RADAR

Oxford Brookes University – Research Archive and Digital Asset Repository (RADAR)

Liu, Y and Zhu, H
A Survey of Research on Power Management Techniques for High Performance Systems.

Liu, Y and Zhu, H (2010) A Survey of Research on Power Management Techniques for High Performance Systems. *Software: practice and experience*, 40 (11). pp. 943–964.
doi: 10.1002/spe.952

This version is available: http://radar.brookes.ac.uk/radar/items/db4b664a-02cf-c9e7-3fd5-ee6fc8b8feb7/1/

Available in the RADAR: January 2011
Copyright © and Moral Rights are retained by the author(s) and/ or other copyright owners. A copy can be downloaded for personal non-commercial research or study, without prior permission or charge. This item cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder(s). The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holders.

This document is the preprint version of the journal article. Some differences between the published version and this version may remain and you are advised to consult the published version if you wish to cite from it.

www.brookes.ac.uk/go/radar

OXFORD BROOKES UNIVERSITY

Directorate of **Learning Resources**

A Survey of the Research on Power Management Techniques for High Performance Systems

Yongpeng Liu* and Hong Zhu**

*School of Computer, National University of Defense Technology, Changsha, China, liuyp@nudt.edu.cn

**School of Technology, Oxford Brookes University, Oxford, UK, hzhu@brookes.ac.uk

Abstract

This paper surveys the research on power management techniques for high performance systems. These include both commercial high performance clusters and scientific high performance computing (HPC) systems. Power consumption has rapidly risen to an intolerable scale. This results in both high operating costs and high failure rates so it is now a major cause for concern. It is imposed new challenges to the development of high performance systems. In this paper, we first review the basic mechanisms that underlie power management techniques. Then we survey two fundamental techniques for power management: metrics and profiling. After that, we review the research for the two major types of high performance systems: commercial clusters and supercomputers. Based on this, we discuss the new opportunities and problems presented by the recent adoption of virtualization techniques, and again we present the most recent research on this. Finally, we summarise and discuss future research directions.

Key Words: High performance systems, Commercial clusters, Power management, Metrics of power consumption efficiency, Energy consumption profiling, High performance computing, Virtual machine.

1. Introduction

In the community of high performance computing (HPC), the word "performance" is synonymous with "speed". With greater demands on performance, the scale of these systems becomes larger and larger, and the integration density of chipsets increases drastically. Consequently, there is a so-called "*Moore's law for power consumption*", that the power consumption of each computer nodes doubles every 18 months [1].

The power consumptions are already tremendous. According to the recent TOP500 list of high performance systems, the most power-consuming supercomputer runs at 6.95

megawatts (MW), and Roadrunner, No.1 in the TOP500 list, runs at 2.48 MW [2]. In 2006, U.S. servers and data centers consumed around 61 billion kilowatt hours (kWh) at a cost of about 4.5 billion US Dollars. This is about 1.5% of the total U.S. electricity consumption or the output of about 15 typical power plants [3]. Many data center projects have been cancelled or delayed because of an inability to meet such enormous power requirements. High density power consumption also causes overheating so energy must also be spent on cooling. For example, 0.7 W of cooling is needed to dissipate every 1.0 W of power consumed by one HPC system at Lawrence Livermore National Laboratory (LLNL) [4]. Construction costs are also increased. For example, statistical data from American Power Conversion (APC) shows that 63% of the infrastructure cost is used for power supply and cooling [5].

High power consumption naturally causes huge environment pollution. According to the U.S Environmental Protection Agency (EPA), each 1000 kWh of energy consumption generates 0.72 tons CO_2 emission. This means that U.S data centers contributed about 44 million tons of CO_2 emission in 2006, equivalent to the output of about 8 million passenger vehicles, or the carbon sequestered by 1.12 billion tree seedlings grown for 10 years [3, 6]. The EPA has therefore appealed to the government to prevent the environment pollution caused by data centers. Finally, there is also a cost to the reliability and availability of the system. Feng used empirical data from leading vendors to calculate that the failure rate of a computing node doubles with every 10^oC increase in temperature [1].

So power consumption is now a major area of concern for researchers and leading vendors. Nearly half of the TOP500 supercomputers report their power usage alongside their performance [2]. The notion of power-aware computing emerged in 1990s in the different context of mobile or embedded systems. These usually run on batteries, so energy consumption ultimately dictates their availability [3, 7]. Nowadays, power management has also become essential for high performance systems.

In addition, high performance systems are inefficient in their energy consumption. Ge *et al*. studied five supercomputers and observed that the average performance of these systems is only 54-71% of the peak performance on the optimized benchmark package. It is even worse on actual scientific applications, where the performance is only about 10% of the peak [8]. That inefficiency is mainly caused by an unequal distribution between the nodes in the cluster of the various computing, communication and I/O activities. During idle or slack times, faster components waste their energy by waiting for slower components. They could be slowed down or even shut down to save energy. Better power management could help a lot with this. The past few years have seen much research in this area, research which we shall now review.

The remainder of the paper is organized as follows. Section 2 introduces the basic

mechanisms that underlie power management techniques for high performance systems. Section 3 reviews the feasible metrics of power efficiency. Section 4 discusses issues related to the profiling of power consumption. Sections 5 and 6 review the power management techniques for commercial high performance clusters and for supercomputers, respectively. These are different because the architectures and application domains are different. Section 7 discusses the new challenges introduced by the use of virtual machines. Section 8 concludes the paper with a summary and a discussion of future directions.

2. Power Management Mechanisms

There are two different types of mechanism for power management.

(1) Dynamic Speed Scaling (DSS).

DSS dynamically changes the performance state of the target component to save power, i.e. it slows it down to reduce power consumption and speeds it up when needed but at the cost of greater power consumption.

One typical example is *Dynamic Voltage Frequency Scaling* (DVFS). Here, the reduction in power consumption is made by reducing the supply voltage or clock frequency [7]. Most existing processors allow this. Examples include Intel Xeon, AMD Atholon, and ATI co-processors. Thermal throttling is another example. Here, it is the processor's temperature that is controlled. This is done, as before, by modulating the duty cycle of the processor clock or by reducing the operating frequency and voltage of the processor [9].

Other examples of DSS include multi-frequency memories, which dynamically scale the working frequency and thus the data rate [10], and multi-speed disks [11, 12, 13].

However, in all of these methods, the transition between different performance states consumes additional energy and causes latency overhead [7, 14].

(2) Dynamic Resource Sleeping (DRS).

DRS dynamically hibernates components to save energy and then wakes them on demand. So each component may be in an active running state, or in one of several sleep states or in the power-off state. In the ACPI industry standard, the active state is called C0 and the sleep states called C1, C2…Cn [14]. For instance, in the Intel Nehalem-EP processor, the transition is made by turning off the clock and other components [9]. Each sleep state consumes less power than idling in the C0 state. The deeper the processor sleeps, the less power it consumes, but the more energy is needed to wake it up [15]. Memory controllers and chipsets may dynamically switch a memory rank's power on and off [10, 16]. Disks may also support active, ready and standby states [17, 18]. So may

I/O buses and other devices [14, 19]. In fact, the entire computer can also be managed as a DRS component with active, suspended, hibernated, and powered off states [20, 21]. Each state transition consumes energy though and takes time. [14, 15].

3. Power Efficiency Metrics

Metrics are essential for us to quantitatively measure and thereby evaluate the efficiency of energy consumption. They form the basis for decision making. In the past few years, many such metrics have been proposed and used. They can be classified into three types: (a) Metrics for solo equipment and devices, (b) Metrics for parallel systems, (c) Metrics for cluster systems.

3.1. Metrics for solo equipment and devices

The most basic metric of power efficiency, borrowed from the circuit design community, is the formula ED^n [22]. Here, E is the energy consumed while running an application, D is the time taken to complete the application, and n is a non-negative integer parameter to characterize the trade-off between E and D. It combines the cost measurements in two different dimensions, energy and time.

ED2P, the special case of ED^n when n=2, was suggested by Martonosi *et al*. [23] for use with DVFS. It cancels out the influence of frequency scaling because E is proportional to the square of the frequency, whereas D^2 is proportional to the *inverse* of the square of the frequency. ED2P considers both performance and energy consumption, but it does not consider the different requirements for different systems. Based on ED2P, Ge *et al*. [8] proposed *Weighed ED2P* as a more general metric:

$$\text{Weighted ED2P} = E^{(1-\partial)} \times D^{2(1+\partial)} \qquad (1)$$

Here, $|\partial| \leq 1$ is a weight factor determined by user preference. This metric is believed to be weighted in favor of performance when $0<\partial$, and in favor of energy when $\partial<0$. If $\partial = 0$, the performance and the energy are treated equally, and the metric reduces to ED2P. Their own experiments used $\partial = 0.2$.

3.2. Metrics for parallel systems

In [24], Hsu *et al*. argued that using a large value of n in ED^n produces a bias in favour of massively parallel systems. In fact, the reciprocal variant of ED^n, i.e. $1/ED^n$, means $performance^n/power$, or $FLOPS^n/W$, where *FLOPS* stands for floating point operations per second. When a supercomputer has s processors and each processor can deliver F flops at P watts, the $FLOPS^n/W$ metric can be re-expressed as follows.

$$\text{FLOPS}^n/\text{W} = \frac{(s \cdot F)^n}{s \cdot P} = s^{n-1} \cdot \frac{F^n}{P} \qquad (2)$$

Hsu *et al*. ranked the top 500 supercomputers with respect to their power efficiency. Their list is called the Green500. (They use $n=1$ because with $n>1$, the metric increases exponenetially with the number of processors s and thus is highly misleading.) For No. 1 on the Green500 list, ranked as No. 422 on the TOP500, the *MFLOPS/W* measure, which actually reduces to F/P, was 536.24. For No. 500, ranked as No. 311 on the TOP500, the measure was 13.03. For No. 1 on the TOP500 list, ranked as No. 4 in Green500, the measure was 444.94 [25]. Notably, the Green500 ranking of supercomputers using (2) with n=1 matches fairly well with the TOP500 [26].

Energy consumption is usually not a linear function of the performance, so Choi *et al*. [27] calculated a relative performance slowdown δ below

$$\delta = \frac{T(f)}{T(f_{\max})} - 1, \qquad (3)$$

where f is the working frequency, f_{max} is the peak CPU frequency, $T(f)$ is the execution time to complete the workload at frequency f. From this, the power efficiency gain can more accurately be calculated, Hsu *et al*. [28] and Ge *et al*. [29] in power management.

Feng *et al*. [1, 24] are concerned with the *total cost of ownership* (TCO). TCO consists of two parts: cost of acquisition and cost of operation. The former is the one-time cost of acquiring a computer system. The latter is a recurring cost made up of multiple components, such as system administration, power consumption, cooling and space. TCO is often quite difficult to calculate though, so they used metrics that are related to TCO, such as performance/power [1]. However, although TCO was initially proposed for supercomputers, it is also used for cluster systems [30].

3.3. Metrics for cluster systems

For commercial servers and data centers, the performance is indicated by the information service output, which is a complex mix of computational processing, data storage and network communication [3]. In order to test the power consumed by servers and data centers when running a full range of applications, Standard Performance Evaluation Corporation (SPEC) set up an industry-standard power-performance benchmark, SPECpower_ssj2008 [31].

For data centers or HPC centers, the power consumption includes not only the power consumed by computer systems but also cooling, lighting, power supplying and all the other facilities. Green Grid [32] proposed the *Power Usage Efficiency* (PUE) metric and its reciprocal, *Datacenter Infrastructure Efficiency* (DcIE), to estimate the energy efficiency of data centers. The PUE metric is defined as follows.

PUE = *Total Facility Power/IT Equipment Power*, (4)

where *Total Facility Power* is the total power dedicated to the whole data center. *IT Equipment Power* is the part used to support all IT equipment in a data center, including both the functional equipment for computing, storage and networking, and the supplemental equipment such as monitors and computers used for monitoring or controlling the data center. The PUE ranges from 1.0 to infinity. A PUE value approaching 1.0 indicates 100% efficiency. This goal has been accepted by many organizations, such as Microsoft, Google, Intel and IBM. Most data centers have values around 3.0, but this can be improved to 1.6 or better with proper design. For example, the data center at Lawrence Berkley National Labs (LBNL) has a PUE value of 1.3 [32].

In addition to the PUE metric, also known as *Site Infrastructure Energy Efficiency Ratio* (SI-EER), the Uptime Institute also uses two other metrics for data centers [33]. One is *IT Productivity per Embedded Watt* (IT-PEW). The other is *Data Center Energy Efficiency and Productivity* (DC-EEP) *Index.* They are defined by the following equations, respectively.

IT-PEW = *IT Productivity/ Embedded Watt* (5)

DC-EEP Index = *IT-PEW/SI-EER = IT Productivity/Total Facility Power* (6)

where *IT Productivity* is the IT service output of the data center, and *Embedded Watt* is a synonym of IT Equipment Power. The IT-PEW metric indicates the power efficiency of IT equipment, while the DC-EEP Index indicates that of the whole data center.

Lim *et al.* [30] created a benchmark with four types of workload to represent the different services in the internet sector of data centers. They used performance per watt (Perf/W), performance per TCO (Perf/TCO-$) and performance per cost of power and cooling (Perf/P&C-$) as their power efficiency metrics in the tests.

4. Profiling Power Consumption

An essential requirement of power management is that the energy usage patterns of the target system be captured precisely [34]. In other words, we need to know where and when the energy has been spent and who is responsible [35]. Building such a profile is called power consumption profiling in the sequel. A number of profiling techniques have been developed. These include the use of simulations, analytical modeling, direct online power consumption measurement, monitoring-based power consumption analysis, power behavior sampling and software instrumentation.

In the *simulation* approach, the power consumption features are embedded into a

traditional hardware simulator based on the power characteristics derived from measurement of sample components [36]. These simulators estimate both performance and power consumption by tracking the running of applications. Simulation has already been employed successfully in profiling power consumptions of microprocessors [37], memories [38], disks [11, 17] and complete machines [39]. The simulation approach is helpful for profiling the power-activity, but there may be a mismatch between the simulation and the real systems due to the inaccuracy of the simulation models [34]. Moreover, simulation is time-consuming.

Analytical modeling techniques abstract the power consumption as analytic functions on a set of parameters. They estimate power consumptions based on the dependency correlations of power on given variables. Pinheiro *et al.* [18] modeled storage systems comprised of disks with redundancy configuration. The power-related parameters are: (a) the powers of each disk in active ready or standby mode, (b) the energies and times taken by a disk to spin up and down. Given the parameters of the disks and their other characteristics, the model predicts the power behavior of the storage systems from the redundancy configuration, the disk management policy and the workloads. Heath *et al.* [40] used the following linear equations (7) to model the power consumed by each node P_i:

$$p_i = B_i + \sum_r M_i^r \times \left(\frac{U_i^r}{C_i^r} \right) \tag{7}$$

where B_i is the power consumption of node i while it is idle, M_i^r is the power consumption of resource r at node i at full utilization, U_i^r is the utilization of resource r and C_i^r is the capacity of resource r. So the power consumed by a cluster is the sum of the power consumed by all hardware resources in the system.

Analytical modeling can also be employed in thermal profiling. Heath *et al*. [41] developed a software suite, called Mercury, for temperature emulation, which computes the temperature based on their power model and the correlation between heat and power consumption. Skdaron *et al.* [42] also developed an architecture-level thermal model based on the resistances and capacitances (RC) of architectural components. Analytical modeling has the advantage of being an all-software solution, but it is coarse-grained and its accuracy depends on the granularity of the model.

Online measurement is a direct solution to the profiling problem. It employs additional measurement tools, such as multimeters, to measure and record power consumption at run-time [43]. A key issue addressed in [34, 44] is the choice of an appropriate application to use as a representative benchmark. However, this approach is impractical for most systems, as the thousands of target components involved would require larger numbers of multimeters.

Monitoring-based power consumption analysis employs performance monitoring counters (PMC), which are embedded into current commodity processers. The technique has been applied successfully in power consumption profiling. Bellosa *et al*. [35, 45, 46, 47] define computation activities as PMC events that contribute significantly to power consumption, such as clock cycles, retired instructions and memory references. The power model is a linear combination of n types of PMC events. The coefficients $V=[V_1, V_2, .., V_n]$ of the model are derived during the training period. For an application software with activity profile $E=[e_1, e_2, \ldots, e_n]$, where e_i is the number of occurrences of event i during the execution, the energy consumption can be predicted by

$$E \cdot V = \sum_{i=1}^{n} e_i \times V_i \,, \qquad (8)$$

This is employed for power and energy management [35, 45, 46, 48, 49] as well as thermal management [47]. However, the accuracy of PMC-based profiling is limited by the types of events that can be monitored by PMC. When embedded in a processor, PMCs are less accurate for describing operations outside the processor, such as direct memory access (DMA) activities. One way to improve this is to employ platform monitors, such as temperature sensors. However, the capability of platform sensor is very limited, too, and the access to these sensors through I^2C or SM buses is usually very slow, taking miliseconds rather than microseconds.

Sampling techniques sample system power consumptions and execution behaviors to find the correlation between them. Flinn *et al*. [43] designed a time-driven sampling tool, called PowerScope. It periodically measures the power consumption along with the program counter (PC) and process ID (PID), and links the sampled power consumptions back to the PIDs (i.e. the processes) and the PCs (i.e. execution phases). Chang *et al.* [50], in contrast, sample the power when the energy consumption changes. Sampling techniques do not need system-dependent hardware components, but it is then difficult to determine which components are using the most power, and the sampling may not be frequent enough.

The *Instrumentation* approach inserts additional pieces of code into the target applications to collect power information and program execution context. Hu *et al.* [51] used a compiler to insert profiling code into the assembly code that measures power consumption. Isci *et al.* [52] used an instrumentation tool to dynamically insert profiling routines into the binary code to collect the values of PC, PMC and power measurement. Instrumentation can be used to build a fine-grained power model, but it requires either a special purpose compiler or other tool support to modify the binary or source code.

5. Power Management for Commercial Clusters

Three goals of power management for commercial clusters, such as data centers and server farms, are improving the energy efficiency, capping the power consumption and throttling the heat dissipation. One key feature of commercial clusters is the amount of interactive transaction processing. So the workloads for transaction-oriented clusters vary significantly depending on the time of day or the day of the week, and there is much scope for reducing nodes below peak capacity to save power [16, 53].

In general, the power and energy management techniques in data centers and server farms can be classified into single-node techniques and cluster techniques [54, 55]. Single-node techniques are beyond the scope of this paper, so we focus on the power management techniques for clusters.

5.1 Improving Energy Efficiency

Noticing the dynamic variation of workload, Pinheiro *et al*. [21] proposed a *load concentration* policy to manage cluster-wide power consumption. This turns nodes in a cluster on or off dynamically, according to the workload imposed on the system, to make sure the expected performance is just about acceptable. When the load is lighter, some nodes are turned off dynamically to save energy. After a node is added or removed, the forthcoming requests need to be redistributed to balance the workload across the active nodes. Chase *et al.* [20] introduced a similar energy-conscious cluster-wide resource management technique, called Muse. It is based on an economic model in which the amount of resources is a function of service quality and it uses the *Service Level Agreement* (SLA) to make dynamic tradeoffs between service quality and energy consumption. In this way, it allocates a suitable share of server resources to each customer's service requests and it dynamically changes the active node set for the service in a way that maximizes the resource efficiency and minimizes unproductive cost. The idle servers then enter low-power states to conserve the energy and cooling costs.

Chen *et al.* [56] also used the SLA to direct the tradeoffs between performance and energy consumption. They noticed the time overhead of turning off/on nodes, the energy consumption in the transition and the effect of start-stop cycle on reliability. So they combined processor DVFS with dynamic cluster reconfiguration. Every T minutes, they regulate the allocation of servers by turning them on or off to accommodate the time-varying workload. Likewise, every t minutes, where $t \leq T$ and T is an integer multiple of t, they adjust each active server to an appropriate frequency. However, they only considered homogeneous clusters. A server is not shared between different applications, and servers allocated to the same application are assumed to run at the same frequency.

Horvath *et al.* [57] also exploited DVFS for use with dynamic reconfiguration for multi-tier server clusters, which is a typical architecture of current server clusters. For example, in a three-tier web server architecture, the first tier presents a web interface, the

second executes business logic scripts, and the third processes database accesses [58]. Each tier has a varying number of servers. All the servers in one tier run the same application, and a request goes through all tiers. Because the users only care about the response time across the entire pipeline, Horvath *et al.* make tradeoff decisions using the end-to-end delay as a simple SLA and they take into consideration the overheads for each transition between multiple sleep states and standby power levels. So their periodic energy optimization policy consisted of both active and inactive portions.

For the active portion, the policy assigns servers to each tier and sets appropriate frequencies to minimize the total energy expenditure of the multi-stage pipeline under the constraints of SLA. The optimization is then performed in two phases. The first phase finds the minimum number of servers in each tier so that the SLA is satisfied using the highest frequency for the machines. The second phase decreases the tier frequencies step by step ensuring that the SLA is still met.

For the inactive portion, the policy used is called sleep energy optimization, as it is concerned with choosing appropriate sleep states for the inactive servers. Horvath *et al.* observed that load fluctuations occur on a large time scale in practice. So the energy consumed during a transition from a sleep state to the active state is negligible but the time delay may make the system unresponsive. The policy therefore makes a tradeoff between system responsiveness and the sleep energy conservation. However, their policy does not consider how the tiers affect each other.

The research above is also limited to homogeneous clusters, but most recent commercial clusters are heterogeneous with respect to their performance, capacity and power consumption. Heath *et al*. [40] argued that one should model the different types of nodes when designing heterogeneous clusters, and used this when building their analytical model of power consumption in heterogeneous systems. This can be used to search for the best request distribution and cluster configuration that achieves minimum power/throughout ratio. If the real workload becomes greater than the maximum throughput in all configurations, then they choose the configuration that provides the maximum throughput. Their experiments were with a heterogeneous cluster that consisted of traditional nodes and blade nodes. They were able to show that their model-based solution conserved 45% more energy than the policy based on homogeneous clusters, where requests are distributed randomly across the active nodes and the decision which nodes to turn on and off is random.

The key issue surrounding these cluster-wide techniques is the decision on whether to reconfigure the system. Two factors must be taken into consideration. The first is the power model, discussed in the previous section. The second is the workload prediction.

Compared with clusters of front-end and computer servers, storage cluster techniques can leverage access-based allocation to nodes and disks [54, 55]. The *Popular*

Data Concentration (PDC) [11] technique conserves the energy of two-speed disk arrays. It copies the popular data to a high-speed disk and it slows down the other. The *Massive Array of Idle Disks* (MAID) [17] technique is one such technique. Considering that disk redundancy is commonly implemented in a storage system, the *Diverted Access* (DIV) technique [18] was proposed to leverage this redundancy and conserve disk energy. DIV handles reads and writes as follows: 1) a read is directed to the original disk only if the load is not too high; 2) when the load is high, a write is performed on all disks. When the load is light or moderate, a write is directed to the original disk immediately, but only propagated to redundant disks periodically. The goal of DIV is to increase idle times for the redundant disks so that they can be put into low power mode (standby) to conserve energy.

5.2 Power Capping

By studying the long-term data gathered from a spectrum of real-world deployment, Ranganathan *et al.* [59] summarized two trends of power consumption in data centers or clusters as follows: 1) resource utilization is low and bursty with spikes being relatively infrequent and of small duration; 2) the probability of synchronized spikes on all servers at the same time is rather low. The gap between the maximum power actually used by a large group of machines and their aggregate theoretical peak usage can be as large as 40% in data centers. *Power capping* is a technique that sets a safe threshold of power consumption and controls the cluster to prevent the actual power from exceeding it. It can enable the running of additional machines while keep the total power consumption under budget. It is also useful as a safety mechanism to prevent power supply spikes [60].

Fan *et al.* noticed that the nameplate ratings of power consumption tend to overestimate the actual maximum power usage [60]. They used the CPU utilization-based power model to estimate the power consumption of servers. By monitoring the power usage of an actual datacenter for six months, they found that power capping is effective for data centers, especially at the cluster level.

Power capping techniques generally consist of power sensing followed by power throttling. Ranganathan *et al.* [59] designed a cluster level power management controller and employed a management agent running on each server. The agent provides local power monitoring by runtime power measurement and power control. The controller periodically collects all local readings and estimates the total power consumption of the cluster. If the total exceeds the predefined power budget, the controller determines which server to throttle based on SLA and directs it to throttle at an appropriate level.

Femal *et al.* [61] noticed the uneven distribution in workload between nodes so they allocated power non-uniformly, according to these demands. Their power management framework is a two-level solution. The cluster-level power manager dynamically collects

information on all the nodes and assigns power to each node to ensure the total power is within budget. The node-level power manager then, at a fine level of granularity, allocates power to each device in the node, making sure its power expenditure is beneath its local threshold.

Wang *et al*. pointed out that the servers in a cluster are usually coupled together because they often run the same application or share a common power supply, and therefore, their management should be coordinated [62]. They proposed a *multi-input-multi-output* (MIMO) control algorithm to control the power consumption of multiple servers simultaneously. In every control cycle, the controller collects 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. Wang *et al*. conducted experiments to compare the coordinated DVFS based on MIMO and the individual DVFS in each local server. The latter is described as a *single-input-single-output* (SISO) controller. Their experiments demonstrated that the greatest improvement MIMO can bring is 15.3% with the same power budget. However, the system used for their experiments has only four nodes. It is unclear whether the centralized MIMO controller is scalable to large-scale systems.

5.3 Thermal Management

Increasing power density leads to high temperatures that exponentially decrease system reliability and increase the cooling cost too. Thermal management techniques are similar to capping power consumption in that they also consist of heat sensing and heat throttling subsystems. Heat can be monitored either by direct thermal profiling or by indirect power profiling, which would then use the correlation between temperature and power consumption. Also, heat can be throttled either by power management or by strengthened cooling. However, the latter of these may increase the cooling cost and the power consumption of cooling facilities. This section focuses on the thermal management techniques based on power management.

Skadron *et al*. [63] used a *Proportional Integral Differential* (PID) controller to implement dynamic thermal management. The control loop is depicted in Figure 1.

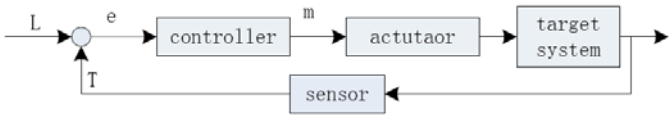

Figure 1: Feedback control loop.

where T is the measured temperature, L is the temperature threshold. The difference $e = L - T$ is the current error, used as the input of the controller. At any time t, controller output $m(t)$ is expressed as:

$$m(t) = K_C\left(e(t) + K_I\int e(t)dt + K_D\frac{de(t)}{dt}\right), \quad (9)$$

where K_C, K_I and K_D are constants that can be set according to the target systems. The sensor uses the resistances and capacitances of architectural components to model temperature [42]. The PID controller corrects error by three actions. The first is *proportional action,* which adjusts power in proportion to the error in direction of reducing the error. The second is *integral action,* which adjusts power in proportion to the time integral of previous errors. This action maintains the system at the zero-error state. The last is *derivative action*, which adjusts the power in proportion to the rate of change of error in direction of reducing rate of change. This action damps the response to avoid overshoot and maintain the stability of controller. If $m(t) < 0$, the system is overheated and the heat dissipation should be reduced during the next internal. Then the actuator disables the instruction fetch at an appropriate rate according to the thermal stress $m(t)$. Skadron *et al*. extended the power simulator Wattch [37] with their thermal model and demonstrated that their PID controller completely prevented thermal emergencies while improving performance by 36% over a hand-designed threshold technique.

Heath *et al.* [41] developed a thermal management system for server clusters, called Freon. The heat is periodically monitored by temperature daemons at each server. They defined two temperature thresholds for each component *c* at each server: high threshold T_h^c and red-line threshold T_r^c. When the temperature of component *c* exceeds T_h^c, Freon reduces the workload on the server by request redirection. When it exceeds T_r^c, Freon will turn the server off. Heath *et al.* also developed the energy conservation analogue to Freon (Freon-EC) which combines the energy conservation and thermal management. Freon-EC turns off as many servers as possible without degrading the performance. The decision whether to add or remove a server from the active cluster configuration is based on the predicted usage. If this figure is higher than a threshold, U_h, (70% in their experiments) then the server will be added. If the usage predicted for each server is lower than another threshold, U_l (60% in their experiments), even after the removal of a particular server, then that server is removed. Unlike load concentration [21] or Muse [20], the policy selects which servers to turn on and off according to their temperatures and physical locations within the data center.

In view of the dramatic increasing cost of cooling, Moore *et al.* [64] studied the use of temperature-aware workload placement as a means to reducing cooling cost. They explored the physics of heat transfer, captured the thermodynamic behavior in the data center, and then prioritized the servers based on their cooling costs. Examining the location-aware cooling cost of each server, they explored the resources supply issue with minimum cooling cost. Unlike the previous schemes, they not only turn servers on and off dynamically but they also adjust the computer room air conditioning (CRAC) supply

temperature. When a server inlet exceeds the safe temperature, the CRCR supply temperature will be lowered to bring the server's temperature back below the system safe threshold. Only if all servers are safe will the CRCR supply temperature be adjusted to as high a value as possible to reduce cooling cost.

6. Power Management for High Performance Computing Systems

A typical HPC system can be divided into three sub-systems: front-end server nodes, computing nodes and storage nodes. The front-end server nodes provide an operating interface to users, impose jobs on computing nodes and monitor their status. The computing nodes provide a high performance computing capability to process jobs. The storage nodes provide massive storage capacity. These three sub-systems are integrated into a tightly coupled system by the high speed inter-connecting network.

Unlike commercial clusters, the scientific applications are generally non-interactive. They involve many, even massively many, coordinated nodes. Communication and synchronization is interlaced with the computation, and the running time is relatively long. Ge *et al*. [8, 65] analyzed the difference in power consumption between the non-interactive scientific parallel HPC and the interactive workloads in distributed cluster systems. They observed that the CPU typically dominates system power consumption in supercomputers. This is demonstrated by the distribution shown in Figure 2 below [65].

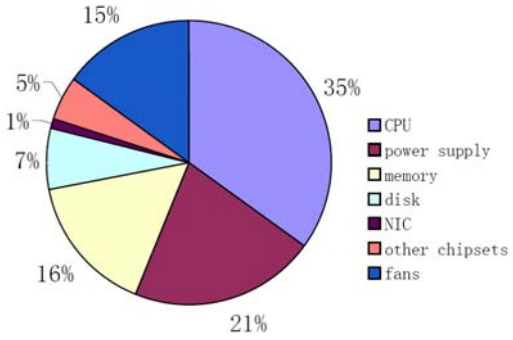

Figure 2: CPU dominates system power consumption in supercomputers.

These characteristics of HPC impose new challenges and opportunities to power management for supercomputers. The techniques for HPC systems can be classified into execution phase analysis and job allocation approaches. They are discussed below.

6.1. Execution Phase Analysis

Parallel applications consist of CPU computing, memory access, I/O, communication and synchronization. In a supercomputer, I/O is implemented as communication with a storage sub-system, so we treat it as a special case of communication. Synchronization is

also a form of communication. So there are really only three types of execution phases: CPU-bound phases, memory-bound phases and the communication-bound phases. When the execution is bounded by one component, the other components can be put into lower power states without significant performance degradation. A number of power management techniques have emerged to take advantage of the boundedness on various components.

A. CPU activity analysis

Processor DVFS based on the CPU computing stress is a traditional power management approach. Linux [66] uses CPU utilization as the metric of CPU-boundedness. Hsu *et al.* [28] in contrast used the number of millions of instruction executed per second (MIPS). They decomposed the workload into two parts: on-chip workload, whose performance depends on CPU frequency, and off-chip workload, whose performance does not. The impact of CPU frequency change on execution-time was modeled as follows [28]:

$$1+\delta=\frac{T(f)}{T(f_{\max})}\approx\frac{mips(f_{\max})}{mips(f)}\approx\beta\frac{f_{\max}}{f}+(1-\beta) \tag{10}$$

where β quantifies the intensity level of off-chip workload, δ is the relative performance slowdown, $T(f)$ is the execution-time of a workload at CPU frequency f, $mips(f)$ is the average MIPS rate for CPU frequency f and f_{max} is the peak CPU frequency. They used a regression method over equation (10) and derived β as a function of MIPS rates history of last intervals and CPU frequencies as equation (11).

$$\beta=\frac{\sum_i(\frac{f_{\max}}{f_i}-1)(\frac{mips(f_{\max})}{mips(f_i)}-1)}{\sum_i(\frac{f_{\max}}{f_i}-1)^2} \tag{11}$$

$$f^*=\max\left(f_{\min},\frac{f_{\max}}{1+\delta/\beta}\right) \tag{12}$$

Using (10) and (11), we can see that the desired CPU frequency f^* can be calculated from equation (12). Given as a parameter the acceptable relative performance slowdown δ, their MIPS-based run-time processor DVFS scheduler used PMC to monitor processor MIPS rates and to dynamically compute the desired CPU frequency.

Malkowski *et al.* [67] took advantage of the memory-bound phase to select CPU frequencies. Their scheme relied on the fact that the lowest level cache misses are good indicators of whether an execution phase is memory-bound. An execution is divided into a sequence of windows. Each window contains a fixed number of clock cycles. At the end of each window, its phase characteristics are determined. They used an L3 (outermost) cache miss rate of 0.4 as the threshold to detect memory-bound phases. Above this

threshold, the new CPU frequency f_{new} is calculated as follows [67].

$$f_{new} = \left\lceil f_{old} \times \frac{Instructions\_executed\_in\_current\_window}{Instructions\_executed\_in\_last\_window \times 100} \right\rceil \times 100 \qquad (13)$$

If the phase is not memory-bound, then the CPU runs at peak frequency.

Ge *et al*. [29] combined MIPS and cache miss rate to detect phase boundedness. Their scheme is close to the work of Hsu *et al.* [28] discussed in the previous subsection. They also decomposed the workload into on-chip and off-chip workloads, but then they further decomposed the off-chip workload into memory accesses w_{mem}, I/O accesses w_{io} and system idle w_0 parts. They used the memory data access events (lowest cache misses) to describe w_{mem} and the statistic data provided by Linux pseudo-file */proc/stat* to describe w_{io} and w_0. The first value of CPU intensiveness κ (equivalent to β in [28]) is calculated from the statistic data and the predicted valuer κ' of the next interval is decided by

$$\kappa'_{i+1} = (1-\lambda)\kappa'_i + \lambda\kappa_i \qquad (14)$$

The smooth factor λ was set as 0.5 in their experiments. The desired CPU frequency is also decided by equation (12).

Freeh *et al.* [68] also detected phases using memory-boundness. Their execution phases are recognized by two steps. In the first step, the program is divided into *blocks* using two rules.

Rule 1: Any *message passing interface* (MPI) [69] operation demarcates the block boundary.

Rule 2: If the memory pressure changes abruptly, a block boundary occurs at this change.

They intercepted MPI calls to implement the first rule and used *operations per miss* (OPM) as the measure of memory pressure. This measure is larger the lower the OPM.

In the second step, the blocks are merged into phases. Two adjacent blocks are merged if the pressure they put on memory is within same threshold. Freeh *et al*. developed an efficient algorithm to assign appropriate CPU frequencies to the phase.

These approaches determine whether or not the CPU bounds the performance of the whole system by using information about CPU activities, such as CPU utilization, MIPS or cache failure rate. They are effective for phases where off-chip operations blocks CPU performance. However, in some situations, the CPU performance is not a bound on the performance of the whole system. For example, the CPU could be in busy loop waiting for other events and it need not run at peak frequency. The above techniques cannot detect such a situation so they would miss the opportunity to save energy.

B. Communication phase analysis

Communication-boundedness happens when one node must wait for another to finish communication, either because the device is slower or because the node has been given more work to do. Techniques have been developed to recognize such situations and take

action to save energy. This section will survey such techniques.

B.1. Communication intensive region analysis

A communication phase is a period in the execution of an application that is communication intensive but not CPU intensive. In [70], Lim *et al*. developed a technique to recognize communication phases online and then apply CPU DVFS to save energy. The technique intercepts and records the sequences of MPI calls during the execution of a program. A segment of program code is recognised as a reducible region if there are high concentrated MPI calls (e.g. calls 1-3 in Figure 3) or if an MPI call is long enough (e.g. call 5 in Figure 3). When the recognized reducible region is entered again, it sets the processor to run at an appropriate lower frequency to save energy with negligible performance degradation. Lim *et al*. used the *micro-operations/microsecond* (OPS) metric to judge the dependence of performance on the CPU in these reducible regions and developed a table to map OPS to CPU frequency that will minimize the energy consumption measured by *ED* [22].

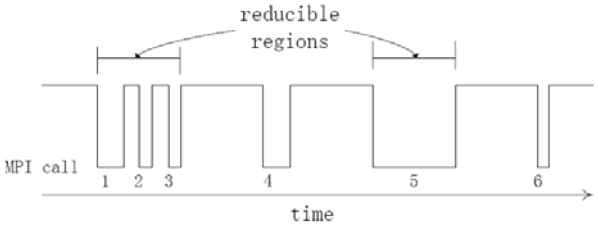

Figure 3: An example trace of an MPI program. The calls 1-3 form a reducible region of multiple calls close to each other. The call 5 is a reducible region of a long call. The call 4 and call 6 are not in reducible communication region because they are neither close enough nor long enough [70].

B.2. Inter-node imbalance analysis

HPC applications are usually parallel programs. When the computational load is not well balanced across the nodes, a node that arrives earlier at a synchronization point must wait for the other slower nodes. Such a situation may occur repeatedly if the code is in a loop. Thus, it is possible and worthwhile to shift the faster node to a lower frequency to conserve power without significant performance loss.

Kappiah *et al*. [71] developed a tool called *Jitter* to recognize slack moments in performance due to the inter-node imbalance and to use DVFS for power reduction. Jitter calculates a node's *slack* as the wait time in an iteration divided by the length of the iteration. A node's *net slack* is computed as the difference between its own slack and the minimum slack of the nodes. In contrast to Lim, *et al*., Jitter adjusts CPU frequency evolutionally rather than predetermining it in advance. In particular, Jitter reduces a node's CPU frequency, if its net slack satisfies the following condition.

$$net\_slack > S \bullet d_g \qquad (15)$$

where S is a user-defined constant, which was 10-20% in their experiments, and d_g is an adaptive factor. Whenever a node reduces frequency, it also increase d_g using the formula d_g=d_g *$bias$. In their experiment, $bias$ = 2 and initially d_g = 1. Jitter increases a node's CPU frequency, if its net slack satisfies the following condition:

$$net\_slack < \alpha \bullet S / u_g \tag{16}$$

where u_g plays the same role as d_g, and α is the stabilizing factor, which must be less than 1. In their experiment, α= 0.5. Their experiments show that Jitter is effective for scientific computations that contain large numbers of iterative loops.

In [72], Rountree *et al.* divided the execution of an application into a number of *tasks*. Each task is an execution segment on a single processor between the completions of two successive MPI communication calls. Each task is further decomposed into a computation portion and a communication portion, and a directed acyclic graph is constructed to analyse the execution of parallel programs. A *critical path* (CP) is the longest path from the source to the sink node in the graph. When a processor executes a task on the critical path, it will not wait for data to arrive during the task. Otherwise, a slack may occur, which is the blocked time during MPI communication calls. Thus, the processor can be slowed to remove the slack without affecting the overall execution time. Rountree *et al.* developed a system called *Adagio* to collect statistical data on task execution slacks, compute the desired frequency and represents the results in a hash table. The first instance of a task always executes at peak frequency. When the task executes again, an appropriate frequency can be found in the hash table. In their experiment, Adagio provided total system energy savings of up to 20% with a less than 1% increase in execution time.

6.2. Job Allocation Approaches

The priciples of power conservation through task allocation developed for clusters can also be applied to HPC. However, the technical solutions employed successfully in data centers, such as load concentration [21] and Muse [20], are ineffective for supercomputers because their workload characteristics and performance demands are different. In the section, we focus on inter-node system-wide job allocation for HPC. Intra-node task scheduling for power conservation is beyond the scope of this paper, and is thus omitted.

SLURM [73] is a popular resource management and job scheduling system for supercomputers. It is employed by about 1000 systems including more than 30% of the systems in the TOP500. It also provides a power management facility for idle nodes. A node that remains idle for a period of time can be placed in a lower power state and then restored to the normal state once a new job is assigned. To avoid an instantaneous surge in power demand due to the number of active nodes being rapidly changed, SLURM can

power nodes up or down gradually. However, it does not provide any power management policy. The user decides after how much idle time the lower power state should be entered, the target power states to be in, and the maximum number of nodes that can change state per minute. SLURM's power management facility is useful, but little research has been conducted on how to use it effectively, especially on how to allocate tasks to nodes to enable power conservation.

Most supercomputers now have heterogeneous architectures [2, 74] and this introduces new challenges to system-wide job allocation. In heterogeneous architectures, the system power model becomes much more complicated because the power consumption patterns, power state spaces, and power management mechanisms are different. So are the computation capabilities and the roles of the different nodes. A system-wide power management technique must take all of these differences into consideration.

Zong *et al*. [75] focused energy-efficient parallel application allocation for heterogeneous architectures. They defined four types of difference in a heterogeneous system as follows. First, different nodes may have different preferences for each task so a node running faster for one type of task may not run faster for other types of tasks. Secondly, a task may have different execution times on different nodes. Thirdly, communication rates depend on the corresponding interconnection networks. Finally, the power consumed by each node and each networking device may be different. Their job allocation algorithm for heterogeneous systems, called *energy efficient task duplication schedule* (EETDS), assigns parallel tasks to the nodes in two phases: a grouping phase and and allocation phase. During the grouping phase, EETDS group tasks based on their relevance to each other, i.e. whether they are in the same critical path. All the tasks in a group are allocated to the same computing node in order to reduce communication overheads. In the process of grouping, task are duplicated and sent to multiple computing nodes if performance can thereby be further improved. During the allocation phase, the groups are allocated to heterogeneous nodes in an energy efficient way. The phase estimates the energy consumption of all groups on different candidate nodes and generates an allocation solution with minimized total energy consumption of the whole system. In their experiments, they set three different environments using four types of processors to simulate computing node heterogeneity, network heterogeneity and different communication to computation ratio. Their experiments showed that EETD can significantly reduce energy consumption up to 47.1% with only a negligible degradation in performance.

7. Power Management in Virtual Environment

Virtual machine (VM) technology was first introduced in the 1960s. It has grown in popularity during recent years. As shown in Figure 4 below, VM technologies allow multiple guest virtual machines to share common physical hardware resources [76, 77, 78]. It enables the division of computer resources into multiple isolated execution environments, which can be used as thought it was a real machine, from the user's view point.

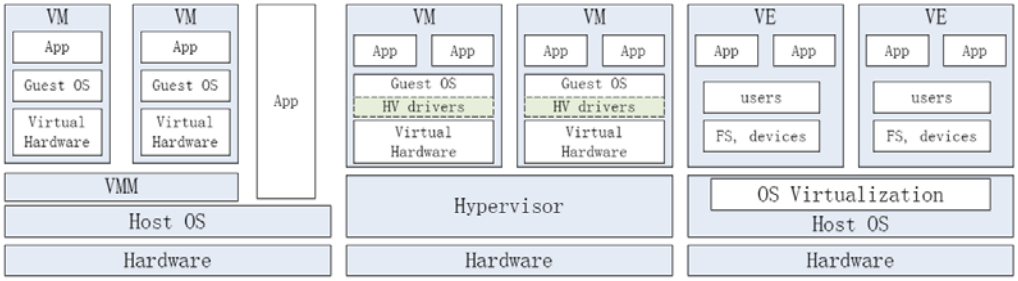

Figure 4: Illustration of the different architectures of virtualization. Virtualization may be implemented in different layer.

In Figure 4, the virtualization layer is the software responsible for hosting and managing the virtual machines, and is also called the *virtual machine monitor* (VMM).

VM technologies can be classified into one of three categories according to the layer in which the VMM resides.

- *Hosted architecture* virtualization (Figure 4.a). The hosted architecture installs and runs the VMM as an application on top of an operating system. A typical example is VMware Workstation [78]. Because all accesses to the hardware must go through the host OS, the performance overhead of the hosted architecture is significant and this method is not suitable for high performance systems.
- *Hypervisor-based* virtualization (Figure 4.b). The VMM, known as a *hypervisor*, is installed on top of bare hardware. Typical examples of hypervisor-based virtual machines are Xen [79]] and VMware ESX [78]. Because the hypervisor accesses the hardware resources directly, it is more efficient than the hosted architecture.
- *OS-level* virtualization (Figure 4.c), also known as *container-based* virtualization. It implements the virtualization at the OS system call/ABI layer. Multiple isolated virtual environments (VE, also known as container, VPS, etc.) are enabled within a single operating system kernel. Typical OS-level VMs include Solaris Zones/Containers [80], Linux VServer [81], OpenVZ [82] and HPVZ [83]. Because applications in VE use the operating system's normal system call

interface, OS-level virtualization imposes little or no performance overhead.

With hardware support and the development of virtualization technologies, the performance overhead of virtualization has become negligible in many scenarios [83, 84]. Hypervisor-based and OS-level VM have been employed successfully in high performance systems [82, 83, 84, 85, 86, 87, 88].

The employment of virtual machines sheds a new light on power management for high performance systems. For example, live task migration is needed for workload consolidation in HPC. However, traditional process migration suffers from a residual dependency problem. That is, a part of the process has to be left on the source or an intermediate node in order to achieve network transparency, e.g. in order to forward subsequent messages to the target node [89]. VM migration provides a promising solution for conquering this problem, thus enabling better power management [90]. For example, Stoess *et al*. [91] developed a multi-tiered infrastructure, which enables intra-node virtual CPU (vCPU) migration and inter-node live VM migration for workload consolidation and thermal balancing. To enable power-aware VM migration, Verma *et al.* [92, 93] investigated job allocation for HPC.

However, the introduction of virtualization imposes new challenges to traditional power management techniques, though solutions are starting to emerge. Although OS-level virtualization has been employed successfully by HPC, there is no work, to our knowledge, on power management based on OS-level virtualization. Instead, existing works on VM-based power management are all based on hypervisor-based virtualization.

As discussed above, power management techniques choose the appropriate state by relying on knowledge about the phases of the applications and/or workload on the whole system. As Stoess *et al*. pointed out [94], in a virtual machine environment, these techniques cannot easily be applied either to the individual VM system or at the hypervisor level. In particular, if the power management technique is applied at VM level, each guest operating system only knows the state of the application(s) or the workload on its own VM, not on the other VMs that share the hardware resources. If one VM changes the state of the hardware based on information about its own local applications and/or workload, then it is inevitable that this will have adverse effects on other VMs. Power management techniques cannot be applied only at the hypervisor level because the hypervisor has insufficient knowledge about the applications, although the hypervisor can control the hardware states just as a traditional OS can.

Stoess *et al.*'s solution [94] supports power management at both host-level and guest-level as shown in Figure 5.

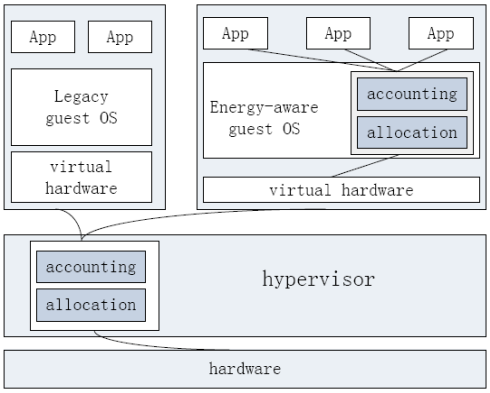

Figure 5: Two-level power management architecture. The host-level subsystem enforces energy constraint on each guest. Guest OS performs its own guest-level energy management [94].

The host-level subsystem enforces system-wide energy conservation between VMs. It measures power consumption by the VMs and it regulates the allocation of physical devices in a coarse-grained manner to ensure that each VM does not consume more than a given power allotment. A power-aware guest OS then redistributes the VM-wide power allotment based on its fine-grained application-level knowledge.

In [95], Nathuji *et al*. also addressed power management problems with virtual machines for distributed systems. They introduced the notion of *soft power states*, which are virtualized versions of the hardware power states. The virtualization of power states is transparent to the VM, and so the guest OS manages the soft power with traditional techniques. Based on Xen hypervisor [79], they developed a coordinated power management system for distributed environment, called VirtualPower, shown as Figure 6.

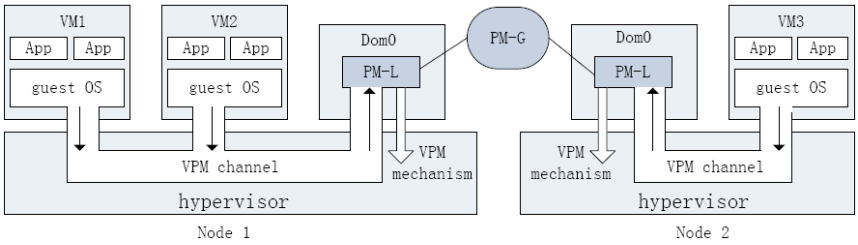

Figure 6: VirutalPower Management Architecture. The VPM channels deliver power management actions from guest OS to PM-L. PM-L regulates the hardware based on its local rules and the coordination from PM-G [95].

VirtualPower maps virtualized soft states to hardware states through *virtual power management* (VPM) channels, which reside in the hypervisor. These VPM channels deliver power management actions from the guest OS to the hypervisor as updates to soft states, and respond as if the hardware had taken the actions. These requests for soft power state changes are interpreted by *local power management policies* (PM-L), which regulate the hardware based on its local rules and on coordination from *global policies* (PM-G).

Based on information about nodes, ranks and cluster, PM-G makes global power management decisions, such as VM remapping or migration. However, PM-G is a centralized scheme, and it is not applicable to large scale systems.

8. Summary

It is widely recognized that power management is crucial in high performance systems. Compared with battery-operated systems, the power management of high performance systems presents new problems so new solutions have emerged in recent years in the literature. Table 1 summarizes these techniques.

In this paper, we surveyed the metrics used for evaluating the effectiveness of power management, and how metrics are chosen in various scenarios. Power consumption profiling is a key technique for developing power models. PMC-based profiling is a promising solution but it is limited by the amount of events which can be defined by the processor PMC. Device monitoring techniques monitor and record the behaviors of the devices by drivers. They can provide a useful supplement to PMC. Building a complete and accurate power model for the whole system should combine the techniques of the PMC and device driver.

The effectiveness of heuristics-based power management techniques relies heavily on predicting the future workload. When the real execution behaviors do not match the prediction, the techniques will seem dubious. Guaranteeing correctness and effectiveness is a challenge. Control-theoretic techniques provide a meaningful hint for future development [62, 63, 96]. For large-scale high performance systems, a trade-off must be made between power management accuracy and effectiveness, on the one hand, and acceptable overheads and costs, on the other. Distributed or multi-level techniques may be viable solutions.

For clusters, power management takes place both on single nodes and on the cluster as a whole. There are some promising cluster-wide power management solutions based on workload redistribution. Examples include load concentration [21] and Freon [41]. The reconfiguration mechanism that redistributes the forthcoming request is efficient for interactive transaction processing in commercial clusters. Power-driven workloads migration across nodes and dynamic system reconfiguration are also viable solutions to cluster-wide power management. However, heterogeneity is a new challenge for cluster-wide power management.

Virtual machine technologies can provide a new solution and they have been employed successfully in high performance systems. They shed a new light on power management by providing new mechanisms for power consumption migration through

virtual machine migration. However, the introduction of virtualization imposes new challenges to traditional power management techniques, and novel solutions have started to emerge for hypervisor-based virtual machines. Supercomputers in the scientific parallel community tend to employ OS-level virtualization. This offers the opportunity to manage a large number of processes in parallel applications by grouping them in the granularity of the VE. However, no work on power management with OS-level virtualization has been reported in the literature as far as we know.

Another recent novelity is the problems related to autonomic computer system and how to apply self-organization techniques, such as multi-agent systems, to power management [97].

Acknowledgement

The authors would like to thank their colleagues in the Applied Formal Methods Research Group at the Department of Computing and Electronics of the Oxford Brookes University for valuable discussions on related topics. They are most grateful to Dr. Ian Bayley for his detailed comments and corrections. The authors would also like to thank the anonymous reviewers of the paper for their critical and constructive comments on the draft versions of the paper.

References

1. Feng W. Making a case for efficient supercomputing. ACM Queue 2003; **1**(7): 54–64.
2. Meuer H, Strohmair E, Simon H. and Dongarra J. 33rd Edition of TOP500 List. http://www.top500.org/lists/2009/06 [Jun. 23, 2009].
3. U.S. Enviromental Protection Agency ENERGY STAR Program. Report to Congress on Server and Data Center Energy Efficiency. http://www.energystar.gov/ia/partners/prod_development/downloads /EPA_Datacenter_Report_Congress_Final1.pdf [Aug. 2, 2007].
4. Seager M. What are the future trends in high-performance interconnects for parallel computers? *Proceedings of the 12th IEEE Symposium on High-Performance Interconnects* (Panel 1). Stanford, Califonia, USA. 25-27 Aug., 2004; 3-3.
5. American Power Conversion (APC). Determining Total Cost of Ownership for Data Center and Network Room Infrastructure. ftp://www.apcmedia.com/salestools/CMRP-5T9PQG R2 EN.pdf [Dec.2003].
6. US Enviroment Protection Agency. http://www.epa.gov/cleanenergy/energy-resouces/calculator.html [Feb. 17, 2009].
7. Venkatachalam V, Franz M. Power reduction techniques for microprocessor systems. *ACM Computing Surveys* 2005; **37**(3): 195-237.
8. Ge R, Feng X, Cameron K. Improvement of power-performance efficiency for high-end computing. *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium* (*IPDPS'05*), Vol. 12. Workshop

11: *the 1st IEEE Workshop on High-Performance, Power-Aware Computing* (*HPPAC'05*). Denvor, Colorado, USA. 4 Apr., 2005;233.2.

9. Intel. Intel 64 and IA-32 Architectures Software Developer's Manual. Volume 3A: System Programming Guide, Part 1. Order Number: 253668-031US. http://www.intel.com [Jun. 2009].
10. Micron Techology Inc. Calculating Memory System Power for DDR3. http://download.micron.com/pdf/technotes/ddr3/TN41_01DDR3 Power.pdf [Aug. 2007].
11. Pinheiro E, Bianchini R. Energy conservation techniques for disk array-based servers. *Proceedings of the 18th ACM/IEEE Conference on Supercomputing* (*SC'04*). Malo, France. Jun. 2004; 68-78.
12. Gurumurthi S. Sivasubramaniam A, Kandemir M, Franks H. DRPM: Dynamic speed control for power management in server class disks. *Proceedings of the 30th Annual International Symposium on Computer Architecture* (*ISCA'03*). San Deigo, Califonia, USA. 9-11 Jun., 2003; 169–181.
13. Carrera E, Pinheiro E, Bianchini R. Conserving disk energy in network servers. *Proceedings of the 17th International Conference on Supercomputing* (*ICS'03*). San Francisco, CA, USA. 23-26 Jun., 2003; 86-97.
14. Hewlett-Packard, Intel, Microsoft, Phoenix and Toshiba. Advanced Configuration and Power Interface Specification Revision 4.0. http://www.acpi.info/DOWNLOADS/ACPIspec40.pdf [Jun. 16, 2009].
15. Intel, What is difference between deep and deeper sleep states?. http://www.intel.com/support/processors/sb/CS-028739.htm [Feb 24, 2009].
16. Pandey V, Jiang W, Zhou Y. Binachini R. DMA-aware memory energy management. *Proceedings of the 12th International Symposium on High-Performance Computer Architecture* (*HPCA-12*), Austin, Texas, USA. 11-15 Feb., 2006; 133-144.
17. Colarelli D, Grunwald D. Massive arrays of idle disks for storage archives. *Proceedings of the 16th ACM/IEEE Conference on Supercomputing* (*SC'02*). Baltimore, Maryland, USA. 16-22 Nov., 2002; 1-11.
18. Pinheiro E, Bianchini R, Dubnicki C. Exploiting redundancy to conserve energy in storage systems. *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems 2006* (*SIGMETRICS/Performace'06*). Saint Malo, France. 26-30 Jun., 2006; 15-26.
19. PCI-SIG. PCI Bus Power Management Interface Specification, Revision 1.2. http://www.pcisig.com/specifications/conventional/pcipm1.2.pdf [March 3, 2004].
20. Chase J, Aderson D, Thakar P, Vahdat A, Doyle R. Managing energy and server resources in hosting centers. *Proceedings of the 18th ACM Symposium on Operating Systems Principles* (*SOSP'01*). Banff, Alberta, Canada. Oct. 2001; 103-116.
21. Pinheiro E, Bianchini R, Carrera E, Health T. Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems. *Technical Report DCS–TR–440*, Department of Computer Science, Rutgers University. May 2001.
22. P´enzes P, Martin A. Energy-delay efficiency of VLSI computations. *Proceedings of the 12th ACM Great Lakes Symposium on VLSI* (*GLSVLSI'02*). New York, NY, USA. 18-19 Apr., 2002;104-111.
23. Martonosi M, Brooks D, Bose P. Modeling and analyzing cpu power and performance: metrics, methods, and abstractions (Tutorial). *Proceedings of Joint International Conference on Measurement and Modeling of Computer Systems* (*SIGMETRICS/Performance'01*). Cambridge, Massachusetts, USA. 16-20 Jun., 2001.
24. Hsu C, Feng W, Archuleta J. Towards efficient supercomputing: a quest for the right metric. *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium* (*IPDPS'05*), Vol. 12, Workshop 11: *the 1st IEEE Workshop on High-Performance, Power-Aware Computing* (*HPPAC'05*). Denver, Colorado, USA. 4, Apr.2005;230.1.
25. Green500 List. 5th Edition of the Green500 List. http://www.green500.org/lists/2009/06/list.php [Jun. 2009].
26. Sharma S, Hsu C, Feng WC. Making a case for a Green500 list. *Proceedings of the 2nd IEEE Workshop on High-Performance, Power-Aware Computing* (*HPPAC'06*). Rhodes, Greece. 25-29 Apr., 2006.
27. Choi, K, Soma R, Pedram M. Dynamic voltage and frequency scaling based on workload decomposition. *Proceedings of International Symposium on Low Power Electronics and Design 2004* (*ISLPED'04*). Newport Beach, Califonia, USA. 9-11 Aug.,2004;174-179.

28. Hsu C, Feng W. A power-aware run-time system for high-performance computing. *Proceedings of the 19th ACM/IEEE Conference on Supercomputing* (*SC'05*). Washington, USA. 12-18 Nov., 2005;1-9.

29. Ge R, Feng X, Feng W, Camron K. CPU MISER: a performance-directed, run-time system for power-aware clusters. *Proceedings of the 2007 International Conference on Parallel Processing* (*ICPP'07*). Xian, China. 10-14 Sept., 2007;18-25.

30. Lim K, Ranganathan P, Chang J, Patel C, Mudge T, Reinhardt S. Understanding and designing new server architecutures for emerging warehouse-computing environments. *Proceedings of the 35th International Symposium on Computer Architure* (*ISCA'08*). Beijing, China. 21-25 Jun., 2008;315-326.

31. Standard Performance Evaluation Corporation. SPECpower_ssj2008. http://www.spec.org/specpower/ [Apr. 15, 2009].

32. Belady C, Rawson A, Pfleuger J, Cader T. Green Grid Data Center Power Efficiency metrics: PUE and DCiE, White Paper #6. http://www.thegreengrid.org/gg_content/ TGG_Data_Center_Power_Efficiency_Metrics_PUE_and_DCiE.pdf [Nov. 2007].

33. Brill K. Data Center Energy Efficiency and Productivity. The Uptime Insititute white paper TUI3004D. http://www.uptimeinstitute.org/symp_pdf/(TUI3004D)DataCenterEnergyEfficiency.pdf [2007].

34. Snowdon D, Petters S, Heiser G. Power measurement as the basis for power management. *Proceedings of the 1st Workshop on Operating System Platforms for Embedded Real-Time Applications* (*OSPERT'05*). Palma, Mallorca, Spain. Jul. 2005.

35. Bellosa F. The Case for Event-Driven Energy Accounting. *Technical Report TR-14-01-07*, Depart of Computer Science, University of Erlangen, Erlangen, Germany. Jun. 2001.

36. Najm F. A survey of power estimation techniques in VLSI circutes. *IEEE Transactions on Very Large Scale Integration* (*VLSI*) *Systems* 1994; **2**(4): 446-455.

37. [70] Brooks D, Tiwari V, Martonosi M. Wattch: a framework for architectural-level power analysis and optimizations. *Proceedings of the 27th International Symposium on Computer Architecture* (*ISCA'00*). Vancouver, British Columbia, Canada10-14. Jun., 2000;83-94.

38. Brockmeyer E, Miranda M, Corporaal H, Catthoor F. Layer assignment techniques for low energy in multi-layered memory organisations. *Proceedings of the 6th ACM/IEEE Design and Test in Europe*. Leuven Belgium. 03-07 Mar., 2003;1070-1075.

39. Gurumurthi S, Sivasubramaniam A, Irwin M, Vijakrishnan N, Kandemir M. Using complete machine simulation for software power estimation: the softwatt approach. *Proceedings of the 8th International Symposium on High Performance Computer Architecture* (*HPCA'02*). Cambridge, UK. 02-06 Feb., 2002;141–150.

40. Heath T, Diniz B, Carrera E, Meira W, Bianchini R. Energy conservation in heterogeneous server clusters. *Proceedings of 10th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (*PPoPP'05*). Chicago, USA. 15-17 Jun., 2005;186-195.

41. Heath T, Centeno A, George P, Ramos L, Jaluria Y, Bianchini R. Mercury and Freon: temperature emulation and management for server systems. *Proceedings of the 12th international conference on Architectural Support for Programming Languages and Operating Systems* (*ASPLOS'06*). San Jose, California, USA. 21-25 Oct., 2006;106-116.

42. Skadron K, Stan M, Sankaranarayanan K, Huang W, Velusamy S, Tarjan D. Temperature-aware microarchitecture: modeling and implementation. *ACM Transactions on Architecture and Code Optimization* (*TACO*) 2004; **1**(1): 94-125.

43. Flinn J, Satyanarayanan M. Powerscope: a tool for profiling the energy usage of mobile applications. *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications* (*WMCSA'99*). New Orleans, Louisiana, USA. Feb. 1999;2-10.

44. Joseph R, Brooks D, Martonosi M. Live, runtime power measurements as a foundation for evaluating power/performance tradeoffs. *Workshop on Comlexity Effetice Design in Conjunction with the 28th International Symposium On Computer Architecture* (*ISCA'01*). Göteborg, Sweden. 30 Jun. -4 Jul., 2001.

45. Bellosa F. The benefits of event-driven energy accounting in power-sensitive systems. *Proceedings of the 9th ACM SIGOPS European Workshop*. NewYork, USA. 17-20 Sept., 2000;37-42.

46. Weissel A, Bellosa F. Process cruise control: event-driven clock scaling for dynamic power management. *Proceedings of the 2002 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems* (*CASES'02*). Grenoble, France. 8-11 Oct., 2002;238-246.

47. Bellosa F, Weissel A, Waitz M, Wessil A. Event–driven energy accounting for dynamic thermal management. *Proceedings of the 4th Workshop on Compilers and Operating Systems for Low Power* (*COLP'03*). New Orleans, Louisiana, USA. 27 Sept., 2003.

48. Isci C, Martonosi M. Runtime power monitoring in high-end processors: methodology and empirical data. *Proceedings of the 36th Annual ACM/IEEE International Symposium on Microarchitecture* (*MICRO-36*). San Deogo, California, USA. 3-5 Dec., 2003; 93-104.

49. Bircher W, Law J, Valluri M, John L. Effective use of performance monitoring counters for run-time prediction of power. *Technical Report TR-041104-01*, University of Texas at Austin, Texas, USA. Nov. 2004

50. Chang F, Farkas K, Ranganathan P. Energy-driven statistical profiling: detecting software hotspots. *Proceedings of the 2nd International Workshop on Power-Aware Computer Systems* (*PACS'02*). Cambridge, MA, USA. 2 Feb., 2002; 110-129.

51. Hu C, Jimenez D, Kremer U. Efficient program power behavior characterization. *High Performance Embedded Architectures and Compilers* (*HiPEAC*). Springer: Berlin. ISSN 1611-3349. Jul. 2007. **4367**:183-197.

52. Isci C, Martonosi M. Phase characterization for power: evaluationg control-flow-based and event-counter-based techniques. *Proceedings of the 12th International Symposium on High-Performance Computer Architecture* (*HPCA-12*). Austin, Texas, USA. 11-15 Feb., 2006; 121-132.

53. Lefurgy C, Rajamani K, Rawson F, Felter W, Kistler M, Keller T. Energy Management for Commercial Servers. *Computer* 2003; **36**(12):39-48.

54. Bianchini R, Rajamony R. Power and energy management for server systems. *Technical Report DCS-TR-528*, Department of Computer Science, Rutgers University, New Jersey, USA. Jun. 2003.

55. Rajamony R, Bianchini R. Energy management for server clusters (Tutorial). *Proceedings of 16th Annual ACM International Conference on Supercomputing* (*SC'02*). New York, USA. 22-26 Jun., 2002; 165.

56. Chen Y, Das A, Qin W, Sivasubramaniam A, Wang Q, Gautam N. Managing server energy and operational costs in hosting centers. *Proceedings of the 2005 ACM International Conference on Measurement and Modeling of Computer Systems* (*SIGMetrics'05*). Banff, Alberta, Canada. 6-10 Jun., 2005; 303 – 314.

57. Horvath T, Skadron K. Multi-mode energy management for multi-tier server clusters. *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques* (*PACT'08*). Toronto, Ontario, Canada. 25-29 Oct., 2008; 270-279.

58. Horvath T, Abdelzaher T, Skadron K, Liu X. Dynamic voltage scaling in multi-tier web servers with end-to-end delay control. *IEEE Transactions on Computers* 2007; **56**(4):444-458.

59. Ranganathan P, Leech P, Irwin D, Chase J. Ensemble-level power management for dense blade servers. *Proceedings of the 33rd International Symposium on Computer Architecture* (*ISCA'06*). Boston, MA, USA. 17-21 Jun., 2006; 66-77.

60. Fan X, Weber W, Barroso L. Power provisioning for a warehouse-sized computer. *Proceedings of the 34th International Symposium on Computer Architecture* (*ISCA'07*), San Diego, California, USA. Jun. 2007; 13-23.

61. Femal M, Freech V. Boosting data center performance through non-uniform power allocation. *Proceedings of the 2nd International Conference on Autonomic Computing* (*ICAC'05*). Seattle, WA,USA. 13-16 Jun., 2005; 250-261.

62. Wang X, Chen M. Cluster-level feedback power control for performance optimization. *Proceedings of the 14th IEEE International Symposium on High-Performance Computer Architecture* (*HPCA'08*). Salt Lake City, Utah. 16-20 Feb., 2008; 101-110.

63. Skadron K, Abdelzaher T, Stan M. Control-theoretic techniques and thermal-rc modeling for accurate and localized dynamic thermal management. *Proceedings of the 8th International Symposium on High-Performance Computer Architecture* (*HPCA'02*). Washington, DC, USA. 02-06 Feb., 2002; 17-33.

64. Moore J, Chase J, Rangantahan P, Sharma R. Making scheduling 'cool': temperature-aware workload placement in data centers. *Proceedings of the 2005 Annual Conference on USENIX Annual Technical Conference* (*USENIX'05*). Marriott Anaheim, Anehim, CA. 10-15 Apr., 2005; 5-18.

65. Ge R, Feng X, Cameron K. Performance-constrained, distributed dvs scheduling for scientific applications on power-aware clusters. *Proceedings of the 19th ACM/IEEE Conference on Supercomputing* (*SC'05*). Seattle, Washington, USA. 12-18 Nov., 2005; 34-44.

66. Linux. ftp://ftp.kernel.org/pub/linux/kernel/v2.6/linux-2.6.28.tar.bz2 [Dec. 24, 2008].

67. Malkowski K, Raghavan P, Kandemir M, Irwin M. Phase-aware adaptive hardware selection for power-efficient scientific computations. *Proceedings of the International Symposium on Low Power Electronics and Design 2007* (*ISLPED'07*). Portland, Oregon, USA. 27–29 Aug., 2007; 403-406.

68. Freeh V, Lowenthal D, Pan F, Kappiah N. Using multiple energy gears in mpi programs on a power-scalable cluster. *Proceedings of the 2005 ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (*PPOPP'05*), Chicago, USA. 15-17 Jun., 2005; 164-173.

69. Messgae Passing Interface Forum. MPI: A Message-Passing Interface Standard (Version 2.1). http://www.mpi-forum.org/ [Jun. 23, 2008].

70. Lim M, Freeh V, Lowenthal D. Adaptive, transparent frequency and voltage scaling of communication phases in mpi programs. *Proceedings of the 20th ACM/IEEE Conference on Supercomputing* (*SC'06*). Tampa, Florida, USA. 11-17 Nov., 2006; 14-27.

71. Kappiah N, Freeh V, Lowenthal D. Just in time dynamic voltage scaling: exploiting inter-node slack to save energy in mpi programs. *Proceedings of the 19th ACM/IEEE Conference on Supercomputing* (*SC'05*). Washington, USA. 12-18 Nov., 2005; 33-41.

72. Rountree B, Lowenthal D, Supinski B, Schulz M, Freeh V, Bletsch T. Adagio: making DVS practical for complex HPC applications. *Proceedings of the 23th International Conference on Supercomputing* (*ICS'09*). Yorktown Heights, New York, USA. 8–12 Jun., 2009; 460-469.

73. LLNL, HP and Bull. The Simple Linux Utility for Resource Management (SLRUM). http://www.llnl.gov/linux/slurm/. revision 2.0.3 [Jun. 2009]

74. Koch K, Henning P. Beyond a single cell. *Cell Workshop*. University of Tennessee. Oct. 25, 2006.

75. Zong Z, Qin X, Ruan X, Bellam K, Nijim M, Alghamdi M. Energy-Efficient Scheduling for Parallel Applications Running on Heterogeneous Clusters. *Proceedings of the 36th International Conference on Parallel Processing* (*ICPP'07*). 10-14 Sept., 2007; 19-26.

76. Smith J, Nair R. The architecture of virtual machines. Computer 2005; **38**(5): 32-38.

77. Rosenblum M, Garfinkel T. Virtual machine monitors: current technology and future trends. Computer 2005; **38**(5): 39-47.

78. VMware. Understanding Full Virtualization, Paravirtualization, and Hardware Assist. *Whitepaper WP-028-PRD-01-01*. http://www.vmware.com/solutions/whitepapers/virtualization.html [Sep. 11, 2007].

79. Pratt I, Fraser K, Hand S, Limpach C, Warfield A. Xen 3.0 and the art of virtualization. *Proceedings of the 2005 Ottawa Linux Symposium*. Ottawa, Canada. 30 Jul., 2005; 65–85.

80. Sun. Solaris Containers: Server Virtualization and Manageability. *Sun White Paper*. http://www.sun.com/software/whitepapers/solaris10/grid_containers.pdf [Sep. 2004]

81. Soltesz S, Potzl H. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems* (*EuroSys'07*). Lisboa. Portugal. 21-23 Mar., 2007; 275-287.

82. Parallels. OpenVZ. http://www.openvz.org/ [Jul. 2009].

83. Lu K, Chi W, Liu Y, Tang H. HPVZ: a high performance virtual computing environment for super computer. *Proceedings of the 8th Advanced Parallel Processing Technology* (*APPT'09*). Rapperswil, Switzerland. 24 - 25 Aug., 2009.

84. Huang W, Liu J. A case for high performance computing with virtual machines. *Proceedings of 20th Annual International Conference on Supercomputing* (*SC'06*). Cairns, Queensland, Australia. 28-30 Jun., 2006; 125-134.

85. Figueiredo R, Dinda P, Fortes J. A case for grid computing on virtual machines. *Proceedings of the 23rd International Conference on Distributed Computing Systems* (*ICDCS'03*). Providence, Rhode Island USA. 19-22 May, 2003; 550–559.

86. Engelmann C, Scott S, Ong H, Configurable virtualized system environments for high performance computing. *Proceedings of the 1st Workshop on System-level Virtualization for High Performance Computing* (*HPCVirt'07*). Lisbon, Portugal. 20 Mar., 2007.

87. Mergen M, Uhlig V, Krieger O, Xenidis J. Virtualization for high-performance computing. *ACM SIGOPS Operating Systems Review* 2006; **40**(2):8-11.

88. Padala P, Zhu X, Wang Z, Singhal S, Shin K. Performance evaluation of virtualization technologies for server consolidation. *Technical Report HPL-2007-59*. HP Laboratories, Palo Alto. 11 Apr., 2007.

89. Milo D, Douglis F. Process migration. *ACM Computing Surveys* 2000; **32**(3):241-299.

90. Clark C, Fraser K, Hand S. Live migration of virtual machines. *Proceedings of the 2nd Conference on Symposium on Networked Systems Design and Implementation* (*NSDI'05*). Boston, MA, USA. Jul. 2005. Vol. 2: 273-286.

91. Stoess J, Klee C, Domthera S, Bellosa F. Transparent, power-aware migration in virtualized systems. *Proceedings of GI/ITG Fachgruppentreffen Betriebssysteme*. Karlsruhe, Germany. 12 Oct., 2007; 3-8.

92. Verma A., Ahuja P., Neogi A. pMapper: power and migration cost aware application placement in virtualized systems. *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware* (*Middleware'08*). Leuven, Belgium. 1-5 Dec., 2008; 243-264.

93. Verma A, Ahuja P, Neogi A.. Power-aware dynamic placement of HPC applications. *Proceedings of the 22nd International Conference on Supercomputing* (*ICS'08*). Island of Kos, Aegean Sea, Greece. 7–12 Jun., 2008; 175-184.

94. Stoess J, Lang C, Bellosa F. Energy management for hypervisor-based virtual machines. *Proceedings of the 2007 USENIX Annual Technical Conference* (*USENIX'07*). Santa Clara, CA. 17-22 Jun., 2007; 1-14.

95. Nathuji R, Schwan K. VirtualPower: Coordinated power management in virtualized enterprise systems. *Proceedings of the 21st ACM SIGOPS Symposium on Operating Systems Principles* (*SOSP'07*). Stevenson, Washington, USA. 14–17 Oct., 2007; 265-278.

96. Lefurgy C, Wang X, Ware M.. Server-level power control. *Proceedings of the 4th IEEE International Conference on Autonomic Computing* (*ICAC'07*). Jacksonville, Florida. 11-15 Jun., 2007; 4-13.

97. Dash R, Kephart J, Lefurgy C, Tesauro G, Levine D, Chan H. Autonomic Multi-Agent Management of Power and Performance in Data Centers. *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems* (*AAMAS'08*): Industrial Track. Estoril, Portugal. 12-16 May, 2008; 107-114.

Table 1 summary of power management techniques for high performance systems

Technique/ Tool	**Constraints Concerned**	**Mechanism**	**Profiling**	**Applicable Systems**
Load Concentration [21]	Acceptable performance	Node on/off		Homogenous clusters
Muse [20]	SLA	Node on/off	Analytical modeling	Homogenous clusters
Chan *et al*. [56]	SLA	Node on/off, CPU DVFS	Analytical modeling	Homogenous cluster
Horvath *et al.* [57,58]	SLA	Node on/off, CPU DVFS, Multiple sleep states	Analytical modeling	Multi-tiers clusters
Heath *et al*.[40]	Power/Throughput	Node on/off	Analytical modeling	Heterogeneous clusters
PDC [11]	Disk bandwidth	Multi-speed disk	Simulation	Disk arrays
MAID [17]	Request time	Spin-down/up	Simulation	Disk arrays
DIV [18]	Redundancy, Throughput	Disk active or standby	Analytical modeling	Disk arrays with redundancy
Ranganathan *et al.* [59]	SLA, Power budget	CPU DVFS	Simulation, Online measurement	Homogenous clusters
Femal *et al.* [61]	Power budget, Minimum service level	CPU DVFS, Device on/off	Analytical modeling	Homogenous clusters with non-uniform workload
MIMO [62]	Power budget	CPU DVFS	Analytical modeling, Online measurement	Homogenous clusters
Mercury and Freon [41]	Temperature threshold, Utilization threshold	Node on/off	Analytical modeling	Heterogeneous clusters
Moore *et al.* [64]	Cooling cost, Temperature threshold	Node on/off, CRAC supply temperature	Analytical modeling	Homogenous data centers
Hsu *et al.* [28]	Relative performance slowdown	CPU DVFS	Analytical modeling	HPC
Ge *et al* [29]	Relative performance slowdown	CPU DVFS	Analytical modeling	HPC
Freeh *et al.*[68]	(Energy saving) / (time delay)	CPU DVFS	Online measurement	HPC
Lim *et al*. [70]	E*D	CPU DVFS		HPC
Jitter [71]	Net slack	CPU DVFS		HPC
Adagio [72]	Slack	CPU DVFS	Analytical modeling, Online measurement	HPC
SLURM [73]				HPC
EETDS [75]	Performance, Energy	Node on/off	Analytical modeling	Heterogeneous HPC
pMapper [92, 93]	Power budget, Performance, Migration cost	Node on/off	Analytical modeling	Virtualized heterogeneous HPC
Stoess *et al.* [94]	Failure rate, Temperature constraints, Power budget, QoS	CPU throttling, Disk active/idle	PMC-based, Analytical modeling	Hypervisor-based virtual machines
VirtualPower [95]	SLA, Power budget, Power/ Performance	DVFS, Node on/sleep/off		Virtualized heterogeneous HPC