IJIRMET ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 2, Issue 8, August 2017

Extending Data Facilities using the Framework – HadoopMapReduce

[1] M.Sanjay (ResearchScholar), [2] A.Kumaravel
[1]Dep. Of Computer Science Engineering, Bharath University,Chennai – 73 , India.
[2] Dean-School of Computing, Bharath University Chennai – 73 , India

Abstract: ***The High performance computing and Grid Computing Communities have been doing large scale data processing by using API's as Message Passing Interface. High Performance Computing distributes the work across the cluster of machines which assess the shared file system hosted by a Storage Area Network. The problem arises when nodes need to access larger data volumes since the network bandwidth is the bottleneck and compute nodes becomes idle. This problem is solved by the MapReduce.MapReduce is a distributed parallel computing process for large scale data intensive applications like data mining and web indexing. MapReducecollacate the data with the compute node. So the data access is fast because it is local. This feature is called data locality , which is the reason for the High Performance of Mapreduce. Hadoop is an open-source implementation of MapReduce. Due to its parallel programming property mapreduce can analyze very large scale data. Mapreduce is a semi structured and record-oriented program. In this paper we summarize the architecture design , development and functioning of MapReduce.***

Keywords- **Map Reduce *(MR); Volunteer, Parallel computing , distributed computing, data locality***

I. INTRODUCTION

Storage capacities of hard drives have increased now and also the speed to access the data is in demand. Reading and writing the data is very slow. The way to reduce the time is to read from multiple disks at once. Most of the analysis tasks need the data to be read from different disks. There are various ways to combine the data from multiple sources. Mapreduce gives a programming model to get the data from different disks and transforming the data into a Key value pairs. Mapreduce is a program for processing the data distributed processing model for the big size of data applications. Mapreduce is an opensource implementation and it is wisely used to process with speed. Mapreduce implementation uses data locality and also Parallel processing concept is used in map reduce. Mapreduce is a batch query processor, this gives an opportunity to the user to query the large data in a innovative way. Seek time is the time taken for a hard disk controller to locate a specific piece of stored data. Transfer rate is the amount of digital data that is moved from one place to another in a given time. Here seek time is improving more slowly than transfer rate.

1. Hadoop

If the data access pattern is dominated by seeks, it will take longer to read or write large portions of the dataset than streaming through it. Unimaginable amount of data expose in every sector. 2.5 exabytes of data are generated every day. Stock exchange , Mobile phones , Utube , Facebook and Twitter uses lots of data. There are three types of data. Structured data(Table format in oracle) ,Semistructured data which does not have formal data model (XML files), Unstructured data which does not have a predefined data model (text files , images , movies). Big data has three characteristics,which are Variety, Velocity and Volume. Variety manages different structures like relational data , logs and raw data. Velocity determines the streaming of data with high speed. Volume denotes the scaling of data like zeta bytes of data.The requirements of big data are iterative and volatile and the data source also keeps changing, so the program should be written in the way to meet all these needs. The following table shows the difference between RDBMS and Hadoop.

RDBMS	Hadoop
Transactions	Jobs
Structured data, Read wtie mode	Unstructured , Readonly mode
Costly servers	Cheap Commoditiyhardwares

Table: 1.1 Difference between RDBMS and Hadoop

As the data is very large , the fault tolerance and availability should be taken care and also the data from different sources should be combined. Hadoop originated from Nutchopensource project.In 2008 Apache tested with 4000 nodes cluster using hadoop and the performance was good when compared to other technologies. In 2009 hadoop stored petabyte of data in 17 hours successfully. By the end of 2011 hadoop released 1.0 version.

II. HADOOPARCHITECTURE:

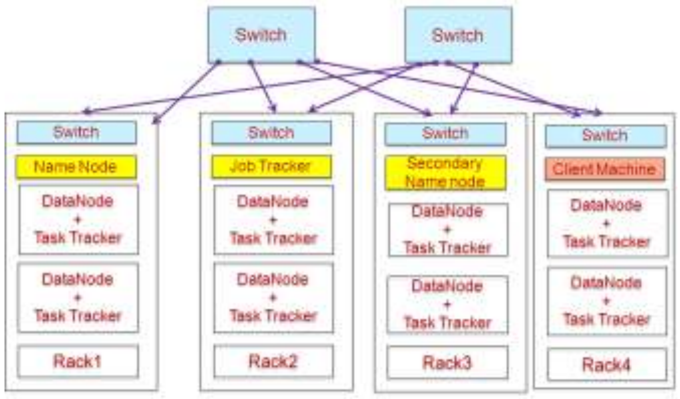

Fig 2.1Hadoop Architecture

Files are divided into blocks and stored across the hadoop cluster.Mapreduce program analyze the data and the results are stored again in the hadoop cluster. The results can be read from the cluster.

3.AMAP REDUCE

Mapreduce is a programming model for processing and generating large set of data with parallel and distributed cluster environment. Mapreduce does the parallel computation across large scale clusters , and also handles machine failures and performance issues. Mapreduce ensures the communication between the nodes. This can be written in Java , C++ , Ruby etc. Mapreduce paper was published in the year of 2004. Apache introduced mapreduce in 2008. Apache introduced mapreduce 2.0 and YARN in 2012. MapReduce takes care of Scheduling ,Task Localization, Error handling and Data synchronization. Scheduling is done by JobTrackers where jobs are broken into smaller chunks called tasks. Tasks are scheduled by the TaskTracker. MapReduce works well on Write Once and Read Many Times (WORM) data. MapReduce allows Parallelism without mutexes. Map and Reduce Operations are performed by the same physical processors. The Minimum overhead and best execution is obtained because the methodologies are basically Reflections. Mapper's StringTokenizer capacity parts each line into words as key worth matches.

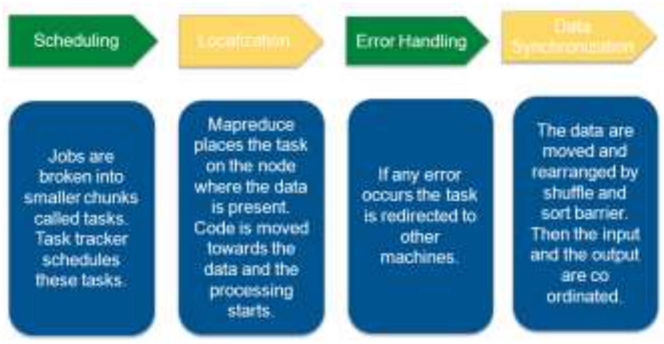
Scheduling
Localization
Error Handling
Data Synchronization
Jobs are broken into smaller chunks called tasks. Task tracker schedules these tasks.
Mapreduce places the task on the node where the data is present. Code is moved towards the data and the processing starts.
If any error occurs the task is redirected to other machines.
The data are moved and rearranged by shuffle and sort barrier. Then the input and the output are co ordinated.

Fig. 3.1 Mapreduce Framework Flow

HDFS

HDFS is a distributed file system, where large files can be stored with streaming Data Access Patterns. HDFS runs on multiple clusters with commodity hardware.HDFS has upholding simple coherency model. The data processing is done in a write-once read-many-timespattern. HDFS uses 64MB block size by default. To minimize the seeking cost the HDFS blocks are designed large. The input files are split up into number of chunks or blocks of pre-defined size. It has name node (brain of the system) , data node and secondary name node. If the name node crashes the entire system is dead. The chunks are stored into 3 data nodes thus providing the fault tolerant by replication. The default setting is 3. This can be modified via dfs.replication parameter in the xml file called hdfs-site.xml. Data pipelining helps the client to retrieve the required data. The role of name node is to store the Meta data, to manage the file, block and replica and also to watch the data nodes through block reports (every 10^{th} heartbeat is called a block report). Secondary name node stores the edit logs and fsimage from the primary name node and updates the fsimage to the name node. HDFS can be accessed by three methods: Command line, JAVA API and Web HDFS. Parallel copying between two clusters can be done through a program named distcp. HadoopArchives compensates the inability of the HDFS to store the small data efficiently bystoring it in the name node in the limited space. Although HAR files are transparent to retrieve the data, it does not permit to add or remove the data without recreation of the file.

The client asks for the namenode to get the location where the storage is available. After getting this information the client starts interacting the data node directly by pushing the data into the blocks. After the completion of wrting the data in a first data node then it will start replicating. The Name node uses Rack Awarness Program to store two duplicates in one rack and alternate duplicate in alternate rack.Datanode send heartbeats to the namenode every 3 seconds through TCP handshake. Missing heart beats signify that the node is lost. Based on the block report from the dead node , the name node decide to store this information in different block. To do so the name node consult the RackAwarness Program to script and directs the data node to do the re-replication.These transactions happens through TCP on port 50010. Balancer utility is the one which is helpful to add new clusters. Factors before planning the clusters are Licence , storage capacity , throughput speed , high availability with automatic failover, services like resource manager, zookeeper ,metrics ,webserver,History Server and also the size of the cluster
either single or medium or Large cluster. Name node requires around 64GB of RAM supports approximately 100 million files.Softwares required isRedhatLinux 5.4 or higher version. Hadoop 2 version with (CDH) Cloudra Distribution 5.1 version.

WorkFlow in Hadoop

MapReduce breaks the process into Map phase and Reduce phase. Map Function passes the information to the reduce function where it is being processed by sorting and shuffling and then the unwanted data are erased by the reduce function.
Each phase has a key and a value. MapReduce functions are developed by the developer to process the data. The output ie, the key value is being sorted out and grouped by the Key before being send to reduce function. The process requires a map function, a reduce function and a code. Mapper class which is associated with the map function and Reducer class which is associated with Reduce function are now being provided by Apache. The output type of map function must match the input type of reduce

function. The MapReduce works in the following way using the Classic framework. The client submits the MapReduce job to the Jobtracker. Job tracker assigns the task to the task tracker and also contacts the distributed file system to retrieve the input splits.

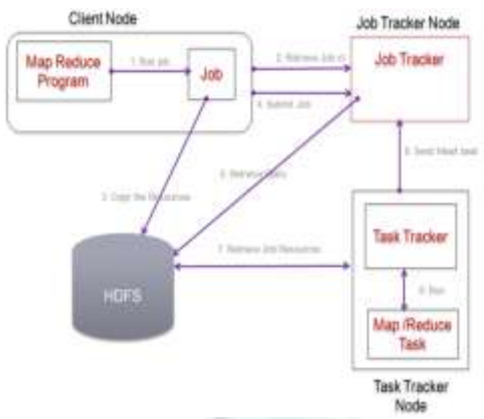

Fig. 3.2MapReduce Diagram

Task trackers which are java applications run the task that has been split by sending the periodic heartbeat to the job tracker.Task tracker retrieves the job resources from the shared file system and it spawns a JVM process for each input split as per the instruction of Job tracker. Now, the JVM runs the Mappercalled the Run Method. Finally, the job tracker cleans up its state and also urges the task tracker to do the same.
Map reduce can be a complement to RDBMS. Mapreduce is the best fit for the applications where the data is written once and read many times. Mapreduce works well with the unstructured data. It is a linearly scalable programming model. If the size of the data is double then the job will run twice as slow.

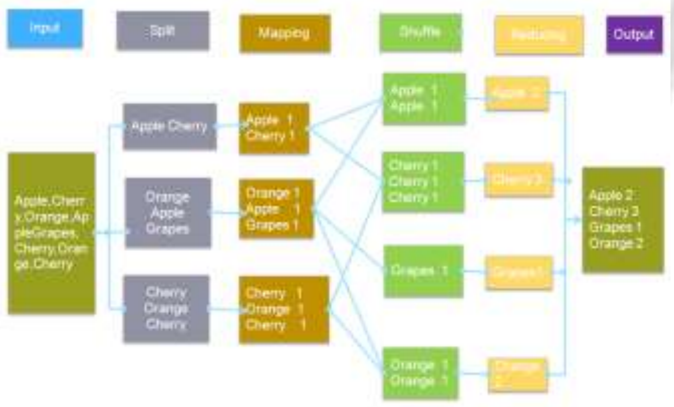

Fig 3.3 Example of Mapreduce Workflow

If the size of the cluster is increased to double then the job will run as faster than the original. This is not true with the database. Mapreduce takes care of the failure situation. The mapreduce implementation detects failed map or reduce tasksand reschedule the job. Mapreduce can do this because it is using the shared-nothing architecture.InShared nothing architecture one task does not dependon another tasks. Mapreduce reruns the map job rather than rerunning the reduce job. Because the reducer has to make sure that it retrieves the correct map outputs. So the order of running the task is not necessary. Map function is a data preparation phase also it drops the bad records. The map function output will be processed by the mapreduce frameworkbefore being sent to the reduce function.

Cluster Configuration

Case study

In this paper we have undergone a case study on the marks of school students using hadoop. In this program we analyze the raw data given by the school. The raw data contains RollNo,Year of passing, Name of the student ,marks in different subjects. The objective is to find the highest mark in physics in each year.

```
1012001Student1198180170
1022001Student2199190180
1032001Student3199199199
1042002Student4170180190
1052002Student5171172173
1062002Student6140136190
1072002Student7178198130
1082003Student8188188188
1082003Student9170170170
1082003Studen10130130130
```

Fig 3.4 Sample raw data

The format of the raw data is as below.

Sr.No	Position	Content
1	1 to 3	Student Id
2	4 to 7	Year
3	8 to 15	Student Name
4	16 to 18	Maths
5	19 to 21	Physics
6	22 to 24	Chemistry

The map reduce program helps to find out the maximum mark in physics for each year.

The above value is presented to the map function as a key and value.The keys are the offset and the values are the lines.

```
(0,1012001Student1198180170)
(24,1022001Student2199190180)
(48,1032001Student3199199199)
(72,1042002Student4170180190)
(96,1052002Student5171172173)
(120,1062002Student6140136190)
(144,1072002Student7178198130)
(168,1082003Student8188188188)
(192,1082003Student9170170170)
(216,1082003Studen10130130130)
```

The map function will extract the year and the physics mark.

```
(2001,180)
```

```
(2001,190)
(2001,199)
(2002,180)
(2002,172)
(2002,136)
(2002,198)
(2003,188)
(2003,170)
(2003,130)
```

The output of the map function is processed by the Mapreudce framework before it goes to the reduce function. It sorts and group the key value pair.

```
(2001,[180,190,199])
(2002,[180,172,136,198])
(2003,[188,170,130])
```

Now the reduce function takes the above value and findout the max value. Below is the final output of the program.

```
(2001,199)
(2002,198)
(2003,188)
```

Year	Physics Marks
2001	199
2002	198
2003	188

Fig. 3.5 output data

```
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;

public class FindMaxPhysicsMapper extends Mapper
<Text,Text,Text,Text>
{

public void map(Text key,Text Value,Context context)
throws IOException, InterruptedException
     {
             String strPhysicsMark=
value.toString().subString(18,21);
             String strYearOfPassing =
value.toString().subString(3,7);       context.write(new
Text(strYearOfPassing),new  Text      (strPhysicsMark));
     }
}
```

Example 3.4 Mapper class for Finding maximum mark

```
import org.apache.hadoop.Text;
import org.apache.hadoop.mapreduce.Reducer;
import java.io.IOException;

public class FindMaxPhysicsReducer extends
Reducer<Text,Text,Text,Text>
{
        @override
        protected void reduce(Text key,Iteratable<Text>
values,
          Context context) throws
IOException,InterruptedException
        {
                int iMaximumPhysicsMark = 0;
                for(Text val : values)
                {
iMaximumPhysicsMark =
Math.max(iMaximumPhysicsMark,val.toString());
                }

                context.write(key,new
Text(iMaximumPhysicsMark));
        }
}
```

Example 3.5Reducer class for Finding maximum mark

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import
org.apache.hadoop.mapreduce.lib.input.KeyValueTextInputFormat;
import org.apache.hadoop.mapreduce.job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
public class Execution
{
        public static void main(String args[])
        {
                Configuration conf = new Configuration();
                Job job = new Job(conf,"MaxMarkFinding");
                job.setJarByClass(Execution.class);
                job.setMapperClass(FindMaxPhysicsMapper);
                job.setReducerClass(FindMaxPhysicsReducer);
                job.setOutputKeyClass(Text.class);
                job.setOutputValueClass(Text.class);
                job.setMapOutputKeyClass(Text.class);
                job.setMapOutputValueClass(Text.class);

        job.setInputFormatClass(KeyValueTextInputFormat.class);

        job.setOutputFormatClass(TextOutputFormat.class);
                FileInputFormat.addInputPath(job,new
Path(args[0]));
                FileOutputFormat.setOutputPath(job,new
Path(args[1]));
          boolean = boolExecutionResult =
job.waitForCompletion(true);
          System.exit(result ? 0 : 1);
        }
}
```

Example 3.6 Client class for Finding maximum mark

We need to make sure that the output directory is not available in the output path.
The above code is executed as below.

3.B Map Reduce Failure and Recovery

The Map Reduce task processes send periodic heart beats to the task tracker. The task tracker in turn sends periodic heartbeats to job tracker to be assigned to a task. If the task assigned by the task tracker does not respond for over 10 minutes or if it throws some exception then the task tracker kills the task and reports to the job tracker about the failure of the task.Job tracker then schedules the task to a different task tracker. If the task fails continuously for more than 4 times then the job fails. Following diagram represents the Failure and recovery mechanism in Map Reduce.

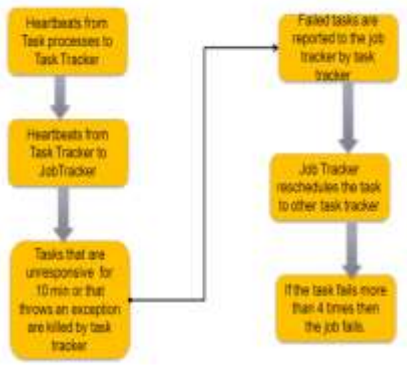
Heartbeats from Task processes to Task Tracker
Heartbeats from Task Tracker to JobTracker
Tasks that are unresponsive for 10 min or that throws an exception are killed by task tracker
Failed tasks are reported to the job tracker by task tracker
Job Tracker reschedules the task to other task tracker
If the task fails more than 4 times then the job fails.

Fig 3.7 Failure and Recovery in Map Reduce

4 .Hadoop installation:

Install sshservers , login ssh , install java 6 , install hadoop, Extract Hadoop Jar , copy the tar file into /usr/local , set path in .bashrc. set environment in hadoop-env.sh , set the core-site.xml file , mapred-site.xml and start the hadoop service.
Core-site.xml where specify the name node. Hdfs-site.xml is where we specify the dfs.replicatin =3 .mapred-site.xml is where jobtrackerip address is mentioned. Edit master file and put master ipaddress , similarly enter slave ip address in the slave file.

Install jdk 7. Install ssh for communicating between nodes.

```
>hadoop1.2.1/bin/start-all.sh
```
is the command to start the hadoop

```
>hadoop1.2.1/bin/jps
2970   JobTracker
2576   Name Node
2885   SecondaryNameNode
3107   TaskTracker
3331   Jps
2736   DataNode
```

Benchmark for the read and write need to be set.
For read operation

```
TestDFSIO –read nrFiles 5 –fileSize 100
```

Set the block size in hdfs-site.xml -dfs.block.size=64MB
Stop-all.sh to stop hadoop service.

Upload the file using the command hadoopfs –copyFromLocal /source hdfs:/myfolder/destination

To see the block size ,localhost :50070
To see the data node 50075
See the size of the block and all.

The below table has the File name where the configuration has to be changed , the port number for the RPC and Web port through which the Namenode, Data Node , Secondary Name Node , TaskTracker and JobTracker can be viewed.

Node	**FileName**	**RPC Port**	**Web Port**
NameNode	Core-site.xml	50000	50070
Jobtracker	mapred-site.xml	50001	50030
secondary name node	masters	empty	50090
Data Node	slaves	50010	50075
Task Tracker	Slaves	50020	50060

Refresh the cluster as below. For decommissioning the data node , we need to do the below steps. Create an exclude file. Put the ip address for the data nodes to be decommissioned.
```
hadoopdfsadmin –refreshNodes
```

IJIRMET ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 2, Issue 8, August 2017

5. Conclusion

This study talks about the function of MapReduce and its benefits for analyzing large amount of data. The performance of MapReduce is proven to be higher than that of the High Performance computing and Grid computing. This performance can further be improved by using the latest version of MapReduce which is called YARN. As a casestudy we have shown the way to analyze the student maximum record by using by developing a MapReduce. Future study will be to analyze the function , architecture and developing YARN.

References:

1. Hadoop Definitive Guide by Tom White,   Third Edition – O'Reilly publications.
2. https://hadoop.apache.org/
3. Hadoop Dummies – Dirk deRoos
4. Hadoop Beginner's guide - Turkington
5. Optimizing Mapreduce functionality in Big data using Cache Manager - Devi L and Gowri S
6. Hadoop Cluster Deployment – Shroff Publishers &Distributors - Publication Year 2013.
7. Predicting Student Performance Using MapReduce - Dr. N. Tajunisha, M. Anjali
8. HybridMR: A New Approach for Hybrid MapReduce CombiningDesktop Grid and Cloud Infrastructures - Bing Tang, Haiwu He2 and Gilles Fedak