Capacity Allocation for Big Data Applications in the Cloud

Michele Ciavotta
Politecnico di Milano,
Dipartimento di Elettronica,
Informazione e Bioingegneria.
Via Golgi 42 20133, Milano,
Italy
michele.ciavotta@polimi.it

Eugenio Gianniti
Politecnico di Milano,
Dipartimento di Elettronica,
Informazione e Bioingegneria.
Via Golgi 42 20133, Milano,
Italy
eugenio.gianniti@polimi.it

Danilo Ardagna
Politecnico di Milano,
Dipartimento di Elettronica,
Informatica e Bioingegneria.
Via Golgi 42 20133, Milano,
Italy
danilo.ardagna@polimi.it

ABSTRACT

The aim of this work is to present the problem of Capacity Allocation for multiple classes of Big Data applications running in the Cloud. The objective is the minimization of the renting out costs subject to the fulfillment of QoS requirements expressed in terms of application deadlines. We propose a preliminary version of a tool embedding a local-search-based algorithm exploiting also an integer nonlinear mathematical formulation and a queueing network simulation to solve the problem.

Keywords

Cloud; Big Data; QoS; Capacity Allocation.

1. INTRODUCTION

Nowadays, Big Data adoption has moved from experimental projects to mission-critical, enterprise-wide deployments providing competitive advantage and business innovation.

Hadoop is one of the widely adopted solutions to support Big Data applications, since it overtakes the scalability level of traditional data warehouse and business intelligence technologies and supports both traditional batch and interactive data analysis applications. However, the adoption of Big Data technologies is complex. The deployment and setup of an implementation is time-consuming, expensive, and resource-intensive. Companies need tools and methodologies to accelerate the deployment of Big Data analytics. For this reason, Cloud Computing is becoming a mainstream solution to provide large clusters on a pay-per-use basis.

Initially, MapReduce jobs were meant to run on dedicated clusters. Now, applications have evolved and large queries, submitted by different users, need to be performed on shared clusters, possibly with some guarantees on their duration. 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 [3].

ICPE '17 Companion, April 22–26, 2017, L'Aquila, Italy

DOI: http://dx.doi.org/10.1145/3053600.3053630

Our focus in this paper is to provide a software tool able to support operators in the capacity planning process of shared Hadoop Cloud clusters supporting applications with deadline guarantees. In a nutshell, we propose a search space exploration able to determine the optimal virtual machine (VM) type and the instance replicas for a set of concurrent applications, considering specific Cloud provider pricing models. The underlying optimization problem is demonstrated to be NP-hard and it is solved heuristically, whereas job execution times are estimated via queueing network (QN) models.

This paper is organized as follows. In Section 2 we present the problem in more details, whilst the proposed tool is introduced in Section 3. Finally, some conclusions are drawn in Section 4.

2. PROBLEM STATEMENT

We envision the following scenario wherein a company needs to set up a cluster to efficiently carry out a set of interactive Big Data queries. A Hadoop cluster featuring the YARN Capacity Scheduler and running on public IaaS Cloud is considered a fitting technological solution.

In particular, the cluster must support the parallel execution of Big Data applications in the form of concurrent jobs. Different classes 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 executed concurrently and complete before a prearranged deadline D_i. Finally, in order to reduce the risk of data corruption and according to the practices suggested by major Cloud vendors, the datasets reside on an external storage infrastructure.

As, in general, IaaS providers feature a limited, but possibly large, catalog of VM configurations 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. In this scenario, a pricing model inspired by *Amazon EC2* [1] is considered. 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; 2) *on demand* VMs, which can be leased with no long term commitments, but at a relatively expensive hourly price; and 3) *spot* VMs, for which customers bid and compete for unused datacenter capacity, yielding very competitive hourly fees.

[1] https://aws.amazon.com/ec2/pricing/

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 Capacity Allocation problem where the renting out costs must be minimized guaranteeing that certain deadlines D_i are met. 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 Z_i, i.e., the system is represented as a closed model subject to a terminal workload. 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 [3], it is possible to derive from the Hadoop logs a *job profile*, which is a concise behavior characterization for each class. Given the amount and type of resources allocated, the concurrency level, and the job profile, the execution time can be obtained with two main approaches: either evaluating approximate closed formulas, as those presented in [2], or via the simulation of QNs, as done in this paper.

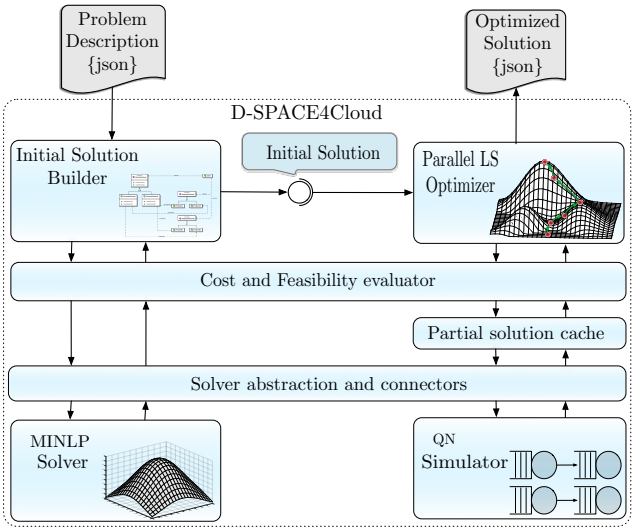

Figure 1: D-SPACE4Cloud architecture

3. D-SPACE4CLOUD

The tool implements an optimization mechanism that efficiently explores the space of possible configurations, henceforth referred to as *Solution space*. Figure 1 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 of 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 for each considered application. The *Initial Solution Builder* generates a starting solution for the problem using a Mixed Integer Nonlinear Programming (MINLP) formulation where the job duration is expressed by means of a convex function: [1] provides further details. The fast MINLP model is exploited to determine the most cost effective VM type for all applications. Yet the quality of the returned solution can still be improved, since the MINLP problem is just an approximate model. For this reason, a more precise QN model is adopted to get a more accurate execution time assessment for each application: the increased accuracy leaves room for further cost reduction. However, since QN simulations are 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 LS 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. In particular, Hill Climbing is a local-search-based procedure that operates on the current solution performing a change (more often referred to as *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 for the problem. This is the case of the considered cost function, which depends linearly on the number of VMs in the cluster, since the VM types to use are fixed in the first phase of the optimization process based on MINLP texhniques. In other words, the joint selection of the VM type and their number is NP-hard, but when the type of VM is fixed in the first phase, the HC obtains the final solution for all classes in polynomial time.

4. CONCLUSIONS

In this paper we briefly introduced the problem of capacity allocation in Big Data Cloud clusters with multiple concurrent applications and a tool to effectively tackle it at design time. The tool currently considers only public Clouds and simple MapReduce jobs, but we are actively working to support DAGs, as well as private Clouds.

Acknowledgments

The research reported in this article is partially supported by the EU grant no. H2020-ICT-2014-1-64486 (DICE).

5. REFERENCES

[1] M. Malekimajd, D. Ardagna, M. Ciavotta, A. M. Rizzi, and M. Passacantando. Optimal Map Reduce job capacity allocation in Cloud systems. *SIGMETRICS Perform. Eval. Rev.*, 42(4):51–61, June 2015.

[2] M. Malekimajd, A. M. Rizzi, D. Ardagna, M. Ciavotta, M. Passacantando, and A. Movaghar. Optimal capacity allocation for executing MapReduce jobs in Cloud systems. In *SYNASC*, 2014.

[3] A. Verma, L. Cherkasova, and R. H. Campbell. ARIA: Automatic resource inference and allocation for MapReduce environments. In *ICAC*, 2011.