RC25200 (W1108-023) August 8, 2011
Computer Science

IBM Research Report

Dynamic Resource Allocation via Distributed Decisions in Cloud Environment

Trieu C. Chieu, Hoi Chan
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Dynamic Resource Allocation via Distributed Decisions in Cloud Environment

Trieu C. Chieu and Hoi Chan
IBM T. J. Watson Research Center
19 Skyline Drive
Hawthorne, NY, USA

e-mail: {tchieu,hchan}@us.ibm.com

Abstract—*The adaptation of virtualization technologies and the Cloud Compute model by Web service providers is accelerating. These technologies commonly known as Cloud Compute Model are built upon an efficient and reliable dynamic resource allocation system. Maintaining sufficient resources to meet peak workloads while minimizing cost determines to a large extend the profitability of a Cloud service provider. Traditional centralized approach of resource provisioning with global optimization and statistical strategies can be complex, difficult to scale, computational intensive and often non-traceable which adds to the cost and efficiency of Cloud operation, especially in industrial environments. As we have learned in real life, the most efficient economic system is the one that provides individuals with incentives for their own decisions. It is also true for computing systems. In this paper, we present an architecture for dynamic resource provisioning via distributed decisions. We will illustrate our approach with a Cloud based scenario, in which each physical resource makes its own utilization decision based on its own current system and workload characteristics, and a light-weight provisioning optimizer with a replaceable routing algorithm for resource provisioning and scaling. This approach enables resource provisioning system to be more scalable, reliable, traceable, and simple to manage. In an industrial setting, the importance of these characteristics often exceeds the goal of squeezing the absolute last CPU cycles of the underlying physical resources.*

Keywords-Cloud computing; scalability; virtualization; virtual machine

I. INTRODUCTION

Virtualization based Cloud Computing [1,2,3] is becoming an increasingly popular enterprise computing model in which applications share the underlying computing resources by running in isolated Virtual Machines (VMs). The vast processing power of Cloud Computing is made possible though distributed, large-scale computing clusters, often in concert with server virtualization software, like VMware ESX Server [4] and Xen [5], and parallel processing. The Cloud Computing model enables users to access supercomputer-level computing power elastically on an on-demand basis, freeing the users from the expense of acquiring and maintaining the underlying hardware and software infrastructure and components. With the Cloud computing model, the providers who achieve the maximum economies of scale while maintaining client satisfaction will result in increased profit. The key to achieving the maximum economies of scale in a Cloud environment is a flexible and efficient provisioning and scaling system that fully utilizes the underlying physical hosts and adjusts to changing workload demand. Typically, efficient provisioning requires two distinct steps or processes: (1) initial static planning stage: the initial set of VMs are grouped, classified and deployed onto a set of physical hosts; and (2) dynamic resource provisioning [6,7,8]: the provisioning of additional resources, creation and migration of VMs, dynamically responds to fluctuating workload. In contrast to Step 1 which is usually performed at the initial system set up time and may only be repeated for overall cleanup and maintenance on a monthly or semi-annually schedule, step 2 runs continuously at production time. Step 1 has been researched extensively with various optimization techniques and strategies while step 2 becomes increasingly important as the size of the Cloud environment [9] grows with complex workload patterns. This paper focuses on step 2.

Traditional approach utilizes a centralized provisioning and scaling system to continuously monitor the performance and capacity characteristics of each of the VMs and their physical hosts and makes provisioning and scaling decisions based on the overall utility of the entire cluster as a whole. The centralized decision making system [10] assumes that a single decision maker will have access to all of its managed systems and possesses all available knowledge and information related to them and has to make decisions in order to achieve a certain objective. These decisions are often based on statistical models [11,27,28] and are often difficult to trace should problems arise. The centralized approach works well as long as the number of VMs and physical resources remains manageable, but becomes computationally expensive and difficult to manage as the number of VMs and physical hosts grows. More importantly, centralized systems may ignore the specific characteristics of each of the VMs and physical hosts due to their inherent complexity, each of them may be unique, e.g. some systems are more expensive to operate, some may be offline soon for maintenance. In the extreme cases, it will put pressure on the limited sources and affects the performance of the entire Cloud infrastructure. As Cloud computing grows in acceptance and service providers [9] expand with physical and software resources scattered and shared around multiple locations and even across continents, these resources become so diverged and large in number that a classical centralized allocation system is inefficient and impractical. All these reasons make decentralized [12,13] or distributed resource allocation decision making a viable and practical solution. This raises the need to explore and structure the decentralized decision process so that the outcomes of the combined individual effort of each of the distribution decisions will achieve or closely match the defined objective of the overall systems.

The advantages of distributed decision making have been researched extensively [14]. In this paper, we will present a distributed resource provisioning decision making system and a Cloud based scenario with Web applications installed in VM instances that are dynamically deployed on a Cloud setting via distributed decision making, and the same scenario but the provision and scaling decisions are made by a centralized decision maker. Using this Web application scenario, we show the operation and advantages of the distributed decision making system as compared with the centralized decision making process. In the rest of this paper, section 2 describes a typical Cloud based Web application usage scenario for dynamic resource allocation. Section 3 describes the classical centralized decision making provisioning and scaling system while section 4 introduces the distributed decision making provisioning and scaling system. In section 5, we compare the centralized and distributed provisioning and scaling systems. Section 6 briefly describes related work and section 7 concludes.

II. Web Application Scenario

We consider a typical Cloud-based commercial Web application scenario to provide quality of services with potentially unlimited number of users accessing the services at any time. Such services demand short response time, uninterrupted reliability and availability from the application. Thus, the Cloud service provider must provision resources to guarantee performance under all workload conditions or risks losing customers. Due to the unpredictable nature of workload patterns, the amount of resources to be provisioned is critical to the profitability of the Cloud service providers. Over-provision results in lower profit margin while under-provision certainly results in customer dissatisfaction. Obviously, the solution is to scale the resources dynamically based on workload demand.

The scenario system uses a front-end load balancer to dynamically route user requests to back-end VM based Web servers [15] that host the Web application. An Apache HTTP Load Balancer is used as a single point of entry for service requests and it routes requests to the underlying servers that host the target Web applications. The Web applications are deployed in Apache HTTP servers installed in Linux VMs. These VMs are initially provisioned and started on-demand by their provisioning and scaling system. The number of Web servers (VMs) will automatically scale up or down according to the number of current active sessions in each Web server instance in order to meet the service quality requirements.

The major scaling indicator [16] selected is the number of active sessions or logon sessions in each Web application. These indicators correlate closely with the capacity and performance of the web application according to the results of a performance and scalability study (reference?). Study shows that each Web application can support up to about 40,000 active, concurrent sessions. Above this threshold, the performance of the Web server system deteriorates rapidly and the system crashes imminently.

The mechanism for the actual resource provisioning of VM instances with web app and physical hosts is beyond the scope of this paper. However, for completeness, we will briefly describe the “Image-based provisioning” (IbP) technique we use in our system -- IbP is a deployment and activation mechanism that clones a “golden” virtual image to create new virtual machine instances. Automating the provisioning of new VMs with unique configuration from a “golden” image template [17] can be accomplished by a combination of template-based automation capabilities and external automation scripts.

III. Classical Centralized Decision Making Provisioning and Scaling System

Figure 1 shows a scalable architecture of a centralized provisioning and scaling management system (PSMS) in a Cloud Computing environment. The architecture design includes a front-end load-balancer (as described in Section 4), VMs with Web application, the collection of physical hosts, a centralized management system which includes a provisioning and scaling sub-system, and a service monitor sub-system embedded with a dynamic scaling algorithm.

The monitoring service of PSMS periodically retrieves performance and capacity data from each of monitoring data collecting agents provided by the application, the VMs and their physical hosts as scaling indicator metrics. The moving averages of these scaling indictors are used as inputs to a threshold based scaling algorithm which makes decision on adjusting resources on each of the active physical hosts to match current workload. Resource adjustment decision is executed by the provisioning service sub-system to initiate actions to scale up or down the current active set of physical resources.

The scaling algorithm is implemented as part of monitoring service of the PSMS. Figure 2 shows the algorithm in pseudo procedures; it is based on the scaling indicator $\mathbf{A}_i$ in each virtual machine instance in the Cloud. For simplicity, we choose a scaling indicator in our implementation that corresponds to the number of active sessions in the web application of each instance.

The algorithm uses statistically determined active sessions thresholds to make resource scaling decision. If all instances have active sessions above the upper threshold, a new VM with Web application instance will be provisioned by the Provisioning Service, started, and then added to the front-end load-balancer.

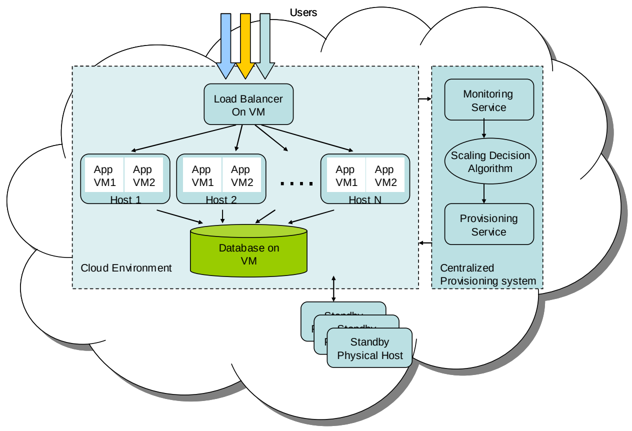

Figure 1. Centralized Provisioning and Scaling Management System (PSMS) in a Cloud Environment.

If a VM instance has active sessions below a given lower threshold and with at least one idle instance (no active session) in the active pool, the idle instance will be removed by the provisioning service from the load-balancer and be shutdown and removed from the system. Consequently, the load factors for all newly created and remaining active instances will be recalculated and applied to the load-balancer for workload re-distribution.

```
For an instance i in $N_{Instance}$
    If ($A_i/S_{Max}$ >= $T_{Upper}$) then
        Increment $N_{Exceed}$
    If ($A_i/S_{Max}$ < $T_{Lower}$) then
        Increment $N_{Below}$
    Record and sort all indexes J in ascending of $A_i/S_{Max}$
If ($N_{Exceed}$ == $N_{Instance}$) then
    Provision and start a new instance
    Increment number of instances: $N_{Instance}$
    Add new instance to Load-Balancer
If ($N_{Below}$ >= 2) then
    Set m equal first index in J
    If ($A_m$ == 0) then
        Remove instance m from Load-Balancer
        Shutdown instance m
        Decrement number of instances: $N_{Instance}$
        Decrement $N_{Below}$ and remove index m from J
        If ($N_{Below}$ >= 2) then
            Set n equal first index in J
            Remove instance n temporary
            Decrement number of instances: $N_{Instance}$
            Set normalized load factor $L_n = 0$
For an instance i in $N_{Instance}$
    Evaluate normalized load factor:
  $L_i = (1-A_i/S_{Max}) / \text{Sum}_{k=1,N_{instance}} [1-A_k/S_{Max}]$
 Apply new load factors $L_i$ to Load Balancer
```

where A_i: Number of active sessions in instance **i**

S_{Max}: Maximum sessions per instance (e.g. 40,000)

T_{upper} : Session upper-threshold (e.g. 80% or 0.8)

T_{Lower} : Session lower-threshold (e.g. 60% or 0.6)

$N_{Instance}$: Number of existing instances

N_{Exceed}: Number of instances exceeding session upper-threshold

N_{Below}: Number of instances below session lower-threshold

Figure 2. Dynamic Scaling Algorithm for Virtual Machine Instances in Cloud.

Once the scaling decision to scale up is made, the provision service will scan the current capacity and performance data from each of the physical hosts, and find the host with the most capacity for the additional VM deployment, and add new physical host if needed. Periodically, the Provision Service will perform maintenance service to consolidate VMs and remove idle physical hosts.

This architecture represents a classical centralized control design in which a single entity possesses all information (via communication with local agents), performs analytics, and makes decisions for the benefit of the entire system. As the number of VMs and physical resources increases and scatter across many geographic and political boundaries, it becomes more difficult to analyze, scale and manage. It efficiency decreases as it does not take into consideration the specific characteristics of each physical systems and VMs. Maintenance and trouble shooting become costly which adds to the operating costs of the service provider. An alternative approach of resource provisioning decision is necessary, which led us to explore the distributed decision making provisioning and scaling system.

IV. Distributed Decision Making Provisioning And Scaling System

Figure 3 shows a scalable architecture of a distributed provisioning and scaling management system (DPSMS) in virtualized Cloud Computing environment with the same scenario. The basic architecture design is similar to its counter part except that there is an Capacity and Utility Agent (CUA) deployed on each physical host under a dedicated service VM running continuously trying to maximize its own utility and PSMS is replaced by the Distributed Capacity Agent Manager (DCAMgr) (Figure 4) which is a light weight agent management system responsible for managing and communicating with the participating agents and directing the resource adjustment actions to the target systems. Below is a description the analytic and capacity agent and the agent management system.

Figure 4 shows the VMs with apps, the service VM with CUA within a physical host. The CUA is designed as an autonomic agent [18,19,20], a default CUA is initially installed in the service VM and boot-started with the Hypervisor as soon as the host system is active. Each agent includes a pluggable Capacity Index (CI) calculator which takes an input XML file describing the specific conditions and parameters unique to the physical host (default calculator and input xml are usually replaced by the user due to their own requirements and conditions).

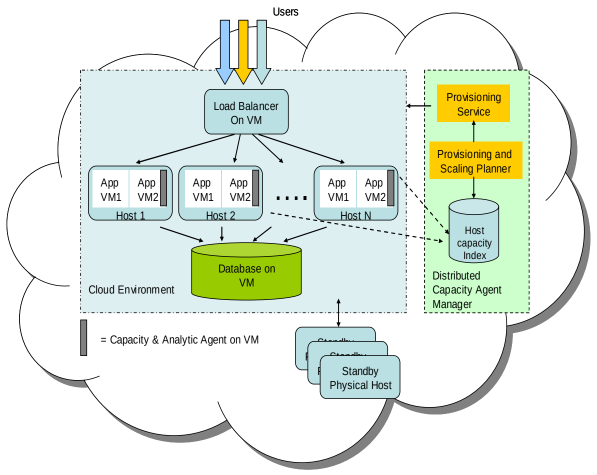

Figure 3. Distributed Provisioning and Scaling Decision Making System (DPSMS) in a Cloud Environment.

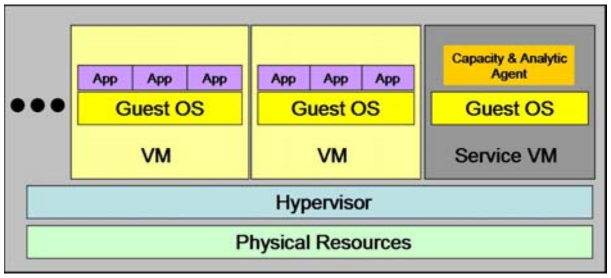

Figure 4. Capacity and Analytic Agent in the Service VM of a Physical Host

The CUA collects capacity and resource data from each of the VMs and the underlying physical host via the Hypervisor APIs and optionally from a system profile which describes the characteristics of the physical host such as power usage and its usage priority. Using the data, it generates a CI based on its own goals and incentives and it is through this CI that the host system controls its own destiny. The CI ranges from 0 to 1, with 1 indicating that the host's capacity is fully utilized and scaling up is needed urgently, and a value of 0 means the system is virtually idle. Figure 5 shows an example of a simple algorithm to determine the CI based on a few general systems performance indicators. In contrast to CPSMS which performs all analytics and make all resource decisions centrally, in the DPSMS, each physical host in the DPSMS indicates its current states of capacity and utility and its willingness to accept new VMs, or its need to scale up through the published CI.

The CUA analyzes the performance of the VM instances running on the local host (in this scenario, we follow the centralized system by using a single parameter "number of sessions" as Web app scaling indicator, and the same scaling algorithm (Figure 2)) and decides if resource scale up/down is needed. If resource scale up is needed and its CI is low, additional VMs will be created *within the local host*. If resource scale down is needed, VMs will be removed from the local host. All local VM deployment or removal actions will trigger re-calculation of the CI.

The CI is sent to the DCAMgr if changed. The DCAMgr maintains a database for all the current CIs and formulate a plan based on them to adjust resources to maximize resource utilization while meeting workload requirements.

CIm = % of current memory usage
CIc = % of current CPU utilization
CId = % of disk utilized
CIi = % of IO capacity

CI overall = max of (CIm, CIc, CId, CIi) *100

Figure 5. Sample Algorithm to deterimine Capacity Index

This algorithm can be replaced by the user's calculator (a Java class implementing the CUA interfaces) [21] and is loaded at runtime during start up to reflect the characteristics of the applications on each of the VMs. Figure 6 shows another example algorithm which takes into consideration of the priority of the underlying running applications.

CI app1 CPU = actual % of CPU used by app1
CI app2 CPU = actual % of CPU used by app2
CI app3 CPU= actual % of CPU used by app3
CI app4 CPU= actual % of CPU used by app4

Since Application 4 represents a higher-valued service, its importance is reflected with a factor >1.

CI overall = max of (CI1 + CI2 + CI3 + 1.5CI4)*100 <=1

Figure 6. Sample Algorithm to deterimine Capacity Index

Figure 7 shows an overview of the Distributed Capacity Agent Manager (DCAMgr), it provides basic services to the distributed CUAs such as discovery and registration and keeps a repository of the current CIs sent from the CUA of each of the physical hosts.

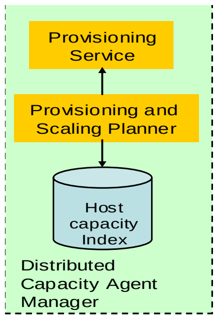
Provisioning Service
Provisioning and Scaling Planner
Host capacity Index
Distributed Capacity Agent Manager

Figure 7. Distributed Capacity Agent Manager Overview

DCAMgr includes 3 major services (1) The Host Capacity Index Repository service receives the capacity index from each of the CUAs under its watch and maintains a copy of them in its database. (2) Provisioning and Scaling Planner service reviews periodically all the CIs and formulate a plan to re-adjust resource allocation, including provisioning and/or removal of VMs as well as physical hosts form the active pool. The Planner includes a pluggable algorithm (a Java Class implementing the Planner interface) to determine the optimal resource allocation based on the indices from each of CAUs. Figure 8 shows an example threshold based allocation strategy which maximizes the CIs of each of the physical host with an acceptable margin. (3) The Provisioning Service carries out the allocation decisions made by the Planner, it includes functions to provision, remove and move VMs and physical hosts.

```
Sort indices in descending order  CI[n] // n=number of
physical hosts
Sort physical hosts in order according to their CIs: H(n)

For each ci in CI[n] ; n from 0 to n-1 {
  if ( i > upperlimit ) {
     if  (ci of CI(n-1) < 0.6 ) {
         createInstance in PH(n-1)
         remove PH(n-1) from the list
     }
     if (numberOfHostCIBelowLimit < 2) {
         addPhysicalHost();
     }
  }
}

Sort indices in descending order  CI[n] // n=number of
physical hosts
Sort physical hosts in order according to their Is:PH(n)

For each ci in CI[i] ; i from n-1 to 0 {
    if (numberOfHostCIBelowLimit > 2) {
         removePhysicalHost(i);
        remove PH(i) from PH(n);
     }
}
```

Figure 8. Sample Algorithm for provisioning

This algorithm seeks to maintain a target overall CI of close to 0.9 for all of its participants. Resource adjustment actions are initiated either by scaling or provisioning new VM instances or physical hosts if any of its CUA reports index close to 1. Similarly, instances and physical resources removal or consolidation actions are initiated if more than 1 of its CUA reports index below 0.1.

V. Comparison with Centralized Provisioning

As mentioned before, the success of an application and consequently its provider depends on the following factors: the application's scalability, availability, reliability, management and maintenance cost, resource utilization, and mostly importantly, its responsiveness. All these factors, especially in virtualized computing settings, to a large extend, depend on an efficient resource allocation system. Based on the initial implementation of the CPSMS and the proposed DPSMS architecture, the proposed distributed decision system offers the following advantages:

Cost reductions: Monolithic decision structure often fails to take advantage of the specific characteristics of its individual physical systems, which may result in over/under resource allocation, adding to the cost of operation.

Benefits to users: Since most physical resources in a virtualized computing environment are shared among multiple users, administrators will have more flexibility in optimizing their resource usages by manually adjusting its CI.

Improved decision-making: More accurate decisions can be made at the individual physical host level as data are readily available without network delay. System and geographic specific information can be incorporated into the decision making process to better reflect local system characteristics and requirements.

Improved scalability: Adding or removing resources can be done incrementally by a simple registration operation; minimum communication is needed between the agents and its agent manager. There is no need to re-compute the model due to resource changes.

Improved decision speed: Most of the decision making and data collection processes are executed in the distributed systems which are analogous to parallel processing. Naturally, for a relatively large Cloud operation, it is orders of magnitude faster than a centralized decision making system.

Improved maintenance: Maintenance is much easier to be done at a smaller isolated environment, especially in trouble shooting should problems arise.

Improved reliability: The crash of individual physical systems does not affect the overall operation. If a physical host crashes, it simply disappears from the active resource pool and the load balancer will re-route the workload to other instances and the system will adjust accordingly.

Distributed ownership: Resources used by a Cloud service provider may scatter across geographic and political boundaries; the distributed resource allocation decision model is consistent with the current geographic and political landscape of the world.

VI. Future works

Scalability has been one of the principal topics discussed regarding Cloud Computing model and is the most important issue for the Cloud service providers. Dynamic resource allocation is unquestionably the preferred solution to efficiently manage a Cloud setting. The coupling of autonomic computing principals [18,19,20] with distributed resource allocation decision making in a Cloud setting provides a new way of managing a Cloud, and raise the question if a hybrid system of mixed centralized and distributed decision making will be more applicable in the Cloud setting. All these are interesting areas for further exploration and its results will ultimately help the further acceptance of the Cloud service model.

VII. CONCLUSION

As we have learned in real life, the most efficient economic system is the one that provides individuals with incentives for their own decisions. It is also true for computing systems. In this paper, we have presented the overview architectures of both centralized and distributed decision making system for Cloud with a scenario. In addition to its major advantage of scalability, the main benefit of distributed decision making for resource allocation is that it enables Cloud service provider to distribute the resource allocation decision to its basic root entities where information and analysis can be done quickly and accurately, and collectively they optimize their computing resources in an efficient and natural way. The comparison of centralized and distributed systems have been studied extensively, as a follow up to this paper and specifically for Cloud setting, we will use simulated Web services data to study the efficiency of these systems with different workloads and sizes of Cloud.

REFERENCES

[1] G. Gruman, "What cloud computing really means", InfoWorld, Jan. 2009.

[2] R. Buyya, Y. S. Chee, and V. Srikumar, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities", Department of Computer Science and Software Engineering, University of Melbourne, Australia, July 2008, pp. 9.

[3] D. Chappell, "A Short Introduction to Cloud Platforms", David Chappell & Associates, August 2008.

[4] VMware ESX Server, VMware Inc., http://www.vmware.com/products/vi/esx/

[5] Xen Hypervisor, http://www.xen.org/

[6] E. Knorr, "Software as a service: The next big thing", InfoWorld, March 2006. P. Padala, K. G. Shin, X. Zhu, M. Uysal, Z. Wang,S. Singhal, A. Merchant, and K. Salem. Adaptive control ofvirtualized resources in utility computing environments. In *ACM SIGOPS/EuroSys European Conference onComputer Systems*, 2007.

[7] M.N.Bennani and D.A.Menasce. Resource allocation for automatic data centers using performance models. In IEEE International Conference on Autonomic Computing (ICAC), 2005.

[8] D.Kusic and N.Kandasamy. Risk-aware limited lookahead control for dynamic resource provisioning in enterprise computing systems. In IEEE International Conference on Autonomic Computing (ICAC0, 2006.

[9] Amazon elastic compute cloud (EC2). http://aws.amazon.com/ec2/.

[10] Centralized and decentralized computing: http://www.brainbell.com/tutorials/Networking/Centralized_And_Distributed_Computing.html

[11] A.C.Davison. "Statistical Models". Dept of Math, Swiss Institude of Technology. http://statwww.epfl.ch/davison/SM/SMsample.pdf.

[12] L. Kleinrock. "Distributed System".Communications of the ACM 28(11):1200-1213, November 1985.

[13] Emin Gun Sirer, Robert Grimm, Arthur J. Gregory, Brian N. Bershad (University of Washington), "Design and implementation of a distributed virtual machine for networked computers" Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP), Charleston, South Carolina, December, 1999: pp: 202-216

[14] J.H. Saltzer, D.P. Reed and D.D. Clark. "End-To-End Arguments in System Design" ACM Transactions on Computer Systems, 4(4):277-288, November 1984

[15] Websphere Application Server. http://www-01.ibm.com/software/webservers/appserv/was/

[16] David Parmenter. "Implemnenting and Using KPIs", John Wiley and Sons, 2009 - Business & Economics

[17] L. He, S. Smith, R. Willenborg, Q. Wang, "Automating deployment and activation of virtual images", IBM developerWorks, Aug. 2007. http://www.ibm.com/developerworks/websphere/techjournal/0708_he/0708_he.html

[18] Pual Horn, "Autonomic Computing: IBM's Perspective on The State of Information Technology", IBM Corp, http://www.research.ibm.com/autonomic/manifesto.

[19] J.O Kephart, D.M. Chess, "The Vision of Autonomic Computing", IEEE Computer Magazine, Jan 2003.

[20] J.O Kephart, W.E. Walsh, "An Artificial Intelligence Perspective on Autonomic Computing Policies", Policies for Distributed Systems and Networks, 2004.

[21] S. R. Safavin and D. Landgrebe. A survey of decision tree classifier methodology. IEEE Trans. on Systems, Man and Cybernetics, 21(3):660-674, 1991

[22] V. Ungureanu, B. Melamed, and M.Katehakis, "Effective Load Balancing for Cluster-Based Servers Employing Job Preemption", Performance Evaluation, 65(8), July 2008, pp. 606-622.

[23] L. Aversa and A. Bestavros. "Load Balancing a Cluster of Web Servers using Distributed Packet Rewriting", Proceedings of the 19th IEEE International Performance, Computing, and Communication Conference, Phoenix, AZ, Feb. 2000, pp. 24-29.

[24] V. Cardellini, M. Colajanni, P. S. Yu, "Dynamic Load Balancing on Web-Server Systems", IEEE Internet Computing, Vol. 33, May-June 1999, pp. 28 -39.

[25] Y. Ajiro and A. Tanaka. "Improving packing algorithms for server consolidation.". In Proceedings of the International Conference for the Computer Measurement Group (CMG).

[26] N. Bobroff, A. Kochut, K. Beaty. "Dyankic Placement of Virtual Machines for Managing SLA Violations". In Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management(IM), 2007

[27] M. N. Bennani and D. A. Menasce. "Resource allocation for autonomic data centers using analytic performance In *IEEE International Conference on Automatic Computing (ICAC)*, 2005.

[28] J. Rolia, L. Cherkasova, M. Arlit, and A. Andrzejak. "A capacity management service for resource pools". In International Workshop on Software and Performance 2005.