An analytical model of a cluster-based service system with application to a cloud environment

Osama Salameh[1] and Sabine Wittevrongel[2*]

[1]Faculty of Engineering, Arab American University, P.O. Box 240, Jenin, Palestine.
[2]Department of Telecommunications and Information Processing (TELIN), Ghent University, Sint-Pietersnieuwstraat 41, B-9000, Gent, Belgium.

*Corresponding author(s). E-mail(s): sabine.wittevrongel@ugent.be;
Contributing authors: osama.salameh@aaup.edu;

Abstract

Cluster-based systems have been extensively used to provide parallel processing of jobs. A distinguishing feature of such systems is that jobs consist of tasks that should run in parallel on different servers. A job does not start execution unless the required number of idle servers is available. This paper proposes a new continuous-time Markov chain that accurately models such cluster-based system with finite buffer size. Extensive performance evaluation is conducted where the influence of several model parameters on a number of performance measures is investigated. Performance measures include the blocking probability of jobs, the average delay of jobs in the queue and the utilization of the servers in the cluster. The application of the model to cloud centers with thousands of servers is shown possible under a typical heterogeneous workload where jobs require either 10 or 100 servers each.

Keywords: Continuous-time Markov chain, cloud computing, cluster-based system, performance analysis

1 Introduction

Cluster-based systems are well known to provide parallel processing of jobs [1]. While the concept of cluster computing has been introduced many years ago [2], the recent interest to study systems that operate clusters is due to the fact that cluster computing is embedded in the heart of popular cloud services [3, 4]. A basic aspect of such cluster-based systems is that cluster jobs consist of smaller units called tasks, which are executed on different servers and at the same time, hence considerably reducing the job execution time. A tracker node is responsible for tracking the available system resources at all times including the idle and busy servers.

Typically jobs arrive at a finite-capacity queue. Jobs are processed first by a job tracker node based on a First-In-First-Out (FIFO) service discipline. The tracker node is a master unit that conducts resource provisioning and keeps track of the cloud resources. Each cluster job consists of a number of tasks (referred to as the job size) to execute in parallel. These tasks are equal in length for some implementations [5]. When the required number of servers is available in accordance with the size of the job at the job tracker, the job is then mapped to the cluster servers, one task per server. Finally, when the execution of a job at the cluster servers finishes, all the servers occupied by the job are released and the job exits the system.

Cluster-based systems are difficult to analyze analytically due to their complexity and closed-form expressions of performance measures can be obtained only when the number of servers is very small as discussed in the next section. In this paper, we therefore propose a new analytical model of a cluster-based system based on a continuous-time Markov chain (CTMC) that keeps track of all jobs in the system: the jobs waiting in the queue, the job at the tracker node and the jobs in execution at the cluster. This way the model can capture the typical dynamics of cluster-based systems.

In our model, we firstly assume that jobs arrive according to a Poisson process with rate λ, as e.g. in [6, 7]. We define the service time of a job at the job tracker as the time required to map the job tasks to the cluster servers. This time starts when the number of servers required for a job is available. When this number of servers is not available yet, the job remains blocked at the tracker node. So, we consider that the job currently at the job tracker is mapped into the cluster only when a sufficient number of servers is available. We assume that the service rate of a job at the job tracker node does not depend on the size (i.e., the number of tasks or required number of parallel servers) of the job. To accommodate for heterogeneous workloads, however, we allow the service rate of a job at the cluster servers to depend on the job size. We assume that all cluster servers are the same and that all servers involved in the execution of a job are released again at the same time, as e.g. in [8]. So, essentially we assume that both the service time of a job at the job tracker and the service time of a job of size n at the cluster are exponentially distributed with mean values $1/\mu_1$ and $1/\mu_{2,n}$ respectively, similar to [6, 9]. As a justification for these assumptions, we first note that even under Poisson arrivals and exponential service times the performance of cluster-based systems with more than two servers is still an open problem (see Section 2). Also, we note that although in reality job arrivals may also be more bursty and service times are not necessarily always exponential, such assumptions of Poisson arrivals and exponential service have been extensively used in the literature for reasons of mathematical tractability, while they can provide an adequate first approximation of real systems, see e.g. [9] and the references therein.

The main contributions of this paper are summarized as follows.

- A novel CTMC model is developed that captures the dynamics of cluster-based systems.
- The performance analysis of cluster-based systems is possible for both homogeneous and heterogeneous workloads.
- The impact of several parameters including the job size distribution on the performance measures of a cluster-based system is studied. The model allows to easily integrate any job size distribution.
- Through the model engineers can quickly gain insight into the expected performance when designing or managing a cloud service center and building service level agreements with customers.

The rest of the paper is organized as follows. In Section 2, we review some of the related work. The CTMC model of the system under study is presented in detail in Section 3. In Section 4, the performance measures are defined and in Section 5, an extensive analysis is conducted with numerical results. In Section 6, we conclude.

2 Related work

In this section, we review work with related analytical models. Few publications [8, 10–13] consider a job as consisting of tasks to be executed at the same time in parallel on different servers. The papers [8, 10, 11] consider a cluster system with two servers that may work either independently or in parallel. The service time is assumed to have an exponential distribution in [8, 10] and a general distribution in [11]. In [10, 11] Poisson arrivals are considered, the queue is unbounded and closed-form expressions for various performance measures of interest are obtained. In [8], the cluster system is modelled as an $MAP/M/2$ system with a Markovian arrival process and a steady-state analysis is conducted. Stability conditions are obtained in [12] for a multiserver system where a job requires a random number of servers to start execution. The paper [13] considers a cluster-based system where the authors develop a general mathematical model to analyze the cluster performance. In this paper, it is considered that the cluster is decomposed into disjoint partitions that execute the same collection of applications and the performance of a partition is reduced to the investigation of the performance of a single node. All the above models either consider a small number of servers in a cluster or decompose a cluster into disjoint partitions where each partition is analyzed separately. All these models cannot be applied to clusters with a larger number of servers including cloud computing centers where no cluster partitioning is performed.

The past decade a large number of publications have investigated the performance of the cloud [14–21]. In [14], the authors propose a classical $M/G/m/K$ queueing model to analyze the performance of a cloud server farm. The performance measures include the blocking probability and the mean number of tasks in the system. In [15], the server farm is represented by an $M/M/m$ system with infinite queue and the probability density function of the waiting time is derived. In [16], an $M/M/m+D$ queueing system is proposed to represent the servers in the cloud. In this system, the jobs are impatient and when a maximum tolerable waiting time D is reached, a job is assigned to a temporary rented server. In this paper, the authors study the profit maximization problem. In [17], the author conducts a performance analysis using stochastic reward nets. Several performance measures are evaluated including the utilization and waiting time. In this work, three variations of the arrival process are considered: a Poisson, a Markov-modulated Poisson (MMPP) and a bursty arrival process. Open Jackson networks are used to model the cloud in [18]. The model consists of a combination of $M/M/1$ and $M/M/m$ queues in sequence. The main performance measure is the total response time. In [19], the cloud is modeled as a finite queue with a load balancing node and m individual servers. The service times on the node and servers are exponentially distributed and arrivals are considered to be according to a Poisson process. In [20], the cloud is represented as an $G/G/m$-like model with finite queue. The authors present an efficient approximate solution to this system. The performance measures include the response time and blocking probability. In [21], an open queueing network is employed to model this cloud where a balancing node is modeled as an $M/M/1/K$ queue and each server (physical machine) is represented as an $M/M/m$ queue, where m is the number of virtual machines running on top of the physical machines. A disadvantage of all the above papers is that they assume that a job runs on exactly one server, i.e., no parallel execution of job tasks is considered.

Parallel processing of a job is considered in [6, 9]. These papers aim to compute the minimum number of resources needed to satisfy SLA (service-level-agreement) requirements. A classical $M/G/1/K$ system is proposed to model the cloud cluster and parallel processing is considered by simply dividing the job execution time of a cluster job (as if it is executed by one server) by the number of parallel servers. In [7], a Deadline Aware Scheduling Scheme is proposed and analyzed through a discrete-time Markov chain model. This model also considers a job to consist of tasks to be executed in parallel. However, the number of idle and busy servers is not captured and the authors assume that when a job arrives at the head of the line, it will be served with probability p that is considered beyond the scope of the paper. In reality, the job will be served if the number of required servers is available and will have to wait otherwise.

Recently, a multiserver queue where a job can simultaneously occupy multiple servers is tackled in [22–25]. The transient analysis of a two-server job queue is conducted in [22]. This paper studies a system with two identical servers, infinite buffer capacity and FIFO scheduling discipline, where the stability detection problem is solved. In [23], the authors propose a three-level modeling approach to study a multiserver job system as a representation of today's data centers. An explicit form of stability condition is obtained using matrix analytic methods for the case when the number of servers is two. In [24, 25], the multiserver job queuing model is defined, where multiserver jobs arrive with rate λ, the servers are homogeneous and the service discipline is FIFO. It has been indicated that almost nothing is known about the performance of such system in [24], even assuming Poisson arrivals and exponential job durations. The analysis of the model with more than two servers is declared as an open problem in [25], even when all jobs have the same exponentially distributed service durations. This paper aims exactly to address this open problem.

3 CTMC model of the system under study

We consider a cluster-based service system as illustrated in Figure 1. The system consists of a queue, a job tracker node and a cluster of N servers. We assume that jobs arrive to the system according to a Poisson process with arrival rate λ. Jobs arrive one at a time into the queue if the job tracker node is occupied or directly to the job tracker if both the queue and the job tracker node are empty. The queue has a finite storage capacity C and an arriving job is blocked and lost if the queue is full upon arrival.

Jobs are first processed by the job tracker node in FIFO order. A job consists of tasks for parallel execution on different servers of the cluster, one task per server. We define the job size as an integer-valued random variable that indicates the number of tasks of the job. The job size then also corresponds to the number of available servers that are required for the job to start execution on the cluster. We assume that the job sizes of the jobs entering the job tracker are independent and identically distributed (i.i.d.) with common probability mass function

$$\text{Prob[job size} = n] = p_n\,, \quad \text{for } 1 \le n \le T\,, \quad (1)$$

where T denotes the maximum job size, with $T \le N$. We do not specify specific values for these probabilities p_n for now. Specific example job size distributions will be considered when deriving numerical results later in the paper, as listed in Section 5.1.

When the required number of servers for a job is available, the job tracker then maps the job currently at the tracker into the cluster. This job mapping is assumed to require an exponentially distributed service time with mean value $1/\mu_1$, where μ_1 will be referred to as the service rate of a job at the job tracker. Also, we assume that the service time of a job of size n at the cluster is exponentially distributed with mean value $1/\mu_{2,n}$, i.e., $\mu_{2,n}$ is the service rate of a job of size n at the service cluster. Jobs leave the cluster one by one and as a result the servers occupied by the exiting job are released and become available again. The parameters of the system model are summarized in Table 1.

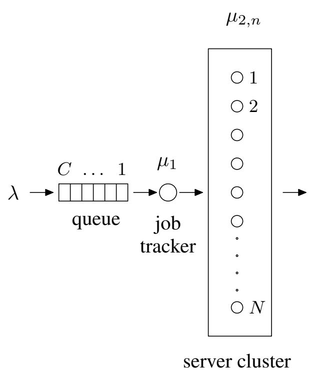

Fig. 1 Queueing model of the studied cluster-based system

Table 1 Model parameters

λ	Job arrival rate
μ_1	Service rate of a job at the job tracker
$\mu_{2,n}$	Service rate of a job of size n at the cluster
N	Number of servers in the cluster
C	Maximum capacity of the queue
T	Maximum job size

To investigate the behavior of the considered system, we now create a CTMC model. To this end, we let system state x correspond to the set $x = (x_1, x_2, V)$, where x_1 is the number of jobs waiting in the queue, x_2 corresponds to the state of the job tracker node and V is a vector that represents the state of the server cluster. More specifically, x_2 is the size of the job in service at the job tracker node, if any, where the job size corresponds to the number of servers needed for this job to start execution at the cluster. When there is no job at the job tracker node, we define $x_2 = 0$. The vector V is an ordered vector that describes all the jobs currently in execution at the cluster through the number of servers each of these jobs occupies in increasing order of the job sizes. When there are no jobs at the cluster, we define $V = (0)$. Clearly, this means that V is a variable-length vector and its length, denoted by ℓ_V, indicates the total number of jobs at the cluster if $V \neq (0)$. For example, $V = (2, 2, 3)$ means that $\ell_V = 3$ jobs are currently getting service at the cluster, two jobs occupy 2 servers each and one job occupies 3 servers. In general, the length ℓ_V of the vector V is thus between 1 and N such that the sum of all elements V_i, $1 \le i \le \ell_V$ of the vector V equals the total number of occupied servers in the cluster and thus is less than or equal to N. For later use, for $V \neq (0)$, we also define $m(n, V)$ as the so-called multiplicity of n in the vector V, i.e., the number of times the job size n occurs in the vector V, which also indicates the number of jobs of size n that are currently being served at the cluster.

The next step in our analysis is now to determine the transition rates $q_{x,y}$ of the CTMC from a system state x to another state y ($x \neq y$). To do so, for each system state $x = (x_1, x_2, V)$ we consider all possible events that cause a transition out of state x to another state y ($x \neq y$) and determine the corresponding rates. It turns out that we need to distinguish 5 different transition cases, as follows.

- First, if in the current system state $x = (x_1, x_2, V)$ both the queue and the job tracker node are empty (i.e., if $x_1 = x_2 = 0$), an arriving job to the system will directly enter the job tracker node and according to the job size distribution (see Equation (1)) the size of this job equals n with probability p_n, $1 \leq n \leq T$. Consequently, with rate $\lambda\, p_n$ we get a transition from state $x = (0, 0, V)$ to state $y = (0, n, V)$ and therefore

$$q_{x,y} = \lambda\, p_n\,, \quad \text{if } y = (0, n, V),\ x_1 = x_2 = 0\,.$$

  The total number of possible state transitions from a given state $x = (0, 0, V)$ due to a job arrival clearly equals the number of possible job sizes T in this case.
- Secondly, if a job arrives to the system (with rate λ) and in the current system state the job tracker is not idle and the queue is not completely full (i.e., if $x_2 > 0$ and $x_1 < C$), the arriving job joins the queue. The corresponding transition equation is then

$$q_{x,y} = \lambda\,, \\ \qquad \text{if } y = (x_1 + 1, x_2, V),\ x_1 < C,\ x_2 > 0\,.$$

  Note that in case $x_2 > 0$ and $x_1 = C$, an arriving job is blocked and the system state simply remains unchanged, so we do not need to consider this case.
- Next, if in the current system state $x = (x_1, x_2, V)$ we have that $V \neq (0)$ and $m(n, V) > 0$, a job of size n finishes execution at the cluster with rate $m(n, V)\, \mu_{2,n}$. Since n servers in the cluster are released upon the departure of the job of size n, we then get a transition to state $y = (x_1, x_2, V - \{n\})$. For example, for given values of x_1 and x_2, if $V = (2, 2, 3)$, then a transition is possible to state $(x_1, x_2, V - \{2\}) = (x_1, x_2, (2, 3))$ with rate $2\, \mu_{2,2}$ or to state $(x_1, x_2, V - \{3\}) = (x_1, x_2, (2, 2))$ with rate $\mu_{2,3}$ respectively. In summary, we thus have

$$q_{x,y} = m(n, V)\, \mu_{2,n}\,, \\ \qquad \text{if } y = (x_1, x_2, V - \{n\}),\ V \neq (0), \\ \qquad m(n, V) > 0\,.$$

  The total number of possible transitions from a given state $x = (x_1, x_2, V)$ with $V \neq (0)$ due to the end of a job execution clearly equals the number of different job sizes currently in service at the cluster, i.e., the number of different job sizes n ($1 \leq n \leq T$) for which $m(n, V) > 0$.
- The fourth transition case corresponds to transitions when the job tracker is not idle (i.e., $x_2 > 0$) and the job currently in the job tracker, which is then of size x_2 ($1 \leq x_2 \leq T$), finishes service at the job tracker node and moves to the cluster, while the queue is currently nonempty (i.e., while $x_1 > 0$). Such transitions are only possible if there are at least x_2 servers available at the cluster, i.e., if in the current system state $x = (x_1, x_2, V)$ we have that $\sum_{i=1}^{\ell_V} V_i + x_2 \leq N$. In this case, since the queue is nonempty, a new job enters the job tracker node. Like for the first transition case, this new entering job at the job tracker is of size n with probability p_n. All these observations lead to the following transition equation:

$$q_{x,y} = \mu_1\, p_n\,, \\ \qquad \text{if } y = (x_1 - 1, n, V + \{x_2\}),\ x_1 > 0, \\ \qquad x_2 > 0,\ \sum_{i=1}^{\ell_V} V_i + x_2 \leq N\,.$$

  Again, the total number of possible state transitions from state x equals the number of possible job sizes T in this case.
- Finally, if in the current system state we have an empty queue and the job tracker is not idle (i.e., $x_1 = 0$, $x_2 > 0$) and the job of size x_2 ($1 \leq x_2 \leq T$) in the tracker moves to the cluster, we get a transition to state $y = (0, 0, V + \{x_2\})$. This transition occurs with rate μ_1. Like for the

previous case, this transition is only possible if at least x_2 servers are available at the cluster, and therefore

$$q_{x,y} = \mu_1 \,, \quad \text{if } y = (0, 0, V + \{x_2\}),\ x_1 = 0,\ x_2 > 0,\ \sum_{i=1}^{\ell_V} V_i + x_2 \leq N \,.$$

These transition rates between states are summarized in Table 2. For all other combinations of system states x and y, with $x \neq y$, the transition rate $q_{x,y}$ from state x to state y equals 0.

The infinitesimal generator or transition rate matrix Q of the CTMC is then completely determined based on the property that the diagonal elements $q_{x,x}$ of the (square) matrix Q are such that the row sums of Q are equal to zero, while the non-diagonal elements $q_{x,y}$ $(x \neq y)$ of Q equal the above-derived transition rates.

Table 2 Transition rates $q_{x,y}$ from state $x = (x_1, x_2, V)$ to state y

State y	**Transition rate**	**Condition**
$(0, n, V)$	$\lambda\, p_n$	$x_1 = x_2 = 0$
$(x_1 + 1, x_2, V)$	λ	$x_1 < C$, $x_2 > 0$
$(x_1, x_2, V - \{n\})$	$m(n, V)\, \mu_{2,n}$	$V \neq (0)$, $m(n, V) > 0$
$(x_1 - 1, n, V + \{x_2\})$	$\mu_1\, p_n$	$x_1 > 0$, $x_2 > 0$, $\sum_{i=1}^{\ell_V} V_i + x_2 \leq N$
$(0, 0, V + \{x_2\})$	μ_1	$x_1 = 0$, $x_2 > 0$, $\sum_{i=1}^{\ell_V} V_i + x_2 \leq N$

Next, we show that the infinitesimal generator Q of the CTMC has a Quasi Birth Death (QBD) structure. Within a QBD matrix, the states are arranged into QBD levels and QBD phases resulting in a block-tridiagonal matrix (see Equation (2) below). A distinguishing feature of this structure is that a transition from a state x to a state y can happen only between adjacent levels or within the same level. We define the level of our QBD as the number of jobs x_1 $(0 \leq x_1 \leq C)$ currently waiting in the queue and we define the phase as a vector (x_2, V) representing the current state of the job tracker x_2 $(0 \leq x_2 \leq T)$ and the state of the cluster described by vector V. In view of the transition rates of Table 2, it can be easily seen that the infinitesimal generator Q has indeed a QBD block structure, where the transition cases one, three and five describe transitions within the same level and the transition cases two and four describe transitions that increase or decrease the level by one respectively. More specifically, the structure of the matrix Q is as follows:

$$Q = \begin{bmatrix} A_{10} & A_{00} & & & & & \\ A_{21} & A_{11} & A_{01} & & & & \\ & A_{22} & A_{11} & A_{01} & & & \\ & & A_{22} & A_{11} & A_{01} & & \\ & & & \cdots & & & \\ & & & & A_{22} & A_{11} & A_{01} \\ & & & & & A_{22} & A_{1C} \end{bmatrix}, \qquad (2)$$

where A_{ij} denotes a submatrix of level j. Level 0 represents the case when the queue is empty. In level 1, there is only one job at the queue, etc. Submatrix A_{0j} contains transitions that result in an increase of the level by one, i.e., a forward submatrix. Likewise, A_{1j} and A_{2j} are the submatrices that describe the transitions within the same level and transitions that result in a decrease of the level by one, i.e., a backward submatrix, respectively. Note that the size of submatrix A_{10} is larger than the size of A_{11} since it includes additional states for level 0, namely the states related to the case when the job tracker node is empty, i.e., when the job tracker state has value 0. Starting from level 1, the job tracker cannot be empty because when a job finishes service in the job tracker node, the head of the line job occupies instantly the job tracker decreasing the level by one. It is also important to note that starting from level 2 until level $C-1$ all the submatrices A_{0j}, A_{1j} or A_{2j} are identical, respectively. This is because the queue length does not influence the transitions within these submatrices.

We now describe the order of the possible phases (x_2, V) within a given level of matrix Q. Firstly, the possible phases are ordered in increasing order of the size x_2 of the job at the job tracker node. For a given value of x_2, the possible states of the vector V are then arranged first in increasing order of the length ℓ_V and for a given length $\ell_V = k \geq 1$ in lexicographical order, as follows: $(1,1,\ldots,1)$, $(1,1,\ldots,2)$, $\ldots$, $(1,1,\ldots,r)$, $(1,1,\ldots,2,2)$, $(1,1,\ldots,2,3)$, $\ldots$, $(1,1,\ldots,2,r)$, $(1,1,\ldots,3,3)$, etc., where the notation r in this series stands for

$$r = \min(N - \sum_{i=1}^{k-1} V_i, T)\,. \tag{3}$$

For example, if $\ell_V = 3$ and $N = T = 6$, the possible states of V are ordered as $(1,1,1)$, $(1,1,2)$, $(1,1,3)$, $(1,1,4)$, $(1,2,2)$, $(1,2,3)$, $(2,2,2)$. The order of the states just described assumes that x_2 can have any value between 1 and T. However, the same logic for the order of the states within the phase applies when x_2 has specific ascending values j_1, j_2, j_3, etc.

Let $\mathcal{S}$ denote the set of all possible states $x = (x_1, x_2, V)$ of the CTMC. We define P_x as the steady-state probability that the system is in state $x \in \mathcal{S}$. We then include the probabilities P_x for all $x \in \mathcal{S}$ in a row vector P, where the states x are also ordered in increasing order of the level x_1, for a given value of x_1 in increasing order of x_2 and then for a given value of x_2 as explained above. This vector P of steady-state probabilities of the CTMC can then be computed by solving the set of balance equations of the CTMC together with the normalization condition:

$$P\,Q = 0\,, \quad \sum_{x\in\mathcal{S}} P_x = 1\,. \tag{4}$$

The QBD structure of the CTMC allows us to efficiently compute the steady-state probabilities P_x. Several numerical approaches are available in the literature, see e.g. [26, 27]. Specifically, to solve the equation $P\,Q = 0$, we use here the Gaussian elimination technique and the concepts of level by level Schur complementation and censored Markov chains applied to a block-structured QBD. For an illustration of the main steps of the computation technique, we refer the reader e.g. to [28] (Section 4.1). Based on the vector P, we derive several performance measures in the next section.

4 Performance measures

We are interested in computing performance measures such as the average delay $E[D]$ of the jobs in the queue, the average utilization $E[U]$ of the cluster servers, the blocking probability B due to a full queue and the mean number of jobs $E[J]$ currently being served at the cluster.

Let $\mathcal{S}_j$ be the set of all states of the CTMC at level j, then the average queue length $E[q]$ is computed as

$$E[q] = \sum_{j=1}^{C} j \sum_{x\in\mathcal{S}_j} P_x\,, \tag{5}$$

where C is the maximum queue capacity. The blocking probability B is the probability that an arriving job finds the queue full and gets lost. This is obtained by the formula

$$B = \sum_{x\in\mathcal{S}_C} P_x\,, \tag{6}$$

where $\mathcal{S}_C$ is the set of all states at level C. Then the average delay experienced by jobs in the queue can be computed as

$$E[D] = \frac{E[q]}{\lambda\,(1-B)}\,. \tag{7}$$

Let $S_{V(x)}$ be the sum of all elements V_i in vector V of state x. The average utilization $E[U]$ of the cluster servers is then computed as

$$E[U] = \sum_{x\in\mathcal{S}} S_{V(x)}\,P_x\,, \tag{8}$$

where as before $\mathcal{S}$ is the set of all system states. Finally, let $N_{V(x)}$ be the number of non-zero elements of the vector V of state x. The average number of jobs served in parallel at the server cluster is calculated as

$$E[J] = \sum_{x\in\mathcal{S}} N_{V(x)}\,P_x\,. \tag{9}$$

5 Numerical results

In this section, we investigate the performance of cluster-based service systems through a number of illustrative numerical examples.

5.1 Overview of the considered scenarios

We mostly, except for Figure 11, consider scenarios where the cluster has a number of servers N of

16 or less. The results then seem to be applicable for small clusters and micro-clouds that typically contain only a few servers, e.g. 10 servers [29]. Figure 11 considers a larger cluster size N above 1000.

For the service times of jobs at the cluster, we consider the service rate to be the same for all job sizes, i.e., $\mu_{2,n} = \mu_2$ for all $1 \leq n \leq T$, in Figures 2–10, while a heterogeneous workload with service rates dependent on the job size is considered in Figure 11.

With respect to the job sizes, we assume that the size of a job can take any integer value between 1 and N in Figures 2–4 and Figures 7–10, i.e., we set $T = N$ in these figures, and we consider the case of values T smaller than N in Figures 5–6.

Our model allows arbitrary values for the probabilities p_n defined in Equation (1). In the examples, several specific distributions are now considered for the job size.

- Uniform. The probability of having a job size n is given by $p_n = 1/T$, for any integer n with $1 \leq n \leq T$.
- (Bounded) normal. The probability p_n is proportional to $\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(n-\eta)^2}{2\,\sigma^2}\right)$, for all integers n in the range $1 \leq n \leq T$, with $\eta = (T+1)/2$, where the proportionality constant is determined such that the resulting probability mass function is normalized, i.e., $\sum_{n=1}^{T} p_n = 1$.
- Harmonic. The probability p_n is proportional to $1/n^{1.5}$, for integer job sizes n with $1 \leq n \leq T$. This distribution gives a higher probability for jobs that require a smaller number of servers. All the distributions mentioned above were used to describe job sizes for parallel computing [30–32].
- Probability distribution derived from real cloud traces. While in a real cloud a job can have any number of tasks between 1 and tens of thousands, it has been found that some job sizes are encountered more often and comprise a major portion of the workload. Taking this fact into consideration, we assume in this case that x_2 has few possible values j_1, j_2, j_3, $\ldots$ with probabilities p_{j_1}, p_{j_2}, p_{j_3}, $\ldots$ respectively. For example, it has been reported [5] that most jobs have a small number of tasks and these jobs are short, while there exist few long jobs with many tasks. The job length and job size are observed to have a bi-modal distribution [5, 33, 34].

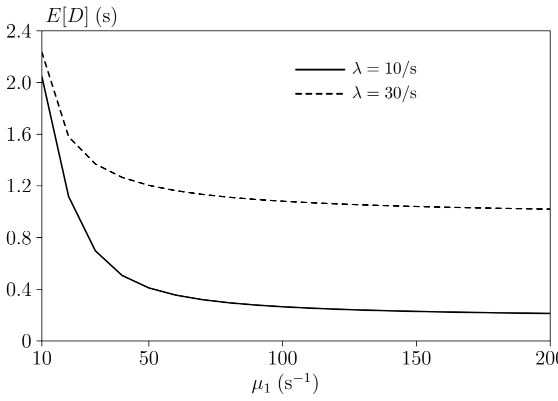

Fig. 2 Average delay of jobs $E[D]$ versus μ_1, for $\mu_2 = 10/s$, $N = T = 16$, $C = 15$, uniform job sizes and different values of λ

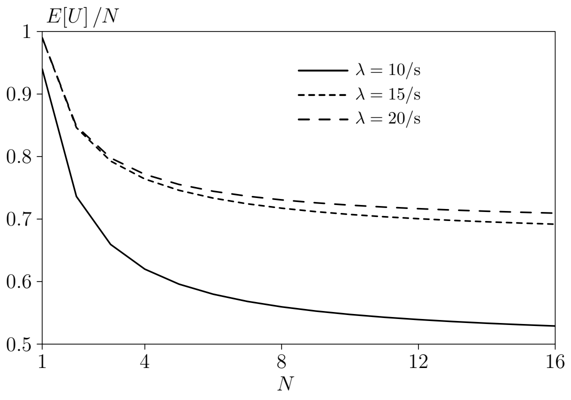

Fig. 3 Utilization of the cluster servers $E[U]$ (normalized by N) versus N, for $\mu_1 = 1000/s$, $\mu_2 = 10/s$, $C = 15$, $T = N$, uniform job sizes and different values of λ

We start in the Figures 2–6 with investigating the impact of different parameters on the performance measures in case the job size is uniformly distributed. Next, in the Figures 7–9, we also explore the effect of other distributions of the job size, namely the (bounded) normal and harmonic distributions. As a special case, in Figure 10, we study the influence of the variance of the job size distribution when the job size distribution is normal. Finally, in Figure 11, we apply the model to the case when the probability distribution of the job size is derived from real traces.

5.2 Discussion of the results

First, we explore the impact of the service rate μ_1 on the average delay of jobs in the queue. In Figure 2, we show the average delay of the jobs $E[D]$ versus μ_1 for different values of the arrival

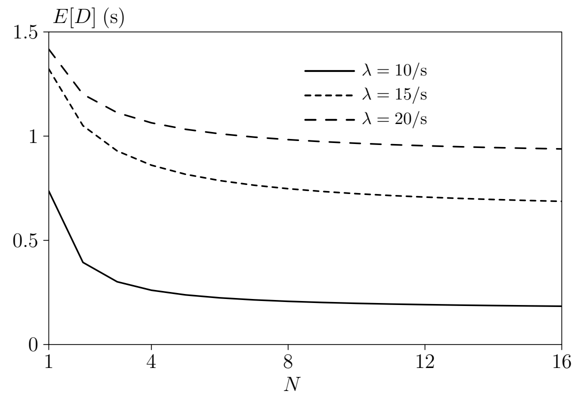

Fig. 4 Average delay of jobs $E[D]$ versus N, for $\mu_1 = 1000/s$, $\mu_2 = 10/s$, $C = 15$, $T = N$, uniform job sizes and different values of λ

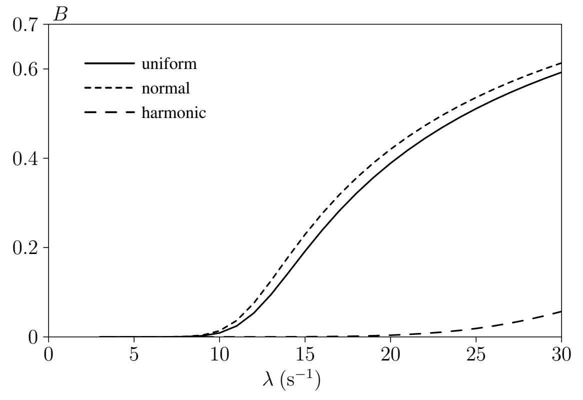

Fig. 7 Blocking probability of jobs B versus λ, for $\mu_1 = 100/s$, $\mu_2 = 10/s$, $N = T = 16$, $C = 15$ and different job size distributions

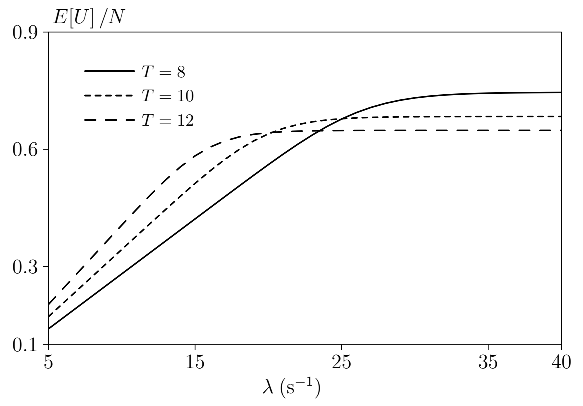

Fig. 5 Utilization of the cluster servers $E[U]$ (normalized by N) versus λ, for $\mu_1 = 100/s$, $\mu_2 = 10/s$, $N = 16$, $C = 15$, uniform job sizes and different values of T

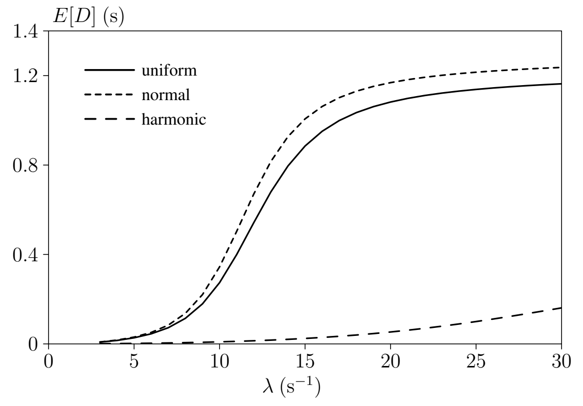

Fig. 8 Average delay of jobs $E[D]$ versus λ, for $\mu_1 = 100/s$, $\mu_2 = 10/s$, $N = T = 16$, $C = 15$ and different job size distributions

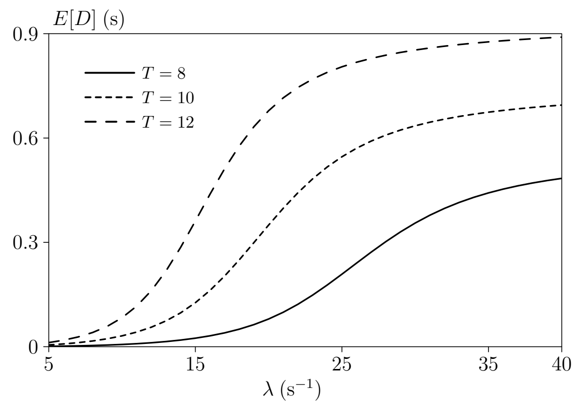

Fig. 6 Average delay of jobs $E[D]$ versus λ, for $\mu_1 = 100/s$, $\mu_2 = 10/s$, $N = 16$, $C = 15$, uniform job sizes and different values of T

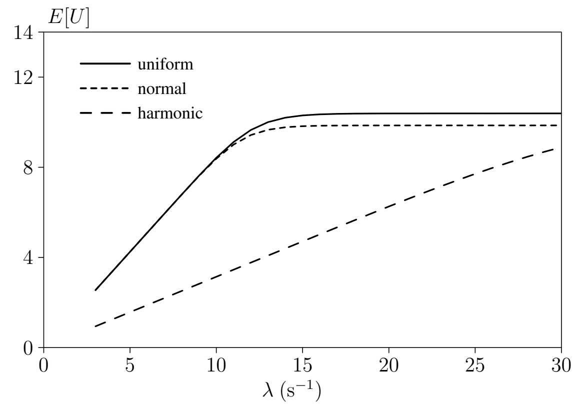

Fig. 9 Utilization of the cluster servers $E[U]$ versus λ, for $\mu_1 = 100/s$, $\mu_2 = 10/s$, $N = T = 16$, $C = 15$ and different job size distributions

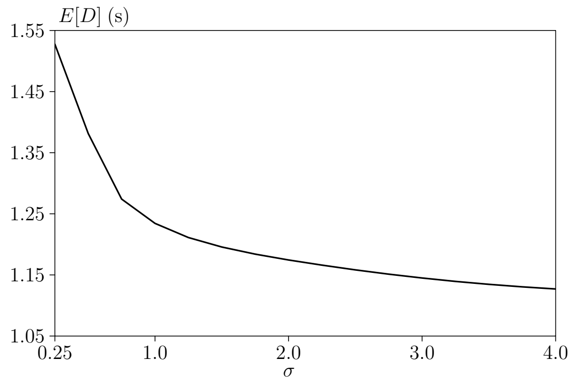

Fig. 10 Average delay of jobs $E[D]$ versus σ, for $\lambda = 30/\text{s}$, $\mu_1 = 200/\text{s}$, $\mu_2 = 10/\text{s}$, $N = T = 15$, $C = 15$ and normal job sizes

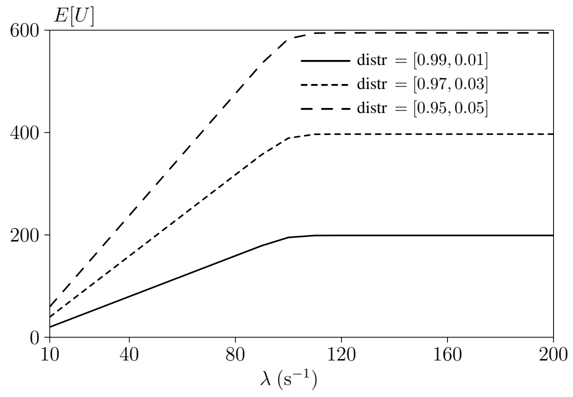

Fig. 11 Utilization of the cluster servers $E[U]$ versus λ, for $\lambda = 30/\text{s}$, $\mu_1 = 100/\text{s}$, bi-modal job sizes with $j_1 = 10$ and $j_2 = 100$, $\mu_{2,j_1} = 10/\text{s}$, $\mu_{2,j_2} = 1/\text{s}$, $N = 1500$, $C = 50$ and different job size distributions

rate λ and a uniform job size distribution. It can be clearly seen that for increasing μ_1 the average delay of the jobs decreases, as expected. This decrease is sharp for smaller values of μ_1 and a further increase of μ_1 has a small effect on the average delay. Similarly, it has been observed that increasing μ_1 beyond a certain value for a specific parameter set has a small effect on the rest of the studied performance measures as well. Based on the working parameters of the system, it is clear that a specific value of μ_1 is needed (and thus enough resources are required) to prevent the job tracker node of becoming the bottleneck of the system.

Next, we investigate the effect of the cluster size N on the average delay of jobs and on the cluster utilization. We choose μ_1 much larger than μ_2 to minimize the effect of the job tracker on the performance of the cluster. In Figure 3, the utilization of the cluster (normalized by N) is plotted versus the cluster size N for uniform job sizes with $T = N$ and different values of λ. We observe that an increase of the cluster size negatively influences the cluster utilization and more resources are wasted. The reason is that when a job demands a larger number of servers than currently available, the job temporarily prevents the smaller ones behind it to receive service because the service discipline is FIFO. Also, it should be noted that when $N = 1$, the model becomes very close to an $M/M/1/C$ queueing model. In Figure 4, the average delay of the jobs is presented versus the cluster size for different values of λ. We notice here that for increasing N, the average delay decreases. This decrease is due to the increase of the number of jobs that are served in parallel. In Figures 3–4, an increase of the arrival rate λ leads to an increase of both the utilization and the average job delay, as expected. However, it should be noted that it seems that for a specific parameter set, an increase of the cluster size beyond a certain value does almost not affect the cluster utilization nor the average delay of jobs. Another interesting fact is that the maximum value of the average number of jobs served in parallel over all scenarios of Figures 3–4 does not exceed 1.34, which is somewhat counter intuitive when the cluster size $N = 16$.

Now, we study the case when the maximum job size $T < N$, i.e., when the cluster size is larger than the requirement of the largest job. The job size distribution is still uniform. In Figure 5, the utilization of the cluster $E[U]$ (normalized by N) is shown versus λ for different values of the maximum job size T. We observe that for smaller values of λ, the utilization is larger for a larger maximum job size T. For larger values of λ, however, an increase of the maximum job size will decrease the utilization because the number of jobs that can run in parallel decreases. In particular, when the maximum job size is only 8, we have that in this case at least 2 jobs of maximum size can always still be executed in parallel, which is not the case for $T > 8$. In Figure 6, the average delay of jobs $E[D]$ is plotted versus λ for different values of the maximum job size T. Clearly, when T increases, more servers may be involved in the execution of a job, so fewer jobs can be processed in parallel, which results in an increase of the number of jobs in the queue and the delay of jobs in the queue.

In a next set of examples, we study the effect of various job size distributions on the performance measures under the condition that the maximum job size $T = N$. For the (bounded) normal job sizes, we have chosen $\sigma = \eta/4$. In Figure 7, the blocking probability B is shown versus λ for different job size distributions. When the arrival rate λ is small, the queue is lightly occupied and the blocking probability is small, as expected. As λ increases, the blocking probability rises and the effect of the job size distribution on the blocking probability can be clearly noticed. It can be seen that for given T, when the job size has a harmonic distribution, the blocking probability is much lower than for the rest of the investigated distributions. This is explained by the fact that in case of harmonic job sizes more smaller jobs arrive than larger ones, so for given T, the average job size is smaller and more jobs can be served in parallel. We also notice that for given T, the system has a similar behavior for uniform and normal job sizes, where the uniform distribution slightly outperforms the case of (bounded) normal job sizes.

Figure 8 shows the average delay of jobs $E[D]$ versus λ, again for different job size distributions. As expected, for given T, also the average delay is much lower for the case of harmonic job sizes. Furthermore, we notice that for larger λ, the average delay for both normal and uniform job sizes increases more slowly as the queue is getting almost full.

In Figure 9, the utilization of the cluster servers is presented versus λ. For smaller λ, the utilization increases almost linearly with increasing λ. A further increase of λ beyond a certain point has almost no effect on the utilization. In summary, Figures 7–9 show that for a given maximum job size T, the cluster-based system behaves the best for the harmonic job size distribution. In this case, both the blocking probability and the average delay of jobs are the lowest for the same λ. The system performs similarly for both normal and uniform job sizes and the uniform job sizes are slightly better in terms of blocking probability and average delay.

Figure 10 illustrates the effect of the standard deviation of the job sizes on the average delay of jobs in case of (bounded) normal job sizes. In particular, Figure 10 shows the delay $E[D]$ versus the parameter σ of the (bounded) normal job size distribution. We see that the average delay of jobs decreases for increasing values of σ. This is explained by the fact that when the standard deviation is small, most jobs are of similar size close to the mean job size and the availability of a number of idle servers smaller than the mean job size is often not made use of. When the standard deviation increases, the utilization of the cluster servers increases, more jobs are served in parallel and the delay decreases.

In a last example, we investigate the performance of a cluster-based system with $N = 1500$ servers, when the workload is approximated by two different job types as follows. Based on a real workload measurement [5], it is indicated that the majority of the jobs require either less than 10 servers or more than 100 servers each. Also, the running time of jobs that require many servers is significantly longer than the length of jobs with a small number of servers. Based on the above, we assume $j_1 = 10$ and $j_2 = 100$ as possible values for the job size for the first and second job type respectively. We also assume that many-server jobs are 10 times longer than few-server jobs. In particular, we assume $\mu_{2,j_1} = 10/\mathrm{s}$ and $\mu_{2,j_2} = 1/\mathrm{s}$ as values for the service rate at the cluster for the first and second job type respectively.

Under such common bi-modal workload, even for $N = 1500$ servers, the total number of jobs in service at the cluster is only 150 or less such that the state space $\mathcal{S}$ of our CTMC and especially the number of possible states of the vector V is not that large and the performance characteristics can be computed in a reasonable time.

Figure 11 shows the utilization of the servers versus the arrival rate λ for a heterogeneous workload with different bi-modal probability distributions of the job size. It can be seen that a small increase in the percentage of long jobs significantly increases the utilization of the servers. This is interesting because in this case the blocking probability B and the mean queue length $E[q]$ are almost the same for a given arrival rate λ. Consequently, the utilization of the servers increases significantly without affecting the average delay of jobs in the queue.

6 Conclusion

In this paper, we have thoroughly analyzed the impact of different parameters on the performance measures of a cluster-based service system. We have proposed a new analytical model based on Markov chain theory that explicitly takes the parallel execution of job tasks into consideration and as such captures the typical dynamics of a cluster-based service system. The model can be used to study clusters of different sizes. We have shown that our model is flexible, efficient and can also be applied to analyze clouds with thousands of servers under a typical heterogeneous workload with bi-modal job sizes (see Figure 11).

Through our analysis, key performance measures of a cluster-based service system such as the job blocking probability, the average job delay in the queue or the average utilization of the cluster servers can be estimated. As has been demonstrated by the given numerical examples, through our model engineers can quickly gain insight into the expected performance when designing or managing a cloud service center and building service level agreements with customers. Such quantitative performance analysis of cloud service centers is extremely important since institutions rely on such systems more and more. Our model can then be used e.g. to predict the number of cluster servers needed to satisfy a given maximum job delay, while e.g. keeping the server utilization above a given threshold. This is particularly useful in the design phase of server clusters. Using the model, such performance-related issues can be handled efficiently, avoiding the need to first build up a complete server setup and then perform time-consuming measurements on it.

A future research direction is the extension of the model to other service disciplines than FIFO. Server clusters often employ priority scheduling disciplines, where higher priority jobs can evict lower priority jobs to meet their service level agreement [34]. We therefore intend to further extend our model to the case of preemptive priority scheduling with 2 priority classes of jobs. Such extension seems justified by the fact that the job length and the job size are typically observed to have a bi-modal distribution [34], where the few long jobs can evict lower priority jobs if their required number of idle servers is not available. Another potential direction for future work is to include the energy consumpton of the cluster in the model [35], where for each server different states could be considered: setup, running and down.

Declarations

- Funding: This research received no external funding.
- Competing interests: The authors have no relevant financial or non-financial interests to disclose.
- Availability of data and materials: The numerical results calculated in this study are presented in the figures in this paper.
- Authors' contributions: All authors contributed to the study conception and the development of the methodology. The numerical results were calculated by Osama Salameh and both authors contributed to the discussion of these results. The first draft of the manuscript was written by Osama Salameh. This draft was commented on and thoroughly revised by Sabine Wittevrongel. All authors read and approved the final manuscript.

References

[1] Valentini, G.L., Lassonde, W., Khan, S.U. et al.: An overview of energy efficiency techniques in cluster computing systems. Cluster Computing 16, 3–15 (2013). https://doi.org/10.1007/s10586-011-0171-x

[2] Buyya, R.: High Performance Cluster Computing: Architectures and Systems, Vol. 1. Prentice Hall (1999)

[3] Wang, L., von Laszewski, G., Younge, A., He, X., Kunze, M., Tao, J., Fu, C.: Cloud computing: a perspective study. New Generation Computing 28, 137–146 (2010). https://doi.org/10.1007/s00354-008-0081-5

[4] Mishra, A.K., Hellerstein, J.L., Cirne, W., Das, C.R.: Towards characterizing cloud backend workloads: insights from Google compute clusters. ACM SIGMETRICS Performance Evaluation Review 37, 34–41 (2010). https://doi.org/10.1145/1773394.1773400

[5] Boutaba, R., Cheng, L., Zhang, Q.: On cloud computational models and the heterogeneity challenge. Journal of Internet Services and Applications 3, 77–86 (2012). https://doi.org/10.1007/s13174-011-0054-7

[6] Salah, K.: A queueing model to achieve proper elasticity for cloud cluster jobs. Proceedings of the IEEE 6th International Conference on Cloud Computing, pp. 755–761. Santa Clara, USA (2013). https://doi.org/10.1109/CLOUD.2013.20

[7] Khabbaz, M., Assi, C.M.: Modelling and analysis of a novel deadline-aware scheduling scheme for cloud computing data centers. IEEE Transactions on Cloud Computing 6, 141–155 (2018). https://doi.org/10.1109/TCC.2015.2481429

[8] Chakravarthy, S.R., Karatza, H.D.: Two-server parallel system with pure space sharing and Markovian arrivals. Computers & Operations Research 40, 510–519 (2013). https://doi.org/10.1016/j.cor.2012.08.002

[9] Salah, K., Alcaraz Calero, J.M.: Achieving elasticity for cloud MapReduce jobs. Proceedings of the IEEE 2nd International Conference on Cloud Networking, CloudNet, pp. 195–199. San Francisco, USA (2013). https://doi.org/10.1109/CloudNet.2013.6710577

[10] Filippopoulos, D., Karatza, H.: An M/M/2 parallel system model with pure space sharing among rigid jobs. Mathematical and Computer Modelling 45, 491–530 (2007). https://doi.org/10.1016/j.mcm.2006.06.007

[11] Filippopoulos, D., Karatza, H.: A two-class parallel system with general service times of the parallel class. Journal of Computer and System Sciences 74, 942–964 (2008). https://doi.org/10.1016/j.jcss.2007.07.001

[12] Rumyantsev, A., Morozov, E.: Stability criterion of a multiserver model with simultaneous service. Annals of Operations Research 252, 29–39 (2017). https://doi.org/10.1007/s10479-015-1917-2

[13] Squillante, M.S., Zhang, Y., Sivasubramaniam, A., Gautam, N., Franke, H., Moreira, J.: Modeling and analysis of dynamic coscheduling in parallel and distributed environments. ACM SIGMETRICS Performance Evaluation Review 30, 43–54 (2002). https://doi.org/10.1145/511334.511341

[14] Khazaei, H., Mišić, J., Mišić, V.B.: Performance analysis of cloud computing centers using M/G/m/m+r queuing systems. IEEE Transactions on Parallel and Distributed Systems 23, 936–943 (2012). https://doi.org/10.1109/TPDS.2011.199

[15] Cao, J., Hwang, K., Li, K., Zomaya, A.Y.: Optimal multiserver configuration for profit maximization in cloud computing. IEEE Transactions on Parallel and Distributed Systems 24, 1087–1096 (2013). https://doi.org/10.1109/TPDS.2012.203

[16] Mei, J., Li, K., Ouyang, A., Li, K.: A profit maximization scheme with guaranteed quality of service in cloud computing. IEEE Transactions on Computers 64, 3064–3078 (2015). https://doi.org/10.1109/TC.2015.2401021

[17] Bruneo, D.: A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems. IEEE Transactions on Parallel and Distributed Systems 25, 560–569 (2014). https://doi.org/10.1109/TPDS.2013.67

[18] Vilaplana, J., Solsona, F., Teixidó, I., Mateo, J., Abella, F., Rius, J.: A queuing theory model for cloud computing. The Journal of Supercomputing 69, 492-–507 (2014). https://doi.org/10.1007/s11227-014-1177-y

[19] Salah, K., Elbadawi, K., Boutaba, R.: An analytical model for estimating cloud resources of elastic services. Journal of Network and Systems Management 24, 285—308 (2016). https://doi.org/10.1007/s10922-015-9352-x

[20] Atmaca, T., Begin, T., Brandwajn, A., Castel-Taleb, H.: Performance evaluation of cloud computing centers with

general arrivals and service. IEEE Transactions on Parallel and Distributed Systems 27, 2341–2348 (2016). https://doi.org/10.1109/TPDS.2015.2499749

[21] El Kafhali, S., Salah, K.: Modeling and analysis of performance and energy consumption in cloud data centers. Arabian Journal for Science and Engineering 43, 7789–7802 (2018). https://doi.org/10.1007/s13369-018-3196-0

[22] Razumchik, R., Rumyantsev, A.: Some ergodicity and truncation bounds for a small scale Markovian supercomputer model. Proceedings of the 36th ECMS International Conference on Modelling and Simulation, ECMS 2022, Communications of the ECMS 36, pp. 324–330. Ålesund, Norway (2022). https://doi.org/10.7148/2022-0324

[23] Rumyantsev, A., Basmadjian, R., Astafiev, S., Golovin, A.: Three-level modeling of a speed-scaling supercomputer. Annals of Operations Research (2022). https://doi.org/10.1007/s10479-022-04830-0

[24] Harchol-Balter, M.: The multiserver job queueing model. Queueing Systems 100, 201–203 (2022). https://doi.org/10.1007/s11134-022-09762-x

[25] Harchol-Balter, M.: Open problems in queueing theory inspired by datacenter computing. Queueing Systems 97, 3–37 (2021). https://doi.org/10.1007/s11134-020-09684-6

[26] Ye, J., Li, S.-Q.: Folding algorithm: a computational method for finite QBD processes with level-dependent transitions. IEEE Transactions on Communications 42, 625–639 (1994). https://doi.org/10.1109/TCOMM.1994.577090

[27] Elhafsi, E.H., Molle, M.: On the solution to QBD processes with finite state space. Stochastic Analysis and Applications 25, 763-–779 (2007). https://doi.org/10.1080/07362990701419946

[28] Salameh, O., De Turck, K., Bruneel, H., Blondia, C., Wittevrongel, S.: Analysis of secondary user performance in cognitive radio networks with reactive spectrum handoff. Telecommunication Systems 65, 539–550 (2017). https://doi.org/10.1007/s11235-016-0250-7

[29] Yazdanov, L., Gorbunov, M., Fetzer, C.: EHadoop: network I/O aware scheduler for elastic MapReduce cluster. Proceedings of the IEEE 8th International Conference on Cloud Computing, pp. 821–828. New York, USA (2015). https://doi.org/10.1109/CLOUD.2015.113

[30] Karatza, H.D.: Gang scheduling in a distributed system under processor failures and time-varying gang size. Proceedings of the 9th IEEE Workshop on Future Trends of Distributed Computing Systems, FTDCS 2003, pp. 330–336. San Juan, USA (2003). https://doi.org/10.1109/FTDCS.2003.1204355

[31] Aida, K.: Effect of job size characteristics on job scheduling performance. Proceedings of the 6th Workshop on Job Scheduling Strategies for Parallel Processing, JSSPP 2000, Lecture Notes in Computer Science 1911, pp. 1–17. Cancun, Mexico (2000). https://doi.org/10.1007/3-540-39997-6_1

[32] Feitelson, D.G.: Workload Modeling for Computer Systems Performance Evaluation. Cambridge University Press (2015)

[33] Chen, Y., Ganapathi, A.S., Griffith, R., Katz, R.H.: Towards understanding cloud performance tradeoffs using statistical workload analysis and replay. Technical Report No. UCB/EECS-2010-81, University of California, Berkeley (2010). http://www2.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-81.pdf

[34] Tirmazi, M., Barker, A., Deng, N., Haque, Md.E., Qin, Z.G., Hand, S., Harchol-Balter, M., Wilkes, J.: Borg: the next generation. Proceedings of the 15th European Conference on Computer Systems, EuroSys'20, pp. 1–14. Heraklion, Greece (2020). https://doi.org/10.1145/3342195.3387517

[35] Forshaw, M., McGough, A.S., Thomas, N.: HTC-Sim: a trace-driven simulation framework for energy consumption in high-throughput computing systems. Concurrency and Computation: Practice and Experience 28, 3260–3290 (2016). https://doi.org/10.1002/cpe.3804