Power Management in Grid Computing with Xen

Fabien Hermenier, Nicolas Loriant, Jean-Marc Menaud

OBASCO project
École des Mines de Nantes - INRIA, LINA
4, rue Alfred Kastler
44307 Nantes Cedex 3, France
{fabien.hermenier,nicolas.loriant,jean-marc.menaud}@emn.fr

Abstract. While chip vendors still stick to Moore's law, and the performance per dollar keeps going up, the performance per watt has been stagnant for last few years. Moreover energy prices continue to rise worldwide. This poses a major challenge to organisations running grids, indeed such architectures require large cooling systems. Indeed the one-year cost of a cooling system and of the power consumption may outfit the grid initial investment.
We observe, however, that a grid does not constantly run at peak performance. In this paper, we propose a workload concentration strategy to reduce grid power consumption. Using the Xen virtual machine migration technology, our power management policy can dispatch transparently and dynamically any applications of the grid. Our policy concentrates the workload to shutdown nodes that are unused with a negligible impact on performance. We show through evaluations that this policy decreases the overall power consumption of the grid significantly.

1 Introduction

The number of applications requiring gigantic storage and computational capabilities has increased the interest in grid computing in the last few years. At the same time, chip vendors have been producing more and more powerful CPUs at lower prices. Past work has mostly focused on designing efficient and scalable grids. Nevertheless, energy consumption has recently become a major concern for organisations running grids. Indeed, power consumption is an important portion of the total ownership cost, it determines the size of the cooling system and of the electrical backup power generators. The cost of a cooling system with the power consumption may sometimes exceed the grid initial investment. Moreover, intensive power consumption increases the chance of component failure. Thus, power consumption must be a key feature in grid design.

In the meantime and despite numerous users submitting jobs, a grid must rarely maintain peak performances constantly. For example, the average load of the grid of the École des Mines de Nantes subatomic research lab is about 70%. Moreover, many applications have specific needs *e.g.* middleware or operating system. For example, `Aliroot`(the application of the subatomic department of the École des Mines of Nantes), requires a specific middleware (Allien) and a

particular Linux distribution (Scientific Linux Cern). All those problems argues for a generic power management system adapted to grid.

In this paper, we present our prototype on power management for grid architectures. Our prototype is based on Xen virtual machine hypervisor so that our solution is transparent to user. Based on workload probes (CPU load, memory, NICs throughput *etc.*), the policy migrates virtual machines to concentrate the workload on fewer nodes of the grid, thus allowing unused nodes to be shut down. We present performance results over different application workloads and show the effictiveness of our algorithm.

The rest of the paper is organised as follows. Section 2 presents existing power management policies for cluster computing. Section 3 reviews the virtual machine technology and then Section 4 describes our implementation. Section 5 shows performance results and Section 6 concludes.

2 Energy management

Most existing work on power management has focused on CPUs and laptops. Indeed, battery duration is a critical problem for users and CPU is by far the most power consuming component of a computer[1]. Today, most processors feature DVFS (Dynamic Voltage and Frequency Scaling) even server designed processor. A DVFS processor has a set of levels (voltage and frequency pairs). A default policy integrated in the BIOS can be overloaded by the operating system. OS and system select the level according to performance and temperature constraints.

The rest of this section describes IVS, CVS and VOVO, three power management policies for single system or clusters and the problem of computation migration in grid computing.

2.1 IVS – Independent Voltage Scaling

Independent Voltage Scaling is a node-local policy for power management. In this approach, every node must be a DVFS processor. Each node adjuts its processor voltage and frequency to the lowest possible value such that there is not much impact on performance. IVS is very simple and it does not require any specific environment or information about running applications. Elnozahy *et al.* [1] report power savings of up to 30% depending on applications. In order to minimize the impact on performance, Chen *et al.* [2] applied IVS only to nodes not in the critical path of running applications.

2.2 CVS – Coordinated Voltage Scaling

In contrast to IVS, Coordinated Voltage Scaling is a global policy. Each node periodically informs a monitor of its workload. The monitor evaluates the overall

[1] Although, network interfaces should not be neglected, especially in grids where low-latency NICs and Gigabit cards are common.

workload, estimates an average load and broadcasts it to every nodes. Then the nodes adapt their voltage and frequency to this average. This strategy tends to decrease the variance of the frequency distribution across the grid.

CVS is effective when the workload is uniformly distributed among nodes. Elnozahy *et al.* [1] evaluate the power savings of CVS to be about 25%. Nevertheless, the effectiveness of CVS is questionable when workload distribution is not uniform, indeed, nodes with an bigger workload are slowed down, thus decreasing the overall performance.

2.3 VOVO – Vary On, Vary off

Pinheiro *et al.* [3] and Chase *et al.* [4] have proposed Vary On Vary Off, which dynamically adapts the number of nodes available according to the grid workload.

VOVO applies a workload concentration strategy. Similarly to CVS, a monitor estimates the overall workload, from which VOVO deduces the number of nodes required. When too many nodes are in use, jobs are migrated or balanced to concentrate the workload. Then unused nodes are simply shut down. The monitor may define a performance tolerance to possibly overload some nodes to further increase power savings.

VOVO has been found to give a power savings of 31% to 86% [1], indeed VOVO is fully effective when a large cluster is almost unused. However VOVO requires job migration technology, that can be implemented at several level. At the application level, users have to develop their own load balancer. While at the operating system level, a specific OS has to be installed on every machines and these must be identical. While this is rarely an issue in clusters, grids are more heterogeneous.

Designing the migration mechanism at the application[5], middleware, or operating system[6] levels is hardly feasible, as energy saving is an issue for organizations running grids, not for users developing applications. In our context, we suppose that grid are built by aggregations of primary cluster. Each cluster is managed by different organizations with their specific applications. Thus, various frameworks and middleware are used to develop grid applications. It is unfeasible to impose one of them on the entire grid as these are specific to the kind of applications developed (computationnal, data, services) [7].

3 Virtual Machines

The goal of a virtual Virtual machines is to provide each of multiple users with the illusion of having an entire computer (a virtual domain), all on a single physical computer (the host). This technology, also known as virtualization, differentiates from application virtual machines such as Sun Microsystem's Java Virtual Machine that isolates Java code from the underlying hardware and operating system thus releasing the developer with porting issues.

In a virtualized environment (see Figure 1), a piece of software called the hypervisor or the virtual machine monitor must enforce isolation between multiple virtual domains and must partition resources (CPUs, RAM, HDDs, NICs, *etc*) among them. The rest of this section describes virtualization and paravirtualization, the two major ways to implement an hypervisor.

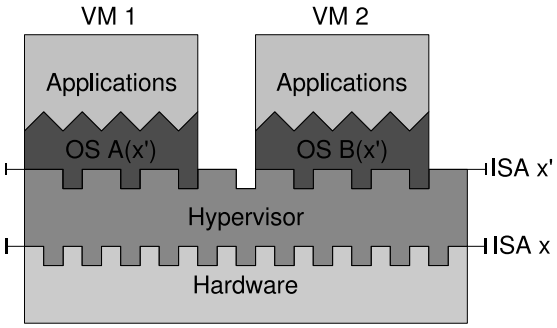


Fig. 1. Architecture of a Xen virtualised System

In contrast to emulation, virtualization reproduces the host architecture identically for the virtual machines. Hypervisors such as VMWare [8] run most instructions of the virtual domains directly on the host processor. The hypervisor must trap sensitive instructions that can not be executed safely such as TLB operations. While virtualization restricts the virtual architecture to the host one, it stills allows running unmodified OS images with much better performance than emulation.

Paravirtualization [9, 10] has been introduced to overcomes the performance loses of virtualization due to sensitive operations. It explicitly requires porting the OS to a particular hypervisor. The architecture presented to the virtual machines slightly differs from the host architecture in order to eliminate "cooperatively" sensitive operations.

Virtualization thus makes it possible hosting a lot of virtual domains on the same machine without constraints. Each domain could use its own operating system, middleware or applications without risk of conflicts with others system. This advantages make that virtualization appreciable in grid computing: Cohabitation with the local organisation is safer and a migration mechanism of virtual domain makes deployment easier[11].

4 Our solution

Our solution is a dynamic placement system for virtual domains on a grid virtualized with Xen (an hypervisor for each node). The placement algorithm aims to save energy by concentrating virtual domains on the smallest subset of nodes possible. Nodes not hosting virtual domains can be stopped, in order to reduce the overall power consumption. At present, the distribution criterion for virtual domains placement is based on CPU usage but could be extended to take other criteria into account, *e.g.* network traffic or memory usage. Distribution of virtual domain is made with a minimal service interruption with xen migration mechanism[12]. The problem of data migration is solved by using a files server that exports operating systems and datas for each virtual domain.

Our solution is transparent for users of virtual domains and does not require any adaptations or specific operating system[2]. The architecture is based on a client/server model. A standard pre-made domain is running on each node. That domain, called Domain 0, runs a tool, the harvesting agent, that monitors CPU, memory and network. This supervisor monitors the resource usage of the node, and how this resources are divided among the node's virtual domains.

Each harvesting agent periodically sends a report to the decision agent which analyses them to get an overview of the grid and of its virtual domain resource usage. Then an algorithm determines actions to be taken by the various nodes (virtual domain migration, boot, shutdown).

4.1 Resource collection

The harvesting agent uses functionalities offered by the `xenstat` library provides with Xen. This library allows the domain-0 to obtain statistics generated by the hypervisor, for each virtual domain, including information concerning allocated memory, bytes sent and/or received by the virtual network interface or the quantity of time used by the virtual CPU. By retrieving this information periodically, it is possible to know the percentage of CPU consumed by each virtual domains and, by summing them, the global CPU usage of the node (in this case, the result obtained does not consider the load of the hypervisor.) A report is made and sent to the decision agent. The harvesting agent is responsible for notifying the decision agent of changes in a node's state: When a node is stopped or ready to accept virtual domains, its harvesting agent send a departure or arrival notification, respectively. Thus, the decision agent knows the placement of virtual domains among the grid, and what nodes are available or offline.

4.2 The decision agent

This agent runs on its own machine. According to reports and notifications packets sent by nodes, the agent constructs two representations of the grid. A

[2] Actually it requires to use a Xenified OS, but that limitation disappears with the help of hardware virtualization

node based view gives information relative to the grid components: IP, MAC address, and state (online or offline) of each nodes. A virtual domain based view gives information about their location, their current resource consumption and the level of saturation of their host. The decision agent then uses this information to concentrate virtual domains in the smallest possible subset of nodes while avoiding saturation. The algorithm is based on two variables: a saturation and an underload threshold. Decisions are taken by comparing the current load of each node with its thresholds.

In order to concentrate virtual domains without overloading nodes, the algorithm selects virtual domains that consumes the highest amount of CPU on the lowest loaded node then move it on a more loaded node. As we suppose an homogeneous environment, CPU consumption of that domain should remain almost identical on the new node. So, the destination node is chosen such that the sum of the current destination node load and the current domain load is under the overload threshold. The first acceptable node is picked. This operation is repeated in the next iteration if the node is still underloaded and if a more appropriate place is available for virtual domains.

Virtual domains concentration could be dangerous in a certain way, as virtual domains does not continually do the same job and thus its CPU consumption may vary. Then, it is possible that if some domains increase their CPU consumption, the nodes that host them will be overloaded. To limit the performance degradation, we reduce the load in the following way: the domain having the lowest load will be moved to a node that can host it. The destination node should have the maximum CPU usage without being overloaded (before and after the migration). If no node is available, a new node in the "offline nodes set" will be initialised and domain will be migrated on.

When a decision is made by the agent, it will send the corresponding action to the appropriate node. Actions are shell commands, that will be executed in Domain 0. As we want to migrate virtual domains to save power by a VOVO system, three kinds of actions can be sent to a node. The most used action is to migrate a specific virtual machine to a new node. Others actions, specific to the VOVO aspect of our solution cause a node to be halted or booted.

5 Evaluation

The evaluation of our solution aims to show that dynamic placement of virtual domains allows a good energy saving with a tolerable performance degradation. We compare energy economy and performance degradation of our solution and a traditional IVS solution.

In order to evaluate our solution in a predictable environment, we developed a benchmark that sends "load request" to clients. This tool was tuned to make each client consumes a certain amount of CPU time. For each experiment we ran the tests three times. First, in a native environment, each application runs on a separate node, without any power management policy. Second, a standard IVS policy is used on each node. Third, we ran the benchmark with our prototype.

In this case, the applications ran on virtual domains that were dynamically migrated between a variable amount of actives nodes, depending on the CPU resource needs).

Our grid is composed by four Sun V20Z stations. Each machine contains 4GB of RAM, 2 Opteron 250 and run a GNU/Linux distribution with Xen 3.0-testing (based on a 2.6.16 kernel). All applications are compiled in 32bits mode. A file server contains all applications and OSs used by nodes in order to make administration easier. All machines (4 nodes and 1 file server) are connected throught a Gigabit network. The IVS software used for the second test is based on `powernowd` and the nodes' overload and underload threshold chosen to migrate virtual machines were respectively 80% and 70%. In all situations, we measure the effective CPU consumption of each node, distribution of CPU time between domains and Watts consumed by the grid (not including the file server).

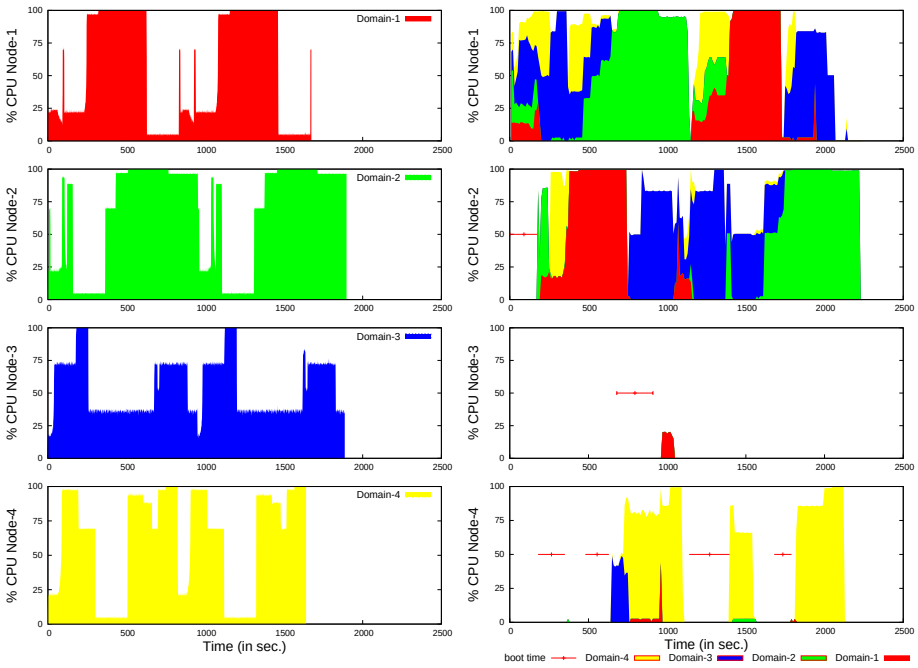
Our benchmark evaluates the dynamic behaviour (migration, turning nodes on and off) of our strategy and its benefits in comparison to IVS. We adapt it to create four different scenarios (one per benchmark client). Each scenario creates high-load peaks or low-usage periods at different times. This evaluation simulates a grid supporting a variety of different services. Figures 2(a) and 2(b), and Table 1 show the nodes' CPU utilization and the domain location over time. First, we observe that the IVS strategy has CPU average load that is almost equal for all four nodes (approximately 60%), whereas our solution tends to concentrate all the work on three nodes. Second, the CPU average load for each node is less variable in time for our solution due to the policy's decisions.

Table 1. CPU average utilization with dynamic evaluation

IVS	$node1 = 51.16\%(\delta = 37.73)$ $node2 = 65.42\%(\delta = 36.23)$ $node3 = 68.11\%(\delta = 22.13)$ $node4 = 59.59\%(\delta = 33.39)$	Virtualization	$node1 = 82.69\%(\delta = 20.35)$ $node2 = 84.08\%(\delta = 6.91)$ $node3 = 09.52\%(\delta = 9.58)$ $node4 = 65.68\%(\delta = 28.3)$

The four different scenarios are built to cover different situations that illustrate advantages and drawbacks of domain distributions. A positive effect of concentration appears around 1100s (Figure 5), where two of the four nodes are disactivated by migrating virtual Domains 1 and 4 to Node 2. These migrations are possible because the two domains consumes almost no resources at that moment. An overview of a virtual domain repartition appears around 1600s. On Node 2 from which virtual Domain 3 was migrated to Node 1, in order to leave CPU time for virtual Domain 2. Thus, domains that need an important amount of resources have a higher priority than small domains. This exclusion of "small" domains avoid performance loss.

A side effects of migration can be seen on Node 4, at about 400s. To free some resource on Node 1, Node 4 was booted, but during the boot time, the load decreased, and finally Node 4 received a virtual domain only for 30 seconds and consumed 3% CPU time before being shut down and turned on again two



(a) Native Execution with IVS (b) Virtual Environment with migrations

Fig. 2. Global CPU consumption

minutes later. As it is impossible to precisely predict the future needs of domains, it is hard to avoid such "useless" boots. Nevertheless, it would be possible to define a minimum up time to avoid node rebooting constantly.

Table 2. Execution time and average energy consumption

Environment	Execution time		Energy	
	Cumulated	Δ with Native	Consumption	Gain[3]
Native	118:39	-	651.12 W	-
IVS	119:01	+0.3%	636.57 W	1.93%
Virtual	139:37	+17.67%	498.13 W	9.97%

We can see in Table 2 and Figure 5, that the energy saving of our solution is appreciable but there is a small performance degradation (the cumulated time of execution increases by 20 minutes). This is due to the time needed by a node to boot (approximately 3 minutes). To evaluate the time lost due to reboot, we re-ran the experiment with virtual environment without shutting down or

[3] Considering time increase

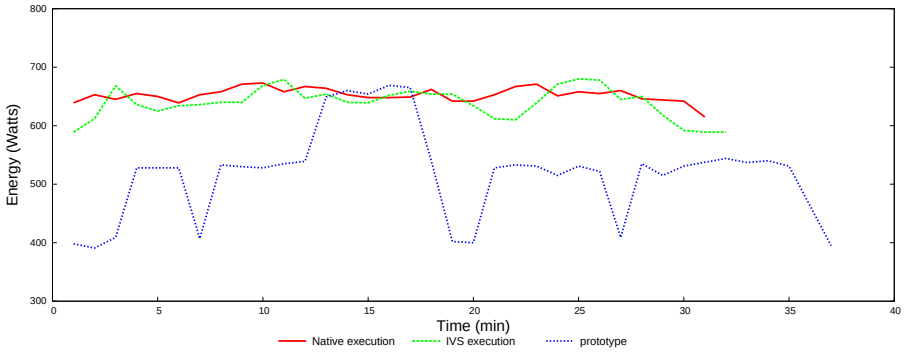


Fig. 3. Global energy consumption

booting nodes. Migration is still effective but boot time is reduced to zero. In this situation, the performance decreases only by 8.53% (with a cumulated execution time increased by 10 minutes).

This evaluation shows that our solution allows important energy savings, especially in a stable environment, but in a very dynamic situation, the boot time of nodes slightly reduces the benefit of our solution. About 50% of the overhead of our solution is due to boot time. Software solution such as hibernate or suspend-to-ram may greatly minimise that issue.

6 Conclusion and future work

In this paper we have show that the Xen live migration allows to adapt the VOVO power saving strategy to grid computing considering its complex infrastructure. Contrary to traditionnal VOVO systems, the abstraction layer offered by Xen provides a generic computation migration process based on virtual domains placement. The computation concentration is done by dynamically migrating virtual domains, considering their resource needs. This concentration of virtual domains on the minimal subset of nodes allows to shut down unused nodes, thus to save energy. By separating the migration and the placement concern at the hypervisor level, our solution is completly transparent to developers and users. Evaluations show our power managment policy based solely on CPU load may significaly decrease power consumption in grid with variously loaded machine.

As future work, we intend to extend our power management policy to a multi-criterion system, especially to include network traffic information. Indeed, the network hardware of a grid also consumes a non neglectible amount of electrical power. Our policy could tighten or regroup on a single node, highly communicating virtual domains in order to reduce network traffic. Utilization of software suspend will allow to reduce performance loss due to node boot time and increase the grid reactivity.

References

1. Elnozahy, E.N., Kistler, M., Rajamony, R.: Energy-efficient server clusters. In: Proceeding of the second Workshop on Power Aware Computing Systems, Cambridge, MA, USA (2002) 179–196
2. Chen, G., Malkowski, K., Kandemir, M., Raghavan, P.: Reducing power with performance constraints for parallel sparse applications. In: IPDPS'05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 11, Washington, DC, USA, IEEE Computer Society (2005) 231.1
3. Pinheiro, E., Bianchini, R., Carrera, E., Heath, T.: Dynamic cluster reconfiguration for power and performance. In Benini, L., Kandemir, M., Ramanujam, J., eds.: Compilers and Operating Systems for Low Power. Kluwer Academic Publishers (2002)
4. Chase, J.S., Anderson, D.C., Thakar, P.N., Vahdat, A.M., Doyle, R.P.: Managing energy and server resources in hosting centers. In: SOSP '01: Proceedings of the eighteenth ACM Symposium on Operating Systems Principles, Banff, Alberta, Canada, ACM Press (2001) 103–116
5. le Mouël, F., André, F., Segarra, M.T.: AeDEn : An adaptive framework for dynamic distribution over mobile environments. Annals of telecommunications **57** (2002) 1124–1148
6. Gallard, P., Morin, C.: Dynamic streams for efficient communications between migrating processes in a cluster. In: Euro-Par 2003: Parallel Processing. Volume 2790 of Lect. Notes in Comp. Science., Klagenfurt, Austria, Springer-Verlag (2003) 930–937
7. Krauter, K., Buyya, R., Maheswaran, M.: A taxonomy and survey of grid resource management systems for distributed computing. Software–Practice and Experience **32** (2002) 135–164
8. Berc, L., Wadia, N., Edelman, S.: VMWare ESX server 2: Performance and scalability evaluation. Technical report, IBM (2004)
9. Whitaker, A., Shaw, M., Gribble, S.D.: Scale and performance in the Denali isolation kernel. In: Proceedings of the 5th symposium on Operating systems design and implementation (OSDI), Boston, MA, USA (2002) 195–209
10. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: Proceedings of the 19th ACM Symposium on Operating Systems Principles, Bolton Landing, NY, USA, ACM Press (2003) 164–177
11. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. Lecture Notes in Computer Science **2150** (2001) 1–??
12. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live migration of virtual machines. In: Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI '05), Boston, MA, USA (2005)