Poster: BDTest, a System to Test Big Data Frameworks

Alexandre Langeois*†, Eduardo Cunha de Almeida‡, Anthony Ventresque*
*Lero@UCD, School of Computer Science, University College Dublin, Ireland Email: anthony.ventresque@ucd.ie
†Ecole Centrale de Nantes, France. Email: alexandre.langeois@eleves.ec-nantes.fr
‡C3SL Labs, UFPR, Brazil. Email: eduardo@inf.ufpr.br

***Abstract*—Testing Big Data Processing systems is a challenging task as these systems are usually distributed on various virtual machines (potentially hosted by remote servers). In this poster[1] we present a platform for testing non-functional properties of Hadoop, a well known big data management and processing platform.**

I. Introduction

"Big Data" has become a reality in the past years, with enormous amounts of data generated humans (e.g., Soocial Networks) and by all sorts of sensors (e.g., Internet of Things). For instance, it is estimated that we create 2.5 quintillion bytes of data every day and that 90% of all data has been created in the last two years [1]. While managing and storing large amount of data has been a focus of interest, processing them is challenging as it is skill and labour intensive [2]. MapReduce [3] and similar frameworks have been proposed to simplify the ingestion, storage and processing of large amount of data on distributed architectures. For instance Hadoop [4] is extensively used by companies by companies, for instance Ebay, Facebook or Google [5].

The quality of such software systems is difficult to assess though, as testing any distributed systems is a known complex challenge. Issues in a distributed software system can come from various elements: software or hardware components, their integration, or the communication between components [6]. Another risk is also to introduce artificial defects or to impact the performance of the system with the monitoring apparatus.

In this poster we present BDTest, a system that aims at testing Big Data Platforms, such as, Hadoop. BDTest uses a lower tester architecture [7] which has a limited impact on the system under test (SUT), while offering the possibility to run complex test scenarios. While BDTest could test functional properties, we focus on non-functional ones - for instance, changing artificially the performance or availability of computing nodes in the SUT, i.e., only for the test scenarios. We plan to address in our future work a variety of research questions, such as:

- Can we test the performance of the partitioning and load balancing mechanisms of Hadoop?
- Can we discover performance bottlenecks in Hadoop using BDTest and, for instance, model-based testing techniques?
- Can we identify portability issues of Hadoop using our testing system and heterogeneous platforms?

[1]This work was supported by Science Foundation Ireland grant 13/RC/2094.

II. Related Work

The increased interest in and use of Big Data platforms like Hadoop has led to a lot of research and the development of testing techniques and tools. One of the most interesting attempts is MRUnit [8], a Java framework that aims at creating JUnit [9] tests like test cases for MapReduce jobs. However, this tool requires to introduce specific calls to its API in the SUT and can generate new bugs or decrease performance of the SUT. BDTest tries to avoid any modification of the SUT.

PigUnit [10] is another interesting tool which uses the unit test model with some success. However, PigUnit works only for Pig, one of the many applications/modules of the Hadoop ecosystem. On the contrary, BDTest being independent of the exact running systems, should be able to interact with any Big Data platform - and in particular any Hadoop module.

Testing distributed systems has been studied quite a lot in the past, however with poor success at times. GridUnit [11], [12] proposed a mixed model: a centralised architecture with the execution of tests on the node level, but GridUnit does not handle the volatility of computing nodes, which is an important element in distributed systems. P2PTester [13] and Pigeon [14] proposed to use a fully distributed architecture but they require to modify the code of the SUT which brings the same limitations as MRUnit for Big Data systems. PeerUnit [15], [7] is a project that inspired BDTest. PeerUnit creates complex unit tests for distributed (peer-to-peer) systems without modifying the SUT. Test scenarios in PeerUnit are synchronised over all the computing nodes using either a centralised or a decentralised architecture [16] - PeerUnit can work using both modes. BDTest follows some of the general ideas proposed by PeerUnit, but we extend PeerUnit in two original directions: (i) BDtest can manage parallel executions (typical of Big Data tasks) and (ii) BDTest can run heterogeneous nodes. HadoopTest [17] is a framework based on PeerUnit created to test Hadoop applications. HadoopTest caters for two types of testers: master and worker (as does BDTest), but can only create a centralised testing architectures. HadoopTest focuses on functional properties, while BDTest aims at addressing also non-functional properties such as, scalability or performance. For instance, HadoopTest can drop nodes (as it is also the case in PeerUnit) while in BDTest we can modify more finely

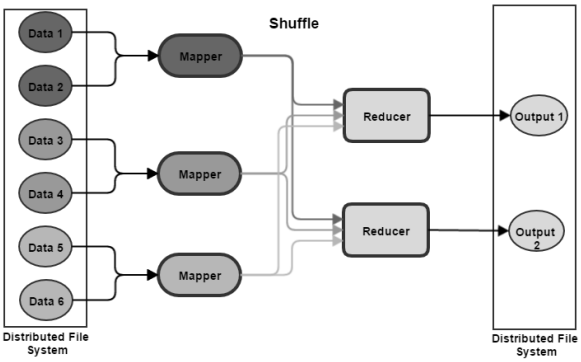

Fig. 1. Execution of Map-Reduce operations

all the properties of each computing node (e.g., constrain the hardware resources such as, CPU or memory).

III. HADOOP

MapReduce uses the potential of parallel execution, it is based on two main functions which are written by the user: Map and Reduce. First the Map function analyses parts of the data and sorts them, then the Reduce function computes the output of the algorithm. There are two types of nodes, the Master which assigns tasks to workers and workers which handle the Map or Reduce function when needed (see Figure 1).

Hadoop is the Apache open-source implementation of Map-Reduce, it is composed of two basic parts: Hadoop MapReduce to process the data and HDFS (Hadoop Distributed File System) the storage part. Apache Hadoop also refers to the whole ecosystem with programming tools (Apache Pig) or distributed database (Hbase).

IV. BDTEST

BDTest is developed in Java and has a Hadoop interface for the moment - we will provide interfaces to other Hadoop components in the future to provide a full library of Big Data testing tools. Each Hadoop node (i.e., master or worker) is coupled to one tester. Those testers can monitor and instrument both the system being tested and the testing system. The testers can for instance limit the resource usage of the system (e.g., cap the CPU utilisation) and run part of global scenarios (e.g., stress test the load balancing algorithm in Hadoop/HDFS). The testers do not modify the SUT when those scenarios are implemented - they merely virtualise the interface between the SUT and the Operating System. BDTest also provides a coordinator agent, which dispatches the actions (of a test case) to the relevant testers. The coordinator also manages the set of testers and in particular synchronises their actions.

BDTest, while implementing a centralised architecture at the moment (see Fig. 2), can run on a centralised or decentralised mode. In the former, BDTest has only one coordinator, which gives actions to all the testers and is responsible for the parallel execution of the tests (see Fig. 2). In the decentralised architecture on the other hand, all testers (but the testers on the

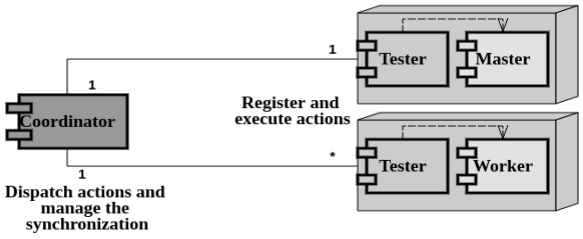

Fig. 2. BDTest (centralised architecture)

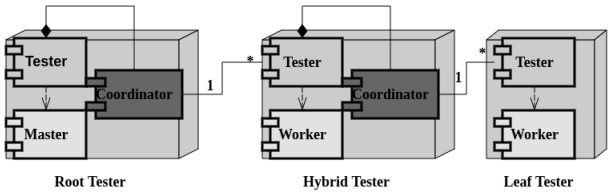

Fig. 3. BDTest (decentralised architecture)

leaves) belong to a tree where testers are also coordinators and give actions to their descendants and report to their parents. The testers subscribe to their parent – their 'coordinator' (see Fig. 3). The root node kicks-off the test cases, communicate with its children which in turn forward the actions on to their own children and so on until the leaves of the tree. This decentralised architecture scales up better compared to the centralised one and limits the communication overhead.

Validating global properties is one of the key challenges in testing distributed systems [18]. BDTest makes sure that each tester manages its own data which is later aggregated in to a global view (a test oracle) by the coordinator (centralised architecture) or the root node (decentralised architecture).

BDTest implements assertions and value comparisons as they are the easiest testing methods [19], [20]. Log file (offline) analysis is not yet implemented in BDTest but this is a potential direction of research.

V. CONCLUSION

This poster presents BDTest, a system to test Big Data platforms, such as Hadoop. Software testers using BDTest create test scenarios using simple unit tests (assertions and value comparisons) and BDTest manages the dissemination of the actions to every tester (each in charge of a computing node). BDTest can simulate all sorts of defects (e.g., performance defects) and as such can be used to test non-functional properties as well as functional characteristics of Big Data platforms.

As a future direction, we would like to extend the number of Big Data modules (from the Hadoop galaxy) that BDTest can work on - e.g., Spark [21]. We would also like to evaluate the performance of Big Data platforms on different architectures and see whether BDTest can be used for benchmarking as well as for testing.

REFERENCES

[1] "What is big data?" https://www-01.ibm.com/software/data/bigdata/what-is-big-data.html.

[2] M. Stonebraker, S. Madden, D. J. Abadi, S. Harizopoulos, N. Hachem, and P. Helland, "The end of an architectural era: (it's time for a complete rewrite)," in *VLDB*, 2007, pp. 1150–1160.

[3] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *OSDI*, pp. 137–150, 2004.

[4] The Apache Software Fondation, "Apache Hadoop," 2017, http://hadoop.apache.org/.

[5] ——, "Who use Hadoop," 2017, https://wiki.apache.org/hadoop/PoweredBy.

[6] X. Bai, M. Li, B. Chen, W.-T. Tsai, and J. Gao, "Cloud Testing Tools," *SOSE*, 2011.

[7] E. C. de Almeida, "Testing and validation of peer-to-peer systems," Ph.D. dissertation, University of Nantes, 2009.

[8] The Apache Software Fondation, "Apache MRUnit," 2016, https://mrunit.apache.org/.

[9] "JUnit," 2017, http://junit.org/junit4/.

[10] The Apache Software Fondation, "JUnit," 2017, https://pig.apache.org/.

[11] A. Duarte, W. Cirne, F. Beasileiro, and P. Machado, "Using the computational grid to speed up software testing," *BSSE*, 2005.

[12] ——, "Gridunit: software testing on the grid," *ICSE*, pp. 779–782, 2006.

[13] F. Dragan, B. Butnaru, I. Manolescu, G. Gardarin, N. Preda, B. Nguyen, R. Pop, and L. Yeh, "P2PTester: a tool for measuring P2P platform performance," *ICDE*, 2007.

[14] Z. Zhou, H. Wang, J. Zhou, L. Tang, K. Li, W. Zheng, and M. Fang, "Pigeon: a framework for testing peer-to-peer massively multiplayer online games over heterogeneous network," *CCNC*, 2006.

[15] "PeerUnit," 2011, http://peerunit.gforge.inria.fr/.

[16] E. C. de Almeida, J. E. Marynowski, G. Sunye, Y. L. Traon, and P. Valduriez, "Efficient Distributed Test Architectures for Large-Scale Systems," *ICTSS*, 2010.

[17] J. E. Marynowski, M. Albonico, E. C. de Almeida, and G. Sunye, "Testing MapReduce-Based Systems," *SSBD*, 2011.

[18] G. Sunye, E. C. D. Almeida, Y. L. Traon, B. Baudry, and J.-M. Jzquel, "Model-based testing of global properties on large-scale distributed systems," *IST*, 2014.

[19] L. Baresi and M. Young, "Test oracles," *Technical Report CI S-TR-01-02, University of Oregon, Dept. of Computer and Information Science, Eugene, Oregon, USA*, 2001.

[20] E. T. Barr, M. Harman, O. McMinn, M. Shahbaz, and S. Yoo, "The oracle Problem in Software Testing : a Survey," *TSE*, 2015.

[21] The Apache Software Fondation, "Apache Spark," 2017, http://spark.apache.org/.