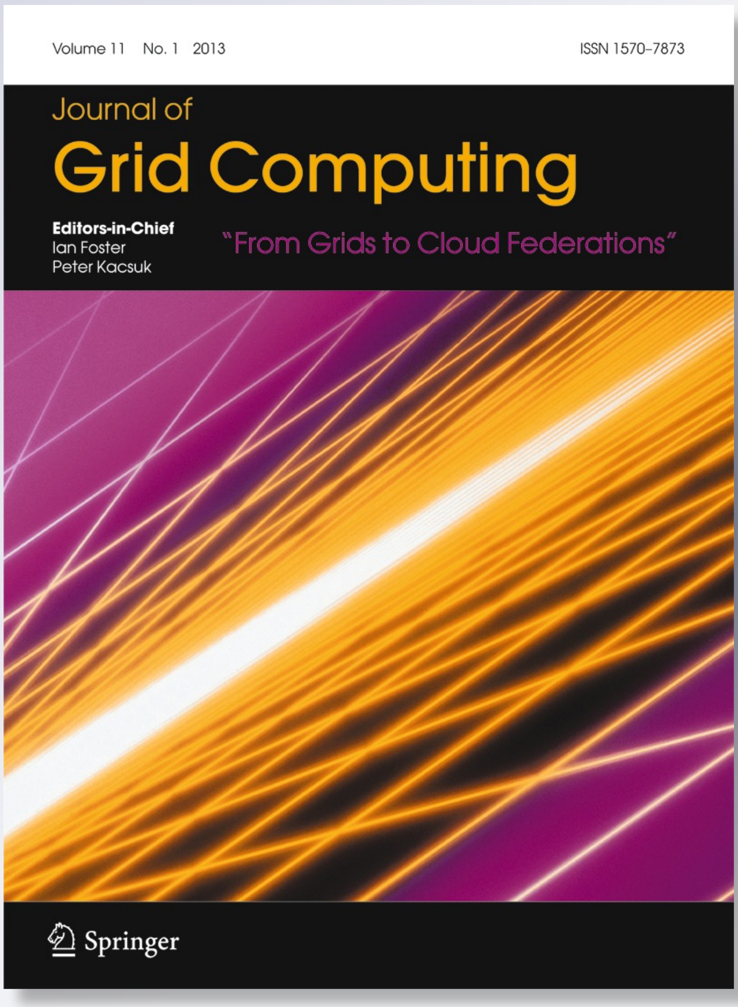
Optimal Load Distribution for Multiple Heterogeneous Blade Servers in a Cloud Computing Environment

Keqin Li

Journal of Grid Computing
From Grids to Cloud Federations

ISSN 1570-7873
Volume 11
Number 1

J Grid Computing (2013) 11:27-46
DOI 10.1007/s10723-012-9239-y



J Grid Computing (2013) 11:27–46
DOI 10.1007/s10723-012-9239-y

Optimal Load Distribution for Multiple Heterogeneous Blade Servers in a Cloud Computing Environment

Keqin Li

Received: 6 September 2011 / Accepted: 14 September 2012 / Published online: 16 October 2012

Abstract Given a group of heterogeneous blade servers in a cloud computing environment or a data center of a cloud computing provider, each having its own size and speed and its own amount of preloaded special tasks, we are facing the problem of optimal distribution of generic tasks over these blade servers, such that the average response time of generic tasks is minimized. Such performance optimization is important for a cloud computing provider to efficiently utilize all the available resources and to deliver the highest quality of service. We develop a queueing model for a group of heterogeneous blade servers, and formulate and solve the optimal load distribution problem of generic tasks for multiple heterogeneous blade servers in a cloud computing environment in two different situations, namely, special tasks with and without higher priority. Extensive numerical examples and data are demonstrated and some important observations are made. It is found that server sizes, server speeds, task execution requirement, and the arrival rates of special tasks all have significant impact on the average response time of generic tasks, especially when the total arrival rate of generic tasks is large. It is also found that the server size heterogeneity and the server speed heterogeneity do not have much impact on the average response time of generic tasks. Furthermore, larger (smaller, respectively) heterogeneity results in shorter (longer, respectively) average response time of generic tasks.

Keywords Blade server · Cloud computing · Optimal load distribution · Queueing model · Response time

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

1 Introduction

Cloud computing delivers hosted services over the Internet, such that access to shared hardware, software, databases, information, and all resources are provided to users on-demand, like electricity and other public utilities. A public or private or hybrid cloud provides easy and scalable access to computing resources and IT services [4]. Typical cloud computing providers deliver common applications online, which are accessed from a web browser, while the software and data are stored on servers. Users no longer need the knowledge of, expertise in, or control over the technology infrastructure in the cloud that supports their computing requirements. Cloud computing provides a new supplement, delivery, and consumption model for IT services based on the Internet. It typically involves the provision of dynamically scalable and often virtualized resources as a service over the Internet. It is a byproduct and consequence of the ease-of-access to remote computing sites provided by the Internet [3].

One form of cloud computing is *server consolidation*, which is an effective approach to the efficient usage of computer server resources in order to reduce the total number of servers or server locations that an organization requires. Currently, server sprawl is a common situation, in which multiple under-utilized servers take up more space and consume more energy than can be justified by their workload. According to a recent report, many companies typically run at 15–20 % of their capacity, which is not a sustainable ratio in the current economic environment [5]. Businesses are increasingly turning to server consolidation as one means of cutting unnecessary costs and maximizing return on investment in the data centers.

One approach to server consolidation is the use of *blade servers* or *blade centers* to maximize the efficient use of space and energy. A blade server is a server chassis housing multiple server blades. Blade servers allow more processing power in less rack space, simplifying cabling and reducing power consumption. Enterprises moving to blade servers can experience as much as 85 % reduction in cabling for blade installations over conventional rack or tower servers. The advantage of blade servers comes not only from the consolidation benefits of housing several servers in a single chassis, but also from the consolidation of associated resources (like storage and networking equipment) into a smaller architecture that can be managed through a single interface [6].

A *server blade* is a thin, modular electronic circuit board containing one, two, or more microprocessors and memory, that can be easily inserted into a blade server, which is a space-saving rack with many similar server blades. Each blade is a server in its own right, often dedicated to specific applications such as file sharing, Web page serving and caching, SSL encrypting of Web communication, transcoding of Web page content for smaller displays, and streaming audio and video content. It is conceivable that as server technologies further develop, server blades can also accommodate general purpose applications (i.e., nondedicated *generic tasks*) which can be executed on any server blades, in addition to special purpose applications (i.e., dedicated *special tasks*) which must be executed on designated server blades.

Like other server systems, a group of blade servers can also be managed by *load distribution and balancing*. Load balancing means distributing workload among two or more servers so that more work gets done in the same amount of time and all users get served faster. Load balancing has been the main reason for computer server *clustering*, i.e., the use of multiple servers to form what appears to users as a single high performance system. Cloud-based applications depend even more heavily on load balancing and optimization than traditional enterprise applications. For end users, load balancing capabilities will be seriously considered when they select a cloud computing provider. For cloud providers, load balancing capabilities will be a source of revenue, which is directly related to service quality (e.g., task response time). Hence, an efficient load balancing strategy is a key component to building out any cloud computing architecture.

For blade servers, the problem of load distribution and balancing has its specific formulation. Since each blade server has its dedicated special tasks, i.e., tasks that need to be performed by the blade server, load distribution and balancing can only be performed for nondedicated generic tasks. Therefore, given a group of heterogeneous blade servers in a cloud computing environment or a data center of a cloud computing provider, each having its own size and speed and its own amount of preloaded special tasks, we are facing the problem of optimal distribution of generic tasks over these blade servers, such that the average response time of generic tasks is minimized. Such performance optimization is important for a cloud computing provider to efficiently utilize all the available resources and to deliver the highest quality of service.

Optimal load distribution of generic tasks without special tasks for a group of heterogeneous multiserver queueing systems was studied in [2]. Optimal load distribution of generic tasks together with special tasks has been studied before for cluster and Grid computing environments, where each server is modeled as a queueing system with a single server [12]. In our investigation of this paper, each blade server is treated as a queueing system with multiple servers (i.e., multiple server blades). To the best of our knowledge, optimal load distribution of generic tasks together with special tasks for a group of heterogeneous multiserver queueing systems has not been considered before.

It should be noticed that our results in this paper not only provide new theoretical insights, but also are applicable to other server systems, such as

- a cluster of traditional heterogeneous clusters of PCs or workstations (i.e., multiple clusters, or a multicluster, or a cluster of clusters),
- or a cluster of heterogeneous multicore server processors (i.e., multiple multicore processors),

as long as the application environment is similar, i.e., there are generic tasks and the servers are preloaded with special tasks.

Load distribution and balancing in general parallel and distributed computing systems have been extensively studied and a huge body of literature exists (see the excellent collection [15]). In particular, the problems of optimal load distribution have been investigated by using queueing models [7, 17], with various performance metrics such as weighted mean response time [9], arithmetic average response time [10], probability of load imbalance [11], probability of load balancing success [13], mean response ratio [16], and mean miss rate [2]. Optimal load distribution in a heterogeneous distributed computer system with both generic and dedicated applications was studied in [1, 14].

The remainder of the paper is organized as follows. In Section 2, we describe a queueing model for a group of heterogeneous blade servers. In Sections 3 and 4 we formulate and solve our optimal load distribution problem of generic tasks for multiple heterogeneous blade servers in a cloud computing environment in two different situations, namely, special tasks with and without higher priority. In Section 5, we demonstrate extensive numerical data and point out several important observations. We conclude the paper in Section 6.

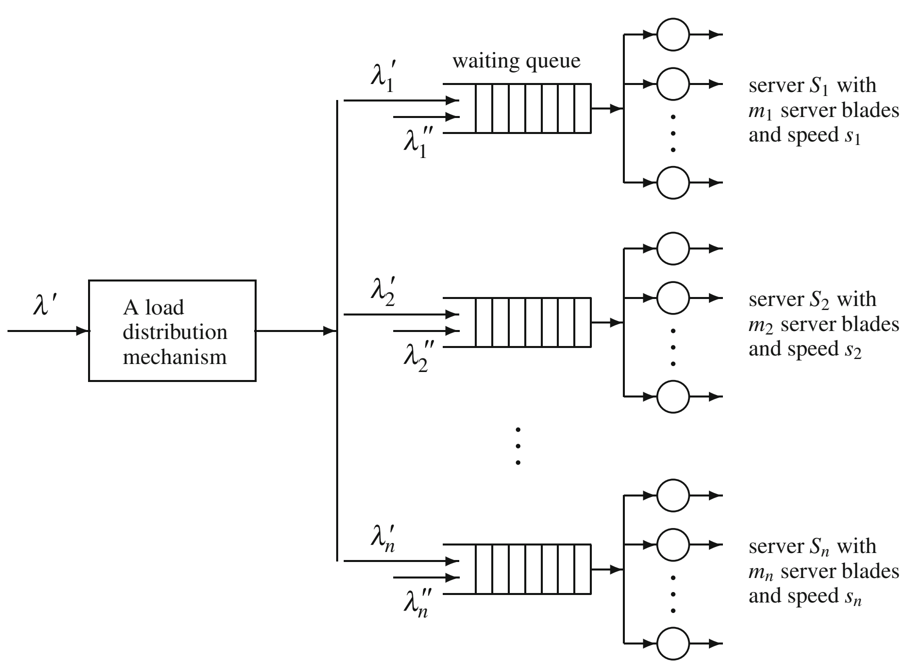
2 Modeling Blade Servers

To formulate and study the problem of optimal load distribution and balancing for multiple heterogeneous blade servers in a cloud computing environment, we need a model for a blade server and a group of blade servers. A queueing model for a group of n heterogeneous blade servers S_1, S_2, ..., S_n of sizes $m_1, m_2, ..., m_n$ is given in Fig. 1. Assume that a blade server S_i has m_i identical server blades (similarly, a cluster of m_i servers or a multicore server processor with m_i cores). In this paper, a blade server (in general, a cluster of traditional servers or a multicore server processor) is treated as an M/M/m queueing system which is elaborated as follows.

There is a Poisson stream of generic tasks with arrival rate λ', i.e., the inter-arrival times are independent and identically distributed (i.i.d.) exponential random variables with mean $1/\lambda'$. A load distribution and balancing algorithm will split the stream into n substreams, such that the ith substream with arrival rate λ'_i is sent to server S_i, where $1 \le i \le n$, and $\lambda' = \lambda'_1 + \lambda'_2 + \cdots + \lambda'_n$. There is an independent Poisson stream of special tasks with arrival rate λ''_i to server S_i. Hence, the actual task arrival stream to S_i is a merged stream of two separate streams, one for generic tasks with arrival rate λ'_i and one for special tasks with arrival rate λ''_i. It is clear that the actual task arrival rate to S_i is $\lambda_i = \lambda'_i + \lambda''_i$.

The task execution requirements (measured by the number of instructions to be executed) are i.i.d. exponential random variables r with mean $\bar{r}$. The m_i server blades of server S_i have identical execution speed s_i (measured by the number of instructions that can be executed in one unit of time). Hence, the task execution times on the server blades of S_i are i.i.d. exponential random variables $x_i = r/s_i$ with mean $\bar{x}_i = \bar{r}/s_i$.

A blade server S_i maintains a queue with infinite capacity for waiting tasks when all the m_i



λ'
A load distribution mechanism
λ_1'
λ_1''
waiting queue
server S_1 with m_1 server blades and speed s_1
λ_2'
λ_2''
server S_2 with m_2 server blades and speed s_2
λ_n'
λ_n''
server S_n with m_n server blades and speed s_n

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

server blades are busy. Two queueing disciplines are considered in this paper.

- *Special tasks without priority* – Both generic and special tasks are processed using the first-come-first-served (FCFS) queueing discipline. All generic and special tasks are mixed in the waiting queue and scheduled in the same way without any difference.
- *Special tasks with priority* – Special tasks have higher priority than generic tasks and are placed ahead of generic tasks in the waiting queue. Special tasks are always scheduled before generic tasks, and generic tasks are scheduled only when there is no special task in the waiting queue.

The processing of a task cannot be interrupted. Once a task starts execution, it is executed until it is completed.

3 Load Distribution with Special Tasks

Let $\mu_i = 1/\bar{x}_i = s_i/\bar{r}$ be the average service rate, i.e., the average number of tasks that can be finished by a server blade of S_i in one unit of time. The server utilization is

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

which is the average percentage of time that a server blade of S_i is busy. Since $\rho_i < 1$, it is required that $\lambda_i < m_i s_i/\bar{r}$.

Let $p_{i,k}$ denote the probability that there are k tasks (waiting or being processed) in the M/M/m system for S_i. Then, we have ([8], p. 102)

$$p_{i,k} = \begin{cases} p_{i,0}\dfrac{(m_i\rho_i)^k}{k!}, & k \le m_i; \\[2ex] p_{i,0}\dfrac{m_i^{m_i}\rho_i^k}{m_i!}, & k \ge m_i; \end{cases}$$

where

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

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

$$
\begin{aligned}
P_{q,i} &= \sum_{k=m_i}^{\infty} p_{i,k} \\
&= \sum_{k=m_i}^{\infty} p_{i,0} \frac{m_i^{m_i} \rho_i^k}{m_i!} \\
&= p_{i,0} \frac{m_i^{m_i}}{m_i!} \sum_{k=m_i}^{\infty} \rho_i^k \\
&= p_{i,0} \frac{m_i^{m_i}}{m_i!} \cdot \frac{\rho_i^{m_i}}{1-\rho_i} \\
&= p_{i,0} \frac{(m_i \rho_i)^{m_i}}{m_i!} \cdot \frac{1}{1-\rho_i} \\
&= \frac{p_{i,m_i}}{1-\rho_i}.
\end{aligned}
$$

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

$$
\begin{aligned}
\bar{N}_i &= \sum_{k=0}^{\infty} k p_{i,k} \\
&= \sum_{k=1}^{m_i} k p_{i,k} + \sum_{k=m_i+1}^{\infty} k p_{i,k} \\
&= \sum_{k=1}^{m_i} k p_{i,0} \frac{(m_i \rho_i)^k}{k!} + \sum_{k=m_i+1}^{\infty} k p_{i,0} \frac{m_i^{m_i} \rho_i^k}{m_i!} \\
&= m_i \rho_i \sum_{k=1}^{m_i} p_{i,0} \frac{(m_i \rho_i)^{k-1}}{(k-1)!} + p_{i,0} \frac{m_i^{m_i}}{m_i!} \sum_{k=m_i+1}^{\infty} k \rho_i^k \\
&= m_i \rho_i \sum_{k=0}^{m_i-1} p_{i,0} \frac{(m_i \rho_i)^k}{k!} + p_{i,0} \frac{m_i^{m_i}}{m_i!} \sum_{k=m_i+1}^{\infty} k \rho_i^k \\
&= m_i \rho_i \sum_{k=0}^{m_i-1} p_{i,k} + p_{i,0} \frac{m_i^{m_i}}{m_i!} \cdot \frac{\rho_i^{m_i+1}}{1-\rho_i} \left(m_i + \frac{1}{1-\rho_i} \right) \\
&= m_i \rho_i \sum_{k=0}^{m_i-1} p_{i,k} + p_{i,0} \frac{(m_i \rho_i)^{m_i}}{m_i!} \cdot \frac{\rho_i}{1-\rho_i} \\
&\quad \times \left(m_i + \frac{1}{1-\rho_i} \right) \\
&= m_i \rho_i \sum_{k=0}^{m_i-1} p_{i,k} + \frac{p_{i,m_i}}{1-\rho_i} \left(m_i \rho_i + \frac{\rho_i}{1-\rho_i} \right) \\
&= m_i \rho_i \sum_{k=0}^{m_i-1} p_{i,k} + P_{q,i} \left(m_i \rho_i + \frac{\rho_i}{1-\rho_i} \right) \\
&= m_i \rho_i \left(\sum_{k=0}^{m_i-1} p_{i,k} + P_{q,i} \right) + \frac{\rho_i}{1-\rho_i} P_{q,i} \\
&= m_i \rho_i \left(\sum_{k=0}^{m_i-1} p_{i,k} + \sum_{k=m_i}^{\infty} p_{i,k} \right) + \frac{\rho_i}{1-\rho_i} P_{q,i} \\
&= m_i \rho_i + \frac{\rho_i}{1-\rho_i} P_{q,i}.
\end{aligned}
$$

Applying Little's result ([8], p. 17), we get the average task response time as

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

Notice that the average length of the waiting queue in S_i is the average number of tasks in the system minus the average number of busy server blades, i.e.,

$$
\bar{N}_{q,i} = \bar{N}_i - m_i \rho_i = \frac{\rho_i}{1-\rho_i} P_{q,i}.
$$

By Little's result again, the average waiting time in the waiting queue is

$$
W_i = T_i - \bar{x}_i = \frac{\bar{N}_{q,i}}{\lambda_i} = \frac{P_{q,i}}{m_i(1-\rho_i)} \bar{x}_i = \frac{W_{0,i}}{1-\rho_i},
$$

where

$$
W_{0,i} = P_{q,i} \frac{\bar{x}_i}{m_i} = P_{q,i} W_i^*,
$$

and

$$
W_i^* = \frac{\bar{x}_i}{m_i}.
$$

Notice that $W_{0,i}$ is the expected time until a server blade is available, i.e., the product of the probability $P_{q,i}$ of queueing and W_i^*, and W_i^* is the expected time until the next completion of a task, i.e., the expectation of the minimum value of m_i i.i.d. exponential random variables with mean $\bar{x}_i$. (It should be mentioned that due to the memoryless property of an exponential distribution, the remaining execution time of a task is always the same random variable as before, no matter how long the task has been executed. Hence, the time until the next completion of a task is always the same random variable, i.e., the minimum value of m_i i.i.d. exponential random variables with mean $\bar{x}_i$.) Notice that $\rho_i = \lambda_i W_i^* = \lambda_i \bar{x}_i / m_i = (\lambda_i' + \lambda_i'') \bar{x}_i / m_i = \rho_i' + \rho_i''$, where $\rho_i' = \lambda_i' \bar{x}_i / m_i = \lambda_i' W_i^*$ and $\rho_i'' = \lambda_i'' \bar{x}_i / m_i = \lambda_i'' W_i^*$.

Notice that due to multiple server blades, a task t does not need to wait until all tasks in front of it are completed. Actually, W_i includes $W_{0,i}$ and $\bar{N}_{q,i} = \lambda_i W_i$ completions of task executions. The reason is that after $W_{0,i}$ amount of time, the first task in the waiting queue will be executed, and after $\bar{N}_{q,i}$ completions of task executions, task t is at the front of the waiting queue and will be scheduled to be executed. Therefore, we have

$$W_i = W_{0,i} + \bar{N}_{q,i} W_i^* = W_{0,i} + \lambda_i W_i W_i^*,$$

which yields

$$W_i = \frac{W_{0,i}}{1 - \lambda_i W_i^*} = \frac{W_{0,i}}{1 - \lambda_i \bar{x}_i / m_i} = \frac{W_{0,i}}{1 - \rho_i}.$$

The above argument is important since it can be adapted to find the average waiting time of generic tasks when special tasks have higher priority (see the next section).

For special tasks without priority, since all generic and special tasks are mixed in the waiting queue and scheduled using the FCFS queueing discipline in the same way with no difference, the average response time T_i' of generic tasks in server S_i is

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

where

$$P_{q,i} = \frac{p_{i,m_i}}{1 - \rho_i},$$

$$p_{i,m_i} = p_{i,0} \frac{(m_i \rho_i)^{m_i}}{m_i!},$$

and

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

Therefore, we get

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

The average response time T' of all generic tasks in the group of n blade servers is

$$T' = \frac{\lambda_1'}{\lambda'} T_1' + \frac{\lambda_2'}{\lambda'} T_2' + \cdots + \frac{\lambda_n'}{\lambda'} T_n',$$

which is the main performance measure that needs to be optimized.

Our *optimal load distribution problem* for multiple heterogeneous blade servers in a cloud computing environment can be specified as follows: given the number of blade servers n, the sizes of the servers $m_1, m_2, ..., m_n$, the execution speeds of the servers $s_1, s_2, ..., s_n$, the arrival rates of special tasks on the servers $\lambda_1'', \lambda_2'', ..., \lambda_n''$, the total arrival rate of generic tasks λ', and the average task execution requirement $\bar{r}$, find the arrival rates of generic tasks on the servers $\lambda_1', \lambda_2', ..., \lambda_n'$, such that the average response time of generic tasks T' is minimized, subject to the constraint

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

where

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

and $\rho_i < 1$, i.e., $\lambda_i' < m_i / \bar{x}_i - \lambda_i''$, for all $1 \le i \le n$.

We can minimize T' by using the method of Lagrange multiplier, namely,

$$\nabla T'(\lambda_1', \lambda_2', ..., \lambda_n') = \phi \nabla F(\lambda_1', \lambda_2', ..., \lambda_n'),$$

that is,

$$\frac{\partial T'}{\partial \lambda_i'} = \phi,$$

for all $1 \le i \le n$, where ϕ is a Lagrange multiplier. It is clear that

$$\frac{\partial T'}{\partial \lambda_i'} = \frac{1}{\lambda'} \left(T_i' + \lambda_i' \frac{\partial T_i'}{\partial \lambda_i'} \right),$$

where

$$\frac{\partial T_i'}{\partial \lambda_i'} = \frac{\partial T_i'}{\partial \rho_i} \cdot \frac{\partial \rho_i}{\partial \lambda_i'} = \frac{\bar{x}_i}{m_i} \cdot \frac{\partial T_i'}{\partial \rho_i},$$

that is,

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

where

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

and

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

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

Given n, $m_1, m_2, ..., m_n$, $\bar{x}_1, \bar{x}_2, ..., \bar{x}_n$, $\lambda_1'', \lambda_2'', ..., \lambda_n''$, and λ', our algorithms to find ϕ and $\lambda_1', \lambda_2', ..., \lambda_n'$ and T' are given in Figs. 2 and 3.

Our main observation in solving the equation $\partial T'/\partial\lambda_i' = \phi$ is that T' is a convex function of λ_i', i.e., $\partial T'/\partial\lambda_i'$ is an increasing function of λ_i'. In Fig. 2, we present an algorithm to find λ_i', when given m_i, $\bar{x}_i$, λ_i'', λ', and ϕ. Basically, the algorithm searches for λ_i' in an interval $[lb, ub]$ by using the bisection method (lines (9)–(18)), where $lb = 0$ (line (2)) such that $\partial T'/\partial\lambda_i'$ (with $\lambda_i' = 0$) $< \phi$, and ub is gradually increased (line (5)), starting from some small value (line (3)), such that $\partial T'/\partial\lambda_i'$ (with $\lambda_i' = ub$) $\ge \phi$. When ub is increased, it is also made sure that $ub < m_i/\bar{x}_i - \lambda_i''$ (lines (6)–(7)). In the algorithm, ϵ is a very small quantity.

In Fig. 3, we present an algorithm to find ϕ and $\lambda_1', \lambda_2', ..., \lambda_n'$ and T', when given n, $m_1, m_2, ..., m_n$, $\bar{x}_1, \bar{x}_2, ..., \bar{x}_n$, $\lambda_1'', \lambda_2'', ..., \lambda_n''$, and λ'. Our main observation is that F is an increasing function of $\lambda_1', \lambda_2', ..., \lambda_n'$, which, when obtained by algorithm Find_λ_i', are also increasing functions of ϕ. In lines (1)–(27), we determine ϕ. Again, ϕ is searched in an interval $[lb, ub]$ by using the bisection method (lines (14)–(26)), where $lb = 0$ (line (12)) such

```
Algorithm: Find_$\lambda_i'$ ($m_i$, $\bar{x}_i$, $\lambda_i''$, $\lambda'$, $\phi$)
Input: $m_i$, $\bar{x}_i$, $\lambda_i''$, $\lambda'$, and $\phi$.
Output: $\lambda_i'$.

//Initialize $[lb, ub]$ for $\lambda_i'$                          (1)
$lb \leftarrow 0$;                                                  (2)
$ub \leftarrow$ a small value;                                      (3)
while ($\partial T'/\partial\lambda_i'$ (with $\lambda_i' = ub$) $< \phi$)   (4)
    $ub \leftarrow 2ub$;                                            (5)
    if ($ub \ge m_i/\bar{x}_i - \lambda_i''$)                       (6)
        $ub \leftarrow (1-\varepsilon)(m_i/\bar{x}_i - \lambda_i'')$;   (7)
end while;                                                          (8)
//Search $\lambda_i'$ in $[lb, ub]$                                 (9)
while ($ub - lb > \varepsilon$)                                     (10)
    $middle \leftarrow (lb + ub)/2$;                                (11)
    if ($\partial T'/\partial\lambda_i'$ (with $\lambda_i' = middle$) $< \phi$)   (12)
        $lb \leftarrow middle$;                                     (13)
    else                                                            (14)
        $ub \leftarrow middle$;                                     (15)
    end if;                                                         (16)
end while;                                                          (17)
$\lambda_i' \leftarrow (lb + ub)/2$;                                (18)
return $\lambda_i'$.                                                (19)
```

Fig. 2 An algorithm to find λ_i'

```
*Algorithm*: Calculate $T'$.
*Input*: $n, m_1, m_2, \ldots, m_n, \bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n, \lambda''_1, \lambda''_2, \ldots, \lambda''_n$, and $\lambda'$.
*Output*: $\phi$ and $\lambda'_1, \lambda'_2, \ldots, \lambda'_n$ and $T'$.

//Initialize $[lb, ub]$ for $\phi$                                   (1)
$\phi \leftarrow$ a small value;                                        (2)
**do**                                                                   (3)
    $\phi \leftarrow 2\phi$;                                             (4)
    $F \leftarrow 0$;                                                    (5)
    **for** ($i \leftarrow 1$; $i \le n$; $i$++)                         (6)
        $\lambda'_i \leftarrow$ Find_$\lambda'_i(m_i, \bar{x}_i, \lambda''_i, \lambda', \phi)$;   (7)
        $F \leftarrow F + \lambda'_i$;                                   (8)
    **end for**;                                                         (9)
**while** ($F < \lambda'$);                                              (10)
//Search $\phi$ in $[lb, ub]$                                            (11)
$lb \leftarrow 0$;                                                       (12)
$ub \leftarrow \phi$;                                                    (13)
**while** ($ub - lb > \varepsilon$)                                      (14)
    $middle \leftarrow (lb + ub)/2$;                                     (15)
    $F \leftarrow 0$;                                                    (16)
    **for** ($i \leftarrow 1$; $i \le n$; $i$++)                         (17)
        $\lambda'_i \leftarrow$ Find_$\lambda'_i(m_i, \bar{x}_i, \lambda''_i, \lambda', middle)$;   (18)
        $F \leftarrow F + \lambda'_i$;                                   (19)
    **end for**;                                                         (20)
    **if** ($F < \lambda'$)                                              (21)
        $lb \leftarrow middle$;                                          (22)
    **else**                                                             (23)
        $ub \leftarrow middle$;                                          (24)
    **end if**;                                                          (25)
**end while**;                                                           (26)
$\phi \leftarrow (lb + ub)/2$;                                           (27)
//Calculate $\lambda'_1, \lambda'_2, \ldots, \lambda'_n$                 (28)
**for** ($i \leftarrow 1$; $i \le n$; $i$++)                             (29)
    $\lambda'_i \leftarrow$ Find_$\lambda'_i(m_i, \bar{x}_i, \lambda''_i, \lambda', \phi)$;   (30)
**end for**;                                                             (31)
//Calculate and return $T'$                                              (32)
$T' \leftarrow 0$;                                                       (33)
**for** ($i \leftarrow 1$; $i \le n$; $i$++)                             (34)
    $T' \leftarrow T' + (\lambda'_i/\lambda')T'_i$;                      (35)
**end for**;                                                             (36)
**return** $T'$.                                                         (37)
```

Fig. 3 An algorithm to calculate the minimized T'

that $F < \lambda'$ (with $\phi = 0$), and ub is gradually increased (line (5)) such that $F \geq \lambda'$ (with $\phi = ub$). Once ϕ is available, we can calculate $\lambda'_1, \lambda'_2, ..., \lambda'_n$ (lines (28)–(31)) and T' (lines (32)–(36)).

Example 1 Let us consider a group of $n = 7$ heterogeneous blade servers. The sizes of the servers are $m_i = 2i$, and the speeds of the servers are $s_i = 1.7 - 0.1i$ (giga instructions per second), for all $1 \leq i \leq n$, i.e., we have a mixture of smaller servers with faster speeds and larger servers with slower speeds. The task execution requirement is $\bar{r} = 1$ (giga instructions). The arrival rates of special tasks are $\lambda''_i = 0.3 m_i / \bar{x}_i$, for all $1 \leq i \leq n$, i.e., each blade server is preloaded by special tasks that contribute 30 % to server utilization. It is clear that the maximum arrival rate of generic tasks is

$$\lambda'_{\max} = \sum_{i=1}^{n} \left(\frac{m_i}{\bar{x}_i} - \lambda''_i \right).$$

In this example, we assume that $\lambda' = 0.5\lambda'_{\max} = 23.52$, which needs to be distributed on the n servers. The value of λ' implies that generic tasks contribute 0.5(100 %–30 %) = 35 % to server utilization, and the overall server utilization is roughly 65 %. The optimal load distribution $\lambda'_1, \lambda'_2, ..., \lambda'_n$ of generic tasks found by our algorithms are given in Table 1. The minimized average response time of generic tasks is $T' = 0.8964703$ s. In Table 1, we also show server utilizations $\rho_1, \rho_2, ..., \rho_n$. It is observed that for the optimal load distribution of generic tasks, the n servers have different utilizations.

Closed-form solutions of $\lambda'_1, \lambda'_2, ..., \lambda'_n$ and T' can be obtained for some very special cases. For instance, let us consider the case where $m_1 = m_2 = \cdots = m_n = 1$, i.e., each server has only one blade.

Theorem 1 *If $m_1 = m_2 = \cdots = m_n = 1$, the average response time T' of all generic tasks in the group of n blade servers is minimized when*

$$\lambda'_i = \frac{1}{\bar{x}_i} \left(1 - \rho''_i - \sqrt{\frac{\bar{x}_i (1 - \rho''_i)}{\lambda' \phi}} \right),$$

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

$$\phi = \left(\left(\frac{1}{\sqrt{\lambda'}} \sum_{i=1}^{n} \sqrt{\frac{1 - \rho''_i}{\bar{x}_i}} \right) \Big/ \left(\sum_{i=1}^{n} \frac{1 - \rho''_i}{\bar{x}_i} - \lambda' \right) \right)^2.$$

Proof When $m_i = 1$, we have

$$p_{i,0} = 1 - \rho_i,$$

$$T'_i = \frac{\bar{x}_i}{1 - \rho_i},$$

$$\frac{\partial p_{i,0}}{\partial \rho_i} = -1,$$

and

$$\frac{\partial T'_i}{\partial \rho_i} = \frac{\bar{x}_i}{(1 - \rho_i)^2},$$

for all $1 \leq i \leq n$. Based on (1), i.e.,

$$\frac{1}{\lambda'} \left(T'_i + \rho'_i \frac{\partial T'_i}{\partial \rho_i} \right) = \phi,$$

we get

$$\frac{1}{\lambda'} \left(\frac{\bar{x}_i}{1 - \rho_i} + \rho'_i \frac{\bar{x}_i}{(1 - \rho_i)^2} \right) = \phi,$$

Table 1 Numerical data in Example 1

i	m_i	s_i	x_i	λ'_i	λ''_i	ρ_i
1	2	1.6	0.6250000	0.6652046	0.9600000	0.5078764
2	4	1.5	0.6666667	1.8802882	1.8000000	0.6133814
3	6	1.4	0.7142857	2.9973639	2.5200000	0.6568290
4	8	1.3	0.7692308	3.9121948	3.1200000	0.6761726
5	10	1.2	0.8333333	4.5646028	3.6000000	0.6803836
6	12	1.1	0.9090909	4.8769307	3.9600000	0.6694644
7	14	1.0	1.0000000	4.6234149	4.2000000	0.6302439

that is,

$$\frac{1}{\lambda'}\cdot\frac{\bar{x}_i}{1-\rho_i}\left(1+\frac{\rho'_i}{1-\rho_i}\right)=\phi.$$

Consequently, we have

$$\frac{1}{\lambda'}\cdot\frac{\bar{x}_i}{1-\rho_i}\cdot\frac{1-\rho''_i}{1-\rho_i}=\phi,$$

that is,

$$(1-\rho_i)^2=\frac{\bar{x}_i(1-\rho''_i)}{\lambda'\phi}.$$

Since $\rho_i<1$, this implies that

$$\rho_i=1-\sqrt{\frac{\bar{x}_i(1-\rho''_i)}{\lambda'\phi}},$$

and

$$\rho'_i=1-\rho''_i-\sqrt{\frac{\bar{x}_i(1-\rho''_i)}{\lambda'\phi}},$$

for all $1\le i\le n$. Since $\rho'_i=\lambda'_i\bar{x}_i/m_i=\lambda'_i\bar{x}_i$, we obtain

$$\lambda'_i=\frac{1}{\bar{x}_i}\left(1-\rho''_i-\sqrt{\frac{\bar{x}_i(1-\rho''_i)}{\lambda'\phi}}\right),$$

for all $1\le i\le n$. By the fact that $\lambda'_1+\lambda'_2+\cdots+\lambda'_n=\lambda'$, we get

$$\lambda'=\sum_{i=1}^{n}\frac{1-\rho''_i}{\bar{x}_i}-\left(\frac{1}{\sqrt{\lambda'}}\sum_{i=1}^{n}\sqrt{\frac{1-\rho''_i}{\bar{x}_i}}\right)\frac{1}{\sqrt{\phi}},$$

which gives rise to

$$\sum_{i=1}^{n}\frac{1-\rho''_i}{\bar{x}_i}-\lambda'=\left(\frac{1}{\sqrt{\lambda'}}\sum_{i=1}^{n}\sqrt{\frac{1-\rho''_i}{\bar{x}_i}}\right)\frac{1}{\sqrt{\phi}},$$

and

$$\sqrt{\phi}=\left(\frac{1}{\sqrt{\lambda'}}\sum_{i=1}^{n}\sqrt{\frac{1-\rho''_i}{\bar{x}_i}}\right)\Bigg/\left(\sum_{i=1}^{n}\frac{1-\rho''_i}{\bar{x}_i}-\lambda'\right),$$

and

$$\phi=\left(\left(\frac{1}{\sqrt{\lambda'}}\sum_{i=1}^{n}\sqrt{\frac{1-\rho''_i}{\bar{x}_i}}\right)\Bigg/\left(\sum_{i=1}^{n}\frac{1-\rho''_i}{\bar{x}_i}-\lambda'\right)\right)^2.$$

The theorem is proved. □

4 Special Tasks of Higher Priority

In this section, we consider special tasks with higher priority than generic tasks, which are placed ahead of generic tasks in a waiting queue and are always scheduled before generic tasks.

First of all, we establish the following theorem, which gives the average response time of generic tasks when they are scheduled with special tasks of higher priority.

Theorem 2 *The average response time of generic tasks in server S_i is*

$$T'_i=\bar{x}_i\left(1+p_{i,0}\frac{m_i^{m_i-1}}{m_i!}\cdot\frac{1}{1-\rho''_i}\cdot\frac{\rho_i^{m_i}}{(1-\rho_i)^2}\right). \quad (2)$$

Remark Compared to T'_i where special tasks do not have priority, the T'_i in this theorem has an extra factor of $1/(1-\rho''_i)$, due to special tasks of higher priority.

Proof To find the average response time of generic tasks when they are scheduled with special tasks of higher priority, we need to figure out the average waiting time of generic tasks. To this end, we also need to find the average waiting time of special tasks.

Let $N_{q,i}=N'_{q,i}+N''_{q,i}$, where $N'_{q,i}$ and $N''_{q,i}$ are the average numbers of generic and special tasks in the waiting queue of server S_i respectively. Let W'_i and W''_i denote the average waiting time of generic and special tasks respectively. Then, by Little's result, we have $N'_{q,i}=\lambda'_iW'_i$ and $N''_{q,i}=\lambda''_iW''_i$. Since all special tasks are in front of generic tasks, we get

$$W''_i=W_{0,i}+N''_{q,i}W^*_i=W_{0,i}+\lambda''_iW''_iW^*_i,$$

which gives

$$W_i'' = \frac{W_{0,i}}{1 - \lambda_i'' W_i^*} = \frac{W_{0,i}}{1 - \lambda_i''(\bar{x}_i/m_i)} = \frac{W_{0,i}}{1 - \rho_i''} = \frac{P_{q,i} W_i^*}{1 - \rho_i''} = \frac{P_{q,i}}{m_i(1 - \rho_i'')}\bar{x}_i.$$

Let $M_i = \lambda_i'' W_i'$ be the average number of special tasks that arrive to server S_i while a generic task is in the waiting queue. Notice that W_i' includes $W_{0,i}$ and $N_{q,i}$ completions of task executions and M_i additional completions of task executions. Therefore, we have

$$\begin{aligned} W_i' &= W_{0,i} + (N_{q,i} + M_i) W_i^* \\ &= W_{0,i} + (N_{q,i}' + N_{q,i}'' + M_i) W_i^* \\ &= W_{0,i} + (\lambda_i' W_i' + \lambda_i'' W_i'' + \lambda_i'' W_i') W_i^* \\ &= W_{0,i} + (\lambda_i W_i' + \lambda_i'' W_i'') W_i^*, \end{aligned}$$

which yields

$$\begin{aligned} W_i' &= \frac{W_{0,i} + \lambda_i'' W_i'' W_i^*}{1 - \lambda_i W_i^*} \\ &= \frac{W_{0,i} + \rho_i'' W_i''}{1 - \rho_i} \\ &= \frac{W_{0,i} + W_{0,i}\rho_i''/(1 - \rho_i'')}{1 - \rho_i} \\ &= \frac{W_{0,i}}{(1 - \rho_i'')(1 - \rho_i)} \\ &= \frac{P_{q,i}}{m_i(1 - \rho_i'')(1 - \rho_i)}\bar{x}_i. \end{aligned}$$

The average response time of generic tasks in server S_i is

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

which is actually

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

This proves the theorem. □

Our optimal load distribution problem for multiple heterogeneous blade servers in a cloud computing environment with prioritized special tasks can be formulated and solved in a similar way to that in the last section. The only difference is that the average response time of generic tasks in server S_i is modified as (2), and the partial derivative $\partial T_i'/\partial \rho_i$ is modified as follows:

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

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

Example 2 Let us consider the same system in Example 1. All parameters remain the same, except that special tasks have higher priority than generic tasks. The optimal load distribution $\lambda_1', \lambda_2', ..., \lambda_n'$ of generic tasks found by our algorithms are given in Table 2. The minimized average response time of generic tasks is $T' = 0.9209392$ s, which is greater than that in Example 1, where special tasks have no priority.

Similar to the last section, we can also consider the case where $m_1 = m_2 = \cdots = m_n = 1$, i.e., each server has only one blade.

Theorem 3 *If $m_1 = m_2 = \cdots = m_n = 1$, the average response time T' of all generic tasks in the group of n blade servers with prioritized special tasks is minimized when*

$$\lambda_i' = \frac{1}{\bar{x}_i}\left(1 - \rho_i'' - \sqrt{\left(\frac{\lambda'\phi}{\bar{x}_i} + \frac{\rho_i''}{1 - \rho_i''}\right)^{-1}}\right),$$

for all $1 \le i \le n$, where ϕ is the root of the following equation:

$$\lambda' = \sum_{i=1}^{n} \frac{1}{\bar{x}_i}\left(1 - \rho_i'' - \sqrt{\left(\frac{\lambda'\phi}{\bar{x}_i} + \frac{\rho_i''}{1 - \rho_i''}\right)^{-1}}\right).$$

Table 2 Numerical data in Example 2

i	m_i	s_i	x_i	λ_i'	λ_i''	ρ_i
1	2	1.6	0.6250000	0.5908113	0.9600000	0.4846285
2	4	1.5	0.6666667	1.7714948	1.8000000	0.5952491
3	6	1.4	0.7142857	2.8813939	2.5200000	0.6430231
4	8	1.3	0.7692308	3.8136848	3.1200000	0.6667005
5	10	1.2	0.8333333	4.5164617	3.6000000	0.6763718
6	12	1.1	0.9090909	4.9419622	3.9600000	0.6743911
7	14	1.0	1.0000000	5.0041912	4.2000000	0.6574422

Proof When $m_i = 1$, we have

$$T_i' = \bar{x}_i \left(1 + \frac{1}{1-\rho_i''} \cdot \frac{\rho_i}{1-\rho_i} \right),$$

and

$$\frac{\partial T_i'}{\partial \rho_i} = \frac{\bar{x}_i}{(1-\rho_i'')(1-\rho_i)^2},$$

for all $1 \le i \le n$. Based on (1), i.e.,

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

we get

$$\frac{1}{\lambda'} \left(\bar{x}_i \left(1 + \frac{1}{1-\rho_i''} \cdot \frac{\rho_i}{1-\rho_i} \right) + \rho_i' \frac{\bar{x}_i}{(1-\rho_i'')(1-\rho_i)^2} \right) = \phi,$$

that is,

$$\frac{\bar{x}_i}{\lambda'} \left(1 + \frac{\rho_i}{(1-\rho_i'')(1-\rho_i)} + \frac{\rho_i'}{(1-\rho_i'')(1-\rho_i)^2} \right) = \phi.$$

Notice that

$$\begin{aligned}
&1 + \frac{\rho_i}{(1-\rho_i'')(1-\rho_i)} + \frac{\rho_i'}{(1-\rho_i'')(1-\rho_i)^2} \\
&= 1 + \frac{1}{1-\rho_i''} \left(\frac{1}{1-\rho_i} - 1 \right) + \frac{\rho_i'}{(1-\rho_i'')(1-\rho_i)^2} \\
&= 1 - \frac{1}{1-\rho_i''} + \frac{1}{(1-\rho_i'')(1-\rho_i)} + \frac{\rho_i'}{(1-\rho_i'')(1-\rho_i)^2} \\
&= -\frac{\rho_i''}{1-\rho_i''} + \frac{1}{(1-\rho_i'')(1-\rho_i)} \left(1 + \frac{\rho_i'}{1-\rho_i} \right) \\
&= \frac{1}{(1-\rho_i)^2} - \frac{\rho_i''}{1-\rho_i''}.
\end{aligned}$$

Therefore, we have

$$\frac{\bar{x}_i}{\lambda'} \left(\frac{1}{(1-\rho_i)^2} - \frac{\rho_i''}{1-\rho_i''} \right) = \phi,$$

that is,

$$\frac{1}{(1-\rho_i)^2} = \frac{\lambda'\phi}{\bar{x}_i} + \frac{\rho_i''}{1-\rho_i''},$$

which implies that

$$\rho_i = 1 - \sqrt{\left(\frac{\lambda'\phi}{\bar{x}_i} + \frac{\rho_i''}{1-\rho_i''} \right)^{-1}},$$

and

$$\rho_i' = 1 - \rho_i'' - \sqrt{\left(\frac{\lambda'\phi}{\bar{x}_i} + \frac{\rho_i''}{1-\rho_i''} \right)^{-1}},$$

and

$$\lambda_i' = \frac{1}{\bar{x}_i} \left(1 - \rho_i'' - \sqrt{\left(\frac{\lambda'\phi}{\bar{x}_i} + \frac{\rho_i''}{1-\rho_i''} \right)^{-1}} \right),$$

for all $1 \le i \le n$. Again, to find ϕ, we use the fact that $\lambda_1' + \lambda_2' + \cdots + \lambda_n' = \lambda'$, namely,

$$\lambda' = \sum_{i=1}^{n} \frac{1}{\bar{x}_i} \left(1 - \rho_i'' - \sqrt{\left(\frac{\lambda'\phi}{\bar{x}_i} + \frac{\rho_i''}{1-\rho_i''} \right)^{-1}} \right).$$

Hence, ϕ is the root of the above equation. □

5 Numerical Data

In this section, we demonstrate some numerical data and provide some important observations. Notice that all parameters are chosen for the purpose of illustration.

Recall that given n heterogeneous blade servers, our parameters for optimal load distribution of generic tasks are

- $m_1, m_2, ..., m_n$: the sizes of the servers;
- $s_1, s_2, ..., s_n$: the execution speeds of the server blades;
- $\bar{r}$: the average task execution requirement;
- $\lambda''_1, \lambda''_2, ..., \lambda''_n$: the arrival rates of special tasks.

Given the above parameters and the total arrival rate of generic tasks λ', the optimal arrival rates $\lambda'_1, \lambda'_2, ..., \lambda'_n$ of generic tasks on the servers and the minimized average response time T' of generic tasks can be found by using our algorithms.

In Figs. 4 and 5, we show the impact of server sizes on the average response time T' of generic tasks. We consider five groups of $n = 7$ heterogeneous blade servers:

- Group 1: $(m_1, m_2, m_3, m_4, m_5, m_6, m_7) = (1, 3, 5, 7, 9, 11, 13)$;
- Group 2: $(m_1, m_2, m_3, m_4, m_5, m_6, m_7) = (1, 3, 5, 8, 10, 12, 14)$;
- Group 3: $(m_1, m_2, m_3, m_4, m_5, m_6, m_7) = (2, 4, 6, 8, 10, 12, 14)$;
- Group 4: $(m_1, m_2, m_3, m_4, m_5, m_6, m_7) = (3, 5, 7, 8, 10, 12, 14)$;
- Group 5: $(m_1, m_2, m_3, m_4, m_5, m_6, m_7) = (3, 5, 7, 9, 11, 13, 15)$.

The total size (i.e., the total number of server blades) is $m = m_1 + m_2 + \cdots + m_n = 49, 53, 56, 59, 63$ in the five groups respectively. The server speeds are $s_i = 1.7 - 0.1i$, for all $1 \le i \le n$. The task execution requirement is $\bar{r} = 1$. The arrival rates of special tasks are $\lambda''_i = 0.3m_i/\bar{x}_i$, for all $1 \le i \le n$. For each group, the minimized average response time T' of generic tasks is plotted as a function of the total arrival rate of generic tasks λ' in two different cases, namely, special tasks without priority in Fig. 4 and special tasks with priority in Fig. 5. It is obvious that the average response time T' of generic tasks with prioritized special tasks is greater than that with non-prioritized special tasks. It is observed that slight increment of m noticeably reduces the average response time T' of generic tasks, i.e., server sizes have significant impact on the average response time T' of generic tasks, especially when λ' is large.

In Figs. 6 and 7, we show the impact of server speeds on the average response time T' of generic tasks. We consider a group of $n = 7$ heterogeneous blade servers of sizes $m_i = 2i$, for all $1 \le i \le n$. The server speeds are $s_i = s - 0.1i$, for all $1 \le i \le n$, with $s = 1.5, 1.6, 1.7, 1.8, 1.9$. The task execution requirement is $\bar{r} = 1$. The arrival rates of special tasks are $\lambda''_i = 0.3m_i/\bar{x}_i$, for all $1 \le i \le n$. For each s, the minimized average response time T' of generic tasks is plotted as a function of the total arrival rate of generic tasks λ' in two different cases, namely, special tasks without priority in Fig. 6 and special tasks with priority in Fig. 7. It is observed that slight increment of s noticeably

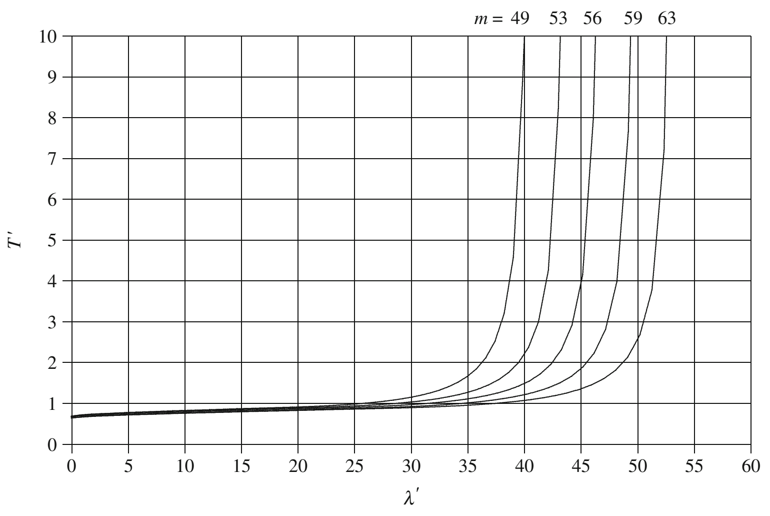
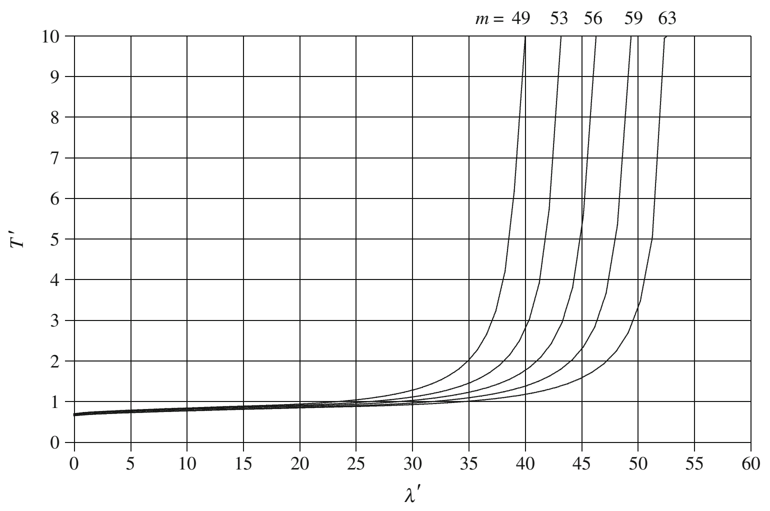


Fig. 4 T' vs. λ' and m (special tasks without priority)

Fig. 5 T' vs. λ' and m (special tasks with priority)

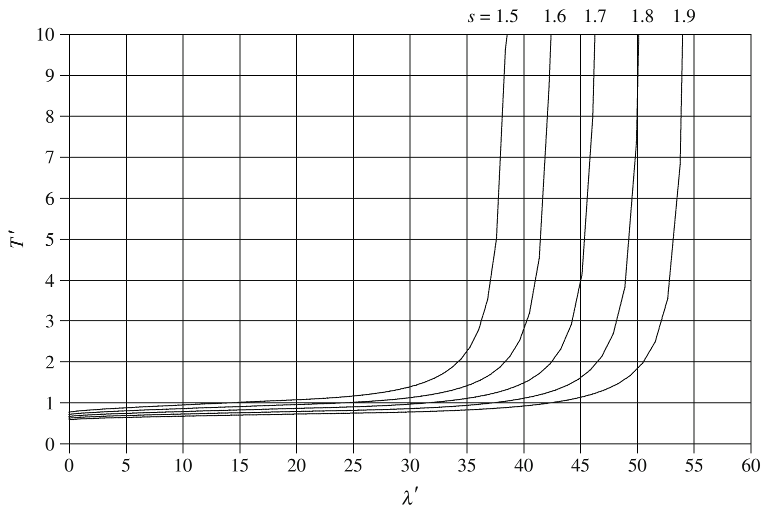


reduces the average response time T' of generic tasks, i.e., server speeds have significant impact on the average response time T' of generic tasks, especially when λ' is large.

In Figs. 8 and 9, we show the impact of the task execution requirement on the average response time T' of generic tasks. We consider a group of $n = 7$ heterogeneous blade servers of sizes $m_i = 2i$, for all $1 \le i \le n$. The server speeds are $s_i = 1.7 - 0.1i$, for all $1 \le i \le n$. The task execution requirement is $\bar{r} = 0.8, 0.9, 1.0, 1.1, 1.2$. The arrival rates of special tasks are $\lambda_i'' = 0.3m_i/\bar{x}_i$, for all $1 \le i \le n$. For each $\bar{r}$, the minimized average response time T' of generic tasks is plotted as a function of the total arrival rate of generic tasks λ' in two different cases, namely, special tasks without priority in Fig. 8 and special tasks with priority in Fig. 9. It is observed that slight increment of $\bar{r}$ noticeably increases the average response time T' of generic tasks, i.e., task execution requirement has significant impact on the average response time T' of generic tasks, especially when λ' is large.

In Figs. 10 and 11, we show the impact of arrival rates of special tasks on the average response time T' of generic tasks. We consider a

Fig. 6 T' vs. λ' and s (special tasks without priority)



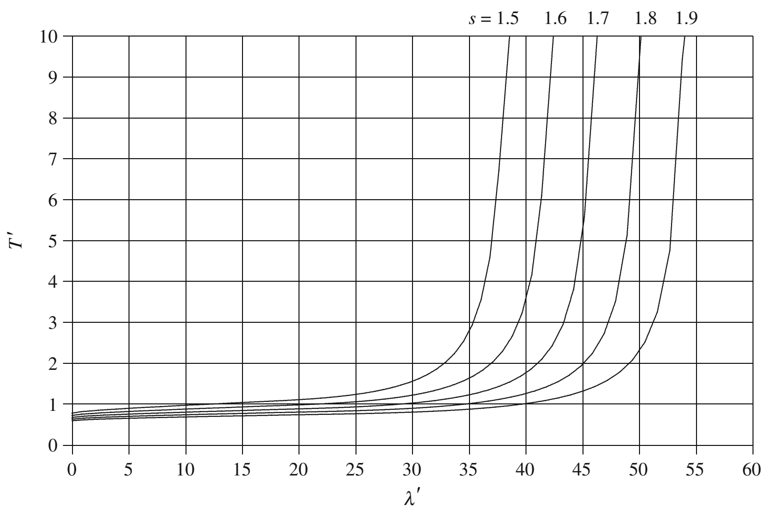


Fig. 7 T' vs. λ' and s (special tasks with priority)

group of $n = 7$ heterogeneous blade servers of sizes $m_i = 2i$, for all $1 \le i \le n$. The server speeds are $s_i = 1.7 - 0.1i$, for all $1 \le i \le n$. The task execution requirement is $\bar{r} = 1$. The arrival rates of special tasks are $\lambda_i'' = y(m_i/\bar{x}_i)$, for all $1 \le i \le n$, where $y = 0.20, 0.25, 0.30, 0.35, 0.40$. For each y, the minimized average response time T' of generic tasks is plotted as a function of the total arrival rate of generic tasks λ' in two different cases, namely, special tasks without priority in Fig. 10 and special tasks with priority in Fig. 11. It is observed that slight increment of the arrival rates of special tasks noticeably increases the average response time T' of generic tasks, i.e., the arrival rates of special tasks have significant impact on the average response time T' of generic tasks, especially when λ' is large.

Recall that the server utilization is

$$\rho_i = (\lambda_i' + \lambda_i'')\frac{\bar{r}}{m_i s_i} < 1,$$

which implies that

$$\lambda_i' < \frac{m_i s_i}{\bar{r}} - \lambda_i''.$$

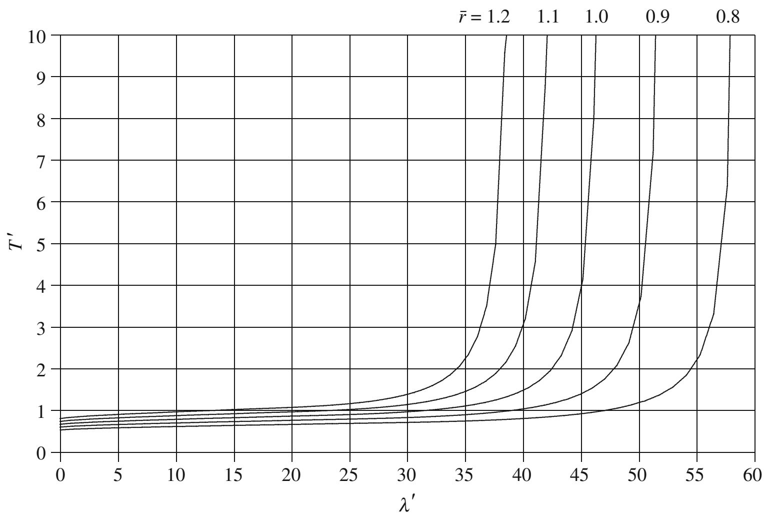


Fig. 8 T' vs. λ' and $\bar{r}$ (special tasks without priority)

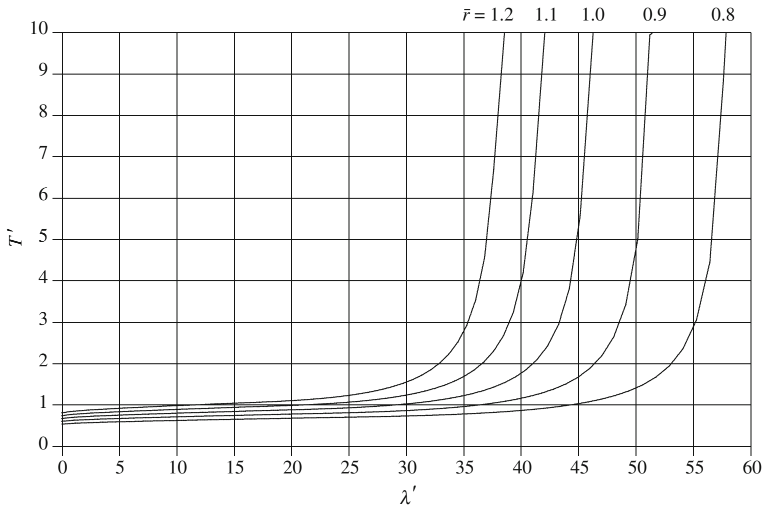


Fig. 9 T' vs. λ' and $\bar{r}$ (special tasks with priority)

The right-hand side of the above inequality is the saturation point of λ'_i. The maximum arrival rate of generic tasks is

$$\lambda'_{\max} = \sum_{i=1}^{n} \left(\frac{m_i s_i}{\bar{r}} - \lambda''_i \right).$$

The right-hand side of the above inequality is the saturation point of λ'. All reduction of the average response time T' of generic tasks is due to the increment of the saturation point of λ'. As a rule-of-thumb, to reduce T', we need to increase $\lambda'_{\max}$, e.g., to increase m_i or s_i, or to reduce $\bar{r}$ or λ''_i.

In Figs. 12 and 13, we show the impact of server size heterogeneity on the average response time T' of generic tasks. We consider five groups of $n = 7$ heterogeneous blade servers:

- Group 1: $(m_1, m_2, m_3, m_4, m_5, m_6, m_7) = (1, 2, 2, 8, 14, 14, 15)$;
- Group 2: $(m_1, m_2, m_3, m_4, m_5, m_6, m_7) = (2, 4, 6, 8, 10, 12, 14)$;

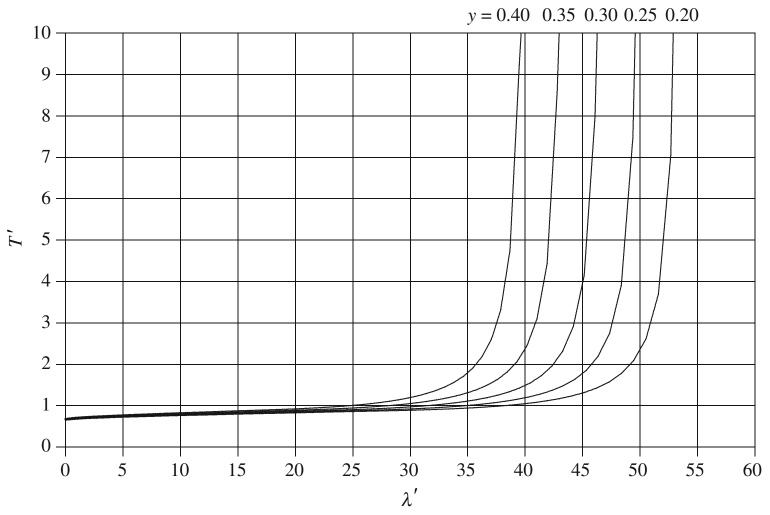


Fig. 10 T' vs. λ' and λ''_i (special tasks without priority)

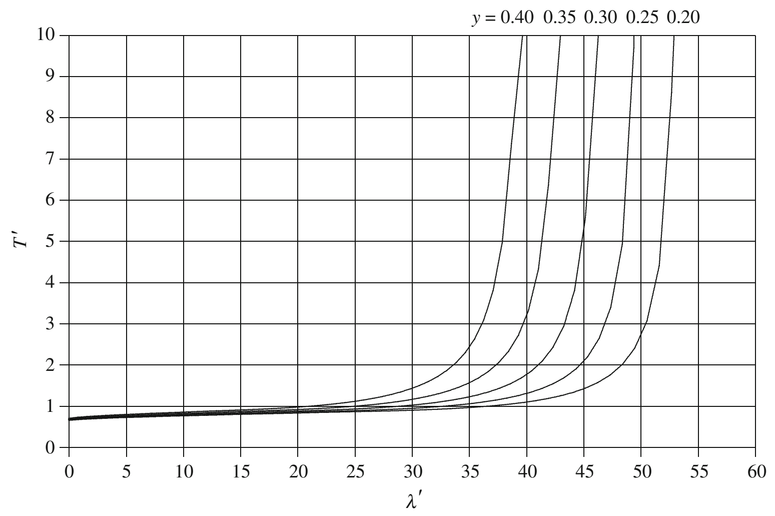


Fig. 11 T' vs. λ' and λ''_i (special tasks with priority)

- Group 3: $(m_1, m_2, m_3, m_4, m_5, m_6, m_7) = (4, 6, 6, 8, 10, 10, 12)$;
- Group 4: $(m_1, m_2, m_3, m_4, m_5, m_6, m_7) = (6, 6, 8, 8, 8, 10, 10)$;
- Group 5: $(m_1, m_2, m_3, m_4, m_5, m_6, m_7) = (8, 8, 8, 8, 8, 8, 8)$.

The server speeds are $s_i = 1.3$, for all $1 \le i \le n$. Hence, all the groups have the same total number of blades, i.e., $m = m_1 + m_2 + \cdots + m_n = 56$, and all the blades have the same speed $s = 1.3$. However, these groups have decreased degree of size heterogeneity, with Group 1 being the most heterogeneous and Group 5 being the least heterogeneous. The task execution requirement is $\bar{r} = 1$. The arrival rates of special tasks are $\lambda''_i = 0.3 m_i/\bar{x}_i$, for all $1 \le i \le n$. Notice that the total arrival rate of special tasks is

$$\lambda'' = \sum_{i=1}^{n} \lambda''_i = \sum_{i=1}^{n} 0.3 \frac{m_i}{\bar{x}_i} = 0.3 \sum_{i=1}^{n} \frac{m_i s}{\bar{r}}$$

$$= 0.3 \frac{s}{\bar{r}} \sum_{i=1}^{n} m_i = 0.3 \times 1.3 \times 56 = 21.84,$$

which is the same for all groups. For each group, the minimized average response time T' of generic tasks is plotted as a function of the total arrival

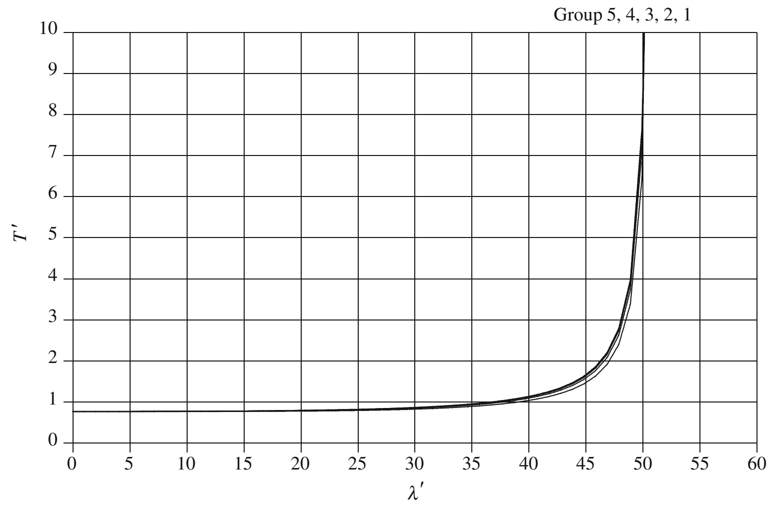


Fig. 12 T' vs. λ' and size heterogeneity (special tasks without priority)

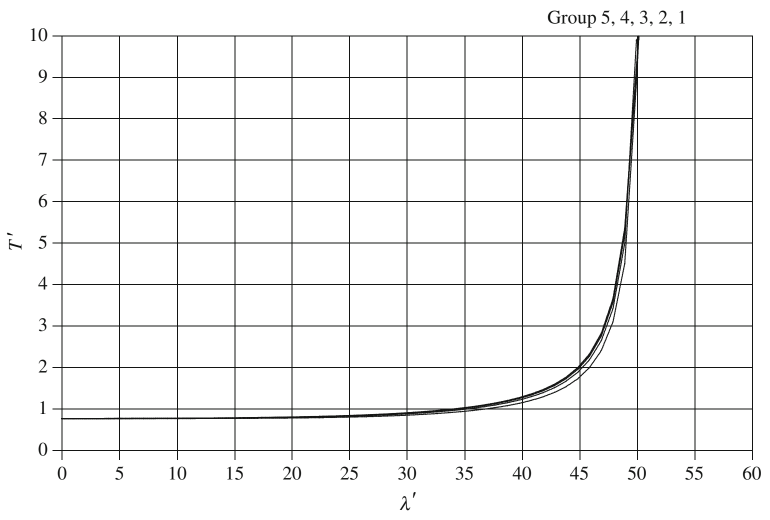


Fig. 13 T' vs. λ' and size heterogeneity (special tasks with priority)

rate of generic tasks λ' in two different cases, namely, special tasks without priority in Fig. 12 and special tasks with priority in Fig. 13. It is observed that although the five groups of servers have significant difference in sizes, they have almost identical average response time T' of generic tasks when the generic tasks are optimally distributed. In other words, the server size heterogeneity does not have much impact on the average response time T' of generic tasks. Surprisingly, from Group 1 to Group 5, T' is actually sightly increasing, i.e., larger (smaller, respectively) size heterogeneity results in shorter (longer, respectively) T'.

In Figs. 14 and 15, we show the impact of server speed heterogeneity on the average response time T' of generic tasks. We consider five groups of $n = 7$ heterogeneous blade servers:

- Group 1: $(s_1, s_2, s_3, s_4, s_5, s_6, s_7) = (0.1, 0.5, 0.9, 1.3, 1.7, 2.1, 2.5)$;
- Group 2: $(s_1, s_2, s_3, s_4, s_5, s_6, s_7) = (0.4, 0.7, 1.0, 1.3, 1.6, 1.9, 2.2)$;
- Group 3: $(s_1, s_2, s_3, s_4, s_5, s_6, s_7) = (0.7, 0.9, 1.1, 1.3, 1.5, 1.7, 1.9)$;
- Group 4: $(s_1, s_2, s_3, s_4, s_5, s_6, s_7) = (1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6)$;

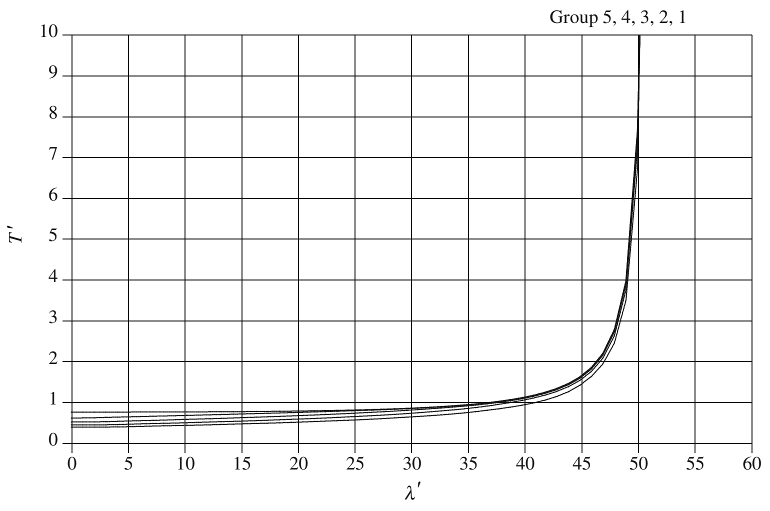


Fig. 14 T' vs. λ' and speed heterogeneity (special tasks without priority)

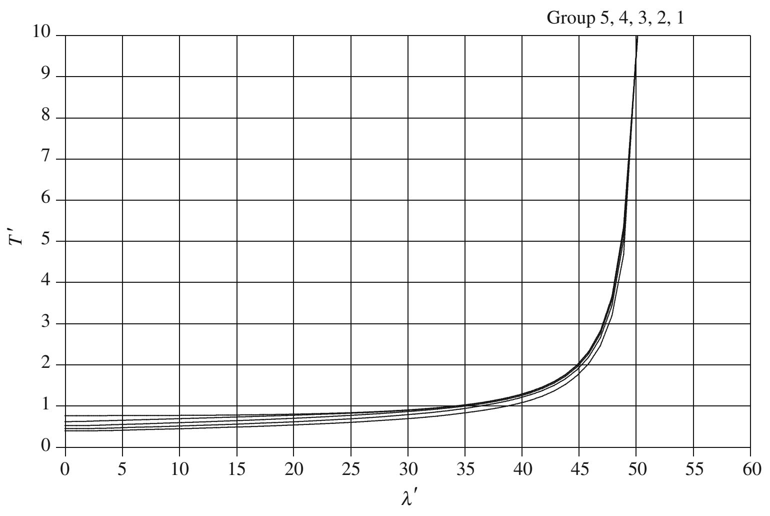


Fig. 15 T' vs. λ' and speed heterogeneity (special tasks with priority)

- Group 5: $(s_1, s_2, s_3, s_4, s_5, s_6, s_7) = (1.3, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3)$.

The server sizes are $m_i = 8$, for all $1 \leq i \leq n$. Hence, all the servers have the same size $m = 8$ and all the groups have the same total speed $m(s_1 + s_2 + \cdots + s_n) = 72.8$, but the blades have different speeds. These groups have decreased degree of speed heterogeneity, with Group 1 being the most heterogeneous and Group 5 being the least heterogeneous. The task execution requirement is $\bar{r} = 1$. The arrival rates of special tasks are $\lambda''_i = 0.3m_i/\bar{x}_i$, for all $1 \leq i \leq n$. Notice that the total arrival rate of special tasks is

$$\lambda'' = \sum_{i=1}^{n} \lambda''_i = \sum_{i=1}^{n} 0.3\frac{m_i}{\bar{x}_i} = 0.3\sum_{i=1}^{n}\frac{ms_i}{\bar{r}}$$
$$= 0.3\frac{m}{\bar{r}}\sum_{i=1}^{n} s_i = 0.3 \times 72.8 = 21.84,$$

which is the same for all groups. For each group, the minimized average response time T' of generic tasks is plotted as a function of the total arrival rate of generic tasks λ' in two different cases, namely, special tasks without priority in Fig. 14 and special tasks with priority in Fig. 15. It is observed that although the five groups of servers have significant difference in speeds, they have very close average response time T' of generic tasks when the generic tasks are optimally distributed. In other words, the server speed heterogeneity does not have much impact on the average response time T' of generic tasks. Surprisingly, from Group 1 to Group 5, T' is actually sightly increasing, i.e., larger (smaller, respectively) speed heterogeneity results in shorter (longer,respectively) T'.

6 Conclusions

We have proposed the problem of optimal load distribution of generic tasks on multiple heterogeneous blade servers preloaded with special tasks in a cloud computing environment. The problem is formulated as a multivariable optimization problem based on a queuing model. We developed algorithms to find the numerical solution of an optimal load distribution and the minimum average response time of generic tasks. We have considered two different situations of special tasks, namely, special tasks with and without higher priority. We also demonstrated extensive numerical examples and data. We found that server sizes, server speeds, task execution requirement, and the arrival rates of special tasks all have significant impact on the average response time of generic tasks, especially when the total arrival rate of generic tasks is large. We also found that the server size heterogeneity and the server speed heterogeneity do not have much impact on the average response time of generic tasks. Furthermore, larger (smaller, respectively) heterogeneity results in shorter (longer, respectively) average response time of generic tasks.

Acknowledgements The author wishes to thank three anonymous reviewers for their constructive comments and suggestions on improving the manuscript.

A preliminary version of the paper was presented on the *8th High-Performance Grid and Cloud Computing Workshop*, Anchorage, Alaska, May 16-20, 2011, and published in *Proceedings of the 25th IEEE International Parallel and Distributed Processing Symposium Workshops*, pp. 943–952.

References

1. Bonomi, F., Kumar, A.: Adaptive optimal load balancing in a nonhomogeneous multiserver system with a central job scheduler. IEEE Trans. Comput. **39**(10), 1232–1250 (1990)
2. He, L., Jarvis, S.A., Spooner, D.P., Jiang, H., Dillenberger, D.N., Nudd, G.R.: Allocating non-real-time and soft real-time jobs in multiclusters. IEEE Trans. Parallel Distrib. Syst. **17**(2), 99–112 (2006)
3. http://en.wikipedia.org/wiki/Cloud_computing. Accessed 17 May 2010
4. http://searchcloudcomputing.techtarget.com/sDefinition/0,,sid201_gci1287881,00.html. Accessed 17 May 2010
5. http://searchdatacenter.techtarget.com/sDefinition/0,,sid80_gci1070272,00.html. Accessed 17 May 2010
6. http://searchdatacenter.techtarget.com/sDefinition/0,,sid80_gci770169,00.html. Accessed 17 May 2010
7. Kameda, H., Li, J., Kim, C., Zhang, Y.: Optimal Load balancing in Distributed Computer Systems. Springer, London (1997)
8. Kleinrock, L.: Queueing Systems, Volume 1: Theory. Wiley, New York (1975)
9. Li, K.: Minimizing mean response time in heterogeneous multiple computer systems with a central stochastic job dispatcher. Int. J. Comput. Appl. **20**(1), 32–39 (1998)
10. Li, K.: Optimizing average job response time via decentralized probabilistic job dispatching in heterogeneous multiple computer systems. Comput. J. **41**(4), 223–230 (1998)
11. Li, K.: Minimizing the probability of load imbalance in heterogeneous distributed computer systems. Math. Comput. Model. **36**(9–10), 1075–1084 (2002)
12. Li, K.: Optimal load distribution in nondedicated heterogeneous cluster and Grid computing environments. J. Syst. Architect. **54**(1–2), 111–123 (2008)
13. Rommel, C.G.: The probability of load balancing success in a homogeneous network. IEEE Trans. Softw. Eng. **17**(9), 922–933 (1991)
14. Ross, K.W., Yao, D.D.: Optimal load balancing and scheduling in a distributed computer system. J. ACM **38**(3), 676–690 (1991)
15. Shirazi, B.A., Hurson, A.R., Kavi, K.M.: Scheduling and load balancing in parallel and distributed systems. In: IEEE Computer Society Press, Los Alamitos, California (1995)
16. Tang, X., Chanson, S.T.: Optimizing static job scheduling in a network of heterogeneous computers. In: Proceedings of International Conference on Parallel Processing, pp. 373–382. Toronto, Canada (2000)
17. Tantawi. A.N., Towsley, D.: Optimal static load balancing in distributed computer systems. J. ACM **32**(2), 445–465 (1985)