International Journal of Advanced Information Science and Technology (IJAIST) *ISSN: 2319:2682*
Vol.10, No.06, June 2021

Hadoop Heterogeneous Environments: Analysis on Map Reduce Scheduling Algorithms

Valiki Vijayabhaser
Assoc. Prof.
Department of ECE
Siddhartha Institute of Technology and Sciences
Narapally, Hyderabad, Telangana, India

Immadisetty Venkata Prakash
Assoc. Prof.
Department of ECE
Siddhartha Institute of Technology and Sciences
Narapally, Hyderabad, Telangana, India

Abstract **- In the last few years, computer technology is maximized has resulted in a massive data flow (i.e., Big Data) that exceeds the capacity of standard processing methods. The processing demands of Big Data in real-time applications make achieving the appropriate levels of performance a significant task. Map Reduce is one of the most efficient parallel distributed programming frameworks for managing large, unstructured datasets in cloud applications. This Map Reduce methodology is implemented in Hadoop, an open source Java-based programming framework widely used in Big Data for large-scale data processing with fast response times. Hadoop is implemented in a homogeneous environment because it saves data transmission costs and ensures that each cluster node has the optimal computational performance and workload. In real-time applications, however, processing nodes may have specialized computing capabilities and workloads that emerge in a heterogeneous environment. In this diverse context, the standard Hadoop implementation fails to deliver the requisite performance.**

Keywords: Big Data, cloud Applications, Hadoop, Java, Map Reduce

1. INTRODUCTION

Most popular real-time applications have become data-intensive in recent years because of its increased growth in data exponentially. The act of storing, processing, and analyzing such large amounts of data is known as Big Data, and it has become a key responsibility in recent years. The World Wide Web (WWWW) has been chosen as a great platform for developing data-intensive real-time applications in recent decades since the web's communication prototype is sufficiently open and dynamic. Data mining, online analytics, and web indexing are examples of data-intensive applications. that are required to handle ever-growing data volumes ranging from a few gigabytes to a few terabytes or even petabytes. One of the most well-known examples is Google, which uses its Map Reduce structure to process 20 petabytes of data per day. In high-performance and large-scale cluster computing systems, Map Reduce is seen as an appealing programming model for effective parallel data processing prototype. Map Reduce runs on a large cluster of product machines and provides fault tolerance that is transparent to programmers [1]. Map Reduce is implemented in Hadoop, a well-known open-source framework [2] designed primarily by Yahoo Inc, which runs tasks yielding several terabytes of data on no less than 10,000 cores [3]. Amazon and Facebook [4] are two more companies that use Hadoop. The Map Reduce model runs on a huge cluster with homogeneous cluster nodes and takes a homogeneous workload into account when making scheduling decisions. Map Reduce is concerned with the intricacies of partitioning input data, fault tolerance, programme scheduling among a group of processors, and handling expected communication between two machines (i.e., inter-machine communication). Map Reduce's performance is based on previous features that are visible in a homogeneous setting. As a result, using the Map Reduce programming model in a heterogeneous environment becomes critical, as the Map Reduce algorithm's execution is influenced by heterogeneity. Several academics [5-9] have looked into the performance degradation of Map Reduce in heterogeneous contexts and offered different algorithms to improve the Map Reduce algorithm's performance. In this study, we examine the strategies for improving the performance of the Map Reduce model in diverse contexts, as well as their relative benefits and drawbacks.

The remainder of this work is structured as follows: The second portion discusses the background, including an overview of Map Reduce algorithms, Hadoop, HDFS, Hadoop Map Reduce, and Map Reduce Scheduling Issues. Table 1 lists the pros and disadvantages of several Map Reduce scheduling algorithms outlined in section 3, as well as their taxonomy. Section 4 discussed the performance of adaptive and non-adaptive scheduling algorithms, while section 5 concluded the analysis.

2. BACKGROUND

2.1. Overview of MapReduce techniques

The Map Reduce methodology [1] was created by Google to run real-time data-intensive applications in a distributed framework like a commodity cluster. The map and reduce primitives demonstrated in Lisp and various other useful languages inspired the Map Reduce concept [1]. Map Reduce allows developers who aren't familiar with distributed programming to create Map Reduce functions that run in parallel across multiple nodes in the cluster by identifying two key functions, one of which is the map function, which executes key/value pairs to produce a set of transitional key/value pairs. The reduction function, on the other hand, consolidates all intermediate measures associated with the relevant intermediate key. Map and reduce functions are executed in parallel by Map Reduce at each cluster node. Because the MapReduce programming model can assist with a few operations like grouping and sorting on a group of key/value pairs, programmers must develop map and reduce functions. Map Reduce is a simple programming approach since programmers just need to focus on data processing operations rather than parallelism aspects.

2.2. Hadoop

Hadoop, an open source software framework supported by the Apache Software Foundation, implements the Map Reduce methodology [10]. Hadoop is broken down into two main components: Hadoop Map Reduce and Hadoop Distributed File System (HDFS). Map Reduce and HDFS, respectively, perform parallel processing and data management. Jobs are divided into tasks, which are then processed in parallel using Map Reduce. The HDFS divides the stored data into blocks that it manages. Cluster nodes are assigned to those data blocks and jobs. Hadoop adopts a master/slave architecture, with the master and slave being referred to as Job Tracker and Task Tracker, respectively. Job Tracker handles work distribution and job scheduling, whereas Task Tracker handles the actual tasks and returns the results to Job Tracker. It makes use of heartbeat messages for communication. When using Hadoop, high-performance computing does not necessitate the use of higher-end processors. A few ordinary machines can be used in conjunction with Hadoop to create a high-performance platform that saves a significant amount of money.

2.3 HDFS

HDFS is based entirely on the Google File System (GFS), an open source file system designed to run on inexpensive hardware. In comparison to other Hadoop technologies, HDFS has emerged as a critical tool for managing large data pools and defending big data analytics applications. It's also designed to be implemented on low-cost technology and is highly fault-tolerant. The Master/Slaver design of HDFS is shown in Fig 1 and consists of a Master Name Node, a Secondary Name Node known as checkpoint, and a few Data Nodes known as slaves. Any requests that arrive in the file system, such as file creation, deletion, and read, pass through the controller named Name Node. The meta-data for access times, licenses, modifications, and disc space allotment is stored in the Name Node. The Name Node is also responsible for block mappings. The file is divided into blocks, each of which has a default size of 64 MB and is freely replicated throughout Data Node to ensure redundancy, as well as sending a report of each current block to the Name Node on a regular basis. The Data Node is in charge of creating, cancelling, and replicating blocks based on the Name Node's instructions. Given that each Data Node can conduct many application jobs at the same time, each cluster may have hundreds of Data Nodes and thousands of HDFS clients. In addition, the Name Node receives a Heartbeat message from the Data Node on a regular basis, with a 3.5s interval by default. If the Data Node and Name Node lose contact, the Name Node will be unable to detect heartbeat messages.

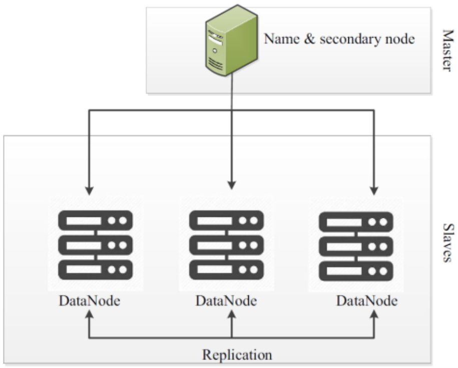

Fig. 1 HDFS architecture

The Data Node hosts the block replicas that are unreachable or dead, and that specific Data Node does not receive any new requests since the Name Node is out of service. The Name Node creates a schedule that includes new replicas of the above-mentioned blocks on separate Data Nodes. At the same time, the Name Node does not see the job of the secondary Name Node as secondary, but rather reads changes in the file system on a regular basis and renders backups for earlier files, completing the updating process. When the cluster environment is large, the secondary Name Node is usually run on a different machine than the primary Name Node and has the same memory needs. Name Node will be able to initiate faster the following time because of this process.

2.4. Hadoop Map Reduce

The Map Reduce engine is depicted in Figure 2 as a collection of components, with the job client serving as the central component that submits the job to the network cluster. The job tracker regulates the task tracker by giving execution plans, organizing the jobs, and scheduling them throughout the task tracker. The task tracker breaks down the jobs into Map and Reduce tasks at the same time. Every task record includes slot execution maps. It reduces and reports the execution progress on a regular basis. All input data is divided into input splits based on the format of the input. The input splits are used to equalize the map jobs, which are conducted in parallel. The way the documents are parsed into the Map Reduce pipeline is determined by the input design. The map transforms input splits into intermediate key/value pairs based on user-defined code. The output of the intermediate key/value pairs is transported to the reducers and then sorted through the key in a shuffle and sort operation. The reducer joins all pairs of related objects that have the same intermediate key and generates an output based on the user-defined code.

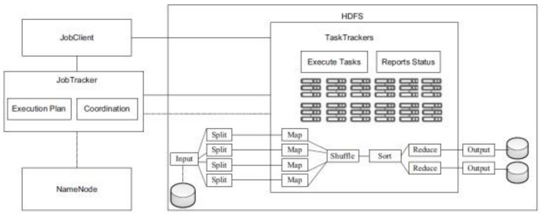

Fig. 2 MapReduce architecture

3. SCHEDULING ALGORITHMS

Because of the challenges discussed in the preceding section, scheduling is regarded as the most important part of Map Reduce. There are various algorithms that address these concerns by recommending various approaches and methodologies, which are mentioned further below.

3.1. Hadoop Default scheduling algorithm

The default scheduler in Hadoop is First in First Out (FIFO), which operates in the order of first-come, first-serve, i.e., if the Job Tracker drags the oldest job from the job queue, it ignores its priority or size [11]. A work is divided into distinct tasks, which are then placed into a job queue and assigned to available slots on Task Tracker. Support for priority assignment of jobs that aren't done by default is required. In most cases, every job would use the entire cluster, thus jobs would have to wait their time. Despite the fact that a shared cluster has tremendous potential for providing large amounts of resources to a large number of users, the issue of efficiently sharing resources among users necessitates the employment of a superior scheduler. When allowing users who are causing lower ad hoc inquiries to acquire outputs in an average time, production jobs must be completed in a timely manner.

4. DISCUSSION AND ANALYSIS

In this work, benefits, demerits, and taxonomy of scheduling algorithms are discussed from several literature [12 - 17]. These are clearly listed in Table 1, where the taxonomy describes the runtime flexibility of algorithms in two categories: adoptive and non-adoptive. When making a decision, the adaptive scheduling algorithm uses the past, current, and future parameter values. Non adaptive scheduling algorithms do not take into account changes in the environment while scheduling tasks, regardless of whether they are performed according to policy or not. The algorithm is implemented using several ideas in order to achieve its goal, but it is still unable to reach the goal. The disadvantage column represents the algorithm performance with null values as a result. According to the suggested scheduling algorithm and the results of these papers [18-22] [31], [33], [34], [37], [38], [39], we can conclude that the proposed scheduling algorithm is devoid of flaws that reduce overall performance. These algorithms are useful for solving one or two difficulties, but they are inefficient for achieving all of our goals.

5. CONLUSION

We explored several Hadoop Map Reduce difficulties, as well as their overall scheduling tasks, in this study. We examine eleven well-known scheduling algorithms in terms of their areas in this research. Every algorithm was discussed in terms of its taxonomy, benefits, and drawbacks. The majority of the methods described in this study were only applicable to one or two problems. The user determines how a particular job is scheduled, and no algorithm can meet all of our needs. We used Map Reduce in a heterogeneous setting, together with COSHH and SAMR algorithms, to improve overall performance. It reduces network traffic and Map Reduce network traffic, making CREST one of its best works.

6.REFERENCES

[1] J. Dean and S. Ghemawat. Map reduce: simplified data processing on large clusters. In Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation - Volume 6, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[2] Hadoop, http://lucene.apache.org/hadoop, (last view June 30, 2012).

[3]Yahoo! launches world's largest hadoop production application, http://developer.yahoo.com/blogs/hadoop/posts/2008/02/yahoo-worlds- largest-production-hadoop/, (last view June 30, 2012).

[4]Applications powered by Hadoop, http://wiki.apache.org/hadoop/PoweredBy, (last view June 30, 2012).

[5] M.Zaharia, A.Konwinski, A.Joseph, Y.zatz, and I.Stoica. Improving mapre- duce performance in heterogeneous environments. In OSDI'08: 8th USENIX Symposium on Operating Systems Design and Implementation, October 2008.

[6] J.Xie, S.Yin, X.Ruan, Z.Ding, Y.Tian, J.Majors, A.Manzanares, and X.Qin. Improving MapReduce Performance through Data Placement in Heteroge- neous Hadoop Clusters. In proceedings of IEEE International parallel and dis- tributed Processing Symposium, 2010.

[7] Q.Chen, D.Zhang, M.Guo, Q.Deng and S.Guo. SAMR: A Self-adaptive MapReduce Scheduling Algorithm In Heterogeneous Environment. In Pro- ceedings of IEEE 10th International Conference on Computer and Infor- mation Technology, 2010.

[8] X.Zhang, Y.Feng, S.Feng, J.Fan and Z.Ming. An Effective Data Locality Aware Task Scheduling Method for MapReduce Framework in Heterogeneous En- vironments. In International Conference on Cloud and Service Computing, 2011.

[9] Z.Guo, G.Fox. Improving MapReduce Performance in Heterogeneous Net-work Environments and Resource Utilization. In proceeding of IEEE/ACM 12th International Symposium on Cluster, Cloud and Grid Computing, 2012.

[10] Shvachko, K., Kuang, H., Radia, S., &Chansler, R. (2010). The hadoop distributed file system. Paper presented at the 2010 IEEE 26th symposium on mass storage systems and technologies (MSST).

[11]Hadoop, "Hadoop home page." http://hadoop.apache.org/.

[12] M. Zaharia, A. Konwinski, A.D. Joseph, R. Katz and I. Stoica, "Improving MapReduce performance in heterogeneous environments " In: OSDI 2008: 8th USENIX Symposium on Operating Systems Design and Implementation 2008.

[13] B.P Andrews and A. Binu, " Analysis on Job Schedulers in Hadoop Cluster ", IOSR Journal of Computer Engineering, Vol.15, NO. 1, Sep. - Oct. 2013, pp. 46-50.

[14]Hadoop'sFairScheduler.https://hadoop.apache.org/docs/r1.2.1/fair_sche duler.

[15] The Apache Hadoop Project. http://www.hadoop.org.

[16] Y. Tao Y, Q. Zhang, L. Shi and P. Chen, " Job scheduling optimization for multi-user MapReduce clusters ", In: The fourth international symposium on parallel architectures, algorithms and programming. IEEE; 2011. p. 213–17.

[17] Q. Chen, D. Zhang, M. Guo, Q. Deng Q and S. Guo, " SAMR: a self-adaptive MapReduce scheduling algorithm in heterogeneous environment " , In: The 10th international conference on computer and information technology. IEEE; 2010. p. 2736–43.

[18] S. Khalil, S.A. Salem, S. Nassar and E.M. Saad, " Mapreduce Performance in Heterogeneous Environments: A Review ", International Journal of Scientific & Engineering Research, Vol. 4, NO. 4, April - 2013, pp. 410-416.

[19] A. Rasooli and D.G. Down, " COSHH: A classification and optimization based scheduler for heterogeneous Hadoop systems", Future Generation Computer Systems, 36, 2014, pp. 1-15.

[20] M. Zaharia, D. Borthakur, J.S. Sarma, K. Elmeleegy, S. Shenker and I. Stoica, " Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling", In: Proceedings of the fifth European conference on computer systems. New York, NY, USA: ACM; 2010, p. 265–278.

[21] A. N Nandakumar and Y. Nandita, " AAnalysis on Data Mining Algorithms on Apache Hadoop Platform", International Journal of Emerging Technology and Advanced Engineering, Vol. 4, NO. 1, January 2014, pp. 563-566.
[22] Z. Tang, L. Jiang, J. Zhou, K. Li, and K. Li, " A self-adaptive scheduling algorithm for reduce start time ", Future Generation Computer Systems, 2014.