Optimal Power Allocation and Load Distribution for Multiple Heterogeneous Multicore Server Processors across Clouds and Data Centers

Junwei Cao, *Senior Member, IEEE* Keqin Li, *Senior Member, IEEE* Ivan Stojmenovic, *Fellow, IEEE*

Abstract—For multiple heterogeneous multicore server processors across clouds and data centers, the aggregated performance of the cloud of clouds can be optimized by load distribution and balancing. Energy efficiency is one of the most important issues for large-scale server systems in current and future data centers. The multicore processor technology provides new levels of performance and energy efficiency. The present paper aims to develop power and performance constrained load distribution methods for cloud computing in current and future large-scale data centers. In particular, we address the problem of optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers. Our strategy is to formulate optimal power allocation and load distribution for multiple servers in a cloud of clouds as optimization problems, i.e., power constrained performance optimization and performance constrained power optimization. Our research problems in large-scale data centers are well defined multivariable optimization problems, which explore the power-performance tradeoff by fixing one factor and minimizing the other, from the perspective of optimal load distribution. It is clear that such power and performance optimization is important for a cloud computing provider to efficiently utilize all the available resources. We model a multicore server processor as a queueing system with multiple servers. Our optimization problems are solved for two different models of core speed, where one model assumes that a core runs at zero speed when it is idle, and the other model assumes that a core runs at a constant speed. Our results in this paper provide new theoretical insights into power management and performance optimization in data centers.

Index Terms—Load distribution, multicore server processor, power allocation, queueing model, response time.

1 Introduction

1.1 Motivation

Cloud computing has recently received considerable attention and is widely accepted as a promising and ultimate way of managing and improving the utilization of data and computing center resources and delivering various computing and IT services. Currently, server sprawl is a common situation, in which multiple under-utilized servers take up more space and consume more energy than can be justified by their workload. According to a recent report [2], many companies typically run at 15-20% of their capacity, which is not a sustainable ratio in the current economic environment. Server consolidation is an effective cloud computing approach to increasing the efficient usage of computer server resources in order to reduce the total number of servers or server locations that an organization requires. By centralized management of computing resources, cloud computing delivers hosted services over the Internet, such that access to shared hardware, software, databases, information, and all resources are provided to users on-demand.

In a data center with multiple servers, the aggregated performance of the data center can be optimized by *load distribution and balancing*. Cloud-based applications depend even more heavily on load balancing and optimization than traditional enterprise applications. For end users, load balancing capabilities will be seriously considered when they select a cloud computing provider. For cloud providers, load balancing capabilities will be a source of revenue, which is directly related to service quality (e.g., task response time). Hence, an efficient load balancing strategy is a key component to building out any cloud computing architecture.

Power and cooling costs for data centers have skyrocketed by 800% since 1996, and the escalating costs see no end in sight [15]. *Energy efficiency* is one of the

- *J. Cao is with the Research Institute of Information Technology, Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing 100084, China. E-mail: jcao@mail.tsinghua.edu.cn*
- *K. Li is with the Department of Computer Science, State University of New York, New Paltz, New York 12561, USA. E-mail: lik@newpaltz.edu (This is the author for correspondence.)*
- *I. Stojmenovic is with the School of Electrical Engineering and Computer Science, University of Ottawa, Ontario K1N 6N5, Canada; and the School of Software, Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing 100084, China. E-mail: ivan@site.uottawa.ca*

most important issues for large-scale server systems in current and future data centers. Cloud computing provides the ultimate way of effective and efficient management of energy consumption. Cloud computing can be an inherently energy-efficient technology, due to centralized energy management of computations on large-scale computing systems, instead of distributed and individualized applications without efficient energy consumption control [6]. Moreover, such potential for significant energy savings can be fully explored with balanced consideration of system performance and energy consumption.

Permanently altering the course of computing, the *multicore processor* technology provides new levels of performance and energy efficiency. Ideal for multitasking, multimedia, and networking applications, multicore technology delivers exceptional energy-efficient performance for the ultimate computing experience. Incorporating multiple processor cores in a single package that delivers parallel execution of multiple software applications, multicore technology enables higher levels of performance and consumes less power typically required by a higher frequency single core processor with equivalent performance.

Software techniques for power reduction are supported by a mechanism called *dynamic voltage scaling* (equivalently, dynamic frequency scaling, dynamic speed scaling, dynamic power scaling). A power-aware system management algorithm can change supply voltage and frequency at appropriate times to optimize a combined consideration of performance and energy consumption. Dynamic power management at the operating system level provides supply voltage and clock frequency adjustment schemes implemented while tasks are running. These energy conservation techniques explore the opportunities for tuning the energy-delay tradeoff [28].

As cloud computing evolves into the next generation, a cloud of clouds is able to utilize computing resources across multiple clouds and data centers. Such cloud federation not only integrates heterogeneous resources and computing power, but also provides more powerful and flexible services to various requests in scientific, business, and industrial applications. In such an environment with multiple heterogeneous servers across multiple clouds and data centers, optimal power allocation and load distribution makes even more impact to the aggregated performance and energy efficiency of a cloud of clouds. Such a situation provides us a more challenging opportunity to address and discuss power and performance in a wider scale and to generate more significant influence.

1.2 Our Contributions

The present paper aims to develop power and performance constrained load distribution methods for cloud computing in current and future large-scale data centers. In particular, we address the problem of optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers. We define two important research problems which explore the power-performance tradeoff in large-scale data centers from the perspective of optimal power allocation and load distribution. Our strategy is to formulate *optimal power allocation and load distribution* for multiple servers in a cloud of clouds as optimization problems. Our problems are defined for multiple multicore server processors with different sizes, and certain workload.

- *Power constrained performance optimization* – Given a power constraint, our problem is to find an optimal power allocation to the servers (i.e., to determine the server speeds) and an optimal workload distribution among the servers, such that the average task response time is minimized and that the average power consumption of the servers does not exceed the given power limit.
- *Performance constrained power optimization* – Given a performance constraint, our problem is to find an optimal power allocation to the servers (i.e., to determine the server speeds) and an optimal workload distribution among the servers, such that the average power consumption of the servers is minimized and that the average task response time does not exceed the given performance limit.

Our research problems in large-scale data centers are well defined multivariable optimization problems, which explore the power-performance tradeoff by fixing one factor and minimizing the other, from the perspective of optimal load distribution. By power constrained performance optimization, a cloud of clouds can deliver the highest quality of service, while maintaining low cost for energy consumption. By performance constrained power optimization, a cloud of clouds can minimize the cost for energy consumption, while guaranteeing certain quality of service.

We model a multicore server processor as a queueing system with multiple servers. Our optimization problems are solved for two different models of core speed, where one model assumes that a core runs at zero speed when it is idle, and the other model assumes that a core runs at a constant speed. Our investigation in this paper makes initial effort to analytical study of power-performance tradeoff in data centers with heterogeneous servers. Our results in this paper provide new theoretical and practical insights into power management and performance optimization in cloud computing. To the best of our knowledge, such optimal power allocation and load distribution for multiple heterogeneous servers has not been investigated before as optimization problems.

2 Related Research

Load distribution and balancing in general parallel and distributed computing systems have been extensively studied and a huge body of literature exists (see the excellent collection [27]). In particular, the problems of optimal load distribution have been investigated by using queueing models [16], [30], with various performance metrics such as weighted mean response time [19], arithmetic average response time [20], probability of load imbalance [21], probability of load balancing success [25], mean response ratio [29], and mean miss rate [14]. Optimal load distribution in a heterogeneous distributed computer system with both generic and dedicated applications was studied in [7], [26]. Optimal load distribution of generic tasks without special tasks for a group of heterogeneous multiserver queueing systems was studied in [14]. Optimal load distribution of generic tasks together with special tasks in cluster and grid computing environments was studied in [22], where each server is modeled as a queueing system with a single server. Optimal load distribution of generic tasks together with special tasks for a group of heterogeneous blade servers in a cloud computing environment was investigated in [24], where each server is modeled as a queueing system with multiple servers. To the best of our knowledge, optimal load distribution with power and performance constraints for multiple queueing systems with multiple servers has not been considered before.

There exists a huge body of literature on energy-efficient computing and communication (see [3], [4], [5], [31], [32] for comprehensive surveys). Efficient power management and performance optimization in large-scale data centers and server clusters has gained much attention in the research community in recent years [9], [10], [11], [12], [13]. In [17], the authors developed a framework for hierarchical autonomic power and performance management in high-performance distributed data centers. In [23], the author formulated and solved the optimal power allocation problem for multiple heterogeneous servers in a data center. In [33], the authors proposed a highly scalable hierarchical power control architecture for large-scale data centers. In [34], the authors presented a novel cluster-level control architecture that coordinates individual power and performance control loops for virtualized server clusters. In [36], the authors proposed an energy proportional model based on queuing theory and service differentiation in server clusters, which can provide controllable and predictable quantitative control over power consumption with theoretically guaranteed service performance, where a server is treated as an M/G/1 queueing system, i.e., a single server system. Furthermore, the vary-on vary-off mechanism is adopted which turns servers on and off to adjust the number of active servers based on the workload.

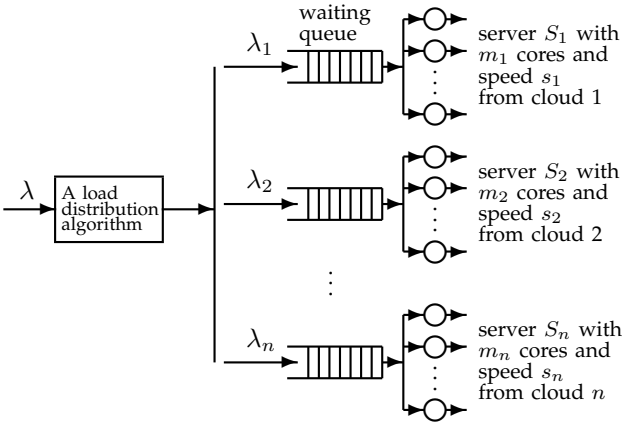

Fig. 1. A group of n heterogeneous multicore servers $S_1, S_2, ..., S_n$.

3 Modeling Multicore Server Processors

To formulate and study the problem of optimal load distribution and balancing for multiple heterogeneous multicore server processors in a cloud computing environment with power and performance constraints, we need a model for a multicore server and a group of multicore servers. A queueing model for a group of n heterogeneous multicore servers S_1, S_2, ..., S_n of sizes m_1, m_2, ..., m_n and speeds s_1, s_2, ..., s_n across multiple clouds and data centers is given in Figure 1. Assume that a multicore server S_i has m_i identical cores with speed s_i. 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 (measured by the number of tasks per second), i.e., the inter-arrival times are independent and identically distributed (i.i.d.) exponential random variables with mean $1/\lambda$. A load distribution and balancing algorithm will split the stream into n substreams, such that the ith substream with arrival rate λ_i is sent to server S_i, where $1 \leq i \leq n$, and $\lambda = \lambda_1 + \lambda_2 + \cdots + \lambda_n$. A multicore server S_i maintains a queue with infinite capacity for waiting tasks when all the m_i cores are busy. The first-come-first-served (FCFS) queueing discipline is adopted by all servers. 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_i cores of server S_i have identical execution speed s_i (measured by billion instructions per second (BIPS)). Hence, the task execution times on the cores of server S_i are i.i.d. exponential random variables $x_i = r/s_i$ with mean $\bar{x}_i = \bar{r}/s_i$.

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 $P = aCV^2f$, where a is an activity factor, C is the loading capacitance, V is the supply voltage, and f is the clock frequency [8]. Since $s \propto f$, where s is the processor speed, and $f \propto V^\phi$ with $0 < \phi \leq 1$ [35], which implies that $V \propto f^{1/\phi}$, we know that power consumption is $P \propto f^\alpha$ and $P \propto s^\alpha$, where $\alpha = 1+2/\phi \geq 3$. 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. Let $\mu_i = 1/\bar{x}_i = s_i/\bar{r}$ be the average service rate, i.e., the average number of tasks that can be finished by a processor core of server S_i in one unit of time. The core utilization is $\rho_i = \lambda_i/(m_i\mu_i) = \lambda_i\bar{x}_i/m_i = (\lambda_i/m_i) \cdot (\bar{r}/s_i)$, which is the average percentage of time that a core of S_i is busy. Since the power for speed s_i is s_i^α, the average amount of energy consumed by a core in one unit of time is $\rho_i s_i^\alpha = (\lambda_i/m_i)\bar{r}s_i^{\alpha-1}$, where we notice that the speed of a core is zero when it is idle. The average amount of energy consumed by an m_i-core server S_i in one unit of time, i.e., the average power supply to server S_i, is $P_i = m_i\rho_i s_i^\alpha = \lambda_i\bar{r}s_i^{\alpha-1}$, where $m_i\rho_i = \lambda_i\bar{x}_i$ is the average number of busy cores in S_i. Since a processor core still consumes some amount of power P_i^* even when it is idle (assume that an idle core consumes certain base power P_i^*, which includes static power dissipation, short circuit power dissipation, and other leakage and wasted power [1]), we will include P_i^* in P_i, i.e., $P_i = m_i(\rho_i s_i^\alpha + P_i^*) = \lambda_i\bar{r}s_i^{\alpha-1} + m_iP_i^*$. Notice that when $P_i^* = 0$, the above P_i is independent of m_i.

In the *constant-speed model*, all cores run at the speed s_i even if there is no task to perform. Again, we use P_i to represent the power allocated to server S_i. Since the power for speed s_i is s_i^α, the power allocated to server S_i is $P_i = m_i(s_i^\alpha + P_i^*)$.

4 Power Constrained Performance Optimization

Let $p_{i,k}$ denote the probability that there are k tasks (waiting or being processed) in the M/M/m system for S_i. Then, we have ([18], p. 102)

$$p_{i,k} = \begin{cases} p_{i,0}\dfrac{(m_i\rho_i)^k}{k!}, & k \leq m_i; \\ \\ p_{i,0}\dfrac{m_i^{m_i}\rho_i^k}{m_i!}, & k \geq m_i; \end{cases}$$

where

$$p_{i,0} = \left(\sum_{k=0}^{m_i-1}\frac{(m_i\rho_i)^k}{k!} + \frac{(m_i\rho_i)^{m_i}}{m_i!}\cdot\frac{1}{1-\rho_i}\right)^{-1}.$$

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

$$P_{q,i} = \frac{p_{i,m_i}}{1-\rho_i} = p_{i,0}\frac{m_i^{m_i}}{m_i!}\cdot\frac{\rho_i^{m_i}}{1-\rho_i}.$$

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

$$\bar{N}_i = \sum_{k=0}^{\infty} kp_{i,k} = m_i\rho_i + \frac{\rho_i}{1-\rho_i}P_{q,i}.$$

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

$$T_i = \frac{\bar{N}_i}{\lambda_i} = \bar{x}_i + \frac{P_{q,i}}{m_i(1-\rho_i)}\bar{x}_i = \bar{x}_i\left(1 + \frac{P_{q,i}}{m_i(1-\rho_i)}\right).$$

In other words, the average task response time on multicore server S_i is

$$\begin{aligned} T_i &= \bar{x}_i\left(1 + p_{i,0}\frac{m_i^{m_i-1}}{m_i!}\cdot\frac{\rho_i^{m_i}}{(1-\rho_i)^2}\right) \\ &= \frac{\bar{r}}{s_i}\left(1 + p_{i,0}\frac{m_i^{m_i-1}}{m_i!}\cdot\frac{\rho_i^{m_i}}{(1-\rho_i)^2}\right). \end{aligned}$$

The average task response time T on a group of n heterogeneous multicore servers $S_1, S_2, ..., S_n$ is

$$T = \frac{\lambda_1}{\lambda}T_1 + \frac{\lambda_2}{\lambda}T_2 + \cdots + \frac{\lambda_n}{\lambda}T_n.$$

Notice that the above T can be treated as a function of load distribution $\lambda_1, \lambda_2, ..., \lambda_n$ and server speeds $s_1, s_2, ...s_n$, i.e., $T(\lambda_1, \lambda_2, ..., \lambda_n, s_1, s_2, ..., s_n)$.

The average power consumption $P(\lambda_1, \lambda_2, ..., \lambda_n, s_1, s_2, ..., s_n)$ of a group of n heterogeneous multicore servers $S_1, S_2, ..., S_n$ is also a function of load distribution $\lambda_1, \lambda_2, ..., \lambda_n$ and server speeds $s_1, s_2, ...s_n$. For the idle-speed model, we have

$$\begin{aligned} &P(\lambda_1, \lambda_2, ..., \lambda_n, s_1, s_2, ..., s_n) = \sum_{i=1}^{n} P_i \\ &= \sum_{i=1}^{n} m_i(\rho_i s_i^\alpha + P_i^*) = \sum_{i=1}^{n}\lambda_i\bar{r}s_i^{\alpha-1} + \sum_{i=1}^{n} m_iP_i^*. \end{aligned}$$

For the constant-speed model, we have

$$\begin{aligned} &P(\lambda_1, \lambda_2, ..., \lambda_n, s_1, s_2, ..., s_n) = \sum_{i=1}^{n} P_i \\ &= \sum_{i=1}^{n} m_i(s_i^\alpha + P_i^*) = \sum_{i=1}^{n} m_i s_i^\alpha + \sum_{i=1}^{n} m_iP_i^*, \end{aligned}$$

which is actually a function of server speeds $s_1, s_2, ...s_n$, i.e., $P(s_1, s_2, ..., s_n)$.

Our *optimal power allocation and load distribution* problem for multiple heterogeneous multicore server processors can be specified as follows: given the number of multicore servers n, the sizes of the servers $m_1, m_2, ..., m_n$, the base power supply $P_1^*, P_2^*, ..., P_n^*$,

the average task execution requirement $\bar{r}$, the task arrival rate λ, and the available power $\tilde{P}$, find the task arrival rates on the servers $\lambda_1, \lambda_2, ..., \lambda_n$, and the server speeds $s_1, s_2, ..., s_n$, such that the average task response time $T(\lambda_1, \lambda_2, ..., \lambda_n, s_1, s_2, ..., s_n)$ is minimized, subject to the constraints $F(\lambda_1, \lambda_2, ..., \lambda_n) = \lambda$, where $F(\lambda_1, \lambda_2, ..., \lambda_n) = \lambda_1 + \lambda_2 + \cdots + \lambda_n$, and $\rho_i < 1$, i.e., $\lambda_i < m_i/\bar{x}_i = m_i s_i/\bar{r}$, for all $1 \le i \le n$; and $P(\lambda_1, \lambda_2, ..., \lambda_n, s_1, s_2, ..., s_n) = \tilde{P}$, namely,

$$G(\lambda_1, \lambda_2, ..., \lambda_n, s_1, s_2, ..., s_n) = \sum_{i=1}^{n} \lambda_i \bar{r} s_i^{\alpha-1} = \tilde{P} - \sum_{i=1}^{n} m_i P_i^*,$$

for the idle-speed model, and

$$G(s_1, s_2, ..., s_n) = \sum_{i=1}^{n} m_i s_i^{\alpha} = \tilde{P} - \sum_{i=1}^{n} m_i P_i^*,$$

for the constant-speed model. The optimization problem defined above is a *power constrained performance optimization* problem, i.e., optimizing performance (minimizing the average task response time) subject to power consumption constraint.

To solve the above well defined multivariable optimization problem, one can employ the classic method of Lagrange multiplier. However, this method results in a system of nonlinear equations with $2(n + 1)$ variables, i.e., $\lambda_1, \lambda_2, ..., \lambda_n$, $s_1, s_2, ..., s_n$, and two Lagrange multipliers. These equations are nontrivial to solve. There is absolutely no closed-form solution. The main contribution of the paper is to develop an effective and efficient algorithm to find a numerical solution. Our strategy is to carefully analyze the problem structure and to reveal the intricate relationship between the speed and the workload of a server. Our analysis successfully reduces the $2(n + 1)$ variables into $n + 1$ variables, i.e., $\lambda_1, \lambda_2, ..., \lambda_n$ and a Lagrange multiplier. We then develop two procedures to find $\lambda_1, \lambda_2, ..., \lambda_n$ and the Lagrange multiplier respectively. Since the standard bisection method is used to search each value in a small interval, our algorithm is very time efficient in finding a solution to the multivariable optimization problem.

The above centralized problem definition and solution can be justified in two ways. First, the main approach in cloud computing is centralized resource management and performance optimization to minimize the cost of computing and to maximize the quality of service. Second, our optimization problem is solved once a while for a cloud based on stable statistical information such as λ and $\bar{r}$. As the user demand in an application environment changes, the optimization problem can be solved again to reconfigure the servers, so that system operation and performance can be optimized for the new user requirement.

5 The Idle-Speed Model

First of all, we establish the following result, which gives the optimal server speed setting for the idle-speed model.

Theorem 1: For the idle-speed model, the average task response time T on n multicore servers is minimized when all servers have the same speed, i.e., $s_1 = s_2 = \cdots = s_n = s$, where

$$s = \left(\frac{1}{\lambda \bar{r}} \left(\tilde{P} - \sum_{i=1}^{n} m_i P_i^* \right) \right)^{1/(\alpha-1)}.$$

Proof. We can minimize T by using the method of Lagrange multiplier, namely,

$$\nabla T(\lambda_1, \lambda_2, ..., \lambda_n, s_1, s_2, ..., s_n) = \phi \nabla F(\lambda_1, \lambda_2, ..., \lambda_n) + \psi \nabla G(\lambda_1, \lambda_2, ..., \lambda_n, s_1, s_2, ..., s_n),$$

that is,

$$\frac{\partial T}{\partial \lambda_i} = \phi \frac{\partial F}{\partial \lambda_i} + \psi \frac{\partial G}{\partial \lambda_i} = \phi + \psi \bar{r} s_i^{\alpha-1},$$

for all $1 \le i \le n$, where ϕ is a Lagrange multiplier and ψ is another Lagrange multiplier, and

$$\frac{\partial T}{\partial s_i} = \psi \frac{\partial G}{\partial s_i} = \psi \lambda_i \bar{r} (\alpha - 1) s_i^{\alpha-2},$$

for all $1 \le i \le n$.

It is clear that

$$\frac{\partial T}{\partial \lambda_i} = \frac{1}{\lambda} \left(T_i + \lambda_i \frac{\partial T_i}{\partial \lambda_i} \right),$$

where

$$\frac{\partial T_i}{\partial \lambda_i} = \frac{\partial T_i}{\partial \rho_i} \cdot \frac{\partial \rho_i}{\partial \lambda_i} = \frac{\bar{r}}{m_i s_i} \cdot \frac{\partial T_i}{\partial \rho_i},$$

that is,

$$\frac{\partial T}{\partial \lambda_i} = \frac{1}{\lambda} \left(T_i + \rho_i \frac{\partial T_i}{\partial \rho_i} \right).$$

Hence, we get

$$\frac{1}{\lambda} \left(T_i + \rho_i \frac{\partial T_i}{\partial \rho_i} \right) = \phi + \psi \bar{r} s_i^{\alpha-1}, \tag{1}$$

where

$$\frac{\partial T_i}{\partial \rho_i} = \frac{m_i^{m_i-1}}{m_i!} \cdot \frac{\bar{r}}{s_i} \left(\frac{\partial p_{i,0}}{\partial \rho_i} \cdot \frac{\rho_i^{m_i}}{(1-\rho_i)^2} + p_{i,0} \frac{\rho_i^{m_i-1}(m_i - (m_i - 2)\rho_i)}{(1-\rho_i)^3} \right),$$

and

$$\frac{\partial p_{i,0}}{\partial \rho_i} = -p_{i,0}^2 \left(\sum_{k=1}^{m_i-1} \frac{m_i^k \rho_i^{k-1}}{(k-1)!} + \frac{m_i^{m_i}}{m_i!} \cdot \frac{\rho_i^{m_i-1}(m_i - (m_i - 1)\rho_i)}{(1-\rho_i)^2} \right),$$

for all $1 \le i \le n$.

It is also clear that

$$\frac{\partial T}{\partial s_i} = \frac{\lambda_i}{\lambda} \cdot \frac{\partial T_i}{\partial s_i}.$$

Let $T_i = (\bar{r}/s_i)D_i = \bar{r}(D_i/s_i)$, where

$$D_i = 1 + p_{i,0}\frac{m_i^{m_i-1}}{m_i!} \cdot \frac{\rho_i^{m_i}}{(1-\rho_i)^2}.$$

Then, we have

$$\begin{aligned}\frac{\partial T_i}{\partial s_i} &= \bar{r}\left(-\frac{D_i}{s_i^2} + \frac{1}{s_i} \cdot \frac{\partial D_i}{\partial \rho_i} \cdot \frac{\partial \rho_i}{\partial s_i}\right) \\ &= -\bar{r}\left(\frac{1}{\bar{r}s_i}T_i + \frac{1}{s_i} \cdot \frac{\partial D_i}{\partial \rho_i} \cdot \frac{\lambda_i \bar{r}}{m_i s_i^2}\right) \\ &= -\bar{r}\left(\frac{1}{\bar{r}s_i}T_i + \frac{\rho_i}{s_i^2} \cdot \frac{\partial D_i}{\partial \rho_i}\right).\end{aligned}$$

Since

$$\begin{aligned}\frac{\partial D_i}{\partial \rho_i} &= \frac{m_i^{m_i-1}}{m_i!}\left(\frac{\partial p_{i,0}}{\partial \rho_i} \cdot \frac{\rho_i^{m_i}}{(1-\rho_i)^2} \right. \\ &\quad \left. + p_{i,0}\frac{\rho_i^{m_i-1}(m_i-(m_i-2)\rho_i)}{(1-\rho_i)^3}\right) \\ &= \frac{s_i}{\bar{r}} \cdot \frac{\partial T_i}{\partial \rho_i},\end{aligned}$$

we get

$$\frac{\partial T_i}{\partial s_i} = -\frac{1}{s_i}\left(T_i + \rho_i \frac{\partial T_i}{\partial \rho_i}\right),$$

which implies that

$$\begin{aligned}\frac{\partial T}{\partial s_i} = \frac{\lambda_i}{\lambda} \cdot \frac{\partial T_i}{\partial s_i} &= -\frac{\lambda_i}{\lambda s_i}\left(T_i + \rho_i \frac{\partial T_i}{\partial \rho_i}\right) \\ &= \psi \lambda_i \bar{r}(\alpha - 1)s_i^{\alpha-2}, \end{aligned} \tag{2}$$

namely,

$$-\frac{1}{\lambda \bar{r}(\alpha-1)s_i^{\alpha-1}}\left(T_i + \rho_i \frac{\partial T_i}{\partial \rho_i}\right) = \psi, \tag{3}$$

for all $1 \le i \le n$.

Based on the above discussion, i.e., Eqs. (1) and (3), we have $-\bar{r}(\alpha-1)s_i^{\alpha-1} = \phi/\psi + \bar{r}s_i^{\alpha-1}$, which gives rise to $s_i = \left(-(\phi/\psi) \cdot 1/(\alpha\bar{r})\right)^{1/(\alpha-1)}$, for all $1 \le i \le n$. The above result means that all servers must have the same speed s, i.e., $s_1 = s_2 = \cdots = s_n = s$, where

$$s = \left(-\frac{\phi}{\psi} \cdot \frac{1}{\alpha\bar{r}}\right)^{1/(\alpha-1)}.$$

Consequently, from the constraint

$$\begin{aligned}&G(\lambda_1, \lambda_2, ..., \lambda_n, s_1, s_2, ..., s_n) \\ &= \sum_{i=1}^{n} \lambda_i \bar{r} s_i^{\alpha-1} = \lambda \bar{r} s^{\alpha-1} = \tilde{P} - \sum_{i=1}^{n} m_i P_i^*,\end{aligned}$$

we get the identical server speed as

$$s = \left(\frac{1}{\lambda\bar{r}}\left(\tilde{P} - \sum_{i=1}^{n} m_i P_i^*\right)\right)^{1/(\alpha-1)}.$$

Hence, the theorem is proven. ∎

By the above theorem, we immediately get

$$\begin{aligned}P_i &= \lambda_i \bar{r} s_i^{\alpha-1} + m_i P_i^* = \lambda_i \bar{r} s^{\alpha-1} + m_i P_i^* \\ &= \frac{\lambda_i}{\lambda}\left(\tilde{P} - \sum_{i=1}^{n} m_i P_i^*\right) + m_i P_i^*,\end{aligned}$$

for all $1 \le i \le n$. It remains to find $\lambda_1, \lambda_2, ..., \lambda_n$. We will describe our method in Section 6.

From the above proof, we also know that $\psi \bar{r} s_i^{\alpha-1} = \psi \bar{r} s^{\alpha-1} = -\phi/\alpha$. Therefore, by Eq. (1), we have

$$\frac{1}{\lambda}\left(T_i + \rho_i \frac{\partial T_i}{\partial \rho_i}\right) = \left(1 - \frac{1}{\alpha}\right)\phi, \tag{4}$$

for all $1 \le i \le n$.

6 The Constant-Speed Model

The following result gives the optimal server speed setting for the constant-speed model.

Theorem 2: For the constant-speed model, the average task response time T on n multicore servers is minimized when $s_i = (\lambda_i/(\beta m_i))^{1/\alpha}$, for all $1 \le i \le n$, where

$$\beta = \frac{\lambda}{\tilde{P} - (m_1 P_1^* + m_2 P_2^* + \cdots + m_n P_n^*)}.$$

Proof. Again, we can minimize T by using the method of Lagrange multiplier, namely,

$$\begin{aligned}&\nabla T(\lambda_1, \lambda_2, ..., \lambda_n, s_1, s_2, ..., s_n) \\ &= \phi \nabla F(\lambda_1, \lambda_2, ..., \lambda_n) + \psi \nabla G(s_1, s_2, ..., s_n),\end{aligned}$$

that is,

$$\frac{\partial T}{\partial \lambda_i} = \phi \frac{\partial F}{\partial \lambda_i} = \phi,$$

for all $1 \le i \le n$, where ϕ is a Lagrange multiplier, and

$$\frac{\partial T}{\partial s_i} = \psi \frac{\partial G}{\partial s_i} = \psi m_i \alpha s_i^{\alpha-1},$$

for all $1 \le i \le n$, where and ψ is another Lagrange multiplier.

Notice that Eq. (1) becomes

$$\frac{1}{\lambda}\left(T_i + \rho_i \frac{\partial T_i}{\partial \rho_i}\right) = \phi. \tag{5}$$

Also, Eq. (2) becomes

$$\frac{\partial T}{\partial s_i} = -\frac{\lambda_i}{\lambda s_i}\left(T_i + \rho_i \frac{\partial T_i}{\partial \rho_i}\right) = \alpha \psi m_i s_i^{\alpha-1},$$

namely,

$$\frac{1}{\lambda}\left(T_i + \rho_i \frac{\partial T_i}{\partial \rho_i}\right) = -\frac{\alpha \psi m_i s_i^{\alpha}}{\lambda_i}. \tag{6}$$

Based on Eqs. (5) and (6), we get $\phi = -\alpha\psi(m_i s_i^{\alpha}/\lambda_i)$, and equivalently, $\lambda_i = -\alpha(\psi/\phi)m_i s_i^{\alpha}$. From the last equation, and the constraint that

$$\begin{aligned}G(s_1, s_2, ..., s_n) &= m_1 s_1^{\alpha} + m_2 s_2^{\alpha} + \cdots + m_n s_n^{\alpha} \\ &= \tilde{P} - \sum_{i=1}^{n} m_i P_i^*,\end{aligned}$$

we obtain

$$\lambda = -\left(\alpha\frac{\psi}{\phi}\right)\left(\tilde{P} - \sum_{i=1}^{n} m_i P_i^*\right),$$

which implies that $\lambda_i = \beta m_i s_i^\alpha$, and $s_i = (\lambda_i/(\beta m_i))^{1/\alpha}$, for all $1 \le i \le n$, where

$$\beta = -\alpha\frac{\psi}{\phi} = \frac{\lambda}{\tilde{P} - (m_1 P_1^* + m_2 P_2^* + \cdots + m_n P_n^*)}.$$

This proves the theorem. ∎

By the above theorem, we immediately get the power allocation as

$$P_i = m_i(s_i^\alpha + P_i^*) = m_i\left(\frac{\lambda_i}{\beta m_i} + P_i^*\right)$$
$$= \frac{\lambda_i}{\lambda}\left(\tilde{P} - \sum_{i=1}^{n} m_i P_i^*\right) + m_i P_i^*,$$

for all $1 \le i \le n$. Notice that the above expression of P_i is the same as that of the idle-speed model; however, the actual value of P_i is different because λ_i is different; also the resulting speed s_i is different.

The above theorem essentially means that T_i can be viewed as a function of λ_i, and T can be viewed as a function of $\lambda_1, \lambda_2, ..., \lambda_n$. It remains to find $\lambda_1, \lambda_2, ..., \lambda_n$. The method will be presented in the next section. Notice that if we keep the relation $s_i = (\lambda_i/(\beta m_i))^{1/\alpha}$, then ρ_i only depends on (and is an increasing function of) λ_i, and we have

$$\rho_i = \frac{\lambda_i \bar{r}}{m_i s_i} = \beta\bar{r}s_i^{\alpha-1} = \beta\bar{r}\left(\frac{\lambda_i}{\beta m_i}\right)^{(\alpha-1)/\alpha}$$
$$= \beta^{1/\alpha}\bar{r}\left(\frac{\lambda_i}{m_i}\right)^{(\alpha-1)/\alpha},$$

which implies that

$$\frac{\partial\rho_i}{\partial\lambda_i} = \frac{\beta^{1/\alpha}\bar{r}}{m_i}(1-1/\alpha)\left(\frac{\lambda_i}{m_i}\right)^{-1/\alpha},$$

and

$$\lambda_i\frac{\partial\rho_i}{\partial\lambda_i} = \beta^{1/\alpha}\bar{r}(1-1/\alpha)\left(\frac{\lambda_i}{m_i}\right)^{1-1/\alpha} = (1-1/\alpha)\rho_i,$$

for all $1 \le i \le n$. Thus, we obtain

$$\frac{\partial T}{\partial\lambda_i} = \frac{1}{\lambda}\left(T_i + \lambda_i\frac{\partial\rho_i}{\partial\lambda_i}\cdot\frac{\partial T_i}{\partial\rho_i}\right) = \frac{1}{\lambda}\left(T_i + (1-1/\alpha)\rho_i\frac{\partial T_i}{\partial\rho_i}\right),$$

and similar to Eq. (4), we get

$$\frac{1}{\lambda}\left(T_i + (1-1/\alpha)\rho_i\frac{\partial T_i}{\partial\rho_i}\right) = \phi, \quad (7)$$

for all $1 \le i \le n$. We observe that Eqs. (4) and (7) are very similar and can be solve by using the same algorithm.

Algorithm: Find_λ_i $(m_i, \bar{r}, \lambda, idle, \phi, s, \beta)$.

Input: $m_i, \bar{r}, \lambda, idle, \phi, s, \beta$.
Output: λ_i.

```
//Initialize [lb, ub] for λ_i                                   (1)
lb ← 0;                                                         (2)
ub ← idle ? m_i s/r̄ : m_i/(β^{1/(α−1)} r̄^{α/(α−1)});           (3)
//Search λ_i in [lb, ub]                                        (4)
while (ub − lb > ε)                                             (5)
    middle ← (lb + ub)/2;                                       (6)
    λ_i ← middle;                                               (7)
    s_i ← idle ? s : (λ_i/βm_i)^{1/α};                          (8)
    if (1/λ(T_i + (idle ? 1 : (1 − 1/α))ρ_i ∂T_i/∂ρ_i)
            < (idle ? (1 − 1/α) : 1)φ)                          (9)
        lb ← middle;                                            (10)
    else                                                        (11)
        ub ← middle;                                            (12)
    end if;                                                     (13)
end while;                                                      (14)
λ_i ← (lb + ub)/2;                                              (15)
return λ_i.                                                     (16)
```

Fig. 2. An algorithm to find λ_i.

7 The Algorithm

Since Eqs. (4) and (7) are very similar, the optimal load distribution $\lambda_1, \lambda_2, ..., \lambda_n$ and the optimal server speeds $s_1, s_2, ..., s_n$ (i.e., the optimal power allocation $P_1, P_2, ..., P_n$) as well as the optimal task response time T can be found by using the same algorithm for the two core speed and power consumption models. We use a boolean parameter $idle$ to distinguish the two models, namely, we refer to the idle-speed model when $idle$ is true, and the constant-speed model when $idle$ is false.

Given n, $m_1, m_2, ..., m_n$, $P_1^*, P_2^*, ..., P_n^*$, $\bar{r}$, λ, $\tilde{P}$, and $idle$, our optimal power allocation and load distribution algorithm to find ϕ, $\lambda_1, \lambda_2, ..., \lambda_n$, $s_1, s_2, ..., s_n$, and T is given in Figure 3. The algorithm uses another subalgorithm Find_λ_i described in Figure 2, which, given m_i, $\bar{r}$, λ, $idle$, ϕ, s, and β, finds λ_i that satisfies Eq. (4) or Eq. (7).

The key observation is that the left-hand sides of Eqs. (4) and (7) are increasing functions of λ_i. Therefore, given ϕ, we can find λ_i by using the bisection method to search λ_i in certain interval $[lb, ub]$ (lines (5)–(14) in Figure 2). We set lb simply as 0 (line (2)). For ub, we notice that

$$\lambda_i = \frac{\rho_i m_i s}{\bar{r}} < \frac{m_i s}{\bar{r}},$$

for the idle-speed model with s given in Theorem 1, and

$$\lambda_i = \frac{m_i \rho_i^{\alpha/(\alpha-1)}}{\beta^{1/(\alpha-1)}\bar{r}^{\alpha/(\alpha-1)}} < \frac{m_i}{\beta^{1/(\alpha-1)}\bar{r}^{\alpha/(\alpha-1)}},$$

for the constant-speed model with β given in Theorem 2, where $1 \le i \le n$. Hence, ub is set in line

Algorithm: Calculate_T.

Input: n, $m_1, m_2, ..., m_n$, $P_1^*, P_2^*, ..., P_n^*$, $\bar{r}$, λ, $\tilde{P}$, $idle$.
Output: ϕ, $\lambda_1, \lambda_2, ..., \lambda_n$, $s_1, s_2, ..., s_n$, and T.

```
s ← ((1/λr̄)(P̃ − (m1P1* + ··· + mnPn*)))^{1/(α−1)};          (1)
β ← λ/(P̃ − (m1P1* + m2P2* + ··· + mnPn*));                  (2)
//Initialize [lb, ub] for φ                                  (3)
φ ← (1/λ)r̄/s;                                               (4)
do                                                           (5)
    φ ← 2φ;                                                  (6)
    for (i ← 1; i ≤ n; i++)                                  (7)
        λi ← Find_λi(mi, r̄, λ, idle, φ, s, β);               (8)
    end for;                                                 (9)
while (λ1 + λ2 + ··· + λn < λ);                              (10)
//Search φ in [lb, ub]                                       (11)
lb ← 0;                                                      (12)
ub ← φ;                                                      (13)
while (ub − lb > ε)                                          (14)
    middle ← (lb + ub)/2;                                    (15)
    φ ← middle;                                              (16)
    for (i ← 1; i ≤ n; i++)                                  (17)
        λi ← Find_λi(mi, r̄, λ, idle, φ, s, β);               (18)
    end for;                                                 (19)
    if (λ1 + λ2 + ··· + λn < λ)                              (20)
        lb ← middle;                                         (21)
    else                                                     (22)
        ub ← middle;                                         (23)
    end if;                                                  (24)
end while;                                                   (25)
φ ← (lb + ub)/2;                                             (26)
//Calculate λ1, λ2, ..., λn and s1, s2, ..., sn             (27)
for (i ← 1; i ≤ n; i++)                                      (28)
    λi ← Find_λi(mi, r̄, λ, idle, φ, s, β);                   (29)
    si ← idle ? s : (λi/βmi)^{1/α};                          (30)
end for;                                                     (31)
//Calculate and return T                                     (32)
T ← (λ1/λ)T1 + (λ2/λ)T2 + ··· + (λn/λ)Tn;                    (33)
return T.                                                    (34)
```

Fig. 3. An algorithm to calculate the minimized T.

(3) based on the above discussion, where the value of a conditional expression $c\ ?\ u\ :\ v$ is u if the condition c is true and v if the condition c is false. For a given λ_i, the server speed s_i is s given by Theorem 1, and $(\lambda_i/\beta m_i)^{1/\alpha}$ given by Theorem 2 (line (8)). The left-hand side of Eq. (4) or (7) is compared with $(1-1/\alpha)\phi$ or ϕ (line (9)), and the search interval $[lb, ub]$ is adjusted accordingly (lines (10) and (12)).

The value of ϕ can also be found by using the bisection method (lines (14)–(25) in Figure 3). The search interval $[lb, ub]$ for ϕ is determined as follows. We set lb simply as 0 (line (12)). As for ub, we notice that ϕ is at least $(1/\lambda)\bar{r}/s$ (line (4)) for both speed models, because $T_i \geq x_i = \bar{r}/s$. Then, ϕ is repeatedly doubled (line (7)), until the sum of the λ_i's found by Find_λ_i (line (8)) is at least λ (lines (5)–(10) and (13)). Once $[lb, ub]$ is decided, ϕ can be searched based on the fact that the λ_i's increase with ϕ (lines (20)–(24)). After ϕ is determined (line (26)), $\lambda_1, \lambda_2, ..., \lambda_n$ and $s_1, s_2, ..., s_n$ can be computed routinely (lines (28)–(31)) and T can be obtained easily (line (33)).

8 The Equal-Power Method

In this section, we analyze a baseline method, i.e., the equal-power method. The baseline method is a simple method without optimization applied.

8.1 The Analysis

In the *equal-power* method, all cores of all servers consume the same amount of power. Let $M = m_1 + m_2 + \cdots + m_n$ be the total number of cores of the n multicore servers.

8.1.1 The Idle-Speed Model

The following result gives the server speed setting for the idle-speed model.

Theorem 3: For the idle-speed model, we have $s_i = c_i/\lambda_i^{1/(\alpha-1)}$, where

$$c_i = \left(\frac{m_i}{\bar{r}}\left(\frac{\tilde{P}}{M} - P_i^*\right)\right)^{1/(\alpha-1)},$$

for all $1 \leq i \leq n$.

Proof. Since a core in server S_i consumes power $\rho_i s_i^\alpha + P_i^*$, and all cores consume the same amount of power, we have

$$\rho_i s_i^\alpha + P_i^* = \frac{\lambda_i \bar{r}}{m_i} s_i^{\alpha-1} + P_i^* = \frac{\tilde{P}}{M},$$

which gives

$$s_i = \left(\frac{m_i}{\lambda_i \bar{r}}\left(\frac{\tilde{P}}{M} - P_i^*\right)\right)^{1/(\alpha-1)},$$

for all $1 \leq i \leq n$. ∎

By the above theorem, we immediately get

$$\begin{aligned} P_i &= \lambda_i \bar{r} s_i^{\alpha-1} + m_i P_i^* \\ &= \lambda_i \bar{r} \cdot \frac{m_i}{\lambda_i \bar{r}}\left(\frac{\tilde{P}}{M} - P_i^*\right) + m_i P_i^* = \frac{m_i}{M} P_i^*, \end{aligned}$$

for all $1 \leq i \leq n$.

Since s_i is a function of λ_i and no longer an independent variable, our optimal power allocation and load distribution problem for multiple heterogeneous multicore server processors is modified as follows: given the number of multicore servers n, the sizes of the servers $m_1, m_2, ..., m_n$, the base power supply $P_1^*, P_2^*, ..., P_n^*$, the average task execution requirement $\bar{r}$, the task arrival rate λ, and the available power $\tilde{P}$, find the task arrival rates on the servers

$\lambda_1, \lambda_2, ..., \lambda_n$, such that the average task response time $T(\lambda_1, \lambda_2, ..., \lambda_n)$ is minimized, subject to the constraint

$$F(\lambda_1, \lambda_2, ..., \lambda_n) = \lambda_1 + \lambda_2 + \cdots + \lambda_n = \lambda,$$

and $\rho_i < 1$, for all $1 \leq i \leq n$.

By using the method of Lagrange multiplier, namely,

$$\nabla T(\lambda_1, \lambda_2, ..., \lambda_n) = \phi \nabla F(\lambda_1, \lambda_2, ..., \lambda_n)$$

that is,

$$\frac{\partial T}{\partial \lambda_i} = \phi \frac{\partial F}{\partial \lambda_i},$$

for all $1 \leq i \leq n$, where ϕ is a Lagrange multiplier, we get

$$\frac{1}{\lambda}\left(T_i + \lambda_i \frac{\partial T_i}{\partial \lambda_i}\right) = \phi.$$

Notice that

$$\frac{\partial s_i}{\partial \lambda_i} = -\frac{1}{\alpha - 1} \cdot \frac{c_i}{\lambda^{\alpha/(\alpha-1)}} = -\frac{1}{\alpha - 1} \cdot \frac{s_i}{\lambda_i},$$

and

$$\begin{aligned}\frac{\partial \rho_i}{\partial \lambda_i} &= \frac{\bar{r}}{m_i s_i} - \frac{\lambda_i \bar{r}}{m_i s_i^2} \cdot \frac{\partial s_i}{\partial \lambda_i} = \frac{\rho_i}{\lambda_i} - \frac{\rho_i}{s_i} \cdot \frac{\partial s_i}{\partial \lambda_i} \\ &= \rho_i \left(\frac{1}{\lambda_i} + \frac{1}{\alpha - 1} \cdot \frac{1}{\lambda_i}\right) = \frac{\alpha}{\alpha - 1} \cdot \frac{\rho_i}{\lambda_i}.\end{aligned}$$

Hence, we get

$$\begin{aligned}\frac{\partial T_i}{\partial \lambda_i} &= -\frac{\bar{r}}{s_i^2} \cdot \frac{\partial s_i}{\partial \lambda_i} \cdot T_i \frac{s_i}{\bar{r}} \\ &\quad + \frac{m_i^{m_i - 1}}{m_i!} \cdot \frac{\bar{r}}{s_i} \left(\frac{\partial p_{i,0}}{\partial \rho_i} \cdot \frac{\rho_i^{m_i}}{(1 - \rho_i)^2} \right. \\ &\quad \left. + p_{i,0} \frac{\rho_i^{m_i - 1}(m_i - (m_i - 2)\rho_i)}{(1 - \rho_i)^3}\right) \frac{\partial \rho_i}{\partial \lambda_i} \\ &= \frac{1}{\alpha - 1} \cdot \frac{T_i}{\lambda_i} \\ &\quad + \frac{\alpha}{\alpha - 1} \cdot \frac{m_i^{m_i - 1}}{m_i!} \cdot \frac{\rho_i}{\lambda_i} \cdot \frac{\bar{r}}{s_i} \left(\frac{\partial p_{i,0}}{\partial \rho_i} \cdot \frac{\rho_i^{m_i}}{(1 - \rho_i)^2} \right. \\ &\quad \left. + p_{i,0} \frac{\rho_i^{m_i - 1}(m_i - (m_i - 2)\rho_i)}{(1 - \rho_i)^3}\right),\end{aligned}$$

where

$$\begin{aligned}\frac{\partial p_{i,0}}{\partial \rho_i} = -p_{i,0}^2 &\left(\sum_{k=1}^{m_i - 1} \frac{m_i^k \rho_i^{k-1}}{(k-1)!} \right. \\ &\left. + \frac{m_i^{m_i}}{m_i!} \cdot \frac{\rho_i^{m_i - 1}(m_i - (m_i - 1)\rho_i)}{(1 - \rho_i)^2}\right),\end{aligned}$$

for all $1 \leq i \leq n$. Consequently, we have

$$\begin{aligned}&\frac{\alpha}{\alpha - 1} \cdot \frac{1}{\lambda}\left(T_i + \frac{m_i^{m_i - 1}}{m_i!} \rho_i \frac{\bar{r}}{s_i} \left(\frac{\partial p_{i,0}}{\partial \rho_i} \cdot \frac{\rho_i^{m_i}}{(1 - \rho_i)^2} \right.\right. \\ &\left.\left. + p_{i,0} \frac{\rho_i^{m_i - 1}(m_i - (m_i - 2)\rho_i)}{(1 - \rho_i)^3}\right)\right) = \phi, \qquad (8)\end{aligned}$$

for all $1 \leq i \leq n$. The above equation looks like

$$\frac{1}{\lambda}\left(T_i + \rho_i \frac{\partial T_i}{\partial \rho_i}\right) = \left(1 - \frac{1}{\alpha}\right)\phi,$$

for all $1 \leq i \leq n$.

8.1.2 The Constant-Speed Model

The following result gives the server speed setting for the constant-speed model.

Theorem 4: For the constant-speed model, we have $s_i = d_i^{1/\alpha}$, where $d_i = \tilde{P}/M - P_i^*$, for all $1 \leq i \leq n$.

Proof. Since a core in server S_i consumes power $s_i^\alpha + P_i^*$, and all cores consume the same amount of power, we have $s_i^\alpha + P_i^* = \tilde{P}/M$, which gives $s_i = \left(\tilde{P}/M - P_i^*\right)^{1/\alpha}$, for all $1 \leq i \leq n$. ∎

By the above theorem, we immediately get the power allocation as

$$P_i = m_i(s_i^\alpha + P_i^*) = \frac{m_i}{M} P_i^*,$$

for all $1 \leq i \leq n$.

Notice that

$$\frac{\partial s_i}{\partial \lambda_i} = 0,$$

and

$$\frac{\partial \rho_i}{\partial \lambda_i} = \frac{\bar{r}}{m_i s_i} - \frac{\lambda_i \bar{r}}{m_i s_i^2} \cdot \frac{\partial s_i}{\partial \lambda_i} = \frac{\rho_i}{\lambda_i}.$$

Hence, we get

$$\begin{aligned}\frac{\partial T_i}{\partial \lambda_i} &= -\frac{\bar{r}}{s_i^2} \cdot \frac{\partial s_i}{\partial \lambda_i} \cdot T_i \frac{s_i}{\bar{r}} \\ &\quad + \frac{m_i^{m_i - 1}}{m_i!} \cdot \frac{\bar{r}}{s_i} \left(\frac{\partial p_{i,0}}{\partial \rho_i} \cdot \frac{\rho_i^{m_i}}{(1 - \rho_i)^2} \right. \\ &\quad \left. + p_{i,0} \frac{\rho_i^{m_i - 1}(m_i - (m_i - 2)\rho_i)}{(1 - \rho_i)^3}\right) \frac{\partial \rho_i}{\partial \lambda_i} \\ &= \frac{m_i^{m_i - 1}}{m_i!} \cdot \frac{\rho_i}{\lambda_i} \cdot \frac{\bar{r}}{s_i} \left(\frac{\partial p_{i,0}}{\partial \rho_i} \cdot \frac{\rho_i^{m_i}}{(1 - \rho_i)^2} \right. \\ &\quad \left. + p_{i,0} \frac{\rho_i^{m_i - 1}(m_i - (m_i - 2)\rho_i)}{(1 - \rho_i)^3}\right),\end{aligned}$$

Consequently, we have

$$\begin{aligned}&\frac{1}{\lambda}\left(T_i + \frac{m_i^{m_i - 1}}{m_i!} \rho_i \frac{\bar{r}}{s_i} \left(\frac{\partial p_{i,0}}{\partial \rho_i} \cdot \frac{\rho_i^{m_i}}{(1 - \rho_i)^2} \right.\right. \\ &\left.\left. + p_{i,0} \frac{\rho_i^{m_i - 1}(m_i - (m_i - 2)\rho_i)}{(1 - \rho_i)^3}\right)\right) = \phi, \qquad (9)\end{aligned}$$

for all $1 \leq i \leq n$. The above equation looks like

$$\frac{1}{\lambda}\left(T_i + \rho_i \frac{\partial T_i}{\partial \rho_i}\right) = \phi,$$

for all $1 \leq i \leq n$.

8.2 The Algorithm

The left-hand sides of Eqs. (8) and (9) are increasing functions of λ_i. Therefore, we can adapt our algorithm in Section 6 to find an optimal equal-power solution.

The Find_λ_i $(m_i, \bar{r}, \lambda, idle, \phi, c_i, d_i)$ algorithm has change in the input parameters, i.e., s and β are replaced by c_i and d_i. For the idle-speed model, since

$$\rho_i = \frac{\lambda_i}{m_i} \bar{r} \frac{\lambda_i^{1/(\alpha-1)}}{c_i} = \frac{1}{c_i} \cdot \frac{\bar{r}}{m_i} \lambda_i^{\alpha/(\alpha-1)} < 1,$$

we need

$$\lambda_i < \left(c_i \frac{m_i}{\bar{r}}\right)^{(\alpha-1)/\alpha},$$

for all $1 \leq i \leq n$. For the constant-speed model, since

$$\rho_i = \frac{\lambda_i}{m_i} \cdot \frac{\bar{r}}{s_i} < 1,$$

we need

$$\lambda_i < \frac{m_i}{\bar{r}} s_i = \frac{m_i}{\bar{r}} d_i^{1/\alpha},$$

for all $1 \leq i \leq n$. Line (3) is therefore changed to

$ub \leftarrow idle \ ? \ (c_i(m_i/\bar{r}))^{(\alpha-1)/\alpha} : (m_i/\bar{r}) d_i^{1/\alpha}$;

According to Theorems 3 and 4, line (8) is changed

$s_i \leftarrow idle \ ? \ c_i/\lambda_i^{1/(\alpha-1)} : d_i^{1/\alpha}$;

According to Eqs. (8) and (9), line (9) is changed

if $(1/\lambda(T_i + \rho_i \partial T_i/\partial \rho_i) < (idle \ ? \ (1 - 1/\alpha) : 1)\phi)$

In the Calculate_T algorithm, the output parameters are changed to ϕ, $\lambda_1, \lambda_2, ..., \lambda_n$, and T. Lines (1) and (2) are changed to

$c_i \leftarrow ((m_i/\bar{r})(\bar{P}/M - P_i^*))^{1/(\alpha-1)}$, for all $1 \leq i \leq n$;

$d_i \leftarrow \bar{P}/M - P_i^*$, for all $1 \leq i \leq n$;

In lines (8), (18), (29), the parameters in calling the Find_λ_i $(m_i, \bar{r}, \lambda, idle, \phi, c_i, d_i)$ algorithm are changed. Line (30) should be identical to line (8) in Find_λ_i.

9 Numerical Examples

In this section, we demonstrate some numerical examples. Notice that all parameters in our examples are for illustration only. They can be changed to any other values in any real clouds.

9.1 The Optimal Method

First, we show two numerical examples of the optimal method.

Example 1. Let us consider a group of $n = 7$ multicore servers $S_1, S_2, ..., S_7$. The server sizes are $m_i = 2i$, where $1 \leq i \leq n$. The base power consumption is $P_i^* = 2$ Watts, for all $1 \leq i \leq n$. The average task execution requirement is $\bar{r} = 1$ (giga instructions). The task arrival rate is $\lambda = 55$ per second. Assume that we are given $\tilde{P} = 200$ Watts. In Table 1, we show the optimal load distribution $\lambda_1, \lambda_2, ..., \lambda_n$, the optimal server speeds $s_1, s_2, ..., s_n$, the optimal power allocation $P_1, P_2, ..., P_n$, the server utilizations $\rho_1, \rho_2, ..., \rho_n$, and the average task response times $T_1, T_2, ..., T_n$ on the n servers, for the idle-speed model. The overall average task response time of the n multicore servers is $T = 0.94682$ seconds.

Example 2. In Table 2, we demonstrate the same data with the same input as in Example 1 for the constant-speed model. It is observed that slightly more load and power are shifted from servers S_1–S_3 to servers S_4–S_7. Due to constant power consumption, server speeds are reduced, which result in increased server utilizations and increased task response times. The overall average task response time of the n multicore servers is $T = 1.20203$ seconds, which is greater than that of the idle-speed model.

Table 1: Numerical Data in Example 1.

i	m_i	λ_i	s_i	P_i	ρ_i	T_i
1	2	1.3999752	1.2649111	6.2399603	0.5533888	1.1395417
2	4	3.4217464	1.2649111	13.4747942	0.6762820	1.0302701
3	6	5.5628638	1.2649111	20.9005821	0.7329717	0.9836460
4	8	7.7653623	1.2649111	28.4245797	0.7673822	0.9564001
5	10	10.0069820	1.2649111	36.0111713	0.7911214	0.9380458
6	12	12.2763491	1.2649111	43.6421585	0.8087755	0.9246230
7	14	14.5667212	1.2649111	51.3067539	0.8225717	0.9142652

Table 2: Numerical Data in Example 2.

i	m_i	λ_i	s_i	P_i	ρ_i	T_i
1	2	1.3313957	1.0212507	6.1302331	0.6518457	1.7026538
2	4	3.3866304	1.1064775	13.4186087	0.7651828	1.4187500
3	6	5.5507005	1.1396518	20.8811207	0.8117539	1.2984137
4	8	7.7718142	1.1583793	28.4349027	0.8386517	1.2279322
5	10	10.0295116	1.1707565	36.0472185	0.8566693	1.1803212
6	12	12.3132388	1.1796966	43.7011820	0.8698026	1.1454166
7	14	14.6167089	1.1865349	51.3867343	0.8799157	1.1184247

9.2 The Equal-Power Method

Now, we show two numerical examples of the equal-power method.

Example 3. Let us consider the same group of multicore servers in Example 1 with the same parameter setting. In Table 3, we show the optimal load distribution $\lambda_1, \lambda_2, ..., \lambda_n$, the optimal server speeds $s_1, s_2, ..., s_n$, the optimal power allocation $P_1, P_2, ..., P_n$, the server utilizations $\rho_1, \rho_2, ..., \rho_n$, and the average task response times $T_1, T_2, ..., T_n$ on the n servers, for the idle-speed model, produced by the equal-power method. It is observed that compared with the optimal solution in Example 1, workloads are

Table 3: Numerical Data in Example 3.

i	m_i	λ_i	s_i	P_i	ρ_i	T_i
1	2	1.6112542	1.3966266	7.1428571	0.5768379	1.0730646
2	4	3.6101020	1.3195249	14.2857143	0.6839776	1.0001381
3	6	5.6846204	1.2878706	21.4285714	0.7356615	0.9703481
4	8	7.7983979	1.2696664	28.5714286	0.7677605	0.9534032
5	10	9.9372898	1.2575155	35.7142857	0.7902320	0.9422039
6	12	12.0940316	1.2486836	42.8571429	0.8071188	0.9341315
7	14	14.2643041	1.2418983	50.0000000	0.8204205	0.9279738

Table 4: Numerical Data in Example 4.

i	m_i	λ_i	s_i	P_i	ρ_i	T_i
1	2	1.5887468	1.1626033	7.1428571	0.6832712	1.6133435
2	4	3.5970867	1.1626033	14.2857143	0.7734983	1.3818819
3	6	5.6794299	1.1626033	21.4285714	0.8141828	1.2820683
4	8	7.7999873	1.1626033	28.5714286	0.8386338	1.2234023
5	10	9.9449398	1.1626033	35.7142857	0.8554027	1.1837327
6	12	12.1072106	1.1626033	42.8571429	0.8678233	1.1546431
7	14	14.2825990	1.1626033	50.0000000	0.8775011	1.1321492

slightly shifted from servers S_5–S_7 to servers S_1–S_4. Furthermore, server speeds are now different. Compared with the optimal solution in Example 1, servers S_5–S_7 have reduced workload, reduced speeds, reduced power consumption, reduced utilization, and increased response time. The overall average task response time of the n multicore servers is $T = 0.94887$ seconds, which is slightly greater than that of the optimal solution.

Example 4. In Table 4, we demonstrate the same data with the same input as in Example 3 for the constant-speed model. Notice that the server speeds happen to be the same, because $P_1^* = P_2^* = \cdots = P_n^*$. Again, compared with the optimal solution in Example 2, servers S_5–S_7 have reduced workload, reduced speeds, reduced power consumption, reduced utilization, and increased response time. The overall average task response time of the n multicore servers is $T = 1.20508$ seconds, which is slightly greater than that of the optimal solution.

10 Performance Comparison

In this section, we compare our optimal solutions with the solutions produced by the equal-power method.

10.1 The Optimal Method

To plot the minimized average task response time T as a function of λ, we need to know the maximum task arrival rate $\lambda_{\max}$. For the idle-speed model, since $\lambda_i < m_i(s_i/\bar{r}) = m_i(s/\bar{r})$, $\lambda_{\max}$ satisfies $\lambda_{\max} < M(s/\bar{r})$, where $M = \sum_{i=1}^{n} m_i$. By Theorem 1, we have

$$s = \left(\frac{1}{\lambda_{\max}\bar{r}}(\tilde{P} - (m_1P_1^* + m_2P_2^* + \cdots + m_nP_n^*))\right)^{1/(\alpha-1)}$$

that is,

$$\lambda_{\max}^{\alpha-1} < \frac{M^{\alpha-1}}{\bar{r}^{\alpha-1}} \cdot \frac{1}{\lambda_{\max}\bar{r}}(\tilde{P} - (m_1P_1^* + m_2P_2^* + \cdots + m_nP_n^*)),$$

or,

$$\lambda_{\max}^{\alpha} < \frac{M^{\alpha-1}}{\bar{r}^{\alpha}}(\tilde{P} - (m_1P_1^* + m_2P_2^* + \cdots + m_nP_n^*)),$$

which implies that

$$\lambda_{\max} < \frac{1}{\bar{r}}M^{(\alpha-1)/\alpha}(\tilde{P} - (m_1P_1^* + m_2P_2^* + \cdots + m_nP_n^*))^{1/\alpha}.$$

For the constant-speed model, by Theorem 2, we have

$$\lambda_i < m_i\frac{s_i}{\bar{r}} = \frac{m_i}{\bar{r}}\left(\frac{\lambda_i}{\beta m_i}\right)^{1/\alpha},$$

that is,

$$\lambda_i < \frac{m_i}{\beta^{1/(\alpha-1)}\bar{r}^{\alpha/(\alpha-1)}},$$

where

$$\beta = \frac{\lambda_{\max}}{\tilde{P} - (m_1P_1^* + m_2P_2^* + \cdots + m_nP_n^*)}.$$

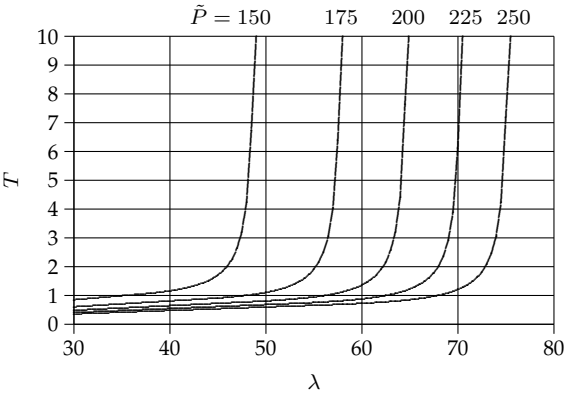

Fig. 4. Average task response time T vs. λ and P (idle-speed model).

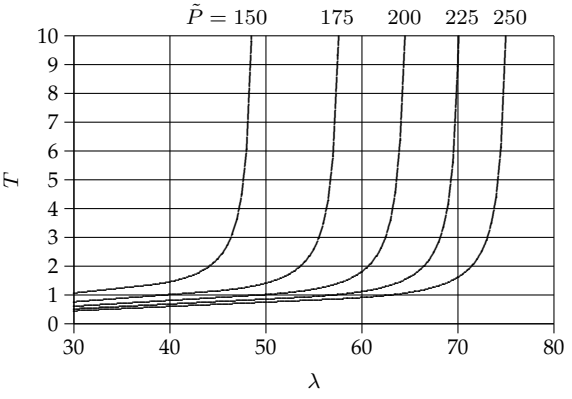

Fig. 5. Average task response time T vs. λ and P (constant-speed model).

Consequently, $\lambda_{\max}$ satisfies

$$\lambda_{\max} < \frac{M}{\beta^{1/(\alpha-1)}\bar{r}^{\alpha/(\alpha-1)}},$$

that is,

$$\lambda_{\max}\beta^{1/(\alpha-1)} < \frac{M}{\bar{r}^{\alpha/(\alpha-1)}},$$

or,

$$\lambda_{\max}^{\alpha-1}\beta < \frac{M^{\alpha-1}}{\bar{r}^{\alpha}},$$

or,

$$\frac{\lambda_{\max}^{\alpha}}{\tilde{P} - (m_1P_1^* + m_2P_2^* + \cdots + m_nP_n^*)} < \frac{M^{\alpha-1}}{\bar{r}^{\alpha}},$$

which implies that

$$\lambda_{\max} < \frac{1}{\bar{r}}M^{(\alpha-1)/\alpha}(\tilde{P} - (m_1P_1^* + m_2P_2^* + \cdots + m_nP_n^*))^{1/\alpha}.$$

Hence, both models have the same $\lambda_{\max}$.

For the same servers in Examples 1 and 2, we display the minimized average task response time T as a function of λ in Figures 4 and 5 for the two core speed models respectively, where $\tilde{P} = 150, 175, 200, 225, 250$ Watts. It is clear that the constant-speed model yields longer average task response time than the idle-speed model.

10.2 The Baseline Method

To plot the average task response time T obtained by the equal-power method as a function of λ, we need to know the maximum task arrival rate $\lambda_{\max}$. For the idle-speed model, by Theorem 3, since $\lambda_i < m_i(s_i/\bar{r}) = (m_i/\bar{r}) \cdot (c_i/\lambda_i^{1/(\alpha-1)})$, we get $\lambda_i < (c_i(m_i/\bar{r}))^{(\alpha-1)/\alpha}$, and $\lambda_{\max}$ satisfies

$$\lambda_{\max} < \sum_{i=1}^{n} \left(c_i \frac{m_i}{\bar{r}}\right)^{(\alpha-1)/\alpha}.$$

For the constant-speed model, by Theorem 4, since

$$\lambda_i < m_i \frac{s_i}{\bar{r}} = \frac{m_i}{\bar{r}} d_i^{1/\alpha},$$

we get

$$\lambda_{\max} < \sum_{i=1}^{n} \frac{m_i}{\bar{r}} d_i^{1/\alpha}.$$

It can be verified that both models have the same $\lambda_{\max}$, i.e.,

$$\lambda_{\max} < \sum_{i=1}^{n} \frac{m_i}{\bar{r}} \left(\frac{\tilde{P}}{M} - P_i^*\right)^{1/\alpha}.$$

Furthermore, we have

$$\sum_{i=1}^{n} \frac{m_i}{\bar{r}} \left(\frac{\tilde{P}}{M} - P_i^*\right)^{1/\alpha} \le \frac{1}{\bar{r}} M^{(\alpha-1)/\alpha} \left(P - \sum_{i=1}^{n} m_i P_i^*\right)^{1/\alpha}$$

where the equality holds only when $P_1^* = P_2^* = \cdots = P_n^*$. To show this, we notice that the inequality is actually

$$\sum_{i=1}^{n} \frac{m_i}{M} \left(\frac{\tilde{P}}{M} - P_i^*\right)^{1/\alpha} \le \left(\frac{\tilde{P}}{M} - \sum_{i=1}^{n} \frac{m_i}{M} P_i^*\right)^{1/\alpha}.$$

Let us consider the function $f(z) = \left(\tilde{P}/M - z\right)^{1/\alpha}$. It is easy to verify that $f(z)$ is a concave function (i.e., $f''(z) < 0$), which implies that

$$\sum_{i=1}^{n} \frac{m_i}{M} f(z_i) \le f\left(\sum_{i=1}^{n} \frac{m_i}{M} z_i\right),$$

where the equality holds only when $z_1 = z_2 = \cdots = z_n$, which is equivalent to $P_1^* = P_2^* = \cdots = P_n^*$, if $z_i = P_i^*$, for all $1 \le i \le n$. The above inequality means that the equal-power method has a smaller saturation point than the optimal solution, and thus results in longer average task response time.

For the same servers in Examples 1–4, we display the average task response time of the equal-power (EP) method and the optimal (OPT) average task response time as a function of λ in Tables 5 and 6 for the idle-speed model and the constant-speed model respectively. The task arrival rate is $\lambda = y\lambda_{\max}$, where $0.80 \le y \le 0.99$, and $\lambda_{\max}$ is the saturation point of the equal-power method. The power constraint is $\tilde{P} = 150, 175, 200, 225, 250$ Watts. It is easily observed that the equal-power method yields noticeably longer average task response time than the optimal average task response time, especially when λ is close to $\lambda_{\max}$ and $\tilde{P}$ is not large.

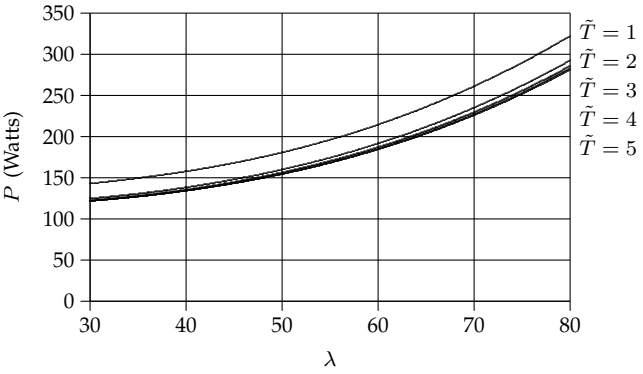

Fig. 6. Average power consumption P vs. λ and $\tilde{T}$ (idle-speed model).

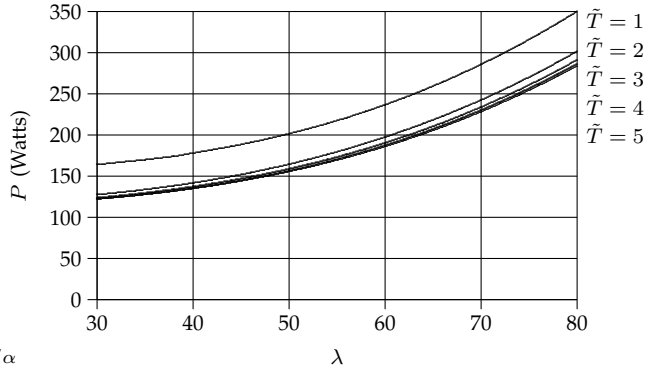

Fig. 7. Average power consumption P vs. λ and $\tilde{T}$ (constant-speed model).

11 Performance Constrained Power Optimization

The *optimal power allocation and load distribution* problem for multiple heterogeneous multicore server processors can also be formulated in a dual form, namely, given the number of multicore servers n, the sizes of the servers $m_1, m_2, ..., m_n$, the base power supply $P_1^*, P_2^*, ..., P_n^*$, the average task execution requirement $\bar{r}$, the task arrival rate λ, and a performance constraint $\tilde{T}$, find the task arrival rates on the servers $\lambda_1, \lambda_2, ..., \lambda_n$, and the server speeds $s_1, s_2, ..., s_n$, such that the average power consumption $P(\lambda_1, \lambda_2, ..., \lambda_n, s_1, s_2, ..., s_n)$ is minimized, subject to the constraints $F(\lambda_1, \lambda_2, ..., \lambda_n) = \lambda$, where $F(\lambda_1, \lambda_2, ..., \lambda_n) = \lambda_1 + \lambda_2 + \cdots + \lambda_n$, and $\rho_i < 1$, i.e., $\lambda_i < m_i/\bar{x}_i = m_i s_i/\bar{r}$, for all $1 \le i \le n$, and $T(\lambda_1, \lambda_2, ..., \lambda_n, s_1, s_2, ..., s_n) = \tilde{T}$. The optimization problem defined above is a *performance constrained power optimization* problem, i.e., minimizing power consumption subject to performance (average task response time) constraint.

Table 5: Comparison of Equal-Power Solutions and Optimal Solutions (Idle-Speed Model).

	$\tilde{P}=150$		$\tilde{P}=175$		$\tilde{P}=200$		$\tilde{P}=225$		$\tilde{P}=250$	
y	EP	OPT	EP	OPT	EP	OPT	EP	OPT	EP	OPT
0.80	1.39038	1.15937	1.05061	0.99560	0.91035	0.88371	0.82457	0.80799	0.76428	0.75252
0.81	1.41988	1.17416	1.07182	1.01352	0.92864	0.90059	0.84110	0.82377	0.77958	0.76735
0.82	1.45231	1.18994	1.09533	1.03318	0.94891	0.91922	0.85942	0.84120	0.79654	0.78377
0.83	1.48825	1.20685	1.12156	1.05490	0.97153	0.93990	0.87988	0.86059	0.81548	0.80205
0.84	1.52847	1.22505	1.15106	1.07904	0.99698	0.96303	0.90289	0.88233	0.83679	0.82256
0.85	1.57389	1.24475	1.18450	1.10608	1.02584	0.98910	0.92897	0.90689	0.86094	0.84576
0.86	1.62569	1.26616	1.22276	1.13658	1.05886	1.01873	0.95883	0.93487	0.88858	0.87222
0.87	1.68542	1.28956	1.26698	1.17129	1.09702	1.05271	0.99333	0.96707	0.92053	0.90272
0.88	1.75513	1.31527	1.31867	1.21117	1.14164	1.09210	1.03367	1.00451	0.95787	0.93824
0.89	1.83759	1.34370	1.37991	1.25747	1.19450	1.13832	1.08146	1.04861	1.00212	0.98015
0.90	1.93669	1.37532	1.45358	1.31188	1.25810	1.19329	1.13895	1.10131	1.05534	1.03035
0.91	2.05804	1.41074	1.54387	1.37675	1.33603	1.25976	1.20940	1.16538	1.12057	1.09154
0.92	2.21004	1.45071	1.65702	1.45537	1.43370	1.34173	1.29771	1.24491	1.20232	1.16775
0.93	2.40589	1.49619	1.80288	1.55262	1.55961	1.44525	1.41153	1.34620	1.30769	1.26519
0.94	2.66757	1.54843	1.99784	1.67592	1.72791	1.58004	1.56367	1.47948	1.44853	1.39410
0.95	3.03466	1.60904	2.27140	1.83723	1.96404	1.76261	1.77713	1.66260	1.64615	1.57246
0.96	3.58627	1.68023	2.68254	2.05715	2.31895	2.02357	2.09797	1.92964	1.94316	1.83522
0.97	4.50705	1.76501	3.36893	2.37436	2.91145	2.42671	2.63358	2.35489	2.43900	2.26025
0.98	6.35086	1.86766	4.74350	2.87114	4.09799	3.13071	3.70619	3.13685	3.43197	3.06371
0.99	11.88700	1.99442	8.87094	3.75885	7.66084	4.66995	6.92695	5.04481	6.41356	5.15204

Table 6: Comparison of Equal-Power Solutions and Optimal Solutions (Constant-Speed Model).

	$\tilde{P}=150$		$\tilde{P}=175$		$\tilde{P}=200$		$\tilde{P}=225$		$\tilde{P}=250$	
y	EP	OPT	EP	OPT	EP	OPT	EP	OPT	EP	OPT
0.80	1.76054	1.45169	1.32357	1.24881	1.14589	1.10918	1.03750	1.01443	0.96141	0.94491
0.81	1.79942	1.47099	1.35228	1.27258	1.17066	1.13186	1.05989	1.03570	0.98214	0.96496
0.82	1.84315	1.49141	1.38462	1.29913	1.19857	1.15730	1.08512	1.05960	1.00550	0.98750
0.83	1.89257	1.51255	1.42123	1.32893	1.23016	1.18596	1.11369	1.08657	1.03194	1.01295
0.84	1.94878	1.53587	1.46292	1.36252	1.26615	1.21843	1.14621	1.11717	1.06206	1.04187
0.85	2.01313	1.56161	1.51070	1.40060	1.30738	1.25544	1.18349	1.15212	1.09657	1.07493
0.86	2.08738	1.59011	1.56588	1.44403	1.35500	1.29791	1.22654	1.19233	1.13642	1.11299
0.87	2.17380	1.62178	1.63014	1.49391	1.41047	1.34705	1.27668	1.23897	1.18284	1.15720
0.88	2.27545	1.65709	1.70577	1.55168	1.47575	1.40442	1.33569	1.29360	1.23747	1.20907
0.89	2.39647	1.69665	1.79586	1.61922	1.55351	1.47214	1.40598	1.35832	1.30254	1.27062
0.90	2.54269	1.74117	1.90474	1.69906	1.64749	1.55312	1.49094	1.43603	1.38119	1.34469
0.91	2.72249	1.79154	2.03867	1.79471	1.76309	1.65144	1.59544	1.53091	1.47793	1.43534
0.92	2.94847	1.84891	2.20703	1.91112	1.90841	1.77312	1.72681	1.64906	1.59954	1.54860
0.93	3.24040	1.91469	2.42458	2.05558	2.09619	1.92723	1.89655	1.79995	1.75668	1.69381
0.94	3.63126	1.99075	2.71588	2.23922	2.34764	2.12835	2.12386	1.99893	1.96711	1.88629
0.95	4.18039	2.07953	3.12519	2.48001	2.70095	2.40123	2.44325	2.27275	2.26277	2.15305
0.96	5.00648	2.18432	3.74100	2.80880	3.23251	2.79179	2.92376	2.67253	2.70761	2.54647
0.97	6.38645	2.30963	4.76976	3.28364	4.12055	3.39571	3.72653	3.30969	3.45076	3.18336
0.98	9.15110	2.46187	6.83089	4.02793	5.89973	4.45099	5.33488	4.48201	4.93967	4.38798
0.99	17.45440	2.65043	13.02142	5.35871	11.24345	6.75923	10.16551	7.34340	9.41159	7.51997

Fortunately, the method we have already developed for power constrained performance optimization is readily applicable to performance constrained power optimization. The key observation is that the average task response time T is a decreasing function of the average power consumption P. Therefore, for a given performance guarantee $\tilde{T}$, we can search P (i.e., the minimized average power consumption) such that $T(\lambda_1, \lambda_2, ..., \lambda_n, s_1, s_2, ..., s_n) = \tilde{T}$ by using the bisection method.

For the same servers in Examples 1 and 2 and Figures 4 and 5, we display the minimized average power consumption P as a function of λ in Figures 6 and 7 for the two core speed models respectively, where $\tilde{T} = 1, 2, 3, 4, 5$ seconds. It is clear that the constant-speed model consumes more average power than the idle-speed model.

12 Concluding Remarks

We have mentioned the importance and significance of performance optimization and power reduction in data centers for cloud computing. We demonstrated the feasibility of studying the power-performance tradeoff for a cloud of clouds with heterogeneous servers in an analytically way. We have described a queueing model for a group of heterogeneous multicore servers with different sizes and speeds. We formulated two optimization problems for optimal power allocation and load distribution on multiple heterogeneous multicore server processors in a cloud computing environment. We proved optimal server speed setting for two different core speed models, i.e., the idle-speed model and the constant-speed model. We also developed our algorithms to solve the optimization problems and demonstrated some numerical examples. Our work makes initial contribution to optimal load distribution with power and performance constraints for multiple queueing systems with multiple servers. Our models, results, and algorithms in this paper are also applicable to other server systems and computing environments.

ACKNOWLEDGMENTS

The authors would like to express their gratitude to the reviewers for their comments on improving the presentation and structure of the paper.

This work is partially supported by Ministry of Science and Technology of China under National 973 Basic Research Grants No. 2011CB302805 and No. 2011CB302505, and National Natural Science Foundation of China Grant No. 61233016.

The work of I. Stojmenovic was supported by the Government of China for the Tsinghua 1000-Plan Distinguished Professorship (2012-2015) and by NSERC Canada Discovery grant.

Part of the work was performed while K. Li was visiting Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, in the winter of 2011 and 2013 and the summer of 2012 as an Intellectual Ventures endowed visiting chair professor (2011-2013).

REFERENCES

[1] http://en.wikipedia.org/wiki/CMOS

[2] http://searchdatacenter.techtarget.com/definition/server-consolidation

[3] S. Albers, "Energy-efficient algorithms," *Communications of the ACM*, vol. 53, no. 5, pp. 86-96, 2010.

[4] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," *Advances in Computers*, vol. 82, pp. 47-111, 2011.

[5] L. Benini, A. Bogliolo, and G. De Micheli, "A survey of design techniques for system-level dynamic power management," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 3, pp. 299-316, 2000.

[6] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, and K. Pentikousis, "Energy-efficient cloud computing," *The Computer Journal*, Advance Access published online on August 19, 2009.

[7] F. Bonomi and A. Kumar, "Adaptive optimal load balancing in a nonhomogeneous multiserver system with a central job scheduler," *IEEE Transactions on Computers*, vol. 39, no. 10, pp. 1232-1250, 1990.

[8] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE Journal on Solid-State Circuits*, vol. 27, no. 4, pp. 473-484, 1992.

[9] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, pp. 103-116, 2011.

[10] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-aware server provisioning and load dispatching for connection-intensive internet services," *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pp. 337-350, 2008.

[11] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, "Managing server energy and operational costs in hosting centers," *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pp. 303-314, 2005.

[12] A. Gandhi, V. Gupta, M. Harchol-Balter, and M. A. Kozuch, "Optimality analysis of energy-performance trade-off for server farm management," *Performance Evaluation*, vol. 67, no. 11, pp. 1155-1171, 2010.

[13] B. Guenter, N. Jain, and C. Williams, "Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning," *Proceedings of INFOCOM*, pp. 1332-1340, 2011.

[14] L. He, S. A. Jarvis, D. P. Spooner, H. Jiang, D. N. Dillenberger, and G. R. Nudd, "Allocating non-real-time and soft real-time jobs in multiclusters, *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 2, pp. 99-112, 2006.

[15] IBM, "The benefits of cloud computing – a new era of responsiveness, effectiveness and efficiency in IT service delivery," *Dynamic Infrastructure*, July 2009.

[16] H. Kameda, J. Li, C. Kim, and Y. Zhang, *Optimal load balancing in distributed computer systems*, Springer-Verlag, London, 1997.

[17] B. Khargharia, S. Hariri, F. Szidarovszky, M. Houri, H. El-Rewini, S. Khan, I. Ahmad, and M. S. Yousif, "Autonomic power and performance management for large-scale data centers," *NFS Next Generation Software Program*, 2007.

[18] L. Kleinrock, *Queueing Systems, Volume 1: Theory*, John Wiley and Sons, New York, 1975.

[19] K. Li, "Minimizing mean response time in heterogeneous multiple computer systems with a central stochastic job dispatcher," *International Journal of Computers and Applications*, vol. 20, no. 1, pp. 32-39, 1998.

[20] K. Li, "Optimizing average job response time via decentralized probabilistic job dispatching in heterogeneous multiple computer systems", *The Computer Journal*, vol. 41, no. 4, pp. 223-230, 1998.

[21] K. Li, "Minimizing the probability of load imbalance in heterogeneous distributed computer systems," *Mathematical and Computer Modelling*, vol. 36, no. 9/10, pp. 1075-1084, 2002.

[22] K. Li, "Optimal load distribution in nondedicated heterogeneous cluster and grid computing environments," *Journal of Systems Architecture*, vol. 54, no. 1-2, pp. 111-123, 2008.

[23] K. Li, "Optimal power allocation among multiple heterogeneous servers in a data center," *Sustainable Computing: Informatics and Systems*, vol. 2, pp. 13-22, 2012.

[24] K. Li, "Optimal load distribution for multiple heterogeneous blade servers in a cloud computing environment," *Journal of Grid Computing*, in press.

[25] C. G. Rommel, "The probability of load balancing success in a homogeneous network," *IEEE Transactions on Software Engineering*, vol. 17, no. 9, pp. 922-933, 1991.

[26] K. W. Ross and D. D. Yao, "Optimal load balancing and scheduling in a distributed computer system," *Journal of the ACM*, vol. 38, no. 3, pp. 676-690, 1991.

[27] B. A. Shirazi, A. R. Hurson, and K. M. Kavi, eds., *Scheduling and Load Balancing in Parallel and Distributed Systems*, IEEE Computer Society Press, Los Alamitos, California, 1995.

[28] M. R. Stan and K. Skadron, "Guest editors' introduction: power-aware computing," *IEEE Computer*, vol. 36, no. 12, pp. 35-38, 2003.

[29] X. Tang and S. T. Chanson, "Optimizing static job scheduling in a network of heterogeneous computers," *Proceedings of International Conference on Parallel Processing*, pp. 373-382, Toronto, Canada, August 2000.

[30] A. N. Tantawi and D. Towsley, "Optimal static load balancing in distributed computer systems," *Journal of the ACM*, vol. 32, no. 2, pp. 445-465, 1985.

[31] O. S. Unsal and I. Koren, "System-level power-aware design techniques in real-time systems," *Proceedings of the IEEE*, vol. 91, no. 7, pp. 1055-1069, 2003.

[32] V. Venkatachalam and M. Franz, "Power reduction techniques for microprocessor systems," *ACM Computing Surveys*, vol. 37, no. 3, pp. 195-237, 2005.

[33] X. Wang, M. Chen, C. Lefurgy, and T. W. Keller, "SHIP: scalable hierarchical power control for large-scale data centers," *Proceedings of the 18th International Conference on Parallel Architectures and Compilation Techniques*, pp. 91-100, 2009.

[34] X. Wang and Y. Wang, "Coordinating power control and performance management for virtualized server clusters," *IEEE Transactions on Parallel and Distributed Systems*, to appear, 2011.

[35] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and practical limits of dynamic voltage scaling," *Proceedings of the 41st Design Automation Conference*, pp. 868-873, 2004.

[36] X. Zheng and Y. Cai, "Achieving energy proportionality in server clusters," *International Journal of Computer Networks*, vol. 1, no. 2, pp. 21-35, 2010.

Junwei Cao received his Ph.D. in computer science from the University of Warwick, Coventry, UK, in 2001. He received his bachelor and master degrees in control theories and engineering in 1996 and 1998, respectively, both from Tsinghua University, Beijing, China. He is currently a Professor and Vice Director, Research Institute of Information Technology, Tsinghua University, Beijing, China. He is also Director of Common Platform and Technology Division, Tsinghua National Laboratory for Information Science and Technology. Before joining Tsinghua University in 2006, he was a research scientist at MIT LIGO Laboratory and NEC Laboratories Europe for about 5 years. He has published over 130 papers and cited by international scholars for over 3,000 times. He is the book editor of *Cyberinfrastructure Technologies and Applications*, published by Nova Science in 2009. His research is focused on advanced computing technologies and applications. Dr. Cao is a senior member of the IEEE Computer Society and a member of the ACM and CCF.

Keqin Li is a SUNY Distinguished Professor of computer science and an Intellectual Ventures endowed visiting chair professor at the National Laboratory for Information Science and Technology, Tsinghua University, Beijing, China. His research interests are mainly in design and analysis of algorithms, parallel and distributed computing, and computer networking. He has contributed extensively to processor allocation and resource management; design and analysis of sequential/parallel, deterministic/probabilistic, and approximation algorithms; parallel and distributed computing systems performance analysis, prediction, and evaluation; job scheduling, task dispatching, and load balancing in heterogeneous distributed systems; dynamic tree embedding and randomized load distribution in static networks; parallel computing using optical interconnections; dynamic location management in wireless communication networks; routing and wavelength assignment in optical networks; energy-efficient computing and communication. Dr. Li has published over 245 journal articles, book chapters, and research papers in refereed international conference proceedings. He has received several Best Paper Awards for his highest quality work. He is currently or has served on the editorial board of *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *Journal of Parallel and Distributed Computing*, *International Journal of Parallel, Emergent and Distributed Systems*, *International Journal of High Performance Computing and Networking*, and *Optimization Letters*.

Ivan Stojmenovic received his Ph.D. degree in mathematics. He is a Full Professor at the University of Ottawa, Canada. He held regular and visiting positions in Serbia, Japan, USA, Canada, France, Mexico, Spain, UK (as Chair in Applied Computing at the University of Birmingham), Hong Kong, Brazil, Taiwan, China, and Australia. He published over 300 different papers, and edited seven books on wireless, ad hoc, sensor and actuator networks and applied algorithms with Wiley. He is an editor of over dozen journals (including *IEEE Network*), the editor-in-chief of *IEEE Transactions on Parallel and Distributed Systems* (2010-13), and the founder and editor-in-chief of three journals (MVLSC, IJPEDS, and AHSWN). Dr. Stojmenovic is one of about 250 computer science researchers with h-index at least 50, has top h-index in Canada for mathematics and statistics, and has more than 12000 citations. He received four best paper awards and the Fast Breaking Paper for October 2003, by Thomson ISI ESI. He is a recipient of the Royal Society Research Merit Award, UK. He is a Tsinghua 1000 Plan Distinguished Professor (2012-15). He is a Fellow of the IEEE (Communications Society, class 2008), and Canadian Academy of Engineering (since 2012). He was an IEEE CS Distinguished Visitor 2010-11 and received 2012 Distinguished Service award from IEEE ComSoc Communications Software TC. He received Excellence in Research Award of the University of Ottawa 2009. Dr. Stojmenovic chaired and/or organized over 60 workshops and conferences, and served in over 200 program committees. He was program co-chair of IEEE PIMRC 2008, IEEE AINA-07, IEEE MASS-04 and 07, founded several workshop series, and is/was Workshop Chair at IEEE ICDCS 2013, IEEE INFOCOM 2011, IEEE MASS-09, ACM Mobihoc-07 and 08.