Computers and Electrical Engineering 50 (2016) 200–216

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Analysis and evaluation of MapReduce solutions on an HPC cluster[☆]

Jorge Veiga[*], Roberto R. Expósito, Guillermo L. Taboada, Juan Touriño

Computer Architecture Group, University of A Coruña, Campus de Elviña, s/n, A Coruña 15071, Spain

A R T I C L E   I N F O

Article history:
Received 28 November 2014
Revised 19 October 2015
Accepted 24 November 2015
Available online 28 December 2015

Keywords:
MapReduce
High Performance Computing (HPC)
Big Data
Energy efficiency
InfiniBand
Solid State Drive (SSD)

A B S T R A C T

The ever growing needs of Big Data applications are demanding challenging capabilities which cannot be handled easily by traditional systems, and thus more and more organizations are adopting High Performance Computing (HPC) to improve scalability and efficiency. Moreover, Big Data frameworks like Hadoop need to be adapted to leverage the available resources in HPC environments. This situation has caused the emergence of several HPC-oriented MapReduce frameworks, which benefit from different technologies traditionally oriented to supercomputing, such as high-performance interconnects or the message-passing interface. This work aims to establish a taxonomy of these frameworks together with a thorough evaluation, which has been carried out in terms of performance and energy efficiency metrics. Furthermore, the adaptability to emerging disks technologies, such as solid state drives, has been assessed. The results have shown that new frameworks like DataMPI can outperform Hadoop, although using IP over InfiniBand also provides significant benefits without code modifications.

1. Introduction

In the past several years, organizations have been adopting the Big Data paradigm to obtain valuable information from large volumes of data. The required capabilities to do so is increasing over and over, since the amount of data managed by organizations grows every year. This situation demands more efficient, scalable and capable systems, while maintaining some of the axioms that unify the way that data are managed and processed in most environments.

In order to overcome the limitations of current systems, some organizations have put an eye on High Performance Computing (HPC), which could provide with the technology to achieve Big Data goals. At the same time, HPC environments could benefit from Big Data programming models, such as MapReduce [1]. MapReduce is a programming model and an associated implementation for generating and processing large data sets, which is widely used to solve many analytic problems in several application fields, from biology to finances. The Apache Hadoop project [2] has gained significant attention in the last years as a popular open-source Java-based implementation of the MapReduce paradigm derived from the Google's proprietary implementation.

The use of Hadoop in HPC systems is expected to increase its performance, which is mainly limited by disk and network bandwidths. In this respect, network bandwidth can be improved by using high-performance interconnects (e.g. InfiniBand [3]), which are usually available in HPC environments. Hadoop relies on the ubiquitous TCP/IP protocol to implement its

[☆] Reviews processed and recommended for publication to the Editor-in-Chief by Guest Editor Dr. Y. Sang.
[*] Corresponding author. Tel.: +34 981 167 000 ext. 1212; fax: +34 981 167 160.
E-mail addresses: jorge.veiga@udc.es (J. Veiga), rreye@udc.es (R.R. Expósito), taboada@udc.es (G.L. Taboada), juan@udc.es (J. Touriño).

http://dx.doi.org/10.1016/j.compeleceng.2015.11.021
0045-7906/© 2015 Elsevier Ltd. All rights reserved.

communications support through Java sockets,and thus it is not able to leverage high-performance interconnects in an optimal way. This issue has caused the appearance of many HPC-oriented MapReduce frameworks like Mellanox UDA [4], RDMA-Hadoop [5], DataMPI [6] and others. These solutions attempt to overcome the limitations of standard Hadoop in HPC environments.

This paper presents an in-depth analysis and evaluation of representative HPC-oriented MapReduce solutions, indicating their similarities and differences. The proposed taxonomy is intended to be used to determine the suitability of each solution in specific use cases. The evaluation of these frameworks has been performed using several representative benchmarks. Furthermore, the trend to increase the number of cores in current systems has turned their power consumption into one of the most relevant issues. Therefore, the evaluation of the frameworks has been assessed not only in terms of performance but also taking into account their energy efficiency. The obtained results can be useful to identify the strengths and weaknesses of each solution in order to get valuable characteristics for future implementations.

The rest of this paper is organized as follows: Section 2 presents background information and related work. Section 3 describes the experiments performed with the different frameworks and analyzes the results in detail. Finally, Section 4 extracts the main conclusions of the paper and proposes ongoing work.

2. Background and related work

This section addresses the MapReduce model and its de-facto standard implementation, Hadoop. It also includes some details about InfiniBand, which has become the most widely adopted interconnect in the TOP500 list [7]. Next, it classifies the different HPC-oriented MapReduce solutions. Finally, related evaluation works are discussed.

2.1. MapReduce and Hadoop

The MapReduce paradigm performs the data processing in two main phases: Map and Reduce, which name the model. In the Map phase, the cluster nodes read the data from their own local filesystem and extract the significant features of each data element. Every feature is represented by a ⟨*key, value*⟩ pair in which every *value* of the significant feature is associated with a *key*. The *key* is used to index the *value* and add it to a group. Elements with the same key belong to the same group. The Reduce phase receives the ⟨*key, value*⟩ pairs after being grouped and sorted. Every *value* is operated with the rest of the elements of its group, leading to a final result. This result is also represented by a ⟨*key, value*⟩ pair, and it can be seen as the valuable information of the group identified by the *key*.

Currently, the most popular MapReduce implementation is Apache Hadoop [2], which is widely used by large corporations like Yahoo!, Facebook, Twitter and many others. Hadoop is an open-source Java-based framework which enables to store and process Big Data workloads. It mainly consists of two components: (1) the Hadoop Distributed File System (HDFS), which distributes the storage of data among the nodes of a cluster; and (2) the Hadoop MapReduce engine, which allocates data processing to the node where it resides.

Fig. 1 shows an overview of the overall MapReduce process performed by Hadoop. At the beginning of the process, the input data set is divided into splits and processed by the map tasks. Each time a map task finishes, the map output is partitioned into different fragments (*p-x-x* in the figure) which are sent to their corresponding reducers and merged to form the reduce input. Each reduce task processes its input to generate the final results and write them to HDFS.

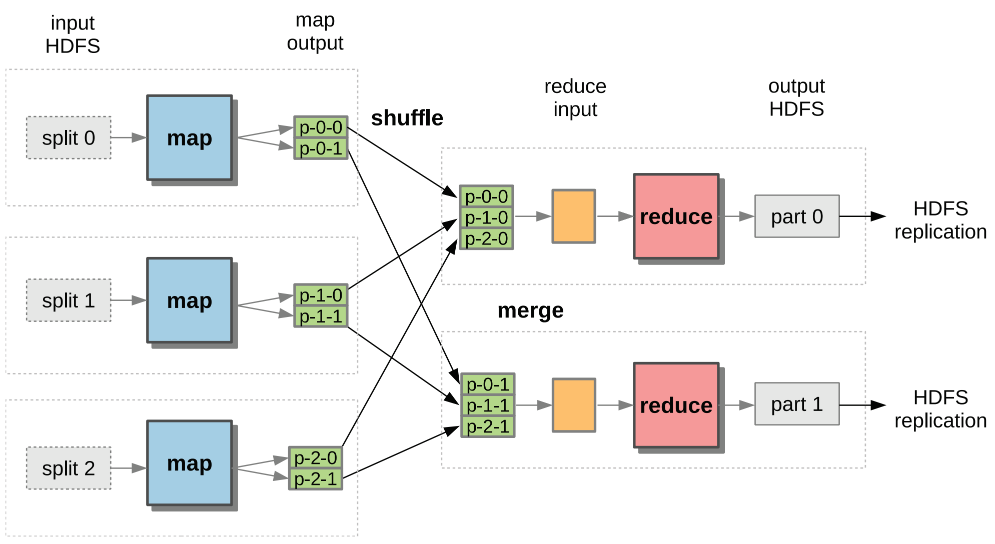

Fig. 1. Hadoop data flow with multiple map and reduce tasks.