View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Archivio istituzionale della ricerca - Politecnico di Milano

A Framework for Joint Resource Allocation of MapReduce and Web Service Applications in a Shared Cloud Cluster

Lorela Cano[a,*], Giuliana Carello[a], Danilo Ardagna[a]

[a] *Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB)*
Politecnico di Milano, Italy

[*]Corresponding author
Email address: `lorela.cano@polimi.it` (Lorela Cano)

Abstract

The ongoing uptake of cloud-based solutions by different business domains and the rise of cross-border e-commerce in the EU require for additional public and private cloud solutions. Private clouds are an alternative for e-commerce sites to host not only Web Service (WS) applications but also Business Intelligence ones that consist of batch and/or interactive queries and resort to the MapReduce (MR) programming model.

In this study, we take the perspective of an e-commerce site hosting its WS and MR applications on a fixed-size private cloud cluster. We assume Quality of Service (QoS) guarantees must be provided to end-users, represented by upper-bounds on the average response times of WS requests and on the MR jobs execution times, as MR applications can be interactive nowadays. We consider multiple MR and WS user classes with heterogeneous workload intensities and QoS requirements. Being the cluster capacity fixed, some requests may be rejected at heavy load, for which penalty costs are incurred. We propose a framework to jointly optimize resource allocation for WS and MR applications hosted in a private cloud with the aim to increase cluster utilization and reduce its operational and penalty costs. The optimization problem is formulated as a non linear mathematical programming model. Applying the KKT conditions, we derive an equivalent problem that can be solved efficiently by a greedy procedure. The proposed framework increases cluster utilization by up to 18% while cost savings go up to 50% compared to a priori partitioning the cluster resources between the two workload types.

Keywords: MapReduce, Web Service, Resource management, Shared clusters, Nonlinear programming

*Corresponding author
Email address: lorela.cano@polimi.it (Lorela Cano)

1. INTRODUCTION

Despite the relatively high level of adoption cloud computing has reached in different business domains (circa 70% of businesses in EU in 2015), Information Technology (IT) analysts expect additional investments in public and private clouds [1]. Moreover, the recent efforts of the EU for a digital single market [2] are pushing cross-border e-commerce on one side and a rising need for scalable IT solutions on the other.

Private clouds become thus an attractive alternative for e-commerce sites to host not only traditional Web Service (WS) applications but also Business Intelligence (BI) ones accessing sometimes sensitive data. For instance, e-commerce market leaders host both the web site and BI applications on the same system to perform Key Performance Indicators (KPIs) analyses or to update the parameters of the recommender systems which is used in production [3, 4]. Such BI applications are a mix of batch and/or interactive queries that resort to the MapReduce (MR) programming model [5], e.g., log analysis, to evaluate the effect of a product campaign. The benefit of shared clusters is twofold: they allow for BI applications to consolidate multiple silos in a single "data lake" [6] while, from a resource management perspective, for reduced costs and higher utilization of the hardware infrastructure [7, 8].

A particular aspect of private clouds is that the available resources are finite whereas workloads are characterized by time-variant intensities. Therefore, private clouds have to cope with varying resource and Quality of Service (QoS) requirements, which makes admission control and capacity allocation very important. Concerning WS applications, capacity planning tools for guaranteeing target QoS levels for end users have been extensively addressed in the literature [9]. Instead, capacity allocation for MR applications has gained momentum only recently. However, to the best of our knowledge, this is the first attempt to jointly optimize WS and MR mangement.

MR applications have evolved from batch analysis on dedicated clusters based on a First In First Out (FIFO) scheduler [10, 11] to supporting both batch and interactive data analysis [12]. Nowadays, multiple users can submit large queries carried out on shared clusters which might have to be delivered within given deadlines. The execution time of a MapReduce job is unknown in advance [13, 14]; as a result, determining the optimal number of nodes in a cluster shared among multiple users that submit heterogeneous tasks is a difficult problem [14].

In this context, we propose an optimization framework that implements Admission Control (AC) and Capacity Allocation (CA) in a shared cluster hosting heterogeneous workloads ranging from WS to MR applications characterized by different resource and QoS requirements. The AC&CA determines the admitted workload (i.e., the number of concurrent MR jobs and the rate of served WS application requests for different user classes) and the optimal amount of resources (number of VMs) to be allocated while satisfying the QoS constraints. The goal of such framework is to optimize the cluster resource allocation with the aim to increase cluster utilization and to reduce costs. A classical QoS performance model has been adopted to predict the response times of WS requests whereas approximate formulae (proposed in [15]) provide estimates on the execution times of MR jobs so that the latter are bounded to given deadlines. The joint AC&CA problem is formulated by means of a mathematical model, whose objective is to minimize the operational costs for running the cluster and the penalty costs incurred from request rejections. Through theoretical analysis, we derive several proprieties characterizing the optimal solution of the proposed model, which allow to obtain closed formulae that translate the number of jobs and request rate for each MR and WS application class, respectively, into the optimal number of VMs to be allocated. The original problem is reduced to an equivalent one that can be solved efficiently by a greedy procedure.

Since the proposed approach allows to dynamically use the available resources between WS and MR application as opposed to an a priori resource partitioning between the two application types, we expect our framework to reduce the penalty cost due to rejected jobs/requests or, alternatively, for a given target rejection rate, to reduce the cluster size whenever the application workloads are shifted (e.g., when requests originate from different time zones). This is verified by the conducted experiments: for significant workload shifts, the cluster utilization is improved by up to 18% whereas cost savings go up to 50%.

The paper is organized as follows: Section 2 describes the proposed framework and the underlying problem assumptions. In Section 3 we introduce an initial formulation of the problem, derive several proprieties characterizing its optimal solution and provide a reduced linear programming (LP) formulation. The experimental settings and the numerical results are discussed in Section 4 whereas Section 5 summarizes the related work. Conclusions are finally drawn in Section 6.

2. A FRAMEWORK FOR JOINT ADMISSION CONTROL AND CAPACITY ALLOCATION OF MAPREDUCE AND WEB SERVICE APPLICATIONS

In this section we frame our reference scenario and describe its underlying assumptions. We take the perspective of an e-commerce site which hosts both MR and WS applications on a private cloud, i.e., on a fixed-size cluster. We account for QoS gurantees that need to be provided to end-users. For MR applications, which nowadays can be interactive [12], an upper bound on the job execution time needs to be provided. In the same way, to improve the user experience of e-commerce customers, the average response time of a WS application request needs to be few seconds. We consider multiple MR and WS user classes which are characterized by heterogeneous workload intensities, resource and QoS requirements. Since the cluster capacity is fixed, that is, the e-commerce site can allocate up to a maximum number of VMs to meet the QoS requirements, as in other approaches [16, 8, 17, 18], some requests may have to be rejected under heavy loads. We assume that penalty costs are incurred when requests are rejected. In addition, we account for the operational costs incurred from running the allocated VMs of the cluster, where the main component of such costs is the data center energy consumption. Therefore, it becomes important to optimally allocate resources and handle rejections in order to minimize the overall cost. In this work, we jointly address the AC&CA in a cluster which is shared between different user classes of MR/WS applications. The goal is to find the allocation at minimum total cost while satisfying the QoS requirements of the admitted requests. The proposed framework is illustrated in Figure 1.

In details, we make the following assumptions for the MR applications: (i) the subset of VMs

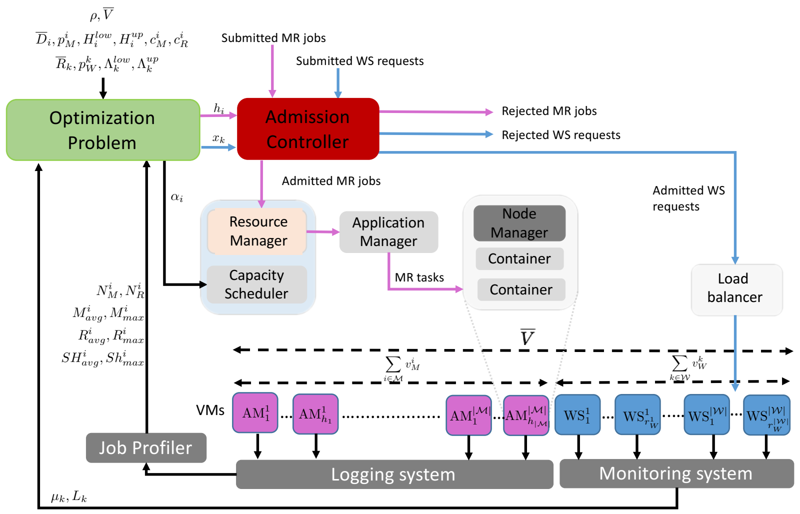


Figure 1: Framework for joint capacity allocation and admission control of MR and WS applications.

allocated to MR applications relies on an external storage (which allows to grow and shrink the number of running VMs [19]), (ii) VMs are shared among a set of competing MR user classes $\mathcal{M}$, and (iii) the YARN capacity scheduler [20] is adopted. A MR user class $i \in \mathcal{M}$ represents users that submit MapReduce jobs with the same *execution profile*, i.e., the jobs are characterized by a similar number of Map and Reduce tasks with similar Map, Reduce, and shuffle duration (e.g., users that run the same query). We consider interactive MapReduce applications which are characterized by given deadlines $\overline{D}_i$. As in [14, 15], we denote with M^i_{avg}, M^i_{max}, R^i_{avg}, R^i_{max}, $Sh^{1,i}_{avg}$, $Sh^{1,i}_{max}$, $Sh^{typ,i}_{avg}$ and $Sh^{typ,i}_{max}$ the average and maximum duration of the Map, Reduce, first Shuffle and typical Shuffle phases, respectively, while N^i_M and N^i_R are the number of Map and Reduce tasks [1]. According to [14], such parameters that consist a job profile can be extracted from Hadoop logs. Job execution times are then estimated as in [15] (see Section 3.1).

Concerning WS applications, we consider a set $\mathcal{W}$ of user classes where each class $k \in \mathcal{W}$ represents user requests that exhibit a similar resource demand. The QoS constraints are expressed in terms of the average response time $E\left[R_k\right]$ which should not exceed a given threshold $\overline{R}_k$. We assume the same application can be supported by multiple VMs that run in parallel. For the sake of simplicity, we further assume the VMs are homogeneous in terms of RAM and CPU capacity and therefore the workload is uniformly shared among them, which is a common practice in current cloud solutions. As in [21, 22], we model each WS application class hosted by a VM as an M/G/1 queue in tandem with a delay center, where the latter allows to account for network and/or protocol delays incurred during connection establishment. We suppose a processor sharing scheduling discipline [23] is adopted to serve requests. The maximum service rate and network delay for executing class k WS applications are denoted by μ_k and L_k, respectively and are supposed to be estimated at run-time from the monitoring system as in [24].

[1]In a shuffle phase, the mapper task data are moved to the nodes where the reduce tasks will be executed. Being the first and the typical shuffle significantly different, we make a distinction betwen the two as in [14].

An Admission Controller handles submitted MR jobs and WS application requests. Thus, some MR jobs [16, 8] and/or WS requests [17, 18] can be rejected. As far as MR applications are concerned, once a submitted job is accepted by the Admission Controller, it is further forwarded to the Resource Manager. The latter then allocates the job Application Master which in turn obtains a certain number of containers. Containers within a class are assumed to be homogeneous in terms of memory and CPUs and they are mapped one-to-one to the number of CPU cores of a VM. We consider a Hadoop 2.x framework, thus, the entire capacity of a container can be assigned to either Map tasks or Reduce tasks. [2] A two layer queue hierarchy in *work-conserving mode* is adopted for the capacity scheduler. The first layer queue, which represents the total cluster capacity allocated to MR applications, is partitioned among $|\mathcal{M}|$ queues and a fraction α_i of resources is dedicated to each MR user class.

Let us denote with h_i the number of simultaneously executed class i MR jobs. H_i^{up} denotes a prediction for the number of class i jobs to be executed. As a result, $h_i \leq H_i^{up}$. Furthermore, in order to avoid job starvation, we also impose h_i to be greater than a given lower bound H_i^{low}. Finally, p_M^i indicates the penalty cost for rejecting a class i MR job. Similarly, we denote by x_k the rate of served requests/throughput of WS applications of class k. Analogously to MR applications, we make sure that x_k lies within given bounds, that is, x_k will be at most Λ_{up}^k, which represents a prediction of the request rate of class k, and at least a given lower bound Λ_{low}^k. p_W^k then indicates the penalty cost for rejecting a class k WS application request. Let ρ be the cost for running each allocated VM and $\overline{V}$ the cluster size, that is, the maximum number of VMs that can be allocated simultaneously.

In essence, the aim is to determine the optimal number of VMs allocated to each class of the two application types, i.e., v_M^i (MR) and v_W^k (WS), the concurrency degree h_i for each MR class $i \in \mathcal{M}$ and the throughput x_k for each WS application class $k \in \mathcal{W}$ so that the total cost incurred from the allocated VMs and from rejections is minimized while QoS constraints are not violated, that is, the execution times of the MR jobs do not exceed the deadlines $\overline{D}_i$ and total average response times of the WS requests are below the thresholds $\overline{R}_k$. The proposed joint AC&CA problem is formally formulated in the next Section.

3. OPTIMIZATION PROBLEM

This section deals with several aspects of the mathematical model that lies behind our proposed framework. In particular, the performance models that have been adopted to estimate the execution

[2]Unlike in Hadoop 1.X, where the resources of each node can be split between slots assigned to Map and Reduce tasks, in Hadoop 2.x, the resource capacity of a container is configured so that it suits both Map and Reduce tasks and cannot be further partitioned [25]. YARN is responsible for pooling together the memory and CPU resources provided by the cluster nodes and handling the requests for containers needed to carry out the Map and Reduce tasks of a given job. It also determines the *slot count*, that is, the maximum number of simultaneous mappers and reducers, according to administrator settings [26]. A task can be carried out by a node when its available memory and CPU can accommodate the task resource requirements. Thus, under such assumptions, the configuration settings make sure that there are no idle vCPU due to incorrect parameter setting.

times of the MR jobs and the average response times of the WS requests are described in Section 3.1. In Section 3.2, we give an initial formulation of the problem and study the proprieties of its optimal solution which allows us to map the QoS requirements into resource requirements and eventually obtain a reduced problem formulation. Proprieties characterizing the optimal solution of the reduced problem and a greedy procedure which solves it to optimality are presented in Section 3.3.

3.1. MR and WS Applications Performance model

The amount of resources allocated to the different classes of MR jobs and WS application requests depends on their corresponding QoS requirements where the latter are expressed in terms of thresholds on the execution times ($\overline{D}_i$ for class i MR jobs) and on the average response times ($\overline{R}_k$ for class k WS application requests). In other words, at least a minimum amount of resources will have to be allocated to each MR/WS application class in order to avoid exceeding such thresholds for the admitted workload. Thus, to determine the optimal allocation of the cluster resources, it is important to appropriately estimate the execution times of the MR jobs and the average response times of the WS application requests.

For MR applications, given the execution profile of a class i job, that is, given the triple of positive constants A_i, B_i and C_i that make up the execution profile, according to [15], the job execution time can be approximated or upper bounded by the following formula (see Appendix A in [27]):

$$T_i = \frac{A_i\, h_i}{s^i_M} + \frac{B_i\, h_i}{s^i_R} + C_i, \qquad \forall i \in \mathcal{M}, \tag{1}$$

where s^i_M and s^i_R represent the number of Map and Reduce slots[3], respectively, devoted to the execution of class i jobs.

Concerning the accuracy of the approximated formulae, in [15], it is experimentally shown that the difference between the actual job execution time and the estimation ranges between 5 and 10% whereas the upper bound gap varies from 11 and 19%. Further, as a single VM can host either c^i_M Map slots of class i or c^i_R Reduce slots of class i, in order to provide s^i_M and s^i_R Map and Reduce slots to class i, $s^i_M/c^i_M + s^i_R/c^i_R$ VMs need to be provisioned.

Instead, for WS applications, as we model each WS class hosted by a VM as an M/G/1 queue in tandem with a delay center [21, 22] and we assume the workload to be evenly shared among the VMs allocated to WS applications of class k, then, the average response time for the execution of class k requests is given by:

$$E\left[R_k\right] = \frac{1}{\mu_k - \dfrac{x_k}{v^k_W}} + L_k, \qquad \forall k \in \mathcal{W}, \tag{2}$$

where L_k represents the network delay. Results reported in [29] demonstrated that this performance model allows to achieve a percentage error on the average response time estimation around 20%, which is appropriate for web systems run-time resource management [30].

[3]With reference to [15], the term slot is used to refer to the total number of containers vCPUs allocated for executing a single job to be in line with other big data frameworks (see, e.g., Apache Spark [28]).

3.2. Problem Formulation

The aim of our joint AC&CA problem is to minimize the total cost while meeting deadlines for MR jobs and not exceeding maximum average response times for WS applications. The total cost includes both the operational costs associated with the allocated VMs and the penalty costs incurred from rejections. Given T, the time for which we solve the AC&CA problem, ρ the cost for operating one VM during T, p_M^i, the penalty cost for rejecting a MR job of class $i \in \mathcal{M}$, p_W^k, the penalty cost for rejecting a WS request of class $k \in \mathcal{W}$, the total cost incurred during time T is then calculated as:

$$\sum_{i \in \mathcal{M}} \rho v_M^i + p_M^i \left(H_i^{up} - h_i\right) + \sum_{k \in \mathcal{W}} \rho v_W^k + p_W^k T \left(\Lambda_k^{up} - x_k\right),$$

where h_i and v_M^i, $\forall i \in \mathcal{M}$ are the decision variables concerning MR jobs, and they represent the class i concurrency level, that is, the number of simultaneously admitted jobs within the class, and its number of allocated VMs, respectively. Analogously, the decision variables concerning WS applications are x_k and v_W^k, $\forall k \in \mathcal{W}$, and they represent the throughput of class k, that is, its rate of served requests, and its number of allocated VMs, respectively.

In details, the objective function terms ρv_M^i and ρv_W^k represent the operational costs incurred during time T associated with the VMs allocated to class i MR jobs and class k WS application requests, respectively. Instead, $p_M^i \left(H_i^{up} - h_i\right)$ and $p_W^k T \left(\Lambda_k^{up} - x_k\right)$ represent the penalty costs incurred from rejecting MR jobs and WS requests. In other words, such costs are paid when not all the expected workload is accommodated, i.e., not all submitted MR jobs ($h_i < H_i^{up}$) and WS requests ($x_k < \Lambda_k^{up}$) are served. However, since $p_M^i H_i^{up}$ and $p_W^k T \Lambda_k^{up}$ are constant terms, the objective can be reduced to:

$$\rho \left(\sum_{i \in \mathcal{M}} v_M^i + \sum_{k \in \mathcal{W}} v_W^k \right) - \sum_{i \in \mathcal{M}} p_M^i h_i - \sum_{k \in \mathcal{W}} p_W^k T x_k.$$

For MR applications, we also need to decide the number of allocated Map and Reduce slots which are represented by variables s_M^i and s_R^i, respectively.

Let (P0) denote the optimization problem formulation defined as follows:

$$\min \rho \left(\sum_{i \in \mathcal{M}} v_M^i + \sum_{k \in \mathcal{W}} v_W^k \right) - \sum_{i \in \mathcal{M}} p_M^i h_i - \sum_{k \in \mathcal{W}} p_W^k T x_k$$

$$\frac{A_i \, h_i}{s_M^i} + \frac{B_i \, h_i}{s_R^i} + E_i \leq 0, \qquad \forall i \in \mathcal{M} \tag{3}$$

$$\frac{s_M^i}{c_M^i} + \frac{s_R^i}{c_R^i} \leq \Xi_i v_M^i, \qquad \forall i \in \mathcal{M} \tag{4}$$

$$\frac{1}{\mu_k - \frac{x_k}{v_W^k}} + L_k \leq \overline{R}_k, \qquad \forall k \in \mathcal{W} \tag{5}$$

$$x_k < \mu_k v_W^k, \qquad \forall k \in \mathcal{W} \tag{6}$$

$$\sum_{i \in \mathcal{M}} v_M^i + \sum_{k \in W} v_W^k \leq \overline{V} \tag{7}$$

$$H_i^{low} \leq h_i \leq H_i^{up}, \qquad \forall i \in \mathcal{M} \tag{8}$$

$$\Lambda_k^{low} \leq x_k \leq \Lambda_k^{up}, \qquad \forall k \in \mathcal{W} \tag{9}$$

$$v_M^i \geq 0, s_M^i \geq 0, s_R^i \geq 0, \qquad \forall i \in \mathcal{M} \tag{10}$$

$$v_W^k \geq 0, \qquad \forall k \in \mathcal{W} \tag{11}$$

where the objective is to minimize the total cost while meeting deadlines for the MR jobs and not exceeding the response times for WS applications.

Constraints (3) are obtained imposing that each job should be executed before the corresponding deadline, that is, $T_i \leq \overline{D}_i$ (see Equations (1) in Section 3.1), where E_i is the difference between C_i and $\overline{D}_i$, i.e., $E_i = C_i - \overline{D}_i < 0$. Constraints (4) make sure that the number of VMs allocated to each MR job class can accommodate the Map and Reduce slots needed to execute them. [4]

Constraints (5) limit the total average response time of each WS application class to its given threshold. Constraints (6) impose the equilibrium conditions for the M/G/1 queues of each WS application class, that is, they make sure that VMs are not saturated. Constraint (7) limits the total number of VMs allocated to both MR and WS applications to the cluster size, that is, to the maximum number of VMs that can be allocated. Constraints (8) and (9) bound the MR concurrency level and the WS throughput for each MR/WS application class whereas (10) and (11) make sure the remaining variables are nonnegative. The complete notation used throughout the paper is summarized in Tables 1 and 2.

The proposed formulation is the continuous relaxation of the real problem since variables v_M^i, v_W^k, s_M^i, s_R^i and h_i should be integer. If we imposed integrality, the problem would be much more difficult. Instead, we address its approximation (this approach has also been adopted by other works, e.g., [31] and [32]). Such approximation is justifiable, since we can round the relaxed variables in post-processing to the closest integer with insignificant increase of the total cost, especially for real systems that require tens or hundreds of relatively cheap VMs per application class.

Problem (P0) has a linear objective function, but two families of non-linear constraints: (3) and (5). Linearizing Constraints (5) is quite trivial: first, we rewrite them [5] as:

$$\frac{v_W^k}{\mu_k v_W^k - x_k} \leq \overline{R}_k - L_k. \tag{12}$$

Let $G_k = \overline{R}_k - L_k$ (maximum sojourn time) and $F_k = 1 - G_k\mu_k$. Multiplying both sides of (12) by $\mu_k v_W^k - x_k$ (where $\mu_k v_W^k - x_k > 0$ due to Constraints (6)) and appropriately collecting linear terms, Constraints (5) can be substituted by:

$$F_k v_W^k + G_k x_k \leq 0, \qquad \forall k \in \mathcal{W}. \tag{13}$$

Instead, Constraints (3) are neither linear nor convex. In [15], it is shown that they can be convexified replacing variables h_i with $1/\Psi_i$, $\forall i \in \mathcal{M}$. Such substitution transforms Constrains (3)

[4]We denote by Ξ_i the ratio $T/\overline{D}_i$ which represents the reuse factor of a VM allocated to class i MR jobs. In other words, Ξ_i accounts for the number of times during T a VM can be used for executing class i MR jobs as we solve the AC&CA problem every T seconds wheres jobs are executed in at most $\overline{D}_i$ seconds. In such a way, VMs do not necessarily have to remain idle in the remaining period $T - \overline{D}_i$. See [15] for further details.

[5]We remark that in any feasible solution v_W^k should be non-zero. Since we consider strictly positive minimum throughputs, that is, since $x_k \geq \Lambda_{low}^k > 0$ and, therefore, due to Constraints (6), we must have $v_W^k > 0$.

Parameters	
T	Time for which we solve the AC&CA problem
$\overline{V}$	Cluster size: maximum number of VMs that can be allocated simultaneously during T
ρ	Operational cost during time T of one VM
$\mathcal{M}$	Set of MR applications classes
c_M^i	Maximum number of class i Map slots that can be hosted in a VM
c_R^i	Maximum number of class i Reduce slots that can be hosted in a VM
p_M^i	Penalty cost for rejecting a class i MR job
$\overline{D}_i$	Execution deadline of class i jobs
A_i	CPU requirement for the Map phase for class i jobs
B_i	CPU requirement for the Reduce phase for class i jobs
C_i	Time constant factor that depends on Map, Copy, Shuffle, and Reduce phases of class i jobs
H_i^{up}	Expected number of class i jobs to be executed concurrently
H_i^{low}	Minimum number of class i MR jobs to be executed concurrently
Ξ_i	Reuse factor during T of a class i VM
$\mathcal{W}$	Set of WS applications classes
$\overline{R}_k$	Maximum tolerated response time for a class k WS application
L_k	Network delay for a class k WS application
u_k	Maximum service rate for a class k WS application
Λ_k^{up}	Expected request rate of class k WS applications
Λ_k^{low}	Minimum throughput of class k WS applications
p_W^k	Penalty cost for rejecting a class k WS request

Table 1: Optimization model: parameters.

Decision Variables	
v_M^i	Number of VMs to be allocated for executing class i MR jobs
s_M^i	Number of slots to be allocated to MR class i for executing the Map tasks
s_R^i	Number of slots to be allocated to MR class i for executing the Reduce tasks
h_i	Number of class i MR jobs to be executed concurrently
v_W^k	Number of VMs to be allocated for serving class k WS application requests
x_k	Throughput of class k WS application requests

Table 2: Optimization model: decision variables.

of (P0) into Constraints (14), where the latter are convex (see Appendix B in [15] for the proof). Finally, problem (P0) becomes equivalent to problem (P1):

$$\min \rho \left(\sum_{i \in \mathcal{M}} v_M^i + \sum_{k \in \mathcal{W}} v_W^k \right) - \sum_{i \in \mathcal{M}} p_M^i \frac{1}{\Psi_i} - \sum_{k \in \mathcal{W}} p_W^k T x_k$$

$$\frac{A_i}{s_M^i \Psi_i} + \frac{B_i}{s_R^i \Psi_i} + E_i \le 0, \qquad \forall i \in \mathcal{M} \tag{14}$$

$$\frac{s_M^i}{c_M^i} + \frac{s_R^i}{c_R^i} \le \Xi_i r_M^i, \qquad \forall i \in \mathcal{M} \tag{15}$$

$$F_k v_W^k + G_k x_k \le 0, \qquad \forall k \in \mathcal{W} \tag{16}$$

$$x_k < \mu_k v_W^k, \qquad \forall k \in \mathcal{W} \tag{17}$$

$$\sum_{i \in \mathcal{M}} v_M^i + \sum_{k \in W} v_W^k \le \overline{V} \tag{18}$$

$$\Psi_i^{low} \le \Psi_i \le \Psi_i^{up}, \qquad \forall i \in \mathcal{M} \tag{19}$$

$$\Lambda_k^{low} \le x_k \le \Lambda_k^{up}, \qquad \forall k \in \mathcal{W} \tag{20}$$

$$v_M^i \ge 0, s_M^i \ge 0, \, s_R^i \ge 0, \qquad \forall i \in \mathcal{M} \tag{21}$$

$$v_W^k \ge 0, \qquad \forall k \in \mathcal{W} \tag{22}$$

where $\Psi_i^{low} = 1/H_i^{up}$ and $\Psi_i^{up} = 1/H_i^{low}$.

A thorough investigation of the optimal solutions of problem (P1) provides several interesting closed formulae that establish relations among the different problem variables and allow us to obtain a much simpler problem formulation (Theorem 3.3). In particular, given a certain job profile (A_i, B_i, E_i, c_M^i, c_R^i) and its admitted concurrency level h_i (i.e., number of class i jobs that are served simultaneously), we can determine the optimal scheduling parameters for MR applications, that is, the number of Map and Reduce slots allocated to that job class and therefore its scheduling ratio (Theorem 3.1). Moreover, we are able to map the demand of each application class into the amount of resources needed to accommodate such demand (Theorem 3.2), that is, the concurrency level h_i for MR applications and the rate of the served requests for WS applications x_k can be translated into the number of VMs, v_M^i and v_W^k, to be allocated to their respective application classes.

In the remaining part of this section we go through the demonstration of these theorems.

Theorem 3.1. *Constraints (14) hold as equalities in any optimal solution of problem (P1). The number of Map and Reduce slots that have to be assigned to class i jobs, s_M^i and s_R^i, can be determined as follows:*

$$s_M^i = -\frac{1}{E_i\Psi_i}\left(A_i + \sqrt{\frac{A_i\,B_i\,c_M^i}{c_R^i}}\right), \tag{23}$$

$$s_R^i = -\frac{1}{E_i\Psi_i}\left(B_i + \sqrt{\frac{A_i\,B_i\,c_R^i}{c_M^i}}\right). \tag{24}$$

Proof. Given that in problem (P1) all constraints are convex ([33], [15]) and Slater constraints qualification hold (see Appendix B in [27]), we can use the *Karush-Kuhn-Tucker* (KKT) conditions.

The Lagrangian function of problem (P1), $\mathcal{L}_{(\mathrm{P1})}\left(v_M^i, s_M^i, s_R^i, \Psi_i, v_W^k, x_k\right)$, is:

$$\begin{aligned}
\mathcal{L}_{(\mathrm{P1})} \;=\; & \rho\left(\sum_{i\in\mathcal{M}} v_M^i + \sum_{k\in\mathcal{W}} v_W^k\right) - \sum_{i\in\mathcal{M}} p_M^i \frac{1}{\Psi_i} - \sum_{k\in\mathcal{W}} p_W^k T x_k \\
& + \sum_{i\in\mathcal{M}} \lambda_i^{(1)}\left(\frac{A_i}{s_M^i\Psi_i} + \frac{B_i}{s_R^i\Psi_i} + E_i\right) + \sum_{i\in\mathcal{M}} \lambda_i^{(2)}\left(\frac{s_M^i}{c_M^i} + \frac{s_R^i}{c_R^i} - \Xi_i v_M^i\right) \\
& + \sum_{k\in\mathcal{W}} \nu_k^{(1)}\left(F_k v_W^k + G_k x_k\right) + \sum_{k\in\mathcal{W}} \nu_k^{(2)}\left(x_k - \mu_k v_W^k\right) \\
& + \xi\left(\sum_{i\in\mathcal{M}} v_M^i + \sum_{k\in W} v_W^k - \overline{V}\right) \\
& + \sum_{i\in\mathcal{M}} \lambda_i^{(3)}(\Psi_i^{low} - \Psi_i) + \sum_{i\in\mathcal{M}} \lambda_i^{(4)}(\Psi_i - \Psi_i^{up}) + \sum_{k\in\mathcal{W}} \nu_k^{(3)}\left(\Lambda_k^{low} - x_k\right) + \sum_{k\in\mathcal{W}} \nu_k^{(4)}\left(x_k - \Lambda_k^{up}\right) \\
& - \sum_{i\in\mathcal{M}} \lambda_i^{(5)} v_M^i - \sum_{i\in\mathcal{M}} \lambda_i^{(6)} s_M^i - \sum_{i\in\mathcal{M}} \lambda_i^{(7)} s_R^i - \sum_{k\in\mathcal{W}} \nu_k^{(5)} v_W^k.
\end{aligned}$$

The KKT conditions for optimality are:

$$\rho - \lambda_i^{(2)}\Xi_i + \xi - \lambda_i^{(5)} = 0, \quad \forall i \in \mathcal{M}, \tag{25}$$

$$-\frac{\lambda_i^{(1)} A_i}{(s_M^i)^2 \Psi_i} + \frac{\lambda_i^{(2)}}{c_M^i} - \lambda_i^6 = 0, \quad \forall i \in \mathcal{M}, \tag{26}$$

$$-\frac{\lambda_i^{(1)} B_i}{(s_R^i)^2 \Psi_i} + \frac{\lambda_i^{(2)}}{c_R^i} - \lambda_i^7 = 0, \quad \forall i \in \mathcal{M}, \tag{27}$$

$$\frac{p_M^i}{\Psi_i^2} - \frac{\lambda_i^{(1)} A_i}{s_M^i \Psi_i^2} - \frac{\lambda_i^{(1)} B_i}{s_R^i \Psi_i^2} - \lambda_i^{(3)} + \lambda_i^{(4)} = 0, \quad \forall i \in \mathcal{M}, \tag{28}$$

$$\rho + \nu_k^{(1)} F_k - \nu_k^{(2)} \mu_k + \xi - \nu_k^{(5)} = 0, \quad \forall k \in \mathcal{W}, \tag{29}$$

$$-p_W^k T + \nu_k^{(1)} G_k + \nu_k^{(2)} - \nu_k^{(3)} + \nu_k^{(4)} = 0, \quad \forall k \in \mathcal{W}, \tag{30}$$

whereas the complementary slackness (CS) constraints are:

$$\lambda_i^{(1)} \left(\frac{A_i}{s_M^i \Psi_i} + \frac{B_i}{s_R^i \Psi_i} + E_i \right) = 0, \;\; \lambda_i^{(1)} \geq 0, \quad \forall i \in \mathcal{M}, \tag{31}$$

$$\lambda_i^{(2)} \left(\frac{s_M^i}{c_M^i} + \frac{s_R^i}{c_R^i} - \Xi_i v_M^i \right) = 0, \;\; \lambda_i^{(2)} \geq 0, \quad \forall i \in \mathcal{M}, \tag{32}$$

$$\nu_k^{(1)} \left(F_k v_W^k + G_k x_k \right) = 0, \;\; \nu_k^{(1)} \geq 0, \quad \forall k \in \mathcal{W}, \tag{33}$$

$$\nu_k^{(2)} \left(x_k - \mu_k v_W^k \right) = 0, \;\; \nu_k^{(2)} \geq 0, \quad \forall k \in \mathcal{W}, \tag{34}$$

$$\xi \left(\sum_{i \in \mathcal{M}} v_M^i + \sum_{k \in W} v_W^k - \overline{V} \right) = 0, \;\; \xi \geq 0, \tag{35}$$

$$\lambda_i^{(3)} (\Psi_i^{low} - \Psi_i) = 0, \;\; \lambda_i^{(3)} \geq 0, \quad \forall i \in \mathcal{M}, \tag{36}$$

$$\lambda_i^{(4)} (\Psi_i - \Psi_i^{up}) = 0, \;\; \lambda_i^{(4)} \geq 0, \quad \forall i \in \mathcal{M}, \tag{37}$$

$$\nu_k^{(3)} \left(\Lambda_k^{low} - x_k \right) = 0, \;\; \nu_k^{(3)} \geq 0, \quad \forall k \in \mathcal{W}, \tag{38}$$

$$\nu_k^{(4)} (x_k - \Lambda_k^{up}) = 0, \;\; \nu_k^{(4)} \geq 0, \quad \forall k \in \mathcal{W}, \tag{39}$$

$$\lambda_i^{(5)} v_M^i = 0, \;\; \lambda_i^{(5)} \geq 0, \quad \forall i \in \mathcal{M}, \tag{40}$$

$$\lambda_i^{(6)} s_M^i = 0, \;\; \lambda_i^{(6)} \geq 0, \quad \forall i \in \mathcal{M}, \tag{41}$$

$$\lambda_i^{(7)} s_R^i = 0, \;\; \lambda_i^{(7)} \geq 0, \quad \forall i \in \mathcal{M}, \tag{42}$$

$$\nu_k^{(5)} v_W^k = 0, \;\; \nu_k^{(5)} \geq 0, \quad \forall k \in \mathcal{W}. \tag{43}$$

Due to Constraints (14), s_M^i, and s_R^i are strictly positive. As a result, from the CS constraints (41) and (42), in any optimal solution, the dual variables $\lambda_i^{(6)}$ and $\lambda_i^{(7)}$ are equal to zero, $\forall i \in \mathcal{M}$. Since $s_M^i > 0$ and $s_R^i > 0$, from Constraints (15), also $v_M^i > 0$. Thus, also dual variables $\lambda_i^{(5)}$ should be equal to zero for the CS constraints (40) to hold.

Setting $\lambda_i^{(5)} = 0$ in the KKT conditions (25), we obtain $\lambda_i^{(2)} = \frac{\rho+\xi}{\Xi_i} > 0$ (since $\rho > 0$, $\Xi_i > 0$ and $\xi \geq 0$). Being $\Psi_i > 0$ (due to Constraints (14)), $s_M^i > 0$, $\lambda_i^{(2)} > 0$, $\lambda_i^6 = 0$ and, from the KKT conditions (26), $\lambda_i^{(1)} = \frac{\Psi_i (s_M^i)^2 \lambda_i^{(2)}}{A_i c_M^i}$, we also have that $\lambda_i^{(1)} > 0$. Then, due to the CS constraints (31), Constraints (14) hold as equalities. Combining the KKT conditions (26) and (27) with Constraints (14), we can express variables s_M^i and s_R^i in terms of variables Ψ_i. We first express

s_R^i in terms of s_M^i using the KKT conditions (26) and (27):

$$\frac{A_i c_M^i}{(s_M^i)^2} = \frac{B_i c_R^i}{(s_R^i)^2} \implies s_R^i = s_M^i \sqrt{\frac{B_i c_R^i}{A_i c_M^i}}. \tag{44}$$

Replacing s_R^i with $s_M^i\sqrt{(B_i c_R^i)/(A_i c_M^i)}$ in (14), we can express s_M^i and s_R^i in closed form as follows:

$$s_M^i = -\frac{1}{E_i \Psi_i}\left(A_i + \sqrt{\frac{A_i\, B_i\, c_M^i}{c_R^i}}\right), \tag{45}$$

$$s_R^i = -\frac{1}{E_i \Psi_i}\left(B_i + \sqrt{\frac{A_i\, B_i c_R^i}{c_M^i}}\right). \tag{46}$$

□

Remark 1. Theorem 3.1 allows to compute the optimal weights α_i of the YARN capacity scheduler in terms of the optimal number of Map and Reduce slots, s_M^i and s_R^i:

$$\alpha_i = \frac{s_M^i + s_R^i}{\sum\limits_{j \in \mathcal{M}} (s_M^j + s_R^j)}. \tag{47}$$

Theorem 3.2. *In any optimal solution of problem (P1) Constraints* (15) *and* (16) *hold as equalities. The demand of each MR and WS class can be translated into the amount of resources needed to support it, that is, the number of VMs allocated to class i MR jobs (v_M^i) can be expressed in terms of its concurrency level (h_i) whereas the number of VMs allocated to class k WS application requests (v_W^k) as a function of the throughput (x_k):*

$$v_M^i = \frac{\gamma_i}{\Xi_i} h_i, \qquad \forall i \in \mathcal{M}, \tag{48}$$

$$v_W^k = -\frac{G_k}{F_k} x_k, \qquad \forall k \in \mathcal{W}. \tag{49}$$

Proof. When proving Theorem 3.1, we have shown that $\lambda_i^{(2)} > 0$. Therefore, because of the CS constraints (32), Constraints (15) hold as equalities. Since we consider strictly positive lower bounds on the MR jobs concurrency level ($H_i^{low} > 0$) and WS requests throughput ($\Lambda_i^{low} > 0$), then $x_k > 0$ and consequently due to Constraints (16) and (17), $v_W^k > 0$. Thus, for the CS constraints (43) to hold, dual variables $\nu_k^{(5)}$ are equal to zero. Moreover, given that Constraints (17) are strict, due to Constraints (34), also dual variables $\nu_k^{(2)}$ are equal to zero. Then, from the KKT conditions (29), $\nu_k^{(1)} = -\frac{1}{F_k}(\rho - \nu_k^{(2)}\mu_k + \xi - \nu_k^{(5)}) = -\frac{1}{F_k}(\rho + \xi)$. Since $F_k < 0$ [6], $\rho > 0$ and $\xi \geq 0 \implies \nu_k^{(1)} > 0$. Thus, because of the CS constraints (33), also Constraints (16) hold as equalities.

[6]Since the service time $\frac{1}{\mu_k}$ should be smaller than the average sojourn time $\mathrm{E}[R_k] - L_k$, that is, $\frac{1}{\mu_k} < (\mathrm{E}[R_k] - L_k)$, then $\frac{1}{\mu_k} < (\overline{R}_k - L_k)$ and $F_k = 1 - (\overline{R}_k - L_k)\mu_k < 0$

According to Theorem 3.1, we can express s_M^i and s_R^i in terms of ψ_i and thus in terms of h_i. Let us denote by γ_i^1 and γ_i^2 the followings:

$$\gamma_i^1 = -\frac{1}{E_i\, c_M^i}\left(A_i + \sqrt{\frac{A_i\, B_i\, c_M^i}{c_R^i}}\right) = \frac{1}{c_M^i} s_M^i\, \Psi_i, \tag{50}$$

$$\gamma_i^2 = -\frac{1}{E_i\, c_R^i}\left(B_i + \sqrt{\frac{A_i\, B_i\, c_R^i}{c_M^i}}\right) = \frac{1}{c_R^i} s_R^i\, \Psi_i. \tag{51}$$

Under this notation and replacing $\Psi_i = 1/h_i$ we can rewrite (23) and (24) as:

$$s_M^i = \gamma_i^1 c_M^i h_i, \tag{52}$$

$$s_R^i = \gamma_i^2 c_R^i h_i. \tag{53}$$

Thus, replacing s_M^i/c_M^i and s_R^i/c_R^i in Constraints (15) we obtain the equivalent Constraints (54):

$$\gamma_i h_i = \Xi_i v_M^i, \qquad \forall i \in \mathcal{M}, \tag{54}$$

where $\gamma_i = \gamma_i^1 + \gamma_i^2$. Finally, (48) derive directly from (54), whereas (49) from Constraints (16) since the latter hold as equalities. $\square$

Theorem 3.3. *Problem (P1) can be reduced to problem (P2):*

$$\min \sum_{i\in\mathcal{M}} \left(\rho - \bar{p}_M^i\right) v_M^i + \sum_{k\in\mathcal{W}} \left(\rho - \bar{p}_W^k\right) v_W^k$$

$$\sum_{i\in\mathcal{M}} v_M^i + \sum_{k\in\mathcal{W}} v_W^k \le \overline{V} \tag{55}$$

$$V_M^{low,i} \le v_M^i \le V_M^{up,i}, \qquad \forall i \in \mathcal{M} \tag{56}$$

$$V_W^{low,k} \le v_W^k \le V_W^{up,k}, \qquad \forall k \in \mathcal{W} \tag{57}$$

where the decision variables are $v_M^i, \forall i \in \mathcal{M}$ and $v_W^k, \forall k \in \mathcal{W}$ and $\bar{p}_M^i = \frac{\Xi_i}{\gamma_i} p_M^i$ and $\bar{p}_W^k = -\frac{F_k}{G_k} T p_W^k$.

Proof. Since in any optimal solution of (P1) Constraints (14), (15) and (16) hold as equalities (Theorem 3.1 and Theorem 3.2), they allow us to establish the relations among the optimal values of the different decision variables of problem (P1). According to Theorem 3.1, the optimal values of s_M^i and s_R^i can be expressed as function of h_i, where the latter, according to Theorem 3.2, can be expressed in terms of v_M^i. Along the same lines, the optimal values of x_k are expressed in terms of v_W^k (Theorem 3.2). As a result, the decision variables of problem (P2) are v_M^i and v_W^k.

Constraints (56) are obtained translating the bounds on the concurrency level of class i MR jobs, $\forall i \in \mathcal{M}$ (Constraints (19) of problem (P1)) into bounds on the number of the VMs allocated to the jobs of that class as follows:

$$V_M^{low,i} = \frac{\gamma_i}{\Xi_i} H_i^{low}, \qquad \forall i \in \mathcal{M}, \tag{58}$$

$$V_M^{up,i} = \frac{\gamma_i}{\Xi_i} H_i^{up}, \qquad \forall i \in \mathcal{M}. \tag{59}$$

Analogously, Constraints (57) are obtained translating the bounds on the throughput of class k WS application requests, $\forall k \in \mathcal{W}$ (Constraints (20) of problem (P1)) into bounds on the number of VMs allocated to the requests of that class:

$$V_W^{low,k} = -\frac{G_k}{F_k}\Lambda_k^{low}, \qquad \forall k \in \mathcal{W}, \tag{60}$$

$$V_W^{up,k} = -\frac{G_k}{F_k}\Lambda_k^{up}, \qquad \forall k \in \mathcal{W}. \tag{61}$$

Finally, since Constraints (16) of problem (P2) hold as equalities (Theorem 3.2) we have that:

$$\frac{x_k}{v_W^k} = -\frac{F_k}{G_k} = \frac{\mu_k G_k - 1}{G_k} = \mu_k - \frac{1}{G_k} < \mu_k \implies x_k < u_k v_W^k.$$

Therefore, Constraints (17) become redundant and are not present in the reduced problem (P2) as they are implied by Constraints (16). □

Remark 2. While γ_i represents the number of VMs needed to serve one MR job of class i, $1/\gamma_i$, represents instead the number of MR jobs of class i served by one VM. We recall that $\Xi_i = T/\overline{D}_i$ indicates how many times during T a VM can be used for executing class i jobs as we solve the AC&CA problem every T seconds whereas jobs are executed in at most $\overline{D}_i$ seconds; if we did not account for Ξ_i, VMs could remain idle in the remaining period $T - \overline{D}_i$.
Remark 3. $\bar{p}_M^i$ represents the penalty cost for not activating a VM to serve class i MR jobs. Similarly, $\bar{p}_W^k$ represent the penalty cost for not activating a VM to serve class k WS application requests.
Remark 4. Being $(\widetilde{v}_M^i, \widetilde{v}_W^k)$ the optimal solution of the problem (P2), the optimal solution of problem (P0), $(\widetilde{h}_i, \widetilde{s}_M^i, \widetilde{s}_R^i, \widetilde{v}_M^i, \widetilde{x}_k, \widetilde{v}_W^k)$ can be derived as follows:

$$\widetilde{h}_i = \frac{\Xi_i}{\gamma_i}\widetilde{v}_M^i, \qquad \forall i \in \mathcal{M}, \tag{62}$$

$$\widetilde{s}_M^i = -\frac{\Xi_i}{\gamma_i}\frac{1}{E_i}\left(A_i + \sqrt{\frac{A_i B_i c_M^i}{c_R^i}}\right)\widetilde{v}_M^i, \qquad \forall i \in \mathcal{M}, \tag{63}$$

$$\widetilde{s}_R^i = -\frac{\Xi_i}{\gamma_i}\frac{1}{E_i}\left(B_i + \sqrt{\frac{A_i B_i c_R^i}{c_M^i}}\right)\widetilde{v}_M^i, \qquad \forall i \in \mathcal{M}, \tag{64}$$

$$\widetilde{x}_k = -\frac{F_k}{G_k}\widetilde{v}_W^k, \qquad \forall k \in \mathcal{W}. \tag{65}$$

3.3. Problem Properties

Similarly to Section 3.2, we derive several proprieties characterizing the optimal solutions of problem (P2) by applying the KKT conditions. Given the cluster size $\overline{V}$, the minimum/maximum demand of each application class ($V_M^{low,i}/V_M^{up,i}$ for the MR and $V_W^{low,k}/\ V_W^{up,k}$ for the WS) and the penalties per application class, $\bar{p}_M^i$ and $\bar{p}_W^k$, we can determine whether the cluster size is large enough to fully satisfy the demand of all classes. In conditions of limited resources, we show how classes with higher penalty costs are prioritized over lower penalty ones, which results in request rejections for the latter.

Let us denote with $\mathcal{M}_1$ the subset of $\mathcal{M}$ consisting of MR job classes for which the cost for operating a VM during T is larger than the penalty for not activating it [7], i.e., $\mathcal{M}_1 = \{i \in \mathcal{M} | \rho > \bar{p}^i_M\}$ whereas $\mathcal{M}_2 = \mathcal{M} \setminus \mathcal{M}_1$. We partition $\mathcal{W}$ in the same fashion: $\mathcal{W}_1 = \{k \in \mathcal{W} | \rho > \bar{p}^k_W\}$ and $\mathcal{W}_2 = \mathcal{W} \setminus \mathcal{W}_1$. Let $\mathcal{U}_1 = \mathcal{M}_1 \cup \mathcal{W}_1$ and $\mathcal{U}_2 = \mathcal{M}_2 \cup \mathcal{W}_2$. To keep the notation light, when we refer to elements of $\mathcal{U}_1$ and $\mathcal{U}_2$ we drop the subscript M and W indicating the application type (i.e., MR or WS, respectively) in all parameters and variables.

The intuition behind the problem proprieties introduced in this section is the following: since allocating a VM to serve requests of a class $u \in \mathcal{U}_2$ reduces the objective function value by $\rho - \bar{p}_u$ (as $\rho - \bar{p}_u \leq 0$), while, vice versa, allocating it to a class $r \in \mathcal{U}_1$ increases its value by $\rho - \bar{p}_r$ (as $\rho - \bar{p}_r > 0$), requests of class u are more likely to be admitted (given the VM availability) while class r will be served to the minimum demand.

Remark 5. The following proprieties concern feasible instances of the proposed AC&CA problem, that is, instances for which $\sum_{i \in \mathcal{M}} V^{low,i}_M + \sum_{k \in \mathcal{W}} V^{low,k}_W \leq \overline{V}$. In other words, the cluster size should be at least large enough to serve all MR and WS classes at the minimum expected concurrency level and throughput, respectively.

Theorem 3.4. *Any optimal solution of (P2), ($\widetilde{v}^i_M$,$\widetilde{v}^k_W$), satisfies the following:*

i) MR jobs of class $i \in \mathcal{M}_1$ and WS application requests of class $k \in \mathcal{W}_1$ are served to the minimum concurrency level and throughput, respectively:

- *a. $\widetilde{v}^i_M = V^{low,i}_M$, i.e., $\widetilde{h}_i = H^{low}_i$, $\forall i \in \mathcal{M}_1$,*
- *b. $\widetilde{v}^k_W = V^{low,k}_W$, i.e., $\widetilde{x}_k = \Lambda^{low}_k$, $\forall k \in \mathcal{W}_1$.*

ii) If $\sum\limits_{i \in \mathcal{M}_1} V^{low,i}_M + \sum\limits_{k \in \mathcal{W}_1} V^{low,k}_W + \sum\limits_{i \in \mathcal{M}_2} V^{up,i}_M + \sum\limits_{k \in \mathcal{W}_2} V^{up,k}_W < \overline{V}$, then no MR jobs from classes in $\mathcal{M}_2$ and no WS requests from classes in $\mathcal{W}_2$ will be rejected:

- *a. $\widetilde{v}^i_M = V^{up,i}_M$, i.e., $\widetilde{h}_i = H^{up}_i$, $\forall i \in \mathcal{M}_2$,*
- *b. $\widetilde{v}^k_W = V^{up,k}_W$, i.e., $\widetilde{x}_k = \Lambda^{up}_k, \forall k \in \mathcal{W}_2$.*

iii) Instead, if $\mathcal{U}_2 \neq \emptyset$ and $\sum\limits_{i \in \mathcal{M}_1} V^{low,i}_M + \sum\limits_{k \in \mathcal{W}_1} V^{low,k}_W + \sum\limits_{i \in \mathcal{M}_2} V^{up,i}_M + \sum\limits_{k \in \mathcal{W}_2} V^{up,k}_W > \overline{V}$, there will be at least one class from $\mathcal{U}_2$, that is, from either $\mathcal{M}_2$ or $\mathcal{W}_2$, whose requests will be rejected.

Let r be a class such that $r \in \mathcal{U}_2$ and $V^{low}_r < \widetilde{v}_r < V^{up}_r$, then:

- *a. $\widetilde{v}_l = V^{low}_l$, $\forall\, l \in \mathcal{U}_2 \mid l \neq r, \bar{p}_l < \bar{p}_r$,*
- *b. $\widetilde{v}_q = V^{up}_q$, $\forall\, q \in \mathcal{U}_2 \mid q \neq r, \bar{p}_q > \bar{p}_r$.*

[7]Notice that such subdivision of the application classes into sets is done in order to account also for particular scenarios, e.g., when operational (energy costs) may vary over time whereas penalties are fixed. Instead, for the instances we have considered, penalty costs are always larger w.r.t. the VM operational cost. Thus, $\mathcal{M}_1$ and $\mathcal{W}_1$ are empty sets.

Proof. The Lagrangian function of problem (P2), $\mathcal{L}_{(\mathrm{P2})}\left(v_M^i, v_W^k\right)$, is:

$$\begin{aligned}\mathcal{L}_{(\mathrm{P2})} \;=\; & \sum_{i\in\mathcal{M}}\left(\rho-\bar{p}_M^i\right)v_M^i+\sum_{k\in\mathcal{W}}\left(\rho-\bar{p}_W^k\right)v_W^k \\ & +\xi\left(\sum_{i\in\mathcal{M}}v_M^i+\sum_{k\in\mathcal{W}}v_W^k-\overline{V}\right) \\ & +\sum_{i\in\mathcal{M}}\lambda_i^{(1)}\left(V_M^{low,i}-v_M^i\right)+\sum_{i\in\mathcal{M}}\lambda_i^{(2)}\left(v_M^i-V_M^{up,i}\right) \\ & +\sum_{k\in\mathcal{W}}\nu_k^{(1)}\left(V_W^{low,k}-v_W^k\right)+\sum_{k\in\mathcal{W}}\nu_k^{(2)}\left(v_W^k-V_W^{up,k}\right).\end{aligned}$$

The KKT conditions associated with problem (P2) can be written as:

$$\rho-\bar{p}_M^i-\lambda_i^{(1)}+\lambda_i^{(2)}+\xi=0, \qquad \forall i\in\mathcal{M}, \tag{66}$$

$$\rho-\bar{p}_W^k-\nu_k^{(1)}+\nu_k^{(2)}+\xi=0, \qquad \forall i\in\mathcal{M}, \tag{67}$$

whereas the CS constraints are:

$$\lambda_i^{(1)}(V_M^{low,i}-v_M^i)=0,\;\; \lambda_i^{(1)}\geq 0, \qquad \forall i\in\mathcal{M}, \tag{68}$$

$$\lambda_i^{(2)}(v_M^i-V_M^{up,i})=0,\;\; \lambda_i^{(2)}\geq 0, \qquad \forall i\in\mathcal{M}, \tag{69}$$

$$\nu_k^{(1)}(V_W^{low,k}-v_W^k)=0,\;\; \nu_k^{(1)}\geq 0, \qquad \forall k\in\mathcal{W}, \tag{70}$$

$$\nu_k^{(2)}(v_W^k-V_W^{up,k})=0,\;\; \nu_k^{(2)}\geq 0, \qquad \forall k\in\mathcal{W}, \tag{71}$$

$$\xi(\sum_{i\in\mathcal{M}}v_M^i+\sum_{k\in\mathcal{W}}v_W^k-\overline{V})=0,\;\; \xi\geq 0. \tag{72}$$

Proof of propriety (i):

a. From the KKT condition (66), $\lambda_i^{(1)}=\lambda_i^{(2)}+\xi+\rho-\bar{p}_M^i$. Since $\lambda_i^{(2)}\geq 0$, $\xi\geq 0$ and $\rho>\bar{p}_M^i, \forall i\in\mathcal{M}_1 \implies \lambda_i^{(1)}>0$. As the optimal solution $\widetilde{v}_M^i$ should satisfy the CS constraint (68), we have that $\widetilde{v}_M^i=V_M^{low,i}$.

b. Similarly, from the KKT condition (67), $\nu_i^{(1)}=\nu_i^{(2)}+\xi+\rho-\bar{p}_W^k$. Since $\nu_k^{(2)}\geq 0$, $\xi\geq 0$ and $\rho>\bar{p}_W^k, \forall k\in\mathcal{W}_1, \implies \nu_k^{(1)}>0$. For $\widetilde{v}_W^k$ to satisfy the CS constraint (70), then $\widetilde{v}_W^k=V_W^{low,k}$.

Proof of propriety (ii): If the cluster size is strictly larger than the number of VMs needed to satisfy the minimum demand of classes in $\mathcal{M}_1$ and $\mathcal{W}_1$ (which is optimal due to propriety (i)) and not to reject any MR/WS request from classes of $\mathcal{M}_2$ and $\mathcal{W}_2$, respectively, in any optimal solution, Constraints (55) will be strict inequalities. As a result, due to the CS constraints (72), ξ should be equal to 0.

a. From the KKT condition (66), $\lambda_i^{(2)}=\lambda_i^{(1)}-\xi+\bar{p}_M^i-\rho=\lambda_i^{(1)}+\bar{p}_M^i-\rho$. Being $\lambda_i^{(1)}\geq 0$ and $\forall i\in\mathcal{M}_2\,|\,\bar{p}_M^i>\rho \implies \lambda_i^{(2)}>0$. Thus, because of Constraint (69), we have $\widetilde{v}_M^i=V_M^{up,i}$, $\forall i\in\mathcal{M}_2$. Instead, for the particular case when $\bar{p}_M^i=\rho$, we have that $\lambda_i^{(2)}=\lambda_i^{(1)}$: it cannot be that $\lambda_i^{(2)}=\lambda_i^{(1)}>0$, as both the lower and upper bound on $\widetilde{v}_M^i$ would have to be satisfied (for Constraints (68), (69) to hold). Finally, $\lambda_i^{(2)}=\lambda_i^{(1)}=0$, which means that any number of VMs in the range within the bounds is optimal[8], i.e., $V_M^{low,i} \leq \widetilde{v}_M^i \leq V_M^{up,i}$.

b. Analogous to point a.

Proof of propriety (iii):

a. Combining the KKT conditions (66) for classes r and l we have that: $\lambda_l^{(1)} = \bar{p}_r - \bar{p}_l + \lambda_l^{(2)} + \lambda_r^{(1)} - \lambda_r^{(2)}$. Since $\widetilde{v}_r < V_r^{up}$, due to Constraints (69)/(71), $\lambda_r^{(2)}$ are equal to zero. Being $\lambda_l^{(2)} \geq 0$, $\lambda_r^{(1)} \geq 0$ and $\bar{p}_r > \bar{p}_l \implies \lambda_l^{(1)} > 0$, thus, from Constraints (68)/(70) we have that $\widetilde{v}_l = V_l^{low}$.

b. Similarly to point a., we have that: $\lambda_q^{(2)} = \bar{p}_q - \bar{p}_r + \lambda_q^{(1)} + \lambda_r^{(2)} - \lambda_r^{(1)}$. Since $\widetilde{v}_r > V_r^{low}$, due to Constraints (68)/(70), $\lambda_r^{(1)} = 0$. Thus, being $\bar{p}_q > \bar{p}_r$ and $\lambda_q^{(1)} \geq 0$, $\lambda_r^{(2)} \geq 0 \implies \lambda_q^{(2)} > 0$. As a result, for Constraints (69)/(71) to hold, we should have $\widetilde{v}_q = V_q^{up}$.

$\square$

Based on these proprieties, we propose a greedy algorithm that solves problem (P2) to optimality (see *Proposition 1* in Appendix C of [27]).

Algorithm 1 Greedy procedure

```
 1: if $\overline{V} < \left(\sum_{i\in\mathcal{M}} V_M^{low,i} + \sum_{k\in\mathcal{W}} V_W^{low,k}\right)$ then
 2:     End
 3: else
 4:     $\widetilde{v}_M^i \leftarrow V_M^{low,i}, \forall i \in \mathcal{M}$
 5:     $\widetilde{v}_W^k \leftarrow V_W^{low,k}, \forall k \in \mathcal{W}$
 6:     $V \leftarrow \overline{V} - \left(\sum_{i\in\mathcal{M}} V_M^{low,i} + \sum_{k\in\mathcal{W}} V_W^{low,k}\right)$
 7:     $\mathcal{R} \leftarrow \mathcal{U}_2$
 8: while ($V > 0$ & $\mathcal{R} \neq \emptyset$) do
 9:     $\bar{q} \leftarrow \operatorname{argmax}_{q\in\mathcal{R}} \bar{p}_q$
10:     $\mathcal{R} \leftarrow \mathcal{R} \setminus \{\bar{q}\}$
11:     if $V > (V_{\bar{q}}^{up} - V_{\bar{q}}^{low})$ then
12:         $\widetilde{v}_{\bar{q}} \leftarrow V_{\bar{q}}^{up}$
13:         $V \leftarrow V - (V_{\bar{q}}^{up} - V_{\bar{q}}^{low})$
14:     else
15:         $\widetilde{v}_{\bar{q}} \leftarrow \widetilde{v}_{\bar{q}} + V$
16:         End
17: End
```

The algorithm starts with a feasibility check (*Step 1*): if the given cluster size, i.e., number of VMs, cannot satisfy the minimum demand of all MR and WS classes, then the procedure is terminated (*Step 2*); otherwise, the minimum number of VMs is allocated to each class (*Steps 4-5*) and the residual number of VMs (V) is calculated (*Step 6*). A set $\mathcal{R}$ containing all elements of set $\mathcal{U}_2$ is defined in *Step 7*. The procedure then enters a while loop in *Steps 8-16* which is repeated as long as there are still VMs that have not been allocated yet ($V > 0$) and MR/WS classes with

[8]This is quite intuitive since classes for which $\rho = \bar{p}_M^i$ have zero contribute in the objective function value.

penalty cost larger than operational cost ($\mathcal{R} \neq \emptyset$). In the while loop body, the current MR/WS class with the largest penalty cost ($\bar{q}$) is selected (*Step 9*) and removed from set $\mathcal{R}$ (*Step 10*). If the current number of available VMs is large enough to fully satisfy all the demand of class $\bar{q}$ (*Step 11*), then all requested VMs are assigned to it (*Step 12*), the number of available VMs is updated (*Step 13*) and then the procedure moves on to the next largest penalty cost class from set $\mathcal{U}_2$ (if any). Otherwise, all available VMs are assigned to application class $\bar{q}$ (*Step 15*) and the procedure terminates (*Step 16*).

Moreover, if instead of considering the exact[9] minimum and maximum resource demands of each MR and WS class as obtained in closed form (Theorem 3.2), which are not necessarily integer values, we consider the corresponding upperbounds, that is, $\lceil V_M^{low,i} \rceil$ and $\lceil V_M^{up,i} \rceil$, $\forall i \in \mathcal{M}$ and $\lceil V_W^{low,k} \rceil$ and $\lceil V_W^{up,k} \rceil$ $\forall k \in \mathcal{W}$, the proposed greedy procedure provides an integer solution since, for such instances, problem (P2) has the integrality propriety (see *Proposition 2* in Appendix C of [27]).

4. EXPERIMENTAL RESULTS

In this section we summarize the results of the tests that we have performed to analyze several aspects of the proposed framework. The section is organized as follows: Section 4.1 describes the tests environment and the parameter setting for the different considered instances. In Section 4.2, we investigate how the total cluster cost is affected by the cluster size for several scenarios and instance sizes. Section 4.3 focuses on a particular setting and provides a detailed analysis of the cluster cost and of the utilization.

4.1. Experimental setting

Tests were run on an Ubuntu 14.04.3 LTS VM hosted on an Intel Xeon E5530 @2.4GHz system with 6 GB RAM and 4 cores. The optimization models have been implemented in AMPL [34]. We have solved the reduced problem (P2) with CPLEX 10.1 [35] whereas the convex problem (P1) with KNITRO 9.0.1 [36].

We have set up a large set of randomly generated instances; their parameter setting is explained in detail in Section 4.1.1.

4.1.1. Problem instances

The AC&CA problem is solved every hour (T=3600s)[10] within a day [37, 17, 38, 39]. The VM cost ρ accounts for the energy cost related to the operation of the physical servers hosting the VMs,

[9]For the demands obtained in closed form according to Theorem 3.2 (which are not necessarily integer values) we have also solved problem (P2) forcing variables r_M^i and r_W^k to be integer. For all considered instances, it results that imposing integrality increases the cluster cost by no more than 4% w.r.t. to solving (P2) in continuous variables (see Section 4.4. in [27]).

[10]We remark that the time to solve the problem under either formulation (P1) or (P2) is below 10 seconds for all considered instances (see Section 4.5. in [27]).

the overhead energy cost due to cluster maintenance (mainly related to server room cooling) and the price of physical servers [40]. The VM unit cost per hour ρ (€ cents) is then calculated as:

$$\rho = (\mathrm{PUE}\,\varepsilon + S)\frac{c}{d}, \tag{73}$$

where ε is an estimate of the energy consumption cost of one hour of operation of a single core server which is then multiplied by the PUE (power usage effectiveness) coefficient to account for indirect energy costs (such as cooling, lighting etc.), S represents the unit (per hour) price of a physical server (obtained depreciating the price of a physical server over 4 years lifetime), c is the number of virtual cores per VM whereas d is the virtual-to-physical core ratio.

The parameters related to the different classes of MR and WS applications are set as follows. We have used realistic job profiles extracted from MapReduce execution logs considering the data reported in [14] and in [41] which considered the TPC-DS benchmark [42], widely used today to analyze data warehouse performance. The job profile parameter values are reported in Table 3, whereas A_i, B_i, C_i are derived from the latter as explained in Appendix A of [27]. Instead, deadlines of each MR class ($\overline{D}_i$) are set to values in the range 15 to 25 minutes [40].

The parameters of the M/G/1 queue model μ_k and L_k of each WS application class k are randomly extracted from the uniform distributions in the ranges reported in Table 5a as in [39, 17, 43, 44, 45]. The maximum response time $\overline{R}_k$ is then set equal to $10/\mu_k + 1.5 \max_{k\in\mathcal{W}} L_k$.

We have used real-life traces for the distribution of WS applications requests throughout a day which were obtained from a large anonymous website. The original workload traces consist of 10 minutes samples logs for 12 days. The workload is periodic and follows a bimodal distribution with two peaks around 11am (11:00) and 4pm (16:00). We normalize the samples of each daily trace with respect to the daily peak and scale them using a random multiplier extracted from the uniform distribution in the range (1000,2000). Thus, using different daily traces and different random multipliers, we obtain distinct workload intensities for each WS application class. An hourly prediction for the request rate of each WS class (Λ_k^{up}) is obtained averaging over the 6 available samples per hour and then applying white noise as in [39, 45, 43]. In order to limit request rejections, we set $\Lambda_k^{low} = 0.8\Lambda_k^{up}$. Hourly predictions for the submitted concurrency level of the MR classes (H_i^{up}) are obtained in the same fashion; random multipliers, extracted from the uniform distribution in the range (5,20), are used for scaling the normalized samples of the original trace to which white noise is added. To avoid job starvation, we also set $H_i^{low} = \lceil 0.8\, H_i^{up} \rceil$.

In order to account for reasonable values for the penalty costs, we calculate the cost for executing a single MR job of class i, c_{mr}^i, setting $H_i^{low} = H_i^{up}$ and thus deactivating the admission control mechanism (i.e., setting $V_M^{low,i} = V_M^{up,i}$ in problem (P2)). Being $\rho V_M^{low,i}$ the cost of allocating $V_M^{low,i}$ VMs, which in turn allows to execute $H_i^{low} = (\Xi_i/\gamma_i)V_M^{low,i}$ class i concurrent jobs (see Theorem 3.2), c_{mr}^i is then equal to $(\gamma_i/\Xi_i)\rho$. Finally, we set p_M^i to random values extracted from the uniform distribution in the range $10\rho\,(\min_{i\in\mathcal{M}} \gamma_i/\Xi_i, \max_{i\in\mathcal{M}} \gamma_i/\Xi_i)$ so that penalties are at least one order of magnitude larger w.r.t. the operational cost ([46]). Along the same lines, being the cost for executing a single WS request, c_{ws}^k, equal to $(-G_k/(T\,F_k))\,\rho$, the respective penalties p_W^k are set to random values from the uniform distribution in range $10\rho\,(\min_{k\in\mathcal{W}} -G_k/(T\,F_k), \max_{k\in\mathcal{W}} -G_k/(T\,F_k))$.

N_M^i	N_R^i	M_{avg}^i (s)	M_{max}^i (s)	$Sh_{max}^{1,i}$ (s)	$Sh_{avg}^{typ,i}$ (s)	$Sh_{max}^{typ,i}$ (s)	R_{avg}^i (s)	R_{max}^i (s)
370	64	30	42	11	37	40	22	44
1024	64	5	16	13	30	50	53	75
168	64	34	40	11	24	30	11	14
425	64	99	120	27	115	142	26	34

Table 3: MapReduce job profiles

We have generated two sets of instances. For all instances of the first set, we consider 4 MR user classes (one for each job profile reported in Table 3) and 5 WS ones whereas the remaining parameters are set as reported in Table 5a. To investigate the impact of the time zone diversity, we have set up 5 families of instances within the first set, denoted by S1, S2, S3, S4 and S5; each family represents a scenario in which requests for applications of different classes originate from different time zones as shown in Table 4 (note that this can be easily achieved for large e-commerce sites, e.g. the Amazon infrastructure is spread on 35 data centers world-wide).

The original workload traces, which represent the reference time zone, are shifted appropriately to reflect the correct daily pattern of request arrivals from different time zones.

	MR				WS				
	m_1	m_2	m_3	m_4	w_1	w_2	w_3	w_4	w_5
S1	0	0	0	0	0	0	0	0	0
S2	0	+1	+3	-6	0	0	+1	+3	-6
S3	0	-6	-6	-6	0	0	0	0	-6
S4	-6	-6	-6	-6	0	0	0	0	0
S5	-9	-9	-9	-9	0	0	0	0	0

Table 4: Shifts in hours of the time zone of each application class w.r.t. the reference time zone for instances S1–S5

An example of generating workload predictions is provided in Figure 2 which illustrates an *original* workload trace for one day (24 hours) and the corresponding *normalized*, *scaled* and *shifted* workload trace that represents requests for a WS application class from a time zone 3 hours ahead the reference one.

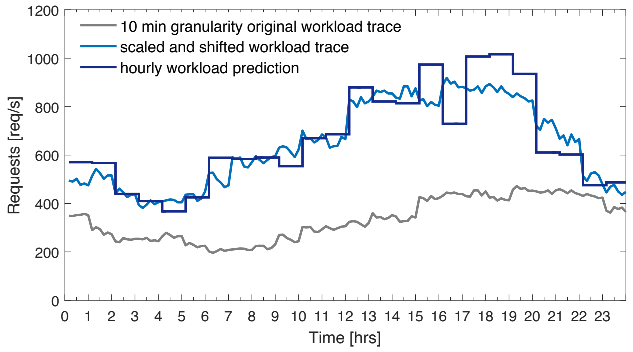


Figure 2: Example of a workload prediction from a time zone 3 hours ahead w.r.t. the original trace

For the second set of instances, we vary $|\mathcal{M}|$ and $|\mathcal{W}|$ from 100 to 500 with a step of 100. We denote the families of these instances as L1 ($|\mathcal{M}|$=$|\mathcal{W}|$=100), L2 ($|\mathcal{M}|$=$|\mathcal{W}|$=200), L3 ($|\mathcal{M}|$=$|\mathcal{W}|$=300), L4 ($|\mathcal{M}|$=$|\mathcal{W}|$=400) and L5 ($|\mathcal{M}|$=$|\mathcal{W}|$=500). The MR job profiles for instances L1–L5 are randomly generated as reported in Table 5b using significant ranges obtained from the 4 original job profiles (Table 3). The cluster cost parameter values are set as in Table 5b, whereas the remaining parameters as described in Section 4.1.1. For all instances of the second set, we consider a 9 hour time shift between all user classes of the two application types, in order to represent an extreme scenario in which all WS application requests are submitted from a time zone 9 hours ahead w.r.t. to the MR ones.

WS		MR	
L_k	(0.01, 0.5) [s]	$\overline{D}_i$	{900, 1100, 1300, 1500} [s]
μ_k	(10,20) [requests/s]	c_M^i, c_R^i	2

Cluster	
PUE	1.9
ε	0.0669 [€ cents/hour]
S	2.0615 [€ cents/hour]
c	2
d	4
ρ	1.094 [€ cents/hour]
T	3600 [s]

(a) Instances S1–S5

MR job profiles		Cluster	
N_M^i	(70, 1120)		
N_R^i	(64, 64)	ε	(0.06008, 0.06690)[€ cents/hour]
M_{max}^i (s)	(16, 120)	S	2.0615 [€ cents/hour]
$Sh_{max}^{typ,i}$ (s)	(30, 150)	PUE	(1.2, 2.2)
R_{max}^i (s)	(15, 75)	c	2
$Sh_{max}^{1,i}$ (s)	(10, 30)	d	(3, 5)
c_M^i , c_R^i	(1,4)		

(b) Instances L1–L5

Table 5: Parameter setting

4.1.2. Resource management approaches

As mentioned, we solve the AC&CA problem for each hour within a day. The cluster size ($\overline{V}$) is fixed throughout the 24 hours span whereas the per-hour prediction of the MR job concurrency level (H_i^{up}) and rate of WS requests (Λ_k^{up}) vary as explained in Section 4.1.1.

We investigate two resource management approaches: (i) a *pre-partitioned cluster* approach, for which we *a priori* partition the cluster into a sub-cluster for MR applications (consisting of $\overline{V}_{\text{MR}}$ VMs) and a sub-cluster for WS applications (consisting of the remaining $\overline{V}_{\text{WS}} = \overline{V} - \overline{V}_{\text{MR}}$ VMs) and (ii) a *shared cluster* approach, for which MR and WS requests can be assigned to any of the $\overline{V}$ VMs in the cluster.

For the *pre-partioned cluster* approach, we split the $\overline{V}$ VMs of the cluster among the MR and the WS applications proportionally to their respective daily peak demands, that is,

$$\overline{V}_{\text{MR}} = \frac{\max\limits_{t\in 1\ldots 24} \sum\limits_{i\in\mathcal{M}} V_M^{up,i,t}}{\max\limits_{t\in 1\ldots 24} \sum\limits_{i\in\mathcal{M}} V_M^{up,i,t} + \max\limits_{t\in 1\ldots 24} \sum\limits_{k\in\mathcal{W}} V_W^{up,k,t}} \overline{V},$$

$$\overline{V}_{\text{WS}} = \frac{\max\limits_{t\in 1\ldots 24} \sum\limits_{k\in\mathcal{W}} V_W^{up,k,t}}{\max\limits_{t\in 1\ldots 24} \sum\limits_{k\in\mathcal{W}} V_W^{up,k,t} + \max\limits_{t\in 1\ldots 24} \sum\limits_{i\in\mathcal{M}} V_M^{up,i,t}} \overline{V}.$$

4.2. Cluster size analysis

In this section, we compare the total cost of the *shared cluster* and *pre-partitioned cluster* approaches when varying the cluster size $\overline{V}$.

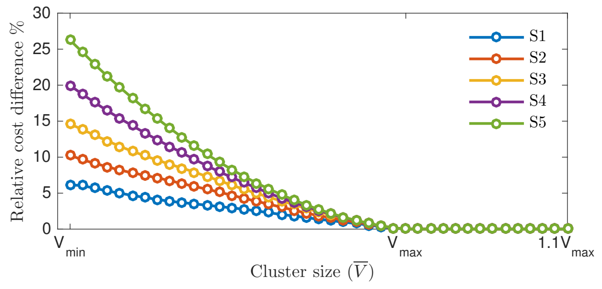
Let $V_{\max}$ be the minimum number of VMs needed to have no rejections under the *pre-partitioned cluster* approach, that is, $V_{\max} = V_{\text{WS}}^{\max} + V_{\text{MR}}^{\max}$ and $V_{\min} = V_{\text{MR}}^{\min} + V_{\text{WS}}^{\min}$ be the minimum number of VMs for the AC&CA problem to be feasible for both the MR and WS sub-clusters. We vary the cluster size $\overline{V}$ from $1.1V_{\max}$ to $V_{\min}$ decreasing it in 40 steps by the same number of VMs $\Delta\overline{V}=(1.1V_{\max} - V_{\min})/40$. The size of the MR and WS sub-clusters are decreased accordingly, that is, $\overline{V}_{\text{MR}}$ is decreased from $1.1V_{\text{MR}}^{\max}$ to $V_{\text{MR}}^{\min}$ by $\Delta\overline{V}_{\text{MR}}=\left(1.1V_{\text{MR}}^{\max} - V_{\text{MR}}^{\min}\right)/40$ at each step and analogously for $\overline{V}_{\text{WS}}$. For each cluster size, we then solve the AC&CA problem with both approaches. Notice that, for each cluster size, under the *pre-partitioned cluster* approach, we solve problem (P2) twice, that is, once for the MR sub-cluster (setting $\overline{V}$ in (P2) to $\overline{V}_{\text{MR}}$ and ignoring WS applications) and once for the WS sub-cluster (setting $\overline{V}$ in (P2) to $\overline{V}_{\text{WS}}$ and ignoring MR applications). Thus, the total cost of the *pre-partitioned cluster* approach is obtained as the sum of the total cost of the two sub-clusters.

We generate 30 random instances for each family/scenario S1–S5 and for each instance and cluster size calculate the relative difference between the total cost of the *pre-partitioned cluster* and *shared cluster* approaches. The average relative difference (over 30 instances) as a function of $\overline{V}$ has been plotted in Figure 3a for each scenario. We can observe a similar trend for all scenarios: as long as the cluster size is large enough not to reject any requests ($\overline{V}$ in the range $\overline{V}_{\max}$ to $1.1V_{\max}$), both approaches incur the same total cost (represented by a 0% relative difference) as no penalty costs are incurred; however, for $\overline{V} < V_{\max}$, if we read this plot from right to left, we observe an increasing relative cost difference with the decrease of the cluster size. Thus, the smaller the cluster size (and as a result the larger the rejection rate/penalty costs), the larger the benefit of the *shared cluster* approach.

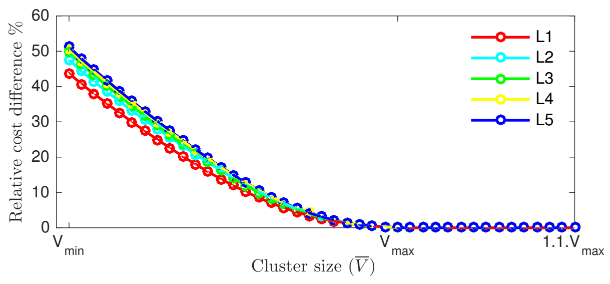
Further, we can assess the impact of the time zone diversity in the cost difference of the two approaches: the *shared cluster* approach provides the smallest gain (up to 5%) in case of no time zone diversity (i.e., for scenario S1 in which all MR and WS requests originate from the same time zone) while such gain goes up to 25% for scenario S5 which represents the largest time zone shift (i.e., 9 hours between all MR and all WS requests).

Figure 3b illustrates the relative cost difference between the two approaches as a function of $\overline{V}$ for the large scale families of instances L1–L5 ($|\mathcal{M}|=|\mathcal{W}|=100\ldots500$); the reported values are obtained as averages over 10 random instances per family. We can see how the overall trend is very similar to the one observed in Figure 3a. However, such trend depends little on the instance size (number of classes) and, at the minimum cluster size, cost penalties are reduced by up to 50%. Although L1–L5 have the same time diversity as S5 (i.e., a 9 hour shift between all MR and all WS requests), having an overall larger number of user classes and equal number of user classes for the two application types, allows to further benefit from the time zone diversity and thus decrease the cost.

Note that, even if the results we achieved are limited in case of no time zone diversity between MR and WS workloads, we argue that our approach can be beneficial also under these circumstances if MR and WS peak ratio changes during the year (which is very common if marketing campaigns are implemented at the e-commerce site).



(a) Instances S1–S5



(b) Instances L1–L5

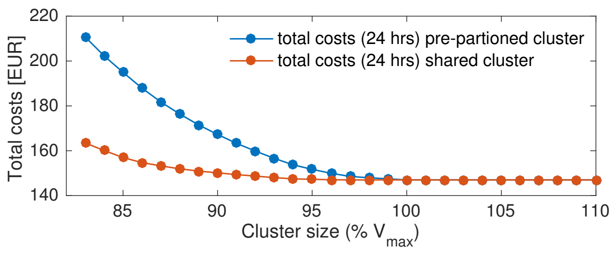
Figure 3: Relative cost difference between *pre-partitioned* and *shared cluster* approaches vs cluster size

4.3. Cluster cost and utilization analysis

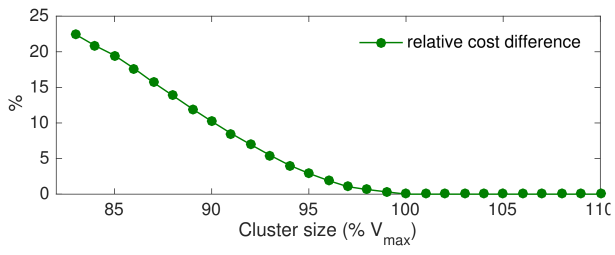
In this section we analyze in detail the cluster cost and utilization for a representative S4 instance, where MR and WS workloads are characterized by a 6 hours shift.

We report the total cost and the relative cost difference as a function of the cluster size for the two resource management approaches in Figure 4a and Figure 4b, respectively. $\overline{V}$ is varied from $1.1V_{\max}$ to $0.83V_{\max}$ so that the AC&CA problem is feasible for both approaches; for the *shared cluster* approach the problem can be feasible for a smaller cluster size w.r.t. to the *pre-partitioned* one since it can exploit unallocated VMs of one sub-cluster for the other.

Assuming we have a fixed budget (i.e., maximum total cost, or alternatively, a maximum level of rejections), from Figure 4a, we can compare the minimum cluster size to guarantee the given level of rejection under both resource management approaches. For instance, fixing the daily budget to 160 €, the cluster size should be at least $0.84V_{\max}$ VMs under the *shared cluster* approach and at least $0.93V_{\max}$ VMs under the *pre-partitioned cluster* approach, that is, the latter requires a 9% larger cluster for the same budget/target level of rejections.



(a)



(b)

Figure 4: Total cost (a) and relative cost difference (b) between *pre-partitioned* and *shared cluster* approaches for increasing cluster size — scenario S4

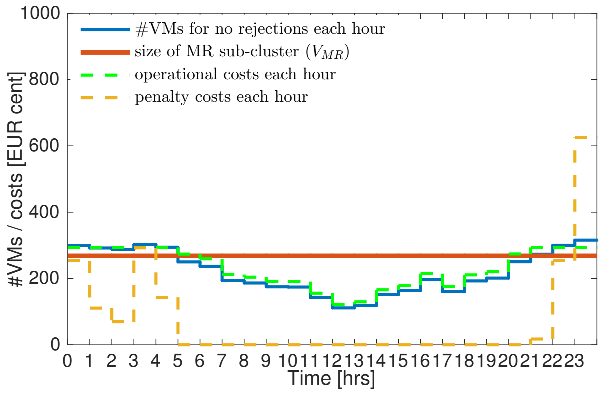
Figure 5 provides a detailed illustration of the AC&CA for each hour within a day for the MR and WS sub-clusters under the *pre-partitioned cluster* approach (Figures 5a, 5b) and for the shared cluster under the *shared cluster* approach (Figure 5c). We set $\overline{V} = 0.85V_{\max}$, which corresponds to a cluster of 667 VMs for the considered instance, represented by the red curve spanning 24 hours in Figure 5c. For the *pre-partitioned cluster* approach, the cluster is split between the MR and the WS applications proportionally to the daily peak which corresponds to 269 VMs for the MR sub-cluster and to 398 VMs for the WS sub-cluster (also represented by red curves in Figure 5a and Figure 5b, respectively). The blue curves in the Figure 5 plots represent the minimum number of VMs that are needed each hour to have no rejections, that is, to satisfy the predicted concurrency level per hour (H_i^{up}) for all MR classes and the request rate per hour (Λ_i^{up}) for all WS classes. Instead, the green curves represent the total operational costs per hour whereas the orange curves, the total penalty costs per hour.

We can see how the hourly operational costs follow the demand (blue curve) as long as the demand is below the cluster size (blue curve below the red one). Instead, the same operational costs are incurred whenever the demand cannot be fully satisfied (blue curve above the red one) as all available VMs of the cluster are being used. In addition, penalty costs are present only in the latter case as part of the requests have to be rejected. Notice that under the *pre-partitioned cluster* approach, some requests have to be rejected (and thus penalties are incurred) during hours 0-4 and 23 for the MR applications (Figure 5a), which can be avoided under the *shared cluster* approach (Figure 5c) exploiting the idle VMs of the WS sub-cluster (Figure 5b). Similarly, for WS request rejections during hours 15-19 (Figure 5b). For the considered instance, the *shared cluster* approach allows to reduce the total cost by 19%.

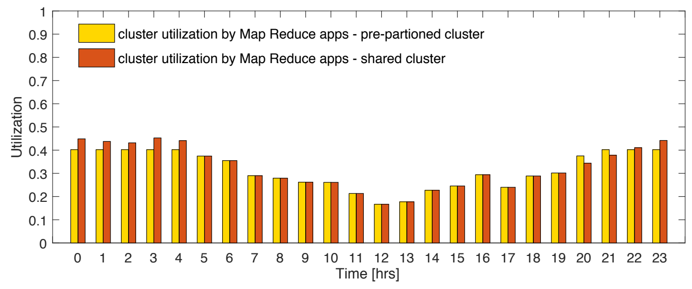
For the same instance (scenario S4, $\overline{V}$=$0.85V_{\max}$), Figure 6 reports the cluster utilization (i.e., the ratio between the number of allocated VMs w.r.t. to the available VMs in the cluster) for all MR (Figure 6a) and WS classes (Figure 6b) and the total utilization (Figure 6c) under both resource management approaches. We can observe a 5% larger cluster utilization under the *shared cluster* approach during hours 0-4 and 23 for the MR applications (Figure 6a) and up to 10% larger for the WS applications during hours 15-19 (Figure 6b). In particular, during hours 20-22, the cluster is fully utilized under the *shared cluster* approach (Figure 6c) and some MR and/or WS requests are still rejected (see Figure 5c); as the AC&CA applies jointly to all MR and WS classes, classes with larger penalties are prioritized to obtain an overall lower cost. This explains the lower cluster utilization of the MR applications during hours 20 and 21 when the cluster is shared compared to when the cluster is partitioned[11].

The largest cluster utilization increase (i.e., for $\overline{V}$=V_{min}) from adopting the *shared* instead of the *pre-partitioned cluster* approach is reported in Table 6a for S1–S5 and in Table 6b for L1–L5 (the values are calculated as an average over 30 random instances per family for S1–S5 and 10 random instances per family for L1–L5). As expected, the largest cluster utilization increase (15%) across scenarios S1–S5 is obtained for S5 having the largest time zone shift (Table 6a). Instead, the

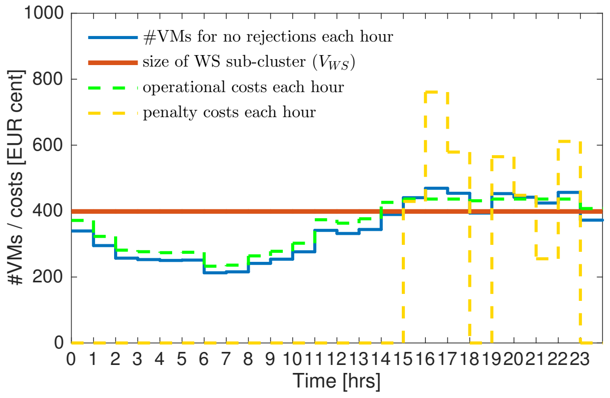
[11] As we are investigating randomly generated instances, this does not mean we are prioritizing WS applications for MR ones; to do so, penalties would have to be set deterministically.



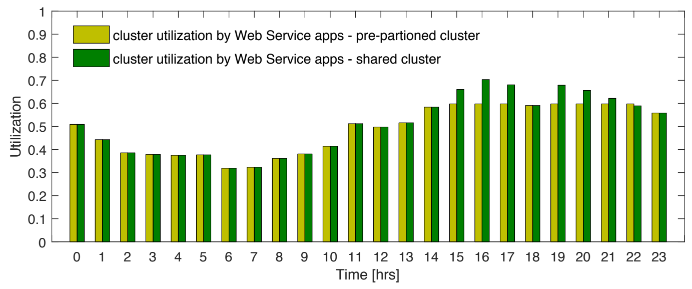
(a) MR sub-cluster



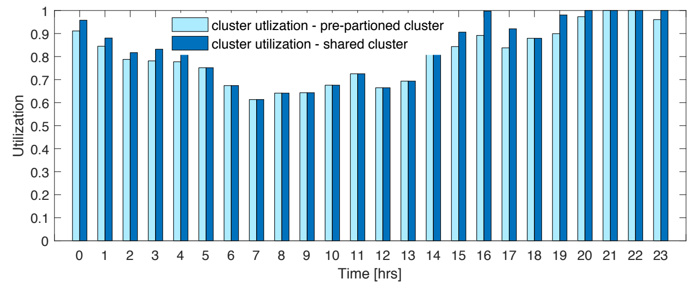
(a) Cluster utilization by MR apps



(b) WS sub-cluster

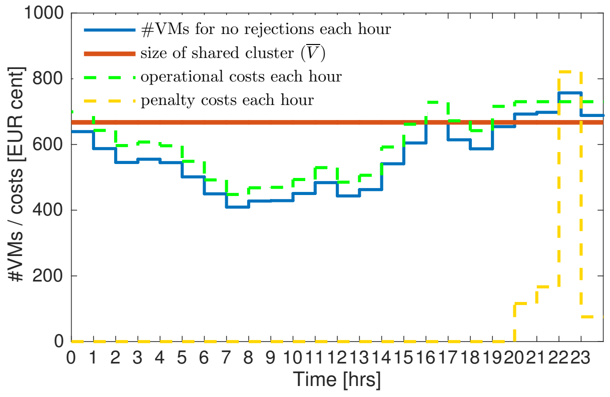


(b) Cluster utilization by WS apps



(c) Total cluster utilization

Figure 6: Cluster utilization — scenario S4, $\overline{V}$ = 85% $V_{\max}$



(c) Shared cluster

Figure 5: Hourly operational and penalty cost — scenario S4, $\overline{V}$ = 85% $V_{\max}$

instance size has little impact on the cluster utilization (Table 6b).

Instance	Cluster utilization increase (%)
S1	3.4340
S2	8.6620
S3	9.6922
S4	12.9158
S5	15.0597

(a) Impact of the time zone diversity

Instance	Cluster utilization increase (%)
L1	17.6642
L2	17.8778
L3	18.2546
L4	18.3204
L5	18.2197

(b) Impact of the number of application classes

Table 6: Largest utilization increase due to the *shared cluster* approach

5. RELATED WORK

The contribution of the research community on QoS models for management of cloud systems is quite vast [9]. A subset of such works focuses on problems of AC and CA, while other application domains, such as load balancing, pricing and energy management, are identified [9]. In our work, we jointly address the problems of AC and CA from the perspective of an e-commerce site hosting its applications on a private cloud. In particular, a joint AC&CA algorithm for Web applications deployed in a virtualized infrastructure is also proposed in [17]. A QoS performance model that allows to account for SLAs between Web application users and virtualized infrastructure providers is embedded in an optimization framework whose goal is the maximization of the provider profit while minimizing the data center energy and penalty costs due to requests rejections and SLA violations.

As far as MR applications are concerned, scheduling and capacity management of Hadoop clusters have also become of particular interest within the research community. The main drivers are the inability of Hadoop schedulers to deal with interactive MapReduce jobs [47] and the paradigm shift from dedicated Hadoop clusters to cloud-based ones [48] which are further shared among several job classes that compete for resources [15]. In [48], authors investigate homogeneous and heterogeneous cloud alternatives for deploying Hadoop clusters at minimum cost while given MR application performance targets, i.e., maximum makespans, have to be met. Differently from our work, a simulation-based framework is proposed, no AC applies and only one application type is considered. We share the MR performance model with [15]. However, in [15] only MR applications are considered and the submitted workload can be hosted by a mixture of reserved and on-demand VMs in a cloud environment, while we consider a fixed-size private cluster.

Aspects similar to our MR performance model are also considered in the ARIA framework [14]. The problem in [14] is to determine the number of slots to allocate to Map and Reduce tasks so that a soft deadline associated with a submitted job is satisfied while over-provisioning costs are reduced. In their performance model, a job execution time estimate is derived from a job profile which reflects the applications characteristics in terms of duration of map, shuffle, sort, and reduce phases. This model has been improved and validated experimentally in [49].

Driven by the need to run multiple computing frameworks (e.g., MapReduce, Dryad, Pregel, etc.) over the same pool of data, [6] proposes Mesos with the aim to avoid costly data replication over multiple framework-dedicated clusters. Mesos introduces a second level scheduler which lies between the individual schedulers of the different framework and the available cluster resources that are offered to the frameworks based on a fair scheme. Similarly to our work, Mesos was benchmarked against a static approach in which the cluster is evenly partitioned among the different frameworks. It is shown to improve cluster utilization, reduce execution times for most types of framework workloads and scale efficiently up to 50,000 emulated nodes.

Similarly to our work, a virtualized cluster shared among interactive (Web applications) and batch (Hadoop jobs but with given deadlines) workloads has been analyzed in [8]. However, the focus of this work is on mitigating the interference introduced from assigning workloads of different characteristics to VMs hosted in the same physical server, while our framework optimizes resource allocation to improve cluster utilization and reduce operational costs. In essence, authors in in [8] improve the performance of the shared cluster by altering the Xen (hypervisor) scheduler to prioritize interactive workloads over batch ones and by adapting the Earliest Deadline First (EDF) Hadoop scheduler to better match jobs to the residual resources of the cluster. They rely on a regression-based model to estimate job execution times as function of the residual capacity with a prediction error below 10%. An admission controller is used to further improve the Hadoop job performance in overload conditions. Such approach improves the throughput of the interactive workload and reduces the number of missed job deadlines despite the variability of the available resources for the latter.

In the context of heterogeneous clusters, in [7], the spare capacity of underutilized clusters running interactive workloads is combined with dedicated nodes for reducing the cost and energy consumption of running data analytics jobs.

6. CONCLUSIONS

We have introduced a novel framework for joint resource management in a fixed-size cloud cluster shared among heterogeneous Web Service and MapReduce applications. Our framework accounts for different QoS requirements and time-varying workload intensities. An admission controller was adopted to handle request rejections during overload, which in turn incur penalty costs. The goal was to optimize the resource allocation in order to reduce operational and penalty costs while meeting QoS targets of the admitted requests. A classical performance model was adopted to model the response times for Web Service applications whereas approximate formulae allow to obtain estimates on the execution times of MR applications. We provide a mathematical model for the joint admission control and capacity allocation problem in which we embed the QoS performance models. The proprieties of the model optimal solution allow to directly translate the workload intensities and QoS requirements of each user class of the two workload types into the optimal amount of required resources (number of VMs). An equivalent reduced problem formulation is then obtained, which can be optimally and efficiently solved with a greedy procedure. We benchmark the joint resource management approach against a static approach in which the cluster VMs are a priori partitioned among the two application types. The two approaches are compared in terms of cluster utilization and total cost for several scenarios of interest representing different mixtures of time zones from which requests can originate. The proposed approach scales well for instances with hundreds of classes per workload type and provides average cost savings up to 50% while increasing the cluster utilization by up to 18%.

ACKNOWLEDGMENT

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

REFERENCES

[1] European Commission, Uptake of Cloud in Europe. Follow-up of IDC Study on Quantitative estimates of the demand for Cloud Computing in Europe and the likely barriers to take-up. Digital Agenda for Europe report., Publications Office of the European Union, Luxembourgdoi:10.2759/791317.

[2] European Commission, A Digital Single Market for Europe: Commission sets out 16 initiatives to make it happen, Press Release. 6 May 2015. [Online] Accessible:, http://europa.eu/rapid/press-release_IP-15-4919_en.htm.

[3] Evolution of the Netflix Data Pipeline, http://techblog.netflix.com/2016/02/evolution-of-netflix-data-pipeline.html (2016).

[4] Amazon's recommendation secret, http://fortune.com/2012/07/30/amazons-recommendation-secret/ (2012).

[5] ONLINE, MapReduce for Business Intelligence and Analytics. Accessible:, http://www.dbta.com/Columns/Applications-Insight/MapReduce-for-Business-Intelligence-and-Analytics-56043.aspx.

[6] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, I. Stoica, Mesos: A platform for fine-grained resource sharing in the data center., in: NSDI, Vol. 11, 2011, pp. 22–22.

[7] R. B. Clay, Z. Shen, X. Ma, Accelerating batch analytics with residual resources from interactive clouds, in: Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on, IEEE, 2013, pp. 414–423.

[8] W. Zhang, S. Rajasekaran, S. Duan, T. Wood, M. Zhu, Minimizing interference and maximizing progress for hadoop virtual machines, SIGMETRICS Performance Evaluation Review 42 (4) (2015) 62–71. doi:10.1145/2788402.2788411.
URL http://doi.acm.org/10.1145/2788402.2788411

[9] D. Ardagna, G. Casale, M. Ciavotta, J. F. Pérez, W. Wang, Quality-of-service in cloud computing: modeling techniques and their applications, Journal of Internet Services and Applications 5 (1) (2014) 1–17.

[10] J. Polo, D. Carrera, Y. Becerra, J. Torres, E. Ayguad, M. Steinder, I. Whalley, Performance-driven task co-scheduling for mapreduce environments, in: NOMS, 2010.

[11] B. T. Rao, L. S. S. Reddy, Survey on improved scheduling in hadoop mapreduce in cloud environments, CoRR abs/1207.0780.
URL http://arxiv.org/abs/1207.0780

[12] C. Shanklin, Benchmarking Apache Hive 13 for Enterprise Hadoop, https://hadoop.apache.org/docs/r2.4.1/hadoop-yarn/hadoop-yarn-site/CapacityScheduler.html.

[13] M. Lin, L. Zhang, A. Wierman, J. Tan, Joint optimization of overlapping phases in MapReduce, SIGMETRICS Performance Evaluation Review 41 (3) (2013) 16–18.

[14] A. Verma, L. Cherkasova, R. H. Campbell, ARIA: Automatic Resource Inference and Allocation for Mapreduce Environments, in: ICAC, 2011. doi:10.1145/1998582.1998637.
URL http://doi.acm.org/10.1145/1998582.1998637

[15] M. Malekimajd, D. Ardagna, M. Ciavotta, A. M. Rizzi, M. Passacantando, Optimal map reduce job capacity allocation in cloud systems, SIGMETRICS Perform. Eval. Rev. 42 (4) (2015) 51–61. doi:10.1145/2788402.2788410.
URL http://doi.acm.org/10.1145/2788402.2788410

[16] C. Curino, D. E. Difallah, C. Douglas, S. Krishnan, R. Ramakrishnan, S. Rao, Reservation-based scheduling: If you're late don't blame us!, in: SoCC, 2014.

[17] J. Almeida, V. Almeida, D. Ardagna, Í. Cunha, C. Francalanci, M. Trubian, Joint admission control and resource allocation in virtualized servers, Journal of Parallel and Distributed Computing 70 (4) (2010) 344–362.

[18] L. Cherkasova, P. Phaal, Session-based admission control: a mechanism for peak load management of commercial web sites, Computers, IEEE Transactions on 51 (6) (2002) 669–685. doi:10.1109/TC.2002.1009151.

[19] Amazon Elastic Map Reduce Nodes, http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-nodes.html.

[20] Capacity scheduler, http://hadoop.apache.org/docs/r2.3.0/hadoop-yarn/hadoop-yarn-site/CapacityScheduler.html.

[21] D. Kumar, L. Zhang, A. Tantawi, Enhanced inferencing: Estimation of a workload dependent performance model, in: Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, p. 47.

[22] L. Zhang, X. Meng, S. Meng, J. Tan, K-scope: Online performance tracking for dynamic cloud applications, Proceedings of the 10th ICAC.

[23] B. Addis, D. Ardagna, B. Panicucci, M. S. Squillante, L. Zhang, A hierarchical approach for the resource management of very large cloud platforms, Dependable and Secure Computing, IEEE Transactions on 10 (5) (2013) 253–272.

[24] D. Ardagna, M. Ciavotta, M. Passacantando, Generalized Nash Equilibria for the Service Provisioning Problem in Multi-Cloud Systems, IEEE Transactions on Services Computing. To Appear.

[25] Getting MapReduce 2 Up to Speed, http://blog.cloudera.com/blog/2014/02/getting-mapreduce-2-up-to-speed/.

[26] Apache Hadoop YARN: Avoiding 6 Time-Consuming "Gotchas", http://blog.cloudera.com/blog/2014/04/apache-hadoop-yarn-avoiding-6-time-consuming-gotchas/.

[27] L.Cano, G.Carello, D.Ardagna, A Framework for joint resource allocation of MapReduce and Web Service applications in a shared Cloud cluster, Tech. rep. (2016).
URL http://home.deib.polimi.it/ardagna/CanoCarelloArdagna-ExtendedVer2017.pdf

[28] K. Ousterhout, A. Panda, J. Rosen, S. Venkataraman, R. Xin, S. Ratnasamy, S. Shenker, I. Stoica, The case for tiny tasks in compute clusters, in: HotOS, 2013.

[29] D. Ardagna, S. Casolari, M. Colajanni, B. Panicucci, Dual time-scale distributed capacity allocation and load redirect algorithms for cloud systems, J. Parallel Distrib. Comput. 72 (6) (2012) 796–808. doi:10.1016/j.jpdc.2012.02.014.
URL http://dx.doi.org/10.1016/j.jpdc.2012.02.014

[30] E. Lazowska, J. Zahorjan, G. Graham, K. Sevcik, Quantitative system performance: computer system analysis using queueing network models, Prentice-Hall, Inc., 1984.

[31] D. Ardagna, B. Panicucci, M. Passacantando, Generalized nash equilibria for the service provisioning problem in cloud systems, Services Computing, IEEE Transactions on 6 (4) (2013) 429–442.

[32] Q. Zhang, Q. Zhu, M. F. Zhani, R. Boutaba, J. L. Hellerstein, Dynamic service placement in geographically distributed clouds, Selected Areas in Communications, IEEE Journal on 31 (12) (2013) 762–772.

[33] D. Ardagna, M. S. Squillante, Special issue on performance and resource management in big data applications, SIGMETRICS Performance Evaluation Review 42 (4) (2015) 2. `doi:10.1145/2788402.2788404`.
URL `http://doi.acm.org/10.1145/2788402.2788404`

[34] AMPL, Ampl modeling language for mathematical programming., `http://www.ampl.com/`.

[35] IBM, IBM ILOG CPLEX Optimizer, `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`.

[36] KNITRO, Artelys Knitro, a non-linear optimization solver, `http://www.artelys.com/en/optimization-tools/knitro`.

[37] A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, Above the clouds: A berkeley view of cloud computing, Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS 28 (2009) 13.

[38] R. Birke, L. Y. Chen, E. Smirni, Data centers in the cloud: A large scale performance study, in: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on, IEEE, 2012, pp. 336–343.

[39] D. Ardagna, M. Ciavotta, M. Passacantando, Generalized nash equilibria for the service provisioning problem in multi-cloud systems, IEEE Transactions on Services Computing (1) 1–1.

[40] E. Gianniti, Game theory models for mapreduce: joint admission control and capacity allocation.

[41] D. Ardagna, S. Bernardi, E. Gianniti, S. K. Aliabadi, D. Perez-Palacin, J. I. Requeno., Modeling Performance of Hadoop Applications: A Journey from Queueing Networks to Stochastic Well Formed Nets, in: ICA3PP 2016 Proceedings. To Appear., 2016.

[42] TPC BenchmarkDS (TPC-DS): The Benchmark Standard for decision support solutions including Big Data, `http://www.tpc.org/tpcds/`.

[43] D. Ardagna, S. Casolari, M. Colajanni, B. Panicucci, Dual time-scale distributed capacity allocation and load redirect algorithms for cloud systems, Journal of Parallel and Distributed Computing 72 (6) (2012) 796–808.

[44] D. Ardagna, B. Panicucci, M. Trubian, L. Zhang, Energy-aware autonomic resource allocation in multitier virtualized environments, Services Computing, IEEE Transactions on 5 (1) (2012) 2–19.

[45] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, G. Jiang, Power and performance management of virtualized computing environments via lookahead control, Cluster computing 12 (1) (2009) 1–15.

[46] J. Anselmi, D. Ardagna, M. Passacantando, Generalized Nash equilibria for SaaS/PaaS Clouds, European Journal of Operational Research 236 (1) (2014) 326–339.

[47] L. T. X. Phan, Z. Zhang, Q. Zheng, B. T. Loo, I. Lee, An empirical analysis of scheduling techniques for real-time cloud-based data processing, in: SOCA, 2011. `doi:10.1109/SOCA.2011.6166240`.
URL `http://dx.doi.org/10.1109/SOCA.2011.6166240`

[48] Z. Zhang, L. Cherkasova, B. T. Loo, Exploiting cloud heterogeneity for optimized cost/performance mapreduce processing, in: CloudDP, 2014.

[49] Z. Zhang, L. Cherkasova, A. Verma, B. T. Loo, Automated Profiling and Resource Management of Pig Programs for Meeting Service Level Objectives, in: ICAC, 2012. `doi:10.1145/2371536.2371546`.
URL `http://doi.acm.org/10.1145/2371536.2371546`