Dynamic Resource Provisioning for an Interactive System

by

ShaoWen Acer Lu

A thesis

presented to the University of Waterloo

in fulfillment of the

thesis requirement for the degree of

Master of Mathematics

in

Computer Science

Waterloo, Ontario, Canada, 2009

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

In a data centre, server clusters are typically used to provide the required processing capacity to provide acceptable response time performance to interactive applications. The workload of each application may be time-varying. Static allocation to meet peak demand is not an efficient usage of resources. Dynamic resource allocation, on the other hand, can result in efficient resource utilization while meeting the performance goals of individual applications.

In this thesis, we develop a new interactive system model where the number of logon users changes over time. Our objective is to obtain results that can be used to guide dynamic resource allocation decisions. We obtain approximate analytic results for the response time distribution at steady state for our model. Using numerical examples, we show that these results are acceptable in terms of estimating the steady state probabilities of the number of logon users. We also show by comparison with simulation that our results are acceptable in estimating the response time distribution under a variety of dynamic resource allocation scenarios. More importantly, we show that our results are accurate in terms of predicting the minimum number of processor nodes required to meet the performance goal of an interaction application. Such information is valuable to resource provisioning and we discuss how our results can be used to guide dynamic resource allocation decisions.

Acknowledgements

It is my pleasure to work under my supervisor, Professor Johnny Wong. The work would not be successful without his expert guidance and experience through my research. His invaluable comments and sights on the research have facilitated the completion of this thesis. His patience and encouragement help me overcome many difficulties. Thanks are given to Professor Martin Karsten and Professor David Taylor for taking time to be the committee readers of this thesis.

I would like to thank all my friends for bringing pleasure time during my master study in Waterloo. I express my sincere thanks to my parents and grandparents for their love, support and encouragement since my childhood.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A data center is typically a large distributed computer environment consisting of heterogeneous computing resources including individual servers, server clusters, databases, etc. It is equipped with communications, power and cooling systems. The data center hosts different application services for networked access from diverse client devices. It processes jobs submitted by users via the Internet.

Due to the decreasing cost of commodity hardware and the advancement in Internet technologies, data centers are increasingly used to host diverse services by web content publishers and application service providers. At the same time, the size of data centers has also increased; more server clusters are used to provide more processing capacity to execute computationally intensive jobs and shorten the response time of interactive requests. Resource management in large data centers is an important issue, especially for the next generation of data centers. Traditionally this is done manually, which takes a long time and often results in poor resource utilization. The resource

management task is also complicated by scenarios where workloads are time-varying. For example, the peak-to-average load ratio at an Internet search service is typically 3:1 over a one-day period [1]. Static allocation to meet peak demand is not cost-effective because of server over-provisioning. Therefore, dynamic resource allocation schemes are needed.

Autonomic resource provisioning is an attractive approach for resource management in data centers. Its objective is to automate the dynamic allocation of resources in order to minimize the mean amount of resources used and meet the performance goals of individual applications as specified in their service level agreements (SLAs).

Jobs processed in data centers can be broadly categorized into two types: batch jobs and interactive jobs. Batch jobs are usually scheduled to run in batch mode because they need to run for a long time. In contrast, interactive jobs are a kind of human-computer conversation and have smaller processing requirements. The performance goals of batch jobs are normally related to maximizing the system throughput or the fraction of jobs that are completed on time. On the other hand, the performance goals of interactive jobs are usually expressed in terms of mean response time or distribution of response time. Because of the above differences, provisioning approaches designed for interactive jobs are not directly applicable to provisioning in a batch environment. This thesis is concerned with dynamic resource provisioning for interactive jobs only.

A dynamic resource allocation scheme is usually implemented in a logical loop of three phases (see Figure 1.1):

i) Measure: collect the workload, state, and performance data such as the arrival rate of jobs, the number of jobs in the system and the mean response time.

ii) Decide: use the measured data and/or historical data as input to a resource allocation algorithm to determine the amount of resources needed.

iii) Execute: implement changes in the amount of resources allocated, if required.

As an example, the IBM Tivoli Intelligent Orchestrator (TIO) is a dynamic resource allocation engine, which implements the logical loop shown in Figure 1.1. It contains four components: data acquisition engine, objective analyzer, resource broker, and deployment engine [2], where the Decide phase is included in the objective analyzer and resource broker components. Figure 1.2 from [2] shows the general framework of TIO.

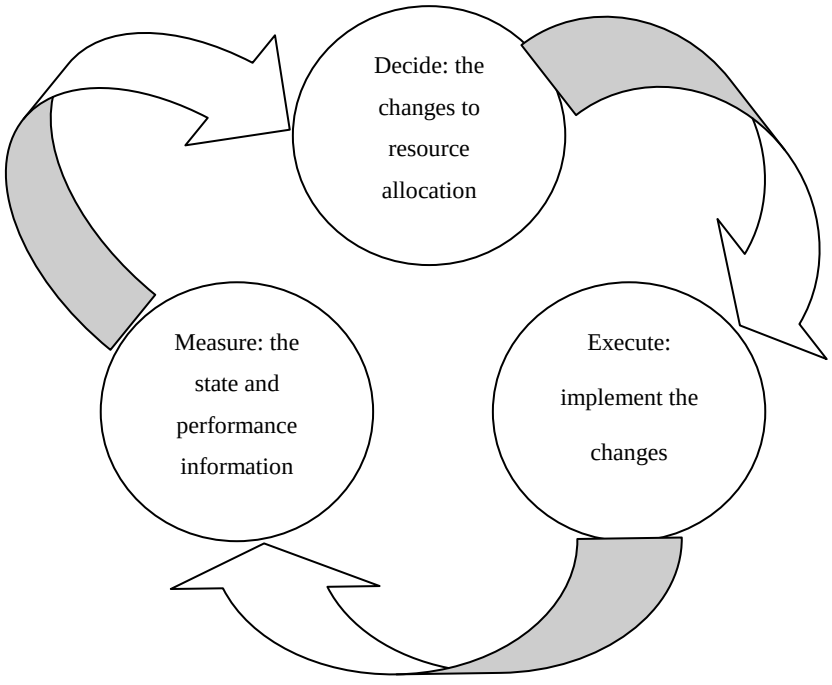

Figure 1.1: The logical loop of three phases

In our investigation, the computing resources under consideration are the servers in a server cluster. Our focus is on resource allocation decisions made in the Decide phase of the logical loop. A considerable amount of work has been done in this area. A summary will be presented in Chapter 2. Our approach is different in the sense that resource allocation decisions are guided by a predictive performance model and the model used captures the time-varying resource requirements of interactive applications. Briefly, our model is developed for a scenario where (i) logon users submit jobs to a system consisting of a pool of m parallel servers, (ii) the number of logon users changes over time as new sessions are established and existing sessions are terminated, (iii) the number of logon users at any time instant is restricted to a maximum. To our knowledge, analytic results for such a model are not available in the open literature.

In our investigation, the performance metric under consideration is response time distribution. The relevant SLA is Prob[response time $\leq t$] $\geq p$. This is different from most other studies where only the mean response time is considered.

The key contribution of this thesis is the derivation of approximate analytic results for the response time distribution for our model. These results are new and are shown to be acceptable by comparison with exact results obtained by numerical methods and by simulation. Our results can be used to predict the minimum number of processor nodes required in order to meet the SLA of an interactive application.

The remainder of this thesis is organized as follows. Chapter 2 contains a review of related work. This includes different approaches for the design of dynamic allocation algorithms. In Chapter 3, the system architecture under consideration is described to a level of detail that is sufficient for the development of a performance model. A web-

based system is also described as an example of such an architecture. Our performance model is presented in Chapter 4. Chapter 5 presents approximate analytic results for the response time distribution for our model. The accuracy of these results is evaluated by comparison with exact analytic results obtained by numerical methods. In Chapter 6, the merit of our approximate analytic results in estimating the response time distribution and predicting the minimum number of processor nodes required in order to achieve a given SLA is evaluated by comparison with results obtained by simulation. Finally, Chapter 7 contains a summary of our findings and a discussion of future work.

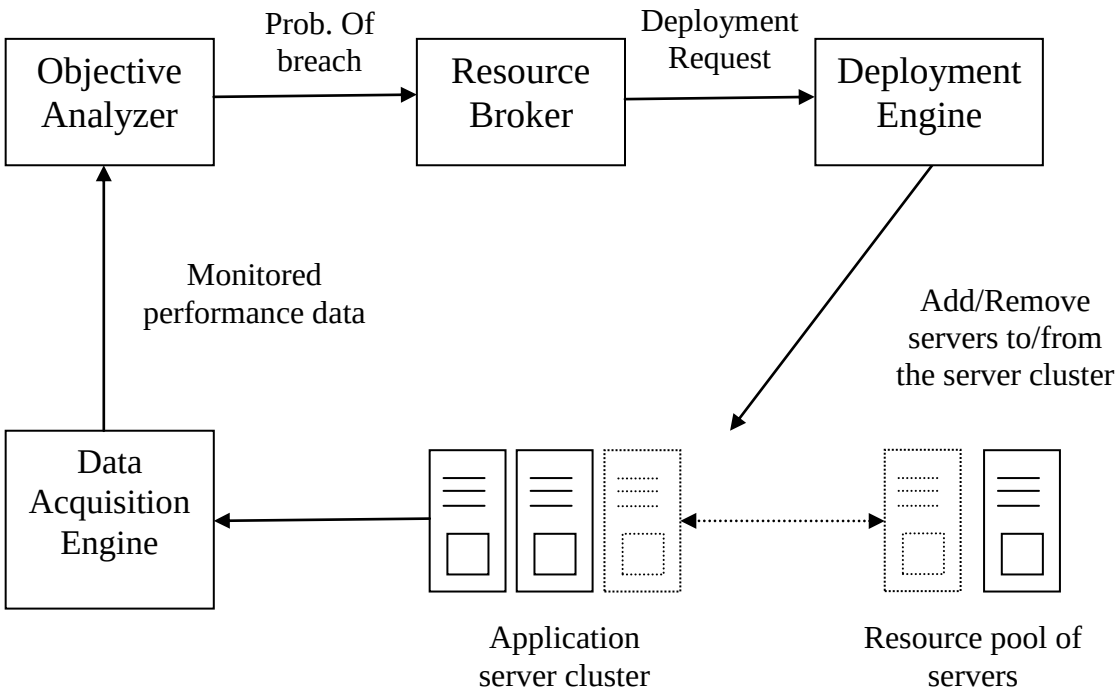

Figure 1.2: TIO framework

Chapter 2

Related work

Approaches that have been used in the design of dynamic resource allocation algorithms can be classified into performance models, control theory, and learning methods. The work presented in this thesis uses the performance model approach because this approach is of great practical and theoretical importance in solving performance issues by using simulation and sophisticated mathematical methods.

2.1 Performance models

Large data centers are used to host multiple application services, whose workloads may vary widely and may be hard to predict. This creates challenges for dynamic resource allocation (e.g., scalability). A solution is presented in [3, 4] where a combination of analytic queueing network models and combinatorial search techniques is used to determine the best possible configuration for the data center. Multiclass queuing network models are used to model interactive and batch jobs [5].

Internet applications usually employ a multi-tier architecture. Each tier provides a certain functionality, such as web server, application server, and database storage. Common examples are e-commerce websites. Dynamic resource provisioning has been used to determine the amount of resources to allocate to each tier of the application. In [6], a statistical regression-based analytic model is developed to approximate the CPU demands of different transaction types along all the tiers in the system. In another study, an analytical model based on queueing theory is presented in [7] to determine the number of servers to be allocated to each tier of a multi-tier application.

Internet applications may also have a single-tier architecture. Examples include clustered HTTP servers. A model-based resource provisioning approach is proposed in [8]. This approach focuses on a coordinated provisioning of memory and storage resources using cache hit ratio and storage response time as provisioning goals.

2.2 Control theory

In control theory, a controller is used to manipulate the inputs to a system to obtain the desired effect on the output of the system. In [9], a feedback-driven resource control system is designed based on a two-layered controller architecture. The objective is to dynamically allocate resources to individual tiers of a multi-tier application to meet quality of service goals. A performance-targeted feedback-controlled real-time scheduling algorithm is described in [10]. It uses a combination of local periodic real-time scheduling and a global feedback control system to maintain the execution rates specified by users while keeping the system utilization at an acceptable level.

In [11], a multiple-input, multiple-output feedback-driven resource control mechanism is presented to adjust the CPU and memory usage in each Apache application to achieve the desired utilization targets.

2.3 Learning methods

In dynamic resource allocation, learning methods are typically used to analyze the performance and workload histories to predict future behaviors of applications in order to achieve the applications' requirement. The workload histories usually consist of measurement data gathered from previous executions under varying assignments of resources. In [12], the Non-invasive Modeling for Optimization prototype is presented for batch applications. This prototype is designed to collect a modest amount of training data to develop accurate models to optimize resource assignments for complex workflows across a networked utility.

Reinforcement learning has also been used in the investigation of dynamic resource provisioning. This method does not require prior training of the application systems so that no workload histories are needed. It searches through all possible allocations to determine the best allocation for a particular system state in order to meet quality of service requirements. In [13], a reinforcement learning-based approach is designed to address the problem of dynamic allocation of storage bandwidth to applications in the context of enterprise-scale storage systems.

Chapter 3

System description

This chapter describes the cluster-based interactive system that we will use in our investigation. As an example, a web-based system is also described.

3.1 System architecture

We consider an interactive computer system consisting of a server cluster. There are a number of logon users who interact with this system; the interaction is based on the client server model. A session has been created for each logon user. Since new sessions may be created and existing sessions may be terminated, the number of logon users may change over time. Specifically, the number of logon users is increased by 1 each time a new session is created and decreased by 1 each time an existing session is terminated.

The server cluster can be viewed as a service facility that has a pool of resources. In our investigation, the resources under consideration are processor nodes. We will refer to these nodes as server nodes. The service facility provides a full server utility

model, where a server node is dedicated to run one application for one user at any time instant. This is in contrast to a shared server utility model where different applications and different users can share the same server node. A software product based on the full server utility model can be found in [14]. It exploits the use of virtual LANs and SANs for partitioning of resources into secure domains called virtual application environments. This system architecture allows resource allocation to be done dynamically, namely, the number of server nodes allocated to an application may change over time in response to changing workload.

In general, dynamic resource allocation has two potentially conflicting goals. One of them is that the system must satisfy the quality of service requirements of individual applications, which are typically defined in service-level agreements (SLAs). For interactive applications, a common example of an SLA is one that is based on the response time performance. The other goal is to provide efficient operation by using as few resources as possible. This corresponds to minimizing the provisioning cost.

In this thesis, we use the following condition to reflect the SLA:

$$\text{Prob [response time} \leq t] \geq p \tag{3.1}$$

where t is a parameter representing the response time objective and the p is a parameter specifying a lower bound for the probability of achieving response time t. Both parameters are specified based on the application's performance goal.

To avoid degradation in response time performance, the system places, for each application, a limit on the number of logon users. Suppose this limit is M. This means that at any time instant, the number of logon users cannot exceed M. This also means that

a request to establish a new session will be rejected if the number of logon users is already at the maximum *M*.

3.2 Dynamic resource allocation

The objective of dynamic resource allocation is to use the minimum number of server nodes to achieve the service level agreements of individual applications. For our system, resource allocation decisions are made at decision points. Such decisions may result in a change to the number of server nodes allocated to a given application. An example is shown in Figure 3.1, where the number of server nodes allocated is increased from 1 to 2 at decision point 4. The time interval between two successive decision points is referred to as an operation interval. The number of server nodes allocated is not changed during the operation interval. For example, in Figure 3.1, 2 server nodes are allocated in the operation interval between decision points 1 and 2.

At each decision point, the resource allocation algorithm makes use of data collected at the Measure phase as well as historical data to determine the number of server nodes that should be allocated for the next operation interval. Examples of measurement data are as follows:

- Workload Data: this includes arrival rate of requests for establishment of new sessions and service time of interactive jobs
- State Information: this includes the number of interactive jobs in queue and in service and the number of logon users.
- Performance Information: for our investigation, the key performance metric is the response time distribution.

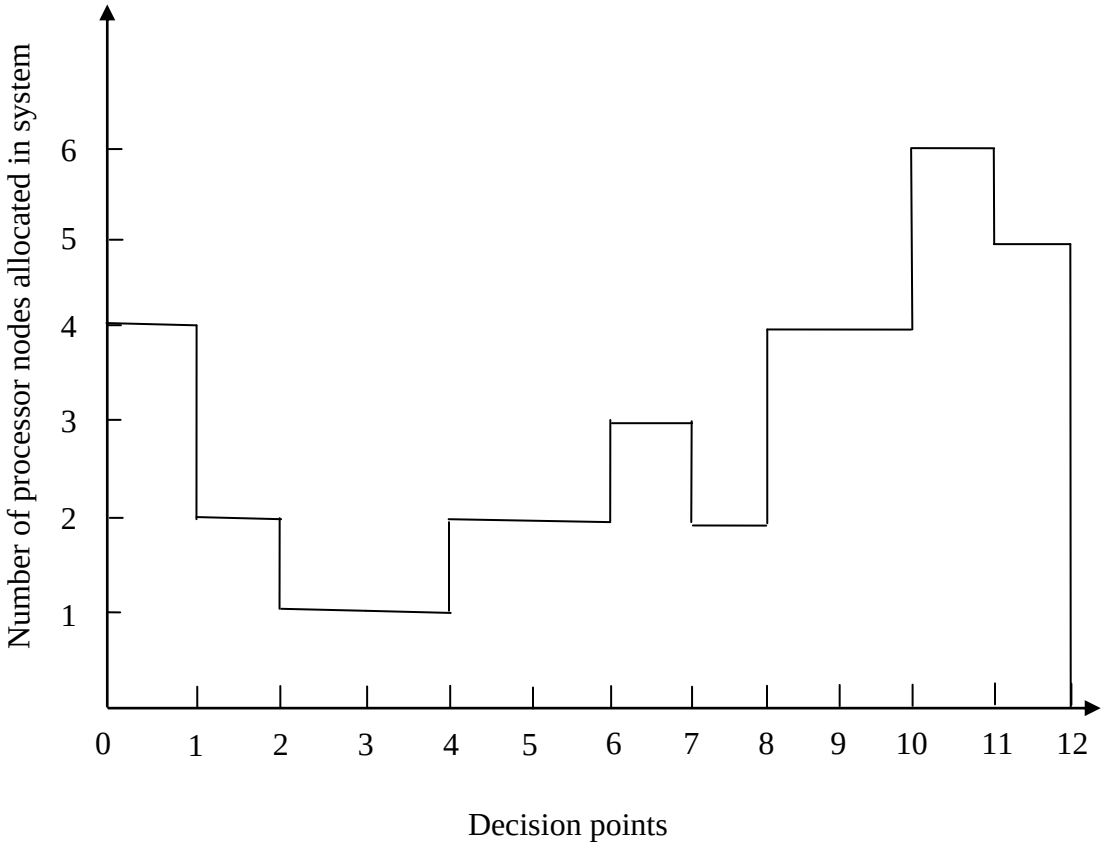

Figure 3.1 Number of processor nodes allocated in system

3.3 Example system

As an example, we describe in this subsection a web-based client-server system. The client-server model is a popular model for network computing. This model, as shown in Figure 3.2, describes the relationship between two types of hosts, clients and servers. Each instance of the client usually runs on a personal computer; it forwards requests to one or more connected servers. The servers, upon receiving these requests, process them and return the corresponding responses to the client. Most business applications today are

written using this model. An example of client-server applications is Amazon.com, which is an online retailer e-commerce company with products such as books, DVD and music CDs. The company's computing infrastructure consists of many server clusters that are geographically distributed and connected via wired networks. Both buyers and sellers use web browsers to access web pages stored in servers or to submit requests (e.g., order books) to the servers.

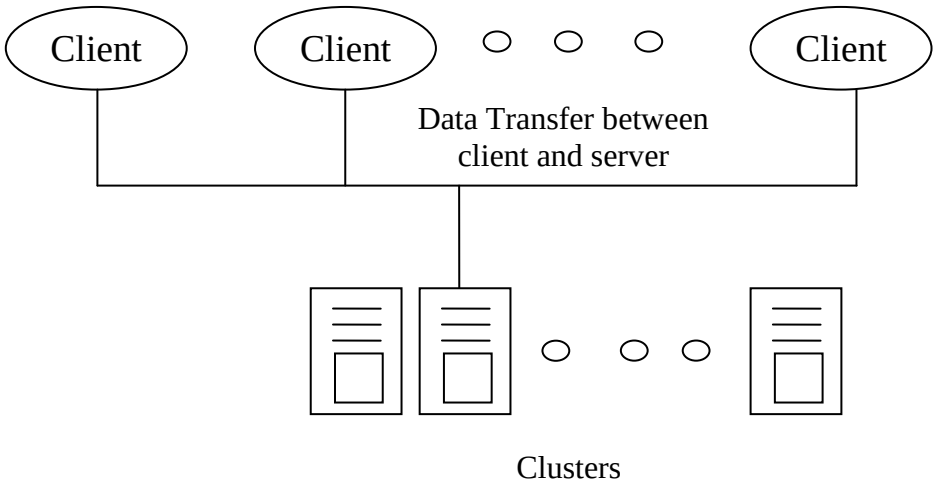

Figure 3.2: Client-server architecture

An important requirement of e-commerce systems is to provide users with a real-time response to access information stored in servers. For example, the online trading server has to notify all buyers and sellers immediately if the price of the products has been changed. Factors that affect response time performance include the number of online users, the number of servers and the internet latency. In our investigation, each client can be viewed as a logon user. When a new user logs onto the system, a new session is created. A logon user may submit requests to view web pages or

update information stored on servers. These requests correspond to jobs to be processed by the processor nodes in the server clusters. When the user decides to log off, the corresponding session is terminated. To avoid degradation in response time performance, the system typically places a limit on the number of logon users.

Chapter 4

Performance model

In this section, we develop a performance model for the interactive system described in Chapter 3. Results from this model will be used to produce processor node provisioning recommendations. The scope of our investigation is restricted to a cluster that hosts one interactive application only.

Our performance model is depicted in Figure 4.1. There are k logon users who interact with a system that consists of m identical processor nodes. A session has been created for each logon user. Since new sessions may be created and existing sessions may be terminated, the number of logon users may change over time. Specifically, the number of logon users is increased by 1 when a new session is created and decreased by 1 when an existing session is terminated. To avoid degradation in response time performance, the system places a limit M on the number of logon users. Arrival of requests for the establishment of a new session is modeled by a Poisson process with arrival rate λ. For

each arriving request, if the number of logon users is already at the maximum *M*, the arriving request is rejected.

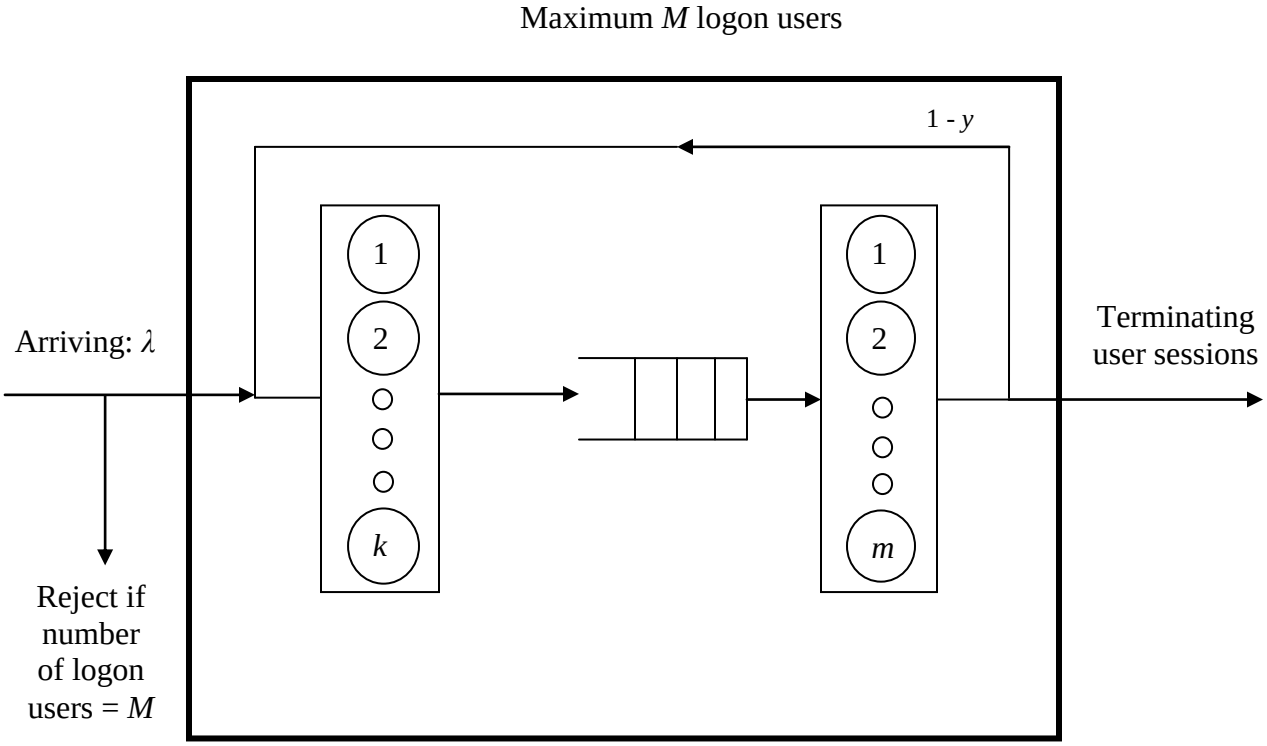

Figure 4.1: Performance model

The sub-model inside the box in Figure 4.1 is similar to a finite population model (or an M/M/m/∞/k model), where *k* logon users at their workstations interact with a service facility containing *m* servers. A logon user, after spending a think time at the user terminal, submits a job to the system. The think times are assumed to be independent and exponentially distributed with a mean of 1/*g*. Jobs submitted by logon users join a single queue; the queuing discipline is First-Come-First-Serve (FCFS). There are *m* identical servers, each of which represents a processor node. The service times of interactive jobs are also independent and exponentially distributed with a mean of $1/\mu$.

When a job completes service at a server, it has probability 1-*y* of returning to the user terminal, indicating that the logon user will continue with his session. On the other hand, there is a probability *y* of leaving the sub-model inside the box; this represents the termination of a user session. The overall performance model can be viewed as a two-level model. The lower level is concerned with client-server interactions involving the logon users, and the higher level is concerned with the establishment of new sessions and the termination of existing sessions.

The user behavior at the higher level model and lower level model during a session is shown in Figure 4.2. This user enters the system at session creation time. He then goes through repeated states of think times and waiting for system responses. After the last job has been processed, the user leaves the system and the corresponding session is terminated.

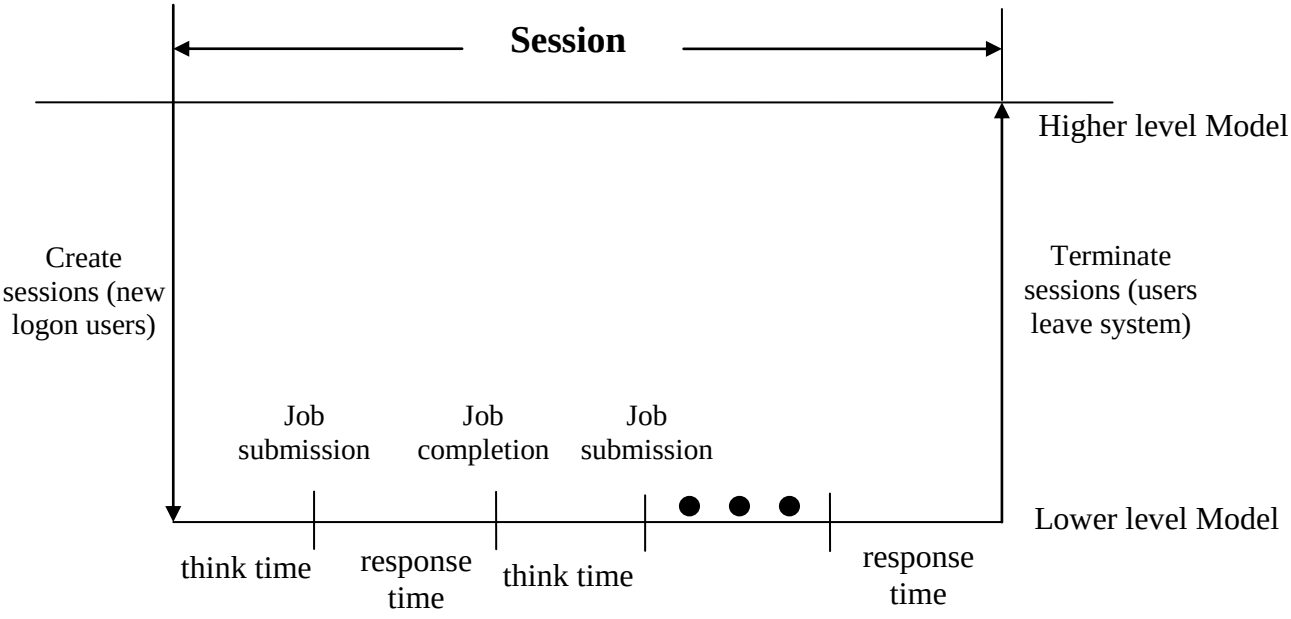

Figure 4.2: User behavior between higher and lower level models

Chapter 5

Analytic results

In this chapter, we derive analytic results for the response time distribution for the performance model shown in Figure 4.1. For this model, response time is defined to be the elapsed time from when a logon user submits a job to when a response is received by this user. Our results will be used as input to the investigation of dynamic resource provisioning in Chapter 6.

Our model, as described in Figure 4.1, is a Markov chain. A state of this model is defined as (n_1, n_2) where n_1 presents the number of logon users who are in the thinking state and n_2 presents the number of jobs in the system. It follows that $n_1 + n_2$ is the total number of logon users. Since the number of logon users cannot exceed M, a feasible state is characterized by $n_1 + n_2 \leq M$.

Based on the model description in Chapter 4, the state transition diagram for our model is shown in Figure 5.1. In this diagram, the state transitions shown in a given row, say the role where $n_1 + n_2 = k$, correspond to state transitions for the M/M/m/∞/k model. The total number of states is $NS = (M + 1)(M+2) / 2$.

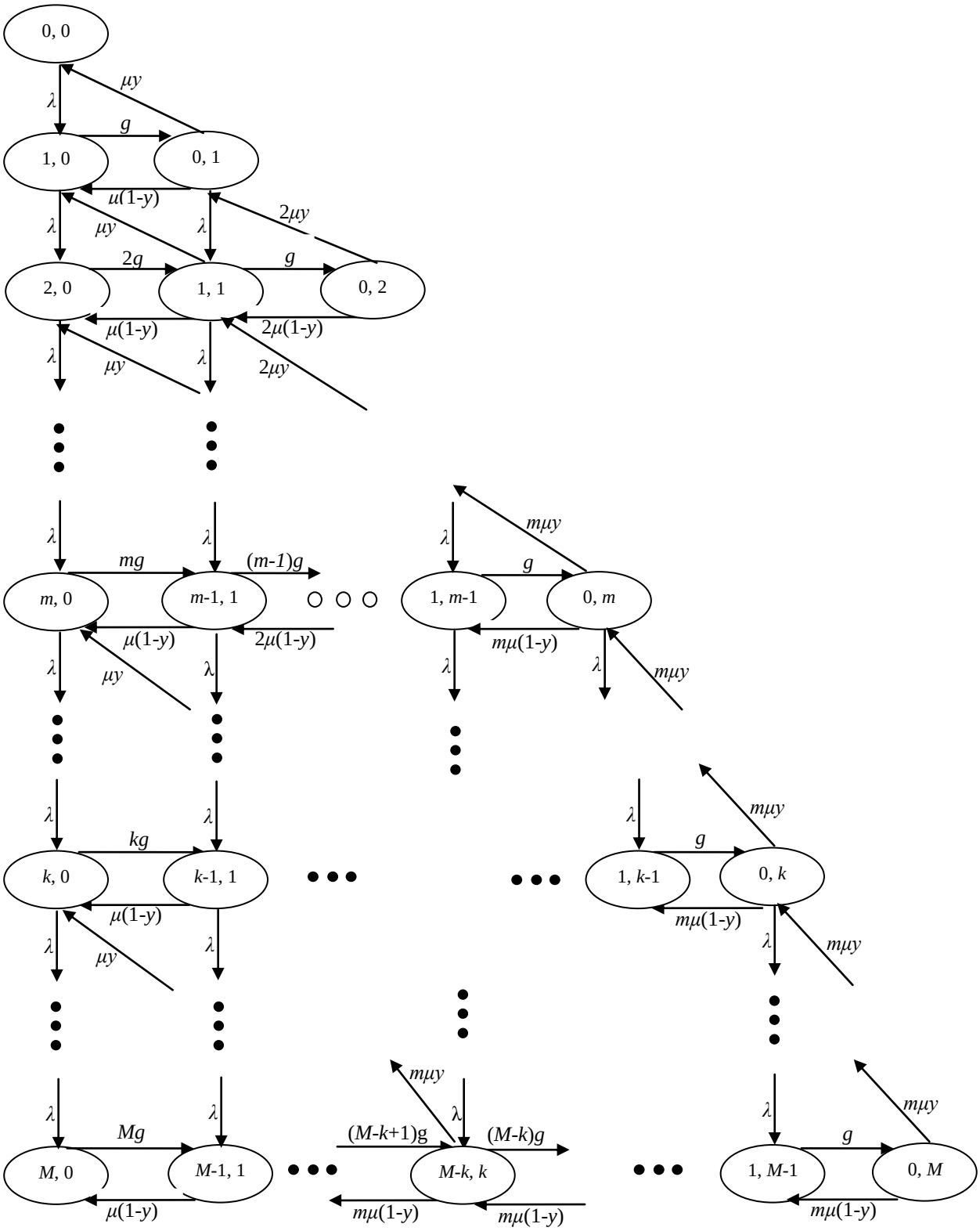

Figure 5.1: State transition diagram

Let $p_{n1,n2}$ be the probability that the system is in state (n_1, n_2) at steady state. From the state transition diagram shown in Figure 5.1, one can obtain the balance equations and solve for the steady state probabilities. These equations do not have a product form solution [15], so an efficient solution method for our model does not appear to be available. Nevertheless, one can obtain a solution to the steady state probabilities by solving the balance equations directly. This involves solving a set of NS linear equations. However, the number of states (or the number of equations) is $O(M^2)$ and solution for large M may not be easy to obtain. Our approach is to develop an efficient approximation method where the number of equations is reduced to $O(M)$. The accuracy of our approximation method will be evaluated by comparison with exact numerical results.

5.1 Approximate analysis

Our approximation method is based on the use of a two-level hierarchical model. The lower level is concerned with the processing of jobs submitted by logon users. The higher level, on the other hand, is concerned with the establishment of new sessions and the termination of existing sessions. In this section, we first obtain analytic results for the steady state probabilities and response time distribution for the lower level model. These results will then be used at the higher level model to obtain the response time distribution for the overall model.

5.1.1 Lower level model

The lower level model is the M/M/m/∞/k model where *k* is the number of logon users. See Figure 5.2. As mentioned in Chapter 4, the think time follows an exponential distribution with mean $1/g$ and the service time is exponentially distributed with mean $1/\mu$.

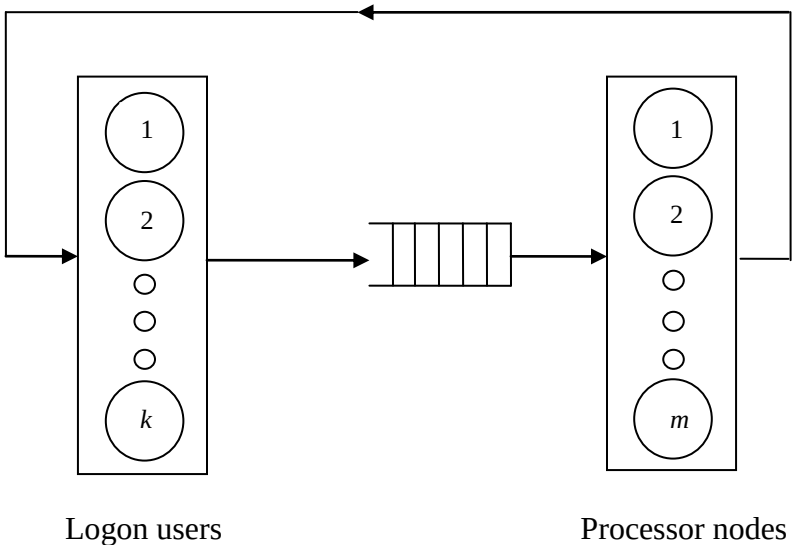

Figure 5.2: Lower level model

Sekino has obtained analytic results for the response time distribution for the M/M/m/∞/k model with the first-come first-served (FCFS) discipline [16, 17]. These results are summarized below.

Let p_n be the steady-state probability that the number of users in the system is *n*. From analytic results for the birth-death model, p_n can be written as:

$$p_n = \begin{cases} \rho^n \binom{k}{n} p_0 & 0 \le n \le m \\ \rho^n \frac{k!}{m! m^{n-m} (k-n)!} p_0 & m < n \le k \end{cases} \quad (5.1)$$

where $\rho = \frac{g}{\mu}$, and

$$p_0 = \left[\sum_{n=0}^{m-1} \rho^n \binom{k}{n} + \sum_{n=m}^{k} \rho^n \frac{k!}{m! m^{n-m} (k-n)!} \right]^{-1} \quad (5.2)$$

Let A_n be the steady state probability that a job submitted by a logon user finds n other jobs in the system. A_n is given by:

$$A_n = \frac{(k-n) g p_n}{\sum_{j=0}^{k} (k-j) g p_j} = \frac{k-n}{k-\overline{Q}} p_n \quad (5.3)$$

where $\overline{Q} = \sum_{j=0}^{k} j \, p_j$ is the mean number of jobs in the system.

We now present the results for the response time distribution under FCFS [16, 17]. Let $f(t)$ be the probability density function of the response time and $P^*(s)$ be its Laplace transform. We can write $P^*(s)$ as follows:

$$P^*(s) = \sum_{n=0}^{k-1} A(n) P^*(s|n) \quad (5.4)$$

where $P^*(s|n)$ is $P^*(s)$ conditioned on a job submitted by a logon user finding n other jobs in the system. Since the service time distribution is exponential and the queuing discipline is FCFS, we can get the following:

$$P^*(s|n) = \begin{cases} \frac{\mu}{s+\mu} & 0 \le n < m \\ \left(\frac{m\mu}{s+m\mu} \right)^{n-m+1} \left(\frac{\mu}{s+\mu} \right) & m \le n < k \end{cases} \quad (5.5)$$

Removing the condition on *n* and after simplification, we get [16, 17]:

$$P^*(s) = \left(\frac{\mu}{s+\mu}\right)\left[\sum_{n=0}^{m-1}\frac{k-n}{k-\bar{Q}}\rho^n\binom{k}{n}p_0 + \sum_{n=m}^{k-1}\frac{k-n}{k-\bar{Q}}\rho^n\frac{k!}{m!m^{n-m}(k-n)!}p_0\left(\frac{m\mu}{s+m\mu}\right)^{n-m+1}\right] \quad (5.6)$$

This can be inverted to obtain the probability density function of response time *f*(*t*). The result is as follows:

$$f(t) = \frac{\mu e^{-\mu t}}{k-\bar{Q}}p_0\sum_{n=0}^{m-1}\left[\binom{k}{n}(k-n)\rho^n\right] + \frac{m\mu}{k-\bar{Q}}\left(\frac{k!}{m!}\right)\rho^m p_0\sum_{n=0}^{k-m-1}\left[\frac{\rho^n}{(k-m-1-n)!(m-1)^{n+1}}\right.$$
$$\left.(e^{-\mu t} - e^{-m\mu t}\sum_{j=0}^{n}\frac{[(m-1)\mu t]^j}{j!})\right] \quad (5.7)$$

The corresponding cumulative distribution function $F(t) = P\{T \leq t\}$ can be obtained by integrating the *f*(*t*) given by Equation 5.7. *F*(*t*) is given by:

$$F(t) = \frac{1-e^{-\mu t}}{k-\bar{Q}}p_0\left[\sum_{n=0}^{m-1}\binom{k}{n}(k-n)\rho^n\right] + \left[\frac{mk!\rho^m p_0}{(k-\bar{Q})m!}\right]\sum_{n=0}^{k-m-1}\left\{\left[\frac{\rho^n}{(k-m-1-n)!m^{n+1}}(1-e^{-m\mu t}\sum_{j=0}^{n}\frac{[m\mu t]^j}{j!})\right]\right.$$
$$\left.-\left[\frac{\rho^n e^{-\mu t}}{(k-m-1-n)!(m-1)^{n+1}}(1-e^{-(m-1)\mu t}\sum_{j=0}^{n}\frac{[(m-1)\mu t]^j}{j!})\right]\right\} \quad (5.8)$$

For the special case of a single server (or $m = 1$), we have the following simplified equations for *f*(*t*) and *F*(*t*):

$$f(t) = \frac{\mu e^{-\mu t}(1+gt)^{k-1}}{\sum_{n=0}^{k-1}\frac{(k-1)!\rho^n}{(k-n-1)!}} \quad (5.9)$$

$$F(t) = 1 - e^{-\mu t}\left\{\frac{\sum_{n=0}^{k-1}\left[\frac{\rho^n}{(k-n-1)!}\sum_{j=0}^{n}\frac{[\mu t]^j}{j!}\right]}{\sum_{n=0}^{k-1}\frac{\rho^n}{(k-n-1)!}}\right\} \quad (5.10)$$

5.1.2 Higher level model

The main idea of our approximate analysis is to approximate the model shown in Figure 4.1 by a birth-death model at the higher level. Our higher level model is shown in Figure 5.3. As mentioned in Chapter 4, requests for the establishment of a new session are modeled by a Poisson process with rate λ. For each arriving request, if the number of logon users is already at the maximum *M*, the arriving request is rejected.

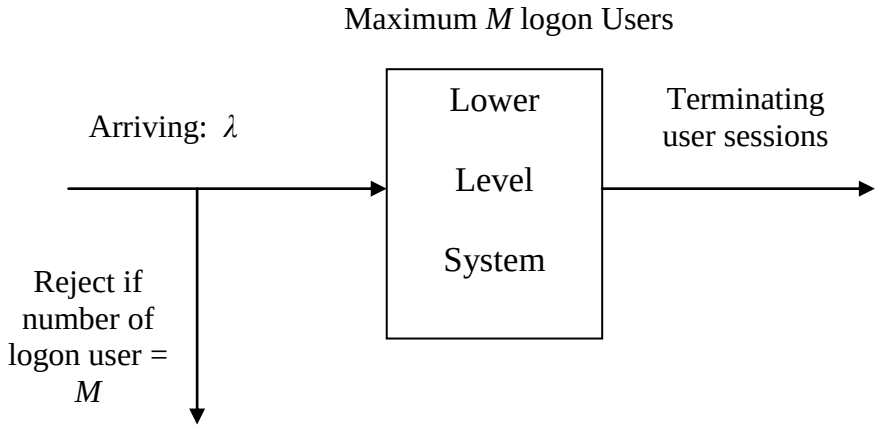

Figure 5.3: Higher level model

Departures from the higher level model correspond to the termination of user sessions. The state of the higher level model is therefore the number of logon users. Let η_k be the service rate when the number of logon users is *k*. The state transition diagram is shown in Figure 5.4. In our analysis, η_k is estimated as follows. At the lower level model

when a job completes service at a server, it has probability *y* of terminating the corresponding user session (see Figure 4.1). This would result in a departure from the higher level model. Let X_k be the throughput (or rate of job completion) at the lower level model when the number of logon users is *k*. X_k can be written as:

$$X_k = \sum_{n=0}^{k-1} p_n (k-n) g \tag{5.11}$$

Since each job completion has probability *y* of ending a session, η_k is estimated by:

$$\eta_k = X_k y \tag{5.12}$$

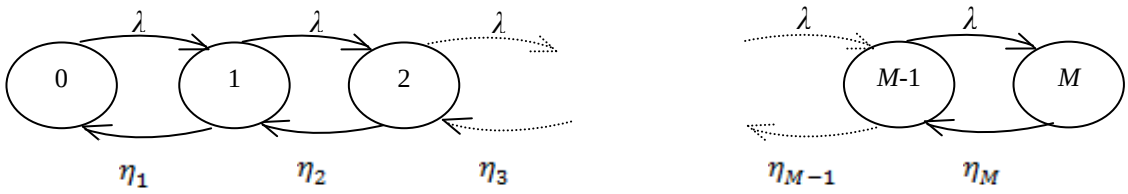

Figure 5.4: Birth-death model of higher level model

We now derive the response time distribution using our high level model. Let q_k be the steady-state probability that there are *k* logon users. From analytic results for the birth-death model, q_k can be written as:

$$q_k = q_0 \prod_{i=0}^{k-1} \frac{\lambda}{\eta_{i+1}} \tag{5.13}$$

and

$$q_0 = \left[\sum_{k=0}^{M} \prod_{i=0}^{k-1} \frac{\lambda}{\eta_{i+1}} \right]^{-1} \tag{5.14}$$

At the lower level model, let $F_k(t)$ be the CDF of response time when the number of logon users is k. $F_k(t)$ can be obtained using Equations (5.8) and (5.10) with k being the number of logon users. Also let $G(t)$ be the CDF of the response time at the higher level model. $G(t)$ corresponds to the response time distribution of the overall system. Our approximation takes into consideration the fraction of time that the number of logon users is k and the response time distribution given that there are k users at the lower level model. We thus have the following results:

$$G(t) = \frac{\sum_{k=1}^{M} q_k F_k(t)}{1-q_0} \tag{5.15}$$

5.2 Accuracy of approximate analysis

In this section, we evaluate the accuracy of our approximation method using numerical examples.

5.2.1 Numerical method to obtain exact results

One can certainly obtain exact analytic results by solving the balance equations for the model shown in Figure 4.1. These results are for the steady state probabilities. In our investigation, we use a different solution approach, namely to compute numerically the state probabilities over an extended time period. This would allow us to obtain results during the transient phase and at steady state. The steady state results will be used to evaluate the accuracy of our approximate analysis while the transient results will be

considered in the next chapter when we investigate the performance of dynamic resource provisioning based on our results.

Our numerical method can be described as follows. Let

$$p_{n_1,n_2}(t) = Prob[system\ state\ at\ time\ t = (n_1, n_2)] \tag{5.16}$$

Consider the state changes from time t to $t+\Delta t$. The state probability at $t+\Delta t$ can be written as [18]:

$$p_{0,0}(t+\Delta t) = p_{0,1}(t)\mu y\Delta t + p_{0,0}(t)(1-\lambda\Delta t) + o(\Delta t)$$

$$\vdots$$

$$p_{m,0}(t+\Delta t) = p_{m-1,0}(t)\lambda\Delta t + p_{m-1,1}(t)\mu(1-y)\Delta t + p_{m,1}(t)\mu y\Delta t$$
$$+p_{m,0}(t)(1-mg\Delta t-\lambda\Delta t) + o(\Delta t)$$

$$\vdots$$

$$p_{M,0}(t+\Delta t) = p_{M-1,0}(t)\lambda\Delta t + p_{M-1,1}(t)\mu(1-y)\Delta t$$
$$+p_{M,0}(t)(1-Mg\Delta t) + o(\Delta t)$$

$$\vdots$$

$$p_{M-k,k}(t+\Delta t) = p_{M-k-1,k}(t)\lambda\Delta t + p_{M-k+1,k-1}(t)(M-k+1)g\Delta t$$
$$+p_{M-k-1,k+1}(t)m\mu(1-y)\Delta t + p_{M-k,k}(t)$$
$$(1-m\mu\Delta t-(M-k)g\Delta t) + o(\Delta t)$$

$$\vdots$$

$$p_{0,M}(t+\Delta t) = p_{1,M-1}(t)g\Delta t + p_{0,M}(t)(1-m\mu\Delta t) + o(\Delta t)$$

$$\vdots$$

$$p_{0,m}(t+\Delta t) = p_{1,m-1}(t)\mathrm{g}\Delta\mathrm{t} + p_{0,m+1}(t)m\mu y\Delta t + p_{0,m}(t)(1 - m\mu\Delta t - \lambda\Delta t) + \mathrm{o}(\Delta\mathrm{t})$$

$$\vdots$$

$$p_{n_1,n_2}(t+\Delta t) = p_{n_1-1,n_2}(t)\lambda\Delta\mathrm{t} + p_{n_1+1,n_2-1}(t)(\mathrm{n}_1+1)\mathrm{g}\Delta\mathrm{t} + p_{n_1-1,n_2+1}(t)(n_2+1)\mu(1-y)\Delta\mathrm{t} + p_{n_1,n_2+1}(t)(n_2+1)\mu y\Delta t + p_{n_1,n_2}(t)(1 - n_2\mu\Delta t - n_1 g\Delta t - \lambda\Delta\mathrm{t}) + \mathrm{o}(\Delta\mathrm{t})$$

(5.17)

where $o(\Delta t)$ has the property that $\lim_{\Delta t\to 0}[o(\Delta t)/\Delta t] = 0$. When Δt is sufficiently small, the term $o(\Delta t)$ in Equation (5.17) can be ignored without introducing inaccuracies in the results. Therefore, $o(\Delta t)$ will not be included in our computation.

Our computation method is as follows. Starting with a given initial condition, namely the state of the system at time 0 ($t = 0$), we compute $p_{n1,n2}(t + \Delta t)$ repeatedly using Equation (5.17) for all feasible states (n_1,n_2). An example of an initial condition is $p_{0,0}(0) = 1$, which corresponds to the case of no logon users at time zero. Our method will allow us to compute $p_{n1,n2}(t)$ for $t = \Delta t, 2\Delta t, \ldots, L\Delta t, \ldots$. The results would initially show state probabilities during the transient phase, and for sufficiently large L, the steady state probabilities.

In our method, we determine the value of Δt using the following steps:

1. Start with an small value of Δt
2. Compute $p_{n1,n2}(t)$ for all feasible states (n_1,n_2). for $t = \Delta t, 2\Delta t, \ldots, L\Delta t, \ldots$
3. Repeat step 2 using a value of Δt which is half of its previous value.

4. Δt is sufficiently small if step 3 yields results for $p_{n1,n2}(t)$ that have less than 0.1% difference from those obtained previously, otherwise repeat steps 3.

Once a sufficiently small Δt is found, steady state can be reached at time $L\Delta t$ if the difference for $p_{n1,n2}(t)$ between $t = L\Delta t$ and $t = (L+1)\Delta t$ is less than 0.001% for all feasible states (n_1,n_2). For our model with $g = 0.1$, $\mu = 1$ and $y = 0.05$, it is found that $\Delta t = 0.001$ is sufficiently small. Therefore, $\Delta t = 0.001$ will be used in all our numerical examples.

To illustrate our computational method, consider an example with the following parameter values:

$g = 0.1$; $\mu = 1$; $y = 0.1$; $M = 3$; $m = 1$; $\lambda = 0.5$

In Figure 5.5, the state probabilities $p_{n1,n2}(t)$ for all feasible states (n_1,n_2) are plotted against t when the initial condition is $p_{0,0}(0) = 1$. We observe that the values of all state probabilities do not change after $t = 20$. This can also be used as an estimate of the length of the transient phase. The steady state probabilities are given in Table 5.1. These results show that $p_{3,0} = 0.712$, indicating that the system has 71.2% chance to have 3 logon users and all of them are in the think state. Also, the results shows that $p_{2,1} = 0.214$, which means that there is a 21.4% probability that there are 3 logon users and one of them is waiting for system response.

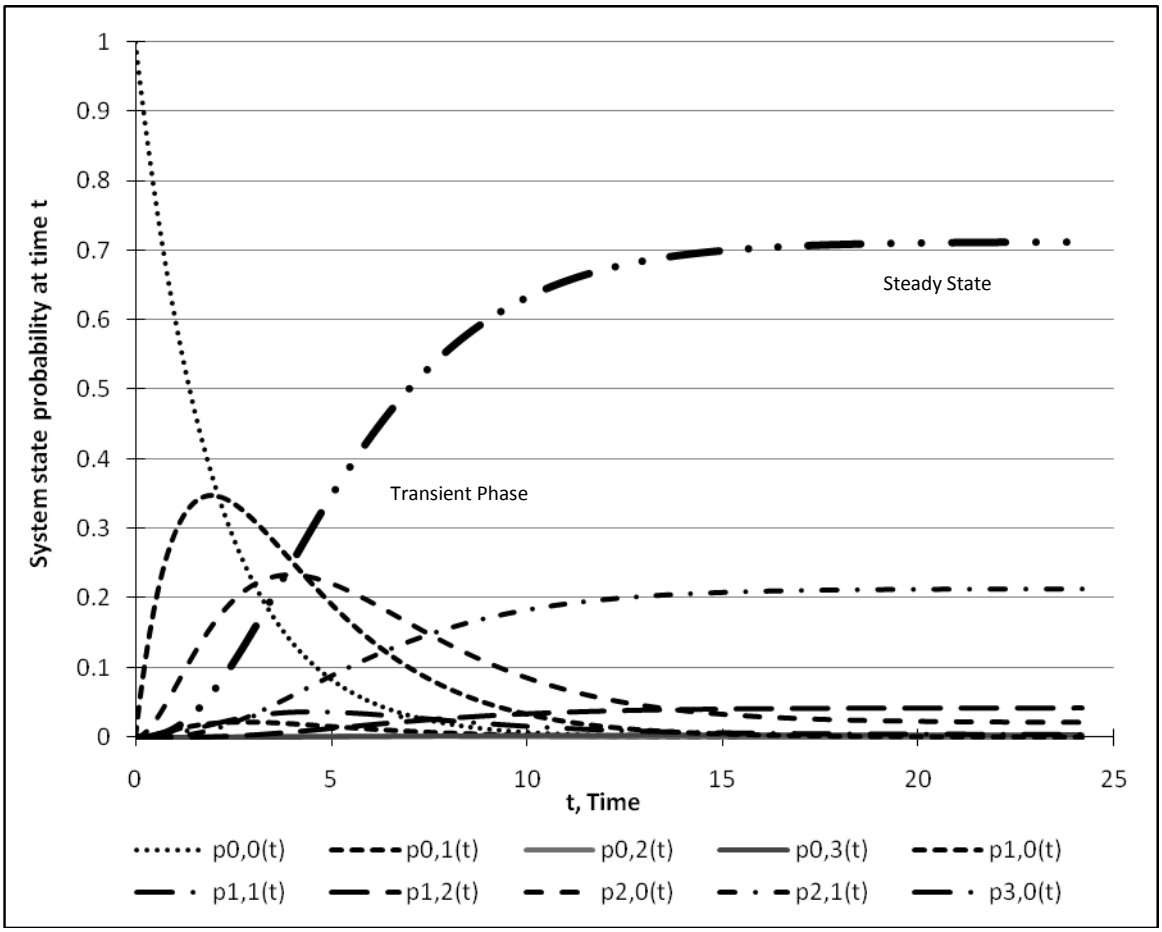

Figure 5.5: State probabilities diagram

State	（0，0）	（0，1）	（0，2）	（0，3）	（1，0）
Probability	0.000014	0.000052	0.000433	0.00427	0.000523
State	（1，1）	（1，2）	（2，0）	（2，1）	（3，0）
Probability	0.00434	0.0427	0.0217	0.214	0.712

Table 5.1: Steady state probability

5.2.2 Accuracy of approximate analysis

In this subsection, we evaluate the accuracy of our approximate analysis using numerical examples. Our approximate analysis yields results for q_k, the steady state probability that there are k logon users, k = 0, 1, …, M (see Equation (5.13)). From the results in subsection 5.2.1, we can determine the exact value for the steady state probability of having k logon users. This is given by:

$$e_k = \sum_{\substack{all\ states\ (n_1,n_2)\ s.t. \\ n_1+n_2=k}} (n_1 + n_2)\, P_{n1,n2} \qquad \text{for } k = 0, 1, \ldots, M. \qquad (5.18)$$

The accuracy of our approximate analysis is evaluated by comparing the values of q_k and e_k, k = 0, 1, …, M. The parameter values used in our evaluation are shown in Table 5.2. Sixteen cases are considered, with two values for y, four values for m, and 2 values for λ. M is selected such that the blocking probability at the higher level, as given by q_M in Equation (5.13) is less than 1%. This means that we are interested in the cases where the blocking probability is small, an important consideration when one considers providing good service to the users.

We show in Table 5.3 the results for q_k, e_k, and the percentage difference, given by $|(q_k - e_k)/e_k| * 100\%$, for the case y = 0.05, m = 3 and λ = 0.05. The value of M such that q_M < 1% is 20. The corresponding results for the mean number of logon users, given by $\bar{q} = \sum_{k=1}^{M} k q_k$ and $\bar{e} = \sum_{k=1}^{M} k e_k$, are also shown. We observe that the percentage difference is small for those states k where e_k > 1%; the maximum difference is about 10.9%. For the other states, the percentage difference could be as high as 16.7%. However, the state probabilities for these other states are small (less than 1%). As a result,

the inaccuracies in these state probabilities will not have a significant impact on the accuracy of the approximation method in terms of performance metrics seen at the higher level. For example, the percentage difference for the mean number of logon users at steady state is about 1.5% only. This percentage difference is given by:

$$B = |\bar{q} - \bar{e}|/\bar{e} * 100\% \tag{5.19}$$

We next present in Table 5.4 the results for the case of $y = 0.05$, $m = 3$ and $\lambda = 0.1$ (the value of M for this case is 40). Compared to the results in Table 5.3, the percentage difference is much higher. In fact, for this case, the approximate analysis is least accurate among all the cases considered. Specifically, for those states with $e_k > 1\%$, the maximum difference is about 34.5%, and for the other states, the difference can be as high as 72%. This is due to the larger value of M which requires a larger number of state probabilities to be estimated, and some of the state probabilities take on very small values. For these state probabilities, even a small difference between q_k and e_k can produce a large percentage difference because of the small value of the state probability in the denominator. Nevertheless, the approximation method yields accurate results for the mean number of logon users at steady state; the percentage difference, as given by B in Equation (5.19) is only 3.35%.

As our third example, we consider the case of $y = 0.1$, $m = 10$, $\lambda = 0.1$ (the value of M is 20). This corresponds to a scenario where the number of servers is larger than those considered in the first two examples. The results are shown in Table 5.5. We observe that the approximation method yields accurate results for all state probabilities. The largest difference is when the state $k = 1$; the difference is about 7.5%. The

difference in terms of the mean number of logon users is less than 1%.

To further evaluate the accuracy of our approximation method, we consider the 16 combinations of parameter values shown in Table 5.2. A summary of our results is present in Table 5.6 where we show the average percentage difference of state probabilities that are larger than 1%. Specifically, suppose S is the set of states k with $e_k >$ 1%. The average percentage difference is given by:

$$A = \frac{\sum_{k \epsilon S}(|e_k - q_k|/e_k)}{|S|} * 100\% \quad (5.20)$$

We also show in Table 5.6 the percentage difference for the mean number of logon users at steady state, given by B in Equation (5.19). We observe from the results in Table 5.6 that in terms of the state probabilities, our approximation method is very accurate for most cases (A is less than 10%). For the three cases where A >10%, the largest value for A is about 15.9%. For this case, a detailed comparison between e_k and q_k has been presented in Table 5.4. For the other two cases, A is approximately 13% and 12%, and the corresponding comparisons between e_k and q_k are shown in Tables 5.7 and 5.8. The reason for the high values of A was explained previously when we discussed the result in Table 5.4. As to the mean number of logon users, the approximation is very accurate; the largest percentage difference for B is 3.3%.

In this Chapter, we have evaluated the accuracy of our approximation method based on its ability to estimate the state probabilities at the higher level. In dynamic resource provisioning, other performance metrics are of interest, e.g., the response time distribution during an operation interval, and the smallest number of processor nodes

required to meet a given SLA. The accuracy of our approximation method in the context of these other performance metrics will be considered in the next Chapter.

g	0.1
μ	1
y	0.05, 0.1
m	2, 3, 4, 10
λ	0.05, 0.1

Table 5.2: Input parameter values

k	Approximate q_k	Exact e_k	% difference
0	0.000017	0.00002	15.0
1	0.000183	0.000214	14.5
2	0.001	0.0012	16.7
3	0.00368	0.00418	12.0
4	0.0101	0.0113	10.6
5	0.0223	0.0245	9.0
6	0.0408	0.0442	7.7
7	0.0642	0.0683	6.0
8	0.0884	0.0925	4.4
9	0.108	0.111	2.7
10	0.119	0.121	1.7
11	0.119	0.119	0.0
12	0.11	0.108	1.9
13	0.0933	0.0901	3.6
14	0.0737	0.0701	5.1
15	0.0544	0.051	6.7
16	0.0377	0.0348	8.3
17	0.0246	0.0224	9.8
18	0.0152	0.0137	10.9
19	0.00895	0.00796	12.4
20	0.00501	0.00443	13.1
Mean no. of logon users	10.988	10.822	1.5

Table 5.3: Approximate vs Exact Results for $y = 0.05$, $m = 3$, $\lambda = 0.05$, and $M = 20$

k	Approximate q_k	Exact e_k	% difference
0 to 3	0	0	0.0
4	0.000006	0.00001	40.0
5	0.00001	0.000016	37.5
6	0.000036	0.000056	35.7
7	0.000114	0.000172	33.7
8	0.000313	0.000461	32.1
9	0.000765	0.0011	30.5
10	0.00169	0.00237	28.7
11	0.00338	0.00462	26.8
12	0.00622	0.00829	25.0
13	0.0106	0.0137	22.6
14	0.0167	0.0211	20.9
15	0.0246	0.0304	19.1
16	0.0342	0.041	16.6
17	0.0446	0.0522	14.6
18	0.0552	0.0629	12.2
19	0.0649	0.0719	9.7
20	0.0726	0.0783	7.3
21	0.0776	0.0814	4.7
22	0.0794	0.081	2.0
23	0.0781	0.0774	0.9
24	0.0739	0.0711	3.9
25	0.0675	0.063	7.1
26	0.0596	0.054	10.4
27	0.0509	0.0448	13.6
28	0.0423	0.036	17.5
29	0.0342	0.0282	21.3
30	0.0269	0.0215	25.1
31	0.0207	0.016	29.4
32	0.0156	0.0116	34.5
33	0.0115	0.00828	38.9
34	0.00837	0.0058	44.3
35	0.00599	0.00401	49.4
36	0.00422	0.00273	54.6
37	0.00295	0.00184	60.3
38	0.00204	0.00123	65.9
39	0.00139	0.000824	68.7
40	0.000949	0.000552	71.9
Mean no. of logon users	22.85	22.11	3.3

Table 5.4: Approximate vs Exact Results for $y = 0.05$, $m = 3$, $\lambda = 0.1$, and $M = 40$

k	Approximate q_k	Exact e_k	% difference
0	0.000017	0.000018	5.6
1	0.000185	0.0002	7.5
2	0.00102	0.00109	6.4
3	0.00372	0.00397	6.3
4	0.0102	0.0108	5.6
5	0.0225	0.0236	4.7
6	0.0413	0.0429	3.7
7	0.0649	0.0669	3.0
8	0.0892	0.0913	2.3
9	0.109	0.111	1.8
10	0.12	0.121	0.8
11	0.12	0.12	0.0
12	0.119	0.118	0.8
13	0.093	0.0915	1.6
14	0.0731	0.0713	2.5
15	0.0536	0.0519	3.3
16	0.0369	0.0354	4.2
17	0.0239	0.0228	4.8
18	0.0146	0.0138	5.8
19	0.00844	0.00797	5.9
20	0.00464	0.00437	6.2
Mean no. of logon users	10.949	10.868	0.9

Table 5.5: Approximate vs Exact Results for $y = 0.1$, $m = 10$, $\lambda = 0.1$, and $M = 20$

λ	m	y	A	B
0.05	2	0.05	6.5	1.7
0.1	2	0.05	3.5	0.4
0.05	3	0.05	5.9	1.5
0.1	3	0.05	15.9	3.3
0.05	4	0.05	5.9	1.5
0.1	4	0.05	13.0	2.6
0.05	10	0.05	5.8	1.5
0.1	10	0.05	11.8	2.5
0.05	2	0.1	1.4	0.5
0.1	2	0.1	3.2	0.9
0.05	3	0.1	1.4	0.5
0.1	3	0.1	2.9	0.8
0.05	4	0.1	1.4	0.5
0.1	4	0.1	2.9	0.8
0.05	10	0.1	1.4	0.5
0.1	10	0.1	2.9	0.7

Table 5.6: Percentage difference A and B

k	Approximate q_k	Exact e_k	% difference
0 to 3	0	0	0.0
4	0.000003	0.000004	25.0
5	0.000012	0.000018	33.3
6	0.000042	0.000063	33.3
7	0.000133	0.000193	31.1
8	0.000366	0.000518	29.3
9	0.000895	0.00124	27.5
10	0.00197	0.00265	25.7
11	0.00394	0.00517	23.9
12	0.00722	0.00926	21.9
13	0.0122	0.0153	20.0
14	0.0192	0.0235	18.3
15	0.0282	0.0335	15.8
16	0.0389	0.0451	13.7
17	0.0504	0.0571	11.7
18	0.0618	0.0682	9.4
19	0.0717	0.0772	7.1
20	0.0791	0.0831	4.8
21	0.0833	0.0852	2.2
22	0.0837	0.0835	0.3
23	0.0805	0.0783	2.8
24	0.0743	0.0705	5.4
25	0.0659	0.061	8.0
26	0.0564	0.0508	11.0
27	0.0465	0.0408	13.9
28	0.0369	0.0316	16.8
29	0.0285	0.0237	20.3
30	0.0213	0.0173	23.1
31	0.0154	0.0122	26.2
32	0.0108	0.00833	29.7
33	0.00739	0.00555	33.2
34	0.00493	0.0036	36.9
35	0.0032	0.00228	40.4
36	0.00203	0.00141	44.0
37	0.00126	0.000852	47.9
38	0.000763	0.000505	51.1
39	0.000453	0.000295	53.6
40	0.000264	0.00017	55.3
Mean no. of logon users	22.167	21.598	2.635

Table 5.7: Approximate vs Exact Results for $y = 0.05$, $m = 4$, $\lambda = 0.1$, and $M = 40$

k	Approximate q_k	Exact e_k	% difference
0 to 3	0	0	0.0
4	0.000003	0.000004	25.0
5	0.000012	0.000018	33.3
6	0.000044	0.000065	32.3
7	0.000138	0.000198	30.3
8	0.000380	0.000532	28.6
9	0.000928	0.00127	26.9
10	0.00204	0.00272	25.0
11	0.00408	0.00532	23.3
12	0.00749	0.00951	21.2
13	0.0127	0.0157	19.1
14	0.0199	0.0241	17.4
15	0.0292	0.0345	15.3
16	0.0402	0.0462	13.0
17	0.052	0.0584	10.9
18	0.0635	0.0696	8.8
19	0.0735	0.0786	6.5
20	0.0809	0.0844	4.1
21	0.0847	0.0863	1.8
22	0.0847	0.0842	0.5
23	0.0811	0.0786	3.2
24	0.0743	0.0703	5.7
25	0.0654	0.0603	8.4
26	0.0553	0.0498	11.0
27	0.0451	0.0396	13.9
28	0.0354	0.0304	16.4
29	0.0269	0.0225	19.6
30	0.0197	0.0162	21.6
31	0.0139	0.0111	25.2
32	0.00962	0.00748	28.6
33	0.00641	0.00486	31.9
34	0.00415	0.00307	35.2
35	0.00261	0.00189	38.1
36	0.00159	0.00113	40.1
37	0.000947	0.000656	44.4
38	0.000548	0.000372	47.3
39	0.000309	0.000206	50.0
40	0.000170	0.000112	51.7
Mean no. of logon users	21.996	21.464	2.479

Table 5.8: Approximate vs Exact Results for $y = 0.05$, $m = 10$, $\lambda = 0.1$, and $M = 40$

Chapter 6

Dynamic resource provisioning

As mentioned in Section 3.2, the objective of dynamic resource allocation is to use the minimum number of server nodes to achieve the service level agreements of individual applications. Resource allocation decisions are made at decision points and the number of server nodes allocated is not changed during the operation interval (or the time interval between consecutive decision points). In this chapter, we evaluate the merit of our approximate analysis in terms of its adequacy in providing results that can be used in resource allocation decisions.

Our evaluation is based on dynamic resource allocation with the following features. Workload data and state information are used as input to resource allocation decisions. For our model, the workload parameters are:

- $1/g$, the mean think time;
- $1/\mu$, the mean service time:
- y, the probability of ending a session; and

- λ, the arrival rate of requests for the creation of new sessions.

The system state is given by (n_1,n_2) where n_1 is the number of logon user in the thinking state and n_2 is the number of the jobs in system. The algorithm produces as output the number of server nodes required in order to meet the SLA of the application given by Prob [response time $\leq t] \geq p$ (denoted by m_{new},).

In general, analytic results for m_{new} are difficult to obtain. The reason is follows. At a decision point, if there is a change in the number of server nodes allocated, the system will go through a transient phase before it exhibits the steady state behavior during the next operation interval. The length of the transient phase is affected by the system state at the decision point. To illustrate this point, consider our model with following parameters: $g = 0.1$; $\mu = 1$; $y = 0.1$; $M = 3$; $m = 1$; $\lambda = 0.5$. The transient behavior for initial condition $p_{0,0}(0) = 1$ (or state at a decision point $t = 0$ is (0,0)) has been shown in Figure 5.5. In Figure 6.1 and 6.2, we show further the values of $p_{3,0}(t)$ and $p_{2,1}(t)$ as a function of t for three different initial conditions, namely $p_{0,0}(0) = 1$, $p_{2,0}(0) = 1$ and $p_{1,2}(0) = 1$. We observe that the lengths of the transient phase for $p_{3,0}(t)$ and $p_{2,1}(t)$ are approximately 20 and 15 time units, respectively. We further observe that the length of the transient phase is affected by the initial conditions. Taking into consideration the transient behavior when determining m_{new} will lead to complexity in mathematical analysis because the impact of the transient behavior on Prob [response time $\leq t] \geq y$ is very difficult to characterize.

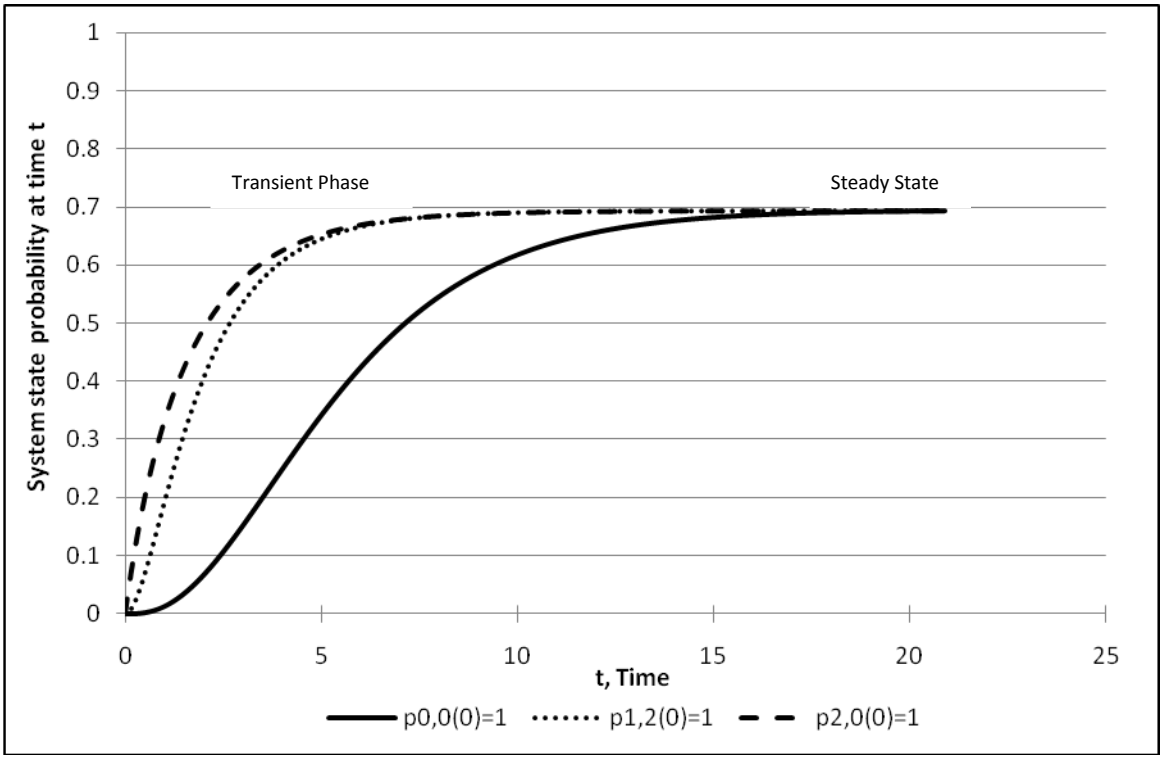

Figure 6.1: State probability diagram for $p_{3,0}(t)$

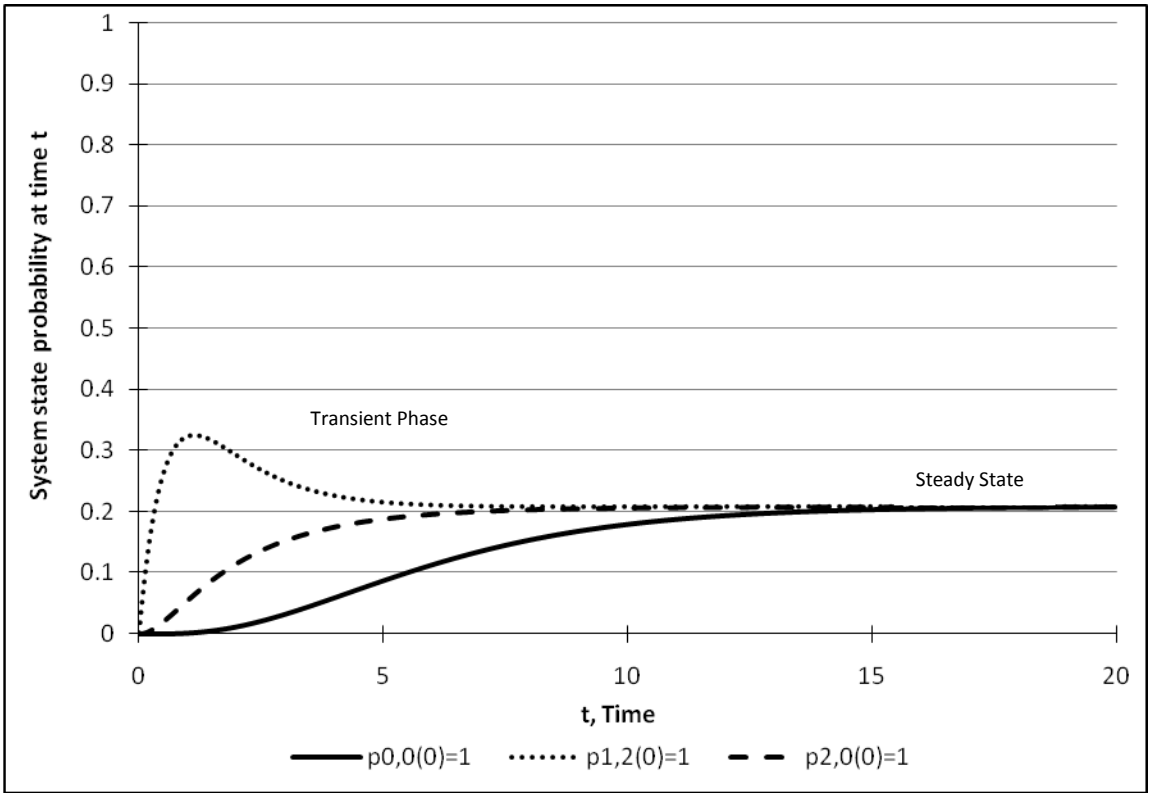

Figure 6.2: State probability diagram for $p_{2,1}(t)$

In dynamic resource allocation, there may be a change in the number of processor nodes at a decision point. If a change is made, the system will go through a transient phase before reaching steady state again. The transient behavior is affected by the initial condition, corresponding to the state of the system at the decision point.

Our approach to determining m_{new} is to use the steady state results for the response time distribution obtained in Chapter 5 as an approximation. This would exclude the transient behavior from the analysis and thus avoid the mathematical complexity mentioned above. It also follows that the state information at a decision point (or the initial condition) will not be necessary in our estimation of m_{new}. In this chapter, we will perform simulation experiments to determine whether our steady state results are adequate in estimating m_{new}, and discuss how these results may be used in dynamic resource allocation.

6.1 Accuracy of steady state results

The steady state result in Chapter 5 is for the response time distribution, as given by $G(t)$ in Equation (5.15). $G(t)$ represents the percentage of jobs that meet a response time goal t. In this section, we evaluate the accuracy of $G(t)$ when it is used as an approximation for the response time distribution between decision points where steady state may not exist at all times. Our evaluation is based on comparison with $G_E(t)$, the CDF of response time obtained by discrete event simulation.

The simulation is based on our model described in Chapter 4 with the additional assumption that first-come first-served (FCFS) scheduling is used at the server at the lower level model. The length of simulation is D, which is the length of the operation

interval (or the time between successive decision points). In dynamic resource allocation, the number of processor nodes allocated during D remains unchanged.

Consider the SLA defined in Equation (3.1), i.e., Prob [response time $\leq t$] $\geq p$. The parameter p is usually at least 80%. In our simulation experiments, 100 cases with $p > 80\%$ will be considered. These cases are selected such that p is evenly distributed between 80% and 99.8%. For each case, the follow steps are used:

1. Set input parameters g, μ and M to 0.1, 1.0 and 60, respectively.
2. Select input parameters λ, y, and m, and the response time goal t such that $G(t)$, as given by Equation (5.15) is larger than 0.8 and $q_M < 1\%$. The range values of λ, y, m, and t are {0.05, 0.1}, {0.05, 0.1}, {1, 10}, and {3.0, 6.0}, respectively.
3. Use the numerical method in Section 5.2.1 to determine T, the length of the transient phase for initial condition $p_{0,0}(0) = 1$. This initial condition normally results in the longest transient phase compared to other initial conditions.
4. Perform separate simulation experiments for three different values of D, namely $D = 0.5T$, $D = T$, and $D = 5T$. For each experiment, run the same simulation 50 times, each with a randomly selected initial condition. Compute $G_E(t)$ which is given by the average results of the 50 runs.

Note that in step 4, different initial conditions are considered in order to capture the effect of transient behavior on the response time distribution.

For each of the 100 cases that we have evaluated, we compute the difference between $G(t)$, as given by Equation (5.15) and $G_E(t)$, as obtained from the simulation experiments. The accuracy of $G(t)$ is measured by its difference from $G_E(t)$. This difference is defined as follows:

$$C = |G_E(t) - G(t)|/G_E(t) * 100\% \quad (6.1)$$

We show in Figures 6.3, 6.4, and 6.5, the values of C for the 100 cases for $D = 0.5T$, T and $5T$, respectively. Note that the y-axis scales of these three figures are not the same. We observe that the steady state results are more accurate when D is larger. This is consistent with the observation that a larger D means that the system is in steady state for a longer time period and as a result, the steady state results should be more accurate. We also observe that, with the exception of a few outliers (less than 10% of the cases), $G(t)$ is quite accurate in estimating the response time distribution; the percentage difference C is less than 20% for most cases, even when $D = 0.5T$. These outliers are likely due to using a short sequence of random numbers when the simulation has duration of $5T$ or less.

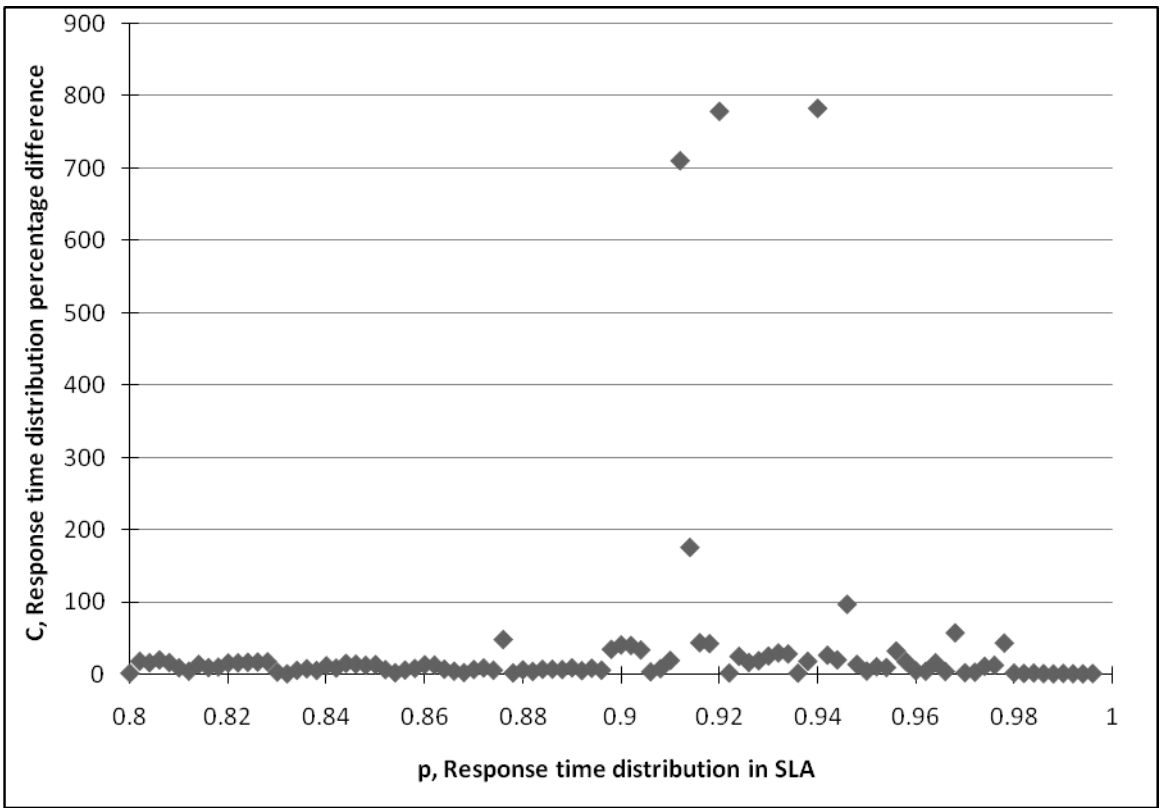

Figure 6.3: Response time distribution percentage difference when $D = 0.5T$

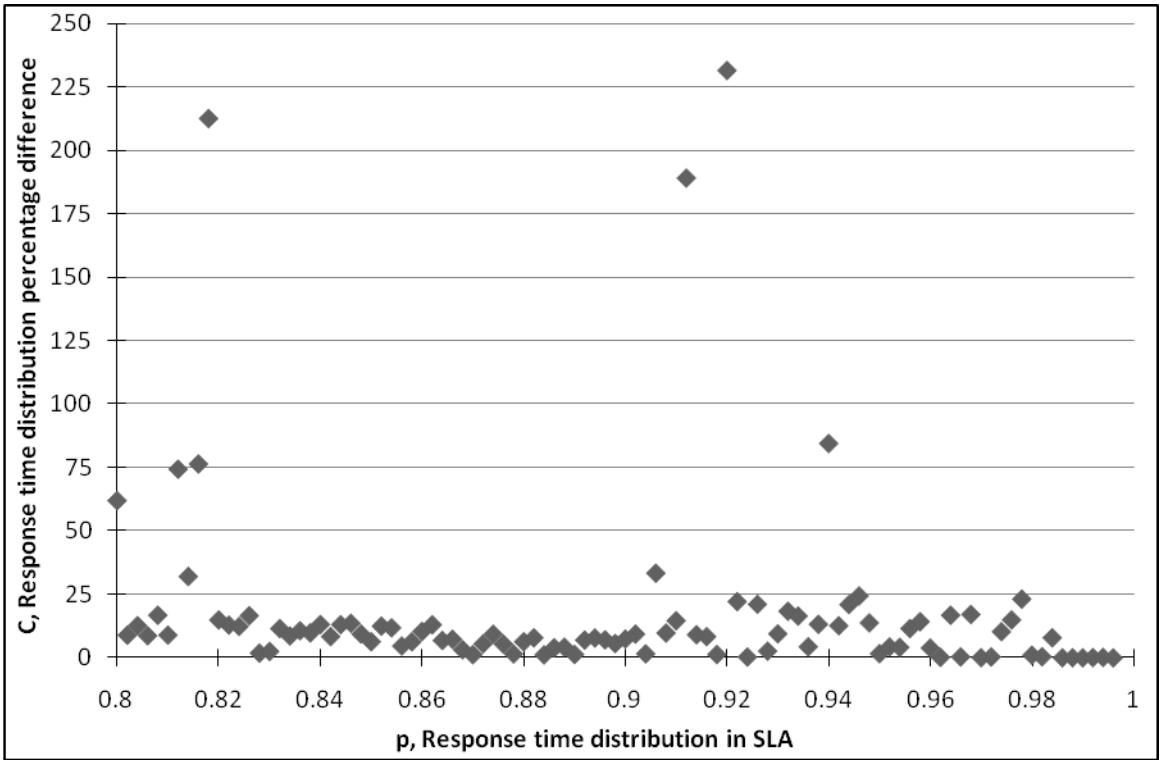

Figure 6.4: Response time distribution percentage difference when $D = T$

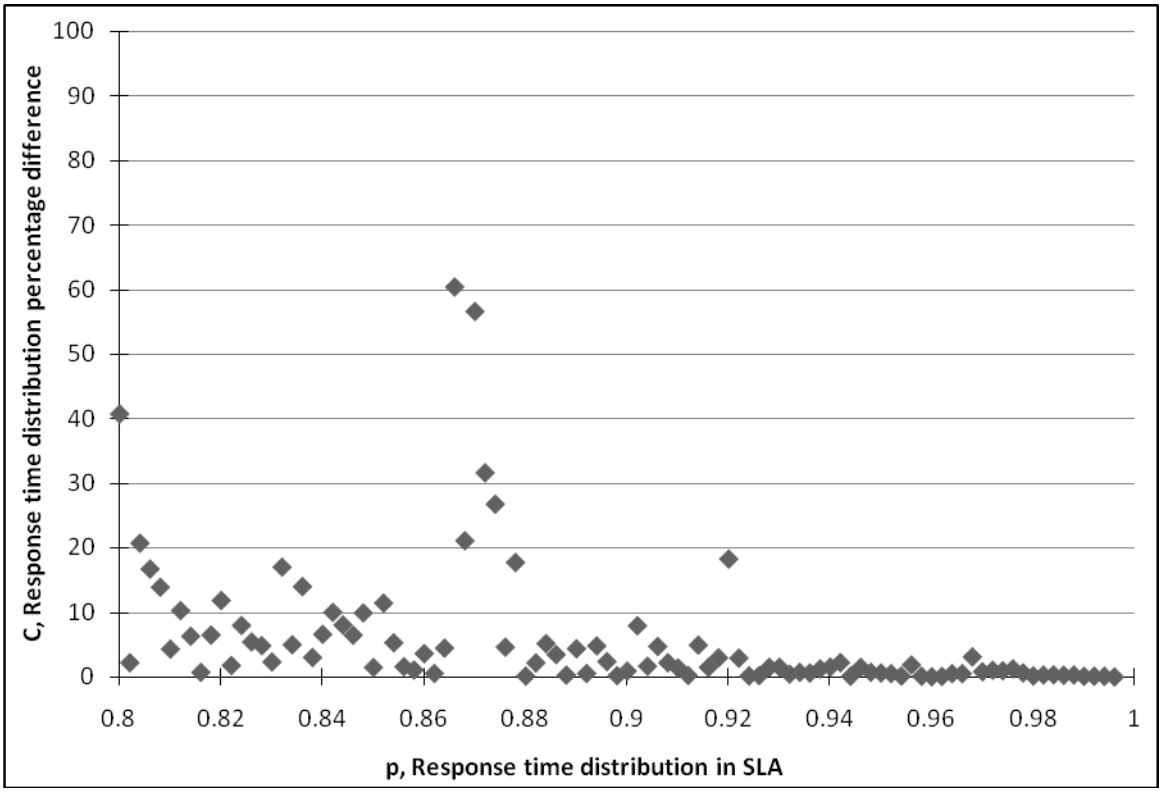

Figure 6.5: Response time distribution percentage difference when $D = 5T$

6.2 Input to dynamic resource provisioning

Using our steady state results in Chapter 5, dynamic resource allocation can determine the number of processor nodes required m_{new} for given values of λ, y, g and μ such that the SLA is met. We use $m(t,p)$ to denote the required number of nodes that would meet the SLA given by Prob [response time $\leq t$] $\geq p$.

6.2.1 Number of processor nodes required

We first evaluate the accuracy of $G(t)$ in estimating $m(t,p)$. As in our previous examples, g, y, and μ are set to 0.1, 0.05, and 1.0, respectively. The other parameters, namely λ, t, and p are shown in Table 6.1. There are 27 combinations and for each combination, we obtain $m(t,p)$ using Equation (5.15) and by simulation. In both cases, $m(t,p)$ is obtained by increasing m from 1 until the conditions Prob [response time $\leq t$] $\geq p$ and $q_M < 1\%$ are met. For simulation of each combination, three values of D, namely, $D = 0.5T$, T, and $5T$ are considered.

g	0.1
μ	1
y	0.05
λ	0.05, 0.075, 0.1
t	3.0, 4.0, 5.0
p	85%, 90%, 95%

Table 6.1: Input parameter values

We show in Tables 6.2 to 6.4 the results for the different combinations of λ, t, and p. We observe that with exception of the case $\lambda = 0.05$, $t = 3.0$ and $D = 0.5T$ in Table 6.4, Equation (5.15) is accurate in predicting the number of processor nodes required. The same number of nodes is obtained by simulation for about 75% of the cases, and for the remaining cases, the maximum difference is one node. We further observe that the accuracy of Equation (5.15) improves when D becomes larger. This is expected because a larger D means that the system is in steady state for a longer period of time and Equation (5.15) is for the response time distribution at steady state.

λ	t	$m(t,p)$ estimated using Equation (5.15)	$m(t,p)$ obtained using simulation $D = 0.5T$	$m(t,p)$ obtained using simulation $D = T$	$m(t,p)$ obtained using simulation $D = 5T$
0.05	3.0	2	2	2	2
0.05	4.0	2	2	2	2
0.05	5.0	2	2	2	2
0.75	3.0	3	2	2	3
0.75	4.0	2	2	2	2
0.75	5.0	2	2	2	2
0.1	3.0	3	3	3	3
0.1	4.0	3	2	2	3
0.1	5.0	3	2	3	3

Table 6.2: $m(t,p)$ when $p = 85\%$

λ	t	$m(t,p)$ estimated using Equation (5.15)	$m(t,p)$ obtained using simulation $D = 0.5T$	$m(t,p)$ obtained using simulation $D = T$	$m(t,p)$ obtained using simulation $D = 5T$
0.05	3.0	2	2	2	2
0.05	4.0	2	2	2	2
0.05	5.0	2	2	2	2
0.75	3.0	3	3	3	3
0.75	4.0	3	2	2	3
0.75	5.0	2	1	2	2
0.1	3.0	4	2	4	4
0.1	4.0	3	2	3	3
0.1	5.0	3	2	2	3

Table 6.3: $m(t,p)$ when $p = 90\%$

λ	t	$m(t,p)$ estimated using Equation (5.15)	$m(t,p)$ obtained using simulation $D = 0.5T$	$m(t,p)$ obtained using simulation $D = T$	$m(t,p)$ obtained using simulation $D = 5T$
0.05	3.0	5	7	5	5
0.05	4.0	2	2	2	2
0.05	5.0	2	2	2	2
0.75	3.0	6	6	5	5
0.75	4.0	3	2	3	3
0.75	5.0	3	2	2	3
0.1	3.0	7	7	6	6
0.1	4.0	3	3	3	3
0.1	5.0	3	3	3	3

Table 6.4: $m(t,p)$ when $p = 95\%$

6.2.2 Dynamic resource allocation decision

In this subsection, we discuss how our steady state results in Equation (5.15) can be used to guide dynamic resource allocation decisions. Ideally, we would like to come up with simple rules that relate $m(t,p)$ mathematically to the input parameters M, λ, y, g and μ. In an attempt to determine whether such rules are possible or not, we use numerical examples to gain a good understanding of the impact of input parameters on $m(t,p)$. In our examples, we use $\mu = 1$ because mean service time is used as our time unit and $g = 0.1$ in order to reduce the number of cases considered. $m(t,p)$ is obtained by increasing m from 1 until the condition Prob [response time $\leq t$] $\geq p$ is met.

We first consider the case where $y = 0.05$, and λ takes on values 0.05, 0.075 and 0.1. In Figures 6.6 and 6.7, $m(t,p)$ is plotted against the response time goal t for $p = 90\%$ and 95%, respectively. As expected, $m(t,p)$ tends to increase with λ because more processor nodes will likely be needed to support a higher rate of requests for establishing new sessions. $m(t,p)$ also tends to decrease with the response time goal t because a larger t means a less demanding response time constraint from the users and fewer nodes can be used to meet the response time requirements. However, we are not able to come up with a simple rule that relates $m(t,p)$ mathematically to the input parameters.

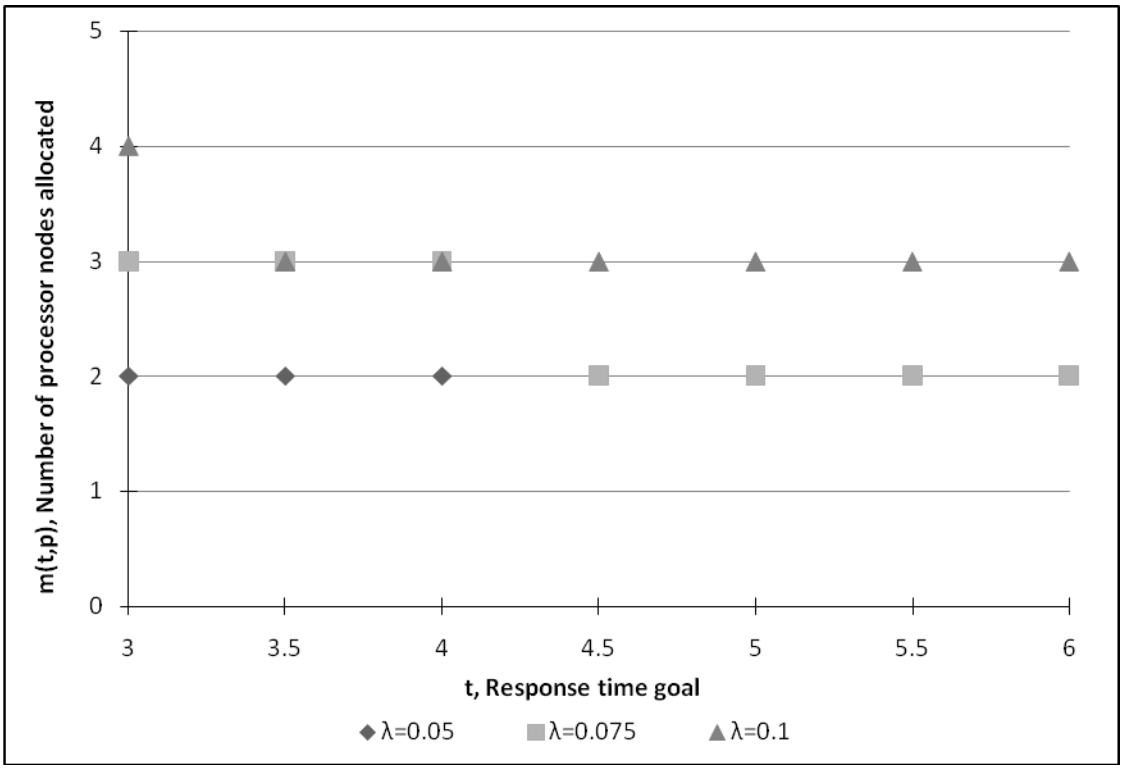

Figure 6.6: $m(t,p)$ when $p = 90\%$

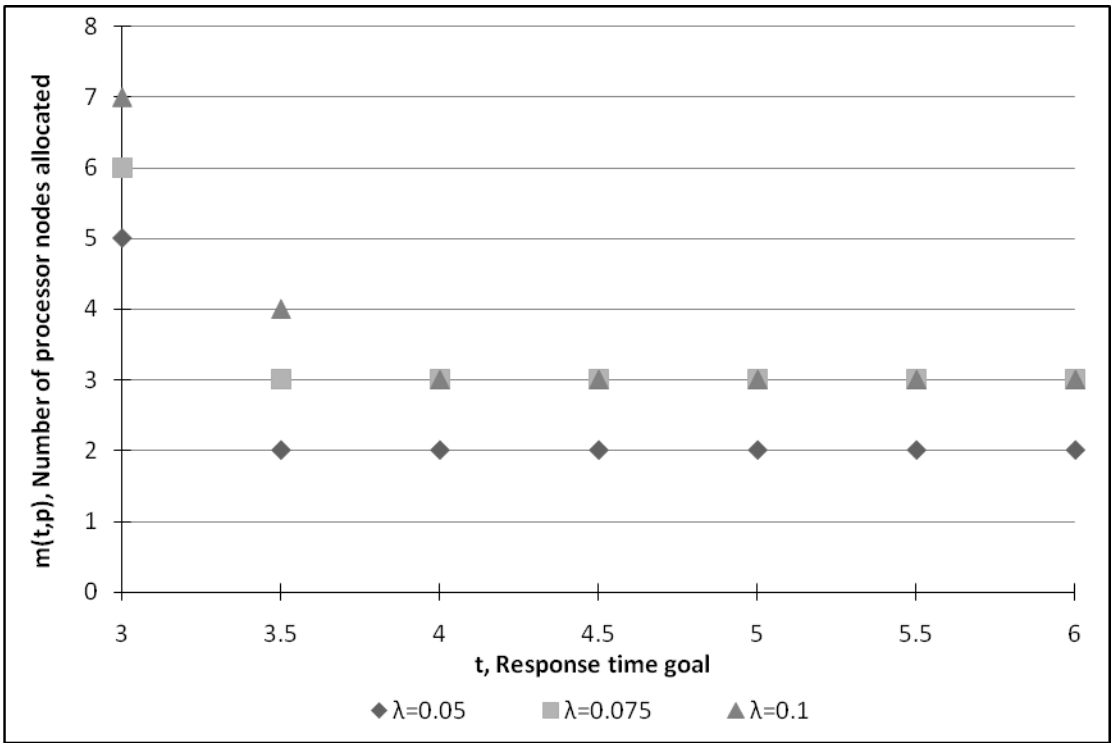

Figure 6.7: $m(t,p)$ when $p = 95\%$

At a minimum, the dynamic resource allocation algorithm can use a table lookup method to determine $m(t,p)$. We illustrate this method by the following example. Suppose it has been determined that $\mu = 1$, $g = 0.1$, and $y = 0.05$. The workload, as characterized by λ may be time-varying, but its value is always within the range of (0.04, 0.1). As to the SLA, the response time goal t considered is in the range (3, 6) and the target probability p is larger than 85%. We pre-compute $m(t,p)$ for different combinations of λ, t, and p, organized by sub-intervals within their respective ranges. The resulting table is shown in Table 6.5. We have also included in this table the recommended minimum value of M such that the blocking probability at the higher level is less than 1% and the SLA is met. With this table, the dynamic resource allocation algorithm can determine $m(t,p)$ by table lookup.

It is important to note that the results in Table 6.5 are for illustration purposes only. In general, more sub-intervals should be defined for each parameter and the size of the table should be much larger.

λ	t	p	$m(t,p)$	Minimum M
0.04 - 0.06	3 - 4	85% - 90%	2	23
0.04 - 0.06	3 - 4	90% - 95%	3	22
0.04 - 0.06	3 - 4	95% - 100%	5	22
0.04 - 0.06	4 - 5	85% - 90%	2	23
0.04 - 0.06	4 - 5	90% - 95%	2	23
0.04 - 0.06	4 - 5	95% - 100%	3	22
0.04 - 0.06	5 - 6	85% - 90%	2	23
0.04 - 0.06	5 - 6	90% - 95%	2	23
0.04 - 0.06	5 - 6	95% - 100%	2	23
0.06 - 0.08	3 - 4	85% - 90%	3	28
0.06 - 0.08	3 - 4	90% - 95%	3	28
0.06 - 0.08	3 - 4	95% - 100%	6	27
0.06 - 0.08	4 - 5	85% - 90%	3	28
0.06 - 0.08	4 - 5	90% - 95%	3	28
0.06 - 0.08	4 - 5	95% - 100%	3	28
0.06 - 0.08	5 - 6	85% - 90%	2	32
0.06 - 0.08	5 - 6	90% - 95%	3	28
0.06 - 0.08	5 - 6	95% - 100%	3	28
0.08 - 0.1	3 - 4	85% - 90%	3	34
0.08 - 0.1	3 - 4	90% - 95%	4	33
0.08 - 0.1	3 - 4	95% - 100%	7	32
0.08 - 0.1	4 - 5	85% - 90%	3	34
0.08 - 0.1	4 - 5	90% - 95%	3	34
0.08 - 0.1	4 - 5	95% - 100%	3	34
0.08 - 0.1	5 - 6	85% - 90%	3	34
0.08 - 0.1	5 - 6	90% - 95%	3	34
0.08 - 0.1	5 - 6	95% - 100%	3	34

Table 6.5: Resource allocation table

Chapter 7

Conclusion and future work

7.1 Conclusion

In this thesis, we have developed a new interactive system model where the number of logon users may change over time and have obtained approximate analytic results on response time distribution for this model. We have also discussed how these results may be used in dynamic resource provisioning. Our contributions are summarized below.

1. We have obtained approximate analytic results for the response time distribution at steady state for a new interactive system model. Our results are shown to be acceptable when compared with exact results obtained by a numerical method.
2. We have evaluated the accuracy of using our steady state results as an approximation to the response time distribution over an operation interval between successive decision points in dynamic resource provisioning.

3. We have used a numerical method to obtain transient results for our model. These results provide valuable insights into the transient behavior of the system for different initial conditions.
4. We have discussed how our steady state results can be used by a dynamic resource allocation algorithm to determine the minimum number of processor nodes required to meet a given service level agreement and an application.

7.2 Future work

Possible future work is discussed below.

1. For our model, the blocking probability at the higher level is given by q_M in Equation (5.13). The trade-off between blocking probability and the number of processor nodes allocated is a topic for future investigation.
2. Our work only considers one class of jobs. Extension of our investigation to multiple job classes will provide results for more complex application scenarios.
3. Our investigation did not reveal any heuristic that mathematically relates the $m(t,p)$ to the input parameters. More work on identifying such a heuristic will be valuable to dynamic resource provisioning.
4. Extension of our work to virtualization of resources is worthy of investigation. With virtualization, the system can allocate a fraction of the server capacity to a job class. This would provide more flexibility in resource allocation than allocating an integer number of processor nodes.

Bibliography

[1] K. Shen, H. Tang, T. Yang and L. Chu, "Integrated resource management for cluster-based Internet services," in *SIGOPS Oper.Syst.Rev.*, vol. 36, pp. 225-238, 2002.

[2] L. Sung, *Automatic resource management for a cluster that executes batch jobs*, Master's thesis, University of Waterloo, School of Computer Science, 2006.

[3] M. N. Bennani and D. A. Menasce, "Resource allocation for autonomic data centers using analytic performance models," in *ICAC '05: Proceedings of the 2nd International Conference on Autonomic Computing*, pp. 229-240, 2005.

[4] D. A. Menasce and M. N. Bennani, "On the use of performance models to design self-managing computer systems," in *Proc. 2003 Computer Measurement Group Conf*, pp. 7-12, 2003.

[5] D.A. Menasce, V.A.F. Almeida and L. W. Dowdy, *Performance by Design: Computer Capacity Planning by Example*, Prentice Hall, Upper Saddle River, NJ, 2004.

[6] Q. Zhang, L. Cherkasova and E. Smirni, "A regression-based analytic model for dynamic resource provisioning of multi-tier applications," in *ICAC '07: Proceedings of the 4th International Conference on Autonomic Computing*, pp. 27, 2007.

[7] B. Urgaonkar and A. Chandra, "Dynamic provisioning of multi-tier internet applications," in *ICAC '05: Proceedings of the 2nd International Conference on Autonomic Computing*, pp. 217-228, 2005.

[8] R. P. Doyle, J. S. Chase, O. M. Asad, W. Jin and A. M. Vahdat, "Model-based resource provisioning in a web service utility," in *USITS'03: Proceedings of the 4th Conference on USENIX Symposium on Internet Technologies and Systems*, pp. 5-5, 2003.

[9] P. Padala, et al, "Adaptive control of virtualized resources in utility computing environments," in *EuroSys '07: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, pp. 289-302, 2007.

[10] Bin Lin, A. I. Sundararaj and P. A. Dinda, "Time-sharing parallel applications with performance isolation and control," in *ICAC '07: Proceedings of the 4th International Conference on Autonomic Computing*, pp. 28-28, 2007.

[11] Y. Diao, et al, "Using MIMO feedback control to enforce policies for interrelated metrics with application to the apache web server," in *In Proceedings of the Network Operations and Management Symposium 2002*, pp. 219-234, 2002.

[12] P. Shivam, S. Babu and J. S. Chase, "Learning application models for utility resource planning," in *ICAC '06: Proceedings of the 3rd International Conference on Autonomic Computing*, pp. 255-264, 2006.

[13] V. Sundaram and P. J. Shenoy, "A practical learning-based approach for dynamic storage bandwidth allocation," in *IWQoS*, pp. 479-497, 2003.

[14] HP Utility Data Center, Hewlett-Packard. http://h71028.www7.hp.com/ enterprise/cache/259007-0-0-225-121.html?jumpid=reg_R1002_USEN.

[15] R. Jain, *The Art of Computer Systems Performance Analysis*, Wiley, New York,1991.

[16] A.Sekino, "Performance Evaluation of Multiprogrammed Time-Shared Computer System," *Project MAC*, Massachusetts Institute of Technology, MAC-TR-103, 1972.

[17] A.Sekino, "Response time distribution of multi-programmed time-shared computer systems," in *Proceedings of the 6th Annual Princeton Conference on Information Sciences and Systems*, pp.613-619, 1972.

[18] L. Kleinrock, *Queueing Systems*, Wiley, New York, 1975.