MRSim: A Discrete Event based MapReduce Simulator

Suhel Hammoud, Maozhen Li, Yang Liu, Nasullah Khalid Alham, Zelong Liu
School of Engineering and Design
Brunel University
Uxbridge, UB8 3PH, UK
Suhel.Hammoud@brunel.ac.uk

Abstract— **Recently MapReduce programming model is becoming popular for large scale data intensive distributed applications due to its efficiency, simplicity and ease of use. The Hadoop implementation of MapReduce is one of the most popular tools for many programmers due its ability to hide details of parallel programming from the users. However work on simulating the Hadoop environment is still in its infancy. Although there are a large number of simulating tools available to simulate distributed environments. However there are only a few simulators available which specifically targets the MapReduce environment. Based on testing we performed; the usability of *these simulators* is not satisfactory due to the simplified design which limits simulating jobs with variance configurations. We have designed and implemented a MapReduce simulator based on discrete event simulation called *MRSim* which accurately simulate the Hadoop environment. The simulator on one hand allows us to measure scalability of MapReduce based applications easily and quickly, on the other hand captures the effects of different configurations of Hadoop setup on MapReduce based applications behavior in terms of Hadoop job completion times and hardware utilization.**

Keywords-hadoop ; MapReuce; simulation

I. INTRODUCTION

MapReduce is a simple parallel programming framework for data intensive applications. MapReduce programming model provides an efficient framework for processing large data sets in an extremely parallel manner [1]. The Google File System (GFS) that underlie a MapReduce provide efficient and reliable distributed data storage required for applications involving large data sets [2]. The basic function of MapReduce model is to iterate over the input, compute key/value pairs from each part of input, group all intermediate values by key, then iterate over the resulting groups and finally reduce each group. The programmer can abstract from the issues of distributed and parallel programming. MapReduce implementation deals with issues such as load balancing, network performance, fault tolerance etc [1]. The Apache Hadoop [3] project is an open-source implementation of Google's MapReduce writing in java for reliable, scalable, distributed computing.

Recently there have been many applications adapted to the MapReduce model; however these applications are tested with small and medium size clusters of participating nodes. Having such clusters of nodes it is difficult to measure the scalability of the algorithms. Scalability is a measure of efficiency of an algorithm with a much larger cluster i.e. hundreds of nodes as well using a much larger training dataset. It is almost impractical to set up a very large cluster consisting hundreds or thousands of nodes to measure the scalability of an algorithm. Hadoop environment set up involves alterations of a great number of parameters which are crucial to achieve best performances. An obvious solution to the above problems is to use a simulator which can simulate the Hadoop environment; a simulator on one hand allows us to measure scalability of MapReduce based applications easily and quickly, on the other hand determines the effects of different configurations of Hadoop setup on MapReduce based applications behavior in terms of speed.
Hadoop is offered as a service on Amazon EC2 where users can launch and terminate instances on demand and pay by the hour for active instances. Also Hadoop cluster can be integrated in Sun Grid Engine. Thus, MapReduce simulators will be very useful utilities to allow users to estimate times and the costs for the jobs before submitting it to EC2 or Sun Microsystems Grid Compute Utility.

We have designed and implemented a MapReduce simulator called MRSim to simulate the behavior of our new developed data mining algorithms on Hadoop environment. This with the real implementation of the algorithms will help us to find the pest values of parameters to tune the cluster for the best performance. MRSim is extending discrete event engine used SimJava [5] to accurately simulate the Hadoop environment. Using SimJava we simulate interactions between different entities within cluster. GridSim [6] package is also used for network simulation. It is written in Java programming language on top of SimJava. Evaluation of MRSim was performed using a number MapReduce based applications which we have implemented recently. Evaluation results show high level of accuracy from different aspects.

II. RELATED WORKS

Although there are a large number of simulating tools available to simulate distributed environments. However to the best our knowledge there are only a two or three simulators available which specifically targets the Hadoop environment, because MapReduce is a fairly new programming model .One of the most recent is called *MRPrf* [4]. Based on testing we performed on the *MRPrf* code released recently; we could not generate accurate results for jobs of different type of algorithms or different cluster configurations.

978-1-4244-5934-6/10/$26.00 ©2010 IEEE

Cardona et al [7] implemented a simulator to simulate MapReduce jobs. They focus on simulating map and reduce functions as well as HDFS in order to simulated different scheduling algorithms. However we think their simulator cannot accurately simulate wide range of MapReduce based applications due to lack of consideration of a number important parameters. Such as interaction between memory buffers, using combiner, background merging and shuffling processes in Map/Reduce tasks . Mumak [8] is another implementation of Hadoop environment simulator. Mumak uses data from real experiment to estimate completion time for Map and Reduce tasks with different scheduling algorithms. In cases which data from real experiments do not exist, Mumak cannot estimate completion time for Map and Reduce tasks. Wang et al [4] introduce a simulation approach to analyze performance of MapReduce setups such as node, rack and network configurations, disk parameters and performance, data layout and application I/O characteristics, etc and uses this information to predict expected application performance. However they use abstraction in a number in occasions by overlapping processes to estimate times which could result in low level of accuracy in simulator' predictions.

In contrast to the existing MapReduce simulators, MRSim simulates much more details; MRSim simulates shared Multi core CPUs, in addition to HDD and network traffic. Cluster configuration setting is considered such as memory buffers, merge parameters, parallel copy and sort parameters, combiners etc. Each of the above parameter affects the performance considerably. Finally MRSim uses the above information to accurately predict the performance of a MapReduce based application.

III. DESIGN

MRSim is simulating Hadoop [3] implementation of MapReduce framework which is described by White [9]. MRSim model and simulates network topology and traffic using GridSim. On the other hand, it models the rest of system entities using SimJava discrete event engine.
The System is designed using object oriented based models. CPU, HDD and Network Interface models were designed to be the basic blocks which can be grouped in PC machine entity as shown in figure1. Each machine is part Network Topology model. Each machine can host Job Tracker process and Task Tracker Process. However there is only one Job Tracker per MapReduce Cluster. Each Task Tracker Model can launch several Map and Reduce tasks up to the maximum allowed number in the configuration files.
MRSim is using very basic simulation for Hadoop distributed file system HDFS it simulates data splits locations and splits replication in local rack only. However this is satisfactory to estimate the effect of replication on the job performance in one rack clusters.

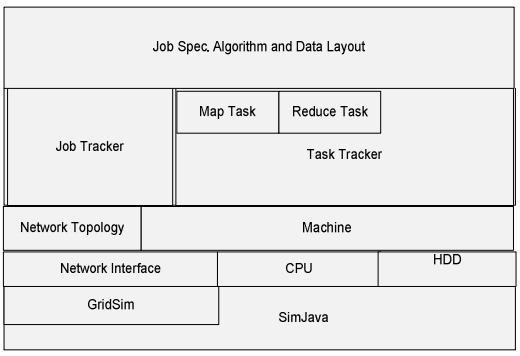

Figure 1. MRSim Architercure

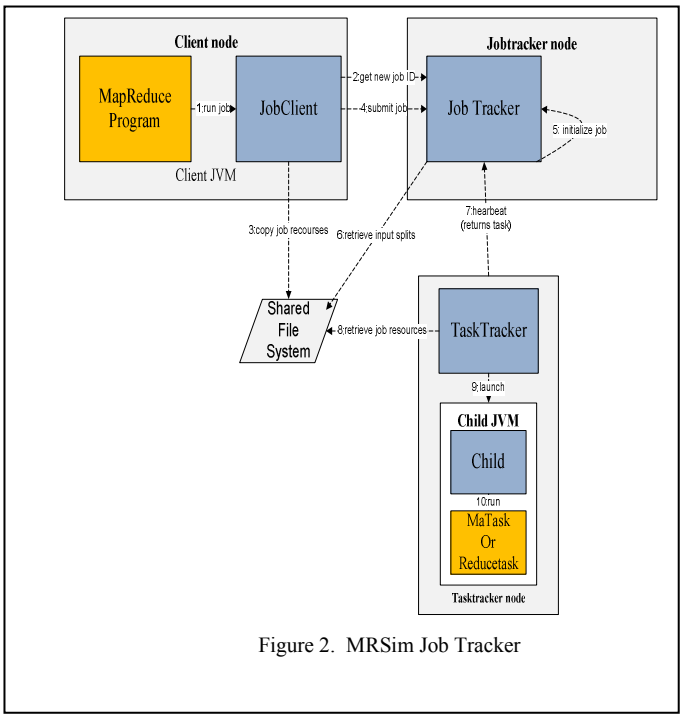

Figure 2. MRSim Job Tracker

Figure 2 shows how Hadoop runs a MapReduce job

A. *Simulating Job Tracker*

The main components of the simulator is Job Tracker that controls generating map and reduce tasks, monitors when different phases complete, and producing the final results as shown in figure 2. Job Tracker is using heartbeat signal to ensure the liveliness of nodes and the tasks launched in the nodes.

B. *Simulating map task*

Map task is started by Job Tracker. The following processes take place;

- A Java VM is instantiated for the task.
- Data is read from the local disk or requested remotely.
- Map, sort, and spill operations are performed on the input data until all of it has been consumed.

- Background file system mergers are merging the output data to reduce the number of output files to one or few files.
- A message indicating the completion of the map task is returned to the Job Tracker.

C. Simulating reduce task

Reduce task starts by receiving a message from the Job Tracker. The following processes take place;

- A once a map task complete several reducers try to get the Map output using several shuffling processes per reducer.
- Data are shuffled to reducer memory buffer the data sizes are less than defined thresholds in job specification. Background in-memory mergers and local file system mergers will try to merge the intermediate data every time group of conditions occurs. These conditions are controlled by job specs and triggered when memory buffers and number of current shuffled files reach the thresholds.
- After shuffling phase is complete the mergers will do one final merge and sort on the data. Then the data will be fed to the reduce function.
- Reduces function processes the data and write the output to distributed file system.
- A message signifying the completion of the reduce task is sent to the Job Tracker.

D. Input sepcification

The user input needed by *MRSim* is divided into two different parts defined in text files of Json format: cluster topology file and job specification file. Topology file consists of several rack parts. Each rack consists of group of machines linked with one router.
Figure 4 shows an example of topology file for one rack of several machines.

```
{ "machines" : [
        { "hostName" : "machine 1",
          "maxMapper" : 10,
          "maxReducer" : 5,
         "baudRate" : 70000000.0,
          "cpu" : { "cores" : 4,   "speed" : 500000000.0   },
          "hardDisk" : { "capacity" : 400000000.0,
              "read" : 40000000.0, "seekTime" : 1.0,
             "write" : 20000000.0
           }
            },
        { "hostName" : "machine 2", ...},
        { "hostName" : "machine 3", ...},
       ....
       ..
       ],
    "router" : "r_01",
    "heartbeat":1.0,
    "propDelay":1.0,
     "maxIM":60000,
     ...
    ..
   }
```

Figure 4. MRSim Topology File

Job specifications file comprises of number of map and reduce tasks, data layout, algorithm description and the job configuration parameters. The data layout offers the location and the replications of the data splits in on the simulated nodes. Algorithm provides information about the MapReduce application, such as number of CPU instructions per record, average record size in MapReduce tasks.

```
{
  "jobName":"201005041935_0032-1,620,000-a",

  "numberOfMappers":60,
  "numberOfReducers":1,
  "data":{
    "name":"data_1",
    "size":4.78627929E8,
    "records":1620000.0,
    "replica":["machine 1","machine 5"]
  },

"algorithm":{
     "mapCost":10000.0,       "mapRecords":50.0,
     "mapOutAvRecordSize":12.0,

     "combineCost":80.0,    "combineRecords":1.0,
     "combineGroups":100000.0,
     "combineOutAvRecordSize":1.0,

     "reduceCost":80.0 , "reduceRecords":0.01,
     "reduceOutAvRecordSize":10.9,


   ....
....
 },
  "ioSortFactor":10,
  "ioSortMb":100.0,
  "ioSortRecordPercent":0.05,
  "ioSortSpillPercent":0.8,
  "mapredChildJavaOpts":200,
  "mapredInmemMergeThreshold":1000,
  "mapredJobReduceInputBufferPercent":0.0,
  "mapredJobShuffleInputBufferPercent":0.7,
  "mapredJobShuffleMergePercent":0.66,
  "mapReduceParallelCopies":5,
  "useCombiner":false,
  "useCompression":false
  "replication":3,
...
}
```

Figure 5. MRSim Job Specification File

More job specifications is provided to control the memory buffer sizes and thresholds for the map and the reduce tasks, number of background spill, shuffle in-memory mergers and local file system mergers process. All these processes are working in parallel with the map and reduce functions.

IV. EVALUATION

The following experiment results were collected in single rack cluster, consisting of four participating nodes. Three of nodes are with Intel CPU Q6600, 3GB RAM and Fedora 12 OS, the fourth one with Intel Core 2 Duo T7300, 4GB RAM and Fedora 12 OS. The experiments focused on job execution times, average task times in every job, intermediate spilled records in Map and Reduce tasks and the amount of HDD read and written for each job. Datasets used with different sizes are used. Algorithm used in the test is word count. Each job consists of 60 Maps and One reducer.

A. Spilled Records:

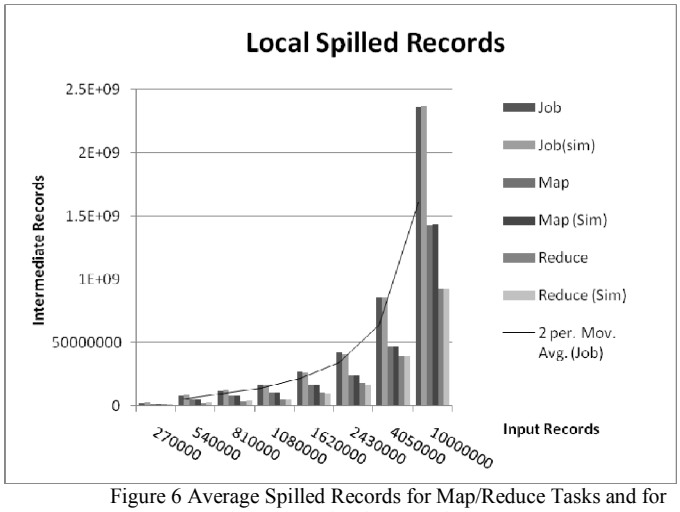

Figure 6 Average Spilled Records for Map/Reduce Tasks and for Jobs. (Sim: Simulator results)

As shown in Figure 6. The average of spilled records increases exponentially when the input data increases. This is because the number of times to flush the memory buffers to the HDD is increasing while the local file system mergers is allowed to merge at most fixed number of files at one. This fixed number is defined by *io.sort.factor* parameter. This means the mergers will recursively remerge the remaining files with already merged files before. More performance is gained by reducing the number of spilled records in Map and Reduce tasks. This is done by increasing the percentage of memory buffers form the virtual memory of the task or by increasing *io.sort.mb* parameter. Parameter *io.sort.factor* can not be increased too much because HDD is showing less overall read and write speed if total number of parallel reads and writes increases.

B. Local File System Reads and Writes.

The intermediate data sizes read or written in Map/Reduce tasks play an important role in task performance. While Data sizes read by Map tasks from to distributed file system or written by reducers to DFS can be calculated without the need of the simulator. The local HDD file reads and writes is affected by the job configuration and by the sequence of events occurs during the job. The exact event sequence can not be predicted without simulation. Experiments shows number of in-memory merges vary if the shuffling speed in reduces varies. DFS reads/writes are fixed per algorithm. Local read/writes varies if job configuration changes.

Again the average of local file reads/writes is increasing exponentially. To reduce it allocates more memory buffers sizes for Map and Reduce tasks. For example, average Map reads for data input size 270000 records is 0. This is because the Map task processed small amount of data. All the Map output can fit in memory buffer and no need for intermediate HDD spills and merges. Figure 7. Show number of bytes read/written for jobs varies in input sizes.

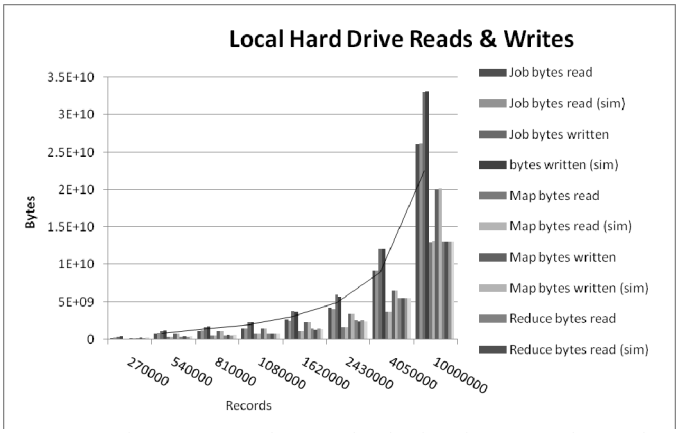

Figure 7 Average bytes read and written in Map, Reduce Tasks and for Jobs. (Sim: Simulator results)

C. Map , Reduce and Job Execution times

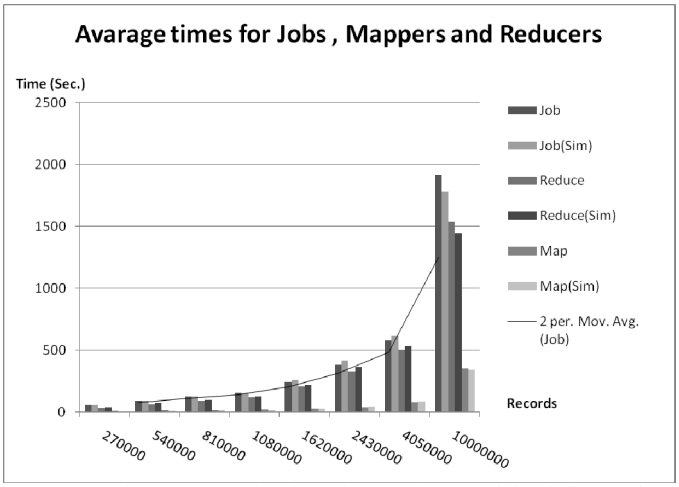

Figure 8. Execution times for experiment and simulation results.

As shown in Figure 8. Average times for Map / Reduce tasks and for the Jobs follow the same pattern of increasing when input data size increases. MRSim simulator showed very good accuracy because it depends on simulating accurately the data flow between the Map and Reduce Tasks. Figure 9 shows

times of the simulated jobs compared to the real time taken form the experiments.

Records	Job	Reduce	Map
270,000.00	101.75%	110.00%	81.00%
540,000.00	100.00%	113.48%	77.40%
810,000.00	100.80%	106.68%	88.14%
1,080,000.00	98.08%	105.79%	65.87%
1,620,000.00	104.86%	106.28%	101.03%
2,430,000.00	108.57%	109.41%	112.67%
4,050,000.00	105.82%	105.73%	108.60%
10,000,000.00	93.16%	93.83%	97.73%

Figure 9 Execution times ratio (MRSim value)/(experiments Value)

V. CONCLUSIONS AND FURTHER WORKS

In this paper we implemented MRSim, a Map Reduce simulator based on discrete event simulation. We performed a number of experiments to evaluate MRSim; MRSim is able to simulate different type of Map Reduce applications with the ability to study with good accuracy the effect of dozens of Job configuration parameters on the job performance. MRSim enables the user to estimate the best job configurations to get optimum performance for certain algorithm. On the other hand, MRSim will capture the effects of different configurations of Hadoop setup on the algorithm's behavior in terms of speed and hardware utilization. As part of future research work we are planning to introduce an advanced load balancing technique targeting the Hadoop environment in order to achieve optimal resource utilization hence shorter Hadoop job completion time.

REFERENCES

[1] R. Lämmel, "Google's MapReduce programming model — Revisited," *Sci. Comput. Program.*, vol. 68, 2007, pp. 208-237.

[2] J. Venner, *Pro Hadoop*, Springer 2009, pp 1-89.

[3] Apache Hadoop! Available at: http://hadoop.apache.org/ [Accessed November 2, 2009

[4] G. Wang, A.R. Butt, P. Pandey, and K. Gupta, "Using realistic simulation for performance analysis of mapreduce setups," *Proceedings of the 1st ACM workshop on Large-Scale system and application performance*, Garching, Germany: ACM, 2009, pp. 19-26.

[5] SimJava. Available at: http://www.dcs.ed.ac.uk/home/hase/simjava/ [Accessed April 21, 2010].

[6] GridSim: A Grid Simulation Toolkit | Get GridSim: A Grid Simulation Toolkit at SourceForge.net. Available at: http://sourceforge.net/projects/gridsim/ [Accessed April 21, 2010].

[7] J. Cardona, M. Georgiopoulos, G. Anagnostopoulos, "A Grid Based System for Data Mining Using MapReduce," 2007; http://cygnus.fit.edu/amalthea/pubs/Cardona_Secretan_TR-2007-02_AMALTHEA.pdf.

[8] Mumak: Map-Reduce Simulator - ASF JIRA. Available at: https://issues.apache.org/jira/browse/MAPREDUCE-728 [Accessed April 21, 2010].

[9] T. White, "Hadoop The Definitive Giude", OReiry, Yahoo Press, pp 1-172.

[10] JSON. Available at: http://www.json.org/ [Accessed April 21, 2010].

[11] MRSim. Map Reduce Simulator: http://code.google.com/p/mrsim/ [Accessed May 10, 2010]