Stochastic Models and Analysis for Resource Management in Server Farms

Thesis Proposal

Varun Gupta*

Thesis Committee:

David G. Andersen*
Anupam Gupta*
Mor Harchol-Balter* (chair)
Alan Scheller-Wolf†
Devavrat Shah‡
Don Towsley+

* Computer Science Department, Carnegie Mellon University
† Tepper School of Business, Carnegie Mellon University
‡ Department of Electrial Engineering and Computer Science, MIT
+ Department of Computer Science, University of Massachusetts (Amherst)

October 2010

Abstract

Server farms are popular architectures for computing infrastructures such as supercomputing centers, data centers and web server farms. As server farms become larger and their workloads more complex, designing efficient policies for managing the resources in server farms via trial-and-error becomes intractable. It is hard to predict the exact effect of various parameters on performance. Stochastic modeling and analysis techniques allow us to understand the performance of such complex systems and to guide design of policies to optimize the performance. However, most existing models of server farms are motivated by telephone networks, inventory management systems, and call centers. Modeling assumptions which hold for these problem domains are not accurate for computing server farms.

There are numerous *gaps* between traditional models of multi-server systems and how today's server farms operate. To cite a few: (*i*) Unlike call durations, supercomputing jobs and file sizes have high variance in service requirements and this critically affects the optimality and performance of scheduling policies. (*ii*) Most existing analysis of server farms focuses on the First-Come-First-Served (FCFS) scheduling discipline, while time sharing servers (e.g., web and database servers) are better modeled by the Processor-Sharing (PS) scheduling discipline. (*iii*) Time sharing systems typically exhibit thrashing (resource contention) which limits the achievable concurrency level, but traditional models of time sharing systems ignore this fundamental phenomenon. (*iv*) Recently, minimizing energy consumption has become an important metric in managing server farms. State-of-the-art servers come with multiple knobs to control energy consumption, but traditional queueing models don't take the metric of energy consumption or these control knobs into account.

In this thesis we attempt to bridge some of these gaps by bringing the stochastic modeling and analysis literature closer to the realities of today's compute server farms. We introduce new models for computing server farms, develop stochastic analysis techniques to evaluate their performance, and propose resource management policies to optimize their performance.

Contents

1 Introduction

1.1 Motivation

Server farms are becoming an increasingly popular architecture of computation since they allow combining low-cost commodity hardware to provide computing power exceeding that of any single device. In addition, the server farm architecture allows the design of fault-tolerant and scalable systems – failures of a few servers don't bring down the entire server farm, and adding capacity is as easy as adding more server. Server farms also lead to the design of energy-efficient computing systems by combining slower but more power-efficient processors to reduce the peak power consumption (for example, the IBM Blue Gene supercomputer [1] and the FAWN cluster architecture [5]). Another benefit of server farms is that by allowing the consolidation of multiple workloads, server farms lead to more efficient resource utilization, such as in cloud computing centers.

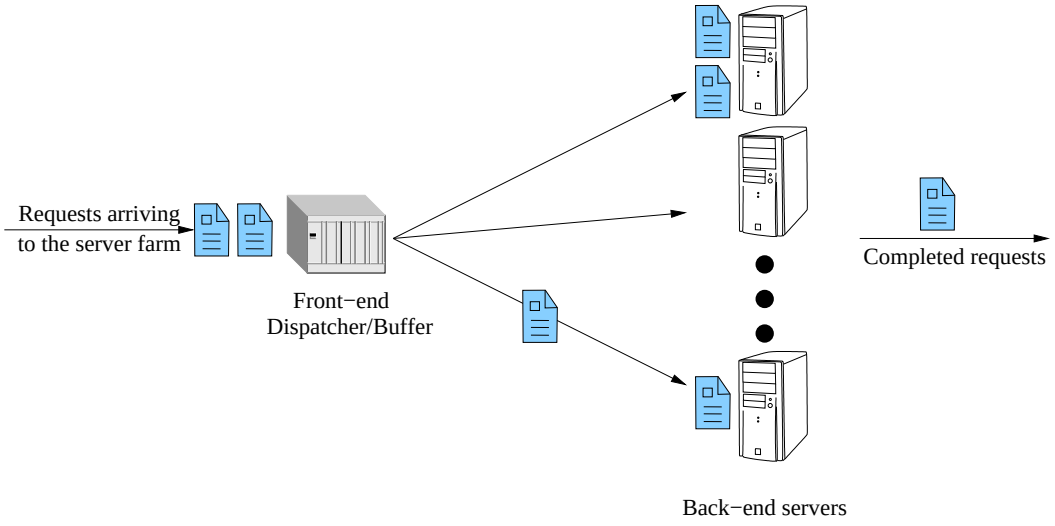


Figure 1: The server farm architecture.

Figure 1 shows the components of a simple server farm. New requests or jobs arrive at the front-end dispatcher, or load balancer, which routes the incoming request to one of the back-end servers. The back-end servers serve the requests dispatched to them. Requests leave the server farm once they complete processing at the back end servers.[1] Even in this simplified setup, there are two fundamental design questions:

Question 1: Dispatching policy: Which back-end server should process the incoming

[1]We present a subset of the request types. There are certainly more complicated scenarios. For example, requests that *fork* into multiple smaller requests and are completed once all the sub-requests are complete (*join*), or map-reduce type tasks which involve multiple of these *fork-join* stages. In this thesis we focus on the simple subset.

request? Should the dispatch be immediate, or can the front-end dispatcher defer the decision? Is the correct dispatching policy dynamic (depending on the current load of back-end servers), or static?

Question 2: Scheduling policy: How should the back-end servers schedule the tasks dispatched to them?

Today's server farms and data centers cater to a wide spectrum of workloads – from database queries, to computationally intensive jobs, to file streaming and web requests, to map-reduce tasks. Each of the aforementioned workloads imposes different constraints on which dispatching policies and scheduling policies are feasible. For example, a database query may only be dispatched to a server that stores the required data (locality) and hence dispatching policies are static, while a job only requiring CPU may be dispatched to any server allowing dynamic dispatching policies. Further, while a database query or a CPU intensive job may be queued for later processing, web and file download requests are latency-sensitive and must begin processing immediately to prevent dropped connections caused by time-outs. As another example, a supercomputing job runs alone on its back-end server in a non-preemptive fashion, whereas web and database requests typically timeshare their back-end server with other requests. It is clear that no one single scheduling/dispatching policy can be optimal for all the scenarios. The question becomes more complex when different kinds of workloads need to be co-scheduled on the same set of servers. In addition to the problem of matching the dispatching and scheduling policies to the workload, server farms are also required to satisfy multiple conflicting performance goals – low mean response times, efficient utilization of resources, flexibility to adapt to varying and unpredictable demands, performance isolation for high priority jobs, minimizing energy, to name a few. This adds at least two more problem dimensions:

Question 3: Dynamic Capacity Scaling/Server Management: When and which back-end servers should be turned off, or hibernate, to save energy? When should servers be turned on to increase capacity?

Question 4: Provisioning: Given a cost budget, how should one design a server farm (in terms of number of servers and server speeds) to maximize performance?

In this thesis we will use queueing theoretic modeling and stochastic analysis techniques to guide these design decisions for modern day server farms. Queueing theoretic modeling allow us to abstract out the important features governing the performance to answer questions of the kind: What is the mean response time? What fraction of jobs experience response time larger than T_{max}? How sensitive are these performance metrics to parameter X? In addition to providing these answers for server farms of arbitrary size, stochastic analysis provides insights into the qualitative effect of various system parameters on the performance which may then be combined with other techniques, such as control theory and feedback systems, for operating the server farm.

However, existing stochastic modeling and analysis results do not directly apply to problems faced by today's server farms. Most existing results in stochastic analysis of multi-server systems were motivated by telephone networks, inventory management systems and call centers, and the assumptions which are valid in these problem domains do not hold for computer systems workloads:

1. **Assumption of FCFS back-end servers:** Most analysis of dispatching policies for multi-server systems assumes that the servers follow non-preemptive First-Come-First-Served (FCFS) scheduling policy. While this model fits problem domains which gave rise to the traditional multi-server models, such as telephone networks, call centers, hospital emergency rooms, queues in super-markets etc., computer systems such as *web servers* are better modeled as time-sharing systems. Designing and analyzing dispatching policies for time-sharing back-end servers is still an open question.

2. **Assumption of ideal time-sharing:** While there is a large body of work on analyzing a single time-sharing server, all the existing work models these time-sharing servers by the ideal Processor Sharing (PS) scheduling policy. Under PS, the server's capacity is independent of the number of concurrently running tasks. However time sharing systems, such as *database servers* and thread-based web servers exhibit *thrashing* which causes loss of server's capacity due to resource contention when too many tasks are concurrently active.

3. **Assumption of low job-size variability:** Even in application scenarios where traditional models of multi-server are a good fit, the existing analysis and approximations are severely lacking because such approximations are derived under the assumption that the service requirements (run time) of the jobs have low variance. One example of such an application scenario is *supercomputing centers* where jobs are typically scheduled in a non-preemptive FCFS fashion and thus the traditional call center model fits. However, existing approximations which are reasonably accurate for the problem domain of call centers can be off by unacceptable margins when the variance in service requirements is significant, which is the case in supercomputing workloads.

4. **Lack of evaluation of energy-performance trade-offs:** Traditional models of server farm have not dealt with the metric of *energy*, and understanding the trade-offs involved between energy and other performance metrics such as the mean response time. This questions becomes even more important in the context of *time-varying demand patterns* faced by today's server farms. Provisioning for peak demand is extremely wasteful of energy and thus it is imperative to develop algorithms to power down servers when demand is low and turn them back on when demand increases. However, the existing analysis does not deal with the associated setup costs and delays of powering servers up and down in a server farm.

5. **Assumption of a single workload:** Almost all stochastic analysis of multi-server systems assumes there is only a single 'job type', for example only web requests, or only

supercomputing jobs. However, increasingly, server farms are being used to *co-schedule multiple kinds of workloads*. The same server farm might be used as a corporation's web server farm, to provide email services, and as a platform for executing simulations for research. This leads to completely new problems in stochastic analysis of systems.

The goal of the thesis is to develop stochastic modeling and analysis techniques to bridge the gap between existing work on analysis of multi-server systems and the problems faced in management of compute server farms today. Before summarizing the research questions addressed in the thesis, we introduce the notation used throughout the document.

1.2 Notation and Preliminaries

The arrival and service processes constitute the most important aspects of a stochastic model. The arrival process specifies the instants at which jobs arrive into the system (server farm), and the service process specifies the size or processing requirements of the jobs. Unless otherwise stated, we assume that the arrival process is a renewal process, by which we mean that the times between consecutive arrivals are independent and identically distributed (*i.i.d.*) random variables. We use A to denote a generic interarrival time. We denote the mean of A by $\mathbf{E}[A] = \frac{1}{\lambda}$. Thus λ denotes the mean rate of arrivals. We also assume the job sizes form a sequence of *i.i.d.* random variables, where the random variable S (for service time distribution) denotes a generic job size. Since we will be dealing with models where the speed of the servers may be heterogeneous or state-dependent, rather than thinking of job sizes as being in units of time, we will use job sizes to denote the amount of work (for example the number of cycles for a CPU job, or file size for an I/O job). We denote the mean of S by $\mathbf{E}[S]$, and assume $\mathbf{E}[S] = 1$ without loss of generality. In empirical computing workloads it is often the case that A and S exhibit high variability. One of the most widely used metrics for characterizing this variability is the *the squared coefficient of variation* (SCV) of the interarrival and service distributions, C_A^2 and C_S^2, respectively:

$$C_A^2 = \frac{var(A)}{\mathbf{E}[A]^2}; \quad C_S^2 = \frac{var(S)}{\mathbf{E}[S]^2},$$

where $var(X)$ stands for the variance of random variable X. For a substantial part of the thesis we will assume that arrival process is Poisson, that is, A obeys the exponential distribution with mean $\frac{1}{\lambda}$, abbreviated as $A \sim \text{Exp}(\lambda)$, and will denote the arrival process by Poisson(λ). We will use the symbol μ to denote the service rate, or capacity of each server. When we talk about homogeneous server farms, where the capacity of each server is the same, we will use $\rho = \frac{\lambda \mathbf{E}[S]}{\mu}$ to denote the 'load' which represents the amount of work coming into the system per unit of time. We use T to denote the random variable for the response time, defined as the time between a job's arrival to the system and its departure from the system. We use D to denote the random variable for delay, defined as the total time spent in the system waiting to receive service.

Since the queueing models considered in the thesis are very different from the classical models, we will extend Kendall's notation to abbreviate them. In Kendall's notation, a queueing system shorthanded as $A/B/C/D$ denotes a system with arrival process A, service time distribution B, number of servers C, and scheduling policy D.[2] Typical values for the arrival process A we will use are: M (for Markovian) for a Poisson arrival process, M_t for a doubly stochastic Poisson process (that is, the mean arrival rate at time t is given by some function $\lambda(t)$), BPP for a Batch Poisson Process (Poisson process with a random number of arrivals at a time), and GI for general $i.i.d.$ interarrival times. Typical values for the service time distribution B we will use are: M for exponentially distributed, D for degenerate (non-random), H_2 for 2-phase hyperexponential (a mixture of two exponential distributions with different means), H_2^* for degenerate hyperexponential (mixture of an exponential distribution and a point mass at 0), and GI for general service time distribution. Typical values for the scheduling policy are FCFS for First-Come-First-Served, and PS for Processor Sharing (if there are n jobs queued at the server, each job gets $\frac{1}{n}$th of the server's capacity). However, we will sometimes combine the dispatching and scheduling policies in D. For example, we will use M/GI/K/JSQ-PS to denote the multi-server system with Poisson arrivals, general service distribution, and K servers where each server follows the processor sharing (PS) scheduling policy and every new arrival joins the shortest queue (JSQ).

1.3 Summary of research questions

We now give a formal summary of the gaps between existing analytical stochastic work on multi-server systems and the needs of practitioners the thesis aims to bridge. The aim of the thesis is not to bridge all the gaps at once, but to analyze each gap individually in depth to show how and where traditional policies and analysis fail. Each gap guides the answer to one or more of the four design decisions presented in Section 1.1. We motivate each gap with a computing application, and develop frameworks for optimizing and analyzing the performance under the unique constraints/opportunities presented by the motivating application. Table 1 provides a brief summary of the various pieces of the thesis.

Gap 1: High job-size variance of computer systems workloads invalidates existing approximations: We begin with an application scenario where traditional queueing models are in fact a good fit, but existing analytical approximations for the mean response time are insufficient when applied to workloads encountered by today's server farms. We illustrate this point by considering the M/GI/K/FCFS multi-server system which is one of the oldest and most classical queueing systems. The M/GI/K/FCFS system has traditionally been used as a model of systems as varying as call centers, manufacturing, hospital emergency rooms, and supercomputing centers. Despite being one of the oldest multi-server models, it is notoriously hard to analyze. Even expressions for the mean response time are not available for general service distributions. In

[2]Note the absence of dispatching policy in Kendall's notation.

	Gap	Motivating Application	Design Decisions Influenced	Contributions	Chpt.
1.	High job-size variability	Supercomputing	Provisioning	new analysis	2
2.	Thrashing/Resource Contention	Database concurrency control, thread-pool management	Scheduling	new model + analysis	3
3.	Server farms with time-sharing servers	Web server farms	Dispatching	new analysis	4
4.	Energy-performance trade-offs	Cloud computing, data centers	Capacity Scaling/ Server Management	new model + analysis	5
5.	Time-varying demands	Cloud computing, data centers	Capacity Scaling/ Server Management	new analysis	6
6.	Co-scheduling heterogeneous workloads	Enterprise server farms	Dispatching, Provisioning	new model + analysis	7

Table 1: A tabular summary of the questions addressed in the thesis.

the absence of such results, the following approximation proposed by Lee and Longton [59] which only involves the first two moments of the service distribution (S) is widely used:

$$\mathbf{E}\Big[D^{M/GI/K/FCFS}\Big] \approx \frac{C_S^2+1}{2}\mathbf{E}\Big[D^{M/M/K/FCFS}\Big]$$

where $D^{M/M/K/FCFS}$ denotes the delay in an M/M/K/FCFS system with the same mean job size, arrival rate and service rate as the M/GI/K/FCFS system ($D^{M/M/K/FCFS}$ has an exact and explicit expression).

We prove that, while being reasonably accurate in manufacturing systems where C_S^2 is small, the above approximation is grossly insufficient for workloads where C_S^2 is high [35]. This is significant because many computer systems workloads such as supercomputing jobs and sizes of files transferred over the Internet exhibit $C_S^2 > 40$. In fact we prove an *inapproximability result*: any approximation based only on the first two moments of the service distribution S must be inaccurate for some service distribution when C_S^2 is high. Motivated by this result, we conjecture tight bounds on the mean delay given the first n moments of S.

Gap 2: Effects of thrashing are ignored while analyzing PS-like systems: Next we consider the problem of developing scheduling policies for back-end servers. The application scenario we have in mind is database servers, and managing the thread pool in web servers. Processor sharing (PS) is an idealized scheduling policy commonly used to model time-sharing systems such as the CPU, bandwidth sharing systems, web and database servers. Under PS, a server shares its capacity equally among all the jobs in its queue. However almost all analytical results on the analysis of PS ignore the effects of *thrashing* [41]. Thrashing, or resource contention, causes the net capacity of a resource to decrease as the number of jobs concurrently sharing the resource increases,

and, in the absence of any concurrency control mechanism, brings the system to a halt. To get around this problem a Multi-Programming-Limit (MPL) is placed on the maximum number of jobs allowed to share the resource simultaneously and is almost always chosen to be the point of maximum efficiency (capacity). Existing work on analysis of PS with an MPL either ignores the effect of C_S^2, or assumes that μ (service rate/capacity) is independent of the state (number of jobs) of the system.

We present an approximate analysis of PS-MPL queueing systems to find the optimal MPL for minimizing the mean response time as a function of C_S^2, λ and the μ-vs.-MPL function [37]. The optimal MPL depends crucially on the arrival rate λ, which may not always be known at the time of system design, or may fluctuate at small time scales. Therefore, we desire a scheme which can self-adapt the MPL to variations in the arrival rate. As a second contribution, we develop traffic-oblivious dynamic MPL control policies which adapt the MPL based on the instantaneous queue length, rather than by attempting to learn the instantaneous arrival rate [37]. Finally, we propose the first heavy-traffic asymptotic scaling for analysis of 'non-work-conserving' time-sharing systems (i.e., systems where, depending on the current state, the service rate can be smaller than the peak service rate) and present a preliminary heavy-traffic approximation for GI/GI/PS-MPL systems.

Gap 3: No analysis of dispatching policies for PS server farms: The next problem we address focuses on developing smart dispatching policies for server farms. Motivated by supercomputing applications, there is a large body of work on analyzing dispatching policies for server farms where the servers operate under FCFS scheduling. The relative performance of different dispatching policies is well understood under FCFS servers. However, in many application scenarios, such as web servers and file downloads, the scheduling policy employed by servers is PS (in other words, preemptive scheduling policies are feasible). Unfortunately, dispatching policies which perform well for FCFS server farms may not perform well for PS server farms. In [39], we show via simulations that Least-Work-Left and Size-Based dispatching, which perform well under FCFS scheduling, are far from optimal under PS scheduling. By contrast Join-the-Shortest-Queue is near optimal, while being oblivious to the job sizes or the service distribution. We also find that JSQ-PS systems exhibit a *near-insensitivity* property: moments of S higher than the mean have minimal impact on the mean response time. We propose to formally investigate this phenomenon under a careful heavy-traffic scaling. We also propose to study the impact of heterogeneity in server capacities on the performance of JSQ-type dispatching policies. Proving the near-optimality of JSQ dispatching for PS servers is also a desirable result, but we consider it beyond the scope of the thesis in the available time frame.

Gap 4: Limited understanding of energy-performance trade-offs: Energy consumption has recently emerged as a key metric for the evaluation of scheduling policies and server management policies. Naturally there are trade-offs involved between minimizing the energy consumed and guaranteeing low response times. While one wants

to turn off idle servers, or put them into some sleep state, the penalties to boot up the servers may be prohibitive. We consider the metric of the product of the mean response time and mean power consumed (Energy-Response time-Product, ERP) to capture the trade-offs involved in minimizing energy consumptions and maximizing performance and analyze server management policies with respect to the ERP metric. Our goal is to show that optimal or near-optimal policies can be found within a substantially small set of policies, and further to provide rules of thumb to choose the right policy from among this set.

Gap 5: Time-varying arrival patterns: Most of the work on stochastic analysis of queueing systems has focused on Poisson or renewal arrival processes. This implies that the mean traffic demand remains constant over time. This assumption is violated in the real world, as the arrival patterns at web server farms and data centers, for example, show strong diurnal and seasonal variations. Designing robust server management policies which may self-adapt the capacity of the server farm to unpredictable arrival patterns is one of the holy grails of capacity provisioning. Unfortunately, systems with time-varying arrival patterns are not even very well understood analytically. In previous work [38], we proved that the answer to the question, *'Is a system with time-varying arrival pattern always worse than a system with constant mean arrival rate,'* is not always yes. While there is existing analytical work on designing server management policies for time-varying arrival patterns, the proposed policies either involve repeated static provisioning, or assume that the arrival pattern is known beforehand. Additionally, it is often assumed that server farm capacity may be increased instantaneously – an assumption that is not always justified. We propose to design and analyze traffic-oblivious server management policies for server farms, with the goal of optimizing energy-performance trade-offs.

Gap 6: Co-scheduling batch and production workloads: Motivated by the requirements of modern day server farms and data centers, the final problem of the thesis proposes a new research direction – designing dispatching policies for server farms serving heterogeneous workloads. Today's server farms rarely provide a single service. Large corporations like Google and Amazon use the same server farm for co-scheduling multiple workloads: answering search queries, mail, instant messaging, file downloads, and running map-reduce tasks for indexing the web and other research problems. Amazon also sells Virtual Machine (VM) instances to others on their clusters using their Elastic Compute Cloud (EC2) service. Naively scheduling all these workloads on the same server farm may cause unpredictable interference and performance. We look at a simple abstraction of this problem: co-scheduling *production* and *batch* jobs. Production jobs are long-running jobs with time-varying resource demands (e.g., jobs service online transactions). Further, production jobs once dispatched to a server are very costly to migrate to a different server. On the other hand, batch jobs are jobs in the usual sense – they arrive into the system according to some process, receive service, and finally depart (e.g., business analytics). However, production jobs have a higher priority, and it is desirable to guarantee them some performance isolation from the low

priority batch jobs. We aim to maximize the throughput of batch jobs, subject to a lower bound on the performance of production jobs (the precise metric of performance to be formalized later). Even if we ignore the presence of production jobs, the question of scheduling batch jobs is similar to the classic online stochastic bin packing problem, but where items may leave the system, and may also be migrated to yield a tighter packing. Even this simpler problem has not been addressed in the literature to the best of our knowledge.

Each gap presented above will correspond to a chapter of the thesis. Chapters 2 to 5 correspond to gaps 1 to 4 and mostly represent finished work. Chapters 6 and 7 correspond to gaps 5 and 6, respectively, are almost entirely open and proposed work.

2 The M/GI/K/FCFS multi-server model

While the majority of the thesis focuses on bridging the gaps between traditional models of queueing systems and the requirements of today's server farms, we begin the thesis by considering one application scenario where the traditional models are in fact a good fit, yet the existing analysis is insufficient when applied to workloads encountered in computing applications. We consider the M/GI/K/FCFS multi-server system with K homogeneous servers each of capacity μ and a FCFS buffer (see Figure 2). Arrivals from outside join the FCFS buffer, and, on becoming idle, a server picks a job from the head of the buffer. The M/GI/K/FCFS system is widely used for modeling systems where the jobs must be served in a non-preemptive fashion, and it is either desirable to serve jobs in the order of their arrival, or is the only option due to lack of information about the service requirements of the jobs. For example, M/GI/K/FCFS system has been widely used as a model for call centers, inventory management systems, hospital emergency rooms, and supercomputing systems, and new applications of this model are continually being discovered. The generality of the M/GI/K/FCFS system makes it a prime target for developing and testing new stochastic analysis techniques. Even small progress in the analysis of M/GI/K/FCFS could potentially have far-reaching consequences to problems beyond the M/GI/K/FCFS system itself. In our work, we take another tiny step towards better understanding the M/GI/K/FCFS system [35].

Despite the simplicity and versatility of the M/GI/K/FCFS model, it defies exact analysis for $K \geq 2$. The first approximation for the mean delay (waiting time) was proposed by Lee and Longton [59] nearly half century ago:

$$\mathbf{E}\Big[D^{M/GI/K/FCFS}\Big] \approx \frac{C_S^2+1}{2}\mathbf{E}\Big[D^{M/M/K/FCFS}\Big] \tag{1}$$

where $\mathbf{E}\Big[D^{M/M/K/FCFS}\Big]$ is the mean delay with Exponentially distributed job sizes with the same mean size $\mathbf{E}[S]$. Approximation (1) is known to be asymptotically tight as the system

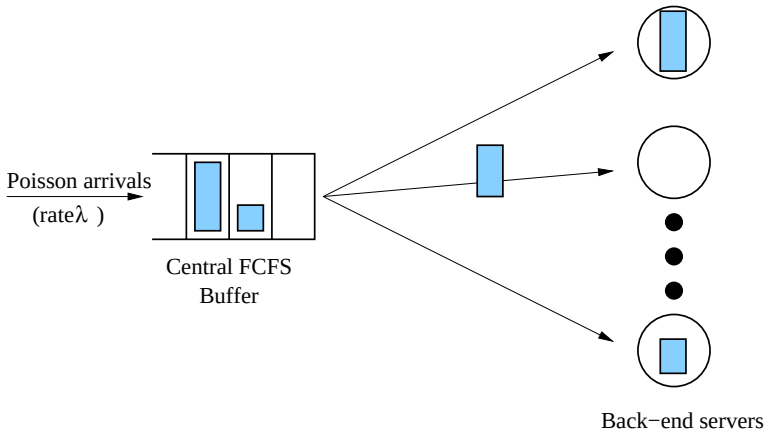


Figure 2: The M/GI/K/FCFS queueing system.

approaches critical stability (heavy traffic) while K remains constant. Many other authors have also proposed similar approximations for the mean waiting time [18, 42, 43, 58, 64, 78], but all these closed-form approximations involve only the *first two moments of the service distribution*.

Whitt [75], while referring to (1) as "usually an excellent approximation, even given extra information about the service-time distribution," hints that approximations based on two moments of the job size distribution may be inaccurate when C_S^2 is large. Similar suggestions have been made by many authors, but there are very limited numerical experiments to support this. While a high C_S^2 may not be of major concern in many applications like manufacturing or customer contact centers, the invalidity of the approximation (1) *is* a major problem in computer and communication systems. Values of $C_S^2 > 40$ are typical for workloads encountered in computer systems, such as the sizes of files transferred over the Internet [10], and the CPU requests of UNIX jobs [24] and supercomputing jobs [40].

2.1 An inapproximability result

In [35], we prove that essentially no approximation based only on the first two moments of S can be accurate for all service distributions with the given first two moments, when C_S^2 is high. This is achieved by studying the range of possible values of $\mathbf{E}\left[D^{M/GI/K/FCFS}\right]$ for general job size distributions with given first two moments, and showing a lower bound on the span of this range. Define

$$D_h^{C_S^2} = \sup\left\{\mathbf{E}\left[D^{M/GI/K/FCFS}\right] \;\middle|\; \mathbf{E}[S]=1, \mathbf{E}\left[S^2\right] = C_S^2+1\right\}, \tag{2}$$

and

$$D_l^{C_S^2} = \inf\left\{\mathbf{E}\left[D^{M/GI/K/FCFS}\right] \;\middle|\; \mathbf{E}[S]=1, \mathbf{E}\left[S^2\right] = C_S^2+1\right\}. \tag{3}$$

The following theorem is a slightly weaker version of the results we prove in [35].

Theorem 2.1 *Let $\rho = \frac{\lambda}{\mu}\mathbf{E}[S]$. For any finite C_S^2,*

$$D_h^{C_S^2} \geq \left(\frac{C_S^2+1}{2}\right)\mathbf{E}\Big[D^{M/M/K/FCFS}\Big] \quad \textit{for all } \rho < K$$

and,

$$D_l^{C_S^2} \leq \begin{cases} \mathbf{E}\Big[W^{M/M/K/FCFS}\Big] & \textit{if } \rho < K-1 \\ \mathbf{E}\Big[D^{M/M/K/FCFS}\Big] + \Big[\frac{\rho-(K-1)}{K-\rho}\Big]\frac{C_S^2-1}{2} & \textit{if } K-1 \leq \rho < K \end{cases}$$

where $\mathbf{E}\Big[D^{M/M/K/FCFS}\Big]$ is the mean delay when service distribution is exponential with mean 1.

If we focus on the parameter regime $\rho < K-1$ (traditionally called the one spare server regime), we observe that the upper bound of the span of possible values of mean delay is at least a factor $\frac{C_S^2+1}{2}$ larger than the lower bound (in fact we prove this to be at least C_S^2+1). Thus any approximation for the mean delay based only on the first two moments of S must be at least a factor $\sqrt{C_S^2+1}$ away for some service distribution, and the relative additive error of any approximation must be at least $\frac{C_S^2}{C_S^2+2} \approx 100\%$. This proves that for workloads with high C_S^2, the error in approximation (1) may be unacceptably high, and further our theorem indicates that this approximation will often be an overestimate of the mean delay. The last observation is verified via simulations in [35].

2.2 Conjectures on tight bounds

Given the insufficiency of the first two moments of S in determining the mean delay, one asks how would the knowledge of higher moments of the service distribution narrow the span of possible values. A few approaches in the literature are promising: Boxma et al. [18] provide a numerical approximation for M/GI/K/FCFS, which they verify to be reasonably accurate for job size distributions with low variability ($C^2 \leq 1$). This approximation is obtained by assuming a parametric form and matching the heavy traffic and light traffic behaviors. Since the light traffic behavior is rather explicit and is dependent on the entire service distribution, this approach is likely to yield more accurate approximations. Recently, Bertsimas and Natarajan [12] have proposed a computational approach based on semidefinite optimization to obtain bounds on the moments of waiting time in GI/GI/K/FCFS queues given the information of moments of the service time and the interarrival time distributions.

In [34], we make conjectures on the precise span of values of $\mathbf{E}\Big[D^{M/GI/K/FCFS}\Big]$ which are supported by our simulation experiments. In a nutshell, our conjectures say that knowledge

of each additional odd (respectively even) moment refines the lower (respectively upper) bound on the span of possible values of mean delay, and the new bound is a decreasing (respectively increasing) function of the moment. Further, these bounds are achieved by distributions which are mixtures of (approximately) $\frac{n}{2}$ point masses, where n is the number of available moment conditions. The conjectures help explain why approximation (1) has been useful in practice: most real world service distributions have a small third moment (relative to the second moment), and thus the mean delay for such service distributions are close to the upper bound. These conjectures echo the results of Stieltjes moment problem (see [50]) which asks for distributions which minimize or maximize the expectation of a function given the first n moments of the distribution, and are also known to hold for GI/M/1 queues with respect to the interarrival time distribution (see [25, 74]). However, proving the conjecture for M/GI/K/FCFS seems highly non-trivial and is outside the scope of the present thesis. A reasonable starting point would be a similar characterization of the mean delay under light traffic ($\rho \ll 1$), for which an explicit expression of the mean delay is known.

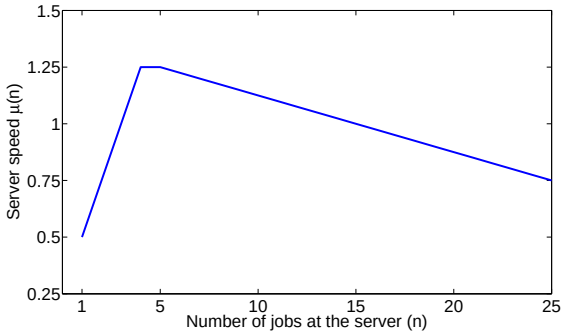
2.2.1 A general moment problem

In this section, we introduce a moment problem in pure analysis. This moment problem is motivated by our conjectures about the M/GI/K/FCFS system, and by the Stieltjes moment problem [50]. As we mentioned above, new analysis techniques for the M/GI/K/FCFS system would most likely have applications to analysis of stochastic models beyond the M/GI/K/FCFS system itself. For the M/GI/K/FCFS system we (i) proved that the first two moments of S are insufficient to approximate the performance, and (ii) conjectured tight bounds via successively higher moments of S. We believe that similar results are likely to hold in other application areas such as systems with fluctuating demands, work-stealing systems, limited server systems, and quantum-based round-robin scheduling in a single server. Observe that solution of all these models can be represented at some level by a stochastic fixed point equation:

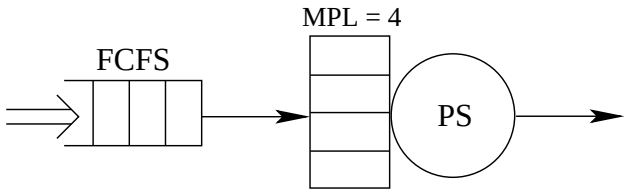
$$X \stackrel{d}{=} \phi(X, S) \tag{4}$$

where X is the unknown random vector capturing the performance of the system, and $\stackrel{d}{=}$ denotes equality in distribution. For example, for the M/GI/K/FCFS system, X is the vector of workloads at the K servers as seen by an arriving job if each job present in the system was assigned to the server it will ultimately serve on. The final performance metric of interest is then $\mathbf{E}[f(X)]$ for some function f.

We now pose the following problem: Our goal is to seek bounds on $\mathbf{E}[f(X)]$, given the first n moments of S. *For what class of functions $\phi(\cdot)$ and $f(\cdot)$ can these bounds be characterized along the lines of our conjectures above?*



(a) A prototypical service rate curve. The peak efficiency point for the curve shown is $K^* = 5$.



(b) A GI/GI/PS-MPL queue with $MPL = 4$.

Figure 3: Illustration of the GI/GI/PS-MPL model.

3 Scheduling Policies for Database Concurrency Control: The GI/GI/PS-MPL model

In this part of the thesis, we focus on designing scheduling policies for the smallest building block of a server farm: an individual back-end server. Two of the most common roles of these servers are to act as database servers, or as web/file servers. The architecture of such servers is based on multi-threading: a database/web server maintains a constant population of threads. Whenever a request arrives and finds a thread idle, the thread is assigned to processing the new request. Requests arriving to find all threads active and busy wait in a buffer. The reason for imposing a limit (called the Multi-Programming-Limit (MPL)) on the maximum number of active threads is thrashing, or resource contention, which causes the aggregate useful capacity of the server to drop beyond a certain level of concurrency [67, 73]. In practice, the MPL is always chosen to be the point maximizing the server's capacity [13, 26, 41]. However, there are no analytical results to understand *if choosing the peak efficiency point as the MPL minimizes the mean response time? If not, what should the MPL be?*

To analytically model such multi-threaded system, we start with a GI/GI/1/PS queue. As before, we assume that the service requirements of the jobs are *i.i.d.* with mean $\mathbf{E}[S] = 1$. In order to capture the fact that the speed of the system depends on the number of jobs at the server, we assume that our GI/GI/1/PS server has state-dependent service rates $\mu(n)$. That is, when the number of jobs at the server is n, the total speed of the server is $\mu(n)$, where $\mu(n)$ is chosen to match the system's service rate curve (Figure 3(a)). As an example, a job with service requirement of x units which is sharing the server with n jobs (including itself) for its entire duration would require $\frac{x}{\mu(n)} \cdot n$ time to complete. We assume that the $\mu(n)$ curve is unimodal, that is, initially it is non-decreasing and then after some point the curve switches to being non-increasing.To complete our model, we now add an MPL parameter which limits the number of jobs that are allowed to concurrently share the server to some

number MPL=K, and forces all remaining jobs to wait in a FCFS buffer. We assume that the service requirements of the jobs in the system are not known, and size-based prioritization is not possible. We denote our model by the notation G/GI/PS-MPL. Figure 3(b) depicts a G/GI/PS-MPL system with MPL=4.

3.1 Overview of prior work

When the service distribution is Exponential, the answer to choosing the optimal MPL is straightforward: We always want to operate at the peak efficiency point, regardless of the arrival process. The question of choosing an optimal MPL becomes interesting when the service distribution exhibits high variability (C_S^2), since with high variability service distributions, it is known that PS has a much better performance than FCFS because time sharing prevents small jobs from getting blocked behind big jobs. However, by letting too many jobs into the server, the system efficiency drops.

Unfortunately, the question of choosing an optimal MPL for high variability service distributions is difficult to answer since there is no known analysis even under a Poisson arrival process with a fixed arrival rate. This is not surprising because the M/GI/PS-MPL model is a generalization of the classical M/GI/K/FCFS multi-server system (the M/GI/K/FCFS multi-server system can be modeled by an M/GI/PS-MPL system with MPL = K, and $\mu(n) = \mu \cdot n$, where μ is the capacity of the individual servers). As we have already see, the performance analysis of the M/GI/K/FCFS system is still largely an open problem. While the performance analysis of the Processor-Sharing queue has been well understood for years, and research on the M/GI/1/PS queue has been abundant [16, 23, 54, 55, 56, 80, 82], very little is known about the M/GI/PS-MPL queue. Most analyses of the M/GI/PS-MPL queue do not allow for load-dependent service rates. For example, Itzhak and Halfin [6] derive an approximation for the mean response time for the M/GI/PS-MPL queue involving the first two moments of S when the service rate is fixed, while Zhang and Zwart [81] derive a heavy-traffic diffusion approximation for M/GI/PS-MPL, again with a fixed service rate. There is one analysis of the M/GI/PS-MPL system that does involve state-dependent service rates, see Rege & Sengupta [66]. However [66] requires that job sizes are exponentially-distributed while we are focusing on high-variability job size distributions which are more representative of computer workloads. Finally none of the above theoretical papers have tried to answer the question of how to set the MPL so as to minimize the mean response time.

3.2 Summary of contributions

Below we summarize the main results of our work [37] and some as yet unpublished results:

1. Optimal traffic-aware static policies: We derive the first approximation for the mean response time for the M/GI/PS-MPL queue with state-dependent service rates, and extend this approximation for GI/GI/PS-MPL systems. The approximation enables us to

choose the MPL that minimizes mean response time. Our approximations involve only the first two moments of interarrival distribution A and service distribution S, and yet always enable finding the optimal MPL, or an MPL with negligible performance deviation from the optimal. Via extensive simulation experiments, we demonstrate that the optimal MPL setting can be much higher than the peak efficiency point under job size variability characteristic of computer workloads.

2. Near-optimal traffic-oblivious dynamic policies: The above results assume that jobs arrive according to a Poisson process with a known arrival rate and propose the best *static* MPL. Our next goal is to design *light-weight* MPL control policies that adapt to the traffic characteristics, such as unknown arrival rates, burstiness or temporal correlations. By light-weight policies, we mean policies which take decisions based only on the instantaneous number of jobs in the buffer, $Q(t)$, and the instantaneous number of jobs at the server, $K(t)$.

Unsurprisingly, static MPLs are very poor at handling uncertainty in the mean arrival rate, and this effect is made worse by burstiness in arrival process. We propose light-weight MPL control policies that robustly handle these uncertainties. The *key idea in our approach* is that by considering a special class of job size distributions, the 2-phase degenerate hyperexponential distribution, we are able to incorporate the effect of job size variability in our optimization problem, while $(Q(t), K(t))$ remains a Markov process. Thus, the control policies we obtain are a function only of $(Q(t), K(t))$. While arriving at these dynamic policies, we develop some new techniques for performing Stochastic Dynamic Programming in Markov chains with infinite state space given they possess a special structure. We show via simulations that our proposed traffic-oblivious policies compare favorably to the optimal traffic-aware static policies.

3. A heavy-traffic scaling for non-work-conserving systems: As we have reiterated many times, systems such as M/GI/K/FCFS and GI/GI/PS-MPL are almost analytically intractable. In the absence of exact results, one looks for asymptotic regimes where exact analysis is possible. That is, one considers a sequence of systems indexed by some parameter r, such that as $r \to \infty$, the system behavior approaches some non-degenerate limit. Such asymptotic analysis provides insights into the most important parameters of the arrival process and service distribution driving the performance of the system and often leads to simple approximations. Zhang and Zwart [81] propose a heavy traffic scaling for the GI/GI/PS-MPL system for the case where service rate is a constant μ, and present a simple diffusion approximation for the mean response time which involves on the first two moments of A and S. The scaling involves increasing the MPL of the system as $K \cdot r$, and at the same time increasing the arrival rate so that $\mu - \lambda \propto \frac{1}{r}$. The GI/GI/PS-MPL system where service rate remains a constant is called *work-conserving* as the server is always processing at its peak capacity. However, we are interested in analyzing *non-work-conserving* GI/GI/PS-MPL systems, that is, with arbitrary service rate curves. Clearly, the naive extension of [81] by 'stretching' the service rate curve and increasing the arrival rate fails in this case and gives a degenerate limit. We propose the first heavy-traffic scaling for analysis of non-work-conserving GI/GI/PS-MPL systems which leads to a meaningful heavy traffic behavior. Our

proposed scaling maintains a constant arrival rate but the rth system has a service rate curve modified in such a way that, asymptotically as $r \to \infty$ under our heavy-traffic scaling, the distribution of the number of jobs in an M/M/PS-MPL with these parameters converges to a non-degenerate limit, this limit being given by a continuous and smooth interpolation of the distribution under the original unscaled system. We have also performed some preliminary analysis under our proposed scaling, but a complete diffusion analysis is left as future work.

4 Dispatching Policies for Web server farms: Join-the-Shortest-Queue routing with PS servers

In this section we consider the problem of analyzing dispatching policies for web server farm architectures. Requests for files (or HTTP pages) arrive at a front-end dispatcher. The dispatcher then *immediately* routes the request to one of the back-end servers in the farm for processing. It is important that the dispatcher not hold back the arriving connection request, or the client will time out and possibly submit more requests. The back-end servers in a web server farm time-share their capacity among the requests they are serving. We model the time-sharing of requests by the idealized PS scheduling policy, thus ignoring the effects of resource contention. Further, we assume that all the back-end servers are of homogeneous capacities.

The *Join-the-Shortest-Queue* (JSQ) dispatching policy is the most popular dispatching policy used in PS server farms today; e.g., it is used in Cisco Local Director, IBM Network Dispatcher, Microsoft Sharepoint and F5 Labs BIG/IP. Under JSQ, an incoming request is routed to the server with the least number of unfinished requests. Thus, JSQ strives to balance load across the servers, reducing the probability of one server having several jobs while another server sits idle. From the point of view of a new arrival, it is a *greedy policy* for the case of PS servers, because the arrival would prefer sharing a server with as few jobs as possible. We refer to a PS server farm with JSQ routing as a *JSQ-PS server farm*. For analytical tractability, we assume that the arrival process is Poisson. Thus the JSQ-PS server farm acts as an M/GI/K/JSQ-PS queueing system (see Figure 4).

Despite the ubiquity of JSQ-PS server farms, there are almost no analytical results available. Most work on dispatching policies in server farms assumes FCFS servers. However the performance of dispatching policies for PS and FCFS servers are very different, both qualitatively (how parameters of the service distribution affect the mean response time) and relative to each other (which dispatching policy is optimal). Our goals are to obtain approximations for the performance of JSQ-PS server farms, and to investigate optimality properties of JSQ routing under PS server farms.

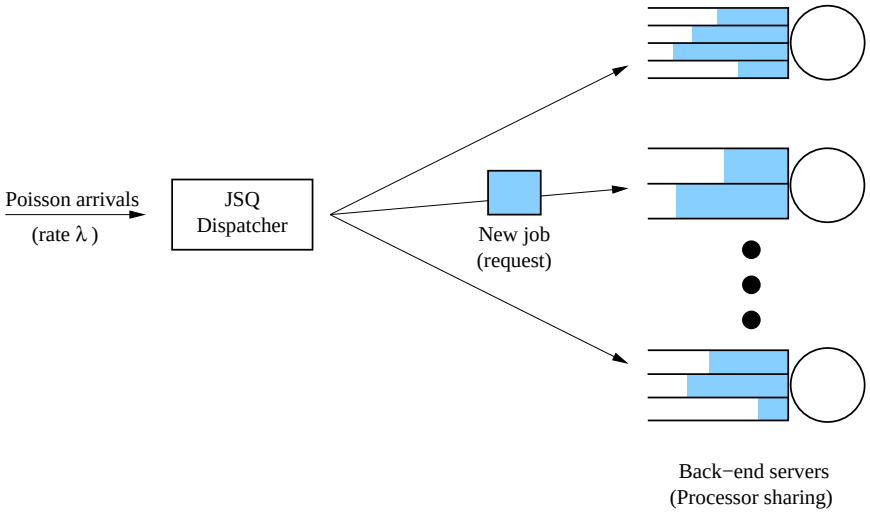


Figure 4: Server farm with JSQ dispatching and K identical PS servers

4.1 Overview of prior work

We emphasize again that there has been no previous mathematical analysis of the M/GI/K/JSQ-PS model. The work of Bonomi [14] who conducted a simulation study for the special case of two servers comes closest to the goals of this section. He showed that, among all policies that base their decisions only on the queue lengths at the servers, JSQ minimizes the mean response time for the PS scheduling policy and exponential service distributions. He showed via simulation that common dispatching schemes that perform well for JSQ-FCFS do not perform well for JSQ-PS. Bonomi observed that, while Least-Work-Left (LWL) is good for FCFS, it is not good for PS.

By contrast, there is a lot of work on the JSQ-FCFS model. However, even the M/M/K/JSQ-FCFS model remains quite intractable. Optimality of JSQ for FCFS under constrained settings has been shown in [27, 72, 77]. Almost all papers analyzing JSQ-FCFS performance are limited to 2 servers, exponential service distribution and the mean response time metric. Among the classic papers are Kingman [53], Flatto and McKean [28], and Wessels, Adan, and Zijm [4] who find the joint distribution as an infinite sum. Heavy traffic approximations for JSQ-FCFS are evaluated in [29, 57]. Lastly, Boxma and Cohen [17] obtain a functional representation for the mean response time using boundary value approach. These methods are exact. However they are not always computationally efficient and do not generalize to higher values of K. For analyzing JSQ-FCFS with more than $K = 2$ servers, again with exponential service distribution, approximate approaches are presented in [60, 63].

Recently Bramson et al. [19] have presented asymptotic analysis of 'JSQ-type' dispatching schemes. They consider dispatching policies of the following kind: an arrival picks d random servers out of K, and joins the most favorable (shortest queue, least work) among them. The

authors consider the limit where $K \to \infty$, and the arrival rate increases as $\lambda = \theta \cdot K$ ($0 < \theta < 1$). For PS servers with shortest queue criterion, the authors are able to show an *insensitivity result*: the mean response time depends only on the mean of the service distribution and not on the higher order characteristics. Intuitively one expects such a result to hold because, as $K \to \infty$, the queues become asymptotically independent of each other. Thus the arrival process into a particular queue is a *state-dependent Poisson process*, and insensitivity of mean response time to higher moments of the service distribution under such an arrival process is a well-known result. Previously, Mitzenmacher [62] had characterized the steady-state joint queue length distribution under the same asymptotic scaling and dispatching policy for exponential service distributions. At the outset, it is not clear if and when such an insensitivity result holds for the JSQ-PS model we consider.

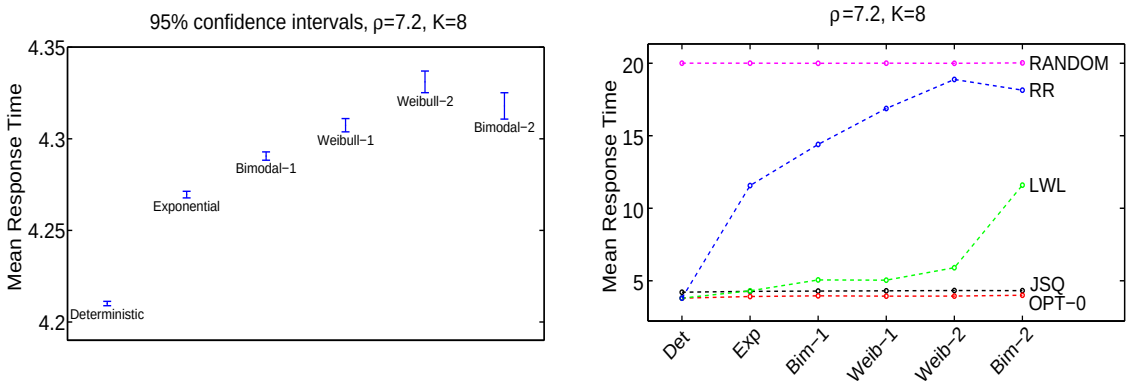
4.2 Summary of contributions

Below we summarize the main results of our work [39] and some proposed work:

Single Queue Approximation (SQA) technique for marginal queue length distributions: The main obstacle in analyzing the JSQ-PS model comes from the fact that the states of all the queues in the server farm are correlated due to the dispatching policy, necessitating a multidimensional state space of the system. The SQA approximation avoids this by analyzing just one of the queues independently of the others. To capture the effect of the dispatching policy under SQA, the arrival rate of jobs into queue q depends on the state of that queue in a manner that compensates for the presence of the other queues (we call these *conditional arrival rates*), but does *not* depend on the state of any other queue. Moreover, we use only the queue length as our state descriptor of the queue. Thus SQA approximates the M/GI/K/JSQ-PS model by an associated M_n/GI/1/PS model, where M_n denotes a state-dependent Markovian arrival process.

Thus, while the average arrival rate into each queue is $\frac{\lambda}{K}$, if we condition on the fact that some designated queue has n jobs, then the arrival rate into that designated queue is no longer $\frac{\lambda}{K}$. We present simple closed-form approximations for these conditional arrival rates, and show via simulations that for exponential service distribution, our method gives less than 2.5% error for both first and second moment of the number of jobs in a queue for K up to 64.

2. (Near)-Insensitivity: One of the most well-known properties of the M/GI/1/PS model is insensitivity: the mean response time is independent of the variability of the service distribution. We prove that under the degenerate hyperexponential (H_2^*) class of service distributions, even the JSQ-PS server farm demonstrates insensitivity. To examine other job-size distributions, we resort to extensive simulations of a wide class of distributions, including hyperexponential distributions, Weibull distributions, deterministic distribution and bimodal distributions (mixture of two point masses) with C_S^2 up to 25. We find, see Figure 5(a), that the JSQ-PS system shows *near-insensitivity* to the variability of the job



(a) Near-insensitivity of the JSQ-PS model

(b) Near-optimality of the JSQ-PS model

Figure 5: Simulation results illustrating the properties of the M/GI/K/JSQ-PS model. The service distributions are arranged on the x-axis in order of increasing C_S^2 (the C_S^2 values are $\{0, 1, 2.25, 5, 19, 24.75\}$, respectively).

size distribution. In fact the mean queue length varies by less than 3% over a huge range of variability, and the second moment of queue length is only slightly more sensitive. We propose to further investigate this interesting observation under a (in fact the only) suitable non-trivial heavy-traffic scaling.

3. Near-Optimality: We compare JSQ with other dispatching policies (Random, Round-Robin (RR), Least-Work-Left (LWL), and OPT-0 (a greedy job-size aware policy proposed by Bonomi [14])) in the PS server farm setting via simulations (see Figure 5(b)). We find that JSQ is impressively close to achieving optimality, despite being oblivious to the job sizes or the service distribution. We conjecture that M/D/K/RR-PS is a lower bound on the mean response time of M/GI/K/X-PS for any service distribution and dispatching policy X. This will be a useful starting point for analytically proving near-optimality of the JSQ dispatching policy.

4.3 Many-servers analysis of JSQ

The above work on analyzing JSQ was limited to homogeneous servers and was mostly based on observations made via empirical evaluations. In fact, we are not aware of any closed-form analytical results regarding JSQ for more than 3 servers. *Is JSQ analytically tractable in some limiting regime? Does the optimality of JSQ for Exponential service distribution extend to heterogeneous servers? What are the causes for the observed near-insensitivity, and can it be quantified?*

We propose to answer the above questions by analyzing the JSQ-PS server farm in, what we believe, one of the only non-trivial heavy-traffic scaling. Here we present the proposed

heavy-traffic scaling for the case of homogeneous servers: we consider a sequence of JSQ-PS systems indexed by a parameter $r \in \mathbb{N}$. The rth system has number of servers $K^{(r)} = r$, and arrival rate $\lambda^{(r)} = r - \theta$, for a positive constant θ. We first justify the scaling. Consider the simpler classical scaling: the number of servers K is held constant while the arrival rate for the rth system increases as $\lambda^{(r)} = K - \frac{1}{r}$. Under this scaling, the number of jobs in the system increases linearly in r, and hence the K-server JSQ-PS server effectively acts as an $M/GI/1/PS$ system with capacity K begetting perfect insensitivity, and essentially behaving as an $M/M/K$. Now consider the scaling $K^{(r)} = r$ and $\lambda^{(r)} = r - g(r)$, where $g(r) = \omega(1)$ (simply put, $g(r) \to \infty$ as $r \to \infty$). Under this scaling, the number of jobs in the system is $r + \Theta\left(\frac{r}{g(r)}\right) \sim r + o(r)$. Thus, the mean response time converges to $\mathbf{E}[S]$, again behaving as an $M/M/K$ and resulting in perfect insensitivity. The scaling $K^{(r)}$, $\lambda^{(r)} = r - o(1)$ is similar to the classical heavy-traffic scaling where the JSQ-PS server farm acts as a single PS server.

Our line of attack will be to consider the JSQ-PS server farm under the proposed heavy-traffic scaling and Exponential service distribution with the aim of developing the first closed-form analytical results on JSQ. Next, we propose to look at the question of optimality of JSQ for heterogeneous server farms. Here it is not clear if JSQ is still optimal even for Exponential service distribution. In fact, a scheme that is commonly proposed is Minimum Expected Delay (MED) [61], where jobs are sent to the server with the shortest queue length scaled by the server's capacity. We also propose to investigate optimality of routing schemes under general service distributions. Finally, time-permitting, we propose to investigate the phenomenon of near-insensitivity of JSQ under homogeneous server farms and two-phase hyperexponential (H_2) service distribution. This allows us to operate within the framework of Markov chains, and hence a countable state space. By varying the parameters of the H_2 distribution, we would be able to see the effects of service distribution on the mean response time. We would like to point out another system for which bounded-sensitivity results are known: $M/GI/1$ with quantum-based Round-Robin scheduling [33], which is a relaxation of PS scheduling policy. The results in [33] were obtained by obtaining bounds on the mean response time via monotone linear system of equations. It is interesting to see if similar techniques may be applied to prove bounded- or near-insensitivity results for general K and λ for the JSQ-PS model. In his dissertation, Don Towsley [69, 70] designed dispatching policies for *closed systems* which attempt to mimic JSQ by probabilistically biasing towards shorter queues, but lead to perfect insensitivity because reversibility holds under the proposed dispatching policies. However, Jonckheere [47] has showed that the optimal perfectly insensitive dispatching policy for PS server farms must be static (that is where the tasks are dispatched probabilistically to back-end servers).

5 Optimizing Energy-Performance Trade-offs

Power management has become a first-order goal in the design and provisioning of server farms today. Server farm power consumption accounts for more than 1.5% of the total electricity usage in the U.S., at a cost of nearly $4.5 billion [71]. The rising cost of energy and the tremendous growth of data centers will result in even more spending on power consumption. Unfortunately, only 20-30% of the total server capacity is used on average [11]. The main culprit of this wastage are idle servers in over-provisioned server farms. While one would like to turn servers off when they become idle to save energy, the large setup cost (both, in terms of setup time and energy penalty) needed to switch the server back on can adversely affect performance. The problem is made more complex by the fact that today's servers provide multiple sleep or standby states which trade off the setup cost with the power consumed while the server is 'sleeping'. With so many controls, finding the optimal server pool management policy is an almost intractable problem.

The goal of this section to find a simple class of server farm management policies, which optimize (or nearly optimize) the aforementioned energy-performance trade-off. We also seek simple rules of thumb that allow designers to choose from this class of near-optimal policies. In doing so, we greatly simplify the job of the server farm manager by reducing the search space of policies that he/she needs to choose from.

To capture the trade-off involved in energy and performance, and to compare different policies, we use the Energy-Response time-Product (ERP) metric, also known as the Energy-Delay Product (EDP) [31, 48, 49, 52, 68]. For a control policy π, the ERP is given by:

$$ERP^{\pi} = \mathbf{E}[P^{\pi}] \cdot \mathbf{E}[T^{\pi}]$$

where $\mathbf{E}[P^{\pi}]$ is the long-run average power consumed under the control policy π, and $\mathbf{E}[T^{\pi}]$ is mean customer response time under policy π. Minimizing the quantity ERP can be seen as maximizing the "performance-per-watt", with performance being defined as the inverse of mean response time. While ERP is widely accepted as a suitable metric to capture energy-performance trade-offs, we believe we are the first to analytically address optimizing the metric of ERP in server farms.

5.1 Overview of prior work

The problem of server farm management is very similar in flavor to two well studied problems in the stochastic analysis community: operator staffing in call centers and inventory management.

The operator staffing problem involves finding the number of operators (servers) which minimize a weighted sum of delay costs experienced by users and the monetary cost of staffing operators. This problem has received significant attention under the assumption of stationary (non-time-varying) demand (see [15] for recent results), while there is very limited work

under time-varying scenarios (see [46]). In [46], the authors consider the problem of dynamic staffing based on knowing the demand pattern so as to maintain a target probability of a user finding all servers busy on arrival. However, all the existing work on operator staffing ignores the setup cost involved.

Within inventory management, the problem of capacity provisioning takes the form: how much inventory should one maintain so as to minimize the total cost of unused inventory (holding cost, in our case *idle* power) and waiting cost experienced by orders when there is no inventory in stock (queueing delay of users). Conceptually this problem is remarkably similar to the problem we consider, and the two common solution strategies employed, known as Make to Order and Make to Stock, are similar in flavor to the commonly employed server farm management policies of always turning off a server when idle and never turning off an idle server, respectively (see [3], for example).

There is a large body of work on power management from the point of view of minimizing worst case sample path costs (see [44] for a recent survey). Again, none of the prior work encompasses a setup time and is more applicable to a single device than a server farm. The performance metrics used are also very different from ERP. The work can primarily be split in terms of results on speed scaling algorithms, and results on algorithms for powering down devices. In the realm of speed scaling, the problem flavors considered have been minimizing energy or maximum temperature while meeting job deadlines [8, 9, 79], minimizing mean response time subject to a bound on total energy [65], and minimizing a weighted sum of mean response time and mean power [7, 76]. However, again all these papers assume that the speed level can be switched *without any setup costs*, and hence are mainly applicable to single stand-alone devices, since in multi-server systems setup costs are required to increase capacity.

The work on powering down devices is more relevant to the problem we consider, and due to sample path guarantees, these results naturally lead to traffic-oblivious powering down schemes. In [45] the authors consider the problem of minimizing total energy consumed under the constraint that a device must instantly turn on when a job arrives. Further, [45] assumes that there is *no setup time* while turning on a device, only an energy penalty. Recent work of Khuller, Li and Saha [51] considers the problem of minimizing the makespan of a set of jobs given a power budget.

5.2 Summary of contributions

In [36], we propose a model for the sleep states of servers and prove results on the optimality or near-optimality of certain simple classes of policies under the assumption of a Poisson arrival process and exponential service distribution. Our results demonstrate that while the optimal policy may be very complex and computationally expensive to obtain due to the high dimensionality of the search space, simple policies suffice for near-optimal operation. The value of our results is further amplified due to the fact that the ERP metric being a

product of two expectations can not be optimized via the standard technique of Stochastic Dynamic Programming.

1. Optimality results for a single server: We prove that the policy that minimizes the ERP metric must be one of the following: (*i*) always keep the server active, even when idle; (*ii*) turn the server off whenever idle, and start turning on the server as soon as a job arrives; or (*iii*) put the server into a chosen sleep state whenever idle, and wake up the server (turn it on) as soon as a job arrives. While this result may seem obvious, we emphasize that it is not. In particular, the optimality results do not hold for other metrics such as weighted sum of mean response time and mean power, or the product of mean power and the square of mean response time, for example.

2. Near-optimality results for multi-server systems: For proving multi-server results, we assume an M/M/K$_t$/FCFS architecture. That is, jobs queue up in a FCFS buffer on arrival. The number of active servers at time t, K$_t$, is dictated by the server farm management policy, but whenever an active server becomes idle, it picks a job to serve from the head of the FCFS buffer. We prove that the multi-server generalizations of the single server policies mentioned above are nearly optimal. That is a near-optimal policy may be found within the following: (*i*) a fixed optimally chosen number K^* of servers are always kept active, even when idle; (*ii*) an idle server is always turned off, and whenever a job arrives a server is turned on to serve the new job; or (*iii*) a fixed optimally chosen number K^* of servers are maintained either in the busy or in a chosen sleep state (same for all servers) – an idle server is put into the sleep state, and if an arriving job finds a server in sleep state, the sleeping server is woken up.

Note that we have assumed in this section that any job may be dispatched to any server. This is a valid assumption when the jobs are CPU bound. However, a significant fraction of the workload in today's cloud computing centers and data centers is I/O bound, which enforces that jobs may only be assigned to a small set of servers capable of processing them. Extending our schemes to such I/O bound workloads is a topic for future research.

6 Dynamic capacity planning under time-varying arrival patterns

While the results we obtained in Section 5 on optimizing energy-performance trade-off for Poisson arrivals are a worthy contribution, they are of limited applicability in practice as the arrival patterns at server farms and cloud computing centers are highly time-varying, exhibiting strong diurnal and seasonal variations in demand. Provisioning for peak demand is highly wasteful of energy. One commonly adopted solution is to do repeated static provisioning [21]. For example, every hour a forecast is made for the demand (mean arrival rate) during the next hour, and a fixed number of servers are commissioned under the assumption that the arrival process during the next hour is Poisson with the forecasted mean arrival

rate. However, we seek simple dynamic capacity provisioning schemes which adapt to the demand while being traffic-oblivious. We propose design and analysis of such schemes as a topic to be explored in the present thesis.

Our first goal is propose heuristic traffic-oblivious dynamic capacity provisioning schemes for the $M_t/M/K_t/FCFS$ model. That is, the arrival process is Poisson with a general time-dependent arrival intensity $\lambda(t)$, and the service distribution is exponential (we believe that the assumption of Exponential service distribution is not really restrictive and the optimality/near-optimality results would extend to general service distributions). We propose to supplement these heuristics with analytical arguments, as well as simulations for cases where analysis is not possible. We will compare our scheme against the scheme proposed in [46] for call center staffing, where the authors propose a capacity provisioning scheme based on knowing the arrival pattern beforehand, and thus can be viewed as an offline optimal scheme. Time permitting, we will also look at the question of dynamic capacity planning for server farms with I/O bound workloads: i.e., where servers are not completely interchangeable and each job is constrained to a small set of back-end servers that it may be dispatched to.

7 Co-scheduling batch and production workloads

In this part of the thesis, we look at a problem which differs from all the problems we have looked so far in at least three ways: (*i*) we look at the problem of co-scheduling high-priority and low-priority tasks, (*ii*) our metric of performance will be throughput rather than response time, and (*iii*) we look at the problem of scheduling *incompressible* resources such as memory (unlike *compressible* resources, such as CPU). To elaborate on the third point, scheduling a memory-intensive job differs from scheduling a CPU-intensive job in that in the latter case, a job not getting enough CPU cycles just takes longer to finish. However, if a memory-bound job does not get enough memory, it can not begin execution. In this sense, scheduling memory-bound jobs is akin to the problem of bin packing.

To make the problem more concrete, we propose to look at the problem of co-scheduling *batch* and *production* jobs in a heterogeneous server farm. Production jobs are jobs which are constantly present in the system, but whose resource demands vary over time. Batch jobs are jobs in the traditional sense: they arrive to the system according to some arrival process (the size of a batch job is given by a pair of numbers, denoting the resource demand (e.g. the memory requirement) and the execution time) and are scheduled on the servers according to some dispatching policy – and thus share the resources with the production jobs.

Even the problem of scheduling only the batch jobs is novel, and combines the problem flavors of online stochastic bin packing [30] and online storage allocation [22]. While in online stochastic bin packing items must be packed while respecting bin boundaries (as is

also the case for batch jobs), the items do not depart the system. On the other hand, in dynamic storage allocation, memory requests may be allocated any contiguous free chunk, but the requests leave (or the memory is freed) at a later point in time similar to batch jobs. Further, in our case, the batch jobs may also be migrated (at some cost) to obtain a tighter packing.

We plan to look at two formulations of the problem:

1. **Dispatching policies:** Given a fixed server farm architecture, i.e. the number of servers and their possibly heterogeneous capacities (memory size or CPU cores), what are optimal or near-optimal heuristics to schedule the jobs.
2. **Server Farm design:** It is clear that having just 1 server with huge capacity is desirable because this allows more efficient packing of the stochastic resource demands of the tasks. However, server costs are typically convex increasing in their capacities – a server of capacity μ is more than twice as expensive as a server with capacity $\frac{\mu}{2}$. Given a fixed monetary budget, how to optimize the server farm architecture (i.e. how many servers to provision and what should be the capacities of the servers), so as to achieve a target throughput of batch jobs and target availability probability of production jobs.

The proposed work is still in the conception stage. We do not expect to solve the problem in the available time, but expect to make significant inroads into the problem. The line of attack will be the following:

1. Perform workload characterization of cluster workloads [2] to create tractable stochastic models for batch and production jobs.
2. Propose and analyze heuristics for optimizing the performance metrics described above given a server farm architecture.
3. Answer the server farm design questions: Does it help to have a few servers with large capacities, or many servers with moderate capacities?

Apart from online stochastic bin packing and online dynamic storage allocation, the problem also combines aspects of scheduling with outliers. Scheduling with outliers [20, 32] is a relatively new research area. Here one asks that if on a given arrival sequence the optimal scheduling algorithm achieves a certain performance, how close can a heuristic perform if it is allowed to drop an ϵ fraction of jobs. However, in stochastic scheduling theory, we may view this question as finding the stability region for the arrival process of the batch jobs.

8 Proposed Time Line

Sections 2 to 7 correspond to Chapters 2 to 7 of the thesis. Sections 2, 3 and 5 are mostly complete. The following sections are largely open proposed work: 2.2, 4.3, 6 and 7. We consider addressing the conjectures in Section 2.2 as being beyond the scope of the thesis and propose them as open questions to be solved in the long term. The following time line is proposed for completing the proposed work in Sections 4.3 (many-servers analysis of JSQ-PS), 6 (dynamic capacity scaling for time-varying demands), and 7 (co-scheduling heterogeneous workloads):

Oct. 2010	Complete the work proposed in Section 6
Oct. 2010 - Nov. 2010	Make progress on the work proposed in Section 7 (submit to Sigmetrics 2011)
Nov. 2010 -Jan. 2011	Complete the work proposed in Section 4.3 (submit to Queueing Systems/Operations Research)
Jan. 2011 - Feb. 2011	Tie up loose ends, attempt some of the harder open questions time-permitting (e.g. Section 2.2)
March - May 2011	Write thesis

References

[1] `http://www.research.ibm.com/bluegene/`.

[2] `http://code.google.com/p/googleclusterdata/`.

[3] I. Adan and J. v. d. Wal. Combining make to order and make to stock. *OR Spektrum*, 20:73–81, 1998.

[4] I. Adan, J. Wessels, and W. Zijm. Analysis of the symmetric shortest queue problem. *Stochastic Models*, 6:691–713, 1990.

[5] D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan. FAWN: A Fast Array of Wimpy Nodes. In *SOSP '09: Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 1–14, New York, NY, USA, 2009. ACM.

[6] B. Avi-Itzhak and S. Halfin. Expected response times in a non-symmetric time sharing queue with a limited number of service positions. *In Proceedings of ITC*, 12:5.4B.2.1–7, 1988.

[7] N. Bansal, H.-L. Chan, and K. Pruhs. Speed scaling with an arbitrary power function. In *SODA '09: Proceedings of the Nineteenth Annual ACM -SIAM Symposium on Discrete Algorithms*, pages 693–701, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[8] N. Bansal, T. Kimbrel, and K. Pruhs. Dynamic speed scaling to manage energy and temperature. In *FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 520–529, Washington, DC, USA, 2004. IEEE Computer Society.

[9] N. Bansal, T. Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *J. ACM*, 54(1):1–39, 2007.

[10] P. Barford and M. Crovella. Generating representative web workloads for network and server performance evaluation. *Proceeding of ACM SIGMETRICS/Performance'98*, pages 151–160, 1998.

[11] L. A. Barroso and U. Hölzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, 2007.

[12] D. Bertsimas and K. Natarajan. A semidefinite optimization approach to the steady-state analysis of queueing systems. *Queueing Syst.*, 56(1):27–39, 2007.

[13] R. Blake. Optimal control of thrashing. In *Proceedings of the 1982 ACM SIGMETRICS Conference on Measurements and Modeling of Computer Systems*, 1982.

[14] F. Bonomi. On job assignment for a parallel system of processor sharing queues. *IEEE Transactions on Computers*, 39(7):858–869, 1990.

[15] S. Borst, A. Mandelbaum, M. I. Reiman, and M. Centrum. Dimensioning large call centers. *Operations Research*, 52:17–34, 2000.

[16] S. Borst and R. Núñez-Queija. Introduction to special issue on queueing models for fair resource sharing. *Queueing Syst.*, 53(1-2):5–6, 2006.

[17] O. Boxma and J. Cohen. *Boundary value problems in queueing system analysis*. North Holland, 1983.

[18] O. Boxma, J. Cohen, and N. Huffels. Approximations in the mean waiting time in an $M/G/s$ queueing system. *Operations Research*, 27:1115–1127, 1979.

[19] M. Bramson, Y. Lu, and B. Prabhakar. Randomized load balancing with general service time distributions. Submitted.

[20] M. Charikar and S. Khuller. A robust maximum completion time measure for scheduling. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 324–333, New York, NY, USA, 2006. ACM.

[21] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao. Energy-aware server provisioning and load dispatching for connection-intensive internet services. In *NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pages 337–350, Berkeley, CA, USA, 2008. USENIX Association.

[22] E. G. Coffman, Jr. and F. T. Leighton. A provably efficient algorithm for dynamic storage allocation. In *STOC '86: Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 77–90, New York, NY, USA, 1986. ACM.

[23] E. G. Coffman, Jr., R. R. Muntz, and H. Trotter. Waiting time distributions for processor-sharing systems. *J. Assoc. Comput. Mach.*, 17:123–130, 1970.

[24] A. Downy and M. Harchol-Balter. Exploiting process lifetime distributions for dynamic load balancing. *ACM Transactions on Computer Systems*, 15(3):253–285, August 1997.

[25] A. Eckberg Jr. Sharp bounds on Laplace-Stieltjes transforms, with applications to various queueing problems. *Math. Oper. Res.*, 2(2):132–142, 1977.

[26] S. Elnikety, E. Nahum, J. Tracy, and W. Zwaenepoel. A method for trasparent admission control and request schedulin in e-commerce web sites. In *World-Wide-Web Conference*, 2004.

[27] A. Ephremides, P. Varaiya, and J. Walrand. A simple dynamic routing problem. *IEEE Transac. on Auto. Cont.*, AC-25(4):690–693, 1980.

[28] L. Flatto and H. McKean. Two queues in parallel. *Communication on Pure and Applied Mathematics*, 30:255–263, 1977.

[29] G. Foschini and J. Salz. A basic dynamic routing problem and diffusion. *IEEE Trans. Comm.*, 26(3):320–328, 1978.

[30] D. Gamarnik and M. S. Squillante. Analysis of stochastic online bin packing processes. *Stochastic Models*, 21:401 – 425, 2005.

[31] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits*, 31(9):1277–1284, 1996.

[32] A. Gupta, R. Krishnaswamy, A. Kumar, and D. Segev. Scheduling with outliers. In *APPROX '09 / RANDOM '09: Proceedings of the 12th International Workshop and 13th International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 149–162, Berlin, Heidelberg, 2009. Springer-Verlag.

[33] V. Gupta. Finding the optimal quantum size: Sensitivity analysis of the $M/G/1$ round-robin queue. *SIGMETRICS Perform. Eval. Rev.*, 36(2):104–106, 2008.

[34] V. Gupta, J. Dai, M. Harchol-Balter, and B. Zwart. The effect of higher moments of job size distribution on the performance of an $M/G/K$ queueing system. Technical Report CMU-CS-08-106, School of Computer Science, Carnegie Mellon University, 2008.

[35] V. Gupta, J. Dai, M. Harchol-Balter, and B. Zwart. On the inapproximability of $M/G/K$: why two moments of job size distribution are not enough. *Queueing Systems*, 64(1):5–48, 2010.

[36] V. Gupta, A. Gandhi, M. Harchol-Balter, and M. Kozuch. Energy-efficient dynamic capacity provisioning in server farms. Technical Report CMU-CS-10-108, School of Computer Science, Carnegie Mellon University, 2010.

[37] V. Gupta and M. Harchol-Balter. Self-adaptive admission control policies for resource-sharing systems. In *Proceedings of ACM SIGMETRICS '09*, pages 311–322, New York, NY, USA, 2009. ACM.

[38] V. Gupta, M. Harchol-Balter, A. Scheller-Wolf, and U. Yechiali. Fundamental characteristics of queues with fluctuating load. In *Proceedings of ACM SIGMETRICS '06*, pages 203–215, 2006.

[39] V. Gupta, M. Harchol-Balter, K. Sigman, and W. Whitt. Analysis of join-the-shortest-queue routing for web server farms. In *PERFORMANCE 2007. IFIP WG 7.3 International Symposium on Computer Modeling, Measurement and Evaluation*, Cologne, Germany, Oct. 2007.

[40] M. Harchol-Balter and B. Schroeder. Evaluation of task assignment policies for supercomputing servers. In *Proceedings of 9th IEEE Symposium on High Performance Distributed Computing (HPDC '00)*, August 2001.

[41] J. L. Hellerstein, V. Morrison, and E. Eilebrecht. Applying control theory in the real world: experience with building a controller for the .net thread pool. *SIGMETRICS Perform. Eval. Rev.*, 37(3):38–42, 2009.

[42] P. Hokstad. Approximations for the $M/G/m$ queue. *Operations Research*, 26(3):510–523, 1978.

[43] P. Hokstad. The steady state solution of the $M/K_2/m$ queue. *Adv. Appl. Prob.*, 12(3):799–823, 1980.

[44] S. Irani and K. R. Pruhs. Algorithmic problems in power management. *SIGACT News*, 36(2):63–76, 2005.

[45] S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. *ACM Trans. Algorithms*, 3(4):41, 2007.

[46] O. B. Jennings, A. M, W. A. Massey, and W. Whitt. Server staffing to meet time-varying demand. *Management Science*, 42:1383–1394, 1996.

[47] M. Jonckheere. Insensitive versus efficient dynamic load balancing in networks without blocking. *Queueing Syst. Theory Appl.*, 54(3):193–202, 2006.

[48] P. Juang, Q. Wu, L.-S. Peh, M. Martonosi, and D. W. Clark. Coordinated, distributed, formal energy management of chip multiprocessors. In *ISLPED '05: Proceedings of the 2005 international symposium on Low power electronics and design*, pages 127–130, New York, NY, USA, 2005. ACM.

[49] C. W. Kang, S. Abbaspour, and M. Pedram. Buffer sizing for minimum energy-delay product by using an approximating polynomial. In *GLSVLSI '03: Proceedings of the 13th ACM Great Lakes symposium on VLSI*, pages 112–115, New York, NY, USA, 2003. ACM.

[50] S. Karlin and W. J. Studden. *Tchebycheff systems: With applications in analysis and statistics*. John Wiley & Sons Interscience Publishers, New York, 1966.

[51] S. Khuller, J. Li, and B. Saha. Energy efficient scheduling via partial shutdown. In *SODA '10: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, 2010.

[52] J. Kin, M. Gupta, and W. Mangione-Smith. The filter cache: an energy efficient memory structure. *Microarchitecture, IEEE/ACM International Symposium on*, 0:184, 1997.

[53] J. Kingman. Two similar queues in parallel. *Biometrika*, 48:1316–1323, 1961.

[54] L. Kleinrock. Analysis of a time-shared processor. *Naval research logistics quarterly*, pages 59–73, 1964.

[55] L. Kleinrock. Time-shared systems: A theoretical treatment. *J. Assoc. Comput. Mach.*, 14:242–261, 1967.

[56] L. Kleinrock. *Queueing Systems; Volume 2: Computer Applications*. Wiley, New York, 1976.

[57] C. Knessl, B. Matkowsky, Z. Schuss, and C. Tier. Two parallel $M/G/1$ queues where arrivals join the system with the smaller buffer content. *IEEE Trans. Comm.*, 35(11):1153–1158, 1987.

[58] J. Köllerström. Heavy traffic theory for queues with several servers. I. *J. Appl. Prob.*, 11:544–552, 1974.

[59] A. Lee and P. Longton. Queueing process associated with airline passenger check-in. *Operations Research Quarterly*, 10:56–71, 1959.

[60] H. Lin and C. Raghavendra. An analysis of the join the shortest queue (JSQ) policy. In *Proc. 12th Int'l Conf. Distributed Computing Systems*, pages 362–366, 1992.

[61] J. Lui, R. Muntz, and D. Towsley. Bounding the mean response time of the minimum expected delay routing policy: an algorithmic approach. *IEEE Trans. Comp.*, 44(12):1371–1382, 1995.

[62] M. Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Trans. Parallel Distrib. Syst.*, 12(10):1094–1104, 2001.

[63] R. Nelson and T. Philips. An approximation to the response time for shortest queue routing. *ACM Perf. Eval. Review*, 17:181–189, 1989.

[64] S. Nozaki and S. Ross. Approximations in finite-capacity multi-server queues with Poisson arrivals. *J. Appl. Prob.*, 15(4):826–834, 1978.

[65] K. Pruhs, P. Uthaisombut, and G. Woeginger. Getting the best response for your erg. *ACM Trans. Algorithms*, 4(3):1–17, 2008.

[66] K. Rege and M. Sengupta. Sojourn time distribution in a multiprogrammed computer system. *AT&T Tech. J.*, 64:1077–1090, 1985.

[67] A. Silberschatz, P. B. Galvin, and G. Gagne. *Operating System Concepts*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[68] M. R. Stan and K. Skadron. Power-aware computing: Guest editorial. *IEEE Computer*, 36(12):35–38, December 2003.

[69] D. Towsley. Queuing network models with state-dependent routing. *J. ACM*, 27(2):323–337, 1980.

[70] D. F. Towsley. *Local balance models of computer systems.* PhD thesis, 1975.

[71] U.S. Environmental Protection Agency. EPA Report on server and data center energy efficiency. 2007.

[72] R. Weber. On optimal assignment of customers to parallel servers. *J. Appl. Prob.*, 15:406–413, 1978.

[73] M. Welsh, D. Culler, and E. Brewer. Seda: an architecture for well-conditioned, scalable internet services. *SIGOPS Oper. Syst. Rev.*, 35(5):230–243, 2001.

[74] W. Whitt. On approximations for queues, I: Extremal distributions. *AT&T Bell Labs Technical Journal*, 63:115–138, 1984.

[75] W. Whitt. Approximations for the $GI/G/m$ queue. *Production and Operations Management*, 2(2):114–161, 1993.

[76] A. Wierman, L. L. H. Andrew, and A. Tang. Power-aware speed scaling in processor sharing systems. *INFOCOM*, 2009.

[77] W. Winston. Optimality of the shortest line discipline. *J. Appl. Prob.*, 14:181–189, 1977.

[78] D. Yao. Refining the diffusion approximation for the $M/G/m$ queue. *Operations Research*, 33:1266–1277, 1985.

[79] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:374, 1995.

[80] S. F. Yashkov. Processor-sharing queues: some progress in analysis. *Queueing Systems Theory Appl.*, 2(1):1–17, 1987.

[81] J. Zhang and B. Zwart. Steady state approximations of limited processor sharing queues in heavy traffic. *Submitted for publication.*

[82] A. P. Zwart and O. J. Boxma. Sojourn time asymptotics in the $M/G/1$ processor sharing queue. *Queueing Systems Theory Appl.*, 35(1-4):141–166, 2000.