A MR Simulator in Facilitating Cloud Computing

Prof R. Palson Kennedy
Department of Computer Science & Engineering ,
Anna University
Chennai, Tamil Nadu, 600025 , India

T.V.Gopal,Ph.D
Department of Computer Science & Engineering,
Anna University, CEG
Chennai , Tamil Nadu,600025 , India

ABSTRACT

MapReduce is an enabling technology in support of Cloud Computing. Hadoop which is a mapReduce implementation has been widely used in developing MapReduce applications. This paper presents Hadoop simulator- HaSim, MapReduce simulator which builds on top of Hadoop. HaSim models large number of parameters that can affect the behaviors of MapReduce nodes, and thus it can be used to tune the performance of a MapReduce cluster. HaSim is validated with both benchmark results and user customized MapReduce applications.

Keywords: MapReduce Hadoop framework, Cloud Computing, HaSim.Simulator Programming models

1. INTRODUCTION

MapReduce [1] is a distributed programming model for data intensive tasks which has become an enabling technology in support of Cloud Computing. Programmatically inspired from functional programming, at its core there are two primary features, namely a map and a reduce operation. From a logical perspective, all data is treated as a Key (K), Value (V) pair. Multiple mappers and reducers can be employed. At an atomic level however a map operation takes a {K1, V 1} pair and emits an intermediate list {K 2, V 2} pairs. A reduce operation takes all values represented by the same key in the intermediate list and processes them accordingly, emitting a final new list {V 2}. Whilst the execution of reduce operations cannot start before the respective map counterparts are finished, all map and reduce operations run independently in parallel. Each map function executes in parallel emitting respective values from associated input. Similarly, each reducer processes keys independently and on currently.

Fig. 1 shows the structure of the MapReduce model. Popular implementations of the MapReduce model include Mars [3], Phoenix [2], Hadoop [1,9] and Google's implementation [6]. Among them, Hadoop has become the most popular one due to its open source feature. However, the large number of configuration parameters of Hadoop brings forth a number of challenges to users.Hadoop application, it is hard to decide on a set of parameters that would help to achieve a good performance, e.g. the number of mappers, the number and the CPU speeds of nodes, and the size of buffers. It would be extremely difficult if not impossible to set up a physical Hadoop environment to evaluate the scalability of a Hadoop application up to a few hundred or even thousand nodes.

These challenges make it a necessity to have a Hadoop simulator in place where it can be used to tune the performance of a Hadoop cluster and to study the behaviors of Hadoop applications. It should be pointed out that few existing MapReduce simulators are available and MRPerf [7,8] is a representative one. However, the accuracy of MRPerf in simulating Hadoop environments is limited to simple behaviors. This paper presents the design and implementation of HaSim, a MapReduce simulator for Hadoop applications.

The key contributions of HaSim lie in its high accuracy in simulating the dynamic behaviors of Hadoop environments and the large number of Hadoop parameters that can be modeled in the simulator.

• Node parameters, which are related to processors, memory, hard disk, network interface, Map and Reduce instances.
• Cluster parameters, which include the number of nodes, node configurations, network routers, job queues and schedulers.
• Hadoop system parameters, which include the size of data chunks, JVM reuse, sort factor, virtual memory, the number of copying threads, data spilled threshold.
• HaSim simulator parameters including simulation speed, system clock, accuracy levels, and system reporter.

The accuracy of HaSim is extensively validated following a two step process. In the first step, HaSim is validated against an authoritative benchmark work.

In the second step, the behaviors of HaSim are evaluated in comparison with that of a physical hadoop cluster using two Hadoop applications. The comparative results show high accuracy and stability of HaSim in simulating the behaviors of Hadoop environments. Using HaSim, the impacts of a number of parameters on Hadoop behaviors are further evaluated.

The rest of this paper is organized as follows. Section 2 reviews some related work in Hadoop simulation. section 3 presents the modeling work on Hadoop parameters

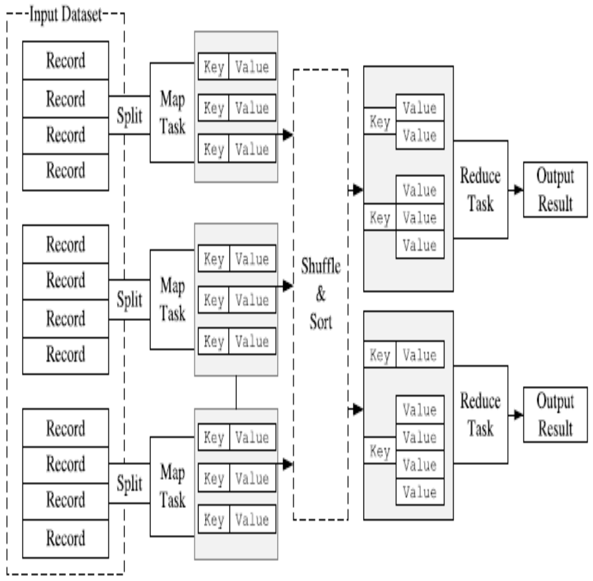

Fig. 1. The MapReduce model.

Section 4 describes in detail the design of HaSim. Section 5 validates HaSim and subsequently evaluates the impacts of a number of parameters n Hadoop behaviors. Section 6 concludes the paper and points out some future work.

2. RELATED WORK .

As mentioned in Section 1, MRPerf is one of the few existing MapReduce simulators that are available. MRPerf can serve as a design tool for MapReduce infrastructure, and as a planning tool for making MapReduce deployment far easier via reduction in the number of parameters that currently have to be manually tuned. From the published testing results, RPerf shows its high accuracy in simulating the impacts of network topologies due to its adoption of NS2 [8] for network simulation. However it should be pointed out that although MRPerf achieves high accuracy in simulating behaviors related to the underlying networks, it can simulate limited behaviors of the Hadoop framework. The behaviors of Hadoop are affected by a large number of parameters. The major limitations of MRPerf are listed below:

• The processing resources for each user are fixed in MRPerf. However, resources in a Hadoop environment are dynamically changing and are usually shared by a number of users dynamically.

• MRPerf cannot simulate the exact behaviors of Map and Reduce phases. In a Map instance, the spilled data will be kept writing into buffer while Map task is running. When the occupied size of the buffer is less than a certain threshold, the in-memory data is also kept spilling into hard disk simultaneously. Due to the highly uncertain real time states of the system, this mechanism significantly affects the number of spilled files which will further affect the IO behaviors. MRPerf simply ignores these procedures and uses a pre-defined data value.

• If the occupied size of the buffer is larger than a certain threshold, the CPU processing will be blocked until the whole content in buffer is flushed. This event can also affect system behaviors but MRPerf does not consider this.

•In the Reduce phase, MRPerf still performs a simple simulation to start reduce tasks simultaneously due to lack of accurate simulations in Map phase.

The limitations of MRPerf motivated the work on HaSim. Our focus in HaSim is to accurately simulate the behaviors of Hadoop framework. Using HaSim, the performances of Hadoop applications can be studied from a number of angles including the impacts of the parameters on the performance of a Hadoop cluster, the scalability of a Hadoop application in terms of the number of nodes used, and the impact of using heterogeneous environments.

3 MODELLING HADOOP PARAMETERS

The performance of a Hadoop application can be affected by a large number of parameters.In this section, we present the modeling work on these parameters.

3.1. Node parameters

• Processor: HaSim supports one processor per computer by default design, but the number of processors could be changed. One processor can have one or more cores. The processing speed of a processor core is defined as the volume of data units processed per seconds which can be measured from real experimental tests.

• Hard disk: In hard disk entity, the speeds of IO operations vary from time to time. Several parameters are introduced to build the degressive reading/writing model. Let xmax represent the maximum reading/writing speed of hard disk. From the experimental results of testing Seagate Barracuda 1 TB hard disk xmax is about 120 MB/s in reading, and 60 MB/s in writing. Let xmin represent the minimum reading/writing speed of hard disk, xmin is around 55 MB/s in reading and 25 MB/s in writing. Another parameter which is degressive factor r is used to represent in each second the value of lost speed. The value of the factor is around 0.0056 based on experimental tests. Using these parameters we can calculate the real time speed x of hard disk using formula (1).

$$X = \frac{XminXmax}{(Xmin-Xmax)e^{-rt} + Xmax} \quad \text{----------------(1)}$$

• Memory: In each memory entity two parameters are modeled, reading and writing. In the experimental tests, the reading speed of standard DDR2-800 memory with dual channel could reach up to 6000 MB/s and the writing speed is up to 5000 MB/s. It is quite obvious that both the reading and writing speeds would not be the bottlenecks of the system due to their fast speeds.

• Ethernet adapter: In each Ethernet adapter entity, two parameters are modelled, upstream bandwidth and downstream bandwidth. The bandwidth can be in the range of 100 and 1000 Mbps.

3.2. Cluster parameters .The cluster parameters represent the details of a simulated Hadoop cluster. It involves several aspects which include the number of nodes, topology and network facilities.

• Number of nodes: The number of nodes can vary from 1 to a few hundreds.

• Topology: The number of nodes can be organized with a certain network topology.

Currently HaSim only supports simple racks.

• Network facilities: The speed of a router can be in the range of 100 and 1000 Mbps. When the bandwidth of a router is defined, a number of standalone computers must be configured to connect to the router to decide on their network capacities.

• Job queue and job schedulers: A job queue holds the waiting job entities. According to different job schedulers, jobs are waiting for processing resources. HaSim supports two job schedulers of Hadoop framework—first come first serve and fair scheduler. These two types of schedulers generate different job processing orders.

3.3. Hadoop system parameters. Before a Hadoop application starts processing data, the data should be saved into Hadoop Distributed File System (HDFS) in advance. The number of files affects the number of Map instances involved.

Normally the number of Map instances equals to the number of file chunks. If the number of chunks is larger than the maximum number of Map instances in the cluster, Map instances will be assigned with data chunks via waves. If a whole dataset is only saved in one file, the single file could be separated into a number of chunks logically via supplied APIs of the Hadoop framework. When data is being processed, it would go through a number of processing steps such as sorting, merging, combining, copying, reducing. These steps highly affect the performances of the system so that several parameters are modelled to control the behaviours of these steps. As these parameters are configurable and most of them are involved in the actual Hadoop framework so we named these parameters hadoop system parameters.

• Job specifications: In a job specification, a number of parameters are involved to describe the properties of a job. Job ID refers to the unique ID assigned to each job for tracking. The JobSize is the total size of the input data.

No matter no many chunks of the data are submitted, this value should be the total size of the hole data. when the simulation starts, the data will be fetched from the HDFS. The NumberOfRecords parameter is used to represent the number of records in the data so that the size of each record can be calculated by this value and the size of the job. In the simulator this parameter is experimentally used to measure the number of records combined by Combiners, which may affect the erformances of thesystem when certain types of Hadoop applications are executed. The MapOutputRatio parameter represents the volume of intermediate data to be generated by Map instances which has an impact on IO performance. The ReduceOutputRatio parameter is quite similar to apOutputRatio. In ome Hadoop applications the Reduce instances do not only copy data from Map instances but also generate their own intermediate data which affects IO performance. This parameter specifies the size of intermediate data to be generated in the Reduce phase. The ReducingRatio parameter represents the size of final results which will be reduced in HDFS. This parameter can affect the performance of the underlying network and also the performance of a local hard disk. The Number of Chunks parameter is used to specify the number of files to be used to carry data. This parameter determines the number of Map instances assigned to the job. If the number of chunks is only one, a number of logically separated files should be specified. The Number of Reducers parameter represents the number of required Reduce instances for the job. Simulated Hadoop parameters: This group of parameters is highly related to Hadoop framework. The io.sort.mb parameter represents the size of memory buffer to use while sorting map output. The io.sort.record.percent parameter represents the proportion of io.sort.mb reserved for storing record boundaries of the map output results. The remaining space is used for the map output records themselves. The io.sort.spill.percent parameter is a threshold that determines when the Map instance should start spilling processes writing data into memory.

If the threshold is reached, the CPU processing will be suspended and the buffer will be flushed, which means all the data saved in virtual memory will be spilled into hard disk. The io.sort.factor one parameter specifies the maximum number of streams to merge when sorting files in the Map phase. It significantly affects the IO performance of the system.Therefore the sorting and merging involve less overhead generated by hard disk. The JVM Reuse parameter is partially simulated in HaSim. Using JVM reuse, the overhead generated by some short-lived tasks will be significantly reduced.

3.4 HaSim Parameters

HaSim itself needs several parameters to control its own behaviors. Five important parameters are introduced in HaSim:

System Clock: The System Clock parameter is an absolutely and continuously timing component. In each change of the system clock, its current value will be added by one second. It is used to record the current system time, and to measure the performances of Hadoop applications in different cluster configurations.

Executing Speed: This parameter controls the execution speeds of all the components in HaSim.

Accuracy Level: For normal Hadoop applications, it is enough to set this parameter to the level of seconds. To maintain high accuracy in simulation, milliseconds can be set for the applications as well.

Shared Parameters: These parameters can control the rates of the shared resources include hard disk and bandwidth. The ratio is defined by R

R= Assigned Resource /Total Resource.

Reporter: This parameter records several important system states for analysis.

S.No	Category	Specification
1	Node Parameters	processor, hard disk, memory, Ethernet card, Map instance, Reduce instance
2	Cluster Parameters	number of nodes, topology, network facilities, job queue, job scheduler
3	Hadoop System Parameters	job specifications, Hadoop parameters
4	HaSim Parameters	system clock, execution speed, accuracy level, shared parameters, reporter

Table 3.1: Summarizes the parameters modelled in HaSim.

4 THE DESIGN OF HASIM

This section presents the design of HaSim in detail. The prototype is based on Hadoop framework.

4.1 HaSim Architecture

Figure 2 shows the data flow of HaSim. To perform a simulation, the Cluster Reader component reads the cluster parameters from the Cluster Spec to create a simulated Hadoop cluster environment. The Job Spec will be processed by the Job Reader component and jobs will be submitted to HaSim for simulation. HaSim follows a master-slave mode. The simulated Map instances (MapperSim), Reduce instances (ReducerSim), JobTracker and Task Trackers are located on these nodes. The Master node is the Name node of Hadoop framework which contains JobTracker to correspond and schedule the tasks. The Slave nodes are the Data nodes of Hadoop framework which contains TaskTrackers. On Slave nodes Map instances and Reduce instances perform data processing tasks .

From Figure 2 it can be observed that when a job is submitted to a simulated Hadoop cluster, the Job Tracker splits the job into several tasks. Then TaskTracker and Job Tracker will communicate with each other via messaging based on heartbeats. One thing should be pointed out that in Hadoop framework, the communications among JobTracker and TaskTrackers are based on HTTP. However in the simulator simplicity has been done. The HTTP communications are not simulated but using the times consumed by the communications to measure the overhead generated by HTTP communications. If the JobTracker finds that all the Map tasks have been finished, and then the Reduce instances will be notified to be ready for merging phase. Moreover if the JobTracker finds all Reduce tasks have been finished, then the job will be considered as finished.[4,5]

If the Map tasks have not been finished yet, the TaskTrackers will be notified to choose a Map task or a Reduce Task based on their availabilities.

4.2.4 JobTracker and TaskTracker .

JobTracker is mainly used to track a simulated job and TaskTracker is used to run individual tasks. When a job is submitted, the job ID will be sent to JobTracker for tracking. The JobTracker starts computing the input splits for the job. Then it creates one map task for each split. TaskTrackers periodically send messages to the JobTracker via heartbeats which tell the JobTracker that a TaskTracker is working. As part of the heartbeat, a TaskTracker will tell that if the current task is finished and ready to run a new task. Figure 4.6 shows the work flows of the components in HaSim.

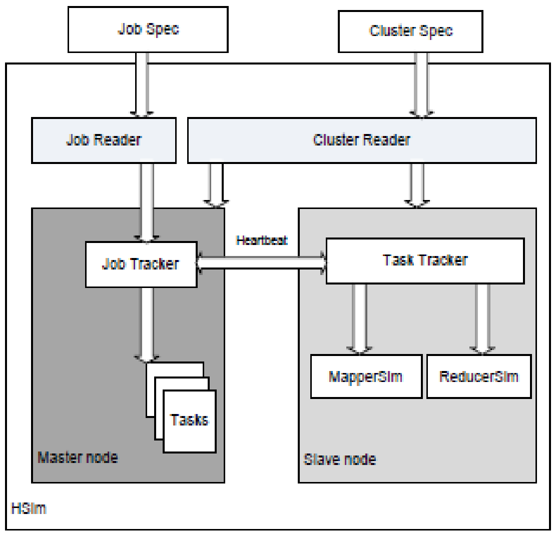

Figure 2: HaSim components.

4.2.2 MapperSim

When a Hadoop application is submitted to HaSim, the input data will be split into a number of data chunks and each chunk is associated with a Map instance. During the processing, each task will be assigned to a Map instance for execution. The operations of a Map instance are simulated by the MapperSim component. MapperSim simulates the operations of a Map instance (mapper) on each node. It copies data which is saved in HDFS to its own local hard disk. Commonly each MapperSim processes one file chunk but if only one file chunk is saved in HDFS, then a logically separated number of chunks can control the number of MapperSim instances involved in the job. When the data is copied and saved in the local hard disk, MapperSim starts processing the data based on the job spec of the simulated Hadoop application. During the processing steps, intermediate data will be generated. To improve the IO performance, the intermediate data will be written into a memory buffer. In the buffer, the data can be pre-sorted to gain high efficiency. As long as data is writing into the buffer, if a threshold is reached, a background thread will start spilling the data to hard disk. The intermediate data will be kept writing into the buffer while the spilling takes place. If the buffer is full during this time, the CPU processing will be blocked until the spill procedure is complete.
Figure 3: Data flows in the MapperSim component.
And if a Combine function is needed, combiner will be involved in this step after sorting.

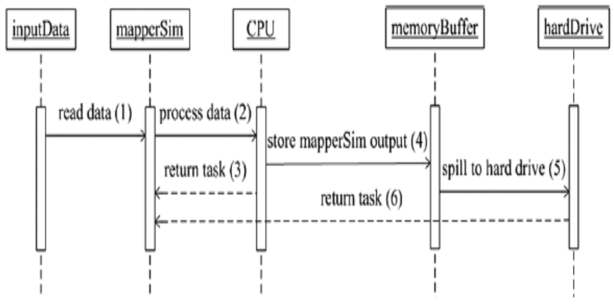

Figure 4: MapperSim sequence diagram

Figure 4 shows a sequential diagram shows the interactions of MapperSim with other components.HaSim.

4.2.3 ReducerSim

The ReducerSim component simulates the Reduce instances in Hadoop framework. It is used to collect the outputs from MapperSim and reduce the final outputs into HDFS. Figure 5 shows the data flows in ReducerSim.

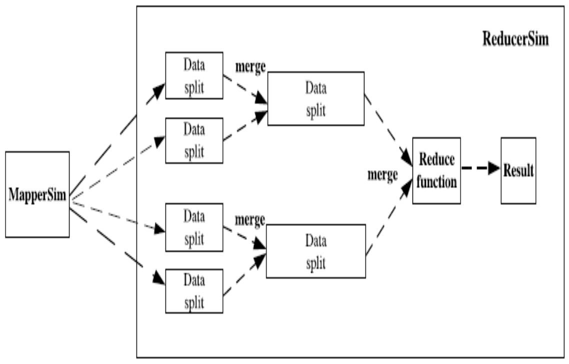

Figure 5: Data flows in the ReducerSim component.

The output files of the MapperSim component are saved in the local hard disk. The ReducerSim component needs the output from several MapperSim components for its particular partition. The ReducerSim starts copying data when an output is ready. Each ReducerSim has a number of copying threads so that it can copy the output results from a number of MapperSim components in parallel. If the size of the output is small, it will be copied into a memory buffer otherwise it will be copied into the hard disk directly. If the output results are copied into memory, when a certain threshold is reached, e.g. a percentage of buffer used or a number of file copied, these outputs will be merged and spilled into hard disk. As the number of files increases, a background thread merges them into larger and sorted files. When all the output results from the MapperSim components have been copied, the sorting step will start. This step merges the map outputs and maintains sorting orders of outputs. After the files have been sorted, they will be reduced into HDFS as one final output. For some Hadoop applications, the Reduce instances may need to process data involving processors but without IO operations. The ReducerSim in HaSim supports this feature. Figure 6 shows its sequence diagram.

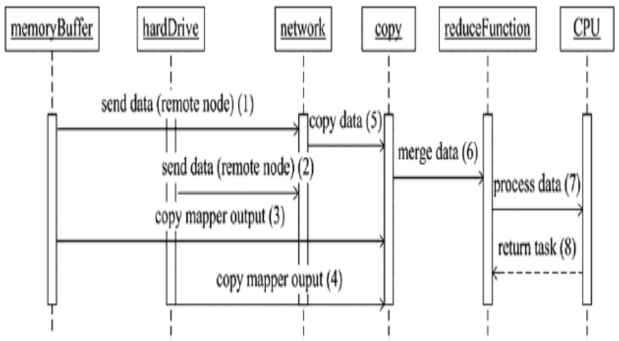

Figure 6: Hardware interactions in ReducerSim.

5. VALIDATIONS OF HASIM

To validate HaSim, a number of tests have been conducted. The performances of HaSim against published benchmark results have been compared. And also an experimental environment of a Hadoop cluster has been set up and the simulator HaSim is evaluated with our Hadoop applications.

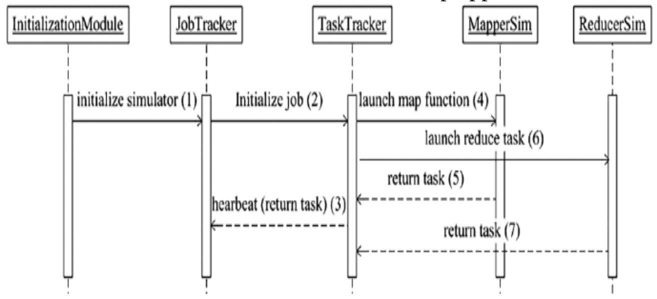

Figure 7: The workflow of HaSim.

5.1 Validating HaSim with Benchmarks

HaSim is validated firstly with 3 benchmark results presented in [6] [5] - Grep Task, Selection Task and UDF Aggregation Task.

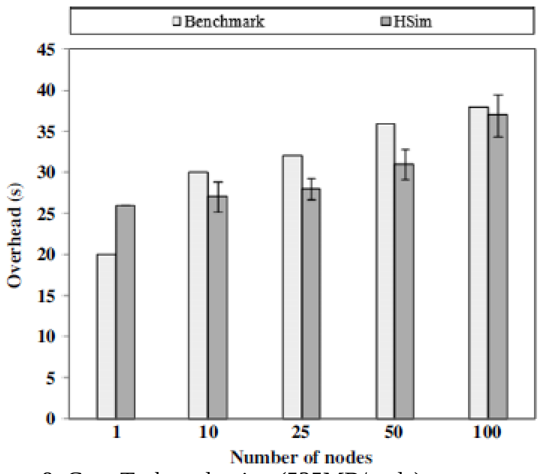

Figure 8: Grep Task evaluation (535MB/node).

5.1.1 Grep Task

This task simulated exactly what [6] [10,11] did in their benchmarking work. HaSim simulated the cluster using 1 node, 10 nodes, 25 nodes, 50 nodes and 100 nodes respectively. Two different scenarios have been tested, one is that each node is assigned 535MB data to process, and the other is that 1TB data is submitted to the cluster. Each scenario was evaluated 5 times. The simulation results are plotted in Figure 8 and Figure 9 respectively which are close to the benchmark results. Both the simulation results and benchmark results are in the same scale. Regarding the complex physical environments, the simulation results can supply acceptable accuracy.

The gaps between simulation results and benchmark results can be ignored. The confidence intervals of the results are small in both scenarios (in the range of 0 and 2.6 seconds in the first scenario and in the range of 4.1 and 7.6seconds in the second scenario) showing a stable performance of HaSim.

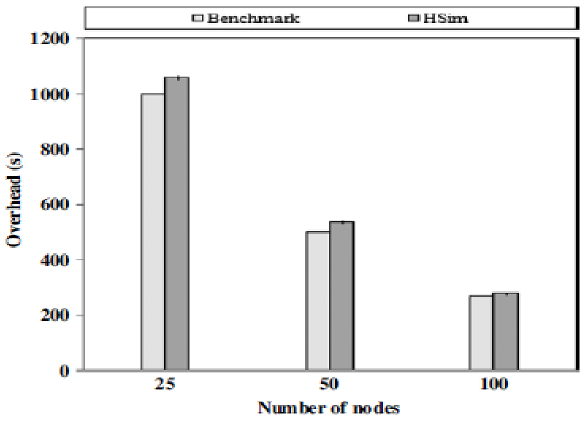

Figure 9 : Grep Task evaluation (1TB/cluster).

5.1.2 Selection Task

The Selection Task was designed to observe the performances of Hadoop framework dealing with complex tasks. Each node processes 1GB ranking table to retrieve the target pageURLs with a user defined threshold. This task is simulated and the results are shown in Figure 10.

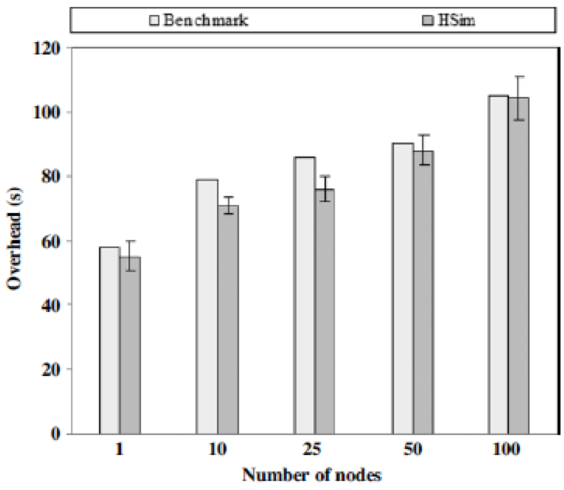

Figure 10: Selection task evaluation.

The simulation results show that considering the complex working mechanisms and parameters of Hadoop framework, the simulator HaSim can supply sufficiently close results compared to the benchmark results. From Figure 10 it can be clearly observed that the simulated results are close to the benchmark results, and the confidence intervals are small, in the range of 2.6 and 6.6 seconds.

5.1.3 UDF Aggregation Task

The UDF Aggregation Task reads the generated document files and searches for all the URLs appeared in the contents. And then for each unique URL, HaSim counts the number of unique pages that refers to that particular URL across the entire set of files.

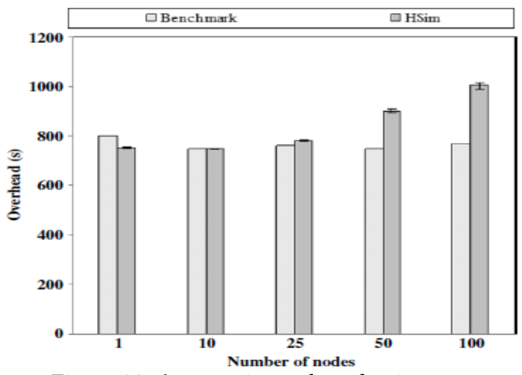

Figure 11: Aggregation task evaluation.

The simulation results are shown in Figure 11 which again are close to the benchmark results considering the complexities of the simulations. The simulation results show a high stability of HaSim for the task

5.2 Evaluating HaSim with Customized Hadoop Applications

Two customized Hadoop applications are involved for validation secondly - one is for information retrieval and the other one is for content based image annotation. The two applications were evaluated in both a Hadoop experimental cluster and HaSim. This section presents the evaluation results.

5.2.1 The Experimental and Simulated Environments

The Hadoop experimental cluster consists of 4 nodes. Three nodes were used as Datanodes with CPU Q6600@2.4G, RAM 3GB, 120GB Seagate Hard Disk, and running OS Fedora 12. One node is used Namenode with CPU C2D7750@2.26G, 2GB RAM and running OS Fedora 12. Each Datanode employed 4 mappers and 1 reducer with default cluster configurations. The network bandwidth is 1Gbps. We used HaSim to simulate a Hadoop cluster with the same configurations as those of the experimental cluster.

5.2.2 MR-LSI

MR-LSI [5] is a MapReduce based distributed LSI algorithm for information retrieval. The details will be described in the next chapter. MR-LSI is designed and implemented using the Hadoop framework. It involves both Map and Reduce functions, and contains a number of IO operations. MR-LSI was evaluated in both an experimental environment and HaSim, and plotted the results in Figure 12. It can be observed that the overall performance of HaSim is substantially close to that of the real Hadoop cluster, especially for scenarios dealing with MapReduce jobs withlarger sizes of datasets and involving an increased number of mappers.

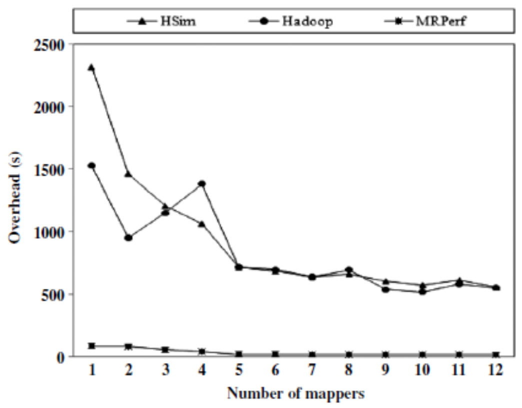

Figure 12: Evaluating HaSim with MR-LSI.

6 CONCLUSION

This chapter presents HaSim, a Hadoop simulator for simulating data intensive MapReduce applications. HaSim was validated with established benchmark results and also with experimental environments which have shown that HaSim can accurately simulator MapReduce behaviors. HaSim can be used to investigate the impacts of the large number of Hadoop parameters by tuning their values. It can also be used to study the scalability of MapReduce applications which might involve hundreds of nodes.

7 REFERENCES

[1] Apache Hadoop! Available at: http://hadoop.apache.org/ [Accessed Feb 2, 2013].

[2] Aarnio, T. (2010). Parallel Data Processing with Mapreduce. TKK T-110.5190, Seminar on Internetworking, Available: http://www.cse.tkk.fi/en/publications/B/5/papers/Aarnio_final.pdf.

[3] Alham, N. K., Li, M., Hammoud, S., Liu, Y., and Ponraj, M. (2010). A distributed SVM for image annotation. In: Proceedings of the 7th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), YanTai, China

[4] He, B., Fang, W., Luo, Q., Govindaraju, N. K., and Wang, T. (2008). Mars: a MapReduce framework on graphics processors. In PACT '08: Proceedings of the 17th international conference on Parallel architectures and compilation techniques, 260–2698

[5] Pavlo, A., Paulson, Madden, and S., Stonebraker, M. (2009). A comparison of approaches to large-scale data analysis.

[6] Taura, K., Kaneda, K., Endo, T., and Yonezawa, A. (2003). Phoenix: a parallel programming model for accommodating dynamically joining/leaving resources. SIGPLAN Not., 38, 216–229. 7[51]

[7]The Network Simulator - ns-2 Available at: http://www.isi.edu/nsnam/ns (Last accessed: 19-May-2013).

[8] Venner, J. (2012). Pro Hadoop (1st ed). New York: Springer.

[9] Wang, G., Butt, A. R., Pandey, P., and Gupta, K. (2009). Using realistic simulation for performance analysis of mapreduce setups.

[10] Wang, G., Butt, A. R., Pandey, P., and Gupta, K. (2011). A Simulation Approach to Evaluating Design Decisions in MapReduce Setups. In: Proceedings of the 17th Annual Meeting of the IEEE/ACM International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '11), London, UK.