Hadoop Performance Modelling for Job Estimation and Resource Allocation

Mr. Ankit Sanghvi[1] , Mr. Prince P.Vishwakarma[2], Mr. Shivmurat S.Yadav[3], Ms. Lavina R. Patil[4]

[1] HOD Dept. of Computer Engineering, Alamuri ratnamala institute of engineering and technology, Shahapur, Thane, India.

[2 ,3 ,4] UG Scholar, Dept. of Computer Engineering, Alamuri ratnamala institute of engineering and technology, Shahapur, Thane, India

1.Abstract:

MapReduce has become a major computing model for data intensive applications. Hadoop, an open source implementation of MapReduce, has been adopted by an increasingly growing user community. Cloud computing service providers such as Amazon EC2 Cloud offer the opportunities for Hadoop users to lease a certain amount of resources and pay for their use. However, a key challenge is that cloud service providers do not have a resource provisioning mechanism to satisfy user jobs with deadline requirements. Currently, it is solely the user's responsibility to estimate the required amount of resources for running a job in the cloud. This paper presents a Hadoop job performance model that accurately estimates job completion time and further provisions the required amount of resources for a job to be completed within a deadline. The proposed model builds on historical job execution records and employs Locally Weighted Linear Regression (LWLR) technique to estimate the execution time of a job. Furthermore, it employs Lagrange Multipliers technique for resource provisioning to satisfy jobs with deadline requirements. The proposed model is initially evaluated on an in-house Hadoop cluster and subsequently evaluated in the Amazon EC2 Cloud. Experimental results show that the accuracy of the proposed model in job execution estimation is in the range of 94.97 and 95.51 percent, and jobs are completed within the required deadlines following on the resource provisioning scheme of the proposed model.

INTRODUCTION:

Many organizations are continuously collecting massive amounts of datasets from various sources such as the World Wide Web, sensor networks and social networks. The ability to perform scalable and timely analytics on these unstructured datasets is a high priority task for many enterprises. It has become difficult for traditional network storage and database systems to process these continuously growing datasets. Map Reduce [1], originally developed by Google, has become a major computing model in support of data intensive applications. It is a highly scalable, fault-tolerant and data parallel model that automatically distributes the data and parallelizes the computation across a cluster of computers [2]. Among its implementations such as Mars[3], Phoenix[4], Dryad[5] and Hadoop [6], Hadoop has received a wide uptake by the community due to its open source nature. Building on the HP model, this system presents an improved HP model for Hadoop job execution estimation and resource provisioning. The major objectives of the system are as follows:

- The improved HP work mathematically models all the three core phases of a Hadoop job. In contrast, the HP work does not mathematically model the non-overlapping shuffle phase in the first wave.
- The improved HP model employs locally weighted linear regression (LWLR) technique to estimate the execution time of a Hadoop job with a varied number of reduce tasks. In contrast, the HP model employs a simple linear regress technique for job execution estimation which restricts to a constant number of reduce tasks.
- Based on job execution estimation, the improved HP model employs Lagrange Multiplier technique to provision the amount of resources for a Hadoop job to complete within a given deadline.

SYSTEM DESIGN:

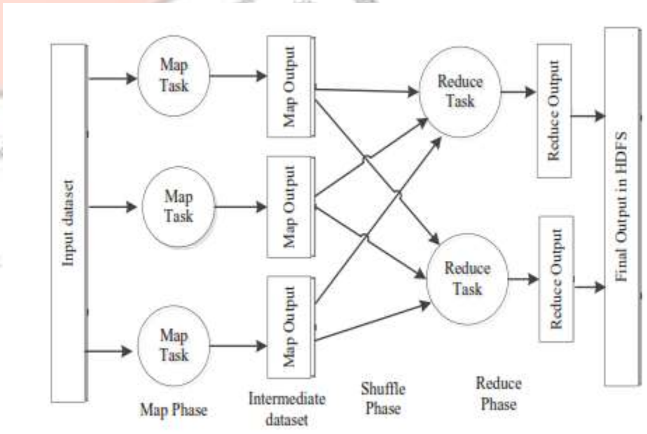


Fig.1. Hadoop job execution flow.

IMPLEMENTATION

3.1Modeling Map Phase

In this phase, a Hadoop job reads an input dataset from Hadoop Distributed File System (HDFS), splits the input dataset into data chunks based on a specified size and then passes the data chunks to a user-define map function. The map function processes the data chunks and produces a map output. The map output is called intermediate data

Modeling Shuffle Phase

In this phase, a Hadoop job fetches the intermediate data, sorts it and copies it to one or more reducers. The shuffle tasks and sort tasks are performed simultaneously, therefore, we generally consider them as a shuffle phase.

Modeling Reduce Phase

In this phase, a job reads the sorted intermediate data as input and passes to a user-defined reduce function. The reduce function processes the intermediate data and produces a final output. In general, the reduce output is written back into the HDFS Job.

Execution Estimation

When a job processes an increasing size of an input dataset, the number of map tasks is proportionally increased while the number of reduce tasks is specified by a user in the configuration file. The number of reduce tasks can vary depending on user's configurations. When the number of reduce tasks is kept constant, the execution durations of both the shuffle tasks and the reduce tasks are linearly increased with the increasing size of the input dataset as considered in the HP model. This is because the volume of an intermediate data block equals to the total volume of the generated intermediate data divided by the number of reduce tasks. As a result, the volume of an intermediate data block is also linearly increased with the increasing size of the input dataset. However, when the number of reduce tasks varies, the execution durations of both the shuffle tasks and the reduce tasks are not linear to the increasing size of an input dataset.

Resource Provisioning

The improved HP model presented in Section III can estimate the execution time of a Hadoop job based on the job execution profile, allocated resources (i.e. map slots and reduce slots), and the size of an input dataset. The improved HP model is further enhanced to estimate the amount of resources for Hadoop jobs with deadline requirements.
Consider a deadline for a job that is targeted at the lower bound of the execution time. To estimate the number of map slots and reduce slots, we consider the non-lapping map phase in the first wave, the map phase in other waves together with the overlapped shuffle phase in the first wave, the shuffle phase in other waves and the reduce phase.

Existin PROPOSED SYSTEM:

- Building on the HP model, this paper presents an improved HP model for Hadoop job execution estimation and resource provisioning. The major contributions of this paper are as follows
- The improved HP work mathematically models all the three core phases of a Hadoop job. In contrast, the HP work does not mathematically model the non-overlapping shuffle phase in the first wave.
- The improved HP model employs locally weighted linear regression (LWLR) technique to estimate the execution time of a Hadoop job with a varied number of reduce tasks. In contrast, the HP model employs a simple linear regress technique for job execution estimation which restricts to a constant number of reduce tasks.

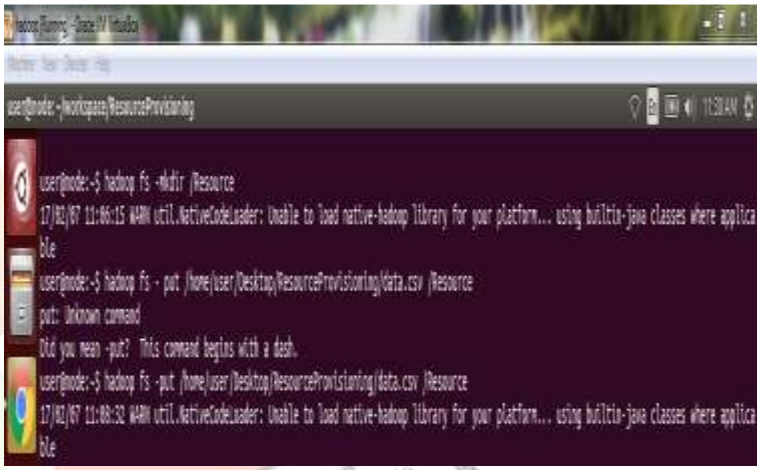
Based on job execution estimation, the improved HP model employs Lagrange Multiplier technique to provision the amount of resources for a Hadoop job to complete within a given deadline.

ADVANTAGES OF PROPOSED SYSTEM:

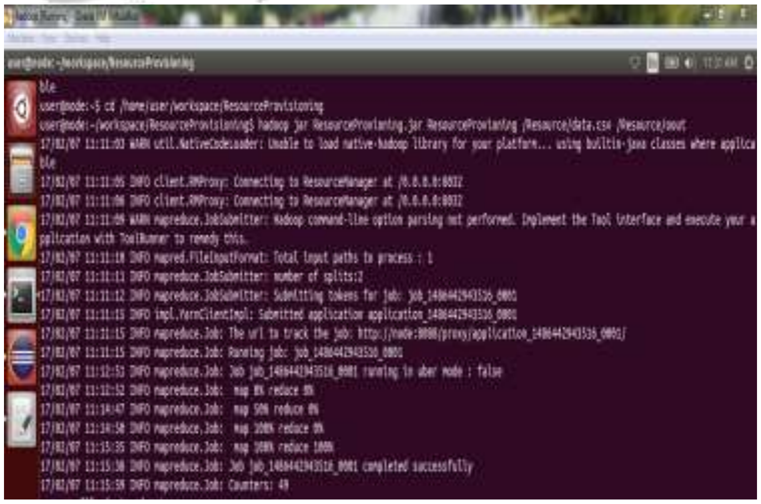
- The experimental results showed that the improved HP model outperforms both Starfish and the HP model in job execution estimation.
- Similar to the HP model, the improved HP model provisions resources for Hadoop jobs with deadline requirements.

The improved HP model is more economical in resource provisioning than the HP model.

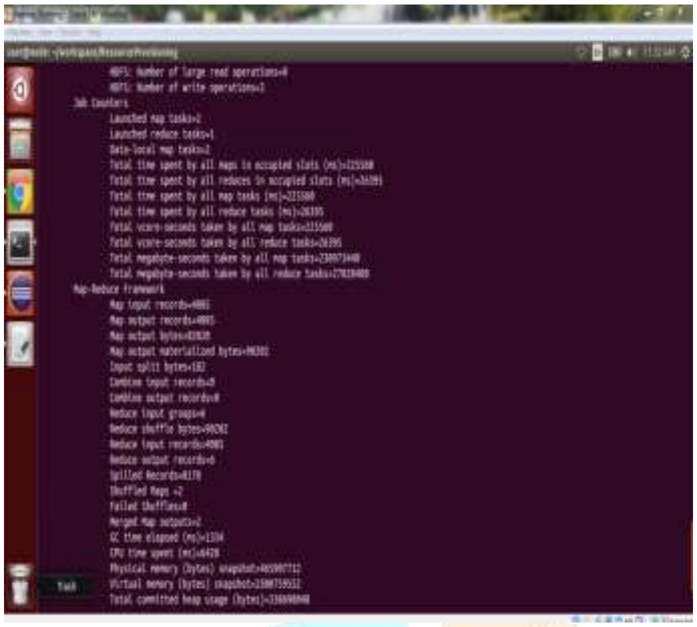
OUTPUT:



```
user@node:~$ hadoop fs -mkdir /Resource
17/02/07 11:06:15 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applica
ble
user@node:~$ hadoop fs - put /home/user/Desktop/ResourceProvisioning/data.csv /Resource
put: Unknown command
Did you mean -put?  This command begins with a dash.
user@node:~$ hadoop fs -put /home/user/Desktop/ResourceProvisioning/data.csv /Resource
17/02/07 11:08:32 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applica
ble
```

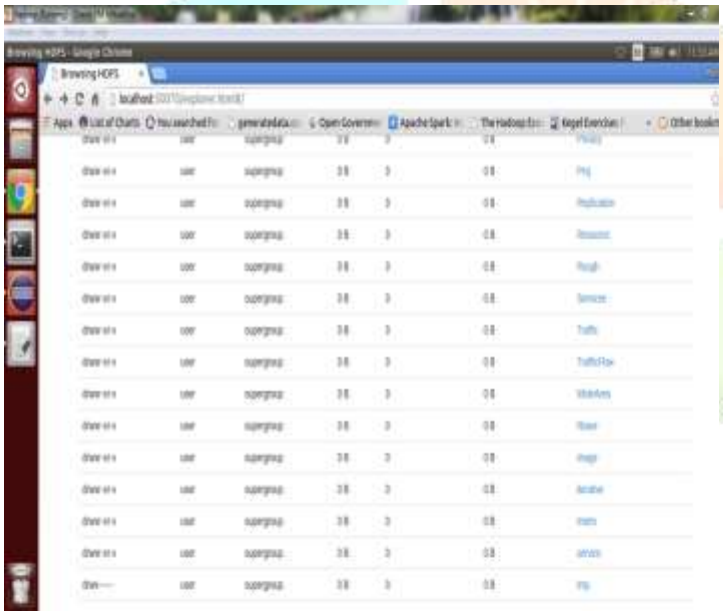


```
ble
user@node:~$ cd /home/user/workspace/ResourceProvisioning
user@node:~/workspace/ResourceProvisioning$ hadoop jar ResourceProvisioning.jar ResourceProvisioning /Resource/data.csv /Resource/oout
17/02/07 11:11:03 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applica
ble
17/02/07 11:11:05 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
17/02/07 11:11:06 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
17/02/07 11:11:09 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your a
pplication with ToolRunner to remedy this.
17/02/07 11:11:10 INFO mapred.FileInputFormat: Total input paths to process : 1
17/02/07 11:11:11 INFO mapreduce.JobSubmitter: number of splits:2
17/02/07 11:11:12 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1486442943536_0001
17/02/07 11:11:15 INFO impl.YarnClientImpl: Submitted application application_1486442943536_0001
17/02/07 11:11:15 INFO mapreduce.Job: The url to track the job: http://node:8088/proxy/application_1486442943536_0001/
17/02/07 11:11:15 INFO mapreduce.Job: Running job: job_1486442943536_0001
17/02/07 11:12:51 INFO mapreduce.Job: Job job_1486442943536_0001 running in uber mode : false
17/02/07 11:12:52 INFO mapreduce.Job:  map 0% reduce 0%
17/02/07 11:14:47 INFO mapreduce.Job:  map 50% reduce 0%
17/02/07 11:14:58 INFO mapreduce.Job:  map 100% reduce 0%
17/02/07 11:15:35 INFO mapreduce.Job:  map 100% reduce 100%
17/02/07 11:15:38 INFO mapreduce.Job: Job job_1486442943536_0001 completed successfully
17/02/07 11:15:39 INFO mapreduce.Job: Counters: 49
```

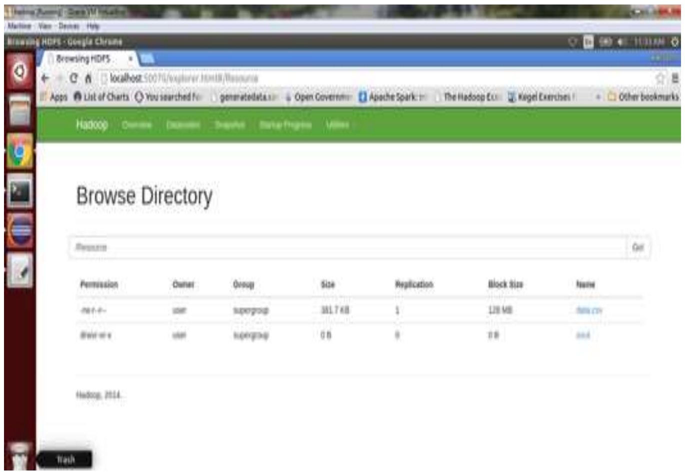


```
		HDFS: Number of large read operations=0
		HDFS: Number of write operations=2
	Job Counters
		Launched map tasks=2
		Launched reduce tasks=1
		Data-local map tasks=2
		Total time spent by all maps in occupied slots (ms)=225500
		Total time spent by all reduces in occupied slots (ms)=26395
		Total time spent by all map tasks (ms)=225500
		Total time spent by all reduce tasks (ms)=26395
		Total vcore-seconds taken by all map tasks=225500
		Total vcore-seconds taken by all reduce tasks=26395
		Total megabyte-seconds taken by all map tasks=230912000
		Total megabyte-seconds taken by all reduce tasks=27028480
	Map-Reduce Framework
		Map input records=4085
		Map output records=4085
		Map output bytes=82028
		Map output materialized bytes=90202
		Input split bytes=182
		Combine input records=0
		Combine output records=0
		Reduce input groups=4
		Reduce shuffle bytes=90202
		Reduce input records=4085
		Reduce output records=6
		Spilled Records=8170
		Shuffled Maps =2
		Failed Shuffles=0
		Merged Map outputs=2
		GC time elapsed (ms)=1334
		CPU time spent (ms)=6420
		Physical memory (bytes) snapshot=465907712
		Virtual memory (bytes) snapshot=2590759552
		Total committed heap usage (bytes)=338690048
```

Line
3D Bar
3D Pie
3D Donut
Resource Provision
job estimation



drwxr-xr-x	user	supergroup	0 B	0	0 B	[illegible]
drwxr-xr-x	user	supergroup	0 B	0	0 B	Proj
drwxr-xr-x	user	supergroup	0 B	0	0 B	Replication
drwxr-xr-x	user	supergroup	0 B	0	0 B	Resource
drwxr-xr-x	user	supergroup	0 B	0	0 B	Rough
drwxr-xr-x	user	supergroup	0 B	0	0 B	Service
drwxr-xr-x	user	supergroup	0 B	0	0 B	Traffic
drwxr-xr-x	user	supergroup	0 B	0	0 B	TrafficFlow
drwxr-xr-x	user	supergroup	0 B	0	0 B	[illegible]
drwxr-xr-x	user	supergroup	0 B	0	0 B	[illegible]
drwxr-xr-x	user	supergroup	0 B	0	0 B	image
drwxr-xr-x	user	supergroup	0 B	0	0 B	[illegible]
drwxr-xr-x	user	supergroup	0 B	0	0 B	matrix
drwxr-xr-x	user	supergroup	0 B	0	0 B	[illegible]
drwx------	user	supergroup	0 B	0	0 B	tmp



Browse Directory

Permission	Owner	Group	Size	Replication	Block Size	Name
-rw-r--r--	user	supergroup	381.7 KB	1	128 MB	data.csv
drwxr-xr-x	user	supergroup	0 B	0	0 B	out

Hadoop, 2014.

CONCLUSION:

Running a MapReduce Hadoop job on a public cloud such as Amazon EC2 necessitates a performance model to estimate the job execution time and further to provision a certain amount of resources for the job to complete within a given deadline. This paper has presented an improved HP model to achieve this goal taking into account multiple waves of the shuffle phase of a Hadoop job. The improved HP model was initially evaluated on an in-house Hadoop cluster and subsequently evaluated on the EC2 Cloud. The experimental results showed that the improved HP model outperforms both Starfish and the HP model in job execution estimation. Similar to the HP model, the improved HP model provisions resources for Hadoop jobs with deadline requirements. However, the improved HP model is more economical in resource provisioning than the HP model. Both models over-provision resources for user jobs with large deadlines in the cases where VMs are configured with a large number of both map slots and reduce slots. One future work would be to consider dynamic overhead of the VMs involved in running the user jobs to minimize resource over-provisioning. Currently the improved HP model only considers individual Hadoop jobs without logical dependencies. Another future work will be to model multiple Hadoop jobs with execution conditions.

FUTURE SCOPE:

Running a MapReduce Hadoop job on a public cloud such as Amazon EC2 necessitates a performance model to estimate the job execution time and further to provision a certain amount of resources for the job
to complete within a given deadline. It has presented an improved HP model to achieve this goal taking into account multiple waves of the shuffle phase of a Hadoop job. The improved HP model was initially
evaluated on an in-house Hadoop cluster and subsequently evaluated on the EC2 Cloud. The experimental results showed that the improved HP model outperforms both Starfish and the HP model in job execution estimation. Similar to the HP model, the improved HP

model provisions resources for Hadoop jobs with deadline requirements. However, the improved HP model is more economical in resource provisioning than the HP model. Both models over-provision resources for user jobs with large deadlines in the cases

where VMs are configured with a large number of both map slots and reduce slots. One future work would be to consider dynamic overhead of the VMs involved in running the user jobs to minimize resource over-provisioning. Currently the improved HP model only considers individual Hadoop jobs without logical dependencies.

REFERENCES:

[1] J. Dean and S. Ghemawat, “MapReduce: Simplified data processing on large clusters,” Commun. ACM, vol. 51, no. 1, pp. 107–113, 2008.

[2] R. L€ammel, “Google’s MapReduce programming model—Revisited,” Sci. Comput. Programm., vol. 70, no. 1, pp. 1–30, 2008.

[3] B. He, W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang, “Mars: A MapReduce framework on graphics processors,” in Proc. 17th Int. Conf. Parallel Archit. Compilation Techn., 2008, p. 260.

[4] K. Taura, T. Endo, K. Kaneda, and A. Yonezawa, “Phoenix: A parallel programming model for accommodating dynamically joining/leaving resources,” ACM SIGPLAN Notices, vol. 38, no. 10, pp. 216–229, 2003.

[5] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, “Dryad: Distributed data-parallel programs from sequential building blocks,” ACM SIGOPS Oper. Syst. Rev., vol. 41, no. 3, pp. 59–72, Mar. 2007.

[6] Apache Hadoop [Online]. Available: http://hadoop.apache.org:/(Last accessed: 1 March 2015).