Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 5.258

IJCSMC, Vol. 5, Issue. 7, July 2016, pg.290 – 296

A Survey on Data Skew Mitigating Techniques in Hadoop MapReduce Framework

S.Dhanalakshmi[1], B.Arputhamary[2], Dr. L. Arockiam[3]

[1]Research Scholar, Department of Computer Science, Bishop Heber College Tiruchirappalli, Tamilnadu, India
[2]Assistant Professor, Department of Computer Applications, Bishop Heber College Tiruchirappalli, Tamilnadu, India
[3]Associate Professor, Department of Computer Science, St. Joseph's College Tiruchirappalli, Tamilnadu, India
[1] dhanas.gopal@gmail.com, [2] arputhambaskaran@rediffmail.com, [3] larockiam@gmail.com

Abstract – In the epoch of Big Data, it produces huge volume of structured and unstructured data. Big Data termed as 5V's – high level of Volume, Variety, Velocity, Value and Veracity. It is difficult to process the data and it requires large number of systems to work parallel. MapReduce is a parallel scale programming model or processing large data in parallel. It is the spirit of the Apache's Hadoop. Hadoop is an open source execution of Google's MapReduce agenda. Mainly it has two components. Hadoop Distributed File System (HDFS) for storage part and MapReduce for processing part. In MapReduce, the job completion time is based on the slowest running task. This type of task is known as straggler. This paper reviews the types of data skew and several skew mitigating techniques to solve the issue.

Keywords – Big Data, MapReduce, Hadoop, HDFS, Straggler, Data skew.

I. INTRODUCTION

In recent years Big Data has become a buzzword [5]. The network bandwidth increases as huge volumes of data are produced by various companies, researchers and governments. Data can come from anywhere either structured or unstructured as sales statistics, financial balances, cell phone signals, business deals, and climate data which are embedded in electronic devices as Global Positioning System (GPS) devices by sensors, consumer feedbacks and comments from various websites, posting a text, image, audio and video on social networking websites [13]. To access the numerous popular applications larger data sets can be used and the data scale is ranging from bits to Yotta bytes [5]. The amount of data generated to process in a single machine will be time consuming. The data itself may be too large to store in a single system. For the issue of time complexity, the data can be processed initially and then stored in separate storage space and the work cargo can be distributed among more than one computer. Google introduced the Google File System (GFS), a disseminated file system model for huge range data processing with MapReduce programming model [2]. With the help of the MapReduce model 20 Peta bytes of data are processed by Google per day.

In MapReduce, the performance and scalability can be increased because the huge data set can be split into many smaller partitions and the job is partitioned into many smaller tasks. These tasks run on the multiple different nodes in a cluster [3]. Various companies like IBM, Facebook, LinkedIn, Twitter and EBay used the Hadoop to handle huge quantity of records [2]. To balance the cargo, Hadoop distributes the data based on available disk spaces to the nodes in the cluster. In homogenous backdrop, the computing and disk capability are similar for all the nodes. In heterogeneous background, the nodes are not similar and may differ in the computing and disk capability. So, there is a chance that the tasks of high performance nodes may take less time

to complete than other tasks. These nodes can process the data faster than the low performance nodes. This causes the skewness of data hence the whole job gets delayed by the slowest processing task which delays the execution of whole job which is known as the straggler [3]. In heterogeneous background, the speculative execution strategy can be used to resolve the straggler nodes issues [11]. The main issues are: data partitioning, skew computing, low degree of parallelism, scheduling delay, load balancing and poor system utilization [6]. Over the past years, many researchers have proposed their ideas for identifying these issues.

In summary, this paper presents the following contributions: Lists the various types of data skew and presents series of techniques to mitigate the issues of data skew. The rest of the paper is organized as follows. Section II presents the overview of Hadoop. Section III presents some background information about MapReduce. Section IV contains the different types of data skew. Section V introduces the different techniques for mitigating the data skew issues. Section VI presents the related works for this study. Section VII concludes this paper and discusses future work.

II. HADOOP

Hadoop is a free java based programming structure which supports the larger data sets in a distributed computing environment. It provides distributed, scalable and portable storage Hadoop Distributed File System (HDFS) and computation (MapReduce) across cluster of computers. It is a highly reliable hardware and it is not expensive [3]. HDFS uses a Master / Slave design. The NameNode or MasterNode does not store any actual data but it handles the meta data about the file system index such as data is stored on which data nodes, active nodes, passive nodes, job tracker and task tracker and other configuration files such as replication of the data [11]. The NameNode is the center piece of the HDFS and it is also considered as the single point of failure [3]. The DataNode or SlaveNode is a node which stores the actual data. It is used to track the on-going jobs which are generated from NameNode [11]. At the same time it has a drawback that it is a single point of failure; when it fails the entire system is also getting failed. The secondary NameNode has been introduced in the updated versions of Hadoop on the separate machine [3]. The NameNode defines the mapping of blocks to the DataNodes.

The DataNode can perform read and write request from clients. It also creates, deletes, moves and replicates the blocks based on the instructions given by NameNode. NameNode regularly receives heartbeats of message and blocks report from all the DataNodes check whether all DataNodes are working properly. In case of a DataNode failure, the NameNode chooses a new DataNode and manages the communication traffic.

MasterNode, called the JobTracker, takes charge of: i. querying the NameNode for the block locations, ii. NameNode retrieves the data, JobTracker schedule the tasks on the slaves called TaskTrackers, and iii. Scrutinize the success and failures of the tasks [10]. Thus the TaskTracker performs the tasks of the Map and Reduce phase [3].

III. MAPREDUCE BACKGROUND

MapReduce is a parallel data processing programming model for distributed computers. The MapReduce programs can be written in various languages like C++, ruby, python and java are run by the Hadoop. All these MapReduce programmes process huge data sets in a parallel manner and these are data intensive in nature [3].

A. Working Process of MapReduce

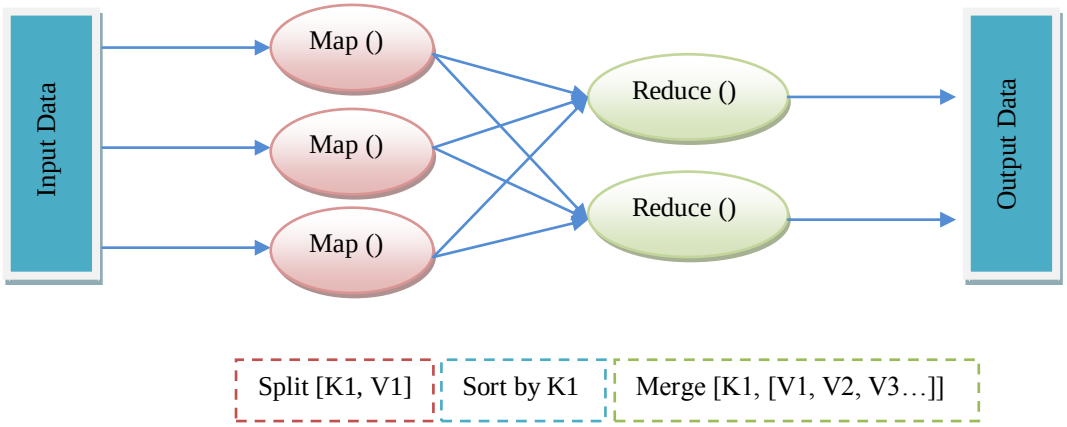

Fig. 1 MapReduce Execution Overview

The MapReduce system is divided into two parts: Map phase and Reduce phase. The Map phase is the first phase which takes the input as raw data such as text file [3]. It divides and distributes the work among the worker nodes. When the distribution is completed, the Reduce phase combines the output and reduces it to make a complete system. Map phase takes the unstructured data and rearrange as a new layout that will match the final outcome [5]. The MasterNode will indicate the SlaveNode for a reply. If the SlaveNode is occupied, then the MasterNode measures it as crash. The MasterNode then assign the job to another SlaveNode. Every Map and Reduce phase is self-sufficient of each other. The processing can be done on the individual nodes where the data is located. The input data from each Map phase is provisionally stored where the Reducers can come in to collect the output [5].

The result of the Map phase is used as the input for the Reduce phase. To rearrange this format Map phase is split into two steps. First, assign a key K1 to sort all the data and a processor using set of data with those keys. It creates the list based on another key K2 for all the pairs of the same key. The function is viewed as Map (Key 1, Value 1) -> List (Key 2, Value 2). The input reader gathers the data from the source and produces the key/value pairs for the input of the Map phase. An output writer then writes the output from the Reduce phase return back to the source.

B. Stragglers in MapReduce

Due to the distribution of imbalance workload, some of the machines may be idle until others complete their task. The slowest processing task delays the execution of whole job and it is known as the 'straggler' [12]. The speculative execution method is used to overcome the straggler caused by the external factors. If there is any faulty hardware in the machine or if a machine fails which leads to performance issue. The issue of straggler can be overcome by shifting the workload to other machine which is having the better performance. Moreover, the speculative execution method could not be used to overcome the straggler caused by the internal factors like physical properties of the data as altitude and mass of human being. The data skew causes this straggler. Skew can occur in both the Map and Reduce phase. The data skew on the Map phase is rarely observed and it is easily mitigated by splitting the Map tasks. Whereas the data skew on the Reduce side is very complex and a demanding issue [3].

IV. DATA SKEW TYPES

MapReduce is an effective tool for parallel data processing. The significant issue in practical MapReduce application is the data skew. The data skew refers to the imbalance in the amount of data assigned to each and every task or the imbalance in the amount of work needed to process the data. In MapReduce, skew happens when one node has more data assigned to be processed than others either in the Map or Reduce phase [10].

A. Mapper Side

Following are some of the reasons for the data skew to occur at mapper side [3], [7], [10], [14]:

a. Data Size Variation

The input size of the data gets varied when data distribution takes place in the Mapper task. Thus the Mapper task is entirely performed based on the CPU oriented algorithms.

b. Slow Performance of CPU

The Mapper task receives the data and change it as (key/value) pairs. Equal amount of data is fed into the each task and does not worry about the amount of time to process the data. When the performance of the machines is compared, some will take maximum time to process and some will process very soon. So, faster machines are idle and will not be able to proceed further.

c. Complexity

Same size of data set is assigned with each Mapper task. Complexity deals with the time and distance factors. Some of the Mapper tasks are complex because of larger time to process the data and the machines are far away from the processor also getting delayed to proceed.

d. Records are Expensive

Map task processes a set of data as key/value pairs, one at a time. Hence, the processing time does not vary and it requires more CPU and memory to process. It shows that expensive records may be larger than other records. PageRank is an algorithm that experiences the Map phase which assigns ranks (weights) to each and every apex in a graph by aggregating ranks of its inbound neighbor.

e. Heterogeneous Map

MapReduce can be used to follow an n-ary operation by plausibly concatenating different data sets as a unique input. Each dataset may need discrete processing which leads to multi-modal distribution of job at run time. CloudBurst is the MapReduce recital of the RMAP algorithm which supports a set of genome series, for tiny read gene alignment2 and reads not in favor of a reference series.

B. Reducer Side

Various causes of the data skew issues at reducer side are as follows [8] - [10], [15]:

a. Data Partitioning

The hash partitioning and range partitioning are mostly used to partition the data sets. Data partitioning can be done either in two ways using hash functions or range keys which yields an uneven partition and finally resulting to failures. Some reducers may get the more data than the others. The partitioning logic must not rely on the tuples hence it reflects a skew at the reducer side.

b. Key Groups are Expensive

Reducer task processes the (key/value) pair. Mapper task processes the expensive records at runtime but Reducer task causes the skew on expensive key groups.

c. Cluster Size

The cluster size is getting varied by each Reducer. Some of them assigned with large clusters and some of them assigned with small clusters. It will affect the execution time. Large clusters may take more time to complete than the small clusters.

V. TECHNIQUES FOR MITIGATING DATA SKEW ISSUES

From the study it is concluded that the data skew is an important issue and it should be carefully considered in Big Data environments, particularly in MapReduce. In this section, some of the existing skew mitigating techniques are identified and presented. One of the solutions which is already available to mitigate the data skew issue is skew-resilient operators.

A. SkewTune

SkewTune is a technique designed for MapReduce programming model to handle the skew dynamically. It re-balances the load automatically across Map and Reduce tasks in a job [14]. Map task consists of granularity of records and Reduce task consists of key-groups (key/value) pairs, thus the load can be re-balanced. It is designed for MapReduce type engines, distinguished by disk-based processing and a record-oriented data model [10], where each operator reads information from disk and thus decoupled from various other operators in the flow of data [14].

SkewTune constantly scrutinizes the execution of a User Defined Operations (UDOs) and identifies the present bottleneck task that controls and delays the job completion. If such a task is classified, the task is stopped, and its unprocessed data is repartitioned among the idle nodes [14]. The repartition only occurs when new nodes are dynamically added to increase the speed of a job or when there is an idle node present. It stops the task with the longest expected time remaining, though a node is idle and at least one task is expected to take more time to finish the job, it re-partitions and parallelizes the remaining input for that task and takes into account the expected failure availability of all nodes in the cluster. The availability is calculated approximately from the evolution of running tasks. SkewTune also minimizes the side-effect of repartitioning and extensively recover the completion time by up to four times.

By repeating three steps, it mitigates the skew at run time [10]:

- *Detect*: Scrutinize the execution of all tasks in a MapReduce model and gathers the estimation of completion time. When a node becomes idle it monitors the straggler, which is the slowest processing task and takes the longest time to complete.
- *Scan*: Brings to an end of the straggler task. It repartitions the unprocessed data either in parallel or locally to collect the data.
- *Plan*: It maps how to repartition the straggler task from the scanned data and the remaining tasks which take the longest time. After the mitigators are scheduled the process goes back to the Detect phase and proceeds further.

B. SkewReduce

The cause of skew is being an uneven data distribution or an uneven running time across UDO partitions; the speculative execution method is not useful. This technique re-executes the same task on the same input data but on a different system. It may lead to the slow execution time.

Instead, another technique is to use finer-grained partitions so as to minimize the size of serial units of job. The issue with this technique is that finer-grained partitions increase overheads, especially in Hadoop where the overhead of each new task is not negligible [14].

SkewReduce system is a proficient optimizer, parameterized by various user-defined cost functions, which identifies how to partition the data at its best to minimize the skew. It has mainly two components: a. An API for expressing spatial feature-extraction algorithms. b.The static optimizer that partitions the data to guarantee skew-resilient processing is possible [10].

C. LIBRA

Another technique to mitigate the skew at reducer side is by splitting the larger data sets, which is known as LIBRA (Light Weight Implementation of Range Assignment). It is a sampling and partition algorithm which balances the load on Reduce tasks. Either it does not need any pre-run sampling of the input data or it avoids overlap between the Map and Reduce tasks [10]. Data skew mitigation in LIBRA can be done by the following procedure: From the Map tasks, a small portion is treated as a sample task and are allotted at any time if any free slot is readily available in the system. During normal Map task processing the intermediate data, the sample tasks collect the statistics and send the data to the MasterNode after they are completed. The MasterNode decides about the partition and, derives an estimate of the data distribution and then transfers it to the SlaveNodes. The SlaveNodes partitions the data as per the partition decision plan and also the normal Map tasks. Reducer task need not wait until the entire Map task completes its tasks; it can be issued after the decision partitions are ready [11]. LIBRA supports the large cluster splitting technique to mitigate the skew. It decreases the inconsistency and increases the job processing time by partitioning the intermediate data more evenly at the Reducer side and it shows four times improved performance. The operating cost is minimal and negligible overhead is caused by the sampling method used in LIBRA [11]. Main goal is to balance the load across reduce tasks by following three steps: a. Sample partial Map tasks, b. Estimate the intermediate data distribution and c. Apply hybrid partition on the data [10].

D. LEEN

In MapReduce, data skew problems are experienced while shuffling the data. The LEEN (Locality Fairness Aware Key Partitioning) approach tries to balance the load i.e. fairness with data locality in the Reducer phase of a job [14]. By default, hash partitioning is used in Hadoop. It works better if all the keys are uniformly present and they are evenly stored in the data nodes in the cluster. This canopy partitioning leads to performance humiliation, network clogging and unfairness in the reducer's input. So, LEEN method gives an ideal way by following the intermediate key frequency and their distribution. The keys are arranged in descending order, for every key value based on the occurrence that every key has fairness score value achieved by this partitioning. This result overcomes the partitioning by implementing the locality concept in Reducer and achieves a high performance while applying [6].

VI. COMPARATIVE ANALYSIS

1. Comparison of Skew Mitigating Techniques

This paper lists out the comparison of various techniques which provide solution to mitigate the data skew issues occurring on mapper or reducer phase.

S. No	Technique	Mapper/Reducer Phase	Solution
1	SkewTune	Mapper phase	Repartitions the load automatically and detects the straggler node and tries to mitigate by allocating the task to the idle node.
2	SkewReduce	Mapper phase	Static optimization of the input data partition can improve the running time in the presence of skew, caused by uneven processing times. The optimization time is short when compared to the actual query execution time.

3	LIBRA	Reducer phase	It balances the work load among Reduce tasks by splitting large data sets and uses the new sampling method which integrates the small portion of the sample tasks into the normal Map tasks.
4	LEEN	Reducer phase	It tries to balance the load with data locality and fairness value. Partitioning the key according to frequencies and fairness in data distribution among different data nodes.

Table.1. Comparison Table of Skew Mitigating Techniques

VII. RELATED WORKS

Hadoop and the basic overflow of the Mapreduce programming algorithm was focused by AiLing Duan [1]. Kenn Slagter et al [2] provided the partitioning algorithm which improves the memory consumption and load balancing. Nawab Wajid et al [3] suggested the types of skew and the techniques to mitigate the skew issues. Sreedhar et al [5] concentrated on functionality of Hadoop framework. Ancy et al [6] proposed various algorithms and methods for data partitioning such as LARTS, LEEN, HOP, Themis, SkewTune and MAESTRO. YongChul Kwon et al [7] presented a detailed study of skew at runtime in MapReduce environment. YongChul Kwon et al [8] implemented the skew mitigation approach, SkewTune and its effectiveness in real applications. Benjamin Gufler et al [9] presented an algorithm to estimate the cost and proposed balancing approaches as fine partitioning and dynamic fragmentation are based on the cost model and focused with both skewed data and complex Reducer tasks. J Vinutha et al [10] explored different types of skew and the techniques to mitigate the data skew issues. V A Nawale [11] presented various techniques to minimize skew for the MapReduce. QiChen et al [12] implemented the LIBRA technique to tackle the reducer side skew issue. D S Tamhane et al [13] proposed a HACE theorem and examined the issues in the Big Data revolution.

VIII. CONCLUSION

In MapReduce, the processing time of a task gets delayed by many reasons. One of the main reasons is data skew. This paper provides a survey of the various types of skew that arises in Hadoop MapReduce framework and also describes the techniques to mitigate the skew such as SkewTune (dynamic), SkewReduce (static), LIBRA and LEEN, to improve the performance and try to avoid the skew problem. In future research will be encompassed by providing new techniques for mitigating the skew issues in Big Data environment.

REFERENCES

[1] Ailing Duan, "*Research and Application of Distributed Parallel Search Hadoop Algorithm*", International Conference on Systems and Informatics (ICSAI), pp. 2462-2465, IEEE, 2012.

[2] Kenn Slagter, Yeh-Ching Chung, Daqiang Zhang and Ching-Hsien Hsu, "*An Improved Mechanism for Optimizing Massive Data Analysis Using MapReduce*", The Journal of SuperComputing, 66(1), pp. 539-555, 2013.

[3] Nawab Wajid, S. Satish and T. N. Manjunath, "*A Survey on Hadoop MapReduce Framework and the Data Skew Issues*", International Journal of Scientific Research Engineering & Technology, ISSN 2278-0882, Volume 4, Issue 4, April 2015.

[4] Hadoop. [Online]. Available: http://hadoop.apache.org/.

[5] C. Sreedhar, D. Kavitha and K. Asha Rani, "*Big Data and Hadoop*", International Journal of Advanced Research in Computer Engineering and Technology, ISSN 2278 – 1323, Volume 3, Issue 5, 2014.

[6] S. Ancy, S. Srilakshmi Shendurika, B. Krithika and K. Nivetha, "*A Survey on MapReduce and Data Partitioning in Hadoop Implementation*", Indian Journal of Research, ISSN 2250-1991, Volume 4, Issue 4, April 2015.

[7] YongChul Kwon, Bill Howe, Magdalena Balazinska and Jerome Rolia, "*A Study of Skew in MapReduce Applications*", in Proc. of the Open Cirrus Summit, 2011.

[8] YongChul Kwon, Bill Howe, Magdalena Balazinska and Jerome Rolia, "*SkewTune: Mitigating Skew in MapReduce Applications*", in Proc. of the ACM SIGMOD International Conference on Management Data, pp. 25-36, ACM, 2012.

[9] Benjamin Gufler, Alfons Kemper, Angelika Reiser and Nikolaus Augsten, "*Handling Data Skew in MapReduce*", International Conference on Cloud Computing and Services Science, CLOSER 2011.

[10] J. Vinutha and R. Chandramma, "*Skew Types Mitigating Techniques to Increase the Performance of MapReduce Applications*", International Journal of Emerging Technology and Advanced Engineering, ISSN 2250 – 2459 (Online), Volume 5, Special Issue 2, May 2015.
[11] V. A. Nawale and Priya Deshpande, "*Survey on Load Balancing and Data Skew Mitigation in MapReduce Applications*", International Journal of Computer Engineering & Technology, ISSN 0976 – 6367 (Print), ISSN 0976 – 6375 (Online), pp. 32-41, Volume 6, Issue 1, January 2015.
[12] QiChen, Jinyu Yao, and Zhen Xiao, "*LIBRA: Light Weight Data Skew MIitigation in MapReduce*", IEEE Transactions on Parallel and Distributed Systems, 26(9), pp. 2520-2533, IEEE, 2015.
[13] D. S. Tamhane and S. N. Sayyad, "*Big Data Analysis Using HACE Theorem*", International Journal of Advanced Research in Computer Engineering & Technology, ISSN 2278 – 1323, Volume 4, Issue 1, January 2015.
[14] YongChul Kwon, Magdalena Balazinska, Bill Howe and KaiRen, "*Managing Skew in Hadoop*", IEEE Eng. Bull., 36(1), pp. 24-33, 2013.
[15] Qifa Ke, Vijayan Prabhakaran, Yuan Yu, Yinglian Xie, Junfeng Yang and Jingyue Wu, "*Optimizing Data Partitioning for Data-Parallel Computing*", in proc. of the HotOS, pp. 13, 2013.
[16] MapReduce. [Online]. Available: www.tutorialspoint.com/hadoop_mapreduce.htm.
[17] Zhihong Liu, Qi Zhang, Raouf Boutaba, Yaping Liu and Baosheng Wang, "*OPTIMA: On-Line Partitioning Skew Mitigation for MapReduce with Resource Adjustment*", Journal of Network and Systems Management, pp. 1-25, 2016.

