Dynamic Configuration of Virtual Machine for Power-proportional Resource Provisioning

Ibrahim Takouna
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany

Wesam Dawoud
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany

Christoph Meinel
Hasso Plattner Institute
University of Potsdam
Potsdam, Germany

emails:{ibrahim.takouna, wesam.dawoud, christoph.meinel}@hpi.uni-potsdam.de

ABSTRACT

Power management is a major concern in datacenters. Using virtualization in datacenters enables applications' consolidation to reduce power consumption. However, processors consume the most fraction of the host's power. Furthermore, the rising number of cores in a single processor extensively contributes to the increase of power consumption if there are no efficient power management solutions. These solutions are considered inefficient if they do not take into account the number of active physical cores and the configuration of a virtual machine, which runs a certain job. In this paper, we analyze power consumption of a multicore processor and develop a CPU power model and a performance model based on the number of active cores and frequency. Then, we propose an optimization solution for power and performance management in virtualized servers. Our optimization model achieves power proportionality and guarantees performance; it is based on a mixed integer programming model. The optimization model provides an optimum configuration for both a host and its VMs in terms of their number of virtual CPU and their proportional weight. Finally, we demonstrate efficiency of the proposed solution via experiments. The results show that between 23% and 48% savings in power consumption compared to a typically provisioned power by hypervisor performance governor.

Categories and Subject Descriptors

K.6 [**MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS**]: General; K.6.2 [**Installation Management**]: Performance and usage measurement

General Terms

Managment, Performance, Experimentaion

GCM'2011, December 12th, 2011, Lisbon, Portugal.

Keywords

Power-aware, Optimization, Virtulization, Multicore

1. INTRODUCTION & RELATED WORK

Datacenter power consumption has become a significant concern with the rapid emergence of cloud services such as Amazon EC2. For example, Hamilton [1] has reported that Amazon's datacenters are facing a highly increased power demand where the servers consume 59% of the total power supply. Furthermore, the U.S. Environmental Protection Agency (EPA) reported that the energy consumption of the datacenters located in U.S. consumed 61 billion kilowatt-hours in 2006, which costs $4.5 billion [2]. This amount of money represents a substantial portion of a datacenter's budget [3].

Furthermore, servers are equipped with multicore processors. According to Moore's law, not only are a number of cores in a single processor doubled every 18 months, but also the power dissipation is doubled. In fact, the processor is the component that consumes the most dynamic power of a computer system [2][4], there have been many proposed approaches for datacenters power management based on Dynamic Voltage Frequency scaling (DVFS) [5]-[10]. DVSF mechanism enables processors to run on different performance states: P0-Pn. CPU P state saves power by changing CPU frequency and voltage. P0 has the highest frequency and the highest power consumption meanwhile Pn has the lowest frequency and the lowest power consumption.

Currently, datacenters leverage virtualization technology such as Xen and VMware. Using virtualization increases servers' utilization by enabling applications consolidation onto fewer physical servers and turning off unused servers to save power. In this work, we used Xen as an open source hypervisor which is widely used in clouds. Its default scheduler called credit-scheduler. Credit-scheduler is a proportional fair share CPU scheduler. The amount of credits assigned for a virtual machine (VM) is based on the capacity and weight of the VM. For example, a half of CPU cycles will be given to a VM with a weight of 1 when is scheduled with a VM with a weight of 2 on a contended server.

Several solutions have been implemented in particularly for virtualized environments at datacenter level [11]-[13]. Kusic et al. [11] have developed a dynamic resource provisioning framework based on lookahead control, which estimates the future workload demand. They concluded that the intensity of the workload directed at the VMs does not

affect the power consumption. Additionally, they reported that the power consumed by a host machine only affected by the number of VMs running on it regardless of the arrival rate experienced by the VMs. In this paper, we show different results where the power computation changes with the workload intensity even when we have one VM. Wang et al. [12] have implemented a central controlled global optimization of resource allocation for multi-tier services. However, the centralized controller could cause a single point of failure. Additionally, collecting information about computing resource allocation to each VM hosted in every server [13] causes an overhead. Our solution differs from [11]-[13] by considering power optimization at server level (i.e., local optimization). As a future extension, we will integrate our solution into virtualized cluster to achieve both levels of optimization: local at the server level and global at the cluster level.

NapSAC [14] presents a totally different approach compared to previous works. They used heterogeneous servers to build a cluster of web applications. The servers have different power consumption and performance capabilities. Unfortunately, this approach is not applicable to current datacenters that could consist of homogeneous multicore processors. These processors have the same power consumption and computing capabilities. Importantly, in this paper, the optimization problem solver could give a solution with heterogeneous settings of cores in terms of frequency. This allows the server to run on a fine-grained level of power consumption. In contrast, Petrucci et al. [15] have implemented a dynamic optimization model for selecting efficient power configuration of a cluster. They only considered homogeneous configuration of the server. However, their work was implemented for cluster of web application.

A heuristic-based solution for the power-aware consolidation problem of virtualized clusters is presented in [16]. Generally, heuristic-based solutions could not guarantee finding a solution that is near to the optimal. In this work, a greedy KnapSack algorithm is used in order to provision an optimum or near optimum of both virtual machines (VM) and host configuration at any given time. Furthermore, our proposed solution could be suitable for virtualized High-Performance Computing applications in Clouds as we consider that a VM could execute a specific job with a certain workload demand. Likewise, Grids, which are based on virtualization, dedicate a shared pool of virtualized resources to job processing [17][18]. Throughout this paper, we use a VM term when discussing the number of virtual CPUs and VMs' weight; a job term is used when discussing the throughput of this job at specific configuration of a VM that hosts this job.

The purpose of this paper is to provide a fine-grained power provisioning in contrast with those approaches that are considered as course-grained where the frequency was set to the physical server as a unit regardless of the number of cores in its processor. Furthermore, this solution concerns power optimization at the server level by dynamic configuration of its VMs. To this end, we develop two models: the first model shows the relationship between the power and VM configuration (i.e., number of vCPUs) and the second one depicts the performance of VM with different configuration. Then, an optimization problem is formulated to find an optimum solution for both VMs and host configuration according to the number of jobs and the workload demand of each job. Finally, we present performance evaluation of our proposed solution to affirm its applicability and performance. The results show that with dynamic configuration of a host and its VMs better power savings could be achieved compared to the baseline. The baseline in this work is the default hypervisor performance-governor which mostly makes cores to run at the highest frequency to guarantee performance.

The rest of this paper is organized as follows. The following section presents models' development. Section 3 presents an overview of system architecture. In Section 4, we formulate the dynamic optimization problem. Performance evaluation and results are presented in Section 5. Finally, our conclusions and future work are presented in Section 6.

2. MODELS DEVELOPMENT

In this section, we present the experimental setup. Then, we present performance model and power model development for CPU-intensive jobs.

2.1 Experimental Setup

The evaluation experiments were performed on Fujitsu PRIMERGY RX300 S5 server that has a CPU Power measurement capability. It has a processor of Intel(R) Xeon(R) CPU E5540 with 4-cores. The frequency range is 2.53GHz to 1.59GHz. Each core enables 2-logical cores. The server is equipped with 12GB physical memory. The experiments were run on a virtualized server using Xen-4.1 hypervisor. To build two models, we used a CPU-intensive program EP Embarrassing Parallel, which is one of NAS Parallel Benchmarks (NPB) [19]. It generates pairs of Gaussian random deviates according to a specific scheme. EP is a multithreaded program which runs a number of threads corresponding to the VM's virtual CPU number. The system throughput was measured by Million Operations Per second (MOPs).

We measured the capacity of a VM for each frequency and vCPUs number combination. Notably, the number of vCPUs is corresponding to the number of active cores. Xenpm command [20] was used to set core's frequency and get the actual running frequency of each core. Finally, to measure CPU Power consumption, we used the CPU-Power measurement capability of our server. In our experiments, the percentile average was considered to get accurate power readings. More details about the two obtained models are presented in the next sections.

2.2 VM performance model

In this section, we present a model of VM performance. This model was built using a VM with different settings of the number of vCPUs. Then, we measured the job's performance (i.e., throughput MOPs) at each VM configuration and cores frequency combination. Figure 1 depicts the surface plot for VM performance with VM configuration and core frequency. It should be noted that we could achieve a certain performance level with different combinations of VM's configuration and cores' frequency. Figure 1 also shows that EP could scale almost linearly with frequency and number of cores. The same result has been found in [21] for BT and FT benchmarks. Takouna et al. [21] have conducted a sensitivity analysis of frequency for NPB suite benchmarks. In our solution, we exploit this linear relationship by increasing number of vCPUs to increase the

throughput of a VM instead of increasing cores' frequency. This will assist in achieving better throughput for multi-threaded applications and efficient utilization of multicores processors. However, as our concern is power efficiency, we study the consumed power for each combination in next section.

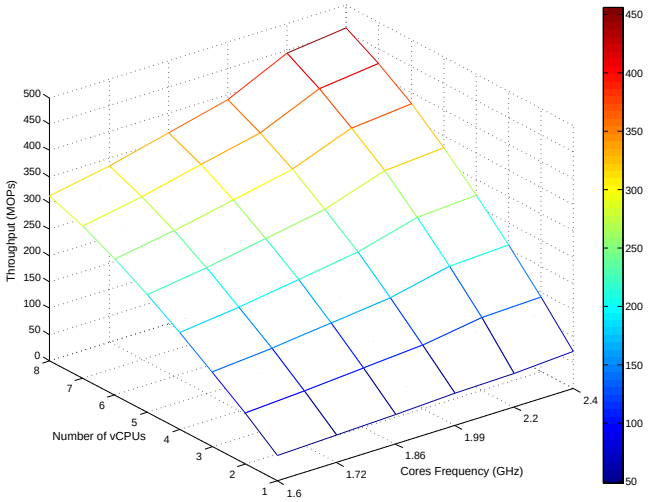

Figure 1: Host-VM Configuration: performance model.

2.3 Host power model

Here, a model of power consumption with Host-VM Configuration and cores frequency is presented. To construct this model, we created a VM with different settings of the number of vCPUs and cores' frequency combinations. Then, we measured the power consumption of the CPU at each configuration. Figure 2 depicts the surface plot for CPU power consumption with VM configuration and cores' frequency. This figure shows a non-liner relationship between frequency and the consumed power. On the other hand, it shows a linear relationship between number of vCPUs and the consumed power. Importantly, Figure 2 suggests utilizing the number of cores to achieve more power savings.

3. SYSTEM ARCHITECTURE OVERVIEW

We need to minimize the consumed power and satisfy throughput demand. The set of currently active cores and frequency configuration should be maintaining our objective function. The server hosts a number of VMs. It supports dynamic optimization according to demand change. The proposed system consists of three modules as depicted in Figure 3: Workload dispatcher, Power Performance Optimizer, and Host-VM Configuration manager.

The workflow of this system is as follows. We assume that load dispatcher has knowledge about the demand throughput for each job, and has knowledge of each VM capacity so it dispatches the jobs to a VM with configuration that achieves throughput demand. So, before the job is sent to a VM, the dispatcher informs the Power-Performance Optimizer about job demand. After the optimizer solves the optimization problem, it sends the suitable configuration of this job for Host-VM Configuration manager. Importantly, the proposed optimization problem was implemented using

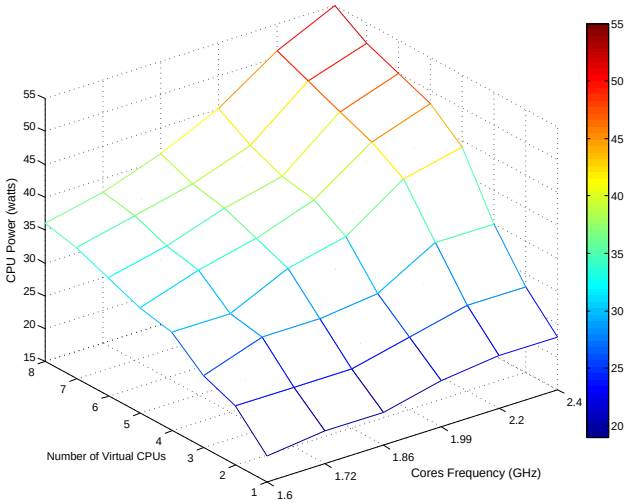

Figure 2: Host-VM Configuration: power model.

IBM ILOG CPLEX Studio [22] which implements efficient algorithms to search for configuration solutions. The problem was always solved in less than 1 second using a computer with Intel Pentium 2.6GHz. Finally, Host-VM Configuration manager applies these configurations to the host and its VMs after calculating suitable weight values for each VM. Then, it informs the dispatcher that the VMs is ready now to execute these jobs. When considering two or more of VMs the provisioned resources will be divided according to the weight setting of VM which is proportional to the workload demand of each job.

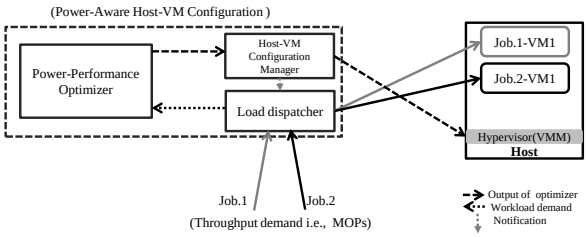

Figure 3: System architecture overview.

4. DYNAMIC OPTIMIZATION CONFIGURATION

Multithreaded jobs could exploit the number of cores in a processor to reduce execution time. A VM optimization problem that we consider is to determine the most efficient power Host-VM configuration to handle throughput demand. To realize fine-grained power provisioning, the optimization problem could produce heterogeneous settings of cores' frequency. For instance, a VM could be scheduled in two physicals cores with frequency 1.6GHz, and the other two could be scheduled on physical cores with frequency 2.2GHz. Furthermore, the need of VMs' pre-configuration is because of the characteristic of these jobs. In other words, when a job starts with a specific number of threads, the job afterwards cannot be enforced to increase or decrease the number of its threads.

4.1 Optimization problem formulation

We introduce the following notation to formulate our optimization problem. We assume that a host has C of cores. These cores are homogeneous cores in terms of clock frequency. Each core c runs on frequency F_c. The frequency level $f \in F_c$ ranges between 1.6GHz and 2.4GHz. This server could host multiple jobs J. Each job $j \in J$ runs on a VM. The parameter D_j represents the workload demand which guarantees the job j to finish before the deadline. The binary decision matrix is defined by $conf_{(c,f)}$ to denote whether the core c has been selected to run on frequency f to handle the total workload of jobs J. The matrix rows represent the number of cores meanwhile columns represent the frequency levels. For instance, $conf_{(1,1)} = 1$ means that the core 1 has been selected to run on frequency level 1. In fact, frequency level 1 gives the slowest clock frequency and the lowest performance. The provisioned capacity $cap_{(c,f)}$ is the capacity of the host when it runs on core c with frequency f. These settings also will provide an amount of power $pow_{(c,f)}$. Thus, the optimization problem is represented by the following mixed integer program (MIP):

$$Minimize:$$

$$\sum_{c \in C} \sum_{f \in F_c} conf_{(c,f)} * pow_{(c,f)} \quad (1)$$

$$Subject\ to:$$

$$\sum_{j \in J} D_j \leq \sum_{c \in C} \sum_{f \in F_c} cap_{(c,f)} * conf_{(c,f)} \quad \forall c \in C, \forall f \in F_c \quad (2)$$

$$\sum_{f \in F_c} conf_{(c,f)} \leq 1 \quad \forall c \in C \quad (3)$$

The objective function given by Equation 1 is to find a Host-VM configuration that minimizes power consumption. As observed experimentally via the model of power in Section 2.3, the power consumed by a server grows linearly with the number of cores at full utilization for the same given CPU frequency. Equation 2 prevents a possible solution in which the demand of all running jobs J exceeding the total capacity of the server at specific configuration. Equation 3 guarantees that only one frequency $f \in F_c$ is assigned to a given core c. For example, the solver could return a solution of a configuration matrix as presented in Figure 4. The matrix of $conf_1$ means that the server has 8 active cores and each VM has 8 vCPUs. All cores run on frequency level 1 (i.e., 1.6Ghz). The second configuration matrix $conf_2$ has heterogeneous configuration where it has two cores run on frequency level 2 (i.e., 1.72Ghz) and the others run on 1.6Ghz. Nevertheless, in case of heterogeneity, CPU affinity is used to guarantee that vCPUs are scheduled on the suitable cores. Indeed, the total capacity provided by these configurations is divided among the hosted VMs according to their demand using the proportionality weight of credit-scheduler. Furthermore, these configurations guarantee the minimal power consumption of the server to execute the jobs.

4.2 VM-weight proportionality to demand

As we mentioned in the previous section that the server could host multiple jobs at a specific time, we need a mechanism to distribute the provisioned resources among the

$$conf_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$conf_2 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 4: Example of two different host's configurations.

running VMs which run the jobs. In this case, Host-VM configuration manager is responsible to set up the suitable weight for each VM to satisfy the assigned job demand. In credit-scheduler of Xen, a VM's weight is a proportional value to divide resources particularly CPU cycles among VMs. Thus, if we set a VM's weight to be proportional to its job demand, we guarantee the fairness share. For instance, two VMs hosts two jobs; job-1 demands 150 MOPs and runs on VM-1 meanwhile job-2 demands 100 MOPs and runs on VM-2. In this scenario, Host-VM configuration manager will set up the weights 3 and 2 for VM-1 and VM-2 respectively. The legal weights range from 1 to 65535 and the default is 256. However, the Host-VM Configuration manager will calculate the suitable Weight of each VM (WVM) using Equation 4.

$$WVM_j = D_j / GCD(D_{[1,J]}) \quad (4)$$

D_j is the demand of job j; $GCD(D_{[1,J]})$ represents Greatest Common Divisor of the jobs' demand. Table 1 presents an example of the calculated weight of each VM according to job demand. The third row of this table presents weight of the previous example. The weights of WVM_1 and WVM_2 are 3 and the weight of the third VM is 2.

Table 1: An example of calculated weight for three VMs with different job demand.

D_1	D_2	D_3	WVM_1	WVM_2	WVM_3
10	12	20	5	6	10
100	100	80	5	5	4
150	150	100	3	3	2

5. PERFORMANCE EVALUATION

In this section, we present two experimental setups to evaluate our proposed solution. First, we show how our solution could achieve power proportionality to workload. Second, we present dynamic changes of VMs configuration to satisfy each VM requirement with efficient power consumption.

5.1 Power proportionality to workload

To show the performance of our solution compared to the baseline power management governors of hypervisor (i.e., Performance-governor), we generate several jobs with a different pattern of demand. Then, by using optimization model, Host-VM Configuration manager configures both of the host and its VMs to suitable configuration that minimize the total power to execute these jobs. In this experiment, we simulated the total demand of a host to achieve the jobs' demand at that time. The trace of the host's throughput shown in Figure 5. The x-axis of Figure 5 represents the configuration index that was obtained by optimizer when the workload state was changed. For instance, index 1 represents the state of the total required jobs' demand which was 147 MOPs at that time. Then, this demand increased at index 4 to become 163 MOPs; a new configuration was generated. Finally, the demand reached its maximum at index 19. The throughput of the host increased due to the increase of jobs' demand. Afterward, the demand ramped down after index 21. Importantly, the solution of this optimization model could result of heterogeneous active cores in terms of frequency. The heterogeneity of cores can assist in achievement of power-proportionality. To achieve fairness, we configured each VM with a number of vCPU, which was suggested by the optimizer. Then, these vCPU were mapped to the physical cores set. For example, at indices 1-3 the optimizer gave a solution with 4 active cores all of them run at the lowest frequency and 4 vCPU for each VM. After this, the optimizer increased number of active cores and frequency according to workload intensity taking into account minimization of power. Figure 5 depicts a big difference between the provisioned power using the performance-governor of hypervisor and our proposed solution of dynamic configuration. However, using the performance-governor caused the gap between the provisioned power and the demand power when executing specific jobs. This gap emerged because of the fixed number of VM at each workload and setting frequency at its maximum. Clearly, this gap was relatively large at low demands, and it vanished when the workload reached its maximum at index 19. Indeed, with dynamic configuration of both host and its VMs, we can realize a fine-grained power proportionality to workload.

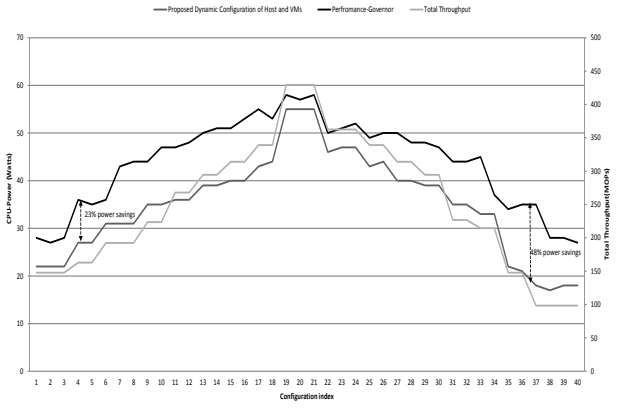

Figure 5: Dynamic Host-VM Configuration: provisioning power proportionality to workload demand.

5.2 VM-weight proportionality to workload

This section presents a dynamic configuration of VM weight to make each VM's capacity proportionality to workload demand of its hosted job. As in previous experiment, we generate workload with changing intensity over time. In this experiment, we assumed that the jobs' should achieve a certain amount of throughput to finish within its deadline. The jobs' demands were changing according to the submitted job size. Then, we recorded the output of Host-VM configuration manager and the achieved the throughput of each job. The trace of the dynamic change of the weight of VM proportionality to workload is presented in Figure 6 and 7. The calculation of the weight of each VM was discussed in Section 4.2. Figure 7 depicts the achieved throughput for each job and the total of jobs demand. In short, we achieved the fairness among VMs and satisfied each VM's demand with more efficient power provisioning.

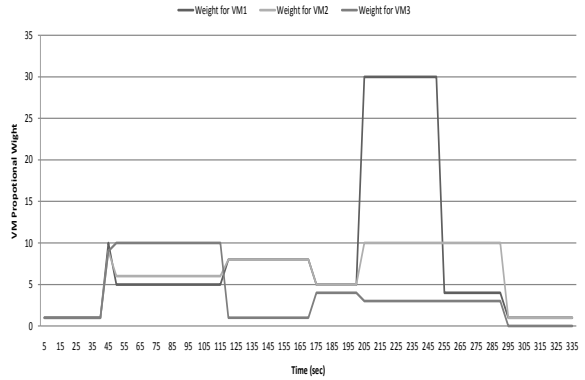

Figure 6: Dynamic settings of the weight of VM proportionality to workload demand.

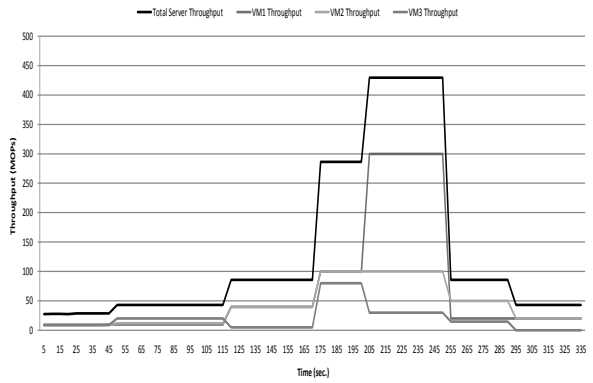

Figure 7: The achieved throughput of each VM proportionality to its weight.

6. CONCLUSIONS AND FUTURE WORK

We present an optimization solution to achieve power proportionality for virtualized servers. Our solution enables a dynamic configuration of host and its VMs as well to satisfy workload demand while minimizing power consumption.

The optimization model could provide a set of heterogeneous cores in terms of frequency. This realizes a fine-grained power provisioning compared to homogeneous cores. We implemented a prototype of proposed solution. Then, we conducted experiments to show its effectiveness in power savings compared to hypervisor performance governor. The results showed that our solution could achieve power savings between 23% to 48% compared to the performance-governor as baseline. As a future work, we will integrate and evaluate our solution into a virtualized cluster environment to achieve a global power optimization by distributing jobs into hosts that can execute the jobs and maintain the jobs' demand with the lowest possible power. Finally, we will consider other virtualization technologies such as VMware.

7. REFERENCES

[1] J. Hamilton. Cooperative expendable micro-slice servers (CEMS): low cost, low power servers for internet-scale services. *In conference on Innovative Data Systems Research CIDR'09*, January 2009.

[2] U.S. Environmental Protection Agency. Epa report on server and data center energy efficiency. ENERGY STAR Program, 2007.

[3] L. A. Barroso and U. HÃűlzle. The Case for Energy-proportionality Computing. *Computer*, vol. 40, pp. 33-37, Dec. 2007.

[4] Y. Ben-Itzhak, I. Cidon, and A. Kolodny. Performance and power aware cmp thread allocation modeling. *In HiPEAC*, 2010, pp. 232-46.

[5] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu. No power struggles: coordinated multi-level power management for the data center. *In Proceedings of ASPLOS*, March 2008.

[6] R. Urgaonkar, U.C. Kozat, K. Igarashi, and M.J. Neely. Dynamic resource allocation and power management in virtualized data centers. *2010 IEEE Network Operations and Management Symposium (NOMS), IEEE,* 2010, pp. 479-486.

[7] L. Bertini, J.C.B. Leite, and D. Mosse. Statistical QoS Guarantee and Energy-Efficiency in Web Server Clusters. *In Proceedings of the 19th Euromicro Conference on Real-Time Systems. IEEE Computer Society*, 2007, pp. 83-92.

[8] J.S. Chase, D.C. Anderson, P.N. Thakar, A.M. Vahdat, and R.P. Doyle. Managing energy and server resources in hosting centers. *In Proceedings of the eighteenth ACM symposium on Operating systems principles*, Banff, Alberta, Canada: ACM, 2001, pp. 103-116.

[9] E. N. Elnozahy, M. Kistler, and R. Rajamony. Energy-efficient server clusters. *In Power-Aware Computer Systems, volume 2325 of Lecture Notes in Computer Science*, pp. 179-197, 2003. ISBN 978-3-540-01028-9.

[10] B. Khargharia, S. Hariri, and M.S. Yousif. Autonomic power and performance management for computing systems. *Cluster Computing*, vol. 11, Jun. 2008, pp. 167-181.

[11] D. Kusic and N. Kandasamy. Power and performance management of virtualized computing environments via lookahead control. *In Proceedings of ICAC*, June 2009.

[12] X. Wang, D. Lan, X. Fang, M. Ye, and Y. Chen. A resource management framework for multi-tier service delivery in autonomic virtualized environments. *In Proceedings of NOMS*, April 2008.

[13] S. Govindan, J. Choi, B. Urgaonkar, A. Sivasubramaniam, and A. Baldini. Statistical profiling-based techniques for effective power provisioning in data centers. *In Proceedings of EuroSys*, April 2009.

[14] A. Krioukov, P. Mohan, S. Alspaugh, L. Keys, D. Culler, and R. H. Katz. NapSAC: design and implementation of a power-proportionality web cluster. *ACM SIGCOMM Computer Communication Review*, vol. 41, 2011, pp.102-108.

[15] V. Petrucci, O. Loques, and D. Mosse. Dynamic optimization of power and performance for virtualized server clusters. *In Proceedings of the 2010 ACM Symposium on Applied Computing*, 2010, pp. 263-264.

[16] Shekhar Srikantaiah et al. Energy aware consolidation for cloud computing. In USENIX Workshop on Power Aware Computing and Systems, 2008.

[17] M. Raboso, L. del Val, M. Jiménez, A. Izquierdo, J. Villacorta, and J. de la Varga.Virtualizing Grid Computing Infrastructures into the Cloud. *International Symposium on Distributed Computing and Artificial Intelligence*. Springer 2011, pp.159-166.

[18] L. Cherkasova, D. Gupta, E. Ryabinkin, R. Kurakin, V. Dobretsov, and A. Vahdat. Optimizing grid site manager performance with virtual machines. *In Proceedings of the 3rd conference on USENIX Workshop on Real*, Large Distributed Systems-Volume 3, 2006, pp.5-5.

[19] R. V. der Wijngaart, NAS Parallel Benchmarks v. 2.4.NAS Technical Report NAS-02-007, October 2002.

[20] Xen power management. http://wiki.xensource.com/xenwiki/xenpm. Accessed 30-6-2011.

[21] I. Takouna, W. Dawoud, and C. Meinel. Efficient Virtual Machine Scheduling-policy for Virtualized Heterogeneous Multicore Systems. *In Proceedings of the 2011 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2011)*, July 2011.

[22] IBM ILOG CPLEX Optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/ Accessed 30-6-2011.