# Performance Evaluation of Cloud Data Centers with Batch Task Arrivals

**Hamzeh Khazaei[1], Jelena Mišić[1] and Vojislav B. Mišić[1]**
[1]*Ryerson University, Toronto, ON, Canada*

## ABSTRACT

Accurate performance evaluation of cloud computing resources is a necessary prerequisite for ensuring that quality of service (QoS) parameters remain within agreed limits. In this chapter, we consider cloud centers with Poisson arrivals of batch task requests under total rejection policy; task service times are assumed to follow a general distribution. We describe a new approximate analytical model for performance evaluation of such systems and show that important performance indicators such as mean request response time, waiting time in the queue, queue length, blocking probability, probability of immediate service, and probability distribution of the number of tasks in the system can be obtained in a wide range of input parameters.

## INTRODUCTION

Cloud computing is a novel computing paradigm in which different computing resources such as infrastructure, platforms and software applications are made accessible over the internet to remote users as services [40]. It is quickly gaining acceptance: According to IDC, 17 billion dollars was spent on cloud-related technologies, hardware and software in 2009, and spending is expected to grow to 45 billion by 2013 [31]. Due to the dynamic nature of cloud environments, diversity of users' requests, and time dependency of load, providing agreed quality of service (QoS) while avoiding over-provisioning is a difficult task [42]. Performance evaluation of cloud centers is therefore an important research task. However, despite considerable research effort that has been devoted to cloud computing in both academia and industry, only a small portion of it have dealt with performance evaluation. In this chapter, we address this deficiency by proposing an analytical model for performance evaluation of cloud centers. The model utilizes queuing theory and probabilistic analysis to allow tractable evaluation of several important performance indicators, including response time and other related measures [41].

We assume that the cloud center consists of a number of servers that are allocated to users in the order of request arrivals. Users may request a number of servers in a single request, i.e., we allow batch arrivals, hereafter referred as super-tasks arrivals. This model is consistent with the so-called *On-Demand* services provided by Amazon Elastic Compute Cloud (EC2) [1]. Such services provide no advance reservation and no long-term commitment, which is why clients may experience delays in fulfillment of requests. (The other types of services offered by Amazon EC2, known as *Reserved* and *Spot* services, have different allocation policies and availability). While many of the large cloud centers employ virtualization to provide the required resources such as servers [10], we consider servers to be physical servers; our model is thus applicable to intra-company (private) clouds as well as to public clouds of small or medium providers.

As the user population size is relatively high and the probability of a given user requesting service is relatively low the arrival process can be adequately modeled as a Markovian process, i.e., super-tasks arrive according to a Poisson process [11]. However, some authors claimed that Poisson process is not adequately modeled the arrival process in real cloud centers [46].

When a super-task arrives, if the necessary number of servers is available, they are allocated immediately; if not, the super-task is queued in the input buffer until the servers become available, or rejected if the input buffer is unable to hold the request. As a result, all tasks within a super-task obtain service, or are rejected, simultaneously. This policy, known as *total rejection policy*, is well suited to modeling the behavior of a cloud center; it is assumed that the users request as many servers as they need, and would not accept a partial fulfillment of their requests.

A request may target a specific infrastructure instance (e.g., a dual- or quad-core CPU with specified amount of RAM), a platform (e.g., Windows, Linux, or Solaris), or a software application (e.g., a database management system, a Web server, or an application server), with different probabilities. Assuming that the service time for each component of the resulting infrastructure-platform-application tuple follows a simple exponential or Erlang distribution, the aggregate service time of the cloud center would follow a hyper-exponential or hyper-Erlang distribution. In this case, the coefficient of variation (CoV, defined as the ratio of standard deviation and mean value) of the resulting service time distribution exceeds the value of one [6]. As a result, the service time should be modeled with a general distribution, preferably one that allows the coefficient of variation to be adjusted independently of the mean value.

Therefore, we model the cloud center as an $M^{[x]}/G/m/m+r$ queuing system which indicates that tasks arrive in batches or groups with exponentially distributed inter-arrival time, that the service time of tasks in a super-task is generally distributed, and that the number of servers is $m$ while the length of the input buffer is $r$ (so that the capacity of system is $m+r$ ). The probability distribution of the number of tasks within a super-task is also generally distributed.

Such queuing system can be analyzed using stochastic processes. First we define a continuous-time process, original process that records the number of tasks in the system during the time. Since the original process is not Markovian, we employ the embedded Markovian processes to analyze the system and obtain desired performance metrics approximately. This chapter has four main contributions:

- We develop approximate but accurate and tractable model of cloud. Our model considers cloud centers with batch task arrivals, general task service time distribution, and general batch size distribution.

- Our model provides full probability distribution of the task response time and the number of tasks in the system (in service as well as in the input buffer). It also provides mean response time for a request, mean waiting time in the input buffer, probability that a super-task is blocked, and probability that a request will obtain immediate service.

- Performance of the cloud center was found to be very dependent on the coefficient of variation, *CoV*, of the task service time as well as the size of batches. Larger batches or/and higher value $(>1)$ of coefficient of variation of service time resulting in longer response time but also in lower utilization for cloud providers.

- Performance might be improved by both partitioning the requests according to the size of batches or based on the coefficient of variation of service times and then processing them through separate sub-centers.

The chapter is organized as follows: in related work section, we survey related work in cloud performance analysis as well as in queuing system analysis. Analytical Model section presents our model and the details of the analysis. Numerical Validation section presents the numerical results obtained from the analytical model, as well as those obtained through simulation. In Performance Improvement Techniques section, we propose two techniques in order to improve the performance of a cloud center. Finally, Conclusion section summarizes our findings and concludes the chapter.

## RELATED WORK

Cloud computing has attracted considerable research attention, but only a small portion of the work done so far has addressed performance issues. In [42], a cloud center is modeled as the classic open network with single arrival, from which the distribution of response time is obtained, assuming that both inter-arrival and service times are exponential. Using the distribution of response time, the relationship among the maximal number of tasks, the minimal service resources and the highest level of services was found.

In [43], the cloud center was modeled as a $M/M/m/m+r$ queuing system from which the distribution of response time was determined. Inter-arrival and service times were both assumed to be exponentially distributed, and the system has 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, according to authors' own argument).

In [4,33] the authors considered the problem of scheduling different classes of tasks on multiple distributed servers to minimize an objective function based on per-class mean response or waiting time. Their allocation of task types to servers, however, is not truly applicable to cloud computing domain since both of the papers corresponded every single server to a separate queue (i.e., combinations of $M/G/1$ queues). Total rejection policy, for example, may not be attainable in such a configuration.

Most theoretical analyses have relied on extensive research in performance evaluation of $M/G/m$ queuing systems [27,29,44,37]. However, the probability distributions of response time and queue length in $M/G/m$ and $M/G/m/m+r$ cannot be obtained exactly, which has motivated the search for an approximate solution.

An approximate solution for steady-state queue length distribution in a $M/G/m$ system with finite waiting space was described in [22]. As the approximation was given in an explicit form, its numerical computation is easier than when using earlier approximations [12,39]. The proposed approach is exact for $M/G/m/m$ and reasonably accurate in the more general case of $M/G/m/m+r$ when $r \neq 0$, but only when the number of servers $m$ is small.

A similar approach in the context of $M/G/m$ queues, but extended so as to approximate the blocking probability and, thus, to determine the smallest buffer capacity such that the rate of lost tasks remains under predefined level, was described in [23]. An interesting finding is that the optimal buffer size depends on the order of convexity for the service time; the higher this order is, the larger the buffer size should be.

An approximation for the average queuing delay in a $M/G/m/m+r$ queue, based on the relationship of joint distribution of remaining service time to the equilibrium service distribution, was proposed in [30]. Another approximation for the blocking probability, based on the exact solution for finite capacity $M/G/m/m+r$ queues, was proposed in [34]. Again, the estimate of the blocking probability is used to guide the allocation of buffers so that the loss/delay blocking probability remains below a specific threshold.

As the above results rely on some approximation(s) to obtain a closed-form solution, their validity is severely limited: in most cases, they accuracy is acceptable only when the number of servers is comparatively small, typically below 10 or so, which makes them unsuitable for performance analysis of cloud computing data centers. Moreover, the approximations are very sensitive to the probability distribution of task service times, and thus become increasingly inaccurate when the coefficient of variation of the service time, *CoV*, increases toward and above the value of one. Finally, approximation errors are particularly pronounced when the traffic intensity $\lambda$ is small and/or when both the number of servers $m$ and the *CoV* of the service time are large [5,24,38].

In [3,2], closed-form solutions for response time and mean queue size in a $G/G/m$ queuing system were found, but under the assumption that both the input arrival process and task service time can be described with a Coxian distribution. Moreover, its computational complexity is $O(m^3)$ and it considers the system with an infinite buffer, both of which render it unsuitable for performance modeling of cloud centers.

Our earlier work [19,14,15,16] presents an analysis of a cloud center under the assumptions of single task arrivals or unlimited buffer space which are much more restrictive than our current work. We also proposed a preliminary version of this work in [17]. Our recent works, [21,18, 20], are more concerned with virtualization, pool management and availability. They are restricted to exponentially distributed service time as opposed to this work that considers generally distributed service time.

Batch arrivals present an additional difficulty. An upper bound for the mean queue length and lower bounds for the delay probabilities (that of an arrival batch and that of an arbitrary task in the arrival batch) was described in [45]. An approximate formula is also developed for the general batch-arrival queue $GI^{[x]}/G/m$. In spite of the simplicity and acceptable performance, the approach is accurate enough for small batch sizes, up to 3, as well as small number of servers (less than 10).

In [9], the authors proposed an approximation method for the computation of the steady-state distribution of the number of tasks in queue as well as the moments of the waiting time distribution. They examined both hypo-exponential and hyper-exponential distribution family for service time, which is necessary for modeling a dynamic system such as cloud farms; however they just performed numerical results for a system with up to seven server and there is no result or indication about the efficiency of the method in case of larger number of servers or a system with finite capacity.

Another approximate formula was proposed in [7]. The authors presented an approximate formula for the steady-state average number of tasks in the $M^{[x]}/G/m$ queuing system. The derivation of the formula is based on a heuristic argument whereby a reformulation of the number of tasks in $M^{[x]}/G/1$ is extended to the multi-server queue. From a computational viewpoint, the approach is simple to apply, though, the relative percentage error incurred seem to be unavoidable when the number of servers is large, the mean batch size is small or the coefficient of variation of service time is larger than one.

A diffusion approximation for a $M^{[x]}/G/m$ queue was developed in [25]. The authors derived an approximate formula for the steady-state distribution of the number of tasks in the system, delay probability and mean queue length. However, the diffusion approach gives unacceptably large errors when batch size is larger than 3.

Overall, existing methods are not well suited to the analysis of cloud center where the number of servers may potentially be huge, the distribution of service times is unknown, and batch arrival of requests is allowed.

## ANALYTICAL MODEL

A system with above mentioned description can be modeled as an $M^{[x]}/G/m/m+r$ queuing system. We adopt a technique similar to *embedded Markov chain* in [35,26] for analyzing the system. We look at the system at moments of super-task arrivals and find the steady-state distribution of number of tasks in system at such instants. Due to the absence of Poisson Arrivals See Time Averages (PASTA) property, we then find the arbitrary-time steady-state distribution of number of tasks in the system using the embedded Markov process. Table 1 shows the symbols and their corresponding descriptions.

**Table 1: Symbols and Corresponding Description.**

| Symbol | Description |
|---|---|
| $g_k$ | probability that the super-task size is equal to *k*. |
| $\overline{g}$ | Mean value of super-task size. |
| $\Pi_g(z)$ | Probability generation function of the super-task size. |
| $\lambda$ | Arrival rate |
| $\mu$ | Service rate |
| $m$ | Number of servers |
| $A(x)$ and $B(x)$ | Probability density function of arrival and service time respectively. |
| $A^*(s)$ and $B^*(s)$ | Laplace Stieltjes Transform (LST) of arrival and service time. |
| $R_k(x)$ | Probability density function of residence time at states |
| $P_z$ | Probability of having no departure. |
| $P_{LL}(i,j,k)$ | Probability of moving from light traffic to light traffic. |
| $P_{LH}(i,j,k)$ | Probability of moving from light traffic to heavy traffic. |
| $P_{HH}(i,j,k)$ | Probability of moving from heavy traffic to heavy traffic. |
| $P_{HL}(i,j,k)$ | Probability of moving from heavy traffic to light traffic. |
| $\sigma_T$ | Standard deviation of response time. |
| $sk$ | Skewness of response time. |
| $ku$ | Kurtosis of response time. |

Let $g_k$ be the probability that the super-task size is equal to $k, k = 1,2,...,MBS$, in which $MBS$ is the maximum batch size; $g_k$ and $MBS$ depend on users' applications. Let $\overline{g}$ and $\Pi_g(z)$ be the mean value and probability generating function (PGF) of the task burst size respectively.

$$\begin{aligned}
&\Pi_g(z) = \sum_{k=1}^{MBS} g_k z^k \\
&g_k = Prob[X_g = k] \quad k = 1,2,...,MBS \\
&\overline{g} = \Pi_g^{(1)}(1)
\end{aligned} \tag{1.1}$$

Super-task request arrivals follow a Poisson process so super-task request inter-arrival time $A$ is exponentially distributed with rate of $\frac{1}{\lambda}$. We denote its, cumulative distribution function (CDF) as

$A(x) = Prob[A < x]$ and its probability density function (pdf) as $a(x) = \lambda e^{-\lambda x}$. The Laplace Stieltjes Transform (LST) of inter-arrival time is

$$A^*(s) = \int_0^\infty e^{-sx} a(x)dx = \frac{\lambda}{\lambda + s}$$

Tasks within a super-task have service times which are identically and independently distributed according to a general distribution *B*, with a mean service time of $\overline{b} = \frac{1}{\mu}$. The CDF of the service time is $B(x) = Prob\left[B < x\right]$, 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 the time interval from an arbitrary point (an arrival point in a Poisson process) during a service time to the end of the service time; we denote it as $B_+$ . *Elapsed task service time* is the time interval from the beginning of a service time to an arbitrary point of the service time; we denote it as $B_-$. It can be shown that both residual and elapsed task service times have the same probability distribution, the LST of which can be calculated as [35]

$$B_+^*(s) = B_-^*(s) = \frac{1 - B^*(s)}{s\overline{b}} \tag{1.2}$$

The *traffic intensity* may be defined as

$$\rho \triangleq \frac{\lambda \overline{g}}{m\mu}$$

For practical reasons and ergodicity condition, we assume that $\rho < 1$. If at the moment of super-task arrival the input queue doesn't have enough space for whole super-task, the super-task would be lost. We also consider total rejection policy for servicing [36] in which the service time of whole tasks in a super-task start at the same time on different servers; in other words, if number of idle servers is less than super-task size then those idle servers remain unused till other servers become free and then the super-task can be fit in servers. Here the waiting time for the first, last and any arbitrary tasks in a super-task is identical because all of them get into service at the same time.

## THE EMBEDED PROCESSES

The number of tasks in a cloud center during the operation hours can be considered as a collection of random variables that are indexed by time, collectively, referred to as a stochastic process. Such original process is non-Markovian since the service time of tasks is not exponentially distributed. However, arrival points have the property that all past states information is irrelevant in determining the future of the process at such epochs. Therefore we may define an embedded semi-Markov process at super-task arrival instances for characterization of the system. We also introduce another process, an embedded Markov process that imitates the original process in a discrete manner, namely, at the instants of super-task arrivals. More specifically, the value of embedded Markov process will be changed only at super-task arrivals. Fig. 1 shows a sample path of original, semi-Markov and Markov processes schematically. With regards to embedded Markov process, we may also recognize an embedded Markov chain at super-task arrivals. Note that in the embedded Markov process the next event after last arrival is the next super-task arrival as opposed to embedded semi-Markov process in which the next event is the next task departure or super-task arrival. The associated embedded Markov chain is shown in Fig. 2, 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 drawn.

Figure 1: A sample path of the original process, embedded semi-Markov process and embedded Markov process

We employ the embedded Markov chain to identify the steady-state distribution of number of tasks in the system at super-task arrivals. However the transition probabilities of the embedded Markov chain cannot be exactly determined due to intractable behavior of the embedded semi-Markov process (*ESMP*). More precisely, in order to determine the associated transition probabilities we need to count the number of task departures between two super-task arrivals; however such counting process is intractable so that we need to resort to approximation. As a result, the embedded Markov chain models the system approximately at super-tasks arrival instances. The approximate embedded Markov chain (*aEMC*) is more elaborated in Transition Matrix section where we calculate the transition probabilities.

Figure 2: Embedded Markov chain associate with the embedded Markov process.

Let $A_n$ and $A_{n+1}$ indicate the moment of $n^{th}$ and $(n+1)^{th}$ super-task 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. If $k$ is the size of super-task and $v_{n+1}$ indicates the number of tasks which depart from the system between $A_n$ and $A_{n+1}$, the following holds:

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

We need to calculate the transition probabilities associated with aEMC, defined as

$$P(i,j,k) \triangleq Prob\left[q_{n+1} = j \mid q_n = i \text{ and } X_g = k\right] \tag{1.4}$$

i.e., the probability that $i+k-j$ customers are served during the interval between two successive super-task arrivals. Such counting process requires the exact system behavior between two super-task arrivals. Obviously for $j > i+k$,

$$P(i,j,k) = 0 \tag{1.5}$$

Since there are at most $i+k$ tasks present between the arrival of $A_n$ and $A_{n+1}$. For calculating the other transition probabilities associated with aEMC, we need to identify the distribution function of residence time for each state in ESMP. We now describe the residence times for states in the ESMP.

- **Case 1:** The state residence time for the first departure is remaining service time, $B_+(x)$, since the last arrival is a random point in the current task's service time.
- **Case 2:** If the second departure is from the same server then clearly the state residence time is the service time ($B(x)$).
- **Case 3:** No departure between two super-task arrivals as well as the last departure before the next arrival makes the state residence time exponentially distributed with the mean value of $\frac{1}{\lambda}$.
- **Case 4:** If $i^{th}$ departure is from another server then the CDF of state residence time is $B_{i+}(x)$. In Fig. 4, for instance, departure $D_{21}$ takes place after departure $D_{11}$. Therefore $D_{11}$ could be considered as an arbitrary point in the remaining service time of the task in server #2; so the CDF of residence time for second departure is $B_{2+}(x)$. As a result the LST of $B_{2+}(x)$ is the same

as $B_{+}^{*}(s)$, Eq. (1.6), though, here we have one more step in recursion. Generally, the LST of residence times between subsequent departures from non-identical servers may be recursively defined as follow

$$B_{i+}^{*}(s)=\frac{1-B_{(i-1)+}^{*}(s)}{s\cdot\overline{b}_{(i-1)+}},\quad i=1,2,3,\cdots \tag{1.6}$$

where

$$\overline{b}_{(i-1)+}=[-\frac{d}{ds}B_{(i-1)+}^{*}(s)]_{s=0}$$

To maintain the consistency in notation we may define the followings:

$$\overline{b}_{0+}=\overline{b}$$
$$B_{0+}^{*}(s)=B^{*}(s)$$
$$B_{1+}^{*}(s)=B_{+}^{*}(s)$$

Let $R_k(x)$ denotes the CDF of residence times at state $k$ in ESMP:

$$R_k(x)=\begin{cases} B_{+}(x), & \text{Case 1} \\ B(x), & \text{Case 2} \\ A(x), & \text{Case 3} \\ B_{i+}(x), & \text{Case 4} \quad i=2,3,\cdots \end{cases} \tag{1.7}$$

Figure 3: System behavior between two observation points and all possible state residence times.

## DEPARTURE PROBABILITIES

To find the elements of the transition probability matrix, we need to count the number of tasks departing from the system in the time interval between two successive super-task arrivals. Therefore at first step, we need to calculate the probability of having $k$ arrivals during the residence time of each state. Let $B_{+}^{n}$ , $B^{n}$ and $B_{i+}^{n}$ $(i=2,3,\cdots)$ indicate the number of arrivals during residence times: $B_{+}(x)$, $B(x)$ and $B_{i+}(x)$ respectively. Due to Poisson arrivals we may define the following probabilities:

$$\begin{aligned}
\alpha_k &\triangleq \text{Prob}[B_{+}^{n}=k]=\int_0^\infty \frac{(\lambda x)^k}{k!}e^{-\lambda x}dB_{+}(x) \\
\beta_k &\triangleq \text{Prob}[B^{n}=k]=\int_0^\infty \frac{(\lambda x)^k}{k!}e^{-\lambda x}dB(x) \\
\gamma_k &\triangleq \text{Prob}[A^{n}=k]=\int_0^\infty \frac{(\lambda x)^k}{k!}e^{-\lambda x}dA_{+}(x) \\
\delta_{ik} &\triangleq \text{Prob}[B_{i+}^{n}=k]=\int_0^\infty \frac{(\lambda x)^k}{k!}e^{-\lambda x}dB_{i+}(x)
\end{aligned} \tag{1.8}$$

In fact we are interested in the probability of having no arrival; because using those probabilities, we are able to calculate the transition probabilities in aEMC.

$$
\begin{aligned}
&P_x = \alpha_0 \triangleq \text{Prob}[B_+^n = 0] = \int_0^\infty e^{-\lambda x} dB_+(x) = B_+^*(\lambda) \\
&P_y = \beta_0 \triangleq \text{Prob}[B^n = 0] = \int_0^\infty e^{-\lambda x} dB(x) = B^*(\lambda) \\
&P_z = \gamma_0 \triangleq \text{Prob}[A^n = 0] = \int_0^\infty e^{-\lambda x} dA_+(x) = A_+^*(\lambda) = A^*(\lambda) \\
&P_{ix} = \delta_0 \triangleq \text{Prob}[B_{2+}^n = 0] = \int_0^\infty e^{-\lambda x} dB_{i+}(x) = B_{i+}^*(\lambda) \\
&P_{xy} = P_x P_y
\end{aligned}
\tag{1.9}
$$

Note that $P_{1x}$=$P_x$. We may also define the probability of having no departure between two super-task arrivals. Let $A$ be an exponential random variable with the parameter of $\lambda$ and $B_+$ be a random variable which is distributed according to $B_+(x)$ (remaining service time). The probability of having no departure is equal to $\text{Prob}[A < B_+]$ :

$$
\begin{aligned}
P_z &= Prob\left[A < B_+\right] = \int_{x=0}^{\infty} P\{A < B_+ \mid B_+ = x\} dB_+(x) \\
&= \int_{x=0}^{\infty} P\{A < x\} dB_+(x) = \int_{x=0}^{\infty} \left( \int_{y=0}^{x} \lambda e^{-\lambda y} dy \right) dB_+(x) \\
&= \int_{x=0}^{\infty} \left[1 - e^{-\lambda y}\right]_{y=0}^{x} dB_+(x) = \int_0^\infty (1 - e^{-\lambda x}) dB_+(x) \\
&= \int_0^\infty dB_+(x) - \int_0^\infty e^{-\lambda x} dB_+(x) \\
&= 1 - B_+^*(\lambda) = 1 - P_x
\end{aligned}
\tag{1.10}
$$

Using probabilities $P_x$, $P_y$ , $P_{xy}$, $P_{ix}$ and $P_z$ we may define the transition probabilities for aEMC.

## TRANSITION MATRIX

Based on servicing policy, we may identify four different regions of operation for which different conditions hold. The numbers on rows and columns correspond to the number of tasks in the system immediately before a super-task arrival (*i*) and immediately upon the next super-task arrival (*j*), respectively. We also have *k* in transition probability which indicates the size of super-task. Each region has a specific transition probability equation that depends on current state, *i*, next state, *j*, batch size, *k*, and number of departures between two super-task arrivals. Note that for all regions if $i + k = j$ the following is held:

$$
P(i, j, k) = P_z \tag{1.11}
$$

## REGION 1

In this region, the input queue is empty and remains empty until next arrival; the transitions originate and terminate on the states that are on the left hand side of state *m* (i.e., lower than *m*). Let us denote the number of tasks which depart from the system between two super-task arrivals as $w(k) = i + k - j$. For $i, j \leq m$, the transition probability is

$$P_{LL}(i,j,k) \square \begin{cases} \sum_{z=0}^{min(i,w(k))} \binom{i+k}{w(k)} P_x^{w(k)} (1-P_x)^j, & \text{if } i+k \leq m \\ \sum_{z=i+k-m}^{min(i,w(k))} \binom{i}{z} P_x^z (1-P_x)^{i-z} \cdot \binom{k}{w(k)-z} P_{xy}^{w(k)-z} (1-P_{xy})^{z-i+j}, & \text{if } i+k > m \end{cases} \quad (1.12)$$

## REGION 2

In this region, the queue is empty before the transition, but not empty afterwards, which means that transitions originate below state *m* and terminate above it: $i < m, j > m$. In this case the arriving super-task can't be accommodated in the idle servers so it will be queued. Transition probabilities are

$$P_{LH}(i,j,k) = \Pi_{s=1}^{w(k)} \left[ (i-s+1) P_{sx} \right] \cdot (1-P_x)^{i-w(k)} \quad (1.13)$$

## NUMBER OF IDLE SERVERS

To calculate the transition probabilities for regions 3 and 4, we may need to know the probability of having *i* idle servers out of *m*. Note that the total rejection policy, explained in Introduction section, means that a super-task may have to wait even if there are some free servers; this happens when the number of idle servers is smaller than the number of tasks in the super-task at the head of the queue. In other words, in order to count the number of departures, the *real service rate* of the system should be determined.

Suppose that we have a Poisson batch arrival process in which the batch size is a generally distributed random variable. Each arriving batch is stored in a finite queue. Storing the arrival batches in the queue will be continued until either the queue gets full or the last arrival batch cannot be fitted in the queue. If the queue size is *t*, the mean batch size is $\overline{g}$, and the maximum batch size is equal to *MBS*, what is the probability (denoted as $P_i(n)$) of having *n* unoccupied spaces in the queue after all? It can be seen that this problem can be reduced to the original (idle servers) problem easily.

It is clear that if the distribution of batch size is deterministic and equal to one, that is single arrival, then the queue will get full eventually. However, if the batch size is generally distributed, which is the case in our scenario, the probability $P_i(n)$ cannot be computed exactly, and an approximate solution is needed. We have built a small simulator using object-oriented Petri net-based simulation engine Artifex by RSoftDesign, Inc. [32], and simulated the queue size for different mean batch sizes; the queue size was fixed at $m = 200$, as the results indicate that it has virtually no impact on the probability $P_i(n)$. The experiment was performed one million times, and the resulting probability distribution is shown in Fig. 6.

Figure 4: Probability of having $n$ idle servers for different batch sizes ( $P_i(n)$ ).

The shape indicates exponential dependency so we have used the interpolation software Curve Expert [8] to find the parameter values that give the best fit. The parameter values that give the best fit for the exponential function $ae^{bx}$ are shown in Table 2, for different values of mean batch size; in all cases, the approximation error remains below 0.18%.

Table 2: Parameters for exponential curves: $ae^{bx}$

| batch size | 2 | 4 | 6 | 8 |
|---|---|---|---|---|
| A | 5.154051E-01 | 2.725053E-01 | 1.839051E-01 | 1.393975E-01 |
| B | -6.918772E-01 | -2.913967E-01 | -1.825455E-01 | -1.334660E-01 |

This allows us to define the transition probabilities for region 3 and 4 in transition probability matrix.

## REGION 3

Region 3 corresponds to the case where the queue is not empty throughout the inter-arrival time, i.e., $i, j > m$. In this case all transitions start and terminate at a state above $m$ in Fig. 2, and the state transition probabilities can be approximately computed as

$$P_{HH}(i,j,k) \square \sum_{\psi=(m-MBS+1)}^{m} \sum_{s_1=\min(w(k),1)}^{\min(w(k),\psi)} \binom{\psi}{s_1} P_x^{s_1}(1-P_x)^{\psi-s_1} \cdot P_i(m-\psi) \cdot$$

$$\sum_{\delta=max(0,m-\psi+s_1-MBS+1)}^{m-\psi+s_1} \sum_{s_2=\min(w(k)-s_1,1)}^{\min(\delta,w(k)-s_1)} \binom{\delta}{s_2} P_{2x}^{s_2}(1-P_{2x})^{\delta-s_2} \cdot P_i(m-\psi+s_1-\delta) \cdot \tag{1.14}$$

$$\sum_{\phi=max(0,m-\psi+s_1-\delta+s_2-MBS+1)}^{m-\psi+s_1-\delta+s_2} \binom{\phi}{w(k)-s_1-s_2} P_{3x}^{w(k)-s_1-s_2}(1-P_{3x})^{\phi-w(k)+s_1+s_2} \cdot$$

$$P_i(m-\psi+s_1-\delta+s_2-\phi)$$

Note that under moderate load it is not likely to have more than a couple of task departures from a single server.

## REGION 4

Finally, region 4, in which $i > m$ and $j \le m$, describes the situation where the first arrival $A_n$ finds non-empty queue which it joins while at the time of the next arrival ( $A_{n+1}$ ) there are $j$ tasks in the system, all of which are in service and the system has at least one idle server. The transition probabilities for this region are

$$P_{HL}(i,j,k) \square \sum_{\psi=(m-MBS+1)}^{m} \sum_{s_1=\min(0,\psi-j)}^{\min(w(k),\psi)} \binom{\psi}{s_1} P_x^{s_1}(1-P_x)^{\psi-s_1} \cdot P_i(m-\psi) \cdot$$

$$\sum_{\delta=max(0,m-\psi+s_1-MBS+1)}^{m-\psi+s_1} \sum_{s_2=\min(w(k)-s_1,\psi-j)}^{\min(\delta,w(k)-s_1)} \binom{\delta}{s_2} P_{2x}^{s_2}(1-P_{2x})^{\delta-s_2} \cdot P_i(m-\psi+s_1-\delta) \cdot \tag{1.15}$$

$$\sum_{\phi=max(0,m-\psi+s_1-\delta+s_2-MBS+1)}^{m-\psi+s_1-\delta+s_2} \binom{\phi}{w(k)-s_1-s_2} P_{3x}^{w(k)-s_1-s_2}(1-P_{3x})^{\phi-w(k)+s_1+s_2} \cdot$$

$$P_i(m-\psi+s_1-\delta+s_2-\phi)$$

## EQUILIBRIUM BALANCE EQUATIONS

After finding matrix **P** we can establish the balance equations. Such balance equations will have a unique steady state solution if the corresponding Markov chain is *ergodic*. The balance equations

$$\pi_i = \sum_{j=\max[0,i-MBS]}^{m+r} \pi_j p_{ji}\,, \quad 0 \le i \le m+r \tag{1.16}$$

That augmented by the normalization equation

$$\sum_{i=0}^{m+r} \pi_i = 1. \tag{1.17}$$

make the equations set. So far we have $m+r+2$ equations which includes $m+r+1$ linearly independent equations from (1.16) and one normalization equation from (1.17); however we have $m+r+1$ variables $[\pi_0, \pi_1, \pi_2, \ldots, \pi_{m+r}]$; so in order to obtain the unique equilibrium solution we need to remove one of the equations; the wise choice would be the last equation in (1.16) due to minimum information this equation holds about the system in comparison with the others. Here, the steady state balance equations can't be solved in closed form, hence we must resort to a numerical solution.

## DISTRIBUTION OF NUMBER OF TASKS IN THE SYSTEM

Once we obtain the steady state probabilities we are able to establish the PGF for the number of tasks in the system at the time of a super-task arrival:

$$\Pi(z) = \sum_{k=0}^{m+r} \pi_z z^k \tag{1.18}$$

Due to batch arrival, the PASTA property doesn't hold; thus, the PGF $\Pi(z)$ of the distribution of number of tasks in system at arrival times is not the same with PGF $P(z)$ for distribution of number of tasks in system at any arbitrary time.

## DISTRIBUTION OF NUMBER OF TASKS IN THE SYSTEM AT ANY ARBITRARY TIME

In order to obtain steady-state distribution at arbitrary time, we employ a technique similar to semi-Markov process in [35]. We use the approximate embedded Markov process (*aEMP*) at super-task arrival points. The aEMP imitates the original process but it will be updated just at the arrival instants. Let $H_k(x)$ be the CDF of the residence time that aEMP stays in the state *k*:

$$H_k(x) \square \, Prob[t_{n+1} - t_n \le x \mid q_n = k] = 1 - e^{\lambda x}, \quad k = 0,1,2,...,m+r \tag{1.19}$$

this in our system does not depend on n. The mean residence time in state *k* is

$$\overline{h}_k = \int_0^{\infty} [1 - H_k(x)] dx = \frac{1}{\lambda}, \qquad k = 0,1,2,...,m+r \tag{1.20}$$

and the steady-state distribution in aEMP is given by [35].

$$p_k^{sm} = \frac{\pi_k \overline{h}_k}{\sum_{j=0}^{m+r} \pi_j \overline{h}_j} = \frac{\pi_k}{\lambda \sum_{j=0}^{m+r} \pi_j 1/\lambda} = \pi_k \tag{1.21}$$

where $\{\pi_k; k = 0,1,...,m+r\}$ is the distribution probability at aEMC. So the steady-state probability of aEMP is identical with the embedded aEMC. We now define the CDF of the elapsed time from the most recent observing point looking form an arbitrary time by

$$H_k^-(y) = \frac{1}{\overline{h}_k} \int_0^{y} [1 - H_k(x)], \quad k = 0,1,2,...,m+r \tag{1.22}$$

the arbitrary-time distribution is given by

$$p_i = \sum_{j=i}^{m+r} p_j^{sm} \int_0^{\infty} \text{Prob[changes in } y \text{ that bring the state from } j \text{ to } i\,] \, dH_j^-(y) = \sum_{j=i}^{m+r} \pi_j P(j,i,0) \tag{1.23}$$

The PGF of the number of tasks in system is given by

$$P(z) = \sum_{i=0}^{m+r} p_i z^i \tag{1.24}$$

Mean number of tasks in the system, then, obtained as

$$\overline{p} = P'(1) \tag{1.25}$$

## BLOCKING PROBABILITY

Since arrivals are independent of buffer state and the distribution of number of tasks in the system was obtained, we are able to directly calculate the blocking probability of a super-task in the system with buffer size of *r*:

$$P_b(r) = \sum_{k=0}^{MBS-1} \left[ \sum_{i=0}^{MBS} p_{m+r-i-k} (1 - G(i)) \right] \cdot P_i(k) \tag{1.26}$$

The appropriate buffer size, $r_{\text{ò}}$, in order to have the blocking probability below the certain value, ò, is:

$$r_{\text{ò}} = \{ r \ge 0 \mid P_b(r) \le \; \& \; P_b(r-1) > \text{ò} \} \tag{1.27}$$

## PROBABILITY OF IMMEDIATE SERVICE

Here we are interested in the probability with that super-tasks will get into service immediately upon arrival, without any queuing. For such super-tasks, the response time would be equal to the service time:

$$P_{nq} = \sum_{k=0}^{MBS-1} \left[ \sum_{j=0}^{m-k-MBS} p_j + \sum_{i=m-k-MBS+1}^{m-k-1} p_i G(m-k-i) \right] \cdot P_i(k) \tag{1.28}$$

## DISTRIBUTION OF RESPONSE AND WAITING TIME

Let $W$ denote the waiting time in the steady state, and similarly let $W(x)$ and $W^*(s)$ be the CDF, of $W$ and it's LST, respectively. For the $M^{[x]}/G/m/m+r$ systems the queue length has the same distribution as $W$, the number of tasks which arrive during the waiting time:

$$Q(z) = W^*(\lambda_e(1-z)) \tag{1.29}$$

where $\lambda_e = \lambda(1-P_b)$.
The left hand side of (1.30) in our system can be calculated as:

$$Q(z) = \sum_{\psi=(m-MBS+1)}^{m} \left[ \sum_{k=0}^{\psi-1} p_k + \sum_{k=\psi}^{m+r} p_k z^{\psi-k} \right] \cdot P_i(m-\psi) \tag{1.30}$$

Hence, we have

$$W^*(s) = Q(z)\big|_{z=1-(s/\lambda_e)} = Q(1-s/\lambda_e) \tag{1.31}$$

Moreover, the LST of response time is

$$T^*(s) = W^*(s)\, B^*(s) \tag{1.32}$$

where $B^*(s)$ is the LST of service time. The *i* th moment, $t^{(i)}$, of the response time distribution is given by

$$t^{(i)} = \int_0^\infty x^i dT(x) = i\int_0^\infty x^{i-1}[1-T(x)]dx = (-1)^i T^{*(i)}(0) \qquad i = 2,3,4,\ldots \tag{1.33}$$

Using the moments we can calculate *standard deviation* as [13]:

$$\sigma_T = \sqrt{t^{(2)} - \overline{t}^2} \tag{1.34}$$

*Skewness* as:

$$sk = \frac{t^{(3)} - 3\overline{t}\,\sigma_T^2 - \overline{t}^3}{\sigma_T^3} \tag{1.35}$$

And *kurtosis* as:

$$ku = \frac{t^{(4)} - 4t^{(3)}\overline{t} - 6t^{(2)}\overline{t}^2 - 3\overline{t}^4}{\sigma_T^4} - 3 \tag{1.36}$$

## NUMERICA VALIDATION

The resulting balance equations of analytical model have been solved using Maple 15 from Maplesoft, Inc. [28]. To validate the analytical solution, we have built a discrete event simulator of the cloud server farm using the Artifex engine [32].

We have configured a cloud center with 200 servers and the capacity of 300 tasks (200 servers plus 100 of input queue space). Traffic intensity was set to $\rho = 0.85$ , which may seem too high but could easily be obtained in private cloud centers or public centers of small/medium size providers. Task service time is assumed to have Gamma distribution, while the distribution of batch size is assumed to be geometric. We set the maximum batch size as $MBS = 2\overline{g} + 1$ (truncated geometrically distributed batch sizes). We chose $2\overline{g} + 1$ for maximum batch size because in any cloud center usually there is an upper limit for number of requesting servers by a customer; if a customer needs more servers he has to submit another request. Gamma distribution is chosen because it allows the coefficient of variation to be set independently of the mean value. Two values are used for CoV; the low value, $CoV = 0.5$, which results in hypo-exponentially distributed service time and the higher value $CoV = 1.4$, which results in hyper-exponentially distributed service time. In all plots, the simulation and analytical results are labeled with *Sim* and *AM*, respectively.

The mean number of tasks in the system and queue are shown in Fig. 7. As can be seen, the queue size increases as the size of batches get increased whereas the mean number of tasks in the system gets decreased at the same time; such a behavior may be attributed to the total rejection policy, which lets some servers to remain idle even though there are some super-tasks in the queue.

Figure 5a: Mean number of tasks in the system.
Figure 5b: Mean queue length.

Figure 5: Number of tasks in the system and queue for a cloud center with 200 servers and capacity of 300.

We also compute the blocking probability, and the results are shown in Fig. 8a. The blocking probability increases as the size of batches get increased. Since the percentage of tasks which can get immediately into service is an important non-functional service property in the service level agreement (SLA), we also calculate the probability of immediate service (which might be termed availability); the result, see Fig. 8b, indicates that the larger the batch size results in the lower the probability of immediate service.

Figure 6a: Blocking probability.
Figure 6b: Probability of getting immediate service.

Figure 6: Probability of queuing and immediate service with regard to batch sizes.

We compute the system response time and queue waiting time for super-tasks. Note that, because of the total rejection policy, the response and waiting times for super-tasks are identical to those individual tasks within the super-task. As depicted in Fig. 9, response time, which is the sum of waiting time and service time, and waiting time are increased linearly at the same rate while the size of batches is getting larger; the reason is that because large super-tasks are more likely to remain longer in the queue in order to be fitted on the idle servers due to the total rejection policy.

Figure 7a: Response time.

Figure 7b: Waiting time in queue.

Figure 7: Response time and waiting time in queue.

Overall, the results suggest that performance is worse when the service time is hyper-exponentially distributed (i.e., $CoV = 1.4$ ). To obtain further insight into cloud center performance, we also calculate the higher moments of response time, as shown in Fig. 10. Standard deviation, shown in Fig 10a, increases as the size of batches gets increased. Moreover, response time is more dispersed when the service time of tasks is hyper-exponentially distributed.

Figure 8a: Standard Deviation
Figure 8b: Skewness.
Figure 8c: kurtosis.
Figure 8: Higher moment related performance metrics.

Skewness is a measure of symmetry of a distribution around the mean; a value of zero indicates a fully symmetric distribution, while positive/negative values indicate that the tails are longer on the left/right hand side of the mean, respectively. Skewness, shown in Fig. 10b, is rather high, and increases as batch size and/or CoV of service time set to large values. This indicates that, the higher the $CoV$ of service time distribution, the longer the tail of the response time distribution will be. Kurtosis indicates whether the distribution is more `peaked' or `flat' with respect to the normal distribution with the same mean and standard deviation. As can be seen from Fig. 10c, kurtosis is high which indicates the response time distribution is relatively peaked around the mean. However, the values obtained for $CoV = 1.4$ is much higher than the corresponding values for $CoV = 0.5$ .

The last two diagrams imply that, in practice, the response time will increase rapidly, have a very pronounced mean value, and then decrease slowly with a rather long tail. A long tail means that some super-tasks may experience much longer response time than the others, esp. when the input traffic consists of super-tasks with widely varying service times and/or large number of tasks in a super-tasks. Consequently, a non-negligible portion of super-tasks will experience extremely long delays, or even blocking, which may be unacceptable for cloud operators, in private as well as in public clouds.

## FUTURE RESEARCH DIRECTIONS

In future, we plan to extend our model for other submission/servicing policies such as partial acceptance. We also plan to investigate the possibility of dynamic allocation of servers to sub-centers, as well as the charging policies that can bring additional incentives for customers whilst maximizing the profit for cloud providers.

## CONCLUSION

Performance evaluation of cloud centers is a crucial task for cloud center providers, as it allows them to tailor their SLAs to the terms they are able to efficiently provide to their customers. In this chapter, we have described the first analytical model for performance evaluation of a cloud computing center under batch arrivals and total rejection policy. To accurately model the cloud environment, we have assumed generally distributed service time for each task within super-tasks, general distribution for the batch size, and a large number of servers. We have solved the approximate model and validated it through simulations. Our results show that the proposed method provided a quite accurate computation of important performance indicators such as the mean number of tasks in the system, queue length, mean response and waiting time, blocking probability and the probability of immediate service under batch arrival and total rejection policy for both admission and servicing. The distribution of response time is also characterized and thus proper configuration for a cloud center was proposed.

Our findings also show that cloud centers which allow requests with widely varying service times may impose longer waiting time on its clients, and lower chance of getting immediate service, while having its servers less utilized, compared to equivalent centers which deal with certain types of tasks. Moreover the bigger batches lead to longer waiting times and less utilization, thereby making operation more costly for the cloud provider.

## REFERENCES

[1] Amazon Elastic Compute Cloud (2010). User Guide. Amazon Web Service LLC or its affiliate, API Version edition.

[2] Bertsimas, D. (1988). An exact FCFS waiting time analysis for a general class of G/G/s queueing systems. Queuing Systems, 3:305–320.

[3] Bertsimas, D. (1990). An analytic approach to a general class of G/G/s queueing systems. Operations Research, 38(1):139–155.

[4] Borst, S. C. (1995). Optimal probabilistic allocation of customer types to servers. SIGMETRICS Perform. Eval. Rev., 23:116–125.

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

[6] Corral-Ruiz, A., Cruz-Perez, F., and Hernandez-Valdez, G. (2010). Teletraffic model for the performance evaluation of cellular networks with hyper-erlang distributed cell dwell time. In Vehicular Technology Conference (VTC 2010-Spring), IEEE 71st, pages 1–6.

[7] Cosmetatos, G. P. (1978). Some practical considerations on multi-server queues with multiple poisson arrivals. Omega, 6(5):443–448.

[8] CurveExpert (2011). Curveexpert professional 1.1.0. Website. http://www.curveexpert.net.

[9] Federgruen, A. and Green, L. (1984). An M/G/c queue in which the number of servers required is random. Journal of Applied Probability, 21(3):583–601.

[10] Fu, J., Hao, W., Tu, M., Ma, B., Baldwin, J., and Bastani, F. (2010). Virtual services in cloud computing. In IEEE 2010 6th World Congress on Services, pages 467–472, Miami, FL.

[11] Grimmett, G. and Stirzaker, D. (2010). Probability and Random Processes. Oxford University Press, 3rd edition.

[12] Hokstad, P. (1978). Approximations for the M/G/m queues. Operations Research, 26:510–523.

[13] Joanes, D. N. and Gill, C. A. (1998). Comparing measures of sample skewness and kurtosis. Journal of the Royal Statistical Society: Series D (The Statistician), 47(1):183–189.

[14] Khazaei, H., Mišić, J., and Mišić, V. B. (2010). Performance analysis of cloud computing centers. In 7th International ICST Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, QShine, Houston, TX.

[15] Khazaei, H., Mišić, J., and Mišić, V. B. (2011a). Modeling of cloud computing centers using M/G/m queues. In The First International Workshop on Data Center Performance, Minneapolis, MN.

[16] Khazaei, H., Mišić, J., and Mišić, V. B. (2011b). On the performance and dimensioning of cloud computing centers. In Wang, L., Ranja, R., Chen, J., and Benatallah, B., editors, Cloud computing: methodology, system, and applications, chapter 8, pages 151–165. FL: CRC Press, Boca Raton.

[17] Khazaei, H., Mišić, J., and Mišić, V. B. (2011c). Performance analysis of cloud centers under burst arrivals and total rejection policy. In Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE, pages 1–6.

[18] Khazaei, H., Mišić, J., and Mišić, V. B. (2012a). A fine-grained performance model of cloud computing centers. IEEE Transactions on Parallel and Distributed Systems, 99(PrePrints):1.

[19] Khazaei, H., Mišić, J., and Mišić, V. B. (2012b). Performance analysis of cloud computing centers using M/G/m/m + r queuing systems. IEEE Transactions on Parallel and Distributed Systems, 23(5):1.

[20] Khazaei, H., Mišić, J., Mišić, V. B., and Beigi Mohammadi, N. (2012c). Availability analysis of cloud computing centers. In Globecom 2012-Communications Software, Services and Multimedia Symposium (GC12 CSSM), Anaheim, CA, USA.

[21] Khazaei, H., Mišić, J., Mišić, V. B., and Rashwand, S. (2012d). Analysis of a pool management scheme for cloud computing centers. IEEE Transactions on Parallel and Distributed Systems, 99(PrePrints).

[22] Kimura, T. (1983). Diffusion approximation for an M/G/m queue. Operations Research, 31:304–321.

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

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

[25] Kimura, T. and Ohsone, T. (1984). A diffusion approximation for an M/G/m queue with group arrivals. Management Science, 30(3):381–388.

[26] Kleinrock, L. (1975). Queueing Systems, volume 1, Theory. Wiley-Interscience.

[27] Ma, B. N. W. and Mark, J. W. (1998). Approximation of the mean queue length of an M/G/c queuing system. Operations Research, 43:158–165.

[28] Maplesoft, Inc. (2012). Maple 15. Waterloo, ON, Canada.

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

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

[31] Patrizio, A. (2011). IDC sees cloud market maturing quickly. Datamation.

[32] RSoft Design (2003). Artifex v.4.4.2. RSoft Design Group, Inc., San Jose, CA.

[33] Sethuraman, J. and Squillante, M. S. (1999). Optimal stochastic scheduling in multiclass parallel queues. SIGMETRICS Perform. Eval. Rev., 27:93–102.

[34] Smith, J. M. (2003). M/G/c/K blocking probability models and system performance. Perform. Eval., 52:237–267.

[35] Takagi, H. (1991). Queuing Analysis, volume 1: Vacation and Priority Systems. North-Holland, Amsterdam, the Netherlands.

[36] Takagi, H. (1993). Queuing Analysis, volume 2: Finite Systems. North-Holland, Amsterdam, The Netherlands.

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

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

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

[40] Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. (2008). A break in the clouds: towards a cloud definition. SIGCOMM Comput. Commun. Rev., 39:50–55.

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

[42] Xiong, K. and Perros, H. (2009). Service performance and analysis in cloud computing. In IEEE 2009 World Conference on Services, pages 693–700, Los Angeles, CA.

[43] Yang, B., Tan, F., Dai, Y., and Guo, S. (2009). Performance evaluation of cloud service considering fault recovery. In First Int'l Conference on Cloud Computing CloudCom 2009, pages 571–576, Beijing, China.

[44] Yao, D. D. (1985a). Refining the diffusion approximation for the M/G/m queue. Operations Research, 33:1266–1277.

[45] Yao, D. D. (1985b). Some results for the queues $M^x$/M/c and $GI^x$/G/c. Operations Research Letters, 4(2):79–83.

[46] Benson, T and Akella, A and A. Maltz, D.A. Network traffic characteristics of
data centers in the wild. In Proceedings of the 10th ACM SIGCOMM conference on Internet
measurement (IMC '10). ACM, New York, NY, USA, 2010, 267-280.

## ADDITIONAL READING

[1] Alam, S., Barrett, R., Bast, M., Fahey, M. R., Kuehn, J., McCurdy, C., Rogers, J., Roth, P., Sankaran, R., and Vetter, J. S. (2008). Early evaluation of IBM BlueGene. 2008 SC International Conference for High Performance Computing Networking Storage and Analysis, pages 1–12.

[2] Baig, A. (2010). UniCloud Virtualization Benchmark Report. White paper. http://www.oracle.com/us/technologies/linux/intel-univa-virtualization-400164.pdf.

[3] Cao, J., Zhang, W., and Tan, W. (2012). Dynamic control of data streaming and processing in a virtualized environment. Automation Science and Engineering, on IEEE Transactions, 9(2):365–376.

[4] Deelman, E., Singh, G., Livny, M., Berriman, B., and Good, J. (2008). The cost of doing science on the cloud: The montage example. 2008 SC International Conference for High Performance Computing Networking Storage and Analysis, C (Nov.):1–12.

[5] Duy, T. V. T., Sato, Y., and Inoguchi, Y. (2010). Performance evaluation of a green scheduling algorithm for energy savings in cloud computing. In Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), on 2010 IEEE International Symposium, pages 1–8.

[6] Feitelson, D. G. (2012). Workload Modeling for Computer Systems Performance Evaluation. Jerusalem, Israel. Version 0.36.

[7] Gandhi, A., Gupta, V., Harchol-Balter, M., and Kozuch, M. A. (2010). Optimality analysis of energy-performance trade-off for server farm management. Performance Evaluation, 67(11):1155–1171.

[8] Hewlett-Packard Development Company, Inc. (2012). An overview of the VMmark benchmark on HP Proliant servers and server blades. White paper. ftp://ftp.compaq.com/pub/products/servers/benchmarks/VMmark_Overview.pdf.

[9] Iosup, A., Ostermann, S., Yigitbasi, N., Prodan, R., Fahringer, T., and Epema, D. (2011a). Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing. IEEE Transactions on Parallel and Distributed Systems, 22(6):931–945.

[10] Iosup, A., Yigitbasi, N., and Epema, D. (2011b). On the performance variability of production cloud services. In 11th IEEE/ACM Interna-tional Symposium on Cluster, Cloud and Grid Computing, pages 104–113.

[11] Kameda, H., Li, J., Kim, C., and Zhang, Y. (1997). Optimal load balancing in distributed computer systems. Springer, London, UK.

[12] Kieffer, S., Spencer, W., Schmidt, A., and Lyszyk, S. (2003). Planning a Data Center. white paper. http://www.nsai.net/White_Paper-Planning_A_Data_Center.pdf

[13] Li, K. (1998). Optimizing average job response time via decentralized probabilistic job dispatching in heterogeneous multiple computer systems. The Computer Journal, 41(4):223–230.

[14] Li, K. (2002). Minimizing the probability of load imbalance in heterogeneous distributed computer systems. Mathematical and Computer Modelling, 36(9-10):1075–1084.

[15] Martinello, M., Kaniche, M., and Kanoun, K. (2005). Web service availability–impact of error recovery and traffic model. Reliability Engineering System Safety, 89(1):616.

[16] Meisner, D., Gold, B. T., and Wenisch, T. F. (2009). Powernap: eliminating server idle power. SIGPLAN Not., 44(3):205–216.

[17] NephoScale (2012). The NephoScale Cloud Servers. Website. http://www.nephoscale.com/nephoscale-cloud-servers.

[18] Palankar, M. R., Iamnitchi, A., Ripeanu, M., and Garfinkel, S. (2008). Amazon S3 for science grids: a viable solution? Proceedings of the 2008 international workshop on Dataaware distributed computing DADC 08, pages 55–64.

[19] Saini, S., Talcott, D., Jespersen, D., Djomehri, J., Jin, H., and Biswas, R. (2008). Scientific application-based performance comparison of sgialtix 4700, IBM power5+, and SGI ICE 8200 supercomputers. 2008 SC International Conference for High Performance Computing Networking Storage and Analysis, pages 1–12.

[20] Sato, N. and Trivedi, K. S. (2007). Stochastic modeling of composite web services for closed-form analysis of their performance and reliability bottlenecks. Service Oriented Computing ICSOC 2007, pages 107–118.

[21] SearchDataCenter.com (2008). The data center purchasing intentions survey report. Special Report. http://searchdatacenter.techtarget.com.

[22] Shirazi, B. A., Kavi, K. M., and Hurson, A. R., editors (1995). Scheduling and Load Balancing in Parallel and Distributed Systems. Wiley-IEEE Computer Society Press, Los Alamitos, CA, USA.

[23] Somani, G. and Chaudhary, S. (2009). Application performance isolation in virtualization. In IEEE International Conference on Cloud Computing, CLOUD 09, pages 41–48.

[24] Tantawi, A. N. and Towsley, D. (1985). Optimal static load balancing in distributed computer systems. J. ACM, 32:445–465.
[25] VMware, Inc. (2006). VMmark: A Scalable Benchmark for Virtualized Systems. Technical Report. http://www.vmware.com/pdf/vmmark_intro.pdf.

[26] VMware, Inc. (2012). VMware VMmark 2.0 benchmark results. Website. http://www.vmware.com/a/vmmark/.

[27] Walker, E. (2008). Benchmarking Amazon EC2 for high-performance scientific computing. LOGIN, 33(5):18–23.

[28] Wang, L., Zhan, J., Shi, W., Liang, Y., and Yuan, L. (2010). In cloud, do mtc or htc service providers benefit from the economies of scale? Proceedings of the 2nd Workshop on ManyTask Computing on Grids and Supercomputers MTAGS 09, 2:1–10.

[29] Ye, K., Jiang, X., Ye, D., and Huang, D. (2010). Two optimization mechanisms to improve the isolation property of server consolidation in virtualized multi-core server. In 12th IEEE International Conference on High Performance Computing and Communications (HPCC), pages
281–288.

[30] Youseff, L., Wolski, R., Gorda, B., and Krintz, R. (2006). Para virtualization for hpc systems. In Proc. Workshop on Xen in High-Performance Cluster and Grid Computing, pages 474–486. Springer.

Keyword: cloud computing, performance modeling, quality of service, response time, queuing theory, stochastic process.