ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 3, March 2014

A Micro Partitioning Technique in MapReduce for Massive Data Analysis

Nandhini.C, Premadevi.P

PG Scholar, Dept. of CSE, Angel College of Engg and Tech, Tiruppur, Tamil Nadu

Assistant Professor, Dept. of CSE, Angel College of Engg and Tech, Tiruppur, Tamil Nadu

ABSTRACT: Over the past years, large amounts of structured and unstructured data are being collected from various sources. These huge amounts of data are difficult to handle by a single machine which requires the work to be distributed across large number of computers. Hadoop is one such distributed framework which process data in distributed manner by using Mapreduce programming model. In order for Mapreduce to work, it has to divide the workload across the machines in the cluster. The performance of Mapreduce depends on how evenly it distributes the workload to the machines without skew and avoids executing job in a poorly running node called straggler. The workload distribution depends on the algorithm that partitions the data. To overcome the problem from skew, an efficient partitioning technique is proposed. The proposed algorithm improves load balancing as well as reduces the memory requirements. Slow running nodes degrade the performance of Mapreduce job. To overcome this problem, a technique called micro partitioning is used that divide the tasks into smaller tasks greater than the number of reducers and are assigned to reducers. Running many small tasks lessens the impact of stragglers, since work that would have been scheduled on slow nodes is only small which can be performed by other idle workers.

Keywords: Hadoop; MapReduce; TeraSort; Partitioning; Skew; Straggler

I. INTRODUCTION

There is a massive improvement in the computer technology which leads to the development of large new devices. Computing devices have several uses and are necessary for businesses, scientists, governments, engineers. The common thing among all computing devices is the potential to generate data. The data may be a person posting to a social media site, a sensor gathering the climate data, a bank or a credit card transaction.

The popularity of the Internet along with a sharp increase in the network bandwidth available to users has resulted in the generation of huge amounts of data. However, the amount of data generated can often be too large for a single computer to process in a reasonable amount of time. Furthermore, the data itself may be too big to store on a single machine. So for reduce the time it takes to process the data, and to have the minimum storage space to store the data, it is necessary to write programs that can execute on two or more computers and distribute the workload among them. While abstractly the computation to perform may be simple, historically the implementation has been difficult.

To address the above issues, Google developed the Google File System (GFS), a distributed file system architecture model for large-scale data processing and created the MapReduce programming model. The MapReduce programming model is a programming abstraction that hides the underlying complexity of distributed data processing. Therefore the difficulty of parallelizing computation, distribute data and handle faults no long become an issue. Hadoop is an open source software implementation of MapReduce, written in Java, originally developed by Yahoo. Since its origin, Hadoop has continued to grow in popularity amongst businesses and researchers.

This work focuses on Hadoop and investigates the load balancing mechanism and straggler problem in Hadoop's MapReduce framework. This is an important area of research because load balancing and straggler problem are the key issues which degrade the performance of Mapreduce job. Moreover this report provides a method to reduce the required memory footprint. Using the proposed methods there will be an improved computation time for MapReduce when these methods are executed on small or medium sized cluster of computers.

The rest of the work is organized as follows: Section 2 deals with the background of MapReduce framework and the motivation for this work. Section 3 gives the problem definition. Section 4 explains the working scenario of Xtrie and Etrie. Section 5 presents the Micro partitioning Technique for MapReduce. Section 6 presents the experimental results Section 7 concludes the report.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 3, March 2014

II. RELATED WORK

The MapReduce is a programming model developed as a way for programs to cope with large amounts of data. This goal achieved by distributing the workload across multiple computers and then working on the data in parallel. From the programmers perspective MapReduce is a relatively easy way to create distributed applications compared to traditional methods. Hadoop is the open source implementation of Mapreduce programming model. Mapreduce works along with a distributed file system called Hadoop Distributed File System (HDFS) in Hadoop. Both Mapreduce and HDFS in hadoop are derived from the Google's Mapreduce and Google File System.

Ghemawat et al [2] presented a File System called Google File System (GFS) which was developed at Google to meet the rapidly growing demands for processing their data. This file system is widely used in Google for storing large data sets. Like other distributed file system, GFS provides high performance, availability,reliability, and scalability. A GFS cluster consists of a single master and multiple chunk servers and is accessed by multiple clients. The master node maintains the file system metadata. Each chunk server contains the storage of chunks which forms the part of the file. These chunks are replicated across multiple machines. The master periodically communicates with the chunk server with the help of heartbeat messages. Chunk replication allows us to tolerate chunk server failures. This File System is taken as the base for developing the file system in Hadoop called HDFS.

Dean et al [1] proposed a framework for processing data in distributed manner called MapReduce. MapReduce is the model for processing and generating large data sets. Programs are automatically parallelized and executed on a large cluster of computers. The run-time system takes care of the details of partitioning the input file, assigning the program's execution across a set of machines, handling machine crashes and managing the required inter-machine communication. Hash partitioning is the default partitioning technique used in MapReduce for dividing the load among the reducers. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system.

Malley [4] proposed sorting terabytes of strings using Hadoop implemented by Apache. Hadoop is an open source software framework for executing tasks in parallel. Hadoop broke the world record by sorting tera bytes of data using the TeraSort method. TeraSort was able to do this sorting process by distributing the workload evenly among the machines. It does this by using custom partitioner. It uses two-level trie to partition the data. Trie is a tree based data structure for storing strings. Hadoop wrote 3 applications for terabyte sort such as TeraGen, TeraSort, TeraValidate.

Zaharia et al [6] presented an algorithm for improving MapReduce performance in heterogeneous environments. A key benefit of MapReduce is that it automatically dealing with failures, beating the complexity of fault-tolerance from the programmer. If a node fails, MapReduce reruns its tasks on a different machine. More importantly, if a node is present but is performing unsuccessfully, a condition that we call a straggler, MapReduce runs a speculative copy of its task on another machine to finish the computation faster. Hadoop's scheduler starts speculative tasks based on a simple heuristic comparing each task's progress to the average progress. Although this program works well in homogeneous environments, it is difficult to differentiate slowly running nodes and the stragglers in heterogeneous environments. To overcome this problem, the author proposed an algorithm called LATE algorithm to address the node heterogeneity. The proposed LATE algorithm reduces Hadoop's response time.

Shafer et al [5] presented a distributed file system called HDFS. This is the file system used by Hadoop which is the clone of GFS. All files in the HDFS follow the write-once, read-many access rule. In HDFS each file is divided into blocks of default size 64MB and are stored in each node. Each block is replicated across multiple nodes for redundancy. Default replication factor in HDFS is 3 which can even set manually if required. The HDFS consists of two main components namely: 1.Namenode 2.Datanode.The namenode is the master node which maintains the metadata for all the files and directories. Namenode knows the datanodes on which all the blocks for a given file are located. The Datanode is the slave node where the actual data is residing. The datanode sends heartbeat message to namenode every 3seconds to report its status.

Gufler et al [3] addressed the Data Skew problem in MapReduce. The author proposed how to efficiently process MapReduce jobs with complex reducer tasks over skewed data. For this purpose two load balancing approaches are proposed namely, fine partitioning and dynamic fragmentation. Fine partitioning splits the input data into a fixed number of partitions. The number of partitions is larger than the number of reducers, and the goal is to distribute the partitions such that the execution times for all reducers are similar. In dynamic fragmentation, expensive partitions are split locally by each mapper while they are created, and tuples are replicated if necessary. As a result, the cost of the partitions is more uniform and a good load balancing is easier to achieve for highly skewed distributions.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 3, March 2014

III. PROBLEM DEFINITION

The tremendous growth in the computer networks enlarge the network bandwidth available to users has resulted in the generation of huge amounts of data. For a single computer the amount of data generated is too large so that the computation of the data takes more time and also increases the storage space. In order to reduce the computation time and decreases the storage space the workload is distributed on two or more computers.

Hadoop sorts terabyte of data called Terasort which uses two-level Trie to partition the data. A Trie is a tree-based data structure for storing strings. In a two-level Trie only first two characters of a string is considered during the partitioning phase. This reduces the effectiveness of the load balancing in TeraSort method. Another problem which degrades performance of Mapreduce job is the slow running node (straggler). A node is available but performing slowly reduces the effectiveness of the Mapreduce job. So it is necessary to have a technique which addresses the problem of Data Skew and Straggler.

IV. XTRIE AND ETRIE

Hadoop sorts terabyte of data using Terasort method. Terasort uses trie to partition the data among the reducers. A trie is a tree based data structure used for storing strings. In terasort only two-level trie is used which considers only first two characters in the string. This two-level trie is build using cut-point algorithm. The cut-points are obtained by dividing the total number of strings to be inserted in trie by number of reducers. Using cut-points strings are divided among the reducers.

However without the knowing the number of strings in each prefix two-level trie leads to load imbalance among the reducers. In order to overcome the load imbalance problem faced in using two-level Trie to partition the data, a technique called Xtrie is proposed to partition the data. In this for each word in the trie it maintains the counter value. Consider the example set of keys, {“act”, “acid”, “add”, “adult”, “born”, “good”, “grip”, “grow”, “peal”, “pearl”, “pedal”, “soft”, “stork”, “storm”}.

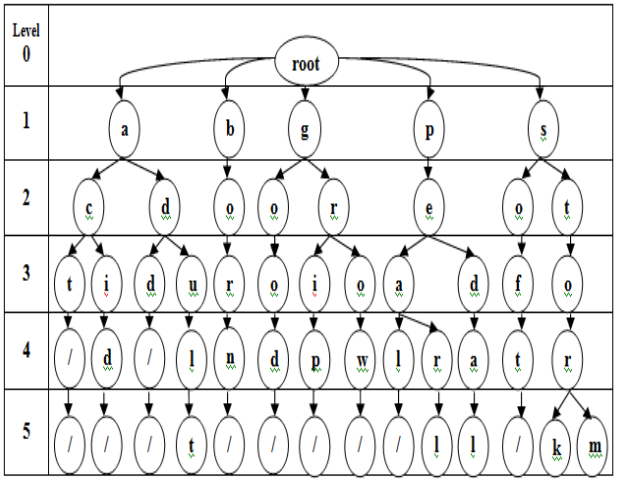

Fig. 1. Strings stored in the Trie

The above strings are stored in the trie as in Figure 1.By using two level trie the above set is reduced to {“ac”, “ad”, “bo”, “go”, “gr”, “pe”, “so”, “st”}.

Consider there are 4 reducers and the above set is divided into 4 partitions. Each partition is Figure 1.Strings stored in the trie send to one reducer. Thus,

Reducer-1 process keys starting with {“ac”, “ad”}.Total:4 keys
Reducer-2 process keys starting with {“bo”, “go”}.Total:2 keys
Reducer-3 process keys starting with {“gr”, “pe”}.Total:5 keys
Reducer-4 process keys starting with {“so”, “st”}.Total:3 keys

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 3, March 2014

This result in workload imbalance among the reducers.To overcomes the above problem, in Xtrie it uses counter for each key value. Using counter, partitioner can distribute the total number of keys among the reducers evenly. This Xtrie working is explained using the Table 1. In the table the partition is made by considering the number of occurrences of each string. So strings with prefix ac and ad goes to partition 1, strings with prefix bo, go and gr goes to partition 2, strings with prefix pe and so goes to partition 3 and finally the string with prefix st goes to partition 4.

The formula for calculating the TrieCode is shown in Equation (1) below which is used as an index to the array.

$$\text{TrieCode} = W_n \times 256^{n-1} + W_{n-1} \times 256^{n-2} + \ldots + W_1 \times 256^0 = \sum_{n=1}^{TotalWord} W_n \times 256^{n-1} \quad (1)$$

The trie is represented by using array. Each node in the trie will contain a maximum of 256 children. It is not possible to have all the 256 children for a single node. This problem will make the trie to occupy lots of memory space.

To reduce the memory requirements of trie, an algorithm called ReMap algorithm is used which reduces the 256 characters on ASCII to 64 elements as required by Etrie method. Using less memory allows deeper tries to be built. Deeper tries also provides the chance of distributing keys evenly among reducers.

This problem can be overcome using deeper tries. By using deeper tries, the length of the prefix can be increased. This increases the number of prefix and reduces the number of keys per prefix. Using this even distribution of keys among the reducers can be created. The above set of keys using three-level is represented using prefix {“gra”, “gre”, “gri”, “gro”, “gru”, “gul”} thus dividing the 8 keys represented by prefix “gr” into five smaller categories.\

V. MICRO PARTITIONING TECHNIQUE FOR MAPREDDUCE

A major benefit of MapReduce is that it automatically handles failures, hiding the complexity of fault-tolerance from the programmer. If a node crashes, MapReduce re-runs its tasks on a different machine. More importantly if a node is available but is performing poorly, a condition called straggler, MapReduce runs a speculative copy of its task or backup task on another machine to finish the computation faster. Without the mechanism of speculative execution, a job would be as slow as the misbehaving task. Stragglers can be caused by hardware failures, transient software issues, or clusters composed of a heterogeneous mix of hardware.

Prefix	Keys	Trie code	Count	Partition
ac	act	0X6163	2	1
	acid			
ad	add	0X6164	2	
	adult			
bo	born	0X626f	1	2
go	good	0X676f	1	
gr	grip	0X6772	2	
	grow			
pe	peal	0X7065	3	3
	pearl			
	pedal			
so	soft	0X736f	1	
st	stork	0X7374	2	4
	storm			

Table 1. Xtrie Partitioner using Two-Level Trie

Straggler is one the common performance issue in Mapreduce system. Stragglers can impact job performance because a job’s completion time is determined by the completion time of its final task. Several techniques have been

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 3, March 2014

developed to address stragglers. Here a new approach called micropartitioning is used for avoiding skew and stragglers during the reduce phase.

The key technique is to run a large number of reduce tasks, splitting the map output into many more partitions than reduce machines in order to produce smaller tasks. These tasks are assigned to reduce machines as workers become idle, allowing the task scheduler to dynamically mitigate skew and stragglers. Running many small tasks lessens the impact of stragglers, since work that would have been scheduled on slow nodes is only small which can now be performed by other idle workers. By assigning smaller units of work, jobs can derive benefit from slower node.

The procedure for implementing the micro partitioning technique is as follows:

Step 1: Take the input samples
Step 2: Store the input samples in trie
Step 3: Build the two-level trie
Step 4: Count the occurrence of each prefix
Step 5: Using cut-point algorithm determine the cut-points (split points)
Step 6: Split points are obtained by dividing the sum of counter value by number of partitions+1
Cut points = sum of counters/number of partitions+ 1
Step 7: Using cut points send the keys to appropriate reducers

```
if (key<cutpoint1)
        Send key to reducer 1
else if (key>=cut-point1&&key<cut-point2)
        Send key to reducer2
else if (key>=cut-point2&&key<cut-point3)
        Send key to reducer3
else
        Send to the finished reducer
```

Step 8: Determine if there is any slow running node by comparing the performance of each node with other
Step 9: If there is any such node move the data that is processing on that node to the free node.

VI. EXPERIMENTAL PROCEDURE

The experiment is conducted by using hadoop-1.2.1.The system requirements are Pentium Dual-core CPU, 2GB RAM, 50GB hard disk, java version 1.6 or any other versions above version 6, Cygwin and Windows Xp.

Hadoop is the open source implementation of Mapreduce. Many versions of Hadoop are available. The Hadoop version 1.2.1 is one of the latest releases and is considered as the stable one.

VII. RESULT

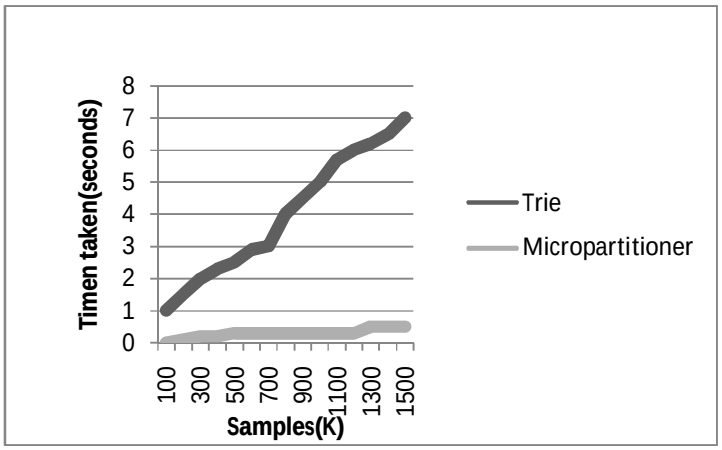

Fig.2. Time Complexity Comparison of Trie and Micro partitioner.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 3, March 2014

The performance of the proposed work is evaluated using the given experimental setup. It is then compared with the existing techniques like trie to show that the performance of the micro partitioning technique has good load balancing among the reducers and also it reduces the impact of stragglers in Mapreduce framework. The figure 2 shows the time complexity comparison of Trie and Micro partitioner.

VIII. CONCLUSION AND FUTURE WORK

Hadoop is the distributed processing environment which is used for processing huge volume of data. It is the partitioning of the data which determines the workload of the reducers. In order to overcome the problem of skew, the data should be partitioned efficiently among the reducers. If the data are partitioned equally among the nodes then the execution time for the overall Mapreduce job is decreased. The proposed techniques can be used to efficiently mitigate the problem of data skew in reducer and thereby decreasing the processing time of Mapreduce job.

Another issue which degrades the performance of Mapreduce job is the straggler. To overcome the straggler problem the output from mappers are divided into smaller tasks larger than the number of reducers and are assigned to reducers in "just-in-time" fashion. This technique is called micro partitioning. Using this technique the problem of both skew and straggler can be overcomed.

REFERENCES.

1. J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters", CommunACM 51:107–113, (2008).
2. S. Ghemawat, H. Gobioff, S-T Leung, "The Google file system", Proceedings of the 19th ACM symposium on operating systems principles (SOSP), (2003).
3. B. Gufler, N. Augsten, A. Reiser and A. Kemper, "Handling Data skew in Mapreduce", Proceedings of the international conference on Cloud Computing and Services Science(CLOSER), pp. 574-582, (2011).
4. O. O'Malley , "TeraByte sort on Apache Hadoop", (2008).
5. J. Shafer, S. Rixner, AL. Cox,"The hadoop distributed filesystem: balancing portability and performance", Proceedings of the IEEE international symposium on performance analysis of system and software (ISPASS), p 123, (2010).
6. M. Zaharia, A. Konwinski, A. Joseph, R. Katz, I. Stoica, "Improving MapReduce Performance in Heterogeneous Environments", 8th USENIX Symposium on Operating Systems Design and Implementation, pp 29-42,(2008).
7. Hadoop, http://hadoop.apache.org/core

BIOGRAPHY

C.Nandhini pursuing M.E., Computer Science and Engineering in Anna University, Chennai. She did her B.E., Computer Science and Engineering in Anna University of Technology, Coimbatore. She is a member of MISTE, IAENG. Her area of interest is Data Mining and Big data. She has published a paper in journal and presented 3 papers in National Conferences and attended various seminars and workshops to improve her knowledge in various domains.

P.Premadevi working as Assistant Professor in Angel College of Engineering and Technology, Tiruppur. She did her M.E., Computer Science and Engineering in Anna University of Technology, Coimbatore. . She did her B.E., Computer Science and Engineering in Anna University, Chennai. She is a member ISTE, IAENG. Her area of interest is Database, Data Mining and Big data. She has published 4 papers in various journals and presented 5 papers in National and International conferences and attended many workshops, seminars and FDP.