This paper appeared in *ACM Operating Systems Review*, Vol. 39, No. 4, pp.37-45, October, 2005.

Performance Analysis of an Admission Controller for CPU- and I/O-Intensive Applications in Self-Managing Computer Systems

Mais Nijim Tao Xie Xiao Qin

Department of Computer Science
New Mexico Institute of Mining and Technology
801 Leroy Place, Socorro, New Mexico 87801-4796
{mais, xietao, xqin}@cs.nmt.edu

Abstract

With rapid advances in processing power, network bandwidth, and storage capacity, computer systems are increasingly becoming extremely complex. Consequently, it becomes expensive and difficult for human beings to manually manage complex computer systems. This problem can be effectively tackled by self-managing computer systems, which are intended to meet high performance requirements in a dynamic computing environment. In this paper, we develop a performance model for self-manage computer systems under dynamic workload conditions, where both CPU- and I/O-intensive applications are running in the systems. In particular, we design in this paper a 2-dimenssional Markov chain model with two different arrival and service rate of CPU- and I/O-intensive jobs. Importantly, two serving probabilities with respect to CPU- and I/O intensive jobs are derived. To validate the analytical model, we developed an adaptive admission controller in which the model is incorporated. Experimental results demonstratively show that the controller is capable of achieving high performance for computer systems under workloads exhibiting high variability.

1. Introduction

Computer systems are increasingly becoming large-scale and complex. This trend is mainly because of rapid advances in processing power, network bandwidth, and storage capacity. As a result, *it becomes* expensive and *inefficient for human* beings *to* manually *maintain complex computer systems*. Self-managing computer systems are emerging to configure and repair

themselves in dynamic computing environments [1][4][6][7].

In this paper we construct an analytical performance model, which can be of help for large-scale computer systems to manage themselves under dynamic and mixed workloads, where both CPU- and I/O-intensive jobs are running in the systems.

I/O intensive applications become widely used, this is partly because a variety of large-scale applications have significant I/O requirements. Typical examples of I/O-intensive applications include long running simulations of time-dependent phenomena that periodically generate snapshots of their state [2], archives of raw and processed remote sensing data [5][11], multimedia and web-based applications [10], to name just a few. These applications share a common feature in that their storage and computational requirements are extremely high. It is worth noting that that a large number of applications are both CPU- and I/O intensive, making it difficult to predict the nature of workload conditions. This paper address the issue of applying autonomic computing techniques to computer systems under mixed workloads where I/O- and CPU- intensive applications (terms applications and jobs will be used interchangeably throughout this paper) are running in the systems. To facilitate the presentation of a analytic performance model, we build an architecture model as well as a queuing model for the computer systems. The analytical model is proposed and integrated into an adaptive admission controller for the systems designed for CPU- and I/O-intensive applications.

The rest of the paper is organized as follows. Related work is briefly discussed in section 2. Section 3 describes the architecture and queuing models. In Section 4 we construct the analytical performance model. Section 5 present experimental results of the admission controller, in which the performance model is incorporated. Finally, section 6 summarizes the main contribution of the paper.

2. Related work

Autonomic computing plays an important role for self-managing computer systems. Lin *et al.* proposed an application of a quality metrics framework to establish a quantitative definition of autonomic computing [7]. Sterrit and Bustard addressed the issue of making use of autonomic computing to provide a framework for dependability, which is an important property for all

computer-based systems [8]. These techniques, however, were not designed for I/O-intensive applications.

Increasing attention has been drawn toward I/O intensive applications. Kandaswamy *et al.* examined optimization techniques and architecture scalability. They evaluated the effect of the techniques using five I/O-intensive applications from both small and large applications domain. Very recently we developed two effective I/O-aware load-balancing schemes, which make it possible to balance I/O load by assigning I/O intensive sequential and parallel jobs to nodes with light I/O loads [10]. However, the above techniques are insufficient for autonomic computing platforms due to the lack of adaptability.

A self-managing computer system was developed by Bennani and Menasce [1]. The main purpose of the approach was to build a required mechanism into a computer system in order to satisfy requirements of the quality of service (QoS) for the system. Additionally, they evaluated the robustness of the self-managing computer system in a way that requirements of the QoS can be constantly met [4]. Gelenbe and Gellman have proposed an autonomous smart routing for network QoS [5]. They developed an autonomous adaptive called Cognitive packet network (CPN) which adaptively select paths so as to enhance the QoS to the end users bases on user defined QoS. Our approach is to modify an existing performance model to design self-managing computer systems [1]. Our work is in sharp contrast to theirs, because we aim at developing an adaptive admission controller needed for mixed workload conditions. In particular, we build a performance model that can handle a class of applications which are I/O and CPU intensive in nature.

3. System model

To facilitate the presentation the analytical performance model, in this section we describe the architecture and queuing models of a computer system.

The architecture of a computer system is depicted in Figure 1. The architecture is comprised of five major components, namely, an admission controller (1), an accepted job queue (2), a scheduler (3), a monitor (4), and an analytical model (5). The admission controller, a centerpiece of the system architecture, receives arriving jobs submitted to the system. It is the admission controller's responsibility to decide the acceptance of the jobs. Admission decisions are made in

accordance to information, e.g., nature of workload conditions, provided by the analytical model. At the time when a job is admitted to the system, the job is stored in the job-accepted queue and it resides in the queue until it is completed by the system.

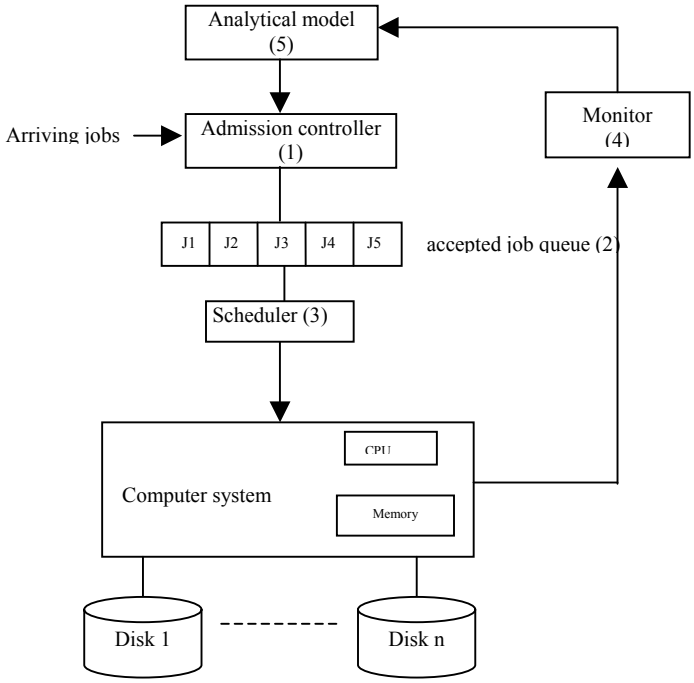

Figure 1. Architecture of the computer system

The scheduler assigns priorities for each job accommodated in the queue before executing them in the system. The monitor is intended to provide the analytical model with required information, including arrival and departure rates for CPU- and I/O-intensive jobs. Finally the analytical model, which will be detailed in Section 4 calculate the serving probabilities in terms of CPU- and I/O-intensive jobs. The probabilities will be used to predict workload conditions, which, in turn provide the admission controller with insights about which incoming jobs have to be accepted.

In order to clearly describe a mixed workload where CPU- and I/O-intensive jobs are running in the system, we built a queuing model for the computer system.

The queuing computer system shown in Figure 2 consists of a multithreaded server as well as two waiting queues. In this study, we consider workloads under which CPU- and I/O-intensive applications are running in the system. Thus, the first queue is dedicated for CPU-intensive jobs

admitted into the system, whereas the second queue is made available for I/O-intensive jobs. The load caused by CPU-intensive jobs can be modeled by an arrival rate, e.g., λ_{cpu} No./Sec., of all coming jobs with high demands on CPU resources. Similarly, an arrival rate, e.g., $\lambda_{I/O}$ No./Sec., of I/O-intensive jobs are used to reflect load imposed by those applications with significant disk I/O requirements.

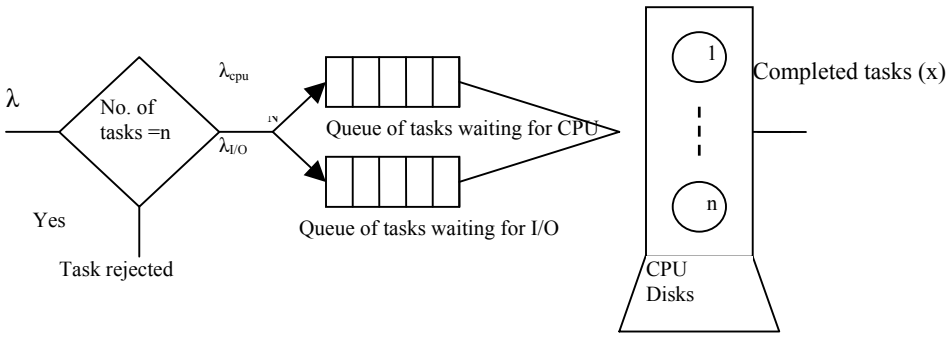

Fig. 2. The queuing model of a Computer System

The adaptive controller depicted in Figure 1 is intended judiciously choose the most appropriate jobs from these two queues to be executed in multithreaded server. The ultimate goal of the admission controller is to maximize the system utilization with respect to both CPU and disk I/O resources. To effectively achieve this objective, we constructed an analytical performance model. The analytical model, which is presented in the subsequent section, makes it possible to predict system performance trends in a system with mixed workloads. With the performance model in place, the admission controller is capable of accepting the most appropriate arriving jobs to make the computer system be readily adapted to workload conditions that are dynamic in nature.

4. Analytical model

The main contribution of this paper is the analytical performance model needed for computing the average response time *R*, average throughput *T*, and probability, e.g., p_{rej}, of an application being rejected by the admission controller when the workload exceeds a certain capacity. The response time and throughout can be expressed as functions of the arrival rates of CPU- and I/O-intensive jobs.

The analytical model can be characterized by a two-dimensional Markov chain. The Markov chain is employed to model two waiting queues, which are depicted in Figures 1 and 2. Without loss of generality, it is assumed that one queue is used for CPU-intensive jobs, while the other is dedicated for I/O-intensive jobs. In case where the workload is overloaded, the system can only simultaneously execute *n* jobs. After each successful completion of a job in the system, the admission controller makes an effort to select the best candidate job from the two waiting queues to bring into the system for execution.

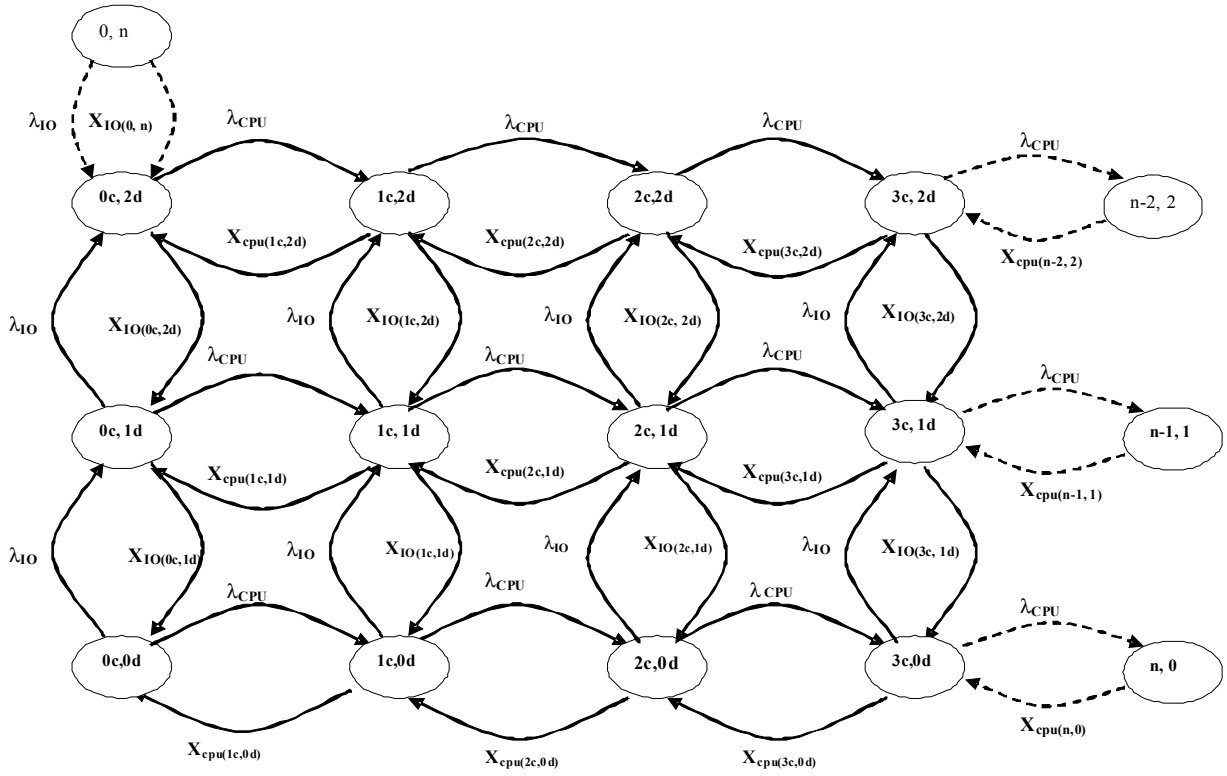

Figure 3. Markov chain for the computer system

We can use the discrete-time Markov chain depicted in Figure 3 to model the system *{c(t), d(t)},* where *c(t)* is the stochastic process representing the number of running CPU-intensive jobs, and *d(t)* is the stochastic process representing the number of I/O-intensive jobs running in the system. In this Markov chain, the one-step transition probabilities are:

$$\begin{cases} P\{i,j \mid i-1,j\} = \lambda_{CPU} & i \in [1,n], i+j \le n \\ P\{i,k \mid i,j-1\} = \lambda_{IO} & j \in [1,n], i+j \le n \\ P\{i,k \mid i+1,j\} = X_{CPU(i+1,j)} & i \in [0,n-1], i+j \le n \\ P\{i,k \mid i,j+1\} = \lambda_{IO(i,j+1)} & j \in [0,n-1], i+j \le n \end{cases} \quad (1)$$

Where we adopt the following short notation:

$$P\{i,k \mid i',k'\} = \ P\{c(t+1)=i, d(t+1)=k \mid c(t)=i', d(t)=k'\}.$$

In expression (1), $X_{CPU(i,j)}$ represents the rate at which a CPU-intensive job is completed under the condition that there are the system is running *i* and *j* CPU- and I/O-intensive jobs. Similarly, $X_{IO(i,j)}$ denotes the rate at which an I/O-intensive job completes its execution when there are *i* and *j* CPU- and I/O-intensive jobs running in the system.

The first equation in expression (1) accounts for the fact that a CPU-intensive job is admitted and loaded in the system. The second equation accounts for the fact that an I/O-intensive job is admitted and loaded in the system. The third equation models the fact that a CPU-intensive job is accomplished. The last equation in expression (1) models the case where an I/O-intensive job is completed.

Initially, the Markov chain starts at state (*c(t)=0, d(t)=0*), which means there is no CPU- and I/O-intensive jobs running in the system. If a CPU-intensive job arrives with arrival rate λ_{cpu}, the number of the CPU-intensive jobs in the system will be increased by 1, while the number of I/O-intensive jobs remains unchanged. Likewise, at the time when an I/O-intensive job arrives with arrival rate $\lambda_{I/O}$, the number of I/O-intensive jobs is increased by one, whereas the number of CPU-intensive will remain constant.

Let $p_{i,j} = \lim_{t\to\infty} P\{c(t)=i, d(t)=j\}, i,j \in [0,n]$ be the stationary distribution of the chain. In what follows, we derive a closed-form solution for the Markov chain.

First note that CPU-intensive jobs, we have the following equation:

$$p_{i,j} = p_{(i-1,j)} \frac{\lambda_{cpu}}{X_{cpu(i,j)}} \quad i \in [1,n], i+j \le n \quad (2)$$

$$= p_{(i-2,j)} \frac{\lambda_{cpu}}{X_{cpu(i,j)}} \times \frac{\lambda_{cpu}}{X_{cpu(i-1,j)}}$$

We obtain the following equation by applying equation (2) repeatedly.

$$p_{i,j} = p_{0,j} \frac{\lambda_{cpu}}{X_{cpu(i,j)}} \times \frac{\lambda_{cpu}}{X_{cpu(i-1,j)}} \times \cdots \times \frac{\lambda_{cpu}}{X_{cpu(1,j)}}$$

$$= p_{0,j} \frac{(\lambda_{cpu})^i}{\prod_{k=1}^{i} X_{cpu(k,j)}} \tag{3}$$

Likewise, we initially have the following equation for I/O-intensive jobs.

$$p_{0,j} = p_{0,j-1} \frac{\lambda_{IO}}{X_{IO(0,j)}}, j \in [1,n], i + j \le n \tag{4}$$

$$= p_{0,j-2} \frac{\lambda_{IO}}{X_{IO(0,j)}} \times \frac{\lambda_{IO}}{X_{IO(0,j-1)}}$$

We are now able to express $p_{o,j}$ by recursively applying equation (4):

$$p_{0,j} = p_{0,0} \frac{\lambda_{IO}}{X_{IO(0,j)}} \times \frac{\lambda_{IO}}{X_{IO(0,j-1)}} \times \cdots \times \frac{\lambda_{IO}}{X_{IO(0,1)}}$$

$$p_{0,j} = \frac{(\lambda_{IO})^j}{\prod_{k=1}^{j} X_{IO(i,k)}} \qquad j \in [0,n]. \tag{5}$$

By equations (3) and (5), all the values of $p_{i,j}$ are expressed as functions of the value $p_{0,0}$, the arrival and departure rates. Thus, $p_{i,j}$ can be rewritten as follows:

$$p_{i,j} = \begin{cases} p_{0,0} \dfrac{(\lambda_{cpu})^i (\lambda_{IO})^j}{\prod_{k=1}^{i} X_{cpu}(k,j) \prod_{k=1}^{j} X_{IO}(i,k)} & i + j <= m \\ P_{0,0} \dfrac{\rho^{(i,j)} X_{cpu}(k,j)^m X_{IO}(i,k)^m}{\prod_{k=1}^{i} X_{cpu}(k,j) \prod_{k=1}^{j} X_{IO}(i,k)} & i + j > m \end{cases} \quad \text{where } \rho = \frac{\lambda_{cpu} + \lambda_{IO}}{X_{cpu}(i,j) + X_{IO}(i,j)}. \tag{6}$$

$p_{0,0}$ is finally simplified as the following equation:

$$p_{0.0} = \left[1 + \sum_{i=1}^{m} \sum_{j=1}^{m-i} \frac{(\lambda_{cpu})^i (\lambda_{IO})^j}{\beta_{cpu}(i,j)\beta_{IO}(i,j)} + \frac{\rho(\lambda_{cpu} + \lambda_{iO})^m (1-\rho^{n-m})}{\prod_{i=1}^{m}\prod_{j=1}^{m-i} X_{cpu}(i,j)X_{IO}(i,j)} \right] \tag{7}$$

where $\beta_{cpu}(i,j)$ and $\beta_{IO}(i,j)$ are determined as follows:

$$\beta_{cpu}(i,j) = \prod_{k=1}^{i} X_{cpu}(k,j) \tag{8}$$

$$\beta_{IO}(i,j) = \prod_{k=1}^{j} X_{IO}(i,k) \tag{9}$$

The analytical model can be used to obtain the average response time, average throughput, and probabilities that the request is rejected. The probability that a job will be rejected by the admission controller is as follow:

$$p_{rej} = p(i,n) \text{ , or } p(n,j) \tag{10}$$

The following two equations represent the average through T and response time R of the system:

$$\begin{aligned} T &= \lambda_{cpu}\lambda_{IO}(1-p_{rej}) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n-i} X_{cpu}(i,j)X_{IO}(i,j)p_{i,j} \text{ .} \end{aligned} \tag{11}$$

$$R = \sum_{i=1}^{n}\sum_{j=1}^{n-i} \frac{(i+j)\times p_{i,j}}{X} \text{.} \tag{12}$$

5. Performance Evaluation

To validate the analytical model, we implemented an adaptive admission controller (hereinafter referred to as AC), in which the model is incorporated. In particular, we conducted a large number of trace-driven simulations. In this section, we compare the performance of the admission controller with baseline schemes. Throughout this section, the two baseline schemes, where the adaptive admission controller is not deployed, are referred to as CPU and IO, respectively. The CPU scheme give the highest priority to CPU-intensive applications, the IO

scheme favors IO-intensive jobs over CPU-intensive jobs.

The performance metric used in our simulations is *slowdown*, since jobs may be delayed because of waiting in queues. We extend the definition of slowdown by incorporating I/O access time. The extended definition of slowdown for a job k is given as:

$$slowdown(k) = \frac{T_{wall}(k)}{T_{CPU}(k) + T_{IO}(k)}, \tag{13}$$

where $T_{wall}(k)$ is the total time the job spends running, and accessing I/O.

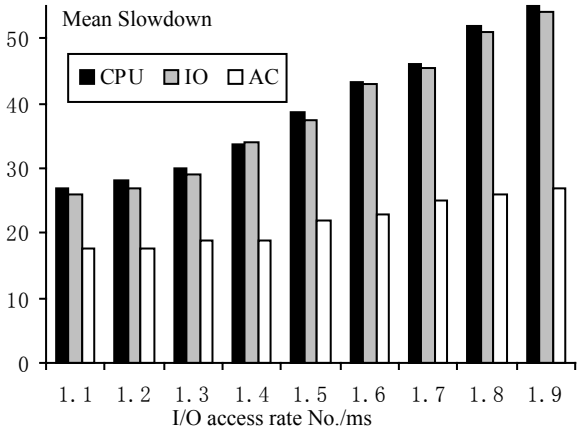

Figure 4. Mean slowdowns as a function of I/O access rate

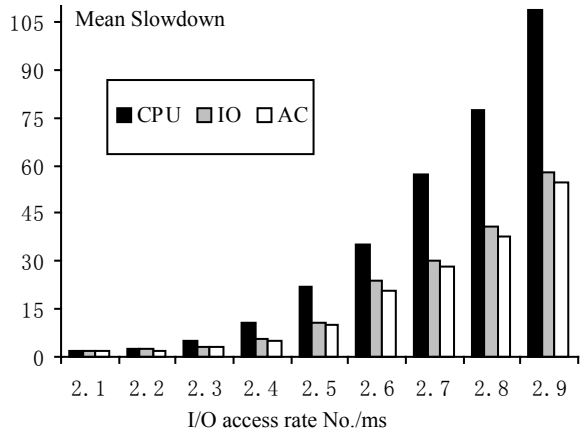

Figure 5. Mean slowdowns as the I/O access rate increases on the trace.

In our first experiment, s*lowdown* is measured as a function of I/O access rate in the range between 1.1 and 1.9 No./ms with increments of 0.1, as shown in Figure 4. We observed from Figure 4 that the mean slowdowns increase with the increasing values of the I/O access rate. The results also reveal that the AC scheme significantly outperforms the two alternatives. This is mainly because AC takes the full advantages of the analytical model to predict the nature of the future workload conditions, thereby noticeably improving the utilizations of computing resources in the system.

To stress the I/O workload, we keep the page fault rate at a very low value. Figure 5 plots the slowdown as a function of I/O access rate for three schemes under the workload conditions that the I/O access rate is chosen in the range between 2.1 and 2.9 No./ms with increments of 0.1. It is also observed from Figure 5 that when I/O access rate is higher than 2.3No./sec, AC consistently

outperform perform CPU and IO. Again, this is because AC improves the utilization of disks, which dominates the overall performance when the system experiences high loads in disk resources.

6. Conclusion

In this paper, we developed a performance model for self-manage computer systems under dynamic workload conditions, where both CPU- and I/O-intensive applications are running in computer systems. To validate the analytical model, we simulated an adaptive admission controller in which the model is incorporated. Experimental results show that the controller is capable of achieving high performance for computer systems under workloads exhibiting high variabilities. As a future research direction, we will apply this performance model to large-scale distributed systems.

Acknowledgements

This work was partially supported by a start-up research fund (103295) from the research and economic development office of the New Mexico Institute of Mining and Technology.

Reference

[1] D.A Menasce and M.Bennani, "ON THE USE OF PERFORMANCE MODELS TO DESIGN SELF-MANAGING COMPUTER SYSTEMS," *Proc. Computer Measurement Group conf., Dec.7-12, 2003*

[2] T. Tanaka, "Configurations of the Solar Wind Flow and Magnetic Field around the Planets with no Magnetic field: Calculation by a new MHD," *Journal of Geophysical Research*, pp. 17251-17262, Oct. 1993.

[3] D.A Menasce and M. Bennani, "Assessing the Robustness of Self-Managing Computer System under Highly variable workload," *Proc. of the International Conference on Autonomic Computing (ICAC '04).*

[4] E. Gelenbe, M. Gellman, R. Lent, P. Liu, P. SU," Autonomous Smart Routing for Network QoS," *Proceeding of the International Conference on Autonomic Computing (ICAC '04)*

[5] C. Chang, B. Moon, A. Acharya, C. Shock, A. Sussman, J. Saltz, "Titan: A High-Performance Remote-sensing Database," *Proc. of* International Conference on Data Engineering, *1997.*

[6] G. Tziallas and B. Theodoulidis," Building Autonomic Computing Systems Based On Ontological Component Models and a Controller Synthesis Algorithm," *Proc. of the 14th International Workshop on Database and Expert Systems Applications (DEXA '03)*

[7] P.Lin, A.MacArthur, J.Leaney," Defining Autonomic Computing: A software Engineering Perspective," *Proc. Australian Software Engineering Conf. 2005.*

[8] R.Sterritt and D.Bustard, "Autonomic Computing-A Means of Achieving Dependability?" *Proc. the 10th IEEE Int'l Conference and workshop on the Engineering of Computer-Based Systems (ECBS'03).*

[9] M. Kandaswamy, M.Kandemir, A.Choudhary, D. Bernholdt, "Performance Implications of Architectural and Software Techniques on I/O Intensive App-lications," *Proc. Int'l Conf. Parallel Processing, 1998.*

[10] X. Qin, H. Jiang, Y. Zhu, and D. Swanson, "Towards Load Balancing Support for I/O-Intensive Parallel Jobs in a Cluster of Workstations," *Proc. of the 5th IEEE Int'l Conf. Cluster Computing* (Cluster 2003), Hong Kong, Dec. 1-4, 2003.

[11] R. Ferreira, B. Moon, J. Humphries, A. Sussman, J. Saltz, R. Miller, and A. Demarzo, "the Virtual Microscope," *Proc. of the 1997 AMIA Annual Fall Symposium*, pp. 449-453, Oct. 1997.