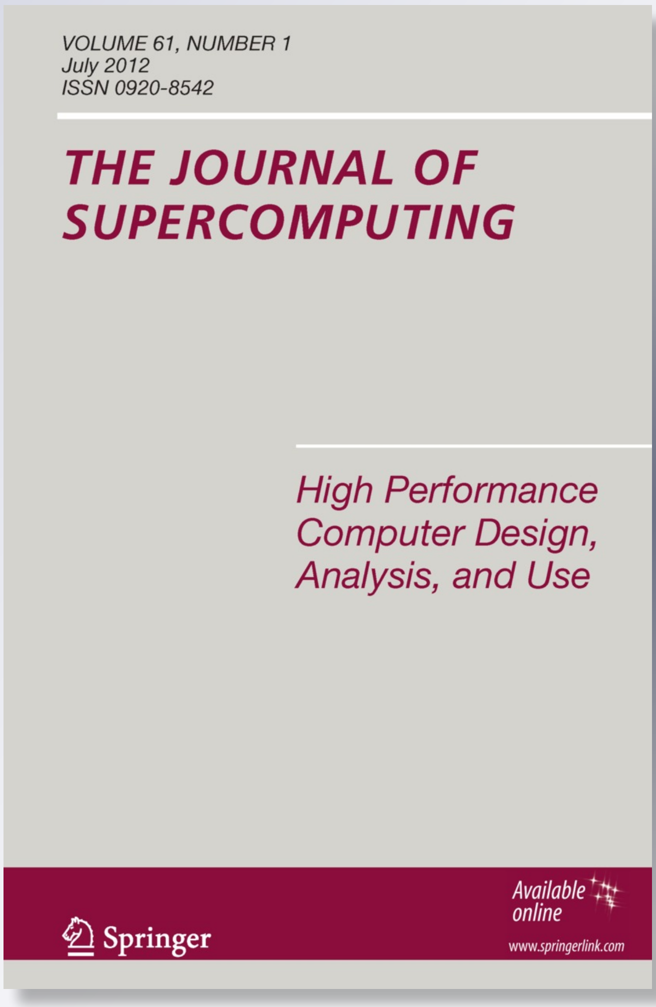
Optimal configuration of a multicore server processor for managing the power and performance tradeoff

Keqin Li

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

ISSN 0920-8542
Volume 61
Number 1

J Supercomput (2012) 61:189-214
DOI 10.1007/s11227-011-0686-1



J Supercomput (2012) 61:189–214
DOI 10.1007/s11227-011-0686-1

Optimal configuration of a multicore server processor for managing the power and performance tradeoff

Keqin Li

Published online: 28 September 2011

Abstract We consider the problem of power and performance management for a multicore server processor in a cloud computing environment by optimal server configuration for a specific application environment. The motivation of the study is that such optimal virtual server configuration is important for dynamic resource provision in a cloud computing environment to optimize the power and performance tradeoff for certain specific type of applications. Our strategy is to treat a multicore server processor as an M/M/m queueing system with multiple servers. 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. Our investigation includes justification of centralized management of computing resources, server speed constrained optimization, power constrained performance optimization, and performance constrained power optimization. Our main results are (1) cores should be managed in a centralized way to provide the highest performance without consumption of more energy in cloud computing; (2) for a given server speed constraint, fewer high-speed cores perform better than more low-speed cores; furthermore, there is an optimal selection of server size and core speed which can be obtained analytically, such that a multicore server processor consumes the minimum power; (3) for a given power consumption constraint, there is an optimal selection of server size and core speed which can be obtained numerically, such that the best performance can be achieved, i.e., the average task response time is minimized; (4) for a given task response time constraint, there is an optimal selection of server size and core speed which can be obtained numerically, such that minimum power consumption can be achieved while the given performance guarantee is maintained.

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

Keywords Energy consumption · Multicore server processor · Power and performance tradeoff · Queueing model · Response time · Server configuration

1 Introduction

As the latest multimedia and networking based applications in the ever increasing cloud computing field provide new features and cutting-edge capabilities, processor development needs to stay ahead of increased demands from software applications. It has been realized that increasing processor speed is not the only concern and solution. Modern large-scale computer systems need to run faster and cooler, occupy less space, and consume less energy. The multicore processor technology helps dealing with these challenges simultaneously. With the power of dozens or even hundreds processor cores on a single chip, multicore processors deliver leading performance and unique features that help server systems in a data center run cooler and more efficient. The evolution of multicore design has allowed for increased performance and higher productivity to meet the needs of next-generation applications. Since multicore processors offer true multitasking capabilities, users can simultaneously run multiple complex applications and successfully complete more tasks in a shorter amount of time. Because they put more processing power into a smaller package, multicore processors help build data center server infrastructures in a cloud computing environment with a smaller footprint, reduced cooling needs, and increased energy efficiency [11].

In this paper, we consider power and performance management for a multicore server processor in a cloud computing environment by optimal server configuration (i.e., optimal server size and core speed determination) for a specific application environment specified by the task arrival rate and the average task execution requirement. Such optimal virtual server configuration is important for dynamic resource provision in a cloud computing environment to optimize the power and performance tradeoff for certain specific type of applications. A multicore server processor is treated as an M/M/m queueing system with multiple servers (i.e., an m-server queueing system with an exponential inter-arrival time distribution and an exponential service time distribution). The system performance measures are the average task response time and the average power consumption. We show the following results.

- *Optimization via Core Distribution* (*Justification of Centralized Management*)—A group of n m-core server processors yield shorter average task response time than a group of n' m'-core server processors do, where all the cores have the same speed and $nm = n'm' = M$ and $n < n'$. Both groups consume the same amount of power. In other words, fewer multicore server processors of larger sizes perform better than more multicore server processors of smaller sizes. Consequently, an mn-core server processor yields shorter average task response time than n m-core server processors do, for all $n > 1$, and the mn-core server processor consumes the same amount of power as n m-core server processors do. This result implies that cores should be managed in a centralized way to provide the highest performance without consumption of more energy in cloud computing.
- *Server Speed Constrained Optimization*—An m-core server processor with core speed s yields shorter average task response time than an m'-core server processor with core speed s' does, where $m < m'$ and $s > s'$ and $ms = m's' = c$ and c is the

server speed, i.e., the product of server size and core speed, or, the combined core speed. In other words, given a fixed server speed c of a multicore server processor, fewer high-speed cores perform better than more low-speed cores. However, when m is too small, the high-speed cores consume significant amount of energy. On the other hand, when m is too large, the overhead in power consumption also causes increased energy waste. Hence, for a fixed server speed c, there is an optimal selection of server size m and core speed s, such that an m-core server processor with core speed s consumes the minimum power.

- *Power Constrained Performance Optimization*—Given a fixed power supply P, there is an optimal selection of server size m and core speed s, such that an m-core server processor with core speed s has the minimum average task response time and that it consumes power P. When m is too small, the increment in core speed s is not enough to handle a given workload. On the other hand, when m is too large, the overhead in power consumption eats up the performance. Hence, there is an optimal selection of server size and core speed, such that the best performance can be achieved (i.e., the average task response time is minimized) by consuming a given energy resource (i.e., power consumption does not exceed the supply).
- *Performance Constrained Power Optimization*—For a given average task response time T, there is an optimal choice of server size m and core speed s, such that an m-core server processor with core speed s consumes the minimum power and that the task response time is T. The reason is similar to server speed constrained optimization, i.e., when m is too small, the high-speed cores consume significant amount of energy; on the other hand, when m is too large, the overhead in power consumption also causes increased energy waste. Hence, there is an optimal selection of server size and core speed, such that minimum power consumption can be achieved while a given performance guarantee is maintained.

Two core speed and power consumption models are considered, namely, the idle-speed model and the constant-speed model (see Sect. 4 for an explanation). We also provide extensive numerical examples and data to demonstrate our results and observations. To the best of our knowledge, such analytical study of power and performance tradeoff using a multiserver queueing model has not been performed before.

The remainder of the paper is organized as follows. In Sect. 2, we review related research in managing the power and performance tradeoff. In Sect. 3, we describe an M/M/m queueing model for a multicore server processor. In Sect. 4, we present two power consumption and core speed models. In Sect. 5, we prove that a centralized management of cores performs better than a distributed management of cores. In Sect. 6, we consider server speed constrained power and performance optimization. In Sect. 7, we study power constrained performance optimization. In Sect. 8, we investigate performance constrained power optimization. We conclude the paper in Sect. 9.

2 Related research

Energy-efficient computing to deal with the power and performance tradeoff has gained increasing research attention in the last few years. Essentially, there are two methods in coping with the energy-delay and power-performance tradeoffs. Both approaches fix one factor and optimize the other. The first method is power or energy constrained performance optimization, which is applied to high-performance computing systems and servers, where power-aware design techniques and algorithms attempt to maximize performance or minimize delay under certain power consumption or energy budget constraint. The second method is performance constrained power minimization, where low-power and energy-efficient design techniques and algorithms aim to minimize energy consumption while still meeting certain performance goal. Such combined consideration and management of power and performance have compelling economic, environmental, and technical reasons. There has been increasing interest and importance in developing high-performance and energy-efficient computing systems and data centers. Reducing processor energy consumption has been an important and pressing research issue in recent years, and an explosive body of literature has been developed for energy-efficient computing and communication. The reader is referred to [1, 5, 36, 37] for comprehensive surveys.

Among the numerous hardware and software techniques ever developed for reducing energy consumption, dynamic power management at the operating system level is one of the most effective and efficient ways of managing the power-performance tradeoff. Such techniques are based on supply voltage and clock frequency adjustment schemes implemented while tasks are running. These power reduction and performance optimization techniques explore the opportunities for fine and ultra-fine tuning of the energy-delay tradeoff [35]. Such management of power and performance can be carried out at different levels, i.e., task level, system level, server cluster, and data center level.

Task Level Power and Performance Management—Power-aware task scheduling on processors with variable voltages and speeds has been extensively studied since mid-1990s. In a pioneering paper [40], the authors proposed a method of energy reduction using fine grain control of processor speed by an operating system scheduler. The main idea is to monitor processor idle time and to reduce energy consumption by reducing clock speed and idle time to a minimum. In [4], the author studied the problems of minimizing the expected execution time given a hard energy budget and minimizing the expected energy expenditure given a hard execution deadline for a single task with randomized execution requirement. In [9], the authors examined the relationship among parallelization of an application, program performance, and energy consumption, and the problem of minimizing energy-delay product. In [15, 20], the authors attempted joint minimization of energy consumption and task execution time for a parallel program or a metatask in a heterogeneous or grid computing environment.

System Level Power and Performance Management—Performance constrained energy reduction in a computing system with multiple tasks was first studied in [42], where the authors analyzed offline and online algorithms for scheduling tasks with arrival times and deadlines on a uniprocessor computer to achieve minimum energy consumption. The research has been extended by a number of researchers in substantial further investigation [3, 7, 19, 26–28, 43]. Significant research has been focused on real-time applications, namely, adjusting the supply voltage and clock frequency to minimize processor energy consumption while still meeting the deadlines for task execution [2, 10, 13, 18, 21, 29–31, 33, 34, 41, 48–51]. Energy and time constrained power allocation and task scheduling on multiprocessor computers with dynamically variable voltage and frequency and speed and power have also been addressed as combinatorial optimization problems [22–25]. Our scheduling problems are defined such that the energy-delay product is optimized by fixing one factor and minimizing the other. In [6], the author considered scheduling jobs with equal requirements on multiprocessors. In [32], the authors investigated the problem of system value maximization subject to both time and energy constraints.

Server Cluster and Data Center Level Power and Performance Management—Efficient power management and performance optimization in large-scale data centers and server clusters has gained much attention in the research community in recent years. In [16], the authors developed a framework for hierarchical autonomic power and performance management in high-performance distributed data centers. In [39], the authors proposed a highly scalable hierarchical power control architecture for large-scale data centers. In [38], the authors presented a novel cluster-level control architecture that coordinates individual power and performance control loops for virtualized server clusters. In [45–47], the authors formulated an optimization problem to get an optimal resource scheduling strategy for a given parallel workload in a server cluster, such that the proposed optimization model provides controllable and predictable quantitative control of power consumption with theoretically guaranteed service performance, where a server is treated as an M/G/1 queueing system, i.e., a single server system.

Our investigation in this paper belongs to the system and server level. Again, we consider both power constrained performance optimization and performance constrained power minimization. However, our approach in this paper is different from all previous studies at the system and server level. Instead of considering power-aware scheduling of a given set of tasks, we investigate power and performance of a multicore server processor by modeling the server as a dynamic queueing system (i.e., an M/M/m queueing system with multiple servers). The motivation and rationale of such a study is to reveal the nature of a multicore server processor in a cloud computing environment, not in a traditional high-performance computing or a real-time computing environment. Our research introduces a number of unique features. First, the performance measure of a multicore server processor treated as a dynamic queueing system is the average task response time, not the makespan in scheduling a given set of tasks. Second, the energy consumption measure is the average power consumption of a multicore server processor, not the total amount of energy consumed to complete a set of tasks. Third, optimization of performance and power is realized by server configuration, which is different from minimization of energy and delay by heuristic scheduling algorithms.

3 A multicore server processor model

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.

There is a Poisson stream of tasks with arrival rate λ (measured in the number of tasks arrived per second), 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 giga 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 giga instructions that can be executed in one second). Hence, the task execution times on the cores of server S are i.i.d. exponential random variables $x = r/s$ (measured in second) 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 second. 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 ([17], p. 102)

$$p_k = \begin{cases} p_0 \frac{(m\rho)^k}{k!}, & k \le m; \\ p_0 \frac{m^m \rho^k}{m!}, & k \ge m; \end{cases}$$

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

$$\bar{N} = \sum_{k=0}^{\infty} k p_k.$$

By straightforward algebraic manipulation, we get

$$\bar{N} = m\rho + \frac{\rho}{1-\rho} P_q.$$

Applying Little's result, we get the average task response time T as

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

where T is measured in second.

4 Power consumption and core speed models

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 $P = aCV^2 f$ (measured in Watt), where a is an activity factor, C is the loading capacitance, V is the supply voltage, and f is the clock frequency [8]. Since $s \propto f$, where s is the processor speed, and $f \propto V^{\phi}$ with $0 < \phi \le 1$ [44], which implies that $V \propto f^{1/\phi}$, we know that power consumption is $P \propto f^{\alpha}$ and $P \propto s^{\alpha}$, where $\alpha = 1 + 2/\phi \ge 3$. For ease of discussion, we will assume that the power allocated to a processor core with speed s is simply s^{α}. (Note: For all numerical examples in this paper, we set $\alpha = 3$.)

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 (e.g., its cache memory consumes energy), we will include P^* in P, i.e.,

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

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 $P = m(s^{\alpha} + P^*)$.

It is clear that the idle-speed model is more difficult to implement than the constant-speed model. However, recent development in processor technologies has supported the idle-speed model. For instance, available in servers built with Intel Xeon processor 5500 series, Intel intelligent power technology conserves power by delivering advanced power-management capabilities [12]. Providing the highest system-level performance per watt, Intel intelligent power technology helps business gain capacity to grow, increase IT performance, and save energy costs. Integrated power gates in Intel intelligent power technology allow individual idling cores to reduce to near-zero power consumption independent from other operating cores. Such low-power states automatically put processors and memories into the lowest available power states to meet requirements of the current workload, while not impacting performance. The Intel intelligent power node manager utilizes instrumentation available on Intel Xeon processor 5500 series and other system components to report and cap system power. This system level technology provides a foundation for rack, group, data center, and facility level management capabilities to improve overall data center energy efficiency and resource flexibility [14].

5 Optimization via core distribution

In this section, we consider performance and power optimization via distribution of cores among servers. In particular, we show that for the same number of cores and core speed, and the same workload, a centralized management of cores yields better performance (i.e., shorter average task response time) than a distributed management of cores, while the power consumption is the same.

5.1 Performance optimization

Theorem 1 *A group of n m-core server processors yield shorter average task response time than a group of n' m'-core server processors do, where all the cores have the same speed s and $nm = n'm' = M$ and $n < n'$.*

Proof Assume that the task arrival rate is λ. It is easy to see that the average task response time of n m-core servers is minimized when all the n servers handle the same amount of workload, i.e., the task arrival rate to each server is λ/n. The utilization of an m-core server is

$$\rho = \frac{(\lambda/n)\bar{r}}{ms} = \frac{\lambda\bar{r}}{mns} = \frac{\lambda\bar{r}}{Ms}.$$

Similarly, the utilization of an m'-core server is

$$\rho' = \frac{(\lambda/n')\bar{r}}{m's} = \frac{\lambda\bar{r}}{m'n's} = \frac{\lambda\bar{r}}{Ms},$$

which is the same as ρ. The average task response time of an m-core server is

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

and the average task response time of an m'-core server is

$$T' = \bar{x}\left(1 + \frac{p_{m'}}{m'(1-\rho)^2}\right).$$

To show that $T < T'$, we only need to show that $p_m/m < p_{m'}/m'$. It suffices to show that for a fixed ρ, p_m/m is a decreasing function of m.

In fact, p_m is a decreasing function of m. To show this, 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,

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

It suffices to show that $e^\rho > e\rho$, i.e., $1/\rho > e^{1-\rho}$, for all $0 \le \rho < 1$. Notice that

$$\frac{1}{\rho} = \frac{1}{1-(1-\rho)} = 1 + (1-\rho) + (1-\rho)^2 + (1-\rho)^3 + \cdots,$$

and

$$e^{1-\rho} = 1 + (1-\rho) + \frac{(1-\rho)^2}{2} + \frac{(1-\rho)^3}{6} + \cdots.$$

Hence, the inequality is obvious. □

By using the closed-form expression of p_m derived in the proof of Theorem 1, we get a closed-form expression of the average task response time as

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

The above closed-form expression of T will be repeatedly used in this paper. In Table 1, we display numerical data to show the relative error of the expression compared with the exact value of T. It can be seen that the closed-form expression of T is very accurate when m is not too small and ρ is not too large.

5.2 Power optimization

Theorem 2 *Consider a group of n m-core server processors and a group of n' m'-core server processors, where all the cores have the same speed s and $nm = n'm' = M$. Both groups consume the same amount of power.*

Table 1 Relative error of the closed-form approximation of T

m	$\rho=0.1$	$\rho=0.2$	$\rho=0.3$	$\rho=0.4$	$\rho=0.5$	$\rho=0.6$	$\rho=0.7$	$\rho=0.8$	$\rho=0.9$
1	−0.17%	−1.84%	−4.39%	−7.36%	−10.32%	−12.83%	−14.32%	−13.91%	−10.13%
2	0.02%	−0.08%	−0.67%	−2.01%	−4.21%	−7.12%	−10.21%	−12.35%	−11.06%
3	0.00%	0.00%	−0.11%	−0.61%	−1.83%	−4.02%	−7.13%	−10.36%	−11.03%
4	0.00%	0.00%	−0.02%	−0.20%	−0.84%	−2.36%	−5.05%	−8.61%	−10.65%
5	0.00%	0.00%	−0.00%	−0.07%	−0.41%	−1.44%	−3.65%	−7.16%	−10.13%
6	0.00%	0.00%	0.00%	−0.02%	−0.21%	−0.90%	−2.68%	−6.00%	−9.56%
7	0.00%	0.00%	0.00%	−0.01%	−0.11%	−0.58%	−2.00%	−5.05%	−9.00%
8	0.00%	0.00%	0.00%	−0.00%	−0.06%	−0.38%	−1.52%	−4.28%	−8.45%
9	0.00%	0.00%	0.00%	−0.00%	−0.03%	−0.25%	−1.17%	−3.66%	−7.93%
10	0.00%	0.00%	0.00%	0.00%	−0.02%	−0.17%	−0.90%	−3.14%	−7.44%
11	0.00%	0.00%	0.00%	0.00%	−0.01%	−0.12%	−0.71%	−2.71%	−6.99%
12	0.00%	0.00%	0.00%	0.00%	−0.00%	−0.08%	−0.56%	−2.35%	−6.57%
13	0.00%	0.00%	0.00%	0.00%	−0.00%	−0.06%	−0.44%	−2.05%	−6.18%
14	0.00%	0.00%	0.00%	0.00%	−0.00%	−0.04%	−0.35%	−1.79%	−5.82%
15	0.00%	0.00%	0.00%	0.00%	−0.00%	−0.03%	−0.28%	−1.58%	−5.48%
16	0.00%	0.00%	0.00%	0.00%	−0.00%	−0.02%	−0.23%	−1.39%	−5.18%
17	0.00%	0.00%	0.00%	0.00%	−0.00%	−0.01%	−0.19%	−1.23%	−4.89%
18	0.00%	0.00%	0.00%	0.00%	−0.00%	−0.01%	−0.15%	−1.09%	−4.63%
19	0.00%	0.00%	0.00%	0.00%	0.00%	−0.01%	−0.12%	−0.97%	−4.38%
20	0.00%	0.00%	0.00%	0.00%	0.00%	−0.01%	−0.10%	−0.87%	−4.15%

Proof We will prove the theorem for both core speed and power consumption models.

In the idle-speed model, an m-core server processor consumes power

$$P=\frac{\lambda}{n}\bar{r}s^{\alpha-1}+mP^*,$$

and a group of n m-core server processors consume power

$$nP=\lambda\bar{r}s^{\alpha-1}+mnP^*.$$

Similarly, an m'-core server processor consumes power

$$P'=\frac{\lambda}{n'}\bar{r}s^{\alpha-1}+m'P^*,$$

and a group of n' m'-core server processors consume power

$$n'P'=\lambda\bar{r}s^{\alpha-1}+m'n'P^*.$$

Clearly, we have

$$nP=n'P'=\lambda\bar{r}s^{\alpha-1}+MP^*.$$

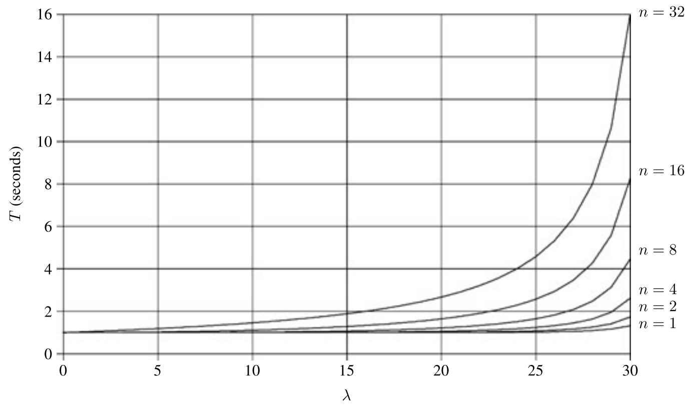


Fig. 1 Average task response time T vs. λ and n

In the constant-speed model, an m-core server processor consumes power

$$P = m\big(s^\alpha + P^*\big),$$

and an m'-core server processor consumes power

$$P' = m'\big(s^\alpha + P^*\big).$$

Again, we have

$$nP = n'P' = M\big(s^\alpha + P^*\big).$$

Hence, for both core speed models, a group of n m-core server processors consume the same amount of power as a group of n' m'-core server processors do. □

5.3 Numerical examples

In Fig. 1, we show the average task response time T (calculated using the exact expression) as a function of λ. We assume that there are $M = 32$ cores. The value of n is set as 1, 2, 4, 8, 16, and 32. The average task execution time is $\bar{x} = 1$ second. We observe that as n increases and m decreases, the average task response time T increases noticeably, as we have proved in Theorem 1. (Notice that T depends on m. Since $m = M/n$, T also depends on n.) Therefore, for a fixed total number M of cores, fewer multicore server processors of larger sizes perform better than more multicore server processors of smaller sizes. Consequently, an M-core server processor yields shorter response time than n m-core server processors do, for all $n > 1$ and $nm = M$. Furthermore, the M-core server processor consumes the same amount of power as n m-core server processors do. The implication of these results is that cores should be managed in a centralized way as suggested in cloud computing to provide the highest performance without increasing power consumption.

6 Server speed constrained optimization

In this section, we consider performance and power optimization with server speed constraints, where the server speed is defined as the product of server size and core speed, i.e., the combined core speed. In particular, we show that for a fixed server speed of a multicore server processor, fewer high-speed cores perform better than more low-speed cores. Furthermore, for a given server speed, there is an optimal selection of server size and core speed, such that the power consumption is minimized.

6.1 Performance optimization

Theorem 3 *An m-core server processor with core speed s yields shorter average task response time than an m'-core server processor with core speed s' does, where $m < m'$ and $s > s'$ and $ms = m's' = c$ and c is the server speed.*

Proof Assume that $ms = c$ for some fixed c. Then we have $\rho = \lambda\bar{r}/c$, which is also fixed. The average task response time is

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

We need to show that T is an increasing function of m, i.e., $\partial T/\partial m > 0$. Let us rewrite T as

$$T = \frac{\bar{r}}{c}\left(m + \frac{F}{1-\rho}\right),$$

where

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

Notice that

$$\begin{aligned}\frac{\partial F}{\partial m} &= -F^2\sqrt{2\pi}(1-\rho)\left(\frac{1}{2\sqrt{m}}\left(\frac{e^\rho}{e\rho}\right)^m + \sqrt{m}\left(\frac{e^\rho}{e\rho}\right)^m \ln\frac{e^\rho}{e\rho}\right)\\ &= -\sqrt{2\pi}(1-\rho)\left(\frac{e^\rho}{e\rho}\right)^m\left(\frac{1}{2\sqrt{m}} + (\rho - \ln\rho - 1)\sqrt{m}\right)F^2,\end{aligned}$$

and

$$\begin{aligned}\frac{\partial T}{\partial m} &= \frac{\bar{r}}{c}\left(1 + \frac{1}{(1-\rho)}\cdot\frac{\partial F}{\partial m}\right)\\ &= \frac{\bar{r}}{c}\left(1 - \sqrt{2\pi}\left(\frac{e^\rho}{e\rho}\right)^m\left(\frac{1}{2\sqrt{m}} + (\rho - \ln\rho - 1)\sqrt{m}\right)F^2\right)\\ &= \frac{\bar{r}}{c}\left(1 - \frac{\sqrt{2\pi}\left(e^\rho/e\rho\right)^m\left(1/(2\sqrt{m}) + (\rho - \ln\rho - 1)\sqrt{m}\right)}{\left(\sqrt{2\pi}(1-\rho)\sqrt{m}(e^\rho/e\rho)^m + 1\right)^2}\right).\end{aligned}$$

It suffices to show that

$$\left(\sqrt{2\pi}(1-\rho)\sqrt{m}\left(\frac{e^\rho}{e\rho}\right)^m+1\right)^2>\sqrt{2\pi}\left(\frac{e^\rho}{e\rho}\right)^m\left(\frac{1}{2\sqrt{m}}+(\rho-\ln\rho-1)\sqrt{m}\right).$$

The above inequality can be seen by observing that for a fixed ρ, the left-hand side has growth rate $\Theta(mG^{2m})$, while the right-hand side has growth rate $\Theta(\sqrt{m}G^m)$, where $G=e^\rho/e\rho>1$. □

6.2 Power optimization

Theorem 4 *For a fixed server speed c, there is an optimal selection of server size m and core speed s, such that an m-core server processor with core speed s consumes the minimum power.*

Proof In the idle-speed model, we have

$$P=\lambda\bar{r}s^{\alpha-1}+\frac{c}{s}P^*.$$

It is clear that when

$$\frac{\partial P}{\partial s}=\lambda\bar{r}(\alpha-1)s^{\alpha-2}-\frac{c}{s^2}P^*=0,$$

that is,

$$s=\left(\frac{cP^*}{\lambda\bar{r}(\alpha-1)}\right)^{1/\alpha}, \tag{1}$$

power consumption is minimized. In the constant-speed model, we have

$$P=c\left(s^{\alpha-1}+\frac{P^*}{s}\right).$$

It is clear that when

$$\frac{\partial P}{\partial s}=c\left((\alpha-1)s^{\alpha-2}-\frac{P^*}{s^2}\right)=0,$$

that is,

$$s=\left(\frac{P^*}{\alpha-1}\right)^{1/\alpha}, \tag{2}$$

power consumption is minimized. □

6.3 Numerical examples

In Fig. 2, we display the average task response time T (calculated using the exact expression) as a function of m. The task arrival rate λ is set as 14, 15, 16, 17, 18, and

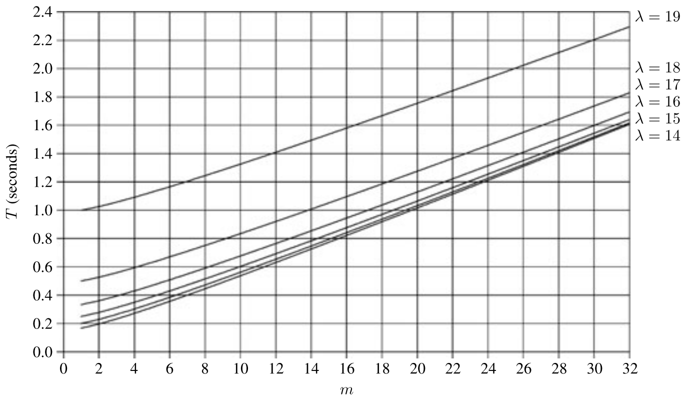


Fig. 2 Average task response time T vs. m and λ

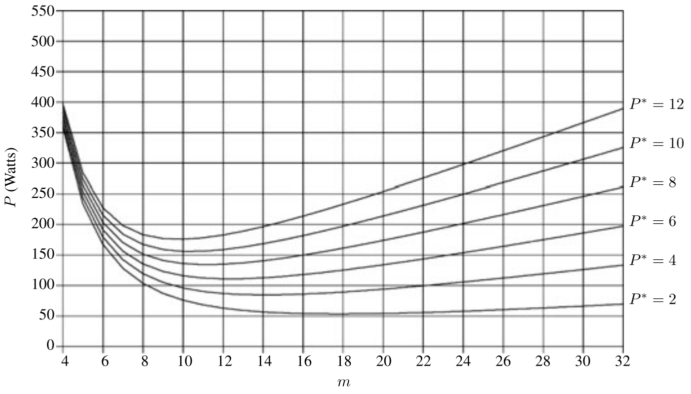


Fig. 3 Power consumption P vs. m and P^* (idle-speed model)

19 tasks per second. The server speed is $c = 20$ giga instructions per second. The task execution requirement is $\bar{r} = 1$ giga instructions. It is observed that T (in second) is an increasing function of m, as we have already proved in Theorem 3. Furthermore, from

$$T = \bar{x}\left(1 + \frac{p_m}{m(1-\rho)^2}\right) = \frac{\bar{r}}{c}\left(m + \frac{p_m}{(1-\rho)^2}\right),$$

where $\bar{r}$, c, and ρ are constants, which is indeed the case in Fig. 2, we know that T is almost a linear function of m, since p_m is very small compared with m, as we can observe from Fig. 2.

In Figs. 3 and 4, we further demonstrate the power consumption P (in Watt) as a function of server size m for the idle-speed model and the constant-speed model, respectively. The base power supply P^* is set as 2, 4, 6, 8, 10, and 12 watts. The server speed is $c = 20$ giga instructions per second. The task arrival rate is $\lambda = 14$ tasks per second. The task execution requirement is $\bar{r} = 1$ giga instructions. It is observed that when m is too small, the high-speed cores consume significant amount

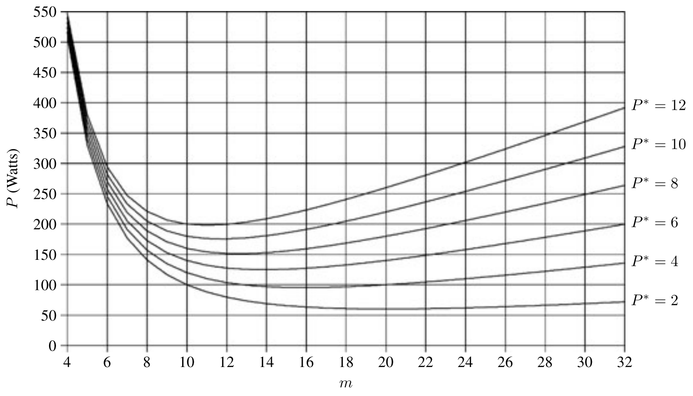


Fig. 4 Power consumption P vs. m and P^* (constant-speed model)

of energy. Hence, we only show P for $m \geq 4$. On the other hand, when m is too large, the overhead P^* in power consumption also causes increased energy waste. Hence, there is an optimal choice of m and s, such that an m-core server processor with core speed s consumes the minimum power and that $c = ms$ is fixed, as claimed in Theorem 4.

For the idle-speed model, the optimal server size m is 16, 13, 11, 10, 9, and 9, when P^* is 2, 4, 6, 8, 10, and 12, respectively. For the constant-speed model, the optimal server size m is 20, 16, 14, 13, 12, and 11, when P^* is 2, 4, 6, 8, 10, and 12, respectively. These values can be obtained by calculating the optimal core speed s using (1) and (2) for the two core speed models and the relation $m = c/s$, and then rounding the solutions to the nearest integers. It is clear that for both core speed models and all values of P^*, the power consumption P when m is a nonoptimal choice can be much more than that when m is an optimal choice.

Our results can be used as design guidelines for a multicore server processor. For instance, for the constant-speed model, when $P^* = 10$ Watts, the power consumption is $P = 205.00$ watts when $m = 8$ and $P = 204.69$ watts when $m = 18$, which are about the same. However, when $\lambda = 14$ tasks per second, the average task response time is $T = 0.445$ seconds when $m = 8$, which is less than half of $T = 0.918$ seconds when $m = 18$. Therefore, a better choice is certainly $m = 8$ and $s = 2.5$ giga instructions per second.

7 Power constrained performance optimization

In this section, we consider power constrained performance optimization, i.e., average task response time optimization with power consumption constraints. In particular, we show that for a given power supply, there is an optimal selection of server size and core speed, such that the average task response time is minimized, while power consumption does not exceed the supply.

7.1 The method

Theorem 5 *Given a fixed power supply P, there is an optimal selection of server size m and core speed s, such that an m-core server processor with core speed s has the minimum average task response time and that it consumes power P.*

Proof Recall that by using the closed-form expression of p_m derived in Sect. 5, we get a closed-form expression of the average task response time as

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

In the constant-speed model, since

$$P = \lambda \bar{r} s^{\alpha-1} + mP^*,$$

we get

$$s = \left(\frac{P - mP^*}{\lambda \bar{r}}\right)^{1/(\alpha-1)}. \tag{3}$$

In the constant-speed model, since

$$P = m\big(s^{\alpha} + P^*\big),$$

we get

$$s = \left(\frac{P}{m} - P^*\right)^{1/\alpha}. \tag{4}$$

Let us rewrite T as

$$T = \frac{\bar{r}}{s}(1 + F),$$

where

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

Therefore, we get

$$\frac{\partial T}{\partial m} = \bar{r}\left(-\frac{1+F}{s^2}\cdot\frac{\partial s}{\partial m} + \frac{1}{s}\cdot\frac{\partial F}{\partial m}\right) = \frac{\bar{r}}{s}\left(-\frac{1+F}{s}\cdot\frac{\partial s}{\partial m} + \frac{\partial F}{\partial m}\right).$$

Notice that

$$\begin{aligned}
\frac{\partial s}{\partial m} &= \frac{1}{\alpha-1}\left(\frac{P - mP^*}{\lambda \bar{r}}\right)^{1/(\alpha-1)-1}\left(-\frac{P^*}{\lambda \bar{r}}\right) \\
&= -\frac{P^*}{\lambda \bar{r}(\alpha-1)}\left(\frac{P - mP^*}{\lambda \bar{r}}\right)^{-(\alpha-2)/(\alpha-1)} \\
&= -\frac{P^*}{\lambda \bar{r}(\alpha-1)} s^{-(\alpha-2)},
\end{aligned}$$

in the idle-speed model, and

$$\begin{aligned}\frac{\partial s}{\partial m} &= \frac{1}{\alpha}\left(\frac{P}{m} - P^*\right)^{1/\alpha-1}\left(-\frac{P}{m^2}\right) \\ &= -\frac{P}{\alpha m^2}\left(\frac{P}{m} - P^*\right)^{-(\alpha-1)/\alpha} \\ &= -\frac{P}{\alpha m^2}s^{-(\alpha-1)},\end{aligned}$$

in the constant-speed model.

We rewrite F as

$$F = \frac{1}{m(1-\rho)(\sqrt{2\pi m}(1-\rho)G+1)} = \frac{1}{\sqrt{2\pi}m^{3/2}(1-\rho)^2G + m(1-\rho)},$$

where

$$G = (e^\rho/e\rho)^m.$$

Notice that

$$\ln G = m\ln\bigl(e^\rho/e\rho\bigr) = m(\rho - \ln\rho - 1).$$

Since

$$\frac{\partial\rho}{\partial m} = -\frac{\lambda\bar{r}}{m^2 s} = -\frac{\rho}{m},$$

we get

$$\frac{1}{G}\frac{\partial G}{\partial m} = (\rho - \ln\rho - 1) + m\left(1 - \frac{1}{\rho}\right)\frac{\partial\rho}{\partial m} = -\ln\rho,$$

and

$$\frac{\partial G}{\partial m} = -G\ln\rho.$$

Now, we have

$$\begin{aligned}\frac{\partial F}{\partial m} &= -F^2\left(\sqrt{2\pi}\left(\frac{3}{2}\sqrt{m}(1-\rho)^2G + m^{3/2}2(1-\rho)\left(-\frac{\partial\rho}{\partial m}\right)G\right.\right. \\ &\quad \left.\left. + m^{3/2}(1-\rho)^2\frac{\partial G}{\partial m}\right) + (1-\rho) + m\left(-\frac{\partial\rho}{\partial m}\right)\right) \\ &= -F^2\left(\sqrt{2\pi}\left(\frac{3}{2}\sqrt{m}(1-\rho)^2G + \sqrt{m}2\rho(1-\rho)G - m^{3/2}(\ln\rho)(1-\rho)^2G\right)\right. \\ &\quad \left. + (1-\rho) + \rho\right) \\ &= -F^2\left(\sqrt{2\pi m}(1-\rho)\left(\frac{3}{2}(1-\rho) + 2\rho - m(\ln\rho)(1-\rho)\right)G + 1\right)\end{aligned}$$

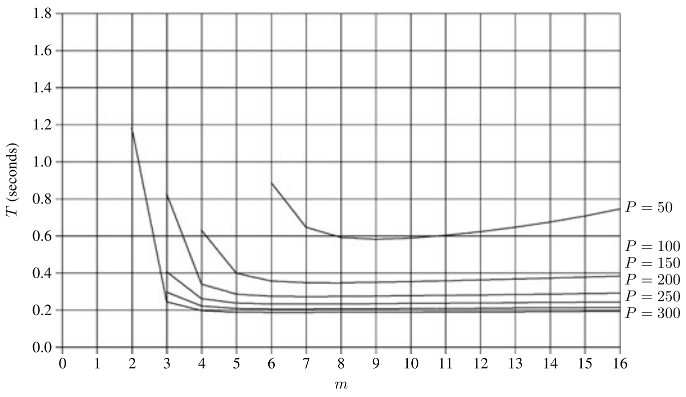


Fig. 5 Average task response time T vs. m and P (idle-speed model)

$$= -F^2\left(\sqrt{2\pi m}(1-\rho)\left(\frac{\rho+3}{2} - m(\ln\rho)(1-\rho)\right)G + 1\right)$$
$$= -F^2\left(\sqrt{2\pi m}(1-\rho)\left(\frac{\rho+3}{2} - m(\ln\rho)(1-\rho)\right)\left(\frac{e^\rho}{e\rho}\right)^m + 1\right).$$

Summarizing the above discussion, we get

$$\frac{\partial T}{\partial m} = \frac{\bar{r}}{s}y,$$

where

$$y = \frac{P^*}{\lambda\bar{r}(\alpha-1)} \cdot \frac{1+F}{s^{(\alpha-1)}} - F^2\left(\sqrt{2\pi m}(1-\rho)\left(\frac{\rho+3}{2} - m(\ln\rho)(1-\rho)\right)\left(\frac{e^\rho}{e\rho}\right)^m + 1\right), \quad (5)$$

in the idle-speed model, and

$$y = \frac{P}{\alpha m^2} \cdot \frac{1+F}{s^\alpha} - F^2\left(\sqrt{2\pi m}(1-\rho)\left(\frac{\rho+3}{2} - m(\ln\rho)(1-\rho)\right)\left(\frac{e^\rho}{e\rho}\right)^m + 1\right), \quad (6)$$

in the constant-speed model. The optimal choice of m can be obtained by solving the equation $y = 0$. (In fact, we have $y < 0$ for small m and $y > 0$ as m increases. Although y is not an increasing function of m, there is a unique m that yields $y = 0$.) □

7.2 Numerical examples

In Figs. 5 and 6, we demonstrate the average task response time T (in second) as a function of server size m for the idle-speed model and the constant-speed model, respectively. The power supply P is set as 50, 100, 150, 200, 250, and 300 watts.

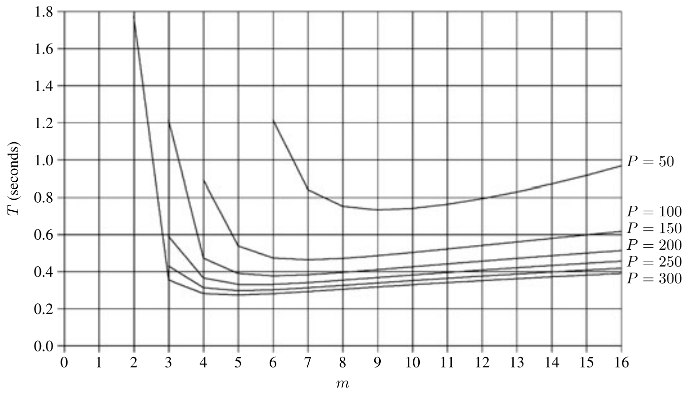


Fig. 6 Average task response time T vs. m and P (constant-speed model)

The task arrival rate is $\lambda = 10$ tasks per second. The task execution requirement is $\bar{r} = 1$ giga instructions. The base power supply is $P^* = 2$ Watts. Let $c = ms$ be the combined speed and computing power of an m-core server processor with core speed s. Then we get

$$c = m\left(\frac{P - mP^*}{\lambda\bar{r}}\right)^{1/(\alpha-1)},$$

in the idle-speed model, and

$$c = m\left(\frac{P}{m} - P^*\right)^{1/\alpha},$$

in the constant-speed model. It is easy to verify that

$$\frac{\partial c}{\partial m} = \frac{(P - mP^*)^{(2-\alpha)/(\alpha-1)}}{(\lambda\bar{r})^{1/(\alpha-1)}}\left(P - \left(\frac{\alpha}{\alpha-1}\right)mP^*\right),$$

in the idle-speed model, and

$$\frac{\partial c}{\partial m} = \left(\frac{P}{m} - P^*\right)^{(1-\alpha)/\alpha}\left(\left(\frac{\alpha-1}{\alpha}\right)\frac{P}{m} - P^*\right),$$

in the constant-speed model. Hence, if

$$P > \left(\frac{\alpha}{\alpha-1}\right)mP^*,$$

which is indeed the case in our example, c is an increasing function of m for both core speed models. It is observed that when the server size m in decreased, the core speed s is increased. However, the server speed c is decreased. When m is too small, the increment in s is not enough for the server to provide enough total speed $c = ms$ to handle a given workload, i.e., $\rho = \lambda\bar{r}/c$ will exceed 1. In the figures, we only show

T for m large enough so that $\rho = \lambda\bar{r}/c < 1$. On the other hand, when m is too large, the overhead P^* in power consumption eats up the performance. Hence, there is an optimal selection of server size m, so that the minimum task response time T can be achieved based on given power constraint P, as we have proved in Theorem 5.

For the idle-speed model, the optimal server size m is 9, 8, 7, 7, 6, and 6, when P is 50, 100, 150, 200, 250, and 300 watts, respectively. For the constant-speed model, the optimal server size m is 9, 7, 6, 6, 5, and 5, when P is 50, 100, 150, 200, 250, and 300 watts, respectively. These values can be obtained by solving the equation $y = 0$ numerically, where y is given in (5) and (6) for the two core speed models, and then rounding the solutions to the nearest integers. The optimal core speed s can be obtained from (3) and (4) for the two core speed models.

It is observed that for both core speed models and all values of P, the average task response time T when m is a nonoptimal choice (especially when m is very small) can be much longer than that when m is an optimal choice.

8 Performance constrained power optimization

In this section, we consider performance constrained power optimization, i.e., minimizing power consumption with task response time constraints. In particular, we show that for a given average task response time, there is an optimal selection of server size and core speed, such that minimum power consumption can be achieved while a given performance guarantee is maintained.

8.1 The method

Theorem 6 *For a given average task response time T, there is an optimal choice of server size m and core speed s, such that an m-core server processor with core speed s consumes the minimum power and that the task response time is T.*

Proof Let us rewrite the average task response time as

$$T = \frac{\rho}{\lambda}\left(m + \frac{1}{(1-\rho)(\sqrt{2\pi m}(1-\rho)(e^{\rho}/e\rho)^m + 1)}\right).$$

It is clear that given m, λ, and T, there is unique ρ which satisfies the above equation. Although there is no closed-form solution, ρ can be found easily by numerical methods. Based on $\rho = \lambda\bar{r}/ms$, we can determine s and P.

We will view the core speed s as a function of server size m, i.e.,

$$s = \frac{\lambda\bar{r}}{m\rho}.$$

To minimize P, we need to have $\partial P/\partial m = 0$. We notice that in the idle-speed model, we have $P = \lambda\bar{r}s^{\alpha-1} + mP^*$ and

$$\frac{\partial P}{\partial m} = \lambda\bar{r}(\alpha-1)s^{\alpha-2}\frac{\partial s}{\partial m} + P^* = (\alpha-1)\lambda\bar{r}\left(\frac{\lambda\bar{r}}{m\rho}\right)^{\alpha-2}\frac{\partial s}{\partial m} + P^*.$$

In the constant-speed model, we have $P = m(s^\alpha + P^*)$ and

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

Furthermore, we have

$$\frac{\partial s}{\partial m} = \lambda\bar{r}\left(-\frac{1}{m^2\rho} - \frac{1}{m\rho^2}\cdot\frac{\partial\rho}{\partial m}\right) = -\frac{\lambda\bar{r}}{m\rho}\left(\frac{1}{m} + \frac{1}{\rho}\cdot\frac{\partial\rho}{\partial m}\right) = -s\left(\frac{1}{m} + \frac{1}{\rho}\cdot\frac{\partial\rho}{\partial m}\right).$$

It remains to find $\partial\rho/\partial m$, which can be obtained even though there is no closed-form expression of ρ in terms of m. We rewrite the above equation of T as

$$\rho(m+F) = T\lambda,$$

where

$$F = \frac{1}{(1-\rho)(\sqrt{2\pi m}(1-\rho)(e^\rho/e\rho)^m + 1)} = \frac{1}{\sqrt{2\pi m}(1-\rho)^2 G + (1-\rho)},$$

with

$$G = \left(\frac{e^\rho}{e\rho}\right)^m.$$

By taking the partial derivative of m on both sides of the equation (notice that T is a fixed value, i.e., the time constraint), we get

$$\frac{\partial\rho}{\partial m}(m+F) + \rho\left(1 + \frac{\partial F}{\partial m}\right) = 0.$$

Since

$$\ln G = m\ln\bigl(e^\rho/e\rho\bigr) = m(\rho - \ln\rho - 1),$$

and

$$\frac{1}{G}\frac{\partial G}{\partial m} = (\rho - \ln\rho - 1) + m\left(1 - \frac{1}{\rho}\right)\frac{\partial\rho}{\partial m},$$

we get

$$\frac{\partial G}{\partial m} = G\left((\rho - \ln\rho - 1) + m\left(1 - \frac{1}{\rho}\right)\frac{\partial\rho}{\partial m}\right).$$

Now, we have

$$\begin{aligned}\frac{\partial F}{\partial m} &= -F^2\left(\sqrt{2\pi}\left(\frac{1}{2\sqrt{m}}(1-\rho)^2 G + \sqrt{m}2(1-\rho)\left(-\frac{\partial\rho}{\partial m}\right)G\right.\right.\\ &\qquad \left.\left. + \sqrt{m}(1-\rho)^2\frac{\partial G}{\partial m}\right) - \frac{\partial\rho}{\partial m}\right)\\ &= -F^2\left(\sqrt{2\pi}\left(\frac{1}{2\sqrt{m}}(1-\rho)^2 G - 2\sqrt{m}(1-\rho)G\frac{\partial\rho}{\partial m}\right.\right.\end{aligned}$$

$$
\begin{aligned}
&\quad + \sqrt{m}(1-\rho)^2 G\left((\rho - \ln\rho - 1) + m\left(1 - \frac{1}{\rho}\right)\frac{\partial\rho}{\partial m}\right)\Bigg) - \frac{\partial\rho}{\partial m}\Bigg) \\
&= -F^2\Bigg(\sqrt{\frac{\pi}{2m}}(1-\rho)^2 G + \sqrt{2\pi}\Bigg(-2\sqrt{m}(1-\rho)G\frac{\partial\rho}{\partial m} \\
&\quad + \sqrt{m}(1-\rho)^2 G(\rho - \ln\rho - 1) - m^{3/2}\frac{(1-\rho)^3}{\rho}G\frac{\partial\rho}{\partial m}\Bigg) - \frac{\partial\rho}{\partial m}\Bigg) \\
&= -F^2\Bigg(\sqrt{\frac{\pi}{2m}}(1-\rho)^2 G + \sqrt{2\pi m}(1-\rho)^2 G(\rho - \ln\rho - 1) \\
&\quad + \sqrt{2\pi}\left(-2\sqrt{m}(1-\rho)G\frac{\partial\rho}{\partial m} - m^{3/2}\frac{(1-\rho)^3}{\rho}G\frac{\partial\rho}{\partial m}\right) - \frac{\partial\rho}{\partial m}\Bigg) \\
&= -F^2\Bigg(\left(\sqrt{\frac{\pi}{2m}} + \sqrt{2\pi m}(\rho - \ln\rho - 1)\right)(1-\rho)^2 G \\
&\quad - \sqrt{2\pi m}(1-\rho)\left(2 + m\frac{(1-\rho)^2}{\rho}\right)G\frac{\partial\rho}{\partial m} - \frac{\partial\rho}{\partial m}\Bigg) \\
&= -F^2\Bigg(\left(\sqrt{\frac{\pi}{2m}} + \sqrt{2\pi m}(\rho - \ln\rho - 1)\right)(1-\rho)^2 G \\
&\quad - \left(\sqrt{2\pi m}(1-\rho)\left(2 + m\frac{(1-\rho)^2}{\rho}\right)G + 1\right)\frac{\partial\rho}{\partial m}\Bigg).
\end{aligned}
$$

The last equation implies that

$$
\begin{aligned}
&(m+F)\frac{\partial\rho}{\partial m} + \rho\Bigg(1 - F^2\left(\sqrt{\frac{\pi}{2m}} + \sqrt{2\pi m}(\rho - \ln\rho - 1)\right)(1-\rho)^2 G \\
&\quad + F^2\left(\sqrt{2\pi m}(1-\rho)\left(2 + m\frac{(1-\rho)^2}{\rho}\right)G + 1\right)\frac{\partial\rho}{\partial m}\Bigg) = 0,
\end{aligned}
$$

that is,

$$
\begin{aligned}
&\left(m + F + \rho F^2\left(\sqrt{2\pi m}(1-\rho)\left(2 + m\frac{(1-\rho)^2}{\rho}\right)G + 1\right)\right)\frac{\partial\rho}{\partial m} \\
&\quad = \rho\left(F^2\left(\sqrt{\frac{\pi}{2m}} + \sqrt{2\pi m}(\rho - \ln\rho - 1)\right)(1-\rho)^2 G - 1\right),
\end{aligned}
$$

which gives rise to

$$
\frac{\partial\rho}{\partial m} = \frac{\rho(F^2(\sqrt{\pi/2m} + \sqrt{2\pi m}(\rho - \ln\rho - 1))(1-\rho)^2 G - 1)}{m + F + \rho F^2(\sqrt{2\pi m}(1-\rho)(2 + m(1-\rho)^2/\rho)G + 1)}.
$$

Summarizing the above discussion, we obtain

$$
\frac{\partial s}{\partial m} = -s\left(\frac{1}{m} + \frac{F^2(\sqrt{\pi/2m} + \sqrt{2\pi m}(\rho - \ln\rho - 1))(1-\rho)^2 G - 1}{m + F + \rho F^2(\sqrt{2\pi m}(1-\rho)(2 + m(1-\rho)^2/\rho)G + 1)}\right).
$$

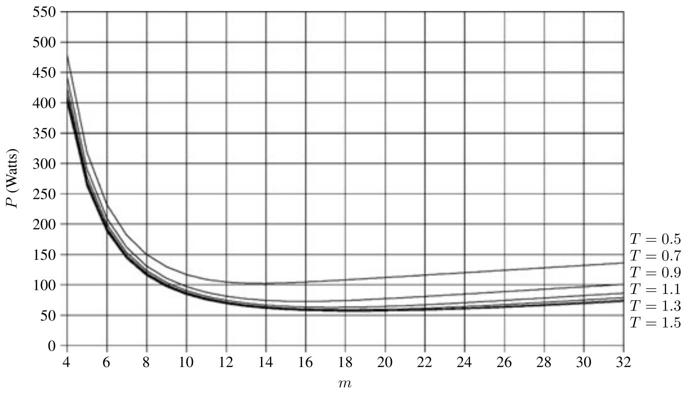


Fig. 7 Power consumption P vs. m and T (idle-speed model)

and

$$\begin{aligned}\frac{\partial P}{\partial m} &= -\lambda\bar{r}(\alpha-1)s^{\alpha-1}\\ &\quad\times\left(\frac{1}{m}+\frac{F^2\big(\sqrt{\pi/2m}+\sqrt{2\pi m}(\rho-\ln\rho-1)\big)(1-\rho)^2G-1}{m+F+\rho F^2\big(\sqrt{2\pi m}(1-\rho)\big(2+m(1-\rho)^2/\rho\big)G+1\big)}\right)\\ &\quad+P^*, \end{aligned} \tag{7}$$

in the idle-speed model, and

$$\begin{aligned}\frac{\partial P}{\partial m} &= \left(1-\alpha m\left(\frac{1}{m}+\frac{F^2\big(\sqrt{\pi/2m}+\sqrt{2\pi m}(\rho-\ln\rho-1)\big)(1-\rho)^2G-1}{m+F+\rho F^2\big(\sqrt{2\pi m}(1-\rho)\big(2+m(1-\rho)^2/\rho\big)G+1\big)}\right)\right)s^{\alpha}\\ &\quad+P^*, \end{aligned} \tag{8}$$

in the constant-speed model. □

8.2 Numerical examples

In Figs. 7 and 8, we demonstrate the power consumption P (in watt) as a function of server size m for the idle-speed model and the constant-speed model, respectively. The time constraint T is set as 0.5, 0.7, 0.9, 1.1, 1.3, and 1.5 seconds. The task arrival rate λ is 18 tasks per second. The task execution requirement is $\bar{r} = 1$ giga instructions. The base power supply is $P^* = 2$ watts. Again, there is an optimal choice of m and s, such that an m-core server processor with core speed s consumes the minimum power while achieving the given average task response time T, as shown in Theorem 6.

For the idle-speed model, the optimal server size m is 14, 16, 17, 18, 18, and 18, when T is 0.5, 0.7, 0.9, 1.1, 1.3, and 1.5 seconds, respectively. For the constant-speed model, the optimal server size m is 12, 15, 17, 18, 18, and 19, when T is 0.5, 0.7, 0.9, 1.1, 1.3, and 1.5 seconds, respectively. These values can be obtained by solving the equation $\partial P/\partial m = 0$ numerically, where $\partial P/\partial m$ is given in (7) and (8) for the

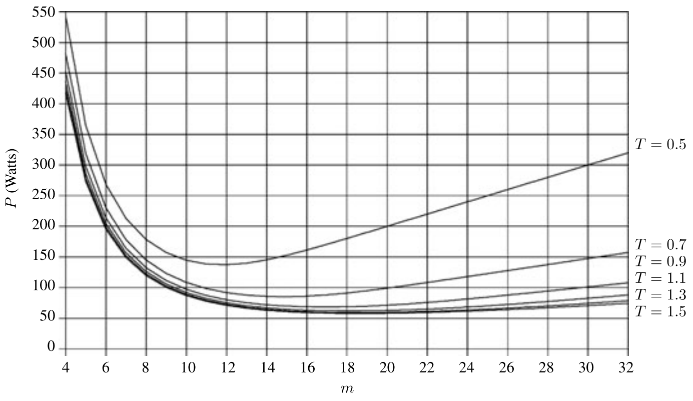


Fig. 8 Power consumption P vs. m and T (constant-speed model)

two core speed models, and then rounding the solutions to the nearest integers. The optimal core speed s can be obtained by using the relation $s = \lambda\bar{r}/m\rho$.

It is observed that for both core speed models, the average power consumption P when m is a nonoptimal choice (especially when m is very small) can be much more than that when m is an optimal choice.

9 Conclusions

By using an M/M/m queueing system with multiple servers to model a multicore server processor, and considering system performance measures of average task response time and average power consumption, we have studied the problem of power and performance management for a multicore server processor in a cloud computing environment by optimal server configuration for a specific application environment. Our conclusions are (1) cores should be managed in a centralized way to provide the highest performance without consumption of more energy in cloud computing; (2) for a given server speed constraint, fewer high-speed cores perform better than more low-speed cores; furthermore, there is an optimal selection of server size and core speed which can be obtained analytically, such that a multicore server processor consumes the minimum power; (3) for a given power consumption constraint, there is an optimal selection of server size and core speed which can be obtained numerically, such that the best performance can be achieved, i.e., the average task response time is minimized; (4) for a given task response time constraint, there is an optimal selection of server size and core speed which can be obtained numerically, such that minimum power consumption can be achieved while the given performance guarantee is maintained. Further investigation can be directed toward application of our models and results to multicore server processors in real cloud computing environments.

References

1. Albers S (2010) Energy-efficient algorithms. Commun ACM 53(5):86–96
2. Aydin H, Melhem R, Mossé D, Mejía-Alvarez P (2004) Power-aware scheduling for periodic real-time tasks. IEEE Trans Comput 53(5):584–600
3. Bansal N, Kimbrel T, Pruhs K (2004) Dynamic speed scaling to manage energy and temperature. In: Proceedings of the 45th IEEE symposium on foundation of computer science, pp 520–529
4. Barnett JA (2005) Dynamic task-level voltage scheduling optimizations. IEEE Trans Comput 54(5):508–520
5. Benini L, Bogliolo A, De Micheli G (2000) A survey of design techniques for system-level dynamic power management. IEEE Trans Very Large Scale Integr (VLSI) Syst 8(3):299–316
6. Bunde DP (2006) Power-aware scheduling for makespan and flow. In: Proceedings of the 18th ACM symposium on parallelism in algorithms and architectures, pp 190–196
7. Chan H-L, Chan W-T, Lam T-W, Lee L-K, Mak K-S, Wong PWH (2007) Energy efficient online deadline scheduling. In: Proceedings of the 18th ACM-SIAM symposium on discrete algorithms, pp 795–804
8. Chandrakasan AP, Sheng S, Brodersen RW (1992) Low-power CMOS digital design. IEEE J Solid-State Circuits 27(4):473–484
9. Cho S, Melhem RG (2010) On the interplay of parallelization, program performance, and energy consumption. IEEE Trans Parallel Distrib Syst 21(3):342–353
10. Hong I, Kirovski D, Qu G, Potkonjak M, Srivastava MB (1999) Power optimization of variable-voltage core-based systems. IEEE Trans Comput-Aided Des Integr Circuits Syst 18(12):1702–1714
11. http://multicore.amd.com/us-en/AMD-multicore/multicore-Advantages.aspx
12. http://www.intel.com/technology/intelligentpower/index.htm
13. Im C, Ha S, Kim H (2004) Dynamic voltage scheduling with buffers in low-power multimedia applications. ACM Trans Embed Comput Syst 3(4):686–705
14. Intel, Automated energy efficiency for the intelligent business. White Paper
15. Khan SU, Ahmad I (2009) A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids. IEEE Trans Parallel Distrib Syst 20(3):346–360
16. Khargharia B, Hariri S, Szidarovszky F, Houri M, El-Rewini H, Khan S, Ahmad I, Yousif MS (2007) Autonomic power and performance management for large-scale data centers. NFS next generation software program
17. Kleinrock L (1975) Queueing systems, volume 1: Theory. Wiley, New York
18. Krishna CM, Lee Y-H (2003) Voltage-clock-scaling adaptive scheduling techniques for low power in hard real-time systems. IEEE Trans Comput 52(12):1586–1593
19. Kwon W-C, Kim T (2005) Optimal voltage allocation techniques for dynamically variable voltage processors. ACM Trans Embed Comput Syst 4(1):211–230
20. Lee YC, Zomaya AY (2011) Energy conscious scheduling for distributed computing systems under different operating conditions. IEEE Trans Parallel Distrib Syst 22(8):1374–1381
21. Lee Y-H, Krishna CM (2003) Voltage-clock scaling for low energy consumption in fixed-priority real-time systems. Real-Time Syst 24(3):303–317
22. Li K (2008) Performance analysis of power-aware task scheduling algorithms on multiprocessor computers with dynamic voltage and speed. IEEE Trans Parallel Distrib Syst 19(11):1484–1497
23. Li K Energy efficient scheduling of parallel tasks on multiprocessor computers. J Supercomput. doi:10.1007/s11227-010-0416-0
24. Li K (2011) Power allocation and task scheduling on multiprocessor computers with energy and time constraints. In: Lee Y-C, Zomaya A (eds) Energy aware distributed computing systems. Wiley series on parallel and distributed computing, vol 1
25. Li K (2011) Algorithms and analysis of energy-efficient scheduling of parallel tasks. In: Ranka S, Ahmad I (eds) Handbook of energy-aware and green computing. Chapman and Hall/CRC Press, London
26. Li M, Yao FF (2006) An efficient algorithm for computing optimal discrete voltage schedules. SIAM J Comput 35(3):658–671
27. Li M, Liu BJ, Yao FF (2006) Min-energy voltage allocation for tree-structured tasks. J Comb Optim 11:305–319
28. Li M, Yao AC, Yao FF (2006) Discrete and continuous min-energy schedules for variable voltage processors. Proc Natl Acad Sci USA 103(11):3983–3987
29. Lorch JR, Smith AJ (2004) PACE: a new approach to dynamic voltage scaling. IEEE Trans Comput 53(7):856–869

30. Mahapatra RN, Zhao W (2005) An energy-efficient slack distribution technique for multimode distributed real-time embedded systems. IEEE Trans Parallel Distrib Syst 16(7):650–662
31. Quan G, Hu XS (2007) Energy efficient DVS schedule for fixed-priority real-time systems. ACM Trans Embed Comput Syst 6(4):29
32. Rusu C, Melhem R, Mossé D (2002) Maximizing the system value while satisfying time and energy constraints. In: Proceedings of the 23rd IEEE real-time systems symposium, pp 256–265
33. Shin D, Kim J (2003) Power-aware scheduling of conditional task graphs in real-time multiprocessor systems. In: Proceedings of the international symposium on low power electronics and design, pp 408–413
34. Shin D, Kim J, Lee S (2001) Intra-task voltage scheduling for low-energy hard real-time applications. IEEE Des Test Comput 18(2):20–30
35. Stan MR, Skadron K (2003) Guest editors' introduction: power-aware computing. IEEE Comput 36(12):35–38
36. Unsal OS, Koren I (2003) System-level power-aware design techniques in real-time systems. Proc IEEE 91(7):1055–1069
37. Venkatachalam V, Franz M (2005) Power reduction techniques for microprocessor systems. ACM Comput Surv 37(3):195–237
38. Wang X, Wang Y (2011) Coordinating power control and performance management for virtualized server clusters. IEEE Trans Parallel Distrib Syst 22(2):245–259
39. Wang X, Chen M, Lefurgy C, Keller TW (2009) SHIP: scalable hierarchical power control for large-scale data centers. In: Proceedings of the 18th international conference on parallel architectures and compilation techniques, pp 91–100
40. Weiser M, Welch B, Demers A, Shenker S (1994) Scheduling for reduced CPU energy. In: Proceedings of the 1st USENIX symposium on operating systems design and implementation, pp 13–23
41. Yang P, Wong C, Marchal P, Catthoor F, Desmet D, Verkest D, Lauwereins R (2001) Energy-aware runtime scheduling for embedded-multiprocessor SOCs. IEEE Des Test Comput 18(5):46–58
42. Yao F, Demers A, Shenker S (1995) A scheduling model for reduced CPU energy. In: Proceedings of the 36th IEEE symposium on foundations of computer science, pp 374–382
43. Yun H-S, Kim J (2003) On energy-optimal voltage scheduling for fixed-priority hard real-time systems. ACM Trans Embed Comput Syst 2(3):393–430
44. 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
45. Zheng X, Cai Y (2010) Optimal server provisioning and frequency adjustment in server clusters. In: 39th international conference on parallel processing workshops, pp 504–511
46. Zheng X, Cai Y (2010) Optimal server allocation and frequency modulation on multi-core based server clusters. Int J Green Comput 1(2):18–30
47. Zheng X, Cai Y (2010) Achieving energy proportionality in server clusters. Int J Comput Netw 1(2):21–35
48. Zhong X, Xu C-Z (2007) Energy-aware modeling and scheduling for dynamic voltage scaling with statistical real-time guarantee. IEEE Trans Comput 56(3):358–372
49. Zhu D, Melhem R, Childers BR (2003) Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems. IEEE Trans Parallel Distrib Syst 14(7):686–700
50. Zhu D, Mossé D, Melhem R (2004) Power-aware scheduling for AND/OR graphs in real-time systems. IEEE Trans Parallel Distrib Syst 15(9):849–864
51. Zhuo J, Chakrabarti C (2008) Energy-efficient dynamic task scheduling algorithms for DVS systems. ACM Trans Embed Comput Syst 7(2):17