Cluster Comput (2014) 17:943–955
DOI 10.1007/s10586-013-0326-z

Managing performance and power consumption tradeoff for multiple heterogeneous servers in cloud computing

Yuan Tian · Chuang Lin · Keqin Li

Received: 7 August 2013 / Accepted: 5 October 2013 / Published online: 26 March 2014

Abstract There are typically multiple heterogeneous servers providing various services in cloud computing. High power consumption of these servers increases the cost of running a data center. Thus, there is a problem of reducing the power cost with tolerable performance degradation. In this paper, we optimize the performance and power consumption tradeoff for multiple heterogeneous servers. We consider the following problems: (1) optimal job scheduling with fixed service rates; (2) joint optimal service speed scaling and job scheduling. For problem (1), we present the Karush-Kuhn-Tucker (KKT) conditions and provide a closed-form solution. For problem (2), both continuous speed scaling and discrete speed scaling are considered. In discrete speed scaling, the feasible service rates are discrete and bounded. We formulate the problem as an MINLP problem and propose a distributed algorithm by online value iteration, which has lower complexity than a centralized algorithm. Our approach provides an analytical way to manage the tradeoff between performance and power consumption. The simulation results show the gain of using speed scaling, and also prove the effectiveness and efficiency of the proposed algorithms.

Y. Tian (✉) · C. Lin
Computer Science and Technology Department, Tsinghua University, Beijing, China
e-mail: tianyuan@csnet1.cs.tsinghua.edu.cn

C. Lin
e-mail: clin@csnet1.cs.tsinghua.edu.cn

K. Li
Department of Computer Science, State University of New York, New Paltz, NY 12561, USA
e-mail: lik@newpaltz.edu

Keywords Cloud computing · Convex optimization · Distributed algorithm · Heterogeneous server · Performance and power tradeoff · Power consumption · Response time

1 Introduction

A typical data center in cloud computing contains tens of thousands of servers. For instance, it is reported that Google has more than 900,000 servers, and the company recently revealed that a container data center holds more than 45,000 servers in a single facility built in 2005 [2]. With the rapid growth of data centers in both quantity and scale, the energy consumption for operating and cooling, directly related to the quantity of hosted servers and their workload, is increasing. It becomes a big challenge for data center owners, because of economical and environmental reasons [7, 11, 15]. On the other hand, the expectation of performance and quality of experience (QoE) for the services provided over the Internet has obviously grown. For instance, Google reports that an extra 0.5s in search page generation will lower user satisfaction, causing in turn a 20 % traffic drop [26]. As a result, all data center must consider both performance and the price of performance [5] to provide cloud computing services, that is, managing the tradeoff between performance metrics and energy cost.

Recently, there are a number of mechanisms proposed to address the problem, e.g., dynamic voltage and frequency scaling (DVFS) [19]. DVFS can dynamically scale the server speed by reducing the processor voltage and frequency when the load is light. Processors today are commonly equipped with the DVFS mechanism to reduce power consumption, such as Intel's Speed-Step technology [28] and AMD's Cool'n'Quiet technology [1]. With the currently available processor technology, the clock frequency and supply voltage can only be set with a few discrete values [13]. However, most of the recent researches model the adjustment of frequency and voltage continuously and unboundedly [23, 31]. In this paper, we will investigate discrete and bounded frequency and voltage adjustment, which will be practically more useful.

Traditional load balancing mechanisms assign loads to the server which has the maximum processing capacity to achieve better performance. These mechanisms do not consider power cost and have poor power efficiency. Thus, we should consider the tradeoff between energy cost and performance metrics. In this paper, we focus on the problem of optimal dynamic speed scaling and job scheduling for multiple heterogeneous servers. Our purpose is to optimize the performance and power consumption tradeoff.

We consider the following problems: (1) optimal job scheduling with fixed service rates; (2) joint optimal service speed scaling and job scheduling. These two problems can be abstracted as convex optimization with a linear constraint. For problem (1), we present the Karush-Kuhn-Tucker (KKT) conditions and provide a closed-form solution. For problem (2), both continuous speed scaling and discrete speed scaling are considered. In discrete speed scaling, the feasible service rates can be discrete and bounded. We formulate this problem as an MINLP problem and propose a distributed algorithm by online value iteration, which has lower complexity than a centralized algorithm. For solving the discrete speed scaling problem with discrete solutions in certain range, we relax the discrete constraint to continuous values in the same range, that is, to solve the continuous speed scaling problem first. The simulation results show the gain of using speed scaling, and also prove the effectiveness and efficiency of the proposed algorithms.

The rest of the paper is organized as follows. The next section briefly reviews some related work. In Sect. 3, we introduce the performance model in terms of response time and the power function which is related to the service rate. Section 4 proposes the optimal job scheduling policy for servers without DVFS. In Sect. 5, we formulate the problem of optimal job scheduling together with dynamic speed scaling for servers with DVFS, and propose a distributed algorithm by online value iteration. In Sect. 6, we present numerical examples to illustrate the analysis method. Finally, we conclude the paper in Sect. 7.

2 Related work

Reducing energy consumption in data centers has been an important research issue recently. The fundamental principle to achieve energy efficiency is to make energy consumption proportional to system utilization [6]. These energy-proportional methods can be implemented at various levels.

At the server level, we have DVFS or speed scaling [4, 14, 17, 18, 20, 32, 33]. A static speed scaling policy is the simplest nontrivial speed scaling method [10]. It usually uses one or more thresholds to determine when to change the server speed during the process of service [31].

A dynamic speed scaling policy design can be more flexible and highly sophisticated. Reference [3] studied speed scaling methods to minimize a weighted sum of response time and energy consumption, and proved that a popular dynamic speed scaling algorithm is 2-competitive for this objective. In [24], the author studied dynamically scaling the server speed according to power allocated, assuming that the processor frequency and supply voltage can change continuously and unboundedly.

At the data center level, there are lots of energy-aware load balancing methods to consider the tradeoff between performance and energy consumption [9, 23, 24, 30]. There are different considerations in dealing with the power-performance tradeoff. One consideration is to optimize the performance under certain energy consumption constraint, which is more adaptive in energy-restricted systems. Reference [13] assumes that a server farm has a fixed peak power budget, and distributes the available power among servers so as to get maximum performance in a variety of scenarios.

Another consideration is to minimize energy consumption while meeting certain performance goal, so as to cut the electricity bill in large-scale servers. In [34–37], the authors addressed optimal performance constrained power minimization in scheduling parallel workloads on a server cluster, such that the proposed optimization model can provide accurate control of power consumption while meeting the QoS. In [25], the author considered minimizing energy consumption with schedule length constraint. Reference [21] addresses the problem of scheduling precedence-constrained parallel applications on multiprocessors and minimizing processor energy while meeting deadlines for task execution.

The interaction between load balancing and speed scaling is also studied. Reference [10] considers static speed scaling and shows that if the heterogeneity of a system is small, the design of load balancing and speed scaling can be decoupled.

The last consideration is the joint optimization of energy consumption and performance. Reference [32] minimizes a weighted sum of mean response time and energy consumption under processor sharing scheduling. In [16], the authors minimized the energy consumption and the makespan while meeting task deadlines and architectural requirements.

In this paper, we take the approach of minimizing a weighted sum of power consumption and performance by both load balancing and speed scaling. We will investigate both continuous and discrete speed scaling. In discrete speed scaling, the speed can only be set with some discrete levels with an upper and a lower bound.

3 The models

3.1 Performance model

Assume that we have N heterogeneous servers, and each server has its own service rate. We assume that the arrival of the jobs conforms to a Poisson process with rate λ. The interval arrival times of Poisson arrival tasks conform to an exponential distribution. According to the additive property of exponential distributions, the jobs assigned to server i is also a Poisson steam with arrival rate λ_i, and $\lambda_1 + \lambda_2 + \cdots + \lambda_N = \lambda$. We model server i with a local queue as an M/G/1 queuing model. Let μ_i be the service rate of server i. Then, the server utilization is $\rho_i = \lambda_i/\mu_i$. The expected value of service time is $\bar{t}_i$ and the coefficient of variation of service time is C_i. Hence, by using the well known Pollaczek-Khinchin mean-value formula, we get the average service time of server i as

$$T_i = \left(1 + \frac{1+C_i^2}{2} \cdot \frac{\rho_i}{1-\rho_i}\right)\bar{t}_i$$
$$= \frac{1}{\mu_i} + \frac{1+C_i^2}{2} \cdot \frac{\lambda_i}{\mu_i(\mu_i - \lambda_i)}. \tag{1}$$

Therefore, the expected response time in the data center with N servers can be expressed as

$$T = \sum_{i=1}^{N} \left(\frac{\lambda_i}{\lambda}\right) T_i. \tag{2}$$

3.2 Power model

The modeling of the power function $P(s)$ of service rate s is an open topic. Many researches show different forms depending on specific systems. According to the data provided by Intel Labs [27], the processor uses the main part of the power consumed by a server. Thus, we characterize the server power consumption by two parts, i.e., the dynamic power consumption generated by the workload running on the server, and the static power consumption independent of the workload.

Dynamic power consumption is created by circuit activity and depends mainly on utilization scenario and clock rate [8], which is approximately $p_d = aCV^2 f$, where a is the switching activity, C is the physical capacitance, V is the supply voltage, and f is the clock frequency. Since $s \propto f$ and $f \propto V$, which implies that $p_d \propto s^\alpha$, where α is around 3 [12], we model dynamic power consumption approximately as $p_d = ks^\alpha$, where k is some constant.

The static power consumption is the power consumption when a server is idle, which is caused by leakage currents

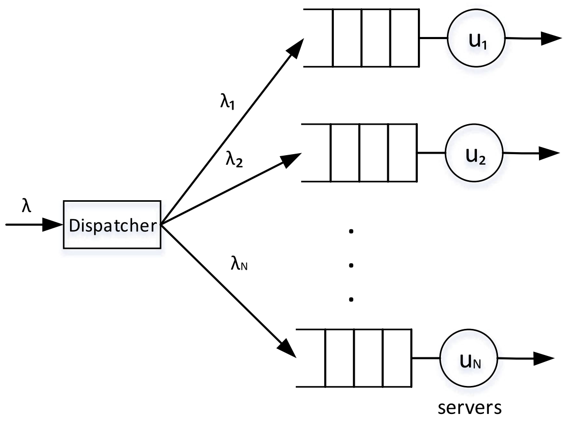

Fig. 1 A diagram of multiservers

independent of clock rate and utilization scenario. Thus, we characterize the power consumption of server i as

$$P_i = \rho_i k_i \mu_i^{\alpha_i} + P_i^*, \tag{3}$$

where ρ_i is the utilization of server i, and P_i^* is the static power consumption.

4 Multiservers without DVFS

We consider N heterogeneous servers with a load dispatcher in Fig. 1, which schedules arrival jobs to servers according to certain metric goal. The metric we choose in this paper is a weighted sum of response time and power consumption cost. Given arrival rate λ, the dispatcher will route a Poisson stream λ_i to server i according to a scheduling policy $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_N)$ to minimize the following metric:

$$f(\boldsymbol{\lambda}) = \sum_{i=1}^{N} \left(\frac{\lambda_i}{\lambda}\right) T_i + \beta \sum_{i=1}^{N} P_i, \tag{4}$$

where $\beta \geq 0$ is used to characterize the tradeoff between power cost and job response time. The larger the value of β, the higher the weight of power cost, which will result in more degradation of response time.

4.1 Problem formulation

Our optimization problem is defined as follows: given job arrival rate λ and service rates $(\mu_1, \ldots, \mu_N)$ for N servers, find the optimal scheduling policy $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_N)$, which minimizes the metric in (4), subject to the equilibrium constraint of arrival steams. In order to ensure stability, we assume $\lambda_i < \mu_i$ for all $i = 1, \ldots, N$, and $T_i = \infty$ when $\lambda_i \geq \mu_i$. The servers are entirely heterogeneous in terms of service rate μ_i and power model P_i with different coefficient k_i, exponent α_i, and static power consumption P_i^*.

Formally, our optimization problem is to find

$$\begin{aligned}
&\min_{\boldsymbol{\lambda}} \quad \left(f(\boldsymbol{\lambda}) = \sum_{i=1}^{N}\left(\frac{\lambda_i}{\lambda}\right)T_i + \beta\sum_{i=1}^{N}P_i\right),\\
&\text{s.t.} \quad \sum_{i=1}^{N}\lambda_i = \lambda;\\
&\qquad\ \lambda_i \ge 0,\ \forall i = 1,\dots,N;\\
&\qquad\ \mu_i - \lambda_i > 0,\ \forall i = 1,\dots,N.
\end{aligned} \tag{5}$$

4.2 Solution methodology

The metric goal in our optimization is defined as $f(\boldsymbol{\lambda}) = \sum_{i=1}^{N} f_i(\lambda_i)$, where $f_i(\lambda_i)$ is

$$\begin{aligned}
f_i(\lambda_i) = {} & \frac{\lambda_i}{\lambda}\left(\frac{1}{\mu_i} + \frac{1+C_i^2}{2}\cdot\frac{\lambda_i}{\mu_i(\mu_i-\lambda_i)}\right)\\
& + \beta\left(\lambda_i k_i \mu_i^{\alpha_i-1} + P_i^*\right).
\end{aligned} \tag{6}$$

The Lagrange duality can relax the original problem (5) by transferring the constraints to the objective in the form of a weighted sum. Thus, the Karush-Kuhn-Tucker (KKT) conditions of (5) are given as

$$\begin{cases}
\text{\\ Lagrangian stationarity}\\
\nabla\sum_{i=1}^{N} f_i(\lambda_i) - \upsilon\nabla(\sum_{i=1}^{N}\lambda_i - \lambda)\\
\quad -\sum_{i=1}^{N}\omega_i\nabla(\mu_i-\lambda_i) - o_i\sum_{i=1}^{N}\lambda_i = 0;\\
\text{\\ Complementary slackness}\\
\omega_i(\lambda_i-\mu_i) = 0, \quad \forall i\in\{1,\dots,N\};\\
o_i\lambda_i = 0, \quad \forall i\in\{1,\dots,N\};\\
\text{\\ Dual feasibility}\\
\upsilon,\omega_i,o_i \ge 0, \quad \forall i\in\{1,\dots,N\};\\
\text{\\ Primal feasibility}\\
\mu_i-\lambda_i > 0, \quad \forall i\in\{1,\dots,N\};\\
\lambda_i \ge 0, \quad \forall i\in\{1,\dots,N\};\\
\sum_{i=1}^{N}\lambda_i - \lambda = 0;
\end{cases} \tag{7}$$

where υ and ω_i are Lagrange multipliers. Notice that

$$\frac{\partial f_i(\lambda_i)}{\partial\lambda_i} = \frac{1}{\lambda\mu_i} + \frac{1+C_i^2}{2\mu_i\lambda}\cdot\frac{2\mu_i\lambda_i-\lambda_i^2}{(\mu_i-\lambda_i)^2} + \beta k_i\mu_i^{\alpha_i-1}. \tag{8}$$

Also, we have

$$\begin{aligned}
&\frac{\partial\upsilon(\sum_{i=1}^{N}\lambda_i-\lambda)}{\partial\lambda_i} = \upsilon,\\
&\frac{\partial o_i(\sum_{i=1}^{N}\lambda_i)}{\partial\lambda_i} = o_i.
\end{aligned} \tag{9}$$

Thus, in addition to $\lambda_1+\lambda_2+\cdots+\lambda_N = \lambda$, for all $i\in\{1,\dots,N\}$, we have N nonlinear equations, $2N$ linear equations, and $4N+1$ linear inequalities in (10).

$$\begin{cases}
\frac{1}{\lambda\mu_i} + \frac{1+C_i^2}{2\mu_i\lambda}\cdot\frac{2\mu_i\lambda_i-\lambda_i^2}{(\mu_i-\lambda_i)^2} + \beta k_i\mu_i^{\alpha_i-1} - \upsilon + \omega_i = 0;\\
\omega_i(\mu_i-\lambda_i) = 0;\\
o_i\lambda_i = 0;\\
\mu_i-\lambda_i > 0;\\
\lambda_i \ge 0;\\
\upsilon,\omega_i,o_i \ge 0.
\end{cases} \tag{10}$$

Because $\mu_i-\lambda_i>0$, we can get $\omega_i = 0$ and eliminate N linear equations and N linear inequalities. By solving the quadratic equations of λ_i, we can formulate λ_i as a function of υ. From the equation $\sum_{i=1}^{N}\lambda_i = \lambda$, we can get the value of υ. Then, the variable λ_i can be obtained.

We consider a closed-form of (10) for a special case when $C_i = 1$, e.g., M/M/1 queueing system. The first equation of (10) can be written as,

$$2\mu_i\lambda_i - \lambda_i^2 - \left(\upsilon - o_i - \beta k_i\mu_i^{\alpha_i-1}\right)(\mu_i-\lambda_i) = 0. \tag{11}$$

From $o_i\lambda_i = 0$, we discuss the possible conditions.

1. If $o_i = 0$ and $\lambda_i \ge 0$, we get

$$\begin{aligned}
&\lambda_i = \mu_i - \sqrt{\frac{\mu_i}{\upsilon-\beta k_i\mu_i^{\alpha_i-1}}};\\
&\upsilon \ge \frac{\lambda\beta k\mu_i^{\alpha_i}+1}{\lambda\mu_i}.
\end{aligned} \tag{12}$$

2. If $o_i \ne 0$ and $\lambda_i = 0$, from $o_i \ge 0$, we get

$$\upsilon < \frac{\lambda\beta k\mu_i^{\alpha_i}+1}{\lambda\mu_i}. \tag{13}$$

Then, we get the λ_i as a function of υ,

$$\lambda_i(\upsilon) = \begin{cases}
\mu_i - \sqrt{\frac{\mu_i}{\upsilon-\beta k_i\mu_i^{\alpha_i-1}}}, & \text{if } \upsilon \ge \frac{\lambda\beta k\mu_i^{\alpha_i}+1}{\lambda\mu_i};\\
0, & \text{if } \upsilon < \frac{\lambda\beta k\mu_i^{\alpha_i}+1}{\lambda\mu_i}.
\end{cases} \tag{14}$$

It is unlikely that the nonlinear equation $\sum_{i=1}^{N}\lambda_i = \lambda$ by considering (14) has a closed-form solution. Due to the fact that $\lambda_i(\upsilon)$ is an increasing function of υ in the domain $[0,\infty)$, we can use the binary search algorithm to find a numerical solution $(\upsilon,\lambda_1,\lambda_2,\dots,\lambda_N)$.

From $\lambda_i \le \lambda$, we can derive that υ has an upper bound. For each i, υ satisfies

$$\upsilon \le \beta k\mu_i^{\alpha-1} + \frac{\mu_i}{(\mu_i-\lambda_i)^2}. \tag{15}$$

Algorithm 1 Binary search algorithm

```
Input:
    $left = 0$: initial left side of search domain;
    $right = \upsilon^{UB}$: initial right side of search domain.
Output:
    $\upsilon$: the output of binary search.
1:  while $right \geq left$ do
2:      set $\upsilon = (left + right)/2$;
3:      if $|\lambda - \sum_{i=1}^{N} \lambda_i(\upsilon)| \leq \epsilon$ then
4:          return $\upsilon$;
5:      end if;
6:      if $\lambda - \sum_{i=1}^{N} \lambda_i(\upsilon) < 0$ then
7:          $right = \upsilon$;
8:      else
9:          $left = \upsilon$;
10:     end if;
11: end while.
```

Thus, the upper bound of υ can be written as

$$\upsilon^{UB} = \max_{i \in [1..N]} \left(\beta k \mu_i^{\alpha-1} + \frac{\mu_i}{(\mu_i - \lambda_i)^2} \right). \tag{16}$$

The binary search algorithm is shown in Algorithm 1.

Especially, when $\beta = 0$, we get λ_i in the following form:

$$\lambda_i = \frac{\mu_i}{\sum_{1 \leq i \leq N} \mu_i} \lambda. \tag{17}$$

The following theorem shows the effectiveness of our method.

Theorem 1 *The problem in* (5) *is a convex optimization problem, and the* λ^* *which satisfies the KKT conditions is the global minimum.*

Proof The first-order derivative of $f_i(\lambda_i)$ is shown in (8). Under the condition $\mu_i - \lambda_i > 0$, we can get $f_i'(\lambda_i) > 0$. The second-order derivative satisfies

$$f_i''(\lambda_i) = \frac{\partial^2 f_i(\lambda_i)}{\partial^2 \lambda_i} = \frac{1 + C_i^2}{2\lambda\mu_i} \cdot \frac{2\mu_i^2}{(\mu_i - \lambda_i)^3} > 0. \tag{18}$$

We can derive that $f_i(\lambda_i)$ is a strictly convex function, due to the additivity of convex functions. The objective function $f(\boldsymbol{\lambda}) = \sum_{i=1}^{N} f_i(\lambda_i)$ is also a convex function. Also, we can observe that the inequality constraint functions are convex and the equality constraint functions are linear. Thus, the problem in (5) is a convex optimization problem. According to the role of the Karush-Kuhn-Tucker (KKT) conditions in providing necessary and sufficient conditions for optimality of a convex optimization problem, the local optimal λ^* is also the global minimum. □

5 Multiservers with DVFS

We consider the case when all the N heterogeneous servers in Sect. 3 can dynamically scale their service rates with DVFS. The higher the service rate, the lower the response time and the higher the power consumption. Thus, in addition to the job scheduling problem, we need to find the optimal speed scaling policy to balance the performance and power consumption tradeoff.

In this section, we study the job scheduling and speed scaling problem to find the optimal $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_N)$ and $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_N)$. Assume that each server has a maximum service rate μ_i^{MAX} and a minimum service rate μ_i^{MIN} it can achieve, which differ for heterogeneous servers. With the technology of DVFS, server i can scale its rate in the range of $[\mu_i^{MIN}, \mu_i^{MAX}]$. Some researches assume that the rate can be scaled continuously, in which the rate can be set with any point in this range. Thus, μ_i is a real number and we will discuss this case in this section as well. However, continuous speed scaling is impractical. Due to the current limited processor technology, a server cannot continuously scale its speed. Instead, a server will allow for some discrete levels which correspond to different proportions of the maximum server speed. Thus, server i will only choose certain service rate from a finite discrete set $F_i = \{\mu_i^1, \ldots, \mu_i^{M_i}\}$, where M_i is the number of feasible service rates.

In addition, with the increase of variable N indicating the number of servers, the problem will rapidly scale up and correspondingly puts forward higher requirements on the handling ability of a centralized load dispatch controller. Thus, we propose a distributed algorithm and spread the computing load on each server. With sufficient online value iteration, the objective function can converge to the optimal.

5.1 Continuous speed scaling

The metric is the same as that in Sect. 4, which is a weighted sum of response time and power consumption cost. Let $\boldsymbol{\lambda} = \{\lambda_1, \ldots, \lambda_N\}$ and $\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_N\}$ be two vectors. The metric goal in our optimization problem is given in (19):

$$f(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \sum_{i=1}^{N} \left(\frac{\lambda_i}{\lambda} \right) T_i + \beta \sum_{i=1}^{N} P_i. \tag{19}$$

We assume that

$$\begin{aligned} f_i(\lambda_i, \mu_i) &= \left(\frac{\lambda_i}{\lambda} \right) T_i + \beta P_i \\ &= \frac{\lambda_i}{\lambda} \left(\frac{1}{\mu_i} + \frac{1 + C_i^2}{2} \cdot \frac{\lambda_i}{\mu_i(\mu_i - \lambda_i)} \right) \\ &\quad + \beta \left(\lambda_i k_i \mu_i^{\alpha_i - 1} + P^* \right). \end{aligned} \tag{20}$$

Our optimization problem is to find the optimal job scheduling policy $\boldsymbol{\lambda}$ together with the optimal speed scaling method $\boldsymbol{\mu}$. It can be specified as (i) a global scheduling algorithm to dispatch jobs to servers (i.e., the arrival rate λ_i to server i), and (ii) a speed μ_i in the range of $[\mu_i^{MIN}, \mu_i^{MAX}]$ for each server i, to minimize the metric in (19), subject to the equilibrium constraint of arrival steam, i.e., $\sum_{i=1}^{N} \lambda_i = \lambda$. Also, we assume that $\lambda_i < \mu_i$ for all $i = 1, \ldots, N$, Formally, our optimization problem is to find

$$\begin{aligned} C0:\quad & \min_{\boldsymbol{\lambda},\boldsymbol{\mu}} \left(\sum_{i=1}^{N} f_i(\lambda_i, \mu_i) \right), \\ & \text{s.t.} \quad \sum_{i=1}^{N} \lambda_i = \lambda; \\ & \qquad \mu_i^{MIN} \le \mu_i \le \mu_i^{MAX}, \quad \forall i = 1, \ldots, N; \\ & \qquad \mu_i - \lambda_i > 0, \quad \forall i = 1, \ldots, N; \\ & \qquad \lambda_i \ge 0, \quad \forall i = 1, \ldots, N. \end{aligned} \tag{21}$$

The Karush-Kuhn-Tucker (KKT) conditions of (21) are given as follows,

$$\begin{cases} \nabla \sum_{i=1}^{N} f_i(\lambda_i, \mu_i) - \upsilon \nabla (\sum_{i=1}^{N} \lambda_i - \lambda) \\ \quad - \sum_{i=1}^{N} \omega 1_i \nabla (\mu_i - \lambda_i) = 0; \\ \sum_{i=1}^{N} \omega 2_i \nabla (\mu_i^{MAX} - \mu_i) - \sum_{i=1}^{N} \omega 3_i \nabla \lambda_i \\ \quad - \sum_{i=1}^{N} \omega 4_i \nabla (\mu_i - \mu_i^{MIN}) = 0; \\ \omega 1_i (\mu_i - \lambda_i) = 0; \\ \omega 2_i (\mu_i^{MAX} - \mu_i) = 0; \\ \omega 3_i \lambda_i = 0; \\ \omega 4_i (\mu_i - \mu_i^{MIN}) = 0; \\ \sum_{i=1}^{N} \lambda_i - \lambda = 0; \\ \mu_i - \lambda_i > 0; \\ \mu_i^{MAX} - \mu_i \ge 0; \\ \lambda_i \ge 0; \\ \omega 1_i, \omega 2_i, \omega 3_i, \omega 4_i \ge 0; \end{cases} \tag{22}$$

where $\omega 1_i, \omega 2_i, \omega 3_i, \omega 4_i$ are Lagrange multipliers. This equation set may have several solutions which are the local optimal solutions. We choose the minimum among these solutions as the global optimal. It is unlikely to get a closed-form solution of the nonlinear equations. However, we can use mathematical tools to get numerical solutions.

5.2 Discrete speed scaling

Discrete speed scaling will restrict the service rate μ_i to some discrete value in a set $F_i = \{\mu_i^1, \ldots, \mu_i^{M_i}\}$. Thus, our optimization problem with discrete speed scaling can be derived from the problem in (21), shown as follows:

$$\begin{aligned} D0:\quad & \min_{\boldsymbol{\lambda},\boldsymbol{\mu}} \left(\sum_{i=1}^{N} f_i(\lambda_i, \mu_i) \right), \\ & \text{s.t.} \quad \sum_{i=1}^{N} \lambda_i = \lambda; \\ & \qquad \mu_i \in \left\{\mu_i^1, \ldots, \mu_i^{M_i}\right\}, \quad \forall i = 1, \ldots, N; \\ & \qquad \mu_i - \lambda_i > 0, \quad \forall i = 1, \ldots, N; \\ & \qquad \lambda_i \ge 0, \quad \forall i = 1, \ldots, N. \end{aligned} \tag{23}$$

We can see that the problem in (21) is the relaxed problem in (23) with variables μ_i relaxed.

The problem $D0$ can be seen as a mixed integer nonlinear programming problem (MINLP), with N real variables λ_i and N integer (discrete) variables μ_i. Fundamental algorithms for solving MINLP are often built by combining existing algorithms from linear programming (LP), mixed integer programming (MIP), and nonlinear programming (NLP) [22], e.g., branch-and-bound, generalized benders decomposition, and outer-approximation. However, the algorithm complexity of MINLP is much higher than any of the LP, MIP, and NLP algorithms. Furthermore, with the increase of variable N, the problem scales up fast and consequently puts forward higher requirements on the handling ability of a centralized load dispatch controller. Thus, these algorithms degrade the performance and have weak robustness especially under burst traffic.

5.3 Dual decomposition based distributed algorithm

In this section, we propose a distributed algorithm by online value iteration instead of the existing centralized algorithms. The distributed algorithm can adapt to both continuous and discrete speed scaling. The obvious benefit of a distributed algorithm is to decompose the problem with $2N$ variables into N subproblems with 2 variables, where each subproblem can be solved much more easily in each server instead of centralized control.

Consider the problem $D0$ in (23). The objective is to minimize the sum of a weighted sum of response time and power cost, with the N real constraints coupled, which readily presents some decomposition possibilities.

We use a dual decomposition approach [29] to solve problem (23), and relax all the coupled constrains. We formalize the Lagrangian associated with problem (23) as follows,

$$L0:\quad \min_{\boldsymbol{\lambda},\boldsymbol{\mu}} \left(\sum_{i=1}^{N} f_i(\lambda_i,\gamma_i)+\nu\left(\sum_{i=1}^{N}\lambda_i-\lambda\right)\right),$$
$$\text{s.t.}\quad \mu_i-\lambda_i>0,\quad \forall i=1,\dots,N;$$
$$\mu_i\in\{\mu_i^1,\dots,\mu_i^{M_i}\},\quad \forall i=1,\dots,N;$$
$$\lambda_i\ge 0,\quad \forall i=1,\dots,N; \tag{24}$$

where ν is the Lagrange multiplier, which relaxes the original problem (23) by transferring the coupled constraints $\sum_{i=1}^{N}\lambda_i=\lambda$ to the objective function. The objective function can be represented as

$$L(\boldsymbol{\lambda},\boldsymbol{\mu},\nu)=\sum_{i=1}^{N} f_i(\lambda_i,\mu_i)+\nu\left(\lambda-\sum_{i=1}^{N}\lambda_i\right). \tag{25}$$

After relaxation, the original problem (23) is decomposed into distributively decoupled solvable subproblems, and the optimization is separated into two levels of optimization, which are then coordinated by a high-level master problem. At the lower level, we have the subproblems in which for each i:

$$L_i(\lambda_i,\mu_i,\nu)=f_i(\lambda_i,\mu_i)-\nu\lambda_i. \tag{26}$$

At the higher level, Lagrange duality ν links the original minimization problem (23), termed primal problem, with a dual maximization problem. The dual objective $g(\nu)$ is defined as the minimum value of the Lagrangian over (λ_i,μ_i),

$$g(\nu)=\inf_{(\lambda_i,\mu_i)} L_i(\lambda_i,\mu_i,\nu). \tag{27}$$

$g(\nu)$ is always concave even if the original problem is not convex, because it is the pointwise infimum of a family of affine functions of ν. The dual function can be maximized to obtain a lower bound on the optimal value $f_i^\star$ of the original problem (23). Thus, we have the master dual problem in charge of updating the dual variable ν by solving the dual problem,

$$\max\left(g(\nu)=\sum_{i=1}^{N} g_i(\nu)+\nu\lambda\right), \tag{28}$$

where $g_i(\nu)$ is the dual function obtained as the maximum value of the Lagrangian solved in (23) for a given ν. This problem is always a convex optimization problem even if the original problem is not convex [29]. Given the current Lagrange multiplier ν, we can get $\lambda_i^*(\nu)$, $\mu_i^*(\nu)$ by solve the subproblem of

$$Li:\quad \min\ \big(L_i(\lambda_i,\mu_i,\nu)\big),$$
$$\text{s.t.}\quad \mu_i-\lambda_i>0;$$
$$\mu_i\in\{\mu_i^1,\dots,\mu_i^{M_i}\},\quad \forall i=1,\dots,N;$$
$$\lambda_i\ge 0. \tag{29}$$

Algorithm 2 Distributed algorithm

```
Input:
    $L_i$: Lagrangian function for each server;
    $\lambda$: total arrival rate.
Output:
    $\lambda_i^*(\nu(t))$: load assigned to server $i$;
    $\mu_i^*(\nu(t))$: service rate of server $i$.
1: Set $t=0$ and $\nu(0)$ equal to some nonnegative value;
2: while true do
3:     Each server locally solves its problem by computing (26) and then gets the solution $\lambda_i^*(\nu(t))$ and $\mu_i^*(\nu(t))$;
4:     Each sever updates its service rate with the new $\mu_i^*(\nu(t))$;
5:     The load dispatcher implements the new dispatch policy $\boldsymbol{\lambda}^*(\nu(t))$;
6:     The load dispatcher updates the Lagrange duality variable $\nu$ with the gradient iterate (13) and gets the new $\nu(t+1)$;
7:     Set $t\leftarrow t+1$;
8: end while.
```

Thus, the original problem which N coupling non-negative real variables and N non-negative integer variables is decomposed into N decoupled subproblems, where each subproblem has one non-negative real variable and one non-negative integer variable which are decoupled. So we can solve each subproblem in parallel at each server node.

The dual function is differentiable. It can be solved by the following gradient method,

$$\nu(t+1)=\left[\nu(t)+\delta\left(\lambda-\sum_i\lambda_i^*\big(\nu(t)\big)\right)\right]^+, \tag{30}$$

where t is the iteration index, δ is a positive scalar step-size, $[\cdot]^+$ denotes the projection onto the set R^+ of non-negative real numbers.

For continuous speed scaling, the Lagrangian associated with problem (21) is the relaxed problem of (23), and the subproblem is the relaxed problem of (26). That is to say, for solving the discrete speed scaling with discrete solutions in range $\{\mu_i^1,\dots,\mu_i^{M_i}\}$, we will preliminary relax the discrete constraint to continuous values in the same range. We now describe the following distributed algorithm in Algorithm 2, where the load dispatcher and each server can solve their own problems with only local information.

At each iteration t, by solving the problem in (26), each server obtains the optimal service rate and dispatched load. Then, the server implements the optimal decision by changing into another service rate or maintaining its current rate. The load dispatcher also implements the new dispatch policy according to the solutions of each server. By updating the Lagrange duality variable with the gradient iterate, the load dispatcher broadcasts the new duality variable to each

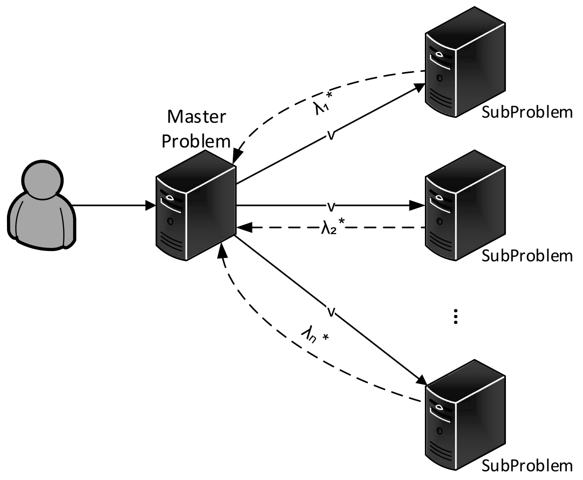

Fig. 2 Feedback information interaction

server for the next iteration. The servers and the load dispatcher make decision according to the optimization results in each iteration, even if the current iteration is not optimal, but it will converge to optimal via sufficient iterations.

In Algorithm 2, there are some feedback information interaction (see Fig. 2). To solve the problem in (26), each server needs to know the offered duality variable of the dispatcher that is using it. This can be obtained by the broadcast notification from the dispatcher. Hence, each server can adjust its service rate itself by the local information, instead of centralized control. So, at the beginning of each iteration, each server must wait to receive the notification and then start its own computation. To update (30) and solve the problem (26), the dispatcher needs to know the allocated service rates vector computed by each server and assign loads to each server.

After the above dual decomposition, the following proposition can be proved using standard techniques in distributed gradient algorithms convergence analysis.

Proposition 1 *The dual variable $\nu(t)$ will converge to the dual optimal ν^* as $t \to \infty$ and the primal variable $\boldsymbol{\lambda}^*(\nu(t))$ will also converge to the primal optimal variable $\boldsymbol{\lambda}^*$ and $\boldsymbol{\mu}^*$.*

Proof Since the primal problem (26) at high level is a strictly convex optimization problem, and the constraints are strictly feasible because μ_i^{MAX} is strictly positive, the Slater's condition for strong duality holds, and the corresponding primal variables λ_i, μ_i give the globally optimal solution of the primal problem (26) by the above distributed Algorithm 2. The speed of convergence is difficult to formulate and will depend on many factors such as step size. □

Consider Step 3 in Algorithm 2, for solving the discrete speed scaling problem (29) with discrete solutions in the range $\{\mu_i^1, \dots, \mu_i^{M_i}\}$, we will preliminary solve the continuous speed scaling problem which is the relaxed problem of (29). In continuous speed scaling, we solve the relaxed problem of (29) with continuous variables, that is μ_i is relaxed as a real value. The solution $\bar{\lambda}_i^*(\nu(t))$ and $\bar{\mu}_i^*(\nu(t))$ of the relaxed problem is the optimal solution in continuous speed scaling.

In discrete speed scaling, we first get several local optimal solutions. Consider each solution. The λ_i has no other constraint in original problem, and $\bar{\lambda}_i^*(\nu(t))$ is the local optimal value. If $\bar{\mu}_i^*(\nu(t))$ is just in the discrete values set, we get $\mu_i^*(\nu(t)) = \bar{\mu}_i^*(\nu(t))$. Otherwise, the $\bar{\mu}_i^*(\nu(t))$ will be in the range of $(\mu_i^{MIN}, \mu_i^{MAX})$. We can get two feasible values of μ_i in the discrete values set which are closest to $\bar{\mu}_i^*(\nu(t))$ from left and right sides. The values must follow the condition $\bar{\lambda}_i^*(\nu(t)) < \mu_i \le \mu_i^{MAX}$. If both values satisfy the condition, the value whose objective function $L_i(\lambda_i, \mu_i, \nu)$ is smaller than the other is the optimal μ_i. Thus, we get several local optimal solutions in discrete speed scaling, and we will choose the minimum as the global optimal solution.

5.3.1 *Analytical results for $\alpha = 2$ and $C_i = 1$*

In Algorithm 2, each server locally solves its problem (26), which is a mixed integer nonlinear programming problem with one real variable λ_i and one integer variable γ_i. First, we relax integer constraint and get the relaxed problem $\overline{Li}$ as follows, which is a subproblem of continuous speed scaling:

$$\begin{aligned} \overline{Li}:\quad \min\quad & \big(L_i(\lambda_i, \mu_i, \nu)\big), \\ \text{s.t.}\quad & \mu_i - \lambda_i > 0; \\ & \mu_i^{MIN} \le \mu_i \le \mu_i^{MAX}; \\ & \lambda_i \ge 0. \end{aligned} \tag{31}$$

As proved in the last section, the problem $\overline{Li}$ is a convex optimization problem. According to the KKT conditions, we get Lagrangian stationarity in (32):

$$\begin{aligned} &\nabla L_i(\lambda_i, \mu_i, \nu) - \omega_1 \nabla(\mu_i - \lambda_i) - \omega_2 \nabla\big(\mu_i^{MAX} - \mu_i\big) \\ &\quad - \omega_3 \nabla \lambda_i - \omega_4 \nabla\big(\mu_i - \mu_i^{MIN}\big) = 0. \end{aligned} \tag{32}$$

The KKT conditions are similar to (22). The difference is that the coupled constraint $\sum_{i=1}^{N} \lambda_i - \lambda = 0$ here is decoupled, and the Lagrangian multiplier ν is a known value by each iteration. We can solve the nonlinear equations but unlikely get a closed-form solution.

Now we consider a closed-form solution to (32) which can be obtained for a special case when $\alpha = 2$ and $C_i = 1$, e.g., an M/M/1 queueing system. The KKT conditions can be written as

$$\begin{cases} \frac{\mu_i}{\lambda(\mu_i-\lambda_i)^2} + \beta k \mu_i - \nu + \omega_1 - \omega_3 = 0; \\ \frac{-\lambda_i}{\lambda(\mu_i-\lambda_i)^2} + \beta k \lambda_i - \omega_1 + \omega_2 - \omega_4 = 0; \\ \omega_1(\mu_i - \lambda_i) = 0; \\ \omega_2(\mu_i^{MAX} - \mu_i) = 0; \\ \omega_3 \lambda_i = 0; \\ \omega_4(\mu_i - \mu_i^{MIN}) = 0; \\ \omega_1, \omega_2, \omega_3, \omega_4 \geq 0. \end{cases} \tag{33}$$

By solving the 5 equations, we assume the kth solution of (33) is $\lambda_i^{(k)}$ and $\bar{\mu}_i^{(k)}$. Consider the 5th equation. We will discuss the following possible cases.

1. For the case $\lambda_i = 0$ and $\omega_3 \geq 0$:
From the equations, we get $\omega_2 - \omega_4 = 0$. If $\omega_2 = \omega_4 > 0$, we get the contradiction equation $\mu_i^{MAX} = \mu_i^{MIN}$. So, $\omega_2 = \omega_4 = 0$. Then, we know that μ_i can be an arbitrary value in set F_i. For practical consideration, if the dispatcher routes no job to this server, the server is idle and should run at the lowest speed, So, we get the first solution $\lambda_i^{(1)} = 0$ and $\mu_i^{(1)} = \mu_i^{MIN}$.
2. For the case $\omega_3 = 0$, if $\omega_2 = 0$ and $\omega_4 = 0$:
From the equations, we get the second solution $\lambda_i^{(2)} = \frac{\nu}{2\beta k} - \sqrt{\frac{1}{\lambda \beta k}}$ and $\bar{\mu}_i^{(2)} = \frac{\nu}{2\beta k}$, which is the solution of continuous speed scaling. For discrete speed scaling, we assume $\lfloor \bar{\mu}_i^{(2)} \rfloor$ is the left side of $\bar{\mu}_i^{(2)}$ in discrete speed value set, and $\mu > \lceil \bar{\mu}_i^{(2)} \rceil$ is the right side of $\bar{\mu}_i^{(2)}$ in discrete speed value set. If both $\lfloor \bar{\mu}_i^{(2)} \rfloor$ and $\lceil \bar{\mu}_i^{(2)} \rceil$ exist, we can obtain $\mu_i^{(2)}$ as
$$\mu_i^{(2)} = \operatorname{argmin}\{L_i(\lfloor \bar{\mu}_i^{(2)} \rfloor), L_i(\lceil \bar{\mu}_i^{(2)} \rceil)\}. \tag{34}$$
3. For the case $\omega_3 = 0$, if $\omega_2 = 0$ and $\bar{\mu}_i = \mu_i^{MIN}$:
From the equations, we get
$$\lambda_i^{(3)} = \mu_i^{MIN} - \sqrt{\frac{\mu_i^{MIN}}{\lambda(\nu - \beta k \mu_i^{MIN})}}.$$
4. For the case $\bar{\mu}_i = \mu_i^{MAX}$:
From the equations, under the condition $\nu - \beta k \mu_i^{MAX} \geq 0$, we get
$$\lambda_i^{(3)} = \mu_i^{MAX} - \sqrt{\frac{\mu_i^{MAX}}{\lambda(\nu - \beta k \mu_i^{MAX})}},$$
$$\mu_i^{(3)} = M.$$

Thus, we get a closed-form solution for all the four local optimal λ_i and μ_i values as a function of ν, $(\lambda_i^{(1)}, \mu_i^{(1)})$, $(\lambda_i^{(2)}, \mu_i^{(2)})$, $(\lambda_i^{(3)}, \mu_i^{(3)})$, $(\lambda_i^{(4)}, \mu_i^{(4)})$. In each iteration t, different $\nu(t)$ will make some of the four solutions not feasible, so we must check the solutions under constraints of (26). If more than one solutions are satisfied, we will choose the unique minimum solution as the global optimal.

Table 1 Main parameters and their explanations

Parameters	Explanations
$N = 3$	Number of types of servers
M_i	Number of feasible service rates for each server: $M_1 = 2$, $M_2 = 7$, $M_3 = 10$
C_i	Coefficient of variation of service time: $C_1 = 1$, $C_2 = 0.01$, $C_3 = 0.1$
$\alpha = 2$	Exponent of power function
$\lambda \in [0, 30]$	Job arrival rate
μ_i	$\mu_1 = 3$, $\mu_2 = 5$, $\mu_3 = 8$, for servers without DVFS
μ_i^{MAX}	$\mu_1^{MAX} = 3$, $\mu_2^{MAX} = 5$, $\mu_3^{MAX} = 8$, for servers with DVFS
μ_i^{MIN}	$\mu_1^{MIN} = \mu_2^{MIN} = \mu_3^{MIN} = 0$, for servers with DVFS
k_i	Coefficient of power function: $k_1 = 0.1$, $k_2 = 0.2$, $k_3 = 0.5$
P_i^*	Static power consumption: $P_1^* = 1$, $P_2^* = 2$, $P_3^* = 5$

6 Numerical results

In this section, we present some numerical and simulation results for the proposed algorithms by considering servers without and with DVFS. We consider three types of heterogeneous servers, and each type owns 100 homogeneous servers. The main parameters are shown in Table 1.

These heterogeneous servers have different power consumption characteristics and processing capacities. Typically, the three types are lightweight servers (denoted by server 1) with low service rate and power consumption, middleweight servers (denoted by server 2) with medium service rate and power consumption, heavyweight servers (denoted by server 3) with high service rate and power consumption. We assume that the service rates are fixed for these three types without DVFS, i.e., $\mu_1 = 3$, $\mu_2 = 5$, $\mu_3 = 8$. For servers with DVFS, a high service rate means a high maximum service rate. Correspondingly, the maximum service rates with DVFS are $\mu_1^{MAX} = 3$, $\mu_2^{MAX} = 5$, $\mu_3^{MAX} = 8$. Besides the maximum rate, each server has different number of discrete service rate M_i. We assume that $M_1 = 2$, $M_2 = 7$, $M_3 = 10$. According to the simulation results, we can know that for servers of the same type, the speed scaling behaviors are consistent and the optimal load dispatch policy is load balanced. Thus, we present the simulation results at the level of server types, instead of specific servers. By server i we mean all servers in type i.

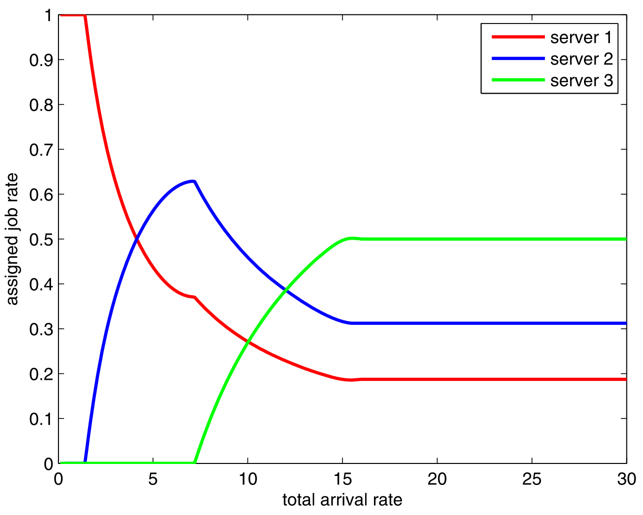

Fig. 3 Dispatched load proportion when $\beta = 1$

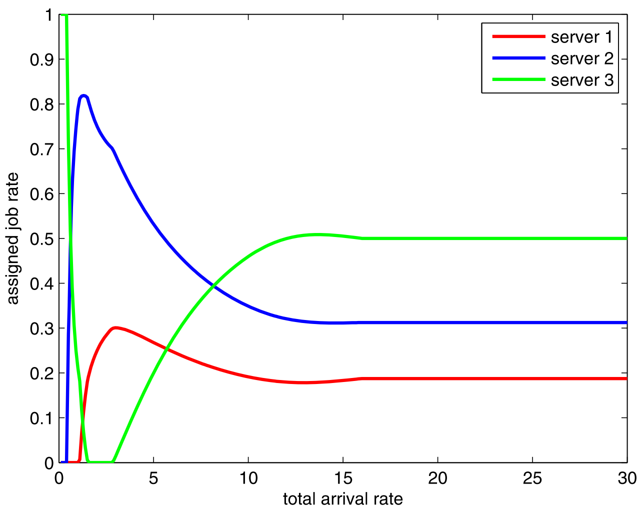

Fig. 4 Dispatched load proportion when $\beta = 0.05$

6.1 Servers without DVFS

The weighting factor β adjusts the tradeoff between performance and power consumption. A larger β means higher weight on power consumption and lower performance restriction, and vice versa.

Figures 3 and 4 illustrate the load dispatch results in terms of percentage in the cases of $\beta = 1$ and $\beta = 0.05$. It can be observed that in the case of $\beta = 1$, when the workload is light, the lightweight server 1 is assigned with most of the load. With the load increasing, the lightweight server 1 cannot satisfy the performance demand, which leads to assigning more proportions to server 2 and server 3. Thus, the proportion of server 1 decreases and those of server 2 and server 3 increase. In addition, the quantity of load assigned to all the servers increases due to the total heavier load. If the load continues to grow, all the servers are at full

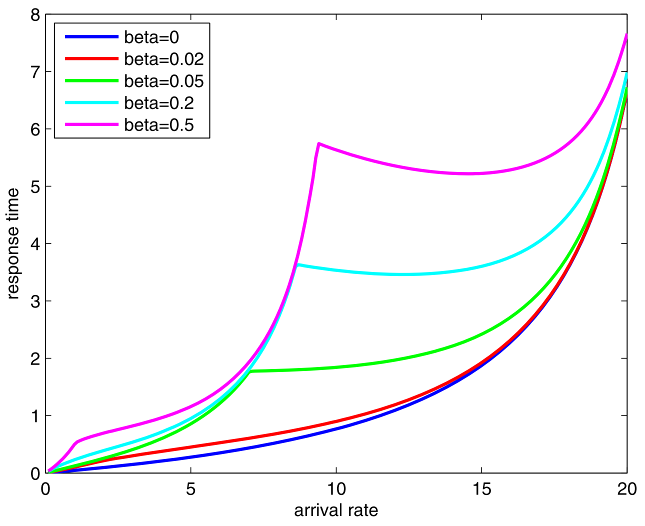

Fig. 5 The response time degradation for different beta values

load, the steady-state proportions are achieved. In the case of $\beta = 0.05$, the performance restriction is higher than the case of $\beta = 1$. So, even the load is light, the proportion of heavyweight server 3 is large so as to lower the response time. The same as $\beta = 1$, when all the servers are at full load, the steady-state proportions are achieved.

Different values of β lead to different levels of performance degradation. In particular, the response time will increase when β grows. We can get the degradation of performance with the growth of β. Figure 5 illustrates the response time degradation when $\beta \in \{0, 0.02, 0.05, 0.2, 0.5\}$. If $\beta = 0$, the portion of power consumption in the objective function vanishes, which means that there is no power consumption concern, and there is no performance degradation. With the growth of β, the response time gets larger. Therefore, we must cautiously choose the value of β according to the service level agreement (SLA) of users.

6.2 Servers with DVFS

A server can adjust its service rate with DVFS, instead of a fixed value. We assume that the servers in the last section are enabled with DVFS, with the maximum service rates the same as the fixed service rates without DVFS.

We simulate Algorithm 2 with step size $\delta = 0.01$. Figure 6 illustrates the convergence property of the proposed algorithm when $\beta = 1$ and $\lambda = 10$. It shows the dual function $L_i(\lambda_i, \mu_i, \nu_i)$ at each iteration. We can see that the dual function converges to the optimal quite fast. Roughly in the 50th iteration, the dual functions reach the steady state and achieve the optima.

Figures 7 and 8 illustrate the load dispatch results in terms of percentage in the cases of $\beta = 1$ and $\beta = 0.05$. It can be observed that proportion curves of assigned load almost conform to those without DVFS shown in Figs. 3

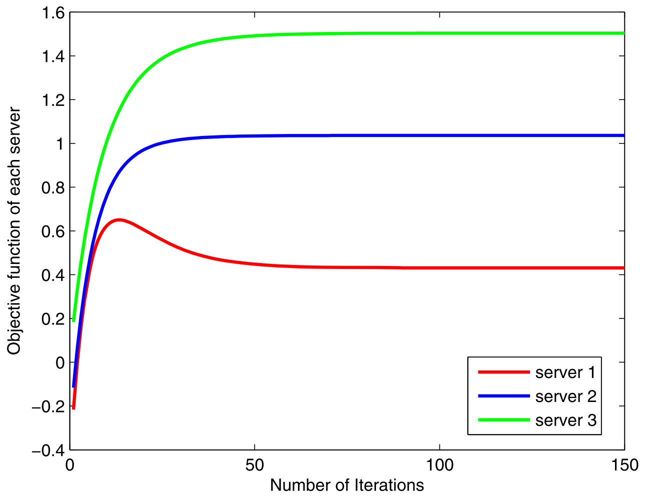

Fig. 6 Illustration of convergence property

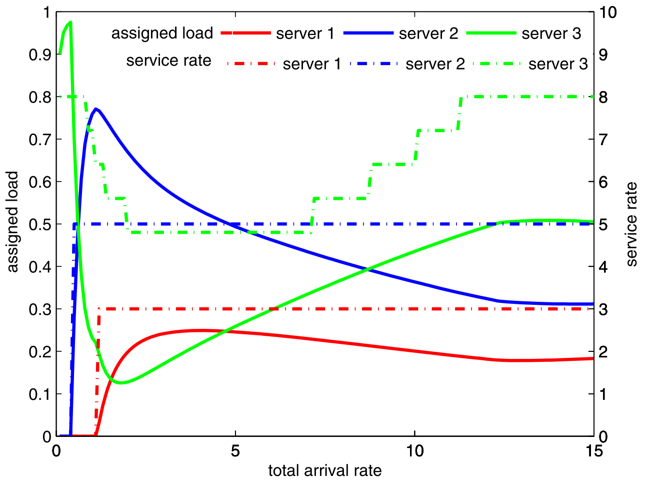

Fig. 8 Dispatched load proportion and speed scaling policy when $\beta = 0.05$

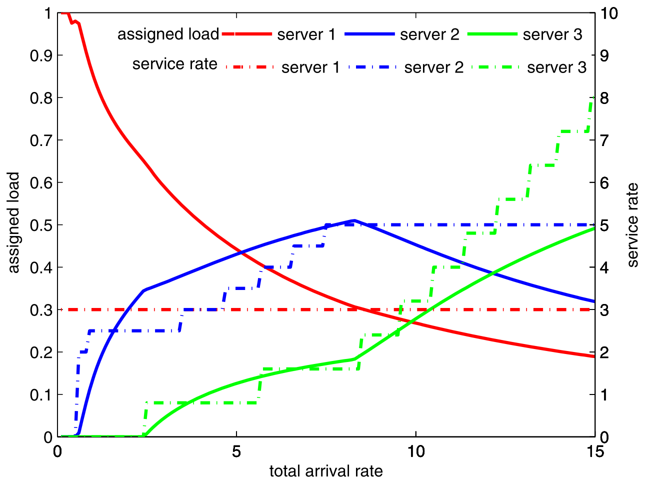

Fig. 7 Dispatched load proportion and speed scaling policy when $\beta = 1$

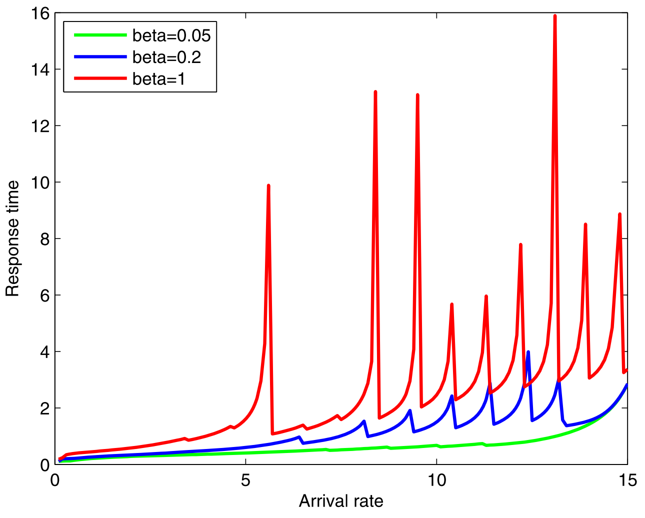

Fig. 9 The response time degradation for different beta values

and 4. Instead of constant service rates, the service rates vary in a way consistent with that proportion curves. In the case of $\beta = 1$, when the load is light, the load proportion of server 1 is large and the service rate of server 1 maintains high. When the load grows, the larger load proportion of servers 2 and 3 can achieve an optimal tradeoff. Thus, the service rate of server 1 decreases and that of servers 2 and 3 increase. If the load continues to full load, server 3 must increase its service rate to meet the performance demands. In the case of $\beta = 0.05$, the performance restriction is higher than the case of $\beta = 1$. So even the load is light, server 3 still maintain its maximum service rate. With the growth of load, server 1 and server 2 will speed up to its maximum capacity.

Figure 9 shows the response time degradation. We can observe that the curve will oscillate around the optimal. This oscillating behavior mathematically results from the integer constraint of the dual function, which is also the reason for the oscillation in Fig. 10.

6.3 Comparison between DVFS and None-DVFS

To show the objective function gain of our Algorithm 2, we provide comparison of the achieved objective function value and the corresponding power consumption.

As shown in Fig. 10, the DVFS can produce a smaller optimal objective function value for the same β value, and a larger β generates more objective function gain.

The corresponding power consumption gain is shown in Fig. 11. The power cost in servers without DVFS increases rapidly with the job arrival rate, especially when the arrival rate is high. However, the power cost in servers with DVFS almost increases linearly with lower gradient. We can con-

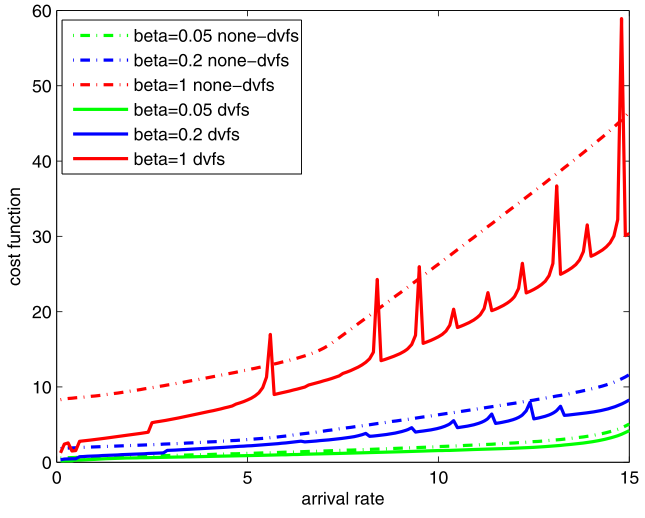

Fig. 10 Comparison of cost function between DVFS and None-DVFS

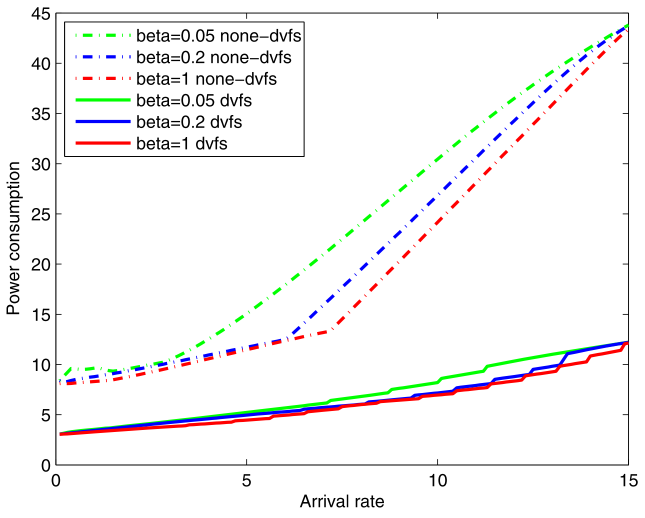

Fig. 11 Comparison of power consumption between DVFS and None-DVFS

clude that servers with DVFS can save at least 50 % power consumption compared with servers without DVFS.

7 Conclusion

In this paper, we have studied the problem of optimal load dispatching and speed scaling for heterogeneous servers in cloud computing. The propose is to provide an analytical way to study the various tradeoff between performance and power consumption by introducing a weighting factor. Our approach is to model a server as an M/G/1 queueing system and formulate the average response time as a function of the service rate. Without DVFS, the service rate is a constant. We prove the convexity of the problem and present the KKT conditions to provide the optimal load dispatching. With DVFS, the feasible service rates are discrete and bounded. We formulate the problem as an MINLP problem. We propose a distributed algorithm by online value iteration, which has lower complexity than a centralized algorithm. The simulation results show the convergence property of the proposed algorithm. Using our distributed algorithm, servers with DVFS can save at least 50 % power cost compared with servers without DVFS.

References

1. AMD Cool'n'Quiet technology. http://www.amd.com/us/products/technologies/cool-n-quiet/Pages/cool-n-quiet.aspx (2012)
2. Google unveils its container data center. http://www.datacenterknowledge.com/archives/2009/04/01/google-unveils-its-container-data-center/ (2012)
3. Andrew, L., Lin, M., Wierman, A.: Optimality, fairness, and robustness in speed scaling designs. ACM SIGMETRICS Perform. Eval. Rev. **38**(1), 37–48 (2010)
4. Bansal, N., Pruhs, K., Stein, C.: Speed scaling for weighted flow time. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 805–813. Society for Industrial and Applied Mathematics, Philadelphia (2007)
5. Barroso, L.: The price of performance. Queue **3**(7), 48–53 (2005)
6. Barroso, L., Holzle, U.: The case for energy-proportional computing. Computer **40**(12), 33–37 (2007)
7. Cao, J., Hwang, K., Li, K., Zomaya, A.: Optimal Multiserver Configuration for Profit Maximization in Cloud Computing (2012)
8. Chandrakasan, A., Sheng, S., Brodersen, R.: Low-power cmos digital design. IEICE Trans. Electron. **75**(4), 371–382 (1992)
9. Chen, G., He, W., Liu, J., Nath, S., Rigas, L., Xiao, L., Zhao, F.: Energy-aware server provisioning and load dispatching for connection-intensive Internet services. In: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, pp. 337–350 (2008). USENIX Association
10. Chen, L., Li, N., Low, S.: On the interaction between load balancing and speed scaling. In: ITA Workshop (2011)
11. Feng, W.: The importance of being low power in high performance computing. CTWatch Q. **1**(3), 11–20 (2005)
12. Floyd, M., Ghiasi, S., Keller, T., Rajamani, K., Rawson, F., Rubio, J., Ware, M.: System power management support in the IBM power6 microprocessor. IBM J. Res. Dev. **51**(6), 733–746 (2007)
13. Gandhi, A., Harchol-Balter, M., Das, R., Lefurgy, C.: Optimal power allocation in server farms. Perform. Eval. Rev. **37**(1), 157 (2009)
14. George, J., Harrison, J.: Dynamic control of a queue with adjustable service rate. Oper. Res. **49**(5), 720–731 (2001)
15. Graham, S., Snir, M., Patterson, C.: Getting up to Speed: The Future of Supercomputing. (2005), National Academy Press
16. Khan, S., Ahmad, I.: A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids. Parallel and Distributed Systems. IEEE Trans. Parallel Distrib. Syst. **20**(3), 346–360 (2009)
17. Krishna, C., Lee, Y.: Voltage-clock-scaling adaptive scheduling techniques for low power in hard real-time systems. In: Sixth IEEE Proceedings of Real-Time Technology and Applications Symposium, 2000. RTAS 2000, pp. 156–165. IEEE Press, New York (2000)
18. Lam, T., Lee, L., To, I., Wong, P.: Speed scaling functions for flow time scheduling based on active job count. In: Algorithms-ESA 2008, pp. 647–659 (2008)

19. Le Sueur, E., Heiser, G.: Dynamic voltage and frequency scaling: the laws of diminishing returns. In: Proceedings of the 2010 International Conference on Power Aware Computing and Systems, pp. 1–8. (2010). USENIX Association
20. Lee, Y., Krishna, C.: Voltage-clock scaling for low energy consumption in fixed-priority real-time systems. Real-Time Syst. **24**(3), 303–317 (2003)
21. Lee, Y., Zomaya, A.: Energy conscious scheduling for distributed computing systems under different operating conditions. IEEE Trans. Parallel Distrib. Syst. **22**(8), 1374–1381 (2011)
22. Leyffer, S.: Deterministic methods for mixed integer nonlinear programming. PhD University of Dundee (1993)
23. Li, K.: Optimal Power Allocation among Multiple Heterogeneous Servers in a Data Center. Sustainable Computing: Informatics and Systems (2011)
24. Li, K.: Optimal configuration of a multicore server processor for managing the power and performance tradeoff. J. Supercomput. **61**(1), 189–214 (2012)
25. Li, K.: Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers. IEEE Trans. Comput. **61**, 1668–1681 (2012). Special issue on energy efficient computing
26. Linden, G.: Marissa Mayer at Web 2.0 2006. http://glinden.blogspot.com/2006/11/marissa-mayer-at-web-20.html (2012)
27. Minas, L., Ellison, B.: Energy Efficiency for Information Technology: How to Reduce Power Consumption in Servers and Data Centers. (2009). Intel Press
28. Pallipadi, V.: Enhanced Intel Speedstep Technology and Demand-Based Switching on Linux. (2008). Intel Developer Service
29. Palomar, D., Chiang, M.: A tutorial on decomposition methods for network utility maximization. IEEE J. Sel. Areas Commun. **24**(8), 1439–1451 (2006)
30. Pinheiro, E., Bianchini, R., Carrera, E., Heath, T.: Load balancing and unbalancing for power and performance in cluster-based systems. In: Workshop on Compilers and Operating Systems for Low Power, vol. 180, pp. 182–195 (2001)
31. Tian, Y., Lin, C., Yao, M.: Modeling and analyzing power management policies in server farms using stochastic petri nets. In: Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy. Computing and Communication Meet, p. 26. ACM, New York (2012)
32. Wierman, A., Andrew, L., Tang, A.: Power-aware speed scaling in processor sharing systems. In: IEEE INFOCOM 2009, pp. 2007–2015. IEEE Press, New York (2009)
33. Yao, F., Demers, A., Shenker, S.: A scheduling model for reduced cpu energy. In: Proceedings of 36th Annual Symposium on Foundations of Computer Science, 1995, pp. 374–382. IEEE Press, New York (1995)
34. Zheng, X., Cai, Y.: Achieving energy proportionality in server clusters. Int. J. Comput. Netw. Commun. **1**(1), 21 (2009)
35. Zheng, X., Cai, Y.: Optimal server allocation and frequency modulation on multi-core based server clusters. Int. J. Green Comput. **1**(2), 18–30 (2010)
36. Zheng, X., Cai, Y.: Optimal server provisioning and frequency adjustment in server clusters. In: 39th International Conference on Parallel Processing Workshops 2010 (ICPPW), pp. 504–511. IEEE Press, New York (2010)
37. Zhong, X., Xu, C.: Energy-aware modeling and scheduling for dynamic voltage scaling with statistical real-time guarantee. IEEE Trans. Comput. **56**(3), 358–372 (2007)

Yuan Tian received the PhD degree in computer science and technology from Tsinahua University, China, in 2013. His current research interests include green network and performance evaluations.

Chuang Lin (SM'04) received the PhD degree in computer science from Tsinghua University in 1994. Currently, he is working as a professor in the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His current research interests include computer networks, performance evaluation, network security analysis, and Petri net theory and its applications. He has published more than 300 papers in research journals and IEEE conference proceedings in these areas and has published three books. He serves as the Technical Program vice chair, the 10th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS 2004); the General Chair, ACM SIGCOMM Asia workshop 2005; the associate editor, IEEE Transactions on Vehicular Technology; the area editor, Journal of Computer Networks; and the area editor, Journal of Parallel and Distributed Computing. He is a member of ACM Council, a senior member of the IEEE, and the Chinese Delegate in TC6 of IFIP.

Keqin Li is a SUNY Distinguished Professor of computer science, and an Intellectual Ventures endowed visiting chair professor at Tsinghua University, China. His research interests are mainly in design and analysis of algorithms, parallel and distributed computing, and computer networking. He has over 290 refereed research publications. He is currently or has served on the editorial board of *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *Journal of Parallel and Distributed Computing*, *International Journal of Parallel, Emergent and Distributed Systems*, *International Journal of High Performance Computing and Networking*, *International Journal of Big Data Intelligence*, and *Optimization Letters*.