International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue14
November 2017

A Survey On Mapreduce Calculations In Virutalized Environment

Daggupati Saidamma
Assistant Professor In Cse Department Geetanjali College Of Engineering And Technology
daggupatisaida@gmail.com

ABSTRACT: *The MapReduce is an open source Hadoop framework implemented for processing and producing distributed large Terabyte data on large clusters. Its primary duty is to minimize the completion time of large sets of MapReduce jobs. Hadoop Cluster only has predefined fixed slot configuration for cluster lifetime. This fixed slot configuration may produce long completion time (Makespan) and low system resource utilization. The current open source Hadoop allows only static slot configuration, like fixed numbers of map slots and reduce slots throughout the cluster lifetime. Such static configuration may lead to long completion length as well as low system resource utilizations. Propose new schemes which use slot ratio between map and reduce tasks as a tunable knob for minimizing the completion length (i.e., makespan) of a given set. By leveraging the workload information of recently completed jobs, schemes dynamically allocates resources (or slots) to map and reduce tasks. Our survey paper emphasizes the state of the art in improving the performance of various applications using recent MapReduce models and how it is useful to process large scale dataset. A comparative study of given models corresponds to Apache Hadoop and Phoenix will be discussed primarily based on execution time and fault tolerance. At the end, a high-level discussion will be done about the enhancement of the MapReduce computation in specific problem area such as Iterative computation, continuous query processing, hybrid database etc.*

Index Terms—*Big Data, - Hadoop, MapReduce, Scheduling, Distributed data processing.*

I. INTRODUCTION

In the present day circumstance, cloud has transformed into an unavoidable need for prevailing piece of IT operational affiliations. Applications, for instance, data accumulating, data recuperation and data conveyability have ended up being basic essentials for dispersed processing. Conveyed registering application handles BigData to beat these preventions. A BigData is enormously far reaching educational accumulations that has unstructured, semi-sorted out and composed data where the data can be burrowed for information. Different applications are being delivered to achieve higher execution. Capable load changing, stack dissemination, perfect resource use, minimum overheads and scarcest possible deferral have been the essential issues for cloud establishment. Hadoop is an open source structure for spread data accumulating and getting ready tremendous data on group of item gear. Customers submit occupations to a line, the gathering system them in the demand of jobs submitted. As the data augments hugely, count to those occupations will similarly increase. The extension in figuring will give us the component like sharing Hadoop aggregate among various customers. Favorable circumstances of sharing Hadoop amass manufactures the data region and besides cost profitable. Hadoop at first made by Google and later made an open source execution by Apache. The Hadoop structure empowers planners to focus on count with parallel dealing with programs for immense data. Hadoop fuses two levels of setting i up) Hadoop Distributed File System (HDFS), which stores enormous measure of data with reduced makespan ii) Hadoop MapReduce: a structure which shapes appropriated data on gatherings. Hadoop is a trustworthy, versatile, passed on figuring stage made by Apache. It offers adjacent figuring and limit which is planned to level up from single datacenter to countless. Hadoop has a scattered amassing structure called Hadoop Distributed File System (HDFS). Hadoop gives a MapReduce engine that continues running over HDFS to process the data. HDFS framework allotments tremendous instructive accumulations into data knots. This structure depends on pro slave outline. The expert center points are known as the NameNode and JobTracker. The slave center points are called DataNode and TaskTrackers. NameNode, HDFS has an expert center point called NameNode which controls slave center points called DataNode. NameNode has every one of the information of where the data is secured, how data is separated into impedes, on which center point the data is put and the prosperity of the scattered record structure. The NameNode is a lone motivation behind dissatisfaction, where each gathering has NameNode. DataNode is a slave center in the Hadoop cluster. It is accountable for data organization and scrutinizes its data from HDFS. DataNode reports back to the NameNode with data organization status and directs data on each physical center point. The client application submits vocations to the JobTracker which in this manner banters with the NameNode to choose the territory of the data. The JobTracker finds available openings in the TaskTracker at or near the data. The movement is submitted to the JobTracker

to the picked TaskTracker and watched. The JobTracker and TaskTrackers passes on using beat signal. In case the beat signal isn't gotten in a period break, they are relied upon to have failed and the work is set up for a substitute TaskTracker. The JobTracker revives its status once the work is done and is submitted to the client application. In this investigation paper, we focus on different sorts of latest arranging frameworks to give us a gainful result with less makespan. Scheduler in Hadoop displayed two schedulers for multi customer condition. Sensible scheduler made at Facebook and Capacity scheduler made at Yahoo.

II. MOTIVATION

In this part examine the vital attributes of my proposed calculation, in view of the difficulties of the Hadoop framework.

1. SCHEDULING BASED ON FAIRNESS, MINIMUM SHARE REQUIREMENTS, AND THE HETEROGENEITY OF JOBS AND RESOURCES.

In a Hadoop framework fulfilling the base offers of the clients is the main basic issue. The following vital issue is reasonableness. I outline a planning calculation which has two phases. In the principal arrange, the calculation considers the fulfillment of the base offer prerequisites of the considerable number of clients. At that point, in the second stage, the calculation considers reasonableness among every one of the clients in the framework. Most present Hadoop planning calculations consider reasonableness and least offer destinations without considering the heterogeneity of the employments and the assets. One of the benefits of my proposed calculation is that while our proposed calculation fulfills the decency and the base offer necessities, it additionally coordinates occupations with assets in view of employment highlights (e.g. evaluated execution time) and asset highlights (e.g. execution rate). Subsequently, the calculation decreases the fulfillment time of employments in the framework.

2. DIMINISHING COMMUNICATION COSTS.

In a Hadoop bunch, the system joins among the assets have differing data transmission capacities. In addition, in a substantial bunch, the assets are regularly situated a long way from each other. The Hadoop framework disperses assignments among the assets to decrease an occupation's fulfillment time. In any case, Hadoop does not consider the correspondence costs among the assets. In a huge group with heterogenous assets, amplifying an errand's dispersion may bring about noteworthy correspondence costs. In this way, the comparing employment's finish time will be expanded. In our proposed calculation, we consider the heterogeneity and dissemination of assets in the undertaking task.

III. MAP REDUCE CLASSIFICATION

Map Reduce data interpretive applications are requested on the start of their abilities [8]:

A. Clustering Based Algorithm:

These computations are memory unstable as pack based figuring required a considerable measure of limit. To gage the parameter estimations of different packs, a tremendous count impacts it to enlist raised strategy. for eg. K-suggests, Fuzzy K-infers, shade batching et cetera.

B. Classification Algorithm:

This figuring tackles a planning set and question set to process k nearest regards which required a satisfactory memory space to store the data. It is also enlist genuine methodology in light of the way that a vector thing is done to figure the closeness between two vectors. For eg. K-nearest neighbor et cetera Author [9] separated the different instrument to improve the memory use on the multi-focus machine for MapReduce. Maker had in like manner researched three given applications with respect to profitable memory utilize. 1) Hash Join-It is a variety of impart join by Blanas et al [10]. In the join operation, simply Map work is used to join two tables i.e. data table (S) and reference table (R). A hash join isn't enroll concentrated application and its shot multifaceted nature is O(|S|). 2) KMeans-K-suggests application is used to divide game plan of n test objects into K bunches for input parameter K. This figuring is memory genuine and process heightened which in this way obliging the amount of gatherings K-means can create. The time diserse quality is O(|n|*|k|). 3) K-nearest neighbors-K-nearest neighbors is a request estimation that uses a tremendous in-memory instructive list. KNN strategy uses two educational lists, a request set Q and a planning set T. It picks K storage space segments in T in light of a prepared division between data centers in the two sets. The time unusualness of the procedure is O(|Q|*|T|) in light of the fact that it figures the detachment between each point in Q and in T. So the KNN is figure concentrated and also memory heightened application.

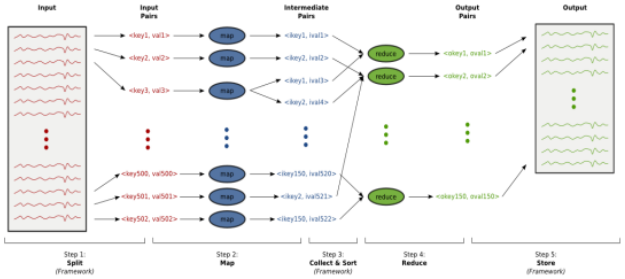

Fig. 1. Map Reduce framework

Initially the paper describes the MapReduce classification as well as an introductory explanation of its applications such as distributed pattern based searching, geospatial query processing, web link graph traversal, distributed sort, machine learning applications etc. The primary focus of this survey paper is to highlight some MapReduce implementation worked well to accomplish a specific purpose and compared with previously available frameworks. A remarkable performance improvement over the existing system seems after comparison. Later we discussed the recent enhancements which help to solve the issues related to iterative computation, efficient continuous queries execution and hybrid database.

IV. MAP REDUCE APPLICATIONS

Map Reduce implementation is used in various data intensive computation because of the functionality of parallel processing of massive data. A short introduction of related applications is given below:

A. Distributed Pattern Based Searching

Distributed grep command is used to search a pattern in the given text distributed over a network. Here map function searches for the pattern and produces the output so no intermediate result writes. Hence reduce function is just copied the intermediate result to output in distributed pattern searching [1].

Example: A big data of medical health record is analyzed using parallelization and pattern searching property of MapReduce taking into consideration [11]:

1) Public dataset- It consists of various reports of patients from US Food and Drug administration.

2) Biometric Datasets- It is having human characteristics like images [12].

3) Bioinformatics Signal Datasets- This dataset represents the recording of vital signs of a patient. e.g. Electrocardiography ECG

4) Biomedical Image Datasets- A dataset having a collection of scanning of medical images such as ultrasound images.

B. Geospatial Query Processing

With the technological advancement in location based service, MapReduce helps to find out the shortest path in Google map for a given location. Here Map function searches all connecting paths from source to destination with distance value. After sorting the keys, the Reduce function emits the path which is of shortest distance. An algorithm LoNARS [13] has implemented to improve Reduce task scheduling by considering data locality and network traffic. Even author achieved 15% gain in data shuffling time and up to 3-4% improvement in job completion time.

C. Distributed Sort

Distributed sort is used to arrange the data in sorted manner split across multiple sites. In Map Reduce implementation, initially input data is given to map function to convert it into intermediate data which is stored in a local disk buffer. In next step, data is transmitted to the appropriate reducer function over the network. A number of reduce functions sort the data according to given key value and writes the output [14].

Author represents massive data sorting using Apache Hadoop open source software framework with the help of three map reduce functions [15]:

Teragen: used to generate input data to be sort.

Terasort: Sample the input data and used them with Map Reduce to sort the data.

Teravalidate: At last sorted output data is validated.

This method is I/O intensive as it works on data input/output.

V. SCHEDULER IMPROVEMENTS

Many researchers are working on opportunities for improving the scheduling policies in Hadoop. Recent efforts such as Delay Scheduler [9], Dynamic Proportional Scheduler [10] offer differentiated service for Hadoop jobs allowing users to adjust the priority levels assigned to their jobs. However, this does not guarantee that the job will be completed by a specific deadline. Deadline Constraint Scheduler [11] addresses the issue of deadlines but focuses more on increasing system utilization. The Schedulers described above attempt to allocate capacity fairly among users and jobs, they make no attempt to consider resource availability on a more fine-grained basis. Resource Aware Scheduler [12] considers the resource availability to schedule jobs. In the following sections we compare and contrast the work done by the researchers on various Schedulers.

5.1 Longest Approximate Time to End (LATE) - Speculative Execution

It is not uncommon for a particular task to continue to progress slowly. This may be due to several reasons like– high CPU load on the node, slow background processes etc. All tasks should be finished for completion of the entire job. The scheduler tries to detect a slow running task to launch another equivalent task as a backup which is termed as speculative execution of tasks. If the backup copy completes faster, the overall job performance is improved. Speculative execution is an optimization but not a feature to ensure reliability of jobs. If bugs cause a task to hang or slow down then speculative execution is not a solution, since the same bugs are likely to affect the speculative task also. Bugs

should be fixed so that the task doesn't hang or slow down. The default implementation of speculative execution relies implicitly on certain assumptions: a) Uniform Task progress on nodes b) Uniform computation at all nodes. That is, default implementation of speculative execution works well on homogeneous clusters. These assumptions break down very easily in the heterogeneous clusters that are found in real-world production scenarios. Zaharia et al [13] proposed a modified version of speculative execution called Longest Approximate Time to End (LATE) algorithm that uses a different metric to schedule tasks for speculative execution. Instead of considering the progress made by a task so far, they compute the estimated time remaining, which gives a more clear assessment of a straggling tasks' impact on the overall job response time. They demonstrated significant improvements by Longest Approximate Time to End (LATE) algorithm over the default speculative execution.

5.2 Delay Scheduling

Fair scheduler is developed to allocate fair share of capacity to all the users. Two locality problems identified when fair sharing is followed are – head-of-line scheduling and sticky slots. The first locality problem occurs in small jobs (jobs that have small input files and hence have a small number of data blocks to read). The problem is that whenever a job reaches the head of the sorted list for scheduling, one of its tasks is launched on the next slot that becomes free irrespective of which node this slot is on. If the head-of-line job is small, it is unlikely to have data locally on the node that is given to it. Head-of-line scheduling problem was observed at Facebook in a version of HFS without delay scheduling. The other locality problem, sticky slots, is that there is a tendency for a job to be assigned the same slot repeatedly. The problems aroused because following a strict queuing order forces a job with no local data to be scheduled. To overcome the Head of line problem, scheduler launches a task from a job on a node without local data to maintain fairness, but violates the main objective of MapReduce that schedule tasks near their input data. Running on a node that contains the data (node locality) is most efficient, but when this is not possible, running on a node on the same rack (rack locality) is faster than running off-rack. Delay scheduling is a solution that temporarily relaxes fairness to improve locality by asking jobs to wait for a scheduling opportunity on a node with local data. When a node requests a task, if the head-of-line job cannot launch a local task, it is skipped and looked at subsequent jobs. However, if a job has been skipped long enough, non-local tasks are allowed to launch to avoid starvation. The key insight behind delay scheduling is that although the first slot we consider giving to a job is unlikely to have data for it, tasks finish so quickly that some slot with data for it will free up in the next few seconds.

5.3 Dynamic Priority Scheduling

Thomas Sandholm et al [10] proposed Dynamic Priority Scheduler that supports capacity distribution dynamically among concurrent users based on priorities of the users. Automated capacity allocation and redistribution is supported in a regulated task slot resource market. This approach allows users to get Map or Reduce slot on a proportional share basis per time unit. These time slots can be configured and called as allocation interval. It is typically set to somewhere between 10 seconds and 1 minute. For example a max capacity of 15 Map slots gets allocated proportionally to three users. The central scheduler contains a Dynamic Priority Allocator and a Priority Enforcer component responsible for accounting and schedule enforcement respectively. This model appears to favor users with small jobs than users with bigger jobs. However Hadoop MapReduce supports scaling down of big jobs to small jobs to make sure that fewer concurrent tasks runs by consuming the same amount of resources.

5.4 Deadline Constraint Scheduler

Deadline Constraint Scheduler [11] addresses the issue of deadlines but focuses more on increasing system utilization. Dealing with deadline requirements in Hadoop-based data processing is done by (1) a job execution cost model that considers various parameters like map and reduce runtimes, input data sizes, data distribution, etc., (2) a Constraint-Based Hadoop Scheduler that takes user deadlines as part of its input. Estimation model determines the available slot based a set of assumptions: All nodes are homogeneous nodes and unit cost of processing for each map or reduce node is equal Input data is distributed uniform manner such that each reduce node gets equal amount of reduce data to process Reduce tasks starts after all map tasks have completed; The input data is already available in HDFS.

5.5 Resource Aware Scheduling

The Fair Scheduler [7] and Capacity Scheduler described above attempt to allocate capacity fairly among users and jobs without considering resource availability on a more fine-grained basis. As CPU and disk channel capacity has been increasing in recent years, a Hadoop cluster with heterogeneous nodes could exhibit significant diversity in processing power and disk access speed among nodes. Performance could be affected if multiple processor-intensive or data-intensive tasks are allocated onto nodes with slow processors or disk channels respectively. This possibility arises as the Job Tracker simply treats each Task Tracker node as having a number of available task "slots". Even the improved LATE speculative execution could end up increasing the degree of congestion within a busy cluster, if speculative copies are simply assigned to machines that are already close to maximum resource utilization.

VI. CONCLUSIONS AND FUTURE WORK

Ability to make Hadoop scheduler resource aware is one the emerging research problem that grabs the attention of most of the researchers as the current implementation is based on statically configured slots. This paper summarizes pros and cons of Scheduling policies of various Hadoop Schedulers developed by different communities. Each of the Scheduler considers the resources like CPU, Memory, Job deadlines and IO etc. All the schedulers discussed in this paper addresses one or more problem(s) in scheduling in Hadoop. Nevertheless all the schedulers discussed above assumes homogeneous Hadoop clusters. Future work will consider scheduling in Hadoop in Heterogeneous Clusters.

Map Reduce was initiated by Google to handle big data analysis which is unstructured data such as web document. We have discussed a number of Map Reduce models still researchers can develop a more efficient Map Reduce with improved functionalities. Similarly a new user friendly data processing language can be introduced to make data handling easier

REFERENCES

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters" In ACM OSDI, 2004.
[2] A. Elsayed, O. Ismail, and M. E. El-Sharkawi, MapReduce: State-of-theArt and Research Directions, International Journal of Computer and Electrical Engineering, Vol. 6, No. 1, February 2014.
[3] "Hadoop," available at http://hadoop.apache.org/.
[4] M. Zaharia, M. Chowdhury, Michael J. Franklin, Scott Shenker and Ion Stoica, Spark: Cluster Computing with Working Sets University of California, Berkeley, May, 2010.
[5] D. Moors, Whitehound Limited, UK, "SASReduce An implementation of MapReduce in BASE/SAS", Paper 1507-2014
[6] E. Bugnion, S. Devine, K. Govil, and M. Rosenblum, Disco: Running Commodity Operating Systems on Scalable Multiprocessors (1997).
[7] J. Dean and S. Ghemawat, MapReduce: A flexible data processing tool, Communications of the ACM, Vol. 53 No. 1, Pages 72-77 10.1145/1629175.1629198
[8] K. Ericson and S. Pallickara, On the Performance of High Dimensional Data Clustering and Classification Algorithms (2013).
[9] Y. Zhang, "Optimized Runtime Systems for MapReduce applications in Multi-core clusters", A thesis in Houston Texas, May 2014.
[10] S. Blanas, J. M. Patel, V. Ercegovac, J. Rao, E. J. Shekita, and Y. Tian, "A Comparison of Join Algorithms for Log Processing in MapReduce," in Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, SIGMOD "10, (New York, NY, USA), pp. 975–986, ACM, 2010.
[11] E. A Mohammed, B. H Far and C. Naugler, Application of the MapReduce programming framework to clinical big data analysis:Current landscape and future trends, BioData Mining. Vol. 7, 22. Oct 29, 2014.
[12] M Jonas, S Solangasenathirajan and D Hett. Annual Update in Intensive Care and Emergency Medicine 2014. New York – USA: Springer. Patient Identification, A Review of the Use of Biometrics in the ICU; pp. 679–688. (2014).
[13] E. Arslan, M. Shekhar & T. Kosar, "Locality and Network-aware reduce task scheduling for data intensive applications", published in Proceedings DataCloud"14 Proceedings of the 5th International workshop on Data Intensive Computing in the Clouds, page 17-24, ISBN:978-1-4799-7034-6
[14] D. Gillick, A. Faria and J. DeNero, "Map Reduce: Distributed Computing and Machine Learning", Dec-2006
[15] Owen O"Malley, "TeraByte Sort on Apache Hadoop", Yahoo! owen@yahoo-inc.com May 2008.
[16] S. Sakr, Processing Large Scale Graph Data: A Guide to Current Technology, National ICT Australia, June 2013.
[17] U Kang et al., GBASE: A Scalable and General Graph Management System, San Diego, California, U.S.A. ACM978-1-4503-0813-7/11/08. Aug-2011.
[18] J. Dean, "Experience with MapReduce, An Abstraction for Large Scale Computation", Google, Inc., proceedings of the 15th international conference on parallel architecture and compilation techniques, ACM New york, US, ISBN:1-59593-264-X doi>10.1145/1152154.1152155
[19] S. Chen and S. W. Schlosser,"Map reduce meets wider varieties of applications", IPR-TR-08-05, Research at Intel (2008).
[20] W. Zhao, H. Ma & Q. He, "Parallel K-means clustering based on mapreduce", cloudcom, LNCS 5931, pp 674-679 © springer-verlag Berlin Heidelberg 2009.
[21] J. Talbot, R. M. Yoo, and C. Kozyrakis, "Phoenix++: Modular mapreduce for shared-memory systems," In Proc. of the second international workshop on MapReduce and its applications, pp. 9–16, 2011.
[22] C. Cao, F. Song, D. G. Waddington, "Implementing a high performance recommendation system using Phoenix++", In Proc. of Internet Technology and Secured Transactions, 8th International Conference for, DOI 10.1109/ICITST.2013.6750200, pages 252-257, dec 2013.
[23] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis. Evaluating MapReduce for multi-core and multiprocessor systems. In Proc. of the 13th Int"l Symposium on High Performance Computer Architecture, pages 13–24, 2007

[24] R. M. Yoo, A. Romano, and C. Kozyrakis. Phoenix rebirth: Scalable MapReduce on a large-scale shared-memory system. In Proc. of the 2009 IEEE Int‟l Symposium on Workload Characterization, pages 198– 207, 2009.
[25] L. Lu, H. Jim, X. Shi, Fedak G.,"Assessing MapReduce for Internet Computing: A Comparison of Hadoop and BitDew-MapReduce", In Proc. of the Grid Computing (GRID), ACM/IEEE 13th International Conference on Grid Computing, DOI:10.1109/Grid.2012.31, ISBN: 978-0-7695-4815-9, pp. 76-84, Sept 2012.
[26] H. Karloff, S. Suri and S. Vassilvitskii, A Model of Computation for MapReduce, at AT & T Labs and Yahoo! Research (2010).
[27] S. Wu, Y. Peng, H. Jin, J. Zhang,"Peacock: a customizable Map Reduce for Multicore plateform, In Proc. of the Journal of Supercomputing, DOI 10.1007/s11227-014-1238-2, Springer Science + Business Media New York pages:1496-1513, June 2014
[28] Li, F., Ooi, B-C., Özsu, M. T., Wu, S., Distributed Data Management Using MapReduce. ACM Comput. Surv. 0, 0, Article A (0), 41 pages. DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000 (2013).
[29] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein,"MapReduce online" in UC Berkeley, 2010.
[30] G. D. F. Morales, A. Gionis and M. Sozio. Social Content Matching in MapReduce. 37th International conference on very large databases, Seattle Washington, Proceedings of the VLDB Endowment, Vol. 4, No. 7 Copyright 2011 VLDB Endowment 2150-8097/11/04.
[31] A. V. Goldberg and S. Rao. "Beyond the flow decomposition barrier", JACM, 45(5):783–797, 1998.