31.54	5.48662	1866.63
	8	19,353,600	19,314,339	50.62% (79.25%)	32.36% (84.5%)	58.11	10.68841	2772.45
Configuration 2: 110,000 resources per Cell	1	2,419,200	2,419,200	1.35% (59.01%)	3.52% (83.91%)	19.65	2.82534	4110.54
	2	4,838,400	4,838,400	1.48% (63.73%)	3.08% (72.52%)	14.22	5.78658	4198.38
	3	7,257,600	7,257,600	1.12% (57.6%)	2.63% (73.35%)	16.22	8.11747	4322.67
	4	9,676,800	9,676,800	1.39% (57.92%)	3.49% (79.06%)	17.44	11.93687	4906.20
	8	19,353,600	19,353,600	1.3% (56.16%)	3.34% (78.81%)	17.63	23.13896	5043.66

(*continued*)

Table 5.5 (continued)

Configuration	*Cells*	*Submitted tasks*	*Accepted tasks*	*Average processor utilisation (over active servers)*	*Average accelerator utilisation (over active servers)*	*Average network utilisation (%)*	*Energy consumption (GWh)*	*Execution time (sec)*
Configuration 3: 1,100,000 resources per Cell	1	2,419,200	2,419,200	0.13% (52.58%)	0.35% (75.92%)	19.82	20.14322	42,716.98
	2	4,838,400	4,838,400	0.14% (58.84%)	0.3% (68.39%)	14.18	40.37729	49,864.84
	3	7,257,600	7,257,600	0.11% (47.56%)	0.26% (61.58%)	16.22	60.06821	46,116.78
	4	9,676,800	9,676,800	0.14% (50.34%)	0.35% (69.99%)	17.68	81.01169	45,414.01
	8	19,353,600	19,353,600	0.13% (48.79%)	0.34% (70.09%)	17.89	161.36398	47,184.41

underlying resources. Since accelerators are more efficient in terms of computational efficiency and energy consumption, the system's choice is apparent.

It can also be seen that the total number of rejected tasks was very low (~0.05%), but the total estimated energy consumption of the cloud was close to the estimations of the traditional delivery system, due to the utilisation of the energy-efficient accelerators. Thus, the SOSM- based cloud environment was able to execute more tasks consuming almost equal energy. This was expected, since the SOSM selects the most efficient resources, executing the task faster, thus freeing those resources faster, and consequently leading to more tasks being accepted.

In order to examine the energy efficiency of the two resource allocation techniques in more detail, the ratio of the total energy consumption of the data centre over the number of accepted tasks was computed for all experiments. In Table 5.6, the number of Wh that is consumed per task for all configurations is presented. It can be observed that the number of Wh per task is substantially smaller when the SOSM engine is used. This is due to the fact that when the SOSM engine is not used, the resources that are utilised are selected randomly, while with the SOSM engine the resources

Table 5.6 Ratio of the total energy consumption of the cloud over the number of accepted tasks for all configurations

Configuration	*Cells*	*Wh per task without SOSM*	*Wh per task with SOSM*
Configuration 1: 11,000 resources per Cell	1	3030.26314	521.38713
	2	1633.66523	539.09557
	3	1268.57330	449.99035
	4	1044.89027	566.98702
	8	1333.53448	553.39248
Configuration 2: 110,000 resources per Cell	1	15,744.33657	1167.88194
	2	10,735.85269	1195.96974
	3	6239.39226	1118.47856
	4	4842.13695	1233.55551
	8	6982.59563	1195.58945
Configuration 3: 1,100,000 resources per Cell	1	142,955.84119	8326.39716
	2	93,556.77024	8345.17402
	3	53,754.62934	8276.59419
	4	39,736.79489	8371.74376
	8	60,682.07036	8337.67258

are selected by the system, according to the predefined strategies; thus, the most energy efficient solution is always chosen.

In Figs. 5.6 and 5.7, time-dependent charts are presented for the last experiment of the third configuration (eight Cells, 1,100,000 servers per Cell). In Fig. 5.6, the energy consumption, the processor utilisation, the accelerator utilisation, and the network utilisation of the cloud are presented with respect to the simulated time for the traditional centralised cloud service delivery. In Fig. 5.7, the energy consumption, the processor utilisation, the accelerator utilisation, and the network utilisation of the cloud are presented through the simulation time when using the SOSM engine.

5.5 Conclusion

This chapter presented the work towards demonstrating the scalability of the CloudLightning simulation framework. Cloud simulation tools are examined, since demonstrating scalability in hyperscale clouds is unfeasible. The design and implementation of the CloudLightning simulation framework were presented, a framework that overcomes the limitations of the existing simulation platforms. The main innovations of the framework lie in the fact that it is implemented for parallel computing systems (using MPI and OpenMP), it is based on a time-advancing loop instead of a discrete sequence of events, it allows the integration of dynamic resource allocation systems such as SOSM, and it supports hybrid CPU-accelerator resources. Finally, the CloudLightning Simulator was developed to be easily extensible, since the time-advancing execution model allows the integration of any strategies or phenomena observed in cloud environments.

From the experiments that were performed, the CloudLightning simulator was found to be capable of simulating clouds with large number of resources. Different executions were performed with the traditional cloud delivery system and with the use of the SOSM framework, for a various number of resources and Cells. Both the simulation platform and the SOSM framework were found to be scalable; simulations up to 8,800,000 hardware resources grouped into eight Cells were performed, only limited by the available hardware used for experimentation. SOSM was found to provide a more balanced distribution of tasks on the available hardware resources, with a much lower number of total rejected tasks. The energy consumption was found to be equivalent to the energy consumed when simulating a traditional cloud delivery system; however, the SOSM system

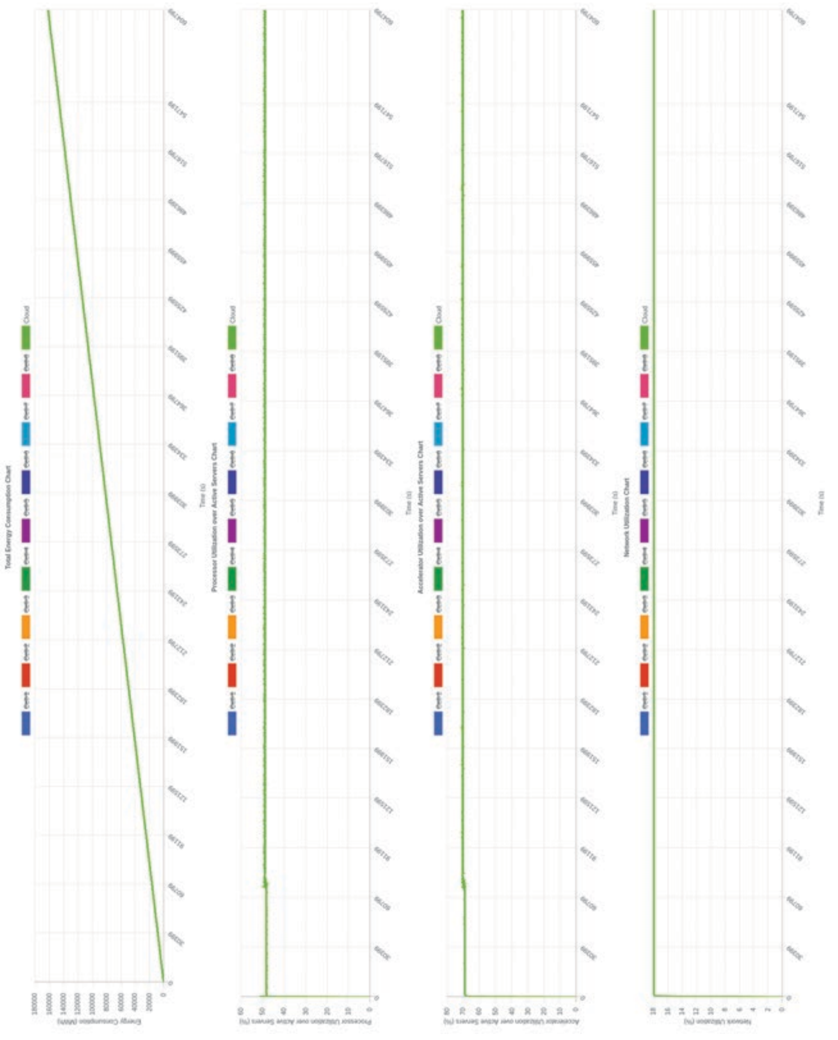

Fig. 5.7 Energy consumption, processor, accelerator, and network utilisation of the cloud over time for the SOSM simulation

was able to service a significantly larger number of tasks. Thus, the energy consumed per task in the SOSM system was substantially reduced compared to the traditional approach.

The CloudLightning Simulator and Simulator Visualization Tool are available for download under the Apache 2 open source licence at https://bitbucket.org/cloudlightning/cloudlightning-simulator and https://bitbucket.org/cloudlightning/cl-simulatorvisualization, respectively.

5.6 Chapter 5 Related CloudLightning Readings

1. Byrne, J., Svorobej, S., Giannoutakis, K., Tzovaras, D., Byrne, P. J., Östberg, P. O., et al. (2017). A review of cloud computing simulation platforms and related environments. In *Proceedings of the 7th International Conference on Cloud Computing and Services Science (CLOSER 2017)* (pp. 679–691). SCITEPRESS-Science and Technology Publications, Lda.
2. Filelis-Papadopoulos, C. K., Gravvanis, G. A., & Kyziropoulos, P. E. (2017). A framework for simulating large scale cloud infrastructures. *Future Generation Computer Systems.* https://doi.org/10.1016/j.future.2017.06.017
3. Filelis-Papadopoulos, C. K., Gravvanis, G. A., & Morrison, J. P. (2017). CloudLightning simulation and evaluation roadmap. In *Proceedings of the 1st International Workshop on Next Generation of Cloud Architectures*, Vol. 2. ACM.
4. Filelis-Papadopoulos, C. K., Grylonakis, E. N. G., Kyziropoulos, P. E., Gravvanis, G. A., & Morrison, J. P. (2016). Characterization of hardware in self-managing self-organizing Cloud environment. In *Proceedings of the 20th Pan-Hellenic Conference on Informatics*, Vol. 56. ACM.
5. Filelis-Papadopoulos, C. K., Giannoutakis, K. M., & Gravvanis, G. A. (2017). Large-scale simulation of a self-organizing self-management cloud computing framework. *The Journal of Supercomputing*. https://doi.org/10.1007/s11227-017-2143-2
6. Giannoutakis, K. M., Makaratzis, A. T., Tzovaras, D., Filelis-Papadopoulos, C. K., & Gravvanis, G. A. (2017, April). On the power consumption modeling for the simulation of Heterogeneous HPC Clouds. In *Proceedings of the 1st International Workshop on Next Generation of Cloud Architectures*, Vol. 1. ACM.

7. Lynn, T., Gourinovitch, A., Byrne, J., Byrne, P. J., Svorobej, S., Giannoutakis, K., et al. (2017). A preliminary systematic review of computer science literature on cloud computing research using Open Source simulation platforms. In *Proceedings of the 7th International Conference on Cloud Computing and Services Science (CLOSER 2017)* (pp. 537–545). SCITEPRESS-Science and Technology Publications, Lda.
8. Makaratzis, A. T., Giannoutakis, K. M., & Tzovaras, D. (2017). Energy modeling in cloud simulation frameworks. *Future Generation Computer Systems*. https://doi.org/10.1016/j.future.2017.06.016

Notes

1. https://www.spec.org/power_ssj2008/results/res2016q2/power_ssj2008-20160607-00734.html
2. http://spec.org/power_ssj2008/results/res2016q2/power_ssj2008-20160328-00719.html

References

Barroso, L. A., Clidaras, J., & Hoelzle, U. (2013). *The datacenter as a computer: An introduction to the design of warehouse-scale machines*. Morgan & Claypool. https://doi.org/10.2200/S00516ED2V01Y201306CAC024

Beloglazov, A., & Buyya, R. (2012). Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice & Experience, 24*(13), 1397–1420. https://doi.org/10.1002/cpe.1867

Byrne, J., Svorobej, S., Giannoutakis, K. M., Tzovaras, D., Byrne, P., Ostberg, P. O., et al. (2017). A review of cloud computing simulation platforms and related environments. In *The 7th International Conference on Cloud Computing and Services Science* (pp. 651–663).

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., & Buyya, R. (2011). CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience, 41*(1), 23–50. https://doi.org/10.1002/spe.995

Cisco Global Cloud Index: Forecast and Methodology, 2015–2020. (2016). Retrieved from https://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.pdf

Dagum, L., & Menon, R. (1998). OpenMP: An industry standard API for shared memory programming. *IEEE Computational Science and Engineering, 5*(1), 46–55. https://doi.org/10.1109/99.660313

Filelis-Papadopoulos, C., Xiong, H., Spataru, A., Castañe, G., Dapeng, D., Gravvanis, G., et al. (2017). A generic framework supporting self-organisation and self-management in hierarchical systems. In *Proceedings of the International Symposium on Parallel and Distributed Computing*.

Filelis-Papadopoulos, C. K., Giannoutakis, K. M., Gravvanis, G. A., & Tzovaras, D. (2017). Large-scale simulation of a self-organizing self-management cloud computing framework. *The Journal of Supercomputing*. https://doi.org/10.1007/s11227-017-2143-2

Filelis-Papadopoulos, C. K., Gravvanis, G. A., & Kyziropoulos, P. E. (2017). A framework for simulating large scale cloud infrastructures. *Future Generation Computer Systems*. https://doi.org/10.1016/j.future.2017.06.017

Filelis-Papadopoulos, C. K., Gravvanis, G. A., & Morrison, J. P. (2017). CloudLightning simulation and evaluation roadmap. In *Proceedings of the 1st International Workshop on Next Generation of Cloud Architectures* (pp. 2:1–2:6). New York, NY: ACM. https://doi.org/10.1145/3068126.3068128

Giannoutakis, K. M., Makaratzis, A. T., Tzovaras, D., Filelis-Papadopoulos, C. K., & Gravvanis, G. A. (2017). On the power consumption modeling for the simulation of heterogeneous HPC clouds. In *Proceedings of the 1st International Workshop on Next Generation of Cloud Architectures* (pp. 1:1–1:6). New York, NY: ACM. https://doi.org/10.1145/3068126.3068127

Gropp, W., Lusk, E., Doss, N., & Skjellum, A. (1996). A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing, 22*(6), 789–828. https://doi.org/10.1016/0167-8191(96)00024-5

Gupta, S. K. S., Gilbert, R. R., Banerjee, A., Abbasi, Z., Mukherjee, T., & Varsamopoulos, G. (2011). GDCSim: A tool for analyzing green data center design and resource management techniques. In *2011 International Green Computing Conference and Workshops* (pp. 1–8). https://doi.org/10.1109/IGCC.2011.6008612

Kliazovich, D., Bouvry, P., & Khan, S. U. (2012). GreenCloud: A packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing, 62*(3), 1263–1283. https://doi.org/10.1007/s11227-010-0504-1

Makaratzis, A., Khan, M., Giannoutakis, K., Elster, A., & Tzovaras, D. (2017). GPU power modeling of HPC applications for the simulation of heterogeneous clouds. In *International Conference on Parallel Processing and Applied Mathematics*.

Mars, J. (2012). *Rethinking the architecture of warehouse-scale computers* (Doctoral dissertation, University of Virginia). https://doi.org/10.18130/V30N5R

Nunez, A., Vazquez-Poletti, J. L., Caminero, A. C., Castañe, G. G., Carretero, J., & Llorente, I. M. (2012). iCanCloud: A flexible and scalable cloud infrastructure simulator. *Journal of Grid Computing, 10*(1), 185–209. https://doi.org/10.1007/s10723-012-9208-5

SPEC. (2008). Standard performance evaluation corporation, server power and performance characteristics. Retrieved from http://www.spec.org/powerssj2008/

Tian, W., Zhao, Y., Xu, M., Zhong, Y., & Sun, X. (2015). A toolkit for modeling and simulation of real-time virtual machine allocation in a cloud data center. *IEEE Transactions on Automation Science and Engineering, 12*(1), 153–161. https://doi.org/10.1109/TASE.2013.2266338

Tighe, M., Keller, G., Bauer, M., & Lutfiyya, H. (2012). DCSim: A data centre simulation tool for evaluating dynamic virtualized resource management. In *2012 8th International Conference on Network and Service Management (CNSM) and 2012 Workshop on Systems Virtualization Management (SVM)* (pp. 385–392).