Optimal Speed Setting for Cloud Servers With Mixed Applications

Keqin Li, Fellow, IEEE

Abstract—The technique of workload dependent dynamic power management can dynamically and flexibly adjust power and speed according to the current workload. It has been well recognized that improving server performance and reducing energy consumption can be achieved by employing the technique of workload dependent dynamic power management. It is an effective way to deal with the power and performance tradeoff for cloud servers. In this study, applications are divided into different classes, which have different characteristics. The server speed is different in processing tasks from different types. Hence, we explore the technique of variable and task type dependent server speed management to optimize the server performance and minimize the power consumption of a server with mixed applications. This is also a kind of workload-dependent dynamic power and speed management to deal with the power and performance tradeoff. We establish an M/G/1 queueing model for a server with variable and task type dependent speed, so that our investigation can be conducted analytically. We formulate the problems of power constrained performance optimization and performance constrained power minimization as multivariable optimization problems, and solve the problems by efficient numerical algorithms. We provide numerical data to compare the performance of a server with the optimal speed setting to that of a server with a constant speed, and to compare the power of a server with the optimal speed setting to that of a server with a constant speed. It is shown that the reduction in the average response time can be as high as 9.9% and the reduction in the average power consumption can be as high as 8.0%.

Index Terms—Average response time, cloud server, mixed applications, optimal speed setting, power consumption, workload-dependent dynamic power management.

I. INTRODUCTION

A. Motivation

The technique of workload-dependent dynamic power management can dynamically and flexibly adjust power and speed according to the current workload, i.e., the number of applications in a server and the characteristics of the applications. When there are more tasks in a server, we can increase the power supply and the server speed to reduce the average response time without significant energy increment. On the other hand, when there are less tasks in a server, we can decrease the power supply and the server speed to reduce the average power consumption without significant performance degradation. Dynamic power and speed adjustment can also be performed when there is substantial change in application characteristics. Such runtime power and speed adjustment can be implemented by the mechanisms of dynamic voltage scaling, dynamic frequency scaling, dynamic speed scaling, and dynamic power scaling [1], [11], [12].

A number of researchers have studied workload-dependent dynamic power management. Typically, the lowest server speed should be chosen for a group of applications, so that the group of applications can be processed with certain required performance constraints [15]. We can carry out dynamic power management with different granularity, i.e., the application level and the phase (of an application) level. At the application (phase, respectively) level, we analyze the overall characteristics of an application (phase, respectively) and determine the server speed based on these properties. For instances, the server speed should be high for CPU-bound applications (phase, respectively) to reduce the execution time; however, the server speed should be low for memory-bound applications (phase, respectively) to save energy without increasing the execution time [3], [20]. Cochran et al. [5] presented an accurate and scalable method that determines the optimal system operating points (i.e., number of threads and dynamic voltage and frequency settings) and optimizes energy efficiency in multicore processors at runtime for parallel workloads with a set of objective functions and constraints. Huang and Feng [9] presented an eco-friendly daemon that reduces energy consumption while maintaining high performance via accurate workload characterization. As an interval-based run-time algorithm, the eco-friendly daemon uses workload characterization to dynamically adjust a processor’s voltage and frequency and to reduce energy consumption with little impact on application performance.

It has been well recognized that improving server performance and reducing energy consumption can be achieved by employing the technique of workload-dependent dynamic power management. It is an effective way to deal with the power and performance tradeoff for cloud servers. Furthermore, analytical studies can be performed for workload-dependent dynamic power management. In [16], we established a queueing model of multicore server processors with the capability of workload-dependent dynamic power management. We proposed several speed schemes and demonstrated that for the same...
average power consumption, the average task response time of a multicore server processor with workload-dependent dynamic power management is shorter than that of a multicore server processor with constant speed (i.e., without workload-dependent dynamic power management). We showed that for certain application environment and average power consumption, there is an optimal speed scheme that minimizes the average task response time. We also pointed out that power reduction subject to performance constraints can be studied in a way similar to performance improvement subject to power constraints.

B. Our Contributions

In this paper, we adopt a different approach from [16], where workload is measured in terms of the number of tasks in a server. The server speed increases (decreases, respectively) when the number of tasks increases (decreases, respectively). In this study, applications are divided into different classes, which have different characteristics. The server speed is different in processing tasks from different types. Hence, we explore the technique of variable and task type dependent server speed management to optimize the server performance and to minimize the power consumption of a server with mixed applications. This is also a kind of workload-dependent dynamic power and speed management to deal with the power and performance tradeoff.

Our main contributions can be summarized as follows.

1) We establish an M/G/1 queueing model for a server with variable and task type dependent speed, so that our investigation can be conducted analytically.

2) We formulate the problems of power constrained performance optimization and performance constrained power minimization as multivariable optimization problems, and solve the problems by efficient numerical algorithms.

3) We provide numerical data to compare the performance of a server with the optimal speed setting to that of a server with a constant speed, and to compare the power of a server with the optimal speed setting to that of a server with a constant speed. It is shown that the reduction in the average response time can be as high as 9.9% and the reduction in the average power consumption can be as high as 8.0%.

To the author’s best knowledge, this is the first work, which analytically studies power and performance optimization using the technique of variable and task type dependent server speed management for a server with mixed applications.

The organization of this paper is as follows. In Section II, we review related research. In Section III, we present the queueing model and the power consumption model. In Section IV, we formulate and solve the problem of power constrained performance optimization, demonstrate numerical data, and conduct performance comparison. In Section V, we formulate and solve the problem of performance constrained power minimization. We conclude the paper in Section VI.

II. RELATED RESEARCH

As one of the fundamental properties of cloud computing, elasticity is the capability to scale computing resources up and down dynamically with minimal friction. It has been recognized that elasticity will eventually manifest all of the benefits of the cloud [22]. Autoscaling means scaling a multiserver to match changing workload without any human intervention. There are two types of autoscaling schemes for elastic and scalable multiserver management, which are defined as follows [10].

1) Scale-out and scale-in autoscaling schemes—This is also called workload-dependent dynamic multiserver size management. When the workload fluctuates, the number of servers (i.e., the size of a multiserver system) can be dynamically changed to provide the required performance and cost objectives. These schemes are also called auto size scaling schemes.

2) Scale-up and scale-down autoscaling schemes—This is also called workload-dependent dynamic multiserver speed management. When the workload fluctuates, the speed of servers (i.e., the speed of a multiserver system) can be dynamically changed to provide the required performance and cost objectives. These schemes are also called auto speed scaling schemes.

Essentially, there are two types of cloud resource scaling in an elastic cloud computing system, i.e., horizontal scalability and vertical scalability [8]. Horizontal scaling (i.e., scaling out and scaling in) means allocation and releasing of homogeneous virtual machines or processing nodes of the same type. Vertical scaling (i.e., scaling up and scaling down) means upgrade or downgrade of the capability (core speed, memory capacity, network bandwidth, etc.) of a server.

Cloud elasticity has also been studied from wider perspectives. Dustdar et al. considered elasticity properties such as cost elasticity (i.e., the responsiveness of resource provision to changes in cost) and quality elasticity (i.e., the responsiveness of quality to changes in resource usage) [6]. Galante and de Bona classified elastic systems in terms of four characteristics, i.e., scope (infrastructure, application, platform), policy (manual, reactive, predictive), purpose (performance, capacity, cost, energy), and method (replication, resizing, migration) [7]. Kuperberg et al. mentioned two kinds of scalability, i.e., application scalability (i.e., the ability of an application to maintain its performance goals and service-level agreement even when its workload increases) and platform scalability (i.e., the ability of a cloud platform to provide as many resources as needed by an application) [14]. Sobeslavsky considered application elasticity, i.e., making an application to be able to adjust to variations in load without the need of intervention of a human administrator and changing its code [21].

Analytical study of cloud elasticity has recently been conducted for both horizontal scalability and vertical scalability. In [16], by using a queueing model, we investigated the technique of workload-dependent dynamic power management (i.e., dynamic power and speed adjustment according to the current workload, which is essentially vertical scalability), so that the system performance can be improved and energy consumption can be reduced. We also studied the auto speed scaling scheme optimization problem to minimize the cost–performance ratio. In [17], we addressed the issue of optimal task dispatching on multiple heterogeneous multiserver systems.
TABLE I
NOTATIONS AND DEFINITIONS

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>the number of types of applications</td>
</tr>
<tr>
<td>(\lambda_i)</td>
<td>the task arrival rate of the (i)th type of applications</td>
</tr>
<tr>
<td>(\lambda)</td>
<td>the total task arrival rate</td>
</tr>
<tr>
<td>(r_i)</td>
<td>the execution requirements of the tasks of the (i)th type of applications</td>
</tr>
<tr>
<td>(s_i)</td>
<td>the execution speed of the server for the (i)th type of applications</td>
</tr>
<tr>
<td>(x_i)</td>
<td>the execution times of the tasks of the (i)th type of applications</td>
</tr>
<tr>
<td>(x)</td>
<td>the execution time of a task of all applications</td>
</tr>
<tr>
<td>(\rho)</td>
<td>the utilization of the server</td>
</tr>
<tr>
<td>(\bar{x}, \bar{x}^2)</td>
<td>the mean and the second moment of (x)</td>
</tr>
<tr>
<td>(W)</td>
<td>the average waiting time of a task</td>
</tr>
<tr>
<td>(\sigma)</td>
<td>(\lambda_1 \bar{x}_1^2 + \lambda_2 \bar{x}_2^2 + \cdots + \lambda_n \bar{x}_n^2)</td>
</tr>
<tr>
<td>(T_i)</td>
<td>the average response time of tasks of the (i)th type of applications</td>
</tr>
<tr>
<td>(\overline{x_i}, \overline{x_i}^2)</td>
<td>the mean and the second moment of (x_i)</td>
</tr>
<tr>
<td>(T)</td>
<td>the average task response time of all tasks</td>
</tr>
<tr>
<td>(P^*)</td>
<td>base power consumption of the server</td>
</tr>
<tr>
<td>(\alpha)</td>
<td>exponent of the power consumption model</td>
</tr>
<tr>
<td>(P)</td>
<td>the average power consumption of the server</td>
</tr>
<tr>
<td>(\bar{P})</td>
<td>power constraint</td>
</tr>
<tr>
<td>(\phi)</td>
<td>a Lagrange multiplier</td>
</tr>
<tr>
<td>(F_i)</td>
<td>a non-linear system of (n + 1) equations</td>
</tr>
<tr>
<td>(y)</td>
<td>((y_0, y_1, \ldots, y_n) = (\phi, s_1, \ldots, s_n))</td>
</tr>
<tr>
<td>(F(y))</td>
<td>((F_0(y), F_1(y), \ldots, F_n(y)))</td>
</tr>
<tr>
<td>(J(y))</td>
<td>Jacobian matrix with (J(y)_{i,j} = \partial F_i(y)/\partial y_j, 0 \leq i, j \leq n)</td>
</tr>
<tr>
<td>(T)</td>
<td>time constraint</td>
</tr>
</tbody>
</table>

with dynamic speed and power management by solving three problems, i.e., optimal task dispatching to minimize average task response time, average power consumption, and average cost–performance ratio, respectively. In [18], we presented a new and quantitative definition of elasticity in cloud computing, developed an analytical model by treating a cloud platform with horizontal scalability as a queueing system, and used a continuous-time Markov chain model to rigorously calculate the elasticity value of a cloud platform by using an analytical and numerical method.

III. MODEL

The reader is referred to Table I for a list of the notations and definitions used in this paper.

In this paper, we use \(\overline{y}\) to represent the expectation of a random variable \(y\) (e.g., \(y\) can be \(x, r_i, \ldots\)).

We consider a server with variable execution speed, which is a continuous variable. The server can be treated accurately as an M/G/1 server using Kendall’s notation. Such a server uses the first-come-first-serve (FCFS) scheduling method and allows task interarrival times to follow an exponential distribution and task execution times to follow an arbitrary probability distribution (a fairly general model without extra assumptions).

There are \(n\) types of applications. (Notice that we use the words “tasks” and “applications” interchangeably.) Assume that the task arrival rate (measured by the number of arrival tasks per second) of the \(i\)th type of applications is \(\lambda_i\), where \(1 \leq i \leq n\). The total task arrival rate is \(\lambda = \lambda_1 + \lambda_2 + \cdots + \lambda_n\).

For the \(i\)th type of applications, the execution requirements (measured by the number of billion instructions to be executed) of the tasks are independent and identically distributed (i.i.d.) random variables \(r_i\). The execution speed (measured by the number of billion instructions that can be executed in one second) of the server for the \(i\)th type of applications is \(s_i\), which is to be determined by an optimizing algorithm in Section IV-A or V-A. Hence, the execution times (measured by seconds) of the tasks of the \(i\)th type of applications are i.i.d. random variables \(x_i = r_i/s_i\).

The execution time of a task is a random variable \(x\) with mean

\[
\overline{x} = \frac{\lambda_1 \overline{x}_1}{\lambda} + \frac{\lambda_2 \overline{x}_2}{\lambda} + \cdots + \frac{\lambda_n \overline{x}_n}{\lambda}.
\]

The utilization of the server is

\[
\rho = \frac{\lambda \overline{x}}{\overline{x} + W} = \frac{\lambda_1 \overline{x}_1}{\overline{x} + W} + \frac{\lambda_2 \overline{x}_2}{\overline{x} + W} + \cdots + \frac{\lambda_n \overline{x}_n}{\overline{x} + W}.
\]

The average waiting time of a task is ([13, p. 190])

\[
W = \frac{\frac{\overline{x}}{\rho}}{2(1 - \rho)} = \frac{\frac{\overline{x}}{\rho}}{2(1 - \rho)}
\]

which can be rewritten as

\[
T_i = \overline{x}_i + W = \overline{x}_i + W = \frac{\lambda_1 \overline{x}_1}{\overline{x} + W} + \frac{\lambda_2 \overline{x}_2}{\overline{x} + W} + \cdots + \frac{\lambda_n \overline{x}_n}{\overline{x} + W}
\]

and

\[
T_i = \frac{\overline{x}_i}{s_i} + W = \frac{\lambda_1 \overline{x}_1}{s_i} + \frac{\lambda_2 \overline{x}_2}{s_i} + \cdots + \frac{\lambda_n \overline{x}_n}{s_i}.
\]

The average task response time of all tasks is

\[
T = \sum_{i=1}^{n} \frac{\lambda_i T_i}{s_i} = \sum_{i=1}^{n} \frac{\lambda_i \overline{x}_i}{s_i} + \frac{\rho}{2(1 - \rho)}
\]

which is actually \(T = \overline{x} + W\), where

\[
\rho = \frac{\lambda_1 \overline{x}_1}{s_1} + \frac{\lambda_2 \overline{x}_2}{s_2} + \cdots + \frac{\lambda_n \overline{x}_n}{s_n}
\]

and

\[
\sigma = \frac{\lambda_1 \overline{x}_1}{s_1} + \frac{\lambda_2 \overline{x}_2}{s_2} + \cdots + \frac{\lambda_n \overline{x}_n}{s_n}.
\]

Assume that the server has a base power consumption \(P^*\), and consumes no dynamic power when it is idle. The average power consumption (measured in Watts) of the server is

\[
P = \sum_{i=1}^{n} \lambda_i \overline{x}_i s_i^\alpha + P^* = \sum_{i=1}^{n} \frac{\lambda_i \overline{x}_i}{s_i} s_i^{\alpha-1} + P^*.
\]

(Note: This is the idle speed model in [16].)
IV. POWER CONSTRAINED PERFORMANCE OPTIMIZATION

A. Optimal Speed Setting

Given task arrival rates $\lambda_1, \lambda_2, \ldots, \lambda_n$, expected task execution requirements $r_1, r_2, \ldots, r_n$, the second moments of task execution requirements $\overline{r_1^2}, \overline{r_2^2}, \ldots, \overline{r_n^2}$, base power consumption $P^*$, and certain power supply $P$, our problem is to find server speeds $s_1, s_2, \ldots, s_n$, such that $T$ is minimized and that $P$ does not exceed $P^*$.

We can solve the above-mentioned optimization problem, which is a multivariable optimization problem with a constraint, by using the method of Lagrange multiplier, namely,

$$\nabla T(s_1, s_2, \ldots, s_n) = \phi \nabla P(s_1, s_2, \ldots, s_n)$$

that is,

$$\frac{\partial T}{\partial s_i} = \phi \frac{\partial P}{\partial s_i}$$

for all $1 \leq i \leq n$, where $\phi$ is a Lagrange multiplier. Since

$$\frac{\partial T}{\partial s_i} = -\frac{\lambda_i \overline{r_i}}{s_i^2}
\quad + \frac{1}{2} \left(1 - \rho \right) \left( -\frac{2\lambda_i \overline{r_i^2}}{s_i^3} \right)
\quad + \frac{\sigma}{(1 - \rho)^2} \left( -\frac{\lambda_i \overline{r_i}}{s_i^2} \right)$$

and

$$\frac{\partial P}{\partial s_i} = (\alpha - 1) \lambda_i \overline{r_i} s_i^{-\alpha - 2}$$

we have

$$-\frac{\lambda_i \overline{r_i}}{s_i^2} - \frac{1}{1 - \rho} \frac{\lambda_i \overline{r_i}}{s_i^2} - \frac{\sigma}{2(1 - \rho)^2} \frac{\lambda_i \overline{r_i}}{s_i^2} = \phi(\alpha - 1) \lambda_i \overline{r_i} s_i^{-\alpha - 2}$$

for all $1 \leq i \leq n$. The last equation can be rewritten as

$$\frac{1}{\lambda} + \frac{1}{1 - \rho} \frac{\overline{r_i}}{\overline{r_i}} \frac{1}{s_i} + \frac{\sigma}{2(1 - \rho)^2} \frac{1}{s_i} = -\phi(\alpha - 1) s_i^\alpha$$

or

$$F_i = \phi(\alpha - 1) s_i^\alpha + \frac{1}{1 - \rho} \frac{\overline{r_i}}{\overline{r_i}} \frac{1}{s_i} + \frac{\sigma}{2(1 - \rho)^2} + \frac{1}{\lambda} = 0$$

for all $1 \leq i \leq n$. The above-mentioned equation together with

$$F_0 = \sum_{i=1}^{n} \lambda_i \overline{r_i} s_i^{-\alpha - 1} + P^* - \bar{P} = 0$$

constitute a nonlinear system of $n + 1$ equations with $n + 1$ unknowns, i.e., $s_1, s_2, \ldots, s_n$, and $\phi$.

The following theorem shows that it is very unlikely that an optimal server speed setting yields a constant speed.

Theorem 1: An optimal server speed setting yields a constant speed, i.e., $s_1 = s_2 = \cdots = s_n$, if and only if all the $\overline{r_i}/\overline{r_i}$ are identical.

Proof: Notice that $\overline{r_i}/\overline{r_i}$ is the only unique term in $F_i$, for all $1 \leq i \leq n$. If all the $\overline{r_i}/\overline{r_i}$ are identical, we have $s_1 = s_2 = \cdots = s_n$. On the other hand, if $\overline{r_i}/\overline{r_i} \neq \overline{r_j}/\overline{r_j}$ for some $i$ and $j$, then $s_i \neq s_j$.

1) Numerical Algorithm: We are going to solve the following nonlinear system of equations:

$$F_0(\phi, s_1, \ldots, s_n) = 0$$

$$F_1(\phi, s_1, \ldots, s_n) = 0$$

$$\vdots$$

$$F_n(\phi, s_1, \ldots, s_n) = 0.$$

The variables $\phi, s_1, \ldots, s_n$ can be represented by using a vector notation as follows:

$$\mathbf{y} = (y_0, y_1, \ldots, y_n) = (\phi, s_1, \ldots, s_n).$$

Hence, we get $F_i(\phi, s_1, \ldots, s_n) = F_i(y)$, where $F_i : \mathbb{R}^{n+1} \rightarrow \mathbb{R}$ maps $(n + 1)$-dimensional space $\mathbb{R}^{n+1}$ into the real line $\mathbb{R}$. Let us define a function $\mathbf{F} : \mathbb{R}^{n+1} \rightarrow \mathbb{R}^{n+1}$ which maps $\mathbb{R}^{n+1}$ into $\mathbb{R}^{n+1}$

$$\mathbf{F}(\mathbf{y}) = (F_0(y_0, y_1, \ldots, y_n), \ldots, F_n(y_0, y_1, \ldots, y_n))$$

namely,

$$\mathbf{F}(\mathbf{y}) = (F_0(y), F_1(y), \ldots, F_n(y)).$$

Then, the above-mentioned nonlinear system of equations becomes $\mathbf{F}(\mathbf{y}) = 0$, where $0 = (0, 0, \ldots, 0)$.

We can solve the above-mentioned nonlinear system of equations by using Newton’s method. For this purpose, we need the Jacobian matrix $J(\mathbf{y})$ defined as

$$J(\mathbf{y}) = \begin{bmatrix}
\frac{\partial F_0(y)}{\partial y_0} & \frac{\partial F_0(y)}{\partial y_1} & \cdots & \frac{\partial F_0(y)}{\partial y_n} \\
\frac{\partial F_1(y)}{\partial y_0} & \frac{\partial F_1(y)}{\partial y_1} & \cdots & \frac{\partial F_1(y)}{\partial y_n} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial F_n(y)}{\partial y_0} & \frac{\partial F_n(y)}{\partial y_1} & \cdots & \frac{\partial F_n(y)}{\partial y_n}
\end{bmatrix}.$$
for all $1 \leq i \leq n$, and

$$
\frac{\partial F_i(y)}{\partial y_i} = \frac{\partial F_i(y)}{\partial s_i} = \phi_{i}(\alpha - 1)s_i^{\alpha - 1}
$$

$$
+ \frac{\gamma_i}{\pi} \left( \frac{1}{(1-\rho)^2} \left( -\frac{\lambda_r t_i}{s_i^2} \right) \right) \frac{1}{s_i} + \frac{1}{1-\rho} \left( -\frac{1}{s_i^2} \right)
$$

$$
+ \frac{1}{2} \left( \frac{1}{(1-\rho)^2} \left( -\frac{2\lambda_r t_i}{s_i^2} \right) \right) \left( 2\sigma \left( \frac{1}{1-\rho^3} \right) \right)
$$

for all $1 \leq i \leq n$, and

$$
\frac{\partial F_i(y)}{\partial y_j} = \frac{\partial F_i(y)}{\partial s_j} = \frac{\gamma_i}{\pi} \left( \frac{1}{(1-\rho)^2} \left( -\frac{\lambda_r t_i}{s_j^2} \right) \right) \frac{1}{s_i}
$$

$$
+ \frac{1}{2} \left( \frac{1}{(1-\rho)^2} \left( -\frac{2\lambda_r t_i}{s_j^2} \right) \right) \left( 2\sigma \left( \frac{1}{1-\rho^3} \right) \right)
$$

for all $1 \leq i \leq n$ and all $1 \leq j \neq i \leq n$.

Algorithm 1 formally describes our numerical algorithm to find an optimal server speed setting $(s_1, \ldots, s_n)$ and the Lagrange multiplier $\phi$, i.e., the vector $y = (\phi, s_1, \ldots, s_n)$, which satisfies the nonlinear system of equations $F(y) = 0$. This is basically the classic Newton’s iterative method ([4, p. 451]). The initial approximation of $y$ is $\phi = -1$ and $s_j = s$ for all $1 \leq j \leq n$ [line (1)], where $s$ is the constant speed of the server, which satisfies

$$
\sum_{i=1}^{n} \lambda_i t_i s^{\alpha - 1} + P^* = P
$$

that is,

$$
s = \left( (P - P^*) \left( \sum_{i=1}^{n} \lambda_i t_i \right)^{-1} \right)^{1/(\alpha - 1)}. \tag{1}
$$

We repeatedly modify the value of $y$ as $y + z$ [line (6)], where $z$ is the solution to the linear system of equations $J(y)z = -F(y)$ [line (5)]. We repeat the above-mentioned modification until $\|z\| \leq \epsilon$ [line (7)], where

$$
\|z\| = \sqrt{z_0^2 + z_1^2 + \cdots + z_n^2}
$$

and $\epsilon$ is a sufficiently small constant, e.g., $10^{-10}$. By using the classic Gaussian elimination with backward substitution algorithm ([4, pp. 268–269]), we can solve the linear system of equations in line (5).

The time complexity of Algorithm 1 is mainly determined by the number of repetitions of the loop in lines (2)–(7), which depends on the accuracy requirement $\epsilon$.

B. Performance Comparison

In the section, the performance of a server with the optimal speed setting is compared with that of a serve with a constant speed.
and the second moment of $r_i$ is
\[
\beta_i (\beta_i - 2) = \frac{1}{\beta_i (\beta_i - 2)}
\]
\[
\tilde{r}_i^2 = \beta_i (\beta_i - 2)
\]

A nice feature of a Pareto distribution is that for any $r_i > 0$ and $\tilde{r}_i > 0$, there are $\tilde{r}_i > 0$ and $\beta_i > 2$, such that the expectation of $r_i$ is $\tilde{r}_i$ and the second moment of $r_i$ is $\tilde{r}_i^2$. Notice that
\[
c_i = \frac{\tilde{r}_i^2}{\bar{r}_i} = (\beta_i - 1)^2 = 1 + \frac{1}{\beta_i (\beta_i - 2)}
\]

namely,
\[
\frac{1}{\beta_i (\beta_i - 2)} = c_i - 1 > 0.
\]

Since the left-hand side of the equation is a decreasing function of $\beta_i$ in the domain $(2, \infty)$ and in the range $(0, \infty)$, there is always a unique $\beta_i > 2$ for any $c_i > 1$. Once $\beta_i$ is known, $\tilde{r}_i$ can be determined as
\[
\tilde{r}_i = \left( \frac{\beta_i - 1}{\beta_i} \right) \bar{r}_i.
\]

For the purpose of illustration, let us consider $n = 6$ types of applications. The task arrival rates are $\lambda_i = 0.5 + 0.1(i - 1)$, for all $1 \leq i \leq n$. The expected task execution requirements are $\bar{r}_i = 1.2 - 0.2(i - 1)$, for all $1 \leq i \leq n$. The second moments of task execution requirements are $\tilde{r}_i^2 = 1.5 + 0.5(i - 1)$, for all $1 \leq i \leq n$. The base power consumption is $P^* = 10$. To ensure $\rho < 1$, we need
\[
\tilde{P} > P^* + \left( \sum_{i=1}^{n} \lambda_i \bar{r}_i \right) ^\alpha.
\]

The given power supply is
\[
\tilde{P} = P^* + (1 + 0.2b) \left( \sum_{i=1}^{n} \lambda_i \bar{r}_i \right) ^\alpha.
\]

Let $T_{var}$ denote the average task response time with the optimal variable server speed setting, $T_{con}$ denote the average task response time with the constant server speed setting. The relative difference between $T_{var}$ and $T_{con}$ is
\[
\Delta_T = \left( \frac{T_{con} - T_{var}}{T_{con}} \right) \times 100%.
\]

In Table II, for $b = 4, 8, 12, 16, 20$, where $b$ decides $\tilde{P}$, we display the power constraint $\tilde{P}$, the optimal server speed setting $s_1, s_2, s_3, s_4, s_5, s_6$, server utilization $\rho$, and the optimal average task response time $T_{var}$. As comparison, we also show the constant speed $s$ and the resulted server utilization $\rho$ and average task response time $T_{con}$. Finally, we give the relative difference $\Delta_T$ between $T_{var}$ and $T_{con}$.

In Fig. 1, we demonstrate $T_{var}$ and $T_{con}$ for $b = 1, 2, 3, \ldots, 20$.

In Fig. 2, we show the relative difference $\Delta_T$ between $T_{var}$ and $T_{con}$ for $b = 1, 2, 3, \ldots, 20$.

The following observations are made.

1) The differences among the $s_i$ s can be very significant, especially when $\tilde{P}$ is large. In particular, the server speed
can be increased for a type of applications with greater task arrival rate and greater coefficient of variation of task execution requirement.

2) The optimal variable speed setting yields higher server utilization than the constant speed setting.

3) There is noticeable difference between \( T_{\text{var}} \) and \( T_{\text{con}} \), which can be as high as 9.9%.

4) The number of repetitions of the loop in Algorithm 1 is between 8 and 9. All the data in Table II and Figs. 1 and 2 can be produced in less than one second.

V. PERFORMANCE CONSTRAINED POWER MINIMIZATION

A. Optimal Speed Setting

Given task arrival rates \( \lambda_1, \lambda_2, \ldots, \lambda_n \), expected execution requirements \( \bar{r}_1, \bar{r}_2, \ldots, \bar{r}_n \), the second moments of task execution requirements \( \bar{r}_1^2, \bar{r}_2^2, \ldots, \bar{r}_n^2 \), average power consumption \( P^* \), and certain quality of service \( \bar{T} \), our problem is to find server speeds \( s_1, s_2, \ldots, s_n \), such that \( T = \bar{T} \), and that \( P \) is minimized.

1) Numerical Algorithm: We can solve the above-mentioned optimization problem by using the bisection method ([4, p. 22]) to search \( P \) in an appropriately chosen interval \([P_{\text{lb}}, P_{\text{ub}}] \), where \( P_{\text{lb}} \) and \( P_{\text{ub}} \) are the lower and upper bounds of the interval, such that when a server is given power supply \( P \), the average task response time is \( \bar{T} \). The value \( P_{\text{lb}} \) is chosen in such a way that when the server is given power supply \( P_{\text{lb}} \), the average task response time is greater than \( \bar{T} \). The value \( P_{\text{ub}} \) is chosen in such a way that when the server is given power supply \( P_{\text{ub}} \), the average task response time is less than \( \bar{T} \). The time complexity of this algorithm is determined the number of times Algorithm 1 is called by the bisection method.

B. Performance Comparison

In this section, we compare the power consumption of a server with the optimal speed setting with that of a server with a constant speed.

For a constant speed server, i.e., \( s_1 = s_2 = \ldots = s_n = s \), we have

\[
\frac{1}{\lambda s} \sum_{i=1}^{n} \lambda_i \bar{r}_i + \frac{1}{2s} \left( \frac{n}{s} - \sum_{i=1}^{n} \lambda_i \bar{r}_i \right) \sum_{i=1}^{n} \lambda_i \bar{r}_i^2 = \bar{T}.
\]

The above-mentioned equation is actually a quadratic equation

\[
2\bar{T} s^2 - 2bs - c = 0,
\]

where

\[
b = \left( \bar{T} + \frac{1}{\lambda} \right) \left( \sum_{i=1}^{n} \lambda_i \bar{r}_i \right)
\]

and

\[
c = \sum_{i=1}^{n} \lambda_i \bar{r}_i^2 - \frac{2}{\lambda} \left( \sum_{i=1}^{n} \lambda_i \bar{r}_i \right)^2.
\]

It is clear that

\[
s = \frac{2b + \sqrt{4b^2 + 8\bar{T} c}}{4\bar{T}} = b + \sqrt{b^2 + 2\bar{T} c}.
\]

Therefore, we obtain

\[
\begin{align*}
\sum_{i=1}^{n} \lambda_i \bar{r}_i & \leq \frac{1}{2\bar{T}} \left( \bar{T} + \frac{1}{\lambda} \right) \left( \sum_{i=1}^{n} \lambda_i \bar{r}_i \right) \\
+ \sqrt{\left( \bar{T} - \frac{1}{\lambda} \right)^2 \left( \sum_{i=1}^{n} \lambda_i \bar{r}_i \right)^2 + 2\bar{T} \left( \sum_{i=1}^{n} \lambda_i \bar{r}_i \right)}.
\end{align*}
\]

The average power consumption of the server is

\[
P = \left( \sum_{i=1}^{n} \lambda_i \bar{r}_i \right)^{s^{\lambda - 1}} + P^*.
\]

which is actually

\[
P = \left( \sum_{i=1}^{n} \lambda_i \bar{r}_i \right) \left( \frac{1}{2\bar{T}} \left( \bar{T} + \frac{1}{\lambda} \right) \left( \sum_{i=1}^{n} \lambda_i \bar{r}_i \right) \right)^{\alpha - 1} + P^*.
\]

Let \( P_{\text{var}} \) denote the average power consumption with the optimal variable server speed setting, \( P_{\text{con}} \) denote the average power consumption with the constant server speed setting. The relative difference between \( P_{\text{var}} \) and \( P_{\text{con}} \) is

\[
\Delta P = \left( \frac{P_{\text{con}} - P_{\text{var}}}{P_{\text{con}}} \right) \times 100%.
\]

Let us consider the same types of applications in Section IV.B. The given quality of service is \( \bar{T} = 0.3b \).

In Table III, for \( b = 4, 8, 12, 16, 20 \), where \( b \) decides \( \bar{T} \), we display the time constraint \( \bar{T} \), the optimal server speed setting \( s_1, s_2, s_3, s_4, s_5, s_6 \), server utilization \( \rho \), and the minimum average power consumption \( P_{\text{var}} \). As comparison, we also show
In Fig. 3, we demonstrate $P_{\text{var}}$ and $P_{\text{con}}$ for $b = 1, 2, 3, \ldots, 20$.

In Fig. 4, we show the relative difference $\Delta P$ between $P_{\text{var}}$ and $P_{\text{con}}$ for $b = 1, 2, 3, \ldots, 20$.

The following observations are made.

1) The differences among the $s_i$ can be very significant, especially when $\bar{T}$ is small. In particular, the server speed can be increased for a type of applications with greater task arrival rate and greater coefficient of variation of task execution requirement.

2) The optimal variable speed setting yields higher server utilization than the constant speed setting.

3) There is noticeable difference between $P_{\text{var}}$ and $P_{\text{con}}$, which can be as high as 8.0%. In fact, it is unbounded as $\bar{T} \to 0$.

4) Algorithm 1 is called 44 times by the bisection method in Section V-A. All the data in Table III and Figs. 3 and 4 can be produced in less than one second.

VI. CONCLUSION

A new kind of workload-dependent dynamic power and the speed management (i.e., variable and task type dependent server speed management) method to deal with the power and performance tradeoff for cloud servers is introduced in this paper. Both power constrained performance optimization and performance constrained power minimization are investigated as optimization problems solved by efficient numerical algorithms. Our main conclusions are two fold. First, it is shown that compared with a server with a constant speed, a server with the optimal speed setting can noticeably reduce the average task response time and the average power consumption. Second, it is also shown that our numerical algorithms are very fast. The research in this paper has made significant contribution to analytical study of power and performance optimization using the technique of variable and task type dependent server speed management for a server with mixed applications.

The research in this paper can be extended in a number of ways. First, an M/G/1 server can be extended to an M/G/m server. Due to lack of an analytical expression of the average task response, such a study is very challenging. Second, multiple M/G/1 and/or M/G/m servers can be investigated. When there are multiple heterogeneous servers with variable and task type dependent server speed management, we are facing the challenges of both optimal load distribution and optimal server speed setting for multiple classes of applications. It is conceivable that such a problem requires extra effort to deal with. Although some attempt has been made toward this direction [19], deeper investigation is required. Third, more sophisticated scheduling strategies other than FCFS can be considered.

ACKNOWLEDGMENT

The author would like to thank three anonymous reviewers and the editor for their comments and suggestions to improve the quality of the manuscript.

REFERENCES


Kegin Li (F’15) received Ph.D. degree in computer science from the University of Houston in 1990. He is a SUNY Distinguished Professor of computer science with the State University of New York. He is also a Distinguished Professor with the Chinese National Recruitment Program of Global Experts (1000 Plan), Hunan University, Changsha, China. He was an Intellectual Ventures endowed Visiting Chair Professor with the National Laboratory for Information Science and Technology, Tsinghua University, Beijing, China, during 2011–2014. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU-GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of things, and cyber-physical systems. He has authored or coauthored more than 570 journal articles, book chapters, and refereed conference papers.

Dr. Li was the recipient of several best paper awards. He is currently serving or has served on the editorial boards of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON CLOUD COMPUTING, the IEEE TRANSACTIONS ON SERVICES COMPUTING, and the IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING.