CHAPTER 1

ENERGY EFFICIENCY IN HPC SYSTEMS

IVAN RODERO[1,2] AND MANISH PARASHAR[1,2]

[1]NSF Center for Autonomic Computing, Rutgers University, New Jersey, USA
[2]Department of Electrical and Computer Engineering, Rutgers University, New Jersey, USA

1.1 INTRODUCTION

Power consumption of High Performance Computing (HPC) platforms is becoming a major concern for a number of reasons including cost, reliability, energy conservation, and environmental impact. High-end HPC systems today consume several megawatts of power, enough to power small towns, and are in fact, soon approaching the limits of the power available to them. For example, the Cray XT_5 Jaugar supercomputer at Oak Ridge National Laboratory (ORNL) with 182,000 processing cores consumes about 7 MW. The cost of power for this and similar HPC systems runs into millions per year.

To further add to the concerns, due to power and cooling requirements and associated costs, empirical data show that every 10 degree Celcius increase in temperature results in a doubling of the system failure rate, which reduces the reliability of these expensive system. As supercomputers, large-scale data centers are meant to be clus-

ters composed by hundreds of thousands or even millions processing cores [11] with similar power consumption concerns.

Existing and ongoing research in power efficiency and power management has addressed the problem at different levels, including, for example, data center design, resource allocation, workload layer strategies, cooling techniques, etc. At the platform level (individual node or server), current power management research broadly falls into the following categories - processor and other subsystems (e.g. memory, disk, etc.) level, Operating System (OS) level and application level. Although the processor is the most power consuming component, other subsystems have incorporated energy management functionalities such as memory, storage and network interfaces (NIC). Within the OS, there are fewer power management techniques available, and include OS control of processor C-states, P-states and device power states or sleep states. At the application level several approaches have been also proposed such as those based on exploiting communication bottlenecks in MPI programs.

In this chapter, we study the potential of proactive application-centric aggressive power management of data center's resources for HPC workloads. Specifically, we consider power management mechanisms and controls (currently or soon to be) available at different levels and for different subsystems, and leverage several innovative approaches that have been taken to tackle this problem in the last few years, that can be effectively used in a cross-layer application-aware manner for HPC workloads. To do this, we firstly profile standard HPC benchmarks with respect to behaviors, resource usage and power dissipation. Specifically, we profile the HPC benchmarks in terms of processor, memory, storage subsystem and NIC usage. From the profiles we observe that across different workloads, the utilization of these subsystems varies significantly and there are significant periods of time in which one or more of these subsystems are idle, but still require a large amount of power. Based on the empirical power characterization and quantification of the HPC benchmarks, we investigate using simulations the potential energy saving of proactive, application-aware, power management strategies. We use traces from different systems and focus on performance and energy consumption metrics. The obtained results show that by using proactive, component-based power management, we can reduce the average energy consumption. The results also show that proactive configuration of subsystems works better with a higher number of nodes and with workloads composed of bursts of job requests with similar requirements, which is common in scientific HPC workflows.

The main contributions of this work are summarized as follows: we (i) argue that different existing techniques for energy management can be combined to improve energy efficiency of data center's servers by configuring them dynamically depending on the workloads' resource requirements, (ii) profile HPC benchmarks with respect to behaviors, resource usage and power impact on individual computing nodes and determine empirically (rather than with estimations) possible ways to save energy, (iii) propose different algorithms for proactive, component-based power management attempting to improve energy efficiency with little or no performance loss, and (iv) quantify possible energy savings of the proposed power management strategies at both server and datacenter levels.

The rest of this chapter is organized as follows:

- Discussion of background and related work (Section 1.2)
- Description of proactive, component-based power management (Section 1.3)
- Quantification of possible power savings through component-based power management (Section 1.4)
- Experimental evaluation and discussion of obtained results (Section 1.5)
- Concluding remarks (Section 1.7)

1.2 BACKGROUND AND RELATED WORK

High Performance Computing (HPC), is the application of parallel processing for running advanced application programs (that are either too large for standard computers or would take too long) efficiently, reliably and quickly. A HPC system, is essentially a network of nodes, each of which contains one or more processing units, as well as its own memory. These systems are ranked by the Top 500 list[1] that lists the fastest supercomputers worldwide based on the highest score measured using the Linpack benchmark suite in terms of TFlops (trillions of floating point operations per second).

As demand for processing power and speed grows, issues related to power consumption, air conditioning, and cooling infrastructures are critical concerns in terms of operating costs. Furthermore, power and cooling rates are increasing eight-fold every year [1], and are becoming a dominant part of IT budgets. Addressing these issues is thus an important and immediate task for HPC systems. While Top 500 focuses on performance, the Green500 list[2] provides rankings of the most energy-efficient supercomputers in the world based on the "Flops-per-Watt" metric [54].

In the following, we review the most significant power management techniques using different approaches, among the vast literature in the area of power management and energy efficiency for HPC.

1.2.1 CPU Power Management

In recent work, Liu et al. [38] survey power management approaches for HPC systems. As they discuss, since processors dominate the system power consumption in HPC systems, processor level power management is the most addressed aspect at server level. The most commonly used technique for CPU power management is Dynamic Voltage and Frequency Scaling (DVFS), which is a technique to reduce power dissipation by lowering processor clock speed and supply voltage [26, 27].

[1]Top 500 Supercomputers Site. http://www.top500.org/
[2]Top 500 Most Energy-Efficient Supercomputer Site. http://www.green500.org

1.2.1.1 OS-level CPU Power Management

OS-level CPU power management involves controlling the sleep states or the C-states [43] and the P-states of the processor when the processor is idle [59]. C-state is the capability of the processor to go in various low power idle states with varying wakeup latency. P-state is the capability of running the processor at different voltage and frequency levels [60]. The Advanced Configuration and Power Interface (ACPI) specification provides the policies and mechanisms to control the C-states and P-states of the processor when they are idle. Modern operating systems (e.g. Linux kernel) implement ACPI-based policies to reduce the processor performance and power when it is less active or in idle state [57].

1.2.1.2 Workload-level CPU Power Management

Several approaches to enforce power management based on the workload characteristics have already been developed. Some of the most successful approaches were based on overlapping computation with communication in MPI programs, and using historical data and heuristics. Kappiah et al. [31] developed a system called Jitter that exploits inter-node bottleneck in MPI programs (i.e. execute blocked processes due to synchronization points in lower P-sates). Lim et al. [37] developed a MPI runtime system that dynamically reduces CPU performance during communication regions assuming that in these regions the processor is not on the critical path. Other approaches have also studied the bound on the energy saving for an application without incurring in significant delay [50]. Freeh et al. proposed a model to predict execution time and energy consumed of an application running at lower P-states [22] and techniques based on phase characterization of the applications, assigning different P-states to phases according the previous measurements and heuristics [21]. Cameron et al. [6] proposed power management strategies based on application profiles but they concentrate only on power management of the CPU using dynamic voltage and frequency scaling (DVFS) and does not implement any power control of the peripheral devices.

Researchers have developed different scheduling algorithms and mechanisms to save energy to provision resources under deadline restrictions. Chen et al. [7] address resource provisioning proposing power management strategies with SLA constraints based on steady state queuing analysis and feedback control theory. They use server turn on/off and DVFS for enhancing power savings.

1.2.1.3 Cluster-level CPU Power Management

Ranganathan et al. [49] designed cluster level power management controller and employed a management agent running on each server and the server which exceeded the power budget according to the SLA, was throttled down to an appropriate level. Horvath et al. [25] exploited DVFS for use with dynamic reconfiguration for multi-tier server clusters, which is a typical architecture of current server clusters. Wang et al. [62] proposed a control algorithm to manage power consumption of multiple servers simultaneously. The controller monitors the power value and CPU utilization of each server, computes a new CPU frequency for each processor and directs each processor to change frequency in a coordinated way. Leveraging DVFS mechanism, Hsu et al. [27] propose automatically-adapting, power aware algorithm that is transparent to

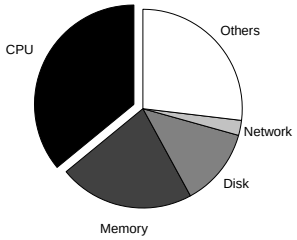

Figure 1.1 Possible distribution of power requirements per component

end-user applications and deliver considerable energy savings with tight control over DVFS-induced performance slowdown. Rountree et al. [51] developed a system called Adagio to collect statistical data on task execution slacks, compute the desired frequency and represent the result in a hash table. When task executes again, an appropriate frequency can be found in the hash table. Raghavendra et al. [47] propose a framework which coordinates and unifies five individual power management solutions (consisting of HW/SW mechanisms). Their work leverages feedback mechanisms to federate multiple power management solutions and builds an approach to unify solutions with minimum interference across controllers. Moreover, their coordination solution gives higher power savings than individual methods.

1.2.2 Component-based Power Management

Although the CPU is the component that requires most power of the server, the relative power demand other components is increasing very quickly, specially for multi- and many- core architectures, where different cores and an important amount of memory are included in the same chip die. Figure 1.1 illustrates a possible decomposition of the power requirements of a server by components for a state of the art multi-core server. We can appreciate that the aggregation of power requirements of memory, disk and network are comparable to the power dissipated by the CPU. However, the power required by each component depend of the workload characteristics [18]. Other components include the motherboard, chipset, fans, power supplies, etc. but we do not consider them in our approach.

In the following, we discuss existing power management approaches at the component level.

1.2.2.1 Memory Subsystem Substantial work has been also done for adapting the RAM memory subsystem for saving energy. Delaluz et al. [14, 12] studied compiler-directed techniques, as well as OS-based approaches [13, 15] to reduce the energy consumed by the memory subsystem. Huang et al. [28] proposed power-aware virtual memory implementation in OS to reduce memory energy consumption. Fradj et al. [20, 19] propose multi-banking techniques that consist of setting indi-

vidually banks in lower power modes when they are not accessed. Diniz et al. [16] study dynamic approaches for limiting the power consumption of main memories by limiting consumption by adjusting the power states of the memory devices, as a function of the memory load. Hur et al. [29] propose using the memory controller (thus, at chip level) to improve RAM energy efficiency. They exploit low power modes of modern RAMs extending the idea of adaptive history-based memory schedulers.

1.2.2.2 Storage Subsystem

Existing research work also addresses the storage subsystem management to improve energy efficiency of servers. Rotem et al. [17] focus on the energy consumed by the storage devices like hard disks in standby mode. They suggest file allocation strategies to save energy with a minimal effect on the system performance i.e. the file retrieval time, while reducing the I/O activity when there is no data transfer. Pinheiro et al. [45] study energy conservation techniques for disk array-based network servers and propose a technique that leverages the redundancy in storage systems to conserve disk energy [46]. Colarelli et al. [9] analyze an alternative design using massive arrays of idle disks (spin-down/up disks) Other approaches have addressed energy efficiency of storage systems by spinning-down/up disk [9] and the reliability of such techniques [63]. Solid State Drive (SSD) disks have been also taken into account towards saving energy consumption for the storage subsystem [53, 34].

The research work discussed above addresses energy efficiency by managing different subsystems individually (e.g. CPU via DVFS). However, recent approaches have proposed energy efficiency techniques for processor and memory adaptations [35, 8]. Li et al. [36] combine memory and disk management techniques to provide performance guarantees for control algorithms. Ranganathan et al. [49] highlight the current issue of under utilization and over-provisioning of the servers. They present a solution of peak power budget management across a server ensemble to avoid excessive over-provisioning considering DVS and memory/disk scaling. In contrast to all these approaches, we consider dynamic configuration of multiple subsystems within a single server. Thus, we propose using different mechanisms and techniques that have been already developed in different domains. Our approach is then complimentary to existing and ongoing solutions for energy management for HPC data centers.

1.2.3 Thermal-Conscious Power Management

Several approaches have been proposed for energy efficiency in datacenters including factors such as cooling and thermal considerations. More et al. [41] propose a method to infer a model of thermal behavior to automatically reconfigure the thermal load management systems, thereby improving cooling efficiency and power consumption. They also propose in [40] thermal management solutions focusing on scheduling workloads considering temperature-aware workload placement. Bash et al. [2] propose a policy to place the workload in areas of a data center that are easier to cool resulting in cooling power savings. Tang et al. [58] formulate and solve a mathematical problem that maximizes the cooling efficiency of a data center. This

is focused on task assignment that maximizes the cooling efficiency. Bianchini et al. [24] propose emulation tools for investigating the thermal implications of power management. In [48], they present C-Oracle, a software prediction infrastructure that makes online predictions for data center thermal management based on load redistribution and DVFS.

1.2.4 Power Management in Virtualized Datacenters

With the raise of Cloud computing, virtualized datacenters are being increasingly considered for traditional HPC applications. In the context of virtualized datacenters, Nathuji et al. [42] investigate the integration of power management and virtualization technologies. In particular they propose VirtualPower to support the isolated and independent operation of virtual machine and control the coordination among virtual machines to reduce the power consumption. Rusu et al. [52] propose a cluster-wide on/off policy based on dynamic reconfiguration and DVS. They focus on power, execution time, and server capacity characterization to provide energy management. Kephart et al. [32, 10] address the coordination of multiple autonomic managers for power/performance tradeoffs by using a utility function approach in a non-virtualized environment.

A large body of work in data center energy management addresses the problem of the request distribution at the Virtual Machine (VM) management level in such a way that the performance goals are met and the energy consumption is minimized. Song et al. [55] propose an adaptive and dynamic scheme for adjusting resources (specifically, CPU and memory) between virtual machines on a single server to share the physical resources efficiently. Kumar et al. [33] present vManage, a practical coordination approach that loosely couples platform and virtualization management towards improving energy savings and QoS, and reducing VM migrations. Soror et al. [56] address the problem of optimizing the performance of database management systems by controlling the configurations of the virtual machines in which they run. Laszewski et al. [61] present a scheduling algorithm for VMs in a cluster to reduce power consumption using DVFS.

1.3 PROACTIVE, COMPONENT-BASED POWER MANAGEMENT

Our approach is based on power management of the different components at the physical server layer in a proactive manner. We assume that the application's profiles (in terms of resource usage) are known in advance. Therefore, we can power down subsystems or use low power modes of a host system that are not required by the jobs mapped to it based on the application's profiles. We also map jobs to physical servers attempting to optimize energy efficiency with minimum penalty in performance.

In contrast to other typical approaches that allocate jobs with non-conflicting, i.e. dissimilar, resource requirements together on the same physical server in order to optimize the performance, our policy is to allocate jobs with similar resource requirements together on the same physical server. This allows us to downgrade the

subsystems of the server that are not required to run the requested jobs in order to save energy. To do this, we consider specific configurations of the physical servers' subsystems to reduce their energy demand. Specifically it follows an energy model that leverages previous research on energy-efficient hardware configurations (e.g. low power modes) in four different dimensions:

- CPU speed using Dynamic Voltage and Frequency Scaling (DVFS). We are able to reduce the energy consumed by those applications that are, for example, memory-bound [30].
- Memory usage. For those applications that do not require high memory bandwidth we consider the possibility of slightly reducing the memory frequency or possibly shutting down some banks or channels of memory in order to save power [3].
- High performance storage. It may be possible to power down unneeded disks (e.g. using flash memory devices that require less power) or by spinning-down disks [4].
- High performance network interfaces. It may be possible to power down some network subsystems (e.g. Myrinet interfaces) or using idle/sleep modes.

1.3.1 Job Allocation Policies

We have implemented two different job allocation policies: a static approach where physical servers maintain their initial subsystem configuration, and a dynamic one that allows the physical servers to be reconfigured dynamically. The algorithm followed by the static resource provisioning approach for a given job is shown in Equation 1.1. For readability, we have simplified the algorithm assuming that each job can be allocated in a single server. The complete approach returns a set of servers. Given the resource requirements of a job request $(req_{cpu}, req_{mem}, req_{disk}, req_{nic})$, the available physical servers $(s_1, \ldots, s_n)$, it returns the most appropriate server to run the requested job. The resource requirements of the job request are the CPU, memory, storage and network demand, respectively.

$$\begin{aligned} \underline{job\_mapping}(&req : job\ request, \\ &(req_{cpu}, req_{mem}, req_{disk}, req_{nic}) : resource\ requirements, \\ &S = (s_1, \ldots, s_n) : physical\ servers) = s_k\ : \\ &s_k \in S\ \wedge\ S' = \underline{match\_reqs}(req, S) \wedge \\ &s_k \in S'\ \wedge\ s_k \in \underline{less\_reqs}(S')\ ) \end{aligned} \tag{1.1}$$

First, the algorithm discards the servers that do not match the resource requirements of the job request. To do this, it uses the $\underline{match\_reqs}$ function, which is defined in Equation 1.2.

$$\underline{match\_reqs}(req, S) = S' \Leftrightarrow S' \subseteq S \ \wedge \ \forall s_i \in S' : \quad (1.2)$$
$$(\ s_{i_{cpu}} \geq req_{cpu} \wedge s_{i_{mem}} \geq req_{mem} \wedge$$
$$s_{i_{disk}} \geq req_{disk} \wedge s_{i_{nic}} \geq req_{nic} \ )$$

If a server that matches the job requirements is not available, the job request cannot be served. If we follow a First Come First Serve (FCFS) scheduling policy with the static approach, a request may remain queued (thus blocking all following queued jobs) until a server with the required configuration becomes available. However, the scheduling policy may decide selecting another job from the queue (e.g. backfilling jobs). Otherwise, we select the server with lowest power requirements (i.e. with the most subsystems disabled or in low power mode) and hosting the fewest jobs from the set of matching servers. It allows us to balance the load among the servers and avoid possible contention of resources. To select the server that best matches the conditions described above, the $\underline{less\_reqs}$ function is used.

In our dynamic approach, when required physical resources are unavailable, we reconfigure an available physical server to provide the appropriate characteristics and then provision it. Specifically, we can reconfigure servers if they are idle, but if there are no idle servers available, we can reconfigure only those servers that are configured to use fewer subsystems than those that are requested (if a server is configured to deliver high memory bandwidth, we cannot reconfigure it to reduce its memory frequency, since that would negatively impact jobs already running on it. However, if a server is configured with reduced memory frequency, we can reconfigure it to deliver full memory bandwidth without negatively impacting running jobs). Moreover, we try to fill servers with requests of similar types. Not only does this efficiently load servers, it allows more servers to remain fully idle, which allows them to be configured to host new jobs.

1.3.2 Workload Profiling

In order to define the application's profiles, we characterize the workload behavior into I/O intensive, memory intensive, communication intensive and compute intensive regions with respect to time. Most of the standard profiling utilities are designed for comparing computation efficiency of the workloads and systems on which they are running, hence their outputs are not very useful from the subsystem usage point of view. Based on the workload characterization we can perform an efficient job allocation as described in section 1.3.1.

We profiled standard HPC benchmarks with respect to behaviors and subsystem usage on individual servers. It allows us to estimate the possibilities of component-based power management in HPC workloads (see section 1.4). To collect run-time OS-level metrics for CPU utilization, hard disk I/O, and network I/O we used different mechanisms such as "mpstat", "iostat", "netstat" or "PowerTOP" from Intel. We also patched the Linux kernel 2.6.18 with the "perfctr" patch so that we can read hardware

performance counters on-line with relatively small overhead. We instrumented the applications with PAPI and, since the server architecture does not support total memory LD/ST counter, we counted the number of L2 cache misses, which indicates (approximately) the activity of memory.

A comprehensive set of HPC benchmark workloads has been chosen. Each stresses a variety of subsystems - compute power, memory, disk (storage), and network communication. They can be classified in three different classes:

- *Standard*: **HPL** Linpack that solves a (random) dense linear system in double precision arithmetic, and **FFTW** that computes the discrete Fourier transform.
- *CPU intensive*: **TauBench**, which is an unstructured grid benchmark of Navier Stokes solver kernels.
- *I/O intensive*: **b_eff_io**, which is a MPI-I/O application, and **bonnie++** that focus on hard drive and file system performance. We ran two distributed instances of bonnie++ using a script and ssh.

Figure 1.2 shows the obtained profiles for three representative benchmarks with different behaviors and trends. Axes of the plots have time as the X-axis and on the Y-axis we show, from the top to the bottom: CPU utilization, memory utilization (L2 cache misses), disk utilization (number of blocks accessed), network utilization (traffic of packets on the NIC), and the average p-state residency of the CPU's cores. The plots show the measurements as well as the bezier curves (dashed lines) to better identify their trends, except the plots of p-state residency that only show the bezier curves, for readability.

The application's profiles show different usage level of the subsystems over time. However, subsystem's usage can be discretized into CPU-, memory-, disk- and network- bound, based on the potential impact of using low power modes on the application execution's performance. For example, in Figure 1.2 we can appreciate that HPL shows low disk usage, b_eff_io shows low CPU usage and bonnie++ shows low CPU and NIC usage. However, there are other subsystems that have low usage only during some intervals of time, such as memory in bonnie++.

In section 1.4 we discuss the trends and quantify the power saving opportunities based on the application profiles such as those shown in Figure 1.2.

1.4 QUANTIFYING ENERGY SAVING POSSIBILITIES

The fundamental requirement to study the potential energy saving with the approach suggested in this chapter is to gather reliable usage data for processor, memory, storage subsystem and the NIC, and their associated power requirements for a set of representative and standard HPC workloads. It allows us to quantify the potentials of component-level power management and to define an upper-bound for possible energy savings. Along with the potential energy savings of using component-based power management, we also study the possible overheads of using these low power

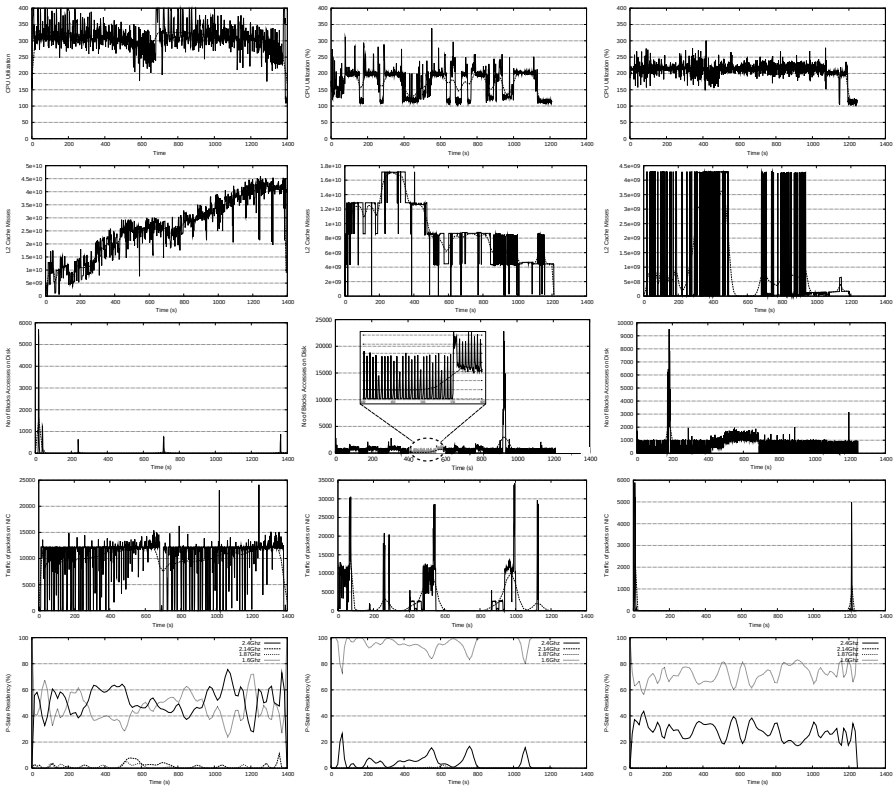

Figure 1.2 Application profiles for HPL (left), b_eff_io (center) and bonnie++ (right) benchmarks

modes and switching between the different modes. To do this, we characterize and analyze the power dissipation of the different subsystems and quantify the possible saving using existing techniques based on using low power modes to reduce the energy consumption. Although we first focus on single servers, using the profiling information we will be able to proactively map job requests to servers configured with the appropriate low power modes at the datacenter level.

1.4.1 Methodology

We conducted experiments with two Dell servers, each with a Intel quad-core Xeon X3220 processors, 4GB of memory, two SATA hard disks, and two 1Gb Ethernet interfaces. We also used a 160GB Intel x25-M Mainstream SATA Solid State Drive (SSD) disk. The processors operate at four frequencies ranging from 1.6GHz to 2.4GHz. This is intended to represent a general-purpose rack server configuration, widely used in large data centers.

To empirically measure the "instantaneous" power consumption of the servers we used a *"Watts Up? .NET"* power meter. This power meter has an accuracy of $\pm 1.5\%$ of the measured power with sampling rate of 1Hz. The meter was attached between the wall power and the server. We estimate the consumed energy integrating the actual power measures over time.

1.4.2 Component-Level Power Requiremets

In order to quantify the possible power savings of using component-based power management in a server, we have studied empirically the power characteristics of different subsystems individually. Specifically, we have studied CPU, RAM memory, disk storage, and NIC. Equation 1.3 shows the simplified dynamic power dissipation model that we consider for CPU, where C is the capacitance of the processor (that we consider fixed), α is an activity factor (also known as switching activity), and V and f are the operational voltage and frequency, respectively.

$$P_{cpu} \sim C \times V^2 \times \alpha \times f \tag{1.3}$$

Table 1.1 summarizes the server's power savings and the associated delays for the different subsystem. For the CPU, the workload was generated with lookbusy (a synthetic load generator). During CPU activity, the power demand differs up to around 82W (i.e. 39% of total server power) depending of the frequency used, but without any load, the difference is only up to around 8W (i.e. 3.78% of total server power). However, although CPU power is the more power demanding subsystem of the server, we rely on the CPU frequency management performed within the OS with "cpufreq" using the "ondemand" governor. For disk storage we consider two different possibilities, on the one hand, using spin down/up techniques with traditional disks, and, on the other hand, using a SSD disk. With a traditional disk we can save almost 10W of power (i.e. around 7.5%). However, there is an overhead for spinning down/up the disk. For spinning down the disk the delay is around 0.05 seconds and for spinning up the delay is around 5-6 seconds. There is also an overhead of energy due to the peak power required to spin up the disk's motor (around 60J of energy, according to our experiments). We also consider using a SSD drive, which can save around 14W of power when it is idle (i.e. 3% less power with respect to a disk in low power mode), according to our experiments. The SSD drive also has a much faster access time and does not require spinning down techniques to reduce its power consumption.

We use low power mode for the network subsystem switching on/off the NIC dynamically. We made the assumption that data centers' servers have usually two different network interfaces (a faster one for actual computations and a slower one for control/administration purposes). Disabling the NIC we can save around 3W (i.e. 2.47%) and the overheads for switching on and switching off the NIC are around 0.15s and 3-4s, respectively.

Memory power dissipation can be classified as being dynamic power dissipation that occurs only during reads and writes, or static power dissipation due to transistor

Subsystem	Savings	Delay
CPU freq (idle)	8 W	"instantaneous"
CPU freq (loaded)	82 W	"instantaneous"
RAM memory	8 W	"instantaneous"
Hard disk	10 W	5-7s
Solid state disk	14 W	"instantaneous"
NIC	3 W	0.15s (on) 3-4s (off)

Table 1.1 Server's power savings and associated delays

leakage. Equation 1.4 shows a simple model for memory static power dissipation, where Vcc is the supply voltage, N is the number of transistors, k_{design} is a design dependent parameter, and I_{leak} is a technology dependent parameter. We will consider k_{design} and I_{leak} fixed parameters.

$$P_{static} = Vcc \times N \times k_{design} \times \hat{I}_{leak} \tag{1.4}$$

Since the increasing contribution of static power is clearly evident even in today's design, we can reduce the static power dissipation reducing either Vcc or N. Some existing approaches based on multi-banking techniques try to set banks of memory in lower power modes when they are not accessed, thus reducing N. Other approaches may dynamically reduce the voltage when memory is not in the critical path of the running workload. Since these techniques are not standardly available in widely used systems (such as ours), we estimate the potential savings from memory removing physically two of the four banks of memory that are available in the server. Using the same subsystems configurations, but with only 2GB of RAM memory installed, we were able to save around 8W of power (i.e. 5.78%), on average. We estimate short delay for switching to low power mode.

1.4.3 Energy Savings

In this subsection, we first present the estimated energy savings for a single server using a power model based on the empirical measurements shown in Table 1.1 and assuming an accurate use of low power modes ("Simulation" in Figure 1.3). Thereby, we assume that the workload profile is known in advance. The simulations were conducted using MATLAB. We used the benchmarks presented in section 1.3.2 that, as we discussed previously, have different requirements and behaviors in terms of subsystems utilization. We also present the energy saving obtained from actual experiments on real hardware ("Validation" in Figure 1.3). We applied low power modes based on the application profiles with minimal penalty in performance. In addition, we performed experiments using SSD technology for storage ("With SSD" in Figure 1.3). Although we present the saving for a single server, the results were obtained using the testbed described in section 1.4.1.

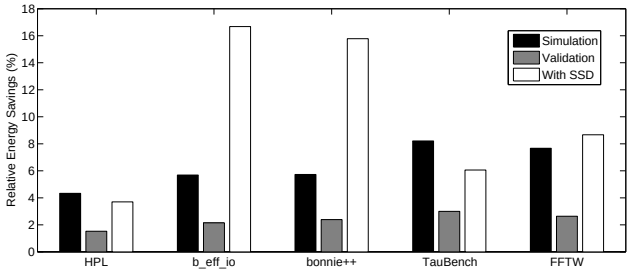

Figure 1.3 Relative energy savings per benchmark

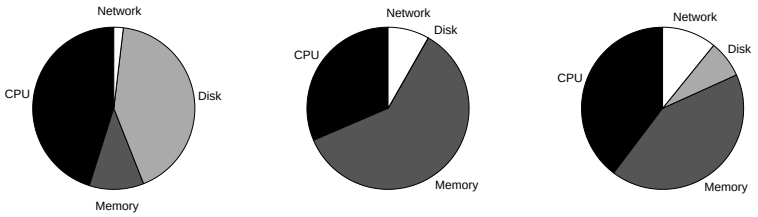

Figure 1.4 Energy savings per component for HPL (left), b_eff_io (center) and bonnie++ (right)

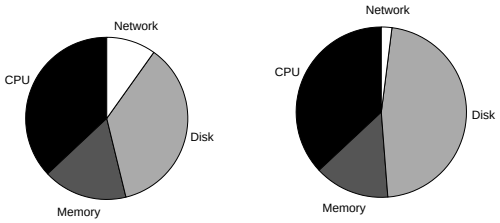

Figure 1.5 Energy savings per component for TauBench (left) and FFTW (right)

Figure 1.3 shows the relative energy savings with respect to the energy used without component-level power management techniques, for each of five different benchmarks. We can appreciate that CPU- and network- intensive benchmarks provide more opportunities of energy savings from disk optimization (e.g. FFTW) while I/O-intensive benchmarks provide more opportunities of energy savings from other subsystems (e.g. NIC). Furthermore, benchmarks with higher utilization of the different subsystems (i.e. HPL) obtains less energy savings. The average energy saving with actual executions is lower than the energy saving estimated through simulations due to the lack of memory power management in the real hardware and the lesser accuracy in switching between power modes. Using SSD technology

reduces the energy consumption significantly. For non-disk intensive benchmarks (e.g. HPL) the savings are moderate while the energy savings are much higher for I/O intensive benchmarks.

Figures 1.4 and 1.5 show the distribution of energy savings per component for the different benchmarks using simulation. We do not observe any clear correlation between the distribution of energy savings per component and the total energy savings because, among other things, the total savings may depend of the amount of time that the subsystems are used in low power mode during the benchmark's execution.

1.5 EVALUATION OF THE PROPOSED STRATEGIES

In this section, we evaluate the possible energy savings that can be achieved at the datacenter level using proactive, component-based power management with a deterministic approach. To do this, we use simulation with traces of parallel workloads. Along with traces, we also use the profiling of workloads and energy savings at server level shown previously.

1.5.1 Methodology

To evaluate the performance and energy efficiency of the proposed approach, we used real HPC workload traces from widely distributed production systems. Since not all of the required information is obtainable from these traces, some data manipulation was needed.

For our experiments, we have used the kento-perf simulator (formerly called Alvio [23]), which is a C++ event driven simulator that was designed to study scheduling and allocation policies in HPC systems. We have simulated a homogeneous cluster system based on servers with 4 CPUs and 8GB of RAM each, which is a state of the art server configuration. We also have considered the number of nodes that conformed the original systems of the workloads described in section 1.5.2.

Using our measurements and existing research [39][44] (e.g. to obtain the power required by a subsystem scaling from the total server power), we configured the simulations with the power required for the different subsystems and the switch latencies shown in Table 1.1. The model has some simplifications, such as using a coarse grain level for switch latencies (we use longer latencies) due to the accuracy of the simulator is by the order of seconds. Specifically, for the CPU we consider three different states: running mode, i.e., C0 C-state and highest P-state (no savings), low power mode, i.e., C0 C-state and the deepest P-state, and idle state, i.e., C-state different to C0 (saving shown in Table 1.1). For the memory and storage subsystems, we consider two states (regular and low power mode) based on Table 1.1 assuming the use of newer technology for memory power management.

Since we assume that modern systems use power management techniques within the operating system, we consider low power mode in our simulations when the servers are idle due to low power modes may be significantly lower than in running state [5]. We also assume that when an application running on a server with one of

its required subsystems in idle mode, the operating system will switch the required subsystems to running mode. As well as taking into account the power required by the previous subsystems, we also include in our model the power required by other components such as motherboard, fans, etc. Therefore, some fixed amount of power is always required, independently of the specific physical server configuration used. However, we do not consider the power required for cooling and to manage external elements.

Although this model is not completely accurate with respect to applications' execution behaviors, it gives us a base framework to evaluate the possibilities of proactive, component-based power management.

1.5.2 Workloads

In the present work, we have used traces from the Grid Observatory[3], which collects, publishes, and analyzes logs on the behavior of the EGEE Grid[4], and traces of the Intel Paragon system located at the San Diego Supercomputer Center (SDSC) from the parallel workload archive. While SDSC is a traditional HPC system composed by 416 homogeneous nodes, the EGEE Grid is a large-scale heterogeneous and distributed system composed of more than 4,200 nodes. Since the traces are in different formats and include data that is not used, they are pre-processed before being input to the simulation framework. First, we convert the input traces to Standard Workload Format (SWF)[5]. We also combine the multiple files of which they are composed into a single files. Then, we clean the trace in SWF format in order to eliminate failed jobs, cancelled jobs and anomalies.

As the traces found from different systems do not provide all the information needed for our analysis, we needed to complete them using a model based on the benchmarking of HPC applications (see Section 1.3.2). After calculating the average percentage of CPU, memory, storage and network usage for each benchmark, we randomly assign one of the possible benchmark profiles to each request in the input trace, following a uniform distribution. We also generate two variants of each trace randomly assigning benchmark profiles by bursts. The bursts of job requests are sized (randomly) from 1 to 5 job requests and from 1 to 10 job requests. These traces are intended to illustrate the submission of scientific HPC workflows which are composed of sets of jobs with the same resource requirements.

1.5.3 Metrics

We evaluate the impact of our approach on the following metrics: makespan (workload execution time, which is the difference between the earliest time of submission of any of the workload tasks, and the latest time of completion of any of its tasks), average bounded slowdown (BSLD), energy consumption (based on both static and

[3]Grid Observatory Site. http://www.grid-observatory.org/
[4]Enabling Grid for E-sciencE Site, http://www.eu-egee.org/
[5]Parallel Workload Archive Site. http://www.cs.huji.ac.il/labs/parallel/workload/

dynamic energy consumption), Energy Delay Product (EDP). We define BSLD for a given job:

$$BSLD_{job} = max\left(1, \frac{runtime_{job} + waittime_{job}}{max(runtime_{job},\ threshold)}\right)$$

We consider a threshold of 60 seconds, which is commonly used in HPC systems to avoid the influence of unrepresentative (very short) jobs.

1.6 RESULTS

We have conducted our simulations using the proposed strategies, workloads and system models described in the previous sections with respect to a reference approach, which represents the most commonly configuration used in HPC datacenters. Specifically, we have evaluated the following strategies:

- **REFERENCE (REF)** - implements the typical reactive power management at the operating system level (i.e. DVFS when the CPU is not loaded). It follows the First-Fit resource selection policy to allocate job requests to servers. This means that it maps a given job to the first available physical servers that match the request requirements.
- **STATIC** - implements our proactive approach with the proposed static allocation policy. It means that the servers' configurations remain constant; therefore a physical server can host only applications that match its specific characteristics. Specifically, we consider eight classes of configurations (one for each possible combination of their subsystems configuration) and we model the same amount of servers with each one.
- **DYNAMIC** - implements our proposed approach similarly to the STATIC strategy but implementing dynamic resource re-configurations when they are necessary. This means that when there are not available resources configured with the requested configuration, it re-configures servers reactively in order to service new application requests.
- **DYNAMIC-2** - implements the same policy of the DYNAMIC strategy but it allows to re-configure the subsystems of a server only when it is idle .

We have used three different variants of the workloads described in section 1.5.2:

- **NO-BURSTS** - follows the original distribution of the workloads described in the previous section.
- **BURSTS-5** and **BURSTS-10** - job requests are by bursts sized randomly from 1 to 5 and from 1 to 10, respectively.

Figures 1.6, 1.7, 1.8 and 1.9 show the relative makespan, average BSLD, energy consumption and EDP results, respectively. In each of these figures, we show the

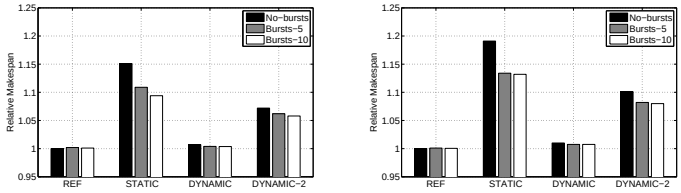

Figure 1.6 Relative makespan for EGEE (left) and SDSC (right)

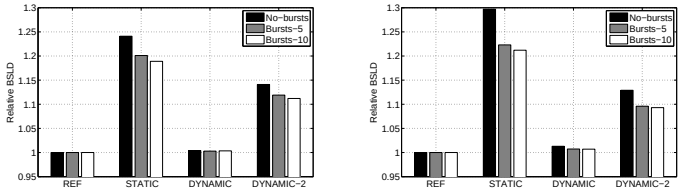

Figure 1.7 Relative BSLD for EGEE (left) and SDSC (right)

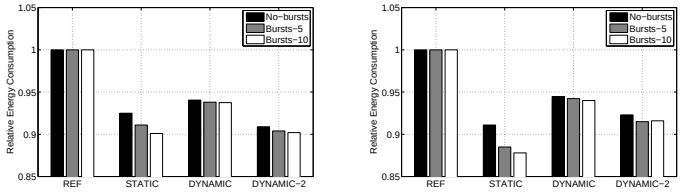

Figure 1.8 Relative energy comsumption for EGEE (left) and SDSC (right)

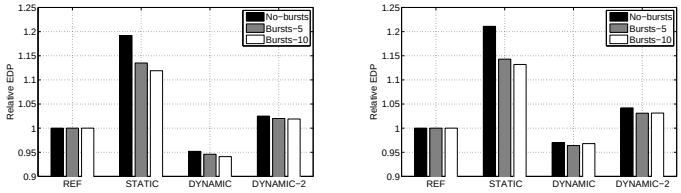

Figure 1.9 Relative EDP for EGEE (left) and SDSC (right)

results obtained with both EGEE and SDSC traces in the same Y axis scale. For readability, the results are normalized to the results obtained with the REFERENCE configuration and the NO-BURSTS workload variant.

Figures 1.6 and 1.7 show that both makespan and BSLD results follow similar patterns. With the REFERENCE strategy, the delays are shorter than those obtained with the STATIC strategy. However, the delays obtained with the REFERENCE and

with the DYNAMIC strategy are very similar. This is explained by the fact that using the STATIC strategy with available resources, matching the request requirements is harder because the number of resources with the same subsystem configuration is fixed. This is specially significant with the SDSC because it has less number of nodes and, therefore, a small number of nodes of each subsystem configuration. In fact, the obtained results with EGEE are, in general, better than those obtained with SDSC because with EGEE there are a much higher number of nodes of each configuration, which results in higher probability to find resources that matches the job's requirements. As with the STATIC strategy, the makespan obtained with the DYNAMIC-2 policy is longer than that obtained with the REFERENCE strategy. In both cases, the makespan is longer due to scheduling issues (resource limitations cause job blocking in the queue) and not due to the use of low power modes.

The BSLD is higher with the STATIC and DYNAMIC-2 strategies. It means that with both of these policies the job waiting times are much longer than the job waiting times with STATIC and DYNAMIC. The impact of the size of the workload' bursts on the different metrics is significant with the STATIC and DYNAMIC-2 strategies. This is explain due to the fact that it is easier to allocate jobs in nodes with the same configuration when they are submitted together (filling servers with jobs with same requirements). However, using the SDSC workload there is not much difference between bursts of up to 5 and bursts up to 10 because in SDSC the number of nodes is smaller.

Although the makespan is shorter with the REFERENCE approach, the energy consumption is lower with our proposed strategies than that obtained with REFERENCE. Specifically, both STATIC and DYNAMIC approaches obtain between 6-12% energy savings with respect to the REFERENCE approach. The EDP obtained with the DYNAMIC strategy is around 5% lower (on average) than the that obtained with the REFERENCE approach. However, the DYNAMIC-2 strategy presents a higher EDP. In both cases, the energy efficiency is better for those workloads composed by bursts of jobs with similar requirements. It allows us to perform a more efficient mapping due to the probabilities to reconfigure a server are lower. It also results in lower over-provisioning of the server's subsystems, which is the difference between the subsystems that are in active mode and the subsystems required by the applications. Although the energy consumption is lower with the STATIC approach with respect to the REFERENCE approach, the EDP is much higher. This is explained due to the fact that the limitations in the subsystems availability results in lower resource utilization and, therefore, even though the energy consumption of the resources is lower, the resources are used during longer time

Therefore, we can conclude that STATIC and DYNAMIC-2 strategies do not provide significant improvements with respect to the REFERENCE approach but DYNAMIC presents better energy efficiency (more than 5%, on average) with very little penalty on the performance (makespan). Moreover, we have stated that a higher number of nodes facilitates the proactive configuration of subsystems in scenarios with restrictions (STATIC and DYNAMIC-2), provides more energy savings, and reduces the over-provisioning (in terms of subsystems).

1.7 CONCLUDING REMARKS

In this chapter, we studied the potential impact of deterministic application-centric power control at the device level on the overall energy efficiency of a system. Specifically, we analyzed the energy consumption of a node according to the usage of its processor, memory and storage subsystem, and the NIC. Moreover, we evaluated the possible energy savings at a datacenter level through the use of proactive power management.

Our simulations showed that proactive, component-based power management can be effective to save energy if the systems have sufficient mechanisms to provide an accurate dynamic management of the subsystems based on the characteristics of the workload. The results stated that using application-aware subsystem power control can save additional energy, so it is fundamental to characterize the workload appropriately.

We conclude that power management at the subsystem level can not be neglected due to the increasing requirements of energy efficiency optimization in large-scale data centers. We believe that our proposed predictive application-aware power management approach has sufficient potential to tackle this problem at the datacenter level. Moreover, we stated that the potential of proactive power management techniques is higher when the number of nodes available is higher. We also can conclude that current and ongoing technologies such as memory that allow DVS must be adopted and supported in large-scale data centers to enhance global energy optimizations. Finally, the findings of this work showed that there are opportunities to improve the job scheduling and resource allocation strategies in HPC systems considering proactive, component-level power management.

1.8 SUMMARY

Energy efficiency of large-scale data centers is becoming a major concern not only for reasons of energy conservation, failures, and cost reduction, but also because such systems are soon reaching the limits of power available to them. High Performance Computing (HPC) systems, may consume power in megawatts, and of all the power consumed by such a system only a fraction is used for actual computations. In this chapter, we studied the potential of application-centric proactive power management of data center's resources for HPC workloads. Specifically, we considered power management mechanisms and controls (currently or soon to be) available at different levels and for different subsystems, and leverage several innovative approaches that have been taken to tackle this problem in the last few years, can be effectively used in a application-aware manner for HPC workloads. To do this, we first profiled standard HPC benchmarks with respect to behaviors, resource usage and power impact on individual computing nodes. Based on the findings at the server level, we proposed proactive, component-based power management techniques with the purpose of improving energy efficiency with little or no performance loss. We then evaluated our proposed algorithm through simulations using empirical power characterization and

quantification. The obtained results showed that by using proactive, component-level power management, we can reduce the average energy consumption without significant penalty in performance if the systems have sufficient mechanisms to provide an accurate dynamic management of the subsystems based on the characteristics of the workload. The results also stated that the potential of proactive power management techniques is higher when the number of nodes available is higher.

Our findings motivate the development of autonomic components responsible for component-based power management and the implementation of power-aware scheduling and resource allocation strategies in HPC systems.

Acknowledgments

The research presented in this work is supported in part by National Science Foundation (NSF) via grants numbers IIP 0758566, CCF-0833039, DMS-0835436, CNS 0426354, IIS 0430826, and CNS 0723594, by Department of Energy via the grant number DE-FG02-06ER54857, by The Extreme Scale Systems Center at ORNL and the Department of Defense, and by an IBM Faculty Award, and was conducted as part of the NSF Center for Autonomic Computing at Rutgers University. This material was based on work supported by the NSF, while working at the Foundation. Any opinion, finding, and conclusions or recommendations expressed in this material; are those of the author and do not necessarily reflect the views of the NSF. The authors would like to thank the Grid Observatory, which is part of the EGEE-III EU project INFSO-RI-222667.

REFERENCES

1. Report to congress on server and data center energy efficiency. Technical report, U.S. Environmental Protection Agency, August 2007.
2. Cullen Bash and George Forman. Cool job allocation: Measuring the power savings of placing jobs at cooling-efficient locations in the data center. In USENIX Annual Technical Conf., pages 363–368, 2007.
3. Hanene Ben Fradj, Cecile Belleudy, and Michel Auguin. Multi-bank main memory architecture with dynamic voltage frequency scaling for system energy optimization. In EUROMICRO Conf. on Digital System Design, pages 89–96, 2006.
4. Timothy Bisson, Scott A. Brandt, and Darrell D.E. Long. A hybrid disk-aware spin-down algorithm with i/o subsystem support. In IEEE Intl. Performance, Computing, and Communications Conf., pages 236–245, 2007.
5. Qiong Cai, José González, Ryan Rakvic, Grigorios Magklis, Pedro Chaparro, and Antonio González. Meeting points: using thread criticality to adapt multicore hardware to parallel regions. In Intl. Conf. on Parallel Architectures and Compilation Techniques, pages 240–249, 2008.

6. Kirk W. Cameron, Rong Ge, and Xizhou Feng. High-performance, power-aware distributed computing for scientific applications. Computer, 38(11):40–47, 2005.

7. Yiyu Chen, Amitayu Das, Wubi Qin, Anand Sivasubramaniam, Qian Wang, and Natarajan Gautam. Managing server energy and operational costs in hosting centers. In ACM SIGMETRICS Intl. Conf. on Measurement and modeling of computer systems, pages 303–314, 2005.

8. Youngjin Cho and Naehyuck Chang. Memory-aware energy-optimal frequency assignment for dynamic supply voltage scaling. In ISLPED '04: Proceedings of the 2004 international symposium on Low power electronics and design, pages 387–392, 2004.

9. Dennis Colarelli and Dirk Grunwald. Massive arrays of idle disks for storage archives. In Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing, pages 1–11, 2002.

10. Rajarshi Das, Jeffrey O. Kephart, Charles Lefurgy, Gerald Tesauro, David W. Levine, and Hoi Chan. Autonomic multi-agent management of power and performance in data centers. In Intl. joint Conf. on Autonomous agents and multiagent systems, pages 107–114, 2008.

11. Jeff Dean. Large-scale distributed systems at google: Current systems and future directions. In 3rd ACM SIGOPS International Workshop on Large Scale Distributed Systems and Middleware, October 2009.

12. Victor Delaluz, Mahmut Kandemir, Narayanan Vijaykrishnan, Anand Sivasubramaniam, and Mary Jane Irwin. Hardware and software techniques for controlling dram power modes. IEEE Trans. Comput., 50(11):1154–1173, 2001.

13. Victor Delaluz, Mahmut T. Kandemir, and Ibrahim Kolcu. Automatic data migration for reducing energy consumption in multi-bank memory systems. In DAC '02: Proceedings of the 39th annual Design Automation Conference, pages 213–218, 2002.

14. Victor Delaluz, Mahmut T. Kandemir, Narayanan Vijaykrishnan, and Mary Jane Irwin. Energy-oriented compiler optimizations for partitioned memory architectures. In CASES '00: Proceedings of the 2000 international conference on Compilers, architecture, and synthesis for embedded systems, pages 138–147, 2000.

15. Victor Delaluz, Anand Sivasubramaniam, Mahmut T. Kandemir, Narayanan Vijaykrishnan, and Mary Jane Irwin. Scheduler-based dram energy management. In DAC '02: Proceedings of the 39th annual Design Automation Conference, pages 697–702, 2002.

16. Bruno Diniz, Dorgival Guedes, Wagner Meira, Jr., and Ricardo Bianchini. Limiting the power consumption of main memory. In ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture, pages 290–301, 2007.

17. Doron Rotem, Ekow Otoo, and Shih-Chiang Tsao. Analysis of Trade-Off Between Power Saving and Response Time in Disk StorageSystems. Fifth Workshop on High-Performance Power-Aware Computing (HPPAC'09) with IPDPS'09, May 2009.

18. Xizhou Feng, Rong Ge, and Kirk W. Cameron. Power and energy profiling of scientific applications on distributed systems. In IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers, page 34, 2005.

19. Hanene Ben Fradj, Cécile Belleudy, and Michel Auguin. Multi-bank main memory architecture with dynamic voltage frequency scaling for system energy optimization. In DSD, pages 89–96, 2006.

20. Hanene Ben Fradj, Cécile Belleudy, and Michel Auguin. System level multi-bank main memory configuration for energy reduction. In PATMOS, pages 84–94, 2006.

21. Vincent W. Freeh and David K. Lowenthal. Using multiple energy gears in mpi programs on a power-scalable cluster. In PPoPP '05: Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming, pages 164–173, 2005.

22. Vincent W. Freeh, Feng Pan, Nandini Kappiah, David K. Lowenthal, and Rob Springer. Exploring the energy-time tradeoff in mpi programs on a power-scalable cluster. In IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers, page 4.1, Washington, DC, USA, 2005.

23. Francesc Guim, Jesus Labarta, and Julita Corbalan. Modeling the impact of resource sharing in backfilling policies using the alvio simulator. In IEEE Intl. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pages 145–150, 2007.

24. Taliver Heath, Ana Paula Centeno, Pradeep George, Luiz Ramos, Yogesh Jaluria, and Ricardo Bianchini. Mercury and freon: temperature emulation and management for server systems. In Intl. Conf. on Architectural Support for Programming Languages and Operating Systems, pages 106–116, 2006.

25. Tibor Horvath and Kevin Skadron. Multi-mode energy management for multi-tier server clusters. In International conference on Parallel architectures and compilation techniques (PACT'08), pages 270–279, 2008.

26. Chung hsing Hsu and Wu chun Feng. A feasibility analysis of power awareness in commodity-based high-performance clusters. In IEEE International Conference on Cluster Computing, pages 1–10, 2005.

27. Chung-Hsing Hsu and Wu chun Feng. A power-aware run-time system for high-performance computing. In ACM/IEEE Conference on High Performance Networking and Computing (SC'05), page 1, 2005.

28. Michael C. Huang, Jose Renau, and Josep Torrellas. Positional adaptation of processors: application to energy reduction. In ISCA '03: Proceedings of the 30th annual international symposium on Computer architecture, pages 157–168, 2003.

29. Ibrahim Hur and Calvin Lin. A comprehensive approach to dram power management. In 14th International Conference on High-Performance Computer Architecture (HPCA), pages 305–316, Salt Lake Citi, UT, USA, 2008.

30. Canturk Isci, Gilberto Contreras, and Margaret Martonosi. Live, runtime phase monitoring and prediction on real systems with application to dynamic power management. In IEEE/ACM Intl. Symp. on Microarchitecture, pages 359–370, 2006.

31. Nandini Kappiah, Vincent W. Freeh, and David K. Lowenthal. Just in time dynamic voltage scaling: Exploiting inter-node slack to save energy in mpi programs. In SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing, page 33, 2005.

32. Jeffrey O. Kephart, Hoi Chan, Rajarshi Das, David W. Levine, Gerald Tesauro, Freeman Rawson, and Charles Lefurgy. Coordinating multiple autonomic managers to achieve specified power-performance tradeoffs. In Intl. Conf. on Autonomic Computing, page 24, 2007.

33. Sanjay Kumar, Vanish Talwar, Vibhore Kumar, Parthasarathy Ranganathan, and Karsten Schwan. vmanage: loosely coupled platform and virtualization management in data centers. In Intl. Conf. on Autonomic Computing, pages 127–136, 2009.

34. Hyo J. Lee, Kyu H. Lee, and Sam H. Noh. Augmenting raid with an ssd for energy relief. In 1st Workshop on Power Aware Computing and Systems (HotPower'08), co-located with OSDI 2008, San Diego, USA, 2008.

35. Xiaodong Li, Ritu Gupta, Sarita V. Adve, and Yuanyuan Zhou. Cross-component energy management: Joint adaptation of processor and memory. ACM Trans. Archit. Code Optim., 4(3):14, 2007.

36. Xiaodong Li, Zhenmin Li, Yuanyuan Zhou, and Sarita Adve. Performance directed energy management for main memory and disks. ACM Transactions on Storage, 1(3):346–380, 2005.

37. Min Yeol Lim, Vincent W. Freeh, and David K. Lowenthal. Adaptive, transparent frequency and voltage scaling of communication phases in mpi programs. In SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing, page 107, 2006.

38. Yongpeng Liu and Hong Zhu. A survey of the research on power management techniques for high-performance systems. Softw. Pract. Exper., 40(11):943–964, 2010.

39. Lauri Minas and Brad Ellison. Energy efficiency for information technology: How to reduce power consumption in servers and data centers. Intel Press, 2009.

40. Justin Moore, Jeff Chase, Parthasarathy Ranganathan, and Ratnesh Sharma. Making scheduling "cool": temperature-aware workload placement in data centers. In Annual Conf. on USENIX Annual Technical Conf., pages 5–5, 2005.

41. Justin D. Moore, Jeffrey S. Chase, and Parthasarathy Ranganathan. Weatherman: Automated, online and predictive thermal mapping and management for data centers. In Intl. Conf. on Autonomic Computing, pages 155–164, 2006.

42. Ripal Nathuji and Karsten Schwan. Virtualpower: coordinated power management in virtualized enterprise systems. In ACM SIGOPS Symp. on Operating Systems Principles, pages 265–278, 2007.

43. Alon Naveh, Efraim Rotem, Avi Mendelson, Simcha Gochman, Rajshree Chabukswar, Karthik Krishnan, and Arun Kumar. Power and Thermal Management in the Intel Core Duo Processor. Technical report, Intel Technology Journal, May 2006.

44. John Pfluenger and Sharon Hanson. Data center efficiency in the scalable enterprise. Dell Power Solutions, February 2007.

45. Eduardo Pinheiro and Ricardo Bianchini. Energy conservation techniques for disk array-based servers. In ICS '04: Proceedings of the 18th annual international conference on Supercomputing, pages 68–78, 2004.

46. Eduardo Pinheiro, Ricardo Bianchini, and Cezary Dubnicki. Exploiting redundancy to conserve energy in storage systems. SIGMETRICS Perform. Eval. Rev., 34(1):15–26, 2006.

47. Ramya Raghavendra, Parthasarathy Ranganathan, Vanish Talwar, Zhikui Wang, and Xiaoyun Zhu. No "power" struggles: coordinated multi-level power management for the data center. SIGOPS Oper. Syst. Rev., 42(2):48–59, 2008.

48. Luiz Ramos and Ricardo Bianchini. C-oracle: Predictive thermal management for data centers. In Intl. Symp. on High-Performance Computer Architecture, pages 111–122, 2008.

49. Parthasarathy Ranganathan, Phil Leech, David Irwin, and Jeffrey Chase. Ensemble-level power management for dense blade servers. SIGARCH Comput. Archit. News, 34(2):66–77, 2006.

50. Barry Rountree, David K. Lowenthal, Shelby Funk, Vincent W. Freeh, Bronis R. de Supinski, and Martin Schulz. Bounding energy consumption in large-scale mpi programs. In SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing, pages 1–9, New York, NY, USA, 2007.

51. Barry Rountree, David K. Lownenthal, Bronis R. de Supinski, Martin Schulz, Vincent W. Freeh, and Tyler Bletsch. Adagio: making dvs practical for complex hpc applications. In ICS '09 Proceedings of the 23rd international conference on Supercomputing, pages 460–469, 2009.

52. Cosmin Rusu, Alexandre Ferreira, Claudio Scordino, and Aaron Watson. Energy-efficient real-time heterogeneous server clusters. In IEEE Real-Time and Embedded Technology and Applications Symp., pages 418–428, 2006.

53. Euiseong Seo, Seon Yeong Park, and Bhuvan Urgaonkar. Empirical analysis on energy efficiency of flash-based ssds. In 1st Workshop on Power Aware Computing and Systems (HotPower'08), co-located with OSDI 2008, San Diego, USA, 2008.

54. S. Sharma, Chung-Hsing Hsu, and Wu chun Feng. Making a case for a green500 list. In Workshop on High-Performance, Power-Aware Computing (HPPAC) in conjunction with IPDPS, page 8, 2006.

55. Ying Song, Yuzhong Sun, Hui Wang, and Xining Song. An adaptive resource flowing scheme amongst vms in a vm-based utility computing. In IEEE Intl. Conf. on Computer and Information Technology, pages 1053–1058, 2007.

56. Ahmed A. Soror, Umar Farooq Minhas, Ashraf Aboulnaga, Kenneth Salem, Peter Kokosielis, and Sunil Kamath. Automatic virtual machine configuration for database workloads. In ACM SIGMOD Intl. Conf. on Management of data, pages 953–966, 2008.

57. Suresh Siddha, Venkatesh Pallipadi, Arjan Van De Ven. Getting Maximum Mileage Out of Tickless. In Ottawa Linux Symposium (OLS'07), pages 201–208, Ottawa,Ontario, Canada, Jun 2007.

58. Qinghui Tang, Sandeep Kumar S. Gupta, and Georgios Varsamopoulos. Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach. IEEE Trans. Parallel Distrib. Syst., 19(11):1458–1472, 2008.

59. Venkatesh Pallipadi and Shaohua Li and Adam Belay. Cpuidle-Do nothing efficiently... In Ottawa Linux Symposium (OLS'07), Ottawa,Ontario, Canada, Jun 2007.

60. Venkatesh Pallipadi and Suresh B Siddha. Processor Power Management Features and Process Scheduler: Do We Need to Tie Them Together? LinuxConf Europe, Sept 2007.

61. Gregor von Laszewski, Lizhe Wang, Andrew J. Younge, and Xi He. Power-aware scheduling of virtual machines in dvfs-enabled clusters. In IEEE Intl. Conf. on Cluster Computing, pages 1–10, 2009.

62. Xiaorui Wang and Ming Chen. Cluster level feedback power control for power optimization. In International Symposium on High Performance Computer Architecture(HPCA'08), pages 101–110, 2008.

63. Shu Yin, Xiaojun Ruan, Adam Manzanares, and Xiao Qin. How reliable are parallel disk systems when energy-saving schemes are involved? In IEEE International Conference on Cluster Computing and Workshops, pages 1–9, 2009.