Investigating the Potential of Application-Centric Aggressive Power Management for HPC Workloads

I. Rodero, S. Chandra, M. Parashar
Center for Autonomic Computing
Rutgers University, Piscataway NJ, USA
{irodero, sharatds, parashar}
@cac.rutgers.edu

R. Muralidhar, H. Seshadri
Intel India, Ltd
{rajeev.d.muralidhar, harinarayanan.seshadri}
@intel.com

S. Poole
Computer Science and Mathematics
& NCCS Divisions
Oak Ridge National Labs
spoole@ornl.gov

Abstract—**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. Like High Performance Computing (HPC) systems, large-scale cluster-based data centers can consume power in megawatts, and of all the power consumed by such a system, only a fraction is used for actual computations.**

In this paper, we study the potential of 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, can be effectively used in a application-aware manner for HPC workloads. To do this, we first profile standard HPC benchmarks with respect to behaviors, resource usage and power impact on individual computing nodes. Based on a power and latency model and the workload profiles, we develop an algorithm that can improve energy efficiency with little or no performance loss. We then evaluate our proposed algorithm through simulations using empirical power characterization and quantification. Finally, we validate the simulation results with actual executions on real hardware. The obtained results show that by using application aware power management, we can reduce the average energy consumption without significant penalty in performance. This motivates us to investigate autonomic approaches for application-aware aggressive power management and cross layer and cross function predictive subsystem level power management for large-scale data centers.

I. 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 clusters composed by hundreds of thousands or even millions processing cores [1] with similar power consumption concerns [2].

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. At the processor level, unlike the earlier generations of servers and HPC systems that supported only a reduced set of sleep states, current generation systems support advanced power management solutions in hardware. For example, the Intel Nehalem processor has an integrated micro-controller called Power Control Unit or PCU. It has its own embedded firmware and dynamic sensors to monitor current temperature and power in real time, and has an integrated power gate, which eliminates the problem of power leakage. 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 paper, we study the potential of 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, can be effectively used in a application-aware manner for HPC workloads. To use these mechanisms effectively we require cross layer solutions that are driven by the application and which adapt themselves according to application demands in terms of physical resources. The goal is to investigate power management strategies and their impact on the overall energy consumption in order to define an upper bound of possible energy savings and gain sufficient experience to develop an autonomic runtime to improve the energy efficiency of data centers' resources.

To do this, we firstly profile standard HPC benchmarks with respect to behaviors, resource usage and power consumption. 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 for substantial time-intervals, but still require a large amount of power.

Based on the empirical power characterization and quantification of the HPC benchmarks, we develop an analytical model that incorporates the applications resource usage patterns, where subsystems in a computing node support different power states, each with specific entry and exit latency, and energy consumption. Furthermore, we use our analysis of HPC workloads to estimate which subsystems are essential for the workload in question, and which subsystems can dynamically enter and exit lower power states (not necessarily idle). We then use the model along with simulations to investigate the potential energy saving of deterministic, application-aware, power management strategies. We then evaluate our proposed algorithm through simulations using empirical power characterization and quantification. Finally, we validate the simulation results with actual executions in real hardware. The obtained results show that by using application aware power management, we can reduce the average energy consumption without significant penalty in performance. Furthermore, the obtained results motivate us to investigate autonomic approaches for application-aware aggressive power management and cross layer and cross function predictive subsystem level power management for large-scale data centers.

The main contributions of this paper 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) formulate an energy consumption model and propose an algorithm attempting to improve energy efficiency with little or no performance loss, and (iv) quantify possible energy savings of application-centric aggressive power management through simulations and actual executions.

The rest of the paper is organized as follows. In Section II, we describe background and related work. In Section III, we quantify possible power savings and profile HPC workloads using standard benchmarks. In Section IV, we develop a power model and an algorithm for predictive and aggressive power management based on workload profiles. In Section V, we discuss our evaluation and present the obtained results. Finally, in Section VI, we conclude the paper and outline directions for future work.

II. Related Work

In recent work, Liu et al. [3] 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. It involves controlling the sleep states or the C-states [4] and the P-states of the processor when the processor is idle [5]. 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 [6]. 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 [7].

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. [8] 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. [9] 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 [10], and implemented solutions for scientific applications [11]. Freeh et al. proposed a model to predict execution time and energy consumed of an application running at lower P-states [12] and techniques based on phase characterization of the applications, assigning different P-states to phases according the previous measurements and heuristics [13]. Cameron et al. [14] 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. Horvath et al. [15] exploited DVFS with dynamic reconfiguration in multi-tier server clusters, which is a typical architecture of current server clusters. Ranganathan et al. [16] designed a cluster level power management controller, which employes a management agent running on each server and the server which exceeded the power budget according to a Service Level Agreement (SLA), is throttled down to an appropriate level. Wang et al. [17] proposed a control algorithm to manage power consumption of multiple servers simultaneously. The controller monitors the power value and CPU utilization of each server to set the frequency of the processors in a coordinated way. Weissel et al. [18] defined an energy-aware scheduling policy that benefits from event counters. By exploiting the information from these counters, the scheduler determines the appropriate clock frequency for each individual thread running in a time-sharing environment. Leveraging DVFS mechanisms, Hsu et al. [19] proposed an automatically-adapting, power aware algorithm that is transparent to end-user applications and deliver considerable energy savings with tight control over DVFS-induced performance slowdown. They used Millions of Instructions Per Second (MIPS) as a metric to measure CPU boundedness and take decisions on DVFS control, whereas Malkowski et al. [20] took advantage of memory-bound phase to select CPU frequencies. Rountree et al developed a system called Adagio to collect statistical data on task execution slacks[21], compute the desired frequency and represent the result in a hash table. When task executes again, an appropriate frequency can be found in a hash table. Other techniques based on switching on/off nodes and other networking devices have been also proposed [22]. These techniques have been also applied to virtualized environments [23] but also DVFS techniques have been exploited [24], [25].

Substantial work has been also done for adapting the RAM memory subsystem for saving energy. Delaluz et al [26], [27] studied compiler-directed techniques, as well as OS-based approaches [28], [29] to reduce the energy consumed by the memory subsystem. Huang et al. [30] proposed power-aware virtual memory implementation in OS to reduce memory energy consumption. Fradj et al. [31], [32] propose multi-banking techniques that consist of setting individually banks in lower power modes when they are not accessed. Diniz et al. [33] 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. [34] 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.

Existing research work also addresses the storage subsystem management to improve energy efficiency of servers. Rotem et al. [35] 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. [36] 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 [37]. Other approaches have addressed energy efficiency of storage systems by spinning-down/up disk [38] and the reliability of such techniques [39]. Solid State Drive (SSD) disks have been also taken into account towards saving energy consumption for the storage subsystem [40], [41].

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 [42], [43]. Li et al. [44] combine memory and disk management techniques to provide performance guarantees for control algorithms. 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. Thus, our approach is complimentary to existing and ongoing solutions for energy management for data centers.

III. Studying Energy Saving Possibilities

The fundamental requirement to study the potential energy saving with the approach suggested in this paper is to gather reliable usage data for processor, memory, storage subsystem and the NIC for a set of representative and standard HPC workloads. However, we first 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. From the profiling information we will be able to obtain patterns (coarse grain) and then identify possible opportunities of energy saving in servers for large-scale data centers.

A. Experimental Environment

The experiments were conducted 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.

B. Power Saving Quantification

In order to quantify the possible power savings of 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 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.

$$P_{cpu} \sim C \times V^2 \times \alpha \times f \tag{1}$$

Table I 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 consuming 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 off/on 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 leakage. Equation 2 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.

$$P_{static} = Vcc \times N \times k_{design} \times \hat{I}_{leak} \quad (2)$$

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 reduce dynamically 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.

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 I
SERVER’S POWER SAVINGS AND ASSOCIATED DELAYS

C. Workload Profiling

The methodology involves profiling 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. We profiled standard HPC benchmarks with respect to behaviors and subsystem usage on individual servers. 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 run two distributed instances of bonnie++ using a script and ssh.

Figure 1 shows the obtained profiles for different benchmarks, along with the server power consumption during their execution. Due to space limitations, we only show the profiles of 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), the average p-state residency of the CPU’s cores, and power consumption. 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. In the following subsection we discuss the trends and the power saving opportunities of the profiles shown in Figure 1.

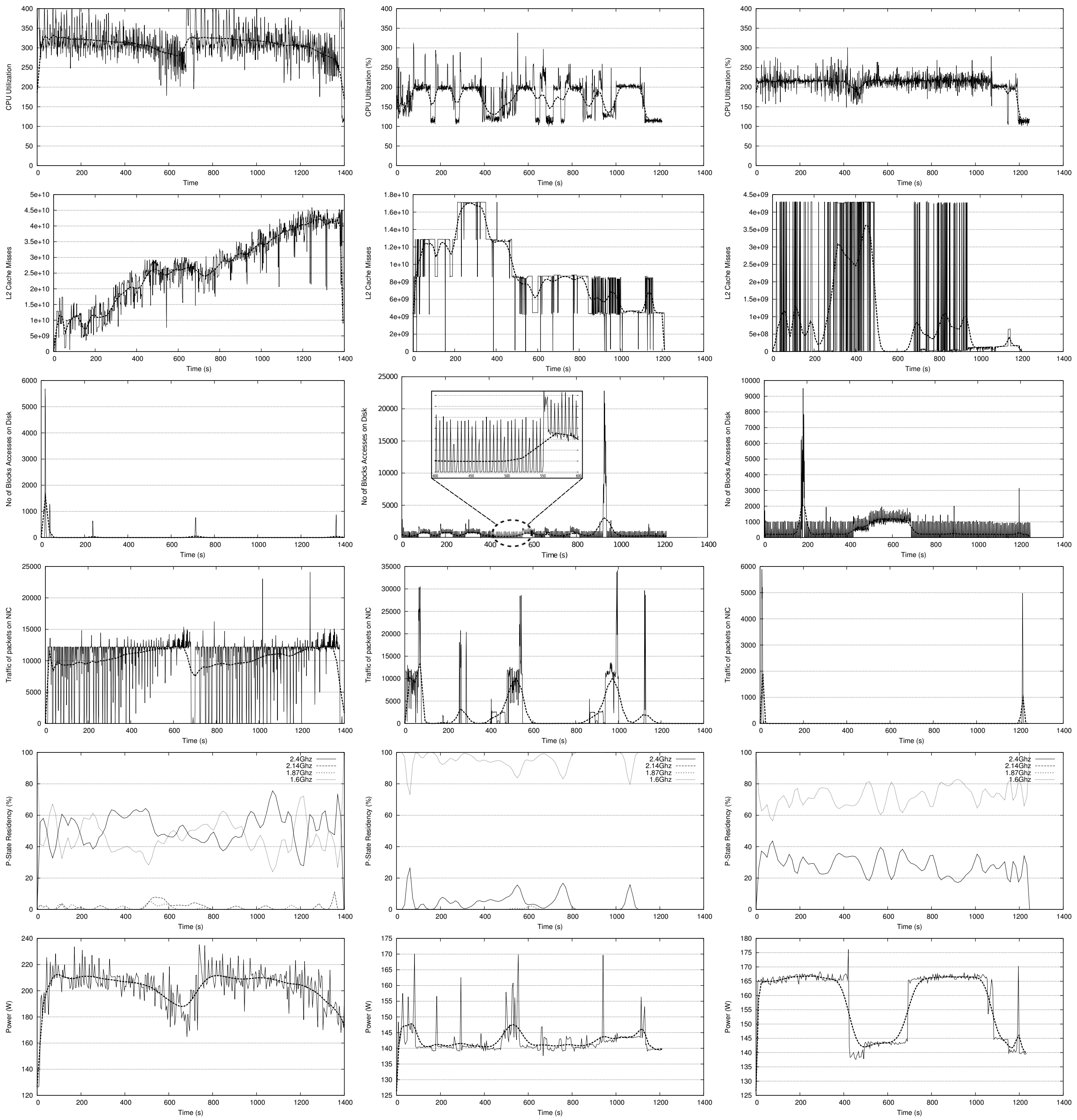

Fig. 1. Profiling information and power consumption of HPL (left), b_eff_io (center) and bonnie++ (right) benchmarks

D. Power Saving Opportunities

The plots included in Figure 1 plot different application profiles for resource utilization and power consumption. Some observations are listed below. The HPL benchmark was configured to run two problems of the same size. Thus, as is shown in the CPU utilization plot, there is an interval of time in the middle of the benchmark's execution with lower CPU utilization. It helps us to appreciate a clear correlation between the CPU utilization and power consumption. L2 cache misses increase steadily during the HPL's execution, which suggests high memory activity. The NIC is used in bursts, except at the end of the two problems solved by HPL. P-state residency is distributed almost evenly among the maximum and minimum CPU frequencies. This is explained due to the fact that the OS scales the frequency down/up dynamically following the iterative compute+synchronization pattern of HPL. Also, since the network was not very fast (100Mbps), the communication slack is significant. In contrast to the other subsystems, disk utilization is scarce. Therefore, we find higher opportunities for energy saving using disk power management techniques. With CPU intensive benchmarks (i.e. TauBench), the CPU utilization steadily remains close to the maximum utilization (400%), and the p-state residency at maximum CPU frequency is at 100% during almost the whole execution of the benchmark. This is because TauBench has much less MPI synchronization than HPL. As both b_eff_io and bonnie++ benchmarks are I/O intensive, the CPU utilization running these benchmarks is low and the p-state residency is predominated by low frequencies. However, memory and disk have high activity. In fact, the disk is accessed frequently (see the disk utilization plot for b_eff_io) thus making it infeasible to spin-up and spin-down the disk. The main difference between b_eff_io and bonnie++ is that b_eff_io has periods of intensive NIC utilization, and bonnie++ does not perform synchronization over the network. Therefore, b_eff_io has lower average CPU utilization than bonnie++, but the CPU utilization is steadier for bonnie++. Moreover, we can appreciate a clear correlation between memory utilization and power consumption with bonnie++ that is not present with b_eff_io. The NIC utilization plot for b_eff_io shows multiple long duration idle periods offering opportunities to save energy even though other subsystems are active most of the time. As a result, with bonnie++ we have higher opportunities for energy saving from NIC and memory because the network traffic is only at the beginning and at the end of its execution, and there are some long intervals of time without L2 cache misses.

IV. Towards Application-Centric Aggressive Power Management

In this section, we build a model with the goal of developing an algorithm for aggressive power management. The power dissipated by a server P can be computed as the sum of it's static and dynamic power as described in Equation 3.

$$P = P_{static} + P_{dynamic} \tag{3}$$

$P_{dynamic}$ is composed of power contributions from the CPU and sub-systems such as memory, storage and the NIC,

$$P_{dynamic} = P_{cpu} + P_{mem} + P_{disk} + P_{nic} \tag{4}$$

Since we assume that the OS manages the CPU power configuration efficiently, we do not consider CPU henceforth. We made the simplification that each subsystem has two operational modes: $active$ (regular configuration) and $idle$ (low power mode). Therefore, we consider the latencies and overheads that would be involved in switching between these subsystem power modes. We define t_{on} as the latency to switch from idle to active mode and t_{off} the latency to switch from active to idle mode. We also define $E_{t_{on}}$ and $E_{t_{off}}$ as their respective associated energy costs. Since we consider that different power configurations for the subsystems can be used during a workload execution, we model the workload execution time (T) as the sum of time intervals where each two consecutive time interval have different power configurations (Equation 5).

$$T = \sum_{i=1}^{N} t_i \tag{5}$$

The energy consumed by the different subsystems (sys={mem, disk, nic}) during the execution of a workload (E) is defined as the sum of the energy consumed in each time interval (Equation 6).

$$E = \sum_{i=1}^{N} \sum^{sys} E_{t_i}^{sys} \tag{6}$$

The energy consumed within a time interval includes the energy overhead of switching between power modes as Equation 7 shows.

$$E_{t_i}^{sys} = P_{active}^{sys} \cdot t_{i,active} + P_{idle}^{sys} \cdot t_{i,idle} + E_{t_{on/off}}^{sys} \tag{7}$$

P_{active}^{sys} is the power consumed if the subsystem sys is active and $t_{i,active}$ is the time duration in active mode in seconds. P_{idle}^{sys} is the power consumed if the subsystem sys is idle (low power mode) and $t_{i,idle}$ is the time duration in idle mode in seconds.

A. Predictive and Aggressive Power Management (PAPM)

Building on the model described above, we develop the PAPM algorithm that transitions the subsystems to the appropriate power mode based on an a-priori knowledge of the application profile. The algorithm takes into account the latency of each device in making a transition from one state to another, as well as the overhead energy consumed to make the transition. Algorithm 1 shows the PAPM algorithm focusing on switching off subsystems and deciding the appropriate low power state to transition a subsystem from an active state. The algorithm to switch back to an active state from an idle state is symmetric to Algorithm 1. It has three conditions: *Idle-Condition*, *Time-Condition* and *Energy-Condition* on which transition to a low power state is dependent.

```
Algorithm
Input: Physical memory, disk, NIC
Output: Power state for a given subsystem
for time = 0 to i do
    for sys = disk, memory, NIC do
        if Idle-Condition then
            if Time-Condition then
                if Energy-Condition then
                    $P^{sys}_{PwrSt} = P^{sys}_{idle}$;
                end
            end
            else
                $P^{sys}_{PwrSt} = P^{sys}_{active}$;
            end
        end
    end
end
```

Algorithm 1: Algorithm for idle state transition of memory, storage and NIC subsystems

Idle-Condition checks if the subsystem is going to be idle in the next time interval and it is given by,

$$WorkloadProfile(t^{sys}_{i+1}) \longrightarrow 0 \tag{8}$$

Equation 8 indicates, based on the workload profile data, that if the next time instance has low or no activity then proceed further to transition the subsystem to a low power state. *Time-Condition* is given by,

$$(t^{sys}_{i+1} - t^{sys}_{i}) > t^{sys}_{off} + t^{sys}_{on} \tag{9}$$

This condition checks if it is feasible to transition to any of the available lower power states based on the latency of the subsystems and the idle time between the two active periods. The active period is denoted by $t^{sys}_{i,active}$. If the time for which subsystem is idle is greater than the latencies to enter and exit any of the low power states, then the system should be transitioned to low power mode. For simplicity, in Algorithm 1 we only consider two possible power modes (active/idle).

Energy-Condition is given by,

$$P^{sys}_{active} \cdot t_i > P^{sys}_{idle} \cdot t_{i,idle} + E^{sys}_{t_{off}} + E^{sys}_{t_{on}} \tag{10}$$

This condition checks if it is worthwhile to transition the system to a low power state and if any energy savings will be achieved. It also takes into consideration the power to transition the subsystem from a low power mode to operating mode. If the sum of energy consumed in low power state and energy needed to bring back the subsystem to active mode is less than the energy used by subsystem when it continues to be in high power state when idle, it is worthwhile to transition the system to a low power state. Other considerations can be incorporated in this model such as ensuring that the energy saving is significant enough to over-ride the cost of reducing the lifetime of the disk.

V. Experimental Evaluation

In this section, we evaluate the energy savings that can be achieved with the PAPM algorithm proposed previously with a deterministic approach, in order to show its potential in a single server. To do this, we firstly use simulation along with the workload profile information discussed in section III-C. Then, we present results obtained from actual executions to validate the former simulations. Finally, we analyze the potential energy saving in large-scale data centers.

A. Simulation

In this subsection, we evaluate the PAPM algorithm in a single server using simulations and present the estimated energy savings. The PAPM algorithm simulation was developed using MATLAB. We used the benchmarks presented in section III that, as we discussed previously, have different requirements and behaviors in terms of subsystems utilization. Although we present the saving for a single server, the results were obtained using the testbed described in section III-A. We also perform our simulations based on the subsystem usage time obtained from the workload profile data of an individual server. Since the approach is deterministic, we assume that the workload profile is known in advance. Following the PAPM algorithm, when a subsystem is switched to another power mode (e.g. spinning up/down the disk), we consider the savings that we quantified in section III-B as well as the associated overheads in terms of delay and energy.

Table II present the obtained results. Specifically, it shows the run time, the energy consumption, and the estimated energy savings of various workloads. The granularity of the delays is in seconds due to the limitations of the instruments, and the fact that our approach is based on a coarse-grained model. It is worth noting that run time and energy consumption are obtained through actual measurements, while the energy savings for memory, disk and NIC were obtained through the simulations. We present the results for each benchmark with two different configurations: DVFS and non-DVFS. The former configuration uses ACPI enabling DVFS with “cpufreq” and the “ondemand“ governor. The latter configuration uses “userspace” governor at the maximum CPU frequency. Therefore, the CPU savings are also obtained from actual measurements. Although the total energy savings are higher considering the CPU, we will consider the savings only from the other subsystems. In fact, the results state that DVFS does not penalize the execution time significantly (0.22% on average) while it provides significant savings of energy (2.23% on average). This supports our argument that modern OS-driven power management mechanisms can be leveraged for application-aware power management, thus not requiring any specific CPU power management.

From Table II we can appreciate that CPU- and network-intensive benchmarks provide more opportunities of energy savings from the disk (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

Benchmark	DVFS	Run Time (s)	Energy (J)	Energy Savings					
				CPU	Memory	Disk	NIC	Total (J)	%
HPL	×	1,382 s	298,546 J	-	1,380 J	5,338 J	240 J	6,958 J	**2.33%**
	√	1,383 s	292,824 J	5,722 J				12,680 J	4.33%
b_eff_io	×	1,206 s	164,224 J	-	5,297 J	-	1,124 J	6,421 J	**3.90%**
	√	1,212 s	161,460 J	2764 J				9,185 J	5.69%
bonnie++	×	1,247 s	190,613 J	-	3,263 J	574 J	3,841 J	7,678 J	**4.03%**
	√	1,248 s	187,533 J	3,080 J				10,758 J	5.73%
TauBench	×	1,134 s	251,904 J	-	3,377 J	7,297 J	1,979 J	12,653 J	**5.02%**
	√	1,136 s	244,473 J	7,431 J				20,084 J	8.21%
FFTW	×	1,052 s	198,621 J	-	2,112 J	6,927 J	297 J	9,336 J	**4.70%**
	√	1,055 s	193,146 J	5,475 J				14,811 J	7.67%

TABLE II
ENERGY SAVINGS WITH PAPM USING SIMULATIONS FOR DIFFERENT BENCHMARKS

Benchmark	Configuration	Run Time (s)	%	Energy (J)	%	EDP	%
HPL	*Reference*	1,383 s	-	292,824 J	-	404,975,592	-
	PAPM	1,385 s	+0.14%	287,906 J	-1.67%	398,749,810	-1.53%
	PAPM+SSD	1,385 s	+0.14%	281,559 J	-3.84%	389,959,215	-3.70%
b_eff_io	*Reference*	1,212 s	-	161,460 J	-	195,689,520	-
	PAPM	1,217 s	+0.41%	157,335 J	-2.55%	191,476,695	-2.15%
	PAPM+SSD	1,134 s	-6.43%	143,768 J	-10.95%	163,032,912	-16.68%
bonnie++	*Reference*	1,248 s	-	187,533 J	-	234,041,184	-
	PAPM	1,249 s	+0.08%	182,904 J	-2.47%	228,447,096	-2.39%
	PAPM+SSD	1,169 s	-6.33%	168,606 J	-10.09%	197,100,414	-15.78%
TauBench	*Reference*	1,136 s	-	244,473 J	-	277,721,328	-
	PAPM	1,139 s	+0.26%	236,496 J	-3.26%	269,368,944	-3.00%
	PAPM+SSD	1,137 s	+0.08%	229,446 J	-6.14%	260,880,102	-6.06%
FFTW	*Reference*	1,055 s	-	193,146 J	-	203,769,030	-
	PAPM	1,057 s	+0.19%	187,677 J	-2.83%	198,374,589	-2.64%
	PAPM+SSD	1,051 s	-0.38%	177,071 J	-8.32%	186,101,621	-8.67%

TABLE III
ENERGY SAVINGS WITH PAPM USING ACTUAL EXECUTIONS

energy savings. Although the average energy saving is around 4%, we can infer that PAPM can improve the energy efficiency and hence the power efficiency of HPC workloads from around 2% to around 5% depending on the workload profile. However, our simulations are very conservative because we only consider static power to estimate the potential energy savings from the memory subsystem (our quantification was done with the server under idle condition). In fact, the simulations may obtain higher energy savings without significant penalty in run time if we consider additional memory management techniques, such as Dynamic Voltage Scaling (DVS), during intervals of time where memory is not in the critical path.

B. Validation

In order to validate our model we implemented the PAPM algorithm and performed actual experiments. They provide us with a very good opportunity to prove the effectiveness of PAPM. Since we do not support any mechanism to reduce the voltage of RAM memory or switching off banks of memory using multi-banking techniques, we focus on in disk storage and NIC. Specifically, we used the following configurations:

- *Reference*: regular execution with DVFS enabled.
- *PAPM*: implementation of PAPM with a script that deterministically switches to the appropriate power state, based on the profile obtained in previous executions of the same benchmark.
- *PAPM+SSD*: use of SSD technology for storage and PAPM algorithm only for NIC.

Table III present the obtained results. It shows the run time and the difference with respect to *Reference* configuration, the energy consumption and the difference with respect to *Reference* configuration and, in order to consider both energy and performance, the Energy Delay Product (EDP) and the difference with respect to *Reference* configuration. With *PAPM* the overheads of switching the power modes do not penalize the run time significantly (0.16%, on average). However, the average energy saving is around 2.5% which is 37.5% lower with respect to the average energy saving estimated through simulations. The percentage of savings for EDP is only slightly lower than the energy savings. We believe that the differences between the results obtained with simulations and with actual experiments are motivated by two factors. On the one hand, the lack of memory power management and, on the other hand, insufficient precision at some power mode modification times that may result in some energy saving loss.

The *PAPM+SSD* configuration allows us to identify the potentials and trade offs of using SSD technology. Since SSD disks do not require spinning down techniques to switch to low power mode, the PAPM algorithm only focus on NIC (potentially it may consider also memory). As is shown in Table III using SSD technology reduces the energy consumption 7.86% on average. For non-disk intensive benchmarks (e.g. HPL) the savings are moderate (5%, on average) while the energy savings are much higher for I/O intensive benchmarks (around 10%, on average). The run times obtained with *PAPM* and with the other configurations are similar for the non-disk intensive benchmarks. However, with I/O intensive benchmarks (b_eff_io and bonnie++) the run times are reduced up to 6.43%. This is explained due to the fact that both b_eff_io and bonnie++ benchmarks access the disk frequently (see Figure 1) and SSD drives have a much shorter access time. As SSD technology is still very expensive, we can assume that only some servers of a data center will have a SSD disk available (at least for cache data). Hence, the problem will include mapping the applications that can take more advantage of SSD technology to the nodes with a SSD installed, depending of the application profile. Therefore, servers with SSD disks may be assigned to I/O intensive applications while non-disk intensive applications (e.g. HPL) may take advantage of power management techniques on traditional disks (spinning dow/up), if the benefit of using SSD does not compensate the cost of SSD technology. It is worth noting that this trade off will also depend of the estimated workload execution time.

C. Scaling to Large HPC Cluster Level

PAPM does power savings on per node basis per run basis, which can produce significant amount of energy savings if scaled to data center magnitudes. We assume that the data center is composed by the server configuration used throughout this paper, the average energy savings obtained from our validation experiments (5.21%) and an average power demand of 200W due to almost continuous load. If the daily energy consumption is $200W \times 24h \times 3,600s = 0.48kWh$, the daily energy saving is around $0.25kWh$. Thus, the yearly energy saving will be $0.25kWh \times 365days = 91.28kWh$ per node. Considering that the average kWh price (March 2010) in United States is $\$0.125$ and €0.36 in Europe, the yearly saving will be $\$11.41$ or €32.86 per node. For a 1,000 node data center it will save approximately $\$11,410$ or €$32,860$ per year only on computational costs. As the size of HPC cluster increases the savings would increase by same order of magnitude.

VI. Conclusions and Future Work

In this paper, we studied the potential impact of deterministic application-centric power control at the device level on the overall energy efficiency of the system. Specifically, we analyzed the power consumption of a node according to the usage of its processor, memory and storage subsystem, and the NIC. Based on a power and latency model and workload profiling, we have developed the PAPM algorithm, which attempts to improve energy efficiency with little or no performance loss. We evaluated our approach with simulations and actual executions to validate them. The simulations used empirical data that was obtained from executions in real hardware to quantify the potential energy savings. The results stated that using application-aware subsystem power control can save additional energy without significant penalty in performance. Furthermore, the results showed that combining PAPM with using low power devices (e.g. SSD technology) can increase remarkably the potential to improve the energy efficiency of the system. It allows cross layer optimizations, such as application-aware mapping across the data center.

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. It motivates us to investigate autonomic approaches for application-aware aggressive power management rather than the deterministic approach presented in this paper. Since HPC applications usually follow iterative patterns, we plan to profile the applications on run time and, then predict the optimal subsystems' power configuration to be applied for the following iterations. We also can conclude that current and ongoing technologies such as SSD disk drives or memories that allow DVS must be adopted and supported in large-scale data centers to enhance global energy optimizations. In conjunction with application aware predictive algorithms we also propose a cross layer and cross function predictive subsystem level power management system as future research directions.

Acknowledgment

The research presented in this paper is supported in 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 National Science Foundation, 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 National Science Foundation.

References

[1] J. 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*, 2009.

[2] "Mission possible – greening the HPC data center," *HPC Wire*, 2009.

[3] Y. Liu and H. Zhu, "A survey of the research on power management techniques for high-performance systems," *Softw. Pract. Exper.*, 2010.

[4] A. Naveh, E. Rotem, A. Mendelson, S. Gochman, R. Chabukswar, K. Krishnan, and A. Kumar, "Power and thermal management in the Intel Core Duo processor," Intel Technology Journal, Tech. Rep., 2006.

[5] V. Pallipadi, S. Li, and A. Belay, "cpuidle-Do nothing efficiently..." in *Ottawa Linux Symposium (OLS'07)*, 2007.

[6] V. Pallipadi and S. B. Siddha, "Processor power management features and process scheduler: do we need to tie them together?" LinuxConf Europe, 2007.

[7] S. Siddha, V. Pallipadi, and A. V. D. Ven, "Getting maximum mileage out of tickless," in *Ottawa Linux Symposium (OLS'07)*, 2007, pp. 201–208.

[8] N. Kappiah, V. W. Freeh, and D. K. Lowenthal, "Just in time dynamic voltage scaling: exploiting inter-node slack to save energy in MPI programs," in *ACM/IEEE conference on Supercomputing (SC'05)*, 2005, p. 33.

[9] M. Y. Lim, V. W. Freeh, and D. K. Lowenthal, "Adaptive, transparent frequency and voltage scaling of communication phases in MPI programs," in *ACM/IEEE conference on Supercomputing (SC'06)*, 2006, p. 107.

[10] B. Rountree, D. K. Lowenthal, S. Funk, V. W. Freeh, B. R. de Supinski, and M. Schulz, "Bounding energy consumption in large-scale MPI programs," in *ACM/IEEE conference on Supercomputing (SC'07)*, 2007, pp. 1–9.

[11] B. Rountree, D. K. Lownenthal, B. R. de Supinski, M. Schulz, V. W. Freeh, and T. Bletsch, "Adagio: making DVS practical for complex HPC applications," in *23rd international conference on Supercomputing (ICS'09)*, 2009, pp. 460–469.

[12] V. W. Freeh, F. Pan, N. Kappiah, D. K. Lowenthal, and R. Springer, "Exploring the energy-time tradeoff in MPI programs on a power-scalable cluster," in *19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, 2005, p. 4.1.

[13] V. W. Freeh and D. K. Lowenthal, "Using multiple energy gears in MPI programs on a power-scalable cluster," in *ACM SIGPLAN symposium on Principles and Practice of Parallel Programming (PPoPP'05)*, 2005, pp. 164–173.

[14] K. W. Cameron, R. Ge, and X. Feng, "High-performance, power-aware distributed computing for scientific applications," *Computer*, vol. 38, no. 11, pp. 40–47, 2005.

[15] T. Horvath and K. Skadron, "Multi-mode energy management for multi-tier server clusters," in *International conference on Parallel Architectures and Compilation Techniques (PACT'08)*, 2008, pp. 270–279.

[16] P. Ranganathan, P. Leech, D. Irwin, and J. Chase, "Ensemble-level power management for dense blade servers," *SIGARCH Comput. Archit. News*, vol. 34, no. 2, pp. 66–77, 2006.

[17] X. Wang and M. Chen, "Cluster level feedback power control for power optimization," in *IEEE 14th International Symposium on High Performance Computer Architecture (HPCA)*, 2008, pp. 101–110.

[18] A. Weissel and F. Bellosa, "Process cruise control: event-driven clock scaling for dynamic power management," in *International conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES'02)*, 2002, pp. 238–246.

[19] C.-H. Hsu and W.-C. Feng, "A power-aware run-time system for high-performance computing," in *ACM/IEEE conference on Supercomputing (SC'05)*. IEEE Computer Society, 2005, p. 1.

[20] K. Malkowski, P. Raghavan, M. Kandemir, and M. J. Irwin, "Phase-aware adaptive hardware selection for power-efficient scientific computations," in *International Symposium on Low Power Electronics and Design (ISLPED'07)*, 2007, pp. 403–406.

[21] B. Rountree, D. K. Lownenthal, B. R. de Supinski, M. Schulz, V. W. Freeh, and T. Bletsch, "Adagio: making DVS practical for complex HPC applications," in *23rd International conference on Supercomputing (ICS'09)*, 2009, pp. 460–469.

[22] T. Heath, B. Diniz, E. V. Carrera, W. M. Jr., and R. Bianchini, "Energy conservation in heterogeneous server clusters," in *ACM SIGPLAN symposium on Principles and Practice of Parallel Programming (PPoPP'05)*, 2005, pp. 186–195.

[23] A. Verma, P. Ahuja, and A. Neogi, "pMapper: power and migration cost aware application placement in virtualized systems," in *9th ACM/IFIP/USENIX International Conference on Middleware (Middleware'08)*, 2008, pp. 243–264.

[24] R. Nathuji and K. Schwan, "VirtualPower: coordinated power management in virtualized enterprise systems," in *ACM SIGOPS Symposium on Operating Systems Principles (SOSP'07)*, 2007, pp. 265–278.

[25] G. Laszewski, L. Wang, A. J. Younge, and X. He, "Power-aware scheduling of virtual machines in DVFS-enabled clusters," in *IEEE International Conference on Cluster Computing and Workshops*, 2009, pp. 1–10.

[26] V. Delaluz, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin, "Energy-oriented compiler optimizations for partitioned memory architectures," in *International conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES'00)*, 2000, pp. 138–147.

[27] V. Delaluz, M. Kandemir, N. Vijaykrishnan, A. Sivasubramaniam, and M. J. Irwin, "Hardware and Software Techniques for Controlling DRAM Power Modes," *IEEE Trans. Comput.*, vol. 50, no. 11, pp. 1154–1173, 2001.

[28] V. Delaluz, M. Kandemir, and I. Kolcu, "Automatic data migration for reducing energy consumption in multi-bank memory systems," in *39th annual Design Automation Conference (DAC'02)*, 2002, pp. 213–218.

[29] V. Delaluz, A. Sivasubramaniam, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin, "Scheduler-based DRAM energy management," in *39th annual Design Automation Conference (DAC'02)*, 2002, pp. 697–702.

[30] M. C. Huang, J. Renau, and J. Torrellas, "Positional adaptation of processors: application to energy reduction," in *30th annual International Symposium on Computer Architecture (ISCA'03)*, 2003, pp. 157–168.

[31] H. B. Fradj, C. Belleudy, and M. Auguin, "System level multi-bank main memory configuration for energy reduction," in *International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2006, pp. 84–94.

[32] H. B. Fradj, C. Belleudy, and M. Auguin, "Multi-bank main memory architecture with dynamic voltage frequency scaling for system energy optimization," in *Euromicro Conference on Digital System Design (DSD)*, 2006, pp. 89–96.

[33] B. Diniz, D. Guedes, W. Meira, Jr., and R. Bianchini, "Limiting the power consumption of main memory," in *34th annual International Symposium on Computer Architecture (ISCA'07)*, 2007, pp. 290–301.

[34] I. Hur and C. Lin, "A comprehensive approach to DRAM power management," in *14th International Conference on High-Performance Computer Architecture (HPCA)*, 2008, pp. 305–316.

[35] D. Rotem, E. Otoo, and S.-C. Tsao, "Analysis of trade-off between power saving and response time in disk storage systems," *Fifth Workshop on High-Performance Power-Aware Computing (HPPAC'09) with IPDPS'09*, 2009.

[36] E. Pinheiro and R. Bianchini, "Energy conservation techniques for disk array-based servers," in *18th International conference on Supercomputing (ICS'04)*, 2004, pp. 68–78.

[37] E. Pinheiro, R. Bianchini, and C. Dubnicki, "Exploiting redundancy to conserve energy in storage systems," *SIGMETRICS Perform. Eval. Rev.*, vol. 34, no. 1, pp. 15–26, 2006.

[38] D. Colarelli and D. Grunwald, "Massive arrays of idle disks for storage archives," in *ACM/IEEE conference on Supercomputing (SC'02)*, 2002, pp. 1–11.

[39] S. Yin, X. Ruan, A. Manzanares, and X. Qin, "An energy-efficient reliability model for parallel disk systems," in *IEEE International Conference on Cluster Computing and Workshops*, 2009, pp. 1–9.

[40] E. Seo, S. Y. Park, and B. 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*, 2008.

[41] H. J. Lee, K. H. Lee, and S. 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*, 2008.

[42] X. Li, R. Gupta, S. V. Adve, and Y. Zhou, "Cross-component energy management: joint adaptation of processor and memory," *ACM Trans. Archit. Code Optim.*, vol. 4, no. 3, p. 14, 2007.

[43] Y. Cho and N. Chang, "Memory-aware energy-optimal frequency assignment for dynamic supply voltage scaling," in *International Symposium on Low Power Electronics and Design (ISLPED'04)*, 2004, pp. 387–392.

[44] X. Li, Z. Li, Y. Zhou, and S. Adve, "Performance directed energy management for main memory and disks," *ACM Transactions on Storage*, vol. 1, no. 3, pp. 346–380, 2005.