2014 Sixth International Conference on Computational Intelligence, Communication Systems and Networks

Cross-Correlation Prediction of Resource Demand for Virtual Machine Resource Allocation in Clouds

Dorian Minarolli and Bernd Freisleben
Department of Mathematics and Computer Science, University of Marburg
Hans-Meerwein-Str. 3, D-35032 Marburg, Germany
{minarolli, freisleb}@informatik.uni-marburg.de

***Abstract*—Cloud computing is aimed at offering elastic resource allocation on demand in a pay-as-you-go fashion to cloud consumers. To achieve this goal in automatic manner, a resource scaling mechanism is needed that maintains application performance according to Service Level Agreements (SLA) and reduces resource costs at the same time. In this paper, we present a cross-correlation prediction approach based on machine learning that predicts resource demands of multiple resources of virtual machines running in a cloud infrastructure. Based on these predictions, a proactive resource allocation scheme is applied that assigns only the required resources to virtual machines to keep their cost to a minimum. Experimental results with the web serving multi-tier application benchmark of CloudSuite show the effectiveness of our approach compared to a non-cross-correlation prediction technique in achieving better prediction accuracy and better application performance.**

Keywords–cloud computing; machine learning; resource prediction; virtual machine

I. Introduction

Cloud computing can be used to allocate computational resources on demand in a pay-as-you-go fashion to cloud consumers. This makes it possible to keep application performance high and to reduce resource costs for both cloud consumers and cloud providers. One of the popular cloud service models is Infrastructure-as-a-Service (IaaS) where computational resources are provided to cloud consumers in the form of virtual machines (VMs) [1]. VMs offer the flexibility to assign a certain amount of resources, such as CPU or memory, to an application and change the assignment at runtime.

Many cloud providers assign resources to VMs statically according to the peak demands. This static allocation leads to a waste of resources, higher costs and inefficient use of computational resources, probably leaving some applications starving for resources and others having more resources than needed. A better way is to allocate resources in an elastic manner dynamically according to the current demand, by just assigning the minimum amount of resources required to satisfy application performance. However, this is a challenging task, since resource demand is dynamic and changes quickly over time. Ideally, what is required is a proactive resource allocation approach that predicts resource usage in advance and allocates resources according to the current demand in an automatic way without human intervention.

In this paper, we present an approach called Automatic Proactive Resource Allocation (APRA) that proactively allocates resources to VMs by predicting resource usage demand using Support Vector Machine (SVM) for time series forecasting. Since there are interdependencies between VMs of a multi-tier application, and the resource usages of resources of multiple VMs of the same application are correlated with each other, we apply cross-correlation prediction by making predictions for multiple resources of a multi-tier application at the same time. We focus on two resources to be assigned to VMs, namely CPU and memory, since current virtualization technologies do not offer adequate support for the allocation of other resources, such as network or disk bandwidth. Based on our predictions for resource usage for the next interval, the proposed approach allocates the most adequate amount of resources to VMs to satisfy the demand. The approach is general and non-intrusive, since its resource allocation decisions are based on resource usage metrics monitored outside of the VMs, and are independent of the type of application. Experimental results of the evaluation of a multi-tier application cloud environment testbed running web serving benchmarks of CloudSuite [4] show better application performance and prediction accuracy than a non-cross-correlation approach for resource demand prediction.

The paper is organized as follows. Section II discusses related work. Section III describes the APRA design. Section IV presents implementation issues, and Section V discusses experimental results. Section VI concludes the paper and outlines areas for future work.

II. Related Work

Several solutions have been proposed for VM resource allocation, such as: a) feed-back control, b) queueing theory, c) machine learning and d) proactive resource allocation.

Feed-back control [7], [11], [14] approaches achieve resource allocation to VMs by keeping certain performance metrics to the desired level. Wang *et al.* [14] developed an approach based on feed-back control for web server CPU partition sizing to lower resource waste and keep response times to acceptable levels. Later on, they extended the approach to include also memory allocation [7] in

978-1-4799-5076-8/14 $31.00 © 2014 IEEE
DOI 10.1109/CICSyN.2014.36

virtualized environments by creating joint CPU and memory controllers. Padala *et al.* [11] similarly use a control-theory feed-back controller to allocate CPU resources to VMs by considering the allocation of several VMs of a multi-tier application. In contrast to the above works that are reactive allocation approaches, we apply proactive resource allocation by predicting VM resource demand and allocate resources accordingly.

Queueing theory [2], [3], [10] approaches apply application performance modelling based on queueing theory to allocate resources and manage performance. Doyle *et al.* [3] propose an approach for CPU, memory and disk I/O allocation to web service applications based on analytical performance modelling. Bennani and Menasce [2] present an approach that applies autonomic computing principles to allocate computational resources to application environments for satisfying their SLA performance metrics by optimizing a global utility function. Later, they extended the work [10] in order to apply it in virtualized environments. The problem with the above approaches is the difficulty of building accurate queueing models of the complex behaviour of applications in virtualized environments. In contrast, our approach does not require to explicitely build any model but is based on resource demand prediction for resource allocation, which is a more general approach.

Machine learning [12], [15], [16] approaches apply machine learning for determining resource allocations to VMs. Wildstrom *et al.* [15] present an approach for CPU and memory resources reconfiguration according to workload changes. The authors apply a propositional rule learner to build an off-line model for predicting application performance given as input resource configurations and low-level system metrics. Rao *et al.* [12] present an approach for dynamic VM resource reconfiguration based on reinforcement learning where an agent interacts with the environment by trying different resource allocation actions to learn an optimal policy. Later, the authors extended the work by developing a unified reinforcement learning approach [16] that combines two reinforcement learning agents, one that applies reinforcement learning for resource reconfiguration of all VMs and the other that applies reinforcement learning for reconfiguration of application parameters. The drawback with reinforcement learning approaches is their poor scalability due to a large state space and long convergence time to an optimal policy, making them impractical in real environments.

Proactive resource allocation [5], [8], [9] approaches apply VM resource allocation based on resource demand prediction. Gong *et al.* [5] developed the PRedictive Elastic reSource Scaling (PRESS) system for dynamic fine-grained resource allocation to VMs in order to reduce resource costs and avoid application SLA performance violations. It achieves this by predicting VM resource demand in the near future using signal processing and machine learning techniques. Islam *et al.* [9] propose and experiment with resource prediction techniques based on machine learning, such as linear regression and artificial neural networks. Although the authors argue that their techniques can support VM allocation, they did not integrate them into any allocation method. Hu *et al.* [8] present KSwSVR, a system for fine-grained VM resource allocation that makes prediction of the VM resource demand multiple steps ahead in the future based on the combination of an improved version of an SVM and a Kalman filter. The proposed proactive VM resource allocation solutions have the drawback that they predict the demand of only one resource or of multiple resources separately without taking into account cross-correlation between resources. Our proposed proactive resource allocation approach makes predictions of multiple resource demands simultaneously, taking into account cross-correlation between resources of multiple VMs of the same multi-tier application.

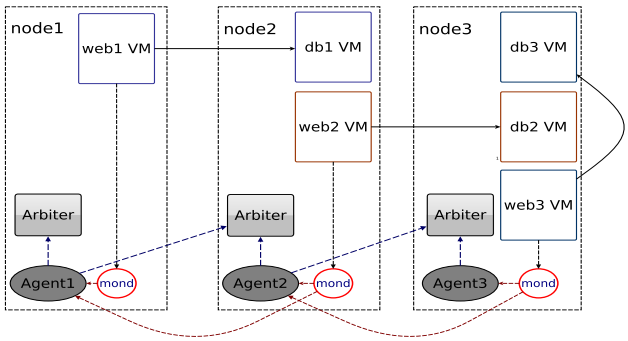


Figure 1. APRA architecture.

III. Automatic Proactive Resource Allocation

In this section, we present the design of our APRA approach by starting with an overview of its architecture and its main components. Then, we continue with describing the application of the SVM approach for cross-correlation prediction of resource demand. Finally, we present how the resource allocation decision to multiple resources is made based on resource demand prediction.

A. APRA Architectural Overview

The architecture of APRA is shown in Fig. 1. It is applied in an IaaS cloud environment composed of several physical nodes on top of which several VMs are running. Each VM is running one tier of a multi-tier application. Fig. 1 shows the case of three physical nodes and three two-tier applications running in total six VMs spread randomly over the physical nodes. The components of APRA are as follows.

In each physical node, a monitoring daemon called *mond* is running that is responsible for gathering, in each time interval, resource usage metrics of all VMs running on the corresponding physical node. Each multi-tier application has

a management module called *Agent* that is responsible for the resource allocation of its own VMs. The *Agent* does not necessarily have to run on the same physical machine. In each time interval, the *Agent* gets resource usage metrics from the *mond* daemons of each physical node that its VMs are running on. Then, it predicts the resource usage demand for the next interval for all its VMs. Based on this prediction, it determines the resource allocations to be assigned to them for the next interval. In each physical node, a module called *Arbiter* is running that gathers resource allocation assignments from all *Agents* that have VMs on the physical node, decides the final resource allocation assignment for all VMs, and actually applies them through the Virtual Machine Monitor (VMM). Its main responsibility is to resolve conflicts that result from assignments that overpass the resource capacity of the physical node.

This architectural arrangement of distributing the resource allocation decisions on several *Agents* running on different physical nodes has the benefit of providing scalability of the resource management in a large scale cloud infrastructure.

B. SVM for Cross-correlation Prediction

One of the responsibilities of the *Agent* module is to take resource usage data from the *mond* daemons, build a time series of resources of all its VMs, and based on that predict the resource usage for the next time interval. This prediction is a function of a number of past time series usage values called a time window. This is given mathematically by:

$$x(t+1) = f(x(t), x(t-1), x(t-2), .., x(t-16)) \quad (1)$$

where $x(t)$ is resource usage at time interval t, $x(t+1)$ is the predicted resource usage for the next interval and $f()$ is the prediction function.

The prediction function is learned using a machine learning approach, i.e., in our proposal an SVM. The motivation for using this technique are its advantages over alternative techniques, as shown by other research work [13] and summarized below:

- Non-parametric method avoiding the need define a model structure beforehand.
- Applicable to non-linear models and non-stationary processes.
- Guaranteed to find the global optimum, unlike artificial neural networks that can get stuck to local minima.
- A small number of parameters are required to tune the algorithm.
- Better generalization by avoiding over-fitting.

SVMs, as other supervised machine learning techniques, undergo a training phase by applying past time series samples to generate a model to be used for prediction. Before the time series data is applied to the learning algorithm, a transformation phase is needed to make it more suitable for learning by removing the temporal ordering of input samples by creating additional inputs called "lagged" variables. The training is performed repeatedly for each time interval in order to adapt to changing characteristics of time series. Since there are dependencies between tiers of a multi-tier application, the resource usage time series of VMs belonging to the same application are cross-correlated, meaning that the usage prediction for one resource depends on past resource usages of other VMs of the same application. To take this cross-correlation into account, we apply the SVM algorithm to model multiple time series simultaneously.

C. Proactive Resource Allocation

The other responsibility of the *Agent* is to determine the resource allocation to be given to all resources of all VMs of the application it is responsible for. This decision is made in discrete time intervals where in each interval, the resource allocation to be given to each resource for the next interval is determined. This decision is based on resource usage prediction of the next interval as a result of the SVM algorithm. More specifically, the amount of resources allocated is estimated as the resource usage prediction plus a small margin of 5% of the resource capacity. This makes it possible to allocate just the needed resources to keep the cost to a minimum and to leave room for any possible prediction error in order to keep the performance to acceptable levels.

Due to prediction errors, it can happen that the resource amount allocated is less than the usage demand, leading to poor performance and influencing the future predictions. To mitigate this problem, in each control interval, if resource starvation is encountered, meaning that the difference between the resource allocated and resource usage is less than 5% of the resource capacity, the resource allocation margin for the next time interval is doubled to 10% of the resource capacity. This leaves room to satisfy the resource demand and get realistic usage data for future predictions. If resource starvation is not encountered in a time interval, the normal resource allocation margin of 5% is set for the next interval.

After determining the resource allocations for all its VMs according to the algorithm above, the *Agent* submits them to the corresponding *Arbiters*. The *Arbiter* collects, in each time interval, resource allocations from all *Agents* that have VMs running on its physical machine and actually applies them through the VMM. In case the sum of the resource allocations of all VMs running on the physical machine is greater than the physical resource capacity, the *Arbiter* redistributes resource allocations to VMs proportionally to their requirements. Each VM gets the final resource allocation after redistribution according to the formula:

$$A = \frac{Alloc}{SumAlloc} * Capacity \quad (2)$$

where A is the final allocation, $Alloc$ is the preliminary allocation, $SumAlloc$ is the sum of all VMs' preliminary allocations and $Capacity$ is the total resource capacity.

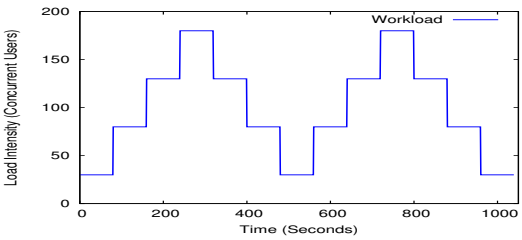


Figure 2. Workload.

IV. Implementation

All APRA components run in Dom0 of the physical nodes and are written in Java, except for the *mond* monitoring daemons that are written in the C language. The *Mond* daemons use the libvirt API to get CPU usage data in each time interval. Since there is no simple way in current VM technology to measure memory consumption of a VM outside of it, we implemented a small daemon called *mmond* running inside each VM to get memory usage data from the /proc/meminfo file system in Linux. *Arbiter* sets CPU and memory allocations through the xm sched-credit and xm mem-set commands of the Xen hypervisor. The *agent* is implemented as a two-threaded application. One thread communicates with *mond*, maintains usage time series and predicts new usage values for the next interval. The other thread makes allocation decisions and communicates with the *Arbiters*. The *Agents* use the SVM implementation of the WEKA [6] machine learning framework for time series cross-correlation forecasting of resource demand. All communication between the components of APRA is based on the IP socket interface. The time interval for sampling resource usage data, training the forecasting model and applying resource allocation decisions, is set to 10 seconds. We have set the time window interval of the past time series usage data to 16 samples according to the results of some experiments and to keep the training time to a minimum.

V. Experimental Results

We have built a testbed composed of 3 physical nodes and 3 two-tier applications (App1, App2, App3) arranged as shown in Fig. 1. Each tier runs inside one VM and the VMs of the same application can run on different physical machines. We used the CloudSuite [4] web-serving benchmark as a representative multi-tier application. We tested APRA with the two-tier configurations where the first tier is the web front-end tier that implements the *olio* web application and the second is the *mysql* database tier that stores information about social events. Three clients are running on the first node that generate load for the three applications through the Faban workload generator. The physical machines are blade servers with two Intel quad-core Xeon 2.00 GHz CPUs with a total of 8 CPU cores and 4 GB of RAM. We used the Xen 4.1.2 Hypervisor as our virtualization platform for managing VMs and used the Ubuntu 12.04 OS with 3.8.0-29-generic Linux kernel in Dom0 and DomU VMs. The machines are connected over a 1 Gbit/sec Ethernet.

In our experiments, we pin Dom0 for all physical nodes to 2 CPU cores and give them 2 VCPUs. We pin web1 VM to one CPU core and assign 1024 MB of RAM, we pin db1 and web2 VMs to 1 CPU core and assign to them 1024 MB of RAM and finally we pin db2, web3, db3 VMs to 2 CPU cores and assign to them 1536 MB of RAM. We assign to all application VMs 1 VCPU. The workload generated by the clients for the three applications with varying intensity lasting for about 17 minutes is shown in Fig. 2. On the y-axis, the workload intensity set by the number of concurrent users that send requests to the web application is shown. The workload starts with 30 concurrent users at time 0 seconds and changes every 80 seconds by $\pm$ 50 users.

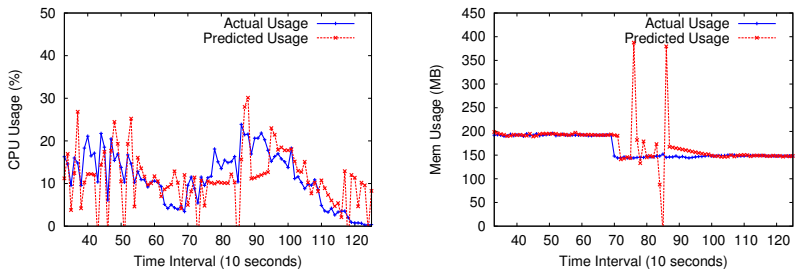
We compared different aspects of two approaches: a) one that applies SVM to make predictions of usage time series of each resource separately and b) the other one that makes predictions of multiple usage time series of the same multi-tier application simultaneously to take cross-correlation into account. Fig. 3 shows the actual and predicted CPU and memory usage of the web front-end VM of App2 with using cross-correlation prediction and without using it. Similar results are taken also for the CPU and memory resources of the database VM, but we do not show these results for space reasons. It is evident especially for the CPU resource that applying cross-correlation leads to better resource usage predictions than without it. This is because there are correlations between resource usage time series of the same multi-tier application as the result of the interdependencies between its VMs, thus making cross-correlation time series predictions a necessity for getting better results.

To verify the prediction accuracy of the two approaches quantitatively, Table I shows the results for three widely used metrics of prediction errors for App2: MAE, RMSE and MAPE. For each resource, we have smaller prediction errors with cross-correlation than without it.

TABLE I
Prediction Errors for 2 Approaches

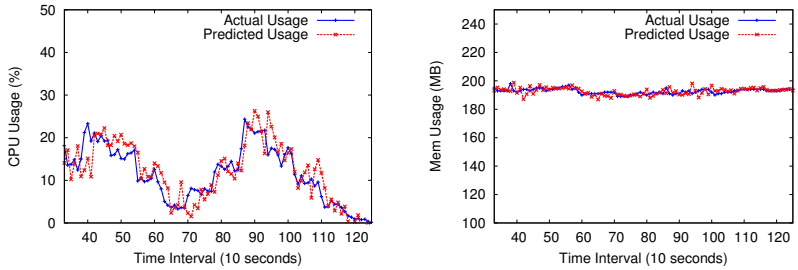
Resource	Approach	MAE	RMSE	MAPE
WEB CPU	*Cross-Correlation*	2.78	3.7	0.26
	No-Cross-Correlation	4.52	5.81	0.53
WEB MEM	*Cross-Correlation*	2.29	3.26	0.011
	No-Cross-Correlation	4.37	10.95	0.026
DB CPU	*Cross-Correlation*	0.48	0.66	0.25
	No-Cross-Correlation	0.79	1.17	0.48
DB MEM	*Cross-Correlation*	0.5	0.71	0.001
	No-Cross-Correlation	9.44	52.54	0.038

To understand how proactive resource allocation based on cross-correlation prediction works, Fig. 4 shows the used and allocated resources determined by APRA for the two



(a) Actual and predicted CPU usage for web front-end VM without cross-correlation

(b) Actual and predicted memory usage for web front-end VM without cross-correlation



(c) Actual and predicted CPU usage for web front-end VM with cross-correlation

(d) Actual and predicted memory usage for web front-end VM with cross-correlation

Figure 3. Actual and predicted resource usage of the web front-end VM for two approaches.

TABLE II
RESPONSE TIMES OF THREE APPLICATIONS FOR THE TWO APPROACHES

Appl.	Approach	Avg RespTime	90th% RespTime
APP1	*Cross-Correlation*	2.05	5.82
	No-Cross-Correlation	2.43	6.25
APP2	*Cross-Correlation*	0.25	0.45
	No-Cross-Correlation	0.4	0.87
APP3	*Cross-Correlation*	0.29	0.63
	No-Cross-Correlation	0.4	0.95

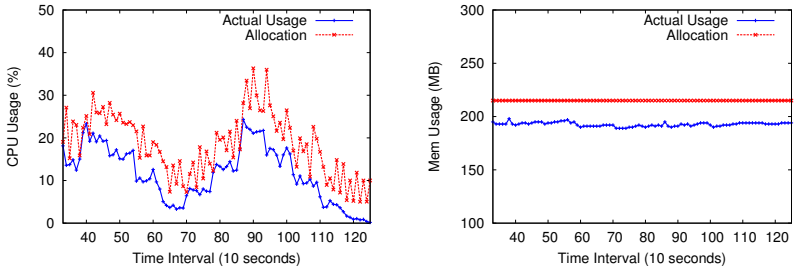
VMs of App2. APRA allocates the minimum amount of resources needed and changes it dynamically according to the demand. This keeps resource costs to a minimum while satisfying application performance. Fig. 4(a) shows that only in a few cases the allocation is equal to the usage demand, while most of the time it is within some margin above the demand, resulting in good application performance.

Finally, we present how the two proactive resource allocation approaches, one based on cross-correlation prediction and the other one based on separate time series predictions, affect application performance. In Table II, the average and the 90th percentile of the response time in seconds for the three applications with cross-correlation prediction and without it is shown. Resource allocation with cross-correlation prediction achieves lower application response times on all applications except for the App1 application. The almost equal performance of the two approaches for the App1 application is explained by the fact that its performance is degraded by the interference of workload generation client VMs running on the same physical node with its web front-end VM. The better application performance with cross-correlation approach is explained by the fact that it achieves better prediction accuracy and therefore better allocations.

VI. CONCLUSION

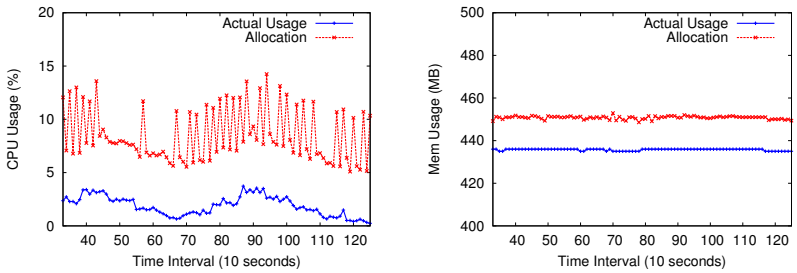
In this paper, Automatic Proactive Resource Allocation (APRA) based on multiple resource demand predictions using SVM has been presented. APRA is motivated by the interdependencies that exist between resources of multiple VMs of the same multi-tier application. This necessitates a cross-correlation resource demand prediction approach for better accuracy. Based on predicting CPU and memory resource demand, APRA makes allocation decision to lower costs and keep application performance to acceptable levels. Experimental results with the CloudSuite web serving multi-tier application benchmark have shown that the cross-correlation prediction approach achieves better prediction accuracy and resource allocation decisions compared to the non-correlation prediction approach.

As future work, we plan to include VM live migration as another resource allocation mechanism to handle the case when the predicted physical machine resource demand overpasses the total capacity. Furthermore, the inclusion of VM replication as another resource allocation mechanism is



(a) Actual and allocated CPU usage for web front-end VM

(b) Actual and allocated memory usage for web front-end VM



(c) Actual and allocated CPU usage for database VM

(d) Actual and allocated memory usage for database VM

Figure 4. Actual and allocated CPU and memory resource usage of web front-end and database VMs with cross-correlation prediction.

an interesting approach for future work.

ACKNOWLEDGMENT

This work is supported by the German Ministry of Education and Research (BMBF) and the Albanian Government.

REFERENCES

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proc. 19th ACM Symposium on Operating Systems Principles*, pages 164–177. ACM Press, 2003.

[2] M. N. Bennani and D. A. Menasce. Resource allocation for autonomic data centers using analytic performance models. In *Proc. 2nd International Conference on Automatic Computing*, pages 229–240. IEEE Press, 2005.

[3] R. P. Doyle. Model-based resource provisioning in a web service utility. In *Proc. 4th USENIX Symposium on Internet Technologies and Systems*, pages 5–5. USENIX Association, 2003.

[4] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi. Clearing the clouds: A study of emerging scale-out workloads on modern hardware. In *Proc. 17th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 37–48. ACM Press, 2012.

[5] Z. Gong, X. Gu, and J. Wilkes. Press: Predictive elastic resource scaling for cloud systems. In *Proc. International Conference on Network and Service Management*, pages 9–16. IEEE Press, 2010.

[6] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[7] J. Heo, X. Zhu, P. Padala, and Z. Wang. Memory overbooking and dynamic control of xen virtual machines in consolidated environments. In *Proc. 11th IFIP/IEEE international conference on Symposium on Integrated Network Management*, pages 630–637. IEEE Press, 2009.

[8] R. Hu, J. Jiang, G. Liu, and L. Wang. Kswsvr: A new load forecasting method for efficient resources provisioning in cloud. In *Proc. IEEE International Conference on Services Computing*, pages 120–127. IEEE Press, 2013.

[9] S. Islam, J. Keung, K. Lee, and A. Liu. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Gener. Comput. Syst.*, 28(1):155–162, Jan. 2012.

[10] D. A. Menasce and M. N. Bennani. Autonomic virtualized environments. In *Proc. International Conference on Autonomic and Autonomous Systems*, pages 28–38. IEEE Press, 2006.

[11] P. Padala, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, and K. Salem. Adaptive control of virtualized resources in utility computing environments. In *Proc. 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, pages 289–302. ACM Press, 2007.

[12] J. Rao, X. Bu, C.-Z. Xu, L. Wang, and G. Yin. Vconf: A reinforcement learning approach to virtual machines auto-configuration. In *Proc. 6th International Conference on Autonomic Computing*, pages 137–146. ACM Press, 2009.

[13] N. I. Sapankevych and R. Sankar. Time series prediction using support vector machines: A survey. *IEEE Computational Intelligence Magazine*, 4(2):24–38, 2009.

[14] Z. Wang, X. Zhu, S. Singhal, and H. Packard. Utilization and slo-based control for dynamic sizing of resource partitions. In *Proc. 16th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, pages 24–26. Springer-Verlag, 2005.

[15] J. Wildstrom, P. Stone, E. Witchel, and M. Dahlin. Machine learning for on-line hardware reconfiguration. In *Proc. 20th International Joint Conference on Artifical Intelligence*, pages 1113–1118. Morgan Kaufmann Publishers Inc., 2007.

[16] C.-Z. Xu, J. Rao, and X. Bu. Url: A unified reinforcement learning approach for autonomic cloud management. *J. Parallel Distrib. Comput.*, 72(2):95–105, 2012.