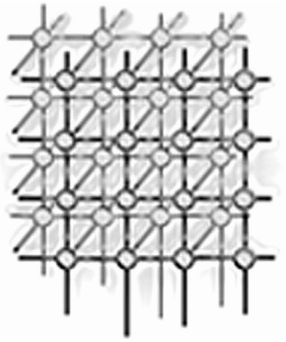
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2010; **00**:1–7 *Prepared using* cpeauth.cls *[Version: 2002/09/19 v2.02]*



Power-aware Provisioning of Virtual Machines for Real-time Cloud Services

Kyong Hoon Kim[1,*,†], Anton Beloglazov[2], and Rajkumar Buyya[2]

[1] *Department of Informatics, Gyeongsang National University Gajwadong 900, Jinju, South Korea*
[2] *Cloud Computing and Distributed Systems (CLOUDS) Lab, Department of Computer Science and Software Engineering, The University of Melbourne, Carlton 3053, Victoria 3010, Australia*

SUMMARY

Reducing power consumption has been an essential technique for Cloud resources and data centers not only to decrease operating costs, but also to improve the system reliability. As Cloud computing becomes emergent for *Anything as a Service (XaaS)* **paradigm, modern real-time Cloud services are also available throughout Cloud computing. In this work, we investigate power-aware provisioning of virtual machines for real-time services. Our approach is (i) to model a real-time service as a real-time virtual machine request; and (ii) to provision virtual machines in Cloud data centers using Dynamic Voltage Frequency Scaling (DVFS) schemes. We propose several schemes to reduce power consumption by hard real-time services, and suggest power-aware profitable provisioning of soft real-time services.**

KEY WORDS: Cloud Computing, Real-time services, Energy-Efficient Computing, Green Data Centers

1. Introduction

The development in computer and communication technologies has led to a new computing paradigm called *Cloud computing*, which delivers computing services to users as utilities in a pay-as-you-go manner [1]. Cloud providers offer various types of services, such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Service providers make use of IaaS and PaaS to deploy their services without concerns about managing

*Correspondence to: Department of Informatics, Gyeongsang National University, Gajwadong 900, Jinju, South Korea
†E-mail: khkim@gnu.ac.kr

Received 1 March 2010
Revised 19 September 2002

physical resources. Under the Cloud computing model, users can access on-demand and pay-per-use services anywhere in the world.

One of big challenges in data centers is to manage system resources in a power-efficient way. Data centers consume from 10 to 100 times more energy per square foot than typical office buildings [21]. They can even consume as much electricity as a city [17]. The main part of power consumption in data centers comes from computation processing, disk storage, network, and cooling systems. Lowering the energy usage of data centers becomes a challenging issue because computing applications and data are growing so quickly that increasingly larger servers and disks are needed to process them fast enough within the required time period [2]. Thus, data center resources need to be managed in an energy-efficient manner to drive *Green Cloud computing* [2]. In this paper, we study energy-efficient management of computing resources via the Virtual Machine (VM) provisioning, which is an essential architectural element in Cloud computing environments.

The pay-as-you-go mechanism in Cloud computing assures *Service Level Agreements (SLAs)* between customers and Cloud providers. SLAs specify the negotiated agreements on the *Quality of Service (QoS)*, such as deadline constraints. Thus, data centers must minimize power consumption without violating the SLAs. As many applications require deadline constraints, this paper focuses on power-aware management of real-time Cloud services, such as financial analysis, distributed image processing, real-time distributed databases, etc. The main contributions of this paper are (i) to provide a real-time Cloud service framework for requesting a virtual platform; (ii) to investigate various power-aware VM provisioning schemes based on Dynamic Voltage Frequency Scaling (DVFS) for hard real-time services; and (iii) to propose and analyze power-aware profitable VM provisioning of soft real-time services.

The remainder of this paper is organized as follows. Section 2 presents related work on power-aware Cloud computing. We propose the real-time Cloud service framework in Section 3. Section 4 describes the problem definition of hard real-time VM provisioning and provides several DVFS schemes. In Section 4, profitable VM provisioning of soft real-time services is analyzed based on DVFS schemes. We evaluate the proposed schemes throughout simulations in Section 6, and conclude the paper with a summary.

2. Related Work

Many of recent research works have focused on reducing power consumption in cluster systems. In [3, 29], high performance clusters with the consideration of power consumption have been designed and developed. As many recent commodity processors provide the DVFS ability, power-aware cluster systems have been built using such processors [12, 13]. Scientific applications developed based on the MPI library are mostly targeted at the reduction of power consumption [14, 12, 23]. Based on the profile of MPI programs, the authors choose an appropriate voltage scaling for each synchronization point.

General purpose cluster systems also have been studied on the reduction of power consumption. Srikantaiah et al. [24] have dealt with online services executing in heterogeneous clusters. When a new request comes, a heuristic for multidimensional bin packing is used to find a server to allocate the request. If a server cannot be found, a new machine is switched on

and all the requests are re-allocated. Chase et al. [9] have aimed at serving web-applications in homogeneous clusters according to a utility function. Gandhi et al. [11] have investigated the problem of minimizing the mean response time of web-applications on heterogeneous clusters. In this work the optimal energy allocation is determined based on a queuing theoretical model.

The recently emerged Cloud computing paradigm leverages virtualization of computing resources and allows the achievement of more efficient allocation of the workload in terms of higher resource utilization and decreased power consumption. Kusic et al. [16] have investigated the problem of minimizing both power consumption and SLA violations for online services in virtualized data centers using a limited look-ahead control. Verma et al. [26, 25] have proposed the *pMapper* architecture to solve the same problem considering the migration cost. Cardosa et al. [8] have presented several techniques for addressing the sharing-aware VM allocation problem. Hypervisor distributes resources among VMs according to a sharing-based mechanism, when the minimum and maximum amount of resources that can be allocated to a VM are specified.

In addition, many studies have focused on power-aware real-time applications in clusters. Rusu et al. [20] have developed a QoS-aware power management scheme by combining cluster-wide (On/ Off) and local (DVFS) power management techniques in the context of heterogeneous clusters. The front-end manager decides which servers should be turned on or off for a given system load, while the local manager reduces power consumption using DVFS. Wang et al. [28] have proposed a threshold-based method for efficient power management of heterogeneous soft real-time clusters as well as the offline mathematical analysis for determining the threshold. In addition, Kim et al. [15] have investigated power-aware algorithms for scheduling of real-time bag-of-tasks applications with deadline constraints in homogeneous clusters.

Considerable amount of work have been done in the area of power-efficient computing, but few of them deal with power-aware scheduling of real-time applications in Cloud computing environments. Buyya et al. [2] have presented their vision, challenges, and architectural elements for energy-efficient management of Cloud computing environments with consideration of QoS expectations. Kim et al. [31] have proposed a framework for provisioning of Cloud resources for hard real-time services. In this paper, we extend our earlier work [31] in order to support general real-time services, including soft real-time model. Thus, this work investigates the problem of provisioning Cloud resources for both hard and soft real-time services in order to minimize power consumption.

3. Framework

3.1. Real-time Service Model

A usual real-time service such as financial analysis, distributed database, or image processing, consists of multiple real-time applications or subtasks. As long as a group of applications for a given real-time service meet all their deadlines, the service accomplishes the QoS agreed with users. A real-time service is defined by $\{\tau_i(r_i, c_i, d_i, p_i, f_i)|i = 1, \ldots, n\}$, where n is the number of subtasks. Each real-time subtask τ_i is defined by the following parameters.

- r_i: release time
- c_i: worst-case execution time
- d_i: relative deadline
- p_i: period
- f_i: finish time

A real-time subtask τ_i can be started at time r_i and requires the worst-case execution time c_i. In order to accomplish the application's objective, it should be completed by the time $r_i + d_i$ after being released. Also, p_i specifies its periodicity so that the application releases a subtask of c_i computation time at the time $(r_i + kp_i)$, and should be finished by $r_i + kp_i + d_i$ $(k = 0, 1, \ldots)$. In case of non-periodic application, p_i is set to zero. We also consider the duration or finish time, f_i, since a user cannot have access to a Cloud computing resource forever, although a periodic real-time task in an embedded system assumes an infinite sequence.

A group of sub-tasks of a real-time service is developed and launched on a specific run-time platform including middleware, operating system, etc. A Cloud computing environment is a suitable solution for real-time services, as it leverages *virtualization*. When users request execution of their real-time services in a Cloud computing environment, appropriate VMs are provisioned for those services.

3.2. Real-Time Virtual Machine Model

According to the deadline model, real-time services can be categorized into two types: *hard* and *soft*. In the hard deadline model, a service receives some penalty if it does not meet the deadline. On the other hand, a service with a soft deadline provides a diminished value or utility even if the execution time exceeds the deadline. A penalty function, such as linear decreasing function, is used in the soft deadline model. Thus, we propose two different real-time VM models for Cloud computing.

3.2.1. Hard Real-Time Virtual Machine Model

In this subsection, we define *HRT-VM (Hard Real-Time Virtual Machine)* as a requirement for a VM providing a hard real-time service. HRT-VM V_i for a hard real-time service is described by three parameters: u_i, m_i, and d_i.

- u_i : the CPU utilization required for the real-time application
- m_i : the number of Million Instructions Per Second (MIPS) required for the base VM
- d_i : the lifetime or deadline

The service is developed and launched on a specific platform or infrastructure (e.g. 1GHz-Linux machine). We select the MIPS rate, m_i, for the specification of the base VM. For a given set of real-time applications, we can analyze the required CPU utilization u_i on the base VM. Thus, the above requirement implies that the real-time service is guaranteed when the allocated VM keeps providing $u_i \times m_i$ of the processing capacity by the deadline d_i. This real-time service on a virtualized Cloud resource is achieved by *compositional real-time computing* and *real-time VM* techniques.

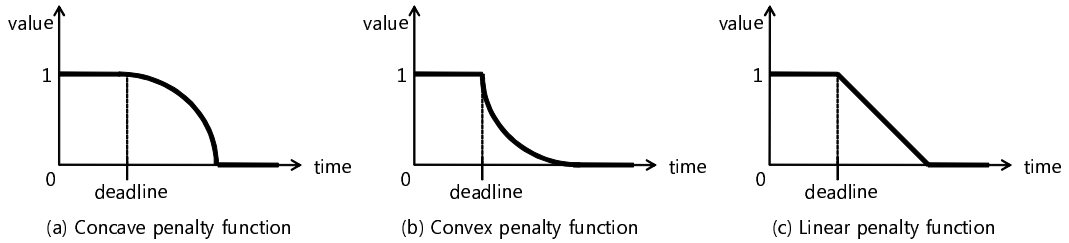


Figure 1. Various soft real-time service models

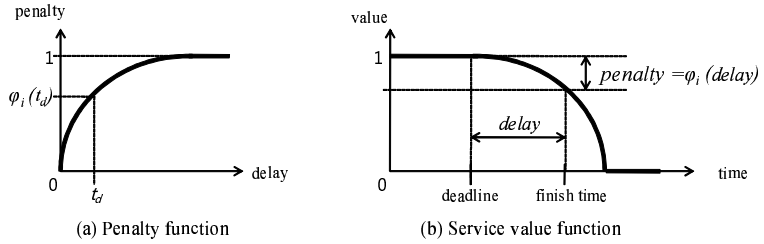


Figure 2. Quality of soft real-time service

3.2.2. Soft Real-Time Virtual Machine Model

The requirement for a VM providing a soft real-time service needs an additional parameter, called the *penalty function*. The penalty function indicates the diminished value of a service by executing a VM that has missed the deadline. Figure 1 shows examples of soft real-time service models with various penalty functions. If a real-time service meets its deadline, it provides the pre-determined value. However, when it misses the deadline, the value or utility of the service decreases according to its penalty function.

We define *SRT-VM (Soft Real-Time Virtual Machine)* as the soft real-time VM model. SRT-VM V_i for a soft real-time service is defined by (u_i, m_i, d_i, φ_i), where φ_i describes the penalty function of soft real-time service model. If the VM V_i is provided with $u_i \times m_i$ of the processing capacity by the deadline d_i, the service quality is guaranteed. However, the processing capacity of $u_i \times m_i$ is provided beyond the deadline, the service quality is defined by the penalty function φ_i. Let us assume that V_i is provided with the capacity past *delay* time units. Then, as shown in Figure 2, the service quality or *value* is given by Equation (1).

$$value = \begin{cases} 1 & \text{if } delay \leq 0 \\ \varphi_i(delay) & \text{if } delay > 0. \end{cases} \quad (1)$$

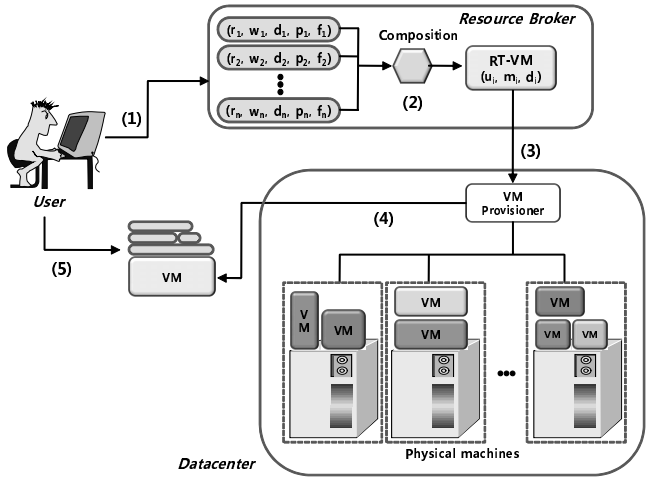


Figure 3. Framework

3.2.3. Real-Time Virtual Machine Request

Cloud resource brokers take the role of finding Cloud resources or VMs for real-time services requested by users. Thus, a user requests VMs by either HRT-VM or SRT-VM depending on deadline types, as described above. We denote a real-time VM request as *RT-VM* regardless of the deadline type.

The compositional or hierarchical real-time framework [10, 22] enables a group of real-time applications to be a single real-time resource requirement to the upper layer of a real-time environment. Thus, we assume that the RT-VM V_i is defined by multiple real-time applications, $\{\tau_k(r_k, w_k, d_k, p_k,\ f_k)|k = 1, \ldots, n\}$, of the service by using the compositional real-time technique. Thus, the VM provisioner in a Cloud maps VMs for the service rather than individual applications. Furthermore, recent work on implementing real-time VMs [27, 30] assures real-time services (e.g. real-time CPU allocation, real-time I/O) of a VM. This paper focuses on how to provision VMs to a given RT-VM request with consideration of power consumption by leveraging the described techniques.

3.3. Real-time Cloud Service Framework

In this subsection, we describe the real-time Cloud service framework based on the real-time VM model. As shown in Figure 3, the steps for a real-time service are as follows.

(1) *Requesting a virtual platform*: A user who wants to launch a real-time service submits all the information about real-time applications to the broker.

(2) *Generating a RT-VM from real-time applications*: The resource broker first analyzes the submitted real-time applications and generates one RT-VM request, $V_i = (u_i, m_i, d_i)$.
(3) *Requesting a real-time VM*: The broker requests a VM for RT-VM V_i from the VM provisioner of a Cloud computing environment.
(4) *Mapping physical processors*: The VM provisioner finds appropriate processing elements that meet the V_i requirements and provides the VM to the user.
(5) *Executing the real-time applications*: The user launches and executes the real-time applications using the provided VM.

3.4. Energy model

The main part of power consumption in data centers comes from computation processing, disk storage, network, and cooling systems. This paper focuses on reduction of CPU power consumption using energy-aware VM provisioning in Cloud computing environments.

The most of power consumption in CMOS circuits is composed of dynamic and static power. We only consider the dynamic power consumption, as it is the dominating factor in the total power consumption [18]. Data centers can increase their profit by reducing the dynamic power consumption. The dynamic power consumption by an application is proportional to V_{dd}^2 and f, where V_{dd} is the supply voltage and f is the frequency [6]. Since the frequency is usually in proportion to the supply voltage, the dynamic power consumption of a processor is defined in Equation (2).

$$P = C \cdot f^3, \tag{2}$$

where C is a proportional coefficient. Let us consider an application with the execution time t running at the CPU with the frequency f_{max}. If the processor runs at the frequency level f ($0 < f \leq f_{max}$), the execution time is defined by $t/\frac{f}{f_{max}}$. Thus, the dynamic power consumption during the task execution is defined by Equation (3).

$$\begin{aligned} E &= \int_0^{t/\frac{f}{f_{max}}} P = C \cdot t \cdot f_{max} \cdot f^2 \\ &= \alpha \cdot t \cdot S^2 \end{aligned} \tag{3}$$

where α is a coefficient and S is the relative processor speed for the frequency f ($S = f/f_{max}$). The DVFS scheme reduces the dynamic power consumption by decreasing the supplying voltage and frequency, which results in a slowdown of the CPU and increased execution time. We assume that each PE (Processing Element) p in a datacenter can adjust its processor frequency from f_p^{min} to f_p^{max} continuously. The relative processor speed S for each frequency f is defined by f/f_{max}, where $f_p^{min}/f_p^{max} < S \leq 1$.

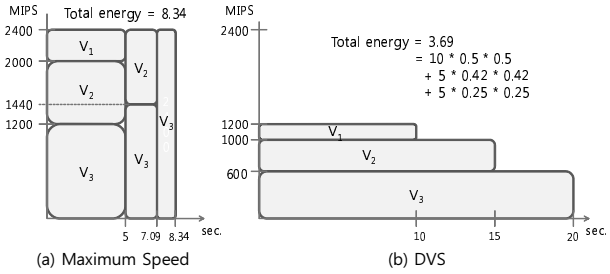


Figure 4. Proportional sharing of VM provisioning and power consumption ($\alpha = 1$)

4. Power-aware Hard Real-Time Cloud Service

4.1. Problem Description

Let us consider a physical machine with one PE of 2400 MIPS and a set of HRT-VMs, $\{V_1(0.2, 1000, 10), V_2(0.8, 500, 15), V_3(0.5, 1200, 20)\}$, as an example. V_1 requires the utilization 20% on 1000-MIPS machine by the deadline 10 sec. Similarly, V_2 and V_3 require 80% and 50% of 500-MIPS and 1200-MIPS machines by 15 and 20 seconds respectively. Figure 4(a) shows the proportional sharing scheduling result of these VMs under the maximum processor capacity. The proportional share of V_i is defined by $m_i \times u_i / \sum_{j=1}^{3}(m_j \times u_j)$. Three HRT-VMs share the processor capacity in proportion to their required MIPS rates, $m_i \times u_i$, and finish before the deadlines. The total power consumption is 8.34 ($= 1 \times 8.34 \times 1.0^2$) according to Equation (3) under the assumption of $\alpha = 1$.

The power consumption can be reduced by combining DVFS and the proportional sharing scheduling. As shown in Figure 4(b), the minimum required processor capacity is allocated to each VM, so that the processor dynamically adjust its speed to $\sum(m_j \times u_j)/2400$. The total power consumption of the DVFS scheme is 3.69 ($= 10 \times 0.5^2 + 5 \times 0.42^2 + 5 \times 0.25^2$). Thus, the DVFS scheme can significantly reduce power consumption compared to the maximum-speed static scheme.

However, there are tradeoffs in the dynamic scaling of the processor speed in on-line real-time Cloud computing. Operation at higher processor speed allows the acceptance of more HRT-VMs with higher power consumption. On the contrary, scaling down to a lower processor speed results in less consumed power with lower acceptance. For example, let us assume that a new HRT-VM V_4 (0.8, 2000, 10) is requested at the time 10. Figure 4(a) accepts V_4 since the processor is idle at the time 10, while the DVFS scheme shown in Figure 4(b) cannot provision it due to lack of the processor capacity.

Data centers can increase their profit by provisioning more VMs to users. However, reducing power consumption increases the profit by reducing the operating cost. To address this trade-

Table I. Remaining service times of Figure 4(a)

	$t=0$	$t=5$			$t=7.09$			$t=8.34$		
	w_i	Q_i [0, 5]	ST_i [0, 5]	w_i	Q_i [5, 7.09]	ST_i [5, 7.09]	w_i	Q_i [7.09, 8.34]	ST_i [7.09, 8.34]	w_i
V_1	2000	400	2000	0	-	-	-	-	-	-
V_2	6000	800	4000	2000	960	2000	0	-	-	-
V_3	12000	1200	6000	6000	1440	3010	2990	2400	2990	0

off, we propose several schemes for power-aware provisioning of real-time VMs for the purpose of maximizing profits in Cloud data centers.

We use the proportional sharing scheduling to schedule multiple VMs on a processor. The proportional sharing scheduling is simple but guarantees the real-time services of HRT-VMs if the total required MIPS rate is less than or equal to the processor capacity. Furthermore, it can be easily implemented. For example, the default scheduling in Xen Hypervisor [19] is *Credit scheduler* which is based on credit value set by weight of each VM. The VMM (Virtual Machine Monitor) can dynamically adjust credit values of VMs according to their required MIPS rates in order to support the proportional sharing scheme.

Before explaining the VM provisioning, we define the remaining service time, w_i, of V_i. The initial value of w_i is defined by $u_i \times m_i \times (d_i - t_s)$, at its submission time t_s. If V_i is provided with q_i MIPS rate for the period t_p, w_i is decreased by $q_i \times t_p$. For instance, Table I shows the remaining service times of the three described HRT-VMs at the time of proportional share change shown in Figure 4(a). V_i finishes its service when w_i becomes zero.

4.2. DVFS-enabled HRT-VM Provisioning

When a data center receives a HRT-VM request from a resource broker, it returns the price of providing the HRT-VM service if it can provide real-time VMs for that request. The broker selects the minimum priced VM among available data centers. Thus, the provisioning policy in this paper is to select the processing element with the minimum price for the sake of users. Figure 5 shows the pseudo-code of the algorithm for provisioning a VM for a given HRT-VM request.

For a given HRT-VM $V_i(u_i, m_i, d_i)$, the data center checks the schedulability of V_i on the processing element PE_k of Q_k MIPS rate. Suppose that the current running HRT-VMs on the processing element PE_k at time t is known as $T_k = \{V_j(u_j, m_j, d_j) | j = 1, \cdots, n_k\}$. And the remaining service time of V_j at the time t is denoted as w_j. Then, the schedulability is guaranteed if it satisfies Equation (4). Since $w_j/(d_j - t)$ is the minimum MIPS rate for V_j by its deadline d_j, Equation (4) means that the total summation of all the required MIPS rates including the new HRT-VM V_i is less than the processor capacity Q_k.

Algorithm Min-Price HRT-VM Provisioning (V_i)
1: $VM \leftarrow null$;
2: $alloc \leftarrow -1$;
3: $e_{min} \leftarrow$ MAX_VALUE;
4: $price_{min} \leftarrow$ MAX_VALUE;
5: **for** k **from** 1 **to** N **do**
6: **if** ($u_i \times m_i + \sum_{j=1}^{n_k} \frac{w_j}{d_j - t} \leq Q_k$) **then**
7: $e_k \leftarrow$ *energy_estimate* (PE_k, V_i);
8: $price_k \leftarrow$ *price for the HRT-VM* V_i *in* PE_k;
9: **if** $price_k < price_{min}$ **or**
10: ($price_k = price_{min}$ **and** $e_k < e_{min}$) **then**
11: $price_{min} \leftarrow price_k$;
12: $e_{min} \leftarrow e_k$;
13: $alloc \leftarrow k$;
14: **endif**
15: **endif**
16: **endfor**
17: **if** $alloc \neq -1$ **then**
18: $VM \leftarrow$ *create_VM* (PE_{alloc}, V_i);
19: **endif**
20: **return** *VM*;

Figure 5. Min-price HRT-VM provisioning

$$u_i \times m_i + \sum_{j=1}^{n_k} \frac{w_j}{d_j - t} \leq Q_k \tag{4}$$

If PE_k is able to schedule V_i, it estimates the energy and price of provisioning (line 7, 8). Since the provisioning policy is to provide lower price to users, the algorithm finds the minimum priced processor. For the same price, less energy is preferable because it produces higher profit (line 9-14). Finally, a VM is mapped on PE_{alloc} if HRT-VM V_i is schedulable in the data center.

When a user launches the service on the VM, the resource provider provisions the VM using DVFS schemes to reduce power consumption. We propose three power-aware VM provisioning schemes: *Lowest-DVFS*, *δ-Advanced-DVFS*, and *Adaptive-DVFS*. The following subsections describe them.

4.2.1. Lowest-DVFS for VM Provisioning

This scheme adjusts the processor speed to the lowest level at which HRT-VMs meet their deadlines. That is, each HRT-VM V_i executes its service at the required MIPS rate, as shown in Figure 4(b). It consumes the lowest energy in the case when the HRT-VM arrival rate is low enough to accept all the requests.

4.2.2. δ-Advanced-DVFS for VM Provisioning

In order to overcome the low service acceptance rate of Lowest-DVFS scheme, this scheme over-scales more up to $\delta\%$ of the required MIPS rate for current HRT-VMs. Thus, it operates the processor speed $\delta\%$ faster in order to increase the possibility of accepting incoming HRT-VM requests. The processor scale s is adjusted as in Equation (5) at the time t for a given HRT-VM set T_k. The proportional share of each VM is in proportion to $w_i/(d_i - t)$.

$$s = \min\left\{1,\ (1 + \frac{\delta}{100}) \times \frac{1}{Q_k} \sum_{Vi \in T_k} \frac{w_i}{d_i - t}\right\} \tag{5}$$

The value of $\delta\%$ is predefined in the system according to the system load. Throughout the simulation results in Section 5, we analyze the impact of δ.

4.2.3. Adaptive-DVFS for VM Provisioning

When the HRT-VM arrival rate and their service times are known in advance, we can analyze the optimal scale. Let us consider the $M/M/1$ queuing model with arrival rate λ and service rate μ. If the processor speed scale is set to s, then the average response time, RT, is given by $RT = 1/(s\mu - \lambda)$, according to the $M/M/1$ queuing model. In addition, the response time should be less than or equal to the average deadline, d, in order to to meet their real-time service requirements ($1/(s\mu - \lambda) \leq d$). Thus, the optimal scale, s^*, to reduce the power consumption is given by Equation (6).

$$s^* = \frac{1}{\mu}\left(\lambda + \frac{1}{d}\right) \tag{6}$$

Adaptive-DVFS scheme manages the average arrival rate $\hat{\lambda}$, the average service rate $\hat{\mu}$, and the average deadline $\hat{d}$ for the last h service requests (e.g. $h = 10$). It adjusts the processor scale s as in Equation (7) at time t for a given HRT-VM set T_k.

$$s = \max\left\{\min\left\{1,\ \frac{1}{\hat{\mu}}(\hat{\lambda} + \frac{1}{\hat{d}})\right\},\ \frac{1}{Q_k} \sum_{Vi \in T_k} \frac{w_i}{d_i - t}\right\} \tag{7}$$

In Equation (7), the optimal scale is calculated by Equation (6) not greater than one. Since it should be greater than the minimum required utilization of the current HRT-VMs on the processor, we select the maximum between the two values. The processor speed is adjusted according to Equation (7) when a new HRT-VM is provided or an existing one finishes its execution.

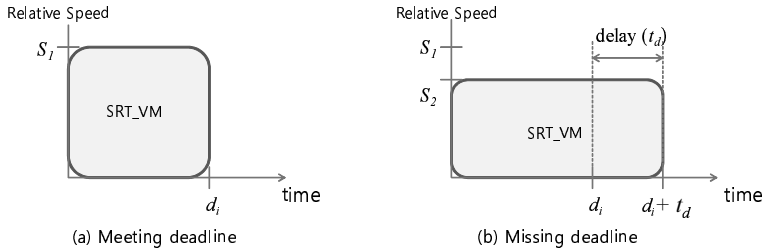


Figure 6. DVFS-based SRT-VM provisioning

5. Power-aware Soft Real-Time Cloud Service

5.1. Profitable Delay Analysis

When a soft real-time service misses its deadline, it gives a diminished value of the service to the user. In this case, the user does not need to pay the whole price for the Cloud resources because the resource provider has not met the QoS requirements. In this paper, we assume that a refund due to a service delay is in proportion to the diminished value of the service. The profit of a Cloud resource provider for providing a soft real-time service i with consideration of a refund is defined by Equation (8).

$$\begin{aligned} profit_i &= price_i \times value_i - \text{cost}_i \\ &= price_i \times (1 - \varphi_i(delay_i)) - \text{cost}_i, \end{aligned} \tag{8}$$

where $value_i$ is the service quality of the soft real-time service depending on the finish time, as shown in Figure 2; and $cost_i$ is the total cost including the power consumption.

Since a soft real-time service gives a value beyond the deadline, the power-aware provisioning may produce more profit in case of delayed service execution. Figure 6(a) shows the case of the highest profit when the deadline is met, as in Figure 4. Let us assume that the cost of t execution is in proportion to the power consumption as in Equation (3). Then, the cost function of t execution time is defined by Equation (9).

$$cost(t) = \beta \cdot t \cdot S^2, \tag{9}$$

where β is a coefficient and S is the associated relative processor speed.

Let us assume that the penalty function of a SRT-VM i is given by a linear function with the penalty rate k, as shown in Figure 1(c). Then, the profit of finishing at the time $d_i + t_d$, as shown in Figure 6(b), is given by Equation (10).

$$profit_i(d_i + t_d) = price_i \times (1 - kt_d) - \beta(d_i + t_d)S_2^2. \tag{10}$$

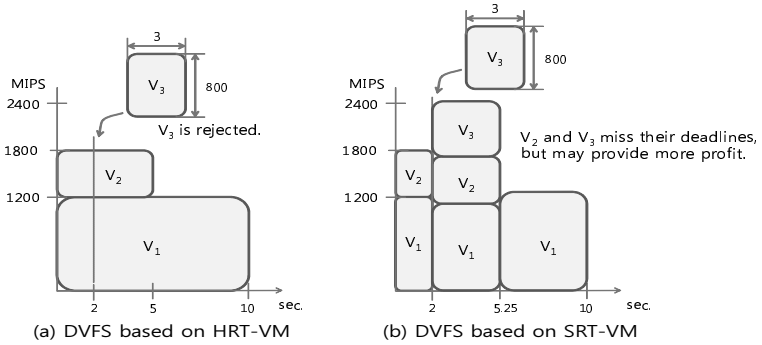


Figure 7. Acceptance of a RT-VM in HRT-VM and SRT-VM models

Thus, the condition that the profit of delaying the service of the SRT-VM i is greater than that of meeting deadline is given by Equation (11). That is, the case of Figure 6(b) may provide more profit to the resource provider than the case of Figure 6(a), if the delayed service time, t_d, satisfies Equation (11).

$$price_i \times (1 - kt_d) - \beta(d_i + t_d)S_2^2 > price_i - \beta d_i S_1^2 \tag{11}$$

The inequality of Equation (11) is shown in terms of the delayed service time, as in Equation (12).

$$t_d < \frac{S_1^2 - S_2^2}{price_i k + \beta S_2^2} \tag{12}$$

5.2. Resource Provisioning of SRT-VM

When an SRT-VM request is received, a data center provides for the resource broker an appropriate VM. Similarly to the HRT-VM provisioning algorithm in Figure 5, a data center finds for the user the minimum priced resource. The difference from the HRT-VM provisioning is the acceptance test of a VM. In Figure 5, a new VM request is accepted if all VMs on a PE including the new one meet their deadlines, as in Equation (4). However, the SRT-VM provisioning algorithm accepts a new VM request as long as the acceptance results in higher profit. For example, let us consider two VM requests V_1 and V_2 accepted at the time 0 on a PE, as shown in Figure 7(a). When a new VM V_3 requiring 800 MIPS rate during 3 time units arrives at the PE, the HRT-VM provisioning algorithm rejects the request since it cannot meet the deadline. On the contrary, SRT-VM can accept V_3 if the acceptance produces higher profit, as shown in Figure 7(b).

Therefore, the provisioning of SRT-VM should consider the profit in the acceptance test. Figure 8 shows the pseudo-code of the algorithm for the SRT-VM provisioning. The left side

Algorithm SRT-VM Provisioning (V_i)
$VM \leftarrow null$;
$alloc \leftarrow -1$;
$profit_{max} \leftarrow -1$;
$price_{min} \leftarrow$ MAX_VALUE;
for k **from** 1 **to** N **do**
  Let T_k be the set of current VMs on PE_k.
  $profit_{cur} \leftarrow$ *Calculate_Profit* (T_k, t);
  $profit_{new} \leftarrow$ *Calculate_Profit* $(T_k \cup \{V_i\}, t)$;
  $profit_k \leftarrow profit_{new} - profit_{cur}$;
  if $profit_k > 0$ **then**
    $price_k \leftarrow price\ of\ V_i\ in\ PE_k$;
    if $price_k < price_{min}$ **or**
      ($price_k = price_{min}$ **and**
       $profit_k > profit_{max}$) **then**
        $price_{min} \leftarrow price_k$;
        $profit_{max} \leftarrow profit_k$;
        $alloc \leftarrow k$;
    endif
  endif
endfor
if $alloc \neq -1$ **then**
  $VM \leftarrow create\_VM\ (PE_{alloc}, V_i)$;
endif
return *VM*;

function *Calculate_Profit* (T, t)
if $T = \phi$ **then return** 0;
$TotalMIPS \leftarrow 0$;
for j **from** 1 **to** $|T|$
  $MIPS_j \leftarrow w_j/(d_j - t)$;
  $TotalMIPS \leftarrow TotalMIPS + MIPS_j$;
endfor
if $TotalMIPS > Q_k$ **then**
  for j **from** 1 **to** $|T|$
    $MIPS_j \leftarrow Q_k \times MIPS_j / TotalMIPS$;
endif
for j **from** 1 **to** $|T|$
  $finishT_j \leftarrow w_j / MIPS_j$;
Let V_m be the VM with the smallest $finishT_j$ in T.
for j **from** 1 **to** $|T|$
  $w_j \leftarrow w_j - MIPS_j \times finishT_j$;
$delay \leftarrow finishT_m - d_m$;
$profit \leftarrow price_m(1 - \varphi(delay))$
    $-\alpha(finishT_m - t)(TotalMIPS/Q_k)^2$
  + *Calculate_Profit*$(T - \{V_m\}, finishT_m)$;
return *profit*
endfunction

Figure 8. Min-price profitable SRT-VM provisioning

of Figure 8 describes the minimum priced VM allocation, in which it calculates the profit by calling the function *Calculate_Profit ()*. The function *Calculate_Profit ()* is shown in the right side of Figure 8 for a given set of VM-request at the time t. If the number of VMs in T is n_k, there may be n_k different finish times. The function calculates the profit for each finish time by adding the VM profit minus the cost. The function *Calculate_Profit ()* implements this by calling the function recursively without deleting the earliest finishing VM, and adding the returned profit. The time complexity of the profit calculation is $O(n_k^2)$, where n_k is the number of VMs on a PE.

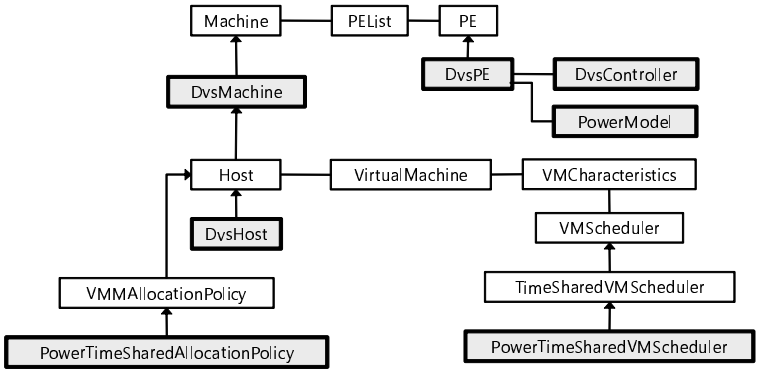


Figure 9. CloudSim architecture

Table II. Characteristics of datacenter

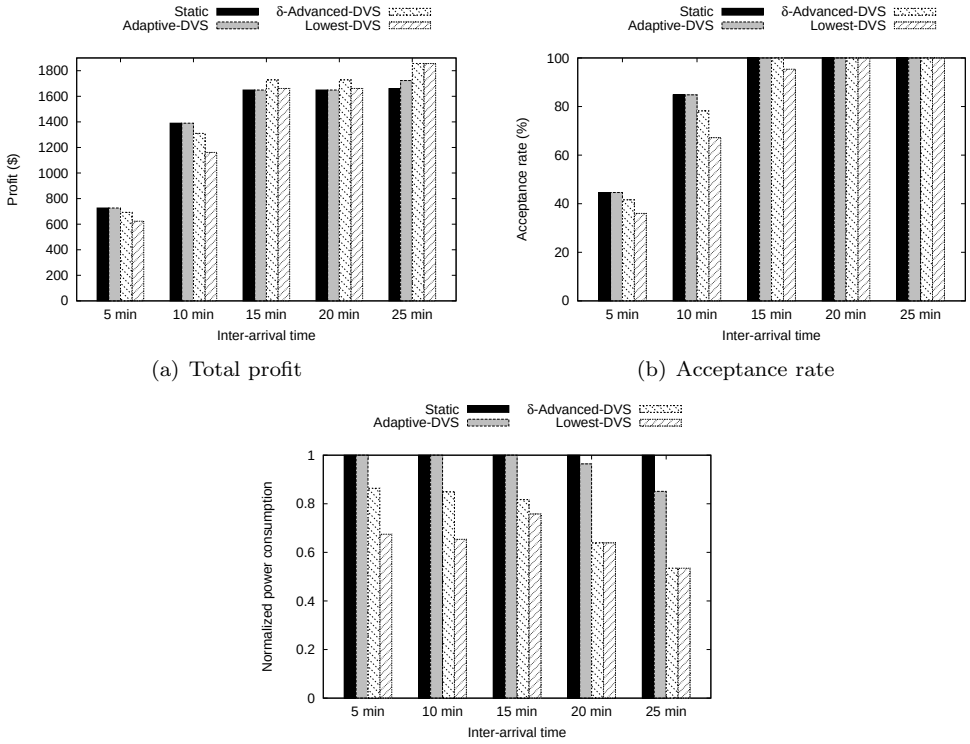
	# of PEs	MIPS of PE	DVFS level	α (10^{-3})
Machine 0	4	1,800	[0, 1.0]	2.92
Machine 1	4	2,400	[0, 1.0]	4.08
Machine 2	4	3,000	[0, 1.0]	5.37
Machine 3	4	3,400	[0, 1.0]	6.21

6. Simulation Results

We evaluate the proposed algorithms by simulations of power-aware real-time services using the CloudSim toolkit [7] with an extension enabling power-aware simulations. In order to support the power-aware VM provisioning, we have implemented additional components in the CloudSim toolkit, as shown in Figure 9. `DvsPE`, `DvsMachine`, and `DvsHost` are DVFS-enabled PE, machine, and host, respectively. The component `PowerTimeSharedAllocationPolicy` inherits the generic VM provisioning class and implements the simulated algorithms. The local scheduler in a VM is implemented by `PowerTimeSharedVMScheduler`.

We have created a data center with four machines with 16 DVFS-enabled processors with the characteristics shown in Table II. The price model in the simulations follows the Amazon EC2 Standard small (default) instance type [4], so that the unit price per hour equals to \$0.10. We use the cost function as the power consumption of each machine in Table II.

In the simulations, we have generated 500 HRT-VMs. The total service length (w_i) of a HRT-VM is randomly selected from 2,400 GIs (10^3 MIs) to 3,600 GIs. The deadline is selected from 10 to 30 minutes more than the execution time based on a 1000-MIPS machine. The



(a) Total profit

(b) Acceptance rate

(c) Normalized power consumption

Figure 10. Simulation results

interarrival time between two HRT-VMs follows a Poisson distribution. We have simulated various interarrival times.

Figure 10(a) shows the total profits generated by each scheme according to the interarrival time. **Static** does not use DVFS so that it runs VMs at the maximum processor speed. In δ**-Advanced-DVFS** we fix δ as 15%. For higher arrival rates, **Static** produces higher profits since it accepts more HRT-VMs. **Adaptive-DVFS** produces not less profit than **Static**, while other DVFS schemes generate higher profit for lower arrival rates due to lower power consumption.

Figure 10 (b) and (c) show the HRT-VM acceptance rate and the normalized power consumption compare to **Static**, respectively. The acceptance rate of **Adaptive-DVFS** is close to **Static** but reduces much energy in case of a low arrival rate. δ**-Advanced-DVFS** shows higher acceptance rate with similar power consumption compared to **Lowest-DVFS**. Generally, δ**-Advanced-DVFS** shows the best performance in terms of profit per unit of consumed power since the amount of scaling up is controlled automatically according to the

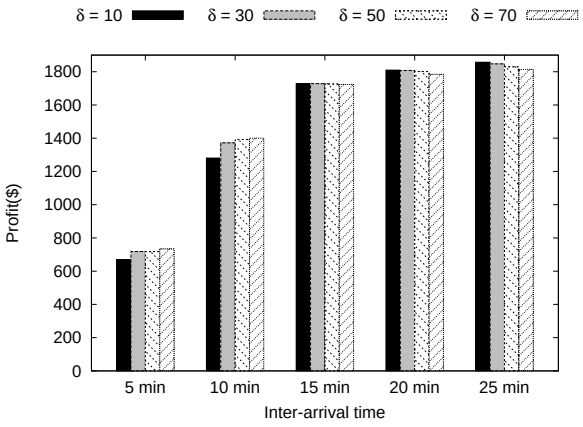


Figure 11. Impact of δ in δ-Advanced-DVFS

system load. In the case of **Adaptive-DVFS**, its performance is limited by the simplified queueing model.

Next, we have varied the value of δ in order to analyze the impact of δ. As shown in Figure 11, higher δ shows better performance for higher arrival rate since it may accept more VMs. On the contrary, lower δ produces higher profit in case of lower arrival rate. In the simulations, the system utilization is generally high regardless of arrival rates, so that δ has little impact on the profit.

7. Conclusion

In this paper, we have proposed a real-time Cloud service framework where each real-time service request is modeled as RT-VM in resource brokers. We have investigated power-aware provisioning of VMs for real-time Cloud services. For hard real-time services, we have provided several schemes and evaluated them using simulations. For soft real-time services, we have analyzed power-aware profitable VM provisioning and proposed a provisioning algorithm. The simulation results have shown that data centers can reduce power consumption and increase their profit using DVFS schemes. The proposed adaptive schemes, Adaptive-DVFS and δ-Advanced-DVFS, produce higher profit with lower power consumption regardless of the system load.

Our on-going work includes more analysis and improvement of the proposed adaptive schemes. For example, we will compare them with other approaches, such as bin packing and linear programming, and analyze the impact of the cooling systems. We also plan to

deeper investigate the soft real-time VM provisioning with the consideration of various penalty functions.

Acknowledgements

This is an extended version of the paper presented at the MGC 2009 Workshop [31]. This work was primarily carried out during the first author's visit to the CLOUDS Lab at the University of Melbourne.

REFERENCES

1. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*. 25(6):599-616, Elsevier Science, Amsterdam, The Netherlands, June 2009.
2. Buyya R, Beloglazov A, Abawajy J. Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges. In *Proc. of the 2010 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2010)*. Las Vegas, USA, July 2010.
3. Adiga ND, et al. An overview of the BlueGene/L supercomputer. In *Proc. of ACM/IEEE Conf. on Supercomputing*. Baltimore, USA, November 2002.
4. Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2.
5. Armbrust M, et al. Above the Clouds: A Berkeley view of cloud computing. *Tech. Report No. UCB/EECS-2009-28*, University of California at Berkeley, USA, February 2009.
6. Burd TD, Brodersen RW. Energy efficient cmos microprocessor design. In *Proc. of Annual Hawaii Intl. Conf. on System Sciences*, pages 288–297, January 1995.
7. Buyya R, Ranjan R, Calheiros RN. Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In *Proc. of the 7th High Performance Computing and Simulation (HPCS 2009)*. Leipzig, Germany, June 2009.
8. Cardosa M, Korupolu MR, Singh A. Shares and utilities based power consolidation in virtualized server environments. In *Proc. of IFIP/IEEE Intl. Symp. on Integrated Network Management*. USA, June 2009.
9. Chase JS, Anderson DC, Thakar PN, Vahdat AM, Doyle RP. Managing energy and server resources in hosting centers. In *Proc. of 8th ACM Symp. on Operating Systems Principles*. Banff, Canada, October 2001.
10. Feng XA, Mok AK. A model of hierarchical real-time virtual resources. In *Proc. of 23rd IEEE Real-Time Systems Symposium*. Austin, USA, Dec. 2002.
11. Gandhi A, Harchol-Balter M, Das R, Lefurgy C. Optimal power allocation in server farms. In *Proc. of Intl. Joint Conf. on Measurement and Modeling of Computer Systems*, pages 157–168. Seattle, USA, June 2009.
12. Ge R, Feng X, Cameron KW. Performance-constrained distributed DVS scheduling for scientific applications on power-aware clusters. In *Proc. of ACM/IEEE Conf. on Supercomputing*. Seattle, USA, November 2005.
13. Hsu C, Feng W. A power-aware run-time system for high-performance computing. In *Proc. of ACM/IEEE Conf. on Supercomputing*. Seattle, USA, November 2005.
14. Kappiah N, Freeh VW, Lowenthal DK. Just in time dynamic voltage scaling: Exploiting inter-node slack to save energy in MPI programs. In *Proc. of ACM/IEEE Conf. on Supercomputing*. Seattle, USA, November 2005.
15. Kim KH, Buyya R, Kim J. Power aware scheduling of bag-of-tasks applications with deadline constraints on DVS-enabled clusters. In *Proc. of 7th IEEE Intl. Symp. on Cluster Computing and the Grid (CCGrid'07)*, pages 541–548. Rio de Janeiro, Brazil, May 2007.
16. Kusic D, Kephart JO, Hanson JE, Kandasamy N, Jiang G. Power and performance management of virtualized computing environments via lookahead control. In *Proc. of 5th IEEE Intl. Conf. on Autonomic Computing (ICAC 2008)*, pages 3–12. Chicago, USA, June 2008.

17. Markoff J, Lohr S. Intel's huge bet turns iffy. *New York Times Technology Section*, September 2002.
18. Niu L, Quan G. Reducing both dynamic and leakage energy consumption for hard real-time systems. In *Proc. of CASES'04*. Washington, DC, USA, Sept. 2004.
19. Ongaro D, Cox A, Rixner S. Scheduling i/o in virtual machine monitors. In *Proc. of ACM SIGPLAN/SIGOPS Intl. Conf. on Virtual Execution Environments*, pages 1–10. Seattle, USA, March 2008.
20. Rusu C, Ferreira A, Scordino C, Watson A, Melhem R, Mosse D. Energy-efficient real-time heterogeneous server clusters. In *Proc. of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 418–428. San Jose, USA, April 2006.
21. Scheihing P. Creating energy efficient data centers. In *Data Center Facilities and Engineering Conference*. Washington, DC, USA, May 2007.
22. Shin I, Lee I. Compositional real-time scheduling framework with periodic model. *ACM Transactions on Embedded Computing Systems*, 7(3), April 2008.
23. Son SW, Malkowski K, Chen G, Kandemir M, Raghavan P. Integrated link/cpu voltage scaling for reducing energy consumption of parallel sparse matrix applications. In *Proc. of 20th IEEE Intl. Parallel and Distributed Processing Symposium*. Greece, April 2006.
24. Srikantaiah S, Kansal A, Zhao F. Energy aware consolidation for cloud computing. In *Workshop on Power Aware Computing and Systems (HotPower '08)*. San Diego, USA, December 2008.
25. Verma A, Ahuja P, Neogi A. pMapper: Power and migration cost aware application placement in virtualized systems. In *Proc. of 9th ACM/IFIP/USENIX Intl. Conf. on Middleware*. Leuven, Belgium, December 2008.
26. Verma A, Ahuja P, Neogi A. Power-aware dynamic placement of HPC applications. In *Proc. of ICS'08*, pages 175–184. Agean Sea, Greece, June 2008.
27. VirtualLogicx Real-Time Virutulalization and VLX. VirtualLogix, http://www.osware.com.
28. Wang L, Lu Y. Efficient power management of heterogeneous soft real-time clusters. In *Proc. of IEEE Real-Time Systems Sym.* Barcelona, Spain, Dec. 2008.
29. Warren W, Weigle E, Feng W. High-density computing: A 240-node Beowulf in one cubic meter. In *Proc. of ACM/IEEE Conf. on Supercomputing*. Baltimore, USA, November 2002.
30. Yoo S, Park M, Yoo C. A step to support real-time in a virtual machine monitor. In *Proc. of 6th IEEE Consumer Communications and Networking Conference*. Las Vegas, USA, January 2009.
31. Kim KH, Beloglazov A, Buyya R. Power-aware provisioning of cloud resources for real-time services. In *Proc. of 7th International Workshop on Middleware for Grids, Clouds and e-Science (MGC 2009)*. Urbana Champaign, USA, December 2009.