BIG DATA STORAGE SYSTEM BASED ON A DISTRIBUTED HASH TABLES SYSTEM

Telesphore Tiendrebeogo and Mamadou Diarra

Department of Mathematics and Computer Science, Nazi Boni University, Bobo-Dioulasso, Burkina Faso

ABSTRACT

The Big Data is unavoidable considering the place of the digital is the predominant form of communication in the daily life of the consumer. The control of its stakes and the quality its data must be a priority in order not to distort the strategies arising from their treatment in the aim to derive profit. In order to achieve this, a lot of research work has been carried out companies and several platforms created. MapReduce, is one of the enabling technologies, has proven to be applicable to a wide range of fields. However, despite its importance recent work has shown its limitations. And to remedy this, the Distributed Hash Tables (DHT) has been used. Thus, this document not only analyses the and MapReduce implementations and Top-Level Domain (TLD)s in general, but it also provides a description of a model of DHT as well as some guidelines for the planification of the future research.

KEYWORDS

Big Data, MapReduce, Distributed Hash Table, Scaling.

1. INTRODUCTION

Big Data is fascinating because of the potentialities it suggests in terms of organizational performance and strategic decision-making [2], [4]. With this excessive growth of its massive data, their analysis has become a tedious challenges and their safety a priority. That's why Google MapReduce has quickly become a highly reputable reference. It is a simple and scalable fault-tolerant data processing environment that allows its users to process massive distributed and large-scale data to extract new knowledge. Recent studies have shown the limitations of MapReduce [5] and have proposed DHTs [6], [7] as solutions for incidents that could impact the digital world. As part of this work, we present our architecture inspired by [6] to which we will add some properties of digital imaging [8] for better data management and we will end with an intuitive comparison of our model with existing models. Our future work will focus on a hybrid DHT - MapReduce architecture that will exploit the limitations of MapReduce and the performance of our DHT and an analysis of the simulation results of our model and existing models.

2. OUR WORK OBJECTIVE

The undeniable interest in Big Data has led to the birth of new massive data processing platforms. The objective of our work is to propose an intuitive "turnkey" model to improve the performance of these platforms secure data storage and reduce processing costs.

The basic idea is to provide multiple backup master nodes, which can substitute for any node to compensate for the failure of the single master node of a MapReduce process.

DOI : 10.5121/ijdms.2020.12501

Our prototype is an adaptive platform exploiting the advantages of DHT and MapReduce. It uses an EHT to manage node intermittencies, MapReduce master node overloads, and work resumption in a decentralized but efficient manner to provide a more robust middleware that can be effectively exploited in dynamics distributed environments.

Our work is based on the following three steps:

1) Related works;

2) Model description;

3) Comparative analysis of our model.

3. RELATIVE WORKS

3.1. Big Data

The term "Big Data" was popularized by John Mashey, a computer scientist at Silicon Graphics in the 1990s. He refers to databases that are too large and complex to be studied with traditional statistical methods and, by extension, to all the new tools for analysing this data. In 2001, Douglas Laney analysed this new trend through a very simple list of three "3V", then expanded to five "5V".

- Volume: the large amount of information contained in these databases.
- Velocity: the speed of their creation, collection, transmission and analysis.
- Variety: the differences in nature, format and structure.
- Value: the ability of these data to generate profit.
- Truthfulness: their validity, i.e. quality and accuracy as well as their reliability.

In short, Big Data represents the art of collecting, storing and processing large masses of data to offer new perspectives [4], [21].

3.2. Mapreduce Programming Model

MapReduce is a massively parallel programming model suitable for processing very large quantities of data. It was popularized by Jeffrey Dean and Sanjay Ghemawat [2]. This programming model revolves around of two main steps Map and Reduce (see Fig. 1). Its principle of operation is to decompose a task into smaller tasks. The decomposition process consists of dividing the initial data volume N into smaller volumes ni, which will be handled separately. MapReduce relies on the manipulation of couples (key, value).

The Reduce() function combines all these results into one pair (key, value)) unique [2] [12].

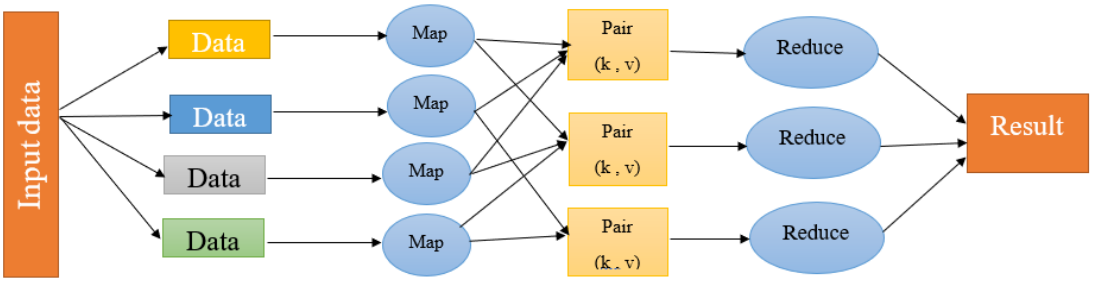

Figure1. MapReduce paradigm diagram

3.3. Hadoop MapReduce

Hadoop is an open source Apache project based on the MapReduce paradigm [2]. It is a fault-tolerant platform that aims to support the logistics of task distribution. Since very large volumes of data are involved, MapReduce is typically used in combination with a distributed file management system, in the case of Hadoop, it is the Hadoop Data File System (HDFS). HDFS has a master/slave architecture. In this logic, a Hadoop cluster consists of a single master server, named NameNode, which manages the file system and access rights; but also servers that are both a calculation tool and a storage tool, named DataNodes, usually one per node.

3.4. Distributed Hash Table (DHT)

DHTs possess several properties that are essential to the operation of peer-to-peer systems in which they are used at the application level. Indeed, they offer a coherent hash function and efficient algorithms the location of the node responsible for a given pair (key, value). These are distributed storage systems that use an infrastructure based on key routing protocols. We have selected four of the major DHTs offering different functionalities such as efficiency and simplicity with Chord, controlled data placement with SkipNet, Pastry routing and localization and other features with Kademlia.

The maintenance of the routing tables used to store information on the evolution of a network is very important unrealistic in a distributed environment, the substitution of DHTs by the creation of virtual networks over the initial networks, allow to reduce the size of the table in each node while considerably increasing the efficiency search algorithm [6]. The construction technique of the overlay remains the same and takes place in three steps regardless of the DHT topology [11]:

- The definition of a so-called "ideal" topology if all nodes are mutually reachable and any query leads to a satisfactory result in an efficient way.
- The description of the arrival and departure operations of the nodes in the network.
- The definition of a maintenance protocol (self-organization) that solves the problem related to the removal of a node and the increased fault tolerance periodically repairs disturbances on the network topology.

3.5. DHTs First Generation

- Chord [13] Organizes its address space whose 2m possible addresses (identifier, id) are ordered on the following date along its circumference. Pair and resource have an identifier (hash function SHA-1 m = 160bits), guaranteeing a homogeneous distribution of resources.

- Pastry [14] Minimizes the message path in terms of number of IP hops. A Pastry node is associated with a 128-bit nodeID key, randomly generated with a hash function, and the nodes thus form a space naming circular of [0, 2 128[.

- Kademlia (kad) [15] Ensures that a node has at least one contact in each subtree, with this contact being the one with the most contacts in each subtree guarantee, it can find any other node whose identifier is different from its own.

- SkipNet [16] Controls the placement of data on the network and the maintenance of routing within an administrative area.

3.6. DHT / MapReduce

The traditional MapReduce platform is centralized, with parallel processing performance managed on one master node supervising the progress of all compute nodes are often limited by bottlenecks when the number of compute nodes increases [17]. To remedy this, several solutions were proposed:

- The P2P-MapReduce model by Marozzo et al [18] performs job state replication, manages the main failures and allows intermittent node participation in a decentralized way but efficace. Using a P2P approach, it extends MapReduce to make it suitable for large-scale, highly dynamic environments where failures need to be managed to avoid a critical waste of resources and computing time.

- ChordMR [19] with an architecture with three basic roles: User, Master and Slave. User nodes are responsible for job submission. Master nodes organized in Chord are responsible for job assignment and execution. And the slave nodes in charge of MapReduce tasks are still kept in the traditional structure like Hadoop.

- ChordReduce [20] takes its name from the two components on which it is built. Chord provides the backbone of the network and file system, offering scalable, distributed storage and fault-tolerant routing. MapReduce runs on top of the Chord network and uses the underlying distributed hash table functionality. ChordReduce is capable of running on any arbitrary distributed configuration. It ensures that no single node is a point of failure and that no single node has to coordinate the efforts of other nodes during processing. Its design is to implement additions to Chord's existing functionality, treating each target task or calculation as data that can be distributed in the same way as routed files.

4. DESCRIPTION OF OUR DHT ARCHITECTURE MAPREDUCE

4.1. Recalls some properties of hyperbolic geometry: Hyperbolic plane and Poincaré disc

The model we use to represent this hyperbolic plane is the Poincaré disk model. In this model, we refer to the points of the plane using complex coordinates. An important property of the hyperbolic plane is that we can pave it with polygons of any size, called p-gons. Each paving is represented by a notation of the form p, q where each polygon has p faces and where q polygons touch each other at each vertex.

This form of notation is called a schläfli symbol. There is a hyperbolic p, q paving for each couple (p, q) obeying the inequality:

$$(p - 2) * (q - 2) > 4.$$

In a tiling, p is the number of faces of the polygons of the primal and q is the number of faces of the polygons of the dual. We make p tend towards infinity, thus transforming the primal into an infinite regular tree of degree q. This particular paving cuts the hyperbolic plane into distinct spaces and builds an address tree having vertices with unique coordinates, as shown by Coxeter et al. in [22][23].

As it is a regular address tree of degree q, the root node can give a unique address to each of its q neighbors and any node other than the root can give a unique address to q-1 neighbors.

In the Poincaré disk model, the distance between any two points z and w is given by a curve that minimizes the length between these two points and is called a geodesic of the hyperbolic plane.

Each node of the network will be assigned a virtual address which will be defined by the point coordinates of the hyperbolic plane noted H2 of curvature radius equal to -1.

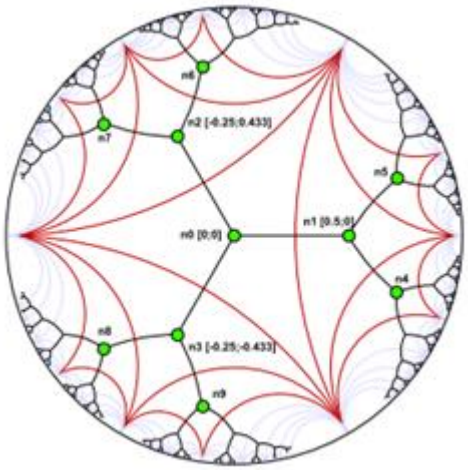

Figure2. 3-regular tree in the hyperbolic plane.

4.2. Architecture presentation

Our prototype is built from the DHT [6] starting point of our research and MapReduce. In our approach, we propose a distributed platform based on a DHT model built on an overlaying network without imposing a particular topology just like the major DHTs proposed in 3.5, but using several master nodes defined by a virtual address, as described below.

Our new architecture proposes master nodes resulting from a two-level virtual addressing (hyperbolic and coloring). It gives us more scalability in a distributed environment and allows us to cope with failures related to the uniqueness of the master node and the intermittencies of the other nodes of the traditional MapReduce platform.

The routing proposed in this overlaying network is a gluttonous routing which is carried out by using hyperbolic virtual distances. Routing operations are carried out on the fly, taking into account only the virtual distance separating each neighbouring node from the destination node. Typically, when the η node tries to reach the μ node, it calculates the distance between its correspondent μ and each of its own neighbours, and it selects, in fine, the neighbours ν having

the shortest hyperbolic distance with the destination node µ. Mathematically, the distance between any two points u and v, taken in the hyperbolic space H, is determined by equation 4.2.1 [7]:

$$d_H(u,v) = \operatorname{arccosh}(1+2\lambda) \quad (4.2.1)$$

Ou

$$\lambda = \frac{|v-u|^2}{(1-|u|^2)*(1-|v|^2)} \quad (4.2.2)$$

4.3. Our Big Data Storage Model using a DHT Structure

Step 1: Each Big Data object is associated to a generation a 512-bit key (Object IDentifier OID) following the principle of (Global Unique Identifier GUI). Let O_i an object we have OID(O_i) = Key(512 bits).

Step 2: The 512-bit key of the object to be stored is sequentially divided into two parts of 256 bits each (called subkey(256)).

Step 3: Each sub-key is mapped as follows :

- The first 24 bits are used to determine the RGB color code of a voxel of coordinates (x_i, Y_i , Z_i) in the hyperboloid.
- Respectively the following 64 bits in decimal determine the value of the Xi coordinates; then, it is the same for the two following series of 64 bits, which allow calculating Y_i, Z_i.
- Note P_1 the voxel colored by the code of the first 24 bits of the first series of 256 bits.
- Note P_2 the colored voxel constructed from the second series of 256 bits of the key Oi.
- Thus the pair (P_1, P_2) is unique for each object to be stored in our structure.

Step 4: For each sub key after determining the RGB color code with the first 24 bits then the coordinates (X_i, Y i, Zi) on (64+64+64) bits = 192 bits, there are 40 bits left. The 40 bits will be split in two and will be used to calculate the coordinates of points Vi1 (x_{i1}, y_{i1}) and V_{i2} (x_{i2}, y_{i2}) which represents the location of the data on the open Poincaré disk with radius 1.

Step 5: As x_{i1}, x_{i2}, y_{i1}, y_{i2} must be inside the Poincaré disk, then $|x_{i1}| < 1$, $|x_{i2}| < 1$, $|y_{i1}| < 1$, |yi2 | < 1. To do so, we perform the following transformation:

Let x_{i1} = 01000110001100011100 = 20 bits.

$$\text{We set } |x_{i1}| = \frac{01000110001100}{11111111111111} < 1$$

Step 6: In the distributed storage mechanism, the object Oi is stored on the node ni closest to Vi1 and then replicated in the node nj closest to Vi2 according to a glutton routing process.

Step 7: In the Poincaré disk model, the hyperbolic tree is scalable [7]. Figure 3 illustrates the object storage process in a DHT-based system.

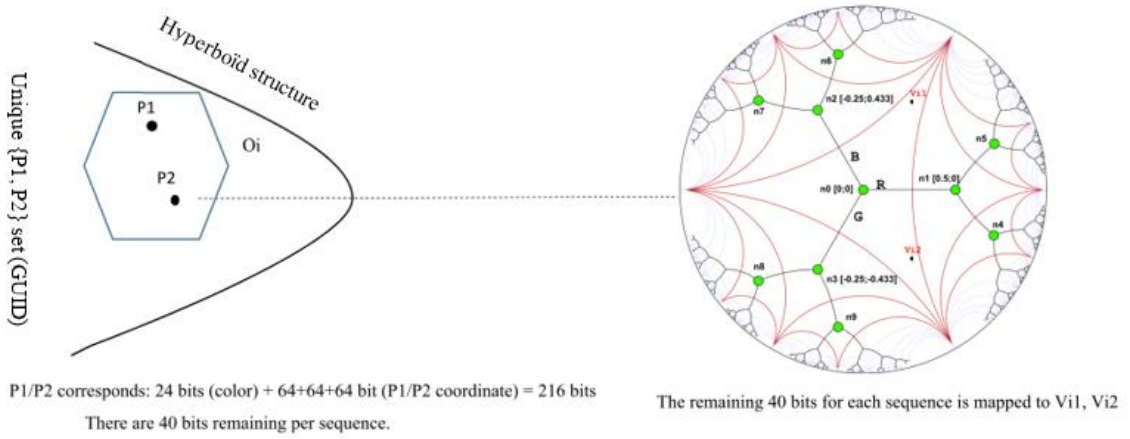

Figure3. big data storage in a DHT MapReduce structure

4.4. Our DHT Mapreduce Model Formal Evaluation

Existing solutions often require a predefined topology (Ring, Ring/Plaxton, Hypercube, Chained List, etc.) and routing tables, often resulting in high bandwidth consumption and high latency [6], [17], [20].

Our architecture does not carry any constraint linked to any topology or routing table. It uses a greedy routing algorithm based on virtual coordinates coming from the hyperbolic plane whose performances have been proven [6].

Thus, it will allow us to solve the latency problem in MapReduce processing by giving a large number of nodes (for example of the order of 108), resulting from a virtual RGB color addressing of a voxel of coordinates (Xi , Yi , Zi) in the hyperboloid, which will relay to the master node in the progressive monitoring of all the compute nodes, in order to avoid bottlenecks when the number of compute nodes increases. Thus, we will be able to overcome the hardware and software problems of the above-mentioned or proprietary solutions and provide a two-tier model of data backup and parallel processing security.

5. CONCLUSION

Our DHT is a model without topological constraints unlike most existing models. It is both robust and efficient, allowing self-organization and optimal management of nodes in its theoretical design, qualities derived from the initial model. Also, the addition of the foundations of digital imaging, allowed us to guarantee a better distributed storage and secularization of data access in a theoretical way that should be simulated.

The main contribution of our work is to provide an autonomous load balancing mechanism by associating to the initial DHT a node replication capability. Thus, several master nodes will be responsible for the execution of MapReduce tasks, backup and recovery in case of failure of other master nodes.

Next, we will implement a fully functional version of our model and perform detailed experiments to test its performance. This will precede the implementation of our hybrid DHT - MapReduce model, which will exploit the limitations of existing models and the advantages of MapReduce.

REFERENCES

[1] Quentin Baert, Anne-Cécile Caron, Maxime Morge, Jean-Christophe Routier. Strat´egie de découpe de tâche pour le traitement de données massives. Journées Francophones sur les Systèmes Multi-Agents, Jul 2017, Caen, France. pp.65-75.

[2] Jeffrey Dean and Sanjay Ghemawat. Mapreduce simplified data processing on large clusters. Communications of the ACM, 51(1) :107–113,2008.

[3] Di Wu, Ye Tian, and Kam-Wing Ng. Analytical study on improving dht lookup performance under churn. In Sixth IEEE International Conference on Peer-to-Peer Computing (P2P'06), pages 249–258. IEEE, 2006.

[4] Josiane Mothe, Yoann Pitarch, and ´Eric Gaussier. Big data : le cas des systèmes d'information. Ingénierie des Systèmes d'Information, 19(3):9–48, 2014.

[5] Koya Mitsuzuka, Ami Hayashi, Michihiro Koibuchi, Hideharu Amano, and Hiroki Matsutani. In-switch approximate processing : Delayed tasks management for mapreduce applications. In 2017 27th International Conference on Field Programmable Logic and Applications (FPL), pages 1–4. IEEE, 2017.

[6] Telesphore Tiendrebeogo, Daouda Ahmat, and Damien Magoni. Evaluation de la fiabilité d'une table de hachage distribuée construite dans un plan hyperbolique. Technique et Science Informatique, TSI, Tech. Sci. Informatiques 33(4): 311-341 (2014)

[7] Telesphore Tiendrebeogo and Damien Magoni. Virtual and consistent hyperbolic tree : A new structure for distributed database management. In International Conference on Networked Systems, pages 411–425. Springer, 2015.

[8] Robert Kleinberg.Geographic routing using hyperbolic space. In IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications, pages 1902–1909. IEEE, 2007.

[9] Changjun Li, M Ronnier Luo, Robert WG Hunt, Nathan Moroney, Mark D Fairchild, and Todd Newman. The performance of ciecam02. In Color and Imaging Conference, volume 2002, pages 28–32. Society for Imaging Science and Technology, 2002.

[10] Ripon Patgiri. Taxonomy of big data : A survey. arXiv preprint arXiv:1808.08474, 2018.

[11] R. Ruslan, A. S. M. Zailani, N. H. M. Zukri, N. K. Kamarudin, S. J. Elias, and R. B. Ahmad, "Routing performance of structured overlay in distributed hash tables (dht) for p2p," Bulletin of Electrical Engineering and Informatics, vol. 8, no. 2, pp. 389–395, 2019.

[12] Quentin Baert, Anne-Cécile Caron, Maxime Morge, and Jean-Christophe Routier. Stratégie de découpe de tâche pour le traitement de données massives. In Journes Francophones sur les Systèmes Multi-Agents, pages 65–75. C´epaud`es ´edition, 2017.

[13] Ion Stoica, Robert Morris, David Liben-Nowell, David R Karger, M Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord : a scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Transactions on Networking (TON), 11(1) :17–32, 2003.

[14] Antony Rowstron and Peter Druschel. Pastry : Scalable, decentralized object location, and routing for large-scale peer-to-peer systems.In IFIP/ACMInternationalConferenceonDistributed Systems Platforms and Open Distributed Processing, pages 329–350. Springer, 2001.

[15] Petar Maymounkov and David Mazieres. Kademlia : A peer-to-peer information system based on the xor metric. In International Workshop on Peer-to-Peer Systems, pages 53–65. Springer, 2002.

[16] Nicholas JA Harvey, John Dunagan, Mike Jones, Stefan Saroiu, Marvin Theimer, and Alec Wolman. Skipnet : A scalable overlay network with practical locality properties. 2002.

[17] Koya Mitsuzuka, Ami Hayashi, Michihiro Koibuchi, Hideharu Amano, and Hiroki Matsutani.In-switch approximate processing: Delayed tasks management for mapreduce applications. In 2017 27th International Conference on Field Programmable Logic and Applications (FPL), pages 1–4. IEEE, 2017.

[18] Fabrizio Marozzo, Domenico Talia, and Paolo Trunfio. P2p-mapreduce: Parallel data processing in dynamic cloud environments. Journal of Computer and System Sciences, 78(5) :1382–1402, 2012.

[19] Jiagao Wu, Hang Yuan, Ying He, and Zhiqiang Zou. Chordmr : A p2p-based job management scheme in cloud. Journal of Networks, 9(3) :541, 2014.
[20] Andrew Rosen, Brendan Benshoof, Robert W Harrison, and Anu G Bourgeois. Mapreduce on a chord distributed hash table. In 2nd International IBM Cloud Academy Conference, volume 1, page 1. 2016
[21] Thomas Bourany. Les 5v du big data. Regards croises sur l'economie, (2) :27–31, 2018.
[22] Harold Scott Macdonald Coxeter and GJ Whitrow. World-structure and non-euclidean honeycombs. Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences, 201(1066) :417–437, 1950.
[23] Harold Stephen Macdonald Coxeter. Regular honeycombs in hyperbolic space. In Proceedings of the International Congress of Mathematicians, volume 3, pages 155–169. Citeseer, 1954.
[24] Changjun Li, M Ronnier Luo, Robert WG Hunt, Nathan Moroney, Mark D Fairchild, and Todd Newman. The performance of ciecam02. In Color and Imaging Conference, volume 2002, pages 28–32. Society for Imaging Science and Technology, 2002.

AUTHORS

Telesphore Tiendrebeogo

He has master's degree in computer networks 2007, master's degree in real-time systems in 2008 and PhD in distributed system en computer network in 2013, I am associate professor and head of the research team in computer networks and distributed systems since 2017.

Mamadou Diarra

He has master's degree in database and software engineering, he is currently a PhD student in computer science on the big data topic.