Dynamic Capacity Scheduling in Hadoop

Shivani Thakur
Research Scholar
Dept.of Computer Science & Engg
Chandigarh University,
Ghauran, Mohali.

Rupinder Singh
Assistant Professor
Dept. of Computer Science & Engg.
Chandigarh University,
Ghauran, Mohali.

Sugandha Sharma
Assistant Professor
Dept. of Computer Science & Engg.
Chandigarh University,
Ghauran, Mohali.

ABSTRACT
Hadoop mapreduce could be a powerful data processing technique giant for large information analysis on distributed artifact between clusters like clouds. In this paper we proposed the improved capacity scheduler to improve the existing scheduler issues that help the scheduler to execute the task in less time. We introduced pipeline and queue management in our proposed work for improving the performance of hadoop. Our experimental result show that our strategies can result in about 29 to 50 % decrease in average response time.

Keywords
Mapreduce, HDFS, Hadoop, Scheduler.

1. INTRODUCTION
In today's world billions of individual's are exploit net and with usage of net at this scale quantity of knowledge that flows over the net is whopping and with cloud computing this data is increasing even at a better rate [1]. This quantity of information that is transferred daily from one location to a different over the web has created computing a awfully sophisticated task. To manage such vast quantity of information is in itself a difficult task. Complicated algorithms have to be compelled to be written to accomplish this task however Hadoop provides U.S. with an honest choice to reduce the quality of managing this information.

Managed by Apache package Foundation [2], Hadoop is open supply package that implements MapReduce algorithmic rule employed by Google to manage knowledge. MapReduce framework operates on pairs, within which I/O is each, a collection of pairs. Hadoop, nowadays is one in every of the foremost trending technology that has gained attention of the many huge enterprises and people. Hadoop solves drawback of managing an excessive amount of knowledge that is mixture of structured and complicated knowledge that doesn't match into tables. Hadoop may be utilized in varied markets like Finance, on-line Retail, Portfolio analysis, Risk Analysis and lots of additional.

Hadoop runs on sizable amount of machines that don't share memory or disks. A Hadoop cluster consists of servers that accommodate 2 to eight CPUs, every hardware consisting one massive disk connected to 4-16 massive processors. Hadoop spreads out silos of information to modify it and everyone these massive processors add parallel to answer sophisticated queries .Processors operating in parallel utilize hardware resources additional effectively. Hadoop provides 3 Job hardwares that are: Job Queue Task Scheduler, truthful hardware and capability Task hardware that are represented later within the paper. Execution time of a job has continually been a priority in Hadoop framework collectively slow job will have an effect on the execution time .Hadoop is run in 3 completely different modes: 1st is standalone mode that is default mode of Hadoop and HDFS isn\'t employed in this mode. Second is Pseudo-Distributed mode that could be a single node cluster and uses HDFS as filing system. Third mode is Fully-Distributed cluster that has multiple nodes and knowledge is employed and processed among these nodes.

In this paper, we proposed a scheduler which helps to reduce the execution time of a job .This paper is divided into 6 section as preliminary study in section 2, implementation of our proposed work in section 3, our proposed work result in section 4, result in section 5 and conclusion and future work is in section 6.

2. PRELIMINARY STUDY

2.1 Mapreduce
The MapReduce performs 2 distinct phases in Hadoop. Initial is Map part that takes set of knowledge and method it in parallel to return a set of knowledge as intermediate result and another is reduce part that takes intermediate results and reduces it in smaller set of knowledge. Overall MapReduce condenses great amount of information in helpful data. It divides a question into multiple steps.

MapReduce has emerged from a stimulating paper by Google in 2004 to a business commonplace. It are often wont to filter documents by tags, count range of words, computing, giant scale PDF generation, to method geographical information and plenty of additional tasks. It must not be used once latency must be certain.

2.2 Hadoop
Hadoop[3] is an open supply Java primarily based framework to store and method massive amounts of knowledge. It permits distributed process of knowledge that is gift over clusters mistreatment purposeful programming model. The storage element of Hadoop is termed HDFS and process element of Hadoop is termed MapReduce.

Hadoop Distributed filing system (HDFS) is java based mostly filing system that provides reliable and sturdy information storage. HDFS consists of NameNodes that manage data and DataNodes wherever file information is keep. Data is painted on NameNodes as in nodes that record data like permissions, access times etc. NameNodes send directions to DataNodes that embody commands in keeping with [3].

MapReduce is that the most vital algorithmic program enforced in Hadoop. Every Map and reduce is freelance of different Maps and Reduces. Process of information is dead in parallel to different processes. Employment hardware or job hunter tracks MapReduce jobs that are being dead. Tasks like Map reduce and Shuffle is accepted from Job tracker by a node referred to as Task tracker. Job tracker is that the master node to any or all slave Task tracker nodes that use resources provided by Job tracker to perform one or multiple tasks.

2.3 Pros and Cons of Hadoop 2.2.0

2.3.1 Pros

Hadoop is very stable and provides distributed computing and storage capabilities. Enterprise Readiness of Hadoop makes it easier to figure upon it by several cloud computing enterprises. Tasks area unit freelance of every different. Hadoop's extreme measurability, handiness and fault tolerance is attributable to replication of information that is termed replication issue by default its price is three. [3] Distributed computing in Hadoop permits parallel operating and offers high output.

2.3.2 Cons

One of the few disadvantages of Hadoop is its poor performance that must be resolved. Another issue is cluster management.

3. IMPLEMENTATION

3.1 Deployment of Hadoop

During this paper, we tend to deploy hadoop 2.2.0 over Ubuntu 12.04 LTS with Linux kernel version 3.8.0-29-generic. Primarily hadoop is written in java. To deploy the on top of, we tend to follow these steps:-

Install latest version of JDK on Ubuntu using theUNIX system Command.

$sudo apt-get installs openjdk-7-jdk

Unzip the hadoop-2.2.0.tar.gz into directory location /usr/local and install it exploitation sudo command.

$sudo tar hadoop-2.2.0.tar.gz –C /usr/local

Setting the Hadoop surroundings Variables:

Set the surrounding variables JAVA-HOME to path of JDK and HADOOP_INSTALL to path of Hadoop and PATH to sbin in

Hadoop folder and HADOOP-MAPRED-HOME, HADOOP-HDFS-HAOME ,YARN-HOME to HADOOP

Now put together yarn-site.xml, core-site.xml, mapred-site.xml and hdfs-site.xml consequently.

3.2 Run Hadoop

First format NameNode that is formed by previous execution of hadoop by

$ hdfs namenode –format

Then run shell files of YARN and DFS which can begin Resource Manager, Node Manager, DataNodes, NameNodes and Secondary NameNodes.

After executing above steps Hadoop services are currently in running state.

3.3 Scheduler

The following types of schedulers will be employed in Hadoop: - capacity scheduler, Job Queue Task scheduler.

3.3.1 Job Queue Task

In the earliest Hadoop MapReduce computing design, the essential job kind is huge batch jobs that one user submits, so Hadoop use accountancy rule out early coming up with algorithmic program .The jobs of all users are spoken only 1 queue. Consistent with the priority level and conjointly the time sequence after they are submitted, the whole job queues are scanned, so a satisfactory job is chosen to execute. FIFO is straight forward; the worth of entire cluster coming up with methodology is a smaller amount. Solely single sort of job is essentially designed for FIFO thus once multiple users at an equivalent time run multiple styles of jobs, performance are reaching to be relatively low. Because the usage rate of Hadoop platform is more and more high, the demand is to boot enlarged. The FIFO formula tends to decrease the overall performance of the platform and also the utilization of system resources, and usually even have an effect on the implementation of jobs.

3.3.1 I Capacity scheduler

Fair computer hardware and capacity planning algorithmic rule is extremely similar however improved capacity scheduler used queue rather than pool. Every queue is assigned to associate organization and resources unit of measurement divided among these queues. Icapacity scheduling puts jobs into varied queues in accordance with the conditions, and allocates certain system capability for each queue. If a queue has serious load, it seeks unallocated resources, then makes redundant resources assigned equally to each job .It re-allocates the resources for empty queue to queues exploitation for maximizing resource. Once jobs arrive therein queue, running tasks square measure completed and resources square measure given back to main queue. It conjointly permits priority based totally programming of jobs in associate organization queue. To use icapacity dynamic scheduler, the subsequent property has to be set in yarn-site.xml like below:

<property>

<name>yarn.resourcemanager.schedular.class</name>

<value>org.apache.hadoop.yarn.server.resourcemanager.schedular.capacity.CapacitySchedular

</value>

</property>

4. PROPOSED WORK

We enforced our methodology on Hadoop 2.2.0 as follows:

At first we tend to organized yarn-site.xml file in Hadoop and additional property values for capacity scheduler. By adding properties for icapacity scheduler we tend to inform to icapacity scheduler algorithmic program, the properties of that are organized in ICapacity-scheduler.xml. The following algorithm of icapacity scheduler is used in which first we initialize the queue, second collect the running container then add to the scheduler .It kill the container, if a queue is below capacity because of lack of demand, and so demand will increase, the queue can solely come to capacity as resources area unit discharged from different queues as containers complete.

Initialize from Config file

Minimum allocation, maximum allocation, node locality, threshold

Initialize queues

For (Csleaf Q: queueManager)

{

Resource. add (resTopreempt)

If (Resource greater than Resource Calculator)

Preempt + resource (quemagr. Get LeafQueue)

```
// collect running container
for (sched: scheds)
{
If (Resource. greater than Resource Calculator)
{
for (appsched: sched.getAppSchedulable)
{
For (RMContainer: getLiveContainer)
{
Running container. add(c)
Apps.put (c, appsched.getApp ())
Queues. put (c, sched)
}}}
//kill container
If (time! = null)
{
If (time + waitTime BeforeKill < clock.getTime ())
Create preemped container Status (container .getContainerId ())
CompletedContainer (containers, status, RMcontainer.KILL)
}} }
```

5. RESULT

We have enforced our methodology on Hadoop 2.2.0 and to achieve to the conclusion we have taken the mean of the execution time values of a specific job for a collection of map slots .For comparison 1st we tend to determined values of default Hadoop for a predefined set of variety of map slots and took mean over the set of values found for a selected range of map slots and for identical set of map slots we have determined the execution result values and took mean over the set of values found .In all we tend to compared the values of default hadoop and projected hadoop for a selected range of map slots we found that there was some vital improvement within the execution time and therefore overall performance of Hadoop is improved. We have calculated our results on the below mentioned performance setting.

Table 1. Performance Environment

Memory	2GB
Kernal	Linux 3.8.0_29
OS	Ubuntu 12.04 LTS
CPU	Cori3/2.13GHZ
Scheduler	Capacity scheduler
Version	Hadoop 2.2.0
Benchmark program	word count

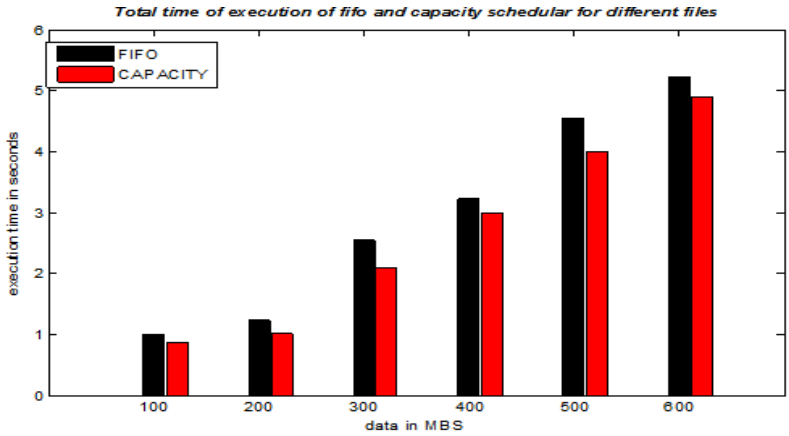

Fig1.Comparitive analysis of execution time of FIFO and Capacity Scheduler for different files

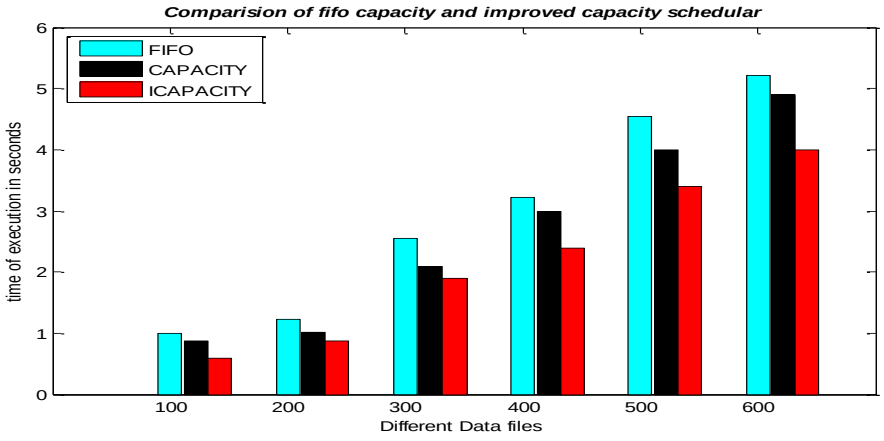

Fig2. Comparative analysis of FIFO, Capacity and Icapacity scheduler

Fig.1 and fig.2 clearly demonstrates the distinction in mean execution times for a selected range of map slots between Default Hadoop and proposed Hadoop.

6. CONCLUSION AND FUTURE WORK

The goal of proposed algorithm during this work is to decrease the completion time of reduced tasks in map-reduce framework. In this implementation the performance of proposed algorithm is estimated from the job-completion time. We compared the proposed capacity scheduler with the default FIFO scheduler; the reduced completion time is low. We introduced queue management and pipelining which help the task to work upon in the queue manner and shared the resources among the sub queue. And it is also proved that the average response time are decreased 29% to 50% when icapacity scheduler is applied .For future we can be explored further in a heterogeneous environment for execution time and output data are consistence

7. REFERENCES

[1] Hadoop, the Apache Software Foundation, May 2012, 1.0.3.

[2] Thusoo,Ashish et al. "Hive : a warehousing solution over map-reduce framework." *Proceedings of the VLDB Endowment* 2.2 pp. 1626-1629,2012.

[3] Hadoop: The Definitive Guide, ed. Third Tokyo: yahoo press [as accessed on Jan 2015].

[4] Shvachko, Konstantin, et al. "The Hadoop Distributed File System."*Mass Storage Systems and Technologies, IEEE*, 2010.

[5] Nicolae, Bogdan, et al."BlobSeer: Bringing High Throughput Under Heavy Concurrency To Hadoop Map-Reduce Applications."*Parallel & Distributed Processing (IPDPS)*, IEEE, 2010.

[6] Xu, Guanghui, Feng Xu, and Hongxu Ma."Deploying and Researching Hadoop in Virtual Machines."*IEEE*, 2012.

[7] Kurazumi, Shiori, et al. "Dynamic Processing Slot Scheduling For I/O." *ICNC*. 2012.