AN ADAPTIVE MANAGEMENT PROPOSAL FOR OPTIMIZING THE PERFORMANCE OF A VIRTUALIZED COMPUTING ENVIRONMENT

João Maria Azedo Rendeiro, Rui Neto Marinheiro, José André Moura, João Carlos Silva
rendeiro,joao@gmail.com; rui@marinheiro.org; jose.moura@iscte.pt; joao.silva@iscte.pt
ISCTE - Instituto Universitário de Lisboa (ISCTE-IUL), Instituto de Telecomunicações
Av. das Forcas Armadas, 1649-026 Lisboa, Portugal

ABSTRACT

The number of virtualized servers is overtaking, by a large amount, the number of physical servers. One of the drawbacks of this new scenario is a much more complex computing infrastructure to manage. In this way, the current paper proposes an adaptive management prototype that controls a virtualized environment. This prototype guarantees an adaptive and automatic solution that efficiently supervises and controls any virtualized environment, without almost any human intervention. In addition, it manages the relevant physical computing resources allocated to each virtual machine, like memory and processing power. The results from our prototype suggest that it is possible to balance memory among various machines and perform an effective control of each machine's workload, with a simple and low cost solution for our initial problem.[1]

Keywords- *Adaptive optimization, dynamic optimization, Autonomic computing, virtual machines, Virtualization.*

I. INTRODUCTION

With the globalization of the world economy and the increasing ecologic conscience, 97% of CIO[2]/CTO[3] is discussing green IT strategies and 45% had already implemented green IT strategies [1]. The increasing cost with power and cooling, as well as with management and administration [2], forced IT entities to push virtualization solutions and take advantage of its benefits.

The number of virtualized servers outgrew the physical servers by 36%[4] and by 2013 the forecast for the virtualized servers in the Western Europe will be 22.6% higher than the physical servers and will be over 2,500,000 units for virtualized servers against approximately 1,900,00 units for physical servers [1]. The same result was stated by a study [3], which indicates that the adoption of x86 server virtualization can decrease both CAPEX[5] and OPEX[6] due to the decreasing of maintenance contracts [4].

Due to the ease and speed at which virtual servers can be provisioned, copied, moved, and modified, this highly dynamic infrastructure can impose new challenges to IT Staff. Because a single host can hold multiple virtual servers, there is a risk to damage the infrastructure, when performing host maintenance and configuration activities, making it highly vulnerable to single point of failure problems. Virtualization simplifies many administration activities, but at the same time it has a disadvantage, since it adds a technological hurdle to be managed by highly specialized staff [5].

The use of Virtualization introduces new operational problems promptly identified by IT entities. A recent survey [3] identifies some of the most relevant problems: c*apacity management and planning*.

The result is a fast growing divergence on maintenance cost between physical server and virtual servers. Since 2006, the costs of the maintenance of physical servers have been increasing slowly and since 2008 have stabilized. Contrary to this, the maintenance cost of virtual servers has increased tremendously. Staff cost is also rising: it represents 31% of the annual budget for Portugal Datacenters; the cost of staff per server remains expensive for small and midsize business[7] and became more appellative for large business[8] [6].

The objective of the current paper is to create a model and subsequently a prototype that allows a more efficient, autonomic and adaptive control of virtual infrastructures, reducing the need of technical interventions. This means that the infrastructure should be able to monitor their surroundings, analyze them and, based in rules and policies, change the virtual server resources[9] in order to adapt them to the dynamic computing requirements.

The article is organized in the following manner: section II discusses related work; section III covers the description of the model and how the individual components interact among each other; section IV details the test conditions of the memory management test and the results; section V details the test conditions of the CPU management test and the results; and finally section VI presents the final conclusions and some future work.

II. RELATED WORK

Virtualization is a concept known for a long time and has been used in many area of expertise. One area of knowledge where the concept is widely used is in Information Technologies. The concept started to evolve in July 1959 when Christopher Strachey published the article "Time Sharing in Large Fast Computers", in New York at the UNESCO conference "International Conference on Information Processing" [7]. In fall of 1964, IBM started the "*CP-40 Project*", a research for a new Operating system, called CP-40, used on an IBM 360/40 mainframe. The system became operational in 1966. CP-40 was introduced in productive environments and had been

[1] This work was supported by the FCT projects:
PTDC/EEA-TEL/120666/2010 and PEst-OE/EEI/LA0008/2011
[2] Chief Information Officer
[3] Chief Technology Officer
[4] UK, Germany, France
[5] Capital Expenses
[6] Operational Expenses
[7] 25 to 99 Servers
[8] Above 500 servers
[9] Memory, Disk, Network and Power consumption

continuously improved till 1970, when was possible to bootstrap a CP-67 onto a System/370, which later, in august 2, 1972 was announced as VM/370 operating system, among other operating systems: DOS/VS[10], OS/VS1[11], OS/VS2[12] [8].

From this point forward, virtualization has been evolving and growing until 1999, when VMware Inc. created a product called VMware Workstation 1.0 for Linux and Windows, for Desktop Virtualization. Two years later VMware started to commercialize VMware Server, opening the market to Server virtualization on x86 platforms and mid-size computers [9].

CP-40 and its successors were what later started to be known as Hypervisor, also called VMM[13]. A hypervisor is a virtualization platform that runs multiple operating systems on a single physical computer called the host [10] and is classified in two types: Type1 which runs directly on the hardware; and Type 2 which runs on top of a typical operating system.

Hypervisor from type 1 is also called Bare Metal, because it runs on top of the host system's physical hardware and the guest operating systems run on top of the Hypervisor. In this category we can find products like Microsoft Hyper-V, Citrix XenServer, VMware ESX Server, Xen and Linux KVM [11]. Hypervisor from type 2 is also called Hosted, because it runs on top of a conventional Operating System. In this category we can find products similar to Microsoft Virtual Server, Microsoft Virtual PC, VMware Server and HXEN[14], which can also run as a type 2 hypervisor [12,7,9]. The products that rely on type 2 hypervisor have in general worst performances than the products that use type 1 hypervisor.

To be called hypervisor, the Virtual Machine Monitor must provide at least three properties to the generic programs under its control: efficiency, resource control, and equivalence. Efficiency means that all harmless instructions are executed by the hardware directly, with no intervention from the control program. Resource control means that it must be impossible for an arbitrary program to affect system resources. Equivalence means that any program running inside a virtual machine is equivalent to running the same program outside the virtual machine [13].

The most common approaches to virtualization are: full virtualization, para-virtualization and hardware-assisted virtualization. With full virtualization the hypervisor can simulate the server physical hardware, thus allowing the virtual machines to run unmodified guests with flexibility and efficiency. Full virtualization is frequently implemented by the combination of binary translation and direct execution.

Para-Virtualization or OS assisted virtualization uses a completely different approach from full virtualization. Instead of running an unmodified operating system with the burden of problem for solving some issues associated to the x86 architecture, e.g. Ring Compression, Ring Aliasing and Non-Privileged Sensitive Instructions [14]; alternatively, the Para-Virtualization changes the guest operating system by modifying the privileged instructions with hyper-calls to communicate directly with the hypervisor.

[10] Virtual storage version of DOS
[11] Virtual storage version of MFT
[12] Virtual storage version of MVT
[13] Virtual Machine Monitor
[14] Hosted Xen Hypervisor

Hardware-assisted virtualization is the hardware vendors' contribution to enhance virtualization. Some hardware extensions have been proposed to simplify virtualization. Intel and AMD implemented similar technologies named respectively Intel-VT and AMD-V. Both added a new operating mode to the processor called guest mode which runs in ring 0 and keeping the already existent host mode but shifting it to a new ring below ring 0, called ring -1. IN this way, when the guest OS performs a privileged operation, it automatically traps the hypervisor and the guest state is stored in Virtual Machine Control Structures[15] or Virtual Machine Control Blocks[16] [9].

III. THE MODEL

When provisioning virtual machines, IT technicians usually configure them with pre-established resources[17], taking into consideration the function of the guest. Because of the administrative burden of configuring the virtual machine every time it is needed, it is usual to change the configuration only when the function of the VM changes or when additional functions are added. This situation is, at best a potential waste in resources mainly because IT technicians tend to overestimate the needed resources. Efficiency would improve substantially if the resources provided to virtual machines could adapt according to the business needs and the user demands.

The hybrid model presented aims the automation of resource allocation and is based on the dual control theory. The the proposed model has implementation both on host, showed on Figure I and on the guest showed on Figure II.

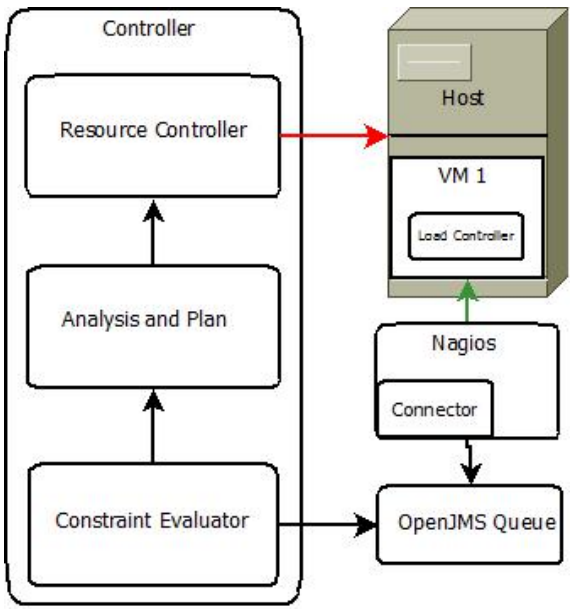

Figure I – Model, Host Control

[15] Used by VT-x
[16] Used by AMD-V
[17] CPU, Memory, Disk, Network

On the host, the model is composed of a monitor and a controller and uses a Multiplicative-Increase / Multiplicative-Decrease – MIMD variant algorithm to control the memory.

A MIMD based algorithm was chosen to control the memory, since it's a combination of an exponential growth when there's no resource starvation with an exponential reduction when starvation takes place, enabling a fast and efficient usage of the resource. Due to the way Xen manages the memory, the multiplicative factor of the MIMD is applied not to the total memory but to the used memory, hence explaining the variant approach.

On the host, as showed on Figure I, the monitor is implemented with Nagios and is responsible for gathering performance information from the host, guest virtual machine and other environment parameters[18] and deposits this information on the OpenJMS Queue.

On the host controller, the constraint evaluator feeds on the OpenJMS Queue and is responsible to decide if the host system and the virtual machine are performing efficiently, based on pre-define rules, and if not triggers the necessary actions. The Analysis and Plan component is responsible for the analysis and planning of the adequate change needed, and pass this information to the resource controller to act. The resource controller is responsible to translate the changes needed into actions onto the virtual machine, thru the use of the multiplicative factor strategy of MIMD applied to the memory.

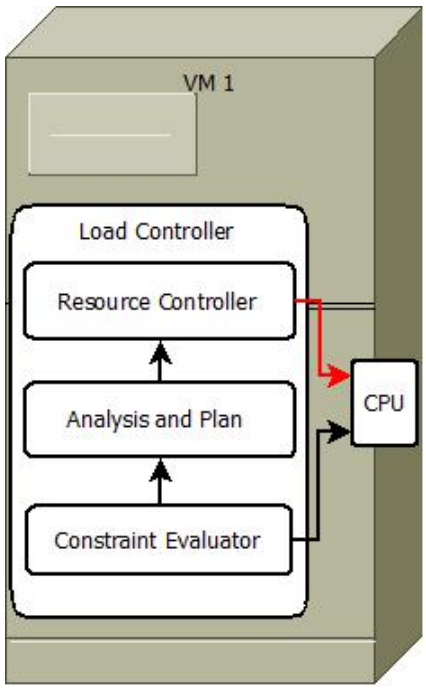

Figure II – Model, Guest Control

As shown in Figure II, in each guest there is a different controller and uses an Additive-Increase / Multiplicative-Decrease – AIMD algorithm to control the CPU.

The AIMD algorithm was chosen to control the CPU, since its allocation is done in a distributed manner. Here, the combination of linear growth when there's no resource starvation, with an exponential reduction when starvation takes place, enables a fairer usage of the resource.

On the virtual machines has showed on Figure II, the load controller monitors the workload and adjusts the CPU time allocated to processes, thru the use of an AIMD algorithm.

A. Memory Managment

Regarding memory management, Nagios Server collects data every minute from the two virtual machine's Linux A and Linux B and feeds the OpenJMS queue. The constraint evaluator selects the relevant data from the queue and passes it to the next stage to analyze and plan the actions needed. When the decided action is defined, it is then passed to the resource controller to be carried out.

The constraint evaluator listens to the queue and when the used memory is above 85%, the controller borrows memory from the virtual machine Linux B and grants the same amount to Linux A. If the used memory is between 80% and 85% the multiplicative factor of MIMD (1.33) is used to compute the new total memory of Linux A. If the used memory is below 70% then the multiplicative factor of MIMD (1.33) is used again to compute the new total memory of Linux A, but in this case the previously borrowed memory is returned to Linux B.

B. CPU Managment

Regarding CPU management, the controller checks the load of the virtual machine every 30 seconds analyzes and plan the actions needed. When the decided action is defined, it is then passed to the resource controller to be carried out.

The constraint evaluator collects the load of the virtual machine and if it is greater than 1[19], selects the processes that are at the top 3 of CPU usage and decreases them, changing the CPU usage to 5%. Once the load of the virtual machine goes below 1, a 5% of CPU usage is added to the previously limited processes, in both cases, the changes are made using a tool called Cpulimit[20]. The Linux Load is a measure of work of the CPU and in this case because the host is like a single-core CPU, a value of 1 means that the CPU is at capacity.

IV. Memory Management

In order to verify the ability of the model in dealing with more than one virtual machine and managing the memory of virtual machines simultaneously, a test has been made where two virtual machine were used, Linux A and Linux B, respectively with 256 Mb and 512 Mb of RAM. Also included was the OpenJMS Server with one queue, the stress program and the controller program to manage the memory.

For the program we considered two upper limits of 80% and 85% to avoid machine unavailability, for the same reason a range between 256 Mb and 1024 Mb of RAM is imposed on every virtual machine. Because memory is a valuable resource and must be managed

[18] Network activity, Disk space, Power consumption

[19] http://www.linuxjournal.com/article/9001

[20] http://cpulimit.sourceforge.net/

efficiently, we considered a 70% lower memory limit, but any other value would also be adequate, although less efficient.

To generate the desired memory workload, the stress program is parameterized with the following: "stress –m x –vm-bytes 16M"[21]. Where x is a number representing the memory multiplication factor and 16 is the amount of memory to be factored. The values used are: 2, 3, and 5, approximately at minutes 04:28, 09:18 and 11:50; these values were chosen in order not to exceed the available memory and still surpass the pre-defined limits.

Figure III and Figure IV show the memory usage of respectively Linux A and Linux B, which are changing according to the memory load of Linux A.

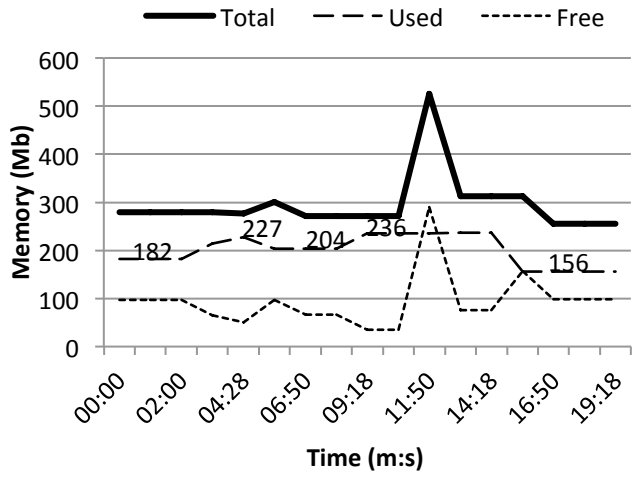

Figure III – Linux A memory usage

It is possible to see that the model was able to collect the memory parameters from Linux A, analyze it against the defined criteria, and change the memory using the scale factor (1.33), maintaining the used memory between 70% and 80% additionally when the 85% upper limit is exceeded. The model is able to borrow memory from Linux B, use it on Linux A and give it back when the 70% lower limit is reached.

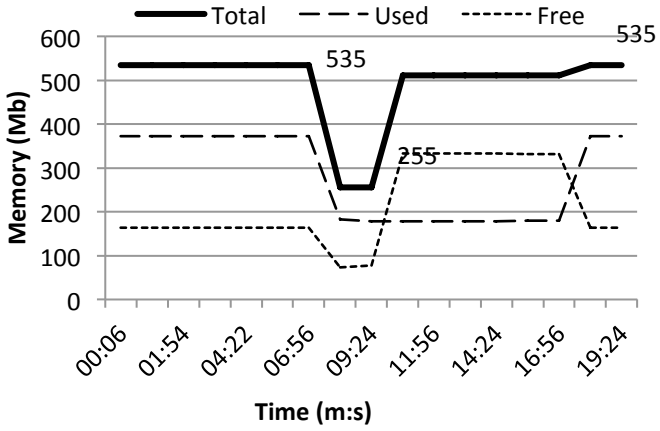

Figure IV – Linux B memory usage

[21] For the manual please check: http://www.stresslinux.org/sl/wiki/Software

V. CPU MANAGEMENT

The load controller is responsible for the management of the workload within the virtual machine. To verify that the load controller can effectively manage the virtual CPU of the virtual machine, a test has been performed with a virtual machine (Linux A), with 512 Mb of RAM, the load controller, the stress program and a program called cpulimit[22] to manage the CPU usage of the processes.

The CPU usage limitation is performed by cpulimit using the following parameters: "cpulimit –p pid –l x –z", where pid and x, are the process id and the CPU usage to be applied to the process, expressed by a percentage. The workload used is generated with the stress program, with the following parameters: "stress –c x", where x is the CPU value index.

Figure V represents the load of the virtual machine. This means the work of the CPU and, in this case, a value of 1 (100%) means that the CPU is at capacity[23], i.e., the CPU supports the system workload[24].

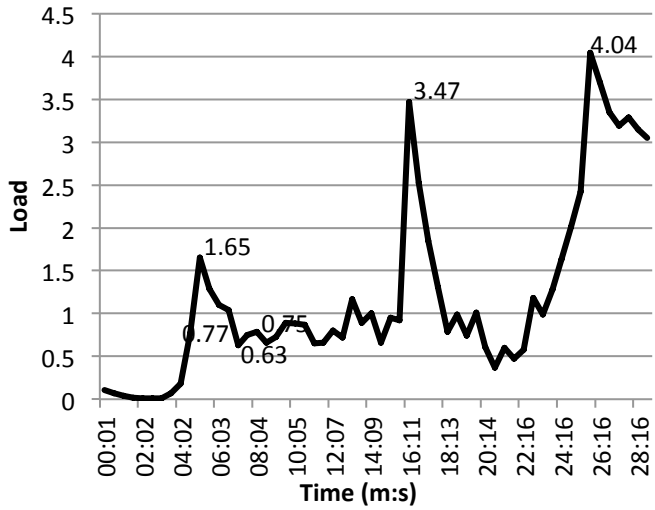

Figure V – Linux A, CPU Load

When the program is started there is no load generated. At the minute 4:32, the load was generated and the program started to control the workload. After minute 23:16 the load generation was stopped.

In Table I, we can see that the mean is 0.99 and that the minimum and maximum values are respectively 0.37 and 3.47. Although the average load lies below 1, there is a load peak at 16:41 in Figure V.

[22] http://cpulimit.sourceforge.net/

[23] http://blog.scoutapp.com/articles/2009/07/31/understanding-load-averages

[24] Above 1, the system slows down

Table I - Data Load Analysis

Load	
Mean	0,9989474
Standard Error	0,0929199
Median	0,875
Mode	0,66
Standard Deviation	0,5727968
Sample Variance	0,3280962
Range	3,1
Minimum	0,37
Maximum	3,47
Sum	37,96
Count	38
Confidence Level (95%)	0,1882736

Based on the last results, the controller can manage the CPU load of the virtual machine during the most part of the time. The average load is 0.99 with a minimum of 0.37 and a maximum of 3.1.

VI. Conclusion

With the increased complexity of virtual/ physical systems, it's imperative to create autonomic and adaptive systems to prevent the increase on administrative IT work. Rather than focusing only on a Host (Hypervisor) or client (virtual client) approach, the presented model, focuses on both ends simultaneously. In this way, the results of the current work suggest that it is possible to have a hybrid, i.e. centralized & distributed solution to manage a virtualized computing environment.

In some specific cases, the prototype controls the computing resources in a sub-efficient way. These situations must be addressed in future work, alongside a study of the scalability of the proposed solution.

References

[1] Symantec. (2009) www.symantec.com. [Online]. http://www.symantec.com/content/en/us/about/media/Green_Data_Center_Survey_Press_Presentation.pdf

[2] Stelios Charalambakis, "Virtualization Adoption & The Next Generation of Virtualization," Greece, Cyprus & Malta, 2009.

[3] Forrester Consulting. (2010) www.ca.com. [Online]. http://www.ca.com/files/IndustryAnalystReports/virtual_mgmt_trends_jan2010_227748.pdf

[4] Bob Laliberte. (2009) bladenetwork.net. [Online]. http://bladenetwork.net/userfiles/file/ESG-Brief-BLADE-VMready-Mar-09_FINAL.pdf

[5] IT Process Institute. www.ca.com. [Online]. http://www.ca.com/files/industryresearch/itpi-virtualization_213801.pdf

[6] Timóteo Figueiró, "Infra-estruturas Convergentes: Como a Integraçãoo Tecnológica pode Reduzir Custos e Aumentar a Capacidade de Resposta do Negócio?," Lisbon, 2009.

[7] Melinda Varian. (1997) www.princeton.edu. [Online]. http://web.me.com/melinda.varian/Site/Melinda_Varians_Home_Page_files/neuvm.pdf

[8] R. J. Creasy, "The Origin of the VM/370 Time-sharing System," vol. 25, no. 5, 1981, In: IBM Journal of Research and Development.

[9] VMware. (2006) www.vmware.com. [Online]. http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf

[10] Mitch Tulloch. (2010) download.microsoft.com. [Online]. http://download.microsoft.com/download/5/B/4/5B46A838-67BB-4F7C-92CB-EABCA285DFDD/693821ebook.pdf

[11] Xen.org. What is Xen Hypervisor. [Online]. http://www.xen.org/files/Marketing/WhatisXen.pdf

[12] Xen.org. www.xen.org. [Online]. http://www.xen.org/products/projects.html

[13] Gerald J. Popek, "Formal Requirements for Virtualizable Third Generation Architectures," *Communications of the ACM*, vol. 17, no. 7, pp. 412 - 421, July 1974.

[14] John Fisher-Ogden. (2011, January) cseweb.ucsd.edu. [Online]. http://cseweb.ucsd.edu/~jfisherogden/hardwareVirt.pdf

[15] H. Unbehauen, "Adaptive dual control systems: a survey," , Lake Louise, Alta. , Canada, 2000.