J Supercomput (2017) 73:2130–2156
DOI 10.1007/s11227-016-1906-5

Performability analysis of cloud computing centers with large numbers of servers

Enver Ever[1]

Published online: 24 October 2016

Abstract The ability to deliver acceptable levels of quality of service is crucial for cloud systems, and this requires performance as well as availability analysis. Existing modeling attempts mainly focus on pure performance analysis; however, the software and hardware components of cloud infrastructures may have limited reliability. In this study, analytical models are presented for performability evaluation of cloud centers. A novel approximate solution approach is introduced which allows consideration of large numbers of servers. The challenges for analytical modeling of cloud systems mentioned in the literature are considered. The analytical models and solutions, therefore, are capable of considering large numbers of facility nodes typically up to orders of hundreds or thousands, and able to incorporate various traffic loads while evaluating quality of service for cloud centers together with server availabilities. The results obtained from the analytical models are presented comparatively with the results obtained from discrete event simulations for validation.

Keywords Cloud computing · Fault-tolerant systems · Performability · Queuing theory

1 Introduction

Cloud computing arises as a new paradigm that aims to provide computing environments which are flexible, reliable, and with good quality of service (QoS). Its ability of transforming the way information is going to be processed, exchanged, stored,

✉ Enver Ever
eever@metu.edu.tr

[1] Computer Engineering, Middle East Technical University, Northern Cyprus Campus, Mersin 10, Güzelyurt, Turkey

and at the same time the highly available and inexpensive computing power it offers, makes the cloud interesting for researchers, engineers, service providers, developers, and users. In recent studies, cloud computing is envisioned as a promising technology which can change the way computing and resources will be accessed in the near future [3,13].

Cloud computing is considered as a new service model, where a service level agreement specifies the service usage and obligations both for clients and service providers. The services provided can be summarized in three main categories as: Platform as a service (PaaS), where computing platforms such as virtualized servers and operating systems are provided and the clients can deploy onto the cloud various client created and/or managed applications which are usually established using various programming languages, libraries, services and some tools; Software as a service (SaaS), where providers' applications are used by the clients and are accessible from various devices and interfaces; and Infrastructure as a service (IaaS), where provision processing, storage, networks, and other fundamental computing resources are provided and the clients are able to deploy and run operating systems as well as applications [3,10].

It is often assumed that cloud computing systems offer computing resources on demand and service level agreement specifies the QoS which includes availability, security and most importantly performance-related measures [3,21,40]. Performance evaluation is particularly important since the effects of different resource management strategies and the choice of best system configuration can be essential for the correct deployment, maintenance, and operation of the cloud systems. The characteristics of cloud systems introduce the following differences compared to the modeling studies of traditional distributed systems [3,21]:

- The cloud centers are usually large-scale systems where the number of servers can go up to orders of hundreds or thousands. Traditional queuing theory analysis rarely considers systems of this size since well-known state space explosion problem is an important difficulty associated with the numerical procedures for steady-state solution of queuing systems [16,39].
- Since the cloud systems provide various services in a dynamic fashion, the cloud center should be able to cope with various loads and provide the expected quality of service.
- Different cloud systems can join together for a common purpose; however, cloud federations are beyond the scope of this study.

Use of simulation for the performance evaluation and comprehensive analysis of large-scale systems such as clouds may not be desirable [5,12], because the simulation runs required for steady state analysis last long and this limits severely the space of parameters that can be explored. Similarly, benchmarking or on the field experiments such as [30,41,44] suffer from the limitations introduced by the time and experimentation cost. Also, it is difficult to extrapolate the results of experiments for alternative scenarios. since a small change in configuration may cause significantly different measures. It is possible to use queuing theory with some approximations to study the performance of cloud systems analytically. Although analytical models are promising candidates, the characteristics mentioned above introduce some limitations for solution approaches and exact modeling of cloud systems become infeasible. In order to sufficiently present an analytical model for cloud systems, the analytical approach should be able to cope with large systems, and allow us to evaluate the systems under study for various parameters (especially various traffic loads).

Similar to many other multi-server systems, the availability of the cloud can be affected by server failures caused by various reasons. Since cloud-based applications aim to provide specific QoS and be accessible anywhere and anytime, the dependability becomes an important issue [11]. In the presence of server failures, the cloud can continue working with degraded performance, and in that sense the cloud systems should be considered as fault-tolerant systems with large numbers of servers. When fault-tolerant systems are considered, pure performance evaluation would cause overestimation of systems ability to serve and the resulting measures would be optimistic since they ignore the failure repair behavior of the servers. When the numbers of servers considered are significantly large, assuming that none of the servers will experience failures and focusing purely on performance would cause even more significant overestimations. Pure availability analysis, on the other hand, would be conservative and restrictive since different levels of performance are not taken into account [15,36]. Therefore, for realistic evaluation and analysis of cloud systems, a composite measure of performance and availability which is performability should be employed.

In this paper, performability analysis of cloud systems is considered in two dimensions where one state variable represents the number of operative servers and the second state variable represents the number of requests in the system. To cope with large-scale nature of cloud computing which introduces numerical difficulties since the size of the state space of the underlying Markov process becomes too large for exact solution approaches, a novel approximate solution approach is introduced. Unlike the existing approaches, the new approach allows us to consider the performance and availability together for the cloud systems even when the numbers of facility nodes are of the order of hundreds or thousands. The results obtained from the new approach are presented in comparison to the discrete event simulation results. Findings show that the analytical modeling approach presented works for large state spaces providing high degree of accuracy, and it is computationally significantly more efficient than the simulation approach. The paper is organized as follows: the next section presents the related studies on analytical modeling for performance and availability evaluation of cloud systems. Section 3 is about the model considered in this study. Section 4 introduces the solution approach, while Sect. 5 provides the details of the simulation process. Section 6 presents numerical results for the performability measures to address the accuracy and the efficacy of the new approach. Relevant conclusions are drawn in the final section.

2 Related work

Cloud computing systems have attracted considerable research interest for performance and availability evaluation and some studies have considered analytical modeling of cloud for pure performance or pure availability-based evaluation. However, because of the limitations mainly caused by the scale of the cloud systems, the performance-related issues have not been sufficiently considered together with server availabilities for large-scale systems in generic level. In other words, existing solution approaches are not able to handle monolithic two-dimensional Markov representations with servers in orders of hundreds to thousands.

The tendency of software to fail or cause a system failure after running continuously for a specific time period is referred to as software aging. In other words as the software gets older, it becomes more fragile and may require rebooting or reinstallation. A well-known approach to cope with this phenomenon is to follow a proactive fault management method and use Monitor, Analyse, Plan, Execute (MAPE) control cycles to look for performance degradation of software components. The process of restarting these components is an important part of software rejuvenation [28]. For the availability of cloud systems, a technique is proposed in [4] to model the aging process of a virtual machine manager of data centers under variable workload conditions. A time-based policy that adapts the rejuvenation timer to the workload condition is also presented.

The availability modeling and analysis of mobile clouds are considered in [25], and with the use of a hierarchical analytical modeling approach a sensitivity analysis study is performed which identifies the bottlenecks for system improvements in terms of availability. The average time between failures and repairs is considered to be exponentially distributed for hardware, update, and application dependent events, and continuous time Markov Chain analysis is employed successfully. The results presented show that for mobile clouds there is room for improvement on system availability by focusing on a reduced set of factors such as cloud infrastructure components.

In [11], availability models are presented for cloud systems based on the Eucalyptus architecture which is a popular open source software framework for IaaS systems [29]. Reliability block diagrams and Markov reward models are employed together to compute average available capacity in the presence of software- and hardware-related failures where the average failure and repair times considered are exponentially distributed.

Analytical models are presented in [40] for pure performance analysis. Cloud platforms are modeled as open Jackson networks to analyze the response time. Open Jackson queuing network-based models are used in [20] as well to characterize the service components in content-delivery-as-a-service (CoDaaS) platform. Using a stochastic reward nets-based analytical approach, in [3] similar interactions are considered for pure performance analysis of systems composed of thousands of resources. The term availability used in [3] is related to the rejection of a request when the queue is full rather than the availability of the servers. The general approach presented is useful to configure the data center parameters for large-scale systems from a pure performance point of view. Analytical Markovian models composed of distinct functional submodels are employed in [22] for pure performance analysis. Effects of various parameters such as the rate of user requests are analyzed in detail. Multi-stage systems with exponentially distributed service and inter-arrival times are considered in [10] as well; however, the main focus is on power efficiency of cloud computing systems while the cost in terms of response time is minimized.

The cloud computing systems are represented as M/M/m queuing models in [6]. The system is analyzed from a pure performance point of view; however, with the help of server speed and power consumption models, the expected service charge to a request is also provided. Numerical results are provided for up to eight server systems.

Similar to [6], stand-alone multi-server systems are employed in [21] as well to model cloud systems. Amazon Elastic Compute Cloud (EC2) is considered as case study and general probability distribution is preferred instead of exponential one. Stand alone M/G/m/m+r queuing system is employed to model cloud centers. Numerical results are presented in [21] for systems with up to 100 servers.

Composite measure of performance and availability provides more realistic evaluation of fault-tolerant systems. The effects of server failures are usually more evident for large-scale systems under heavy loads. In [43], cloud service response time is analyzed considering service and task failures. Probability distribution of service response time is derived; however, the framework presented does not consider steady-state analysis of large-scale cloud systems in detail. A hierarchical model is presented in [17] for systems with exponentially distributed inter-arrival, service, failure, and repair times. Specific models are considered where virtual machines are deployed in physical machines and this allows the successful use of hierarchical modeling approach proposed. The main contribution of this study is the reduction in number of states using hierarchical approach instead of a monolithic model. As the scale of the cloud increases, the monolithic model used to check the accuracy of the hierarchical one in [17] becomes intractable and does not produce results. For the hierarchical approaches, the accuracy and the efficacy of the approach are highly dependent on interactions among sub-models, and as the scale of the system increases to the orders of thousands, this may cause degradations in terms of accuracy. Instead, in our approach, a monolithic modeling approach is considered in generic level even for the cases where the system considered is large scale.

In this study, we assumed that the time between failures and repair times is exponentially distributed which means that the next break down is the result of some suddenly appearing failure, rather than gradual deterioration of servers which is the case mostly for the software-based failures [38]. The availability models considered in [4,11,17,25,43] as well assume exponentially distributed time between failures, which means software-based independent failures dominate the availability model; however, it should be noted that alternative distributions (e.g., weibull and gamma distributions for breakdowns and lognormal distribution for repairs) have also been reported [34].

Service times are also assumed to be exponentially distributed in this study. There are some applications in cloud systems where service times should be modeled using general distributions as suggested in [21]. However, analysis of some real traces on cloud systems such as Amazon S3 suggests that service time can also be modeled using exponentially distributed random variables [9,37]. The same assumption is employed to focus on various factors while mathematical tractability is provided with the assumption of exponential distribution for service times in studies such as [3,6,10,17,20,40,43]. In this study, as well the main focus is on analytical performance modeling of cloud systems in the presence of failures for more realistic evaluation while the number of serving nodes can go from few hundreds to thousands. Numerical results presented clearly show the accuracy and efficacy of the presented approach to model and solve such large-scale systems.

3 The system model

This study explores resource management in cloud computing, which is very important for efficient cloud system planning [22,32,40]. Similar to the studies such as [6,21,40], the model presented is a high level abstraction where the service facilities of cloud are considered in generic level. The service behavior of the model can be configured to represent IaaS Clouds where the service requests for various physical and/or logical resources are handled by systems which are large scale.

In [21], the cloud center is modeled as a queuing system with single task arrivals and a task buffer of finite capacity which is quite similar to our approach. It is assumed that the tasks sent to the cloud center are serviced within a suitable server and when the service is completed the task leaves the center. The servers are referred to as facility nodes, and with the assumption of specifying the service usage and obligations with service level agreements (SLA), the main focus is given to the resource management rather than the details of the stages of service procedure. The stages prior to the processing servers are abstracted using an incoming traffic flow to represent those stages. One of the gaps [21] tries to fill is the presentation of an analytical model for large-scale cloud centers. Systems with up to hundreds of servers are considered.

In [40], a queuing theory-based model is used to study computing service QoS in cloud computing, where the cloud architecture is modeled using well-known open Jackson networks of *M/M/m* servers. A number of systems are represented using feed forward, open queuing systems. The most critical stage of presented network of queues is the processing server (PS) which is modeled as a single queue multiple identical servers queuing system with exponentially distributed inter-arrival and service times just like the model presented in our study. The arrival process of the multi-server model is the throughput of the single server one. The model presented in [40] is used to scale the cloud system optimally to guarantee the QoS, and even though simulated predictions based on the model are not in real scale, they are defined as reliable and useful for generating an accurate approximation of the task response times to avoid exceeding the SLA.

M/M/m queuing model is employed in [6] as well to study optimal multi-server configuration for profit maximization in a cloud computing environment. The cost/profit optimization is modeled based on the performance in terms of average response time. In case the response time is less than the time agreed by SLA, the customers are fully charged. Otherwise, there is a penalty for late completion. Cloud computing service provider is defined as a collection of blades/processors/cores and referred in generic level as servers. The application environment is considered with specific workloads which include the task arrival rate and the average task execution requirement. The main focus is on optimization through use of a generic model with a common queue and multiple identical servers which consider resource contention and numbers of servers to maximise the profit.

In [26], actual cloud computing platforms such as Amazon EC2, IBM blue cloud, and private clouds, which come with large numbers of work nodes and cloud managers such as Eucalyptus, Open Nebula and Nimbus, are considered. The abstraction provided to represent these systems is an *M/M/m* +*D* queuing model. Commonly used job queuing systems such as Sun Grid Engine (SGE), Portable Batch System (PBS), or Condor are assumed to be used for a single common queue, and the jobs are provided resources in the form of virtual machines. Since the behavior of the queuing systems is modeled from a resource allocation and contention control point of view, the multi-server model with a service request queue, which is widely adopted for cloud computing in existing literature, is considered as a reasonable abstraction. The waiting time is investigated to satisfy the SLA in terms of response time. Systems with up to 25 servers are considered.

The queuing system considered in [43] for incoming subtasks with a common queue and a number of identical virtual nodes (referred to as servers) is quite similar to the one considered in this study. Furthermore in [31], a server site is modeled as *M/M/m/K* system, where m is the number of virtual machines assigned to the service provider. An analytical performance model consisting of stochastic sub-models which are composed of single queue multiple server systems is proposed for mobile cloud computing in [32]. Up to 40 physical machines are considered.

A generic multi-server system model is presented in this study for performability evaluation of cloud computing systems. Unlike the existing studies, the model and solution approach presented can handle large numbers of servers together with potential server failures. The multi-server queuing model abstracts away architectural/platform/hardware/implementation details; however, it preserves the process level and server availability contents adequate for performance and availability analysis in a context of queueing networks similar to studies in [3,6,10,17,20,21,26,40,43].

As identified in [14], interactive services are implemented requiring multiple components, and a single client request flows through a sequence of components, generating multiple sub-requests. However, when the cloud systems are considered in generic level, it is possible to provide an abstraction where these requests compete for the available resources. This study is the first one to model this contention with servers up to orders of thousands together with server availabilities.

When practical implementations are considered, embarrassingly parallel or pleasingly parallel applications are quite commonly used in various high performance, tightly or loosely coupled facilities. These applications require little or no effort to separate the main problem into a number of parallel tasks. Therefore, the computation can be easily separated into a number of parallel tasks that are executed on separate machines [42]. Cloud computing is one of the computing environments which are flexible, and cost effective with popular embarrassingly parallel applications such as Monte Carlo simulations, Basic Local Alignment Search Tool (BLAST) searches for bioinformatics, parametric studies, and image processing applications [18,33]. Similarly, when the heterogeneity of the servers is considered, one big advantage of using homogeneous servers is the ease of management. For example, OrionVM, which is an Australian IaaS provider, builds their cloud infrastructure utilizing a homogeneous node architecture, where each node has computing, memory and storage assets. They mention the ability of delivering greater performance at a reduced cost by utilizing homogeneous servers [1]. Homogeneous virtual machines are considered for analytical modeling in [8] as well, and in [24] a new algorithm is proposed for routing in systems with identical nodes such as data servers.

It is assumed that the requests from the users arrive to the available facility nodes referred to as servers in this study. The resources can be employed for various appli-

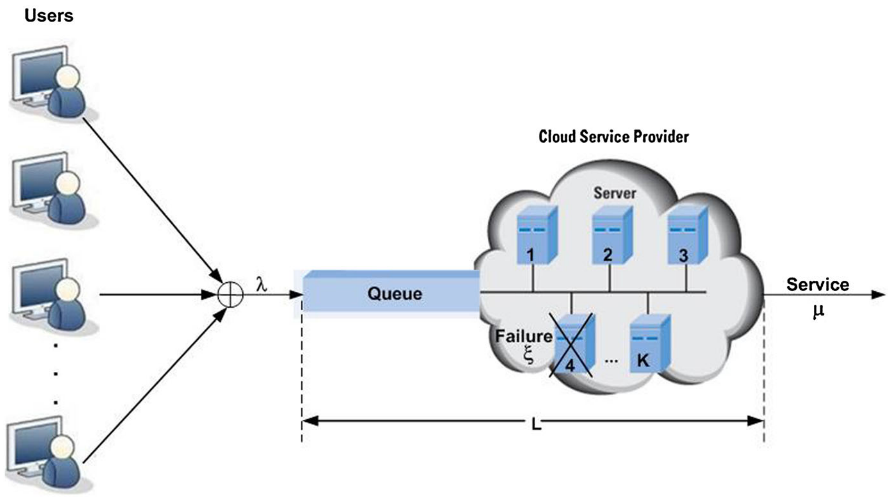


Fig. 1 Queuing model of the proposed system

cations such as web servers, database servers and others. The requests arrive to the servers from different users independently and consecutively. As explained above, the requests follow Poisson distribution which means that the inter-arrival time of requests is exponentially distributed with arrival rate of λ. In addition, the service times for each request are independent and identically distributed with exponential distribution with the service rate μ. There are K number of servers in the system as shown in Fig. 1. The capacity of the system is L which means that the buffer size for incoming requests is equal to $L - K$. The queuing discipline is assumed to be First Come First Serve (FCFS).

When a user has a request from the cloud system, in case at least one of the servers are idle, the request will be handled by one of the idle servers. Instead, if all the servers are busy with other requests and the queue is sufficient to accommodate the incoming request, it will join the queue. If the queue is full, the incoming request will be rejected. The servers considered are prone to failures where operative periods are exponentially distributed with a mean failure time of $1/\xi$. At the end of an operative period, when server $k(k = 1, \ldots, K)$ fails, it requires an exponentially distributed repair time with mean $1/\eta$. It is also assumed that the operative servers cannot be idle if there are requests waiting, and repair facility cannot be idle if there are failed servers waiting for repair. Services that are interrupted by failures are eventually resumed from the point of interruption or repeated with re-sampling (perhaps on a different server). All inter-arrival, service and repair times are independent of each other.

Figure 2 shows the state diagram of the proposed system. The states of the system are described by *i* and *j*, specifying the server configuration and number of requests in the system, respectively. Thus, $P_{i,j}$s are steady-state probabilities of having *i* number of operative servers and *j* number of requests in the system. In Fig. 2, there are $K + 1$ server configurations ($i = 0, 1, \ldots, K$). These $K + 1$ configurations are used to represent the operative states of the available servers in the system. L represents the

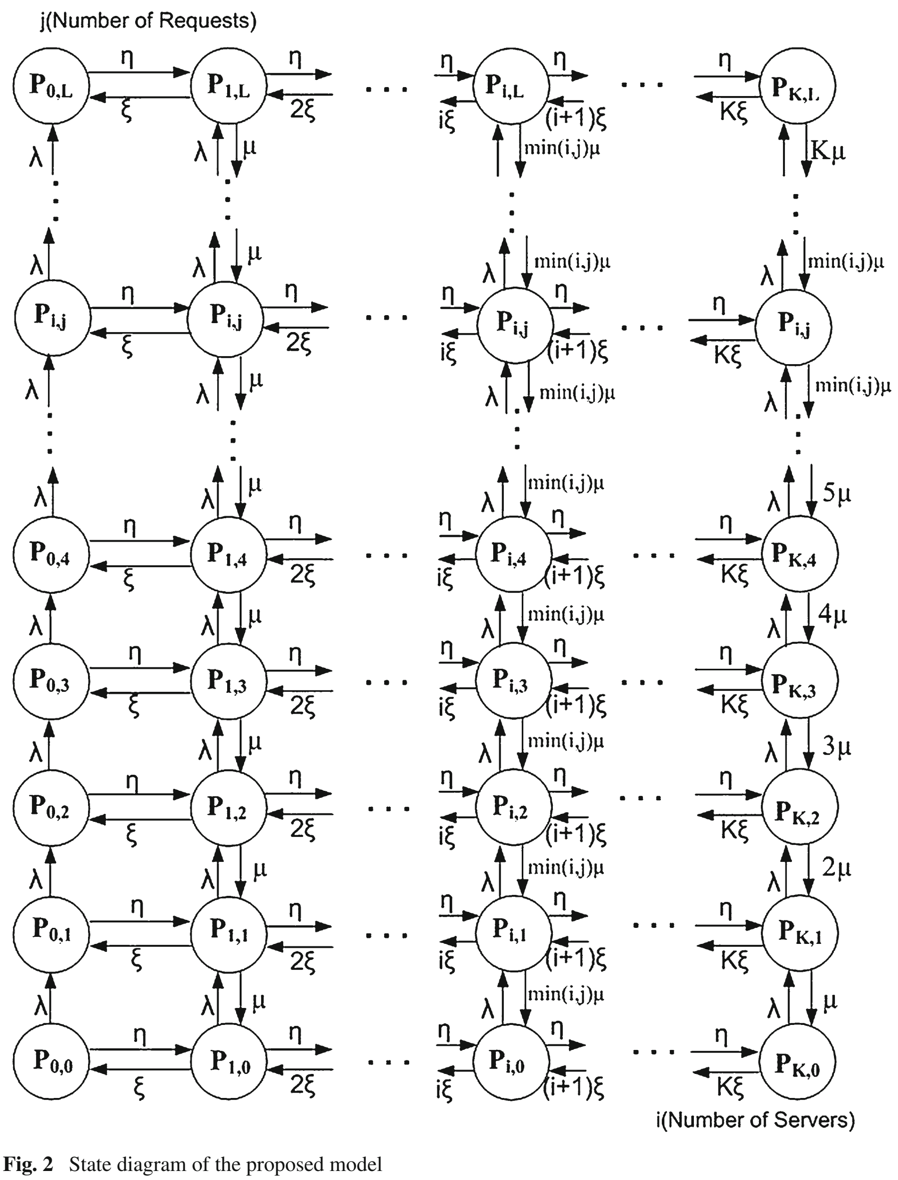


Fig. 2 State diagram of the proposed model

maximum number of requests in the system (the requests being served as well as the requests in the queue). The downward transitions indicate that the requests are being served with service rate μ which depends on the number of operative servers as well as the number of requests in the system. On the other hand, upward transitions take place because of new arrivals with rate λ to the system. The lateral transitions indicate failures and repairs of the servers. The lateral transition to left hand side shows failures with rate ξ and transitions to the right represent repair of failed servers with rate η.

4 The analytical solution approach

The analytical solution approach introduced is able to handle large-scale systems in the presence of failures and repairs. The efficacy of the approximation is significantly superior to simulation while the accuracy is within the desired levels.

There are various analytical approaches that can be employed for the solution of two-dimensional models, where one of the state variables is used to represent the number of requests in the system and the other one is used to represent the number of available servers. Matrix-geometric and Spectral-expansion methods are two examples for exact solution of these systems [19,27]. However, when the number of servers is large, exact solutions cannot handle the resulting state space models. For cloud computing systems, two main steps are considered to deal with numbers of servers in orders of hundreds to thousands. First, an approximation is provided for handling the availability of the servers and the obtained probabilities of server configurations are employed to compute the approximate state probabilities of the two-dimensional model considered. Following this, the balance equations are employed together with the approximate state probabilities computed in the first step and an iterative procedure is chosen for obtaining high accuracy.

4.1 Approximate decomposition

The initial decomposition of the state variables will consider the state probabilities of server configurations. In other words, system in Fig. 2 is first considered for pure availability modeling. Since the failure and repair-related transitions are lateral transitions, by considering the continuous time Markov chain (CTMC) for pure availability as shown in Fig. 3, we can compute the probability for each operative state, which would give us the approximate sum of probabilities in each column of Fig. 2.

Considering the CTMC for pure availability modeling, the sum of probabilities in each column (P_i's) can be given as:

$$P_i' = \frac{\left(\frac{\eta}{\xi}\right)^i}{i!\sum_{k=0}^{K}\frac{\left(\frac{\eta}{\xi}\right)^k}{k!}} \tag{1}$$

where $P_i' = \sum_{j=0}^{L} P_{i,j}, i = 0, 1, \ldots, K$.

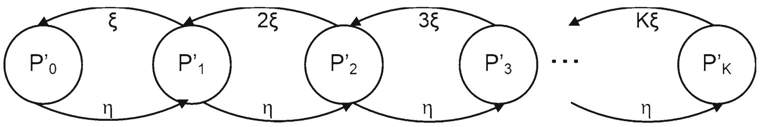


Fig. 3 CTMC for computing sum of probabilities in each column

The normalizing equation for the state probabilities of the two dimensional system is used in the decomposition process and can be given as:

$$\sum_{i=0}^{K}\sum_{j=0}^{L} P_{i,j} = 1 \tag{2}$$

Once approximate sum of probabilities in each column is computed, it is also possible to consider the transitions in each column individually only with one step upward and one step downward transitions. In this case, the transitions for failure, and repairs are not considered. The $P_{i,j}$s can then be expressed in terms of $P_{i,0}$.

$$P_{i,j} = \begin{cases} \frac{\rho^j}{j!} \cdot P_{i,0} & 0 \le j \le i \\ \frac{\rho^j}{j! i^{j-i}} \cdot P_{i,0} & i+1 \le j \le L \end{cases} \tag{3}$$

where $\rho = \frac{\lambda}{\mu}$ and $1 \le i \le K$. Then, using the P'_is as the sum of all probabilities in a column where the columns are numbered as $i = 0, 1, \ldots, K$, the following expression can be derived:

$$P'_i = \begin{bmatrix} \frac{\rho^0}{0!} + \frac{\rho^1}{1!} + \frac{\rho^2}{2!} + \cdots + \frac{\rho^i}{i!} + \\ \frac{\rho^{i+1}}{i! i^1} + \frac{\rho^{i+2}}{i! i^2} + \frac{\rho^{i+3}}{i! i^3} + \cdots + \frac{\rho^L}{i! i^{L-i}} \end{bmatrix} P_{i,0} \tag{4}$$

Equation 4 can be generalized as:

$$P'_i = \left[\sum_{j=0}^{i} \frac{\rho^j}{j!} + \sum_{j=i+1}^{L} \frac{\rho^j}{i! i^{j-i}} \right] P_{i,0} \tag{5}$$

for $i = 1, 2 \ldots, K$.

From Eqs. 1 and 5, with some simplifications, $P_{i,0}$ can be computed as:

$$P_{i,0} = \frac{\left(\frac{\eta}{\xi}\right)^i}{i! \sum_{k=0}^{K} \frac{\left(\frac{\eta}{\xi}\right)^k}{k!}} \left[\sum_{j=0}^{i} \frac{\rho^j}{j!} + \frac{(i^{L-i}\rho^{i+1}) - \rho^{L+1}}{i! i^{L-i}(i-\rho)} \right]^{-1} \tag{6}$$

for $i = 1, 2 \ldots, K$.

The general expression for the approximate state probabilities where $i = 1, 2 \ldots, K$ can then be obtained using Eqs. 5 and 6 as follows:

$$P_{i,j} = \begin{cases} \dfrac{\left(\frac{\eta}{\xi}\right)^i}{i! \sum_{k=0}^{K} \dfrac{\left(\frac{\eta}{\xi}\right)^k}{k!}} \left[\sum_{j=0}^{i} \frac{\rho^j}{j!} + \frac{(i^{L-i} \rho^{i+1}) - \rho^{L+1}}{i! i^{L-i} (i - \rho)} \right]^{-1} \frac{\rho^j}{j!} \\ j = 0, 1, 2, \ldots, i \\ \dfrac{\left(\frac{\eta}{\xi}\right)^i}{i! \sum_{k=0}^{K} \dfrac{\left(\frac{\eta}{\xi}\right)^k}{k!}} \left[\sum_{j=0}^{i} \frac{\rho^j}{j!} + \frac{(i^{L-i} \rho^{i+1}) - \rho^{L+1}}{i! i^{L-i} (i - \rho)} \right]^{-1} \frac{\rho^j}{j! i^{j-i}} \\ j = i + 1, i + 2, \ldots, L \end{cases} \tag{7}$$

The initial approximate state probabilities for the system considered can be computed using Eq. 7. In this equation, the state probabilities are expressed in terms of arrival, service, failure, and repair characteristics of the system. These initial approximations are in turn fed into an iterative procedure to provide higher accuracy for the performance measures of interest. The initial approximations are quite important for fast convergence of the iterative method similar to the studies in [7,35].

The left most column in Fig. 2 is a special one since downward transitions are not present in case there are no operative servers. An approximation is employed for the probabilities in this column using P'_0 which assumes that λ/η requests join the queue while the server is being repaired.

4.2 Balance equations and iterative solution

Considering lateral transitions for computation of P'_is, using them as sums for each column, and then focusing on the distribution of this sum for computation of $P_{i,j}$s is an approximation, since lateral and horizontal transitions are not considered together. There are well-known analytical solution approaches which can consider the lattice in Fig. 2 for exact solution, and the major ones such as the Matrix-geometric and Spectral-expansion methods have been reviewed in [19,27]. However, these methods cannot be employed for exact solution of models representing cloud systems, since they are not capable of handling hundreds to thousands of servers.

In this study, the approximate state probabilities computed using Eqs. 1–7 are employed together with the balance equations of the lattice strip presented in Fig. 2 for an iterative solution approach, in order to incorporate the effects of lateral and horizontal transitions together. The transitions in Fig. 2 lead to the following balance equations:

$i = 0;$

$$P_{0,0} = \frac{\xi}{\xi + \lambda} P_{1,0} \tag{8}$$

$$P_{0,j} = \frac{\xi P_{1,j} + \lambda P_{0,j-1}}{\xi + \lambda}, \quad j = 1, 2, \ldots, L-1 \tag{9}$$

$$P_{0,L} = \frac{\xi P_{1,L} + \lambda P_{0,L-1}}{\eta} \tag{10}$$

$1 \leq i < K;$

$$P_{i,0} = \frac{(i+1)\xi P_{i+1,0} + \eta P_{i-1,0} + \mu P_{i,1}}{i\xi + \eta + \lambda} \tag{11}$$

$$P_{i,j} = \frac{(i+1)\xi P_{i+1,j} + \lambda P_{i,j-1} + \eta P_{i-1,j} + (j+1)\mu P_{i,j+1}}{j\mu + i\xi + \eta + \lambda}, \quad 1 \leq j < i \tag{12}$$

$$P_{i,j} = \frac{(i+1)\xi P_{i+1,j} + \lambda P_{i,j-1} + \eta P_{i-1,j} + i\mu P_{i,j+1}}{i\mu + i\xi + \eta + \lambda}, \quad i \leq j < L \tag{13}$$

$$P_{i,L} = \frac{(i+1)\xi P_{i+1,L} + \lambda P_{i,L-1} + \eta P_{i-1,L}}{i\mu + i\xi + \eta} \tag{14}$$

$i = K;$

$$P_{K,0} = \frac{\eta P_{K-1,0} + \mu P_{K,1}}{K\xi + \lambda} \tag{15}$$

$$P_{K,j} = \frac{\lambda P_{i,j-1} + \eta P_{i-1,j} + (j+1)\mu P_{i,j+1}}{j\mu + i\xi + \lambda}, \quad 1 \leq j < i \tag{16}$$

$$P_{K,j} = \frac{\lambda P_{i,j-1} + \eta P_{i-1,j} + i\mu P_{i,j+1}}{i\mu + i\xi + \lambda}, \quad i \leq j < L \tag{17}$$

$$P_{K,L} = \frac{\lambda P_{K,L-1} + \eta P_{K-1,L}}{K\mu + K\xi} \tag{18}$$

The iterative procedure is employed to accurately calculate the $P_{i,j}$s, and then compute various performance measures in the presence of failures and repairs such as mean queue length (MQL), throughput (γ) and response time (RT). These measures are very important for realistic evaluation of cloud systems. The iterative procedure can be given as follows:

Iterative Procedure Steps to follow for accurate $P_{i,j}$s

1: Approximate steady state probabilities are computed using Eqs. 1–7. These computations may not give accurate approximations, however they are particularly useful to have faster convergence.

2: The approximate $P_{i,j}$s are used for the calculations of performance measures such as mean queue length, throughput, and response time as

$$\text{MQL} = \sum_{i=0}^{K}\sum_{j=0}^{L} i P_{i,j}$$

$$\gamma = \sum_{i=1}^{K}\sum_{j=0}^{L} \mu P_{i,j}$$

$$\text{RT} = \frac{\text{MQL}}{\gamma}$$

Let's refer to the performance measures computed in this step as MQL_{old}, γ_{old}, and RT_{old}.

3: The balance equations (Eqs. 8–18) are used to calculate the correct steady state probabilities.

4: Performance measures MQL_{new}, γ_{new}, and RT_{new} are calculated using the new state probabilities obtained in step 3.

5: One of the performance measures are chosen to check the accuracy. In case MQL is chosen, lets define $\epsilon = |\text{MQL}_{\text{new}} - \text{MQL}_{\text{old}}|$.

6: In case the ϵ value is less than a prespecified, and acceptable threshold value, the iteration will be finalized and the most recent steady state probabilities will be used to compute all the required performance measures. Otherwise the procedure will assign new values of performance measures to old ones (e.g., $\text{MQL}_{\text{old}} = \text{MQL}_{\text{new}}$) and continue from step 3.

This iteration method stops when convergence criterion is satisfied, (i.e., the change in the average queue sizes is less than ϵ) as explained in the procedure. The approach used to stop the iteration is commonly used in the literature and two examples can be found in [7,35]. In this study, $\epsilon = 0.001$ is used. Since the minimum values of the quantities dealt with such as MQL and γ are around 50 requests in the computations, a deviation of 0.002 % is a good approximation for high accuracy. The accuracy and the efficacy of the analytical solution approach are discussed in Sect. 6.

5 Simulation

Simulation modeling is used for the validation of the analytical model and the iterative solution approach in terms of accuracy as well as efficacy in this study. For this purpose, the results obtained from the analytical model and the iterative solution approach are presented comparatively with the results from a simulation software written in C++ language and validated to simulate the actual system. The simulation program developed is validated using well-known queuing theory models such as *M/M/1*, and *M/M/c* as well as results from the literature [2,23].

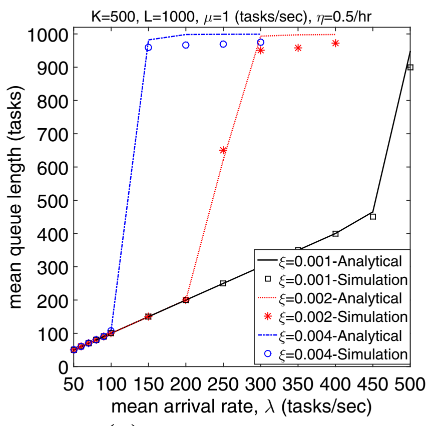
The program employed is a discrete event simulation program. An event-based scheduling approach is taken into account which depends on the events and their effects to the system state. Relative precision which is one of the most commonly used stopping criterion is employed for the simulation. In this method, the simulation is stopped at the first checkpoint when the condition $\delta \leq \delta_{\max}$ is met. Where $\delta_{\max}$ is the maximum acceptable value of the relative precision of confidence intervals at the $100\,(1-\alpha)\,\%$ significance level, $0 < \delta_{\max} < 1$. The results obtained from the simulations are within the confidence interval of 5 % with a confidence level of 95 %; therefore in the simulation, both default values for α and δ are set to 0.05.

6 The accuracy and efficacy of the analytical approach

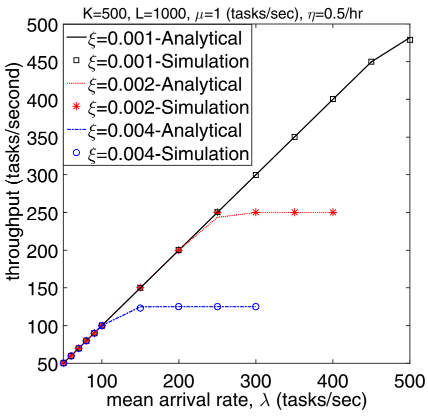
In this section, results are presented for the performability analysis of cloud centers. The numbers of servers considered for numerical results are of the orders of hundreds to thousands for typical cloud centers [3,21]. Results are presented for systems with up to 5000 servers ($K = 5000$, $L = 6000$). In addition, numerical results are presented comparatively with the results obtained from discrete event simulation. To demonstrate the effectiveness and the accuracy of the proposed analytical model, numerical results are presented for MQL, throughput, and response time. The CPU times for the computations using the analytical approach are also presented comparatively with the CPU times of discrete event simulation in an attempt to show the efficacy of the proposed approach. It is well known that analytical methods are in general faster than simulations tools; however, since the proposed analytical solution is iterative, it is important to show that the proposed approach is significantly more efficient than the simulation and computation times are not even comparable. The studies similar to the ones presented in [21,40] use product-form solutions and non-iterative approximations. They assume failure-free systems and avoid fault tolerant nature of cloud. Non-iterative solutions with much smaller state spaces deal with simpler computations; therefore, their computation times are expected to be shorter. However, avoiding the potential failures especially for systems with large numbers of servers may cause overestimations of system capacity as mentioned in [15,36]. All the numerical results presented are obtained using workstations with Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz, 16GB RAM, and 64-bit operating system.

In Fig. 4, MQL, throughput, and response time results are presented as a function of mean arrival rate (λ) for both proposed analytical model and simulation. A system with $K = 500$, and $L = 1000$ is considered where $\mu = 1$/s, and $\eta = 0.5$/h. The parameters for inter-arrival and service times may vary according to the application area the cloud system is being used for. In this study, fine grained applications such as web services are chosen similar to the studies in [9,10,40]; however, the approach presented is flexible and can be used for various applications. Similarly, the average failure times of the servers considered are chosen from studies considering the availability of cloud computing systems such as [11,25]. Time between failures considered is, therefore, taken as 250, 500, and 1000 h for each server (ξ is 0.004, 0.002, and 0.001, respectively).

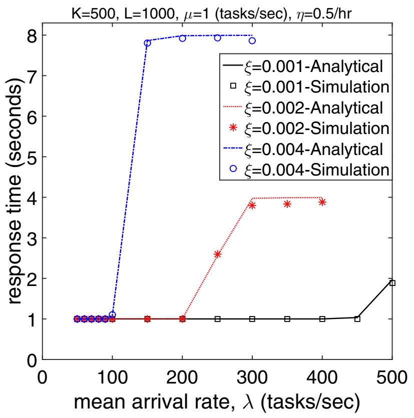
The effects of failures on the QoS of the system can be quite significant. When lower values of ξ are considered, unless the system is heavily loaded with high arrival rates, an arriving request is likely to receive service from such a system with high numbers of serving nodes. In other words, decreasing the time between failures of the



(a) Mean Queue Length



(b) Throughput



(c) Response Time

Fig. 4 The performability results as a function of λ for $K = 500$ and $L = 1000$ with different ξ

servers increases both MQL and response time of the systems. In case of high failure rates, or loaded systems, the incoming requests start to accumulate in the queue, and the throughput value saturates according to the average value of operative servers which is highly dependent on the failure, and repair rates. For example, in Fig. 4 when relatively lighter traffic loads are considered such as $\lambda = 100$ tasks/s, the MQL of systems with $\xi = 0.001, \xi = 0.002, \xi = 0.004$ is 100, 100, 109.204, respectively. When heavier loads are considered such as $\lambda = 250$ tasks/s, the MQL values become 250, 621.582, 998.983 for systems with $\xi = 0.001, \xi = 0.002, \xi = 0.004$, respectively. Let us define u as the utilization of a system without failures, where $u = \lambda/(\mu K)$. Similarly, in Fig. 4, the MQL value is around 300 when $\xi = 0.001$ and utilization $u = 0.6$ is considered. On the other hand, increasing failure rate to $\xi = 0.002$ and $\xi = 0.004$ when $u = 0.6$, the MQL values increases to 993.37 and 999.27, respectively. With the increased failure rates, the average number of requests in the system becomes same as the maximum capacity of the system L. The throughput of the system increases as λ increases; however, the throughput saturates depending on the number of available servers. As expected, Fig. 4b shows that because of the average number of operative servers, the saturation values are lower for the systems with higher failure rates. Similar behavior is observed for response times.

To further emphasize the accuracy of the proposed model and the solution approach, Table 1 presents MQL, throughput and response time results comparatively with the simulation results. The discrepancies between the new approach and simulation are presented in Table 1 as well. The maximum discrepancies for MQL, throughput and response times are less than 4.98, 0.6, and 0.38 %, respectively, which is less than the confidence interval of the simulation specified as 5 %.

A system with greater number of servers $K = 1000$ and larger capacity $L = 2000$ is considered for Fig. 5. The rest of the parameters are considered same as the ones used in Fig. 4 for a fair comparison.

The results presented in Fig. 5 have similar trends with results in Fig. 4. For the systems considered, in order to satisfy the explicit ergodicity condition, which requires that the offered load is less than the average number of operative processors, the parameters should satisfy the condition given as $\lambda/\mu < K\eta/(\eta + \xi)$ [19]. In cases where this condition is not satisfied, the system becomes highly saturated. When MQL, throughput and response time results are compared for Figs. 4 and 5, the numerical results are not significantly different. The system with larger number of servers does not perform significantly better mainly because of the effects of failures and limited repair facilities which are chosen specifically to show that considering large-scale multi-server systems such as cloud purely from performance point of view can cause significant overestimations. In other words, because of the failures in the system, the effective number of operative servers does not vary greatly when systems with large numbers of servers such as cloud are considered and the repair facilities are not in desirable levels.

The response time eventually increases when systems with larger overall capacities are considered. For instance, the maximum response time is 7.998 s when the $\xi = 0.004$ for $K = 500$ and $L = 1000$ as shown in Fig. 4c. On the other hand, Fig. 5c clearly shows that the maximum response time for $K = 1000$ and $L = 2000$ is 15.889 s when the $\xi = 0.004$ for the proposed model. This is mainly because of the larger queuing capacity of the latter system which causes longer waiting times. The results shown in Figs. 4 and 5 indicate that the proposed model makes it possible to handle larger state spaces accurately; however, for the system with $K = 1000$, and $L = 2000$ as well numerical results and discrepancies are presented in Table 2. In Table 2, the maximum discrepancies for MQL, throughput and response times are less than 4.75, 1.9, and 2.5 %, respectively. These results clearly show that even when systems with one thousand servers are considered in the presence of failures, the maximum discrepancy between the analytical approach and the simulation is less than 5 % which is the confidence interval of the simulation.

The proposed analytical model is based on an approximation of initial conditions by considering certain behavior of system in terms of performance and availability individually, and then using an iterative approach which is heavily based on $(K + 1) \times (L + 1) + 1$ number of equations and same number of unknowns. For example, for

Table 1 Comparison of the results for $K = 500$, $L = 1000$, $\xi = 0.001$

λ	Mean queue length (tasks)			Throughput (tasks/s)			Response time (s)		
	Simulation	Analytical	Discrepancy (%)	Simulation	Analytical	Discrepancy (%)	Simulation	Analytical	Discrepancy (%)
50	50.0004	50	0.0008	50.0145	50	0.029	0.9997	1	0.03
60	59.9985	60	0.0025	60.0166	60	0.0276	0.9997	1	0.03
70	69.9938	70	0.0088	70.0091	70	0.013	0.9998	1	0.02
80	79.9969	80	0.0038	80.0132	80	0.0165	0.9998	1	0.02
90	90.0029	90	0.0032	90.0229	90	0.0254	0.9998	1	0.02
100	99.9806	100	0.0194	100.0085	100	0.0085	0.9997	1	0.03
150	149.9888	150	0.0074	150.0275	150	0.01833	0.9997	1	0.03
200	200.0032	200	0.0016	200.044	200	0.022	0.9998	1	0.02
250	250.0101	250	0.0040	250.0576	250	0.02304	0.9998	1	0.02
300	299.9932	300	0.0022	300.061	300	0.0203	0.9998	1	0.02
350	349.9722	350	0.0079	350.0612	350	0.0174	0.9997	1	0.03
400	400.0162	400.014	0.0005	400.0982	400	0.02455	0.9998	1.0003	0.0234
450	450.4805	465.031	3.1289	450.1111	449.827	0.0631	1.0008	1.0337	3.1920
500	899.8818	947.036	4.9791	479.1148	481.91	0.5800	1.8908	1.9651	3.7844

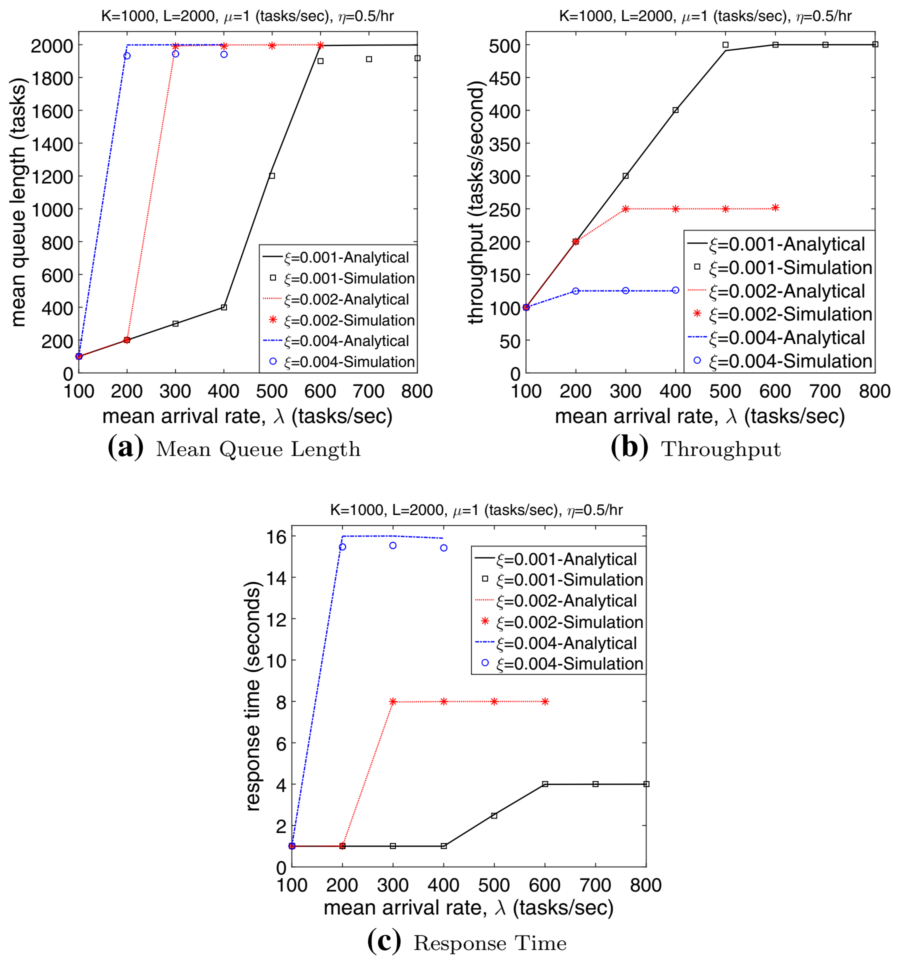


Fig. 5 The performability results as a function of λ for $K = 1000$ and $L = 2000$ with different ξ

the results presented in Table 2 the number of states to be considered is two million. Since the system to be solved is large and an iterative approach is employed, the processing times are also presented numerically in comparison with the processing times of the simulation. The CPU times for systems with $K = 500$, $L = 1000$ and $K = 1000$, $L = 2000$ are presented in Tables 3 and 4, respectively.

The results presented for CPU times clearly show the computational efficiency of the proposed approach compared to the simulation in both Tables 3 and 4. For instance, in Table 3 the maximum CPU time for simulation is 138.80 h for $K = 500$, $L = 1000$ whereas the maximum CPU time of the analytical approach is less than 2.1 s. When the number of states is increased, the CPU time of the analytical approach also increases as expected. However, the same behavior is observed for the simulation, since the scale of the system to be considered increases. When the system with $K = 1000$ and $L = 2000$ is considered, the CPU time of the analytical approach is still not very

Table 2 Comparison of the results for $K = 1000$, $L = 2000$, $\xi = 0.001$

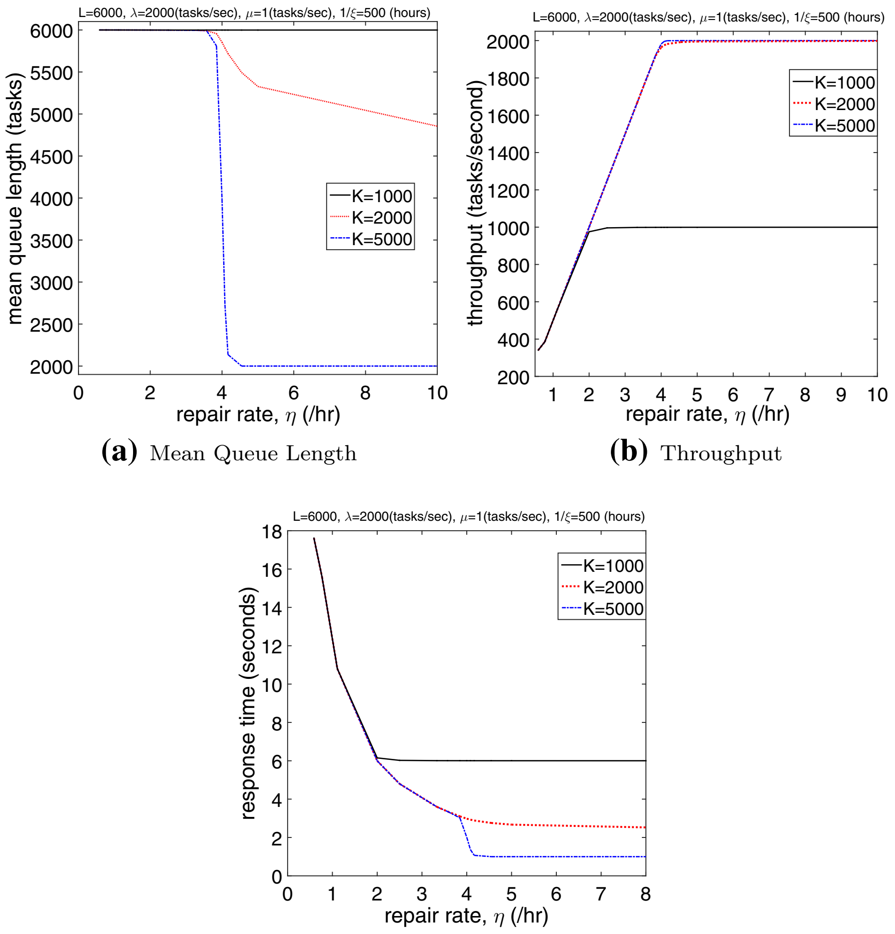
λ	Mean queue length (tasks)			Throughput (tasks/s)			Response time (s)		
	Simulation	Analytical	Discrepancy (%)	Simulation	Analytical	Discrepancy (%)	Simulation	Analytical	Discrepancy (%)
100	99.9814	100	0.0186	100.0122	100	0.0122	0.9997	1	0.03
200	199.9898	200	0.0051	200.0391	200	0.01955	0.9998	1	0.02
300	299.9952	300	0.0016	300.0647	300	0.0215	0.9998	1	0.02
400	399.9655	400.009	0.0108	400.0631	400	0.0157	0.9998	1.0002	0.0222
500	1199.9998	1244.97	3.6121	500.1051	490.917	1.8716	2.4739	2.5360	2.4490
600	1900.0181	1994.62	4.7428	500.146	500	0.0292	3.9998	3.9892	0.2647
700	1911.0324	1997.45	4.3263	500.1713	500	0.0342	3.9998	3.9949	0.1226
800	1918.913	1998.32	3.9736	500.5204	500	0.1040	3.9999	3.9966	0.0815

Table 3 Comparison of CPU times for $K = 500$, $L = 1000$

λ	$K = 500$, $L = 1000$, $\xi = 0.001$		$K = 500$, $L = 1000$, $\xi = 0.002$		$K = 500$, $L = 1000$, $\xi = 0.004$	
	Analytical (s)	Simulation (h)	Analytical (s)	Simulation (h)	Analytical (s)	Simulation (h)
50	1.983	0.8591	1.883	0.8591	1.897	1.0609
60	1.875	1.0537	1.897	2.0837	1.876	1.0609
70	1.796	1.2906	1.891	3.6852	1.873	1.2892
80	1.851	1.5346	1.895	5.6681	1.867	1.5349
90	1.852	1.7946	1.873	1.8337	1.885	1.7983
100	2.058	2.0837	1.88	2.1155	1.882	2.0907
150	1.942	3.6852	1.88	3.7425	1.888	3.8006
200	1.885	5.6681	1.884	5.8106	1.881	69.3252
250	1.882	8.0025	1.884	8.3019	1.892	69.2339
300	1.838	10.7599	1.892	11.2788	1.867	61.0211
350	1.851	13.8991	1.868	138.8009		
400	1.96	17.3522	1.87	97.0007		
450	1.852	20.9723				
500	1.891	36.4614				

Table 4 Comparison of CPU times for $K = 1000$, $L = 2000$

λ	$K = 1000, L = 2000, \xi = 0.001$		$K = 1000, L = 2000, \xi = 0.002$		$K = 1000, L = 2000, \xi = 0.004$	
	Analytical (s)	Simulation (h)	Analytical (s)	Simulation (h)	Analytical (s)	Simulation (h)
100	14.954	3.413796	14.956	3.442453	14.955	3.387297
200	14.947	8.299431	14.943	8.449214	14.847	8.823661
300	14.93	14.74293	14.915	15.02022	14.874	200.7832
400	14.898	22.76573	14.943	23.4686	14.682	200.7715
500	15.08	32.2715	14.932	34.0433		
600	15.032	43.40501	14.897	350.8363		
700	15.166	56.41779				
800	15.068	317.4744				



(a) Mean Queue Length **(b)** Throughput **(c)** Response Time

Fig. 6 The performability results of the proposed system as a function of η for different K

extensive with maximum less than 16 s. However, to perform the same computations using simulation took up to 350 h. In this sense, our approach is efficient in evaluation of large-scale fault tolerant systems such as clouds.

To show the importance of repair facilities, in Fig. 6 MQL, throughput and response time of various systems are presented as a function of repair rate. Systems with $K = 1000, 2000$, and 5000 are considered where $L = 6000$, $\mu = 1$/s, and $1/\xi = 500$ h. Figure 6a shows that the mean number of jobs in the system is around the system capacity for low values of the η (e.g., $\eta = 2$). However, MQL results start to decrease after $\eta = 4$. This is mainly because of the quick recovery of the servers in the system which causes increased numbers of servers to be available. For instance, when $\eta = 10$, the MQL results are 5999, 4854.6 and 2000 for $K = 1000$, $K = 2000$, $K = 5000$, respectively. Similar trends are observed for the response time and throughput as well.

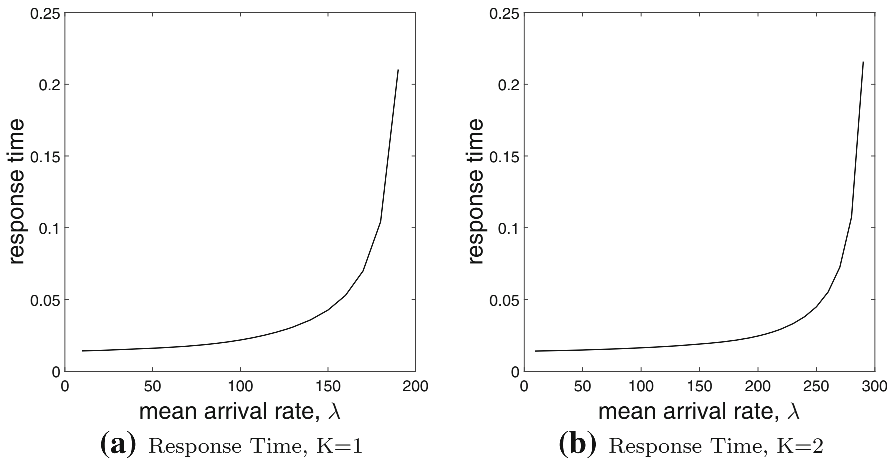


Fig. 7 Analytical model results to be compared with the experimental results presented in [40]

The throughput increases where the response time decreases dramatically as the repair rate increases. Both throughput and response time in turn saturate to a value depending on the number of servers in each system. The results in Fig. 6 show that the shorter the recovery time $1/\eta$, the lower MQL, response time and higher the throughput for each case. More importantly, the results clearly show that when the repair facilities are sufficient, using large numbers of servers will increase the QoS significantly.

To validate the model further, the experiments and the approach used to validate the usefulness of the model presented in [40] are adopted. In [40], the model to represent the cloud systems is implemented using the open-source cloud platform OpenStack, which can be used for adding virtual machines as computational resources. Openstack is used to create virtual servers with an entering server, four servicing nodes, a MySQL database with a database server as well as output and client servers. Each of the virtual machines has 4 GB RAM and two cores of an AMD processor, which runs at 2.1 GHz. In [40], the plots which show the results of the open queuing network simulation and the empirical results of the experiments have very similar shapes. It is clearly explained that although it is quite difficult to have absolute similarities for the values of the response times in simulation and test bed, the similarities in shapes of the plots representing the simulation results and empirical results prove the "good behavior" of the model. In this study, the following figures are provided for the same purpose. Similar to the test-bed and the simulation setup presented in [40], systems with one and two servers are chosen. The other parameters are $\mu = 100/\text{h}$, $1/\xi = 1000$ h, $1/\eta = 2$ h, and $L = 1000$.

The results presented in Fig. 7a, b are computed in the presence of failures. However, it is still possible to clearly see the similarities explained in [40] when the Fig. 7a, b are compared with the figures presented for results obtained through the test-bed, as well as simulation in [40].

Overall, the numerical results presented in this section show the importance of considering the performance together with factors affecting availability of the servers, such as failure and repair behavior for cloud systems. The results also show the efficiency of the proposed analytical model while the discrepancy of the numerical results is less than the confidence interval of the discrete event simulation employed.

7 Conclusions

The analytical model considered is similar to the ones presented in [6,21]; however unlike these studies, the monolithic system model presented can consider availability of the serving nodes as well. The numerical results presented show that the numerical solution approach can handle 5000 servers where the queue capacity is 6000 tasks as opposed to the existing studies which present numeric results for systems up to 100 serving nodes. This shows that the solution approach can handle up to 30,000,000 states of the monolithic model presented. The comparative results presented for up to 1000 servers and 2000 overall queue capacity (2,000,000 states) show that the discrepancy between the simulation and the analytical model presented is less than 5 %, while in terms of efficacy, the analytical model outperforms the simulation significantly. Especially for loaded and large-scale cases, the simulation times are quite extensive (up to 350 h) as expected.

The key contributions of this study are:

1. A generic approach is presented for large-scale homogeneous multi-server systems which is useful for end-to-end performability analysis and can be easily adopted to variety of cloud services. The model and solution approach can be used in capacity planning, optimization of cloud configuration, forecasting, and bottleneck analysis.
2. Since the cloud systems are considered as fault-tolerant systems, the effects of failures on the overall performance can be analyzed effectively. That means capacity planning can also involve possibility of using redundant servers, and optimization of repair facilities.
3. To the best of our knowledge, this study is the first one to consider hundreds to thousands of servers in the presence of failures with one level monolithic models where the accuracy and the efficacy of the approach are validated through comparison with results obtained by simulation runs.

The models developed, and solution approach are flexible and they can be used for performability evaluation of similar large-scale systems with various characteristics. The parameters used in this study can easily be altered and analysis of systems with various loads, service times, and availability-related characteristics can be considered.

References

1. Andrikopoulos V, Binz T, Leymann F, Strauch S (2013) How to adapt applications for the cloud environment. Computing 95(6):493–535
2. Banks J, Carson J, Nelson B, Nicol DM (2000) Discrete-event system simulation. Prentice-hall, Englewood Cliffs
3. Bruneo D (2014) A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems. IEEE Trans Parallel Distrib Syst 25(3):560–569
4. Bruneo D, Distefano S, Longo F, Puliafito A, Scarpa M (2013) Workload-based software rejuvenation in cloud systems. IEEE Trans Comput 62(6):1072–1085

5. Calheiros RN, Ranjan R, Beloglazov A, De Rose CA, Buyya R (2011) Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exp 41(1):23–50
6. Cao J, Hwang K, Li K, Zomaya AY (2013) Optimal multiserver configuration for profit maximization in cloud computing. IEEE Trans Parallel Distrib Syst 24(6):1087–1096
7. Chakka R, Mitrani I (1996) Approximate solutions for open networks with breakdowns and repairs. Stochastic networks, theory and applications. R Stat Soc Lect Notes Ser 4:267–280
8. Chang X, Wang B, Muppala JK, Liu J (2016) Modeling active virtual machines on IaaS clouds using an M/G/m/m+ K queue. IEEE Trans Serv Comput 9(3):408–420
9. Chen S, Sun Y, Kozat U, Huang L, Sinha P, Liang, G, Liu X, Shroff N (2014) When queueing meets coding: Optimal-latency data retrieving scheme in storage clouds. In: 2014 Proceedings IEEE INFOCOM, pp 1042–1050
10. Chiang YJ, Ouyang YC, Hsu CH (2015) An efficient green control algorithm in cloud computing for cost optimization. IEEE Trans Cloud Comput 3(2):145–155
11. Dantas J, Matos R, Araujo J, Maciel P (2015) Eucalyptus-based private clouds: availability modeling and comparison to the cost of a public cloud. Computing 97:1121–1140
12. Deelman E, Singh G, Livny M, Berriman B, Good J (208) The cost of doing science on the cloud: the montage example. In: 2008 SC—International Conference for High Performance Computing, Networking, Storage and Analysis, pp 1–12
13. Duan Q, Yan Y, Vasilakos A (2012) A survey on service-oriented network virtualization toward convergence of networking and cloud computing. IEEE Trans Netw Serv Manag 9(4):373–392
14. Dustdar S, Guo Y, Satzger B, Truong HL (2011) Principles of elastic processes. IEEE Internet Comput 15(5):66–71
15. Entezari-Maleki R, Trivedi K, Movaghar A (2015) Performability evaluation of grid environments using stochastic reward nets. IEEE Trans Dependable Secur Comput 12(2):204–216
16. Ever E, Gemikonakli O, Kocyigit A, Gemikonakli E (2013) A hybrid approach to minimize state space explosion problem for the solution of two stage tandem queues. J Netw Comput Appl 36(2):908–926
17. Ghosh R, Longo F, Naik VK, Trivedi KS (2013) Modeling and performance analysis of large scale IaaS clouds. Future Gen Comput Syst 29(5):1216–1234 Special section: Hybrid Cloud Computing
18. Gunarathne T, Wu TL, Choi JY, Bae SH, Qiu J (2011) Cloud computing paradigms for pleasingly parallel biomedical applications. Concurr Comput Pract Exp 23(17):2338–2354
19. Haverkort BR, Marie R, Rubino G, Trivedi KS (2001) Performability modelling: techniques and tools. John Wiley, UK
20. Jin Y, Wen Y, Zhang W (2014) Content routing and lookup schemes using global bloom filter for content-delivery-as-a-service. IEEE Syst J 8(1):268–278
21. Khazaei H, Misic J, Misic V (2012) Performance analysis of cloud computing centers using M/G/m/m+r queuing systems. IEEE Trans Parallel Distrib Syst 23(5):936–943
22. Khazaei H, Misic J, Misic V (2013) A fine-grained performance model of cloud computing centers. IEEE Trans Parallel Distrib Syst 24(11):2138–2147
23. Law AM (2007) Statistical analysis of simulation output data: the practical state of the art. In: IEEE Simulation Conference, pp 77–83
24. Maguluri ST, Srikant R (2014) Scheduling jobs with unknown duration in clouds. IEEE/ACM Trans Netw 22(6):1938–1951
25. Matos R, Araujo J, Oliveira D, Maciel P, Trivedi K (2015) Sensitivity analysis of a hierarchical model of mobile cloud computing. Simul Model Pract Theory 50:151–164 (Special Issue on Resource Management in Mobile Clouds).
26. Mei J, Li K, Ouyang A, Li K (2015) A profit maximization scheme with guaranteed quality of service in cloud computing. IEEE Trans Comput 64(11):3064–3078
27. Mitrani I, Chakka R (1995) Spectral expansion solution for a class of Markov models: application and comparison with the matrix-geometric method. Perform Eval 23(3):241–260
28. Muñoz-Escoí FD, Bernabéu-Aubán JM (2016) A survey on elasticity management in paas systems. Computing 98:1–40
29. Nurmi D, Wolski R, Grzegorczyk C, Obertelli G, Soman S, Youseff L, Zagorodnov D (2009) The Eucalyptus open-source cloud-computing system. In: 9th IEEE/ACM International Symposium on Cluster Computing and the Grid. CCGRID '09, pp 124–131
30. Ostermann S, Iosup A, Yigitbasi N, Prodan R, Fahringer T, Epema D (2010) A performance analysis of EC2 cloud computing services for scientific computing. In: Cloud computing. Springer, pp 115–131

31. Qian H, Medhi D, Trivedi K (2011) A hierarchical model to evaluate quality of experience of online services hosted by cloud computing. In: 2011 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp 105–112
32. Raei H, Yazdani N (2015) Analytical performance model for mobile network operator cloud. J Supercomput 71(12):4555–4577
33. Schadt EE, Linderman MD, Sorenson J, Lee L, Nolan GP (2011) Cloud and heterogeneous computing solutions exist today for the emerging big data problems in biology. Nat Rev Genet 12(3):224–224
34. Schroeder B, Gibson G et al (2010) A large-scale study of failures in high-performance computing systems. IEEE Trans Dependable Secur Comput 7(4):337–350
35. Shi Z, Beard C, Mitchell K (2013) Analytical models for understanding space, backoff, and flow correlation in CSMA wireless networks. Wirel Netw 19(3):393–409
36. Smith R, Trivedi K, Ramesh A (1988) Performability analysis: measures, an algorithm, and a case study. IEEE Trans Comput 37(4):406–417
37. Tian W, Zhao Y, Xu M, Zhong Y, Sun X (2015) A toolkit for modeling and simulation of real-time virtual machine allocation in a cloud data center. IEEE Trans Autom Sci Eng 12(1):153–161
38. Trivedi KS (2008) Probability and statistics with reliability, queuing and computer science applications. 2nd ed. Wiley, New jersey
39. Tschaikowski M, Tribastone M (2014) Tackling continuous state-space explosion in a Markovian process algebra. Theor Comput Sci 517:1–33
40. Vilaplana J, Solsona F, Teixidó I, Mateo J, Abella F, Rius J (2014) A queuing theory model for cloud computing. J Supercomput 69(1):492–507
41. Voorsluys W, Broberg J, Venugopal S, Buyya R (2009) Cost of virtual machine live migration in clouds: a performance evaluation. In: Cloud Computing. Springer, pp 254–265
42. Wang D, Joshi G, Wornell G (2014) Efficient task replication for fast response times in parallel computation. In: ACM SIGMETRICS Performance Evaluation Review, vol. 42, ACM. pp 599–600
43. Yang B, Tan F, Dai YS (2013) Performance evaluation of cloud service considering fault recovery. J Supercomput 65(1):426–444
44. Yigitbasi N, Iosup A, Epema D, Ostermann S (2009) C-meter: a framework for performance analysis of computing clouds. In: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid. IEEE Computer Society, pp 472–477