Modeling and Simulation of QoS-Aware Power Budgeting in Cloud Data Centers

Jakub Krzywda*, Vinícius Meyer†, Miguel G. Xavier†, Ahmed Ali-Eldin*‡,
Per-Olov Östberg*, César A. F. De Rose† and Erik Elmroth*
*Department of Computing Science, Umeå University, Sweden
†Polytechnic School, Pontifical Catholic University of Rio Grande do Sul, Brazil
‡College of Information and Computer Sciences, University of Massachusetts Amherst, USA
Email: jakub@cs.umu.se

***Abstract*—Power budgeting is a commonly employed solution to reduce the negative consequences of high power consumption of large scale data centers. While various power budgeting techniques and algorithms have been proposed at different levels of data center infrastructures to optimize the power allocation to servers and hosted applications, testing them has been challenging with no available simulation platform that enables such testing for different scenarios and configurations. To facilitate evaluation and comparison of such techniques and algorithms, we introduce a simulation model for Quality-of-Service aware power budgeting and its implementation in CloudSim. We validate the proposed simulation model against a deployment on a real testbed, showcase simulator capabilities, and evaluate its scalability.**

I. Introduction

The carbon footprint of large-scale data centers has been actively studied for the past decade. In 2016, it has been estimated that the total data center power consumption in the US alone is equivalent to the power consumption of around 6.4 million average American homes [1]. With the rise of mega-scale data centers that sometimes consume the power equivalent of thousands of homes, many researchers considered how to improve the efficiency and efficacy of server power consumption. For data center operators, the economical and environmental impact due to the increased cost of electric and power infrastructure needed, and the increase in the power consumption, has sparked interest in new techniques and mechanisms to help reduce or eliminate these impacts. In addition, current trends of increasing the size of data centers, with the goal of building exascale computing facilities in the near future, exacerbate the problem of controlling power consumption and makes it more critical [2].

In parallel to the growth in data centers infrastructures, recent advances in server power management, virtualization, and new hardware technologies have helped reduce the overall growth of data center power consumption [1]. However, it is still important to continue the development of new techniques that reduce the power consumption across the entire computing stack to make sure that data centers carbon footprint does not significantly increase [1]. These techniques should span better software development [3], better hardware [4], and better server power management [5].

While previous work has looked at different aspects of controlling data center power consumption, the reduction of CPU power consumption has received special attention. It is estimated that up to 42% of all the power consumed in a data center are used by the CPUs [6]. Techniques such as dynamic voltage and frequency scaling (DVFS) [7], per-core power gating [8], and scheduling for CPU deep sleep [9] have been proposed as possible mechanisms to control the CPU power consumption.

Another popular solution to reduce the overall data center power consumption, including that consumed by the CPUs, is power budgeting, an approach to reduce the negative consequences of high power consumption by limiting the peak power consumption of a data center facility [10]. Power budgeting techniques have gained popularity due to their practicality, and are currently being used in many large scale data centers [10].

However, limiting the power consumption of servers during periods of high workloads comes at a cost. Typically, power budgeting can result in performance degradation of the hosted applications in the data center, specially under workload pressure. Applications can start experiencing slow downs, thrashing, or become totally unresponsive if they are not allocated enough effective power. For this solution to be viable, data center operators have to minimize the impact of limiting power consumption on application performance. Today, no tools exist that enable data center operators to navigate the impact of power budgeting on the hosted applications without actually running this in real deployments.

In this work, we tackle the problem of how to evaluate optimization algorithms for application performance aware power budgeting at multiple levels of data center infrastructure (server, cluster, whole data center). We propose a framework to model and simulate data center infrastructures with limited power budgets, together with algorithms for server power capping, cluster power shifting and adjusting the data center power budget. We have implemented the proposed framework as an extension to the CloudSim simulator [11]. To achieve the application performance awareness, we have added server power models, application power-performance models, performance-cost models, and electricity cost models. Our solution can simulate the peak power and total energy consumption of the facility, the performance of hosted applications, track the application Quality-of-Service (QoS) violations, and calculate data center operational costs.

II. Background and State of the Art

A. Power-Aware Modeling of Cloud Computing Servers

In data centers, power consumption of computing nodes is mostly determined by the CPU, memory, disk storage and network interfaces [12]. Processors are the elements responsible for the largest part of energy consumption [13]. The workload and the frequency of a CPU have a considerable impact on its power usage. The lower a processor's frequency, the slower it computes but also the less energy it consumes [14]. It has been shown that the nodes power consumption can be described by a linear relationship between the power consumption and CPU utilization [13].

For computer systems energy consumption, power models are normally divided into two components, *static* and *dynamic* power. Static power is the power consumption when the node is powered on and idle. Dynamic power depends on the current utilization of the CPU [13].

For environments which experience only few idle periods, such as HPC infrastructures, Dynamic Voltage and Frequency Scaling (DVFS) appears as an alternative to switching off machines [14]. DVFS technique temporarily decreases voltage supply level at the expense of lowering processing speed according to the application workload [12].

B. Data Center Power Budgeting

In order to decrease the adverse effects of high power consumption, power budgeting emerges as a solution to limit the peak power consumption of data center infrastructures [15]. Below, we introduce approaches that take advantage of this technique.

Power Capping allows limiting the power consumption of a single server. Traditionally, DVFS was used to reduce CPU power consumption. Other techniques, such as CPU pinning [16] and Forced Idleness [17], were also utilized to achieve that goal. More recently, Running Average Power Limit (RAPL) [18] has been proposed as an alternative, that enables direct control over the power consumption of CPU and memory of a single server.

Power Shifting facilitates dynamic reconfiguration of power limits among multiple servers to match the workload levels and priorities of hosted applications. Felter et al. [19] define this technique as the capability of sharing a system's power budget among its infrastructure.

Data Center Power Budget Adjustment enables reduction of operational expenditures through utilizing dynamic pricing of electricity [20] or reduction of environmental impact by adapting the data center total power consumption according to the availability of renewable energy sources [21].

C. Power-Aware Cloud Simulation

CloudSim [11] is a discrete event simulator that provides a virtualization ecosystem with features for modeling the management of virtual machines in a data center, including policies for provisioning of virtual machines to hosts, scheduling of resources of hosts among virtual machines, scheduling of tasks in virtual machines, and modeling of costs incurring in such operations. Energy-aware extensions have been developed and incorporated into CloudSim, e.g., power models [22], DVFS modeling [23] and ACPI Global/Sleep states [24]. These extensions enhanced CloudSim's original abstraction for representing power consumption over distributed cloud computing infrastructures.

Since we are interested in simulating infrastructures that provide power budgeting capabilities and none of the above mentioned simulators offer them, we have decided to extend the existing simulator, namely the DVFS version of CloudSim [23], with these features.

III. Simulation Model

In order to simulate the QoS-aware power budgeting, we first need to define what metrics we are interested in, which infrastructure entities and their configurations have an influence on these metrics, and what are the relations among all these parameters. Therefore, in this section, we explain how we model the data center system including servers and their power budgets, applications and their performance, as well as how we calculate the data center operational costs.

We consider a data center with operational costs depending on the total energy consumption, peak power consumption, and QoS violation penalties. The data center houses servers that support power capping technologies and host diverse applications. Applications have a fixed placement over the servers and the application performance depends on the power budget allocated to the host server. Both the data center power budget and the server power budgets may be over-subscribed—the sum of theoretical maximal power consumption of all the hosted applications can exceed the data center and/or server power budget. This leads to a control problem where data center operators, in times of high workloads, have to choose between increased electricity costs or to dynamically adjust the data center power budget and distribute it among the servers and applications.

A. Data Center Infrastructure

The data center consists of I servers and hosts J applications. The power consumption of a server is denoted by P^{server} and is a sum of a static part $P^{\text{server}}_{\text{static}}$ and a dynamic part $P^{\text{server}}_{\text{dynamic}}$, i.e., $P^{\text{server}} = P^{\text{server}}_{\text{static}} + P^{\text{server}}_{\text{dynamic}}$.

We assume that applications consume the whole assigned power budget. Therefore, the dynamic part of the server power consumption is equal to a sum of power budgets of applications hosted on the server, $P^{\text{server(i)}}_{\text{dynamic}} = \sum_{j=1}^{J} \left(P^{\text{app(j)}}_{\text{budget}} z_i^j \right)$,where z_i^j is the placement of application j (equal to 1 if the application j is hosted on the server i and 0 otherwise).

The power consumption of the data center is a sum of power consumption of all I servers, $P^{\text{dc}} = \sum_{i=1}^{I} P^{\text{server}(i)}$.

The power consumption of data center infrastructure components is limited by hard constraints $P^{\text{server}}_{\text{limit}}$ for servers and $P^{\text{dc}}_{\text{limit}}$ for the whole data center. The nature of the limits is physical, e.g., a power breaker limit of the power delivery infrastructure or the amount of heat that the cooling system may transfer. Power consumption is constrained by budgets $P^{\text{dc}}_{\text{budget}}$ and $P^{\text{server}}_{\text{budget}}$. Their nature is financial, e.g., reduction of the electricity costs, or results from the decisions of the power shifting controller, which prioritizes one server over another.

B. Application Performance and Cost

We use the application performance and QoS violation models introduced in ALPACA [25]. The performance models define two application classes: applications with gradual performance degradation that continue running with a limited power

budget, and applications with abrupt performance degradation that effectively fail if the power budget is below a certain threshold.

For applications with a gradual performance degradation we model the relation between the application power budget $P^{\text{app}}_{\text{budget}}$ and the application performance p^{app} using one of the following functions: linear (Eq. 1), exponential (Eq. 2), or sigmoid (Eq. 3).

$$p^{\text{app}} = \max\left(0, \min\left(1, a \cdot P^{\text{app}}_{\text{budget}} + b\right)\right), \tag{1}$$

$$p^{\text{app}} = \max\left(0, \min\left(1, a \cdot e^{b \cdot P^{\text{app}}_{\text{budget}}} + d\right)\right), \tag{2}$$

$$p^{\text{app}} = \max\left(0, \min\left(1, \frac{a}{1 + e^{-b \cdot \left(P^{\text{app}}_{\text{budget}} - c\right)}} + d\right)\right), \tag{3}$$

where a, b, c, and d are application specific parameters determined using linear regression for the linear model and nonlinear least squares for exponential and sigmoid models.

For applications with an abrupt performance degradation, we use the following model of the relation between the application power budget $P^{\text{app}}_{\text{budget}}$ and the application performance p^{app}:

$$p^{\text{app}} = \begin{cases} 0, & \text{if } P^{\text{app}}_{\text{budget}} < P^{\text{app}}_{\text{min}}, \\ p^{\text{app}}_{\text{target}}, & \text{otherwise,} \end{cases}$$

where $P^{\text{app}}_{\text{min}}$ is the minimal power budget needed to run the application.

Each application has a defined target performance constraint $p^{\text{app}}_{\text{target}}$ and a threshold when the application becomes unusable $p^{\text{app}}_{\text{unusable}}$. A violation of a performance constraint results in a penalty. The performance degradation of an application with actual performance p^{app} can be defined in various ways, e.g.:

$$p^{\text{app}}_{\text{degr}} = \begin{cases} \frac{p^{\text{app}}_{\text{target}} - p^{\text{app}}}{p^{\text{app}}_{\text{target}} - p^{\text{app}}_{\text{unusable}}}, & \text{linear,} \\ \left(\frac{p^{\text{app}}_{\text{target}} - p^{\text{app}}}{p^{\text{app}}_{\text{target}} - p^{\text{app}}_{\text{unusable}}}\right)^2, & \text{quadratic,} \\ A \exp\left(\frac{p^{\text{app}}_{\text{target}} - p^{\text{app}}}{p^{\text{app}}_{\text{target}} - p^{\text{app}}_{\text{unusable}}}\right) - A, & \text{exponential,} \end{cases}$$

where $A = \frac{1}{\exp(1) - 1}$.

The cost of QoS violations of an application with a performance degradation $p^{\text{app}}_{\text{degr}}$ is then defined as follows:

$$c^{\text{app}}_{\text{q}} = \begin{cases} 0, & \text{if } p^{\text{app}} \geq p^{\text{app}}_{\text{target}}, \\ \lambda^{\text{app}}, & \text{if } p^{\text{app}} \leq p^{\text{app}}_{\text{unusable}}, \\ \lambda^{\text{app}} p^{\text{app}}_{\text{degr}}, & \text{otherwise,} \end{cases}$$

where λ^{app} is the cost of QoS violation for the studied application.

C. Electricity Costs

The electricity cost consists of two parts:

- the cost of energy consumed by the data center over the billing period T, $c_{\text{e}} = \epsilon \sum_{t \in T} P^{\text{dc}}(t)$, where ϵ is a unit cost per Wh; and
- an additional cost proportional to the peak power draw by the data center during that period, $c_{\text{p}} = \pi \max_{t \in T} P^{\text{dc}}(t)$, where π is a unit cost per W of peak power consumption [26].

The total cost of electricity equals $c_{\text{E}} = c_{\text{e}} + c_{\text{p}}$.

D. Optimization

Finally, we formulate the optimization model as follows:

$$\text{minimize} \quad \sum_{j=1}^{J} c^j_{\text{q}} + c_{\text{E}} \tag{4}$$

$$\text{subject to} \quad P^{\text{dc}} \leq P^{\text{dc}}_{\text{limit}} \tag{5}$$

$$\forall_i P^{\text{server}_i} \leq P^{\text{server}_i}_{\text{limit}}, \tag{6}$$

where c^j_{q} is the QoS cost for application j.

The goal of the optimization is to minimize the sum of QoS violation costs aggregated over all J hosted applications and the electricity cost (Eq. 4). While, the power consumption of the data center cannot exceed the data center power limit (Eq. 5) and the power consumption of each server cannot exceed the server power limit (Eq. 6).

IV. Model Implementation in CloudSim

After defining the requirements for the QoS-aware power budgeting models in the previous section, we have analyzed the models available in the original CloudSim and identified what needs to be extended in order to simulate intended metrics.

A. Limitations of the Original CloudSim Models

The original CloudSim provides an abstract power consumption implementation (`PowerModel`), which allows its extension to support different power consumption models [11]. DVFS extension applied `Linear` power model [23] and in recent releases, CloudSim incorporates more energy models, such as: `Square`, `Cubic`, `Square root` and `Linear interpolation` [27].

Although CloudSim supports the development of custom application service models (by extending `Cloudlet`), its original implementation provides a type of application which is assigned with a pre-configured processing capability, in MIPS (million instructions per second) [11], following the batch job model. Hence, simulation of application at request-level, more specifically latency-sensitive services, are not supplied by default.

For power budgeting simulation, we extend infrastructure power and application performance models, as well as, optimization capabilities.

B. Data Center Infrastructure

Extensions to the original CloudSim infrastructure power models cover three levels of the infrastructure: CPUs (`PowerModelRapl`), servers (`PowerHostWithBudget`), and the whole data center (`PowerDatacenterWithBudget`). At CPU level, we add support for modelling RAPL capabilities. Power limits are specified at data center and server level. Power budgets are supported at all levels.

C. Application Performance and Cost

The extended simulation model adds application performance awareness by allowing characterization of a wider range of application types, capturing power budget, application performance, and QoS violation cost relationships.

`PowerPerformanceAwareCloudlet` extends the original `Cloudlet` with capabilities to model applications with varying workload and limited power budget.

`PowerPerformanceModel` calculates the application performance based on the captured application models. Application performance degradation can be modelled using various functions: `step`, `linear`, `exponential`, and `sigmoid`, using appropriate model parameters ($P^{\text{app}}_{\text{min}}, a, b, c, d$).

`PerformanceCostModel` computes QoS violation cost based on the previously calculated application performance level p^{app}, user-defined performance thresholds indicating the target $p^{\text{app}}_{\text{target}}$ and unusable $p^{\text{app}}_{\text{unusable}}$ performance levels, as well as, the penalty value λ. Various functions are available to model the dependency between the application performance degradation and QoS violation cost: `linear`, `quadratic`, and `exponential`.

D. Optimization Algorithms

In our model, we support three power budget techniques: server power capping, power shifting and data center power budget adjustment. The model provides abstract classes, which can be extended in order to implement a specific algorithm to be evaluated. To showcase the simulator capabilities, we have implemented multiple algorithms for all these techniques, details are presented in Section V-A.

`ServerPowerCappingController` allows distribution of a given power budget among hosted cloudlets. `ClusterPowerShiftingController` enables distribution of a given power budget among servers. `DataCenterPowerBudgetController` is used for adjusting the power budget of the whole data center. Moreover, our implementation supports tracking the optimization time for future analysis, e.g., of the algorithm scalability.

E. Model Validation

Before using the simulator to evaluate power budgeting algorithms, we validate our simulation model against a real deployment on a small scale testbed. We perform the validation using three servers with Intel Xeon E5-2620 v2 processor, which supports RAPL and consists of 2 CPU sockets, each with 6 CPU cores. The idle CPU socket consumes approximately 17.5 W. We note that both the original CloudSim and our extension allow modeling and simulation of heterogeneous infrastructures, but in this validation we use only one CPU model.

Simulated application performance metrics depend on the application type. For the batch jobs, the most important characteristic is the completion time, while for the latency-sensitive services the tail response time is crucial. Below, we describe the model validation method and results for both types of applications.

Batch Job Model. To evaluate the batch job simulation model, we run the following experiment on our testbed and in the simulator. We execute five instances of SysBench, a CPU intensive application that performs prime number computations. In this experiment we verify which numbers up to 100 000 are prime numbers using six simultaneous threads. Each SysBench instance is pinned to a different CPU socket and initially all sockets are assigned an equal power budget of 22 W. The first SysBench instance is initiated after 15 s and subsequent instances are started with a delay of 10 s. At time 85 s the socket power budgets are changed to different levels (22, 25, 27, 30, and 35 W), what affects their execution times. We record the power consumption of each CPU socket using RAPL monitoring capabilities and register job completion times. The experiment is completed when the last instance finishes. Figure 1a shows the power consumption over the testbed run. Next, we model and simulate this scenario in CloudSim. Figure 1b shows the simulated experiment. Finally, we compare the simulated execution times and power consumption against the testbed run. The average error for execution time is 4% and the error for the total power consumption is 5%.

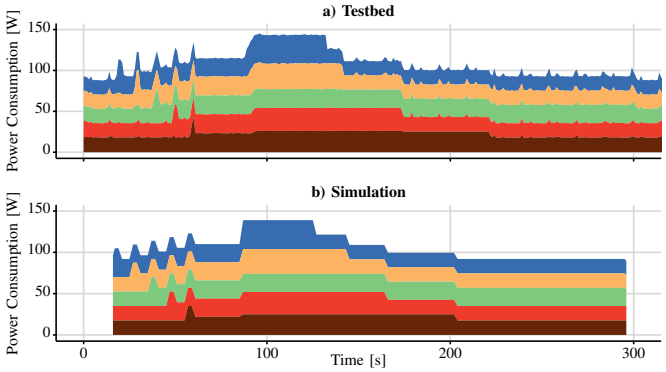

Fig. 1. Power consumption of five SysBench instances in testbed and in simulation. Power consumption values are stacked on each other.

Latency-Sensitive Service Model. For validation of the latency-sensitive service model we use Web Search benchmark application from CloudSuite that builds on the Apache Solr search engine framework. Since in the simulation we directly use `PowerPerformanceModel` to obtain the application performance level, here we validate the precision of the captured model. During each experiment run, we pin the server application to a single CPU socket, set the power budget of that socket, and execute the benchmark. We repeat the experiment for 10 power budget levels between 22 and 31 W. Figure 2 shows how the application performance and socket power consumption change with the power budget. We model the application performance using an exponential model and the power consumption using a piece-wise linear model. The average difference for response time is 0.04 s and the mean error for power consumption is 3%.

V. Evaluation of the Simulator

This section evaluates some aspects of the simulator extension. It also demonstrates how the extended simulator can be used by researchers and data center operators to evaluate and compare various power budgeting strategies.

A. Algorithms for Power Budgeting

First, we describe a set of algorithms for power budgeting that we have used to present the capabilities of the proposed CloudSim extension. The algorithms support single server power capping, power shifting among servers, and adjusting the data center power budget.

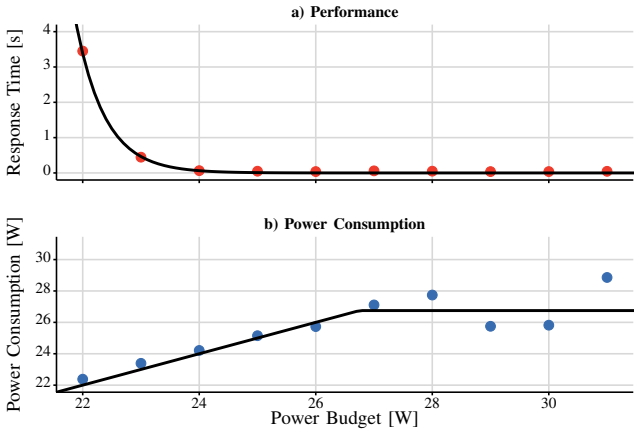

Fig. 2. Models of Web Search (a) performance and (b) power consumption. Circles represent the observed values and lines show the derived models.

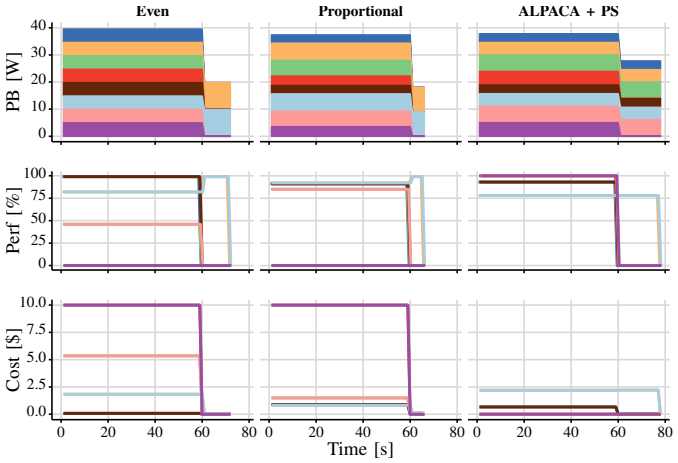

Fig. 3. Selected cloudlet metrics recorded over the simulation time. Values of power budgets assigned to each cloudlet shown in the top row are stacked, while the cloudlet performance and application cost metrics are presented as absolute values.

Power Capping. For single server power capping we use three algorithms: *even distribution* that gives each application identical power budget: $P^{\text{app}}_{\text{budget}} \leftarrow \frac{P^{\text{server}}_{\text{budget}}}{J^{\text{server}}}$; *proportional distribution* that divides the server power budget considering the requested application power budget: $P^{\text{app}}_{\text{budget}} \leftarrow \frac{P^{\text{server}}_{\text{budget}}}{R^{\text{server}}_{\text{budget}}} R^{\text{app}}_{\text{budget}}$; and *ALPACA* that minimizes the server operational costs (QoS violations and electricity costs): $\min \sum_{j=1}^{J^{\text{server}}} c^j_{\text{q}} + c_{\text{E}}$.

Power Shifting. For data center power shifting we use three algorithms: *even distribution* that gives each server identical power budget: $P^{\text{server}}_{\text{budget}} \leftarrow \frac{P^{\text{dc}}_{\text{budget}}}{J}$; *proportional distribution* that divides the data center power budget considering the requested server power budget (sum of the requested power budget of all hosted applications): $P^{\text{server}}_{\text{budget}} \leftarrow \frac{P^{\text{dc}}_{\text{budget}}}{R^{\text{dc}}_{\text{budget}}} R^{\text{server}}_{\text{budget}}$; and *Power Shepherd* that minimizes the data center operational costs (QoS violations and electricity costs): $\min \sum_{i=1}^{I} c^i_{\text{q}} + c_{\text{E}}$.

Adjusting Data Center Power Budget. For this showcase we use an algorithm that sets the data center power budget at a fixed value based on the power limit $P^{\text{dc}}_{\text{limit}}$ and a given constant α, i.e. $P^{\text{dc}}_{\text{budget}} \leftarrow \alpha P^{\text{dc}}_{\text{limit}}$. More sophisticated algorithms can be used for adjusting the data center power budget over the experiment time in response to the varying electricity prices or based on the historical statistics regarding the electricity usage and workload predictions, e.g., using Cumulative Distribution Functions (CDFs) of power consumption.

B. Simulator Capabilities

To show the capabilities of the extended simulator, we model and execute a scenario with a mix of latency-sensitive applications and batch jobs. We simulate application execution optimized with various algorithms: a) even power capping and even power shifting, b) proportional power capping and power shifting, and c) ALPACA for power capping and Power Shepherd for power shifting. The extension is capable of recording varying metrics over the experiment time, as well as, calculating aggregated metrics summarizing the whole experiment.

Figure 3 shows the power budget assigned to each cloudlet by the optimization algorithms (top row), the cloudlet performance achieved at the given power budget level (middle row), and the application cost (bottom row). These kinds of visualizations help to understand the behaviour of applications in response to the optimization actions.

TABLE I
SELECTED METRICS SUMMARIZING THE SIMULATION RUN

Metric	Even	Proportional	ALPACA + PS
Peak Power [W]	70.0	67.9	68.3
Total Energy [Wh]	1.29	1.17	1.30
QoS Cost [$]	2043	1559	417
Electricity Cost [$]	71	69	70
Total Cost [$]	2114	1628	487

In Table I we show some of the aggregated metrics that are reported by the simulator. They are part of the simulation model introduced in Section III and show elements that contribute towards the data center operational costs. These kinds of summaries help to evaluate the effectiveness of optimization strategies and compare different strategies against each other.

C. Simulator Scalability

Finally, we evaluate the scalability of the extended simulator by performing the following experiment. We simulate a data center consisting of 10–10 000 servers with 4–8 applications hosted on each server (each application is deployed in a separate virtual machine).

For each data center size we simulate the execution of applications optimized with various algorithms: a) even power capping and power shifting, b) proportional power capping and power shifting, and c) ALPACA for power capping and Power Shepherd for power shifting. The amount of simulated work is constant within each data center size, but the time elapsed in simulation varies between 66 and 143 seconds, depending on the number of applications and optimization method.

In order to ensure that our implementation handles different data center configurations, several experiments were conducted using the simulation platform, focusing on addressing proposed power budget algorithms. These algorithms were tested for different number of servers and VMs, as depicted in Table II. Each VM receives only one application instance to execute and has one CPU (*Processor Element*) to run it.

TABLE II
SCALABILITY EVALUATION OF THE EXTENDED SIMULATOR

Servers	Apps	Total Execution Time [s]			Optimization Time [s]		
		Even	Prop.	A + PS	Even	Prop.	A + PS
10	40	0.08	0.07	7.08	0.01	0.01	7.01
10	80	0.21	0.20	81.66	0.01	0.01	81.55
100	400	0.36	0.37	46.11	0.01	0.01	45.82
100	800	1.12	1.15	825.52	0.01	0.01	824.75
1000	4000	3.12	3.28	429.11	0.02	0.01	426.41
1000	8000	13.42	13.82	7570.08	0.02	0.02	7557.92
10000	40000	339.43	339.04	4650.94	0.13	0.13	4309.31
10000	80000	1115.81	1114.12	–	0.15	0.12	–

Table II shows the scalability of the extended simulator. For each combination of data center size (defined by the number of servers and the number of applications) and optimization method, we report the total execution time and optimization time. Optimization time is the accumulated time that the simulator spent on running algorithms for power capping and power shifting. We did not manage to obtain results for the biggest instance with ALPACA and Power Shepherd due to memory and time constraints (the simulation of one minute experiment will take approximately 22 hours). It is worth noting that the total time spent performing *Even* and *Prop.* optimizations was less than 1% of total execution time, and *A+PS* was 98%, on average. However, the simulation execution follows the sequential execution and in real deployment the power capping algorithms will be running on all servers in parallel. Moreover, we report times for the initial step of simulation when we run power capping algorithm for every server. In a long-lasting scenario, the optimization will not be executed at every server at the same time, but only in response to change of the server power budget or application workload.

VI. SUMMARY

In this paper, we have proposed a model for QoS-aware power budgeting and implemented it in a well-established cloud simulator–CloudSim. We have validated the simulation model against the real testbed and achieved high precision with the error not exceeding 5% for both application performance and power consumption. We have also shown the capabilities of the simulator by comparing multiple algorithms for power capping and power shifting. The scalability evaluation has shown that the extended simulator can handle large-scale experiments in a reasonable time. We believe that the proposed extension will be useful for development, evaluation, and comparison of various power budgeting strategies.

ACKNOWLEDGMENT

This work is funded by the Swedish Government's strategic research project eSSENCE and the European Union's Horizon 2020 research and innovation programme under grant agreement No. 732667 (RECAP). This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Finance Code 001. This work has been partially supported by the project "GREEN-CLOUD: Computação em Cloud com Computação Sustentável" (#16/2551-0000 488-9), from FAPERGS and CNPq Brazil, program PRONEX 12/2014. Ahmed Ali-Eldin was partially funded by NSF grant 1836752.

REFERENCES

[1] A. Shehabi *et al.*, "United States data center energy usage report," Tech. Rep. LBNL-1005775, 2016. [Online]. Available: https://eta.lbl.gov/publications/united-states-data-center-energy

[2] "U.S. DoE: The Challenges of Exascale," 2016. [Online]. Available: https://science.energy.gov/ascr/research/scidac/exascale-challenges/

[3] I. Manotas *et al.*, "An empirical study of practitioners' perspectives on green software engineering," in *38th International Conference on Software Engineering (ICSE)*. IEEE, 2016, pp. 237–248.

[4] B. Nie *et al.*, "Characterizing temperature, power, and soft-error behaviors in data center systems: Insights, challenges, and opportunities," in *MASCOTS*. IEEE, 2017, pp. 22–31.

[5] H. Yang *et al.*, "PowerChief: Intelligent power allocation for multi-stage applications to improve responsiveness on power constrained CMP," in *ACM/IEEE ISCA*, vol. 45, no. 2, 2017, pp. 133–146.

[6] L. A. Barroso and U. Hölzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Synthesis lectures on computer architecture*, vol. 4, no. 1, pp. 1–108, 2009.

[7] A. Gandhi *et al.*, "Optimal power allocation in server farms," in *SIGMETRICS*, 2009, pp. 157–168.

[8] J. Leverich and all, "Power management of datacenter workloads using per-core power gating," *IEEE Computer Architecture Letters*, vol. 8, no. 2, pp. 48–51, 2009.

[9] D. Meisner and T. F. Wenisch, "Dreamweaver: architectural support for deep sleep," in *ASPLOS*, vol. 47, no. 4, 2012, pp. 313–324.

[10] X. Zhan and S. Reda, "Power Budgeting Techniques for Data Centers," *IEEE Transactions on Computers*, vol. 64, no. 8, pp. 2267–2278, 2015.

[11] R. N. Calheiros *et al.*, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Pract. and Exp.*, vol. 41, no. 1, pp. 23–50, 2011.

[12] A. Beloglazov *et al.*, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[13] M. Dayarathna *et al.*, "Data center energy consumption modeling: A survey," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 732–794, 2016.

[14] F. D. Rossi *et al.*, "E-eco: Performance-aware energy-efficient cloud data center orchestration," *J. of Net. and Comp. App.*, pp. 83–96, 2017.

[15] H. Lim *et al.*, "Power budgeting for virtualized data centers," in *2011 USENIX Annual Technical Conference*, 2011, pp. 59–72.

[16] R. Cochran *et al.*, "Pack & Cap: Adaptive DVFS and thread packing under power caps," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, 2011, pp. 175–185.

[17] A. Gandhi *et al.*, "Power capping via forced idleness," in *Workshop on Energy Efficient Design*, 2009, pp. 1–6.

[18] H. David *et al.*, "RAPL: Memory power estimation and capping," in *2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*, 2010, pp. 189–194.

[19] W. Felter *et al.*, "A performance-conserving approach for reducing peak power consumption in server systems," in *ICS*, 2005.

[20] H. Chen *et al.*, "Dynamic server power capping for enabling data center participation in power markets," in *Proceedings of the International Conference on Computer-Aided Design*, 2013, pp. 122–129.

[21] S. Caux *et al.*, "IT optimization for datacenters under renewable power constraint," in *Euro-Par*, 2018, pp. 339–351.

[22] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Pract. and Exp.*, vol. 24, pp. 1397–1420, 2012.

[23] T. Gurout *et al.*, "Energy-aware simulation with DVFS," *Simulation Modelling Practice and Theory*, vol. 39, pp. 76–91, 2013.

[24] M. G. Xavier *et al.*, "Modeling and simulation of global and sleep states in ACPI-compliant energy-efficient cloud environments," *Concurrency and Computation: Practice & Experience*, vol. 29, no. 4, 2017.

[25] J. Krzywda *et al.*, "ALPACA: Application performance aware server power capping," in *ICAC*, 2018, pp. 41–50.

[26] S. Govindan and othersn, "Benefits and limitations of tapping into stored energy for datacenters," in *ISCA*, 2011, pp. 341–352.

[27] A. Makaratzis *et al.*, "Energy modeling in cloud simulation frameworks," *Future Generation Computer Systems*, vol. 79, pp. 715–725, 2018.