Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 310 (2015) 49–63

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Asymptotic Behavior and Performance Constraints of Replication Policies

Davide Cerotti, Marco Gribaudo
Pietro Piazzolla, Giuseppe Serazzi[1]

Dip. di Elettronica, Informazione e Bioingegneria
Politecnico di Milano
Via Ponzio 34/5, 20133 Milano, Italy

Abstract

Spreading the workload among a pool of replicated servers is a technique typically adopted to reduce the response time and increase the throughput in complex systems. In several actual computing environments, virtual machines can be provisioned in a fast and convenient way, and the replication has assumed an important role for the efficient system management. However, in order to provide economically acceptable solutions, the number of replica should be limited to the minimum required to match the given performance goal. In this paper we propose a simple replication policy to match thresholds on the system response times. The analytical relationships that exist between the performance objective values of some metrics and the number of replica are derived. Analytical and experimental validations with single and multi-class workload are presented. Open and closed models, and NO-SQL database have been considered.

Keywords: Replication, Multiclass workload, Cloud computing and virtualization, Analytical techniques, Asymptotic techniques

1 Introduction

By definition, a cloud computing infrastructure consists of a large number of virtualized resources which are dynamically allocated to user requests. To cope with the high fluctuations of workloads and the heterogeneity of available systems, dynamic capacity provisioning is typically used. Replication is a popular technique adopted to meet the performance objectives, usually described through a set of values referred to as Service Level Agreements (SLAs). The replication technique essentially consists in the provision of a variable number of replica of virtual machines that share the workload to match the user requirements.
When one or more performance objectives are not met, penalties and economic losses occur on both sides of the users and the cloud providers. Therefore, it is of

[1] Email: [davide.cerotti,marco.gribaudo,pietro.piazzolla,giuseppe.serazzi]@polimi.it

http://dx.doi.org/10.1016/j.entcs.2014.12.012

paramount importance to minimize the cases of SLA violations.
Over-provisioning is one of the most simple replication techniques that is used, but, of course, it is very inefficient and expensive. To handle with the dynamic variations of the arriving traffic and of the load being processed, replication policies that automatically adapt their behavior to the dynamic variations of the workload are required.
Key features of a replication policy handling a workload with multi-time scale fluctuations, such as that of clouds, should be: to identify the minimum number of replica in order to achieve the objectives set by the SLAs, to minimize the SLA violations, to require minimum amount of computing resources, and to be robust with respect to all possible network failures. Typically, SLAs are formulated as a set of numerical values representing the thresholds of the correspondent metrics. Depending on the type of metric associated, the control on the respect of the threshold values may require very different levels of difficulty. For example, to control the response time or the utilization level of a resource is easier than to evaluate the mean number of requests in the resource.
In this paper, we build up on the results presented in [9]: in that work, the optimal number of replica necessary to serve a given transactional workload is determined for open models. In this paper we find analogous results for closed systems, and we exploit both the relations given in that work, and the ones newly presented here, to derive analytical relationships between the achieved response time as the number of replica tends asymptotically to infinity. The existence of these relationships enables the design of control policies on one or more metrics allowing at the same time to satisfy constraints on other metrics. For example, the control that the average response time of a system is less than 5 seconds implies that the response time of bottleneck resource is less than 3.5 seconds. So, to control if a threshold of a metric is not violated, we can monitor the value of another metric that is most convenient to measure. Through these relationships, it is possible to exert the control on the values of several metrics simply controlling only one of them.

This paper is organized as follows. In Section 2, the related work on model-analysis and optimization in cloud computing is presented. The response time metric and the corresponding number of replica to satisfy the constraints is analyzed in Section 3. The asymptotic behavior of the performance indexes with respect to the number of replica in either open and closed systems is investigated in Sections 4 and 5, respectively. Experimental validations are reported in Section 6. Section 7 concludes the paper.

2 Related work

The replication of resources has recently attracted new interests due to its connection with virtualization and cloud computing. In particular, the flexibility of provisioning Virtual Machines (VMs) has made both static and dynamic (i.e. using adaptive techniques) replication a simple and effective technique to achieve performance goals without the need of large over-provisioning. However, the use of cloud computing has increased unpredictability of the system, making model-based analysis much harder [8] and introducing further factors that must be considered. For instance, the different Virtual Machine Manager allocation policies have a strong impact on both CPU and I/O performance as evaluated in [13]. Several approaches have been proposed to account the effect of virtualization: in queuing network models, the demand of CPU is scaled according to the proportion of operations that need to be emulated [12], similarly in a probabilistic model the authors in [15] define the resource utilization of a virtual machine as the ratio between its resource consumption and resource allocation.

Optimal resource allocation mechanisms adapt to the load fluctuations in order to preserve a desired level of performance. These techniques can be reactive or proactive: in [10] server consolidation using virtualization is implemented monitoring key performance metrics and using the data to trigger migration of VMs within physical servers, in the same work heuristics are proposed to minimize the costs of migration/consolidation and maintain acceptable application performance levels. Instead, the authors of [4] presents a pro-active algorithm that adapts to demand changes and migrates VMs between physical hosts providing probabilistic SLA guarantees; time series forecasting techniques and bin packing heuristic are combined to minimize the number of physical machines required to support a workload.

Techniques to estimate resource requirements and models to estimate resource needs for combined workloads are investigated in [7], specifically for relational database management systems running on cloud environments. Also non relational database (i.e. no-SQL, map-reduce database) are considered: for instance [8] describes and evaluates a statistical framework that uses Kernel Canonical Correlation Analysis (KCCA) to predict the execution time of map-reduce jobs. KCCA-based predictions are used for optimizing decisions including job scheduling, resource allocation, and workload management. Moreover, due to the large number of applications/resources in a cloud environment, as shown in [1] also Mean Field Analysis can be used to identify the optimal allocation of VMs.

3 Impact of replication on response time

In this section the impact of a replication policy on the response time of different type of systems subject to fluctuating workloads is investigated. Here we will focus on the qualitative behavior of some performance indexes as a function of the number of replica required to meet a given objective. In the following sections, the analytical description of the policy and of the results obtained is presented.

We consider a system that must be suitably sized to provide an average response time less than a given threshold value (referred to as *performance constraint*, PC). To achieve this objective we assume to split evenly the flow of arriving requests among various servers that are dynamically allocated as the PC is violated. Indeed, with the increase of the number of executing requests, the mean response time increases gradually until it reaches or exceed the PC value. When this occurs, the server with the highest utilization, i.e., the bottleneck, is duplicated and the service

is replicated on it.
The new system configuration is then analyzed to check whether the PC is now satisfied. In fact, it may happen that, after duplication, the bottleneck migrate to another resource or that more servers are saturated at the same time and duplicating one of them has no effect on performance. If the constraint is still violated, the most utilized server of the new configuration is identified and replicated. This iterative process continues until the PC is satisfied with the workload considered. With this algorithm we will determine the *minimum* number of replica required to meet the desired PC.

We will model two types of systems, closed and open, executing homogeneous and heterogeneous workloads. Closed models have a fixed population, i.e., the number of requests in execution is constant, while in open models the population varies over time.

Initially, we consider a closed system executing homogeneous requests, i.e., a single class workload. The average response time of the system must be less than the PC value $T = 1.8$ *sec*. The system is composed of a single server and the total amount of service time required by a complete execution of a request, i.e., the *service demand*, is $D = 100$ *msec*. To compute the response time, throughput and other performance indexes of this system we use the analytical solution technique Mean Value Analysis (MVA)[14]. The top plot of Fig.1a shows the behavior of system response time R as the number of requests in execution increases up to 200 requests. Only one server is able to satisfy the PC with population sizes less than 20 requests. With population sizes greater than this value, the server is replicated and the number of replica increases of one unit, as shown in the bottom plot of Fig.1a. As the population size increases, it can be clearly observed a linear behavior of the number of replica needed to satisfy the PC.

We consider now a closed system with two resources that execute a workload consisting of heterogeneous requests. We assume that the requests can be classified into two classes with a population mix $\beta_1 = 0.8$ (see Section 5.3 for the definition of β_1) having different mean service demands to the two resources. This workload can be described by the following demand matrix:

$$\mathbf{D} = \begin{vmatrix} 100 & 90 \\ 120 & 80 \end{vmatrix} \tag{1}$$

where the rows correspond to the resources of the system and the columns to the two classes, and the values are expressed in *msec*. Figure 1b shows the system response time R, and the per-class system response time R_{c_1} of class 1 and R_{c_2} of class 2 requests. As the population size increases indefinitely, not only the system response time but also the per-class system response times tend to constant asymptotic values that are related to each other. Thus, the replication policy can also be triggered by thresholds of per-class response times. As described in the following sections, we derived simple closed form expressions illustrating the relationships among the number of replica and the bounds of per-class response times

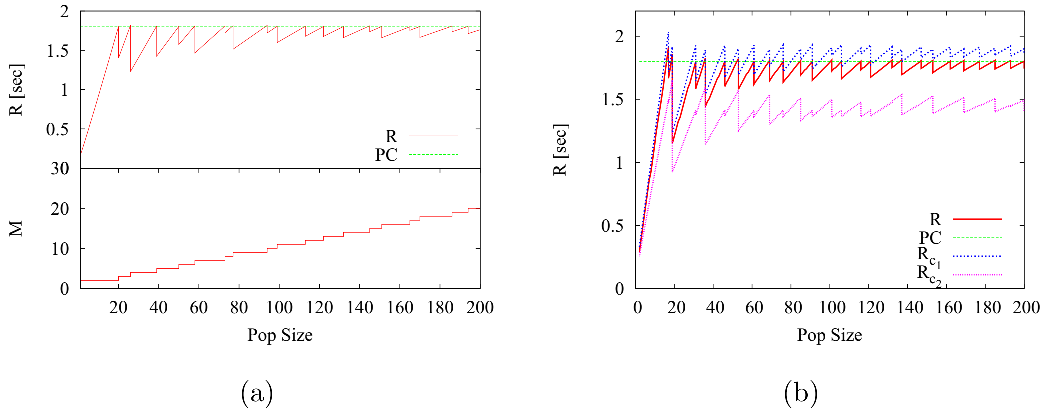

Fig. 1. System response time and number of replica in a single-class closed model (a); system response time and per-class response times in a two-class closed model (b). For both configurations, the threshold of the system response time is 1.8 *sec* in both systems.

using bounding techniques.

As a last example, we consider an open model, where the workload is characterized by increasing the arrival rate of requests. The same behavior of the system response times can also be seen in this case. The system has three resources and the workload consists of two classes of requests whose service demands are the following:

$$\mathbf{D} = \begin{vmatrix} 391 & 238 \\ 281 & 346 \\ 223 & 450 \end{vmatrix} \tag{2}$$

The threshold of the system response time is $T = 2$ *sec*. Figure 2 shows the behavior of the system response time, and of the response times of the single resources as a function of the arrival rate of requests. Also in this case the asymptotic values of the performance indexes are clearly identified.

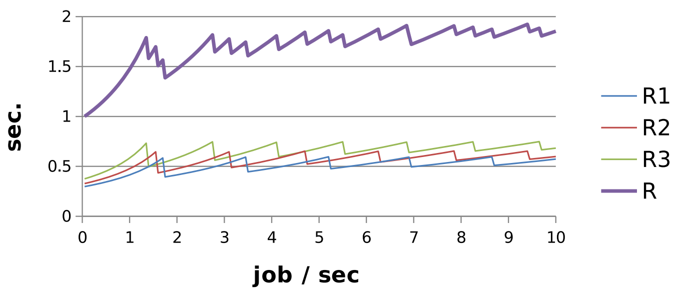

Fig. 2. Behavior of system and resource response times in a open model with three resources and two-class workload.

The notations that will be used through the paper are summarized in Table 1. Notice that the values of a performance index at each instance of a server are the same because the replications shares uniformly the same workload.

Table 1
Performance indexes and constraints

Index	Description	Threshold
R_{rc}	Residence time at resource r of a given class c	θ_{rc}
$R_r = \sum_c \frac{X_c}{\sum_{c'} X_{c'}} R_{rc}$	Aggregated residence time at a given resource r	θ_r
$R_c = \sum_r R_{rc}$	System response time of class c requests	θ_c
$R = \sum_c \frac{X_c}{\sum_{c'} X_{c'}} R_c$	System response time	θ

4 Asymptotic replication in open models

When considering open models, asymptotic behavior can be easily determined following the results presented in [9]. In that work, the optimal number of replica required to satisfy a given PC in open model were analytically determined using closed form expressions. In particular, it was shown that given the total number of stations in the system, the optimal value is proportional to the considered arrival rate vector $\Lambda = \sum_c \lambda_c$ (where λ_c is the arrival rate of class c jobs). Let us define with $\beta_c = \lambda_c/\Lambda$ as the fraction of arrivals that belongs to class c (clearly, we have $\sum_c \beta_c = 1$). For a given threshold θ and arrival rate Λ, it is possible to determine the total number of replica M_r of resource r to ensure that the PC is respected. In particular, if we consider a PC on the system response time $R = \theta$, it is shown that the values of M_r can be computed as:

$$M_r = \frac{\theta\Lambda \sum_c \beta_c d_{rc}}{\theta - \sum_c \beta_c \sum_r d_{rc}}. \tag{3}$$

Let us first imagine that M_r is an integer value for all resources. If we perform the replication and we share uniformly the workload among the identical resources, we produce a system characterized by the following demand matrix:

$$\mathbf{D} = \left| \begin{array}{l} M_1 \left\{ \begin{array}{ccc} \frac{d_{11}}{M_1} & \cdots & \frac{d_{1C}}{M_1} \\ & \vdots & \\ \frac{d_{11}}{M_1} & \cdots & \frac{d_{1C}}{M_1} \end{array} \right. \\ \vdots \\ M_R \left\{ \begin{array}{ccc} \frac{d_{R1}}{M_R} & \cdots & \frac{d_{R1}}{M_R} \\ & \vdots & \\ \frac{d_{R1}}{M_R} & \cdots & \frac{d_{RC}}{M_R} \end{array} \right. \end{array} \right| \tag{4}$$

By applying the classical queueing network results (see for example [11]), we can first verify that the system response $R = \theta$, as required by threshold. Then we can see that the mean residence time at a resource r, the response time of a class c, and the residence time of a class c job at a resource r are respectively defined as:

$$R_r = \theta \frac{\sum_c \beta_c d_{rc}}{\sum_c \beta_c \sum_r d_{rc}} \tag{5}$$

$$R_c = \theta \frac{\sum_r d_{rc}}{\sum_c \beta_c \sum_r d_{rc}} \tag{6}$$

$$R_{rc} = \theta \frac{d_{rc}}{\sum_c \beta_c \sum_r d_{rc}} \tag{7}$$

We will provide the derivations only of Equation 7 to simplify the presentation.

$$R_{rc} = M_r \frac{\frac{d_{rc}}{M_r}}{1 - \Lambda \sum\limits_c \beta_c \frac{d_{rc}}{M_r}} = \frac{d_{rc}}{1 - \Lambda \frac{\theta - \sum\limits_c \beta_c \sum\limits_r d_{rc}}{\theta \Lambda \sum\limits_c \beta_c d_{rc}} \sum_c \beta_c d_{rc}} \tag{8}$$

$$= \frac{d_{rc}}{1 - \frac{\theta - \sum\limits_c \beta_c \sum\limits_r d_{rc}}{\theta}} = \theta \frac{d_{rc}}{\sum_c \beta_c \sum_r d_{rc}}$$

In Equation 8, the term M_r that multiplies the first expression accounts the fact that resource r has been replicated M_r times. The value of M_r at the denominator is replaced with the expression given in Equation 3. In the derivations, M_r and Λ cancels out, making the expressions asymptotically valid for $\Lambda \to \infty$ also for cases in which M_r are not integer. For example, if we apply the results in Equation 5 to the model presented in Figure 2, with the threshold on the system response time $T = 2sec$, we obtain that the asymptotic response times for the three resources are respectively: $R_1 = 0.596\ sec$, $R_2 = 0.656\ sec$, $R_3 = 0.748\ sec$. The three asymptotes are confirmed from the results shown in Figure 2.

5 Asymptotic replication in closed models

In this Section, the relations between the asymptotic behavior of the system response time and both the class and resource aggregated residence time will be derived also for a closed model. A set of situations will be considered corresponding to the combinations of two factors. The type of workload of the model, that is single (S) or multiple classes (M), and the characteristics of the original system to be replicated: homogeneous (H), i.e. all its resources have the same demands or eterogeneous (N), i.e. the resources of the original system differ. A set of three cases will be studied: *SH*, *SN*, *MH*. The closed-form expressions for the optimal number of replica required to satisfy a given response time PC is given for all cases.

5.1 SH: Single class workload and homogeneous systems

This class of systems, although very simple, can be used to model for example a NO-SQL application running on a single-node DBMS: the considered resource represents the node and the network population corresponds to the tasks accessing the DBMS. The proposed model is shown in Figure 3a). In general, a node repeatedly executes its task on the data stored locally, allowing us to model the application as a closed queueing network.

When the application is executed on a set of nodes, data are shared between them. Moreover, the application input is partitioned evenly among nodes and tasks work on a specific data partition, reducing their final workload. The multi-node configuration system is shown in Figure 3b). A statistical equivalence assumption among nodes allows us to model each of them as a replica of the one in the single-node configuration. Tasks working on a data partition are routed by the scheduler to one of the available nodes. Let us call M the number of nodes, N the total task population. We denote with D the demand of the single node configuration and D_i the demands of the node i in the multi-node configuration. We assume the demands independent either on N or on M. Since each data partition has equal size, we have $D_i = D/M$, i.e. each node in the multi-node configuration process a fraction $1/M$ of the data processed in the single node configuration.

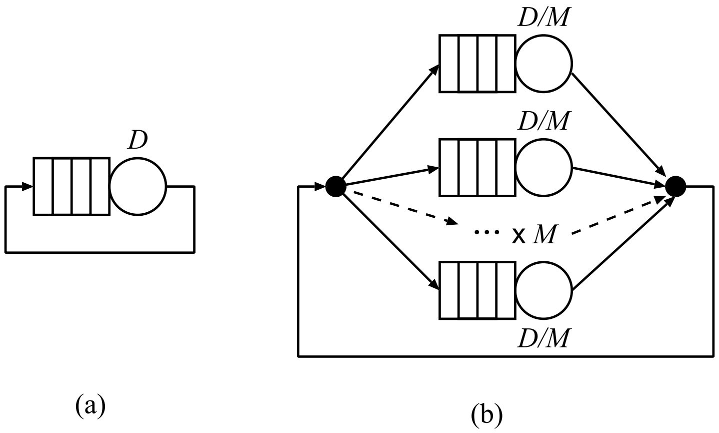

Fig. 3. Single class workload and homogeneous system

Parallelism is controlled by the distribution of tasks among the node queues through the router that connects all the replica together. This router models a task coordinator. We assume that the time required for the coordination is negligible with respect to the total demand of the nodes.

In general, the response time of a job depends on the scheduling policy implemented by the task coordinator. In particular, NO-SQL database can set policies with which data is distributed across nodes.

From now on let us focus on a random policy. Given that in the proposed replication process the data is evenly distributed among the nodes, on average each of them processes a similar amount of data, so the resulting system is always balanced. From [11], the throughput for balanced closed system is:

$$X = \frac{N}{N+M-1}\frac{1}{D_i} \tag{9}$$

We have that the demand of each nodes is $D_i = D/M$, therefore applying Little's law we obtain the system response time as following:

$$R = (N+M-1)\frac{D}{M} \tag{10}$$

By inverting the formula, we can determine the minimum number of replica required to obtain a mean response time less than or equal θ. We have:

$$M \geq \left\lceil \frac{(N-1)D}{T-D} \right\rceil \tag{11}$$

The previous equation holds for $N \geq 2$, for $N = 1$ it sufficient that $T \geq D$ to ensure the satisfaction of the constraints. Note also that for $T > D$ the problem has no solution: in this case the minimum number replica would be negative.

5.2 SN: Single class workload and heterogeneous systems

In this Section, we consider applications with a complex workflow, where each data must be processed by a series of services. Each service is carried out by a different and independent nodes, and thus it can work in parallel with other services working on different data in the same dataset. Such applications can be analyzed through a single class closed model with several resources. Each of them can be replicated, independently from the others, to reduce the total response time. Figure 4(a) shows the application model without replication, while the configuration where each service i is replicated M_i times is represented in Figure 4(b).

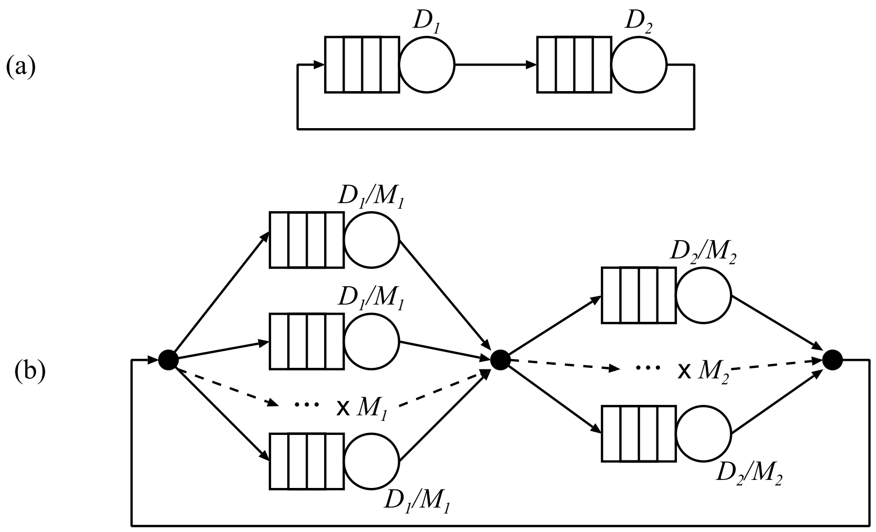

Fig. 4. Multiple class workload and heterogeneous system

Although this model seems structurally different from the model shown in Fig. 3, they have a similar behavior as the number of replica increases. Let D_i the total service demand of a job at the node dedicated to service . If the node is replicated M_i times the service demand of each replica i will be $D'_i = D_i/M_i$. The replication process always duplicates the actual bottleneck resources. As a result, as the process goes on, the bottleneck switches among all the resources and the system converges to a state in which all the demands are similar:

$$D'_i \approx D'_j \qquad \frac{D_i}{M_i} \approx \frac{D_j}{M_j} \qquad \forall i \neq j \tag{12}$$

Let us call $M = \sum_j M_j$, $D = \sum_j D_j$ and let us define:

$$\gamma_j = \frac{D_j}{D} \tag{13}$$

By definition, we have that $\sum_j \gamma_j = 1$, from which we can find:

$$1 = \sum_j \gamma_j = \sum_j \frac{D_j}{D} \approx \frac{1}{D} \sum_j M_j \frac{D_i}{M_i} = \frac{M}{D}\frac{D_i}{M_i} = \gamma_i \frac{M}{M_i} \tag{14}$$

that is: $M_i = \gamma_i M$. Using Equation 11, we have:

$$M_j \geq \left\lceil \gamma_i \frac{(N-1)D}{T-D} \right\rceil = \left\lceil \frac{(N-1)D_j}{T - \sum_j D_j} \right\rceil ; \quad M \geq \left\lceil \frac{(N-1)D}{T - \sum_j D_j} \right\rceil \tag{15}$$

Computing the residence time due to resource j when the threshold is set to a value θ can be done in this way: since each replica asymptotically tends to have the same demand, also their residence time will be the same, thus $R_{rep} = \frac{\theta}{M}$. We know that the total number of replica of resource j is M_j, thus with N growing to infinite we have:

$$\theta_{r_j} \approx M_j \frac{\theta}{M} = \frac{D_j}{D}\theta \tag{16}$$

Thus, the knowledge of the demand matrix and the value of threshold in the system response time allows to predict the asymptotic behavior of the residence times of each resource.

5.3 MH: Multi class workload and homogeneous systems

In this scenario we have a set of common nodes which must perform different applications, to consider actions on different set of data. This is a very common setting, where a company has several big data task to execute on a symmetric set of identical machines, all deployed in a cloud environment.

We model this by a single resource queueing network, where the resource can be replicated m times. However, the resource is able to handle different actions for different applications. Each application is modeled by a job class, and the number of data that must be handled by each action, is considered as the population of the corresponding class.

Consider a multiclass balanced network and let D_c/M the demand of the class c workload of each node. From [16] Equation (12), we have that the per-class throughput of a closed model with N_c jobs of class c is:

$$X_c(N_c) = \frac{N_c M}{(N+M-1)D_c} \tag{17}$$

applying Little's law, we can derive the per-class response time:

$$R_c(N_c) = \frac{(N+M-1)D_c}{M} \tag{18}$$

The system response time is defined as the per-class response time weighted with respect to the relative per-class throughput, thus using Equations (17) and (18) with a few algebraic passages we obtain:

$$\begin{aligned} R(N) &= \sum_c^C \frac{X_c(N_c)}{\sum_{c'}^C X_{c'}(N_{c'})} R_c(N_c) = \sum_c^C \frac{\frac{N_c M}{(N+M-1)D_c}}{\sum_{c'}^C \frac{N_{c'} M}{(N+M-1)D_{c'}}} \frac{(N+M-1)D_c}{M} \\ &= \frac{N+M-1}{M} \sum_c^C \frac{\frac{N_c M}{(N+M-1)}}{\sum_{c'}^C \frac{N_{c'} M}{(N+M-1)D_{c'}}} = \frac{N+M-1}{M} \sum_c^C \frac{N_c}{\sum_{c'}^C \frac{N_{c'}}{D_{c'}}} \\ &= \frac{N+M-1}{M} \frac{N}{\sum_{c'}^C \frac{N_{c'}}{D_{c'}}} = \frac{N+M-1}{M\Delta} \end{aligned} \tag{19}$$

where $\Delta = \sum_{c=1}^{C} \frac{\beta_c}{D_c}$.

From Equation 19 the number of machines needed to obtain a system response time below a value θ can be derived as:

$$R(N) = \frac{N+M-1}{M\Delta} \leq \theta \Rightarrow M \geq \frac{N-1}{\theta\Delta - 1} \tag{20}$$

Substituting the minimum value of M in Equation 18 we can obtain:

$$\theta_{c_i} \approx \frac{(N+M-1)D_c}{M} = \frac{D_c(\theta\Delta - 1)\left(N + \frac{N-1}{\theta\Delta-1} - 1\right)}{N-1} = \theta D_c \Delta \tag{21}$$

Notice that in multi class models the asymptotic value of the per-class residence time depends on the demand matrix, the threshold on the system response time and the population mix through the term Δ.

6 Validation

6.1 *Experimental results for a single class testbed*

The most widely used big data applications are based on NO-SQL and MapReduce paradigms that allow the processing of structured and unstructured data. A common problem to these paradigms is the computation of the number of nodes that must be allocated to meet the constraints on response time. Typically policies that replicates the most congested nodes are adopted in order to reduce the arrival rate of requests to each node.

In this section we use the techniques described in Section 5.1 to analyze an example of application based on the Apache Cassandra database management system. Cassandra [2] is a very popular NO-SQL open source database that provides high availability, fault-tolerance, and elastically scalable configurations.

It is able to work with commodity hardware across multiple data center racks, geographically dispersed. In Cassandra there is no master-slave relationship between nodes, as each one is identical to all others. A peer-to-peer protocol is present and a gossip system ensures status communication among nodes, easing the inclusion of new nodes to scale the cluster up or down.

One of the simplest techniques to improve the efficiency of a DB by exploiting parallelism is *data partitioning*. In particular, the dataset of the DB is divided into multiple tables, each containing a portion of the data. This type of organization is best suited to handle situations where records can be trivially divided into different sets, based for example on the first letter of the name of an employer or on the geographical location of a contact. A visual representation of this type of application is given in Figure 5.

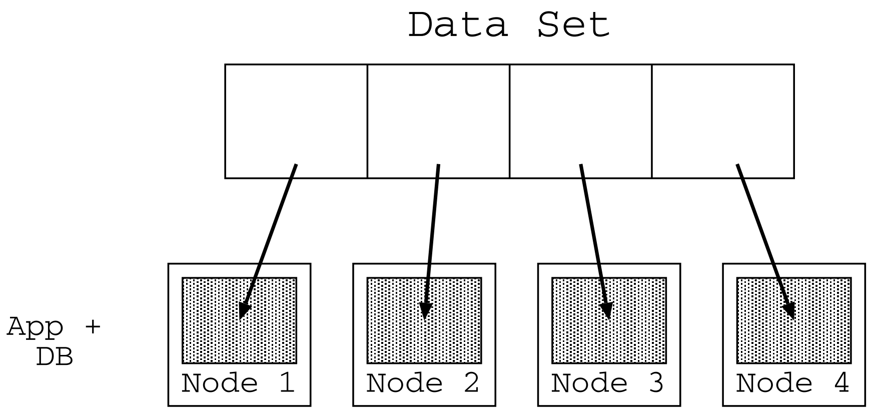

Fig. 5. An application with a partitioned data set.

We consider a setup with $M = [1 - 4]$ nodes, on which a large number of records (up to $N = 1500000$) are inserted by a benchmarking application. We use Cassandra as the DB, and we insert record using *Cassandra-stress*: one of the standard benchmarks included in the Datastax Cassandra distribution [5]. The Cassandra nodes are deployed on Amazon EC2 $m1.xlarge$ VMs, making sure that the provisioned machines are of comparable speed. Since, as described in [6], one of the most sensitive parameters of the performance variability of a VM is determined by the different CPU architecture on which it may be allocated by Amazon EC2 system, we ensured that at least all the VMs have the same type of CPU.

This application corresponds to the model presented in Section 5.1, where the nodes corresponds to the different partitions of the DB. We are interested in determining the minimum number of nodes required M to insert N records in less than a threshold of $T = 80s$.

We first collect response time by the benchmark application using different number of nodes. We then determine the mean demand of each job with a fitting procedure on the real data and we obtain $D = 0.207ms$. Fitting is performed in Microsoft Excel using the GRG non-linear solver. For each N, we compare the measure response time with the one forecast by the model. Results are shown in Figure 6: as it can be seen, the model is fully capable of capturing the real behavior of the benchmark application.

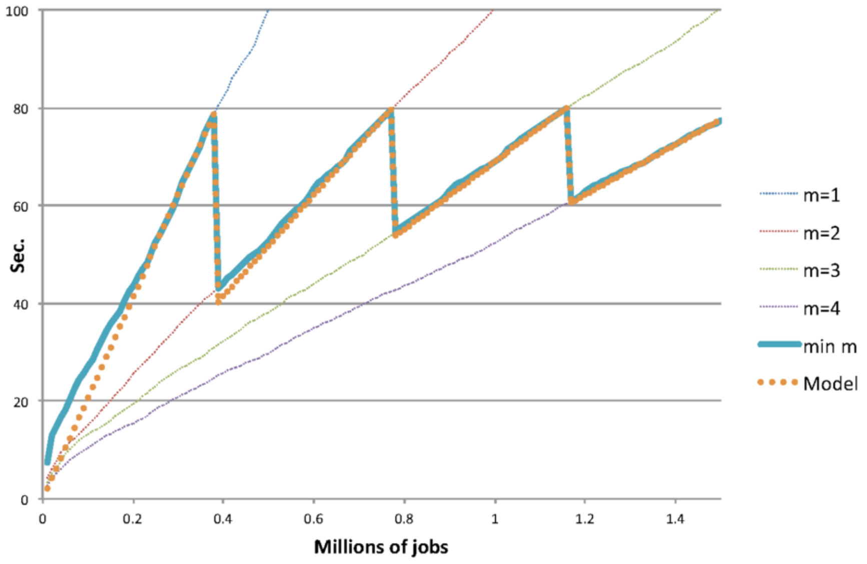

Fig. 6. Response time of a partitioned data set

6.2 *Simulation of the asymptotic behavior in closed models*

Several tests were performed using the MVA algorithm implemented in the JMT tool [3] in order to validate the analytic approaches proposed for closed models in Sections 5.2 and 5.3. In this Section the results of two tests are shown to highlight the difference between the behavior of the per-class and per-resource residence time.

The values provided by Equation 16 have been compared with the results obtained by a single class closed model with two resources and a system response time below the value of $\theta = 1.8 sec.$ The matrix vector of the demands was $\mathbf{D} = |70\ 90|\ msec.$ According to Equation 16 the asymptotic values of the residence time of each resource should be $\theta_{r_1} = 0.7875\ sec.$ and $\theta_{r_2} = 1.0125\ sec.$, respectively. Figure 7(a) shows the comparison between the results. Despite the presence of significant fluctuations in the residence time values, due to the effect of the replication, with a number of jobs greater than 400 the asymptotic behavior starts to appear and it is estimated quite-well by the proposed formula.

Equation 21 have been verified through the results obtained by the analysis of a multi class closed model with a single resource and a system response time below the value of $\theta = 1.8 sec.$ The matrix vector of the demands was $\mathbf{D} = |100\ 40|\ msec.$ The asymptotic values of the residence time of each class given by Equation 16 should be $\theta_{r_1} = 2.34\ sec.$ and $\theta_{r_2} = 0.936\ sec.$, respectively. Figure 7(b) shows that the per-class residence times have a better asymptotic behavior than the per-resource residence times shown in Figure 7(a). The per-class residence time trends appear

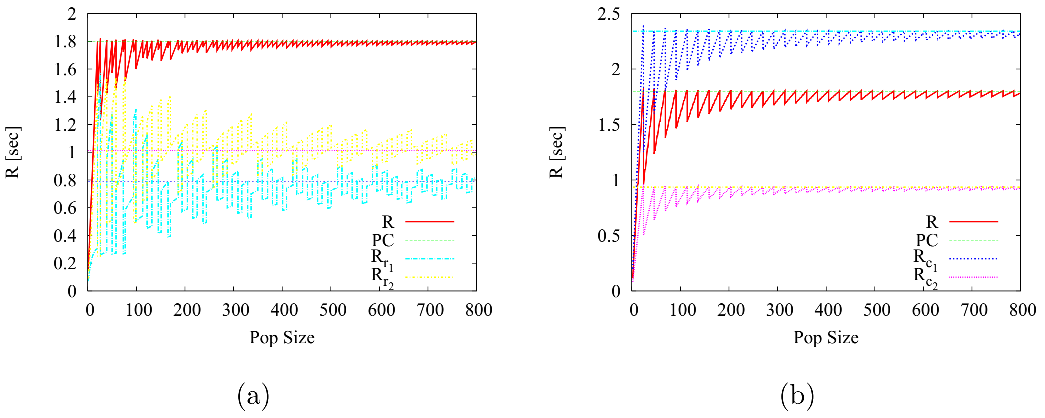

Fig. 7. System response time and residence time in a single class model (a); system and per-class residence time in a two class model with $[0 - 800]$ jobs. In both cases the horizontal lines represent the asymptotic bounds given by Equations 16 and 21, respectively.

more regular, with lighter fluctuations and it is less affected by the replication. For such reasons, its asymptotic behavior is estimated very well by Equation 21.

7 Conclusions and future work

In this work we have shown that performance objectives based on different metrics are not independent. The replication of services to reduce a specific metric below a given threshold affects also the other performance indexes. Relationships among indices can be exploited to design smart control policies that can achieve several performance objectives by controlling a single parameter. In this paper, we have investigated the relationships between system response time, per-class response time, and per-resource residence time as the number of replica tends to infinite. Through the definition of open and closed queuing network models, a set of relationships between the considered response and residence times at different levels has been derived. The optimal number of replica needed to satisfy a given performance objective was identified for different system models - in previous works such results were available only for open models. The proposed relationships were validated by comparison with results obtained using MVA algorithm.

Future works will go in three directions: first the relations among different metrics, such as utilization and queue length, will be considered. Then, we will complete the analytical computation of the optimal number of replica by including the case that it is still missing: multi-class closed models. Finally, dynamic replication policies that exploits the result proposed in this paper will be defined, tested and validated.

References

[1] Abbaldo, D., Gribaudo, M., Manini, D., 2012. Evaluation of different scheduling policies in iaas applications by mean field analysis. In: VALUETOOLS. pp. 307–316.

[2] Apache Cassandra, 2013. http://cassandra.apache.org/.

[3] Bertoli, M., Casale, G., Serazzi, G., 2006. Java modelling tools: an open source suite for queueing network modelling and workload analysis. In: Proc. of the 3rd Conf. on Quantitative Evaluation of Systems (QEST). IEEE, pp. 119–120.

[4] Bobroff, N., Kochut, A., Beaty, K., 21 2007-yearly 25 2007. Dynamic placement of virtual machines for managing sla violations. In: Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on. pp. 119 –128.

[5] Cassandra, A., 2013. Cassandra-stress benchmarking tool.
URL
http://www.datastax.com/documentation/cassandra/1.2\cassandra/tools/toolsCStress_t.html

[6] Cerotti, D., Gribaudo, M., Piazzolla, P., Serazzi, G., 2012. Flexible cpu provisioning in clouds: A new source of performance unpredictability. In: QEST. pp. 230–237.

[7] Curino, C., Jones, E. P., Madden, S., Balakrishnan, H., 2011. Workload-aware database monitoring and consolidation. In: Proceedings of the 2011 international conference on Management of data. SIGMOD '11. ACM, New York, NY, USA, pp. 313–324.

[8] Ganapathi, A., Chen, Y., Fox, A., Katz, R., Patterson, D., march 2010. Statistics-driven workload modeling for the cloud. In: Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on. pp. 87 –92.

[9] Gribaudo, M., Piazzolla, P., Serazzi, G., 2012. Consolidation and replication of vms matching performance objectives. In: ASMTA. pp. 106–120.

[10] Khanna, G., Beaty, K. A., Kar, G., Kochut, A., 2006. Application performance management in virtualized server environments. In: NOMS. pp. 373–381.

[11] Lazowska, E. D., Zahorjan, J., Graham, G. S., Sevcik, K. C., 1984. Quantitative System Performance. Prentice-Hall.

[12] Menascé, D. A., 2005. Virtualization: Concepts, applications, and performance modeling.

[13] Ongaro, D., Cox, A. L., Rixner, S., 2008. Scheduling i/o in virtual machine monitors. In: Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments. VEE '08. ACM, New York, NY, USA, pp. 1–10.

[14] Reiser, M., 1981. Mean-value analysis and convolution method for queue-dependent servers in closed queueing networks. Performance Evaluation 1, 7–18.

[15] Watson, B. J., Marwah, M., Gmach, D., Chen, Y., Arlitt, M., Wang, Z., 2010. Probabilistic performance modeling of virtualized resource allocation. In: Proceedings of the 7th international conference on Autonomic computing. ICAC '10. ACM, New York, NY, USA, pp. 99–108.

[16] Zahorjan, J., Sevcik, K. C., Eager, D. L., Galler, B., Feb. 1982. Balanced job bound analysis of queueing networks. Commun. ACM 25 (2), 134–141.