D-SPACE4Cloud: A Design Tool for Big Data Applications

Michele Ciavotta, Eugenio Gianniti$^{(\boxtimes)}$, and Danilo Ardagna

Dip. di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy
{michele.ciavotta,eugenio.gianniti,danilo.ardagna}@polimi.it

Abstract. The last years have seen a steep rise in data generation world-wide, with the development and widespread adoption of several software projects targeting the Big Data paradigm. Many companies currently engage in Big Data analytics as part of their core business activities, nonetheless there are no tools or techniques to support the design of the underlying infrastructure configuration backing such systems. In particular, the focus in this paper is set on Cloud deployed clusters, which represent a cost-effective alternative to on premises installations. We propose a novel tool implementing a battery of optimization and prediction techniques integrated so as to efficiently assess several alternative resource configurations, in order to determine the minimum cost cluster deployment satisfying Quality of Service constraints. Further, the experimental campaign conducted on real systems shows the validity and relevance of the proposed method.

Keywords: MapReduce · Optimization · Queueing networks

1 Introduction

Nowadays, the Big Data adoption has moved from experimental projects to mission-critical enterprise-wide deployments, providing new insights, competitive advantage, and business innovation [20]. IDC estimates that the Big Data market grew from $3.2 billion in 2010 to $16.9 billion in 2015 with a compound annual growth rate of 39.4%, about seven times the one of the overall ICT market [5]. From the technological perspective, the MapReduce programming model is one of the most widely used solutions to support Big Data applications [25]. Its open source implementation, Apache Hadoop, is able to manage large datasets over either commodity clusters or high performance distributed topologies [40]. MapReduce has attracted the interest of both industry and academia, as it overtakes the scalability level that can be achieved by traditional data warehouse and business intelligence technologies [25].

However, the adoption of Hadoop and other Big Data technologies is complex. The deployment and setup of an implementation is time-consuming, expensive, and resource-intensive. Companies need an *easy button* to accelerate the deployment of Big Data analytics [18]. The pay-per-use approach and the almost infinite capacity of Cloud infrastructures can be used efficiently in supporting data intensive computations. Many Cloud providers already include in their offering MapReduce based platforms, among which Microsoft HDInsight [8] or Amazon Elastic MapReduce [2]. IDC estimates that, by 2020, nearly 40% of Big Data analyses will be supported by public Clouds [5], while Hadoop touched half of the world data last year [22].

In the very beginning, MapReduce jobs were meant to run on dedicated clusters to support batch analyses via a FIFO scheduler [32,33]. Nevertheless, MapReduce applications have evolved and, nowadays, they entail also interactive queries, submitted by different users and performed on shared clusters, possibly with some guarantees on their execution time [42,43]. In such systems, capacity allocation becomes one of the most important aspects. Determining the optimal number of nodes in a cluster shared among multiple users performing heterogeneous tasks is an important and difficult problem [37]. In this context, one of the main challenges [37,43] is that the execution time of a MapReduce job is generally unknown in advance.

Our focus in this paper is to provide a software tool able to support system administrators and operators in the capacity planning process of shared Hadoop 2.x Cloud clusters supporting both batch and interactive applications with deadline guarantees. Having such information available at design-time enables operators to make more informed decisions about the technology to use and to fully exploit the potential offered by the Cloud infrastructure. We formulate the capacity planning problem by means of a mathematical model, with the aim of minimizing the cost of Cloud resources. The problem considers multiple VM types as candidates to support the execution of Hadoop applications from multiple user classes. Through a search space exploration, our approach optimizes the configuration of a shared cluster in terms of VM type and instances number considering specific Cloud provider pricing models (namely, reserved and spot instances [1]).

Our work is one of the first contributions facing the problem of optimal sizing of Hadoop 2.x Cloud systems adopting the Capacity Scheduler [6]. We demonstrate the effectiveness of our approach by considering Hive queries [4] and the TPC-DS industry benchmark for business intelligence and data warehouses [9] as reference application. Amazon EC2 and the CINECA Italian supercomputing center have been considered as target deployments.

This paper is organized as follows. Section 2 presents in detail the problem addressed in the paper. In Sect. 3 we focus on the formulation of the optimization problem and on the design-time exploration algorithm to solve it implemented by our D-SPACE4Cloud tool. In Sect. 4 we evaluate the effectiveness of our optimization method. Finally, in Sect. 5 we compare our work with other proposals available in the literature and draw the conclusions in Sect. 6.

2   Problem Statement

In this section we aim at introducing some important details of the problem addressed in this work. We envision the following scenario, wherein a company needs to set up a cluster to carry out efficiently a set of interactive Big Data queries. A Hadoop 2.x cluster featuring the YARN Capacity Scheduler and running on a public Cloud IaaS is considered a fitting technological solution for the requirements of the company.

In particular, the cluster must support the parallel execution of Big Data applications in the form of Hadoop jobs or Hive/Pig queries. Different classes $\mathcal{C} = \{i \mid i = 1, \ldots, n\}$ gather applications that show a similar behavior. The cluster composition and size, in terms of type and number of VMs, must be decided in such a way that, for every application class i, H_i jobs are guaranteed to execute concurrently and complete before a prearranged deadline D_i.

Moreover, YARN is configured in a way that all available cores can be dynamically assigned to either Map or Reduce tasks. Finally, in order to limit the risk of data corruption and according to the practices suggested by major Cloud vendors [2,8], the datasets reside on an external storage infrastructure [3,7] accessible at quasi-constant time.

As, in general, IaaS providers feature a limited, but possibly large, catalog of VM configurations $\mathcal{V} = \{j \mid j = 1, \ldots, m\}$ that differ in capacity (CPU speed, number of cores, available memory, etc.) and cost, making the right design-time decision poses a challenge that can lead to important savings throughout the cluster life-cycle. We denote with τ_i the VM type j used to support jobs of class i and with ν_i the number of VMs of such a kind allocated to class i. In this scenario, we consider a pricing model derived from *Amazon EC2* [1]. The provider offers: (1) *reserved* VMs, for which it adopts a one-time payment policy that grants access to a certain number of them for the contract duration; and (2) *spot* VMs, for which customers bid and compete for unused datacenter capacity, yielding very competitive hourly fees. In order to obtain the most cost-effective configuration, we rely on reserved VMs for the bulk of computational needs and complement them with spot VMs. In the following, R_i is the number of reserved VMs assigned to class i, whilst s_i is the number of spot VMs. Let σ_{τ_i} be the unit cost for spot VMs of type τ_i, whilst π_{τ_i} is the effective hourly cost for one reserved VM, i.e., it is the unit upfront payment normalized over the contract duration. Overall, the cluster hourly renting out costs can be calculated as follows:

$$\text{cost} = \sum_{i \in \mathcal{C}} (\sigma_{\tau_i} s_i + \pi_{\tau_i} R_i) \tag{1}$$

Let $\nu_i = R_i + s_i$: as the reliability of spot VMs depends on market fluctuations, to keep a high Quality of Service (QoS) the number of spot VMs is bounded not to be greater than a fraction η_i of ν_i for each class i.

Reducing the operating costs of the cluster by using efficiently the leased virtual resources is in the interest of the company. This translates into a Resource
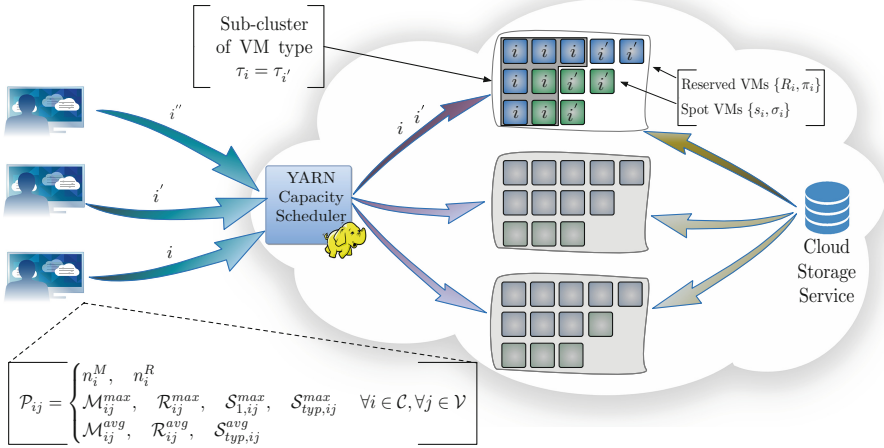
Fig. 1. Reference system

Provisioning problem where the renting out costs must be minimized subject to the fulfillment of QoS requirements, namely H_i per-class concurrency level given certain deadlines D_i. In the following we assume that the system supports H_i users for each class and that users work interactively with the system and run another job after a think time exponentially distributed with mean Z_i, i.e., the system is represented as a closed model [24]. In order to rigorously model and solve this problem, it is crucial to predict with fair confidence the execution times of each application class under different conditions: level of concurrency, cluster size, and composition. Following the approach presented in [37], it is possible to derive from Hadoop logs a *job profile*, i.e., a concise behavior characterization for each class. Following the notation brought forth in [27,37], given a certain VM of type j, the job profile $\mathcal{P}_{ij}$ for application class i aggregates the following information: (1) n_i^M and n_i^R, respectively the total number of Map and Reduce tasks per job; (2) $\mathcal{M}_{ij}^{max}$, $\mathcal{R}_{ij}^{max}$, $\mathcal{S}_{1,ij}^{max}$, and $\mathcal{S}_{typ,ij}^{max}$, the maximum duration of a single Map, Reduce, and Shuffle task (notice that the first Shuffle wave of a given job is distinguished from all the subsequent ones); (3) $\mathcal{M}_{ij}^{avg}$, $\mathcal{R}_{ij}^{avg}$, and $\mathcal{S}_{typ,ij}^{avg}$, i.e., the average duration of Map, Reduce, and Shuffle tasks, respectively. Given the amount and type of resources allocated, the concurrency level, and the job profile, the estimated execution time can generically be expressed as in (2):

$$T_i = \mathcal{T}\left(\mathcal{P}_{i,\tau_i}, \nu_i; H_i, Z_i\right), \quad \forall i \in \mathcal{C}. \tag{2}$$

What is worthwhile to note is that the previous formula represents a general relation describing either closed form results based on bounds, as those presented in [27], or the average execution times derived via simulation, the approach adopted in this paper. Since the execution of jobs on a suboptimal VM type

might give rise to performance disruptions, it is critical to avoid assigning tasks belonging to class i to the wrong VM type $j \neq \tau_i$. Indeed, YARN allows for specifying Node Labels and partitioning nodes in the cluster according to these labels, then it is possible to enforce this separation. Our configuration statically splits different VM types with this mechanism and adopts within each partition either a further static separation in classes or a work conserving scheduling mode, where idle resources can be assigned to jobs requiring the same VM type. The assumption on the scheduling policy governing the exploitation of idle resources is not critical: it only affects the interpretation of results, where the former case leads to sharp predictions, while in the latter the outcomes of the optimization algorithm are upper bounds, with possible performance improvements due to a better cluster utilization. Equations (2) can be used to formulate the deadline constraints as:

$$T_i \leq D_i, \quad \forall i \in \mathcal{C}. \tag{3}$$

In light of the above, we can say that the ultimate goal of the proposed approach is to determine the optimal VM type selection τ_i and number and pricing models of VMs $\nu_i = R_i + s_i$ for each class i such that the sum of costs is minimized, while the deadlines and concurrency levels are met.

Table 1. Model parameters

Parameter	Definition
$\mathcal{C}$	Set of application classes
$\mathcal{V}$	Set of VM types
H_i	Number of concurrent users for class i
Z_i	Class i think time [ms]
D_i	Deadline associated to applications of class i [ms]
η_i	Maximum percentage of spot VMs allowed to class i
σ_j	Unit hourly cost for spot VMs of type j [€/h]
π_j	Effective hourly price for reserved VMs of type j [€/h]
$\mathcal{P}_{ij}$	Job profile of class i with respect to VM type j

The reader is referred to Fig. 1 for a graphical overview of the main elements of the considered resource provisioning problem. Furthermore, in Table 1 a complete list of the parameters used in the models presented in the next sections is reported, whilst Table 2 summarizes the decision variables.

3   Problem Formulation and Solution

In the following we present the optimization model and techniques exploited by the D-SPACE4Cloud tool in order to determine the optimal VM mix given

Table 2. Decision variables

Variable	Definition
ν_i	Number of VMs assigned for the execution of applications from class i
R_i	Number of reserved VMs booked for the execution of applications from class i
s_i	Number of spot VMs assigned for the execution of applications from class i
x_{ij}	Binary variable equal to 1 if class i is hosted on VM type j

the profiles characterizing the applications under study and the possible Cloud providers to host the virtual cluster. Further, we describe the heuristic algorithm adopted to efficiently tackle the resource provisioning problem by exploiting the presented models.

3.1 Optimization Model

Basic building blocks for this tool are the models of the system under study. First of all, we need a quick, although rough, method to estimate completion times and operational costs: to this end, we exploit a mathematical programming formulation based on jobs execution time bounds (see [27]). In this way, it is possible to swiftly explore several possible configurations and point out the most cost-effective among the feasible ones. Afterwards, the required resource configuration can be fine-tuned using more accurate, even if more time consuming and computationally demanding, queueing network (QN) simulations, reaching a precise prediction of the expected response time.

According to the previous considerations, the first step in the optimization procedure consists in determining the most cost-effective resource type, based on their price and the expected performance. This will be done by exploiting a set of logical variables x_{ij}: we will enforce that only $x_{i,\tau_i} = 1$, thus determining the optimal VM type τ_i for application class i. We address this issue proposing the following mathematical programming formulation:

$$\min_{\mathbf{x},\boldsymbol{\nu},\mathbf{s},\mathbf{R}} \quad \sum_{i \in \mathcal{C}} (\sigma_{\tau_i} s_i + \pi_{\tau_i} R_i) \tag{P1a}$$

subject to:

$$\sum_{j \in \mathcal{V}} x_{ij} = 1, \quad \forall i \in \mathcal{C} \tag{P1b}$$

$$\mathcal{P}_{i,\tau_i} = \sum_{j \in \mathcal{V}} \mathcal{P}_{ij} x_{ij}, \quad \forall i \in \mathcal{C} \tag{P1c}$$

$$\sigma_{\tau_i} = \sum_{j \in \mathcal{V}} \sigma_j x_{ij}, \quad \forall i \in \mathcal{C} \tag{P1d}$$

$$\pi_{\tau_i} = \sum_{j \in \mathcal{V}} \pi_j x_{ij}, \quad \forall i \in \mathcal{C} \tag{P1e}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i \in \mathcal{C}, \forall j \in \mathcal{V} \tag{P1f}$$

$$(\boldsymbol{\nu}, \mathbf{s}, \mathbf{R}) \in \arg\min \sum_{i \in \mathcal{C}} (\sigma_{\tau_i} s_i + \pi_{\tau_i} R_i) \tag{P1g}$$

subject to:

$$s_i \leq \frac{\eta_i}{1 - \eta_i} R_i, \quad \forall i \in \mathcal{C} \tag{P1h}$$

$$\nu_i = R_i + s_i, \quad \forall i \in \mathcal{C} \tag{P1i}$$

$$\mathcal{T}(\mathcal{P}_{i,\tau_i}, \nu_i; H_i, Z_i) \leq D_i, \quad \forall i \in \mathcal{C} \tag{P1j}$$

$$\nu_i \in \mathbb{N}, \quad \forall i \in \mathcal{C} \tag{P1k}$$

$$R_i \in \mathbb{N}, \quad \forall i \in \mathcal{C} \tag{P1l}$$

$$s_i \in \mathbb{N}, \quad \forall i \in \mathcal{C} \tag{P1m}$$

Problem (P1) is a bilevel resource allocation problem where the outer objective function (P1a) considers running costs. The first set of constraints, (P1b), associates each class i with only one VM type j, hence the following constraints, ranging from (P1c) to (P1e), pick the values for the inner problem parameters.

The inner objective function (P1g) has the same expression as (P1a), but in this case the prices σ_{τ_i} and π_{τ_i} are fixed, as they have been chosen at the upper level. The following constraints, (P1h), enforce that spot instances do not exceed a fraction η_i of the total assigned VMs and constraints (P1i) add all the VMs available for class i, irrespective of the pricing model. Further, constraints (P1j) mandate to respect the deadlines D_i. In the end, all the remaining decision variables are taken from the natural numbers set, according to their interpretation.

The presented formulation of Problem (P1) is particularly difficult to tackle, as it is a mixed integer nonlinear programming (MINLP) problem depending on $\mathcal{T}$. According to the literature about complexity theory [17], integer programming problems belong to the NP-hard class, hence the same applies to (P1). However, since there is no constraint linking variables belonging to different application classes, we can split this general formulation into several smaller and independent problems, one per class i. As it will be discussed in the following section, we evaluated the average job completion time $\mathcal{T}$ by considering bounds and relying on QN models simulation.

3.2   Solution Technique

The aim of this section is to provide a brief description of the optimization approach embedded in D-SPACE4Cloud. The tool implements an optimization mechanism that efficiently explores the space of possible configurations.

Figure 2 depicts the main elements of the D-SPACE4Cloud architecture that come into play in the optimization scenario. The tool takes as input a description

of the considered problem, consisting in a set of applications, a set of suitable
VMs for each application along with the respective job profiles for each machine,
and QoS constraints expressed in terms of deadlines D_i for each considered appli-
cation. Specifically, all these parameters are collected in a JSON file provided as
input to the tool. The *Initial Solution Builder* generates a starting solution for
the problem using a MINLP formulation where the time expression $\mathcal{T}$ appearing
in constraint (P1j) is a convex function: then the inner level of (P1) is a convex
nonlinear problem and we exploit the Karush-Kuhn-Tucker conditions to speed
up the solution process. For more details the reader is referred to [27]. It must
be highlighted, at this point, that the quality of the returned solution can still
be improved: this because the MINLP relies on an approximate representation
of the Application-Cluster liaison. For this reason, a QN model is exploited to
get a more accurate execution time assessment. This model allows to estimate
MapReduce jobs execution time with an average error around 14% with respect
to real systems. The increased accuracy leaves room for further cost reduction;
however, since QNs simulation is time consuming, the space of possible cluster
configurations has to be explored in the most efficient way, avoiding to evaluate
unpromising configurations.

In the light of such considerations, a heuristic approach has been adopted and
a component called *Parallel Local Search Optimizer* has been devised. Internally,
it implements a parallel hill climbing (HC) technique to optimize the number
of replicas of the assigned resource for each application; the goal is to find the
minimum number of resources to fulfill the QoS requirements. This procedure is
applied independently, and in parallel, on all application classes and terminates
when a further reduction in the number of replicas would lead to an infeasible
solution. As soon as all the classes reach convergence, it is possible to retrieve
from the D-SPACE4Cloud tool a JSON file listing the results of the overall
optimization procedure.

For the sake of clarity, HC is a local-search-based procedure that operates
on the current solution performing a change, a so-called *move*, in the structure
of the solution in such a way that the newly generated solution could possibly
show an improved objective value. If the move is successful it is applied again
on the new solution and the process is repeated until no further improvement is
possible. The HC algorithm stops when a local optimum is found; however, if the
objective to optimize is convex, HC is able to find the global optimum. This is
the case of the considered cost function (1), which is linear in the number of VMs
in the cluster, since VM prices are fixed at the first level. Hence, every feasible
instance of the inner problem can be heuristically solved to optimality through
HC. The initial solution S, obtained from the MINLP solution, is evaluated using
the QN model and each one of its parts is optimized separately and in parallel.
If the partial solution S_i is infeasible the size of its personal cluster is increased
by one unit until it becomes feasible. Otherwise, the procedure attempts to
decrease the cost function by reducing the cluster size. One-VM moves might
seem problematic, but, given the quite accurate initial solution, the tool only
needs to explore a small neighborhood of possible configurations. Finally, it is
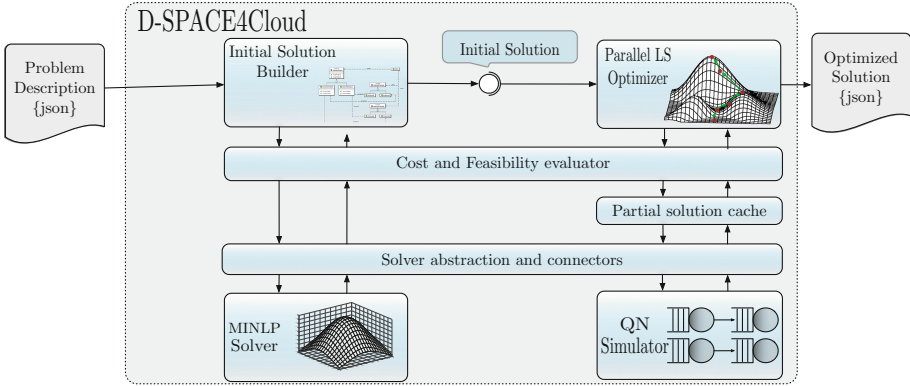
Fig. 2. D-SPACE4Cloud architecture

worth pointing out that every time the total number of machines in a cluster is incremented or decremented the best mix of pricing models (i.e., R_i, s_i) is computed so as to minimize the configuration cost.

4   Experimental Analysis

In this section we show the results of several experiments performed to validate the proposed approach. All these experiments have been performed on two Ubuntu 14.04 VMs hosted on an Intel Xeon E5530 2.40 GHz equipped server. The first VM ran D-SPACE4Cloud and the MINLP solver which was used to generate an initial solution for the optimization problem presented in Sect. 3.1 (see [27] for further details). The second one, instead, ran JMT 0.9.3 [13], a QN simulator.

4.1   Experimental Setup and Design of Experiments

In order to obtain job profiles, we devised a set of five SQL queries denoted with R1–5 (see [16]). We then generated synthetic data compliant with the specifications of the industry standard benchmark TPC-DS [9] and executed the queries on Apache Hive [4]. Notice that we generated data at several scale factors ranging from 250 GB to 1 TB. Since profiles collect statistical information about jobs, we repeated the profiling runs at least twenty times per query. Properly parsing the logs allows to extract all the parameters composing every query profile, for example average and maximum task execution times, number of tasks, etc. Profiling has been performed on Amazon EC2, by considering m4.xlarge instances, and on PICO[1], the Big Data cluster offered by CINECA, the Italian supercomputing center. The cluster rented on EC2 was composed of 30 computational

---

[1] http://www.hpc.cineca.it/hardware/pico.

nodes, for a total of 120 vCPUs hosting 240 containers, whilst on PICO we used up to 120 cores configured to host one container per core. In the first case every container had 2 GB RAM and in the second 6 GB. Along with profiles, we also collected lists of task execution times to feed into the replayer in JMT service centers. In the end, we determined query profiles for different VM types.

4.2   Queueing Network Validation

This section shows results validating the accuracy of the underlying QN model. Table 3 reports the average percentage error ϑ obtained by comparing the queries execution time extracted from logs, T, with the ones evaluated from QN simulation, denoted with τ: $\vartheta = \frac{\tau - T}{T}$. Among these experiments, we considered both single user scenarios, repeatedly running the same query on a dedicated cluster with $Z_i = 10$ s, and multiple users scenarios. In the worst case, the relative error can reach up to 32.97%, which is perfectly in line with the expected accuracy in the performance prediction field [24], while the average relative error is 14.13% overall.

Table 3. Queueing network model validation

Query	H_i	#Cores	Dataset [GB]	#Maps	#Reducers	T [ms]	τ [ms]	ϑ [%]
R1	1	240	250	500	1	55410	50753.34	−8.40
R2	1	240	250	65	5	36881	27495.31	−25.45
R3	1	240	250	750	1	76806	77260.03	0.60
R4	1	240	250	524	384	92197	78573.96	−14.72
R1	1	60	500	287	300	378127	411940.93	8.94
R3	1	100	500	757	793	401827	524759.36	30.59
R3	1	120	750	1148	1009	661214	759230.77	14.82
R4	1	60	750	868	910	808490	844700.85	4.48
R3	1	80	1000	1560	1009	1019973	1053829.78	−1.00
R5	1	80	1000	64	68	39206	36598.32	−6.65
R1	3	20	250	144	151	1002160	1038951.05	3.67
R1	5	20	250	144	151	1736949	1215490.20	−30.02
R2	3	20	250	4	4	95403	112050.45	17.45
R2	5	20	250	4	4	145646	97619.46	−32.97
R1	5	40	250	144	151	636694	660241.29	3.70
R2	3	40	250	4	4	86023	105785.41	22.97
R2	5	40	250	4	4	90674	103173.38	13.78

4.3   Scenario-Based Experiments

The optimization approach described in Sect. 3 needs to be validated, ensuring that it is capable of catching realistic behaviors we expect of the system under analysis. We test this property with a set of assessment runs where we fix all the problem parameters but one and verify that the solutions follow an intuitive evolution.

The main axes governing performance in Hadoop clusters hosted on public Clouds are the level of concurrency and the deadlines. In the first case, increasing H_i and fixing all the remaining parameters, we expect a need for more VMs to support the rising workload, thus leading to an increase of renting out costs. On the other hand, if at fixed parameters we tighten the deadlines D_i, again we should observe increased costs: the system will require a higher parallelism to shrink response times, hence more computational nodes to support it.

For the sake of clarity, we performed single-class experiments: considering only one class per experiment allows for an easier interpretation of the results. Figure 3 reports the solutions obtained with the 250 GB dataset profiles. The average running time for these experiments is about two hours. All the mentioned figures show the cost in €/h plotted against decreasing deadlines in ms for both the real VM types considered: CINECA is the 20-core node available on PICO, whilst m4.xlarge is the 4-core instance rented on Amazon AWS. In Figs. 3a and b the expected cost increase due to tightening deadlines is apparent for two representative queries, R1 and R3, considering 10 concurrent users. Further, in both cases it is cheaper to provision a Cloud cluster consisting of the smaller Amazon-offered instances, independently of the deadlines. It is then interesting to observe that R1 shows a different behavior if the required concurrency level increases. Figure 3c shows that, as the deadlines become tighter and tighter, it is possible to identify a region where executing the workload on larger VMs becomes more economic.

5   Related Work

Capacity planning and architecture design space exploration are important problems analyzed in the literature [10,14]. High level models and tools to support software architects (see, e.g., Palladio Component Model and PerOptirex design environment [12,23], or stochastic process algebra [36] and the PEPA Eclipse plugin [30]) have been proposed for identifying the best configuration given a set of QoS requirements; unfortunately they neither support Cloud-specific abstractions nor do directly address the problem of deriving an optimized configuration for Cloud and Big Data clusters. On the other side, capacity management and cluster sizing for Big Data applications has received also a widespread interest by both academia and industry. The starting point is the consideration that Hadoop often requires an intense tuning phase in order to exhibit its full potential. For this reason, *Starfish*, a self-tuning system for analytics on Hadoop, has been proposed [19]. The resource provisioning problem, instead, has been faced by Tian and Chen [35]. The goal is the minimization of the execution cost for a
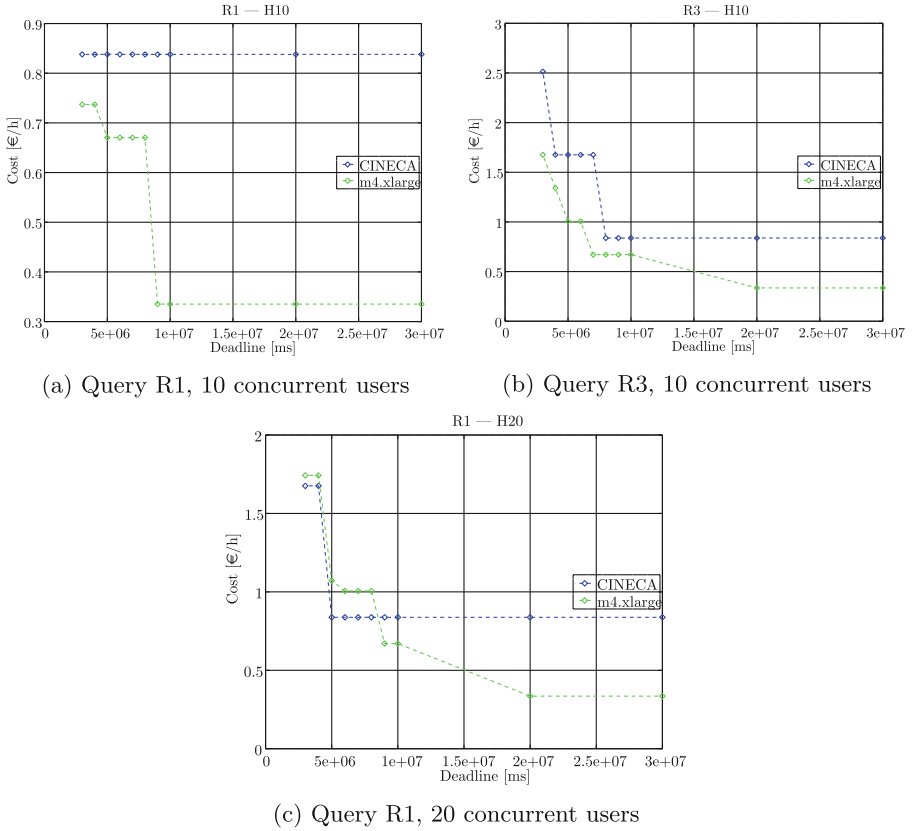
(a) Query R1, 10 concurrent users

(b) Query R3, 10 concurrent users

(c) Query R1, 20 concurrent users

Fig. 3. Cluster costs with varying deadlines

single application. They present a cost model that depends on the dataset size and on some characteristics of the considered application.

Verma et al. [38] proposed a framework for the profiling and duration prediction of applications running on heterogeneous resources. An approach to this problem based on closed QNs is presented in [11]. This work is noteworthy as it explicitly considers contention and parallelism on compute nodes to evaluate the execution time of a MapReduce application. However, the weak spot of this approach is that it contemplates the Map phase alone. Vianna et al. [39] worked on a similar solution; however the validation phase has been carried out considering a cluster dedicated to the execution of a single application at a time. Both Map and Reduce phases are considered in [34]. In this work the Map phase is modeled as an $M/G/1$ queue, whereas for the Reduce phase a multi-server queue have been used.

Castiglione et al. [15] introduce a novel modeling approach based on mean field analysis and provide fast approximate methods to predict the performance of Big Data systems. Deadlines for MapReduce jobs are considered in [31].

The work proposes to adapt to the problem some classical multiprocessor scheduling policies; in particular, two versions of the Earliest Deadline First heuristic are presented and proved to outperform off-the-shelf schedulers. A similar approach is presented in [41], where the authors present a solution to manage clusters shared among Hadoop application and more traditional Web systems. The problem of progress estimation of multiple parallel queries is addressed in [29]. To this aim, the authors present Parallax, a tool able to predict the completion time of MapReduce jobs. ParaTimer [28], an extension of Parallax, features support to multiple parallel queries expressed as directed acyclic graphs (DAGs). Recently, also the integration of Big Data and high performance computing (HPC) applications received attention in the literature. In [21], the authors compare and contrast the two paradigms, highlighting the similarities that can be exploited to devise integrated deployments. An integration proposal is presented in [26], where RADICAL-Pilot is adopted to run jobs in a hybrid Hadoop-HPC environment.

In [37] the ARIA framework is presented. This work is the closest to our contribution and focuses on clusters dedicated to single user classes handled by the FIFO scheduler. The framework addresses the problem of calculating the most suitable number of resources to allocate to Map and Reduce tasks in order to meet a user-defined due date for a certain application; the aim is to avoid as much as possible costs due to resource over-provisioning. We borrow from this work the compact job profile definition, used there to calculate a lower bound, an upper bound, and an estimation of application execution times. Finally, they present a performance model eventually improved in [43] and then validated through a simulation study and an experimental campaign on a 66-node Hadoop cluster. The same authors, in a more recent work [42], provided a solution for optimizing the execution of a workload specified as a set of DAGs under the constraints of a global deadline or budget. All the above mentioned works are based on Hadoop 1.0, where CPU slots are statically allocated to Map and Reduce tasks and the basic FIFO scheduler is considered. To the best of our knowledge, ours is one of the first contribution coping with Hadoop 2.x shared clusters based on the Capacity scheduler, hence relaxing Hadoop 1.0 limitations.

6   Conclusions

In this paper we have proposed a novel approach to provisioning Cloud clusters to support data intensive applications over Hadoop YARN managed clusters. We have developed a mathematical programming formulation of the underlying optimization problem. In order to achieve a favorable trade-off between prediction accuracy and running times, we have adopted a heuristic approach that exploits the fast solvers available for mathematical programming problems for the initial exploration of the solution space and then relies on the precise, but slower, QN simulation. Moreover, our experimental validation shows how our tool is a valuable contribution towards identifying the best VM type, since we have highlighted situations where sticking to small instances and scaling out

proves to be less economic than switching to better equipped VMs that allow for a smaller number of replicas: the decreased replication factor compensates the increased unit price in a not obvious way.

Moving from the presented results, an interesting research direction for our future work lies in the characterization of complex workflows expressed as DAGs, e.g., Tez or Spark jobs. Another relevant aspect to investigate is the usage of more sophisticated techniques for the heuristic exploration of the solution space, in order to attain further speedup and, possibly, extend our method to the runtime cluster management scenario.

References

1. Amazon EC2 pricing. http://aws.amazon.com/ec2/pricing/
2. Amazon Elastic MapReduce. https://aws.amazon.com/elasticmapreduce/
3. Amazon Simple Storage Service. https://aws.amazon.com/s3/
4. Apache Hive. https://hive.apache.org
5. The digital universe in 2020. http://idcdocserv.com/1414
6. Hadoop MapReduce next generation — Capacity Scheduler. http://hortonworks.com/blog/benchmarking-apache-hive-13-enterprise-hadoop/
7. Microsoft Azure Storage. http://azure.microsoft.com/en-us/services/storage/
8. Microsoft HDInsight. http://azure.microsoft.com/en-us/services/hdinsight/
9. TPC-DS benchmark. http://www.tpc.org/tpcds/
10. Aleti, A., Buhnova, B., Grunske, L., Koziolek, A., Meedeniya, I.: Software architecture optimization methods: a systematic literature review. IEEE Trans. Softw. Eng. **PP**(99), 1 (2013)
11. Bardhan, S., Menascé, D.A.: Queuing network models to predict the completion time of the map phase of MapReduce jobs. In: International CMG Conference (2012)
12. Becker, S., Koziolek, H., Reussner, R.: The Palladio component model for model-driven performance prediction. J. Syst. Softw. **82**(1), 3–22 (2009)
13. Bertoli, M., Casale, G., Serazzi, G.: JMT: performance engineering tools for system modeling. SIGMETRICS Perform. Eval. Rev. **36**(4), 10–15 (2009)
14. Brosig, F., Meier, P., Becker, S., Koziolek, A., Koziolek, H., Kounev, S.: Quantitative evaluation of model-driven performance analysis and simulation of component-based architectures. IEEE Trans. Softw. Eng. **41**(2), 157–175 (2015)
15. Castiglione, A., Gribaudo, M., Iacono, M., Palmieri, F.: Exploiting mean field analysis to model performances of big data architectures. Future Gener. Comput. Syst. **37**, 203–211 (2014)
16. Ciavotta, M., Gianniti, E., Ardagna, D.: D-SPACE4Cloud: a design tool for big data applications. Technical report (2016). arXiv:1605.07083
17. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1990)
18. Greene, M.A., Sreekanti, K.: Big data in the enterprise: we need an "easy button" for Hadoop (2016)
19. Herodotou, H., Lim, H., Luo, G., Borisov, N., Dong, L., Cetin, F.B., Babu, S.: Starfish: a self-tuning system for big data analytics. In: CIDR (2011)
20. Jagadish, H.V., Gehrke, J., Labrinidis, A., Papakonstantinou, Y., Patel, J.M., Ramakrishnan, R., Shahabi, C.: Big data and its technical challenges. Commun. ACM **57**(7), 86–94 (2014)

21. Jha, S., Qiu, J., Luckow, A., Mantha, P., Fox, G.C.: A tale of two data-intensive paradigms: applications, abstractions, and architectures (2016). http://arxiv.org/abs/1403.1528

22. Kambatla, K., Kollias, G., Kumar, V., Grama, A.: Trends in big data analytics. J. Parallel Distrib. Comput. **74**(7), 2561–2573 (2014)

23. Koziolek, A., Koziolek, H., Reussner, R.: PerOpteryx: automated application of tactics in multi-objective software architecture optimization. In: QoSA 2011 Proceedings, QoSA-ISARCS 2011, pp. 33–42. ACM, New York (2011)

24. Lazowska, E.D., Zahorjan, J., Graham, G.S., Sevcik, K.C.: Quantitative System Performance. Prentice-Hall, Englewood Cliffs (1984)

25. Lee, K.H., Lee, Y.J., Choi, H., Chung, Y.D., Moon, B.: Parallel data processing with MapReduce: a survey. SIGMOD Rec. **40**(4), 11–20 (2012)

26. Luckow, A., Paraskevakos, I., Chantzialexiou, G., Jha, S.: Hadoop on HPC: integrating Hadoop and pilot-based dynamic resource management. In: 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 1607–1616 (2016)

27. Malekimajd, M., Ardagna, D., Ciavotta, M., Rizzi, A.M., Passacantando, M.: Optimal map reduce job capacity allocation in cloud systems. SIGMETRICS Perform. Eval. Rev. **42**(4), 51–61 (2015)

28. Morton, K., Friesen, A., Balazinska, M., Grossman, D.: Estimating the progress of MapReduce pipelines. In: ICDE (2010)

29. Morton, K., Balazinska, M., Grossman, D.: ParaTimer: a progress indicator for MapReduce DAGs. In: SIGMOD (2010)

30. OMG: PEPA: performance evaluation process algebra (2015). http://www.dcs.ed.ac.uk/pepa/tools/

31. Phan, L.T.X., Zhang, Z., Zheng, Q., Loo, B.T., Lee, I.: An empirical analysis of scheduling techniques for real-time cloud-based data processing. In: SOCA (2011)

32. Polo, J., Carrera, D., Becerra, Y., Torres, J., Ayguadé, E., Steinder, M., Whalley, I.: Performance-driven task co-scheduling for MapReduce environments. In: NOMS (2010)

33. Rao, B.T., Reddy, L.S.S.: Survey on improved scheduling in Hadoop MapReduce in Cloud environments (2012)

34. Tan, J., Wang, Y., Yu, W., Zhang, L.: Non-work-conserving effects in MapReduce: diffusion limit and criticality. In: SIGMETRICS (2014)

35. Tian, F., Chen, K.: Towards optimal resource provisioning for running MapReduce programs in public clouds. In: CLOUD (2011)

36. Tribastone, M., Gilmore, S., Hillston, J.: Scalable differential analysis of process algebra models. IEEE Trans. Softw. Eng. **38**(1), 205–219 (2012)

37. Verma, A., Cherkasova, L., Campbell, R.H.: ARIA: automatic resource inference and allocation for MapReduce environments. In: Proceedings of the Eighth International Conference on Autonomic Computing, June 2011

38. Verma, A., Cherkasova, L., Campbell, R.H.: Profiling and evaluating hardware choices for MapReduce environments: an application-aware approach. Perform. Eval. **79**, 328–344 (2014)

39. Vianna, E., Comarela, G., Pontes, T., Almeida, J.M., Almeida, V.A.F., Wilkinson, K., Kuno, H.A., Dayal, U.: Analytical performance models for MapReduce workloads. Int. J. Parallel Program. **41**(4), 495–525 (2013)

40. Yan, F., Cherkasova, L., Zhang, Z., Smirni, E.: Optimizing power and performance trade-offs of MapReduce job processing with heterogeneous multi-core processors. In: CLOUD (2014)

41. Zhang, W., Rajasekaran, S., Duan, S., Wood, T., Zhu, M.: Minimizing interference and maximizing progress for Hadoop virtual machines. SIGMETRICS Perform. Eval. Rev. **42**(4), 62–71 (2015)
42. Zhang, Z., Cherkasova, L., Loo, B.T.: Exploiting cloud heterogeneity to optimize performance and cost of MapReduce processing. SIGMETRICS Perform. Eval. Rev. **42**(4), 38–50 (2015)
43. Zhang, Z., Cherkasova, L., Verma, A., Loo, B.T.: Automated profiling and resource management of pig programs for meeting service level objectives. In: ICAC (2012)