Toward a New HDFS Functionalities Using a Probabilistic Distributed Algorithm

Ismail Hind and Ali Dahmani

***Abstract*—Distributed systems are computing systems where a number of components cooperate by communicating over a network. Few systems are still working in a stand-alone system where the user interface, persistent data and applications resided in one computer. However, the majority of nowadays systems are designed to work in distributed systems.**

Provide mechanisms for the storage and the manipulation of large amount of data is one of the largest technological challenges in software systems research today. Social media and web services produce an impressive amount of data daily. In this context, Hadoop Distributed file system (HDFS) is an open source software framework for distributed storage and processing of very large data (Big Data).

In this paper we introduce new functionalities for Hadoop Distributed File System using probabilistic distributed algorithms, our proposition is working in both homogenous and heterogeneous HDFS nodes which can reduce the communication cost.

***Index Terms*—Distributed systems, big data, Hadoop distributed file system, probabilistic distributed algorithms.**

I. Introduction

Distributed systems are a collection of independent nodes (computers) that appears to its users as a single coherent system [1]. It has the potential to be more reliable than stand-alone systems.

The main motivation behind studying those systems is that they increase the performance, the resource sharing, and the reliability of the network, and also allows users to share a common large data set.

The use of concurrent processes that communicate by message-passing holds the attentions of many researches since 1960s. The predecessor of the Internet, called ARPANET, was introduced in the late 1960s in which E-mail became its most successful application [2]. This technology is probably the earliest example of the large-scale distributed applications.

The study of distributed systems had become a branch of computer science in the late 1970s. The first conference in this field, called "Symposium on Principles of Distributed Computing (PODC)", was organized in 1982.

Moving the data while performing computations is the main issue in distributed systems, while dealing with a large scale of data, which requires a high bandwidth.

Hadoop Distributed File System is an open-source software framework for distributed storage and distributed processing of very large data sets (*Big Data*). It is designed to have a highly fault-tolerance, to be deployed on low-cost hardware providing a high throughput access to application data and to be suitable for application that have a large data sets.

The biggest two criterion imposed by HDFS provider are first, satisfying the minimum shares and second, the fairness, considering fairness disallow starvation of any user, dividing in fair manner the resources among the users. In bellow we present some of its assumptions and goals:

- Hardware Failure: Detection of faults and quick, automatic recovery from them is the biggest goal of HDFS.
- Streaming Data Access: Applications need streaming access to their data sets.
- Large Data Sets: A typical file in HDFS is gigabytes to terabytes in size.
- Simple Coherency Model: Based on write-once-read-many files, HDFS files application can run in a simple coherency model.

In this paper we will suggest and analyze new functionalities for Hadoop Distributed File System (HDFS) using the probabilistic distributed algorithms for uniform election introduced in [3].

The paper is organized as follows: Section II exposes some definitions and tools necessary for understanding the rest of the document. Section III is devoted to describe our propositions whereas Section IV presents a general analysis of the introduced methods. Finally, Section V summarizes the paper and shows our further work.

II. Tools and Assumptions

In this section we present the different tools used in this study. We start by describe HDFS architecture, then, we show the used probabilistic distributed algorithm for uniform election.

A. Hadoop Architecture

Hadoop is a well-known implementation of the MapReduce model platform. It consists of two main parts: First, MapReduce responsible of parallel computing and second, Hadoop Distributed File System which is the responsible for data management. Hadoop can directly work with different distributed file system, but the most common file system used by Hadoop is the Hadoop Distributed File System.

The HDFS systems were initially designed to optimize the performance with a highly fault-tolerance and to be deployed on low-cost hardware providing a high throughput access to application data. However, due to HDFS advantages, the number of its applications is increasing which leads to grow

Manuscript received January 2, 2015; revised March 6, 2015.

The authors are with FS– Abdelmalek Essa âdi University, M'Hannech II 93030 Tetuan, Morocco (e-mail: ismailhind@gmail.com).

DOI: 10.7763/IJFCC.2015.V4.385

the demand for its platform. One of the most important aspects which should be considered is homogeneity of the system.

Hadoop uses a *master/slave* architecture for both distributed computation and distributed storage. The HDFS cluster is formed by a single NameNode (master server which greatly simplifies the system architecture) manages the file system *namespace* and controls access to files by clients. In addition, there are a number of DataNodes (slaves) which manage storage attached to the nodes that they run on. HDFS makes visible a file system namespace allowing user data to be stored in files. HDFS file is split into one or more blocks which are stored in a set of DataNodes.

The NameNode can opens, closes and renames the namespace files and directories. It also directs the mapping of blocks to DataNodes.

The DataNodes are responsible for distributing read and write requests from the file system's clients. It can also create and deletes blocks, and replication upon the NameNode instructions.

All applications that deal with large data sets are compatible with HDFS. HDFS holds up the write-once-read-many semantics on files. Internally, typical block size is 64 MB which splits the HDFS file into 64 MB chunks.

When the client wants to create a file, his request does not reach immediately the NameNode. In fact, the HDFS client hides initially the file data into a temporary local file. The NameNode allocates a data block for the client after inserting the file name into the file system hierarchy. So, it responds to his request by identifying the DataNode and the destination data block. Then the client takes the data block from the local temporary file to chosen DataNodes. After closing a file, the remaining un-flushed data in the temporary local file is transferred to the DataNode. Then, the client informs the NameNode that the file is closed. However, if the NameNode dies before the file is closed, the file is lost.

The following figure shows HDFS architecture.

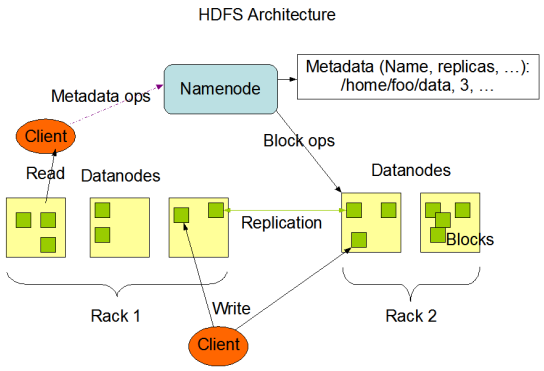

Fig. 1. HDFS architecture.

As it shown in the Fig. 1, the main function of the NameNode is to maintain and to execute the file system namespace. Indeed, all the modifications in the file system namespace or in its properties are tracked by the NameNode. In addition, it has other various functionalities we cite bellow the principle ones:

- NameNode remains a record of the way the HDFS files are divided into blocks where the nodes of these blocks are stored.
- It is responsible for mapping a file name to a set of blocks and then mapping the block to the DataNodes where it is located.
- NameNode also records the metadata of all the cluster files, e.g. the location, permissions, hierarchy, the size of the files, etc.
- To make sure that the DataNodes are working properly, NameNode regularly receives a Heartbeat and a Block report from all the DataNodes in the cluster. This block report holds a list of all blocks on a DataNode.
- In case of a DataNode failure, new DataNodes for new replicas are chosen by the NameNode which also treats the communication traffic to the DataNodes.

In the other side, the DataNodes also had various functionalities which could be summarized in the ones below:

- DataNodes effectuate the low-level read and write requests from the file system's clients.
- Based on the decisions taken by the NameNode, DataNodes are responsible for deleting and creating blocks and also replicating.
- They send regularly a report about the blocks present in the cluster to the NameNode.
- They redirect data to other specified DataNodes.
- DataNodes send heartbeats to the NameNode reporting the overall health of HDFS.
- The DataNode stores in separate files each block of HDFS data in its local file system.

B. *Uniform Election*

When you submit your final version, after your paper has been accepted, prepare it in two-column format, including figures and tables.

The election problem is a well-known combinatorial optimization problem which holds the attention of many researchers since it was first suggested by LE LANN [4]. The election is to choose one and only one element from a network. However, the uniform election is special case of election where all the elements of an undirected and connected network must had the same chance to be elected. Therefore, this problem has been studied under various hypotheses: oriented or not oriented network, synchronous or asynchronous system, anonymous or with identifier elements, etc.

Many researches known in the literature has treated this type of election, such as, uniform election in trees [5], polyominoids [6] and triangular grid graphs [3]. The last Local computation algorithm is used in this paper. This algorithm is based on local computations [7] and graph relabeling systems [8].

III. Proposed Hadoop Functionalities

HDFS architecture is designed to have a highly fault-tolerance, to be deployed on low-cost hardware providing a high throughput access to application data and to be suitable for application that have a large data sets. However some of it functionalities are still in development. Indeed, in this section, we will introduce our proposed HDFS functionalities.

The main motivation for our study is as follows:

- To reduce the communication cost in the entire HDFS;
- To reduce the search overhead for resources and matching jobs.

To reduce the communication cost between the elements of HDFS we proposed a new topology for clustering DataNodes taking under consideration the homogeneity of clusters. So, as it shown in Fig. 2, the master organizes virtually the DataNodes under a specific topology called Triangular grid graph which simplify applying the uniform election algorithm introduced in [3].

The following figure represents HDFS cluster architecture.

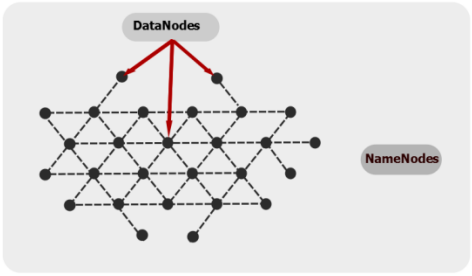

Fig. 2. HDFS cluster architecture.

As it shown in the Fig. 1, the cluster comprises of a single NameNode (Master) and a number of DataNodes (Slave nodes). The set of DataNodes is represented by triangular grid graph where each vertex represents a DataNode and the link of communications is represented by edges. The master can communicate with *special* DataNodes in a cluster. Those vertices are the ones with degree $\in \{5, 6\}$ which can be used later to transfer messages and jobs.

In fact, the main aim of choosing the maximum degree vertices is that they are in the middle of the graph which on one hand, decrease the mean of communication costs, on the other hand, simplify the communications between vertices leading to improve the system performance.

So in the next we introduce our two methods.

A. Based on Homogeneous DataNodes

In this method, we supposed the homogeneity of the DataNodes. We represent the DataNodes by a graph *G* which had the topology of triangular grid graph as it shown in Fig. 2.

We distinguish between 3 levels of vertices:

- Vertices of level 1 will be localized at the leaf of G i.e. the vertices of degree < 2 ;
- Vertices of level 2 whom degree in {3, 4};
- And finally vertices of degree >4 which are considered as level 3 vertices.

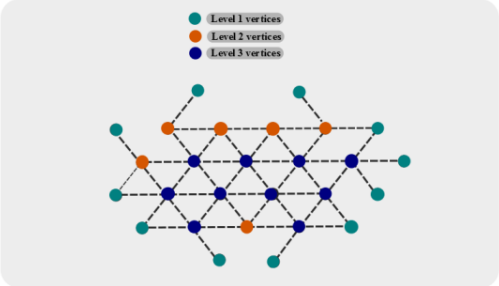

Fig. 3. DataNodes levels inside a cluster.

Fig. 3 represents the different levels of vertices.

The master communicates with the level 3 vertices, which simplified the message transmissions and the communication between nodes and master. Therefore, the master uses the probabilistic distributed algorithm for uniform election to decide which DetaNode will take the charge.

The main goal for this method is to decrease the charge on the middle vertices by rearranging the all the graph.

B. Based on Heterogeneous DataNodes

In this method, we keep the triangular grip graph topology but we supposed that the graph *G* could be heterogeneous. Every period *k*, where *k* defined as the necessary time to refresh the graph *G*, i.e., each time the master received a failure in a DataNode, or else in one hour, taking under consideration that it received the heartbeat from the entire nodes every three seconds. In addition, the central vertices must be a chosen from the high performance nodes. We distinguish here between three levels:

- Low quality nodes considered as vertices of level 1;
- Medium quality nodes considered as vertices of level 2;
- And finally high quality nodes considered as level 4 vertices.

So, the vertices are localized at *G* according to their quality. In [9] the authors proved that use of heterogeneous DataNodes reduced the communication cost. So, and for our topology, the middle DataNodes take a higher charge than others which explains the use of high quality nodes in the middle of clusters.

IV. Analysis

In this section, we analyze the proposed HDFS functionaries. Indeed, for the homogeneous datanodes, the central vertices are rearranged in duration *k*, where *k* represents the necessary time for master to receive the heartbeat of the entire cluster, or else, to discover the failure of one those vertices.

This rearrangement is necessary to avoid the charge on the middle vertices caused by communications. So, according to the vertices level, the master redesigns the graph adding a level to each vertex each *k*-duration:

- Vertices of level one will be located at the level two vertices;
- Vertices of level two will be located at the middle of the graph (changed to level three);
- Vertices of level three will be located at leafs of graph.

V. Conclusion

In this paper, we have introduced and analyzed two functionalities to HDFS using a probabilistic distributed algorithm for uniform election. According to our propositions, and in each cluster, the HDFS DataNodes will be represented by a triangular grid graph where each vertex *v* represents a single DataNode and the edges represent the links between DataNodes. Indeed, the proposed methods support both, homogeneous and heterogeneous DataNodes.

As perspective, we aim to experiment our work in a real application. Also, we attend to prove that mean of HDFS communication cost is lower when using our suggestions.

REFERENCES

[1] G. R. Andrews, *Foundations of Parallel and Distributed Programming*, 1st ed., Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[2] I. Peter, *History of the Internet*, Autopilot Productions [Brisbane], 2004.

[3] M. Stouti, I. Hind, and A. E. Hibaoui, "Probabilistic distributed algorithm for uniform election in triangular grid graphs," *International Journal of Advanced Computer Science and Applications*, vol. 4, 2013.

[4] G. L. Lann, "Distributed systems - Toward a formal approach," in *Proc. of the IFIP Congress 77*, 1977, pp. 155-160.

[5] Y. Métivier, N. Saheb, and A. Zemmari, "A uniform randomized election in trees (extended abstract)," in *Proc the 10th International Colloquium on Structural Information and Communication Complexity (SIROCCO 10)*, Carleton University Press, 2003, pp. 259-274.

[6] A. E. Hibaoui, J. M. Robson, N. Saheb-Djahromi, and A. Zemmari, "Uniform election in trees and polyominoids," *Discrete Appl. Math.*, vol. 158, no. 9, pp. 981-987, May 2010.

[7] A. Sellami, "Des calculs lacaux aux algorithmes distribués," Ph.D.dissertation, Université Bordeaux I, 2004.

[8] I. Litovsky, Y. Métivier, and E. Sopena, "Graph relabelling systems and distributed algorithms," in *Handbook of Graph Grammars and Computing by Graph Transformation*, H. Ehrig, H. Kreowski, U. Montanari, and G. Rozenberg, Eds., World Scientific, 1999, vol. 3, pp. 1-56.

[9] A. Rasooli and D. G. Down, "A hybrid scheduling approach for scalable heterogeneous hadoop systems," *IEEE Computer Society*, 2012, pp. 1284-1291.

Ismail Hind was born in Fnideq north, Morocco in 1987. He is a PhD student at Abdelmalek Essaadi University, working on probabilistic distributed algorithms from 2010 to 2015. He got the master in engineering and computer science at Tetouan, Morocco, 2010, and the bachelor degree in mathematical sciences at Tetouan, Morocco, 2005.