Journal Homepage: - www.journalijar.com

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)

Article DOI: 10.21474/IJAR01/6672
DOI URL: http://dx.doi.org/10.21474/IJAR01/6672

RESEARCH ARTICLE

THE AMBIENT SCRUTINIZE OF SCHEDULING ALGORITHMS IN BIG DATA TERRITORY.

Dr. Yusuf Perwej.
Ph.D (Computer Science & Engineering), M.Tech, Assistant Professor , Department of Information Technology, AL Baha University, AL Baha, Kingdom of Saudi Arabia (KSA).

Manuscript Info

Manuscript History

Received: 05 January 2018
Final Accepted: 07 February 2018
Published: March 2018

Keywords:-
Big Data, Scheduling Algorithms, Optimization Techniques, Hadoop, Processing, MapReduce.

Abstract

Today scenario, we live in the data age and a key metric of existing times is the amount of data that is originates ubiquitously around us. At present-time intense increase in the number of Internet subscriber and connected devices, as well as rising of the IoT. As an outcome, quantities of data are originated (so called Big Data), such as user data (structured, unstructured, or semi structured), sensor data and log files. It is an increasingly business for companies to collect and analysis Big Data and provides insights to their client. In general processing such spacious amount of data with multifarious formats can be time consuming. The Hadoop is an open source framework that is used to process spacious amounts of data in an economical and proficient way, and job scheduling has become a significant factor to attain high performance in Hadoop cluster. The job scheduling algorithms are essential for efficient make use of cluster resources and executing them in short time. The fundamental purpose of this paper is to present a classification of Hadoop schedulers along with their existing scheduling algorithm in Hadoop territory. In addition, this paper paraphrases the features, advantages, disadvantages, and limitations of several Hadoop scheduling algorithms.

Introduction:-

Nowadays, Big Data is very popular, because it has proved to be much success in many fields such as social media, transactions, banking, [1] online and on-site purchasing, E-commerce, healthcare, astronomy, oceanography, finance and business, engineering, and many other fields. Big data depict the tools and technologies expected to store, distribute, capture, manage, [2] and analyze petabyte or enormous sized datasets having various structures with a high momentum. In this new era with the advancement in the technological world the data storage, analysis becomes a major problem. Although the availability [3] of different data storage component like electronic storage such as hard drive or virtual storage such as cloud still the problems remains. The major issue is processing the data because usually the data have been in several formats and size. Hadoop system is used to process large datasets. The Hadoop is an open, sources Java based [4] framework which can run applications in the cluster that consist of reasonably priced hardware, for processing and storing large amount of data in a distributed computing environment [5]. The data sets are very complex and growing day by day in humongous volume. Raw data are continuously generated from social media, online transactions, etc [6]. Due to continuous increase in volume, velocity and variety, complexity increases, it induces lots of difficulties and challenges in data processing. Big Data becomes a complex process in terms of correctness, transform, match, relates, etc.

Corresponding Author:- Yusuf Perwej.
Address:- Ph.D (Computer Science & Engineering), M.Tech, Assistant Professor , Department of Information Technology, AL Baha University, AL Baha, Kingdom of Saudi Arabia (KSA).

The min aim of scheduling algorithms in the Big Data processing is to plan the processing and completion of as many tasks as possible by handling and altering data in a proficient way with a minimum number of changes [4]. Hadoop with job scheduling plays a vital role to achieve the performance in Big Data. Hadoop allows the user to configure the job, submit it, control its execution, and query the state. Every job consists of independent tasks [7], and all the tasks need to have a system slot to running. Again Hadoop all scheduling and allocation verdict are made on a job and node slot level for both the map and reduce phases. When a user submits a transaction, Hadoop will create a job and put it in the queue of jobs waiting for the job scheduler to dispatch it [8]. Then, the job will be divided into a series of tasks, which can be executed in parallel. The task scheduler is responsible for dispatching tasks by certain task [9] scheduling algorithm. In this paper, we present a comprehensive study of all Hadoop schedulers in the Big Data Territory. This paper will be useful for both beginners and researchers in understanding Hadoop job scheduling algorithms in the Big Data processing.

II. The Primary Challenges of Scheduling in Big Data Environments

There is a requirement for efficient scheduling algorithms on the handle of big data on different nodes in Hadoop clusters [10]. There are different factors that influence the performance of scheduling policies like as speed (data velocity), security, data volume (storage), the format of data sources (data variety), and privacy, connectivity, data sharing and cost. To instate better utilization of resources and handling of the big data, scheduling policies are designed [11][12]. The primary challenges related to job scheduling in big data in a nutshell as follows.

Data Diversification: The majority of the datasets is different semantics, type, granularity, homogeneous as well as heterogeneous in structure, organization and accessibility. A skilled data presentation should be designed to replicate the structure, hierarchy and diversity of the data and an unification technique should be designed to enable efficient operations across various datasets.

Data Storage and Handling: Primarily big data are dependent on comprehensive storage capacity and data volumes grow exponentially, the present data management systems cannot propitiate the exigency of big data due to limited storage capacity. Another vital issue of big data is that it is very enormous and includes data that is in an unstructured format, which makes it rigorous to organize the data for analysis. The storage of unstructured data is not a simple task. Therewith, not an effortless task to store data effectively in view of the fact that the homogeneous as well as the heterogeneity of big data.

Rapidity: At present, everybody expects everything to be done instantaneously. Nowadays, hyper-competitive business environment, companies not only have to discover and analyze the episodic data they expected, they must discover it rapidly. Again, visualization helps organizations perform analyses and make decisions much more fastly, but the challenge is going through the sheer volumes of data and accessing, the level of detail be expected and all of a high motion. The motion is a foremost issue in big data. The motion of big data is restricted by several difficulties, namely actual time & offline difficulty, statistical analysis difficulty, import & export difficulty, and query & retrieval difficulty.

Data Processing and Analysis: The data are ubiquitously and are not ever presented in the same manner every database has its personal format. For some experts, the big difficulty in big data is more one of processing than of volume. The query response time is an important difficulty in big data, as adequate time is required when traversing data in a database and performing actual time analytics. A resilient and reconfigured grid along with the bigdata preprocessing increase and consolidation of application and data parallelization strategy can be more influential approaches to quotation more meaningful knowledge of the given data sets.

Energy Management: The data transmission, storage and processing will unquestionably consume more energy, as data volume and analytics demand increases. The energy consumption of huge scale computing systems has attracted greater anxiety from environmental and economic perspectives and to endow extensibility and accessibility system level power control and management mechanisms must be considered in a big data system.

Expense: The cost is also main matter in big data. The cost up-gradation matter in big data are cost comparison between alteration of nodes, make better, and master and slave nodes.

Data Privacy and Security: Data privacy and security is one of the biggest matter for big data since the host of data or other critical operations can be performed by third-party services or infrastructures, security matter is witnessed with respect to big data storage and processing. The present technologies used in data security are mainly static data-oriented, albeit big data entails the dynamic transformation of current and extra data or variations in attributes. In Privacy-preserving data mining without the show up sensitive personal information is another challenging field to be calibrate. Today scenario increasing of online services and mobile phones, privacy and security concerns regarding accessing and analyzing personal information are growing. It is rigorous to understand what support for privacy must be provided at the platform level to remove privacy leakage. We necessity to consider all the data privacy and security rules.

Syncing Across Data Sources: On one occasion you import data into big data platforms you may also realize that data copies transmigration from a wide range of sources on various rates and schedules can expeditiously get out of the synchronization with the originating system. This allude that the data coming from one source is not out of date as compared to the data coming from another source. It also signifies the concepts, commonality of data definitions, metadata and the like. The conventional data management and data warehouses, the sequence of data alteration, extraction and transmigration all arise the circumstance in which there are hazards for data to become unsynchronized.

Approximate Analytics: The analysis of thorough dataset is becoming more rigorous as data sets grow and the actual time requirements become stricter. One solution to extricate this difficulty is to endow estimated outcome by means of estimate query. The estimate query has two dimensions such as that the accuracy of the outcome and the groups leave out from the output.

Scalability: In scalability is the primary challenge with the big data. You want to be able to scale very expeditiously and elastically, whenever and wherever you want. There is a requirement of powerful solution to enable the processing of the enormous volume of data, scalable storage, feasible and cost-effective.

Connecting Social Media and Data Sharing: The social media have peerless properties such as elaboration, statistical redundancy and the availability of user response. To recognize references from social media to distinguish product names, locations or human beings on websites, several extraction techniques have been successfully used. The applications can instate high levels of precision and well-defined points of view by connecting inter-field data with social media. In data sharing are still matters that requirement to be considered. There are different dispute in big data related to data sharing namely interfaces and data criterion, shared protocols, and access authorization etc.

Additional Miscellaneous Challenges: Additional challenges may occur while incorporating big data. Some of the challenges include unification of data, the volume of data, expertise availability, solution cost, the rate of alteration of data, truthfulness and legitimacy of the data. The capacity to intermingle data that is not similar in source or structure and to do so at a rational cost and in time. It is also a challenge to process an enormous amount of data at a rational speed so that information is available for the data client when they exigency it. The legitimacy of data set is also fulfilled while relocating data from one source to another or to clients as well.

III. A Necessity for Scheduling Algorithm in Big Data Environments

The common necessity for scheduling in big data platforms define the functional and nonfunctional specifications for a scheduling service. The below common requirements are as follows.

Cost Efficiency: The scheduler should bottommost the total cost of execution by decreasing the total number of resources used and respect the total money budget. This aspect expects efficient resource usage.

Elasticity and Scalability: A scheduling algorithm must take into contemplation the peta-scale data volumes and hundred thousand of processors that can be associated with the processing job. The scheduler must be conscious of the execution environment transformed and be able to adapt to workload transformation by provisioning resources.

Usual Intention: A scheduling method should make assumptions about and have few restrictions to different types of applications that can be carried out. Interactive jobs, parallel and distributed applications, as well as non-interactive batch tasks should all be supported by high performance.

Disinterestedness: The job sharing resources among users in a fair way to assure that each user receive resources on demand and in a pay-per-use model in the cloud, a cluster of resources can be allocated dynamically or can be reserved in advance.

Time Management Ability: The scheduler should make better the performance of scheduled jobs as much as possible using various heuristics and state estimation appropriate for specific job models. Multitasking systems can process several data sets for multiple users simultaneous by mapping the tasks to resources in a way that make the best their use.

Dynamicity: The scheduling algorithm should make use of the full extent of available resources and may transform its behavior to cope. The scheduler exigency to continually adapt to resource availability alteration, paying special attention to cloud systems and HPC clusters, as authentic solutions for big data [13].

Load Balancing: This is used as a scheduling procedure to share the load among all receivables resources. There are classical method like round-robin scheduling, but also, the new method that cope with huge scale and heterogeneous systems were proposed such as slow start time, agent-based adaptive balancing , minimal connection.

Endorsement of Data Heterogeneity and Various Processing Models: By managing several concurrent input streams, unstructured content and structured, multimedia content, and state-of-the-art analytics.

Unification with Shared Distributed Middleware: The scheduler must think about different systems and middleware frameworks, namely the sensor integration in any place following the IOT instance, or even mobile cloud solutions that use offloading techniques to scrimp energy.

Based Upon Resource Availability: Primarily, this scheduling action plan is based on the resource requirement of the job. Under this action plan, resource utilization namely I/O, , disk storage, memory utilization, and CPU time, after that job performance is excellent.

IV. Taxonomy for Hadoop Scheduling Used in Big Data

The Hadoop job schedulers are designed for superior utilization of resources and performance enlargement. It is also a confer promise capacity to production job and superior response time to interactive jobs while allocating resources impartially between client. The taxonomy for Hadoop scheduling shown in figure 1. The Hadoop job scheduler can be classified in terms of the following parameters such as environment, priority, energy, resource awareness, namely free slot, disk space, CPU time, I/O utilization, time, and action plan [5]. The usual goal of scheduling algorithms is to alleviate the execution time of parallel applications and also to extricate matter related to data processing. The primary consideration behind scheduling is to keep down overhead, resources, and completion time, and to make as large throughput by allocating jobs to the processor. At this place, the classification of schedulers is based on the obtainable resources efficient and effective, scheduling action plan, resource availability, and moment [14][15][16].

Static Scheduler Policy

In static scheduling policy to job allocation in processors, it is attaining before the program to begin the job running time, in ending time. The processing resource and job running time are recognized only at the time of ending. The main aim of static scheduling is to keep down the overall execution time of existing programs [17].

Dynamic Scheduler Policy

In dynamic scheduling policy for allocation of jobs to the processors is done during execution time. The scheduler has some understanding about the resource before execution, but the environment in which the job [18] will be executed is totally unacquainted, and the job will be executed during their lifespan. Meanwhile a job is executed, a decision is made and dynamic environment applies to the processes [19].

Based on Available Resource Scheduler Policy

Fundamentally, this scheduling policy is based on the resource requirement of the job [19]. This scheduling is to ameliorate the job performance and resource utilization. The resources can be I/O, disk storage, memory, CPU time [20].

Time Based Scheduler Policy
This scheduling policy is based on time therein, job ending depends on the user provided finishing date. In this scheduling policy, there is a time limit within which the job must be done. The user specified ending date, then scrutinize whether the job is [21] finished within the given limit or not. The finishing date can be specified by the user and then it can examine whether job executes within that finishing date or not.

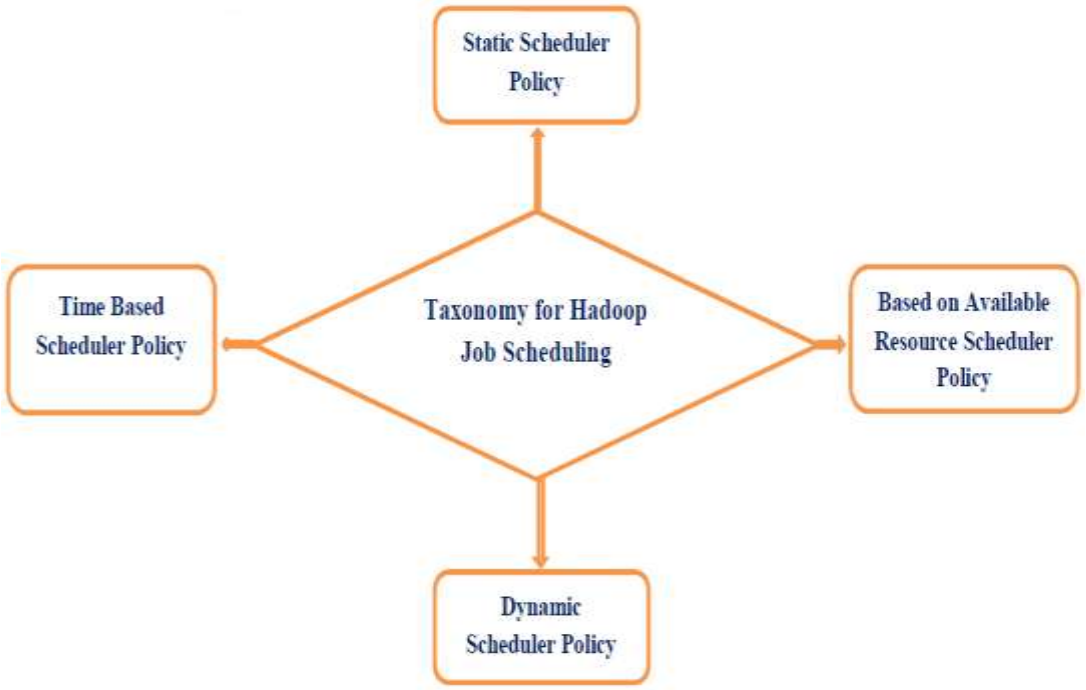

Static Scheduler Policy
Time Based Scheduler Policy
Taxonomy for Hadoop Job Scheduling
Based on Available Resource Scheduler Policy
Dynamic Scheduler Policy

Figure 1:- The Taxonomy for Hadoop Scheduling Policy

V. The Hadoop Scheduling Algorithm
The scheduling algorithm is one of the basic technologies of Hadoop platform, its primary function is to manage the order of job execution and assign the user's job to execute upon resources. Familiar with that Hadoop is a general-purpose system that empower high-performance processing of data over a set of distributed nodes. However, according to this definition is the fact that Hadoop is a multi-tasking system that can process several data sets for several jobs for several users simultaneously.This ability [22] of multi processing means that Hadoop has the chance to more optimally map jobs to resources in a way that renovate their use. Afterwards, Hadoop jobs are sharing the cluster resources with scheduling policy based on the scheduling contrivance when and where jobs have to be running. Not only numerous types of application on Hadoop platform shared among several users are growing more, therewith also combination of batch long jobs and interactive short ones which access identical data set has become a typical way of job running. Here upon, in a homogeneous environment, the main intention of multi-user scheduling algorithm [23] is to fulfill the balance between proficiency of Hadoop cluster and the dexterity of resource allocation among jobs. The purpose of scheduling is to reduce the closure time, enhanced throughput, reduce overhead, and balance available resources of a parallel application by appropriately allocating the jobs to the processors. Many of them get [24] focus to reform data locality and many of them implements to confer synchronization processing. Because the pluggable scheduler was implemented, many scheduler algorithms have been developed for it. In this section, we will investigate the different algorithms available and when it makes sense to use them.

Longest Approximate Time to End (LATE) Scheduling Algorithm
The Longest Approximate Time to End (LATE) algorithm is based on three theories make a priority task to conjecture, choose fast nodes to execute on, and cap imaginary tasks to inhibit thrashing. The Zaharia , Konwinski et al. introduced Longest Approximate Time to End (LATE) [25] scheduler to robustly make better performance by the deficiency overhead of speculation execution tasks. The primary goal of LATE scheduler is to optimize the performance of jobs and to reduce job response time. When short jobs are executing, the response time is fast, which

is very vital. In spite of, it executes long jobs very stilly due to many matters, namely huge number of background processes, stilly background process, inaccessibility of resource, CPU load etc. LATE algorithm is robust to node heterogeneity, because only some of the sluggish speculative tasks are restarted. This technique does not break the synchronization phase between the map and reduce phases, but only takes steps on suitable stilly tasks.

FIFO Scheduling Algorithm

The primary goal of the FIFO scheduler to schedule jobs based on their priorities in first-come, first serve basis and Hadoop by default uses the FIFO scheduling algorithm. FIFO stands for first in, first out which at its Job Tracker pulls longstanding job first from job queue and it doesn't be interested in priority or the size of the job. According to the priority level and the time sequence when they are put forward, the entire job queues are scanned, and then a tolerable [26] job is choosing to execute. The FIFO scheduling strategy is used when the order of execution of jobs has no significance and this scheduling algorithm is straightforward, but there are a lot of restrictions. It is invented only for a single type of job, so when multiple users at the same time run multiple types of jobs, performance will be relatively minimal. As the usage rate of Hadoop platform is progressively high, the demand is also extended [26]. This algorithm tends to reduce the overall performance of the platform and the utilization of system resources, and infrequently even influence the implementation of jobs.

Hybrid Scheduling Algorithm

The H. Nguyen, T. Simon et al. introduced Hybrid Scheduler algorithm to robustly renovate response time for multi-user hadoop environments [27]. The primary aim of Hybrid Scheduler algorithm based on dynamic priority in order to lower the delay for variable length concurrent jobs, and relax the order of jobs to sustain data locality. Additionally, it's endows a user-defined service level value of quality of service [27]. This algorithm is designed for data profound workloads and tries to sustain data locality during job execution. Their trust, average response time for the workloads almost 2.2x faster over the Hadoop Fairs with a criterion deviation of 1.4x. It attains this excellent response time by means of relaxing the rigid proportional fairness with a simple exponential strategy model. This algorithm is an intense and resilient scheduler that ameliorates response time for multi-user Hadoop territory.

Fair Share Scheduling Algorithm

The Fair scheduling is a hand over resources to jobs, namely on average, all jobs avail, an equal share of resources during a time. The fair scheduler was introduced by Facebook. The core thinking behind the fair share scheduler was to assign resources [28] to jobs, namely on average over time, each job gets an equal share of the resources at hand. The outcome is that jobs that need less time are able to access the CPU and ending intermixed with the execution of jobs that need more time to execute. Again, this ongoing permit for few interactivity among Hadoop jobs and assent greater accountability for the Hadoop cluster to the diversify of job types produced.

The situation of the Fair scheduling algorithm is to do an identical distribution of computing resources among the users & jobs in the system. The scheduler typically organizes [29] jobs by resource pool, and shares resources fairly among these pools. The fair allocation of resources among the pools with the MapReduce task slot. If any pool is free i.e. there are not being used, then their inactive slots will be used by the other pools. Supposing the similar user or same pool sends too several jobs , then the fair scheduler can limit these jobs by marking the jobs as not executable [29]. Fair share scheduling algorithm endorsement job taxonomy scheduling, so that various types of jobs receive various resources, thereby the quality of service is refined and paralleling number is [29] dynamically adjusted. In as much as, it makes the job fully use system resources, [28] and refine the utilization degree of the system. You configure fair share in the mapred-site.xml file. This file explains the properties that collectively govern the behavior of the fair share scheduler. An XML file designated to by the property mapred.fairscheduler.allocation.file explain the allocation of shares to each pool and improve for job size, you can set the mapread.fairscheduler.sizebasedweight to hand over shares to jobs as a function of their size.

Self-Adaptive MapReduce (SAMR) Scheduling Algorithm

The Q. Chen, D. Zhang et al. proposed Self-Adaptive MapReduce (SAMR) scheduling algorithm to reduction the execution time of [30] MapReduce jobs, in particular distinct environments. The primary aim of SAMR scheduling algorithm ameliorates MapReduce by saving execution time and system resources. It defined intense nodes and stilly nodes, to be nodes, which can ending a task in a concise time and longer time than most other nodes. The process of SAMR algorithm includes reading the historical information and tuning parameters, updating the historical information, discovery the stilly tasks, launching backup tasks, collecting outcome and discovery the stilly Task Tracker.

Accordingly, it gets the progress of each task correctly and quest which tasks requiring for backup tasks. What's more, it identifies stilly nodes and classifies them into the sets of stilly nodes dynamically [31]. In pursuance of the information of these sluggish nodes, SAMR will not launch backup tasks on them, ensuring the backup tasks will not be sluggish tasks any more. It gets the final outcome of the fine-grained tasks when either sluggish tasks or backup tasks completion first.

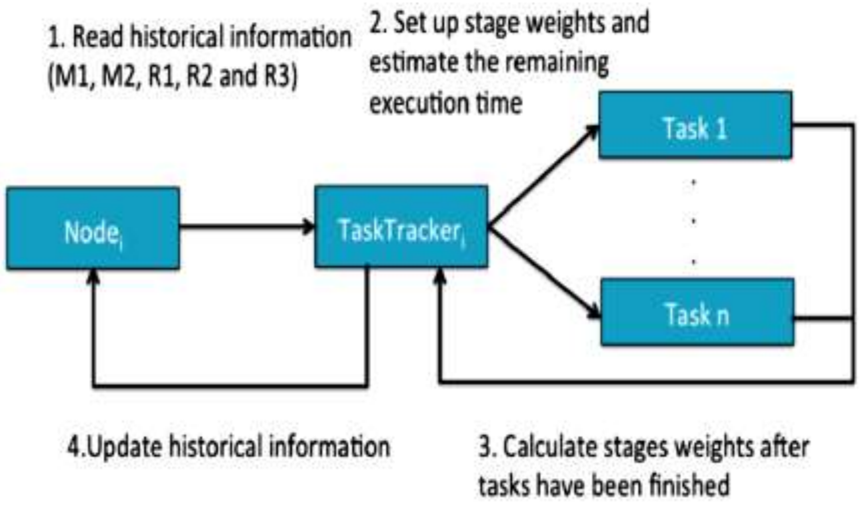

1. Read historical information (M1, M2, R1, R2 and R3)
2. Set up stage weights and estimate the remaining execution time
Task 1
Node$_i$
TaskTracker$_i$
Task n
4.Update historical information
3. Calculate stages weights after tasks have been finished

Figure 2:- The Process of Using and Updating the Historical

The figure 2 shows the process of using and updating the historical information in SAMR. First, Task Tracker read historical information from the nodes where they are execution of. These historical information includes historical values of M1, M2, R1, R2 and R3. Now TTs tune M1, M2, R1, R2 and R3 according to historical information and information collected from the current running system. Hence, TTs collect values of M1, M2, R1, R2 and R3 according to the real running information after the tasks ended. In the end, TTs write these updated historical information into the xml file in every of the nodes.

Capacity Scheduling Algorithm

The capacity scheduler is designed to execute [29] Hadoop applications as a shared, multi-tenant cluster in an operator- matey fashion while maximizing the throughput and the utilization of the cluster. The capacity scheduler was basically developed by Yahoo. The primary goal of this scheduler is to maximize the utilization of resources and throughput in a cluster environment. The capacity scheduler shares some of the principles of the fair scheduler, but has segregate dissimilarity, too. First, capacity scheduling was defined for huge clusters, which may have several, self-sufficient consumers and target applications. For this cause, capacity scheduling provides greater control as well as the capability to provide a least capacity assurance and share excess capacity among users. Its uses queue instead of the pools distinct fair scheduler. Every queue is stipulated to an organization and resources are divided among these queues. That is to say, capacity scheduling algorithm puts jobs into several queues in accordance with the conditions, and allocates certain system capacity for each queue. If a queue has bulky load, it seeks unallocated resources, then makes unnecessary resources allocated evenly to each job. For maximizing resource utilization, if a queue is not dissipated its allocated capacity, this surplus capacity can be nonce allocated to other queues. When new jobs come in a queue, the resources are [29] allocate back to the foregoing queue after ending of the currently running jobs. Another distinction of fair scheduling is the potential to prioritize jobs within a queue. Normally, jobs with a topmost priority have access to resources sooner than bottommost priority jobs. If you need to configure the capacity scheduler within multiple Hadoop configuration files the queues are defined within hadoop-site.xml, and the queue configurations are set with capacity-scheduler.xml. You can also configure ACLs within mapred-queue-acls.xml.

Delay Scheduling Algorithm

This Delay scheduling is introduced by applying transformation to MapReduce with data locality to obtain better performance and least response time for the map task. The Facebook uses the same waiting technique to obtain locality in the Hadoop cluster. The M. Zaharia, D. Borthakur et al. proposed a straightforward algorithm which named delay scheduling to address the clash between locality and fairness. Delay scheduler uses the expectant technique for magnify the locality [32]. The Delay scheduling is used to ameliorate data locality by asking jobs to wait for its turn for scheduling on a node with local data. When a node appeal a task and if the head-of-line job cannot launch a local task, then it is omitted and looked at next jobs. But if a job has been omitted for long enough, then non-local tasks are permitted to launch to prevent starvation. In this algorithm in spite of the first slot given for a job is not likely [33] to have data for it, but tasks come to an end very hastily that some other slot accommodate data for it will free within a small period of time. Delay scheduler endeavor to obtain fairness with locality. This scheduler relaxes harsh job order for the task, handing over.

Context Aware Scheduling Algorithm

The K.A Kumar, V.K Konishetty et al. introduced a context-aware scheduler [34]. This algorithm uses the current heterogeneity of most clusters and the workload amalgam, offer optimizations for jobs using the same dataset. The scheduler must gather context information i.e. available resources on the nodes to find out dynamic transformation. In slave JobTrackers must in contact periodically with the master TaskTrackers to keep up-to-date information and let the scheduler prepare to the new context. The design is based on two key expedients. First, a huge percentage of Map Reduce jobs execute periodically and roughly have [35] the same essential quality relating to network requirements, CPU, disk. Second, in a Hadoop cluster, the nodes become heterogeneous over time due to lack of success, when newer node replacement old ones. Supposing Hadoop does not perform preemption & migration of tasks, the participation of speculative tasks and context aware scheduler may contribute to avoid the bottlenecks caused by the resources mutability.

Deadline Constraint Scheduling Algorithm

The Deadline Constraint Scheduler addresses the matter of time limit, but focuses more on increasing system utilization. The user specified time limit constraints at the time of scheduling the jobs make sure of that the jobs scheduled for execution meet the time limit [36]. It attention on the matter of deadlines and raise system utilization. It deals with the time limit requirement by the cost model of job execution, which considers parameters, namely the input size of data, map and reduce execution time, data distribution etc. While any job is scheduled, it is examined by the scheduler whether it will be ended within the time specified by the time limit or not. Deadline Constraint Scheduler demonstrated the following statements firstly multiple source of independent aperiodic tasks can be considered estimated to a single one and secondly, when the number of approximate resources go beyond a data center capacity, the tasks migration between various regional centers is the appropriate solving a problem with respect to the global deadline and lastly in a heterogeneous data center, we requirement higher number of resources for the same request with respect to the deadline limitation [37].

Resource Aware Scheduling Algorithm

The Polo J, Castillo C et al. evolved resource-aware scheduling technique for Map Reduce with multi-job workloads that objective to enhance resource utilization across machines while observing closure time [38]. The resource-aware scheduling dynamically ascertain the number of job slots, and their position in the cluster at execution time. The resource-aware scheduling endows scheduler with the adaptability necessity to respond to transposition conditions in resource requisition and availability. In this scheduling different resources like CPU utilization, I/O utilization, network utilization, memory utilization, and disk utilization are shared more desired result. In this strategy, the scheduling is accomplished by two nodes named Master Node and Work Node, which are also known as Job Tracker and Task Tracker, systematically. The Job Tracker handles lists of tasks allocated to each Task Tracker, states of Task Trackers in the cluster, and the queue of the currently execution jobs, while the Task Tracker is accountable for the execution of each task configured with the utmost number of obtainable slots.

Matchmaking Scheduling Algorithm

The C. He, Y. Lu, et al. evolved a new matchmaking algorithm to refine the data locality rate and the average repercussion time of MapReduce clusters [39]. Matchmaking scheduling, attention on the improvement of data locality of map tasks. The main opinion behind this technique is to give every slave node a fair opportunity to grab local tasks prior to any non-local tasks are allocated to them. A task is called local task when it is running on the node where its data is obtainable. This scheduler discovers for matches, for instance, in the case when the slave node holds some input data, each Map task is not allocated. Every node is marked by a locality marker to reaffirm that every node gets equal chance to seize local tasks. This scheduling has the smallest reaction time, but the towering data locality for map tasks. In the end, all slave nodes' locality markers will be cleared when a new job is added to the job queue. So far as a new job may [39] comprise new local tasks for some slave nodes, upon the new job's coming, our algorithm resets the status of all nodes and again starts the all-to-all task-to-node matchmaking process.

Enhanced Self Adaptive MapReduce Scheduling Algorithm
The Enhanced Self-Adaptive MapReduce scheduling algorithm to do better for the speculative re-execution of sluggish tasks in MapReduce. In ESAMR, in order to recognize sluggish tasks accurately, [40] we differentiate historical platform weights information on every node and divide them into K clusters using a K-means clustering algorithm and when running a job's tasks on a node. ESAMR categorize the tasks into one of the clusters and uses the cluster's weights to guess the running time of the job's tasks on the node. ESAMR steerage to the smallest error in task execution time guess and know sluggish tasks most as a matter of fact. Again, ESAMR uses a machine learning technique to classify the historical information stored on every node into k clusters. Supposing a running job has completed some map tasks on a node, then ESAMR records the job's nonpermanent map phase weight on the node according to the job's map tasks endowed on the node and uses the nonpermanent weight to discover the cluster whose weight is the nearest.

Combination Re-Execution Scheduling Algorithm
The L. Lei, T. Wo, et al. evolved amalgamation Re-Execution scheduling algorithm for assistance in increased running time for speculative map tasks and reduce the response time of MapReduce jobs [41]. The primary goal behind this is executed repeatedly the amalgamation of tasks on couple of nodes may outcome in quicker progress compared to subjecting a task directly and he target node, due to data locality. The assessment conducted demonstrates that CREST can decrease the running time of speculative Map tasks by 75% on the best cases and 55% on average, compared to LATE. This amalgamation Re-Execution scheduling algorithm is superior than LATE and brings reform by re-executing an amalgamation of tasks on a group of nodes.

Locality-Aware Reduce Task Scheduling Algorithm
The Locality-aware reduce task scheduling algorithm introduced by M. Hammoud, M. Sakr, et al [42]. This technique to keep away from scheduling lateness, scheduling skew, substandard system utilization, and a small degree of parallelism. The Locality-aware reduce task scheduling employs a relaxed policy and fragments, some reduce tasks among many cluster nodes. This algorithm uses a practical policy that leverages network locations and sizes of partitions to exploit data locality. The reduce phase scheduling is improved to become aware of the split, locations, and size, of deficiency network traffic. Again, this algorithm incorporates network locations and sizes of diminishing split in its scheduling decisions in order to alleviate network traffic and ameliorate MapReduce performance.

MAESTRO Scheduling Algorithm
The Maestro is a replica aware scheduler algorithm introduced by S. Ibrahim, H. Jin, et al. This technique defers the difficulty of non-local tasks that depend on replica of map tasks [43]. The Maestro keeps track of the chunks and duplicate locations, along with the number of other chunks hosted by each node. This will assistance maestro to launch the map tasks without making an impression other nodes' tasks as well as depending on the number of map tasks that are hosted and on their input data duplicate scheme it fills the empty slots of the node. The likelihood of queuing a map task based on duplicate of the tasks on the given machine, execution scheduling is calculated. It does the work in two ripple firstly stow the empty slots of each data node based on the number of hosted map task and on the duplicate scheme for their input data. Secondly execution scheduling takes into account the likelihood of scheduling a map task on a given machine depending on the duplicate of the task's input data. These two ripples lead to higher locality in the running of map tasks.

MapReduce with Communication Overlap Scheduling Algorithm
The MapReduce with communication overlap scheduling algorithm initiate by F. Ahmad, S. Lee, et al. MapReduce with communication overlap scheduling to instate nearly full overlap through the modern scheme of including the sort and reduce in the overlap [44]. The elementary Hadoop data flow was modified assent the operation of Reduce tasks on fractional data. MapReduce with communocation overlap breaks make smaller into many smaller invocations on fractional data from some map tasks, and a concluding reducing step re-reduces all the fractional

reduce outputs to produce the concluding output. MapReduce with communication overlap approach of hiding the latency of the on a mandatory basis high shuffle volume of shuffle-heavy Map Reductions are elemental for attaining performance.

Center-of-Gravity Reduce Scheduling Algorithm

The M. Hammoud, M. Rehman, et al. developed Center-of-Gravity Reduce Scheduling Algorithm [45]. This technique to designs a locality-aware, skew-aware reduce task scheduler for stop MapReduce network traffic. The Center-of-Gravity reduce task scheduling effort to schedule every reduce job, at its center-of-gravity node determined by the network position of alimentation nodes and the skew in the sizes of alimentation map tasks split. The network is normally a congestion in MapReduce-based systems and scheduling reducers at their center-of-gravity nodes, we dissert for reduced network traffic, which can perchance permit more MapReduce jobs to stick together on the same system. Center-of-Gravity reduce task scheduling controllably cast aside scheduling skew, a circumstance where some nodes receive more reduce tasks than others, and encourage pseudo-asynchronous map and reduce phases and in scheduling reducers at their nodes, they dissert for diminishing network traffic, which may perchance assent more MapReduce task to stick together on the identical system.

Self-Adaptive Reduce Scheduling Algorithm

The Z. Tang, L. Jiang, et al. introduced Self-Adaptive Reduce scheduling algorithm. The most important aim of Self-Adaptive Reduce scheduling algorithm analyses the map and reduce phases in detail and it is capable of [46] determine dynamically the start time of the decrease task depending on the size of the output of map tasks. The Self-Adaptive Reduce scheduling algorithm rise the resource utilization rate by keeping down reduce tasks expectant time. In this algorithm, firstly by keeping down tasks read the each map output from data blocks. This is copy phase. Secondly, these outputs from map phase are ordered and combine. This data are split into two types, one is data in memory and other is data in the circular buffer. The memory that is assigned to decrease tasks is limited, which is why the buffer data must be written to disk regularly. In exterior sorting, the new data must combine with data on the disks frequently. Supposing there are a huge number of map tasks or huge sized map tasks, exterior sorting must be done many times as well as copy stage and sort stage together is called shuffle stage. This algorithm proved that the average reaction time has fallen 12% to 30% when SARS algorithms are applied conventional job scheduling algorithm such as that capacity scheduler, FIFO and fair scheduler.

COSHH Scheduling Algorithm

The A. Rasooli, D.G. Down, et al. developed new scheduling algorithm that called COSHH, which is implemented and executed for Hadoop, it contemplate heterogeneity at both cluster and application levels [47]. The most important goal of COSSH uses system information and takes scheduling judgment which enhance the performance of the overall system. The two main steps in COSSH are, assign to the job in a suitable queue when the new job coming and allocate job to free resource upon receiving a heartbeat. The scheduler triggers the routing process to allocate a job to the existing free resource. This COSHH algorithm is proposed to ameliorate the mean ending time of jobs.

Weighted Round-Robin Scheduling Algorithm

The Y. Zhang, P.G. Harrison, et al. evolved Weighted Round-Robin scheduling algorithm [48]. The main concept of our weighted round-robin policy is to assign a weight to each queue, then scheduling tasks of various sub-queue according to weight. In respect of Weighted Round-Robin scheduling algorithm, it can endow better fairness when the size of each task is same. Forasmuch, it will bring unfairness for the smaller queues if the size of the task is incompatible. The algorithm is simple to be implemented with a small cost and appropriate for the Hadoop platform.

Natjam Scheduling Algorithm

The B. Cho, M. Rahman, et al. developed new scheduling algorithm that called Natjam [49]. The most important aim of Natjam endows random job priorities, arduous actual time scheduling and more efficient misappropriation for MapReduce clusters those are resource constrained. In the MapReduce skeleton jobs comes with arduous priority, high priority job (small closure time) and small priority jobs(long closure time) the traditional way to extricate the issue of priority is setting various clusters, which later lead to the incapacitated resource utilization and long job closure time. The algorithm goal, firstly to execute all the types of job, regardless of priority, closure time in the same MapReduce cluster. Secondly achieve a less closure time for higher priority jobs and thirdly optimizing closure times of bottommost priority jobs. The Natjam perform superior than current viewpoint for a diversification of clusters, under actual workloads and sizes.

DynMR Scheduling Algorithm

The J. Tan, A. Chin, et al. introduced new scheduling algorithm that called DynMR [50]. The primary goal of DynMR is used to enhance the performance of MapReduce. The consequential issue that persists in the current MapReduce implementation. Firstly the problem in selecting par excellence performance parameters for a single job in a fixed, committed environment, and lack of ability to configure parameters that can perform optimally in a dynamic, multi-job cluster; secondly the long job execution resulting from a task long-tail effect, often caused by reduce task data skew or heterogeneous computing nodes and thirdly ineffectual use of hardware resources. The DynMR uses the interleave way of execution where many partially accomplished reduce tasks and map tasks executes. It be made up of three components. Firstly a running reduce task uses a detection algorithm to recognize resource underutilization during the shuffle phase. It then gives up the assign hardware resources efficiently to the next task. Secondly a number of reduce task is gradually amassed in a progressive queue, according to a flow control algorithm in runtime. These tasks execute in an interleaved staggering. Moreover reduce task can be put adaptively to the progressive queue if the full fetching capacity is not attainable. Thirdly merge threads of each decrease task are extracted out as standalone services within the associated JVM. This design permits the data segments of several partially-complete reduce task to live in the same JVM heap.

Throughput Driven Task Scheduling Algorithm

The X. Wang, D. Shen, et al. evolved Throughput Driven Task Scheduler algorithm [51]. In view of the fact that, task scheduling in MapReduce is very vital for the job execution and has a marked influence on the system performance. To the best of our comprehension, the foregoing scheduling algorithms rarely consider the job-intensive environments and are not able to provide high system throughput. Here this Scheduler algorithm offers a way for enhancing throughput using job intensive scheduling. A latest job scheduling technique is offered, throughput driven task scheduler for obtaining high system the throughput in the job intensive MapReduce environment and this algorithm abbreviate many aspects which can influence the throughput of a job intensive MapReduce environment.

CooMR Scheduling Algorithm

The X. Li, Y. Wang, et al. developed new scheduling algorithm that called CooMR. The main goal of cross-task coordination CooMR is designed for efficient data management in MapReduce programs. The CooMR be made up of three component schemes [52] including cross-task opportunistic memory sharing and log-structured I/O consolidation, which are designed to make easier task coordination, and the key-based in-situ merge algorithm which is designed to make able the merging & sorting of Hadoop intermediate data without in fact moving the <key, value> pairs. The CooMR leads to enhancement task coordination, intensify system resources throughput and outstandingly ameliorate the process time of MapReduce jobs.

ARIA Scheduling Algorithm

The B A. Verma, L. Cherkasova, et al. introduced new algorithm that called ARIA [53]. The matter in the shared MapReduce cluster is to keep track of the resource allocation of various applications for attaining their performance aim. The connatural Hadoop scheduler, no such job scheduler is present that can suitably allocate the resources for the job once given a job closure deadline. The primary aim of Natjam endows AIRA deals with the above mentioned difficulty. ARIA be made up of three interrelated components. At the beginning, a job that is continuously executed on a new dataset, job profile is maintained, which contains the info about Map and Reduce tasks. Eventually, the MapReduce performance model is made in estimating the amount of resources for job ending.

HScheduler Scheduling Algorithm

The W. Tian, G. Li, et al. presented HScheduler algorithm [54]. The most important aim of HScheduler is to design the MapReduce scheduler, which detract the make span of the jobs. The MapReduce cluster, the running time of the job depends on the way the map task is scheduled, the overall make span and the resource utilization depends on the scheduling of the map and reduce tasks and via comprehensive actual data tests, the HScheduler has superior performance than the best-known technique by 10.8–11.9 % on average for offline scheduling and 9–11 % on average for online scheduling. The HScheduler can be applied to ameliorate accountable time, energy dexterity in cloud computing and throughput.

VI. The Optimization Techniques for Scheduling

The scheduling in big data platforms is the primary building block for making data centers more available to applications and user group. An instance of optimization is multiobjective and multiconstrained scheduling of many tasks in Hadoop or improve small jobs with Hadoop. The optimization policy for scheduling are specific to each model and for every type of application [55]. The most used techniques for scheduling optimization are as follows.

Linear Programming: Linear programming is the process of taking different linear inequalities relating to some circumstance, and discovers the best value obtained under those conditions. In real life, linear programming is part of a very vital area of mathematics called optimization techniques. This field of study is used every day in the organization and allocation of resources. Linear programming permits the scheduler to discover the appropriate resources or cluster of resources, based on defined constraints.

Dynamic Splits: Dynamic splits divided intricate applications in a cluster of tasks and schedule every cluster with a specific scheduling algorithm.

The Combinatorial Optimization: The Combinatorial optimization techniques are the discovery for maxima or minima of an objective function whose domain is a discrete unless huge configuration space as opposed to an N-dimensional continuous space and its discover an optimal allocation solution for a finite set of resources. This is a time-consuming technique, and it is not suitable for real-time processing.

Task-Oriented Optimization: Task-oriented optimization considers the task’s properties, coming time (slack optimization), and frequency (for periodic tasks).

Stochastic Optimization: Stochastic optimization uses random variables that become visible in the formulation of the optimization issue itself and are used especially for applications that have a deterministic behavior, that is, a normal distribution (for periodic tasks) or Poisson distribution (for sporadic tasks).

Resource-Oriented Optimization: Resource-oriented optimization considers ending time constraints while making the decision to maximize resource utilization.

Evolutionary Optimization: Evolutionary optimization goal to discover an optimal configuration for a specific system within specific constraints and consists of specific bio-inspired techniques, namely ant and bee computing, particle swarm optimization, immune and genetic algorithms. These techniques normally explore a near-optimal solution for the scheduling issue.

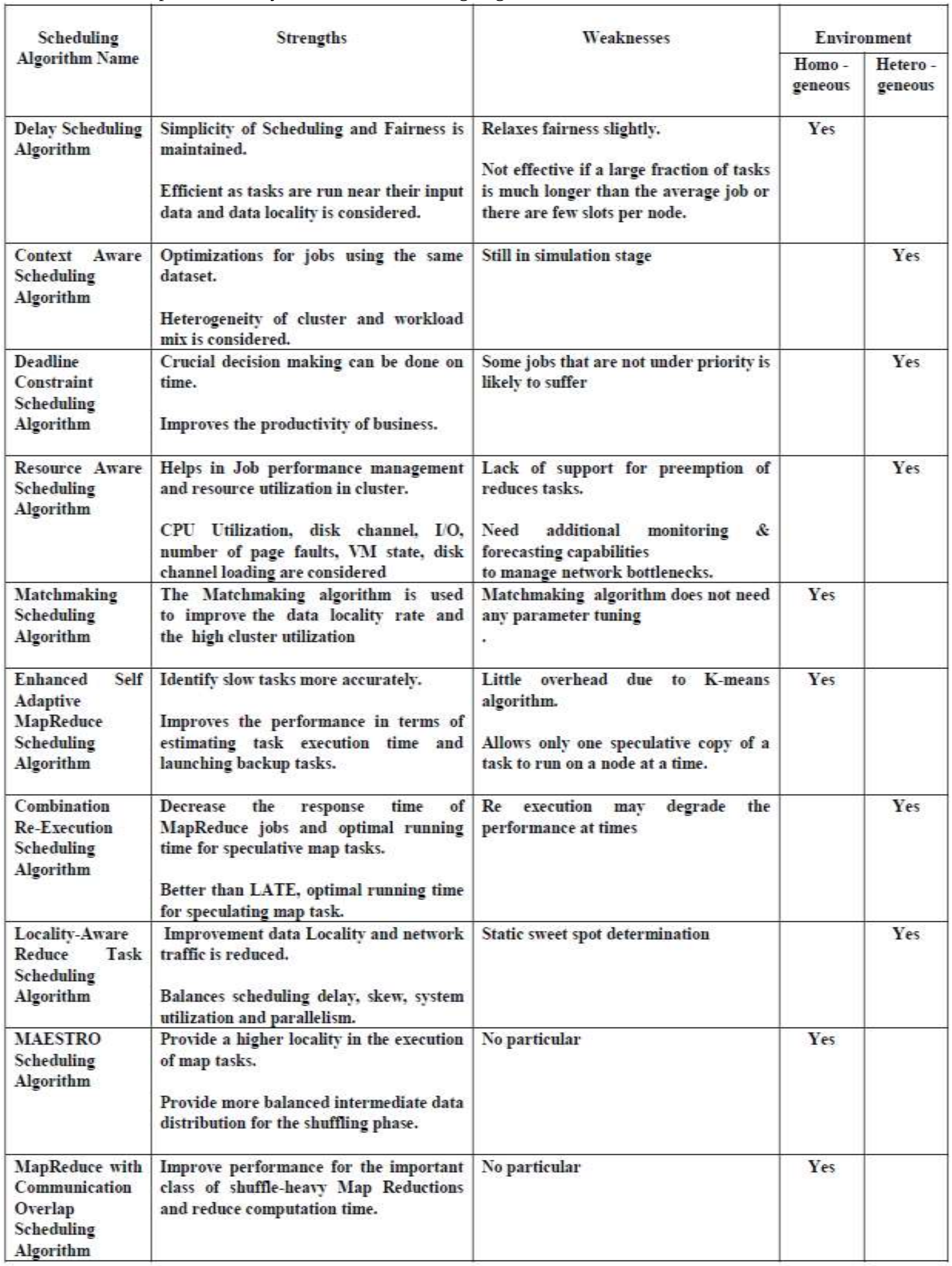
VII. The Comparative Analysis Various Scheduling Algorithms in Big Data Territory

In this section, we are comparative analysis various scheduling algorithms in the Big Data territory has shown in below table1, table2, and table 3. In addition, this paper paraphrases the Strengths, Weaknesses, and Environment of scheduling algorithms on Big Data.

Table 1:- The Comparative Analysis Various Scheduling Algorithms

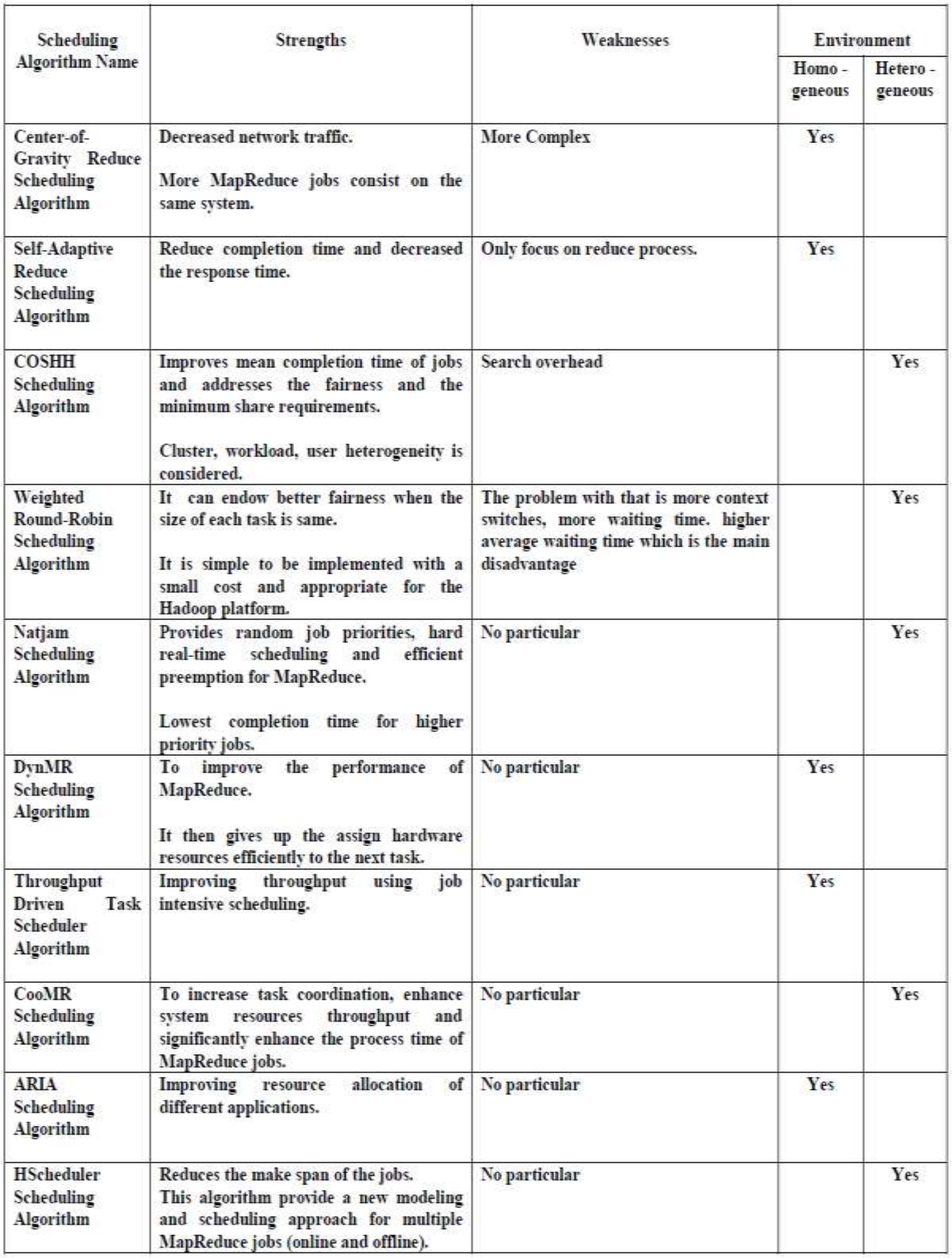
Scheduling Algorithm Name	Strengths	Weaknesses	Environment	
			Homo - geneous	Hetero - geneous
Longest Approximate Time to End (LATE) Scheduling Algorithm	Speculatively executes only tasks that will improve job response time, rather than any slow tasks. LATE scheduler optimizes performance of jobs and minimizes job response time as much as possible. This scheduler technique is highly robust in terms of heterogeneity.	The poor performance due to the static manner in computing the progress of the tasks. Not always reliable due to bugs in tasks. This scheduler technique does not guarantee reliability.		Yes
FIFO Scheduling Algorithm	FIFO scheduling technique is the simplest and most efficient among all the schedulers and high throughput. The cost of entire cluster scheduling process is less. The jobs are executed in the same order in which they are submitted.	Low performance when run multiple types of jobs. No heterogeneity of workload and performance constraints is considered. A major drawback of FIFO scheduling is that it is not pre-emptive. Therefore, it is not suitable for interactive jobs.	Yes	
Hybrid Scheduling Algorithm	This is a fast and flexible scheduler. Improves response time for multi-user Hadoop environments.	No particular		Yes
Fair Share Scheduling Algorithm	it can provide fast response times for small jobs mixed with larger jobs and .less complex. It has the ability to fix the number of concurrent running jobs from each user and pool.	Larger average completion time and more complicated configurations. This scheduler does not consider the weight of each job, which leads to unbalanced performance in each pool/node. Pools have a limitation on the number of running jobs under fair scheduling.		Yes
Self-Adaptive MapReduce (SAMR) Scheduling Algorithm	Improve the overall MapReduce performance in the heterogeneous environments. Uses historical information to tune weights of map and reduce stages.	Do not consider the data locality management for launching backup tasks. It does not consider that the dataset sizes and the job types can also affect the stage weights of map and reduce tasks.		Yes
Capacity Scheduling Algorithm	Capacity scheduling policy maximizes utilization of resources and throughput in cluster environment. This scheduler guarantees the reuse of the unused capacity of the jobs within queues. It also supports the features of hierarchical queues, elasticity, and operability and it can allocate and control memory based on the available hardware resources.	The most complex among three schedulers and configuration complicated. User needs to know system information and make queue set for the job. With regard to pending jobs it has some limitations in ensuring stability and fairness of the cluster from a queue and single user.		yes

Table 2:- The Comparative Analysis Various Scheduling Algorithms



Scheduling Algorithm Name	Strengths	Weaknesses	Environment	
			Homo - geneous	Hetero - geneous
Delay Scheduling Algorithm	Simplicity of Scheduling and Fairness is maintained. Efficient as tasks are run near their input data and data locality is considered.	Relaxes fairness slightly. Not effective if a large fraction of tasks is much longer than the average job or there are few slots per node.	Yes	
Context Aware Scheduling Algorithm	Optimizations for jobs using the same dataset. Heterogeneity of cluster and workload mix is considered.	Still in simulation stage		Yes
Deadline Constraint Scheduling Algorithm	Crucial decision making can be done on time. Improves the productivity of business.	Some jobs that are not under priority is likely to suffer		Yes
Resource Aware Scheduling Algorithm	Helps in Job performance management and resource utilization in cluster. CPU Utilization, disk channel, I/O, number of page faults, VM state, disk channel loading are considered	Lack of support for preemption of reduces tasks. Need additional monitoring & forecasting capabilities to manage network bottlenecks.		Yes
Matchmaking Scheduling Algorithm	The Matchmaking algorithm is used to improve the data locality rate and the high cluster utilization	Matchmaking algorithm does not need any parameter tuning .	Yes	
Enhanced Self Adaptive MapReduce Scheduling Algorithm	Identify slow tasks more accurately. Improves the performance in terms of estimating task execution time and launching backup tasks.	Little overhead due to K-means algorithm. Allows only one speculative copy of a task to run on a node at a time.	Yes	
Combination Re-Execution Scheduling Algorithm	Decrease the response time of MapReduce jobs and optimal running time for speculative map tasks. Better than LATE, optimal running time for speculating map task.	Re execution may degrade the performance at times		Yes
Locality-Aware Reduce Task Scheduling Algorithm	Improvement data Locality and network traffic is reduced. Balances scheduling delay, skew, system utilization and parallelism.	Static sweet spot determination		Yes
MAESTRO Scheduling Algorithm	Provide a higher locality in the execution of map tasks. Provide more balanced intermediate data distribution for the shuffling phase.	No particular	Yes	
MapReduce with Communication Overlap Scheduling Algorithm	Improve performance for the important class of shuffle-heavy Map Reductions and reduce computation time.	No particular	Yes	

Table 3:- The Comparative Analysis Various Scheduling Algorithms



Scheduling Algorithm Name	Strengths	Weaknesses	Environment	
			Homo - geneous	Hetero - geneous
Center-of-Gravity Reduce Scheduling Algorithm	Decreased network traffic. More MapReduce jobs consist on the same system.	More Complex	Yes	
Self-Adaptive Reduce Scheduling Algorithm	Reduce completion time and decreased the response time.	Only focus on reduce process.	Yes	
COSHH Scheduling Algorithm	Improves mean completion time of jobs and addresses the fairness and the minimum share requirements. Cluster, workload, user heterogeneity is considered.	Search overhead		Yes
Weighted Round-Robin Scheduling Algorithm	It can endow better fairness when the size of each task is same. It is simple to be implemented with a small cost and appropriate for the Hadoop platform.	The problem with that is more context switches, more waiting time, higher average waiting time which is the main disadvantage		Yes
Natjam Scheduling Algorithm	Provides random job priorities, hard real-time scheduling and efficient preemption for MapReduce. Lowest completion time for higher priority jobs.	No particular		Yes
DynMR Scheduling Algorithm	To improve the performance of MapReduce. It then gives up the assign hardware resources efficiently to the next task.	No particular	Yes	
Throughput Driven Task Scheduler Algorithm	Improving throughput using job intensive scheduling.	No particular	Yes	
CooMR Scheduling Algorithm	To increase task coordination, enhance system resources throughput and significantly enhance the process time of MapReduce jobs.	No particular		Yes
ARIA Scheduling Algorithm	Improving resource allocation of different applications.	No particular	Yes	
HScheduler Scheduling Algorithm	Reduces the make span of the jobs. This algorithm provide a new modeling and scheduling approach for multiple MapReduce jobs (online and offline).	No particular		Yes

Conclusion:-

Today, almost everyone is connected to the Internet and uses different Cloud solutions to deliver, store, and process data. The rapid growth of data volume requires processing of petabytes of data per day. Many applications generate big data, like social networking and social influence programs, Cloud applications, public websites, scientific experiments and simulations, data warehouses, monitoring platforms, and e-government services. To process big data proper scheduling is required to achieve greater performance. The main purpose of scheduling in big data platforms is to the processing and completion of as numerous tasks as possible by managing and transforming data in a potential way with a less number of transmigration. There are various factors that affect the performance of scheduling policies such as data volume, format of data sources, data velocity, security and privacy, cost, connectivity, and data sharing. The aim of scheduling of jobs is to enable faster processing of jobs and to reduce the response time as much as possible by using better techniques for scheduling depending on the jobs, along with the best utilization of resources. We explained several popular scheduling algorithms in this field. In the end, in this paper, we are extensive overview of job scheduling, classification of the scheduler, and comparison of various existing algorithms with benefit, deficiency, limitations in the big data territory.

References:-

1. Kim, G.-H., Trimi, S., & Chung, J.-H. (2014). Big-data applications in the government sector. Communications of the ACM, 57(3), pp 78–85.
2. S. Suthaharan, "Big data classification: Problems and challenges in network intrusion prediction with machine learning", ACM SIGMETRICS Performance Evaluation Review 41.4, pp. 70-73, 2014.
3. Dr. Yusuf Perwej, "An Experiential Study of the Big Data," for published in the International Transaction of Electrical and Computer Engineers System (ITECES), USA, ISSN (Print): 2373-1273 ISSN (Online): 2373-1281, Vol. 4, No. 1, page 14-25, March 2017, DOI:10.12691/iteces-4-1-3.
4. Y oo, D., K. M. Sim. "A Comparative Review of Job Scheduling for MapReduce." , In: IEEE International Conference on Cloud Computing and Intelligence Systems, 2011, pp. 353-358.
5. Nikhat Akhtar, Firoj Parwej, Dr. Yusuf Perwej, "A Perusal Of Big Data Classification And Hadoop Technology," for published in the International Transaction of Electrical and Computer Engineers System (ITECES), USA, ISSN (Print): 2373-1273 ISSN (Online): 2373-1281, Vol. 4, No. 1, page 26-38, May 2017, DOI: 10.12691/iteces-4-1-4**.**
6. M. Khan, M. Li, P. Ashton, G. Taylor, and J. Liu, "Big data analytics on PMU measurements," in Fuzzy Systems and Knowledge Discovery (FSKD), 2014 11th International Conference on, 2014, pp. pp 715–719.
7. Z. Tang, L. Jiang, J. Zhou, K. Li, and K. Li, " A self -adaptive scheduling algorithm for reduce start time ", Future Generation Co mputer Systems, 2014.
8. Rasooli, D. G. Down, "A hybrid scheduling approach for scalable heterogeneous hadoop systems", High Performance Computing Networking Storage and Analysis (SCC) 2012 SC Companion:. IEEE, pp. 1284-1291, 2012.
9. Guilherme W. Cassales, Andrea S. Char~ao, Manuele Kirsch Pinheiro, Carine Souveyet, Luiz A. Steffenel, Context-aware scheduling for apache hadoop over pervasive environments, Procedia Comput. Sci. 52 (2015) 202–209.
10. Firoj Parwej, Nikhat Akhtar, Dr. Yusuf Perwej, "A Close-Up View About Spark in Big Data Jurisdiction," for published in the International Journal of Engineering Research and Applications (IJERA), ISSN: 2248-9622 (Online), www.ijera.com, Vol. 8, Issue 1, (Part -I1), page 26-41, January 2018.
DOI: 10.9790/9622-0801022641
11. Yongcai Tao, Qing Zhang, Lei Shi, Pinhua Chen, "Job Scheduling Optimization for Multi-User MapReduce Clusters", 2011 Fourth International Symposium on Parallel Architectures, Algorithms and programming, IEEE, Pg 213-217, 978-0-76954575-2/11, DOI 10.1109/PAAP.2011.33
12. Songcheng Jin, Shuqiang Yang, Yan Jia, "Optimization of Task Assignment Strategy for Map-Reduce", 2012 2nd International Conference on Computer Science and Network Technology, 978-1-4673-2964-4/12, pg 57-61, IEEE 2012, Changchun, China.
13. L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, and F. C. M. Lau. Moving big data to the cloud: An online cost-minimizing approach. Selected Areas in Communications, IEEE Journal on, vol. 31, no. 12, pp. 2710–2721, 2013
14. S. Liu, J. Xu, Z. Liu, X. Liu, Evaluating task scheduling in hadoop-based cloud systems, in: 2013 IEEE International Conference on Big Data, IEEE, 2013, pp. 47–53.
15. Lisia S. Dias, Marianthi.G. Ierapetritou, Integration of scheduling and control under uncertainties: review and challenges, Chem. Eng. Res. Des. 116 (December 2016) pp. 98–113.

16. Xu X, Cao L, Wang X. Adaptive task scheduling strategy based on dynamic workload adjustment for heterogeneous Hadoop clusters. IEEE Systems Journal. 2016 Jun;10(2):pp 471- 482
17. C. He et al., "Matchmaking: A new MapReduce scheduling technique", Cloud Computing Technology and Science (CloudCom) 2011 IEEE 3rd Int. Conf., 2011.
18. X. Qin, H. Jiang, "A dynamic and reliability-driven scheduling algorithm for parallel real-time jobs executing on heterogeneous clusters", J. Parallel Distrib. Comput., vol. 65, no. 8, pp. 885-900, Aug. 2005.
19. X Bu, J Rao, C Xu, Interference and Locality-Aware Task Scheduling for MapReduce Applications in Virtual Clusters HPDC'13, New York, NY, USA:Copyright 2013 ACM 978–1-4503-1910-2/13/06, pp. 17-21, June 2013.
20. Kao Yu-Chon, Chen Ya-Shu, Data-locality-aware mapreduce realtime scheduling framework The Journal of Systems and Software, vol. 112, pp. 65-77, 2016.
21. H. Chang, M. Kodialam, R. R. Kompella, T. Lakshman, M. Lee, S. Mukherjee, "Scheduling in mapreduce-like systems for fast completion time", IEEE INFOCOM. IEEE, 2011.
22. Casavant et al., "A taxonomy of scheduling in general-purpose distributed computing systems", Software Engineering IEEE Transactions on 14**,** no. 2, pp. 141-154, 1988.
23. P. Bellavista et al., "Priority-based Resource Scheduling in Distributed Stream Processing Systems for Big Data Applications", ACM 7th International Conference on Utility and Cloud Computing, 2014.
24. Rasooli, D. G. Down, "A hybrid scheduling approach for scalable heterogeneous hadoop systems", High Performance Computing Networking Storage and Analysis (SCC) 2012 SC Companion:. IEEE**,** pp. 1284-1291, 2012.
25. 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.
26. P. Nguyen, T. Simon, M. Halem, D. Chapman and Q. Le, "A hybrid scheduling algorithm for data intensive workloads in a MapReduce environment", In: Proceedings of the 2012 IEEE/ ACM fifth international conference on utility and cloud computing. Washington, DC, USA: IEEE computer society; UCC'12, 2012, pp. 161-168.
27. Rasooli, D. G. Down, "A Hybrid Scheduling Approach for Scalable Heterogeneous Hadoop Systems", High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion, IEEE International Conference on Salt Lake City, UT, USA, pp. 1284-1291, 10-16 Nov. 2012, DOI: 10.1109/SC.Companion.2012.155
28. Ya-Wen Cheng, Shou-Chih Lo, "Improving Fair Scheduling Performance on Hadoop", Platform Technology and Service (PlatCon), IEEE International Conference on Busan, South Korea, 13-15 Feb. 2017, DOI: 10.1109/PlatCon.2017.7883710
29. G.Sharma, A. Ganpati, "Performance evaluation of fair and capacity scheduling in Hadoop YARN", Green Computing and Internet of Things (ICGCIoT), IEEE International Conference on Noida, India, Pages: 904 - 906 , 8-10 Oct. 2015
DOI: 10.1109/ICGCIoT.2015.7380591
30. 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.
31. C. Tian, H. Zhou, Y. He, and L. Zha, "A dynamic MapReduce scheduler for heterogeneous workloads," in Proceedings of the 2009 Eighth International Conference on Grid and Cooperative Computing-Volume 00, pp. 218–224, IEEE Computer Society, 2009.
32. 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, pp. 265–278.
33. Qiaomin Xie, Mayank Pundir, Yi Lu, Cristina L. Abad, Roy H. Campbell, Pandas: robust locality-aware scheduling with stochastic delay optimality, IEEE/ACM Trans. Netw. pp.1–14, 2016, doi.org/10.1109/TNET.2016.2606900.
34. K.A Kumar, V.K Konishetty, K. Voruganti and G. Rao, "CASH: context aware scheduler for Hadoop", In: Proceedings of the international conference on advances in computing, communications and informatics. New York, NY, USA: ACM; 2012. p. 52–61.
35. Cassales GW, Charao AS, Pinheiro MK, Souveyet C, Steffenel LA, "Context-Aware Scheduling for Apache Hadoop over Pervasive Environments‖", The 6th International Conference on Ambient Systems, Networks and Technologies, Procedia Computer Science, Vol- 52, pp. 202 – 209, ISSN: 1877- 0509, 2015

36. K. Kc and K. Anyanwu. "Scheduling hadoop jobs to meet deadlines." Cloud Computing Technology and Sci. (CloudCom), 2010 IEEE 2nd Int. Conf. IEEE, pp. 388-392, 2010.
37. Dazhao Cheng, Jia Rao, Changjun Jiang, Xiaobo Zhou, Resource and deadline aware job scheduling in dynamic hadoop clusters, in: IEEE 29th International Parallel and Distributed Processing Symposium, 2015.
38. Polo J, Castillo C, Carrera D, Becerra Y, Whalley I, Steinder M, Torres J, Ayguade E ,” Resource-Aware Adaptive Scheduling for MapReduce Clusters‖ Middleware”, LNCS 7049, pp. 187–207, 2011, ISSN: 0302-9743.
39. C. He, Y. Lu, and D. Swanson, “Matchmaking: A new Mapreduce scheduling technique,” Cloud Computing Technology and Sci (CloudCom), 2011 IEEE 3rd Int. Conf. on IEEE, pp. 40-47, 2011.
40. Xiaoyu Sun, Chen He and Ying Lu “ESAMR: An Enhanced Self-Adaptive MapReduce Scheduling Algorithm”(2012) IEEE 18th International Conference on Parallel and Distributed Systems.
41. L. Lei, T. Wo and C. Hu, “ CREST: Towards fast speculation of straggler tasks in MapReduce”, In: The eighth international conference on e-business engineering. IEEE; 2011, pp. 311-316
42. M. Hammoud and M. Sakr, “ Locality-aware reduce task scheduling for MapReduce”, In: The third international conference on cloud computing technology and science. IEEE, 2011, p. 570–576.
43. S. Ibrahim, H. Jin, L. Lu, B. He, G. Antoniu and S. Wu,“ Maestro: replica-aware map scheduling for MapReduce”, In: The 12th international symposium on cluster, cloud and grid computing. IEEE/ACM; 2012, p. 435– 477.
44. F. Ahmad, S. Lee, M. Thottethodi and T. Vijaykumar, “ MapReduce with communication overlap (MARCO) ”, J Parallel Distrib Comput, Vol. 73, NO. 5, 2013, pp. 608–628.
45. M. Hammoud, M. Rehman and M. Sakr, “Center-of-Gravity reduce task scheduling to lower MapReduce network traffic”, In: International conference on cloud computing. IEEE, 2012, p. 49–58.
46. 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.
47. 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.
48. Y. Zhang, P.G. Harrison, “Performance of a priority weighted round robin mechanism for differentiated service networks,” In the Proc. of 16th International Conference on Computer Communications and Network (ICCCN), 1198-1203.2007
49. B. Cho, M. Rahman, T. Chajed, I. Gupta, C. Abad, N. Roberts and P. Lin, ”Natjam: design and evaluation of eviction policies for supporting priorities and deadlines in MapReduce clusters”, Proceeding SOCC '13 Proceedings of the 4th annual Symposium on Cloud Computing Article No. 6 , doi:10.1145/2523616.2523624
50. J. Tan, A. Chin, Z. Z. Hu, Y. Hu, S. Meng, X. Meng and L. Zhang, “DynMR: dynamic MapReduce with ReduceTask interleaving and MapTask backfilling”, April 2014, EuroSys '14: Proceedings of the Ninth European Conference on Computer Systems, doi:10.1145/2592798.2592805
51. X. Wang, D. Shen, G. Yu, T. Nie and Y. Kou, “A Throughput Driven Task Scheduler for Improving MapReduce Performance in Job-Intensive Environments”, 2013 IEEE International Congress on Big Data, 2013, pp: 211 -218 , doi:10.1109/BigData.Congress.2013.36
52. X. Li, Y. Wang, Y. Jiao, C. Xu and W. Yu,”CooMR: cross-task coordination for efficient data management in MapReduce programs”, Proceeding SC '13 Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, ISBN: 978-1-4503-2378-9, Article No. 42, November 17 - 21, 2013, doi:10.1145/2503210.2503276
53. Verma, L. Cherkasova and R. H. Campbell, “ARIA: automatic resource inference and allocation for MapReduce environments”, Proceeding ICAC '11 Proceedings of the 8th ACM international conference on Autonomic computing, ISBN: 978-1-4503-0607-2 ,Pages 235-244, 2011., doi:10.1145/1998582.1998637
54. W. Tian, G. Li, W. Yang and R. Buyya, “HScheduler: an optimal approach to minimize the make span of multiple MapReduce jobs”, The Journal of Supercomputing June 2016, volume 72, Issue 6, pp 2376–2393, doi:10.1007/s11227-016-1737-4
55. K. Elmeleegy. Piranha: Optimizing short jobs in Hadoop. Proceedings of the VLDB Endowment, vol. 6, no. 11, pp. 985–996, 2013.