Enacting SLAs in Clouds Using Rules

Michael Maurer[1], Ivona Brandic[1], and Rizos Sakellariou[2]

[1] Vienna University of Technology, Distributed Systems Group,
Argentinierstraße 8, 1040 Vienna, Austria
{maurer,ivona}@infosys.tuwien.ac.at
[2] University of Manchester, School of Computer Science, U.K.
rizos@cs.man.ac.uk

Abstract. The emergence of Cloud Computing raises the question of dynamically allocating resources of physical (PM) and virtual machines (VM) in an on-demand and autonomic way. Yet, using Cloud Computing infrastructures efficiently requires fulfilling three partially contradicting goals: first, achieving low violation rates of Service Level Agreements (SLA) that define non-functional goals between the Cloud provider and the customer; second, achieving high resource utilization; and third achieving the first two issues by as few time- and energy consuming reallocation actions as possible. To achieve these goals we propose a novel approach with escalation levels to divide all possible actions into five levels. These levels range from changing the configuration of VMs over migrating them to other PMs to outsourcing applications to other Cloud providers. In this paper we focus on changing the resource configuration of VMs in terms of storage, memory, CPU power and bandwidth, and propose a knowledge management approach using rules with threat thresholds to tackle this problem. Simulation reveals major improvements as compared to recent related work considering SLA violations, resource utilization and action efficiency, as well as time performance.

1 Introduction

One of the main challenges Cloud Computing providers face is the question of dynamically allocating resources in an on-demand way. *Service Level Agreements* (SLAs) settle non-functional requirements between the Cloud Computing providers and their customers. These SLAs contain Quality of Service (QoS) goals, which are expressed as, e.g., "storage $\geq 1000GB$". Penalties that have to be paid to the customers in case these goals are violated are also part of the SLA. Thus, Cloud Computing providers face three contradicting challenges: First, they aim for providing enough resources for every application. Second, they try to efficiently use their resources and only allocate what applications currently really need. Third, they consider energy consumption of reallocation actions and strive for an efficient usage of these executed actions.

In [10] we presented Case Based Reasoning (CBR) for decision making in the MAPE-K (Monitoring, Analysis, Planning, Execution, Knowledge) cycle of an autonomic SLA enactment environment in Clouds. We evaluated it by a generic simulation engine we developed and showed the suitability of CBR for resource-efficient SLA management. However, we also determined some drawbacks of CBR as far as its learning performance and its scalability were concerned. Therefore, in this paper we design and implement a rule-based knowledge management (KM) approach, and utilize the same simulation engine enhanced by more accurate and general utility functions to evaluate it and reevaluate CBR. Using rules we attempt to improve not only SLA adherence and resource allocation efficiency as discussed in [10], but also new aspects, i.e., the efficient use of reallocation actions, as well as scalability. Additionally, we adapt a wholesome view of different adaptation levels like virtual machine (VM) configuration or VM migration, and propose a hierarchical model of so called *escalation levels* for dynamically and efficiently managing resource allocation for Cloud Computing infrastructures.

The challenge in this work is to evaluate KM techniques for autonomic SLA enactment in Cloud Computing infrastructures that fulfill the three following conflicting goals: (i) achieving low SLA violation rates; (ii) achieving high resource utilization such that the level of allocated but unused resources is as low as possible; and (iii) achieving (i) and (ii) by as few time- and energy consuming reallocation actions as possible. We will call this problem *resource allocation problem* throughout the rest of the paper.

The main contributions of this paper are:

1. partitioning the resource allocation problem for Cloud infrastructures into several subproblems by proposing *escalation levels* that structure all possible reaction possibilities into different subproblems using a hierarchical model.
2. designing, implementing and evaluating a rule-based approach to propose a solution for one of the subproblems presented in 1), i.e., for *virtual machines* in Cloud infrastructures, and comparing it to the CBR approach.

2 Related Work

Concerning related work, we have determined two different ways to compare our work with other achievements in this area. Whereas the first level compares other works dealing with SLA enactment and resource efficiency, the second one considers the area of knowledge management.

At first, considerable work on optimizing resource usage while keeping QoS goals has been conducted. One general main difference to our approach consists of the fact that related work examines either only certain subsystems of large-scale distributed systems, as [8] the performance of memory systems, or constrain themselves to one or two specific SLA parameters. Whereas Petrucci et al. [14] or Bichler et al. [4] consider one general resource constraint, Khanna et al. [2] only investigate response time and throughput, and Kalyvianaki [6] CPU usage. [5,17] examine specific use cases of web servers deployed in Cloud-like environments by investigating horizontal scaling of servers. The Sandpiper framework [18], which offers black-box and gray-box resource management for VMs, provides a quite similar approach to ours. Contrary to our project, though, Sandpiper plans reactions just after violations have occurred. Also the VCONF model by Rao et al. [15] has similar goals as presented in Section 1, but depends on specific parameters, can only execute one action per iteration and neglects the energy consumption of executed actions. Other papers focus on different escalation levels (as described in Section 3). [19,12] focus on VM migration and [11] on turning on and off physical machines, whereas our paper focuses on VM re-configuration.

Secondly, there has been work on KM of SLAs, especially rule-based systems. Paschke et al. [13] look into a rule based approach in combination with the logical formalism ContractLog. It specifies rules to trigger after a violation has occurred, e.g., it obliges the provider to pay some penalty, but it does not deal with avoidance of SLA violations. Others inspected the use of ontologies as knowledge bases (KBs) only at a conceptual level. Koumoutsos et al. [9] view the system in four layers (i.e., business, system, network and device) and break down the SLA into relevant information for each layer, but give no implementation details. Bahati et al. [3] 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. A bad region would be, e.g., response time > 500 (too slow), fair region response time < 100 (too fast, consuming unnecessary resources) and a good region $100 \leq$ response time ≤ 500. 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. Moreover, our paper provides a more wholesome approach than related work and integrates the different action levels that work has been carried out on.

3 Escalation Levels

This section presents a methodology of dividing the resource allocation problem into smaller subproblems using a hierarchical approach. It demonstrates which actions can be executed in what level to achieve SLA adherence and efficient resource allocation for Cloud infrastructures. We call these levels *escalation levels* and present them in Table 1. The idea is that every problem that occurs should be solved on the lowest escalation level. Only if this is not possible, the problem is tried to be solved on the next level, and again, if this fails, on the next one, and so on. The levels are ordered in a way such that lower levels offer faster and more local solutions than higher ones. The first escalation level (“change VM configuration”) works locally on a PM and tries to change the amount of storage or memory, e.g., that is allocated to the VM from the PM resources. Then, migrating applications (escalation level 2) is more light-weight than migrating VMs and turning PMs on/off (escalation levels 3 and 4). For all three escalation levels already the whole system state has to be taken into account to find an optimal solution. The problem stemming from escalation level 3 alone can be formulated into a *Binary integer problem* (BIP), which is known to be NP-complete [7]. The proof is out of scope for this paper, but a similar approach can

Table 1. Escalation levels

1. Change VM configuration.
2. Migrate applications from one VM to another.
3. Migrate one VM from one PM to another or create new VM on appropriate PM.
4. Turn on / off PM.
5. Outsource to other Cloud provider.

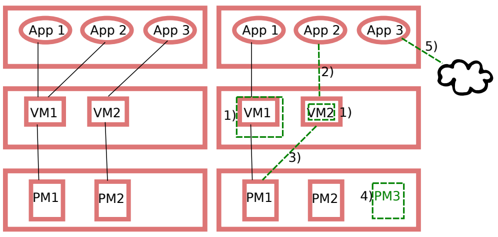
be seen in [14]. The last escalation level has least locality and greatest complexity, since the capacity of other Cloud infrastructures have to be taken into account too, and negotiations have to be started with them as well.

Also the rule-based approach benefits from this hierarchical action level model, because it provides a salience concept for contradicting rules. Without this concept it would be troublesome to determine which of the actions, e.g., "Power on additional PM with extra-storage and migrate VM to this PM", "Increase storage for VM by 10%" or "Migrate application to another VM with more storage" should be executed, if a certain threshold for allocated storage has been exceeded. The proposed rule-based approach will present a solution for escalation level 1.

Figure 1 visualizes the escalation levels from Table 1 in the context of Infrastructure as a Service (IaaS) before and after actions are executed. Figure 1(a) shows applications *App1* and *App2* deployed on *VM1* that is itself deployed on *PM1*, whereas *App3* runs on *VM2* running on *PM2*. Figure 1(b) shows example actions for all five escalation levels. The legend numbers correspond to the respective numbering of the escalation levels.

- *Escalation level 1*: At first, the autonomic manager tries to change VM configuration. Actions 1) show *VM1* being up-sized and *VM2* being down-sized.
- *Escalation level 2*: If the attempt to increase a certain resource for a VM in escalation level 1 fails, because some resource cannot be increased anymore due to the constraints of the PM hosting the VM, in level 2 the autonomic manager tries to migrate the application to another larger VM that fulfills the required specifications from level 1. So if, e.g., provided storage needs to be increased from 500 to 800GB, but only 200 GB are available on the respective VM, then the application has to be migrated to a VM that has at least the same resources as the current one plus the remaining 100GB of storage. Action 2) shows the re-deployment of *App2* to *VM2*. Due to possible confinements of some applications to certain VMs, e.g., a user deployed several applications that need to work together on one VM, this escalation might be skipped in some scenarios.
- *Escalation level 3*: If there is no appropriate VM available in level 2, in level 3 the autonomic manager tries to create a new VM on an appropriate PM or migrate the VM to a PM that has enough available resources. Action 3) shows the re-deployment of *VM2* to *PM1*.

– *Escalation level 4*: Again, if there is no appropriate PM available in level 3, the autonomic manager suggests turning on a new PM (or turning it off if the last VM was emigrated from this PM) in level 4. Action 4) shows powering on a new PM (*PM3*).
– *Escalation level 5*: Finally, the last escalation level 5 tries to outsource the application to another Cloud provider as explained, e.g., in the Reservoir project [16]. Action 5) outsources *App3* to another Cloud provider.



(a) Before action execution (b) After action execution

Fig. 1. Actions used in 5 escalation levels: before and after action execution

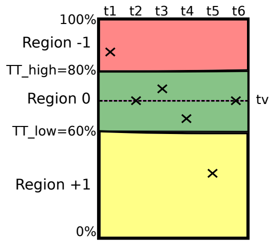


Fig. 2. Example behavior of actions at time intervals t1-t6

4 Rule-Based Approach for VM Level

This section describes the rule-based approach for escalation level 1.

4.1 Prerequisites

For resource management, we need to define how the *measured*, *provided* and *agreed* values interrelate, and what an SLA violation actually is [10]. The *measured* value (1) represents the amount of a specific resource that is currently used by the customer. The amount of *allocated* (2) resource determines to what extent a specific resource can be used by the customer, i.e., how much of the resource is allocated to the VM hosting the application. The *agreed* value (3) corresponds to the Service Level Objective (SLO) agreed in the SLA. An *SLA violation* occurs, if less is provided (2) than the customer utilizes (or wants to utilize) (1) with respect to the limits set in the SLA (3).

Dealing with SLA-bound resource management, where resource usage is paid for on a "pay-as-you-go" basis with SLOs that guarantee a minimum capacity of these resources as described above, raises the question, whether the Cloud provider should allow the consumer to use more resources than agreed. We will refer to this behavior as *over-consumption*. Since the consumer will pay for every additional resource, it should be in the Cloud provider's interest to allow over-consumption as long as this behavior does not endanger the SLAs of other consumers. Thus, Cloud providers should not allow over-consumption when the

Table 2. Resource policy modes

green	Plenty of resources left. Over-consumption allowed.
green-orange	Heavy over-consumption is forbidden. All applications that consume more than τ% (threshold to be specified) of the agreed resource SLO are restrained to $\tau/2$% over-consumption
orange	Resource is becoming scarce, but SLA demand can be fulfilled if no over-consumption takes place. Thus, over-provisioning is forbidden.
orange-red	Over-provisioning forbidden. Initiate outsourcing of some applications.
red	Over-provisioning forbidden. SLA resource requirements of all consumers cannot be fulfilled. If possible, a specific choice of applications is outsourced. If not enough, applications with higher reputation points or penalties are given priority over applications with lower reputation points / penalties. SLAs of latter ones are deliberately broken to ensure SLAs of former ones.

resulting penalties they have to pay are higher than the expected revenue from over-consumption. To tackle this problem, we introduce five policy modes for every resource that describe the interaction of the five escalation levels. As can be seen in Table 2 the policy modes are green, green-orange, orange, orange-red and red. They range from low utilization of the system with lots of free resources left (policy mode green) over a scarce resource situation (policy mode orange) to an extreme tight resource situation (policy mode red), where it is impossible to fulfill all SLAs to its full extent and decisions have to be made which SLAs to deliberately break and which applications to outsource.

4.2 Design and Implementation

In order to know whether a resource r is in danger of under-provisioning or already is under-provisioned, or whether it is over-provisioned, we calculate the current utilization $ut^r = \frac{\text{use}^r}{\text{pr}^r} \times 100$, where use^r and pr^r signify how much of a resource r was used and provided, respectively, and divide the percentage range into three regions by using the two "threat thresholds" TT^r_{low} and TT^r_{high}:

- Region -1: Danger of under-provisioning, or under-provisioning ($> TT^r_{high}$)
- Region 0: Well provisioned ($\leq TT^r_{high}$ and $\geq TT^r_{low}$)
- Region $+1$: Over-Provisioning ($< TT^r_{low}$)

The idea of this rule-based design is that the ideal value that we call *target value* $tv(r)$ for utilization of a resource r is exactly in the center of region 0. So, if the utilization value after some measurement leaves this region by using more (Region -1) or less resources (Region +1), then we reset the utilization to the target value, i.e., we increase or decrease allocated resources so that the utilization is again at

$$tv(r) = \frac{TT^r_{low} + TT^r_{high}}{2}\%.$$

As long as the utilization value stays in region 0, no action will be executed. E.g., for $r =$ storage, $TT^r_{low} = 60\%$, and $TT^r_{high} = 80\%$, the target value would be $tv(r) = 70\%$. Figure 2 shows the regions and measurements (expressed as utilization of a certain resource) at time steps $t_1, t_2, \ldots, t_6$. At t_1 the utilization of the resource is in Region -1, because it is in danger of a violation. Thus, the KB recommends to increase the resource such that at the next iteration t_2 the utilization is at the center of Region 0, which equals the target value. At time steps t_3 and t_4 utilization stays in the center region and consequently, no action is required. At t_5, the resource is under-utilized and so the KB recommends the decrease of the resource to $tv(r)$, which is attained at t_6. Additionally, if over-provisioning is allowed in the current policy mode, then the adjustment will always be executed as described regardless of what limit was agreed in the SLA. On the other hand, if over-provisioning is not allowed in the current policy mode, then the rule will allocate at most as much as agreed in the SLA (SLO^r).

The concept of a rule increasing resource r is depicted in Figure 3. The rule executes if the current utilization ut^r and the predicted utilization $ut^r_{predicted}$ of the next iteration (cf. next paragraph) both exceed TT^r_{high} (line 2). Depending on what policy level is active the rule either sets the provided resource pr^r to the target value $tv(r)$ for policy levels green and green-orange (line 3) or to at most what was agreed in the SLA (SLO^r) plus a certain percentage ϵ to account for rounding errors when calculating the target value in policy levels orange, orange-red and red (line 5). A similar rule scheme for decreasing a resource can be seen in Figure 4. The main difference is that it does not distinguish between policy modes and that it sets the provisioned resource to at least a minimum value $minPr^r$, which may be 0, that is needed to keep the application alive (line 4). The rule is executed if the current utilization ut^r and the predicted utilization $ut^r_{predicted}$ of the next iteration both lie below TT^r_{low} (line 2).

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. However, to further reduce the risk of oscillations, we suggest to calculate a prediction for the next value based on the latest measurements. Thus, an action is only invoked when the current AND the predicted measurement exceed the respective TT. So, especially when only one value exceeds the TT, no action is executed.

```
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 green, green-orange.
5   Set $pr^r$ to $\min(\frac{\text{use}^r}{tv(r)}, SLO^r * (1+\epsilon/100))$ for policy modes orange, orange-red, red.
```

Fig. 3. 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 $\max(\frac{\text{use}^r}{tv(r)}, minPr^r)$.
```

Fig. 4. Rule scheme for decreasing a resource

The rules have been implemented using the Java rule engine Drools [1]. The Drools engine sets up a knowledge session consisting of different rules and a working memory. Rules get activated when specific elements are inserted into the working memory such that the conditional "when" part evaluates to true. Activated rules are then triggered by the simulation engine. In our case, the simulation engine inserts measurements and SLAs of applications into the working memory. 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. As opposed to the CBR based approach in [10], the rule-based approach is able to fire more than one action at the same iteration, which inherently increases the flexibility of the system. Without loss of generality we can assume that one application runs on one VM (several applications' SLAs can be aggregated to form one VM SLA) and we assume the more interesting case of policy modes orange, orange-red or red, where over-provisioning is not allowed.

5 Evaluation

In this section we evaluate the quality of the proposed rule-based approach measured by a utility function, as well as its time performance and scalability.

5.1 Utility-Driven Evaluation

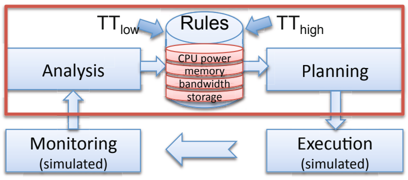


Fig. 5. Simulation engine evaluating a rule-based knowledge management system

We evaluated the rule-based approach for escalation level 1 with the simulation engine described in [10]. This simulation engine simulates measurements of SLA parameters for an arbitrary number of VMs, forwards them to the KB, asks the KB for appropriate actions, and simulates the execution of these actions; thus it traverses the complete MAPE cycle in one iteration as depicted in Figure 5. As resources for IaaS one can use all parameters that can be adapted on a VM. For the evaluation we chose to take the following parameters and SLOs: *storage* $\geq$ 1000GB, *incoming bandwidth* $\geq$ 20 Mbit/s, *outgoing bandwidth* $\geq$ 50 Mbit/s, *memory* $\geq$ 512 MB, and *CPU power* $\geq$ 100 MIPS (Million Instructions Per Second).

The simulation engine keeps track of the SLA of every VM, the violations thereof, resource utilization and costs of action execution. All evaluations for this subsection are executed with 100 iterations and 500 applications. We investigate low, middle and high values for TT^r_{low} and TT^r_{high}, where $TT^r_{\text{low}} \in \{30\%, 50\%, 70\%\}$ and $TT^r_{\text{high}} \in \{60\%, 75\%, 90\%\}$ for all resources stated above. We combine the TTs to form eight different scenarios as depicted in Table 3. The workload follows an (increasing or decreasing) trend for an a-priori unknown period of time and different for every resource. As the intensity of the trend varies for every iteration, the simulation comprises both, slow developments and rapid changes, and thus simulates workload in a quite general way.

To be able to compare the utility of the individual threshold pairs, we define a generic cost function that maps SLA violations, resource wastage and the costs of

Table 3. 8 Simulations Scenarios for TT_{low} and TT_{high}

	Scenarios							
	1	2	3	4	5	6	7	8
TT_{low}	30%	30%	30%	50%	50%	50%	70%	70%
TT_{high}	60%	75%	90%	60%	75%	90%	75%	90%

executed actions into a monetary unit, which we want to call *Cloud EUR.* First, we define a penalty function $\mathfrak{p}^r(p) : [0, 100] \to \mathbb{R}^+$ that defines the relationship between the percentage of violations p (as opposed to all possible violations) and the penalty for a violation of resource r. Second, we define a function wastage $\mathfrak{w}^r(w) : [0, 100] \to \mathbb{R}^+$ that relates the percentage of unused resources w to the energy in terms of money that these resources unnecessarily consume. Third, we define a cost function $\mathfrak{a}^r(a) : [0, 100] \to \mathbb{R}^+$ from the percentage of executed actions a (as opposed to all possible actions that could have been executed) to the energy and time costs in terms of money. The total cost c is then defined as

$$c(p, w, c) = \sum_r \mathfrak{p}^r(p) + \mathfrak{w}^r(w) + \mathfrak{a}^r(a). \tag{1}$$

We assume functions $\mathfrak{p}^r$, $\mathfrak{w}^r$ and $\mathfrak{a}^r$ for this evaluation with $\mathfrak{p}^r(p) = 100p$, $\mathfrak{w}^r(w) = 5w$, and $\mathfrak{a}^r(a) = a$ for all r. The intention behind choosing these functions is (i) to impose very strict fines in order to proclaim SLA adherence as top priority, (ii) to weigh resource wastage a little more than the cost of actions.

In Figure 6 we compare the outcome of the rule-based approach evaluating the aforementioned eight scenarios. From Figure 6(a) we see that in terms of SLA violations Scenario 1 achieves the best result, where only 0.0908% of all possible violations occur, and the worst result with Scenario 8, with a still very low violation rate of 1.2040%. In general, the higher the values are for TT_{high}, the worse is the outcome. The best result achieved with CBR was at 7.5%.

Figure 6(b) shows resource utilization. We see that the combination of high TT_{low} and high TT_{high} (Scenario 8) gives the best utilization (83.98%), whereas low values for TT_{low} and TT_{high} lead to the worst utilization (62.03% in Scenario 1). Still, compared to CBR which scored a maximum of 80.36% and a minimum of 51.81%, the rule-based approach generally achieves better results. Also, when comparing resource allocation efficiency (RAE), which is defined as

$$\text{RAE} = \frac{u}{v+1}, \tag{2}$$

where u is the average utilization over all resources and v is the number of violations, the rule-based approach achieves a maximum of 795.9 and a minimum of 69.8 (see Figure 6 (e)), whereas CBR achieves 10.0 at most.

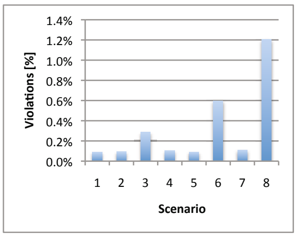
The percentage of all executed actions as compared to all possible actions that could have been executed is shown in Figure 6(c). One observes that the greater the span between TT_{low} and TT_{high} is, the less actions have to be executed. Most actions (60.75%) are executed for Scenario 7 (span of only 5% between TT values), whereas least actions (5.44%) are executed for Scenario 3 (span of 60%

between TT values). CBR almost always recommended exactly one action and hardly ever (in about 1% of the cases) recommended no action.

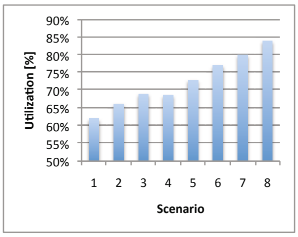
Figure 6(d) shows the costs for each scenario using Equation 1. The best trade-off between the three terms is achieved by Scenario 5 that has medium values for TT^r_{low} and TT^r_{high}. It has a very low violation rate of 0.0916%, a quite elaborate utilization of 72.90%, but achieves this with only 19.79% of actions. Scenario 7 achieves a better violation and utilization rate but at the cost of an action rate of 60.75%, and consequently has higher costs. The lowest cost value for CBR is 923 Cloud EUR, the highest 2985 Cloud EUR.

If the utility of the decision decreases for a certain time frame (as cost increases), the KB could determine the cost summand in Equation 1 that contributes most to this decrease. For any resource r, if the term is $\mathfrak{p}$, then decrease TT^r_{high}. If the term is $\mathfrak{w}$, then increase TT^r_{low}. Otherwise, if the term is $\mathfrak{c}$, then widen the span of TT^r_{high} and TT^r_{low}, i.e., increase TT^r_{high} and decrease TT^r_{low}. We plan to investigate this in our future research.

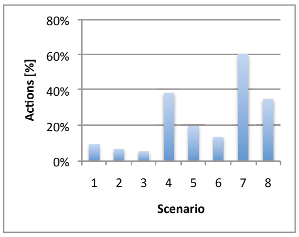
Summarizing, we have seen that in all 8 scenarios the proposed approach outperforms the CBR approach with respect to the SLA violation rate (up to 82 times better results) and the resource allocation efficiency (up to 80 times better results). 7 out of 8 scenarios achieved better results in terms of actions needed and were better than the worst CBR value for utilization, whereas only one scenario was better than the best CBR utilization value. However, accumulating these results into cost, all rule-based scenarios outperform CBR by a factor of at least 4 (worst rule-based scenario (236) compared to best CBR result (923)), which to a large extent is due to the huge number of violations that the rule-based approach is able to prevent and the high number of actions it can save.



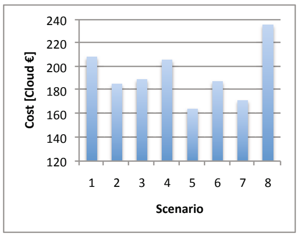
(a) Violations



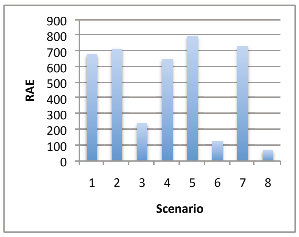
(b) Utilization



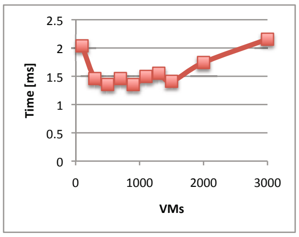
(c) Actions



(d) Cost



(e) Resource allocation efficiency



(f) Average execution time per VM

Fig. 6. Violations, Utilization, Actions and Utility for Scenarios 1-8, Execution time

5.2 Performance-Driven Evaluation

As far as time performance and scalability is concerned, the performance tests are very encouraging. We executed 100 iterations from 100 to 3000 VMs. We performed every test twice and calculated the average execution time as well as the average time it took for the simulation engine to handle one VM. As shown in Figure 6(f) the execution time per VM stays quite constant for up to 1500 VMs, and thus average execution time is about linear. For 3000 VMs, it took $647s/100 = 6.47s$ for one iteration to treat all VMs. The high time consumption per VM for 100 VMs in Figure 6(f) is due to the initialization of the rule knowledge base which takes over-proportionally long for just a small number of VMs and does not weigh so much for more VMs.

CBR took 240s for 50VMs and 20 iterations. Thus, CBR took $240s/20 = 12s$ for one iteration to treat all VMs, which is twice as long as the rule-based approach takes, which even has 60 times more VMs. However, CBR implements learning features, what the rule-based currently does not, and could be sped up by choosing only specific cases to be stored in the KB.

6 Conclusion and Outlook

This paper structured the set of possible actions to govern Cloud infrastructures into five escalation levels from changing the configuration of virtual machines over migrating them to other physical machines to outsourcing applications to other Cloud providers. A use case has been developed together with resource policy modes that govern the high-level behavior of Cloud infrastructures. We developed a rule-based knowledge management approach to tackle the first of the five escalation levels: dynamic adaptation of VM configuration in an energy-efficient way. We proposed a rule-based approach and showed that it had several advantages over an approach using Case Based Reasoning (CBR). We tested the rule-based approach using 8 scenarios that differed in the threat thresholds employed to mark the limits between “regular performance”, over- and under-utilization. In almost all scenarios, we gained even better results with the rule-based approach than with CBR. In the future we want to ameliorate the cost functions by relating them to real-world measurements of energy consumption and with it learn and adjust the high and low threat thresholds.

Acknowledgments. The work described in this paper is supported by the Vienna Science and Technology Fund (WWTF) under grant agreement ICT08-018 Foundations of Self-Governing ICT Infrastructures (FoSII) and by COST-Action IC0804 on Energy Efficiency in Large Scale Distributed Systems.

References

1. Drools, `http://www.drools.org`
2. Application Performance Management in Virtualized Server Environments (2006), `http://dx.doi.org/10.1109/NOMS.2006.1687567`

3. Bahati, R.M., Bauer, M.A.: Adapting to run-time changes in policies driving autonomic management. In: ICAS 2008: Proceedings of the 4th Int. Conf. on Autonomic and Autonomous Systems. IEEE Computer Society, Washington, DC, USA (2008)
4. Bichler, M., Setzer, T., Speitkamp, B.: Capacity Planning for Virtualized Servers. Presented at Workshop on Information Technologies and Systems (WITS), Milwaukee, Wisconsin, USA (2006)
5. Dutreilh, X., Rivierre, N., Moreau, A., Malenfant, J., Truck, I.: From data center resource allocation to control theory and back. In: 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD), 2010, pp. 410–417 (July 2010)
6. Kalyvianaki, E., Charalambous, T., Hand, S.: Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters. In: Proceedings of the 6th International Conference on Autonomic Computing, ICAC 2009, pp. 117–126. ACM, New York (2009), http://doi.acm.org/10.1145/1555228.1555261
7. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computations: Proc. of a Symp. on the Complexity of Computer Computations, pp. 85–103. Plenum Press (1972)
8. Khargharia, B., Hariri, S., Yousif, M.S.: Autonomic power and performance management for computing systems. Cluster Computing 11(2), 167–181 (2008)
9. Koumoutsos, G., Denazis, S., Thramboulidis, K.: SLA e-negotiations, enforcement and management in an autonomic environment. In: Modelling Autonomic Communications Environments, pp. 120–125 (2008)
10. Maurer, M., Brandic, I., Sakellariou, R.: Simulating autonomic SLA enactment in clouds using case based reasoning. In: Di Nitto, E., Yahyapour, R. (eds.) ServiceWave 2010. LNCS, vol. 6481, pp. 25–36. Springer, Heidelberg (2010)
11. Mazzucco, M., Dyachuk, D., Deters, R.: Maximizing cloud providers' revenues via energy aware allocation policies. In: CLOUD 2010, pp. 131–138 (2010)
12. Meng, X., Isci, C., Kephart, J., Zhang, L., Bouillet, E., Pendarakis, D.: Efficient resource provisioning in compute clouds via VM multiplexing. In: Proceeding of the 7th International Conference on Autonomic Computing, ICAC 2010, pp. 11–20. ACM, New York (2010), http://doi.acm.org/10.1145/1809049.1809052
13. Paschke, A., Bichler, M.: Knowledge representation concepts for automated SLA management. Decision Support Systems 46(1), 187–205 (2008)
14. Petrucci, V., Loques, O., Mossé, D.: A dynamic optimization model for power and performance management of virtualized clusters. In: e-Energy 2010, pp. 225–233. ACM, New York (2010)
15. Rao, J., Bu, X., Xu, C.-Z., Wang, L., Yin, G.: Vconf: a reinforcement learning approach to virtual machines auto-configuration. In: ICAC 2009, pp. 137–146. ACM, New York (2009), http://doi.acm.org/10.1145/1555228.1555263
16. Rochwerger, B., et al.: The RESERVOIR model and architecture for open federated cloud computing. IBM Journal of Research and Development 53(4) (2009), http://www.research.ibm.com/journal/rd/534/rochwerger.pdf
17. Singh, R., Sharma, U., Cecchet, E., Shenoy, P.: Autonomic mix-aware provisioning for non-stationary data center workloads. In: ICAC 2010, pp. 21–30. ACM, New York (2010), http://doi.acm.org/10.1145/1809049.1809053
18. Wood, T., Shenoy, P., Venkataramani, A., Yousif, M.: Sandpiper: Black-box and gray-box resource management for virtual machines. Computer Networks 53(17), 2923–2938 (2009)
19. Yazir, Y.O., Matthews, C., Farahbod, R., Neville, S., Guitouni, A., Ganti, S., Coady, Y.: Dynamic resource allocation in computing clouds using distributed multiple criteria decision analysis. In: 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD), pp. 91–98 (2010)