Int. J. Appl. Math. Comput. Sci., 2014, Vol. 24, No. 3, 535–550
DOI: 10.2478/amcs-2014-0039

amcs

USING A VISION COGNITIVE ALGORITHM TO SCHEDULE VIRTUAL MACHINES

JIAQI ZHAO *, YOUSRI MHEDHEB **, JIE TAO **, FOUED JRAD **, QINGHUAI LIU *, ACHIM STREIT **

* School of Basic Science
Changchun University of Technology, Yan An Street 2005, 130012 Changchun, China
e-mail: scorpiozhao@yahoo.com.cn, liuqinghuai21@126.com

** Steinbuch Center for Computing
Karlsruhe Institute of Technology, Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany
e-mail: {yousri.mhedheb, jie.tao, foued.jrad, Achim.streit}@kit.edu

Scheduling virtual machines is a major research topic for cloud computing, because it directly influences the performance, the operation cost and the quality of services. A large cloud center is normally equipped with several hundred thousand physical machines. The mission of the scheduler is to select the best one to host a virtual machine. This is an NP-hard global optimization problem with grand challenges for researchers. This work studies the Virtual Machine (VM) scheduling problem on the cloud. Our primary concern with VM scheduling is the energy consumption, because the largest part of a cloud center operation cost goes to the kilowatts used. We designed a scheduling algorithm that allocates an incoming virtual machine instance on the host machine, which results in the lowest energy consumption of the entire system. More specifically, we developed a new algorithm, called vision cognition, to solve the global optimization problem. This algorithm is inspired by the observation of how human eyes see directly the smallest/largest item without comparing them pairwisely. We theoretically proved that the algorithm works correctly and converges fast. Practically, we validated the novel algorithm, together with the scheduling concept, using a simulation approach. The adopted cloud simulator models different cloud infrastructures with various properties and detailed runtime information that can usually not be acquired from real clouds. The experimental results demonstrate the benefit of our approach in terms of reducing the cloud center energy consumption.

Keywords: cloud computing, vision cognitive algorithm, VM scheduling, simulation.

1. Introduction

The concept of cloud computing (Armbrust *et al.*, 2009; Mell and Grance, 2013; Wang *et al.*, 2010a; 2013a) was introduced by Amazon with its Elastic Compute Cloud EC2 (Amazon, 2013a) and Simple Storage Service S3 (Amazon, 2013b). This concept was immediately accepted by both the industry and the academy due to its special features of provisioning computing capacities as services in a self-service and easy-to-use manner. According to the definition of the National Institute of Standards and Technology (NIST), cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (networks, servers, storage, applications, and so on) that can be rapidly provisioned and released with minimal management effort or service provider interaction (Mell and Grance, 2013).

The services provided by existing cloud infrastructures can be classified into three models, i.e., Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) (Mell and Grance, 2013; Kahn *et al.*, 2013) . IaaS targets on an on-demand provision of the computational resources, normally in the form of virtual machines. Customers can install software packages on the machines to establish their own computing environments. SaaS provides customers with a functionality of using the provider's applications in the form of Web services that run centrally on the computing infrastructure of the service provider. PaaS targets on an entire platform including the hardware and the application development environment, which

allows the users to develop their applications using the provider's computing platform. Existing clouds are mostly IaaS-featured. Examples include Amazon EC2, SmartCloud (IBM, 2013), Flexiscale (FlexiScale Ltd., 2013), as well as academic implementations like Eucalyptus (Nurmi *et al.*, 2008), OpenNebula (Sotomayor *et al.*, 2008), Nimbus (Keahey and Freeman, 2008) and OpenStack (Openstack, 2013).

Cloud computing is enabled by several underlying technologies, where the virtualization technology is especially important. Actually, the virtualization approach was proposed in the late 1950s with an initial goal of running different binary codes on the expensive hardware. Today, cloud computing makes virtualization a hot topic because it relies on this technology to provide on-demand, elastic computing resources. The virtualization itself is also developed from a simple approach to a mature technique with a standard virtualization layer as middleware. This additional system layer, the so-called hypervisor or virtual machine monitor (Rosenblum and Garfinkel, 2005), exists between the operating system and the system hardware to enable the running of multiple operating systems on the same physical machine. The functionality of a hypervisor is to provide each operating system with a virtual set of hardware components, including the processor, the memory and devices. Xen (Barham *et al.*, 2003), VMware (VMware Inc., 2013) and KVM (KVM, 2013) are the three well-known hypervisors among the existing implementations. The Xen hypervisor is an open source development and is widely used for research purposes. KVM is also an open source implementation. It provides virtualization capacities directly in the Linux kernel, achieving the thinnest hypervisor of only a few hundred thousand lines of code. VMware is a commercial product and is mainly used for server consolidation.

Virtualization brings a new computing methodology but also new research issues, especially with respect to performance (González-Vélez and Kontagora, 2011). One of the major research topics for virtualized infrastructures is VM scheduling that decides which physical machine shall host a new virtual machine. The scheduling result directly influences the performance and efficiency of the entire virtual infrastructure.

Scheduling is not a novel topic. This issue exists in various scenarios, like task scheduling in parallel and distributed systems and job scheduling in computing grids (Kołodziej and Xhafa, 2011). Researchers have also proposed a number of algorithms for tackling different scheduling problems. The design of a VM scheduling scheme on the cloud, however, is more challenging because the following facts with computing clouds:

- large system scale with several hundred thousand computing nodes (and this scale has been growing up);
- business model that expects a low operation cost and a high Quality of Services (QoS) with respect to the response time on a VM request, system performance, service availability, etc.;
- large number of customers/users, meaning also a large number of VM requests;
- workloads cannot be pre-defined or predicted.

Considering these features we proposed a scheduling scheme and a global optimization algorithm to address virtual machine allocation on the cloud. Our first concern is the energy consumption because large part of the cloud center operation cost is contributed by this factor. The second consideration is the service quality, especially the response time to a VM request. For this purpose the scheduling process must be fast and efficient with low overheads, thus to be capable of handling unexpected workloads. We also aim at maintaining the performance of the running virtual machine. With these concerns as background we designed and implemented the scheduling policies and algorithms. In summary, our approach contributes to the stand of research on VM scheduling with the following features:

- *Thermal-aware scheduling:* The temperature of a host machine is modelled and involved in the VM scheduling to prohibit a processing core from reaching a temperature limit. A higher temperature of the processor not only results in more energy consumption but also increases the possibility of hardware failure.
- *Load-aware scheduling:* The scheduling scheme guarantees that none of the physical hosts is overloaded in order to maintain the performance of applications running on a virtual machine.
- *Energy-based global optimization*: A vision cognitive algorithm was developed to search for the best host for a VM instance with the result of minimal energy consumption. The novel algorithm uses overlook points to reduce the number of candidate hosts and hence converges faster than the traditional pairwise comparison methods.

All of these policies are specifically designed for lower operation cost as well as service performance and quality of a cloud center. Thermal-aware targets on the operation cost and service availability with balancing temperature across processing cores, load-aware addresses the performance issue of cloud services, and global optimization aims at a minimal energy consumption while keeping the response time possibly low for ensuring the service quality.

The proposed approach was validated using CloudSim (Calheiros *et al.*, 2011), a well-know simulator for research works on cloud computing. The reason for applying a simulator rather than a real cloud is that validating this concept requires detailed information about the target cloud, including the system configuration, the runtime workload as well as the temperature of all physical hosts. These details cannot be gained from real clouds. On the other hand, the simulator allows us to model various system scales and use scenarios. Therefore, it is the best choice for us to study the feasibility and functionality of the proposed algorithms.

In addition to the proposed scheduling scheme, we also implemented several other scheduling policies for a comparative study. The experimental results show that the proposed approach performs better than others with the lowest energy consumption while the service quality is granted in most cases.

The remainder of the paper is organized as following. Section 2 introduces the related work on scheduling approaches and global optimization algorithms. Section 2 describes the design of the thermal-aware and load-aware scheduling scheme. This is followed by describing the vision cognitive algorithm in Section 3. The implementation on top of CloudSim is then detailed in Section 5. Section 6 shows the experimental results with discussions. Finally, the paper concludes in Section 7, with a brief summary and several future directions.

2. Related work

This research work mainly addresses two research topics, i.e., task scheduling and global optimization algorithms. In this section the related works in both areas are described in three categories: cluster task scheduling, VM scheduling and global optimization algorithms.

2.1. Cluster task scheduling. Scheduling is a hot topic in the area of parallel and distributed computing. A common scenario is the scheduling of parallel tasks on a cluster system (Valentini *et al.*, 2013; Min-Allah *et al.*, 2012). There are a number of production schedulers that are widely used in both commercial and research computing infrastructures. Condor (Tannenbaum *et al.*, 2002), Torque (Staples, 2006; Adaptive Computing Inc., 2013), PBS (Altair Engineering Inc., 2013) and SUN/Oracle Grid Engine (Gentzsch, 2001; ORACLE, 2013) are several representative examples.

In addition to these mature products, researchers have been studying on scheduling issues for various use cases, with the earlier implementation focusing on load-balancing and recently with energy-awareness (Hsu and Feng, 2005; Kołodziej, *et al.*, 2013a; Wang *et al.*, 2012a; 2011b; Zhang *et al.*, 2013) as additional features.

Kim (1988) as well as Wu and Gajski (1990) introduce a graph clustering technique that applies static scheduling heuristics to schedule parallel tasks. Given a task graph, "clustering" is the process of mapping task graph nodes onto labelled clusters. All tasks of the same cluster are executed on the same processor. This approach is similar to the list scheduling algorithm described by Li and Huang (2007) as well as Mtibaa *et al.* (2007). List-based scheduling algorithms assign priorities to tasks and sort tasks into a list ordered in decreasing priority. The tasks are then scheduled based on the priorities.

As computing systems consume more and more energy (Wang and Khan, 2013b; Bilal *et al.*, 2013), a growing number of research works on task scheduling show the feature of energy-awareness. Most of these research works adopt the technique of Dynamic Voltage and Frequency Scaling (DVFS). It has been proven to be a feasible solution for reducing processor energy consumption (Hsu and Feng, 2005). By lowering the processor clock frequency and supply voltage during some time slots (for example, idle or communication phases), energy consumption can be significantly reduced with a slight performance loss. Therefore, the DVFS technique has been applied in high performance computing fields, in large data centers, etc., to reduce the energy consumption while gaining high reliability and availability.

The works of Yao *et al.* (1995) as well as Manzak and Chakrabarti (2003) focus on scheduling independent tasks with DVFS on a single processor. Gruian and Kuchcinski (2001) propose a list-based low energy scheduling algorithm—LEneS. It smartly introduces enhanced task-graphs and energy gain in the list-based scheduling. In the work presented by Martin *et al.* (2002) a hybrid global/local search optimization framework is developed for DVFS with simulated heating. The research work of Zong *et al.* (2011) studies two energy-aware duplication scheduling algorithms for parallel tasks on homogeneous clusters. Lee and Zomaya (2009) propose an energy-conscious scheduling heuristic for parallel tasks on heterogeneous computing systems. In addition (Wang *et al.*, 2010b; 2013c) exploit DVFS to implement a power-aware task clustering algorithm for parallel HPC tasks. The algorithm takes care of the slack time for non-critical jobs, extends their execution time and reduces the energy consumption without increasing the task's execution time as a whole. The algorithm is validated using a simulation approach. Lin and Qiu (2010) apply a thermal-aware strategy based on the RC thermal model (Skadron *et al.*, 2002) to reduce the peak temperature of HPC servers under stochastic workloads. The approach by Zhang and Chatha (2007) combines the two techniques used by Wang *et al.* (2010b) as well as Lin and Qiu (2010) to solve a temperature-aware scheduling problem. For this, the researchers implement an approximation

algorithm based on the lumped RC thermal model and DVFS to study the effect of using the thermal constraints on maximizing the performance of tasks running on some CPU architectures.

2.2. VM scheduling. As the virtualization technology is getting hot, the problem of virtual machine scheduling has also been studied. Kim *et al.* (2008) implement a guest-aware priority-based scheduling scheme, which is specifically designed to support latency-sensitive workloads. The proposed scheme prioritizes the virtual machines to be allocated by using the information about the priorities and status of guest-level tasks in each VM. It preferentially selects the VMs that run latency-sensitive applications to be scheduled, thus reducing the response time to the I/O events of latency-sensitive workloads. The algorithm was integrated in the Xen hypervisor.

Wang *et al.* (2012b) propose a novel VM scheduling algorithm for virtualized heterogonous multicore architectures. It exploits the core performance heterogeneity to optimize the overall system energy efficiency, and uses a metric termed energy-efficiency factor to characterize the power and performance behavior of the applications hosted by VMs on different cores. The VM's energy-efficiency factors are calculated, and, based on the values, virtual machines are mapped to heterogeneous cores with the final goal of maximizing the energy efficiency of the entire system. Similarly to the previous work, this scheduling scheme is also implemented within the Xen hypervisor.

Takouna *et al.* (2011) also addresses the VM scheduling of heterogeneous multicore machines. A scheduling policy is designed to schedule each virtual machine to an appropriate processing core based on the performance sensibility to the CPU clock frequency and the performance dependency on the host. The policy is validated using the Xen hypervisor.

In addition to the research work on VM schedulers of general purposes, the specific problem of VM scheduling on the cloud has also been addressed over the last years.

The work presented by Fang *et al.* (2010) develops a two-layer scheduling model, where the first layer creates the description of resource requirement of a virtual machine while the second one takes care of the VM scheduling on the host. By allocating the VMs to the hosts that closely meet the resource requirement, this approach tries to better use the cloud hardware resources. The approach was evaluated on CloudSim.

Lin *et al.* (2011) propose a dynamic round-robin scheduling scheme for deploying and migrating virtual machines to or across the servers on the cloud. The scheme uses two scheduling rules, whereby the first one avoids allocating additional virtual machines to a retiring physical machine that will be shut down while the second one speeds up the consolidation process. The main goal of this approach is to reduce the number of physical machines used for saving energy.

Hu *et al.* (2010) propose a scheduling strategy for VM scheduling on the cloud with load balancing. The scheduling decision is based on the historical information and the current state of the system. The influence on the system after the deployment of the required VM resources is pre-estimated, and based on this information the best solution is chosen, which will introduce the best load balancing and reduce or avoid dynamic VM migration.

Jang *et al.* (2012) design a task scheduling model with the QoS taken into account. For each task the model creates a set of chromosomes that contain the task allocation and virtual machine information. The model actually handles the problem of scheduling computing tasks to virtual machines, rather than scheduling VMs to physical hosts. The chromosomes are possible task and VM pairs. The model then calculates a fitness value for each chromosome to show how well the virtual machine matching the task requirement, and thereby find the best matches with respect to the user's satisfaction and VM availability.

Knauth and Fetzer (2012) propose a scheduler called OptSched, which schedules virtual machines based on the knowledge about the duration of timed instances to optimize the virtual to physical machine assignment. It co-locates timed instances with similar expiration times. For each host, OptSched calculates the expiration time delta between the instance to be scheduled and the instance with the longest remaining execution time. It then schedules the new instance on the host with the smallest delta. If the new instance shows a longer execution time than the remaining execution time of all running instances, the new instance is scheduled on the host where it extends the host's uptime minimally. The main goal of this scheduler is to reduce the cumulative machine uptime and thereby save the energy consumption.

The work by Beloglazov and Buyya (2010) is also one of the few approaches which take care of energy issues of resource allocation on the cloud. The authors implement a simulation environment based on CloudSim (Calheiros *et al.*, 2011) to evaluate various power-aware scheduling policies for VMs running on a large scale cloud center. Furthermore, they show how VM migration and VM pining techniques can optimize the load-balancing and the total energy consumption of the cloud center.

2.3. Global optimization algorithms. Another topic related to this work is to study global optimization problems. Global optimization is a mathematical problem, widely investigated in recent years (Mesghouni *et al.*, 2004). As a consequence, several intelligent optimization algorithms have been developed, including the tabu search algorithm, the quasi annealing algorithm,

neural network algorithm, the genetic algorithm and the ant colony algorithm.

The tabu search algorithm (Glover, 1989; 1990) is a global optimization algorithm aiming to simulate human intelligence. This algorithm has a better ability in local optimization but it suffers from a premature termination phenomenon. The quasi annealing algorithm (Lundy and Mess, 1986) is a low intelligent algorithm. It simulates annealing objects and thereby forms a modern optimization algorithm. The computation amount of this algorithm is lower than that of the Monte Carlo method. Nevertheless, its global convergence is quite poor. The neural network algorithm (Xing and Xie, 2007) is a kind of intelligent algorithm that constructs an artificial neural network model. The algorithm is fast and simple, but it also falls easily into a local optimal solution. The genetic algorithm (Wang *et al.*, 2007) is an intelligent optimization algorithm based on the biological capacity of the living beings under the use of Darwin's evolution theory of the "survival in the fittest". Genetic algorithms search the solution globally in the entire space and therefore deliver more accurate results, but also require a long computational time as well as suffer from the premature convergence phenomenon. The ant colony algorithm (Wang *et al.*, 2007) is a distributed intelligent simulation algorithm that imitates the social behavior shown by ants by relying on pheromones to communicate. This algorithm can achieve better global optimal solutions, has the feature of strong robustness, and is a parallel algorithm. Its disadvantage lies in the fact that the algorithm involves a lot of parameters and there are no deterministic methods to assign the parameters with a fixed value, and hence can only rely on experiments or experiences. Other deficits of the algorithm include long calculation time and easy occurrence of deadlocks or interruption. The lookout algorithm (Cai *et al.*, 2006) was proposed based on the knowledge of determining the highest point of a mountain by simple observation. The algorithm contains several components including the lookout management mechanism, the strategies for creating the lookout points, and the building and solving of local problems. This algorithm performs a global search but it is easy to produce the leak phenomenon.

Overall, all the algorithms, including tabu search, simulated annealing, the neural network, the genetic algorithm, ant colony and lookout, may create the leak phenomenon in solving global optimization problems or only achieve a local optimization result. Therefore, we propose a kind of vision cognitive algorithm to solve the global optimization problem in VM scheduling on the clouds. This novel approach, similar to the lookout algorithm, is also based on the behavior of human beings locating on the highest/lowest objects by simply observing. However, it ensures that no leak phenomenon occurs while producing the observation points. We used mathematical methods to prove that the sequences generated by the algorithm converge in probability to the global minimum.

2.4. Summary. Overall, task scheduling is a hot topic for parallel and distributed computing and has been addressed not only in the field of cluster computing but also other fields like grid computing (Kołodziej *et al.*, 2012; 2013b; Chen *et al.*, 2013; 2010; Wang *et al.*, 2011a). On the cloud specifically, task scheduling deals actually with the allocation of virtual machines. In the last years the topic of VM scheduling on the cloud has been studied and load-balancing schemes have been also investigated. However, thermal-aware scheduling on the cloud is still not touched upon. This is a special feature of the work presented in this paper. More specifically, the proposed scheduling approach solves a global optimization problem in order to schedule the virtual machine on a host that results in the minimal system energy consumption. The work of Lin *et al.* (2011) touches explicitly upon the energy issues on the cloud, however, the approach is limited to simply shutting down physical hosts without a global view of the entire system. The developed new vision cognitive algorithm speeds up the process of finding the best target host. Therefore, the proposed approach scales well with current and future cloud infrastructures that are getting larger and larger.

3. Scheduling scheme

3.1. Problem description. VM scheduling on a cloud centre handles the task of finding a physical host in response to a VM request of a user. The VM request contains the users requirement on the hardware (potentially also software user's) configuration of the host machine. For example, a user may ask for a physical machine with minimum four computing cores of Intel processors, a three-level cache hierarchy, a memory size larger than 2 GB, and so on. We can use the following mathematical form to describe such a request:

$$R = \{R_1, R_2, R_3, \ldots, R_m\},$$

where R_i is a single requirement attribute in the form of $A > (or <, \geq, \leq)\ value$ (e.g., $memorysize \geq 2GB$).

The scheduler's task is to match this request to a host that also has its individual hardware configuration and system status, like free resources, CPU utilization and temperature. We use the following mathematical form to describe a host machine:

$$H_i = \{P_1, P_2, P_3, \ldots, P_n\},$$

where P shows the hardware parameters and current state of the host. The following host, for example, is equipped

with 64 processing cores and has a free memory of 4 GB, a free disk space of 40 GB, a CPU load of 60%, etc.:

$$H_i = \{64\ cores, 4G\ RAM, 40Gdisk, 60\%CPU, \ldots\}.$$

In a cloud centre there are hundred thousand physical hosts. This work aims to solve a problem of finding one of them that (i) best matches the users requirement; (ii) both its CPU load and temperature do not exceed a pre-defined threshold once the VM runs on it; and (iii) its additional energy demand for running the VM is lower than that of any other host machines.

3.2. Models. To implement a scheduling scheme with the three goals described above, we need to model the CPU temperature and the energy consumption of a computing system.

For measuring the CPU temperature we apply the RC thermal model with single core processors (Skadron *et al.*, 2002). According to Skadron *et al.* (2002), for an integrated circuit at the die level, heat conduction is the dominant mechanism that determines the temperature. There exists a well-known duality between heat transfer and electrical phenomena. Any heat flow can be described as a *current* and the passing of this heat flow through a thermal *resistance* leads to a temperature difference equivalent to a *voltage*. Figure 1 depicts the RC model, where C stands for the thermal capacitance of the processor, R for the thermal resistance, P for the processor's dynamic energy consumption, and T_{amb} for the ambient temperature.

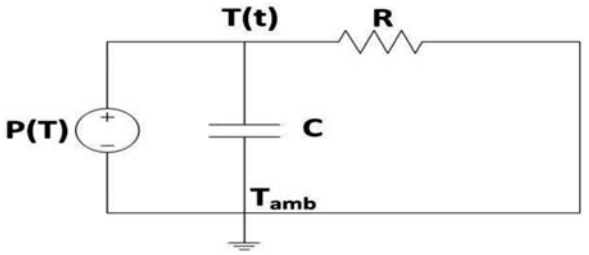

Fig. 1. Lumped RC model of a single core processor.

The relationship between the ambient temperature and the dynamic energy consumption can be modelled as

$$T_{\mathrm{amb}} = RC\frac{\mathrm{d}T}{\mathrm{d}t} + T - RP,$$

and the dynamic energy consumption of a processor is determined by

$$P = kf,$$

where f is the CPU frequency and k is a constant which presents the CPU feature and is determined by the processor vendor and the design of the processor.

Given an initial die temperature of T_{init} at the time zero and assuming that P remains unchanged during the time period $[0, t]$, the temperature of the processor at time t can be computed with (Lin and Qiu, 2010; Zhang and Chatha, 2007)

$$T(t) = \mathrm{PR} + T_{\mathrm{amb}} - (PR + T_{\mathrm{amb}} - T_{\mathrm{init}})e^{-\frac{t}{RC}}.$$

The second model is the energy model used to compute the energy consumption of a server, i.e., a host machine in a cloud centre. Recent studies (Beloglazov and Buyya, 2010) show that the energy consumption by servers can be accurately described by a linear relationship between the energy consumption and the CPU utilization, even when the dynamic voltage and frequency scaling scheme is applied. The reason lies in the limited number of states that can be set to the frequency and voltage of the CPU and the fact that voltage and performance scaling are not applied to other system components, such as the memory and the network interface. Moreover, these studies show that on average an idle server consumes approximately 70% of the energy consumed when it is fully utilized. Therefore, the energy consumption of a server E can be modelled as a function of the CPU utilization u:

$$E(u) = (1-u)0.7E_{\max} + uE_{\max},$$

where $E_{\max}$ is the power consumption of a server in full use. For example, a server of a data centre is in 90% used and 10% the idle. In this case, the energy consumption of the server is $0.9 \times E_{\max}$ in the busy status and $0.1 \times 0.7 \times E_{\max}$ in idle status. The total energy consumption of the server is the sum of the two parts.

3.3. VM scheduling. Based on the models described above, we designed a scheduling scheme for allocating virtual machines on a cloud infrastructure with the following three steps:

- *Filtering by the user requirement:* The requirement parameters in a VM request are compared with the configuration and current status of available host machines of the target cloud. The hosts that do not match users' requirement are filtered out and the other hosts serve as the input for the next step.
- *Threshold with load and temperature:* The second step of the scheduling is to check whether a host machine is overloaded or its processor temperature is above a pre-defined threshold, in the case that the incoming virtual machine runs on it. Those hosts that do not pass the examination are removed from the candidate list.
- *Minimal energy consumption:* In the last step of the scheduling, the energy consumption of the candidate

host machines before and after hosting the virtual machine is calculated. The host that causes the smallest increase in the energy consumption of the system is chosen as the dedicated machine for the new VM instance because it results in the minimal energy consumption of the whole system.

The final step of the scheduling is a global optimization problem, where the additional energy demand of all candidate host machines for running the incoming VM instance has to be compared to find the one with the minimal value. In this case we propose a novel vision cognitive algorithm to enable an accurate result with a short computing time.

The second step of the scheduling deals with two thresholds, one is the CPU utilization and the other is the processor temperature. The problem can be simply solved by the traditional approach of cutting off the values over the thresholds.

The first step of the scheduling handles two parameter sets, one is the VM request of users and the other is the host information. It solves the problem of matching one set to the other to see whether the value in the host set is equal to or larger/smaller than the requirement for each parameter attribute specified in the set of a VM request. Depending on the number of parameters, solving this problem can be complicated and rather time-consuming. Currently, we use a simple approach of comparing two data sets. However, a fast mathematical algorithm is needed for large cloud systems with a huge computing capacity. This is a research focus in the next step of this work.

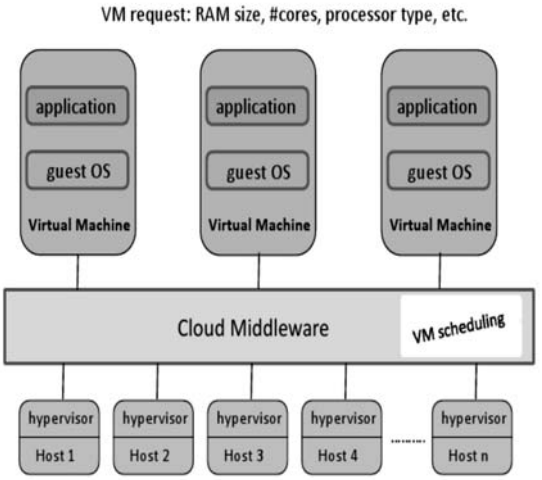

Fig. 2. Interaction of the VM scheduler with the cloud.

The proposed VM scheduler is designed to be integrated in a cloud middleware to replace its original one, usually a round-robin scheme that allocates the VMs to each host equally in a cyclic order. As depicted in Fig. 2, the VM scheduler is a part of a cloud middleware that is responsible for the resource management, VM issues, and so on. Users' requests for creating a VM are delivered to the Cloud middleware, where the scheduler determines a target host based on the proposed strategies described above. The hosts are virtualized using a certain hypervisor. Finally, the Cloud middleware starts a VM on the chosen host with the user-given machine images. We implemented this infrastructure on top of a Cloud simulator. The implementation details will be shown later after the description of the algorithm for global optimization.

4. Vision cognitive algorithm

The proposed novel algorithm solves the following global optimization problems:

$$c^* = \min_{x \in \mathbb{R}^n} f(x). \tag{1}$$

4.1. Algorithm. This subsection first introduces the algorithm. The proof and validation are given in the following two subsections.

Assumption 1. $f : \mathbb{R}^n \to \mathbb{R}$ is continuous and has a lower bound.

Assumption 2. There exists a real number c_0, which makes the level set

$$H_{c_0} = \{x \in \mathbb{R}^n | f(x) \leq c_0\}$$

not empty and bounded.

4.2. Sampling and updating. In the algorithm, the initial density is chosen as $\delta_0(x) = U(D)$, where $D \subset \mathbb{R}^n$ is a hypercube large enough. For other steps we choose the following kernel density function:

$$\delta_{k+1}(x) = \frac{1}{N' h_{N'}^n} \sum_{i=1}^{N'} \ker\Big|\frac{x - \hat{\mathrm{X}}_i^k}{h_{N'}}\Big|, \qquad k = 0, 1, \ldots \tag{2}$$

We use here the relative entropy method (de Boer, 2005) to update the sampling density function $\delta_{k+1}(x)$.

In order to simplify the computation, the kernel function is selected as follows:

$$\ker(x) = \prod_{j=1}^{n} \frac{1}{\pi(1 + x_j^2)}, \tag{3}$$

where x_j presents the j-th element of the set $x = (x_1, x_2, \ldots, x_n)$. Such a kernel function simplifies the task of both the sampling and updating of the density

Algorithm 1. Vision cognition.

Step 1. Input a sufficiently small minimum $\varepsilon > 0$, the number of sample points $N \in \mathbb{Z}^+$, the order $k = 0$, and $c_0 = f(x_0)$.

Step 2. Sampling. Using the order $k = k + 1$ to produce the sample point set $\{X_i^k, i = 1, 2, \ldots, \mathrm{N}\} \sim \delta_k(x)$, where $\delta_k(x)$ is the sampling density function.

Step 3. This step creates the overlooking memory mechanism. First the function $f(X_i^k)$, $X^k = \{X_i^k | f(X_i^k) \leq c_{k-1}, i = 1, 2, \ldots, N\}$ is computed and then $f(\mathrm{X}_i^k)$, $\mathrm{X}_i^k \in \mathrm{X}^k$ is sorted in the increasing order to acquire the order overlook sample set:
$\hat{X^k} = \{\hat{X_i^k}, i = 1, 2, \ldots, N'\}, \qquad N' \leq N.$

Step 4. Calculating the current numerical overlooking level $c_k = f(\hat{X_1^k})$.

Step 5. Calculating ${\sigma_k}^2 = \sum_{x \in \hat{X^k}} (f(x) - c_k)^2$.

Step 6. If ${\sigma_k}^2 < \varepsilon$, the loop terminates and $c^* = c_k$ is the sought approximate total extremum. Otherwise, the sampling density function $\delta_{k+1}(x) = \delta_k(x_1, \hat{X^k})$ is updated and the computation goes back to Step 2.

function. Based on Eqns. (2) and (3), we achieve the sampling kernel density function as

$$\delta_{k+1}(x) = \frac{1}{N' h_{N'}^n} \sum_{i=1}^{N'} \left| \prod_{j=1}^{n} \frac{1}{\pi \left| 1 + \left| \frac{x_j - \hat{X}_{ij}^k}{h_{N'}} \right|^2 \right|} \right|, \quad (4)$$

where $\hat{X_{ij}^k}$ is the j-th element of the i-th sampling point in the k-th priority sampling set ($k = 0, 1, \ldots$).

4.3. Convergence of the algorithm.

Theorem 1. *Assuming that $\{c_k\}$ is the set created by the algorithm, there exists a c such that*

$$p(\lim_{k \to \infty} c_k = c) = 1. \quad (5)$$

Proof. From Algorithm 1 we know that

$$c_k = f(\hat{X_1^k}) \leq f(\hat{X_1^{k-1}}) = c_{k-1}.$$

Therefore, $\{c_k\}$ is a monotonically decreasing sequence. In addition, according to Assumption 1, c_k is bounded. Together, it can be concluded that

$$p(\lim_{k \to \infty} c_k = c) = 1.$$

■

Lemma 1. *If $c_k \xrightarrow{a.s} c$, $\exists \bar{x} \in \mathbb{R}^n$, $\bar{x} \notin \hat{X}^k$ $(k = 1, 2, \ldots)$ and $f(\bar{x}) \leq c$, then $\exists \bar{X}^k \in \hat{X^k}$, $k = 1, 2, \ldots$, so that $\bar{X}^k \to \bar{x} (k \to \infty)$.*

Proof. Assume that there is no sequence $\{\bar{X}^k\}$ with $\bar{x}$ as the limit point. Then there is $\varepsilon > 0$ to make $\forall \hat{X^k} \notin \bigcup(\bar{x}, \varepsilon)$. As $k \to \infty$, selecting the priority sampling set using the designed sampling density function $\delta_k(x)$, we get that $\exists \varepsilon > 0$, which yields $\forall \hat{X^k} \in \bigcup(\bar{x}, \varepsilon)$. This contradicts the assumption. ■

Theorem 2. *A necessary and sufficient condition for $c = \min_{x \in \mathbb{R}^n} f(x)$ is that when the algorithm creates $c_k \xrightarrow{a.s} c$, we have $\forall x \in H_c, f(x) = c$.*

Proof. We prove first the necessity. Given $c = \min_{x \in \mathbb{R}^n} f(x), H_c = \{x | f(x) \leq c\}$. Then for $\forall x \in H_c$ there must be $f(x) = c$.

Now we prove the sufficiency. Given that for $\forall x \in H_c$ and $c_k \xrightarrow{a.s} c$ there exists $f(x) = c$, then for $\forall x \in \mathbb{R}^n$ there holds $f(x) \geq c$.

Finally, we prove the theorem by contradiction. Assume the following:

- There exists k so that $x_0 \in \hat{X^k} \subset \mathbb{R}^n$ and $f(x_0) < c$.

  According to the algorithm, $c_k = f(\hat{X_1^k})$. Therefore, $f(\hat{X_1^k}) \leq f(x_0)$ and $c_k \leq f(x_0)$. Moreover, we have $c_k \xrightarrow{a.s} c$. According to Theorem 1, $c \leq c_k \leq f(x_0)$, which contradicts the assumption.

- $\exists \bar{x} \in \mathbb{R}^n$ and $\bar{x} \notin \hat{X^k}$, $f(\bar{x}) < c$.

According to Lemma 1, $\exists \bar{X}^k \in \hat{X^k}, k = 1, 2 \ldots$, which makes $\bar{X}^k \to \bar{x}$ $(k \to \infty)$. Furthermore, according to the algorithm, $c_k = f(\hat{X_1^k})$ and $f(\hat{X_1^k}) \leq f(\bar{x})$. Therefore, $c_k \leq f(\bar{x})$. In addition, $c_k \xrightarrow{a.s} c$. Based on Theorem 1, it can be concluded that $c \leq c_k \leq f(\bar{x})$. This again conflicts with the assumption. Combining all the proofs above, we conclude that $\forall x \in \mathbb{R}^n$ there holds $f(x) \geq c$. ■

Theorem 3. *Given $c \leq c_k \leq f(\bar{x})$, c is the global optimum and H_c is the global minimal point set of Eqn. (1).*

Proof. As $k \to \infty$, ${\sigma_k}^2 = \sum_{x \in \hat{\mathrm{X}^k}} (f(x) - c_k)^2 \to 0$, we have $|f(x) - c_k| \to 0$. Furthermore, $c_k \xrightarrow{a.s} c$. Therefore, $\forall x \in H_c$, if $x \in \hat{X^k}$ then $f(x) = c$ when $k \to \infty$. If $x \notin \hat{X^k}$ and $f(\bar{x}) < c$, from Lemma 1 we deduce that $\exists \bar{X}^k \in \hat{X^k}, k = 1, 2, \ldots$, which results in $\bar{X}^k \to \bar{x}$ $(k \to \infty)$ and further $f(x) = c$. According to Theorem 2, $c = \min_{x \in \mathbb{R}^n} f(x)$, and correspondingly H_c is the optimal point set.

Since ${\sigma_k}^2 = \sum_{x \in \hat{X^k}} (f(x) - c_k)^2 = 0$, we get $|f(x) - c_k| = 0$. Moreover, since $c_k \xrightarrow{a.s} c$ it is true that for $\forall x \in H_c$ there holds $f(x) = c$. Based on Theorem 2 it can be concluded that $c = \min_{x \in \mathbb{R}^n} f(x)$, and correspondingly H_c is the optimal point set. ■

Validation. We validated the algorithm using MATLAB with several sample problems. The first example is the

Rosenbrock function

$$F_2(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2, \quad x_i \in [-10, 10], \quad i = 1, 2$$

with $\min F_2(x_1, x_2) = 0$.

Based on the proposed algorithm, we achieved the optimum point of $(1, 1)^T, \min F_2(x_1, x_2) = 0$ by using the point number $N = 2000$ and the convergence number $k = 4$.

The second example is the Easom (ES) function:

$$ES(x_1, x_2) = -\cos x_1 \cos x_2 \exp(-((x_1 - \pi)^2 + (x_2 - \pi)^2))$$

with the search space $-100 \leq x_i \leq 100$, $i = 1, 2$, and the global optimum $x^* = (\pi, \pi), ES(x^*) = -1$. Applying the proposed algorithm we acquired a result of the optimum $(3.1423, 3.1489)^T$ and the optimal value -0.999989, when using $N = 4000$ and $k = 3$.

The third example solves the function sphere model:

$$F_4(x_1, x_2, \ldots, x_n) = x_1^2 + x_2^2 + \cdots + x_n^2, \quad x_i \in [-100, 100], \quad i = 1, \ldots, m,$$

with $n = 3$ and the optimum at $x_i = 0\,(i = 1, 2, 3)$, $\min F_4(x_1, x_2, x_3) = 0$.

Using this algorithm, we achieved a solution at $(0.0020, 0.0019, 0.0021)^T, \min F_4(x_1, x_2, \ldots, x_8) = 0.00001202$ by $N = 3000$ and $k = 9$.

For the VM scheduling on the cloud, the proposed algorithm actually deals with the function $F(x_i) = E_{\text{cpu}}(x_i)_{\text{after}} - E_{\text{cpu}}(x_i)_{\text{before}}$ with a search space of $0 < x < N$, where N is the number of host machines that have passed through the matching process of the first two scheduling steps. The function calculates the additional energy consumption of a host in case a new VM runs on it. The optimum for this problem is $\min F(x) = 0$ because we are looking for the host that introduces the minimal energy increase after the new VM is scheduled. The energy consumption of a host is calculated with our energy model on the runtime when a cloud is working.

To make it clear how the algorithm works for VM scheduling, the functionality of the algorithm for finding the host with a minimal energy increase is demonstrated in Fig. 3. As shown there, the algorithm first uses *overlook point 1* to cut off the hosts, whose additional energy consumption is higher than the value at this point. It then applies *overlook point 2* to cut more hosts off. This process goes on further till there is only one host left.

It can be seen that the performance of the algorithm is significantly influenced by the first overlook point because it directly determines the numbers of cuttings up till convergence. We start simply with a constant for the current implementation. The plan is a more intelligent approach that dynamically computes the initial overlook point based on the history information.

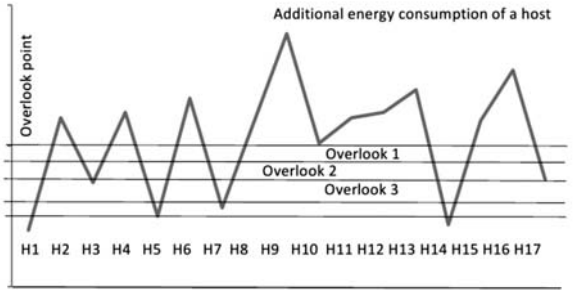

Fig. 3. Mapping the vision cognitive algorithm to the host selection.

5. Implementation

We implemented the proposed algorithm as well as the scheduling scheme, and validated the approach using a simulator for computing Clouds, the CloudSim toolkit (Calheiros *et al.*, 2011). It models large scale data centres with an internal broker, sensors, a physical host, as well as virtual machines. CloudSim also provides the functionality to configure datacenter resources. This allows us to model cloud providers with different system scale and hardware resources.

Figure 4 shows the core architecture of CloudSim. It can be seen that this simulator is well structured with a clear layer definition. It is easy for us to extend CloudSim with additional components both at the user level for specifying users VM requests and at the simulation level for implementing the scheduling scheme. In addition, CloudSim enables also the simulation of dynamic load, which allows us to study the influence of different kinds of workloads on the energy consumption of the host machines.

The three steps of the scheduling scheme are implemented in different classes. While the vision cognitive algorithm for the last step is described above, the algorithms for the first step is demonstrated with the following pseudo-code:

```
Input: VM request (processor type P, RAM size R,
          #cores N, disk size D, OS type OS, ....)
Input: AllHostList
Output: MatchHostList
  For each host in AllHostList Do
    If getProcessorType (host) == P and getRAM (host))
      >= R and getNumbersCores (host) >= N and
      getDisk (host) >= D and ...
    Then add host to MatchHostList
    Endif
  Endfor
Return MatchHostList
```

The task of the first scheduling step is to find those physical hosts that meet the hardware and software requirements specified in a VM request. As shown in the pseudo-code, the algorithm has both the VM request and the full list of cloud centre available hosts as inputs. It goes through the host list to check each parameter of the request and fills thereby the matching hosts in a new

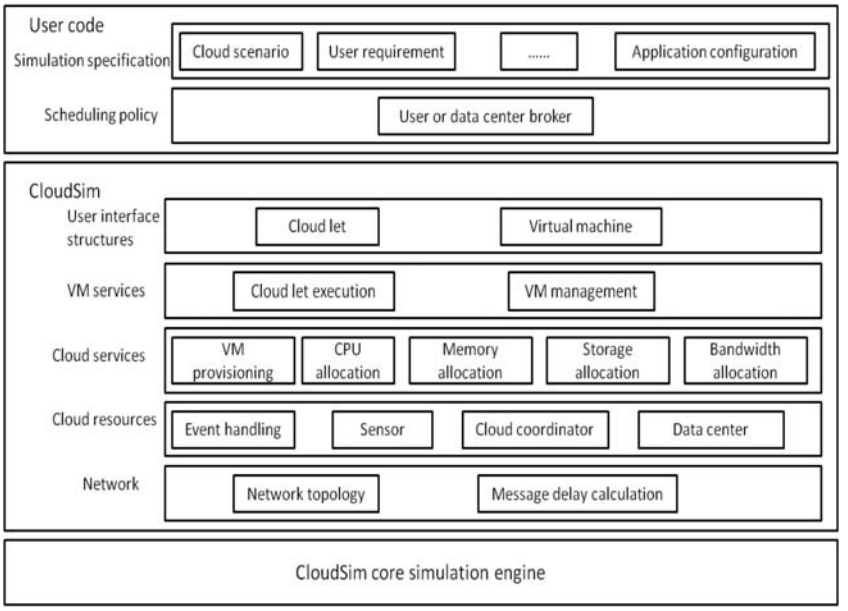

Fig. 4. Software architecture of CloudSim.

list of hosts. This list serves as the input for the second scheduling step.

As demonstrated in the following pseudo-code, in the second step the scheduler goes through all the matching hosts detected by the last scheduling step to find those hosts whose temperature and CPU usage exceed the specified thresholds. The detected hosts are then removed from the list of matching hosts to form a new list of hosts that are candidates for running the requested virtual machine. We also output the list of overheated hosts and the list of overloaded hosts for the reason of evaluation.

```
Input: MatchHostList, ThresholdTemperature,
       ThresholdUtilization
Output: CandidateHostlist, overThresholdTempHosts,
        overUtilizedHosts
  For each host in MatchHostList Do
    If isHostOverThresholdTemperature(host)
    Then overThresholdTempHosts <- add host
    Else
      If isHostOverThresholdUtilization(host)
      Then overUtilizedHosts <- add host
      Endif
    Endif
  Endfor
  CandidateHostList <- MatchHostList -
    overThresholdTempHosts - overUtilizedHosts
Return CandidateHostList, overThresholdTempHosts,
       overUtilizedHosts
```

The algorithm uses two thresholds, one for limiting the number of workloads on a host and the other for preventing the processors from a high temperature. Both the thresholds aim at reducing the cloud centre energy consumption while ensuring the quality of cloud services. The former concerns the performance and the latter enhances the service availability by reducing the potential hardware defect. Both the thresholds may be determined based on the hardware properties of the processors. For example, the threshold for CPU utilization can be set to 100% for most of the vendor processors, and the threshold for chip temperature varies from one type of processor to another.

When considering the energy consumption, however, the selection of both the thresholds is not that simple with the only referring to the hardware feature of the processors. We performed several experiments to investigate on this issue with the simulation approach. The details of the simulation setups will be given in the next section, together with more results about the validation of the proposed scheduling concept. Figures 5 and 6 show the experimental results with the thresholds.

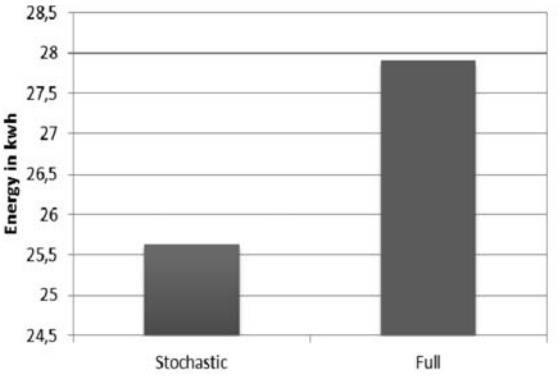

Fig. 5. System energy consumption with different workloads (y-axis: energy consumption, x-axis: workload).

Figure 5 depicts the energy consumption of the

entire system (all host machines) by running two different types of workloads. The left bar in the figure presents the result with a stochastic workload, where jobs are coming with an interval calculated using an exponential distribution and the job size is generated using a variety of probabilistic distributions. The right bar is the result with a full CPU usage. It can be seen that the energy consumption for a workload with a full CPU usage is larger than the same metric measured with the stochastic workload. This means that a full utilization of the CPU results in a higher energy consumption, and for energy optimization of a cloud centre the CPUs will not be 100% loaded. The experimental results of a research work of Beloglazov and Buyya (2012) demonstrated that the energy consumption rises considerably from a CPU utilization of 90%. Therefore, we chose a value of 0.9 as the threshold of the CPU utilization for our validation tests.

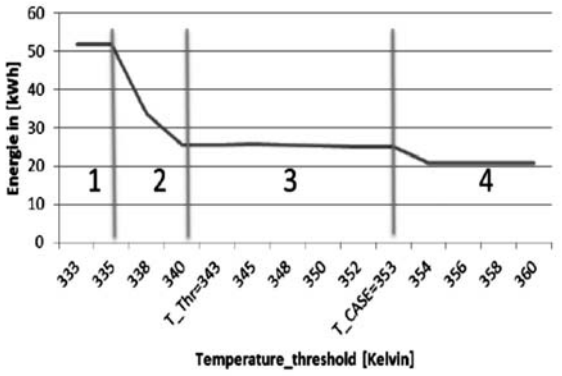

Fig. 6. Influence of the temperature threshold on the energy consumption.

Similarly to the CPU utilization, different temperature thresholds result also in diversified energy consumption. We used a set of temperature thresholds to study this issue. Figure 6 shows the result. As depicted there, the temperature thresholds applied range from 333 Kelvin (unit of temperature) to 360 Kelvin. The value presented in the y-axis is the total energy consumption of a cloud centre in a single day.

Observing the curve in the figure, it can be seen that the energy consumption with different temperature thresholds can be divided into four areas of different patterns. In the first area, i.e., the temperature threshold between 333 and 335, the energy consumption maintains constant with a value of 52 kWh. In the second region, i.e., $335 < \mathit{temperature\ threshold} < 340$, the energy consumption of the system goes down as the threshold is getting larger. This behavior stops at threshold 340, where a new phase starts, again with constant energy consumption. After this phase, a slight reduction in the energy consumption can be seen, but generally the temperature threshold does not change the energy behavior in phase 4. The best case for our experiments is the point with 343 Kelvin.

6. Validation results

6.1. Experimental setup. As described above, we applied a simulation approach to validate the developed scheduling strategies. The reason for using simulation, rather than a real cloud platform lies in the fact that we need the status information of physical hosts on a cloud centre. Currently, it is not possible to gain this information from a commercial cloud, which prohibits us from validating the full functionality of the implemented scheduler and from studying the behavior of the designed scheduling as well as global optimization algorithms. In contrast, CloudSim implements a view of infinite computing resources, and it is easy to extend this simulator for both monitoring and additional functionalities.

Table 1. Configuration of virtual machines.

VM Type	1	2	3	4
VM_MIPS	500	1000	2000	2500
VM_Cores	1	1	1	1
VM_RAM [MB]	613	1740	1740	870
VM_BW [Mbit/s]	100	100	100	100
VM_Size [GB]	2.5	2.5	2.5	2.5

For the experiments, we modelled the scheduling task of a whole day with a random generation of VM requests. A VM request contains four requirement attributes, i.e., the number of CPU cores, the CPU frequency, the RAM size and the bandwidth. The properties of the modelled VM types are described in Table 1. As shown there, we use different VMs with various values in MIPS and the RAM size to model real scenarios.

The thermal constants used in the simulation for the lumped RC thermal model are listed in Table 2. These are typical values of a single core CPU obtained from (LAVA Lab, 2013).

We configured CloudSim for a cloud center with 50 different hosts. A half of the hosts are modelled as HP ProLiant G4 servers and the other half as HP ProLiant G5 servers. The characteristics of the servers are given in Table 3. The frequency of each core on the HP ProLiant G4 server is 1860 MIPS and for the HP ProLiant G5 server the value is 2660 MIPS. Each server is modelled with a connection of 1 GB/s bandwidth. The CPU utilization behavior of the servers is acquired from SpecPower08 (SPEC, 2013). The threshold for the CPU utilization is

Table 2. Thermal constants for the lumped RC thermal model.

Thermal Parameter	Value	Unit
Initial CPU Temperature (T_{init})	318	Kelvin
Ambient Temperature (T_{amb})	308	Kelvin
Case Temperature (T_{case})	353	Kelvin
Thermal Capacity C	340	Joule/Kelvin
Thermal Resistance R	0.34	Kelvin/Watt

Table 3. Simulated physical machines in a cloud centre.

Server Host Type	HP_Proliant_G4	HP_Proliant_G5
Host_MIPS	1860	2660
Host_Cores	1	1
Host_RAM [MB]	2048	4096
Host_BW [Gbit/s]	1	1
Host_Storage [TB]	1	1

set to 0.9 while the temperature threshold is chosen as 343 Kelvin, based on the test results shown in the previous section.

In order to observe the efficiency of the implemented scheduler, we also implemented three other scheduling schemes for a comparative study. The first one is called Non_power_aware, which schedules the virtual machines without considering the CPU usage. It reflects the energy consumption in a cloud center with full CPU utilization. The second scheme is DVFS. It schedules tasks on the basis of the CPU voltage and frequency. It relies on the information from the CPU performance and power model to set the priorities for the VM placement. The energy consumption is calculated as a function of the CPU usage, and is regulated automatically and dynamically based on DVFS. The last scheme is Power_aware_ThrMu (Beloglazov and Buyya, 2010). This scheduling algorithm focuses on minimizing the CPU usage by setting up hosts in the idle mode. It migrates the running VMs of a host with CPU usage over a threshold to other hosts. We use a utilization threshold of 0.9 for this scheme, the same as our scheduler.

6.2. Experimental results. The main goal of the proposed scheduling scheme is to reduce the energy consumption by scheduling a virtual machine to the host that causes a minimal increase in energy consumption. Therefore, our first experiment was done to see how the hosts vary from one another in the additional energy requirement after hosting a new virtual machine.

Figure 7 depicts the experimental results. We choose hosts 1 to 10 as examples. The x-axis of the diagram in the figure shows the hosts, while the y-axis is the additional energy requirement of an individual host in case a new VM is run on it. The data were measured at a certain time point. From the figure it can be clearly seen that the hosts show individual additional energy consumption with the largest difference of 21%. This number will get larger for a cloud centre with more different types of hosts. As mentioned, for this experiment we only configured two types of processors.

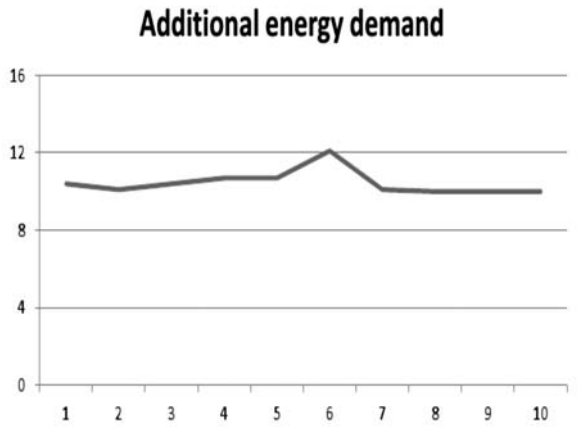

Fig. 7. Additional energy consumption of different hosts in case of launching a new VM instance (x-axis: hosts, y-axis: energy increase in KW).

The next experiment studies the feasibility of the proposed scheduling scheme by comparing the energy consumption in the cases of applying different scheduling schemes, i.e., Non_power_aware, DVFS, Power_aware_ThrMu and our thermal-aware scheme (Thas).

Figure 8 depicts the result of the experiment, where the energy consumption was measured during a single simulation run with all four algorithms. Comparing the proposed scheduler with the other scheduling algorithms, it can be seen that our scheduler achieves the lowest energy demand with a value of 25.64 KWh per day, while the energy consumption with other schemes is 150.68 KWh with Non_power_aware, 52.98 KWh with DVFS and 28.9 KWh with Power_aware_ThrMu. It can be concluded that combining the power-aware with the thermal-aware scheduling strategies provides the best results for energy consumption.

To have a deeper insight in the different scheduling schemes, we measured the energy consumption at different times of day. Figure 9 presents the experimental results. The x-axis of the diagram in the figure is the time points where the energy consumption calculated. The y-axis presents the total energy consumption of the simulated cloud centre.

The data were first gathered after 2 hours of a simulated run, and then at a time interval. The four curves in the figure present the results with the four scheduling algorithms. The curves go up with the duration of the simulation because a cloud centre requires more energy

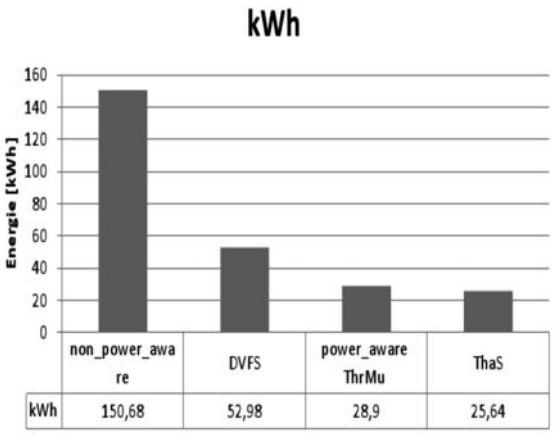

Fig. 8. Data center one-day energy consumption with several scheduling schemes (x-axis: scheduling schemes, y-axis: energy consumption in KWh with the numbers shown in the lower table).

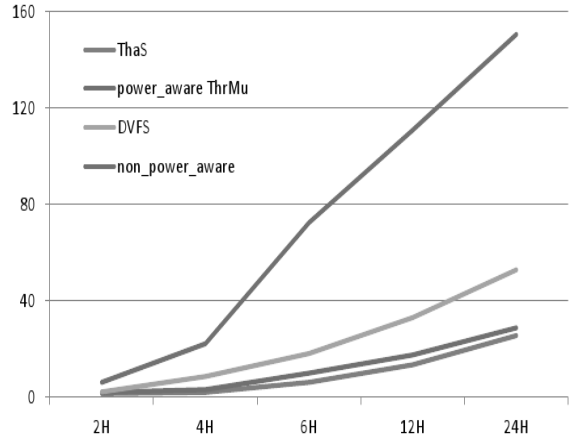

Fig. 9. Energy consumption at different time points (x-axis: time points, y-axis: total energy consumption in KWh).

for its operation with time. It is interesting to see that the behavior difference of the algorithms gets clearer as the simulation is running. From the figure it can be seen that after a two-hour run the energy consumption for the four scheduling schemes varies only slightly. However, the difference gets larger and larger. After 24 hours, for example, the total cloud centre energy consumption for the proposed scheduling scheme is only 17% of that caused by the Non_power_aware scheme. This means that the cloud centre will significantly save its energy for a long-term operation by using the proposed approach.

However, the gain in energy is achieved with some violation to the Service Level Agreement (SLA) that defines the quality of the services. We measured an SLA violation of 14%, where the VM request cannot be processed within the promised response time. This indicates that the proposed algorithms have to be optimized towards less computation time.

7. Conclusions

The paper describes a novel scheduling scheme for allocating virtual machines on the physical hosts of a cloud centre. The main goal of the scheduling is to achieve minimal system energy consumption while ensuring the performance and quality of cloud services. For this purpose, we propose to use an optimization algorithm to compute the best location for a virtual machine with fast convergence so that most of the VM requests can be processed in the given response time.

In the next step of this research work, the energy model will be improved for covering heterogeneous architectures. Additionally, the vision cognitive algorithm has to be improved to use the history information to determine the first overlook point. More importantly, we will study another mathematical algorithm for mapping the VM request set to the host configuration set. It is expected that with these optimizations the proposed scheduling algorithm will scale well for larger systems without a violation to the SLA of a cloud centre.

Acknowledgment

This work was supported (in part) by the German Research Foundation (DFG) through the Priority Programme 1648: *Software for Exascale Computing (SPPEXA)*.

References

Adaptive Computing Inc. (2013). TORQUE Resource Manager, `http://www.adaptivecomputing.com/products/open-source/torque/`.

Altair Engineering Inc. (2013). PBS Works—Enabling On-Demand Computing, `http://www.pbsworks.com/`.

Amazon (2013a). Amazon Elastic Compute Cloud, `http://aws.amazon.com/ec2/`.

Amazon (2013b). Simple Storage Service, `http://aws.amazon.com/s3/`.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. and Zaharia, M. (2009). Above the clouds: A Berkeley view of cloud computing, *Technical report*, University of California at Berkeley, Berkeley, CA.

Barham, P., Dragovic, B. and Fraser, K. (2003). Xen and the art of virtualization, *Proceedings of the 19th ACM Symposium on Operating Systems Principles, Bolton Landing, NY, USA*, pp. 164–144.

Beloglazov, A. and Buyya, R. (2010). Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers, *Proceedings of the 8th*

International Workshop on Middleware for Grids, Clouds and e-Science, Bangalore, India, pp. 4:1–6.

Beloglazov, A. and Buyya, R. (2012). Optimal online deterministic algorithms and adaptive heuristic for energy and performance efficient dynamic consolidation of virtual machines in cloud datacenters, *Concurrency and Computation: Practice and Experience* **24**(3): 1397–1420.

Bilal, K., Khan, S.U., Madani, S.A., Hayat, K., Khan, M.I., Min-Allah, N., Kołodziej, J., Wang, L., Zeadally, S. and Chen, D. (2013). A survey on green communications using adaptive link rate, *Cluster Computing* **16**(3): 575–589.

Cai, Y., Qian, J. and Sun, Y. (2006). Outlook algorithm for global optimization, *Journal of Guangdong University of Technology* **23**(2): 1–10.

Calheiros, R.N., Ranjan, R., Beloglazov, A., e Rose, C.A.F.D. and Buyya, R. (2011). CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, *Software: Practice and Experience* **41**(1): 23–50.

Chen, D., Li, D., Xiong, M., Bao, H. and Li, X. (2010). GPGPU-aided ensemble empirical mode decomposition for EEG analysis during anaesthesia, *IEEE Transactions on Information Technology in BioMedicine* **14**(6): 1417–1427.

Chen, D., Wang, L., Wu, X., Chen, J., Khan, S., Kołodziej, J., Tian, M., Huang, F. and Liu, W. (2013). Hybrid modelling and simulation of huge crowd over a hierarchical grid architecture, *Future Generation Computer Systems* **29**(5): 1309–1317.

de Boer, P. (2005). A tutorial on the cross-entropy method, *Annals of Operations Research* **134**(2): 19–67.

Fang, Y., Wang, F. and Ge, J. (2010). A task scheduling algorithm based on load balancing in cloud computing, *Proceedings of the 2010 International Conference on Web Information Systems and Mining, Sanya, China*, pp. 271–277.

FlexiScale Ltd. (2013). FlexiScale: Utility Computing on Demand, `http://www.flexiscale.com/`.

Gentzsch, W. (2001). Sun grid engine: Towards creating a compute power grid, *Proceedings of the 1st International Symposium on Cluster Computing and the Grid, Washington, DC, USA*, pp. 35–36.

Glover, F. (1989). Tabu search: Part I, *ORSA Journal on Computing* **21**(1): 190–206.

Glover, F. (1990). Tabu search: Part II, *ORSA Journal on Computing* **21**(2): 4–32.

González-Vélez, H. and Kontagora, M. (2011). Performance evaluation of MapReduce using full virtualisation on a departmental cloud, *International Journal of Applied Mathematics and Computer Science* **21**(2): 275–284, DOI: 10.2478/v10006-011-0020-3.

Gruian, F. and Kuchcinski, K. (2001). LEneS: Task scheduling for low-energy systems using variable supply voltage processors, *Proceedings of the 2001 Asia and South Pacific Design Automation Conference, Yokohama, Japan*, pp. 449–455.

Hsu, C. and Feng, W. (2005). A feasibility analysis of power awareness in commodity-based high-performance clusters, *Proceedings of Cluster Computing, Burlington, VT, USA*, pp. 1–10.

Hu, J., Gu, J., Sun, G. and Zhao, T. (2010). A scheduling strategy on load balancing of virtual machine resources in cloud computing environment, *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Programming, Dalian, China*, pp. 89–96.

IBM (2013). IBM SmartCloud, `http://www.ibm.com/cloud-computing/`.

Jang, S., Kim, T., Kim, J. and Lee, J. (2012). The study of genetic algorithm-based task scheduling for cloud computing, *International Journal of Control and Automation* **5**(4): 157–162.

Kahn, S., Bilal, K., Zhang, L., Li, H., Hayat, K., Madani, S., Min-Allah, N., Wang, L., Chen, D., Iqbal, M., Xu, C. and Zomaya, A. (2013). Quantitative comparisons of the state of the art data center architectures, *Concurrency and Computation: Practice & Experience* **25**(12): 1771–1783, DOI:10.1002/cpe.2963.

Keahey, K. and Freeman, T. (2008). Science clouds: Early experiences in cloud computing for scientific applications, *Proceedings of the 1st Workshop on Cloud Computing and Its Applications, Chicago, IL, USA*.

Kim, D., Kim, H., Jeon, M., Seo, E. and Lee, J. (2008). Guest-aware priority-based virtual machine scheduling for highly consolidated server, *Proceedings of the 14th International Conference on Parallel and Distributed Computing (Euro-Par 2008), Las Palmas de Gran Canaria, Spain*, pp. 285–294.

Kim, S. (1988). A general approach to multiprocessor scheduling, *Technical report*, University of Texas at Austin, Austin, TX.

Knauth, T. and Fetzer, C. (2012). Energy-aware scheduling for infrastructure clouds, *Proceedings of the IEEE International Conference on Cloud Computing Technology and Science, Taipei, Taiwan*, pp. 58–65.

Kołodziej, J., Khan, S., Wang, L., Kisiel-Dorohinicki, M. and Madani, S. (2012). Security, energy, and performance-aware resource allocation mechanisms for computational grids, *Future Generation Computer Systems* **31**: 77–92, DOI: 10.1016/j.future.2012.09.009.

Kołodziej, J., Khan, S., Wang, L., Byrski, A., Nasro, M. and Madani, S. (2013a). Hierarchical genetic-based grid scheduling with energy optimization, *Cluster Computing* **16**(3): 591–609, DOI: 10.1007/s10586-012-0226-7.

Kołodziej, J., Khan, S., Wang, L. and Zomaya, A. (2013b). Energy efficient genetic-based schedulers in computational grids, *Concurrency and Computation: Practice & Experience*, DOI:10.1002/cpe.2839.

Kołodziej, J. and Xhafa, F. (2011). Modern approaches to modeling user requirements on resource and task allocation in hierarchical computational grids, *International Journal of Applied Mathematics and Computer Science* **21**(2): 243–257, DOI: 10.2478/v10006-011-0018-x.

KVM (2013). Kernel Based Virtual Machine, http://www.linux-kvm.org/.

LAVA Lab (2013). Hotspot, http://lava.cs.virginia.edu/HotSpot/.

Lee, Y. and Zomaya, A. (2009). Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling, *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, Washington, DC, USA*, pp. 92–99.

Li, R. and Huang, H. (2007). List scheduling for jobs with arbitrary release times and similar lengths, *Journal of Scheduling* **10**(6): 365–373.

Lin, C., Liu, P. and Wu, J. (2011). Energy-aware virtual machine dynamic provision and scheduling for cloud computing, *Proceedings of the IEEE International Conference on Cloud Computing, Washington, DC, USA*, pp. 736–737.

Lin, S. and Qiu, M. (2010). Thermal-aware scheduling for peak temperature reduction with stochastic workloads, *Proceedings of IEEE/ACM RTAS WIP, Chicago, IL, USA*, pp. 53–56.

Lundy, M. and Mess, A. (1986). Convergence of an annealing algorithm, *Journal on Mathematical Programming* **34**(1): 111–124.

Manzak, A. and Chakrabarti, C. (2003). Variable voltage task scheduling algorithms for minimizing energy/power, *IEEE Transactions on Very Large Scale Integration Systems* **11**(2): 270–276.

Martin, S., Flautner, K., Mudge, T. and Blaauw, D. (2002). Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads, *Proceedings of the 2002 IEEE/ACM International Conference on Computer-aided Design, San Jose, CA, USA*, pp. 721–725.

Mell, P. and Grance, T. (2013). The NIST Definition of Cloud Computing, http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf.

Mesghouni, K., Hammadi, S. and Borne, P. (2004). Evolutionary algorithms for job-shop scheduling, *International Journal of Applied Mathematics and Computer Science* **14**(1): 91–103.

Min-Allah, N., Khan, S.U., Ghani, N., Li, J., Wang, L. and Bouvry, P. (2012). A comparative study of rate monotonic schedulability tests, *The Journal of Supercomputing* **59**(3): 1419–1430.

Mtibaa, A., Ouni, B. and Abid, M. (2007). An efficient list scheduling algorithm for time placement problem, *Journal of Computers and Electrical Engineering* **33**(4): 285–298.

Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L. and Zagorodnov, D. (2008). The Eucalyptus open-source cloud-computing system, *Proceedings of Cloud Computing and Its Applications*, http://eucalyptus.cs.ucsb.edu/wiki/Presentations.

Openstack (2013). OpenStack Cloud Software, http://openstack.org/.

ORACLE (2013). Oracle Grid Engine, http://www.oracle.com/us/products/tools/oracle-grid-engine-075549.html.

Rosenblum, M. and Garfinkel, T. (2005). Virtual machine monitors: Current technology and future trends, *Computer* **38**(5): 39–47.

Skadron, K., Abdelzaher, T. and Stan, M.R. (2002). Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management, *Proceedings of the 8th International Symposium on High-Performance Computer Architecture, HPCA '02, Washington, DC, USA*, pp. 17–28.

Sotomayor, B., Montero, R., Llorente, I. and Foster, I. (2008). Capacity leasing in cloud systems using the OpenNebula engine, *The First Workshop on Cloud Computing and Its Applications, Chicago, IL, USA*.

SPEC (2013). SpecPower08, http://www.spec.org.

Staples, G. (2006). TORQUE resource manager, *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, Tampa, FL, USA*.

Takouna, I., Dawoud, W. and Meinel, C. (2011). Efficient virtual machine scheduling-policy for virtualized heterogeneous multicore systems, *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA2011), Las Vegas, NV, USA*.

Tannenbaum, T., Wright, D., Miller, K. and Livny, M. (2002). *Beowulf Cluster Computing with Linux*, MIT Press, Cambridge, MA, pp. 307–350.

Valentini, G., Lassonde, W., Khan, S., Min-Allah, N., Madani, S., Li, J., Zhang, L., Wang, L., Ghani, N., Kołodziej, J., Li, H., Zomaya, A., Xu, C., Balaji, P., Vishnu, A., Pinel, F., Pecero, J., Kliazovich, D. and Bouvry, P. (2013). An overview of energy efficiency techniques in cluster computing systems, *Cluster Computing* **16**(1): 3–15.

VMware Inc. (2013). VMware, http://www.vmware.com.

Wang, D., Wang, J., Wang, H., Zhang, R. and Guo, Z. (2007). *Intelligent Optimization Approaches*, High Education Publish House, Beijing.

Wang, L., Laszewski, G., Younge, A., He, X., Kunze, M., Tao, J. and Fu, C. (2010a). Cloud computing: A perspective study, *New Generation Computing* **28**(2): 137–146.

Wang, L., Tao, J., von Laszewski, G. and Chen, D. (2010b). Power Aware scheduling for parallel tasks via task clustering, *Proceedings of the IEEE 16th International Conference on Parallel and Distributed Systems (ICPADS), Shanghai, China*, pp. 629–634.

Wang, L., Chen, D. and Huang, F. (2011a). Virtual workflow system for distributed collaborative scientific applications on Grids, *Computers & Electrical Engineering* **37**(3): 300–310.

Wang, L., von Laszewski, G., Huang, F., Dayal, J., Frulani, T. and Fox, G. (2011b). Task scheduling with ANN-based temperature prediction in a data center: A simulation-based study, *Engineering with Computers* **27**(4): 381–391.

Wang, L., Khan, S. and Dayal, J. (2012a). Thermal aware workload placement with task-temperature profiles in a data center, *The Journal of Supercomputing* **61**(3): 780–803.

Wang, Y., Wang, X. and Chen, Y. (2012b). Energy-efficient virtual machine scheduling in performance-asymmetric multi-core architectures, *Proceedings of the 8th International Conference on Network and Service Management and 2012 Workshop on Systems Virtualization Management, Las Vegas, NV, USA*, pp. 288–294.

Wang, L., Chen, D., Hu, Y., Ma, Y. and Wang, J. (2013a). Towards enabling cyberinfrastructure as a service in clouds, *Computers & Electrical Engineering* **39**(1): 3–14.

Wang, L. and Khan, S. (2013b). Review of performance metrics for green data centers: A taxonomy study, *The Journal of Supercomputing* **63**(3): 639–656.

Wang, L., Khan, S., Chen, D., Kołodziej, J., Ranjan, R., Xu, C. and Zomaya, A. (2013c). Energy-aware parallel task scheduling in a cluster, *Future Generation Computer Systems* **29**(7): 1661–1670.

Wu, M. and Gajski, D. (1990). Hypertool: A programming aid for message-passing systems, *IEEE Transactions on Parallel and Distributed Systems* **1**(3): 330–343.

Xing, W. and Xie, J. (2007). *Modern Optimization Algorithms*, Qinghua University, Beijing.

Yao, F., Demers, A. and Shenker, S. (1995). A scheduling model for reduced CPU energy, *Proceedings of the 36th Annual Symposium on Foundations of Computer Science, Milwaukee, WI, USA*, pp. 374–382.

Zhang, S. and Chatha, K.S. (2007). Approximation algorithm for the temperature-aware scheduling problem, *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, San Jose, CA, USA*, pp. 281–288.

Zhang, W., Wang, L., Song, W., Ma, Y., Liu, D., Liu, P. and Chen, D. (2013). Towards building a multi-datacenter infrastructure for massive remote sensing image processing, *Concurrency and Computation: Practice & Experience* **25**(12): 1798–1812.

Zong, Z., Manzanares, A., Ruan, X. and Qin, X. (2011). EAD and PEBD: Two energy-aware duplication scheduling algorithms for parallel tasks on homogeneous clusters, *IEEE Transactions on Computers* **60**(3): 360–374.

Jiaqi Zhao obtained her M.Sc. degree at the Northeast Normal University of China in 2006. Currently she works as a lecturer at the Changchun University of Technology. Her major research areas are data analysis algorithms and data-intensive computing. She has participated in the development of several research projects. Her research results have been published in several international conferences and journals.

Yousri Mhedheb is a Ph.D. student at the Steinbuch Centre for Computing of the Karlsruhe Institute of Technology (KIT), Germany. He received his diploma in electrical engineering from the same university. His current research interest is high performance and large scale computing.

Foued Jrad is a Ph.D. student at the Steinbuch Centre for Computing of the Karlsruhe Institute of Technology (KIT), Germany. He received his diploma in electrical engineering from the University of Hanover, Germany. His current research interests include intercloud computing, cloud interoperability and cloud service brokerage.

Jie Tao obtained her Ph.D. degree at the Department of Computer Science of the Munich University of Technology, Germany. She is currently a senior research associate at the Steinbuch Centre for Computing, Karlsruhe Institute of Technology. Dr. Tao's research work targets mainly at parallel and distributed computing, data-intensive computing and grid & cloud computing. She has published a number of articles in peer-reviewed international journals and leading conferences. She serves as a co-chair or PC member in a set of international conferences and workshops. She is a guest editor of several international journals.

Qinghuai Liu is a full professor at the School of Basic Science, Changchun University of Technology. His major research interest is optimization algorithms and their applications. His research results have been published in peer-reviewed international conferences and journals.

Achim Streit is the director of the Steinbuch Centre for Computing, Karlsruhe Institute of Technology, and a full professor of distributed and parallel high performance systems. He received his Ph.D. in computer science from Paderborn University. From 2005 to 2010 he led the Division of Distributed Systems and Grid Computing at the Juelich Supercomputing Centre. As the director of the SCC, Professor Streit is responsible for HPC, scientific computing and simulation science, large scale and federated data management and analysis, grid and cloud computing. He is an author of more than 50 scientific peer-reviewed papers, journal articles and book chapters, as well as a PC member of more than 30 international workshops and conferences.

Received: 19 August 2013
Revised: 18 January 2014