IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 3, 2018, accepted September 25, 2018, date of publication October 1, 2018, date of current version October 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2872638

Optimal Temporal Partitioning of a Multicore Server Processor for Virtual Machine Allocation

KEQIN LI, (Fellow, IEEE)
Department of Computer Science, State University of New York, New Paltz, NY 12561, USA
e-mail: lik@newpaltz.edu

This work was supported by SUNY New Paltz through the Professional Expenses Fund.

ABSTRACT The problem of optimal temporal partitioning of a multicore server processor for virtual machine allocation in cloud computing is addressed as multivariable optimization problems and solved algorithmically and numerically. Analytical models for virtual machines are developed, i.e., partially available multi-server systems. The problem of optimal temporal partitioning of a multicore server processor for virtual machine allocation is formulated, where the overall performance (i.e., the average task response time) of a group of virtual machines is optimized. An algorithm is developed to solve the problem numerically. The problem of optimal temporal partitioning of a multicore server processor with power consumption constraint is also formulated and solved, where the overall performance of a group of virtual machines is optimized and the total power consumption of the virtual machines does not exceed certain available power. A virtual machine is treated as a queuing system with multiple servers, i.e., an M/M/m queuing system. 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. Numerical examples are presented to demonstrate our methods.

INDEX TERMS Energy consumption, multicore server processor, queuing model, response time, temporal partitioning, virtual machine.

I. INTRODUCTION

A. MOTIVATION

Cloud computing provides an effective way of sharing computing resources over the Internet to achieve economies of scale similar to a utility such as the electricity grid [1]. Sharing of computing resources can be implemented in such a way that a large server is partitioned into multiple small servers or a physical server is configured into multiple virtual servers. These small or virtual servers are called *virtual machines* (VM) in cloud computing, which are implemented on a large multicore server processor, where multiple virtual machines can be deployed simultaneously [5]. Each VM acts as a physically or logically independent and self-contained server with its own processor cores, main memory, input/output devices, and network resources, and its own copy of an operating system. Each VM is employed to run one type of applications, or allocated to an end user for his/her own applications.

The partitioning and sharing of a physical server can be implemented in two different ways, namely, spatial partitioning and temporal partitioning [7]. *Spatial partitioning* (horizontal partitioning) involves the ability to divide a single large server into multiple physically independent small servers, with each partition functioning independently [4]. Each partition is typically 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. Server partitioning technologies allow system administrators to consolidate multiple applications into one physical server box, thereby promoting centralized server management, saving space, and reducing administrative and management costs. Server and system partitioning technology has been around for a while in the mainframe space and large-scale parallel processing systems [10], [18], [22], but it started to gain attention in distributed, grid, Internet computing only in the past few years.

Temporal partitioning (vertical partitioning) involves the ability to divide a time interval into multiple subintervals, with each subinterval allocated to a logically independent VM. The underlying concept of cloud computing comes from time-sharing [21], i.e., allowing multiple users to share a costly large-scale mainframe for eliminating periods of inactivity and a greater return of investment [3]. Server virtualization is a major priority for large enterprises, since it provides an effective technology for consolidation, i.e., reducing

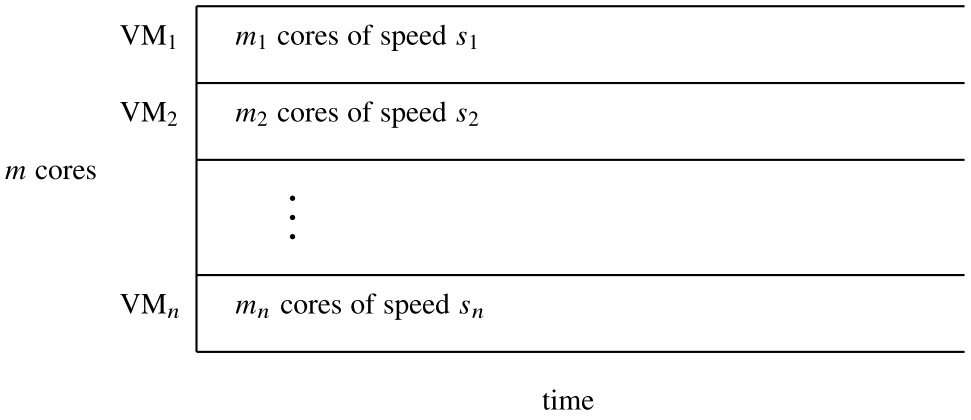

FIGURE 1. A spatial (horizontal) partition of a multicore server processor into small servers.

server sprawl, increasing server utilization, and reducing power/cooling expenses [20]. Virtualization has been considered as a catalyst of IT modernization, which helps IT behave as a cloud computing and service provider, and business as a consumer [6]. Virtualization has been developed into a key technology in cloud computing because of its fundamental advantages such as increased computing resource utilization and improved system manageability. Virtualization is also a key aspect that makes a cloud computing infrastructure fundamentally different from a grid computing infrastructure.

Figure 1 illustrates an m-core server processor divided into n virtual machines VM_1, VM_2, ..., VM_n with $m_1, m_2, \ldots, m_n$ cores and core speeds $s_1, s_2, \ldots, s_n$ respectively. Figure 2 illustrates an m-core server processor shared by n virtual machines VM_1, VM_2, ..., VM_n with a time interval of length τ being divided into subintervals of lengths $\tau_1, \tau_2, \ldots, \tau_n$, and core speeds $s_1, s_2, \ldots, s_n$ respectively.

Due to increased popularity of cloud computing, optimal server partitioning and performance analysis of VMs have been important research problems with significant practical impact. In [17], the problem of *optimal spatial partitioning of a multicore server processor* in a cloud computing environment is considered, i.e., optimal virtual server configuration for some given types of applications [8], [11], [23], [24]. 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. Performance analysis of VMs has been investigated by a number researchers [12]–[14], [19]. However, the problem of *optimal temporal partitioning of a multicore server processor* has not been well studied. The problem involves optimal virtual machine configuration for several end users with different service requirements, such that the overall system performance is optimized without exceeding certain energy consumption budget. Such optimization is significant in cloud computing to provide the best quality of service and customer satisfaction, especially for users sharing the same resources.

B. CONTRIBUTIONS

The contributions of the present paper are three fold.

- First, we develop analytical models for virtual machines, i.e., partially available multi-server systems. A Markov model is obtained for a multi-server system with randomized availability. An accurate model is obtained for a multi-server system with deterministic availability. Such models will be extremely useful in many other studies in cloud computing.
- Second, we formulate the problem of optimal temporal partitioning of a multicore server processor for virtual machine allocation, such that the overall performance (i.e., the average task response time) of a group of virtual machines is optimized. We develop an algorithm to solve the problem numerically.
- Third, we formulate and solve the problem of optimal temporal partitioning of a multicore server processor with power consumption constraint, such that the overall performance of a group of virtual machines is optimized and that the total power consumption of the virtual machines does not exceed certain available power.

A virtual machine is treated as a queuing system with multiple servers, i.e., an M/M/m queuing system. 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 optimal temporal partitioning problems are formulated as multivariable optimization problems and solved algorithmically and numerically. To the best of our knowledge, such analytical investigation of optimal temporal partitioning of a multicore server processor for virtual machine allocation has not been seen in the literature.

The rest of the paper is organized as follows. In Section 2, we present our server and power consumption models. In Section 3, we develop analytical models for virtual machines, i.e., partially available multi-server systems. In Section 4, we formulate and solve the problem of optimal temporal partitioning of a multicore server processor for virtual

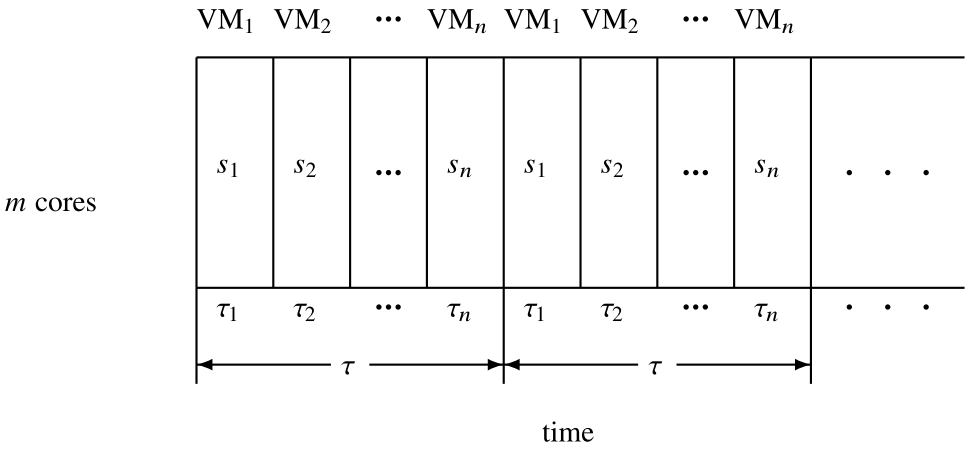

FIGURE 2. A temporal (vertical) partition of a multicore server processor into virtual servers.

machine allocation. In Section 5, we formulate and solve the problem of optimal temporal partitioning of a multicore server processor with power consumption constraint. We also present numerical examples to demonstrate our methods. In Section 6, we conclude the paper.

II. THE SERVER AND POWER MODELS

A. THE SERVER 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 queuing system which is elaborated as follows [16]. 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) queuing 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 ([15, p. 102])

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

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 queuing (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

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

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,

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

By 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.

B. THE POWER MODEL

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 $P = aCV^2f$, where a is an activity factor, C is the loading capacitance, V is the supply voltage, and f is the clock frequency [9]. Since $s \propto f$, where s is the processor speed, and $f \propto V^\phi$ with $0 < \phi \leq 1$ [25], 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 $P = m(s^\alpha + P^*)$.

III. PARTIALLY AVAILABLE MULTI-SERVER SYSTEMS

In a partially available multi-server system, the time interval is divided into alternate subintervals $A_1, U_1, A_2, U_2, A_3, U_3, \ldots$. During subintervals $A_1, A_2, A_3, \ldots$, the server system is available for service. During subintervals $U_1, U_2, U_3, \ldots$, the server system is not available for service; however, the system still accepts arrival tasks and puts the tasks into a waiting queue. When the A_j's and the U_j's are random variables, a server system has variable and randomized availability. When the A_j's and the U_j's are known values, a server system has fixed and deterministic availability.

A. RANDOMIZED AVAILABILITY

Assume that the arrival tasks form a Poisson stream of intensity λ, i.e., the interarrival times are i.i.d. exponential random variables with mean $1/\lambda$. Task service times are i.i.d. exponential random variables with mean $1/\mu$. The lengths of the subintervals $A_1, A_2, A_3, \ldots$ are i.i.d. exponential random variables with mean $1/\zeta$. The lengths of the subintervals $U_1, U_2, U_3, \ldots$ are i.i.d. exponential random variables with mean $1/\eta$.

A partially available m-server queuing system with the above assumptions can be modeled by a Markov chain, which contains states $0, 2, 4, 6, \ldots, 2i, \ldots$, and $1, 3, 5, 6, \ldots, 2i+1, \ldots$, where state $2i$, $i \geq 0$, means that there are i tasks in the queueing system and the m servers are available, while state $2i+1$, $i \geq 0$, means that there are i tasks in the queueing system and the m servers are not available. Figure 3 shows the state-transition-rate diagram for the first $2(K+1)$ states.

Let p_i denote the probability that a partially available m-server system is in state i, where $i \geq 0$. Unfortunately, there is no closed-form expression of p_i. To obtain a numerical solution, we can use the first $2(K+1)$ states as an approximation, where K is sufficiently large, so that the numerical solution is accurate enough. The p_i's, $0 \leq i \leq 2K+1$, can be obtained from the following system of linear equations:

$$\begin{aligned}
(\lambda + \zeta)p_0 &= \eta p_1 + \mu p_2; \\
(\lambda + i\mu + \zeta)p_{2i} &= \lambda p_{2i-2} + \eta p_{2i+1} + (i+1)\mu p_{2i+2}, \\
&\qquad 1 \leq i \leq m-1; \\
(\lambda + m\mu + \zeta)p_{2i} &= \lambda p_{2i-2} + \eta p_{2i+1} + m\mu p_{2i+2}, \\
&\qquad m \leq i \leq K-1; \\
(m\mu + \zeta)p_{2K} &= \lambda p_{2K-2} + \eta p_{2K+1}; \\
(\lambda + \eta)p_1 &= \zeta p_0;
\end{aligned}$$

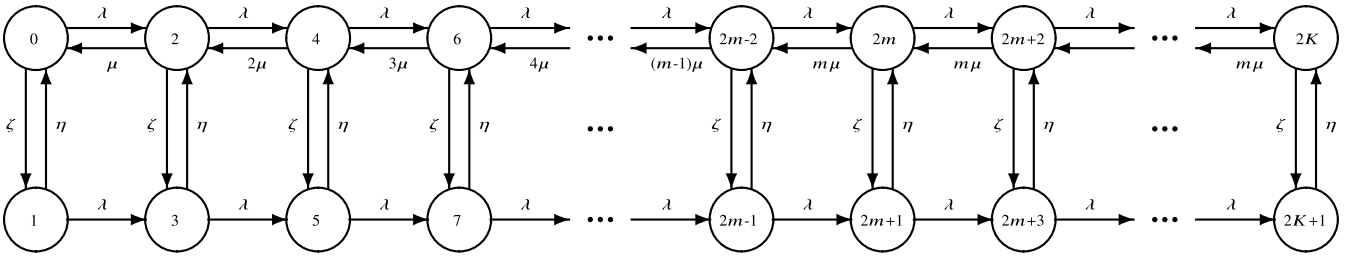

FIGURE 3. A Markov chain for a partially available multi-server.

$$(\lambda+\eta)p_{2i+1} = \lambda p_{2i-1} + \zeta p_{2i}, \quad 1 \le i \le K-1;$$
$$\eta p_{2K+1} = \lambda p_{2K-1} + \zeta p_{2K};$$
$$\sum_{i=0}^{2K+1} p_i = 1.$$

The above system of linear equations can be solved by using any available algorithm.

The average availability of the server system is

$$\frac{1/\zeta}{1/\zeta + 1/\eta} = \frac{\eta}{\zeta+\eta},$$

which is also the maximum server utilization. This implies that

$$\rho = \frac{\lambda}{m\mu} < \frac{\eta}{\zeta+\eta},$$

and

$$\lambda < \left(\frac{\eta}{\zeta+\eta}\right) m\mu.$$

The average number of tasks in the queueing system is

$$N = \sum_{i=0}^{K} i(p_{2i} + p_{2i+1}).$$

By Little's result, the average task response time is

$$T = \frac{N}{\lambda}.$$

Unfortunately, there is no closed-form expression of T.

The above analytical result can be easily validated by simulation. In Table 1, we compare the simulation results and analytical data on the average task response time of a partially available server system, where the parameters are set as $m = 7$, $\mu = 1$, $\zeta = \eta = b$ with $b = 1, 10, 100$, and $\lambda = z\eta/(\zeta+\eta)m\mu$ with $z = 0.50, 0.55, 0.60, \ldots, 0.95$. For each combination of b and λ, we calculate the analytical datum by using the above procedure, where $K = 600$. The simulation result is obtained by simulating a partially available server system for 1,000,000 tasks, recording the response time of each task, and reporting the average task response time. The maximum 99% confidence interval of the simulation results is ±0.256%. We have two important observations. First, our analytical data are very close to the simulation results, which confirms the correctness of our analysis. Second, as b increases, a partially available server system becomes an M/M/m queueing system with service rate $(\eta/(\zeta+\eta))\mu$, whose average task response time is also given in the table.

TABLE 1. Comparison of simulation and analytical results (exponential distribution).

λ	Simulation	Analytical	M/M/m
$b = 1$			
1.750	2.8051094	2.8064316	2.0435420
1.925	2.9438625	2.9405445	2.0730830
2.100	3.1413434	3.1207221	2.1178954
2.275	3.3793388	3.3655376	2.1851268
2.450	3.7337659	3.7060195	2.2864178
2.625	4.2133340	4.1981358	2.4424327
2.800	4.8510105	4.9539804	2.6941966
2.975	6.1771680	6.2353149	3.1371127
3.150	8.5022658	8.8277202	4.0570208
3.325	16.0107848	16.6570775	6.8825840
$b = 10$			
1.750	2.1054689	2.1092990	2.0435420
1.925	2.1472942	2.1472890	2.0730830
2.100	2.2014389	2.2037689	2.1178954
2.275	2.2760945	2.2870707	2.1851268
2.450	2.4083747	2.4107950	2.2864178
2.625	2.5891870	2.5991420	2.4424327
2.800	2.8847404	2.9002389	2.6941966
2.975	3.4368166	3.4261124	3.1371127
3.150	4.4550110	4.5125608	4.0570208
3.325	7.4762543	7.8382076	6.8825840
$b = 100$			
1.750	2.0460469	2.0499726	2.0435420
1.925	2.0772322	2.0803138	2.0730830
2.100	2.1258739	2.1262459	2.1178954
2.275	2.1970549	2.1950366	2.1851268
2.450	2.2914388	2.2985240	2.2864178
2.625	2.4487758	2.4577268	2.4424327
2.800	2.6934643	2.7143811	2.6941966
2.975	3.1298577	3.1655530	3.1371127
3.150	4.0271629	4.1020783	4.0570208
3.325	6.5987589	6.9776162	6.8825840

B. DETERMINISTIC AVAILABILITY

As we have already known, if $\zeta, \eta \to \infty$ and $\eta/(\zeta+\eta)$ is a fixed constant, then a partially available server system becomes an M/M/m queueing system with service rate $(\eta/(\zeta+\eta))\mu$. In fact, if the A_j's and the U_j's have uniform distributions, a partially available server system behaves

TABLE 2. Comparison of simulation and analytical results (uniform distribution).

λ	Simulation	M/M/m
$\delta = 0.9$		
1.750	2.0799412	2.0435420
1.925	2.1156512	2.0730830
2.100	2.1563804	2.1178954
2.275	2.2369541	2.1851268
2.450	2.3550578	2.2864178
2.625	2.5359346	2.4424327
2.800	2.8339722	2.6941966
2.975	3.3319350	3.1371127
3.150	4.3264528	4.0570208
3.325	7.8260679	6.8825840
$\delta = 0.5$		
1.750	2.0567588	2.0435420
1.925	2.0876383	2.0730830
2.100	2.1383291	2.1178954
2.275	2.2052814	2.1851268
2.450	2.3119378	2.2864178
2.625	2.4730523	2.4424327
2.800	2.7184047	2.6941966
2.975	3.2357289	3.1371127
3.150	4.2755368	4.0570208
3.325	7.3483642	6.8825840
$\delta = 0.1$		
1.750	2.0454333	2.0435420
1.925	2.0741896	2.0730830
2.100	2.1152680	2.1178954
2.275	2.1906800	2.1851268
2.450	2.2868527	2.2864178
2.625	2.4487524	2.4424327
2.800	2.7092054	2.6941966
2.975	3.1910062	3.1371127
3.150	4.1527899	4.0570208
3.325	6.8467575	6.8825840

TABLE 3. Comparison of simulation and analytical results (deterministic availability).

λ	Simulation	M/M/m
$b = 1$		
1.750	2.0880713	2.0435420
1.925	2.1203512	2.0730830
2.100	2.1715738	2.1178954
2.275	2.2406319	2.1851268
2.450	2.3403652	2.2864178
2.625	2.5005713	2.4424327
2.800	2.7602904	2.6941966
2.975	3.1774348	3.1371127
3.150	4.1100304	4.0570208
3.325	7.0243599	6.8825840
$b = 10$		
1.750	2.0387074	2.0435420
1.925	2.0691853	2.0730830
2.100	2.1201014	2.1178954
2.275	2.1880840	2.1851268
2.450	2.2955224	2.2864178
2.625	2.4355425	2.4424327
2.800	2.6948706	2.6941966
2.975	3.1167363	3.1371127
3.150	4.1956368	4.0570208
3.325	7.0118653	6.8825840
$b = 100$		
1.750	2.0452136	2.0435420
1.925	2.0687288	2.0730830
2.100	2.1168513	2.1178954
2.275	2.1834663	2.1851268
2.450	2.2915307	2.2864178
2.625	2.4379384	2.4424327
2.800	2.6832632	2.6941966
2.975	3.1317989	3.1371127
3.150	4.0239272	4.0570208
3.325	6.4827814	6.8825840

close to an M/M/m queueing system even when b is small. In Table 2, we compare the simulation results and analytical data on the average task response time of a partially available server system, where the parameters are set as $m = 7$, $\mu = 1$, $\zeta = \eta = b = 5$, and $\lambda = z\eta/(\zeta + \eta)m\mu$ with $z = 0.50, 0.55, 0.60, \ldots, 0.95$. The A_j's have a uniform distribution in the range $[(1-\delta)/\zeta, (1+\delta)/\zeta]$, and the U_j's have a uniform distribution in the range $[(1-\delta)/\eta, (1+\delta)/\eta]$, where $\delta = 0.9, 0.5, 0.1$. The maximum 99% confidence interval of the simulation results is $\pm 0.254\%$. It is observed that a partially available server system has less average response time and behaves closer to an M/M/m queueing system, if the A_j's and the U_j's have smaller variations.

The best case is when $A_j = 1/\zeta$ and $U_j = 1/\eta$ for all $j \geq 1$, i.e., a partially available server system has deterministic availability. In Table 3, we compare the simulation results and analytical data on the average task response time of a partially available server system, where the parameters are set as $m = 7$, $\mu = 1$, $\zeta = \eta = b$ with $b = 1, 10, 100$, and $\lambda = z\eta/(\zeta + \eta)m\mu$ with $z = 0.50, 0.55, 0.60, \ldots, 0.95$. The maximum 99% confidence interval of the simulation results is $\pm 0.254\%$. It is observed that no matter how big or small b is, a partially available server system behaves very close to an M/M/m queueing system.

Our main conclusion in modeling a partially available server system can be summarized as follows. Assume that an m-core processor or any physical machine with m processors is shared by n virtual machines $\mathrm{VM}_1, \mathrm{VM}_2, \ldots, \mathrm{VM}_n$ in such a way that during every τ time, VM_i is allocated τ_i time, where $\tau_1 + \tau_2 + \cdots + \tau_n = \tau$. If VM_i has a Poisson stream of arrival tasks, and the physical machine has exponential service time with service rate μ_i in the τ_i time allocated to VM_i, then VM_i can be treated as an M/M/m server with service rate $(\tau_i/\tau)\mu_i$.

IV. OPTIMAL TEMPORAL PARTITIONING

A. PROBLEM DEFINITION

Assume that we have a multicore server processor with m cores. There are n types of applications. The tasks of the ith type form a Poisson stream with arrival rate λ_i, where $1 \leq i \leq n$. Let $\lambda = \lambda_1 + \lambda_2 + \cdots + \lambda_n$. The task execution requirements of the ith type are i.i.d. exponential random variables with mean $\bar{r}_i$, There are n virtual machines VM_1, $\mathrm{VM}_2, \ldots, \mathrm{VM}_n$, where VM_i is used to process requests of the ith type of applications. We divide a time interval of length τ into n subintervals $\tau_1, \tau_2, \ldots, \tau_n$, such that τ_i is allocated to VM_i, where $1 \leq i \leq n$, and $\tau_1 + \tau_2 + \cdots + \tau_n = \tau$.

Based on our discussion in the last section, VM_i is treated as an M/M/m server with effective service rate $\mu_i' = (\tau_i/\tau)\mu_i$, where $\mu_i = s_i/\bar{r}_i$, and s_i is the execution speed of the m cores when they are allocated to VM_i. The average task response time of VM_i is

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

where $s_i' = (\tau_i/\tau)s_i$ is the effective execution speed of VM_i, and ρ_i is the utilization of VM_i given by

$$\rho_i = \frac{\lambda_i}{m\mu_i'} = \frac{\tau}{\tau_i}\cdot\frac{\lambda_i}{m\mu_i}. = \frac{\tau}{\tau_i}\cdot\frac{\lambda_i\bar{r}_i}{ms_i} = \frac{\lambda_i\bar{r}_i}{ms_i'}.$$

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

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

We will view T as a function of $\tau_1, \tau_2, \ldots, \tau_n$, represented by $T(\tau_1, \tau_2, \ldots, \tau_n)$.

Our *optimal temporal multicore server processor partitioning* problem can be formally defined as a multivariable optimization problem. 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 core speeds $s_1, s_2, \ldots, s_n$, and the length of a time interval τ, the problem is to find lengths of subintervals $\tau_1, \tau_2, \ldots, \tau_n$, such that $T(\tau_1, \tau_2, \ldots, \tau_n)$ is minimized subject to the constraint that

$$J(\tau_1, \tau_2, \ldots, \tau_n) = \tau,$$

where

$$J(\tau_1, \tau_2, \ldots, \tau_n) = \tau_1 + \tau_2 + \cdots + \tau_n.$$

B. THE METHOD

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

$$\nabla T(\tau_1, \tau_2, \ldots, \tau_n) = \phi\nabla J(\tau_1, \tau_2, \ldots, \tau_n),$$

that is,

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

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

Let us rewrite T_i as

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

where

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

It is clear that

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

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

We rewrite F_i as

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

where

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

Notice that

$$\frac{\partial G_i}{\partial \rho_i} = m\left(\frac{e^{\rho_i}}{e\rho_i}\right)^{m-1}\frac{1}{e}\cdot\frac{e^{\rho_i}\rho_i - e^{\rho_i}}{\rho_i^2} = m\left(\frac{e^{\rho_i-1}}{\rho_i}\right)^{m-1}e^{\rho_i-1}\cdot\frac{\rho_i - 1}{\rho_i^2} = me^{m(\rho_i-1)}\cdot\frac{\rho_i - 1}{\rho_i^{m+1}}.$$

Therefore, we have

$$\frac{\partial F_i}{\partial \rho_i} = -F_i^2\left(\sqrt{2\pi}m^{3/2}\left(-2(1-\rho_i)G_i + (1-\rho_i)^2\frac{\partial G_i}{\partial \rho_i}\right) - m\right) = F_i^2\left(\sqrt{2\pi}m^{3/2}\left(2(1-\rho_i)G_i + me^{m(\rho_i-1)}\cdot\frac{(1-\rho_i)^3}{\rho_i^{m+1}}\right) + m\right).$$

Since

$$\frac{\partial \rho_i}{\partial \tau_i} = -\frac{\tau}{\tau_i^2}\cdot\frac{\lambda_i\bar{r}_i}{ms_i} = -\frac{\rho_i}{\tau_i},$$

we get

$$\frac{\partial F_i}{\partial \tau_i} = \frac{\partial F_i}{\partial \rho_i}\cdot\frac{\partial \rho_i}{\partial \tau_i} = -\frac{\rho_i}{\tau_i}\cdot\frac{\partial F_i}{\partial \rho_i} = -\frac{\rho_i}{\tau_i}F_i^2\left(\sqrt{2\pi}m^{3/2}\left(2(1-\rho_i)G_i + me^{m(\rho_i-1)}\cdot\frac{(1-\rho_i)^3}{\rho_i^{m+1}}\right) + m\right),$$

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

Summarizing the above discussion, we get

$$\frac{\partial T}{\partial \tau_i} = -\tau\cdot\frac{\lambda_i}{\lambda}\cdot\frac{\bar{r}_i}{s_i}\left(\frac{1+F_i}{\tau_i^2} + \frac{\rho_i}{\tau_i^2}F_i^2\left(\sqrt{2\pi}m^{3/2}\left(2(1-\rho_i)G_i + me^{m(\rho_i-1)}\cdot\frac{(1-\rho_i)^3}{\rho_i^{m+1}}\right) + m\right)\right) = \phi,$$

for all $1 \le i \le n$. By observing that

$$\frac{\tau}{\tau_i}\cdot\frac{\lambda_i\bar{r}_i}{s_i} = m\rho_i,$$

we have

$$\frac{\partial T}{\partial \tau_i} = -\frac{m}{\lambda} \cdot \frac{\rho_i}{\tau_i}\left(1 + F_i + \rho_i F_i^2\left(\sqrt{2\pi} m^{3/2}\left(2(1-\rho_i)G_i + me^{m(\rho_i-1)} \cdot \frac{(1-\rho_i)^3}{\rho_i^{m+1}}\right) + m\right)\right) = \phi,$$

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

C. AN ALGORITHM

It is clear that there is no closed-form solution to $\tau_1, \tau_2, \ldots, \tau_n$ and ϕ. Instead, we will describe an algorithm to find numerical solutions.

We observe that $\partial T/\partial \tau_i < 0$ is an increasing function of τ_i. Hence, for a given ϕ, we can find $\tau_i(\phi)$ which satisfies $\partial T/\partial \tau_i = \phi$ by using the classic bisection method to search for $\tau_i(\phi)$ in an interval (lb, ub). Based on the condition that $\rho_i < 1$, we can set $lb = (\lambda_i \bar{r}_i/ms_i)\tau$. The ub can be τ. Furthermore, we notice that

$$J(\phi) = \sum_{i=1}^{n} \tau_i(\phi)$$

is an increasing function of ϕ. Therefore, $\phi < 0$ that satisfies $J(\phi) = \tau$ can also be found by the bisection method to search for ϕ in an interval (lb, ub), where lb is sufficiently small such that $J(\phi) < \tau$, and $ub = 0$.

D. A NUMERICAL EXAMPLE

Notice that since

$$\rho_i = \frac{\tau}{\tau_i} \cdot \frac{\lambda_i \bar{r}_i}{ms_i} < 1,$$

we have

$$\frac{\lambda_i \bar{r}_i}{s_i} < \frac{\tau_i}{\tau} m,$$

which gives rise to

$$\sum_{i=1}^{n} \frac{\lambda_i \bar{r}_i}{s_i} < m.$$

Our input parameters must satisfy the above condition.

We consider a physical server with $m = 64$ cores to be temporally partitioned into $n = 8$ virtual machines. The task arrival rates are $\lambda_i = ((i+5)/76)\lambda$, for all $1 \le i \le n$. The mean task execution requirements are $\bar{r}_i = 1$, and the core speeds are $s_i = 1$, for all $1 \le i \le n$. The length of a time interval is $\tau = 1$. In Table 4, for each $\lambda = 12, 24, 36, 48, 60$, we display $\lambda_i, s_i, \tau_i, \rho_i, T_i$, 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^{-13}.

We have the following observations.

- τ_i is determined in such a way that a virtual machine VM_i with larger service requirement is allocated a larger τ_i.
- The n virtual machines have different utilization. A virtual machine with heavy service requirement has higher utilization than a virtual machine with light service requirement.
- The n types of applications have different average task response times. A type of application with heavy service requirement has shorter average task response time than a type of application with light service requirement.

V. OPTIMAL SERVER SPEEDS

We will consider two types of power consumption and core speed models. In the *idle-speed model*, a core runs at zero speed when there is no task to perform. Since the power of a core with speed s is s^α, the average amount of energy consumed by an m-core VM_i in one unit of time, i.e., the average power supply P_i to VM_i, is

$$\frac{\tau_i}{\tau} m \rho_i s_i^\alpha,$$

where τ_i/τ is the actual percentage of time a physical machine is allocated to VM_i, and ρ_i is the utilization of VM_i. Since a processor core still consumes some amount of power P^* even when it is idle, we will include P^* in P_i, i.e.,

$$P_i = \frac{\tau_i}{\tau} m(\rho_i s_i^\alpha + P^*) = \lambda_i \bar{r}_i s_i^{\alpha-1} + \frac{\tau_i}{\tau} m P^*,$$

for all $1 \le i \le n$. In the *constant-speed model*, all cores of VM_i run at the speed s_i even if there is no task to perform. Therefore, the power allocated to VM_i is

$$P_i = \frac{\tau_i}{\tau} m(s_i^\alpha + P^*),$$

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

A. PROBLEM DEFINITION

In this section, we will allow the virtual machines to set their execution speeds, such that the overall performance of a group of virtual machines is optimized and that the total power consumption of the virtual machines does not exceed certain available power. The average task response time of all the n types of applications is viewed as a function of $\tau_1, \tau_2, \ldots, \tau_n$ and $s_1, s_2, \ldots, s_n$, represented by $T(\tau_1, \tau_2, \ldots, \tau_n, s_1, s_2, \ldots, s_n)$.

Our *optimal temporal multicore server processor partitioning with power constraint* problem can be formally defined as a multivariable optimization problem. 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 length of a time interval τ, the base power consumption P^*, and the total available power P, the problem is to find lengths of subintervals $\tau_1, \tau_2, \ldots, \tau_n$ and the core speeds $s_1, s_2, \ldots, s_n$ of the virtual machines, such that $T(\tau_1, \tau_2, \ldots, \tau_n, s_1, s_2, \ldots, s_n)$ is minimized subject to the constraints that

$$J(\tau_1, \tau_2, \ldots, \tau_n) = \tau,$$

where

$$J(\tau_1, \tau_2, \ldots, \tau_n) = \tau_1 + \tau_2 + \cdots + \tau_n,$$

TABLE 4. Numerical data for optimal temporal partitioning.

i	λ_i	s_i	τ_i	ρ_i	T_i
$\lambda = 12.0, T = 7.8801279$					
1	0.9473684	1.0000000	0.1000927	0.1478893	9.9907434
2	1.1052632	1.0000000	0.1081124	0.1597387	9.2496311
3	1.2631579	1.0000000	0.1155770	0.1707678	8.6522376
4	1.4210526	1.0000000	0.1225880	0.1811267	8.1574078
5	1.5789474	1.0000000	0.1292191	0.1909243	7.7387966
6	1.7368421	1.0000000	0.1355261	0.2002431	7.3786530
7	1.8947368	1.0000000	0.1415524	0.2091470	7.0645224
8	2.0526316	1.0000000	0.1473324	0.2176871	6.7873736
$\lambda = 24.0, T = 7.8801279$					
1	1.8947368	1.0000000	0.1000927	0.2957786	9.9907434
2	2.2105263	1.0000000	0.1081124	0.3194774	9.2496311
3	2.5263158	1.0000000	0.1155770	0.3415357	8.6522376
4	2.8421053	1.0000000	0.1225880	0.3622533	8.1574078
5	3.1578947	1.0000000	0.1292191	0.3818485	7.7387966
6	3.4736842	1.0000000	0.1355261	0.4004861	7.3786530
7	3.7894737	1.0000000	0.1415524	0.4182941	7.0645224
8	4.1052632	1.0000000	0.1473324	0.4353743	6.7873736
$\lambda = 36.0, T = 7.8801965$					
1	2.8421053	1.0000000	0.1000807	0.4437208	9.9919343
2	3.3157895	1.0000000	0.1080995	0.4792732	9.2507332
3	3.7894737	1.0000000	0.1155633	0.5123643	8.6532647
4	4.2631579	1.0000000	0.1225738	0.5434429	8.1583546
5	4.7368421	1.0000000	0.1292057	0.5728318	7.7396022
6	5.2105263	1.0000000	0.1355183	0.6007635	7.3791059
7	5.6842105	1.0000000	0.1415636	0.6273915	7.0640631
8	6.1578947	1.0000000	0.1473950	0.6527841	6.7847650
$\lambda = 48.0, T = 7.9088955$					
1	3.7894737	1.0000000	0.0972162	0.6090602	10.2864130
2	4.4210526	1.0000000	0.1050643	0.6574923	9.5184469
3	5.0526316	1.0000000	0.1125770	0.7012746	8.8849378
4	5.6842105	1.0000000	0.1201491	0.7392130	8.3296476
5	6.3157895	1.0000000	0.1281244	0.7702216	7.8200097
6	6.9473684	1.0000000	0.1366053	0.7946446	7.3471920
7	7.5789474	1.0000000	0.1455165	0.8137979	6.9124149
8	8.2105263	1.0000000	0.1547472	0.8290262	6.5164564
$\lambda = 60.0, T = 8.7908889$					
1	4.7368421	1.0000000	0.0805510	0.9188360	13.2021899
2	5.5263158	1.0000000	0.0932632	0.9258601	11.5325594
3	6.3157895	1.0000000	0.1059729	0.9312213	10.2557137
4	7.1052632	1.0000000	0.1186752	0.9354920	9.2471130
5	7.8947368	1.0000000	0.1313685	0.9390021	8.4297947
6	8.6842105	1.0000000	0.1440521	0.9419566	7.7536826
7	9.4736842	1.0000000	0.1567261	0.9444904	7.1848005
8	10.2631579	1.0000000	0.1693910	0.9466963	6.6992855

and that

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

where

$$\begin{aligned} K(s_1, s_2, \ldots, s_n) &= \sum_{i=1}^{n} \left(\lambda_i \bar{r}_i s_i^{\alpha-1} + \frac{\tau_i}{\tau} m P^* \right) \\ &= \sum_{i=1}^{n} \lambda_i \bar{r}_i s_i^{\alpha-1} + m P^*, \end{aligned}$$

for the idle-speed model, or

$$K(\tau_1, \tau_2, \ldots, \tau_n, s_1, s_2, \ldots, s_n) = P,$$

where

$$\begin{aligned} K(\tau_1, \tau_2, \ldots, \tau_n, s_1, s_2, \ldots, s_n) &= \sum_{i=1}^{n} \frac{\tau_i}{\tau} m (s_i^{\alpha} + P^*) \\ &= m \sum_{i=1}^{n} \frac{\tau_i}{\tau} s_i^{\alpha} + m P^*, \end{aligned}$$

for the constant-speed model.

B. THE METHOD

We can minimize T by using the method of Lagrange multiplier.

For the idle-speed model, we have

$$\nabla T(\tau_1, \tau_2, \ldots, \tau_n, s_1, s_2, \ldots, s_n) = \phi \nabla J(\tau_1, \tau_2, \ldots, \tau_n) + \psi \nabla K(s_1, s_2, \ldots, s_n),$$

that is,

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

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

For the constant-speed model, we have

$$\nabla T(\tau_1, \tau_2, \ldots, \tau_n, s_1, s_2, \ldots, s_n) = \phi \nabla J(\tau_1, \tau_2, \ldots, \tau_n) + \psi \nabla K(\tau_1, \tau_2, \ldots, \tau_n, s_1, s_2, \ldots, s_n),$$

that is,

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

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

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

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

We have already obtained $\partial T / \partial \tau_i$ in Section 4.2, i.e.,

$$\frac{\partial T}{\partial \tau_i} = -\frac{m}{\lambda} \cdot \frac{\rho_i}{\tau_i} \left(1 + F_i + \rho_i \frac{\partial F_i}{\partial \rho_i} \right),$$

for all $1 \leq i \leq n$. As for $\partial T / \partial s_i$, let us rewrite T_i as

$$T_i = \frac{m \rho_i}{\lambda_i} (1 + F_i).$$

Since

$$\frac{\partial \rho}{\partial s_i} = -\frac{\tau}{\tau_i} \cdot \frac{\lambda_i \bar{r}_i}{m s_i^2} = -\frac{\rho_i}{s_i},$$

we have

$$\begin{aligned} \frac{\partial T}{\partial s_i} &= \frac{\lambda_i}{\lambda} \cdot \frac{\partial T_i}{\partial s_i} \\ &= \frac{m}{\lambda} \left(\frac{\partial \rho_i}{\partial s_i} (1 + F_i) + \rho_i \frac{\partial F_i}{\partial \rho_i} \cdot \frac{\partial \rho_i}{\partial s_i} \right) \\ &= \frac{m}{\lambda} \cdot \frac{\partial \rho_i}{\partial s_i} \left(1 + F_i + \rho_i \frac{\partial F_i}{\partial \rho_i} \right) \\ &= -\frac{m}{\lambda} \cdot \frac{\rho_i}{s_i} \left(1 + F_i + \rho_i \frac{\partial F_i}{\partial \rho_i} \right), \end{aligned}$$

where

$$\frac{\partial F_i}{\partial \rho_i} = F_i^2 \left(\sqrt{2\pi} m^{3/2} \left(2(1 - \rho_i) G_i + m e^{m(\rho_i - 1)} \cdot \frac{(1 - \rho_i)^3}{\rho_i^{m+1}} \right) + m \right),$$

with

$$G_i = (e^{\rho_i} / e \rho_i)^m,$$

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

C. THE IDLE-SPEED MODEL

We have the following theorem for the idle-speed model.

Theorem 1: For the idle-speed model, we have

$$\tau_i = \left(\frac{\tau}{P - mP^*} \right) \lambda_i \bar{r}_i s_i^{\alpha - 1},$$

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

Proof: First, we have

$$\frac{\partial T}{\partial \tau_i} = -\frac{m}{\lambda} \cdot \frac{\rho_i}{\tau_i} \left(1 + F_i + \rho_i \frac{\partial F_i}{\partial \rho_i} \right) = \phi,$$

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

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

for all $1 \leq i \leq n$. The above two equations imply that

$$\tau_i = \frac{\psi}{\phi} (\alpha - 1) \lambda_i \bar{r}_i s_i^{\alpha - 1},$$

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

$$\tau = \sum_{i=1}^{n} \tau_i = \frac{\psi}{\phi} (\alpha - 1) \sum_{i=1}^{n} \lambda_i \bar{r}_i s_i^{\alpha - 1},$$

and

$$\sum_{i=1}^{n} \lambda_i \bar{r}_i s_i^{\alpha - 1} = P - mP^*,$$

we get

$$\tau = \frac{\psi}{\phi} (\alpha - 1)(P - mP^*),$$

and

$$\frac{\psi}{\phi} (\alpha - 1) = \frac{\tau}{P - mP^*}.$$

Consequently, we have

$$\tau_i = \left(\frac{\tau}{P - mP^*} \right) \lambda_i \bar{r}_i s_i^{\alpha - 1},$$

for all $1 \leq i \leq n$. ∎

The above theorem means that the average task response time is now a function of execution speeds $s_1, s_2, \ldots, s_n$, denoted by $T(s_1, s_2, \ldots, s_n)$. Our optimization problem is to find the core speeds $s_1, s_2, \ldots, s_n$ of the virtual servers, such that $T(s_1, s_2, \ldots, s_n)$ is minimized subject to the constraint that $K(s_1, s_2, \ldots, s_n) = P$.

To solve this problem, we consider

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

that is,

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

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

Notice that

$$\begin{aligned}\rho_i &= \frac{\tau}{m}\lambda_i\bar{r}_i \cdot \frac{1}{\tau_i s_i} \\ &= \frac{\tau}{m}\lambda_i\bar{r}_i\left(\left(\frac{\tau}{P-mP^*}\right)\lambda_i\bar{r}_i s_i^{\alpha}\right)^{-1} \\ &= \left(\frac{P}{m}-P^*\right)\frac{1}{s_i^{\alpha}},\end{aligned}$$

and

$$\frac{\partial\rho_i}{\partial s_i} = \left(\frac{P}{m}-P^*\right)(-\alpha)\frac{1}{s_i^{\alpha+1}} = -\alpha\frac{\rho_i}{s_i},$$

for all $1 \le i \le n$. Hence, we get

$$\begin{aligned}\frac{\partial T}{\partial s_i} &= \frac{m}{\lambda}\cdot\frac{\partial\rho_i}{\partial s_i}\left(1+F_i+\rho_i\frac{\partial F_i}{\partial\rho_i}\right) \\ &= -\alpha\frac{m}{\lambda}\cdot\frac{\rho_i}{s_i}\left(1+F_i+\rho_i\frac{\partial F_i}{\partial\rho_i}\right) \\ &= \xi\lambda_i\bar{r}_i(\alpha-1)s_i^{\alpha-2},\end{aligned}$$

that is,

$$-\frac{\alpha}{\alpha-1}\cdot\frac{m}{\lambda}\cdot\frac{\rho_i}{\lambda_i\bar{r}_i s_i^{\alpha-1}}\left(1+F_i+\rho_i\frac{\partial F_i}{\partial\rho_i}\right) = \xi,$$

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

1) AN ALGORITHM

It is clear that there is no closed-form solution to $s_1, s_2, \ldots, s_n$ and ξ. Instead, we will describe an algorithm to find numerical solutions.

We observe that the left-hand side of the last equation is an increasing function of s_i. Hence, for a given ξ, we can find $s_i(\xi)$ which satisfies the last equation by using the classic bisection method to search for $s_i(\xi)$ in an interval (lb, ub). Based on the condition that $\rho_i < 1$, we can set $lb = (P/m - P^*)^{1/\alpha}$. The ub can be a sufficiently large position number. Furthermore, we notice that

$$K(\xi) = \sum_{i=1}^{n}\lambda_i\bar{r}_i(s_i(\xi))^{\alpha-1} + mP^*$$

is an increasing function of ξ. Therefore, $\xi < 0$ that satisfies $K(\xi) = P$ can also be found by the bisection method to search for ξ in an interval (lb, ub), where lb is sufficiently small such that $K(\xi) < P$, and $ub = 0$.

D. THE CONSTANT-SPEED MODEL

We have the following theorem for the constant-speed model.

Theorem 2: For the constant-speed model, all virtual machines have the same execution speed, i.e., $s_1 = s_2 = \cdots = s_n = s$, where

$$s = \left(\frac{\phi}{\psi\alpha-1}\cdot\frac{\tau}{m}\right)^{1/\alpha},$$

and

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

Proof: First, we have

$$\frac{\partial T}{\partial\tau_i} = -\frac{m}{\lambda}\cdot\frac{\rho_i}{\tau_i}\left(1+F_i+\rho_i\frac{\partial F_i}{\partial\rho_i}\right) = \phi+\frac{m}{\tau}s_i^{\alpha},$$

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

$$\frac{\partial T}{\partial s_i} = -\frac{m}{\lambda}\cdot\frac{\rho_i}{s_i}\left(1+F_i+\rho_i\frac{\partial F_i}{\partial\rho_i}\right) = \psi\frac{\tau_i}{\tau}m\alpha s_i^{\alpha-1},$$

for all $1 \le i \le n$. The above two equations imply that

$$\phi+\frac{m}{\tau}s_i^{\alpha} = \psi\alpha\frac{m}{\tau}s_i^{\alpha},$$

that is

$$s_i = \left(\frac{\phi}{\psi\alpha-1}\cdot\frac{\tau}{m}\right)^{1/\alpha},$$

for all $1 \le i \le n$. By the constraint

$$\begin{aligned}K(\tau_1,\tau_2,\ldots,\tau_n,s_1,s_2,\ldots,s_n) &= \sum_{i=1}^{n}\frac{\tau_i}{\tau}m(s_i^{\alpha}+P^*) \\ &= m(s^{\alpha}+P^*) = P,\end{aligned}$$

we get

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

This proves the theorem. ∎

The above theorem means that for the constant-speed model, the optimal temporal multicore server processor partitioning with power constraint problem becomes the optimal temporal multicore server processor partitioning problem, and can be solved by using the method in Section 4.2 and the algorithm in Section 4.3.

E. A NUMERICAL EXAMPLE

Let us consider a physical server with $m = 64$ cores to be temporally partitioned into $n = 8$ virtual machines. The task arrival rates are $\lambda_i = ((i+5)/76)\lambda$, for all $1 \le i \le n$. The mean task execution requirements are $\bar{r}_i = 1$, for all $1 \le i \le n$. The length of a time interval is $\tau = 1$. The base power is $P^* = 2$ and the total power is $P = 192$. We set $\alpha = 3$. In Table 5, for the idle-speed model and for each $\lambda = 12, 24, 36, 48, 60$, we display $\lambda_i, s_i, \tau_i, \rho_i, T_i$, 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^{-13}.

We have the following observations.

- τ_i is determined in such a way that a virtual machine VM_i with larger service requirement is allocated a larger τ_i.
- However, s_i is determined in such a way that a virtual machine VM_i with larger service requirement is assigned a slower core speed s_i.
- The n virtual machines have different utilization. A virtual machine with heavy service requirement has higher utilization than a virtual machine with light service requirement.

TABLE 5. Numerical data for optimal server speeds.

i	λ_i	s_i	τ_i	ρ_i	T_i
$\lambda = 12.0, T = 3.4021363$					
1	0.9473684	2.5408652	0.0955657	0.0609614	4.1182833
2	1.1052632	2.4637252	0.1048263	0.0668688	3.8720192
3	1.2631579	2.3987991	0.1135705	0.0724467	3.6706310
4	1.4210526	2.3429519	0.1218869	0.0777517	3.5017063
5	1.5789474	2.2940975	0.1298409	0.0828256	3.3571962
6	1.7368421	2.2507815	0.1374824	0.0877001	3.2316152
7	1.8947368	2.2119516	0.1448506	0.0924002	3.1210751
8	2.0526316	2.1768234	0.1519768	0.0969461	3.0227301
$\lambda = 24.0, T = 4.8113473$					
1	1.8947368	1.7966630	0.0955657	0.1724250	5.8241321
2	2.2105263	1.7421168	0.1048263	0.1891334	5.4758621
3	2.5263158	1.6962071	0.1135705	0.2049101	5.1910561
4	2.8421053	1.6567171	0.1218869	0.2199150	4.9521605
5	3.1578947	1.6221719	0.1298409	0.2342661	4.7477923
6	3.4736842	1.5915428	0.1374824	0.2480533	4.5701941
7	3.7894737	1.5640860	0.1448506	0.2613474	4.4138667
8	4.1052632	1.5392466	0.1519768	0.2742050	4.2747860
$\lambda = 36.0, T = 5.8926730$					
1	2.8421053	1.4669691	0.0955657	0.3167649	7.1330764
2	3.3157895	1.4224324	0.1048263	0.3474603	6.7065344
3	3.7894737	1.3849473	0.1135705	0.3764439	6.3577197
4	4.2631579	1.3527039	0.1218869	0.4040097	6.0651336
5	4.7368421	1.3244978	0.1298409	0.4303743	5.8148347
6	5.2105263	1.2994893	0.1374824	0.4557030	5.5973221
7	5.6842105	1.2770708	0.1448506	0.4801258	5.4058607
8	6.1578947	1.2567897	0.1519769	0.5037466	5.2355208
$\lambda = 48.0, T = 6.8072381$					
1	3.7894737	1.2681767	0.0952266	0.4902987	8.2806000
2	4.4210526	1.2296763	0.1044545	0.5378084	7.7854183
3	5.0526316	1.1972823	0.1131699	0.5826534	7.3802897
4	5.6842105	1.1694803	0.1214720	0.6252033	7.0394156
5	6.3157895	1.1454106	0.1294703	0.6654515	6.7436874
6	6.9473684	1.1247815	0.1373336	0.7027413	6.4753657
7	7.5789474	1.1076890	0.1452997	0.7357775	6.2177147
8	8.2105263	1.0941142	0.1535735	0.7635053	5.9610026
$\lambda = 60.0, T = 8.0891572$					
1	4.7368421	1.0475527	0.0812196	0.8699060	12.0000369
2	5.5263158	1.0413014	0.0936286	0.8856674	10.5611207
3	6.3157895	1.0370097	0.1061239	0.8967090	9.4312000
4	7.1052632	1.0338501	0.1186630	0.9049555	8.5240070
5	7.8947368	1.0314060	0.1312251	0.9114043	7.7808777
6	8.6842105	1.0294449	0.1437992	0.9166228	7.1614699
7	9.4736842	1.0278270	0.1563792	0.9209583	6.6374225
8	10.2631579	1.0264626	0.1689613	0.9246356	6.1883235

- The n types of applications have different average task response times. A type of application with heavy service requirement has shorter average task response time than a type of application with light service requirement.

For the constant-speed model, we have $s = 1$ by Theorem 2. Thus, all the data are identical to those in Table 4. It is noticed that the idle-speed model results in higher core speeds and shorter average task response times than the constant-speed model.

VI. CONCLUDING REMARKS

We have emphasized the importance of studying the problem of optimal temporal partitioning of a multicore server processor for virtual machine allocation in cloud computing. For the first time in the literature, we have addressed the problem as multivariable optimization problems and solved them algorithmically and numerically. Our approach is based on a queueing model of a virtual machine, i.e., an accurate model for a multi-server system with deterministic availability. The model will be extremely useful in many other studies of virtual machines in cloud computing.

REFERENCES

[1] Accessed: Sep. 3, 2018. [Online]. Available: http://en.wikipedia.org/wiki/Cloud_computing
[2] Accessed: Sep. 3, 2018. [Online]. Available: http://en.wikipedia.org/wiki/CMOS

[3] Accessed: Sep. 3, 2018. [Online]. Available: http://en.wikipedia.org/wiki/Time-sharing
[4] Accessed: Sep. 3, 2018. [Online]. Available: http://www.computerworld.com/s/article/41632/Server_Partitioning
[5] I. Ali and N. Meghanathan, "Virtual machines and networks—Installation, performance, study, advantages and virtualization options," *Int. J. Netw. Secur. Appl.*, vol. 3, no. 1, pp. 1–15, 2011.
[6] T. J. Bittman, "Server virtualization: One path that leads to cloud computing," Gartner RAS Core Res. Note G00171730, Oct. 2009. Accessed: Sep. 30, 2018. [Online]. Available: http://www.acma.com/acma/Gartner,%20Server%20Virtualization%20Leads%20to%20Cloud%20Computing.pdf
[7] R. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exper.*, vol. 41, no. 1, pp. 23–50, 2011.
[8] S. Chaisiri, R. Kaewpuang, B.-S. Lee, and D. Niyato, "Cost minimization for provisioning virtual servers in Amazon elastic compute cloud," in *Proc. 19th IEEE Int. Symp. Modeling, Anal., Simulation Comput. Telecommun. Syst.*, Jul. 2011, pp. 85–95.
[9] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE J. Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, Apr. 1992.
[10] M.-S. Chen and K. G. Shin, "Processor allocation in an N-cube multiprocessor using Gray codes," *IEEE Trans. Comput.*, vol. C-100, no. 12, pp. 1396–1407, Dec. 1987.
[11] J. Dejun, G. Pierre, and C.-H. Chi, "Autonomous resource provisioning for multi-service Web applications," in *Proc. 19th Int. World-Wide Web Conf.*, 2010, pp. 471–480.
[12] R. Ghosh, K. S. Trivedi, V. K. Naik, and D. S. Kim, "End-to-end performability analysis for infrastructure-as-a-service cloud: An interacting stochastic models approach," in *Proc. 16th IEEE Pacific Rim Int. Symp. Dependable Comput.*, Dec. 2010, pp. 125–132.
[13] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, and D. H. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 6, pp. 931–945, Jun. 2011.
[14] H. Khazaei, J. Mišić, and V. B. Mišić, "A fine-grained performance model of cloud computing centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 11, pp. 2138–2147, Nov. 2013.
[15] K. Li, *Queueing Systems: Theory*, vol. 1. New York, NY, USA: Wiley, 1975.
[16] K. Li, "Optimal configuration of a multicore server processor for managing the power and performance tradeoff," *J. Supercomput.*, vol. 61, no. 1, pp. 189–214, 2012.
[17] K. Li, "Optimal partitioning of a multicore server processor," *J. Supercomput.*, vol. 71, no. 10, pp. 3744–3769, 2015.
[18] K. Li and K. H. Cheng, "Complexity of resource allocation and job scheduling problems in partitionable mesh connected systems," in *Proc. 1st IEEE Symp. Parallel Distrib. Process.*, Jun. 1989, pp. 358–365.
[19] F. Longo, R. Ghosh, V. K. Naik, and K. S. Trivedi, "A scalable availability model for infrastructure-as-a-service cloud," in *Proc. IEEE/IFIP 41st Int. Conf. Dependable Syst. Netw.*, Jun. 2011, pp. 335–346.
[20] M. R. Marty and M. D. Hill, "Virtual hierarchies to support server consolidation," in *Proc. 34th Int. Symp. Comput. Archit.*, 2007, pp. 46–56.
[21] R. A. Meyer and L. H. Seawright, "A virtual machine time-sharing system," *IBM Syst. J.*, vol. 9, no. 3, pp. 199–218, 1970.
[22] H. J. Siegel, *Interconnection Networks for Large-Scale Parallel Processing*. Lexington, MA, USA: Heath, 1985.
[23] B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal, and T. Wood, "Agile dynamic provisioning of multi-tier Internet applications," *ACM Trans. Auton. Adapt. Syst.*, vol. 3, no. 1, p. 1, 2008.
[24] D. Villela, P. Pradhan, and D. Rubenstein, "Provisioning servers in the application tier for e-commerce systems," *ACM Trans. Internet Technol.*, vol. 7, no. 1, p. 7, 2007.
[25] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and practical limits of dynamic voltage scaling," in *Proc. 41st Design Autom. Conf.*, 2004, pp. 868–873.

KEQIN LI (F'15) is a SUNY Distinguished Professor of computer science with the State University of New York. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU–GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of Things, and cyber-physical systems. He has published over 590 journal articles, book chapters, and refereed conference papers. He received several best paper awards. He is currently serving or has served on the editorial boards of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON SERVICES COMPUTING, and IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING.