Proceedings of 2013 International Conference on Fuzzy Theory and Its Application
National Taiwan University of Science and Technology, Taipei, Taiwan, Dec. 6-8, 2013

Load Prediction of Virtual Machine Servers Using Genetic Expression Programming

Lung-Hsuan Hung and Chih-Hung Wu
Department of Electrical Engineering, National University of Kaohsiung,
No. 700, Kaohsiung University Road, Nan-Tzu District, Kaohsiung 811, Taiwan
Tel: +886-7-5919752, Fax: +866-7-5919446
Email: m1005111@mail.nuk.edu.tw, johnw@nuk.edu.tw

***Abstract*—Virtualization is a key technology for cloud-computing, which creates various types of virtual computing resources on physical machines. A center of virtual machine (VM) servers manages different load situations of servers and adjusts flexibly the consumptions of physical resources to achieve better cost-performance efficiency. One of the key problems in the management of VM servers (VMSs) is load prediction with which decisions for load-balance as well as other management issues can be engaged. This study employs genetic expression programming (GEP) for deriving regression models of load of VMSs. GEP regression models are "white-boxes" that have visible structures and can be modified and integrated with other VM management mechanisms. Data representing the types of VM resources, VM loads, etc., are collected for training GEP models. With the GEP models, one can predict the work load of VMSs so that precise decisions of load-balance can be made. The experimental results show that GEP can generate precise models for load prediction of VMSs than other methods.**

***Index Terms*—Cloud Computing; Virtual Machine; Performance Modeling; Genetic Programming; Genetic Express Programming**

I. Introduction

Cloud computing has been widely used in our daily lives. Virtualization is a key technology for cloud applications, which creates various types of virtual computing resources on physical machines [1]. Such platforms support the execution of several virtual machines (guests) and operating systems simultaneously on one physical machine (host). A center of virtual machine (VM) servers manages different load situations and adjusts flexibly the consumptions of physical resources to achieve better cost-performance efficiency, while guaranteeing service level agreements (SLA) to the end users.

Migration for load-balancing is an important technology in the management of VM servers (VMSs). It moves a guest form a host to another, while the guest is still powered on. When a VM center has a lot of under-utilized hosts, scattered guests can be merged to fewer hosts via the live migration function. These freed-up hosts can either be powered off or suspend to save energy and reduce the cost. Live migration also can keep the load of hosts to meet SLA [2]. A need for effective live-migration for load-balance is the prediction of loads of VMs. With such models, one can predict the performance of VMSs before and after the live migration of VMs.

Some studies present the importance of load prediction for load-balance of VMSs. Khanna *et al* presented the concept of server consolidation using virtualization in [3] and pointed out some associated performance issues. Checconi *et al* [4] traced real-time issues (VLC video server) in live migration of VMs. Voorsluys *et al* [5] evaluated the performance of VM live migration for applications running on VMs that are deployed in Xen. Zhao and Huang [6] implemented a simple model for decreasing the migration time of VMs by shared storage and fulfilling the zero-downtime relocation of VMs by transforming them as Red Hat cluster services. An application level load balancing system has implemented by Alonso-Calvo *et al* [7], which reduces processing time from 35% to 50%. Doong *et al* [8] presented a "black-boxed" multi-kernel support vector regression model for describing VMs performance models. A scheduling strategy on load balancing of VM resources that refers historical data and current state of the system using genetic algorithm is developed by Hu *et al* [9]. Ardagna *et al* [10] proposed a capacity allocation algorithm that is able to coordinate multiple distributed resource controllers operating in geographically distributed cloud sites.

Most studies describe the behaviors of VMs using statistical models. This study employs the technique of genetic expression programming (GEP) for deriving regression models of load of VMSs. The GEP regression models are "white-boxes" that have visible structures and can be modified and integrated with other VM management mechanisms. In this study, VMSs are implemented on Xen [11] and data describing the consumptions of various VM resources, user requests, VM loads, etc., are collected for training GEP models. The performances of the GEP models are compared with other regression models. The experimental results show that GEP can generate precise models for load prediction of VMSs than other methods.

II. The Load Model

The so-called "load" of a server is how it consumes all types of resources, such as CPU, network bandwidth, memory, etc. The load of a VMS is the accumulation of all loads of VMs. All processes of either VM or the VMS host are prioritized according to some parameters pre-defined in the VMS hypervisor when they are considered to be executed by the VMS host. The parameters associated with the scheduling priority dominate the performance of VMs as well as the load of VMSs. A number of performance parameters are designed

978-1-4799-0386-3/13/$31.00 ©2013 IEEE

on various platforms. This study employs Xen as the VM platform and uses the parameters associated with Xen.

According to [12], several criteria to CPU load of VM in Xen are used. This study adopts `Credit`, a non-preemptive scheduler of Xen. It supports the proportional share scheduler through the `weight` parameter, i.e., an instantaneous form of resources sharing that allocates CPU to VMs in proportion to the number of shares (`weight`) that each VM receives. `Credit` also supports work-conserving (WC-mode) and/or non workconserving (NWC-mode) modes for CPU scheduling in Xen. In the WC-mode, the shares are merely guarantees, while the shares are caps in the NWC-mode. The non-preemptive scheduler `Credit` reruns the scheduling decision only when the running client gives up the CPU.

Each domain including the host domain (Dom0) receives a default `weight` of 256 and a default `cap` of 0. Legal `weights` range from 1 to 65535. The `cap` parameter fixes the maximum amount of CPU a domain can consume. The default setting of 0 means there is no upper `cap` (WC-mode). A `cap` of 100 indicates one physical CPU, 50 is half a CPU, 400 is 4 CPUs, etc. Additionally, the `Credit` scheduler does a global load balancing job on multi-core systems. In this study, the performance model of a VM is described by `weight` and `cap`. Two performance indicators are used, the calculation power ('`cal`') and the network throughput ('`netperf`'). It is expected that load models of '`cal`' and '`netperf`' of a VMS can be described by the parameters `weight` and `cap` associated with the VMs on the VM host.

III. Genetic Expression Programming

GEP [13] is a version of genetic programming (GP). It describes problems in the same tree-like representation of GP, but the entities produced by GEP are the expression of a genome. The use of expression trees brings efficiency to GEP because of easier applications of genetic operations on a simple linear structure. A gene expression in GEP consists of *head* and *tail*. The head contains symbols that represent both *functions* and *terminals*, whereas the tail contains only terminals. For each problem, the length of the head h is chosen, whereas the length of the tail t is a function of h and the number of arguments of the function with the most arguments n, and is evaluated by $t = h \times (n-1) + 1$. Suppose that symbols $a, b, \ldots, e$ are terminals (input variables or constant numbers) and $+$, $-$, $\times$, $\div$, $s(sqrt)$ are operators (functions). In the gene expression $\times+\div\times csdabcaddecbabdaa$, the first 10 elements belong to the head part and the remaining 11 ones (all terminals) are in the tail. Therefore, the above linear gene expression represents the algebraic expression $(a \times b + c) \times \sqrt{c} \div d$. The genetic operators associated with GEP are reproduction, mutation, transposition, and recombination. Reproduction and mutation are similar to that in traditional GP. Transposition is to prune and insert a segment of gene to a randomly selected position. Recombination is similar to cross-over but breaks and recombines two genes. With this linear expression, the computational complexity of search for specific gene elements in a genetic operation can be reduced from $O(n)$ to $O(1)$. For more details on GEP, please refer to [13].

IV. Problem Formulation

Regression is a method for finding a descriptive model for the input and output of a problem so that a rational output corresponding to a new input is predictable. A formal description is given below. Suppose that a data set $I=\{\langle \vec{x}_i, y_i\rangle | i = 1, \ldots, n\}$ is observed. Let U be the universe of $\vec{x}_i$. The purpose of regression is to find a function $g(\vec{x})$ such that $g(\vec{x}_i)$ approximates to y_i for all possible $\vec{x}_i$, i,e., $y_i \cong g(\vec{x}_i)$, $\forall \vec{x}_i \in U$. To measure the effectiveness of $g(\vec{x})$, an *error function*, $\mathcal{E}$, is usually defined over all errors between y_i and $g(\vec{x}_i)$, as

$$\mathcal{E} = \frac{1}{n}\sum_{i=1}^{n} \varepsilon_i^2, \tag{1}$$

where $\varepsilon_i = y_i - g(\vec{x}_i)$, $0 \le i \le n$. In other words, regression is to find the best expressive function $g(\cdot)$ by minimizing $\mathcal{E}$.

The main theme of this study is to find a regression model describing the load of VMSs that is described by `weight` and `cap`. For this purpose, a set of training data consisting of the performance of VM and the loads of VMSs must be collected. Thus, for each VM, two regression models are discussed; one is for `cal` and one is for `netperf`. Both are described by `weight` and `cap`. For a VMS running k VMs, a load model can be described by the 2(k+1) variables that are from the `weight` and `cap` associated with the k VMs and the VM hypervisor. The training data set for the load model `cal` of the VMS is

$$I_c = \{\langle (w_{0_i}, w_{1_i}, \ldots, w_{k_i}, c_{0_i}, c_{1_i}, \ldots, c_{k_i}), p_i\rangle | i = 1, \ldots, n\}, \tag{2}$$

where p_i is the `cal` performance measured on the VMS and w_{0_i} and c_{0_i} is the `weight` and `cap` associated with the VM hypervisor, $w_{i_{j\ge 1}}$ and $c_{i_{j\ge 1}}$ is the `weight` and `cap` with the jth VM. Similarly, the training data set for the load model `netperf` of the VMS is

$$I_t = \{\langle (w_{0_i}, w_{1_i}, \ldots, w_{k_i}, c_{0_i}, c_{1_i}, \ldots, c_{k_i}), t_i\rangle | i = 1, \ldots, n\}, \tag{3}$$

where n_i is the `netperf` performance measured on the VMS. The purpose of regression is to find for `cal` a descriptive model $g_c(\vec{x}_i)$ from I_c by minimizing

$$\mathcal{E}_c = \frac{1}{n}\sum_{i=1}^{n} (p_i - g_c(\vec{x}_i))^2, \tag{4}$$

where $\vec{x}_i = (w_{0_i}, w_{1_i}, \ldots, w_{k_i}, c_{0_i}, c_{1_i}, \ldots, c_{k_i})$. Also, for `netperf`, a descriptive model $g_t(\vec{x}_i)$ is derived from I_t by minimizing

$$\mathcal{E}_t = \frac{1}{n}\sum_{i=1}^{n} (t_i - g_t(\vec{x}_i))^2. \tag{5}$$

Both models are derived by GEP.

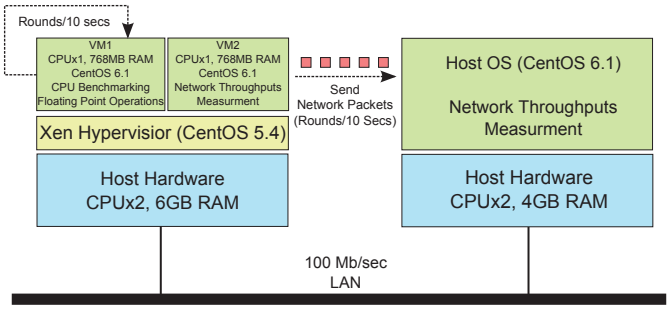


Fig. 1. The configuration of the VMS used in the experiment

TABLE I
PARAMETER SETTINGS OF GEP REGRESSION

Parameters	Value
Termination conditions:	
Fitness	1.00
Generation	5000
Population size	100
Prob. crossover	0.30
Prob. mutation	0.45
Prob. reproduction	0.10
Gene head length	16

V. EXPERIMENTS

A. Data collection & parameter settings

In order to test our ideas, a sufficient number of VM load data were collected in the following VMS environment. The VMS is installed using Xen 3.4.1, which runs CentOS 5.4 on an Intel dual core 2.33GHz CPU (Xeon 3065) with 6 GB memory. Two guest VMs (VM1 and VM2) were created with 768 MB memory each. For the comprehension of VM tasks, VM1 is intended to run calculation intensive tasks, and VM2 runs network intensive tasks. In order to measure the performance, benchmark tests are executed simultaneously on both VMs. VM1 runs the Timer and TimerTask classes in Java to determine how many floating point operations can be performed within a fixed period of 10 seconds. The floating point operation is "`Math.sin(Math.random())`". The Java Timer class observes the time clock in VM1. The Java program terminates in 10 seconds, and reports how many floating point operations (MOPS/sec) have been performed. VM2 runs the netperf package [14] for the measurement of network throughputs. VM2 continuously sends network packets to a physical machine in the same local area network and collects the netperf throughput data every 10 seconds. The network back bone is a 100Mb/sec local area network, and netperf reports a typical test result of tens of Mb/sec. The actual performance of both VMs depends on the `Credit` parameters (`cap` and `weight`) set for them. The configuration of the VMS is depicted in Figure 1.

The GEP programs are implemented in C, running on an 2.4GHz Intel Core i5-520 CPU and 2GB RAM. The hyper-parameters associated with GEP are presented in Table I. In the experiment, `weight` is randomized ranging from 1 to 1024 and `cap` is from 1 to 100. Both parameters are set for VM1 and VM2. The VM host works with various `weight` values and a constant `cap`=0. This is a service guarantee for the host machine to work normally. In such settings, there have five terminal variables ($c1, c2, w0, w1, w2$) and two target variables (`weight` and `cap`). Various combinations of `weight` and `cap` are tested on the VMS and the corresponding values of '`cal`' and '`netperf`' of both VMs are collected. Among all simulation data, a portions of 90% of data are sampled for training GEP and the remaining 10% of data are used for testing. We employed three indexes to evaluate the results of regression, the mean squared error (MSE), mean absolute error (MAE), and mean absolute percentage error (MAPE).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2, \quad (6)$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - y_i'|, \quad (7)$$

$$MAPE = (\frac{1}{n}\sum_{i=1}^{n}|\frac{y_i - y_i'}{y_i}|) \times 100 \quad (8)$$

where y_i (y_i') can either p_i ($g_c(\vec{x}_i)$) in Eq.(4) or t_i ($g_t(\vec{x}_i)$) in Eq.(5). The effectiveness of regression models obtained from GEP is compared with some other white-box methods, e.g., linear regression (LIN), polynomial regression (POL), exponential regression (EXP), and power regression (POW). All methods were trained by the same data set. To avoid biased results, each test is performed for five runs and the values of '`cal`' and '`netperf`' are averaged.

B. Results

Table II presents the comparisons on the regression results of all methods. In the regression of '`cal`', POW outperforms other methods and EXP is the worst. LIN and GEP have very similar results in terms of MSE, MAE, and MAPE, and can be considered as the second best ones. The MSE and MAE of GEP are very close to POW. In '`netperf`', GEP has the best performance in all aspects and POL is the second best. The performance of LIN is far behind GEP. Again, EXP is the worst. It may conclude that the regression of '`cal`' is easier than that of '`netperf`' because a simple regression method like LIN can have a satisfactory result. However, in the regression of '`netperf`', only GEP provides satisfactory performance. GEP can be considered as an effective method.

Table III compares the number of operators used in each regression model. LIN produces very simple regression models with fewer operators. Clearly, the terminal variables are not just linearly connected so a simple method like LIN can not obtain precise models with just few operators. Methods like POL and EXP target on non-linear regression and can produce better regression results. However, the regression models is composed of only polynomial or exponential functions that may not have enough explanation power to express the relations among `weight`, `cap`, `cal`, and `netperf`. GEP takes 16 operators for the regression of '`cal`' and 11 ones

TABLE II
COMPARISONS ON ACCURACY – ALL METHODS(**Boldface**: BEST IN THE ROW)

cal	LIN	EXP	POW	POL	GEP
MSE	1.76	17.37	**1.43**	2.75	1.72
MAE	0.93	3.01	**0.76**	1.14	0.97
MAPE	12.81	31.54	**6.55**	15.12	12.02
netperf					
MSE	89.86	184.50	118.10	27.70	**23.16**
MAE	7.80	11.70	8.81	4.29	**3.69**
MAPE	46.55	45.50	22.90	17.05	**16.87**

TABLE III
COMPARISONS ON THE NUMBER OF OPERATORS – ALL METHODS

	LIN	EXP	POW	POL	GEP
cal	10	11	10	29	16
netperf	10	11	10	29	11

for 'netperf'. This may due to the simplicity of the 'cal' regression problem and GEP overfits the training data. Conversely, GEP takes only 11 operators and best fits the training data. GEP may be suitable for the regression of complicated problems than for simple problems; this is consistent with the other studies.

Finally, we discuss on the distributions of errors of all regression models. Figure 2 and Figure 3 presents the percentage statistics on the accuracy of 'cal' and 'netperf' models obtained from various methods, respectively. Observe the first bin of the distributions of errors in Figure 2 and Figure 3. Most of the regression methods, except EXP, have low errors in terms of MSE and MAE. It concludes that LIN, POW, POL, and GEP are effective for the regression of 'cal' and 'netperf'. The EXP errors spread around all bins, indicating that the regression of 'cal' and 'netperf' may not fit for exponential functions. For both regression problems, less than 50% of low MAPE (5%) is observed, as shown in Figure 2(c) and Figure 3(c). All methods do not generate the optimal models for 'cal' and 'netperf'. MAPE reveals the stability of the regression models. The distributions of MAPE associated with 'netperf' spreads wider in all bins than that of 'cal'. This may due to the quality of the 'netperf' data which are collected from the Internet with stochastic traffics. In 'cal', GEP behaves similarly to POW. In 'netperf', GEP performs as the best. Therefore, GEP can be considered as a competitive method for the regression of VMS load models.

VI. CONCLUSION AND DISCUSSION

In this paper, the generation of "white-box" load prediction models of VMS using GEP is presented. For training VMS load models, data describing the consumptions of processing and networking resources are collected on a Xen platform. The performances of the GEP models are compared with other regression models. The experimental results show that GEP can generate rational models for load prediction of VMS than other methods. Current load models are featured by 'cal' and 'netperf' and are learned from a VMS with 2 VMs. More descriptors associated with VM performance that are collected with more VMs may improve the training results. Effective features of VMS load models need advanced studies. A VMS load-balancing mechanism based on our proposed method is under-development.

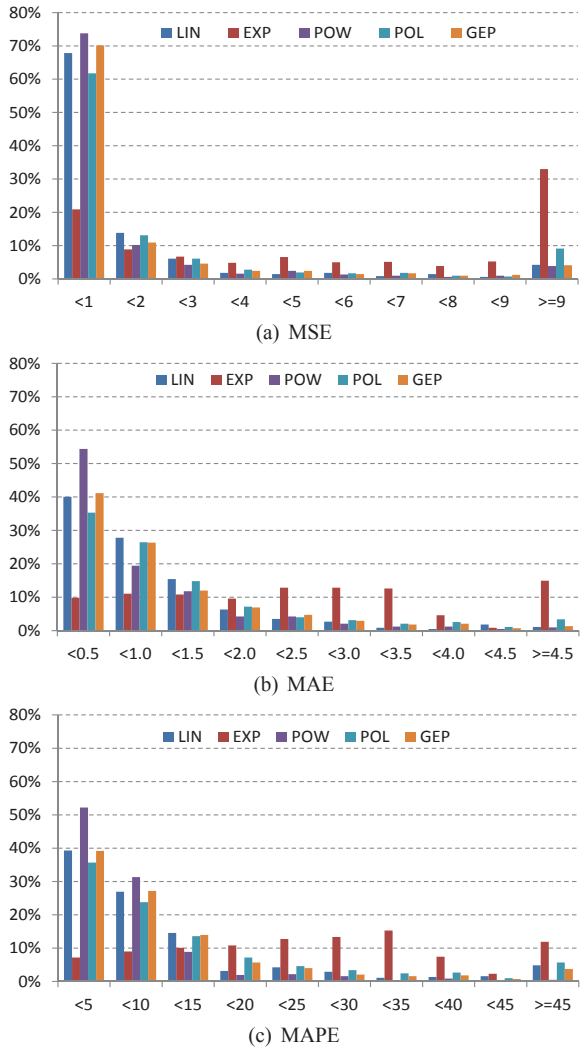


Fig. 2. Percentage statistics on 'cal' of all methods

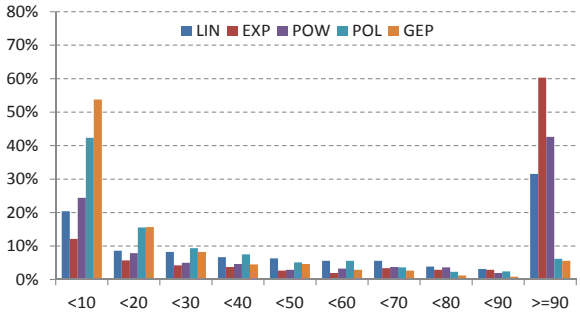
ACKNOWLEDGMENT

This work was partially supported by National Science Council (NSC), Taiwan, under Grant No. NSC 101-2221-E-390-029.

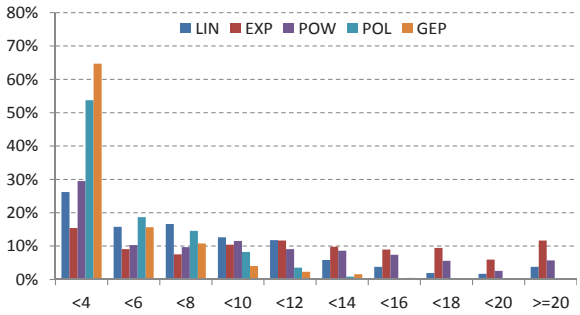
REFERENCES

[1] M. Rosenblum and T. Garfinkel, "Virtual machine monitors: current technology and future trends," *Computer*, vol. 38, no. 5, pp. 39–47, 2005.

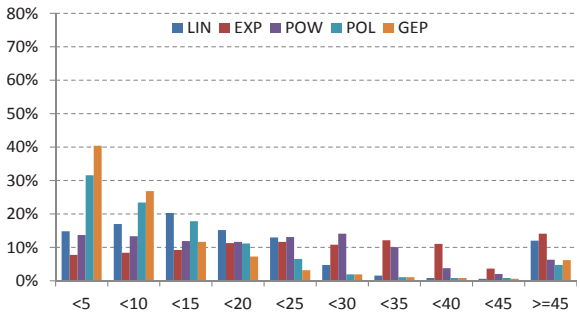
[2] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.



(a) MSE



(b) MAE



(c) MAPE

Fig. 3. Percentage statistics on '`netperf`' of all methods

[3] G. Khanna, K. A. Beaty, G. Kar, and A. Kochut, "Application performance management in virtualized server environments," in *Proceedings of the 10th IEEE/IFIP Network Operations and Management Symposium*, J. L. Hellerstein and B. Stiller, Eds. IEEE, 2006, pp. 373–381.

[4] F. Checconi, T. Cucinotta, and M. Stein, "Real-time issues in live migration of virtual machines." in *Proceedings of the 2009 international conference on parallel processing*, ser. Lecture Notes in Computer Science, H.-X. Lin, M. Alexander, M. Forsell, A. Knupfer, R. Prodan, L. Sousa, and A. Streit, Eds., vol. 6043. Springer, 2009, pp. 454–466.

[5] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of virtual machine live migration in clouds: A performance evaluation," in *Proceedings of the 1st International Conference on Cloud Computing*, ser. CloudCom'09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 254–265.

[6] Y. Zhao and W. Huang, "Adaptive distributed load balancing algorithm based on live migration of virtual machines in cloud," in *Proceedings of the 2009 Fifth International Joint Conference on INC, IMS and IDC*, ser. NCM '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 170–175.

[7] R. Alonso-Calvo, J. Crespo, M. Garcia-Remesal, A. Anguita, and V. Maojo, "On distributing load in cloud computing: A real application for very-large image datasets." *Procedia Computer Science*, vol. 1, no. 1, pp. 2669–2677, 2010.

[8] S. H. Doong, C. C. Lai, S. J. Lee, C. S. Ouyang, and C. H. Wu, "Performance modeling of virtual machines hosted on Xen," in *Proceedings of the 2012 International Conference on Cloud Computing and Virtualization*, 2010.

[9] J. Hu, J. Gu, G. Sun, and T. Zhao, "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment," in *Proceedings of the 2010 3rd International Symposium on Parallel Architectures, Algorithms and Programming*, ser. PAAP '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 89–96.

[10] D. Ardagna, S. Casolari, M. Colajanni, and B. Panicucci, "Dual time-scale distributed capacity allocation and load redirect algorithms for cloud systems." *Journal of Parallel and Distributed Computing*, vol. 72, no. 6, pp. 796–808, 2012.

[11] (2003) The xen project website. [Online]. Available: http://www.xenproject.org/

[12] L. Cherkasova, D. Gupta, and A. Vahdat, "Comparison of the three CPU schedulers in Xen," *SIGMETRICS Performance Evaluation Review*, vol. 35, no. 2, pp. 42–51, Sep. 2007.

[13] C. Ferreira, "Gene expression programming: a new adaptive algorithm for solving problems," *Complex Systems*, vol. 13, no. 2, pp. 87–129, 2001.

[14] (1996) The netperf website. [Online]. Available: http://www.netperf.org/netperf/