On the Impact of Management on the Performance of a Managed System: A JMX-Based Management Case Study

Abdelkader Lahmadi, Laurent Andrey, and Olivier Festor

LORIA - INRIA Lorraine - Université de Nancy 2
615 rue du Jardin Botanique
F-54602 Villers-lès-Nancy, France
{Abdelkader.Lahmadi,Laurent.Andrey,Olivier.Festor}@loria.fr

Abstract. Studying the performance of a distributed system without taking care on the impact of its management system will falsify the understanding of its overall performance, especially its productivity. We propose a metric called *MIM (Management Impact Metric)* to evaluate this impact by varying one or several impact factors related to the management system within a management strategy of the managed system. We show the accuracy and interest of our metric on a managed J2EE application server that uses a management architecture based on the JMX standard.

Managed systems performance, productivity, manageability.

1 Introduction

The essence of modern networks and services (home gateways, sensor networks, application servers, grids) lies in the optimal utilization of resources within a dynamic and large working environment. A key component required in this respect is the management framework that monitors these systems and orchestrates their activities to improve and maintain their performance. The goal of management is to ensure that the managed systems operate with the efficiency and effectiveness predefined in the quality of service parameters. Since current network management architectures are often integrated in the service activities, it is essential to be able to know the overhead of these management activities and their impact on the overall performance of the managed systems. A basic question we are trying to answer here is: *How do management activities impact the overall managed system performance ?* and *How can we minimize this impact ?*

Management architectures and their associated activities are becoming very complex and diverse. Over the last 20 years, new and enhanced management architectures appeared, varying from OSI and SNMP(v1,v2,v3) to Web Sservices-based management including Java specific approaches like the JMX standard which became very popular. Such management architectures have the following characteristics:(1) their activities are essential to manage the system;(2) they offers a set of management strategies that operate differently on the managed system (e.g., polling vs notification); (3) they can severely impair the performance of the user's work (referred to as **productivity**) if their

overhead cost per management strategy is not well defined and studied. In the literature, many studies [1,2] evaluate the performance of these management architectures, and they focus especially on comparing their performance. However, the question of how management activities impacts the performance of managed systems has not been studied so far. The variability of performance captures the impact of management on the performance of a managed system.

A metric that quantifies this impact should be defined. It must put in relation the performance of management and managed systems. To address this issue, we propose an analytical model of this impact that combines the performance of the management and the managed systems over varying management profiles and under an impact factor.

The paper is structured as follows: Section 2 reviews related works. Section 3 presents the set of the impact modifiers and factors of a management system that impact the performance of a managed system. Section 4 introduces the analytical model of our impact function. Section 5 presents an example of using the impact metric on a JMX based management activity of the JBoss application server. Section 6 contains a brief summary of this contribution as well as an outlook.

2 Related Work

Several papers separate investigations on the performance of distributed systems and the performance of management systems. Woodside and al. [3] define the performance referred as **productivity** of a system as the relation between the rate of providing valuable services, the quality of those services and the cost of providing those services. Another definition is proposed in [4], where performance is viewed as the response time, seen by a user under normal working conditions, coupled with the cost of the system - hardware requirements - per user. We will adopt the performance metric proposed for distributed systems to assess the performance of the managed and management systems. In fact, we will use the same productivity definition as defined in [3] to quantify the performance of the managed system under an impact factor.

Several studies are related to the performance evaluation of specific management systems. The focus of most of these studies has been to model the performance of management architectures and their associated cost. Their performance models quantify response time of agents [1,5], the volume of management traffic [2] and resources usage [6,7]. Nevertheless, all performance studies related to management architectures that take as elementary performance metrics : response time, management requests rate and resource usage will benefit from our management impact metric. Our proposed metric is based on the efficiency [8] of the managed system. This function is defined as the useful work of the managed system divided by the total work (productivity+manageability). By continuously computing the impact metric, a management system will regulate its activities to minimize its impact or adapt the management profile (strategy) parameters within the managed system. This metric provides an auto-tuning criterion for the management system [9], which allows the managed system to be more self-managing and more efficient.

3 Management Profiles

Despite the wide variety of management technologies and products, most management system infrastructures fall into an architecture pattern referred to as Manager-Agent. There are three basic components in this architecture: managed system, agent and management applications. The management application is responsible for providing the infrastructure and user interfaces to manage a system and it is conducted by a management profile or strategy that defines manageability tasks and patterns (see figure 3).

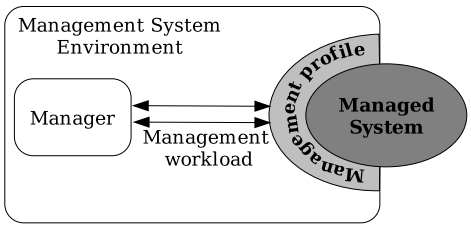


Fig. 1. The Management system components

> *Definition:* A management profile is a quantitative characterization of how a system is managed. The profile summarizes key interaction parameters between the management system environment and the managed system.

A management profile covers the most important parameters related to the management system and its interaction with the managed one. It is important to identify those parameters, which, if varied, will change the management profile within the managed system. Parameters that are changed from a management profile to another are called *impact modifiers*. These modifiers, if varied, will have a significant impact on the performance of the managed system. The impact modifiers might improve, maintain or degrade a given managed system performance. Examples of impact modifiers are the agent deployment patterns within the managed system, management tasks and their operation patterns (polling-driven or events-driven), the design patterns of management objects within the managed system. The management profile is controlled by the management workload that includes management requests. The management workload characterization parameters represents the set of *impact factors*.The impact factors denote a set of impact variables, determined by the management profile within the managed system. The impact metric is analysed by varying the impact factor within a management profile. Table 1 displays a non exhaustive list of management profiles and factors that affect the performance of a managed system. Within these profiles, three parameters are chosen and studied in more details in the paper. For each parameter, the intuitive impact on the performance of the managed system is listed.

3.1 Management Tasks

As defined in [10], management system tasks are the following:

– Monitoring: the ability to capture runtime and historical events from a particular component. This task is continuous over the execution time of the managed system and it is executed concurrently with users on each monitoring cycle.
– Tracking: the ability to observe aspects of a single unit of work or thread of execution across multiple components (e.g., tracking messages from senders to receivers). This task is executed less frequently than the monitoring on a period from the execution time of multiple components within the managed system.
– Control: the ability to alter the runtime behavior of a managed component (e.g., changing the logging level of an application). The execution of this task can result from the first two ones when problems detected by monitoring or tracking need to be resolved by controlling the managed system. This task is executed on a precise period from the execution time of the managed system.

Management profile parameters (Impact modifiers)	**Management Impact factors**
Management approaches: - Client-server - Hierarchical static - Weak mobility - Strong mobility	- Number of management requests - Number of notifications - Number of management objects within the managed system - Management requests mixes : * Read requests * Write requests
Management enabling technologies: SNMP, RMON/RMON2, CMIP, WBEM, PBN, Mobile agents, DCOM, CORBA, JMX	
Management tasks: - Monitoring - Tracking - Control	
Management operation models: - Polling - Notification	
Agent deployment models: - Deamon - Component - Driver	
Management instrumentation models: - External - Internal	

Table 1. The management profile parameters and impact factors.

Thus, it is easy to see that the monitoring task will introduce a periodically impact on the performance of a managed system. However, the control and tracking tasks do not permanently affect the performance of the managed system (An example of a management profile for the JBoss server is given in section 5).

3.2 Management Agents Deployment Models

The way in which the management agent is deployed within the managed system is an impoortant profile parameter. In [11], the authors identify 3 management agent deployment models: daemon, component, and driver. In the daemon model, the agent owns its own process separate from the application. In this case, the managed component and the agent do not share the same resources and may running on two different hosts. The sole overhead introduced by the agent on the managed component is the communication cost to retrieve management data from the managed resource. In the component

model, the agent runs in the process owned by the application and they share the same resources. Hence, the overhead of the managers interacting with the agent is added to the resources used by the managed application. The driver deployment model is similar to the component model. Rather than a component, the agent become the core of the system. In this case, all manageability work is executed concurrently with the users work.

3.3 Management Instrumentation Patterns

This management profile parameter specifies the way in which the management object (e.g.,MBeans for JMX) retrieves the management data from the managed resource. Two patterns are identified [11]: internal instrumentation and external instrumentation. In the internal instrumentation the management object is part of the managed resource and management tasks are executed directly on it. External instrumentation is defined and executed outside the managed resource. From these definitions, internal instrumentation might affect more significantly the performance of the managed resource rather than the external one.

We can see clearly that the choice of a management profile or a strategy rather than another will modify the potential impact of the management system on the performance of the managed system.

4 The Impact Function

The impact function is designed to capture the performance variability of a managed system under a management profile at a given impact factor value. We named the impact metric MIM as *Management Impact Metric*. $MIM(k)$ is a function that maps the impact factor k to a value within the closed interval [0, 1]. It indicate whether a performance degradation has occurred, and includes an indication of the degree of that degradation. The $MIM(k)$ function distinguishes between an unacceptable impact of a management system (for which $MIM(k)$ is close to 1) and an acceptable impact (for which $MIM(k)$ is close to 0).

Instead of productivity, which is the performance metric (production work) related to the managed system, we name the performance of the management system as **manageability** (management work) [10]. We denote $F(k)$ as the productivity of the managed distributed system and $G(k)$ as the manageability of the management system at an impact factor k. Hence, the efficiency of the managed system at the impact factor k is given by:

$$E(k) = \frac{F(k)}{F(k)+G(k)} \tag{1}$$

We adopt the productivity F(k) of the managed system or the manageability G(k) of the management system defined in [3] as follows:

$$F(k) = \lambda_1(k).\frac{f(k)}{C(k)}, G(k) = \lambda_2(k).\frac{g(k)}{C(k)} \tag{2}$$

Where $\lambda_1(k)$, $\lambda_2(k)$ are respectively the users work throughput in responses/sec of the managed system and the management throughput in responses/sec of the management

system at an impact factor k. The function $f(k)$, respectively $g(k)$, is an average value of each response calculated from its quality of service at the impact factor k. The value function $f(k)$ is determined by evaluation of the performance of a system (managed and management ones), and may be a function of any appropriate system measure including delay measures (mean, variance or jitter, probability of delay exceeding a threshold). In this work we will consider only the mean response time $T(k)$ at the impact factor k, normalized to a target value $\bar{T}$ (response time quality of service), in the following value function [12]:

$$f(k) = \frac{1}{1 + (\frac{T(k)}{\bar{T}})} \tag{3}$$

The target value $\bar{T}$ is an optional upper bound for the delay that can be specified for an impact state to be acceptable. If we do not specify the delay target value, the value function $f(k)$ (respectively $g(k)$) will be the following [3]: $f(k) = \frac{1}{T(k)}$. In this case the productivity is given by:

$$F(k) = \frac{\lambda_1(k)}{T_1(k)}.C(k), G(k) = \frac{\lambda_2(k)}{T_2(k)}.C(k) \tag{4}$$

$C(k)$ is the cost function at the impact factor k, expressed as the running cost per second to be uniform with λ_1 (respectively λ_2). The cost may be a function of any appropriate weighted sum of resources utilization metrics such as cpu, memory and network. The weight coefficients imply their importance on the managed system. Then $C(k) = a.CPU(k) + b.Memory(k) + c.Network(k)$, where a, b and c are the weights of the resources consumed either by the managed system or the management one. The function $E(k)$ denote the efficiency of the managed system associated with an impact state k, under a management profile characterized by its manageability $G(k)$. The impact function $MIM(k)$ relating the efficiency of the managed system at two different impact states is then defined as:

$$MIM(k_0, k) = 1 - \frac{E(k)}{E(k_0)} \in [0, 1] \tag{5}$$

This is the impact function that is used through the paper. The intuition behind our function is to capture the behavior of the performance of the managed system. This behavior is observed from a baseline configuration and define which cases the performance of the system is unacceptable under a management impact factor. The efficiency $E(k_0)$ denote a baseline configuration of the managed system with a value k_0 of the impact factor. A way to determine the baseline configuration is to suspend all management activities for a period and measure the performance of the target managed system during that period as the baseline. In this case, the value of the manageability $G(0) = 0$ and $E(0) = 1$. Then, for the baseline configuration of the managed system, the baseline efficiency is equal to 1 and the impact function is given by :

$$MIM(k) = 1 - E(k) = 1 - \frac{F(k)}{F(k) + G(k)}; \text{ where } k \geq 1 \tag{6}$$

From the management efficiency aims, a managed system is isoefficiency managed if its overall efficiency is maintained at a desired value such as $0 \leq E(k) \leq 1$ which implies that the useful work performed by the managed system (productivity) should grow

at least at the same rate as the management overhead (manageability) to keep managed system efficiency constant. Let α denote the value of $G(k)$ normalized with respect to $F(k)$. α denotes the fraction from the managed system productivity attributed to the management activities. Then, $G(k) = \alpha.F(k)$ and we obtain $MIM(k) = \frac{\alpha}{1+\alpha}$. Figure 2 depicts the behavior of the impact metric according to a linear fashion of manageability work and the managed system productivity. When $\alpha = 1$, the manageability has the same rate as productivity and in this case we reach the bound of the isoeffective management. Beyond that value, the management strategy becomes ineffective.

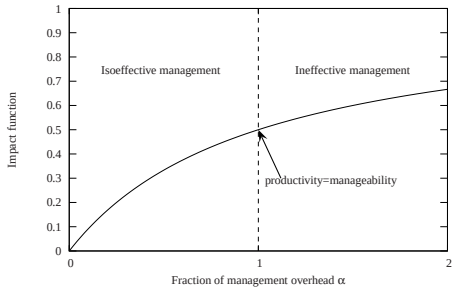


Fig. 2. The impact metric behavior of a linear model between manageability and productivity.

4.1 Computing the Impact Function

Calculation of this metric depends on the performance technique used to evaluate both the performance of the managed system and the management one. Analytical and simulation techniques are more suitable to calculate it. These two techniques are more flexible [13] than the measurement technique and they can handle a wide variety of configurations of the managed system by varying the impact factor and management profiles. Their disadvantage is that they need the availability of analytical models both for the managed system and the management one. It is not easy to obtain them for complex distributed systems. We define the following steps to calculate the impact function. We first determine the baseline performance of the target system. The baseline value will capture the performance of the system under fixed states of user's work and scalability values (e.g, a fixed number of users, a fixed number of requests per unit of time, a fixed number of servers, etc). By varying the baseline configuration of the managed system we can capture its performance under different users workload or scalability factors. Secondly, we define the management profile of the managed distributed system and the impact factor. The productivity of the managed system and the manageability of the management one are computed as follows:

1. Fix a management profile which includes management strategy parameters as described in section 3.
2. Choose an impact factor and fix other factors.
3. Managed system productivity prediction: the productivity is computed by measuring the average system throughput in number of responses per second and the average response time per response. Measure the running cost per second on the

managed system. The cost is the sum of resources utilization expressed on a measurement unit (e.g., average percent).
4. Management system manageability prediction : needs measurement of the average management throughput in number of management responses per second from the agent and the average response time. The cost represents the overhead running cost per second due to management activities.
5. Compute the impact function according to equation 2 in case of use of the value function $f(k)$ or equation 4 in case of use of only the response time $T(k)$.
6. Vary the chosen impact factor value and goto the step 3.

Computing the impact metric by varying the impact factor k within an interval allows us to find its bound value, beyond which the management highly impacts the productivity. In that case, a stronger justification for its benefit on the overall service delivery is required. The bound value of the impact factor k correspond to an impact metric value equal to 0.5.

5 Experimental Assessment

To assess the applicability of our impact metric, we did study it in the context of a JMX based management of a J2EE application server such as JBoss [14]. The management profile that we evaluate from the MBean server within the JBoss server is the following management profile

- Client-Server approach,
- JMX based management,
- Monitoring task,
- Polling based monitoring by using *getAttribute* operation.
- The JMX agent is deployed as a driver: the MBean server within the JBoss server is implemented as the kernel of the server,
- The management instrumentation is internal and the attribute that we solicit is retrieved from the system. The *getFreeMemory* operation, at the JMX level, calls the JVM system function *Runtime.freeMemory* to retrieve the amount of free memory in the JVM.

We take the management input workload expressed in number of requests per second as the impact factor. Previous work [15] gives us an idea of a realistic range for this factor. We vary the management workload from 1 to 400 requests/s. If we go beyond this value, the number of lost management requests increase consequently due to timed-out RMI connections (we keep the default value of 15 seconds of the RMI timeout).

5.1 Testbed Setup

We used a JBoss v3.2.1, running on a Sun JDK v1.4.2 and hosted on a bi-processor PIII 550MHZ with 512MB RAM, with the Slackware 9.1 operating system. The testbed workstations are connected to a 100Mbps Ethernet switch. The testbed is alone on this network, and we can assume that network is not a limitation. To emulate users activity against the JBoss server, we use RUBiS [16] as a benchmarking tool to evaluate the performance of the JBoss server and to measure its productivity. RUBis is modeled after

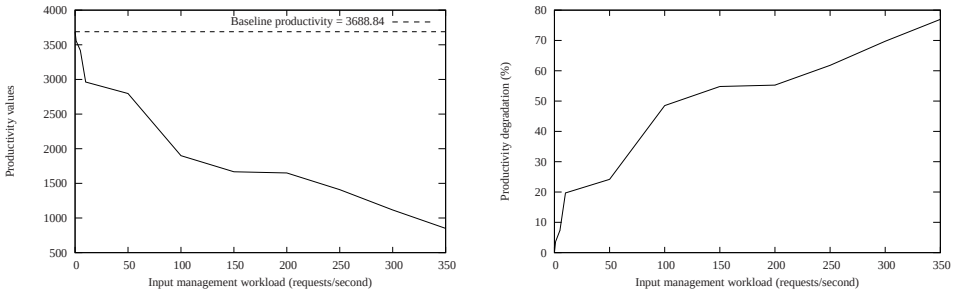
an auction site (eBay.com). For our experiments, we chose to use an EJB variant from RUBiS, which is entirely based on stateless session beans, as it is the best performing EJB variant according to [17]. For the measurements shown here, we used a steady users workload. The number of emulated users is kept constant (100 clients) and they have a mean *thinking time* of 7 seconds. To capture the management impact, we developed an emulator management client that implements only the monitoring task. The management emulator client sends a number of requests per second, that represents the impact factor, by using the *getAttribute* operation exposed by the MBeanServer within the JBoss server. We retrieve the value of the *FreeMemory* attribute from the *ServerInfo* MBean. In the current work, all management requests solicited the same MBean and the same attribute. Each measurement has a duration of 15 minutes and a warm-up period of 2 minutes both for users emulator client and management emulator client. Our experiments proceed as follows. The set of 100 emulated users were running. They execute browsing-only transactions against the JBoss server. The response time and the average number of responses per second is measured on the users client emulator side. Concurrently to users workload, the management client emulator executes a fixed number of *getAttribute* requests per second against the MBeanServer within the JBoss server. On the management client side, we measure the number of management responses per second and the response time per request. We use the *sar* [18] tool to measure the resource utilization (cpu,memory and network) on the JBoss server side. Response times measurements are taken using the *System.currentTimeMillis()* method included in the API of Sun's JDK.

5.2 Experimental Results

Throughput (responses/s)	Response time (second)	Ressource utilization (average %)			Productivity value
		CPU	Memory	Network	
15	0.466	88	24	2,6	3688.84

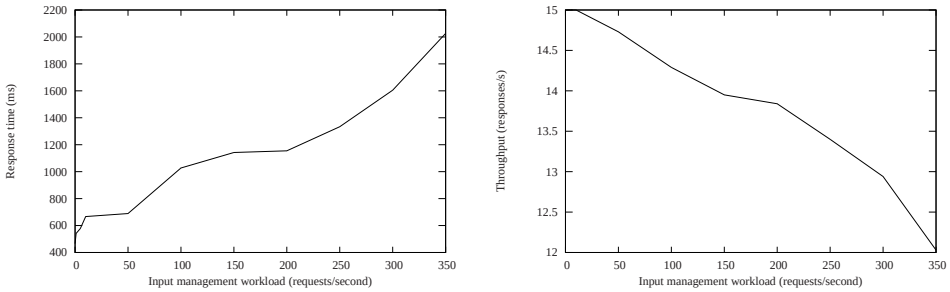
Table 2. Results of the baseline productivity of the JBoss server without any ongoing monitoring activity and under a steady users workload (100 browsing clients).

In a first step, we measure the baseline performance of the JBoss server without management workload and where only the users steady workload is executed against the server. Table 2 displays baseline average values of the throughput, response time and resource utilization and their corresponding baseline productivity of the JBoss server. In a second step, we vary the management input workload and measure the productivity, the manageability and the impact metric at each impact factor value. From the plot of figure 3 we can observe a decrease of the server productivity values due to the high management input load and the management overhead within the server. The productivity degradation from the baseline configuration, where only the users workload is executed against the JBoss server, varies between 24% for 50 management requests per second and 74% for 400 requests/s. The JBoss server productivity degradation is caused by the increase of response times $T(k)$ and the decrease of the throughput $\lambda(k)$ as shown in figure 4. From figure 5, we see the increase of manageability. This is trivial, due to the increase of the management load. Figure 6 shows the increase of the impact



(a) JBoss server productivity decrease **(b) JBoss server productivity degradation in (%)**

Fig. 3. Productivity decrease of the JBoss server (a) and the corresponding degradation from the baseline server state for a steady users workload and under an increasing management workload input in number of requests per second using the *getAttribute* operation that retrieves the FreeMemory attribute from the ServerInfo MBean.



(a) JBoss server response times increase **(b) JBoss server throughput decrease**

Fig. 4. Response times increase (a) and throughput decrease (b) of the JBoss server under an increasing management workload and a steady users workload (100 browsing clients).

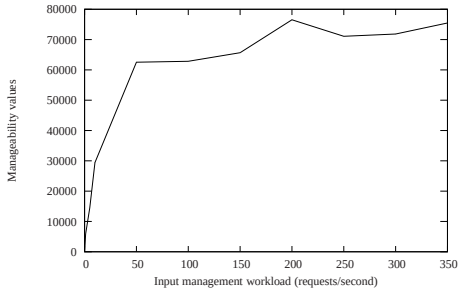
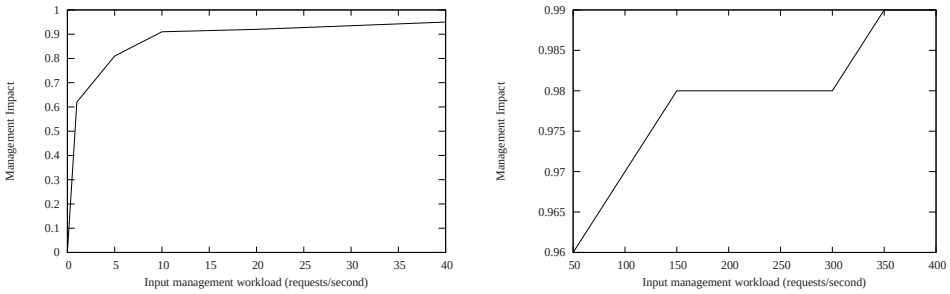


Fig. 5. Manageability growth by increasing the management workload input in requests per second by using the *getAttribute* operation that retrieves the FreeMemory attribute from the ServerInfo MBean.



(a) Low management workload **(b) High management workload**

Fig. 6. Management Impact Metric behavior for the JBoss server under a low management workload input less than 50 req/s (a) and a high management workload input greater than 50 req/s (b) by using the *getAttribute* operation that retrieves the FreeMemory attribute from the ServerInfo MBean.

metric. Here we note that the management agent (the MBeanServer from the JMX terminology) of the JBoss server is the core of the application server (driver model). It is the components container for the server design level (not ejb level). Thus, the input management workload is executed concurrently with user's workload and the impact is quickly seen.

6 Conclusion and Future Work

We defined a metric that captures the impact of a management profile or strategy on the performance of a managed system. The objective is to provide a metric that evaluates a management strategy and enables it to support its intended target environment. Our experiments confirmed that a management strategy within a managed system is associated with a degradation of the overall system efficiency, which may not be acceptable in all cases. Here we should note that our experiments overestimated the manageability, because we take into account a high management input rates and we use the same value of the running resource consumption both for computing manageability and productivity on the server side, which explains the fast growth of the impact metric. A more accurate analysis and estimation of the running resource utilization of manageability should be done by using complementary techniques such as the profiling component resource consumption presented in [19]. Our impact metric is computed using the measurement technique to evaluate the performance of the managed JBoss server. This technique needs a lot of time to collect measurements to be credible and requires available system prototypes. Other techniques (analytical modeling or simulation) are more flexible and less time consuming than the measurement technique [13]. They will be used for evaluating the performance of the managed system under a given management impact factor and computing the impact metric. In this work, we examine only the case of a degradation impact of management activities on the performance of a managed system. We plan to investigate other cases where the impact of management activities, such as load balancing and admission control, might improve and maintain the performance of a managed system. In parallel, we continue to setup and run performance tests of

the mannagemet plane on large sale systems e.g. Grids to massively deploy agents and evaluate the behavior of both manager and agent systems.

References

1. Pavlou, G., Flegkas, P., Gouveris, S., Liotta, A.: On management technologies and the potential of web services. Communications Magazine, IEEE **42** (2004) 58–66 ISSN: 0163-6804.
2. Neisse, R., Vianna, R.L., Granville, L.Z., Almeida, M.J.B., Tarouco, L.M.R.: Implementation and bandwidth consumption evaluation of SNMP to web services gateways. In: NOMS (Network Operations & Managament Symposium). Volume 9. (2004)
3. Jogalekar, P., Woodside, C.: Evaluating the scalability of distributed systems. IEEE Trans. Parallel Distrib. Syst. **11** (2000) 589–603
4. Burness, A., Titmuss, R., Lebre, C., Brown, K., Brookland, A.: Scalability evaluation of a distributed agent system. Distributed Systems Engineering **6** (1999) 129–134
5. Pattinson, C.: A study of the behaviour of the simple network management protocol. In: 12th International Workshop on Distributed Systems: Operations and Management (DSOM). (2001) 305–314
6. Subramanyan, R., Miguel-Alonso, J., Fortes, J.: A scalable SNMP-based distibuted monitoring system for heterogeneous network computing. In: Proceedings of the 2000 ACM/IEEE conference on Supercomputing, IEEE Computer Society (2000) 14
7. Pras, A., Drevers, T., de Meent, R.V., Quartel, D.: Comparing the performance of SNMP and web services-based management. eTransactions on Network and Service Management(eTNSM) **1** (2004)
8. Mitra, A., Maheswaran, M.: Measuring scalability of resource management systems. Technical Report SOCS-04.5, School of Computer Science,McGill University (2004)
9. Diao, Y., Hellerstein, J., Parekh, S., Griffith, R., Kaiser, G., Phung, D.: Self-managing systems: A control theory foundation. In: 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05), Greenbelt, Maryland (2005) 441–448
10. Murray, J.: Designing manageable applications. WEB Developper's Journal (2003)
11. Kreger, H., Harold, W., Willamson, L.: Java and JMX: Building Manageable Systems. Addison-Wesley (2003) ISBN: 0672324083.
12. Grama, A., Gupta, A., Kumar, V.: Isoefficiency function: A scalability metric for parallel algorithms and architectures. IEEE PDT **1** (1993) 12–21
13. Jain, R.: The art of Computer Systems Performance Analysis. John Wiley & Sons, Inc (1991) ISBN : 0-471-50336-3.
14. JBoss: The professional open source company. http://www.jboss.org (1999)
15. Lahmadi, A., Andrey, L., Festor, O.: Performances et résistance au facteur d'échelle d'un agent de supervision basé sur jmx : Méthodologie et premiers résultats. In: Colloque GRES 2005 : Gestion de REseaux et de Services, Luchon, France. Volume 6. (2005) 269–282 ISBN : 2-9520326-5-3.
16. ObjectWeb: Rubis: Rice university bidding system. http://rubis.objectweb.org (2002)
17. Cecchet, E., Marguerite, J., Zwaenepoel, W.: Performance and scalability of ejb applications. In: Oopsla'02. (2002) http://rubis.objectweb.org/download/perf_scalability_ejb.pdf.
18. Godart, S.: system performance tools for linux os. http://perso.wanadoo.fr/sebastien.godard/ (2003)
19. Stewart, C., Shen, K.: Performance modeling and system management for multi-component online services. In: The 2nd Symposium on Networked System Design and Implementation (NSDI2005),Boston, MA, USENIX (2005)