POWER OF d CHOICES FOR LARGE-SCALE BIN PACKING: A LOSS MODEL

BY

XIAOBO DONG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Adviser:

Professor R. Srikant

ABSTRACT

A system with N parallel servers is considered in our thesis. Each server consists of B units of a resource and jobs arrive at this system according to a Poisson process. Each job stays in the system for an exponentially distributed amount of time. Moreover, each job may request different units of the resource from the system. Our goal is to understand how to route arriving jobs to the servers to minimize the probability that an arriving job does not find the required amount of resource at the server, i.e., the goal is to minimize blocking probability. Our motivation arises from the design of cloud computing systems in which the jobs are virtual machines (VMs) that request resources such as memory from a large pool of servers. In our thesis, we consider power-of-d-choices routing, where a job is routed to the server with the largest amount of available resources among $d \geq 2$ randomly chosen servers. We consider a fluid model that corresponds to the limit as N goes to infinity, and use numerical methods to approximate the blocking probability. Moreover, we also show the simulation for the system.

To my parents, for their love and support.

ACKNOWLEDGMENTS

First, I thank my advisers, Prof. R. Srikant and Prof. Lu, for their guidance and support. I also thank Xie who taught me many aspects of queueing theory. Most importantly, I thank my parents for their unconditional love without which I would not have studied abroad.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Widespread adoption of cloud services, particularly public cloud services, is driving the demand for cloud equipment among cloud-based IT service providers. For example, Amazon Elastic Compute Cloud (Amazon EC2) [1] is a virtual machine service that provides resizable compute capacity in the cloud. Another prominent company named its virtual machine the Google Cloud Platform which is a set of modular cloud-based services that allow you to create anything from simple websites to complex applications [2]. Azure [3] is Microsoft's cloud platform; it is a growing collection of integrated services including compute, storage, data, networking, and APP that enable users to move faster and do more. This list of virtual machines is by no means exhaustive, it is provided as a representation of growing ubiquity of cloud computing. Figure 1.1 [4] shows the tremendous compound annual revenue growth of cloud-based IT service.

With the motivation that cloud-based IT service becomes more and more ubiquitous, we believe that cloud computing optimization will play an important role in the future. One of the key procedures or steps in cloud computing is the load balancing, that is the special case of resources allocation, where the processors in cloud computing are resources to be allocated. For example, in Amazon EC2, the strategy used is called Elastic Load Balancing that automatically distributes incoming application traffic across multiple Amazon EC2 instances in the cloud. It enables the users to achieve greater levels of fault tolerance in the applications, seamlessly providing the required amount of load balancing capacity needed to distribute the application traffic. Therefore, we have a very clear view of the advantages of using a good load balancing strategy. Regardless of the commercial expression, load balancing in the cloud computing provides an efficient solution to various issues residing in cloud computing environment setup and usage. In general, load

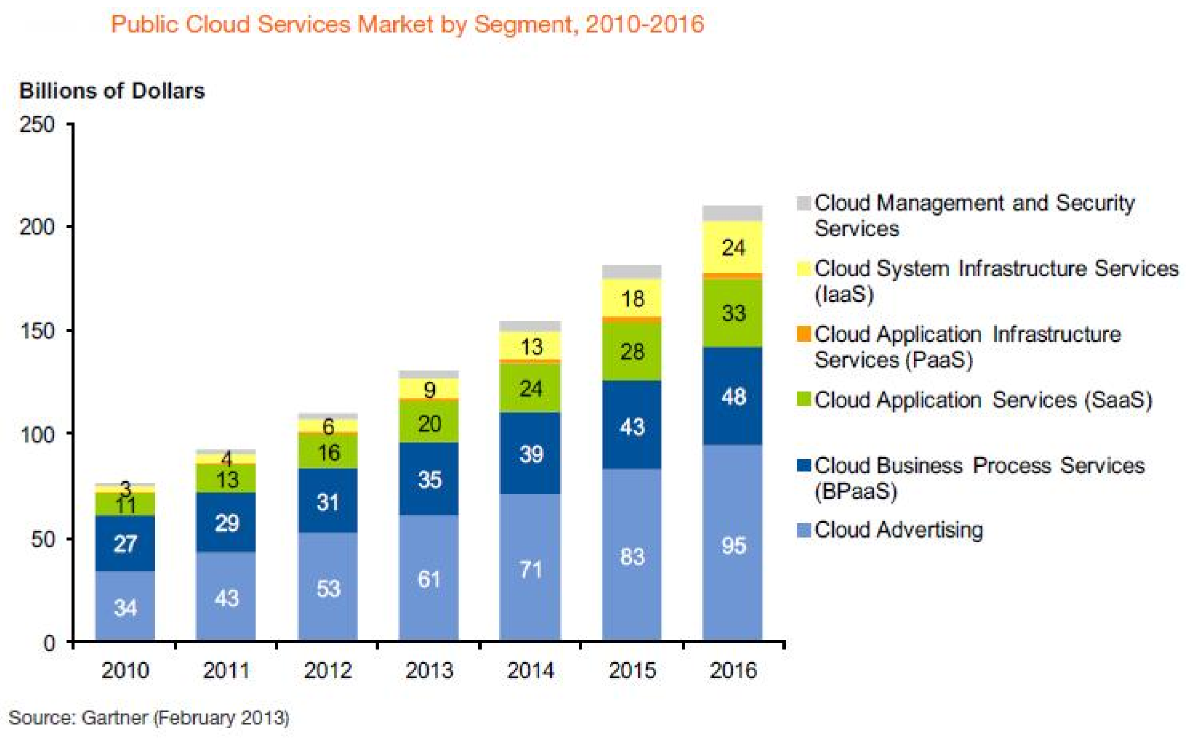

Figure 1.1: The annual revenue growth of cloud-based IT service

balancing distributes workloads across multiple computing resource, such as computers, a computer cluster, network links, central processing units or disk drives. Load balancing aims to optimize resource use, maximize throughput, minimize response time, and avoid overload of any single resource. Before discussing a complicated system, we consider a simple and traditional scenario of the bin packing problem, which has been considered in earlier work [5]. Suppose that n balls are placed into n bins, and the goal is to have the number of balls in a bin with the most balls as small as possible. It is obvious that the optimal solution is 1, but in order to achieve the optimal solution we need to record the bins which have already been occupied, i.e. whenever there is a ball, we put it into the empty bin. However, for simplicity, we can place each ball into a bin chosen independently and uniformly at random, which totally gets rid of the record list. Then from the result in [5] with high probability, the maximum load in any bin is approximately $\frac{\log n}{\log \log n}$. Suppose instead that each ball is placed sequentially into the least full of d bins chosen independently and uniformly at random. The work in [5] has also shown that the maximum load is then only $\frac{\log \log n}{\log d} + O(1)$ with high probability. Thus, with only one more choice for a ball, the result leads to an exponential improvement. The bin packing problem is a static problem, where the balls never leave the bin once they arrive. What we are interested in or what a real virtual machine problem looks like is a dynamic problem, where a job arrives in a server and stays in the server for processing for a while, and then leaves the server. With the idea of bin packing problem in mind, we will use the idea in our problem.

Now let us simply and intuitively discuss the model. The model that we consider in our thesis is a cloud computing system with N parallel servers, and each server consists of B units of a resource. Jobs arrive at this system according to a Poisson process, and each job stays in the system for an exponentially distributed amount of time. Each job may request different units of the resource from the system. If the job selects a server that does not have adequate available resources to support the job, the job will be dropped without service permanently. Whenever there is a dropped job, the system may request the data from the disk again or even wait for finishing that job to process the next task, such that it will increase the system delay or require extra cost of the energy. Therefore, the goal is to understand how to route arriving jobs to the servers to minimize the probability that an arriving job does not find the required amount of resources at the server, i.e., the goal is to minimize blocking probability. The motivation for this problem arises from the design of cloud computing systems in which the jobs are virtual machines (VMs) that request resources such as memory from a large pool of servers. And we believe this problem will play an important role in the future. Figure 1.2 demonstrates the model we are presenting.

There exists a very intuitive strategy that whenever there is an arriving job, it will join the server with the smallest load, i.e., the load with the largest available resource in the server. If there are even resources among several servers, the job will select one of the servers uniformly at random. This is an optimal strategy in the sense that in theory joining the server with the largest amount of the available resource may achieve the best performance. However, it is not optimal in the sense that in the real system each query of a server may slow down the speed of processing. Basically, whenever there is a query of a server, the server has to pause the current process and send the information of the resource to the central system which selects a server

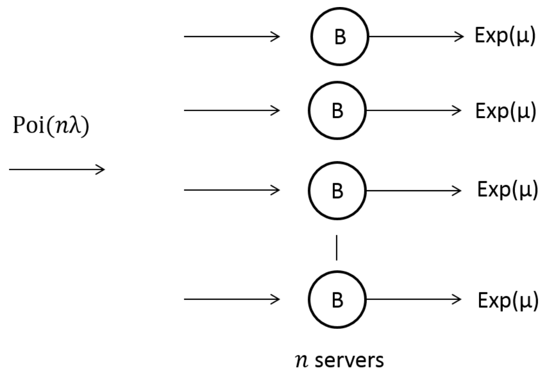

Figure 1.2: High-level representation of our model

for a job. However, in the real system, there are thousands of the servers, so that searching through all the servers for the smallest load is impractical. Finding the server with smallest load may take a lot of time, and result in a tremendous performance delay.

Since the price of knowing all the information of the servers is very high, it is obvious to consider the scenario of knowing nothing about the servers. In this case, the central system does not query any information from the servers. Thus, the only work it can do is to randomly select a server for all the arriving jobs. Randomly selecting a server totally gets rid of querying the information about a server, and at least this strategy improves the delay performance. However, the performance is not as good as we expected, and a detailed analysis will be shown in Chapter 3. Here we can give a very intuitive and brief explanation. The key idea comes from the fact that separating a stream of Poisson arrival uniformly at random into N different streams, and each stream is still a Poisson stream. With this fact, each server will act as an $M/M/s/s$ queue, where the queueing system has s servers and no buffer. And the performance of the overall system is closed to one $M/M/s/s$ queue.

Inspired by the bin packing problem, we consider power-of-d-choices routing, where a job is routed to the server with the largest amount of the available resource among $d \geq 2$ randomly chosen servers. Now the scenario switches from knowing all the information or knowing nothing to knowing partial information of the servers. Obviously, in the blocking probability sense knowing partial information cannot defeat knowing all the information. However, it is much more practical than knowing all the information. On the other hand, knowing partial information definitely has a greater advantage than knowing nothing in the blocking probability sense. The power-of-d strategy seems to be mediocre, but such a mediocre strategy has a very favorable performance both in practice and in theory. For simplicity, we will consider the case that all the jobs use one unite of resource and the central system just selects two candidates from all the servers for each arriving job, i.e., $d = 2$.

This thesis is organized as follows. In Chapter 2, we will first review a model called the supermarket model in [5] and also analyze the random-choose-one strategy. In the supermarket model, we will present a fluid limit analysis, and give a flavor of the idea of the fluid limit analysis. In Chapter 3, we will define the system and explain the notation at the beginning. Then we will represent our fundamental result, the analysis and the simulation results. In Chapter 4, we conclude our thesis with potentially further analysis improvement and future work.

CHAPTER 2

REVIEW

The review section contains two parts. In the first part, the *supermarket model* which is introduced in early work [5] and [6] will be discussed and then an analysis using the idea of fluid limits will be provided. The model inspired us so much that we choose to present this model at the beginning of this chapter. The second part will contain the analysis of one of the cases in the model we mentioned in Chapter 1 that the arriving jobs know nothing about the servers.

2.1 Supermarket Model

The differences between the *supermarket model* and our model are the following:

(1) In the *supermarket model*, each server has a queue that the arriving jobs or customers can stay in to wait for the service. However, in our model each server has a certain amount of the resource and there is no queue in the server. Therefore, if a job joins the server without enough of the resource, the job will be dropped without service. Therefore, we call our model the loss model.

(2) The goal for the *supermarket model* is to minimize the overall delay of the system. However, in the loss model, the number of servers we selected is very small and negligible, and the waiting time for getting the load information of a server is negligible. Therefore, a delay in performance will not be considered in the loss model. Indeed, the performance criterion for the loss model we considered is the blocking probability.

In Chapter 1, we discussed the bin packing problem, where a ball never leaves the bin or box once it arrives. That is a static problem. The *supermarket model* is a dynamic version of the bin packing problem.

Now let us consider the following dynamic model:

Description: *There are N servers in the system. Customers arrive as a Poisson stream of rate $N\lambda$, where $\lambda < 1$. We allow each customer to choose d of servers independently and uniformly at random from the N servers. The customer waits at the server currently with the fewest customers (ties being broken arbitrarily). Moreover, customers who come first are served first, and the service time for each customer is exponentially distributed with mean 1.*

This model can be used to represent many systems in real life. Among all the systems in real life, the *supermarket model* is one of the most representative. This model can vividly describe the customers and cashiers in the supermarket, which is where the name orginates. Whenever you finish shopping in one of your favorite supermarkets, you would like to find the cashier you expect has the shortest waiting time. As an individual, you definitely do not want to spend excessive time in the waiting line. As a whole system or a whole supermarket, the owner is concerned about the overall system performance, like the accumulative waiting time for all of the customers. Now we are interested in the owner's benefit; that is the expected waiting time for all of the customers in the supermarket. Moreover, in general, waiting time spent in the system in equilibrium is a natural measure of a system performance.

Now consider the system we are exploring. In the description of the system, we know the average arrival rate for each queue is $\lambda < 1$, and the average service rate for each queue is 1. From the basic queueing theory knowledge, it is trivial to have the conclusion that the expected number of customers per queue remains finite in equilibrium. Furthermore, we assume that the time for customers to move to a server is 0, and the service rate is the same at each server.

It is obvious to see that the optimal strategy for each customer in theory is to select the cashier with the shortest line. Still, it is practical in some small markets, such that finding the shortest line is doable. However, as you may

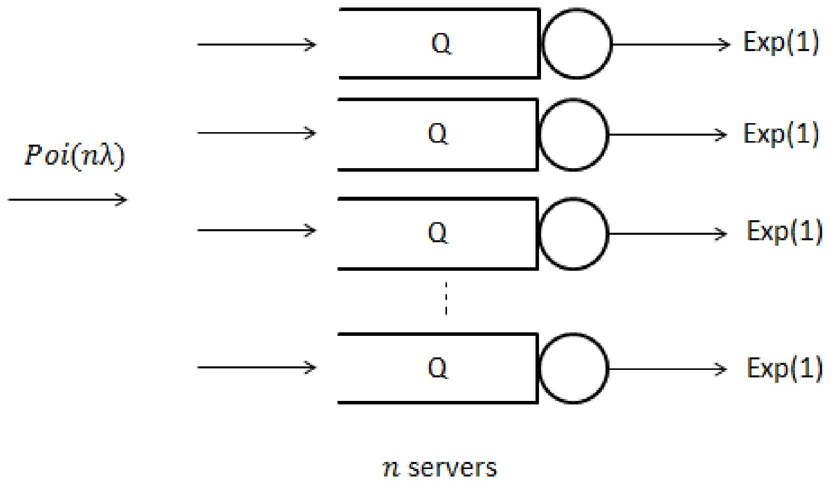

Figure 2.1: Simplified version of the loss model

have experienced, looking through all the lines to find the shortest one may take a lot of time in a large supermarket. Therefore, normally, what we did is to randomly select a portion of the cashiers. Then we selected the cashier with the shortest line among the portion. In the *supermarket model*, there still are different strategies.

(1) The customers obtain all the information about the cashiers.

(2) The customers just obtain the information about their favorite (randomly selected) portion of the cashiers.

(3) The customers have no information about the cashiers.

Now let us consider the scenario that each customer does not have any information about the cashiers. Therefore, what a customer can do is randomly select one of the cashiers and join the line to wait. In this case, a customer randomly selects one cashier, i.e., $d = 1$. A very useful fact is given below referred from [7].

FACT 1 *Assume that $N(t)$ is a Poisson process. We can generate K random process $N_1(t), ..., N_K(t)$ as follows: when there is an arrival according to $N(t)$, make it an arrival process $N_k(t)$, with probability p_k, where $\sum_{k=1}^{K} p_k = 1$. Then, $N_1(t), ..., N_K(t)$ are independent Poisson processes with parameters $\lambda p_1, ..., \lambda p_K$, respectively.*

Then we know the fact that a Poisson stream with rate $n\lambda$ randomly selected with probability p from the arrival is still a Poisson process with rate $pn\lambda$. Therefore, the arrival Poisson stream can be split into independent Poisson streams for each server. Therefore, each server itself is a simple M/M/1 queue as in Figure 2.2.

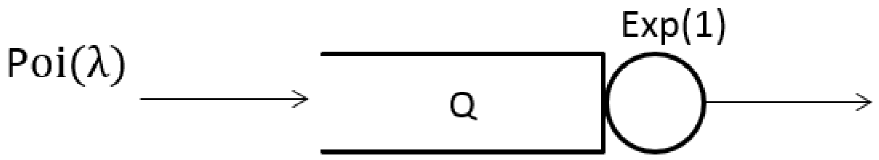

Figure 2.2: M/M/1 queue

Further, we know that the M/M/1 queue can be modeled as a Markov chain as shown in Figure 2.3.

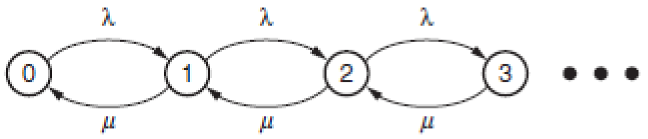

Figure 2.3: Markov chain for an M/M/1 queue

Let $q(t)$ denote the number of customers in the system, which forms a time-homogeneous continuous time Markov chain. Let $\pi(n)$ be the steady-state probability that there are n customers in the system. Note that $P_{ij} = 0$ if $j \neq i-1$ or $i+1$ and the inter-arrival times and service times are exponential, so the transition rate matrix $\mathbf{Q}$ is given by

$$Q_{ij} = \begin{cases} \lambda & \text{if } j = i+1 \\ \mu & \text{if } j = i-1 \\ -\lambda-\mu & \text{if } j = i \\ 0 & \text{otherwise} \end{cases}$$

We know that stationary distribution $\boldsymbol{\pi}$ (where the bold $\boldsymbol{\pi}$ represents a vector with size of $n+1$) satisfies the following condition

$$\boldsymbol{\pi}\mathbf{Q} = 0$$

Then it is equivalent to have

$$\begin{aligned}\pi(1) &= \rho\pi(0)\\ \pi(2) &= \rho\pi(1) = \rho^2\pi(0)\\ &\vdots\\ \pi(n) &= \rho\pi(n-1) = \rho^n\pi(0)\end{aligned}$$

where $\rho = \frac{\lambda}{\mu}$. Since we know $\sum\limits_{n=0}^{\infty} \pi(n) = 1$, we have

$$\pi(0)\sum_{n=0}^{\infty}\rho^n = 1$$

Since we know that $\lambda < 1$ and $\mu = 1$, then $\rho < 1$. Therefore, we have

$$\sum_{n=0}^{\infty}\rho^n = \frac{1}{1-\rho}$$

which yields

$$\pi(n) = \rho^n(1-\rho)$$

Then the mean number of customers in the system is given by:

$$L = \sum_{n=0}^{\infty} n\pi(n) = \frac{\rho}{1-\rho}$$

And the expected waiting time can be obtained by the following Little's law referred from [7]:

FACT 2 *The long-term average number of customers in a stable system L is equal to the long-term average effective arrival rate λ multiplied by the average time a customer spends in the system W or expressed: $L = \lambda W$.*

Thus, the expected waiting time for $d = 1$ is $\frac{1}{1-\rho}$.

In the first case, the customers obtain all the information about the cashiers. Indeed, this is not practical, so we will totally ignore this case.

Now we have the expected waiting time in the third case that the customers have no information about the cashiers. Then let us consider the case that the customers have the information about their favorite portion of the cashiers.

For simplicity, let us consider the case where the customers just know the information about randomly selected cashiers, i.e., $d = 2$.

For $d = 2$, it is difficult to analyze the system, where the length of one queue can affect the distribution of the length of the other queues. Our analysis is mainly based on [5]. We focus on one of the servers (i.e. server 1), and let i denote the number of jobs in server 1. Then we can have the following Markov chain in Figure 2.4, where λ_i is a function that depends on many things, like whether server 1 is selected, and if the other selected server has more jobs than server 1.

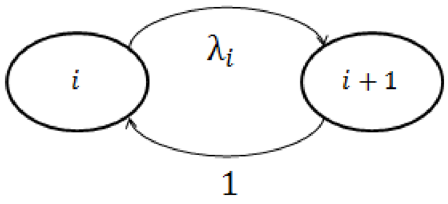

Figure 2.4: State i in the Markov chain

Now we will analyze the $d = 2$ case using the mean-field approach. In the mean-field approach, we analyze the system as n goes to infinite. Therefore, we can make a reasonable assumption about the pairwise independence among those n servers. Let π_i denote the probability that a server has i jobs and let s_i denote the probability that a server has at least i jobs.

$$\begin{aligned}
P^* &= P(\text{routing to server 1, when it has } i \text{ packets}) \\
&= \sum_{j=2}^{n} P(\text{server 1 and server } j \text{ selected})\times \\
&\left(\frac{1}{2}P(\text{server } j \text{ has } i \text{ packets}) + P(\text{server } j \text{ has more than } i \text{ packets})\right) \\
&= \frac{2}{n}(\frac{1}{2}\pi_i + \sum_{k=i+1}^{\infty} \pi_k) \\
&= \frac{2}{n}\left(\frac{1}{2}(s_i - s_{i+1}) + s_{i+1}\right) \\
&= \frac{1}{n}\left(s_i + s_{i+1}\right)
\end{aligned}$$

Therefore, we can get

$$\lambda_i = n\lambda P^* = \lambda(s + s_{i+1})$$

Using the local balance equation on the Markov chain, we have

$$\begin{aligned}
&\lambda_i \pi_i = \pi_{i+1} \\
&(s_i + s_{i+1})\lambda(s_i - s_{i+1}) = (s_{i+1} - s_{i+2}) \\
&\lambda(s_i^2 - s_{i+1}^2) = (s_{i+1} - s_{i+2})
\end{aligned}$$

Solving the recursive equation, we have

$$s_n = \lambda^{\frac{2^n-1}{2-1}} = \lambda^{2^n-1}$$

The expected queue length is given by

$$E[Q] = \sum_{i=1}^{\infty} i\pi_i = \sum_{i=1}^{\infty} i(s_i - s_{i+1}) = \sum_{i=1}^{\infty} s_i$$

By Little's law, the expected waiting time is

$$E[W] = \frac{\sum\limits_{i=1}^{\infty} \lambda^{2^i-1}}{\lambda} = \sum_{i=1}^{\infty} \lambda^{2^i-2}$$

Lemma 1 *For $\lambda \in [0,1]$, $T_2(\lambda) \leq c_2(\log T_1(\lambda))$ for some constant c_2. T_1 denotes the expected waiting time in the random-choose-one system, i.e. $T_1(\lambda = \frac{1}{1-\lambda})$ and similarly T_2 denotes the expected waiting time in the random-choose-two system.*

Then

$$\lim_{\lambda \to 1^-} \frac{T_2(\lambda))}{\log T_1(\lambda)} = \frac{1}{\log 2}$$

Given by Lemma 1.1, the expected waiting time is

$$E[W] \to \frac{\log \frac{1}{1-\lambda}}{\log 2} \qquad \text{as } \lambda \to 1$$

Compared with the case $d = 1$, where $T_1 = \frac{1}{1-\lambda}$, the expected waiting time decreases exponentially.

2.2 Knowing Nothing about the Server

Now let us consider the scenario of knowing nothing about the servers. In this case, the central system does not query any information from the servers. Thus, the only work it can do is randomly select a server for all the arriving jobs. Using Fact 1, we know for each server the stream of arriving jobs is a Poisson process. Therefore, like the case in the *supermarket model*, we can draw a conclusion on each server that each server acts as an M/M/B/B loss model. And again, B is the amount of the resource in each server.

Consider an M/M/B/B loss model, where the queueing system has s servers and no buffer. We are interested in the blocking probability, i.e., the probability that an arriving job is blocked and lost due to the fact that there is no resource available to support the job. As in the case of the M/M/1 queue, the number of customers in the M/M/B/B queue evolves as a Markov chain, and is shown in Figure 2.5. The transition rates can be derived as in the case of the M/M/1 queue. Next, we will derive the stationary distribution $\boldsymbol{\pi}$ of the number of customers in the system for the M/M/B/B loss model.

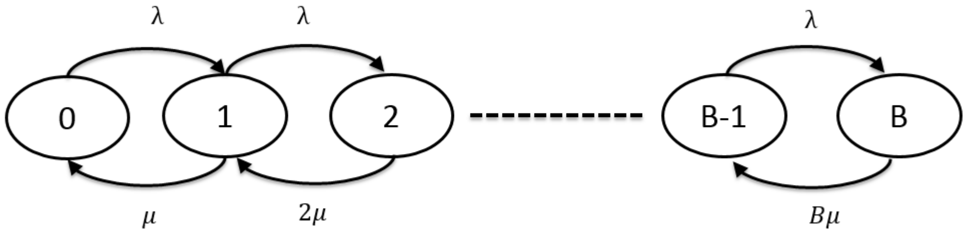

Figure 2.5: Markov chain for M/M/B/B loss model

Next, we will derive the stationary distribution of the number of jobs in a server for the M/M/B/B loss model. According to the local balance equation, we have that

$$\begin{aligned}\pi(1) &= \rho\pi(0)\\ \pi(2) &= \frac{1}{2}\rho\pi(1)\\ \pi(3) &= \frac{1}{3}\rho\pi(2)\\ &\vdots\\ \pi(B) &= \frac{1}{B}\rho\pi(B-1)\end{aligned}$$

Thus, for $0 \leq n \leq B$, we have

$$\pi(n) = \frac{\rho^n}{n!}\pi(0)$$

where, as before, $\rho = \frac{\lambda}{\mu}$

Using the fact that $\sum\limits_{k=0}^{B} \pi(k) = 1$, we obtain

$$\pi(0) = \frac{1}{\sum\limits_{k=0}^{B} \frac{\rho^k}{k!}}$$

and

$$\pi(n) = \frac{\frac{\rho^n}{n!}}{\sum\limits_{k=0}^{B} \frac{\rho^k}{k!}}$$

Thus, the probability that an arriving job is blocked and lost due to the fact that there is no resource available to support the job is given by

$$\pi(B) = \frac{\frac{\rho^B}{B!}}{\sum\limits_{k=0}^{B} \frac{\rho^k}{k!}}$$

We will present our main results in Chapter 3.

CHAPTER 3

MAIN RESULT

Let us review the system. We consider a system of N parallel servers. Each server has B units of a resource. Jobs arrive at the system according to a Poisson process with rate $N\lambda$, and each job requires 1 unit of a resource and stays in the system for an exponentially distributed amount of time with mean $\mu = 1$. Each arriving job is routed to a server according to a routing policy and requires zero-delay service. If the selected server has sufficient resources to accommodate the arriving job, the job will be processed immediately. Otherwise the job is blocked, i.e., it leaves the system immediately without being served. The goal is to study the blocking probability of the power-of-d-choices routing: under this routing scheme, upon each job arrival, d servers are selected uniformly at random and the job is routed to the least loaded of the servers (the one with the least amount of resource used). If none of the selected servers has a sufficient amount of the resource, then the arriving job is blocked and lost. There are three different scenarios:

(1) The arriving job knows the information about all of the servers. Whenever there is an arriving job, it will join the server with the least load, i.e., with largest available resource among all of the servers. If there are even loads among several servers, the job will select one of the servers uniformly at random.

(2) The arriving job knows the information about a portion of the servers. Whenever there is an arriving job, it will be assigned to d number of servers selected uniformly at random among all of the servers. Then, the job will join the server with the largest available resource among these d servers. If there is an even number among several servers, the job will select one of the servers uniformly at random.

(3) The arriving jobs know nothing about the servers. Whenever there

is an arriving job, it will join one of the servers selected uniformly at random among all of the servers.

In Chapter 2, we analyzed the third case of the model using the traditional queueing theory. Now let us consider the third case of the model in the fluid limits sense.

First consider one of the internal state i of the continuous time Markov chain (CTMC) as shown in Figure 3.1.

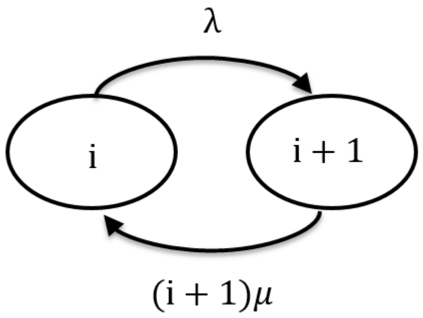

Figure 3.1: The internal state i in the Markov chain

Let $q(t)$ denote the number of customers in the system, which forms a time-homogeneous continuous time Markov chain. Let π_n be the steady-state probability that there are n customers in the system. For the state i, we have the local balance equation:

$$\lambda\pi_i = (\imath + 1)\mu\pi_{i+1}$$

and we know that the service rate $\mu = 1$ in our model. Moreover, we introduce a helper function

$$p(x) = \pi_{B-x\sqrt{B}}$$

and

$$x = \frac{B-i}{\sqrt{B}}$$

Therefore, we have

$$\pi_i = p(x)$$

Now let us scale the arrival rate λ, which is motivated by earlier analysis of loss models in [8], [9] as follows

$$\lambda = B - \gamma\sqrt{B}$$

Then the local balance equation can be rearranged as

$$(B - \gamma\sqrt{B})p(x) = (B - x\sqrt{B} + 1)p(x - \frac{1}{\sqrt{B}}) \tag{3.1}$$

where the last term above by the definition of $p(x)$ and x is

$$p(x - \frac{1}{\sqrt{B}}) = \pi_{B-\sqrt{B}\frac{B-i-1}{\sqrt{B}}} = \pi_{i+1}$$

Now by rearranging Equation (3.1), we have

$$Bp(x) - Bp(x - \frac{1}{\sqrt{B}}) = \gamma\sqrt{B}p(x) - (x\sqrt{B} - 1)p(x - \frac{1}{\sqrt{B}}) \tag{3.2}$$

Then divide $\sqrt{B}$ on both sides of Equation (3.2), and we get

$$\sqrt{B}\big(p(x) - p(x - \frac{1}{\sqrt{B}})\big) = \gamma p(x) - (x - \frac{1}{\sqrt{B}})p(x - \frac{1}{\sqrt{B}}) \tag{3.3}$$

$$\frac{p(x) - p(x - \frac{1}{\sqrt{B}})}{\frac{1}{\sqrt{B}}} = \gamma p(x) - (x - \frac{1}{\sqrt{B}})p(x - \frac{1}{\sqrt{B}}) \tag{3.4}$$

As $B \to \infty$, we have

$$\frac{dp(x)}{dx} = (\gamma - x)p(x)$$

Then rearrange the above equation, and we get

$$\frac{dp(x)}{p(x)} = (\gamma - x)dx \tag{3.5}$$

Now integrate Equation (3.5) on both sides, we have

$$\ln p(x) + C = \gamma x - \frac{x^2}{2}$$

where C is some constant. Then

$$p(x) = C_1 e^{\gamma x - \frac{x^2}{2}}$$

where $C_1 = e^C$, and C_1 can be determined by solving the equation,

$$\int\limits_{x=0}^{\infty} C_1 e^{\gamma x - \frac{x^2}{2}} = 1$$

and

$$C_1 = \frac{1}{\int\limits_{x=0}^{\infty} e^{\gamma x - \frac{x^2}{2}}}$$

For the blocking probability, we are interested in π_B, and we know $x = \frac{B-i}{\sqrt{B}}$. Then

$$\pi_B = p(0) = C_1 = \frac{1}{\int\limits_{x=0}^{\infty} e^{\gamma x - \frac{x^2}{2}}} \tag{3.6}$$

From Chapter 2, we know the blocking probability is

$$\pi(B) = \frac{\frac{\rho^B}{B!}}{\sum\limits_{k=0}^{B} \frac{\rho^k}{k!}}$$

And Equation (3.3) is a fluid approximation of it in a heavy traffic scaling of $\lambda = B - \gamma\sqrt{B}$.

Now let us consider the second case in our model. The arriving job knows the information about a portion of the servers. Whenever there is an arriving job, it will be assigned to d number of servers selected uniformly at random among all of the server. Then, the job will join the server with the largest available resource among these d servers. If there are even resources among several servers, the job will select one of the servers uniformly at random. For simplicity, let us consider the case $d = 2$. Since the number of servers is very large, we can assume the pairwise independence among the servers. Let $\boldsymbol{\pi} = (\pi_0, \pi_1, ...\pi_B)$ be the probability of occupancy of a server. Because of the symmetry, each server should have the same probability of occupancy. Without loss generosity, let us consider server 1. Then we can establish the following Markov chain shown in Figure 3.2.

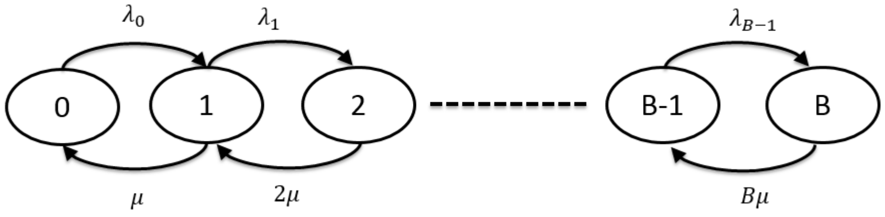

Figure 3.2: The Markov chain of our model

In order to calculate π, we need to find λ_0, λ_1,...,λ_{B-1}. And λ_i is given by the following:

$$\lambda_i = N\lambda P(\text{server 1 is selected}\big|\text{it has } i \text{ objects in it})$$

First consider the term $P(\text{server 1 is selected}\big|\text{it has } i \text{ objects in it})$:

$$\begin{aligned}&P(\text{server 1 is selected}\big|\text{it has } i \text{ objects in it}) = \\ &P(\text{server 1 is chosen and the other server has more than } i \text{ objects}) \quad + \\ &\frac{1}{2}P(\text{server 1 is chosen and the other also has } i \text{ objects}) \\ &= \frac{N-1}{\binom{N}{2}}\big(\sum_{j=i+1}^{B}\pi_j + \frac{1}{2}\pi_i\big) \\ &= \frac{2}{N}\big(\sum_{j=i+1}^{B}\pi_j + \frac{1}{2}\pi_i\big)\end{aligned}$$

Let $s_i = P(\text{the number of objects is more than } i) = \sum_{j=i}^{B}\pi_j$. Then we have

$$\begin{aligned}P(\text{server 1 is selected}\big|\text{it has } i \text{ objects in it}) &= \frac{2}{N}(s_{i+1} + \frac{1}{2}(s_i - s_{i+1})) \\ &= \frac{2}{N}(\frac{1}{2}(s_i + s_{i+1})) \\ &= \frac{1}{N}(s_i + s_{i+1})\end{aligned}$$

Therefore,

$$\lambda_i = N\lambda\frac{1}{N}(s_i + s_{i+1}) = \lambda(s_i + s_{i+1})$$

Now the Markov chain can be represented as shown in Figure 3.3

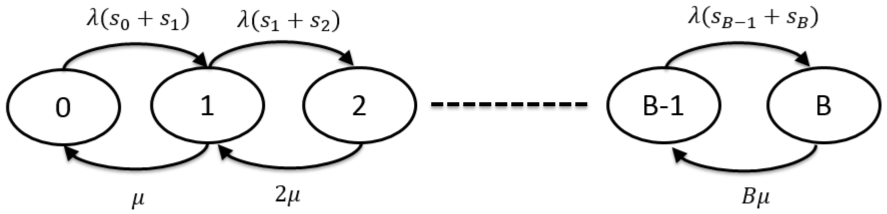

Figure 3.3: Updated Markov chain for our model

Given the local balance equation, we have

$$\lambda(s_i + s_{i+1})\pi_i = (i+1)\mu\pi_{i+1} \qquad \text{for } i \in \{0, 1, ..., B-1\}$$

Since $\pi_i = s_i - s_{i+1}$ for $i \in \{0, 1, ..., B-1\}$, then we have:

$$\lambda(s_i + s_{i+1})(s_i - s_{i+1}) = (i+1)\mu(s_{i+1} - s_{i+2}) \qquad \text{for } i \in \{0, 1, ..., B-2\}$$
$$\lambda(s_{B-1} + s_B)(s_{B-1} - s_B) = B\mu(s_B)$$

Define $\rho = \frac{\lambda}{\mu}$, and we can simplify the above recursive equations to be

$$\begin{aligned} s_0 &= 1 \\ (s_i^2 - s_{i+1}^2)\rho &= (i+1)(s_{i+1} - s_{i+2}) \qquad \text{for } i \in \{0, 1, ..., B-2\} \\ (s_{B-1}^2 - s_B^2)\rho &= Bs_B \end{aligned}$$

For the heavy traffic, we scale $\lambda = B - \gamma\sqrt{B}$, $x = \frac{B-i}{\sqrt{B}}$, and define $g(x) = s_i$. Then the recursive equation becomes

$$\lambda(s_{i-2}^2 - s_{i-1}^2) = (i-1)(s_{i-1} - s_i)$$
$$(B - \gamma\sqrt{B})(g^2(x + \frac{2}{\sqrt{B}}) - g^2(x + \frac{1}{\sqrt{B}})) = (B - x\sqrt{B} - 1)(g(x + \frac{1}{\sqrt{B}}) - g(x))$$

divided by B on both side and let $B \to \infty$

$$(1 - \frac{\gamma}{\sqrt{B}})\frac{dg^2(x)}{dx} = (1 - \frac{x}{\sqrt{B}} - \frac{1}{B})\frac{dg(x)}{dx}$$
$$\frac{dg^2(x)}{dx} = g'(x) \qquad \text{since } B \to \infty$$
$$2g\frac{dg(x)}{dx} = g'(x)$$

The result is:

$$g'(x) = 0 \quad \text{or} \quad g(x) = \frac{1}{2}$$

Given by the result above, we can expected the heuristic distribution of s_i should be 1 at the beginning and kept 1 for amount of time and then suddenly jumps to 0 and remains 0 forever. In order to check our heuristic, we numerically solve the recursive equation and also simulate the stochastic system and calculate the distribution for s_i. The graph in Figure 3.4 shows our numerical result and simulation result.

Note: There is an equilibrium tail distribution for the case $d = 2$ with server capacity $B = 50$ at three different loads. The values for the stationary point are obtained numerically by solving the recursive equation. Simulation results are from a finite system with $N = 500$.

Now we have shown that the stochastic system can be approximated by the fluid model. Then the blocking probability p_d can be approximated by the solution of the fixed point equation. The numerical result has been shown in Table 3.1.

Moreover, as we have shown in the previous analysis of the fluid limits, we use a square root gap in the heavy traffic limit, i.e., $\lambda = B - \gamma\sqrt{B}$. We also do the simulation on the log gap, i.e., $\lambda = B - \gamma \log B$, and make a comparison of the numerical result and simulation result both for the square root gap and the log gap.

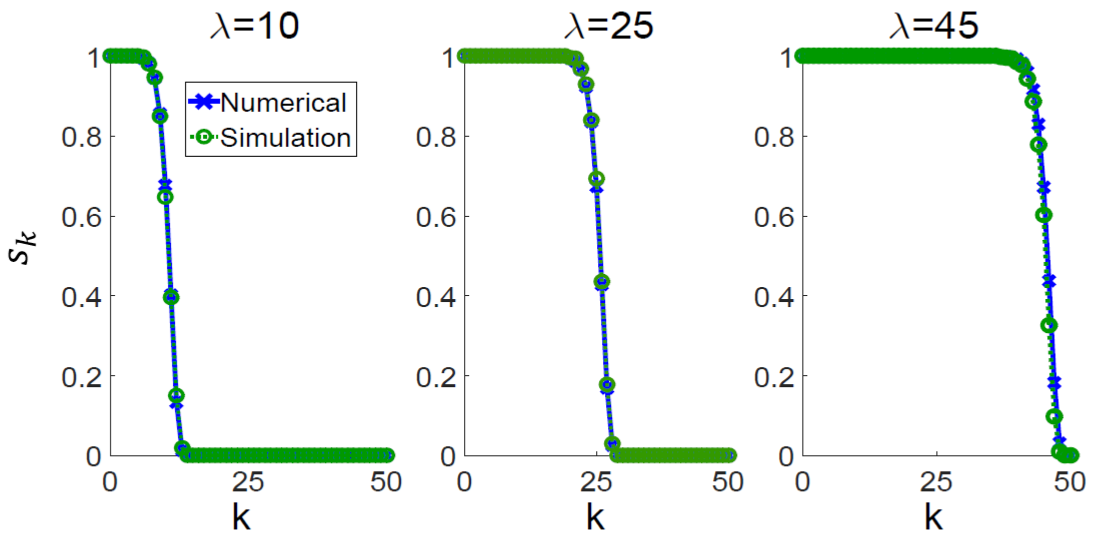

Figure 3.4: The simulation results for different arrival rates

Table 3.1: The blocking probability for the power-of-two-choices policy with $B = 50$ at different load.

$\rho = \lambda/B$	Fluid limit
0.6	0
0.8	0
0.84	0.0000
0.88	1.873×10^{-25}
0.9	5.229×10^{-13}
0.92	8.240×10^{-7}
0.94	7.854×10^{-4}

Figure 3.5 shows the blocking probability for the power-of-two-choices algorithm with $B - \lambda = \sqrt{\lambda}$ and $B - \lambda = 2\log\lambda$, both by solving the recursive equation above numerically and by simulating a finite system with $N = 1000$. Note that the y-axis is in log scale. We can see that even for small B, the blocking probability P_b exhibits qualitatively different behavior in these two regions: with $\log\lambda$ load gap, P_b decays exponentially; while for $\sqrt{\lambda}$ load gap, P_b decays much faster. For $B = 30$, P_b is of order 10^{-10} with $\sqrt{\lambda}$ load gap. It requires very long simulation time in order to observe a blocking event. We simulated around 10^{10} arrivals and no job blocking was observed for $B \geq 30$.

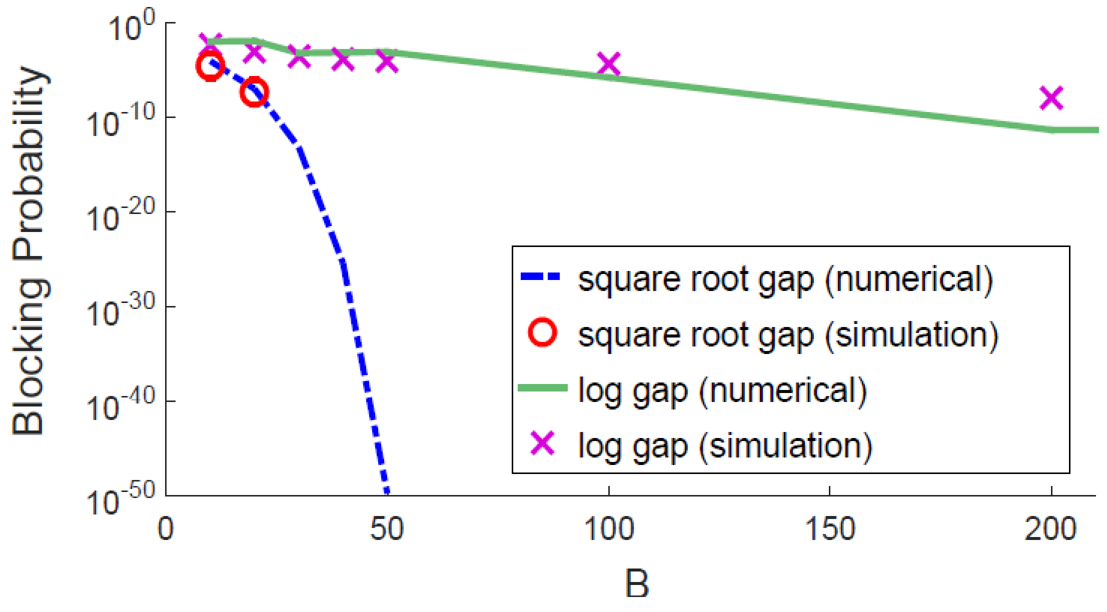

Figure 3.5: The simulation results for different capacities with different scaling methods

CHAPTER 4

CONCLUSION

In our thesis, we considered a loss model for the virtual machine assignment problem in a cloud system. The overall goal is to study how to route arriving jobs to the servers in order to minimize the probability that an arriving job does not find the required number of resources in the system. Using the fluid model approach, we showed that when arrivals are routed to the least utilized of d randomly selected servers, the blocking probability decays exponentially or doubly exponentially. This is a substantial improvement over the random policy.

In our analysis, we just demonstrate the case that all the jobs use the same amount of a resource. For further consideration, we could consider the case in which we can have more than one type of job. Further, the underline system can form a two-dimensional Markov chain as shown Figure 4.1.

Figure 4.1 shows the state transition rate diagram for one of the servers with B units of a resource and two types of job arrivals. We can use some mathematical tools to form a one-dimensional recursion equation for it. Further, some other scenarios have partially been considered in [10].

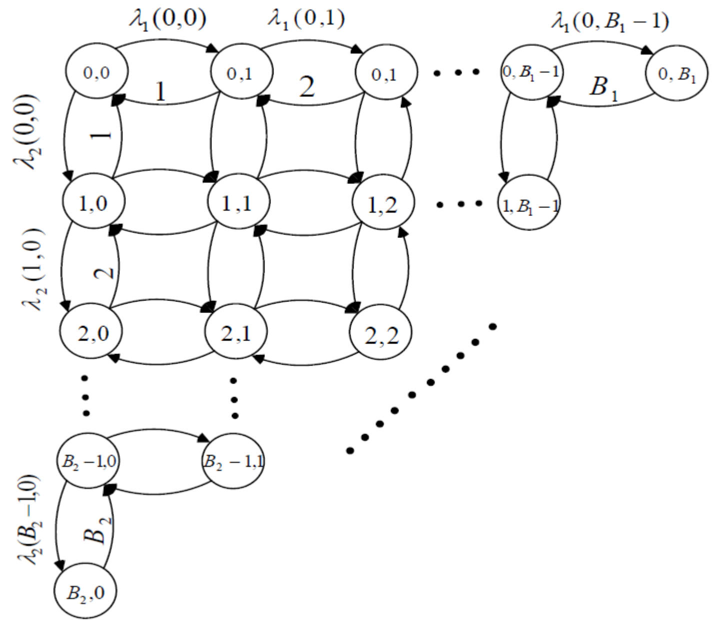

Figure 4.1: The Markov chain for two different jobs

REFERENCES

[1] "Amazon EC2," http://aws.amazon.com/ec2/.

[2] "Google App Engine," https://cloud.google.com/appengine/docs?csw=1.

[3] "Azure," http://azure.microsoft.com/en-us/.

[4] "Roundup Of Cloud Computing Forecasts And Market Estimates," http://www.forbes.com/sites/louiscolumbus/2015/01/24/roundup-of-cloud-computing-forecasts-and-market-estimates-2015/.

[5] M. Mitzenmacher, "The power of two choices in randomized load balancing," Ph.D. dissertation, University of California, Berkeley, 1996.

[6] N. D. Vvedenskaya, R. L. Dobrushin, and F. I. Karpelevich, "Queueing system with selection of the shortest of two queues: An asymptotic approach," *Probl. Peredachi Inf.*, vol. 32, no. 1, pp. 20–34, 1996.

[7] R. Srikant and L. Ying, *Communication Networks: An Optimization, Control, and Stochastic Networks Perspective.* Cambridge University Press, 2014.

[8] A. A. Borovkov, *Stochastic Processes in Queueing Theory.* Springer, 1976.

[9] W. Whitt, "Heavy-traffic approximations for service systems with blocking," *AT&T Bell Laboratories Technical Journal*, vol. 63, no. 5, pp. 689–708, 1984. [Online]. Available: http://dx.doi.org/10.1002/j.1538-7305.1984.tb00102.x

[10] Y. L. Q. Xie, X. Dong and R. Srikant, "Power of d choices for large-scale bin packing: A loss model," *ACM SIGMETRICS*, 2015.