www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 6 Issue 9 September 2017, Page No. 22457-22462
Index Copernicus value (2015): 58.10 DOI: 10.18535/ijecs/v6i9.11

Review on various scheduler used by Hadoop framework

Usha Rani[1], Anurag Rana, Sahil Barjtya[3]

Arni University Dept. of Computer Science Engineering
Kangra Himachal Pradesh, India

Abstract
A distributed system is a collection of independent computers that appears to its user a single coherent system. This definition can explain several important aspects. The first fact is that a distributed system is the collection of different types of component's for example computer, networking devices, storage, printers etc. In this paper we presented a detailed review of all the scheduling technique used by Hadoop framework this paper provides you deep insight of these scheduler working.

Keywords — Hadoop, HDFS, MapReduce, FIFO scheduler. Fair Scheduler

Introduction

Hadoop is a distributed computing architecture based upon the open source implementation of Google's MapReduce which supports processing of huge amount of data sets across multiple distributed systems. The present technological era does not depend only on a standalone computation, rather demands huge data computation through distributed computing along with performance. Almost all the technological giants like Yahoo, Google and Facebook use data intensive computation for their business [1, 2]. Handling high amounts of work load for computation is somehow a challenging task since it is bounded with the performance constraints and availability of the resources. Hadoop has proved to be an effective platform for this purpose. Hadoop is designed such that it can accommodate scaling up from a single standalone systems to excessively large number of systems where each machine provides both computation and storage together [1]..

Major components of Hadoop

MapReduce: MapReduce break down the computation of the jobs in two phases : Map and Reduce. MapReduce architecture consists of one master node (Jobtracker) and many worker nodes (Tasktrackers). The Jobtracker receives the job submitted from the user and split it down into map and reduce tasks, assign the task to the Tasktrackers , monitors the progress of the tasks and report back to the user once the job is completed. Task tracker has a fixed number of map and reduce tasks that it can run at a particular time [3]. Typically Map phase deals with Map function created by the user which splits the job into intermediate sub processes. Reduce phase deals with Reduce function which merges all the intermediate sub processes. Following a illustration of MapReduce for word count process [1].

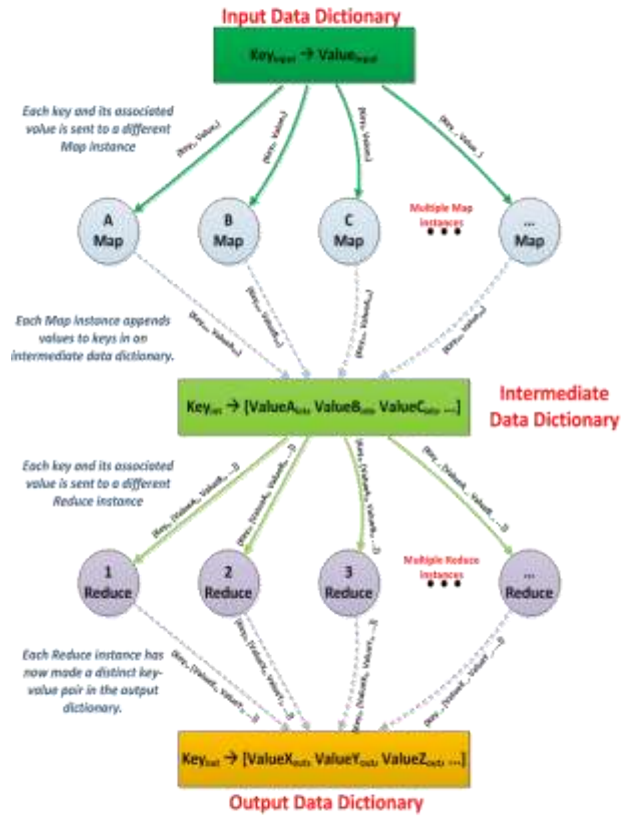
Input Data Dictionary
Key_input → Value_input
Each key and its associated value is sent to a different Map instance
A Map
B Map
C Map
Multiple Map instances
... Map
Each Map instance appends values to keys in an intermediate data dictionary.
Key_int → [ValueA_int, ValueB_int, ValueC_int, ...]
Intermediate Data Dictionary
Each key and its associated value is sent to a different Reduce instance
1 Reduce
2 Reduce
3 Reduce
Multiple Reduce instances
... Reduce
Each Reduce instance has now made a distinct key-value pair in the output dictionary.
Key_out → [ValueX_out, ValueY_out, ValueZ_out, ...]
Output Data Dictionary

Figure 1: MapReduce process

HDFS: HDFS is a Hadoop distributed file system that provides high throughput access to the data. It divides the data into the blocks of size 64 MB and above. In Hadoop architecture, HDFS work as a storage system for both the input and output of the MapReduce jobs. A HDFS cluster primarily consists of a NameNode that manages the file system metadata and DataNodes that store the actual data. Hadoop instance typically has a one Namenode and a cluster of Datanodes in the HDFS. HDFS stores large files (gigabytes to terabytes) spread across multiple machines which achieves reliability by replicating the data across multiple hosts. Datanodes can talk to each other to rebalance data, to move copies around and to keep the replication of data high . Typically each Datanode serves up blocks of data over the network using a block protocol specific to HDFS [2].

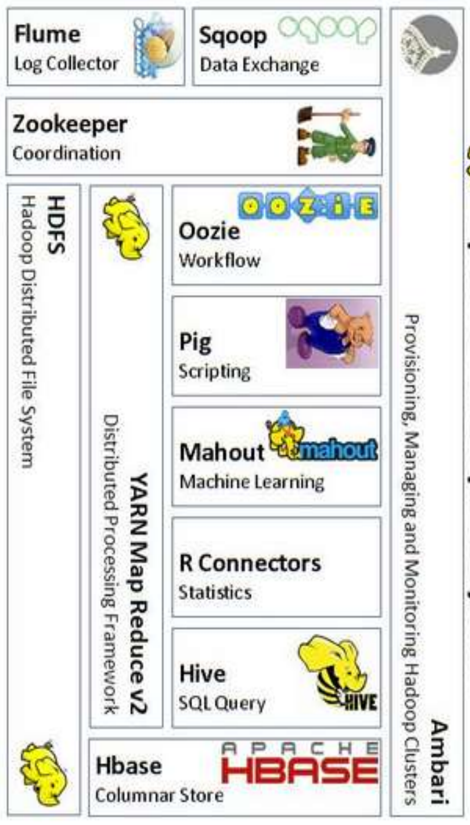

Figure 2: Hadoop Ecosystem

Literature Review

Task scheduling process is a critical part of the Hadoop platform which controls the allocation of resources and sequence of the tasks. It is directly related to the utilization of system resources and overall performance of the platform [3, 4, 5, 6]. Various parameters and suites are applied to measure the performance of the system [7, 8].

All title and author details must be in single-column format and must be centered.

Currently the state of art schedulers allocate the available resources in the form of slots and these are allocated on the basis of fairness only [9, 10]. These schedulers are limited in packing the multiple resources to address the requirement of the particular job. As consequences, it results in the form of fragmentation, over allocation of resources and compromise with the performance for obtaining fairness [10]. These schedulers cannot pack different resources together because they define slots usually on the basis of one or two resources (memory, CPU). Job completion times and packing efficiency suffer as an effect of this limitation and degrades the performance [11, 12].

Slots: When scheduler divide resources into slots (based upon memory and cores) it leads to resource fragmentation, the extent of which increases with the number of resources being allocated (over allocation) [13, 14]. The statically sizing the slots results in wastage of resources and brings fragmentation. While dynamically sizing of slots avoids wastage of resources on slots on which they are defined, but end up being over allocated [15, 16].

Fairness: Fairness picks tasks from the job which are farthest from the fair share. But the problem with fairness based schedulers is that they uses the few resources and do not consider multiple resource requirement of the job for scheduling [17].

For an example, due to the problem of fragmentation and over-allocation of resources, the state-of-the-art schedulers in Facebook’s and Bing’s analytics clusters delay job completions and increase makespan by over 45% [10]. Reference number [10] addresses the issue of multi packing of resources. Multi resource cluster scheduler packs the tasks to the machine based upon the requirements of the job and overcomes the allocation issues, but still it suffers from unfairness to compete with the performance.

Scheduling algorithms in Hadoop

FIFO scheduling

The default Hadoop computing architecture utilize FIFO . The basic job type is large batch job that a single user submits [18]. In FIFO the jobs are submitted to the queue and executed according to the priority level and as per the sequence of their submission. FIFO is considered to be the cost effect and simple in implementation. Even though FIFO is simple, it suffers from large number of limitations. It is basically developed for single kind of jobs and demonstrates degradation in performance when multiple jobs are required to be executed by multiple users. If a job occupies resources for a longer period of time, then the subsequent jobs waiting for the execution may suffer with longer waiting period [19].

Capacity scheduler

Capacity scheduler provides a provision to support multiple numbers of queues where individual node is accommodated with the certain amount of resources. The resources are bounded with the comprehensive set of upper and lower limits to prevent a single job, user and queue from monopolizing resources of the queue or the cluster as a whole [19]. Each queue in turn uses FIFO. Capacity scheduler is elastic in nature and dynamically provisions resources to the heavily loaded queues. During scheduling, it may compute the ratio between computing resources allocated for computing and the number of tasks in execution and selects the smallest ratio [20]. The basic advantages of capacity scheduler are that it supports multiple jobs along with multiple users and provide the provision of dynamically adjusting resource allocation. It also provides job priority feature (which is disabled by default) where a higher priority jobs will have access to the resources. But once a job is running, the preemption for high priority job is not supported. The major disadvantage of capacity scheduler is that user needs to gain knowledge of the system information to select and set up a queue which turns out to be a bottleneck in overall performance of the system.

Fair scheduling

In this scheduling approach , all the jobs on an average gets the equal amount of resources [11, 19, 20]. Distinct to the default Hadoop scheduler FIFO, this allows short jobs to finish in reasonable time while not starving long lived jobs. Fair sharing also accommodate job priorities, the fair scheduler plan the fairness decisions only on memory. It can be further configured to schedule with both memory and CPU, using the notion of Dominant Resource Fairness.

Authors **Wenhong Tian, GuozhongLi**tries to grab attention on-Scheduling element It is observed that jobs are executed can have a significant impact on their overall makespans and resource utilization. In this work, we consider a scheduling model for multiple MapReduce jobs. The goal is to design a job scheduler that minimizes the makespan of such a set of MapReduce jobs. We exploit classical Johnson model and propose a novel framework HScheduler, which combines features of both classical Johnson’s algorithm and MapReduce to minimize the makespan for both offline and online jobs. In this work, by adopting a new strategy, implementation of allocating available MapReduce slots, and combining the features of classical Johnson’s algorithm, we propose and validate new scheduling algorithms for MapReduce framework to mini-mize the makespan.

Authors **Aysan Rasooli, Douglas G. Down** tries to grab attention on-Scheduling elementThe examination paper authors have described the major factors of hadoop scheduling. Bunch or cluster - A Hadoop bunch is a unique kind of computational group outlined particularly for putting away and dissecting enormous measures of unstructured information in a dispersed processing environment.

Workload - approaching occupations are heterogeneous with respect to different highlights, for example, number of errands, information and reckoning necessities, landing rates, and execution times. Reported investigation on Hadoop frameworks discovered their workloads to a great degree heterogeneous with altogether different execution times [9]. Besides, the quantity of little employments (with short execution times) surpasses bigger size occupations in normal Hadoop workloads.

Clients (user): Doled out needs and least impart prerequisites vary between clients. Besides, the sort and number of employments relegated by every client can be distinctive.

Here authors was discussed about only three factors that is users, workload, clusters for batter performance can consider some more factors also that is locality, priority. In this paper face problems with little job starvation.

DOI: 10.18535/ijecs/v6i9.11

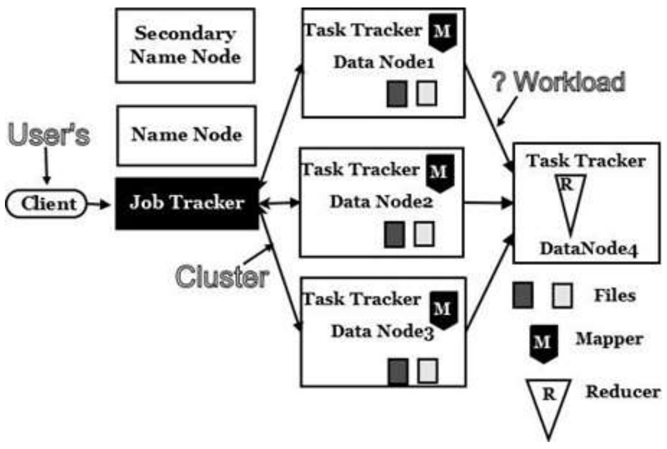

Figure 3 Parameters of mapping and reducing

Authors **MichaelIsard et al** tries to grab attention on- Existing Scheduling's Fair Sharing The authors have described about fair sharing. The center thought behind the decent amount scheduler was to appoint assets to occupations such that by and large over the long run, every employment gets an equivalent offer of the accessible assets. The outcome is that occupations that oblige less time have the capacity to get to the CPU and completion intermixed with the execution of employments that oblige of an opportunity time to execute. This conduct takes into account some intuitiveness among Hadoop employments and licenses more noteworthy responsiveness of the Hadoop bunch to the mixed bag of occupation sorts submitted. The reasonable scheduler was produced by Facebook.

The Hadoop usage makes an arrangement of pools into which employments are set for determination by the scheduler. Every pool can be allocated an arrangement of shares to adjust assets crosswise over employments in pools (more imparts equivalents more noteworthy assets from which occupations are executed). Naturally, all pools have equivalent shares, yet arrangement is conceivable to give more or less imparts relying on the employment sort. The quantity of occupations dynamic at one time can likewise be obliged, if sought, to minimize blockage and permit work to complete in an opportune way.

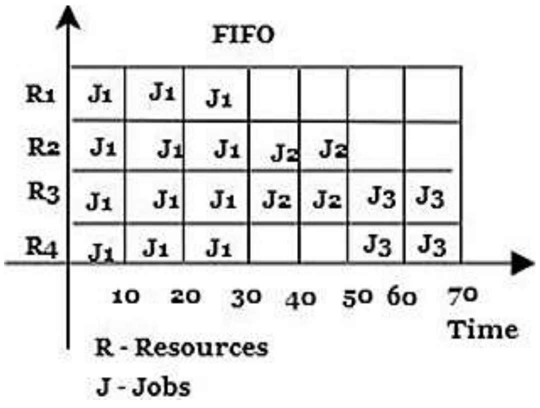

Figure4: Default scheduling scheme

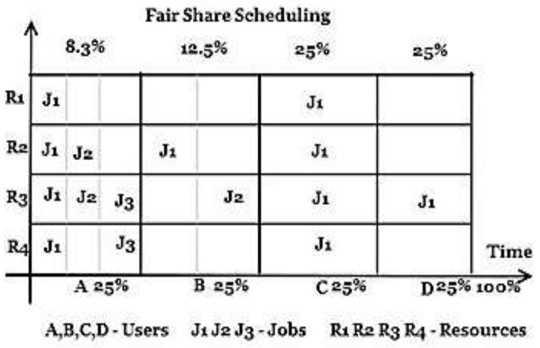

Figure 5: Facebook or fair scheduling scheme

Authors **AysanRasooli, Douglas G. Down** tries to grab attention on- COSHH and Capacity schedulers

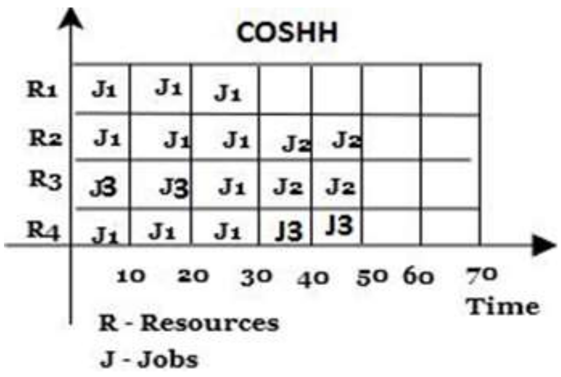

Figure 5: heterogeneous or coshh scheme

In this paper authors have described about capacity scheduling for overcome the sticky slot. The limit scheduler imparts a percentage of the standards of the reasonable scheduler yet has particular contrasts, as well. Initially, limit booking was characterized for substantial groups, which may have numerous, autonomous buyers and target applications. Consequently, limit planning gives more noteworthy control and also the capacity to give a base limit ensure and offer overabundance limit among clients. The limit scheduler was created by Yahoo!

In limit booking, rather than pools, a few lines are made, each with a configurable number of guide and decrease openings. Every line is additionally appointed an ensured limit (where the general limit of the bunch is the aggregate of each line's ability).

The scheduler usage stays informed regarding the process time for every employment in the framework. Occasionally, the scheduler examines occupations to process the distinction between the register time the occupation got and the time it ought to have gotten in a perfect scheduler. The outcome decides the shortage for the assignment. The employment of the scheduler is then to guarantee that the errand with the most astounding shortage is booked next. Authors **DongjinYoo, Kwang Mong Sim** tries to grab attention on-Strength and weakness of scheduling.In fifthreview: The author introduced why some scheduler algorithm may fail underheterogeneous environments. Meanwhile, the paper also discussed a new scheduleralgorithm called COSHH. To ensure the validity of its statement, the author alsoimplemented a series experiment to compare COSHH with current algorithms. In this fifth review author was discussed with contradiction of each available scheduling schemes in current scenario. Then give some countermeasure for that like LATE and COSHH.

Conclusions

For the resource allocation at clusters, the current state of art schedulers such as FIFO, Fair and capacity are suffering from at least one of the following problems: fragmentations, over allocation and scarification of performance over fair allocation. Even though solutions such as Multi resource packing scheduler are being developed to lower down the problem of Fragmentation, over allocation and improve the performance but still they are suffering from the problem of fairness. To gain the performance either fairness is compromised or to gain the fairness, performance is sacrificed. Driven by these problems, a full fledge approach is still required to achieve both fairness and performance together without comprising each other. It is a need of the hour to resolve the problem of over allocation, fragmentation and alongside address the competing objectives such as job completion time and achieving fairness

.

References

1. Hadooptutorial.wikispaces.com, "Hadoop", [Online] Available: http://hadooptutorial.wikispaces.com/Hadoop+architecture
2. Sachin P Bapallege,"An Introduction to Apcahe Hadoop," [Online] Available:
3. http://opensource.com/life/14/8/intro-apache-hadoop-big-data
4. Balaji Palanisamy, A. Singh and Ling Liu, "Cost-effective resource provisioning for MapReduce in a Cloud," IEEE Transactions on Parallel and Distributed Systems, vol. PP, Issue no. 99, p. 1 2014.
5. Yang Wang, Wei Shi, "Budget-Driven Scheduling Algorithms for Batches of MapReduce Jobs in Heterogeneous Clouds," IEEE transaction on cloud computing, vol. 2, issue. 1, p. 306-319, 2013.
6. M. Hammoud and M.F. Sakr, " Locality-aware reduce task scheduling for mapreduce," In Cloud Computing Technology and Science (CloudCom) IEEE Third International Conference, p. 570-576, 2011.
7. Jayalath, C., Stephen, J., Eugster, P., "From the cloud to the atmosphere: Running MapReduce across Data Centers," IEEE Transactions on Computers, vol.63, no.1, p.74-87, 2014.
8. Verma, A., Cherkasova, L., Campbell, R.H., "Orchestrating an Ensemble of MapReduce Jobs for Minimizing Their Makespan," IEEE Transactions on dependable and secure Computing, vol.10, no.5, p. 314-327, 2013.
9. Y. Chen, A. Ganapathi, R.Griffith, R. Katz, "The case for evaluating MapReduce performance using workload suites," In IEEE 19th International Symposium on modeling ,analysis and simulation of computer and tel. com. systems, p. 390-399, 2011.
10. Weikuan Yu ,Yandong Wang , Xinyu Que , "Design and Evaluation of Network-Levitated Merge for Hadoop Acceleration," IEEE Transactions on Parallel and Distributed Systems, vol. 25, Issue 3, p. 602-611, 2014.
11. Robert Grandi, Ganesh A, S Kandula, S Rao, A AAkella, "Multi resource packing for cluster schedulers," In Proc. of the ACM Conference on SIGCOMM 14, p. 445-446, 2014.

DOI: 10.18535/ijecs/v6i9.11

12. M. Zaharia et al. Delay Scheduling, “Delay Scheduling :A Technique For Achieving Locality And Fairness In Cluster Scheduling,” In EuroSys 5th European conference on Computer systems, p. 265-278, 2010.
13. Zhuo Tang, Junqing Zhou, Kenli Li, Ruixuan Li, “A Task Scheduling Algorithm for MapReduce Base on Deadline Constraints,” In Proc. of IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum, 2013.
14. A. Ghodsi et al., “Dominant Resource Fairness : Fair Allocation Of Multiple Resource Types,” In ACM 8th USENIX conference on Networked systems design and implementation, p. 323-336, 2011.
15. Kamal Kc, Kemafor Anyanwu, “Scheduling Hadoop Jobs to Meet Deadlines,” IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), IEEE , p. 338-392, 2010.
16. Aysan Rassoli, Douglas G Down, “A hybrid scheduling approach for scalable heterogeneous Hadoop systems,” In ACM proceedings of SCC’12, p. 1284-1291, 2012.
17. Jeffrey Dean, Sanjay Ghemawat, “MapReduce: simplified data processing on large clusters,” Communications of the ACM, vol. 51 Issue 1, p. 107-113, 2008.
18. M. Zaharia et al., “Delay scheduling : A simple technique for achieving locality and fairness in cluster scheduling,” In Proc. of ACM EuroSys, p. 265-278, 2010.
19. M. Isard, M. Budiu, Y. Yu, “Distributed Data-Parallel Programs from Sequential Building Blocks,” In Proc. of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems, p.59-72, 2009.
20. Jilan Chen,Dan Wang and Wenbing Zhao, “A task scheduling algorithm for hadoop platform,” Journal of computers,” vol. 8, no.4, 2013.
21. Apache, “Capacity Scheduler Guide,” [Online]
22. Available: http://hadoop.apache.org/docs/stable/capacity_scheduler.html
23. Yong Chul Kwon1, Kai Ren2, Magdalna Balazinska1, and Bill Howe11, “Managing Skew in Hadoop,” In IEEE Data engineering Bulletin, vol. 36, no. 1, p. 350- 353, 2012.