Chapter 1

On the Dynamic Shifting of the MapReduce Timeout

Bunjamin Memishi
Universidad Politecnica de Madrid, Spain

Maria S. Perez
Universidad Politecnica de Madrid, Spain

Shadi Ibrahim
Inria, France

Gabriel Antoniu
Inria, France

ABSTRACT

MapReduce has become a relevant framework for Big Data processing in the cloud. At large-scale clouds, failures do occur and may incur unwanted performance degradation to Big Data applications. As the reliability of MapReduce depends on how well they detect and handle failures, this book chapter investigates the problem of failure detection in the MapReduce framework. The case studies of this contribution reveal that the current static timeout value is not adequate and demonstrate significant variations in the application’s response time with different timeout values. While arguing that comparatively little attention has been devoted to the failure detection in the framework, the chapter presents design ideas for a new adaptive timeout.

INTRODUCTION

The ever growing size of data (i.e., Big Data) has motivated the development of data intensive processing frameworks and tools. In this context, MapReduce (Dean & Ghemawat, 2004; Jin et al., 2011) has become a relevant framework for Big Data processing in the clouds, thanks to its remarkable features including simplicity, fault tolerance, and scalability. The popular open source implementation of MapReduce, Hadoop (Apache Hadoop Project, 2015), was developed primarily by Yahoo!, where it processes hundreds of Terabytes of data on at least 10,000 cores, and is now used by other companies, including Facebook, Amazon, Last.fm, and the New York Times (Powered By Hadoop, 2015).

Undoubtedly, failure is a part of everyday life, especially in current data-centers which comprise thousands of commodity hardware and software (Chandra, Prinja, Jain, & Zhang, 2008; Oppenheimer, Ganapathi, & Patterson, 2003; Pinheiro, Weber, & Barroso, 2007). Consequently, MapReduce was designed with hardware failure in mind. In particular, Hadoop tolerates machine failures (crash failures)

DOI: 10.4018/978-1-4666-9767-6.ch001
by re-executing all the tasks of the failed machine by the virtue of data replication. Furthermore, in order to mask temporary failures caused by network or machine overload (timing failure) where some tasks are performing relatively slower than other tasks, Hadoop re-launches other copies of these tasks on other machines.

Foreseeing MapReduce usage in the next generation Internet (Mone, 2013), a particular concern is the aim of improving the MapReduce’s reliability by providing better fault tolerance mechanisms. While the handling and recovery in MapReduce fault-tolerance via data replication and task re-execution seem to work well even at large scale (Ko, Hoque, Cho, & Gupta, 2010; Ananthanarayanan et al., 2011; Zaharia, Konwinski, Joseph, Katz, & Stoica, 2008), there is relatively little work on detecting failures in MapReduce. Accurate detection of failures is as important as failures recovery, in order to improve applications’ latencies and minimize resource waste.

At the core of failure detection mechanism is the concept of heartbeat. Any kind of failure that is detected in MapReduce has to fulfill some preconditions, in this case to miss a certain number of heartbeats, so that other entities in the system detect the failure. Currently, a static timeout based mechanism is applied for detecting fail-stop failure by checking the expiry time of the last received heartbeat from a certain machine. In Hadoop, each TaskTracker sends a heartbeat every 3s, the JobTracker checks every 200s the expiry time of the last reported heartbeat. If no heartbeat is received from a machine for 600s, then this machine will be labeled as a failed machine and therefore the JobTracker will trigger the failure handling and recovery process. However, some studies have reported that the current static timeout detector is not effective and may cause long and unpredictable latency (Dinu & Ng, 2011, 2012).

An accurate timeout detector is important not only to improve application’s latency but also to improve resource utilization, especially in the cloud where you pay for the resources you use. Accordingly, a series of experiments is conducted to measure the performance of Hadoop with different timeout values while varying the application input sizes and the failure injection times. The authors find that, in the presence of single machine failure the applications’ latencies vary not only in accordance to the occurrence time of the failure, similar to (Dinu & Ng, 2012), but also vary with the job length (short or long). Furthermore, the experimental results report not only a noticeable variation of the application’s latency with different timeout value, but also demonstrate the opportunity to achieve better performance. For example, when using a small timeout value of 10s, it is achieved a 3X performance improvement for the sort benchmark compared to the default one. Thus, a significant potential exists for performance improvement in MapReduce applications, when choosing the appropriate timeout failure detector. The authors believe that a new methodology to adaptively tune the timeout detector, at runtime and according to the job progress, can significantly improve the overall performance of MapReduce applications under failure. Therefore, they discuss three potential fine-tuning approaches that could be used towards an optimal timeout failure detector in MapReduce.

In order to clarify some preliminary knowledge, the background section comes first. Then, it follows the section of general fault tolerance in Hadoop. The timeout problem is described in details after this. The methodology overview comes next, which is closely linked with experimental analysis section. There are given open challenges in the following section. A related work on improving fault-tolerance mechanism in MapReduce is presented after, before concluding the chapter with the summary section.
20 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the product's webpage: www.igi-global.com/chapter/on-the-dynamic-shifting-of-the-mapreduce-timeout/143336?camid=4v1


Related Content

Fog Computing Qos Review and Open Challenges
www.igi-global.com/article/fog-computing-qos-review-and-open-challenges/210568?camid=4v1a

Security Issues of Cloud Computing and an Encryption Approach
www.igi-global.com/chapter/security-issues-of-cloud-computing-and-an-encryption-approach/102420?camid=4v1a

Workload Migration to Cloud
www.igi-global.com/chapter/workload-migration-to-cloud/168158?camid=4v1a

Novel Taxonomy to Select Fog Products and Challenges Faced in Fog Environments