Performance investigation of selected NoSQL databases for massive remote sensing image data storage

Yosra Hajjaji, Imed Riadh Farah

Performance investigation of selected NoSQL databases for massive remote sensing image data storage

Yosra Hajjaji
Riadi Laboratory
National School of Computer Science
Manouba University, Tunisia
hajjajiyosra05@gmail.com

Imed Riadh farah
Telecom-Bretagne
National School of computer science
Manouba University, Tunisia
riadh.farah@ensi.rnu.tn

***Abstract*—today's sensors are like eyes in the sky, thanks to the growth of satellite remote sensing technologies. Therefore, we see a steady evolution of the usage of different types of sensor, from airborne and satellites platforms which are generating large quantities of remote sensing image for divers applications such as; smart city, disaster management, military intelligence and others. As a result, the rate of growth in the amount of data by satellite is increasing dramatically. The velocity has exceeded 1TB per day and it will certainly increase in the future. However, it becomes crucial for these huge volume data to be stored. So, how to store and manage it efficiently becomes a real challenge because traditional ways have intensive issues; they are expensive and difficult to extend. Therefore, we need some scalable and parallel models for remote sensing data storage and processing. In this paper, we describe a scalable and distributed architecture for massive remote sensing data storage based on three No SQL databases (Apache Cassandra, Apache HBase, MongoBD). Also, a Hadoop-based framework is proposed to manage the big remote sensing data in a distributed and parallel manner.**

***Index Terms*—Big data,Remote sensing data, NoSQL databases, Cassandra,HBase, MongoDB, cluster, Hadoop/ MapReduce.**

I. Introduction

Today, with the high tech world the amount of data and especially Remote sensing data are growing from day to day. Actually, very large volume of remote sensing data are now freely available from the NASA Open Government Initiative. The Earth Science Data and information System (ESDIS), is one of NASA archives, holds about 7.5 PB of data with nearly 7000 unique data sets and 1.5 million users in 2013 [1]. Based on this it can be truly stated that we are already living in the age of Remote sensing Big Data. Generally, data collected from sensors are a sort of measurements and reports of some properties of the environment, such as the pressure, humidity, temperature and radiation [2]. Generally, databases are one of the best ways to store these measurements and lately processed to find particular situations. However, traditional approaches of remote sensing data storage based on traditional relational databases could not longer deal with the Big Data challenges, they either do not support such volumes or face performances issues, especially where the volume (scale of Data) and velocity (different forms of Data) of the data raise at a remarkable rate. In the last few years, several researchers and engineers demand new ways to manage image data rapidly, in other word big volumes of remote sensing image data need to be stored and processed in real time or nearly-real time [3]. As a result, in order to deal with these problems, it becomes necessary to adapt new technologies, such as the technologies of Big Data in remote sensing Big Data storage and processing approaches.

In order to address the challenges proposed by storing and processing Big Remote Sensing data, this paper outlines a method to store massive image data based on No SQL databases (Cassandra [7], HBase [8] and mongoDB [9]). We aim to choose the most suitable Database to store our data. These databases are distributed, so the data stored in each one of them are actually stored in several different nodes. In addition, we also use the Hadoop framework [10] to efficiently manage and process Remote sensing Big Data.

Outline: The rest of the paper is organized as follows. Section II provides a brief survey of some of the previous works done in this field. Section III introduces the key technologies used in the storage strategy. In section IV, the method of image data storage based on No SQL database and big data frameworks is discussed. Data source, cluster configuration and experimental results obtained are presented in section V. Finally, in Section VI, we conclude this paper.

II. Related works

A variety of researches has been recently based on big data technologies such as the new movement of No SQL databases and big data frameworks such as Apache Hadoop into remote sensing missions. They aimed to solve the storage and processing challenges of image data. Therefore, several

domain experts have conducted the research from many viewpoints.

Zhifeng Xiao, Yimin Liu [11], provides a view on the capabilities on the new movement of NoSQL DBs tested in global image systems. Therefore, they build an application which they called HBGIS based on HBase as the tile source. As result, they have come out with that NoSQL database can involve the huge volume of data, replication and network partitioning, as well as the good performance in the simulation of large volume data read and write requests. Jan Sipke and all in [12] presented a comparative study of three databases on their relative performance with the regards to sensor data storage. Results show that Cassandra read performance is heavily affected by virtualization, both positively and negatively while MongoDB performance impact is moderate, with PostgreSQL Small writes are slow, but are positively affected by virtualization. Yahoo Liu and all [13] presented a method to store big remote sensing data based on HBase and process it with Hadoop/MapReduce. They used a multi-resolution pyramid model to divide the image into multiples layers and blocks, then they designed two tables to store respectively the meta-data and spatial blocks data of the images. Experiments results of the former model of remote sensing, storage and process show that, every time they increase both data volume and data node, the time of data import and process decreases and the quality of images processed with Mapreduce is much better than before.

LI Chaokui and YANG Wu [14] presented a distributed Storage strategy research of Remote Sensing image based on both MongoDB and SQL server. Their objective was to improve the efficiency of a NoSQL database on the promise of high concurrent access, without significant decline of read/write speed. The results shows that MongoDB has higher time efficiency in data loader and deal with concurrent access better than SQL server. Lin Wang and all [15] provide a study of distributed management of massive remote sensing image data. In the first step, they used an image division technique called Geosot (a global discrete grid system) to get image data blocks. Then, they store them into HBase (a NoSQL database of Hadoop). Experiments results of the former model of remote sensing, storage and process show that, every time they increase both data volume and data node, the time of data import and process decreases.

All of this works mainly focuses on No SQL databases and relational databases for image data management. They either compare between No SQL databases and SQL databases to find how gives better results on distributed parallel storage of image data, or they focuses on a single No SQL database to extract their benefits. In this case, we had found that probably most of the works are sure that No SQL databases are efficient for image data storage and the performance advantage is more noticeable for large mass of image data compared to Relational databases ones. Therefore, According to these results we are almost certain that No SQL databases are efficient to store mass image data. So that, we had seen that is better to compare these No SQL databases themselves (Hbase, MongoDB) which are used in the former works, as a second step. Also, some authors think that a comparison with other massive data management systems is an important issue, for that reason we had added another database which is Cassandra to be compared with them (which is not used in the above literature). In this work we aimed to use larger image data size and the cloud computing as a framework (which is not used in the above literature).

III. Preliminaries

A. Sensors and NoSQL-databases

Due to the rapid advances in satellite technologies, sensors are everywhere. Therefore, the size and variety of the data they generate are growing from day to day and increasing with important rates [2]. Consecutively, new concepts and technologies are rising as the types and usage of sensors expands regularly. Storing and processing such Big Data require humongous costs with traditional relational database SQL. Therefore, non-rational database has arisen and developing very fast [4]. These new ways of creating and manipulating data stores are known as NoSQL databases (commonly defined as "Not Only SQL") [5], are mainly aimed to respond on new requirements, for sensor data storage platform such as the scalability and availability. These databases have the ability to efficiently distribute data over very large number of servers and dynamically add new attributes to data records. No SQL databases are the next generation supposed to respond some of this points; mainly being non relational, open source, distributed, and horizontally scalable [6].

Souvas MAzumder proposed four main layer structure of NoSQL databases which are divided into four groups according to their caracteristics [16] (Fig.1). This paper mainly introduces Apache Cassandra, Apache HBase, MongoDB.

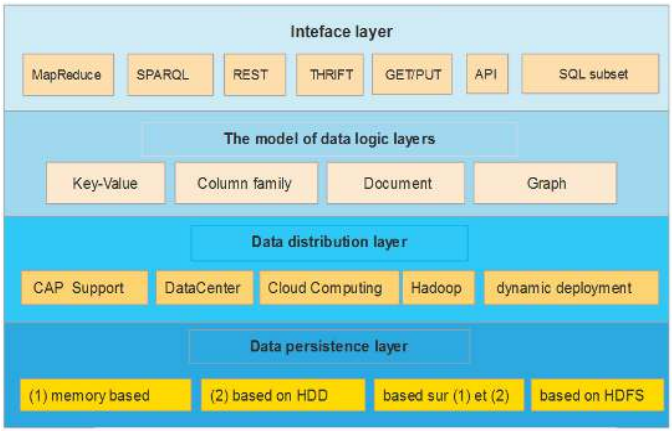


Fig. 1. The system architecture of NoSQL

B. Apache Hadoop, HDFS and MapReduce

Apache Hadoop is an open source project of the Apache Foundation. This framework is intended for distributed storage and distributed processing of very massive amounts of data on computer clusters built from commodity hardware. It is the best answer to parallel data processing, Hadoop is an optimized framework to handle large amounts of data type (structured, unstructured, semi structured) [10]. The main core of Apache Hadoop consists of two sub projects which are HDFS [17] (the Hadoop distributed file system) as a storage part, and MapReduce [18] as a processing part.

1) Key Components of Apache Hadoop:

- *Hadoop Common :* The common utilities that support the other Hadoop modules.
- *HDFS :* A distributed file system that provides high-throughput access to application data.
- *Hadoop YARN :* A framework for job scheduling and cluster resource management.
- *Hadoop MapReduce :* A YARN-based system for parallel processing of large data-sets.

The fundamental assumption of Hadoop , is that all it's modules are designed to automatically handle any hardware failures. First of all, HDFS splits files into blocks, and distributes them across different nodes in the cluster. Then, to process those data, it transfer packaged code for nodes to be processed in parallel by Mapreduce. To resume, Apache Hadoop framework, is a best solution to develop applications which are capable to be runs on clusters of computers, moreover it could perform complete statistical analysis for large volume of data [10].

C. Blocs Image Encoding Method

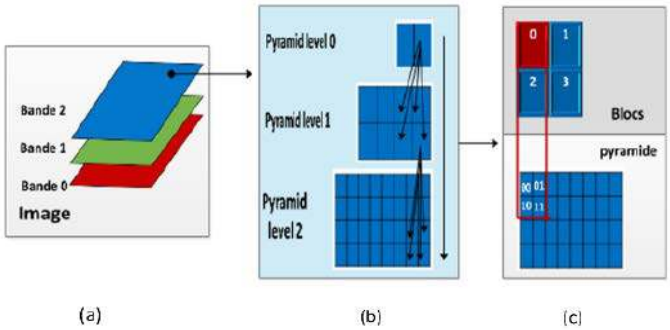


Fig. 2. Pyramid map model of bloc

Is a kind of storage structure of multi-resolution images, in accordance with the resolution of the image from high to low level (see (a) in Fig.2). According to the multi-resolution pyramid design, a large map should separate to some different zoom layers and split it into some same scale little tile (see (b) in Fig.2). Later this tiles can be stored in the NoSQL database by using its ID as the row key and its data as the value. The subdividing image process is as follow; first original image is considered as the bottom layer of the pyramid, signed as level 0. Then a new upper layer will be created by image re-sampling method. The next step to do is to repeat the step two to the top layer which is defined in advance. The pyramid dimension, reading speed, image depository with four or five layers are very important information which can be given to take highest performance .

IV. System design

We have illustrated our approach in Fig.3. The system architecture was mainly based on big data technologies in order to provide a distributed and scalable infrastructure for remote sensing image data management. However, it consists of three main parts which are ; (1)Remote sensing data collection, and (2) Remote sensing data storage subsystem.

A. Remote sensing data collection

In this study we used remote sensing sensors as data generators. It is important to keep in mind that sensors have served in the advancement of various technologies, we can list world mapping, arial photography for military surveillance, assessment of condition of rural roads ; GPS (Global positioning system) and so on. However, sensors applications are not limited as only imaging devises. Instead they have much more missions. For example, Remote sensing which is one of the various innovations that were possible, thanks to many satellites roaming around the earth. In our project we used "Copernicus" [19] as our Remote Sensing data source. However our system architecture is convenient for other types of sensor networks.

B. Remote sensing data Storage subsystem

Usually, data collected from the sensors devices are stored in some sort of a data storage solution. However it becomes a non-trivial task to continuously store these data's, especially because the number of sensors and the amount of data continues to increase. The traditional sensor data storage solutions are able to store data's for only few period of times. However there is a big value in the data collected from sensors since they might carry hidden motifs for faults or diagnostic information. As result, we aim in this section to create a scalable and distributed data storage subsystem in order to store massive satellite images from sensors, until they are processed and lately analyzed. To this end we come out with the new movement of NoSQL databases. These databases are open source and provide an efficient alternatives for big amount of sensor data storage. In this study we used three popular databases (See the former section), MongoDB [9], Cassandra [7] and HBase [8]. MongoDB the document-oriented database which store data on a JSON-Style documents, provides high availability, high performance and easy scalability. Cassandra the column oriented database, provides a very high availability with No Single Point Of Failure (NSPOF), high scalability,

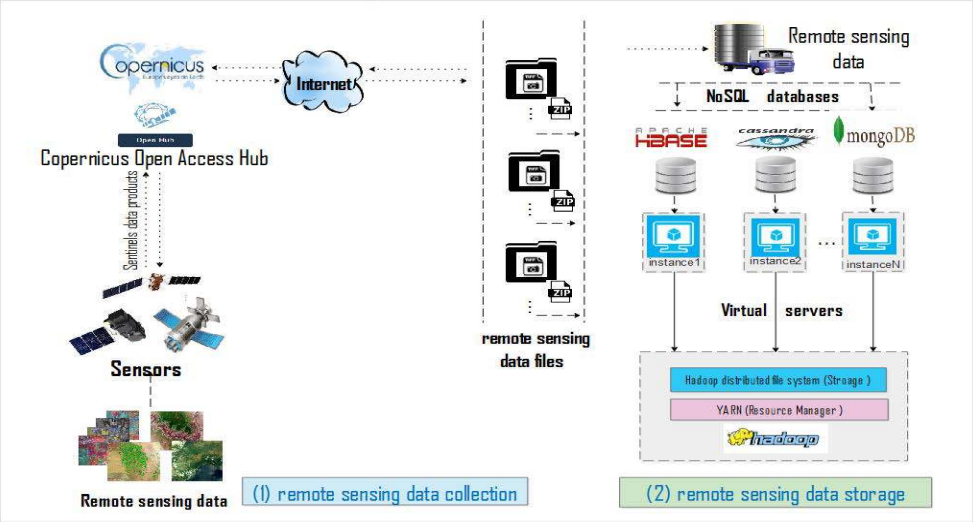


Fig. 3. Architecture of Remote Sensing data management system.

fault tolerant and a tunable consistency. This database is totally distributed thanks to it's "master-less" architecture. HBase, the Hadoop database which is also a column oriented database. It provides a quick random access to huge amounts of structured data and it's run on the top of HDFS (Hadoop Distributed File System).

C. Storage method

According to the idea of partition "the model of Blocks of pyramid map", (Fig. 2) ,we used to divide the image data into various layers and various blocks. Each image has several bands, for example (multi-spectral, hyper-spectral and ultra-spectral images). Each band is a compound of multiple pyramid layers and a number of blocks can constitute a pyramid layer. The block can be used to identifies in the pyramid layer with its row number and column number. The implementation of our methodology is illustrated in flow chart in Fig. 4. First of all, we have an image data which is considered as our data-set. Then for each band, a group of pyramids are created. For each pyramid, we extract the blocks in sequence and write them into each database API (HBase, Cassandra, MongoDB). Lastly, we insert a new record into each database table's or collection's to represent the new image file.

1) Data model storage on Hbase: we designed two tables, HRaster MetaDataInfo ColumnFamily and HRaster DataInfo ColumnFamily, to respectively store the metadata and the spatial data block of the images. When a new image is imported, a new record will be created in HRaster MetaDataInfo ColumnFamilye to store meta-data and a new HRaster DataInfo ColumnFamily will be created to store the specifc blocks. As result,there will be many HRaster DataInfo ColumnFamily Column Family's to store different images.

2) Data model storage on MongoDB: In order to store the our images data in the MongoDB databases which are in XML format were transformed into JSON files. Then we have used the GridFS (Grid File system) which is a specification for storing and retrieving files that exceed the BSON-document size limit of 16 MB. Instead of storing a file in a single document, GridFS divides the file into parts, or chunks, and stores each chunk as a separate document. By default, GridFS uses a chunk size of 255 kB. The GridFS uses two collections to store files. One collection stores the file chunks, and the other stores file metadata.

3) Data model storage on Cassandra: Just like in HBase, we designed two tables, CRaster MetaDataInfo ColumnFamily and CRaster DataInfo ColumnFamily, to respectively store the metadata and the spatial data block of the images. When a new image is imported, a new record will be created in CRaster MetaDataInfo ColumnFamilye to store metadata and a new CRaster DataInfo ColumnFamily will be created to store the specific blocks. As result,there will be many CRaster DataInfo ColumnFamily Column Family's to store different images.

V. Evaluation

A. Experiment Environment and Datasets

We designed five groups of Hadoop and databases cluster for our tests, each cluster has different number of nodes. The Specific composition and node feature are respectively shown in the following tables I, II

TABLE I
USED DATA NODES UNDER DIFFERENT CLUSTER CONFIGURATIONS

Cluster	Members
1 node	master
2 nodes	master, 1*slave
4 nodes	master, 3*slave
8 nodes	master, 7*slave
16 nodes	master, 15*slave

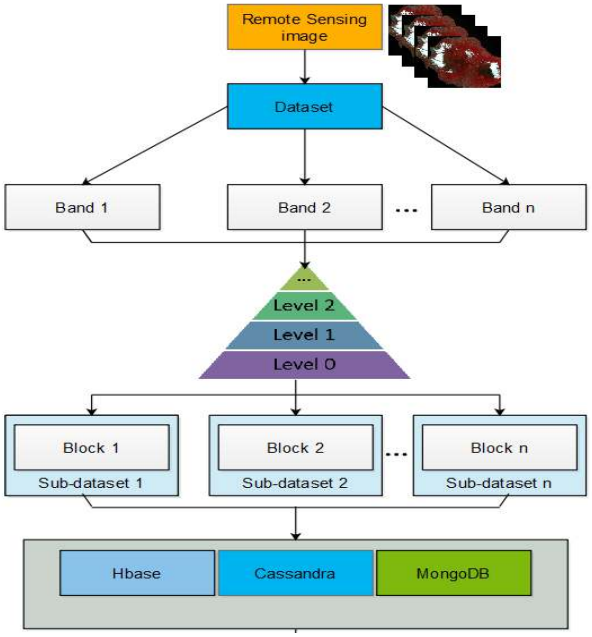


Fig. 4. Implementation of the storage schema

TABLE II
CLUSTER NODE FEATURES INFORMATION

Node type	Master	Slave
Model	m4.large	t2.small
vCPU	2	1
RAM	8 GB	2 GB
Credit CPU per hour	12 $ /hour	6 $ /hour
Processor	2,3 GHz Intel Xeon	2,4 GHz Intel Xeon

B. Remote sensing image data information

We used five different groups of remote sensing image data in this experiment respectively as follows 274MB, 1.16GB, 8GB, 16GB 100GB more detailed descriptions of these remote sensing images are in the table below III

TABLE III
REMOTE SENSING IMAGE INFORMATION

File groups	Number of images	Size	Bands	Dimension (width*height)
1 group	1 image	274MB	4	6000*6000
2 group	1 image	1.16GB	4	21267*14652
3 group	6 image	8GB	4	21267*14652
4 group	13 image	16GB	4	21267*14652
5 group	86 image	100GB	4	21267*14652

1) Scalability on cluster size: to measure the performance of the cluster databases implementations with regard to computation time, the total image size was kept constant while increasing the number of nodes in the cluster. First, we evaluate the computation time on a single computer and then increase number of nodes in the system until 16 nodes. Fig.5 show the time consumed to store the eight images files from the first group (Tab.III). From this figure it can be observed that the importation time significantly decreases with increasing number of computers for all databases. The Cassandra database has better performance compared to the HBase and MongoDB databases. For more clarity of the observation above we used to compute the **Speedup** and **Efficiency** from the above results.

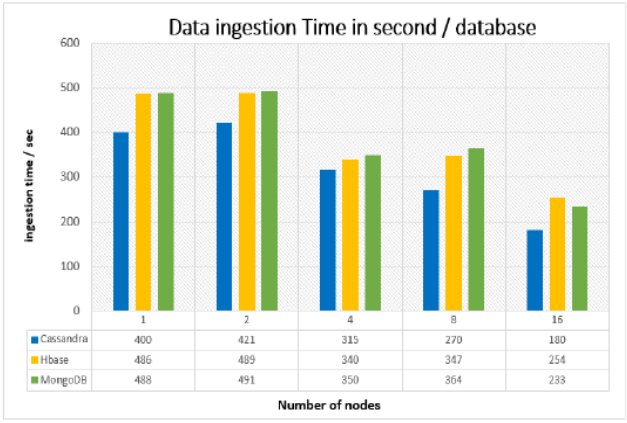


Fig. 5. Histogram of Scalability of Cassandra, HBase, and MongoDB databases

$$Speedup(n) = \frac{Time\ on\ single\ node}{Time\ on\ n\ nodes} \quad (1)$$

$$Efficiency(n) = \frac{speedup\ (n)}{n} \quad (2)$$

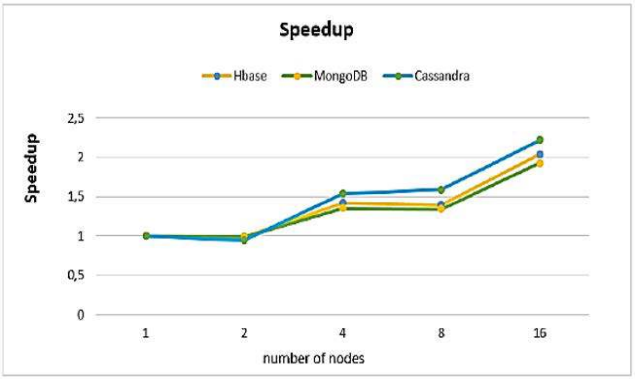


Fig. 6. Speedup

In an ideal situation the speedup increases linearly with increase in number of computing processors. According to [20] processes with greater than 0.5 (50%) efficiency are considered to have achieved good performance. However, a significant deterioration of speed up and efficiency is observed when the four node was added to HBase and MongoDB while Cassandra database retains its linearity. Then the speedup increased again when the 8th node is

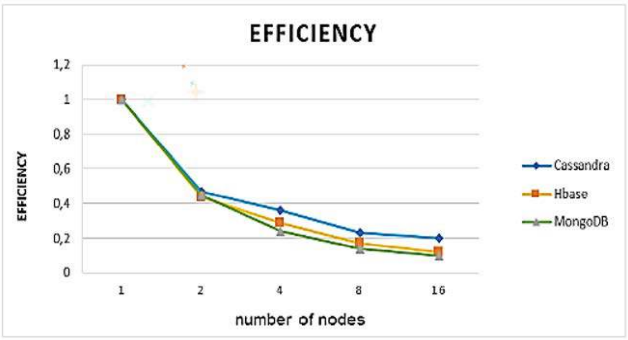
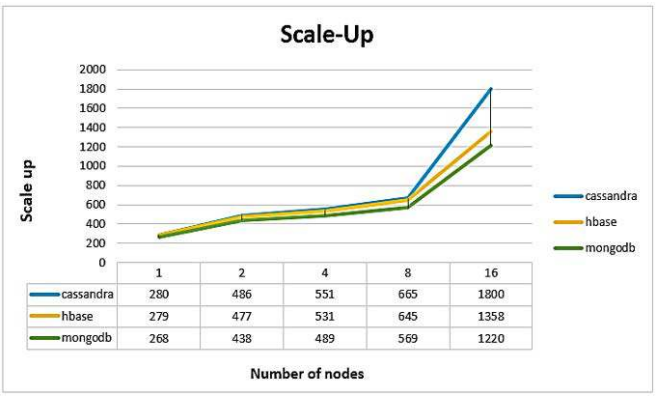


Fig. 7. Efficiency

added to the system and better than the previous rate. From this it can be deduced that better performance can be achieved if the number of computing nodes are increased or if there is a high-speed network connection between the nodes.

2) Scalability on size of the data: To investigate how the each database model reacts to very large remote sensing images, the Scale-up approach was evaluated where the importation time was measured while increasing the data size and number of nodes by the same fold. By using the Scale-up approach we can investigate the bottlenecks in the distributed system as the each computer node is storing the same amount of data and the effect load balancing problem and inefficiencies caused by the distributed method is avoided according to [21]. In our opinion this metric is the best way of evaluating the capability of the databases implementation to cope with different sizes of data. Ideally the graph of Scale-up will have a straight horizontal line where data size makes no impact with the importation time staying constant. In this experiment we started with data size of 1.16 GB then the size data was increased by adding other data size respectively , 4.64 GB, 8 GB, 16 GB, 100 GB and every time we add an instance node.



	1	2	4	8	16
cassandra	280	486	551	665	1800
hbase	279	477	531	645	1358
mongodb	268	438	489	569	1220

Fig. 8. Scale-Up

Fig 8 shows the Scale-Up results. All databases (Cassandra, HBase, and MongoDB) scale well until the second computing node but it increases again when the third computer is added to the system indicating that the network connection bandwidth is the main bottleneck of the distributed system. According to this result we can infer that if the data-nodes has been connected through high speed network, they would have scaled-up well and the graph would have looked more flat. Cassandra database Scale well, better that the other two databases.

VI. Conclusion

In this work we studied the distributed management of massive remote sensing image data based on NoSQL database and pyramid map . We presented a storage method to divide remote sensing image data into blocks based on pyramid map and store the data blocks into three different database model Cassandra, HBase and Mongodb. The feasibility of the presented storage model for massive remote sensing image data has been verified. Finally, we came out with the Cassandra as the most suitable database model for our big remote sensing data management approach.

References

[1] By Mingmin Chi, Antonio Plaza, Jon Atli Benediktsson, Zhongyi Sun, "Big Data for Remote Sensing: Challenges and Opportunities". Proceedings of the IEEE. Vol. 104, No. 11, November 2016.
[2] G. Aydin, I. Riza Hallac, and B. Karakus, "Architecture and Implementation of a Scalable Sensor Data Storage and Analysis System Using Cloud Computing and Big Data Technologies", Journal of Sensors Volume 2015, Article ID 834217, 11 pages http://dx.doi.org/10.1155/2015/834217, 2015.
[3] Y. Liu, H.Yu, Y.Zhao, Z. Huang, Y. Fang,"Applying GPU and POSIX Thread Technologies in Massive Remote Sensing Image Data Processing," Proceedings of the 19th International Conference on Geoinformatics, pp. 1-6, 2011.
[4] Yuehu Liu1, Bin Chen1*, Wenxi He2,Yu Fang1, "Massive Image Data Management using HBase and MapReduce,2015.
[5] https://nosql-database.org/
[6] Book of Martin Fowler and Pramod J. Sadalage, "NoSQL Distilled : A Brief Guide to the Emerging World of Polyglot", 2013.
[7] https://hbase.apache.org/
[8] http://cassandra.apache.org/
[9] https://www.mongodb.com/fr
[10] http://hadoop.apache.org/
[11] Zhifeng Xiao, Yimin Liu, 'Remote sensing image database based on NoSQL Database'. IEEE 978-1-61284-848-8/11/26.00 ,2011
[12] Jan Sipke van der Veen, Bram van der Waaij, Robert J. Meijer, 'Sensor Data Storage Performance : SQL or NoSQL, Physical or Virtual'. IEEE Fifth International Conference on Cloud Computing, 2012.
[13] Yuehu Liu ,Bin Chen ,Wenxi He ,Yu Fang, 'Massive Image Data Management using HBase and MapReduce', Published in : Geoinformatics (GEOINFORMATICS) 21st International Conference on 20-22 June, 2013.
[14] LI Chaokui, YangWu, 'The Distributed Storage Strategy Research of Remote Sensing Image based on Mongo BD'. IEEE 978-1-4799-4184-1/14, 2014.
[15] Lin Wang, Chengqi Cheng, 'Massive remote sensing image data management based on Hbase and Geosot'. IEEE 978-1-4799-7929-5/15, 2015.
[16] Shen Shu. Research on NoSQL data technology and application[D]. Nanjing: Nanjing University of Infonnation Science and Technology, 2012.
[17] https://hadoop.apache.org/docs/r1.2.1/hdfs-design.html
[18] https://hadoop.apache.org/docs/r1.2.1/mapred-tutorial.html
[19] http://www.copernicus.eu/
[20] Eager, D. L., J. Zahorjan, and E. D. Lazowska, Speedup versus efficiency in parallel systems, Computers,IEEE Transactions, 1989.
[21] I. Glendinning, Goller, A., D. Bachmann, and R. Kalliany, "Parallel and Distributed Processing, in Digital Image Analysis" , Springer New York, 2001.