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 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. 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

Manuscript received May 2, 2018; revised July 6, 2018; accepted July 12, 2018. Paper no. TII-18-1103. The research is supported in part by the Key Program of National Natural Science Foundation of China under Grant No. 61432005.

The author is with the College of Information Science and Engineering, Hunan University, Hunan 410082, China, and also with the Department of Computer Science, State University of New York, New Paltz, NY 12561, USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/TII.2018.2856909

1551-3203 © 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

IEEE Proof

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

IEEE Proof

TABLE I
NOTATIONS AND DEFINITIONS

Notation	Definition
n	the number of types of applications
λ_i	the task arrival rate of the ith type of applications
λ	the total task arrival rate
r_i	the execution requirements of the tasks of the ith type of appli.
s_i	the execution speed of the server for the ith type of applications
x_i	the execution times of the tasks of the ith type of applications
x	the execution time of a task of all applications
ρ	the utilization of the server
$\overline{x}$, $\overline{x^2}$	the mean and the second moment of x
W	the average waiting time of a task
σ	$\lambda_1\overline{x_1^2}+\lambda_2\overline{x_2^2}+\cdots+\lambda_n\overline{x_n^2}$
T_i	the average response time of tasks of the ith type of applications
$\overline{r_i}$, $\overline{r_i^2}$	the mean and the second moment of r_i
T	the average task response time of all tasks
P^*	base power consumption of the server
α	exponent of the power consumption model
P	the average power consumption of the server
$\tilde{P}$	power constraint
ϕ	a Lagrange multiplier
F_i	a non-linear system of $n+1$ equations
$\mathbf{y}$	$(y_0, y_1, ..., y_n) = (\phi, s_1, ..., s_n)$
$\mathbf{F}(\mathbf{y})$	$(F_0(\mathbf{y}), F_1(\mathbf{y}), ..., F_n(\mathbf{y}))$
$J(\mathbf{y})$	Jacobian matrix with $J(\mathbf{y})_{i,j} = \partial F_i(\mathbf{y})/\partial y_j$, $0 \le i, j \le n$
$\tilde{T}$	time constraint

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, etc.).

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 ith type of applications is λ_i, where $1 \le i \le n$. The total task arrival rate is $\lambda = \lambda_1 + \lambda_2 + \cdots + \lambda_n$.

For the ith 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 ith 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 ith 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

$$\overline{x} = \frac{\lambda_1}{\lambda}\overline{x_1} + \frac{\lambda_2}{\lambda}\overline{x_2} + \cdots + \frac{\lambda_n}{\lambda}\overline{x_n}.$$

The utilization of the server is $\rho = \lambda\overline{x} = \lambda_1\overline{x_1} + \lambda_2\overline{x_2} + \cdots + \lambda_n\overline{x_n}$. It is noticed that the server utilization depends on the arrival rates, the execution requirements, and the execution speeds of all the n types of applications. The second moment of x (i.e., the mean of x^2) is

$$\overline{x^2} = \frac{\lambda_1}{\lambda}\overline{x_1^2} + \frac{\lambda_2}{\lambda}\overline{x_2^2} + \cdots + \frac{\lambda_n}{\lambda}\overline{x_n^2}.$$

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

$$W = \frac{\lambda\overline{x^2}}{2(1-\rho)} = \frac{\sigma}{2(1-\rho)}$$

where $\sigma = \lambda_1\overline{x_1^2} + \lambda_2\overline{x_2^2} + \cdots + \lambda_n\overline{x_n^2}$. The average response time of tasks of the ith type of applications is

$$T_i = \overline{x_i} + W = \overline{x_i} + \frac{\sigma}{2(1-\rho)}$$

which can be rewritten as

$$T_i = \overline{x_i} + \frac{\lambda_1\overline{x_1^2} + \lambda_2\overline{x_2^2} + \cdots + \lambda_n\overline{x_n^2}}{2(1 - \lambda_1\overline{x_1} - \lambda_2\overline{x_2} - \cdots - \lambda_n\overline{x_n})}$$

and

$$T_i = \frac{\overline{r_i}}{s_i} + \frac{\lambda_1\overline{r_1^2}/s_1^2 + \lambda_2\overline{r_2^2}/s_2^2 + \cdots + \lambda_n\overline{r_n^2}/s_n^2}{2(1 - \lambda_1\overline{r_1}/s_1 - \lambda_2\overline{r_2}/s_2 - \cdots - \lambda_n\overline{r_n}/s_n)}.$$

The average task response time of all tasks is

$$T = \sum_{i=1}^{n}\frac{\lambda_i}{\lambda}T_i = \frac{1}{\lambda}\sum_{i=1}^{n}\frac{\lambda_i\overline{r_i}}{s_i} + \frac{\sigma}{2(1-\rho)}$$

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

$$\rho = \lambda_1\frac{\overline{r_1}}{s_1} + \lambda_2\frac{\overline{r_2}}{s_2} + \cdots + \lambda_n\frac{\overline{r_n}}{s_n}$$

and

$$\sigma = \lambda_1\frac{\overline{r_1^2}}{s_1^2} + \lambda_2\frac{\overline{r_2^2}}{s_2^2} + \cdots + \lambda_n\frac{\overline{r_n^2}}{s_n^2}.$$

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

$$P = \sum_{i=1}^{n}\lambda_i\overline{x_i}s_i^{\alpha} + P^* = \sum_{i=1}^{n}\lambda_i\overline{r_i}s_i^{\alpha-1} + P^*.$$

(Note: This is the idle speed model in [16].)

IEEE Proof

IV. Power Constrained Performance Optimization

A. Optimal Speed Setting

Given task arrival rates $\lambda_1, \lambda_2, \ldots, \lambda_n$, expected task execution requirements $\overline{r_1}$, $\overline{r_2}$, …, $\overline{r_n}$, 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 $\tilde{P}$, 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 $\tilde{P}$.

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 $1 \le i \le n$, where ϕ is a Lagrange multiplier. Since

$$\begin{aligned}\frac{\partial T}{\partial s_i} = &- \frac{1}{\lambda} \cdot \frac{\lambda_i \overline{r_i}}{s_i^2}\\ &+ \frac{1}{2}\left(\frac{1}{1-\rho}\left(-\frac{2\lambda_i \overline{r_i^2}}{s_i^3}\right) + \frac{\sigma}{(1-\rho)^2}\left(-\frac{\lambda_i \overline{r_i}}{s_i^2}\right)\right)\end{aligned}$$

and

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

we have

$$\begin{aligned}&-\frac{1}{\lambda} \cdot \frac{\lambda_i \overline{r_i}}{s_i^2} - \frac{1}{1-\rho} \cdot \frac{\lambda_i \overline{r_i^2}}{s_i^3} - \frac{\sigma}{2(1-\rho)^2} \cdot \frac{\lambda_i \overline{r_i}}{s_i^2}\\ &= \phi(\alpha - 1)\lambda_i \overline{r_i} s_i^{\alpha-2}\end{aligned}$$

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

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

or

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

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

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

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

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 $\overline{r_i^2}/\overline{r_i}$ are identical.

Proof: Notice that $\overline{r_i^2}/\overline{r_i}$ is the only unique term in F_i, for all $1 \le i \le n$. If all the $\overline{r_i^2}/\overline{r_i}$ are identical, we have $s_1 = s_2 = \cdots = s_n$. On the other hand, if $\overline{r_i^2}/\overline{r_i} \ne \overline{r_j^2}/\overline{r_j}$ for some i and j, then $s_i \ne s_j$.

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

$$\begin{aligned}F_0(\phi, s_1, \ldots, s_n) &= 0\\ F_1(\phi, s_1, \ldots, s_n) &= 0\\ &\vdots\\ F_n(\phi, s_1, \ldots, s_n) &= 0.\end{aligned}$$

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 $F_i(\phi, s_1, \ldots, s_n) = F_i(y_0, y_1, \ldots, y_n) = F_i(\mathbf{y})$, where $F_i : \mathbb{R}^{n+1} \to \mathbb{R}$ maps $(n+1)$-dimensional space $\mathbb{R}^{n+1}$ into the real line $\mathbb{R}$. Let us define a function $\mathbf{F} : \mathbb{R}^{n+1} \to \mathbb{R}^{n+1}$ 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,

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

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

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

$$J(\mathbf{y}) = \begin{bmatrix} \frac{\partial F_0(\mathbf{y})}{\partial y_0} & \frac{\partial F_0(\mathbf{y})}{\partial y_1} & \cdots & \frac{\partial F_0(\mathbf{y})}{\partial y_n} \\ \frac{\partial F_1(\mathbf{y})}{\partial y_0} & \frac{\partial F_1(\mathbf{y})}{\partial y_1} & \cdots & \frac{\partial F_1(\mathbf{y})}{\partial y_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_n(\mathbf{y})}{\partial y_0} & \frac{\partial F_n(\mathbf{y})}{\partial y_1} & \cdots & \frac{\partial F_n(\mathbf{y})}{\partial y_n} \end{bmatrix}.$$

We can calculate the various components of the above-mentioned matrix as follows. First, we have

$$\frac{\partial F_0(\mathbf{y})}{\partial y_0} = \frac{\partial F_0(\mathbf{y})}{\partial \phi} = 0$$

and

$$\frac{\partial F_0(\mathbf{y})}{\partial y_j} = \frac{\partial F_0(\mathbf{y})}{\partial s_j} = (\alpha - 1)\lambda_j \overline{r_j} s_j^{\alpha-2}$$

for all $1 \le j \le n$. Next, we have

$$\frac{\partial F_i(\mathbf{y})}{\partial y_0} = \frac{\partial F_i(\mathbf{y})}{\partial \phi} = (\alpha - 1)s_i^{\alpha}$$

IEEE Proof

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

$$\frac{\partial F_i(\mathbf{y})}{\partial y_i} = \frac{\partial F_i(\mathbf{y})}{\partial s_i} = \phi\alpha(\alpha-1)s_i^{\alpha-1} + \frac{\overline{r_i^2}}{\overline{r_i}}\left(\frac{1}{(1-\rho)^2}\left(-\frac{\lambda_i\overline{r_i}}{s_i^2}\right)\frac{1}{s_i} + \frac{1}{1-\rho}\left(-\frac{1}{s_i^2}\right)\right) + \frac{1}{2}\left(\frac{1}{(1-\rho)^2}\left(-\frac{2\lambda_i\overline{r_i^2}}{s_i^3}\right) + \frac{2\sigma}{(1-\rho)^3}\left(-\frac{\lambda_i\overline{r_i}}{s_i^2}\right)\right)$$

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

$$\frac{\partial F_i(\mathbf{y})}{\partial y_j} = \frac{\partial F_i(\mathbf{y})}{\partial s_j} = \frac{\overline{r_i^2}}{\overline{r_i}} \cdot \frac{1}{(1-\rho)^2}\left(-\frac{\lambda_j\overline{r_j}}{s_j^2}\right)\frac{1}{s_i} + \frac{1}{2}\left(\frac{1}{(1-\rho)^2}\left(-\frac{2\lambda_j\overline{r_j^2}}{s_j^3}\right) + \frac{2\sigma}{(1-\rho)^3}\left(-\frac{\lambda_j\overline{r_j}}{s_j^2}\right)\right)$$

for all $1 \le i \le n$ and all $1 \le j \ne i \le n$.

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

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

that is,

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

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

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

and ϵ 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 ϵ.

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.

Algorithm 1: Optimal Server Speed Setting.

Input: Parameters $\lambda_1, \lambda_2, \ldots, \lambda_n$, $\overline{r_1}, \overline{r_2}, \ldots, \overline{r_n}$, $\overline{r_1^2}, \overline{r_2^2}, \ldots, \overline{r_n^2}$, P^*, and $\tilde{P}$.
Output: An optimal server speed setting and ϕ, i.e., $\mathbf{y} = (\phi, s_1, \ldots, s_n)$, which satisfies $\mathbf{F}(\mathbf{y}) = \mathbf{0}$.

$\mathbf{y} \leftarrow (-1, s, \ldots, s)$; (1)
repeat (2)
  Calculate $J(\mathbf{y})$,
  where $J(\mathbf{y})_{i,j} = \partial F_i(\mathbf{y})/\partial y_j$ for $0 \le i, j \le n$; (3)
  Calculate $\mathbf{F}(\mathbf{y}) = (F_0(\mathbf{y}), F_1(\mathbf{y}), \ldots, F_n(\mathbf{y}))$; (4)
  Solve the linear system of equations
  $J(\mathbf{y})\mathbf{z} = -\mathbf{F}(\mathbf{y})$; (5)
  $\mathbf{y} \leftarrow \mathbf{y} + \mathbf{z}$; (6)
until $\|\mathbf{z}\| \le \epsilon$. (7)

For a constant speed server, i.e., $s_1 = s_2 = \cdots = s_n = s$, we have

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

The above-mentioned server speed yields

$$\rho = \left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)^{\alpha/(\alpha-1)}\left(\frac{1}{\tilde{P} - P^*}\right)^{1/(\alpha-1)}$$

and

$$\sigma = \left(\sum_{i=1}^{n}\lambda_1\overline{r_i^2}\right)\left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)^{2/(\alpha-1)}\left(\frac{1}{\tilde{P} - P^*}\right)^{2/(\alpha-1)}.$$

The average task response time of all tasks is

$$T = \frac{\rho}{\lambda} + \frac{\sigma}{2(1-\rho)}$$

which is

$$T = \frac{1}{\lambda}\left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)^{\alpha/(\alpha-1)}\left(\frac{1}{\tilde{P} - P^*}\right)^{1/(\alpha-1)} + \frac{\left(\sum_{i=1}^{n}\lambda_1\overline{r_i^2}\right)\left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)^{2/(\alpha-1)}\left(\frac{1}{\tilde{P}-P^*}\right)^{2/(\alpha-1)}}{2\left(1 - \left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)^{\alpha/(\alpha-1)}\left(\frac{1}{\tilde{P}-P^*}\right)^{1/(\alpha-1)}\right)}.$$

We consider a Pareto distribution [2] of r_i with pdf

$$\frac{\beta_i\tilde{r}_i^{\beta_i}}{r_i^{\beta_i+1}}$$

in the range $r_i \in [\tilde{r}_i, \infty)$, where $\tilde{r}_i \ge 0$ and $\beta_i > 2$. The expectation of r_i is

$$\overline{r_i} = \frac{\beta_i\tilde{r}_i}{\beta_i - 1}$$

IEEE Proof

and the second moment of r_i is

$$\overline{r_i^2} = \left(\frac{\beta_i}{\beta_i - 2}\right)\tilde{r}_i^2.$$

A nice feature of a Pareto distribution is that for any $\overline{r_i} > 0$ and $\overline{r_i^2} > \overline{r_i}^2$, there are $\tilde{r}_i > 0$ and $\beta_i > 2$, such that the expectation of r_i is $\overline{r_i}$ and the second moment of r_i is $\overline{r_i^2}$. Notice that

$$c_i = \frac{\overline{r_i^2}}{\overline{r_i}^2} = \frac{(\beta_i - 1)^2}{\beta_i(\beta_i - 2)} = 1 + \frac{1}{\beta_i(\beta_i - 2)}$$

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 β_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 β_i is known, $\tilde{r}_i$ can be determined as

$$\tilde{r}_i = \left(\frac{\beta_i - 1}{\beta_i}\right)\overline{r_i}.$$

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 $1 \le i \le n$. The expected task execution requirements are $\overline{r_i} = 1.2 - 0.2(i - 1)$, for all $1 \le i \le n$. The second moments of task execution requirements are $\overline{r_i^2} = 1.5 + 0.5(i - 1)$, for all $1 \le i \le n$. The base power consumption is $P^* = 10$. To ensure $\rho < 1$, we need

$$\tilde{P} > P^* + \left(\sum_{i=1}^{n} \lambda_i \overline{r_i}\right)^{\alpha}.$$

The given power supply is

$$\tilde{P} = P^* + (1 + 0.2b)\left(\sum_{i=1}^{n} \lambda_i \overline{r_i}\right)^{\alpha}.$$

Let $T_{\rm var}$ denote the average task response time with the optimal variable server speed setting, $T_{\rm con}$ denote the average task response time with the constant server speed setting. The relative difference between $T_{\rm var}$ and $T_{\rm con}$ is

$$\Delta_T = \left(\frac{T_{\rm con} - T_{\rm var}}{T_{\rm con}}\right) \times 100\%.$$

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 ρ, and the optimal average task response time $T_{\rm var}$. As comparison, we also show the constant speed s and the resulted server utilization ρ and average task response time $T_{\rm con}$. Finally, we give the relative difference Δ_T between $T_{\rm var}$ and $T_{\rm con}$.

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

In Fig. 2, we show the relative difference Δ_T between $T_{\rm var}$ and $T_{\rm con}$ 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

TABLE II
NUMERICAL DATA FOR POWER CONSTRAINED OPTIMIZATION

	$b=4$	$b=8$	$b=12$	$b=16$	$b=20$
$\tilde{P}$	49.5136000	67.0752000	84.6368000	102.1984000	119.7600000
s_1	3.3919967	3.9322954	4.4204710	4.8683928	5.2847059
s_2	3.4983475	4.1145223	4.6595711	5.1526454	5.6062546
s_3	3.6384553	4.3433847	4.9533109	5.4978463	5.9942584
s_4	3.8361407	4.6514946	5.3409733	5.9488753	6.4984547
s_5	4.1497472	5.1178076	5.9169350	6.6129478	7.2372215
s_6	4.7909610	6.0253151	7.0176979	7.8711810	8.6305931
ρ	0.7559313	0.6340678	0.5567680	0.5020824	0.4607612
$T_{\rm var}$	1.8390434	0.8972191	0.5978647	0.4521420	0.3661072
s	3.7565942	4.5148643	5.1629449	5.7382924	6.2609903
ρ	0.7453560	0.6201737	0.5423261	0.4879500	0.4472136
$T_{\rm con}$	2.0092226	0.9934964	0.6635606	0.5013570	0.4051139
Δ_T	8.4699031	9.6907596	9.9005139	9.8163635	9.6285648

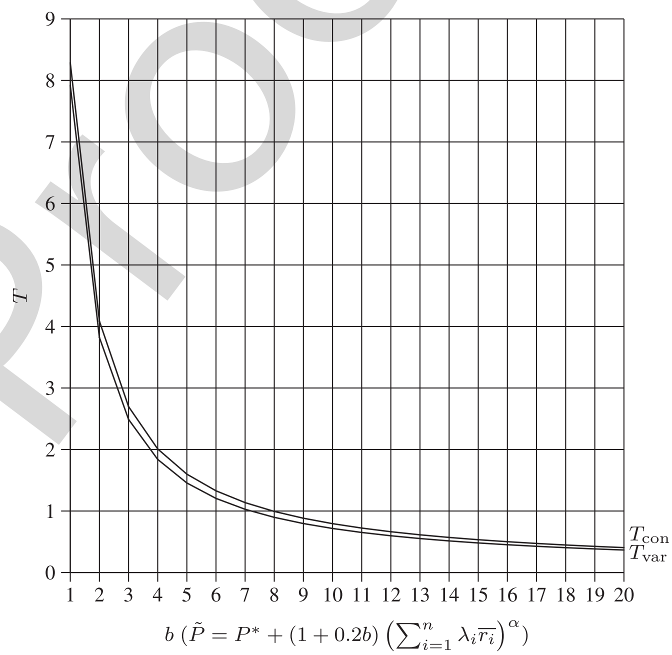

Fig. 1. Average task response time versus power supply.

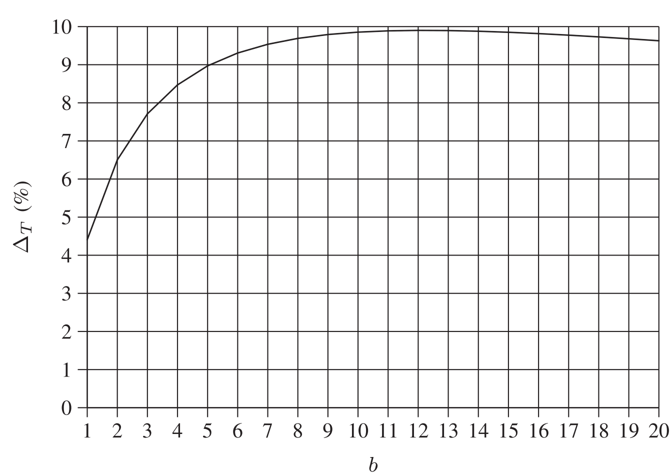

Fig. 2. Relative difference Δ_T between $T_{\rm var}$ and $T_{\rm con}$.

IEEE Proof

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 $T_{\rm var}$ and $T_{\rm con}$, 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 task execution requirements $\overline{r_1}$, $\overline{r_2}$,..., $\overline{r_n}$, the second moments of task execution requirements $\overline{r_1^2}$, $\overline{r_2^2}$,..., $\overline{r_n^2}$, base power consumption P^*, and certain quality of service $\tilde{T}$, our problem is to find server speeds $s_1, s_2, \ldots, s_n$, such that $T = \tilde{T}$, 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 $[P_{\rm lb}, P_{\rm ub}]$, where $P_{\rm lb}$ and $P_{\rm ub}$ 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 $\tilde{T}$. The value $P_{\rm lb}$ is chosen in such a way that when the server is given power supply $P_{\rm lb}$, the average task response time is greater than $\tilde{T}$. The value $P_{\rm ub}$ is chosen in such a way that when the server is given power supply $P_{\rm ub}$, the average task response time is less than $\tilde{T}$. 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 serve with a constant speed.

For a constant speed server, i.e., $s_1 = s_2 = \cdots = s_n = s$, we have

$$\frac{1}{\lambda s}\sum_{i=1}^{n}\lambda_i\overline{r_i} + \frac{1}{2s\left(s-\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)}\sum_{i=1}^{n}\lambda_i\overline{r_i^2} = \tilde{T}.$$

The above-mentioned equation is actually a quadratic equation $2\tilde{T}s^2 - 2bs - c = 0$, where

$$b = \left(\tilde{T} + \frac{1}{\lambda}\right)\left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)$$

and

$$c = \sum_{i=1}^{n}\lambda_i\overline{r_i^2} - \frac{2}{\lambda}\left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)^2.$$

It is clear that

$$s = \frac{2b + \sqrt{4b^2 + 8\tilde{T}c}}{4\tilde{T}} = \frac{b + \sqrt{b^2 + 2\tilde{T}c}}{2\tilde{T}}$$

where

$$b^2 + 2\tilde{T}c = \left(\tilde{T} - \frac{1}{\lambda}\right)^2\left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)^2 + 2\tilde{T}\left(\sum_{i=1}^{n}\lambda_i\overline{r_i^2}\right).$$

Therefore, we obtain

$$s = \frac{1}{2\tilde{T}}\left(\left(\tilde{T} + \frac{1}{\lambda}\right)\left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right) + \sqrt{\left(\tilde{T} - \frac{1}{\lambda}\right)^2\left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)^2 + 2\tilde{T}\left(\sum_{i=1}^{n}\lambda_i\overline{r_i^2}\right)}\right).$$

The average power consumption of the server is

$$P = \left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)s^{\alpha-1} + P^*$$

which is actually

$$P = \left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)\left(\frac{1}{2\tilde{T}}\left(\left(\tilde{T} + \frac{1}{\lambda}\right)\left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right) + \sqrt{\left(\tilde{T} - \frac{1}{\lambda}\right)^2\left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)^2 + 2\tilde{T}\left(\sum_{i=1}^{n}\lambda_i\overline{r_i^2}\right)}\right)\right)^{\alpha-1} + P^*.$$

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

$$\Delta_P = \left(\frac{P_{\rm con} - P_{\rm var}}{P_{\rm con}}\right) \times 100\%.$$

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

In Table III, for $b = 4, 8, 12, 16, 20$, where b decides $\tilde{T}$, we display the time constraint $\tilde{T}$, the optimal server speed setting $s_1, s_2, s_3, s_4, s_5, s_6$, server utilization ρ, and the minimum average power consumption $P_{\rm var}$. As comparison, we also show

TABLE III
Numerical Data for Performance Constrained Optimization

	$b=4$	$b=8$	$b=12$	$b=16$	$b=20$
$\tilde{T}$	1.2000000	2.4000000	3.6000000	4.8000000	6.0000000
s_1	3.6728137	3.2629304	3.1165324	3.0407684	2.9943405
s_2	3.8206132	3.3485019	3.1770230	3.0875972	3.0325554
s_3	4.0097190	3.4632989	3.2602903	3.1531054	3.0866063
s_4	4.2688202	3.6284473	3.3836464	3.2520687	3.1694096
s_5	4.6676909	3.8960336	3.5904651	3.4221005	3.3143226
s_6	5.4574749	4.4564372	4.0420459	3.8056090	3.6498308
ρ	0.6862822	0.7943325	0.8447458	0.8745980	0.8945243
$P_{\rm var}$	58.4027860	45.5883307	41.2403610	39.0241444	37.6731106
s	4.2711786	3.6211022	3.3747446	3.2432770	3.1611412
ρ	0.6555568	0.7732452	0.8296924	0.8633243	0.8857561
$P_{\rm con}$	61.0803072	46.7146674	41.8889237	39.4527683	37.9798784
Δ_P	4.3836079	2.4110986	1.5482915	1.0864229	0.8077114

IEEE Proof

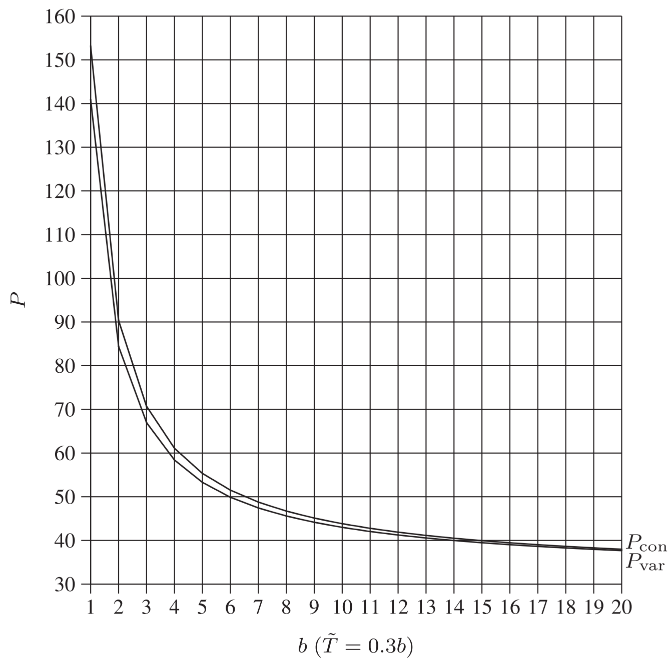

Fig. 3. Average power consumption versus quality of service.

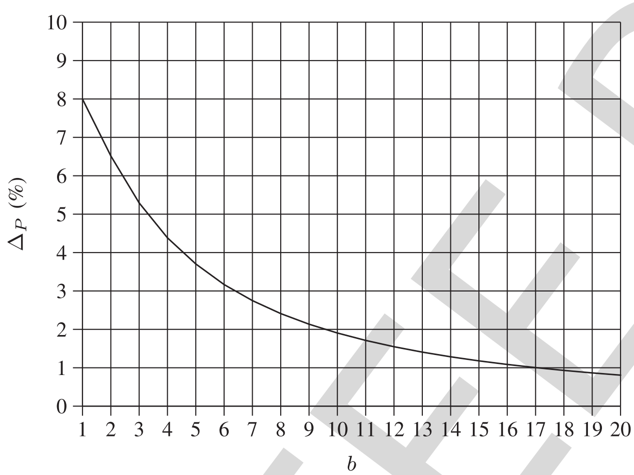

Fig. 4. Relative difference Δ_P between P_{var} and P_{con}.

the constant speed s and the resulted server utilization ρ and average power consumption P_{con}. Finally, we give the relative difference Δ_P between P_{var} and P_{con}.

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

In Fig. 4, we show the relative difference Δ_P between P_{var} and P_{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 $\tilde{T}$ 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_{var} and P_{con}, which can be as high as 8.0%. In fact, it is unbounded as $\tilde{T} \to 0$.
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

[1] 2018. [Online]. Available: http://en.wikipedia.org/wiki/Dynamic_voltage_scaling
[2] 2018. [Online]. Available: http://en.wikipedia.org/wiki/Pareto_distribution
[3] W. L. Bircher and L. K. John, "Analysis of dynamic power management on multicore processors," in *Proc. 22nd ACM Int. Conf. Supercomput.*, 2008, pp. 327–338.
[4] R. L. Burden, J. D. Faires, and A. C. Reynolds, *Numerical Analysis*, 2nd ed. Boston, MA, USA: Prindle, Weber & Schmidt, 1981.
[5] R. Cochran, C. Hankendi, A. Coskun, and S. Reda, "Identifying the optimal energy-efficient operating points of parallel workloads," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2011, pp. 608–615.
[6] S. Dustdar, Y. Guo, B. Satzger, and H.-L. Truong, "Principles of elastic processes," *IEEE Int. Comput.*, vol. 15, no. 5, pp. 66–71, Sep./Oct. 2011.

IEEE Proof

[7] G. Galante and L. C. E. de Bona, “A survey on cloud computing elasticity,” in *Proc. IEEE/ACM 5th Int. Conf. Utility Cloud Comput.*, 2012, pp. 263–270.

[8] N. R. Herbst, “Quantifying the impact of platform configuration space for elasticity benchmarking,” Study thesis, Dept. Informats., Karlsruhe Inst. Technol., Karlsruhe, Germany, 2011.

[9] S. Huang and W. Feng, “Energy-efficient cluster computing via accurate workload characterization,” in *Proc. 9th IEEE/ACM Int. Symp. Cluster Comput. Grid*, 2009, pp. 68–75.

[10] K. Hwang, X. Bai, Y. Shi, M. Li, W.-G. Chen, and Y. Wu, “Cloud performance modeling with benchmark evaluation of elastic scaling strategies,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 1, pp. 130–143, Jan. 2016.

[11] B. Kar, H. K. Wu, and Y. D. Lin, “Energy cost optimization in dynamic placement of virtualized network function chains,” *IEEE Trans. Netw. Serv. Manage.*, vol. 15, no. 1, pp. 372–386, Mar. 2018.

[12] E. Kim, Y. Ko, and S. Ha, “An adaptive frames per second-based CPU-GPU cooperative dynamic voltage and frequency scaling governing technique for mobile games,” *J. Low Power Electron.*, vol. 12, no. 4, pp. 309–322, 2016.

[13] L. Kleinrock, *Queueing Systems, Volume 1: Theory*. New York, NY, USA: Wiley, 1975.

[14] M. Kuperberg, N. Herbst, J. von Kistowski, and R. Reussner, “Defining and quantifying elasticity of resources in cloud computing and scalable platforms,” Karlsruhe Inst. Technol., Karlsruhe, Germany, Rep. no. 16, Informat., 2011.

[15] S. J. Lee, H.-K. Lee, and P.-C. Yew, “Runtime performance projection model for dynamic power management,” in *Proc. 12th Asia-Pac. Comput. Syst. Archit. Conf.*, 2007, pp. 186–197.

[16] K. Li, “Improving multicore server performance and reducing energy consumption by workload dependent dynamic power management,” *IEEE Trans. Cloud Comput.*, vol. 4, no. 2, pp. 122–137, Apr./Jun. 2016.

[17] K. Li, “Optimal task dispatching for multiple heterogeneous multiserver systems with dynamic speed and power management,” *IEEE Trans. Sustain. Comput.*, vol. 2, no. 2, pp. 167–182, 2017.

[18] K. Li, “Quantitative modeling and analytical calculation of elasticity in cloud computing,” *IEEE Trans. Cloud Comput.*, to be published.

[19] K. Li, “Optimal load distribution for multiple classes of applications on heterogeneous servers with variable speeds,” *J. Softw., Pract. Exp.*, to be published.

[20] D. C. Snowdon, E. Le Sueur, S. M. Petters, and G. Heiser, “Koala a platform for OS-level power management,” in *Proc. 4th ACM Eur. Conf. Comput. Syst.*, 2009, pp. 289–302.

[21] P. Sobeslavsky, *Elasticity Cloud Comput.*, Master Thesis, Distributed, Embedded, Mobile, Interactive and Parallel Systems, Joseph Fourier University, Grenoble, France, 2011.

[22] J. Varia, “Architecting for the cloud: Best practices,” Amazon, 2010. [Online]. Available: https://jineshvaria.s3.amazonaws.com/public/cloudbestpractices-jvaria.pdf

Keqin 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.

IEEE Proof

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 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. 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

Manuscript received May 2, 2018; revised July 6, 2018; accepted July 12, 2018. Paper no. TII-18-1103. The research is supported in part by the Key Program of National Natural Science Foundation of China under Grant No. 61432005.

The author is with the College of Information Science and Engineering, Hunan University, Hunan 410082, China, and also with the Department of Computer Science, State University of New York, New Paltz, NY 12561, USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/TII.2018.2856909

1551-3203 © 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

IEEE Proof

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

IEEE Proof

TABLE I
NOTATIONS AND DEFINITIONS

Notation	Definition
n	the number of types of applications
λ_i	the task arrival rate of the ith type of applications
λ	the total task arrival rate
r_i	the execution requirements of the tasks of the ith type of appli.
s_i	the execution speed of the server for the ith type of applications
x_i	the execution times of the tasks of the ith type of applications
x	the execution time of a task of all applications
ρ	the utilization of the server
$\overline{x}$, $\overline{x^2}$	the mean and the second moment of x
W	the average waiting time of a task
σ	$\lambda_1\overline{x_1^2}+\lambda_2\overline{x_2^2}+\cdots+\lambda_n\overline{x_n^2}$
T_i	the average response time of tasks of the ith type of applications
$\overline{r_i}$, $\overline{r_i^2}$	the mean and the second moment of r_i
T	the average task response time of all tasks
P^*	base power consumption of the server
α	exponent of the power consumption model
P	the average power consumption of the server
$\tilde{P}$	power constraint
ϕ	a Lagrange multiplier
F_i	a non-linear system of $n+1$ equations
$\mathbf{y}$	$(y_0,y_1,...,y_n)=(\phi,s_1,...,s_n)$
$\mathbf{F}(\mathbf{y})$	$(F_0(\mathbf{y}),F_1(\mathbf{y}),...,F_n(\mathbf{y}))$
$J(\mathbf{y})$	Jacobian matrix with $J(\mathbf{y})_{i,j}=\partial F_i(\mathbf{y})/\partial y_j$, $0\le i,j\le n$
$\tilde{T}$	time constraint

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, etc.).

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 ith type of applications is λ_i, where $1\le i\le n$. The total task arrival rate is $\lambda=\lambda_1+\lambda_2+\cdots+\lambda_n$.

For the ith 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 ith 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 ith 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

$$\overline{x}=\frac{\lambda_1}{\lambda}\overline{x_1}+\frac{\lambda_2}{\lambda}\overline{x_2}+\cdots+\frac{\lambda_n}{\lambda}\overline{x_n}.$$

The utilization of the server is $\rho=\lambda\overline{x}=\lambda_1\overline{x_1}+\lambda_2\overline{x_2}+\cdots+\lambda_n\overline{x_n}$. It is noticed that the server utilization depends on the arrival rates, the execution requirements, and the execution speeds of all the n types of applications. The second moment of x (i.e., the mean of x^2) is

$$\overline{x^2}=\frac{\lambda_1}{\lambda}\overline{x_1^2}+\frac{\lambda_2}{\lambda}\overline{x_2^2}+\cdots+\frac{\lambda_n}{\lambda}\overline{x_n^2}.$$

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

$$W=\frac{\lambda\overline{x^2}}{2(1-\rho)}=\frac{\sigma}{2(1-\rho)}$$

where $\sigma=\lambda_1\overline{x_1^2}+\lambda_2\overline{x_2^2}+\cdots+\lambda_n\overline{x_n^2}$. The average response time of tasks of the ith type of applications is

$$T_i=\overline{x_i}+W=\overline{x_i}+\frac{\sigma}{2(1-\rho)}$$

which can be rewritten as

$$T_i=\overline{x_i}+\frac{\lambda_1\overline{x_1^2}+\lambda_2\overline{x_2^2}+\cdots+\lambda_n\overline{x_n^2}}{2(1-\lambda_1\overline{x_1}-\lambda_2\overline{x_2}-\cdots-\lambda_n\overline{x_n})}$$

and

$$T_i=\frac{\overline{r_i}}{s_i}+\frac{\lambda_1\overline{r_1^2}/s_1^2+\lambda_2\overline{r_2^2}/s_2^2+\cdots+\lambda_n\overline{r_n^2}/s_n^2}{2(1-\lambda_1\overline{r_1}/s_1-\lambda_2\overline{r_2}/s_2-\cdots-\lambda_n\overline{r_n}/s_n)}.$$

The average task response time of all tasks is

$$T=\sum_{i=1}^{n}\frac{\lambda_i}{\lambda}T_i=\frac{1}{\lambda}\sum_{i=1}^{n}\frac{\lambda_i\overline{r_i}}{s_i}+\frac{\sigma}{2(1-\rho)}$$

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

$$\rho=\lambda_1\frac{\overline{r_1}}{s_1}+\lambda_2\frac{\overline{r_2}}{s_2}+\cdots+\lambda_n\frac{\overline{r_n}}{s_n}$$

and

$$\sigma=\lambda_1\frac{\overline{r_1^2}}{s_1^2}+\lambda_2\frac{\overline{r_2^2}}{s_2^2}+\cdots+\lambda_n\frac{\overline{r_n^2}}{s_n^2}.$$

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

$$P=\sum_{i=1}^{n}\lambda_i\overline{x_i}s_i^{\alpha}+P^*=\sum_{i=1}^{n}\lambda_i\overline{r_i}s_i^{\alpha-1}+P^*.$$

(Note: This is the idle speed model in [16].)

IEEE Proof

IV. Power Constrained Performance Optimization

A. Optimal Speed Setting

Given task arrival rates $\lambda_1, \lambda_2, \ldots, \lambda_n$, expected task execution requirements $\overline{r_1}$, $\overline{r_2}$, …, $\overline{r_n}$, the second moments of task execution requirements $\overline{r_1^2}$, $\overline{r_2^2}$, …, $\overline{r_n^2}$, base power consumption P^*, and certain power supply $\tilde{P}$, 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 $\tilde{P}$.

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 $1 \le i \le n$, where ϕ is a Lagrange multiplier. Since

$$\begin{aligned}\frac{\partial T}{\partial s_i} = & -\frac{1}{\lambda} \cdot \frac{\lambda_i \overline{r_i}}{s_i^2} \\ & + \frac{1}{2}\left(\frac{1}{1-\rho}\left(-\frac{2\lambda_i \overline{r_i^2}}{s_i^3}\right) + \frac{\sigma}{(1-\rho)^2}\left(-\frac{\lambda_i \overline{r_i}}{s_i^2}\right)\right)\end{aligned}$$

and

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

we have

$$\begin{aligned}& -\frac{1}{\lambda} \cdot \frac{\lambda_i \overline{r_i}}{s_i^2} - \frac{1}{1-\rho} \cdot \frac{\lambda_i \overline{r_i^2}}{s_i^3} - \frac{\sigma}{2(1-\rho)^2} \cdot \frac{\lambda_i \overline{r_i}}{s_i^2} \\ & = \phi(\alpha - 1)\lambda_i \overline{r_i} s_i^{\alpha-2}\end{aligned}$$

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

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

or

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

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

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

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

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 $\overline{r_i^2}/\overline{r_i}$ are identical.

Proof: Notice that $\overline{r_i^2}/\overline{r_i}$ is the only unique term in F_i, for all $1 \le i \le n$. If all the $\overline{r_i^2}/\overline{r_i}$ are identical, we have $s_1 = s_2 = \cdots = s_n$. On the other hand, if $\overline{r_i^2}/\overline{r_i} \ne \overline{r_j^2}/\overline{r_j}$ for some i and j, then $s_i \ne s_j$.

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

$$\begin{aligned}F_0(\phi, s_1, \ldots, s_n) &= 0 \\ F_1(\phi, s_1, \ldots, s_n) &= 0 \\ &\vdots \\ F_n(\phi, s_1, \ldots, s_n) &= 0.\end{aligned}$$

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 $F_i(\phi, s_1, \ldots, s_n) = F_i(y_0, y_1, \ldots, y_n) = F_i(\mathbf{y})$, where $F_i : \mathbb{R}^{n+1} \to \mathbb{R}$ maps $(n+1)$-dimensional space $\mathbb{R}^{n+1}$ into the real line $\mathbb{R}$. Let us define a function $\mathbf{F} : \mathbb{R}^{n+1} \to \mathbb{R}^{n+1}$ 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,

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

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

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

$$J(\mathbf{y}) = \begin{bmatrix} \frac{\partial F_0(\mathbf{y})}{\partial y_0} & \frac{\partial F_0(\mathbf{y})}{\partial y_1} & \cdots & \frac{\partial F_0(\mathbf{y})}{\partial y_n} \\ \frac{\partial F_1(\mathbf{y})}{\partial y_0} & \frac{\partial F_1(\mathbf{y})}{\partial y_1} & \cdots & \frac{\partial F_1(\mathbf{y})}{\partial y_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_n(\mathbf{y})}{\partial y_0} & \frac{\partial F_n(\mathbf{y})}{\partial y_1} & \cdots & \frac{\partial F_n(\mathbf{y})}{\partial y_n} \end{bmatrix}.$$

We can calculate the various components of the above-mentioned matrix as follows. First, we have

$$\frac{\partial F_0(\mathbf{y})}{\partial y_0} = \frac{\partial F_0(\mathbf{y})}{\partial \phi} = 0$$

and

$$\frac{\partial F_0(\mathbf{y})}{\partial y_j} = \frac{\partial F_0(\mathbf{y})}{\partial s_j} = (\alpha - 1)\lambda_j \overline{r_j} s_j^{\alpha-2}$$

for all $1 \le j \le n$. Next, we have

$$\frac{\partial F_i(\mathbf{y})}{\partial y_0} = \frac{\partial F_i(\mathbf{y})}{\partial \phi} = (\alpha - 1)s_i^\alpha$$

IEEE Proof

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

$$\frac{\partial F_i(\mathbf{y})}{\partial y_i} = \frac{\partial F_i(\mathbf{y})}{\partial s_i} = \phi\alpha(\alpha-1)s_i^{\alpha-1} + \frac{\overline{r_i^2}}{\overline{r_i}}\left(\frac{1}{(1-\rho)^2}\left(-\frac{\lambda_i\overline{r_i}}{s_i^2}\right)\frac{1}{s_i} + \frac{1}{1-\rho}\left(-\frac{1}{s_i^2}\right)\right) + \frac{1}{2}\left(\frac{1}{(1-\rho)^2}\left(-\frac{2\lambda_i\overline{r_i^2}}{s_i^3}\right) + \frac{2\sigma}{(1-\rho)^3}\left(-\frac{\lambda_i\overline{r_i}}{s_i^2}\right)\right)$$

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

$$\frac{\partial F_i(\mathbf{y})}{\partial y_j} = \frac{\partial F_i(\mathbf{y})}{\partial s_j} = \frac{\overline{r_i^2}}{\overline{r_i}} \cdot \frac{1}{(1-\rho)^2}\left(-\frac{\lambda_j\overline{r_j}}{s_j^2}\right)\frac{1}{s_i} + \frac{1}{2}\left(\frac{1}{(1-\rho)^2}\left(-\frac{2\lambda_j\overline{r_j^2}}{s_j^3}\right) + \frac{2\sigma}{(1-\rho)^3}\left(-\frac{\lambda_j\overline{r_j}}{s_j^2}\right)\right)$$

for all $1 \le i \le n$ and all $1 \le j \ne i \le n$.

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

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

that is,

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

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

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

and ϵ 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 ϵ.

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.

Algorithm 1: Optimal Server Speed Setting.

Input: Parameters $\lambda_1, \lambda_2, ..., \lambda_n$, $\overline{r_1}, \overline{r_2}, ..., \overline{r_n}$, $\overline{r_1^2}, \overline{r_2^2}, ..., \overline{r_n^2}$, P^*, and $\tilde{P}$.
Output: An optimal server speed setting and ϕ, i.e., $\mathbf{y} = (\phi, s_1, ..., s_n)$, which satisfies $\mathbf{F}(\mathbf{y}) = \mathbf{0}$.

$\mathbf{y} \leftarrow (-1, s, ..., s)$; (1)
repeat (2)
  Calculate $J(\mathbf{y})$,
  where $J(\mathbf{y})_{i,j} = \partial F_i(\mathbf{y})/\partial y_j$ for $0 \le i, j \le n$; (3)
  Calculate $\mathbf{F}(\mathbf{y}) = (F_0(\mathbf{y}), F_1(\mathbf{y}), ..., F_n(\mathbf{y}))$; (4)
  Solve the linear system of equations
  $J(\mathbf{y})\mathbf{z} = -\mathbf{F}(\mathbf{y})$; (5)
  $\mathbf{y} \leftarrow \mathbf{y} + \mathbf{z}$; (6)
until $\|\mathbf{z}\| \le \epsilon$. (7)

For a constant speed server, i.e., $s_1 = s_2 = \cdots = s_n = s$, we have

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

The above-mentioned server speed yields

$$\rho = \left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)^{\alpha/(\alpha-1)}\left(\frac{1}{\tilde{P} - P^*}\right)^{1/(\alpha-1)}$$

and

$$\sigma = \left(\sum_{i=1}^{n}\lambda_1\overline{r_i^2}\right)\left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)^{2/(\alpha-1)}\left(\frac{1}{\tilde{P} - P^*}\right)^{2/(\alpha-1)}.$$

The average task response time of all tasks is

$$T = \frac{\rho}{\lambda} + \frac{\sigma}{2(1-\rho)}$$

which is

$$T = \frac{1}{\lambda}\left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)^{\alpha/(\alpha-1)}\left(\frac{1}{\tilde{P} - P^*}\right)^{1/(\alpha-1)} + \frac{\left(\sum_{i=1}^{n}\lambda_1\overline{r_i^2}\right)\left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)^{2/(\alpha-1)}\left(\frac{1}{\tilde{P}-P^*}\right)^{2/(\alpha-1)}}{2\left(1 - \left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)^{\alpha/(\alpha-1)}\left(\frac{1}{\tilde{P}-P^*}\right)^{1/(\alpha-1)}\right)}.$$

We consider a Pareto distribution [2] of r_i with pdf

$$\frac{\beta_i\tilde{r}_i^{\beta_i}}{r_i^{\beta_i+1}}$$

in the range $r_i \in [\tilde{r}_i, \infty)$, where $\tilde{r}_i \ge 0$ and $\beta_i > 2$. The expectation of r_i is

$$\overline{r_i} = \frac{\beta_i\tilde{r}_i}{\beta_i - 1}$$

IEEE Proof

and the second moment of r_i is

$$\overline{r_i^2} = \left(\frac{\beta_i}{\beta_i - 2}\right)\tilde{r}_i^2.$$

A nice feature of a Pareto distribution is that for any $\overline{r_i} > 0$ and $\overline{r_i^2} > \overline{r_i}^2$, there are $\tilde{r}_i > 0$ and $\beta_i > 2$, such that the expectation of r_i is $\overline{r_i}$ and the second moment of r_i is $\overline{r_i^2}$. Notice that

$$c_i = \frac{\overline{r_i^2}}{\overline{r_i}^2} = \frac{(\beta_i - 1)^2}{\beta_i(\beta_i - 2)} = 1 + \frac{1}{\beta_i(\beta_i - 2)}$$

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 β_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 β_i is known, $\tilde{r}_i$ can be determined as

$$\tilde{r}_i = \left(\frac{\beta_i - 1}{\beta_i}\right)\overline{r_i}.$$

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 $1 \le i \le n$. The expected task execution requirements are $\overline{r_i} = 1.2 - 0.2(i - 1)$, for all $1 \le i \le n$. The second moments of task execution requirements are $\overline{r_i^2} = 1.5 + 0.5(i - 1)$, for all $1 \le i \le n$. The base power consumption is $P^* = 10$. To ensure $\rho < 1$, we need

$$\tilde{P} > P^* + \left(\sum_{i=1}^{n} \lambda_i \overline{r_i}\right)^{\alpha}.$$

The given power supply is

$$\tilde{P} = P^* + (1 + 0.2b)\left(\sum_{i=1}^{n} \lambda_i \overline{r_i}\right)^{\alpha}.$$

Let $T_{\rm var}$ denote the average task response time with the optimal variable server speed setting, $T_{\rm con}$ denote the average task response time with the constant server speed setting. The relative difference between $T_{\rm var}$ and $T_{\rm con}$ is

$$\Delta_T = \left(\frac{T_{\rm con} - T_{\rm var}}{T_{\rm con}}\right) \times 100\%.$$

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 ρ, and the optimal average task response time $T_{\rm var}$. As comparison, we also show the constant speed s and the resulted server utilization ρ and average task response time $T_{\rm con}$. Finally, we give the relative difference Δ_T between $T_{\rm var}$ and $T_{\rm con}$.

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

In Fig. 2, we show the relative difference Δ_T between $T_{\rm var}$ and $T_{\rm con}$ 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

TABLE II
NUMERICAL DATA FOR POWER CONSTRAINED OPTIMIZATION

	$b=4$	$b=8$	$b=12$	$b=16$	$b=20$
$\tilde{P}$	49.5136000	67.0752000	84.6368000	102.1984000	119.7600000
s_1	3.3919967	3.9322954	4.4204710	4.8683928	5.2847059
s_2	3.4983475	4.1145223	4.6595711	5.1526454	5.6062546
s_3	3.6384553	4.3433847	4.9533109	5.4978463	5.9942584
s_4	3.8361407	4.6514946	5.3409733	5.9488753	6.4984547
s_5	4.1497472	5.1178076	5.9169350	6.6129478	7.2372215
s_6	4.7909610	6.0253151	7.0176979	7.8711810	8.6305931
ρ	0.7559313	0.6340678	0.5567680	0.5020824	0.4607612
$T_{\rm var}$	1.8390434	0.8972191	0.5978647	0.4521420	0.3661072
s	3.7565942	4.5148643	5.1629449	5.7382924	6.2609903
ρ	0.7453560	0.6201737	0.5423261	0.4879500	0.4472136
$T_{\rm con}$	2.0092226	0.9934964	0.6635606	0.5013570	0.4051139
Δ_T	8.4699031	9.6907596	9.9005139	9.8163635	9.6285648

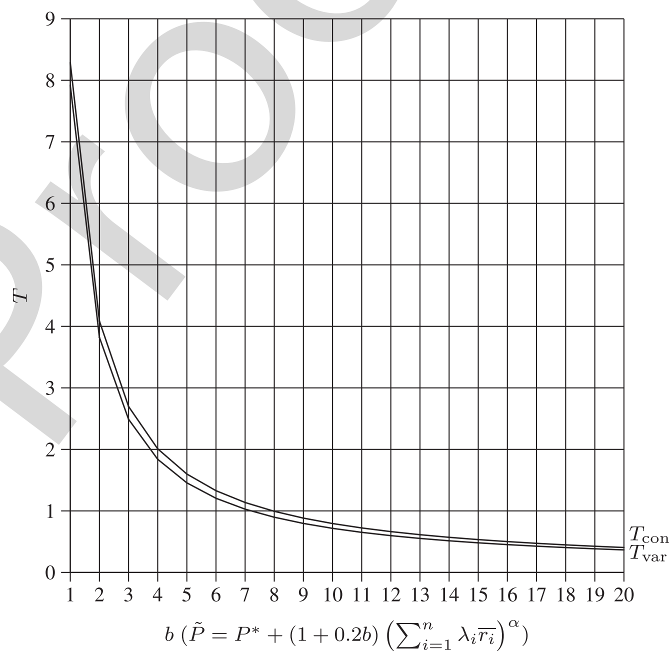

Fig. 1. Average task response time versus power supply.

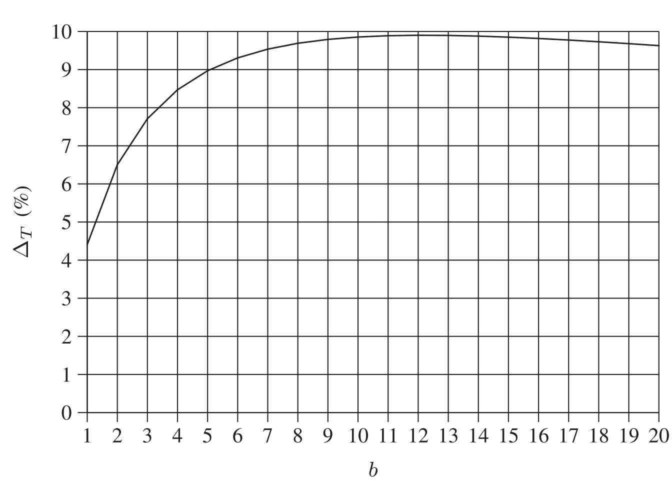

Fig. 2. Relative difference Δ_T between $T_{\rm var}$ and $T_{\rm con}$.

IEEE Proof

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 $T_{\rm var}$ and $T_{\rm con}$, 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 task execution requirements $\overline{r_1}$, $\overline{r_2}$,. . ., $\overline{r_n}$, the second moments of task execution requirements $\overline{r_1^2}$, $\overline{r_2^2}$,. . ., $\overline{r_n^2}$, base power consumption P^*, and certain quality of service $\tilde{T}$, our problem is to find server speeds $s_1, s_2, \ldots, s_n$, such that $T = \tilde{T}$, 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 $[P_{\rm lb}, P_{\rm ub}]$, where $P_{\rm lb}$ and $P_{\rm ub}$ 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 $\tilde{T}$. The value $P_{\rm lb}$ is chosen in such a way that when the server is given power supply $P_{\rm lb}$, the average task response time is greater than $\tilde{T}$. The value $P_{\rm ub}$ is chosen in such a way that when the server is given power supply $P_{\rm ub}$, the average task response time is less than $\tilde{T}$. 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 serve with a constant speed.

For a constant speed server, i.e., $s_1 = s_2 = \cdots = s_n = s$, we have

$$\frac{1}{\lambda s}\sum_{i=1}^{n}\lambda_i\overline{r_i} + \frac{1}{2s\left(s-\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)}\sum\nolimits_{i=1}^{n}\lambda_i\overline{r_i^2} = \tilde{T}.$$

The above-mentioned equation is actually a quadratic equation $2\tilde{T}s^2 - 2bs - c = 0$, where

$$b = \left(\tilde{T} + \frac{1}{\lambda}\right)\left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)$$

and

$$c = \sum_{i=1}^{n}\lambda_i\overline{r_i^2} - \frac{2}{\lambda}\left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)^2.$$

It is clear that

$$s = \frac{2b+\sqrt{4b^2+8\tilde{T}c}}{4\tilde{T}} = \frac{b+\sqrt{b^2+2\tilde{T}c}}{2\tilde{T}}$$

TABLE III
Numerical Data for Performance Constrained Optimization

	$b = 4$	$b = 8$	$b = 12$	$b = 16$	$b = 20$
$\tilde{T}$	1.2000000	2.4000000	3.6000000	4.8000000	6.0000000
s_1	3.6728137	3.2629304	3.1165324	3.0407684	2.9943405
s_2	3.8206132	3.3485019	3.1770230	3.0875972	3.0325554
s_3	4.0097190	3.4632989	3.2602903	3.1531054	3.0866063
s_4	4.2688202	3.6284473	3.3836464	3.2520687	3.1694096
s_5	4.6676909	3.8960336	3.5904651	3.4221005	3.3143226
s_6	5.4574749	4.4564372	4.0420459	3.8056090	3.6498308
ρ	0.6862822	0.7943325	0.8447458	0.8745980	0.8945243
$P_{\rm var}$	58.4027860	45.5883307	41.2403610	39.0241444	37.6731106
s	4.2711786	3.6211022	3.3747446	3.2432770	3.1611412
ρ	0.6555568	0.7732452	0.8296924	0.8633243	0.8857561
$P_{\rm con}$	61.0803072	46.7146674	41.8889237	39.4527683	37.9798784
Δ_P	4.3836079	2.4110986	1.5482915	1.0864229	0.8077114

where

$$b^2 + 2\tilde{T}c = \left(\tilde{T} - \frac{1}{\lambda}\right)^2\left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)^2 + 2\tilde{T}\left(\sum_{i=1}^{n}\lambda_i\overline{r_i^2}\right).$$

Therefore, we obtain

$$s = \frac{1}{2\tilde{T}}\left(\left(\tilde{T}+\frac{1}{\lambda}\right)\left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right) + \sqrt{\left(\tilde{T}-\frac{1}{\lambda}\right)^2\left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)^2 + 2\tilde{T}\left(\sum_{i=1}^{n}\lambda_i\overline{r_i^2}\right)}\right).$$

The average power consumption of the server is

$$P = \left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)s^{\alpha-1} + P^*$$

which is actually

$$P = \left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)\left(\frac{1}{2\tilde{T}}\left(\left(\tilde{T}+\frac{1}{\lambda}\right)\left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right) + \sqrt{\left(\tilde{T}-\frac{1}{\lambda}\right)^2\left(\sum_{i=1}^{n}\lambda_i\overline{r_i}\right)^2 + 2\tilde{T}\left(\sum_{i=1}^{n}\lambda_i\overline{r_i^2}\right)}\right)\right)^{\alpha-1} + P^*.$$

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

$$\Delta_P = \left(\frac{P_{\rm con} - P_{\rm var}}{P_{\rm con}}\right)\times 100\%.$$

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

In Table III, for b = 4, 8, 12, 16, 20, where b decides $\tilde{T}$, we display the time constraint $\tilde{T}$, the optimal server speed setting $s_1, s_2, s_3, s_4, s_5, s_6$, server utilization ρ, and the minimum average power consumption $P_{\rm var}$. As comparison, we also show

IEEE Proof

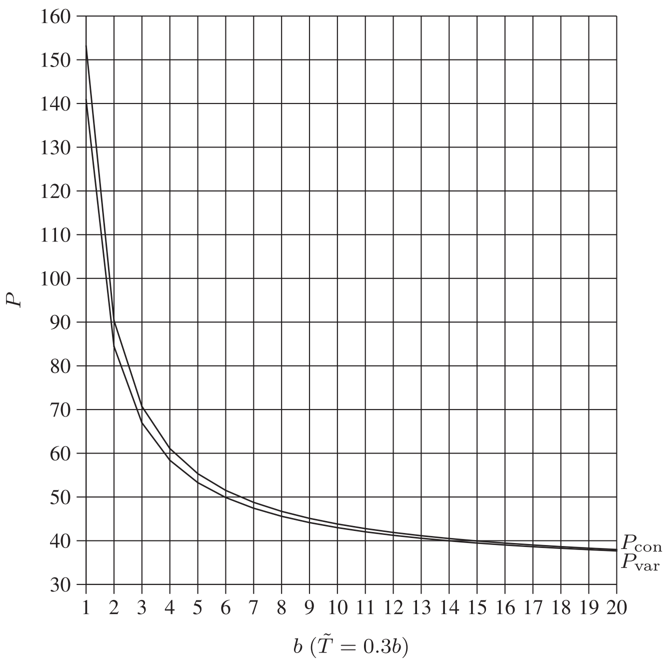

Fig. 3. Average power consumption versus quality of service.

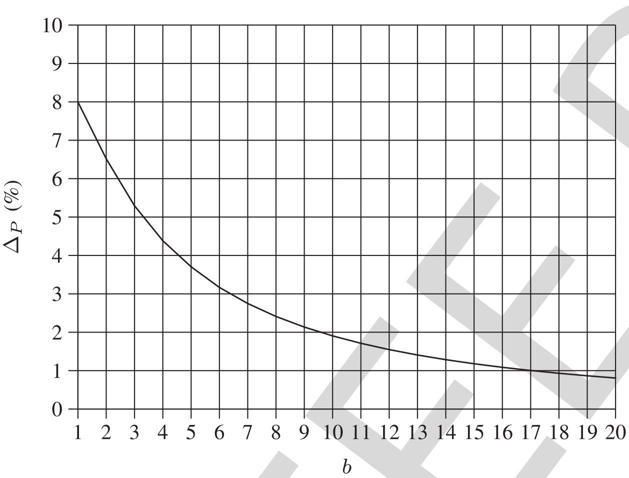

Fig. 4. Relative difference Δ_P between $P_{\rm var}$ and $P_{\rm con}$.

the constant speed s and the resulted server utilization ρ and average power consumption $P_{\rm con}$. Finally, we give the relative difference Δ_P between $P_{\rm var}$ and $P_{\rm con}$.

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

In Fig. 4, we show the relative difference Δ_P between $P_{\rm var}$ and $P_{\rm con}$ 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 $\tilde{T}$ 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 $P_{\rm var}$ and $P_{\rm con}$, which can be as high as 8.0%. In fact, it is unbounded as $\tilde{T} \to 0$.
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

[1] 2018. [Online]. Available: http://en.wikipedia.org/wiki/Dynamic_voltage_scaling
[2] 2018. [Online]. Available: http://en.wikipedia.org/wiki/Pareto_distribution
[3] W. L. Bircher and L. K. John, "Analysis of dynamic power management on multicore processors," in *Proc. 22nd ACM Int. Conf. Supercomput.*, 2008, pp. 327–338.
[4] R. L. Burden, J. D. Faires, and A. C. Reynolds, *Numerical Analysis*, 2nd ed. Boston, MA, USA: Prindle, Weber & Schmidt, 1981.
[5] R. Cochran, C. Hankendi, A. Coskun, and S. Reda, "Identifying the optimal energy-efficient operating points of parallel workloads," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2011, pp. 608–615.
[6] S. Dustdar, Y. Guo, B. Satzger, and H.-L. Truong, "Principles of elastic processes," *IEEE Int. Comput.*, vol. 15, no. 5, pp. 66–71, Sep./Oct. 2011.

IEEE Proof

[7] G. Galante and L. C. E. de Bona, "A survey on cloud computing elasticity," in *Proc. IEEE/ACM 5th Int. Conf. Utility Cloud Comput.*, 2012, pp. 263–270.

[8] N. R. Herbst, "Quantifying the impact of platform configuration space for elasticity benchmarking," Study thesis, Dept. Informats., Karlsruhe Inst. Technol., Karlsruhe, Germany, 2011.

[9] S. Huang and W. Feng, "Energy-efficient cluster computing via accurate workload characterization," in *Proc. 9th IEEE/ACM Int. Symp. Cluster Comput. Grid*, 2009, pp. 68–75.

[10] K. Hwang, X. Bai, Y. Shi, M. Li, W.-G. Chen, and Y. Wu, "Cloud performance modeling with benchmark evaluation of elastic scaling strategies," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 1, pp. 130–143, Jan. 2016.

[11] B. Kar, H. K. Wu, and Y. D. Lin, "Energy cost optimization in dynamic placement of virtualized network function chains," *IEEE Trans. Netw. Serv. Manage.*, vol. 15, no. 1, pp. 372–386, Mar. 2018.

[12] E. Kim, Y. Ko, and S. Ha, "An adaptive frames per second-based CPU-GPU cooperative dynamic voltage and frequency scaling governing technique for mobile games," *J. Low Power Electron.*, vol. 12, no. 4, pp. 309–322, 2016.

[13] L. Kleinrock, *Queueing Systems, Volume 1: Theory*. New York, NY, USA: Wiley, 1975.

[14] M. Kuperberg, N. Herbst, J. von Kistowski, and R. Reussner, "Defining and quantifying elasticity of resources in cloud computing and scalable platforms," Karlsruhe Inst. Technol., Karlsruhe, Germany, Rep. no. 16, Informat., 2011.

[15] S. J. Lee, H.-K. Lee, and P.-C. Yew, "Runtime performance projection model for dynamic power management," in *Proc. 12th Asia-Pac. Comput. Syst. Archit. Conf.*, 2007, pp. 186–197.

[16] K. Li, "Improving multicore server performance and reducing energy consumption by workload dependent dynamic power management," *IEEE Trans. Cloud Comput.*, vol. 4, no. 2, pp. 122–137, Apr./Jun. 2016.

[17] K. Li, "Optimal task dispatching for multiple heterogeneous multiserver systems with dynamic speed and power management," *IEEE Trans. Sustain. Comput.*, vol. 2, no. 2, pp. 167–182, 2017.

[18] K. Li, "Quantitative modeling and analytical calculation of elasticity in cloud computing," *IEEE Trans. Cloud Comput.*, to be published.

[19] K. Li, "Optimal load distribution for multiple classes of applications on heterogeneous servers with variable speeds," *J. Softw., Pract. Exp.*, to be published.

[20] D. C. Snowdon, E. Le Sueur, S. M. Petters, and G. Heiser, "Koala a platform for OS-level power management," in *Proc. 4th ACM Eur. Conf. Comput. Syst.*, 2009, pp. 289–302.

[21] P. Sobeslavsky, *Elasticity Cloud Comput.*, Master Thesis, Distributed, Embedded, Mobile, Interactive and Parallel Systems, Joseph Fourier University, Grenoble, France, 2011.

[22] J. Varia, "Architecting for the cloud: Best practices," Amazon, 2010. [Online]. Available: https://jineshvaria.s3.amazonaws.com/public/cloudbestpractices-jvaria.pdf

Keqin 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.

IEEE Proof