Reducing Wasted Resources to Help Achieve Green Data Centers

Jordi Torres, David Carrera, Kevin Hogan, Ricard Gavaldà, Vicenç Beltran, Nicolás Poggi
Barcelona Supercomputing Center (BSC) - Technical University of Catalonia (UPC)
torres@ac.upc.edu

Abstract

In this paper we introduce a new approach to the consolidation strategy of a data center that allows an important reduction in the amount of active nodes required to process a heterogeneous workload without degrading the offered service level. This article reflects and demonstrates that consolidation of dynamic workloads does not end with virtualization. If energy-efficiency is pursued, the workloads can be consolidated even more using two techniques, memory compression and request discrimination, which were separately studied and validated in previous work and are now to be combined in a joint effort. We evaluate the approach using a representative workload scenario composed of numerical applications and a real workload obtained from a top national travel website. Our results indicate that an important improvement can be achieved using 20% less servers to do the same work. We believe that this serves as an illustrative example of a new way of management: tailoring the resources to meet high level energy efficiency goals.

1. Introduction

Companies are now focusing more attention than ever on the need to improve energy efficiency. Up to now, the notion of "performance" has been much related with "speed". This requires data and supercomputing centers to consume huge amounts of electrical power and produces a large amount of heat that requires expensive cooling facilities. Besides the cost of energy, a new challenge is the increasing pressure to reduce the carbon footprint. Due to many UK and EU regulations and campaigns demanding greener businesses, a cap-and-trade system for carbon credits is to be introduced in 2010. Since current energy costs are rising, and Data Center equipment is stressing the power and cooling infrastructure[15], nowadays there is a big interest in "Green" data and supercomputer centers [9,28]. In this area, the research community is being challenged to rethink data center strategies, adding energy efficiency to a list of critical operating parameters that already includes service ability, reliability and performance.

A large variety of power-saving proposals have been presented in the literature such as dynamic voltage scaling and frequency scaling [14, 23]. However, some authors [12, 7] have argued that workload consolidation and powering off spare servers is a good effective way to save power and cooling energy. The low average utilization of servers is a well known cost concern in data center management. It has only been a short while since "One application – one server" was the dominant paradigm. This situation clearly implies server sprawl where the servers are underutilized. Data centers started to solve this by packing through consolidation to reduce the number of machines required. Server consolidation implies combining workloads from separate machines and different applications into a smaller number of systems and has become very popular following the advances in virtualization technologies [16]. This solves some interesting challenges; less hardware is required, less electrical consumption is needed for server power and cooling and less physical space is required. This is a widely adopted strategy used by companies [27] to increase the efficiency in managing their server environment and is assumed to maximize the utilization of their existing resources. As we will discuss further in this paper, we should consider new techniques complementary to consolidation to dramatically reduce the energy consumption and further reduce the resources required. Request discrimination is introduced to identify and reject those requests that consume system resources but have no value for an application (e.g. requests coming from web crawlers created by competitor businesses for spying purposes). We will also consider another technique, memory compression, which converts CPU power into extra memory capacity to overcome system underutilization scenarios caused by memory constraints. We study the use of these techniques by describing a representative scenario composed of a realistic heterogeneous workload. The techniques described here reduce the number of nodes necessary to meet a certain service level criteria. The main contribution of this article is to reflect and demonstrate that the consolidation through virtualization of heterogeneous

978-1-4244-1694-3/08/$25.00 ©2008 IEEE

workloads, on its own, does not go far enough in saving energy, and we will present ways of rescuing resources by reducing the wastage. The rest of the paper is organized as follows. Section 2 describes the basics of dynamic resource management and the techniques that we applied in our work. Section 3 discusses our studies and obtained results. In Section 4 we study the related work found in the literature. Finally, some conclusions and future work are discussed in sections 5 and 6.

2. Consolidated environment

2.1 Managing a consolidated and virtualized environment

Consolidation and virtualization can be combined to reduce the management complexity of large data centers as well as to increase the energy efficiency of such a system. But even in a scenario where the resources are consolidated and virtualized, utilizing all the capacity of the components that are switched on (and consuming power) is not always easy. Deciding a collocation of a set of applications in a node to perfectly fit and exploit all the resources of the system is a hard problem to solve, especially when tenths or even hundreds of nodes and applications can be found in a data center. Furthermore, the fact that the demand associated with each resource from a given application may not be related in any way to its demand for other resources (i.e. an application with a large memory footprint may not be very demanding in terms of CPU power) makes it a structural problem that requires some constraints to be relaxed if we want to overcome it.

The techniques proposed in this paper are studied in the context of a shared data center running a set of applications which are operated by an automatic service management middleware such as that described in [21,25] but other approaches could be considered. The management middleware monitors the actual service level offered to each running application and dynamically changes the configuration of the system to make the applications meet their goals. In particular, the system has to decide in what nodes these instances are going to be placed: this is what is known as the placement problem.

The placement problem is to find a placement for applications on servers, known to be NP-hard [11,6] and heuristics must be used to solve it. Given a certain workload, changing the allocated CPU power for an application makes a significant difference in the service level offered. Changing the amount of memory allocated to an application results in an even higher impact, because the application can either be placed or it can’t, depending on whether the amount of memory reserved to run it is enough or not. This leads to a scenario where the placement problem can be represented as two different problems: placing applications following memory constraints and spreading CPU resources amongst the placed instances. The objective of our work is not to try and solve the placement problem but to introduce a new degree of freedom to allow the system find a new set of application placements that offer the same service level to each application but require different resource allocations. This objective is achieved by relaxing the allocation constraints, and by relaxing the hardest constraint in the system: the available physical resources in each node of the data center.

For the purpose of our work, we assume the data center controls the resources allocated to each application using virtualization technology [2], and runs each instance inside a virtual machine container. In the scope of this paper we use a simple instance placement algorithm to illustrate the benefits of our techniques, but it doesn’t have to be limited to this. In order to better define the placement scenario, it can be assumed that the system is able to derive the relation between resource allocation and obtained service level for each application in the system, as is reported in [21].

2.2 Recycling through resource transformation

After virtualizing a system, some resources may still be unused by the applications. The demand associated with each resource in the system for a given application may not be related in any way to the demand for other resources (i.e. an application with a large memory footprint might have a small CPU power footprint), which can potentially lead to an underutilization of some resources in the system. To illustrate this situation we will show a typical placement problem: some applications could all be placed on a node in terms of CPU power (they would meet their performance goals), but the memory capacity of the system makes it impossible to place all the applications together. As a result, an extra node must be used to place one of the applications, and both nodes end up being underutilized in terms of CPU.

Memory compression is a widely studied topic that can be helpful in the placement problem. It allows the system to increase the density of the placement (i.e. the number of applications on a node) and better exploit the resources of the system. This process can be understood as a resource transformation: CPU cycles are converted into extra memory. The amount of extra memory produced using this technique can potentially go beyond consolidation through virtualization in two aspects: firstly, allowing the placement of an extra application that did not fit in a node before, therefore reducing node underutilization; and secondly, increasing the performance of a placed application that, with a given amount of memory, can still run but at a fraction of the maximum achievable performance (i.e. producing a big volume of swapping activity).

Some of our recent work is focused on revisiting the memory compression topic by targeting advanced hardware architectures (current multiprocessor and multi-core technologies such as CELL and Niagara [3]). Our study concludes that memory compression can be carried out without observing a significant performance impact in many commercial applications (the study is performed using the SPECWeb2005 [26] application). The relation between the CPU power dedicated to compress memory and the memory gain obtained for three different levels of memory compressibility is represented in [3]. Obviously, this relation is always defined by the level of memory compression achievable given a set of applications. From the point of view of the applications, the overhead produced by memory compression techniques is negligible because although accessing compressed data is slower than accessing regular memory, it is still faster than accessing a standard SCSI disk. This means that the reduction in swapping as well as being able to cache more data can still result in a performance improvement for most applications.

2.3 Reduction through discrimination

A fraction of the resources are wasted on work that yields no added value for the application or the company running it: consider a webserver for an e-commerce site, and the amount of work performed for customers that will not buy. An even greater problem is clients that request work that can be harmful to the system: consider requests to the webserver during denial-of-service attacks, or the traffic generated by malicious bots or web crawlers created by competitor businesses for spying purposes. Any potentially harmful requests that can be detected should be banned.

Let us comment on the work in [19], which addresses the problem of detecting malicious bots for the purpose of banning them. The case study in this work is a national online travel agency working as an electronic intermediary between customers and service providers (such as airline companies). More precisely, in [19,20] and later experiments we have used web traffic logs from different periods of the year, ranging from one day to a week of traffic, with up to 3,7 million transactions. Each transaction is a particular request to the web site (e.g. a page download, possibly including many database searches). Transactions are grouped into user sessions, with an average length of about 8 transactions per session for non-buying sessions, and 18 transactions per session for sessions that end in a purchase. Around 6.7% of transactions belong to sessions which will end in purchase. The problem tackled in [19] is that of detecting stealing bots in e-commerce applications. Content stealing on the web is becoming a serious concern for information and e-commerce websites. In the practices known as web fetching or web scraping [10], a stealer Bot simulates a human web user to extract desired content off the victim's website. Not only that, but in a B2B scenario, the victim incurs the costs of searching the provider's web for a supposed "customer" that will never buy, and loses the real customers who will instead buy via the stealer's site.

The work in [19] investigated whether it was possible to identify with reasonable certainty bots accessing a web site for automated banning so that the system could stop the corresponding session and free the allocated resources. In the online travel agent website, [19] concluded that around 15% to 20% of the traffic corresponds to bots other than simple crawlers. Note that a feature of stealer bots is the large amount of search requests, hence this large traffic figure. Applying machine learning techniques, the authors were able to detect around 10%-12% of the total traffic as bots with a low % of "false alarms" and negligible overhead at runtime. This percentage of traffic could be banned in the real scenario, even when the system is not overloaded, since it is actually harmful to the company's interests to serve them. While the interest of the authors in [19] is leveraging revenue loss from the spurious transactions, it is easy to see this technique as a way to reduce the allocated resources: If we expect that we could ban 10%-12% of the incoming traffic, we could reduce the resources assigned to the application by a similar percentage when deploying it.

In any case, a key point is finding the relation between load reduction and resource reduction. The experiments in some of our EU-funded projects [5], where we have researched the dynamic management of resources, let us conclude that there is essentially a linear relation among load volume and CPU usage. That is, if we reduce the number of requests by 10% or 15%, the CPU requirements will be reduced by at least 10% or 15%. The reduction will probably be larger if the transactions we discard are especially heavy ones (which is the case for stealing bots). We cannot at this moment, make similar claims for other resources, such as memory, which we are still investigating. For this reason we center our work only on CPU even though we believe that we will be able to extend the conclusions to other resources soon.

3. Experiments

In this section we evaluate our proposal. First we will demonstrate the waste of resources using the state-of-the-art automatic management middleware, considering only a small set of web applications in order to facilitate the explanation of the idea. Later we will demonstrate the impact of the proposal using a simulation that reproduces a heterogeneous workload scenario.

3.1 Waste of resources

In order to demonstrate that the current state of virtualization is wasting resources, we will consider a set of 4 different web applications. The characteristics of each application are described in table 1. Neither allocation restrictions nor collocation restrictions are defined, but placement is still subject to resource constraints, such as the node memory and CPU capacity.

Table 1. Memory and CPU required by the Applications

	App 1	App 2	App 3	App 4
Minimum Memory	2300 MB	1300 MB	1100 MB	1000 MB
Maximum CPU required	2200Mhz	2000 Mhz	2000 Mhz	1900 Mhz

We consider that each server has four 2.2GHz CPUs and 4GB of memory (based on an IBM JS21 blade). We assume that the virtualization overhead is 1GB of memory and 1 CPU. This assumption is based on our previous experience [5]. Table 2 summarizes the specifications of each node. Notice that application 1 can not be placed together with any other application because of the memory constraints. Applications 2, 3 and 4 can be collocated, but only two of them can be placed together in each node. Table 1 only indicates the maximum CPU required (spike) for each application over time to meet its service level goals. That is, the maximum value for the minimum amount of CPU power that must be allocated to each application if its service level goal is to be met. There is no overloading at any point during the experiment (the aggregated CPU power can satisfy the requirement of all applications over the time). This placement leads to a situation where the three nodes are clearly underutilized in terms of CPU since the maximum amount of CPU required at any point during the execution is 8100 Mhz while we have a total of 19800 Mhz at our disposal from the 3 servers.

Table 2. Memory and CPU capacity of each node before and after considering the virtualization overhead

No virtualization		Virtualization overhead	
CPU capacity	Memory capacity	Effective CPU capacity	Effective mem. capacity
4x 2.2Ghz	4096MB	3x 2.2Ghz (6.6 Ghz)	3072 MB

3.2 Tailoring of resources

3.2.1 Baseline placement. In this section we describe what a modern management middleware would be expected to do in the scenario described above. As we said before, application 1 can not be placed together with any other of the other applications because of its memory requirements. Given that the CPU demand of application 1 can be satisfied by one single node, we assume that this application would be placed in one node for the whole length of the experiment. The other applications must be placed in the two remaining nodes. Given that all three applications don't fit in one single node due to the memory constraint, two of them will have to be placed together while the other application will be alone on one node. Thus, the placement algorithm should decide at this point what two applications are going to be placed together. For this experiment we decide to pick application 2 and 3 to be deployed on node 2, and application 4 to be placed in node 1. Notice that other choices are possible but that the result would be analogous to that presented here.

3.2.2 Adding Tailoring Resources. At this point, we introduce the use of memory compression to increase the memory capacity of a node on demand. The memory, as discussed in Section 2.2, is produced at a cost in terms of CPU power. Notice that in the scenario described in section 3.1, memory constraints lead to a situation where the three nodes are clearly underutilized in terms of CPU power. Looking at the data provided in Section 3.1 (which is based on real experiments conducted with realistic applications on top of an IBM JS21 blade server) one can observe how, depending on the compression rate achievable for a given set of applications placed in a node, a relation can be established between the CPU power required to compress memory and the increase in available memory observed. In the scope of this example, we assume an achievable compression factor of 47%, and will use an increased memory capacity for each active node of 6GB at a cost of 1320MHz of CPU power. With the new constraints, a new range of possible placements is opened up, including the option of having all four applications placed together on one single node if the amount of CPU power required can be satisfied by that single node. When the aggregated CPU demand exceeds the capacity of a single node an application is migrated to a second node which is switched on for this purpose. If at a given point the aggregated CPU demand for a set of applications can be satisfied again with one single node, all these applications are placed in the same node again. More details on this can be found in the report [32].

Regarding the CPU, we make use of the request discrimination technique described earlier in section 2.3. With the help of machine learning it is possible to determine the characteristics of requests that are of no benefit to the company running the service; for example Bots. These have been estimated to account for 15% to 20% of all web traffic so the filtering out of these can reduce the load by a significant amount. We assume that we can filter out 10% of all the web traffic, based on the figures that the authors of [19] achieved in their work. There is a direct linear correlation between the amounts of CPU required to process requests, so if we reduce the number of requests by 10%, we are effectively reducing the amount of CPU needed by the same amount [5].

3.3 Heterogeneous workload

3.3.1 Workload Description. To generate the heterogeneous workload we modify and extend the previous workload, described in table 1, by creating a scenario in which we consider a total of 12 applications. The first set of 4 are designated as "Web" applications, and have the same memory and CPU requirements as defined in table 1, but are scheduled to run at times that conform to the workload of a travel agency website during the high-season. There are clearly visible patterns in the load over the day and week, containing spikes during the day, troughs at night and generally lower loads at the weekend. The next set of 4 applications are also "Web" applications but are scheduled to run throughout the whole simulation. They use the same memory as before but have a variable CPU demand that is roughly in accordance with the demand on the travel agency website. Only the CPU is varied since it has been discovered through other work [5] that there is a highly linear correlation between the CPU and the workload level, whereas the same does not hold true for the memory.

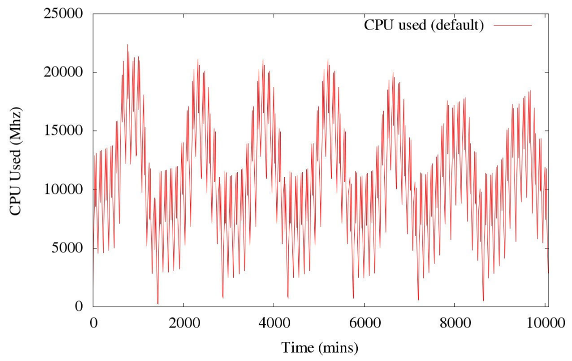

Figure 1. The amount of CPU used over time in default setup

The last set of 4 applications are numerical applications which do not display the same workload characteristics as the previous ones. We have taken two specific, but representative, numerical applications for this simulation; BLAST[29] is a bioinformatics application which generally has short job running times (in the order of 15 to 30 minutes), and ImageMagick [30] is an image rendering application which has longer job running times (in the order of 120 minutes). The numerical applications are considered to have static CPU and memory demands when they are running and we have scheduled them so that the short running jobs arrive every hour or two and the long running jobs only arrive once or twice a day. The exact needs of each numerical application can be seen in Table 3.

Table 3. Requirements of the Numerical Applications

Application	Memory	CPU (Mhz)	Running Time	# of runs per day
BLAST1	550 MB	4400	15 min	24
BLAST2	550 MB	4400	30 min	12
ImageMagick1	750 MB	2200	127 min	1
ImageMagick2	750 MB	2200	100 min	2

3.3.2 Baseline Placement. After generating the heterogeneous workload above, we first ran the simulation with the default baseline placement algorithm. The workload pattern of the travel website is easy to see in figure 1, where we show the amount of CPU used over a simulated time of one week.

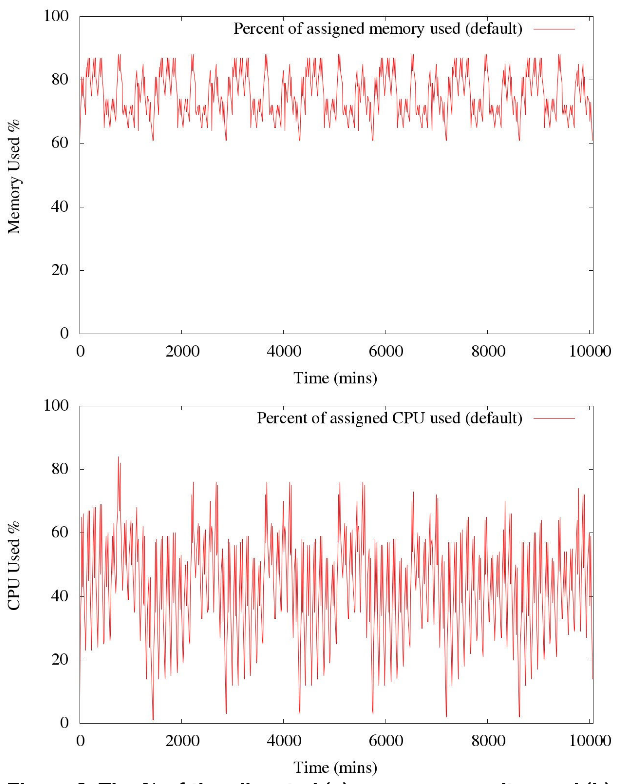

Figure 2. The % of the allocated (a) memory capacity, and (b) CPU, being used under the baseline placement

While we are benefiting from consolidation and virtualization, reducing the number of servers we need to run the applications when compared with the "one application – one server" paradigm, there is still considerable wastage in the system. We can measure the exact amount of resources being wasted at each moment in time by subtracting the total load from the sum of the capacities of each server allocated to us during that moment in time. For example, if we have 2 servers allocated to us and they both have 1GB of memory free, this means that we are wasting 2GB of memory in total. For 2 servers this means that we are only using 66% of the memory available to us since the effective memory capacity available to each server after virtualization is 3072 MB. Conversely, it also means that we are wasting 33% of that resource. Figures 2 show the percentage of the allocated resources that are being used/wasted over time using the default baseline placement, when considering memory and CPU respectively. The graphics show that there is very little memory being wasted, but the CPU appears to be highly underutilized. It suggests that the memory is acting as the largest constraint when we are placing applications on the servers.

3.3.3 Tailored Placement. In the next stage of our experiment we used the same workload with a simple demand based placement algorithm which can make use of the compression and request discrimination techniques. Note that during this simulation the numerical applications do not take advantage of any tailoring. The advantages that numerical applications can gain from the techniques used in tailoring are currently being investigated, and while it looks promising that they can benefit from it also, the exact figures are not yet known so have been left out of the current work.

From figure 3 it can be seen that the CPU needed in the tailored scenario is slightly higher than the CPU needed in the default baseline scenario. By using request discrimination we are reducing the demand of the Web applications by 10%, but we experience a hit on the CPU due to the compression technique. In the worst case for our workload it amounts to an extra 2068 Mhz, which is equivalent to 31% of a single server's CPU capacity.

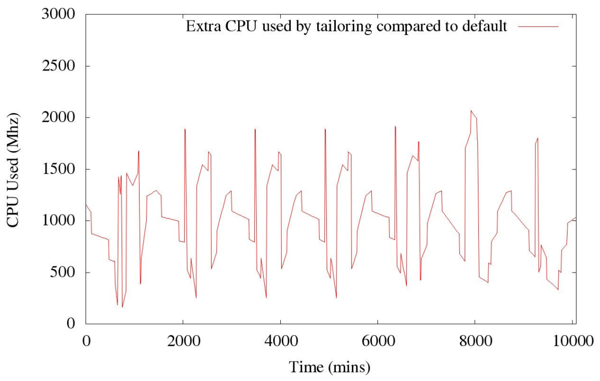

Figure 3. The extra amount of CPU used in the tailored setup compared to that used in the default

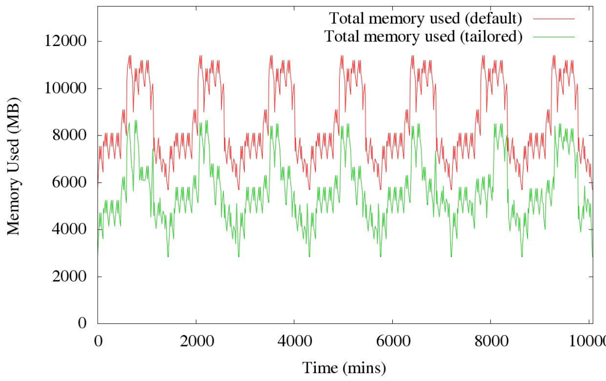

Figure 4. The amount of memory used in the default setup versus that used with tailoring

The next graphic in figure 4 shows us the other side of the coin as we can see the large difference between the memory requirements of the tailored environment and the default one over the time of the simulation. For the memory, the tailored environment requires a considerable amount less since it is able to squeeze more out of the memory available to it when it uses compression. We have essentially traded some of our excess CPU power for extra memory when we used the tailoring.

The big advantage that we gain by using tailoring can be summed up in figure 5, where we show the difference in the amount of servers required to satisfy the workload over time. Over the whole simulation time the tailored setup never requires more servers than the default setup. At the lowest points of demand our tailored environment is able to get by with a single server, whereas at those points the standard environment needs three. This could be used to achieve a saving in the cost of running the website, and a reduction to its carbon footprint, especially if the resources are being contracted dynamically based on demand.

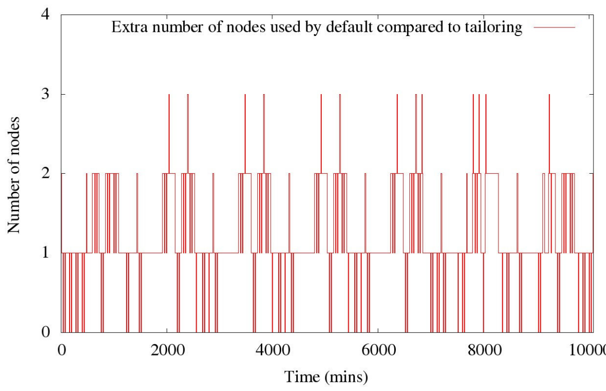

Figure 5. The difference in the amount of nodes needed for the default setup versus those needed with tailoring

Even if the website is not dynamically obtaining servers to deal with the load and has a static set of servers in-house, the tailored setup would allow them to use 20% less servers, since the maximum amount of servers needed by the tailored setup to satisfy the workload is 4 whereas the corresponding value for that of the default setup is 5. This represents a significant saving in the hardware needs, and when put in this context, the default setup appears to be suffering from over-provisioning. There is a huge environmental impact by being able to turn off unneeded servers so this is an important step in making Data centers "greener".

During the simulation we also recorded the amount of migrations that were needed to achieve the placements. The figures for the default baseline scenario and the tailored one were 27 and 192 respectively. While the difference in these two figures appears large, since the experiment had a simulated time of one week, it should not pose much problems. It works out that there would be just over one migration per hour using the tailored set up. We are currently doing work in our group to research ways of having transparent migrations for users using the techniques in [1] which may be of help in this process.

3.3.4 Bigger Workloads. To establish how the tailored and default scenarios cope with larger workloads, we multiplied our previous load by a factor of four and reran the simulation. Under this load, which has a set of 48 applications looking for resources in the same way as the previous simulation, the differences between the default setup and the tailored one become even more pronounced. Figure 6 shows that the tailored environment requires fewer servers throughout the entire simulated time of this heavier load. The maximum amount of servers required by the default and tailored setup are 18 and 14 respectively, which would allow a saving of just over 20%. The number of migrations in both cases increased by a factor just greater than 4.

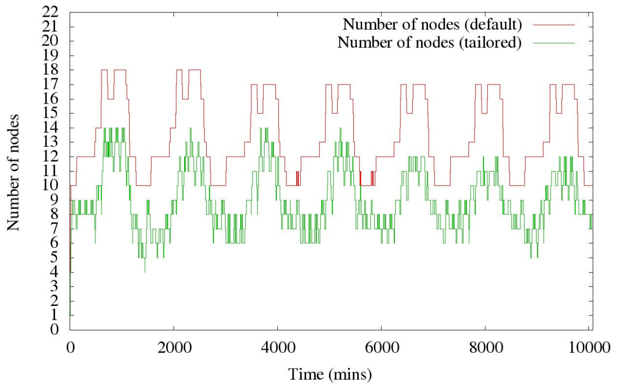

Figure 6. The amount of nodes needed for the default setup versus those needed with tailoring

4. Related Work

The placement problem itself is out of the scope of our work but the techniques described in this paper can be helpful to any placement algorithm, by relaxing one of the hardest constraints they have to deal with: the system capacity. Existing dynamic application placement proposals provide automation mechanisms by which resource allocations may be continuously adjusted to the changing workload. Previous work focuses on different goals, such as maximizing resource utilization [11] and allocating resources to applications according to their service level goals [21, 6]. Our proposal could apply to and improve any of those. Space does not permit a full discussion of the various types of virtualization and their relative merits here; the reader is referred instead to [16,8,2]. The dynamic allocation of server resources to applications has been extensively studied [4,6,11,13,22], however any of these proposals can go beyond virtualization and could be beneficiaries of the proposals presented in this paper. Another important issue is the problematic consolidation of multi-tier applications considered in [18] that can be complementary to our proposal. Also of great importance is the topic considered in [17], regarding the power-efficient management of enterprise workloads which exploits the heterogeneity of the platforms. Our proposals could be included in the analytical prediction layer proposed by the authors. Finally let us remark that our proposals could be combined with power-saving techniques at the lowest level such as dynamic voltage scaling and frequency scaling [7,14,24]. In a recent work [12], the authors use frequency scaling in a scheme that trades off web application performance and power usage while coordinating multiple autonomic managers. In this case the proposals of this article could be included in the utility function that they are using.

5. Conclusions

In this paper we demonstrate how consolidation with energy efficiency goals still has a long way to do beyond the use of virtualization. In this work, we identify new opportunities to improve the energy efficiency of systems, reducing the resources required, without negatively impacting the performance or user satisfaction. The obtained results show that the combined use of memory compression and request discrimination can dramatically boost the energy savings in a data center. Our interest as a group involves creating power-aware middleware to contribute to building energy-efficient data centers. The increased awareness of green issues is simply accelerating improvements in efficiency that any data center should have been implementing in the near future anyway. Somehow, the next generation of computing systems must achieve significantly lower power needs, higher performance/watt ratio, and higher reliability than ever before.

6. Future Work

We would like to extend our work to consider other techniques that could be added in terms of availability such as self-healing techniques [1] and therefore take better advantage of the resources available. We are already working on the implementation of a prototype system that applies the techniques described in this paper. We will also extend the tailoring techniques further than just web applications and take numerical applications into account. The advantages that numerical applications can gain from the technique used to compress memory are currently being investigated in our group and it looks promising that they can benefit from it. Many systems kill jobs after an estimated time by the user (indicated in the user-provided job script) has elapsed and it is a well-documented fact that user-provided runtime estimates are very often inaccurate [31]. In this case we are working to find a way of using discrimination techniques to detect and filter jobs that have no chance of completing successfully.

7. Acknowledgments

This work is supported by the Ministry of Science and Technology of Spain and the European Union (FEDER funds) under contracts TIN2004-07739-C02-01 and TIN2007-60625.

8. References

[1] J. Alonso, L. Silva, A. Andrzejak, P. Silva and J. Torres "High-Availability Grid Services through the use of Virtualized Clustering". The 8th IEEE/ACM Int.l Conf. on Grid Computing (GRID 2007). September 19-21, 2007, Austin, Texas, USA.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization" in Symposium on Operating Systems Principles (SOSP), Bolton Landing, NY, 2003

[3] V.Beltran, J. Torres and E. Ayguade "Improving Disk Bandwidth-Bound Applications Through Main Memory Compression" MEDEA Workshop MEmory performance: DEaling with Applications, systems and architecture. Brasov, Romania. In conjunction with PACT 2007 Conf. Sept. 2007.

[4] N. Bobroff, A. Kochut, and K. Beatty, "Dynamic placement of virtual machines for managing SLA violations," in Integrated Network Management, Munich, Germany, May 2007.

[5] BREIN Project. http://www.eu-brein.com/

[6] D. Carrera, M. Steinder, I. Whalley, J. Torres and E. Ayguadé. Utility-based Placement of Dynamic Web Applications with Fairness Goals. Submitted to IEEE/IFIP Network Operations and Management Symposium (NOMS 2008).

[7] J. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," in ACM Symposium on Operating Systems Principles, 2001.

[8] R. Figueiredo, P. Dinda, and J. Fortes, "A case for grid computing on virtual machines" in International Conference on Distributed Computing, Providence, RI, May 2003.

[9] Green Grid Con.sortium , http://www.thegreengrid.org/

[10] Hepp, M., D. Bachlechner, and K. Siorpaes. Harvesting Wiki Consensus - Using Wikipedia Entries as Ontology Elements. Proceedings of the ESWC2006, Budva, Montenegro, 2006.

[11] A. Karve, T. Kimbrel, G. Pacifici, M. Spreitzer, M. Steinder, M. Sviridenko, A. Tantawi, "Dynamic placement for clustered web applications" In WWW Conf., Scotland (2006)

[12] J. O. Kephart, H. Chan, R. Das, D. W. Levine, G. Tesauro, and F. R. an C. Lefurgy, "Coordinating multiple autonomic managers to achieve specified power-performance tradeoffs," in IEEE Fourth Int. Conf. on Autonomic Computing, Jun. 2007.

[13] T. Kimbrel, M. Steinder, M. Sviridenko, A. Tantawi, "Dynamic application placement under service and memory constraints". In International Workshop on Efficient and Experimental Algorithms, Santorini Island, Greece (2005)

[14] B. Khargharia, S. Hariri, and M. S. Youssif, "Autonomic power and performance management for computing systems," in IEEE Int. Conference on Autonomic Computing, June 2006.

[15] J. Koomey. "Estimating Regional power consumption by servers: A technical note" Dec 5, 2007. Available at: http://www.amd.com/us-en/assets/content_type/DownloadableAssets/Koomey_Study-v7.pdf

[16] S. Nanda and T. Chiueh, "A survey of virtualization technologies" Stony Brook University, Tech. Rep. 179, 2005.

[17] R. Nathuji, C. Isci, E. Gorbatov. "Exploiting Platform Heterogeneity for Power Efficient Data Centers". In IEEE Fourth International Conference on Autonomic Computing, June. 2007.

[18] P. Padala, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, K. Salem, and K. Shin. "Adaptive control of virtualized resources in utility computing environments". In Proc. European Conference on Computer Systems (EuroSys'07), 2007.

[19] N. Poggi, J.L. Berral, T. Moreno, R. Gavaldà and J. Torres. "Automatic Detection and Banning of Content Stealing Bots for E-commerce". In Workshop on Machine Learning in Adversarial Environments for Computer Security (NIPS 2007).British Columbia, Canada. Dec. 2007

[20] N.Poggi, T. Moreno, J. Berral, R. Gavaldà, J. Torres. "Web Customer Modelling for Automated Session Prioritization on High Traffic Sites". In 11th International Conference on User Modelling. Corfu, Greece, June, 2007.

[21] M. Steinder, I. Whalley, D. Carrera, I. Gaweda and D. Chess. "Server virtualization in autonomic management of heterogeneous workloads". In 10th IFIP/IEEE International Symposium on Integrated Management (IM 2007), May 2007.

[22] C.-H. Tsai, K. G. shin, J. Reumann, and S. Singhal, "Online web cluster capacity estimation and its application to energy conservation," IEEE Transactional on Parallel and distributed Systems, vol. 18, no. 7, 2007.

[23] X. Wang, D. Lan, G. Wang, X. Fang, M. Ye, Y. Chen, Q. Wang. "Appliance-Based Autonomic Provisioning Framework for Virtualized Outsourcing Data Center". in IEEE Fourth International Conference on Autonomic Computing, June 2007.

[24] M. Wang, N. Kandasamy, A. Guea, and M. Kam, "Adaptive performance control of computing systems via distributed cooperative control: Application to power management in computing clusters," IEEE 3rd International Conference on Autonomic Computing, June 2006.

[25] "WebSphere eXtended Deployment," http://www-306.ibm.com/software/webservers/appserv/extend/

[26] Standard Performance Evaluation Corporation. SPECweb2005. http://www.spec.org/web2005/

[27] "Usage of Virtualization Technology at Small and Midsize Businesses", Computerworld White Paper, October 2007.

[28] The Green500 list. http://www.green500.org/

[29] (BLAST), The National Center for Biotechnology Information. http://www.ncbi.nlm.nih.gov/blast/

[30] ImageMagick(TM), http://www.imagemagick.org/

[31] C.B. Lee, A. Snavely. On the User–Scheduler Dialogue: Studies of User-Provided Runtime Estimates and Utility Functions International Journal of High Performance Computing Applications, Vol. 20, No. 4, 495-506 (2006).

[32] BCN Placement Simulator. Available: http://PlacementSimulator.energy-efficient-computing.org/