Dedicated Client Architecture in MapReduce and its Implications on Performance Considerations

Ragav Krishna.R
Carnegie Mellon University
Pittsburg, Pennsylvania,
USA

Sushma R
CMR Institute of Technology
Bangalore, Karnataka
India

ABSTRACT
Big data refers to a large quantity of data that has to be processed at one time. With the advancement of social media and the virtual world, a vast amount of data is created every second. A technique has to be designed to effectively process this ever-increasing collection of information. One such algorithm is the MapReduce algorithm. The result or output of the algorithm provides useful insights about the data used as input and can be further used for Decision Making and Prediction algorithms. Also, new data is generated frequently for Big Data processing. Hence, MapReduce implementations must not only be accurate but also as instantaneous as possible. This paper discusses not only the details of Map Reduce Algorithm; it also suggests architecture called Dedicated Client Architecture which would increase the efficiency of the algorithm.

General Terms
Big Data, Map Reduce, Client Server, dedicated client.

Keywords
Big Data, Map Reduce, Client Server Architecture, Dedicated Client

1. INTRODUCTION
The topic of Big data deals with all the data produced on a daily basis in the order of exa or peta bytes. This data needs to be processed to extract the useful information and should then be accurately classified. This document covers the implementation of MapReduce Algorithm to efficiently deal with the incoming flood of data. The shortcomings of MapReduce algorithm are discussed and enhancement in the operation to overcome these shortcomings is also stated.

2. BACKGROUND
There has been abundant research in the field of Big data and Big data analytics. Each of them may be concerned with how the data must be divided among the various reducing systems in order to maximize efficiency.

Network of Workstations (NOW) as explained in [1] has been shown to increase performance considerations substantially. Memory management here is provided by simple OS primitives. The paper uses a one-pass single-node sort mechanism with Read, Sort and Write as the primviple steps in the mechanism.

Combining DBMS and Search engine mechanism is a direct solution to face the increase in the amount of data to be searched.[2] Using BigTable is a solution opted by many Google applications like Google Earth and Google Finance and it provides flexible and high quality solutions for all of these products[3].

For cluster-based internet services, optimization is achieved by using Data Aggregation Call [4], the architecture of which has been designed in [4] to prevent any hardware/software failures. A more scalable operation is implemented in [5]

Gamma is an RDBMS quering system that has been evaluated in its initial stages in [6] and has produced promising results for the same. With the exponential growth in the amount of data each year, new scientific methods are required to analyse and process data. Hence, the implementation of Scientific Data management (SDM) is almost mandatory [7].

Another example of scientific programming model is shown in [8] which can interpret voluminous data sets using parallel analysis of BigData as used in Sawzall.

All in all, the domains in which BigData can be implemented is diverse [9] providing a wide scope for the improvement in the current standards of application and the discovery of more utilization of the present techniques.

3. MAPREDUCE
Big data is the term used when dealing with unreasonably large amounts of data (usually in the order of exabytes) which cannot be managed by the commonly used data processors. A definitive design strategy is required to deal with such enormous quantities of data. MapReduce is one such algorithmic schemes which allows us to process massive data efficiently in-parallel in a fault tolerant manner.

MapReduce consists two kinds of methods – Map and Reduce. Mapper or the Event Manager splits the incoming data into an intermediate form by mapping input <key/value> pairs for each data division. Each of these data packets will be executed in parallel. Not only this, the Event Manager is also responsible for scheduling, monitoring and (if required) re-executing the data packets. So at the end of this first step, the Event manager takes the input and divides it into a number of sub problems and distributes it to the various workers or the Task managers. The task manager may do this again if it results in higher efficiency. Each of the task managers executes the allocated data and returns the answer back to the Event Manager. In the next step, called Reduce, the Event Manager collects all the data from the workers and combines them into one an answer that is the solution to the original problem.

4. DRAWBACKS
- The event manager using the MapReduce algorithm is used to divide the incoming data and then distributing it to the various Task Managers in the system Also, after the execution of the different sub-problems; all results are returned to the Event Manager. The collected results are combined

together by the Event Manager to form the solution to the first problem.

- The Event Manager is in charge of the division, distribution, collection and combination of data whereas the Task Manager only executes the data given to it. Since the execution of the data is more or less constant in different systems, it should be noted that a small variation in any one of the Event manager's work will decrease the time taken and therefore increase the efficiency drastically.
- Another point to be noted is that while the event manager is performing its preliminary function of division of data, the Task manager is essentially without a job. It has to wait for an allotment of a task by the Event Manager. Similarly, after returning its sub-problem's results to the Event Manager, while the event manager is performing 'Reduce', the Task Managers are once again, idle.
- The MapReduce algorithm is specifically designed to work only when all the incoming data are collected. The EventManager then divides the input into sub problems. When this data is in the form of a stream, we do not know of the total size of data and hence the EventManager cannot predict the size of each packet.

5. PROBLEM STATEMENT

- To find the most optimal method to distribute incoming data between the event manager and the task manager using the implementation of the MapReduce Algorithm.
- To device a method to tackle a flowing stream of incoming data using the MapReduce Algorithm.

6. EXISTING SYSTEM

The ordered list of data in the semantic form of (key, value) pairs is predominantly used in the MapReduce functions.

Map Function: As we are aware, data exists in different domains and the function of the Map function is to take data in one domain and transform it to a list of pairs in another domain. Considering the quantum of data, the Map function is applied in parallel across all the pairs in the input set.

Reduce Function: The reduce function produces collection of value, one value for each pair which can be colloquially termed as sublet result. The returns of all calls are returned as desired result.

Input Reader: A stable storage stores data from which the input reader reads data. The input reader divides the data into appropriate blocks and subsequently generates key value pairs.

Partition Function: The partition function allocates the output of the map function to a respective reducer. Given the key and the number of reducers this function returns the index of the desired reducer. A has key – hash map function is the bottom line used for this.

Output Writer: The value output that is generated for each (key, value) pair is stored back in the stable storage using the output writer.

7. PROPOSED SYSTEM

The MapReduce process uses a single Master and multiple client architecture. This leads to several problems excessive dependence on the client, problems in dealing with streaming data and so on.

A simple solution to solve the problem is to reverse the physical location of the clients and Master. The streaming data comes into the client who is then transformed to the Master which processes the data and stores the result.

The sub problem that arises with this solutions are:

- The order in which the client pass the data to the Master is important and out of order data transfer can cause some problems in processing.
- The capacity of the client is important since it needs to accept a stream of data and pass it on to the client.
- A client that has transferred its data to the Master will be idle for the time during which data is streaming into another client.
- There needs to be a way to synchronize the computation of the results and storing of results in the Master.
- How to transfer the result back to the system that requested for service is also an important decision.

The above drawbacks are overcome when we use three different components – Master, Slaves and a Client. The following solutions using this method are dependent on the position of the Client in the system.

Solution 1: External dedicated Client

In this architecture, the Client is external. It does not interfere with the individual functions of the Slaves and the Master. Data that flows into the system is directed to the slaves and it is transported to each slave in a daisy chain fashion. This way, the order of the data packets does not change during the process thus preventing and confusion.

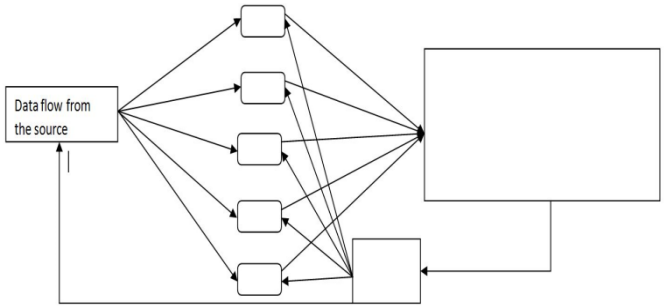

Fig 1: External Dedicated Client

The External client is in charge of monitoring the capacity of the packets and the amount of data. The Client continuously monitors the amount of data in a slave, and when it exceeds a limit it transfers some data to the next slave. This prevents any overloading and loss of data. The Client also ensures smooth, uninterrupted performances during the failure of any one of the slaves.

The result is then computed by the Master and returned to the source. The External Client used in this Solution can be used to perform various kinds of operations like error checking which do not reduce the efficiency of the system due to any overlapping of functions. Although this architecture is more optimal, it is a waste of time when the Client and Master are idle.

Solution 2: Internal Dedicated Client

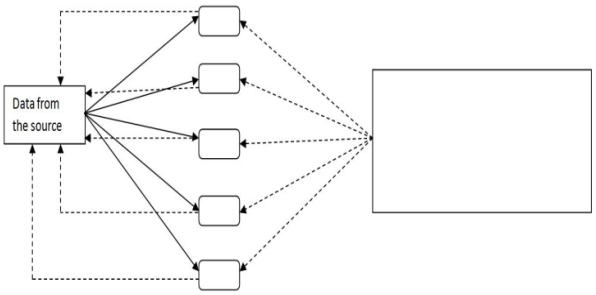

Fig 2: Internal Dedicated Client

This method is very similar to the previous one. The only difference is that in this method, the client is one of the slaves. Data flows from one of the slaves as in daisy chain fashion to all slaves. One of the slaves is chosen as the Client. This client is capable of performing all the functions that the Client in the previous solution was able to. The result is collected by the Master and returned to the source.

8. CONCLUSION

An enormous amount of data is generated across the globe every second. This data has to be processed such that the useful portions are extracted from the pool of disordered information. Algorithms such as MapReduce perform this task of segregation and obtaining useful material.

Our design of the MapReduce Master-Slave architecture has overcome most of the problems initially faced by the original MapReduce architecture. The drawbacks previously incurred include excessive dependence on the client, handling continuous flow of data, etc. We have introduced a concept of 'Client' in the master-slave architecture to facilitate the speed and accuracy of the Big Data processed.

Big data processing is the newest venture in speed and certainty algorithms, and it needs to process data as quickly as possible and at the same time produce accurate results. For this reason Big Data analytics is among the most prominent areas of research in the field of Computer Science.

9. FUTURE ENHANCEMENTS

MapReduce algorithm is the most prominent and efficient technique to deal with large quantities of data in the domain of Big Data analytics. We have devices two solutions to deal with a few drawbacks of the existing algorithm. Also we have suggested changes in the architecture like the inclusion of the 'Client' and the reversal of the client and the server. These alterations have helped the system to handle the vast amount of data more efficiently. Our first solution uses an external client to perform functions like error checking and other non-overlapping functions. This solution is not useful when both the client and the master are idle. In the second solution, one of the slaves itself is used as a Client. Both these solutions put a greater load on the system because of the introduction of client.

The problem of dealing with streamlined data is not yet fully solved. Although the client can load each slave till a particular threshold is reached, it has no way of knowing how large the incoming data is. This way, there is possibility of the client overloading one single slave and leaving all other slaves idle when it encounters a small quantity of data.

10. REFERENCES

[1] A. C. Arpaci-Dusseau et al. High-Performance Sorting on Networks of Workstations. In SIGMOD, pp. 243–254, 1997.

[2] E. A. Brewer. Combining Systems and Databases: A Search Engine Retrospective. In J. M. Hellerstein and M. Stonebraker, editors, Readings in Database Systems, Fourth Edition, Cambridge, MA, 2005. MIT Press.

[3] F. Chang et al. Bigtable: A Distributed Storage System for Structured Data. In OSDI, 2008, pp. 205–218.

[4] L. Chu et al. Optimizing Data Aggregation for Cluster-Based Internet Services. In PPOPP, pages 119–130. ACM, 2003.

[5] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In OSDI, pages 137–150, 2004.

[6] D. J. DeWitt et al. GAMMA - A High Performance Dataflow Database Machine. In VLDB 1986, pages 228–237, 1986.

[7] J. Gray et al. Scientific data management in the coming decade. SIGMOD Record, 34(4):34–41, 2005.

[8] R. Pike et al. Interpreting the Data: Parallel Analysis with Sawzall. Scientific Programming Journal

[9] Big Data, Bigger Opportunities, Author: Jean Yan, April 2013.