ISSN: 0975-766X
CODEN: IJPTFI
Available Online through
Research Article
www.ijptonline.com

JOB SPLITTING BASED ON DATA ANALYSIS

Maria Anu.V[1], P.Pavithra, N.T.Pooja
Assistant Professor, Faculty of Student, Faculty of Computing, Faculty of Computing, Computing, Sathyabama University, Sathyabama University, Chennai.
Email: mariaanu18@g mail.com

Received on 25-02-2016 *Accepted on 19-03-2016*

Abstract

We propose an efficient dynamic data splitting strategy on Hadoop which monitors the samples while running batch jobs and allocate resources to slaves depending on the complexity of data and the time taken for processing. We also show the effectiveness of Web crawling using Hadoop eliminating DDOS attack detection scenarios that will happen on the servers we are crawling. Query processing done through Map Reduce in traditional Hadoop clusters is replaced by another technique we developed ie. Block chain query processing and we compare the response times to show its effectiveness. Block chain proved to be as best as Map Reduce and can be used in data intense results. This project presents LIBRA, a lightweight strategy to provide solution for the data skew problem for reduce -side applications in Map Reduce and to show the effectiveness of Hardtop on Web Crawling of Large Datasets form Web Servers.

Keywords: Big data; Data Skew; Map Reduce; Web Crawling; Data Analysis; Query Processing.

I. Introduction

The past decade has witnessed the explosive growth of data for processing. Big Internet companies routinely create hundreds of tera-bytes of logs and operation records. Map Reduce has proven itself to be an useful tool to process such huge data sets. It divides a job into multiple small tasks and assigns them to a huge number of nodes for parallel processing. Due to its remarkable simplicity and fault acceptance, MapReduce has been mostly used in various applications, adding web indexing, log analysis, data min ing, scientific simulations, machine translation, etc. There are several parallel computing frameworks that support Map Reduce, such as Apache Hadoop, Google MapReduce, and Microsoft Dryad, of which Hadoop is open-source and widely used. The job completion time in Map Reduce relays on the slowest running job in the job. If one task takes significantly longer to finish than others (the so-called

straggler) it can slow down the progress of the entire job. Stragglers can occur due to different reasons, among which data skew is a prime one. Data skew refers to the imbalance in the amount of data assigned to each job, or the imbalance in the amount of job required to process such data. The basic logic of data skew is that data sets in the real world are usually skewed and that we do not know the distribution of the information beforehand. Note that this problem cannot be solved by the suppositional execution strategy in Map Reduce.

Map Reduce is a useful tool for parallel data generating. One significant problem in practical Map Reduce applications is data skew. The data skew problem is a common and important problem that needs to be solved in distributed systems. It has been studied in the parallel database area, but only limited on join group and aggregate operations. Some of these technologies have previously been carried to Map Reduce, like Skewed Join in Pig. However, users generally still need to implement their own methods for their specific applications to tackle the data skew, such as Cloud Burst and Skew Reduce. Data skew has also been studied in the Map Reduce environment during the past three years. Okcanetal. Propose a skew optimization for the theta join by adding two pre-run sampling and counting jobs. Kwon et al. provide a system called Skew Reduce which optimizes the data splitation for the spatial feature extraction application by operating preprocessing filtering and sampling procedures. Although these solutions can mit igate data skew to some extent, they have significant overhead owing to the pre-run jobs and are applicable only to certain applications.

II. Literature Survey

[4] Large enterprises have been relying on parallel database management systems (PDBM S) to process their ever-increasing data content and complex queries. The scalability and accomplishment of a PDBMS comes from load assess on all nodes in the system. Skewed imp lementation will significantly slow down query response time and degrade the overall system accomplishment. Business intelligence tools used by enterprises mostly generate a large number of outer joins and require high performance from the underlying database systems. Although wide research has been done on controlling skewed processing for inner joins in PDBMS, there is no famous research on data skew handling for parallel outer joins. We put forward a simple and excellent outer join algorithm called OJSO (Outer Join Skew Optimization).

[5][6] We present an approach to use with skew in parallel joins in DB systems Our approach is easily implementable within present parallel DBMS, and performs well on skewed data without break down the performance of the system on non - skewed data The main concept is to use multiple algorith ms, each specialized for a dierent degree of skew,

and to deal a small sample of the relations being joined to identify which algorithm is appropriate We developed, implemented, and experimented with four new skew - controlling parallel jo in algorithms; one, which we specify virtual processor range separating, was the clear winner in big skew cases, while traditional mixed hash join was the clear success in lower skew or no skew cases We present experimental outputs from an execution of all four algorithms on the Gamma parallel database machine To our idea, these are the rst reported skew - controlling numbers from an actual implementation

[7] Data analyzing and processing are important jobs in cloud computing. In this field, the MapReduce framework has become a more and more popular tool to analyze large -scale data over huge clusters. Compared with the parallel relational database, it has the merits of superb scalability and good fault tolerance. However, the performance of join operation using MapReduce is harm as that of parallel relational database. Thus, how to optimize theta-join operations using MapReduce is an attractive position to which researchers have been paying attention. In this paper, a randomized procedure named Strict - Even-Join(SEJ) is planned to solve the multi-way theta-joins in a one MapReduce job. Moreover, a dynamic programming algorithm is detailed to optimize the multi-way theta-joins by calling the SEJ algorithm. The output of experiments show that our approach is feasible and effective.

[8] Many commercial database systems use some form of statistics, typically histograms, to brief the contents of relations and permit efficient cost of required quantities. While there has been acceptable work done on identifying excellent histograms for the estimation of query-result sizes, less attention has been paid to the cost of the data distribution of the result, which is of significance in query optimization. In this paper, we prove that the optimal histogram for analy zing the size of the output of a join operator is optimal for calculat ing its data distribution as well. We also study the effectiveness of these optimal histograms in the context of an significant application that requires estimates for the data distribution of a query output: load-balancing for parallel Hybrid hash joins. We simplify a cost formula to capture the effect of info skew in both the input and output connect on the load and use the optimal histograms to estimate this cost most accurately. We have designed and carried out a load balancing algorithm using these histograms on a simulator for the Gamma parallel DB system. The experiments establish the superiority of this way compared to earlier ones in controlling all kinds and levels of skew while incurring negligible overhead.

III. Proposed Structure

We propose an efficient dynamic data splitting strategy on Hadoop which monitors the samples while running batch

jobs and allocate resources to slaves depending on the complexity of data and the time taken for processing. We also show the effectiveness of Web crawling using Hadoop eliminating DDOS attack detection scenarios that will happen on the servers we are crawling. Query processing done through MapReduce in traditional Hadoop clusters is replaced by another technique we developed ie. Block chain query processing and we compare the response times to show its effectiveness. Block chain proved to be as best as MapReduce and can be used in data intense results. Unlike previous work, LIBRA does not require any pre-run sampling of the input data or stop the overlap between the map and the reduce stages. It uses a unique sampling method which can achieve a highly exact approximation to the distribution of the in between data by sampling only a small fraction of the in between data during the normal map processing. It allows the reduce jobs to start copying as soon as the selected sample map tasks (only a small fraction of map jobs which are issued first) complete. It supports the split of huge keys when application semantics permit and the whole order of the output data. It considers the heterogeneity of the calculating resources when balancing the load among the reduce tasks appropriately. LIBRA is applicable to a huge range of applications and transparent to the users. We executed LIBRA in Hadoop and our experiments show that LIBRA has negligible overhead and can increment the execution of some well known applications by up to a factor of 4.

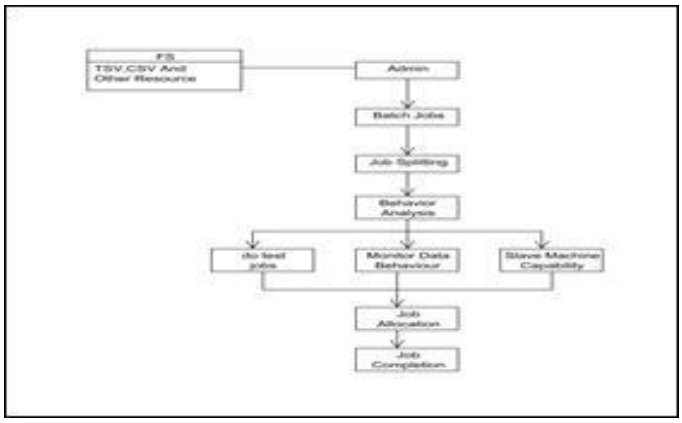

Fig. 1. Query Processing using Mapreduce.

A. Data Skew and its problem

The imbalance in the amount of data assigned to each task causes some tasks to take much longer to complete than others and can significantly impact accomplishment. This paper presents LIBRA, a lightweight strategy to provide solution for the data skew problem for decrease-side applications in Map Reduce. In Traditional Hadoop clusters the data is divided into partitions on a static way that each slave node will be allocated with equal size of data which might lead to the Data Skew problem. As all slave nodes are not equally capable and also the complexity of data will not also be similar some slave nodes may process the task assigned for a longer time than the other slave nodes in the

cluster.

B. Web Crawling using Hadoop

Web crawling is the fetching of data Resources residing in any of the web server with or without the knowledge of the Resource provider. Generally the Resource servers will incur high traffic and load while web crawling is done and it can be detected by traditional DOS (Den ial of Service) detection schemes when implemented through a single server. So we implement an efficient way to crawl resources residing any server through our Hadoop cluster in which the traditional DOS detection methods could fail to detect the attack. All the Resources crawled will be stored for future use. Web crawling jobs include PDF Web crawling stored as PDF format and medical question and answers web crawling stored as CSV (Co mma Separated Values) format.

C. Data Analysis by Sampling and Data Skew Mitigation

Since data skew is difficult to solve if the input distribution is unknown, a natural thought is to examine the data before deciding the partition. There are two common ways to do this. One approach is to launch some pre-run jobs which examine the data, collect the distribution statistics, and then decide an appropriate partition. The drawback of this approach is that the real job cannot start until those pre-run jobs finish. The Hadoop range partitioned belongs to this category; it can increase the job execution time significantly. The other approach is to integrate the sampling into the normal map process and generate the distribution statistics after all map tasks finish. Since reduce tasks cannot start until the partition decision is made, this approach cannot take advantage of parallel processing between the map and the reduce phases. We take a different approach that the data assigned earlier depending upon the capacity of machines is dynamically split based on the complexity, job execution rate and assigned to other nodes. We show the completion of jobs of slave nodes in a optimal manner.

D. Query Processing using Map Reduce and Block Chain

The Query processing can be done using Map Reduce Framework of Hadoop system and the results are rendered to the client page. Multiple Reduce tasks are done on the map task to give the reduced object and the results can be rendered as and when user needs. The answering time is calculated for Querying with Map Reduce. We also imp lement the same Querying with our own approach Block Chain Algorithm which stores the Map task output locally on the slave machine and uses cache based rendering of results. The Response time is compared with the Map Reduce response time and is up to the mark. This can show more effectiveness when used with large data with small cluster setup. As shown in Fig.1.

IV. Background and Related Work

Mapreduce Programming Model

It is taken as the heart of the hadoop. MapReduce is an inherently parallel data implementing programming model. There are different type of frameworks like the google MapReduce, Microsoft Dryad and Apache's Hadoop which provides MapReduce programming model. Among all these, the Apache's hadoop is an open source. The MapReduce programs executed by the Hadoop can be written in various languages like java, python, ruby and c++. All these MapReduce programs are info intensive in nature and process very large data sets in a parallel fashion.

A. Data processing in MapReduce

The working of MapReduce is based on these two phases. Map phase composed of the Map function and Reduce phase consists of the Reduce function.

The first phase is the map phase which takes the core data such as the text file as the input. The input is spliced into the several parts known as splits and each split is injected to the separate map task. The size of the split is same as the size of the info block on the HDFS. Map task transforms the input info into the (key, value) pairs which is also known as the intermediate data. Combiner functions. The jointer function output serves as the input to the reduce jobs. The combiner functions help to send off the (key, value) pairs that share the same key to the same partition. The partitions are finalized by a default paritioner such as hash partitioner or some user defined paritioners. The spots about these partitions are sent to the NameNode. The NameNode then assigns a reduce tasks to the nodes and also passes the information about these partitions to the hose nodes. Thus the reduce nodes communicates with those partitiones and are injected by (key, value) pairs present in those partitiones. Thus the reduce task is performed which processes and simplifies the intermediate data. The reducer output is stored directly on the HDFS wh ile the Mapper output is stored on the local file system.

Hadoop provides the data locality optimization which helps to run a map task on the node where the input data resides on the HDFS. The MapReduce always splits the input data for parallel process. Each split is smaller as compared to whole file and thus each split takes less time to get processed. So when we process the large number of splits in parallel, the processing is better load equalized.

Hence, a faster machine will be able to process more splits over the course of the job than a slower machine. The splits must not be too small otherwise we may face an overhead of managing the splits. In most cases, the size of splits is same as size of HDFS b lock (64M B). If there is a single reduce task, the output from all the map tasks is fed

to that single reduce task. Thus there is no data locality optimization in reduce tasks. Here the map outputs have to be moved across the network to the node where the reduce task is carried out. To guarantee th e reliability, the reduce outputs are stored in HDFS and are replicated across the nodes with one replica on the local node. BUT when there are mu ltiple reducers, a partitioning function is needed to separate the map results into various partitiones. Each partition gets the (key, value) pairs with same keys. Thus the number of partitiones is analogous to the number of various keys.

B. Stragglers in MapReduce

As discussed earlier, the straggler is the slowest running task which delays the execution of entire job. Stragglers are caused by various factors.

It is very easy to overcome the straggler caused by the external factors. The commonly used method is called as the speculative execution. If the machine is performing slowly, or if a machine fails or if there is any faulty hardware in the machine, we can overcome it by simply shifting the workload to some other machine which is performing well.

In this way we can easily overcome the straggler issue.

However, the speculative execution cannot be used when the straggler is caused by the internal factors of the data like the physical properties of data (height of persons, weight of persons), the speculative execution cannot be used since moving the work to other device does not change the characteristic of the data and thus cannot overcome the straggler issue. This straggler caused by this issue is known as the data skew. Data skew can occur in both the phases of the MapReduce.

When the data skew occurs in the Map phase, it can be easily mitigated by splitting the map tasks. The more comp lex data, that takes time and is difficult to process, is responsible for the data skew on the Map phase. The data skewlon the map phase is rarely observed. The data skew on the reduce side is very difficult to overcome and is a challenging problem. A number of data-intensive applications like the data mining and web indexing applications as well as the scientific data-intensive applications have witnessed the same data skew problems.

V. Algorithm

A. Block Chain Algorithm

A block chain is a transaction database shared by all nodes joining in a system based on the Bitcoin protocol. A whole copy of a currency's block chain contains all transaction ever performed in the currency. With this data, one can examine how much value belonged to each address at any point in history.

Every block composes a hash of the earlier block. This has the effect of creating a chain of blocks from the genesis block to the present block. Each block is guaranteed to come after the earlier block chronologically becaus e the earlier block's hash would otherwise not be known. Each block is too computationally impractical to modify once it has been in the chain for a while since every block after it would also have to be again created. These characters are what make double - spending of bitcoins very arduous. The block chain is the main innovation of Bitcoin.

B. MapReduce Algorithm

The MapReduce algorithm composed of 2 important tasks,namely Map and Reduce.

The map task is implemented by means of Mapper Class.

The reduce task is done by means of Reducer Class.

Mapper class considers the input, tokenizes it, maps and sorts it. The result of Mapper class is applied as input by Reducer class, which in turn searches matching pairs and decreases them. MapReduce implements various mathematical algorithms to splice a task into minute parts and allots them to multiple devices. In technical terms, MapReduce algorithm assistance in sending the Map & Reduce tasks to exact servers in a cluster.

VI. Conclusion

Hence we designed and developed a Hadoop cluster which can mitigate Data Skew problem and we show its effectiveness using Block Chain algorith m and Map reduce Algorithm with comparison. We also demonstrate the effectiveness of Hadoop cluster for web crawling in various real time web servers.

References

1. J. Dean and S. Ghemawat, Map reduces: Simplified data processing On large clusters,‖ Commun. ACM, vol. 51, pp. 107–113, Jan.
2. M. Isard, M. Budiu, Y. Yu,. Birrell, and D. Fetterly, Dryad: Distributed Data-parallel programs from sequential building blocks,‖ In Proc. ACM SIGOPS/EuroSys Eur. Conf. Comput. Syst ., 2007, pp. 59– 72.
3. Y. Kwon, M. Balazinska, and B. Howe, A study of skew in map reduceApplications,‖ in Proc. Open Cirrus Summit, 2011.
4. Y. Xu and P. Kostamaa, Efficient outer join data skew handling in Parallel dbms,‖ Proc. VLDB Endowment, vol. 2, no. 2, pp. 1390–1396, 2009.
5. C. B. Walton, A. G. Dale, and R. M. Jenevein, A taxonomy and Performance model of data skew effects in parallel joins,‖ in Proc. Int. Conf. Very Large Data Bases, 1991, pp. 537–548.

6. D. J. DeWitt, J. F. Naughton, D. A. Schneider, and S. Seshadri, Practical skew handling in parallel joins,‖ in Proc. Int. Conf. Very Large Databases, 1992, pp. 27–40.
7. J. W. Stamos and H. C. Young, A symmetric fragment and rep licate Algorithm for distributed joins,‖ IEEE Trans. Parallel Distrib. Syst., vol. 4, no. 12, pp. 1345–1354, 1993.
8. V. Poosala and Y. E. Ioannidis, Estimation of query -result distribution And its application in parallel-join load balancing,‖ in Proc. Int. Conf. Very Large Data Bases, 1996, pp. 448–459.
9. Rajalakshmi V., Jothi Nisha V. and Dhanalakshmi S(2015), ‖ Efficient retrieval of data from cloud using data partitioning method for banking applications [rbac], ARPN Journal of Engineering and Applied Sciences, ISSN: 1819-6608 , Vol. 10, no. 4, pp. 1834-1838.(Scopus, Google Scholar Indexed)
10. Gowri, S.and Anandha Mala, G. S. (2015), Strategic Enhancement of Collaborative Framework for Novelty in Retrieval from Digital T extual Data Deploying DSPC and RBWM Algorithms for Forensic Analysis‖,
11. Journal of Engineering and Research, Vol. 3, No. 4, ISSN:0974-6846. Impact Factor: 0.128. (WOS &Annexure-I -Scopus Indexed).
12. Devi, L. and Gowri, S. (2015), Optimizing mapreduce functionality in big data using cache manager‖, ARPN Journal of Engineering and Applied Sciences, Vol. 10, No. 12, pp.5223-5228,ISSN: 1819-6608.(Scopus Indexed).

Corresponding Author:
Maria Anu.V,
Email: *mariaanu18@g mail.com*