ISSN 2350-1022

International Journal of Recent Research in Mathematics Computer Science and Information Technology
Vol. 2, Issue 1, pp: (114-121), Month: April 2015 – September 2015, Available at: www.paperpublications.org

SURVEY ON PERFORMANCE OF HADOOP MAPREDUCE OPTIMIZATION METHODS

Mr. Marisiddanagouda. M[1], Mr. Raghu MT[2]

[1]PG Student, [2]Asst Prof, Dept. Of CS&E, MITE, Moodabidri, Mangalore, India

Abstract: **Hadoop is an open source software framework for storage and processing large scale of datasets on clusters of commodity hardware. Hadoop provides a reliable shared storage and analysis system, here storage provided by HDFS and analysis provided by MapReduce. MapReduce frameworks are foraying into the domain of high performance of computing with stringent non-functional requirements namely execution times and throughputs. MapReduce provides simple programming interfaces with two functions: map and reduce. The functions can be automatically executed in parallel on a cluster without requiring any intervention from the programmer. Moreover, MapReduce offers other benefits, including load balancing, high scalability, and fault tolerance. The challenge is that when we consider the data is dynamically and continuously produced, from different geographical locations. For dynamically generated data, an efficient algorithm is desired, for timely guiding the transfer of data into the cloud over time for geo-dispersed data sets, there is need to select the best data center to aggregate all data onto given that a MapReduce like framework is most efficient when data to be processed are all in one place, and not across data centers due to the enormous overhead of inter-data center data moving in the stage of shuffle and reduce. Recently, many researchers tend to implement and deploy data-intensive and/or computation-intensive algorithms on MapReduce parallel computing framework for high processing efficiency.**

Keywords: **Hadoop, HDFS, MapReduce, Optimization methods, MapReduce Performance.**

1. INTODUCTION

Hadoop is a open source software framework for storage and processing big data in a distributed fashion on large clusters of commodity hardware. Here Hadoop provides a reliable shared storage and analysis system.[1] Here storage provided by HDFS and analysis provided by MapReduce. MapReduce is a programming model for processing and generating large datasets on clusters of computers(Introduced by Google). Hadoop is an open source implementation of GFS and

Map Reduce. Hadoop MapReduce framework consists of a JobTracker running on master node and TaskTracker running on slave nodes. The HDFS manages storage on the cluster by breaking incoming files into pieces called "blocks," and storing each of the blocks redundantly on pool of servers. In the common case, HDFS stores three complete copies of each file by copying each piece to three different servers. HDFS data are stored in object form, and each object occupies about 150 byte. If there is a large number of these small files for storage, NameNode will request for a lot of space. That will severely restrict the scalability of the cluster. JobTracker may be over loaded since it is responsible for monitoring and dispatching simultaneously. Hadoop is similar to the database. It requires specialized optimization according to actual application needs. Number of experiments is on going to improve the performance of Hadoop. To be able to process large-scale datasets, the fundamental design of the standard Hadoop places more emphasis on high throughput of data than on job execution performance. This causes performance limitation when Hadoop MapReduce is used to execute short jobs that require quick responses. [2] Hadoop MapReduce ecosystem software market is predicted to grow at 60.2% compound annual growth rate between 2011 and 2016.[3] From small sized clusters to clusters with well over thousands of nodes, Hadoop technology is being used to perform myriad of functions- search optimizations, data mining, click stream analytics and machine learning to name a few. So we used some optimization methods to improve the performance of Hadoop MapReduce. Sometimes Hadoop does not perform well. [4] The main reason for this performance degradation is the interaction and heavy dependency across different MapReduce phases. This happens because the data placement and task execution are highly coupled in the MapReduce paradigm. [5]

Users usually expect short execution or quick response time from a short MapReduce job. To provide SQL like queries or analysis, some query systems are available, such as Google's Sawzall , Facebook's Hive and Yahoo!'s Pig. These systems execute users' requests by converting SQL-like queries to a series of MapReduce jobs which are usually short. Obviously, these systems are very sensitive to the execution time of underlying MapReduce jobs. Therefore, reducing the executing time of short jobs is very important to these types of applications. So for this an optimized version of Hadoop which is designed to reduce the time consumed in the execution process of a job. For improving the performance of MapReduce due to above reason, many performance optimization methods are introduced.

Map-Reduce have inherent limitations on its performance and efficiency. Therefore, many studies have endeavored to overcome the limitations of the Map- Reduce framework. In the replica selection of the input files for map tasks, the Map-Reduce framework does not take into account the distribution of the input data blocks in the distributed file system and the load of the computing nodes themselves, which leading to increase the amount of network data transfer and system load when running map tasks. Especially when the framework uses the FIFO job scheduling strategy to deal with a large number of small jobs, the performance of the framework will be very low.

2. WORKING OF HADOOP MAPREDUCE

Thanks to Google, Doug and Yahoo because it all started with Google, which in 2003 and 2004 released two academic papers describing Google technology: the Google File System (GFS) and MapReduce The two together provided a platform for processing data on a very large scale in a highly efficient manner. Doug started work on the implementations of these Google systems, and Hadoop was born, and became top-level project within the Apache open source foundation. At its core, Hadoop is an open source platform that provides implementations of both the MapReduce and GFS technologies and allows the processing of very large data sets across clusters of low-cost commodity hardware. In addition to often publicizing some of the largest Hadoop deployments in the world, it has contributed some of its own internally developed Hadoop improvements and extensions remains a major Hadoop contributor. The new Hadoop is nothing less than the apache foundations [1] [6] attempts to create a whole new general framework for the way big data can be stored, mined and processed and is a open source software framework for operating system of sorts for the data vertical used to store and process large volumes of data with the help of disparate computing system instead of few super computers. The hardware is cheap and the software is open-source leveling the field for all players and emerged as more of an operating system that manages resources secures data and performs other functions and applications written on top of it help process data and cross-pollinate data points. Hadoop is an apache top-level project Where several million pieces of content is a lot and cannot be easily be analyzed within any excel spreadsheet. A traditional ETL (Extract, Transform, Load) process extracts data from multiple sources, then cleanses, formats and loads it into a data warehouse for analysis when the source data sets are large, fast, and unstructured, traditional ETL can become the bottleneck because it is too complex to develop too expensive to operate and takes too long to execute. Various criticisms of Hadoop have resolved around its scaling limitations but the biggest constraint on scale has been its job scheduling in Hadoop are run as batch processes through a single daemon called JobTracker which creates a scalability and processing speed. Thus Apache Hadoop is an open source distributed software platform for storing and processing data written in java it runs on clusters of industry standards servers configured with direct attached storage using Hadoop can store petabytes of data reliably on tens of thousands of servers while scaling performance cost-effective by merely adding inexpensive nodes to the clusters because of its batch processing Hadoop should be deployed in situations such as index building, pattern recognitions, creating recommendation engines and sentiment analysis. All situations where data is collected in ever escalating sheer volumes with variable semantic contexts, stored in Hadoop, queried at later using MapReduce functions.

Common building blocks:

- Both HDFS and MapReduce exhibit important architectural principles
- Designed to run on clusters of commodity (that is, low-to-medium specification) servers

ISSN 2350-1022

International Journal of Recent Research in Mathematics Computer Science and Information Technology
Vol. 2, Issue 1, pp: (114-121), Month: April 2015 – September 2015, Available at: www.paperpublications.org

- Both scale their capacity by adding more servers (scale-out) with different mechanisms for identifying and working around failures
- Provide many of their services transparently, allowing the user to concentrate on the problem at hand
- Both have an architecture where a software cluster sits on the physical servers and controls all aspects of system execution

2.1 HADOOPDISTRIBUTED FILE SYSTEM (HDFS):

HDFS [7] the only building block in the Hadoop stack. Hadoop is a file system capable of storing bytes of data, if we take a file system view of the world then users want a common distributed file system and HDFS is a perfectly reasonable alternative. It has one main difference between others file system is "Write-Once-Read-Many" times' model support concurrency control, high fault-tolerance, and self-healing and high throughput access. Unique attribute of HDFS [6] is to locate pre-processing logic near the data rather than moving the data to the application space by Storing petabytes of web data, logs, and web snapshots runs on Commodity X86 servers, storage(SAN), Gigabit Ethernet LAN with Open source software that Keep track per-node costs down to afford more nodes computation in individual server.

HDFS consists of following daemons: HDFS master "NameNode", worker "DataNode", Secondary-NameNode.

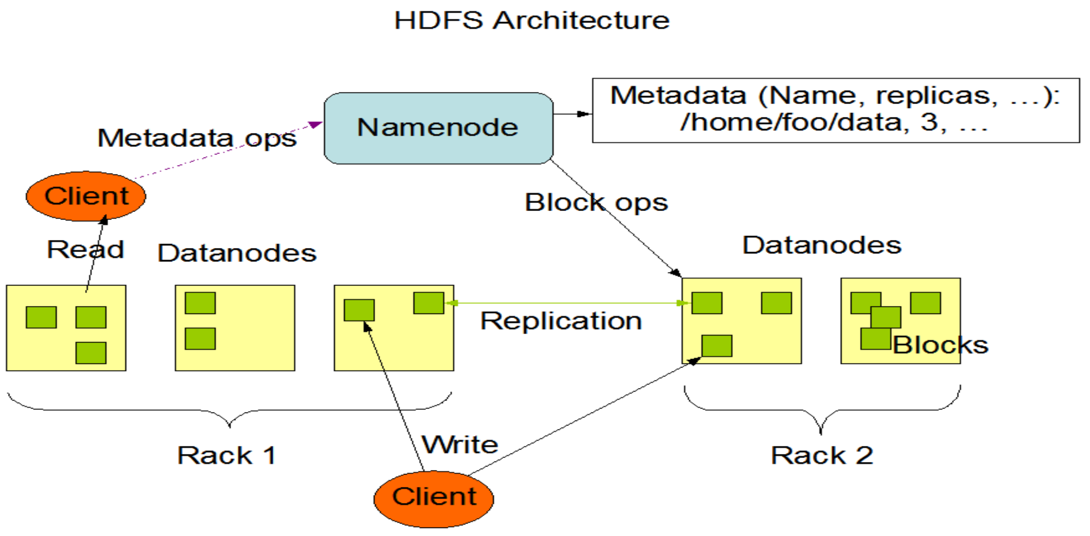


Figure1: HDFS Architecture.

Name Node-A master server manages the file system namespace and regulates access to files by clients using Meta-data in memory, the entire metadata is in main memory with Meta-data types, lists of blocks for each file, datanodes for each block and file attributes.

DataNode-One-per node in the cluster manages storage attached to the nodes that they run on, by block server stores data in the local file system in terms of (/ext3) format and Serves data and metadata to clients with block reports and sends existing blocks to the NameNode

Secondary Name Node-Not used as host, stand-by or mirror node. Backup NameNode periodically wakes up and processes check point and update the NameNode. And Directory is same as NameNode except it keeps previous checkpoint versions in addition to current. Used to restore failed Namenode.

2.2 MAPREDUCE:

Model for processing large amounts of data in parallel on commodity hardware and Lots of nodes Derived from functional programming known Map and reduce functions can be implemented in multiple languages Java, C++, Ruby, Python etc, imposing key-value input/output. Defines map and reduce functions map: (K1, V1) gives the list (K2, V2) and reduce: (K2, list (V2)) gives the list (K3, V3). Map function is applied to every input key-value pair. Map function generates intermediate key-value pairs, intermediate key-values are sorted and grouped by key, and Reduce is applied to sorted and grouped intermediate key-values. Reduce emits result key-values. MapReduce framework takes care ofdistributed processing and coordination in terms of Scheduling Jobs is broken down into smaller chunks called tasks These tasks are scheduled with respect to Localization with Data and Framework strives to place tasks on the nodes that host the segment of data to be processed by that specific task and where code is moved to the data is located with Error Handling without failures expectations so that tasks are automatically re-tried on other machines with Data Synchronization, Shuffle and Sort re-arranges and moves data between machines and Input and output are coordinated by the framework.

Map Function

```
map(inut_record){
emit(k1,v1)
emit(k2,v2)
}
```

Reduce Function

```
reduce (key,values){
while(values.has_next){
aggregate=merge(values.next)}
collect(key,aggregate)
} }
```

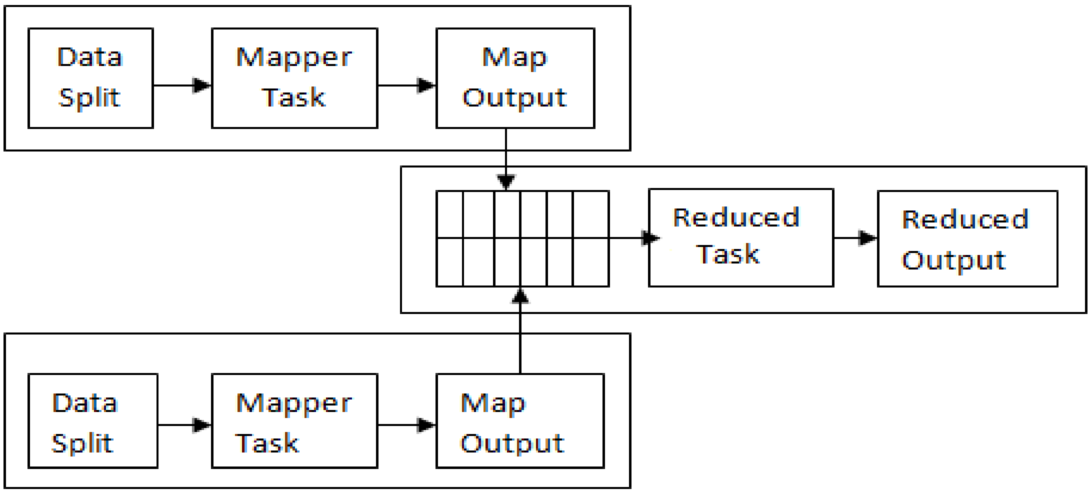


Figure 2: MapReduce Data Flow

3. HADOOP MAPREDUCE OPTIMIZATION METHODS

Here we need to improve the performance of hadoop mapreduce so for this reason we concentrated on some optimization methods shown below:

A. From the perspective of application for optimization[8]: Since MapReduce parses the data file iteratively and line by line, programming application programs with high efficiency under this circumstance is a way to optimization. Performance of MapReduce can be improved form the following aspects:

- Avoid unnecessary Reduce tasks.
- Pull in external file.

- Add a Combiner to Job.
- Reuse Writable type.
- Use StringBuffer instead of String to track program bottlenecks.

B. Hadoop system parameter optimization: There are over 190 configuration parameters in current Hadoop system. How to adjust these parameters so that jobs can run as fast as possible is also a kind of optimization idea.

Hadoop system parameters optimization can start from the following three aspects:

- Parameters of Linux file system.
- General parameters of Hadoop.
- Hadoop jobs parameters.

C. Hadoop job scheduling algorithm optimization: The fact that Hadoop configuration based on cluster hardware information and the number of nodes can greatly improve the performance of Hadoop cluster has been proved. However, this method just optimizes the performance statically. It cannot modify the configuration files, load them or put them into force dynamically during the running time. Optimizing the job scheduling algorithm can solve this problem well. The scheduler is a pluggable module in Hadoop, and users can design their own dispatchers according to their actual application requirements.

Here are three task dispatchers.

1. The default dispatcher: This dispatcher adopts FIFO algorithm, which is simple and clear, and the burden of Jobtracker is not so heavy.

2. Dispatcher with computational capability: This kind of dispatcher supports multiple queues. Each queue has a certain amount of resources and uses FIFO scheduling policy. Jobs are scheduled in accordance with job priority and the order of submitted time.

3. Fair share scheduling algorithms: This solution is proposed by Facebook. The design philosophy is to ensure that all jobs can obtain the amount of resources as equal as possible. When there is only one task running in the system, it will monopolize all the resources of the cluster. When there is more than one task, there will be TaskTracker being released and assigned to the newly submitted job to ensure all the tasks can obtain the same computing resources roughly.

D. Data Transfer Bottlenecks: Big challenge is to how to minimize the cost of data transmission for cloud user. Map-Reduce-Merge [9] is a new model that adds a Merge phase after Reduce phase that combines two reduced outputs from two different MapReduce jobs into one, Map-Join-Reduce adds Join stage before Reduce stage. T. Nykiel proposed MRShare[10] is a sharing framework that transforms a batch of queries into a new batch that can be executed more efficiently by merging jobs into groups. Further it evaluates each group as a single query. Data skew is also an important factor that affects data transfer cost. In order to overcome this, a method that divides a MapReduce job into two phases was proposed: sampling MapReduce job and expected MapReduce job was proposed. The first phase is to sample the input data, gather the inherent distribution on keys' frequencies and then make a partition scheme. In the second phase, expected MapReduce job to group the intermediate keys quickly applies partition scheme to every mapper.

E. Effect by increasing number of Mappers and reducers slots:

In the multi node cluster sizing the number of Mappers and reducers slots correctly is very important to get the maximum performance of a MapReduce job. The optimum CPU utilization occurs when every machine in the cluster has something to do at any given time when a job is executed. "*mapred .tasktracker.map.tasks.maximum*" and "*mapred. tasktracker. Reduce .tasks.maximum*" are parameters to improve the number of slots, but this must be done smartly in accordance with our cluster resources such as CPU (with/without the hyper threading factor) and available memory. Otherwise, the overall cluster performance will downgrade.

It is not necessary to set the mappers or reducers count to the same value, because it depends on CPU utilization per task and how the job is done by each mappers and reducers to get the final result.

We can distribute CPU resources 50 percent for mappers and 50 percent for reducers or we can use any distributions such as two-third of the CPU resources for mappers and one-third for reducers. Totally our aim is to utilize cluster resources optimally. We considered number of mappers and reducers slots are same so,

F. Different Block Sizes:

Vary the block sizes of the data that is used by the HDFS file system for data distribution. Block sizes have the following effects on the performance metrics. Higher the block sizes it is easier for the HDFS to manage the information in the Namenode and lesser is the communication to the Namenode. Furthermore, when multiple set of data blocks for different applications are placed on a data node, accesses to the I/O (through I/O scheduling) by different applications will have an impact on the completion times. On the other hand, increasing the block size reduces the parallelism that can be exploited across the clusters. When the cluster size is large, this will have a huge impact on the overall execution time.

Here we surveyed above methods to improve the performance of Hadoop MapReduce and below we showed some experimental results, how Hadoop will performs by increasing the number of Mappers and Reducers and increasing the Blocksize and tested on some benchmarks like WordCount.

4. EXPERIMENTAL SETUP

Processor	**Intel Pentium® CPU G2020 @ 2.90GHz*2**
OS	**Ubuntu 12.04 lts 32-bit**
Memory	**4GB**
Harddisk	**500GB**
The native Hadoop	**Hadoop-1.0.3**

5. RESULTS

The below table and graph display the input and output values considered for execution of WordCount program using Map Reduce framework.

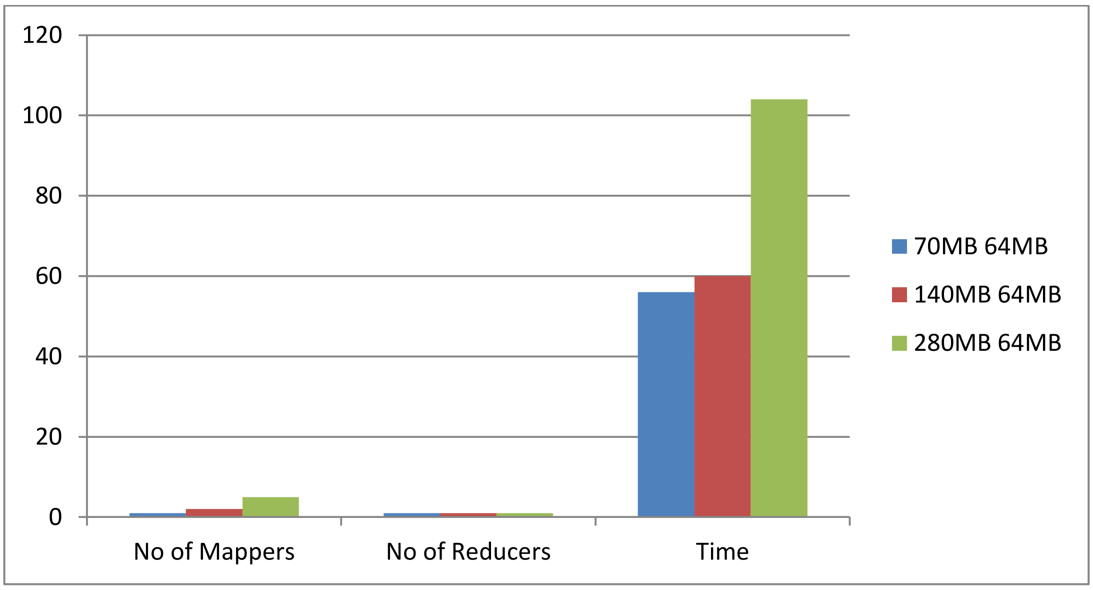
We are checking how much time taken to execute the jobs without changing any parameters in traditional Hadoop and also we checked the time after changing the parameters the slight variation is happened that can be showed below.

Table1: Traditional Hadoop

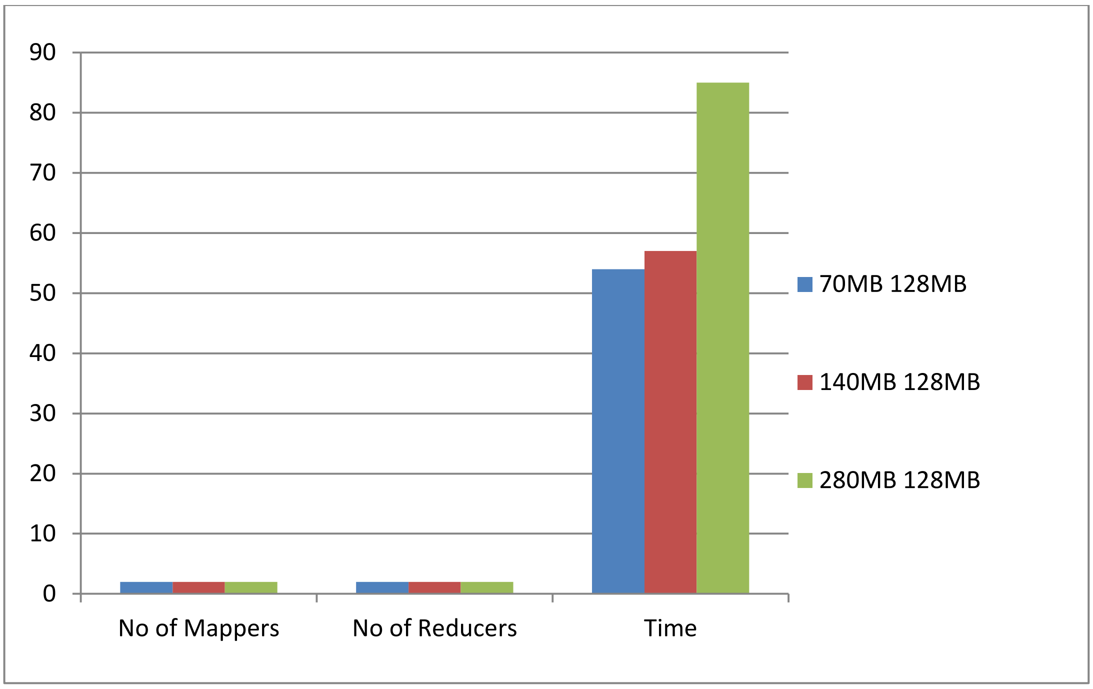
File size	Block size	No of Mappers	No of Reducers	CPU Time
70MB	**64MB**	**1**	**1**	**36600**
140MB	**64MB**	**2**	**1**	**75690**
280MB	**64MB**	**5**	**1**	**153140**

Table2: Traditonal Hadoop

Filesize	Blocksize	No of Mappers	No of Reducers	CPU Time
70MB	**128MB**	**2**	**2**	**35300**
140MB	**128MB**	**2**	**2**	**67280**
280MB	**128MB**	**2**	**2**	**141340**



Graph1:Traditonal Hadoop



Graph2: Modified some parameters in Hadoop

6. CONCLUSION AND FUTURE WORK

This survey paper gives a brief idea about performance of Hadoop MapReduce methods. Hadoop achieves more and more attention by the academia industry. And its application is more and more widespread. Though Hadoop shows good performance in dealing with large datasets concurrently, there are still some shortcomings. This paper gives some ideas or suggestions of Hadoop MapReduce optimization methods.

So our future work is that, still Hadoop mainly concentrated on high throughput of data than on job execution performance. This causes the performance limitation when we use Hadoop MapReduce to excute jobs, so for this reason we are checking how to improve those things using some optimization methods after that we are going to check on some variety of benchmark applications and datasets for further improvements.

ISSN 2350-1022

International Journal of Recent Research in Mathematics Computer Science and Information Technology
Vol. 2, Issue 1, pp: (114-121), Month: April 2015 – September 2015, Available at: www.paperpublications.org

REFERENCES

[1] Apache Hadoop. http://hadoop.apache.org.

[2] Jinshuang Yan, Xiaoliang Yang, Rong Gu, Chunfeng Yuan, and Yihua Huang," Performance Optimization for Short MapReduce Job Execution in Hadoop", 2012 Second International Conference on Cloud and Green Computing

[3] IDC Releases First Worldwide Hadoop-MapReduce Ecosystem Software Forecast, Strong Growth Will Continue to Accelerate as Talent and Tools Develop - prUS23471212," [Online]. Available:http://www.idc.com/ getdoc.jsp?containerId=prUS23471212. [Accessed 07 October 2012].

[4] Benjamin Heintz, Chenyu Wang, Abhishek Chandra, and Jon Weissman,"Cross-Phase Optimization in MapReduce ", 2013 IEEE International Conference on Cloud Engineering.

[5] XiaohongZhang, GuoweiWang, ZijingYang, YangDing," A Two-phase Execution Engine of Reduce Tasks In Hadoop MapReduce ", 2012 International Conference on Systems and Informatics (ICSAI 2012).

[6] Hadoop Distributed File System http://hadoop.apache.org/hdfs.

[7] Sanjay Ghemawat, Howard Gobioff, and Shun-TakLeung.The Google file system. In 19th Symposium on Operating Systems Principles, pages 29–43, Lake George, New York, 2003.

[8] Xin Daxin, Liu Fei. Research on optimization techniques for Hadoop cluster performance [J]. Computer Knowledge and Technology, 2011,8(7):5484~5486.

[9] H. Yang, A. Dasdan, R. Hsiao, and D. Parker, "Map-reduce-merge: simplified relational data processing on large clusters," in Proceedings of the 2007 ACM SIGMOD international conference on Management of data. ACM, 2007, pp. 1029–1040.

[10] T. Nykiel, M. Potamias, C. Mishra, G. Kollios, and N. Koudas, "Mrshare: Sharing across multiple queries in mapreduce," Proceedings of the VLDB Endowment, vol. 3, no. 1-2, pp. 494–505, 2010.

Author's Profile:

Mr. Marisiddanagouda M completed the bachelor's degree in Information Science and Engineering from SIET, Tumkur, and Pursuing M.Tech in Computer Network Engineering at MITE, Mangalore under VTU, Belgaum.

Mr. Raghu MT Assistant professor MITE, Mangalore. Completed his M.Tech in digital communication networking having 2 years of academic experience and his areas of interest are Image processing , Cloud Computing , Big Data and Hadoop , Distributed Systems.