Performance Evaluation of Cloud Computing Centers with General Arrivals and Service

Tulin Atmaca, Thomas Begin, Alexandre Brandwajn, Hind Castel-Taleb

Performance Evaluation of Cloud Computing Centers with General Arrivals and Service

Tulin Atmaca,Thomas Begin,Alexandre Brandwajn, and Hind Castel-Taleb

Abstract—Cloud providers need to size their systems to determine the right amount of resources to allocate as a function of customer's needs so as to meet their SLAs (Service Level Agreement), while at the same time minimizing their costs and energy use. Queueing theory based tools are a natural choice when dealing with performance aspects of the QoS (Quality of Service) part of the SLA and forecasting resource utilization. The characteristics of a cloud center lead to a queueing system with multiple servers (nodes) in which there is potentially a very large number of servers and both the arrival and service process can exhibit high variability. We propose to use a *G/G/c*-like model to represent a cloud system and assess expected performance indices. Given the potentially high number of servers in a cloud system, we present an efficient, fast and easy-to-implement approximate solution. We have extensively validated our approximation against discrete-event simulation for several QoS performance metrics such as task response time and blocking probability with excellent results. We apply our approach to examples of system sizing and our examples clearly demonstrate the importance of taking into account the variability of the tasks arrivals and thus expose the risk of under- or over-provisioning if one relies on a model with Poisson assumptions.

Index Terms—Cloud computing, performance evaluation, quality of service, blocking probability, response time, approximation, queueing model, general distribution.

1 Introduction

Cloud-based services have become ubiquitous and permeate our every-day life. Unlike the traditional approach where companies rely on their own computing, storage and network resources to handle the user's demands, cloud computing provides their users with on-demand services that are accessed over a network (most often Internet) [DIK09, ARM10, XU12]. SaaS (Software as a Service) environments, in which software is hosted centrally in a cloud, is a case in point, and has become a standard model for a number of business and multimedia applications. Other approaches to cloud computing include IaaS (Infrastructure as a Service) and PaaS (Platform as a Service). In either case, the cloud architecture offers many advantages including economies of scale, fast deployment of new features, quick bug fixes and potential cost-saving through the "pay-as-you-go" model.

A cloud computing center typically consists of many computing nodes that process the tasks (sometimes also called transactions or requests) initiated by users as illustrated in Figure 1.

The cloud provider and the user agree together on a contract, referred to as the SLA (Service Level Agreement), which formally defines the expected scope and quality of service. The issues addressed by this contract include security, reliability, availability and performance. The performance obligations are often expressed in terms of such QoS indicators as the average delay before being served, i.e., the expected waiting time for a task, the task blocking probability, the no-wait probability, etc.

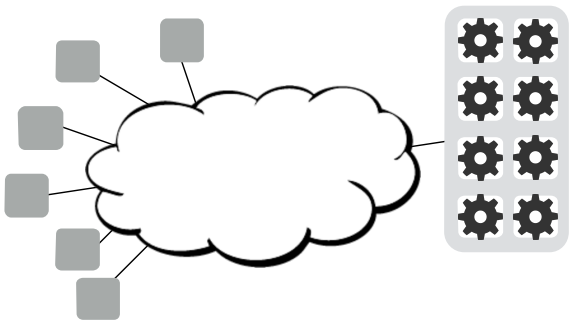


Fig. 1: Overview of a cloud center providing service to remote users.

On the other hand, cloud providers have a degree of control over the management of their resources and can tune some parameters. They strive to make the cloud as elastic as possible in order to provision the right amount of resources for each user according to the user's demand. In other words, cloud providers attempt to avoid over-provisioning while at the same time striving to meet their negotiated SLAs. In this perspective, the resource utilization becomes a key parameter that reflects how well resources are utilized.

Thus, cloud providers are faced with the problem of sizing their systems so as to meet their SLAs, while at the same time minimizing their costs and energy use. This

- *T. Atmaca and H. Castel-Taleb are with Institut Telecom, Telecom Sud-Paris, France.*
- *T. Begin is with Université Lyon 1 / LIP (UMR INRIA, ENS Lyon CNRS, UCBL), France.*
  E-mail: see http://perso.ens-lyon.fr/thomas.begin/
- *A. Brandwajn is with Baskin School of Engineering, University of California Santa Cruz, USA.*

Manuscript received Month Day, Year; revised Month Day, Year.

implies determining the right (as little as possible) amount of resources to allocate as a function of customer's needs. To assist them in this provisioning task, cloud providers need appropriate tools. In that respect, queueing theory based tools seem a natural choice when dealing with performance aspects of the QoS, viz. blocking probability, average delay for a task, etc, as well as resource utilization.

A cloud center can be viewed as a set of computing nodes that execute user tasks. User tasks arrive to the computing nodes according to some pattern (arrival process), potentially queue for service and are eventually treated by a node in accordance with the needs of the task (service process). This leads naturally to a queueing system with multiple servers (nodes). However, cloud-based systems present a number of challenges for standard queueing models:

- there is a potentially very large number of computing resources (several hundred);
- the service process can be highly variable;
- the arrival process can exhibit high variability as well.

Standard queueing models rarely consider hundreds of servers. High variability of the service process means that the coefficient of variation, i.e., the ratio of the standard deviation to the mean service time can be much larger than one. The coefficient of variation (cv) is a unitless measure of variability. The more variable the distribution, the greater its coefficient of variation. Similarly, high variability of the arrival process means that the coefficient of variation of the time between task arrivals to the computing nodes can be much larger than 1. This in turn implies that queueing models with Poisson arrivals, which assume a very specific pattern of arrivals with a coefficient of variation of exactly 1, are likely not an adequate general representation of cloud systems.

In this paper, as a generalization of the work of previous authors [XIO09, YAN09, KHA12], we propose to use a model with multiple servers, general times between task arrivals and general task service time (*G/G/c*-like queue) and we present a simple-to-implement, efficient and accurate approximate solution which uses available off-the-shelf components. Because the multiple servers in our model, like in the work cited before, are assumed to be statistically identical, the model is a high-level general representation of a system with reasonably homogeneous processing nodes. For systems in which the nodes are far from homogeneous, it may be possible to account for the differences in processing speeds by adjusting appropriately the variability of the service times in the model. A significant level of complexity would be added if nodes were to be considered individually.

The next section is devoted to a brief review of existing related work. Section 3 presents our model and its solution, including the easy derivation of performance metrics of interest. Section 4 summarizes the results of an extensive validation of our approximate solution. It presents also data on its convergence and relative speed. Section 5 shows examples of the application of our model to the sizing of a cloud system. Section 6 concludes this paper.

2 Related work

Despite the ubiquitous presence of cloud systems and the considerable research attention devoted to such systems, there seems to be a relatively limited number of studies which apply queueing theory models to cloud systems. Initial research work in this area started by assuming exponential distributions throughout the system. Xiong and Perros in 2009 [XIO09] model the cloud as an open queueing network and obtain the approximate distribution of the task response time and related performance metrics under the assumption of exponential service and inter-arrival time (coefficients of variation equal to 1 for both). Yang and al. [YAN09] the same year obtain the approximate distribution of task response time in a multi-server queue under the same restrictive assumption regarding service and inter-arrival distributions. Subsequent research work seeks to relax the restrictive exponential assumption. Khazaei et al. in 2012 [KHA12] present an approximate solution of a multi-server queue with general service time but Poisson arrivals (i.e. exponentially distributed times between arrivals). Yang et al. [YAN13] extend in 2013 their approximate solution [YAN09] to account for general non-Poisson task arrivals while the service times are assumed to be exponential. Additionally, in 2013, Khazaei et al. [KHA13] and Singh et al. [SIN13] provide more fine-grained performance models for the cloud systems but retain the restrictive Poisson arrival assumption.

Our model and its approximate solution allow us to relax both restrictive assumptions on service and inter-arrival distributions. With general distributions for the service times and the times between arrivals, the resulting model is known as the *G/G/c/N* queue where N is the finite capacity of the system in terms of the numbers of tasks it can accept at any given time (also referred to as the buffer space). The general analytical solution of this model is not known so that a common approach (barring simulation) is to represent the "general" distributions by their phase-type equivalents. Well established techniques exist and are readily available as a free software service to effect this translation [HOR02, BOB05, OSO06]. The resulting *Ph/Ph/c/N* queueing system is then solved numerically as a specific system of linear equations.

As long as the number of servers c and the number of phases (used to represent the general distributions) in the model remain moderate, the equations of the *Ph/Ph/c/N* queue can be solved via direct iteration [TAK76, SEE86] or, more efficiently and elegantly, via matrix-geometric methods [RAM86, LAT99, BIN05]. However, as the number of servers and phases grows, the number of equations to solve grows combinatorially ("dimensionality curse"), effectively precluding the exact numerical solution of systems with larger numbers of servers. This may be the case for *Ph/Ph/c/N* queues with as few as 32 servers and a buffer space as small as 64. Unfortunately, larger number of servers is precisely what is needed to represent many cloud environments. Additionally, to represent a realistic arrival pattern it may be necessary to use more than just a few phases.

While a few approximations have been proposed in the literature (see [WHI93] and [BOL05] for an overview), most of these approximations turn out to be of questionable

accuracy [WOL77, WHI80, GUP10, BEG13]. This is the case for approximations based on the first two moments of the service time and inter-arrival time distributions. Other existing approximations (e.g. [WHI04, TAK07, KIM03]) may rely on "heavy traffic limits" and come with their own specific complexity and limitations. Thus, there is a clear need for an efficient and reasonably simple solution of such queues that scales easily as the number of servers increases. We propose precisely such a solution.

3 Model and its Solution

The *Ph/Ph/c/N* queueing model under consideration is represented in Figure 3. We denote by a and b the number of phases used to represent the distributions of the time between arrivals and of the service time, respectively. A phase-type distribution is a representation of a general distribution as a set of exponential phases, and any general distribution can be represented arbitrarily closely by a finite number of such exponential phases [BOL05].

The service time distribution represented as a phase-type distribution in Figure 2. This distribution comprises b exponential phases, where μ_i is the parameter of the corresponding exponential phase i (phase intensity) with $i = 1, \ldots b$. With probability σ_i, the service starts in phase i. Upon completion of phase i, with probability $\hat{q}_i$ the task service is over, and with probability q_{ik} the service continues in phase k. For simplicity, we consider acyclical distributions, i.e. $q_{ik} = 0$ if $k \leq i$. Similarly, the arrival process is represented by a phase-type distribution with a exponential phases where λ_j is the intensity of phase j, and the parameters τ_j, $\hat{r}_j$ and r_{jm} correspond to parameters σ_i, $\hat{q}_i$ and q_{ik} of the service time distribution, respectively.

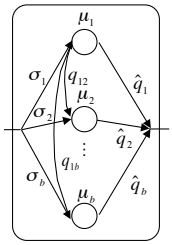


Fig. 2: Phase-type representation of a general service time distribution.

For simplicity of exposition, the above description assumes no state-dependency of the service times and times between arrivals. Without difficulty, the model can be extended to include state dependencies by making the phase intensities, as well as the probabilities of selecting a specific phase depend on the current number of users of tasks in the system. Note that a stochastic queueing model like the one considered in this paper incorporates by its very nature a level of dynamic behavior, and even more so with state-dependent phase parameters.

We denote by $p(n), n = 0, ..., N$ the steady-state probability that there are n tasks in the system (queued and in service).

In our method, we iterate between the solutions of simpler *M/Ph/c/N* and *Ph/M/c/N* queues, in which one of the

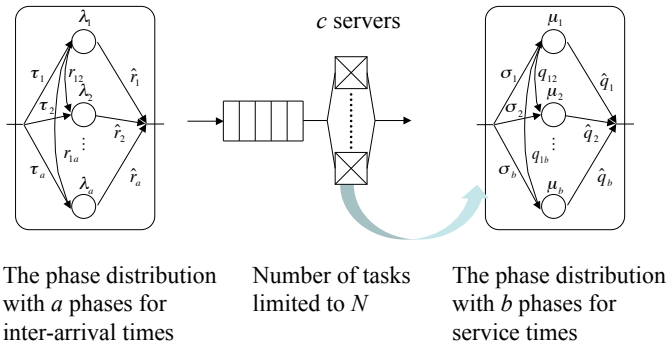


Fig. 3: The *Ph/Ph/c/N* queue.

phase distributions is replaced by a memoryless (the *M* part of notation) state-dependent distribution. For the *M/Ph/c/N* queue, the arrivals are represented by a state-dependent rate of arrivals $w(n), n \geq 0$, and the service time distribution is the complete phase-type distribution with b phases. The solution of this queue produces approximate values for $p(n)$ and the conditional rate of service $u(n), n \geq 1$ given that there are n tasks in the system. This rate of service is used to solve the *Ph/M/c/N* queue with the complete phase-type distribution of the time between arrivals with a phases. The solution of this queue produces approximate values for $p(n)$, as well as the conditional rate of arrivals given that there are n tasks in the queue, $w(n), n \geq 0$ (see Figure 4).

Thus, we need the $w(n)$ to solve the *M/Ph/c/N* queue and obtain the $u(n)$ needed to solve the *Ph/M/c/N* queue to produce the values of $w(n)$, naturally leading to a fixed-point iteration. We stop the iteration when the steady-state distributions $p(n)$ produced by the two models become sufficiently close, as measured by the mean number of tasks in the system $L = \sum_{n=0}^{N} n.p(n)$.

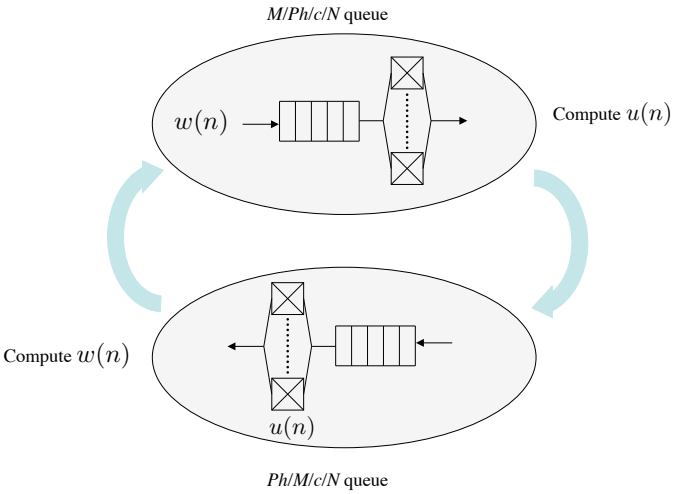


Fig. 4: Schematic view of the approximate solution.

The resulting fixed-point iteration is summarized in Algorithm 1.

Note that we have $u(n) = p(n-1).w(n-1)/p(n)$ and that the steady-state probability $p(n)$ can be expressed as

$$p(n) = \frac{1}{G} \prod_{i=1}^{n} \frac{w(i-1)}{u(i)} \text{ for } n = 0, 1, ..., N \qquad (1)$$

Algorithm 1 Fixed-point solution

1: Initialize the values of $w(n), n \geq 0$ to the inverse of the mean time between arrivals.
2: Solve the *M/Ph/c/N* queue using the current values of $w(n), n \geq 0$ (these values come from Step 1 on the first iteration and from Step 3 afterwards).
   a: Obtain current values for $p(n)$ and for $u(n)$.
   b: Compute the current value of L from this model.
3: Solve the *Ph/M/c/N* queue using the current values of $u(n), n \geq 1$ from Step 2.
   a: Obtain current values for $p(n)$ and for $w(n)$.
   b: Compute the current value of L from this model.
4: If the values of L from Step 2 and Step 3 deviate by less than $\epsilon > 0$ then stop the iteration, otherwise go to Step 2.
5: Use the current values of $p(n)$ and $w(n)$ to compute desired performance metrics.

where G is a normalizing constant, i.e. $G = \left[1 + \sum_{n=1}^{N} \prod_{i=1}^{n} \frac{w(i-1)}{u(i)}\right]^{-1}$. We solve the *Ph/M/c/N* queue using the fast and stable recurrence described in [BRA12], which produces directly the values of $w(n), n \geq 0$. We use the reduced-state approximation to solve the *M/Ph/c/N* queue [BRA14]. The state-space complexity in this approximation grows linearly with the number of servers and service time phases.

Note that formula (1) would be exact if we had the exact values of the conditional rate of arrivals $w(n)$ and of the conditional service rate $u(n)$.

Having obtained the steady-state probabilities $p(n)$, it is a straightforward matter to compute the probabilities of the number of tasks found by a task upon arrival to the system, denoted by $P_A(n)$ as:

$$P_A(n) = \frac{w(n)p(n)}{\sum_{i=0}^{N} w(i)p(i)} \text{ for } n = 0, \ldots N. \tag{2}$$

The blocking probability p_{block} is then simply $P_A(N)$ i.e., the probability that an arriving task finds the system full to capacity. Note that in the particular case when the time between arrivals is exponentially distributed (Poisson process), the probabilities upon arrivals $P_A(n)$ happen to be exactly the same as the steady-sate probabilities $p(n)$ [BOL05]. We assume in our model that tasks continue to arrive to the system even when it has reached its capacity. It is not difficult to treat the case where the arrivals become blocked until there is room in the system. In our solution, this requires only a straightforward modification of the *Ph/M/c* part of the solution, as described in [BRA12]. No modification is required in the *M/Ph/c* part of our solution as the blocking of arrivals will be automatically represented by the state-dependent rate of arrivals in this model.

The probability that a task experiences no wait, $p_{\text{no\_wait}}$, can be computed as:

$$p_{\text{no\_wait}} = \sum_{n=0}^{c-1} P_A(n). \tag{3}$$

TABLE 1: Notation used.

Symbol	Description
a	Number of phases for the inter-arrival time distribution
b	Number of phases for the service time distribution
cv_a	Coefficient of variation for the inter-arrival time distribution
cv_s	Coefficient of variation for the service time distribution
c	Number of servers
N	Buffer space, i.e. maximum number of tasks in the system (queued and in service)
$p(n)$	Marginal probability that there are n tasks in the system
$u(n)$	Overall departure rate from the set of c servers given that the current number of tasks in the system is n
$w(n)$	Arrival rate at the queue given that the current number of tasks in the system is n
$P_A(n)$	Probability that an arriving task finds n tasks in the system
p_{block}	Blocking probability (i.e. probability that a task finds the system at capacity full upon arrival)
$p_{\text{no\_wait}}$	No-wait probability (i.e. probability that a task experiences no wait)
U	Server utilization
L	Mean number of tasks in the system
Θ	Mean task throughput
R	Mean task response time

The server utilization (per server) can be expressed as:

$$U = \sum_{n=1}^{c-1} p(n).n/c + \sum_{n=c}^{N} p(n). \tag{4}$$

The mean number of tasks in the system can be obtained as:

$$L = \sum_{n=1}^{N} n.p(n). \tag{5}$$

The task throughput, i.e., the number of tasks processed per time unit is given by:

$$\Theta = \sum_{n=1}^{N} u(n).p(n). \tag{6}$$

Hence, the mean response time of a task, denoted by R, can be computed using Little's formula [BOL05]:

$$R = \frac{L}{\Theta}. \tag{7}$$

Regrettably, we don't have a theoretical proof of the convergence of the proposed fixed-point iteration to a unique solution. In the several thousand cases we have explored using the value of $\epsilon = 10^{-9}$ for the exit test, the method never failed to converge within typically just a few tens of iterations.

The proposed approach decomposes the solution of a *Ph/Ph/c/N* queue into the solution of an *M/Ph/c/N* queue with state-dependent rate of arrivals, and that of a *Ph/M/c/N* queue with state-dependent service rates. Such decomposition would be exact if we knew the rates of arrivals as a

function of both the number of tasks n and the current phase of the service process, as well as the rates of service as a function of n and of the current phase of the arrival process. Since we determine them only as a function of n, the method is approximate. Intuitively, this would only matter when the number of servers is small (say, less than 4).

4 Accuracy and Performance of the Proposed Approach

We explored a number of cases to study the accuracy of our approximation. The cases considered encompass the following ranges of parameter values:

- Number of servers c : 32, 64, 128, 256, 512;
- System capacity (buffer space): N : $2c$ and $3c$;
- Offered load : 0.4, 0.6, 0.8, 1. 2;
- Coefficient of variation of the service time cv_s: 0.5, 2, and 3;
- Coefficient of variation of the time between arrivals cv_a: 0.5, 2, and 3.

We keep the mean service time set to 1. To define different workload levels, in the case of a model without state dependencies we use the notion of offered load per server, defined as the ratio of the mean rate of task arrivals (including arrivals lost due to system overflow, i.e. blocking). This definition stems from the fact that the maximum processing capacity of the system is determined by the product of the number of servers (c) times the mean processing rate for a task (taken to be 1 here). Thus, an offered load per server of 1 or more corresponds to a system operating at its maximum processing capacity where the mean rate of task arrivals is equal to or exceeds the processing capacity of the servers.

We use the coefficients of variation to describe the variability of the times between task arrivals and their service times, denoted cv_a and cv_s, respectively. Recall that the coefficient of variation is defined as the ratio of the standard deviation to the mean of a random variable (or a sample).

We use discrete-event simulation as comparison basis to assess the accuracy of our approach. Our simulation runs employ the independent replication method with 7 replications of 50,000,000 task completions each. The resulting estimated confidence intervals at 95% confidence level are so small that we use only the mid-point in our validation.

Table 2 summarizes the relative error of the proposed solution for the mean number of tasks L. In this set of examples, the mean relative error is 0.3%, the median error is 0.1%, and there are no cases where the relative error exceeds 5%.

TABLE 2: Distribution of the relative errors for the mean number of tasks L.

Mean	Median	<1%	1-5%	5-10%	>10%
0.3%	0.1 %	93.6%	6.4%	0%	0%

Table 3 shows the relative error for the blocking probability p_{block}. Note that for the blocking probability to have meaningful values, we include in this table only cases where the blocking probability exceeds 0.01. We observe that the mean relative error is below 1%, the median error 0.1% and in over 93% of cases the relative error remains below 5%.

TABLE 3: Distribution of the relative errors for the blocking probability p_{block}.

Mean	Median	<1%	1-5%	5-10%	>10%
0.8%	0.1 %	89.6%	4.0%	4.0%	2.4%

Table 4 summarizes the relative error for the mean task response time R. We observe that the mean error is 0.3%, the median error is a mere 0.1% and in only some 8% of cases the error is greater than 1%.

TABLE 4: Distribution of the relative errors for the mean task response time R.

Mean	Median	<1%	1-5%	5-10%	>10%
0.3%	0.1 %	92.4%	7.6%	0%	0%

Table 5 shows the relative error for the server utilization. This error is computed as $|1 - U_{\text{approximation}}/U_{\text{simulation}}|$. Analogous computation was used for relative errors presented in Tables 2, 3 and 4. We observe the excellent accuracy of our method for this metric.

TABLE 5: Distribution of the relative errors for the server utilization U.

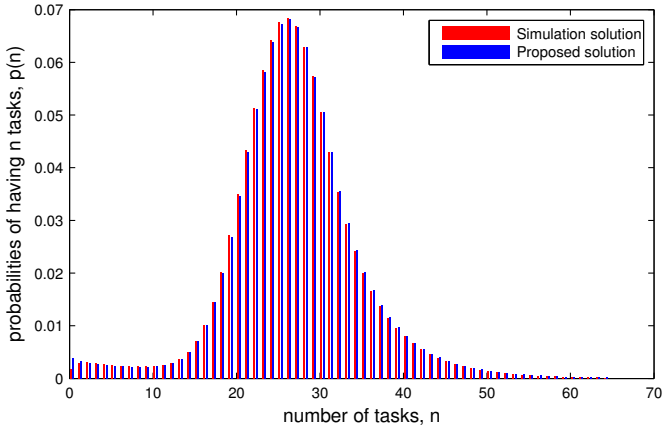
Mean	Median	<1%	1-5%	5-10%	>10%
0.08%	0.03 %	98.7%	1.3%	0%	0%

The proposed approximation produces accurate results not only for the performance metrics considered but also generally correctly reproduces the shape of the steady-state distribution of the number of tasks in the system, $p(n)$. Figure 5 illustrates this fact for an example with the following parameters: $c = 32$, $N = 2c$, $cv_a = 2$, $cv_s = 2$ and an offered load of 0.8 and 1.0. We observe a close agreement between the results of our approximation and the exact probability distribution.

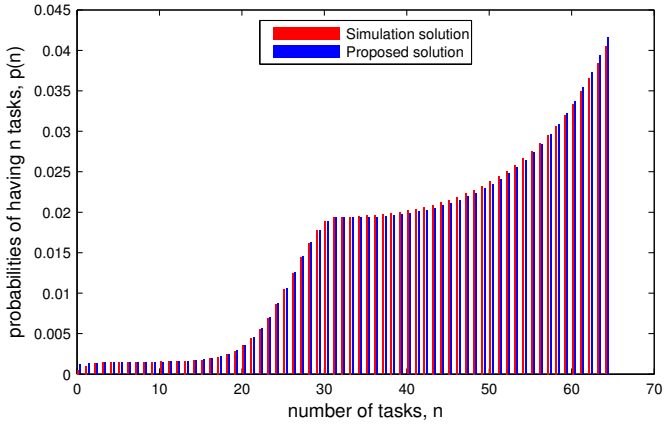
As a final example, we consider a system with state dependencies. In this example, the service times of tasks increase as the number of tasks increases beyond the number of servers c. Specifically the service rate of each server decreases linearly with the length of the queue from 1 with no tasks queued down to 0.7 when the system is at capacity. Figure 6 shows the mean number of tasks in the system as a function of the offered load per server. We observe that the values produced by our approximation closely match the exact values.

In all examples presented in this paper, both the distributions of the service time and of the time between arrivals comprise 4 phases and are obtained using an adaptation of the algorithm by Bobbio et al. [BOB05]. The choice of 4 phases is purely arbitrary and in no way a limitation of the proposed approach. Note that if only the first two moments of the empirical distribution are known and its coefficient of variation is greater than 0.7, then a two-phase distribution is sufficient to match the known two moments.

Note also that distribution fitting algorithms such as [BOB05, OSO06] produce acyclical distribution as assumed in our paper. This allows us also to use the simple recurrence solution of the *Ph/M/c* queue.



(a) Offered load of 0.8



(b) Offered load of 1.0

Fig. 5: Distributions of the number of tasks, $p(n)$, produced by the exact and our approximate solutions.

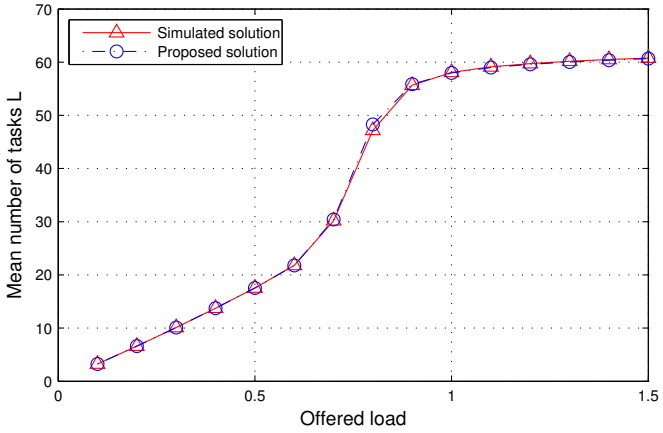


Fig. 6: Mean number of tasks L in the system with $c = 32$, $N = 2c$, $cv_a = 3$, $cv_s = 3$ and state dependencies for the service process for various levels of workload.

Additionally, it is worth noting that when the coefficients of variation for the time between arrivals and the service time are both equal to 1 (exponential distributions) our approach produces exact results.

Since our method iterates between the simpler solutions of the *M/Ph/c/N* and *Ph/M/c/N* queues, it is interesting to examine the speed of convergence of this iteration on the whole set of our examples. Table 6 summarizes the number of iterations needed to attain convergence (with $\epsilon = 10^{-8}$). We observe that on average some 6 iterations are needed. It is important to note that each iteration comprises the execution of an approximate *M/Ph/c/N* solution and an exact recurrent solution of the *Ph/M/c/N* queues. The execution time of the former grows somewhat faster than linearly with the number of servers, while the complexity of the recurrent solution grows only linearly with the number servers.

TABLE 6: Number of iterations till convergence in the approximate solution.

Mean	Median	<5	5-10	10-20	>20
6.0	4.0	59.9%	21.3%	14.6%	4.2%

A more detailed study of the convergence behavior of the proposed approach reveals that, if one considers the mean number in the system, the convergence to the final result is very fast: on average 3 iterations suffice. This is illustrated in Figure 7, as well as in Table 7, which show that after 3 iterations the relative difference between the current result and the final value is below 1% (in absolute value) in all cases considered.

TABLE 7: Number of cases having a discrepancy less than 1% with the (approximate) value of L found at convergence.

After iteration	1	2	3
	91.1%	97.8%	100%

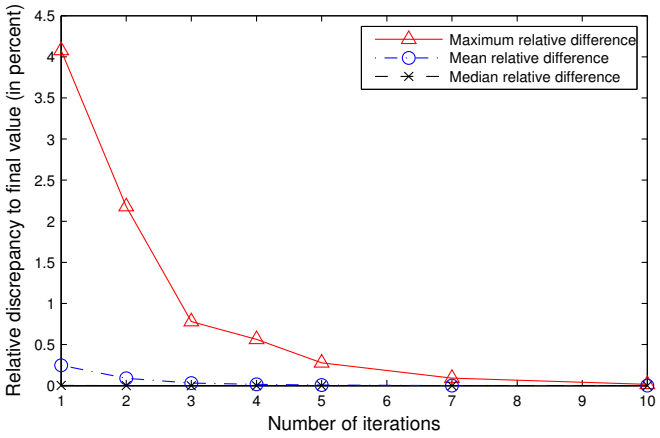


Fig. 7: Relative discrepancy of intermediate results of the proposed solution (at each iteration) with its final solution.

As a final point in this section, we look at the relative speed of the proposed approach as compared to discrete-event simulation (with the simulation parameters used throughout this paper). We observe in Table 8 that our method is some 2 to 4 orders of magnitude faster.

Overall, based on the results shown and on many more results not reported in this paper, we conclude that the relative error of our approximate solution for the performance indices considered L, R, p_{block} and U can be expected to

TABLE 8: Relative execution time compared to simulation.

Mean	Median	<100	100-1,000	1,000-10,000	>10,000
4154	298	35.2%	33.9%	19.6%	11.3%

be well under 5%, and rarely, if ever, exceed 10% for an extremely large range of values of model parameters. Compared to simulation the proposed solution is, on average, 3 orders of magnitude faster.

5 Example of System Sizing

To illustrate the application of our model, we consider the problem of determining the right amount of resources in a cloud system so as to meet a given QoS level. We start by the probability that tasks don't have to wait for service $p_{\text{no\_wait}}$. Our objective is for 95% of tasks to experience no wait, i.e. $p_{\text{no\_wait}} \geq 0.95$. We are seeking to determine the minimum number of servers which will allow us to meet the specified QoS objective. We maintain the maximum system capacity at twice the number of servers. The mean service time is normalized to 1 and the service time coefficient of variation is set to 2. The offered load is kept at 0.8 and the buffer space is set to $N = 2c$.

The results of our model shown in Figure 8 indicate that when the coefficient of variation of the time between arrivals cv_a is small (0.5), it is sufficient to provision 50 server to attain this QoS level. With Poisson arrivals ($cv_a = 1$), this number becomes 75 and increases all the way to almost 110 servers with more variable times between arrivals ($cv_a = 3$). It is thus plainly clear that using a model with Poisson arrivals leads to significant under- or over-provisioning depending on the variability of the arrivals.

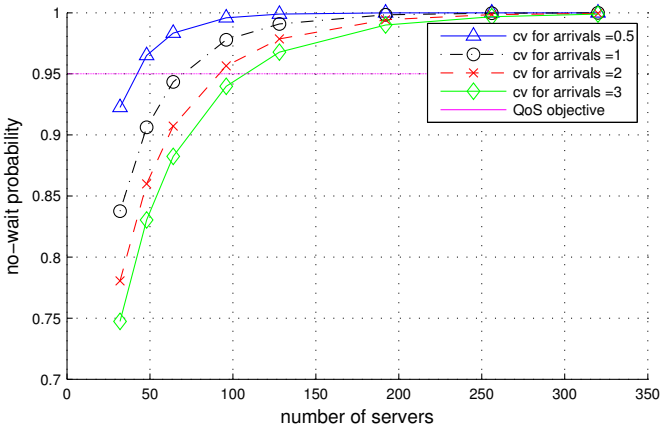


Fig. 8: Sizing the cloud system according to the predicted no-wait probability $p_{\text{no\_wait}}$.

In our second example, we examine the sizing of the maximum system capacity so as to maintain a blocking probability p_{block} not exceeding 1% ($p_{\text{block}} \leq 0.01$). The number of server is kept constant at $c = 256$ and the offered load is 1. As in our first example, the mean service time is normalized to 1 and the service time coefficient of variation is set to 2.

We observe in Figure 9 that with low arrival variability ($cv_a = 0.5$), a total system capacity of 315 is adequate. With Poisson arrivals ($cv_a = 1$), the necessary system capacity is 360, and it grows to almost 770 when the variability of the time between arrivals reaches 3 ($cv_a = 3$). It important to note that the necessary waiting space size i.e., $N - c$, to meet the specified QoS objective grows almost fivefold (from a little over 100 to 500) as the coefficient of variation varies from 1 (Poisson arrivals) to 3. As in our first example, it is clear that one cannot rely on the results of a model with Poisson arrivals to reliably size a cloud system.

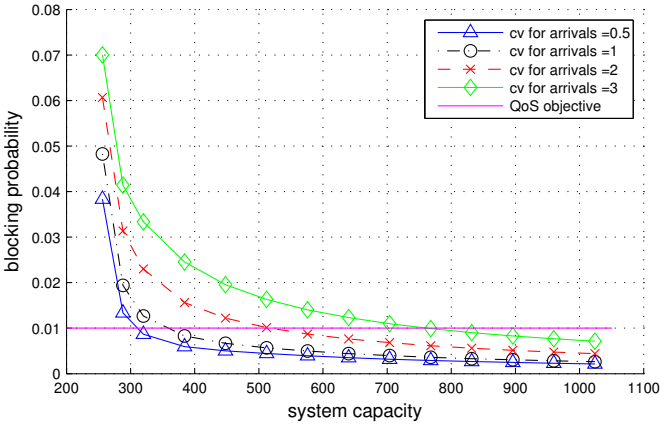


Fig. 9: Sizing the cloud system according to the predicted blocking probability p_{block}.

The next section summarizes the conclusions of this paper.

6 Conclusion

In this paper we argue the importance of performance evaluation tools to properly size a cloud system so as to meet performance aspects of the SLA, as well as minimize the amount resources the cloud provider needs to provision to meet the agreed upon QoS levels. Due to the variability of cloud workloads, we propose to use a *G/G/c*-like model to represent a cloud-based system and compute expected performance indices. The advantage of such a model is that it represents general distributions of workloads in the cloud system with respect to both the arrival and service patterns.

Given the potentially high number of servers in a cloud system, we present an efficient approximate solution. We have extensively validated our approximation against discrete-event simulation for several QoS performance metrics such as task response time and blocking probability with excellent results. Our validation is limited to the accuracy of the approximate solution. It does not validate the adequacy of the general model itself in a specific cloud environment. Given the number of potential issues in such a validation (e.g. erroneous or inconsistent data, non-representative behavior [FEI14]), this is the intended subject of future work.

The fast execution speed of our approximation allows us to quickly explore large ranges of parameters in order to determine appropriate resource provisioning levels. We apply our approach to examples of system sizing and our examples clearly demonstrate the importance of taking into account the variability of the tasks arrivals and thus expose

the risk of under- or over-provisioning if one relies on a model with Poisson assumptions.

A simple implementation of the proposed model is available for experimentation on the website http://queueing-systems.ens-lyon.fr. Extensions of the proposed approach to include state-dependent arrivals and service are possible.

ACKNOWLEDGMENTS

The authors would like to express their sincere thanks to the anonymous referees for their remarks and comments.

REFERENCES

[ARM10] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., ... and Zaharia, M. A view of cloud computing. Communications of the ACM, 53(4), pp. 50-58 (2010).

[BEG13] Begin, T., and Brandwajn, A. A note on the accuracy of several existing approximations for *M/Ph/m* queues. In Proceedings of HSNCE (2013).

[BIN05] Bini, D. A., Latouche, G., and Meini, B. Numerical Methods for Structured Markov Chains. Oxford University Press, Inc. (2005).

[BOB05] Bobbio, A., Horváth, A., and Telek, M. Matching three moments with minimal acyclic phase type distributions, Stochastic Models. 21, pp. 303-326 (2005).

[BOL05] Bolch, G., Greiner, S., Meer, H., and Trivedi, K. Queueing Networks and Markov Chains. Second Edition, Wiley-Interscience (2005).

[BRA12] Brandwajn, A., and Begin, T. A Recurrent Solution of *Ph/M/c/N*-like and *Ph/M/c*-like Queues. Journal of Applied Probability, 49.1, pp. 84-99 (2012).

[BRA14] Brandwajn, A., and Begin, T. Reduced Complexity in *M/Ph/c/N* queues. Performance Evaluation, 78, pp. 42-54 (2014).

[DIK09] Dikaiakos, M. D., Katsaros, D., Mehra, P., Pallis, G., and Vakali, A. Cloud computing: Distributed internet computing for IT and scientific research. Internet Computing, IEEE, 13(5), pp. 10-13 (2009).

[FEI14] Feitelson, D. G., Tsafrir, D., and Krakov, D. Experience with using the parallel workloads archive. Journal of Parallel and Distributed Computing, 74(10), pp. 2967-2982 (2014).

[GUP10] Gupta, V., Harchol-Balter, M., Dai, J., and Zwart, B. On the inapproximability of *M/G/K*: why two moments of job size distribution are not enough. Queueing Systems, 64 (1), pp. 5-48 (2010).

[HOR02] Horváth, A., and Telek, M. Phfit: A general phase-type fitting tool. In Proceedings of Computer Performance Evaluation: Modelling Techniques and Tools (TOOLS) pp. 82-91 (2002).

[KHA12] Khazaei, H., Misic, J., and Misic, V. B. Performance analysis of cloud computing centers using M/G/m/m+ r queuing systems. Parallel and Distributed Systems, IEEE Transactions on, 23(5), pp. 936-943 (2012).

[KHA13] Khazaei, H., Misic, J., and Misic, V. B. A fine-grained performance model of cloud computing centers. Parallel and Distributed Systems, IEEE Transactions on, 24(11), pp. 2138-2147 (2013).

[KIM03] Kimura, Toshikazu. A consistent diffusion approximation for finite-capacity multiserver queues. Mathematical and computer modelling, 38.11, pp. 1313-1324 (2003).

[LAT99] Latouche , G.,and Ramaswami, V. Introduction to Matrix Analytic Methods in Stochastic Modeling, ASA (1999).

[OSO06] Osogami, T., and Harchol-Balter, M. Closed form solutions for mapping general distributions to quasi-minimal PH distributions. Performance Evaluation, 63(6), pp. 524-552 (2006).

[RAM86] Ramaswami, V., and Lucantoni, D.M. Algorithms for the multi-server queue with phase type service, Stochastic Models, 1, pp. 393-417 (1985).

[SEE86] Seelen, L. P. An Algorithm for *Ph/Ph/c* Queues, European Journal of the Operations Research Society, 23, pp. 118-127 (1986).

[SIN13] Singh, R., Shenoy, P., Natu, M., Sadaphal, V., and Vin, H. Analytical modeling for what-if analysis in complex cloud computing applications. ACM SIGMETRICS Performance Evaluation Review, 40(4), pp. 53-62 (2013).

[TAK76] Takahashi, Y., and Takami, Y. A Numerical Method for the Steady-State Probabilities of a *GI/G/s* Queueing system in a General Class, Journal of the Operations Research Society of Japan, 19, pp. 147-157 (1976).

[TAK07] Takahashi, Akira, et al. Diffusion approximations for the *GI/G/c/K* queue. In Proceedings of 16th IEEE International Conference on Computer Communications and Networks (ICCCN), 2007.

[WHI80] Whitt, Ward. The effect of variability in the *GI/G/s* queue. Journal of Applied Probability, pp. 1062-1071 (1980).

[WHI93] Whitt, Ward. Approximations for the *GI/G/m* queue. Production and Operations Management, 2.2, pp. 114-161(1993).

[WHI04] Whitt, Ward. A diffusion approximation for the *G/GI/n/m* queue. Operations Research, 52.6, pp. 922-941 (2004).

[WOL77] Wolff, Ronald W. The Effect of Service Time Regularity on System Performance. No. ORC-77-7. California Univ Berkeley Operations Research Center (1977).

[XIO09] Xiong, K., and Perros, H. Service performance and analysis in cloud computing. In IEEE World Conference on Services-I (2009).

[XU12] Xu, X. From cloud computing to cloud manufacturing. Robotics and computer-integrated manufacturing, 28(1), pp. 75-86 (2012).

[YAN09] Yang, B., Tan, F., Dai, Y. S., and Guo, S. Performance evaluation of cloud service considering fault recovery. In Cloud computing (2009).

[YAN13] Yang, B., Tan, F., and Dai, Y. S. Performance evaluation of cloud service considering fault recovery. The Journal of Supercomputing, 65(1), pp. 426-444 (2013).