ijecs open access www.ijecs.in

International Journal Of Engineering And Computer Science ISSN:2319-7242

Volume 5 Issue 8 August 2016 Page No. 17737-17743

Scheduling Algorithms in Big Data: A Survey

Nagina, Dr. Sunita Dhingra

Research Scholar,
CSE, UIET,
MDU, ROHTAK
nagina260@gmail.com
Assistant Professor,
CSE, UIET,
MDU, ROHTAK
sunitadhingramdu@rediffmail.com

Abstract- **It is not easy to quantify the large amount of data stored electronically. Data is in the unit of zettabytes or exabytes referred as Big Data. Hadoop system is used to process large datasets. Map Reduce program is used to collect data according to the request. To process big data proper scheduling is required to achieve greater performance. Scheduling is a technique of assigning jobs to available resources in a manner to minimize starvation and maximize resource utilization. Performance of scheduling technique can be improved by applying deadline constraints on jobs. The objective is to study Map Reduce and different scheduling algorithms that can be used to achieve better performance.**

Keywords: Big Data; Hadoop; Map Reduce; HDFS; Scheduling Algorithms.

I. INTRODUCTION

Data exists in various forms (structured, unstructured and semi structured) generated from different fields like network, economic development and business was so massive that surmounts the competence to store and analyse [27]. Data has increased in various fields in large scale. Under the explosive increase in large scale, the term big data is used to define huge data sets. Big Data is high volume, high variety and high velocity information that require new processing techniques to enable improved decision making, process optimization and insight discovery. Big Data requires novel architecture, intelligent algorithms and processing techniques for proper scheduling in Hadoop [31]. Big data is characterized from the context of volume, velocity, veracity and variety [44]. Volume indicates the size of the data. Velocity means data are streaming at faster rates than that can be managed by conventional algorithms and systems. Veracity recommends that the quality of data, despite the data being available. So, one cannot presuppose that big data have higher quality. In fact, size raises quality issues, which desire to be either undertaken at the data pre-processing stage or by the learning algorithm. Variety represents different types of data and modalities for a given object. Big Data encompasses different challenges including analysis of data, capture and data curation, efficient storage, transfer and visualization of data along with the privacy and security of data [24]. Hadoop is an open source framework for processing big data using simple programming models. Components of Hadoop are Hadoop Distributed File System for storage and MapReduce for processing. MapReduce functionality depends on two functions: Map and Reduce function. Both functions are written by user. The Map function takes an input pair and generates a set of intermediate key/value pairs. The MapReduce library gathers all intermediate values associated with the same intermediate key and transfers them to the Reduce function. The Reduce function gets an intermediate key with associated set of values. It combines these associated values to make a smaller set of values.
The rest of this part is organized as follows. Section 2 describes Hadoop. Section 3 describes scheduling algorithms. Section 4 describes Performance Metrics. Section 5 describes performance issues. Section 6 describes observations. Section 7 describes conclusion.

II. HADOOP

Hadoop is an open-source framework, created by Doug Cutting, the creator of open source search technology. Hadoop allows to store and process big data in a distributed environment with asymmetrical clusters of computers using simple programming models. It is designed to expand from single server to thousands of machines, each machine offering local computation and storage [33]. Because of its distributed file system, it can run applications that involve thousands of nodes containing terabytes of data [48]. A single node failure doesn’t affect the ruinous system failure.

Steps of Hadoop Work [17]:

1) A job is submitted to the Hadoop job client for requisite process by user or application on defining the following items:
   a) The input and output files location in the distributed file system.
   b) The jar files comprising the implementation of map and reduce functions.
   c) The job configuration defined by setting different parameters specific to the job.

2) The job and job configuration is submitted to the JobTracker from Hadoop job client. Master Job Tracker distributes the software/configuration to the slaves, schedule tasks and monitor them, provide status and logical information to the job-client.

3) The Task Trackers on different nodes execute tasks according to MapReduce implementation and the output is stored into the output files on the HDFS file system.

Hadoop system can be depicted based on three factors: cluster, user and workload [13]. Each factor is either homogeneous or heterogeneous, which reflects the degree of heterogeneity.

Cluster: Cluster is a group of linked resources, where each resource (*Rj*) has a data storage unit and a computation unit. The computation unit consists of a set of slots with given execution rate. In Hadoop systems, each CPU core is referred as one slot. Similarly, the data storage unit has a given storage capacity and data retrieval rate. In Hadoop system, data is organized into files, which are usually large. Each file is partition into small pieces, called slices and all slices have the same size.

User: User submits jobs to the system. Hadoop allots minimum share and priority to each user on the basis of particular policy (e.g. the pricing policy). The user's minimum share is the minimum number of slots assured for the user at each point in time.

Workload: Workload is a set of jobs, where each job (*Ja*) has number of map tasks and reduce tasks. A map task performs a process on the slice, which have required data for this task. A reduce task processes the results of a subset of a job's map tasks. The value *m (Ja, Rb)* defines the mean execution time of job *Ja* on resource *Rb*. Hadoop workloads can classify into classes of "common jobs" .We define the class of common jobs are the set of jobs whose mean execution times (on each resource) are in the same range.

MapReduce

MapReduce is a framework for parallel processing of big data in a reliable, fault-tolerant manner on large clusters. The data have to be clustered based on their priorities, data dependence, deadline schedule for processing and processing of data clusters. For example, if processing of one data needs the output of another data as input then these can be combined to form a cluster [4, 39].

The MapReduce refers to the following two tasks that Hadoop programs perform [15]:

- *The Map Task:* It takes input data and transforms it into a set of data (key/value pairs).

- *The Reduce Task:* It takes set of data (key/value pairs) as input and combines those data tuples to form a smaller set [40].

The MapReduce framework consists of a master JobTracker and slave Task Tracker. The master JobTracker provides resource management such as schedule the tasks on the slave TaskTrackers, monitor and re-execute the failed tasks and track resource consumption/availability. The slave TaskTrackers execute tasks and give task-status information to the master periodically. There is a single point of failure, which means if JobTracker fails, all running jobs are halted. In case of a single node failure, map tasks and incomplete reduce tasks will be reexecuted instead of the complete map tasks and reduce tasks to achieve the minimum execution time [11, 46].

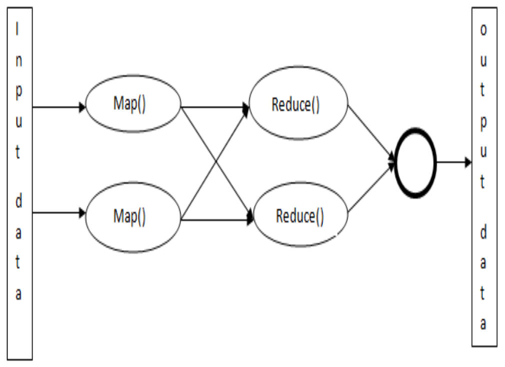
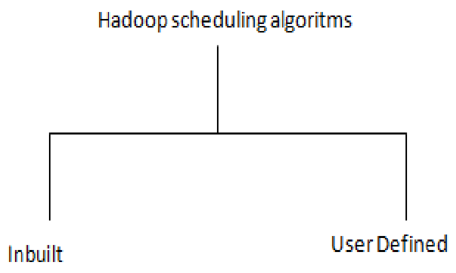


Fig: Execution flow of MapReduce.

Hadoop Distributed File System

Hadoop can use any of the distributed file system such as HFTP FS, Local FS, S3 FS, but the file system used by Hadoop is known as Hadoop Distributed File System (HDFS). The HDFS depends on the Google File System (GFS) and offers a distributed file system to run an application in a fault-tolerant and reliable manner on large clusters (thousands of computers) of small computer machines. HDFS uses master/slave architecture. Master is expressed by a single Name Node that stores the file system metadata. One or more slave DataNodes store the original data. The DataNodes perform read and write operation with the HDFS. They also perform block creation, replication and deletion based on instruction specified by NameNode. A file is partitioned into several blocks and those blocks are stored in set of DataNodes. The NameNode find out the mapping of blocks to the DataNodes [30, 39].

III. SCHEDULING ALGORITHM



a) Inbuilt Algorithm:

- *First In First Out (FIFO) Scheduling*

FIFO scheduling is based on queue mechanism. Job is divided into several tasks and submitted to free slots available on TaskTracker nodes. Jobs have to wait for execution due to acquisition of clusters. This leads to wait for other task for their turn [4]. All jobs need to complete in a time manner and provide better response time to every job [29].

- *Fair Scheduling*
  Facebook develops the Fair Scheduler to manage access the Hadoop cluster. The objective is to assign each user a fair share of the cluster capacity over a time. User group jobs in to job pools, with a guaranteed minimum number of Map and Reduce slots. It supports preemption i.e. to give the slots to the pool running under capacity, the scheduler forcefully kill tasks in job pools running over capacity. Priority is also assigned to various pools [4][9][29][49].

- *Capacity Scheduling*
  Yahoo developed the capacity scheduler to concentrate on a conventional situation where the number of users is large and the goal is to ensure a fair allocation of resources amongst users. Capacity scheduler shares fair percent of cluster. It supports FIFO scheduling within each queue with preemption. When a TaskTracker slot becomes free, the job with the lowest load and lowest arrival time is chosen. A task is then scheduled from that job [9, 29,50].

b) User Defined Algorithm:

- *Dynamic Proportional Scheduling:*
  According to Sandholm and Lai (2010), Dynamic Proportional Scheduling provides more job sharing and prioritization that result in increasing share of cluster resources and more differentiation in service levels of different jobs. This algorithm improves response time for multi-user Hadoop environments. Zhang et al. (2014**),** provides scheduling of multitasking and dynamic workloads for big-data analytics, it requires a significant amount of parameter sweeping and iterations. Ordinal optimization using fast simulation and rough models is introduced to obtain suboptimal solutions in much shorter time frame. If the scheduling solution for each period is not the best, ordinal optimization can processed fast in an evolutionary and iterative way to capture the details of big-data workload dynamism [25].

- *Resource-Aware Adaptive Scheduling(RAS):*
  Polo et al. (2011) proposed resource-aware scheduling technique for Map Reduce with multi-job workloads that aim to improve resource utilization across machines while observing completion time. RAS dynamically determines the number of job slots, and their position in the cluster at run-time.RAS provides scheduler with the adaptability needed for respond to changing conditions in resource demand and availability. In RAS three resource capacities were considered: CPU, memory and I/O. Zhao et al. (2013) gives task scheduling algorithm based on resource attribute selection (RAS) to determine its resource attributes by sending a set of test tasks to an execution node before a task is scheduled; and then select optimal node to execute a task according to resource requirements and appropriateness between the resource node and the task, which uses history task data if exists

- *MapReduce task scheduling with deadline constraints (MTSD) algorithm:*
  According to Tang et al. (2012), scheduling algorithm sets two deadlines: map-deadline and reduce-deadline. Reduce-deadline is just the users' job deadline. To get map-deadline, the proportion of Map task's time to the task's execution time should be needed. In a cluster with limited resources, Map and Reduce slot number is decided. For arbitrary submitted jobs with deadline constraints, the scheduler has to schedule the remaining resources in order to ensure that all jobs should be finished before the deadline constraints. Pop et al. (2014) presents the classical approach for aperiodic task scheduling by considering a scheduling system with different queues for periodic and aperiodic tasks and deadline as the main constraint, and develops a method to estimate the number of resources needed to schedule a set of aperiodic tasks, by considering both execution and data transfers costs. Based on a mathematical model, and by using different simulation scenarios, MTSD proved the following statements: (1) multiple source of independent aperiodic tasks can be considered approximating to a single one; (2) when the number of estimated resources go beyond a data center capacity, the tasks migration between different regional centers is the suitable solution with respect to the global deadline; and (3) in a heterogeneous data center, we need higher number of resources for the same request with respect to the deadline constraints. Wang and Li (2015) elaborated the task scheduling in MapReduce for distributed data centers on heterogeneous networks with adaptive heartbeats, job deadlines and data locality. Job deadlines are split according to the maximum data volume of tasks. With the considered constraints, the task scheduling is created as an assignment problem in each heartbeat, in which adaptive heartbeats are calculated by the processing times of tasks and jobs are sequencing in terms of the divided deadlines and tasks are scheduled by the Hungarian algorithm. On the basis of data transfer and processing times, the most suitable data center for all mapped jobs are determined in the reduce phase.

- *Delay Scheduling*
  The objective is to address the conflict between locality and fairness. When a node requests for a task, if the head-of-line job cannot project a local task, scheduler skip that task and looks at subsequent jobs. If a job has been skipped for long, we allow it to project non- local tasks, to avoid starvation. Delay scheduling temporarily relaxes fairness to improve locality by allowing jobs to wait for scheduling on a node with local data. Song et al. (2013) gave a game theory based method to solve scheduling problems by dividing a Hadoop scheduling problem into two levels—job level and task level. For

the job level scheduling, use a bid model to provide guarantee to the fairness and reduce the average waiting time. For tasks level, change scheduling problem into assignment problem and use Hungarian Method to optimize the problem. Wan et al. (2013) gives multi-job scheduling algorithm in MapReduce based on game theory which deals with the contest for resources between multiple jobs.

- *Multi Objective Scheduling:*
  Nita et al. (2015) explains scheduling algorithm named MOMTH by considering objective functions related to resources and users in the same time with constraints like deadline and budget. The implementation model consider as all MapReduce jobs are independent, there is no nodes failure before/during scheduling computation and scheduling decision is taken only based on the current knowledge. Bian et al. (2013) presents scheduling strategy based on fault tolerance. According to this scheduling strategy, the cluster finds the speed of the current nodes and creates some backups of the intermediate MapReduce data results on to a high performance cache server. The data produced by that node may go wrong soon. Thus the cluster may resume the execution to the previous level rapidly if there are several nodes going wrong, the Reduce nodes read the Map output from the cache server or from both the cache and the node, and keeps its high performance [14,16,43,45].

- *Hybrid Multistage Heuristic Scheduling (HMHS):*
  Chen et al. (2013) elaborates heuristic scheduling algorithm called HMHS, which tries to solve the scheduling problem by splitting it into two subproblems: sequencing and dispatching. For sequencing, we use a heuristic based on Pr*i* (the modified Johnson's algorithm). For dispatching, they recommend two heuristics Min-Min and Dynamic-Min-Min[38].

- *Utility-Driven Share Scheduling Algorithm:*
  Wan et al. (2013) provides utility-driven share scheduling algorithm by considering cost and time. Algorithm offers a global optimization scheduling scheme according to the workload of the job which maximize the user satisfaction. The scheduling algorithm consists of three functions: job added, assign task and utility. They would be called when a job added and response the heartbeat request of Task Tracker. When a new job arrives, first of all, we calculate a set of decision variables $P=\{ p1 , p2 ... pn \}$ using the utility as the optimization objective. Then, if the new utility greater than before, the resource should be allocated to each job depending on recalculated decision variable P. Otherwise, the new job will be rejected. When idle slot appears, calculate the hungry degree which is defined as the number of assigned slot divided by the number of required slot of each job and allocate the slot to the hungriest job.

- *Quality of Service (QoS) Based scheduling*
  Sandhu and Sood (2014) proposed QoS based scheduling for big data application towards distributed cloud datacenter at two levels - coarse grained and fine grained. On coarse grained level, accurate local datacenter is selected on the basis of network throughput, network distance between user and datacenter, available resources by using adaptive K nearest neighbor algorithm. On fine grained level, probability triplet (C, I, M) is predicted by using naïve Bayes algorithm which gives probability of new application in compute intensive (C), input/output intensive (I) and memory intensive (M) categories. Each datacenter is reformed into a pool of virtual clusters capable of executing jobs with some specific (C, I, M) requirements by using self organized maps. Novelty of study is to represent datacenter resources in a topological ordering and execute new incoming jobs in their respective predefined virtual clusters according to their respective QoS requirements.

- *Classification and Optimization based scheduler for Heterogeneous Hadoop (COSHH):*
  Pakize (2014) introduced a scheduling system that called COSHH, which is designed and implemented for heterogeneity at both application and cluster levels in Hadoop. The main approach is to use system information to make scheduling decisions better, which improves the performance. COSHH consists of two main processes, where each process is activated by receiving one of these messages. On receiving a new job, the scheduler performs the queuing process to store the incoming job in proper queue. On receiving a heartbeat message, the scheduler activates the routing process to assign a job to the free resource. COSHH improves the mean completion time of jobs.

- *MapReduce workflow scheduling algorithm (MRWS):*
  On analyzing the defects in system schedule in the heterogeneous cluster environment, Tang et al. (2014) gave MRWS algorithm, which consists of a job prioritizing phase and a task assignment phase. The job prioritizing phase can be classified as I/O-intensive and computing-intensive. The priorities of all jobs are estimated according to their corresponding types and then suitable slots are allocated for each block. The MapReduce tasks are scheduled with respect to data locality.

- *Longest Approximate Time to End(LATE) -Speculative Executions*
  Sometime task will be completed slowly, due to heavy load on the node, some failure or slow background processes. The technique works for MapReduce in heterogeneous environment. The scheduler finds real slow tasks by computing remaining time of all the tasks. The scheduler ranks tasks by estimated remaining time and starts a copy of the highest ranked task, which has a progress rate lower than the slow task threshold. LATE is based on three principles: prioritizing tasks, selecting fast nodes to run and restricting speculative tasks to prevent thrashing. This method would be optimal if all

nodes ran at consistent speeds and have no cost to launch a speculative task[32].

- *Service Level Agreement (SLA) scheduling:*
  Ayesha (2015) proposed SLA driven resource provisioning and scheduling in multiple data centre. The user requests in terms of SLA (deadline and budget) are stored at an entry point from where user request for user information is sent to cloud provider. The cloud provider receives SLA constraints and user's job details, then checks all the data centres for availability of resources and decide the data centre at which the user application can be installed without violating the SLA and budget constraints.

- *Max Percentages(MP) Algorithm*
  Li et al. (2015) gave MP algorithm, which consider the inherent correlation of information about resources and tasks and utilizes the percentages of resource service times to emphasize the overall performance. MP algorithm search the resource which is most affected on its service time if one task was mapped on it, and then allocate this task to it. The percentages of the completion time of each available task taken, the total used time of each resource to measure the effects on resources and the task which has the max percentage will be scheduled to be executed.

- *Context-Aware Scheduling:*
  According to Cassales et al. (2015), main goal is to improve the scheduling of Hadoop by providing support to dynamic changes in the availability of resources. The scheduler must collect context information i.e. available resources on the nodes to detect dynamic changes. Slave JobTrackers must communicate periodically with the master TaskTrackers to keep updated information and let the scheduler prepare to the new context. The design is based on two key measures. First, a large percentage of Map Reduce jobs run periodically and roughly have the same characteristics regarding CPU, disk and network requirements. Second, in a Hadoop cluster, the nodes become heterogeneous over time due to failures, when newer nodes replace old ones. If Hadoop does not perform preemption/migration of tasks, the involvement of speculative tasks and context-aware scheduler may contribute to avoid the bottlenecks caused by the resources variability.

IV. PERFORMANCE METRICS

Hadoop performance metrics are used to evaluate scheduling algorithms [22]:

- *Average Completion Time*: sum of completion time of all jobs divided by number of jobs.
- *Dissatisfaction*: amount of satisfaction provided by scheduling algorithm in minimum share requirement of users.
- *Fairness*: how fair the scheduling algorithm is in dividing the resources among users [26].
- *Locality:* proportion of tasks which are running on the same resource as where their stored data are located.
- *Scheduling Time*: total time spent for scheduling all of the incoming jobs. This measures the overhead of each Hadoop scheduler.

V. PERFORMANCE ISSUES

Drawbacks of scheduling algorithm to heterogeneity level to each hadoop factor [22, 47]**:**

- *Jobs Starvation*: If required resources are not allocated to job since long time is known as starvation
- *Sticky Slots:* The scheduler assigns a job to the same resource every time.
- *Resource and Job Mismatch*: In a heterogeneous Hadoop system, resources can acquire different features with respect to their computation or storage units. Moreover, jobs in a heterogeneous workload have different requirements. To reduce the average completion time, it is critical to assign the jobs to resources by examining resource features and job requirements.
- *Scheduling Complexity:* The complexity of scheduling algorithms can result from different features, such as gathering more system parameters and state information, and considering various factors in making scheduling decisions.

VI. OBSERVATIONS

Hadoop schedulers are based on system heterogeneity. When the system is homogeneous in all three factors: cluster, workload and user, the FIFO algorithm is selected. When the job size is small and users are heterogeneous, select Fair Sharing algorithm to improve the fairness. COSHH algorithm is the recommended in heterogeneous Hadoop. Scheduling algorithms emphasis on data locality, sharing, fairness and fault tolerant ability

VII. CONCLUSION

To overcome the issue of Big Data storage and processing the open source framework named Hadoop is developed by Apache can be used. Hadoop gives a source to Big Data processing with its components like Map Reduce and HDFS. To process with the Big Data the default scheduler called FIFO has been used. Different scheduling techniques to enhance the data locality, makespan, efficiency, fairness and performance are discussed.

VIII. REFERENCES

[1] Sandholm T, Lai K "Dynamic proportional share scheduling in hadoop" Proceedings of the 15th Workshop on Job Scheduling Strategies for Parallel Processing, Vol-6253, pp. 110–131, 2010, ISBN: 978-3-642-16504-7.

[2] Chen Y, Ganapathi A, Griffith R, Katz RH "The case for evaluating MapReduce performance using workload suites" Proceedings of the 19th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication

Systems, Washington, pp. 390–399, 2011, ISSN: 1526-7539.

[3] 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.

[4] Wolf J , Balmin A , Rajan D , Hildrum K, Khandekar R, Parekh S, Wu KL, Vernica R "On the optimization of schedules for MapReduce workloads in the presence of shared scans" The VLDB Journal, Vol-21, pp. 589–609, 2012, ISSN: 1066-8888.

[5] Tang Z, Zhou J, Li K, Li R "A MapReduce task scheduling algorithm for deadline constraints" Cluster Comput, , pp. 651–662,2012, ISSN: 1386-7857.

[6] Song G, Yu L, Meng Z, Lin X "A Game Theory Based MapReduce Scheduling Algorithm" Emerging Technologies for Information Systems, Computing, and Management, Vol-236, pp. 287-296, 2013, ISBN: 978-1-4614-7009-0.

[7] Chen H, Shen Y, Chen Q, Guo M "HMHS: Hybrid Multistage Heuristic Scheduling Algorithm for Heterogeneous MapReduce System" ICA3PP, Part I, LNCS 8285, pp. 196–205, 2013, ISSN: 0302- 9743.

[8] Zhao Y, Chen L, Li Y, Liu P, Li X, Zhu C "RAS: A Task Scheduling Algorithm Based on Resource Attribute Selection in a Task Scheduling Framework" IDCS, LNCS 8223, pp. 106–119, 2013, ISSN: 0302-9743.

[9] Yuan Z, Wang J "Research of Scheduling Strategy Based on Fault Tolerance in Hadoop Platform " GRMSE, Part II, CCIS 399, pp. 509–517, 2013, ISSN : 1865- 0929.

[10] Wan C, Wang C, Yuan Y, Wang H "Game-Based Scheduling Algorithm to Achieve Optimize Profit in MapReduce Environment" ICIC, LNCS 7995, pp. 234–240, 2013, ISSN: 0302- 9743.

[11] Xie J, Meng FJ, Wang HL, Pan HF, Cheng JH, Qin X "Research on Scheduling Scheme for Hadoop clusters" International Conference on Computational Science, ICCS, Procedia Computer Science, Vol-18, pp. 2468 – 2471, 2013, ISSN: 1877- 0509.

[12] Wan C, Wang C, Yuan Y, Wang H, Song X "Utility-Driven Share Scheduling Algorithm in Hadoop" ISNN, Part II, LNCS 7952, pp. 560–568, 2013, ISSN: 0302-0947.

[13] Hassan MM, Song B, Hossain MS, Alamri A "Efficient Resource Scheduling for Big Data Processing in Cloud Platform" IDCS, LNCS 8729, pp. 51–63, 2014, ISSN: 0302 9743.

[14] Kalra S, Lamba A "A Review on HADOOP MAPREDUCE-A Job Aware Scheduling Technology" International Journal of Computational Engineering Research (IJCER), Vol-04, Issue 5, May 2014, ISSN: 2250 – 3005.

[15] Saranya N, Yoganandh T "An Efficient Map Reduce Task Scheduling and Micro-Partitioning Mechanism for Optimizing Large Data Analysis" International Journal On Engineering Technology and Sciences – IJETS, Vol-1, Issue 7, November 2014, ISSN: 2349-3968.

[16] Mashayekhy L, Nejad MM, Grosu D, Zhang Q, Shi W "Energy-Aware Scheduling of MapReduce Jobs for Big Data Applications"IEEE Transactions On Parallel And Distributed Systems, Vol.- 25, No. X, pp. 1-14, 2014, ISSN: 1045- 9219.

[17] Mridul M, Khajuria A, Dutta S, Kumar N Prasad MR " Analysis of Bidgdata using Apache Hadoop and Map Reduce" International Journal of Advanced Research in Computer Science and Software Engineering, Vol-4, Issue 5, pp. 555-560, May 2014, ISSN: 2277- 128X.

[18] Sandhu R, Sood SK "Scheduling of big data applications on distributed cloud based on QoS parameters" Cluster Comput, pp.817–828,2014, ISSN: 1386-7857.

[19] Pop F, Dobre C, Cristea V, Bessis N, Xhafa F, Barolli L "Deadline scheduling for aperiodic tasks in inter-Cloud environments: a new approach to resource management" J Supercomput, pp. 1754–1765,2014, ISSN: 0920-8542.

[20] Pakize SR "A Comprehensive View of Hadoop MapReduce Scheduling Algorithms" International Journal of Computer Networks and Communications Security ,Vol-2, Issue-9, pp. 308–317, September 2014, ISSN 2308-9830.

[21] Zhang F, Cao J, Tan W, Khan SU, Li K, Zomaya AY "Evolutionary Scheduling of Dynamic Multitasking Workloads for Big-Data Analytics in Elastic Cloud" IEEE transactions on Emerging Topics in Computing, ,Vol- 2, pp. 338-351,2014, ISSN: 2168-6750.

[22] Rasooli A, Down DG "Guidelines for Selecting Hadoop Schedulers Based on System Heterogeneity" J Grid Computing, Vol- 12 , pp. 499–519,2014, ISSN: 1570-7873.

[23] Rasooli A, Down DG "A classification and optimization based scheduler for heterogeneous Hadoop systems" Future Generation Computer Systems, Vol-36,pp. 1-15, 2014, ISSN: 0167- 739X.

[24] Singh D, Reddy CK "A survey on platforms for big data analytics" Journal of Big Data,pp. 1-20, 2014, ISSN: 2196-1115.

[25] Harshitha R, Rekha GS, Guruprasad HS "A Survey on Scheduling Techniques in Hadoop" IJEDR ,Vol-3, Issue 1,pp. 248-254, 2014, ISSN: 2321-9939

[26] Suresh S, Gopalan NP "An Optimal Task Selection Scheme for Hadoop Scheduling" International Conference on Future Information Engineering IERI Procedia, Vol-10, pp 70-75, 2014, ISSN: 2212-6678.

[27] Chen M, Mao S, Liu Y "Big Data: A Survey" Mobile Netw Appl, Vol-19, pp. 171–209,2014, ISSN: 1383-469X.

[28] 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, 2015, ISSN: 1877-0509.

[29] Nikhil B, Riddhikesh B, Balu P, Mukesh T "A Survey On Scheduling In Hadoop For Bigdata Processing" Multidisciplinary Journal of Research in Engineering and Technology, Volume 2, Issue 3, pp. 497-501,2015, ISSN: 2348 – 6953.

[30] Divya S, Rajesh KR, Nithila RMI, Vinothini M "Big Data Analysis and Its Scheduling Policy – Hadoop"

IOSR Journal of Computer Engineering (IOSR-JCE), Vol-17, Issue 1, pp 36-40, Jan – Feb. 2015, ISSN: 2278-8727.

[31] Tsai CW, Lai CF, Chao HC, Vasilakos AV "Big data analytics: a survey" Journal of Big Data,pp. 1-32, 2015, ISSN: 2196-1115.

[32] Sreedhar C, Kasiviswanath N , Reddy PC " A Survey on Big Data Management and Job Scheduling" International Journal of Computer Applications, Vol-130 ,Issue 13,pp. 41-49, November 2015, ISSN: 0975-8887.

[33] Raj ED, L.D DB "A Two Pass Scheduling Policy based Resource allocation for MapReduce" International Conference on Information and Communication Technologies (ICICT 2014) Procedia Computer Science, Vol-46 ,pp. 627 – 634,2015, ISSN:1877 -0509.

[34] Nita MC, Pop F, Voicu C, Dobre C, Xhafa F "MOMTH: multi-objective scheduling algorithm of many tasks in Hadoop" Cluster Comput, pp. 1011–1024,2015, ISSN: 1386-7857.

[35] Wang J, Li X "Task scheduling for MapReduce in heterogeneous networks" Cluster Comput, pp. 1-14, 2015, ISSN: 1386-7857.

[36] Ayesha S "SLA-aware and Cost-aware Provisioning and Scheduling of Cloud Resources across Multiple Data centres" International Journal of Advanced Research in Computer and Communication Engineering, Vol-4, Issue 5, pp. 238-242, May 2015, ISSN: 2319-5940.

[37] Saranya N, Yoganandh T, "Task Scheduling Algorithm for Map Reduce To Control Load Balancing In Big Data" National Conference on Research Advances in Communication, computation, electrical science and structures, pp. 27-32, 2015, ISSN: 2348-8387.

[38] Thangaselvi R, Ananthbabu S, Aruna R " An efficient Mapreduce scheduling algorithm in hadoop" International Journal of Engineering Research & Science (IJOER),Vol-1, Issue-9,pp. 102-108, December 2015, ISSN 2278- 0181.

[39] Ghazia MR, Gangodkara D "Hadoop, MapReduce and HDFS: A Developers Perspective" International Conference on Intelligent Computing, Communication & Convergence (ICCC-2014) Procedia Computer Science, Vol-48, pp. 45 – 50, 2015, ISSN 1877- 0509.

[40] K KKH, Vignesh R, Long CAI, Sugumaran R "Big Data - Reduced Task Scheduling" International Conference on Systems, Science, Control, Communication, Engineering and Technology, Vol-01, pp. 79-84, 2015, ISBN: 978-81-929866-1-6.

[41] Li X, Song J, Huang B "A scientific workflow management system architecture and its scheduling based on cloud service platform for manufacturing big data. analytics" Int J Adv Manuf Technol, pp. 119-131,2015, ISSN: 0268-3768.

[42] Tang Z, Liu M, Ammar A, Li K, Li K "An optimized MapReduce workflow scheduling algorithm for heterogeneous computing" J Supercomput, pp. 3-23, 2014, ISSN: 0920-8542.

[43] Qin P, Dai B, Huang B, Xu G "Bandwidth-Aware Scheduling With SDN in Hadoop: A New Trend for Big Data" IEEE Systems Journal, pp. 1-8, 2015, ISSN: 1932-8184.

[44] Anuradha IJ " A Brief Introduction on Big Data 5Vs Characteristics and Hadoop Technology" Procedia Computer Science, Vol- 48 , pp. 319 – 324,2015, ISSN: 1877-0509.

[45] Rani PS, Shalini S, Keerthika JR, Shanthini A "Energy Efficient Scheduling of Map Reduce for Evolving Big Data Applications" International Journal of Advanced Research in Computer and Communication Engineering, Vol.-5, Issue 2, pp. 54-58, February 2016, ISSN: 2278-1021.

[46] Mamdapure S, Ginwala M, Papat N "Task Scheduling in Hadoop" Imperial Journal of Interdisciplinary Research (IJIR), Vol.-2, Issue-1 ,pp. 22-25, 2016, ISSN : 2454-1362.

[47] Ling X, Yuan Y, Wang D, Liu J, Yang J, "Joint scheduling of MapReduce jobs with servers: Performance bounds and experiments" J. Parallel Distrib. Comput, Vol- 90–91, pp 52–66, 2016, ISSN: 0743-7315.

[48] https://hadoop.apache.org/

[49] https://hadoop.apache.org/docs/r1.2.1/fair_scheduler.html

[50] https://hadoop.apache.org/docs/r1.2.1/capacity_scheduler.html