Turkish Online Journal of Qualitative Inquiry (TOJQI)
Volume 12, Issue 8, July 2021: 2867-2875

Efficient Job Scheduling and Resource Allocation using Load Rebalancing on Big Data

Anilkumar Ambore[a], Dr. Udaya Rani V[b]

[a] Research Scholar, VTU, Department of Computer Science and Engineering,
REVA Institute of Technology and Management, Bangalore, India.
[b] Associate Professor, Department of Computer Science and Engineering,
REVA Institute of Technology and Management, Bangalore, India.

Abstract

In recent days, managing big data has been one of the key challenges for managing data effectively and efficiently. This data is generally utilized in all online media, web-based business, and web applications. To manage and store huge volumes of data sets the Hadoop Distributed File System is quite possibly the most broadly utilized frameworks. With respect to job scheduling, HDFS is additionally testing as it assumes a critical part in upgrading time in huge information. Even though there are many scheduling algorithms in the existing works because they are not very efficient in working with dynamic Hadoop environment that is Hadoop cluster with dynamically available resources due to various issues. For example, there is no time limit for the tasks allocated for the dynamic resource allocation. To deal with such issues, this paper presents efficient scheduling and dynamic resource allocation using load rebalancing techniques that take into account future asset accessibility when limiting job deadline misses. Existing problems can define a job scheduling problem with an optimized scheduling cycle by minimizing iteration, and then dynamically allocating resources using the proposed Load Rebalancing technique. The tasks differ in the existing algorithms and offer algorithms for experiments to prove time and time complexity and their implementation is performed in an open-source Hadoop environment. Experiments have proven that the proposed job scheduling algorithm reduces the quantity of repetitions and improves time productivity by dynamically allocating resources compared to the deadline-aware scheduling algorithm.

Keywords: Job Scheduling, Load Rebalancing Technique, Big Data, Dynamic Resource Allocation, HDFS.

1. Introduction

In data innovation, big data is a well-known term. Big data [23] can handle many data structures. Most of the conventional systems are failed to manage big data. It can be utilized for many things. In the most recent investigation, enormous datasets are on the ascent. These datasets are complex and troublesome. Information in many fields has expanded impressively. Information comes from sources, for example, media destinations, business transactions, web-based handling, designing, cosmology, social events, clinical science, banking, and a lot more fields.

Hadoop, the open-source usage of the MapReduce programming model, has become a true standard strategy for storing and analyzing scale-level data in practical manner. For instance, the Hadoop Data Warehouse bunch at Facebook has 3000 PCs and hosts a normal of 25,000 MapReduce occupations for each day. [3] However, studies [9] show that the recent utilization of Hadoop in exploration and

organizations actually has a gap in Hadoop. Improved Hadoop Task Efficiency and Cluster Leverage Hadoop there is a strong demand for unsurprising service for Hadoop clients who have severe necessities on when jobs are done. (E.g. deadline)

However, scheduling meetings are difficult in today's Hadoop platform. First of all, because jobs that have multiple asset prerequisites, it is difficult to decide the number assets each job requires avoiding missing deadlines. Second, Hadoop groups frequently share numerous positions, and the planning succession of these assignments can influence finishing times [20], so adequate resource allocation alone may not ensure the completion of the job, While the scheduler available in Hadoop, for example default FIFO scheduler, reasonable scheduler, limit scheduler, RAS scheduler [18] and varieties [7, 19] Optimize an opportunity to complete tasks regardless of deadlines. Is committed to ensuring the deadline of jobs in Hadoop workloads by estimating the time, the tasks are completed and managing the job queue [20] or scheduling [6].

The most recent trend in utilizing Hadoop in hybrid environments twists the issue. To seek cost viability, Hadoop clusters can be controlled by a combination of sustainable power and conventional network [5, 7, 15] or sharing the similar cloud foundation with intuitive remaining task at hand [15, 19] or run temporary assets, for example Amazon Spot occurrences. In these circumstances, the accessible assets in the Hadoop group are very unique because of the variable sustainable power supply, the changing severity of coexistence, or the termination of the resources accordingly. Sudden market the dynamics in Hadoop cluster capabilities pose a challenge to satisfactory scheduling First of all, it is difficult to predict completion times with feasible dynamic resources. The forecasting model has to be effective for different cluster capacities. Second, the implementation of tasks and scheduling tasks is more complex.

In this paper, to control the resource allocation that each task processes, the appropriate scheduling algorithm. In this fair, all the resources are in fair works. Fair Scheduling main aim is to ensure that all users have a decent amount of cluster capabilities over the long run. Users allocate task to work groups utilizing fewer maps and fewer slots.

This task will then introduce a load balancing mechanism that progressively designates assets to tasks dependent on the distributed hashing mechanism in the payload.

We develop and deploy enhanced Fair Scheduler techniques in open-source Hadoop deployments and conduct comprehensive assessments with various Hadoop workloads.

2.Review of Related Studies

This section presents various research works carried out in the Cloud computing and big data on resource scheduling technique. This sections the limitations of existing works carried out so far.

[2, 3, 24] Presented HDFS as a distributed file system (DFS) in which distributed files are put away in a storage system dependent on HDFS node. Here, large information is spitted as blocks and each block size is typically 64MB or 128MB. Duplicates are made, and duplicates of these blocks will be stored on three unique machines. The HDFS has two fundamental components like Name Node and Data Node. The HDFS design follows the master-slave design. The Name Node is the principle parts, it stores metadata and slave node is called Data Node, it stores application information.

MapReduce [4], It is a programming configuration style utilized for intensive information preparing in distributed computing environments. The author examines MapReduce design how to schedule of jobs in MapReduce. MapReduce's processing techniques are based on distributed processing MapReduce introduces a programming model consisting of two significant capacities [5, 22, 25]. First function in which, it is a map that accepts a set of data key-value sets as input and output as another intermediate key-value pair. Second function is the reduce function, it accepts intermediate key-value pairs generated by the map function as input and reduces the set of intermediate key-value pairs to smaller pairs, MapReduce is paralleled on top of HDFS and processes the address data. On HDFS, in common MapReduce jobs, input files are read from HDFS, and each map job should be scheduled on a machine with model-related input information.

The HDFS and MapReduce provides replication features, which is utilized in order to improve performance. MapReduce's architecture comprises one master hub (Job Tracker) and multiple Slave hub (Task Trackers). A Job Tracker deals with various slave nodes, called Task Trackers. The name node address where the master node deals with file system namespace it keeps up the file system structure and metadata for all files and directories in the tree. Information nodes typically have one node for each node, storing HDFS data in the local file system. Whenever there is an instruction from the client or the Name Node Data Nodes store and retrieve blocks and periodically reports back to the Name Node with a block list.[13] Offers two vitality models for interchange between employee work modes. They examined the true power estimation arising from the involvement of workers' AC to decide how the powers used to express the inertia, the residual state, and the closed state to affect this interstate exchange. It is possible that exchanging between power modes requires investment and can mean corrupt operation if the load is over blue. Additionally, the stacking of workers serving the stack can continually change, encouraging short-lived opportunities for most workers.

The work done by [13] considers a solitary assignment running in the cloud domain to be a key unit for power profiling. With this strategy, Chen and her associates saw that using all the power from the two missions was not equal to the total power spent of the individual due to the reservation of expenses. They made a force model for all out-vitality utilization which centurion capacity, calculation, and correspondence assets.

[12, 13, 14, 15] The file system name space and files are managed and controlled by the Name Node that is accessed by customers. File system operations (opening, shutting, renaming of record and catalogues) is performed by Name Node. The information hub is answerable for reserving read and write request from file system clients and also defining block mapping from the information hub.

Here the author [1, 16] discussed about large and complex data set. In conventional data management applications, it is hard to manage process and schedule large volumes of data. Storing, analysing, searching, sharing, and visualizing huge volumes of information. Big data is similar to small data as well. But the main thing is that they are bigger. Big data has 3 V-numbers that represent amount, variety, and speed.

Large information is an organized and unstructured information model in which unstructured information isn't put away in a precise and very much characterized way and organized information is efficient like Wikipedia, Google, Facebook, amazon etc, contain unstructured data formats and ecommerce sites with structured data formats. The dataset is increasing day by day. Traditional

systems present many challenges in dealing with unstructured data. For this reason, handling unstructured data is very important. To handle unstructured data, Apache Hadoop [7, 8, 9, 10] is quite possibly the most well-known open-source structures Hadoop has two parts, one is the Hadoop Distributed file system (HDFS) and the other is MapReduce. [21] Discussed different data placement strategies.

3.Objectives of The Study

- Efficient Fair Scheduling algorithm that allocates all the jobs among various resources in fair manner.
- Improved load Rebalancing mechanism that progressively designates assets to various jobs dependent on the distributed hashing mechanism in the payload.

4. Method

This section describes efficient scheduling and dynamic resource allocation using load balancing techniques that take into account future resource availability when the least deadlines are missed. The techniques presented are as follows:

4.1 Fair Scheduling Technique

This procedure controls the asset distribution of each undertaking of the cycle. In this task, all the assets are in the Fair. Fair Scheduling is created by Facebook [1, 5, 4, 22]. The MapReduce is a primary function of Hadoop. Facebook has built Fair Scheduling to oversee MapReduce groups in Hadoop in all reduction map. Will be handled by scheduling the main purpose of proper scheduling is to make sure all users share their cluster capabilities over a period of time. Users allocate jobs to job groups using fewer maps and less slots. Fair Scheduling supports early release. This scheduling allows the pool processing slot to exceed its capacity. Fair Scheduling is like capacity scheduling with a couple of contrasts. In capacity scheduling, multiple queues are used. Fair Scheduling also uses multiple queue conditions as a pool. When the job is collected from Pools and also part of the resource is received. Let's say another job comes to the pool. For this situation, capacity scheduling is performed on a FIFO schedule, so high-priority processes will have to wait a long time. This issue can be improved in an appropriate scheduling algorithm.

Collect jobs that have been waiting for a long time. Then will proceed in parallel fair scheduling improves productivity in the Hadoop distributed file systems by optimizing resource sharing for all undertakings. Resources are allotted similar to each task for good results by using available resources. It creates a pool of tasks standing on a configurable attribute such as a username known as Pools. Defaults Pools have assets that are shared equally, if there are any pools that are not filled, then other pools are full. Submit bulk increments, Fair Scheduling projects detect them and identify those jobs that are marked as non-performing. This Fair Scheduling algorithm works well on with little and enormous datasets.

Table.1. An Example Improved Fair Sharing Scheduing with Processor.

File size	First chunk	Second chunk	Third chunk
12	4	4	4

11	5	3	3
22	12	5	5
17	8	6	3

4.2. File Allocation mechanism

This mechanism partitioned the large files into different number of number of chunks and then allocates to the centralized chunk server. The files in the cloud can be uploaded, deleted dynamically in the centralized chunks server. This will help in avoiding data loss. Figure 1 describes how the files are split into different chunks and subsequently distributed over number of centralized Chunks server defined as Fs.

Chunks files (Fs= F1, f2 ...Fn)..... (1)

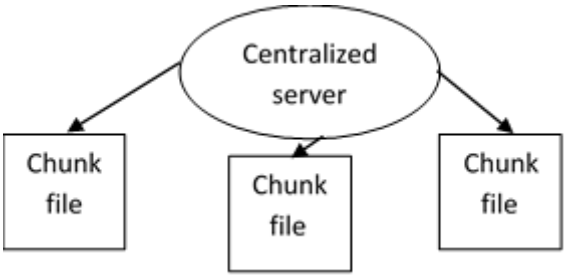


Figure. 1 Splitting of files into different chunks.

4.3. Hash based Distribution (HBD) mechanism

This mechanism is mainly used for storing the chunk files with unique identifier in the cloud where each chunk file have encrypted hash based key generated using advanced encrypting technique. The generated key is stored under HBD table. This HBD mechanism guarantees whenever the file is added to the chunk server automatically put away in the replacement files and in the event that any files deleted or modified then the chunks are moved to next replacement files. In existing distributed file system, the location of the files are integrated in the network of the system, in the proposed system servers nodes are stored along with namespace for easy retrieval using HBD mechanism.

4.4. Load Rebalancing Technique for dynamic allocation of workload

This technique is used to find out if files are under loading, overloaded, and appropriately allocated to all Junks on each sub-server. The file will reflect under if chunks < (1+ΔC) and the file is too large if chunks > < (1+ΔC), described in (2)

$$Nuber\ of\ chuks > (1+\Delta C).L \quad \text{---------- (2)}$$

$$Number\ of\ chunks < (1-\Delta L)\ A) \quad \text{-------- (3)}$$

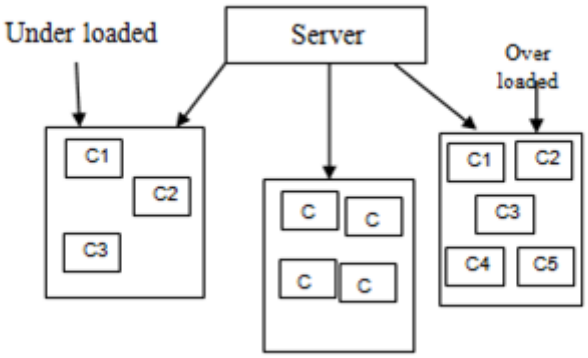
$$Number\ of\ chunks > (1-\Delta U)\ A) \quad \text{---------- (4)}$$

Where ΔL(O ≤ ΔL < 1) and ΔC (0 ≤ ΔC <1)

Entire overloaded files share their load under the loaded node. If node "n‟ is under load, it transfers the load to the "n + 1‟ puzzle and then immediately joins in. As a successor from overloading Request for files loaded under for chunks from overloaded files and sent via physical network links.

Consider case studies, each server with a capacity of 1024 MB is 3 chunks, at that point as per thing 3 and load balancing technique, chunkserver1 is light node overload 384 MB (no. Of chunk < (1-ΔL) A) (2) and Chunk server 3, Is an overloaded file with a load of 640 MB (Number of chunks> (1- ΔU) A) (3). Utilizing this method, the overloaded file moves its heap to the light node, such as the Chunk 3 server, transferring some of the load to the light node. Chunk 1 server.

Figure. 2 Under stacked and over-burden files in the chunk server.



However, load distribution in individual subsystems can affect power consumption due to the non-linear interactions between subsystems

5. Results and Discussion

Cloud environment is utilized for simulating workload allocation issues. To streamline the deployment of genuine machines in the system for testing the adjusting issues happening in the organization, virtual machine parts are utilized. The presentation of the algorithm was computed using simulation. The simulation is applied to the thread.

5.1 Environment Arrangement

Simulation takes place using a simulation toolkit workflow on a cloud environment. In this arrangement, one input and 3 virtual machines must be considered. The input properties and virtual machines shown in Table 2 and Table 3, respectively

Table.2. Datacentre Properties

Cloud Architecture	OS	Virtual machine management	Time Zone	Cost	Cost per Memory	Cost per storage	Efficiency
X86	Linux	Xen	10.0	0.5	0.10	0.2	098

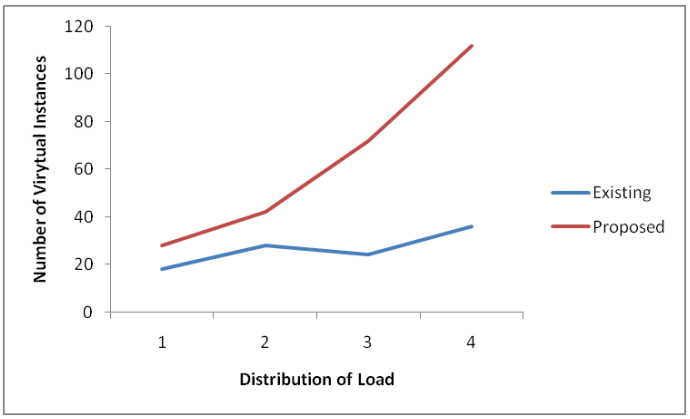
Table.3. Virtual Machine Properties

Image size (MB)	VM Mem (MB) RAM	Millions of instruction per second (MIPS)	Band Width(BW)	PesNo(Number of CPUs)	Virtual Machine Management (VMM)
10000	512	10000	10000	1	Xen

For simulation reason, a dataset log files will be utilized as a load. Each F file has number of files f and f is divided into file chunks. File fragments are circulated around virtual machines. Loads are most uniformly distributed in this virtual machine.

The Log files holds the simulation results performed, Figure 3 shows, Load dissemination between the virtual machines considered for different algorithms, i.e. The Existing load balancing technique and proposed load rebalancing strategy

Figure.3 Load Rebalancing among various nodes vs Number of virtual instances.



In above Figure 3, as contrast with Existing load balancing procedure is contrasted and proposed load rebalancing strategy that disseminate load reliably in some expand. In vm1, vm2and vm3 the load dispersions are almost equivalent in sum. Figure 4 describes time taken for execution of each works assigned over the virtual instances

Figure.4 Load rebalancing among various virtual instances vs. Time Taken for execution

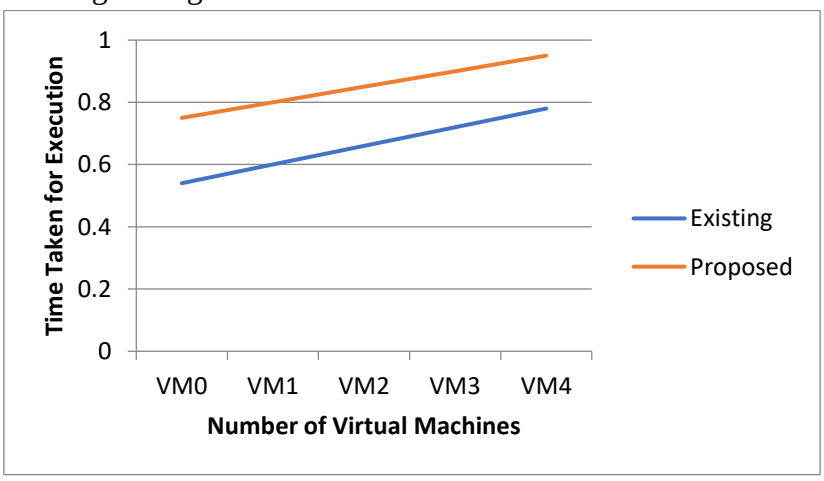
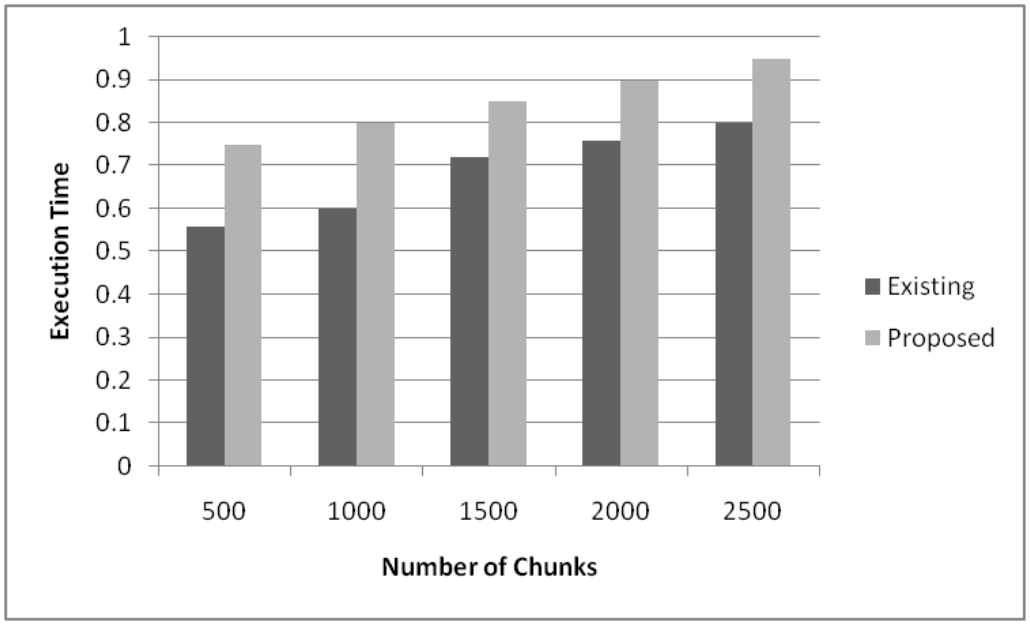


Figure.5 Required Execution time vs. number of chunks.



6.Conclusion

This work offers efficient job scheduling and Dynamic Resource Allocation using Load Rebalancing Technique that takes future asset accessibility when minimizing job deadline misses. This work defines the job scheduling issue with the time cycle of fair scheduling by limiting iteration and then dynamically allocating the resources using proposed Load Rebalancing technique. The results were drawn from considering about number of different jobs in the existing algorithm and proposed algorithm for experiments, demonstrating the time unpredictability and timing and their implementation is performed in Hadoop environment. Open-source Experimental results prove that proposed job scheduling algorithm reduces the quantity of iterations and improves time effectiveness by dynamic resource allocation compared existing dead-line aware scheduling algorithm

References (APA)

[1]. D. Pan & Y. Yang, "A Constant Complexity Fair Scheduler with O (log N) Delay Guarantee", ITC'19, Proceedings of Poster Beijing University of Posts and Telecommunications Press, pp. 2401-2410, 2019

[2]. B. J. Mathiya and V. L. Desai, "Apache Hadoop Yarn Parameter configuration Challenges and Optimization," *2015 International Conference on Soft-Computing and Networks Security (ICSNS)*, Coimbatore, 2015, pp 1-6. Doi: 10.1109/ICSNS.2015.7292373

[3]. M. Usama, M. Liuiu, M. Chen, "Job Schedulers for Big Data Processing in Hadoop Environment: Testing Real-Life Schedulers using Benchmark Programs", vol. 3, No. 4, p.p. 260-273, 2017

[4]. J. V. Gautam, H. B. Prajapati, V. K. Dabhi and S. Chaudhary, "A survey on job scheduling algorithms in Big data processing," *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, Coimbatore, 2015, pp. 1-11.Doi: 10.1109/ICECCT.2015.7226035

[5]. S. Bende, R. Shedge, "Dealing with Small Files Problem in Hadoop Distributed File System", Procedia Computer Science, Vol. 79, pp. 1001-1012, 2016

[6]. C. Sreedhar, N. Kasiviswanath and P. C. Reddy, "A Survey on Big Data Management and Job Scheduling", International Journal of Computer Applications, 130(13), p.p. 41-49. 2015.

[7]. 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 5th European conference on Computer systems (EuroSys '10). ACM, USA, pp. 265-278, 2010, DOI: 10.1145/1755913.1755940

[8]. K. Shvachko, H. Kuang, S. Radia and R. Chansler, "The Hadoop Distributed File System", IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), Incline Village, NV, pp. 1-10, 2010, DOI: 10.1109/MSST.2010.5496972.

[9]. J. Dean, S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", Communications of the ACM, 2008

[10].] S. Shaikh and D. Vora, "YARN versus Map Reduce: A comparative study". 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, pp. 1294-1297, 2016

[11]. D. Borthakur, K. Muthukkaruppan, K. Ranganathan, S. Rash, J.-S. Sarma, N. Spiegelberg, D. Molkov, R. Schmidt, J. Gray, H. Kuang, A. Menon, A. Aiyer, "Apache Hadoop goes Realtime at Facebook", in: SIGMOD '11, Athens, Greece, 2011

[12]. F. Chang, J. Dean, S. Ghemawat, W.-C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fiker, R.E. Gruber, "BigTable: a distributed storage system for structured data", in 7th USENIX Symposium on Operating Systems Design and Implementation, OSDI'06, pp.205–218, 2006

[13]. Q. Chen, D. Zhang, M. Guo, Q. Deng, S. Guo, "SAMR: a self-adaptive MapReduce scheduling algorithm in heterogeneous environment", in: IEEE 10th International Conference on Computer and Information Technology (CIT), pp.2736–2743, 2010, DOI: 10.1109/CIT.2010.458

[14]. J. Dean, S. Ghemawat, "MapReduce: simplified data processing on large clusters", in: OSDI '04, pp.137–150, 2004

[15]. S. Ghemawat, H. Gobioff, S.-T. Leung, " The Google file system", in: Proc. SOSP 2003, pp.29–43

[16]. B. He, W. Fang, Q. Luo, N. Govindaraju, T. Wang, "Mars: a MapReduce framework on graphics processors", in: ACM 2008, pp.260–269

[17]. G. Lee, B.G. Chun, R.H. Katz, "Heterogeneity-aware resource allocation and scheduling in the cloud", in: Proceedings of the 3rd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud, vol.11, 2011

[18]. C. Tian, H. Zhou, Y. He, L. Zha, "A dynamic MapReduce scheduler for heterogeneous workloads", in: Eighth International Conference on Grid and Cooperative Computing, GCC'09, IEEE, 2009. DOI: 10.1109/GCC.2009.19

[19]. M. Zaharia, A. Konwinski, A.D. Joseph, R. Katz, I. Stoica, "Improving MapReduce performance in heterogeneous environments", in: Proc. OSDI, San Diego, CA, pp.29–42, 2008

[20]. J. Shen, D. Liu, J. Shen, Q. Liu, and X. Sun, "A secure cloud-assisted urban data sharing framework for ubiquitous-cities", *Pervasive and Mobile Computing*, vol. 41, pp. 219–230, 2017.

[21]. Ambore and U.R.V, "A Survey on Data Placement Strategy in Big Data Heterogeneous Environments", 3rd International Conference on Trends in Electronics and Informatics (ICOEI), 2019, DOI: 10.1109/ICOEI.2019.8862676

[22]. Scheduling in Hadoop, Available in: https://developer.ibm.com/articles/os-hadoop-scheduling/, accessed on 15, February 2018

[23]. Apache Software Foundation, Hadoop Apache, Accessed on 10th April 2018. URL: https ://hadoop.apache.org/

[24]. D. Hu, D. Feng, Y. Xie, G. Xu, X. Gu and D. Long, "Efficient Provenance Management via Clustering and Hybrid Storage in Big Data Environments," in *IEEE Transactions on Big Data*, vol. 6, no. 4, pp. 792-803, 1 Dec. 2020. Doi: 10.1109/TBDATA.2019.2907116

[25]. M. S. Mahmud, J. Z. Huang, S. Salloum, T. Z. Emara and K. Sadatdiynov, "A survey of data partitioning and sampling methods to support big data analysis," in *Big Data Mining and Analytics*, vol. 3, no. 2, pp. 85-101, June 2020. Doi: 10.26599/BDMA.2019.9020015