A Review on Data locality in Hadoop MapReduce

Anil Sharma
Associate Professor,
School of Computer Applications,
Lovely Professional University,
Phagwara, Punjab, India
anil.19656@lpu.co.in

Gurwinder Singh
Research Scholar,
School of Computer Applications,
Lovely Professional University,
Phagwara, Punjab, India
gurwinder.11@gmail.com

***Abstract*— MapReduce has emerged as a strong model for processing parallel and distributed data for huge datasets. Hadoop an open source implementation of MapReduce has approved MapReduce widely. Hadoop fragments the input file into number of data blocks to allocate them to various DataNodes in cluster. Hadoop must provide effective scheduling to process these data blocks in efficient way. One of the issues that play vital role in efficient processing of MapReduce is Data Locality which is caused due to overhead of network. Data locality is equipped for moving the computation adjacent to the data where it dwells. It is a key resource in distributed environment which influences the tasks accomplishing time. The issues which troubles data locality are cluster and network load, resource sharing, cluster environment, size of data blocks, number of mappers and reducers.This paper aims to review various algorithms that are aware of data locality in scheduling, along with their strengths and weaknesses.**

Keywords*— *MapReduce, Hadoop, HDFS, Scheduling, Data locality

I. INTRODUCTION

A distributed database system is a group of self-governing machines coupled by participating networks that acts as single workstation [1]. To solve a complex problem requires the problem to be partitioned into sub problems where every sub problem is tackled by at least one of the computing node. These computing nodes can talk to one another via sending and receiving messages. The present scenario concerns about Big Data focuses on compute and data intensive tasks. A collection of huge data sets which are non-manageable for conventional tool are termed as Big Data [2]. Organizing Big Data is vital for business and big data analytics in terms of accessing data that arouses the requirement of such applications that can handle distributed processing of huge amount data existing in different type of formats [1][2][3].

MapReduce [4] is one of the main programming model that supports parallel and distributed processing of a huge datasets. It is extremely scalable and efficient platform because it allows use of number of commodity machines for distributed computing and furthermore it provides application programmers a linear execution of logic stated with mappers and reducers. The lower level of parallelization particulars are taken care at runtime. Important characteristics of MapReduce includes scaling, fault tolerant and it can be easily applied to data mining, machine learning and technical simulations. Among Hadoop, Sphere, Mars etc. Hadoop is commonly and extensively used MapReduce platform [5][6][7].

For storage of data Hadoop consists of Hadoop Distributed file system (HDFS) and processing of data is done via MapReduce model in HDFS following master-slave architecture as in Fig.1 [8][9]. The JobTracker is master node and the TaskTrackers are slave nodes. Job is split into number of map and reduce tasks. The input of a job in Hadoop is divided into fragments of same size called input splits. Map tasks results are deposited to local disk and these intermediate results are input for reduce tasks. Further, the output of reduce task is generated and stored in HDFS [8].

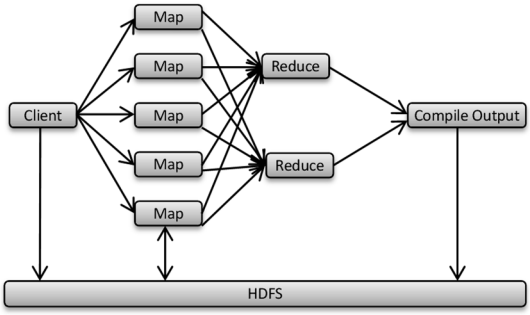

Fig-1 MapReduce Structure [8]

Scheduling the reduce tasks in Hadoop comes up with several problems related to congestion because task scheduling process in Hadoop is based on pulling the data from source node. TaskTracker is responsible for sending a heartbeat message periodically and carting map-reduce tasks by issuing requests. JobTracker is responsible for ensuring each task to run on the designated TaskTracker which holds the required input splits. Thus this is the reason for MapReduce being data local at the time of map tasks scheduling. One of the negatives of Hadoop is that any reduce task which is still not running is scheduled to any of the demanding TaskTracker irrespective of location of TaskTracker in network [10][11][12].

II. DATA LOCALITY IN HADOOP

Data locality is an influential metric of Hadoop which makes considerable impact on system performance. Elements that play vital role in data locality include number of replicas, cluster size, and job execution time and stage. Creating more number of replicas results in improved data locality but it employs more storage space. If size of cluster is huge then it ventures reduced data locality. When job is in initial stage, the input data is on nodes and there are more number of tasks that are not mapped then there is high probability of data locality, consequently at time of job end the probability of data locality is low. Default scheduling algorithm of Hadoop, First in first out (FIFO), supports concept of data locality. The master JobTracker node when receives a heartbeat message from slave node, then initially attempts to look for a map task in the queue for a job whose input data is with that node. If searching of data is successful it results in node level locality and task is launched on particular node, but if node level locality not achieved then JobTracker

efforts for rack locality as shown in Fig-2. Further if again flops then a task is picked randomly which provides rack-off locality. FIFO is in support of data locality but it does have shortcoming of performing scheduling task to task irrespective of its impact on other tasks [6][8].

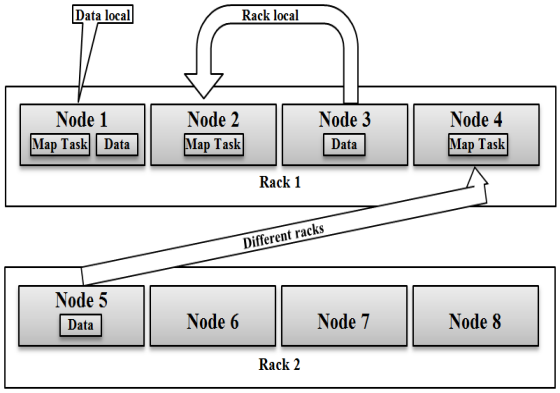

Fig-2 Data Locality in Hadoop [4]

In view of balancing the load, data is disseminated by Hadoop to numerous nodes on the basis of approachability of disk space. The default policy for data locality is concrete and effective for homogeneous environment when nodes are indistinguishable i.e. workload of all nodes is equal which signifies that there is no requirement of shifting data from one to another node. When cluster environment is heterogeneous, local data can be processed fast by a node with high-performance than a node with low performance. Once the high-performance node completes local disks data processing, it is given the data residing in a distant sluggish node. Moving unprocessed data from a sluggish node to a high-performance node increases overhead if volume of data is enormous [9][10].

III. Literature Review

Google [1] has been productively using the MapReduce programming model for diverse purposes and this achievement can be endorsed for numerous causes:

i. Even the programmers who have no exposure to distributed and parallel systems can use MapReduce model as it conceals the subtle elements of parallelization, fault tolerance, locality optimization, and load balancing.

ii. Extensive problems like generating data for Google's Web search service, sorting, machine learning, data mining, and lots of other systems, can be expressed easily as MapReduce computations.

iii. Made it possible for MapReduce to scale to huge clusters consist of enormous number of machines by efficiently utilizing the resources and is appropriate for computational problems come across at Google

Degraded performance of Hadoop due to heterogeneity inspires to propose a data placement strategy [8] for placing data across nodes, to improve data locality, in manner that every node has stable load of data processing. Furthermore this scheme smartly performs load balancing to attain enriched data-processing on the data stored in each node. The performance evaluation on actual data-intensive applications Grep and WordCount reveals that data rebalancing among participating nodes earlier than data-intensive operation consequently improves performance of heterogeneous clusters in MapReduce.

A new task scheduler [10] introduced specifically for scheduling reduce tasks to enhance its data locality to effectively increasing the processing of requesting nodes. The technique confirms run the reduce tasks right TaskTracker so that network bandwidth can be used better. This proposed scheduler can minimize local data shuffling by 11-80 % as shown in experiments.

Dynamic task scheduler [13] used for predicting performance of number of MapReduce jobs and dealing with designation of resources to each job at runtime to accomplish the target without squandering resources. This scheduler foresees the expected finishing time for individual MapReduce job by exploiting fact that job in MapReduce is comprised of several tasks (include mappers and reducers) to accomplish, in addition the number of tasks are deliberated ahead of time amid the phase of job initialization at the time of input split, and observing performance of job at runtime. Considering both individually submitted and yet not finished jobs, the scheduler observes the time taken by previously finished tasks to know average task length for envisaging the job finishing time. This estimation allows the scheduler to adapt required task slots for each job to be allocated at runtime. Experiments conducted to study the influence of data locality and the problems allied with usage of memory in MapReduce phase of Hadoop reveals that reading of data locally is quicker (around 1.5-2 seconds instead of 3 seconds) than reading from distant nodes in HDFS which attributes to distress data transfer on network, and large inputs leads to failure of one slave node which in turn affects the other slave node resulting in degrading performance in terms to run task and serving data to distant TaskTracker.

Prefetching and Pre-shuffling schemes [14], implemented in High Performance MapReduce Engine (HPMR), proved to enhance the performance of shared environment of MapReduce computation, as a solution to problem that Hadoop- On-Demand (HOD) upsurges the use of resources on the cost of performance when number of users are in race for network and hardware resources which significantly deteriorates the performance, in contrast to dedicated environment for MapReduce. Prefetching scheme enhances data locality using intra-block prefetching and inter-block prefetching. Two issues of intra-block prefetching: essential to synchronize the computation and catch suitable prefetching rate, tackled by introducing concept of bi-directional processing bar to synchronize computation along with assessing the efficacy of technique, look for suitable prefetching rate to maximize performance by reducing I/O overhead and eliminating duplicate read operations. Inter-block prefetching, pre-fetch the required replica of block to the local rack by perused it from the node with minimum loaded replica's so as not to limit the effect of performance as whole. Pre-shuffling drops the shuffling's required for intermediary outputs by observing the participating node data or input splits at map phase to envisage which reducer the pairs of key-value are segregated. Data locality is main concern of prefetching scheme while pre-shuffling is responsible for moderating the overhead of shuffling overhead in reduce phase.

In [15] authors designed a parallel task scheduler which can control the allotted capacity of resources to users via regulating their time spent and capable of effective preemption and work conservation. This dynamic priority technique provides effective decisions regarding prioritization of jobs and tool for users for improving and altering the capacity to cope with their job requirements. Due to light-weight design its plus side includes improved data locality, enhanced scalability and overhead of virtualization. Whenever required the users can also scale back the consumption of resources, but the cost of this is too expensive. The major drawback of this approach is not strictly imposing the properties of isolation and no proper mechanism of dealing with long running tasks.

A fair scheduler [16] is designed, to elucidate the clash among data locality and fairness, by proposing a delay scheduling technique, for a cluster of 600 nodes, at Facebook. The jobs are scheduled according to fairness and need to wait slightly to let the other jobs to launch tasks. This algorithm claims to attain almost optimum data locality and fairness for variety of assignments and upturns the throughput. If fairness is imposed strictly it results in loss of locality in two ways: sticky slots and head-of-line scheduling, although 100% locality can be attained by slightly compromising fairness. Whenever the number of jobs is changed, resources are reallocated by killing the executing tasks to entertain new job and wait to let the running task finish for better fair sharing. This delay scheduling method has been executed in Hadoop Fair Scheduler and improves the response times by 5x for light jobs while and doubles the throughput for heavy assignments.

A mathematical model was proposed in [17] to calculate the cost involved in assigning tasks for MapReduce framework of Hadoop. Outcomes of this model include: Optimal assignment of tasks cannot be achieved unless P = NP, discards the supposition that more replicas into system always results in better load balancing, algorithm for overall Hadoop task assignment by offering maximum flow and improved threshold procedures using additional constant that bank on merely number of servers and cost function for data locality.

To improve data locality in homogeneous environment [18] introduces NKS (next-k-node) scheduling strategy for map task by analyzing probability of all map tasks to schedule highest probability task. Low probability tasks can be reserved for nodes that satisfies node locality. The problem with default policy is that it processes some of the map tasks irrespective of node locality by choosing the task which is traversed first, irrespective of whether the nodes having input data for task's may issue requests later on. Experimental results of NKS strategy shows: 77 % network load reduction, 78% decrease in those map tasks that are processed without node locality and enriched performance of MapReduce.

Distributed adaptive data replication algorithm [19] aims to increase data locality with an adaptive replication method having minimum overhead, via automatically creating more replicas for popular datasets and least replications for unpopular datasets. The method promises minimum network traffic, flexibly adapting changing file access methods and put a budgetary limit on additional memory space to be occupied by replicas of data. By using probabilistic sampling along with aging algorithm individually for respective node this technique resolve issues of creating lots of replicas for every file and where to store these replicas. It increases data locality by 7x as compared to FIFO scheduler and reduces turnaround time by 19% and job slowdown 25%.

The algorithm in [20] proposes a method to reduce access latency by foreseeing the forthcoming file usages. In order to increase locality Predictive Hierarchal Fast Spread (PHFS) algorithm pre-replicates data in hierarchal order and attempts to upsurge locality in accesses by calculating user's dynamic adaptability for replicas and with assumption that users working in on the similar environment will demand high probability files. As compared to common fast spread, PHFS results in improved access latency.

An algorithm to provide optimal data locality by scheduling multiple tasks simultaneously by emphasizing merely on locality at node level i.e. placing data and compute on same node is proposed in [21]. Scheduling the tasks one-by-one without caring about its influence of one task on other tasks makes default scheduling methods non-optimal. Therefore Linear Sum Assignment Problem has been integrated with proposed method to consider all the available multiple tasks and idle slots at once. Considering the state when tasks to be scheduled are not more than available idle slots, experiments performed on different parameters: number of tasks, number of nodes, slots per node, idle slots ratio and replication factor that affects the impact of data locality reflects enhancement in goodness of data locality by 14%. Authors mainly focus on determining optimality of default algorithms in Hadoop as compared to Linear Sum Assignment Problem.

To address the overheads allied to data locality in Hadoop [22] proposed a replication scheme for data on the basis mechanism access count prediction to enhance data locality and minimizing data transfer time resulting in reduction of over-all processing time. This technique uses Lagrange's interpolation to envisage the succeeding access count of the data files, consequently to decide about whether to create a new replica or to use dynamically the loaded block of data in place of cache in order to optimize replication factor. The experimental results illustrates that mapping phase completion time shrinks by 9.6%, map tasks with node locality grows by 6.1% , decrease of map tasks by 45.6% in rack locality and 56.5% in rack-off locality.

To interpret problem of data locality in MapReduce from perspective of network [23] introduced a joint scheduler for scheduling and routing with an aim of utilizing the resources and network by pre-processing the routing information of few tasks, and not waiting for some idle machine to demand it. This scheduler can work efficiently with any workload within the determined capacity region. It considerably increases the throughput by more than 30% and minimizes delay for different workloads.

Pause-resume preemption [24] algorithm, which first time considers both Map and Reduce functions to improve execution time and data locality. It safeguards schedulers to choose among kill and wait processes. Furthermore, the preemption increases the overhead but it has improved

map tasks locality by nearly 5%. This algorithm performs better for the circumstances where the share changes continuously because of appearance of fresh jobs.

Based on prefetching the resources, [25] gives a novel scheduling policy to improve data locality by evaluating left over time to finish some task. It pre-fetches the resources for a remote map tasks with some overhead for disk space and network to provide better data locality. The outcomes with experimental setup of 4 computers, 1-JobTracker, 3-TaskTrackers and Hadoop0.20.2 on dataset WordCount, the algorithm illustrates enriched data locality of map tasks by nearly 15%, and a little bit reduction in response time of a job.

To increase overall working of MapReduce along with data locality and task completion time in heterogeneous cluster environments [26] offers techniques to contribute following:

i. Every node implements dynamic data partitioning regardless of various nodes in network.

ii. Particular physical machine is allocated reducer on the basis of virtual machine's readiness and size of partition.

iii. Offers priority to those specific reducers that do not obtain proportional data from particular machine.

By considering speed of every virtual machine the reducers with high workloads are allocated to those with fast processing speed.

A new task scheduling method [27] proposed for MapReduce framework in Hadoop which emphasizes to increase both locality of cache and locality of data along with minimizing the data transfer cost for task operation. The relationship among tasks and resources is given by a selection matrix in addition with bipartite graph according to locality of data. Experiments conducted in environment consisting one NameNode and 3-racks, Ubuntu 14.04.1, and Hadoop 2.3.0 with every rack comprises two DataNodes results in efficient improvement in data locality and cache locality.

A novel dynamic IaaS architecture [28] for Hadoop cluster designed by adding the features: monitoring, scheduling, Virtual machine management and Virtual machine migration as solutions to scheduling resources and data locality. The monitoring unit gathers load of Virtual machines and physical hosts for fabricating and fixing data locality and resource scheduling. Resource scheduling based on load-feedback succeeds in solid scaling for virtual clusters by varying the count of virtual machines. These techniques results in overall better system and efficient load balance.

A multi-objective model [29] proposed with an intend to build up the connection between assignment of resources and scheduling of jobs for streamlining MapReduce task scheduling in the cloud considering minimizing cost and time completion models. It offers enhanced throughput using low latency and reduced runtime of complex jobs in a parallel environment. Furthermore, this model attains a prediction of MapReduce job with high probability on cloud using the multi-objective scheme to confirm noble trade-off conclusions.

Using algorithm [30] execution time variation of Map tasks' in MapReduce framework, as a result of data skew and intensified by interference of virtual machines, can be handled. Randomly dividing tasks among subsets, proposed method result in reducing virtual machine cost and further reduced execution time by 46.7% when contrasted with some past methodologies. Experiments were performed in homogeneous cluster environment with assumption that standard deviation of task and mean are identical for each node. Consequently, heterogeneous groups were not considered because of need of evaluating the normal distribution of assignments for every arrangement independently.

IV. FINDINGS

The Table-1, Comparison of Scheduling Algorithms, illustrates the Key features and drawbacks of scheduling algorithms that are aware of Data locality in Hadoop.

Table 1: Comparison of Scheduling Algorithms

Sr. No.	Algorithm/ Technique/ Method	Key features	Drawbacks
1	Data placement strategy[8]	Considers Heterogeneity of nodes. It offers Enhanced Data Locality and improved Load balancing.	Suffers from data redundancy problem while allocating data in cluster. The mechanism for distribution of data is static.
2	Prefetching and Pre-shuffling schemes [14]	No wastage of system resources. Predicts the straggler. Reduces execution time in shared environment.	Performance reduces in when with complex load.
3	Parallel Task scheduler [15]	Regulates the time spent. Effective pre-emption. Work conservation. Improves Data Locality, scalability, and reduces virtualization overhead	It is expensive. The properties of isolation not strictly implemented. No appropriate tool for long running tasks.
4	Delay Scheduler [16]	Nearly optimum data locality and fairness for variety of assignments. Increases the throughput.	Does not allow resource sharing. It is suitable only for homogeneous environment. Not suitable when more number of tasks are heavy than average size of task.
5	NKS (next-k-node) scheduling strategy [18]	It boosts Data locality. Capably manage and reduces network load.	Only suitable for homogeneous environment.
6	Distributed adaptive data replication algorithm [19]	Increase data locality. Minimum overhead of replicas and network traffic. Reduces turnaround time.	No progress is shown for the output-bound tasks by dynamic replication.
7	Predictive hierarchical fast spread (PHFS) [20]	Reduces access latency. Upturn the data locality. Efficient for such tasks where clients works on a particular framework.	Not suitable when clients requests are random.
8	Linear Sum Assignment Problem lsap-sched [21]	Provide optimal data locality at low cost. Able to Schedule multiple tasks simultaneously.	Performs well when resources at disposal are more.

9	Adaptive Data Replication Scheme [22]	Enhance data locality and minimizes data transfer time. Reduction of over-all processing time, rack locality and rack-off locality. Optimized node locality, replication factor.	Node locality drops as the size of data blocks is increased.
10	Joint Scheduler [23]	Advance routing information of some tasks results in better network load balancing. Improves throughput and delay performance.	Not able to include scheduling at job-level.
11	Pause-resume pre-emption algorithm [24]	Improves execution time and data locality. Allows pre-emption of Map and Reduce functions.	Pre-emption increases the overhead.
12	Job scheduling algorithm based on data locality [25]	Improve data locality. Pre-fetches the resources for a remote map tasks.	Increases the overhead for disk space and network.
13	Load Aware Virtual Machine Mapper [26]	Divides input data dynamically to improve data locality, total job completion time and the Reduce-phase completion time.	Cannot decide on count of reducers to be used in MapReduce phase which results in more cost.
14	Map task scheduling method [27]	Progresses both data and cache locality. Minimizes the data transfer cost for task operation.	No consideration given to clusters current load which results in load imbalance.
15	Multi-objective algorithm [29]	Improve the workflow by minimizing cost and time. Efficient resource usage. Offer better throughput with minimum delay.	Increased number of tasks results in longer finishing time for job.
16	Locality and Interference aware scheduler [30]	Enhances data locality. Tackles the time variation when executing map tasks' in MapReduce phase. Execution time improves and cost of virtual machines reduced.	Suitable only for homogeneous clusters.
17	Data locality based scheduler [31]	Nodes allocated data blocks on basis of processing capacity. It is suitable for heterogeneous environment. Minimizes average job execution time and improves data locality	Suitable for only small clusters.

V. DISCUSSION

Various scheduling algorithms of Hadoop that attempts to resolve data locality issue has been reviewed in this paper. In order to improve one aspect some algorithms lacks in other aspects. For example scheduler in [13] consumes more energy and lacks in data placement. HPMR uses Prefetching and Pre-shuffling for improving performance of shared environment of MapReduce computation but it fails in reducing map tasks existence. Kevin Lai et al. [15] propose a technique which enables users with a mechanism of allocation of resources amending their spending, but it does not pay attention to deadline requirements. In [16] all choices are made on basis of snapshots at some point in cluster, but lacks in finishing number of tasks at time. Technique mentioned in [18] is only for homogeneous environment. DARE scheduler in [19] improves data locality but it lacks in a way that it is costly to generate replicas in huge cluster, more number of replicas means more space which again increases cost, increased overhead of network. W. Wang and L. Ying in [23] considers data locality from perspective of network first time, however they paid no attention to delay in performance.

VI. CONCLUSION AND FUTURE SCOPE

This paper, device a review on data locality in MapReduce to find out few factors that troubles data locality and harms overall performance. A couple of issues that inconveniences data locality are mechanism for distribution of data, cluster and network load, complex load, cost, resource sharing, cluster environment (homogeneous or heterogeneous), unplanned clients requests, size of data blocks, number of mappers and reducers. In future there is need to develop/improve techniques to efficiently handle the heavy load for heterogeneous clusters.

REFERENCES

[1] F. Chang *et al.*, "Bigtable : A Distributed Storage System for Structured Data," *ACM Trans. Comput. Syst.*, vol. 26, no. 2, pp. 1–14, 2008.

[2] A. El Abbadi, "Big Data and Cloud Computing : Current State and Future Opportunities," in *EDBT/ICDT '11 Proceedings of the 14th International Conference on Extending Database Technology*, 2011, pp. 530–533.

[3] D. Agrawal, S. Das, and A. El Abbadi, "Big Data and Cloud Computing : New Wine or just New Bottles ?," in *36th International Conference on Very Large Data Bases, September 13-17, 2010, Singapore*, 2010, pp. 1647–1648.

[4] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Commun. ACM*, vol. 51, no. 1, p. 107, 2008.

[5] Q. Chen, D. Zhang, M. Guo, Q. Deng, and S. Guo, "SAMR : A Self-adaptive MapReduce Scheduling Algorithm In Heterogeneous Environment," in *2010 10th IEEE International Conference on Computer and Information Technology (CIT 2010)*, 2010, pp. 2736–2743.

[6] M. Khan, Y. Liu, and M. Li, "Data Locality in Hadoop Cluster Systems," in *11th International Conference on Fuzzy Systems and Knowledge Discovery Data*, 2014, pp. 720–724.

[7] V. Kalavri and V. Vlassov, "MapReduce: Limitations, optimizations and open issues," in *Proceedings - 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2013*, 2013, pp. 1031–1038.

[8] J. Xie, S. Yin, X. Ruan, Z. Ding, and Y. Tian, "Improving MapReduce Performance through Data Placement in Heterogeneous Hadoop Clusters," in *2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, 2010, p. 9.

[9] Z. Guo, G. Fox, M. Zhou, and Y. Ruan, "Improving Resource Utilization in MapReduce," in *IEEE International Conference on Cluster Computing*, 2012, pp. 402–410.

[10] J. Geetha, N. Udaybhaskar, and P. Chennareddy, "Data-local Reduce Task Scheduling," in *Procedia Computer Science*, 2016, vol. 85, no. CMS 2016, pp. 598–605.

[11] M. R. Ghazi and D. Gangodkar, "Hadoop , MapReduce and HDFS : A Developers Perspective," *Procedia Comput. Sci.*, vol. 48, pp. 45–50, 2015.

[12] W. Wang, K. Zhu, L. Ying, J. Tan, and L. Zhang, "MapTask Scheduling in MapReduce with Data Locality: Throughput and Heavy-Traffic Optimality," *IEEE/ACM Trans. Netw.*, vol. 24, no. 1, pp. 190–203, 2016.

[13] J. Polo *et al.*, "Adaptive Task Scheduling for MultiJob MapReduce Environments," 2009.
[14] S. Seo, I. Jang, K. Woo, I. Kim, J. Kim, and S. Maeng, "HPMR : Prefetching and Pre-shuffling in Shared MapReduce Computation Environment," in *IEEE International Conference on Cluster Computing and Workshops, 2009. CLUSTER '09.*, 2009, p. 8.
[15] T. Sandholm and K. Lai, "Dynamic proportional share scheduling in Hadoop," in *LJSSPP'10 Proceedings of the 15th international conference on Job scheduling strategies for parallel processing Atlanta, GA*, 2010, pp. 110–131.
[16] M. Zaharia and S. Shenker, "Delay Scheduling : A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling," in *EuroSys'10 Proceedings of the 5th European conference on Computer systems*, 2010, pp. 265–278.
[17] M. J. Fischer, X. Su, and Y. Yin, "Assigning Tasks for Efficiency in Hadoop," in *SPAA '10 Proceedings of the twenty-second annual ACM symposium on Parallelism in algorithms and architectures*, 2010, pp. 30–39.
[18] X. Zhang *et al.*, "Improving Data Locality of MapReduce by Scheduling in Homogeneous Computing Environments," in *Ninth IEEE International Symposium on Parallel and Distributed Processing with Applications*, 2011, vol. 2, pp. 1–7.
[19] C. L. Abad, Y. Lu, and R. H. Campbell, "DARE : Adaptive Data Replication for Efficient Cluster Scheduling," in *2011 IEEE International Conference on Cluster Computing DARE:*, 2011, pp. 159–168.
[20] L. Mohammad, A. Isazadeh, and T. N. Shishavan, "PHFS : A dynamic replication method , to decrease access latency in the multi-tier data grid," *Futur. Gener. Comput. Syst.*, vol. 27, no. 3, pp. 233–244, 2011.
[21] Z. Guo, G. Fox, and M. Zhou, "Investigation of data locality and fairness in MapReduce," in *Proceedings of third international workshop on MapReduce and its Applications Date*, 2012, pp. 25–32.
[22] J. Lee, J. Lim, and D. Jung, "Adaptive Data Replication Scheme Based on Access Count Prediction in Hadoop," in *WORLDCOMP'13 - The 2013 World Congress in Computer Science, Computer Engineering, and Applied Computing*, 2013, pp. 1–7.
[23] W. Wang and L. Ying, "Data Locality in MapReduce : A Network Perspective," in *Fifty-second Annual Allerton Conference Allerton House, UIUC, Illinois, USA*, 2014, pp. 1110–1117.
[24] T. A. Phuong, "Reliable and Locality Driven Scheduling in Hadoop," 2014.
[25] J. Bo, W. Jiaying, S. Xiuyu, and H. Ruhuan, "Hadoop Scheduling Base On Data Locality," arXiv: 1506.00425 [cs.DC], 2015.
[26] C. H. Hsu, K. D. Slagter, and Y. C. Chung, "Locality and loading aware virtual machine mapping techniques for optimizing communications in MapReduce applications," *Futur Gener. Comput. Syst.*, vol. 53, pp. 43–54, 2015.
[27] C. Li and Y. Zhao, "An improved task scheduling algorithm based on cache locality and data locality in Hadoop," in *17th International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2016, pp. 244–249.
[28] D. Tao, Z. Lin, and B. Wang, "Load Feedback-Based Resource Scheduling and Dynamic Migration-Based Data Locality for Virtual Hadoop Clusters in OpenStack-Based Clouds," *Tsinghua Sci. Technol.*, vol. 22, no. 2, April 2017, pp.149–159, 2017.
[29] I. A. T. Hashem, N. B. Anuar, M. Marjani, A. Gani, A. K. Sangaiah, and A. K. Sakariyah, "Multi-objective scheduling of MapReduce jobs in big data processing," *Multimed. Tools Appl.*, vol. 77, no. 8, pp. 9979–9994, 2017.
[30] S. M. Nabavinejad and M. Goudarzi, "Data Locality and VM Interference Aware Mitigation of Data Skew in Hadoop Leveraging Modern Portfolio Theory," in *SAC '18 Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018, pp. 175–182.
[31] N. Srinivas, A. Negi, T. B. B. R, and R. Anitha, "A data locality based scheduler to enhance MapReduce performance in heterogeneous environments," *Futur. Gener. Comput. Syst.*, vol. 90, pp. 423–434, 2018.