Improving Multicore Server Performance and Reducing Energy Consumption by Workload Dependent Dynamic Power Management

Keqin Li, *Fellow, IEEE*

Abstract—The technique of using workload dependent dynamic power management (i.e., variable power and speed of processor cores according to the current workload) to improve system performance and to reduce energy consumption is investigated. Typically, the power supply and the core speed are increased when there are more tasks in a server, such that tasks can be processed faster and the average task response time is reduced. On the other hand, the power supply and the core speed are decreased when there are less tasks in a server, such that energy consumption can be reduced without significant performance degradation. A queueing model of multicore server processors with workload dependent dynamic power management is established. Several speed schemes are proposed and it is demonstrated that for the same average power consumption, it is possible to design a multicore server processor with workload dependent dynamic power management, such that its average task response time is shorter than a multicore server processor of constant speed (i.e., without workload dependent dynamic power management). It is shown that given certain application environment and average power consumption, there is an optimal speed scheme that minimizes the average task response time. For two-speed schemes, the problem of optimal design of a two-speed scheme for given power supply and power consumption model is formulated and solved. It is pointed out that power consumption reduction subject to performance constraints can be studied in a similar way as performance improvement (i.e., average task response time reduction) subject to power consumption constraints. To the best of our knowledge, this is the first work on analytical study of workload dependent dynamic power management.

Index Terms—Dynamic power management, energy consumption, multicore server processor, queueing model, response time

1 Introduction

1.1 Motivation

Data centers have become the backbone of modern economy, from server rooms that power small- to medium-sized organizations, to server farms that support major companies and corporations, and to hyper-scale enterprise data centers that provide cloud computing services hosted by Amazon, Facebook, Google, and others [7]. However, the explosive development of information technologies has also made data centers one of the fastest-growing consumers of electricity. A typical 5,000 Ft^2 data center demands 1.127 MW electrical power [14]. In 2013, data centers in the US consumed approximately 91 billion KWH of electricity, equivalent to the annual output of 34 large (500 MW) coal-fired power plants. Data center electricity consumption is projected to increase to roughly 140 billion KWH annually by 2020, equivalent to the annual output of 50 power plants, costing American businesses 13 billion US dollars annually in electricity bills, and emitting nearly 100 million metric tons of carbon pollution per year [7]. Electricity consumption cost has become an increasingly significant fraction of the total cost of ownership of current and future data centers.

• *The author is with the Department of Computer Science, State University of New York, New Paltz, New York 12561. E-mail: lik@newpaltz.edu.*

Manuscript received 28 Nov. 2014; revised 3 Mar. 2015; accepted 13 May 2015. Date of publication 3 June 2015; date of current version 8 June 2016. Recommended for acceptance by C. Mastroianni, S.U. Khan, and R. Bianchini. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TCC.2015.2440238

It is possible to dramatically reduce electrical energy consumption of a typical data center through appropriate design of resource management strategies of physical infrastructures. The Advanced Configuration and Power Interface (ACPI), an open industry standard, allows an operating system to directly control the power-saving aspects of its underlying hardware [1]. Some programs allow users to adjust the voltages supplied to the CPU through a process called undervolting, which reduces both the amount of electricity consumed and heat produced. Some CPUs can automatically undervolt the processor, depending on the workload. Such a technology is called "SpeedStep" on Intel processors, "PowerNow!" and "Cool'n'Quiet" on AMD chips, LongHaul on VIA CPUs, and LongRun on Transmeta processors [3]. Reducing processor energy consumption has significant impact on the energy efficiency of a data center. With the cascade effect, one Watt reduction at the server component level (processors, memories, hard disks, etc.) results in an additional 1.84 Watts saving in the power supply, power distribution system, UPS system, cooling system, and building entrance switchgear and medium voltage transformer. Consequently, every Watt of saving that can be achieved on the processor level creates approximately 2.84 Watts of saving for the data center [14].

The technique of *workload dependent dynamic power management* refers to dynamic core power and speed adjustment according to the current workload, i.e., the number of applications in a server system, the distribution of the applications among the cores, and the characteristics of the applications. For instance, the power supply and the core speed are increased when there are more tasks in a server, such that tasks can be processed faster and the average task

response time is reduced. On the other hand, the power supply and the core speed are decreased when there are less tasks in a server, such that energy consumption can be reduced without significant performance degradation. Such runtime power and speed adjustment can be supported by a mechanism called *dynamic voltage scaling*, or equivalently, dynamic frequency scaling, dynamic speed scaling, or dynamic power scaling [2].

1.2 Related Work

Computational efficiency of data centers and cloud infrastructures has been considered together with energy-efficient and environment-conscientious resource scheduling and optimization. Recent survey of green data centers and energy-efficient cloud computing systems can be found in [8], [26]. An overview of energy efficiency techniques in cluster computing systems was provided in [25]. In [12], the authors proposed a method to route a request to the data center that is closest in terms of geographical distance, costs the least in electricity, and emits the smallest amount of carbon. In [21] the authors studied the problem of scheduling batch jobs for multiple geographically distributed data centers, and proposed a provably-efficient online scheduling algorithm, which optimizes the energy cost and fairness among different organizations, subject to queueing delay constraints, while satisfying the maximum server inlet temperature constraints. In [22], the authors proposed a data center-wide energy-efficient resource scheduling framework, which schedules data center resources according to the current workload of a data center, so that redundant servers can be powered off for energy economy. In [28], the authors developed a novel energy-aware scheduling algorithm for real-time, aperiodic, independent tasks, and proposed two strategies in terms of resource scaling up and scaling down to make a good trade-off between task schedulability and energy conservation.

Workload dependent dynamic power management has been studied by a number of researchers. For each group of applications, we should choose the lowest core speed so that the group of applications can be completed within certain specified performance constraints [20]. Dynamic power management can be carried out with different granularity at the application level and the phase (of an application) level. At the application (phase, respectively) level, the overall characteristics and behavior of an application (phase, respectively) is analyzed and the core speed is determined based on the characteristics. For instance, for CPU-bound applications (phase, respectively), the core speed should be high to reduce the execution time; however, for memory-bound applications (phase, respectively), the core speed should be low to save energy without increasing the execution time [9], [23].

Another workload related power management technique is called *workload placement*. The method is to maximize both spatial and temporal idleness through workload placement. In the space dimension, applications are placed on a few active cores which run at high speed, while other cores are in an idle and low power consuming state. In the time dimension, applications are scheduled to run together at high speed, so that cores can be in an idle and low power consuming state in other times. Such workload dependent dynamic power management involves observing the current core utilization and workload statistics, re-evaluating the number of required cores to achieve the projected performance, re-mapping the workload to the minimized set of active cores, and setting other inactive cores to a power-saving state [18]. It has been observed that such idle-maximization can reduce average power consumption [13].

On a heterogeneous multicore processor which contains a set of cores that achieve different performance levels and consume different power levels, it is required that an operating system should be able to choose the most appropriate core for an application based on the characteristics of the application, to meet its specific performance requirements and to minimize energy consumption [19].

The technique of workload dependent dynamic power management has been implemented by the current processor technologies. The Intel turbo boost technology is one of the many exciting new features that Intel has built into the latest generation Intel microarchitecture (Nehalem) based processors [6]. It automatically allows processor cores to run faster than the current operating frequency if it is operating below the expected performance goal. With Intel turbo boost technology, depending upon the workload, the power and frequency can be distributed across many cores or focused on fewer cores through the use of power gating—an internal power management unit. An Intel turbo boost technology enabled processor can boost performance on the core or cores still active by raising the clock speed. Intel turbo boost technology is activated when the operating system requests the highest processor performance state. The amount of time a processor spends in the Intel turbo boost technology state depends on the workload and operating environment, providing the performance you need [16]. When Intel hyper-threading technology and Intel turbo boost technology are combined in processors based on Intel microarchitecture, intelligent server processors deliver better performance by adapting to the workload and automatically disabling inactive cores and increasing processor frequency on remaining cores to get the tasks done quicker [5].

1.3 Our Contributions

It is clear that energy-aware server resource monitoring and control is a pressing research issue for data centers. However, although there has been much effort made by many researchers, there is no solid theoretical model and foundation for energy-aware dynamic workload monitoring and auditing in cloud computing. Despite the existing experimental research effort, there is lack of analytical modeling of any workload dependent dynamic power management method. In this paper, we investigate the technique of using workload dependent dynamic power management (i.e., variable power and speed of processor cores according to the current workload) to improve system performance and to reduce energy consumption. Although our modeling is at the multicore server processor level [4], [24], it is also applicable to the cluster system level in terms of [11] and the multi-server system level in terms of [15].

Our contributions in this paper as well as the organization of the paper are summarized as follows.

- We establish a queueing model (Section 2) of multicore server processors with workload dependent dynamic power management (Section 3).

- We propose several speed schemes and demonstrate that for the same average power consumption, it is possible to design a multicore server processor with workload dependent dynamic power management, such that its average task response time is shorter than a multicore server processor of constant speed (i.e., without workload dependent dynamic power management) (Sections 4.1, 4.2, and 4.3).
- We show that given certain application environment and average power consumption, there is an optimal speed scheme that minimizes the average task response time (Section 4.4).
- For two-speed schemes, we formulate and solve the problem of optimal design of a two-speed scheme for given power supply and power consumption model (Section 5).
- We point out that power consumption reduction subject to performance constraints can be studied in a similar way as performance improvement (i.e., average task response time reduction) subject to power consumption constraints (Section 6).

To the best of our knowledge, this is the first work on analytical study of workload dependent dynamic power management. (For reader's convenience, Section 1 of the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCC.2015.2440238, gives a summary of notations and their definitions in the order introduced in the paper.)

2 Modeling a Multicore Server Processor

2.1 A Queueing Model

Assume that a multicore server processor S has m identical cores. In this paper, a multicore server processor is treated as an M/M/m queueing system which is elaborated as follows. There is a Poisson stream of tasks with arrival rate λ, i.e., the inter-arrival times are independent and identically distributed (i.i.d.) exponential random variables with mean $1/\lambda$. A multicore server S maintains a queue with infinite capacity for waiting tasks when all the m cores are busy. The first-come-first-served (FCFS) queueing discipline is adopted. The task execution requirements (measured by the number of instructions to be executed) are i.i.d. exponential random variables r with mean $\bar{r}$. The m cores of server S have identical execution speed s (measured by the number of billion instructions that can be executed in one unit of time). Hence, the task execution times on the cores of server S are i.i.d. exponential random variables $x = r/s$ with mean $\bar{x} = \bar{r}/s$.

Let $\mu = 1/\bar{x} = s/\bar{r}$ be the average service rate, i.e., the average number of tasks that can be finished by a processor core of server S in one unit of time. The core utilization is $\rho = \lambda/m\mu = \lambda\bar{x}/m = (\lambda/m)\cdot(\bar{r}/s)$, which is the average percentage of time that a core of S is busy. Let p_k denote the probability that there are k tasks (waiting or being processed) in the M/M/m system for S. Then, we have ([17], p. 102)

$$p_k = \begin{cases} p_0 \frac{(m\rho)^k}{k!}, & k \le m; \\ p_0 \frac{m^m \rho^k}{m!}, & k \ge m; \end{cases}$$

where

$$p_0 = \left(\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!} \cdot \frac{1}{1-\rho} \right)^{-1}.$$

The probability of queueing (i.e., the probability that a newly arrived task must wait because all processor cores are busy) is

$$P_q = \sum_{k=m}^{\infty} p_k = \frac{p_m}{1-\rho} = p_0 \frac{(m\rho)^m}{m!} \cdot \frac{1}{1-\rho}.$$

The average number of tasks (in waiting or in execution) in S is

$$\bar{N} = \sum_{k=0}^{\infty} k p_k = m\rho + \frac{\rho}{1-\rho} P_q.$$

Applying Little's result, we get the average task response time as

$$T = \frac{\bar{N}}{\lambda} = \bar{x} + \frac{P_q}{m(1-\rho)}\bar{x} = \bar{x}\left(1 + \frac{p_m}{m(1-\rho)^2}\right).$$

2.2 Power Consumption Models

Power dissipation and circuit delay in digital CMOS circuits can be accurately modeled by simple equations, even for complex microprocessor circuits. CMOS circuits have dynamic, static, and short-circuit power dissipation; however, the dominant component in a well designed circuit is dynamic power consumption P (i.e., the switching component of power), which is approximately $P = wCV^2 f$, where w is an activity factor, C is the loading capacitance, V is the supply voltage, and f is the clock frequency [10]. Since $s \propto f$, where s is the processor speed, and $f \propto V^{\phi}$ with $0 < \phi \le 1$ [27], which implies that $V \propto f^{1/\phi}$, we know that power consumption is $P \propto f^{\alpha}$ and $P \propto s^{\alpha}$, where $\alpha = 1 + 2/\phi \ge 3$. For ease of discussion, we will assume that the power allocated to a processor core with speed s is simply s^{α}.

We will consider two types of core speed models. In the *idle-speed model*, a core runs at zero speed when there is no task to perform. Since the power for speed s is s^{α}, the average amount of energy consumed by a core in one unit of time is $\rho s^{\alpha} = (\lambda/m)\bar{r}s^{\alpha-1}$, where we notice that the speed of a core is zero when it is idle. The average amount of energy consumed by an m-core server S in one unit of time, i.e., the power supply to server S, is $P = m\rho s^{\alpha} = \lambda\bar{r}s^{\alpha-1}$, where $m\rho = \lambda\bar{x}$ is the average number of busy cores in S. Since a processor core still consumes some amount of power P^* (e.g., static power dissipation) even when it is idle, we will include P^* in P, i.e., $P = m(\rho s^{\alpha} + P^*) = \lambda\bar{r}s^{\alpha-1} + mP^*$. Notice that when $P^* = 0$, the above P is independent of m.

In the *constant-speed model*, all cores run at the speed s even if there is no task to perform. Again, we use P to represent the power allocated to server S. Since the power for speed s is s^{α}, the power allocated to server S is $P = m(s^{\alpha} + P^*)$.

3 Workload Dependent Dynamic Power Management

The technique of *workload dependent dynamic power management* refers to dynamic core (server) power and speed

adjustment according to the current workload (i.e., the number of tasks in the server system). A multicore server processor with workload dependent dynamic power management can be characterized by a birth-death process ([17], p. 53). The states are $0, 1, 2, \ldots, k, \ldots$, where state k means that there are k tasks in the server system. The birth rate (i.e., the task arrival rate) is fixed at λ. The death rates (i.e., the task service rates) are μ_k with $k \geq 1$. Let the speed of the m cores be s_k when there are $k \geq 1$ tasks in the queueing system. Then, we have

$$\mu_k = \begin{cases} k\frac{s_k}{\bar{r}}, & 1 \leq k \leq m-1; \\ m\frac{s_k}{\bar{r}}, & k \geq m. \end{cases}$$

This implies that ([17], p. 92)

$$\begin{aligned} p_k &= p_0 \frac{\lambda^k}{\mu_1 \mu_2 \cdots \mu_k} \\ &= \begin{cases} p_0 \frac{(\lambda\bar{r})^k}{k!} \cdot \prod_{j=1}^{k} \frac{1}{s_j}, & 1 \leq k \leq m-1; \\ p_0 \frac{(\lambda\bar{r})^k}{m!m^{k-m}} \cdot \prod_{j=1}^{k} \frac{1}{s_j}, & k \geq m; \end{cases} \end{aligned}$$

where

$$p_0 = \left(1 + \sum_{k=1}^{m-1} \frac{(\lambda\bar{r})^k}{k!} \cdot \prod_{j=1}^{k} \frac{1}{s_j} + \sum_{k=m}^{\infty} \frac{(\lambda\bar{r})^k}{m!m^{k-m}} \cdot \prod_{j=1}^{k} \frac{1}{s_j}\right)^{-1}.$$

Based on the p_k's, we get the average number of tasks (in waiting or in execution) as

$$\bar{N} = \sum_{k=1}^{\infty} k p_k.$$

By Little's result, the average task response time is $T = \bar{N}/\lambda$. The average number of busy servers is

$$B = \sum_{k=0}^{m-1} k p_k + \sum_{k=m}^{\infty} m p_k,$$

and the average server utilization is $\rho = B/m$. The average core speed is

$$\bar{s} = \sum_{k=0}^{\infty} p_k s_k.$$

The average power consumption is

$$\begin{aligned} P &= \sum_{k=0}^{m-1} p_k (k(s_k^\alpha + P^*) + (m-k)P^*) + \sum_{k=m}^{\infty} p_k m(s_k^\alpha + P^*) \\ &= \sum_{k=0}^{m-1} p_k (ks_k^\alpha + mP^*) + \sum_{k=m}^{\infty} p_k (ms_k^\alpha + mP^*) \\ &= \sum_{k=0}^{m-1} p_k k s_k^\alpha + \sum_{k=m}^{\infty} p_k m s_k^\alpha + mP^*, \end{aligned}$$

for the idle-speed model, and

$$P = \sum_{k=0}^{\infty} p_k m(s_k^\alpha + P^*) = m \sum_{k=0}^{\infty} p_k s_k^\alpha + mP^*,$$

for the constant-speed model.

4 Speed Schemes

Definition 1. *The sequence of core speeds $(s_1, s_2, s_3, \ldots)$ is called a* speed scheme. *A speed scheme is valid if it results in a stable queueing system, i.e., $p_0 > 0$.*

A speed scheme reflects and represents a strategy of workload dependent dynamic power management. If $s_1 = s_2 = s_3 = \cdots = s$, then we have a *single-speed scheme* for workload independent dynamic power management, i.e., a standard M/M/m queueing system.

Define $\rho_k = \lambda\bar{r}/ms_k$ for all $k \geq 1$.

Theorem 1. *A speed scheme $(s_1, s_2, s_3, \ldots)$ is valid if and only if the summation*

$$\sum_{k=m}^{\infty} \rho_1 \rho_2 \cdots \rho_k$$

converges.

Due to space limitation, all proofs are given in Section 2 of the supplementary material, available online.

Corollary 1. *If there is some $n \geq 1$, such that $\rho_k < 1$ for all $k \geq n$, then a speed scheme $(s_1, s_2, s_3, \ldots)$ is valid.*

In this section, we give several examples of speed schemes for workload dependent dynamic power management. Our main purpose is to compare the performance of workload dependent dynamic power management with workload independent dynamic power management. Our method of comparison is as follows. Assume that certain speed scheme consumes power P. We can also design a multicore server processor with a constant single speed s (notice that $s \neq \bar{s}$) such that its power consumption is the same as P, i.e., that of the above multicore server processor with workload dependent dynamic power management. We then compare their average task response times.

4.1 Arithmetic-Speed Schemes

Definition 2. *An* arithmetic-speed scheme *with parameters (c, d) is defined in such a way that the sequence of speeds $s_1, s_2, s_3, \ldots$ form an arithmetic progression, i.e., $s_k = c + kd$ for all $k \geq 1$, where c and d are two constants.*

Corollary 2. *Any arithmetic-speed scheme with $c > \lambda\bar{r}/m$ or $d > 0$ is valid.*

Then, we have

$$\mu_k = \begin{cases} k\left(\frac{c+kd}{\bar{r}}\right), & 1 \leq k \leq m-1; \\ m\left(\frac{c+kd}{\bar{r}}\right), & k \geq m. \end{cases}$$

The philosophy of an arithmetic-speed scheme is that the speed of the cores (servers) should be increased if there are more tasks in the system, so that the task response times can be reduced; and that the speed of the cores (servers) should be decreased if there are fewer tasks in the system, so that power consumption can be reduced without significant increment of task response times. Such a shift of power consumption from lightly utilized circumstance to heavily utilized circumstance can reduce the average task response time without increment of the average power consumption.

TABLE 1
Numerical Data for Arithmetic-Speed Schemes (Idle-Speed Model)

	Arithmetic-Speed Scheme			Single-Speed Scheme			
λ	ρ	$\bar{s}$	T	ρ	s	T	P
			$d = 0.01$				
1.0	0.1400955	1.0098068	0.9806785	0.1400760	1.0198544	0.9805444	15.0401030
2.0	0.2775919	1.0194446	0.9722297	0.2775171	1.0295377	0.9720989	16.1198956
3.0	0.4125684	1.0290674	0.9689148	0.4124019	1.0392083	0.9696054	17.2398615
4.0	0.5448102	1.0392667	0.9816663	0.5444904	1.0494741	0.9858179	18.4055834
5.0	0.6732601	1.0515950	1.0318998	0.6726279	1.0619330	1.0479929	19.6385081
6.0	0.7946822	1.0698623	1.1643713	0.7932693	1.0805194	1.2272890	21.0051334
7.0	0.9002268	1.1036689	1.4809844	0.8967337	1.1151582	1.7794541	22.7050450
8.0	0.9712818	1.1727667	2.1595835	0.9636753	1.1859359	3.8051765	25.2515518
9.0	0.9967105	1.2891468	3.2127421	0.9861568	1.3037626	8.3547994	29.2981712
10.0	0.9998780	1.4286989	4.2869893	0.9895633	1.4436383	9.8777782	34.8409144
			$d = 0.02$				
1.0	0.1375089	1.0192514	0.9625692	0.1374375	1.0394333	0.9620734	15.0804216
2.0	0.2702625	1.0378553	0.9463818	0.2699998	1.0582019	0.9456533	16.2395824
3.0	0.3986444	1.0560702	0.9345041	0.3980867	1.0765781	0.9346972	17.4770611
4.0	0.5226576	1.0746932	0.9336647	0.5216627	1.0953987	0.9379273	18.7995929
5.0	0.6414613	1.0955120	0.9551204	0.6397330	1.1165372	0.9713139	20.2332764
6.0	0.7526194	1.1219942	1.0166182	0.7494946	1.1436278	1.0669875	21.8473068
7.0	0.8509488	1.1604714	1.1462241	0.8451030	1.1832877	1.3015093	23.8011882
8.0	0.9278242	1.2209176	1.3807348	0.9174015	1.2457545	1.8654789	26.4152350
9.0	0.9749401	1.3129189	1.7384383	0.9593229	1.3402310	3.0541280	30.1659720
10.0	0.9943709	1.4347013	2.1735064	0.9759561	1.4637661	4.4526516	35.4261132
			$d = 0.03$				
1.0	0.1350765	1.0283662	0.9455402	0.1349290	1.0587579	0.9445123	15.1209684
2.0	0.2635874	1.0553734	0.9228895	0.2630636	1.0861036	0.9212696	16.3592419
3.0	0.3863215	1.0814062	0.9045130	0.3852502	1.1124496	0.9036223	17.7126326
4.0	0.5036165	1.1073645	0.8947041	0.5018000	1.1387576	0.8976445	19.1870752
5.0	0.6150337	1.1350225	0.9001501	0.6121158	1.1669127	0.9134287	20.8084261
6.0	0.7189435	1.1673839	0.9299108	0.7142291	1.2000952	0.9682527	22.6413710
7.0	0.8120735	1.2091289	0.9958517	0.8043563	1.2432302	1.0951624	24.8193487
8.0	0.8894046	1.2667499	1.1114581	0.8770804	1.3030244	1.3580160	27.5829815
9.0	0.9455848	1.3470068	1.2852105	0.9275754	1.3861022	1.8495515	31.2915142
10.0	0.9787598	1.4526054	1.5086848	0.9559374	1.4944194	2.5591524	36.3328938
			$d = 0.04$				
1.0	0.1327811	1.0371788	0.9294711	0.1325394	1.0778466	0.9277840	15.1617534
2.0	0.2574623	1.0721091	0.9013633	0.2566303	1.1133302	0.8986675	16.4790082
3.0	0.3752766	1.1053494	0.8779114	0.3736261	1.1470596	0.8756435	17.9472372
4.0	0.4869122	1.1378890	0.8618062	0.4842175	1.1801073	0.8628829	19.5706126
5.0	0.5923226	1.1714476	0.8572380	0.5882177	1.2143220	0.8670003	21.3728891
6.0	0.6905073	1.2086655	0.8694395	0.6843335	1.2525222	0.8981966	23.4128705
7.0	0.7793374	1.2532657	0.9045204	0.7700321	1.2986471	0.9734406	25.8053898
8.0	0.8556023	1.3100225	0.9688203	0.8417880	1.3576544	1.1209254	28.7458034
9.0	0.9157085	1.3839946	1.0666516	0.8962369	1.4345697	1.3787466	32.5219124
10.0	0.9574127	1.4785230	1.1963076	0.9323142	1.5322855	1.7622513	37.4789876

This is exactly the idea of workload dependent dynamic power management.

The definition of the s_k's implies that

$$p_k = \begin{cases} p_0 \frac{(\lambda\bar{r})^k}{k!} \cdot \prod_{j=1}^{k} \frac{1}{(c+jd)}, & 1 \le k \le m-1; \\ p_0 \frac{(\lambda\bar{r})^k}{m!m^{k-m}} \cdot \prod_{j=1}^{k} \frac{1}{(c+jd)}, & k \ge m; \end{cases}$$

where p_0 is

$$\left(1 + \sum_{k=1}^{m-1} \frac{(\lambda\bar{r})^k}{k!} \prod_{j=1}^{k} \frac{1}{c+jd} + \sum_{k=m}^{\infty} \frac{(\lambda\bar{r})^k}{m!m^{k-m}} \prod_{j=1}^{k} \frac{1}{c+jd}\right)^{-1}.$$

In Table 1, we demonstrate numerical data for arithmetic-speed schemes with the idle-speed model. We assume that there are $m = 7$ cores with base power supply $P^* = 2$ Watts. The task arrival rate λ is $1, 2, \ldots, 10$ per second. The average task execution requirement is $\bar{r} = 1$ giga instructions. The speed schemes are $c = 1$ and $d = 0.01$,0.02, 0.03,0.04, i.e., $s_k = c + kd$ giga instructions per second, for all $k \ge 1$, For each combination of λ and d, we show the average server utilization ρ, the average core speed $\bar{s}$, the average task response time T (in seconds), and the average power consumption P (in Watts). We also find the speed s of a single-speed scheme which consumes power P. It is easy to see that $s = ((P - mP^*)/\lambda\bar{r})^{1/(\alpha-1)}$. For such

TABLE 2
Numerical Data for Arithmetic-Speed Schemes (Constant-Speed Model)

	Arithmetic-Speed Scheme			Single-Speed Scheme			
λ	ρ	$\bar{s}$	T	ρ	s	T	P
			$d = 0.01$				
1.0	0.1400955	1.0098068	0.9806785	0.1414563	1.0099033	0.9902070	21.2100355
2.0	0.2775919	1.0194446	0.9722297	0.2802128	1.0196331	0.9815878	21.4204434
3.0	0.4125684	1.0290674	0.9689148	0.4163504	1.0293528	0.9792866	21.6346799
4.0	0.5448102	1.0392667	0.9816663	0.5496156	1.0396877	0.9968402	21.8669564
5.0	0.6732601	1.0515950	1.0318998	0.6787960	1.0522833	1.0635284	22.1563551
6.0	0.7946822	1.0698623	1.1643713	0.8001655	1.0712069	1.2580522	22.6043519
7.0	0.9002268	1.1036689	1.4809844	0.9035673	1.1067244	1.8796149	23.4889134
8.0	0.9712818	1.1727667	2.1595835	0.9691544	1.1792312	4.4177858	25.4787590
9.0	0.9967105	1.2891468	3.2127421	0.9901854	1.2984582	11.6502314	29.3243453
10.0	0.9998780	1.4286989	4.2869893	0.9930145	1.4386209	14.6123271	34.8418897
			$d = 0.02$				
1.0	0.1375089	1.0192514	0.9625692	0.1401076	1.0196246	0.9807654	21.4202574
2.0	0.2702625	1.0378553	0.9463818	0.2751047	1.0385658	0.9636097	21.8415168
3.0	0.3986444	1.0560702	0.9345041	0.4054175	1.0571113	0.9525379	22.2691362
4.0	0.5226576	1.0746932	0.9336647	0.5309975	1.0761416	0.9572866	22.7238130
5.0	0.6414613	1.0955120	0.9551204	0.6507560	1.0976244	0.9959542	23.2567667
6.0	0.7526194	1.1219942	1.0166182	0.7616502	1.1253759	1.1064441	23.9767918
7.0	0.8509488	1.1604714	1.1462241	0.8573995	1.1663175	1.3851680	25.1057624
8.0	0.9278242	1.2209176	1.3807348	0.9284521	1.2309274	2.1021243	27.0555543
9.0	0.9749401	1.3129189	1.7384383	0.9682273	1.3279054	3.8216131	30.3907721
10.0	0.9943709	1.4347013	2.1735064	0.9831269	1.4530895	6.2215940	35.4770746
			$d = 0.03$				
1.0	0.1350765	1.0283662	0.9455402	0.1388068	1.0291795	0.9716593	21.6308233
2.0	0.2635874	1.0553734	0.9228895	0.2703358	1.0568867	0.9468348	22.2638681
3.0	0.3863215	1.0814062	0.9045130	0.3955190	1.0835672	0.9284651	22.9056487
4.0	0.5036165	1.1073645	0.8947041	0.5146835	1.1102525	0.9236388	23.5799507
5.0	0.6150337	1.1350225	0.9001501	0.6271424	1.1389531	0.9442915	24.3422618
6.0	0.7189435	1.1673839	0.9299108	0.7306988	1.1730453	1.0118003	25.2990631
7.0	0.8120735	1.2091289	0.9958517	0.8212093	1.2177164	1.1700004	26.6396926
8.0	0.8894046	1.2667499	1.1114581	0.8929496	1.2798674	1.5131941	28.6755021
9.0	0.9455848	1.3470068	1.2852105	0.9413042	1.3658861	2.2115446	31.8378070
10.0	0.9787598	1.4526054	1.5086848	0.9672947	1.4768730	3.3495175	36.5490096
			$d = 0.04$				
1.0	0.1327811	1.0371788	0.9294711	0.1375503	1.0385812	0.9628628	21.8418652
2.0	0.2574623	1.0721091	0.9013633	0.2658633	1.0746660	0.9311100	22.6879756
3.0	0.3752766	1.1053494	0.8779114	0.3864749	1.1089245	0.9065785	23.5456162
4.0	0.4869122	1.1378890	0.8618062	0.5001445	1.1425269	0.8943379	24.4399253
5.0	0.5923226	1.1714476	0.8572380	0.6066270	1.1774711	0.9024947	25.4274349
6.0	0.6905073	1.2086655	0.8694395	0.7044379	1.2167757	0.9441525	26.6104211
7.0	0.7793374	1.2532657	0.9045204	0.7907564	1.2646120	1.0427377	28.1569590
8.0	0.8556023	1.3100225	0.9688203	0.8618155	1.3261042	1.2413165	30.3241649
9.0	0.9157085	1.3839946	1.0666516	0.9143427	1.4061623	1.6094428	33.4627602
10.0	0.9574127	1.4785230	1.1963076	0.9478859	1.5071132	2.2068990	37.9626966

a single-speed scheme, we also show the average server utilization ρ and the average task response time T. It is observed that as λ increases, P increases because of the increased $\bar{s}$ and s. The speed s of a single-speed scheme is greater than the average speed $\bar{s}$ of the corresponding arithmetic-speed scheme. It is also observed that except for very small λ, the average task response time T of a single-speed scheme is greater than that of the corresponding arithmetic-speed scheme, especially when λ is large and ρ is close to 1. This means that for a heavily loaded server, a workload dependent dynamic power management scheme is able to improve the server performance significantly without increasing energy consumption.

In Table 2, we demonstrate numerical data for arithmetic-speed schemes with the constant-speed model. All the parameters are set in the same way as Table 1. The speed s of a single-speed scheme which consumes power P is $s = (P/m - P^*)^{1/\alpha}$. Our observations in Table 2 are similar to those in Table 1. In fact, the average task response time T of an arithmetic-speed scheme is consistently shorter than that of the corresponding single-speed scheme.

4.2 Geometric-Speed Schemes

Definition 3. *A* geometric-speed scheme *with parameters* (q, a) *is defined in such a way that the sequence of speeds* $s_1, s_2, s_3, \ldots$ *form a geometric progression, i.e.,* $s_k = qa^k$ *for all* $k \geq 1$, *where* q *and* $a \geq 1$ *are two constants.*

TABLE 3
Numerical Data for Geometric-Speed Schemes (Idle-Speed Model)

	Geometric-Speed Scheme			Single-Speed Scheme			
λ	ρ	$\bar{s}$	T	ρ	s	T	P
			$a = 1.01$				
1.0	0.1400761	1.0098532	0.9805426	0.1400559	1.0200012	0.9804032	15.0404025
2.0	0.2774929	1.0196262	0.9718765	0.2774138	1.0299210	0.9717353	16.1214746
3.0	0.4122991	1.0294731	0.9682090	0.4121197	1.0399199	0.9689144	17.2443002
4.0	0.5442389	1.0400091	0.9802178	0.5438871	1.0506383	0.9845273	18.4153632
5.0	0.6721738	1.0528659	1.0283589	0.6714615	1.0637777	1.0451006	19.6581149
6.0	0.7927126	1.0720192	1.1533751	0.7910841	1.0835041	1.2179178	21.0438873
7.0	0.8969668	1.1071236	1.4392599	0.8928904	1.1199583	1.7286075	22.7801464
8.0	0.9676224	1.1766016	2.0038880	0.9586207	1.1921891	3.3835681	25.3705195
9.0	0.9951257	1.2908045	2.7998110	0.9816853	1.3097010	6.3942395	29.4378507
10.0	0.9996778	1.4289085	3.5373180	0.9849849	1.4503485	6.9553099	35.0351088
			$a = 1.02$				
1.0	0.1374383	1.0194242	0.9620747	0.1373615	1.0400089	0.9615409	15.0816186
2.0	0.2699165	1.0385169	0.9451536	0.2696247	1.0596741	0.9443342	16.2458185
3.0	0.3977417	1.0575093	0.9322104	0.3971027	1.0792457	0.9323080	17.4943140
4.0	0.5208434	1.0772156	0.9295069	0.5196681	1.0996030	0.9338281	18.8365073
5.0	0.6382874	1.0995064	0.9469640	0.6361844	1.1227653	0.9635821	20.3030095
6.0	0.7475547	1.1279520	0.9987391	0.7436643	1.1525939	1.0491673	21.9708358
7.0	0.8436464	1.1687221	1.1037238	0.8362942	1.1957515	1.2488748	24.0087509
8.0	0.9190260	1.2306606	1.2797948	0.9058662	1.2616180	1.6769413	26.7334406
9.0	0.9672664	1.3213506	1.5239037	0.9470042	1.3576649	2.4164369	30.5892852
10.0	0.9901430	1.4393381	1.7946832	0.9643937	1.4813156	3.0998271	35.9429590
			$a = 1.03$				
1.0	0.1349312	1.0287293	0.9445233	0.1347672	1.0600289	0.9433798	15.1236612
2.0	0.2629006	1.0567383	0.9204601	0.2622926	1.0892959	0.9185604	16.3731311
3.0	0.3845883	1.0843158	0.9002023	0.3832951	1.1181238	0.8989064	17.7506026
4.0	0.5002604	1.1123262	0.8875133	0.4979863	1.1474785	0.8900381	19.2668280
5.0	0.6094280	1.1425640	0.8877682	0.6056558	1.1793592	0.9005777	20.9544406
6.0	0.7104877	1.1780242	0.9072556	0.7042530	1.2170950	0.9437092	22.8879213
7.0	0.8004787	1.2230939	0.9524496	0.7901782	1.2655374	1.0406429	25.2110942
8.0	0.8753298	1.2832707	1.0280052	0.8588727	1.3306478	1.2216012	28.1649891
9.0	0.9312544	1.3636080	1.1326980	0.9069745	1.4175859	1.5049430	32.0859487
10.0	0.9672516	1.4658647	1.2558325	0.9352087	1.5275429	1.8288546	37.3338727
			$a = 1.04$				
1.0	0.1325441	1.0377832	0.9278122	0.1322670	1.0800662	0.9258772	15.1665430
2.0	0.2563763	1.0743457	0.8975316	0.2553719	1.1188163	0.8942478	16.5034997
3.0	0.3726110	1.1100430	0.8713818	0.3705258	1.1566574	0.8682044	18.0135691
4.0	0.4818921	1.1457428	0.8515474	0.4783470	1.1945899	0.8514407	19.7081806
5.0	0.5841856	1.1830930	0.8410529	0.5786011	1.2345046	0.8491106	21.6200086
6.0	0.6786143	1.2246131	0.8430241	0.6700225	1.2792748	0.8679598	23.8192640
7.0	0.7634339	1.2736483	0.8601173	0.7503665	1.3326820	0.9168953	26.4322901
8.0	0.8362814	1.3340723	0.8937673	0.8169015	1.3990147	1.0059704	29.6579369
9.0	0.8948602	1.4095033	0.9430966	0.8675102	1.4820740	1.1394109	33.7688890
10.0	0.9380167	1.5020436	1.0040363	0.9020423	1.5837079	1.2992547	39.0813063

Corollary 3. *Any geometric-speed scheme with $q > \lambda\bar{r}/m$ or $a > 1$ is valid.*

Then, we have

$$\mu_k = \begin{cases} k\frac{qa^k}{\bar{r}}, & 1 \le k \le m-1; \\ m\frac{qa^k}{\bar{r}}, & k \ge m. \end{cases}$$

This implies that

$$p_k = \begin{cases} p_0 \frac{(\lambda\bar{r})^k}{k!} \cdot \frac{1}{q^k a^{k(k+1)/2}}, & 1 \le k \le m-1; \\ p_0 \frac{(\lambda\bar{r})^k}{m!m^{k-m}} \cdot \frac{1}{q^k a^{k(k+1)/2}}, & k \ge m; \end{cases}$$

where p_0 is

$$\left(1 + \sum_{k=1}^{m-1} \frac{(\lambda\bar{r})^k}{k!} \cdot \frac{1}{q^k a^{k(k+1)/2}} + \sum_{k=m}^{\infty} \frac{(\lambda\bar{r})^k}{m!m^{k-m}} \cdot \frac{1}{q^k a^{k(k+1)/2}}\right)^{-1}.$$

In Table 3, we demonstrate numerical data for geometric-speed schemes with the idle-speed model. We assume that there are $m = 7$ cores. The task arrival rate λ is 1, 2, ..., 10. The speed schemes are $q = 1$ and $a = 1.01$,1.02,1.03,1.04. For each combination of λ and a, we show the average server utilization ρ, the average core speed $\bar{s}$, the average task response time T, and the average power consumption P. We also find the speed s of a single-speed scheme which

TABLE 4
Numerical Data for Geometric-Speed Schemes (Constant-Speed Model)

	Geometric-Speed Scheme			Single-Speed Scheme			
λ	ρ	$\bar{s}$	T	ρ	s	T	P
			$a = 1.01$				
1.0	0.1400761	1.0098532	0.9805426	0.1414495	1.0099516	0.9901597	21.2110695
2.0	0.2774929	1.0196262	0.9718765	0.2801609	1.0198220	0.9814050	21.4245681
3.0	0.4122991	1.0294731	0.9682090	0.4161795	1.0297754	0.9788672	21.6440867
4.0	0.5442389	1.0400091	0.9802178	0.5492048	1.0404654	0.9959527	21.8846237
5.0	0.6721738	1.0528659	1.0283589	0.6779270	1.0536322	1.0613146	22.1877615
6.0	0.7927126	1.0720192	1.1533751	0.7984133	1.0735578	1.2500586	22.6611252
7.0	0.8969668	1.1071236	1.4392599	0.9003443	1.1106862	1.8307022	23.5911832
8.0	0.9676224	1.1766016	2.0038880	0.9650117	1.1842936	3.9369406	25.6272283
9.0	0.9951257	1.2908045	2.7998110	0.9869249	1.3027478	8.8265032	29.4767261
10.0	0.9996778	1.4289085	3.5373180	0.9899242	1.4431119	10.2210676	35.0376914
			$a = 1.02$				
1.0	0.1374383	1.0194242	0.9620747	0.1400819	1.0198115	0.9805857	21.4243383
2.0	0.2699165	1.0385169	0.9451536	0.2749155	1.0392804	0.9629441	21.8577153
3.0	0.3977417	1.0575093	0.9322104	0.4048217	1.0586672	0.9510851	22.3057026
4.0	0.5208434	1.0772156	0.9295069	0.5296471	1.0788854	0.9544677	22.7907108
5.0	0.6382874	1.0995064	0.9469640	0.6481532	1.1020323	0.9900475	23.3687358
6.0	0.7475547	1.1279520	0.9987391	0.7571090	1.1321260	1.0913183	24.1573958
7.0	0.8436464	1.1687221	1.1037238	0.8502797	1.1760837	1.3351459	25.3870881
8.0	0.9190260	1.2306606	1.2797948	0.9191126	1.2434354	1.8979444	27.4576049
9.0	0.9672664	1.3213506	1.5239037	0.9586773	1.3411336	3.0113073	30.8855088
10.0	0.9901430	1.4393381	1.7946832	0.9747872	1.4655213	4.2596189	36.0330396
			$a = 1.03$				
1.0	0.1349312	1.0287293	0.9445233	0.1387518	1.0295874	0.9712743	21.6399005
2.0	0.2629006	1.0567383	0.9204601	0.2699454	1.0584152	0.9454619	22.2997744
3.0	0.3845883	1.0843158	0.9002023	0.3943338	1.0868240	0.9255912	22.9861926
4.0	0.5002604	1.1123262	0.8875133	0.5121093	1.1158332	0.9184066	23.7251414
5.0	0.6094280	1.1425640	0.8877682	0.6224467	1.1475452	0.9344936	24.5780940
6.0	0.7104877	1.1780242	0.9072556	0.7230510	1.1854528	0.9910761	25.6614040
7.0	0.8004787	1.2230939	0.9524496	0.8099711	1.2346119	1.1187571	27.1731445
8.0	0.8753298	1.2832707	1.0280052	0.8783873	1.3010857	1.3693135	29.4175625
9.0	0.9312544	1.3636080	1.1326980	0.9251438	1.3897453	1.7991517	32.7889998
10.0	0.9672516	1.4658647	1.2558325	0.9517265	1.5010315	2.3603359	37.6737733
			$a = 1.04$				
1.0	0.1325441	1.0377832	0.9278122	0.1374570	1.0392861	0.9622097	21.8578436
2.0	0.2563763	1.0743457	0.8975316	0.2652232	1.0772598	0.9288599	22.7510345
3.0	0.3726110	1.1100430	0.8713818	0.3845919	1.1143538	0.9020339	23.6865111
4.0	0.4818921	1.1457428	0.8515474	0.4961878	1.1516376	0.8864644	24.6916702
5.0	0.5841856	1.1830930	0.8410529	0.5996754	1.1911207	0.8888863	25.8294714
6.0	0.6786143	1.2246131	0.8430241	0.6935729	1.2358367	0.9187698	27.2123884
7.0	0.7634339	1.2736483	0.8601173	0.7753532	1.2897348	0.9901737	29.0175585
8.0	0.8362814	1.3340723	0.8937673	0.8420849	1.3571757	1.1224995	31.4987184
9.0	0.8948602	1.4095033	0.9430966	0.8917579	1.4417751	1.3333412	34.9792812
10.0	0.9380167	1.5020436	1.0040363	0.9247200	1.5448692	1.6116262	39.8091182

consumes power P. For such a single-speed scheme, we also show the average server utilization ρ and the average task response time T. In Table 4, we demonstrate numerical data for geometric-speed schemes with the constant-speed model. All the parameters are set in the same way as Table 3. Our observations in Tables 3-4 are similar to those in Tables 1-2.

4.3 Two-Speed Schemes

Definition 4. *A* two-speed scheme *with parameters* (b, s_1, s_2) *is defined in such a way that the speed of the* m *cores is* s_1 *when there are* $k \leq b$ *tasks, and* s_2 *when there are* $k \geq b+1$ *tasks, where* $b \geq m$.

Corollary 4. *Any two-speed scheme with* $\rho_2 < 1$ *is valid.*

Then, we have

$$\mu_k = \begin{cases} k\frac{s_1}{\bar{r}}, & 1 \leq k \leq m-1; \\ m\frac{s_1}{\bar{r}}, & m \leq k \leq b; \\ m\frac{s_2}{\bar{r}}, & k \geq b+1. \end{cases}$$

This implies that

$$p_k = \begin{cases} p_0 \frac{(\lambda\bar{r})^k}{k!} \cdot \frac{1}{s_1^k}, & 1 \leq k \leq m-1; \\ p_0 \frac{(\lambda\bar{r})^k}{m!m^{k-m}} \cdot \frac{1}{s_1^k}, & m \leq k \leq b; \\ p_0 \frac{(\lambda\bar{r})^k}{m!m^{k-m}} \cdot \frac{1}{s_1^b s_2^{k-b}}, & k \geq b+1. \end{cases}$$

TABLE 5
Numerical Data for Two-Speed Schemes (Idle-Speed Model)

	Two-Speed Scheme			Single-Speed Scheme			
λ	ρ	$\bar{s}$	T	ρ	s	T	P
				$s_2 = 1.45$			
1.0	0.1428571	1.0000000	1.0000141	0.1428571	1.0000001	1.0000141	15.0000003
2.0	0.2857054	1.0000089	1.0009024	0.2856986	1.0000550	1.0009067	16.0002201
3.0	0.4282515	1.0003199	1.0077055	0.4280043	1.0013250	1.0080078	17.0079551
4.0	0.5682428	1.0031857	1.0286766	0.5658526	1.0098541	1.0325054	18.0792215
5.0	0.6987374	1.0155483	1.0678159	0.6881731	1.0379448	1.0879676	19.3866468
6.0	0.8091442	1.0479986	1.1231273	0.7828085	1.0949586	1.1839371	21.1936058
7.0	0.8917791	1.1082209	1.1983095	0.8498859	1.1766285	1.3325106	23.6911828
8.0	0.9466852	1.1961720	1.3298552	0.9007568	1.2687743	1.6071908	26.8783061
9.0	0.9797433	1.3059710	1.7052809	0.9464495	1.3584605	2.3945962	30.6087346
10.0	0.9981258	1.4304456	7.4652567	0.9928273	1.4388922	14.2386060	34.7041064
				$s_2 = 1.50$			
1.0	0.1428571	1.0000000	1.0000141	0.1428571	1.0000001	1.0000140	15.0000003
2.0	0.2857049	1.0000094	1.0008985	0.2856966	1.0000619	1.0008998	16.0002475
3.0	0.4282326	1.0003389	1.0076082	0.4279375	1.0014814	1.0078423	17.0088953
4.0	0.5680782	1.0033504	1.0278995	0.5652483	1.0109337	1.0311565	18.0879478
5.0	0.6980736	1.0162121	1.0644925	0.6857003	1.0416879	1.0814222	19.4255686
6.0	0.8075476	1.0495952	1.1133113	0.7769832	1.1031677	1.1613431	21.3018745
7.0	0.8891124	1.1108876	1.1742520	0.8404167	1.1898860	1.2728306	23.9108002
8.0	0.9432198	1.1996373	1.2706899	0.8883520	1.2864913	1.4638377	27.2404797
9.0	0.9759235	1.3097908	1.5066857	0.9318834	1.3796944	1.9475770	31.1320087
10.0	0.9943278	1.4342436	2.8016334	0.9765765	1.4628361	4.5629432	35.3988954
				$s_2 = 1.55$			
1.0	0.1428571	1.0000000	1.0000141	0.1428571	1.0000002	1.0000140	15.0000003
2.0	0.2857043	1.0000100	1.0008950	0.2856946	1.0000689	1.0008928	16.0002756
3.0	0.4282153	1.0003561	1.0075200	0.4278693	1.0016409	1.0076737	17.0098536
4.0	0.5679302	1.0034983	1.0272070	0.5646380	1.0120264	1.0297959	18.0967898
5.0	0.6974867	1.0167990	1.0615998	0.6832354	1.0454459	1.0749707	19.4647860
6.0	0.8061601	1.0509827	1.1050315	0.7712642	1.1113479	1.1401316	21.4105649
7.0	0.8868307	1.1131693	1.1548562	0.8312286	1.2030385	1.2209344	24.1311110
8.0	0.9402921	1.2025650	1.2265947	0.8763875	1.3040546	1.3521201	27.6044677
9.0	0.9727265	1.3129878	1.3841614	0.9178579	1.4007771	1.6657839	31.6595887
10.0	0.9911701	1.4374013	1.9806025	0.9609220	1.4866674	2.8496312	36.1018007
				$s_2 = 1.60$			
1.0	0.1428571	1.0000000	1.0000140	0.1428571	1.0000002	1.0000140	15.0000004
2.0	0.2857038	1.0000105	1.0008918	0.2856925	1.0000761	1.0008855	16.0003044
3.0	0.4281995	1.0003719	1.0074396	0.4277999	1.0018036	1.0075019	17.0108311
4.0	0.5677966	1.0036320	1.0265860	0.5640207	1.0131341	1.0284214	18.1057626
5.0	0.6969642	1.0173215	1.0590597	0.6807723	1.0492285	1.0685946	19.5044019
6.0	0.8049431	1.0521997	1.0979571	0.7656289	1.1195279	1.1201006	21.5200561
7.0	0.8848564	1.1151436	1.1389034	0.8222735	1.2161404	1.1751591	24.3529825
8.0	0.9377860	1.2050712	1.1925212	0.8647941	1.3215366	1.2621043	27.9716727
9.0	0.9700116	1.3157027	1.3012250	0.9042956	1.4217854	1.4708662	32.1932631
10.0	0.9885034	1.4400680	1.6395753	0.9457880	1.5104562	2.1322169	36.8147801

Let $\rho_1 = \lambda\bar{r}/ms_1$, and $\rho_2 = \lambda\bar{r}/ms_2$. Then, we obtain

$$p_k = \begin{cases} p_0 \frac{(m\rho_1)^k}{k!}, & 1 \le k \le m-1; \\ p_0 \frac{m^m}{m!}\rho_1^k, & m \le k \le b; \\ p_0 \frac{m^m}{m!}\rho_1^b\rho_2^{k-b}, & k \ge b+1; \end{cases}$$

where p_0 is

$$\left(1 + \sum_{k=1}^{m-1} \frac{(m\rho_1)^k}{k!} + \sum_{k=m}^{b} \frac{m^m}{m!}\rho_1^k + \sum_{k=b+1}^{\infty} \frac{m^m}{m!}\rho_1^b\rho_2^{k-b}\right)^{-1},$$

or,

$$\left(1 + \sum_{k=1}^{m-1} \frac{(m\rho_1)^k}{k!} + \frac{m^m}{m!} \cdot \frac{\rho_1^m - \rho_1^{b+1}}{1-\rho_1} + \frac{m^m}{m!}\rho_1^b \cdot \frac{\rho_2}{1-\rho_2}\right)^{-1}.$$

It is clear that two-speed schemes are practically more feasible to implement than arithmetic-speed schemes and geometric-speed schemes, since a two-speed scheme requires much less core speed change than an arithmetic-speed scheme and a geometric-speed scheme.

In Table 5, we demonstrate numerical data for two-speed schemes with the idle-speed model. We assume that there are $m = 7$ cores. The task arrival rate λ is 1, 2, ..., 10. The speed

TABLE 6
Numerical Data for Two-Speed Schemes (Constant-Speed Model)

	Two-Speed Scheme			Single-Speed Scheme			
λ	ρ	$\bar{s}$	T	ρ	s	T	P
			$s_2 = 1.45$				
1.0	0.1428571	1.0000000	1.0000141	0.1428571	1.0000000	1.0000142	21.0000003
2.0	0.2857054	1.0000089	1.0009024	0.2857104	1.0000134	1.0009486	21.0002820
3.0	0.4282515	1.0003199	1.0077055	0.4283636	1.0004852	1.0088968	21.0101944
4.0	0.5682428	1.0031857	1.0286766	0.5686925	1.0048112	1.0388686	21.1015217
5.0	0.6987374	1.0155483	1.0678159	0.6981864	1.0230587	1.1152596	21.4954848
6.0	0.8091442	1.0479986	1.1231273	0.8024964	1.0680956	1.2688819	22.5295961
7.0	0.8917791	1.1082209	1.1983095	0.8750070	1.1428480	1.5320696	24.4487289
8.0	0.9466852	1.1961720	1.3298552	0.9238529	1.2370553	1.9953976	27.2515097
9.0	0.9797433	1.3059710	1.7052809	0.9612454	1.3375505	3.1900575	30.7505318
10.0	0.9981258	1.4304456	7.4652567	0.9950024	1.4357468	20.3066566	34.7172258
			$s_2 = 1.50$				
1.0	0.1428571	1.0000000	1.0000141	0.1428571	1.0000000	1.0000142	21.0000004
2.0	0.2857049	1.0000094	1.0008985	0.2857100	1.0000149	1.0009471	21.0003135
3.0	0.4282326	1.0003389	1.0076082	0.4283417	1.0005363	1.0088427	21.0112673
4.0	0.5680782	1.0033504	1.0278995	0.5684290	1.0052769	1.0382766	21.1114006
5.0	0.6980736	1.0162121	1.0644925	0.6968389	1.0250371	1.1115102	21.5390536
6.0	0.8075476	1.0495952	1.1133113	0.7987850	1.0730583	1.2517438	22.6490410
7.0	0.8891124	1.1108876	1.1742520	0.8684548	1.1514704	1.4730089	24.6870136
8.0	0.9432198	1.1996373	1.2706899	0.9150434	1.2489650	1.8228458	27.6379409
9.0	0.9759235	1.3097908	1.5066857	0.9509490	1.3520328	2.5859391	31.3005444
10.0	0.9943278	1.4342436	2.8016334	0.9837147	1.4522213	6.4356053	35.4386008
			$s_2 = 1.55$				
1.0	0.1428571	1.0000000	1.0000141	0.1428571	1.0000000	1.0000142	21.0000004
2.0	0.2857043	1.0000100	1.0008950	0.2857096	1.0000164	1.0009456	21.0003453
3.0	0.4282153	1.0003561	1.0075200	0.4283198	1.0005876	1.0087884	21.0123466
4.0	0.5679302	1.0034983	1.0272070	0.5681661	1.0057421	1.0376862	21.1212781
5.0	0.6974867	1.0167990	1.0615998	0.6955091	1.0269969	1.1078342	21.5823789
6.0	0.8061601	1.0509827	1.1050315	0.7951729	1.0779327	1.2355967	22.7674441
7.0	0.8868307	1.1131693	1.1548562	0.8621473	1.1598947	1.4212449	24.9232959
8.0	0.9402921	1.2025650	1.2265947	0.9066026	1.2605933	1.6876094	28.0224228
9.0	0.9727265	1.3129878	1.3841614	0.9410810	1.3662100	2.2043221	31.8505030
10.0	0.9911701	1.4374013	1.9806025	0.9728692	1.4684105	3.9788538	36.1636098
			$s_2 = 1.60$				
1.0	0.1428571	1.0000000	1.0000140	0.1428571	1.0000000	1.0000142	21.0000005
2.0	0.2857038	1.0000105	1.0008918	0.2857091	1.0000180	1.0009440	21.0003776
3.0	0.4281995	1.0003719	1.0074396	0.4282976	1.0006393	1.0087335	21.0134348
4.0	0.5677966	1.0036320	1.0265860	0.5679028	1.0062083	1.0370953	21.1311863
5.0	0.6969642	1.0173215	1.0590597	0.6941910	1.0289470	1.1042137	21.6256523
6.0	0.8049431	1.0521997	1.0979571	0.7916369	1.0827475	1.2202721	22.8854542
7.0	0.8848564	1.1151436	1.1389034	0.8560341	1.1681777	1.3752122	25.1589878
8.0	0.9377860	1.2050712	1.1925212	0.8984593	1.2720189	1.5780644	28.4071709
9.0	0.9700116	1.3157027	1.3012250	0.9315643	1.3801670	1.9398974	32.4031821
10.0	0.9885034	1.4400680	1.6395753	0.9623938	1.4843939	2.9500737	36.8952560

schemes are $b = 10$, $s_1 = 1$, and $s_2 = 1.45, 1.50, 1.55, 1.60$. For each combination of λ and s_2, we show the average server utilization ρ, the average core speed $\bar{s}$, the average task response time T, and the average power consumption P. We also find the speed s of a single-speed scheme which consumes power P. For such a single-speed scheme, we also show the average server utilization ρ and the average task response time T. In Table 6, we demonstrate numerical data for two-speed schemes with the constant-speed model. All the parameters are set in the same way as Table 5. Our observations in Tables 5-6 are similar to those in Tables 1-4.

4.4 Optimal Speed Schemes

Notice that for a given application environment, i.e., the number m of cores, the base power supply P^*, the task arrival rate λ, the average task execution requirement $\bar{r}$, and total power supply P, there is an optimal choice of (c, d) such that an arithmetic-speed scheme is optimized in the sense that the average task response time T is minimized, while the power consumption is no more than (actually fixed at) P. In Table 7, we show numerical data for an optimal arithmetic-speed scheme. We assume that there are $m = 7$ cores with base power supply $P^* = 2$ Watts. The task arrival rate λ is 10 per second. The average task execution requirement is $\bar{r} = 1$ giga instructions. The total power supply P is 40, 50, and 60 Watts. The speed schemes are (c, d), where $c = 0.6, 0.7, \ldots, 1.5$ and d is determined in such a way that the power consumption is P. For each combination of c and P, we show d and T for both idle-speed and constant-speed models. It is observed that as c increases, T

TABLE 7
Numerical Data for Optimal Arithmetic-Speed Schemes

	Idle-Speed Model		Constant-Speed Model	
c	d	T	d	T
	$P = 40$			
0.6	0.0929066	0.9537828	0.0891469	0.9860015
0.7	0.0851351	0.9375412	0.0806331	0.9776426
0.8	0.0768130	0.9253396	0.0715767	0.9746921
0.9	0.0680165	0.9171214	0.0620785	0.9775803
1.0	0.0588194	0.9130845	0.0522425	0.9872790
1.1	0.0492976	0.9137676	0.0421840	1.0056240
1.2	0.0395349	0.9202453	0.0320416	1.0360497
1.3	0.0296322	0.9345790	0.0219990	1.0854837
1.4	0.0197263	0.9610067	0.0123325	1.1704824
1.5	0.0100331	1.0099896	0.0035591	1.3491911
	$P = 50$			
0.6	0.1727911	0.6259255	0.1554743	0.6662418
0.7	0.1607321	0.6211085	0.1422760	0.6656074
0.8	0.1483832	0.6168461	0.1288044	0.6660509
0.9	0.1357573	0.6131738	0.1150851	0.6677395
1.0	0.1228689	0.6101408	0.1011490	0.6709037
1.1	0.1097344	0.6078146	0.0870339	0.6758685
1.2	0.0963730	0.6062884	0.0727882	0.6831086
1.3	0.0828073	0.6056922	0.0584753	0.6933505
1.4	0.0690656	0.6062126	0.0441822	0.7077845
1.5	0.0551834	0.6081280	0.0300353	0.7285644
	$P = 60$			
0.6	0.2386776	0.5255107	0.2053281	0.5686600
0.7	0.2247955	0.5217938	0.1900552	0.5679225
0.8	0.2106517	0.5183222	0.1745311	0.5677274
0.9	0.1962509	0.5151105	0.1587679	0.5681496
1.0	0.1815982	0.5121758	0.1427812	0.5692843
1.1	0.1667001	0.5095393	0.1265910	0.5712554
1.2	0.1515645	0.5072268	0.1102226	0.5742278
1.3	0.1362007	0.5052707	0.0937092	0.5784285
1.4	0.1206201	0.5037119	0.0770945	0.5841827
1.5	0.1048365	0.5026035	0.0604376	0.5919807

TABLE 8
Numerical Data for Optimal Geometric-Speed Schemes

	Idle-Speed Model		Constant-Speed Model	
q	a	T	a	T
	$P = 40$			
0.6	1.0774638	1.1331786	1.0766326	1.1445382
0.7	1.0704462	1.0404180	1.0688102	1.0619486
0.8	1.0621832	0.9833436	1.0597090	1.0165381
0.9	1.0534633	0.9479121	1.0502142	0.9944455
1.0	1.0447234	0.9271041	1.0408119	0.9894410
1.1	1.0361986	0.9175987	1.0317702	0.9995887
1.2	1.0280219	0.9182976	1.0232536	1.0262626
1.3	1.0202794	0.9299921	1.0153903	1.0752675
1.4	1.0130458	0.9561711	1.0083270	1.1626422
1.5	1.0064219	1.0068590	1.0023261	1.3464702
	$P = 50$			
0.6	1.1442630	0.6838787	1.1382275	0.7046971
0.7	1.1283144	0.6610225	1.1207464	0.6880649
0.8	1.1134146	0.6440438	1.1044648	0.6777222
0.9	1.0995130	0.6312066	1.0893379	0.6720897
1.0	1.0865250	0.6215146	1.0752821	0.6703787
1.1	1.0743634	0.6143744	1.0622116	0.6722802
1.2	1.0629487	0.6094388	1.0500522	0.6778562
1.3	1.0522129	0.6065321	1.0387460	0.6875563
1.4	1.0420997	0.6056202	1.0282563	0.7023818
1.5	1.0325644	0.6068166	1.0185732	0.7243517
	$P = 60$			
0.6	1.1884241	0.5745136	1.1771506	0.5970545
0.7	1.1689402	0.5583398	1.1556994	0.5859288
0.8	1.1512090	0.5454972	1.1362192	0.5783049
0.9	1.1349416	0.5351049	1.1184017	0.5733879
1.0	1.1199104	0.5266164	1.1020030	0.5707316
1.1	1.1059364	0.5196757	1.0868322	0.5701025
1.2	1.0928779	0.5140445	1.0727396	0.5714189
1.3	1.0806213	0.5095630	1.0596091	0.5747313
1.4	1.0690750	0.5061277	1.0473515	0.5802340
1.5	1.0581650	0.5036800	1.0359019	0.5883169

decreases; however, after certain value of c, further increment of c makes T longer. Hence, there is an optimal selection of (c, d).

In Table 8, we show numerical data for an optimal geometric-speed scheme. All parameters are the same as those in Table 7. The speed schemes are (q, a), where $q = 0.6, 0.7, \ldots, 1.5$ and a is determined in such a way that the power consumption is P. In Table 9, we show numerical data for an optimal two-speed scheme. All parameters are the same as those in Tables 7-8. The speed schemes are (b, s_1, s_2), where $b = 10$, $s_1 = 0.6, 0.7, \ldots, 1.5$ and s_2 is determined in such a way that the power consumption is P. We observe similar phenomenon, i.e., there is an optimal choice of (q, a) such that a geometric-speed scheme is optimized, and there is an optimal choice of (s_1, s_2) such that a two-speed scheme with a given b is optimized.

It is an interesting problem to find an optimal speed scheme. To this end, we need closed-form expressions of T and P. Fortunately, such closed-form expressions of T and P are available for two-speed schemes.

5 Two-Speed Scheme Optimization

In this section, we consider the following two-speed scheme optimization problem, namely, given m, P^*, λ, $\bar{r}$, $\tilde{P}$, and b, find (s_1, s_2) such that T is minimized and that the total power consumption is $\tilde{P}$.

First of all, we need closed-form expressions of T and P. Based on the p_k's, we get

$$\bar{N} = \sum_{k=1}^{\infty} k p_k = p_0 \left(\sum_{k=1}^{m-1} k \frac{(m\rho_1)^k}{k!} + \sum_{k=m}^{b} k \frac{m^m}{m!} \rho_1^k + \sum_{k=b+1}^{\infty} k \frac{m^m}{m!} \rho_1^b \rho_2^{k-b} \right) = p_0 \left(\sum_{k=1}^{m-1} \frac{(m\rho_1)^k}{(k-1)!} + \frac{m^m}{m!} \cdot \frac{m\rho_1^m - (m-1)\rho_1^{m+1} - (b+1)\rho_1^{b+1} + b\rho_1^{b+2}}{(1-\rho_1)^2} + \frac{m^m}{m!} \rho_1^b \cdot \frac{(b+1)\rho_2 - b\rho_2^2}{(1-\rho_2)^2} \right),$$

and

$$T = \frac{p_0}{\lambda} \left(\sum_{k=1}^{m-1} \frac{(m\rho_1)^k}{(k-1)!} + \frac{m^m}{m!} \cdot \frac{m\rho_1^m - (m-1)\rho_1^{m+1} - (b+1)\rho_1^{b+1} + b\rho_1^{b+2}}{(1-\rho_1)^2} + \frac{m^m}{m!} \rho_1^b \cdot \frac{(b+1)\rho_2 - b\rho_2^2}{(1-\rho_2)^2} \right).$$

TABLE 9
Numerical Data for Optimal Two-Speed Schemes

	Idle-Speed Model		Constant-Speed Model	
s_1	s_2	T	s_2	T
		$P = 40$		
0.6	1.6870432	1.4813822	1.6869560	1.4815686
0.7	1.7110938	1.4126843	1.7107618	1.4132782
0.8	1.7403700	1.3389759	1.7392710	1.3405895
0.9	1.7749908	1.2617630	1.7717452	1.2656227
1.0	1.8135613	1.1835489	1.8049032	1.1918941
1.1	1.8517645	1.1078151	1.8308139	1.1246307
1.2	1.8807003	1.0389438	1.8348324	1.0719454
1.3	1.8855982	0.9828273	1.7964653	1.0500572
1.4	1.8467847	0.9502880	1.7039769	1.1021530
1.5	1.7509926	0.9697020	1.5918002	1.3382961
		$P = 50$		
0.8	2.2082834	1.0660192	2.2067640	1.0663726
0.9	2.2929497	1.0187349	2.2887351	1.0195235
1.0	2.3927614	0.9682241	2.3821614	0.9697877
1.1	2.5064609	0.9157517	2.4819466	0.9185616
1.2	2.6296306	0.8628638	2.5768521	0.8675510
1.3	2.7533918	0.8111208	2.6463129	0.8186132
1.4	2.8627466	0.7618773	2.6555601	0.7738928
1.5	2.9339840	0.7162187	2.5481962	0.7370723
1.6	2.9300745	0.6751416	2.2480173	0.7205925
1.7	2.7925009	0.6403154	1.8122557	0.7892641
		$P = 60$		
1.0	2.8768794	0.9215253	2.8666372	0.9221828
1.1	3.0501656	0.8756876	3.0271291	0.8768202
1.2	3.2476745	0.8287305	3.1993502	0.8305161
1.3	3.4654711	0.7819520	3.3698072	0.7845822
1.4	3.6956785	0.7365072	3.5148950	0.7402207
1.5	3.9252099	0.6932815	3.5952414	0.6984757
1.6	4.1337777	0.6528583	3.5447125	0.6604214
1.7	4.2904808	0.6155598	3.2472866	0.6281926
1.8	4.3474922	0.5815377	2.5181674	0.6134728
1.9	4.2270632	0.5509313	1.9000000	0.6446852

5.1 The Idle-Speed Model

The average power consumption by the m cores is

$$\begin{aligned}
P &= \sum_{k=1}^{m-1} p_k k s_1^\alpha + \sum_{k=m}^{b} p_k m s_1^\alpha + \sum_{k=b+1}^{\infty} p_k m s_2^\alpha + mP^* \\
&= \left(\sum_{k=1}^{m-1} k p_k + m \sum_{k=m}^{b} p_k\right) s_1^\alpha + m\left(\sum_{k=b+1}^{\infty} p_k\right) s_2^\alpha + mP^* \\
&= p_0\left(\left(\sum_{k=1}^{m-1} k\frac{(m\rho_1)^k}{k!} + m\sum_{k=m}^{b}\frac{m^m}{m!}\rho_1^k\right)s_1^\alpha \right. \\
&\quad \left. + m\left(\sum_{k=b+1}^{\infty}\frac{m^m}{m!}\rho_1^b\rho_2^{k-b}\right)s_2^\alpha\right) + mP^* \\
&= p_0\left(\left(\sum_{k=1}^{m-1}\frac{(m\rho_1)^k}{(k-1)!} + \frac{m^{m+1}}{m!}\cdot\frac{\rho_1^m-\rho_1^{b+1}}{1-\rho_1}\right)s_1^\alpha \right. \\
&\quad \left. + \frac{m^{m+1}}{m!}\rho_1^b\frac{\rho_2}{1-\rho_2}s_2^\alpha\right) + mP^*,
\end{aligned}$$

for the idle-speed model. It can be verified that if the speed of the m cores is a constant speed s (i.e., $s_1 = s_2 = s$ and $\rho_1 = \rho_2 = \rho$), the average power consumption is

$$\begin{aligned}
P &= p_0\left(\sum_{k=1}^{m-1}\frac{(m\rho)^k}{(k-1)!} + \frac{m^{m+1}}{m!}\cdot\frac{\rho^m}{1-\rho}\right)s^\alpha + mP^* \\
&= \left(m\rho\sum_{k=1}^{m-1} p_0\frac{(m\rho)^{k-1}}{(k-1)!} + mp_0\frac{(m\rho)^m}{m!}\cdot\frac{1}{1-\rho}\right)s^\alpha + mP^* \\
&= \left(m\rho\sum_{k=0}^{m-2} p_k + m\cdot\frac{p_m}{1-\rho}\right)s^\alpha + mP^* \\
&= m\rho\left(\sum_{k=0}^{m-2} p_k + \frac{p_m}{\rho(1-\rho)}\right)s^\alpha + mP^* \\
&= m\rho\left(\sum_{k=0}^{m-1} p_k + \frac{p_m}{\rho(1-\rho)} - p_{m-1}\right)s^\alpha + mP^* \\
&= m\rho\left(\sum_{k=0}^{m-1} p_k + \frac{p_m}{\rho(1-\rho)} - \frac{p_m}{\rho}\right)s^\alpha + mP^* \\
&= m\rho\left(\sum_{k=0}^{m-1} p_k + \frac{p_m}{1-\rho}\right)s^\alpha + mP^* \\
&= m\rho\left(\sum_{k=0}^{m-1} p_k + \sum_{k=m}^{\infty} p_k\right)s^\alpha + mP^* \\
&= m\rho s^\alpha + mP^*,
\end{aligned}$$

for the idle-speed model.

Our optimization problem is to find s_1 and s_2 such that $T(s_1, s_2)$ is minimized subject to the constraint that $P(s_1, s_2) = \tilde{P}$, where

$$\begin{aligned}
P(s_1, s_2) = p_0\left(\left(\sum_{k=1}^{m-1}\frac{(m\rho_1)^k}{(k-1)!} + \frac{m^{m+1}}{m!}\cdot\frac{\rho_1^m-\rho_1^{b+1}}{1-\rho_1}\right)s_1^\alpha \right. \\
\left. + \frac{m^{m+1}}{m!}\rho_1^b\frac{\rho_2}{1-\rho_2}s_2^\alpha\right) + mP^*.
\end{aligned}$$

We can minimize $T(s_1, s_2)$ by using the method of Lagrange multiplier, namely, $\nabla T(s_1, s_2) = \phi\nabla P(s_1, s_2)$, that is, $\partial T/\partial s_i = \phi\partial P/\partial s_i$, for $i = 1, 2$, where ϕ is a Lagrange multiplier.

Let us rewrite T as $T = (1/\lambda)p_0 D$, where

$$\begin{aligned}
D = &\sum_{k=1}^{m-1}\frac{(m\rho_1)^k}{(k-1)!} + \frac{m^m}{m!} \\
&\cdot\frac{m\rho_1^m - (m-1)\rho_1^{m+1} - (b+1)\rho_1^{b+1} + b\rho_1^{b+2}}{(1-\rho_1)^2} \\
&+ \frac{m^m}{m!}\rho_1^b\cdot\frac{(b+1)\rho_2 - b\rho_2^2}{(1-\rho_2)^2}.
\end{aligned}$$

It is clear that

$$\frac{\partial\rho_i}{\partial s_i} = -\frac{\lambda\bar{r}}{ms_i^2} = -\frac{\rho_i}{s_i},$$

for $i = 1, 2$. Notice that

$$\begin{aligned}
\frac{\partial T}{\partial s_i} &= \frac{\partial T}{\partial\rho_i}\cdot\frac{\partial\rho_i}{\partial s_i} = \frac{1}{\lambda}\left(\frac{\partial p_0}{\partial\rho_i}D + p_0\frac{\partial D}{\partial\rho_i}\right)\left(-\frac{\rho_i}{s_i}\right) \\
&= -\frac{\rho_i}{\lambda s_i}\left(\frac{\partial p_0}{\partial\rho_i}D + p_0\frac{\partial D}{\partial\rho_i}\right),
\end{aligned}$$

for $i = 1, 2$. Also, we have

$$\frac{\partial p_0}{\partial \rho_1} = -p_0^2 \Bigg(m \sum_{k=1}^{m-1} \frac{(m\rho_1)^{k-1}}{(k-1)!} + \frac{m^m}{m!} \cdot \frac{m\rho_1^{m-1} - (m-1)\rho_1^m - (b+1)\rho_1^b + b\rho_1^{b+1}}{(1-\rho_1)^2} + \frac{m^m}{m!} b\rho_1^{b-1} \cdot \frac{\rho_2}{1-\rho_2} \Bigg),$$

and

$$\frac{\partial p_0}{\partial \rho_2} = -p_0^2 \frac{m^m}{m!} \rho_1^b \cdot \frac{1}{(1-\rho_2)^2}.$$

Furthermore, we have

$$\begin{aligned}\frac{\partial D}{\partial \rho_1} &= m \sum_{k=1}^{m-1} \frac{k(m\rho_1)^{k-1}}{(k-1)!} + \frac{m^m}{m!} \\ &\quad \cdot \frac{1}{(1-\rho_1)^3} \left(m^2 \rho_1^{m-1} - (2m^2 - 2m - 1)\rho_1^m \right. \\ &\quad + (m-1)^2 \rho_1^{m+1} - (b+1)^2 \rho_1^b + (2b^2 + 2b - 1) \\ &\quad \left. \cdot \rho_1^{b+1} - b^2 \rho_1^{b+2} \right) + \frac{m^m}{m!} b\rho_1^{b-1} \cdot \frac{(b+1)\rho_2 - b\rho_2^2}{(1-\rho_2)^2},\end{aligned}$$

and

$$\frac{\partial D}{\partial \rho_2} = \frac{m^m}{m!} \rho_1^b \cdot \frac{(b+1) - (b-1)\rho_2}{(1-\rho_2)^3}.$$

Let us rewrite P as

$$P = p_0 G + mP^*,$$

where

$$\begin{aligned}G &= \left(\sum_{k=1}^{m-1} \frac{(m\rho_1)^k}{(k-1)!} + \frac{m^{m+1}}{m!} \cdot \frac{\rho_1^m - \rho_1^{b+1}}{1-\rho_1} \right) s_1^\alpha \\ &\quad + \frac{m^{m+1}}{m!} \rho_1^b \frac{\rho_2}{1-\rho_2} s_2^\alpha.\end{aligned}$$

Notice that

$$\frac{\partial P}{\partial s_i} = \frac{\partial p_0}{\partial s_i} G + p_0 \frac{\partial G}{\partial s_i},$$

for $i = 1, 2$. Also, we have

$$\frac{\partial p_0}{\partial s_i} = \frac{\partial p_0}{\partial \rho_i} \cdot \frac{\partial \rho_i}{\partial s_i} = -\frac{\rho_i}{s_i} \cdot \frac{\partial p_0}{\partial \rho_i},$$

for $i = 1, 2$. Furthermore, we have

$$\begin{aligned}\frac{\partial G}{\partial s_1} &= \left(m \sum_{k=1}^{m-1} \frac{k(m\rho_1)^{k-1}}{(k-1)!} + \frac{m^{m+1}}{m!} \right. \\ &\quad \left. \cdot \frac{m\rho_1^{m-1} - (m-1)\rho_1^m - (b+1)\rho_1^b + b\rho_1^{b+1}}{(1-\rho_1)^2} \right) \left(-\frac{\rho_1}{s_1} \right) s_1^\alpha \\ &\quad + \left(\sum_{k=1}^{m-1} \frac{(m\rho_1)^k}{(k-1)!} + \frac{m^{m+1}}{m!} \cdot \frac{\rho_1^m - \rho_1^{b+1}}{1-\rho_1} \right) \alpha s_1^{\alpha-1} \\ &\quad + \frac{m^{m+1}}{m!} b\rho_1^{b-1} \left(-\frac{\rho_1}{s_1} \right) \frac{\rho_2}{1-\rho_2} s_2^\alpha \\ &= -\left(m \sum_{k=1}^{m-1} \frac{k(m\rho_1)^{k-1}}{(k-1)!} + \frac{m^{m+1}}{m!} \right. \\ &\quad \left. \cdot \frac{m\rho_1^{m-1} - (m-1)\rho_1^m - (b+1)\rho_1^b + b\rho_1^{b+1}}{(1-\rho_1)^2} \right) \rho_1 s_1^{\alpha-1} \\ &\quad + \left(\sum_{k=1}^{m-1} \frac{(m\rho_1)^k}{(k-1)!} + \frac{m^{m+1}}{m!} \cdot \frac{\rho_1^m - \rho_1^{b+1}}{1-\rho_1} \right) \alpha s_1^{\alpha-1} \\ &\quad - \frac{m^{m+1}}{m!} b\rho_1^b \frac{1}{s_1} \cdot \frac{\rho_2}{1-\rho_2} s_2^\alpha,\end{aligned}$$

and

$$\begin{aligned}\frac{\partial G}{\partial s_2} &= \frac{m^{m+1}}{m!} \rho_1^b \left(\frac{1}{(1-\rho_2)^2} \left(-\frac{\rho_2}{s_2} \right) s_2^\alpha + \frac{\rho_2}{1-\rho_2} \alpha s_2^{\alpha-1} \right) \\ &= \frac{m^{m+1}}{m!} \rho_1^b \left(-\frac{\rho_2}{(1-\rho_2)^2} s_2^{\alpha-1} + \frac{\rho_2}{1-\rho_2} \alpha s_2^{\alpha-1} \right).\end{aligned}$$

Now, let us go back to the equation

$$\frac{\partial T}{\partial s_i} = -\frac{\rho_i}{\lambda s_i} \left(\frac{\partial p_0}{\partial \rho_i} D + p_0 \frac{\partial D}{\partial \rho_i} \right) = \phi \frac{\partial P}{\partial s_i},$$

and we obtain

$$s_i = -\frac{\rho_i}{\phi\lambda} \left(\frac{\partial p_0}{\partial \rho_i} D + p_0 \frac{\partial D}{\partial \rho_i} \right) \Bigg/ \frac{\partial P}{\partial s_i} = \frac{M_i}{\phi}, \tag{1}$$

where

$$M_i = -\frac{\rho_i}{\lambda} \left(\frac{\partial p_0}{\partial \rho_i} D + p_0 \frac{\partial D}{\partial \rho_i} \right) \Bigg/ \frac{\partial P}{\partial s_i},$$

for $i = 1, 2$. By the constraint $P(s_1, s_2) = \tilde{P}$, that is,

$$p_0 \left(\left(\sum_{k=1}^{m-1} \frac{(m\rho_1)^k}{(k-1)!} + \frac{m^{m+1}}{m!} \cdot \frac{\rho_1^m - \rho_1^{b+1}}{1-\rho_1} \right) s_1^\alpha + \frac{m^{m+1}}{m!} \rho_1^b \frac{\rho_2}{1-\rho_2} s_2^\alpha \right) + mP^* = \tilde{P},$$

or,

$$p_0 \left(\left(\sum_{k=1}^{m-1} \frac{(m\rho_1)^k}{(k-1)!} + \frac{m^{m+1}}{m!} \cdot \frac{\rho_1^m - \rho_1^{b+1}}{1-\rho_1} \right) \left(\frac{M_1}{\phi} \right)^\alpha + \frac{m^{m+1}}{m!} \rho_1^b \frac{\rho_2}{1-\rho_2} \left(\frac{M_2}{\phi} \right)^\alpha \right) = \tilde{P} - mP^*,$$

we get

$$\phi = \left(\frac{p_0}{\tilde{P} - mP^*}\left(\left(\sum_{k=1}^{m-1}\frac{(m\rho_1)^k}{(k-1)!} + \frac{m^{m+1}}{m!}\cdot\frac{\rho_1^m - \rho_1^{b+1}}{1-\rho_1}\right)M_1^\alpha + \frac{m^{m+1}}{m!}\rho_1^b\frac{\rho_2}{1-\rho_2}M_2^\alpha\right)\right)^{1/\alpha}. \tag{2}$$

5.2 The Constant-Speed Model

The average power consumption by the m cores is

$$\begin{aligned}
P &= \sum_{k=0}^{b} p_k m(s_1^\alpha + P^*) + \sum_{k=b+1}^{\infty} p_k m(s_2^\alpha + P^*)\\
&= m\left(\sum_{k=0}^{b} p_k\right)s_1^\alpha + m\left(\sum_{k=b+1}^{\infty} p_k\right)s_2^\alpha + mP^*\\
&= mp_0\left(\left(\sum_{k=0}^{m-1}\frac{(m\rho_1)^k}{k!} + \sum_{k=m}^{b}\frac{m^m}{m!}\rho_1^k\right)s_1^\alpha + \left(\sum_{k=b+1}^{\infty}\frac{m^m}{m!}\rho_1^b\rho_2^{k-b}\right)s_2^\alpha\right) + mP^*\\
&= mp_0\left(\left(\sum_{k=0}^{m-1}\frac{(m\rho_1)^k}{k!} + \frac{m^m}{m!}\cdot\frac{\rho_1^m - \rho_1^{b+1}}{1-\rho_1}\right)s_1^\alpha + \frac{m^m}{m!}\rho_1^b\frac{\rho_2}{1-\rho_2}s_2^\alpha\right) + mP^*,
\end{aligned}$$

for the constant-speed model. It can be verified that if the speed of the m cores is a constant speed s (i.e., $s_1 = s_2 = s$ and $\rho_1 = \rho_2 = \rho$), the average power consumption is $P = m(s^\alpha + P^*)$ for the constant-speed model.

Our optimization problem for the constant-speed model is defined and solved in a similar way to the idle-speed model, with the following modifications. The function $P(s_1, s_2)$ becomes

$$P(s_1, s_2) = mp_0\left(\left(\sum_{k=0}^{m-1}\frac{(m\rho_1)^k}{k!} + \frac{m^m}{m!}\cdot\frac{\rho_1^m - \rho_1^{b+1}}{1-\rho_1}\right)s_1^\alpha + \frac{m^m}{m!}\rho_1^b\frac{\rho_2}{1-\rho_2}s_2^\alpha\right) + mP^*,$$

which can be rewritten as

$$P = mp_0 G + mP^*,$$

where

$$G = \left(\sum_{k=0}^{m-1}\frac{(m\rho_1)^k}{k!} + \frac{m^m}{m!}\cdot\frac{\rho_1^m - \rho_1^{b+1}}{1-\rho_1}\right)s_1^\alpha + \frac{m^m}{m!}\rho_1^b\frac{\rho_2}{1-\rho_2}s_2^\alpha.$$

Notice that

$$\frac{\partial P}{\partial s_i} = m\left(\frac{\partial p_0}{\partial s_i}G + p_0\frac{\partial G}{\partial s_i}\right),$$

for $i = 1, 2$. Furthermore, we have

$$\begin{aligned}
\frac{\partial G}{\partial s_1} &= \left(m\sum_{k=1}^{m-1}\frac{(m\rho_1)^{k-1}}{(k-1)!} + \frac{m^m}{m!}\cdot\frac{m\rho_1^{m-1} - (m-1)\rho_1^m - (b+1)\rho_1^b + b\rho_1^{b+1}}{(1-\rho_1)^2}\right)\left(-\frac{\rho_1}{s_1}\right)s_1^\alpha\\
&\quad + \left(\sum_{k=0}^{m-1}\frac{(m\rho_1)^k}{k!} + \frac{m^m}{m!}\cdot\frac{\rho_1^m - \rho_1^{b+1}}{1-\rho_1}\right)\alpha s_1^{\alpha-1}\\
&\quad + \frac{m^m}{m!}b\rho_1^{b-1}\left(-\frac{\rho_1}{s_1}\right)\frac{\rho_2}{1-\rho_2}s_2^\alpha\\
&= -\left(m\sum_{k=1}^{m-1}\frac{(m\rho_1)^{k-1}}{(k-1)!} + \frac{m^m}{m!}\cdot\frac{m\rho_1^{m-1} - (m-1)\rho_1^m - (b+1)\rho_1^b + b\rho_1^{b+1}}{(1-\rho_1)^2}\right)\rho_1 s_1^{\alpha-1}\\
&\quad + \left(\sum_{k=0}^{m-1}\frac{(m\rho_1)^k}{k!} + \frac{m^m}{m!}\cdot\frac{\rho_1^m - \rho_1^{b+1}}{1-\rho_1}\right)\alpha s_1^{\alpha-1}\\
&\quad - \frac{m^m}{m!}b\rho_1^b\frac{1}{s_1}\cdot\frac{\rho_2}{1-\rho_2}s_2^\alpha,
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial G}{\partial s_2} &= \frac{m^m}{m!}\rho_1^b\left(\frac{1}{(1-\rho_2)^2}\left(-\frac{\rho_2}{s_2}\right)s_2^\alpha + \frac{\rho_2}{1-\rho_2}\alpha s_2^{\alpha-1}\right)\\
&= \frac{m^m}{m!}\rho_1^b\left(-\frac{\rho_2}{(1-\rho_2)^2}s_2^{\alpha-1} + \frac{\rho_2}{1-\rho_2}\alpha s_2^{\alpha-1}\right).
\end{aligned}$$

By the constraint $P(s_1, s_2) = \tilde{P}$, that is,

$$mp_0\left(\left(\sum_{k=0}^{m-1}\frac{(m\rho_1)^k}{k!} + \frac{m^m}{m!}\cdot\frac{\rho_1^m - \rho_1^{b+1}}{1-\rho_1}\right)s_1^\alpha + \frac{m^m}{m!}\rho_1^b\frac{\rho_2}{1-\rho_2}s_2^\alpha\right) + mP^* = \tilde{P},$$

or,

$$mp_0\left(\left(\sum_{k=0}^{m-1}\frac{(m\rho_1)^k}{k!} + \frac{m^m}{m!}\cdot\frac{\rho_1^m - \rho_1^{b+1}}{1-\rho_1}\right)\left(\frac{M_1}{\phi}\right)^\alpha + \frac{m^m}{m!}\rho_1^b\frac{\rho_2}{1-\rho_2}\left(\frac{M_2}{\phi}\right)^\alpha\right) = \tilde{P} - mP^*,$$

we get

$$\phi = \left(\frac{mp_0}{\tilde{P} - mP^*}\left(\left(\sum_{k=0}^{m-1}\frac{(m\rho_1)^k}{k!} + \frac{m^m}{m!}\cdot\frac{\rho_1^m - \rho_1^{b+1}}{1-\rho_1}\right)M_1^\alpha + \frac{m^m}{m!}\rho_1^b\frac{\rho_2}{1-\rho_2}M_2^\alpha\right)\right)^{1/\alpha}. \tag{3}$$

5.3 A Numerical Method

It is unlikely that there exists an analytical solution of (s_1, s_2). However, a solution can be found numerically. We observe that for the idle-speed model, we have a nonlinear system of three equations specified in (1) and (2). Similarly, for the constant-speed model, we have a nonlinear system of three equations specified in (1) and (3). Since the left-

TABLE 10
Numerical Data for Two-Speed Scheme Optimization

	Idle-Speed Model			Constant-Speed Model		
$\tilde{P}$	s_1	s_2	T	s_1	s_2	T
40	1.4259389	1.8273691	0.9484016	1.2941887	1.8001971	1.0499463
50	1.8193931	2.3349393	0.6170329	1.5911250	2.2837112	0.7203401
60	2.1047420	2.7088408	0.5071361	1.7873109	2.6383719	0.6128423

hand side of each equation is a single variable, the above nonlinear systems of equations can be solved by using the traditional iterative method. However, there are two major difficulties in doing so. First, it is very hard to find an appropriate initial value of (s_1, s_2) for the iterative method to converge. For instance, for the constant-speed model, when $m = 7$, $P^* = 2$, $\lambda = 10$, $\bar{r} = 1$, $\tilde{P} = 40$, and $b = 10$, the iterative algorithm does not converge even if we set $s_1 = 1.3$ and $s_2 = 1.8$, which is very close to the actual solution. Second, even though the iterative algorithm converges, it may converge to a global optimal solution or another local optimal solution which is not the solution we are looking for. Again, for the idle-speed model, when $m = 7$, $P^* = 2$, $\lambda = 10$, $\bar{r} = 1$, $\tilde{P} = 40$, and $b = 10$, if we set $s_1 = 1.425$ and $s_2 = 1.828$, which is very close to the actual solution, the iterative algorithm converges to $s_1 = 1.5314194$ and $s_2 = 0.6930019$, which results in $T = 0.5783720$. This solution does not make sense and has no physical meaning, since we need $s_2 > \lambda\bar{r}/m = 1.4285714$, so that $\rho_2 < 1$. In other words, we are searching for a local optimal solution in certain region. To this end, we need a special technique which we describe below.

To find s_1, s_2, and ϕ which satisfy (as required by (1)) $\phi = M_1/s_1 = M_2/s_2$, or, $\phi = (\partial T/\partial s_1)/(\partial P/\partial s_1) = (\partial T/\partial s_2)/(\partial P/\partial s_2)$, we search (s_1, s_2) along the path $P(s_1, s_2) = \tilde{P}$, i.e., the power consumption constraint. First, we notice that given m, P^*, λ, $\bar{r}$, $\tilde{P}$, b, and s_1, we can find s_2 such that $P(s_1, s_2) = \tilde{P}$ by using the classic bisection method. This is based on the fact that $P(s_1, s_2)$ is an increasing function of s_2. We can search s_2 in an interval $[s_1, s_1 + \delta]$ with δ reasonably chosen. Next, assuming that (s_1^*, s_2^*) is the solution we are seeking, our numerical computations reveal the fact that $\Delta(s_1) = M_1/s_1 - M_2/s_2$ is an increasing function of s_1 along the path $P(s_1, s_2) = \tilde{P}$ when s_1 is close to s_1^*. (Notice that along the path $P(s_1, s_2) = \tilde{P}$, s_2 is a function of s_1 and no longer a free variable.) Hence, we again use the bisection method to find s_1^* (and s_2^* as well by the constraint $P(s_1, s_2) = \tilde{P}$), by choosing a search interval $[lb, ub]$ appropriately, such that $\Delta(lb) < 0$ and $\Delta(ub) > 0$. What we are looking for is s_1^* such that $\Delta(s_1^*) = 0$.

In Table 10, we display our optimal two-speed schemes with $m = 7$, $P^* = 2$, $\lambda = 10$, $\bar{r} = 1$, and $b = 10$, for both idle-speed and constant-speed models. For $\tilde{P} = 40, 50, 60$, the search interval $[lb, ub]$ is [1.400,1.450], [1.800,1.850], [2.075,2.125], respectively under the idle-speed model, and [1.275,1.325], [1.575,1.625], [1.750,1.800], respectively under the constant-speed model. The minimized average task response time T is also given. The values of lb and ub are obtained based on the calculations performed in Table 9. For instance, for $\tilde{P} = 40$ and the idle-speed model, we find that T is minimized when s_1 is close to 1.425. Therefore, we can set the search interval $[lb, ub]$ as [1.400,1.450]. While for the constant-speed model, we find that T is minimized when s_1 is close to 1.300. Therefore, we can set the search interval $[lb, ub]$ as [1.275,1.325].

6 Duality of Performance and Power

Our discussion so far has been focusing on performance improvement (i.e., average task response time reduction) subject to power consumption constraints. We would like to mention that our investigation can also be focused on power consumption reduction subject to performance constraints. We can show that to achieve the same performance goal (i.e., average task response time), a multicore server processor with workload dependent dynamic power management consumes less energy than a multicore server processor without workload dependent dynamic power management. We can also show that given certain application environment and performance goal, there is an optimal speed scheme that minimizes the average power consumption. We can also find optimal two-speed schemes for different performance constraints and power consumption models. Furthermore, the same numerical procedures can be employed for this purpose. Let us consider two optimization problems. The first problem is to minimize $T(s_1, s_2)$ subject to the constraint $P(s_1, s_2) = \tilde{P}$. The second problem is to minimize $P(s_1, s_2)$ subject to the constraint $T(s_1, s_2) = \tilde{T}$. The first problem has been solved in Section 5. To solve the second problem, we need to consider $\partial P/\partial s_i = \phi\partial T/\partial s_i$, for $i = 1, 2$. Hence, our method for finding s_1, s_2, and ϕ is identical to that in Section 5.3. Further details and examples of power consumption reduction subject to performance constraints are elaborated in Section 3 of the supplementary material, available online.

7 Conclusions

We have investigated system performance improvement and energy consumption reduction for multicore server processors by using the technique of workload dependent dynamic power management. Our investigation is based on a queueing model of multicore server processors with variable power and speed according to the current workload. Such analytical study of workload dependent dynamic power management has not been conducted before. We proposed several speed schemes and demonstrated that for the same average power consumption, it is possible to design a multicore server processor with workload dependent

dynamic power management, such that its average task response time is shorter than a multicore server processor of constant speed (i.e., without workload dependent dynamic power management). We showed that given certain application environment and average power consumption, there is an optimal speed scheme that minimizes the average task response time. For two-speed schemes, we formulated and solved the problem of optimal design of a two-speed scheme for given power supply and power consumption model. We pointed out that power consumption reduction subject to performance constraints can be studied in a similar way as performance improvement (i.e., average task response time reduction) subject to power consumption constraints.

Acknowledgments

The author would like to express his gratitude to the three anonymous reviewers for their suggestions to improve the manuscript.

References

[1] [Online]. Available: http://en.wikipedia.org/wiki/Advanced_Configuration_and _Power_Interface, 2014.
[2] [Online]. Available: http://en.wikipedia.org/wiki/Dynamic_voltage_scaling, 2014.
[3] [Online]. Available: http://en.wikipedia.org/wiki/Green_computing, 2014.
[4] [Online]. Available: http://en.wikipedia.org/wiki/multicore_processor, 2014.
[5] [Online]. Available: http://www.intel.com/technology/platform-technology/hyper-threading/index.htm, 2014.
[6] [Online]. Available: http://www.intel.com/technology/turboboost/index.htm, 2014.
[7] [Online]. Available: http://www.nrdc.org/energy/data-center-efficiency-assessment.asp, 2014.
[8] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, A taxonomy and survey of energy-efficient data centers and cloud computing systems. [Online]. Available: http://arxiv.org/abs/1007.0066, 2014.
[9] W. L. Bircher and L. K. John, "Analysis of dynamic power management on multicore processors," in *Proc. 22nd ACM Int. Conf. Supercomput.*, 2008, pp. 327–338.
[10] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE J. Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, Apr. 1992.
[11] W. Chedid and C. Yu, Survey on power management techniques for energy efficient computer systems," unpublished manuscript. [Online]. Available: http://academic.csuohio.edu/yuc/mcrl/survey-power.pdf, 2014.
[12] J. Doyle, R. Shorten, and D. O'Mahony, "Stratus: Load balancing the cloud for carbon emissions control," *IEEE Trans. Cloud Comput.*, vol. 1, no. 1, pp. 116–128, Jan.-Jun. 2013.
[13] R. Elnaggar, Towards energy efficient workload placement in data centers. unpublished manuscript. [Online]. Available: http://web.cecs.pdx.edu/~walpole/class/cs533/papers /raniarpe.pdf, 2014.
[14] Emerson Network Power, Energy logic: Reducing data center energy consumption by creating savings that cascade across systems. [Online]. Available: http://www.emersonnetworkpower.com/documentation/en-us/latest-thinking/edc/documents/white, 2014.
[15] H. T. Fung. (2007, 18 Sep.). Dynamic power and workload management for multi-server system, US Patent 7 272 735. [Online]. Available: http://www.patentstorm.us/patents/7272735.html
[16] Intel, "Intel turbo boost technology in Intel core microarchitecture (Nehalem) based processors," White Paper, http://files.shareholder.com/downloads/INTC/0x0x348508/C9259E98-BE06-42C8-A433-E28F64CB8EF2/TurboBoostWhitePaper.pdf, 2015.
[17] L. Kleinrock, *Queueing Systems, Volume 1: Theory*. New York, NY, USA: Wiley, 1975.
[18] P. Kukkala, T. Arpinen, M. Setälä, M. Hännikäinen, and T. D. Hämäläinen, "Dynamic power management for UML modeled applications on multiprocessor SoC," in *Proc. SPIE 6507, Multimedia on Mobile Devices*, Feb. 2007, doi: 10.1117/12.707328.
[19] R. Kumar, K. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen, "Processor power reduction via single-ISA heterogeneous multi-core architectures," *IEEE Comput. Archit. Letters*, vol. 2, no. 1, p. 2, Jan.-Dec. 2003.
[20] S. J. Lee, H.-K. Lee, and P.-C. Yew, "Runtime performance projection model for dynamic power management," in *Proc. 12th Asia-Pacific Comput. Syst. Archit. Conf.*, 2007, pp. 186–197.
[21] M. Polverini, A. Cianfrani, S. Ren, and A. V. Vasilakos, "Thermal-aware scheduling of batch jobs in geographically distributed data centers," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 71–84, Jan.-Mar. 2014.
[22] J. Shuja, K. Bilal, S. A. Madani, and S. U. Khan, "Data center energy efficient resource scheduling," *Cluster Comput.*, vol. 17, pp. 1265–1277, 2014.
[23] D. C. Snowdon, E. Le Sueur, S. M. Petters, and G. Heiser, "Koala a platform for OS-level power management," in *Proc. 4th ACM Eur. Conf. Comput. Syst.*, 2009, pp. 289–302.
[24] A. C. Sodan, J. Machina, A. Deshmeh, K. Macnaughton, and B. Esbaugh, "Parallelism via multithreaded and multicore CPUs," *IEEE Comput.*, vol. 43, no. 3, pp. 24–32, Mar. 2010.
[25] G. L. Valentini, W. Lassonde, S. U. Khan, N. Min-Allah, S. A. Madani, J. Li, L. Zhang, L. Wang, N. Ghani, J. Kolodziej, H. Li, A. Y. Zomaya, C.-Z. Xu, P. Balaji, A. Vishnu, F. Pinel, J. E. Pecero, D. Kliazovich, and P. Bouvry, "An overview of energy efficiency techniques in cluster computing systems," *Cluster Comput.*, vol. 16, no. 1, pp. 3–15, 2013.
[26] L. Wang and S. U. Khan, "Review of performance metrics for green data centers: a taxonomy study," *J. Supercomput.*, vol. 63, no. 3, pp. 639–656, 2013.
[27] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and practical limits of dynamic voltage scaling," in *Proc. 41st Des. Autom. Conf.*, 2004, pp. 868–873.
[28] X. Zhu, L. T. Yang, H. Chen, J. Wang, S. Yin, and X. Liu, "Real-time tasks oriented energy-aware scheduling in virtualized clouds," *IEEE Trans. Cloud Comput.*, vol. 2, no. 2, pp. 168–180, Apr.-Jun. 2014.

Keqin Li is a SUNY distinguished professor of computer science. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU-GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, and peer-to-peer file sharing systems. He has published more than 350 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is a fellow of the IEEE.