Analyzing the Effects of Load Distribution Algorithms on Energy Consumption of Servers in Cloud Data Centers

Matthias Splieth, Sascha Bosse, Christian Schulz, Klaus Turowski

Otto von Guericke University Magdeburg
Faculty of Computer Science, Very Large Business Applications Lab
{splieth,sbosse,christian.schulz,klaus.turowski}@ovgu.de

Abstract. Cloud computing has become an important driver for IT service provisioning in recent years. It offers additional flexibility to both customers and IT service providers, but also comes along with new challenges for providers. One of the major challenges for providers is the reduction of energy consumption since today, already more than 50% of operational costs in data centers account for energy. A possible way to reduce these costs is to efficiently distribute load within the data center. Although the effect of load distribution algorithms on energy consumption is a topic of recent research, an analysis-framework for evaluating arbitrary load distribution algorithms with regard to their effects on the energy consumption of cloud data centers is still nonexistent. Therefore, in this contribution, a concept of a simulation-based, quantitative analysis-framework for load distribution algorithms in cloud environments with respect to the energy consumption of data centers is developed and evaluated.

Keywords: cloud computing, load distribution, energy consumption, simulation

1 Introduction

The paradigm of IT service-orientation has led to an increase of the usage of IT services on the part of private and enterprise consumers. In recent years, this trend has been intensified by the advent of cloud computing, which is a concept that allows customers to obtain computing resources on-demand. Cloud service providers make such a deployment process possible by using virtualization technologies to provide customers with exactly the resources they need [1]. Although this paradigm significantly improves the flexibility for customers, still new challenges arise for providers due to the IT service-orientation: the increasing demand for computational power [2] leads to constantly growing data centers since more resources are needed in order to satisfy customer demands, especially during peak times. Even though hardware constantly becomes more energy-efficient [3], the energy consumption of data centers increased by 56% from 2005 to 2010 [4]. Taking into account that energy prices and demand will continue to rise, the energy costs will therefore become the dominant factor in the total cost of ownership of data centers [5]. In addition to that, the climate change and the trend of sustainable power generation make energy efficiency of data centers not only an economic, but also an environmental necessity.

12th International Conference on Wirtschaftsinformatik,
March 4-6 2015, Osnabrück, Germany

Splieth, M.; Bosse, S.; Schulz, C.; Turowski, K. (2015): Analyzing the Effects of Load Distribution Algorithms on Energy Consumption of Servers in Cloud Data Centers, in: Thomas. O.; Teuteberg, F. (Hrsg.): Proceedings der 12. Internationalen Tagung Wirtschaftsinformatik (WI 2015), Osnabrück, S. 287-301

Since the energy costs for the IT equipment and for cooling account for more than 50 % of the total operational costs of data centers [6] and due to the characteristics of cloud computing, an automatic consolidation of resources can be used in order to reduce the energy consumption in cloud data centers. One important method to achieve this goal is to find an optimal technique for load distribution in terms of scheduling – which means placing and migrating – virtual machines within the cloud data center [7, 8]. However, virtualization and rapid elasticity make this task very difficult and cause traditional a priori approaches for analyzing load distribution strategies from other distributed systems (such as grid computing) to fail [9, 10]. Testing load distribution strategies during the operation of data centers is also not feasible since service level agreements may be violated. Hence, there are several methodologies that can be used to investigate and experiment with large scale systems, such as clouds. These methodologies can be in-situ, emulation, benchmarking, simulation or mathematical-analytical modeling [10, 11]. In order to investigate clouds, simulation is a suitable approach [12] that is widely used since, in contrast to the other methodologies, it ensures the repeatability and controllability of experiments [12, 13]. However, existing simulation approaches often have limitations regarding the possibility to test arbitrary load distribution algorithms and to investigate their effects on the energy consumption in cloud data centers, confer [14]. Additionally, modeling limitations in existing approaches restrict their suitability for energy consumption analysis. For instance, several simulators do not consider different energy models for mapping component attributes to energy consumption [14]. In order to resolve these deficits, a quantitative simulation-based analysis-framework is conceptualized, implemented and evaluated in this paper according to the design science research methodology, confer e.g. [15]. This contribution should be a first step for understanding the influences and effects of load distribution algorithms on the energy consumption of cloud data centers. The prototypical implementation of the conceptualized framework might be used by researchers for designing new energy efficient load distribution algorithms, while practitioners could use the prototype as a tool for decision support.

The remainder of this paper is organized as follows: Section 2 provides insights into related work, discussing existing simulation approaches for analyzing cloud data centers. Section 3 presents the conceptual model for the analysis-framework as well as its prototypical implementation. In Section 4, the proposed concept is evaluated. First, the verification of the implemented prototype is presented as a proof-of-concept. Second, an experiment is conducted in order to show the ability of the model to analyze the effects of different load distribution algorithms in specific scenarios. Section 5 concludes the contribution by providing a summary, a discussion of the results of the paper and an outlook on further research activities.

2 Related Work

Since simulation is a suitable approach when investigating large scale systems such as clouds [11, 12], several simulators for investigating clouds have been proposed in recent years. However, these often have limitations regarding the simulation of energy consumption [14] and correspondingly regarding the determination of the effects of load distribution algorithms on energy consumption.

CloudSim is a frequently used toolkit for modeling and simulating cloud computing environments [9]. It enables users to define the parameters of data centers, such as the number of servers and their configuration, the network topology and virtualization-concepts. Furthermore, *CloudSim* enables users to develop algorithms for the placement/migration of virtual machines on/between physical hosts and also provides functionalities for power modeling. However, a major disadvantage of *CloudSim* is the network component. Although the simulator was evaluated by a proof-of-concept with different experiments, the authors of [16] have recently invalidated its network model. Furthermore, there is a bug[1] in the most recent version of *CloudSim* (version 3.0.3). This bug causes different network topologies not to lead to different results.

GreenCloud is a packet-level cloud simulator [17] that has been developed as an extension of the network simulator *ns-2*[2]. In contrast to *CloudSim*, *GreenCloud* has been explicitly designed for determining the energy consumption of cloud data centers. Unlike other simulators, it determines the energy consumption not only for servers, but also for network components. Since *GreenCloud* is based on the packet-level simulator *ns-2*, it is very accurate regarding network aspects, although this leads to high time exposure for simulation and limitations in terms of scalability [16]. Although many relevant components of data centers can be modeled, there are some shortcomings regarding these modeling capabilities. For example, aspects that are highly relevant for cloud computing, such as rapid elasticity, are omitted by *GreenCloud*. Thus, it is questionable if *GreenCloud* is suitable for investigating cloud environments.

iCanCloud is also a packet-level simulator and is based on *OMNET++*[3] in conjunction with its extension-package *INET*[4] [12]. Numerous components of a data center can be modeled with *iCanCloud*. Regarding cloud computing, however, several important aspects are excluded. For example, it does neither provide a feature for simulating virtual machine scheduling nor for aspects that concern energy consumption. Therefore, *iCanCloud* is not suitable for determining the energy consumption of clouds.

SimGrid – confer for example [18] – is an open source library that has originally been developed for the simulation of grids. Since it has only recently been adapted to the simulation of clouds, many aspects that are important for clouds have not been covered yet. For example, rapid-elasticity is not considered by *SimGrid*.

GDCSim is a simulator that can be used for estimating the energy efficiency and thermal properties of data centers and that has been implemented in order to iteratively design green data centers [19]. But since virtualization concepts are not part of the simulator, *GDCSim* is not suitable for investigating clouds.

[1] Confer https://code.google.com/p/cloudsim/issues/detail?id=46 (last accessed: 25.07.2014)
[2] For further information, confer http://www.isi.edu/nsnam/ns/
[3] For further information, confer http://www.omnetpp.org
[4] For further information, confer http://inet.omnetpp.org

MDCSim is a proprietary discrete event simulator that was developed at the Pennsylvania State University [20]. A major disadvantage of *MDCSim* is that all resources in the simulator are modeled as an M/M/1 queue. These assume that service request arrivals are determined by a Poisson process and that their processing time is exponentially distributed. However, Reiss et al. recently demonstrated that simple statistical distributions [21] are not suitable for clouds. Therefore, modeling all components as an M/M/1 queue is restricting and limits the applicability of *MDCSim*.

A weakness that is shared by all simulators is that they do not provide a method for implementing arbitrary load distribution algorithms. For example, each of the aforementioned simulators is limited to centralized load distribution algorithms. Decentralized algorithms are omitted by *CloudSim*, *GreenCloud*, *iCanCloud and SimGrid*. For *GDCSim* and *MDCSim*, no statement can be made regarding this aspect, since the possibility of modeling decentralized load distribution algorithms is not mentioned in the respective publications. A further weakness that all simulators have in common is that many parameters need to be defined prior to a simulation run but should be varying at runtime. For example, the number of virtual machines needs to be defined in every simulator before starting an experiment. Especially in infrastructure as a service (IaaS) environments, however, virtual machines can be started at any time and therefore the exact number of virtual machines cannot be known beforehand.

3 Conceptual Model

After having analyzed related work, it can be stated that none of the considered simulators provides the opportunity to test arbitrary load distribution algorithms with regard to their effects on energy consumption. Therefore, a novel concept that aims to address this problem is introduced in the first part of this section. Subsequently, the prototype implementation of this concept is described.

3.1 Component Model

In order to analyze the effects of load distribution to energy consumption in cloud data centers, the components that are relevant for this problem have to be modeled. These components include, among other things, the energy consuming components in a data center such as network, server and cooling components [22]. A power and energy model is assigned to each energy consuming component. This model maps the current state of a component to its power consumption, so it can be used to compute the energy consumption of a component by integrating the respective power consumption over time. A load distribution algorithm is then used to control the migration of virtual machines between servers. The components of this concept and their relations are illustrated as a UML class diagram in Fig. 1 and are explained in detail in the following.

Server. The individual servers provide the resources of a cloud data center. The resources of a server are determined by several entities, such as computing power or network adapters. The resources provided by a server are allocated by virtual machines that are used in order to serve user requests.

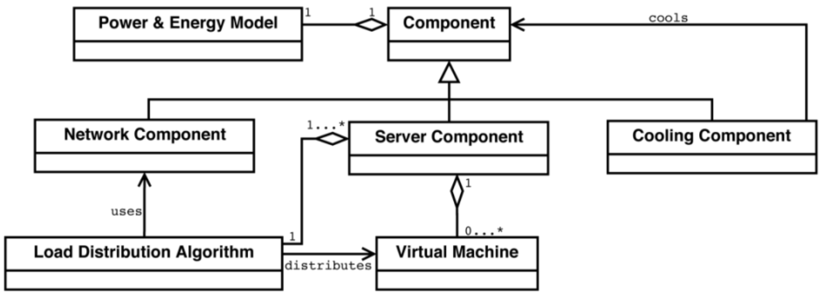

Fig. 1. High-Level Concept as a UML Class Diagram

Network. The *"Network"* components define a structure that connects the single servers within a data center. Among simulating the communication between components, the network is used to distribute load between servers. In order to define such a structure, different network models can be used. According to [16], three different types of network models can be distinguished: delay models, flow-level models, and packet-level models. Packet-level models are very accurate since they simulate the movements of all network packets. However, this level of detail is not needed when simulating clouds [16]. In a flow-level model, each communication is simulated as an entity which leads to a higher scalability. Delay models simulate network delays between servers. Such models are not very accurate regarding the results that concern network-issues, but very scalable. Besides such models, also network devices and different types of network topologies are captured by these components.

Cooling. Cooling is a crucial task in data centers since cooling systems remove the heat generated by the equipment [3]. In order to cool the equipment, some hierarchy of loop systems is needed, which brings in a cold medium that is used for heat exchange and needs to be cooled afterwards. This can be achieved by different devices, confer [3]. Therefore, some type of cooling unit as well as a thermal model is important in order to simulate cooling processes in data centers.

Power & Energy Model. Energy and power models can be used in order to estimate the consumption of components or of an entire infrastructure [5]. The power model of a component depicts the relation between the current state of the component and its power consumption. A power-consuming component c can refer to a component of the set of servers S, the set of network components N or the set of cooling components C. Then, $P_c(t)$ denotes the power consumption of the component c at the time t. Let $\boldsymbol{a}_c(t)$ be a vector of attributes of the component c at time t. The power model of component c, denoted m_c, maps the attribute vector of c to P_c according to equation 1.

$$\mathrm{P_c}(t) = \mathrm{m_c}(\mathbf{a}_c(t)), c \in S \cup N \cup C \tag{1}$$

The total energy consumption E_c of a component c over the simulation time T can be computed by integrating the power consumption over time (equation 2).

$$\mathrm{E_c}(T) = \int_0^T \mathrm{P_c}(t)dt \tag{2}$$

Virtual Machine. A virtual machine (VM) is created by a user request and processes the workload that is generated by the user. Therefore, a certain amount of resources is initially allocated on the server on which the virtual machine is placed. A scaling of the allocated resources is conducted with respect to the actual workload of the user. Virtual machines can be moved between different physical servers at runtime.

Load Distribution Algorithm. Load distribution in clouds can either refer to task scheduling or to virtual machine scheduling [13]. Task scheduling is used to assign tasks to virtual machines while virtual machine scheduling refers to the placement or migration of VMs.

Initially, a virtual machine is placed on a server that fulfills specific criteria that are defined by an algorithm. Under certain conditions, a VM may be migrated to another server in the data center. Regarding the migration of a VM, an algorithm addresses three sub problems [7]: (1) when to migrate VMs, (2) which VMs to migrate, and (3) where to migrate VMs. In the course of this contribution, load distribution always refers to virtual machine scheduling since VM scheduling can significantly contribute to lowering the costs for energy and is furthermore one of the major challenges regarding cloud computing [2].

3.2 Prototypical Implementation

In the prototypical implementation, the previously defined high-level components of the concept are instantiated as described in the following. The prototype was implemented in Java using the multi-method simulation tool *AnyLogic*[5] as a basis.

Server. The components of a server that are instantiated in the prototypical implementation are CPU, main memory, and disk space, which define the amount of available resources of a server.

Network. Since the level of detail provided by packet-level models is not necessary for the purpose of this paper and since a flow-level model is only needed when network contention is to be simulated [16], a delay-model is used in this contribution in order to implement a feasible model. In this model, all servers of a data center are arranged in a graph that represents a grid topology. If a VM is transferred between

[5] For further information, confer http://www.anylogic.com

two servers, all connections between the source server and the destination server defined by a routing algorithm have to process a load according to the allocated VM resources which leads to a delay.

Cooling. A model for cooling is not included in the prototypical implementation of the conceptual framework. In general, a computational fluid dynamics (CFD) approach is used in order to model cooling systems in data centers. Nonetheless, these models can be very complex while still showing a root mean square error of up to 100% [23]. Therefore, and since cooling is not the focus of this contribution, it has been excluded from the prototypical implementation.

Power & Energy Model. For the proof-of-concept developed in this contribution, a power model was implemented for servers only – confer equation 3.

$$\mathrm{P_c}(t) = \begin{cases} \mathrm{m_c}(\mathbf{a}_c(t)) \\ 0, \text{otherwise} \end{cases} \tag{3}$$

There are different ways to build the attribute vector $\boldsymbol{a}_c$ since servers have multiple energy consuming components [3, 24]. Within a server, the CPU is the component that consumes the greatest proportion of energy [3, 24] and also has the greatest dynamic power range [25] while further studies showed that disk and network components almost have a constant consumption [26]. Therefore, the CPU power consumption is used for approximating the server power consumption like for instance also done in Google data centers [24]. The power model applied in the prototype builds the attribute vector based on the CPU utilization since such models are a common and flexible implementation of power models, confer [27]. Since there are different positions in the scientific discussion how to model the relationship between utilization and power consumption (confer e.g. [24, 28]), empirical data from the industry-standard benchmark *SPECpower_ssj2008*[6] was analyzed. This benchmark was chosen since it is an industry standard and therefore provides a lot of data for different types of servers.

The results of this analysis indicate that there are linear as well as non-linear relations between the utilization and the power consumption of servers. There are examples of linear (HP ProLiant DL385-G6), convex (Fujitsu PRIMERGY TX300 S7) and concave (ASUS RS160-E5) functions mapping utilization to power consumption as illustrated in Fig. 2. In order to be able to model linear as well as non-linear relations of utilization and power, the following power model is proposed (equation 4).

$$\mathrm{m_c}(t) = \mathrm{m_c}(P_{idle}, P_{max}, u(t), r) = P_{idle} + (P_{\max} - P_{idle}) \cdot \mathrm{f_r}(u(t)) \tag{4}$$

P_{idle} denotes the constant amount of power that is consumed even if the server is idle. P_{max} defines the upper bound of the power consumption when the server is fully utilized. *u(t)* is the CPU usage ratio at a given time *t* with $u \mapsto [0,1]$. f_r describes an

[6] http://www.spec.org/power_ssj2008/

arbitrary function that characterizes the dependency of the CPU's power consumption and its utilization using the shape parameter *r*.

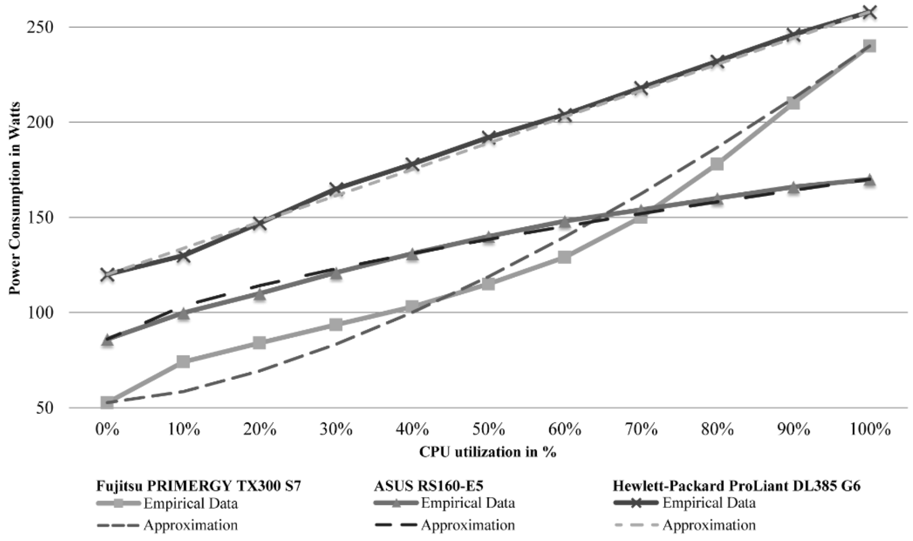

Fig. 2. Three Empirical Power Models and their Approximations using Equations 4 and 5

This power model is a generalization of the power model proposed in [24], where the authors used $f_r(u(t)) = 2\cdot u(t) - u(t)^r$ in order to fit non-linear power profiles. However, the power model defined in equation 5 is used in this paper since, in comparison to the empirical values, the root mean square error is smaller for the model defined in equation 5 than for the model proposed in [24].

$$\mathrm{f}_{\mathrm{r}}(u(t)) = \mathrm{u}(t)^{r} \tag{5}$$

Regarding the servers mentioned before, we computed r=1 for the HP ProLiant DL385 G6, r=1.5 for the Fujitsu PRIMERGY TX300 S7, and r=0.68 for the ASUS RS160-E5 in order to minimize the root mean square error between empirical and predicted values. In Fig. 2, the graphs of the computed power models and the resp2ective empirical values are illustrated.

Using the computed power consumption for each simulation time step, the power consumption function of the component can be derived by connecting the single values with a linear spline. Hence, the energy consumption of a component *c* is defined by the mean consumed power P_m over the simulation time *T* as $E_c = P_m \cdot T$.

Virtual Machine. In the prototypical implementation, virtual machines have a plain lifecycle which is a simplified version of the lifecycle presented in [29]. The lifecycle starts with the creation of a virtual machine on a server that provides enough resources. VMs are created to process user requests arriving in the system according to a defined random distribution. Subsequently, the VM will stay active for a randomly distributed time until it can be shut down [29]. Between creation and shutdown, a scaling may be conducted according to the actual demand of a user, which ensures that rapid elasticity and on-demand self-service are, in contrast to the simulators outlined in Section 2, supported by the prototype. This time span is also defined by a random distribution. Virtual machines can be moved to other servers. During this process, a virtual machine exists on the source server as well as on the destination server. First, the needed resources are allocated on the target server. Second, the VM is copied to the target server. Afterwards, the VM on the source server will be removed. Therefore, this process includes an expensive moment regarding energy consumption since two servers consume energy for the same virtual machine during the migration [30].

Load Distribution Algorithms. Load distribution can be handled in centralized or decentralized manner. In the former, a single load distribution instance on a coordination server orchestrates in the data center. In the latter, a system-wide choreography distributes the load by applying load distribution instances of the load distribution algorithm on every server [31]. Both approaches are implemented in the prototype. In order to incorporate arbitrary load distribution algorithms, five policies are modeled which provide a generic interface for implementing arbitrary load distribution algorithms [31]. These policies are defined as follows:

- The **transfer policy** defines when a server is meant to initiate a load distribution. This can either refer to "*threshold-based*" algorithms or to "*relative*" algorithms. "*Threshold-based*" algorithms will initiate load distribution as soon as a certain threshold is exceeded. In "*relative*" algorithms, loads will be swapped as long as there is an imbalance between servers.
- The **selection policy** provides criteria for the selection of VMs on a server that are meant to be migrated. A common implementation is to select the VMs via a *"First In, First Out"* or via a *"Last in, Fist out"* approach.
- The **location policy** is concerned with the selection of an appropriate destination server. A "*local*" execution implies that only servers in a n-neighborhood are considered whereas "*global*" approaches consider all servers in the network.
- The **information policy** declares what information about the state of a server is collected and when and where this information is collected. In "*demand-driven*" approaches, servers only start to collect information about potential partner servers once they are determined as initiating servers, whereas in a "*state-driven*" approach, all servers immediately spread state changes to other servers. Information about potential target servers can also be collected by conducting a "*periodic*" approach. Since the period interval is not dynamically adjusted according to the current load of a system, busy systems tend to slow down due to the additional load.
- The **initiation policy** defines which server initiates a load distribution task. *"Sender-initiated"* algorithms will be triggered by reaching an upper threshold and try to distribute their loads to other servers. In a *"receiver-initiated"* approach, load distribution will be conducted if the load of a server falls short of a lower threshold. A *"symmetric"* approach combines both approaches in order to take all advantages.

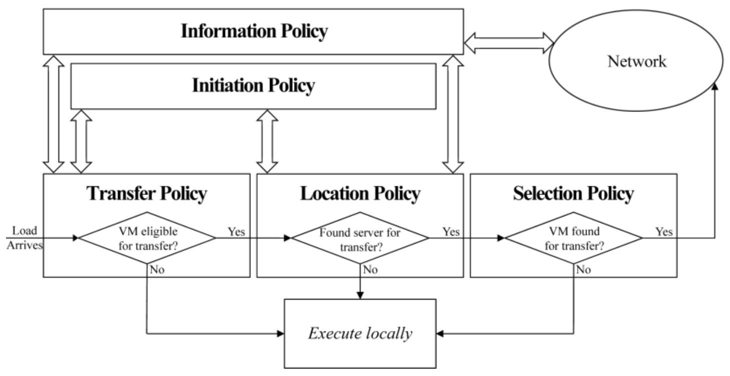

Fig. 3. Interaction between the different Load Distribution Policies

Fig. 3 illustrates how these policies interact. In the prototype, different types of algorithms are instantiated. These can be distinguished into *direct* and *iterative* algorithms. *Direct* algorithms determine sender or receiver servers and exchanges loads between two servers. The *sender-initiated* algorithm belongs to the group of direct algorithms and was implemented in the prototype. In *iterative* algorithms, servers in a *n*-neighborhood continuously try to distribute loads among each other [31]. A widespread representative of those is the *gradient model* algorithm. The realization of the policies for both algorithms is depicted in Table 1.

Table 1. Realized Policies for the Implemented Load Distribution Algorithms

Policy	Sender-Initiated	Gradient Model
Transfer	Threshold-based	Relative
Selection	First in, First out	First in, First out
Location	Global	Local
Information	Demand-driven	Demand-driven
Initiation	Sender-initiated	Sender-initiated

4 Evaluation

After the prototypical implementation of the conceptual framework that is described above, its correctness and suitability are investigated in the course of this section. Therefore, a verification and an experiment were conducted.

In order to verify the implemented prototype, first the behavior of the simulation model was tested by conducting an event validity test and a fixed value test. The former was carried out in order to compare the occurrence of events in the simulation model to the expected behavior. The latter was used in order to eliminate any stochastic influence from the simulation model. Thus, the simulation results can be compared to analytically obtained results [32]. Conducting these tests, the correctness of VM and server behavior as well as of the implemented load distribution algorithms could be verified.

In addition to that, an experiment was performed in order to demonstrate the ability of the developed concept to support the analysis of load distribution algorithms and their energy consumption in specific scenarios. Therefore, two scenarios were defined, both referring to real-world cloud infrastructure services: in the first scenario, a high performance cloud was modeled (high performance computing as a service – HPCaaS [33]) in which consumers demand VMs with plenty of resources to process problems with high complexity. In the second scenario, an IaaS provider was modeled for micro VM hosting that allows users to obtain VMs with limited resources in order to save costs (referred to as "micro infrastructure as a service" – μIaaS). VM instances similar to the VMs defined for the HPCaaS scenario as well as in the μIaaS are provided, for example, in the popular cloud platforms Amazon EC2[7], Google Compute Engine[8] and Rackspace[9].

The different parameters for both scenarios are presented in Table 2. In both scenarios, the data center was modeled as a composition of two of the servers described above, the ASUS RS160-R5 (4x2.5 GHz, 16 GiB RAM, 100 GB HDD) and the Fujitsu PRIMERGY TX300 S7 (8x2.2 GHz, 8 GiB RAM, 100 GB HDD. The parameters have been chosen to make the scenarios comparable in terms of the processed workload. Hence, they lead to a mean CPU utilization of about 50% in both scenarios.

Table 2. Experimental Setup

Parameter		**Scenario 1 - HPCaaS**	**Scenario 2 - μIaaS**
Requests	Time to request	~exp(30 min)	~exp(10 min)
	Time to scaling	~exp(30 min)	~exp(30 min)
	Time to shutdown	~exp(2 d)	~exp(8 h)
Resources	CPU	~Gaussian(2 GHz, 200 MHz)	~Gaussian(750 MHz, 150 MHz)
	RAM	~Gaussian(2 GiB, 512 MiB)	~Gaussian(512 MiB, 128 MiB)
	HDD	~Gaussian(10 GB, 2 GB)	~Gaussian(2.5 GB, 250 MB)
Server count	ASUS RS160	19	77
	Fujitsu PRIMERGY	77	19

The two scenarios were simulated each with the iterative gradient model and the direct sender-initiated load distribution for 10,000 minutes (about one week) in 1,000 replications. For the confidence level, α=0.1 was chosen as significance level. The

[7] Confer the instance types at http://aws.amazon.com/ec2/instance-types/
[8] Confer the instance types at https://developers.google.com/compute
[9] Confer the instance types at http://www.rackspace.com/cloud/pricing/

resulting confidence intervals of the energy consumption in the experiments are presented in Fig. 4. Since the confidence intervals do not intersect, the results are significantly different. In the HPCaaS scenario, the gradient model provides better results in terms of data center energy consumption than the sender-initiated load distribution (mean value 2,187 kWh in comparison to 2,208 kWh), whereas in the µIaaS scenario, the situation is vice versa (2,184 kWh to 2,106 kWh).

In the case of the sender-initiated load distribution, heavy loaded servers start to distribute VMs to slightly loaded servers. However, due to the larger amount of allocated resources in the HPCaaS scenario, VM migration from various overloaded servers to a less loaded target server may consequently overload this server. The reason for this is that lower loads simply do not lead to overloaded states as quickly as higher loads. Since the fact that slightly loaded servers are thus more likely to become overloaded and have to distribute loads again in the next step, the sender-initiated algorithm is more stable with smaller VM sizes, such as those defined in the µIaaS scenario. Therefore, the algorithms archive opposite results in the respective scenarios.

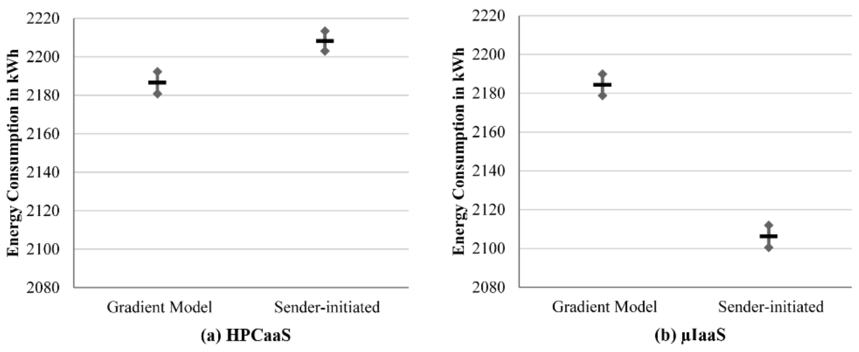

Fig. 4. Experimental Results of Energy Consumption for the tested Algorithms

The experimental results show that the prototype of the developed concept has the ability to analyze the effect of different load distribution algorithms on energy consumption. Furthermore it can be pointed out that different scenarios lead to different results while applying the same load distribution algorithm. Hence, it cannot be decided which load distribution algorithm is most energy-efficient in general. A concrete load distribution algorithm must always be analyzed with respect to the concrete scenario.

5 Conclusion

In this contribution, a quantitative, simulation-based analysis-framework was conceptualized. It was designed to investigate the influences and effects of load distribution algorithms on the energy consumption of cloud data centers. In order to demonstrate the ability of the conceptual framework and its prototypical implementation, an experiment that examined two different scenarios in heterogeneous data centers was conducted. The experiment revealed that the effect of load distribution algorithms on the energy consumption in clouds seems to be strongly dependent on the configuration of the data center, for example in terms of hardware and offered services.

The conceptualized prototype allows to model specific scenarios for heterogeneous cloud data centers and to evaluate their energy consumption. The presented approach allows for analyzing mean, worst- or best-case energy consumption over a simulated lapse of time. Different types of load distribution algorithms can be compared in terms of their energy efficiency. Additionally, the implemented prototype can be used in order to simulate changes in the cloud data center. For instance, a varying number of different servers could be simulated in order to foresee their effects on the overall energy consumption. Furthermore, a sensitivity analysis could be performed by changing the operational profile (e.g. the runtime of services) in order to analyze the energy consumption under varying conditions. Therefore, the conceptual framework can be used as a basis for a decision support system for managing the energy consumption of cloud data centers. Supplementary, it can be used by researchers to evaluate new algorithms and architectures with regard to energy consumption, prior to a deployment in a real environment.

The developed concept addresses the shortcomings of existing approaches as revealed in Section 2. Arbitrary load distribution algorithms can be implemented and investigated by using a generic policy scheme, even centralized and decentralized algorithms can be implemented. The proposed energy model is very flexible and can map the different types of power profiles that have been identified in the course of an analysis of empirical data from the *SPECpower_ssj2008* benchmark. In contrast to existing simulation approaches, relevant aspects of clouds can be simulated, such as the unilateral scaling of resources according to a user's current demand (rapid elasticity & on-demand self service).

Nevertheless, there are also disadvantages that derive from the simulation approach. Since each replication only describes one possible system behavior, simulation results can only be approximated by confidence intervals using a high number of replications. Depending on the experiment, this can result in a high effort in terms of computing time and computational resources that are needed. However, massive parallelization could mitigate this disadvantage. In the initial implementation, several simplifying assumptions were made in order to reduce the complexity of the prototype. For example, a simple VM lifecycle has been implemented and cooling has been omitted. Regarding the energy consumption, only the CPU was considered as consumer. Although the CPU is suitable for approximating the power consumption of servers, the proportion is likely to decrease in future. Hence, also other components of servers need to be considered, especially memory and disks [3, 5, 25].

Future work should identify the influences of the respective aspects of a scenario. For example, such aspects can be the application of energy models, the effects of heterogeneity and homogeneity of components, and parameters of the operational profile, such as the runtime of services. These aspects need to be addressed by extending the conceptual model. Additionally, the load distribution scheme needs to be extended in order to include genetic algorithms since these cannot be described by the current scheme. Regarding the component that addresses power & energy models, it should be verified that polynomial models are sufficient for modeling non-linear dependencies between CPU utilization and power consumption. Otherwise, another approximation for non-linear dependencies needs to be defined. Other aspects of cloud data centers, such as performance and dependability, should be integrated into the concept in order to be able to consider service level agreements. Finally, the components regarding cooling need to be integrated in future iterations of the concept since these were excluded in the initial prototype implementation.

References

1. Mell, P., Grance, T.: The NIST Definition of Cloud Computing. Technical report, National Institute of Standards and Technology (NIST) (2011)
2. Zhang, Q., Cheng, L., Boutaba, R.: Cloud computing: state-of-the-art and research challenges. Journal of Internet Services and Applications. 1, 7–18 (2010)
3. Barroso, L.A., Clidaras, J., Hölzle, U.: The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Second edition. Synthesis Lectures on Computer Architecture. 8, 1–154 (2013)
4. Koomey, J.: Growth in data center electricity use 2005 to 2010. Technical report, Analytics Press (2011)
5. Orgerie, A.-C., De Assuncao, M.D., Lefevre, L.: A Survey on Techniques for Improving the Energy Efficiency of Large Scale Distributed Systems. ACM Computing Surveys. 46, 1–35 (2014)
6. Hamilton, J.: Cooperative Expendable Micro-Slice Servers (CEMS): Low Cost, Low Power Servers for Internet-Scale Services. In: 2009 Conference on Innovative Data Systems Research (2009)
7. Beloglazov, A., Buyya, R.: Energy Efficient Resource Management in Virtualized Cloud Data Centers. In: 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. pp. 826–831 (2010)
8. Vaquero, L.M., Rodero-Merino, L., Buyya, R.: Dynamically Scaling Applications in the Cloud. ACM SIGCOMM Computer Communication Review. 41, 45–52 (2011)
9. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience. 41, 23–50 (2011)
10. Sakellari, G., Loukas, G.: A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing. Simulation Modelling Practice and Theory. 39, 92–103 (2013)
11. Gustedt, J., Jeannot, E., Quinson, M.: Experimental Methodologies for Large-Scale Systems: a Survey. Parallel Processing Letters. 19, 399–418 (2009)
12. Núñez, A., Vázquez-Poletti, J.L., Caminero, A.C., Castañé, G.G., Carretero, J., Llorente, I.M.: iCanCloud: A Flexible and Scalable Cloud Infrastructure Simulator. Journal of Grid Computing. 10, 185–209 (2012)
13. Beloglazov, A., Abawajy, J.H., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. Future Generation Computer Systems. 28, 755–768 (2012)
14. Splieth, M., Bosse, S., Turowski, K.: Analysis of Simulation Tools for Determining the Energy Consumption of Data Centers for Cloud Computing. In: 13th International Conference on Modeling and Applied Simulation. pp. 149–158 (2014)
15. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design Science in Information Systems Research. Management Information Systems Quarterly. 28, 75–105 (2004)

16. Velho, P., Schnorr, L.M., Casanova, H., Legrand, A.: On the validity of flow-level tcp network models for grid and cloud simulations. ACM Transactions on Modeling and Computer Simulation. 23, 23 (2013)
17. Kliazovich, D., Bouvry, P., Khan, S.U.: GreenCloud: a packet-level simulator of energy-aware cloud computing data centers. The Journal of Supercomputing. 62, 1263–1283 (2012)
18. Casanova, H., Legrand, A., Quinson, M.: SimGrid: A Generic Framework for Large-Scale Distributed Experiments. In: 10th International Conference on Computer Modeling and Simulation. pp. 126–131 (2008)
19. Gupta, S.K., Gilbert, R.R., Banerjee, A., Abbasi, Z., Mukherjee, T., Varsamopoulos, G.: GDCSim: A Tool for Analyzing Green Data Center Design and Resource Management Techniques. In: International Green Computing Conference and Workshops. pp. 1–8 (2011)
20. Lim, S.-H., Sharma, B., Nam, G., Kim, E.-K., Das, C.R.: MDCSim: A Multi-tier Data Center Simulation Platform. In: International Conference on Cluster Computing and Workshops. pp. 1–9 (2009)
21. Reiss, C., Tumanov, A., Ganger, G.R., Katz, R.H., Kozuch, M.A.: Heterogeneity and Dynamicity of Clouds at Scale: Google Trace Analysis. In: ACM Symposium on Cloud Computing (2012)
22. Brown, R., Koomey, J., Masanet, E., Nordman, B., Tschudi, B., Shehabi, A., Stanley, J., Sartor, D., Chan, P., Loper, J., Capana, S., Hedman, B., Duff, R., Haines, E., Sass, D., Fanara, A.: Report to Congress on Server and Data Center Energy Efficiency: Public Law 109-431. Technical report, Lawrence Berkeley National Laboratory (2007)
23. Rambo, J., Joshi, Y.: Modeling of data center airflow and heat transfer: State of the art and future trends. Distributed and Parallel Databases. 21, 193–225 (2007)
24. Fan, X., Barroso, L.A., Weber, W.-D.: Power Provisioning for a Warehouse-sized Computer. In: ACM International Symposium on Computer Architecture. pp. 13–23 (2007)
25. Barroso, L.A., Hölzle, U.: The Case for Energy-Proportional Computing. IEEE Computer. 40, 33–37 (2007)
26. Heath, T., Diniz, B., Carrera, E.V., Meira, W., Jr, Bianchini, R.: Energy Conservation in Heterogeneous Server Clusters. In: 10th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. pp. 186–195 (2005)
27. Rivoire, S., Ranganathan, P., Kozyrakis, C.: A Comparison of High-level Full-system Power Models. In: Conference on Power Aware Computing and Systems. pp. 3–3 (2008)
28. Raghavendra, R., Ranganathan, P., Talwar, V., Wang, Z., Zhu, X.: No “Power” Struggles: Coordinated Multi-level Power Management for the Data Center. ACM SIGARCH Computer Architecture News. 36, 48–59 (2008)
29. Willems, C., Meinel, C.: Online Assessment for Hands-On Cyber Security Training in a Virtual Lab. In: 2012 IEEE Global Engineering Education Conference. pp. 1–10 (2012)
30. Orgerie, A.-C., Lefevre, L., Gelas, J.-P.: Demystifying energy consumption in Grids and Clouds. In: Green Computing Conference. pp. 335–342 (2010)
31. Wu, J.: Distributed System Design. CRC Press (1998)
32. Sargent, R.D.: Verification and validation of simulation models. Journal of Simulation. 7, 12–24 (2013)
33. Mauch, V., Kunze, M., Hillenbrand, M.: High performance cloud computing. Future Generation Computer Systems. 29, 1408–1416 (2013)