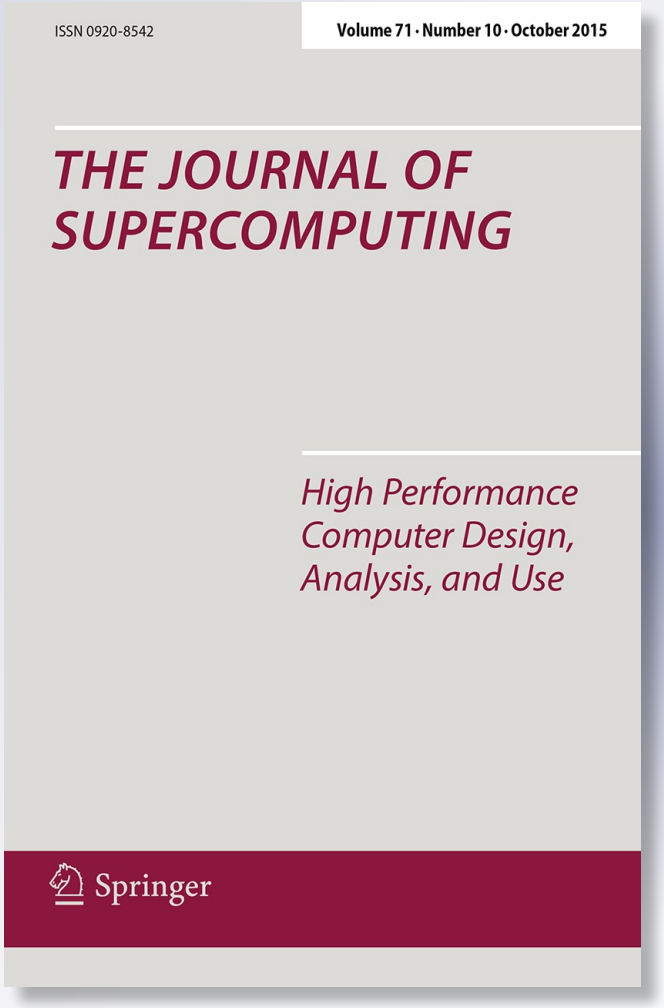
Optimal partitioning of a multicore server processor

Keqin Li

The Journal of Supercomputing
An International Journal of High-Performance Computer Design, Analysis, and Use

ISSN 0920-8542
Volume 71
Number 10

J Supercomput (2015) 71:3744-3769
DOI 10.1007/s11227-015-1463-3



Springer

J Supercomput (2015) 71:3744–3769
DOI 10.1007/s11227-015-1463-3

Optimal partitioning of a multicore server processor

Keqin Li[1]

Published online: 3 July 2015

Abstract Optimal partitioning of a multicore server processor in a cloud computing environment, i.e., optimal system (virtual server) configuration for some given types of applications is considered in this paper. Such optimization is important for dynamic resource provision and on-demand server customization in a cloud computing environment for certain specific types of applications, such that the overall system performance is optimized without exceeding certain energy consumption budget. A multicore server processor is treated as a group of queueing systems with multiple servers, i.e., M/M/m queueing systems. The system performance measures are the average task response time and the average power consumption. Two core speed and power consumption models are considered, namely, the idle-speed model and the constant-speed model. Three problems are formulated and solved, namely, optimal multicore server processor partitioning, optimal multicore server processor partitioning with power constraint, and optimal power allocation. All these problems are well-defined optimization problems. It is shown that although these problems are sophisticated, they can be solved by numerical algorithms. Numerical data are demonstrated for each problem.

Keywords Energy consumption · Multicore server processor · Processor partition · Queueing model · Response time

1 Introduction

Traditional single-CPU processors have been facing dual challenges and conflicting requirements of high computing speed and high energy efficiency. On the one hand, as the latest multimedia- and networking-based applications provide new features and cutting-edge capabilities, processor development needs to stay ahead of increased

✉ Keqin Li
lik@newpaltz.edu

[1] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

demands from software applications and provides increased computing speed and enhanced computing capabilities. On the other hand, it has been realized that increasing processor speed is not the only concern and solution. Modern computers need to run faster and cooler, occupy less space, and consume less energy. Power and energy efficiency should be a major consideration in modern processor technologies. The *multicore processor technology* is able to deal with these challenges simultaneously. Multicore processors offer true multitasking capabilities, so that users can simultaneously run multiple complex applications and successfully complete more tasks in a shorter amount of time. With the power of two or more processor cores on a single chip, multicore processors deliver leading performance and unique features that help systems run cooler and more efficient. Because they put more processing power into a smaller package, multicore processors help to build server infrastructures with a smaller footprint, reduced cooling needs, and increased energy efficiency [3].

Virtually, all major processor vendors such as AMD, Intel, and IBM are developing high performance and highly energy efficient multicore processors. Current 32 nm Intel logic technology provides for one, two, four, and eight cores in a single processor. Most of the current commercial multicore processors contain no more than 8 cores, with a few processors reaching 64 cores [19]. The general trend in processor development has been from multicore to manycore: from dual-, tri-, quad-, hexa-, octo-core chips to ones with tens or even hundreds of cores. It is conceivable that future architectures can hold dozens or even hundreds of processors on a single die [5]. For instance, Adapteva's Epiphany scalable manycore architecture consists of hundreds and thousands of RISC microprocessors, all sharing a single flat and unobstructed memory hierarchy, which allows cores to communicate with each other very efficiently with low core-to-core communication overhead. The number of cores in this new type of massively parallel multicore architecture can be up to 4096 [1]. The Epiphany manycore architecture has been designed to maximize floating point computing power with the lowest possible energy consumption, aiming to deliver 100 and more gigaflops of performance at under 2 W of power [6].

It is clear that as multicore server processors become larger and larger and more and more powerful, more applications can be executed on a multicore server processor. Such a large multicore server processor can be managed by the technique of server partitioning. *Server partitioning* involves the ability to divide a single large server into multiple smaller subsystems, with each partition (i.e., subsystem) running its own copy of an operating system [4]. Each partition acts as a physically independent and self-contained server with its own processor cores, main memory, input/output devices, and network resources. Server partitioning is extremely useful in multitier application environments. Each partition (which is also a multicore server) is employed to run one type of applications, such as enterprise resource planning, serving and caching Web pages, retrieving and managing databases, data warehousing, encrypting secure communications, and streaming multimedia. Due to the different nature of different types of applications, it is more efficient to partition a large multicore server processor into disjoint subsystems and to configure each subsystem so that its processor, memory, and networking features are best configured to fit the computation and communication requirements of one type of applications. Because of limited core and energy resources,

such partitioning should be performed in a way that the overall system performance is optimized based on the given resource constraints.

Server partitioning technologies offer unique advantages to information technology departments. It allows system administrators to host diversified applications on different partitions within a single server [17]. It allows administrators to consolidate multiple applications into one physical server box, thereby promoting centralized server management, saving space, and reducing administrative and management costs. It allows companies to consolidate the work previously done by multiple independent servers for different types of workloads into a single server. Server partitioning technology has been around for a while in the mainframe space and large-scale parallel processing systems [11,16,18], but it started to gain attention in distributed, grid, Internet computing only in the past few years. The trend toward server consolidation has driven much of the interest in server partitioning, which is likely to be adopted in future cloud computing, where server partitioning also implements virtual server configuration and provision.

In this paper, we consider the problem of *optimal partitioning of a multicore server processor* in a cloud computing environment, i.e., optimal system (virtual server) configuration for some given types of applications [9,12,20,21]. Such optimization is important for dynamic resource provision and on-demand server customization in a cloud computing environment for certain specific types of applications, such that the overall system performance is optimized without exceeding certain energy consumption budget. A multicore server processor is treated as a group of queueing systems with multiple servers, i.e., M/M/m queueing systems. The system performance measures are the average task response time and the average power consumption. Two core speed and power consumption models are considered, namely, the idle-speed model and the constant-speed model.

Three problems are formulated and solved.

- Optimal multicore server processor partitioning—given task arrival rates and mean task execution requirements for several types of applications, the number of available cores, and core speed, we find the server sizes such that the average task response time of all applications is minimized.
- Optimal multicore server processor partitioning with power constraint—given task arrival rates and mean task execution requirements for several types of applications, the number of available cores, and the total available power, we find the server sizes and the server speeds such that the average task response time of all applications is minimized and that the total average power consumption does not exceed the total available power.
- Optimal power allocation—given task arrival rates and mean task execution requirements for several types of applications, the server sizes, and the total available power, we find the server speeds such that the average task response time of all applications is minimized and that the total average power consumption does not exceed the total available power.

All the above problems are well-defined optimization problems. We show that although these problems are sophisticated, they can be solved by numerical algorithms. We demonstrate numerical data for each problem.

It has come to the author's attention that while server partitioning is critical to virtual server configuration and provision, the problem of optimal partitioning of a multicore server processor in a cloud computing environment has not been treated rigorously in an analytical way. Our investigation in this paper makes effort in this direction, and the method can be applied to dynamic resource provision, system performance optimization, and energy consumption reduction in cloud computing.

2 Modeling a multicore server processor

Assume that a multicore server processor S has m identical cores. In this paper, a multicore server processor is treated as an M/M/m queueing system which is elaborated as follows [14]. There is a Poisson stream of 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 multicore server S maintains a queue with infinite capacity for waiting tasks when all the m cores are busy. The first-come-first-served (FCFS) queueing discipline is adopted. 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 cores of server S have identical execution speed s (measured by the number of instructions that can be executed in one unit of time). Hence, the task execution times on the cores of server S are i.i.d. exponential random variables $x = r/s$ with mean $\bar{x} = \bar{r}/s$.

Let $\mu = 1/\bar{x} = s/\bar{r}$ be the average service rate, i.e., the average number of tasks that can be finished by a processor core of server S in one unit of time. The core utilization is

$$\rho = \frac{\lambda}{m\mu} = \frac{\lambda\bar{x}}{m} = \frac{\lambda}{m} \cdot \frac{\bar{r}}{s},$$

which is the average percentage of time that a core of S is busy. Let p_k denote the probability that there are k tasks (waiting or being processed) in the M/M/m system for S. Then, we have ([13], p. 102)

$$p_k = \begin{cases} p_0 \dfrac{(m\rho)^k}{k!}, & k \leq m; \\ \\ p_0 \dfrac{m^m \rho^k}{m!}, & k \geq m; \end{cases}$$

where

$$p_0 = \left(\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!} \cdot \frac{1}{1-\rho} \right)^{-1}.$$

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

$$P_q = \frac{p_m}{1-\rho} = p_0 \frac{(m\rho)^m}{m!} \cdot \frac{1}{1-\rho}.$$

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

$$\bar{N} = \sum_{k=0}^{\infty} k p_k = m\rho + \frac{\rho}{1-\rho} P_q.$$

Applying Little's result, we get the average task response time as

$$T = \frac{\bar{N}}{\lambda} = \bar{x} + \frac{P_q}{m(1-\rho)}\bar{x} = \bar{x}\left(1 + \frac{P_q}{m(1-\rho)}\right) = \bar{x}\left(1 + \frac{p_m}{m(1-\rho)^2}\right).$$

To formulate and solve our optimization problems analytically, we need a closed-form expression of T. To this end, let us use the following closed-form approximation,

$$\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} \approx e^{m\rho},$$

which is very accurate when m is not too small and ρ is not too large. We also need Stirling's approximation of $m!$, i.e.,

$$m! \approx \sqrt{2\pi m}\left(\frac{m}{e}\right)^m.$$

Therefore, we get the following closed-form approximation of p_0,

$$p_0 \approx \left(e^{m\rho} + \frac{(e\rho)^m}{\sqrt{2\pi m}} \cdot \frac{1}{1-\rho}\right)^{-1},$$

and the following closed-form approximation of p_m,

$$p_m \approx \frac{\frac{(e\rho)^m}{\sqrt{2\pi m}}}{e^{m\rho} + \frac{(e\rho)^m}{\sqrt{2\pi m}} \cdot \frac{1}{1-\rho}},$$

namely,

$$p_m \approx \frac{1-\rho}{\sqrt{2\pi m}(1-\rho)(e^\rho/e\rho)^m + 1}.$$

Using the above closed-form expression of p_m, we get a closed-form approximation of the average task response time as

$$T \approx \frac{\bar{r}}{s}\left(1 + \frac{1}{m(1-\rho)(\sqrt{2\pi m}(1-\rho)(e^\rho/e\rho)^m + 1)}\right).$$

Our discussion in this paper is based on the above closed-form expression.

Power dissipation and circuit delay in digital CMOS circuits can be accurately modeled by simple equations, even for complex microprocessor circuits. CMOS circuits have dynamic, static, and short-circuit power dissipation; however, the dominant component in a well-designed circuit is dynamic power consumption P (i.e., the switching component of power), which is approximately $P = aCV^2 f$, where a is an activity factor, C is the loading capacitance, V is the supply voltage, and f is the clock frequency [10]. Since $s \propto f$, where s is the processor speed, and $f \propto V^{\phi}$ with $0 < \phi \leq 1$ [22], which implies that $V \propto f^{1/\phi}$, we know that power consumption is $P \propto f^{\alpha}$ and $P \propto s^{\alpha}$, where $\alpha = 1 + 2/\phi \geq 3$. For ease of discussion, we will assume that the power allocated to a processor core with speed s is simply s^{α}.

We will consider two types of core speed models. In the *idle-speed model*, a core runs at zero speed when there is no task to perform. Since the power for speed s is s^{α}, the average amount of energy consumed by a core in one unit of time is

$$\rho s^{\alpha} = \frac{\lambda}{m} \bar{r} s^{\alpha - 1},$$

where we notice that the speed of a core is zero when it is idle. The average amount of energy consumed by an m-core server S in one unit of time, i.e., the power supply to server S, is

$$P = m\rho s^{\alpha} = \lambda \bar{r} s^{\alpha - 1},$$

where $m\rho = \lambda \bar{x}$ is the average number of busy cores in S. Since a processor core still consumes some amount of power P^* even when it is idle (assume that an idle core consumes certain base power P^*, which includes static power dissipation, short-circuit power dissipation, and other leakage and wasted power [2]), we will include P^* in P, i.e.,

$$P = m(\rho s^{\alpha} + P^*) = \lambda \bar{r} s^{\alpha - 1} + mP^*.$$

Notice that when $P^* = 0$, the above P is independent of m.

In the *constant-speed model*, all cores run at the speed s even if there is no task to perform. Again, we use P to represent the power allocated to server S. Since the power for speed s is s^{α}, the power allocated to server S is $P = m(s^{\alpha} + P^*)$.

Notice that the above two core speed models which characterize different ways of power consumption have been used in studying various aspects of multicore server processors [7,8,14].

3 Optimal processor partitioning

Assume that we have a multicore server processor with m cores of the same speed s. There are n types of applications, such that the task arrival rate of the ith type is λ_i, and the task execution requirements of the ith type are i.i.d. exponential random variables with mean $\bar{r}_i$, where $1 \leq i \leq n$. Let $\lambda = \lambda_1 + \lambda_2 + \cdots + \lambda_n$. We divide the m

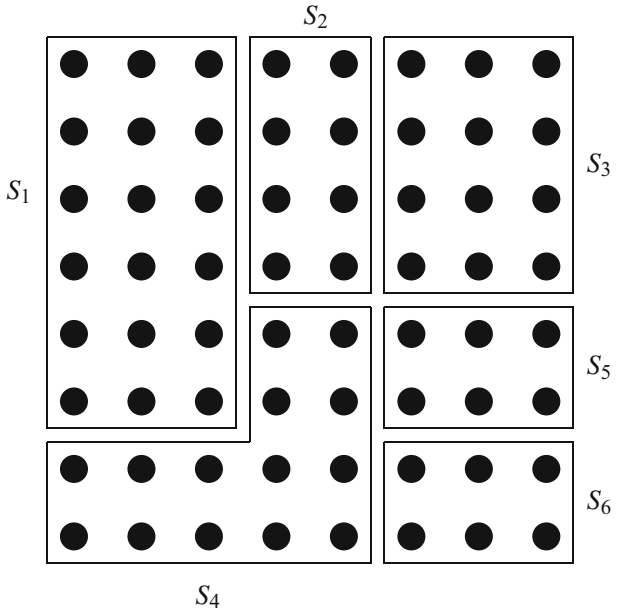


Fig. 1 A multicore server processor partitioned into several servers

cores into n servers $S_1, S_2, \ldots, S_n$, such that S_i contains m_i cores, where $1 \leq i \leq n$, and $m_1 + m_2 + \cdots + m_n = m$. In Fig. 1, we illustrate a 64-core server processor which is divided into several servers. The average task response time of S_i is

$$T_i = \frac{\bar{r}_i}{s}\left(1 + \frac{1}{m_i(1-\rho_i)(\sqrt{2\pi m_i}(1-\rho_i)(e^{\rho_i}/e\rho_i)^{m_i}+1)}\right),$$

where

$$\rho_i = \frac{\lambda_i \bar{r}_i}{m_i s}.$$

The average task response time of all the n types of applications is

$$T = \frac{\lambda_1}{\lambda}T_1 + \frac{\lambda_2}{\lambda}T_2 + \cdots + \frac{\lambda_n}{\lambda}T_n.$$

Our *optimal multicore server processor partitioning* problem can be formally defined as follows. Given task arrival rates $\lambda_1, \lambda_2, \ldots, \lambda_n$, mean task execution requirements $\bar{r}_1, \bar{r}_2, \ldots, \bar{r}_n$, the number of available cores m, and core speed s, we find $m_1, m_2, \ldots, m_n$ such that T is minimized subject to the constraint that

$$J(m_1, m_2, \ldots, m_n) = m,$$

where

$$J(m_1, m_2, \ldots, m_n) = m_1 + m_2 + \cdots + m_n.$$

We can minimize T using the method of Lagrange multiplier, namely,

$$\nabla T(m_1, m_2, \ldots, m_n) = \phi \nabla J(m_1, m_2, \ldots, m_n),$$

that is,

$$\frac{\partial T}{\partial m_i} = \phi \frac{\partial J}{\partial m_i} = \phi,$$

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

Strictly speaking, the above problem and the problem of the next section are discrete and combinatorial optimization problems, since the m_is are integers. It is not clear whether there exist any efficient algorithms to solve these problems. Our strategy to solve the problems is a two-step process. First, we treat the m_is as real numbers and the problems as continuous and multi-variable optimization problems, so that the problems can be solved using standard methods from multi-variable calculus. Second, once the optimal real values of the m_is are obtained, they are rounded to the nearest integers. In addition, for the purpose of formulating our multi-variable optimization problems, we employ a closed-form approximation of the average task response time, as we have done in Sect. 2. Although there is no rigorous proof of the ultimate optimality of our approach, our extensive numerical calculations demonstrate its effectiveness.

Let us rewrite T_i as

$$T_i = \frac{\bar{r}_i}{s}(1 + F_i),$$

where

$$F_i = \frac{1}{m_i(1-\rho_i)(\sqrt{2\pi m_i}(1-\rho_i)(e^{\rho_i}/e\rho_i)^{m_i} + 1)}.$$

It is clear that

$$\frac{\partial T}{\partial m_i} = \frac{\lambda_i}{\lambda} \cdot \frac{\partial T_i}{\partial m_i} = \frac{\lambda_i}{\lambda} \cdot \frac{\bar{r}_i}{s} \cdot \frac{\partial F_i}{\partial m_i} = \frac{m_i \rho_i}{\lambda} \cdot \frac{\partial F_i}{\partial m_i},$$

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

We rewrite F_i as

$$F_i = \frac{1}{m_i(1-\rho_i)(\sqrt{2\pi m_i}(1-\rho_i)G_i + 1)} = \frac{1}{\sqrt{2\pi} m_i^{3/2}(1-\rho_i)^2 G_i + m_i(1-\rho_i)},$$

where

$$G_i = (e^{\rho_i}/e\rho_i)^{m_i}.$$

Notice that

$$\ln G_i = m_i \ln(e^{\rho_i}/e\rho_i) = m_i(\rho_i - \ln \rho_i - 1).$$

Since

$$\frac{\partial \rho_i}{\partial m_i} = -\frac{\lambda_i \bar{r}_i}{m_i^2 s} = -\frac{\rho_i}{m_i},$$

we get

$$\frac{1}{G_i}\frac{\partial G_i}{\partial m_i} = (\rho_i - \ln \rho_i - 1) + m_i\left(1 - \frac{1}{\rho_i}\right)\frac{\partial \rho_i}{\partial m_i} = -\ln \rho_i,$$

and

$$\frac{\partial G_i}{\partial m_i} = -G_i \ln \rho_i.$$

Now, we have

$$\begin{aligned}
\frac{\partial F_i}{\partial m_i} &= -F_i^2\left(\sqrt{2\pi}\left(\frac{3}{2}\sqrt{m_i}(1-\rho_i)^2 G_i + m_i^{3/2} 2(1-\rho_i)\left(-\frac{\partial \rho_i}{\partial m_i}\right)G_i + m_i^{3/2}(1-\rho_i)^2\frac{\partial G_i}{\partial m_i}\right)\right.\\
&\quad \left.+ (1-\rho_i) + m_i\left(-\frac{\partial \rho_i}{\partial m_i}\right)\right)\\
&= -F_i^2\left(\sqrt{2\pi}\left(\frac{3}{2}\sqrt{m_i}(1-\rho_i)^2 G_i + \sqrt{m_i}2\rho_i(1-\rho_i)G_i - m_i^{3/2}(\ln \rho_i)(1-\rho_i)^2 G_i\right)\right.\\
&\quad \left.+ (1-\rho_i) + \rho_i\right)\\
&= -F_i^2\left(\sqrt{2\pi m_i}(1-\rho_i)\left(\frac{3}{2}(1-\rho_i) + 2\rho_i - m_i(\ln \rho_i)(1-\rho_i)\right)G_i + 1\right)\\
&= -F_i^2\left(\sqrt{2\pi m_i}(1-\rho_i)\left(\frac{\rho_i + 3}{2} - m_i(\ln \rho_i)(1-\rho_i)\right)G_i + 1\right),
\end{aligned}$$

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

Summarizing the above discussion, we get

$$\frac{\partial T}{\partial m_i} = -\frac{m_i \rho_i}{\lambda} F_i^2(\sqrt{2\pi m_i}(1-\rho_i)((\rho_i + 3)/2 - m_i(\ln \rho_i)(1-\rho_i))G_i + 1) = \phi,$$

for all $1 \le i \le n$. Hence, we get a nonlinear system of $(n+1)$ equations specified in the last equation and the constraint $J(m_1, m_2, \ldots, m_n) = m$. It is unlikely that this nonlinear system of equations accommodates any analytical solution.

To solve the above equations numerically, we notice that $\partial T/\partial m_i < 0$ (i.e., T is a decreasing function of m_i) and is an increasing function of m_i (i.e., T is a convex function of m_i). Hence, given $\lambda_i, \bar{r}_i, s, \lambda$, and ϕ, we can find m_i that satisfies $\partial T/\partial m_i = \phi$ using the classic bisection method, i.e., searching for m_i in an interval $[lb, ub]$. The lower bound is simply $lb = 0$. The upper bound ub should be some value greater than $\lambda_i \bar{r}_i/s$ such that $\rho_i < 1$ and should be large enough such that $\partial T/\partial m_i$ with $m_i = ub$ is greater than ϕ.

Given n, m, $\lambda_1, \lambda_2, \ldots, \lambda_n$, $\bar{r}_1, \bar{r}_2, \ldots, \bar{r}_n$, and s, the optimal multicore server processor partitioning problem can be solved as follows. Again, we find ϕ using the bisection method, i.e., searching for ϕ in an interval $[lb, ub]$. Since $\partial T/\partial m_i < 0$ for all $1 \leq i \leq n$, the upper bound is simply $ub = 0$. The lower bound lb is chosen such that $m_1 + m_2 + \cdots + m_n < m$, where m_i is determined with $\phi = lb$. As the search interval $[lb, ub]$ shrinks, we will eventually obtain the m_is which satisfy $J(m_1, m_2, \ldots, m_n) = m$ with arbitrary numerical accuracy.

Example 1 Let us consider $m = 64$ cores to be partitioned into $n = 8$ subsystems. The core speed is $s = 1$. The task arrival rates are $\lambda_i = ((i + 5)/76)\lambda$, for all $1 \leq i \leq n$, where $\lambda = 12, 24, 36, 48, 60$. The mean task execution requirements are $\bar{r}_i = 1$ for all $1 \leq i \leq n$. In Table 1, for each λ, we show λ_i, m_i, ρ_i, and T_i, for all $1 \leq i \leq n$, as well as the minimized average task response time T. All the data are calculated with the length of a search interval reduced to no longer than 10^{-14}. Notice that all the T_is and T are calculated based on the m_is which are not integers. In reality, such fractional servers cannot be implemented. Hence, the m_is should be rounded to the nearest integers, and the T_is and T need to be re-calculated. In Table 2, we demonstrate the same information as Table 1 after the m_is are rounded to the nearest integers. It is observed that rounding the m_is damages the data smooth. For instances, the ρ_is are no longer increasing with i and the T_is are no longer decreasing with i. Furthermore, when λ is large, such rounding may cause significant increment of some ρ_i (e.g., when $\lambda = 60$, ρ_5 is over 98 %). Since T_i has sharp turn and dramatic increase when ρ_i is close to 1, such increment of ρ_i may cause significant increment of T_i and T as well (e.g., when $\lambda = 60$, T_5 increases from 2.3303738 to 9.6888741, and T increases from 2.3551706 to 3.3894671).

To demonstrate the optimality of our solution, we perform extra computation as follows. Let us consider the case when $\lambda = 48$. For each m_i, its fraction part is truncated, and we obtain

$$(m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8) = (5, 6, 6, 7, 8, 9, 9, 10).$$

The four additional cores are allocated to the eight servers in the following way, i.e., we choose four servers out of the eight and allocate one extra core to each server. It is clear that there are $\binom{8}{4} = 70$ different ways. For all these 70 different partitions of the m cores, we calculate the average task response time and show the results in Table 3. It can be seen that the optimal partition is indeed No. 18 (in boldface):

$$(m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8) = (5, 6, 7, 8, 8, 9, 10, 11),$$

Table 1 Numerical data for Example 1 (nonintegral m_is)

i	λ_i	m_i	ρ_i	T_i
$\lambda = 12.0$, $T = 1.0000250$				
1	0.9473684	6.4514121	0.1468467	1.0000347
2	1.1052632	6.9331540	0.1594171	1.0000311
3	1.2631579	7.3905088	0.1709162	1.0000283
4	1.4210526	7.8281601	0.1815309	1.0000261
5	1.5789474	8.2494732	0.1913998	1.0000242
6	1.7368421	8.6569601	0.2006296	1.0000227
7	1.8947368	9.0525537	0.2093041	1.0000214
8	2.0526316	9.4377781	0.2174910	1.0000202
$\lambda = 24.0$, $T = 1.0024579$				
1	1.8947368	6.0226870	0.3145999	1.0033208
2	2.2105263	6.6205406	0.3338891	1.0030055
3	2.5263158	7.1980626	0.3509716	1.0027585
4	2.8421053	7.7590715	0.3662945	1.0025589
5	3.1578947	8.3063258	0.3801795	1.0023937
6	3.4736842	8.8418950	0.3928665	1.0022541
7	3.7894737	9.3673778	0.4045394	1.0021344
8	4.1052632	9.8840397	0.4153426	1.0020304
$\lambda = 36.0$, $T = 1.0267450$				
1	2.8421053	5.6823799	0.5001611	1.0354128
2	3.3157895	6.3688886	0.5206229	1.0322786
3	3.7894737	7.0404076	0.5382463	1.0298051
4	4.2631579	7.6997116	0.5536776	1.0277928
5	4.7368421	8.3488133	0.5673671	1.0261168
6	5.2105263	8.9892273	0.5796412	1.0246943
7	5.6842105	9.6221272	0.5907436	1.0234686
8	6.1578947	10.2484443	0.6008614	1.0223988
$\lambda = 48.0$, $T = 1.1439974$				
1	3.7894737	5.3952278	0.7023751	1.1864634
2	4.4210526	6.1537637	0.7184307	1.1713207
3	5.0526316	6.9035007	0.7318941	1.1592493
4	5.6842105	7.6460485	0.7434181	1.1493436
5	6.3157895	8.3825789	0.7534423	1.1410311
6	6.9473684	9.1139761	0.7622764	1.1339298
7	7.5789474	9.8409269	0.7701457	1.1277743
8	8.2105263	10.5639775	0.7772192	1.1223737
$\lambda = 60.0$, $T = 2.3551706$				
1	4.7368421	5.1370964	0.9220855	2.7230421
2	5.5263158	5.9586854	0.9274388	2.5935556
3	6.3157895	6.7780520	0.9318001	2.4893641
4	7.1052632	7.5956024	0.9354443	2.4031870

Table 1 continued

i	λ_i	m_i	ρ_i	T_i
5	7.8947368	8.4116321	0.9385499	2.3303738
6	8.6842105	9.2263644	0.9412386	2.2677971
7	9.4736842	10.0399726	0.9435966	2.2132661
8	10.2631579	10.8525947	0.9456870	2.1651942

Table 2 Numerical data for Example 1 (integral m_is)

i	λ_i	m_i	ρ_i	T_i
$\lambda = 12.0$, $T = 1.0000364$				
1	0.9473684	6.0000000	0.1578947	1.0000927
2	1.1052632	7.0000000	0.1578947	1.0000270
3	1.2631579	7.0000000	0.1804511	1.0000619
4	1.4210526	8.0000000	0.1776316	1.0000186
5	1.5789474	8.0000000	0.1973684	1.0000387
6	1.7368421	9.0000000	0.1929825	1.0000120
7	1.8947368	9.0000000	0.2105263	1.0000235
8	2.0526316	9.0000000	0.2280702	1.0000431
$\lambda = 24.0$, $T = 1.0025887$				
1	1.8947368	6.0000000	0.3157895	1.0034388
2	2.2105263	7.0000000	0.3157895	1.0017184
3	2.5263158	7.0000000	0.3609023	1.0036278
4	2.8421053	8.0000000	0.3552632	1.0018526
5	3.1578947	8.0000000	0.3947368	1.0035339
6	3.4736842	9.0000000	0.3859649	1.0018521
7	3.7894737	9.0000000	0.4210526	1.0033011
8	4.1052632	10.0000000	0.4105263	1.0017737
$\lambda = 36.0$, $T = 1.0278840$				
1	2.8421053	6.0000000	0.4736842	1.0240583
2	3.3157895	6.0000000	0.5526316	1.0491241
3	3.7894737	7.0000000	0.5413534	1.0311350
4	4.2631579	8.0000000	0.5328947	1.0203740
5	4.7368421	8.0000000	0.5921053	1.0368101
6	5.2105263	9.0000000	0.5789474	1.0244432
7	5.6842105	10.0000000	0.5684211	1.0165795
8	6.1578947	10.0000000	0.6157895	1.0278915
$\lambda = 48.0$, $T = 1.1525194$				
1	3.7894737	5.0000000	0.7578947	1.3156170
2	4.4210526	6.0000000	0.7368421	1.2055178
3	5.0526316	7.0000000	0.7218045	1.1435108
4	5.6842105	8.0000000	0.7105263	1.1048626
5	6.3157895	8.0000000	0.7894737	1.2093855
6	6.9473684	9.0000000	0.7719298	1.1493165

Table 2 continued

i	λ_i	m_i	ρ_i	T_i
7	7.5789474	10.0000000	0.7578947	1.1108489
8	8.2105263	11.0000000	0.7464115	1.0846752
$\lambda = 60.0, T = 3.3894671$				
1	4.7368421	5.0000000	0.9473684	3.9295564
2	5.5263158	6.0000000	0.9210526	2.4126818
3	6.3157895	7.0000000	0.9022556	1.8742025
4	7.1052632	8.0000000	0.8881579	1.6083999
5	7.8947368	8.0000000	0.9868421	9.6888741
6	8.6842105	9.0000000	0.9649123	3.5025880
7	9.4736842	10.0000000	0.9473684	2.3349963
8	10.2631579	11.0000000	0.9330144	1.8636161

Table 3 Optimality of Example 1

Number	$(m_1, m_2, \ldots, m_8)$	T (approximation)	T (original)	Relative error (%)
1	(5, 6, 6, 7, 9, 10, 10, 11)	1.1861112	1.2280137	3.4122167
2	(5, 6, 6, 8, 8, 10, 10, 11)	1.1787331	1.2185658	3.2688176
3	(5, 6, 6, 8, 9, 9, 10, 11)	1.1743611	1.2128488	3.1733333
4	(5, 6, 6, 8, 9, 10, 9, 11)	1.1914442	1.2346613	3.5003177
5	(5, 6, 6, 8, 9, 10, 10, 10)	1.1824834	1.2233516	3.3406813
6	(5, 6, 7, 7, 8, 10, 10, 11)	1.1642696	1.2007067	3.0346368
7	(5, 6, 7, 7, 9, 9, 10, 11)	1.1598976	1.1949897	2.9366052
8	(5, 6, 7, 7, 9, 10, 9, 11)	1.1769807	1.2168022	3.2726324
9	(5, 6, 7, 7, 9, 10, 10, 10)	1.1680199	1.2054926	3.1084949
10	(5, 7, 6, 7, 8, 10, 10, 11)	1.1906762	1.2339132	3.5040558
11	(5, 7, 6, 7, 9, 9, 10, 11)	1.1863042	1.2281962	3.4108597
12	(5, 7, 6, 7, 9, 10, 9, 11)	1.2033873	1.2500087	3.7296846
13	(5, 7, 6, 7, 9, 10, 10, 10)	1.1944265	1.2386991	3.5741203
14	(6, 6, 6, 7, 8, 10, 10, 11)	1.1856146	1.2272941	3.3960509
15	(6, 6, 6, 7, 9, 9, 10, 11)	1.1812425	1.2215771	3.3018443
16	(6, 6, 6, 7, 9, 10, 9, 11)	1.1983257	1.2433896	3.6242789
17	(6, 6, 6, 7, 9, 10, 10, 10)	1.1893648	1.2320800	3.4669112
18	**(5, 6, 7, 8, 8, 9, 10, 11)**	**1.1525194**	**1.1855418**	**2.7854213**
19	(5, 6, 7, 8, 8, 10, 9, 11)	1.1696026	1.2073542	3.1268094
20	(5, 6, 7, 8, 8, 10, 10, 10)	1.1606417	1.1960446	2.9599964
21	(5, 6, 7, 8, 9, 9, 9, 11)	1.1652305	1.2016373	3.0297586
22	(5, 6, 7, 8, 9, 9, 10, 10)	1.1562697	1.1903276	2.8612224
23	(5, 6, 7, 8, 9, 10, 9, 10)	1.1733529	1.2121401	3.1998984
24	(5, 7, 6, 8, 8, 9, 10, 11)	1.1789260	1.2187483	3.2674716
25	(5, 7, 6, 8, 8, 10, 9, 11)	1.1960092	1.2405608	3.5912458

Table 3 continued

Number	$(m_1, m_2, \ldots, m_8)$	T (approximation)	T (original)	Relative error (%)
26	(5, 7, 6, 8, 8, 10, 10, 10)	1.1870483	1.2292511	3.4332120
27	(5, 7, 6, 8, 9, 9, 9, 11)	1.1916371	1.2348438	3.4989550
28	(5, 7, 6, 8, 9, 9, 10, 10)	1.1826763	1.2235342	3.3393298
29	(5, 7, 6, 8, 9, 10, 9, 10)	1.1997595	1.2453466	3.6606011
30	(5, 7, 7, 7, 8, 9, 10, 11)	1.1644626	1.2008892	3.0333064
31	(5, 7, 7, 7, 8, 10, 9, 11)	1.1815457	1.2227017	3.3659871
32	(5, 7, 7, 7, 8, 10, 10, 10)	1.1725849	1.2113921	3.2035205
33	(5, 7, 7, 7, 9, 9, 9, 11)	1.1771737	1.2169847	3.2712838
34	(5, 7, 7, 7, 9, 9, 10, 10)	1.1682128	1.2056751	3.1071585
35	(5, 7, 7, 7, 9, 10, 9, 10)	1.1852960	1.2274876	3.4372298
36	(6, 6, 6, 8, 8, 9, 10, 11)	1.1738644	1.2121291	3.1568234
37	(6, 6, 6, 8, 8, 10, 9, 11)	1.1909475	1.2339416	3.4842904
38	(6, 6, 6, 8, 8, 10, 10, 10)	1.1819867	1.2226320	3.3244117
39	(6, 6, 6, 8, 9, 9, 9, 11)	1.1865755	1.2282246	3.3910044
40	(6, 6, 6, 8, 9, 9, 10, 10)	1.1776147	1.2169150	3.2295077
41	(6, 6, 6, 8, 9, 10, 9, 10)	1.1946978	1.2387275	3.5544295
42	(6, 6, 7, 7, 8, 9, 10, 11)	1.1594009	1.1942701	2.9197058
43	(6, 6, 7, 7, 8, 10, 9, 11)	1.1764840	1.2160826	3.2562349
44	(6, 6, 7, 7, 8, 10, 10, 10)	1.1675232	1.2047729	3.0918454
45	(6, 6, 7, 7, 9, 9, 9, 11)	1.1721120	1.2103656	3.1604953
46	(6, 6, 7, 7, 9, 9, 10, 10)	1.1631512	1.1990559	2.9944190
47	(6, 6, 7, 7, 9, 10, 9, 10)	1.1802343	1.2208684	3.3282941
48	(6, 7, 6, 7, 8, 9, 10, 11)	1.1858075	1.2274766	3.3946955
49	(6, 7, 6, 7, 8, 10, 9, 11)	1.2028907	1.2492891	3.7139863
50	(6, 7, 6, 7, 8, 10, 10, 10)	1.1939298	1.2379795	3.5581881
51	(6, 7, 6, 7, 9, 9, 9, 11)	1.1985186	1.2435721	3.6229076
52	(6, 7, 6, 7, 9, 9, 10, 10)	1.1895578	1.2322625	3.4655507
53	(6, 7, 6, 7, 9, 10, 9, 10)	1.2066409	1.2540749	3.7823905
54	(5, 7, 7, 8, 8, 9, 9, 11)	1.1697955	1.2075368	3.1254723
55	(5, 7, 7, 8, 8, 9, 10, 10)	1.1608347	1.1962271	2.9586722
56	(5, 7, 7, 8, 8, 10, 9, 10)	1.1779178	1.2180396	3.2939628
57	(5, 7, 7, 8, 9, 9, 9, 10)	1.1735458	1.2123226	3.1985557
58	(6, 6, 7, 8, 8, 9, 9, 11)	1.1647339	1.2009176	3.0130085
59	(6, 6, 7, 8, 8, 9, 10, 10)	1.1557730	1.1896080	2.8442111
60	(6, 6, 7, 8, 8, 10, 9, 10)	1.1728562	1.2114205	3.1833946
61	(6, 6, 7, 8, 9, 9, 9, 10)	1.1684841	1.2057035	3.0869395
62	(6, 7, 6, 8, 8, 9, 9, 11)	1.1911405	1.2341241	3.4829293
63	(6, 7, 6, 8, 8, 9, 10, 10)	1.1821796	1.2228145	3.3230619
64	(6, 7, 6, 8, 8, 10, 9, 10)	1.1992628	1.2446270	3.6448040
65	(6, 7, 6, 8, 9, 9, 9, 10)	1.1948907	1.2389100	3.5530634

Table 3 continued

Number	$(m_1, m_2, \ldots, m_8)$	T (approximation)	T (original)	Relative error (%)
66	(6, 7, 7, 7, 8, 9, 9, 11)	1.1766770	1.2162651	3.2548880
67	(6, 7, 7, 7, 8, 9, 10, 10)	1.1677162	1.2049554	3.0905108
68	(6, 7, 7, 7, 8, 10, 9, 10)	1.1847993	1.2267679	3.4210717
69	(6, 7, 7, 7, 9, 9, 9, 10)	1.1804273	1.2210509	3.3269417
70	(6, 7, 7, 8, 8, 9, 9, 10)	1.1730491	1.2116030	3.1820536

which leads to $T = 1.1525194$, as obtained in Table 2.

Furthermore, for each case of Table 3, we also display the T obtained from the original expression, and the relative error of our closed-form approximation of T. It is clear that for No. 18, T obtained from the original expression is 1.1855418, and the relative error of our closed-form approximation of T is 2.7854213 %, which is the smallest among all the 70 cases. □

4 Optimal processor partitioning with power constraint

We extend the optimal multicore server processor partitioning problem by allowing each server S_i to have its own speed s_i. The average task response time of S_i is

$$T_i = \frac{\bar{r}_i}{s_i}\left(1 + \frac{1}{m_i(1-\rho_i)(\sqrt{2\pi m_i}(1-\rho_i)(e^{\rho_i}/e\rho_i)^{m_i}+1)}\right),$$

where

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

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

Our *optimal multicore server processor partitioning with power constraint* problem can be formally defined as follows. Given task arrival rates $\lambda_1, \lambda_2, \ldots, \lambda_n$, mean task execution requirements $\bar{r}_1, \bar{r}_2, \ldots, \bar{r}_n$, the number of available cores m, the base power supply P^*, and the total available power P, we find $m_1, m_2, \ldots, m_n$ and the server speeds $s_1, s_2, \ldots, s_n$, such that T is minimized subject to the constraint that

$$J(m_1, m_2, \ldots, m_n) = m,$$

where

$$J(m_1, m_2, \ldots, m_n) = m_1 + m_2 + \cdots + m_n,$$

and the constraint that

$$K(s_1, s_2, \ldots, s_n) = P,$$

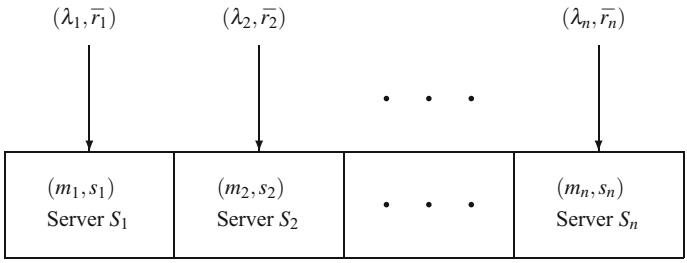


Fig. 2 Processor partitioning with power constraint

where

$$K(s_1, s_2, \ldots, s_n) = \sum_{i=1}^{n} m_i(\rho_i s_i^{\alpha} + P^*) = \sum_{i=1}^{n} \lambda_i \bar{r}_i s_i^{\alpha-1} + mP^*,$$

for the idle-speed model, and the constraint that

$$K(m_1, m_2, \ldots, m_n, s_1, s_2, \ldots, s_n) = P,$$

where

$$K(m_1, m_2, \ldots, m_n, s_1, s_2, \ldots, s_n) = \sum_{i=1}^{n} m_i(s_i^{\alpha} + P^*) = \sum_{i=1}^{n} m_i s_i^{\alpha} + mP^*,$$

for the constant-speed model. An illustration of the problem is given in Fig. 2.

We can minimize T using the method of Lagrange multiplier. For the idle-speed model, we have

$$\begin{aligned}\nabla T(m_1, m_2, \ldots, m_n, s_1, s_2, \ldots, s_n) = {} & \phi \nabla J(m_1, m_2, \ldots, m_n) \\ & + \psi \nabla K(s_1, s_2, \ldots, s_n),\end{aligned}$$

that is,

$$\frac{\partial T}{\partial m_i} = \phi \frac{\partial J}{\partial m_i} = \phi,$$

for all $1 \le i \le n$, where ϕ is a Lagrange multiplier, and

$$\frac{\partial T}{\partial s_i} = \psi \frac{\partial K}{\partial s_i} = \psi \lambda_i \bar{r}_i (\alpha - 1) s_i^{\alpha-2},$$

for all $1 \le i \le n$, where ψ is another Lagrange multiplier. For the constant-speed model, we have

$$\nabla T(m_1, m_2, \ldots, m_n, s_1, s_2, \ldots, s_n) = \phi \nabla J(m_1, m_2, \ldots, m_n) + \psi \nabla K(m_1, m_2, \ldots, m_n, s_1, s_2, \ldots, s_n),$$

that is,

$$\frac{\partial T}{\partial m_i} = \phi \frac{\partial J}{\partial m_i} + \psi \frac{\partial K}{\partial m_i} = \phi + \psi s_i^{\alpha},$$

for all $1 \le i \le n$, where ϕ is a Lagrange multiplier, and

$$\frac{\partial T}{\partial s_i} = \psi \frac{\partial K}{\partial s_i} = \psi m_i \alpha s_i^{\alpha - 1},$$

for all $1 \le i \le n$, where ψ is another Lagrange multiplier.

Using the same calculation in the last section, we get

$$\frac{\partial T}{\partial m_i} = -\frac{\lambda_i \bar{r}_i}{\lambda s_i} F_i^2 \left(\sqrt{2\pi m_i}(1 - \rho_i) \left(\frac{\rho_i + 3}{2} - m_i (\ln \rho_i)(1 - \rho_i) \right) G_i + 1 \right),$$

where

$$G_i = \left(\frac{e^{\rho_i}}{e \rho_i} \right)^{m_i},$$

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

To calculate $\partial T / \partial s_i$, we rewrite T_i as

$$T_i = \frac{\bar{r}_i}{s_i}(1 + F_i),$$

where

$$F_i = \frac{1}{m_i(1 - \rho_i)(\sqrt{2\pi m_i}(1 - \rho_i)(e^{\rho_i}/e\rho_i)^{m_i} + 1)}.$$

It is clear that

$$\frac{\partial T}{\partial s_i} = \frac{\lambda_i}{\lambda} \cdot \frac{\partial T_i}{\partial s_i} = \frac{\lambda_i \bar{r}_i}{\lambda} \left(-\frac{1 + F_i}{s_i^2} + \frac{1}{s_i} \cdot \frac{\partial F_i}{\partial s_i} \right) = \frac{\lambda_i \bar{r}_i}{\lambda s_i} \left(-\frac{1 + F_i}{s_i} + \frac{\partial F_i}{\partial s_i} \right),$$

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

Again, we rewrite F_i as

$$F_i = \frac{1}{m_i(1 - \rho_i)(\sqrt{2\pi m_i}(1 - \rho_i) G_i + 1)} = \frac{1}{\sqrt{2\pi} m_i^{3/2}(1 - \rho_i)^2 G_i + m_i(1 - \rho_i)}.$$

Notice that

$$\ln G_i = m_i \ln(e^{\rho_i}/e\rho_i) = m_i(\rho_i - \ln \rho_i - 1).$$

Since

$$\frac{\partial \rho_i}{\partial s_i} = -\frac{\lambda_i \bar{r}_i}{m_i s_s^2} = -\frac{\rho_i}{s_i},$$

we get

$$\frac{1}{G_i}\frac{\partial G_i}{\partial s_i} = m_i\left(1 - \frac{1}{\rho_i}\right)\frac{\partial \rho_i}{\partial s_i} = \frac{m_i(1-\rho_i)}{s_i},$$

and

$$\frac{\partial G_i}{\partial s_i} = \left(\frac{m_i(1-\rho_i)}{s_i}\right) G_i.$$

Now, we have

$$\begin{aligned}
\frac{\partial F_i}{\partial s_i} &= -F_i^2\left(\sqrt{2\pi}m_i^{3/2}\left(-2(1-\rho_i)\frac{\partial \rho_i}{\partial s_i}G_i + (1-\rho_i)^2\frac{\partial G_i}{\partial s_i}\right) - m_i\frac{\partial \rho_i}{\partial s_i}\right) \\
&= -F_i^2\left(\sqrt{2\pi}m_i^{3/2}\left(\frac{2\rho_i(1-\rho_i)}{s_i} + \frac{m_i(1-\rho_i)^3}{s_i}\right)G_i + \frac{m_i\rho_i}{s_i}\right) \\
&= -\frac{F_i^2 m_i}{s_i}\left(\sqrt{2\pi m_i}(1-\rho_i)\left(2\rho_i + m_i(1-\rho_i)^2\right)G_i + \rho_i\right),
\end{aligned}$$

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

As for $\partial T/\partial s_i$, we have

$$\frac{\partial T}{\partial s_i} = -\frac{\lambda_i \bar{r}_i}{\lambda s_i^2}(1 + F_i + F_i^2 m_i(\sqrt{2\pi m_i}(1-\rho_i)(2\rho_i + m_i(1-\rho_i)^2)G_i + \rho_i)),$$

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

4.1 The idle-speed model

For the idle-speed model, we have

$$\frac{\partial T}{\partial m_i} = -\frac{m_i \rho_i}{\lambda}F_i^2(\sqrt{2\pi m_i}(1-\rho_i)((\rho_i+3)/2 - m_i(\ln \rho_i)(1-\rho_i))G_i + 1) = \phi, \tag{1}$$

for all $1 \le i \le n$. Also, we have

$$\frac{\partial T}{\partial s_i} = -\frac{\lambda_i \bar{r}_i}{\lambda s_i^2}(1 + F_i + F_i^2 m_i(\sqrt{2\pi m_i}(1-\rho_i)(2\rho_i + m_i(1-\rho_i)^2)G_i + \rho_i)) = \psi \lambda_i \bar{r}_i (\alpha - 1) s_i^{\alpha-2},$$

that is,

$$s_i = \left(-\frac{1}{\psi\lambda(\alpha-1)}(1 + F_i + F_i^2 m_i(\sqrt{2\pi m_i}(1-\rho_i)(2\rho_i + m_i(1-\rho_i)^2)G_i + \rho_i))\right)^{1/\alpha}, \tag{2}$$

for all $1 \le i \le n$. Hence, we get a nonlinear system of $(2n+2)$ equations specified in (1) and (2) and the two constraints $J(m_1, m_2, \ldots, m_n) = m$ and $K(s_1, s_2, \ldots, s_n) = P$.

4.2 The constant-speed model

For the constant-speed model, we have

$$\frac{\partial T}{\partial m_i} = -\frac{m_i \rho_i}{\lambda} F_i^2 (\sqrt{2\pi m_i}(1-\rho_i)\,((\rho_i + 3)/2 - m_i(\ln \rho_i)(1-\rho_i))\,G_i + 1) = \phi + \psi s_i^{\alpha}, \tag{3}$$

for all $1 \le i \le n$. Also, we have

$$\frac{\partial T}{\partial s_i} = -\frac{\lambda_i \bar{r}_i}{\lambda s_i^2}(1 + F_i + F_i^2 m_i(\sqrt{2\pi m_i}(1-\rho_i)(2\rho_i + m_i(1-\rho_i)^2)G_i + \rho_i)) = \psi m_i \alpha s_i^{\alpha-1},$$

that is,

$$s_i = \left(-\frac{\rho_i}{\psi\lambda\alpha}(1 + F_i + F_i^2 m_i(\sqrt{2\pi m_i}(1-\rho_i)(2\rho_i + m_i(1-\rho_i)^2)G_i + \rho_i))\right)^{1/\alpha}, \tag{4}$$

for all $1 \le i \le n$. Hence, we get a nonlinear system of $(2n+2)$ equations specified in (3) and (4) and the two constraints $J(m_1, m_2, \ldots, m_n) = m$ and $K(m_1, m_2, \ldots, m_n, s_1, s_2, \ldots, s_n) = P$.

4.3 A numerical procedure

The above nonlinear systems of equations are extremely sophisticated to solve. A special numerical procedure consisting of four subalgorithms A_1, A_2, A_2, and A_4 has been developed to solve the equations.

- (A_1) First, we notice that the right-hand side of (2) or (4) is a decreasing function of s_i. Thus, given m_i, λ_i, $\bar{r}_i$, λ, and ψ, we can find s_i using the bisection method in an appropriately chosen interval $[lb_1, ub_1]$, for all $1 \le i \le n$.

- (A_2) Second, we notice that the left-hand side of (1) (i.e., $\partial T/\partial m_i$) or $\partial T/\partial m_i - \psi s_i^{\alpha}$ from (3) is an increasing function of m_i. Thus, given λ_i, $\bar{r}_i$, λ, ϕ, and ψ, we can find m_i using the bisection method in an appropriately chosen interval $[lb_2, ub_2]$, for all $1 \leq i \leq n$.
- (A_3) Third, we notice that for a given ϕ, $J(m_1, m_2, \ldots, m_n)$ is a decreasing function of ψ. Thus, given the λ_is, $\bar{r}_i$s, m, and ϕ, we can find ψ using the bisection method in an appropriately chosen interval $[lb_3, ub_3]$.
- (A_4) Finally, we notice that $K(s_1, s_2, \ldots, s_n)$ or $K(m_1, m_2, \ldots, m_n, s_1, s_2, \ldots, s_n)$ is an increasing function of ϕ. Thus, given the λ_is, $\bar{r}_i$s, m, P^*, and P, we can find ϕ using the bisection method in an appropriately chosen interval $[lb_4, ub_4]$.

In the above algorithm, all the intervals for the bisection method should be carefully determined and they are very sensitive to the input data. The subalgorithm A_j calls A_{j-1} for all $2 \leq j \leq 4$.

Example 2 Let us consider $m = 42$ cores to be partitioned into $n = 7$ subsystems. The task arrival rates are $\lambda_i = ((i + 5)/63)\lambda$, for all $1 \leq i \leq n$, where $\lambda = 35$. The mean task execution requirements are $\bar{r}_i = 1$ for all $1 \leq i \leq n$. The base power is $P^* = 2$ and the total power is $P = 160$. In Table 4, for both idle-speed model and constant-speed model, we show λ_i, m_i, ρ_i, and T_i, for all $1 \leq i \leq n$, as well as the minimized average task response time T. The intervals are set as $[lb_1, ub_1] = [0.5, 10.0]$, $[lb_2, ub_2] = [0.55, 15.0]$ for the idle-speed model and $[lb_2, ub_2] = [0.65, 13.0]$ for the constant-speed model, $[lb_3, ub_3] = [-0.008, -0.001]$, and $[lb_4, ub_4] = [-0.007, -0.005]$. All the data are calculated with the length of a search interval reduced to no longer than 10^{-14}. It is observed that the two speed models do yield different results. For

Table 4 Numerical data for Example 2

i	λ_i	m_i	s_i	ρ_i	T_i
Idle-speed model, $T = 0.7148433$					
1	3.3333333	4.4055420	1.4824260	0.5103950	0.7207803
2	3.8888889	4.9528719	1.4789350	0.5309081	0.7183528
3	4.4444444	5.4889139	1.4761277	0.5485385	0.7164583
4	5.0000000	6.0157562	1.4738066	0.5639483	0.7149322
5	5.5555556	6.5349134	1.4718461	0.5775974	0.7136725
6	6.1111111	7.0475257	1.4701614	0.5898186	0.7126121
7	6.6666667	7.5544769	1.4686936	0.6008598	0.7117052
Constant-speed model, $T = 0.9306993$					
1	3.3333333	4.4096443	1.1907179	0.6348429	0.9753489
2	3.8888889	4.9538643	1.2014184	0.6534121	0.9579657
3	4.4444444	5.4880227	1.2103478	0.6691006	0.9438416
4	5.0000000	6.0139980	1.2179592	0.6826121	0.9320632
5	5.5555556	6.5331488	1.2245561	0.6944264	0.9220409
6	6.1111111	7.0464954	1.2303514	0.7048843	0.9133744
7	6.6666667	7.5548266	1.2354991	0.7142361	0.9057812

instance, the server speed s_i decreases with i for the idle-speed model while increases with i for the constant-speed model. Notice that all the T_is and T are calculated based on the m_is which are not integers. In reality, the m_is should be rounded to the nearest integers. Based on the new m_is, the core speeds $s_1, s_2, \ldots, s_n$ and the T_is and T need to be re-calculated. This is a new problem called optimal power allocation, which we address in the next section. □

5 Optimal power allocation

Given task arrival rates $\lambda_1, \lambda_2, \ldots, \lambda_n$, mean task execution requirements $\bar{r}_1, \bar{r}_2, \ldots, \bar{r}_n$, the server sizes $m_1, m_2, \ldots, m_n$, the base power supply P^*, and the total available power P, the *optimal power allocation* problem is to find the server speeds $s_1, s_2, \ldots, s_n$, such that T is minimized subject to the constraint that

$$K(s_1, s_2, \ldots, s_n) = P,$$

where

$$K(s_1, s_2, \ldots, s_n) = \sum_{i=1}^{n} m_i(\rho_i s_i^{\alpha} + P^*) = \sum_{i=1}^{n} \lambda_i \bar{r}_i s_i^{\alpha-1} + mP^*,$$

for the idle-speed model, and

$$K(s_1, s_2, \ldots, s_n) = \sum_{i=1}^{n} m_i(s_i^{\alpha} + P^*) = \sum_{i=1}^{n} m_i s_i^{\alpha} + mP^*,$$

for the constant-speed model.

We can minimize T using the method of Lagrange multiplier, namely,

$$\nabla T(s_1, s_2, \ldots, s_n) = \psi \nabla K(s_1, s_2, \ldots, s_n),$$

that is,

$$\frac{\partial T}{\partial s_i} = \psi \frac{\partial K}{\partial s_i} = \psi \lambda_i \bar{r}_i (\alpha - 1) s_i^{\alpha-2},$$

for all $1 \le i \le n$ and the idle-speed model, and

$$\frac{\partial T}{\partial s_i} = \psi \frac{\partial K}{\partial s_i} = \psi m_i \alpha s_i^{\alpha-1},$$

for all $1 \le i \le n$ and the constant-speed model, where ψ is a Lagrange multiplier. Following the same derivations of the last section, we reach (2) and (4). Hence, we can use A_1 to find the s_is and a method similar to A_4 to find ψ by noticing that $K(m_1, m_2, \ldots, m_n)$ is an increasing function of ψ.

Table 5 Numerical data for Example 3

i	λ_i	m_i	s_i	ρ_i	T_i
Idle-speed model, $T = 0.7169628$					
1	3.3333333	4.0000000	1.5354778	0.5427192	0.7191793
2	3.8888889	5.0000000	1.4720125	0.5283772	0.7200782
3	4.4444444	5.0000000	1.5294003	0.5812009	0.7121010
4	5.0000000	6.0000000	1.4732169	0.5656556	0.7159418
5	5.5555556	7.0000000	1.4358554	0.5527373	0.7205760
6	6.1111111	7.0000000	1.4719721	0.5930927	0.7131587
7	6.6666667	8.0000000	1.4372932	0.5797936	0.7185203
Constant-speed model, $T = 0.9337903$					
1	3.3333333	4.0000000	1.2662899	0.6580905	0.9632283
2	3.8888889	5.0000000	1.1934519	0.6517043	0.9609477
3	4.4444444	5.0000000	1.2858668	0.6912760	0.9284561
4	5.0000000	6.0000000	1.2188160	0.6837237	0.9327767
5	5.5555556	7.0000000	1.1711067	0.6776930	0.9397011
6	6.1111111	7.0000000	1.2347930	0.7070139	0.9129517
7	6.6666667	8.0000000	1.1894521	0.7006027	0.9217223

Example 3 Let us consider $n = 7$ servers with sizes

$$(m_1, m_2, m_3, m_4, m_5, m_6, m_7) = (4, 5, 5, 6, 7, 7, 8),$$

which are obtained by rounding the m_is in Example 2 to the nearest integers. All other parameters are the same as those in Example 2. In Table 5, for both idle-speed model and constant-speed model, we show λ_i, m_i, ρ_i, and T_i, for all $1 \le i \le n$, as well as the minimized average task response time T. All the data are calculated with the length of a search interval reduced to no longer than 10^{-14}. It is observed that for moderate server utilization, rounding the m_is slightly increases T.

To demonstrate the optimality of our solution, we perform extra computation as follows. For each m_i, its fraction part is truncated, and we obtain

$$(m_1, m_2, m_3, m_4, m_5, m_6, m_7) = (4, 4, 5, 6, 6, 7, 7).$$

The three additional cores are allocated to the seven servers in the following way, i.e., we choose three servers out of the seven and allocate one extra core to each server. It is clear that there are $\binom{7}{3} = 35$ different ways. For all these 35 different partitions of the m cores, we calculate the average task response time and show the results in Tables 6 and 7. It can be seen that for the idle-speed model, the optimal partition is indeed No. 9 (in boldface):

$$(m_1, m_2, m_3, m_4, m_5, m_6, m_7) = (4, 5, 5, 6, 7, 7, 8),$$

Table 6 Optimality of Examples 2 and 3 (idle-speed model)

Number	$(m_1, m_2, \ldots, m_7)$	T (approximation)	T (original)	Relative error (%)
1	(4, 4, 5, 6, 7, 8, 8)	0.7220880	0.7276647	0.7663863
2	(4, 4, 5, 7, 6, 8, 8)	0.7243295	0.7304033	0.8315701
3	(4, 4, 5, 7, 7, 7, 8)	0.7223547	0.7279937	0.7745883
4	(4, 4, 5, 7, 7, 8, 7)	0.7243922	0.7304778	0.8331019
5	(4, 4, 6, 6, 6, 8, 8)	0.7222914	0.7279083	0.7716543
6	(4, 4, 6, 6, 7, 7, 8)	0.7203330	0.7255228	0.7153242
7	(4, 4, 6, 6, 7, 8, 7)	0.7223490	0.7279757	0.7729184
8	(4, 5, 5, 6, 6, 8, 8)	0.7188959	0.7237573	0.6716867
9	**(4, 5, 5, 6, 7, 7, 8)**	**0.7169628**	**0.7214077**	**0.6161394**
10	(4, 5, 5, 6, 7, 8, 7)	0.7189458	0.7238141	0.6725999
11	(5, 4, 5, 6, 6, 8, 8)	0.7227611	0.7284785	0.7848352
12	(5, 4, 5, 6, 7, 7, 8)	0.7207927	0.7260786	0.7280127
13	(5, 4, 5, 6, 7, 8, 7)	0.7228219	0.7285501	0.7862497
14	(4, 4, 6, 7, 6, 7, 8)	0.7225609	0.7282411	0.7799965
15	(4, 4, 6, 7, 6, 8, 7)	0.7246079	0.7307386	0.8389737
16	(4, 4, 6, 7, 7, 7, 7)	0.7226206	0.7283113	0.7813623
17	(4, 5, 5, 7, 6, 7, 8)	0.7191556	0.7240763	0.6795935
18	(4, 5, 5, 7, 6, 8, 7)	0.7211688	0.7265265	0.7374378
19	(4, 5, 5, 7, 7, 7, 7)	0.7192074	0.7241359	0.6806047
20	(4, 5, 6, 6, 6, 7, 8)	0.7171516	0.7216310	0.6207322
21	(4, 5, 6, 6, 6, 8, 7)	0.7191436	0.7240501	0.6776433
22	(4, 5, 6, 6, 7, 7, 7)	0.7171985	0.7216836	0.6214789
23	(5, 4, 5, 7, 6, 7, 8)	0.7230346	0.7288169	0.7933834
24	(5, 4, 5, 7, 6, 8, 7)	0.7250951	0.7313335	0.8530176
25	(5, 4, 5, 7, 7, 7, 7)	0.7230974	0.7288914	0.7949023
26	(5, 4, 6, 6, 6, 7, 8)	0.7209962	0.7263221	0.7332720
27	(5, 4, 6, 6, 6, 8, 7)	0.7230348	0.7288068	0.7919740
28	(5, 4, 6, 6, 7, 7, 7)	0.7210539	0.7263893	0.7345191
29	(5, 5, 5, 6, 6, 7, 8)	0.7176004	0.7221717	0.6329918
30	(5, 5, 5, 6, 6, 8, 7)	0.7196054	0.7246091	0.6905454
31	(5, 5, 5, 6, 7, 7, 7)	0.7176503	0.7222284	0.6338827
32	(4, 5, 6, 7, 6, 7, 7)	0.7194079	0.7243756	0.6857878
33	(5, 4, 6, 7, 6, 7, 7)	0.7233133	0.7291521	0.8007744
34	(5, 5, 5, 7, 6, 7, 7)	0.7198736	0.7249402	0.6988957
35	(5, 5, 6, 6, 6, 7, 7)	0.7178482	0.7224641	0.6389132

which leads to $T = 0.7169628$, as obtained in Table 5. For the constant-speed model, the optimal partition is indeed No. 9 (in boldface):

$$(m_1, m_2, m_3, m_4, m_5, m_6, m_7) = (4, 5, 5, 6, 7, 7, 8),$$

Table 7 Optimality of Examples 2 and 3 (constant-speed model)

Number	$(m_1, m_2, \ldots, m_7)$	T (approximation)	T (original)	Relative error (%)
1	(4, 4, 5, 6, 7, 8, 8)	0.9411695	0.9650705	2.4766035
2	(4, 4, 5, 7, 6, 8, 8)	0.9444952	0.9693359	2.5626486
3	(4, 4, 5, 7, 7, 7, 8)	0.9415750	0.9656583	2.4939775
4	(4, 4, 5, 7, 7, 8, 7)	0.9446116	0.9695176	2.5689079
5	(4, 4, 6, 6, 6, 8, 8)	0.9414687	0.9654968	2.4886806
6	(4, 4, 6, 6, 7, 7, 8)	0.9385720	0.9618472	2.4198514
7	(4, 4, 6, 6, 7, 8, 7)	0.9415758	0.9656673	2.4948000
8	(4, 5, 5, 6, 6, 8, 8)	0.9366506	0.9595204	2.3834575
9	**(4, 5, 5, 6, 7, 7, 8)**	**0.9337903**	**0.9559134**	**2.3143358**
10	(4, 5, 5, 6, 7, 8, 7)	0.9367436	0.9596739	2.3893882
11	(5, 4, 5, 6, 6, 8, 8)	0.9421207	0.9664011	2.5124582
12	(5, 4, 5, 6, 7, 7, 8)	0.9392065	0.9627301	2.4434230
13	(5, 4, 5, 6, 7, 8, 7)	0.9422347	0.9665802	2.5187261
14	(4, 4, 6, 7, 6, 7, 8)	0.9418804	0.9660926	2.5061985
15	(4, 4, 6, 7, 6, 8, 7)	0.9449347	0.9699735	2.5813905
16	(4, 4, 6, 7, 7, 7, 7)	0.9419922	0.9662690	2.5124262
17	(4, 5, 5, 7, 6, 7, 8)	0.9370440	0.9600936	2.4007700
18	(4, 5, 5, 7, 6, 8, 7)	0.9400466	0.9639140	2.4760937
19	(4, 5, 5, 7, 7, 7, 7)	0.9371415	0.9602530	2.4068102
20	(4, 5, 6, 6, 6, 7, 8)	0.9340647	0.9563099	2.3261573
21	(4, 5, 6, 6, 6, 8, 7)	0.9370349	0.9600913	2.4014855
22	(4, 5, 6, 6, 7, 7, 7)	0.9341533	0.9564583	2.3320490
23	(5, 4, 5, 7, 6, 7, 8)	0.9425412	0.9670083	2.5301791
24	(5, 4, 5, 7, 6, 8, 7)	0.9456204	0.9709198	2.6057155
25	(5, 4, 5, 7, 7, 7, 7)	0.9426600	0.9671934	2.5365557
26	(5, 4, 6, 6, 6, 7, 8)	0.9395088	0.9631608	2.4556726
27	(5, 4, 6, 6, 6, 8, 7)	0.9425546	0.9670325	2.5312420
28	(5, 4, 6, 6, 7, 7, 7)	0.9396183	0.9633347	2.4619076
29	(5, 5, 5, 6, 6, 7, 8)	0.9346817	0.9571719	2.3496528
30	(5, 5, 5, 6, 6, 8, 7)	0.9376759	0.9609830	2.4253458
31	(5, 5, 5, 6, 7, 7, 7)	0.9347769	0.9573287	2.3556964
32	(4, 5, 6, 7, 6, 7, 7)	0.9374390	0.9606784	2.4190540
33	(5, 4, 6, 7, 6, 7, 7)	0.9429865	0.9676541	2.5492168
34	(5, 5, 5, 7, 6, 7, 7)	0.9380888	0.9615814	2.4431209
35	(5, 5, 6, 6, 6, 7, 7)	0.9350712	0.9577504	2.3679671

which leads to $T = 0.9337903$, as obtained in Table 5.

Furthermore, for each case of Table 6, we also display the T obtained from the original expression, and the relative error of our closed-form approximation of T. It is clear that for No. 9, T obtained from the original expression is 0.7214077, and the

relative error of our closed-form approximation of T is 0.6161394 %, which is the smallest among all the 35 cases.

Similarly, for each case of Table 7, we also display the T obtained from the original expression, and the relative error of our closed-form approximation of T. It is clear that for No. 9, T obtained from the original expression is 0.9559134, and the relative error of our closed-form approximation of T is 2.3143358 %, which is the smallest among all the 35 cases. □

6 Conclusions

We have formulated and solved three optimization problems related to optimal system (virtual server) configuration for some given types of applications in a cloud computing environment, namely, optimal multicore server processor partitioning, optimal multicore server processor partitioning with power constraint, and optimal power allocation. Such optimal multicore server processor partitioning has important applications in dynamic resource provision in a cloud computing environment for certain specific types of applications, such that the overall system performance is optimized without exceeding certain energy consumption budget. We provided numerical procedures to solve the above complicated problems and demonstrated numerical data.

Our investigation in this paper implies that dynamic resource provision, system performance optimization, and energy consumption reduction should be considered in an integrated and analytical way.

Notice that in this paper, we have assumed that the size (i.e., the number of cores) of a server can be any positive integer. Also, the speed of a server can be any positive real number. In a real multicore server processor, the size of a server might only be some pre-determined values. Furthermore, the speed of a server might only be some pre-determined values, and becomes a discrete variable. In such circumstances, the extension and optimality of our method in this paper need further investigation and examination.

Finally, it would be interesting and important to test our method in a real data center to find an optimal system partitioning for a real environment with given types of applications. This will be our future effort and investigation.

Acknowledgments The author would like to express his gratitude to five anonymous reviewers for their suggestions to improve the manuscript. A preliminary version of the paper [15] was presented on Workshop on Parallel Computing and Optimization in conjunction with the 26th IEEE International Parallel and Distributed Processing Symposium, Shanghai, China, May 21–25, 2012.

Compliance with ethical standards This research does not have any source of funding and potential conflict of interest (financial or non-financial), and does not involve any human participant and animal.

References

1. http://en.wikipedia.org/wiki/Adapteva. Accessed 24 June 2015
2. http://en.wikipedia.org/wiki/CMOS. Accessed 24 June 2015
3. http://multicore.amd.com/us-en/AMD-multicore/multicore-Advantages.aspx. Accessed 24 June 2015
4. http://www.computerworld.com/s/article/41632/Server_Partitioning. Accessed 24 June 2015

5. http://www.intel.com/multicore/. Accessed 24 June 2015
6. http://www.multicoreinfo.com/2011/10/adapteva-2/. Accessed 24 June 2015
7. Cao J, Hwang K, Li K, Zomaya A (2013) Optimal multiserver configuration for profit maximization in cloud computing. IEEE Trans Parallel Distrib Syst 24(6):1087–1096
8. Cao J, Li K, Stojmenovic I (2014) Optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers. IEEE Trans Comput 63(1):45–58
9. Chaisiri S, Kaewpuang R, Lee B-S, Niyato D (2011) Cost minimization for provisioning virtual servers in Amazon elastic compute cloud. In: Proceedings of 19th IEEE international symposium on modelling, analysis, and simulation of computer and telecommunication systems, pp 85–95
10. Chandrakasan AP, Sheng S, Brodersen RW (1992) Low-power CMOS digital design. IEEE J. Solid-State Circuits 27(4):473–484
11. Chen M-S, Shin KG (1987) Processor allocation in an N-cube multiprocessor using gray codes. IEEE Trans Comput 36(12):1396–1407
12. Dejun J, Pierre G, Chi C-H (2010) Autonomous resource provisioning for multi-service web applications. In: Proceedings of the 19th international world-wide web conference
13. Kleinrock L (1975) Queueing systems, theory, vol 1. Wiley, New York
14. Li K (2012) Optimal configuration of a multicore server processor for managing the power and performance tradeoff. J Supercomput 61(1):189–214
15. Li K (2012) Optimal partitioning of a multicore server processor. In: Proceedings of the 26th IEEE international parallel and distributed processing symposium workshops (workshop on parallel computing and optimization), Shanghai, China, May 21–25, pp 1797–1805
16. Li K, Cheng KH (1989) Complexity of resource allocation and job scheduling problems in partitionable mesh connected systems. In: Proceedings of the 1st IEEE symposium on parallel and distributed processing, pp 358–365
17. Marty MR, Hill MD (2007) Virtual hierarchies to support server consolidation. In: Proceedings of the 34th international symposium on computer architecture, pp 46–56
18. Siegel HJ (1985) Interconnection networks for large-scale parallel processing. D. C. Heath, Massachusetts
19. Sodan AC, Machina J, Deshmeh A, Macnaughton K, Esbaugh B (2010) Parallelism via multithreaded and multicore CPUs. IEEE Comput 43(3):24–32
20. Urgaonkar B, Shenoy P, Chandra A, Goyal P, Wood T (2008) Agile dynamic provisioning of multi-tier Internet applications. ACM Trans Autonom Adapt Syst 3(1)
21. Villela D, Pradhan P, Rubenstein D (2007) Provisioning servers in the application tier for e-commerce systems. ACM Trans Internet Technol 7(1)
22. Zhai B, Blaauw D, Sylvester D, Flautner K (2004) Theoretical and practical limits of dynamic voltage scaling. In: Proceedings of the 41st design automation conference, pp 868–873