MRCP-RMD: Resource allocation and Scheduling of jobs with Deadlines

Shashank G Hegde, Vinayak Suresh Pai, Vinitha S Bhat,Srinidhi L N

Final year B.E Students(UG)
Guided by: Mrs. Visalini S, Assistant Professor Department of ISE, TOCE
The Oxford College of Engineering, Bengaluru ,India

Abstract : MRCP-RM: Big Data is the problem raised due to large storage of dataset and traditional data processing .Data processing from large set of data holds great disadvantages over time constraint ,in order to overcome this constraint Deadline(priority scheduling) is introduced in this research. The method for Resource Allocation and Scheduling of Map Reduce Jobs with Deadlines, which focuses on resource allocation and scheduling of MapReduce jobs according to client's requirements. Big data is the domain used for resource allocation and scheduling of jobs. Parallel processing on distributed data of commodity hardware in a reliable, fault-tolerant manner. An open stream of MapReduce jobs with SLA (Service Level Agreement) on a distributed computing environment i.e. HDFS (Hadoop Distributed File System). Hadoop distributed file system allows storing and effective retrieval of data. Existing Hadoop scheduler do not support completion time guarantee and performance of completion of jobs is less. Algorithm used is MRCP-RM (MapReduce Constraint Programming based on resource allocation). MRCP-RM not only maps all the newly submitted jobs but also remaps the tasks of jobs that have previously been executed but have not started executing. Scheduler consists of MRCP-RM code in which the data are sorted and shuffled and deadline control is achieved by setting job priorities at scheduling phase. The goal of this project is to enhance the performance, scalability and throughput over deadline control by 15% to 18%.

Index Terms—Resource allocation, constraint programming and scheduling of jobs.

1. INTRODUCTION

Big Data is the term that refers not only to the large volumes of data but it is also concerned about the complexity of the data and the speed at which it is getting generated. It is generally described by using three characteristics, widely known as 3 V's:

- VOLUME

The size is one of the characteristics that define big data. Big data consists of very large data sets. However, it should be noted that it is not the only one parameter and for data to be considered as big data, other characteristics must also be evaluated.

- VELOCITY

The speed at which the data is being generated is an important factor. For example, in every one second thousands of tweets are tweeted on microblogging platform, Twitter. Even if the size of each individual tweet is 140 characters, the speed at which it is getting generated makes it an eligible data set that can be considered as big data.

- VARIETY

Big data comprises data in all formats: structured, unstructured or combination of both. Generally, it consists of data sets, so complex that traditional data processing applications are not sufficient to deal with them. All these characteristics make it difficult for storing and processing big data using traditional data processing application software's. Two papers published by Google, build the genesis for Hadoop. Hadoop is an open source frame-work used for distributed storage and parallel processing on big data sets. Two core components of Hadoop are:

- HADOOP DISTRIBUTED FILE SYSTEM (HDFS)

Used for distributed storage of data. The input file is first split into blocks of equal size except the last block which are then replicated across Data Nodes. Currently, default block size is 128 MB which was previously 64 MB and default replication factor is 3. Block size and replication factors are configurable parameters.

- MAPREDUCE

For parallel processing on distributed data on cluster of commodity hardware in a reliable, fault-tolerant manner.
A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks.
There are plenty of resources available that describe the detailed architecture of Hadoop and about how it works. Anyone who is not aware about what Hadoop is at all and how it can help manage big data is suggested to get a basic understanding of it before continuing here since basic understanding of Hadoop is required to understand the concepts discussed in following sections.

2. LITERATURE REVIEW

Deadline-based Workload Management for MapReduce Environments: Pieces of the Performance Puzzle.
Author: Abhishek Verma, Ludmila Cherkasova, Vijay S. Kumar, Roy H. Campbell.
Design of new schedulers for Mapreduce environment to enhance the workload management for processing Mapreduce jobs with deadlines. A policy for job ordering in the processing queue. Allocation and de-allocation spare resources in the system among the active jobs.

Engineering Resource Management Middleware for Optimizing the Performance of Clouds Processing MapReduce Jobs with Deadlines
Norman Lim, Shikharesh Majumdar, & Peter Ashwood-Smith
Software packages to formulate and solve the matchmaking and scheduling problems. High resource utilization and adequate revenue. Achieve high system performance.

A Constraint Programming-Based Resource Management Technique for Processing MapReduce Jobs with SLAs on Clouds
Peter Ashwood-Smith Norman Lim and Shikharesh Majumdar.
Effective technique for resource management on clouds for jobs characterized by an end-to-end SLA comprising an earliest start time, execution time, and deadline.

Scheduling in MapReduce-like Systems for Fast Completion Time
Hyunseok Chang, Murali Kodialam, Ramana Rao Kompella, T. V. Lakshman, Myungjin Lee, Sarit Mukherjee
A linear program that minimizes the job completion times to solve the problem. Achieve feasible schedules within a small constant factor of the optimal value of the objective function.

MARLA: MapReduce for Heterogeneous Clusters
Zacharia Fadika 1, Elif Dede 2, Jessica Hartog 3, Madhusudhan Govindaraju
Efficient and swift processing of large scale data with a cluster of compute nodes. Performing well not only in homogeneous settings, but also when the cluster exhibits heterogeneous properties.
The performance gains exhibited by our approach against Apache Hadoop and MARIANE in data intensive and compute intensive applications.

3. SYSTEM ARCHITECTURE

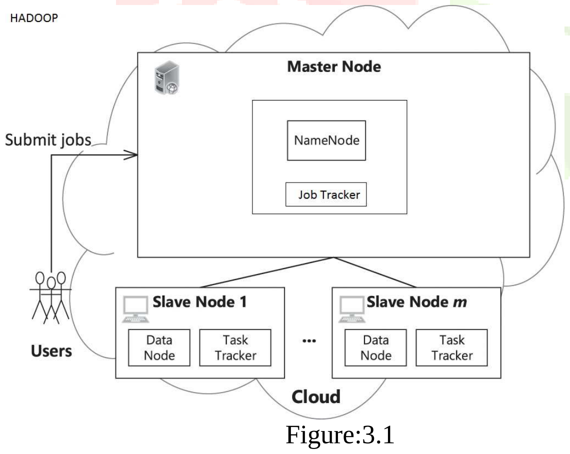


Figure:3.1

HDFS(STORAGE):
HDFS is designed for storing very large files with streaming data access and running on clusters of commodity hardware.
A dataset is typically generated or copied from source and then various analysis is performed on that data set over time.
MAPREDUCE(COMPUTATION):
Map –Reducing is a processing technique that allows scalability across hundreds or thousands of servers in a Hadoop clusters.
Map-Reduce algorithms contain two important tasks map and reduce. The shuffle and sort process is dependent mainly on value of data sets, At last we are scheduled and Map-Reducing jobs
JOB TRACKER:
This resides in master node and is responsible for accepting job compliances from clients scheduling task to run on nodes, and providing executive functions such as condition and task progress watching to the bunch.
TASK TRACKER:

This accepts task from the JobTracker, initiate to the user program to execute these jobs locally and reports are sent to the JobT immediately.
NAMENODE (MASTER NODE):
The name node is a master of all daemons and is use to map the block and store file in HDFS format.
SECONDARY NAME NODE:
Performs name node operation log check point. Acts as a backup for Name node
DATA NODE (SLAVE NODE):
The Data Node is used in creation, insertion and deletion of data or files based on the order of name node.

Data flow diagrams:

DFD-L0:

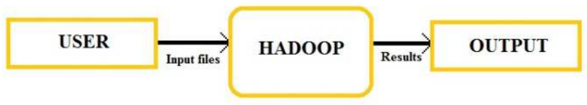


Figure:3.2

DFD-L1:

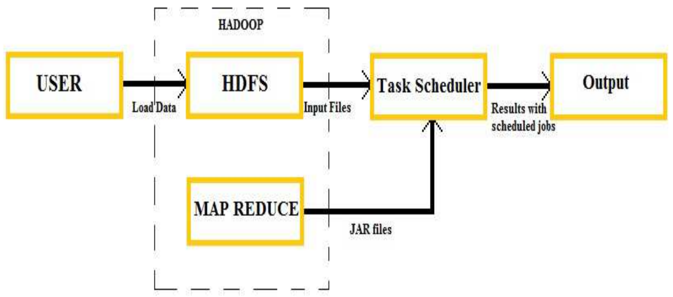


Figure:3.3

DFD-L2:

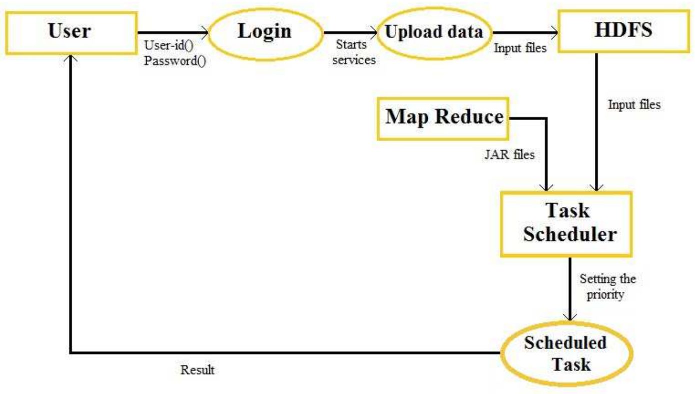


Figure:3.4

Data Flow:
The data flow diagram shows the flow of data from end user to appliances. The user has to register using user id and password in order to submit the query in the cloud server. The query will be processed using process user query event. The HDFS redirects the query to the task scheduler which resides in the Name Node.
The task will be scheduled and directed to the data node. The jobs are scheduled and submitted back to task scheduler. The chunks of data are merged and the results are produced to user

4. MODULE SETS

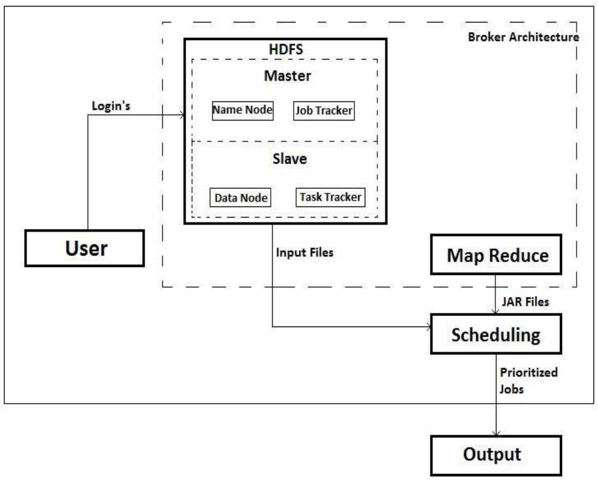


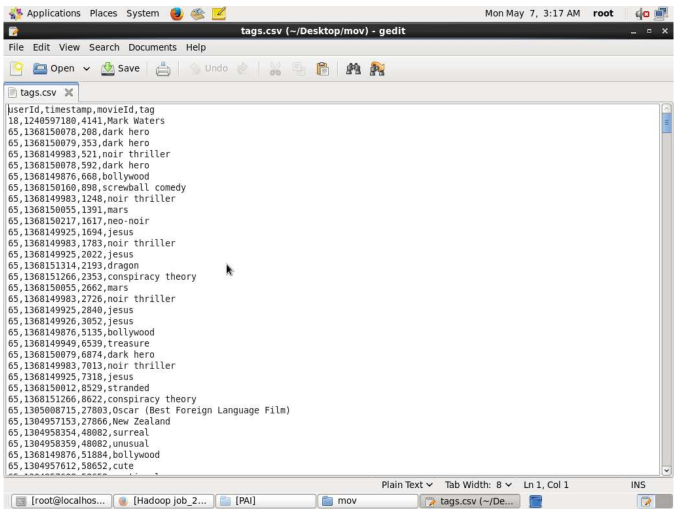
Figure:4.1

Admin/User: User logs into system using user-id and password. Authorization is needed to identify the user. Only authenticated users can access the system. Admin/User has to start all Hadoop services.
HDFS: It is used to store files in distributed manner. Namenode has meta data where as Datanode will be having actual data. All the input files are stored from local to hdfs using commands. Files are stored inform of chunks, default size is 64MB and can vary upto 1024MB. Replication Factor is 3. Parallel processing is done, to extract data from HDFS.
Map Reduce: It produces key; value pairs. It consists of 3 classes Mapper, Reducer and Driver class. Driver class is a main class used during compilation. It provides key and value pairs for given input files and also removes duplication. All programs are compiled here using JAR files, Driver class name, Input file and Output directory.
Scheduling: Each JAR file generates unique job-id. Priorities to each job-id is assigned according to client's/user requirement. It prioritizes each jobs and executes it as early as possible by increasing scalability and throughput.

5. IMPLEMENTATION

STEP: 1

- User login's to system.
- Opens terminal and starts all hadoop services using "start-all.sh" command.
- 3 times password should be entered to start all the services.
- Using "jps" command, we can check all the hadoop daemons running and also to check the activities of all the nodes.


Applications Places System
Mon May 7, 3:17 AM root
tags.csv (~/Desktop/mov) - gedit
File Edit View Search Documents Help
Open Save Undo
tags.csv
Plain Text Tab Width: 8 Ln 1, Col 1 INS
[root@localhos... [Hadoop job_2... [PAI] mov tags.csv (~/De...

```
userId,timestamp,movieId,tag
18,1240597180,4141,Mark Waters
65,1368150078,208,dark hero
65,1368150079,353,dark hero
65,1368149983,521,noir thriller
65,1368150078,592,dark hero
65,1368149876,668,bollywood
65,1368150160,898,screwball comedy
65,1368149983,1248,noir thriller
65,1368150055,1391,mars
65,1368150217,1617,neo-noir
65,1368149925,1694,jesus
65,1368149983,1783,noir thriller
65,1368149925,2022,jesus
65,1368151314,2193,dragon
65,1368151266,2353,conspiracy theory
65,1368150055,2662,mars
65,1368149983,2726,noir thriller
65,1368149925,2840,jesus
65,1368149926,3052,jesus
65,1368149876,5135,bollywood
65,1368149949,6539,treasure
65,1368150079,6874,dark hero
65,1368149983,7013,noir thriller
65,1368149925,7318,jesus
65,1368150012,8529,stranded
65,1368151266,8622,conspiracy theory
65,1305008715,27803,Oscar (Best Foreign Language Film)
65,1304957153,27866,New Zealand
65,1304958354,48082,surreal
65,1304958359,48082,unusual
65,1368149876,51884,bollywood
65,1304957612,58652,cute
```

Figure:5.1

STEP: 2

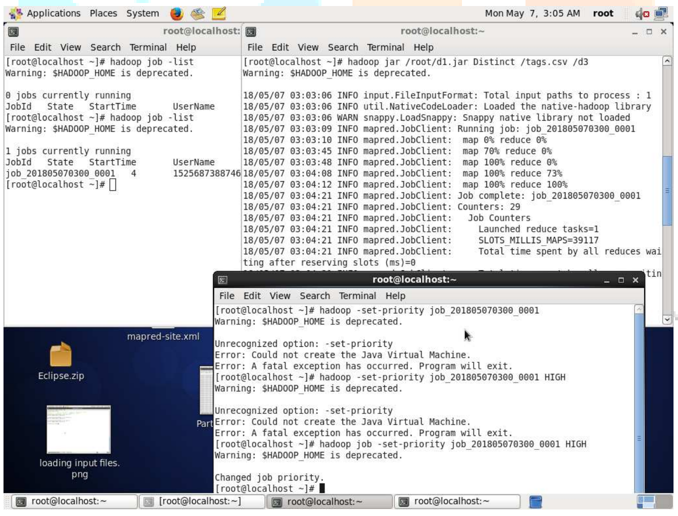
- Hadoop uses HDFS to store files in form of chunks.
- Create a directory under root, where all the input data is to be stored.
- Using "cd" change the directory, where input data is stored.
- Using "hadoop fs –put filename.extension /(enter)" command, load all the required input data files from local to hadoop (HDFS).
- Open browser and type http://localhost:50070.
- Click on Browse File System to see whether the files are load into hadoop(HDFS).

STEP: 3

- In this, we have totally 4 programs i.e.

1. Map Reduce.
2. Partitioner.
3. Frequency.
4. Distinct.

- Create jar files of these programs using Eclipse.
- In terminal, using "hadoop jar /root/name_of_jar.jar driver_class_name / input_file /output_directory" to execute the input files using jar files.
- For each program single job-id will be created.
- Change the configuration of fair scheduler and mapred xml files in filesystem/usr/local/Hadoop/conf folder.
- Reboot the system once configuration are changed.
- Run the jar files once again
- Using "hadoop job –list" command check currently running jobs.
- Once the job-id is created.
- Set priority to jobs using "hadoop job –set-priority".
- Valid values for priority are : VERY_HIGH, HIGH, NORMAL, LOW, VERY_LOW.
- Syntax is "hadoop job –set-priority values job-id".



```
[root@localhost ~]# hadoop job -list
Warning: $HADOOP_HOME is deprecated.

0 jobs currently running
JobId   State   StartTime       UserName
[root@localhost ~]# hadoop job -list
Warning: $HADOOP_HOME is deprecated.

1 jobs currently running
JobId   State   StartTime       UserName
job_201805070300_0001   4       1525687388746
[root@localhost ~]#
```

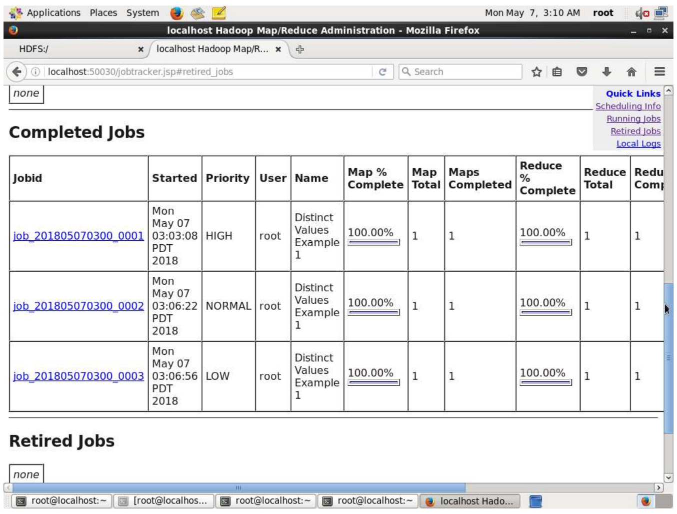
```
[root@localhost ~]# hadoop jar /root/d1.jar Distinct /tags.csv /d3
Warning: $HADOOP_HOME is deprecated.

18/05/07 03:03:06 INFO input.FileInputFormat: Total input paths to process : 1
18/05/07 03:03:06 INFO util.NativeCodeLoader: Loaded the native-hadoop library
18/05/07 03:03:06 WARN snappy.LoadSnappy: Snappy native library not loaded
18/05/07 03:03:09 INFO mapred.JobClient: Running job: job_201805070300_0001
18/05/07 03:03:10 INFO mapred.JobClient:  map 0% reduce 0%
18/05/07 03:03:45 INFO mapred.JobClient:  map 70% reduce 0%
18/05/07 03:03:48 INFO mapred.JobClient:  map 100% reduce 0%
18/05/07 03:04:08 INFO mapred.JobClient:  map 100% reduce 73%
18/05/07 03:04:12 INFO mapred.JobClient:  map 100% reduce 100%
18/05/07 03:04:21 INFO mapred.JobClient: Job complete: job_201805070300_0001
18/05/07 03:04:21 INFO mapred.JobClient: Counters: 29
18/05/07 03:04:21 INFO mapred.JobClient:   Job Counters
18/05/07 03:04:21 INFO mapred.JobClient:     Launched reduce tasks=1
18/05/07 03:04:21 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=39117
18/05/07 03:04:21 INFO mapred.JobClient:     Total time spent by all reduces wai
ting after reserving slots (ms)=0
```

```
[root@localhost ~]# hadoop -set-priority job_201805070300_0001
Warning: $HADOOP_HOME is deprecated.

Unrecognized option: -set-priority
Error: Could not create the Java Virtual Machine.
Error: A fatal exception has occurred. Program will exit.
[root@localhost ~]# hadoop -set-priority job_201805070300_0001 HIGH
Warning: $HADOOP_HOME is deprecated.

Unrecognized option: -set-priority
Error: Could not create the Java Virtual Machine.
Error: A fatal exception has occurred. Program will exit.
[root@localhost ~]# hadoop job -set-priority job_201805070300_0001 HIGH
Warning: $HADOOP_HOME is deprecated.

Changed job priority.
[root@localhost ~]#
```

Figure:5.2

STEP: 4

- Open browser and type http://localhost:50030.
- Click on completed jobs to view the output.



none

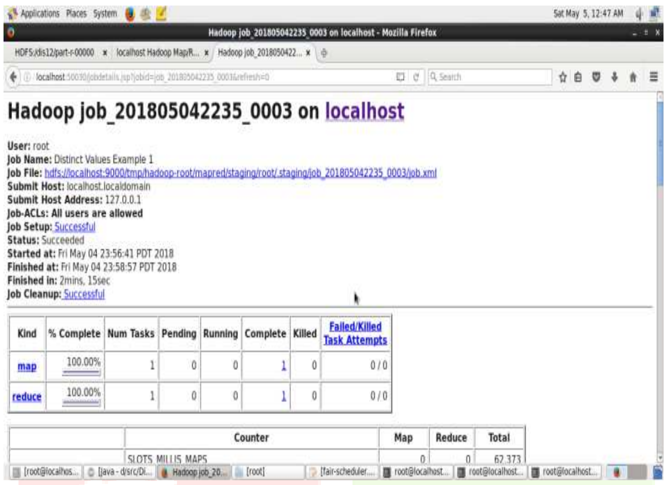
Completed Jobs

Jobid	Started	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Redu Comp
job_201805070300_0001	Mon May 07 03:03:08 PDT 2018	HIGH	root	Distinct Values Example 1	100.00%	1	1	100.00%	1	1
job_201805070300_0002	Mon May 07 03:06:22 PDT 2018	NORMAL	root	Distinct Values Example 1	100.00%	1	1	100.00%	1	1
job_201805070300_0003	Mon May 07 03:06:56 PDT 2018	LOW	root	Distinct Values Example 1	100.00%	1	1	100.00%	1	1

Retired Jobs

none

Figure:5.3


Applications Places System
Sat May 5, 12:47 AM
Hadoop job_201805042235_0003 on localhost - Mozilla Firefox
HDFS:/dis12/part-r-00000
localhost Hadoop Map/R...
Hadoop job_20180504422...
localhost:50030/jobdetails.jsp?jobid=job_201805042235_0003&refresh=0
Search
[root@localhos...
[java - d/src/Di...
Hadoop job_20...
[root]
[fair-scheduler...
root@localhost...
root@localhost...
root@localhost...

Hadoop job_201805042235_0003 on localhost

User: root
Job Name: Distinct Values Example 1
Job File: hdfs://localhost:9000/tmp/hadoop-root/mapred/staging/root/.staging/job_201805042235_0003/job.xml
Submit Host: localhost.localdomain
Submit Host Address: 127.0.0.1
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Succeeded
Started at: Fri May 04 23:56:41 PDT 2018
Finished at: Fri May 04 23:58:57 PDT 2018
Finished in: 2mins, 15sec
Job Cleanup: Successful

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	1	0	0	1	0	0 / 0
reduce	100.00%	1	0	0	1	0	0 / 0

	Counter	Map	Reduce	Total
	SLOTS_MILLIS_MAPS	0	0	62,373

Figure:5.4

6. EXPERIMENTAL RESULTS

In this research there are three jobs given as input logs i.e., tags.csv,movies.csv,ratings.csv. These file consist details like movie id,timestamp,ratings,year,type of movies .
The final browser screen would appear with columns as JOBID,STARTED,PRIORITY,USER,NANE,MAP AND COMPLETED REDUCE TOTAL,REDUCE COMPLETE. The job id is unique for every job. This id will display the starting and ending time of the job.The priorities are like HIGH,VERY HIGH,NORMAL,LOW,and VERY LOW.
The priority is set during scheduling.

SCALABILITY:

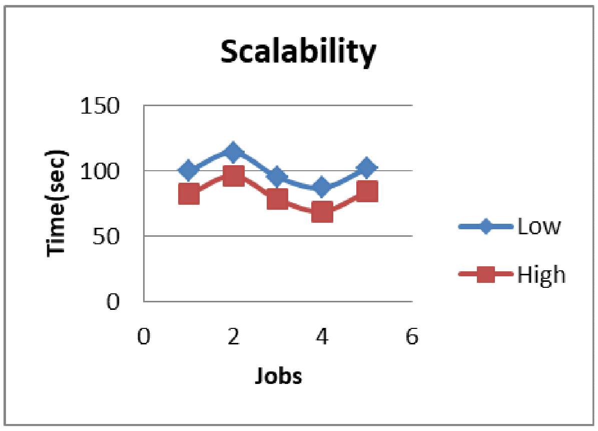


Figure:6.1

Existing system takes more time to complete 5 jobs because of low priority, whereas in proposed system it takes less time to complete 5 jobs because we are setting priority to high. Thus we are able to complete more number of jobs compared to existing one. With extra 15% to 18% efficiency than existing one.

THROUGHPUT:

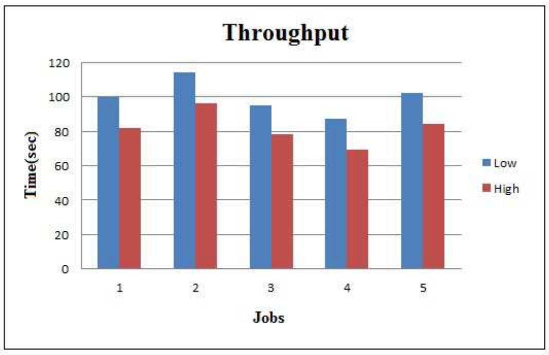


Figure:6.2

It takes less time to complete jobs. Thus we can complete extra 15% to 18% of the jobs early compared to existing system.

7. CONCLUSION AND FUTURE WORK

The goal of this research is devising effective matchmaking and scheduling techniques for efficiently processing an open stream of MapReduce jobs with SLAs on a distributed computing environment with m resources, such as a private cluster or a set of resources acquired a priori from a public cloud. A key research objective is to achieve high system performance while ensuring matchmaking and scheduling overhead is low. A constraint programming based resource management technique called MRCP-RM that can efficiently perform matchmaking and scheduling of an open stream of MapReduce jobs with SLAs .A constraint programming based resource management technique called MRCP-RMD that can efficiently perform matchmaking and scheduling of an open stream of MapReduce jobs has been implemented. The implementation was successful with effective prioritization of jobs; hence one can infer that hadoop is apt when it comes to resource allocation and management.

Demonstration of effectiveness as reflected in the ability to achieve high system performance while incurring a small overhead was possible

Following achievements were made:

1Throughput control.

2) Improved Scalability.

3) Effective usage of resources.

All the factors mentioned above leads to customer satisfaction and also helps cut costs for the company by effective resource management.

REFERENCES

1. N. Lim, S. Majumdar, and P. Ashwood-Smith, “Engineering resource management middleware for optimizing the performance of clouds processing MapReduce jobs with deadlines,” in Proc. Int. Conf. Perform. Eng., Mar. 24-26, 2014, pp. 161–172.

2. N. Lim, S. Majumdar, and P. Ashwood-Smith, “A constraint programming-based resource management technique for processing. MapReduce jobs with SLAs on clouds,” in Proc. Int. Conf. Parallel
Process, Sep. 9-12, 2014, pp. 411–421

3. Verma, L. Cherkasova, V. S. Kumar, and R. H. Campbell, “Deadline-based workload management for MapReduce environments: Pieces of the performance puzzle,” in Proc. Netw. Operations Manage. Symp, Apr.16-20, 2012, pp. 900–905.

4. Z. Fadika, E. Dede, J. Hartog, and M. Govindaraju, “MARLA: MapReduce for heterogeneous clusters,” in Proc. IEEE/ACM Int. Symp. Cluster Cloud Grid Comput, May 13-16, 2012, pp. 49–56

5. H. Chang, M. Kodialam, R. R. Kompella, T. V. Lakshman, M. Lee, and S.Mukherjee, “Scheduling in mapreduce-like systems for fast completion time,” in Proc. IEEE INFOCOM, Apr. 10-15, 2011, pp. 3074–3082.