ASEE 2014 Zone I Conference, April 3-5, 2014, University of Bridgeport, Bridgeport, CT, USA

Improvements in Big Data Hadoop

Several hybrid efficiency methods

Ye Zhou[1]
Department of Electrical and Computer Engineering
University of New Haven
West Haven, CT, USA
yzhou2@unh.newhaven.edu[1]

Amir Esmailpour[2]
Department of Electrical and Computer Engineering
University of New Haven
West Haven, CT, USA
aesmailpour@newhaven.edu[2]

Abstract **- In recent years, database management system (DBMS) has become more prominent in the industry due to the staggering amount of structured and unstructured data. Dealing with management of such data is a challenge for DBMSs. According to several surveys, many company engineers choose Hadoop for their premier management system [5], due to its flexibility, ease of management and being open source software. However, more recently, database administrators of big company cite that Hadoop does not always work in their best interest in specific areas of data management. In order to solve big-data problems by providing considerable power of computability, Hadoop sacrifices efficient data processing. In the real-time data processing, Hadoop has lower speed than traditional DBMSs because it does not use high-level language, such as SQL and other query optimized languages. In this paper, we discuss methods to improve efficiency. Such improvements increase the ability of data processing in real-time, hence making Hadoop more useful.**

Keywords-- Efficiency, Hadoop, MapReduce, Big Data

I. INTRODUCTION

Hadoop framework was initially developed by Doug Cutting and Mike Cafarella [3] in 2005 to support part of the Nutch project [4]. At birth, Hadoop was a giant development for dealing with the new and unexpected amount of data generated by several industries such as health sciences, security, and social media. Hadoop includes MapReduce engine using Java implementation and uses specific database file system called Hadoop Distributed File System (HDFS). MapReduce engine includes two functions: Map and Reduce. The main purpose for this engine is the data processing from different locations and then merging them together to achieve the final result. Comparing with traditional database, Hadoop can process vast amount of data in different locations and brings them all together in one central point. Figure 1 shows the basic components of Hadoop.

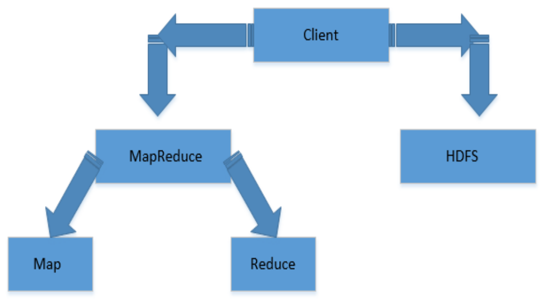

Figure 1: General components of Hadoop

II. PROBLEMS WITH HADOOP

Even though Hadoop has many advantages, it still faces some problems. The main issue for Hadoop is lower data processing efficiency than traditional DBMSs [2].

A. Lack of query optimization

HDFS cannot support SQL language and query optimized techniques. Map and Reduce functions use Java to complement the process. This requires that users create a new function with Java language if they need special functions to satisfy their own requirements. MapReduce encounters difficulties trying to access data from SQL database, and since SQL database is still used in major database required systems, this issue presents a problem for Hadoop.

Figure 2 illustrates MapReduce functionality within Hadoop. Before Mapper function works, one file input is loaded into HDFS, and the file is partitioned into multiple blocks during loading. Every block assigns to each mapper task and gains intermediate result. All intermediate result transfer to Reducer task via HTTPS protocol.

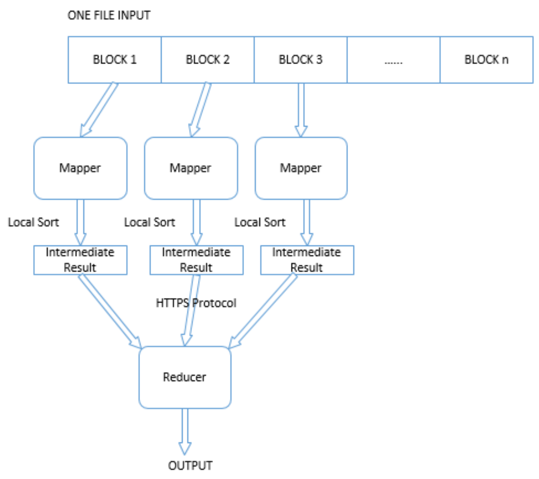

Figure 2: Graphical representation of Map and Reduce collective functions

MapReduce works after all data is loaded into central storage from different locations and processes data without data modeling. This process will be idle since MapReduce engine does not work until getting the data. With traditional databases, different systems fit into different data modeling and minimize data transfer. However, Hadoop will ignore all execution plans and optimizes plans for data modeling and processing of raw data. Therefore MapReduce has lower performance than DBMS.

B. Parallelism is absent in Hadoop

Parallel processing refers to several processors accessing data at the same time; however, MapReduce uses a single workflow processing instead of parallel data processing. In the meantime, MapReduce designs a single read input and output, which means regardless of how many data processing requests messages are sent, MapReduce only works on one at-a-time. A transition to next stage will not be granted until current stage is finished; therefore, MapReduce is a blocking operation.

C. Simplified Scheduling Schemes in Hadoop

In order to achieve ease of usage and fault tolerance, Hadoop has simple scheduling to process data. Hadoop uses default FIFO scheduling for JobTracker nodes. In this method, JobTracker processes data one by one, first in and first out, hence this simple scheduling is not able to handle the realistic problem. Consequently, this one-to-one strategy and simple running scheduler shows lower efficiency in real-time processing.

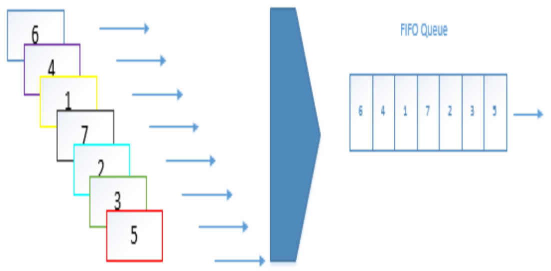

Figure 3: Default FIFO scheduling in Hadoop

Another issue with Hadoop is lacking data optimization before the central station accesses the local results. In DBMS, data is always protected before commit, but in real-time processing, because data never be stored, the system must provide a properly method to avoiding data missing, processing delay or overhead of data.

III. Proposed improvements to Hadoop problems

Since many issues have surfaced since Hadoop was developed, users in big organizations complain that Hadoop is sometimes slow. Many developers in this area try to find the best way to satisfy real-time processing applications, which need high-volume data streams, instead of traditional database systems [1].

In real data processing, data is always moving. Large storage and specific technologies are required. Unique techniques could provide potential improvements for efficiency of Hadoop framework. In the following section we provide some recommendations.

A. Keeping Big Data in movement

In the real-time data processing, to achieve higher efficiency, a system must make sure all request messages are processed without critical processing path. For implementing this process, system should use special functions to deal with the data processing.

We introduce PCF and DCF into MapReduce function of Hadoop. This hybrid method can efficiently improve the speed of data transfer.

MAC sub-layer for wireless network includes two fundamental access methods: distributed coordination function (DCF) and the point coordination function (PCF). DCF utilizes the carrier sense multiple access with collision avoidance (CSMA/CA) approach. PCF is based on polling to determine the station that can transmit next. According to these methods, we can use similar idea with data processing.

During data transfer from different locations, we can add DCF and PCF between MapReduce function and locations.

For example, for DCF, it utilizes the CSMA/CA approach. During the transmission, when one location requests occupies medium, it will send a Request to Send (RTS) to the MapReduce, after the time of SIFS (Short IFS) passes, the MapReduce will send Clear to Send (CTS) back to confirm, then they can connect successfully. When other locations see CTS, they will stop trying to send request.

For PCF, a point coordinator (PC) is an important role in the whole process. The PCF requires PC to control all the transmission. During a special period of time, the PC will ask all locations to set their requirements and make an order list. In this polling processing, it can occur additional overhead for systems. However, it will avoid unexpected collisions and back-off. We will see the architecture for PCF in Figure 4. A PC plays a controller and controls all different nodes, then make a job list to processing.

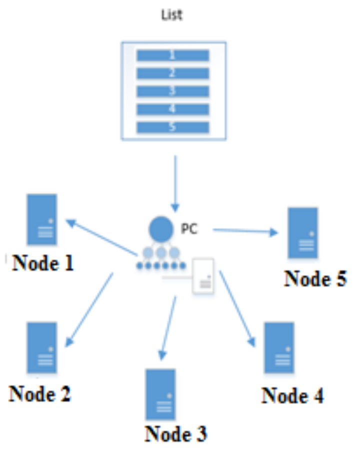

Figure 4: Proposed architecture of PCF for Big Data

Therefore, when the large number of data processing involve into polling, data processing are polled sequentially from a polling list maintained by the PC, which avoids collision and makes the process more efficient.

B. Massive Parallel Operations

With the considerable computation power, MapReduce uses one single I/O operations. Like MPP (Massive Parallel Processing), we should distribute MapReduce function into different locations connecting with a switch. In the switch, it should keep an optimizer, which can be used for data mining. Using this hybrid method, it will improve the processing efficiently. Furthermore, in real-time system, a process wait is not a good idea, so it needs a limitation for time.

Figure 5 shows optimizer switch to distribute the requests among MapReduce units in different locations. After local MapReduce functions work, all results will be sent to the optimizer, which can perform data mining and check data error detection.

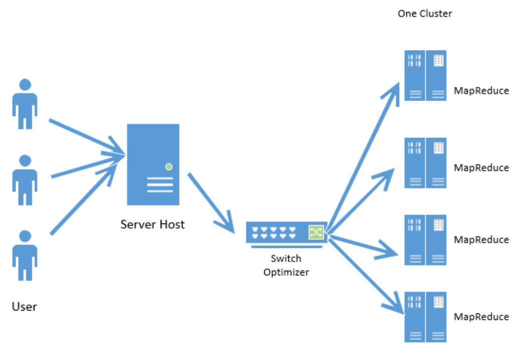

Figure 5: optimizer switch to distribute the requests among MapReduce units

During the transmission of several queues (data processing), the system can divide queues into different segments and process them using different transactions, as illustrated in Figure 6. This method could save time when many data processing are submitting requests simultaneously.

IV. Conclusion

In this paper, we talk about the problem of MapReduce in Hadoop. MapReduce is perfect good for scalability data and also guarantee fault-tolerance. However, MapReduce does not like regular database system, use Java implement all functions. MapReduce encounters difficulties trying to access data from SQL database, because SQL language is still used in major language of database required systems. In addition, the simple scheduler and one single data stream is also the big problem of MapReduce.

In the future, we will establish one simulator, which install a Hadoop and Massive Parallel Processing to simulate the hybrid methods. Also we try to access Mapreduce for several users in same time and test the efficiency of data transfer.

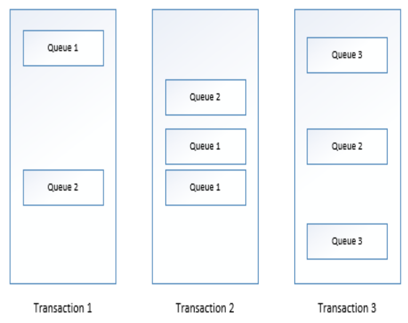

Figure 6: Segregation of data processing among transactions and queues

REFERENCES

[1] Michael Stonebraker, Uğur Çetintemel, Stan Zdonik: The 8 Requirements of Real-Time Stream Processing. ACM SIGMOD Volume 34 Issue 4, December 2005.

[2] Kyong-Ha Lee, Yoon-Joon Lee, Hyunsik Choi, Yon Dohn Chung, Bongki Moon: Parallel Data Processing with MapReduce: A Survey. ACM SIGMOD Volume 40 Issue 4, December 2011.

[3] MikeCafarella and Doug Cutting. Documents available on line in March 2014 at:
http://web.eecs.umich.edu/~michjc/
http://cutting.wordpress.com/

[4] Nutch Project available on line in March 2014 at:
https://nutch.apache.org/

[5] JESSICA E. VASCELLARO. "Hadoop Has Promise but Also Problems (2012, Feb 23), available on line in March 2014 at:
http://online.wsj.com/news/articles/SB10001424052970203358704577237341031692560