International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 05
April 2017

ANALYSIS OF BIG DATA PROCESSING BY DISTNCT USE OF HADOOP'S MAPREDUCE

I. Geervani[1], S. Kavya[2], K. Abdul Hannan[3], N. Venkatadri[4]

[1]B. Tech, CSE, TKREC, Hyderabad, Email:geervaniiji@gmail.com
[2]B. Tech, CSE, TKREC, Hyderabad, Email:kvsarikonda@gmail.com
[3]B. Tech, CSE, TKREC, Hyderabad, Email:habdul1995@gmail.com
[4]Professor, CSE Dept., TKREC, Hyderabad, TS-India, Email:nagala.venkat@gmail.com

ABSTRACT

Data has become an indispensable part of every economy, industry, organization, business function and individual and such datasets that are beyond the size that traditional databases can handle are termed as Big Data. Hence companies today use a tool called Hadoop. Even sufficiently large amount of data warehouses are unable to satisfy the needs of data storage. Hadoop is designed to store large amount of data sets reliably through HDFS and MapReduce for storing and processing respectively. It is an open source software which supports parallel and distributed data processing. Hadoop also provide fault tolerance mechanism by replication. In this paper, we present introduction to HDFS and MapReduce and survey the performance of sufficiently large dataset processing using MapReduce technique. We propose that performance of the dataset processing can be optimized by leveraging MapReduce in different ways. We analysed the performance of datasets by varying the approach to process it.

Keywords: Big Data, Data Warehouses, Replication, HDFS, MapReduce.

1. INTRODUCTION

The biggest problem with true big data (massive, less structured, heterogeneous, unwieldy data up to, including and beyond the petabyte range) is that it's incomprehensible to humans at scale. [1].We can't get machines to help us enough. And yet big data keeps getting bigger. So we're drowning in our own data. The rise of ubiquitous computing and more and more endpoints communicating in their own feedback loops with the cloud keeps data growth going at double digit rates. We can't keep up with it. The exponential growth of data first presented challenges to cutting-edge businesses such as Google, Yahoo, Amazon, Microsoft, Facebook, Twitter etc. [2]. Data volumes to be processed by cloud applications are growing much faster than computing power. This growth demands new strategies for processing and analysing information. Google over a decade ago developed a way that Yahoo cloned to spread data out across huge commodity clusters and process simple batch jobs to begin to mine big datasets on an ad-hoc batch basis cost effectively. That method has evolved as Hadoop. Map reduce is a software frame work introduced by Google in 2004 to support distributed computing on large data sets on clusters of computers.

The original MapReduce implementation by Google, as well as its open-source counterpart, Hadoop, is aimed for parallelizing computing in large clusters of commodity machines. Map-Reduce is a programming model that is used to

analysis the big data in cloud environment and used to retrieve the data from the Hadoop cluster. In this model, processing of large data is efficient, easy to use, it splits the tasks and executes on the various nodes in parallel. Thus it will speed up the computation and retrieve the required data from a huge data set in a faster manner. We introduce the map-reduce programming model [6] for analysis the big data in efficient manner using Hadoop.

Fig 1. Survey on increase in the data generated per year.

It provides a well-organized data analysis, performance analysis and executes process in parallel distributed manner. By using this programming model, Performance of the system is increased, highly fault tolerant and scalable(HDFS), [4] The Hadoop Distributed File System is a specialized file system to store large amounts of data across a distributed system of computers with very high throughput and multiple replications on a cluster. This MapReduce Originally conceived by Google as a way of handling the enormous amount of data produced by their search bots, it has been adapted in a way that it can run on a cluster of normal commodity machines. This is open source and distributed by Apache Hadoop.

2. EXISTING SYSTEM

Grid Resource Allocation Manager [11] (or GRAM) is a software component that can locate, submit, monitor, and cancel jobs on computing resources. It provides reliable operation, stateful monitoring, credential management, and file staging. GRAM does not provide job scheduler functionality and is in fact just a front-end to the functionality provided by an external scheduler. The jobs submitted to GRAM are targeted at a single computation resource, and consist of an optional input file staging phase, job execution, and an optional output file staging and clean-up stage. Works well for predominantly compute intensive jobs, but it becomes a problem when nodes need to access larger data volumes (hundreds of gigabytes), since the network bandwidth is the bottleneck and compute nodes become idle. It explicitly manages its own check pointing and recovery of tasks which is a time taking task. Big Data is known as to be extremely large datasets that are hard to deal with using operational databases. It is required for parallel processing on of data on hundreds of machines.

3. PROPOSED SYSTEM

MapReduce is a data processing or parallel programming model introduced by Google. In this model, a user specifies the computation by two functions, Map and Reduce. In the mapping phase, MapReduce takes the input data and feeds each data element to the mapper. In the

reducing phase, the reducer processes all the outputs from the mapper and arrives at a final result. In simple terms, the mapper is meant to filter and transform the input into something that the reducer can aggregate over. The underlying MapReduce library automatically parallelizes the computation, and handles complicated issues like data distribution, load balancing and fault tolerance.

It employs a master/slave architecture, where the master is in charge of management and Scheduling [4], and the slaves are responsible for data storage and task processing. Indeed, Hadoop provides some fault tolerance mechanisms through both HDFS and MapReduce. First, HDFS provides storage layer of fault tolerance by replication. That is, HDFS keeps multiple replicas of each data block in several different nodes, so that if any one node is down, data could still be restored from other replicas. Second, Hadoop MapReduce provides job level fault tolerance [5]. That is, if a Map or Reduce task fails, the scheduler would re -assign the task to another node. Here we propose an approach to increase the performance of processing large datasets by efficient usage of MapReduce technique and generating multiple yields in a single-go.

4. ARCHITECTURE

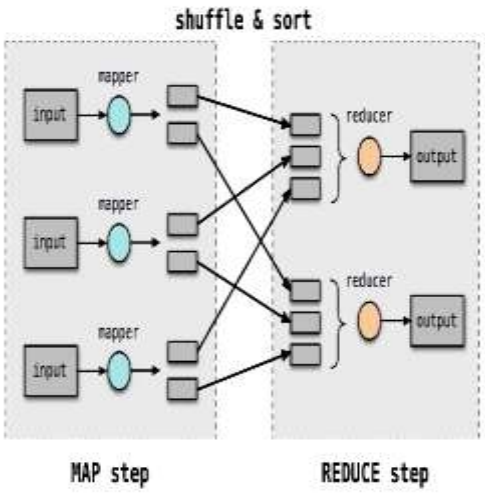

Fig 4. MapReduce architecture

MapReduce mainly composed of two phases Map and Reduce. The MapReduce is a master slave architecture as in HDFS. There are two types of nodes Task-tracker and JobTracker. TaskTracker act as master node and JobTracker as slave. Map, written by the user, takes an input pair and produces a set of intermediate key/value pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key is and passes them to the reduce function. The reduce function, also written by the user, accepts an intermediate key I and a set of values for that key. It merges these values together to form a possibly smaller set of values.

Typically just zero or one output value is produced per reduce invocation. But through leveraging the MapReduce in different ways via mapper and reducer, multiple yields can be obtained which increases the performance by optimizing the execution time of the dataset. This is achieved by altering the mapper and reducer functionality.

5. ALGORITHM

The algorithm used to achieve the desired multiple yields is MapReduce algorithm. For this it works as follows on the given dataset.

MapReduce will first divide the data into N partitions i.e., blocks. Then it will start many programs on a cluster. One of program is the master program; the others are workers, which can execute their work assigned by master.

Map stage: Each worker node applies the *map*() function to the local data, and writes the output to a temporary storage. A master node ensures that only one copy of redundant input data is processed.

Shuffle/Sort stage: Worker nodes redistribute data based on the output keys (produced by the *map*() function), such that all data belonging to one key is located on the same worker node.

Reduce stage: Worker nodes now process each group of output data, per key, in parallel and reduced outputs are segregated based on the given cluster references and outputs are stored in the HDFS in the file format.

6. MODULES

The modules that are considered to have much significance in this are as follows.

6.1 Storage

Hadoop's storage component, Hadoop Distributed File System will perform the key storage activities. And HDFS is High reliable and fault tolerant architecture. The data which is inserted is either in any form of data like structured, semi structured and unstructured data. Here we are concentrating all the data in any type with any constrains. HDFS stores large files (typically in the range of gigabytes to terabytes) across multiple machines. It achieves reliability by replicating the data across multiple hosts. With the default replication value, 3, data is stored on three nodes: two on the same rack, and one on a different rack. Data nodes can talk to each other to rebalance data, to move copies around, and to keep the replication of data high.

6.2 Processing

Hadoop is a batch processing framework and data to be processed are stored in the HDFS, a powerful tool designed to manage large datasets with high fault-tolerance. *MapReduce*, the heart of Hadoop, is a programming model that allows processing a substantial amount of data in parallel. An example of the MapReduce model has three major processing phases: *Map*, *Shuffle*, and *Reduce*. Traditional relational database organizes data into rows and columns and stores the data in tables. MapReduce uses a different way, it uses key/value pairs.

7. RESULT ANALYSIS

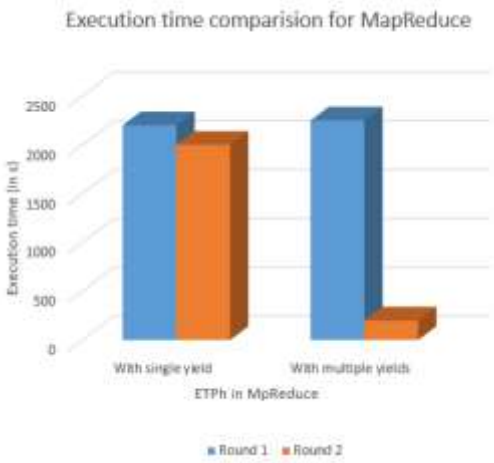

Fig 7. Execution time comparisons for MapReduce stage

Here we compared the execution times for a large dataset which has single or zero output for each execution round and the one that yields multiple outputs for the first execution round. When the dataset size is increased the execution time increases with every execution round and thus yielding multiple desired outputs through Map and Reduce tasks through first pass saves the re-execution time and searching for the output from the HDFS will only be the task rather than executing it all over again.

8. CONCLUSION

Optimizing the performance is a challenging problem for Hadoop/ MapReduce workloads because of the large size of the datasets. In this paper, we propose a way to utilize the application specific parameters that are generated by the reduce phase in the first pass. We evaluate the approach using a large dataset and varying the traditional to leveraged approach and compare the results. The results show that our approach speeds up Hadoop programs significantly due to the concept of generating multiple output files through an efficient mapper and reducer functionality.

9. FUTURE ENHANCEMENTS

MapReduce and HDFS together with our approach had shown significant speedups in the execution time of the Hadoop programs. Future work can be done on optimizing the user written part of the mapper and reducer functionality. The main issue that needs further focus is lack of flexible resource management and multiple data source support. Proper skill training is also needed for achieving large scale data analysis. These challenges needs to be furthermore focused to achieve full potential Hadoop data management.

10. REFERENCES

[1] R. Taylor. An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics BMC bioinformatics,11(Suppl 12):S1, 2010.

[2] A. Pavlo et al . A comparison of approaches to large-scale data analysis. In Proceedings of the ACM SIGMOD, pages 165178, 2009.

[3] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility, Future Generation Computer Systems 25 (2009) 599616.

[4] Hadoop Distributed File Systemhttp://hadoop.apache.org/hdfs

[5] W. Jiang et al . A Map-Reduce System with an Alternate API for Multi-core Environments. In Proceedings of the 10th IEEE/ACM CCGrid, pages 8493, 2010.

[6] Map-Reduce: Simplied Data Processing on LargeClusters, by Jerey Dean and SanjayGhemawat; fromGoogle Research

[7] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello, "Sindice.com: A document oriented lookup index for open linked data," Int. J. Metadata Semantics Ontologies, vol. 3, pp. 37–52, Nov. 2008.

[8] L. Lie, "Heuristic artificial intelligent algorithm for genetic algorithm," Key Eng. Materials, vol. 439, pp. 516–521, 2010.

[9] K. S. Beyer, V. Ercegovac, R. Gemulla, A. Balmin, M. Eltabakh, C.-C. Kanne, F. Ozcan, and E. J. Shekita, "Jaql: A scripting language for large scale semistructured data analysis," in Proc. VLDB Conf., Sep. 2011, pp. 1272–1283.

[10] H. Herodotou, "Hadoop performance models," Duke Univ., Durham, NC, USA, Tech. Rep. CS-2011-05, 2011.

[11] GRAM: Grid Resource allocation manager, http://www.javamakeuse.com/2016/04/hadoop-vs-grid-computing.html

[12] Jeffrey Dean and Sanjay Ghemawat: "MapReduce: Simplied Data Processing on Large Clusters".