Energy-efficient and SLA-Aware Management of IaaS Clouds

Damien Borgetto[1]
IRIT
University of Toulouse
Toulouse, France
borgetto@irit.fr

Michael Maurer[1]
Distributed Systems Group
Vienna University of Technology
Vienna, Austria
maurer@infosys.tuwien.ac.at

Georges Da-Costa
IRIT
University of Toulouse
Toulouse, France
Georges.Da-Costa@irit.fr

Jean-Marc Pierson
IRIT
University of Toulouse
Toulouse, France
Jean-Marc.Pierson@irit.fr

Ivona Brandic
Distributed Systems Group
Vienna University of Technology
Vienna, Austria
ivona@infosys.tuwien.ac.at

ABSTRACT

Cloud computing utilizes arbitrary mega-scale computing infrastructures and is currently revolutionizing the ICT landscape by allowing remote access to computing power and data over the Internet. Besides the huge economical impact Cloud technology exhibits a high potential to be a cornerstone of a new generation of sustainable and energy-efficient ICT. The challenging issue thereby is the energy-efficient utilization of physical machines (PMs) and the resource-efficient management of virtual machines (VMs) while attaining promised non-functional qualities of service expressed by means of Service Level Agreements (SLAs). Currently, there exist solutions for PM power management, VM migrations, and dynamic reconfiguration of VMs. However, most of the existing approaches consider each of them alone, and only use rudimentary concepts for migration costs or disrespect the nature of the highly volatile workloads. In this paper we present an integrated approach for VM migration and reconfiguration, and PM power management. Thereby, we incorporate an autonomic management loop, where proactive actions are suggested for all three areas in a hierarchically structured way. We evaluate our approach with both, synthetic workload data and real-word monitoring data of a Next Generation Sequencing (NGS) application used for the protein folding in the bioinformatics area. The efficacy of our approach is evaluated by considering classical algorithms like First Fit, Monte Carlo and Vector Packing, adapted for energy-efficient reallocation. The results show energy savings up to 61.6% while keeping acceptably low SLA violation rates.

e-Energy 2012, May 9-11 2012, Madrid, Spain.

1. INTRODUCTION

Cloud computing operators provide services and infrastructures to thousands of users, enabling them to have full control over the provided platform. This is achieved by using virtualization technologies allowing isolation of an environment as well as reconfiguration and migration of the Virtual Machines (VM). However, even in the state-of-the-art data centres, the massive amount of physical machines (PMs) leads to high power consumption and carbon footprint of the data center, as well as high operation costs [11]. Furthermore, not violating so-called Service Level Agreements (SLAs), which guarantee by contract Quality of Service (QoS) requirements to the users, is of another paramount, yet conflicting importance for the Cloud provider.

There are several distinct possibilities to save energy on Cloud computing infrastructures while keeping SLAs. On the one hand, using VM migration, one could reallocate the VMs running on a platform to either consolidate them if the platform is underutilized, or to load-balance them if disparities appear in PM load, as well as to reduce load on overloaded (or to-be overloaded) PMs. On the other hand, the best allocation of VMs to PMs can still waste a lot of energy, if the resources allocated to these VMs are overestimated. This is usually the case, as SLAs, which the size of a VMs is normally based upon, are often specified rather pessimistically to guarantee for peak workload. Furthermore, powering off the exact number of unused PMs such that they are again available when needed is another crucial question. Since the workload of a service running on a VM might change quite unexpectedly over time, the problem of providing proper resources and allocations of VMs does not only have to be solved once, but has to be re-evaluated and re-configured frequently over time. Consequently, finding usable solutions in an acceptable small amount of time, leading to high scalability of the solution, is an additional constraint to this problem.

Unfortunately, VM migrations and powering on/off PMs

[1]contributed equally to this paper

do have a cost. The impact they have on the energy consumption of the system is another factor for this problem: First, VM reconfiguration triggers migrations that would not have occurred for a static system. Second, the migrations of VMs cannot be done instantly, whether with offline (pause, migrate, unpause) or live migration (migrate while the VM still runs at the cost of slower and higher network data transfer). The usual solution to diminish the time taken to migrate a VM is to use network storage to store system images, as most of the data transfer will be done locally. However, migration still takes a significant amount of time, especially with services that use a high amount of memory, and when several migrations are triggered at the same time. During the migration one VM requires two running PMs (source and target PMs). This is not energy-efficient.

The research presented in this paper unifies all the three mentioned methodologies (VM reconfiguration, VM migration, PM power management), presents concrete implementations for each of them, and tackles all of the mentioned constraints (migration and power management costs, and scalability). The ultimate goal of the implemented framework is to (a) *minimize energy consumption*, while (b) *minimizing SLA violations*. For the VM reconfiguration we adapt a rule-based approach formerly introduced in [18]. For VM allocation and reallocation we present four different strategies, where we modify simple algorithms like First Fit, but also smarter algorithms based on a Monte Carlo method or vector packing. Finally, power management relies on threat thresholds and statistics about recent PM utilization. As to migrations and power management, we do not relate their energy costs to some arbitrary cost function, but to the time it takes to perform the respective actions. The basis of this work is an autonomic control loop using a knowledge base that autonomically governs the infrastructure based on high-level input: SLAs and energy performance parameters.

We evaluate the described approach with well-established synthetic workloads for Cloud Computing infrastructures [17, 18], as well as real-world data stemming from monitoring a scientific bioinformatic workflow for Next Generation Sequencing (NGS). NGS is a recently introduced technology for identifying nucleotide molecules like RNA or DNA in biomedical samples. It has been successfully adapted as a Cloud computing application [6].

The main contributions of this paper are:

1. Combining resource-efficient and SLA-enacting VM reconfiguration with energy-efficient VM allocation and reallocation, and PM power management.

2. Naturally integrating costs for migrations and PM power management into an existing power model.

3. Evaluating the proposed framework with several workloads achieving energy savings up to 61.6% only by VM reconfiguration, and additional up to 37% by VM reallocation algorithms while keeping SLA violations at a minimum.

The remainder of this paper is organized as follows: Section 2 presents related work. Whereas Section 3 generally describes the models used for energy, VM reconfiguration and migration, as well as PM power management, Section 4 details the implemented solutions. Afterwards, Section 5 discusses the evaluation of the presented approach, and Section 6 concludes the paper.

2. RELATED WORKS

We can divide related work in this area into two parts: resource-efficient and SLA-enacting VM and knowledge management, and energy-efficient and/or SLA-aware PM management.

As to the first part, there has been some considerable work on optimizing resource usage while keeping QoS goals. These papers, however, concentrate on specific subsystems of Large Scale Distributed Systems, as [13] on the performance of memory systems, or only deal with very generic SLA parameters. Petrucci [23] or Bichler [4] investigate one general resource constraint, whereas we focus on two concrete parameters, where we also allow for easy extension to others. The Sandpiper framework [28] offers black-box and gray-box resource management for VMs. Contrary to our approach, though, it plans reactions just after violations have occurred. Also Rao et al. [24] with their VCONF model pursue SLA violation minimization, but can only execute one action per iteration and neglect the energy consumption of executed actions. Additionally, none of the presented papers uses a knowledge base (KB) for possible reconfiguration and self-adaptation. Hoyer et al. [8] also undertake a speculative approach as in Section 4.1, but by overbooking PM resources. They assign VMs to PMs that would exceed their maximum resource capacities, because VMs hardly ever use all their assigned resources. Computing this allocation they also take into consideration workload correlation of different VMs. Stillwell et al. [25] in a similar setting define the resource allocation problem for static workloads, present the optimal solution for small instances and evaluate heuristics by simulations. Nathani et al. [21], e.g., also deal with VM placement on PMs using scheduling techniques. [10] reacts to changing workload demands by starting new VM instances; taking into account VM startup time, they use prediction models to have VMs available already before the peak occurs. Summarizing we can say that there has been a great deal of work on the different action levels, whereas VM reconfiguration has not been observed yet.

As to KM and autonomic management of SLAs, especially rule-based systems have gained some interest. Paschke [22] et al. investigate a rule-based approach in combination with the logical formalism ContractLog. Their formalism specifies rules to trigger after a violation has occurred, but it does not deal with avoidance of SLA violations. Bahati et al. [2] also use policies, i.e., rules, to achieve autonomic management. They provide a system architecture including a KB and a learning component, and divide all possible states of the system into so called regions, which they assign a certain benefit for being in this region. The actions are not structured, but are mixed together into a single rule, which makes the rules very hard to manage and to determine a salience concept behind them. However, we share the idea of defining "over-utilized", "neutral" and "under-utilized" regions. Our KM system allows to choose any arbitrary number of resource parameters that can be adjusted on a VM.

As far as the second level is concerned, several papers focus on different levels (as described in Section 3). [29, 20, 15, 26] focus on VM migration and [19] on turning on and off physical machines, whereas our paper achieves a more holistic approach taking all these mentioned level plus VM reconfiguration into account. Voorsluys et al. [27] tackle the cost of live migration of virtual machines regarding the response time of the services inside the VMs in order to

match the response time with the SLA requirements of the services. Liu et al. [14] also have studied live migration of virtual machines in order to model the performance and energy of the migration. They show that migration is an I/O intensive application, and that it consumes energy on both ends. The architectural framework proposed in [3] for green clouds also achieves VM reconfiguration, allocation and reallocation. The authors use a CPU power model to monitor the energy consumption of the cloud. The algorithm they propose to dynamically consolidate VMs significantly reduces the global power consumption of their infrastructure. Their work, however, differs from our approach in several points. Most importantly, they use a different VM migration model, reactive VM reconfiguration actions instead of proactive ones, and they neglect the time to power on and off PMs. Our research provides a more wholesome approach than related work and integrates most of the different possible action levels seen in the literature.

3. ACTION, MIGRATION AND POWER MANAGEMENT MODELS

In this section we will model our Cloud infrastructure and its autonomic control loop including VMs, PMs, SLAs, migration, PM power management, and energy consumption, and the governance thereof.

Governing the Cloud infrastructure is achieved by an autonomic control loop called MAPE-K [9], where we monitor (M) relevant metrics on the infrastructure; next we analyze (A) them and plan (P) proactive and reactive actions, which are then executed (E). All this is done in combination with a knowledge base (K), which stores relevant information and together with a decision mechanism tries to recommend appropriate actions to take. This is summarized in Figure 1. The monitoring phase puts measurements into the KB (step 1), and the analysis phase asks it to recommend actions (step 2). The returned actions (step 3) are finally reflected in the KB (step 4). The KB itself stores knowledge about applications, VMs, and PMs as their current and past CPU or memory utilization. It also keeps track of agreed SLAs, the inserted measurements, the recommended actions, and optionally federated clouds that allow outsourcing of applications.

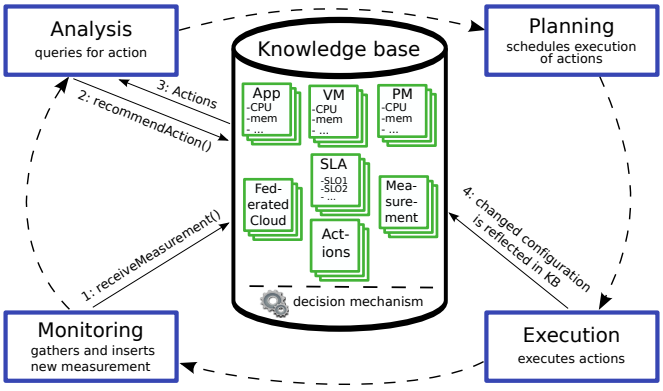

Figure 1: Autonomic MAPE-K loop for Cloud Computing infrastructures

As possible reconfiguration actions we divide this resource allocation problem into the following levels:

1. VM reconfiguration
2. VM migration
3. PM powering on/off

A fourth possibility, e.g., would be VM outsourcing to Cloud federations [12], but we focus on solving the resource allocation problem inside a Cloud first.

As far as our energy model is concerned, we define the energy consumption E of a PM j as

$$E^j = E^j_{min} + ut^{CPU,j} \cdot (E^j_{max} - E^j_{min}), \qquad (1)$$

where E^j_{min} and E^j_{max} represent the minimum and maximum energy consumption of a certain PM j, respectively, and $ut^{CPU,j}$ signifies the utilization of the CPU of PM j, with values between 0 and 1. Thus, in our model energy consumption only depends on CPU utilization, and we make a linear interpolation of the PM's energy consumption at idle state (E_{min} when $ut^{CPU} = 0$) and when fully loaded (E_{max} when $ut^{CPU} = 1$). While this energy model might not be fully realistic, it is corroborated by experiments in the literature such as [7, 3].

We assume that one VM resides on exactly one PM except when it is migrated. Then it resides on exactly two PMs. We consider a heterogeneous system, thus VMs and PMs with possibly different amounts of resources, and PMs with different energy characteristics. Furthermore, we assume that the amount of resources a VM is provided can be adapted from one iteration to the next, and that PMs can be powered on and off. However, VM migrations and powering PMs on and off is not considered "free of charge", as far as energy is concerned. We define *migration_time*, *startup_time* and *shutdown_time* as the time (in iterations through the MAPE-K cycle) it takes a VM to migrate, and a PM to start up or shut down, respectively. Figure 2 shows how migrations are handled in our energy model. We do not simply add any arbitrary value as a penalty for migrations, but we place the VMs on both PMs, the PM the VM is migrating from and the PM it is migrating to, for *migration_time* iterations. Thus, the energy cost of a migration is indirectly measured by occupying the resources of two PMs and leading to CPU utilization on the source and target PMs for *migration_time* iterations. We similarly proceed with the power management of PMs. We assume that powering on and off PMs takes some time as defined in *startup_time* and *shutdown_time*. This is shown in Figures 3 and 4, respectively.

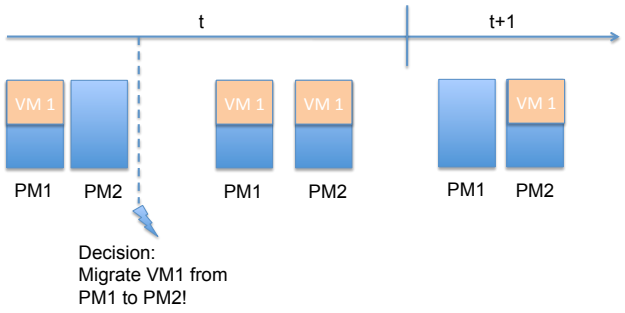

Figure 2: Concept for migrations with *migration_time*=1

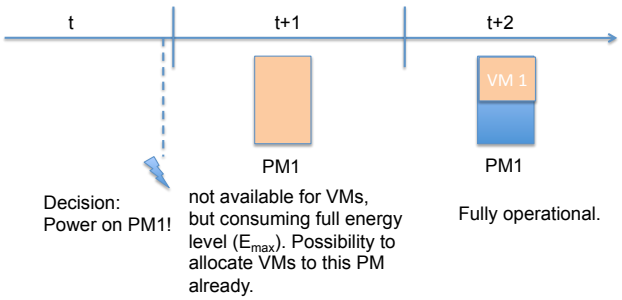

Figure 3: Concept for powering on PMs with *startup_time*=1

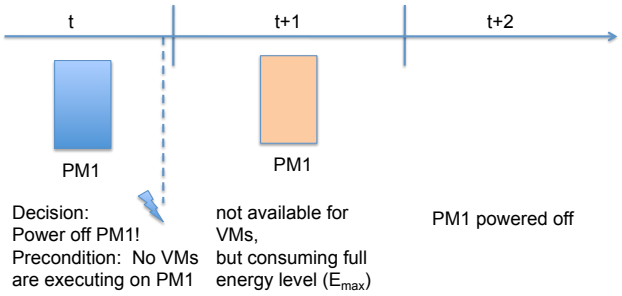

Figure 4: Concept for powering off PMs with *shutdown_time*=1

Figure 5 shows a possible configuration of a sample Cloud infrastructure at six time steps. There are 2 PMs and 2 VMs at time step t: PM 1 hosts the 2 VMs and PM 2 is powered off. Only 2 resources (CPU and memory) are shown in the figure. At $t+1$, the CPU and memory consumption of the VMs are increasing requiring the second PM to be powered on. The machine is effectively powered on at $t+2$ and can be used, so we begin the migration of the VM 1 from PM 1 to PM 2. The migration will last for one time step. At $t+3$ the migration ended and we do not need to modify the system again. At $t+4$ the resource needs of VM 2 have greatly decreased, making it possible to consolidate the system safely by beginning to migrate VM1 back from PM 2 to PM 1. At the last time step we can shutdown PM 2, which is unused.

4. AUTONOMIC ADAPTATION

This section describes the mechanisms for the three identified areas for re- and proactive actions: VM reconfiguration, VM reallocation, and PM power management.

4.1 VM reconfiguration

For VM reconfiguration, we adapt a rule-based approach first described in [18].

As introduced in [17], we distinguish between agreed, used and provided values for a certain resource. A Service Level Objective (SLO) as part of an SLA is of the form "$r_{min} \leq$ SLA parameter $\leq r_{max}$". For *memory*, e.g., an SLO might look like $8MB \leq$ memory $\leq 512MB$. We will refer to r_{max} as the agreed value. Furthermore, we measure how much a VM currently *uses* and how much we *provide* to it on any PM, and can thus calculate the utilization of a resource as $ut^r = \frac{\text{used}^r}{\text{provided}^r}$. Consequently, an SLA violation occurs if

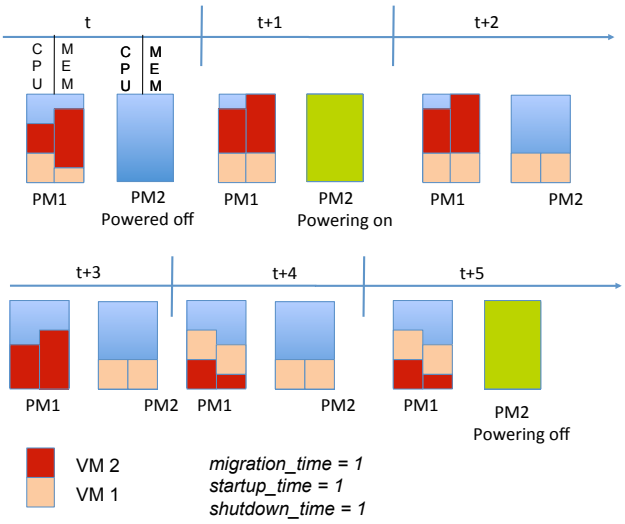

Figure 5: Configuration sample at six time steps

one would like to use more than provided, but less then agreed.

From the discussion and the definition of the overall goals (a) and (b) from Section 1 we determine the following subgoals for this action level.

1. Minimize number of SLA violations
2. Maximize VM resource utilization
3. Minimize number of reallocation actions

When viewing the system only at the level of VMs we can thus directly infer subgoal (1) from goal (a). However, as no energy can be measured when ignoring PMs, we have to infer two new subgoals out of goal (b): (2) and (3). Neglecting subgoal (3) would lead to unstable system behavior enchaining a high probability of necessary migrations. Of course, when there are no reallocation actions, there is no need for migrations once the PMs have been consolidated (see Section 4.2).

We now take a speculative approach and investigate whether we can *provide* less than *agreed*, but more than currently *used*, so that we do not run into an SLA violation, but keep resource utilization high. Not taking this approach would mean provisioning all VMs at their maximum resource configuration. Usually, the maximum resource configuration is set so high that it can handle peak usage. Always providing this maximum would thus result into huge resource wastage for all non-peak time instances.

The main idea of the rule-based approach is that utilization ut^r of any resource r is divided into three regions +1, 0, and −1 signifying under-, optimal, and over-utilization, respectively. The regions are identified by two so called threat thresholds (TTs), i.e., TT^r_{low} and TT^r_{high}, and are defined as

- Region +1: $0 \leq ut^r < TT^r_{low}$
- Region 0: $TT^r_{low} \leq ut^r \leq TT^r_{high}$
- Region -1: $TT^r_{high} < ut^r \leq 100$.

Every resource can have its own favorable pair of TTs. We call the center of region 0 the target value tv. Figure

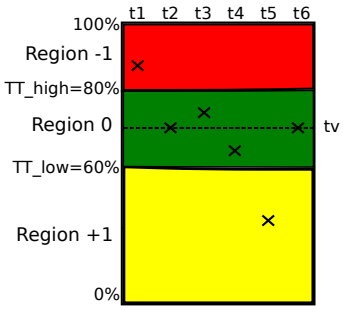

Figure 6: Principle of rule-based approach

6 depicts a sample utilization situation for an arbitrary resource at six succeeding time steps. At t_1 the utilization of the resource is in region +1, which means that the resource is under-provisioned. Thus, the KB recommends to increase the provided resource in order to move utilization to tv, which in this example is situated at 70%. As long as the utilization stays in region 0, which is true for t_2, t_3, and t_4, the KB recommends no action. Only when the utilization leaves again this region as at t_5, utilization is set back to tv. A large enough span between the thresholds TT^r_{low} and TT^r_{high} helps to prevent oscillations of repeatedly increasing and decreasing the same resource. Figures 7 and 8 depict rule schemes for increasing and decreasing resources, respectively.

1 **IF**
2 $ut^r > TT^r_{\text{high}}$ AND $ut^r_{predicted} > TT^r_{\text{high}}$
3 **THEN**
4 Set pr^r to $\frac{\text{use}^r}{tv(r)}$ for policy modes with plenty of resource left.
5 Set pr^r to $\min(\frac{\text{use}^r}{tv(r)}, SLO^r * (1 + \epsilon/100))$ for policy modes with scarce overall resource situation.

Figure 7: Rule scheme for increasing a resource

1 **IF**
2 $ut^r < TT^r_{\text{low}}$ AND $ut^r_{predicted} < TT^r_{\text{low}}$
3 **THEN**
4 Set pr^r to $\max(\frac{\text{use}^r}{tv(r)}, minPr^r)$.

Figure 8: Rule scheme for decreasing a resource

The rules have been implemented using the Java rule engine Drools [1]. Different policy modes will load slightly modified rules into the Drools engine and thus achieve a high adaptability of the KM system reacting to the general performance of the Cloud infrastructure.

Preliminary evaluation results in [18] show that the rule-based approach outperforms many other KM techniques useful for VM resource allocation as Case Based Reasoning, Situation Calculus or Default Logic [16]. It achieves SLA violation rates below 1% and resource utilization above 70% with a maximum number of reconfiguration actions of 20%.

4.2 Power-Aware Reallocation: VM Migrations

In order to achieve power-aware allocation and reallocation of the VMs, we implemented several algorithms with different behaviours. For each algorithm we implemented a first allocation version, which will do the initial mapping of the virtual machines to the physical machines. We then implemented a reallocation algorithm that will output a mapping out of an initial allocation, using the VM migration model.

First Fit

The FirstFit algorithm for the first allocation problem is the well-known mapping algorithm that will allocate each VM to the first host on which it fits. We, however, added a power-aware component to the algorithm, as we will try to allocate first on the PM that will have the smallest maximum power consumption which, as shown in [5] has proven to consume less energy.

When reallocating the VMs the usual packing algorithms cannot be used, since we have to take into account where the VMs were allocated at the previous time step, and the fact that they will consume resources on both source and destination PMs. The algorithm works in two steps. First we pick the most loaded PM and we distribute half its load in a first fit fashion. Then we pick the least loaded PM, and distribute all its load in a first fit fashion.

Round Robin

The RoundRobin algorithm for the first allocation allocates one VM to each PM until no VM is left to allocate. We added the same power-aware component as the FirstFit algorithm.

When reallocating the VMs, we first take the most loaded PM and spread its load on other non empty PMs in a round robin fashion, then we take one VM on each PM and put it on an empty host. This reallocation algorithm, even if it does not perform well power-wise, it will serve as a baseline for other algorithms.

Monte Carlo

The MonteCarlo algorithm works on the basis of the well known Monte Carlo method, which uses probabilistic techniques to compute a numerical value. In our case, we will do the first allocation using the RoundRobin algorithm, which allows us to have an evenly balanced system, and that way converge faster to a good solution.

For the reallocation, the algorithm computes the cost of the current allocation using several parameters in order to:

- increase the cost for each job migrating.
- increase the cost for each overloaded host.
- decrease the cost for each host that is empty or will be empty.

Each parameter can be changed to modify the weight of the migrations, PM overloads and PM powering down. In our tests, we will use the values 1 for the migrating jobs, 4 for the overloaded hosts and -10 for a PM that will be empty. This means that if we take for instance a system with 3 PMs and 1 VM migrating from PM1 to PM2 we will get the cost $1\times1+4\times0+(-10\times2) = -15$. The algorithm then computes a random set of job migrations, in order to get the new cost of the reallocation. If the cost is lower, we keep the set of migrations. The algorithm repeats those operations a fixed number of times, to emerge the best set of migrations for the next iteration.

Vector Packing

The VectorPacking algorithm tries for the first allocation to allocate each job, beginning with the jobs with the highest resource needs on the hosts with the lowest maximum power consumption. It uses a vector packing technique that sorts the jobs according to their highest resource need. It then allocates each job picking it from the list that will counter the current imbalance in the PM's resource loads.

For the reallocation the algorithm consolidates the VMs as much as possible using the same heuristic as the allocation. It then load balances some VMs on the most loaded PMs that will not be empty in the future iterations. That way, the algorithm aims to pack the VMs to a minimum number of PMs and then load balance the VMs between those PMs.

4.3 PM Power Management

Eventually, actual power can only be saved, when PMs are powered off. However, the question of how many PMs should be powered off when in order not to risk future SLA violations is not trivial. Thus, we designed the following powering off strategy: We consider all empty PMs at the current iteration, i.e., all PMs that have no VMs running on them. We decide to switch off a certain fraction a of them, i.e.,

$$\text{Number of PMs to switch off} = \frac{\text{Number of empty PMs}}{a}$$

This means that when the number of empty PMs stays constant, this technique turns off all but one PMs in an exponential manner. Thus, when n represents the number of empty PMs, we want to know when there will be only 1 PM left. So we need to solve $n \cdot a^{-t} = 1$ for t, which results into $t = \lceil -\log^a \frac{1}{n} \rceil$. This is the number of iterations it will take to power off all (but one) PMs. This last PM is kept as a spare PM in order to serve sudden increases in demand as a first resort. Consequently, this technique allows to power off all machines very quickly in case of stable VMs, but always keeps a certain fraction powered on in case VM resources start to increase again.

As far as powering on machines is concerned, we monitor average utilization for every resource on all PMs and define – similarly to the rule-based approach (cf. Section 4.1) – resource-dependent threat thresholds. If *any* of these resources exceeds its TT, we power on as many PMs such that the average resource utilization again falls below its TT.

In both cases, we always power on most energy-efficient PMs first, and power off least energy-efficient PMs first.

5. EVALUATION

In this section we will evaluate the framework of VM reconfiguration and reallocation algorithms together with PM management. We divide the evaluation into 4 experiments. In the first experiment (Section 5.1) we determine the energy gain VM reconfiguration brings alone. In the second experiment (Section 5.2) we focus on the four different reallocation algorithms to see which one performs best. In the third experiment (Section 5.3) we more deeply investigate some of the parameters for the two best reallocation algorithms. Finally, with the fourth experiment we evaluate the scalability (Section 5.4) of the algorithms.

We simulate 100 PMs with 1.1GHz processors and 4GB memory, that consume 20W at idle (E_{min}) and 100W when fully loaded (E_{max}). VMs were each 500 MHz of CPU and 500 MB of memory at $t = 0$. We used several different workloads for the 100 VMs of the system, two synthetic ones and one based on real measurements of a scientific bioinformatic workflow presented in [6]. For the synthetic workloads we distinguish between LIGHT workload volatility, i.e., workload does not change a lot (up to 10% from one iteration to the other), and the opposite MEDIUM_HEAVY (up to 50% from one iteration to the other workload volatility. A more detailed description of the workload generation can be found in [18]. We will abbreviate the bioinformatic workflow with BOKU. We evaluated the algorithms with 100 iterations and the PM powering off strategy with $a = 2$, unless stated otherwise.

5.1 Impact of VM reconfiguration over energy consumption

We perform the first set of runs to experiment on the effect of the VM reconfiguration on the energy consumption of the system. In order to do so, we run the four algorithms with a fixed workload volatility class (here MEDIUM_HEAVY), and a fixed set of TT pairs. We then compare to the same runs with VM reconfiguration disabled. Evaluation parameters can be found in Table 1.

Parameter	Evaluated values
Tested workloads	MEDIUM_HEAVY volatility
VM reconfiguration	turned on/off
VM reconfiguration TT pairs	[20%, 40%]
VM reallocation algorithms	RoundRobin, FirstFit, MonteCarlo, VectorPacking
tt_{cpu}	0.8
tt_{memory}	0.8

Table 1: Evaluation input parameters for Experiment 1

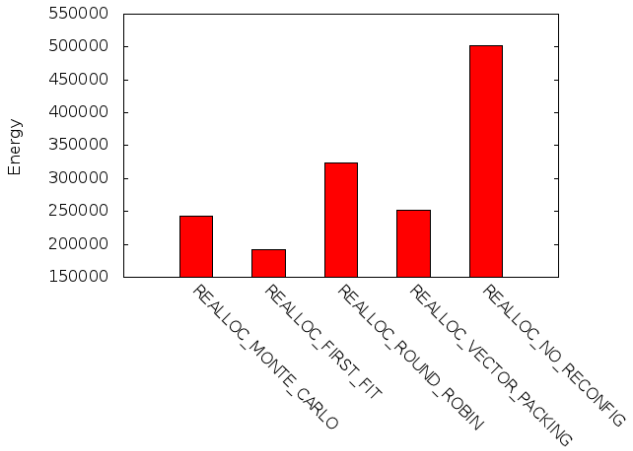

Figure 9: Energy consumption for Experiment 1

Figure 9 shows the total energy consumption over the 100 time steps. We only plotted one of the no reconfiguration results since all the 4 runs had the same energy consumption. The reason is that all algorithms achieve the optimal first allocation of 2 VMs per PM, since the initial CPU requirement of VMs 500MHz, and this requirement does not

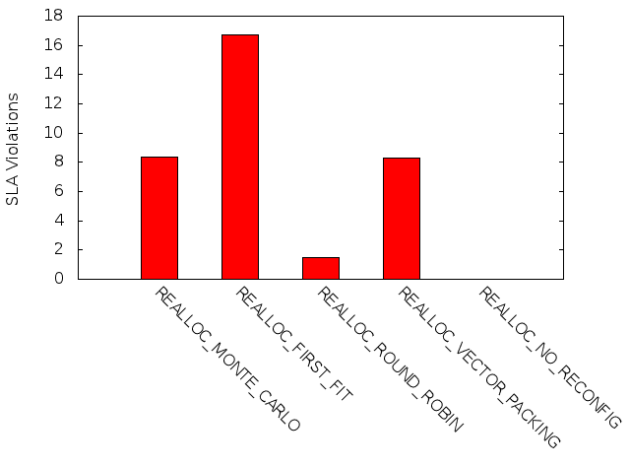

Figure 10: SLA violations for Experiment 1

change throughout the run. The difference between the best achieving algorithm with VM reconfiguration on and off goes up to 61.6%, since our PMs have a low E_{min} compared to E_{max}, which enables a better performance for underutilized VMs. However, this gain in the energy consumption can be made at the price of more SLA violations as shown if Figure 10 due to the speculative reconfiguration approach.

5.2 Evaluation of VM reallocation algorithms

In order to evaluate the performance of the VM reallocation algorithms, we run the simulations for the 4 algorithms with the VM reconfiguration turned on, and with one set of TT pairs. We evaluate three different workloads: low and medium-high volatility of the resource needs, and the bioinformatic workflow. Parameters are shown in Table 2.

Parameter	Evaluated values
Tested workloads	LIGHT, MEDIUM_HEAVY, BOKU
VM reconfiguration	turned on
VM reconfiguration TT pairs	[20%, 40%]
VM reallocation algorithms	RoundRobin, FirstFit, MonteCarlo, VectorPacking
tt_{cpu}	0.8
tt_{memory}	0.8

Table 2: Evaluation input parameters for Experiment 2

Figure 11 shows the total energy consumption of the PMs over the 100 time steps for the 3 workloads. As the figure shows, for LIGHT volatility workloads, the MonteCarlo algorithm performs best, closely followed by VectorPacking and FirstFit. The RoundRobin algorithm performs badly since it is consuming twice as much energy. If we look at the MEDIUM_HEAVY volatility workload, we can see that the FirstFit algorithm outperforms all other algorithms going up to 37% less energy consumption compared to RoundRobin. Finally, for the BOKU workload, which stresses the resources more than the two other workloads, the results are the same as for the LIGHT workload, only with a generally much higher energy consumption.

The reason behind these differences is partially shown in Figure 13, which shows the average number of powered on PMs during the run. As we can see, the RoundRobin will load balance the VMs on every PM, thus preventing the autonomic manager to shut down empty PMs. The algorithm that performs best, however, is the VectorPacking algorithm, since it is designed to heavily consolidate the VMs, while load balancing if possible on the PMs that remain powered on. We note that, except for FirstFit and RoundRobin, the number of powered on PMs increases as the system resource consumption becomes more volatile. This can be explained by the fact that the FirstFit algorithm is less proactive than the others, leading to problems as we will see in Figure 12.

Figure 12 plots the SLA violation percentage of the cloud for each algorithm. Only the LIGHT and MEDIUM_HEAVY workloads are plotted, since the BOKU workload is much less volatile than the others and has an SLA violation rate of 0%. As we see, the RoundRobin has the least violation percentage of all the algorithms, since it uses all the PMs. The small amount of violations is generated by the VM reconfiguration. The VectorPacking and MonteCarlo algorithms are around 4% and 8% of SLA violations. Last, the FirstFit algorithm which performs better for the LIGHT volatility workload, performs poorly when the volatility increases, since it goes up to over 16% SLA violations.

To examine the performance of the algorithms, we have to account for both the energy consumption of the cloud, and the SLA violations that the reconfiguration of the VMs and the PMs have induced. The perfect example is when looking at the FirstFit algorithm for the MEDIUM_HEAVY workload. For these parameters, the algorithm performs extremely well energy-wise, outperforming *smarter* algorithms, but the setback is to have over 16% SLA violations. Looking at the global picture, we have a 60kW difference between the two algorithms for a 8 point difference of SLA violations.

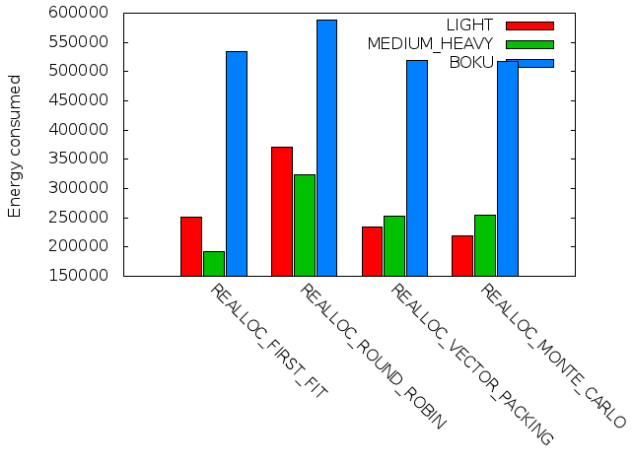

Figure 11: Energy consumption of the algorithms under different workloads

5.3 Evaluation of VM and PM Threat Thresholds

As the next evaluation step we focus on the two threat threshold pairs we use: One for PM power management (average CPU and memory utilization of all PMs), which we call PM-TTs, and one for the VM reconfiguration (TT_{min}, TT_{max})

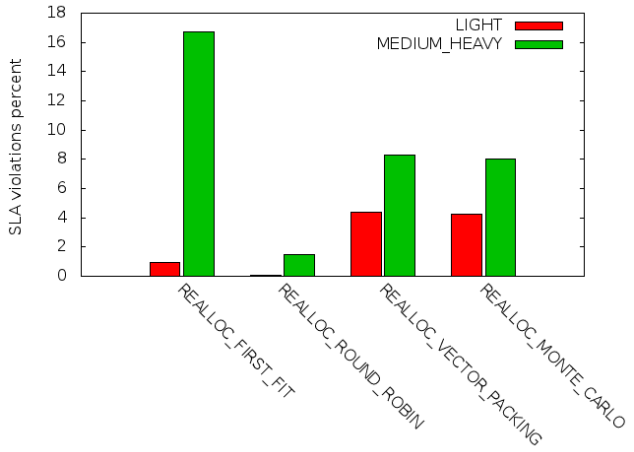

Figure 12: SLA violation percentages of the VMs under different workloads

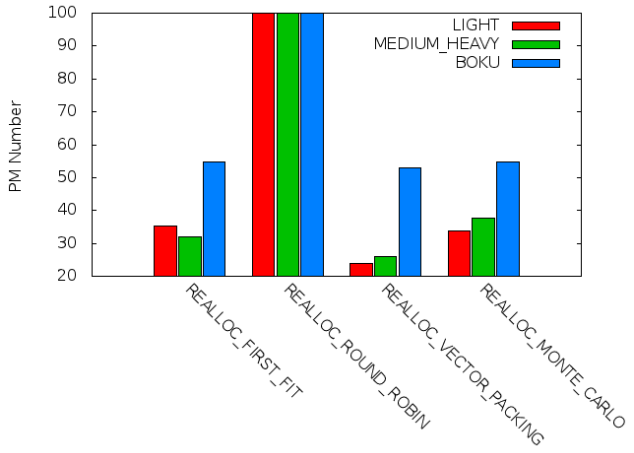

Figure 13: Average number of powered on machines

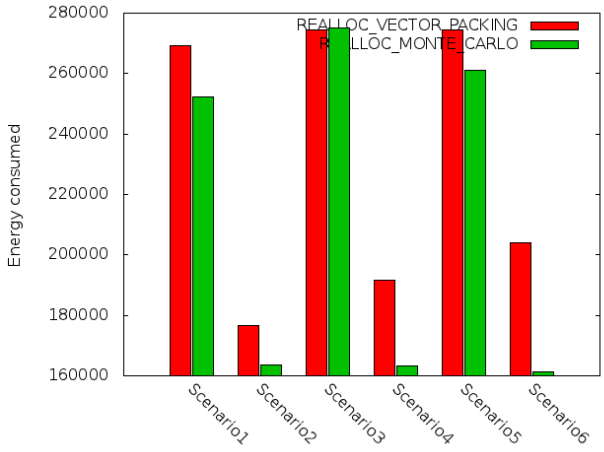

Figure 14: Energy consumption for varying VM and PM thresholds

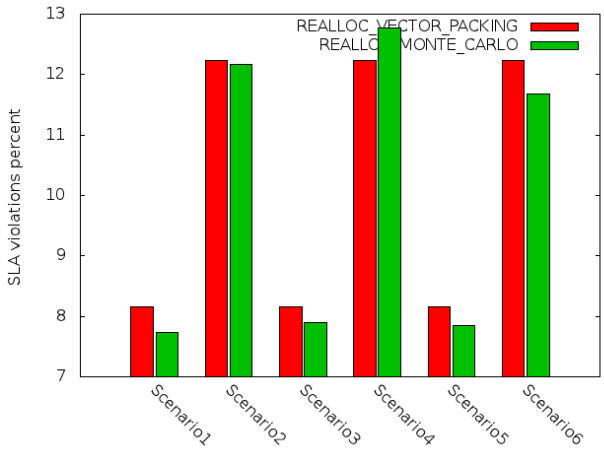

Figure 15: SLA violations for varying VM and PM thresholds

named VM-TTs. We evaluate them on the two VM reallocation algorithms that achieve best results in the previous allocations: MonteCarlo and VectorPacking. We analyze three different PM-TTs [80%, 80%], [60%, 80%], [60%, 60%] in the format $[CPU, memory]$. We use a more cautious TT for CPU in one case, because this showed to be the resource which is usually fluctuating more quickly than memory. For the VM-TTs we use the standard interval [50%, 75%] found in [18] to be a good general setting additional to the very cautious setting of [20%, 40%] used in the previous evaluation. All the resulting scenarios are depicted in Table 3.

Figures 14 and 15 show that MonteCarlo is almost always better in terms of energy and violations. However, it takes much longer processing time as presented in Section 5.4. For Scenarios 2, 4 and 6 the difference in favor of MonteCarlo is extremely large, as far as energy consumption is concerned. Not surprisingly, the scenarios, where energy consumption is lowest has the highest number of SLA violations and vice versa. Generally speaking, the even and the odd scenarios show similar behavior meaning the VM-TTs have a higher impact on the outcome as compared to the PM-TTs. Moreover, lowering PM-TTs increases energy consumption, but does not lower SLA violations in all cases. Finally, the better results of MonteCarlo can also be explained when looking at the number of PMs that were powered on or off. VectorPacking powers on at least as much (if not more) PMs as MonteCarlo, and MonteCarlo also spends less energy on powering off again PMs that were unnecessarily powered off by VectorPacking.

5.4 Scalability

Figure 16 shows the runtime of the reallocation algorithms for 100, 200, 400 and 800 VMs. These runtimes contain both the VM reconfiguration decisions and the reallocation algorithm. As we can see, the MonteCarlo algorithm is taking six time as much time to compute a solution each time step at 100 VMs, as the others are computing in a reasonable time (around half a second for 100 VMs with the reconfiguration overhead). The MonteCarlo algorithm, even if it performs rather well, will not scale well for two reasons. The first is that it has to compute lots of time the solution (100 in our tests), thus taking more and more time

	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6
PM-TTs=[CPU,Memory]	[80%, 80%]	[80%, 80%]	[60%, 80%]	[60%, 80%]	[60%, 60%]	[60%, 60%]
VM-TTs=$[TT_{low}, TT_{high}]$	[20%, 40%]	[50%, 75%]	[20%, 40%]	[50%, 75%]	[20%, 40%]	[50%, 75%]

Table 3: Scenarios for Experiment 3

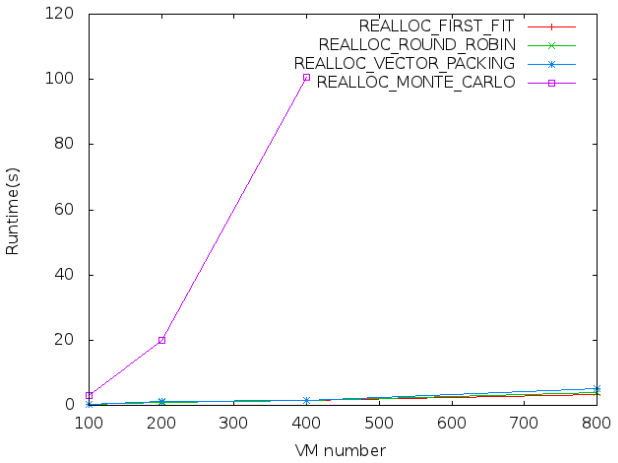

Figure 16: Runtime of the algorithms for 100 VMs

to compute. The second reason is that with an increasing number of VMs and PMs, in order to achieve a near optimal solution every time step, the algorithm has to increase its iteration number to increase the chance of a good solution to emerge. If we increase the number of VMs and PMs without increasing the number of iterations of the MonteCarlo, the quality of the results will become more sporadic, and the average quality of the solution will decrease.

Figure 16 shows that the MonteCarlo algorithm is not scalable, unlike the other 3 for which the runtime seems to grow linearly with the number of VMs, and that it takes around an acceptable 5s to compute a solution for 800 VMs.

6. CONCLUSION

In this paper we have presented a management framework for governing Cloud Computing infrastructures to achieve two goals: reduce energy consumption while keeping predefined Service Level Agreements (SLAs). We have devised a multi-level action approach that breaks down the NP-hard resource allocation problem for Clouds. We have specialized on several views of the Cloud Computing infrastructure, i.e., VM reconfiguration, VM migration, and PM power management, in order to reduce the problem's complexity. In each of these views we have defined a subproblem and solved it using a wide variety of heuristics ranging from rules over random methods, i.e., Monte Carlo, to vector packing algorithms. We have evaluated the sequential execution of these views. We showed for the first time that the VM reconfiguration algorithm alone, which was already known to minimize SLA violations and decrease resource wastage, also effectively saves up to 61.6% of energy. Considering scalable algorithms, these energy savings can still be increased by up to 37% in the best case and 11% in the worst case while keeping SLA violations at 0% for the workload of a bioinformatic scientific workflow, below 4% for synthetic workloads with low volatility for all VM migration algorithms, and below 8% for synthetic workloads with higher volatility for the smarter VM migration algorithms. For future work we plan to focus more on a possible heterogeneity of the systems, refining the migration model, and integrating the framework into a real-world Cloud computing environment.

Acknowledgments

The work described in this paper has been funded by COST-Action IC0804 on energy efficiency in large scale distributed systems and by the Vienna Science and Technology Fund (WWTF) through project ICT08-018.

7. REFERENCES

[1] Drools, www.drools.org.

[2] Raphael M. Bahati and Michael A. Bauer. Adapting to run-time changes in policies driving autonomic management. In *ICAS '08: Proceedings of the 4th Int. Conf. on Autonomic and Autonomous Systems*, Washington, DC, USA, 2008. IEEE Computer Society.

[3] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, (0):–, 2011.

[4] Martin Bichler, Thomas Setzer, and Benjamin Speitkamp. Capacity Planning for Virtualized Servers. *Presented at Workshop on Information Technologies and Systems (WITS), Milwaukee, Wisconsin, USA, 2006*, 2006.

[5] Damien Borgetto, Georges Da Costa, Jean-Marc Pierson, and Amal Sayah. Energy-Aware Resource Allocation. In *Proc. of the Energy Efficient Grids, Clouds and Clusters Workshop (E2GC2)*, page (electronic medium). IEEE, October 2009.

[6] Vincent C. Emeakaroha, Pawel Labaj, Michael Maurer, Ivona Brandic, and David P. Kreil. Optimizing bioinformatics workflows for data analysis using cloud management techniques. In *The 6th Workshop on Workflows in Support of Large-Scale Science (WORKS11)*, 2011.

[7] Xiaobo Fan, Wolf dietrich Weber, and Luiz André Barroso. Power provisioning for a warehouse-sized computer. In *In Proceedings of ISCA*, 2007.

[8] Marko Hoyer, Kiril Schröder, and Wolfgang Nebel. Statistical static capacity management in virtualized data centers supporting fine grained qos specification. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, e-Energy '10, pages 51–60, New York, NY, USA, 2010. ACM.

[9] Markus C. Huebscher and Julie A. McCann. A survey of autonomic computing—degrees, models, and applications. *ACM Comput. Surv.*, 40(3):1–28, 2008.

[10] Sadeka Islam, Jacky Keung, Kevin Lee, and Anna Liu. Empirical prediction models for adaptive resource

provisioning in the cloud. *Future Generation Computer Systems*, 28(1):155 – 162, 2012.

[11] Peter Johnson and Tony Marker. Data center energy efficiency product profile. Technical report, 2009.

[12] Gabor Kecskemeti, Gabor Terstyanszky, Peter Kacsuk, and Zsolt Neméth. An approach for virtual appliance distribution for service deployment. *Future Gener. Comput. Syst.*, 27:280–289, March 2011.

[13] Bithika Khargharia, Salim Hariri, and Mazin S. Yousif. Autonomic power and performance management for computing systems. *Cluster Computing*, 11(2):167–181, 2008.

[14] Haikun Liu, Cheng-Zhong Xu, Hai Jin, Jiayu Gong, and Xiaofei Liao. Performance and energy modeling for live migration of virtual machines. In *Proceedings of the 20th international symposium on High performance distributed computing*, HPDC '11, pages 171–182, New York, NY, USA, 2011. ACM.

[15] Liang Liu, Hao Wang, Xue Liu, Xing Jin, Wen Bo He, Qing Bo Wang, and Ying Chen. Greencloud: a new architecture for green data center. In *Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session*, pages 29–38, New York, NY, USA, 2009.

[16] Michael Maurer, Ivona Brandic, Vincent C. Emeakaroha, and Schahram Dustdar. Towards knowledge management in self-adaptable clouds. In *IEEE 2010 Fourth International Workshop of Software Engineering for Adaptive Service-Oriented Systems*, Miami, USA, 2010.

[17] Michael Maurer, Ivona Brandic, and Rizos Sakellariou. Simulating autonomic sla enactment in clouds using case based reasoning. In *ServiceWave 2010*, Ghent, Belgium, 2010.

[18] Michael Maurer, Ivona Brandic, and Rizos Sakellariou. Enacting slas in clouds using rules. In *Euro-Par 2011*, Bordeaux, France, 2011.

[19] M. Mazzucco, D. Dyachuk, and R. Deters. Maximizing cloud providers' revenues via energy aware allocation policies. In *CLOUD 2010*, pages 131 –138, 2010.

[20] Xiaoqiao Meng, Canturk Isci, Jeffrey Kephart, Li Zhang, Eric Bouillet, and Dimitrios Pendarakis. Efficient resource provisioning in compute clouds via vm multiplexing. In *Proceeding of the 7th international conference on Autonomic computing*, ICAC '10, pages 11–20, New York, NY, USA, 2010. ACM.

[21] Amit Nathani, Sanjay Chaudhary, and Gaurav Somani. Policy based resource allocation in iaas cloud. *Future Generation Computer Systems*, 28(1):94 – 103, 2012.

[22] Adrian Paschke and Martin Bichler. Knowledge representation concepts for automated SLA management. *Decision Support Systems*, 46(1):187–205, 2008.

[23] Vinicius Petrucci, Orlando Loques, and Daniel Mossé. A dynamic optimization model for power and performance management of virtualized clusters. In *e-Energy '10*, pages 225–233, New York, NY, USA, 2010. ACM.

[24] Jia Rao, Xiangping Bu, Cheng-Zhong Xu, Leyi Wang, and George Yin. Vconf: a reinforcement learning approach to virtual machines auto-configuration. In *ICAC '09*, pages 137–146, New York, NY, USA, 2009. ACM.

[25] Mark Stillwell, David Schanzenbach, Frederic Vivien, and Henri Casanova. Resource allocation algorithms for virtualized service hosting platforms. *Journal of Parallel and Distributed Computing*, 70(9):962 – 974, 2010.

[26] H. Viswanathan, E.K. Lee, I. Rodero, D. Pompili, M. Parashar, and M. Gamell. Energy-aware application-centric vm allocation for hpc workloads. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 890 –897, may 2011.

[27] William Voorsluys, James Broberg, and Srikumar Venugopal. Cost of virtual machine live migration in clouds: A performance evaluation.

[28] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. Sandpiper: Black-box and gray-box resource management for virtual machines. *Computer Networks*, 53(17):2923 – 2938, 2009.

[29] Y.O. Yazir, C. Matthews, R. Farahbod, S. Neville, A. Guitouni, S. Ganti, and Y. Coady. Dynamic resource allocation in computing clouds using distributed multiple criteria decision analysis. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 91 –98, 2010.