*International Journal of Industry and Sustainable Development (**IJISD**)*
Volume 1, No.1, pp. 39- 48, January 2020, ISSN 2682-4000

Survey of Apache spark optimized job scheduling in big data

Walaa Ali Khalil*, Hanaa Torkey and Gamal Attiya

Computer Science and engineering dept. Faculty of Electronics Engineering, Menofiua university, Egypt

*Corresponding Author: walaaali412@gmail.com

Abstract

Big data have acquired big attention in recent years. As big data makes its way into companies and business so there are some challenges in big data analytics. Apache spark framework becomes very popular for using in distributed data processing. Spark is an analytic machine for big data processing with various modules for SQL, streaming, graph processing and machine learning. Different scheduling algorithms vary with its behavior, design and also the goal required solving a problem like data locality, energy and time. The main goal in this research is to represent a comprehensive survey on job scheduling modes using in spark, the types of different scheduler, and existing algorithms with advantages and issues. In this paper, various adaptive ways to schedule jobs on spark and development algorithms to improve performance in Spark will be discussed, analyzed and evaluated. A comparison between different scheduling algorithms, strength and weakness points of them are provided. This can aid to the researchers understanding of which scheduling mechanisms best applied for Big Data.

Keywords: *Big Data, Spark; Scheduler; Scheduling algorithm*

1. Introduction

With the appearance of big data [1], and is becoming of equal importance to business and society. More data needs more accurate analysis, better management and decisions, higher operational efficiencies and cost reducing. Big Data analysis is a complex process of collecting data from various resources so it can be organized and then analyzing those sets of data to discover reliable facts from gathering these data. Big data is generally characterized using the three V's. One of them is the volume which means huge amounts of data that is produced every second. The second one is Variety that refers to increase different forms that data can come in. The most important one is velocity that refers to the data speed can be produced. Some articles include two additional V's, which Veracity refers to the abnormality and noise in data, and Valence refers to the connectedness of big data like atoms in the form of graphs.

As big data makes its way into companies and business so there are some challenges in big data analytics. Such as which technology works best for big data developed without the introduction of new problems, and the questions arise if proper insights from big data analytics will be gained. Other challenges are very detrimental, such as wrong insights and messages when merging between data sets while ignoring data diversity. Security is also a big concern for companies with big data stores and it includes risks related with big data when it comes to the privacy of data. The storage of this huge amount of data is becoming a real challenge. Data storage options like data lakes or warehouses are commonly used to collect and store large

amount of structured and unstructured data in its local format. The real problem increases when data storage tries to combine unstructured and inconsistent data from various resources. In big data analytics there are another challenge is how to plan the completion of tasks and processing in proficient way.

The process of jobs/tasks scheduling in the cluster to reduce resource utilization of the resources and time of completed jobs is called job scheduling. The major demands of job scheduling are cost efficiency, handling various kinds of processing models, scalability, and others. Another major objective of task scheduling is minimizing the task migrations and allocating the number of dependent and independent tasks. The successfulness in performing these objectives in a near optimal manner can reduce the computation time of the jobs and enhance cluster resources utilization.

The structure of this paper is as follows. Section 2 illustrates the problem statement. Section 3 shows job scheduling techniques in apache Hadoop. While, Section 4 explains job scheduling techniques in apache spark. In Section 5, we represent our discussion about these different techniques. Finally, Section 6 is the conclusion.

2. Problem Statement

The main factor to achieve the high-performance goal in big data analyzing is job scheduling. Nominal and proficient scheduling strategies to reach the performance can be influenced by some problems like energy, locality, fairness and synchronization. Huge quantities of energy in handling the data are required, the resources are common between the workers, and fair measures are demanded in scheduling the jobs. Another issue of big data scheduling is data locality. Finally cost reduction in processing time in scheduling jobs in big data analysis must be taken as a goal. In the other word, scheduling aims is to minimize the response time, by using best procedures for scheduling, with the better resources utilization and to make job faster in processing. The processing of big data runs on any framework cluster by separating a job into lesser tasks and classified the load to the worker nodes as shown in Fig. 1.

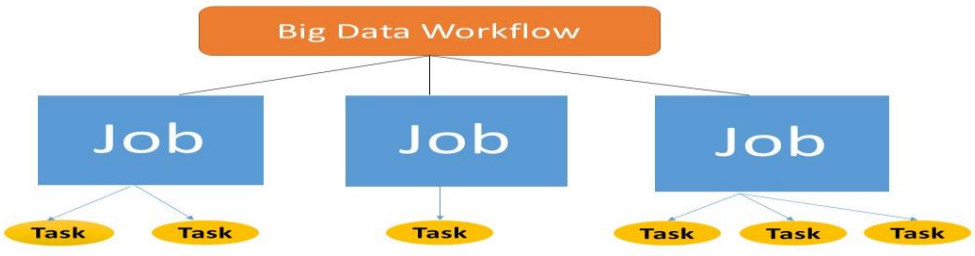

Fig. 1. Split job into tasks to assign on cluster nodes.

How to assign these tasks into slave nodes through the different two types of priority scheduling in the cluster [2] is the main point to analyzing big data. Usually master node is responsible for distributing the tasks to the slave nodes and makes the best scheduling techniques to execute. Some scheduling methods and strategies can be used to get the best

implementation depending on the goal of these scheduling. There are more many schedulers in big data analysis framework like apache Hadoop [3] and Apache Spark [4] to determine the suitable one to use there are many factors, such as type, approach, when to use each scheduler and the goal from using it as illustrate in Fig. 2.

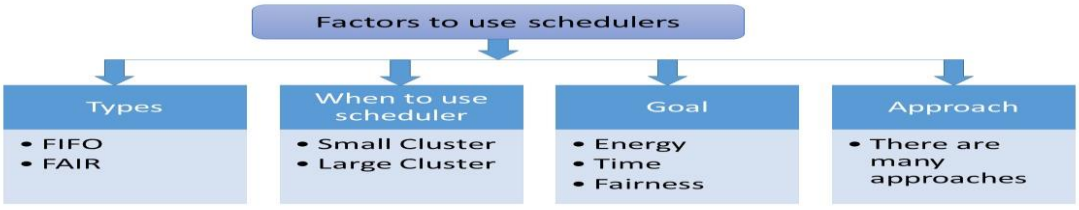

Fig.2. Factors to use schedulers

3. Job scheduling in Hadoop

Apache Hadoop is the most favorite frameworks of big data processing. A Hadoop cluster containa master node and various slave nodes. The master node encompasses four units; a JobTracker, TaskTracker, Data node and Name node. Jobtracker main role is to control the task trackers. JobTracker is a node that manages the execusion of job. TaskTracker also delivers the reports to Jobtracker. MapReduce operate double functions; Map and Reduce operations. scheduling mechanisms are being shared with Hadoop jobs. Schedulers used in Hadoop are:

I. Default FIFO scheduler

FIFO scheduler is running as a default algorithm on Hadoop [5]. The way that this algorithm work is depend on the priority of the job, that mean all jobs will be executed has a priority to run on the available resources on cluster. The jobs arranged on a queue with their priorities.

II. Fair scheduler

The priorities for each job are used by this scheduler[6] relying on the weights to transact with the portions of the total resources. The job will be split to number of tasks and the available slots can ready for processing. The scheduler examines the time deficit against the ideal fair allocation of this job. if the tasks have finished and the slot is ready for next scheduling, then high priority tasks are assigned to the free slot.

III. Delay scheduler

In this scheduler, when the data is not ready, a task tracker stays for a specific time [7]. If there is task assigning requested the node, The size of the job will be reviewed by delay schedulers; when the job is very short, it will be cancelled and if any later jobs ready to run. The important problem that resolved by these schedulers is the locality problem.

IV. Capacity scheduler

Capacity scheduler is utilized when different companies need to share the huge cluster with less capacity and sharing overflowing capacity between users [8]. MapReduce slots is configure any existing queue. The queue has priority with FIFO. resources can be accessed by high priority jobs, matched to the jobs with minimum priority. Scheduled tasks are knew by

memory exhaustion of each task. Also, the scheduler has capable of monitor memory with available resources.

V. Matchmaking scheduler

Matchmaking scheduling can improve Data localities of map tasks [9]. Scheduler guarantees that assign job first by slave nodes before assigning non local tasks. Scheduler hold on trying to detect matches with a slave node. The node will be sign by locality marker and ensures that each node pulls the tasks.

VI. LATE (Longest Approximate Time to End) scheduler

LATE [10] scheduling algorithm tries to improve Hadoop by seeking to find real slow tasks by calculating remaining time of all the tasks. It is based on prioritizing tasks to speculate then selecting fast nodes to run on, and capping speculative tasks to block thrashing. This method only takes action on appropriate slow tasks and does not break the synchronization phase between the map and reduce phase.

VII. Deadline constraint scheduler

Deadline constraints [11] can be appropriated when the jobs scheduling guarantees that jobs met deadline. Deadline constraint scheduler enhances system utilization dealing with data processing. It is achieved by cost model for job execution and Hadoop scheduler with constraint. Cost model with job execution considers some parameters such as the size of input data, task with runtime distribution of job .

VIII. Resource aware scheduler

Resource utilization is minimized by this scheduler in Hadoop . These schemes focus on how effectively resource utilization has been done with different types of utilization like IO, disk, memory, network and CPU utilization[12]. dynamic free slot was being used in these scheduler.

IX. Energy aware scheduler

The enormous clusters within data centers can run big data applications, where their energy costs make executing energy efficiency. This scheduler[13] used for enhancing energy efficiency of applications in MapReduce leads to a reduction of the cost in data centers. Table 1 provides a comparison among different Hadoop job scheduling techniques.

Table 1. Different schedulers used with Hadoop

scheduler	Description	Resource sharing	Advantages	Disadvantages
FIFO	Default scheduler, not depend on priority in job queue, job allocation: static, homogenous environment.	NO	Simple and easy to implement	Job starvation Data locality
Fair	Used weight of job assigned, depend on priority in job queue, job allocation: static, used in homogenous environment.	yes	very effective in allocating resources, Fairness	Configuration problems,

Delay	review the size of the job and decide to run it or not if data of task not ready and task tracker wait, used priority of job queue, job allocation: static, homogenous environment	No	No overhead Simple and easy implement	wasting the work of killed task Not effective.
Capacity	allow sharing a large cluster while giving each organization capacity guarantees ,Not need priority of job queue, job allocation: static, homogenous	yes	Hierarchical Queues, Capacity Guarantees Security,	Complexity choosing scheduler
Matchmaking	guarantees that all tasks appeals a slave node to specifies job first, used priority of job queue, job allocation: static, homogenous environment	yes	High data locality. Utilization level is high	None
LATE	distinguish a moderate running task to dispatch another comparable assignment as a backup, used priority of job queue, job allocation: static, homogenous & heterogeneous	yes	Heterogeneity more robust	Not reliable
Deadline constraint	addresses the issue of deadlines but focuses more on increasing system utilization, used priority of job queue, job allocation: dynamic, homogenous and heterogeneous environment	yes	Helps in Optimization of Hadoop implementation	Uniform nodes to afford cost
Resource aware	Each Task Tracker node observes resources, used priority of job queue, job allocation: dynamic, homogenous & heterogeneous	yes	Performance resource utilization	Monitoring bottlenecks
Energy aware	Minimize energy emitting from data centers, used priority of job queue, dynamic, homogenous & heterogeneous	yes	Energy optimized	Multiple jobs needed

4. Job scheduling in Spark

Batch processing can be implemented with Hadoop MapReduce programming module. There are some restrictions in Hadoop such as there are some constraints in Map and reduce steps when executes big data pipeline. Hadoop is not efficient for performing big data pipeline. There are another bottleneck in Hadoop affects performance, MapReduce relies on reading data from disc that particularly problem for iterative algorithms. To overcome the shortcoming on Hadoop , Spark came out to do this and expand the MapReduce framework.

Spark uses an abstraction called RDDs (Resilient Distributed Datasets) for in memory processing that make it more efficient [14]. RDD is containers where the data stored in memory, and it support fault tolerant. RDDs are immutable and can be created by a series of one transformation. Actions are applied on RDD to guide spark to make computation and retrieve the result to driver program. Spark has two basic components. Master node that has driver program and another component is worker nodes as shown in Fig. 3, Spark context can be created at first inside driver program. Spark Context is the point which services enter to the Spark and receives the job, breaks it in tasks and distribute them to the worker nodes on the cluster. Different jobs can be managed by Cluster Manager and also Spark Context. Cluster

Manager is capable of make scheduling and allocation of resources at the host machines on the cluster. It handles all resources which formed the cluster such as CPU and memory, as well as handling the resource sharing between Spark applications. Worker nodes are the slave nodes whose job is to essentially implement the tasks. A worker node in Spark running Java virtual machine, called JVM or executor. This JVM is the heart of worker node and it consider a core that all the computation is executed, and this is the interface of the storage systems and tools of Big Data.

Fig.3. Apache Spark Architecture

4.1. Apache Spark Ecosystem

The Apache Spark provides API environment including some libraries of code for use in analyzing the data applications. These libraries are Spark SQL, Spark Streaming, Spark MLlib and Spark GraphX as illustrates in Fig. 4 and there are shown in details.

Spark consist of Spark SQL that is the most commonly used component on Spark. It enable to querying the structure and unstructured data. Data sources can be connected with Spark SQL to convert the query result to RDDs in Java, Scala, and Python programs because it contains API responsible for these conversion. To deploy complex operation to data set, DataFrame can be created by the SQLContext. DataFrame is considered as programming abstraction.

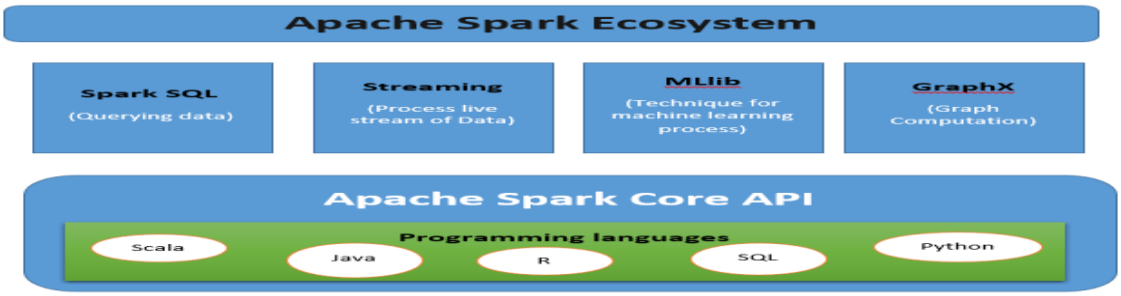

Fig.4. Main modules in Spark ecosystem

Spark Streaming is one of spark component that use to implement and process live stream of data such as sensor data to follow up weather conditions or log files[16]. Spark Streaming support Scala, Java, Python, R and can manipulate data streams that like the Spark Core's RDD API.

Spark GraphX is a library for executing graph-parallel computations and manipulating graphs. GraphX is generated on special RDDs for vertices and edges. It supports a library of combined graph algorithms. It can be write the iterative graph algorithm and fastest graph system make GraphX more performance while keep Spark's flexibility, fault tolerance, and ease of use and this consider as a benefit of GraphX.

Spark MLlib contains many techniques and algorithms that used in machine learning process. It is provides some algorithms to build models for regression, clustering and classification and also more than one technique to evaluate resulting models . Spark enables MLlib to run fast at iterative computation so it contains high quality algorithms to deal with iteration and generates better results [17] .

4.2. job scheduling in Spark

The activity planning for Spark is where job scheduler determine occupation to the following order schedulers to also plan and execute [18]. The Directed Acyclic Graph (DAG) scheduler in Spark produces numerous jobs by executing the submitted application programs. Worker nodes recomputed the data automatically depended on the DAG information from the scheduling systems, which aids in managing data-loss.

There are three kinds of Cluster Manager, there are standalone Cluster Manager[19], Hadoop Yarn[20], and Apache Mesos[21].

5. Discussion

In this papers, the different job scheduling techniques in popular big data platforms, and task scheduling for big data analysis and streaming processing is reviewed. To improve efficiency of jobs execution, some researches proposed pool scheduling pool. In this technique, users send their jobs to scheduling pool which responsible for scheduling its internal jobs. That is mean the scheduling range of scheduling pool is restricted to the inner of application. Sharing and reusing jobs of various application program become very difficult because of the failure of interacting directly with the scheduling pool. In [22], an algorithm support the guarantee, the fairness and the reuse mechanism is proposed. The algorithm can be passed by many steps; a scheduling pool should be selected as mother pool of reuse matching jobs and has reserved the most of cluster resources, then cache the first job in this pool and then cache the result of its execution, another pool contain number of jobs which job scheduler will select.

Another issue is based on streaming data. In Spark streaming, many small sets of records can processes together that called micro-batches then they can be treated by arrived continuously to RDDs. Meanwhile, Spark Streaming buffer the incoming stream and convert it into micro batches. It is essential to improve architecture of the job scheduler. A research at paper [23] proposed a scheduler used for schedules parallel micro-batch jobs in Spark streaming. It improves efficiency because it automatically adjusts scheduling parameters and resource performance. It uses FIFO scheduling with the dependent job and uses fair scheduling for jobs in the independent job pool. There are two factors which restrict the overall efficiency of job scheduling, the complexity of the tasks and the heterogeneity of the cluster, specially Spark.. To solve this scheduling problem, it is vital to determine the priority of scheduling depend on tasks complexity. The scheduling that depends on only number of cores is not suitable for actual heterogeneous cluster. A paper at [24] presented a node scheduling algorithm to optimize the local scheduling in Spark.

Table 2:Comparison between optimized various scheduling algorithms used with spark.

Scheduling strategy	Environment	Job allocation	Resource sharing	Features	Drawbacks
FIFO Fair	Such as in Hadoop				
Scheduling pool scheduling algorithm	Homogenous only	Static	yes	Reduce waiting &execution time of job	Time wasted in matching jobs
Adaptively scheduler to parallel micro-batch jobs in Spark Streaming	Homogenous, Heterogeneous	Dynamic	yes	reducing end-to-end latency	Streaming data only
Scheduling strategy in distributed cluster heterogeneity	Heterogeneous only	Dynamic	yes	Less running time, lower CPU usage	Complexity Heterogon-ous Spark
Operative real time improvement approach to the streaming system	Homogenous, Heterogeneous	Dynamic	yes	Operative real time improvement approach to the streaming system	Depend on past job statistics
Delay scheduler	Homogenous	Static	No	Locality, fairness	wasting work of killed task

Its strategy depends on two points, calculating static level of nodes and dynamic factors. These algorithm is Node stratifying calculate the level of computing performance of node Lj by some steps based on time array then sort with the first place in this array is considered as T[0] and first node as L1 then the layered result of each node can be calculate by this equation:

$$Lj = \begin{cases} Lj - 1 \times \alpha, & 1 \leq j < K \\ 1, & j = 0 \end{cases}$$

Which k is total of layers and Lj is the level of the computing performance of node j and α is constant. From these steps and this equation the level of computing performance of node can be calculated and saved to array L then return array by the same steps to each workers in the cluster.

Another approach is proposed a simple scheduling approach to minimize the case of processing time by dynamic detecting the time window of batch intervals.In an application, the spark submit jobs for each Dstream. jobs are split into a group of tasks. In Spark Streaming, event stream is partitioned into batches in a fixed time interval. In a same batch every event is submitted for processing, the event that comes previously in a time window

consider with bigger delay. So the total delay time of event (e) in a batch is equal time window, data processing time and scheduling delay time:

$$t_t = t_w + t_s + t_p$$

Supposing that scheduling data processing and delay time are constant . the total delay of event processing can reduce by minimizing t_w. A comparison among these optimized various scheduling algorithms is proved in Table 2.

6. Conclusion

Job scheduling shows an important vision to reach the performance in big data analysis. This paper makes an attempt to present background of Job scheduling for big data analysis in Hadoop, Spark architecture, various feature and modules in the spark ecosystem and the job of these modules. Then we discussed the scheduling of spark and the main component which responsible for scheduling, the modes which scheduling occur then make comparison between them and show the main scheduling algorithms in spark. Different optimization mechanics are used to efficiently employ of the resources was discussed. We discover that it is essential to consider various factors when designed a job scheduling algorithm namely; data locality, resource workload, energy consumption, job attributes, deadline of the job, and others. Also, it is noticed that existing task scheduling algorithms are either concentrated on user centric or resource centric paradigm as they failed to address the two factors at once. Our future work will focus on developing a novel job scheduling algorithm considering of all the parameters describe in this paper which can provide better performance.

References

[1] Kaur M, Shilpa, “Big data visualization tool with advancement of challenges.”, International Journal of Advanced Research in Computer Science and Software Engineering 4.
[2] M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, “Job scheduling for multi-user mapreduce clusters”, UC Berkeley, Tech. Rep. Technical Report UCB/EECS-2009-55, April 2009.
[3] S.Kuchipudi, T.S.Reddy, "Applications of Big data in Various Fields", International Journal of Computer Science and Information Technologies (IJCSIT), Vol.6, No.5, Pp.4629-4632, 2015.
[4] Y. Jun, X. Hai et a1., “Spark Core Technology and Advanced Applications”, China Machine Press, pp.238-253, 2015.
[5] M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, “Job scheduling for multi-user mapreduce clusters,” UC Berkeley, Tech. Rep. Technical Report UCB/EECS-2009-55, April 2009.
[6] F. Hamed,” An Overview of Hadoop Scheduler Algorithms”, modern applied science, 2018.
[7] M. Zaharia., D. Borthakur, J. Sensarma, “Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster”, European Conference on Computer Systems, pp. 265-278, 2010.
[8] D. Swans on. Matchmaking,” A New MapReduce Scheduling Technique”, IEEE International Conference on Cloud Computing Technology and Science, pp. 40-47, 2011.
[9] M. Zaharia et al., “Improving MapReduce Performance in Heterogeneous Environments”, Operating systems design and implementation, Pp 29-42, 2008 .
[10] A. Raut et al.,” Deadline aware scheduler for Hadoop Yarn”, IEEE International Parallel and Distributed Processing Symposium, Pp 956-965, 2012.
[11] S. Kalra and A lamba, “A Review on HADOOP MAPREDUCE-A Job Aware Scheduling Technology”, International Journal of Computational Engineering Research (IJCER), Vol. 04, Issue 5, 2014.
[12] L. Mashayekhy, M. M. Nejad, D. Grosu, D. Lu, W. Shi, “Energy-aware Scheduling of MapReduce Jobs”, IEEE International Congress on Big Data, 2014.
[13] S. Ryza, U. Laserson, S. Owen, J. Wills, “Advanced Analytics with Spark”, April 2015.
[14] DAG: https://data-flair.training/blogs/dag-in-apache-spark/.

[15] M. Armbrusty, R. S. Xiny, C. Liany, Y. Huaiy, D. Liuy, J. K Bradleyy,X. Mengy, T. Kaftanz, M.. Franklinyz, A. Ghodsiy and M. Zahariay"Spark SQL: Relational Data Processing in Spark", AMPLab, UC Berkeley, 2015.
[16] S. Ryza, U. Laserson, S. Owen, J. Wills, "Advanced Analytics with Spark", April 2015.
[17] Job scheduing in Spark: http://spark.apache.org/docs/latest/job-scheduling.html.
[18] Standalone Cluster Manger: http://www.agildata.com/apache-spark-cluster-managers-yarn-mesos-or-standalone/.
[19] VK. Vavilapalli et al., "Apache Hadoop YARN: yet another resource negotiator", Proceedings of the 4th Annual Symposium on Cloud Computing, 2013.
[20] B. Hindman et al," Mesos: a platform for fine-grained resource sharing in the data center" ,NSDI, 2013.
[21] Iman Elghandour, Ashraf Aboulnaga, "Reusing Results of MapReduce Jobs", ReStore, pp:586-598, 2012.
[22] D. Cheng, Y. Chen , X. Zhou , D. Gmach and D. Milojicic, " Adaptive Scheduling of Parallel Jobs in Spark Streaming", 2016.
[23] X. Zhang, Z. Li, G. Liu, J. Xu, T. Xie and J. P. Nees1, " A Spark Scheduling Strategy for Heterogeneous Cluster", CMC, vol.55, no.3, pp.405-417, 2018
[24] X. Liao, Z. Gao, W. Ji, Y. Wang, "An Enforcement of Real Time Scheduling in Spark Streaming", Beijing Institute of Technology, 2015.