Towards Control of MapReduce Performance and Availability

Sophie Cerf, Mihaly Berekmeri, Bogdan Robu, Nicolas Marchand, Sara Bouchenak

Towards Control of MapReduce Performance and Availability

Sophie Cerf[1], Mihaly Berekmeri[1], Bogdan Robu[1], Nicolas Marchand[1] and Sara Bouchenak[2]
[1] Univ. Grenoble Alpes, CNRS, GIPSA-lab, F-38402 Grenoble, France
[2] LIRIS UMR 5205, INSA de Lyon, France
{sophie.cerf, mihaly.berekmeri, bogdan.robu, nicolas.marchand}@gipsa-lab.fr, sara.bouchenak@insa-lyon.fr

***Abstract*—MapReduce is a popular programming model for distributed data processing and Big Data applications. Extensive research has been conducted either to improve the dependability or to increase performance of MapReduce, ranging from adaptive and on-demand fault-tolerance solutions, adaptive task scheduling techniques to optimized job execution mechanisms. This paper investigates a novel solution that controls MapReduce systems and provides guarantees in terms of both performance and availability, while reducing utilization costs. We follow a control theoretic approach for MapReduce cluster scaling and admission control. Preliminary results based on a simulation environment, previously validated on a real MapReduce cluster, show the effectiveness of the proposed control solutions for a Hadoop MapReduce cluster.**

I. Introduction

The amount of data produced by everything from mobile phones, tablets, computers to smart watches brings novel challenges in data storage and analysis. Many solutions have arisen in research and industry to handle these large amounts of data. MapReduce is a popular programming model and execution environment for developing and executing distributed data-intensive and computer-intensive applications [3]. However, the complexity of configuration of such a system is continuously increasing while the user expectations remain the same: continuous availability and fast response times are the required norm. Moreover, with the advent of cloud solutions, the environments where these systems need to run is becoming more and more dynamic.

Ensuring performance and dependability of MapReduce systems still poses several challenges. Although the framework hides the complexities of parallelism to the users, deploying an efficient MapReduce implementation poses multiple challenges. Furthermore many factors have been identified that negatively influence the performance of MapReduce jobs (CPU, input/output and network skews, hardware and software failures, node homogeneity assumption not holding up, bursty workloads) and extensive research has been conducted to improve dependability or performance of MapReduce by changing the behavior and/or algorithms of the MapReduce framework itself (see [7] for instance). Although these solutions improve upon how MapReduce works, no guarantees are provided in term of performance and availability. Some solutions for performance modeling [5] and control [2] can be found in the literature. However to the best of our knowledge, there are no work to provide concurrent guarantees in terms of combined dependability and performance, while reducing utilization costs. The presented work is in line with our previous modeling and control (see [1]) by extending it to optimaly deal with performance-availability-costs compromise.

To that end, we design and implement the first optimal control algorithm for MapReduce systems that ensures the desired service availability and performance while continuously minimizing costs as well. Preliminary simulation based on a model validated on a real MapReduce system shows the soundness of the approach.

II. Background and Motivation

MapReduce is a programming paradigm developed for parallel, distributed computations over large amounts of data. It has been developed in 2008 by Google to automatically handle most of the complexity of parallel computing and nowadays it is backed by Big Data industry leaders such as Facebook, Yahoo or LinkedIn (see [4], [3] among others). The most used open source implementation of the MapReduce programming model is Hadoop [6].

The choice of control variables out of Hadoop's many parameters (more than 170) is not straightforward. After an in depth analysis of Hadoop's behavior (see [1]), we chose the MapReduce cluster size (N) and the the maximum number of admitted clients (MC) as our tunable signals that we can use to control MapReduce behavior, as they are known to have a high influence on both service availability and performance. MapReduce admission control is a classical technique to prevent server thrashing since it consists of limiting the maximum number of clients that are allowed to concurrently send requests to the central controller.

One of the important challenges in current MapReduce deployments is assuring certain service time and availability thresholds for jobs while minimizing cost. These specifications are given as the maximum response time Rt_{max} and the minimum availability Av_{min} to be guaranteed by the MapReduce system.

III. Proposed Solution

We proceed in two steps: first we study the impact of control signals on measured variables, which is called modeling and second, based on this model, we design the control algorithm which, after taking into account the system states and its environment, automatically decides the values of the control variables. A schematic overview of this approach is shown in Fig. 1.

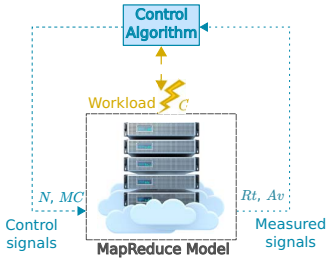

Fig. 1. MapReduce control schema

a) Modeling MapReduce: Capturing the complex behavior of MapReduce systems is highly challenging. A model that captures the dynamics of MapReduce systems and can be used to predict performance and availability levels was already validated in [1]. The model is built as a set of difference equations that describe the impact of changes in the control variables on system's outputs.

The input variables of the MapReduce model are the number of nodes N of the MapReduce cluster and the maximum number of clients MC concurrently admitted. The workload C (the number of concurrent clients that are sending requests to the central controller) is considered as an exogenous signal that we can not control. The outputs, measured variables, which we desire to keep at certain levels, are the average response time Rt of a MapReduce client request and the availability Av level of MapReduce to its clients. Note that there is no limit on job size or nature in our modeling process, we only assume that variance in job properties is no more than 25%.

b) Control of MapReduce: The aim of our control algorithm is to guarantee a given performance and availability, using a minimal amount of resources, even when the system face jumps in workload. This 3-fold objective consists the first major improvement upon our previous work [1]. For that purpose, we define a *cost function* that will be minimized, which has two components: the first one is the difference between the response time (resp. the availability) and the reference values (Rt_{max} and Av_{min}), while the second one is composed of the cluster size (N). Moreover, each of the above mentioned components are in fact considered over a time horizon, called *prediction horizon*, over which the model presented in the previous section is used to foretell the behavior of the MapReduce system. The algorithm chooses and periodically updates a profile for the control signals which minimizes this cost function over the whole prediction horizon, that is to say a profile which guarantees performance and availability while reducing the resource consumption N. This *optimal* computation over a time horizon is the second major advance of our work. Furthermore, by adding weights to the different components of the cost function, we can give more importance to particular specifications, as these components can express contradictory requirements (availability vs. performance trade-off for instance).

IV. Preliminary Results

For preliminary validation, we simulated the behavior of a MapReduce framework with and without our control strategy under a changing workload. For this example, we decide that the availability should not be less than 98% *and* the response time should be at most 120 seconds *while* always minimizing the number of nodes. When the workload is too constraining and these specifications cannot be fulfilled, we give higher priority to the response time constraint and momentarily leave out constraints on control signal N. However, this priority order can be changed at any time.

Fig. 2 shows the results of our simulations. We see that our strategy enables to have the desired availability while significantly improving MapReduce performance by scaling the cluster and rejecting some requests according to workload fluctuations. Without any control strategy all specifications can not be kept at the same time.

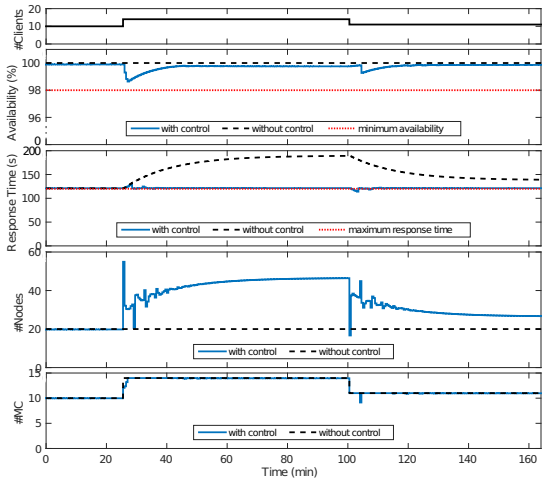

Fig. 2. Comparison of MapReduce performance and availability with and without control ($Av_{min} = 98\%$ and $Rt_{max} = 120\ s$)

V. Conclusion & Future Works

In order to improve both performance and availability of a MapReduce deployment subject to a changing workload we introduce a control strategy that, through mathematical optimization, scales the resources of the cluster and performs admission control. The solution we develop is a small add-on to the system that runs in parallel of MapReduce in a non intrusive way. We simulated our control algorithm using a previously on-line validated model, results are promising as they show significant performance improvements of MapReduce, while guaranteeing almost full availability and minimizing resource usage.

Future works are being investigated that aim at validating our control strategy using a real MapReduce deployment. Moreover, we would like to improve our algorithm by reducing the number of cluster reconfigurations, as it is a costly issue in cloud computing, while still providing mathematical performance and availability guarantees through formal optimization.

References

[1] M. Berekmeri, D. Serrano, S. Bouchenak, N. Marchand, and B. Robu. Feedback autonomic provisioning for guaranteeing performance in mapreduce systems. *IEEE Transactions on Cloud Computing*, 2016.

[2] M. Cardosa, P. Narang, A. Chandra, H. Pucha, and A. Singh. Steamengine: Driving mapreduce provisioning in the cloud. In *High Performance Computing (HiPC), 2011 18th International Conference on*, pages 1–10. IEEE, 2011.

[3] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[4] Y. Shen. *Enabling the New Era of Cloud Computing: Data Security, Transfer, and Management: Data Security, Transfer, and Management*. IGI Global, 2013.

[5] A. Verma, L. Cherkasova, and R.H. Campbell. Resource provisioning framework for MapReduce jobs with performance goals. In *Middleware 2011*, volume 7049 of *Lecture Notes in Computer Science*, pages 165–186. Springer Berlin Heidelberg, 2011.

[6] T. White. *Hadoop: the definitive guide*. O'Reilly Media, CA, 2012.

[7] M. Zaharia, M. Chowdhury, M. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10:10–10, 2010.