International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 20 Issue 3 – MARCH 2016.

EFFECTIVE DATA SKEW MITIGATION

[1]S.Thilagavathi, [2]G.Vidhya, [3]R.Sowmya
[1,2]*UG Scholar, Computer Science and Engineering, Dhanalakshmi College of Engineering*
[3]*Assistant Professor, Department of Computer Science and Engineering, Dhanalakshmi College of Engineering*

thilagavathisivaraj@gmail.com
vidhyarajan94@gmail.com
sowmyabuddy90@gmail.com

Abstract **- The production of data is expanding at an astonishing pace. The dramatic rise of unstructured data like photos, videos and social media has ushered in a new breed of non-relational databases and which are termed as "Big Data". In 2012, the amount of information stored worldwide exceeded 2.8 Zetabytes. By 2020, the total amount of data stored is expected to be 50 times larger than today. Big Data has to be processed and analysed to produce potential concepts and an ultimate knowledge has to be understood from this ocean of data. A popular framework Hadoop is used currently for processing such huge data. Here we implement LIBRA with different approach that significantly reduces the data skew-(a common issue in map-reduce) by dynamic splitting strategy without pre-sampling. To provide query-processing, a new algorithm Block-Chain is introduced.We also show the effectiveness of Web crawling using Hadoop eliminating DDOS attack detection scenarios.**

Index Terms **– Block chain, data skew, map reduce, query processing.**

I. INTRODUCTION

Large Internet companies routinely generate hundreds of tera-bytes of logs and operation records. Map Reduce has proven itself to be an effective tool to process such large data sets. An effective parallel computing framework that supports Map-reduce is Hadoop. Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. A Hadoop frame-worked application works in an environment that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key.

MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large As the sequence of the name MapReduce implies, the reduce task is always performed after the map job. The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model. One significant issue in practical MapReduce applications is Data skew- the imbalance in the amount of data assigned to each task, or the imbalance in the amount of work required to process such data. The fundamental reason of data skew is that data sets in the real world are often skewed and that we do not know the distribution of the data beforehand. Note that this problem cannot be solved by the speculative execution strategy in Map Reduce.

In this paper we address the problem of efficiently processing MapReduce jobs with complex reducer tasks over skewed data. The data skew problem in MapReduce has been studied. Among the solutions proposed, some are specific to a particular type of applications, some require a pre-sample of the input data, and some cannot preserve the total ordered result as the applications require. To make matters more complicated, the computing environment for MapReduce in the real world can be heterogeneous as well—multiple generations of hardware likely to co-exist in the same data center .When MapReduce runs in a virtualized cloud computing environment such as Amazon EC2 , the computing and storage resources of the underlying virtual machines (VMs)can be diverse for a variety of reasons. A good partition method should take this into consideration instead of always dividing the work evenly among all reducers.

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 i.e. **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 prevent the overlap between the map and the reduce stages. It uses an innovative sampling method which can achieve a highly accurate approximation to the distribution of the intermediate data by sampling only a small fraction of the intermediate data during the normal map processing. It allows the reduce tasks to start copying as soon as the chosen sample map tasks (only a small fraction of map tasks which are issued first) complete. It supports the split of large keys when application semantics permit and the total order of the output data. It considers the heterogeneity of the computing resources when balancing the load among the reduce tasks appropriately.

II. EXISTING SYSTEM

A. Map-Reduce Framework

A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

Typically the compute nodes and the storage nodes are the same, that is, the MapReduce framework and the Hadoop Distributed File System are running on the same set of nodes. This configuration allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in very high aggregate bandwidth across the cluster.

In a MapReduce system, a typical job execution consists of the following steps:

1) After the job is submitted to the Map-Reduce system, the input files are divided into multiple parts and assigned to a group of map tasks for parallel processing.

2) Each map task transforms its input (K1, V1) tuples into intermediate (K2, V2) tuples according to some user defined map and combine functions, and outputs them to the local disk.

3) Each reduce task copies its input pieces from all map tasks, sorts them into a single stream by a multiway merge, and generates the final (K3, V3) results according to some user defined reduce function..

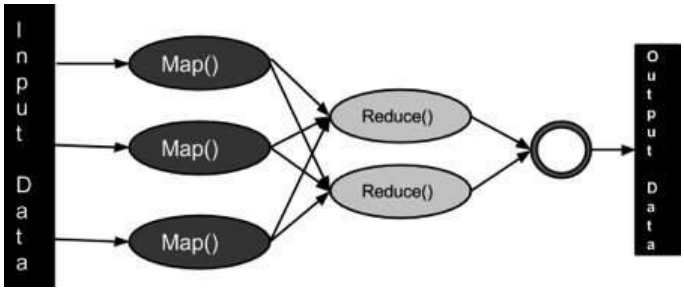

Fig. 1 MapReduce Architecure

B. Data Skew In Mapreduce

MapReduce systems have become popular for processing large data sets and are increasingly being used in e-science applications. In particular, (a) the runtime complexity of the reducer task is typically high, and (b) scientific data is often skewed**.** The imbalance in the amount of data assigned to each task causes some tasks to take much longer to finish than others and can significantly impact performance. 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. Master node will look up for the completion of all Map Tasks and in this scenario will wait for the long running task on the slave node to complete. This time lag should be eliminated so that the master to through the overall output.

To maximize performance, ideally we want all tasks to finish around the same time. When some tasks takes usually long time to complete, it is called straggler and can delay the progress of the job significantly. For stragglers caused by the external factors such as faulty hardware, slow machines, etc. To address this type of skew, MapReduce and Hadoop include a mechanism where the last few tasks in a job are speculatively replicated on a different machine and the job completes when the fastest replicas of these final tasks complete. More commonly, skew occurs when data is not evenly partitioned across tasks. This type of skew, called data skew, typically affects reducers since map tasks are normally assigned same-size chunks of input data. For reducers, the problem can occur either when a reducer processes a larger number of keys or a larger number of values than other reducers.

Data skew can occur in both the map phase and the reduce phase. Map skew occurs when some input data are more difficult to process than others, but it is rare and can be easily addressed by simply splitting map tasks. In contrast, data skew in the reduce phase (also called reduce skew or partitioning skew) is much more challenging. The MapReduce framework requires that all tuples sharing the same key be dispatched to the same reducer. However, for an arbitrary MapReduce application, the distribution of the intermediate data cannot be determined ahead of time.

We implement a LIBRA approach in which the data skew is completely removed and query processing by the end-user has less response time in comparative with the existing system. This approach never uses pre-sampling of the input data. Instead, it performs the data analysis by memory loading.

C. LIBRA

The existing system implements the LIBRA approach to solve data skew for general applications. The design goals of LIBRA include the following:

Transparency: Data skew mitigation should be transparent to the users who do not need to know any sampling and partitioner details.

Parallelism: It should preserve the parallelism of the original MapReduce framework as much as possible. This precludes

any pre-run sampling of the input data and overlaps the map and the reduce stages as much as possible.

Accuracy: Its sampling method should be able to derive a reasonably accurate estimate of the input data distribution by sampling only a small fraction of the data.

Total order: It should support total order of the output data. This saves applications which require such ordering an extra round of sorting at the end.

Large cluster splitting: When application semantics permit, it should be able to split data associated with a single large cluster to multiple reducers while preserving the consistency of the output.

Heterogeneity consideration: When the performance of the worker nodes is heterogeneous, it should be able to adjust the data partition accordingly so that all reducers finish around the same time.

With consideration to the parallelism, this approach makes the reduce tasks to wait until the partition decision is made. This has been improved by handling the LIBRA with a different approach.

III. PROPOSED SYSTEM

A. Data Analysis By Memory Loading 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.

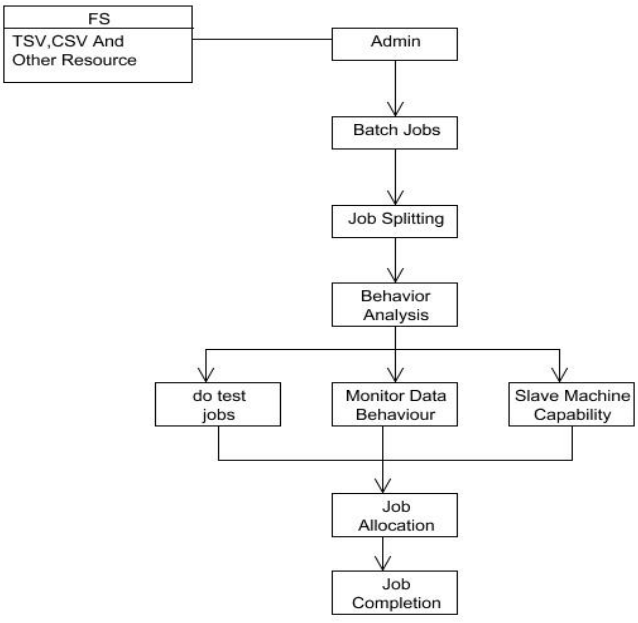

Fig. 2 System Architecure

The existing LIBRA takes a different approach by integrating sampling into a small percentage of the map tasks. It then prioritizes the execution of those sampling tasks over that of the normal map tasks: whenever the system has free slots, it launches the sampling tasks first. Since there are only a small percentage of them, they are likely to finish quite early in the map phase. There is an obvious trade-off between the sampling overhead and the accuracy of the result. But still the data skew occurs between the nodes in clusters as it is not aware the time factor and complexity accurately.

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 an optimal manner.

B. Web Crawling

The amount of web information is increasing rapidly with advanced wireless networks and emergence of diverse smart devices like i-Phone, i-Pad and so on. The information is continuously being produced and updated in anywhere and anytime by means of easy web platforms, and social networks. Now, it is becoming a hot issue how frequently updated web data has to be refreshed in data integration and retrieval domain. In this paper, we propose dynamic web-data crawling methods, which include sensitive checking of web site changes, and dynamic retrieving of web pages from target web sites. The World Wide Web is an interlinked collection of billions of documents formatted using HTML. Ironically the very size of this collection has become an obstacle for information retrieval. The user has to shift through scores of pages to come upon the information he/she desires. Web crawlers are the heart of search engines. Web crawlers continuously keep on crawling the web and find any new web pages that have been added to the web, pages that have been removed from the web. Due to growing and dynamic nature of the web; it has become a challenge to traverse all URLs in the web documents and to handle these URLs. A focused crawler is an agent that targets a particular topic and visits and gathers only relevant web pages.

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(Denial 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 (Comma Separated Values) format.

C. Query Processing Using Block Chain

Enterprises today acquire vast volumes of data from different sources and leverage this information by means of

data analysis to support effective decision-making and provide new functionality and services. The key requirement of data analytics is scalability, simply due to the immense volume of data that need to be extracted, processed, and analyzed in a timely fashion. The Query processing can be done using MapReduce 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 response time is calculated for Querying with MapReduce.

We also implement 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. Block chain proved to be as best as MapReduce and can be used in data intense results.

IV. CONCLUSIONS

The Data skew mitigation is important in improving MapReduce performance. So we designed and developed a Hadoop cluster which can mitigate Data Skew problem and we show its effectiveness using Block Chain algorithm and Map reduce Algorithm with comparison. Job execution , complexity are effectively managed in this paper. This paper also implement the 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.

REFERENCES

[1] Q. Chen, J. Yao, and Z. Xiao, "LIBRA: Lightweight Data Skew Mitigation in MapReduce," IEEE Transactions on Parallel and Distributed Systems (TPDS), vol. 26, September 2015.

[2] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," Commun. ACM, vol. 51, January 2008.

[3] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," in Proc. of the ACM SIGOPS/EuroSys European Conference on Computer Systems (EuroSys), 2007.

[4] Y. Kwon, M. Balazinska, and B. Howe, "A study of skew in mapreduce applications," in Proc. of the Open Cirrus Summit, 2011.

[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. Of the International Conference on Very Large Data Bases (VLDB), 1991.

[6] 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.

[7] J. W. Stamos and H. C. Young, "A symmetric fragment and replicate algorithm for distributed joins," IEEE Transactions on Parallel and Distributed Systems (TPDS), vol. 4, 1993.

[8] V. Poosala and Y. E. Ioannidis, "Estimation of query-result distribution and its application in parallel-join load balancing," in Proc. of the International Conference on Very Large Data Bases (VLDB), 1996.

[9] Y. Xu and P. Kostamaa, "Efficient outer join data skew handling in parallel dbms," Proc. of the VLDB Endowment, vol. 2, no. 2, 2009.

[10] S. Acharya, P. B. Gibbons, and V. Poosala, "Congressional samples for approximate answering of group-by queries," in Proc. of the ACMSIGMOD International Conference on Management of Data, 2000.

[11] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica,"Improving mapreduce performance in heterogeneous environments," in Proc. USENIX Conf. Oper. Syst. Des. Implementation, 2008.

[12] G. Benjamin, A. Nikolaus, R. Angelika, and K. Alfons, "Load balancing in mapreduce based on scalable cardinality estimates," in Proc. Int. Conf. Data Eng., 2012.

[13] Y. Kwon, M. Balazinska, B. Howe, and J. Rolia, "Skewtune: Mitigating skew in mapreduce applications," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2012.

[14] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," IEEE Trans. Parallel Distrib. Syst.), vol. 24, no. 6, pp. 1107–1117, Jun 2013.

[15] X. Zhou, M. E. Orlowska: Handling Data Skew in Parallel Hash Join Computation Using Two-Phase Scheduling. Proc. ICA3PP Conf., Brisbane, 1995.