Hardware Accelerators in Data Centers

Springer

Chapter 1
Energy Efficient Servers and Cloud

Huanhuan Xiong, Christos Filelis-Papadopoulos, Dapeng Dong, Gabriel G. Castañé, Stefan Meyer, John P. Morrison

Abstract For clouds to be successful they need to continue to provide affordable, reliable, computing and storage services. They must continue to adapt to an ever growing user community whose service needs are unpredictable and are rapidly changing. They must embrace heterogeneous hardware as they endeavour to provide specialist services. Furthermore, as they struggle to come to grips with energy usage, they must enact strategies to minimise energy consumption per unit of service.

1.1 Introduction

As the sizes of cloud infrastructures continue to grow, the complexity of the cloud is becoming more and more difficult to manage. Currently, centralised management schemes dominate and there are already signs that these are no longer fit for purpose. Elasticity, for example, (the ability of the cloud to respond to rapidly changing demands for resources) is currently being supported by over-provisioning. Over-provisioning is a strategy of effectively under-utilising hardware so that some is always available to absorb unpredictable peaks in demand. This strategy is not sustainable, since the infrastructure costs and the energy it consumes, even when idle, are significant. In 2010 Gartner Research [14] reported that the average server utilisation in large data-centres is 18%, while the utilisation of x86 servers is even lower at 12%. These results confirmed earlier estimations that the average server utilisation is in the range of 10% - 30% [8]. Subsequent studies have not contradicted these finding [19, 16].

Cloud computing is evolving from it homogeneous roots and is being seriously regarded by once highly specialised application domains like High Performance Computing (HPC).

To support this trend, heterogeneity is a must [11]. HPC technology trends in coprocessors are on the increase and NVIDIA GPGPUs and Intel Xeon Phi are gaining increased traction. Low power processors are beginning to find their place in the HPC ecosystem and HPC public cloud revenue could range from $1.56 billion (low forecast) to $3.7 billion (high forecast) by 2017 and for HPC public custom cloud computing, worldwide revenue could range from $0.87 billion (low forecast) to $1.5 billion (high forecast) in the same period.

Incorporating energy efficient heterogeneous resources help to address power consumption in the cloud. However, their inclusion also adds to the complexity of the already overburdened cloud management scheme and this must be explicitly addressed. Energy efficiency in cloud computing can be considered as a complex optimisation problem, which attempts to minimise power consumption while satisfy Quality-of-Service (QoS) or service level agreement (SLA) requirements specified by users. Energy-aware resource provisioning and allocation is a way to improve energy efficiency without violating the negotiated SLAs or application performance [3, 4, 15]. Research shows that the objective of energy efficiency is not an independent or stand-alone issue from other cloud resource management objectives, such as QoS/SLA, resource utilisation, and workload performance (e.g., execution time, intensity), for example, and, unfortunately, there would appear to be no single equation capable of expressing all the inter-dependences between the multiple objectives.

The CloudLightning Project takes a novel route, making use of self-organisation techniques to address the problems emerging from the confluence of issues in the emerging cloud: rising complexity and energy costs, problems of management and efficiency of use, the need to efficiently deploy services to a growing community of non-specialist users and the need to facilitate solutions based on heterogeneous components. Thus, this approach attempts to address:

- Energy efficiency.
  Self-organisation is a powerful tool for addressing complexity of large- to hyper-scale cloud resource management. It has proven itself time and time again in nature and has been applied successfully in complex engineering projects [12]. Of self-organisation, Alan Turing once observed that global order arises from local interactions. We contend that when self-organisation is applied to self-management, local interactions can give rise to scalable global management. Moreover, given the appropriate evolutionary stimuli, the resultant global management can be optimised for specific characteristics. CloudLightning proposes a self-organised self-managed (SOSM) framework for providing energy-efficient cloud resource provisioning and allocation (see Section 1.3).
- Improved accessibility to cloud.
  The CloudLightning SOSM system provides cloud service consumers with a user-friendly service level interface to explicitly declare their requirements for service delivery. Through the assembly of dynamic resource coalitions, the

SOSM system automatically and intelligently locates the required resources and chooses the most appropriate configuration to deliver that service, while respecting both the user-level SLA and the business objectives of the cloud service providers (CSP). CSPs are thus enabled to provide energy-efficient, scalable management of their cloud infrastructures and better overall utilisation of service.

- Supporting heterogeneity.
  To support heterogeneity, CloudLightning brings various coprocessors into existing homogeneous cloud environments. These include Graphic Processing Units - GPUs, Many Integrated Cores - MICs, Field Programmable Gate Arrays - FPGAs. The availability of different resource types can alter the way that solutions are designed. Mapping a problem onto an architecture specifically designed with that problem in mind can greatly improve efficiency and simplify implementation. Secondary benefits are often obtained, such as speed, improved precision of solution and reduced power consumption. These are important drivers for both the cloud provider and for the end-user. CloudLightning also provides a plug-in mechanism for incorporating heterogeneous resources into the self-organising cloud. Pertinent characteristics of these resources are surfaced, through the plug-in mechanism, to the end-user via the service description language; making these resources easier to consume.

The three objectives listed above are tightly coupled aspects of the CloudLightning system. A complete description of the CloudLightning system is necessary to express the subtle interplay between the objectives and the architecture of the solution needed to address them. However, here the CloudLightning system is described predominantly from the energy efficiency perspective and the advantages that flow from exploiting hardware accelerators and the challenges associated with balancing energy consumption with improved service delivery.

The remainder of this chapter is organised as follows. Section 1.2 presents the CloudLightning hierarchical architecture and its main components. The self-organised, self-managed, framework with respect to energy efficient resource management is described in Section 1.3. Finally, Section 1.4 presents the evaluation of our proposed approach and Section 1.5 concludes with some final thoughts.

1.2 CloudLightning Architecture

Large-scale data-centers typically make use of a hierarchical model for organising the compute, storage and network infrastructures. The Warehouse Scale Computer (WSC) [2] is a typical hierarchical architecture widely used by companies like Google, Yahoo, Amazon, Facebook, Microsoft and Apple [1, 9, 17] for this purpose.

The CloudLightning architecture is also a hierarchical organisation of physical infrastructure but unlike traditional organisations it makes use of a resource management framework that is locically hierarchical. The bottom layer of this framework hierarchy consists of many resource managers. These managers are autonomous

and, in contrast to traditional systems, each manages a relatively small number of physical resources. Since the number of physical resources is restricted, each manager can efficiently control the collection - allocating tasks and, where appropriate, virtualising resources in response to service requests. This arrangement is self-limiting in the sense that an increase in the number of physical resources attached to such a manager spontaneously results in a new manager being brought into existence to assume the control over the extra resources should the increase in physical resources exceed a specific cost management threshold. In this way, a first step is taken to tackle the problem of scalability in resource allocation. However, this enhancement to the topology of the hierarchy forms only a partial solution, since no mechanism is yet provided to identifying an appropriate resource manager, from many potential candidates, that will make the final resource allocation decision.

The CloudLightning architecture is depicted in Figure 1.1.

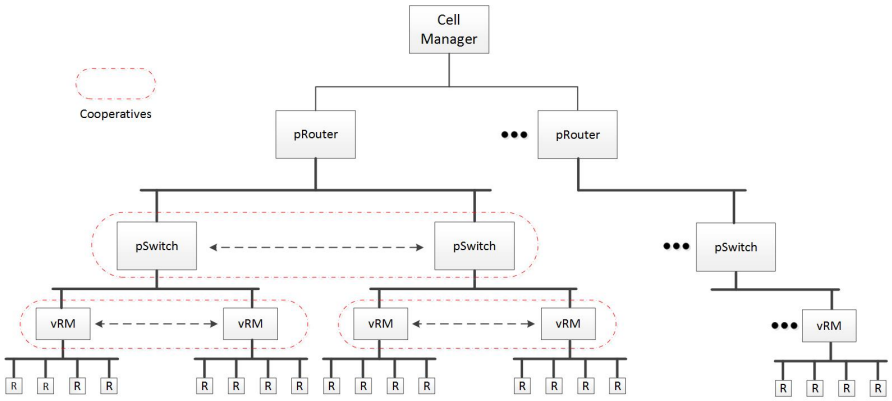

Fig. 1.1: A Representation of the CloudLightning Architecture

1.2.1 Cell

At the top of the hierarchy, a Cell represents the entire set of physical resources. These are partitioned into different hardware types (including CPUs, CPU-GPU pairs, CPU-MIC pairs, and CPU-FPGA pairs) and each partition is accessed via a dedicated pRouter.

1.2.2 pRouter

Each pRouter provides access to hardware resources of the same type which, in turn, is managed by a specific resource abstraction method (such as OpenStack Nova [1] to manage virtual machines on commodity servers, Kubernetes [2], Mesos [3] [10], and/or Docker Swarm [4] to manage containers on GPUs and MICs, and OpenStack Ironic [5] to manage bare metal servers using DFEs).

Some examples of the constitution of the pRouters in CloudLightning architecture include:

- OpenStack-CPUs-VMs: Commodity machines (CPUs) pre-installed with OpenStack services.
- OpenStack-FPGAs-accelerators: FPGAs configured as compute accelerators using the OpenStack Nova service.
- Mesos-GPUs-accelerators: GPUs configured as compute accelerators using the Mesos framework and Docker Engine.
- Marathon-MICs-accelerators: Marathon as the Resource Manager managing a cluster of Xeon servers with attached MICs.

Each pRouter is connected to one or more pSwitches.

1.2.3 pSwitch

pSwitches are used to further partition the resource space into smaller and more manageable domains composed of multiple virtual Rack Managers (vRMs). However, the number of pSwitches per pRouter is not fixed and can change over time. Similarly, the number of vRMs, being managed by each pSwitch, can also change over time in response to dynamic grow and shrinkage of the resource fabric.

pSwitches and vRMs can self-organise by exchanging constituent members. Thus, two of more, pSwitches may exchange control over a subset of their respective vRMs and , similarly, twowithin groups, which will be called Cooperatives, to emphasise their self-organising nature. To prohibit the creation of Cooperatives with different resource types, pSwitch Cooperatives cannot span pRouters. Similarly, to minimise administrative overhead, vRM Cooperatives cannot span pSwitches.

As the CloudLightning system evolves, it is anticipated that the number of pSwitches connected to a pRouter will change and will converge to some optimal number with respect to some global goal set by the cloud service provider. This goal

[1] OpenStack Nova: http://docs.openstack.org/developer/nova/

[2] Kubernetes: http://kubernetes.io/

[3] Apache Mesos: http://mesos.apache.org/

[4] Docker Swarm: https://github.com/docker/swarm/

[5] OpenStack Ironic: http://docs.openstack.org/developer/ironic/deploy/user-guide.html

is expressed as a vector of weights that are propoagated down through the management hierarchy and alter the perceived importance of the underlying behaviours. As part of the self-organisation process, pSwitches and vRMs can be created, destroyed, merged and split.

1.2.4 vRack Manager

At the bottom of the hierarchy are a collection of resource managers knows as the virtual Rack Managers (vRMs). These are responsible for the efficient management of collection of resources directly under their control. vRMs also communicate weighted status information pertaining to theese resources upwards through the hierarchy.

It is anticipated that the number of vRMs connected to a pSwitch and the number of pSwitches connected to a pRouter will change and will converge to some optimal number; derived from the weights coming from the pRouter and from the vRMs and pSwitch's efforts to convergy to a local goal state. As part of the self-organisation process, vRMs and pSwitches can be created, destroyed, merged and split. Furthermore, they may exchange control over resources in an effort to maximise resource utilisation, to minimise energy consumption and to optimise management utility.

1.3 Hyperscale Resource Management for Energy Efficiency

In the CloudLightning system, the process of identifying an appropriate resource manager to affect the next resource allocation decision is distributed throughout the entire logical hierarchy. It begins at the vRM level, where information relating to the functional capabilities, and the nonfunctional behaviors, of its constituent resources forms a view of these resources, which is then propagated upwards through the hierarchy. In this upwards propagation, and at each intermediate level in the hierarchy, this information may be combined into a higher-level view, in many different ways, with similar information emerging from different elements from the lower-level of the hierarchy. These views are called *Perceptions* and are used to guide resource allocation requests entering the system at the top of the hierarchy. The contention is that the most appropriate resource allocation is to be found by following the path exhibiting the greatest perception, since this path simultaneously maximises the chances of locating the requisite resources and of optimising the non-functional behaviours.

CloudLightning develops a strategic self-organised self-managed framework to support distributed resource allocation decisions and that can be dynamically populated with strategies to reflect the ever-growing number of diverse objectives as they become evident in the evolving cloud infrastructure. In the CLoudLightning approach, cloud service providers can define various strategies by which cloud re-

sources are allocated, and system/cloud objectives (from energy efficiency perspective) are attained.

In the CloudLightning approach, these objectives can be expressed as assessment functions (see Section 1.3.1). This outputs of assessment functions are aggregated as they pass upwrds through the hierarchy, the components in each layer of the hierarchy (i.e., Cell manager, pRouters, pSwitchs and vRMs layers) then use various strategies to calculate a value known as the Suitability Index (SI) (see Section 1.3.2). This index indicates the suitability of the the underlying region to host the next service request. Thus, a pathway guiding a service request to the place where it is most likely to find resources to host it is determined by always choosing to enter a region having the greatest SI. This is illustrated in Figure 1.2.

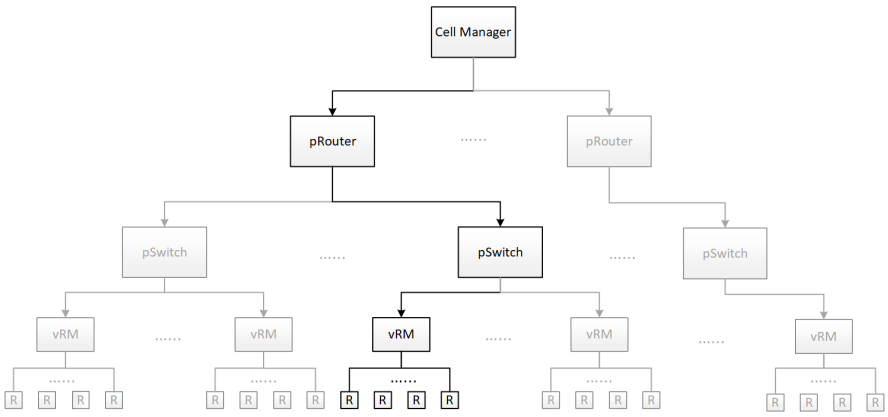

Fig. 1.2: An example of resource allocation path based on SI

1.3.1 Assessment functions

In CloudLightning, assessment functions output metrics that are used for monitoring and reflecting the state of the cloud infrastructure, including functional and non-functional behaviors, such as computation performance, power consumption, and management cost. Achieving energy efficiency in the cloud is not simpy a matter of reducing power consumption in isolation. Power reduction must be done in the context of guaranteeing workload computation performance; maximising the computation performance within a particular power consumption budget. Therefore, it makes sense to express the various aspects being captured by the CloudLightning assessment functions in terms of energy efficiency.

Performance per watt is a measure of the energy efficiency of a particular computer architecture or computer hardware.

$$f_{hw} = \frac{Performance}{Power} \tag{1.1}$$

Computational performance could be evaluated in measurable, technical terms, using one or more metrics, such as CPU/memory utilisation, throughput, floating point operations per second (FLOPS), millions of instruction per second (MIPS), and bandwidth.

Usually, the performance of a hardware configuration can be measured by any appropriate benchmark (such as SPECpower [6] and EEMBC [7]). Power models vary for different hardware types and hardware usage. For example, a commonly used linear power model [4] for CPU based servers is:

$$P(u) = P_{min} + (P_{max} - P_{min})u \tag{1.2}$$

where $u \in [0,1]$ is the CPU utilisation, P_{min} is the idle power consumption, $P_{\max}$ is the maximum power consumption.

1.3.2 Suitability index

In CloudLIghtning, the concept of **Suitability Index (SI)** is created for measuring how close a component is to its desired state, and hence how suitable its operating characteristics are for contributing to the global goal.

$$\underset{\mathbf{w}^\ell, \mathbf{m}^\ell \in \mathbb{R}^N}{\arg\max}\ \eta\left(\mathbf{w}^\ell, \mathbf{m}^\ell\right) \tag{1.3}$$

where $\mathbf{w}^\ell$ is an N-dimensional vector of weights corresponding to the Impetus in the ℓ-th level and $\mathbf{m}^\ell$ is an N-dimensional vector of metrics corresponding to the Perception in the ℓ-th level. General speaking, $\mathbf{w}^\ell$ presents the influence factor from the upper level indicating the perspectives from application characteristics, system objectives, service level agreement, etc., and $\mathbf{m}^\ell$ is the perception value (i.e., mean and/or maximum) of the lower level giving the average and/or the best performance view over the underlying system.

Overall, in terms of different characteristics of assessment functions, SI can be used to indicate the most suitable location to host incoming service requests. Thus, when the assessment functions are chosen to reflect the energy consumption with respect to different aspects of computation performance, the SI will indicate the most energy efficient location in the cloud resource fabric for hosting the next incoming service request. Similarly, if the assessment functions are chosen to reflect the management costs associated with different cloud configurations, the SI will indicate to the most efficient place to host the next incoming service request, with respect to that management cost.

---

[6] SPECpower: https://www.spec.org/power_ssj2008/

[7] EEMBC: http://www.eembc.org/

Figure 1.3 depicts the various applications of using the Suitability Index to achieve different goals.

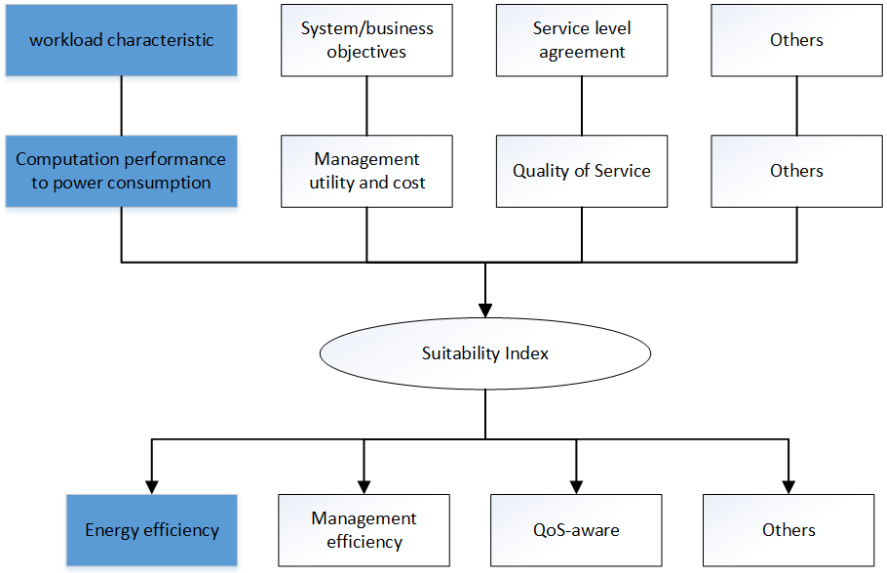

Fig. 1.3: Suitability Index Use Cases

1.3.2.1 SI strategy on CM

In the CloudLightning system, the Cell Manager needs to decide on which resource type is the most appropriate for hosting the next incoming service request. When user SLA objectives are satisfied, a choice can be made from the remaining types which maximises system objective (e.g., maximising energy efficiency).

The energy consumption for a specific service is often modelled as the integral of the power consumption function over the execution time for completing the application (mostly for batch and HPC workloads, not for the long-run web applications):

$$E = \int_{t_0}^{t_1} P(u(t))dt, \tag{1.4}$$

where $u(t)$is the CPU utilization function of time, which may change over time due to the workload variability. $t_1 - t_0$ denotes the application execution time.

The SI strategy on Cell Manager is to find the most suitable pRouter (that it, a specific resource type) for a specific service workload with the respect to energy efficiency. Therefore, the Cell Manager has to calculate/predict the energy consumption for each possible hardware type which might run that application workload based on

their current state and with knowledge of the service characteristics and hardware characteristics.

In some cloud simulators [6, 13, 18], service workloads can be modelled with three parameters: (i) input size, (ii) processing length, measured in Millions of Instructions (MI) and (iii) output size. The number of instructions can be calculated by the computational capacity (capability) of the processors (can be virtual machines or accelerators) multiplied by the service workload execution time. The service execution time can be computed from profiling the application with respect to parameters such as input size. Thus, the estimated execution time of the application workload running on a specified hardware resource (represented by a pRouter i) is:

$$t = \frac{N^{MI}}{N_i^{MIPS}} \tag{1.5}$$

where N^{MI} is the number of instructions in that service, and $N_i^{MIPS}(MAX)$ is the maximum number of instructions per second (MIPS) from pRouter i.

The total computational capacity of pRouter i hosting $N_{servers}$ number of servers, N_{acc} number of accelerators per server, with C_{proc} the combined computational capacity (in MIPS) of all processors (CPUs) of a server and C_{acc} the computational capacity (in MIPS) of an accelerator, can be defined as follows:

$$C_i^{pr} = C_{proc}N_{servers} + C_{acc}N_{acc}N_{servers}\ (MIPS). \tag{1.6}$$

The computational capacity (in MIPS) of the i-th pRouter (C_i^{pr}) can be defined in term of servers executing a task:

$$C_i^{pr} = C_{proc}\frac{N_{proc}^u}{N_{proc}} + C_{acc}N_{acc}^u\ (MIPS). \tag{1.7}$$

where N_{proc} is the number of processing units per server, $N_{proc}^u \in [0, N_{proc}N_{servers}]$ is the number of utilized processing units with respect to all servers under a pRouter and $N_{acc}^u \in [0, N_{acc}N_{servers}]$ is the number of utilized accelerators with respect to all servers under a pRouter. The result of eq. (1.7) would be similar for all resources at the beginning since no task has entered the system. Thus, in order to distinguish between resources, a random number can be added on the SI.

In order to compute the power consumption per pRouter we can utilize the model of Eq. (1.2). The power consumption for the accelerators is considered to be binary, since accelerators are either used or not and cannot be shared between virtual machines. The power consumption of the pRouter i can be estimated as follows:

$$P_i^{pr} = (P_{min} + P_{min}^{acc}N_{acc})N_{servers} + (P_{max} - P_{min})\frac{N_{proc}^u}{N_{proc}} + (P_{max}^{acc} - P_{min}^{acc})N_{acc}^u\ (W) \tag{1.8}$$

where N_{proc} is the number of processing units per server, $N_{proc}^u \in [0, N_{proc}N_{servers}]$ is the number of utilized processing units with respect to all servers under a pRouter and $N_{acc}^u \in [0, N_{acc}N_{servers}]$ is the number of utilized accelerators with respect to all

servers under a pRouter. The quantities P_{min}^{acc} and P_{max}^{acc} are the idle and the maximum power consumptions of an accelerator, respectively.

Therefore, the computation of the Suitability Index (SI) of each pRouter, at the Cell Manager, can be performed as follows:

$$SI_i^{pr} = C_i^{pr} / P_i^{pr} (MIPS/W). \tag{1.9}$$

This metric can be used in conjunction with availability (status) information to guide a task to the most efficient resource in terms of higher $MIPS/W$. The aforementioned equations can be reformed to take into account the overcommitment of resources. Guiding the task below the pRouter level can be performed using the same definition for the SI or the definition given in [5].

Similarly, the computation of the SI of each pSwtich (at a pRouter), and the computation of the SI of each vRM (at a pSwitch), will follow the same pattern as described in Eq.(1.7), (1.8) and (1.9).

1.3.2.2 SI strategy on vRM

The SI strategy on vRMs is quite different from the Cell Manager, pRouters and pSwitches, since the vRMs have up-to-date state information for all the resources under their control. vRMs can make accurate and timely decisions about the resource allocations with the respect to energy efficiency locally.

In CloudLightning, three strategies are applied to achieve energy efficient resource allocation.

(1) Best energy efficient node

The simplest strategy for a vRM to achieve energy efficiency is to deploy the virtual machine (VM) onto the most power efficient node (i.e., server). This strategy would appear to prioritize server utilization, however, success in selection depends on there being sufficient resources available from that server to satisfy the VM's requirement.

(2) Bin-packing

The objective of this approach is to minimize the energy consumption by placing VMs onto the minimum number of hosts. The model can be described as follows [7]:

$$
\begin{aligned}
\min \sum_{i \in V} \sum_{j \in H} p_{ij} v_{ij} + \sum_{j \in H} P_j h_j & & & \\
\text{s.t.} \sum_{i \in V} r_i v_{ij} & \leqslant & C h_j & \forall j \in H \\
\sum_{j \in H} v_{ij} & = & 1 & \forall i \in V \\
v_{ij} & \leqslant & h_j & \forall i \in V, j \in H \\
v_{ij} & \in & \{0,1\} & \forall i \in V, j \in H \\
h_j & \in & \{0,1\} & \forall j \in H
\end{aligned} \tag{1.10}
$$

where H is the set of hosts, V is the set of VMs in the cloud fabric, the objective is to decide how to rearrange V on H such that the total power consumption in the system is minimized. All $v \in V$ requirements r_i, such as CPU, memory, storage, must be satisfied by the targeting host; each h has a resource capacity limit C. The total power consumption is the sum of power p_{ij} consumed by CPUs of each VM i on host j, plus a fixed power P_j consumed by the other components of host j, such as memory and I/O. Let $h_j = 1$ represent choosing host j to be switched on, and 0 otherwise. Also, let $v_{ij} = 1$ represent the assignment of VM i to host j, and 0 otherwise. The first constraint enforces the capacity limit on each host. The second constraint ensures that each VM is assigned to exactly one host. The third constraint guarantees a host to be switched on if and only if a VM has been assigned to that host. The last two constraints indicate the state of a VM or host is either to be on or off.

However, this approach does not take the service workload characteristics and overcommitment into account, the next strategy applies the bin-packing with overcommitment ratio to further optimize the energy efficiency within a vRM.

(3) Bin-packing with overcommitment

There are three basic types of scheduling execution for VMs residing on a server: space-sharing, time-sharing and hybrid (time-space) policies.

- Space-sharing policies divide the system into partitions of processors so that more than one task can run on the system simultaneously; each on its own group of processors.
- In a time-sharing model, computer resources are committed and tasks are executed at the same time among many users. Time-sharing policies adopts preemption to alternate processors among a number of tasks that is usually determined by the multiprogramming level.
- Hybrid models (space-time sharing) combines the benefits of the two aforementioned types of policies. In a hybrid model, the system is divided in multiple processor groups and each group adopts a time shared policy by a distinct set of application tasks.

The time-sharing and hybrid model are commonly used in overcommitment of CPU and memory resources, which can increase server utilization. However, for compute-intensive workloads, overcommitment of resources can reduce performance and subsequently increase energy consumption, since the tasks are compet-

ing for resources. While for the communication-intensive workloads, the overcommitment of CPU and memory resources can be less harmful with respect to performance.

Thus, each vRM can customize its own overcommitment ratio with respect to the available physical resources (i.e., CPU, memory and network) and the workload characteristics, dynamically adjusting the overcommitment ratio of its associated resources to maximize the energy efficiency without violating the service performance.

1.4 Evaluation

In this Section, the scheme for computing the SI, in the Cell Manager level, is evaluated. Without loss of generality a Cell is considered as having two types of hardware. The characteristics of these resources are given in Table 1.1. For simplicity tasks are considered to fully utilize underlying resources and implementations exist for both tasks. The length of the task queue at each time-step was computed as a rounded random value, obtained the uniform random distribution, in the interval $[0, 0.85]$. Thus, the total number of tasks that entered the system was 35629. The tasks required $2, 4, 8, 16$ virtual cores and $1, 2, 3, 4$ accelerators, respectively, following uniform random distribution. The number of instructions for a task was computed in the interval $[100, 5000]$ MI using a uniform random distribution. The simulated time was 172800 seconds. Tasks enter the system with respect to the aforementioned parameters for the first 86400 seconds, while for the last 86400 the system is left to finish execution without incoming tasks. The energy consumption is computed as the integral of power consumption over time. The integral is computed numerically using the rectangle method.

Table 1.1: Characteristics of the resources

Resource	C_{proc}	C_{acc}	N_{proc}	N_{acc}	$N_{servers}$	P_{min}	P_{max}	P^{acc}_{min}	P^{acc}_{max}
1	160000	0	16	0	500	100	500	0	0
2	160000	480000	16	4	500	100	500	50	250

The first experiment concerns the computation of the SI using Eq. (1.6), (1.8) and (1.9). In Fig. 1.4 the Suitability Indices $(MIPS/W)$ computed with Eq. (1.6), (1.8) and (1.9) for the two types of hardware are given. In the beginning, and up to time-step 1841, tasks flow to the hardware of Type 2, since its SI has the highest value. When the value of the SI of hardware Type 2 reaches the value of the SI of hardware Type 1, tasks start to flow to the resources of Type 1. From time-step 1842 to time-step 2350, the two pRouters are competing for the incoming tasks, since the SIs are almost similar in value. From the time-step 2351 to time-step 86000 the pRouter hosting hardware of Type 2 is almost completely utilized, thus most of

the tasks flow to the pRouter hosting hardware of Type 1. This process leads to an overall power consumption of 25.5129 MWh.

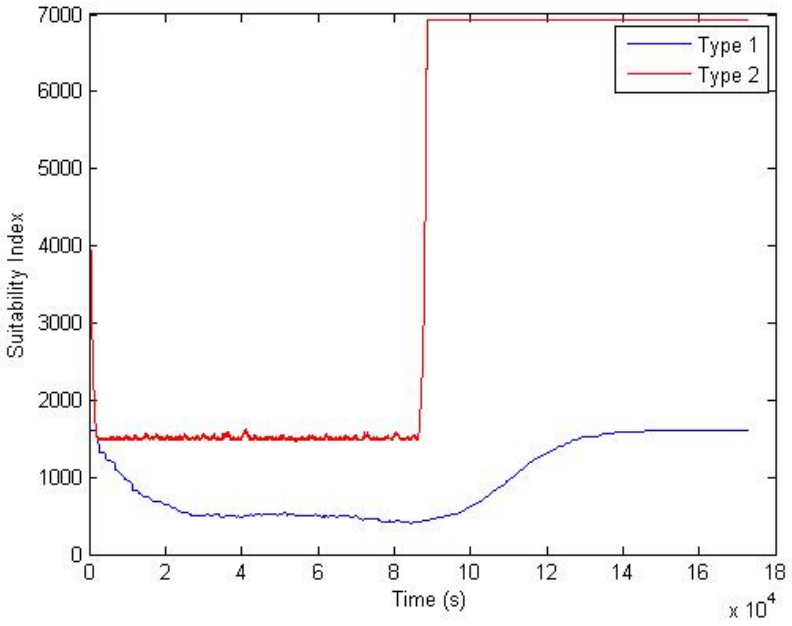

Fig. 1.4: Suitability Indices $(MIPS/W)$ computed with eq. (1.6), (1.8) and (1.9) for the two types of hardware.

The second experiment concerns the computation of the SI using Eq. (1.7), (1.8) and (1.9). In Fig. 1.5 the Suitability Indices $(MIPS/W)$ computed with Eq. (1.7), (1.8) and (1.9) for the two types of hardware are given (first task arriving to pRouter hosting hardware of Type 1). In this experiment the initial value of both SIs is 0, thus the first task flows to the first available pRouter (e.g. Type 1). The following tasks also flow to pRouter hosting hardware of type 1 until it is almost completely utilized. Then, tasks start to flow to the pRouter hosting hardware of Type 2. Due to the large value of the SI of the pRouter hosting hardware of Type 2 the majority of the tasks continue to flow there, while a small fraction of the tasks flow to the pRouter hosting hardware of Type 1. This process leads to an overall power consumption of 26.2064 MWh.

The initial hardware choice dictates the first task flows and impacts the direction taken by the following tasks. In Fig. 1.6 the Suitability Indices $(MIPS/W)$ computed with Eq. (1.7), (1.8) and (1.9) for the two types of hardware are given (first task arriving to pRouter hosting hardware of Type 2). This process leads to an overall power consumption of 25.6172 MWh. Choosing the fastest hardware type first leads

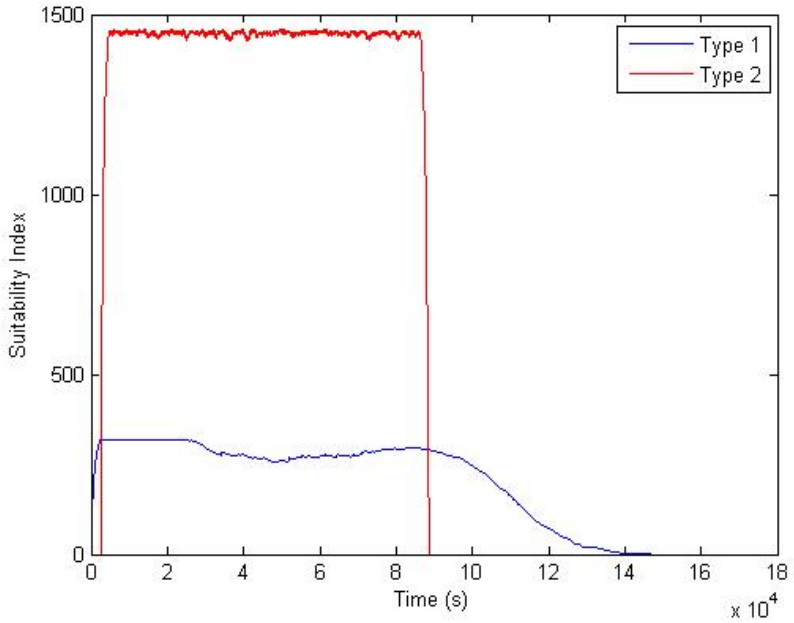

Fig. 1.5: Suitability Indices $(MIPS/W)$ computed with eq. (1.7), (1.8) and (1.9) for the two types of hardware (first task arriving to pRouter hosting hardware of Type 1).

to improved energy consumption, since the CPU-Accelerator pair executes tasks faster.

The energy consumption using Eq. (1.6), (1.8) and (1.9) is slightly reduced compared to the other two approaches.

The last experiment concerns the computation of the SI following the initial approach given in [5], with:

$$f_1(N^u_{proc}) = C\frac{N^{total}_{proc} - N^u_{proc}}{N^{total}_{proc}}, \tag{1.11}$$

$$f_2(N^u_{proc}) = \frac{P_i(N^{total}_{proc} - N^u_{proc})}{PN^u_{proc} + P_i(N^{total}_{proc} - N^u_{proc})}, \tag{1.12}$$

and the vector of weights $\mathbf{w} = [1\ 1]$. The Suitability Index is computed as $SI = w_1 f_1 + w_2 f_2$, [5]. For the two types of hardware (1 and 2) the values of the relative computational capability C were 1.0 and 3.0, respectively, [5]. The values for the relative power consumption P and the relative idle power consumption P_i for hardware Type 1 were 1.0 and 0.2, respectively, while for hardware Type 2 were 1.5 and 0.2, [5]. For hardware Type 2, Eq. (1.11) and (1.12) are computed with respect to the total number of accelerators as well as the number of utilized accelerators. In

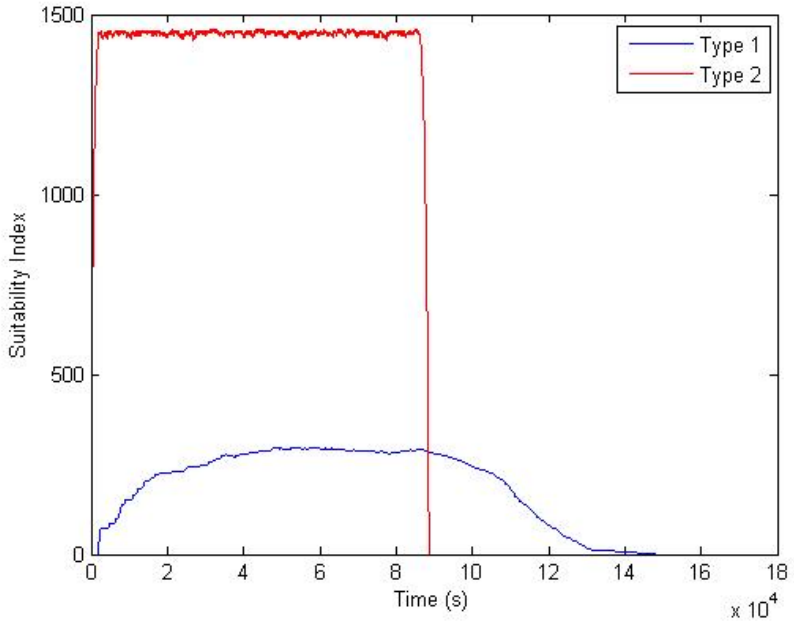

Fig. 1.6: Suitability Indices $(MIPS/W)$ computed with Eq. (1.7), (1.8) and (1.9) for the two types of hardware (first task arriving to pRouter hosting hardware of Type 2).

Fig. 1.7 the Suitability Indices computed with respect to the original design given in [5], are presented. In this approach the two pRouters continuously compete for acquiring tasks from time-step 0 to 86400. The pRouter hosting hardware of Type 2 is the first receiving tasks until the point where its SI becomes comparable to the SI of the pRouter hosting hardware of type 1. The energy consumption of the system was 27.1687 MWh.

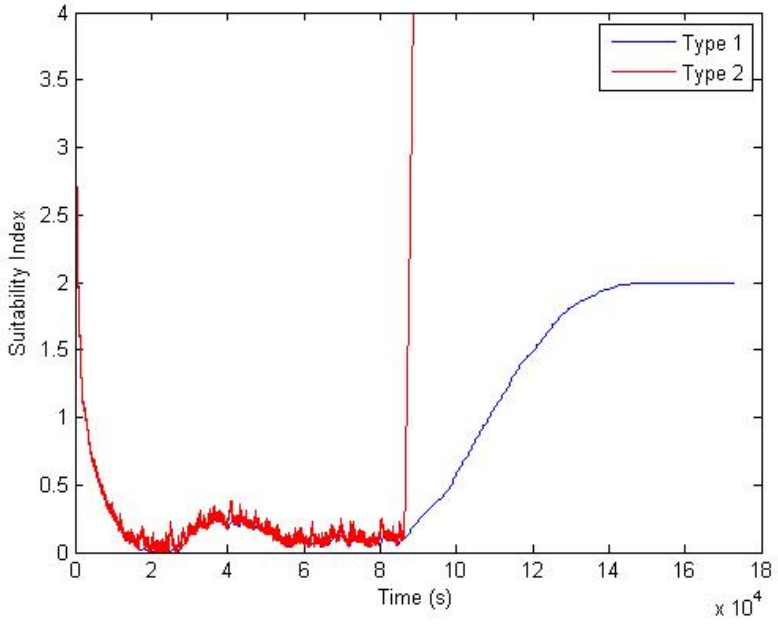

Fig. 1.7: Suitability Indices computed with respect to the original design given in [5].

The proposed approaches for computing the SI lead to an overall improvement between approximately $3.54\% - 6.1\%$ for the energy consumption. The computation of the SI based on Eq. (1.6), (1.8) and (1.9) leads to the greater improvement, followed up closely by the approach using eq. (1.7), (1.8) and (1.9) with first task arriving to pRouter hosting hardware of Type 2.

1.5 Conclusion

The CloudLightning project attempts to address resource management issues associated with hyper-scale cloud deployments. The complexity of these deployments makes the selection of the most suitable resource to host the next service request a very challenging task. Decentralising the decision process is, itself, not sufficient to adequately address this problem, since any distributed collection of resource managers still have to share relevant information and have to together decide on that resource that meets both the service requirements and the business objectives of the cloud service provider. These business objectives can be many and varied and in the CloudLightning project, they are captured through the use of Weighted Assessment Functions. These functions are used to measure various aspects of the status of the system and this information is subsequently used to determine how close the

system is to achieving the business object associated with each respective function. By dynamically weighting Assessment Functions, the cloud service provider can steer the evolution of the system in the direction of those objects that reflect the providers immediate needs. As status information is propagated upwards through the CloudLightning hierarchy, it is combined into a view of the underlying levels. This view reflects how well those parts of the system are performing with respect to the business objectives and, by extension, how suitable those levels are to respond to imminent resource requests. The suitability is reflected in a measure known as the Suitability Index.

An important business object for cloud providers is to minimise energy consumption. The CloudLightning architecture embodies many heterogeneous resources, each with its own energy consumption and exploitation characterises. The work described here illustrates how the Suitability Index can be specifically tailored in such a complex environment in support of globally minimising energy consumption. This specially tailored form of the Suitability Index was evaluated empirically and the results were presented, showing and improvement of about 6% over the unspecialised Suitability Index calculation.

The CloudLightning architecture is designed so that it can be easily extended with a multiplicity of heterogeneous resource types. This achieved using a Plug and Play mechanism. When a new resource is added to the system using this mechanism, it is assigned to an appropriate resource manager so that it can be effectively managed. Decision on the most appropriate resource manager to manage a resource is an analogous process to deciding on the most appropriate resource to host a service. Thus, the descriptor, representing a new addition to the resource fabric, follows the path of lowest (as apposed to the highest) Suitability Indices until it becomes associated with an appropriate resource Manager. This ensures that the resource is placed into the system so as to maximise its utility in meeting the business objectives and in balancing the values of the Suitability Indices across the cloud.

Acknowledgement

This work is funded by the European Union's Horizon 2020 Research and Innovation Programme through the CloudLightning project under Grant Agreement Number 643946.

References

[1] Ahuja M, Chen CC, Gottapu R, Hallmann J, Hasan W, Johnson R, Kozyrczak M, Pabbati R, Pandit N, Pokuri S, et al (2009) Peta-scale data warehousing at yahoo! In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, ACM, pp 855–862

[2] Barroso LA, Clidaras J, Hölzle U (2013) The datacenter as a computer: An introduction to the design of warehouse-scale machines. Synthesis lectures on computer architecture 8(3):1–154

[3] Beloglazov A, Buyya R, Lee YC, Zomaya A, et al (2011) A taxonomy and survey of energy-efficient data centers and cloud computing systems. Advances in computers 82(2):47–111

[4] Beloglazov A, Abawajy J, Buyya R (2012) Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future generation computer systems 28(5):755–768

[5] C Filelis-Papadopoulos, H Xiong, A Spataru, G Castane, D Dong, G Gravvanis and J P Morrison (2017) A Generic Framework Supporting Self-organisation and Self-management in Hierarchical Systems. The 16th International Symposium on Parallel and Distributed Computing (ISPDC 2017), Paper accepted

[6] Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R (2011) Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exper 41(1):23–50, URL http://dx.doi.org/10.1002/spe.995

[7] Dong D, Herbert J (2013) Energy efficient vm placement supported by data analytic service. In: Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on, IEEE, pp 648–655

[8] Etro F (2009) The economic impact of cloud computing on business creation, employment and output in europe. Review of business and economics 54(2):179–208

[9] Hauswald J, Laurenzano MA, Zhang Y, Li C, Rovinski A, Khurana A, Dreslinski RG, Mudge T, Petrucci V, Tang L, et al (2015) Sirius: An open end-to-end voice and vision personal assistant and its implications for future warehouse scale computers. In: Proceedings of the Twentieth International Conference

on Architectural Support for Programming Languages and Operating Systems, ACM, pp 223–238

[10] Hindman B, Konwinski A, Zaharia M, Ghodsi A, Joseph AD, Katz R, Shenker S, Stoica I (2011) Mesos: A platform for fine-grained resource sharing in the data center. In: Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation (NSDI 2011), pp 295–308

[11] Joseph E, Conway S, Dekate C, Cohen L (2014) Idc hpc update at isc14

[12] Marinescu DC (2016) Complex Systems and Clouds: A Self-Organization and Self-Management Perspective. Morgan Kaufmann

[13] Núñez A, Vázquez-Poletti JL, Caminero AC, Castañé GG, Carretero J, Llorente IM (2012) icancloud: A flexible and scalable cloud infrastructure simulator. Journal of Grid Computing 10(1):185–209

[14] Schubert L, Jeffery K, Neidecker-Lutz B (2010) The future of cloud computing: Opportunities for european cloud computing beyond 2010. Expert Group report, public version 1

[15] Sohrabi S, Moser I (2015) A survey on energy-aware cloud. European Journal of Advances in Engineering and Technology 2(2):80–91

[16] Sverdlik Y (2014) Survey: Industry average data center pue stays nearly flat over four years. Data Center Knowledge 2(06)

[17] Tang L, Mars J, Zhang X, Hagmann R, Hundt R, Tune E (2013) Optimizing google's warehouse scale computers: The numa experience. In: High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on, IEEE, pp 188–197

[18] Tian W, Xu M, Chen A, Li G, Wang X, Chen Y (2015) Open-source simulators for cloud computing: Comparative study and challenging issues. Simulation Modelling Practice and Theory 58, Part 2:239 – 254, special issue on Cloud Simulation

[19] Whitney J, Delforge P (2014) Data center efficiency assessment. National Resources Defense Council, New York