Resource Provisioning for Web Applications under Time-varying Traffic

by

Ali Rajabi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2016

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Cloud computing has gained considerable popularity in recent years. In this paradigm, an organization, referred to as a subscriber, acquires resources from an infrastructure provider to deploy its applications and pays for these resources on a pay-as-you-go basis. Typically, an infrastructure provider charges a subscriber based on resource level and duration of usage. From the subscriber's perspective, it is desirable to acquire enough capacity to provide an acceptable quality of service while minimizing the cost. A key indicator of quality of service is response time. In this thesis, we use performance models based on queueing theory to determine the required capacity to meet a performance target given by Pr[response time $\leq x$] $\geq \beta$.

We first consider the case where resources are obtained from an infrastructure provider for a time period of one hour. This is compatible with the pricing policy of major infrastructure providers where instance usage is charged on an hourly basis. Over such a time period, web application traffic exhibits time-varying behavior. A conventional traffic model such as Poisson process does not capture this characteristic. The Markov-modulated Poisson process (MMPP), on the other hand, is capable of modeling such behavior. In our investigation of MMPP as a traffic model, an available workload generator is extended to produce a synthetic trace of job arrivals with a controlled level of time-variation, and an MMPP is fitted to the synthetic trace. The effectiveness of MMPP is evaluated by comparing the performance results through simulation, using as input the synthetic trace and job arrivals generated by the fitted MMPP.

Queueing models with MMPP arrival process are then developed to determine the required capacity to meet a performance target over a one-hour time interval. Specifically, results on response time distribution are used in an optimization to obtain estimates of the required capacity. Two models are of interest to our investigation: a single-server model and a two-stage tandem queue. For both models, it is assumed that service time is represented by a phase-type (PH) dis-

tribution and queueing discipline is FCFS. The single-server model is therefore the MMPP/PH/1 (FCFS) model. Analytic results for time-dependent response time distribution of this model are first obtained. Computation of numerical results, however, is very costly. Through numerical examples, it is found that steady-state results are a good approximation for a time interval of one hour; the computation requirement is also significantly lower. Steady-state results are then used to determine the required capacity. The effectiveness of this model in terms of predicting the required capacity to meet the performance target is evaluated using an experimental system based on the TPC-W benchmark. Results on the impact of MMPP parameters on the required capacity are also presented. The second model is a two-stage tandem queue. The accuracy of the required capacity obtained via steady-state analysis is also evaluated using the TPC-W benchmark.

We next consider the case where the infrastructure provider uses a time unit (TU) of less than one hour for charging of resource usage. We focus on scenarios where TU is comparable to the average sojourn time in an MMPP state. A one-hour operation interval is divided into a number of service intervals, each having the length one TU. At the beginning of each service interval, an estimate of the arrival rate is used as input to the M/PH/1 (FCFS) model to determine the required capacity to meet the performance target over the upcoming service interval; three heuristic algorithms are developed to estimate the arrival rate. The merit of this strategy, in terms of meeting the performance target over the operation interval and savings in capacity when compared to that determined by the single-server model, is investigated using the TPC-W benchmark.

Acknowledgements

First and foremost, I thank my supervisor, Professor Johnny Wong. Without Johnny's advice and encouragement, this work would not have been made possible. His patience was a source of inspiration for me. I learned so much from him, and I will always be grateful to have the opportunity to be his student.

I want to appreciate Professor Ahmed Kamal, Professor Gordon Agnew, Professor Peter Forsyth, and Professor Ken Salem for serving on my doctoral committee and providing me with invaluable comments and recommendations. I would like to thank all the professors, staff and students, who make University of Waterloo such a warm and welcoming environment.

I also want to thank my undergraduate advisor, Professor Ahmad Khonsari. He introduced me to the world of research, and set me on the path of pursuing my dreams. Over the course of my study, I again and again recalled his words and advice, and I am very grateful to him.

This brings me to my family. My most profound respect and gratitude goes to my mother, Rouhangiz Mohammad Beigy, and my father Mohammad Kazem Rajabi. I would not be who I am if it was not for their unconditional love and support. My brother, Mahdi, is about the best brother one could possibly ask for. It would not be an exaggeration to claim I have the most supportive extended family, and I would like to recognize each and everyone of them: My aunts, Young Mi, Farahnaz, Mehrangiz (yoo-yoo) and Sima, my uncles, Rasoul and Naser, my cousins, Sevda, Sina and Sahand, and Ali Ajorloo and Rasoul Hosseinpour whom I have always regarded as my uncles.

A Ph.D. program can occasionally get intense. This is why having supportive friends is invaluable. Among others, I acknowledge and thank my dear friends, Leticia Muñoz, Melanie McDougall, Audrey Bothwell, Sarah Rogalla, Emily Pittman, Krzysztof Borowska and Daniel Kozimor for their friendship and support.

Dedication

To my mother and father for their unconditional love and support

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many organizations offer services over the Internet. These services are provided through applications that are deployed in a datacenter consisting of hardware and software resources. To construct a datacenter to host these applications, one of the challenges an organization faces is the sizing of the datacenter, i.e., the amount of processing power, memory, storage and network bandwidth required. Since the future resource demand by the applications cannot be precisely determined, the organization may set up a larger datacenter than necessary. This may result in low resource utilization. On the other hand, a datacenter may be configured with insufficient resources. Consequently, users of the applications may experience poor quality of service. In addition to sizing risks, setting up a datacenter typically requires an up-front capital investment. Also, operation and maintenance of the datacenter is costly due to power consumption, technical staff and equipment upgrade in response to changes in workload or aging of equipment.

Cloud computing is a model of computing that has the potential to address these challenges [16]. In this model, an organization, referred to as a *subscriber*, acquires resources from an *infrastructure provider* through the Internet to deploy its applications, and pays for these resources on

a pay-as-you-go basis. This is similar to the manner a utility such as electricity is delivered and billed. The resource level can be adjusted in near real-time on subscriber's demand; this feature is called *resource elasticity* [16]. The benefits of cloud computing to subscribers are evident: it eliminates the up-front capital investment and the operational and maintenance costs, and allows the subscriber to adjust the resource level as demand changes.

Infrastructure providers are typically well-established companies such as Amazon, Microsoft and Google. They have been operating large-scale datacenters for their own purposes long before providing computing resources for public use. They have the hardware, software, technical support and administrative infrastructures required to serve as a provider. The expected cost of becoming an infrastructure provider for such companies is not excessive compared to the revenue it brings in [16]. In addition, studies have shown that hardware, network bandwidth and power can be purchased five to seven times cheaper for large datacenters (tens of thousands of computers) than for medium-size datacenters (hundreds or thousands of computers) [49]. The economy of scale due to bulk purchasing allows large providers to offer service at a reasonably cheap level, and still make profit [16].

Virtualization technologies have played an important role in the evolution of cloud computing. These technologies allow multiple virtual machines (VM's) to share the capacity of a physical machine. In a virtualized datacenter, a physical server may be shared by two or more applications, each running on its own VM's. Higher resource utilization can potentially be achieved by allocating fractions of the capacity of an underutilized physical server to other VM's. Examples of virtualization technologies include Oracle VirtualBox [6], VMware [8] and Xen [22].

In this thesis, we focus on the problem of resource provisioning from the perspective of a subscriber. We consider a scenario where a subscriber offers services to its users by deploying transactional web applications on resources acquired from an infrastructure provider. Sufficient

resources should be obtained to provide the users with an acceptable quality of service. In our investigation, we develop performance models of web applications, derive analytic results for these models, and use the results to determine the amount of resources required.

Analytic modeling is a solution technique that allows us to write a functional relation between system parameters and a chosen performance metric in terms of equations that are analytically or numerically solvable [62]. Among mathematical disciplines that have been employed in the context of analytic modeling, we adopt queueing theory due to its ability to study a broad range of computing environments. Queueing models have the potential to provide relatively high accuracy at relatively low cost [67]. Such models allow us to predict the end-to-end delays experienced by the users under different workload intensities [34]. In our work, queueing theory is used to estimate the resource requirement of an application, and the accuracy of these estimates is evaluated by comparison against measurement data obtained from an experimental system.

The system under consideration is depicted in Figure 1.1. It features an infrastructure provider, a subscriber and users of the subscriber's service. The infrastructure provider operates a datacenter which hosts the computing resources. These resources are offered in terms of VM's with various capacities. The capacity of a VM is usually specified in terms of the amount of CPU, memory and storage allocated to the VM. A VM is priced for rent according to the infrastructure provider's pricing policy which is typically based on VM capacity and duration of usage.

Subscriber's service is a three-tier web application consisting of a web tier to generate the user interface, an application tier to implement the business logic, and a database tier for persistent storage. These three tiers are deployed across two servers, namely, a front-end web/application server (referred to as web server from here on) and a back-end database server. In a typical configuration [89, 106], the subscriber acquires a VM for each server from the infrastructure

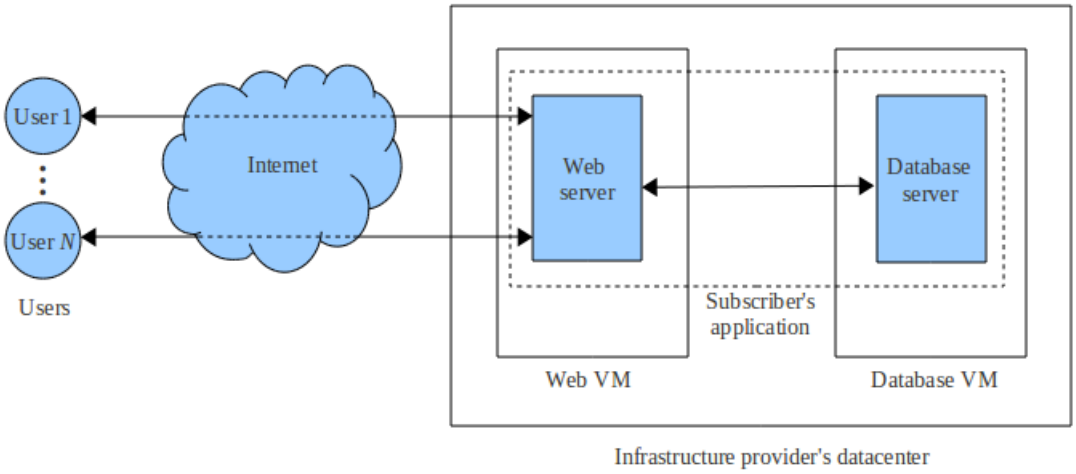

Figure 1.1: System specification

provider; this configuration is referred to as a two-tier architecture. A user of the service starts a session, sends a sequence of logically-related HTTP requests to the web application and receives an HTML response for each request. There is a think time between when a response is received and when the next request is submitted by this user. The number of users varies with time as new users arrive and existing users leave.

Figure 1.2 illustrates the processing of an HTTP request (or job) by the application. In this figure, a user is represented by a session. Jobs submitted by the sessions are processed by the VM's that host the web and database servers, referred to as the web VM and the database VM, respectively. Each job makes an initial visit to the web VM. After finishing service at the web VM, it may go through several cycles of visiting the database VM followed by a return visit to the web VM. Each cycle corresponds to the web server making a call to the database server. A job leaves the system from the web VM; this corresponds to returning an HTML response to the session that submitted the job. The processing of a job within a VM involves CPU executions

and I/O operations. The details of the scheduling disciplines applied to the CPU and I/O devices are out of the scope of this investigation.

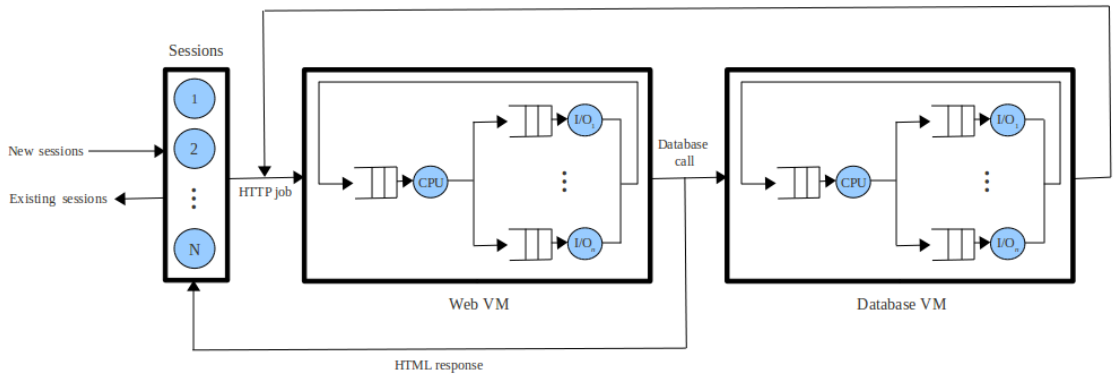

Figure 1.2: Job processing by the system

Response time has a significant impact on the success of a transactional web application; it is defined as the elapsed time from when a job arrives at the web application to when the corresponding response is returned. According to [105], 75% of visitors to a slow e-commerce website will never shop on that site again, resulting in loss of potential revenue. As another example, Amazon sustained a 1% drop in sales for a 100 millisecond extra delay in page generation [71]. Therefore, a subscriber typically enforces stringent constraints on the mean or a high-percentile of the response time. In our work, we assume that the subscriber's objective is to meet a *performance target* in the form of $\Pr[\text{response time} \leq x] \geq \beta$ over all the jobs processed by the system; x and β are referred to as *response time threshold* and *target probability*, respectively. Such performance target is preferred over the mean response time as the latter is not adequate for many applications [97], especially when the response times are widely dispersed around the mean.

Response time is affected by the capacities of the web and database VM's. In order to deter-

mine these VM capacities such that Pr[response time $\leq x$] $\geq \beta$, the response time distribution is required. A straightforward solution method is to develop a comprehensive queueing model for the system in Figure 1.2 that features session-based workload and detailed processing of jobs by the two VM's. This model, however, is complex and analytic results for response time distribution are difficult to obtain. Our approach is therefore to approximate this model by two simpler models:

1. A traffic model in the form of an arrival process to represent jobs submitted by all the sessions; this model is discussed in Section 1.1.

2. A queueing model for the overall system consisting of the web and database VM's with the job arrival process developed in Step 1.

For the second model, the derivation of response time distribution is mathematically tractable.

We now present the research problems investigated in this thesis. These problems are closely related to the two models defined above, and include the development of a traffic model for web applications and the investigation of resource provisioning strategies using queueing theory.

1.1 Traffic model

The first research problem is concerned with the development of a traffic model for jobs submitted by user sessions. Studies have shown that the job arrival process to a web application is time-varying with intervals of high demand intervening periods of moderate load [35, 73, 93]. This variation manifests itself over *fine* and *coarse-scale* time intervals. Fine-scale variation is characterized by fluctuation of traffic over time periods of tens of seconds. Evidence of such variation is available in [72], where the number of job arrivals per second at an online book-store was monitored for an hour. This fluctuation is induced by variation in session length (number

of jobs in a session), session think time and service time [63, 72, 94]. Coarse-scale variation, on the other hand, is attributed to phenomena such as time-of-day or seasonal effect [86, 88]. For example, there is a noticeable surge in workload level during specific times of a day.

We focus on fine-scale time-varying traffic which is consistent with session-based job arrivals [63, 72, 94]. A traffic model with the potential to capture fine-scale variation is Markov-modulated Poisson process (MMPP); MMPP is a generalization of the Poisson process where the job arrival rate changes with time. We use the following procedure to evaluate the effectiveness of this model. An available tool called BURN [27] is extended to generate a synthetic trace of job arrivals from user sessions. An MMPP is fit to the synthetic trace. This MMPP is then employed to generate a second trace of job arrivals, referred to as the MMPP trace. A simulation program is used to compare the performance results when jobs are submitted in accordance with (*i*) the synthetic trace and (*ii*) the MMPP trace. We use a synthetic trace instead of a real trace available in the public domain because the synthetic trace tool can be tuned to generate traces with controlled levels of time-variation.

1.2 Resource provisioning strategies

The subscriber's resource provisioning strategy is affected by the infrastructure provider's pricing policy. Typically, an infrastructure provider specifies a minimum time unit (TU) for charging of resource usage. A resource is billed for the number of TU's held by a subscriber; a partial TU is billed as a full TU. For example, infrastructure providers such as Amazon EC2 [1], Microsoft Azure [2] and Google App Engine [5] charge for instance usage on an hourly basis, i.e., TU is one hour.

From the subscriber's perspective, resource provisioning can be organized into *service intervals* of length one TU as shown in Figure 1.3. At the beginning of a service interval, referred to as

a *decision point*, the amount of resources required for the upcoming service interval is estimated and requested from the infrastructure provider. The resource level remains constant during this service interval. With time-varying traffic, the queue size at the web and database VM's grows when traffic level is high and shrinks when it is moderate. The objective is to acquire sufficient resources to meet the performance target when one considers all jobs processed during the service interval, while keeping the cost at a minimum.

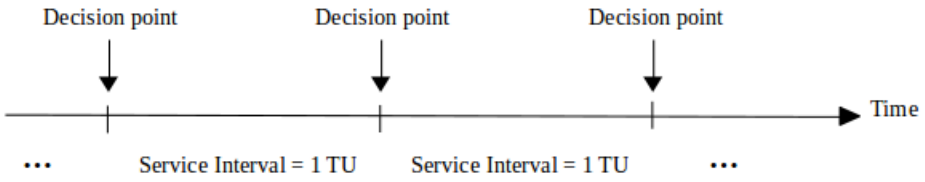

Figure 1.3: Service interval

The following methodology is used to determine the required capacities. A queueing model for the system under investigation is solved to obtain the cumulative distribution function (CDF) of response time, which is used to compute Pr[response time $\leq x$]. This probability is affected by the capacities[1] of the web and database VM's. We use the response time analysis in an optimization problem to determine the VM capacities with minimal cost such that Pr[response time $\leq x$] $\geq \beta$ over a service interval. This optimization problem is solved using a two-dimensional search over the space of available VM capacities for the web and database servers.

For the system in Figure 1.2, a commonly used queueing model is an open tandem queue with two servers, representing the web and database VM's (see for example [28, 103, 106]). This model is depicted in Figure 1.4. In our work, the arrival process is assumed to be MMPP. As to service

[1]In our investigation, the term "capacity" means the size of a VM, i.e., the amount of CPU, memory and storage.

time, web applications typically provide a variety of services to their users, each corresponding to a transaction type. The service requirement is usually variable across these transaction types, with some transactions placing a higher demand on system resources than others [84]. It has been shown that the hyper-Erlang (HE) distribution is sufficiently accurate in capturing such variability in service time [27, 78]. HE is a combination of one or more Erlang distributions, each having a different mean and variance; it is a special case of the phase-type (PH) distribution [62]. In our investigation, we assume that the job service time at a server follows a PH distribution. To obtain analytic results for the response time distribution, one needs to specify the scheduling discipline. For mathematical tractability, we assume that scheduling discipline at each server is FCFS.

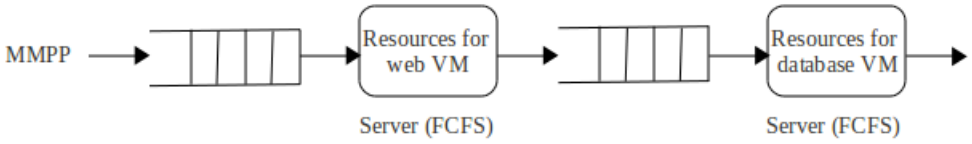

Figure 1.4: Two-stage tandem queue

For the two-stage tandem queue shown in Figure 1.4, the numerical computation of response time CDF is expensive. In addition, this model presumes the availability of service time distribution at each stage, which can be obtained using application instrumentation, operating system logs, or third-party monitoring tools such as HP (Mercury) Diagnostics [28]. This, however, requires technical knowledge at the application, operating system or tool level; such knowledge may not be readily available. The above observations motivate us to consider a simplified model where the resources obtained for the web and database VM's are treated as a "*black-box*"; this black-box is then modeled as a single server with FCFS scheduling discipline. The service time of a job at the black-box is the same as the response time when there is no queueing, and can be

readily obtained by recording the arrival and departure times of jobs processed by the system. As mentioned previously, the distribution of service time is PH. We thus have an MMPP/PH/1 (FCFS) queue. This single-server model is shown in Figure 1.5.

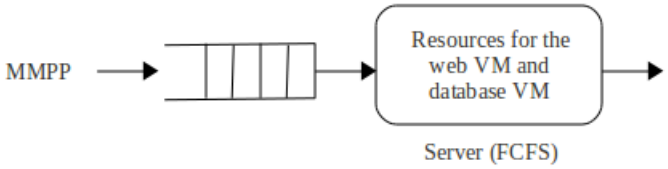

Figure 1.5: Single-server model

Both the single-server model and the two-stage tandem queue are considered in our investigation. For both models, the response time CDF is used in an optimization problem to determine the required capacities. This CDF is time-dependent initially, referred to as transient phase. The transient phase is followed by steady-state behavior where the CDF becomes unchanging in time. The computation of time-dependent CDF is very expensive compared to the steady-state results. Consider the single-server model as an example. We derive analytic results for the time-dependent response time CDF of the MMPP/PH/1 (FCFS) queue; these results are in the form of a system of partial differential equations that can be solved numerically. For parameter values representative of those used in our experiments, a solution for this system took several months for a time interval of 200 seconds. We conclude that it is not practical to use time-dependent analysis for resource provisioning purposes. The steady-state results, however, can be used as an approximation if the length of transient phase is small compared to that of the service interval.

We first consider a case where transient phase is sufficiently small so that the steady-state results can be used. Our numerical examples for time-dependent response time CDF of the single-server model reveal that for parameter values representative of those used in our experiments, steady-state results are a good approximation for a one-hour TU (or service interval), as defined

by infrastructure providers such as Amazon, Microsoft and Google; such a TU value is of interest to this investigation.

Two research problems are investigated. In these problems, we use the single-server model and two-stage tandem queue, respectively, to obtain estimates of the required capacity, and evaluate the accuracy of these estimates using an experimental system.

Single-server model

The first research problem is related to the application of the single-server model in resource provisioning. Analytic results for steady-state response time CDF of the MMPP/PH/1 (FCFS) queue have been reported in [42, 76]. Instead of using these results, we develop an alternative derivation based on Markov chain analysis. Our derivation is more suitable for numerical computation. As mentioned earlier, the response time CDF is used in an optimization problem to determine the required capacities. The solution to this problem is only an estimate of the actual capacities required because of the assumptions made over the course of model development. To evaluate the accuracy of these estimates, we compare the results obtained from the model against measurement data from an experimental system. In our experimental system, resources are offered in terms of VM's with different capacities. An existing implementation of the TPC-W benchmark [3] is deployed over two VM's on this system, hosting the web and database servers, respectively. TPC-W is a specification for an online bookstore and is commonly used as a benchmark in evaluation of resource provisioning strategies for web applications.

Our evaluation procedure is as follows. For a given MMPP and performance target, the required VM capacities are determined by the optimization problem. The web and database servers are deployed over VM's with these capacities. Jobs are then submitted to the web server in accordance with the same MMPP for one service interval, and the response times of these jobs

are measured. The fraction of jobs that meet the response time threshold x is computed; let β' denote this fraction. The metric $e = \beta' - \beta$ indicates the effectiveness of the model. A value of zero for e means the estimated capacities are precisely on target. A positive value indicates over-provisioning of capacities, while a negative value implies that the resources are not adequate. Results showing the values of e for a range of input parameter values are presented.

Two-stage tandem queue

The second research problem pertains to the application of the two-stage tandem queue in resource provisioning. This model is a more detailed representation of the system than the single-server model. However, the numerical computation of steady-state response time CDF for this model is very demanding. In addition, the model parameterization requires expertise in application instrumentation, operating system or third-party monitoring tools. On the other hand, it can be mathematically proved that under a modest set of conditions, the capacity estimates from this model are lower than those of the single-server model in order to meet the same performance target. There is therefore a trade-off between the additional cost in model parameterization and computation and potentially better resource estimates.

As for the single-server model, the response time analysis of the two-stage tandem queue is used in an optimization problem to determine the required capacities. Steady-state results for the response time CDF are available in [52, 96]. Again, we present an alternative derivation which is more suitable for numerical computation. This derivation is an extension of our analysis for the single-server model. As mentioned previously, solutions to the optimization problem are estimates of the actual capacities required because of the assumptions made in model development. The objective of the second research problem is to experimentally assess the improvement in resource estimate, if any, in view of the increased computation cost.

Our methodology to evaluate the accuracy of the resource estimate is the same as that for the single-server model. Briefly, an implementation of TPC-W benchmark is deployed over two VM's with different capacities; the capacities of these VM's are determined using results for the two-stage tandem queue. Jobs are submitted to the web server in accordance with an MMPP, and the fraction of response times meeting the performance target, denoted by β', is measured. The difference between β' and the target probability β is used to assess the accuracy of the estimates.

Our experimental results suggest that the two-stage tandem queue often underestimates the capacities required to meet the performance target. In addition, the computation cost is significantly higher than that for the single-server model. Based on these observations, we conclude that the single-server model is the preferred model for resource provisioning. We therefore focus on the single-server model in the rest of our investigation. This completes the description of the two research problems.

We next consider the conceivable scenario where an infrastructure provider uses a smaller TU than one hour for pricing purposes in the future. This leads to a shorter service interval between decision points. The subscriber is then able to make more frequent adjustments to the resource level; this may result in reduction in resource provisioning cost. Our objective is to develop resource provisioning strategies that can be used by the subscriber to achieve cost reduction. Such strategies should be of interest to the subscriber.

In our investigation, job arrivals are assumed to follow an MMPP. We first note that if steady-state results for the MMPP/PH/1 (FCFS) model are used for resource provisioning, the capacity estimates for each service interval are the same regardless of the length of the service interval (or the value of TU). This approach, therefore, does not lead to any savings in cost. However, for sufficiently short TU and under time-varying traffic, the job arrival rates at various service intervals could be quite different. If resource provisioning is based on the estimates of these arrival

rates, there may be opportunities for cost savings. As an example, suppose the arrival process is a two-state MMPP, and TU has length comparable to the average time the MMPP spends in one of these states (referred to as the sojourn time). A promising strategy is then to obtain an estimate of MMPP state for the upcoming service interval and acquire resources based on the arrival rate at this state. Specifically, more resources are acquired when the system is subject to high arrival rate, and unnecessary resources are released when the arrival rate is low. Since the capacities are different across service intervals, there is a potential to reduce the long-term cost of resource provisioning. The case of small TU's is of interest to our investigation.

We now describe our resource provisioning methodology for the case of small TU's (or small service intervals). Our approach is to determine, at each decision point, the resource requirement such that the performance target is met over the upcoming service interval. Again, analytic results for the response time CDF of the single-server model are used in an optimization to determine the required capacities. An important consideration is the assumption regarding the job arrival process. Each MMPP state is associated with a Poisson process. Since TU is comparable to the average sojourn time in an MMPP state, a natural choice for the arrival process is Poisson. As to the service time, it is assumed that any capacity adjustment at a decision point will take effect immediately, and the service time is modeled using a PH distribution. We thus have an M/PH/1 (FCFS) model. We recognize that with a short service interval, the length of transient phase may not be small compared to that of the service interval; therefore, steady-state results may not be a good approximation. The computation of time-dependent results, however, is prohibitively expensive. We therefore use steady-state results as a heuristic.

Our methodology requires an estimate of the job arrival rate for each service interval. Three algorithms for estimating this arrival rate are investigated. The first two algorithms (Algorithms 1 and 2) take advantage of advance knowledge that input traffic is MMPP in the sense that the

arrival rate can take on a limited number of values corresponding to the various MMPP states. Our objective is to gain insight into whether or not such knowledge would result in effective resource provisioning. For Algorithm 1, the arrival rate estimate is based on the number of arrivals over the last service interval, while for Algorithm 2, the estimate is based on whether or not the performance target is met over last service interval. For the third algorithm (Algorithm 3), the measured arrival rate over the previous service interval is taken as the estimate of the arrival rate for the upcoming service interval.

We compare the performance of the three algorithms in terms of the estimated required capacity and whether or not the performance target is met. With a small TU (or service interval), it is sensible to evaluate the effectiveness of our methodology in terms of meeting the performance target over a time interval spanning multiple service intervals; to facilitate the comparison of results against those obtained from the MMPP/PH/1 (FCFS) model, we set the length of this time interval to one hour. The evaluation procedure is as follows. A time interval of length one hour is divided into a number of service intervals, each having the length equal to one TU. To reduce the computation overhead at a decision point, the capacity requirement for different arrival rates are precomputed (using steady-state results of the M/PH/1 (FCFS) model) and stored in a table. Jobs are submitted to the web server in accordance with an MMPP. At each decision point, the job arrival rate is estimated using the algorithm being evaluated, and the corresponding required capacities are looked up from the table. The capacities of the web and database VM's are then adjusted accordingly. At the end of the one hour interval, the fraction of jobs meeting the performance target over the entire interval is measured; let this fraction be β'. The difference $e = \beta' - \beta$ is used to evaluate the effectiveness of the three algorithms. We also compare the performance of these algorithms against the earlier case where resources are provisioned for a one-hour TU using the MMPP/PH/1 (FCFS) model; this allows us to assess whether or not the above algorithms, designed for a small TU, would result in cost saving.

1.3 Thesis organization

The remainder of this thesis is organized as follows. In Chapter 2, we present the related work on workload characterization of web applications and resource provisioning methodologies in cloud computing environment. In Chapter 3, we discuss our methodology for characterizing job arrivals by an MMPP and evaluate the effectiveness of MMPP in modeling time-varying traffic. Chapter 4 is concerned with the transient results for the MMPP/PH/1 (FCFS) model. In Chapters 5 and 6, we present our resource provisioning methodologies based on the single-server model and two-stage tandem queue, respectively, and discuss the corresponding results on their effectiveness in estimating the required capacity. Chapter 7 is concerned with resource provisioning for small service intervals. Results on the merit of the three algorithms for estimating arrival rates are presented. Finally, in Chapter 8, we summarize our findings and discuss potential research problems for future work.

Chapter 2

Related work

The application of analytic modeling in resource provisioning for web applications has been the topic of many studies. Of interest to our investigation are the following two issues: the characterization of the job arrival process, and the development of analytic models and the application of these models to resource provisioning in cloud computing. In this chapter, we present a survey of related work on these two issues.

2.1 Job arrival process

As mentioned in Chapter 1, web application workload is time-varying, and our focus is on fine-scale time-variation. One way to quantify this type of variation is burstiness in job arrivals. In a bursty arrival process, there is a correlation between the number of job arrivals in successive time intervals. Burstiness has a significant impact on system performance. Therefore, concerted effort has been dedicated to developing bursty workload models for web applications. The results are typically used as input to a performance model of web applications for resource provisioning purposes, or to generate synthetic workloads to evaluate the performance of a web application under

a variety of workload conditions. In this section, we present the related work on characterization of the job arrival process.

In [72], e-business traffic has been shown to exhibit both short and long-range dependence. Statistical data obtained from monitoring an online bookstore and an auction website suggest a strong correlation in the arrival stream of HTTP requests. As discussed in [72], detecting such dependencies would allow one to not only study the mean behavior, but also predict the phases of increase or decrease in the volume of requests, i.e., burstiness in traffic.

Several methods have been proposed to model burstiness in web application traffic. In [64], a session-based model where new sessions are created in accordance with a Poisson process is considered. Each session consists of a number of logically-related requests (or jobs). It is observed that high variability in session lengths (the number of jobs in a session) and think times will cause burstiness in job arrivals. A heavy-tail distribution such as bounded Pareto is used to model the session length and think time distributions.

Casale et al [27] consider the problem of generating web application workload with a given degree of burstiness. A metric called overdemand is defined to quantify the level of burstiness. Given a set of available benchmarks (such as TPC-W [24, 44] or RUBiS [4]), a linear programming problem is formulated to compute the policy on how these benchmarks should be mixed in order to achieve a given overdemand in the resulting workload. In [26], the index of dispersion, as opposed to overdemand, is used as the measure of burstiness; this results in a non-linear optimization problem instead of a linear programming problem.

Krishnamurthy et al [63] propose a two-step algorithm to generate job arrivals to a web application. First, a trace of sessionlets (a sessionlet is a logically-related sequence of request types within a session) is used to generate an intermediate trace of sessionlets that matches a given workload mix. Secondly, the intermediate trace is transformed into the final trace of

sessions using system-specific URL generation rules and the think time distribution. This final trace, along with session inter-arrival times, are then used as input to httperf tool [74] to generate a sequence of web requests.

The work presented in [46] assumes that the job arrival process is periodic and determines the most likely period at which the arrival process is repeated. Two methods are presented to determine this period, namely, Fourier analysis and autocorrelation analysis. The resulting period is then used to generate synthetic traces which are representative of the arrival process.

Renewal models, and in particular the Poisson model, have been widely used to approximate time-varying traffic [23, 30, 35, 55, 58, 106]. In these studies, an operation interval is divided into a number of subintervals, and the traffic during each subinterval is assumed to be Poisson with possibly different rates at different subintervals. The arrival rate for each subinterval is estimated using statistical data collected over previous subintervals or over the same time period in previous days.

2.2 Markov-modulated Poisson process

As mentioned in Chapter 1, the input traffic to the performance models investigated in this thesis is Markov-modulated Poisson process (MMPP). In this section, related work on the application of MMPP in a number of research areas, MMPP fitting algorithms, and queueing models with MMPP arrival process are presented.

MMPP is a generalization of the Poisson process in which the arrival rate varies over time in accordance with a Markov chain. It has been used as a model for job arrivals to queueing systems. In [104], analytic results are presented for a single-server queue where the arrival process is a two-state MMPP and the service time distribution is exponential. Neut [75] considers a queueing system with MMPP arrivals and general service time distribution. The number of

MMPP states is arbitrary, and the service time distribution may be different in different MMPP states. The transient behaviour of a finite buffer queue with MMPP arrivals and general service time distribution has been presented in [31].

Steady-state analysis of a queueing system with MMPP arrivals and general service time distribution can be found in [42]. This paper presents a survey of different aspects of MMPP arrival streams, queueing models with MMPP job arrivals, and applications of MMPP. It also provides references on MMPP fitting algorithms. Approximation of a queue with MMPP arrivals and exponential service time distribution by an M/M/1 queue can be found in [32].

Several algorithms have been proposed to determine the parameters of an MMPP using measurement data. In [51], an algorithm is presented to determine the parameters of a two-state MMPP from the mean, index of dispersion, and the third moment of the inter-arrival time. This work has been generalized to an arbitrary number of states in [83]. An efficient method to estimate the parameters of an MMPP using the expectation maximization technique has been reported in [36]. In [21], a heuristic method to estimate the parameters of an MMPP with an arbitrary number of states is presented. This heuristic divides the observed arrival rates into a number of regions where each region corresponds to an MMPP state.

Results on queueing network models with MMPP arrivals have been reported in [52, 53]. Each node in the network is a single server queue with general service time distribution. The number of buffers may be finite or infinite. For the case of a two-state MMPP, a decomposition technique is used to obtain analytic results for the performance metrics of interest. In [96], the response time distribution of a two-stage tandem queue with MMPP arrivals is derived using a method based on the age of the leading job at the first stage.

MMPP has been used in a number of research areas to model time-varying traffic. In the area of web applications, Andersson et al use MMPP to model input traffic to an Apache web server

[13]. HTTP requests are generated in accordance with an MMPP. Recently, Escheikh et al [40] use MMPP to model time-varying traffic at virtualized server clusters.

The work in [50] is concerned with the characterization of time-varying workload experienced by a networked storage system comprising of traditional hard disk drives as well as flash-based storage. A hidden Markov model is constructed and parameterized using data available in both synthetic and real traces. Since a hidden Markov model can be mapped to an MMPP, the results in [50] can be used to estimate MMPP parameters.

In [102], message arrivals from a source node in a torus interconnection network are modeled by MMPP. It is argued that the Poisson process cannot model the inherent burstiness in the arrival stream. MMPP has also been used in the modeling of voice traffic in a data network [51]. In this work, the superposition of multiple voice sources is approximated by a two-state MMPP. A similar approach has been used to model the superposition of multiple arrival streams at an asynchronous transfer model (ATM) switch [42]. In addition, MMPP has been used to model overflow from a finite trunk group [42]. Other works include the use of MMPP to model user arrivals to a public wireless LAN [20], the bursty nature of arrival streams to an ISDN channel [79], and input traffic to an ATM network [61].

2.3 System modeling and resource provisioning

Resource provisioning for web applications in cloud computing environment has received considerable attention in recent years. Studies in this area generally fall into two categories. The first category considers resource provisioning from the perspective of a subscriber. In this case, the subscriber aims to determine the smallest amount of resources required by an application in order to meet a given performance target; this should result in minimum cost when resources are requested from an infrastructure provider. The second category investigates resource pro-

visioning from the perspective of an infrastructure provider. An infrastructure provider has a pool of computing resources which is available to many subscribers. These resources are allocated to subscribers in order to meet the infrastructure provider's objective, e.g., maximizing the resource utilization and revenue and minimizing the energy consumption. Our focus is on resource provisioning from the perspective of a subscriber. However, performance models of web applications, as well as resource provisioning strategies that may be of interest to subscribers, are often included in studies pertaining to infrastructure providers. For this reason, we also present related work on resource provisioning from an infrastructure provider's perspective.

In [84], Sharma et al present a model of a multi-tier web application based on a tandem network of M/G/1 queues with processor sharing (PS) discipline. The service time distribution is a mix of shifted exponentials. An approximation for the end-to-end response time distribution is derived. The results are then used in an optimization problem to determine the number of (possibly heterogeneous) servers for each application tier in order to meet a performance target based on response time percentile. The TPC-W benchmark is used to evaluate the accuracy of the model.

A capacity provisioning scheme for a multi-tier web application is presented in [86]. Each tier is modeled by a G/G/1 queue with FCFS discipline; different request classes are represented by a general service time distribution. Analytic results are used to determine the number of servers for each tier to meet a tier-specific performance target based on response time percentile.

In [33], a queueing network model for a multi-tier web application is considered. A load balancer is included as a separate tier and modeled by an M/G/1 queue with FCFS discipline. The application server consists of CPU, cache and memory subsystems. Each of these subsystems, as well as the database server, is also modeled by an M/G/1 (FCFS) queue. Results from the overall network model are used in an optimization problem to determine the number of servers

allocated to the application and database tiers to meet a performance target based on response time percentile. The accuracy of this model is evaluated using the RUBiS benchmark.

Gandhi et al [43] model an application by a queueing network with different types of requests. Each tier is modeled by an M/G/1 queue with PS discipline; it may be replicated across multiple homogeneous servers. A Kalman filter is used to determine the values of input parameters that cannot be directly measured. Results for tier utilization and end-to-end average response time of each request type are obtained. These results are then used to determine the number of servers at each tier such that a performance target based on average response time is met.

A dynamic strategy to provision capacities for a two-tier web application is presented in [81]. The application is modeled by a queueing network with two nodes, each node consisting of CPU and I/O servers. User sessions are represented by delay stations. The number of sessions for the next operation interval is estimated using the autoregressive moving average method. A resource provisioning algorithm which uses this estimate as input is presented. This algorithm starts by assigning a server to each tier, and on each iteration, the bottleneck tier is identified and replicated. An optimization problem based on receding horizon control is formulated to determine the number of servers for each tier to meet a mean response time constraint while minimizing provisioning and reconfiguration costs.

In [103], a web application is modeled by a tandem network where each stage is an M/G/1 queue with PS discipline. This model is used to determine the best performance that can be achieved for a given CPU budget. A two-level controller is developed to determine, in an online manner, the fraction of resources that needs to be allocated to each tier in order to meet a performance target based on mean response time.

Zhang et al [107] consider the problem of dynamically adjusting the number of servers hosting instances of an application in a number of geographically distributed datacenters. Their formula-

tion takes into account such issues as different pricing policies at various datacenters, fluctuating traffic and the cost of server start-up and shut-down. They provide a control-based algorithm which aims at reducing resource requirement and reconfiguration costs. A server is modeled by an M/M/1 queue; this model is used to determine the mean delay experienced by a job. The contention among infrastructure providers in the presence of capacity constraint at each datacenter is represented by a model based on game theory.

Andrew et al [14] consider the problem of dynamically adjusting the server capacity of an M/G/1 queue such that a metric based on a linear combination of average response time and energy cost per user is minimized. They also consider static provisioning where the capacity of the server is fixed over the entire operation interval. The scheduling disciplines examined are PS and shortest remaining processing time. The merit of each discipline is assessed by comparing the above-mentioned metric for this discipline against that obtained from an optimal offline algorithm. Both worst case and stochastic analyses are presented.

The work in [68] is also concerned with dynamic resource provisioning. The objective is to dynamically adjust the number of active servers in a datacenter to meet an application service level agreement (SLA) while minimizing the operating and switching costs; switching cost is incurred when there is a need to wake up a server that has been put into power saving mode. Each server is modeled by an M/G/1 queue, and power consumption is represented by a linear function of the arrival rate. An optimization problem is formulated, and an algorithm is presented to find the optimal online solution, assuming that the load parameters are known within a given time window into the future. The performance of this algorithm is compared against an optimal offline solution.

In [41], a layered queueing network is used to determine the latency of an application deployed over edge and core clouds. The edge cloud is for caching purposes and is typically located within

the same geographical region as the clients using the application. It has limited processing and storage capacities and occasionally needs to communicate with the core cloud to process a request. A distributed application consisting of both streaming and transactional workload is used to evaluate the accuracy of the network model.

Woodside et al [100] use a layered queueing model to determine the number of servers required, as well as the maximum number of threads in each server, such that an SLA based on average response time is met. The average service times at different layers are computed using an extended Kalman filter. This filter continuously compares the measured performance metrics against those predicted by the model, and uses the error as a guide to adjust the value of the model parameters. Once the error is smaller than some given threshold, the model is used to determine the number of servers at different layers.

The work in [56] is concerned with the provisioning of computing resources for a web application comprising of service modules organized as a directed acyclic graph. The SLA is defined for the front-end service, i.e., the one that is directly invoked by a user. Each service module negotiates with the other service modules its estimated performance if it has one more or one less unit of capacity; the performance metric is average response time. The M/M/1 (PS) and M/M/n (PS) models are used for the single-CPU and multi-core configurations, respectively. The front-end service makes the final decision on the resource allocation for each service such that the SLA is met.

Dutta et al [38] investigate the problem of capacity reconfiguration of an application subject to dynamic workload to meet the application demand while reducing the reconfiguration cost. They propose an architecture that utilizes both horizontal and vertical scaling to achieve this objective. In horizontal scaling, a VM is cloned. In vertical scaling, on the other hand, the resource share of a VM is dynamically modified. The effectiveness of the proposed solution is

evaluated using an available social network benchmark.

Zhang et al [106] argue that web application workload is session-based, i.e., the sequence of requests issued by a single user are correlated, with think times intervening the individual requests. Performance studies using queueing models, on the other hand, often assume that successive requests are independent of each other. A technique to convert a session-based request mix to a single request stream is presented. Next, a queueing network model is used to determine the CPU requirement of each tier of a multi-tier application to meet a performance target based on average response time.

The work in [55] is concerned with resource provisioning for two interactive applications. Two allocation strategies are studied: shared allocation in which the two applications are allocated a shared resource pool, and dedicated allocation where each application is allocated its own resources. An M/M/n queue with FCFS discipline is used to compare the performance of the two strategies. A heuristic algorithm to determine the strategy with a smaller number of required servers is also presented.

When resources are provisioned for individual VM's separately, each VM typically operates in capped mode, i.e., it cannot use more resources than its allocation. Meng et al [73] propose a technique to combine multiple VM workloads and provision resources based on an estimate of the aggregate workload. This technique allows a VM to borrow resources from other VM's if these resources are not fully utilized.

The work in [29] investigates the problem of capacity planning for a web application subject to uncertain demand when the infrastructure provider uses both on-demand and reservation-based pricing policies. Reserved resources are offered at a lower price than on-demand resources. Under-subscription of reserved resources can be compensated for by obtaining on-demand resources at an extra cost. Over-subscription of reserved resources, on the other hand, would result in over-

provisioning. An optimization problem based on stochastic integer programming is formulated to obtain the optimal configuration of on-demand and reserved resources.

The work reported in [39] is concerned with the assignment of reserved and shared resources in a datacenter to a number of subscribers. Reserved resources are dedicated to high-priority subscribers, while shared resources can be utilized by all subscribers. An M/M/C/C model with multiple service classes is used to determine the blocking probability of requests initiated by a subscriber. This model is also used for datacenter dimensioning purposes, i.e., determining the size of reserved and shared resource pools.

Ali-Eldin et al [10] propose a hybrid methodology to dynamically adjust the capacity allocated to a web application deployed over the cloud. They consider horizontal elasticity where scale-up and scale-down are performed by adding and removing VM's, respectively. Two adaptive controllers are designed to monitor and maintain a given SLA. A G/G/N queue is used to model the web application.

Sedaghat et al [82] consider the problem of dynamic provisioning of resources for an application deployed over a cloud with multiple VM types. Each VM type is a specific combination of hardware resources such as CPU, memory and storage. They consider a scenario where the ratio of price to performance is more advantageous for larger VM's. Initially, small VM instances are allocated in response to increasing workload. With further load increase, it may become advantageous to reconfigure the total capacity using larger VM's. An integer linear programming problem is formulated to determine the new configuration; the formulation takes into account the reconfiguration cost and its impact on application performance.

The work in [101] is concerned with resource provisioning in a software-as-a-service cloud. The provisioning mechanism takes into account customer profile (such as company size), provisioning cost, dynamic workload, response time SLA violations and availability of heterogeneous resources

in the cloud. Scheduling algorithms that minimize the total cost and the number of performance violations are presented.

The work in [57] is concerned with the problem of just-in-time scalability in cloud environment, i.e., the ability of an application to dynamically expand or shrink its resource requirement in response to changes in workload. A platform-independent profile-based approach to address just-in-time scalability is proposed; the profile contains expert's knowledge of scaling application servers in cloud environment.

Wildstrom et al [98] investigate the conditions under which it is worth obtaining additional resources or releasing extra resources. For a given workload and configuration (in terms of the amount of memory), low level OS statistics are used to compute the difference between the average response time of the current system and that of a system with one more or one less unit of memory. This experiment is repeated for several workloads and system configurations. The WEKA [99] package is then used to learn two regression models. These models are used to estimate the change in value function when the system has one more or one less unit of memory.

Mao et al [70] consider the problem of resource allocation to a sequence of jobs with specified deadlines. The objective is to meet the deadlines with the smallest amount of resources. Their investigation includes such issues as workload fluctuation, provisioning delay from when a resource is requested to when it is ready to use, and prevalent pricing policies where resources are priced on an hourly basis.

Capacity provisioning for a batch service in a federated cloud environment is investigated in [25]. The objective is to use the resources of multiple cloud providers to minimize cost and maintain a good quality of service described in terms of mean or maximum response time and request drop rate. A decentralized self-adapting solution is presented; the effectiveness of this solution is evaluated using simulation.

The work in [85] is concerned with resource provisioning for a batch job executed on a single machine. A number of machine configurations and input data are used to train a multi-variate linear regression model. For a given input size, the resource provisioning algorithm uses the regression model to determine the configuration with minimum completion time.

In [28], a web application is modeled using a two-stage tandem queue where sessions are represented by delay stations. The service times are characterized by a Markovian arrival process (MAP). The two stages of the tandem queue are approximated by a single flow-equivalent server with state-dependent service time distribution; this distribution is also characterized by a MAP. The resulting model is solved to obtain the system throughput and average response time.

A general framework of a control system for autonomic computing is presented in [37]. The control input is a set of controllable parameters (e.g., the number of CPU cycles of a server) that are relevant to a target system (e.g., a web application deployed on a server). The measured output is typically a performance metric, e.g., average response time. The target system is subject to changes in workload and other factors that may affect the measured output. At a decision point, the controller determines the new value of control input based on the difference between the performance target and the measured output.

The work in [11] is concerned with resource allocation and admission control in a virtualized server environment. A VM hosting a web service is modeled by an M/M/1 queue with FCFS discipline. Each subscriber's performance target is expressed in terms of guarantees on the tail of response time distribution. Chebyshev's inequality is used to obtain an upper bound for this probability in terms of the mean and variance of response time. An optimization problem is then formulated to determine the capacity assigned to each VM in order to maximize the infrastructure provider's revenue, minimize its cost and satisfy the performance targets.

Ardagna et al [15] consider the problem of dynamically allocating the resources of a number

of geographically distributed datacenters to multiple web service classes. Resources are obtained for a medium-to-long time scale to account for typical pricing policies adopted by infrastructure providers. Dynamic load redirection is performed on a shorter time scale to balance the load among the different sites. Each VM is modeled by a finite population M/G/1 queue with PS discipline. The performance target is described in terms of bounds on the mean response time. A non-linear optimization problem is formulated to determine the VM capacities. The accuracy of the model is evaluated using a prototype deployed over Amazon EC2.

Goudarzi et al [48] consider the problem of assigning VM's to a number of geographically distributed datacenters. An optimization problem is defined which takes into consideration the heterogeneity of resources, service delays, VM migration costs, peak power capacity and power usage effectiveness of a datacenter. An M/M/1 model is used to obtain an estimate of the service delay.

The problem of assigning VM's to a number of geographically distributed datacenters has also been investigated in [47]. Two classes of SLA's are considered: gold customers are guaranteed a certain quality of service in terms of response time, and bronze customers have a utility function which is based on response time. Application tiers, each hosted by a VM, and request dispatchers are modeled using a network of M/M/1 queues with generalized PS discipline at each queue.

The work in [65] is concerned with the allocation of resources of a datacenter to multiple subscribers in the presence of workload variations and computer failures. The problem is formulated as a sequential optimization problem and solved using a limited lookahead control approach. Three customer classes, namely, gold, silver and bronze are defined. The SLA adopted for each class is represented by a stepwise non-linear graph. The use of limited lookahead control in sequential optimization for resource provisioning has also been reported in [66]. In this work, an optimization problem is formulated to minimize the power cost as well as penalties incurred in

case of SLA violations.

Addis et al [9] propose a hierarchical distributed algorithm to allocate the resources in a datacenter among competing applications. A mixed-integer non-linear optimization problem is formulated to maximize availability, minimize energy cost and meet the SLA of individual applications. The SLA is represented by a linear function describing the revenue and penalty for a given average response time. A network of M/G/1 queues is used to determine the average response times.

The work in [58] is concerned with allocating the available resources of a datacenter to heterogeneous applications. A technique based on layered queueing networks combined with discrete gradient search and bin packing optimization is used to compute the optimal configuration for a number of workloads. The application of layered queueing models in cloud-based resource management has also been reported in [19] where urgent and dynamic customers are considered. Urgent customers are those demanding resources at short notice, while dynamic customers must adapt to frequent changes in workload, system configuration or cloud server availability.

Bennani et al [23] adopt a multiclass queueing network model and a local beam search to optimally allocate the available resources of a datacenter to competing applications. Their approach considers both CPU and I/O resources and supports multiple transaction types for each application.

In [30], the problem of resource allocation to multiple applications is considered. A system of queues (one for each application) governed by generalized PS discipline is used to characterize the relationship between the time-dependent workload of each application and the required resource level to provide a desired quality of service. A non-linear optimization problem is formulated to compute the resource share of each application in case the total demand exceeds the available capacity.

In [35], a web application is modeled by a tandem queue; each application component is represented by a stage in the tandem queue. Each component runs on a separate VM. For the case of multiple applications, the physical resources are dynamically allocated to the VM's that host these applications. An optimization problem is formulated to determine the best configuration in the presence of contention.

The problem of resource provisioning for the database tier of a multi-tier web application is considered in [45]. A regression technique is used to train a non-linear function that correlates the number of database servers (or replicas) as well as various database parameters to the perceived response time. This function is updated when the difference between the predicted and observed values is beyond a certain threshold. An optimization problem is formulated to determine, for a given set of applications, the minimum number of replicas needed by each application to meet its SLA. The optimization problem is solved using a greedy search algorithm.

Tesauro et al [90] consider the problem of resource provisioning for multiple application environments (AE's). Each AE computes a local utility function that indicates the value of every possible resource level from the application's point of view. The AE's may select between two different techniques to compute this value function: the M/M/1 queueing model or reinforcement learning. A global arbiter allocates the available resources among AE's in order to maximize a global utility function.

The work in [92] is concerned with the problem of autonomic resource allocation in datacenters. The investigation takes into consideration such issues as statefulness, volatile workload and stringent SLA's in the form of a high percentile of response time meeting a given threshold. A control framework is developed to reconfigure the applications' resource level to meet their respective SLA's. The effectiveness of this approach is evaluated using a system running on Amazon EC2.

2.4 Service time distribution

A job submitted to a web application places a demand on system resources. In queueing theory, the duration of this demand is the job's service time. For mathematical tractability, service time is often represented by an exponential distribution [11, 39, 47, 48, 55, 56, 90, 107]. Other distributions have also been used in the context of resource provisioning for web applications. A common approach is to fit a probability distribution to measurement data on service time.

For web applications, it has been observed that job service times can be quite different from one job class to another [33, 84, 86]. In [84], a clustering algorithm is used to partition service time data into clusters; the statistical properties of these clusters are quite different. For each cluster, the fitted service time distribution is shifted exponential. The overall service time distribution is given by a linear combination of the per-cluster distributions, resulting in a weighted sum of shifted exponentials. In [33, 86], the k-means clustering algorithm is used to partition a sample of service time data into a number of job classes, and a distribution is fit to each class. The work in [86] further assumes that the number of classes is constant, yet the frequencies of each class in the overall workload may change with time.

The Phase-type (PH) distribution is a method to represent a non-exponential distribution in terms of exponentials, and can be used to approximate any positive-valued distribution [18]. It can therefore be used as a close approximation to a general service time distribution. It also allows us to model multiple classes of jobs, each having quite different service requirement. Efficient fitting of PH distributions to measurement data has been the topic of many studies. Asmussen et al [18] consider the problem of fitting a PH distribution to observed data using the expectation maximization (EM) algorithm.

In [54], a tool called PhFit that fits an acyclic discrete or continuous-time PH to an empirical distribution is described. The procedure starts with a random initial point for the PH parameter

values. A distance function between the empirical and fitted distributions is defined. At each iteration, partial derivatives of this distance function with respect to PH parameters are numerically computed. The simplex method is then used to determine the direction along which the distance function decreases optimally. A solution is found when the value of the distance function falls below a threshold.

In [27], a hyper-Erlang distribution is fit to service time data. It has been shown that hyper-Erlang is theoretically as powerful as PH in terms of representing a general service time distribution [91]. Panchenko et al [77] present a tool called GFIT to fit a hyper-Erlang distribution to a trace of measured job service times. This tool creates an aggregate trace from the original trace in two steps. First, the observed range of values for service times is divided into a number of subintervals; the length of these subintervals are fixed on a logarithmic scale. Secondly, all observed service times belonging to a given subinterval are represented in the aggregate trace by the average of these service times and the fraction of service times that fall into that subinterval. The EM algorithm is used to fit a hyper-Erlang distribution to the aggregate trace.

Fitting special cases of PH distribution has also been reported in the literature. In [60], a hyper-exponential distribution is fitted to measurement data using the EM algorithm, while the study in [80] considers the fitting of a mixture of Erlang and hyper-exponential distributions. Finally, the characterization of a general service time distribution in terms of the first and the second moments is presented in [9, 14, 43, 86].

In this thesis, a PH distribution is used to characterize job service times. We believe a queueing model with MMPP arrival process and PH service time is capable of capturing the key aspects of web application workload, while remaining mathematically tractable.

Chapter 3

Traffic model

In this chapter, we evaluate the effectiveness of Markov-modulated Poisson process (MMPP) as a traffic model for web applications. An overview of the methodology used in our evaluation is as follows. Time-variation in input traffic is quantified by a metric called *overdemand* [27]. This metric allows us to generate job arrivals with a controlled level of time-variation in a systematic and computationally efficient manner. An available tool called BURN [27] is extended to generate a synthetic trace of job arrivals with a given overdemand. We then fit an MMPP to the synthetic trace. Simulation is conducted to obtain performance results when jobs are submitted in accordance with (i) the synthetic trace and (ii) the fitted MMPP. The effectiveness of MMPP is evaluated by comparing these results over a range of operating conditions.

The remainder of this chapter is organized as follows. In Section 3.1, we present the preliminaries related to the development of our methodology. Our methodology is described in Section 3.2. Results on the effectiveness of MMPP are presented in Section 3.3. Finally, Section 3.4 concludes the chapter.

3.1 Preliminaries

In our investigation, service time is modeled by a phase-type (PH) distribution. The BURN tool uses empirical data, obtained from executing benchmarks such as TPC-W over an experimental system, to generate a synthetic trace. We use a synthetic trace instead of a real trace available in the public domain because the BURN tool can be tuned to generate traces with controlled levels of time-variation. In this section, we first present an overview of MMPP. This is followed by brief discussions of the PH distribution, TPC-W specification and the structure of BURN.

3.1.1 Markov-modulated Poisson process

MMPP is a generalization of the Poisson process where the arrival rate changes over time. It is characterized by an M-state irreducible continuous-time Markov chain. When the Markov chain is in state j, arrivals occur in accordance with a Poisson process with rate λ_j. We denote by q_{ij} the transition rate from state i to state $j \neq i$ of the Markov chain ($q_{ii} = -\sum_{j \neq i} q_{ij}$). From the description above, an M-state MMPP can be fully characterized by parameters λ_j and q_{ij}, i, j $= 1, \cdots, M$.

3.1.2 Phase-type distribution

The PH distribution is a method to represent a non-exponential distribution in terms of exponentials, and can be used to approximate any positive-valued distribution [18]. It is characterized by an absorbing Markov chain with one absorbing state, labeled 0, and K transient states. An example with $K = 3$ is shown in Figure 3.1. For transient states k and $l \neq k$, let s_{kl} and s_{k0} be the transition rates from k to l and from k to the absorbing state, respectively. We have $s_{kk} = -(s_{k0} + \sum_{l \neq k} s_{kl})$. The Markov chain starts in a transient state, where ζ_k is the probability

that the initial state is k. A PH distribution is fully characterized by parameters ζ_k and s_{kl}, $k, l = 1, \cdots, K$.

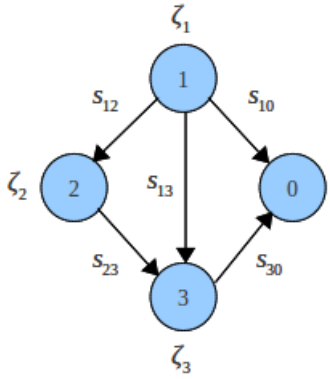

Figure 3.1: Graphical representation of PH distribution

Let $F(y)$ be the CDF of the PH distribution. $F(y)$ is the probability that the time to absorption is at most y. Also, let $\boldsymbol{\zeta}$ and $\mathbf{S}$ be, respectively, a row vector with entries ζ_k and a square matrix with entries s_{kl}. We have

$$F(y) = 1 - \boldsymbol{\zeta} e^{\mathbf{S}y} \mathbf{1} \tag{3.1}$$

where $e^{\mathbf{S}y}$ is the matrix exponential, and $\mathbf{1}$ is a column vector where each entry has the value 1.

3.1.3 TPC-W

TPC-W is a transactional web benchmark. A brief description of TPC-W as well as an implementation of this benchmark can be found in [44]. Another implementation is described in [24]. TPC-W has been used to evaluate the performance of web applications (see for example [12, 98, 106]). It is defined for a client/server architecture with a front-end web/application server and a back-end database server. A client program emulates multiple users concurrently submit-

ting jobs to the web server. The number of users is an input parameter to the client program. Each user starts a session and submits a sequence of logically-related jobs to the server and receives a response for each job. The response is in the form of an HTML page. There is a think time between when a user receives a response and when he submits the next job. The think time distribution is exponential with mean seven seconds. The session length is exponential with mean 15 minutes[1]. A number of transaction types are defined in the benchmark. The rate at which transactions of each type are submitted by the users is governed by the workload mix. TPC-W defines three different workloads, namely browsing, shopping and ordering.

3.1.4 BURN

BURstiness eNabling method (BURN) is a tool that generates synthetic workload with controlled time-variation [27]. The workload consists of a series of sessions that are created and terminated over time. The service time of a session, also referred to as the session-level service time, is the sum of service times of individual jobs within the session. Let $\bar{d}$ be the average service time of a session. In BURN, time-variation is quantified in terms of overdemand, denoted by ϕ and defined as

$$\phi = \Pr\left[\text{service time of current session} \geq \bar{d} \;\middle|\; \text{service time of last session} \geq \bar{d}\right]$$

Overdemand can be viewed as the tendency to keep one or more resources busy serving a burst of requests for a continuous period of time [27].

BURN takes as input the session-level service time distribution of $N \geq 2$ benchmarks[2], a benchmark mix $\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_N]$ ($\sum_i \alpha_i = 1$) and a target overdemand ϕ. Any session-based transactional benchmark such as TPC-W can be used. It is assumed that session-level service time

[1] In TPC-W specification, the session length is the duration of the session measured in time. Session length can also be defined in terms of the number of jobs in a session. In this thesis, the latter definition is adopted.

[2] At least two benchmarks are required in order to achieve a range of values for overdemand.

has PH distribution. BURN computes a policy matrix $\mathbf{E}$ to mix the sessions sampled from the N benchmarks; the result is an output mix where the fraction of sessions from the i^{th} benchmark is α_i, and the overdemand is ϕ. $\mathbf{E}$ is an $N \times N$ matrix where the ij^{th} element is the probability that the next session in the output mix is sampled from the j^{th} benchmark given that the current session is sampled from the i^{th} benchmark.

3.2 Methodology

In this section, we describe the methodology that we use to evaluate the effectiveness of MMPP as a traffic model for web applications. Our methodology consists of three steps. First, we generate a trace of job arrivals with a given overdemand, which we call the *synthetic trace*. Next, we fit an MMPP to the synthetic trace and use an MMPP load generator to produce another trace of job arrivals, which we call the *MMPP trace*. Finally, the two traces are used as input to a simulation to evaluate the effectiveness of MMPP.

The generation of the synthetic trace requires the availability of service time data at the job and session levels. In our investigation, service time data is collected by running an available benchmark such as TPC-W over an experimental system. This allows us to obtain measurement data on service time under realistic conditions.

3.2.1 Synthetic trace

Figure 3.2 depicts the procedure to generate the synthetic trace. Each component in the figure, denoted by a rectangle, generates its output when all of its inputs become available. The procedure requires the availability of $N \geq 2$ benchmarks; the i^{th} benchmark is denoted by BM_i. Each benchmark is session-based and transactional. For BM_i, the distributions of think time and session length (the number of jobs in a session) are known and assumed to be given by probability

density functions (pdf's) $z_i(\cdot)$ and $p_i(\cdot)$, respectively. The use of a continuous pdf for session length distribution is common in the literature [63, 64]. A non-integer session length sampled from a continuous pdf is rounded to the closest integer.

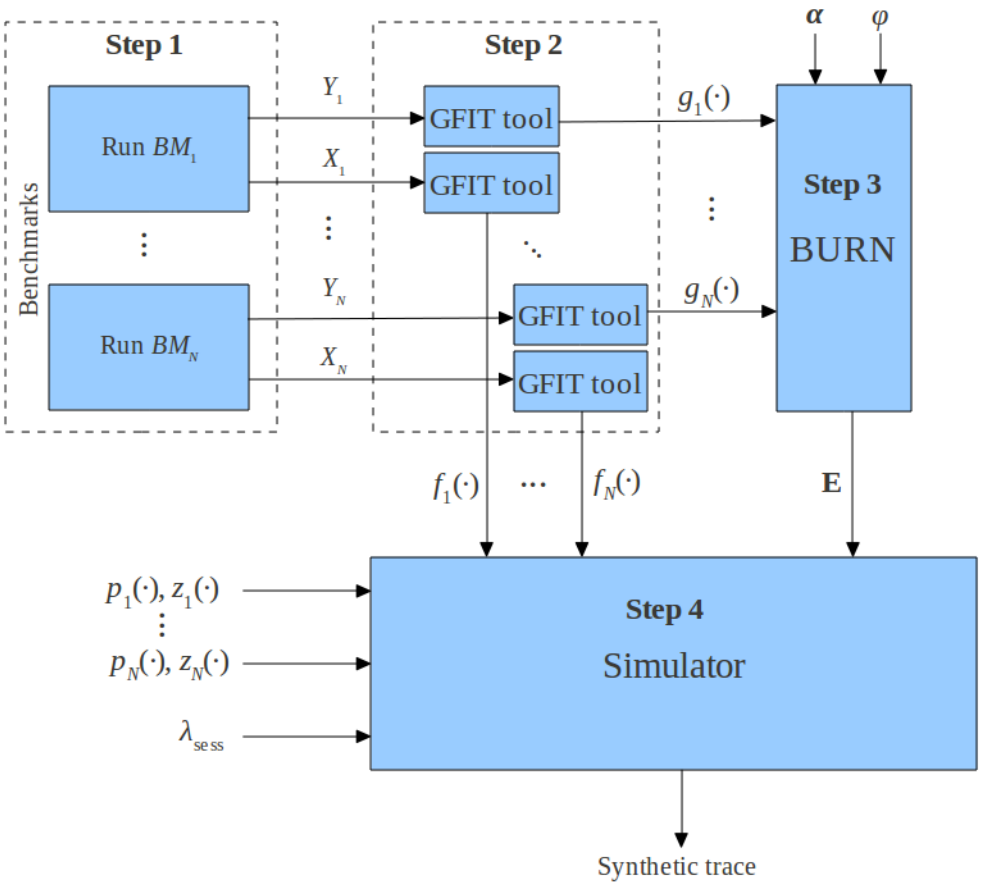

Figure 3.2: Procedure to generate the synthetic trace

The steps to obtain the synthetic trace are as follows:

Step 1 - Each benchmark is run separately on an experimental system. There is only one user in the system. User sessions are created serially, and jobs within a session are executed one at a time. Data on job and session-level service times are collected. Suppose we execute BM_i for

a total of L sessions, and the length of the l^{th} session ($1 \leq l \leq L$) is L_l. Let x_{lj} be the job-level service time of the j^{th} job of the l^{th} session, $1 \leq l \leq L$ and $1 \leq j \leq L_l$. The x_{lj}'s form a sample of observations for the job-level service times. For the l^{th} session, the session-level service time is the sum of the service time of jobs executed within the session, and is given by $y_l = \sum_{j=1}^{L_l} x_{lj}$. The y_l's form a second sample for the session-level service times. For BM_i, the two samples are denoted by X_i and Y_i, respectively.

Step 2 - In this step, we determine the job and session-level service time distributions. For each benchmark BM_i, the GFIT tool [77] is used to fit two PH distributions, one to X_i and the other to Y_i. Let $f_i(\cdot)$ and $g_i(\cdot)$ be, respectively, the job and session-level service time distributions, obtained from GFIT; $f_i(\cdot)$ and $g_i(\cdot)$ have the form shown in Equation 3.1.

Step 3 - The $g_i(\cdot)$'s ($1 \leq i \leq N$) are used as input to BURN along with a benchmark mix $\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_N]$ and a target overdemand ϕ. The output is a policy matrix $\mathbf{E}$ as defined in Section 3.1.4.

Step 4 - In this step, we run a simulation program for the queueing model shown in Figure 3.3 to generate the synthetic trace. The number of active sessions changes over time as new sessions are created and old sessions are terminated. The time between the creation of two successive sessions is exponentially distributed with mean λ_{sess}^{-1}. For the first session, the benchmark from which the session is drawn is determined randomly according to the steady-state distribution of $\mathbf{E}$. For subsequent sessions, given that the current session was drawn from BM_i, the next session is drawn from BM_j with probability e_{ij} (the ij^{th} element of $\mathbf{E}$). For a session drawn from BM_i, the simulator determines random variates for the number of jobs in the session, the think times between job submissions, and the service time of each job using, respectively, $p_i(\cdot)$, $z_i(\cdot)$ and $f_i(\cdot)$. During the simulation, an entry is created in the synthetic trace each time a job is submitted by an active session; this entry contains the job arrival and service times. The simulation terminates

after a trace of N_J jobs is generated.

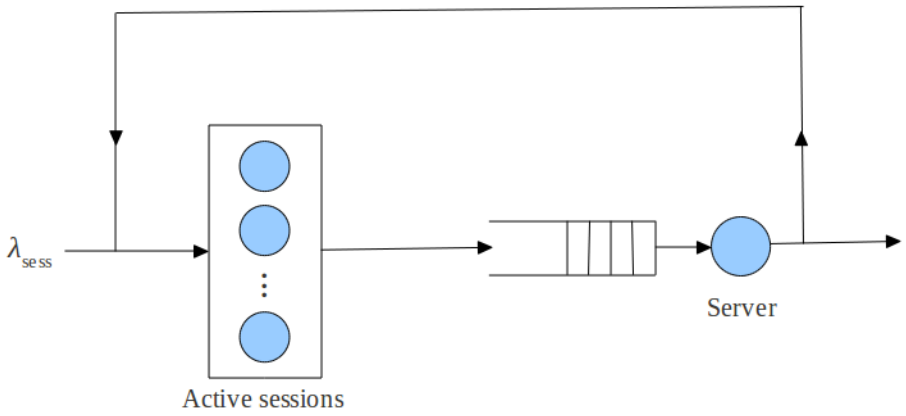

Figure 3.3: Simulated queueing model to generate the synthetic trace

3.2.2 MMPP trace

We next consider algorithms to fit an MMPP to the synthetic trace. The algorithm in [21], which we refer to as Algorithm 1, is attractive since it places no restriction on the number of MMPP states and is computationally efficient. It determines the number of arrivals per second (or the arrival rate) in the synthetic trace. As an example, Figure 3.4 shows the number of arrivals per second for 9200 seconds. The data from the first 1800 seconds are discarded to account for the warm-up period in the simulation to generate the synthetic trace.

Let $\lambda_{\min}$ and $\lambda_{\max}$ be, respectively, the minimum and maximum observed rates. Algorithm 1 computes regions $(\delta_2, \delta_1], (\delta_3, \delta_2], \cdots, (\delta_{M_R+1}, \delta_{M_R}]$ such that $\lambda_{\max} = \delta_1$ and $\lambda_{\min} \in (\delta_{M_R+1}, \delta_{M_R}]$.

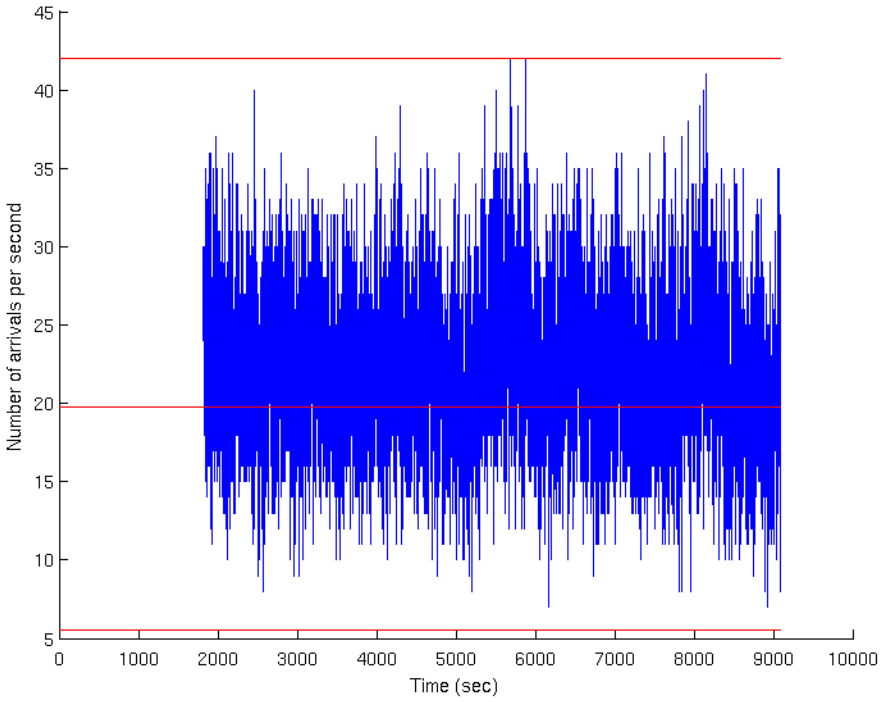

Figure 3.4: Arrival rates obtained from the synthetic trace

The region boundaries δ_i's are determined by solving the following system of equations:

$$\begin{aligned} &\xi_1 + 2\sqrt{\xi_1} = \lambda_{\max} \\ &\xi_i + 2\sqrt{\xi_i} = \xi_{i-1} - 2\sqrt{\xi_{i-1}} \quad i = 2, \cdots, M_R \\ &\delta_1 = \lambda_{\max} \\ &\delta_{i+1} = 2\xi_i - \delta_i \qquad\qquad\qquad i = 1, \cdots, M_R \end{aligned} \tag{3.2}$$

where ξ_i is the average of the two boundaries of region $(\delta_{i+1}, \delta_i]$. Each region corresponds to one MMPP state. The resulting MMPP has therefore $M = M_R$ states. For our example in Figure 3.4, the data result in two regions $(19.77, 42]$ and $(5.54, 19.77]$. Therefore, the estimated MMPP

has $M = 2$ states. The arrival rate at state i (denoted by λ_i) is the average of the two boundaries of the i^{th} region $(\delta_{i+1}, \delta_i]$, i.e.,

$$\lambda_i = \frac{\delta_i + \delta_{i+1}}{2} \quad i = 1, \cdots, M \tag{3.3}$$

The method to compute q_{ij} is the same as that presented in [21].

Equation 3.3 may not be a good estimate for λ_i in the presence of outliers. For example, in Figure 3.4, the arrival rate for MMPP state 1, i.e., the state corresponding to the region $(19.77, 42]$, is $\lambda_1 = (\delta_1 + \delta_2)/2 = 30.89$ jobs/second in spite of the fact that a majority of arrival rates in this region fall below 30.89 jobs/second. This is because of a large value for δ_1 caused by an outlier at 42 jobs/second. As a result, $\lambda_1 = 30.89$ jobs/second does not accurately represent the arrival rate observations.

We propose a variant of Algorithm 1, which we call Algorithm 2. Similar to Algorithm 1, the range of observed arrival rates is divided into M_R regions, and the region boundaries δ_i are determined using Equation 3.2. The arrival rate at state i, however, is computed differently. Let N_{R_i} be the number of observed arrival rates falling into region i (or $(\delta_{i+1}, \delta_i]$) and λ_{ij} $(1 \leq j \leq N_{R_i})$ be the j^{th} observed rate in region i. The arrival rate at MMPP state i is computed by

$$\lambda_i = \frac{1}{N_{R_i}} \sum_j \lambda_{ij} \quad i = 1, \cdots, M$$

In other words, λ_i is the average of all the observed rates falling into region i. Using Algorithm 2, the arrival rate for MMPP state 1 is given by $\lambda_1 = 24.73$ jobs/second, which is consistent with the observed data. The q_{ij}'s for Algorithm 2 are computed as in the case of Algorithm 1.

Using an MMPP with the estimated parameters, a load generator is used to generate the MMPP trace. The trace contains N_J jobs, i.e., the same number of jobs as the synthetic trace.

For each job in the MMPP trace, the service time is sampled from the distribution $f(\cdot)$ which is a weighted sum of N job-level service time distributions from the original benchmarks (or $f_i(\cdot)$'s); the weight of the i^{th} distribution is given by $\omega_i = \bar{n}_i\alpha_i / \sum_j \bar{n}_j\alpha_j$ ($\bar{n}_i$ is the average session length for BM_i), i.e.,

$$f(\cdot) = \sum_{i=1}^{N} \omega_i \; f_i(\cdot)$$

ω_i can be interpreted as the fraction of jobs from BM_i in the synthetic trace.

3.2.3 Effectiveness of MMPP

In [21], the effectiveness of MMPP is evaluated based on the results of a simulation study. A brief overview of their methodology is as follows. The simulation model used is a finite-buffer single-server queue with constant service time. Two performance metrics are considered, namely, the average queueing delay and the job drop rate. Two simulation experiments are performed. The server capacity is the same in both experiments. In the first experiment, jobs are submitted to the server in accordance with a trace obtained from an operational system. An MMPP is fit to this trace; job arrivals are generated in accordance with this MMPP in the second experiment. The effectiveness of MMPP is evaluated by comparing the respective values of the two performance metrics obtained from the two simulation experiments.

Similar to [21], our evaluation is based on a single-server simulation model. This model is an infinite-buffer single-server queue with PH service time distribution. It is assumed that the server capacity can be adjusted to meet a given performance target. Consistent with the approach presented in [21], two metrics (defined below) are used in our evaluation. Our methodology to assess the effectiveness of MMPP is as follows:

1. The procedure in Section 3.2.1 is used to generate a synthetic trace with a given overdemand.

This trace contains the job arrival and service times.

2. An MMPP is fitted to the arrival time data in the synthetic trace. This MMPP is used by a load generator as described in Section 3.2.2 to create the MMPP trace. The MMPP trace also contains the job arrival and service times.

3. Simulation experiments are performed to determine the server capacity denoted by C such that Pr[response time $\leq x$] $= \beta$ when jobs are submitted in accordance with the synthetic trace. We assume that job service times in the synthetic (or MMPP) trace (denoted by d) are for the case where the server in Figure 3.3 is operating at full capacity (i.e., 100% capacity). The required capacity C is determined using a binary search. Throughout the search, we maintain a lower bound $C_{\min}$ and an upper bound $C_{\max}$, initially set to zero and a large number, respectively. At each iteration of the search, a simulation is performed with server capacity set to $C_{\text{avg}} = (C_{\min} + C_{\max})/2$. For each job, the service time used in the simulation is given by d/C_{avg}. The simulation run yields the fraction of jobs whose response time is at most x; let β^* denote this fraction. If $\beta^* > \beta + \epsilon$ (where ϵ is a small positive number), $C_{\max}$ is set to C_{avg}, and $C_{\min}$ remains unchanged; if $\beta^* < \beta - \epsilon$, $C_{\min}$ is set to C_{avg}, and $C_{\max}$ remains unchanged. In both cases, the search proceeds to the next iteration. On the other hand, if $|\beta^* - \beta| < \epsilon$, then $C = C_{\text{avg}}$ is the required capacity.

4. A second simulation experiment is performed with server capacity set to C (obtained from Step 3) and jobs submitted in accordance with the MMPP trace. Again, the service time used in the simulation is given by d/C where d is the service time in the MMPP trace. The fraction of jobs meeting the response time threshold x is measured; let this fraction be β'. The relative error between β and β', denoted by re_1, is used as the first metric in our

evaluation; re_1 is given by

$$re_1 = \frac{|\beta - \beta'|}{\beta} \times 100 \tag{3.4}$$

5. Similar to Step 3, a binary search, with a simulation run at each iteration, is performed to determine the server capacity C' such that Pr[response time $\leq x$] $= \beta$ when jobs are submitted in accordance with the MMPP trace. The relative error between C and C' is used as the second metric in our evaluation. This metric, denoted by re_2 is given by

$$re_2 = \frac{|C - C'|}{C} \times 100 \tag{3.5}$$

For the two metrics re_1 and re_2, the smaller is the relative error, the more effective is the MMPP.

3.3 Evaluation

As mentioned earlier, an experimental system is used to collect service time data in order to obtain realistic service time distribution for our evaluation. In this section, we first describe the experimental system and the methodology used to obtain service time data. We then present results on the effectiveness of MMPP as a traffic model.

3.3.1 Experimental system

Our experimental system consists of three nodes in an IBM blade center model H chassis. The TPC-W implementation from university of Wisconsin [3] is deployed on this system. Specifically, the web and database servers are running on two separate nodes, each is configured with 2 AMD dual-core CPUs at 2.0 GHz, 10 GB of memory, and 36 GB of internal disk. The client emulator runs on a third node which has two AMD dual-core CPUs at 2.4 GHz, 8 GB of memory, and 36

GB of internal disk. All three nodes run the Ubuntu 11.04 operating system. The three nodes are connected through a 100 Mbit/second Ethernet link. The database server is MySQL version 5.1. The web server is Apache Tomcat version 6.0. We use the default configuration for both the web and database servers.

3.3.2 Service time data

$N \geq 2$ benchmarks are used in our procedure to generate a synthetic trace (see Section 3.2.1). Each benchmark is run in single-user mode, and service time data are collected. In our investigation, system resources that host the application are treated as a black-box. Since there is no queueing in the system, the service time in the black-box is the same as the response time. Measurement data on response time for N_J jobs are collected; this data constitutes a sample of job-level service times. Note that the job-level service time is the aggregate service time at the two nodes that host the web and database servers. The session-level service time is given by the sum of job-level service times of all jobs in the session.

To obtain synthetic traces with a range of values for overdemand, we follow the suggestion in [63] and use two benchmarks: BM_1 is TPC-W ordering mix with session length distribution $p_1(\cdot)$ given by a truncated Pareto, and BM_2 is TPC-W browsing mix with session length distribution $p_2(\cdot)$ being a truncated shifted exponential. The parameter values for these two distributions are shown in Table 3.1; these values are consistent with the empirical data available in [64]. The think time distribution is exponential with mean seven seconds and is truncated at 70 seconds.

Each benchmark BM_i, $i = 1, 2$, is executed for a duration of 24 hours, resulting in two samples X_i and Y_i, for job and session-level service times, respectively; X_i and Y_i are then used as input to the GFIT tool to obtain the corresponding PH distributions $f_i(\cdot)$ and $g_i(\cdot)$. The parameter values for these distributions are presented in Table 3.2. Note that only non-zero values of ζ_k and

Table 3.1: Parameter values for session length distribution

Benchmark	Session length distribution	Parameters
BM_1	Truncated Pareto	Min value = 3 Max value = 120 Shape parameter = 1.2085
BM_2	Truncated shifted exponential	Min value = 3 Max value = 120 Rate = 0.1553

s_{kl} are shown.

Table 3.2: Parameter values for job and session-level service time distributions

CDF	K	ζ_k	s_{kl}
$f_1(\cdot)$	7	$\zeta_1 = 0.012$ $\zeta_4 = 0.988$	$s_{12} = s_{23} = s_{30} = 10.9$ $s_{45} = s_{56} = s_{67} = s_{70} = 407.2$
$g_1(\cdot)$	8	$\zeta_1 = 0.346$ $\zeta_2 = 0.654$	$s_{10} = 4.1$ $s_{23} = s_{34} = s_{45} = s_{56} = s_{67} = s_{78} = s_{80} = 147.8$
$f_2(\cdot)$	7	$\zeta_1 = 0.117$ $\zeta_3 = 0.883$	$s_{12} = s_{20} = 6.1$ $s_{34} = s_{45} = s_{56} = s_{67} = s_{70} = 459.8$
$g_2(\cdot)$	8	$\zeta_1 = 0.676$ $\zeta_4 = 0.324$	$s_{12} = s_{23} = s_{30} = 5.1$ $s_{45} = s_{56} = s_{67} = s_{78} = s_{80} = 88.9$

3.3.3 Results and discussion

In this section, we present simulation results on the effectiveness of MMPP as a traffic model. A number of scenarios are considered in our evaluation. Recall that the synthetic trace is generated by mixing sessions sampled from input benchmarks according to a policy matrix generated by BURN. Since there are two benchmarks, the session mix is given by $\boldsymbol{\alpha} = [\alpha_1, \alpha_2]$. Three input mixes are considered; they are $\boldsymbol{\alpha} = [0.3, 0.7], [0.5, 0.5]$ and $[0.7, 0.3]$. This ensures that our evaluation includes cases where either of the two input benchmarks account for a larger fraction of

sessions in the output mix, as well as a case of equal contribution by the two benchmarks. For each input mix, we used the methodology presented in [27] and obtained the range of feasible values for ϕ denoted by $[\phi_{\min}, \phi_{\max}]$. We also consider three values for the average arrival rate of new sessions; they are $\lambda_{sess} = 2.22$ (as in [27]), 5 and 10 sessions/second. The performance target is of the form Pr[response time $\leq x$] $= \beta$. The values of x and β are 3 and 5 seconds, and 0.9 and 0.95, respectively. The distributions $f_i(\cdot)$, $g_i(\cdot)$, $p_i(\cdot)$ and $z_i(\cdot)$ are kept the same throughout our experiments. The number of jobs in the synthetic (or MMPP) trace, denoted by N_J, is 200000.

The effectiveness is evaluated in terms of the two metrics re_1 and re_2 as defined in Equations 3.4 and 3.5, respectively. Each simulation experiment is replicated five times, and presented results are in the form of mean and 95% confidence interval.

In our first set of results, we set $\lambda_{sess} = 2.22$ sessions/second, and study the effectiveness of MMPP for different values of ϕ, $\boldsymbol{\alpha}$, x and β. We first compare the performance of the two MMPP fitting algorithms mentioned in Section 3.2.2, namely Algorithms 1 and 2. Figure 3.5 shows the results for re_1 as a function of ϕ for $\boldsymbol{\alpha} = [0.3, 0.7]$, $x = 3$ seconds and $\beta = 0.9$. As can be observed, when Algorithm 2 is used to determine the MMPP parameters, the relative errors are less than 8%. For Algorithm 1, the relative errors can be as high as 80%. The corresponding results for re_2 are shown in Figure 3.6. Again, when Algorithm 2 is used, the relative errors are less than 4%, while with Algorithm 1, the relative errors can be as high as 20%. We believe the inaccuracy of Algorithm 1 is because of the presence of outliers in the job arrival data as explained in Section 3.2.2. Our conclusion is that Algorithm 2 is a more effective algorithm to determine the MMPP parameters. In the rest of our evaluation, we only present results for Algorithm 2.

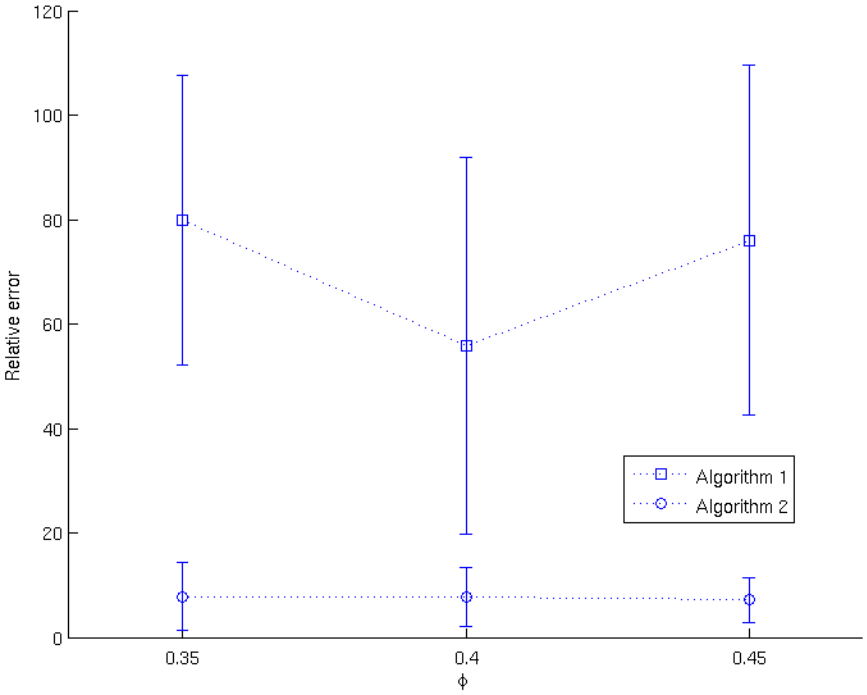

Figure 3.5: Comparison of Algorithms 1 and 2 in terms of re_1

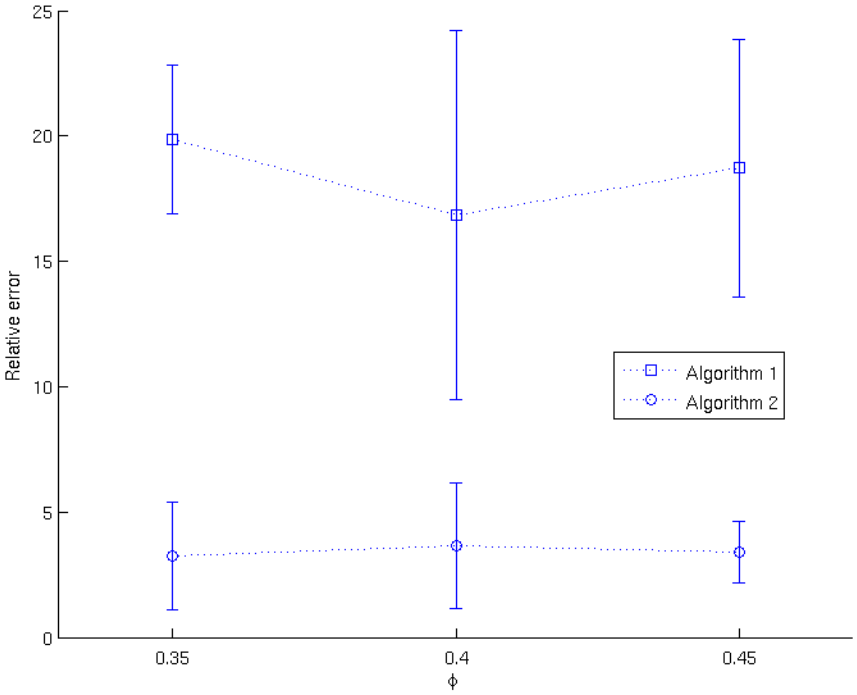

Figure 3.6: Comparison of Algorithms 1 and 2 in terms of re_2

In Figures 3.7 and 3.8, re_1 is plotted against ϕ for $\boldsymbol{\alpha} = [0.3, 0.7]$, $x = 3$ and 5 seconds, and $\beta = 0.9$ and 0.95. It can be observed that the relative errors remain below 8% when $x = 3$ seconds and below 5% when $x = 5$ seconds. The corresponding results for re_2 are shown in Figures 3.9 and 3.10, where the relative errors are below 4% in a majority of cases and reach as high as 6.5% in some cases. These observations suggest that MMPP is capable of capturing the time-variation in the synthetic trace.

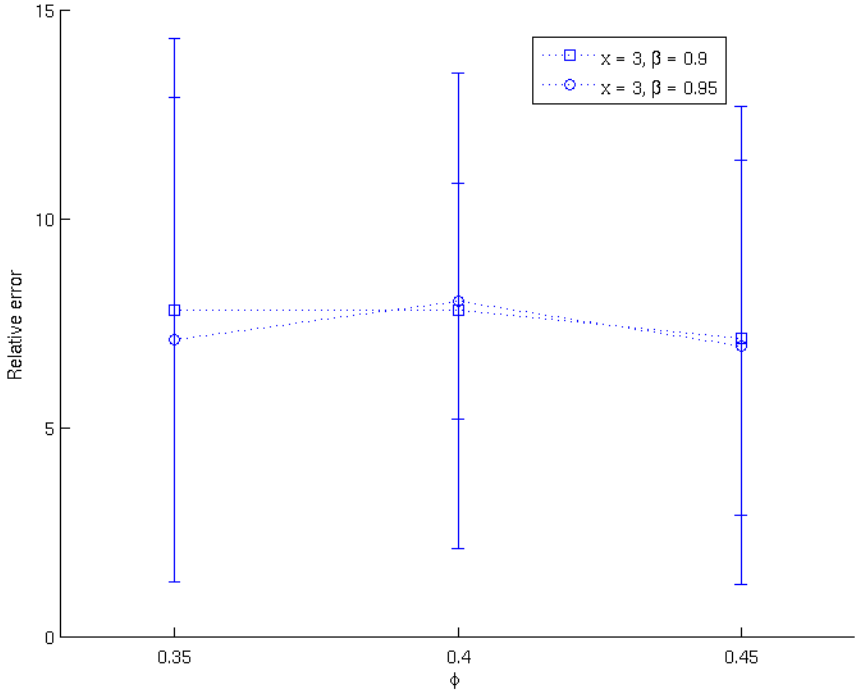

Figure 3.7: re_1 versus ϕ for $\lambda_{sess} = 2.22$ sessions/second and $\boldsymbol{\alpha} = [0.3, 0.7]$

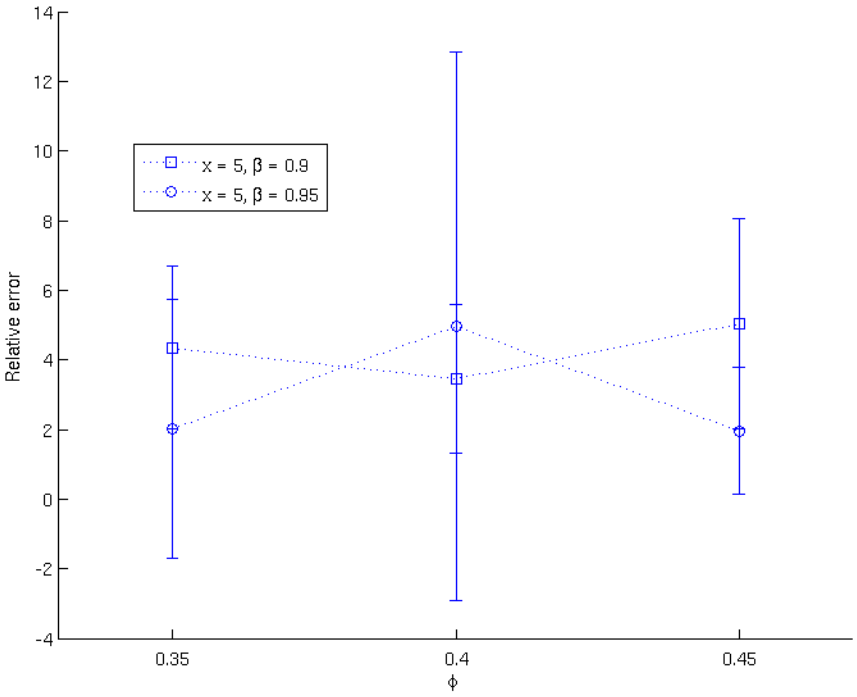

Figure 3.8: re_1 versus ϕ for $\lambda_{sess} = 2.22$ sessions/second and $\boldsymbol{\alpha} = [0.3, 0.7]$

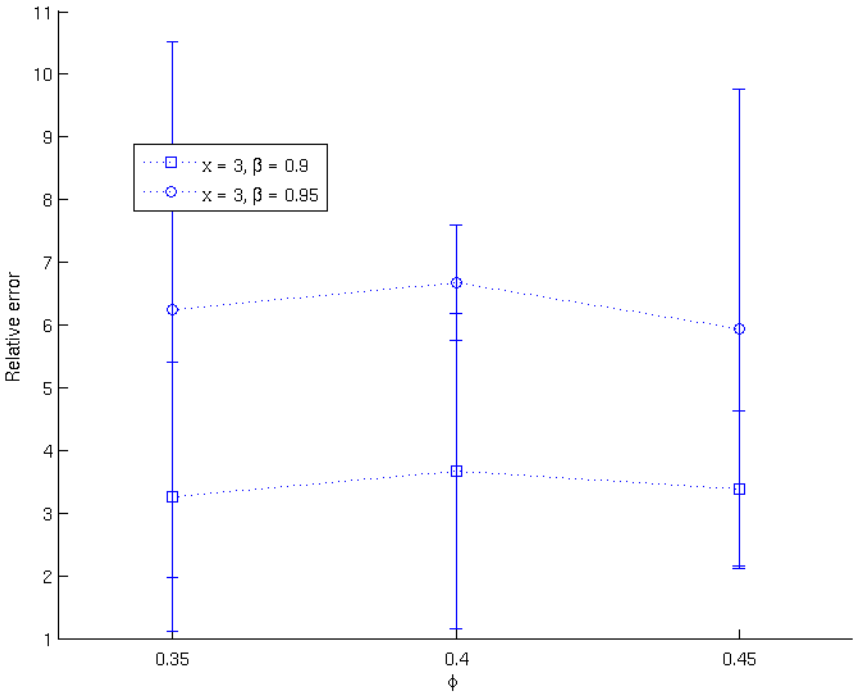

Figure 3.9: re_2 versus ϕ for $\lambda_{sess} = 2.22$ sessions/second and $\boldsymbol{\alpha} = [0.3, 0.7]$

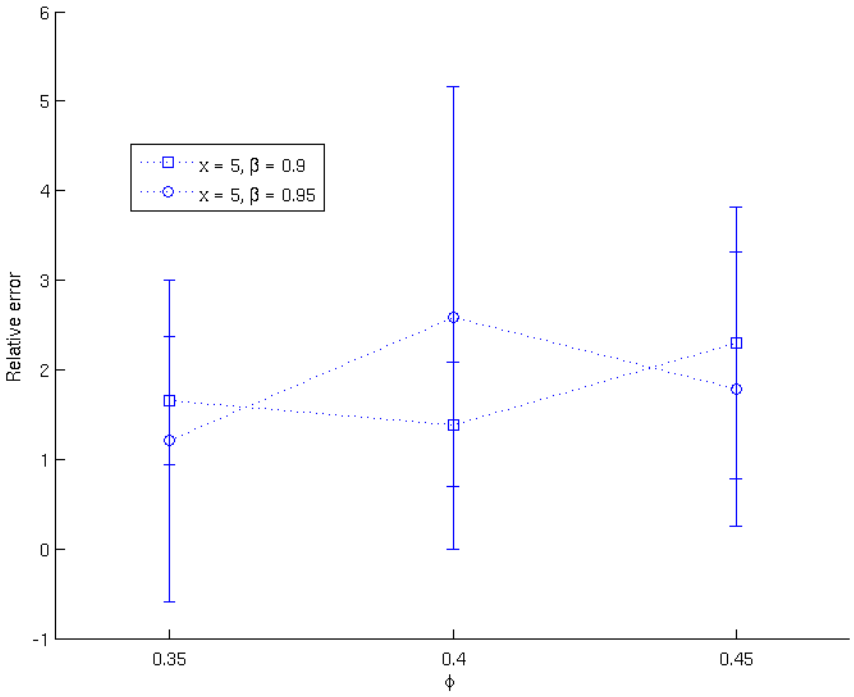

Figure 3.10: re_2 versus ϕ for $\lambda_{sess} = 2.22$ sessions/second and $\boldsymbol{\alpha} = [0.3, 0.7]$

In Figures 3.11 and 3.12, re_1 is plotted against ϕ for $\boldsymbol{\alpha} = [0.5, 0.5]$ and $[0.7, 0.3]$, respectively. In each figure, results are shown for different combination of values for x and β. For clarity, confidence intervals are not included. It is observed that the relative errors remain below 4% in a majority of cases; the largest observed value is 7.5%. The corresponding results for re_2 are shown in Figures 3.13 and 3.14. The relative errors are below 4.5% in a majority of cases, and the largest observed value is 7%. These observations further suggest that MMPP is an acceptable traffic model for time-varying traffic.

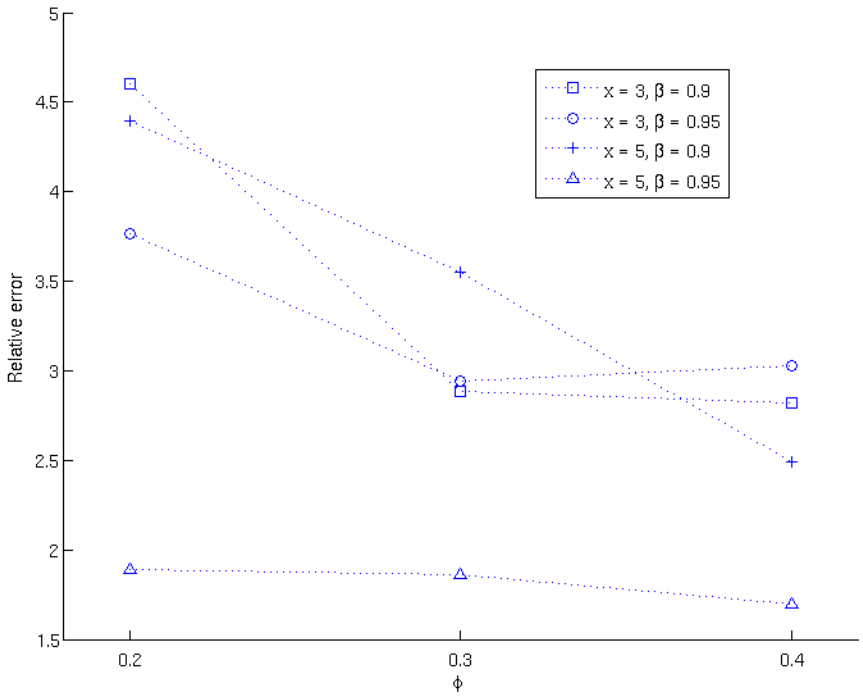

Figure 3.11: re_1 versus ϕ for $\lambda_{sess} = 2.22$ sessions/second and $\boldsymbol{\alpha} = [0.5, 0.5]$

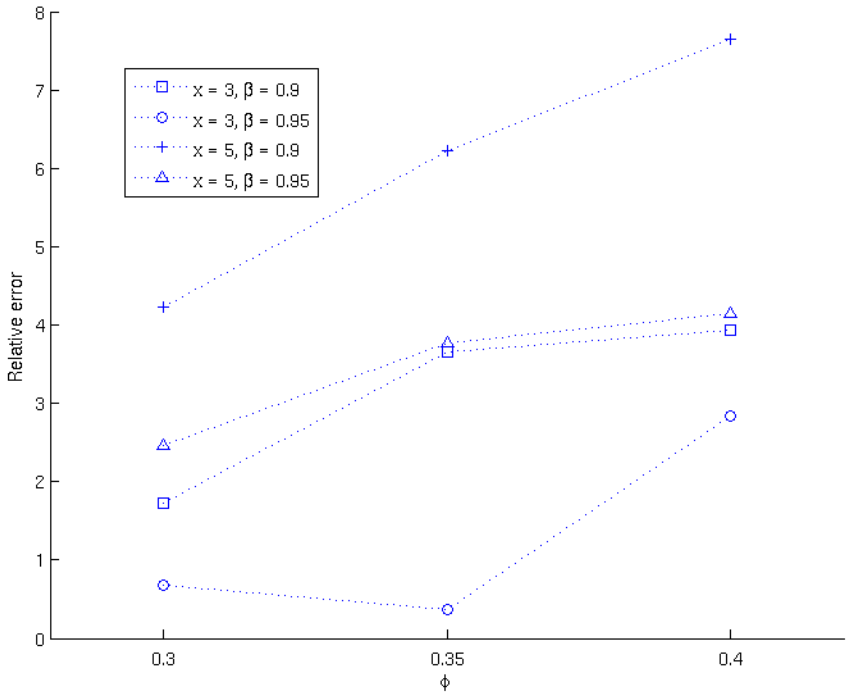

Figure 3.12: re_1 versus ϕ for $\lambda_{sess} = 2.22$ sessions/second and $\boldsymbol{\alpha} = [0.7, 0.3]$

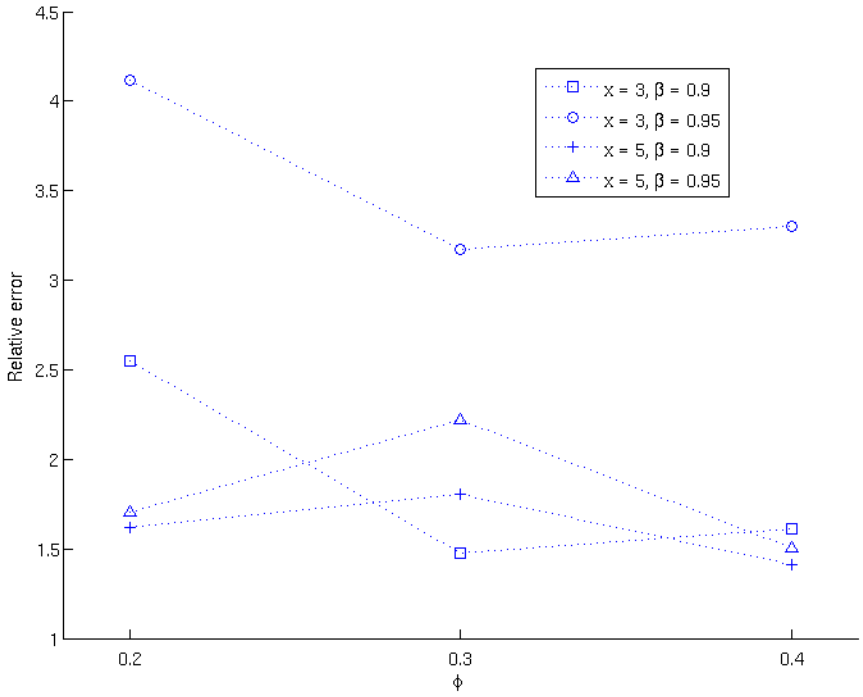

Figure 3.13: re_2 versus ϕ for $\lambda_{sess} = 2.22$ sessions/second and $\boldsymbol{\alpha} = [0.5, 0.5]$

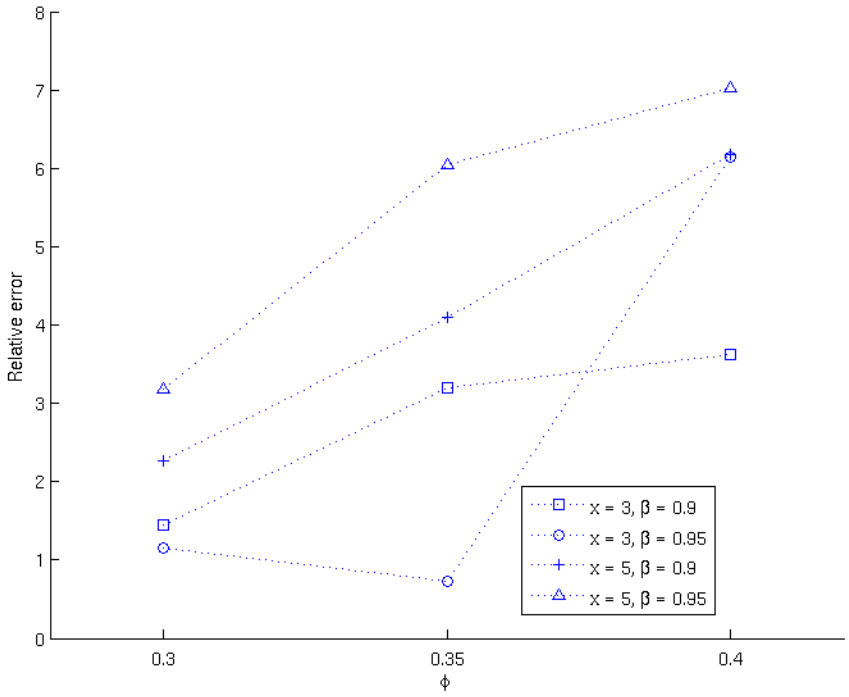

Figure 3.14: re_2 versus ϕ for $\lambda_{sess} = 2.22$ sessions/second and $\boldsymbol{\alpha} = [0.7, 0.3]$

In the next set of experiments, we study the effectiveness of MMPP for larger values of session arrival rates. In Figure 3.15, re_1 is plotted against ϕ for $\lambda_{sess} = 5$ sessions/second, $\boldsymbol{\alpha} = [0.5, 0.5]$, and different combination of values for x and β. The corresponding results for $\lambda_{sess} = 10$ sessions/second are shown in Figure 3.16. In both graphs, the relative errors remain below 6% in a majority of cases, while reaching as high as 9.5% in two cases.

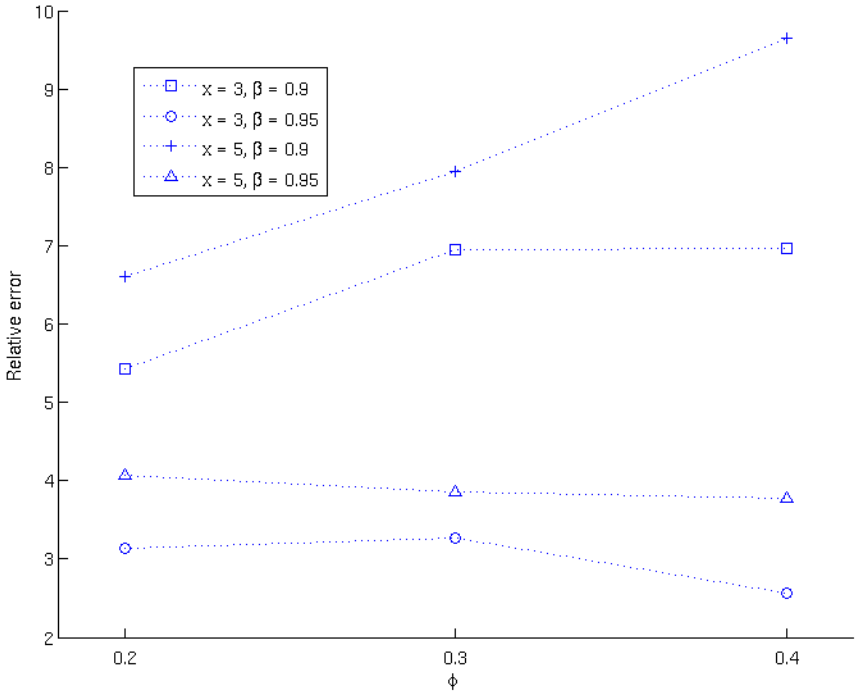

Figure 3.15: re_1 versus ϕ for $\lambda_{sess} = 5$ sessions/second and $\boldsymbol{\alpha} = [0.5, 0.5]$

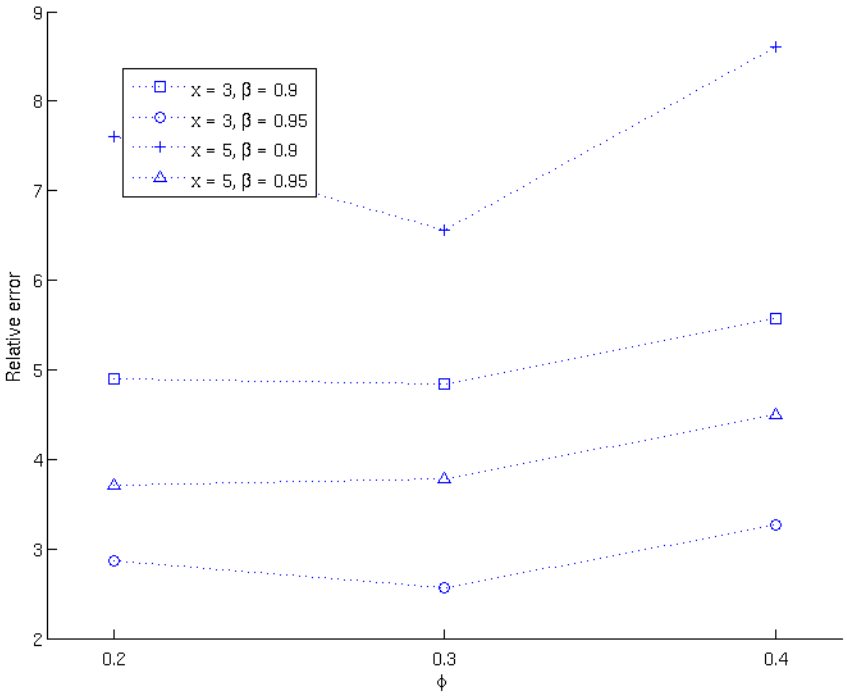

Figure 3.16: re_1 versus ϕ for $\lambda_{sess} = 10$ sessions/second and $\boldsymbol{\alpha} = [0.5, 0.5]$

The corresponding results for re_2 are shown in Figures 3.17 and 3.18. We observe that the relative errors fall below 6%. We have also conducted experiments for different combinations of $\lambda_{sess} = 5$ and 10 sessions/second, and $\boldsymbol{\alpha} = [0.3, 0.7]$ and $[0.7, 0.3]$; the results are not shown as similar observations were made. Based on the results presented in this section, we conclude that MMPP is an acceptable traffic model for time-varying traffic.

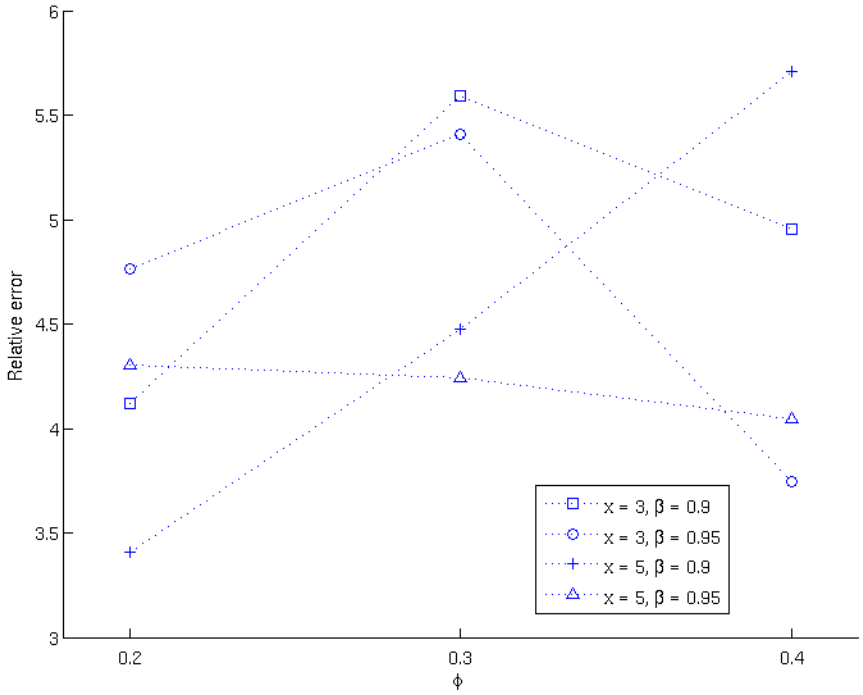

Figure 3.17: re_2 versus ϕ for $\lambda_{sess} = 5$ sessions/second and $\boldsymbol{\alpha} = [0.5, 0.5]$

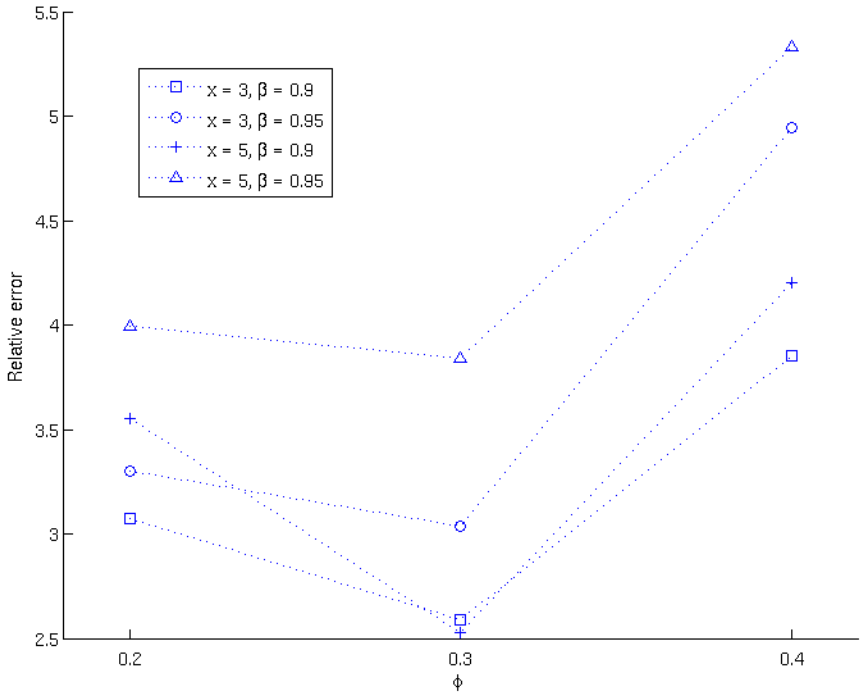

Figure 3.18: re_2 versus ϕ for $\lambda_{sess} = 10$ sessions/second and $\boldsymbol{\alpha} = [0.5, 0.5]$

3.4 Conclusion

We evaluated the effectiveness of MMPP in modeling time-varying traffic. Our approach entails the generation of a synthetic trace with a given overdemand and the fitting of an MMPP to this trace. We also proposed a variant of an available MMPP fitting algorithm that has the desired property of robust performance in the presence of outliers. Our results show that MMPP is effective in modeling fine-scale time-variation; the relative errors remain reasonably small in all of the experiments.

In our investigation, time-variation is characterized in terms of overdemand. The choice of overdemand facilitates the generation of synthetic traces in a systematic and computationally efficient manner. We recognize that there are other metrics to quantify time-variation, e.g., index of dispersion [26]. Nevertheless, we believe that our methodology provides insight into the suitability of MMPP as a traffic model for web applications. Evaluation using other measures of time-variation or other workloads is out of the scope of this thesis.

Chapter 4

MMPP/PH/1 (FCFS) model - Time-dependent results

In Chapter 3, we evaluated the effectiveness of MMPP as a traffic model for web applications. The rest of this thesis is concerned with resource provisioning strategies for a web application assuming that the arrival process is MMPP. Our approach is to use analytic results from a queueing model to determine the resource requirement of an application to meet a given performance target.

A key step in our methodology is to solve a queueing model for the system under investigation to obtain the response time CDF. As discussed in Section 1.2, two queueing models are considered in this thesis, namely, a two-stage tandem queue and a single-server model, and the single-server model is used in the first research problem investigated under resource provisioning strategies. Consistent with this approach, the results presented in this chapter are based on the single-server model. This model is shown in Figure 4.1; it is an MMPP/PH/1 queue with FCFS discipline.

Since resource provisioning is performed for a finite-duration service interval, the queueing system is in transient phase initially; this is followed by steady-state behavior. The response time

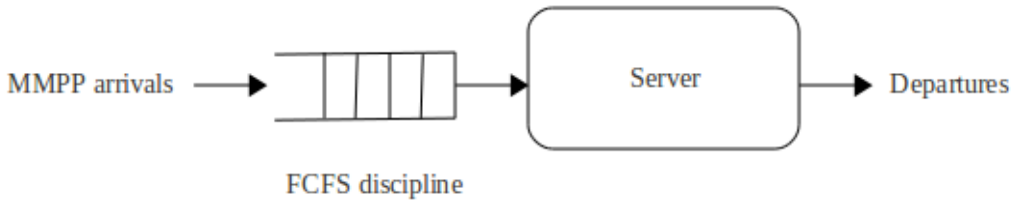

Figure 4.1: Single-server model

CDF is time-dependent during the transient phase and becomes unchanging in time at steady state. Steady-state results are significantly less costly to compute than time-dependent results, and can be used as an approximation if the length of transient phase is small compared to that of the service interval. In this chapter, we present a time-dependent analysis of the MMPP/PH/1 (FCFS) model; this analysis is used to identify service interval lengths where steady-state results are a good approximation.

The rest of this chapter is organized as follows. The time-dependent response time analysis of the MMPP/PH/1 (FCFS) model is presented in Section 4.1. In Section 4.2, techniques to compute numerical results are discussed. Numerical examples on the response time CDF are presented in Section 4.3. These examples allow us to draw conclusions on service interval lengths where steady-state results are a good approximation. Finally, Section 4.4 concludes the chapter.

4.1 Analysis

In this section, we present the time-dependent response time analysis of the MMPP/PH/1 (FCFS) model. Recall from Sections 3.1.1 and 3.1.2 that MMPP and PH parameters are denoted, respectively, by λ_j and q_{ij}, and ζ_k and s_{kl}. The service time CDF is denoted by $F(y)$, and has the

form shown in Equation 3.1. We also denote by

$$h(y) = \frac{dF(y)/dy}{1 - F(y)}$$

the instantaneous service completion rate. The expression

$$\frac{F(y + \Delta t) - F(y)}{1 - F(y)} = h(y)\Delta t + o\left(\Delta t\right)$$

is the probability that a service takes at most $y + \Delta t$ given that it takes longer than y. The term $o(\Delta t)$ has the property that $\lim_{\Delta t \to 0} \frac{o(\Delta t)}{\Delta t} = 0$.

Let t_0 denote the beginning of a time interval of length t as shown in Figure 4.2. Denote by $r_{ij}(y, k, x, v, t)$ the joint probability that (i) there are $v \geq 0$ arrivals in $(t_0, t_0 + t)$, (ii) MMPP state at $t_0 + t = j$, and (iii) the unfinished work in the system immediately after $t_0 + t$ is at most x, given that (i) the MMPP state at $t_0 = i$, (ii) there are $k \geq 0$ jobs in the system at t_0, and if $k \geq 1$, the first job has been served for $y \geq 0$ (for $k = 0$, the only valid value for y is zero), (iii) and there is an arrival at $t_0 + t$.

Now consider a time interval of length $t+\Delta t$ and assume that $k \geq 1$. To obtain $r_{ij}(y, k, x, v, t+\Delta t)$, we condition on the events that may occur in $(t_0, t_0 + \Delta t]$:

1. An arrival with probability $\lambda_i \Delta t + o(\Delta t)$; in this case,

$$r_{ij}(y, k, x, v, t + \Delta t) = \begin{cases} 0 & v = 0 \\ r_{ij}(y + \Delta t, k + 1, x, v - 1, t) & v \geq 1 \end{cases}$$

2. A change in MMPP state from i to $\sigma \neq i$ with probability $q_{i\sigma}\Delta t + o(\Delta t)$; in this case,

$$r_{ij}(y, k, x, v, t + \Delta t) = r_{\sigma j}(y + \Delta t, k, x, v, t)$$

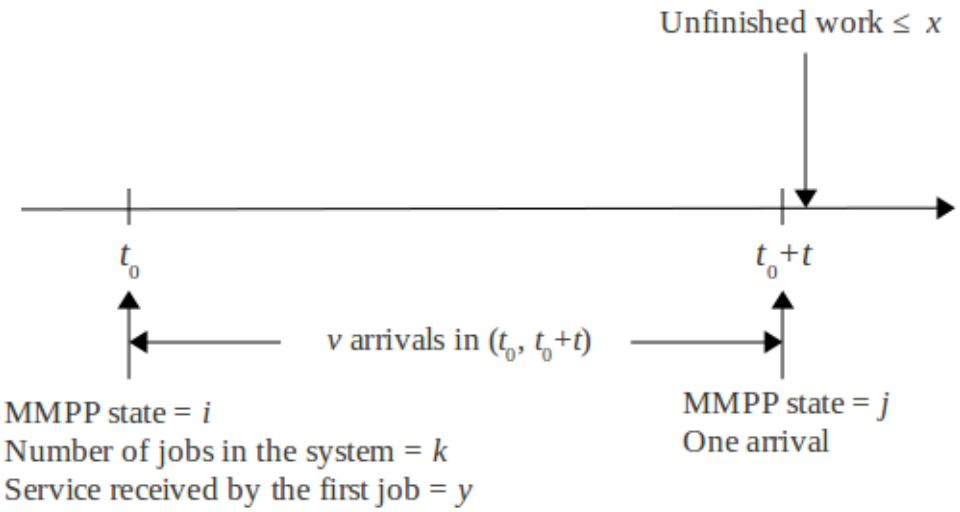

Figure 4.2: Definition of $r_{ij}(y, k, x, v, t)$

3. A service completion with probability $h(y)\Delta t + o(\Delta t)$; in this case,

$$r_{ij}(y, k, x, v, t + \Delta t) = r_{ij}(0, k - 1, x, v, t)$$

In addition, we have

4. The probability of any combination of two or more events occurring in $(t_0, t_0 + \Delta t]$ is $o(\Delta t)$.

5. The probability of no event occurring in $(t_0, t_0 + \Delta t]$ is $1 - (\lambda_i + h(y) - q_i)\Delta t + o(\Delta t)$, where $q_i = -\sum_{\sigma \neq i} q_{i\sigma}$; in this case,

$$r_{ij}(y, k, x, v, t + \Delta t) = r_{ij}(y + \Delta t, k, x, v, t)$$

Let δ_{kl} denote the Kronecker delta function [75] defined as

$$\delta_{kl} = \begin{cases} 1 & k = l \\ 0 & k \neq l \end{cases}$$

Using the law of total probability, we have

$$\begin{aligned} r_{ij}(y,k,x,v,t+\Delta t) &= (1-\delta_{v0})\ \lambda_i \Delta t\ r_{ij}(y+\Delta t, k+1, x, v-1, t) \\ &+ \textstyle\sum_{\sigma \neq i} q_{i\sigma} \Delta t\ r_{\sigma j}(y+\Delta t, k, x, v, t) \\ &+ h(y)\Delta t\ r_{ij}(0, k-1, x, v, t) \\ &+ (1-(\lambda_i + h(y) - q_i)\,\Delta t)\ r_{ij}(y+\Delta t, k, x, v, t) \\ &+ o(\Delta t) \end{aligned}$$

which results in the following partial differential equation (PDE) as $\Delta t \to 0$

$$\begin{aligned} \tfrac{\partial}{\partial t} r_{ij}(y,k,x,v,t) - \tfrac{\partial}{\partial y} r_{ij}(y,k,x,v,t) &= (1-\delta_{v0})\ \lambda_i\ r_{ij}(y, k+1, x, v-1, t) \\ &+ \textstyle\sum_{\sigma \neq i} q_{i\sigma}\ r_{\sigma j}(y, k, x, v, t) \\ &+ h(y)\ r_{ij}(0, k-1, x, v, t) \\ &- (\lambda_i + h(y) - q_i)\ r_{ij}(y, k, x, v, t) \end{aligned} \tag{4.1}$$

Following a similar approach for the case where $k = 0$, we obtain the following ordinary differential equation.

$$\begin{aligned} \tfrac{d}{dt} r_{ij}(0,0,x,v,t) &= (1-\delta_{v0})\ \lambda_i\ r_{ij}(0, 1, x, v-1, t) \\ &+ \textstyle\sum_{\sigma \neq i} q_{i\sigma}\ r_{\sigma j}(0, 0, x, v, t) \\ &- (\lambda_i - q_i)\ r_{ij}(0, 0, x, v, t) \end{aligned} \tag{4.2}$$

The initial condition, given by $r_{ij}(y,k,x,v,0)$, corresponds to a time interval of length zero. In a time interval of length zero, there is no change in the MMPP state and no job arrival. As a

result, $r_{ij}(y,k,x,v,0) = 0$ for all $j \neq i$ or $v > 0$. For $j = i$ and $v = 0$, $r_{ii}(y,k,x,0,0)$ is the same as the probability that the unfinished work immediately after t_0 is at most x, given that there are $k+1$ jobs in system at t_0, and if $k \geq 1$, the first job has been served for $y \geq 0$. Denote this probability by $H(y,k,x)$. We have

$$r_{ij}(y,k,x,v,0) = \delta_{ij}\ \delta_{v0}\ H(y,k,x) \tag{4.3}$$

To obtain an equation for $H(y,k,x)$, we note that if $k = 0$, the unfinished work immediately after t_0 is the same as the service time of the arrival at t_0; $H(0,0,x)$ is therefore given by $F(x)$ (or the service time CDF). For $k \geq 1$, the unfinished work immediately after t_0 is given by the residual service time of the first job plus the total service time of k jobs. For $k = 1$ and $k \geq 2$, $H(y,k,x)$ is given by

$$H(y,1,x) = \int_0^x \frac{F(y+x-z) - F(y)}{1 - F(y)}\, dF(z)$$

and

$$H(y,k,x) = \int_0^x H(y,k-1,x-z)\, dF(z)$$

respectively.

To numerically solve the PDE in Equation 4.1, a boundary condition is required. We were not able to establish the boundary condition under a general PH service time distribution. However, for a broad subset of PH distributions, including the hyper-Erlang distribution, which satisfy the conditions presented in [95], the instantaneous service completion rate $h(y)$ is independent of y as $y \to \infty$, i.e., $\lim_{y\to\infty} h(y)$ exists, and the boundary condition is given by

$$\lim_{y\to\infty} \frac{\partial}{\partial y} r_{ij}(y,k,x,v,t) = 0 \tag{4.4}$$

Let $\psi_i(v,t)$ be the probability that there are $v \geq 0$ arrivals in (t_0, t_0+t), given that the MMPP state at t_0 is i. It can be shown that

$$\psi_i(v,t) = \sum_j r_{ij}(y,k,+\infty,v,t)$$

The equation holds for any valid combination of values for k and y.

We now derive an expression for the time-dependent response time CDF of the MMPP/PH/1 (FCFS) model over a time interval of length T (or $[0,T]$). Suppose that at time zero, the system is empty and the MMPP state is i. For $v \geq 1$, we have

$$\Pr\left[v \text{ arrivals in } [0,T] \text{ and response time of the } l^{th} \text{ arrival} \leq x\right]$$

$$= \int_0^T \sum_j r_{ij}(0,0,x,l-1,t)\ \lambda_j\ \psi_j(v-l,T-t)\ dt \qquad 1 \leq l \leq v$$

Considering all possible values of v and l, we have

$$\Pr\left[\text{at least one arrival in } [0,T] \text{ and response time of one of these arrivals selected at random} \leq x\right]$$

$$= \sum_{v=1}^{\infty} \sum_{l=1}^{v} \frac{1}{v} \int_0^T \sum_j r_{ij}(0,0,x,l-1,t)\ \lambda_j\ \psi_j(v-l,T-t)\ dt$$

The probability that there is at least one arrival in $[0,T]$ is equal to $1-\psi_i(0,T)$. Let $r(x,T)$ be the response time CDF over $[0,T]$, given that the system is empty at time zero and there is at least one arrival in $[0,T]$. $r(x,T)$ is given by

$$r(x,T) = \sum_i \frac{\pi_i}{1-\psi_i(0,T)} \sum_{v=1}^{\infty} \sum_{l=1}^{v} \frac{1}{v} \int_0^T \sum_j r_{ij}(0,0,x,l-1,t)\ \lambda_j\ \psi_j(v-l,T-t)\ dt \qquad (4.5)$$

where π_i is the probability that MMPP state at time zero is i.

4.2 Numerical computation

To facilitate numerical computation, it is advantageous to represent Equations 4.1 to 4.5 in matrix form. Let $\mathbf{R}(y,k,x,v,t)$ be a square matrix with entries $r_{ij}(y,k,x,v,t)$. Equations 4.1 and 4.2 can be combined into the following equation:

$$\begin{aligned} \tfrac{\partial}{\partial t}\mathbf{R}(y,k,x,v,t) - \bar{\delta}_{k0}\,\tfrac{\partial}{\partial y}\mathbf{R}(y,k,x,v,t) &= \bar{\delta}_{v0}\,\boldsymbol{\Lambda}\,\mathbf{R}(y,k+1,x,v-1,t) \\ &+ \bar{\delta}_{k0}\,h(y)\,\mathbf{R}(0,k-1,x,v,t) \\ &- \left(\boldsymbol{\Lambda} + \bar{\delta}_{k0}\,h(y)\,\mathbf{I} - \mathbf{Q}\right)\,\mathbf{R}(y,k,x,v,t) \end{aligned} \tag{4.6}$$

where $\bar{\delta}_{ij} = 1 - \delta_{ij}$, $\boldsymbol{\Lambda}$ is a diagonal matrix with entries λ_i on the main diagonal, $\mathbf{Q}$ is a matrix with entries q_{ij}, and $\mathbf{I}$ is the identity matrix.

The initial and boundary conditions in matrix form for Equations 4.3 and 4.4 are given by

$$\mathbf{R}(y,k,x,v,0) = \delta_{v0}\,H(y,k,x)\,\mathbf{I}$$

and

$$\lim_{y\to\infty}\frac{\partial}{\partial y}\mathbf{R}(y,k,x,v,t) = \mathbf{0}$$

respectively, where $\mathbf{0}$ is a matrix with each entry having the value 0.

Finally, Equation 4.5 can be written as

$$r(x,T) = \boldsymbol{\pi}\mathbf{D}\sum_{v=1}^{\infty}\sum_{l=1}^{v}\frac{1}{v}\int_0^T \mathbf{R}(0,0,x,l-1,t)\,\boldsymbol{\Lambda}\,\boldsymbol{\Psi}(v-l,T-t)\,dt \tag{4.7}$$

where $\boldsymbol{\pi}$ is a row vector with entries π_i, $\mathbf{D}$ is a diagonal matrix with entries $\frac{1}{1-\psi_i(0,T)}$ on the main

diagonal, and $\mathbf{\Psi}(v,t)$ is a column vector with entries $\psi_j(v,t)$.

Equation 4.7 can be used to compute $r(x,T)$ for given values of x and T. This requires the computation of $\mathbf{R}(y,k,x,v,t)$ using the PDE in Equation 4.6. In this section, we present the numerical scheme used for the computation of this PDE. We solve this PDE in a finite four-dimensional space denoted by

$$0 \leq y \leq y_{\max}, \quad 0 \leq k \leq k_{\max}, \quad 0 \leq v \leq v_{\max}, \quad 0 \leq t \leq t_{\max} = T$$

This space can be represented by two separate two-dimensional grids k-v and t-y. Variables k and v are integers. Variables t and y are continuous; they are discretized using uniform step sizes Δt and Δy along the t and y dimensions, respectively. At each grid point (k,v) in k-v, the values of $\mathbf{R}(y,k,x,v,t)$ are numerically computed over all grid points (t_m, y_n) in t-y, $m = 0, 1, \cdots, t_{\max}/\Delta t$, $n = 0, 1, \cdots, y_{\max}/\Delta y$.

It is clear from Equation 4.6 that the computation of $\mathbf{R}$ at grid point (k,v) requires the values of $\mathbf{R}$ at grid points $(k+1, v-1)$ and $(k-1, v)$. This can be achieved by computing $\mathbf{R}(y,k,x,v,t)$'s in the following order:

for $v = 0$ **to** $v_{\max}$
    for $k = 0$ **to** $k_{\max}$
        Compute $\mathbf{R}(y,k,x,v,t)$ over all grid points (t_m, y_n) in t-y

We use a finite-difference method [17] to approximate partial derivatives in Equation 4.6. Specifically, we use a one-step (or a two-level) scheme [17] where the partial derivatives with respect to t and y at grid point (t_m, y_n) are approximated by a one-sided difference at this point,

i.e.,

$$\left[\frac{\partial}{\partial t}\mathbf{R}(y,k,x,v,t)\right]_{t_m}^{y_n} \approx \frac{\mathbf{R}(y_n,k,x,v,t_m)-\mathbf{R}(y_n,k,x,v,t_{m-1})}{\Delta t}$$

and

$$\left[\frac{\partial}{\partial y}\mathbf{R}(y,k,x,v,t)\right]_{t_m}^{y_n} \approx \frac{\mathbf{R}(y_{n+1},k,x,v,t_m)-\mathbf{R}(y_n,k,x,v,t_m)}{\Delta y}$$

This finite-difference scheme is backward in time and forward in space (BTFS), and is first-order accurate with respect to both time and space variables.

A stable numerical scheme ensures that the accumulated errors as a result of approximating partial derivatives by finite differences remain bounded as the scheme proceeds through time. In the following, we present the stability analysis for the BTFS scheme applied to the PDE in Equation 4.6.

At time step $t = t_m$, $0 \le m \le t_{\max}/\Delta t$, define

$$u_m = \min_{i,j,n,k,v} r_{ij}(y_n,k,x,v,t_m) \quad \text{and} \quad U_m = \max_{i,j,n,k,v} r_{ij}(y_n,k,x,v,t_m)$$

where $1 \le i,j \le M$, $0 \le n \le y_{\max}/\Delta t$, $0 \le k \le k_{\max}$, and $0 \le v \le v_{\max}$; u_m and U_m are, respectively, the minimum and maximum observed values among solution functions $r_{ij}(y,k,x,v,t)$ at time step $t = t_m$. We show that for $0 < m \le t_{\max}/\Delta t$,

$$u_m \ge 0 \quad \text{and} \quad U_m \le U_{m-1} \tag{4.8}$$

Since the initial condition at $t = 0$ is positive and bounded, Inequality 4.8 ensures that the solution remains positive and bounded as the PDE evolves through time (or the solution method is unconditionally stable).

Applying the BTFS scheme to the PDE in equation 4.6, we have

$$\begin{aligned}(1+F)\ r_{ij}(y_n,k,x,v,t_m) &= \bar{\delta}_{v0}\ \lambda_i\Delta t\ r_{ij}(y_n,k+1,x,v-1,t_m)\\ &+ \sum_{\sigma\neq i} q_{i\sigma}\Delta t\ r_{\sigma j}(y_n,k,x,v,t_m)\\ &+ \bar{\delta}_{k0}\ h(y_n)\Delta t\ r_{ij}(0,k-1,x,v,t_m)\\ &+ \bar{\delta}_{k0}\ \tfrac{\Delta t}{\Delta y}\ r_{ij}(y_{n+1},k,x,v,t_m)\\ &+ r_{ij}(y_n,k,x,v,t_{m-1})\end{aligned} \tag{4.9}$$

where

$$F = \bar{\delta}_{k0}\,\frac{\Delta t}{\Delta y} + \left(\lambda_i + \bar{\delta}_{k0}\,h(y_n) - q_i\right)\Delta t$$

The coefficients on the right hand side of Equation 4.9 are positive. This yields

$$(1+F)\ r_{ij}(y_n,k,x,v,t_m) \le F'\,U_m + U_{m-1} \tag{4.10}$$

where

$$F' = \bar{\delta}_{k0}\,\frac{\Delta t}{\Delta y} + \left(\bar{\delta}_{v0}\,\lambda_i + \bar{\delta}_{k0}\,h(y_n) - q_i\right)\Delta t$$

Note that when $v = 0$, $F > F'$; otherwise, $F = F'$. Inequality 4.10 holds for all values of $r_{ij}(y,k,x,v,t)$ at $t = t_m$ including U_m. We thus have

$$(1+F)\,U_m \le F'\,U_m + U_{m-1}$$

which yields

$$U_m \le \frac{1}{1+F-F'}\,U_{m-1} \le U_{m-1}$$

The last inequality holds since $1+F-F' \ge 1$. A similar argument yields $(1+F-F')\,u_m \ge u_{m-1}$. Since the initial condition at $t = 0$ is positive and $1+F-F' > 0$, one can use mathematical

induction to conclude $u_m \geq 0$. This completes the stability analysis.

It should be mentioned that semi-Lagrangian schemes can also be used to solve the PDE in Equation 4.6. Such schemes have gained increased popularity [17, 69, 87] due to their inherent stability for large Δt [69]. They are based on the observation that the value of a multi-variate function $\mathcal{F}$ remains constant through the function's characteristic curves. Hence, the value of $\mathcal{F}$ at grid point (t_m, y_n) is computed by determining the characteristic curve (or an approximation for this curve) through (t_m, y_n), computing the value of this characteristic at time t_{m-1} (let y' denote this value), and evaluating $\mathcal{F}$ at (t_{m-1}, y'). Note that (t_{m-1}, y') does not necessarily coincide with a grid point; in this case, an approximation for $\mathcal{F}(t_{m-1}, y')$ is obtained by applying an interpolation to the values of $\mathcal{F}$ at nearby grid points to (t_{m-1}, y').

Semi-Lagrangian schemes expedite the numerical computation (by allowing a larger step size along the time dimension) at the cost of extra complexity. The use of a small Δt, however, is preferred in the computation of $\mathbf{R}(y, k, x, v, t)$. This is because the functions $\mathbf{R}(0, 0, x, v, t)$ are integrated with respect to time in Equation 4.7, and a small step size improves the accuracy of the integration. We hence use the one-sided BTFS scheme in our investigation.

4.3 Numerical examples

We consider three examples. The MMPP and PH parameter values for these examples are shown in Tables 4.1 and 4.2, respectively. These values are representative of those used in subsequent chapters when queueing models with MMPP arrivals are used for resource provisioning. Note that for the PH parameters, only non-zero values of ζ_k and s_{kl} are shown. For each example, the MMPP starts in state one, and $r(x, T)$ is computed for $x = 3$ seconds and $T = 10, 20, \cdots, 200$ seconds.

Table 4.1: MMPP parameters

Example	λ_j	q_{ij}
Example 1	$\lambda_1 = 0.7143$ $\lambda_2 = 2.1429$	$q_{12} = -q_{11} = 0.05$ $q_{21} = -q_{22} = 0.2$
Example 2	$\lambda_1 = 0.8571$ $\lambda_2 = 2.5714$	$q_{12} = -q_{11} = 0.05$ $q_{21} = -q_{22} = 0.2$
Example 3	$\lambda_1 = 0.8571$ $\lambda_2 = 2.5714$	$q_{12} = -q_{11} = 0.05$ $q_{21} = -q_{22} = 0.2$

Table 4.2: PH service time distribution

Example	K	ζ_k	s_{kl}
Example 1	1	$\zeta_1 = 1.00$	$s_{10} = 2.43$
Example 2	4	$\zeta_1 = 0.94$ $\zeta_3 = 0.06$	$s_{12} = s_{20} = 5.54$ $s_{34} = s_{40} = 135.2$
Example 3	4	$\zeta_1 = 0.89$ $\zeta_2 = 0.11$	$s_{10} = 2.8$ $s_{23} = s_{34} = s_{40} = 295.64$

In Figure 4.3, $r(3,T)$ is plotted against T for Example 1. When T is close to zero,

$$\Pr\left[\text{one arrivals in } [0,T] \,\middle|\, \text{one or more arrivals in } [0,T]\right] \approx 1$$

This implies that $r(3,0) \approx \Pr\left[\text{service time} \leq 3 \text{ seconds}\right]$. For Example 1, this probability has value 0.9993. As T increases, our results show that $r(3,T)$ decreases from 0.9993 at $T \approx 0$ to 0.9522 at $T = 100$ seconds and then to 0.9503 at $T = 200$ seconds. This behavior is consistent with the assumption that the system is empty at time zero. As jobs arrive after time zero, the server becomes more busy, resulting in smaller values for $r(3,T)$, or smaller probabilities that the response time is at most three seconds.

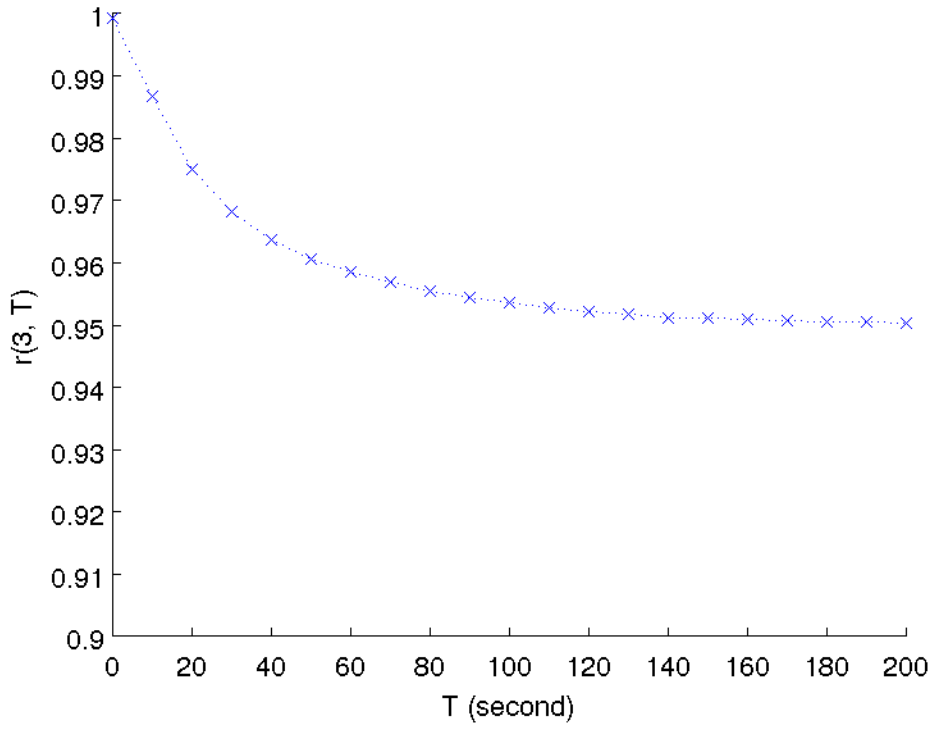

Figure 4.3: $r(3,T)$ as a function of T for Example 1

Consider the values of $r(3,T)$ from $T = 100$ seconds to $T = 140$ seconds at increments of 10 seconds. Let $d(T)$ be the rate at which $r(3,T)$ decreases when T is increased from T to $T+10$ seconds, i.e.,

$$d(T) = \frac{r(3,T) - r(3,T+10)}{10}$$

The values of $r(3,T)$ ($T = 100$ to 140 seconds) and $d(T)$ ($T = 100$ to 130 seconds) are shown in Table 4.3. We observe that the rates $d(T)$ are small (less than 0.01% when compared to $r(3,T)$) indicating that the value of $r(3,T)$ does not change significantly from T to $T+10$. We also observe that $d(T)$ is a decreasing function of T. It is therefore expected that any increase in T beyond 120 seconds would not result in a significant decrease in $r(3,T)$. For example, when T is increased to 200 seconds, $r(3,120) - r(3,200) = 1.9 \times 10^{-3}$ (or about 0.2% of $r(3,120)$).

Table 4.3: $r(3,T)$ and $d(T)$ for Example 1

T	100	110	120	130	140
$r(3,T)$	0.9536	0.9528	0.9522	0.9517	0.9512
$d(T)$	7.3×10^{-5}	6.2×10^{-5}	5.3×10^{-5}	5.1×10^{-5}	–

The above observations lead to the following conclusion. As defined earlier, $r(x, T)$ is the response time CDF of an arrival in $[0, T]$ selected at random. Since the queueing system is in transient phase initially, followed by the steady state, we have the scenario shown in Figure 4.4. When the system is in steady state, the response time CDF is independent of time. Our results do not provide any information on the length of transient phase (denoted by T_P). However, they indicate that for any value of $T > T_0 = 120$ seconds (or two minutes), $r(3, T)$ is not sensitive to changes in T; this implies that response time during the transient phase does not have a significant impact on $r(3, T)$. Our conclusion is that for Example 1, steady-state results are a good approximation when the service interval is longer than two minutes.

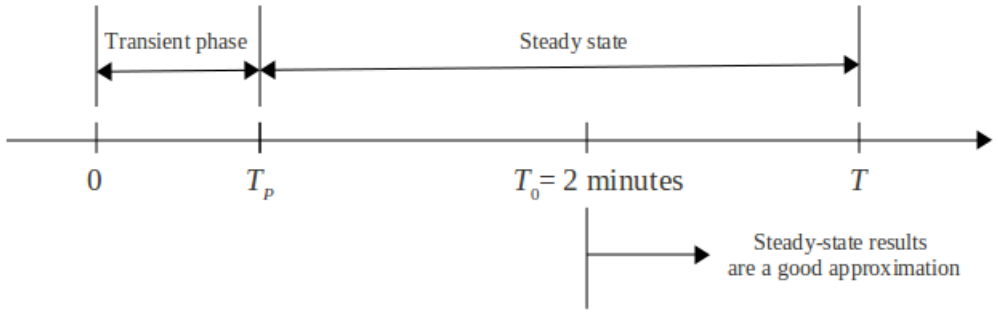

Figure 4.4: Steady-state results are a good approximation for $T > T_0$

Results for Examples 2 and 3 are shown in Figures 4.5 and 4.6, respectively; the corresponding values of $r(3,T)$ ($T = 100$ to 140 seconds) and $d(T)$ ($T = 100$ to 130 seconds) are shown in Tables 4.4 and 4.5. The observations are similar to those for Example 1, leading to the conclusion that for Examples 2 and 3, steady-state results are also a good approximation for any service interval longer than two minutes.

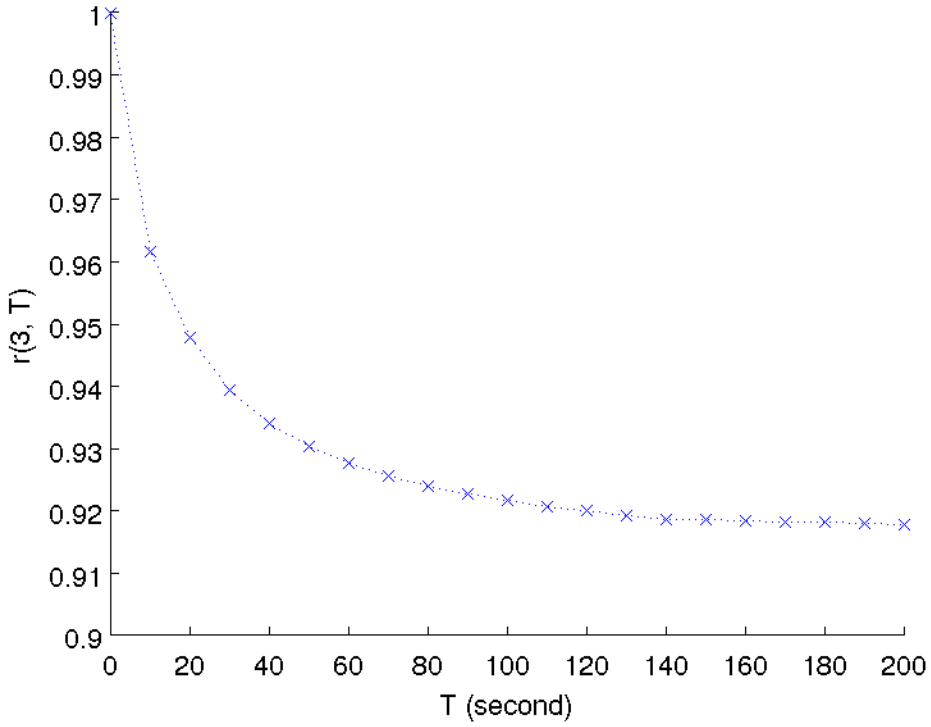

Figure 4.5: $r(3,T)$ as a function of T for Example 2

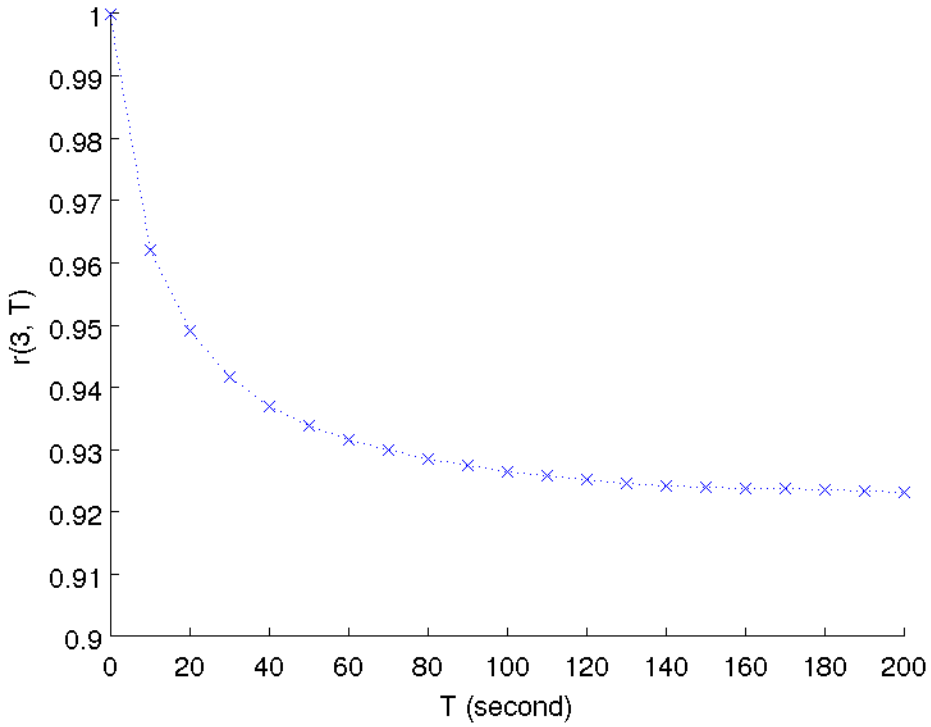

Figure 4.6: $r(3, T)$ as a function of T for Example 3

Table 4.4: $r(3,T)$ and $d(T)$ for Example 2

T	100	110	120	130	140
$r(3,T)$	0.9216	0.9207	0.92	0.9193	0.9187
$d(T)$	8.8×10^{-5}	7.6×10^{-5}	6.6×10^{-5}	6.1×10^{-5}	–

Table 4.5: $r(3,T)$ and $d(T)$ for Example 3

T	100	110	120	130	140
$r(3,T)$	0.9265	0.9258	0.9252	0.9246	0.9241
$d(T)$	7.3×10^{-5}	6.2×10^{-5}	5.5×10^{-5}	5.2×10^{-5}	–

4.4 Conclusion

In this chapter, we derived analytic results for the time-dependent response time CDF of the MMPP/PH/1 (FCFS) model. The results are new, and represent advancement to the state of knowledge in the field. Methods to compute numerical results from the analytic formulas were also discussed. Three examples were considered; the MMPP and PH parameter values in these examples are representative of those used in our experiments in subsequent chapters. For the these examples, we concluded that steady-state results for response time CDF are a good approximation if the service interval is longer than two minutes.

Chapter 5

Resource provisioning - Single-server model

Resource provisioning strategies considered in this thesis are based on two key factors, namely, (i) the queueing model used to obtain the response time CDF and (ii) the length of the service interval, or the time interval for which resource provisioning is performed.

As mentioned in Chapter 1, two queueing models are considered in this thesis, namely, a two-stage tandem queue and a single-server model. The two stages in the tandem queue represent VM's that host the web and database tiers, respectively (see for example [28, 103, 106]). For this model, service time data at each tier are needed to determine the service time distribution at the corresponding stage. The collection of service time data requires technical knowledge at the application, operating system or monitoring tool level.

The single-server model is the MMPP/PH/1 (FCFS) model presented in Chapter 4. In this model, the web and database VM's are treated as a black-box and represented by a single server. The service time at this server is given by the aggregate service time at the web and database

tiers. It is the same as the response time when there is no queueing. Since response time can be readily obtained by recording job arrival and departure times, measurement of service time data is quite straightforward. In addition, numerical computation of response time results is less costly for this model than the two-stage tandem queue.

As to service interval, two cases are considered. In the first case, the length of service interval (or TU) is one hour. This is compatible with the pricing policy of infrastructure providers such as Amazon EC2 [1], Google App Engine [5] and Microsoft Azure [2] where resource usage is charged on an hourly basis. In Chapter 4, we presented a time-dependent analysis for the response time distribution of the MMPP/PH/1 (FCFS) model. Our results indicate that for the three examples considered in that chapter, steady-state response time distribution can be used for resource provisioning when the service interval is longer than two minutes. For MMPP and PH parameter values similar to those used in Chapter 4, the minimum time period for steady-state results to be a good approximation may be longer than two minutes, yet it is almost certainly less than one hour. Therefore, steady-state results are used in our investigation.

In the second case, the service interval is small in the sense that it is comparable to the average sojourn time in an MMPP state. For parameter values representative of those used in our investigation, the average sojourn time is less than two minutes. This implies that steady-state results for the MMPP/PH/1 (FCFS) model cannot be used, and other queueing theory results need to be considered.

In this chapter, we present results on resource provisioning based on the single-server model for a service interval of one hour. Resource provisioning using the two-stage tandem queue is considered in Chapter 6; the results are also for the case of a one-hour service interval. Resource provisioning strategies for small service intervals are presented in Chapter 7.

The rest of this chapter is organized as follows. In Section 5.1, we derive analytic results

for the steady-state response time CDF of the MMPP/PH/1 (FCFS) queue. In Section 5.2, the procedure to determine the service time distribution is presented. In Sections 5.3, our resource provisioning methodology is discussed. Section 5.4 provides a description of the experimental environment used to evaluate our resource provisioning methodology. The evaluation results are presented in Section 5.5. In Section 5.6, we study the impact of traffic parameters on the total required capacity to meet a performance target. Finally, Section 5.7 concludes the chapter.

5.1 Steady-state analysis

Steady-state results for the response time CDF of the MMPP/PH/1 (FCFS) queue have been reported in [42, 76]. Instead of using these results, we develop an alternative derivation based on Markov chain analysis. Our derivation is more suitable for numerical computation. Recall from Sections 3.1.1 and 3.1.2 that MMPP and PH parameters are denoted, respectively, by λ_j and q_{ij}, and ζ_k and s_{kl}. We first define the state of the Markov chain and derive the state probabilities in Subsection 5.1.1, and then use the results to obtain the response time CDF in Subsection 5.1.2.

5.1.1 State probabilities

The MMPP/PH/1 (FCFS) queue can be viewed as a Markov chain $\mathcal{M}_S$ with state given by (i, n, k) where i is the MMPP state; $n \geq 0$ is the number of jobs in the system; and k is the state of the PH distribution for the job receiving service ($n = 0$ implies $k = 0$). For this Markov chain, the events that cause a transition out of state (i, n, k), and the corresponding transition rate and next state, are as follows:

1. A change in MMPP state from i to j - The transition rate is q_{ij} and the next state is (j, n, k).

2. An arrival - There are two cases. First, the system is empty at the instant of arrival. For this case, the current state is $(i, 0, 0)$. An arrival occurs at rate λ_i and the arriving job starts service in service state l with probability ζ_l, resulting in a transition rate equal to $\lambda_i\, \zeta_l$. The next state is $(i, 1, l)$. Secondly, the system is non-empty at the instant of arrival. The corresponding transition rate is λ_i, and the next state is $(i, n+1, k)$.

3. A change in the service state from a transient state k to another transient state l - The transition rate is s_{kl} and the next state is (i, n, l).

4. A departure - There are two cases. First, there is only one job in the system. For this case, the current state is $(i, 1, k)$. The transition rate is the same as the absorption rate from service state k, which is given by s_{k0}, and the next state is $(i, 0, 0)$. Secondly, there are two or more jobs in the system. Absorption of the job in service occurs at rate s_{k0}, and after the departure of this job, the next pending job starts service in service state l with probability ζ_l. The transition rate is therefore $s_{k0}\, \zeta_l$ and the next state is $(i, n-1, l)$.

A summary of the state transitions is given in Table 5.1.

Table 5.1: State transitions for $\mathcal{M}_S$

Event	Condition	Current state	Next state	Rate
MMPP state change	–	(i, n, k)	(j, n, k)	q_{ij}
Arrival	$n, k = 0$	$(i, 0, 0)$	$(i, 1, l)$	$\lambda_i\, \zeta_l$
	$n, k \geq 1$	(i, n, k)	$(i, n+1, k)$	λ_i
Service state change	$n, k \geq 1$	(i, n, k)	(i, n, l)	s_{kl}
Service completion	$n = 1$	$(i, 1, k)$	$(i, 0, 0)$	s_{k0}
	$n \geq 2$	(i, n, k)	$(i, n-1, l)$	$s_{k0}\, \zeta_l$

Let $P(i, n, k)$ be the steady-state probability of being in state (i, n, k); $P(i, n, k)$'s can be obtained by solving the balance equations for the Markov chain $\mathcal{M}_S$ with transitions shown in Table 5.1.

5.1.2 Response time distribution

We now derive the response time CDF under FCFS discipline using the state probabilities $P(i, n, k)$ obtained in Subsection 5.1.1. Consider a "tagged" job arriving at the system. Under FCFS discipline, its response time is given by the unfinished work in the system immediately after its arrival (this includes the service time of the tagged job). Jobs arriving after the tagged job will have no impact on the response time. These jobs are excluded from the analysis by cutting off the arrival process. From this point on, the behavior of the model can be characterized by a Markov chain $\mathcal{M}'_S$ with state (n, k) where n is the number of jobs in the system and k is the state of the PH distribution for the job receiving service ($n = 0$ implies $k = 0$). The transition rates of $\mathcal{M}'_S$ are the same as those discussed under events 3 and 4 in Subsection 5.1.1, i.e., events that are not caused by the arrival process. A summary of state transitions, and the corresponding transition rates and next states, are shown in Table 5.2.

Table 5.2: State transitions for $\mathcal{M}'_S$

Event	Condition	Current state	Next state	Rate
Service state change	$n, k \geq 1$	(n, k)	(n, l)	s_{kl}
Service completion	$n = 1$	$(1, k)$	$(0, 0)$	s_{k0}
	$n \geq 2$	(n, k)	$(n-1, l)$	$s_{k0}\,\zeta_l$

The Markov chain $\mathcal{M}'_S$ has the following properties: (i) It is an absorbing Markov chain where state $(0, 0)$ is the absorbing state and every other state is transient. (ii) The absorption time from a transient state to the absorbing state follows a PH distribution; the transition rate matrix of this PH distribution is obtained from the transition rate matrix of $\mathcal{M}'_S$ by eliminating the row and column corresponding to the state $(0, 0)$.

For the tagged job, the probability that the Markov chain $\mathcal{M}_S$ is in state (i, n, k) when it

arrives is given by

$$A(i,n,k) = \frac{\lambda_i}{\lambda}\, P(i,n,k)$$

where $\lambda = \sum_i \theta_i \lambda_i$ is the average arrival rate; θ_i is the long-term fraction of time that MMPP spends in state i. The state of the Markov chain $\mathcal{M}'_S$ immediately after the arrival is

$$(n+1,k) \text{ when } n \geq 1$$

and

$$(1,l) \text{ with probability } \zeta_l \text{ when } n = 0$$

The response time of the tagged job is the same as the unfinished work immediately after its arrival. It is given by the time to absorption of $\mathcal{M}'_S$ from the state immediately after the arrival. Let $r(x)$ be the response time CDF and $G_{n,k}(x)$ be the CDF of the time to absorption of $\mathcal{M}'_S$ from state (n,k). Using the law of total probability, we have

$$\begin{aligned} r(x) \quad &= \textstyle\sum_i \sum_l A(i,0,0)\, \zeta_l\, G_{1,l}(x) \\ &+ \textstyle\sum_i \sum_{n\geq 1} \sum_k A(i,n,k)\, G_{n+1,k}(x) \end{aligned} \tag{5.1}$$

From Equation 5.1, we observe that $r(x)$ is a linear combination of PH distributions $G_{n,k}(x)$, and hence is of PH itself.

Numerical computation of state probabilities $P(i,n,k)$ and PH distributions $G_{n,k}(x)$ can be done using standard MATLAB libraries. One also needs to set an upper bound on n so that the number of states (i,n,k) remains finite. The upper bound N_u is selected in such a way that the probability of having $n > N_u$ jobs in system is negligible.

5.2 Service time

For the single-server model, the service time is given by the sum of the service times at the two VM's hosting the web and database servers. This service time is affected by the capacities allocated to the web and database VM's. Capacity often includes resources such as CPU, memory and storage. To process a job, these resources interact in a sophisticated manner. The simplification that we use is to treat the resources obtained for the web and database VM's as a black-box. The service time at the black-box is the same as the response time when there is no queueing, and can be readily obtained by submitting one job at a time to the system and recording its arrival and departure times. In the following, we present a procedure to determine the service time distribution for a given combination of web and database server capacities, denoted by C_w and C_d, respectively:

1. VM's with capacities C_w and C_d are obtained from the infrastructure provider.
2. The web and database servers are deployed on these VM's.
3. A load generator is used in single-user mode to submit a total of N_J jobs to the web server. A job is submitted only after the response for the previous job has been received. This ensures that at any given time, at most one job is present in the system, and hence there is no queueing. The response time of each job (or the job's service time) is measured. The result is a sample X of N_J service times.
4. GFIT tool [77] is used to fit a PH distribution to X.

5.3 Methodology

In this section, we present our methodology for determining the required capacity to meet a performance target. An infrastructure provider typically charges a subscriber based on capacity level and duration of usage. We assume that the infrastructure provider offers N_C levels of capacity denoted by $\mathbb{B} = \{B_1, B_2, \cdots, B_{N_C}\}$. As to duration of usage, the infrastructure provider specifies a TU for charging of resource usage. A resource is billed for the number of TU's it is held by a subscriber; a partial TU is billed as a full TU. We denote by $c(B_j)$ the cost incurred by a subscriber to obtain the capacity $B_j \in \mathbb{B}$ for one TU. This cost typically increases with capacity size. We assume that members of $\mathbb{B}$ are ordered in increasing cost (or equivalently increasing size), i.e., $c(B_{j-1}) < c(B_j)$, $2 \leq j \leq N_C$.

Consider a service interval of length one TU (TU = one hour). Assume that job arrivals follow an MMPP with known parameter values. The subscriber's objective is to determine the web and database server capacities, denoted by C_w and C_d, respectively, to meet a performance target of the form Pr[response time $\leq x$] $\geq \beta$ over all jobs processed in the service interval, while keeping the cost at a minimum. For the case of a two-tier application, this objective is fulfilled by obtaining a solution to the following optimization problem:

$$\text{minimize} \quad c(C_w) + c(C_d)$$

$$s.t. \quad \Pr[\text{response time} \leq x] \geq \beta$$
$$C_w, C_d \in \mathbb{B}$$

Since the service time distribution and cost are affected by the manner an infrastructure provider defines and charges for units of capacity, the optimization problem should be viewed as being formulated for a specific infrastructure provider, referred to as the *target provider*.

We use a search based on a variant of the hill climbing algorithm called the steepest gradient method [59] to obtain a solution to the above optimization problem. The search is conducted over the space of available capacities for the web and database servers, i.e., $(C_w, C_d) \in \mathbb{B}^2$. It starts with random initial point $(C_w, C_d) = (B_i, B_j)$, $1 \leq i, j \leq N_C$. The feasibility of this point is verified following the steps shown in Figure 5.1.

1. Experiments are performed to obtain service time data for capacity allocations $C_w = B_i$ and $C_d = B_j$, and a PH distribution is fit to the measurement data.
2. This PH distribution along with the given MMPP are used as input to the single-server model to obtain $\beta^* = \Pr[\text{response time} \leq x]$.
3. If $\beta^* \geq \beta$, the allocation $C_w = B_i$ and $C_d = B_j$ is a feasible solution.

Figure 5.1: Steps to verify the feasibility of capacity allocation $(C_w, C_d) = (B_i, B_j)$

Consider first the case where $(C_w, C_d) = (B_i, B_j)$ is a feasible solution. We check for the possibility of meeting the performance target with a lower capacity level for either of web or database servers (or both). This is achieved by repeating the steps in Figure 5.1 for three neighboring points $(C_w, C_d) = (B_{i-1}, B_j)$, (B_i, B_{j-1}) and (B_{i-1}, B_{j-1}). A better solution exists if any of these points leads to a feasible solution. In case where multiple solutions exist, the one with the lowest cost is selected. A tie-breaking rule is required if there are two or more solutions with the same lowest cost. In our approach, we select the solution with a lower capacity level for the server (web or database) that has the smaller average service time under B_{N_C} (i.e., the largest available capacity). The rationale behind this approach is as follows. Intuitively, there is more room to reduce the capacity (or cost) along the dimension corresponding to the server with the smaller average service time under B_{N_C}. After a new solution is found, the search continues with checking the neighbors of this solution until no further improvement is viable.

This method finds a local minimum. If the optimization problem is convex, the local minimum is also the global minimum. However, we were not able to prove that the optimization problem is convex. One approach to address this issue is to repeat the search multiple times; each search starts at a different initial point and yields a local minimum. The smallest of these values is taken as an approximation for the global minimum. This approach is called shot-gun hill climbing, and is often used as a heuristic in non-convex optimization [59].

Determining the service time distribution for (C_w, C_d) is significantly more costly than computing the response time CDF. We therefore define the time complexity of a search as the number of times the service time distribution must be determined, which is the same as the number of points visited during the search.

An upper bound on the number of points visited during a search corresponds to a scenario where the optimal solution is $(C_w, C_d) = (B_1, B_1)$, i.e., the smallest value for both web and database server capacities. We therefore consider the computation complexity for this scenario. Without loss of generality assume that $j \geq i$ (the argument for $i > j$ is similar).

Suppose that $i > 1$ and $j > 1$. In the first iteration, the three neighbors examined are (B_{i-1}, B_j), (B_i, B_{j-1}) and (B_{i-1}, B_{j-1}). All three neighbors are feasible solutions because their capacity levels are not smaller than (B_1, B_1), and (B_1, B_1) is a feasible solution. Among the three neighbors, (B_{i-1}, B_{j-1}) results in maximal reduction in cost. Therefore, the first iteration returns (B_{i-1}, B_{j-1}) as an intermediary solution. Following a similar argument, the first $i-1$ iterations of the search yield the following sequence of intermediary solutions:

$$(B_{i-1}, B_{j-1}), (B_{i-2}, B_{j-2}), \cdots, (B_1, B_{j-i+1})$$

For each iteration, three points are examined by the search, resulting in a total of $3(i-1)$ points. From this point on, the capacity of the web server cannot be decreased, and the solution sequence

is

$$(B_1, B_{j-i}), (B_1, B_{j-i-1}), \cdots, (B_1, B_1)$$

Only one point is examined for each solution, resulting in a total of $j-i$ points. The total number of points visited during the search is therefore given by $1+3(i-1)+(j-i) = j+2i-2 \leq 3N_C-2$ which is $O(N_C)$, i.e., linear in the number of available capacities.

Consider next the case where the random initial point $(C_w, C_d) = (B_i, B_j)$ is not a feasible solution. A maximum of five attempts are then made to find a feasible solution using different random initial points. If all these attempts fail, (B_{N_C}, B_{N_C}) is taken as the initial point, i.e., the largest capacity for both web and database servers. If this point fails as well, there is no solution to the optimization problem. The implication is that the available server capacities are not sufficient to meet the performance target under the given workload.

5.4 Experimental environment

In this section, we present the experimental environment used to evaluate the effectiveness of our resource provisioning methodology. This environment consists of a target provider, a subscriber and users of the subscriber's application.

5.4.1 Target provider

The target provider's computing resources consist of two nodes. Each node is configured with 4 GB of memory, 220 GB of storage space, and 2 AMD Athlon 64 CPU cores, each having 1024 KB of cache and operating at 1 GHz. The two nodes are connected through a 100 Mb/second Ethernet switch.

The target provider offers capacity in terms of VM's. A VM is bundled with 2 GB of memory,

20 GB of storage and a fraction of the processing power of a single CPU core. The VM's run the Ubuntu 12.4 operating system. Oracle VirtualBox 4.1 is used as the virtualization technology.

For the web application under consideration, we learned through experimentation that CPU is the bottleneck resource, i.e., the resource with the greatest impact on system performance. Therefore, we only consider capacities which differ in CPU processing power. The target provider offers $N_C = 20$ levels of capacities; these levels are $B_1 = 5\%, B_2 = 10\%, \cdots$ and $B_{N_C} = 100\%$ of the processing power of a CPU core. For convenience, we assume that the cost function $c(B_j)$ is linear with respect to B_j, i.e., $c(B_j) = \kappa B_j$, where κ is a constant. Our work can be easily extended to other types of cost functions.

Note that the experimental system described here is different from the one used in Chapter 3 to evaluate the effectiveness of MMPP as a traffic model. The reason is that the system in Chapter 3 is quite unstable; this raises concerns about its suitability for large number of experiments required to evaluate the effectiveness of our resource provisioning methodology.

5.4.2 Subscriber

The subscriber's application is an available implementation of the TPC-W benchmark [3]. It is based on a two-tier architecture with a front-end web server and a back-end database server. Apache Tomcat 6.0 and MySQL Server 5.5 are used as the web and database servers, respectively. Each application tier is hosted on a separate VM obtained from the target provider. These VM's reside in different nodes.

TPC-W defines three types of workload: browsing, shopping and ordering. Our investigation indicates that all three workloads place a disproportionately heavier load on the database server than the web server. In order to experiment with more balanced workloads on the two servers, we focus on the browsing mix and add a random delay to the web server for each transaction; different

delays result in different workloads. The three workloads we experiment with are presented in Table 5.3. In this table, $\bar{b}_w$ and $\bar{b}_d$ denote, respectively, the average service time on the web and database servers when both servers operate at full capacity of a VM, i.e., $C_w = C_d = B_{N_C}$. Workload 1 represents a slightly heavier load on the web server. Workloads 2 and 3, on the other hand, represent a moderately heavier and significantly heavier load on the database server, respectively.

Table 5.3: Workloads

Workload	Average delay (msec)	$\bar{b}_w/\bar{b}_d$
Workload 1	110	1.25
Workload 2	37	0.5
Workload 3	0	0.07

5.4.3 Users

The users are represented by a load generator which is hosted on a third node with resource profile similar to that described in Section 5.4.1. This load generator emulates multiple user sessions that concurrently submit jobs to the web server. A user session can be in two states, namely, waiting or idle. A waiting session is one that has submitted a job to the web server and is waiting for the response. An idle session, on the other hand, has received the response for its most recently submitted job and is in thinking state. Newly created sessions are considered as idle.

The load generator starts off by creating one idle session. A sequence of job arrival instants are then generated (in an offline manner) in accordance with an MMPP. This sequence spans a time interval of length one TU and is replayed in real time. At each arrival instant, the following actions are taken:

1. An idle session is selected at random. If all the sessions are waiting, a new session is created.

2. For the session selected or created at Step 1, an HTTP job is generated and submitted to the web server. This session is marked as waiting.

For any session, when a response is received, the corresponding session is marked as idle.

In our experiments, job arrivals are generated according to a two-state MMPP. States one and two can be viewed as moderate load and a burst of arrivals, respectively. A two-state MMPP is fully characterized by four parameters: λ_1, λ_2, $q_{12} = -q_{11}$ and $q_{21} = -q_{22}$. For convenience, denote q_{jj} by q_j, $j = 1, 2$. We note that $(-q_j)^{-1}$ is the average sojourn time in one visit to state j. We found it more informative to describe the parameters of a two-state MMPP as follows:

$$\begin{aligned} \rho &= \lambda_2/\lambda_1 \\ \gamma &= q_2^{-1}/q_1^{-1} \\ \eta &= -(q_1^{-1} + q_2^{-1}) \\ \lambda &= (q_2\lambda_1 + q_1\lambda_2)/(q_1 + q_2) \end{aligned} \tag{5.2}$$

We assume that arrivals occur at a higher rate when MMPP is in state two, i.e., $\lambda_2 > \lambda_1$, and on average, a period of moderate load is longer than that when a burst of arrivals occurs [35], i.e., $(-q_1)^{-1} > (-q_2)^{-1}$. Therefore, we only consider $\rho > 1$ and $\gamma < 1$. The parameter η is the average amount of time to complete one cycle of moderate load followed by a burst of arrivals. The value of this parameter indicates the time scale of variation in traffic; a value of tens of seconds is adequate for representing fine-scale variation. The parameter λ is the average arrival rate.

5.4.4 Provisioning delay

When a decision on capacity adjustment is made at the beginning of a service interval (or a decision point), there may be a delay before the new capacity level becomes effective; this delay is referred to as the *provisioning delay*. In general, the provisioning delay is defined as the time period from when a resource is requested from an infrastructure provider until when it becomes ready to use. In our experimental environment, a resource request takes the form of an adjustment in CPU capacity of a VM. We therefore define the provisioning delay as the amount of time required to adjust the CPU capacity of both web and database VM's. The provisioning delay should be small compared to the length of service interval.

VirtualBox uses a client-server architecture to create and manage VM's [7]. To adjust the CPU capacity of a VM, a client process invokes, in a synchronized manner, a remote procedure in the server process, which subsequently sets a cap on the percentage of CPU cycles per second allocated to that VM; this cap is an integer in the range 1-100. Let t be the execution time of this procedure call. Since procedure calls are synchronized, an upper bound for the provisioning delay is given by t. For our experimental setting, the average value of t obtained from 1000 experiments, each with different values of C_w and C_d chosen at random, is 0.6 seconds. Since the provisioning delay is not significant when compared to a service interval of one hour, it has little or no impact on the accuracy of the evaluation results.

5.5 Evaluation

The capacities obtained from the optimization problem in Section 5.3 are estimates of the required capacity. The following procedure is used to evaluate the accuracy of these estimates:

1. For a given set of MMPP parameters, a workload (see Table 5.3) and a performance target,

the optimization problem described in Section 5.3 is solved to determine the web and database server capacities $C_w, C_d \in \mathbb{B}$ to meet the performance target with minimal cost.

2. The web and database servers are deployed on VM's with capacities C_w and C_d, respectively.

3. Jobs are submitted to the web server for one hour in accordance with the MMPP, and the response time of each job is measured. From the measurement data, the fraction of jobs meeting the response time threshold x, denoted by β', is determined, and the metric $e = \beta' - \beta$ is computed; this metric represents the deviation from the target probability β.

4. Step 3 is replicated five times. The mean value of e over the five replications is used to evaluate the effectiveness of our resource provisioning methodology. A positive e indicates that the performance target is met. When e is negative, the capacity estimates are not sufficient to meet the performance target.

Three scenarios are considered in our evaluation; they correspond to the three workloads shown in Table 5.3. The default values of MMPP parameters for each workload are shown in Table 5.4. Unless stated otherwise, these MMPP parameters are used in our experiments.

Table 5.4: Default parameter values for a two-state MMPP

Workload	ρ	γ	η (second)	λ (jobs/second)
Workload 1	3	0.25	25	1
Workload 2	3	0.25	25	1.5
Workload 3	3	0.25	25	1.5

The values of x and β considered are 3 and 5 seconds, and 0.9 and 0.95, respectively. When determining the service time distribution in Section 5.2, the number of jobs submitted by the load generator (denoted by N_J) is set to 2000. When computing the steady-state response time CDF, the upper bound for the number of jobs in system (denoted by N_u) is set to 50. The

search procedure in Section 5.3 is repeated three times, and the best solution among the three local minimums is taken as the estimate of the required capacity. Obviously, a larger number of search repetitions increases the probability of finding the global optimum. However, a search can be potentially costly since the service time distribution needs to be determined for all the points visited during that search. We feel that repeating the search three times is a reasonable strategy considering the trade-off between computation cost and quality of the solution.

We next present our experimental results and discuss the effectiveness of our methodology.

5.5.1 Workload 1

We first consider the accuracy of capacity estimates for different values of the mean arrival rate λ. In Figures 5.2 and 5.3, the mean and 95% confidence interval of e are plotted against λ for different values of the response time threshold x and target probability β. The values of λ considered are 1, 1.5 and 2 jobs/second. We observe that e is positive, i.e., the performance target is met in all of the cases. The smallest and largest values of e for $\beta = 0.9$ are 0.02 and 0.076, respectively. The corresponding values for $\beta = 0.95$ are 0.012 and 0.028.

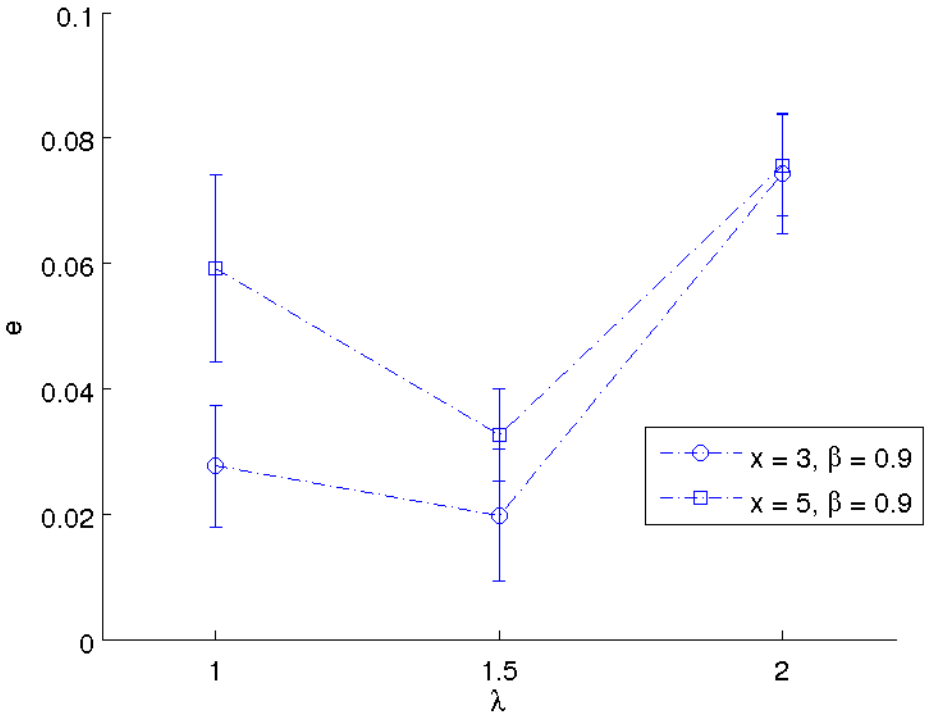

Figure 5.2: e vs. λ

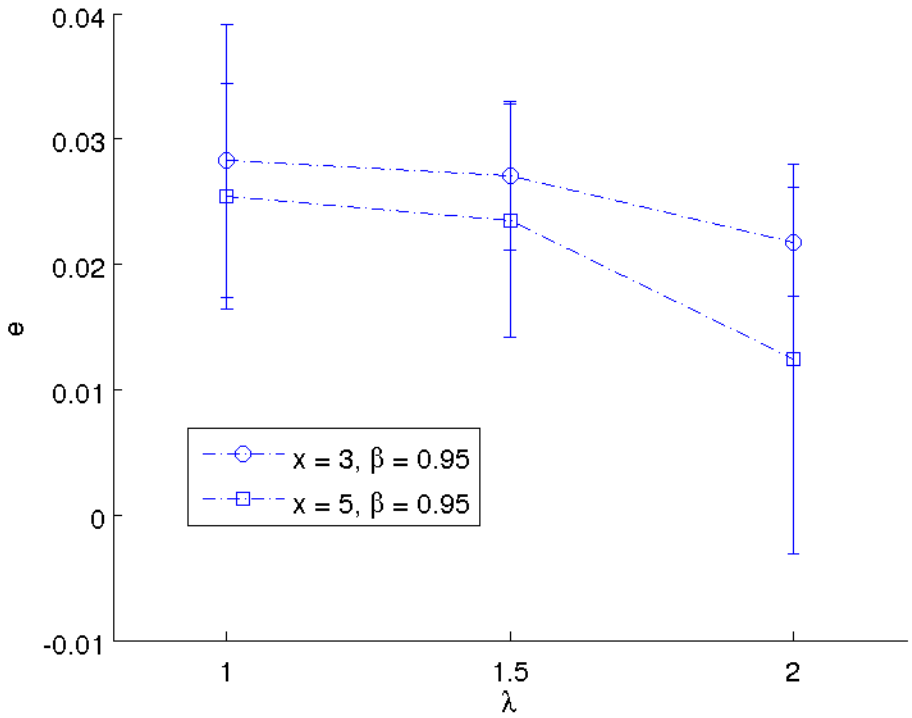

Figure 5.3: e vs. λ

A non-zero value of e indicates inaccuracy in resource estimate. We identify a number of reasons for this inaccuracy. First, there are sources of inaccuracy that potentially lead to over-provisioning:

1. The optimization problem may yield a local minimum instead of the global minimum, resulting in additional capacities being provisioned.

2. VM capacities in $\mathbb{B}$ are defined in increments of 5%. It is unlikely that any capacity setting (C_w, C_d) would result in the ideal outcome of $\Pr[\text{response time} \leq x] = \beta$. The optimization problem looks for the smallest setting among capacities in $\mathbb{B}$ such that $\Pr[\text{response time} \leq x] > \beta$; this again may lead to over-provisioning.

3. In the MMPP/PH/1 (FCFS) model, jobs are processed one at a time, with service time equal to the aggregate service time at the web and database VM's. In practice, multiple jobs may be processed in parallel by the two VM's, which tends to decrease the response time. A larger response time predicted by the model may lead to a higher resource estimate.

Secondly, the assumptions made in the development of the MMPP/PH/1 (FCFS) model would lead to inaccuracies in predicting the response time CDF; such inaccuracies may result in under or over-provisioning of resources. These assumptions include:

1. The distribution of service time is obtained from measurement data collected under single-user mode. When there are multiple jobs in system, the service time may be affected by such factors as synchronization or contention for database resources.

2. The complex interaction between computing resources in a virtualized environment and different scheduling disciplines at various system resources are simply represented by a single server with FCFS discipline.

3. The flow of a job between web and database servers is not captured by the single-server model.

The sources of inaccuracy mentioned above explain the deviation of e from the ideal value of zero. As mentioned earlier, the value of e is positive (or the performance target is met) for all the cases shown in Figures 5.2 and 5.3. A positive value of e indicates that more resources than required are provisioned. This implies that the factors contributing to over-provisioning more than cancel the impact of those leading to under-provisioning.

We next consider the impact of ρ on the accuracy of capacity estimates. A larger ρ means a larger ratio of the arrival rate in MMPP state two to that in state one, which can be viewed as a higher degree of burstiness. In Figures 5.4 and 5.5, e is plotted against ρ for different values of x and β. The values of ρ considered are 3, 5 and 7. We observe that the performance target is met in 10 of the 12 cases considered. The smallest and largest values of e for $\beta = 0.9$ are 0.005 and 0.062, respectively. The corresponding values for $\beta = 0.95$ are -0.008 and 0.042.

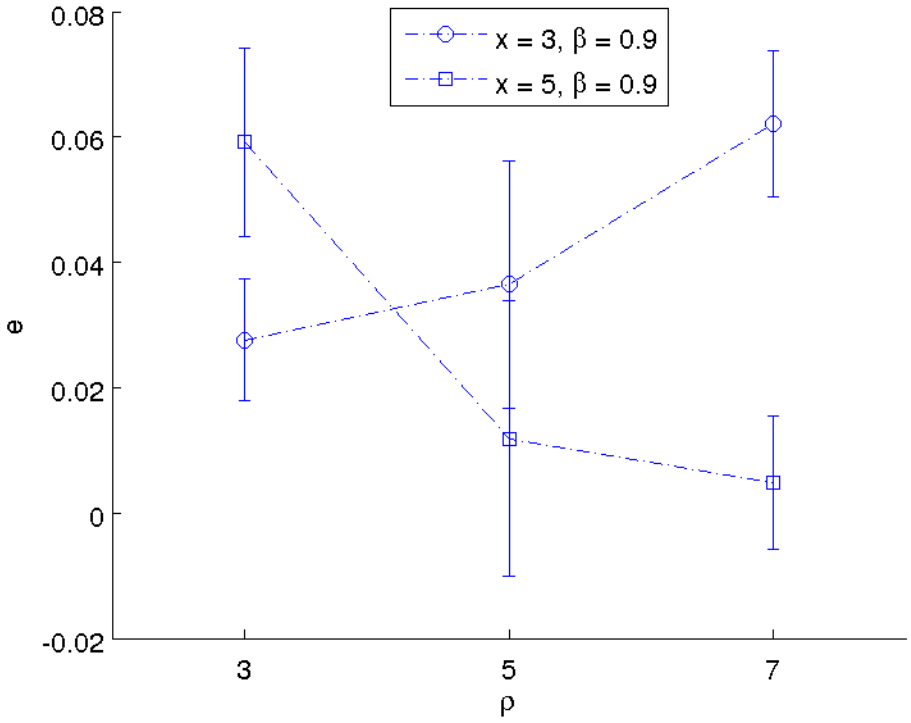

Figure 5.4: e vs. ρ

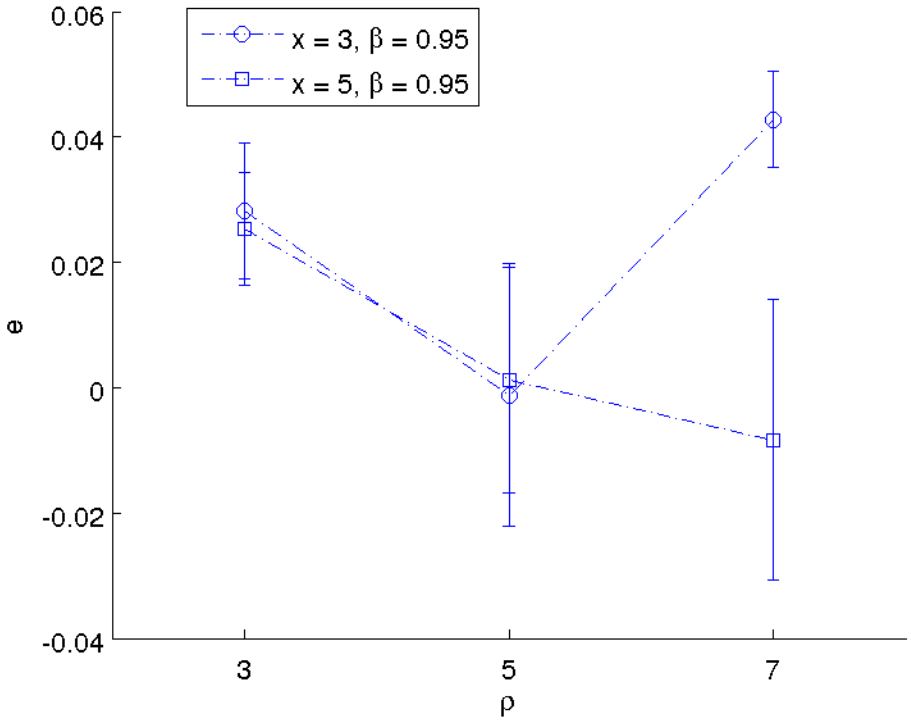

Figure 5.5: e vs. ρ

A negative value of e implies that the capacity estimate from our methodology is not sufficient to meet the performance target. For the two cases where $e < 0$, however, the deviation from the target probability is small ($|e| < 0.01$). Experiments are performed to gain insight into the capacity level that needs to be added in order to meet the performance target. In general, selecting the server with larger average service time under B_{N_C} for capacity increase should lead to more significant performance improvement because this server has a heavier load. Since $\bar{b}_w > \bar{b}_d$ for Workload 1 (see Table 5.3), we select the web server for capacity increase. The results indicate that an increase of 5% in web server capacity (i.e., the smallest amount defined by the target provider) is sufficient to meet the target in both cases.

In Figures 5.6 and 5.7, e is plotted against the average cycle time η for different values of x and β. As mentioned earlier, a value of tens of seconds for η is adequate for representing fine-scale variation in traffic. The values of η considered are 5, 25 and 50 seconds. We observe that the performance target is met in all of the cases. The smallest and largest values of e for $\beta = 0.9$ are 0.028 and 0.08, respectively. The corresponding values for $\beta = 0.95$ are 0.003 and 0.039.

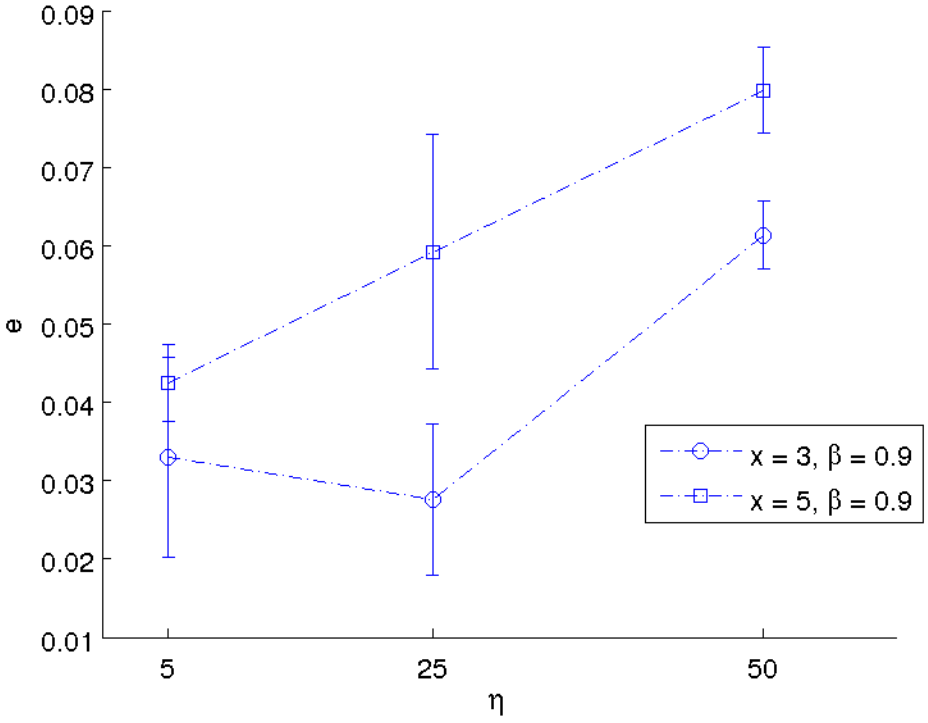

Figure 5.6: e vs. η

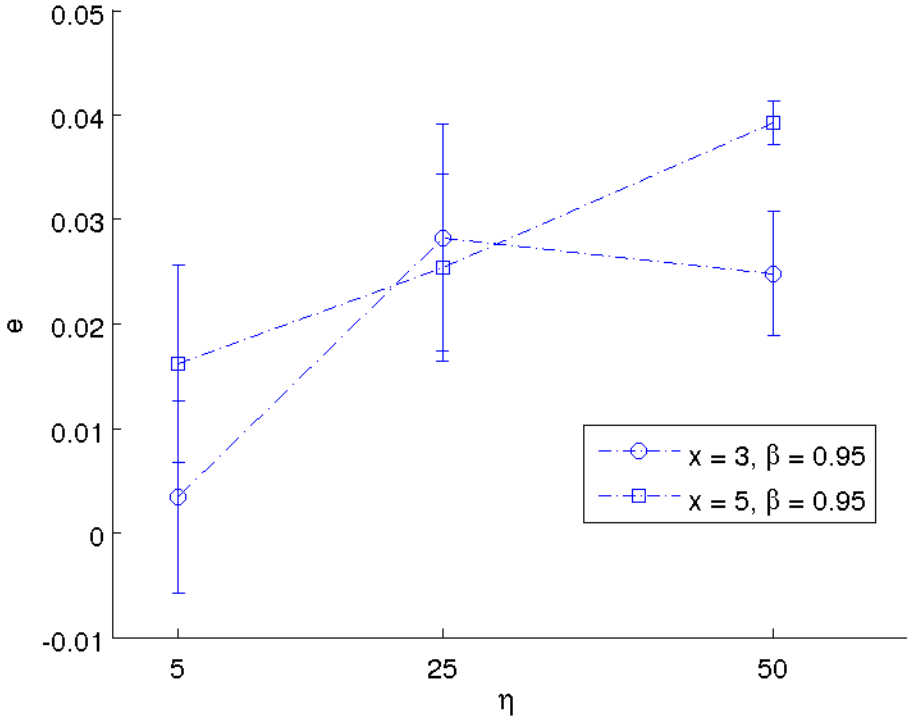

Figure 5.7: e vs. η

We also conducted experiments to investigate the accuracy of the model for different values of γ. The values of γ considered are 0.25, 0.5 and 0.75. These results are shown in Figures 5.8 and 5.9. Again, we observe that the performance target is met in all of the cases. The smallest and largest values of e for $\beta = 0.9$ are 0.028 and 0.08, respectively. The corresponding values for $\beta = 0.95$ are 0.018 and 0.038.

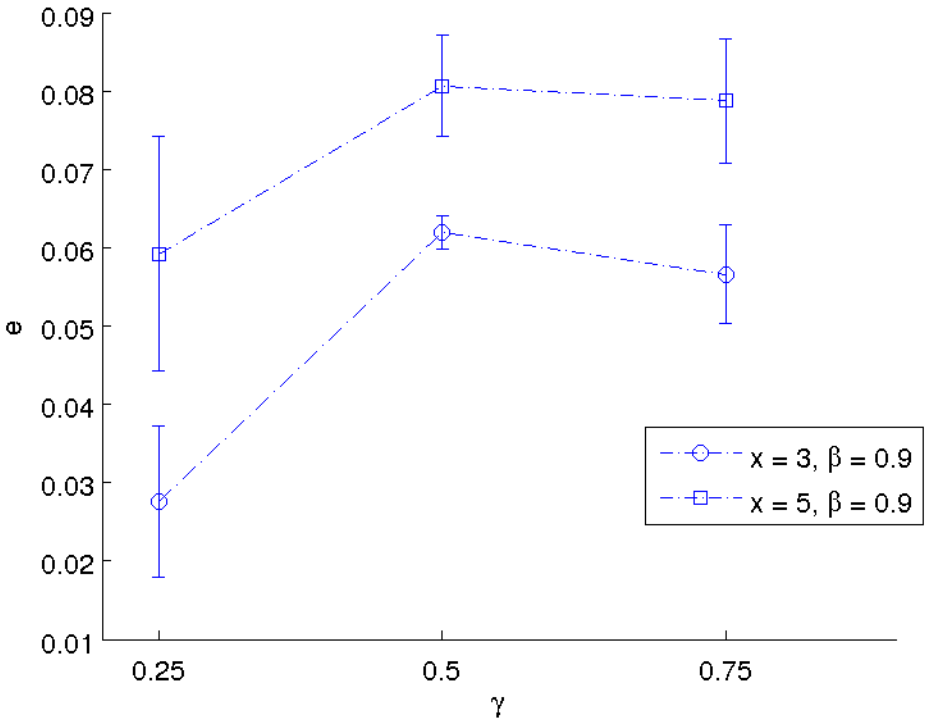

Figure 5.8: e vs. γ

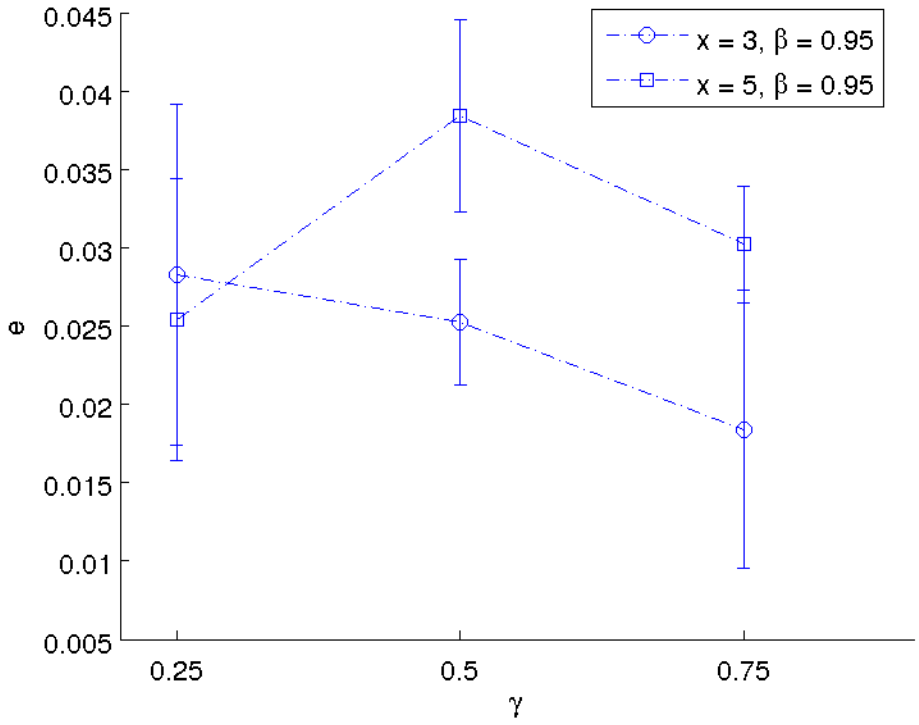

Figure 5.9: e vs. γ

5.5.2 Workloads 2 and 3

In Figures 5.10 and 5.11, e is plotted against λ for different values of x and β for Workload 2. The corresponding results for Workload 3 are shown in Figures 5.12 and 5.13. Since the load on the web server is smaller for Workloads 2 and 3 (compared to Workload 1), we may experiment with a larger range of values for λ. The values of λ considered are 1.5, 2 and 2.5 jobs/second. For both workloads, we observe that the performance target is met in all but one case. The smallest and largest values of e for $\beta = 0.9$ are, respectively, 0.006 and 0.071 (for Workload 2), and 0.04 and 0.09 (for Workload 3). The corresponding values for $\beta = 0.95$ are -0.009 and 0.02 (for Workload 2), and 0.013 and 0.041 (for Workload 3). For the only case that the performance target is not met (i.e., $\lambda = 2.5$ jobs/second, $x = 5$ seconds and $\beta = 0.95$ in Figure 5.11), the deviation from the target probability is small ($|e| < 0.01$). Again, we select a server for capacity increase based on the values of $\bar{b}_w$ and $\bar{b}_d$ in Table 5.3; since $\bar{b}_w < \bar{b}_d$ for Workload 2, the database server is selected. Our experiments indicate that an increase of 5% in database server capacity (i.e., the smallest amount defined by the target provider) is sufficient to meet the target.

Results on e for different values of ρ, γ and η have also been obtained for workloads 2 and 3. These results lead to similar conclusions and are not presented here.

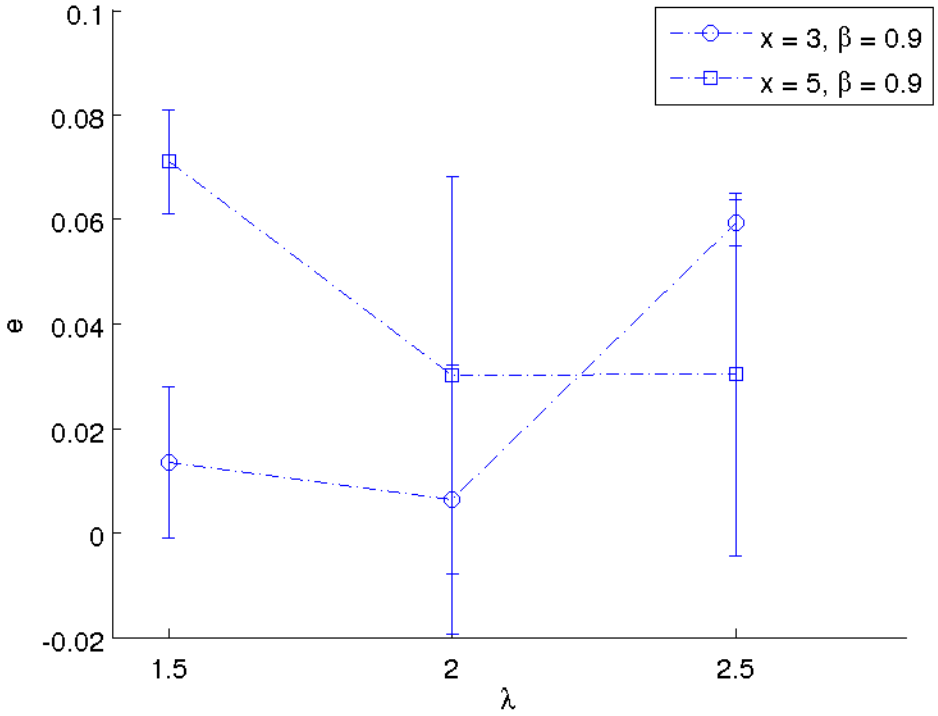

Figure 5.10: e vs. λ

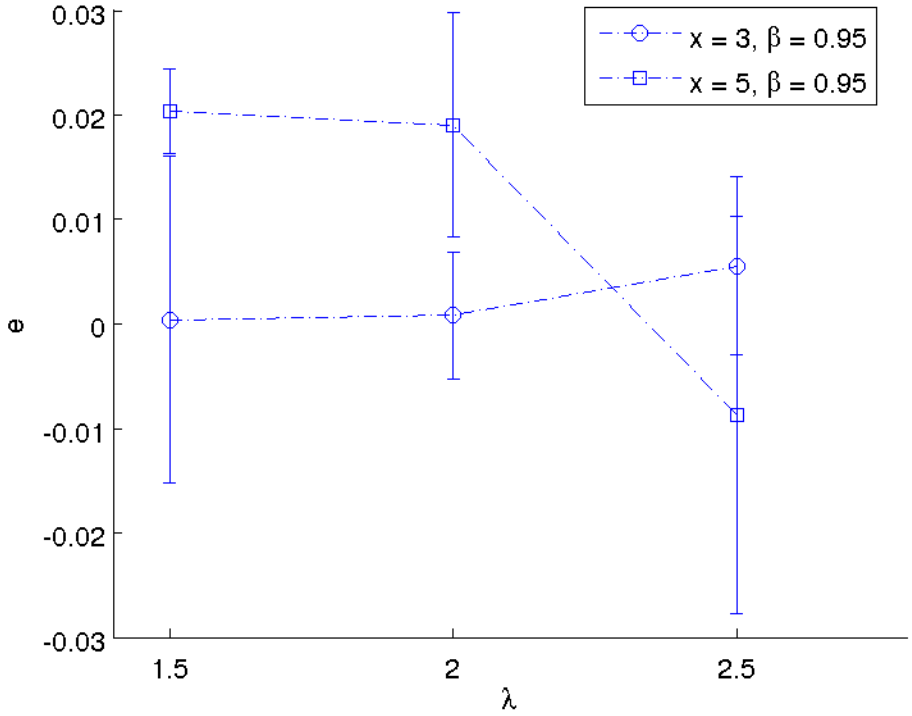

Figure 5.11: e vs. λ

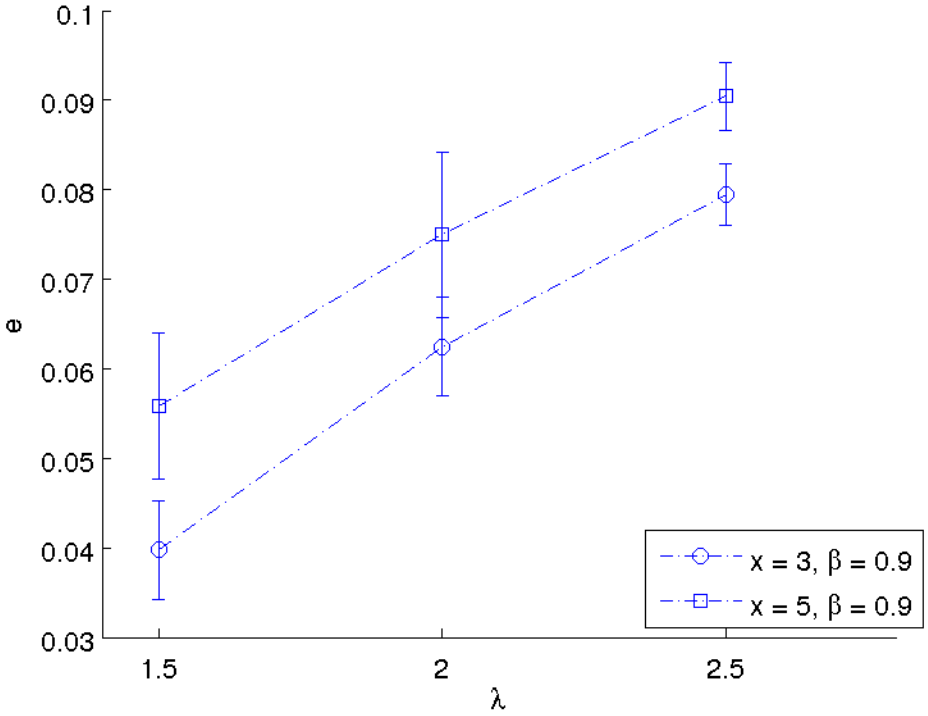

Figure 5.12: e vs. λ

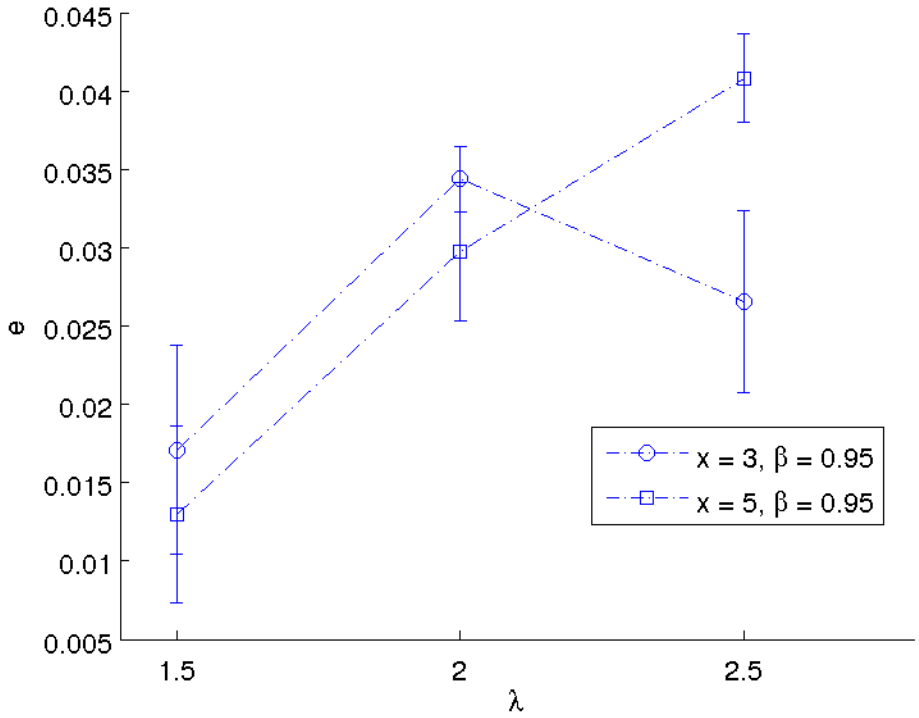

Figure 5.13: e vs. λ

5.5.3 Degree of over-provisioning

The results presented in Subsections 5.5.1 and 5.5.2 show that e is quite large for some of the cases. As an example, for $\beta = 0.9$, the largest values of e observed in Figures 5.2 and 5.4 are 0.076 and 0.062, respectively. This implies that the corresponding measured values for Pr[response time $\leq x$] are 0.976 and 0.962, while the target probability is 0.9. As another example, the largest values of e for $\beta = 0.95$, as observed in Figures 5.3 and 5.5, are 0.028 and 0.042, respectively. A large value of e indicates that an excessive amount of resources are provisioned. To gain insight into the level of excessive resources, a metric called the *degree of over-provisioning* (denoted by OV) is defined. Suppose an exhaustive search (based on response time data obtained from system measurement) is performed to determine the smallest total capacity, denoted by C_e, to meet the performance target. Also, let C_o be the corresponding minimal total capacity obtained from the optimization problem. The degree of over-provisioning is defined as follows:

$$OV = \frac{C_o - C_e}{C_e} \times 100$$

The computation cost of C_e is tremendous since an exhaustive search is performed. For this reason, we present results on OV for only a selected number of cases. Intuitively, OV should be higher for larger values of e. We hence place emphasis on cases where e is large. The results are shown in Tables 5.5 and 5.6. It is observed that the value of OV ranges from 14% to 37.5% (when $\beta = 0.9$) and from 5% to 16% (when $\beta = 0.95$). As to the other cases considered in Subsections 5.5.1 and 5.5.2, the values of OV are not expected to be significantly larger than those presented in Tables 5.5 and 5.6. We also feel that these OV values are representative of the degree of over-provisioning when our methodology is used to obtain the resource estimate.

Table 5.5: Degree of over-provisioning for $\beta = 0.9$

Parameter values	e	C_o	C_e	OV
Workload 1				
$\lambda = 1$, $\eta = 50$, $\gamma = 0.25$, $\rho = 3$, $x = 5$	0.08	165	120	37.5%
$\lambda = 2$, $\eta = 25$, $\gamma = 0.25$, $\rho = 3$, $x = 5$	0.076	195	160	22%
Workload 2				
$\lambda = 1.5$, $\eta = 25$, $\gamma = 0.25$, $\rho = 3$, $x = 5$	0.071	135	100	35%
$\lambda = 2.5$, $\eta = 25$, $\gamma = 0.25$, $\rho = 3$, $x = 3$	0.059	165	145	14%
Workload 3				
$\lambda = 2.5$, $\eta = 25$, $\gamma = 0.25$, $\rho = 3$, $x = 5$	0.09	110	95	16%

Table 5.6: Degree of over-provisioning for $\beta = 0.95$

Parameter values	e	C_o	C_e	OV
Workload 1				
$\lambda = 1$, $\eta = 25$, $\gamma = 0.25$, $\rho = 7$, $x = 3$	0.042	200	180	11%
$\lambda = 1$, $\eta = 25$, $\gamma = 0.25$, $\rho = 3$, $x = 3$	0.028	180	155	16%
Workload 2				
$\lambda = 1.5$, $\eta = 25$, $\gamma = 0.25$, $\rho = 3$, $x = 5$	0.02	135	120	12.5%
Workload 3				
$\lambda = 2.5$, $\eta = 25$, $\gamma = 0.25$, $\rho = 3$, $x = 5$	0.041	110	105	5%

Considering the sources of inaccuracy in our methodology, we believe such levels of over-provisioning are acceptable. Also, a reasonable level of over-provisioning may not be undesirable as the extra capacity provides protection against deviation from the expected workload.

5.5.4 Summary

Our experimental results indicate that the performance target is met for most of the cases considered. In the three cases where the performance target is violated, the deviation from the target probability is small ($|e| < 0.01$), and the problem can be alleviated by adding 5% (i.e., the small-

est capacity increment defined by the target provider) to the server selected for capacity increase. We thus conclude that our methodology is effective. We also conclude that the MMPP/PH/1 (FCFS) model is adequate in predicting the response time CDF, despite the fact that the web application is running on two VM's, and non-FCFS discipline may be used at system resources.

The results also indicate that the capacities obtained from our methodology can be excessive, i.e., the performance target can be met with a lower capacity level. The degree of over-provisioning is less than 37.5% among the selected cases where the value of e is large. We believe such degree of over-provisioning is not overly excessive and protects the subscriber against deviations from expected workload.

5.6 Impact of traffic characteristics on required capacity

In this Section, we study the relationship between traffic parameters and the total required capacity to meet a performance target. Our focus is on the impact of the mean arrival rate and traffic burstiness on the required capacity. We start this section by a brief discussion of the descriptor used to quantify the burstiness of a two-state MMPP. This facilitates understanding of the observed trends when the total required capacity is plotted against different traffic parameters.

5.6.1 Index of dispersion

Several descriptors have been used to quantify the burstiness in web application traffic. For example, in [27], burstiness is defined in terms of the correlation between the service demand of successive sessions. In our investigation, we aim to generate different levels of burstiness by controlling the parameters of the arrival process. Therefore, a burstiness descriptor based on the statistical properties of the arrival process is preferred. One such descriptor is the index of

dispersion for counts (IDC) [62], which is defined as

$$\text{IDC} = \lim_{t \to \infty} \frac{\text{Var}[N_t]}{\text{E}[N_t]}$$

where N_t is a random variable denoting the number of arrivals in an interval of length t. IDC measures the long-term dispersion of the number of arrivals around the mean. A larger value of IDC means burstier traffic.

The expression for the IDC of a two-state MMPP has been given in [51]. It can be expressed as a function of the parameters introduced in Equation 5.2 as follows:

$$\text{IDC} = 1 + 2\lambda\eta \left(\frac{\gamma}{1+\gamma}\right)^2 \left(\frac{\rho - 1}{1 + \gamma\rho}\right)^2$$

As discussed in Section 5.4.3, the region of interest is characterized by $\rho > 1$, $\gamma < 1$, $\eta > 0$ and $\lambda > 0$. One can easily verify that over this region, IDC is an increasing function of λ, η and ρ, which means that a larger value of λ, η or ρ results in a higher degree of burstiness. Also, IDC is an increasing function of γ over $(0, \frac{1}{\sqrt{\rho}}]$ and decreases with γ over $[\frac{1}{\sqrt{\rho}}, 1)$.

5.6.2 Results and discussion

In this Subsection, we study the impact of traffic parameters on the total required capacity $(C_w + C_d)$ as obtained from the optimization problem. Only results for Workloads 1 and 2 are presented. Results for Workloads 3 are not shown as similar behavior was observed.

Figures 5.14, 5.15, 5.16 and 5.17 show, respectively, the total required capacity against λ, ρ, η and γ for Workload 1. Consider first the impact of λ. We observe that the total required capacity increases with λ. This is expected since λ is the mean arrival rate, and a higher arrival rate requires a higher capacity to meet the same performance target. In some cases, such as $\lambda = 2$

and $x = 3$, the total required capacity is the same for $\beta = 0.9$ and $\beta = 0.95$. This is because of the step size of 5% used in capacity provisioning.

We also observe that in general, more capacity is required for smaller values of x and larger values of β when the value of λ is fixed. This is quite intuitive because a smaller value of x or a larger value of β means a more demanding performance target.

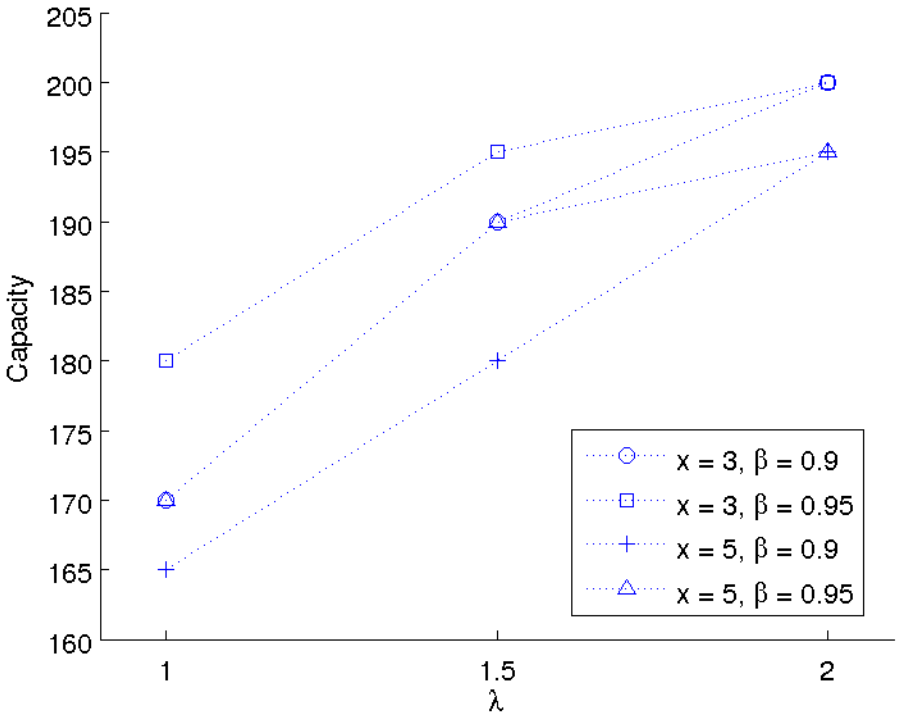

Figure 5.14: Total capacity versus λ

As to the impact of ρ, we observe that a larger ρ results in a larger total required capacity. This is because the level of traffic burstiness increases with ρ, and a higher capacity level is required to process the job arrivals during the bursty period.

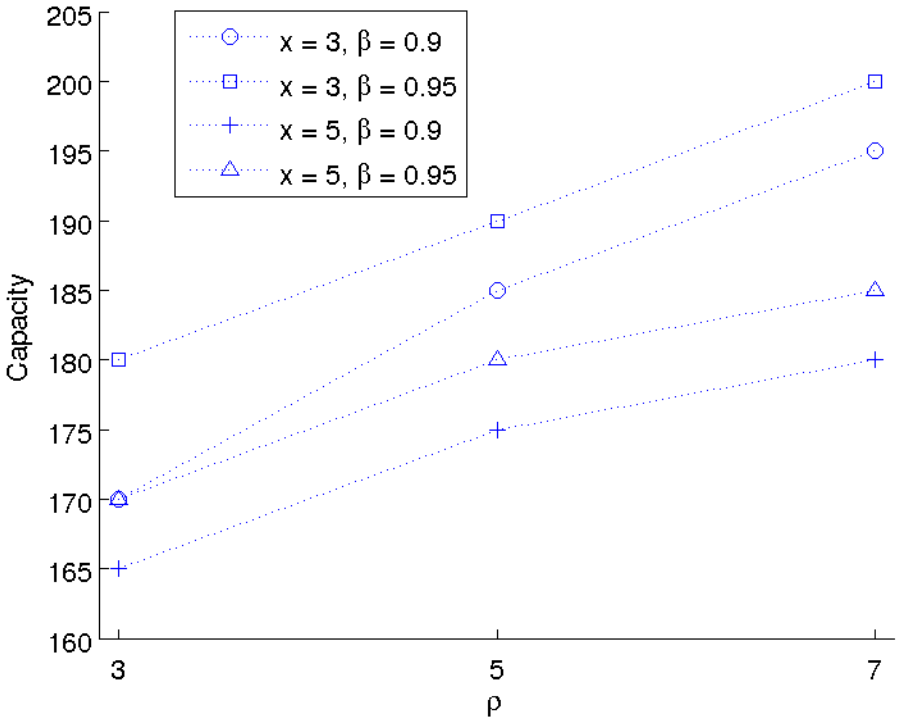

Figure 5.15: Total capacity versus ρ

As η increases, so does the traffic burstiness, and hence the total required capacity is expected to increase as well. This is the general trend in Figure 5.16. The only exception if when $x = 5$ and $\beta = 0.9$ where the total required capacity does not increase as η changes from 25 seconds to 50 seconds. This is because the capacity offered by the target provider is in 5% increments. A finer-grained increment should lead to an increase in total required capacity.

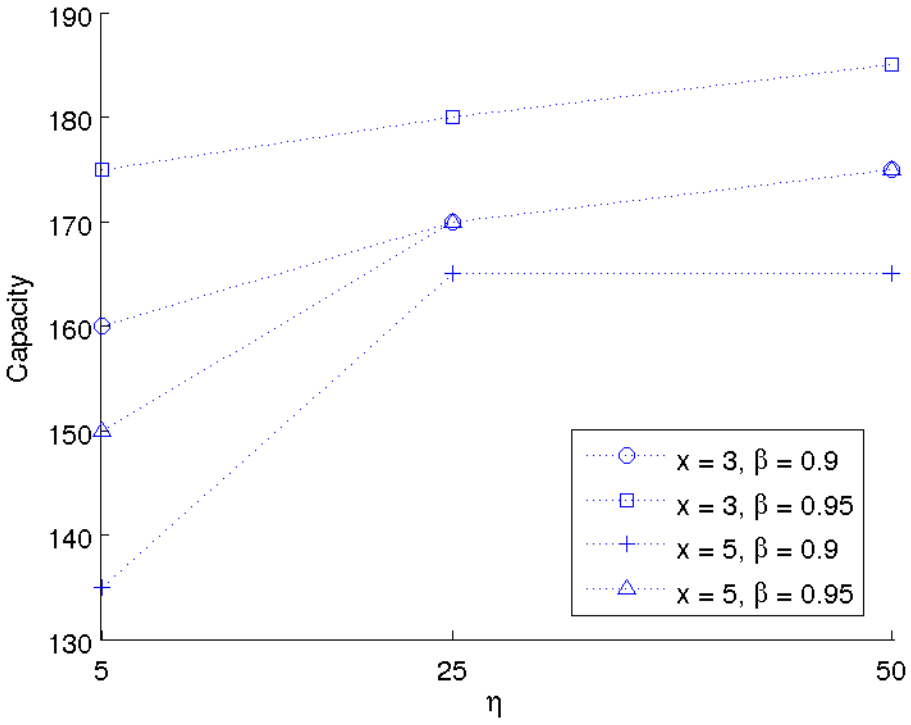

Figure 5.16: Total capacity versus η

In Figure 5.17, the total required capacity is plotted against γ (for $\rho = 3$). We observe that for most of the cases, the total required capacity increases with γ as γ is changed from 0.25 to 0.5, and decrease when γ is changed from 0.5 to 0.75. This is due to the fact that at $\rho = 3$, traffic burstiness increases when $\gamma \in (0, 0.58)$ and decreases when $\gamma \in (0.58, 1)$. In two of the cases considered ($x = 5$ and $\beta = 0.9, 0.95$) the total required capacity does not change as γ is increased from 0.25 to 0.5; this is again because of the step size of 5% in capacity definition.

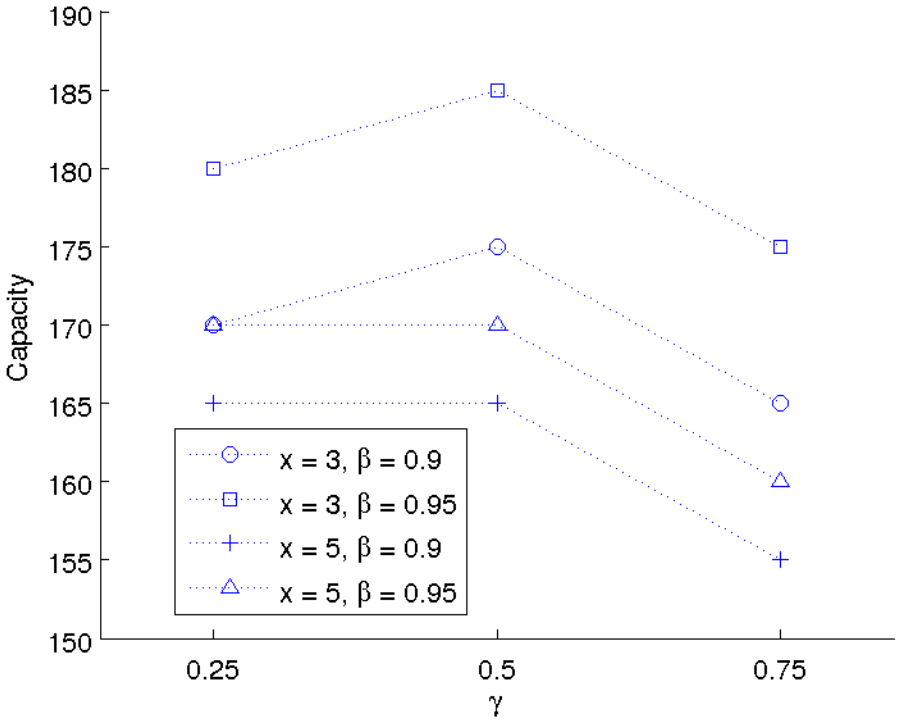

Figure 5.17: Total capacity versus γ

We also plot in Figures 5.18, 5.19, 5.20 and 5.21 results for the total required capacity as a function of different traffic parameters for Workload 2. Similar observations to those for Workload 1 are made in all four graphs, namely, a larger λ, ρ or η typically results in a larger total required capacity. An exception is that in most of the cases considered, no change in the total required capacity is observed as γ is increased.

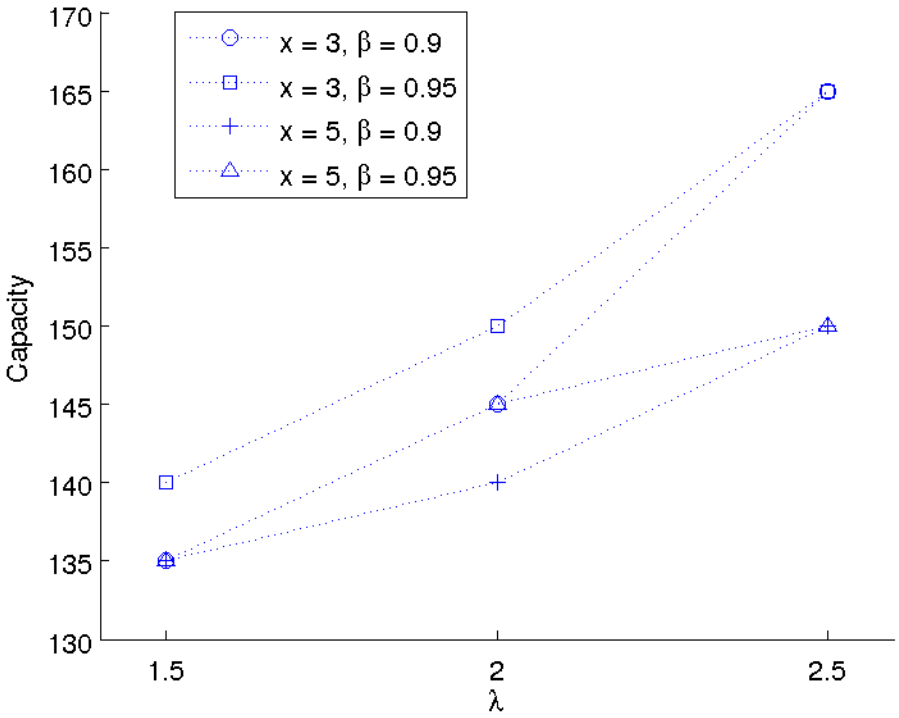

Figure 5.18: Total capacity versus λ

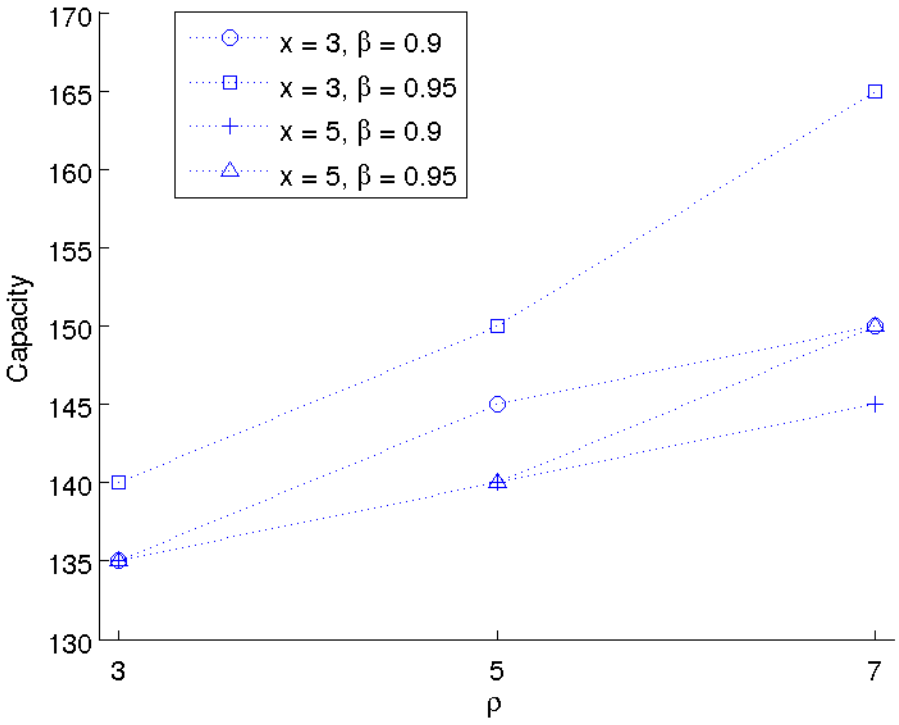

Figure 5.19: Total capacity versus ρ

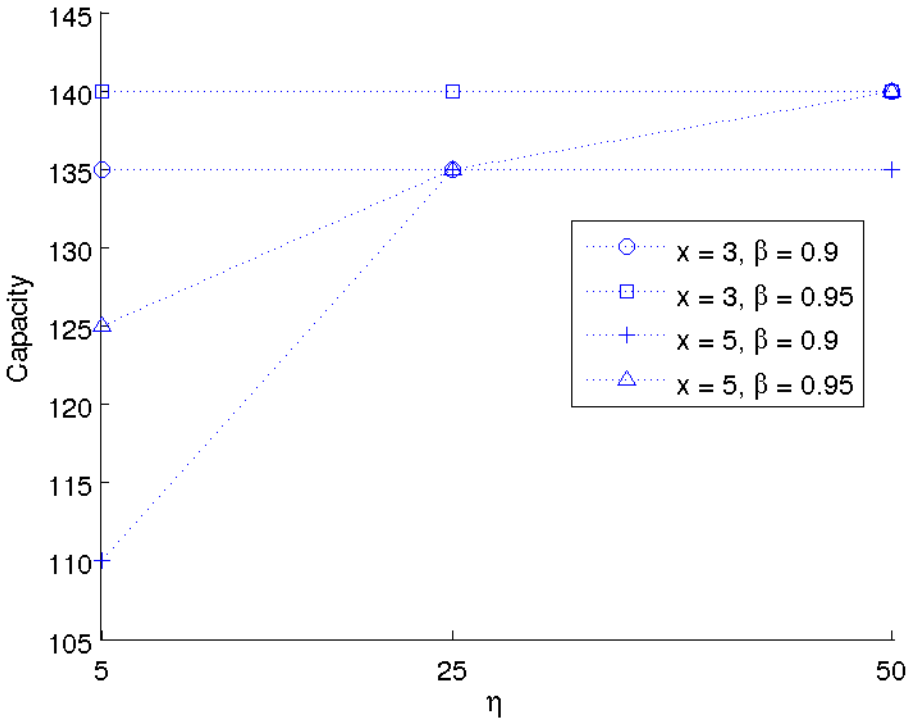

Figure 5.20: Total capacity versus η

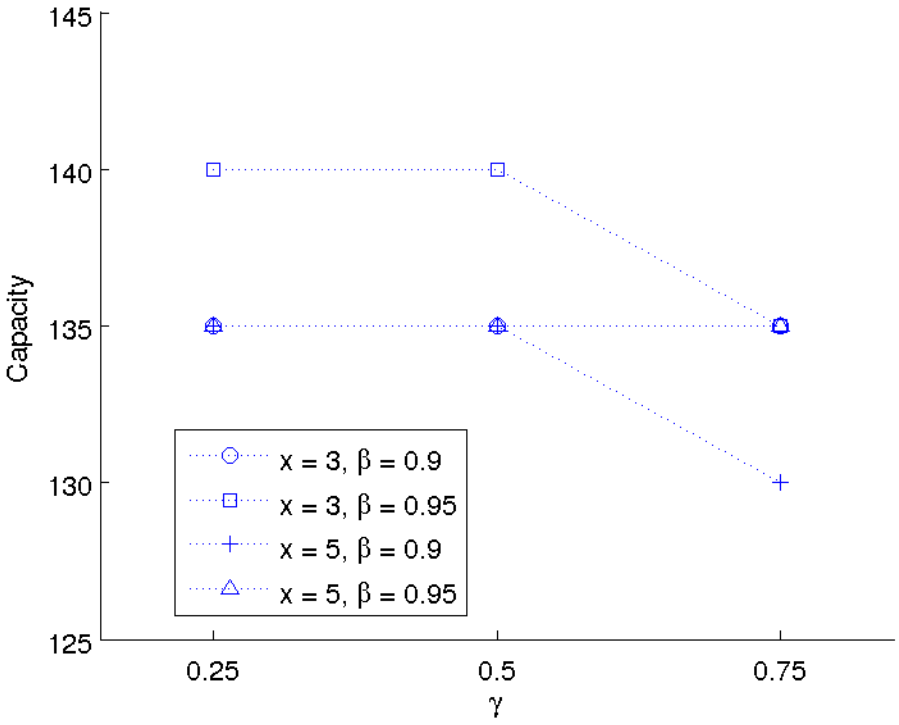

Figure 5.21: Total capacity versus γ

5.7 Conclusion

In this chapter, we considered the issue of resource provisioning for a two-tier transactional web application such that a performance target of the form Pr[response time $\leq x$] $\geq \beta$ is met over a service interval of one hour. A methodology based on the MMPP/PH/1 (FCFS) model was developed to determine the required capacity. Although the steady-state response time analysis of this model is available in the literature, we proposed an alternative approach based on a Markov chain which is more suitable for numerical computation.

Experimental results show that the performance target is met in a majority of the cases considered. For those cases where the performance target is violated, the deviation from the target probability is small and can be corrected by increasing the total capacity by 5%. Results also show that our methodology could lead to over-provisioning of computing resources; among the cases considered, the highest degree of over-provisioning is 37.5%. We feel that such level of over-provisioning is not excessive. Furthermore, over-provisioning of resources may not be undesirable because the extra capacity provides protection against deviation from the expected workload. We therefore conclude that our resource provisioning methodology is acceptable in determining the required capacity.

In addition, we have studied the relationship between the total required capacity and traffic characteristics, namely, the mean arrival rate and traffic burstiness. The burstiness indicator used is the index of dispersion for counts. We observed that the total required capacity increases with the mean arrival rate. It also increases with traffic burstiness; this implies that for a given mean arrival rate, a higher capacity level is required to meet the same performance target as the traffic becomes burstier.

Chapter 6

Resource provisioning - Two-stage tandem queue

In this chapter, we present results for our resource provisioning strategy based on a two-stage tandem queue for a service interval of one hour. The tandem queue model is shown in Figure 6.1. The arrival process is MMPP, and the two stages represent the VM's that host the web and database servers, respectively. At each stage, the service time distribution is PH and the scheduling discipline is FCFS. Compared to the single-server model, the two-stage tandem queue is a more detailed representation of the web application, yet more effort is required to obtain service time data for each of the two stages. The computation of response time results is also more costly. It can be shown that under a modest set of conditions, the capacity estimates from this model are potentially lower than those from the single-server model in order to meet the same performance target. There is therefore a trade-off between the additional cost in model parameterization and computation and lower resource estimates.

For the single-server model, we showed that for parameter values considered in Chapter 4,

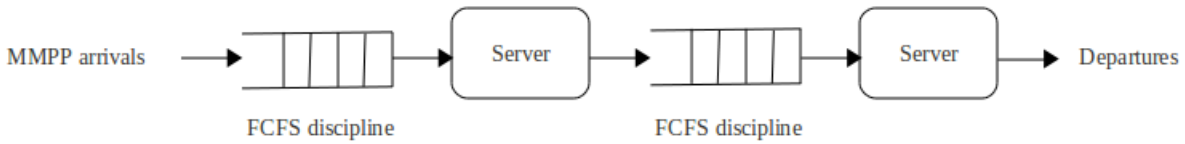

Figure 6.1: Two-stage tandem queue

the steady-state response time CDF is a good approximation when service interval is longer than two minutes. We further argued in Chapter 5 that steady-state results can also be used for parameter values similar to those in Chapter 4 when service interval is one hour long. We feel that steady-state results for the two-stage tandem queue are also a good approximation for a one-hour service interval. This is because each stage, when considered separately, will reach steady state in a similar time scale as the single-server model. Since the arrival process to stage two may not become stationary until stage one is in steady state, extra time may be required for stage two to reach steady state. This extra time, however, is not expected to be sizable. The overall effect is that the minimum time period for steady-state results to be a good approximation for the two-stage tandem queue may be longer than that for the single-server model, yet it is almost certainly less than one hour. Therefore, steady-state results are used in our investigation.

The rest of this chapter is organized as follows. In Section 6.1, we present a proof that the two-stage tandem queue results in a larger value for Pr[response time $\leq x$] than the single-server model. The steady-state response time CDF is derived in Section 6.2. In Section 6.3, a procedure to determine the service time distribution at each stage is presented. Our resource provisioning methodology is presented in Section 6.4. In Section 6.5, experimental results are used to evaluate the effectiveness of our methodology. Section 6.6 concludes the chapter.

6.1 Two-stage tandem queue versus single-server model

In this section, we present a proof that under FCFS discipline, the two-stage tandem queue results in a larger value for Pr[response time $\leq x$] than the single-server model. Let SS and TN denote the single-server model and two-stage tandem queue, respectively. Also let $\mathcal{R}_{SS}$ and $\mathcal{R}_{TN}$ be random variables denoting, respectively, the response time of SS and TN under FCFS discipline. We show that for an arbitrary job arrival sequence and service times, $\Pr[\mathcal{R}_{SS} \leq x] \leq \Pr[\mathcal{R}_{TN} \leq x]$.

Consider a time interval of length T. Assume that k jobs are processed during this time interval. Let $a_1 < a_2 < \cdots < a_k < T$ be the time instants at which jobs are submitted to both SS and TN. Also, let $\tau_1 < \tau_2 < \cdots < \tau_k$ and $t_1 < t_2 < \cdots < t_k$ be the departure times of the corresponding jobs from SS and TN, respectively. We use d_{j1} and d_{j2} to denote the service time of the j^{th} job at the first and second stages in TN. The service time of the j^{th} job at the server in SS is given by $d_{j1} + d_{j2}$.

We first use induction to show that for $1 \leq j \leq k$, $\tau_j \geq t_j$. Since the first job finds both systems empty, we have, for $j = 1$, $\tau_1 = t_1 = a_1 + d_{11} + d_{12}$, and the relationship $\tau_1 \geq t_1$ is met. Assume that $\tau_{j-1} \geq t_{j-1}$ for $j \geq 2$. To obtain a relationship between τ_j and t_j, we consider the following three scenarios depending on the value of a_j:

1. $t_{j-1} \leq \tau_{j-1} \leq a_j$: The j^{th} job finds both SS and TN empty. Therefore, $\tau_j = t_j = a_j + d_{j1} + d_{j2}$.

2. $t_{j-1} \leq a_j \leq \tau_{j-1}$: The j^{th} job finds SS non-empty and TN empty. We thus have $t_j = a_j + d_{j1} + d_{j2} \leq \tau_{j-1} + d_{j1} + d_{j2} = \tau_j$.

3. $a_j \leq t_{j-1} \leq \tau_{j-1}$: The j^{th} job finds both SS and TN non-empty. For TN, a possible scenario is that the j^{th} job is processed at the first stage while the $(j-1)^{st}$ job is processed at the second stage. We therefore have $t_j \leq t_{j-1} + d_{j1} + d_{j2} \leq \tau_{j-1} + d_{j1} + d_{j2} = \tau_j$.

For all three cases, $\tau_j \geq t_j$. We have therefore shown by induction that $\tau_j \geq t_j$ for $1 \leq j \leq k$.

We next show that $\Pr[\mathcal{R}_{SS} \leq x] \leq \Pr[\mathcal{R}_{TN} \leq x]$. The response times of the j^{th} job in SS and TN are given by $\tau_j - a_j$ and $t_j - a_j$, respectively. Since $\tau_j \geq t_j$, we have $\tau_j - a_j \geq t_j - a_j$ for any job j. This implies that for a given response time threshold x, $\tau_j - a_j \leq x$ is a sufficient condition for $t_j - a_j \leq x$ (the converse does not necessarily hold). Define

$$I_x(y) = \begin{cases} 1 & y \leq x \\ 0 & y > x \end{cases}$$

We can write $I_x(\tau_j - a_j) = 1 \implies I_x(t_j - a_j) = 1$. It follows that

$$\sum_j I_x(\tau_j - a_j) \leq \sum_j I_x(t_j - a_j)$$

We thus have

$$\begin{aligned} \Pr[\mathcal{R}_{SS} \leq x] &= \tfrac{1}{k} \textstyle\sum_{j=1}^{k} I_x(\tau_j - a_j) \\ &\leq \tfrac{1}{k} \textstyle\sum_{j=1}^{k} I_x(t_j - a_j) \\ &= \Pr[\mathcal{R}_{TN} \leq x] \end{aligned}$$

which completes the proof.

6.2 Steady-state analysis

In this section, we derive the steady-state response time CDF of the two-stage tandem queue. The results are used in our resource provisioning methodology in Section 6.4. Steady-state response time analysis of the two-stage tandem queue is available in [52, 96]. Again, we provide an alternative derivation based on Markov chain analysis which is more suitable for numerical computation. This derivation is an extension of our analysis for the single-server model presented

in Chapter 5. The MMPP parameters are denoted by λ_j and q_{ij}. The PH distribution parameters at stage m ($m = 1, 2$) are denoted by $\zeta_{m,k}$ and $s_{m,kl}$. We first define the state of the Markov chain and derive the state probabilities in Subsection 6.2.1, and then use the results to obtain the response time distribution in Subsection 6.2.2.

6.2.1 State probability

The two-stage tandem queue under consideration can be viewed as a Markov chain $\mathcal{M}_T$ with state given by (i, n_1, k_1, n_2, k_2) where i is the MMPP state; $n_m \geq 0$ is the number of jobs in stage m, and k_m is the state of the PH distribution for the job receiving service at stage m, $m = 1, 2$ ($n_m = 0$ implies $k_m = 0$). For this Markov chain, the events that cause a transition out of state (i, n_1, k_1, n_2, k_2), and the corresponding transition rate and next state, are as follows:

1. A change in MMPP state from i to j - The transition rate is q_{ij}, and the next state is (j, n_1, k_1, n_2, k_2).

2. An arrival to stage one - There are two cases. First, stage one is empty at the instant of arrival. For this case, the current state is $(i, 0, 0, n_2, k_2)$. An arrival occurs at rate λ_i and the arriving job starts service at service state l_1 with probability ζ_{1,l_1}, resulting in a transition rate equal to $\lambda_i\, \zeta_{1,l_1}$. The next state is $(i, 1, l_1, n_2, k_2)$. Secondly, stage one is non-empty at the instant of arrival. The corresponding transition rate is λ_i, and the next state is $(i, n_1 + 1, k_1, n_2, k_2)$.

3. A change in the service state at stage one from a transient state k_1 to a transient state l_1 - The transition rate is $s_{1,k_1 l_1}$ and the next state is (i, n_1, l_1, n_2, k_2).

4. A departure from stage one - There are four cases. First, there is only one job in stage one and stage two is empty, i.e., the current state is $(i, 1, k_1, 0, 0)$. The departure at stage one

occurs at rate $s_{1,k_1 0}$, and upon arrival to stage two, the job starts service in service state l_2 with probability ζ_{2,l_2} resulting in a transition rate equal to $s_{1,k_1 0}\,\zeta_{2,l_2}$. The next state is $(i, 0, 0, 1, l_2)$. Secondly, there is only one job in stage one, and stage two is non-empty. In this case, the current state is $(i, 1, k_1, n_2, k_2)$. The transition rate is $s_{1,k_1 0}$, and the next state is $(i, 0, 0, n_2 + 1, k_2)$. Third, there are more than one jobs in stage one, and stage two is empty. In this case, the departure at stage one occurs at rate $s_{1,k_1 0}$; the departing job arrives at stage two immediately and starts service in service state l_2 with probability ζ_{2,l_2}; and the next pending job at stage one starts service in service state l_1 with probability ζ_{1,l_1}. The transition rate is $s_{1,k_1 0}\,\zeta_{1,l_1}\,\zeta_{2,l_2}$, and the next state is $(i, n_1 - 1, l_1, 1, l_2)$. Finally, there are more than one jobs in both stages. In this case, the departure at stage one occurs at rate $s_{1,k_1 0}$, and the next pending job at stage one starts service in service state l_1 with probability ζ_{1,l_1}, resulting in a transition rate of $s_{1,k_1 0}\,\zeta_{1,l_1}$. The next state is $(i, n_1 - 1, l_2, n_2 + 1, k_2)$.

5. A change in the service state at stage two from a transient state k_2 to another transient state l_2 - The transition rate is $s_{2,k_2 l_2}$ and the next state is (i, n_1, k_1, n_2, l_2).

6. A departure from stage two - There are two cases. First, there is only one job at stage two. The departure at stage two occurs at rate $s_{2,k_2 0}$, and the next state is $(i, n_1, k_1, 0, 0)$. Secondly, there are two or more jobs at stage two. The departure occurs at rate $s_{2,k_2 0}$, and the next pending job starts service in service state l_2 with probability ζ_{2,l_2}, resulting in a transition rate equal to $s_{2,k_2 0}\,\zeta_{2,l_2}$. The next state is $(i, n_1, k_1, n_2 - 1, l_2)$.

A summary of the state transitions is given in table 6.1.

Let $P(i, n_1, k_1, n_2, k_2)$ be the steady-state probability of being in state (i, n_1, k_1, n_2, k_2). The probabilities $P(i, n_1, k_1, n_2, k_2)$ can be obtained by solving the balance equations for Markov chain $\mathcal{M}_T$ with transitions shown in Table 6.1.

Table 6.1: State transitions for $\mathcal{M}_T$

Event	Condition	Current state	Next state	Rate
MMPP state change		(i, n_1, k_1, n_2, k_2)	(j, n_1, k_1, n_2, k_2)	q_{ij}
Arrival	$n_1, k_1 = 0$	$(i, 0, 0, n_2, k_2)$	$(i, 1, l_1, n_2, k_2)$	$\lambda_i\ \zeta_{1,l_1}$
	$n_1, k_1 \geq 1$	(i, n_1, k_1, n_2, k_2)	$(i, n_1 + 1, k_1, n_2, k_2)$	λ_i
Service state change at stage one	$n_1, k_1 \geq 1$	(i, n_1, k_1, n_2, k_2)	(i, n_1, l_1, n_2, k_2)	$s_{1,k_1 l_1}$
Departure from stage one	$n_1 = 1,\ n_2, k_2 = 0$	$(i, 1, k_1, 0, 0)$	$(i, 0, 0, 1, l_2)$	$s_{1,k_1 0}\ \zeta_{2,l_2}$
	$n_1 = 1,\ n_2, k_2 \geq 1$	$(i, 1, k_1, n_2, k_2)$	$(i, 0, 0, n_2 + 1, k_2)$	$s_{1,k_1 0}$
	$n_1 \geq 2,\ n_2, k_2 = 0$	$(i, n_1, k_1, 0, 0)$	$(i, n_1 - 1, l_1, 1, l_2)$	$s_{1,k_1 0}\ \zeta_{1,l_1}\ \zeta_{2,l_2}$
	$n_1 \geq 2,\ n_2, k_2 \geq 1$	(i, n_1, k_1, n_2, k_2)	$(i, n_1 - 1, l_1, n_2 + 1, k_2)$	$s_{1,k_1 0}\ \zeta_{1,l_1}$
Service state change at stage two	$n_2, k_2 \geq 1$	(i, n_1, k_1, n_2, k_2)	(i, n_1, k_1, n_2, l_2)	$s_{2,k_2 l_2}$
Departure from stage two	$n_2 = 1$	$(i, n_1, k_1, 1, k_2)$	$(i, n_1, k_1, 0, 0)$	$s_{2,k_2 0}$
	$n_2 \geq 2$	(i, n_1, k_1, n_2, k_2)	$(i, n_1, k_1, n_2 - 1, l_2)$	$s_{2,k_2 0}\ \zeta_{2,l_2}$

6.2.2 Response time distribution

We now derive the response time CDF under FCFS discipline using the state probabilities $P(i, n_1, k_1, n_2, k_2)$ obtained in Subsection 6.2.1. Consider a "tagged" job arriving to the system. Under FCFS scheduling, jobs arriving after the tagged job will have no impact on the response time. These jobs are excluded from the analysis by cutting off the arrival process. From this point on, the behavior of the model can be characterized by a Markov chain $\mathcal{M}'_T$ with state description (n_1, k_1, n_2, k_2); $n_m \geq 0$ is the number of jobs in stage m, and k_m is the state of the PH distribution for the job receiving service at stage m, $m = 1, 2$ ($n_m = 0$ implies $k_m = 0$). The transition rates of $\mathcal{M}'_T$ are the same as those for events 3-6 discussed in Subsection 6.2.1, i.e., events that are not affected by the arrival process. A summary of state transitions, and the corresponding transition rates and next states, are shown in Table 6.2.

The Markov chain $\mathcal{M}'_T$ has the following properties: (i) It is an absorbing Markov chain where state $(0, 0, 0, 0)$ is the absorbing state, and every other state is transient. (ii) The absorption time from a transient state to the absorbing state follows a PH distribution; the transition rate matrix of this PH distribution is obtained from the transition rate matrix of $\mathcal{M}'_T$ by eliminating the row and column corresponding to the state $(0, 0, 0, 0)$.

For the tagged job, the probability that the Markov chain $\mathcal{M}_T$ is in state (i, n_1, k_1, n_2, k_2) when it arrives is given by

$$A(i, n_1, k_1, n_2, k_2) = \frac{\lambda_i}{\lambda} P(i, n_1, k_1, n_2, k_2)$$

where λ is the mean arrival rate. The state of the Markov chain $\mathcal{M}'_T$ immediately after the arrival is

$$(n_1 + 1, k_1, n_2, k_2) \text{ when } n_1 \geq 1$$

Table 6.2: State transitions for $\mathcal{M}'_T$

Event	Condition	Current state	Next state	Rate
Service state change at stage one	$n_1, k_1 \geq 1$	(n_1, k_1, n_2, k_2)	(n_1, l_1, n_2, k_2)	$s_{1,k_1 l_1}$
Departure from stage one	$n_1 = 1,\ n_2, k_2 = 0$	$(1, k_1, 0, 0)$	$(0, 0, 1, l_2)$	$s_{1,k_1 0}\ \zeta_{2,l_2}$
	$n_1 = 1,\ n_2, k_2 \geq 1$	$(1, k_1, n_2, k_2)$	$(0, 0, n_2 + 1, k_2)$	$s_{1,k_1 0}$
	$n_1 \geq 2,\ n_2, k_2 = 0$	$(n_1, k_1, 0, 0)$	$(n_1 - 1, l_1, 1, l_2)$	$s_{1,k_1 0}\ \zeta_{1,l_1}\ \zeta_{2,l_2}$
	$n_1 \geq 2,\ n_2, k_2 \geq 1$	(n_1, k_1, n_2, k_2)	$(n_1 - 1, l_1, n_2 + 1, k_2)$	$s_{1,k_1 0}\ \zeta_{1,l_1}$
Service state change at stage two	$n_2, k_2 \geq 1$	(n_1, k_1, n_2, k_2)	(n_1, k_1, n_2, l_2)	$s_{2,k_2 l_2}$
Departure from stage two	$n_2 = 1$	$(n_1, k_1, 1, k_2)$	$(n_1, k_1, 0, 0)$	$s_{2,k_2 0}$
	$n_2 \geq 2$	(n_1, k_1, n_2, k_2)	$(n_1, k_1, n_2 - 1, l_2)$	$s_{2,k_2 0}\ \zeta_{2,l_2}$

and

$$(1, l_1, n_2, k_2) \text{ with probability } \zeta_{1,l_1} \text{ when } n_1 = 0$$

The response time of the tagged job is given by the time to absorption of $\mathcal{M}'_T$ from the state immediately after the arrival. Let $r(x)$ be the response time CDF and $G_{n_1,k_1,n_2,k_2}(x)$ be the CDF of the time to absorption of $\mathcal{M}'_T$ from state (n_1, k_1, n_2, k_2). Using the law of total probability, we have

$$\begin{aligned} r(x) = &\textstyle\sum_i \sum_l \sum_{n_2} \sum_{k_2} A(i, 0, 0, n_2, k_2)\ \ \zeta_{1,l_1}\ \ G_{1,l_1,n_2,k_2}(x) \\ &+ \textstyle\sum_i \sum_{n_1 \geq 1} \sum_{k_1} \sum_{n_2} \sum_{k_2} A(i, n_1, k_1, n_2, k_2)\ \ G_{n_1+1,k_1,n_2,k_2}(x) \end{aligned} \tag{6.1}$$

From Equation 6.1, one can observe that $r(x)$ is a linear combination of PH distributions $G_{n_1,k_1,n_2,k_2}(x)$, and hence is of PH itself.

Numerical computation of $P(i, n_1, k_1, n_2, k_2)$ and $G_{n_1,k_1,n_2,k_2}(x)$ can be done using standard MATLAB libraries. One also needs to set an upper bound on $n_1 + n_2$ so that the number of states (i, n_1, k_1, n_2, k_2) remains finite. The upper bound N_u is selected in such a way that the probability of having $n_1 + n_2 > N_u$ jobs in system is negligible.

6.3 Service time

Similar to the case of the single-server model, the service time distribution at each stage of the two-stage tandem queue is obtained by system measurement. A web application is run in single-user mode. When a job is processed, the web server may make several calls to the database server. Since there is no queueing, the duration of a database call can be obtained by placing timestamps immediately before and after the code segment on the web server that makes the call, and computing the difference of the two timestamps. Let N_b be the number of database calls made by a job, and $b_d(j)$ be the duration of the j^{th} database call ($1 \leq j \leq N_b$). We

approximate the total service time of a job at the database server (denoted by b_d) by the sum of the duration of individual database calls, i.e., $b_d = \sum_j b_d(j)$. As to the web server, the black-box approach described in Chapter 5 is used to obtain the aggregate service time at the web and database servers (denoted by b). The total service time of a job at the web server is then given by $b_w = b - b_d$.

The service times b_w and b_d are affected by the capacities of the web and database servers, denoted by C_w and C_d, respectively. The following procedure is used to determine the service time distribution at each stage for a given combination of C_w and C_d:

1. VM's with capacities C_w and C_d are obtained from the infrastructure provider.

2. The web and database servers are deployed on these VM's.

3. A load generator is used to submit a total of N_J jobs to the web server. This load generator consists of a single session which generates a sequence of jobs. A job is submitted only after the response for the previous job has been received. This ensures that at any given time, there is only one job in the system, and hence there is no queueing. For each job, the total service time at the web and database servers, given by b_w and b_d, are obtained using the procedure described above. The result is two samples of service times for the web and database servers, denoted by X_w and X_d, respectively.

4. GFIT tool [77] is used to fit two PH distributions, one to X_w and the other to X_d.

6.4 Methodology

The resource provisioning methodology based on the two-stage tandem queue is similar to that based on the single-server model (see Section 5.3). For given MMPP parameters and performance

target, a search is conducted to determine the capacity estimates with minimal cost such that the performance target is met. There are two differences between the two methodologies:

1. For a capacity setting (C_w, C_d), the service time distributions at the web and database servers are determined using the procedure described in Section 6.3.

2. The response time CDF is determined using steady-state results for the two-stage tandem queue presented in Section 6.2.

6.5 Evaluation

In this section, we evaluate the effectiveness of resource provisioning based on the two-stage tandem queue. The experimental environment used in our evaluation is the same as that described in Section 5.4. Briefly, the infrastructure provider offers $N_C = 20$ levels of VM capacity. Each VM is configured with 2 GB of memory, 20 GB of storage, and a fraction of the capacity of a single CPU core; the capacity levels are $B_1 = 5\%, B_2 = 10\%, \cdots, B_{20} = 100\%$. A VM is priced proportional to its CPU level, i.e., $c(B_j) = \kappa B_j$ where κ is a constant. The application is a two-tier implementation of TPC-W. Each tier is deployed on a VM, and the two VM's are hosted on separate physical nodes. A third node is used to generate HTTP jobs in accordance with a two-state MMPP.

To evaluate the effectiveness of our resource provisioning methodology, we use a similar procedure to that described in Section 5.5. Specifically, the capacities of the web and database servers are set to those obtained from the optimization problem. An MMPP traffic generator is then used to submit jobs to the web server for one hour, and the response time of each job is measured. Let β' be the fraction of jobs meeting the response time threshold x. The difference between β' and the target probability β, denoted by $e = \beta' - \beta$, is used to evaluate the effectiveness of our

methodology.

The three workloads used in our evaluation are the same as those shown in Table 5.3; the corresponding default values of the MMPP parameters are shown in Table 5.4. The values of x and β considered are 3 and 5 seconds, and 0.9 and 0.95, respectively. When determining the service time distribution at each stage, the number of jobs submitted by the load generator is $N_J = 2000$. When computing the response time CDF, the upper bound for the number of jobs in system is set to $N_u = 50$.

As in the case of the single-server model, the optimization problem is solved using a search based on the hill climbing algorithm [59]. Each search may require the computation of response time CDF several times. We found that the computation of response time CDF for the two-stage tandem queue is significantly more costly than that for the single-server model; this is because of extra dimensions in state definition (see Section 6.2). An additional search could hence lead to a significant increase in total computation time. To save computation time, the following approach is used.

A single search is conducted using the optimal solution based on the single-server model as the initial point, and the results are evaluated using our experimental system. We feel that this initial point is a more suitable candidate than one selected at random, as it is the best solution among three local optimums for a similar optimization based on the single-server model. In addition, with this initial point, the capacity estimate from the methodology based on the two-stage tandem queue is potentially lower than that based on the single-server model. A proof for this property is presented below.

Suppose the optimal solution based on the single-server model is (B_i, B_j). It follows that for capacity settings (B_{i-1}, B_j), (B_i, B_{j-1}) and (B_{i-1}, B_{j-1}), $\Pr[\mathcal{R}_{SS} \leq x] < \beta$. In Section 6.1, we have shown that for arbitrary job arrival sequence and service times, $\Pr[\mathcal{R}_{SS} \leq x] \leq \Pr[\mathcal{R}_{TN} \leq x]$.

It is therefore plausible that for setting (B_{i-1}, B_j), (B_i, B_{j-1}) or (B_{i-1}, B_{j-1}), $\Pr[\mathcal{R}_{TN} \leq x] \geq \beta$. This would potentially lead to lower capacity estimates for the methodology based on the two-stage tandem queue.

In general, a lower capacity estimate leads to a smaller β' (or smaller e). We therefore expect to observe smaller values of e than those for the single-server model. This may lead to more cases where e is negative (or the performance target is violated). If the results for e provide sufficient evidence that the capacity estimates are inadequate, one can conclude that the methodology is not acceptable without the need for additional search. The reason is that any additional search can only lead to a lower capacity estimate (which in turn leads to a smaller e); this is the case when a better local optimum is found. This would not change the conclusion that the methodology is not acceptable.

On the other hand, if the results based on a single search indicate that the methodology is acceptable, then additional search will need to be performed.

We next present our experimental results.

6.5.1 Workload 1

In Figures 6.2 and 6.3, the mean and 95% confidence interval for e are plotted against the mean arrival rate λ for $\beta = 0.9$ and $\beta = 0.95$, respectively. The values of λ considered are 1, 1.5 and 2 jobs/second. As discussed earlier, these results are based on a single search. We observe that the value of e is negative in 6 out of 12 cases. This means that the capacity estimates obtained from our methodology are not sufficient to meet the performance target in half the number of cases considered. The worst case values of e are -0.19 and -0.274 for $\beta = 0.9$ and 0.95, respectively. These values represent significant deviations from the target probability.

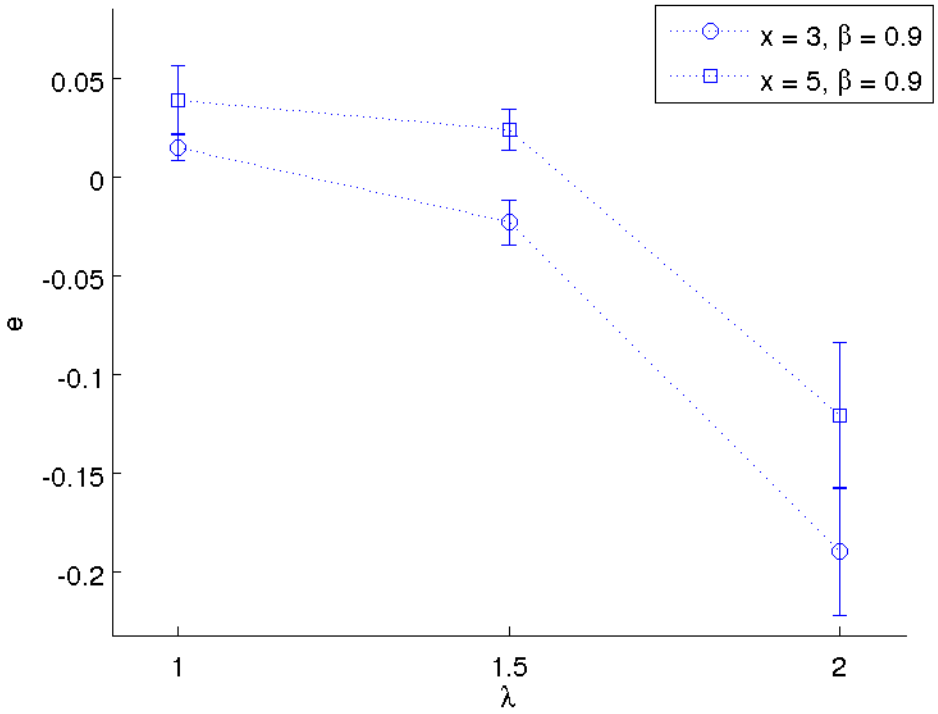

Figure 6.2: e vs. λ

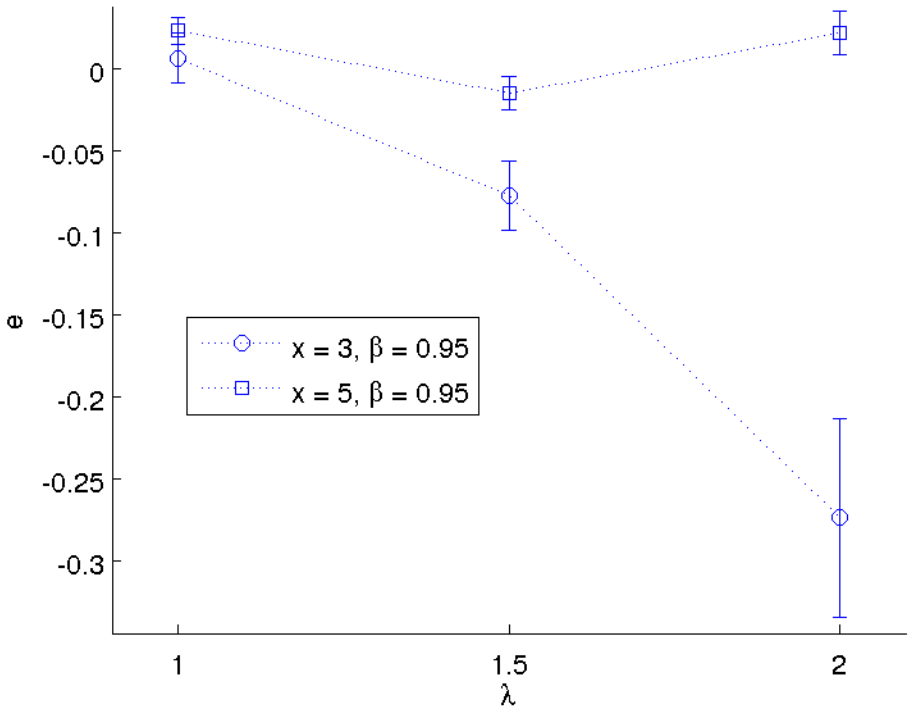

Figure 6.3: e vs. λ

A non-zero value of e indicates inaccuracy in resource estimate. As discussed in Section 5.5.1, the sources of inaccuracy that potentially lead to over-provisioning are (i) the optimization problem may return a local (instead of the global) optimum, and (ii) VM capacities are defined in increments of 5%. Another source of inaccuracy is the assumptions made in the development of the two-stage tandem queue; they include (i) the service time distributions at the web and database servers are obtained by system measurement, (ii) the complex interaction between computing resources at each stage is modeled by a server with FCFS discipline, and (iii) the flow of a job between the two servers is represented by a simple routing based on a tandem queue. These assumptions lead to inaccuracy in predicting the response time CDF.

The results in Figures 6.2 and 6.3 indicate that the inaccuracy due to model assumptions lead to under-provisioning of resources, and the level of under-provisioning may far outweigh the potential over-provisioning due to local optimum or 5% increments in VM capacity definition.

In Figures 6.4 and 6.5, e is plotted against ρ for $\beta = 0.9$ and $\beta = 0.95$, respectively. The values of ρ considered are 3, 5 and 7. Again, these results are based on a single search. We observe that in 8 out of 12 cases, the capacity estimates are not sufficient to meet the performance target. The worst case value of e is -0.115 and -0.105 for $\beta = 0.9$ and $\beta = 0.95$, respectively. Again, these results indicate significant violations of the performance target.

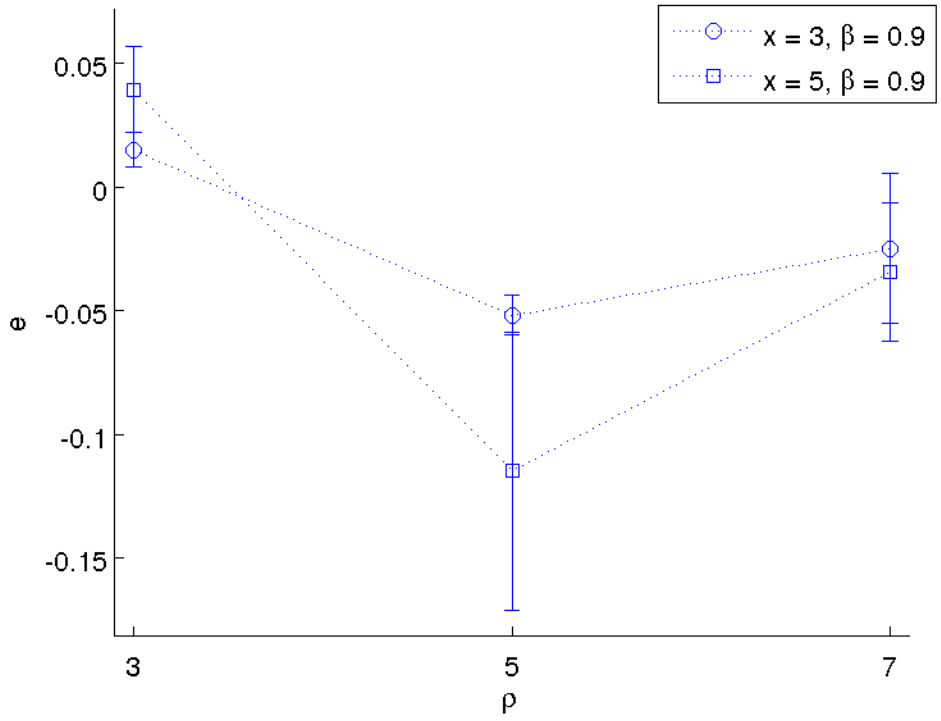

Figure 6.4: e vs. ρ

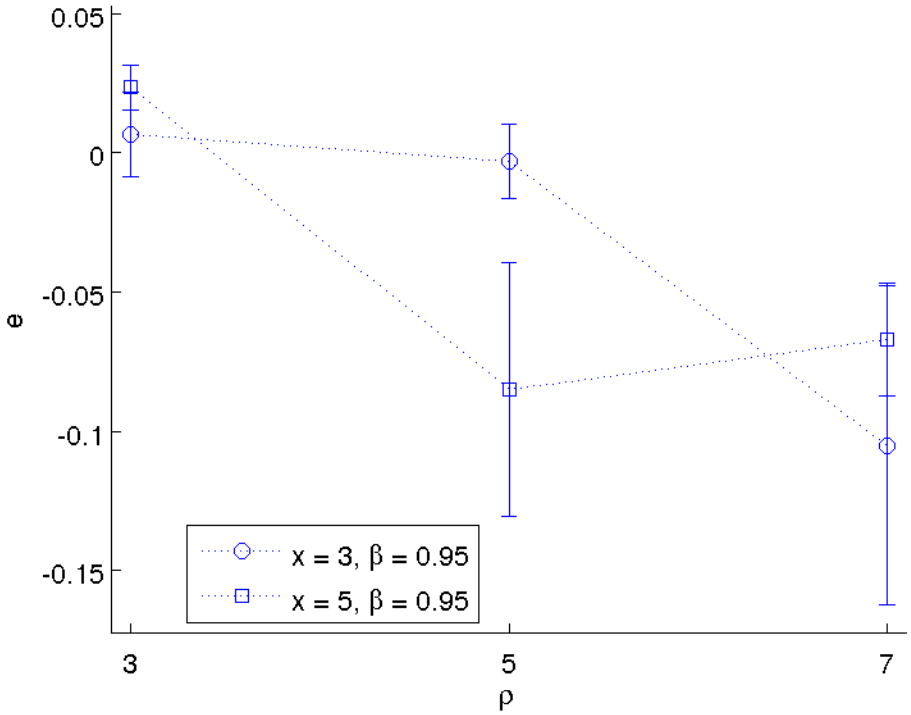

Figure 6.5: e vs. ρ

As a summary, the results in Figures 6.2 to 6.5 show negative e in 58% of the cases considered, and significant deviations from the target probability for some of these cases. We therefore conclude that for Workload 1, the methodology based on the two-stage tandem queue is not acceptable and that a single search is sufficient to draw this conclusion.

6.5.2 Workloads 2

In Figures 6.6 and 6.7, e is plotted against λ for $\beta = 0.9$ and $\beta = 0.95$, respectively. The values of λ considered are 1.5, 2 and 2.5 jobs/second. These results are also based on a single search. We again observe that in 8 out of 12 cases, the value of e is negative, or the performance target is not met. The worst case value of e is -0.21 and -0.15 for $\beta = 0.9$ and $\beta = 0.95$, respectively.

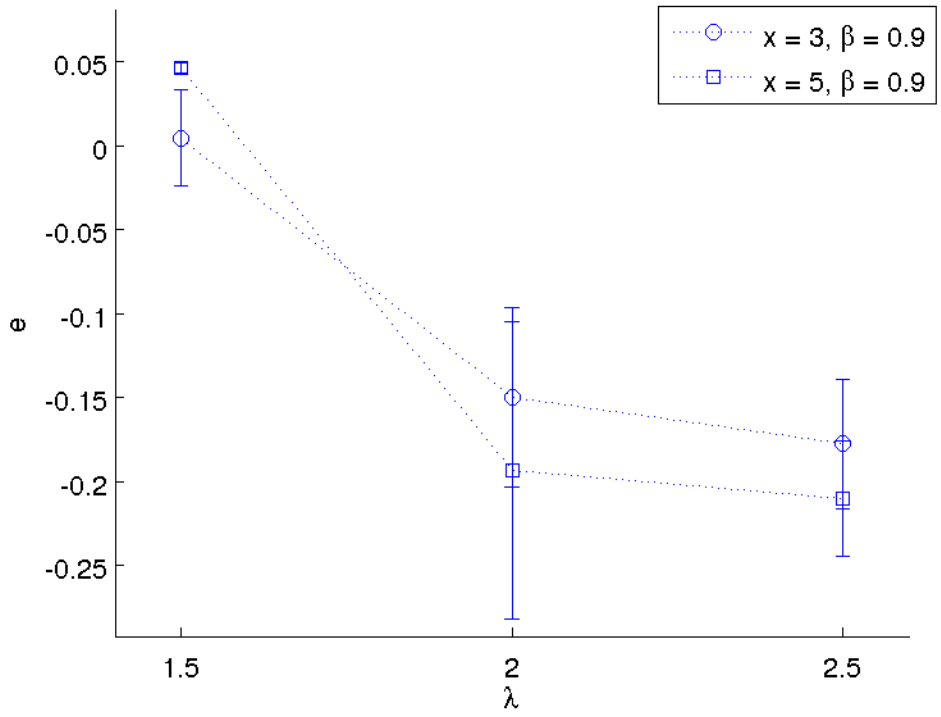

Figure 6.6: e vs. λ

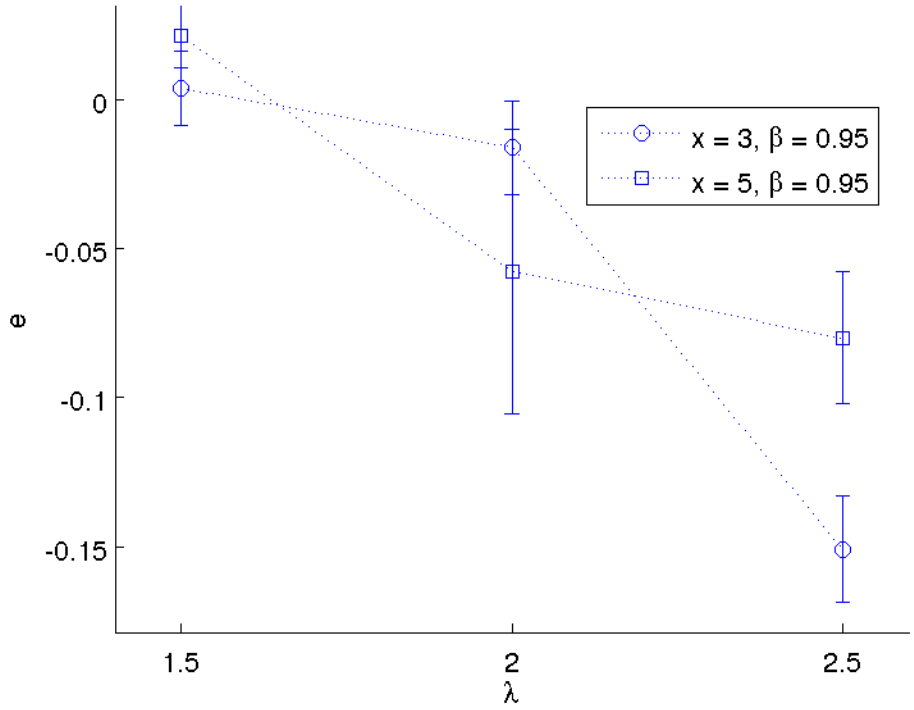

Figure 6.7: e vs. λ

Again, the results for Workloads 1 and 2 show that the value of e is negative in 60% of the cases considered, and that significant deviations from the target probability are observed for some of the cases. These results provide strong evidence that the methodology based on the two-stage tandem queue is not effective for resource provisioning purposes. Experiments with Workload 3 are not expected to affect this conclusion, and hence are not performed.

6.6 Conclusion

In this chapter, we evaluated the effectiveness of a methodology based on a two-stage tandem queue for resource provisioning. Analytic results for the steady-state response time CDF of the two-stage tandem queue are presented. This CDF is used in an optimization problem to determine the required capacities to meet a performance target with minimal cost.

We have also shown that theoretically, the methodology based on the two-stage tandem queue yields a lower capacity level compared to that based on the single-server model. However, our evaluation results indicate that the assumptions made in development of this model lead to the violation of performance target in 60% of the cases considered and significant deviations from the target probability for some of these cases. We therefore conclude that our methodology based on the two-stage tandem queue is not acceptable for resource provisioning purposes.

In Chapter 5, we have shown that capacity estimates from a search based on the single-server model are sufficient to meet the performance target in a majority of cases considered, and violations of the performance target, if any, are small and can be corrected with a 5% increase in capacity (i.e., the smallest capacity increment defined by the target provider). In view of the importance of meeting the performance target from subscriber's perspective, we conclude that the methodology based on the single-server model is the preferred methodology.

Chapter 7

Resource provisioning for small service interval

In this chapter, we consider a scenario where an infrastructure provider defines a smaller TU than one hour for pricing purposes. As mentioned in Chapter 1, it is conceivable that such pricing policy will be in effect in the future, as it allows the subscriber to make more frequent adjustments to resource level in response to changing workload; this has the potential to reduce the resource provisioning cost.

In Chapter 5, we evaluated the effectiveness of our methodology based on the MMPP/PH/1 (FCFS) model in terms of meeting a performance target of the form Pr[response time $\leq x$] $\geq \beta$ over a service interval (or TU) of one hour. To facilitate the comparison against results in Chapter 5, we aim, in this chapter, to meet the same performance target over a time interval of one hour; this interval is referred to as the *operation interval*. With a smaller TU than one hour, an operation interval consists of consecutive service intervals each having length one TU as shown in Figure 7.1. For convenience, we assume that one hour is equal to an integer number of

TU's.

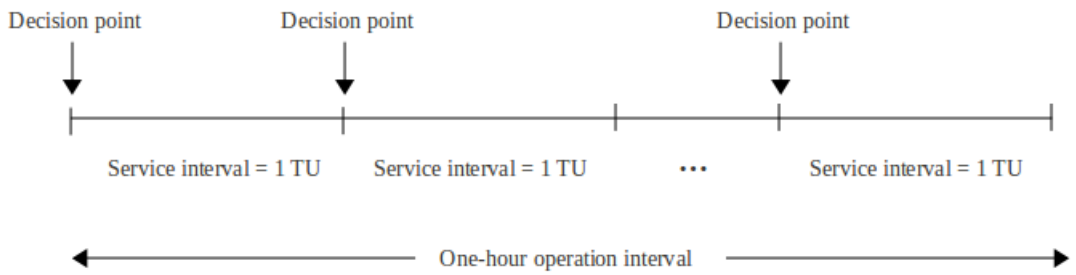

Figure 7.1: One-hour operation interval organized into service intervals of length one TU

Since the arrival process over the operation interval is MMPP, a straightforward approach is to use the steady-state results for the MMPP/PH/1 (FCFS) model discussed in Chapter 5 to determine the capacity estimate, and request this capacity for every service interval. The resulting capacity level for the entire operation interval is the same regardless of the size of TU. We refer to this approach as *static provisioning*. The capacity level for static provisioning is the same as that determined in Chapter 5 for a one-hour TU. A smaller TU, hence, does not provide any advantage in terms of reduction in capacity requirement.

In this chapter, we consider the case where TU is small in the sense that it is comparable to the average sojourn time in an MMPP state. With such a TU and MMPP arrivals, the job arrival rates at different service intervals can be quite different. Our approach is to estimate, at the beginning of each service interval (also known as a decision point), the arrival rate over that service interval and determine the required capacity based on this estimate. The subscriber can then acquire more resources or release unnecessary resources accordingly. Since the capacities can be different across service intervals, there is a potential to reduce the long-term cost of resource provisioning.

Based on the conclusions in Chapter 6, we use a single-server model in our methodology to

determine the resource estimate. For this model, a natural choice for the arrival process is Poisson because TU is comparable to the average sojourn time in an MMPP state and each MMPP state represents a Poisson process. As to service time, we assume that it follows a PH distribution, and any capacity adjustments at a decision point take effect immediately. We thus have an M/PH/1 (FCFS) model. We recognize that for small service intervals, time-dependent results for the response time CDF may be required. However, the computation of time-dependent results is prohibitively expensive. We therefore use steady-state results as an approximation.

At each decision point, the resource requirement to meet the performance target over the upcoming service interval is determined using an optimization problem based on the M/PH/1 (FCFS) model. Our methodology requires an estimate of job arrival rate for each service interval. Three heuristic algorithms for estimating this arrival rate are investigated. The effectiveness of our methodology, as well as the merit of the three algorithms, are evaluated by checking whether the performance target is met for the one-hour operation interval and the average capacity over this operation interval. The average capacity is also compared against that obtained by static provisioning to examine whether or not there are savings in cost.

The rest of this chapter is organized as follows. In Section 7.1, we present analytic results on steady-state response time CDF of the M/PH/1 (FCFS) model. In Section 7.2, our methodology based on the M/PH/1 (FCFS) model is described. Three heuristic algorithms to obtain an estimate of the arrival rate for the upcoming service interval are presented in Section 7.3. The first two algorithms take advantage of advance knowledge that the traffic is MMPP. The third algorithm presumes no advance knowledge regarding the nature of the arrival process. Evaluation results on the effectiveness of these algorithms are presented in Section 7.4. Finally, Section 7.5 concludes the chapter.

7.1 Analysis of the M/PH/1 (FCFS) model

The M/PH/1 (FCFS) model is a special case of the MMPP/PH/1 (FCFS) model where the MMPP has only one state. The derivation of the steady-state response time CDF is thus similar to that in Section 5.1. An overview of the derivation is shown below.

1. The Poisson process is fully characterized by the mean arrival rate denoted by λ.

2. The state of the Markov chain $\mathcal{M}_S$ is characterized by (n, k) where n is the number of jobs in system, and k is the state of the PH distribution for the job receiving service ($n = 0$ implies $k = 0$). The transitions from state (n, k), as well as the corresponding next states and transition rates, are shown in Table 7.1. The steady-state probability of being in state (n, k), denoted by $P(n, k)$, can be obtained by solving the balance equations for the Markov chain $\mathcal{M}_S$ with transitions shown in Table 7.1.

Table 7.1: State transitions for $\mathcal{M}_S$

Event	Condition	Current state	Next state	Rate
Arrival	$n, k = 0$	$(0, 0)$	$(1, l)$	$\lambda\, \zeta_l$
	$n, k \geq 1$	(n, k)	$(n + 1, k)$	λ
Service state change	$n, k \geq 1$	(n, k)	(n, l)	s_{kl}
Service completion	$n = 1$	$(1, k)$	$(0, 0)$	s_{k0}
	$n \geq 2$	(n, k)	$(n - 1, l)$	$s_{k0}\, \zeta_l$

3. Since the arrival process is Poisson, the steady-state probability that an arrival finds the system in state (n, k), denoted by $A(n, k)$, is the same as $P(n, k)$ [62].

4. The absorbing Markov chain $\mathcal{M}'_S$ is obtained from $\mathcal{M}_S$ by eliminating the transitions caused by the arrival process; it is also characterized by state (n, k) where n and k have the same interpretation as in state definition of $\mathcal{M}_S$. The state transitions for $\mathcal{M}'_S$ are shown in

Table 7.2.

Table 7.2: State transitions for $\mathcal{M}'_S$

Event	Condition	Current state	Next state	Rate
Service state change	$n, k \geq 1$	(n, k)	(n, l)	s_{kl}
Service completion	$n = 1$	$(1, k)$	$(0, 0)$	s_{k0}
	$n \geq 2$	(n, k)	$(n-1, l)$	$s_{k0}\ \zeta_l$

5. Let $G_{n,k}(x)$ be the CDF of the time to absorption of $\mathcal{M}'_S$ from state (n, k); $G_{n,k}(x)$ is of PH. The transition rate matrix of this PH distribution is obtained from the transition rate matrix of $\mathcal{M}'_S$ by eliminating the row and column corresponding to the state $(0, 0)$. The state of the Markov chain $\mathcal{M}'_S$ immediately after the arrival is $(n + 1, k)$ when $n \geq 1$ and $(1, l)$ with probability ζ_l when $n = 0$. As before, we denote by $r(x)$ the steady-state response time CDF. Using the law of total probability, we obtain the following expression for $r(x)$:

$$r(x) = \sum_l A(0,0)\ \zeta_l\ G_{1,l}(x) + \sum_{n\geq 1} \sum_k A(n,k)\ G_{n+1,k}(x)$$

Numerical computation of state probabilities $P(n, k)$ and PH distributions $G_{n,k}(x)$ can be done using standard MATLAB libraries.

7.2 Methodology

In this section, we present our resource provisioning methodology for a service interval of length one TU. It consists of two steps:

1. Obtain an estimate of the job arrival rate over the service interval.

2. Solve the following optimization problem to determine the required web and database server capacities (denoted by C_w and C_d, respectively) such that the cost is minimized and Pr[response time $\leq x$] $\geq \beta$ for all the jobs processed over the service interval.

$$\text{minimize} \quad c(C_w) + c(C_d)$$

$$s.t. \quad \Pr[\text{response time} \leq x] \geq \beta$$
$$C_w, C_d \in \mathbb{B}$$

For Step 1, algorithms to determine the arrival rate will be presented in Sections 7.3. As to Step 2, the solution method for the optimization problem is the same as that based on the MMPP/PH/1 (FCFS) model described in Section 5.3, except that the response time CDF (for the arrival rate obtained in Step 1) is determined using the steady-state results of the M/PH/1 (FCFS) model presented in Section 7.2. Note that this solution method involves fitting of a PH distribution to service time data obtained using a black-box approach, and solving the optimization problem by a hill-climbing search method [59].

In our methodology, the capacity estimate (C_w and C_d) is a function of the arrival rate λ. λ can take on any value over a continuous interval. For convenience, we consider a scenario where the arrival rate is taken from a finite set $\mathbb{A}$ that is known in advance. As an example, for an M-state MMPP, $\mathbb{A} = \{\lambda_1, \cdots, \lambda_M\}$ where λ_j is the arrival rate in MMPP state j. Since resource provisioning is done over an operation interval consisting of multiple service intervals, the estimated arrival rate may be the same for two or more service intervals. To avoid repeatedly solving the optimization problem for the same arrival rate, we determine the capacity estimate for each arrival rate in $\mathbb{A}$ offline and store the results in a rate-to-capacity table (RCT). RCT is indexed by arrival rate and the corresponding entry is the capacity estimate determined using our

methodology. With this scheme, the computation overhead for solving the optimization problem at a decision point is avoided. This is particularly beneficial if the computation time is not insignificant compared to the service interval. Our resource provisioning methodology using the RCT is shown in Figure 7.2.

At a decision point:

1. Obtain an estimate of the job arrival rate for the upcoming service interval.
2. Retrieve the capacity estimate corresponding to this arrival rate from the RCT.

Figure 7.2: Resource provisioning methodology for a service interval

Since the arrival rate estimate and the resulting capacity level may be different from one service interval to another, the methodology in this chapter is referred to as *dynamic provisioning*.

7.3 Algorithms

In this section, we present three heuristic algorithms to estimate the arrival rate for the upcoming service interval, as indicated in Step 1 of our methodology in Figure 7.2. The first two algorithms take advantage of the knowledge that the traffic is an M-state MMPP with arrival rate in state j given by λ_j, while the third algorithm presumes no such knowledge. For simplicity, all three algorithms obtain an estimate of arrival rate for the upcoming service interval using only the statistics collected over the most recent service interval.

7.3.1 Algorithm 1

Algorithm 1 uses a probabilistic method to obtain an estimate of the arrival rate. Consider two consecutive service intervals k and l as shown in Figure 7.3. At the decision point, Algorithm 1 determines the most likely MMPP state using as input the number of arrivals over service interval k. The arrival rate corresponding to this MMPP state is taken as the estimate of the arrival rate for service interval l.

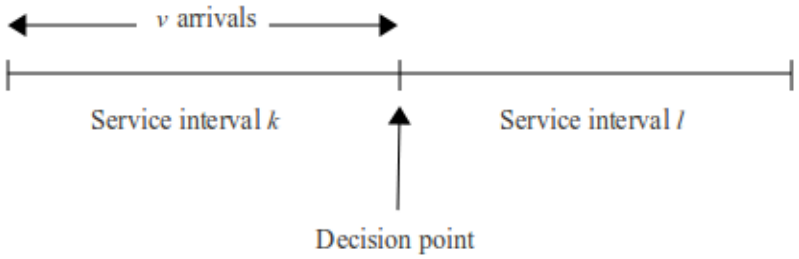

Figure 7.3: Algorithm 1

Let Φ be a random variable denoting the estimated MMPP state at the decision point. Also, let V be a random variable denoting the number of arrivals over service interval k. Algorithm 1 is based on the values of the conditional probability $\Pr[\Phi = j \mid V = v],\ j = 1, \cdots, M,\ v = 0, 1, \cdots$. An analytic expression for $\Pr[\Phi = j \mid V = v]$ can be obtained using Bayes' theorem, i.e.,

$$\Pr[\Phi = j \mid V = v] = \frac{\Pr[V = v \mid \Phi = j]\ \Pr[\Phi = j]}{\Pr[V = v]} \tag{7.1}$$

In this equation, $\Pr[\Phi = j]$ is the long-term fraction of time that the MMPP is in state j; it is obtained by computing the stationary distribution of the MMPP transition rate matrix $\mathbf{Q}$. An approximate expression for $\Pr[V = v \mid \Phi = j]$ is obtained by assuming that the MMPP is in state

j (or the arrival process is Poisson with rate λ_j) over service interval k. We thus have

$$\Pr[V = v \mid \Phi = j] = \frac{(\lambda_j t)^v}{v!} e^{-\lambda_j t} \tag{7.2}$$

where t is the length of service interval (equal to one TU). As to $\Pr[V = v]$, applying the law of total probability yields

$$\Pr[V = v] = \sum_j \Pr[V = v \mid \Phi = j] \Pr[\Phi = j] \tag{7.3}$$

Algorithm 1 first determines v, the value of the random variable V, by measuring the number of arrivals in service interval k. It then computes $\Pr[\Phi = j \mid V = v]$, $j = 1, \cdots, M$, using Equations 7.1, 7.2 and 7.3. The state m that maximizes $\Pr[\Phi = j \mid V = v]$, i.e.,

$$m = \text{argmax}_j \Pr[\Phi = j \mid V = v]$$

is taken as the estimate of the MMPP state for service interval l. The corresponding arrival rate estimate is given by λ_m.

Algorithm 1 requires the computation of m at each decision point. Since the MMPP parameters and the length of service interval are known, the values of m can be precomputed in an offline manner for $v = 0, 1, \cdots$, and the corresponding arrival rates are stored in a number-to-rate table (NRT). NRT is indexed by v and the corresponding entry is the rate (or λ_m) for this value of v. With this approach, the estimated arrival rate at a decision point can be obtained by a simple table look-up operation.

The rationale behind the design of Algorithm 1 is as follows. If v is large, there is a chance that a backlog is accumulated over service interval k. Algorithm 1 reacts by estimating a high

arrival rate for service interval l, even though this estimate may not be accurate because of a possible change in MMPP state. The high arrival rate, however, would result in a high capacity level for service interval l and thereby clear the backlog. A small v, on the other hand, means that the backlog may be small. Algorithm 1 responds by estimating a low arrival rate, which leads to a low capacity level for service interval l.

To summarize, an outline of our dynamic provisioning methodology using Algorithm 1 is shown in Figure 7.4.

At a decision point:

1. Obtain an estimate of the job arrival rate for the upcoming service interval as follows:
   a. Use measurement data on the number of arrivals collected over the previous service interval to determine v.
   b. Index NRT by v to retrieve λ_m.
2. Retrieve the capacity estimate corresponding to λ_m from the RCT.

Figure 7.4: Dynamic provisioning using Algorithm 1

7.3.2 Algorithm 2

Algorithm 2 uses a performance measurement based method to obtain an estimate of the arrival rate. We focus on the case where the number of MMPP states is two. Extensions of Algorithm 2 to more than two states is not straightforward and will not be considered. Nevertheless, we believe that the results for the case of two states are sufficient to show the effectiveness of a performance measurement based method for resource provisioning.

Consider two consecutive service intervals k and l as shown in Figure 7.5. Suppose the arrival

rate in state two is greater than that in state one, i.e., $\lambda_2 > \lambda_1$. At the decision point, Algorithm 2 determines from measurement data the fraction of jobs that meet the response time threshold x over service interval k (denoted by β^*). Based on this value of β^*, Algorithm 2 determines m, i.e., the estimated MMPP state for service interval l; the arrival rate estimate is then given by λ_m. The value of m is affected by the values of β (i.e., the target probability), β^* and a parameter denoted by ϵ $(0 \leq \epsilon \leq 1 - \beta)$; the procedure to determine m is shown in Figure 7.6.

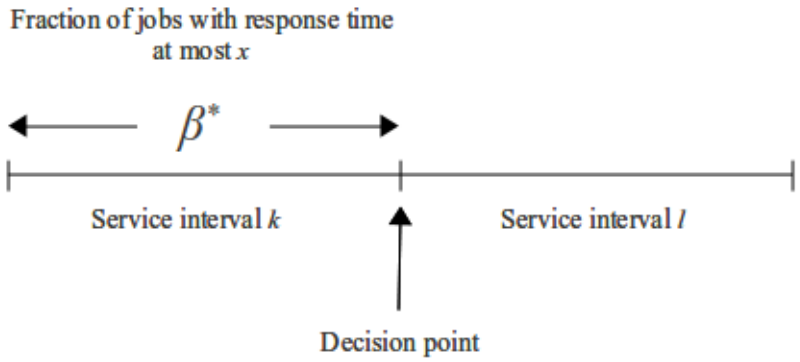

Figure 7.5: Algorithm 2

if $\beta^* \leq \beta + \epsilon$
  $m = 2$ (state with high arrival rate)
else
  $m = 1$ (state with low arrival rate)
endif

Figure 7.6: Procedure to determine m for service interval l

The rationale behind the design of Algorithm 2 is as follows. When β^* falls below $\beta + \epsilon$, there is a chance that a backlog is accumulated over service interval k. Algorithm 2 reacts by estimating a high arrival rate for service interval l which leads to a high capacity level to clear

the backlog. If, on the other hand, β^* is larger than $\beta + \epsilon$, the backlog is likely to be small. The algorithm responds by estimating a low arrival rate for service interval l, yielding a low capacity level.

Note that the objective of Algorithm 2 is not to achieve a target probability given by $\beta + \epsilon$. A larger value of ϵ only indicates a stronger tendency to provision capacity level high when a decision is made for the upcoming service interval. For example, for $\epsilon = 0.02$ and $\beta = 0.9$, a value of 0.93 for β^* would trigger a capacity adjustment to low. With $\epsilon = 0.04$, however, the same value for β^* would result in a capacity adjustment to high. Due to the complex nature of the system, it is not practical to determine the long-term impact of ϵ on the average capacity level when one considers all the service intervals over the operation interval. The value of ϵ is selected in a subjective manner; a higher value should be viewed as an attempt to remain conservative (in terms of provisioning the capacity level high) for the upcoming service interval.

To summarize, an outline of our dynamic provisioning methodology using Algorithm 2 is shown in Figure 7.7.

At a decision point:

1. Obtain an estimate of the job arrival rate for the upcoming service interval as follows:
   a. Use measurement data on response time collected over the previous service interval to determine β^*.
   b. Determine the estimated MMPP state m using β^* as input to the procedure in Figure 7.6.
2. Retrieve the capacity estimate corresponding to λ_m from the RCT.

Figure 7.7: Dynamic provisioning using Algorithm 2

7.3.3 Algorithm 3

Unlike Algorithms 1 and 2, Algorithm 3 is not based on advance knowledge that the traffic is MMPP. Consider again consecutive service intervals k and l as shown in Figure 7.8. Let λ be the arrival rate estimate for service interval l. Also, let v be the number of arrivals over service interval k, obtained by measurement. Algorithm 3 determines the value of λ using the following equation:

$$\lambda = \frac{v}{\text{TU}} \tag{7.4}$$

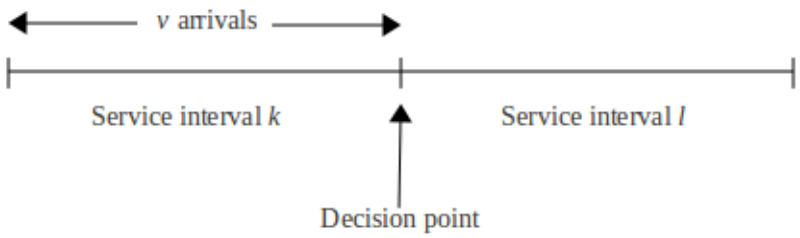

Figure 7.8: Algorithm 3

As mentioned in Section 7.2, we consider a scenario where the arrival rates are taken from a finite set $\mathbb{A}$ that is known in advance, and the capacity estimate for each arrival rate in $\mathbb{A}$ is determined offline and stored in the RCT. The arrival rate estimate obtained from Equation 7.4 can take on any positive value. The members of $\mathbb{A}$ should therefore be representative of the typical arrival rates that one expects to encounter. In addition, the value of λ may not be identical to any of the arrival rates in $\mathbb{A}$. In this case, the smallest member of $\mathbb{A}$ which is $\geq \lambda$, denoted by λ^*, is taken as an approximation for λ.

The rationale behind the design of Algorithm 3 is similar to that for Algorithm 1. When v is large, a higher arrival rate (and subsequently capacity level) is estimated. This would potentially

clear the backlog accumulated over the previous service interval. With a small v, on the other hand, the backlog is likely to be small, and a low arrival rate estimate would lead to releasing unnecessary capacity.

To summarize, an outline of our dynamic provisioning methodology using Algorithm 3 is shown in Figure 7.9.

At a decision point:

1. Obtain an estimate of the job arrival rate for the upcoming service interval as follows:
   a. Use measurement data on the number of arrivals collected over the previous service interval to determine v.
   b. Compute λ using Equation 7.4.
   c. Set λ^* to the smallest member of $\mathbb{A}$ which is $\geq \lambda$.
2. Retrieve the capacity estimate corresponding to λ^* from the RCT.

Figure 7.9: Dynamic provisioning using Algorithm 3

7.4 Evaluation

In this section, we present results on the effectiveness of our dynamic provisioning methodology. The experimental environment described in Section 5.4 is used in our evaluation. Briefly, the environment consists of a two-tier implementation of the TPC-W benchmark deployed over two VM's, each hosted on a separate node. The processing capacity of a VM ranges from 5% to 100% of the capacity of a single CPU core at increments of 5%, resulting in $N_C = 20$ levels of capacities; these capacities are denoted by $B_1 = 5\%, \cdots, B_{N_C} = 100\%$. A load generator resides

on a third node and submits jobs to the web server in accordance with a two-state MMPP over an operation interval of one hour.

The RCT table is populated in an offline manner as described in Section 7.2. To populate the RCT, one needs to define the set of arrival rates $\mathbb{A}$. For Algorithms 1 and 2, $\mathbb{A} = \{\lambda_1, \lambda_2\}$, where λ_j is the arrival rate in MMPP state j, $j = 1, 2$. For Algorithm 3, $\mathbb{A}$ contains arrival rates ranging from $\lambda_1 = \lambda_{\min} = 0.5$ to $\lambda_{\max}$ at increments of 0.5; $\lambda_{\max}$ is the largest arrival rate for which the optimization problem in Section 7.2 yields a feasible solution. In addition, for Algorithm 1, the NRT table is populated offline as described in Section 7.3. One also needs to determine the arrival rate for the first service interval. Since the system is assumed to be empty at the beginning of the operation interval (i.e., there is no backlog), the smallest member of $\mathbb{A}$ is taken as the estimate of arrival rate for the first service interval.

We now present the evaluation procedures for our dynamic provisioning methodology based on Algorithms 1, 2 and 3. Let $N_T = \textit{3600 seconds / TU in seconds}$ be the number of service intervals in a one-hour operation interval. Also, let $C_{w,l}$ and $C_{d,l}$ be the web and database server capacities estimated by our methodology for service interval l, $l = 1, \cdots, N_T$. The average capacity over the one-hour operation interval (denoted by C_a) is then given by:

$$C_a = \frac{1}{N_T} \sum_{l} (C_{w,l} + C_{d,l})$$

As in Chapters 5 and 6, let β' be the fraction of jobs meeting the response time threshold x over the one-hour operation interval. The deviation of β' from the target probability, given by $e = \beta' - \beta$, and the average capacity C_a are used as metrics in our evaluation.

The details of evaluation procedures for our methodology based on Algorithms 1, 2 and 3 are presented in Figures 7.10, 7.11 and 7.12, respectively. For convenience, we assume that for the two-state MMPP, $\lambda_2 > \lambda_1$.

Job arrivals

Start the load generator to submit jobs to the web server in accordance with an MMPP for a duration of one hour.

Service interval 1

At the beginning of service interval 1
  Retrieve capacity estimates $C_{w,1}$ and $C_{d,1}$ from the RCT using λ_1 as index.
  Deploy web and database servers on VM's with capacities $C_{w,1}$ and $C_{d,1}$, respectively.
During service interval 1
  Collect measurement data on the number of arrivals.
  Collect measurement data on the response time of each departure.

Service intervals 2 to N_T

for $l = 2$ **to** N_T
  At the beginning of service interval l
    Use the procedure in Figure 7.4 to determine capacity estimates $C_{w,l}$ and $C_{d,l}$.
    if $(C_{w,l}, C_{d,l}) \neq (C_{w,l-1}, C_{d,l-1})$
      Adjust the capacity of the web and database VM's to $C_{w,l}$ and $C_{d,l}$, respectively.
    endif
  During service interval l
    Collect measurement data on the number of arrivals.
    Collect measurement data on the response time of each departure.
endfor

Metrics e and C_a

From measurement data on response time, determine β', i.e., the fraction of jobs that meet the response time threshold over the one-hour operation interval.
Compute $e = \beta' - \beta$.
Compute $C_a = \frac{1}{N_T} \sum_l (C_{w,l} + C_{d,l})$.

Figure 7.10: Evaluation methodology for Algorithm 1

Job arrivals

Start the load generator to submit jobs to the web server in accordance with an MMPP for a duration of one hour.

Service interval 1

At the beginning of service interval 1
  Retrieve capacity estimates $C_{w,1}$ and $C_{d,1}$ from the RCT using λ_1 as index.
  Deploy web and database servers on VM's with capacities $C_{w,1}$ and $C_{d,1}$, respectively.
During service interval 1
  Collect measurement data on the response time of each departure.

Service intervals 2 to N_T

for $l = 2$ **to** N_T
  At the beginning of service interval l
    Use the procedure in Figure 7.7 to determine capacity estimates $C_{w,l}$ and $C_{d,l}$.
    if $(C_{w,l}, C_{d,l}) \neq (C_{w,l-1}, C_{d,l-1})$
      Adjust the capacity of the web and database VM's to $C_{w,l}$ and $C_{d,l}$, respectively.
    endif
  During service interval l
    Collect measurement data on the response time of each departure.
endfor

Metrics e and C_a

From measurement data on response time, determine β', i.e., the fraction of jobs that meet the response time threshold over the one-hour operation interval.
Compute $e = \beta' - \beta$.
Compute $C_a = \frac{1}{N_T} \sum_l (C_{w,l} + C_{d,l})$.

Figure 7.11: Evaluation methodology for Algorithm 2

Job arrivals

Start the load generator to submit jobs to the web server in accordance with an MMPP for a duration of one hour.

Service interval 1

At the beginning of service interval 1
  Retrieve capacity estimates $C_{w,1}$ and $C_{d,1}$ from the RCT using λ_1 as index.
  Deploy web and database servers on VM's with capacities $C_{w,1}$ and $C_{d,1}$, respectively.
During service interval 1
  Collect measurement data on the number of arrivals.
  Collect measurement data on the response time of each departure.

Service intervals 2 to N_T

for $l = 2$ **to** N_T
  At the beginning of service interval l
    Use the procedure in Figure 7.9 to determine capacity estimates $C_{w,l}$ and $C_{d,l}$.
    if $(C_{w,l}, C_{d,l}) \neq (C_{w,l-1}, C_{d,l-1})$
      Adjust the capacity of the web and database VM's to $C_{w,l}$ and $C_{d,l}$, respectively.
    endif
  During service interval l
    Collect measurement data on the number of arrivals.
    Collect measurement data on the response time of each departure.
endfor

Metrics e and C_a

From measurement data on response time, determine β', i.e., the fraction of jobs that meet the response time threshold over the one-hour operation interval.
Compute $e = \beta' - \beta$.
Compute $C_a = \frac{1}{N_T} \sum_l (C_{w,l} + C_{d,l})$.

Figure 7.12: Evaluation methodology for Algorithm 3

7.4.1 Parameter values

The three workloads in Table 5.3 are considered in our investigation. The default values of MMPP parameters for each workload are shown in Table 5.4. Unless stated otherwise, these values are used in our experiments. Consistent with the notation in Chapter 5, let $(-q_i)^{-1}$ be the average sojourn time in MMPP state i, $i = 1, 2$. We assume that the average sojourn time is smaller in MMPP state two than state one (i.e., smaller sojourn time at higher arrival rate). The values of TU considered in our investigation are $(-q_2)^{-1}$, $2(-q_2)^{-1}$ and $(-q_1)^{-1} + (-q_2)^{-1}$; the first two values are comparable to the average sojourn time in MMPP state two, and the last value is the same as the average cycle time. We recognize that it is uncertain whether or not such TU's will be used by an infrastructure provider in the future. Our objective, however, is not to make recommendations on TU sizes, but to explore whether dynamic provisioning would lead to cost savings when TU is comparable to average sojourn time in an MMPP state. For TU's that are noticeably larger than the average sojourn time in an MMPP state, static provisioning can be applied for resource provisioning purposes.

The values of x and β considered are 3 and 5 seconds, and 0.9 and 0.95, respectively. Our methodology in Section 7.2 may require the computation of service time distribution. When determining the service time CDF using the black-box approach, the number of jobs submitted by the load generator (denoted by N_J) is set to 2000. When computing the response time CDF for the M/PH/1 (FCFS) model, the upper bound on the number of jobs in system (denoted by N_u) is set to 50. The optimization problem is solved by conducting the search three times, and the best solution (i.e., the one with lowest capacity level) is taken as an approximation for the global optimum.

A parameter denoted by ϵ is included in Algorithm 2. Experiments are performed to determine the impact of this parameter on the metrics e and C_a. In these experiments, the default values

of MMPP parameters for Workload 1 are used. Each experiment is replicated five times, and the presented results are in the form of the mean of the five replications. For each value of β, two values of ϵ are considered; these values are 0.02 and 0.04 for $\beta = 0.9$, and 0.02 and 0.03 for $\beta = 0.95$.

The results for e and C_a for $\beta = 0.9$ are shown in Table 7.3. These results suggest that for a given TU, the two values of e for $\epsilon = 0.02$ and $\epsilon = 0.04$ (and the respective values of C_a) are quite similar. The largest observed difference between the two values of e is equal to 0.016 for TU = 25 seconds, $x = 5$ seconds and $\beta = 0.9$. As to C_a, the largest difference, given by 3.4% of the CPU capacity, is also observed at TU = 25 seconds, $x = 5$ seconds and $\beta = 0.9$. Based on the above observations, for $\beta = 0.9$, only $\epsilon = 0.04$ is considered in subsequent experiments for Algorithm 2.

Results for $\beta = 0.95$ are shown in Table 7.4. Again, it is observed that for a given TU, the difference between the two values of e for $\epsilon = 0.02$ and 0.03, and the respective difference between the values of C_a, are quite small, with the largest difference equal to 0.018 (for e) and 2.2% of the CPU capacity (for C_a), both observed at TU = 25 seconds, $x = 5$ seconds and $\beta = 0.95$. Thus, in subsequent experiments for Algorithm 2, we only consider $\epsilon = 0.03$ for $\beta = 0.95$.

Table 7.3: Comparison of results for $\epsilon = 0.02$ and 0.04, and $\beta = 0.9$

	TU = 5 seconds		TU = 10 seconds		TU = 25 seconds	
	$\epsilon = 0.02$	$\epsilon = 0.04$	$\epsilon = 0.02$	$\epsilon = 0.04$	$\epsilon = 0.02$	$\epsilon = 0.04$
(x, β)	e					
$(3, 0.9)$	0.033	0.029	0.032	0.041	0.026	0.038
$(5, 0.9)$	0.02	0.012	0.012	0.013	0.009	-0.007
(x, β)	C_a					
$(3, 0.9)$	144	144.7	147.7	148.3	150.7	151.2
$(5, 0.9)$	117.2	117.7	123	124.9	127.4	130.8

Table 7.4: Comparison of results for $\epsilon = 0.02$ and $\epsilon = 0.03$, and $\beta = 0.95$

	TU = 5 seconds		TU = 10 seconds		TU = 25 seconds	
	$\epsilon = 0.02$	$\epsilon = 0.03$	$\epsilon = 0.02$	$\epsilon = 0.03$	$\epsilon = 0.02$	$\epsilon = 0.03$
(x, β)	e					
$(3, 0.95)$	-0.005	-0.006	0	0.001	0.005	0.001
$(5, 0.95)$	0.008	0.011	0.008	0.006	0.012	-0.006
(x, β)	C_a					
$(3, 0.95)$	151.4	151.1	154.8	154.9	161.4	160.6
$(5, 0.95)$	135.4	135.3	140.2	140.1	145.7	147.9

7.4.2 Results and discussion

In this subsection, we present our experimental results and discuss the effectiveness of our dynamic resource provisioning strategy. Each experiment is replicated five times, and results presented for e and C_a are in the form of the mean and 95% confidence interval.

Workload 1

In the first set of experiments, the MMPP parameters are set to the default values for Workload 1, i.e., $\lambda = 1$ job/second, $\eta = 25$ seconds, $\gamma = 0.25$ and $\rho = 3$. With these values, the average sojourn time in MMPP states one and two are $(-q_1)^{-1} = 20$ and $(-q_2)^{-1} = 5$ seconds, respectively. The values of TU considered are 5 seconds $\big($or $(-q_2)^{-1}\big)$, 10 seconds $\big($or $2(-q_2)^{-1}\big)$, and 25 seconds $\big($or $(-q_1)^{-1} + (-q_2)^{-1}\big)$.

In Figure 7.13, e (the deviation from target probability) and C_a (the average capacity per service interval) are plotted against TU for $x = 3$ seconds and $\beta = 0.9$; the corresponding results for $x = 5$ and $\beta = 0.9$ are shown in Figure 7.14. Consider first the results for e. We observe that when TU = 5 seconds, all three algorithms yield positive values of e, implying that the performance target is met. When TU = 10 or 25 seconds, the performance difference of the three algorithms becomes quite noticeable. This is illustrated in Table 7.5 where results for (i) the number of cases where e is negative and (ii) the smallest and largest observed values of e (among the six cases considered[1]) are shown for each of the three algorithms.

[1]The six cases correspond to the six combinations of values for TU and response time threshold x (TU = 5, 10 and 25 seconds, and $x = 3$ and 5 seconds).

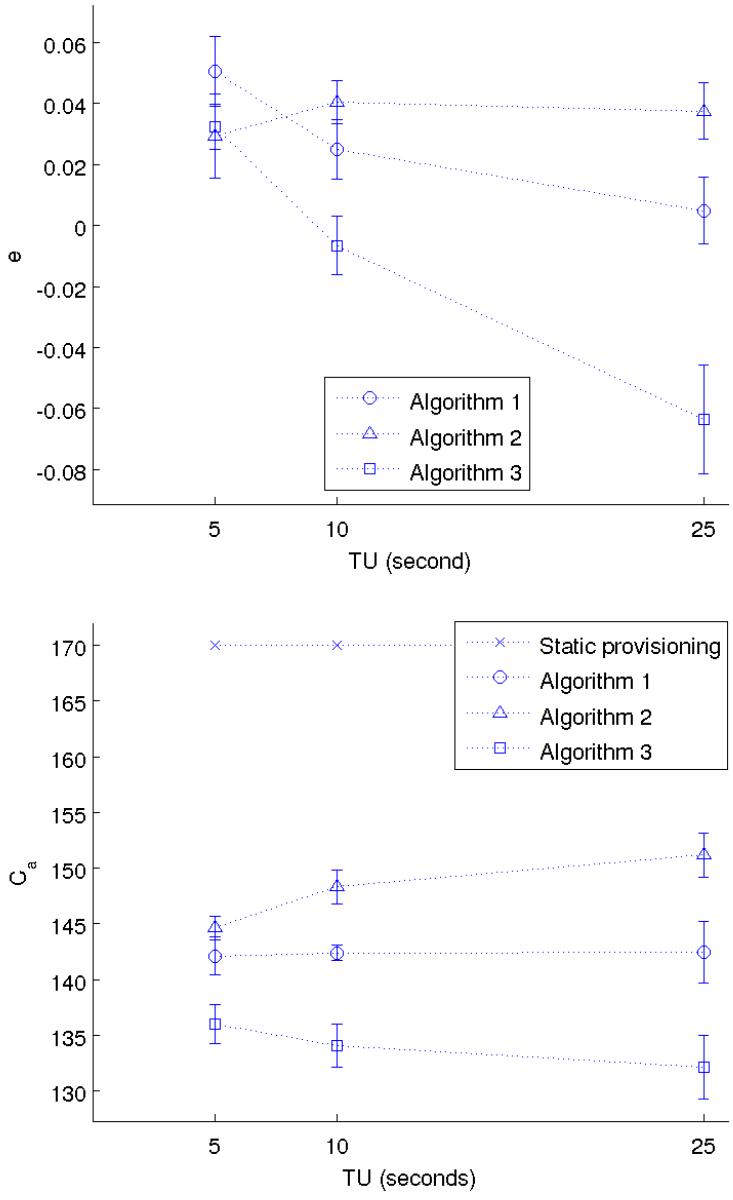

Figure 7.13: e and C_a vs. TU

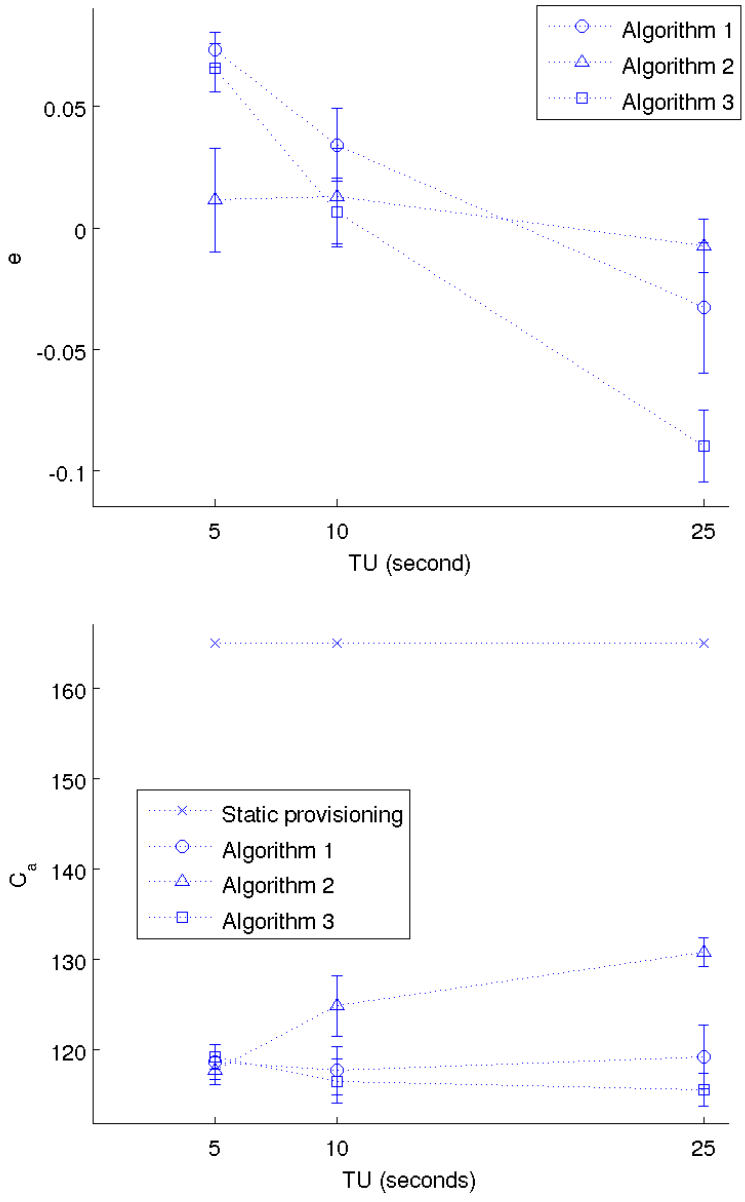

Figure 7.14: e and C_a vs. TU

Table 7.5: Observations on e and σ

Algorithm	Number of cases where $e < 0$			min e	max e	min σ	max σ
	TU = 5	TU = 10	TU = 25				
Algorithm 1	0	0	1	-0.033	0.074	16%	29%
Algorithm 2	0	0	1	-0.007	0.041	11%	29%
Algorithm 3	0	1	2	-0.090	0.066	20%	29%

The deviation of e from the ideal value of zero indicates inaccuracy in resource estimate. Similar to results in Chapters 5 and 6, the sources of inaccuracy are as follows: (*i*) The M/PH/1 (FCFS) model is used to predict the response time distribution; (*ii*) the optimization problem may return a local (instead of a global) optimum; (*iii*) VM capacities are defined in increments of 5%; and (*iv*) the service time distribution is obtained via system measurement using a black-box approach. For the dynamic provisioning strategy, the manner in which the arrival rates are estimated, the assumption that adjustment in capacity at a decision point takes effect immediately (i.e., zero provisioning delay instead of the 0.6 seconds discussed in Section 5.4.4), and the use of steady-state response time CDF instead of the time-dependent results for small service intervals are additional sources of inaccuracy.

Since e is positive for all three algorithms when TU = 5 seconds, we conclude that for the parameter values considered, our dynamic provisioning strategy with Algorithms 1, 2 or 3 yields acceptable results when TU is equal to the average sojourn time in MMPP state two (or the state with higher arrival rate and shorter average sojourn time), despite the sources of inaccuracy mentioned above. A positive value of e also indicates that an excessive amount of resources are provisioned. For the dynamic provisioning strategy, results for the degree of over-provisioning (defined in Section 5.5.3) are not available because it is not straightforward to determine, in an empirical manner, the capacity allocation for each service interval such that the performance target over the operation interval is met with minimal average capacity. Nevertheless, the largest

observed values of e for TU = 5 seconds (0.05 in Figure 7.13 and 0.074 in Figure 7.14) are similar to those observed for static provisioning in Chapter 5. We therefore feel that the levels of over-provisioning for our dynamic strategy is not significantly different from that for the static provisioning. Also, over-provisioning may not be undesirable because the extra capacity provides protection against deviation from the expected workload.

For TU = 10 and 25 seconds, we observe that for Algorithms 1 and 3, e tends to decrease with TU. Negative values of e are also observed for more cases. Among the five cases (out of 18) where e is negative, four occur at TU = 25 seconds (i.e., when TU is equal to the average cycle time). The results indicate that for the parameter values considered, Algorithm 2 is superior to Algorithms 1 and 3 because in the worst case, the deviation from the target probability for Algorithm 2 (given by 0.007) is significantly smaller than those for Algorithms 1 and 3 (given by 0.033 and 0.09, respectively). Furthermore, for Algorithm 2, e is observed to be rather insensitive to changes in TU. With e being negative for three of the six cases considered, Algorithm 3 exhibits the worst performance; it also yields the largest deviation of 0.09 from the target probability.

We next consider the results for C_a. We mentioned earlier that for static provisioning, the capacity level for each service interval is the same as that obtained from the MMPP/PH/1 (FCFS) model. Results for C_a for static provisioning are also plotted in Figures 7.13 and 7.14. We observe that dynamic provisioning (with Algorithms 1, 2 or 3) yields lower required capacity than static provisioning. This confirms our earlier remark that dynamic provisioning has the potential to lead to savings in capacity. Let C_{static} and $C_{dynamic}$ be the average capacity per service interval for static and dynamic provisioning, respectively. To measure the saving in capacity, we define a metric σ as follows:

$$\sigma = \frac{C_{static} - C_{dynamic}}{C_{static}} \times 100$$

The smallest and largest savings in capacity (or σ) for those cases where the performance target

is met (or e is positive) are also shown in Table 7.5. The capacity savings for Algorithm 1 are 16% and 29%, respectively. The corresponding values for Algorithms 2 and 3 are 11% and 29%, and 20% and 29%. We feel that the savings in capacity are sufficiently significant to support the conclusion that dynamic provisioning is effective in lowering the capacity requirement when compared to static provisioning.

Results for $\beta = 0.95$, and $x = 3$ and 5 seconds are shown in Figures 7.15 and 7.16, respectively. A summary of the values for e and σ is presented in Table 7.6. We again observe that when TU = 5 seconds (i.e., average sojourn time in MMPP state two), the performance of all three algorithms are acceptable; the performance target is met in five of the six cases considered, and for the only case where e is negative, the deviation from target probability is below 0.01. When TU = 10 or 25 seconds, Algorithm 2 consistently yields acceptable results; in only one case (out of four), e is negative, and for this case, the deviation from target probability is below 0.01. Algorithms 1 and 3, on the other hand, result in three cases (out of four) with negative values of e and significantly larger deviations from target probability (0.041 and 0.07, respectively). Given these observations, we again conclude that for the parameter values considered, Algorithm 2 exhibits superior performance compared to Algorithms 1 and 3. As to the average capacity, savings of up to 30% can be observed. This again supports the conclusion that our dynamic provisioning strategy is effective in reducing provisioning cost.

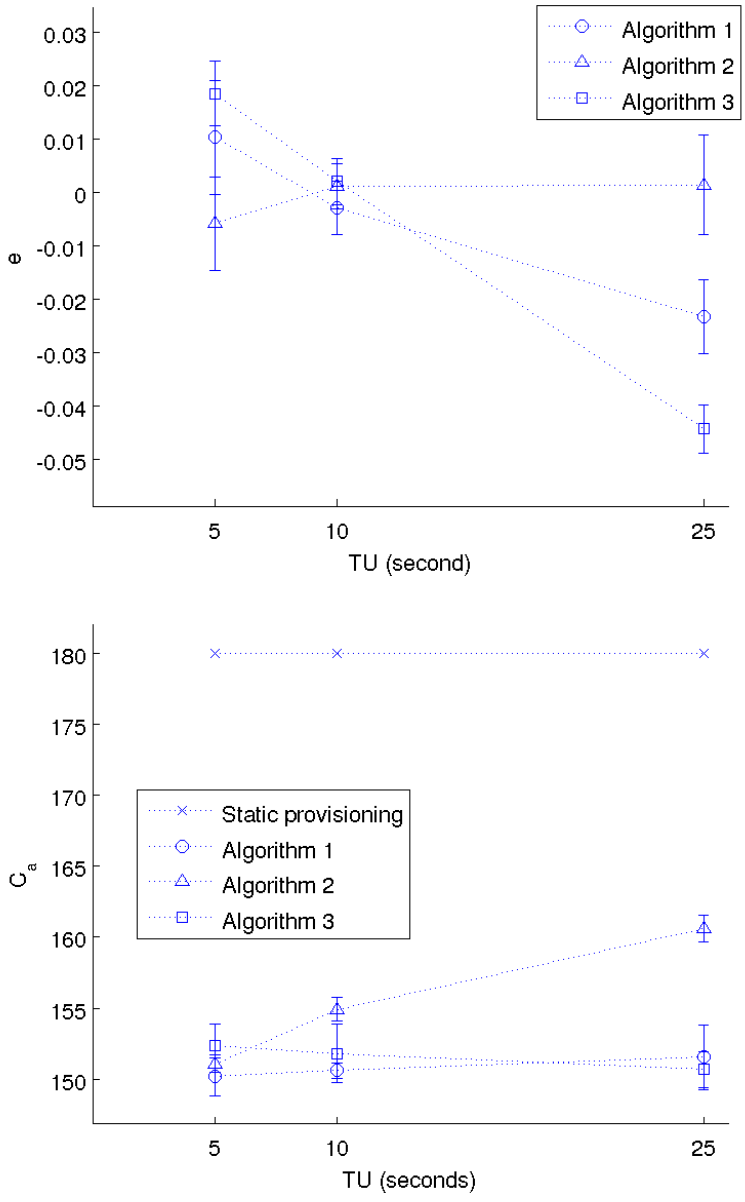

Figure 7.15: e and C_a vs. TU

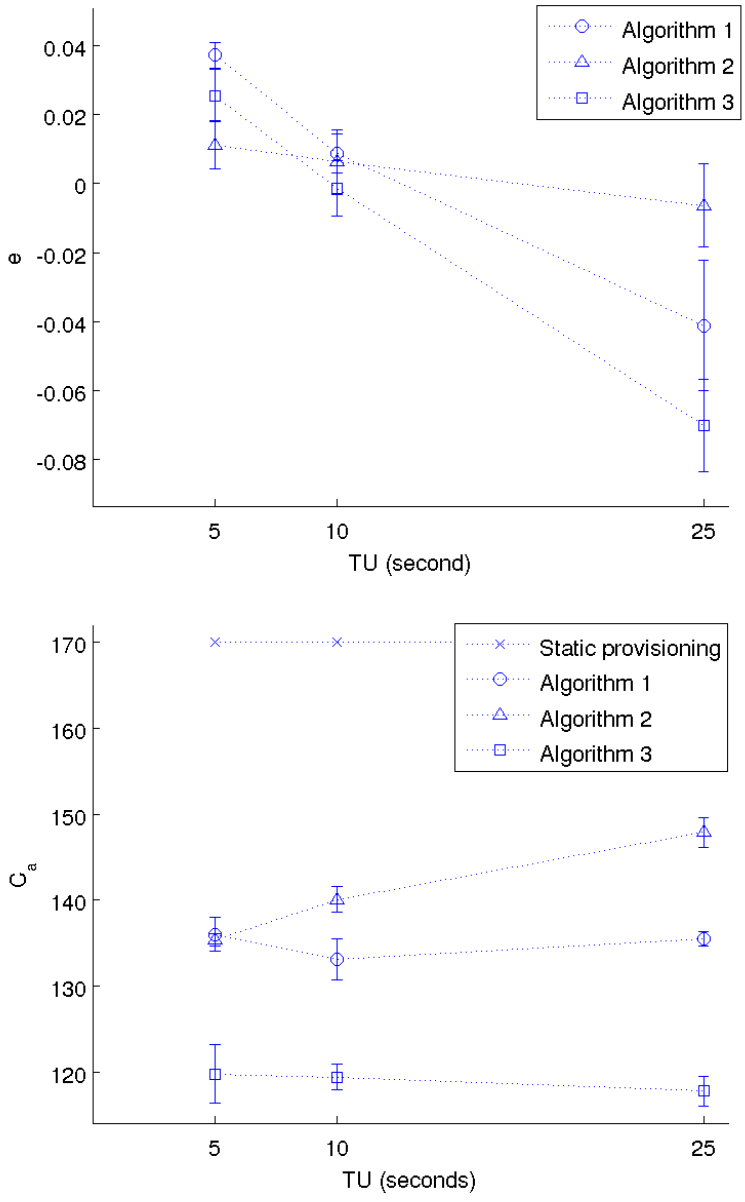

Figure 7.16: e and C_a vs. TU

Table 7.6: Observations on e and σ

Algorithm	Number of cases where $e < 0$			min e	max e	min σ	max σ
	TU = 5	TU = 10	TU = 25				
Algorithm 1	0	1	2	-0.041	0.037	17%	22%
Algorithm 2	1	0	1	-0.006	0.011	11%	20%
Algorithm 3	0	1	2	-0.070	0.026	15%	30%

We next consider the case where the mean arrival rate λ is increased from 1 to 1.5 jobs/second. In Figures 7.17 and 7.18, e and C_a are plotted against TU for $\beta = 0.9$, and $x = 3$ and 5 seconds, respectively. The corresponding results for $\beta = 0.95$ are shown in Figures 7.19 and 7.20. For each algorithm, 12 cases[2] are considered, and a summary of the observed values of e and σ is presented in Table 7.7. Similar conclusions as those for the case of $\lambda = 1$ job/second are drawn, namely, Algorithm 2 exhibits consistently acceptable results for the range of TU's considered, while the performance of Algorithms 1 and 3 is acceptable when TU = 5 seconds, and deteriorates as TU increases. The deviation from target probability, if any, is very small for Algorithm 2 (below 0.01), while that of Algorithms 1 and 3 reaches as high as 0.07 and 0.079, respectively. Capacity savings of up to 27% are observed.

[2]The 12 cases correspond to the 12 combinations of values for TU, response time threshold x and target probability β (TU = 5, 10 and 25 seconds, $x = 3$ and 5 seconds, and $\beta = 0.9$ and 0.95).

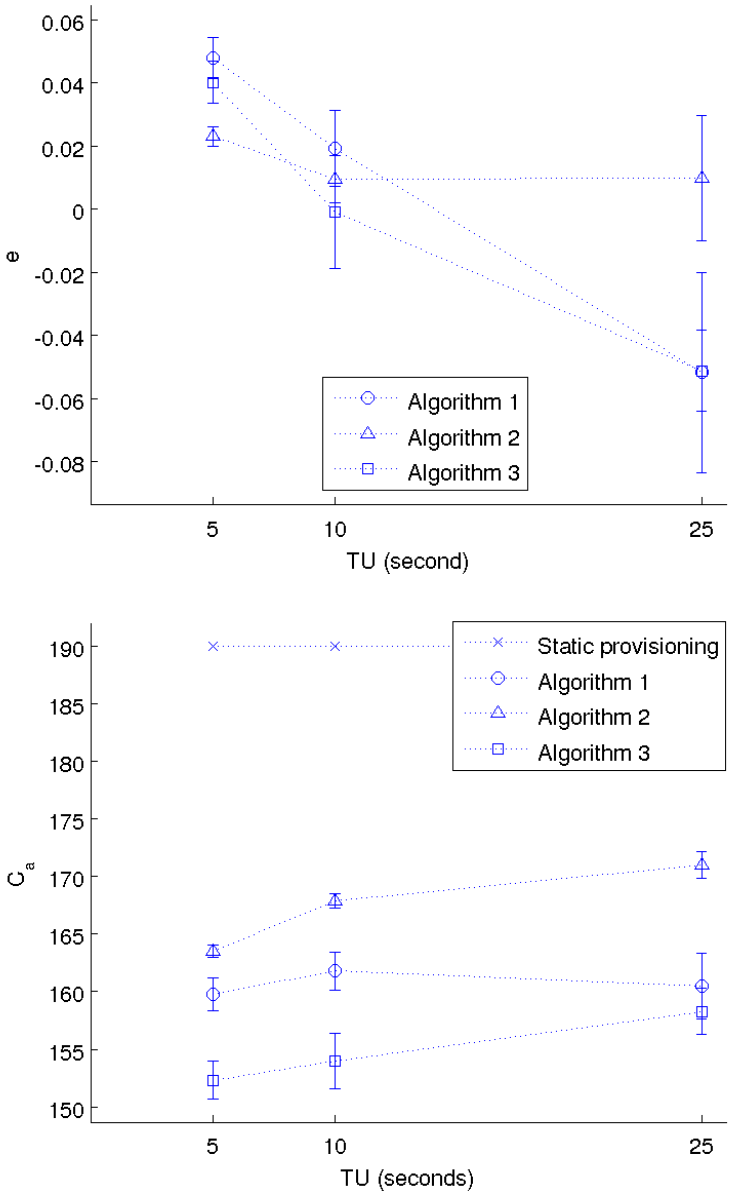

Figure 7.17: e and C_a vs. TU

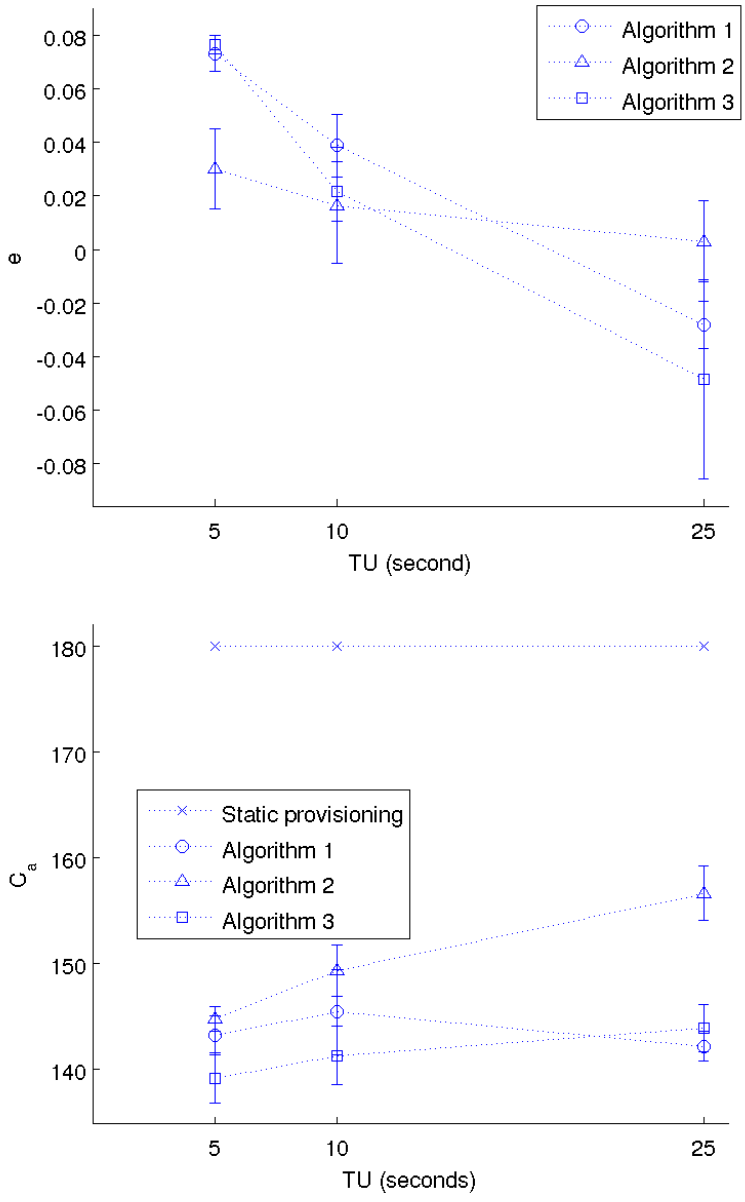

Figure 7.18: e and C_a vs. TU

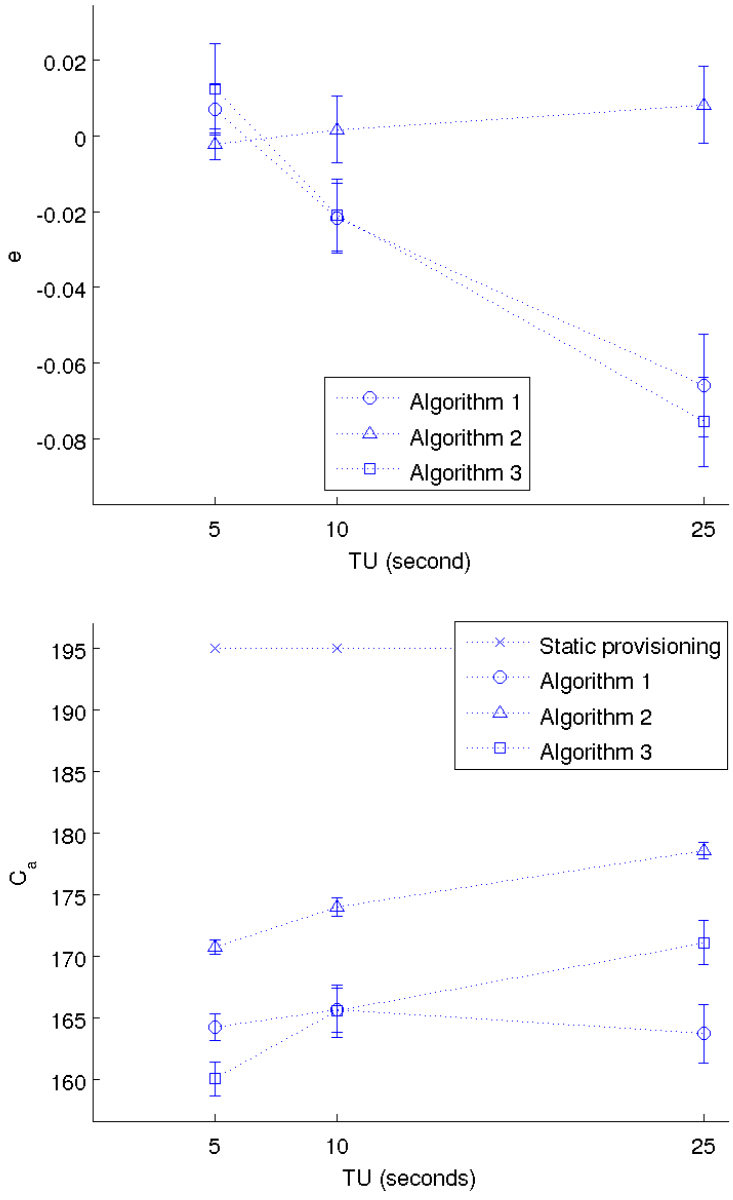

Figure 7.19: e and C_a vs. TU

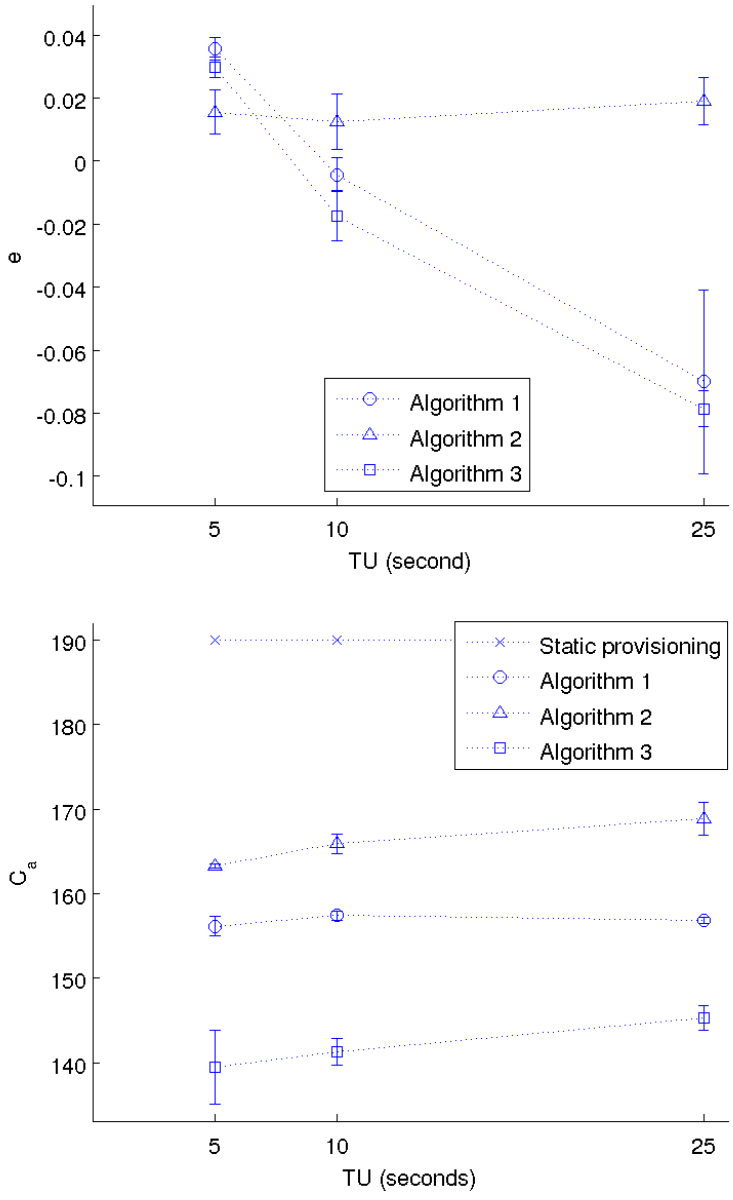

Figure 7.20: e and C_a vs. TU

Table 7.7: Observations on e and σ

Algorithm	Number of cases where $e < 0$			min e	max e	min σ	max σ
	TU = 5	TU = 10	TU = 25				
Algorithm 1	0	2	4	-0.070	0.073	15%	20%
Algorithm 2	1	0	0	-0.002	0.030	8%	20%
Algorithm 3	0	3	4	-0.079	0.077	18%	27%

In our next set of experiments, the value of λ is reset to 1 job/second, and ρ is increased to 5. Results for e and σ are obtained for TU = 5, 10 and 25 seconds, $x = 3$ and 5 seconds, and $\beta = 0.9$ and 0.95; the total number of cases considered for each algorithm is again 12. The observed trends are similar to those in Figures 7.13 to 7.20. We therefore do not present our results in the form of graphs, and present a summary of the observed values of e and σ in Table 7.8. The conclusions are similar, i.e., for the range of TU's considered, Algorithm 2 exhibits better performance compared to Algorithms 1 and 3; the latter two algorithms perform well only when TU is equal to the average sojourn time in MMPP state two (or five seconds). Capacity savings of up to 36% are observed.

Table 7.8: Observations on e and σ

Algorithm	Number of cases where $e < 0$			min e	max e	min σ	max σ
	TU = 5	TU = 10	TU = 25				
Algorithm 1	1	4	4	-0.270	0.034	25%	36%
Algorithm 2	1	0	1	-0.003	0.019	13%	34%
Algorithm 3	0	4	4	-0.167	0.063	22%	34%

Finally, we consider an MMPP with a longer average sojourn time at each state, specifically, $(-q_1)^{-1} = 40$ seconds and $(-q_2)^{-1} = 10$ seconds. There is a corresponding increase in the values of TU considered to 10, 20 and 50 seconds. The MMPP parameter values are $\lambda = 1$ job/second, $\eta = 50$ seconds, $\gamma = 0.25$ and $\rho = 3$. A summary of the results for e and σ is shown in Table

7.9. These results again lead to the conclusion that for the range of TU's considered, Algorithm 2 is superior to Algorithms 1 and 3, and there is a noticeable cost saving compared to static provisioning. Note that when TU is equal to 10 seconds, Algorithms 1 and 3 yield negative values of e in, respectively, one and three of the four cases considered for each algorithm. However, the worst case deviation from target probability is quite small, namely, 0.005 and 0.012 for Algorithms 1 and 3, respectively. We hence feel that Algorithms 1 and 3 still provide acceptable results when TU is equal to the average sojourn time in MMPP state 2 (or 10 seconds).

Table 7.9: Observations on e and σ

Algorithm	Number of cases where $e < 0$			min e	max e	min σ	max σ
	TU = 10	TU = 20	TU = 50				
Algorithm 1	1	3	4	-0.088	0.041	19%	29%
Algorithm 2	0	1	0	-0.001	0.038	11%	26%
Algorithm 3	3	4	4	-0.137	0.011	30%	30%

Workloads 2 and 3

In Tables 7.10 and 7.11, summaries of results for e and σ are presented for Workloads 2 and 3, respectively. For each workload, the MMPP parameters are set to the default values. The values of x, β and TU are set to 3 and 5 seconds, 0.9 and 0.95, and 5, 10 and 25 seconds, respectively. The conclusions are similar to those drawn for Workload 1. Capacity savings of up to 54% and 33% are observed for Workloads 2 and 3. Among a total of 24 cases considered for the two workloads, Algorithm 2 results in negative values of e in three cases, compared to 11 and 10 cases for Algorithms 1 and 3, respectively. For two of the three cases where e is negative for Algorithm 2, the deviation from target probability is below 0.01. We also observe that for Algorithm 2, the capacity saving is very small, i.e., 1%, in one case. Nevertheless, when both e and σ are taken into consideration, our conclusion is that Algorithm 2 has the best performance over the range

of TU values considered. We have conducted additional experiments for Workloads 2 and 3 as those for Workload 1. These results are not presented as similar observations are made.

Table 7.10: Observations on e and σ

Algorithm	Number of cases where $e < 0$			min e	max e	min σ	max σ
	TU = 5	TU = 10	TU = 25				
Algorithm 1	1	1	4	-0.077	0.042	24%	49%
Algorithm 2	0	0	0	0.001	0.053	7%	42%
Algorithm 3	0	1	3	-0.076	0.074	18%	54%

Table 7.11: Observations on e and σ

Algorithm	Number of cases where $e < 0$			min e	max e	min σ	max σ
	TU = 5	TU = 10	TU = 25				
Algorithm 1	1	1	3	-0.063	0.048	20%	28%
Algorithm 2	2	1	0	-0.018	0.039	1%	16%
Algorithm 3	1	2	3	-0.036	0.027	24%	33%

7.5 Conclusion

In this chapter, we presented a dynamic strategy for resource provisioning. Our strategy requires the estimation of job arrival rate for each service interval. Three algorithms to obtain an estimate of the arrival rate were presented. We evaluated the effectiveness of our dynamic strategy for the case where TU (or service interval) is comparable to the average sojourn time in an MMPP state.

The evaluation results indicate that our dynamic provisioning strategy has the potential to meet a performance target with a lower capacity level than static provisioning. Among the three algorithms used to estimate the arrival rate, Algorithm 2 consistently yields acceptable results for the range of TU's considered, while Algorithms 1 and 3 tend to perform well only when TU is equal to the average sojourn time in MMPP state two (the state with the higher arrival rate and

shorter average sojourn time). These results indicate that a performance-base algorithm (such as Algorithm 2), which uses response time data for resource provisioning decisions, is more effective than an algorithm which uses job arrival data, as in Algorithms 1 and 3.

In evaluation of our dynamic strategy with Algorithm 2, negative values of e were observed in 10 of the 72 cases considered. The deviation from target probability, however, is small (i.e., less than 0.01 in nine cases and equal to 0.018 in one case). A negative e is an indication that the average capacity per service interval (or C_a) is not sufficient to meet the performance target over the operation interval. Similar to Chapter 5, experiments are performed to gain insight into the increase in capacity in order to meet the performance target. Again, we select the web or database VM for capacity increase based on the values of $\bar{b}_w$ and $\bar{b}_d$ (see Table 5.3). Capacity increase is carried out at every decision point. Our results indicate that a capacity increase of 5% (i.e., the smallest amount defined by the target provider) is sufficient to meet the target in 8 of the 10 cases.

Our conclusion is that Algorithm 2 is the preferred algorithm for our dynamic provisioning methodology. This algorithm uses a parameter ϵ. Determining the optimal value of ϵ, i.e., the smallest value to meet the performance target over the operation interval, is not practical. However, our results indicate that for $\epsilon = 0.02$ and 0.04 when $\beta = 0.9$, and $\epsilon = 0.02$ and 0.03 when $\beta = 0.95$, the performance of Algorithm 2 is insensitive to the values of ϵ considered.

Finally, for Algorithm 2, e is observed to be rather insensitive to changes in TU. This means that our dynamic provisioning strategy with Algorithm 2 can be potentially effective when TU is smaller than one hour, yet longer than the average cycle time of the MMPP. On the other hand, given the trends observed in our evaluation, it is unlikely that Algorithms 1 and 3 would lead to acceptable results for such TU's. Evaluation of dynamic provisioning based on Algorithm 2 for longer TU's is left as future work.

Chapter 8

Conclusion and future work

In this thesis, we investigated the problem of resource provisioning for a web application under time-varying traffic. Queueing models were developed to determine the capacity requirement of an application to meet a performance target in the form of Pr[response time $\leq x$] $\geq \beta$. Major infrastructure providers such as Amazon, Google and Microsoft typically charge a resource on an hourly basis. We therefore investigated resource provisioning for service intervals of length one hour. Our work is concerned with the following three areas:

1. Identify a suitable traffic model to characterize job arrivals at a web application over a time interval of one hour.

2. Develop resource provisioning strategies to determine the capacity requirement of an application to meet a performance target over a time interval of one hour, and evaluate the effectiveness of these strategies using an experimental system.

3. Explore potential benefits of charging for resources using smaller time units than one hour in terms of savings in capacity (or cost) from the perspective of a subscriber.

The rest of this chapter is organized as follows. In Section 8.1, we summarize our findings and the contributions of this thesis. A description of research problems where further investigation can be conducted is presented in Section 8.2.

8.1 Contributions

In Chapter 3, we focused on fine-scale time-variation caused by the session-based nature of web traffic. We identified a suitable traffic model, namely, Markov-modulated Poisson process (MMPP), that has the potential to characterize such variation, and evaluated the effectiveness of this traffic model in the context of resource provisioning for web applications. Our evaluation methodology entails extending an available tool called BURN to generate, in a systematic and computationally efficient manner, a synthetic trace of job arrivals with a controlled level of time-variation, and fitting an MMPP to this trace. Our evaluation results confirmed the effectiveness of MMPP in modeling fine-scale time-variation in web application traffic. To the best of our knowledge, this study is the first to successfully use MMPP as a traffic model in the context of resource provisioning for web applications. We also proposed a variant of an available MMPP fitting algorithm that has the desired property of robust performance in the presence of outliers.

We next considered the problem of resource provisioning for a two-tier web application for a service interval of one hour. The first performance model is a single-server queue with MMPP arrivals, PH service time and FCFS discipline (i.e., an MMPP/PH/1 (FCFS) model). Analytic results for response time distribution of this model are used in an optimization problem to determine the capacity required to meet a given performance target. In Chapter 4, we presented a time-dependent analysis of the response time CDF of the MMPP/PH/1 (FCFS) model. These results are new, and represent advancement to the state of knowledge in the field. However, the computation of numerical results is prohibitively expensive. Numerical examples were presented

for a selected number of cases; these examples suggested that for parameter values consistent with those used in our investigation, the steady-state response time CDF is a good approximation when the service interval has length one hour.

Resource provisioning based on steady-state results for the single-server model was investigated in Chapter 5. As an alternative to results available in the literature, we developed an analysis based on Markov chain which is more suitable for numerical computation. A proof that the response time CDF of the MMPP/PH/1 (FCFS) model is of type PH was also presented. This response time CDF is then used in an optimization to determine the capacity required to meet the performance target. Our methodology is sufficiently general in the sense that it can be applied to any capacity definition so long as (i) capacities can be ordered in terms of increasing size and (ii) larger capacities are charged at higher rates. We evaluated the effectiveness of our methodology using a two-tier implementation of the TPC-W specification where the client emulator is set up to generate jobs in accordance with an MMPP. Our evaluation results indicated that the performance target is met in a majority of cases, and when it is violated, the deviations are small and can be corrected with a slight adjustment in capacity. Our results also indicated that the degree of over-provisioning, if any, is not overly excessive. These results led to the important finding that our methodology based on the single-server model is effective in predicting the required capacity to meet a given performance target. We also studied the relationship between MMPP parameters and required capacity. These results suggested that the required capacity tends to increase with the arrival rate and traffic burstiness.

A two-stage tandem queue was investigated in Chapter 6. We first proved that under a modest set of conditions, our methodology based on the two-stage tandem queue always yields lower capacity estimates than the single-server model; this would lead to potential savings in capacity. An extension of the steady-state analysis for the single-server model was developed

to obtain the response time CDF of the two-stage tandem queue. This CDF was then used in an optimization problem to predict the capacity requirement. Through experiments based on the TPC-W benchmark, we observed that the two-stage tandem queue does indeed yield lower capacity estimates, yet due to the assumptions made in the development of the methodology, these capacity estimates lead to violations of the performance target in a significant number of cases. We therefore concluded that between the two models, the single-server model is the preferred model for resource provisioning.

In Chapter 7, we investigated a scenario where an infrastructure provider uses smaller TU's than one hour for pricing purposes, and explored resource provisioning strategies for such service intervals. We focused on TU's that are comparable to the average sojourn time in an MMPP state. An operation interval of one hour was divided into multiple service intervals; each has length one TU. A dynamic provisioning strategy was developed where at a decision point, an estimate of the arrival rate is used as input to the M/PH/1 (FCFS) model to determine the required capacity such that the performance target is met over the upcoming service interval. Three algorithms to obtain an estimate of arrival rate were considered. Our evaluation results showed that for the range of TU's considered, our dynamic provisioning strategy has the potential to meet the performance target with smaller average capacity (or lower cost) compared to that obtained using the MMPP/PH/1 (FCFS) model in Chapter 5. We also found that among the three algorithms for estimating the arrival rate, an algorithm that takes advantage of advance knowledge that traffic is MMPP and uses performance data instead of number of job arrivals to estimate the arrival rate has the best performance. This algorithms also exhibited robustness in terms of acceptable performance over the complete range of TU's considered. It is also a good candidate for dynamic provisioning for larger values of TU.

8.2 Future work

We identify the following research problems where further investigation can be conducted in the future:

8.2.1 Time-varying traffic

Our work in Chapter 3 to evaluate the effectiveness of MMPP as traffic model entails generating a synthetic trace with a controlled level of time-variation. We extended an available tool called BURN to generate the synthetic trace; in this tool, time-variation is quantified in terms of the overdemand metric, which can be viewed as the tendency to keep one or more resources busy for a continuous period of time. We recognize that time-variation can be quantified in terms of other measures such as index of dispersion. MMPP is a versatile traffic model in the sense that the number of states can be aptly augmented to represent a diverse range of traffic patterns. Investigation with other measures of time-variation and other methods of generating the synthetic trace would provide additional evidence that MMPP is a suitable model for fine-scale time-variation.

8.2.2 Three-tier architecture

Throughout this thesis we focused on resource provisioning for a two-tier architecture, i.e., a web server and an application server consolidated in one machine, and a database server hosted by a second machine (each machine represents a tier). An interesting extension to our work would be a three-tier architecture where the web and application servers are hosted by separate machines (or tiers). This extension would require the modification of the optimization problem so that the search is conducted over the three-dimensional space (C_w, C_a, C_d) representing the capacities of the web, application and database servers, respectively. A suitable queueing model is a three-

stage tandem queue. However, the computation requirements for a model with three-dimensional state space is likely to be prohibitively expensive. Of interest is whether the approach presented in Chapter 5 where the three tiers are represented by a single-server model would result in accurate estimates of the required capacity.

8.2.3 Multiple applications

Another interesting extension to our work is the problem of resource provisioning for multiple applications. Each application has a different workload and performance target. The total amount of resources required may be reduced by organizing the applications into groups and provisioning the capacity at the group level (instead of the application level). For example, for the case of two applications, the two possible groupings are:

1. Provisioning resources for each application separately; in this case, there are two groups, each having one application.

2. Pooling the two applications together and provisioning resources for the application mix; in this case, there is one group with two applications.

For groups with one application, the MMPP/PH/1 (FCFS) model can be used directly to determine the required capacity. For groups with two or more applications, the aggregate arrival process is MMPP because it is the superposition of two or more MMPP's. The service time distribution is PH because it is a weighted sum of two or more PH distributions. Therefore, the MMPP/PH/1 (FCFS) model can be used. Each application grouping yields a different total required capacity to meet the performance target of all the applications. Of interest is to develop guidelines for application grouping such that the total capacity (or cost) is minimized.

8.2.4 Infrastructure providers

In our investigation, we focused on the provisioning of one resource only, i.e., the CPU. Infrastructure providers such as Amazon, on the other hand, organize an array of resources (e.g., CPU, memory and non-volatile storage) into packages (or instance types). These packages are charged for hourly usage according to a pricing policy. In our resource provisioning methodologies, the definition of capacity is quite generic; it is therefore suitable for resource provisioning for a broad range of operating environments. A comprehensive evaluation of our resource provisioning methodologies on computing environments governed by different pricing policies and platform setups would provide valuable insights into cost/performance trade-offs offered by various infrastructure providers. In addition, resource packages are typically VM's sharing a physical infrastructure. A related research problem is the evaluation of our methodologies on such platforms where several VM's contend for the underlying infrastructure.

8.2.5 Dynamic provisioning with Algorithm 2

In Chapter 7, we considered TU's which are comparable to the average sojourn time in an MMPP state. We observed that Algorithm 2, i.e., a performance-based algorithm with the advance knowledge that traffic is MMPP, led to consistently acceptable performance for the range of TU's considered. We feel that this algorithm would potentially perform well for a wider range of TU's. A selected number of experiments with default MMPP parameters for Workload 1 were conducted; in these experiments, $x = 3$ and 5 seconds, $\beta = 0.9$ and 0.95, and TU = 60, 120 and 300 seconds. These TU's are significantly longer than the average sojourn time in MMPP state two (or 5 seconds). Our evaluation results indicated that the performance target is met in all of the 12 cases; the smallest and largest observed values of e are 0.002 and 0.046, and the smallest and largest values of σ are 6% and 18%, respectively. These preliminary results are encouraging.

A comprehensive evaluation of dynamic provisioning with Algorithm 2 is hence a topic worthy of further investigation.

Bibliography

[1] http://aws.amazon.com/ec2.

[2] http://azure.microsoft.com/en-us.

[3] http://jmob.ow2.org/tpcw.html.

[4] http://rubis.ow2.org.

[5] https://cloud.google.com/products/app-engine.

[6] https://www.virtualbox.org.

[7] https://www.virtualbox.org/sdkref/index.html.

[8] http://www.vmware.com.

[9] Addis, B., Ardagna, D., Panicucci, B., Squillante, M., and Zhang, L. A hierarchical approach for the resource management of very large cloud platforms. *IEEE Transactions on Dependable and Secure Computing* (2013), 1.

[10] Ali-Eldin, A., Tordsson, J., and Elmroth, E. An adaptive hybrid elasticity controller for cloud infrastructures. In *Network Operations and Management Symposium (NOMS), 2012 IEEE* (2012), IEEE, pp. 204–212.

[11] Almeida, J., Almeida, V., Ardagna, D., Cunha, Í., Francalanci, C., and Trubian, M. Joint admission control and resource allocation in virtualized servers. *Journal of Parallel and Distributed Computing 70*, 4 (2010), 344–362.

[12] Amza, C., Chanda, A., Cox, A., Elnikety, S., Gil, R., Rajamani, K., Zwaenepoel, W., Cecchet, E., and Marguerite, J. Specification and implementation of dynamic web site benchmarks. In *Workload Characterization, 2002. WWC-5. 2002 IEEE International Workshop on* (2002), IEEE, pp. 3–13.

[13] Andersson, M., Cao, J., Kihl, M., and Nyberg, C. Performance modeling of an apache web server with bursty arrival traffic. In *In Proceedings of International Conference on Internet Computing (IC)* (2003).

[14] Andrew, L. L., Lin, M., and Wierman, A. Optimality, fairness, and robustness in speed scaling designs. In *ACM SIGMETRICS Performance Evaluation Review* (2010), vol. 38, ACM, pp. 37–48.

[15] Ardagna, D., Casolari, S., Colajanni, M., and Panicucci, B. Dual time-scale distributed capacity allocation and load redirect algorithms for cloud systems. *Journal of Parallel and Distributed Computing 72*, 6 (2012), 796–808.

[16] Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al. Above the clouds: A berkeley view of cloud computing. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28* (2009).

[17] Ascher, U. M. *Numerical methods for evolutionary differential equations*, vol. 5. Siam, 2008.

[18] Asmussen, S., Nerman, O., and Olsson, M. Fitting phase-type distributions via the em algorithm. *Scandinavian Journal of Statistics* (1996), 419–441.

[19] Bacigalupo, D. A., Van Hemert, J., Chen, X., Usmani, A., Chester, A. P., He, L., Dillenberger, D. N., Wills, G. B., Gilbert, L., and Jarvis, S. A. Managing dynamic enterprise and urgent workloads on clouds using layered queuing and historical performance models. *Simulation Modelling Practice and Theory 19*, 6 (2011), 1479–1495.

[20] Balachandran, A., Voelker, G., Bahl, P., and Rangan, P. Characterizing user behavior and network performance in a public wireless lan. In *ACM SIGMETRICS Performance Evaluation Review* (2002), vol. 30, ACM, pp. 195–205.

[21] Bali, S., and Frost, V. An algorithm for fitting mmpp to ip traffic traces. *Communications Letters, IEEE 11*, 2 (2007), 207–209.

[22] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating systems principles* (2003), ACM, p. 177.

[23] Bennani, M., and Menasce, D. Resource allocation for autonomic data centers using analytic performance models. In *Autonomic Computing, 2005. ICAC 2005. Proceedings. Second International Conference on* (2005), pp. 229–240.

[24] Cain, H., Rajwar, R., Marden, M., and Lipasti, M. An architectural evaluation of java tpc-w. In *hpca* (2001), Published by the IEEE Computer Society, p. 0229.

[25] Calcavecchia, N. M., Caprarescu, B. A., Di Nitto, E., Dubois, D. J., and Petcu, D. Depas: a decentralized probabilistic algorithm for auto-scaling. *Computing 94*, 8-10 (2012), 701–730.

[26] Casale, G., Kalbasi, A., Krishnamurthy, D., and Rolia, J. Automatically generating bursty benchmarks for multitier systems. *ACM SIGMETRICS Performance Evaluation Review 37*, 3 (2010), 32–37.

[27] Casale, G., Kalbasi, A., Krishnamurthy, D., and Rolia, J. Burn: Enabling workload burstiness in customized service benchmarks. *Software Engineering, IEEE Transactions on*, 99 (2011), 1–1.

[28] Casale, G., Mi, N., Cherkasova, L., and Smirni, E. Dealing with burstiness in multi-tier applications: Models and their parameterization. *Software Engineering, IEEE Transactions on 38*, 5 (2012), 1040–1053.

[29] Chaisiri, S., Lee, B.-S., and Niyato, D. Optimization of resource provisioning cost in cloud computing. *Services Computing, IEEE Transactions on 5*, 2 (2012), 164–177.

[30] Chandra, A., Gong, W., and Shenoy, P. Dynamic resource allocation for shared data centers using online measurements. *Lecture notes in computer science* (2003), 381–400.

[31] Chydzinski, A. Transient analysis of the mmpp/g/1/k queue. *Telecommunication Systems 32*, 4 (2006), 247–262.

[32] Ciciani, B., Santoro, A., and Romano, P. Approximate analytical models for networked servers subject to mmpp arrival processes.

[33] Computing, A., Seracini, F., Menarini, M., and Krüger, I. A comprehensive resource management solution for web-based systems. 233–239.

[34] Cremonesi, P., and Sansottera, A. Indirect estimation of service demands in the presence of structural changes. *Performance Evaluation 73* (2014), 18–40.

[35] Cunha, I., Almeida, J., Almeida, V., and Santos, M. Self-Adaptive Capacity Management for Multi-Tier Virtualized Environments. In *Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on* (2007), pp. 129–138.

[36] Deng, L., and Mark, J. Parameter estimation for markov modulated poisson processes via the em algorithm with time discretization. *Telecommunication Systems 1*, 1 (1993), 321–338.

[37] Diao, Y., Hellerstein, J., Parekh, S., Griffith, R., Kaiser, G., and Phung, D. A control theory foundation for self-managing computing systems. *IEEE journal on selected areas in communications 23*, 12 (2005), 2213.

[38] Dutta, S., Gera, S., Verma, A., and Viswanathan, B. Smartscale: Automatic application scaling in enterprise clouds. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on* (2012), IEEE, pp. 221–228.

[39] Ellens, W., Zivkovic, M., Akkerboom, J., Litjens, R., and van den Berg, H. Performance of cloud computing centers with multiple priority classes. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on* (2012), IEEE, pp. 245–252.

[40] Escheikh, M., Jouini, H., and Barkaoui, K. A versatile traffic and power aware performability analysis of server virtualized systems.

[41] Faisal, A., Petriu, D. C., and Woodside, C. M. Network latency impact on performance of software deployed across multiple clouds. In *CASCON* (2013), pp. 216–229.

[42] Fischer Kathleen, W. The markov-modulated poisson process (mmpp) cookbook. *Performance Evaluation 18*, 2 (1993), 149–171.

[43] Gandhi, A., Dube, P., Karve, A., Kochut, A., and Zhang, L. Adaptive, model-driven autoscaling for cloud applications. In *Proceedings of the 11th International Conference on Autonomic Computing, Philadelphia, PA, USA* (2014).

[44] García, D., and García, J. Tpc-w e-commerce benchmark evaluation. *Computer 36*, 2 (2003), 42–48.

[45] Ghanbari, S., Soundararajan, G., Chen, J., and Amza, C. Adaptive learning of metric correlations for temperature-aware database provisioning. In *Fourth International Conference on Autonomic Computing, 2007. ICAC'07* (2007), pp. 26–26.

[46] Gmach, D., Rolia, J., Cherkasova, L., and Kemper, A. Workload analysis and demand prediction of enterprise data center applications. In *Workload Characterization, 2007. IISWC 2007. IEEE 10th International Symposium on* (2007), IEEE, pp. 171–180.

[47] Goudarzi, H., and Pedram, M. Multi-dimensional sla-based resource allocation for multi-tier cloud computing systems. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on* (2011), IEEE, pp. 324–331.

[48] Goudarzi, H., and Pedram, M. Geographical load balancing for online service applications in distributed datacenters. In *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on* (2013), IEEE, pp. 351–358.

[49] Hamilton, J. Cost of power in large-scale data centers, November 2008.

[50] Harrison, P., Harrison, S., Patel, N., and Zertal, S. Storage workload modelling by hidden markov models: Application to flash memory. *Performance Evaluation 69*, 1 (2012), 17–40.

[51] Heffes, H., and Lucantoni, D. A markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance. *Selected Areas in Communications, IEEE Journal on 4*, 6 (1986), 856–868.

[52] Heindl, A. Decomposition of general tandem queueing networks with mmpp input. *Performance Evaluation 44*, 1 (2001), 5–23.

[53] Heindl, A. Decomposition of general queueing networks with mmpp inputs and customer losses. *Performance Evaluation 51*, 2-4 (2003), 117–136.

[54] Horváth, A., and Telek, M. Phfit: A general phase-type fitting tool. In *Computer Performance Evaluation: Modelling Techniques and Tools*. Springer, 2002, pp. 82–91.

[55] Hu, Y., Wong, J., Iszlai, G., and Litoiu, M. Resource provisioning for cloud computing. In *Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research* (2009), ACM, pp. 101–111.

[56] Jiang, D., Pierre, G., and Chi, C.-H. Autonomous resource provisioning for multi-service web applications. In *Proceedings of the 19th international conference on World wide web* (2010), ACM, pp. 471–480.

[57] Jie, Y., Qiu, J., and Li, Y. A profile-based approach to just-in-time scalability for cloud applications. In *Cloud Computing, 2009. CLOUD'09. IEEE International Conference on* (2009), IEEE, pp. 9–16.

[58] Jung, G., Joshi, K., Hiltunen, M., Schlichting, R., and Pu, C. Generating adaptation policies for multi-tier applications in consolidated server environments. In *Autonomic Computing, 2008. ICAC'08. International Conference on* (2008), pp. 23–32.

[59] Keshav, S. *Mathematical foundations of computer networking*. Pearson Education, 2012.

[60] Khayari, R. E. A., Sadre, R., and Haverkort, B. R. Fitting world-wide web request traces with the em-algorithm. *Performance Evaluation 52*, 2 (2003), 175–191.

[61] Kim, Y., and Un, C. Performance analysis of statistical multiplexing for heterogeneous bursty traffic in an atm network. *Communications, IEEE Transactions on 42*, 234 (1994), 745–753.

[62] Kobayashi, H., and Mark, B. L. *System modeling and analysis: foundations of system performance evaluation*. Pearson Education India, 2009.

[63] Krishnamurthy, D., Rolia, J., and Majumdar, S. A synthetic workload generation technique for stress testing session-based systems. *IEEE Transactions on Software Engineering* (2006), 868–882.

[64] Krishnamurthy, D., Rolia, J., and Xu, M. Wam- the weighted average method for predicting the performance of systems with bursts of customer sessions. *Software Engineering, IEEE Transactions on 37*, 5 (2011), 718–735.

[65] Kusic, D., and Kandasamy, N. Risk-aware limited lookahead control for dynamic resource provisioning in enterprise computing systems. *Cluster Computing 10*, 4 (2007), 395–408.

[66] Kusic, D., Kephart, J. O., Hanson, J. E., Kandasamy, N., and Jiang, G. Power and performance management of virtualized computing environments via lookahead control. *Cluster computing 12*, 1 (2009), 1–15.

[67] Lazowska, E. D., Zahorjan, J., Graham, G. S., and Sevcik, K. C. *Quantitative system performance: computer system analysis using queueing network models*. Prentice-Hall, Inc., 1984.

[68] Lin, M., Wierman, A., Andrew, L. L., and Thereska, E. Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Transactions on Networking (TON) 21*, 5 (2013), 1378–1391.

[69] Manson, J. R., Wallis, S. G., and Hope, D. A conservative semi-lagrangian transport model for rivers with transient storage zones. *Water resources research 37*, 12 (2001), 3321–3329.

[70] Mao, M., and Humphrey, M. Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis* (2011), ACM, p. 49.

[71] Mazzucco, M. Towards autonomic service provisioning systems. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing* (2010), IEEE Computer Society, pp. 273–282.

[72] Menascé, D., Almeida, V., Riedi, R., Ribeiro, F., Fonseca, R., and Meira Jr, W. In search of invariants for e-business workloads. In *Proceedings of the 2nd ACM conference on Electronic commerce* (2000), ACM, pp. 56–65.

[73] Meng, X., Isci, C., Kephart, J., Zhang, L., Bouillet, E., and Pendarakis, D. Efficient resource provisioning in compute clouds via VM multiplexing. In *Proceeding of the 7th international conference on Autonomic computing* (2010), ACM, pp. 11–20.

[74] Mosberger, D., and Jin, T. httperf a tool for measuring web server performance. *ACM SIGMETRICS Performance Evaluation Review 26*, 3 (1998), 31–37.

[75] Neuts, M. A queue subject to extraneous phase changes. *Advances in Applied Probability 3*, 1 (1971), 78–119.

[76] Neuts, M. Generalizations of the pollaczek-khinchin integral equation in the theory of queues. *Advances in applied probability* (1986), 952–990.

[77] Panchenko, A., and Thummler, A. Efficient phase-type fitting with aggregated traffic traces. *Performance Evaluation 64*, 7-8 (2007), 629–645.

[78] Rajabi, A., and Wong, J. Mmpp characterization of web application traffic. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2012 IEEE 20th International Symposium on* (2012), IEEE, pp. 107–114.

[79] Revzina, J. Possibilities of mmpp processes for bursty traffic analysis. *Reliability and Statistics in Transportation and Communication* (2010).

[80] Riska, A., Diev, V., and Smirni, E. An em-based technique for approximating long-tailed data sets with ph distributions. *Performance Evaluation 55*, 1 (2004), 147–164.

[81] Roy, N., Dubey, A., and Gokhale, A. Efficient autoscaling in the cloud using predictive models for workload forecasting. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on* (2011), IEEE, pp. 500–507.

[82] Sedaghat, M., Hernandez-Rodriguez, F., and Elmroth, E. A virtual machine re-packing approach to the horizontal vs. vertical elasticity trade-off for cloud autoscaling. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference* (2013), ACM, p. 6.

[83] Shah-Heydari, S., and Le-Ngoc, T. MMPP models for multimedia traffic. *Telecommunication Systems 15*, 3 (2000), 273–293.

[84] Sharma, U., Shenoy, P., and Towsley, D. F. Provisioning multi-tier cloud applications using statistical bounds on sojourn time. In *Proceedings of the 9th international conference on Autonomic computing* (2012), ACM, pp. 43–52.

[85] Shivam, P., Babu, S., and Chase, J. Learning application models for utility resource planning. In *IEEE International Conference on Autonomic Computing, 2006. ICAC'06* (2006), pp. 255–264.

[86] Singh, R., Sharma, U., Cecchet, E., and Shenoy, P. Autonomic mix-aware provisioning for non-stationary data center workloads. In *Proceedings of the 7th international conference on Autonomic computing* (2010), ACM, pp. 21–30.

[87] Staniforth, A., and Côté, J. Semi-lagrangian integration schemes for atmospheric models-a review. *Monthly weather review 119*, 9 (1991), 2206–2223.

[88] Sung, L. G. A., and Wong, J. W. Autonomic resource management for a cluster that executes batch jobs. In *Cluster Computing, 2006 IEEE International Conference on* (2006), IEEE, pp. 1–10.

[89] Sutton, C. A., and Jordan, M. I. Inference and learning in networks of queues. In *International Conference on Artificial Intelligence and Statistics* (2010), pp. 796–803.

[90] Tesauro, G., Das, R., Walsh, W., and Kephart, J. Utility-function-driven resource allocation in autonomic systems. In *Second International Conference on Autonomic Computing, 2005. ICAC 2005. Proceedings* (2005), pp. 342–343.

[91] Thümmler, A., Buchholz, P., and Telek, M. A novel approach for fitting probability distributions to real trace data with the em algorithm. In *DSN* (2005), vol. 5, pp. 712–721.

[92] Trushkowsky, B., Bodík, P., Fox, A., Franklin, M. J., Jordan, M. I., and Patterson, D. A. The scads director: Scaling a distributed storage system under stringent performance requirements. In *FAST* (2011), pp. 163–176.

[93] Urgaonkar, B., Shenoy, P., Chandra, A., Goyal, P., and Wood, T. Agile dynamic provisioning of multi-tier Internet applications. *ACM Transactions on Autonomous and Adaptive Systems 3*, 1 (2008).

[94] Vallamsetty, U., Kant, K., and Mohapatra, P. Characterization of e-commerce traffic. *Electronic Commerce Research 3*, 1-2 (2003), 167–192.

[95] Van Doorn, E. A., and Pollett, P. K. Survival in a quasi-death process. *Linear Algebra and its applications 429*, 4 (2008), 776–791.

[96] Van Houdt, B., and Alfa, A. S. Response time in a tandem queue with blocking, markovian arrivals and phase-type services. *Operations Research Letters 33*, 4 (2005), 373–381.

[97] Watson, B. J., Marwah, M., Gmach, D., Chen, Y., Arlitt, M., and Wang, Z. Probabilistic performance modeling of virtualized resource allocation. In *ICAC '10: Proceeding of the 7th international conference on Autonomic computing* (New York, NY, USA, 2010), ACM, pp. 99–108.

[98] Wildstrom, J., Stone, P., and Witchel, E. Carve: A cognitive agent for resource value estimation. In *Autonomic Computing, 2008. ICAC'08. International Conference on* (2008), IEEE, pp. 182–191.

[99] Witten, I., and Frank, E. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[100] Woodside, M., Zheng, T., and Litoiu, M. Service system resource management based on a tracked layered performance model. In *IEEE International Conference on Autonomic Computing, 2006. ICAC'06* (2006), pp. 175–184.

[101] Wu, L., Garg, S. K., Versteeg, S., and Buyya, R. Sla-based resource provisioning for hosted software as a service applications in cloud computing environments. *IEEE Transactions on Services Computing* (2013), 1.

[102] Wu, Y., Min, G., Ould-Khaoua, M., and Yin, H. Modelling and analysis of pipelined circuit switching in interconnection networks with bursty traffic and hot-spot destinations. *Journal of Systems and Software* (2011).

[103] Xiong, P., Wang, Z., Malkowski, S., Wang, Q., Jayasinghe, D., and Pu, C. Economical and robust provisioning of n-tier cloud workloads: A multi-level control approach. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on* (2011), IEEE, pp. 571–580.

[104] Yechiali, U., and Naor, P. Queuing problems with heterogeneous arrivals and service. *Operations Research 19*, 3 (1971), 722–734.

[105] Yue, C., and Wang, H. Profit-aware admission control for overload protection in e-commerce web sites. In *Quality of Service, 2007 Fifteenth IEEE International Workshop on* (2007), IEEE, pp. 188–193.

[106] Zhang, Q., Cherkasova, L., and Smirni, E. A regression-based analytic model for dynamic resource provisioning of multi-tier applications. In *Fourth International Conference on Autonomic Computing, 2007. ICAC'07* (2007), pp. 27–27.

[107] Zhang, Q., Zhu, Q., Zhani, M. F., Boutaba, R., and Hellerstein, J. L. Dynamic service placement in geographically distributed clouds. *Selected Areas in Communications, IEEE Journal on 31*, 12 (2013), 762–772.

Appendices

Appendix A

Notation

Table A.1: Notation

Symbol	Interpretation
a_j	The arrival time of the j^{th} job at SS or TN
b	The total service time of a job at the web and database servers
b_d	The total service time of a job at the database server
$\bar{b}_d$	The average service time on the database server when $C_w = C_d = B_{N_C}$
$b_d(j)$	The service time of the j^{th} database call
b_w	The service time of a job at the web server
$\bar{b}_w$	The average service time on the web server when $C_w = C_d = B_{N_C}$
$c(B_j)$	The cost of obtaining B_j for one TU
d	The service time of a job
$\bar{d}$	The average service time of a session

Continued on next page

Table A.1 – Continued from previous page

Symbol	Interpretation
d_{j1}	The service time of the j^{th} job at the first stage of TN
d_{j2}	The service time of the j^{th} job at the second stage of TN
e	$= \beta' - \beta$
e_{ij}	The probability that the next session is from BM_j given that the current session is from BM_i
$f(\cdot)$	The job-level service time CDF in MMPP trace
$f_i(\cdot)$	The job-level service time CDF for BM_i
$g_i(\cdot)$	The session-level service time CDF for BM_i
$h(y)$	The instantaneous service completion rate
$k_{\max}$	The upper bound on k dimension in k-v grid
$\bar{n}_i$	The average session length for BM_i
$p_i(\cdot)$	The session length pdf for BM_i
q_i	$= -\sum_{j \neq i} q_{ij}$
q_{ij}	The transition rate from MMPP state i to MMPP state j
$r_{ij}(y,k,x,v,t)$	the joint probability that (i) there are $v \geq 0$ arrivals in $(t_0, t_0 + t)$, (ii) MMPP state at $t_0 + t = j$, and (iii) the unfinished work in the system immediately after $t_0 + t \leq x$, given that (i) the MMPP state at $t_0 = i$, (ii) there are $k \geq 0$ jobs in the system at t_0, and if $k \geq 1$, the first job has been served for $y \geq 0$ (for $k = 0$, the only valid value for y is zero), (iii) and there is an arrival at $t_0 + t$
$r(x)$	The steady-state response time CDF

Continued on next page

Table A.1 – Continued from previous page

Symbol	Interpretation
$r(x,T)$	The time-dependent response time CDF of the MMPP/PH/1 (FCFS) model
re_1	The relative error between C and C'
re_2	The relative error between β and β'
s_{k0}	The transition rate from PH transient phase k to absorbing state 0
s_{kl}	The transition rate from PH transient phase k to transient phase l
$s_{m,k0}$	The transition rate from PH transient phase k to absorbing state 0 at stage m $(= 1, 2)$
$s_{m,kl}$	The transition rate from PH transient phase k to transient phase l at stage m $(= 1, 2)$
t_0	The beginning of a time interval
t_j	The departure time of the j^{th} job from TN
$t_{\max}$	The upper bound on t dimension in t-y grid
$v_{\max}$	The upper bound on v dimension in k-v grid
x	The response time threshold
x_{lj}	The service time of the j^{th} job of the l^{th} session
y_l	The session-level service time of the l^{th} session
$y_{\max}$	The upper bound on y dimension in t-y grid
$z_i(\cdot)$	The think time pdf for BM_i
$\mathbb{A}$	A set consisting of the arrival rate estimates

Continued on next page

Table A.1 – Continued from previous page

Symbol	Interpretation
$A(n,k)$	The steady-state probability of being in state (n,k) of the Markov chain $\mathcal{M}_S$ $(M=1)$ from the perspective of an arrival
$A(i,n,k)$	The steady-state probability of being in state (i,n,k) of the Markov chain $\mathcal{M}_S$ from the perspective of an arrival
$A(i,n_1,k_1,n_2,k_2)$	The steady-state probability of being in state (i,n_1,k_1,n_2,k_2) of the Markov chain $\mathcal{M}_T$ from the perspective of an arrival
$\mathbb{B}$	The set containing all the available capacities
B_j	An available capacity level
BM_i	The i^{th} input benchmark
C	The server capacity to meet a performance target when jobs are submitted in accordance with the synthetic trace
C'	The server capacity to meet a performance target when jobs are submitted in accordance with the MMPP trace
C_a	The average capacity per service interval
C_{avg}	The average of $C_{\min}$ and $C_{\max}$
C_d	The capacity of the database server
$C_{d,l}$	The capacity of the database server in service interval l
$C_{dynamic}$	The average capacity from dynamic provisioning
C_e	The optimal total capacity obtained from experiment
$C_{\max}$	The upper bound on C or C'
$C_{\min}$	The lower bound on C or C'

Continued on next page

Table A.1 – Continued from previous page

Symbol	Interpretation
C_o	The optimal total capacity obtained from the optimization problem
C_{static}	The average capacity from static provisioning
C_w	The capacity of the web server
$C_{w,l}$	The capacity of the web server in service interval l
$\mathbf{D}$	A diagonal matrix with $\frac{1}{1-\psi_i(0,T)}$'s on the main diagonal
$\mathbf{E}$	The policy matrix computed by BURN
$F(y)$	The service time CDF
$G_{n,k}(x)$	The CDF of the time to absorption of $\mathcal{M}'_S$ from state (n,k)
$G_{n_1,k_1,n_2,k_2}(x)$	The CDF of the time to absorption of $\mathcal{M}'_T$ from state (n_1,k_1,n_2,k_2)
$H(y,k,x)$	the probability that the unfinished work immediately after $t_0 \leq x$, given that there are $k+1$ jobs in system at t_0, and if $k \geq 1$, the first job has been served for $y \geq 0$
$\mathbf{I}$	The identity matrix
$I_x(y)$	$= \begin{cases} 1 & y \leq x \\ 0 & y > x \end{cases}$
IDC	The index of dispersion for counts
K	The number of PH transient phases
L	The number of sessions
L_l	The number of jobs in the l^{th} session

Continued on next page

Table A.1 – Continued from previous page

Symbol	Interpretation
M	The number of MMPP states
M_R	The number of regions the synthetic trace is divided into
$\mathcal{M}_S$	The Markov chain representing the MMPP/PH/1 (FCFS) (or M/PH/1 (FCFS)) model
$\mathcal{M}'_S$	The Markov chain derived from $\mathcal{M}_S$ after cutting off the arrival process
$\mathcal{M}_T$	The Markov chain representing the two-stage tandem queue
$\mathcal{M}'_T$	The Markov chain derived from $\mathcal{M}_T$ after cutting off the arrival process
N	The number of input benchmarks
N_b	The number of database calls by a job
N_C	The number of available capacities
N_J	The number of jobs
N_{R_i}	The number of observed arrival rates falling into region $(\delta_{i+1}, \delta_i]$
N_t	The number of arrivals in a time interval of length t
N_T	The number of service intervals in the operation interval
N_u	The upper bound on the number of jobs in system
NRT	The number of arrivals to arrival rate look-up table
OV	The degree of over-provisioning
$P(n,k)$	The steady-state probability of being in state (n,k) of the Markov chain $\mathcal{M}_S$ $(M=1)$

Continued on next page

Table A.1 – Continued from previous page

Symbol	Interpretation
$P(i,n,k)$	The steady-state probability of being in state (i,n,k) of the Markov chain $\mathcal{M}_S$
$P(i,n_1,k_1,n_2,k_2)$	The steady-state probability of being in state (i,n_1,k_1,n_2,k_2) of the Markov chain $\mathcal{M}_T$
$\mathbf{Q}$	The MMPP transition rate matrix
$\mathbf{R}(y,k,x,v,t)$	A square matrix with entries $r_{ij}(y,k,x,v,t)$
$\mathcal{R}_{SS}$	A random variable denoting the response time of the single-server model
$\mathcal{R}_{TN}$	A random variable denoting the response time of the two-stage tandem queue
RCT	The rate to capacity look-up table
$\mathbf{S}$	The PH transition rate matrix
SS	The single-server model
TN	The two-stage tandem queue
TU	The minimum time unit for charging of resources
V	A random variable denoting the number of arrivals over a service interval
X	A sample of service times for the single-server model
X_d	A sample of job service times at the database server
X_i	A sample of job-level service times for BM_i
X_w	A sample of job service times at the web server

Continued on next page

Table A.1 – Continued from previous page

Symbol	Interpretation
Y_i	A sample of session-level service times for BM_i
$\boldsymbol{\alpha}$	The session mix in the synthetic trace
α_i	The fraction of sessions from BM_i in the synthetic trace
β	The target probability
β'	The fraction of jobs meeting the response time threshold obtained from experiment
β^*	A temporary variable denoting the fraction of jobs meeting the response time threshold x
γ	The ratio of the average sojourn time in MMPP state two to that in MMPP state one
δ_i	The boundary line between regions i and $i-1$
δ_{ij}	The Kronecker delta function
$\bar{\delta}_{ij}$	$= 1 - \delta_{ij}$
ϵ	A small positive number
$\boldsymbol{\zeta}$	The PH initial probability distribution
ζ_k	The probability that PH initial phase is k
$\zeta_{m,k}$	The probability that PH initial phase at stage m $(= 1, 2)$ is k
η	The average cycle time
θ_i	The long-term fraction of time spent in MMPP state i
κ	A constant used in the definition of cost function

Continued on next page

Table A.1 – Continued from previous page

Symbol	Interpretation
λ	The mean arrival rate
λ^*	The smallest member of $\mathbb{A}$ which is $\geq \lambda$
λ_{ij}	The j^{th} observed arrival rate in region $(\delta_{i+1}, \delta_i]$
λ_j	The arrival rate in MMPP state j
$\lambda_{\max}$	The maximum arrival rate
$\lambda_{\min}$	The minimum arrival rate
λ_{sess}	The session arrival rate
ξ_i	The average of two boundaries of region $(\delta_{i+1}, \delta_i]$
τ_j	The departure time of the j^{th} job from SS
$\boldsymbol{\pi}$	A row vector with elements π_i
π_i	The probability that initial MMPP state is i
ρ	The ratio of the arrival rate in MMPP state two to that in MMPP state one
σ	The saving in capacity
ϕ	The overdemand metric
$\phi_{\max}$	The maximum achievable value for overdemand
$\phi_{\min}$	The minimum achievable value for overdemand
$\psi_i(v, t)$	The probability that there are $v \geq 0$ arrivals in $[t_0, t_0 + t]$, given that the MMPP state at t_0 is i
ω_i	The fraction of jobs from BM_i in the synthetic trace

Continued on next page

Table A.1 – Continued from previous page

Symbol	Interpretation
$\mathbf{\Lambda}$	A diagonal matrix with λ_j's on the main diagonal
Φ	A random variable denoting the estimated MMPP state at a decision point
$\mathbf{\Psi}(v,t)$	A column vector with entries $\psi_i(v,t)$