iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 4 **Issue:** VIII **Month of publication:** August 2016

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

Volume 4 Issue VIII, August 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Enhanced Scheduling Algorithm for Avoiding Delay in MapReduce

Nadhiya.H[1], Nilavunesan.D[2], Raju.S[3], Anitha.R[4], Rajalakshmi.S[5]
[1,3,4,5] *Department of Computer Science and Engineering, Sri Venkateswara College of Engineering, Sriperambudur, Tamil Nadu - 602117, India.*
[2] *Department of Applied Science and Technology, Anna University, Chennai - 600025, India*

Abstract -- MapReduce is regarded as an adequate programming model for large scale data-intensive applications. When to start the reduce tasks is one of the main problems to elevate performance in the MapReduce. The performance of a MapReduce scheduling algorithm will be influenced seriously, when the output of map tasks becomes large. The aim is to formulate the causes for the waste of system slot resources, through analysis for the current MapReduce scheduling mechanism, which results in the reduce tasks waiting around and the proposes SARS (Self Adaptive Reduce Scheduling) for reduce tasks start times in Hadoop platform, it can determine the start time point of each reduce task dynamically according to each job context, including the completion time of the task and the size of map output. On comparing with the other algorithms SARS algorithm will decrease the job completion time, reduce completion time and system average time.
Keywords: MapReduce, Hadoop, SARS, Algorithm, Reduce tasks.

I. INTRODUCTION

MapReduce is an amazing model for distributed computing, introduced Google in 2004. It is an important programming model for computing in distributed and parallel, due to its easy of usage scalability and generality. Hadoop is an open source implementation version because it has been used in industry around the world [1]. The rapid growth of the internet has led to vast amount of information available online.

A. Big Data

Data growth challenges and opportunities as being three dimensional, they are high volume (amount of data), velocity (speed of data in and out),and variety (range of data types and sources). Big data is an all-encompassing term for any collection of data sets so large and complex that it becomes difficult to process them using traditional data processing applications. The challenges include analysis, capture, search sharing, storage, transfer, and visualization and privacy violations.

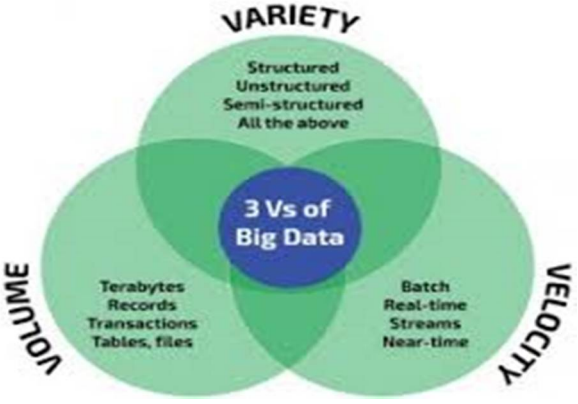


Fig 1. Vs of Big data

B. Hadoop Distributed File System (HDFS)

Hadoop is Java based file system that provides scalable and reliable data storage that is designed to span large clusters of commodity servers. To understand how it is possible to scale a Hadoop cluster to hundreds and thousands of nodes. Data in Hadoop cluster is broken down into smaller pieces or blocks and distributed throughout the cluster. In this way, the map and reduce function can be executed on smaller subsets of large data sets and this provides the scalability that is needed for bigdata processing .

C. Yet Another Resource Negotiator (YARN)

YARN is a next generation framework for hadoop data processing extending mapreduce capabilities by supporting non mapreduce workload associated with other program models. It is one of the main features in the second generation Hadoop 2.x version of the Apache software foundation's open source distributed processing framework. Originally described by Apache as a redesigned resource manager, YARN is now characterized large scale operating system for big data applications.

Resource Manager, which has the ultimate authority that arbitrates resources among all the applications in the cluster, it coordinates the allocation of compute resources on the cluster. Scheduler is main component of the scheduler.

Fig 2 : Yet Another Resource Negotiator

D. Related work

In [3] MapReduce overlapping using MPI, which is an adapted structure of the MapReduce programming model for fast intensive data processing. This implementation is based on running the map and the reduce functions concurrently in parallel by exchanging partial intermediate data between them in a pipeline fashion using MPI. At the same time, it maintains the usability and the simplicity of MapReduce. Ex Hadoop platform in cloud computing, When the map tasks output become large the performance of a MapReduce scheduling of reduce tasks will lead to untimely scheduling of a MapReduce scheduling algorithm influenced seriously. Especially, When multiple tasks are running simultaneously, inappropriate scheduling of reduce tasks will lead to untimely Scheduling the other jobs in the system. The main disadvantage of this technique stumbling block of the Hadoop popularization. The independent batch scheduling in CG as a bi-objective minimization problem with makespan and energy consumption as the scheduling criteria. Here use the Dynamic Voltage Scaling (DVS) methodology for scaling and possible reduction of cumulative power energy utilized by the system resources. The major issue for this method is incomplete, imprecise, fragmentary and overloading[4].

A Hierarchical MapReduce framework that gathers computation resources from different clusters and runs MapReduce jobs across them. The applications implemented in this framework adopt the Map-Reduce-Global Reduce model where computations are expressed as three functions: Map, Reduce, and Global Reduce. Two scheduling algorithms are introduced: Compute Capacity Aware Scheduling for our framework. Compute-intensive jobs and Data Location Aware Scheduling for data-intensive jobs. The traditional MapReduce scheduling result says the Reduce task should begin when all the map tasks are completed. The output of map tasks should be read and written to the reduce tasks in the copy process. Through the analysis of the slot resource usage in the reduce process, this illustrates that data transfer will result in slot idle and delay [5].

The Dynamic Priority (DP) parallel task scheduler for Hadoop. It allows user to control their allocated capacity by adjusting their spending over time. This simple mechanism allows the scheduler to make more efficient decisions about which jobs and users to prioritize and gives users the tool to optimize and customize their allocations to fit the importance and requirements of their jobs. Dynamic proportional scheduler provides more job sharing and prioritization capability in scheduling. The main disadvantage of these techniques increasing the share of cluster resources and differentiation in service [6].However, there is a conflict between fairness in scheduling and data locality (placing tasks on nodes that contain their input data). To address the conflict between locality and fairness, those propose a simple algorithm called delay scheduling: when the job that should be scheduled next according to fairness cannot launch a local task, it waits for a small amount of time, letting other jobs launch tasks instead. Here find that delay scheduling achieves nearly optimal data locality in a variety of workloads and can increase throughput by up to 2 X while preserving fairness. In addition, the simplicity of delay scheduling makes it applicable under a wide variety of scheduling policies beyond fair sharing. The main drawback of this paper is this conflict between data locality and fairness.[7]

II. METHODOLOGY

A. Scheduling techniques in yarn

Scheduler is in charge of allocating resources. The resource Container incorporates elements such as memory, cup, disk, network etc. Scheduler just has the resource allocation function, has no responsible for job status monitoring. And the scheduler is pluggable, can be replaced by other scheduler plug-in. There are many scheduling techniques in resource manager.

B. FIFO scheduler

FIFO scheduler is a non-adaptive scheduling algorithm. It schedules jobs based on their priorities in first come first serve. The main advantage of this Scheduling algorithm is cost of entire cluster scheduling process is less and simple to implement and efficient. The disadvantage of the FIFO is designed only for single type jobs and poor response times for short jobs compared to large jobs.

C. Fair scheduler

Fair scheduler is adaptive scheduling algorithm. It does equal distribution of compute resources among the user jobs in the system. It is less complex, Work well when both small and large clusters. It can provide fast response time for small jobs mixed with larger but, it doesn't consider job weight of each node.

D. Capacity scheduler

Capacity scheduler is adaptive scheduling algorithm. It maximizes the resources utilization and throughput in multi-tenant cluster environment. The main advantage of the capacity scheduler ensure guaranteed access with the potential reuse capacity and priorities jobs within queues over the large cluster. The drawback of this algorithm is more complex
This paper mainly focused on the reduce scheduling problem, which refers tothestarting timeof the reduce tasks

E. Runtime analysis of capacity scheduler in mapreduce

The existing Mapreduce program frameworks work the jobs as a whole process. Here, The difference between mat and reduce tasks are not considered. Since map and reduce task execution time are not related, it is not accurate to compute the average task execution time by taking map and reduce tasks.
An input split is a chunk of the input that is processed by a single map. Each map processes a single split. Each split is divided into records, and the map processes each record a key-value pair. When the map function starts producing output, it is not simply written to disk. The process is more involved, and takes advantage of buffering writes in memory and doing some presorting for efficiency reasons

F. MapTask

When the contents of the buffer reaches a certain threshold size a background thread will start to spill the contents to disk. Map outputs will continue to be written to the buffer while the spill takes place, but if the buffer fills up during this time, the map will block until the spill is complete. Spills are written in round-robin fashion to the directories specified by the mapreduce property, in a job-specific subdirectory. Before it writes to disk, the thread first divides the data into partitions corresponding to the reducers that they will ultimately be sent to. Within each partition, the background thread performs an in-memory sort by key, and if there is a combiner function, it is run on the output of the sort.
Running the combiner function makes for a more compact map output, so there is less data to write to local disk and to transfer to the reducer. Each time the memory buffer reaches the spill threshold, a new spill file is created, so after the map task has written its last output record there could be several spill files. Before the task is finished, the spill files are merged into a single partitioned and sorted output file.
the maximum number of streams to merge at once; the default is 10.If there are at least three spill files then the combiner is run again before the output file is written. Combiners may be run repeatedly over the input without affecting the final result. To compress the map output as it is written to disk, makes it faster to write to disk, saves disk space, and reduces the amount of data to transfer to the reduce

G. Reduce Task

The map output file is sitting on the local disk of the machine that ran the map task. The reduce task needs the map output for its

Volume 4 Issue VIII, August 2016
ISSN: 2321-9653

particular partition from several map tasks across the cluster

H. Copy phase of reduce

The map tasks may finish at different times, so the reduce task starts copying their outputs as soon as each completes. The reduce task has a small number of copier threads so that it can fetch map outputs in parallel. The map outputs are copied to reduce task JVM's memory otherwise, they are copied to disk. When the in-memory buffer reaches a threshold size or reaches a threshold number of map outputs it is merged and spilled to disk. If a combiner is specified it will be run during the merge to reduce the amount of data written to disk. The copies accumulate on disk, a background thread merges them into larger, sorted files. This saves some time merging later on.

Any map outputs that were compressed have to be decompressed in memory in order to perform a merge on them. When all the map outputs have been copied, the reduce task moves into the sort phase which merges the map outputs, maintaining their output.

I. Sort phase

This is done in rounds. For example, if there were 50 map outputs, and the merge factor was 10, then there would be 5 rounds. Each round would merge 10 files into one, so at the end there would be five intermediate files. These five files into a single sorted file, the merge saves a trip to disk by directly feeding the reduce function. This final merge can come from a mixture of in-memory and on-disk segments.

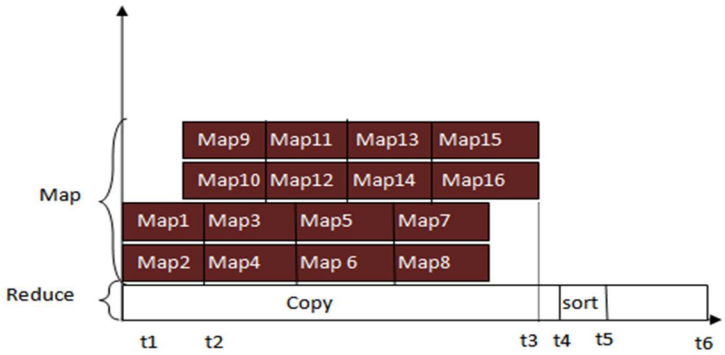


Fig 3: Analysis of capacity scheduling process

During the reduce phase, the reduce function is invoked for each key in the sorted output. The output of this phase is written directly to the output file system, typically HDFS. In the case of HDFS, since the Tasktracker node is also running a Datanode, the first block replica will be written to the local disk

The default output format, Text Output Format, writes records as lines of text. Its keys and values may be of any type.

In fig1.3 shows that during the t1 to t3, t1 is start time of the map1, map2 and the reduce task. The major work of the reduce task is to copy the output from map1 to map 14.The output of map 15 and map 16 will be copied by the reduce task from t3 to t4. The duration from t4 to t5 so called the sort stage. This ranks the intermediate results according to the key values .The reduce function is at the time t5 ,which continues from t5 to t6.because during t1 to t3,in this copy phase ,the reduce task only copies the output data intermittently, once map task is completed, for most time it is always waiting around. We hope to make the copy operation completed at a concentrated duration, which can decrease the waiting time of the reduce tasks [2] .

J. Proposed System

Hadoop employs the greedy strategy to schedule the reduce tasks, to schedule the reduce task fastest. Some reduce tasks will always take up the system resources without actually performing operations in long time. The reduce task start time is determined by this advanced algorithm SARS. In this method, the start times of the reduce tasks are delayed for certain duration to lessen the utilization of system resources. The SARS algorithm schedules the tasks at a special moment, when some map tasks are finished but not all. By this means, how to select an optimal time point to start the reduce scheduling is the key problem of the algorithm. Distinctly, minimize the system delay and maximize the resource utilization.

This paper makes the following contributions

1) Illustrate the reasons of system slot resource wastages, which results in the reduce tasks waiting around

Volume 4 Issue VIII, August 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

2) The development model details the start time of the tasks dynamically according to the each job context. Including the job completion time and the size of the each map output

K. Self adaptive scheduling algorithm

The Self Adaptive Scheduling Algorithm works by delaying the reduce processes. The reduce tasks are scheduled when part but not all of the map tasks are finished. For a special key value if we assume that the map slots and map tasks in the current system, the completion time and the size of the output data of each map task are denoted. Then the amount of map tasks data can be calculated as,

Amount of map task data = sum of output of each map task ---- (1)

Start time of reduce task = (Average completion timeof map task –
Data transmission which produced by map task) ---- (2)

In order to predict the time required to transmit the data, this defines the speed of data transmission from the map tasks to the reduce tasks as transmission speed in the cluster environment, and the number of concurrent copy threads with reduce tasks is denoted as copythread. Therefore, we use the average completion time of map tasks to minus data transmission time which produced by the map task.

Average completion time of map task = (start time of the map task + 1 / mapslots × sum of completion time of the each map task) ---- (3)

The average completion time of the map task is calculated using the start time of the map task and sum of completion time of the each map task. Start time of the map task is initial time of the first map task enter in to the map task execution.

TaskTrackers has a fixed number of slots for map tasks and for reduces tasks which are set independently, the scheduler will fits the empty map task slots before reduce task slots. Map task chosen depends on the data locality and TaskTracker's network location. After, complete the map tasks to calculate the sum of completion time of the map task.

$$\text{Data transmission time which produced by the map task} = \frac{\text{(Amount of map task data)}}{\text{(Transmission speed * copy thread)}} \quad \text{----- (4)}$$

The reduce task will be started before the map tasks are finished. The completed output of a map task is given as the input to the reduce task. While the transmission speed produced by the map task is to be calculating using transmission speed and copy thread. Copythread is defined as the number of threads which fetch the data blocks from the map tasks and the transmission speed is known as the data transmission.

Fig 4 shows, the reduce task at t2 ,which can be calculated using Eq (3) and make sure these tasks can be finished before t6,then during t1 to t2 ,the slots can be used by any other reduce tasks. But we let the copy operation start at t3, because the output of all map tasks should be copied from t3.the delay will be produce in this case. As shown in fig 2,the copy phase starts at t2, which just collect the output of the map tasks intermittently, contrast the waiting time is decreased obviously in fig 3, in which case the copy operations are start at t2.

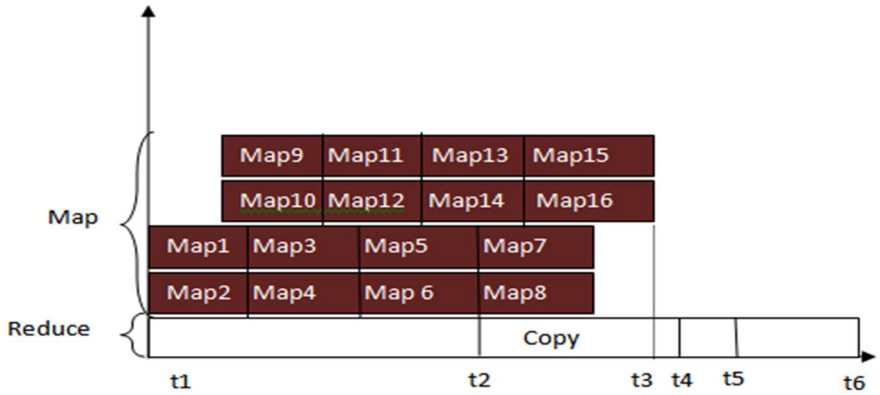


Fig 4: Analysis of SARS scheduler

III. CONCLUSION

In this paper three scheduler have been analyzed such as FIFO scheduler, Fair scheduler and capacity scheduler. Hadoop 2.7.0 build the capacity scheduler so that here mainly focused on the capacity scheduler. While run the Program map and copy process start almost at the same time, causes the massive waste of slot resource. The goal of the improved algorithm to decrease the time of the reduce tasks in the map reduce framework. in this paper, the performance of this algorithm is estimated from the job completion time and reduce completion time.

REFERENCES

[1] Lizhe Wang, Jie Tao, Rajiv Ranja, Holger Marten, Achim Sterit, Jingying Chen ,Dan Chen, G-Hadoop: MapReduce across distributed data centers for data intensive computing, Future Gener. Comput. Syst. 29 (3) (2013) 739–750.

[2] Zhuo Tang, Lingang Jiang, Junqing Zhou, Kenli Li, Keqin Li, A self adaptive scheduling algorithm for reduce start time, future Gener.Comput. Syst. 43-44 (2015) 51-60.

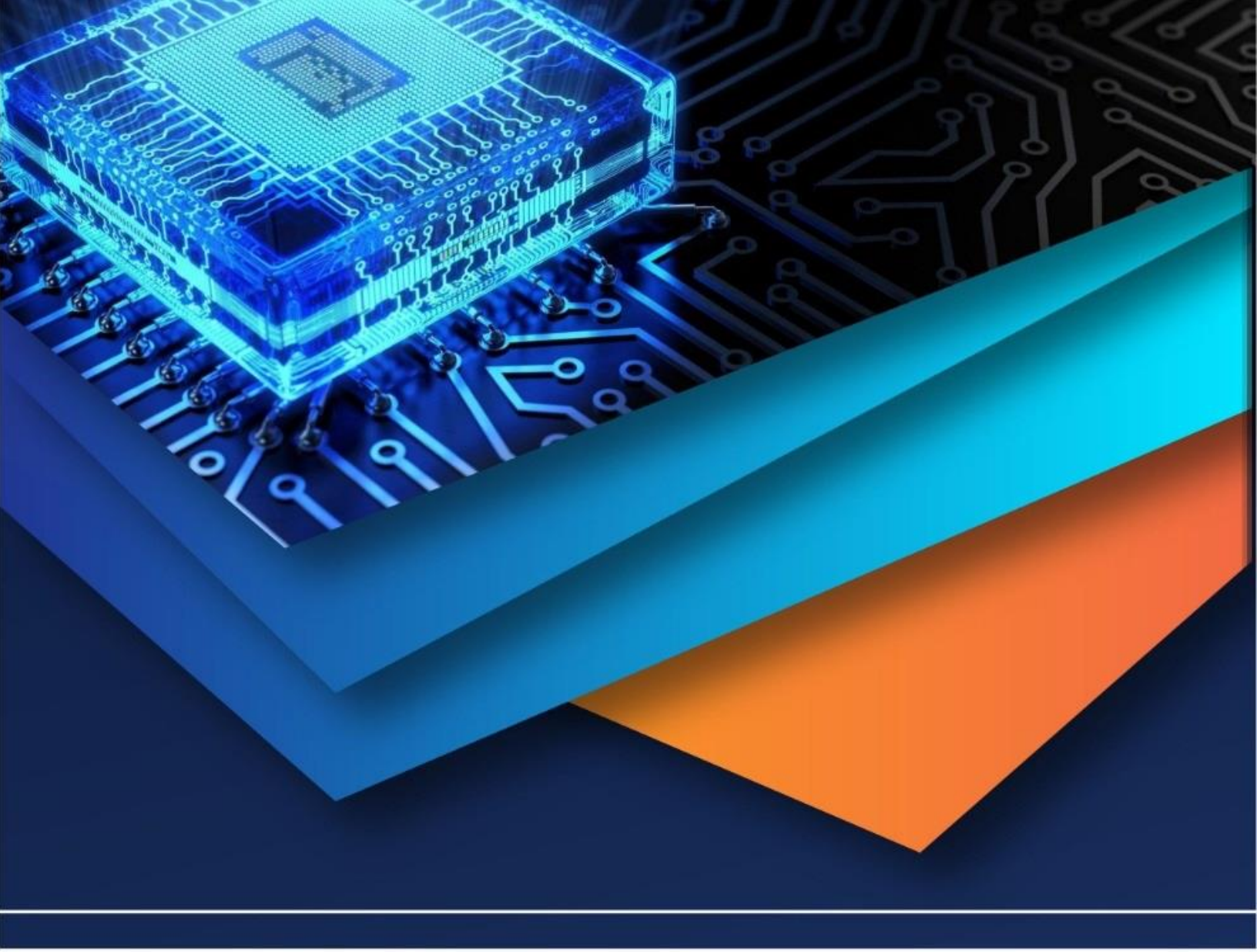
[3] Hisham Mohamed, Stephane Marchand-Mailet,MRO_MPI:Mapreduce overlapping using MPI an optimized data exchange policy,parallel Comput. 39 (12) (2013) 851-856.

[4] T.Sandholm, K. Lai, Dynamic share scheduling in hadoop, in: proceedings of the 15thWorkshop on job scheduling strategies for parallel processing, 2010, pp.265-278.

[5] M.Zaharia ,D.Borthakur, J.S. Sharma, K.Elmeleegy,S.Shenker,I.Stoica, Delay Scheduling:a simple technique for achieving locality and fairness in cluster scheduling,in: proceeding of the 5th Duropean Conference on Computer systems,2010,pp.265-278

[6] Yuan Luo, Beth plale, Hierarchical Mapreduce programming model and scheduling algorithms, in: cluster computing and the grid, IEEE international symposium on,pp.769-774.s

[7] Joanna kolodziej,Samee Ullah KHAN,Lizie Wang, Aleksander Byrski, Nasro min-Allah, sajjad Madani, Hierarchial geniticbased grid scheduling with energy optimization,cluster comput.16(3) (2013) 591-609.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY