CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2012; **24**:1397–1420
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.1867

Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers

Anton Beloglazov* and Rajkumar Buyya

Cloud Computing and Distributed Systems (CLOUDS) Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia

SUMMARY

The rapid growth in demand for computational power driven by modern service applications combined with the shift to the Cloud computing model have led to the establishment of large-scale virtualized data centers. Such data centers consume enormous amounts of electrical energy resulting in high operating costs and carbon dioxide emissions. Dynamic consolidation of virtual machines (VMs) using live migration and switching idle nodes to the sleep mode allow Cloud providers to optimize resource usage and reduce energy consumption. However, the obligation of providing high quality of service to customers leads to the necessity in dealing with the energy-performance trade-off, as aggressive consolidation may lead to performance degradation. Due to the variability of workloads experienced by modern applications, the VM placement should be optimized continuously in an online manner. To understand the implications of the online nature of the problem, we conduct competitive analysis and prove competitive ratios of optimal online deterministic algorithms for the single VM migration and dynamic VM consolidation problems. Furthermore, we propose novel adaptive heuristics for dynamic consolidation of VMs based on an analysis of historical data from the resource usage by VMs. The proposed algorithms significantly reduce energy consumption, while ensuring a high level of adherence to the Service Level Agreements (SLA). We validate the high efficiency of the proposed algorithms by extensive simulations using real-world workload traces from more than a thousand PlanetLab VMs. Copyright © 2012 John Wiley & Sons, Ltd.

Received 16 May 2011; Accepted 1 September 2011

KEY WORDS: Green IT; Cloud computing; resource management; virtualization; dynamic consolidation

1. INTRODUCTION

The Cloud computing model leverages virtualization of computing resources allowing customers to provision resources on-demand on a pay-as-you-go basis [1]. Instead of incurring high upfront costs in purchasing IT infrastructure and dealing with the maintenance and upgrades of both software and hardware, organizations can outsource their computational needs to the Cloud. The proliferation of Cloud computing has resulted in the establishment of large-scale data centers containing thousands of computing nodes and consuming enormous amounts of electrical energy. Based on the trends from American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) [2], it has been estimated that by 2014 infrastructure and energy costs would contribute about 75%, whereas IT would contribute just 25% to the overall cost of operating a data center [3].

*Correspondence to: CLOUDS Lab, Department of Computer Science and Software Engineering, The University of Melbourne, Australia. Email: abe@csse.unimelb.edu.au

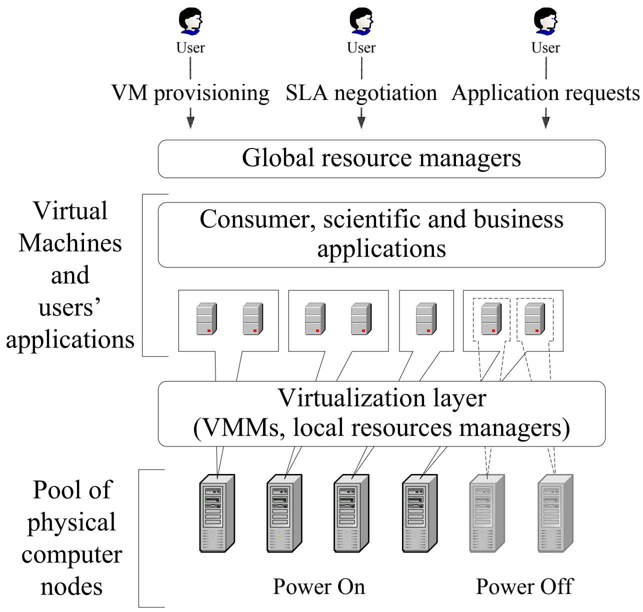


Figure 1. The system view

The reason for this extremely high energy consumption is not just the quantity of computing resources and the power inefficiency of hardware, but rather lies in the inefficient usage of these resources. Data collected from more than 5000 production servers over a six-month period have shown that although servers usually are not idle, the utilization rarely approaches 100% [4]. Most of the time servers operate at 10-50% of their full capacity, leading to extra expenses on over-provisioning, and thus extra Total Cost of Acquisition (TCA) [4]. Moreover, managing and maintaining over-provisioned resources results in the increased Total Cost of Ownership (TCO). Another problem is the narrow dynamic power range of servers: even completely idle servers still consume about 70% of their peak power [5]. Therefore, keeping servers underutilized is highly inefficient from the energy consumption perspective. Assuncao et al. [6] have conducted a comprehensive study on monitoring energy consumption by the Grid'5000 infrastructure. They have shown that there exist significant opportunities for energy conservation via techniques utilizing switching servers off or to low power modes. There are other crucial problems that arise from high power and energy consumption by computing resources. Power is required to feed the cooling system operation. For each watt of power consumed by computing resources, an additional 0.5-1 W is required for the cooling system [7]. In addition, high energy consumption by the infrastructure leads to substantial carbon dioxide (CO_2) emissions contributing to the greenhouse effect [8].

One of the ways to address the energy inefficiency is to leverage the capabilities of the virtualization technology [9]. The virtualization technology allows Cloud providers to create multiple Virtual Machine (VMs) instances on a single physical server, thus improving the utilization of resources and increasing the Return On Investment (ROI). The reduction in energy consumption can be achieved by switching idle nodes to low-power modes (i.e., sleep, hibernation), thus eliminating the idle power consumption (Figure 1). Moreover, by using live migration [10] the VMs can be dynamically consolidated to the minimal number of physical nodes according to their current resource requirements. However, efficient resource management in Clouds is not trivial, as modern service applications often experience highly variable workloads causing dynamic resource usage patterns. Therefore, aggressive consolidation of VMs can lead to performance degradation when an application encounters an increasing demand resulting in an unexpected rise of the resource usage. If the resource requirements of an application are not fulfilled, the application can face increased response times, time-outs or failures. Ensuring reliable Quality of Service (QoS) defined via Service Level Agreements (SLAs) established between Cloud providers and their customers

DOI: 10.1002/cpe.1867

is essential for Cloud computing environments; therefore, Cloud providers have to deal with the energy-performance trade-off – the minimization of energy consumption, while meeting the SLAs.

The focus of this work is on energy and performance efficient resource management strategies that can be applied in a virtualized data center by a Cloud provider (e.g. Amazon EC2). We investigate performance characteristics of online algorithms for the problem of energy and performance efficient dynamic VM consolidation. First, we study a simplified problem of determining the time to migrate a VM from an oversubscribed host to minimize the cost consisting of the cost of energy consumption and the cost incurred by the Cloud provider due to the SLA violation. We determine and prove the cost of an optimal offline algorithm for this problem, as well as the competitive ratio of an optimal online deterministic algorithm. Next, we investigate a more complex problem of dynamic consolidation of VMs considering multiple hosts and multiple VMs. We find and prove the competitive ratio of an optimal online deterministic algorithm for this problem.

It is widely known that randomized online algorithms usually provide better performance than deterministic algorithms designed for the same problems [11]. Therefore, we enhance deterministic algorithms and propose and evaluate novel heuristics that adapt their behavior based on an analysis of historical data from the resource usage by VMs. We evaluate the proposed algorithms by extensive simulation using the CloudSim toolkit and the workload data from 10 days of the resource usage by more than a thousand PlanetLab VMs provisioned for multiple users. The algorithms significantly reduce energy consumption, while providing a high level of adherence to the SLAs. The main contributions of this paper are the following.

1. Formal definitions of optimal online deterministic and offline algorithms for the single VM migration and dynamic VM consolidation problems.
2. A proof of the cost incurred by an optimal offline algorithm for the single VM migration problem.
3. Competitive analysis and proofs of the competitive ratios of optimal online deterministic algorithms for the single VM migration and dynamic VM consolidation problems.
4. Novel adaptive heuristics for the problem of energy and performance efficient dynamic consolidation of VMs that outperform an optimal online deterministic algorithm.
5. An extensive simulation-based evaluation and performance analysis of the proposed algorithms.

The remainder of the paper is organized as follows. In Section 2 we discuss the related work. In Sections 3 and 4 we present a thorough analysis of the single VM migration and dynamic VM consolidation problems respectively. In Section 5 we introduce the system model used in the development of heuristics for the dynamic VM consolidation problem. We propose our adaptive heuristics in Section 6, continuing with an evaluation in Section 7 and analysis of the obtained experiment results in Section 7. We discuss future research directions and conclude the paper in Section 8.

2. RELATED WORK

One of the first works, in which power management has been applied in the context of virtualized data centers, has been done by Nathuji and Schwan [12]. The authors have proposed an architecture of a data center's resource management system where resource management is divided into local and global policies. At the local level the system leverages the guest OS's power management strategies. The global manager gets the information on the current resource allocation from the local managers and applies its policy to decide whether the VM placement needs to be adapted. However, the authors have not proposed a specific policy for automatic resource management at the global level.

Kusic et al. [13] have defined the problem of power management in virtualized heterogeneous environments as a sequential optimization and addressed it using Limited Lookahead Control (LLC). The objective is to maximize the resource provider's profit by minimizing both power consumption and SLA violation. Kalman filter is applied to estimate the number of future requests to predict the future state of the system and perform necessary reallocations. However, in contrast

DOI: 10.1002/cpe.1867

to heuristic-based approaches, the proposed model requires simulation-based learning for the application-specific adjustments, which cannot be implemented by Infrastructure as a Service (IaaS) Cloud providers, such as Amazon EC2. Moreover, due to the model complexity the execution time of the optimization controller reaches 30 minutes even for 15 nodes, which is not suitable for large-scale real-world systems. On the contrary, our approach is heuristic-based, which does not require simulation based learning prior to the application deployment and allows the achievement of high performance even for a large scale as shown by our experiments.

Srikantaiah et al. [14] have studied the problem of request scheduling for multi-tier web-applications in virtualized heterogeneous systems to minimize energy consumption, while meeting performance requirements. The authors have investigated the effect of performance degradation due to high utilization of different resources when the workload is consolidated. They have found that the energy consumption per transaction results in a "U"-shaped curve, and it is possible to determine the optimal utilization point. To handle the optimization over multiple resources, the authors have proposed a heuristic for the multidimensional bin packing problem as an algorithm for the workload consolidation. However, the proposed approach is workload type and application dependent, whereas our algorithms are independent of the workload type, and thus are suitable for a generic Cloud environment. Cardosa et al. [15] have proposed an approach for the problem of power-efficient allocation of VMs in virtualized heterogeneous computing environments. They have leveraged the min, max and shares parameters of Xen's VMM, which represent minimum, maximum and proportion of the CPU allocated to VMs sharing the same resource. However, the approach suits only enterprise environments as it does not support strict SLAs and requires the knowledge of application priorities to define the shares parameter. Other limitations are that the allocation of VMs is not adapted at run-time (the allocation is static).

Verma et al. [16] have formulated the problem of power-aware dynamic placement of applications in virtualized heterogeneous systems as continuous optimization: at each time frame, the placement of VMs is optimized to minimize power consumption and maximize performance. Like in [14], the authors have applied a heuristic for the bin packing problem with variable bin sizes and costs. Similarly to [12], live migration of VMs is used to achieve a new placement at each time frame. The proposed algorithms, on the contrary to our approach, do not support SLAs: the performance of applications can be degraded due to the workload variability. In their more recent work [17], Verma et al. have proposed dividing VM consolidation strategies into static (monthly, yearly), semi-static (days, weeks) and dynamic (minutes, hours) consolidation. In the paper, the authors have focused on static and semi-static consolidation techniques, as these types of consolidation are easier to implement in an enterprise environment. In contrast, in this work we investigate the problem of dynamic consolidation to take advantage of fine-grained optimization. Gandhi et al. [18] have investigated the problem of allocating an available power budget among servers in a virtualized heterogeneous server farm, while minimizing the mean response time. To investigate the effect of different factors on the mean response time, a queuing theoretic model has been introduced, which allows the prediction of the mean response time as a function of the power-to-frequency relationship, arrival rate, peak power budget, etc. The model is used to determine the optimal power allocation for every configuration of the above factors.

Jung et al. [19], [20] have investigated the problem of dynamic consolidation of VMs running a multi-tier web-application using live migration, while meeting SLA requirements. The SLA requirements are modeled as the response time precomputed for each type of transactions specific to the web-application. A new VM placement is produced using bin packing and gradient search techniques. The migration controller decides whether there is a reconfiguration that is effective according to the utility function that accounts for the SLA fulfillment. However, this approach can be applied only to a single web-application setup and, therefore, cannot be utilized for a multi-tenant IaaS environment. Zhu et al [21] have studied a similar problem of automated resource allocation and capacity planning. They have proposed three individual controllers each operating at a different time scale: longest time scale (hours to days); shorter time scale (minutes); and shortest time scale (seconds). These three controllers place compatible workloads onto groups of servers, react to changing conditions by reallocating VMs, and allocate resources to VMs within

Concurrency Computat.: Pract. Exper. (2012)
DOI: 10.1002/cpe.1867

the servers to satisfy the SLAs. The middle-scale controller is the closest to the scope of our work. This approach is in line with our previous work [22] and applies an approach based on the idea of setting fixed utilization thresholds. However, fixed utilization thresholds are not efficient for IaaS environments with mixed workloads that exhibit non-stationary resource usage patterns.

Kumar et al. [23] have proposed an approach for dynamic VM consolidation based on an estimation of "stability" – the probability that a proposed VM reallocation will remain effective for some time in the future. Predictions of future resource demands of applications are done using a time-varying probability density function. The problem is that the authors assume that the parameters of the distribution, such as the mean and standard deviation, are known *a priori*. They assume that these values can be obtained using offline profiling of applications and online calibration. However, offline profiling is unrealistic for IaaS environments. Moreover, the authors assume that the resource utilization follows a normal distribution, whereas numerous studies [24], [25], [26] have shown that resource usage by applications is more complex and cannot be modeled using simple probability distributions. Berral et al [27] have studied the problem of dynamic consolidation of VMs running applications with deadlines that are set in the SLAs. Using machine learning techniques they optimize the combination of energy consumption and SLA fulfillment. The proposed approach is designed for specific environments, such as High Performance Computing (HPC), where applications have deadline constraints. Therefore, such an approach is not suitable for environments with mixed workloads.

In contrast to the discussed studies, we propose efficient adaptive heuristics for dynamic adaption of VM allocation at run-time according to the current utilization of resources applying live migration, switching idle nodes to the sleep mode, and thus minimizing energy consumption. The proposed approach can effectively handle strict QoS requirements, multi-core CPU architectures, heterogeneous infrastructure and heterogeneous VMs. The algorithms adapt the behavior according to the observed performance characteristics of VMs. Moreover, to the best of our knowledge, in the literature there have not been results in competitive analysis of online algorithms for the problem of energy and performance efficient dynamic consolidation of VMs.

3. THE SINGLE VM MIGRATION PROBLEM

In this section we apply competitive analysis [28] to analyze a subproblem of the problem of energy and performance efficient dynamic consolidation of VMs. There is a single physical server, or host, and M VMs allocated to that host. In this problem the time is discrete and can be split into N time frames, where each time frame is 1 second. The resource provider pays the cost of energy consumed by the physical server. It is calculated as $C_p t_p$, where C_p is the cost of power (i.e., energy per unit of time), and t_p is a time period. The resource capacity of the host and resource usage by VMs are characterized by a single parameter, the CPU performance. The VMs experience dynamic workloads, which means that the CPU usage by a VM arbitrarily varies over time. The host is oversubscribed, i.e., if all the VMs request their maximum allowed CPU performance, the total CPU demand will exceed the capacity of the CPU. We define that when the demand of the CPU performance exceeds the available capacity, a violation of the SLAs established between the resource provider and customers occurs. An SLA violation results in a penalty incurred by the provider, which is calculated as $C_v t_v$, where C_v is the cost of SLA violation per unit of time, and t_v is the time duration of the SLA violation. Without loss of generality, we can define $C_p = 1$ and $C_v = s$, where $s \in \mathbb{R}^+$. This is equivalent to defining $C_p = 1/s$ and $C_v = 1$.

At some point in time v, an SLA violation occurs and continues until N. In other words, due to the over-subscription and variability of the workload experienced by VMs, at the time v the overall demand for the CPU performance exceeds the available CPU capacity and does not decrease until N. It is assumed that according to the problem definition, a single VM can be migrated out from the host. This migration leads to a decrease of the demand for the CPU performance and makes it lower than the CPU capacity. We define n to be the stopping time, which is equal to the latest of either the end of the VM migration or the beginning of the SLA violation. A VM migration takes time T. During a migration an extra host is used to accommodate the VM being migrated, and therefore,

Concurrency Computat.: Pract. Exper. (2012)
DOI: 10.1002/cpe.1867

the total energy consumed during a VM migration is $2C_pT$. The problem is to determine the time m when a VM migration should be initiated to minimize the total cost consisting of the energy cost and the cost caused by an SLA violation if it takes place. Let r be the remaining time since the beginning of the SLA violation, i.e., $r = n - v$.

3.1. *The Cost Function*

To analyze the problem, we define a cost function as follows. The total cost includes the cost caused by the SLA violation and the cost of the *extra* energy consumption. The extra energy consumption is the energy consumed by the extra host where a VM is migrated to, and the energy consumed by the main host after the beginning of the SLA violation. In other words, all the energy consumption is taken into account except for the energy consumed by the main host from t_0 (the starting time) to v. The reason is that this part of energy cannot be eliminated by any algorithm by the problem definition. Another restriction is that the SLA violation cannot occur until a migration starting at t_0 can be finished, i.e., $v > T$. According to the problem statement, we define the cost function $C(v, m)$ as shown in (1).

$$C(v,m) = \begin{cases} (v-m)C_p & \text{if } m < v, v-m \geq T, \\ (v-m)C_p + 2(m-v+T)C_p + (m-v+T)C_v & \text{if } m \leq v, v-m < T, \\ rC_p + (r-m+v)C_p + rC_v & \text{if } m > v. \end{cases} \quad (1)$$

The cost function C defines three cases, which cover all possible relationships between v and m. We denote the cases of (1) as C_1, C_2, and C_3 respectively. C_1 describes the case when the migration occurs before the occurrence of the SLA violation ($m < v$), but the migration starts not later than T before the beginning of the SLA violation ($v - m \geq T$). In this case the cost is just $(v - m)C_p$, i.e., the cost of energy consumed by the extra host from the beginning of the VM migration to the beginning of the potential SLA violation. There is no cost of SLA violation, as according to the problem statement the stopping time is exactly the beginning of the potential SLA violation, so the duration of the SLA violation is 0.

C_2 describes the case when the migration occurs before the occurrence of the SLA violation ($m \leq v$), but the migration starts later than T before the beginning of the SLA violation ($v - m < T$). C_2 contains three terms: (a) $(v - m)C_p$, the cost of energy consumed by the extra host from the beginning of the migration to the beginning of the SLA violation; (b) $2(m - v + T)C_p$, the cost of energy consumed by both the main host and the extra host from the beginning of the SLA violation to n; (c) $(m - v + T)C_v$, the cost of the SLA violation from the beginning of the SLA violation to the end of the VM migration. C_3 describes the case when the migration starts after the beginning of the SLA violation. In this case the cost consists of three terms: (a) rC_p, the cost of energy consumed by the main host from the beginning of the SLA violation to n; (b) $(r - m + v)C_p$, the cost of energy consumed by the extra host from the beginning of the VM migration to n; (c) rC_v, the cost of SLA violation from the beginning of the SLA violation to n.

3.2. *The Cost of an Optimal Offline Algorithm*

Theorem 1
An optimal offline algorithm for the single VM migration problem incurs the cost of $\frac{T}{s}$, and is achieved when $\frac{v-m}{T} = 1$.

Proof
To find the cost incurred by an optimal offline algorithm, we analyze the range of the cost function for the domain of all possible algorithms. The quality of an algorithm for this problem depends of the relation between v and m, i.e., on the difference between the time when the VM migration is initiated by the algorithm and the time when the SLA violation starts. We can define $v - m = aT$, where $a \in \mathbb{R}$. Therefore, $m = v - aT$, and $a = \frac{v-m}{T}$. Further, we analyze the three cases defined by the cost function (1).

DOI: 10.1002/cpe.1867

1. $m < v$, $v - m \geq T$. Thus, $aT \geq T$ and $a \geq 1$. By the substitution of $m = v - aT$ in the first case of (1), we get (2).

$$C_1(v, a) = (v - v + aT)C_p = aTC_p \tag{2}$$

2. $m \leq v$, $v - m < T$. Thus, $a \geq 0$ and $aT < T$. Therefore, $0 \leq a < 1$. By the substitution of $m = v - aT$ in the second case of (1), we get (3).

$$\begin{aligned} C_2(v, a) &= (v - v + aT)C_p + 2(v - aT - v + T)C_p + (v - aT - v + T)C_v \\ &= aTC_p + 2T(1 - a)C_p + T(1 - a)C_v \\ &= T(2 - a)C_p + T(1 - a)C_v \end{aligned} \tag{3}$$

3. $m > v$. Thus, $a < 0$. By simplifying the third case of (1), we get (4).

$$\begin{aligned} C_3(v, m) &= rC_p + (r - m + v)C_p + rC_v \\ &= (2r - m + v)C_p + rC_v \end{aligned} \tag{4}$$

For this case, r is the time from the beginning of the SLA violation to the end of the migration. Therefore, $r = m - v + T$. By the substitution of m, we get $r = T(1 - a)$. By the substitution of m and r in (4), we get (5).

$$\begin{aligned} C_3(v, a) &= (2T - 2aT - v + aT + v)C_p + T(1 - a)C_v \\ &= T(2 - a)C_p + T(1 - a)C_v \\ &= C_2(v, a) \end{aligned} \tag{5}$$

Since $C_3(v, a) = C_2(v, a)$, we simplify the function to just two case. Both cases are linear in a and do not depend on v (6).

$$C(a) = \begin{cases} T(2 - a)C_p + T(1 - a)C_v & \text{if } a < 1, \\ aTC_p & \text{if } a \geq 1. \end{cases} \tag{6}$$

According to the problem definition, we can make the following substitutions: $C_p = 1/s$ and $C_v = 1$ (7).

$$C(a) = \begin{cases} \frac{T(2-a)}{s} + T(1 - a) & \text{if } a < 1, \\ \frac{aT}{s} & \text{if } a \geq 1. \end{cases} \tag{7}$$

It is clear that (7) reaches its minimum $\frac{T}{s}$ at $a = 1$, i.e., when $m = v - T$. This solution corresponds to an algorithm that always initiates the VM migration exactly at $m = v - T$. Such an algorithm must have perfect knowledge of the time when the SLA violation will occur before it actually occurs. An algorithm that satisfies this requirement is an optimal offline algorithm for the single VM migration problem.

□

3.3. *An Optimal Online Deterministic Algorithm*

In a real world setting, a control algorithm does not have the complete knowledge of future events, and therefore, has to deal with an *online problem*. According to Borodin and El-Yaniv [28], optimization problems in which the input is received in an online manner and in which the output must be produced online are called *online problems*. Algorithms that are designed for online problems are called *online algorithms*. One of the ways to characterize the performance and efficiency of online algorithms is to apply competitive analysis. In the framework of competitive analysis, the quality of online algorithms is measured relatively to the best possible performance of algorithms that have complete knowledge of the future. An online algorithm ALG is *c-competitive* if there is a constant a, such that for all finite sequences I:

$$ALG(I) \leq c \cdot OPT(I) + a, \tag{8}$$

DOI: 10.1002/cpe.1867

where $ALG(I)$ is the cost incurred by ALG for the input I; $OPT(I)$ is the cost of an optimal offline algorithm for the input sequence I; and a is a constant. This means that for all possible inputs, ALG incurs a cost within the constant factor c of the optimal offline cost plus a constant a. c can be a function of the problem parameters, but it must be independent of the input I. If ALG is c-competitive, we say that ALG attains a *competitive ratio* c. In competitive analysis, an online deterministic algorithm is analyzed against the input generated by an omnipotent malicious adversary. Based on the knowledge of the online algorithm, the adversary generates the worst possible input for the online algorithm, i.e., the input that maximizes the competitive ratio. An algorithm's *configuration* is the algorithm's state with respect to the outside world, which should not be confused with the algorithm's internal state consisting of its control and internal memory.

We continue the analysis of the single VM migration problem by finding an optimal online deterministic algorithm and its competitive ratio.

Theorem 2
The competitive ratio of an optimal online deterministic algorithm for the single VM migration problem is $2 + s$, and the algorithm is achieved when $m = v$.

Proof

Using the cost function found in Theorem 1, the competitive ratio of any online algorithm is defined as in (9).

$$\frac{ALG(I)}{OPT(I)} = \begin{cases} \frac{T(2-a)+sT(1-a)}{s} \cdot \frac{s}{T} = 2 + s - a(1+s) & \text{if } a < 1, \\ \frac{aT}{s} \cdot \frac{s}{T} = a & \text{if } a \geq 1, \end{cases} \tag{9}$$

where $a = \frac{v-m}{T}$. The configuration of any online algorithm for the single VM migration problem is the current time i; the knowledge of whether an SLA violation is in place; and v if $i \geq v$. Therefore, there are two possible classes of online deterministic algorithms for this problem:

1. Algorithms ALG_1 that define m as a function of i, i.e., $m = f(i)$ and $a = \frac{v-f(i)}{T}$.
2. Algorithms ALG_2 that define m as a function of v, i.e., $m = g(v)$ and $a = \frac{v-g(v)}{T}$.

For algorithms from the first class, a can grow arbitrarily large, as m is not a function of v, and the adversary will select v such that it is infinitely greater than $f(i)$. As $a \to \infty$, $\frac{ALG_1(I)}{OPT(I)} \to \infty$; therefore, all algorithms from the first class are not competitive.

For the second class, $m \geq v$, as m is a function of v, and v becomes known for an online algorithm when $i = v$. Therefore $\frac{ALG_2(I)}{OPT(I)} = 2 + s - a(1+s)$, where $a \leq 0$. The minimum competitive ratio of $2 + s$ is obtained at $a = 0$. Thus, an optimal online deterministic algorithm for the single VM migration problem is achieved when $a = 0$, or equivalently $m = v$, and its competitive ratio is $2 + s$. □

4. THE DYNAMIC VM CONSOLIDATION PROBLEM

In this section we analyze a more complex problem of dynamic VM consolidation considering multiple hosts and multiple VMs. For this problem, we define that there are n homogeneous hosts, and the capacity of each host is A_h. Although VMs experience variable workloads, the maximum CPU capacity that can be allocated to a VM is A_v. Therefore, the maximum number of VMs allocated to a host when they demand their maximum CPU capacity is $m = \frac{A_h}{A_v}$. The total number of VMs is nm. VMs can be migrated between hosts using live migration with a migration time t_m. As for the single VM migration problem defined in Section 3, an SLA violation occurs when the total demand for the CPU performance exceeds the available CPU capacity A_h. The cost of power is C_p, and the cost of SLA violation per unit of time is C_v. Without loss of generality, we can define $C_p = 1$ and $C_v = s$, where $s \in \mathbb{R}^+$. This is equivalent to defining $C_p = 1/s$ and $C_v = 1$. We assume that when a host is idle, i.e., there are no allocated VMs, it is switched off and consumes no power,

DOI: 10.1002/cpe.1867

or switched to the sleep mode with negligible power consumption. We call non-idle hosts active. The total cost C is defined as follows:

$$C = \sum_{t=t_0}^{T} \left(C_p \sum_{i=0}^{n} a_{ti} + C_v \sum_{j=0}^{n} v_{tj} \right), \tag{10}$$

where t_0 is the initial time; T is the total time; $a_{ti} \in \{0, 1\}$ indicating whether the host i is active at the time t; $v_{tj} \in \{0, 1\}$ indicating whether the host j is experiencing an SLA violation at the time t. The problem is to determine what time, which VMs and where should be migrated to minimize the total cost C.

4.1. An Optimal Online Deterministic Algorithm

Theorem 3

The upper bound of the competitive ratio of an optimal online deterministic algorithm for the dynamic VM consolidation problem is $\frac{ALG(I)}{OPT(I)} \leq 1 + \frac{ms}{2(m+1)}$.

Proof

Similarly to the single VM migration problem, an optimal online deterministic algorithm for the dynamic VM consolidation problem migrates a VM from a host when an SLA violation occurs at this host. The algorithm always consolidates VMs to the minimum number of hosts, ensuring that the allocation does not cause an SLA violation. The omnipotent malicious adversary generates the CPU demand by VMs in a way that cause as much SLA violation as possible, while keeping as many hosts active (i.e., consuming energy) as possible.

As $mA_v = A_h$, for any $k > m$, $k \in \mathbb{N}$, $kA_v > A_h$. In other words, an SLA violation occurs at a host when at least $m + 1$ VMs are allocated to this host, and these VMs demand their maximum CPU capacity A_v. Therefore, the maximum number of hosts that experience an SLA violation simultaneously n_v is defined as in (11).

$$n_v = \left\lfloor \frac{nm}{m+1} \right\rfloor. \tag{11}$$

In a case of a simultaneous SLA violation at n_v hosts, the number of hosts not experiencing an SLA violation is $n_r = n - n_v$. The strategy of the adversary is to make the online algorithm keep all the hosts active all the time and make n_v hosts experience an SLA violation half of the time. To show how this is achieved, we split the time into periods of length $2t_m$. Then $T - t_0 = 2t_m\tau$, where $\tau \in \mathbb{R}^+$. Each of these periods can be split into two equal parts of length t_m. For these two parts of each period, the adversary acts as follows :

1. During the first t_m, the adversary sets the CPU demand by the VMs in a way to allocate exactly $m + 1$ VMs to n_v hosts by migrating VMs from n_r hosts. As the VM migration time is t_m, the total cost during this period of time is $t_m n C_p$, as all the hosts are active during migrations, and there is no SLA violation.
2. During the next t_m, the adversary sets the CPU demand by the VMs to the maximum causing an SLA violation at n_v hosts. The online algorithm reacts to the SLA violation, and migrates the necessary number of VMs back to n_r hosts. During this period of time, the total cost is $t_m(nC_p + n_v C_v)$, as all the hosts are again active, and n_v hosts are experiencing an SLA violation.

Therefore, the total cost during a time period $2t_m$ is defined as follows:

$$C = 2t_m n C_p + t_m n_v C_v. \tag{12}$$

This leads to the following total cost incurred by an optimal online deterministic algorithm (ALG) for the input I:

$$ALG(I) = \tau t_m (2nC_p + n_v C_v). \tag{13}$$

DOI: 10.1002/cpe.1867

An optimal offline algorithm for this kind of workload will just keep m VMs at each host all the time without any migrations. Thus, the total cost incurred by an optimal offline algorithm is defined as shown in (14).

$$OPT(I) = 2\tau t_m n C_p. \tag{14}$$

Having determined both costs, we can find the competitive ratio of an optimal offline deterministic algorithm (15).

$$\frac{ALG(I)}{OPT(I)} = \frac{\tau t_m(2nC_p + n_v C_v)}{2\tau t_m n C_p} = \frac{2nC_p + n_v C_v}{2nC_p} = 1 + \frac{n_v C_v}{2nC_p}. \tag{15}$$

Via the substitution of $C_p = 1/s$ and $C_v = 1$, we get (16).

$$\frac{ALG(I)}{OPT(I)} = 1 + \frac{n_v s}{2n}. \tag{16}$$

First, we consider the case when $\mod \frac{nm}{m+1} = 0$, and thus $n_v = \frac{nm}{m+1}$. For this case (ALG_1), the competitive ratio is shown in (17).

$$\frac{ALG_1(I)}{OPT(I)} = 1 + \frac{nms}{2n(m+1)} = 1 + \frac{ms}{2(m+1)}. \tag{17}$$

The second case (ALG_2) is when $\mod \frac{nm}{m+1} \neq 0$. Then due to the remainder, n_v is less than in the first case. Therefore, the competitive ratio is defined as in (18).

$$\frac{ALG_2(I)}{OPT(I)} < 1 + \frac{ms}{2(m+1)}. \tag{18}$$

If we combine both cases, the competitive ratio can be defined as in (19), which is an upper bound on the competitive ratio of an optimal online deterministic algorithm for the dynamic VM consolidation problem.

$$\frac{ALG(I)}{OPT(I)} \leq 1 + \frac{ms}{2(m+1)}. \tag{19}$$

□

4.2. *Non-Deterministic Online Algorithms*

It is known that non-deterministic, or randomized, online algorithms typically improve upon the quality of their deterministic counterparts [29]. Therefore, it can be expected that the competitive ratio of online randomized algorithms for the single VM migration problem (Section 3), which falls back to the optimal online deterministic algorithm when $i \geq v$, lies between $\frac{T}{s}$ and $2 + s$. Similarly, it can be expected that the competitive ratio of online randomized algorithms for the dynamic VM consolidation problem should be improved relatively to the upper bound determined in Theorem 3. In competitive analysis, randomized algorithms are analyzed against different types of adversaries than the omnipotent malicious adversary used for deterministic algorithms. For example, one of these adversaries is the oblivious adversary that generates a complete input sequence prior to the beginning of the algorithm execution. It generates an input based on knowledge of probability distributions used by the algorithm.

Another approach to analyzing randomized algorithms is finding the average-case performance of an algorithm based on distributional models of the input. However, in a real world setting, the workload experienced by VMs is more complex and cannot be modeled using simple statistical distributions [24]. For example, it has been shown that web workloads have such properties as correlation between workload attributes, non-stationarity, burstiness, and self-similarity [25]. Job arrival times in Grid and cluster workloads have been identified to exhibit such patterns as pseudo-periodicity, long range dependency, and multifractal scaling [26]. In Section 6, we propose adaptive algorithms that rely on statistical analysis of historical data of the workload. One of the assumptions is that workloads are not completely random, and future events can be predicted based on the past data. However, such algorithms cannot be analyzed using simple distributional or adversary models, such as oblivious adversary, as realistic workloads require more complex modeling, e.g. using Markov chains [30]. We plan to investigate these workload models in future work.

DOI: 10.1002/cpe.1867

5. THE SYSTEM MODEL

In this paper, the targeted system is an IaaS environment, represented by a large-scale data center consisting of N heterogeneous physical nodes. Each node i is characterized by the CPU performance defined in Millions Instructions Per Second (MIPS), amount of RAM and network bandwidth. The servers do not have local disks, the storage is provided as a Network Attached Storage (NAS) to enable live migration of VMs. The type of the environment implies no knowledge of application workloads and time for which VMs are provisioned. Multiple independent users submit requests for provisioning of M heterogeneous VMs characterized by requirements to processing power defined in MIPS, amount of RAM and network bandwidth. The fact that the VMs are managed by independent users implies that the resulting workload created due to combining multiple VMs on a single physical node is mixed. The mixed workload is formed by various types of applications, such as HPC and web-applications, which utilize the resources simultaneously. The users establish SLAs with the resource provider to formalize the QoS delivered. The provider pays a penalty to the users in cases of SLA violations.

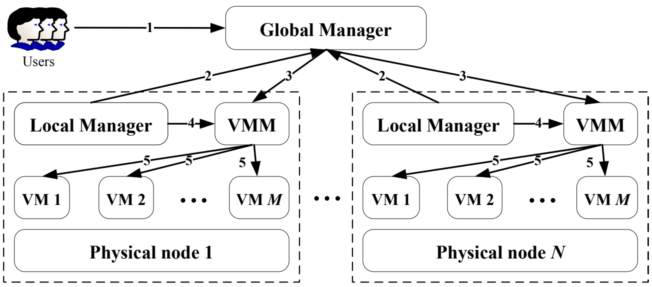


Figure 2. The system model

The software layer of the system is tiered comprising local and global managers (Figure 2). The local managers reside on each node as a module of the VMM. Their objective is the continuous monitoring of the node's CPU utilization, resizing the VMs according to their resource needs, and deciding when and which VMs should to be migrated from the node (4). The global manager resides on the master node and collects information from the local managers to maintain the overall view of the utilization of resources (2). The global manager issues commands for the optimization of the VM placement (3). VMMs perform actual resizing and migration of VMs as well as changes in power modes of the nodes (5).

5.1. *Multi-Core CPU Architectures*

In our model, physical servers are equipped with multi-core CPUs. We model a multi-core CPU with n cores each having m MIPS as a single-core CPU with the total capacity of nm MIPS. This is justified as applications, as well as VMs, are not tied down to processing cores and can be executed on an arbitrary core using a time-shared scheduling algorithm. The only limitation is that the CPU capacity required for a VM must be less or equal to the capacity of a single core. The reason is that if the CPU capacity required for a VM higher than the capacity of a single core, then a VM must be executed on more than one core in parallel. However, we do not assume that VMs can be arbitrarily parallelized, as there is no *a priori* knowledge of the applications running on a VM and automatic parallelization is a complex research problem.

5.2. *Power Model*

Power consumption by computing nodes in data centers is mostly determined by the CPU, memory, disk storage, power supplies and cooling systems [31]. Recent studies [5], [13] have shown that the power consumption by servers can be accurately described by a linear relationship between the power consumption and CPU utilization, even when Dynamic Voltage and Frequency Scaling

Concurrency Computat.: Pract. Exper. (2012)
DOI: 10.1002/cpe.1867

Table I. Power consumption by the selected servers at different load levels in Watts

Server	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HP ProLiant G4	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HP ProLiant G5	93.7	97	101	105	110	116	121	125	129	133	135

(DVFS) is applied. The reason lies in the limited number of states that can be set to the frequency and voltage of a CPU and the fact that voltage and performance scaling is not applied to other system components, such as memory and network interfaces. However, due to the proliferation of multi-core CPUs and virtualization, modern servers are typically equipped with large amounts of memory, which begins to dominate the power consumption by a server [31]. This fact combined with the difficulty of modeling power consumption by modern multi-core CPUs makes building precise analytical models a complex research problem. Therefore, instead of using an analytical model of power consumption by a server, we utilize real data on power consumption provided by the results of the SPECpower benchmark†.

We have selected two server configurations with dual-core CPUs published in February 2011: HP ProLiant ML110 G4 (Intel Xeon 3040, 2 cores × 1860 MHz, 4 GB), and HP ProLiant ML110 G5 (Intel Xeon 3075, (2 cores × 2660 MHz, 4 GB). The configuration and power consumption characteristics of the selected servers are shown in Table I. The reason why we have not chosen servers with more cores is that it is important to simulate a large number of servers to evaluate the effect of VM consolidation. Thus, simulating less powerful CPUs is advantageous, as less workload is required to overload a server. Nevertheless, dual-core CPUs are sufficient to evaluate resource management algorithms designed for multi-core CPU architectures.

5.3. Cost of VM Live Migration

Live migration of VMs allows transferring a VM between physical nodes without suspension and with a short downtime. However, live migration has a negative impact on the performance of applications running in a VM during a migration. Voorsluys et al. have performed an experimental study to investigate the value of this impact and find a way to model it [32]. They have found that performance degradation and downtime depend on the application's behavior, i.e., how many memory pages the application updates during its execution. However, for the class of applications with variable workloads, such as web-applications, the average performance degradation including the downtime can be estimated as approximately 10% of the CPU utilization. Moreover, in our simulations we model that the same amount of CPU capacity is allocated to a VM on the destination node during the course of migration. This means that each migration may cause some SLA violation; therefore, it is crucial to minimize the number of VM migrations. The length of a live migration depends on the total amount of memory used by the VM and available network bandwidth. This is justified as to enable live migration, the images and data of VMs must be stored on a Network Attached Storage (NAS); and therefore, copying the VM's storage is not required. Thus, for our experiments we define the migration time and performance degradation experienced by a VM j as shown in (20).

$$T_{m_j} = \frac{M_j}{B_j}, \qquad U_{d_j} = 0.1 \cdot \int_{t_0}^{t_0+T_{m_j}} u_j(t)\, dt, \tag{20}$$

where U_{d_j} is the total performance degradation by VM j, t_0 is the time when the migration starts, T_{m_j} is the time taken to complete the migration, $u_j(t)$ is the CPU utilization by VM j, M_j is the amount of memory used by VM j, and B_j is the available network bandwidth.

†The SPECpower benchmark. http://www.spec.org/power_ssj2008/

Prepared using cpeauth.cls
DOI: 10.1002/cpe.1867

5.4. *SLA Violation Metrics*

Meeting QoS requirements is extremely important for Cloud computing environments. QoS requirements are commonly formalized in the form of SLAs, which can be determined in terms of such characteristics as minimum throughput or maximum response time delivered by the deployed system. As these characteristics can vary for different applications, it is necessary to define a workload independent metric that can be used to evaluate the SLA delivered to any VM deployed in an IaaS. For our experiments, we define that the SLAs are delivered when 100% of the performance requested by applications inside a VM is provided at any time bounded only by the parameters of the VM. We propose two metrics for measuring the level of SLA violations in an IaaS environment: (1) the percentage of time, during which active hosts have experienced the CPU utilization of 100%, SLA violation Time per Active Host (SLATAH); and (2) the overall performance degradation by VMs due to migrations, Performance Degradation due to Migrations (PDM) (21). The reasoning behind the SLATAH is the observation that if a host serving applications is experiencing 100% utilization, the performance of the applications is bounded by the host capacity, therefore, VMs are not being provided with the required performance level.

$$SLATAH = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{s_i}}{T_{a_i}}, \qquad PDM = \frac{1}{M}\sum_{j=1}^{M}\frac{C_{d_j}}{C_{r_j}}, \tag{21}$$

where N is the number of hosts; T_{s_i} is the total time during which the host i has experienced the utilization of 100% leading to an SLA violation; T_{a_i} is the total of the host i being in the active state (serving VMs); M is the number of VMs; C_{d_j} is the estimate of the performance degradation of the VM j caused by migrations; C_{r_j} is the total CPU capacity requested by the VM j during its lifetime. In our experiments, we estimate C_{d_j} as 10% of the CPU utilization in MIPS during all migrations of the VM j. Both the SLATAH and PDM metrics independently and with equal importance characterize the level of SLA violations by the infrastructure, therefore, we propose a combined metric that encompasses both performance degradation due to host overloading and due to VM migrations. We denote the combined metric SLA Violation (SLAV), which is calculated as shown in (22).

$$SLAV = SLATAH \cdot PDM. \tag{22}$$

6. ADAPTIVE HEURISTICS FOR DYNAMIC VM CONSOLIDATION

According to the analysis presented in Sections 3 and 4, in this section we propose several heuristics for dynamic consolidation of VMs based on an analysis of historical data of the resource usage by VMs. We split the problem of dynamic VM consolidation into four parts: (1) determining when a host is considered as being overloaded requiring migration of one or more VMs from this host; (2) determining when a host is considered as being underloaded leading to a decision to migrate all VMs from this host and switch the host to the sleep mode; (3) selection of VMs that should be migrated from an overloaded host; and (4) finding a new placement of the VMs selected for migration from the overloaded and underloaded hosts. We discuss the defined subproblems in the following sections.

The general algorithm of VM placement optimization is shown in Algorithm 1. First, the algorithm looks through the list of hosts and by applying the overloading detection algorithm checks whether a host is overloaded. If the host is overloaded, the algorithm applies the VM selection policy to select VMs that need to be migrated from the host. Once the list of VMs to be migrated from the overloaded hosts is built, the VM placement algorithm is invoked to find a new placement for the VMs to be migrated. The second phase of the algorithm is finding underloaded hosts and a placement of the VMs from these hosts. The algorithm returns the combined migration map that contains the information on the new VM placement of the VM selected to be migrated from both overloaded and underloaded hosts. The complexity of the algorithm is $2N$, where N is the number of hosts.

Concurrency Computat.: Pract. Exper. (2012)
DOI: 10.1002/cpe.1867

Algorithm 1: VM placement Optimization

```
Input: hostList  Output: migrationMap
foreach host in hostList do
    if isHostOverloaded (host) then
        vmsToMigrate.add(getVmsToMigrateFromOverloadedHost (host)
migrationMap.add(getNewVmPlacement (vmsToMigrate))
vmsToMigrate.clear()
foreach host in hostList do
    if isHostUnderloaded (host) then
        vmsToMigrate.add(host.getVmList ()
        migrationMap.add(getNewVmPlacement (vmsToMigrate))
return migrationMap
```

6.1. Host Overloading Detection

6.1.1. An Adaptive Utilization Threshold: Median Absolute Deviation. In our previous work we have proposed a heuristic for deciding the time to migrate VMs from a host based on utilization thresholds [22]. It is based on the idea of setting upper and lower utilization thresholds for hosts and keeping the total utilization of the CPU by all the VMs between these thresholds. If the CPU utilization of a host falls below the lower threshold, all VMs have to be migrated from this host and the host has to be switched to the sleep mode in order to eliminate the idle power consumption. If the utilization exceeds the upper threshold, some VMs have to be migrated from the host to reduce the utilization in order to prevent a potential SLA violation.

However, fixed values of utilization thresholds are unsuitable for an environment with dynamic and unpredictable workloads, in which different types of applications can share a physical resource. The system should be able to automatically adjust its behavior depending on the workload patterns exhibited by the applications. Therefore, we propose **novel techniques** for the auto-adjustment of the utilization thresholds based on a statistical analysis of historical data collected during the lifetime of VMs. We apply robust methods that are more effective than classical methods for data containing outliers or coming from non-normal distributions. The main idea of the proposed adaptive-threshold algorithms is to adjust the value of the upper utilization threshold depending on the strength of the deviation of the CPU utilization. The higher the deviation, the lower the value of the upper utilization threshold, as the higher the deviation, the more likely that the CPU utilization will reach 100% and cause an SLA violation.

Robust statistics provides an alternative approach to classical statistical methods [33]. The motivation is to produce estimators that are not unduly affected by small departures from model assumptions. The Median Absolute Deviation (MAD) is a measure of statistical dispersion. It is a more robust estimator of scale than the sample variance or standard deviation, as it behaves better with distributions without a mean or variance, such as the Cauchy distribution. The MAD is a robust statistic, being more resilient to outliers in a data set than the standard deviation. In the standard deviation, the distances from the mean are squared, so on average, large deviations are weighted more heavily, and thus outliers can heavily influence it. In the MAD, the magnitude of the distances of a small number of outliers is irrelevant.

For a univariate data set $X_1, X_2, ..., X_n$, the MAD is defined as the median of the absolute deviations from the data's median:

$$MAD = median_i(|X_i - median_j(X_j)|), \tag{23}$$

that is, starting with the residuals (deviations) from the data's median, the MAD is the median of their absolute values. We define the upper utilization threshold (T_u) as shown in (24).

$$T_u = 1 - s \cdot MAD, \tag{24}$$

Concurrency Computat.: Pract. Exper. (2012)
DOI: 10.1002/cpe.1867

where $s \in \mathbb{R}^+$ is a parameter of the method that defines how aggressively the system consolidates VMs. In other words, the parameter s allows the adjustment of the safety of the method, the lower s, the less the energy consumption, but the higher the level of SLA violations caused by the consolidation.

6.1.2. An Adaptive Utilization Threshold: Interquartile Range. In this section, we propose the second method for setting an adaptive upper utilization threshold based on another robust statistic. In descriptive statistics, the interquartile range (IQR), also called the midspread or middle fifty, is a measure of statistical dispersion, being equal to the difference between the third and first quartiles: $IQR = Q_3 - Q_1$. Unlike the (total) range, the interquartile range is a robust statistic, having a breakdown point of 25%, and is thus often preferred to the total range. For a symmetric distribution (so the median equals the midhinge, the average of the first and third quartiles), half the IQR equals the MAD. Using IQR, similarly to (24) we define the upper utilization threshold as shown in (25).

$$T_u = 1 - s \cdot IQR, \tag{25}$$

where $s \in \mathbb{R}^+$ is a parameter of the method defining the safety of the method similarly to the parameter s of the method proposed in Section 6.1.1.

6.1.3. Local Regression. We base our next algorithm on the Loess method (from the German *löss* – short for *local regression*) proposed by Cleveland [34]. The main idea of the method of local regression is fitting simple models to localized subsets of data to build up a curve that approximates the original data. The observations (x_i, y_i) are assigned neighborhood weights using the *tricube weight function* shown in (26).

$$T(u) = \begin{cases} (1 - |u|^3)^3 & \text{if } |u| < 1, \\ 0 & \text{otherwise}, \end{cases} \tag{26}$$

Let $\Delta_i(x) = |x_i - x|$ be the distance from x to x_i, and let $\Delta_{(i)}(x)$ be these distances ordered from smallest to largest. Then the neighborhood weight for the observation (x_i, y_i) is defined by the function $w_i(x)$ (27).

$$w_i(x) = T\left(\frac{\Delta_i(x)}{\Delta_{(q)}(x)}\right), \tag{27}$$

for x_i such that $\Delta_i(x) < \Delta_{(q)}(x)$, where q is the number of observations in the subset of data localized around x. The size of the subset is defined by a parameter of the method called the *bandwidth*. For example, if the degree of the polynomial fitted by the method is 1, then the parametric family of functions is $y = a + bx$. The line is fitted to the data using the weighted least-squares method with weight $w_i(x)$ at (x_i, y_i). The values of a and b are found by minimizing the function shown in (28).

$$\sum_{i=1}^{n} w_i(x)(y_i - a - bx_i)^2. \tag{28}$$

We utilize this approach to fit a trend polynomial to the last k observations of the CPU utilization, where $k = \lceil q/2 \rceil$. We fit a polynomial for a single point, the last observation of the CPU utilization, the right boundary x_k of the data set. The problem of the boundary region is well known as leading to a high bias [35]. According to Cleveland [36], fitted polynomials of degree 1 typically distort peaks in the interior of the configuration of observations, whereas polynomials of degree 2 remove the distortion but result in higher biases at boundaries. Therefore, for our problem we have chosen polynomials of degree 1 to reduce the bias at the boundary.

Let x_k be the last observation, and x_1 be the k^{th} observation from the right boundary. For our case, we let x_i satisfy $x_1 \leq x_i \leq x_k$, then $\Delta_i(x_k) = x_k - x_i$, and $0 \leq \frac{\Delta_i(x_k)}{\Delta_1(x_k)} \leq 1$. Therefore, the tricube weight function can be simplified as $T^*(u) = (1 - u^3)^3$ for $0 \leq u \leq 1$, and the weight function is the following:

Concurrency Computat.: Pract. Exper. (2012)
DOI: 10.1002/cpe.1867

$$w_i(x) = T^* \left(\frac{\Delta_i(x_k)}{\Delta_1(x_k)} \right) = \left(1 - \left(\frac{x_k - x_i}{x_k - x_1} \right)^3 \right)^3. \tag{29}$$

In our algorithm (LR), using the described method derived from Loess, for each new observation we find a new trend line $\widehat{g}(x) = \widehat{a} + \widehat{b}x$. This trend line is used to estimate the next observation $\widehat{g}(x_{k+1})$. The algorithm decides that the host is considered overloaded and some VMs should be migrated from it if the inequalities (30) are satisfied.

$$s \cdot \widehat{g}(x_{k+1}) \geq 1, \qquad x_{k+1} - x_k \leq t_m, \tag{30}$$

where $s \in \mathbb{R}^+$ is the safety parameter; and t_m is the maximum time required for a migration of any of the VMs allocated to the host. We denote this algorithm Local Regression (LR).

6.1.4. Robust Local Regression. The version of Loess described in Section 6.1.3 is vulnerable to outliers that can be caused by leptokurtic or heavy-tailed distributions. To make Loess robust, Cleveland has proposed the addition of the robust estimation method *bisquare* to the least-squares method for fitting a parametric family [37]. This modification transforms Loess into an iterative method. The initial fit is carried out with weights defined using the tricube weight function. The fit is evaluated at the x_i to get the fitted values $\widehat{y_i}$, and the residuals $\widehat{\epsilon_i} = y_i - \widehat{y_i}$. At the next step, each observation (x_i, y_i) is assigned an additional *robustness weight* r_i, whose value depends on the magnitude of $\widehat{\epsilon_i}$. Each observation is assigned the weight $r_i w_i(x)$, where r_i is defined as in (31).

$$r_i = B\left(\frac{\widehat{\epsilon_i}}{6s} \right), \tag{31}$$

where $B(u)$ is the *bisquare weight function* (32), and s is the MAD for the least-squares fit or any subsequent weighted fit (33).

$$B(u) = \begin{cases} (1-u^2)^2 & \text{if } |u| < 1, \\ 0 & \text{otherwise,} \end{cases} \tag{32}$$

$$s = median|\widehat{\epsilon_i}|. \tag{33}$$

Using the estimated trend line, we apply the same method described in Section 6.1.3 to estimate the next observation and decide that the host is overloaded if the inequalities (30) are satisfied. We denote this overloading detection algorithm Local Regression Robust (LRR).

6.2. VM Selection

Once it has been decided that a host is overloaded, the next step is to select particular VMs to migrate from this host. In this section we propose three policies for VM selection. The described policies are applied iteratively. After a selection of a VM to migrate, the host is checked again for being overloaded. If it is still considered as being overloaded, the VM selection policy is applied again to select another VM to migrate from the host. This is repeated until the host is considered as being not overloaded.

6.2.1. The Minimum Migration Time Policy. The Minimum Migration Time (MMT) policy migrates a VM v that requires the minimum time to complete a migration relatively to the other VMs allocated to the host. The migration time is estimated as the amount of RAM utilized by the VM divided by the spare network bandwidth available for the host j. Let V_j be a set of VMs currently allocated to the host j. The MMT policy finds a VM v that satisfies conditions formalized in (34).

$$v \in V_j | \forall a \in V_j, \frac{RAM_u(v)}{NET_j} \leq \frac{RAM_u(a)}{NET_j}, \tag{34}$$

where $RAM_u(a)$ is the amount of RAM currently utilized by the VM a; and NET_j is the spare network bandwidth available for the host j.

Concurrency Computat.: Pract. Exper. (2012)
DOI: 10.1002/cpe.1867

6.2.2. The Random Choice Policy. The Random Choice (RC) policy selects a VM to be migrated according to a uniformly distributed discrete random variable $X \stackrel{d}{=} U(0, |V_j|)$, whose values index a set of VMs V_j allocated to a host j.

6.2.3. The Maximum Correlation Policy The Maximum Correlation (MC) policy is based on the idea proposed by Verma et al. [17]. The idea is that the higher the correlation between the resource usage by applications running on an oversubscribed server, the higher the probability of the server overloading. According to this idea, we select those VMs to be migrated that have the highest correlation of the CPU utilization with other VMs. To estimate the correlation between CPU utilizations by VMs, we apply the *multiple correlation coefficient* [38]. It is used in multiple regression analysis to assess the quality of the prediction of the dependent variable. The multiple correlation coefficient corresponds to the squared correlation between the predicted and the actual values of the dependent variable. It can also be interpreted as the proportion of the variance of the dependent variable explained by the independent variables.

Let $X_1, X_2, ..., X_n$ be n random variables representing the CPU utilizations of n VMs allocated to a host. Let Y represent one of the VMs that is currently considered for being migrated. Then $n-1$ random variables are independent, and 1 variable Y is dependent. The objective is to evaluate the strength of the correlation between Y and $n-1$ remaining random variables. We denote by $\mathbf{X}$ the $(n-1) \times n$ augmented matrix containing the observed values of the $n-1$ independent random variables, and by $\mathbf{y}$ the $(n-1) \times 1$ vector of observations for the dependent variable Y (35). The matrix $\mathbf{X}$ is called augmented because the first column is composed only of 1.

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & \dots & x_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1,1} & \dots & x_{n-1,n-1} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \tag{35}$$

A vector of predicted values of the dependent random variable $\widehat{Y}$ is denoted by $\widehat{\mathbf{y}}$ and is obtained as shown in (36).

$$\widehat{\mathbf{y}} = \mathbf{X}\mathbf{b} \qquad \mathbf{b} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}. \tag{36}$$

Having found a vector of predicted values, we now can compute the multiple correlation coefficient $R^2_{Y,1,...,n-1}$, which is equal to the squared coefficient of correlation between the observed values $\mathbf{y}$ of the dependent variable Y and the predicted values $\widehat{\mathbf{y}}$ (37).

$$R^2_{Y,X_1,...,X_{n-1}} = \frac{\sum_{i=1}^{n}(\mathbf{y}_i - m_Y)^2(\widehat{\mathbf{y}}_i - m_{\widehat{Y}})^2}{\sum_{i=1}^{n}(\mathbf{y}_i - m_Y)^2 \sum_{i=1}^{n}(\widehat{\mathbf{y}}_i - m_{\widehat{Y}})^2}, \tag{37}$$

where m_Y and $m_{\widehat{Y}}$ are the sample means of Y and $\widehat{Y}$ respectively. We find the multiple correlation coefficient for each X_i, which is denoted as $R^2_{X_i,X_1,...,X_{i-1},X_{i+1},...,X_n}$. The MC policy finds a VM v that satisfies the conditions defined in (38).

$$v \in V_j | \forall a \in V_j, R^2_{X_v,X_1,...,X_{v-1},X_{v+1},...,X_n} \geq R^2_{X_v,X_1,...,X_{a-1},X_{a+1},...,X_n}. \tag{38}$$

6.3. VM Placement

The VM placement can be seen as a bin packing problem with variable bin sizes and prices, where bins represent the physical nodes; items are the VMs that have to be allocated; bin sizes are the available CPU capacities of the nodes; and prices correspond to the power consumption by the nodes. As the bin packing problem is NP-hard, to solve it we apply a modification of the Best Fit Decreasing (BFD) algorithm that is shown to use no more than $11/9 \cdot OPT + 1$ bins (where OPT is the number of bins provided by the optimal solution) [39]. In the modification of the BFD algorithm denoted Power Aware Best Fit Decreasing (PABFD) proposed in our previous work [22] we sort all the VMs in the decreasing order of their current CPU utilizations and allocate each VM to a host that provides the least increase of the power consumption caused by the allocation. This allows the

DOI: 10.1002/cpe.1867

leveraging the nodes' heterogeneity by choosing the most power-efficient ones first. The pseudo-code for the algorithm is presented in Algorithm 2. The complexity of the algorithm is nm, where n is the number of nodes and m is the number of VMs that have to be allocated.

Algorithm 2: Power Aware Best Fit Decreasing (PABFD)

```
Input: hostList, vmList   Output: allocation of VMs
vmList.sortDecreasingUtilization()
foreach vm in vmList do
    minPower ← MAX
    allocatedHost ← NULL
    foreach host in hostList do
        if host has enough resources for vm then
            power ← estimatePower(host, vm)
            if power < minPower then
                allocatedHost ← host
                minPower ← power
    if allocatedHost ≠ NULL then
        allocation.add(vm, allocatedHost)
return allocation
```

6.4. Host Underloading Detection

For determining underloaded hosts we propose a simple approach. First, all the overloaded hosts are found using the selected overloading detection algorithm, and the VMs selected for migration are allocated to the destination hosts. Then, the system finds the host with the minimum utilization compared to the other hosts, and tries to place the VMs from this host on other hosts keeping them not overloaded. If this can be accomplished, the VMs are set for migration to the determined target hosts, and the source host is switched to the sleep mode once all the migrations have been completed. If all the VMs from the source host cannot be placed on other hosts, the host is kept active. This process is iteratively repeated for all hosts that have not been considered as being overloaded.

7. PERFORMANCE EVALUATION

7.1. Experiment Setup

As the targeted system is an IaaS, a Cloud computing environment that is supposed to create a view of infinite computing resources to users, it is essential to evaluate the proposed resource allocation algorithms on a large-scale virtualized data center infrastructure. However, it is extremely difficult to conduct repeatable large-scale experiments on a real infrastructure, which is required to evaluate and compare the proposed algorithms. Therefore, to ensure the repeatability of experiments, simulations have been chosen as a way to evaluate the performance of the proposed heuristics.

The CloudSim toolkit [40] has been chosen as a simulation platform, as it is a modern simulation framework aimed at Cloud computing environments. In contrast to alternative simulation toolkits (e.g. SimGrid, GangSim), it allows the modeling of virtualized environments, supporting on-demand resource provisioning, and their management. It has been extended to enable energy-aware simulations, as the core framework does not provide this capability. Apart from the energy consumption modeling and accounting, the ability to simulate service applications with dynamic workloads has been incorporated. The implemented extensions have been included in the 2.0 version of the CloudSim toolkit.

We have simulated a data center that comprises 800 heterogeneous physical nodes, half of which are HP ProLiant ML110 G4 servers, and the other half consists of HP ProLiant ML110 G5 servers. The characteristics of the servers and data on their power consumption are given in Section 5.2.

DOI: 10.1002/cpe.1867

Table II. Workload data characteristics (CPU utilization)

Date	Number of VMs	Mean	St. dev.	Quartile 1	Median	Quartile 3
03/03/2011	1052	12.31%	17.09%	2%	6%	15%
06/03/2011	898	11.44%	16.83%	2%	5%	13%
09/03/2011	1061	10.70%	15.57%	2%	4%	13%
22/03/2011	1516	9.26%	12.78%	2%	5%	12%
25/03/2011	1078	10.56%	14.14%	2%	6%	14%
03/04/2011	1463	12.39%	16.55%	2%	6%	17%
09/04/2011	1358	11.12%	15.09%	2%	6%	15%
11/04/2011	1233	11.56%	15.07%	2%	6%	16%
12/04/2011	1054	11.54%	15.15%	2%	6%	16%
20/04/2011	1033	10.43%	15.21%	2%	4%	12%

The frequency of the servers' CPUs are mapped onto MIPS ratings: 1860 MIPS each core of the HP ProLiant ML110 G5 server, and 2660 MIPS each core of the HP ProLiant ML110 G5 server. Each server is modeled to have 1 GB/s network bandwidth. The characteristics of the VM types correspond to Amazon EC2 instance types[‡] with the only exception that all the VMs are single-core, which is explained by the fact that the workload data used for the simulations come from single-core VMs (Section 7.3). For the same reason the amount of RAM is divided by the number of cores for each VM type: High-CPU Medium Instance (2500 MIPS, 0.85 GB); Extra Large Instance (2000 MIPS, 3.75 GB); Small Instance (1000 MIPS, 1.7 GB); and Micro Instance (500 MIPS, 613 MB). Initially the VMs are allocated according to the resource requirements defined by the VM types. However, during the lifetime, VMs utilize less resources according to the workload data, creating opportunities for dynamic consolidation.

7.2. *Performance Metrics*

In order to compare the efficiency of the algorithms we use several metrics to evaluate their performance. One of the metrics is the total energy consumption by the physical servers of a data center caused by the application workloads. Energy consumption is calculated according to the model defined in Section 5.2. Metrics used to evaluate the level of SLA violations caused by the system are SLAV, SLATAH and PDM defined in Section 5.4. Another metric is the number of VM migrations initiated by the VM manager during the adaptation of the VM placement. The main metrics are energy consumption by physical nodes and SLAV, however, these metrics are typically negatively correlated as energy can usually be decreased by the cost of the increased level of SLA violations. The objective of the resource management system is to minimize both energy and SLA violations. Therefore, we propose a combined metric that captures both energy consumption and the level of SLA violations, which we denote Energy and SLA Violations (ESV) (39).

$$ESV = E \cdot SLAV. \tag{39}$$

7.3. *Workload Data*

To make a simulation-based evaluation applicable, it is important to conduct experiments using workload traces from a real system. For our experiments we have used data provided as a part of the CoMon project, a monitoring infrastructure for PlanetLab [41]. We have used the data on the CPU utilization by more than a thousand VMs from servers located at more than 500 places around the world. The interval of utilization measurements is 5 minutes. We have randomly chosen 10 days from the workload traces collected during March and April 2011. The characteristics of the data for each day are shown in Table II. The data confirm the statement made in the beginning: the average CPU utilization is far below 50%. During the simulations, each VM is randomly assigned a workload trace from one of the VMs from the corresponding day. In the simulations we do not limit

[‡]Amazon EC2 Instance Types. http://aws.amazon.com/ec2/instance-types/

DOI: 10.1002/cpe.1867

Table III. Comparison of VM selection policies using paired T-tests

Policy 1 (ESV $\times 10^{-3}$)	**Policy 2 (ESV $\times 10^{-3}$)**	**Difference ($\times 10^{-3}$)**	**P-value**
RS (4.03)	MC (3.83)	0.196 (0.134, 0.258)	P-value < 0.001
RS (4.03)	MMT (3.23)	0.799 (0.733, 0.865)	P-value < 0.001
MC (3.83)	MMT (3.23)	0.603 (0.533, 0.673)	P-value < 0.001

Table IV. Tukey's pairwise comparisons using the transformed ESV. Values that do not share a letter are significantly different.

Policy	**SQRT(ESV) ($\times 10^{-2}$)**	**95% CI**	**Grouping**
THR-MMT-0.8	6.34	(5.70, 6.98)	A
IQR-MMT-1.5	6.16	(5.44, 6.87)	A
MAD-MMT-2.5	6.13	(5.49, 6.77)	A
LRR-MMT-1.2	4.82	(4.22, 5.41)	B
LR-MMT-1.2	4.37	(3.83, 4.91)	B

the VM consolidation by the memory bounds, as this would constrain the consolidation, whereas the objective of the experiments is to stress the consolidation algorithms.

7.4. Simulation Results and Analysis

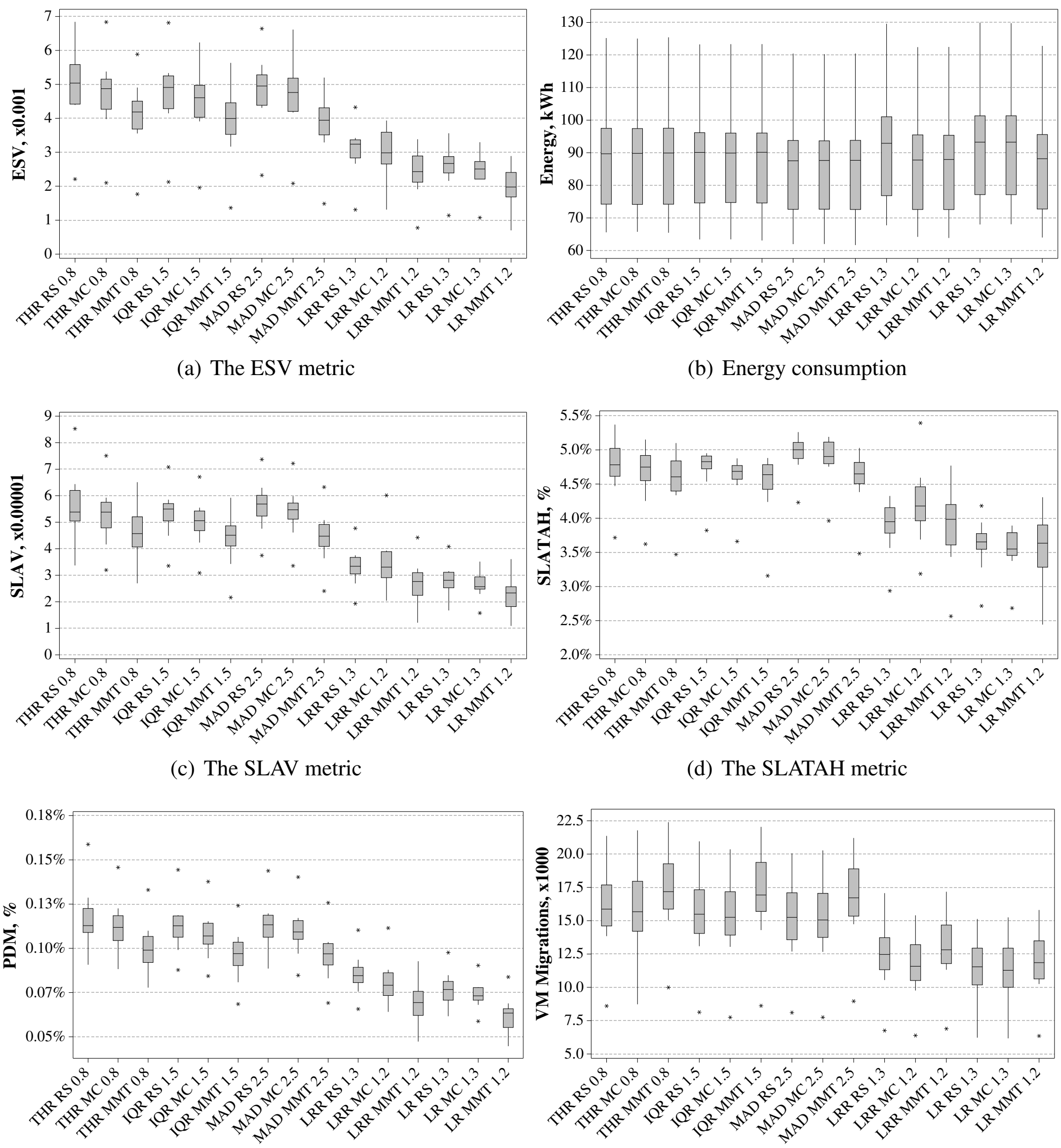
Using the workload data described in Section 7.3, we have simulated all combinations of the five proposed host overloading detection algorithms (THR, IQR, MAD, LR, and LRR) and three VM selection algorithms (MMT, RS, and MC). Moreover, for each overloading detection algorithm we have varied the parameters as follows: for THR from 0.6 to 1.0 increasing by 0.1; for IQR and MAD from 0.5 to 3.0 increasing by 0.5; for LR and LRR from 1.0 to 1.4 increasing by 0.1. These variations have resulted in 81 combinations of the algorithms and parameters. According to Ryan-Joiner's normality test, the values of the ESV metric produced by the algorithm combinations do not follow a normal distribution with the P-value < 0.01. Therefore, we have used the median values of the ESV metric to compare algorithm combinations and select the parameter of each algorithm combination that minimizes the median ESV metric calculated over 10 days of the workload traces. The results produced by the selected algorithms are shown in Figure 3.

According to Ryan-Joiner's normality test, the values of the ESV metric produced by the selected algorithm combinations follow a normal distribution with the P-value > 0.1. We have conducted three paired T-tests to determine the VM selection policy that minimizes the ESV metric across all algorithm combinations (Table III). The T-tests have shown that the usage of the MMT policy leads to a statistically significantly lower value of the ESV metric with the P-value < 0.001. Further, we analyze the combinations of the overloading detection algorithms with the MMT policy.

To meet the assumptions of the ANOVA model, we have transformed the values of the ESV metric for the algorithm combinations with the MMT policy using the square root function. The standardized residuals from the transformed data pass Ryan-Joiner's test with the P-value $>$ 0.1, justifying the assumption that the sample comes from a normal distribution. A plot of the standardized residuals against the fitted values has shown that the assumption of equal variances is met. Having the assumptions of the model met, we have applied the F-test to check whether there is a statistically significant difference between the results produced by the combinations of the overloading detection algorithms with the MMT policy with the selected parameters. The test has shown that there is a statistically significant difference between the results with the P-value < 0.001. Tukey's pairwise comparisons are summarized in Table IV.

According to results of Tukey's pairwise comparisons, we conclude that there is no statistically significant difference between the THR-MMT-0.8, IQR-MMT-1.5 and MAD-MMT-2.5 algorithms (group A), and between the LRR-MMT-1.2 and LR-MMT-1.2 algorithms (group B). However, there is a statistically significant difference between the local regression based algorithms and the other algorithms. Nevertheless, a paired T-test for a comparison of the means of the ESV metric produced

DOI: 10.1002/cpe.1867



(a) The ESV metric

(b) Energy consumption

(c) The SLAV metric

(d) The SLATAH metric

(e) The PDM metric

(f) VM migrations

Figure 3. Algorithm combinations with best parameters by the ESV metric

by LRR-MMT-1.2 and LR-MMT-1.2 shows that there is a statistically significant difference with the P-value < 0.001. The mean difference is 4.21×10^{-4} with a 95% CI: (3.23×10^{-4}, 5.19×10^{-4}). As a paired T-test provides more precise results than Tukey's pairwise comparisons, we can conclude that LR-MMT-1.2 provides the best results compared to all the other combinations of algorithms in regard to the ESV metric. Moreover, the trade-off between energy consumption and SLA violations can be adjusted by varying the safety parameter of the LR algorithm. The results of the combinations of each overloading detection algorithm with the best parameters and the MMT policy, along with the benchmark algorithms are shown in Table V. The benchmark policies include Non Power Aware (NPA), DVFS and the optimal online deterministic algorithm combined with the MMT policy. The NPA policy makes all the hosts consume the maximum power all the time. The optimal online deterministic algorithm corresponds to the fixed threshold algorithm with the threshold set to 100%, therefore, it is named THR-1.0.

Concurrency Computat.: Pract. Exper. (2012)
DOI: 10.1002/cpe.1867

Table V. Simulation results of the best algorithm combinations and benchmark algorithms (median values)

Policy	**ESV** ($\times 10^{-3}$)	**Energy** (kWh)	**SLAV** ($\times 10^{-5}$)	**SLATAH**	**PDM**	**VM migr.** ($\times 10^{3}$)
NPA	0	2419.2	0	0%	0%	0
DVFS	0	613.6	0	0%	0%	0
THR-MMT-1.0	20.12	75.36	25.78	24.97%	0.10%	13.64
THR-MMT-0.8	4.19	89.92	4.57	4.61%	0.10%	17.18
IQR-MMT-1.5	4.00	90.13	4.51	4.64%	0.10%	16.93
MAD-MMT-2.5	3.94	87.67	4.48	4.65%	0.10%	16.72
LRR-MMT-1.2	2.43	87.93	2.77	3.98%	0.07%	12.82
LR-MMT-1.2	1.98	88.17	2.33	3.63%	0.06%	11.85

From the observed simulation results, we can make several conclusions: (1) dynamic VM consolidation algorithms significantly outperforms static allocation policies, such as NPA and DVFS; (2) heuristic-based dynamic VM consolidation algorithms substantially outperform the optimal online deterministic algorithm (THR-1.0) due to a vastly reduced level of SLA violations; (3) the MMT policy produces better results compared to the MC and RS policies, meaning that the minimization of the VM migration time is more important than the minimization of the correlation between VMs allocated to a host; (4) dynamic VM consolidation algorithms based on local regression outperform the threshold-based and adaptive-threshold based algorithms due to better predictions of host overloading, and therefore decreased SLA violations due to host overloading (SLATAH) and the number of VM migrations; and (5) the algorithm based on local regression produces better results than its robust modification, which can be explained by the fact that for the simulated workload it is more important to react to load spikes instead of smoothing out such outlying observations.

The mean value of the sample means of the time before a host is switched to the sleep mode for the LR-MMT-1.2 algorithm combination is 1933 seconds with the 95% CI: (1740, 2127). This means that on average a host is switched to the sleep mode after approximately 32 minutes of activity. This value is effective for real-world systems, as modern servers allow low-latency transitions to the sleep mode consuming low power. Meisner et al. [42] have shown that a typical blade server consuming 450 W in the fully utilized state consumes approximately 10.4 W in the sleep mode, while the transition delay is 300 ms. The mean number of host transitions to the sleep mode for our experiment setup (the total number of hosts is 800) per day is 1272 with 95% CI: (1211, 1333). The mean value of the sample means of the time before a VM is migrated from a host for the same algorithm combination is 15.3 seconds with the 95% CI: (15.2, 15.4). The mean value of the sample means of the execution time of the LR-MMT-1.2 algorithm on a server with an Intel Xeon 3060 (2.40 GHz) processor and 2 GB of RAM is 0.20 ms with the 95% CI: (0.15, 0.25).

8. CONCLUDING REMARKS AND FUTURE DIRECTIONS

To maximize their ROI Cloud providers have to apply energy-efficient resource management strategies, such as dynamic consolidation of VMs and switching idle servers to power-saving modes. However, such consolidation is not trivial, as it can result in violations of the SLA negotiated with customers. In this paper we have conducted competitive analysis of the single VM migration and dynamic VM consolidation problems. We have found and proved competitive ratios for optimal online deterministic algorithms for these problems. We have concluded that it is necessary to develop randomized or adaptive algorithms to improve upon the performance of optimal deterministic algorithms. According to the results of the analysis, we have proposed novel adaptive heuristics that are based on an analysis of historical data on the resource usage for energy and performance efficient dynamic consolidation of VMs.

We have evaluated the proposed algorithms through extensive simulations on a large-scale experiment setup using workload traces from more than a thousand PlanetLab VMs. The results of

Concurrency Computat.: Pract. Exper. (2012)
DOI: 10.1002/cpe.1867

the experiments have shown that the proposed local regression based algorithm combined with the MMT VM selection policy significantly outperforms other dynamic VM consolidation algorithms in regard to the ESV metric due to a substantially reduced level of SLA violations and the number of VM migrations. In order to evaluate the proposed system in a real Cloud infrastructure, we plan to implement it by extending a real-world Cloud platform, such as OpenStack[§]. Another direction for future research is the investigation of more complex workload models, e.g. models based on Markov chains, and development of algorithms that will leverage these workload models. Besides the reduction in infrastructure and on-going operating costs, this work also has social significance as it decreases carbon dioxide footprints and energy consumption by modern IT infrastructures.

ACKNOWLEDGEMENTS

This paper is a substantially extended version of the paper published in MGC 2010 [43]. We thank Yoganathan Sivaram (Melbourne University) for his constructive suggestions on enhancing the quality of the paper.

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* 2009; **25**(6):599–616.
2. ASHRAE Technical Committee 99. Datacom equipment power trends and cooling applications 2005.
3. Belady C. In the data center, power and cooling costs more than the it equipment it supports 2007. URL http://www.electronics-cooling.com/articles/2007/feb/a3/.
4. Barroso LA, Holzle U. The case for energy-proportional computing. *Computer* 2007; **40**(12):33–37.
5. Fan X, Weber WD, Barroso LA. Power provisioning for a warehouse-sized computer. *Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA 2007)*, ACM New York, NY, USA, 2007; 13–23.
6. de Assunao MD, Gelas JP, Lefevre L, Orgerie AC. The Green Grid'5000: Instrumenting and using a Grid with energy sensors. *Proceedings of the 5th International Workshop on Distributed Cooperative Laboratories: Instrumenting the Grid (INGRID 2010)*, Poznan, Poland, 2010.
7. Ranganathan P, Leech P, Irwin D, Chase J. Ensemble-level power management for dense blade servers. *Proceedings of the 33rd International Symposium on Computer Architecture (ISCA 2006)*, Boston, MA, USA, 2006; 66–77.
8. The green grid consortium 2011. URL http://www.thegreengrid.org.
9. Xen and the art of virtualization. *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP 2003)*, Bolton Landing, NY, USA.
10. Clark C, Fraser K, Hand S, Hansen JG, Jul E, Limpach C, Pratt I, Warfield A. Live migration of virtual machines. *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI 2005)*, USENIX, Boston, MA, USA, 2005.
11. Ben-David S, Borodin A, Karp R, Tardos G, Wigderson A. On the power of randomization in on-line algorithms. *Algorithmica* 1994; **11**(1):214.
12. Nathuji R, Schwan K. Virtualpower: Coordinated power management in virtualized enterprise systems. *ACM SIGOPS Operating Systems Review* 2007; **41**(6):265–278.
13. Kusic D, Kephart JO, Hanson JE, Kandasamy N, Jiang G. Power and performance management of virtualized computing environments via lookahead control. *Cluster Computing* 2009; **12**(1):1–15.
14. Srikantaiah S, Kansal A, Zhao F. Energy aware consolidation for cloud computing. *Cluster Computing* 2009; **12**:1–15.
15. Cardosa M, Korupolu M, Singh A. Shares and utilities based power consolidation in virtualized server environments. *Proceedings of the 11th IFIP/IEEE Integrated Network Management (IM 2009)*, Long Island, NY, USA, 2009.
16. Verma A, Ahuja P, Neogi A. pMapper: Power and migration cost aware application placement in virtualized systems. *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware (Middleware 2008)*, Springer, Leuven, Belgium, 2008; 243–264.
17. Verma A, Dasgupta G, Nayak TK, De P, Kothari R. Server workload analysis for power minimization using consolidation. *Proceedings of the 2009 USENIX Annual Technical Conference*, San Diego, CA, USA, 2009; 28–28.
18. Gandhi A, Harchol-Balter M, Das R, Lefurgy C. Optimal power allocation in server farms. *Proceedings of the 11th International Joint Conference on Measurement and Modeling of Computer Systems*, ACM New York, NY, USA, 2009; 157–168.
19. Jung G, Joshi KR, Hiltunen MA, Schlichting RD, Pu C. Generating adaptation policies for multi-tier applications in consolidated server environments. *Proceedings of the 5th IEEE International Conference on Autonomic Computing (ICAC 2008)*, Chicago, IL, USA, 2008; 23–32.

[§]The OpenStack Cloud Computing Platform. http://www.openstack.org/

Concurrency Computat.: Pract. Exper. (2012)
DOI: 10.1002/cpe.1867

20. Jung G, Joshi KR, Hiltunen MA, Schlichting RD, Pu C. A cost-sensitive adaptation engine for server consolidation of multitier applications. *Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware (Middleware 2009)*, Urbana Champaign, IL, USA, 2009; 1–20.
21. Zhu X, Young D, Watson BJ, Wang Z, Rolia J, Singhal S, McKee B, Hyser C, Gmach D, Gardner R, *et al.*. 1000 islands: Integrated capacity and workload management for the next generation data center. *Proceedings of the 5th International Conference on Autonomic Computing (ICAC 2008)*, Chicago, IL, USA, 2008; 172–181.
22. Beloglazov A, Abawajy J, Buyya R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems* 2011; doi:10.1016/j.future.2011.04.017.
23. Kumar S, Talwar V, Kumar V, Ranganathan P, Schwan K. vManage: loosely coupled platform and virtualization management in data centers. *Proceedings of the 6th international conference on Autonomic computing (ICAC 2009)*, Barcelona, Spain, 2009; 127–136.
24. Barford P, Crovella M. Generating representative web workloads for network and server performance evaluation. *ACM SIGMETRICS Performance Evaluation Review* 1998; **26**(1):151–160.
25. Feitelson DG. Workload modeling for performance evaluation. *Lecture notes in computer science* 2002; **2459**:114–141.
26. Li H. Workload dynamics on clusters and grids. *The Journal of Supercomputing* 2009; **47**(1):1–20.
27. Berral JL, Goiri , Nou R, Juli F, Guitart J, Gavald R, Torres J. Towards energy-aware scheduling in data centers using machine learning. *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, Passau, Germany, 2010; 215–224.
28. Borodin A, El-Yaniv R. *Online computation and competitive analysis*, vol. 53. Cambridge University Press, New York, 1998.
29. David SB, Borodin A, Karp R, Tardos G, Widgerson A. On the power of randomization in online algorithms. *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, Baltimore, MD, USA, 1990; 379–386.
30. Song B, Ernemann C, Yahyapour R. Parallel computer workload modeling with Markov chains. *Proceedings of the 11th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2005)*, Cambridge, MA, USA, 2005; 47–62.
31. Minas L, Ellison B. *Energy Efficiency for Information Technology: How to Reduce Power Consumption in Servers and Data Centers*. Intel Press, 2009.
32. Voorsluys W, Broberg J, Venugopal S, Buyya R. Cost of virtual machine live migration in clouds: A performance evaluation. *Proceedings of the 1st International Conference on Cloud Computing (CloudCom 2009)*, Springer, Beijing, China, 2009.
33. Huber PJ, Ronchetti E, Corporation E. *Robust statistics*, vol. 1. Wiley Online Library, 1981.
34. Cleveland WS. Robust locally weighted regression and smoothing scatterplots. *Journal of the American statistical association* 1979; **74**(368):829–836.
35. Kendall MG, Ord JK. *Time-series*. Oxford University Press, Oxford, 1973.
36. Cleveland WS, Loader C. Smoothing by local regression: Principles and methods. *Statistical theory and computational aspects of smoothing* 1996; **1049**:10–49.
37. Cleveland WS. *Visualizing data*. Hobart Press, Summit, New Jersey, 1993.
38. Abdi H. Multiple correlation coefficient. *Encyclopedia of Measurement and Statistics (edited by Neil J. Salkind)*, Sage, Thousand Oaks, CA, USA, 2007; 648–651.
39. Yue M. A simple proof of the inequality FFD (L)< 11/9 OPT (L)+ 1,for all l for the FFD bin-packing algorithm. *Acta Mathematicae Applicatae Sinica (English Series)* 1991; **7**(4):321–331.
40. Calheiros RN, Ranjan R, Beloglazov A, Rose CAFD, Buyya R. CloudSim: a toolkit for modeling and simulation of Cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience* 2011; **41**(1):23–50.
41. Park KS, Pai VS. CoMon: a mostly-scalable monitoring system for PlanetLab. *ACM SIGOPS Operating Systems Review* 2006; **40**(1):74.
42. Meisner D, Gold B, Wenisch T. PowerNap: eliminating server idle power. *ACM SIGPLAN Notices* 2009; **44**(3):205–216.
43. Beloglazov A, Buyya R. 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, 2010; 4.

DOI: 10.1002/cpe.1867