Chapter 3

Dynamic Allocation of Virtual Machines in Cloud Center

3.1 Introduction

One of the important requirements of a cloud center in a cloud computing environment is to render reliable Quality of Services (QoS) [44, 45]. Sluggish access to data, web pages and applications or poor performance may discourage customers. Service provider's reliability is a direct reflection of its QoS. Therefore, we need an efficient model to provide a tradeoff between the performance and resource utilization [11]. QoS in terms of Service Level Agreement(SLA) has the characteristics of minimal throughput, maximal response time or latency delivered by the deployed system [9]. Virtual machines (VMs) share the same physical computing resources by isolating each other through virtualization technologies [46]. However, due to variability of the workload, some VMs may not get the required amount of resources as requested. This may lead to performance loss in terms of increased response time. Therefore, cloud service providers have to deal with those applications which need more resources during the busy period. To achieve the aim of dynamic scaling, cloud controllers need proper tools and models to find the runtime requirements of web applications [47].

The number of cloud applications and cloud providers are increasing firmly due to the flexibility, scalability and ease of maintenance. As a result, computing resource scheduling and performance management have been one of the most important aspects of cloud computing. When a number of web applications are distributed into a cloud environment, dynamic allocation of the computing resources on demand to the web applications has a positive effect not only on the performance of web applications, but also on the energy saving. The solution to eliminate this obstacle is to automatically scale up and down without violating service level agreements (SLA) in response to load the web application providers by optimizing the computing resource requests [48, 49].

In this chapter, we proposed an analytical queueing based model for performance management on cloud. A Schematic representation for a multi server queueing system is shown in Figure 3.1. Each client request to a web application waits in a queue if all the VMs are busy. The virtual machines are modeled as service centers. A queueing theory model has been applied to analyze the need to dynamically create and remove virtual machines in order to implement scaling up and down.

3.2 System Model

To tackle with the problems such as obstructing the smooth provisioning and delivery of application services mentioned in chapter 2 (uncertain behavior, estimation error and dynamic workload), we proposed an adaptive provisioning mechanism. The high level architecture of the approach is shown in Figure 3.2. The following components are decisive to the overall functionality of the system (i) Application provisioner, which receives information about the accepted requests from workload analyzer, load predictor, performance modeler and provisions virtual machines and application instances based on the input; (ii) Workload analyzer, which estimates the future demands for the application. This information is communicated to the load predictor and

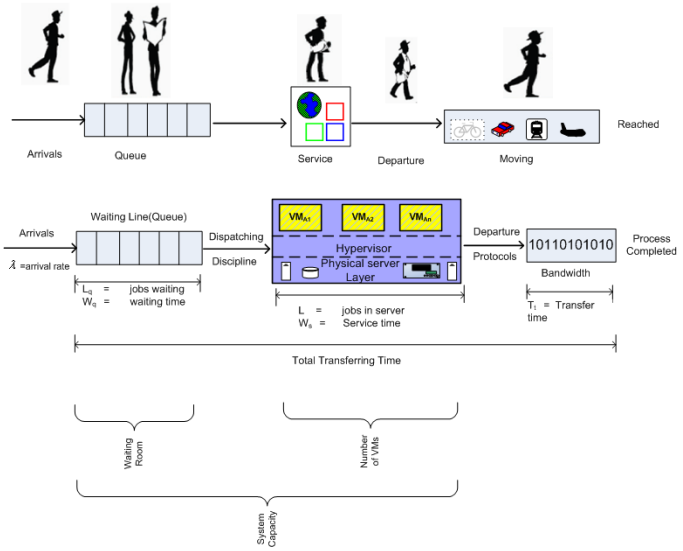

Figure 3.1: A Schematic representation for a multi server queueing system.

performance modeler component; (iii) Load predictor and performance modeler, which decides the number of VM instances that should be allocated to an application.

Performance is the response time of the web application for an individual user. Response time is a measure of amount of time the application consumes while processing a client request [50]. When a web application is deployed on a cloud, the cloud controller as the portal of the cloud establishes a queue to hold the client requests. In a single server process, the cloud computing user (CCU) after detecting the server busy, enters the buffer and waits in the queue for its turn. In a multi server cloud environment the load balancer detects the lightly loaded VM in the cloud system and the request is forwarded to the VM. If the VM is busy, the newly assigned request waits in the queue. When the queue length exceeds a specified limit, a new VM is dynamically provisioned by Cloud Controller on a cloud node. The number of initially created VMs can be specified by the SLA. The number of live VMs are dynamically created or removed at runtime depending on the size of the waiting queue.

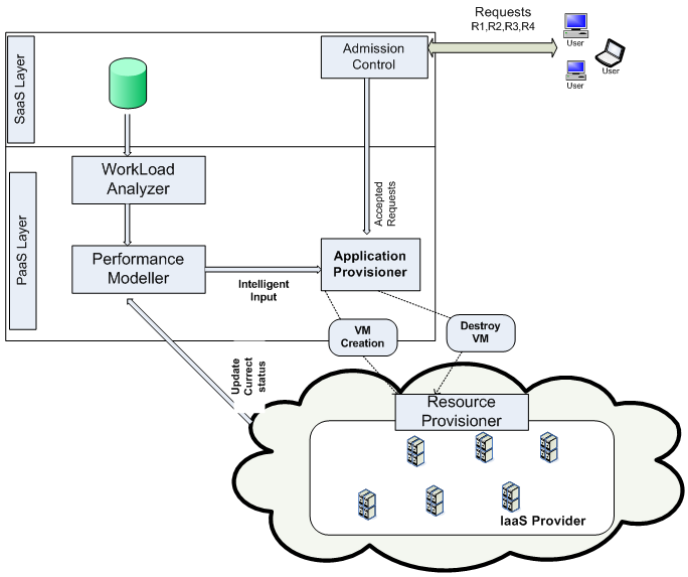

Figure 3.2: System model.

Either a single cloud node may contain all the VMs of the web applications or they may be contained in multiple cloud nodes. The VMs may be allocated with the different computing resources. In each of the VMs, an instance of the web application runs irrespective of the physical location of the VM. As a result, a cluster is formed consisting of all the VMs and processing the requests in the corresponding queue.

When the cloud controller receives a request from a client, the dispatcher in the cloud forwards the request to the queue of the specified web application. The web application running in the VMs act as service centers to process the requests in the queue. The process of client requests of a web application in a cloud has the following properties:

- There is a common probability distribution in the inter-arrival times between any two successive client requests and are independent of each other .

- If in a specific time interval the requests are processed by the web application the clients will receive response otherwise the cloud controller

may abort the client request or may send exception(s) due to the time out of waiting.

- The client requests are processed in any one of the possible organizations, such as first come first served (FCFS), last come first served (LCFS), shortest job first (SJF), random order, round robin and so on. However, the prevailing one is the first come first served.

- The cloud controller can create a single VM or a cluster of VMs to process the client requests. Therefore the application capacity varies depending on the active available VMs in the system.

- The number of waiting requests in the application is limited as the buffer(s) of the VMs is finite. Because of this limitation the service requests are dropped from the application when the waiting room is fully occupied.

3.3 Analysis of the Model

We consider a finite buffer queueing system with queue dependent heterogeneous VMs server [51]. We assume that the inter-arrival times of the client requests are independent and exponentially distributed with mean arrival rate λ. There is a provision of r heterogeneous VMs in the system that turn on one by one according to a defined rule and provide service according to an exponential distribution with mean service rate μ_j for j^{th} $(1 \leq j \leq r)$ server [52, 53]. The waiting space is finite buffer of size K. Each web application on cloud has one or multiple instance running on the VMs, and each instance of VM can serve a certain number of requests. Since the web applications are modeled as queues and the VMs are modeled as service centers, we can dynamically create and remove VMs according to the number of necessary service centers in order to scale up and down. Because the cloud controller is the portal of the cloud, all the requests will be sent to dispatcher and forward to queues, and all responses will be sent back to clients via dispatcher. As a

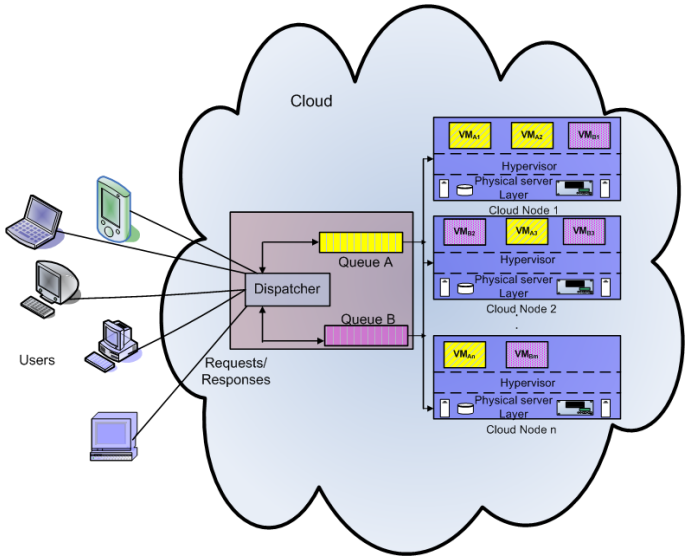

Figure 3.3: Queue Model of Web Application on Cloud.

result the dispatcher can record the response time exactly. The queue model of web application on cloud is shown in Figure 3.3. The number of VMs used depends upon the number of client requests present in the system according to a threshold policy as follows:

- The first VM is permanently available in the system.
- When the number of client requests waiting in the buffer reaches to N_1, the second VM will begin its service. But it will be dynamically took out its service and removed from the cloud system when the length of the buffer will become less than N_1.
- Whenever the size of the waiting client requests in the system attains a particular level N_{j-1}, the j^{th} $(j = 2, 3, \ldots, r)$ VM will commence its service. Whenever the buffer length again reduces to less than N_{j-1}, the j^{th} VM will be took out from the cloud system.

For mathematical analysis we use Markovian process to model the system behavior. Figure 3.4 gives the sate transition diagram for a multi server queueing model. We define P_n be the steady state probability such that

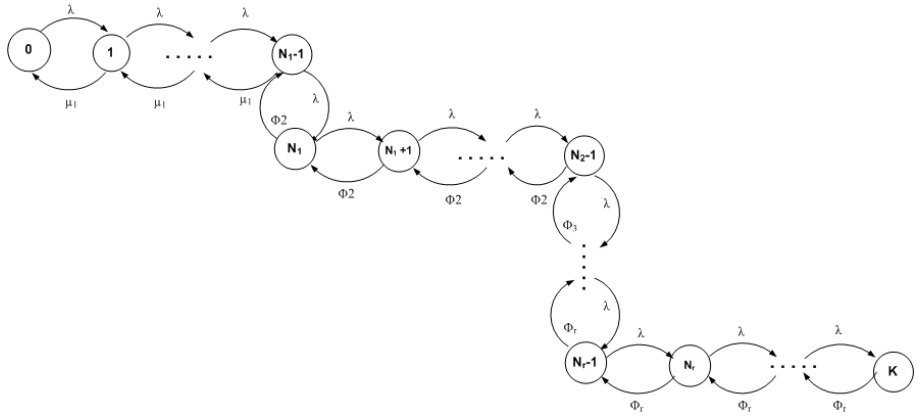

Figure 3.4: Sate transition diagram for a multi server queueing model.

there are n client requests in the cloud system for an application. The steady-state equations for finite buffer multi-server queueing system with r queue dependent heterogeneous VMs are given by

$$\begin{aligned}
\lambda P_0 &= \mu_1 P_1, && (3.1)\\
(\lambda+\mu_1)P_n &= \lambda P_{n-1} + \mu_1 P_{n+1}, \\
&\quad 1 \le n \le N_1 - 2, && (3.2)\\
(\lambda+\phi_j)P_{N_j-1} &= \lambda P_{N_j-2} + \phi_{j+1} P_{N_j}, \\
&\quad 1 \le j \le r-1, && (3.3)\\
(\lambda+\phi_j)P_n &= \lambda P_{n-1} + \phi_j P_{n+1}, 2 \le j \le r-1, \\
&\quad N_{j-1} \le n \le N_j - 2, && (3.4)\\
(\lambda+\phi_r)P_n &= \lambda P_{n-1} + \phi_r P_{n+1}, \\
&\quad N_{r-1} \le n \le K-1, && (3.5)\\
\phi_r P_K &= \lambda P_{K-1}. && (3.6)
\end{aligned}$$

where $\phi_j = \sum_{i=1}^{j} \mu_i$. We obtain steady state queue size by solving equations $(3.1) - (3.6)$ recursively as

$$\begin{aligned}
P_n &= \rho_1^n P_0, 1 \le n \le N_1 - 1, && (3.7)\\
P_n &= \left\{ \prod_{i=1}^{j-1} \rho_i^{N_i - N_{i-1}} \right\} \rho_j^{n - N_{j-1}+1} P_0,
\end{aligned}$$

$$j = 2, 3, \ldots, r-1, N_{j-1} \leq n \leq N_j - 1, \tag{3.8}$$

$$P_n = \left\{ \prod_{i=1}^{r-1} \rho_i^{N_i - N_{i-1}} \right\} \rho_r^{n - N_{r-1} + 1} P_0,$$

$$N_{r-1} \leq n \leq K, \tag{3.9}$$

where $N_0 = 1$, and $\rho_j = \lambda/\phi_j$.

Now P_0 can be obtained by using the normalization condition $\sum_{i=0}^{N_1-1} P_i + \sum_{i=1}^{r-1} \sum_{j=N_{i-1}}^{N_i-1} P_i + \sum_{i=N_{r-1}}^{K} P_i = 1$. Hence,

$$P_0 = \left[\frac{1-\rho_1^{N_1}}{1-\rho_1} + \sum_{i=2}^{r-1} \prod_{j=1}^{i-1} \rho_j^{N_j - N_{j-1}} \rho_i \left(\frac{1-\rho_i^{N_i - N_{i-1}}}{1-\rho_i} \right) + \prod_{j=1}^{r-1} \rho_j^{N_j - N_{j-1}} \rho_r \left(\frac{1-\rho_r^{K - N_{r-1}+1}}{1-\rho_r} \right) \right]^{-1}. \tag{3.10}$$

System characteristics:

Some performance measures using steady-state queue size distribution are as follows.

- Probability that the j^{th} $(j = 1, 2, \ldots, r-1)$ VMs is operating in the system $(P(j))$, is given by

$$P(j) = Prob\{N_{j-1} \leq n \leq N_j - 1\} = \prod_{i=1}^{j-1} \rho_i^{N_i - N_{i-1}} \rho_j \left(\frac{1-\rho_j^{N_j - N_{j-1}}}{1-\rho_j} \right) P_0.$$

- Probability that only one VM is operating in the system $(P(1))$, is given by

$$P(1) = Prob\{1 \leq n \leq N_1 - 1\} = \sum_{n=1}^{N_1-1} P_i = \sum_{n=1}^{N_1-1} \rho_1^n P_0 = \begin{cases} \frac{\rho_1(1-\rho_1^{N_1-1})P_0}{1-\rho_1}, & if \rho_1 \neq 1, \\ (N_1 - 1)P_0, & if \rho_1 = 1 \end{cases}$$

- Probability that all r VMs are operating in the system

$$\begin{aligned} P(r) &= Prob\{N_{r-1} \leq n \leq K\} \\ &= \prod_{i=1}^{r-1} \rho_i^{N_i - N_{i-1}} \rho_r \left(\frac{1 - \rho_r^{K - N_{r-1}+1}}{1 - \rho_r} \right) P_0. \end{aligned}$$

- Probability that j^{th} $(j = 1, 2, \ldots, r)$ VMs being in busy state is

$$P_B(j) = \sum_{i=j}^{r} P(i).$$

Let $L[r : N_1, N_2, \ldots, N_{r-1}]$ denote the average number of client requests in the cloud system with r VMs, which starts its service at the threshold levels $N_1, N_2, \ldots, N_{r-1}$, respectively. Then the expression for $L[r : N_1, N_2, \ldots, N_{r-1}]$ is obtained as follows, $L[r : N_1, N_2, \ldots, N_{r-1}] = \sum_{n=0}^{K} nP_n$

$$\begin{aligned} = & \left[\frac{\rho_1 \{1 - N_1 \rho_1^{N_1 - 1} + (N_1 - 1)\rho_1^{N_1}\}}{(1 - \rho_1)^2} \right. \\ & + \sum_{j=2}^{r-1} \left\{ \prod_{i=1}^{j-1} \rho_i^{N_i - N_{i-1}} \right\} \left\{ \frac{N_{j-1}\rho_j (1 - \rho_j^{N_j - N_{j-1}})}{1 - \rho_j} + \right. \\ & \left. \left. \frac{\rho_j^2 \{1 - \rho_j^{N_j - N_{j-1}} - (N_j - N_{j-1})\rho_j^{N_j - N_{j-1} - 1}(1 - \rho_j)\}}{(1 - \rho_j)^2} \right\} \right. \\ & + \prod_{i=1}^{r-1} \rho_i^{N_i - N_{i-1}} \left\{ \frac{N_{r-1}\rho_r (1 - \rho_r^{K - N_{r-1}+1})}{1 - \rho_r} + \right. \\ & \left. \left. \frac{\rho_r^2 \{1 - \rho_r^{K - N_{r-1}} (1 + (K - N_{r-1})(1 - \rho_r))\}}{(1 - \rho_r)^2} \right\} \right]. \end{aligned}$$

3.4 Numerical Illustration

This section illustrates the numerical tractability of the optimal threshold policy provided. We developed a computational program using MATLAB. For the validation of the analytical results, the numerical results for the following two models have been computed.

Model 1: In this model, the homogeneous VMs turn on one by one with

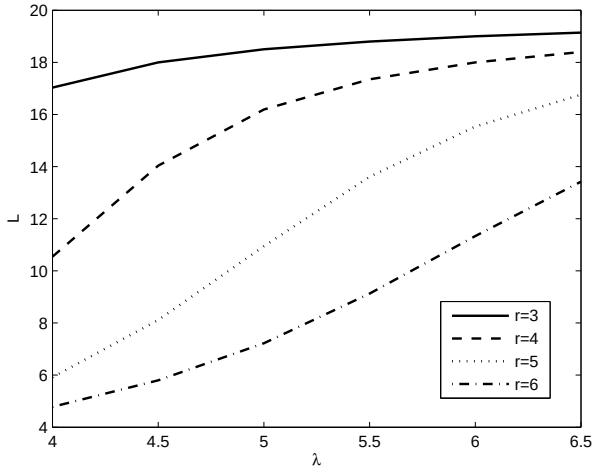

Figure 3.5: Impact of L on λ for model I.

the arrival of each client request. That is, $N_i = i + 1, i = 1, 2, \ldots, r - 1$. Also,service rate has been considered as $\mu_i = 1, i = 1, 2, \ldots, r$.

Model 2: In this case, the heterogeneous VMs turn on one by one with the arrival of each client request. That is, $N_i = i + 1, i = 1, 2, \ldots, r - 1$. The service rate has been taken as $\mu_i = 1 + 0.2(i - 1), i = 1, 2, \ldots, r$.

Mathematical results for several system performances are presented in Tables 3.1-3.4. Tables 3.1 and 3.2 show the probability for the $j^{th}(j = 2, \ldots, r)$ server being busy by fixing capacity $K = 20$ for model I and model II, respectively. It is observed that the probability that the j_{th} VM being busy, $P_B(j)$ increases with arrival rate λ but it decreases as the the number of VMs increases for both the models. Tables 3.3 and 3.4 depict the probability for the $j^{th}(j = 2, \ldots, r)$ server being busy by varying λ and K for model I and model II, respectively. It is seen that the probability for j^{th} VM being busy increases with capacity size K for both the models.

Figures 3.5 and 3.6 depict the average number of customer requests (L) in the system versus arrival rate λ by varying the number of servers for model I and model II, respectively. It is seen that the average number of customers (L) in the system increases with the arrival rate λ. But it decreases by

Table 3.1: Probabilities for j^{th} VM being busy for Model I by varying λ and r.

r	λ	$P_B(2)$	$P_B(3)$	$P_B(4)$	$P_B(5)$	$P_B(6)$
4	4	0.987058	0.971567	0.949432		
4	4.5	0.991051	0.974576	0.949864		
4	5	0.997820	0.993279	0.985709		
4	5.5	0.999478	0.998262	0.996033		
4	6	0.999868	0.99953	0.998854		
4	6.5	0.999964	0.999864	0.999647		
5	4	0.936105	0.871347	0.754951	0.657234	
5	4.5	0.965465	0.901888	0.806524	0.699239	
5	5	0.971505	0.921781	0.871608	0.750345	
5	5.5	0.994510	0.981736	0.958317	0.926115	
5	6	0.998202	0.993577	0.984329	0.970456	
5	6.5	0.999430	0.997823	0.994342	0.988685	
6	4	0.943500	0.843451	0.715355	0.557670	0.443212
6	4.5	0.951690	0.862756	0.729356	0.579280	0.444211
6	5	0.976760	0.887688	0.774358	0.598760	0.456654
6	5.5	0.985061	0.950300	0.886572	0.798945	0.702556
6	6	0.998202	0.963577	0.934329	0.840456	0.825540
6	6.5	0.996989	0.988508	0.970133	0.940272	0.901454

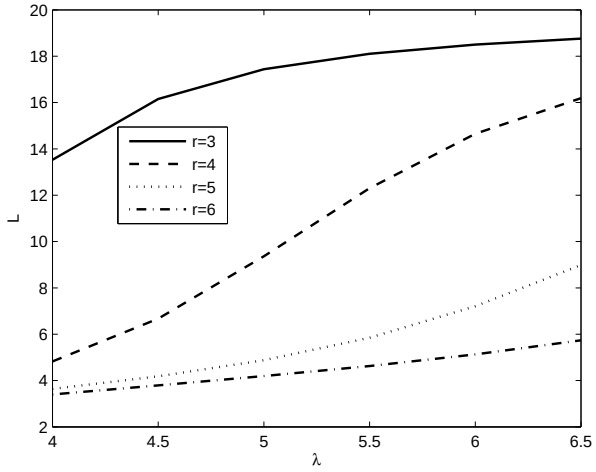

Figure 3.6: Impact of L on λ for model II.

Table 3.2: Probabilities for j^{th} VM being busy for Model II by varying λ and r.

r	λ	$P_B(2)$	$P_B(3)$	$P_B(4)$	$P_B(5)$	$P_B(6)$
4	4	0.887173	0.723062	0.540715		
4	4.5	0.931521	0.816917	0.673662		
4	5	0.964993	0.898691	0.806605		
4	5.5	0.985337	0.954319	0.906929		
4	6	0.994720	0.982377	0.961805		
4	6.5	0.998227	0.993686	0.985488		
5	4	0.866105	0.671347	0.454951	0.288492	
5	4.5	0.903271	0.741389	0.539037	0.363924	
5	5	0.931505	0.801781	0.621608	0.448364	
5	5.5	0.953196	0.854187	0.702923	0.542932	
5	6	0.969808	0.899230	0.781599	0.645871	
5	6.5	0.982048	0.936079	0.853080	0.749331	
6	4	0.857527	0.650293	0.420033	0.242911	0.141698
6	4.5	0.895287	0.720044	0.500990	0.311424	0.189560
6	5	0.923204	0.777757	0.575747	0.381507	0.242764
6	5.5	0.943882	0.825171	0.643806	0.451979	0.301257
6	6	0.959272	0.864063	0.705382	0.522288	0.365351
6	6.5	0.970795	0.896014	0.760992	0.592214	0.435492

Table 3.3: Probabilities for j^{th} VM being busy for Model I by varying λ and K.

K	λ	$P_B(2)$	$P_B(3)$	$P_B(4)$	$P_B(5)$
20	4	0.936105	0.871347	0.754951	0.657234
20	4.5	0.965465	0.901888	0.806524	0.699239
20	5	0.971505	0.921781	0.871608	0.750345
20	5.5	0.994510	0.981736	0.958317	0.926115
20	6	0.998202	0.993577	0.984329	0.970456
20	6.5	0.999430	0.997823	0.994342	0.988685
30	4	0.928712	0.871108	0.794656	0.687460
30	4.5	0.968706	0.911098	0.824684	0.727470
30	5	0.978756	0.923095	0.864687	0.857470
30	5.5	0.998098	0.993673	0.985559	0.974404
30	6	0.999716	0.998985	0.997523	0.995330
30	6.5	0.999959	0.999843	0.999591	0.999182
40	4	0.938212	0.899767	0.814530	0.690454
40	4.5	0.969698	0.913915	0.830240	0.736106
40	5	0.998975	0. 939923	0.996837	0.890653
40	5.5	0.999294	0.997652	0.994642	0.990503
40	6	0.999954	0.999837	0.999601	0.999248
40	6.5	0.999997	0.999989	0.99997	0.999941

Table 3.4: Probabilities for j^{th} VM being busy for Model II by varying λ and K.

K	λ	$P_B(2)$	$P_B(3)$	$P_B(4)$	$P_B(5)$
20	4	0.866105	0.671347	0.454951	0.288492
20	4.5	0.903271	0.741389	0.539037	0.363924
20	5	0.931505	0.801781	0.621608	0.448364
20	5.5	0.953196	0.854187	0.702923	0.542932
20	6	0.969808	0.899230	0.781599	0.645871
20	6.5	0.982048	0.936079	0.853080	0.749331
30	4	0.866109	0.671360	0.454971	0.288518
30	4.5	0.903300	0.741468	0.539178	0.364119
30	5	0.931642	0.802176	0.622362	0.449464
30	5.5	0.953689	0.855724	0.706054	0.547750
30	6	0.971166	0.903761	0.791421	0.661797
30	6.5	0.984689	0.945485	0.874700	0.786218
40	4	0.866110	0.671360	0.454971	0.288518
40	4.5	0.903301	0.741469	0.539179	0.364121
40	5	0.931647	0.802189	0.622388	0.449502
40	5.5	0.953733	0.855860	0.706332	0.548177
40	6	0.971441	0.904679	0.793409	0.665021
40	6.5	0.985693	0.949058	0.882911	0.800228

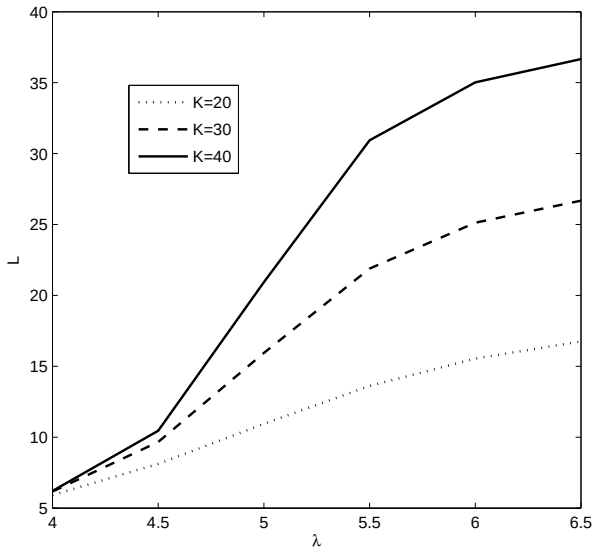

Figure 3.7: Impact of L on λ for Model I.

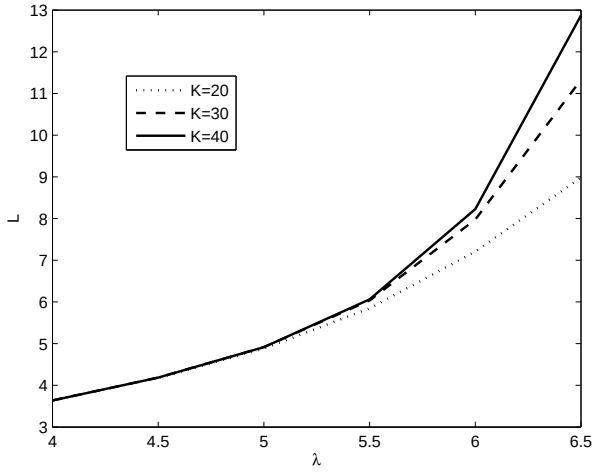

Figure 3.8: Impact of L on λ for Model II.

increasing the number of servers for both the models. It is to noted that as additional server being employed L decreases as λ increases.

Figures 3.7 and 3.8 illustrate the average number of customer requests (L) in the system versus arrival rate λ by varying the capacity size K for model I and model II, respectively. It is observed that the average number of customers (L) in the system increases monotonically with the increase of arrival rate λ. It can be seen that for fixed λ the average number of customers increases as K increases for both the models.

3.5 Conclusion

Cloud computing has emerged as a highly cost-effective computation paradigm for IT enterprise applications, scientific computing, and personal data management. In this chapter, we have proposed a queueing model for studying the performance of computer services in cloud computing. In this model, the virtual machines are taken as service centers and the web applications are modeled as queues. We employed the finite buffer multiserver queueing system with queue dependent heterogeneous servers, the number of servers changes depending on the queue length. The service load in cloud computing is dynamically scaled up and down depending upon end users service requests. Steady state queue size distribution is obtained using a recursive method assuming Markovian arrival and service times. It has been shown that our queueing based model is effective in the web applications on cloud and that no VM live migration is involved. It will be useful in the service performance prediction of cloud computing system.