A Review on Methods to Improve Map Reduce Performance

Ayesha Saad, M. Akheela Khanum

Abstract — The "Big Data" term refers to a collection of large datasets that may not be processed using traditional database management tools. The Map Reduce programming paradigm used in the context of Big Data, is one of the popular approaches that abstract the characterstics of parallel and distributed computing which comes off as a solution to Big Data. Improving performance of Map Reduce is a major concern as it affects the energy efficiency. Improving the energy efficiency of Map Reduce will have significant impact on energy savings for data centers. There are many parameters that influence the performance of Map Reduce. We conducted a systematic literature review to assess research contributions to Map Reduce. Based on the survey conducted it was observed that various parameters like scheduling, resource allocation and data flow have a significant impact on Map Reduce performance.

Index Terms— Big Data, Data flow, Energy Efficiency, Map Reduce, Performance, Resource Allocation, Scheduling

1 INTRODUCTION

Various companies (ranging in size and computing maturity) are adopting Cloud computing technology to perform their business processes, mainly driven by the fact that it reduces the cost of computing infrastructure deployment and management. It is also useful to note that the business case for migrating to Cloud computing systems has often centered on the cost savings that would arise due to reduced use of energy at a client site.

There is also currently significant interest in performing various types of analysis over ''big-data'' with Cloud-based infrastructures using Hadoop [1], [2]. There are three main things in Hadoop development Client machine, Master, Slaves. The Master nodes supervise two important aspects of Hadoop: storing large amount of data (HDFS), and running parallel calculations on all that data (Map Reduce). The Name Node manages and organizes data storage capacity (HDFS), while the Job Tracker administers and arranges the parallel processing of information utilizing Map Reduce. Slave means both a Data Node and Task Tracker which is used to communicate with and accept the command from their master nodes. The Task Tracker work under the Data node and job tracker works under the Name Node. As the writing procedure, Hadoop partitions the data into blocks with a predefined block estimate. The blocks are then composed and copied in the HDFS. The blocks can be copied various times in view of a particular esteem which is set to 3 times by default. The MapReduce function is distributed file system. Basically a large file is distributed into block of identical size and they are split across the cluster for storage. In MapReduce implementation there are three stages: Map, Shuffle, and Reduce . The map stage concern as map function to all input, it is utilized to handle the blocks in the input file that are kept into the PCs nearby capacity. All mappers do their work in parallel and they can work in parallel and separately to each other. In cluster if one computer fails then result can be recomputed on another computer.

A mapper procedures the substance of a piece line by line, translating every line as a key-value match. The real map function is called separately for each of these sets and makes a self-assertively expansive file of new key-value sets from it:

Map (key, value) -> List (key', value')

After Map function finish its process it will pass the result to Shuffle function to arrange the resulting pair with their keys then pass it to Reducer as per their keys. The structure ensures all sets with a similar key are appointed to a similar reducer. Now all pair of key is gathered by reducer gather and creates a sorted list from the values. Input for the reduce function is key and the sorted list of values. To make a size of list very small, reduce function compact the list of values it returns a single value as its output. Reduce function creates a list of key-value pairs, just like the map function:

Reduce (key, List (values)) -> List (key', value')

Hadoop is a framework for data intensive (analysis) applications on large computing clusters by the use of the Map/Reduce paradigm [3]. It has become very popular within social media data analysis projects in order to tackle the scalability of analysis required across large data volumes that could not be performed with traditional paradigms or technologies. Furthermore, MapReduce has also become a useful benchmarking tool [4] due to its high storage, computing power and network requirements for comparing the performance of various computing architectures.

Understanding how Map Reduce could be efficiently executed across a Cloud environment remains an important challenge. The objective of this work is to address this challenge, by determining how data intensive computation could be carried out over a Cloud computing environment and what significant changes in performance are achieved.

A systematic survey of various literatures has been conducted

- *Ayesha Saad is currently pursuing masters degree program in Computer Science in Integral University, India, PH-918009954776. E-mail: ayesha-saad75@gmail.com*
- *M.Akheela Khanum is an Associate Professor currently at Department Of Computer Science and Engineering, Integral University, India, PH-919651722460. E-mail: akheela@iul.ac.in*

and what issues in Map Reduce they cover and techniques used by various authors to deal with these issues have been orderly listed. Our proposed model has been specified along with Hadoop Map Reduce workflow. Conclusion is subsequently outlined.

2 Related Work

Eugen Feller, Lavanya Ramakrishnan, Christine Morin[5] in their work the authors evaluated Hadoop performance in the traditional model of collocated data and compute services and also impact of separating the services. They found that separation provides more flexibility in virtualized environment. They also conducted energy efficiency evaluation of Hadoop and along with many conclusions founded that progress of completion associates with power consumption and power consumption is heavily application specific.

S Ibrahim, T Phan, A Carpen-Amarie, H Chihoub, D Moise, G Antoniu[6] in their paper, the authors focused on Map Reduce processing and investigated the impact of scaling dynamically the frequency of compute nodes. They conducted a series of experiments to find out the implication of Dynamic Voltage and Frequency Scaling (DVFS) settings on power consumption in Hadoop Clusters. They enabled various DVFS governers ie: performance, ondemand, conservative and userspace and observed noteworthy changes in performance and power consumption across different application.

Mukhtaj Khan, Yong Jin, Maozhen Li, Yang Xiang and Changjun Jiang[7] present a Hadoop job performance model that accurately estimates job completion time and further makes out the required amount of resources for a job to be completed within a deadline. The proposed model employs a novel technique to estimate the execution time of a job. And a technique for resource provisioning to satisfy jobs with deadline requirements.

Javier Conejero, Omer Rana, Peter Burnap, Jeffrey Morgan, Blanca Caminero, Carmen Carrión[8] in their paper have measured energy consumption of a number of virtual machines running the Hadoop system. The objective of this work has been to measure and characterize power consumption for high throughput workloads (using Hadoop). They concluded that there is a non-linear relationship between the number of virtual machine, the workloads that these VMs execute and the power consumption. They found that deploying 8 or more VMs on the same physical machine accounts for the maximum power consumption possible for a particular cloud infrastructure.

Jacob Leverich, Christos Kozyrakis[9] in their paper presented a work on modifying Hadoop to allow downsize of operational clusters. Their findings concluded that running Hadoop clusters in fractional configurations can save between 9% and 50% of energy consumption.

Rini T. Kaushik and Milind Bhandarkar[10] in their work designed a variant of HDFS called as Green HDFS. It allows for scale-down by classifying subset of servers that are not accessed in past n days. This subset forms a Cold Zone, the probability of it being accessed again is lower. This will ensure period of idleness in cold zone and allow a large number of servers in cold zone to transition to inactive power modes. Thus ensuring significant reduction in energy cost of cluster. Real life HDFS traces from a Hadoop Cluster at Yahoo shows a 26% energy consumption reduction by doing only cold zone power management. The savings of $2.4 million annually was observed when GreenHDFS technique was applied across all Hadoop clusters (amounting to 38000 servers) at Yahoo.

Yanpei Chen, Archana Ganapathi, Randy H. Katz[11] proposed an analogy of whether to compress or not to compress. They developed a decision algorithm that helps Map Reduce users to compression. They concluded that using compression gives savings of energy upto 60%.

Xiaoli Wang, Yuping Wang, Yue Cui[12] proposed a novel energy aware multi-job scheduling model based on map reduce. Firstly, the authors noticed the changes in energy consumption with performance. Then the network bandwidth issue was considered and finally the task scheduling strategies are considered by formulating the problem as integer bi-level programming model.

M. Malik, H. Humayoun[13], In their work, through methodical investigation of power, performance measurements and comprehensive system level analysis, authors demonstrate that low power embedded architectures can provide significant energy-efficiency for processing big data analytics applications.

L Mashayekhy, M M Nejad, D Grosu, Q Zhang, W Shi[14], in their paper the authors proposed a framework for improving the energy efficiency of MapReduce applications, while fulfilling the service level agreement (SLA). The authors firstly modeled the energy-aware scheduling problem of a single MapReduce job as an Integer Program, and developed two novel algorithms, that find the allocation of map and reduce tasks to the machine slots in order to minimize the energy consumed while application execution. The authors performed various experiments and found that with their proposed system the scheduling consumed approximately 40 percent less energy on average than the schedules obtained by a common scheduler.

Z Tang, L Jiang, J Zhou, K Li, K Li[15], the authors focused on the point that when there is a large output from map task, the performance of MapReduce gets influenced significantly.They drew attention that waiting time for reduce tasks increases due to the system slot resources waste, they finally proposed an optimal reduce scheduling policy for reduce tasks' start times which decides the start time point of each reduce task according to each job context dynamically, including the task completion time and the size of map output. The experiment results illustrated that the reduce completion time is decreased sharply. It is also proved that the average response time is decreased by 11% to 29% with this algorithm.

W Yu, Y Wang, X Que, C Xu[16] in their work the authors focused on the shuffling phase that involves globally exchanging the intermediate data generated by the mapping phase. They proposed a new scheme of virtual shuffling to enable easy data movement and reduce I/O for MapReduce shuffling, thus contributing towards energy efficiency. Their experimental results showed that virtual shuffling significantly speeds up data movement in MapReduce and achieves faster

job execution and accounts for about 12% savings in power consumption.

Kumar et al. [17] have proposed a solution for heterogeneous environments by using a context-aware scheduler. The author designed the system using two key basics firstly, a large percentage of MapReduce jobs are run periodically and roughly have the same characteristics regarding CPU, network, and disk requirements. Second, the nodes in a Hadoop cluster become heterogeneous over time due to failures, when newer nodes replace old ones. Thus, the system optimizes the jobs using same datasets.

Chen et al.[18] also came up with an approach for heterogeneous environments. They developed a novel algorithm for scheduling which aims to improve MapReduce by saving execution time and system resources. In MapReduce, slow tasks prolong the execution time of an entire job. In heterogeneous clusters, nodes take different times to complete the same tasks due to their differences in various parameters. The scheduler uses historical information of each cluster node to tune parameters and discover slow tasks. Thus the scheduler classifies slow nodes and launch backup tasks.

Tian et al.[19] proposed another dynamic scheduler whose main goal was to improve the hardware utilization rate when different MapReduce workloads run on the cluster. The proposed scheduler distributes workloads into three queues: CPU-bound, I/O-bound, and Wait-queue. The authors came up with an analytical model to compute and classify the workloads at runtime. Firstly, the jobs that arrive are put into the waiting queue. Next, the scheduler assigns one Map task to every TaskTracker for predicting the job type. This architecture balances the usage of both CPU and disk I/O, improving Hadoop throughput by about 30% under heterogeneous workloads.

He et al.[20] propose a new scheduler with the premise that local Map tasks are always preferred over non-local Map tasks, no matter which job a task belongs to. A marker is used to categorize nodes and to ensure each node has a fair chance to grab its local tasks. To accomplish this, the scheduler relaxes the job order for task scheduling. Doing so, it improves performance by avoiding data transfer in Map tasks, which may degrade job execution performance.

Zhang et al.[21] in his work also deals with the data locality of reduce task. The authors propose a two-phase execution engine of Reduce tasks to deal with access delays that may degrade system performance. The degradation is related to massive remote I/O operations while copying intermediate results. In the designed system's first phase, the engine first selects the nodes to run Reduce tasks and informs the selected nodes to fetch intermediate results for Reduce tasks. In the second phase, only the selected nodes allocate computing and memory resources to execute the Reduce tasks.

Zhang et al.[22] propose a scheduling method called next-k-node scheduling (NKS) that helps in enhancing the data locality of map tasks. Firstly, the system calculates probability of each map task and according to where their data is stored, generating low probability for tasks that have its input data stored on the next node. And the schedules the task with highest probability, and reserving task with lesser probability to nodes holding their input data, this improves data locality.

Seo et al.[23] present a scheme that can improve the overall performance in shared MapReduce by using concepts of prefetching and preshuffling. The prefetching scheme that exploits data locality, is divided in two steps. First, prefetching of data present within a single block this is called intra-block prefetching. Second, prefetching of an entire block replica to a local rack is done this is called inter-block prefetching. In the pre-shuffling scheme that is designed to reduce the network overhead required to shuffle key-value pairs, the task scheduler observes the input splits of the Map phase and then predicts how to partition the key-value pairs while keeping in mind the Reducer locations. The expected data are assigned to a Map task near the future Reducer before the execution of the Mapper.

Nykiel et al.[24] proposed a module named MRShare. He proposed system transforms a batch of queries into a new batch, by merging jobs into groups and evaluating each group as a single query. Merging multiple jobs allows the entire batch of jobs to be analyzed and this maximizes the degree of resource sharing while minimizing the consumption of resources.

Yu, Yandong Wang and Xinyu Que[25] in their work proposed an implementation called Hadoop-A, an acceleration framework that allows plug-in components to deal with performance issues in various ways. The authors firstly developed a new merge algorithm that avoids various rounds of disk accesses to merge same intermediate data segments that comes from the map tasks. Secondly, it delays the reduce tasks till all the intermediate data have been merged. The serialization barrier that occurs due to delayed reduce phase is broken by a redesigned pipeline of shuffle, merge, and reduce phase for reduce tasks. In this particular pipeline the map tasks map the data splits as soon as possible. Finally, the authors proposed an algorithm that merges data without the use of disks. This framework speeds up data movement in map reduce and doubles throughput of Hadoop and reduces CPU utilization.

Vernica et al.[26] also describes solutions to improve Hadoop's performance by focusing on the interaction of Mappers, introducing an asynchronous communication channel between Mappers. Using a meta-data store (DMDS), Mappers can post metadata about their state and can also update themselves about the state of all other Mappers. This makes mappers more aware othier environment and results in flexibility and adaptivity of Hadoop.

Ahmad et al.[27] proposed MaRCO (MapReduce with communication overlap), which is directed to the overlapping of the Shuffle with the Reduce computation. The proposed system takes partial data from map tasks and breaks Reduce task into smaller invocations, and applies a final reducing step that re-reduces all partial reduce output to produce final output.

Lin et al.[28] proposed an overlapping model between map and shuffle phases. This approach is based on two complementary scheduling algorithms called MaxSRPT and SplitSRPT. While the former, minimizes the average response time of the queue, latter addresses the poor performance of MaxSRPT when jobs are more unbalanced.

Ho et al.[29] and Ibrahim et al.[31] focused on improving performance by changing the data flow between Mappers and Reducers. As Hadoop uses a many-to-many communication

model between Mappers and Reducers. This results in saturation of network bandwidth during the shuffle phase.

Ho et al.[30] dealt with this problem by modeling the traffic in a multiple-racks environment. The authors developed a greedy algorithm to find the optimal solution for the problem.

Ibrahim et al. address the problem by efficiently partitioning the intermediate keys to decrease the amount of shuffled data. This guarantees fair distribution of the Reducers' inputs, improving the overall performance.

Jiang et al.[31] propose known alternative methods as solutions to tuning MapReduce performance. They enhanced the way a reader retrieves data from the storage system with a direct I/O support, which outperforms streaming I/O by 10%. They implemented an indexing scheme for sorted files, improving the Hadoop performance. Finally, the authors also proposed an optimization to the HDFS to deal with small files; The approach allows DataNodes to save some metadata of small files in their memory, improving performance when dealing with small files.

Hammoud et al.[32] proposed an approach named Center-of-Gravity Reduce Scheduler (CoGRS). The work designs a Reduce task scheduler that is locality aware for saving MapReduce network traffic. The scheduler aim towards decreased network traffic, which allows for more Map jobs to co-exist in the same system.

Park et al.[33] proposed a novel approach to virtualization. A dynamic virtual machine reconfiguration technique on virtualized cloud environments running Hadoop was developed by authors. This techniques improves the overall job throughput by improving the input data locality of a virtual MapReduce cluster. DRR temporarily increases the number of cores to virtual machines to run local tasks. Scheduling of tasks is also done based on data locality, the system also adjusts the compuatational capability of virtual nodes.

TABLE 1
COMPARATIVE ANALYSIS

Parameters That Affect Performance In Map Reduce	**Issues Dealt**	**Techniques Used**
Scheduling	Heterogenity	Multi Queue Schedulers
	Data Locality	Data Locality Aware Scheduler
	Inter Job Parallelism	Reusing Map Reduce Job among common Data sets
Data Flow	Throughput Increased/ Reduce CPU Utilization	Authors propose a full pipeline to overlap shuffle, reduce and merge phases.
	Minimize Average Response time of Queue	Author proposed an overlapping model between map and shuffle phase.
	Minimize the saturation of Network Bandwidth due to Hadoop all- all communication model	Authors alteredthe Hadoop Map/Reduce data flow(Modeled the traffic in multiple rack environment)
Resource Allocation	Minimize Network Traffic	Reduce Task Scheduler
	Overall Job Throughput Increased due to improved data locality	Dynamic Virtual Reconfiguration Technique

3 MAP/REDUCE WORKFLOW IN NATIVE HADOOP

MapReduce workflow in native Hadoop works as follows:

Step 1: Client " A" sends a request to NameNode. The request includes the need to copy the data files to DataNodes.

Step 2: NameNode replays with the IP address of DataNodes.

Step 3: Client " A" accesses the raw data for manipulation in Hadoop.

Step 4: Client "A" formats the raw data into HDFS format and divides blocks based on the data size. In the above example the blocks B1to B4 are distributed among the DataNodes.

Step 5: Client "A" sends the three copies of each data block to different DataNodes.

Step 6: In this step, client "A" sends a MapReduce job (job1) to the JobTracker daemon with the source data file name(s).

Step 7: JobTracker sends the tasks to all TaskTrackers holding the blocks of the data.

Step 8: Each TaskTracker executes a specific task on each block and sends the results back to the JobTracker.

Step 9: JobTracker sends the final result to Client "A". If client "A" has another job that requires the same datasets it repeats the set 6-8.

Step10: In native Hadoop client "B" with a new MapReduce job (job2) will go through step 1-5 even if the datasets are already available in HDFS. However, if client "B" knows that the data exists in HDFS, it will send job2 directly to JobTracker.

Step 11: JobTracker sends job2 to all TaskTrackers.
Step12: TaskTrackers execute the tasks and send the results back to the JobTracker.
Step 13: JobTracker sends the final result to Client "B".

4 Proposed Work

MapReduce workflow in our proposed system has been explained as:
Step 1 to Step 8: remain in the same workflow as native Hadoop. Except results from the first 7 steps are stored in the CJBT.
Step 9: JobTracker sends the result to Client "A". In this step, NameNode keeps the names of the blocks that produced the results in the local lookup table (CJBT) by the Common Job Name (Job1) that has common feature as explained above.
Step 10: Client "B" sends a new MapReduce job "Job2" to the JobTracker with the same common job name and same common feature or super-sequence of "Job1".
Step 11: JobTracker sends "job2" to TaskTrackers who hold the blocks, which have the first result of the MapReduce "Job1". In this step, the JobTracker starts with checking the CJBT first to find if it is a new job which has the same common name and common features of any previous ones or not – In this case yes. Then the JobTracker sends "Job2" only to those task trackers that hold the results of "job1" executed previously. We may assume here that the lookup table will be updated with more details OR just remain as is because every time we have a new job that may carry the same name of "Job1".
Step 12: TaskTrackers execute the tasks and send the results back to the JobTracker.
Step 13: JobTracker sends the final result to Client "B".

5 Conclusion

In this paper, a systematic literature review has been conducted regarding Map/Reduce performance. The study showed that major areas that influence the performance of Hadoop Map/Reduce are Scheduling, Data flow, Resource Allocation.
Our paper distinctively signifies the issues that the various literature surveys conducted so far have covered and techniques used to achieve an improvement in performance. With improved performance, energy efficiency is achieved which our proposed model aims to do. Our proposed system uses a Common Job Table that saves the execution time by retrieving the results of job having common features and name and not re-executing them. Thus this system focuses on the Data flow aspect of Map/Reduce.

References

[1] C. Lam, Hadoop in Action, Manning Publications, 2010.
[2] T. White, Hadoop: The Definitive Guide, O'Reilly, 2009.K.
[3] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, Commun. ACM 51(1)(2008)107–113.
[4] CloudSuite 1.0, Web page at http://parsa.epfl.ch/cloudsuite/cloudsuite.html (Last access: 26.06.14).
[5] Eugen Feller,Lavanya Ramakrishnan,Christine Morin," Performance and energy efficiency of big data applications in cloud environments: A Hadoop case study", Journal of Parallel and Distributed Computing,Elsevier (2015)
[6] Shadi Ibrahim,Tien-Dat Phan,Alexandra Carpen-Amarie,Houssem-Eddine Chihoub,Diana Moise,Gabriel Antoniu," Governing energy consumption in Hadoop through CPU frequency scaling: An analysis", Future Generation Computer Systems,Elsevier (2016)
[7] Mukhtaj Khan, Yong Jin, Maozhen Li, Yang Xiang and Changjun Jiang, "Hadoop Performance Modeling for Job Estimation and Resource Provisioning", IEEE Transactions on Parallel and Distributed Systems.
[8] Javier Conejero, Omer Rana, Peter Burnap, Jeffrey Morgan, Blanca Caminero, Carmen Carrión," Analyzing Hadoop power consumption and impact on application QoS", Future Generation Computer Systems 55 (2016)
[9] Jacob Leverich, Christos Kozyrakis" On the energy (in)efficiency of Hadoop clusters", Volume 44 Issue 1,January2010, Pages61-65 ,ACM New York, NY, USA
[10] Rini T. Kaushik, Milind Bhandarkar" GreenHDFS: Towards An Energy-Conserving, Storage-Efficient, Hybrid Hadoop Compute Cluster", HotPower'10 Proceedings of the 2010 international conference on Power aware computing and systems, Article No. 1-9, Vancouver, BC, Canada
[11] Yanpei Chen, Archana Ganapathi" GreenHDFS: Towards An Energy-Conserving, Storage-Efficient, Hybrid Hadoop Compute Cluster", HotPower'10 Proceedings of the 2010 international conference on Power aware computing and systems, Article No. 1-9, Vancouver, BC, Canada
[12] Xiaoli Wang, Yuping Wang, Yue Cui" An energy-aware bi-level optimization model for multi-job scheduling problems under cloud computing", SpringerLink, January 2016, Volume 20, Issue 1, pp 303–317
[13] Maria Malik, Houman Homayoun, "Big data on low power cores: Are low power embedded processors a good fit for the big data workloads?", Computer Design (ICCD), 2015 33rd IEEE International Conference, 2015
[14] Lena Mashayekhy, Daniel Grosu, Quan Zhang, Mahyer Movahed Nejad, Weisong Shi, "Energy Aware Scheduling of MapReduce Jobs for Big Data Applications", IEEE Transactions on Parallel and Distributed Systems, Volume: 26, Issue: 10, Oct. 1 2015
[15] Zhuo Tang, Lingang Jiang, Junging Zhou, Kenli Li, Keqin Li "A self-adaptive scheduling algorithm for reduce start time", Future Generation Computer System, Volumes 43–44, Pages 51–60 (2015)
[16] Weikuan Yu, Yandong Wang, Xinyu Que, Cong Xu "Virtual Shuffling for Efficient Data Movement in MapReduce", IEEE Transactions on Computers, Volume: 64, Issue: 2 (2015)
[17] Kumar KA, Konishetty VK, Voruganti K, Rao GVP. CASH: Context Aware Scheduler for Hadoop. In: Proceedings of the International Conference on Advances in Computing, Communications and Informatics. New York, NY,USA: ACM; 2012. p. 52–61.
[18] Chen Q, Zhang D, Guo M, Deng Q, Guo S. SAMR: A self-adaptive MapReduce scheduling algorithm in heterogeneous environment. In: 10th International Conference on Computer and Information Technology. IEEE; 2010 p. 2736–43.
[19] Tian C, Zhou H, He Y, Zha L. A dynamic MapReduce scheduler for heterogeneous workloads. In: 8th International Conference on Grid and Cooperative Computing. 2009. p. 218–24.
[20] He C, Lu Y, Swanson D. Matchmaking: A new MapReduce schedul-

ing technique. In: Third International Conference on Cloud Computing Technology and Science. IEEE; 2011. p. 40–7.

[21] Zhang X, Wang G, Yang Z, Ding Y. A two-phase execution engine of reduce tasks in Hadoop MapReduce. In: International Conference on Systems and Informatics. 2012. p. 858–64.

[22] Zhang X, Zhong Z, Feng S, Tu B, Fan J. Improving data locality of MapReduce by scheduling in homogeneous computing environments. In: 9th International Symposium on Parallel and Distributed Processing with Applications. IEEE; 2011. p. 120–6. IEEE.

[23] Seo S, Jang I, Woo K, Kim I, Kim JS, Maeng S. HPMR: Prefetching and preshuffling in shared MapReduce computation environment. In: International Conference on Cluster Computing and Workshops. IEEE; 2009. p. 1–8.

[24] Nykiel T, Potamias M, Mishra C, Kollios G, Koudas N. MRShare: sharing across multiple queries in MapReduce. Proceedings of the VLDB Endowment 2010;3(1-2):494–505.

[25] Weikuan Yu,Yandong Wang, and Xinyu Que, "Design and Evaluation of Network-Levitated Merge for Hadoop Acceleration", IEEE transactions on parallel and distributed systems, vol. 25, no. 3, March 2014.

[26] Vernica R, Balmin A, Beyer KS, Ercegovac V. Adaptive MapReduce using situation-aware mappers. In: Proceedings of the 15th International Conference on Extending Database Technology. New York, NY, USA: ACM; 2012, p. 420–31.

[27] Ahmad F, Lee S, Thottethodi M, Vijaykumar T. Mapreduce with communication overlap (marco). Journal of Parallel and Distributed Computing 2013;73(5):608 –20.

[28] Lin M, Zhang L, Wierman A, Tan J. Joint optimization of overlapping phases in mapreduce. Performance Evaluation 2013;70(10):720 –35. Proceeding of {IFIP} Performance 2013 Conference.

[29] Ho LY, Wu JJ, Liu P. Optimal algorithms for cross-rack communication optimization in MapReduce framework. In: International Conference on Cloud Computing. IEEE; 2011. p. 420–7.

[30] Ibrahim S, Jin H, Lu L, Wu S, He B, Qi L. LEEN: Locality/fairness-aware key partitioning for MapReduce in the cloud. In: Second International Conference on Cloud Computing Technology and Science. 2010. p. 17–24.

[31] Jiang D, Ooi BC, Shi L, Wu S. The performance of MapReduce: an in-depth study. Proceedings of the VLDB Endowment 2010;3(1-2):472–83.

[32] Hammoud M, Rehman M, Sakr M. Center-of-Gravity reduce task scheduling to lower MapReduce network traffic. In: International Conference on Cloud Computing. IEEE; 2012. p. 49–58.

[33] Park J, Lee D, Kim B, Huh J, Maeng S. Locality-aware dynamic VM reconfiguration on MapReduce clouds. In: Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing. New York, NY, USA: ACM; 2012. p. 27–36. ACM.