A Dynamic Power Management Controller for Optimizing Servers' Energy Consumption in Service Centers

Tudor Cioara[1], Ioan Salomie[1], Ionut Anghel[1], Iulian Chira[1], Alexandru Cocian[1], Ealan Henis[2], and Ronen Kat[2]

[1] Technical University of Cluj-Napoca, Computer Science Department, Cluj-Napoca, Romania
{Tudor.Cioara,Ioan.Salomie,Ionut.Anghel}cs.utcluj.ro
[2] IBM Haifa Research Laboratory, Haifa, Israel
{Ealan,Ronenkat}@il.ibm.com

Abstract. This paper proposes the development of a management controller, which balances the service center servers' workload and hardware resources usage to locally optimize energy consumption. The controller exploits energy saving opportunities due to short-term fluctuations in the performance request levels of the server running tasks. The paper proposes Dynamic Power Management strategies for processor and hard disks which represent the main elements of the controller energy consumption optimization process. We propose techniques for identifying the over-provisioned resources and putting them into low-power states until there is a prediction for a workload requiring scaling-up the server's computing capacity. Virtualization techniques are used for a uniform and dependence free management of server tasks.

Keywords: Dynamic Power Management, Service Center Server, Energy Efficiency.

1 Introduction and Related Work

The past decade in the field of computing has been marked by the advent of web-based applications and online services, which translated into an exponential growth of the underlying service center infrastructure. A 2007 report [1] to Congress, by the U.S. Environmental Protection Agency, shows that in just five years, the electricity consumed by service centers and their additional infrastructure will be doubled and the trend is expected to accelerate driven by the shift towards cloud computing.

The GAMES (Green Active Management of Energy in IT Service Centers) [2] research project aims at developing a set of innovative methodologies, metrics, services and tools for the active management of energy efficiency of IT service centers. In the GAMES project we approach the service center energy efficiency problem by using two types of energy aware control loops: a global control loop associated with the entire service center and a set of local control loops associated with each service

E.M. Maximilien et al. (Eds): ICSOC 2010 Workshops, LNCS 6568, pp. 158–168, 2011.

center server. The global control loop uses service center energy/ performance data for taking run-time adaptation decisions to enforce the predefined Green and Key Performance Indicators (GPIs / KPIs). The local control loops balance the service center servers' workload and hardware resources usage to locally optimize the servers' energy consumption, without considering the entire service center state.

In this paper we propose the development of an autonomic management controller, which targets the objectives of the local control loop, for a virtualized server of a service center. The controller, also called the Local Loop Controller, manages virtualized tasks at the server level so that the performance levels required by the tasks running on the server are achieved and maintained while maximizing the server energy efficiency. The controller exploits energy saving opportunities presented by short-term fluctuations in the performance request levels of the server running tasks. We use virtualization as an abstraction level, because it allows us to manage the tasks in a uniform manner, without worrying about application dependencies and low-level details. The virtualized tasks are annotated with Quality-of-Service (QoS) requirements representing the tasks' performance request levels. Local Loop Controller energy consumption optimization process is based on Dynamic Power Management (DPM) actions aiming at putting the over-provisioned resources into low-power states. The resources remain in low-power states until there is a predictive workload that requires scaling-up the server's computing capacity.

The service center servers' energy efficiency is a complex issue and existing research covers a wide variety of techniques which have to be taken in a holistic approach. It involves concepts such as resource provisioning and allocation, virtualization and dynamic power management. Khargharia [3] proposes a theoretical methodology and an experimental framework for autonomic power and performance management of high-performance server platforms. The over-provisioned server resources are transitioned to low-power states until there is an increase in the workload that would require scaling-up the server capacity again. Similarly, in [4] the performance-power management problem is decomposed into smaller sub-problems implemented locally by individual controllers within each hardware component. A fuzzy resource allocation approach where a fuzzy logic controller is used to dynamically adjust the number of resources needed in serving requests is presented in [5]. The main problem with DPM techniques is that changing between power states has performance implications, and sometimes energy penalties [6]. Algorithms have been developed for DPM by considering the power/performance characteristics of main hardware devices, mainly the processor and the hard disk drive and, to some extent, the system memory (mostly hardware controllers) [7]. Bircher and John [8] perform an ample analysis of power management on recent multi-core processors using the functions provided by regular operating systems. For hard drive power management Chung [9] proposes an active-idle states transition method based on adaptive learning tree in which idle and active periods are stored as tree nodes. In [10], [11] and [14] power-aware storage cache management algorithms are presented. They also provide opportunities for the underlying disk power management schemes to save energy, such as off-line greedy algorithms and on-line cache write policies. The use of virtualization together with DPM techniques raises serious challenges due the lack of power metering for virtual machines. Unlike physical servers that provide in-hardware or outlet level power measurement functions, it is very difficult to estimate the power

required by a virtual machine when hosted on a certain hardware configuration [12]. The performance and health monitoring counters exposed by the virtual machine hypervisor, correlated with relevant power information about the hosting hardware and simple adaptive power models can offer relatively accurate information [13].

The rest of this paper is organized as follows: Section 2 presents the architecture of the Local Loop Controller, Section 3 describes the proposed DPM algorithms, Section 4 shows a the experimental results, while Section 5 concludes the paper.

2 Local Loop Controller Architecture

The Local Loop Controller acts at the server level with the goal of maximizing server energy efficiency while preserving the performance levels required by the running tasks. From a power management perspective, we emphasize that the most power-hungry components within a server are the CPU, the HDD (Hard Disk Drive) and, to some extent, the internal memory. The Local Loop Controller exploits the non-mutually exclusive energy / performance related behavior of these components and sets them to low-power states. The workload is composed of virtualized tasks annotated with Quality-of-Service (QoS) requirements. Fig. 1 shows the general architecture of the Local Loop Controller. The following subsections present the role and responsibility of each architectural module, together with their interactions.

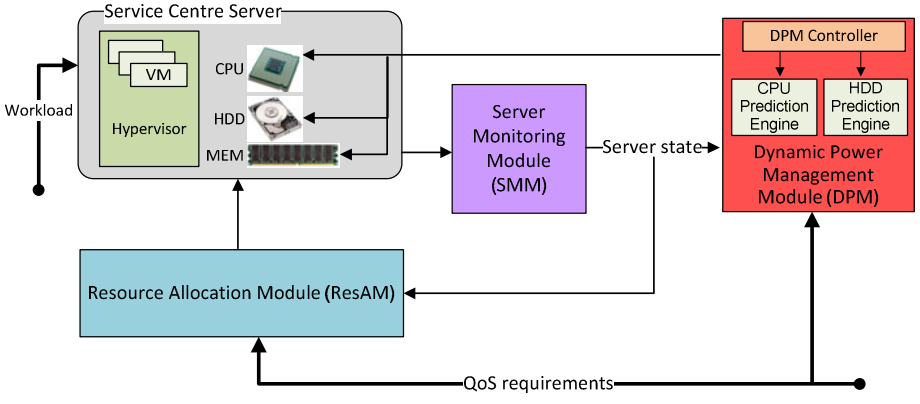

Fig. 1. Local Loop Controller architecture

Resource Allocation Module (ResAM module). The purpose of the ResAM module is to allocate the hardware resources for the virtual machines that are running on the server. The ResAM module considers the performance implications of virtualization and tasks QoS requests, to design a resource allocation strategy which meets performance requests while assuring high energy efficiency.

When dealing with *CPU resource provisioning* the following aspects are considered for a virtual machine: (i) the number of logical processors (i.e. the number of virtual processors to be assigned to the virtual machine); (ii) CPU reservation, representing the minimum percentage of resources that are guaranteed to a specific

virtual machine and (iii) CPU upper limit, representing the maximum percentage of resources that a specific virtual machine is able to use. Another important aspect for the CPU provisioning is to set processor's affinity to a virtual machine, taking into account whether the processor frequency can be scaled per socket or per core. We have defined two CPU resource provisioning strategies: (i) one that tries to distribute as many tasks as possible per core (allowing the unused cores to switch to low power) and per processor (allowing the unused processors to stay in idle as much as possible) and (ii) one that evenly distributes tasks to get a lower overall utilization and thus enable the transition of the entire core/socket to inferior frequencies.

When dealing with the *HDD resource provisioning,* the locality and performance aspects are considered. In a server system having several disk drives used as a cumulative storage space, we distribute the virtual machines as locally as possible. We consider that the virtual machines use a simple file as hard-drive and we can allocate that file on any physical disk for having more drives in spin-down mode as long as possible. This way, all the disks will be accessed, some of them being more active while others having longer idle periods, allowing thus for the dynamic power management controller to spin them down more often and save energy. The major bottleneck in virtual machine performance is the I/O sub-system of the host. Hard disk drives suffer from serious performance degradation when it comes to multi-tasking since most I/O controllers serve request on a first-come, first-serve basis. Therefore, running a large number of virtual machines on a single hard drive, although it has enough storage space, will ultimately result in unacceptable performance penalties. Each virtual disk file has an associated Performance Impact (PI), ranging from 0 to 100. As a result, the cumulative PIs for the virtual disk files stored on a physical drive cannot exceed 100.

The *MEM resource provisioning* is a rather simple process. The virtual machine has a memory requirement and the hypervisor meets it by allocating the specified amount of memory to the virtual machine. The MEM allocation strategy is based on a modified version of the dynamic memory allocation principle. Consider the situation in which we detect that the internal memory allocated to the machine is not enough and that the virtualized guest OS is frequently accessing the swap memory, which in turn translates into hard-drive accesses for the physical host system. If the physical system has unused internal memory, it can supplement the memory size for the virtual machine. This rudimentary caching mechanism will provide the Local Loop Controller with more opportunities to spin-down the hard drive. To reconfigure without powering down the virtual machines, the guest operating system should be able to handle memory as a hot-plug device, i.e. when new memory is added, the gust OS must recognize and use it.

Server Monitoring Module (SMM module). The roles of the SMM module (see Fig. 2) is to continuously analyze the state of the host service center server and of the guest virtual machines and provide the relevant performance and energy consumption data to the ResAM and DPM modules. The ResAM module uses the data about the usage degree of the hardware resources of the host server to decide on an appropriate assignment of virtual machines, while the DPM module uses the data about the host server hardware resources current usage and workload to evaluate the opportunity for a power state transition. For a server, two different aspects are monitored: (i) the

workload generated by the programs which are directly executed on the physical system and monitored by the operating system (such as the operating system, the device drives associated with physical components and the Local Loop Controller itself) and (ii) the server total workload which includes in addition the workload generated by the hosted virtual machines, both in terms of CPU cycles and the amount of data read from / written to the physical hard drives; the virtual machines residing on a server are monitored by means of the virtualization software (hypervisor).

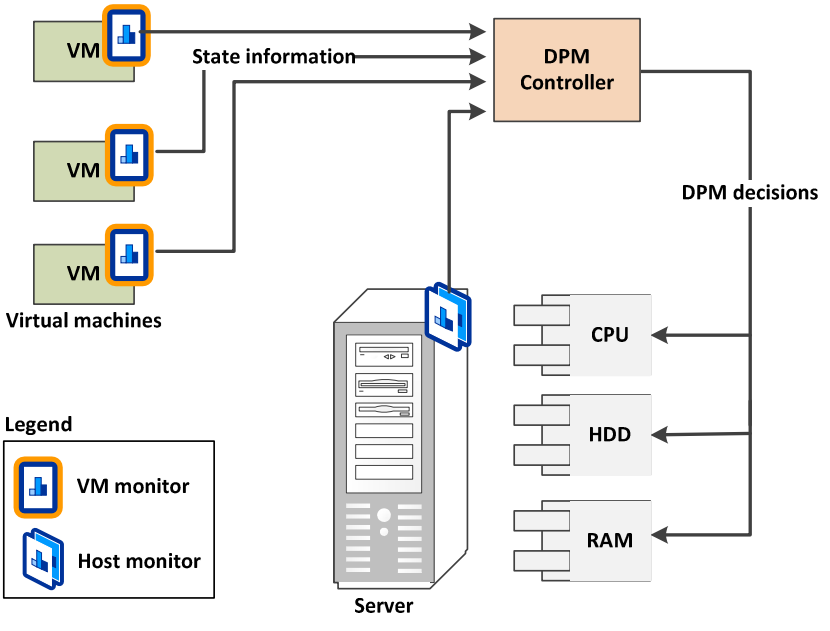

Fig. 2. Server Monitoring Module architecture

Dynamic Power Management (DPM) module. The DPM module uses the data provided by the SMM module, analyzes the possibility of optimizing the server energy efficiency and decides on two types of power management actions: (1) processor power adaptations actions – the processor's frequency is scaled up or down, by means of P-state management, consequently modifying its performance capacity and the amount of energy it requires and (2) hard disk sleep transitions – when the controller detects an idle period longer than the HDD break-even time and decides to put the HDD to sleep state thus reducing its energy consumption. For each of the processor and the hard disk drives, there is an actuator that enforces power management decisions and a prediction engine component that records relevant patterns in workload fluctuations and determines (probable) future idle periods (see Fig. 3). For CPU predictions, a Fuzzy Controller is used, while for HDD a specific idle period prediction engine is used. The power state transition decision for a CPU or HDD must take into account all the virtual machines stored on the host server. A correlation engine analyzes the individual pattern predictions and outputs a global prediction. Regarding the CPU, the correlation engine has to select an appropriate P-state such that the

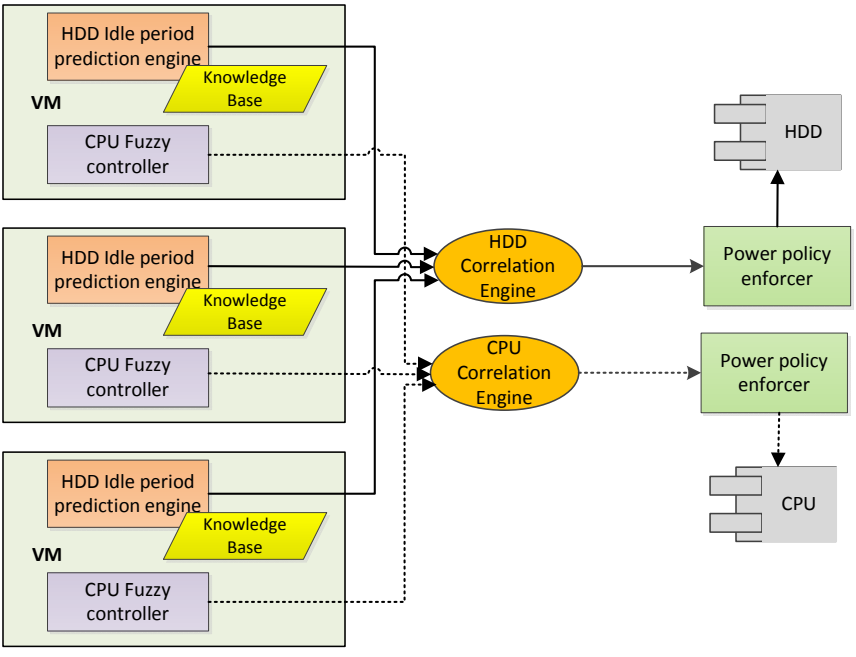

Fig. 3. The DPM Controller

processor-related QoS requirements are satisfied for all the virtual machines under the current workload. A HDD drive can be powered down only if no load is expected in the near future for all virtual machines that use it. As a result, the correlation engine has to check whether all virtual machine associated predictors have issued long idle periods which have not been revoked.

3 Dynamic Power Management (DPM) Strategies

We have defined customized DPM strategies/algorithms for the processor and HDD.

The *power management algorithm for processor* is based on adaptively changing the processor P-states (defining the processor performance levels) taking into account the incoming workload. The algorithm must filter the situations in which the workload fluctuates for short periods of time because the transition cost (in terms consumed energy) can outweigh the benefit of the adaptation. Therefore, if the CPU is currently in a low power state and the load exhibits an isolated spike, the CPU must not enter full mode, because even if there will be a delay in servicing the requests of the load spike, it is tolerable in the context of the overall computational throughput. Given the above considerations, we have decided to design and implement a processor power management algorithm based on fuzzy controller. The approach is somewhat similar to that proposed in [5], however instead of using it to globally manage the service center sizing according to workload, we employ it to control the processor's capacity. The advantage of fuzzy-logic is the ability to filter-out noise and to

adapt progressively to changes. The fuzzy controller has two input variables: the workload represented by the instructions that the processor must execute, and the processor usage ratio. For each of the processor power states, we define three fuzzy sets, LOW, MEDIUM and HIGH, represented by f_l, f_m and f_h functions. After each time window, the fuzzy controller (represented by f_c function) evaluates the f_l, f_m and f_h functions for the current load and updates f_c value according to following equation:

$$f_c = f_c + \propto f_h(load) - \beta f_l(load) + \gamma f_m(load) . \quad (1)$$

When the fuzzy controller reaches the values 0 or 1, the CPU is transitioned to an inferior or a superior power state as appropriate (if there is one available) and its f_c value is reset to the default value of 0.5. By varying α, β and γ, we can control the compromise between energy efficiency and performance degradation: a large α or β value means that the load spikes will not be filtered, while small α and β induce delays that lead to performance degradation and thus increasing energy efficiency.

The *power management algorithm for the hard disk* identifies the hard disk drive access patterns and decides if it is possible to spin-down the drive. Unlike the processor, which is very flexible when it comes to power management, the hard disk state transitions are more rigid and involve significant performance degradation when the power management decision is not adequately performed. For a HDD we identify three distinct power states: (1) *active (Read*, *Write*, *ReadWrite)* - the device is busy, spinning at maximum speed, (2) *standby (Idle)* - the device is not used and it spins at low speed and (3) *sleep* - the device is completely shut down.

The proposed power management algorithm for the HDD, transforms the hard disk access sequences followed by idle periods into discrete quantifiable events and

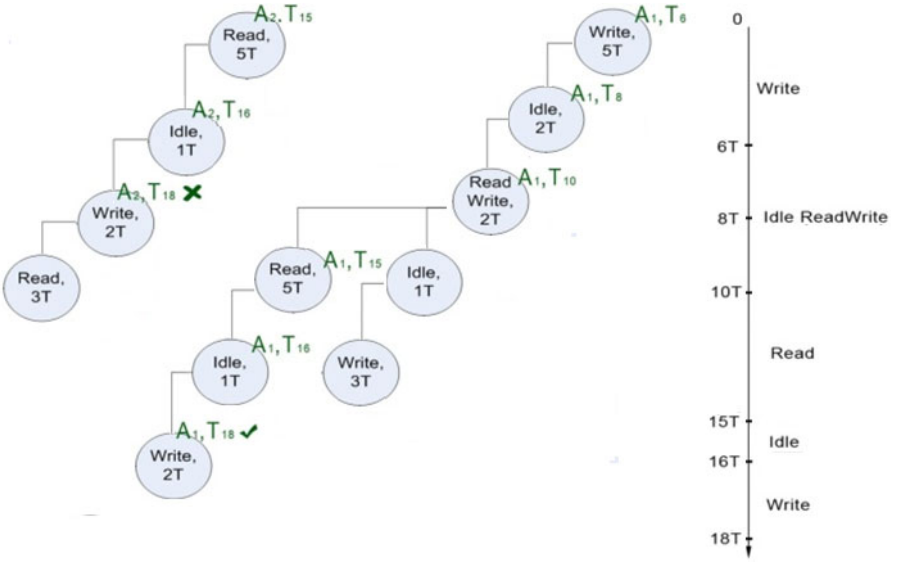

Fig. 4. The development of an adaptive learning tree for identifying HDD idle periods

stores them as nodes in a tree data structure (agent oriented adaptive learning tree [9]). To obtain more accurate results we use an instance of the algorithm for each virtual machine and we additionally define a prediction synchronization phase to determine the opportunity of disk spin down. Every time an event is triggered (state change), a new agent is created in the root of the adaptive learning tree and on every state change the agent will try to advance in the tree. When an agent reaches a leaf, it means that the current sequence found in the adaptive learning tree is supposed to be followed by an idle period. A reward/ penalty approach is used to mark the credibility of the sequence (misprediction – penalty, correct prediction – reward). Fig. 4 shows a possible pattern in the adaptive learning tree knowledge base where T is the length of a monitoring sample period. At t = 6T, a transition is detected (Write->Idle), so an agent (A1) is created that tries to navigate through the learning tree, knowing that the previous state was "Write for 5T". When the next transition is detected (Idle->ReadWrite), A1 advances in the tree, if the current location contains a child annotated with " ReadWrite for 2T". The agent will continue to explore the adaptive learning tree based on state transition events until it reaches a leaf and signals that a known sequence has been detected and the hard drive can be put in *standby* state.

4 Case Study and Results

The test server used to validate the Local Loop Controller power management algorithms is an IBM/PC computer having an Intel Core 2 CPU E6600@2.40GHz (2 cores, 2 logical processors), 2.00 GB Physical Memory (RAM) DDR2@800Mhz and two hard disk drives - 250GB@7200rpm and 320GB@7200rpm. The HDD's operate on the SATA 3 GB/s interface, offering a sustained transfer rate of 126MB/s. The processor provides hardware assisted virtualization functions in the form of Intel Virtualization Technology (Intel-VT) which allows running of Type 1 (native) hypervisors. Two P-states are provided, one at 1.60 GHz (66.67% of standard frequency) and one at 2.40 GHz (100% of standard frequency). The hard disk drives present an active state when they run at full speed, standby states when they spin down and a sleep state when they are completely powered off.

The server underlying operating system is Windows 2008R2 X64. The current implementation use Hyper-V R2 for virtualization support. Hyper-V R2 is a Type 1, hypervisor, providing good performance, flexibility for configuring virtual machines and live migration features. For system monitoring, we use WMI (Windows Management Instrumentation) implemented in the OS and in the hypervisor.

For testing the dynamic power management algorithms, we have defined a test case scenario based on two virtual machines. The first virtual machine (VM1) is configured as having 1GB of RAM and 2 virtual processors. The virtual machine runs an instance of Microsoft SharePoint Server 2007 and an ASP.NET web based application, heavily relying on SQL Server 2005 for data storage. From three computers connected to the same intranet we simulate 15 independent users accessing the application and performing common tasks (such as web pages access, document creating, list management) requiring medium CPU usage and a relatively heavy I/O workload. The second virtual machine (VM2) is configured with 512MB of RAM and one virtual processor and runs a processor intensive application that periodically drives the

CPU to 100% usage level, followed by a lower processor usage. From an I/O perspective we simulate a random light workload.

Since the two virtual machines run on the system, the Local Loop Controller monitors the workload level and takes DPM decisions accordingly. Fig. 5 shows the fuzzy algorithm behavior on a complex workload. As pointed by the green markers in the graph, the algorithm is able to filter random request spikes in the workload of the virtualized applications. The yellow markers show that there is a slight delay between the workload fluctuation and the P-state transition.

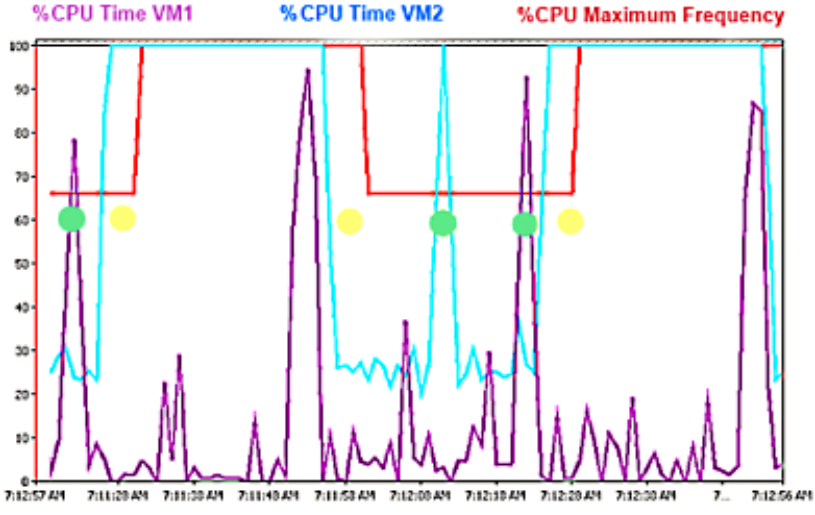

Fig. 5. Results of fuzzy logic based power management algorithm for processor

To fully evaluate the advantages brought by the proposed power management algorithms, the processor P-state transition results were compared to the results obtained by using the Balanced Power Scheme implemented in Windows Server 2008R2. As illustrated in Fig. 6, the Balanced Power Scheme processor transitions are more aggressive, and even isolated request spikes determine P-state changes. This shows that, the operating system's emphasis is on performance and not on energy efficiency and also that the power management algorithm is fairly rudimentary. Using our processor power management algorithm, the processor spends 39% of the time in low power state, while using the Balanced Power Scheme it is at low frequency for 23% of time (for the same workload).

Regarding the Balanced Power Scheme for HDD, Windows never spins down the drives; it does allow setting a time-out, but not less than 1 minute. Considering a light I/O workload, during a one hour test using the Balanced Power Scheme and with the spin-down time-out set at the minimum allowed of 1 minute, the hard drive was in idle for 312 seconds (~5.2 minutes). On the other hand using the same test conditions, the Local Loop Controller has transitioned the drive to idle for a total 1447 seconds (~24.5 minutes), resulting an increase of the hard disk idle period with 470%.

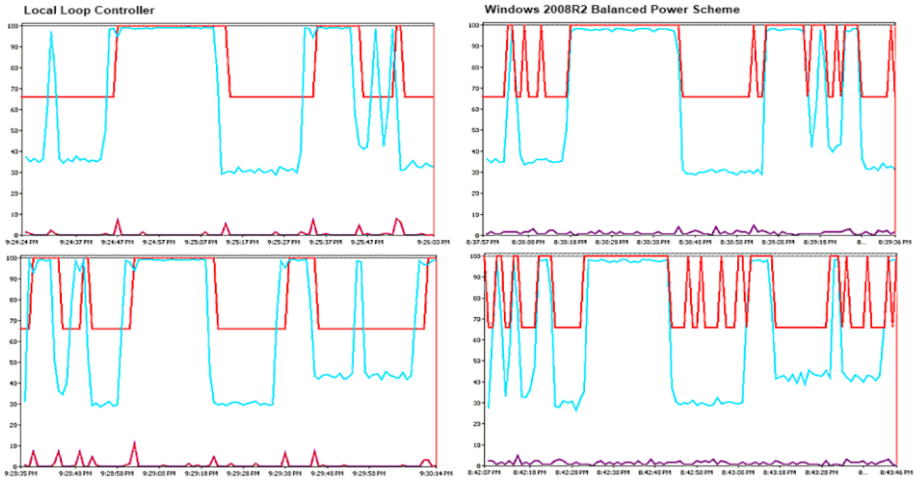

Fig. 6. P-state transition results: Local Loop Controller vs. Windows Balanced Power Scheme

Considering the hardware configuration of the test system, the hardware manufacturers technical data and the processor/hard disks idle periods we can estimate the energy consumption for the system at different load levels. In our test conditions, we estimate an energy consumption of 245Wh (watt hour) for the Balanced Power Scheme and of 230 Wh for our Local Loop Controller. This results in an estimated energy savings of around 15 Wh. Using the time to execute the workload as performance factor we estimate a 4% performance degradation when our Local Loop Controller algorithms are used, compared to the OS default power scheme.

5 Conclusions

This paper proposes the development of a management controller, which balances the service center servers' workload and hardware resource usage to locally optimize the servers' energy consumption. The results are promising showing that using the proposed dynamic power management strategies the hard disk idle periods increase with approximately 470% compared with Windows Balanced Power Scheme. Also an improvement of 16% of the processor low power state periods is obtained, when using the Local Loop Controller instead of the Balanced Power Scheme.

Acknowledgments

This work has been supported by the European Commission within the GAMES project [2] funded under EU FP7.

References

1. U.S. Environmental Protection Agency, ENERGY STAR Program, Report to Congress on Server and Data Center Energy Efficiency, Public Law 109-431 (2007)
2. GAMES Research Project, `http://www.green-datacenters.eu/`

3. Khargharia, B., Hariri, S., Yousif, M.: Autonomic power and performance management for computing systems. Cluster Computing 11(2), 167–181 (2008) ISSN:1386-7857
4. Wang, N., Kandasamy, N., Guez, A.: Distributed Cooperative Control for Adaptive Performance Management. IEEE Internet Computing 11(1), 31–39 (2007)
5. Jeske, J.C., Julia, S., Valette, R.: Fuzzy Continuous Resource Allocation Mechanisms in Workflow Management Systems. In: IEEE International Conference on Information Reuse and Integration, pp. 472–477 (2006) ISBN: 0-7803-9788-6
6. Minerick, R., Freeh, V., Kogge, P.: Dynamic Power Management using Feedback. In: Proceedings of Workshop on Compilers and Operating Systems for Low Power (2002)
7. Gupta, R., Irani, S., Shukla, S.: Formal Methods for Dynamic Power Management. In: IEEE/ACM International Conference on Computer-Aided Design, p. 874 (2003)
8. Bircher, L., John, L.: Analysis of Dynamic Power Management on Multi-CoreProcessors. In: Proc. of the 22nd Annual International Conference on Supercomputing, pp. 327–338 (2008)
9. Chung, E., Benini, L., De Micheli, G.: Dynamic Power Management Using Adaptive Learning Tree. In: IEEE/ACM Int. Conf. on Computer-Aided Design, pp. 274–279 (1999)
10. Bisson, T., Brandt, S., Long, D.: NVCache: Increasing the effectiveness of disk spin-down algorithms with caching. In: Proceedings of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation, pp. 422–432 (2006)
11. Zhu, Q., David, F., Devaraj, C., et al.: Reducing Energy Consumption of Disk Storage Using Power-Aware Cache Management. In: Proceedings of the 10th International Symposium on High Performance Computer Architecture, p. 118 (2004)
12. Stoess, J., Lang, C., Bellosa, L.: Energy Management for Hypervisor-Based Virtual Machines. In: USENIX Annual Technical Conference (2007)
13. Kansal, A., Zhao, F., Liu, J., et al.: Virtual Machine Power Metering and Provisioning. In: The 1st ACM Symposium on Cloud Computing, pp. 39–50 (2010) ISBN:978-1-4503-0036-0
14. Colarelli, D., Grunwald, D.: Massive arrays of idle disks for storage archives. In: Proceedings of the ACM/IEEE Conference on Supercomputing, pp. 1–11 (2002)