Scientific Journal of Impact Factor (SJIF): 4.72

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 4, Issue 5, May -2017

Efficient Job Execution for Map Reduce Using Phase-Level Scheduling Algorithm.

Nisha Shinde[1], Trupti Patil[2], Prajkta Shinde[3], Himani Mavchi[4]

[1,2,3,4] *Student Dept. of Computer Engineering., AISSMS's Institute Of information Technology, Pune, Maharashtra, India*

Abstract – ***Technology's*** *role in society today has a major impact on our overall sense of living and that is why in the 21st century, it is offered as a subject. The fast and improved speed of computer systems are making humans life easier and giving him new opportunity to create an impossible. To improve the processing speed of systems different technologies are now used like distributed system, parallel computing. The map reduce which is used in parallel computing is one of the popular data model for high speed computation in computation technology. The Existing map reduce focuses on scheduling at the task-level. But unfortunately, the task-level scheduling leads to inefficient job schedules with low resource utilization and long job execution time.*

In this concept we divide the tasks into unequal parts called as phases and apply phase-level scheduling to these phases and achieve efficient resource usage. In this paper, we present a Scheduler, a Phase and Resource Information-aware Scheduler for MapReduce clusters that performs resource-aware scheduling at the level of task phases. Specifically, we show that for most MapReduce applications, the run-time task resource consumption can vary significantly from phase to phase. Therefore, by considering the resource demand at the phase level, it is possible for the scheduler to achieve higher degrees of parallelism while avoiding resource contention.

Keywords- *Map Reduce; Scheduling; Cloud Computing; Hadoop; Resource Allocation;*

I. INTRODUCTION

A good way to understand how a processor works is to imagine that your job is to tell a thousand people how to do their jobs. The faster you can do that, the faster everyone works. Computer processors operate in a similar fashion. A processor provides the instructions that multiple applications and processes need to perform their jobs. As day by day technology is improving and the use of technology is improving the more processing speed is needed the high speed computation is a need of today's Critical computational tasks. And the todays fastest processor chips are failed to achieve that speed because of the limitations.

Solution to that the new processing techniques are developed such as distributed system where multiple computer synchronize their work and perform it.

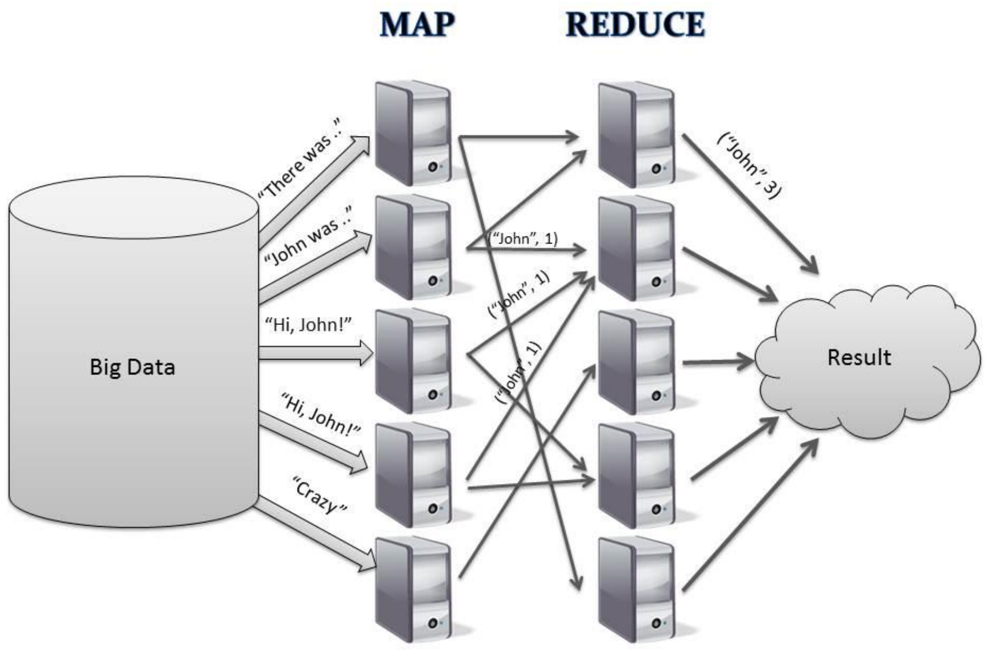
Also parallel system where task are performed parallel. The Hadoop is a framework that supports the processing and storage of extremely large data sets in a distributed computing environment. It is very popular for distributed system. For processing Hadoop uses map reduce is a programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster. A Map Reduce program is composed of a Map() procedure (method) that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name) and a Reduce() method that performs a summary operation (such as counting the number of students in each queue, yielding name frequencies). The "Map Reduce System" (also called "infrastructure" or "framework") orchestrates the processing by marshalling the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing for redundancy and fault tolerance.

A central component to a MapReduce system is its job scheduler. Its role is to create a schedule of Map and Reduce tasks, spanning one or more jobs, that minimizes job completion time and maximizes resource utilization. A schedule with too many concurrently running tasks on a single machine will result in heavy resource contention and long job completion time. MapReduce can significantly reduce the running time of data-intensive jobs. However, despite recent efforts toward designing resource-efficient MapReduce schedulers, existing solutions that focus on scheduling at the task-level still offer sub-optimal job performance. In this paper we introduced a fine-grained, phase and resource-aware Map Reduce Scheduler that divides tasks into phases, where each phase has a constant resource usage profile, and performs scheduling at the phase level. System can provide accurate resource information that can be used by the scheduler so that it can take effective scheduling decisions and reduce the job execution time. The phase-level scheduling achieves higher resource utilization when compared to task level schedulers. The improvement in the processing speed is quite significant by the use of phase level scheduler.

II. RELATED WORK

Apache Hadoop is an open-source software framework for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common and should be automatically handled by the framework The core of Apache Hadoop consists of a storage part, known as Hadoop Distributed File System (HDFS), and a processing part called Map Reduce. Hadoop splits files into large blocks and distributes them across nodes in a cluster. To process data, Hadoop transfers packaged code for nodes to process in parallel based on the data that needs to be processed.

There are two aspects that differentiate scheduling in Map Reduce from traditional cluster scheduling. The first aspect is the need for data locality, i.e., placing tasks on nodes that contain their input data. Locality is crucial for performance because the network bisection bandwidth in a large cluster is much lower than the aggregate bandwidth of the disks in the machines. Traditional cluster schedulers that give each user a fixed set of machines, like Torque, significantly degrade performance, because files in Hadoop are distributed across all nodes as in GFS. Grid schedulers like Condor support locality constraints, but only at the level of geographic sites, not of machines, because they run CPU-intensive applications rather than data-intensive workloads like Map Reduce.



.

Figure 1. MapReduce structure.

Even with a granular fair scheduler, we found that locality suffered in two situations: concurrent jobs and small jobs. We address this problem through a technique called delay scheduling that can double throughput.

Map Reduce uses scheduler to schedule the task. There are Several scheduler schemes are there for scheduling like FIFO scheduler ,fair scheduler ,capacity scheduler ,resource adaptive scheduler and Dominant resource fair scheduler. This job schedulers are integrated with the job tracker. The job tracker pulls the job from system and give it to the scheduler for scheduling. The simplest scheduling algorithm is a FIFO. It works on the FIFO manner of queue (first come first serve basis). The Fair scheduler is the default. Resources are shared evenly across pools and each user has its own pool by default. You can configure custom pools and guaranteed minimum access to pools to prevent starvation. This scheduler supports pre-emption. The goal of the fair scheduler is to give all users equitable access between pools to cluster resources. Each pool has configurable guaranteed capacity in slots. Each Pool is equal to the number of jobs. Jobs are placed in flat pools. The default is 1 pool per user.

Hadoop also supports the idea of provisioning virtual clusters from within larger physical clusters, called Hadoop on Demand (HOD). The HOD approach uses the Torque resource manager for node allocation based on the needs of the virtual cluster. With allocated nodes, the HOD system automatically prepares configuration files, and then initializes the system based on the nodes within the virtual cluster. Once initialized, the HOD virtual cluster can be used in a relatively independent way. HOD is also adaptive in that it can shrink when the workload changes. HOD automatically de-allocates nodes from the virtual cluster after it detects no running jobs for a given time period.

So we can say that MapReduce work is to schedule the task in different levels. In a MapReduce technique, it is a collection of jobs and can be scheduled concurrently on multiple machines, resulting in reduction in job running time. Any companies refer MapReduce to process large volume of data. But they refer at task level to perform these data. Initially the task level performs two phases one is Map phase and other is reduce phase. In map phase, it takes data blocks Hadoop distributed file system and it maps, merge the data and stored in the multiple files. Then the second, reducer phase will fetch data from mapper output and shuffle, sort the data in a serialized manner. At the task level, performance and effectiveness become very important factor in day to day life. Task level scheduler is not capable to manage the available resources effectively. In the end it won't be able to give optimize performance.

The Hadoop was designed for large batch jobs. Hadoop uses Map Reduce to complete this batch job. Map Reduce uses scheduler to schedule the task. There are Several scheduler schemes are there for scheduling like FIFO scheduler ,fair scheduler ,capacity scheduler ,resource adaptive scheduler and Dominant resource fair scheduler. This job schedulers are integrated with the job tracker. The job tracker pulls the job from system and give it to the scheduler for scheduling. The simplest scheduling algorithm is a FIFO. It works on the FIFO manner of queue (first come first serve basis). The Fair scheduler is the default. Resources are shared evenly across pools and each user has its own pool by default. You can configure custom pools and guaranteed minimum access to pools to prevent starvation. This scheduler supports pre-emption. The goal of the fair scheduler is to give all users equitable access between pools to cluster resources. Each pool has configurable guaranteed capacity in slots. Each Pool is equal to the number of jobs. Jobs are placed in flat pools. The default is 1 pool per user.

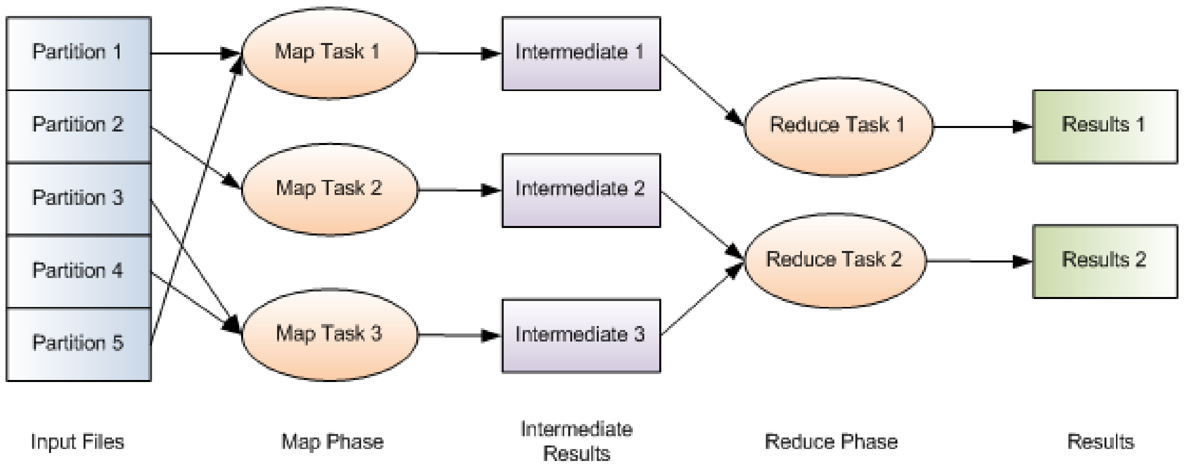


Figure 2. Typical MapReduce Job.

Hadoop also ships with the Capacity scheduler; here resources are shared across queues. You may configure hierarchical queues to reflect organizations and their weighted access to resources. You can also configure soft and hard capacity limits to users within a queue. Queues have ACLs to prevent rogues from accessing the queue. This scheduler supports resource-based scheduling and job priorities. The goal of the capacity scheduler is to give all queues access to cluster resources. Shares are assigned to queues as percentages of total cluster resources.

Although not a scheduler per se, Hadoop also supports the idea of provisioning virtual clusters from within larger physical clusters, called Hadoop on Demand (HOD). The HOD approach uses the Torque resource manager for node allocation based on the needs of the virtual cluster. With allocated nodes, the HOD system automatically prepares configuration files, and then initializes the system based on the nodes within the virtual cluster. Once initialized, the HOD virtual cluster can be used in a relatively independent way. HOD is also adaptive in that it can shrink when the workload changes. HOD automatically de-allocates nodes from the virtual cluster after it detects no running jobs for a given time period. This behavior permits the most efficient use of the overall physical cluster assets.

So basically MapReduce work is to schedule the task in different levels. In a MapReduce technique, it is a collection of jobs and can be scheduled concurrently on multiple machines, resulting in reduction in job running time. Any companies refer MapReduce to process large volume of data. But they refer at task level to perform these data. Initially the task level performs two phases one is Maper phase and other is reducer phase. In mapper phase, it takes data blocks Hadoop distributed file system and it maps, merge the data and stored in the multiple files. Then the second, reducer phase will fetch data from mapper output and shuffle, sort the data in a serialized manner. At the task level, performance and effectiveness become very important factor in day to day life. Task level scheduler is not capable to manage the available resources effectively. In the end it won't be able to give optimize performance.

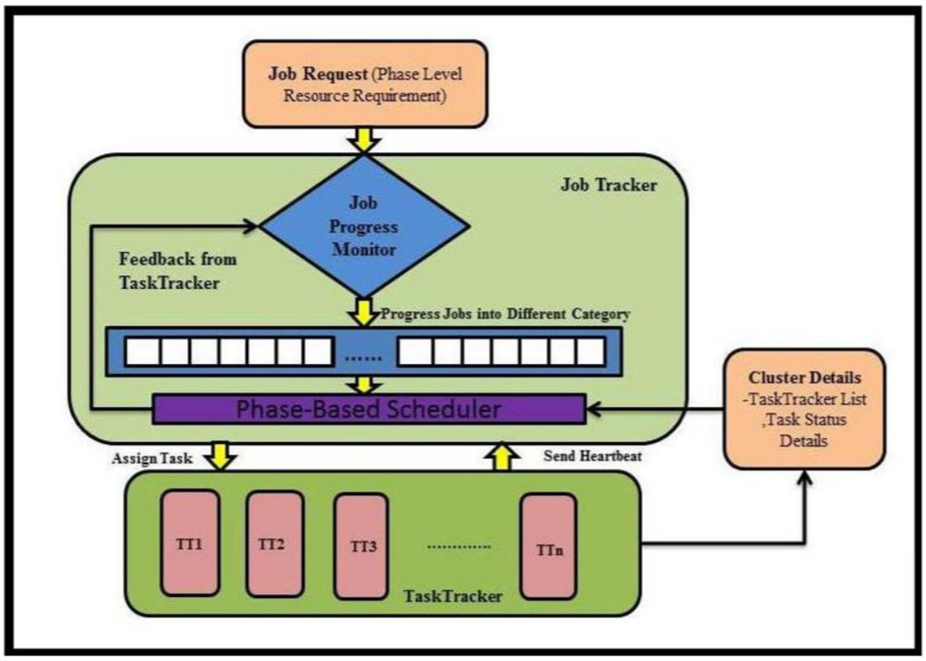

Job Request (Phase Level Resource Requirement)
Job Tracker
Job Progress Monitor
Feedback from TaskTracker
Progress Jobs into Different Category
Phase-Based Scheduler
Cluster Details -TaskTracker List ,Task Status Details
Assign Task
Send Heartbeat
TT1
TT2
TT3
TTn
TaskTracker

Figure 3. Phase Based Scheduler Architecture

In this system we use the phase level scheduling algorithm instead of task level scheduling algorithm. The job requester send s the request to the resource manager in which the job progress monitor will check the total space and number of job request. Progress monitor can sends job to the phase level scheduler it can divide the job in to different jobs. These jobs can sends to the node manager if space is available then allocate the job.

If the space is not available the node manager again send the job to the phase base scheduler and it allocate the virtual space. The phase base scheduling algorithm is to schedule the phase on every map and reduce task .the phase based scheduling is to schedule the phase based on the resource utilization of that node the phase based scheduling algorithm is a set off phases that scheduled on machine.

III. PROPOSED SYSTEM

In the proposed system we present the fine-grained resource-aware scheduler, which performs scheduling at phase-level. It allows the job owners to specify the phase-level requirements. The architecture states that there are majorly three components: a scheduler called the phase-based scheduler which is located at the master node, local node managers that coordinate phase transitions with the scheduler and a Resource manager that is indeed used resource information at the phase-level. For scheduling whenever a task needs to be scheduled, the scheduler replies with a message with the task scheduling request. Then the node manager assigns that task. Each time a task finishes executing a phase it notifies and asks permission of the node manager to go to the next phase. The node manager than forwards the permission request to the scheduler through the regular message .If sufficient resources are available the scheduler decides and informs its decision to the local node manager whether it can proceed or wait the execution of the next phase. Finally, if the task is given permission to execute the next phase, the node manager grants the task to continue its duty. Once the task is completed, the task status is forwarded to the node manager and then forwarded again to the scheduler.

We suggest that it would be more efficient if we make the scheduler to work at the phase-level instead of the task-level. The reason is because the task demands a lot of requirements during its lifetime. In this concept we divide the tasks into unequal parts called as phases and apply phase-level scheduling to these phases and achieve efficient resource usage

3.1 Working Principle

The system functions can be described as follows:

When a User wants to perform the parallel computation the system must request for a job from user to perform parallel computation. And be sure that job is fit for parallel execution. So node manager has to send a message for request to schedule the job and whenever scheduler is ready it has to response to that request message.

Scheduling of job is the main functionality of this system, scheduling is the method by which work specified by some means is assigned to resources that complete the work. Here we first allocate the job to the scheduler and then divide this job into smaller task. After that This Smaller task are assign to node manager.

After that we have to Assign the resources to the job in the effective manner so that overall performance should be improve. At the end system should calculate the time and space to measure the performance improvement of system over traditional system.

3.2 Phase level scheduling algorithm.

The main steps in scheduling algorithm are as follows

1) The system is modeled as S = {s, e, X, Y, Jf, Jp, Jn} Where, s is the initial state and the user will input the data (D).
2) X = Set of inputs in the system X = {D, k, tm} where D = {d1, d2, d3,.,.,.,dn} and Y = Set of outputsY
3) Jp= Job performance and Jf=job fairness.
4) Allocation of job to multiple task .Reduce the delay of particular task.
5) If isPendingMapTask () is true then Calculate number of map task and Running Map Task.
6) Else if isShuffleTask () is true then Calculate number of Shuffle task Running Map Task.
7) Else Calculate no of reduce task and Running Reduce Task
8) At the end Calculate Jf, Jp.
9) End.

IV. System Architecture

In the Phase Level Scheduling algorithm we will going the design the Scheduler which will work on Hadoop MapReduce which will improve the performance of parallel computation by better Resource utilization. As you can see in diagram our system mainly has three module as scheduler, Resource manager and node manager.

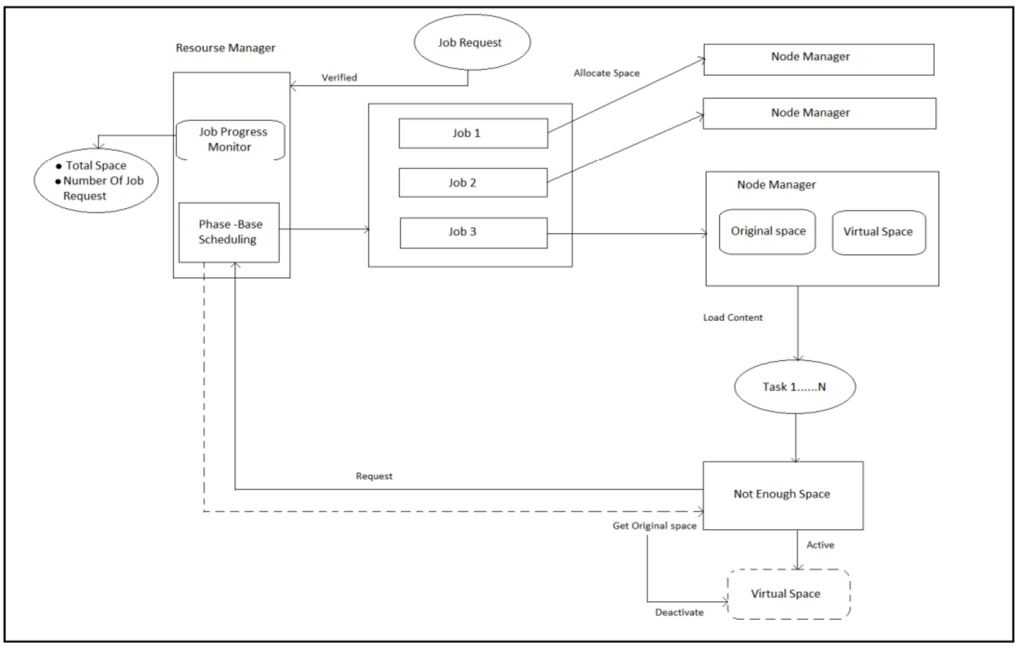


Figure 4. Phase Level architecture diagram.

Also the other modules are users, job requester and tasks. The basic work flow of the system is User first create the resources and jobs. After creating the job requester Request for a job. Then the Job is allocated to the scheduler. Scheduler Divide job into smaller tasks and Assign each task to the node scheduler and also Schedule the Resources. Then the node scheduler Performs Computation and calculate the time and space. While proceeding in a phase level, phase-based scheduler send message to node manager. Upon receiving heartbeat message from node manager reporting resource availability on node, the scheduler must select which phase should be scheduled on node. For each job J consists of two types of tasks: map task M and reduce task R. We define the Utility function with machine n and assigning phase I as shown in equation. In utilization, PRISM is able to achieve shorter results and is able to achieve shorter job running time while maintaining high resource utilization for large workloads containing a mixture of jobs, which are same cluster.

V.OUTPUT

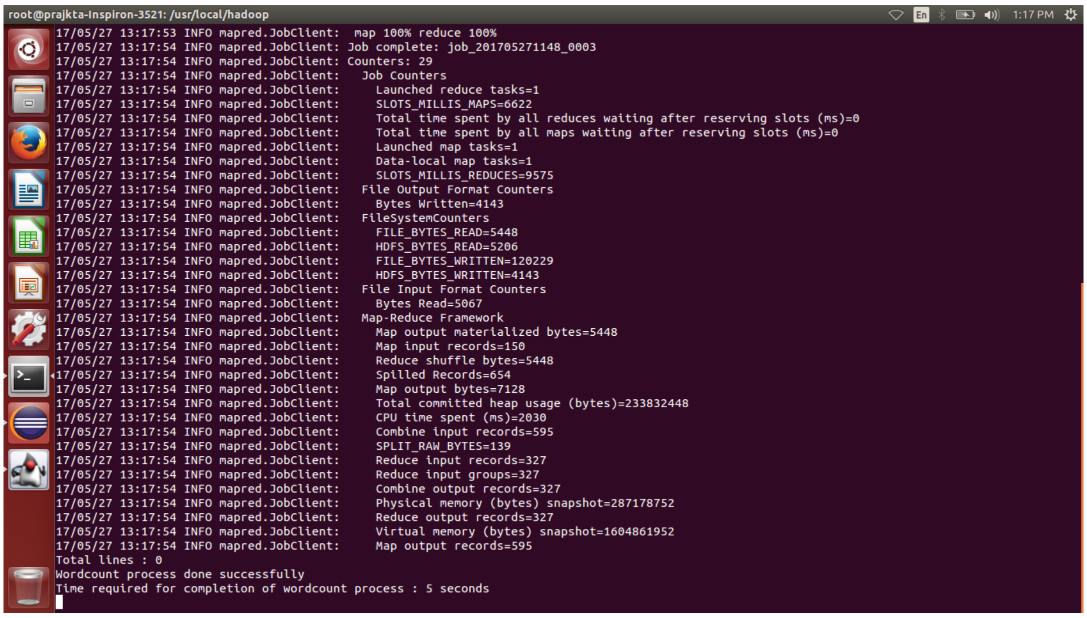
5.1 Scheduling Reduce Tasks

Fig1 shows While Map Tasks are small tasks that can run independently in parallel, Reduce Tasks are long-running tasks that contain copy/shuffle and reduce phases. In most existing schedulers, Reduce Tasks are not preemptive, i.e., a Reduce

Task will not release the occupied slot until it's reduce phase completes. It is feasible to introduce Reduce Task preemptions in engineering realization.

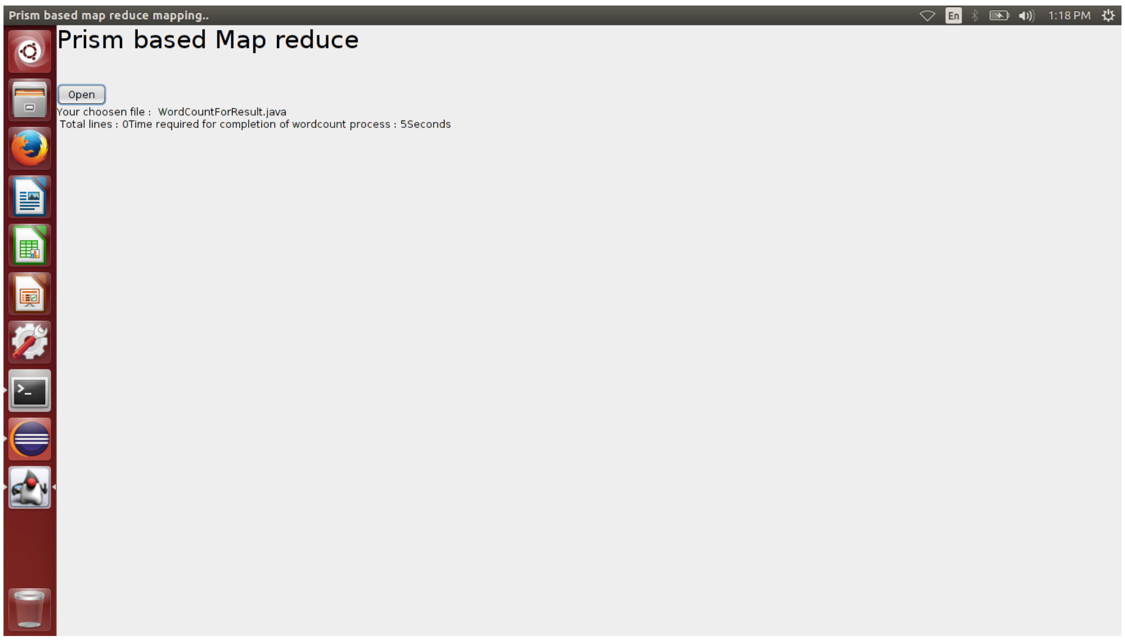
- **Job tracker**

In Hadoop system the Job Tracker will take care of scheduling a new map task. If a new task is started after the failure of the reduced task, then all the input data that was sent to the failed reduced attempt must be again sent by the new reduce instance. The MapReduce process is divided as the map task and reduce task. There are large number of the commodity machines available in the Hadoop.



```
root@prajkta-Inspiron-3521: /usr/local/hadoop
17/05/27 13:17:53 INFO mapred.JobClient:  map 100% reduce 100%
17/05/27 13:17:54 INFO mapred.JobClient: Job complete: job_201705271148_0003
17/05/27 13:17:54 INFO mapred.JobClient: Counters: 29
17/05/27 13:17:54 INFO mapred.JobClient:   Job Counters
17/05/27 13:17:54 INFO mapred.JobClient:     Launched reduce tasks=1
17/05/27 13:17:54 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=6622
17/05/27 13:17:54 INFO mapred.JobClient:     Total time spent by all reduces waiting after reserving slots (ms)=0
17/05/27 13:17:54 INFO mapred.JobClient:     Total time spent by all maps waiting after reserving slots (ms)=0
17/05/27 13:17:54 INFO mapred.JobClient:     Launched map tasks=1
17/05/27 13:17:54 INFO mapred.JobClient:     Data-local map tasks=1
17/05/27 13:17:54 INFO mapred.JobClient:     SLOTS_MILLIS_REDUCES=9575
17/05/27 13:17:54 INFO mapred.JobClient:   File Output Format Counters
17/05/27 13:17:54 INFO mapred.JobClient:     Bytes Written=4143
17/05/27 13:17:54 INFO mapred.JobClient:   FileSystemCounters
17/05/27 13:17:54 INFO mapred.JobClient:     FILE_BYTES_READ=5448
17/05/27 13:17:54 INFO mapred.JobClient:     HDFS_BYTES_READ=5206
17/05/27 13:17:54 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=120229
17/05/27 13:17:54 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=4143
17/05/27 13:17:54 INFO mapred.JobClient:   File Input Format Counters
17/05/27 13:17:54 INFO mapred.JobClient:     Bytes Read=5067
17/05/27 13:17:54 INFO mapred.JobClient:   Map-Reduce Framework
17/05/27 13:17:54 INFO mapred.JobClient:     Map output materialized bytes=5448
17/05/27 13:17:54 INFO mapred.JobClient:     Map input records=150
17/05/27 13:17:54 INFO mapred.JobClient:     Reduce shuffle bytes=5448
17/05/27 13:17:54 INFO mapred.JobClient:     Spilled Records=654
17/05/27 13:17:54 INFO mapred.JobClient:     Map output bytes=7128
17/05/27 13:17:54 INFO mapred.JobClient:     Total committed heap usage (bytes)=233832448
17/05/27 13:17:54 INFO mapred.JobClient:     CPU time spent (ms)=2030
17/05/27 13:17:54 INFO mapred.JobClient:     Combine input records=595
17/05/27 13:17:54 INFO mapred.JobClient:     SPLIT_RAW_BYTES=139
17/05/27 13:17:54 INFO mapred.JobClient:     Reduce input records=327
17/05/27 13:17:54 INFO mapred.JobClient:     Reduce input groups=327
17/05/27 13:17:54 INFO mapred.JobClient:     Combine output records=327
17/05/27 13:17:54 INFO mapred.JobClient:     Physical memory (bytes) snapshot=287178752
17/05/27 13:17:54 INFO mapred.JobClient:     Reduce output records=327
17/05/27 13:17:54 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=1604861952
17/05/27 13:17:54 INFO mapred.JobClient:     Map output records=595
Total lines : 0
Wordcount process done successfully
Time required for completion of wordcount process : 5 seconds
```

Figure 5.Task Tracker Process



Prism based map reduce mapping..

Prism based Map reduce

Your choosen file : WordCountForResult.java
Total lines : 0Time required for completion of wordcount process : 5Seconds

Figure 6.Execution time of file.

VI. CONCLUSION

MapReduce is a popular data model for data related high speed computing. However, despite recent efforts toward designing resource-efficient MapReduce schedulers, existing work mainly focuses on designing task-level schedulers, they work fine but Still Provides Sub-Optimal Job Performance Because Varying Resource Requirements of tasks. Also task level scheduler does not able to utilize the resources effectively and it effects the performance of the system.

It is oblivious to the fact that the execution of each task can be divided into phases with drastically different resource consumption characteristics. That's when phase level scheduler comes into picture. The phase level scheduler which works on phase instead of task. Phase-Level consist of how run-time resources can be used and how it varies over the long life time. In short they utilize resources effectively. And it in the result performance gets improved.

REFERENCES

[1] Qi Zhang, Mohamed Faten Zhani, Yuke Yang and Raouf Boutaba, "PRISM: Fine-Grained Resource-Aware Scheduling for MapReduce", IEEE Transaction on cloud computing, VOL. 3, NO. 2, Page Number, APRIL/JUNE 2015**.**

[2] Suryakant S. Bhalke, "Allocation of Phase-Based Scheduler for MapReduce Job Scheduling", IJARCCE, ISO 3297:2007 Certified Vol. 5, July 2016.

[3] Dhanya Sudhakaran and Shini Renjith," Phase Based Resource Aware Scheduler with Job Profiling of Map Reduce." International Journal of Latest Trend in Engineering and Technology, July 2015.

[4] Ms.Savitri.D.H and Narayana H.M, "PRISM: allocation of resources in phase-level using map-reduce in Hadoop", International Journal of Research in Science & Engineering e-ISSN: 2394-8299 Volume: 1 Special Issue: 2, May 2014.

[5] Mr.Surya Bahadur and S. Ramachandra, "Resource-Diversity Tolerant: Resource Allocation in the Cloud Infrastructure Services" , IOSR journal of computer engineering (IOSR-JCE)-OCT 2015.

[6] G.Hemalatha and S.Shibia Malar," PROCESSOR LEVEL RESOURCE-AWARE SCHEDULING FOR MAPREDUCE IN HADOOP", (IJETCSE) ISSN: 0976-1353 Volume 22 Issue 1 – MAY 2016.