Chapter 5

QoS of Cloud Centers With Different Arrival Modes

A cloud is a parallel as well as distributed system consisting of a collection of interconnected virtualized computers. The virtualized computers can be dynamically provisioned based on the service-level agreements(SLA) between the service provider and consumer. The requests are processed in the cloud centers in different modes depending on the SLA. For the commercial success of this new computing paradigm, the ability to deliver guaranteed Quality of Services (QoS) is crucial. This chapter analyzes a finite-buffer multi-server queuing system where client requests have two different arrival modes. Various performance measures are obtained and optimal cost policy is presented with numerical results. A genetic algorithm is employed to search the optimal values of various parameters for the system.

5.1 Introduction

Cloud computing technology has been developed from the concepts of virtualization, utility computing, Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS), etc [66]. A user can hire all the services such as software, hardware, data and information from the

cloud [67]. The cloud computing platform can be subdivided into three layers. SaaS delivers the software through web browsers as a service of cloud computing platform. PaaS provides one platform for the users and developers with application development, test and deployment [68, 69].

There are some QoS parameters to be considered in cloud computing environment, such as time, cost, reliability and trust. In particular, QoS requirements are not static and need to be updated dynamically over the time due to continuous changes in the operating environments. There must be greater importance given to users' time as they pay for using services from the clouds based on time. In this chapter, we examine various system measures when two categories of client requests come to the cloud center, one may be from the premier subscribers and other from the ordinary subscribers. We analyze the cloud model using a finite-buffer multi-server queuing system where client requests have two arrival modes. We assumed that each categories of request is served by one or more VMs, and the different categiries have equal probabilities of getting service. The inter-arrival and service-times are exponentially distributed. To minimizes the total expected cost a cost model has been developed to determine the optimum number of VMs and the optimal system capacity. The numerical searching approach for the cost function is implemented using a genetic algorithm. A genetic algorithm is a search technique used in computing to find exact or approximate solutions to optimization and search problems. Various performance measures are provided and sensitivity analysis is carried out.

5.2 System Description

Cloud computing delivers infrastructure, platform and software as services. These services are made available using pay-as-you-use model to customers, regardless of their location. Based on the priority of the client request the requests are classified into two different modes; High Priority jobs (category 1) and low priority jobs (category 2). Whenever the request for the service

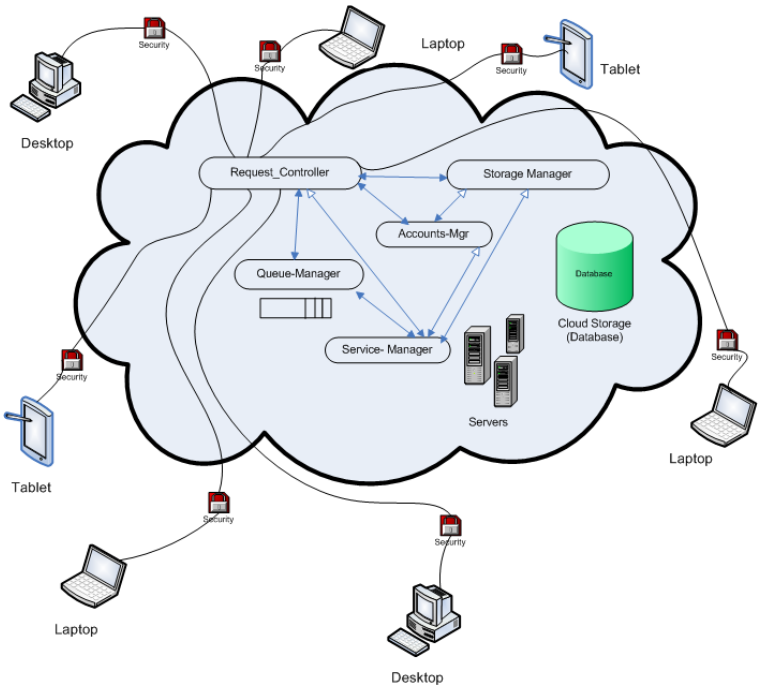

Figure 5.1: Cloud Architecture to support scheduling.

from the user i ($service-request_i$) arrives at the cloud, the Request controller estimates the service time ST^i_{est} required to complete that $service-request_i$ based on the type of the service request and the average time required for the type of service request.

Figure 5.1 shows the system architecture of the model supporting Priority, subscription-category (SUB_{CAT}) and SLA based service scheduling in Clouds and data centers. There are basically five main components involved in this design [70]:

- Users: Users can submit their requests for the jobs from anywhere in the world to the cloud center through the secured connection.

- Request-Controller (Req-Controller): The Req-Controller act as an interface between the cloud service provider and the users. In order to schedule the services, it requires the interaction with the other entities to support priority, admission control and subscription category based

service management.

Every user has to subscribe to the cloud before accessing it for submission of service request. During this subscription phase SLA is signed between the cloud service provider and the user. The Req-controller analyzes the submitted request for QoS requirements based on the priority. When the service request is first submitted, the Req-controller checks whether the service request can be admitted to the cloud or not by computing the tolerable delay for that request using the following algorithm.

$$delay_i^T = (DL_i^T - C_i^T),$$

where the current time, deadline given by $service-request_i$ and maximum tolerable time of i^{th} service-request are C_i^T, DL_i^T, $delay_i^T$, respectively. Req-controller also estimates the total servicing time required, ST_{est}^T, for all the client requests in the queue (Y):

$$ST_{est}^T = \sum_{i=1}^{Y} ST_{est}^i$$

Algorithm 5.1 Algorithm to Check whether a request can be accepted

if $delay_i^T \geq ((ST_{est}^T/c) + ST_{est}^i)$ **then**
  admit the service-request;
else
  reject the service-request;
end if

In our model, we consider a single queue of buffer size N to hold the user requests if all the VMs are busy. Before deciding whether to accept or reject the request, Request-Controller communicates with the Service manager to ensure that there is no overloading of resources. It also needs the latest information regarding resource availability from Service-manager in order to make resource allocation decisions effectively. Then it assigns requests to VMs.

- Queuing-manager: Queuing-manager is responsible for keeping all the service-requests in the queue using priority scheduling. In case of equal priority first-come first-served (FCFS) discipline is used. Maps should be scanned every turn to modify the priority level of each task. Some restrictive conditions like maximum time a user can wait, should be measured as extra factors. It releases the service-requests for servicing according to the instruction given by Request-controller with Service-manager.

- Storage-manager: Storage-manager stores and retrieves the data required for processing of the jobs.

- Accounting-Module: Accounting-module decides how the completed service-requests should be charged. For instance service-request can be charged based on the total time taken to complete the service and billing rates.

5.3 Model description and its analysis

We assume that in the cloud environment there are c VMs and the buffer size is N. The arriving client requests get together in a single waiting line buffer based on the order of their arrivals, that is, in a FCFS discipline and priority of the client request. Each VM serves only one request at a time. If client requests upon entering into the system finds the VMs are busy waits in the waiting buffer until a VM is available. Optimal management of a finite buffer M/M/R queuing system with two arrival modes has been discussed in [71]. Figure 5.2 shows the state-transition diagram for an M/M/c/N queuing system with two categories of client request modes by considering number of VMs $c = 3$ and buffer size $N = 4$.

The system defined can be studied as a continuous time parameter Markov chain having states $\{(i, j)|\ 0 \leq i + j \leq N\}$, where i is the number of client requests of category 1 and j is the number of client requests of category

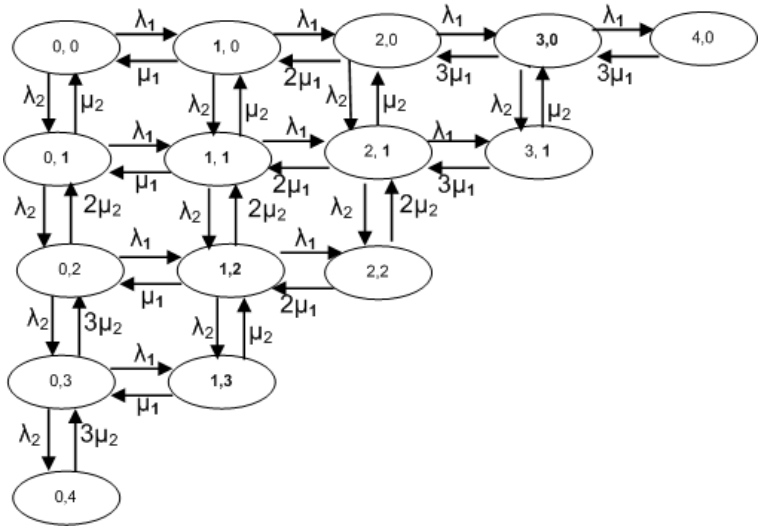

Figure 5.2: State-transition-rate diagram for an M/M/c/N system with two client arrival modes and $c = 3$.

2. Let $P_{i,j}$ be the steady state probability that there are i and j customer requests of categories 1 and 2 in the system, respectively. The mean arrival rate for request categories 1 and 2 are λ_1 and λ_2, respectively. There are c VMs, and each VM has independently and identically distributed exponential service-time distribution with mean $1/\mu_1$ and $1/\mu_2$, respectively, for mode 1 and mode 2. If there are c or more requests in the system, then all c servers must be busy. Since each server processes request with rate μ_k $(k = 1, 2)$, the combined service completion rate for the system is $c\mu_k$. When there are fewer than c requests in the system, $n < c$, only n of the c servers are busy and the combined service completion rate for the system is $n\mu_k$. The steady-state probabilities are given by:

$$(\lambda_1 + \lambda_2)P_{0,0} = \mu_1 P_{1,0} + \mu_2 P_{0,1}, \tag{5.1}$$

$$(\lambda_1 + \lambda_2 + i\mu_1)P_{i,0} = \lambda_1 P_{i-1,0} + \mu_2 P_{i,1} + (i+1)\mu_1 P_{i+1,0}, \quad 1 \le i \le N-1, \tag{5.2}$$

$$c\mu_1 P_{N,0} = \lambda_1 P_{N-1,0}, \tag{5.3}$$

$$
\begin{aligned}
(\lambda_1+\lambda_2+j\mu_2)P_{0,j} &= \lambda_2 P_{0,j-1}+\mu_1 P_{1,j}+(j+1)\mu_2 P_{0,j+1}, \\
&\qquad 1\le j\le N-1, && (5.4)\\
c\mu_2 P_{0,N} &= \lambda_2 P_{0,N-1}, && (5.5)\\
(\lambda_1+\lambda_2+i\mu_1+j\mu_2)P_{i,j} &= \lambda_1 P_{i-1,j}+\lambda_2 P_{i,j-1}+(i+1)\mu_1 P_{i+1,j} \\
&\quad +(j+1)\mu_2 P_{i,j+1},\ 1\le i,j\le N-1, \\
&\quad 2\le i+j\le N-1, && (5.6)\\
(i\mu_1+(N-i)\mu_2)P_{i,N-i} &= \lambda_1 P_{i-1,N-i}+\lambda_2 P_{i,N-i-1}, \\
&\quad 1\le i\le N-1. && (5.7)
\end{aligned}
$$

Solving Equations (5.1)-(5.7) recursively, we get

$$P_{i,j}=\frac{\rho_1^i\rho_2^i}{\prod_{n=1}^{i} min(n,c)\prod_{n=1}^{j} min(n,c)}P_{0,0},\quad 1\le i+j\le N,\quad 0\le i,j\le N.$$

where $\rho_1=\frac{\lambda_1}{\mu_1}$, $\rho_2=\frac{\lambda_2}{\mu_2}$ and

$$\prod_{k=1}^{n} min(k,c) = \begin{cases} n!\ \ if \quad 1\le n\le c, \\ c!c^{n-c}\ \ if \quad c+1\le n\le N. \end{cases} \tag{5.8}$$

Using the normalizing condition $\sum_{i=0}^{N}\sum_{j=0}^{N-i}P_{i,j}=1$, we compute $P_{0,0}$ as

$$P_{0,0}=\left[1+\sum_{j=1}^{N}P_{0,j}+\sum_{i=0}^{N}\sum_{j=0}^{N-i}P_{i,j}\right]^{-1}. \tag{5.9}$$

5.4 System performance measures

Performance measures are the means to examine the efficiency of the queueing model under consideration. As the steady-state probabilities are known, performance measures of the queueing model can be computed. The average number of client requests in the system (L_1) for request category 1, the average number of client requests in the system (L_2) for request category 2, the average number of requests in the system (L_s), the average number of client requests in the waiting buffer (L_q) are given by

$$L_1 = \sum_{i=1}^{N} i\left(\sum_{j=0}^{N-i}P_{i,j}\right),$$

$$
\begin{aligned}
L_2 &= \sum_{j=1}^{N} j \left(\sum_{i=0}^{N-j} P_{i,j} \right), \\
L_s &= \sum_{i+j=0}^{N} (i+j) P_{i,j}, \\
L_q &= \sum_{i=c+1}^{N} (i-c) \sum_{j=0}^{N-c-1} P_{i,j} + \sum_{j=c+1}^{N} (j-c) \sum_{i=0}^{N-c-1} P_{i,j}.
\end{aligned}
$$

The expected number of idle VMs ($E[I]$) and the expected number of busy VMs ($E[B]$) are given by

$$
\begin{aligned}
E[I] &= \sum_{i=0}^{c-1} \sum_{j=0}^{c-i-1} [c-i-j] P_{i,j}, \\
E[B] &= c - E[I].
\end{aligned}
$$

The probability of loss or blocking is $P_{loss}=\sum_{i=0}^{N} P_{i,N-i}$. Using Little's rule, the average waiting time of a request in the system (W_s) is given by

$$W_s = L_s / \lambda^{'},$$

where $\lambda^{'} = \lambda(1 - P_{loss})$ is the effective arrival rate.

5.4.1 Cost analysis

The total expected cost function per unit time for an M/M/c queueing system with finite capacity N and two request categories, in which c and N are two decision variables can be defined as follows:

$$
\begin{aligned}
F(c,N) = C_h L_s + C_1 E[I] + C_2 E[B] + \lambda_1 C_3 + \lambda_2 C_4 + \\
C_5(\lambda_1 + \lambda_2) P_N
\end{aligned} \tag{5.10}
$$

where $P_N = \sum_{i+j=N} P(i,j)$ and
C_h =holding cost per unit time for each client request present in the system.
C_1 =cost per unit time when at least one VM is idle.
C_2=cost incurred per unit time for keeping the VMs busy.
C_3=fixed cost per unit time during the busy period for mode 1 and

C_4 =fixed cost per unit time during the busy period for mode 2.
C_5 =fixed cost for every lost client.

We are interested in determining the optimum number of VMs c, say c^*, and the optimum system capacity N, say N^*, simultaneously, so as to minimize the function F(c, N). It is a difficult task to develop analytic results for the optimum value of c and N, because the expected cost function is highly complex. We note that the derivatives of the operating cost function per unit time are not easily available [72]. We have implemented the numerical searching approach for the cost function using the genetic algorithm. The genetic algorithm is a probabilistic search algorithm that iteratively transforms a set (called a population) of mathematical objects, each with an associated fitness value, into a new population of offspring objects using the natural selection and mutation [69, 73, 74].

Genetic algorithm is an adaptive search algorithm depends on the evolutionary ideas of natural selection and genetics. It represents possible solutions using bit strings of a constant length. By analogy to genetics, the strings can be rendered as chromosomes with soul bits referring the presence (bit = 1) or absence (bit = 0) of a gene. A genetic algorithm permits a population composed of many individuals to germinate under assigned selection rules that minimize the fitness function, that is, the cost function in this chapter.

A population of different possible solutions (chromosomes) is produced and allowed to germinate through a number of generations. Old generations generate new generations in a fashion that mimics genetic change in nature. The solution procedure is as follows:
INPUT: λ_1, λ_2, μ_1, μ_2, C_h, C_1, C_2, C_3,C_4, C_5, c,N and genes, probability of crossover, and probability of mutation.
OUTPUT: approximate solution c^*, N^*, F^*.
Step 1: Population Initialization. An implementation of genetic algorithm initiates with a encoding of each input into a chromosomes. Each gene value either 0 or 1 is randomly generated.
Step 2: Fitness Computation. To determine the optimal expected profit per

unit time for optimal VMs and system capacity, the fitness of chromosomes are computed using the expected cost function F(c,N) in equation (5.10).
Step 3: Selection and Crossover. The selection is a process which mimics the natural survival of the fittest creatures. Each chromosome has a fitness value obtained through the fitness function. The chromosomes which perform better fitness values are given more chances and it discards poor quality genes based on their fitness value. Crossover is done by selecting two parents during reproduction and combining their genes to produce offspring. The parent chromosomes are then mated to generate a new set of offspring chromosomes. This mated procedure is also called crossover.
Step 4: Mutation. Mutation is the random changing of one or more bits in a chromosome. It is useful to create new genes that are not in the initial set of population, or ones that have evolved out of the population, but now would be beneficial.
Step 5: Repeat Step 2 - Step 4 until the stopping criterion is met. We use 50 generations as our stopping criterion.

5.5 Numerical Illustration

This section illustrates the numerical tractability of the optimal threshold policy provided. Numerical results for different system performances are measured in Tables 5.1-5.3. Table 5.1 shows the expected number of client requests L_s in the cloud center for varying number of VMs and buffer size when $(\lambda_1, \mu_1) = (20, 10)$ and $(\lambda_2, \mu_2) = (10, 10)$. It is observed that the number of client requests increases in the system when the buffer size increases but decreases when the number of VMs increases. But one can observe that for a constant N the expected number of client requests decreases as c increases up to a certain value of c, after that the expected number of client requests in the system is static. Tables 5.2 and 5.3 depicts the expected number of client requests in the system for category 1 (L_1) and category 2 (L_2) for (λ_1, μ_1)=(20,10) and (λ_2, μ_2)=(10,10) for varying number of VMs

Table 5.1: Expected no of customer requests in the system L_s.

N	$c=1$	$c=2$	$c=3$	$c=4$	$c=5$	$c=6$
3	1.73	1.34	1.30	1.30	1.30	1.30
4	2.28	1.54	1.44	1.43	1.43	1.43
5	2.81	1.67	1.50	1.48	1.48	1.48
6	3.34	1.75	1.53	1.50	1.50	1.50
7	3.86	1.80	1.54	1.50	1.50	1.50
8	4.38	1.83	1.55	1.51	1.51	1.51
9	4.89	1.85	1.55	1.51	1.51	1.51
10	5.40	1.85	1.55	1.51	1.51	1.51
11	5.91	1.86	1.55	1.51	1.51	1.51
12	6.42	1.86	1.55	1.51	1.51	1.51

and queue length, respectively.

Table 5.4 presents the optimal cost and the expected number of client request in the queue (L_q), the expected number of idle VMs ($E[I]$), the expected number of busy VMs ($E[B]$) and loss probability (P_{loss}) using the genetic algorithm. A numerical illustration is provided by considering the following cost parameters: $C_h = 8, C_1 = 10, C_2 = 30, C_3 = 5, C_4 = 25, C_5 = 10$.We observe that for fixed ρ_2 as ρ_1 increases: (i) The optimum N^* increases. (ii) The optimum cost and the other performance indices increases except $E[I]$. But for fixed ρ_1 as ρ_2 increases: (i) The optimum c^* and N^* increase. (ii) The optimum cost and the other performance indices increases except P_{loss}. This is because the number of the clients in the system increases with the increasing of buffer size N. Thus, L_q, $E[I]$ and $E[B]$ all increase which result in the increasing of the optimal cost.

Figure 5.3 depicts the effect of ρ_1 on the expected number of client requests in the system (L_s) for various number of VMs. The parameters for this graph are taken as $N = 10$ and $\rho_2 = 2.0$. It is seen that as ρ_1 increases L_s increases monotonically. For fixed ρ_1, L_s decreases as the number of VMs increases in the system. Figure 5.4 shows the expected number of client requests of mode 1 (L_1) increases when ρ_1 increases and it decreases when

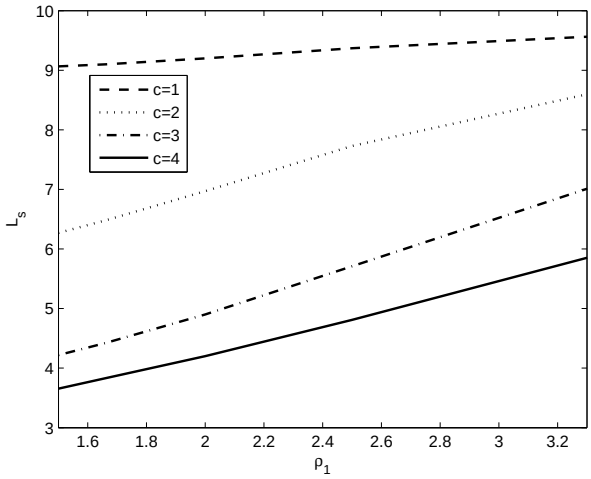

Figure 5.3: Impact of L_s on ρ_1.

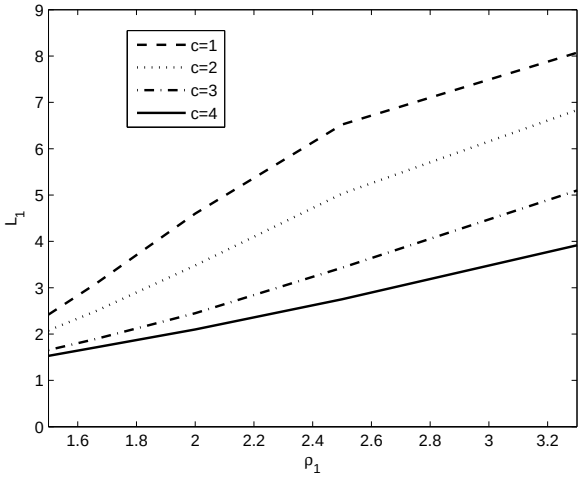

Figure 5.4: Impact of L_1 on ρ_1.

Table 5.2:Expected no of customer requests in the system for request category 1 L_1.

N	$c=1$	$c=2$	$c=3$	$c=4$	$c=5$	$c=6$
3	0.47	0.43	0.43	0.43	0.43	0.43
4	0.56	0.48	0.48	0.48	0.48	0.48
5	0.63	0.51	0.49	0.49	0.49	0.49
6	0.68	0.52	0.50	0.50	0.50	0.50
7	0.72	0.53	0.50	0.50	0.50	0.50
8	0.75	0.53	0.50	0.50	0.50	0.50
9	0.78	0.53	0.50	0.50	0.50	0.50
10	0.80	0.53	0.50	0.50	0.50	0.50
11	0.82	0.53	0.50	0.50	0.50	0.50
12	0.83	0.53	0.50	0.50	0.50	0.50

Table 5.3: Expected no of customer requests in the system for request category 2 L_2.

N	$c=1$	$c=2$	$c=3$	$c=4$	$c=5$	$c=6$
3	1.27	0.91	0.67	0.67	0.67	0.67
4	1.72	1.06	0.81	0.81	0.81	0.81
5	2.19	1.16	0.86	0.86	0.86	0.86
6	2.66	1.23	0.87	0.87	0.87	0.87
7	3.14	0.92	0.88	0.88	0.88	0.88
8	3.62	0.92	0.88	0.88	0.88	0.88
9	4.11	0.92	0.88	0.88	0.88	0.88
10	4.60	0.92	0.88	0.88	0.88	0.88
11	5.09	0.92	0.88	0.88	0.88	0.88
12	5.58	0.92	0.88	0.88	0.88	0.88

Table 5.4:System performance measures under the optimal operating conditions for various values of ρ_1 and ρ_2.

ρ_1	ρ_2	c^*, N^*	$F(c^*, N^*)$	$E[I]$	$E[B]$	L_q	$Ploss$
0.5	2.0	4, 8	501.56286	1.82085	2.17915	0.14414	0.01824
	4.0	7, 11	604.17600	2.96648	4.03352	0.11412	0.01605
	6.0	9, 15	702.48061	2.99269	6.00731	0.27723	0.01130
0.75	2.0	4, 8	509.52326	1.69234	2.30766	0.15083	0.02349
	4.0	7, 12	610.35339	2.87019	4.12981	0.14639	0.01099
	6.0	9, 15	708.16852	2.94535	6.05465	0.28574	0.01284
1.0	2.0	4, 9	517.17149	1.53260	2.46740	0.19701	0.01519
	4.0	7, 12	616.75266	2.78415	4.21585	0.14985	0.01328
	6.0	9, 16	713.91573	2.85838	6.14162	0.32959	0.00964

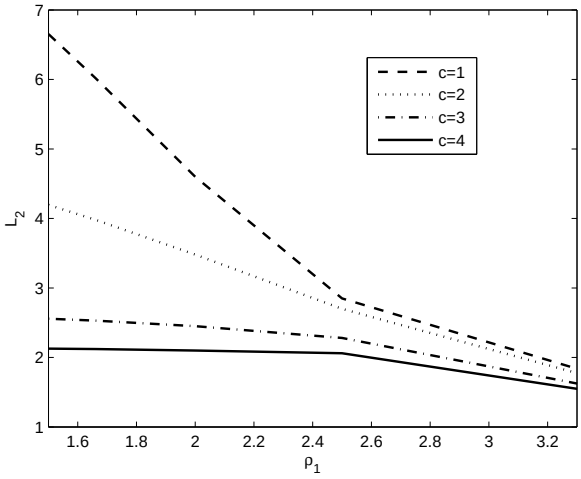

Figure 5.5: Impact of L_2 on ρ_1.

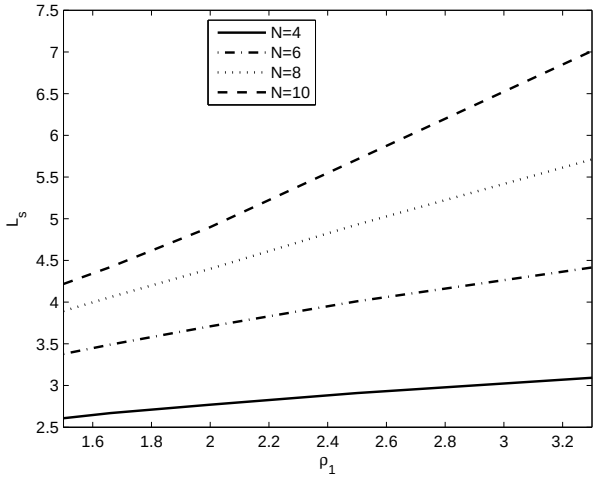

Figure 5.6: Impact of L_s on ρ_1.

number of VMs c in the system increases.

Figure 5.5 shows the expected number of client requests of mode 2 (L_2) decreases when ρ_1 increases and it decreases when number of VMs c in the system increases. Figure 5.6 plots the impact of ρ_1 on expected number of client requests (L_s) in the system for different buffer size N. The parameters are taken as $c = 3$ and $\rho_2 = 2.0$. It can be observed that the expected number of client requests (L_s) in the system increases with the increase in ρ_1 and buffer size N of the cloud system.

The impact of ρ_1 on the expected number of busy VMs ($E[B]$) in the cloud system is shown in Figure 5.7. The parameters are taken as $N = 10$ and $\rho_2 = 2.0$. It is seen that in the cloud system when VM $c = 1$ and $c = 2$, with the increase of ρ_1 the expected number of busy VMs ($E[B]$) is almost static. But when $c > 2$, as ρ_1 increases the expected number of busy VMs increases. Figure 5.8 provide the effect of buffer size (N) on the total cost by varying the number of VMs c. The parameters are taken as $C_h = 8, C_1 = 10, C_2 = 30, C_3 = 5, C_4 = 25, C_5 = 10$, $\lambda_1 = 15$, $\lambda_2 = 20$, $\mu_1 = 20$ and $\mu_2 = 10$. The cost decreases as both N and c increases. To accomplish the minimum average cost, we can carefully setup the buffer size

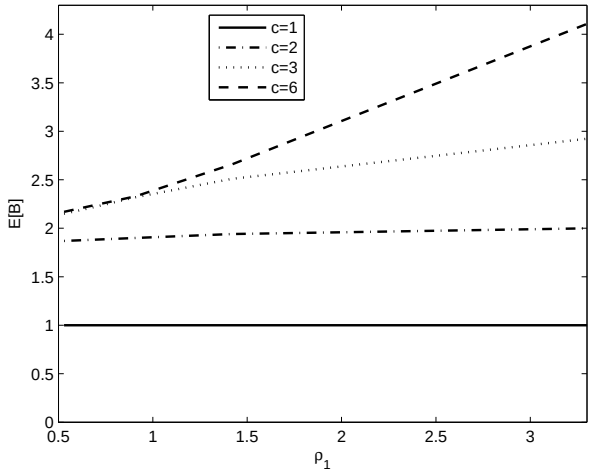

Figure 5.7: Impact of $E[B]$ on ρ_1.

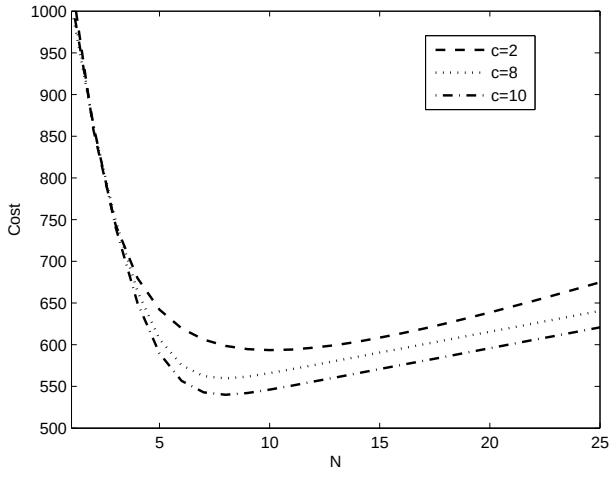

Figure 5.8: Cost Versus buffer size.

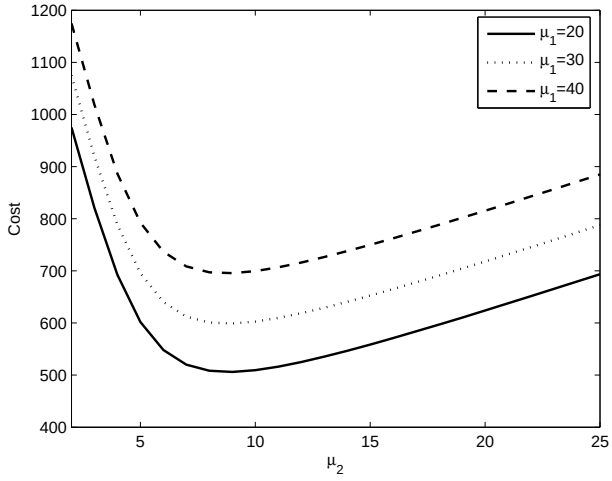

Figure 5.9: Cost Versus μ_2.

N and c in the system.

Figure 5.9 depicts the impact of μ_2 on total expected cost with different service rates μ_1. The parameters are taken as $C_h = 8, C_1 = 10, C_2 = 30, C_3 = 5, C_4 = 25, C_5 = 10$, $C = 4$ and $N = 8$. One may observe that for fixed μ_1, the total expected cost decreases as service rate μ_2 increases and then it becomes almost static. This implies that increasing the service rate μ_2 beyond a certain level cannot further reduce total expected cost. However, the total expected cost reduces considerably when the service rate μ_1 is decreased. It can be seen that the cost is minimized when μ_1=20 and $\mu_2 = 9$ and the minimum value is 506.16. These computed performance measures illustrate the performance effects of various parameters for the system.

5.6 Conclusion

This chapter focused on the finite-buffer two arrival modes multi-server queuing system with client requests in a cloud computing environment. We assumed that the customer requests in any arrival mode is served by one or more VMs, and both the modes gets the service with the equal probability.

We found that the arrival rates of two modes, server sizes, server speeds, buffer size, and the task execution requirement all have significant impact on the average response time of tasks. A variety of numerical results in the form of tables and graphs are discussed to display the effect of the system parameters on the performance measures. Cost analysis has been done to improve the grade of service by selection of appropriate system parameters. By using the analytic properties of the cost function, genetic algorithm is applied to search the optimal values of c and N. It is a trade-off that potential applications need to consider in deciding the performance evaluation of server farms as an significant view of cloud computing which is of essential concern for both providers and customers of cloud system. We can evaluate the impact of operation performance on user-perceived service performance using this model. We plan to develop the analytical performance model for priority queuing systems with impatient customers, where requests are dropped from the queue once the service time is greater than the delay requirements. As future work, research will be carried out on useful algorithms for measuring deployment costs of virtual resources in multi-cloud environments.