Data Searching In Cloud Computing Environment Based on Map Reduce

Devarapalli Narasa Reddy[1], Dasari Rajesh[2]

[1, 2]Computer Science and Engineering, Rise Group of Institutions, Ongole, India

Abstract-***Cloud has been an revolution in the IT-Services computing. cloud frames availability of large amount of data resources for its users which signifies the fact that an additional data modeling structure has to designed for faster and efficient search and provide the right content to the analyst. Time consuming and non rapid searches cannot withstand the given scenario, which prompted many researches to work in these area, in these paper we present an case study based on MapReduce and yarn over Hadoop or HDFS and show case an Trusted and Efficient data Searching Mechanism using MapReduce over Cloud Environment.***

Keywords: Hadoop, HDFS, Map Reduce, Cloud Environment

1. Introduction

Cloud computing [1] is entering our lives and changing the way people consume information dramatically. Clouds transform IT infrastructures with an emphasis on making them flexible, affordable, and capable of serving millions of users, satisfying their computing or storage demands. The design of early cloud computing systems has evolved from, and was dominated by, the concepts of cluster and grid computing. Currently, as the concepts of the cloud become advanced and mature, cloud networking and communication processes begin playing a central role. Cloud Networking has emerged as a promising direction for cost-efficient and reliable service delivery across data communication networks. The dynamic location of service facilities and the virtualization of hardware and software elements are stressing the communication network and protocols, especially when datacenters are interconnected through the Internet.

The optimisation of cloud networking can significantly increase system performance, reducing energy consumption and save costs not only inside individual data centers, but also globally, on the Internet scale. Developing novel network architectures would facilitate adoption of modular container-based data centres. Advancements in internet working become key enabler for building hybrid clouds and federations of clouds. Service provisioning over heterogeneous connections and wireless links can enhance computational capacity and enrich application experience of mobile users. Efficient resource management and scheduling in data centres and cloud infrastructures is open research challenge that has to be addressed and novel architectures, telecommunication technologies, and protocols must be developed to ensure efficiency of future cloud computing systems.
Apache Hadoop with MapReduce is the workhorse of distributed data processing[2]. With its unique scale-out physical cluster architecture and its elegant processing framework initially developed by Google, Hadoop has fostered explosive growth in the new field of big data processing. Hadoop has also developed a rich and diverse ecosystem of applications, including Apache Pig[4], which is a powerful scripting language, and Apache Hive[3], which is a data warehouse solution with a SQL-like interface. Unfortunately, this ecosystem is built on a programming paradigm that cannot solve all problems in big data. MapReduce provides a specific programming model that, although simplified with tools like Pig and Hive, is not a big data panacea. Let's begin our introduction to MapReduce 2.0 (MRv2) — or Yet Another Resource Negotiator (YARN) — with a quick review of the pre-YARN Hadoop architecture.

The concept of MapReduce [7] is very simple to understand for those who are familiar with clustering and data processing techniques. It is a programming model that is associated with the implementation of processing and generating large data sets.

MapReduce[2] function on data sets of key & value pair is the programming paradigm of large distributed operation [1]. The data flow architecture of MapReduce shows the techniques for analyses and produce the indexes [10] and is shown in figure1.

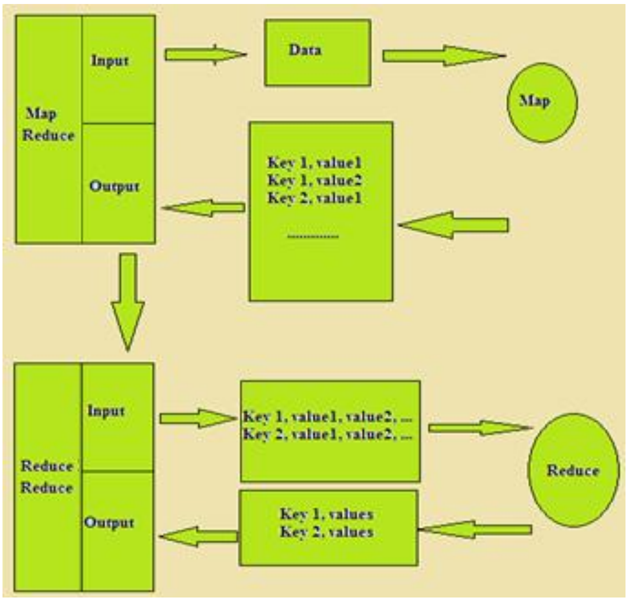

Figure 1: MapReduce data flow

Our main purpose of this work is to develop a technique for a fast and efficient way of searching data in the MapReduce paradigm of the Hadoop Distributed File System The framework of Hadoop MapReduce is a master – slave architecture, which has a single master server called *job tracker* and several slave servers called *task trackers*, which run one per node in the cluster [3] MapReduce operation on data sets of key and value pair in the programming paradigm of large distributed operation [1].

In this paper, we have presented novel architecture and design with dynamic scaling scenario to investigate the performance of Hadoop in high speed retrieval of data in the cloud environment by replacing the map phase of the MapReduce paradigm with a web server (jetty). These servers (jetty) is a pure Java based server consists of Java servlets. We also implement a searchable mechanism in HDFS using indexing in both Name Node and Data Node.

The rest of the paper is structured as follows. Module II & III highlights the related works and preliminaries. Module IV narrates the proposed methodology technology. Module V narrates the implementation details followed by the results and analysis in module VI. Finally work done and future scope is concluded in module VII.

2. Related Work

In February 2003 first MapReduce library was introduced @Google. MapReduce was made easy for data processing on large cluster in 2004 [4].

In December 2005 Doug cutting notices that the MapReduce uses Nutch which is a searching engine. After that Hadoop moves out of the Nutch in February 2006. Hadoop was at first designed on the basis of Google's MapReduce in which an application was run down into many small parts. In April 2007 yahoo made Hadoop to run on more than 1000 nodes in a cluster [9]

In 2008 information retrieval framework consists of network nodes, which was shared and peer was maintained in B+ tree containing IP hash values [3]. In January 2008 Hadoop made apache as a top level project center.

In July 2009 new Hadoop subproject was found where Hadoop core was renamed as Hadoop Common. In this subproject Hadoop distributed file system (HDFS) and MapReduce get separated for doing separate work. HDFS is designed to store very large data sets [7]. In 2010 web application based process is used to handle high throughput traffic [6].

Facebook in 2010 has declared that they have the largest Hadoop cluster in the world with 21 PetaByte of storage, and it's grown up to 30PetaByte in 2011 July.

In 2011 May MapRreduce have reported the availability of an alternate file system for Hadoop, that will replace the file system of HDFS with a full random access to read and write file system [5].

3. Preliminaries

3.1. Hadoop Architecture

Hadoop is an open source cloud computing environment which supports the large data sets in a distributed computing environment, primarily in data warehouse systems and plays an important role in support of big data analytics to understand the user behavior and their needs in web services. It stores application data and file system separately [11].

Hadoop Distributed File System is highly fault-tolerant, which provides high throughput access to the application data and is a perfect application that has large data sets. It has the Master and Slave architecture. The cluster of Hadoop Distributed File System consists of DataNode and NameNode [11] shown in Figure 2.

NameNode which is a central server, also called as a Master server, is responsible for managing the file system namespace and client access to files and DataNode in the cluster node are responsible for files storage.

Google introduced MapReduce to hold up large distributed computing, on large clusters of computers to implement huge amounts of data set. MapReduce is a structure that processed parallel problems of big data set using a large number of computers that is collectively referred to as a cluster or a grid.

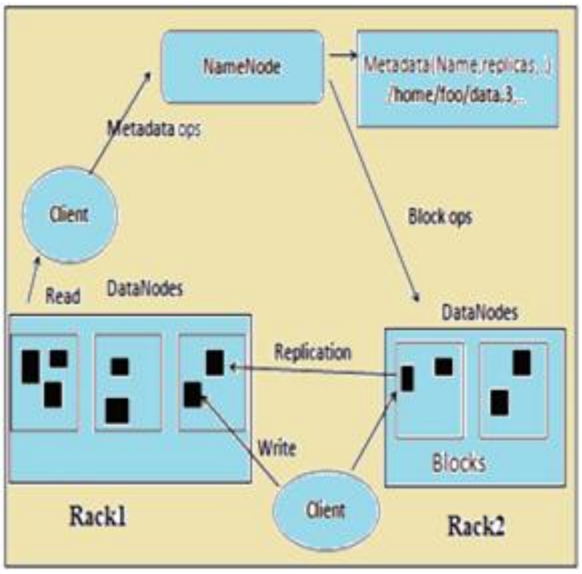

Figure 2: HDFS Architecture

4. Proposed Methodology

The data in Hadoop Distributed File System is scattered which makes searching and data retrieval is one of the most challenging tasks in HDFS. So to overcome this challenge the proposed system provides a searchable mechanism in HDFS. The MapReduce mechanism of Hadoop is considered as the web server in the map phase. This web server is implemented by Application Server Tomact and is considered as the web interface. Web applications are generated in the HDFS and the web server is started in the form of Hadoop job(Json Creator) by submitting our keys to URL Hadoop url/file/key shown in the figure 3.

Figure 3 gives the insight of the proposed architecture. Following steps explain the steps of the proposed technology.

1) When the client makes a request in the Hadoop *i.e. Hadoop url/file/key,* the key is passed to the NameNode.
2) The web server which is running in the NameNode is divided into the master server and a slave server. The master server will have multilevel index and slave will have data and key index.
   a) The master server will do the entire indexing task and slave server will do the entire retrieval task.
   b) The master server will take the key from the client and give to slave server.
   c) These slave servers will check for the key inthe index of NameNode
   d) On finding, the key will be sent to the Master server and the Master server will pass the received key to the DataNode.
3) In the DataNode the web server is again divided in Master and slave server which do the indexing and retrieval task of data.
   a) Master server takes the key from the NameNode and passes the key to the slave server.
   b) Slave server takes the key and goes to the header where key of different ranges [a-m, n-z and 1-9] have been stored.
   c) The slave server with key directly goes to the record where ranges have been stored.
   d) After getting the key and its value it sends the key & value pair to Master server
   e) The Master server in turn passes the key & value pair to the NameNode.
   f) Finally NameNode's Master passes the value to the requesting client.

4.1. Web server in the NameNode

The web server in the NameNode is divided into Master and Slave server. The Master server contains a multi - level index and slaves have data and key index. When the data request is made, the web server will pass the key on to the multi-level index. Indexing all the process is done by Master server and all retrieval process are done by slave servers.

4.2. Application Web server in the DataNode

The web server in the DataNode is also divided into Master and Slave server and the DataNode contains all the records and B+ tree as a header in web server.

4.3. Read

B+ tree is implemented in the header of the DataNode, which finds the address of the data stored in the record by using the key from the web server (figure 5).

4.4. Write

After writing data in the DataNode the header length is updated (figure 4). The load balancer is implemented in between clients and servers to balance the work load of the web server

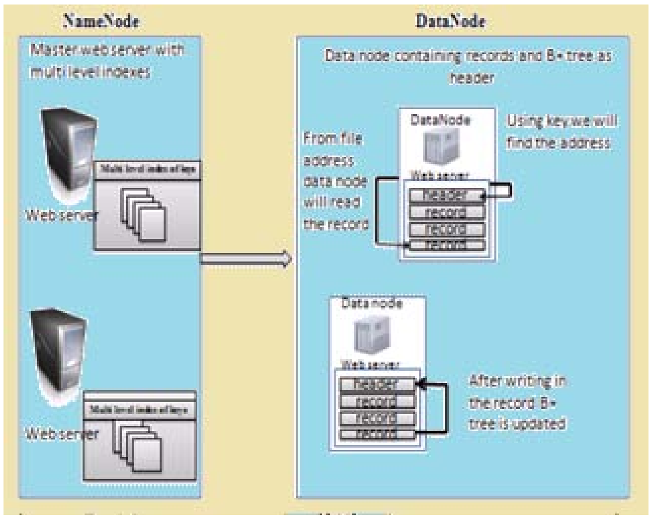
NameNode
DataNode
Master web server with multi level indexes
Web server
Multi level index of keys
Web server
Multi level index of keys
Data node containing records and B+ tree as header
DataNode
From file address data node will read the record
Using key we will find the address
Web server
header
record
record
record
Data node
Web server
header
record
record
record
After writing in the record B+ tree is updated

Figure 3: Proposed Architecture

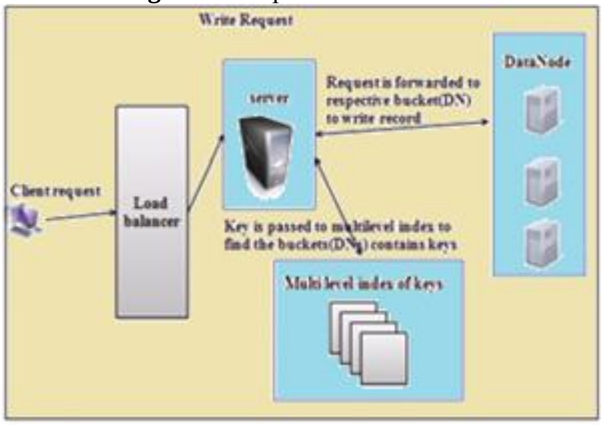
Write Request
server
Request is forwarded to respective bucket(DN) to write record
DataNode
Client request
Load balancer
Key is passed to multilevel index to find the buckets(DNs) contains keys
Multi level index of keys

Figure 4: Flow diagram for write request

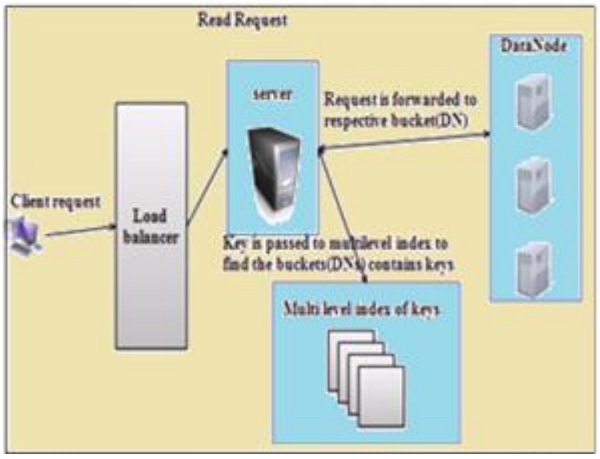
Read Request
server
Request is forwarded to respective bucket(DN)
DataNode
Client request
Load balancer
Key is passed to multilevel index to find the buckets(DNs) contains keys
Multi level index of keys

Figure 5: Flow diagram for read request

5. Implementation

With the new power that YARN provides and the capabilities to build custom application frameworks on top of Hadoop, you also get new complexity. Building applications for YARN is considerably more complex than building traditional MapReduce applications on top of pre-YARN Hadoop because you need to develop an ApplicationMaster, which is the ResourceManager launch when a client request arrives. The ApplicationMaster has several requirements, including implementation of a number of required protocols to communicate with the ResourceManager (for requesting resources) and NodeManager (to allocate containers). For existing MapReduce users, a MapReduce ApplicationMaster minimizes any new work required, making the amount of work required to deploy MapReduce jobs similar to pre-YARN Hadoop.

In many cases, the life cycle of an application in YARN is similar to MRv1 apps. YARN allocates a number of resources within a cluster, performs processing, exposes touchpoints for monitoring of the progress of the application, and finally releases resources and does general cleanup when the application is complete. A boilerplate implementation of this life cycle is available under a project called Kitten (see Resources). Kitten is a set of tools and code that simplifies the development of applications in YARN, allowing you to focus on the logic of your application and initially ignore the details of negotiation and running with the constraints of the various entities in a YARN cluster. If you want to go further, however, Kitten provides a set of services that you can use to handle interactions with other cluster entities (such as the ResourceManager). Kitten comes with its own ApplicationMaster, which is usable but shipped primarily as an example. Kitten makes strong use of Lua script as its configuration service.

6. Proof of Concept

We considered Data base of Hospital and created an unending data considering of patient information. Apply the concept of searching key value over the large patient data and extracting the required information using the MapReduce with Yarn in Hadoop Environment has yielded less time.

7. Conclusion and Future Scope

In this paper, an architecture and design with dynamic scaling scenario to investigate the performance of Hadoop in high speed retrieval of data in a cloud environment is presented. Although Hadoop continues to grow in the big data market, it has begun an evolution to address yet-to-be-defined large-scale data workloads. YARN is still under active development and may not be suitable for production environments, but YARN provides significant advantages over traditional MapReduce. It permits the development of new distributed applications beyond MapReduce, allowing them to coexist simultaneously with one another in the same cluster. YARN builds upon existing elements of current Hadoop clusters but also refines elements such as the JobTracker to increase scalability and enhance the ability to share clusters by many differing applications. YARN, with its new capabilities and new complexity, will soon be coming to a Hadoop cluster near you.

References

[1] L. Youseff, M. Butrico, and D. Da Silva, ―Towards a unified ontology of cloud computing," in *Proc. 2008 Grid Computing Environments Workshop.*

[2] Apache Hadoop project .[online] http://www.ibm.com/developerworks/library/bd-adoopyarn/

[3] Apache Hive .[online] http://www-01.ibm.com/software/data/infosphere/hadoop/hive/

[4] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, ―Hive – a warehousing solution over a Map-Reduce framework," Proceeding very large databases (PVLDB), 2009, vol. 2, pp. 1626–1629.

[5] Bhandarkar, M. ―MapReduce programming with apache Hadoop", parallel & distributed processing international symposium on digital object identifier, IEEE, April 20 2010, page 1-5.

[6] G. SubhaSadasivam, V.Karthikeyan, P. Raja, ―Design and implementation of a novel peer-to- peer information retrieval framework", March 2008, pages: 1-9.

[7] J. Dean and S. Ghemawat, ―MapReduce: Simplified data processing on large clusters," in Operating system design and implementation (OSDI), a computer science conference, 2004, pages: 137–150.

[8] Jin Ming Shih;Chih Shan;Ruay Shiung Chang, ―Simplifying MapReduce data processing", Utility and cloud computing(UCC), 4th IEEE International conferences on digital object identifier,2011,pages:366-370.

[9] Krishnan S.counio, J.C. ―Pepper: An Elastic web server farm for cloud based on Hadoop" Cloud computing technology and science, IEEE second international conferences on digital object identifier ,2010,pages 741-747.

[10] Tom White, ―Hadoop: The Definitive Guide", First Edition, Yahoo press, June 2009.

[11] Apache Hadoop project (2012). [Online]. Available:http://hadoop.apache.org/.

[12] Yongqiang He, Rubao Lee, Yin Huai, Zheng Shao, Jain,N., Xiaodong Zhang, ZhiweiXu, ―RCFile: A fast and space efficient data placement structure in MapReduce based warehouse systems", in Data engineering(ICDE), IEEE 27th international conference on digital object identifier,2011,pages 1199-1208.

[13] Jin Ming shih, Chin-shan lao,Ruay-shing chang , ―Simplifying MapReduce Data Processing", 2011 fourth IEEE international . Conference on Utility and cloud computing, pages-366-370.

[14] Xingguo Cheng, Nanfeng Xiao,Faliang Huang ―Research on HDFS-based Web Server Cluster", 2011 International conference on Digital object identifier, pages: 1-4.

[15] Steffen Heinzl,Christoph Metz, ―Toward a Cloud-ready Dynamic Load Balancer Apache Web Server", 2013 Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise, pages 242-245.

[16] NIST Definition of Cloud computing v15, csrc.nist.gov/groups/SNS/cloud-computing/cloud-efv15. doc.

[17] Daniel Grosu, M.S, ―Load Balancer in Distributed Systems: A Game Theoretic Approach", The University of Texes at San Antonio, May 2003.