Leveraging cloud unused resources for Big data application while achieving SLA

Jean-Emile Dartois, Ivan Meriau, Mohamed Handaoui, Jalil Boukhobza, Olivier Barais

▶ **To cite this version:**

Jean-Emile Dartois, Ivan Meriau, Mohamed Handaoui, Jalil Boukhobza, Olivier Barais. Leveraging cloud unused resources for Big data application while achieving SLA. MASCOTS 2019 - 27th IEEE International Symposium on the Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, Oct 2019, Rennes, France. pp.1-2. hal-02362257

HAL Id: hal-02362257
https://inria.hal.science/hal-02362257

Submitted on 13 Nov 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Leveraging cloud unused resources for Big data application while achieving SLA

Jean-Emile Dartois*†, Ivan Meriau*§, Mohamed Handaoui*‡, Jalil Boukhobza*‡, and Olivier Barais*†
*b<>com Institute of Research and Technology, †Univ Rennes, Inria, CNRS, IRISA, ‡Univ. Bretagne Occidentale,§Orange
Email: jean-emile.dartois@b-com.com, ivan.meriau@b-com.com, boukhobza@univ-brest.fr, barais@irisa.fr

***Abstract*—In this demo paper, we present an architecture that leverages unused but volatile Cloud resources to run big data jobs. It is based on a learning algorithm that accurately predicts future availability of resources to automatically scale the ran jobs. We also designed a mechanism that avoids interference between the Big data jobs and co-resident workloads. Our solution is based on Open-Source components such as kubernetes and Apache Spark.**

***Keywords*-cloud, unused resources, spare resources,big data, Apache Spark, autoscaler, kubertenes.**

I. Introduction

According to recent estimations [1] by 2025 the amount of data generated by humanity will be about 160 Zettabytes. To process this data, many companies rely on cloud platforms that have large-scale physical resources. From the customers' point of view, cloud platforms have numerous benefits such as on-demand access to scalable, elastic, reliable computing resources. From a cloud provider's (*CPs*) view, the main objectives is to ensure a good quality of service (QoS) for customers while reducing their TCO.

Managing resources in order to improve their utilization and reduce costs is a major concern for (*CPs*). Although the use of virtualization has improved the use of computing resources in data centers [2], several studies have demonstrated that the average usage of resources remains low, between 20% to 50% in case of CPU [3]. To address such an issue, making profit of those unused resources appears to be a very interesting solution to optimize the total cost of ownership.

We work on a project that aims to make unused and heterogeneous private IT resources available through a highly secured distributed Cloud to deploy applications at a cheaper price. The first use case of the project is to provide a framework that leverages unused Cloud resources to deploy big data jobs. The main challenges to leverage cloud unused resources for deploying such jobs while achieving SLA are the following:

Users SLA guarantee: When running applications on allocated but unused resources, *CPs* should hedge against violating SLA for the users having reserved those resources. Thus, *CPs* must be able to quickly allocate, react and adapt the unused resource provisioning in a way to avoid degrading the QoS for the users that have reserved those resources.

Resources volatility: In Cloud systems, users are able to unilaterally reserve, consume and release computing resources on-the-fly. On the other hand, the nature of workloads is highly heterogeneous and their intensity may significantly vary according to users behavior. One of the challenge is to be able to predict those variations.

Cloud heterogeneity: Cloud infrastructures are built upon heterogeneous resources to avoid vendors lock-in effect and due to frequent hardware updates. Resources reclamation need to be flexible to the storage and processing capacities.

To tackle these challenges, we first propose an architecture that leverages unused Cloud resources to efficiently run Apache Spark without interfering with co-located workloads. Second, we implement this architecture using on-the-shelves Open-Source components, namely kubernetes, Apache Spark, and Zeppelin (see Section II).

We aim to demonstrate how an organization can reclaim its resources for its own needs to deploy Apache Spark and process big data. We also show that Apache Spark can automatically scale-in/out depending on predicted future availability. We show the big data users' point of view using Zeppelin, with a specific example of word counting functionality. We finally deliberately inject unpredictable workloads to highlight that the Node Evictor is preventing all interferences on regular users' workloads.

II. On the shelves Open-source components used

Kubertenes (commonly referred to k8s) offers an easy way to automate and deploy containers on an elastic infrastructure with a fine container orchestration [4]. Also, K8s provide a way to divide cluster resources between multiple teams, or projects using the *Namespaces* concept. k8s proposes by default three Quality of Service for containers (*i.e.,* Guaranteed, Burstable, and Best-Effort). When kubernetes creates a container it assigns one of these QoS classes. Note that kubertenes makes a clear distinction between *Compressible resources* that can be throttled (the user will be slowed down proportionally to the throttling, but will otherwise proceed normally), and *Incompressible resources* that cannot be throttled without causing failure (*e.g.*, for memory).

Apache Spark is a framework that is used for processing, querying and analyzing Big data. Apache Spark framework uses a master/slave architecture. The master is the coordinator and many distributed slaves are providing the resources. Cloud unused resource reclamation techniques are ideal for running delay-tolerant batch as the ones proposed by Spark, that are requiring large amounts of computing resources. Also, Apache Spark provides by default many interesting properties such as data locality and fault-tolerance techniques (*e.g.,* data replication or node failure) that mitigate the impact of volatility.

Zeppelin aims to provide a web interface (*i.e.,* notebook) to easily analyze and format, visually and interactively, large

volumes of data processed via the distributed computing framework such as Apache Spark.

III. An architecture to leverage cloud unused resources

The demo uses four mechanisms we have designed (i) a Forecasting builder to predict resource volatility, (ii) a spare resource allocator (iii) a Node Evictor to ensure users SLA guarantee by avoiding interference (iv) and a predictive autoscaler to automatically scale in/scale out Apache Spark orchestrated by kubernetes(see Figure 1).

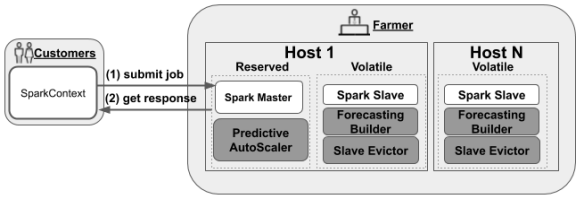

Figure 1. Overall project overview

A. Forecasting builder

The objective of the *Forecasting Builder* module is to estimate the future amount of used resources for each host. By doing so, it can to estimate the available resources for running Apache Spark Slaves. In a previous work [3] we have shown that quantile regression may increase the amount of savings by up to 20% compared to traditional approaches. In this demo, we have re-used the same configuration proposed in [3].

B. Spare Resources Allocator

The goal of the Spare Resources allocator is to automatically allocate unused resources to Apache Spark of *ephemeral customers*.

To achieve that, we introduced a new class of QoS in kubernetes called *non-production*. This class aims to avoid any interference on regular workloads even *best-effort* ones. This QoS class is automatically reclaiming unused resources for allocating them to Apache Spark. The non-production QoS class relies on a specific configuration of the Linux kernel module called *cgroup*. The *cgroup* functionality offers the possibility to limit and prioritize resource usage (*e.g.,* CPU, block I/O, network, etc.) for each container.

Figure 2 shows our configuration of cgroups for the CPU. The same configuration is applied for I/O and network resources. The *root* and *spare allocator* groups receive all the amount of available processor time. Then, *non-production* group is configured for receiving only the amount of available processor time that is not consumed by the *kubepods* group. Then, the configuration is the same as the default kubernetes (*e.g.,* the child container(s) of *kubepods* group in yellow belong to the guaranteed class.

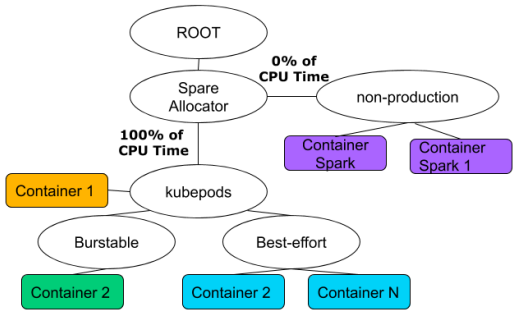

Figure 2. cgroups hierarchy

C. Slave Evictor

The Slave Evictor implements a mechanism that reacts to underestimation of the used resources from the Forecasting builder. This is critical in a context of incompressible compute resources, such as memory or disk space. In a previous work [5], we proposed a mechanism that proportionally kills applications enabling to free the safety margin (*i.e.,* Keep a portion of CPU or disk space unused to prevent interference related to memory occupation or disk occupation)

D. Predictive AutoScaler

This module uses the predictions of the Forecasting builder to control all spark scaling activities for each physical machine. The safety margin value (described in Section III-C) is also taken into account.

IV. Conclusion and Future work

In this paper, we presented an architecture based on the shelves components (i.*e.,* kubernetes, Apache Spark, Zeppelin) to leverage cloud unused resources. As future work we plan to use reinforcement learning to decrease the impact of unused resources volatility on QoS by including in our model reserved resources. We would expect by this approach to reach a compromise between costs management and impacts on QoS without prior knowledge.

Acknowledgment

This work was supported by the Institute of Research and Technology b-com, dedicated to digital technologies, funded by the French government through the ANR Investment referenced ANR-A0-AIRT-07.

References

[1] IDC/Seagate, "Data age 2025; the evolution of data to life-critical." Website, 2017. Accessed May, 1st, 2019.

[2] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via vm multiplexing," in *Proceedings of the 7th international conference on Autonomic computing*, pp. 11–20, ACM, 2010.

[3] J.-E. Dartois, A. Knefati, J. Boukhobza, and O. Barais, "Using quantile regression for reclaiming unused cloud resources while achieving sla," in *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 89–98, IEEE, 2018.

[4] K. Hightower, B. Burns, and J. Beda, *Kubernetes: Up and Running Dive into the Future of Infrastructure*. O'Reilly Media, Inc., 1st ed., 2017.

[5] J.-E. Dartois, H. B. Ribeiro, J. Boukhobza, and O. Barais, "Cuckoo: a mechanism for exploiting ephemeral and heterogeneous cloud resource," in *IEEE International Conference on Cloud Computing*, IEEE, 2019.