Optimal Task Dispatching on Multiple Heterogeneous Multiserver Systems with Dynamic Speed and Power Management

Keqin Li, Fellow, IEEE

Abstract—Cloud load balancing is the process of distributing workloads across multiple computing resources in a cloud environment. Load distribution in cloud computing systems is more challenging than in other systems. The purpose of the paper is to address the issue of optimal task dispatching on multiple heterogeneous multiserver systems with dynamic speed and power management. The main contributions of the paper are to solve three problems, i.e., the optimal task dispatching problems with minimized average task response time, minimized average power consumption, and minimized average cost-performance ratio, for multiple heterogeneous multiserver systems with dynamic $d$-speed and power management. In our study, multiserver systems with dynamic speed and power management are modeled as queueing systems, so that fundamental performance and cost metrics such as the average task response time and the average power consumption can be obtained analytically. Our research problems are formulated as multi-variable optimization problems and solved numerically. To the best of our knowledge, this is the first work that addresses load distribution for performance optimization, power minimization, and cost-performance ratio optimization, collectively on multiple heterogeneous servers with dynamic speed and power management.

Index Terms—Cost-performance ratio, dynamic speed and power management, multiserver system, power consumption, queueing model, response time, task dispatching

1 INTRODUCTION

1.1 Motivation

Cloud load balancing is the process of distributing workloads across multiple computing resources in a cloud environment [1]. Load balancing allows enterprises to manage application demands by allocating workload among multiple computers or servers [3]. Load distribution has been a classic research problem in distributed computing, cluster computing, and grid computing [36], and continues to be a fundamental issue in cloud computing, to effectively increase the quality of service to cloud users and to enhance the utilization of resources in cloud systems.

Load distribution in cloud computing systems is more challenging than in other systems due to several reasons. First, energy consumption has become a key issue for the normal operation and maintenance of cloud computing platforms and datacenters, raising serious concerns from cloud providers (see [7], [25], [31] for recent research on green data centers, cloud computing systems, and distributed systems), and load balancing becomes more difficult when reducing energy consumption is also taken into consideration. Second, modern servers deployed in cloud computing have become more and more sophisticated due to the multicore processor architectures, the technique of workload dependent dynamic power management [28], and heterogeneous servers which are different in computing capacity and capability, power consumption model, and dynamic speed and power management scheme. Third, the objective of traditional load distribution is essentially to reduce the average task response time (i.e., to increase the quality of service); however, in cloud computing, there are diversified objectives such as to reduce energy consumption (i.e., to decrease the cost of service) and to optimize the cost-performance ratio.

1.2 Our Contributions

The purpose of the paper is to address the issue of optimal task dispatching on multiple heterogeneous multiserver systems with dynamic speed and power management. The main contributions of the paper are to solve three problems, i.e., the optimal task dispatching problems with minimized average task response time, minimized average power consumption, and minimized average cost-performance ratio, for multiple heterogeneous multiserver systems with dynamic $d$-speed and power management. In our study, multiserver systems with dynamic speed and power management are modeled as queueing systems, so that fundamental performance and cost metrics such as the average task response time and the average power consumption can be obtained analytically (Sections 2, 3, 4, and 5). Our research problems are formulated as multi-variable optimization problems and solved numerically (Sections 6, 7, and 8). To the best of our knowledge, this is the first work that addresses load distribution for performance optimization, power minimization, and cost-performance.
ratio optimization, collectively on multiple heterogeneous multiservers with dynamic speed and power management.

The rest of the paper is organized as follows. In Section 2, we characterize a multiserver system using a queuing model. In Section 3, we describe our server speed and power consumption models. In Section 4, we characterize a dynamic speed and power management scheme using a birth-death process. In Section 5, we consider the class of $d$-speed schemes. In Sections 6, 7, and 8, we define and solve the three optimization problems respectively, present numerical data, and conduct performance comparison. In Section 9, we review related research in cloud load balancing. In Section 10, we conclude the paper.

2 MULTISERVER SYSTEMS

To formulate and study the problem of optimal task dispatching and load distribution for multiple heterogeneous multiserver systems with dynamic speed and power management in a cloud computing environment, we need an analytical model for a multiserver system. A queuing model for a group of $n$ heterogeneous multiserver systems $S_1, S_2, \ldots, S_n$ of sizes $m_1, m_2, \ldots, m_n$ and speeds $s_1, s_2, \ldots, s_n$ will be employed in this paper. Assume that a multiserver system $S_i$ has $m_i$ identical servers with speed $s_i$. Such a multiserver system can be treated as an $M/M/m$ queueing system which is elaborated as follows.

There is a Poisson flow of tasks with arrival rate $\lambda$ (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 task dispatching and load distribution algorithm splits the stream into $n$ substreams, such that the $i$th substream with arrival rate $\lambda_i$ is sent to multiserver system $S_i$, where $1 \leq i \leq n$, and $\lambda = \lambda_1 + \lambda_2 + \cdots + \lambda_n$. A multiserver system maintains a queue with infinite capacity for waiting tasks when all its $m_i$ servers are busy. The first-come-first-served (FCFS) queueing discipline is adopted by all multiserver systems. The task execution requirements (measured by the number of billion instructions to be executed) are i.i.d. exponential random variables $r$ with mean $\bar{r}$. The $m_i$ servers of system $S_i$ have identical execution speed $s_i$ (measured by billion instructions per second (BIPS)). Hence, the task execution times on the servers of system $S_i$ are i.i.d. exponential random variables $x_i = r/s_i$ with mean $\bar{x}_i = \bar{r}/s_i$.

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 server of $S_i$ in one unit of time. The server utilization is

$$\rho_i = \frac{\lambda_i}{m_i \mu_i} = \frac{\lambda_i \bar{r}}{m_i s_i},$$

which is the average percentage of time that a server of $S_i$ is busy. Let $p_{i,k}$ denote the probability that there are $k$ tasks (waiting or being processed) in the $M/M/m$ queueing system for $S_i$. Then, we have (24, p. 102)

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

where

$$p_{i,0} = \left( \sum_{k=0}^{m_i-1} \frac{(m_i \rho_i)^k}{k!} + \frac{(m_i \rho_i)^{m_i}}{m_i!} \cdot \frac{1}{1 - \rho_i} \right)^{-1}.$$
• In the idle-speed model, a server runs at zero speed when there is no task to perform. Since the power for speed $s_i$ is $\xi^k s^{\alpha_i}_i$ and there are $m_i$ servers, the average power consumption of multiserver system $S_i$ is $P_i = m_i(\rho_i s^{\alpha_i}_i + P^*_i) = \lambda_i \cdot \rho_i s^{\alpha_i}_i + m_i P^*_i$.
• In the constant-speed model, a server of $S_i$ still runs at the speed $s_i$ even if there is no task to perform. Hence, the power consumption of multiserver system $S_i$ is $P_i = m_i(\xi s^{\alpha_i}_i + P^*_i)$.

### 4 Dynamic Speed and Power Management

The technique of dynamic speed and power management refers to dynamic server speed and power adjustment according to the current workload (i.e., the number of tasks in a multiserver system). Let the speed of the $m_i$ servers of $S_i$ be $s_{i,k}$ when there are $k$ tasks in the queueing system, where $k \geq 0$. A sequence of server speeds $(s_{i,0}, s_{i,1}, s_{i,2}, s_{i,3}, \ldots)$ is called a speed scheme of $S_i$, which reflects and represents a strategy of workload dependent dynamic speed and power management. If $s_{i,1} = s_{i,2} = s_{i,3} = \cdots = s_{i,\rho}$, then we have a single-speed scheme for workload independent dynamic speed and power management, i.e., a standard M/M/m queueing system. Furthermore, if $s_{i,0} = 0$, we have the idle-speed mode; and if $s_{i,0} = s_i$, we have the constant-speed mode.

A multiserver system $S_i$ with dynamic speed and power management can be characterized by a birth-death process ([24], p. 53). The states are $0, 1, 2, \ldots, k, \ldots$, where state $k$ means that there are $k$ tasks in the multiserver system. The birth rate (i.e., the task arrival rate) is fixed at $\lambda_i$. The death rates (i.e., the task service rates) are $\mu_{i,k}$ with $k \geq 1$. Then, we have

$$\mu_{i,k} = \begin{cases} k \cdot \frac{s_{i,k}}{r}, & 1 \leq k \leq m_i - 1; \\ m_i \cdot \frac{s_{i,k}}{r}, & k \geq m_i. \end{cases}$$

This implies that ([24], p. 92)

$$p_{i,k} = p_{i,0} \frac{\lambda_i^k}{\mu_{i,1} \mu_{i,2} \cdots \mu_{i,k}} = \begin{cases} p_{i,0} \left( \frac{\lambda_i^k}{k!} \cdot \frac{1}{s_{i,1} s_{i,2} \cdots s_{i,k}} \right), & 1 \leq k \leq m_i - 1; \\ p_{i,0} \left( \frac{\lambda_i^k}{k! m_i m_{i-k}^{m_i}} \cdot \frac{1}{s_{i,1} s_{i,2} \cdots s_{i,k}} \right), & k \geq m_i; \end{cases}$$

where

$$p_{i,0} = \left( 1 + \sum_{k=1}^{m_i-1} \frac{(\lambda_i^k)}{k!} \cdot \frac{1}{s_{i,1} s_{i,2} \cdots s_{i,k}} \right)^{-1} + \sum_{k=m_i}^{\infty} \frac{(\lambda_i^k)}{k! m_i m_{i-k}^{m_i}} \cdot \frac{1}{s_{i,1} s_{i,2} \cdots s_{i,k}}.$$

The average number of busy servers in $S_i$ is

$$B_i = \sum_{k=0}^{m_i-1} k p_{i,k} + \sum_{k=m_i}^{\infty} m_i p_{i,k}.$$ 

and the average server utilization of $S_i$ is

$$\rho_i = \frac{B_i}{m_i}.$$ 

The average speed of the $i$th multiserver is

$$s_i = \sum_{k=0}^{\infty} p_{i,k} s_{i,k}.$$ 

The average power consumption of $S_i$ is

$$P_i = \sum_{k=0}^{m_i-1} p_{i,k} (k \cdot s_{i,k}^{\alpha_i} + P^*_i) + \sum_{k=m_i}^{\infty} p_{i,k} m_i (s_{i,k}^{\alpha_i} + P^*_i)$$

for the idle-speed model, and

$$P_i = \sum_{k=0}^{\infty} p_{i,k} m_i (s_{i,k}^{\alpha_i} + P^*_i)$$

for the constant-speed model.

### 5 d-Speed Schemes

A $d_i$-speed scheme of $S_i$ can be represented by

$$\psi_i = (b_{i,1}, b_{i,2}, \ldots, b_{i,d_i-1}; s_{i,1}, s_{i,2}, \ldots, s_{i,d_i}),$$

where $m_i < b_{i,1} < b_{i,2} < \cdots < b_{i,d_i-1},$ and $s_{i,1} < s_{i,2} < \cdots < s_{i,d_i}$. The speed of the $m_i$ servers is $s_{i,1}$ when there are $k \leq b_{i,1}$ tasks, and $s_{i,2}$ when there are $b_{i,1} + 1 \leq k \leq b_{i,2}$ tasks, and so on. Notice that the speed of an idle server is immaterial in this section. Therefore, we have

$$\mu_{i,k} = \begin{cases} k \cdot \frac{s_{i,1}}{r}, & 1 \leq k \leq m_i - 1; \\ m_i \cdot \frac{s_{i,1}}{r}, & m_i \leq k \leq b_{i,1}; \\ m_i \cdot \frac{s_{i,j}}{r}, & b_{i,j-1} + 1 \leq k \leq b_{i,j}; \\ m_i \cdot \frac{s_{i,1}}{r}, & k \geq b_{i,d_i-1} + 1. \end{cases}$$

A speed scheme is valid if it results in a stable queueing system, i.e., $p_{i,0} > 0$. Based on the $p_{i,k}$’s, we get the average number of tasks (in waiting or in execution) in $S_i$ as
This implies that
\[
\begin{align*}
p_{i,k} &= \begin{cases} 
(\lambda \bar{r})^k \left( \frac{1}{s_{i,1}^k} \right), & 1 \leq k \leq m_i - 1; \\
(\lambda \bar{r})^k \left( \frac{1}{m_i!} \right), & m_i \leq k \leq b_{i,1}; \\
(\lambda \bar{r})^k \left( \frac{1}{m_i!} \right), & k \geq b_{i,d_i-1} + 1.
\end{cases}
\end{align*}
\]

Let us define
\[
\rho_{i,j} = \frac{\lambda \bar{r}}{m_i s_{i,j}},
\]
which is the server utilization of \(S_i\) when its server speed is \(s_{i,j}\), where \(1 \leq i \leq n, 1 \leq j \leq d_i\). Then, we obtain
\[
\begin{align*}
p_{i,0} &= \left( 1 + \sum_{k=1}^{m_i-1} \frac{(m_i \rho_{i,1})^k}{k!} \right) + \frac{b_{i,1}}{m_i!} \sum_{k=m_i}^{b_{i,1} - 1} \rho_{i,1}^k \\
p_{i,k} &= \frac{m_i^{m_i}}{m_i!} \sum_{j=1}^{d_i-1} \sum_{k=b_{i,j}}^{b_{i,j+1} - 1} \frac{m_i^{m_i}}{m_i!} \sum_{j=1}^{d_i-1} \left( \prod_{l=1}^{j} \rho_{i,l} \right) \rho_{i,j} \\
k \geq b_{i,d_i-1} + 1.
\end{align*}
\]

In the above equation, we assume that \(b_{i,0} = 0\) for all \(1 \leq i \leq n\).

To continue the evaluation of \(p_{i,0}\), we have
\[
p_{i,0} = \left( 1 + \sum_{k=1}^{m_i-1} \frac{(m_i \rho_{i,1})^k}{k!} \right) + \frac{b_{i,1}}{m_i!} \sum_{k=m_i}^{b_{i,1} - 1} \rho_{i,1}^k \\
+ \frac{m_i^{m_i}}{m_i!} \sum_{j=2}^{d_i-1} \sum_{k=b_{i,j}}^{b_{i,j+1} - 1} \left( \prod_{l=1}^{j} \rho_{i,l} \right) \rho_{i,j} \\
+ \frac{m_i^{m_i}}{m_i!} \sum_{k=b_{i,d_i-1} + 1}^{b_{i,d_i} - 1} \left( \prod_{l=1}^{d_i-1} \rho_{i,l} \right) \rho_{i,d_i}.
\]

A speed scheme is valid if it results in a stable queueing system, i.e., \(p_{i,0} > 0\). It is clear that a \(d_i\)-speed scheme is valid if \(\rho_{i,d_i} < 1\), i.e., \(s_{i,d_i} > \lambda \bar{r}/m_i\).

In the following, we derive closed-form expressions of several major quantities of \(S_i\), i.e., the average task response time \(T_i\), the average server utilization \(\rho_{i}\), the average server speed \(s_{i}\), and the average power consumption \(P_i\). These closed-form expressions are critical to formulate and solve the optimization problems to be addressed in this paper.

Based on the \(p_{i,k}\)'s, we get
\[
\tilde{N}_i = \sum_{k=0}^{\infty} \frac{k}{\rho_{i,k}},
\]
\[
= p_{i,0} \left( \sum_{k=1}^{m_i-1} \frac{(m_i \rho_{i,1})^k}{(k-1)!} + \frac{m_i^{m_i}}{m_i!} \right),
\]
\[
m_i \rho_{i,1} - (m_i - 1) \rho_{i,1}^{b_{i,1}+1} - (b_{i,1} + 1) \rho_{i,1}^{b_{i,1}+2} + b_{i,1} \rho_{i,1}^{b_{i,1}+2}.
\]
A speed scheme is valid if it results in a stable queueing system, i.e., \(p_{i,0} > 0\). It is clear that a \(d_i\)-speed scheme is valid if \(\rho_{i,d_i} < 1\), i.e., \(s_{i,d_i} > \lambda \bar{r}/m_i\).

In the following, we derive closed-form expressions of several major quantities of \(S_i\), i.e., the average task response time \(T_i\), the average server utilization \(\rho_{i}\), the average server speed \(s_{i}\), and the average power consumption \(P_i\). These closed-form expressions are critical to formulate and solve the optimization problems to be addressed in this paper.

Based on the \(p_{i,k}\)'s, we get
\[
\tilde{N}_i = \sum_{k=0}^{\infty} \frac{k}{\rho_{i,k}},
\]
\[
= p_{i,0} \left( \sum_{k=1}^{m_i-1} \frac{(m_i \rho_{i,1})^k}{(k-1)!} + \frac{m_i^{m_i}}{m_i!} \right),
\]
\[
m_i \rho_{i,1} - (m_i - 1) \rho_{i,1}^{b_{i,1}+1} - (b_{i,1} + 1) \rho_{i,1}^{b_{i,1}+2} + b_{i,1} \rho_{i,1}^{b_{i,1}+2}.
\]

A speed scheme is valid if it results in a stable queueing system, i.e., \(p_{i,0} > 0\). It is clear that a \(d_i\)-speed scheme is valid if \(\rho_{i,d_i} < 1\), i.e., \(s_{i,d_i} > \lambda \bar{r}/m_i\).

In the following, we derive closed-form expressions of several major quantities of \(S_i\), i.e., the average task response time \(T_i\), the average server utilization \(\rho_{i}\), the average server speed \(s_{i}\), and the average power consumption \(P_i\). These closed-form expressions are critical to formulate and solve the optimization problems to be addressed in this paper.
Hence, by using the above $\tilde{N}_i$, the average task response time of $S_i$ is

$$T_i = \frac{\tilde{N}_i}{\lambda_i}$$

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

$$+ \frac{m_i}{m_i!} \sum_{j=1}^{d_i-1} \left( \prod_{l=1}^{j-1} \rho_{i,j-l} \right) \left( (b_{i,j-l} + 1)\rho_{i,j} - b_{i,j-l}^2 \right)$$

$$+ \frac{m_i}{m_i!} \sum_{j=1}^{d_i-1} \left( \prod_{l=1}^{j-1} \rho_{i,j-l} \right) \left( b_{i,j-l} + 1 \right)$$

$$+ \frac{b_i^2}{b_i!} \sum_{j=1}^{d_i-1} \frac{1}{\left( 1 - \rho_{i,j} \right)^2} \left( \prod_{l=1}^{j-1} \rho_{i,j-l} \right) \left( b_{i,j-l} + 1 \right)$$

$$+ \frac{b_i^2}{b_i!} \sum_{j=1}^{d_i-1} \frac{1}{\left( 1 - \rho_{i,j} \right)^2} \left( \prod_{l=1}^{j-1} \rho_{i,j-l} \right) \left( b_{i,j-l} + 1 \right),$$

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

The average server utilization of $S_i$ is $\rho_i = B_i/m_i$, where $B_i$ is the average number of busy servers in $S_i$ calculated by

$$B_i = \sum_{k=1}^{m_i-1} k p_{i,k} + \sum_{k=1}^{\infty} m_i p_{i,k}$$

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

The average server speed of $S_i$ is

$$\tilde{s}_i = \sum_{k=0}^{m_i-1} p_{i,k} s_{i,k} + \sum_{k=1}^{\infty} m_i p_{i,k} s_{i,k}$$

$$+ \sum_{j=2}^{d_i-1} \sum_{k=b_{i,j-1}+1}^{\infty} p_{i,k} s_{i,k} d_i$$

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

$$+ \frac{m_i}{m_i!} \sum_{j=2}^{d_i-1} \left( \prod_{l=1}^{j-1} \rho_{i,l} \right) \left( \rho_{i,j} - \rho_{i,j-1}^2 \right) s_{i,j}$$

$$+ \frac{m_i}{m_i!} \sum_{j=2}^{d_i-1} \left( \prod_{l=1}^{j-1} \rho_{i,l} \right) \left( \rho_{i,j} - \rho_{i,j-1}^2 \right) s_{i,j}.$$
6 Minimizing Average Task Response Time

In this section, we formulate and solve our optimal task dispatching problem with minimized average task response time for multiple heterogeneous multiserver systems with dynamic d-speed and power management.

6.1 Problem Definition

Our optimal task dispatching problem with minimized average task response time for multiple heterogeneous multiserver systems can be specified as follows: given the number of multiserver systems 

∀1 ≤ i ≤ n, the power consumption model parameters ξ1, ξ2, ξ3, ..., ξn, the base power consumption P0, P1, ..., Pn, the average task execution requirement r, and the task arrival rate λ, find a load distribution, i.e., the task arrival rates λ1, λ2, ..., λn to the multiserver systems, such that the average task response time T(λ1, λ2, ..., λn) is minimized, subject to the constraint

\[ F(λ1, λ2, ..., λn) = λ, \]

where

\[ F(λ1, λ2, ..., λn) = λ1 + λ2 + ... + λn, \]

and \( ρ_i < 1 \), for all \( 1 ≤ i ≤ n \).

6.2 An Algorithm

The above optimization problem can be solved by using the method of Lagrange multiplier, i.e.,

\[ \nabla T(λ1, λ2, ..., λn) = φ \nabla F(λ1, λ2, ..., λn), \]

that is,

\[ \frac{∂T}{∂λ_i} = φ \frac{∂F}{∂λ_i} = φ, \]

for all \( 1 ≤ i ≤ n \), where φ is a Lagrange multiplier.

As we see below, \( \frac{∂T}{∂λ_i} \) is an extremely complicated function of λi. Hence, an analytical solution is virtually impossible to find. Instead, an algorithm for finding numerical values of λ1, λ2, ..., λn and φ can be developed. The algorithm works as follows. We notice that \( \frac{∂T}{∂λ_i} \) is an increasing function of λi. Therefore, given a φ, we can find λi, 1 ≤ i ≤ n, by the bisection algorithm. The obtained λ1, λ2, ..., λn are used to verify the condition

\[ F(λ1, λ2, ..., λn) = λ, \]

and such verification can be employed to find φ, again by the bisection method.

In the following, we give \( \frac{∂T}{∂λ_i} \). Notice that

\[ \frac{∂ρ_{i,j}}{∂λ_i} = \frac{r}{m_i s_{i,j}}, \]

for all \( 1 ≤ i ≤ n \), and \( 1 ≤ j ≤ d_i \).

Hence, we have

\[
\frac{∂T}{∂λ_i} = \frac{1}{λ} \left( T_i + λ \frac{∂T_i}{∂λ_i} \right),
\]

where

\[
\frac{∂T_i}{∂λ_i} = -\frac{T_i}{λ_i} + \frac{T_i}{λ_i} \frac{∂ρ_{i,0}}{∂λ_i} + \frac{p_{i,0}}{λ_i} \left( \sum_{k=1}^{m_i} \frac{m_i^k}{(k-1)!} \frac{d_{i,k}}{m_i s_{i,k}} \right) \frac{r}{m_i s_{i,1}} + \frac{m_i^m}{m_i!} \left( \sum_{k=1}^{m_i} \frac{m_i^k}{(k-1)!} \frac{d_{i,k}}{m_i s_{i,k}} \right) \frac{r}{m_i s_{i,1}} + \frac{m_i m_i^m}{m_i!} \left( \sum_{k=1}^{m_i} \frac{m_i^k}{(k-1)!} \frac{d_{i,k}}{m_i s_{i,k}} \right) \frac{r}{m_i s_{i,1}} + \frac{b_{i,1}(1-ρ_i)}{(1-ρ_i)^2} \frac{1}{m_i s_{i,1}} + \frac{b_{i,1}(1+2)b_{i,2}}{(1-ρ_i)^2} \frac{1}{m_i s_{i,1}} + 2 \frac{b_{i,1}(1+1)b_{i,2}}{(1-ρ_i)^2} \frac{1}{m_i s_{i,1}}. \]
In our example, we have the same size but different speeds schemes. The sizes are specified in Table 1. We observe that when λ becomes large, S₄ receives less load than S₃, because the ultimate speed of S₄ is higher than that of S₃. Similar situation also exists in the group of heterogeneous multiserver systems with dynamic speed and power management. In particular, we turn each multiserver system Sᵢ into a scheme of speed sᵢ. The speed sᵢ is determined in such a way that the power consumption of Sᵢ is still Pᵢ. Hence, we have

\[ sᵢ = \left( \frac{Pᵢ - mᵢPᵢ^*}{λᵢξᵢ} \right)^{1/αᵢ-1}, \]

for the idle-speed model, and

\[ sᵢ = \left( \frac{1}{ξᵢ} \left( \frac{Pᵢ}{mᵢ} - Pᵢ^* \right) \right)^{1/αᵢ}, \]

for the constant-speed model.

Consider the same group of heterogeneous multiserver systems with static speed and power management. In particular, we turn each multiserver system Sᵢ into a scheme of speed sᵢ. The speed sᵢ is determined in such a way that the power consumption of Sᵢ is still Pᵢ. Hence, we have
TABLE 2
Numerical Data for Response Time Comparison (Idle-Speed Model)

<table>
<thead>
<tr>
<th>i</th>
<th>λ_i</th>
<th>ρ_i</th>
<th>s_i</th>
<th>T_i</th>
<th>(\lambda = 28.71)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.5364377</td>
<td>0.7681015</td>
<td>1.0773778</td>
<td>1.3206634</td>
<td>0.7295129</td>
</tr>
<tr>
<td>2</td>
<td>2.0498104</td>
<td>0.6658866</td>
<td>1.0174135</td>
<td>1.2972947</td>
<td>0.6569453</td>
</tr>
<tr>
<td>3</td>
<td>4.8672405</td>
<td>0.8436211</td>
<td>1.1298270</td>
<td>1.814020</td>
<td>0.8089300</td>
</tr>
<tr>
<td>4</td>
<td>4.4113331</td>
<td>0.8254002</td>
<td>1.0568664</td>
<td>1.2763990</td>
<td>0.8018232</td>
</tr>
<tr>
<td>5</td>
<td>3.5715823</td>
<td>0.7076388</td>
<td>1.0066777</td>
<td>1.2117941</td>
<td>0.7052512</td>
</tr>
<tr>
<td>6</td>
<td>6.0374878</td>
<td>0.8270231</td>
<td>1.0354752</td>
<td>1.2119036</td>
<td>0.8139019</td>
</tr>
<tr>
<td>7</td>
<td>5.2361082</td>
<td>0.7436926</td>
<td>1.0043228</td>
<td>1.1737509</td>
<td>0.7422977</td>
</tr>
</tbody>
</table>

Average 0.7828258 | 1.0479060 | 1.2235758 | 0.7659695 | 1.0823589 | 1.3178383 | 21.5840358 |

\[\lambda = 33.93\]

<table>
<thead>
<tr>
<th>i</th>
<th>λ_i</th>
<th>ρ_i</th>
<th>s_i</th>
<th>T_i</th>
<th>(\lambda = 39.15)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.0375862</td>
<td>0.8608391</td>
<td>1.1516896</td>
<td>1.509202</td>
<td>0.8102051</td>
</tr>
<tr>
<td>2</td>
<td>2.7406338</td>
<td>0.8354326</td>
<td>1.0781120</td>
<td>1.5921013</td>
<td>0.8033635</td>
</tr>
<tr>
<td>3</td>
<td>5.3106865</td>
<td>0.8871277</td>
<td>1.1750096</td>
<td>1.2852018</td>
<td>0.8521317</td>
</tr>
<tr>
<td>4</td>
<td>5.6730410</td>
<td>0.9439792</td>
<td>1.1906290</td>
<td>1.3515626</td>
<td>0.8970483</td>
</tr>
<tr>
<td>5</td>
<td>4.2826612</td>
<td>0.8285640</td>
<td>1.0280283</td>
<td>1.4200655</td>
<td>0.8183943</td>
</tr>
<tr>
<td>6</td>
<td>6.8832824</td>
<td>0.9032609</td>
<td>1.0806526</td>
<td>1.3598737</td>
<td>0.8818255</td>
</tr>
<tr>
<td>7</td>
<td>6.0010889</td>
<td>0.8402473</td>
<td>1.0171540</td>
<td>1.3373252</td>
<td>0.8349485</td>
</tr>
</tbody>
</table>

Average 0.8776920 | 1.1019695 | 1.4118793 | 0.8506748 | 1.1502849 | 1.6537761 | 24.7737396 |

\[\lambda = 44.37\]

<table>
<thead>
<tr>
<th>i</th>
<th>λ_i</th>
<th>ρ_i</th>
<th>s_i</th>
<th>T_i</th>
<th>(\lambda = 49.59)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.6742732</td>
<td>0.9414832</td>
<td>1.2832745</td>
<td>2.0150094</td>
<td>0.8962696</td>
</tr>
<tr>
<td>2</td>
<td>3.8851955</td>
<td>0.9728436</td>
<td>1.3222216</td>
<td>2.3854605</td>
<td>0.9253194</td>
</tr>
<tr>
<td>3</td>
<td>5.9913695</td>
<td>0.9416697</td>
<td>1.2566004</td>
<td>1.6540466</td>
<td>0.9138010</td>
</tr>
<tr>
<td>4</td>
<td>6.8582394</td>
<td>0.9878085</td>
<td>1.3838993</td>
<td>2.0114442</td>
<td>0.9494535</td>
</tr>
<tr>
<td>5</td>
<td>7.7243267</td>
<td>0.9992725</td>
<td>1.5455929</td>
<td>2.6593743</td>
<td>0.9616366</td>
</tr>
<tr>
<td>6</td>
<td>7.9033268</td>
<td>0.9639673</td>
<td>1.1650665</td>
<td>1.7587030</td>
<td>0.9417963</td>
</tr>
<tr>
<td>7</td>
<td>8.3333592</td>
<td>0.9897380</td>
<td>1.2007419</td>
<td>2.2478005</td>
<td>0.9657327</td>
</tr>
</tbody>
</table>

Average 0.9475432 | 1.3073782 | 2.107654 | 0.9419364 | 1.3601353 | 3.0876894 | 36.2025506 |

\[\lambda = 55.99\]

with dynamic speed and power management; (3) \(\rho_i, s_i, T_i\) and \(\rho(\lambda_1, \lambda_2, \ldots, \lambda_i), s(\lambda_1, \lambda_2, \ldots, \lambda_i), T(\lambda_1, \lambda_2, \ldots, \lambda_i)\) with static speed and power management; (4) \(P_i\) and \(P(\lambda_1, \lambda_2, \ldots, \lambda_i)\). It is observed that for the same \(P_i\), the server \(S_i\) with dynamic speed and power management has higher average server utilization, slower average server speed, and shorter average task response time than the server \(S_i\) with static speed and power management. The difference is more noticeable when the server utilization gets higher.

7 Minimizing Average Power Consumption

In this section, we formulate and solve our optimal task dispatching problem with minimized average power consumption for multiple heterogeneous multiserver systems with dynamic \(d\)-speed and power management.

7.1 Problem Definition

Our optimal task dispatching problem with minimized average power consumption for multiple heterogeneous multiserver systems...
systems with dynamic d-speed and power management can be specified as follows: given the number \( n \) of multiserver systems, the sizes of the multiserver systems \( m_1, m_2, \ldots, m_n \), a \( d_i \)-speed scheme \( \psi_i = (b_{i1}, b_{i2}, \ldots, b_{id_i}, s_{i1}, s_{i2}, \ldots, s_{id_i}) \) of \( S_i \), for all \( 1 \leq i \leq n \), the power consumption model parameters \( \xi_1, \alpha_1, \xi_2, \alpha_2, \ldots, \xi_n, \alpha_n \), the base power consumption \( P^*_1, P^*_2, \ldots, P^*_n \), the average task execution requirement \( \bar{r} \), and the task arrival rate \( \lambda \), find a load distribution, i.e., the task arrival rates \( \lambda_1, \lambda_2, \ldots, \lambda_n \) to the multiserver systems, such that the average power consumption \( P(\lambda_1, \lambda_2, \ldots, \lambda_n) \) is minimized, subject to the constraint

\[
F(\lambda_1, \lambda_2, \ldots, \lambda_n) = \lambda,
\]

where

\[
F(\lambda_1, \lambda_2, \ldots, \lambda_n) = \lambda_1 + \lambda_2 + \cdots + \lambda_n,
\]

and \( \rho_i < 1 \), for all \( 1 \leq i \leq n \).
7.2 An Algorithm

The above optimization problem can be solved by using the method of Lagrange multiplier, i.e.,

\[ \nabla P(\lambda_1, \lambda_2, \ldots, \lambda_n) = \phi \nabla F(\lambda_1, \lambda_2, \ldots, \lambda_n), \]

that is,

\[ \frac{\partial P}{\partial \lambda_i} = \phi \frac{\partial F}{\partial \lambda_i} = \phi. \]

for all \( 1 \leq i \leq n \), where \( \phi \) is a Lagrange multiplier.

As we see below, \( \frac{\partial P}{\partial \lambda_i} \) is an extremely complicated function of \( \lambda_i \). Hence, an analytical solution is virtually impossible to find. Instead, an algorithm for finding numerical values of \( \lambda_1, \lambda_2, \ldots, \lambda_n \) and \( \phi \) can be developed. The algorithm works as follows. We notice that \( \frac{\partial P}{\partial \lambda_i} \) is an increasing function of \( \lambda_i \). Therefore, given a \( \phi \), we can find \( \lambda_i \), \( 1 \leq i \leq n \), by the bisection algorithm. The obtained \( \lambda_1, \lambda_2, \ldots, \lambda_n \) are used to verify the condition \( F(\lambda_1, \lambda_2, \ldots, \lambda_n) = \lambda_i \) and such verification can be employed to find \( \phi \), again by the bisection method.

In the following, we give \( \frac{\partial P}{\partial \lambda_i} \). It is clear that

\[ \frac{\partial P}{\partial \lambda_i} = \frac{1}{\lambda} \left( P_i + \lambda_i \frac{\partial P_i}{\partial \lambda_i} \right), \]

where

\[ \frac{\partial P_i}{\partial \lambda_i} = \frac{P_i - m_i P_{in}^i}{\rho_i} \frac{\partial \rho_i}{\partial \lambda_i} + \frac{\xi_i \rho_i}{\lambda_i} \left( \frac{1}{\sum_{k=1}^{m_i} \left( m_k - b_k \frac{m_k^{k-1} - \rho_i}{m_i s_{i,k}} \right) s_{i,k} \right) \frac{1}{\sum_{k=1}^{m_i} \left( m_k - b_k \frac{m_k^{k-1} - \rho_i}{m_i s_{i,k}} \right) s_{i,k}} \]

\[ + \frac{m_i^{m_i+1}}{m_i!} \sum_{j=2}^{m_i+1} \left( \frac{1}{\prod_{l=1}^{j-1} \rho_{i,l}} \frac{\rho_{i,j} - \rho_{i,j-1}}{m_i s_{i,j}} \right) \frac{1}{\rho_{i,j-1}} \]

\[ + \frac{1}{\prod_{l=1}^{j-1} \rho_{i,l}} \frac{1}{\rho_{i,j}} \frac{\rho_{i,j} - \rho_{i,j+1}}{m_i s_{i,j+1}} \left( \frac{1}{\rho_{i,j} - \rho_{i,j+1}} \frac{\rho_{i,j} - \rho_{i,j+1}}{m_i s_{i,j+1}} \right) s_{i,j+1} \]

\[ + \frac{m_i^{m_i+1}}{m_i!} \left( \frac{1}{\prod_{l=1}^{j-1} \rho_{i,l}} \frac{\rho_{i,j} - \rho_{i,j+1}}{m_i s_{i,j}} \right) \frac{1}{\rho_{i,j}} \frac{\rho_{i,j} - \rho_{i,j+1}}{m_i s_{i,j+1}} \left( \frac{1}{\rho_{i,j} - \rho_{i,j+1}} \frac{\rho_{i,j} - \rho_{i,j+1}}{m_i s_{i,j+1}} \right) s_{i,j+1} \]

\[ + \frac{1}{\prod_{l=1}^{j-1} \rho_{i,l}} \frac{1}{\rho_{i,j}} \frac{\rho_{i,j} - \rho_{i,j+1}}{m_i s_{i,j+1}} \left( \frac{1}{\rho_{i,j} - \rho_{i,j+1}} \frac{\rho_{i,j} - \rho_{i,j+1}}{m_i s_{i,j+1}} \right) s_{i,j+1} \]

\[ \left( \frac{1}{\rho_{i,j}} \frac{\rho_{i,j} - \rho_{i,j+1}}{m_i s_{i,j+1}} \right) s_{i,j+1} \]

\[ \left( \frac{1}{\rho_{i,j}} \frac{\rho_{i,j} - \rho_{i,j+1}}{m_i s_{i,j+1}} \right) s_{i,j+1} \]

for all \( 1 \leq i \leq n \).

7.3 Numerical Data

Consider the same group of heterogeneous multiserver systems specified in Section 6.3. In Tables 4 and 5, for the idle-speed model and the constant-speed model respectively, we show the optimal load distribution \( \lambda_1, \lambda_2, \ldots, \lambda_i \), which gives the minimized average power consumption for \( \lambda = (2j - 1)\lambda_{step} \), where \( \lambda_{step} = \lambda_{max}/20 \) and \( j = 1, 2, 3, \ldots, 10 \). We observe that optimal task dispatching with minimized average power consumption is trickier than optimal task dispatching with minimized average task response time due to situations of underflow and overflow. Let us consider

\[ \beta_i = P_i + \lambda_i \frac{\partial P_i}{\partial \lambda_i} \]

where \( 1 \leq i \leq n \). It is required that \( \beta_i = \lambda \phi \) for all \( 1 \leq i \leq n \).

It is clear that \( \beta_i \geq P_i \) and \( \beta_i \geq m_i P_{in}^i \) for the idle-speed model and \( \beta_i \geq m_i, s_{i,\lambda} = P_i \) for the constant-speed model. Hence, if \( \lambda \) is too small, the condition \( \beta_i = \lambda \phi \) may not be satisfied by some multiserver system \( S_i \). In this case, we have to set \( \lambda_i = 0 \), which implies that \( P_i = m_i P_{in}^i \) (for the idle-speed model) or \( P_i = m_i, s_{i,\lambda} = P_i \) (for the constant-speed model). Hence, \( \beta_i = P_i \) (which is greater than \( \lambda \phi \), i.e., underflow), and \( \rho_i = 0 \). For instance, in Table 4, the above situation happens to \( S_6 \) and \( S_7 \) when \( \lambda = 2.61 \). In Table 5, the above situation happens to \( S_3, S_1, S_5, S_6, S_7 \) when \( \lambda = 2.61 \), and to \( S_6 \) and \( S_7 \) when \( \lambda = 7.83 \) and \( \lambda = 13.05 \). Furthermore, it is observed that \( \frac{\partial P_i}{\partial \lambda_i} \) approaches an upper bound \( \pi_i \), which increases by \( \lambda \) increases. For example, for \( S_1, S_5, S_6, S_7 \), \( \pi_1 = 8.4208957 \) for the idle-speed model and \( \pi_1 = 8.1331346 \) for the constant-speed model. Hence, if \( \lambda \) is too large, the condition \( \beta_i = \lambda \phi \) may not be satisfied by some multiserver system \( S_i \). In this case, we can only set \( \lambda_i \) sufficiently close to \( m_i s_{i,\lambda} \) and \( \rho_i \) is sufficiently close to 1, and the \( \beta_i \) is sufficiently close to \( m_i, s_{i,\lambda} = P_i \) (for the constant-speed model). Hence, \( \beta_i = P_i \) (which is greater than \( \lambda \phi \), i.e., overflows). For instance, in Table 4, this value is 64.1440360 for the idle-speed model and 62.8491057 for the constant-speed model. For instances, in Table 4, the above situation happens to \( S_1 \) when \( \lambda = 39.15 \), and to \( S_1, S_2, S_3 \) when \( \lambda = 44.37 \), and to \( S_1, S_2, S_3, S_4, S_5, S_6 \) when \( \lambda = 49.59 \). In Table 5, the above situation happens to \( S_1, S_2, S_3 \) when \( \lambda = 44.37 \), and to \( S_1, S_2, S_3, S_4, S_5, S_6 \) when \( \lambda = 49.59 \).

7.4 Performance Comparison

We compare the cost (i.e., the average power consumption) of a group of heterogeneous multiserver systems with dynamic speed and power management with that of the same group of heterogeneous multiserver systems with static speed and power management. In particular, we turn each multiserver system \( S_i \), with a \( d_i \)-speed scheme into a system with a 1-speed scheme of speed \( s_i \). The speed \( s_i \), which is determined in such a way that the average task response time of \( S_i \) is still \( T_{i} \).

Consider the same group of heterogeneous multiserver systems specified in Section 6.3. For \( \lambda = x \lambda_{max} \), where \( x = 0.35, 0.45, 0.55, 0.65 \), (which are chosen such that both 0.35 underflow and overflow do not happen to any server,) we show in Table 6 and 7 (for the idle-speed model and the constant-speed model respectively): (1) the optimal load distribution \( \lambda_1, \lambda_2, \ldots, \lambda_n \), (2) \( \rho_i, s_i, P_i \), and \( \rho(\lambda_1, \lambda_2, \ldots, \lambda_n) \),
For a multiser ver system \( S_i \) with dynamic speed and power management, our cost measure is mainly \( \sum_i \beta_i T_i \) and \( \mathbf{R}_i \) with static speed and power management. Since the number of servers \( m_i \) is fixed in dynamic speed and power management, our cost measure is mainly the cost of cloud computing.

For a multiser ver system \( S_i \), our performance measure is \( \frac{1}{T_i} \), which is inversely proportional to the average task response time \( T_i \), the higher, the better. There are many different factors which determine the cost of cloud computing. Since the number of servers \( m_i \) is fixed in dynamic speed and power management, our cost measure is mainly the cost of cloud computing \( P_i \), the lower, the better. The cost-performance (or price-performance) ratio (CPR) refers to a product’s ability to deliver performance for its cost.

Generally speaking, products with a lower CPR are more desirable, excluding other factors. In this paper, we define CPR as cost/performance, i.e., \( R_i = P_i T_i \). The average cost-performance ratio \( R \) of a group of \( n \) heterogeneous multiser ver systems \( S_1, S_2, \ldots, S_n \) is

\[
R(\lambda_1, \lambda_2, \ldots, \lambda_n) = \frac{\lambda_1}{\alpha} P_1 T_1 + \frac{\lambda_2}{\alpha} P_2 T_2 + \cdots + \frac{\lambda_n}{\alpha} P_n T_n,
\]

where \( \alpha \) is the total utilization of the system.
for heterogeneous workloads on enterprise grids and clouds, and reviewed load balancing strategies for cloud infrastructures. In [20], the author surveyed various dynamic load balancing algorithms in cloud with discussion and comparison of the pros and cons of these algorithms. In [21], the authors presented a comparative study of various load balancing schemes in different cloud environments based on requirements specified in service level agreement. In [22], the authors gave an overview of many load balancing algorithms which help to achieve better throughput and improve the response time in cloud environments. In [23], the authors gave an overview of load balancing in cloud computing by exposing the most important research challenges. In [30], the authors investigated the different algorithms proposed to resolve the issue of load balancing and task scheduling in cloud computing, and discussed and compared these algorithms to provide an overview of the latest approaches in the

<table>
<thead>
<tr>
<th>( \lambda )</th>
<th>Parameter</th>
<th>( i = 1 )</th>
<th>( i = 2 )</th>
<th>( i = 3 )</th>
<th>( i = 4 )</th>
<th>( i = 5 )</th>
<th>( i = 6 )</th>
<th>( i = 7 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.61</td>
<td>( \lambda_i )</td>
<td>1.1728906</td>
<td>1.4371094</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>13.05</td>
<td>( \beta_i )</td>
<td>12.3837221</td>
<td>12.3837221</td>
<td>20.0000000</td>
<td>20.0000000</td>
<td>20.0000000</td>
<td>28.0000000</td>
<td>28.0000000</td>
</tr>
<tr>
<td>18.27</td>
<td>( \rho_i )</td>
<td>0.3887616</td>
<td>0.4782331</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>33.93</td>
<td>( \beta_i )</td>
<td>0.8290179</td>
<td>0.8670383</td>
<td>0.6608810</td>
<td>0.7475620</td>
<td>0.8124483</td>
<td>0.3899562</td>
<td>0.4959042</td>
</tr>
<tr>
<td>39.15</td>
<td>( \rho_i )</td>
<td>0.8528755</td>
<td>0.8684634</td>
<td>0.6604304</td>
<td>0.7472092</td>
<td>0.8121575</td>
<td>0.0518330</td>
<td>0.9968196</td>
</tr>
<tr>
<td>49.59</td>
<td>( P_i )</td>
<td>30.5138652</td>
<td>18.2119946</td>
<td>25.1021726</td>
<td>23.7307516</td>
<td>28.0000000</td>
<td>29.3596132</td>
<td>28.3094296</td>
</tr>
<tr>
<td>( \beta_i )</td>
<td>31.5452797</td>
<td>31.5452797</td>
<td>31.5452797</td>
<td>31.5452797</td>
<td>31.5452797</td>
<td>31.5452797</td>
<td>31.5452797</td>
<td></td>
</tr>
<tr>
<td>( \rho_i )</td>
<td>0.3960005</td>
<td>0.9435795</td>
<td>0.8285024</td>
<td>0.9639069</td>
<td>0.9089080</td>
<td>0.8292750</td>
<td>0.8768217</td>
<td></td>
</tr>
<tr>
<td>621</td>
<td>( d )</td>
<td>4.0988190</td>
<td>4.0575231</td>
<td>5.7585727</td>
<td>5.4359878</td>
<td>5.4333716</td>
<td>7.0119530</td>
<td>7.0437279</td>
</tr>
<tr>
<td>621</td>
<td></td>
<td>52.5138652</td>
<td>22.9452928</td>
<td>29.9036852</td>
<td>27.1574566</td>
<td>25.7737144</td>
<td>32.9683220</td>
<td>32.681024</td>
</tr>
<tr>
<td>621</td>
<td>( P_i )</td>
<td>60.8379086</td>
<td>60.8379086</td>
<td>60.8379086</td>
<td>60.8379086</td>
<td>60.8379086</td>
<td>60.8379086</td>
<td>60.8379086</td>
</tr>
<tr>
<td>( \beta_i )</td>
<td>0.9954356</td>
<td>0.9813833</td>
<td>0.9245658</td>
<td>0.9284289</td>
<td>0.9534291</td>
<td>0.9127652</td>
<td>0.9377844</td>
<td></td>
</tr>
<tr>
<td>( \rho_i )</td>
<td>0.0000000</td>
<td>1.0000000</td>
<td>1.0000000</td>
<td>1.0000000</td>
<td>1.0000000</td>
<td>1.0000000</td>
<td>1.0000000</td>
<td></td>
</tr>
</tbody>
</table>

where \( R \) is treated as a function of load distribution \( \lambda_1, \lambda_2, \ldots, \lambda_n \).

In this section, we formulate and solve our optimal task dispatching problem with minimized average cost performance ratio for multiple heterogeneous multiserver systems with dynamic \( d \)-speed and power management.

(Due to space limitation, the remaining content of this section is moved to the supplementary file, online available.)

9 RELATED RESEARCH

There have been extensive research in cloud load balancing and load distribution. Several surveys and comparative studies have been conducted. In [4], existing load balancing techniques in cloud computing were discussed and compared based on various parameters. In [16], the authors explored autonomic approaches for optimizing provisioning for heterogeneous workloads on enterprise grids and clouds, and reviewed load balancing strategies for cloud infrastructures. In [20], the author surveyed various dynamic load balancing algorithms in cloud with discussion and comparison of the pros and cons of these algorithms. In [21], the authors presented a comparative study of various load balancing schemes in different cloud environments based on requirements specified in service level agreement. In [22], the authors gave an overview of many load balancing algorithms which help to achieve better throughput and improve the response time in cloud environments. In [23], the authors gave an overview of load balancing in cloud computing by exposing the most important research challenges. In [30], the authors investigated the different algorithms proposed to resolve the issue of load balancing and task scheduling in cloud computing, and discussed and compared these algorithms to provide an overview of the latest approaches in the
field. In [33], the authors provided a comprehensive review on the existing load balancing strategies and presented load balancer as a service model adopted by the major market players. In [35], the authors presented a survey of dynamic load balancing strategies on cloud, with focus on various metrics to analyze the efficacy of existing techniques. In [37], the authors provided a comprehensive review of load balancing strategies on cloud, with focus on various load balancing algorithms were compared on the basis of their metrics.

Numerous researchers have investigated various approaches to cloud load balancing. In [5], the authors showed a new approach to dynamic load balancing using the concept of mobile agent, i.e., a software program which executes independently and performs the basic task. In [10], the authors proposed a novel load balancing strategy using a genetic algorithm, which thrives to balance the load of a cloud infrastructure while trying to minimize the makespan of a given task set. In [11], the authors proposed an algorithm named honey bee behavior inspired load balancing, which aims to achieve well balanced load across virtual machines for maximizing the throughput and minimizing the amount of waiting time of the tasks. In [12], the authors proposed a novel approach to dynamic load balancing in cloud computing systems based on the phenomena of self-organization in a game theoretical spatially generalized prisoner’s dilemma model defined on the two-dimensional cellular automata space. In [14], the authors focused on two load balancing algorithms defined on the two-dimensional cellular automata space. In [15], the authors used an agent-based dynamic load balancing approach which greatly reduces the communication cost of servers, accelerates the rate of load balancing, and improves the throughput and response time of the cloud. In [27], the author studied the problem of optimal distribution of generic tasks over a group of heterogeneous blade servers in a cloud computing environment or a data center, such that the average response time of generic tasks is minimized. In [34], the authors introduced a

| Table 6: Numerical Data for Power Consumption Comparison (Idle-Speed Model) |
|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
| $i$ | $\lambda_i$ | $\rho_i$ | $s_i$ | $P_i$ | $T_i$ |
| 1 | 2.4242222 | 0.7434831 | 1.0645909 | 12.3017405 | | 0.6894322 | 1.1721016 | 12.6609003 | 1.2898882 |

**Dynamic Management**

| $i$ | $\lambda_i$ | $\rho_i$ | $s_i$ | $P_i$ | $T_i$ |
| 1 | 2.5444642 | 0.7972133 | 1.0544334 | 12.1862428 | | 0.7416644 | 1.1435826 | 12.652086 | 1.5005392 |

**Static Management**

| $i$ | $\lambda_i$ | $\rho_i$ | $s_i$ | $P_i$ | $T_i$ |
| 1 | 2.5444642 | 0.7972133 | 1.0544334 | 12.1862428 | | 0.7416644 | 1.1435826 | 12.652086 | 1.5005392 |
threshold based dynamic compare and balance algorithm for cloud server optimization, which also minimizes the number of host machines to be powered on for reducing the cost of cloud services. In [38], the authors proposed an autonomous agent-based load balancing algorithm, which provides dynamic load balancing for cloud environment. In [39], an enhanced shortest job first scheduling algorithm was used to achieve reduced response time and reduced starvation and job rejection rate. In [41], the authors developed an approach from machine learning to learn task arrival and execution patterns online, i.e., automatically acquiring such knowledge without any beforehand modeling and proactively allocating tasks on account of the forthcoming tasks and their execution dynamics. In [42], the authors studied the collaboration among benevolent clouds that are cooperative in nature and willing to accept jobs from other clouds, and took advantage of machine learning, and proposed a distributed scheduling mechanism to learn the knowledge of job model, resource performance, and others’ policies. In [43], the authors proposed a fairness-aware load balancing algorithm, where the load balancing problem is defined as a game, and the Nash equilibrium solution for this problem minimizes the expected response time, while maintaining fairness.

Cloud load distribution has been considered together with energy consumption. In [6], the authors conducted a survey of research in energy-efficient computing and proposed architectural principles for energy-efficient management of clouds and energy-efficient resource allocation policies and scheduling algorithms considering QoS expectations and power usage characteristics of the devices. In [8], the authors addressed optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers as optimization problems. In [13], the authors proposed a new power-aware load balancing algorithm based on artificial bee colony to detect both over-utilized and under-utilized hosts for 678

679

680

681

682

683

684

685

686

687
effective power management. In [17], the authors studied the problem of power consumption minimization with performance constraint in heterogeneous distributed embedded systems by optimal load distribution. In [19], the authors discussed existing load balancing techniques in cloud computing and further compared them based on various parameters and discussed these techniques from energy consumption and carbon emission perspective. In [26], the author considered the problem of optimal power allocation among multiple heterogeneous servers, i.e., minimizing the average task response time of multiple heterogeneous computer systems with energy constraint. In [29], the authors modeled a data center as a cyber physical system to capture the thermal properties exhibited by the data center, where software aspects such as scheduling, load balancing, and computation are the cyber component, and hardware aspects such as servers and switches are the physical component. In [32], the authors investigated load distribution strategies to minimize electricity cost and increase renewable energy integration subject to compliance with service level agreement, with consideration of the adverse effects of switching the servers. In [40], the authors investigated performance and power tradeoff for multiple heterogeneous servers by considering two problems, i.e., optimal job scheduling with fixed service rates and joint optimal service speed scaling and job scheduling. In [44], the authors employed a game theoretic approach to solving the problem of minimizing energy consumption as a Stackelberg game, and modeled the problem of minimizing average task response time as a noncooperative game among decentralized scheduler agents as they compete with one another in the shared resources.

## 10 Conclusion

We have formulated and solved three optimization problems, i.e., the optimal task dispatching problems with minimized average task response time, minimized average power consumption, and minimized average cost-performance ratio, on multiple heterogeneous multiserver systems with dynamic d-speed and power management. We have also demonstrated numerical data and conducted performance comparison between dynamic management and static management of speed and power.

In this paper, each server has a known speed scheme. As a further research direction, the optimal task dispatching problem in this paper can be extended to the optimal task dispatching problem in which the speed scheme of a server is also to be determined in such a way that the overall power consumption of the multiserver system does not exceed a given power budget. This is an extremely difficult problem, since the choice of a speed scheme can be arbitrarily complicated. Even though we only consider a d-speed scheme, it still has 2d − 1 parameters in \( \psi_i \). By including the task arrival rate \( \lambda_i \), each multiserver system has 2d parameters to determine, and our optimization problem has 2nd parameters to determine. When \( d = 2 \), we still have 4n parameters. It is conceivable that the optimization problem is very sophisticated. However, we would like to mention that when \( d = 1 \), i.e., for single-speed multiserver systems, the optimal load distribution and power allocation (i.e., speed determination) problem has been solved [8].

## References


