International Journal
of Computer Sciences
ISSN Number: 4142 3453

NOVEL SCHEDULING ALGORITHM FOR EFFICIENT DEPLOYMENT OF MAPREDUCE APPLICATIONS IN HETEROGENEOUS COMPUTING ENVIRONMENT

S. Vara Lakshmi[1], Shaik Jaffar Hussain[2]

[1]M.Tech (CSE),PG Scholar ,Department of CSE, Srinivasa Institute of Technology and Science Kadapa,AP.

[2]Associate Professor ,Department of CSE, Srinivasa Institute of Technology and Science Kadapa,AP.

ABSTRACT

The open source Hadoop and Mapreduce framework are the defacto platform for scalable analysis on large data sets. How to reduce the completion length of a set of MapReduce jobs is one of the primary concerns in Hadoop The MapReduce is an open source Hadoop framework implemented for processing and producing distributed large Terabyte data on large clusters. . The current open source Hadoop allows only static slot configuration, like fixed numbers of map slots and reduce slots throughout the cluster lifetime. Such static configuration may lead to long completion length as well as low system resource utilizations. Its primary duty is to minimize the completion time of large sets of MapReduce jobs. Hadoop Cluster only has predefined fixed slot configuration for cluster lifetime. This fixed slot configuration may produce long completion time (Makespan) and low system resource utilization. Our proposed scheme is to allocate resources dynamically to MapReduce tasks. It can be done by following slot ratio configuration between map and reduce tasks, by updating the workload information of recently completed tasks. Many scheduling methodologies are discussed that aim to improve completion time goal. Propose new schemes which use slot ratio between map and reduce tasks as a tunable knob for minimizing the completion length (i.e., makespan) of a given set. By leveraging the workload information of recently completed jobs, schemes dynamically allocates resources (or slots) to map and reduce tasks.

Index Terms- MapReduce, Makespan, Workload, Dynamic Slot Allocation.

1. INTRODUCTION

A classic Hadoop cluster has a single name node and multiple data nodes. The name node, which is configured with job tracker, is responsible for job scheduling and job execution co-ordination. Each data node configured with task tracker, which manages MapReduce slots. Hadoop has a static slot configuration, which means a fixed number of map slots and reduce slots which are only used for processing map reduce tasks. Map tasks can run by map slots, and reduce tasks can run in reduce slots. This static slot configuration may lead to poor performance and low resource utilization. Apache Hadoop components are responsible for running large data sets. Main Hadoop parallel processing components are Hadoop Distributed File System (HDFS), Hadoop YARN, and Hadoop MapReduce.

We propose dynamic slot configuration, which dynamically allocates slots for map and reduce tasks. Our aim is to modify name node functionality, which means to increase additional responsibility for monitoring workload information, dynamic slot assignment, and scheduling. Also, we need to modify the task tracker slot allocation policy to dynamically allocate tasks to MapReduce tasks without any slot specification. We can make use of map task slots (map slots) to reduce slots and vice versa. The main idea behind dynamic slot configuration is to avoid idle slot in the MapReduce slots. The job tracker estimates the current workloads in each task tracker using workload monitoring component.

MapReduce[1] for processing big data in parallel. Its open source implementation Apache Hadoop [2] has popular platform for data processing and information analysis. With the rise of cloud computing, It is now convenient for a regular user to launch a MapReduce cluster on the cloud, e.g., AWS MapReduce, for data-intensive applications. How to improve the performance of a MapReduce cluster becomes a focus of research and development [3– 11]. As a complex system, Hadoop is configured with a large set of system parameters. While it provides the flexibility to customize the cluster for different applications, it is challenging for users to understand and set the optimal values for those parameters. In this paper, aim to develop algorithms for adjusting a basic system parameter with the goal to improve the performance (i.e., reduce the makespan) of a batch of MapReduce jobs.

A Hadoop cluster has single masternode and multiple slave nodes. The master node runs the JobTracker routine which is responsible for scheduling jobs and coordinating the execution of tasks of each job. Each slave node runs the TaskTracker for hosting the execution of MapReduce jobs. The concept of "slot" is used to indicate the capacity of accommodating tasks on each node. In a Hadoop system, a slot is assigned as a map slot or a reduce slot serving map tasks or reduce tasks, respectively. At any given time, only one task can be running per slot. The number of available slots per node indeed provides the maximum degree of parallelization in Hadoop. Here shown that the slot configuration has a significant impact on system performance. The Hadoop framework use fixed numbers of map slots and reduce slots at each node as the default setting throughout the lifetime of a cluster. The values in this fixed configuration are usually heuristic numbers without considering job characteristics. Therefore, this static setting is not well optimized and may hinder the performance improvement of the entire cluster. In this work, propose and implement a new mechanism to dynamically allocate slots for map and reduce tasks. The primary goal of the new mechanism is to improve the completion time (i.e., the makespan) of a batch of MapReduce jobs while retain the simplicity in implementation and management of the slot- based Hadoop design.

The key idea of this new mechanism, named TuMM, is to automate the slot assignment ratio between map and reduce tasks in a cluster as a tunable knob for reducing the makespan of MapReduce jobs.The Workload Monitor (WM) and the Slot Assigner (SA) are the two major components introduced by TuMM. The WM that resides in the JobTracker periodically collects the execution time information of recently finished tasks and estimates the present map and reduce workloads in the cluster. The SA module takes the estimation to decide and adjust the slot ratio between map and reduce tasks for each slave node. With TuMM, the map and reduce phases of jobs could be better pipelined under priority based schedulers, and thus the makespan is reduced. Further the dynamic slot assignments in heterogeneous environments, and propose a new version of TuMM, named H TuMM, which sets the slot configurations for each individual node to reduce the makespan of a batch of jobs.

The slot assigner component decides the optimum slot for assigning tasks. The schedulers are used to schedule the tasks in the data nodes.
The task tracker sends the status report to the job tracker for every 3 minutes. Failure tasks are assigned to the next nodes based on this status report. The job tracker is always monitoring the task execution and slot assignment.

The resources are allocated to map and reduce tasks by job tracker based on different job schedulers and resource allocation policies. Various schedulers are used that include FIFO, capacity, SLO, task schedulers, fair scheduler.

These schedulers follow different resource allocation strategies that include the Longest Approximation Time to End, delay, resource aware, deadline constraint, epoch based, moldable, malleable, fair4s job scheduling to improve MapReduce completion time and Hadoop performance.

The purpose of this study is to analyze the various slot configurations, advantages, and disadvantages of all schedulers and also different resource allocation policies in MapReduce.

2. MAP REDUCE THEORY

MapReduce: MapReduce is a framework for processing parallelizable problems across huge datasets using a large number of computers, collectively referred to as a cluster (if all nodes are on the same local network and use similar hardware) or a grid (if the nodes are shared across geographically and administratively distributed systems, and use more heterogeneous hardware). Computational processing can occur on data stored either in a file system (unstructured) or in a database (structured). MapReduce takes advantage of the locality of data, data processing on or near the storage assets in order to decrease the data transmission. Figure 2 describes Hadoop MapReduce process, which involves input dat, split phase, Map phase, Intermediate data, Reduce phase and Output data. HDFS: Hadoop uses Hadoop distributed File System (HDFS) which is an open source implementation of the Google File System (GFS) for storing data. HDFS is a distributed file system that not only stores the data but also ensures fault tolerance through replication designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is highly faulttolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets.

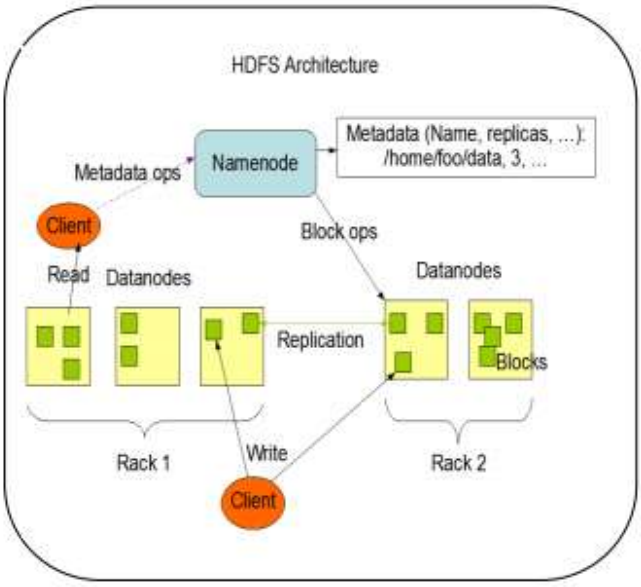


Fig 2.HDFS Architecture

Figure 2 describe HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The DataNodes are responsible for serving read and write requests from the file system's clients. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

Improvements In Job Scheduling

Delay Scheduling is an outcome of strict implementation of fair sharing compromising locality. To resolve this problem of locality, Delay scheduling algorithm was proposed, in which a job waits for a limited amount of time for a scheduling opportunity on a node that has data for it. The main goal of Delay Scheduling is to statistically multiplex clusters while maintaining minimal impact on fairness and achieving high data locality. Delay scheduling algorithm temporarily relaxes fairness to improve locality by asking jobs to wait for a scheduling opportunity on a node with local data. Two locality problems were identified from fair scheduler are: head-of-line scheduling and sticky notes. The first locality problem occurs in small jobs. Whenever a job reaches the head of the sorted list for scheduling, one of its tasks is launched on the next slot that becomes free irrespective of which node the slot is on.

3. RELATED WORK

In literature, there was research study on performance optimization of Hadoop MapReduce jobs. An essential way for upgrading the performance of a MapReduce job is dynamic slot configuration and job scheduling. J. Polo et al. [2] calculated the map and reduce task completion time dynamically and update it every minute during job execution. Task scheduling policy was based on the priority of each job. Priority was estimated based on the concurrent allocation of jobs. The dynamic scheduler is pre-emptive. It affects resource allocation of low priority jobs. J. Wolf et al. [3] implemented flexible scheduling allocation scheme with Hadoop fair scheduler. A primary concern is to optimize scheduling theory metrics, response time, makespan, stretch, and Service Level Agreement. They proposed penalty function for measurement of job completion time, epoch scheduling for partitioning time, moldable scheduling for job parallelization, and malleable scheduling for different interval parallelization.

J. Dean et al. 2008 [1] discussed MapReduce programming model. The MapReduce model performs operations using the map and reduces functions. Map function gets input from user documents. It generates intermediate key/value for reducing function. It further processes intermediate key/value pairs and provide output key/value pairs. At an entry level, MapReduce programming model provided the best data processing results. Currently, it needs to process the large volume of data. So it provides some consequences while processing and generating data sets. It takes much execution time for task initialization, task coordination, and task scheduling. Parallel data processing may lead to inefficient task execution and low resource utilization.

Verma et al. [5] proposed deadline aware scheduler, called SLO scheduler. The SLO scheduler takes decisions of job ordering and slot allocation. This scheduler's primary duty is to maximize the utility function by implementing the Earliest Deadline First algorithms. It measures how many numbers of slots required for scheduling the slots dynamically with a particular job deadline. B. Sharma et al. [7] proposed a global resource manager for the job tracker and a local resource manager for the task tracker. A global resource manager function is to manage each MapReduce task. It processes resource needs and resource assignments for each task. A local resource manager's duty is to identify each task. It examines resource usage and task completion time of the task. It deals with detecting bottlenecks with resources and resource contention.

Apache Hadoop released next generation MapReduce, called YARN [8]. It replaces MRv1 fixed slot configuration. YARN deals with CPU cores and memory requirements. It splits the job tracker into two components; they are resource managements and job scheduling. MapReduce tasks assignment is based on CPU cores and memory requirement of each task. YARN users simply update their MRv1 by installing mrv2 compatibility API and recompile the MRv1 application. J.

Wang et al. [9] proposed fair slot setting for dynamically allocate available slots to particular tasks. They used FRESH for static and dynamic slot configuration. The static slot configuration slots are allocated before cluster launch based on previous task execution records. It uses deduct workload function to update current workloads of running jobs in the cluster. The fair scheduler was proposed to achieve fairness metric. The dynamic slot assignment slots are allocated during task execution. It used Johnson indices to represent the level of fairness.

S. Tang et al. [11] proposed three techniques to improve MapReduce performance. They categorized utilized slot into the busy slot and idle slot respectively. The primary concern is to increase the number of the busy slots and decrease number of idle slots. Dynamic Hadoop Slot Allocation observes idle map and reduce slots. DHSA allocated the task to the unallocated map slots for overflowed reduce slots. Speculative Execution Performance Balancing provides performance upgrade for a batch of jobs. It gives the highest priority to failed tasks and next level priority to pending tasks. The slot prescheduling improves the performance of slot utilization with data locality without any negative effects on fairness metric. A.U. Patil et al. [13] discussed scheduling algorithms in the MapReduce environment. They analyzed schedulers and scheduling policies. The default FIFO scheduler follows the First in First Out queue for schedules the job. A single job is divided into a small number of chunks called tasks. The FIFO queue allocates tasks to free slots presented in the task tracker. The fair scheduler provides the fair share of resources to cluster users. Capacity scheduler estimates the number of users sharing cluster resources and focus fair allocation of resources to users. The primary concern is to maximize the throughput and utilization of entire cluster. Scheduling policies include the Longest Approximation Time to End, Deadline constraint, delay scheduling, resource aware, and Fair4s scheduling.

Z. Liu focused [14] partition skew problem. Data skewness causes the problem in execution time for larger and smaller tasks. Commonly this can be raised while partitions are unevenly distributed by the hash function. They proposed a new architecture called DREAMS. It predicts partition size and estimates reduce task performance metrics like CPU and memory impacts. Reduce phase performance model also detects the relationship between partition size and task execution time. After completion of reduce task performance estimation, DREAMS allocates resources to tasks.

Y. Yao et al. [15] proposed Tunable knob for reducing the Makespan of MapReduce (TUMM) for dynamic slot configuration. They modified the job tracker functionality by adding additional components. Main components are workload monitor and slot assigner. The workload monitor collects information about running and completed workloads. The slot assigner finds the best slot for dynamically assigning MapReduce tasks in the task tracker. They used FIFO schedulers for both static and dynamic slot configuration. They also introduced slot configuration for homogeneous and heterogeneous clusters. For the heterogeneous environment, H_TUMM slot assignment algorithm was implemented. Authors used work count, histogram rating, classification, inverted index, and grep jobs for experimental results.

SEARCH PROCESS

The manual search process is done for reviewing the conference and journal papers. This manual searching gives various papers related to dynamic slot configuration and scheduling concepts from 2008. The sequential and random search processes are done manually. Research references are collected from various sources like search engines, staff members, and web links. Search engines like Google, Bing provide papers randomly. Transaction papers like IEEE provide papers sequentially.

RESEARCH METHOD

This study is examined to be an evaluation over the dynamic slot configuration and scheduling techniques for MapReduce cluster. The questions are always given new innovative research ideas and clarity about research.

Research questions play a vital role for identifying concepts, and issues in the survey. The questions related to our study are given below.

A. Which is optimal slot configuration method either static or dynamic?

B. Why authors prefer FIFO schedulers for task assignment?

C. What is the reason to use different schedulers for slot assignment?

D. Why authors prefer single node Hadoop cluster for experimental results?

E. What is the Reason for using independent and dependent pools in slot allocation?

4.OBSERVATION

A. Which is optimal slot configuration method either static or dynamic?

The dynamic slot configuration is always optimal because the static slot configuration assigns the task to MapReduce slots before the cluster launch. So the number of idle slots may increase due to the completion of map slots, and also chances of occurring overloaded reduce slots. Surely it affects completion time of the task. Unlike static slot configuration, dynamic slot configuration allocates slots during task execution time. It reduces the number of idle slots and increases busy slots

B. Why authors prefer FIFO schedulers for task assignment?

The FIFO scheduler is the default Hadoop scheduler implemented in MapReduce applications. Some authors

still prefer FIFO scheduler for their research, especially Y. Yao et al. [15]. There are two Common reasons are to select default first in first out schedulers. Firstly, N numbers of jobs are waiting for acquiring resources. Secondly, all jobs can get resources without any starvation.

C. What is the reason to use different schedulers for slot assignment?
Schedulers are classified based on the performance metric, deadline aware, fairness metric, delay, resource aware, and fair4s scheduling. Each scheduler has configured with one of the metric specified above. Based on research preference different schedulers are used.

D. Why most people prefer single node Hadoop cluster for performance optimization?
Apache Hadoop installation comes with single node Hadoop cluster and multi-node Hadoop cluster. Mostly researcher configures single node cluster because it is easy to install with low cost and easy analysis of performance results. Hadoop multi-node cluster configuration needs more amount of hardware and network facilities. Multi node configuration is possible only to create a cloud environment. This is the reason for configuring single node cluster.

E. What is the Reason for using independent and dependent pools in slot allocation?
A task can be allocated with in the pool and across the pool. The fair scheduler allocates the same amount of resource to active tasks. Sometimes the task within the pool needs resource across the pool. This is the main reason for dividing pools into independent and dependent. The dependent pool slots can dynamically allocate across the pool. But independent pool slots only allocates within the pool dynamically.

SYSTEM MODEL AND DYNAMIC SLOT CONFIGURATION UNDER HETEROGENEOUS ENVIRONMENTS

Heterogeneous environments are fairly common in today's cluster systems. For example, system managers of a private data center could always scale up their data center by adding new physical machines. Therefore, physical machines with different models and different resource capacities can exist simultaneously in cloud Servers.

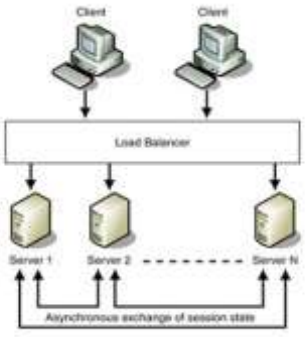


Figure:1. Architecture design the shade rectangles indicate our new/modified components in Hadoop

When deploying a Hadoop cluster **[Figure 1]** in such a heterogeneous environment, tasks from the same job may have different execution times when running on different nodes. In this case, a task's execution time highly depends on a particular node where that task is running. A job's map tasks may run faster on a node which has faster cpu per slot while its reduce tasks may experience shorter execution times on the other nodes that have more memory per slot. Estimating the remaining workloads and deciding the slot configuration in heterogeneous Hadoop cluster becomes more complex.

For example, consider a Hadoop job with 7 map tasks and a Hadoop cluster with two heterogeneous nodes such that node 1 is faster than node 2. Consider a cluster configured with 4 map slots in total, and one map task of that job takes 1 second and 2 seconds to finish on node 1 and node 2, respectively. We note that in this heterogeneous Hadoop cluster, various slot configurations will yield different performance (e.g., the execution time) of this job

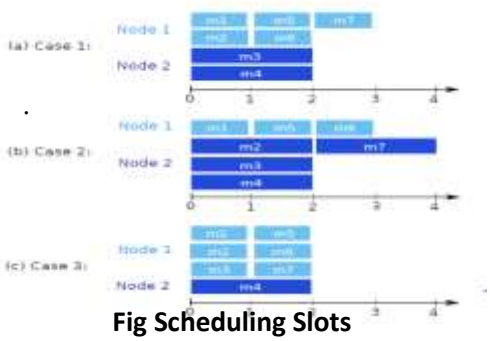


Fig Scheduling Slots

As illustrated in **[Figure 2]** case 1, the total execution time of the map phase takes slot on node 2. However, the map phase execution time can be improved to 3 seconds if we change the slot configures on these two nodes, i.e., 3 map slot on node 1 and 1 map slots on node 2. This situation indicates that it is harder to predict the time needed to finish the map phase or reduce phase in the heterogeneous environment, and evenly distribute the map (or reduce) slot assignments across the cluster will no longer work well. Which utilizes the overall workload information to set the slot assignments over the entire cluster does not work well any more when the nodes in the cluster become heterogenous. New version of TuMM, named H TuMM, which dynamically sets the slot configurations for each node in a heterogeneous Hadoop cluster in order to reduce the makespan of Hadoop jobs.

Algorithm Design: H TuMM H TuMM shares

International Journal
of Computer Sciences
ISSN Number: 4142 3453

the similar idea of TuMM, i.e., dynamically assign slots to map and reduce tasks to align the process of map and reduce phase based on the collected workload information. The key difference of H TuMM is to set the slot configurations for each node indivi0dually in a heterogeneous cluster, i.e., each of those nodes will have different slot assignment ratio between map and reduce tasks. To accomplish it, H TuMM collects the workload information on the entire cluster and on each individual node as well: when a map/reduce task is finished on node i, the workload collector updates

(1) The average execution time of map/reduce tasks, i.e., t_m/t_r;

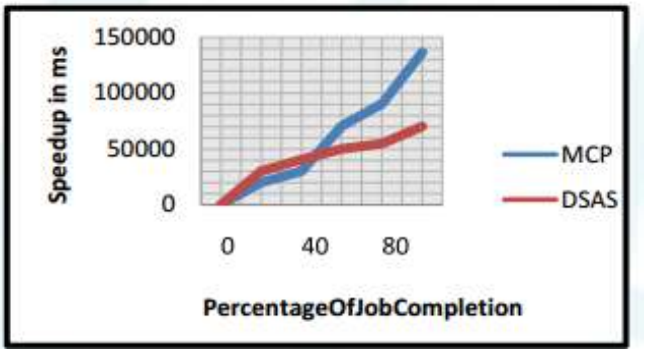
(2) The average execution of map/reduce tasks that ran on node i, i.e., t^i

Algorithm:Slot Assignment for Node :

0: **Input:** Average task execution time on n
the cluster, and the remaining task nu
running jobs;
0: **When** Node i has free slots and ask for
ment through the heartbeat message;
1: $s_m^i \leftarrow \lfloor S^i * \frac{\frac{t_m^i * n_m'}{t_m^i}}{\frac{t_m^i * n_m'}{t_m^i} + \frac{t_r^i * n_r'}{t_r^i}} \rfloor$;
2: $s_r^i \leftarrow \lfloor S^i * \frac{\frac{t_r^i * n_r'}{t_r^i}}{\frac{t_m^i * n_m'}{t_m^i} + \frac{t_r^i * n_r'}{t_r^i}} \rfloor$;
3: **if** $s_m^i + s_r^i \leq S^i$ **then**
4: **if** $\frac{t_m^i}{t_m} > \frac{t_r^i}{t_r}$ **then**
5: $s_r^i \leftarrow S^i - s_m^i$;
6: **else**
7: $s_m^i \leftarrow S^i - s_r^i$.
8: **if** $(s_m^i - rt_m^i) > (s_r^i - rt_r^i)$ **then**
9: assign a map task to node i;
10: **else**
11: assign a reduce task to node i;

5. Results

In This section we have shown the working of the proposed system. The data shows the total required time for the completion of task for proposed system in contrast with the existing system which is Maximum Cost Performance. In graph it also shows the specific time required for all the three techniques Dynamic
Hadoop Slot Allocation (DHSA), Speculative Execution Performance Balancing (SEPB) and Slot Pre-Scheduling. The graph 1 shows the time required to complete the tasks in MCP is higher as compared to DSAS. The performance of the MCP degrades as the time speeds up.



Graph Performance improvement of the system

6.CONCLUSION

Dynamic slot configuration is one of the important factors while processing a large data set with MapReduce paradigm. It optimizes the performance of MapReduce framework. Each job can be scheduled using any one of the scheduling policies by the job tracker. The task managers which are present in the task tracker allocate slots to jobs. From the examined paper, it is concluded to prefer a dynamic slot allocation strategy that includes active jobs workload estimation, optimal slot assignment, and scheduling policy.

REFERENCES

[1] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters", in Communications of the ACM, vol. 51, 2008.

[2] J. Polo, D. Carrera, Y. Becerra et al., "Performance-driven task co- scheduling for MapReduce environments", in NOMS'10, 2010.

[3] J. Wolf, D. Rajan, K. Hildrum, R. Khandekar, V. Kumar, S. Parekh,

K.-L. Wu, and A. Balmin, "Flex: A slot allocation scheduling optimizer for MapReduce workloads", in Middleware 2010, ser. Lecture Notes in Computer Science, I. Gupta and C. Mascolo, Eds. Springer Berlin / Heidelberg, vol. 6452. pp. 1, 2010.

[4] Apache Hadoop. Reference link: http://hadoop.apache.org.

[5] A. Verma, L. Cherkasova, and R. H. Campbell, "ARIA: Automatic Resource inference and allocation for MapReduce environments", in International Conference on Autonomic Computing, 2011.

[6] Apache Hadoop YARN (yet another resource negotiator) Reference

Link: https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop- yarn-site/YARN.html.

[7] B. Sharma, R. Prabhakar, S.-H. Lim et al., "Mrorchestrator: A fine-

grained resource orchestration framework for MapReduce clusters", in CLOUD'12, 2012.

[8] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth et al., "Apache Hadoop yarn: Yet another resource negotiator", in

Proceedings of the 4th annual Symposium on Cloud Computing.

ACM, 2013.

[9] Jiayin Wang, Yi Yao, Ying Mao, Bo Sheng, N. Mi, "FRESH: Fair and Efficient Slot Configuration and Scheduling for Hadoop Clusters", IEEE 7th International Conference on Cloud Computing,

DOI: 10.1109/CLOUD.2014.106. Anchorage, AK, pp 761, June

2014.

[10] Capacity scheduler Reference Link: https://hadoop.apache.org/docs/r1.2.1/capacity_scheduler.html

[11] S. Tang, B. Lee, and B. He," Dynamic MR: A Dynamic Slot Allocation Optimization Framework for MapReduce Clusters",

IEEE Transactions on Cloud Computing, vol. 2, issue. 3, September

2014.

[12] A. Bansal, A. Deshpande, P. Ghare, S. Dhikale, B. Bodkhe, " Healthcare Data Analysis using Dynamic Slot Allocation in Hadoop", International Journal of Recent Technology and Engineering Vol. 3 Issue 5, November 2014.

[13] A.U.Patil, T.I Bagban , A.P.Pande, "Recent Job Scheduling

Algorithms in Hadoop Cluster Environments: A Survey", International Journal of Advanced Research in Computer and Communication Engineering Vol. 4, no. 2, February 2015.

[14] Z. Li, Q. Zhang, M. F. Zhani, R. Boutaba, Y. Liu, Z. Gong et al., "DREAMS: Dynamic Resource Allocation for MapReduce with Data Skew", Integrated Network Management (IM), DOI 10.1109/INM.2015.7140272, 2015 IFIP/IEEE International

Symposium on May 2015.

[15] Y. Yao, J. Wang, B. Sheng, C. Tan and N. Mi, "Self-Adjusting Slot Configurations for Homogeneous and Heterogeneous Hadoop Clusters", IEEE Transactions on Cloud Computing, Vol. PP,

September 2015.