ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

Enhancing Map Reduce Performance in Heterogeneous Distributed Environment

[1] Suyash Mishra, [2] Dr. Anuranjan Mishra
[1] Ph.D. Computer science, [2] H.O.D (CS, IT),
[1][2] Noida International University, G.B Nagar (U.P), India.

***Abstract* - Now a day's size of the data used in today's enterprises worlds has been growing at exponential rates day by day. This had triggered need to process and analyze the large volumes of data for business decision making quickly as well. MapReduce is considered as a core-processing engine of Hadoop, which is prominently used to cater continuously increasing demands on computing resources imposed by massive data sets. Highly scalable feature of MapReduce processing, allows parallel and distributed processing on multiple computing nodes. This paper talks about various scheduling methodologies and most appropriate one can be used for improving MapReduce processing .Also tried to identify scheduling methods scaling or processing limitations along with the situations wherein they can be best suited. Map Reduce is used majorly for short jobs, which eventually require low response time. The current Hadoop implementation assumes underline computing nodes in a cluster are homogeneous, have same processing capability and memory. Hadoop's scheduler suffers from severe performance degradation in heterogeneous environments. In heterogeneous environment, Longest Approximate Time to End (LATE) scheduling can be most efficient in comparison to other scheduling .It has been seen in various studies that LATE has improved Hadoop response times by approximately two times in a clusters.**

***Key Words*— Big data, HDFS, Hadoop, Map-Reduce, Scheduling Algorithm, LATE**

I. INTRODUCTION

Traditional data storage and processing capabilities were limited and was dependent on available hardware, storage and processing requirements, which deemed to be very different from today. Thus, those approaches and databases are facing tremendous threat while coping Big Data processing and storing demands.

Now a days Industry is focusing and making huge investment to conclude how to make better use of Big Data and identify beneficial business insights to lead a better business decisions .Which help business or industry to increase profit. MapReduce is a highly scalable programming model capable of processing huge volume of data in parallel execution fashion on a huge number of commodity computing nodes. Google [3] recently popularized it.MapReduce paradigm has been implemented in many open source projects, but most prominently adopted by the Apache Hadoop later [4].

MapReduce follows a flexible computation model with a simple interface consisting of a map and reduce functions. Application developers can customize these two functions implementations. Main advantage of Map Reduce paradigm is efficient node failure handling while hiding the complexity to manage fault-tolerance from the programmer. In case of a homogeneous environment, all nodes are considered identical in both in terms of computation power and disk capacity. In event of node crash Map Reduce identifies another idle or underutilized node to reruns failed tasks on it. There may be case when a node is processing a task poorly or slowly which is known as straggler task which may be time taking, rather than waiting for task to complete Map Reduce runs a backup copy of slow running on another machine which is able to finish task quickly. Without speculative execution, a job will be as slow as the misbehaving task. Straggler task can occur due to various reasons e.g. faulty hardware and misconfiguration. It has been observed that speculative execution can improve job response times by 44%.

Speculative execution problem aggravates in Heterogeneous environment, which deteriorate MapReduce performance due to difference in nodes storage and processing capacity.

2. HADOOP

Hadoop is an open source-processing engine designed to process extremely high volumes of structured or unstructured data. Hadoop is based on distributed processing which provides resilience, fault tolerance and

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

scalability along with efficient Big data processing. Primarily Hadoop has two main components HDFS and MapReduce: Hadoop distributed file system (HDFS), which stores data in structured relational form or unstructured form or any variety of data. HDFS is a highly distributed file system that ensures high turnaround times to retrieve data along with high storage scalability. Second component of Hadoop is MapReduce, which is meant for managing and running applications on multiple distributed servers. It is a framework for executing high performance distributed data processing MapReduce is based on divide and aggregate paradigm. HDFS, or the Hadoop Distributed File System, allows unlimited storage with minimal impact on data retrieval time.
Figure2 below depicts Hadoop and it's both components.

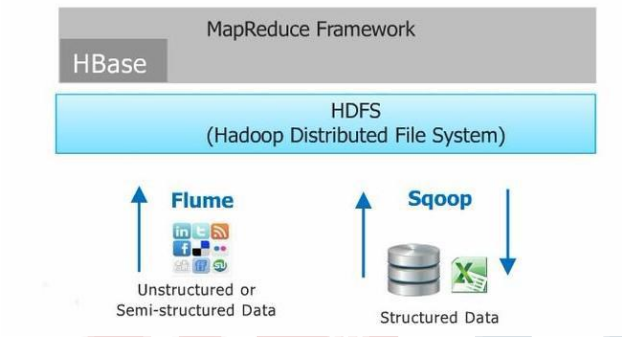

Figure2.

3. MAPREDUCE- PROCESSING OF BIGDATA

MapReduce is a programming model designed for processing large volumes of data in parallel fashion by breaking the work into a set of independent sub processing units.one of the MapReduce most significant advantages is that it hides many system level details and complexities from programmer and provides convenient to programmer. It processes data by dividing the progress into two phases: Map and Reduce. Each Map function takes a split data file as its input data, which can be locates in the distributed file system and keeps the key/value data. The split data file can be co-located with the Map function or not. If data and the Map function do not reside in the same node, then the system will attempt to move the split file to the Map function where it exists. This is why MapReduce follows "Moving data closer to compute" concept to reduce processing time. Further Reduce function is applied to all values that associated with the same intermediate key and generates output key/value pairs as the result. The MapReduce framework works on master/slave architecture. There is a single master node, which has JobTracker and several slave nodes, which runs assigned task, and Tasktrackers maintain assigned task's progress status which executes one per node in the cluster. The JobTracker acts as interface between users who submits the job and the underline framework. Users submit map/reduce jobs to the JobTracker and jobs are executes basis of a first come/first-served basis. The JobTracker manages the allocation of map and reduce tasks to the slave nodes running tasktrackers. The Tasktrackers accepts and run tasks based on instruction from the JobTracker, manages data transfer between the map, and reduce phases.

1. Map – The master node takes the input, partitions it up into smaller sub-problems, and distributes them to slave nodes. A slave node may do this again in turn, leading to a multilevel tree structure. Map takes one pair of data with a type in one data domain, and returns a list of pairs in a different domain: Map (k1, v1) → list (K2, v2)

2. Execute user-submitted Map logic – Map logic will is run once for each K1 key value, produce output grouped by key values K2.

3. Send the Map output to the Reduce processors – the Map Reduce system designates a node to run Reduce function, assigns the K2 key value to each processor, and provides that node with all the Map-produced data associated with that key value.

4. Execute the user-provided Reduce code – Reduce is run exactly once for each K2 key value produced by the Map step.

5. Generate the final output – the Map Reduce system combines all the Reduce output, and sorts them by K2 to produce the outcome. Below diagram elaborates above-mentioned steps using word count example following big data Map Reduce processing..

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

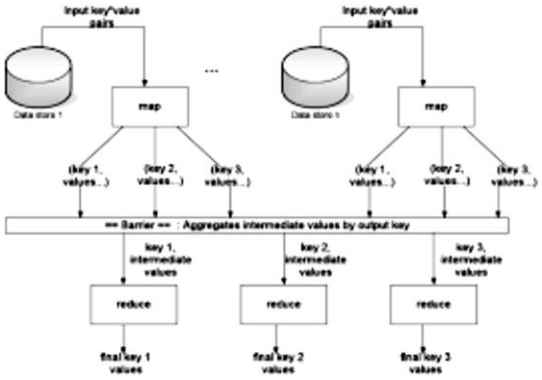

Figure3.

In a homogeneous environment where all nodes are same in both computing speed and disk capacity. In case of node failure, MapReduce attempts to reruns failed tasks on a different idle/less-occupied node. There may be case when a node is processing a task poorly/slowly known as straggler; Map Reduce runs a speculative copy of straggler task ("backup task") on another machine able to process quickly. In absence of speculative execution, a job will be as slow as the misbehaving task. Stragglers can arise due to numerous reasons including faulty hardware and misconfiguration. Google [8] .It has been observed that speculative execution can improve job response times by 44%.In Heterogeneous environment this problem is deteriorate the execution of speculative execution due to difference in nodes storage and processing capability.

4. SPEECULATIVE EXECUTION IN HADOOP

When HADOOP observes that s node is processing a task poorly/slowly which is known as straggler; Map Reduce runs a speculative copy of straggler task ("backup task") on another machine able to process quickly. The selection of idle or underutilized node is done on basis of three conditions utmost priority is allocated to any failed tasks and secondly, non-running tasks are considered for processing. For maps, nodes where local data resides at node are chosen first. Thirdly, Hadoop selects a speculative task to execute. For selection of speculative tasks, Hadoop monitors task progress using a progress score between zero and one. For a map, the progress score is the fraction of input data read. For a reduce task, the execution is divided into three steps, each steps is given credited for 1/3 of the score:

• Copy phase, when the task receives map outputs.
• Sort phase, when map outputs are sorted by key.
• Reduce phase, when a user-defined function executes on the list of map outputs with each key.

In each steps, the score is the fraction of data processed. For example, a task halfway through the copy step gets a progress score of 1/ 2 * 1/3=1/6 ,While a task halfway through Sort step gets progress score 1/3+(1/2*1/3)=3/6=1/2 , while a task halfway through the reduce step scores 1/ 3 + 1/ 3 + (1/ 2 * 1/ 3) = 5 /6 . Hadoop monitors average progress score of each category of tasks (maps and reduces) to compute a maximum time allowed for execution for speculative execution. All tasks beyond the allowed time is considered as "Slow". The speculative scheduling works quite well in homogenous environments because of similar processing speeds of each nodes tasks begin processing at same time and finish is roughly the same times and speculation only begin when the slow task is running. During multiple jobs, Hadoop uses a FIFO scheduling to compute the order of running a job where the first submitted job will be running first job, followed by next submitted job in sequence etc.

Hadoop scheduler follow below implicit assumptions during working on speculative scheduling.

1. All nodes are considered of similar hardware configuration hence possess same as benmap or reduce) takes approximately same amount of time.

Above Hadoop assumption of holds quite well in Homogeneous environment however, assumptions 1 and 2 suffers processing in heterogeneous cluster. However, in homogeneous environment also, assumptions 3, 4 and 5 can be violated and can result into performance degrade. Due to this constraint, Yahoo disabled speculative execution on some jobs because it degrades performance, and monitors faulty machines via different alternative approaches. Facebook also adopted similar method and disables speculation for reduce tasks. Tasks in MapReduce should be small else, a single large task will bring down the entire job processing and destroy the data parallelism feature objective of MapReduce. In a well-behaved MapReduce job, the separation of input into equal parts and the division of the key space among

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

reducers ensures roughly equal amounts of work. If this is not the case, then launching a few extra speculative tasks is not harmful as long as obvious stragglers are also detected.
When there is heterogeneous environment so it's hard to utilize the power of each node correctly this may cause performance degradation

5. SCHEDULING IN HADOOP MAPREDUCE

There are various scheduling approaches, which has been recommended to reduce speculative tasks .Major scheduling algorithms, below are listed which are prominently used in MapReduce. Each scheduling features, advantages, and disadvantages has been elaborated to decide which scheduling is best suited for which scenario..

5.1 FIFO SCHEDULING

In FIFO scheduling, jobs are processed and prioritized on their sequence of arrival basis. This approach is quite easy to implement and cost of implementation is not high. This is best suited for single type of job; however, this suffers from performance issues while multiple kind of jobs are assigned. Performance also degrades on the processing of shorter job in comparison to process long jobs.
Advantages: This is simple to implement and efficient for mono kind of jobs.
Disadvantages: Has no priority consideration or size of the job consideration.

5.2 FAIR SCHEDULING

In Fair scheduling assignment of resources to jobs is done is such a manner so that all jobs get, on average, an equal amount of resources over time. Job uses the entire resources when there is a single job in a cluster. When new jobs arrives, tasks slots that free up are assigned to the new jobs, so that each job gets fair amount of CPU time. Unlike the default Hadoop scheduler, which makes a queue of jobs, this makes short jobs finish in reasonable time while not starving long jobs fair scheduling can also work with job priorities - the priorities are used as weights to determine the fraction of total compute time that each job gets.
Advantages:
1. Less complex
2. Works well with both small and large clusters
3. Provides faster response for small jobs mixed with larger jobs Disadvantages

Disadvantages: Does not consider the job weight of each node

5.3 CAPACITY SCHEDULAR

Capacity scheduling is based on queues concept , which are assigned resources, and they uses FIFO strategy in itself. At the time of scheduling, all queues are monitored, if a queue does not use its allocated capacity, the underutilized capacity can be utilized by assigning to other queues. Jobs with a higher priority can access to resources sooner than lower priority jobs. Yahoo! developed capacity scheduler.
Advantages: Optimized utilization of resources and throughput in multitenant cluster environment.
Disadvantages: Does not ensure guaranteed access with the potential to reuse unused capacity and prioritize jobs within queues over large cluster.

5.4 DYNAMIC PRIORITY SCHEDULING

In dynamic priority scheduling of Hadoop, users are allowed to increase and decrease their queue priorities continuously to meet the current workloads requirements. Scheduler is aware of the current demand and makes it more expensive to boost the priority under peak usage times. Thus users who move their workload to low usage times are being discounted. Priorities can only be increased within a specified quota limit. All users are assigned a quota, which is deducted and calculated periodically in configurable accounting intervals. Deducible budget is determined by a per-user consumption rate, which may vary time to time directly by the user. It maintains capacity distribution dynamically among concurrent users based on priorities of the users. It ensures users to get Map or Reduce slot on a proportional share basis per time unit. These time slots can be configured and called as allocation interval. This model encourages users with small tasks than users with longer time-consuming jobs.
Following are primitive features of Dynamic priority scheduling.
1. Discourages the free riding and gaming by users.
2. Possible starvation of low-priority tasks can be reduced by using the standard approach in Hadoop of restricting the time each task allowed to run on a node.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

3. Define a time budgets for different users and let them individually decide whether the current price of preempting running tasks is within their budget or if they should wait until the current users run out of their budget.
Advantages: Can be easily configured in comparison to other scheduling methods.
Disadvantages: All unfinished low priority processes gets lost at the time of system crash.

5.5 DELAY SCHEDULING

In delay scheduling when a node requests a task, if the head-of-line job cannot launch a local task, task is not processed that time and delayed at moment, look at subsequent jobs. However, if a job has been delayed from processing long enough, job is allowed to launch non-local tasks, to avoid starvation. The key objective behind delay scheduling is that although the first slot we consider giving to a job is unlikely to have data for it, tasks finish so quickly that some slot with data for it will be processed.

5.6 RESOURCE AWARE SCHEDULING

Resource Aware Scheduling in Hadoop is a biggest research Challenges in Cloud Computing. Job Tracker running at master node take scheduling decisions in Hadoop. The Job Tracker maintains a queue of currently running jobs, states of Task Trackers in a cluster and nodes, and list of allocated tasks to each Task Tracker. Each Task Tracker node is currently configured with a maximum number of available computation slots. Each Task Tracker node monitors resources such as CPU utilization, disk channel IO in bytes/s, and the number of page faults per unit time for the memory subsystem.
Below Are two resource aware scheduling approaches:
I. Dynamic Free Slot Advertisement
II. Free Slot Priorities/Filtering

6 HOW MAPREDCUE ASSUMPTIONS ARE VIOLETED IN HETEROGENEOUS ENVIROUNMENT

6.1 HETEROGENITY

The first two assumptions are about homogeneous nature of clusters nodes. There may be multiple generations of hardware in a non-virtualized data center. However, in a virtualized data center, multiple virtual machines run on each physical host, such as Amazon EC2, colocation of VMs may cause heterogeneity [8] having different processing capacity.
Enhancing MapReduce performance in heterogeneous environments and propose solutions to improve its performance. Each approach attempts to improve one of underperforming areas of Map reduce features in a heterogeneous cluster.
The algorithms that were identified to improve processing can be categorized into below two categories:
1. Data Locality Algorithms.
2. Fault Tolerance Algorithms.

6.1.1. DATA LOCALITY ALGORITHMS

Data placement strategy plays vital role, as in case of homogeneous environment all nodes are assumed similar in processing capability and storage capacity. In heterogeneous Hadoop cluster, a high-performance node can complete processing local data faster than low-performance node. Data placement strategy advocate how data placement can be optimized in Heterogeneous Hadoop Clusters to improve MapReduce performances.

6.1.1.1 DATA PLACEMENT IN HETEROGENEOUS HADOOP CLUSTERS

In heterogeneous Hadoop cluster, a high-performance node can complete processing local data faster than low-performance node. After the fast node finished processing data residing in its local disk, the fast node has to take care of processing of the unprocessed data in remote slow node. The overhead of transferring unprocessed data from slow node to fast node is very high if the amount of transferred data is huge. An approach to improve Map Reduce performance in heterogeneous environments is to reduce the amount of data moved between slow and fast nodes in a heterogeneous clustering
J. Xie et al. [9] advises a data placement mechanism in HDFS that distributed and stored a large data set across multiple heterogeneous nodes in function of nodes computing capacity. In other words, the number of file.

6.1.1.2 INITIAL DATA PLACEMENT

The initial data placement chops a large input file into a number of even-sized parts. A data distribution server handles the responsibility of distributing the file portions across the nodes of the cluster. It applies the round-robin algorithm to assign the input file portions to the heterogeneous nodes based on their computing ratios. A

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

small value of computing ratio indicates a high speed of node, meaning that the fast node must process a large number of fragments. In addition, a large value of computing ratio of a node indicates a low speed of the node, meaning that the slow node must process a small number of file fragments.

6.1.1.3 DATA REDISTRIBUTION

Input file portions distributed by the initial data placement algorithm [10] might be disrupted due to the following reasons:

1. New data is added and appended to an existing input file.
2. Data blocks are deleted from the existing input File
3. New data computing nodes are added into an existing cluster. To address this dynamic data load-balancing problem.

6.1.1.4 DATA LOCALITY AWARE TASK SCHEDULING METHOD FOR HETEROGENEOUS ENVIRONMENTS

Zhuoyao zhang [9] advocated method to improve data locality of Map reduce in homogeneous computing environments. The method considers that noes will finish is same amount of time as due to same processing capability of each node .However, this assumption cannot be very effective in terms of performance in heterogeneous environment due to numerous factors that can change the processing speed of the processors such as, the heterogeneity of the computational resources and its dynamic workload.

X. Zhang et al. [9] introduced a data locality aware scheduling method for heterogeneous Hadoop cluster. There are two factors affect the efficiency of map tasks execution-waiting time is the shortest time that the task has to wait before it can be scheduled to one of the nodes that have the input data, transmission time is the time needed to copy the input data of the task to the requesting node.

The goal is to make a balance between the waiting time and transmission time at runtime when schedule a task to a node to obtain the optimal task execution time. In event of task process, request priority is given to task whose input data is present in the requesting node. If no such tasks, the method selects the task having input stored in the nearest to requesting. Further method calculates wait time and determine transmission time of the allocated task. If the waiting time is less than transmission time, then method reserves the task for the node having the input data. Otherwise, it schedules the task to the requesting node.

6.2 FAULT TOLERANCE ALGORITHMS

Main advantage of Map reduce is its ability to identify, manage node failures and hides the complexity of the fault tolerance from the programmers. Hadoop's performance is dependent on its task scheduler, which considers that the cluster nodes are homogeneous and tasks make progress linearly. Hadoop's scheduler uses these assumptions to decide when to speculatively re-execute tasks that appear to be stragglers. Hadoop's scheduler starts speculative tasks based on a simple heuristic comparing each task's progress to the average progress of each task. This heuristic gives better results in the homogeneous environments where the stragglers are obvious. Hadoop's scheduler can degrade server performance in heterogeneous environments because the underlying assumptions are broken.[6]

We will be discussing the algorithms to improve fault tolerance support in the heterogeneous Hadoop.

6.2.1 LONGEST APPROXIMATE TIME TO END

Longest Approximate Time to End Algorithm (LATE) is based on approach that if a node has an empty task slot, Hadoop chooses a task for it from one of three categories. First, priority is given to a failed tasks Second, non-running tasks are considered, specially the map tasks that have local data on this node. Third, speculative execution task.

Hadoop defines a threshold for speculative execution using the average progress score of each category of tasks (maps and reduces). When a task's progress score is less than the average off its category, and the task has run at least one minute, it is flagged as a straggler. LATE always speculatively executes the task which will finish longest in the future. LATE estimates the task's finish time based on the progress score provided by Hadoop. Hadoop computes the progress rate of each task as Progress Score, and then compute the task's finish time.

There are scenarios where this method can be unsuccessful, but this runs well in typical Hadoop jobs. Ideally, we should only launch speculative tasks on fast nodes --not stragglers. We do this through a simple heuristic -do not launch speculative tasks on nodes that

are below some threshold, Slow Node Threshold, of total work performed (sum of progress scores for all succeeded and in progress tasks on the node). This approach gives better performance than assigning a speculative task to the first available node.

Advantages of LATE algorithm [6] has several advantages. It is robust in heterogeneous environment, because it will re-launch only the slowest tasks, and only a small number of tasks.

LATE prioritizes among the slow tasks based on how much they affect job response time.

LATE also limits the number of speculative tasks to limit contention for shared resources. Comparatively, Hadoop's native scheduler has a fixed threshold, after which all tasks that are slow enough and have an equal chance of being launched. This fixed threshold can cause excessively many tasks to be speculated upon.

LATE takes into account node heterogeneity when deciding where to run speculative tasks. Whereas, Hadoop's core scheduler assumes that any node, which completes a task and requests for a new one, is likely to be a fast node, i.e. that slow nodes will never finish their original tasks and never be member for running speculative tasks.

At last but not least, by focusing on estimated time left instead of progress rate, LATE speculatively executes only those many tasks that will improve job response time, rather than any slow tasks.

7. CONCLUSION AND FUTURE WORK

MapReduce has been regarded as prominent programming paradigm to cope with Big data processing .Though MapReduce offers numerous advantages but there are few trade-offs faced in meeting, the rapidly growing computing demands of Big Data in heterogeneous environment. There are many scheduling methodologies proposed .Our aim is to identify and categorize related scheduling algorithms, their capability to address MapReduce challenge to work efficiently in Heterogeneous environment. .This enables better planning of Big data projects. Future work on this will be develop a scheduling change in Hadoop MapReduce which will work efficiently using LATE scheduling approach as this is described as most suited approach among all proposed scheduling methodologies .

8 REFERENCES

[1] Dean, J., Ghemawat, S.: MapReduce: Simplified data processing on large clusters. In: OSDI 2004, San Francisco, CA, pp. 137–150 (December 2004)

[2] Hadoop MapReduce, http://hadoop.apache.org/mapreduce/

[3]. Thusoo, A., Shao, Z., Anthony, S., Borthakur, D., Jain, N., Sen Sarma, J., Murthy, R., Liu, H.: Data warehousing and analytics infrastructure at facebook. In: Proceedings of the 2010 International Conference on Management of Data, SIGMOD 2010, pp. 1013–1020. ACM, New York (2010)

[4]. Ananthanarayanan, G., Kandula, S., Greenberg, A., Stoica, I., Lu, Y., Saha, B., Harris, E.: Reining in the outliers in map-reduce clusters using mantri. In: OSDI 2010, pp. 1–16. USENIX Asoc., Berkeley (2010)

[5]. Polo, J., Carrera, D., Becerra, Y., Steinder, M., Whalley, I.: Performance-driven task co-scheduling for MapReduce environments. In: Network Operations and Management Symposium, NOMS, pp. 373–380. IEEE, Osaka (2010)

[6] Wolf, J., Rajan, D., Hildrum, K., Khandekar, R., Kumar, V., Parekh, S., Wu, K.-L., Balmin, A.: Flex: A Slot Allocation Scheduling Optimizer for Mapreduce Workloads. In: Gupta, I., Mascolo, C. (eds.) Middleware 2010. LNCS, vol. 6452, pp. 1–20. Springer, Heidelberg (2010)

[7] Dynamic Proportional share scheduling in Hadoop Thomas sandholm and Kevin Springer Berlin Heidelberg Volume 6253, 2010, pp 110-131

[8]Improving Map Reduce Performance through Data Placement in Heterogeneous Hadoop Clusters- Jiong Xie, Shu Yin, Xiaojun Ruan, Zhiyang Ding, Yun Tian, James Majors, Adam Manzanares, and Xiao Qin -Department of Computer Science and Software Engineering Auburn University, Auburn, AL 36849-5347

[9] An Empirical Analysis of Scheduling techniques for Real-time cloud based data processing-linh T.X. Phan

Zhuoyao zhang, Qi Zheng Boon Thau Loo University of Pennsylvania

[10] Herodotou, H., and Babu, S. Profiling, what-if analysis, and cost-based optimization of MapReduce programs. In Proc. Int' Conf. on Very Large Data Bases (VLDB) (2011).

[11] MapR. The executive's guide to big data. http://www.mapr.com/resources/white-papers.

[12] Pettijohn, E., Guo, Y., Lama, P., and Zhou, X. User-centric heterogeneity-aware mapreduce job provisioning in the public cloud. In Proc. Int'l Conference on Autonomic Computing (ICAC) (2014).

13] Herodotou, H., Lim, H., Luo, G., Borisov, N., Dong, L., Cetin, F. B., and Babu, S. Starfish: A self-tuning system for big data analytics. In Proc. Conference on Innovative Data Systems Research (CIDR) (2011).

[14] Lama, P., and Zhou, X. Aroma: Automated resource allocation and configuration of mapreduce environment in the cloud. In Proc. Int'l Conf. on Autonomic computing (ICAC) (2012).

[15] Li, X., Wang, Y., Jiao, Y., Xu, C., and Yu, W. Coomr: Cross-task coordination for efficient data management in mapreduce programs. In Proc. Int'l Conference for High Performance Computing, Networking, Storage and Analysis (SC) (2013).

[16] Kambatla, K., Pathak, A., and Pucha, H. Towards optimizing hadoop provisioning in the cloud. In Proc. USENIX HotCloud Workshop (2009).

[17] Li, Z., Cheng, Y., Liu, C., and Zhao, C. Minimum standard deviation difference-based thresholding. In Proc. Int'l Conference on Measuring Technology and Mechatronics Automation (ICMTMA) (2010).

[18] Jinda, A., Quian-Ruiz, J., and Dittrich, J. Trojan data layouts: Right shoes for a running elephant. In Proc. of ACM Symposium on Cloud Computing (SoCC) (2011).