*Lizhe Wang, Rajiv Ranjan, Jinjun Chen, Boualem Benatallah CRC, Taylor & Francis group*

# *Cloud Computing: methodology, system, and applications*

# *Contributors*

**Hamzeh Khazaei**
University of Manitoba
Winnipeg, Manitoba

**Jelena Mišić**
Ryerson University
Toronto, Ontario

**Vojislav B. Mišić**
Ryerson University
Toronto, Ontario

**Michael Aftosmis**
NASA Ames Research Center
Moffett Field, California

**Pratul K. Agarwal**
Oak Ridge National Laboratory
Oak Ridge, Tennessee

**Sadaf R. Alam**
Oak Ridge National Laboratory
Oak Ridge, Tennessee

**Gabrielle Allen**
Louisiana State University
Baton Rouge, Louisiana

**Martin Sandve Alnæs**
Simula Research Laboratory and
University of Oslo, Norway
Norway

**Steven F. Ashby**
Lawrence Livermore National
Laboratory
Livermore, California

**David A. Bader**
Georgia Institute of Technology
Atlanta, Georgia

**Benjamin Bergen**
Los Alamos National Laboratory
Los Alamos, New Mexico

**Jonathan W. Berry**
Sandia National Laboratories
Albuquerque, New Mexico

**Martin Berzins**
University of Utah
Salt Lake City, Utah

**Abhinav Bhatele**
University of Illinois
Urbana-Champaign, Illinois

**Christian Bischof**
RWTH Aachen University
Germany

**Rupak Biswas**
NASA Ames Research Center
Moffett Field, California

**Eric Bohm**
University of Illinois
Urbana-Champaign, Illinois

# 1

# *Cloud Computing: Performance Analysis*

**Hamzeh Khazaei**

*University of Manitoba, Winnipeg, Manitoba*

**Jelena Mišić**

*Ryerson University, Toronto, Ontario*

**Vojislav B. Mišić**

*Ryerson University, Toronto, Ontario*

## CONTENTS

Cloud computing is a computing paradigm in which different computing resources, including infrastructure, hardware platforms, and software applications, are made accessible to remote users as services. Successful provision of infrastructure-as-a-service (IaaS) and, consequently, widespread adoption of cloud computing necessitates accurate performance evaluation that allows service providers to dimension their resources in order to fulfil the service level agreements with their customers. In this chapter, we describe an analytical model for performance evaluation of cloud server farms, and demonstrate the manner in which important performance indicators such as request waiting time and server utilization may be assessed with sufficient accuracy.

## 1.1 Introduction

Significant innovations in virtualization and distributed computing, as well as improved access to high-speed Internet, have accelerated interest in cloud computing [19]. Cloud computing is a general term for system architectures that involves delivering hosted services over the Internet. These services are broadly divided into three categories: Infrastructure-as-a-Service (IaaS), which includes equipment such as hardware, storage, servers, and networking components are made accessible over the Internet); Platform-as-a-Service (PaaS), which includes computing platforms—hardware with operating systems, virtualized servers, and the like; and Software-as-a-Service (SaaS), which includes sofware applications and other hosted services [2]. A cloud service differs from traditional hosting in three principal aspects. First, it is provided on demand, typically by the minute or the hour; second, it is elastic since the user can have as much or as little of a service as they want at any given time; and third, the service is fully managed by the provider – user needs little more than computer and Internet access. Cloud customers pay only for the services they use by means of a customized service level agreement (SLA), which is a contract negotiated and agreed between a customer and a service provider: the service provider is required to execute service requests from a customer within negotiated quality of service(QoS) requirements for a given price.

Due to dynamic nature of cloud environments, diversity of user's requests and time dependency of load, providing expected quality of service while avoiding over-provisioning is not a simple task [21]. To ensure that the QoS perceived by end clients is acceptable, the providers must exploit techniques and mechanisms that guarantee a minimum level of QoS. Although QoS has multiple aspects such as response time, throughput, availability, reliability, and security, the primary aspect of QoS considered in this work is related to response time [20].

Cloud computing has been the focus of much research in both academia and industry, however, implementation-related issues have received much more attention than performance-related ones; here we describe an analytical model for evaluating the performance of cloud server farms and verify its accuracy with numerical calculations and simulations. we assume that any request goes through a *facility node* and then leaves the center. A facility node may contain different computing resources such as web servers, database servers, and others, as shown in Fig. 1.1. We consider the time a request spends in one of those facility node as the response time; response time does not follow any specific distribution. Our model is flexible in terms of cloud center size and service time of customer requests; We model the cloud environment as an $M/G/m$ queuing system which indicates that inter-arrival time of requests is exponentially distributed, the service time is generally distributed and the number of facility nodes is $m$. Also, due to the the nature of cloud environment (i.e., it is

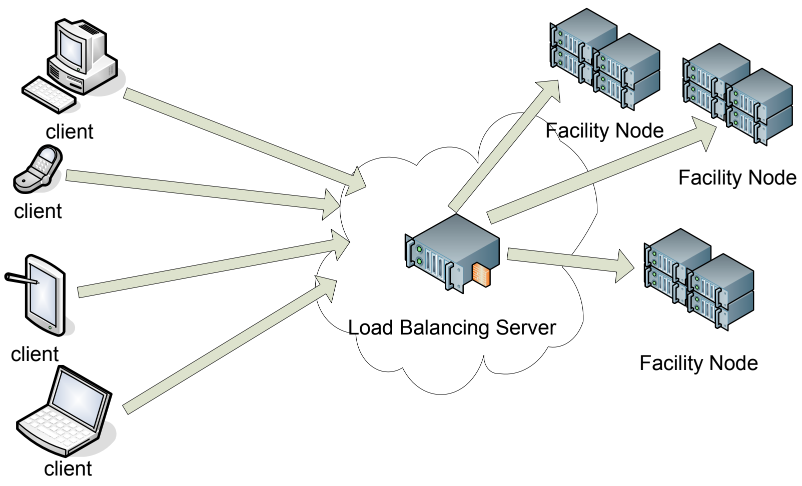


**FIGURE 1.1**
Cloud clients and service provider.

a service provider with potentially many customers), we pose no restrictions on the number of facility nodes. These two characteristics, general service time and large number of nodes, have not been adequately addressed in previous research.

## 1.2 Related Work

Cloud computing has attracted considerable research attention, but only a small portion of the work done so far has addressed performance issues, and the rigorous analytical approach has been adopted by only a handful among these. In [24], the authors studied the response time in terms of various metrics, such as the overhead of acquiring and realizing the virtual computing resources, and other virtualization and network communication overhead. To address these issues, they have designed and implemented C-Meter, a portable, extensible, and easy-to-use framework for generating and submitting test workloads to computing clouds. Most of the research related to cloud computing has dealt with implementation issues, while performance-related issues have received much less attention.

In [21], the authors consider a cloud center which is modelled as the classic open network; they obtained the distribution of response time based on assumption that inter-arrival time and service time are both exponential. Using the distribution of response time, they found the relationship among the maximal number of tasks, the minimal service resources and the highest level of services.

Theoretical analyses have mostly relied on extensive research in performance evaluation of $M/G/m$ queuing systems [1, 3, 4, 8, 10, 11, 12, 16, 18, 23]. As solutions for distribution of response time and queue length in $M/G/m$ systems and $M/G/m/m+r$ can't be obtained in closed form, suitable approximations were sought. However, most of these provide reasonably accurate estimates of mean response time only when number of servers is comparatively small as well as small *coefficient of variation* of service time, $CV$, (less than unity), but fail for large number of servers and higher $CV$. Approximation errors are particularly pronounced when the offered load $\rho$ is small, and/or when both the number of servers $m$ and the $CV$ of the service time, are large [1, 4, 18].

A closed form expression for the blocking probability in $M/G/m/m + r$ based on the exact solution for finite capacity exponential queues was proposed in [14]. There are essentially two problems of interest in this paper; the first is how to estimate the blocking probability and the second problem concerns the allocation of buffers so that the loss/delay blocking probability will be below a specific threshold. The building block of this approach is the exact solution of $M/M/m/m + r$ queuing system so this approach is more likely to be suitable for service time distributions for which the $CV$ does not exceed one.

An approximations for the mean queue length in the $M/G/m/m+r$ queue without deriving the distribution of number of tasks in system is proposed in [11]. Moreover their methods were given in transform and required lots of computation in order to be evaluated.

In [22], the cloud center was modelled as an $M/M/m/m + r$ queuing system, which has been used to compute the distribution of response time. Interarrival and service times were both assumed to be exponentially distributed, and the system had a finite buffer of size $m + r$. The response time was broken down into waiting, service, and execution periods, assuming that all three periods are independent which is unrealistic, based on their own argument.

For an $M/G/m/m + r$ queues there is no explicit formula for probability distribution of number of tasks in system except in a few special cases: if $G = M$, $r = 0$ or $m = 1$; then the exact and close form of distribution of tasks in system are attainable; M denotes the exponential cumulative distribution function. However it is quite difficult to obtain explicit formula for probability distribution of the number of tasks in the system in general case.

The author in [6] proposed a transform free approach for steady-state queue length distribution in an $M/G/m$ system with finite waiting space; his approach was given in an explicit form and hence its numerical computation is easier than that for previous approximations [3, 17]. Although the approach is exact for $M/M/m/m + r$, $r = 0$ and reasonably accurate for general case, apparently the method is suitable for small number of servers only.

In [5], the author considered the problem of optimal buffer designing for $M/G/m$ in order to determine the smallest buffer capacity such that the rate of lost tasks remains under predefined level. Here, Kimura used the same approach in [6] in order to approximate the blocking probability and then applied the approximate formula to the buffer design problem. Based on convex order value, Kimura concluded that the higher the order of convexity for service time, leads to the bigger the optimal buffer size.

As a result, the former approaches are not directly applicable to performance analysis of cloud computing server farms where the number of servers is huge and service request arrival distribution is not generally known.

## 1.3 The Analytical Model

We model a cloud server farm as a $M/G/m$ queuing system which indicates that the inter-arrival time of requests is exponentially distributed, the service times of customers' requests are independent and identically distributed random variables with a general distribution whose service rate is $\mu$; both $\mu$ and $CV$, the coefficient of variation defined as standard deviation divided by the mean, are finite.

A $M/G/m$ queuing system may be considered as a Markov process which can be analysed by applying the embedded Markov chain technique. Embedded Markov Chain techique requires selection of Markov points in which the state of the system is observed. Therefore we monitor the number of the tasks in the system (both in service and queued) at the moments immediately before the task request arrival. If we consider the system at Markov points and number these instances 0, 1, 2, . . . , then we get a Markov chain [7]. Here, the system under consideration contains $m$ servers, which render service in order of task request arrivals.

Task requests arrival process is Poisson. Task request interarrival time $A$ is exponentially distributed with rate to $\frac{1}{\lambda}$. We will denote its Cumulative Distribution Function (CDF) as $A(x) = Prob[A < x]$ and its probability density function (pdf) as $a(x) = \lambda e^{-\lambda x}$. Laplace Stieltjes Transform (LST) of interarrival time is $A^*(s) = \int_0^\infty e^{-sx}a(x)dx = \frac{\lambda}{\lambda+s}$.

Task service times are identically and independently distributed according to a general distribution $B$, with a mean service time equal to $\overline{b} = \frac{1}{\mu}$. The CDF of the service time is $B(x) = Prob\,[B < x]$, and its pdf is $b(x)$. The LST of service time is $B^*(s) = \int_0^\infty e^{-sx}b(x)dx$.

Residual task service time is time from the random point in task execution till the task completion. We will denote it as $B_+$. This time is necessary for our model since it represents time distribtion between task arrival $z$ and departure of the task which was in service when task arrival $z$ occured. It can be shown as well that probability distrubtion of elapsed service time (between start of the task execution and next arrival of task request $B_-$ has the same probability distribtion [15].

The LST of residual and elapsed task service times can be calculated in [15] as

$$B_+^*(s) = B_-^*(s) = \frac{1 - B^*(s)}{s\bar{b}} \tag{1.1}$$

The offered load may be defined as

$$\rho \triangleq \frac{\lambda}{m\mu} \tag{1.2}$$

For practical reasons, we assume that the system never enters saturation, which means that any request submitted to the center will get access to the required facility node after a finite queuing time. Furthermore, we also assume each task is serviced by a single server (i.e., there are no batch arrivals), and we do not distinguish between installation (setup), actual task execution, and finalization components of the service time; these assumptions will be relaxed in our future work.

### 1.3.1 The Markov chain

We are looking at the system at the moments of task request arrivals – these points are selected as Markov points. A given Markov chain has a steady-state solution if it is ergodic. Based on conditions for ergodicity [7] and the above-mentioned assumptions, it is easy to prove that our Markov Chain is ergodic. Then, using the steady-state solution, we can extract the distribution of number of tasks in the system as well as the response time.

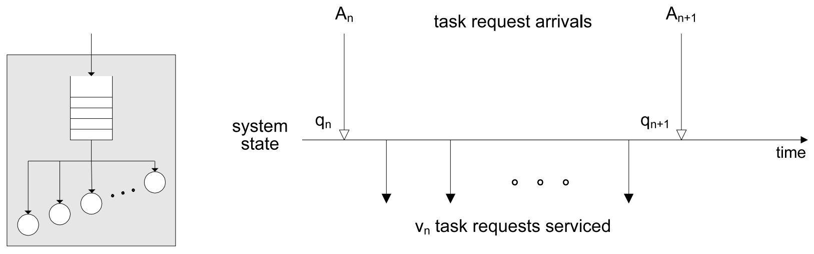


**FIGURE 1.2**
Embedded Markov points.

Let $A_n$ and $A_{n+1}$ indicate the moment of $n^{th}$ and $(n+1)^{th}$ arrivals to the system, respectively, while $q_n$ and $q_{n+1}$ indicate the number of tasks found in the system immediately before these arrivals; this is schematically shown in Fig. 1.2. If $v_{n+1}$ indicates the number of tasks which are serviced and depart from the system between $A_n$ and $A_{n+1}$, the following holds:

$$q_{n+1} = q_n - v_{n+1} + 1 \tag{1.3}$$

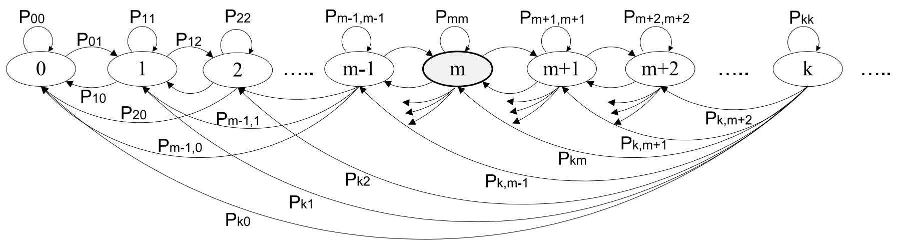


**FIGURE 1.3**
State-transition-probability diagram for the $M/G/m$ embedded Markov chain.

We need to calculate the transition probabilities associated with this Markov chain, defined as

$$p_{ij} \triangleq Prob\left[q_{n+1} = j | q_n = i\right] \tag{1.4}$$

i.e., the probability that $i + 1 - j$ customers are served during the interval between two successive task request arrivals. Obviously for $j > i + 1$

$$p_{ij} = 0 \tag{1.5}$$

since there are at most $i+1$ tasks present between the arrival of $A_n$ and $A_{n+1}$. The Markov state-transition-probability diagram as in Fig. 1.3, where states are numbered according to the number of tasks currently in the system (i.e those in service and those awaiting service). For clarity, some transitions are not fully drown, esp. those originating from states above $m$. We have also highlighted the state $m$ because the transition probabilities are different for states on the left and right hand side of this state (i.e., below and above $m$).

## 1.3.2 Departure Probabilities

Due to ergodicity of the Markov chain, an equilibrium probability distribution will exist for the number of tasks present at the arrival instants; so we define

$$\pi_k = \lim_{n \to +\infty} Prob\left[q_n = k\right] \tag{1.6}$$

From [15], the direct method of solution for this equilibrium distribution requires that we solve the following system of linear equations:

$$\pi = \pi \mathbf{P} \tag{1.7}$$

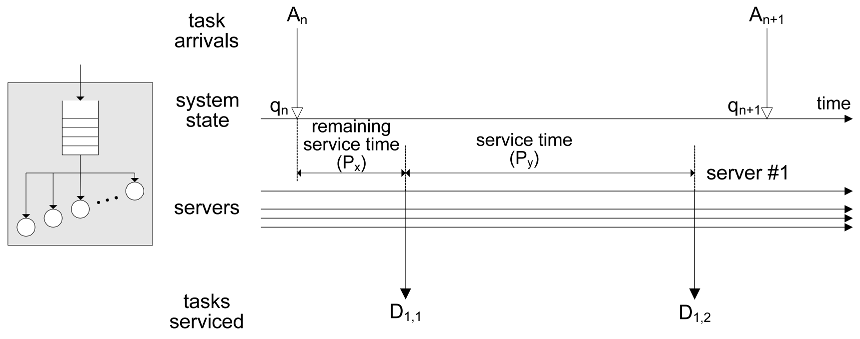


**FIGURE 1.4**
System behaviour in between two arrivals.

where $\pi = [\pi_0, \pi_1, \pi_2, \ldots]$, and $\mathbf{P}$ is the matrix whose elements are one-step transition probabilities $p_{ij}$.

To find the elements of the transition probability matrix, we need to count the number of tasks departing from the system in between two successive arrivals. Consider the behaviour of the system, as shown in Fig. 1.4. Each server has zero or more departures during the time between two successive task request arrivals (the inter-arrival time). Let us focus on an arbitrary server, which (without loss of generality) could be the server number 1. For a task to finish and depart from the system during the inter-arrival time, its remaining duration (residual service time defined in (1.1)) must be shorter than the task inter-arrival time. This probability will be denoted as $P_x$, and it can be calculated as

$$\begin{aligned} P_x = Prob\,[A > B_+] = \int_{x=0}^{\infty} P\{A > B_+ | B_+ = x\} P\{B_+ = x\} \\ = \int_0^{\infty} e^{-\lambda x} dB_+(x) = B_+^*(\lambda) \end{aligned} \tag{1.8}$$

Physically this result presents probability of no task arrivals during residual task service time.

In the case when arriving task can be accommodated immediately by an idle server ( and therefore queue length is zero) we have to evaluate the probability that such task will depart before next task arrival. We will denote this probability as $P_y$ and calculate it as:

$$\begin{aligned} P_y = Prob\,[A > B] = \int_{x=0}^{\infty} P\{A > B | B = x\} P\{B_+ = x\} \\ = \int_0^{\infty} e^{-\lambda x} dB(x) = B^*(\lambda) \end{aligned} \tag{1.9}$$

However, if queue is non-empty upon task arrival following situation may happen. If between two successive new task arrivals a completed task departs from a server, that server will take a new task from the non-empty queue. That task may be completed as well before the next task arrival and if the queue is still non-empty new task may be executed, and so on until either queue gets empty or new task arrives. Therefore probability of $k > 0$ job departures from a single server, given that there are enough jobs in the queue can be derived from expressions (1.8) and (1.9) as:

$$P_{z,k} = B_{+}^{*}(\lambda)(B^{*}(\lambda))^{k-1} \tag{1.10}$$

note that $P_{z,1} = P_x$.

Using these values we are able to compute the transition probabilities matrix.

### 1.3.3 Transition Matrix

Based on our Markov chain, we may identify four different regions of operation for which different conditions hold; these regions are schematically shown in Fig. 1.5, where the numbers on horizontal and vertical axes correspond to the number of tasks in the system immediately before a task request arrival ($i$) and immediately upon the next task request arrival ($j$), respectively.

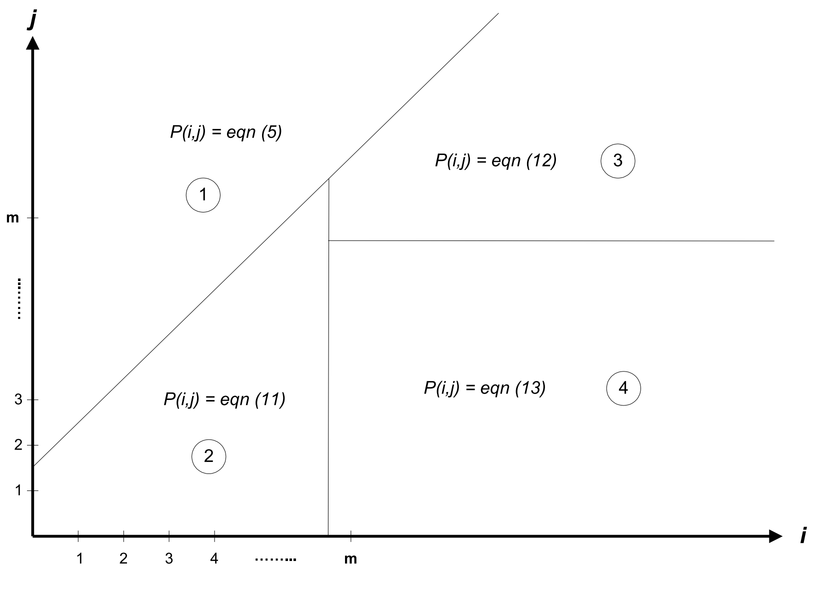


**FIGURE 1.5**
Range of validity for $p_{ij}$ equations.

Regarding the region labelled 1, we already know from Eq. 1.5 that $p_{ij} = 0$ for $i + 1 < j$.

In region 2, no tasks are waiting in the queue, hence $i < m$ and $j \leq m$. In between the two successive request arrivals, $i+1-j$ tasks will complete their service. For all transitions located on the left side of state $m$ in Fig. 1.3, the probability of having $i+1-j$ departures is

$$p_{ij} = \binom{i}{i-j} P_x^{i-j}(1-P_x)^j P_y + \binom{i}{i+1-j} P_x^{i+1-j}(1-P_x)^{j-1}(1-P_y) \qquad \text{for } i < m, j \leq m \tag{1.11}$$

Region 3 corresponds to the case where all servers are busy throughout the inter-arrival time, i.e., $i, j \geq m$. In this case all transitions remain to the right of state $m$ in Fig. 1.3, and state transition probabilities can be calculated as

$$p_{ij} = \sum_{s=\phi}^{\sigma} \binom{m}{s} P_x^s (1-P_x)^{m-s} P_{z,2}^{i+1-j-s}(1-P_{z,2})^s \qquad \text{for } i, j \geq m \tag{1.12}$$

In the last expression, the summation bounds are $\sigma = min\,[i+1-j, m]$ and $\phi = min\,[i+1-j, 1]$.

Finally, region 4, in which $i \geq m$ and $j \leq m$, describes the situation where the first arrival ($A_n$) finds all servers busy and a total of $i-m$ tasks waiting in the queue, which it joins; while at the time of the next arrival ($A_{n+1}$) there are exactly $j$ tasks in the system, all of which are in service. The transition probabilities for this region are

$$p_{ij} = \sum_{s=1}^{\sigma} \binom{m}{s} P_x^s (1-P_x)^{m-s} \binom{\eta}{\alpha} P_{z,2}^{\psi}(1-P_{z,2})^{\zeta}\beta \qquad \text{for } i \geq m, j < m \tag{1.13}$$

where we used the following notation:

$$\begin{array}{rl} \sigma = & min\,[m, i+1-j] \\ \eta = & min\,[s, i+1-m] \\ \alpha = & min\,[s, i+1-j-s] \\ \psi = & max\,[0, i+1-j-s] \\ \zeta = & max\,[0, j-m+s] \\ \beta = & \begin{cases} 1 & \text{if } \psi \leq i+1-m \\ 0 & \text{otherwise} \end{cases} \end{array} \tag{1.14}$$

## 1.4 Numerical Validation

The steady-state balance equations outlined above can't be solved in closed form, hence we must resort to a numerical solution. To obtain the steady-state probabilities $\pi = [\pi_0, \pi_1, \pi_2, ...]$, as well as the mean number of tasks in the system (in service and in the queue) and the mean response time, we have used the probability generating functions (PGFs) for the number of tasks in the system:

$$P(z) = \sum_{k=0}^{\infty} \pi_z z^k \tag{1.15}$$

and solved the resulting system of equations using Maple 13 from Maplesoft, Inc. [9]. Since the PGF is an infinite series, it must be truncated for numerical solution; we have set the number of equations to twice the number of servers, which allows us to achieve satisfactory accuracy (as will be explained below), plus the necessary balance equation

$$\sum_{i=0}^{2m} \pi_i = 1. \tag{1.16}$$

the mean number of tasks in the system is, then, obtained as

$$E[QS] = P^{'}(1) \tag{1.17}$$

while the mean response time is obtained using Little's law as

$$E[RT] = E[QS]/\lambda \tag{1.18}$$

We have assumed that the task request arrivals follow the gamma distribution with different values for shape and scale parameters; however, our model may accommodate other distributions without any changes. Then, we have performed two experiments with variable task request arrival rate and coefficient of variation $CV$ (which can be adjusted in the gamma distribution independently of the arrival rate).

To validate the analytical solutions we have also built a discrete even simulator of the cloud server farm using object-oriented Petri net-based simulation engine Artifex by RSoftDesign, Inc. [13].

The diagrams in Fig. 1.6 show analytical and simulation results (shown as lines and symbols, respectively) for mean number of tasks in the system as functions of the offered load $\rho$, under different number of servers. Two different values of the coefficient of variation, $CV = 0.7$ and 0.9, were used; the corresponding results are shown in Figs. 1.6(a) and 1.6(b). As can be seen, the results obtained by solving the analytical model agree very well with those obtained by simulation.

The diagrams in Fig. 1.8 show the mean response time, again for the same range of input variables and for the same values of the coefficient of variation. As above, solid lines correspond to analytical solutions, while different symbols correspond to different number of servers. As could be expected, the response time is fairly steady up to the offered load of around $\rho = 0.8$, when it begins to increase rapidly. However, the agreement between the analytical solutions and simulation results is still very good, which confirms the validity of our modelling approach.

## 1.5 Conclusions

Performance evaluation of server farms is an important aspect of cloud computing which is of crucial interest for both cloud providers and cloud customers. In this chapter we have proposed an analytical model for performance evaluation of a cloud computing center. Due to the nature of the cloud environment, we assumed general service time for requests as well as large number of servers; in the other words, our model is flexible in terms of scalability and diversity of service time. We have further conducted numerical experiments and simulation to validate our model. Numerical and simulation results showed that the proposed method provided a quite accurate computation of the mean number of tasks in the system and mean response time.

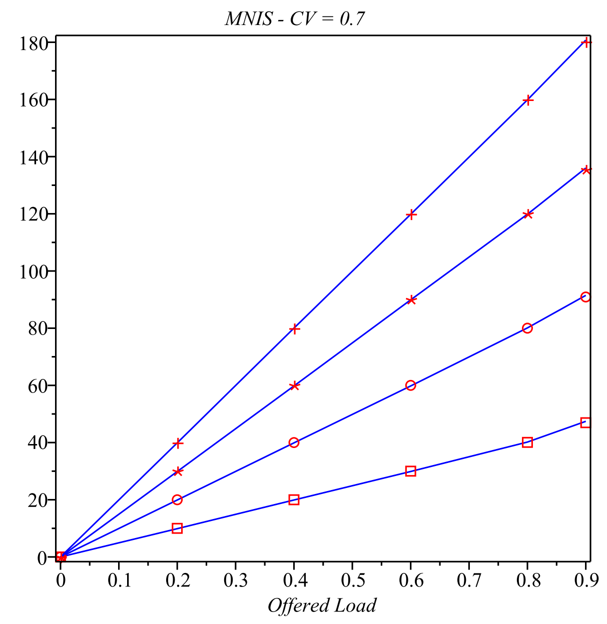
The model could be extend for burst arrivals of requests or a kind of task including several subtasks; examining other types of distributions as service time which are more realistic in cloud computing area, e.g. Log-Normal distribution and looking in to the facility node and breaking down the response time into several components such as setup, execution, return and clean up time could be another dimension of extension. The authors will address all these issues in future work.

## 1.6 Glossary

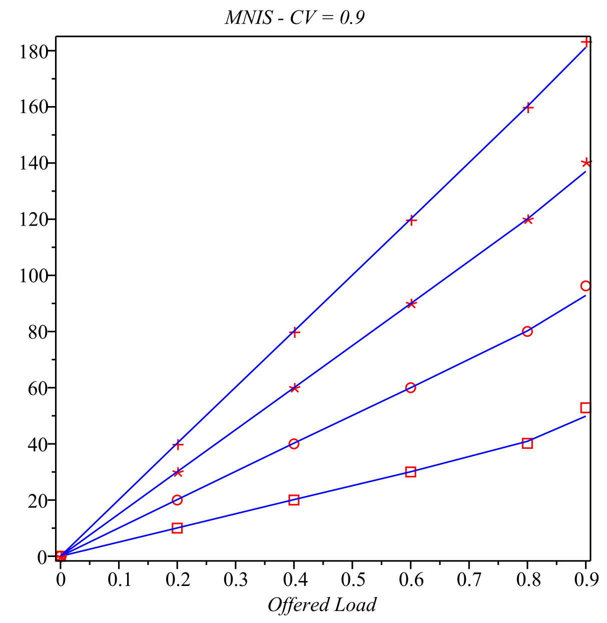
**Markov Process:** In probability theory and statistics, a Markov process, named after the Russian mathematician Andrey Markov, is a time-varying random phenomenon for which a specific property (the Markov property) holds. In a common description, a stochastic process with the Markov property, or memorylessness, is one for which conditional on the present state of the system, its future and past are independent.

**Markov Chain:** A Markov chain is a random process with the Markov property, i.e. the property, simply said, that the next state depends only on the current state and not on the past.

**Embedded Markov Chain Technique:** One method of finding the stationary probability distribution, $\pi$, of an ergodic continuous-time Markov process, $Q$, is by first finding its embedded Markov chain (EMC). Strictly speaking, the EMC is a regular discrete-time Markov chain, sometimes referred to as a jump process. Each element of the one-step transition probability matrix of the EMC, $S$, is denoted by $p_{ij}$, and represents the conditional probability of transitioning from state i into state j.

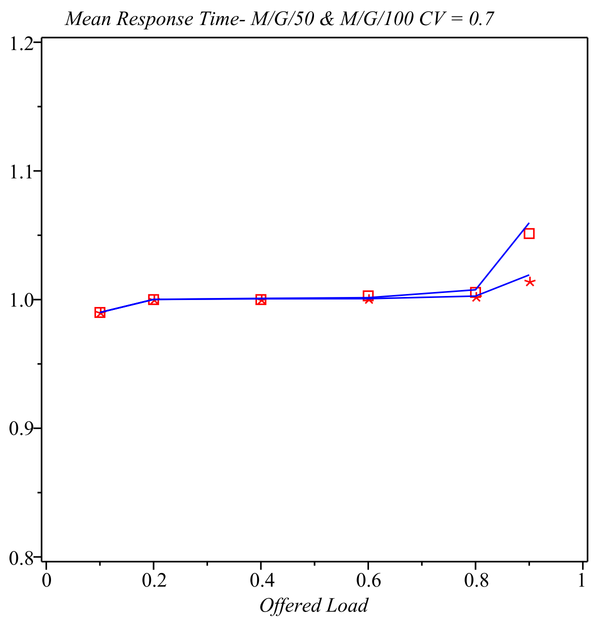


(a) $CV = 0.7$.

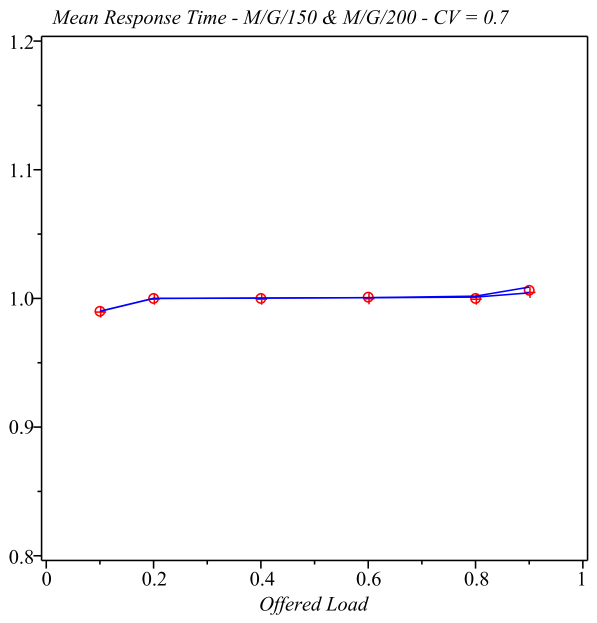


(b) $V = 0.9$.

**FIGURE 1.6**
Mean number of tasks in the system: $m = 50$ (denoted with squares), 100 (circles), 150 (asterisks), and 200 (crosses).

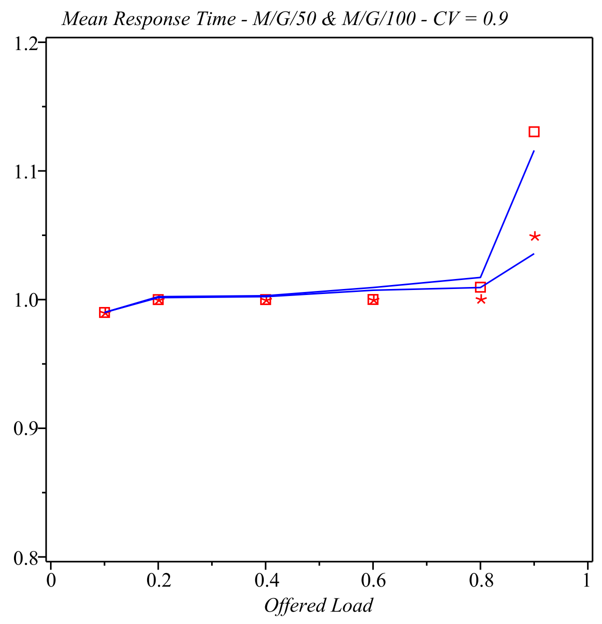


(a) Results for $CV = 0.7$, $m = 50$ and 100 servers.

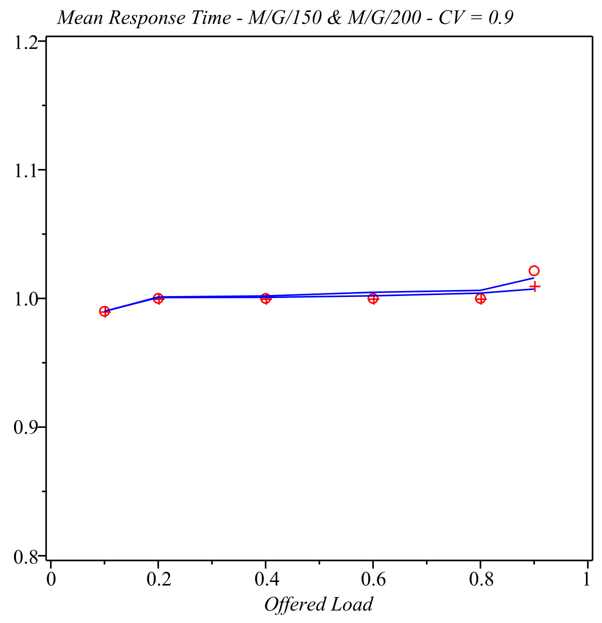


(b) Results for $CV = 0.7$, $m = 150$ and 200 servers.

**FIGURE 1.7**
Mean response time $CV = 0.7$, $m = 50$ (denoted with squares), 100 (asterisks), 150 (circles), and 200 (crosses).



(a) Results for $CV = 0.9$, $m = 50$ and 100 servers.



(b) Results for $CV = 0.9$, $m = 150$ and 200 servers.

**FIGURE 1.8**
Mean response time for $CV = 0.9$, $m = 50$ (denoted with squares), 100 (asterisks), 150 (circles), and 200 (crosses).

# Bibliography

[1] O. J. Boxma, J. W. Cohen, and N. Huffel. Approximations of the mean waiting time in an $M/G/s$ queueing system. *Operations Research*, 27:1115–1127, 1979.

[2] B. Furht. Cloud computing fundamentals. In *Handbook of Cloud Computing*, pages 3–19. Springer US, 2010.

[3] P. Hokstad. Approximations for the $M/G/m$ queues. *Operations Research*, 26:510–523, 1978.

[4] T. Kimura. Diffusion approximation for an $M/G/m$ queue. *Operations Research*, 31:304–321, 1983.

[5] T. Kimura. Optimal buffer design of an $M/G/s$ queue with finite capacity. *Communications in Statistics Stochastic Models*, 12(6):165–180, 1996.

[6] T. Kimura. A transform-free approximation for the finite capacity $M/G/s$ queue. *Operations Research*, 44(6):984–988, 1996.

[7] L. Kleinrock. *Queueing Systems, volume 1, Theory.* Wiley-Interscience, 1975.

[8] B. N. W. Ma and J. W. Mark. Approximation of the mean queue length of an $M/G/c$ queueing system. *Operations Research*, 43:158–165, 1998.

[9] Maplesoft, Inc. *Maple 13.* Waterloo, ON, Canada, 2009.

[10] M. Miyazawa. Approximation of the queue-length distribution of an $M/GI/s$ queue by the basic equations. *Journal of Applied Probability*, 23:443–458, 1986.

[11] S. A. Nozaki and S. M. Ross. Approximations in finite-capacity multi-server queues with poisson arrivals. *Journal of Applied Probability*, 15:826–834, 1978.

[12] E. Page. Tables of waiting times for $M/M/n$, $M/D/n$ and $D/M/n$ and their use to give approximate waiting times in more general queues. *J. Operational Research Society*, 33:453–473, 1982.

[13] RSoft Design. *Artifex v.4.4.2.* RSoft Design Group, Inc., San Jose, CA, 2003.

[14] J. M. Smith. $M/G/c/K$ blocking probability models and system performance. *Perform. Eval.*, 52:237–267, May 2003.

[15] H. Takagi. *Queueing Analysis*, volume 1: Vacation and Priority Systems. North-Holland, Amsterdam, The Netherlands, 1991.

[16] Y. Takahashi. An approximation formula for the mean waiting time of an $M/G/c$ queue. *J. Operational Research Society*, 20:150–163, 1977.

[17] H. C. Tijms. Heuristics for finite-buffer queues. *Probability in the Engineering and Informational Sciences*, 6:277–285, 1992.

[18] H. C. Tijms, M. H. V. Hoorn, and A. Federgru. Approximations for the steady-state probabilities in the $M/G/c$ queue. *Advances in Applied Probability*, 13:186–206, 1981.

[19] L. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1), 2009.

[20] L. Wang, G. V. Laszewski, A. Younge, X. He, M. Kunze, J. Tao, and C. Fu. Cloud computing: a perspective study. *New Generation Computing*, 28:137–146, 2010.

[21] K. Xiong and H. Perros. Service performance and analysis in cloud computing. In *Proceedings of the 2009 Congress on Services - I*, pages 693–700, Los Alamitos, CA, USA, 2009.

[22] B. Yang, F. Tan, Y. Dai, and S. Guo. Performance evaluation of cloud service considering fault recovery. In *Cloud Computing*, volume 5931 of *Lecture Notes in Computer Science*, pages 571–576. Springer Berlin Heidelberg, 2009.

[23] D. D. Yao. Refining the diffusion approximation for the $M/G/m$ queue. *Operations Research*, 33:1266–1277, 1985.

[24] N. Yigitbasi, A. Iosup, D. Epema, and S. Ostermann. C-meter: A framework for performance analysis of computing clouds. In *CCGRID '09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 472–477, Washington, DC, USA, 2009.