Volume 5 Issue 2 December 2017
ISSN: 2321-6840

Data Skew Handling in Heterogeneous Hadoop Cluster

Abhash Visoriya[1], Deepak Barade[2] and Sunita Varma[3]
[1]M.E. Scholar, Department of Computer Engineering, SGSITS Indore, Madhya Pradesh, India.
Email: abhasvisoriya@gmail.com
[2]M.E. Scholar, Department of Computer Engineering, SGSITS Indore, Madhya Pradesh, India.
Email: barade.deepak@gmail.com
[3]Head, Department of Information Technology, SGSITS Indore, Madhya Pradesh, India.
Email: sunita.varma19@gmail.com

Abstract: Map reduce has been accepted as an important distributed processing model for processing big data which is generated by data intensive applications. Hadoop is an open source implementation which uses map reduce as programming model for processing big data. There are various programming tools for processing data but most of them are not suitable for processing big data.

In the current hadoop implementation it is assumed that all nodes are homogeneous in nature. Homogeneous means all nodes have same computation power. But in the practical scenario it is possible to have heterogeneous environment. We can improve the performance of map reduce by considering heterogeneous environment.

The second issue while processing the data with MapReduce framework is data skew. Uneven distribution of the data to each task is called data skew. so when data skew arises in system, then the tasks with skewed data take much longer time to complete compare than other tasks, this leads to performance degradation of the overall system.

In this paper we will focus on how data should be placed across all the nodes and how to process the data by taking into consideration the data skew problem so that we will get maximum performance from given resources. Our data handling strategy distributes and processes the data in such a way that we get the maximum performance from each node which in turn increases the overall performance of map reduce.

Keywords: Data Skew, HDFS, Hadoop, MapReduce, Heterogeneous

I. Introduction

As the technology is growing day by day, data is becoming the most important thing. With the increasing number of internet users, size of the data is also increasing exponentially. Since the world wide web came into the picture lots of data intensive application have been developed. There are lot of web applications like e-mail, online auction, social networking etc. are available to the users. Applications like Data mining and web indexing needs to access the data from few giga byte to peta bytes. For example facebook generates 500 tera bytes per day so there is a need of efficient programming model to process big data. Map reduce provides a parallel computation environment for processing big data with the help of high performance clusters. Map reduce is proven to be highly scalable because when we submit a job to map reduce then it splits into many small jobs which runs on multiple machine in hadoop cluster. Hadoop was basically developed by yahoo for processing large data sets. It is an open source implementation of Googles MapReduce model [1]. There are various servers like Yahoo, facebook, Amazon etc., which use hadoop framework for processing vast amount of data. For example, face-book uses hadoop to process hundreds of terabytes of data on daily basis [2]. Hadoop is not limited to web applications only, rather various scientific and research applications are taking maximum benefits from it.

Mapreduce is a parallel programming computation model for distributed data sets. A map reduce programmer need not to have knowledge about distributed computing. Programmer can write their own map and reduce functions depending on the application requirement without any specific knowledge of distributed computing. The mapreduce framework simplifies complexity of executing distributed data processing functions among multiple nodes in a cluster. After performing map and reduce tasks on various nodes, the mapreduce framework automatically collects results from multiple nodes, join them to single result and returns the result to master node. Fault tolerance and transparency to programmers are salient features of mapreduce [3].

To process the data sets, data should be local to each machine. So it is observed that data locality is the most important factor for mapreduce performance. In the Hadoop cluster

data is distributed according to the free secondary storage to each machine. In various observations it is seen that this data placement strategy is good when all nodes in the cluster have same computation power. In other words we can say that if nodes are homogeneous then this data placement strategy is good. There is no need to move data from one machine to another machine because all nodes will finish there work in almost same time duration. but if we have some high performance nodes in our cluster then there is a need of data movement from one node to another node. If the size of data to be transferred is huge then this is overhead which is undesirable. An approach to improve the performance of map reduce framework is to reduce the data transfer overheads due to heterogeneous environment. Basically our aim is to develop a mechanism which should balance the load to all the nodes with respect to their performance so that no node become idle and there will be no need to transfer data from slower nodes to faster nodes.

After placement of the data, the job in MapReduce is divided into multiple small tasks, which are assigned to multiple nodes in the cluster So, finish time of any job depends on the task which takes more time. Performance of the job degrades significantly if a task consumes much longer time for its completion compared to other tasks. Slowest running task in the job is called straggler. There are various reasons for straggler to occur in the system. Data Skew is the most common among them. Uneven distribution of the data to each task is called data skew. As we dont know the distribution of data prior to the execution and real world data is more often skewed.

In this study our aim is to develop a data distribution mechanism which is based on performance of the nodes and developing a cluster splitting strategy applied on clusters causing data skew. in Hadoop cluster data distribution is the work of HDFS so we need to make some changes in HDFS block placement policy and developing an algorithm to split skewed cluster after map phase.

II. Background

A. MapReduce

There are various programming tool to process data but most of them are not suitable for big data. Map reduce is an efficient framework for processing large data sets which are generated by data intensive applications. Map reduce has two integral parts Map and Reduce. Map function is directly applied to input data and generates ¡key, value¿ pairs then reduce function is applied on intermediate data. A user can write his own map and reduce functions according to the application requirement. A programmer just need to focus on programming function rather than parallel functioning.

B. HDFS

Hadoop uses HDFS (Hadoop distributed file system) for storing the data across all the nodes for processing. So, we can say that hadoop uses distributed and parallel computing both, data is distributed to all nodes and computing is parallel with the help of mapreduce framework. In a typical hadoop cluster there is a master node and many slave nodes, master node contains every information in the cluster. It is the responsibility of master node to assign the task to the slave nodes.

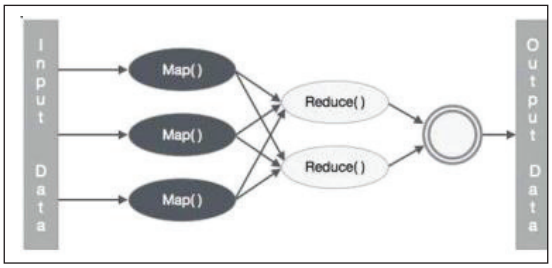


Fig. 1: MapReduce Programming Model

When an input is given to the hadoop system then it splits it into the equal chunks or fragments then these fragments are stored across all nodes. Hadoop is highly fault tolerant for this purpose it make replicas of each fragments. HDFS is a file system which is very similar to Google File System or GFS [4]. HDFS is very fault tolerant and to make it realize, it makes three copies of each fragment and stores them separately on three different nodes of the cluster. Thus, Hadoop by de-fault uses three way replication but this replication factor can be customized.

C. Skewed Data in MapReduce

To increase system performance, all the tasks must be completed approximately in the same time duration. If a task takes longer time to finish com-pare to other tasks, then it is called a straggler and it can significantly degrade system performance. Stragglers can arise due to some other factors such as failure of hardware resources, low performance systems, etc. Speculative execution is a technique to counter the effects of stragglers, in which the slow tasks are made to run on some other high performance system with the expectation that it can complete its execution here faster. It has been found that the task completion time with the help of speculative execution has been reduced significantly [5]. But, when data skew causes straggler to happen, then simply replicating the task on other system cannot solve the skew problem. Skewed data can happen in both the map and the reduce phases. Skew in the map phase happens when data in the input are very tough to process, but it rarely happens and this skew in map phase can be solved by split of map tasks. Lin [6] provided a solution which is bound to a particular application only. It splits large records into smaller records. While, data skew after map phase is harder to solve. The MapReduce requires that all the records having the identical key ought to be sent to the same reducer. However, any application employing MapReduce, the intermediate data

distribution cannot be known in advance. Many Applications in the real world produce huge skewed data, for example aggregation, group and join operations in distributed database [7], [8] applications like search engines which issued to find Inverted Index, Page Rank, & scientific applications [9] etc.

III. Custom Data Placement and Skew Handling

A. Data Placement in Hadoop Cluster

Whenever data is to be processed, it must be available to local machine. If data is not locally available to machine then we have to transfer huge data from one machine to another machine with the help of interconnecting networks. This can lead to huge network congestion, which in turn deteriorate overall system performance.

In a hadoop cluster there is a possibility that different nodes have different computation power. So, it is possible that faster nodes may finish their work earlier than slower nodes, then there are two possibilities either the node remain idle or transfer unprocessed data from slower nodes to these idle nodes. But, in practical scenario transferring huge amount of data, itself aids to system performance degradation. Hence we aim to reduce this data transfer among nodes in a cluster. To achieve this we need a data placement strategy that distributes and stores data across multiple nodes according to their computational capacities.

To gain the best I/O performance, local copy of the input data can be made available on each node by making replicas of input data, but, data replication itself has several drawbacks First making replicas of huge amount of data is itself a overhead to system. Second, transferring these replicas to nodes wastes lot of network bandwidth unnecessarily. Third, storing such huge replicas requires vast amount of disk space, which may increase the cost of hadoop cluster.

To overcome the limitation of data replication approach, our focus is to develop a data placement mechanism where file are partitioned and distributed across all the nodes in hadoop cluster. Our data placement mechanism does not require any data movement and data replicas overhead.

B. Custom Block Placement

Our data placement mechanism is divided into two halves. In the first half of the mechanism, we will measure the heterogeneity of hadoop cluster in terms of processing power. For this purpose we will take the data from user for processing. Now we will take random fragment of data as sample data. This sample data will be distributed to all nodes in cluster for processing. For comparing the processing power fairly we ensure that all nodes must process the same amount of data. After processing the data we will record the response time of each node. Now we will prepare a table, called as Processing Ratio Table as shown below:

Table I: Processing Ratio Table

Node	Elapsed Time allocated	Processing Ratio	File fragments Speed to be

According to fastest node or the node which takes less time to process the sample data, we will normalize all the response time measurements.

These normalized values are called processing ratio. In the next half, data fragments will be distributed according to this processing ratio. For example, suppose we have three nodes I, II and with different processing powers. Now we will distribute around 1GB sample data to each of these three nodes. After distributing the sample processing will start at each node. After processing sample data we will store the response time in the processing table.

For example the response times of nodes I, II and III are 10, 20, 30 Seconds respectively. As the response time of the node I is less among all. Therefore, the processing ratio for this node is set to 1. It is used as reference to find processing ratio of nodes II and III. Thus, the processing ratio of nodes II and III are 2 and 3 respectively. Table II shows the response times and processing ratios for each node in a hadoop cluster. Table II also shows the number of the fragments to be allocated. Suppose the file is divided in 60 fragments. So, the fastest processing node I will be given 30 fragments while the nodes II and III will get 20 and 10 fragments respectively. Table II is shown below:

Table II: Response Times, Processing Ratio and Number of Fragments to be Allocated Among Three Nodes in Cluster

Node	Elapsed Time	Processing Ratio	File fragments to be allocated	Speed
I	10	1	30	Fastest
II	20	2	20	Average
III	30	3	10	Slowest

Now, we can conclude that if above data place-ment mechanism is employed then the performance in heterogeneous clusters can be improved significantly.

C. Splitting Clusters Causing Skew

The MapReduce programming model needs that for any cluster i.e. the group of records having the same key must be processed by a same reducer. Splitting clusters can handle data skew very effectively. If split of the cluster is not employed, whole cluster will be allotted to a particular reducer. If intermediate data distribution has many keys having very large number of records than other keys, then even the best skew handling strategy wont

perform well. For example, assume the intermediate data have 3 keys namely X, Y & Z and these keys have 100, 10 & 10 records respectively. This intermediate data is to be processed by 2 reducers. When splitting of cluster is not employed, then best strategy for key distribution is shown in below figure. Reducer r1 gets key x and the other reducer r2 gets keys y & z. Figure shows it still shows data skew.

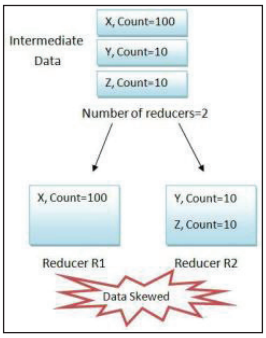


Fig. 2: Data Skew Problem

whilst [6] and [24] show special techniques for splitting large clusters in the cloud burst and join applications but they are specific to some particular applications only and aids extra sampling cost which is non-negligible. As far as our knowledge is concerned, no existing work talks about a generalized method for splitting large cluster. We have implemented a cluster split technique which splits large clusters in such a manner that skew can be handled to some good extent. Using this technique, the skew problem as shown above in fig 4 is handled by allocating 60 records of the larger key X to reducer r1 and the remaining keys to reducer r2 as shown in below figure. Using this technique, the data skew can be handled.

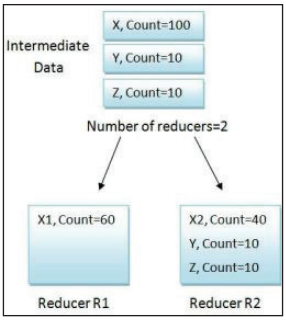
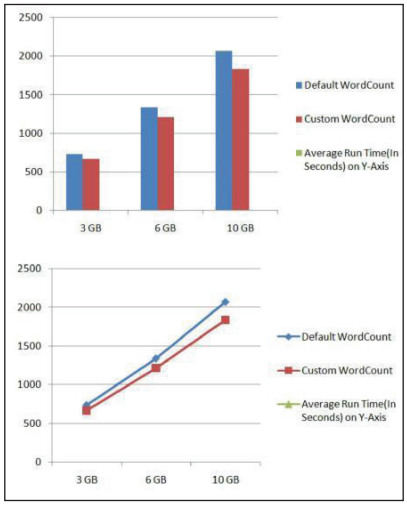


Fig. 3: Data Skew Handling

IV. Evaluation

A. Experimental Evaluation

We have set up a heterogeneous hadoop cluster of four nodes. We have used WordCount application to evaluate the performance of our data placement policy in a heterogeneous Hadoop cluster. This cluster is comprised of 3 heterogeneous data nodes and a name node. The configurations of all 3 data nodes are different in all respects. Each having different processing power, RAM and hard disks. For heterogeneity we have used sampling technique. We have taken sample data in it is run multiple times to for getting the performance of each node. After getting the responses the ranks have been assigned to each node. According to ranks allotted to nodes, data of different sizes has been placed on the cluster. Now wordcount application is run and total time is recorded. Now wordcount is run using skew handling technique and total time has been recorded. After this cluster is Stopped and formatted. Now the cluster is restarted and data is placed according to default data placement technique. Now wordcount application is run on this cluster and total time is recorded. The various results are shown in the following graphs. These graphs shows that a significant improvement in system performance.



V. Related Work

Lot of researches has been done on perfor-mance evaluation and implementation of MapRe-duce Model [11][12][8][10]. For example, Ranger did implementation of mapreduce for shared mem-ory systems [12]. Bingsheng et al. Developed Mars- a map reduce framework for graphics processors [11]. The main feature of Mars is to abstract programming complexity of GPUs. Zaharia et al. Developed a scheduler called- LATE in Hadoop with the help of speculative execution of jobs [13].

Skew in data has also been observed in the MapReduce applications recently. Okcan et al. propose a skew optimization for the theta join by adding two pre-run sampling and counting jobs. Kwon et al. provide a system called SkewReduce which optimize the data partition for the spatial feature extraction application by operating prepro-cessing extracting and sampling procedures [12]. Although these solutions can handle data skew to some extent, they have significant overhead due to the pre-run jobs and are applicable only to certain publications. Although the above works can improve the MapReduce performance in heterogeneous environment, but they did not consider data locality and data transfer overheads.

VI. Conclusion

With the above study we can conclude that there is performance degradation in heterogeneous hadoop clusters. So, by observing this performance degradation caused by heterogeneity, we have de-signed a data placement mechanism in HDFS. This mechanism is purely based on nodes processing powers. Our approach will improve performance of heterogeneous hadoop clusters significantly.

Handling skew is vital in for MapReduce performance improvement. It has introduced a technique that handles skew in an existing MapReduce system. The salient characteristic of this approach is that it can split a large cluster and it is also aware with heterogeneity of the cluster. When we evaluated performance of the system then it shows that the improvement in performance is significant and the Overheads are negligible.

References

[1] http://lucene.apache.org/hadoop.

[2] R. Pike, S. Dorward, R. Griesemer, and S. Quinlan, "Interpreting the data: Parallel analysis with Sawzall," vol. 13. IOS Press, 2005.

[3] J. Dean, and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *OSDI '04: 6th Symposium on Operating Systems Design and Implementation,* pp. 137-150, 2008.

[4] S. Ghemawat, H. Gobioff, and S. Leung, "The google file sys-tem," *SIGOPS Oper. Syst. Rev.,* vol. 37, no. 5, p. 2943, 2003.

[5] J. Dean, and S. Ghemawat, "Mapreduce: Simplified data pro-cessing on large clusters," *Commun. ACM,* vol. 51, Jan. 2008.

[6] L. Jimmy, "The curse of zipf and limits to parallelization: A look at the stragglers problem in mapreduce," In *7th Workshop on Large-Scale Distributed Systems for Information Retrieval (LSDS-IR),* 2009.

[7] C. B. Walton, A. G. Dale, and R. M. Jenevein, "A taxonomy and performance model of data skew effects in parallel joins," In *Proc. of the International Conference on Very Large Data Bases (VLDB),* 1991.

[8] D. J. DeWitt, J. F. Naughton, D. A. Schneider, and S. Seshadri, "Practical skew handling in parallel joins," In *Proc. of the International Conference on Very Large DataBases (VLDB),* 1992.

[9] R. P. Mount, "The office of science data-management chal-lenge," Department of Energy, *Tech. Rep.,* 2004.

[10] M. C. Schatz, "Cloudburst: Highly sensitive read mapping with mapreduce," *Bioinformatics,* vol. 25, no. 11, 2009.

[11] B. He, W. Fang, Q. Luo, N. Govindaraju, and T. Wang, "Mars: A MapReduce framework on graphics processors," *ACM,* 2008.

[12] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis, "Evaluating mapreduce for multi-core and multi-processor systems," *High-Performance Computer Architecture, International Symposium on, 0:1324,* 2007.

[13] M. Zaharia, A. Konwinski, A. Joseph, Y. Zatz, and I.Stoica. Improving mapreduce performance in heterogeneous environ-ments. *In OSDI08: 8th USENIX Symposium on Operating Systems Design and Implementation,* October 2008.