CHAPTER 3

HADOOPSIM: A MAPREDUCE SIMULATOR

MapReduce is a promising technology in support of Cloud Computing. Hadoop which is a MapReduce implementation has been broadly used in developing distributed MR applications. This chapter presents Hadoop Simulator (HSim), a MapReduce simulator which builds on top of Hadoop. HSim models a great number of parameters that can upset the behavior of MapReduce nodes, and thus it can be used to tune the performance of a MapReduce cluster. HSim is validated with both standard results and user personalized MR applications.

3.1 SHAPEING HADOOP PARAMETERS

The performance of a Hadoop application can be changed by a big number of parameters. This section, discusses the modeling work on these parameters.

3.1.1 Parameters of Node

A) Processor: HSim supports one processor per computer by default design, but the number of processors could be altered. 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 second which can be measured from actual experimental tests.
B) Hard disk: In this entity, the speeds of IO operations fluctuate from time to time. Numerous parameters are introduced to build the digressive reading/writing model.

Let $x_{\text{max}}$ represent the maximum reading/writing speed of hard disk. For example from the experimental results of testing Seagate Barracuda 1 TB hard disk $x_{\text{max}}$ is about 120MB/s in reading, and 60MB/s in writing. Let $x_{\text{min}}$ represent the minimum reading/writing speed of hard disk, $x_{\text{min}}$ is around 55MB/s in reading and 25MB/s in writing. Another parameter which is degressive factor $r$ is used to represent in each second the value of last speed. The value of the factor is around 0.0055 based on experimental tests. Using these parameters we can calculate the actual time speed $x$ of hard disk using formula given in the Equation (3.1)

$$H(t) = \frac{x_{\text{min}} x_{\text{max}}}{(x_{\text{min}} - x_{\text{max}})e^{-rt} + x_{\text{max}}} \quad (3.1)$$

C) Memory: In every memory entity two parameters are modeled, reading and writing. In our experimental tests, the reading speed of standard DDR2-800 memory with dual-channel could reach up to 6000MB/s and the writing speed is up to 5000 MB/s. It is quite evident that both the reading and writing speeds would not be the blocks of the system owing to their fast speeds.

D) Ethernet adaptor: In every Ethernet adaptor entity, two parameters are modeled, upstream bandwidth and downstream bandwidth. The bandwidth can be in the range of 100Mbps and 1000Mbps.
3.1.2 Parameters of Cluster

The cluster parameters denote the details of a simulated Hadoop cluster. It involves numerous aspects which include the number of nodes, topology, and network facilities.

**Number of nodes:** The number of nodes can vary from one to hundred.

**Topology:** The number of nodes can be organized with a certain network topology. At present HSim only supports easy racks.

**Network facilities:** The speed of a router can be in the range of 100Mbps and 1000Mbps. When the bandwidth of a router is defined, a number of standalone computers must be prepared to connect to the router to fix 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. HSim supports two job schedulers of Hadoop framework – first come first served and fair scheduler. These two types of schedulers generate different job processing orders.

3.1.3 Parameters of Hadoop System

Prior to a Hadoop application starts processing the data, the data should be saved into Hadoop Distributed File System (HDFS) by White (2012) in advance. The number of files changes the number of Map instances implicated. Normally the number of Map instances matches 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 to data chunks via waves. If a whole data set is saved only in one file, the single file could be separated into a number of chunks logically via supplied APIs of the Hadoop
framework. When the data are being processed, they would go through a number of processing steps such as sorting, merging, combining, copying, reducing. These steps extremely affect the performances of the system so that several parameters are demonstrated to control the actions of these steps. As these parameters are configurable and most of them are involved in the actual Hadoop framework, we named these parameters as Hadoop system parameters.

**Job specifications:** Here, a number of parameters are involved to describe the properties of a job. Job ID refers the unique id assigned to each job for tracking. The Job Size is the total size of the input data. Regardless of how many chunks of the data are submitted, this value should be the total size of the whole data. When the simulation starts, the data will be obtained from the HDFS. The number of records 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 performances of the system when certain types of Hadoop applications are executed. The Map output ratio parameter represents the volume of intermediate data to be generated by Map instances which has an impact on IO performance. The reduce output ratio parameter is quite similar to Map output ratio. In some Hadoop applications the Reduce instances do not only copy the data from Map instances but also generate their own intermediate data which affect IO performance. This parameter specifies the size of intermediate data to be generated in the Reduce phase. The Reducing Ratio parameter represents the size of final results which will be reduced in HDFS. This parameter can influence 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 the data. This parameter concludes 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. If this parameter is defined, then a number of reduce instances will be allocated for the current job according to their availabilities.

**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 amount of io.sort.mb reserved for storing record boundaries of the map output results. The left over 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 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. The mapred.reduce.parallel.copies parameter refers to the number of threads used to copy map outputs to the reducer. Using a suitable number of copying threads according to hardware resources, the performances of the system would be enhanced. The io.sort.factor parameter corresponds to the maximum number of streams to merge when sorting files is carrying out in the reduce phase. The mapred.job.shuffle.input.buffer.percent parameter is the proportion of total heap size to be allocated to the map outputs buffer during the copy phase of the shuffle. The mapred.inmem.merge.threshold parameter represents the threshold number of map outputs for starting the process of merging the outputs and spilling to hard disk. Using this parameter a number of smaller mapper outputs could be operated in memory but not local hard disk. Therefore the sorting and
merging involve less overhead generated by hard disk. The JVM reuse parameter is partially simulated in HSim. Using JVM reuse, the overhead generated by some short-lived tasks will be significantly reduced.

3.1.4 Parameters of HSim

HSim itself needs some parameters to control its performance. Five important parameters are introduced in HSim.

System Clock: The system clock parameter is an absolutely and continuously timing component. In every 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 HSim.

Accuracy Level: For normal Hadoop applications, it is enough to set this parameter to the level of seconds. To sustain 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 that include hard disk and bandwidth. The ratio is defined by

\[ r = \frac{\text{Assigned Resource}}{\text{Total Resource}} \]

Reporter: This parameter records several important system states for analysis.
### Table 3.1 Summary of the parameters modeled in HSim

<table>
<thead>
<tr>
<th>S.No</th>
<th>Category</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Parameters of Node</td>
<td>processor, hard disk, memory, Ethernet card, Map instance, Reduce instance</td>
</tr>
<tr>
<td>2</td>
<td>Parameters of Cluster</td>
<td>number of nodes, topology, network facilities, job queue, job scheduler</td>
</tr>
<tr>
<td>3</td>
<td>Parameters of Hadoop System</td>
<td>job specifications, Hadoop parameters</td>
</tr>
<tr>
<td>4</td>
<td>Parameters of HSim</td>
<td>system clock, execution speed, accuracy level, shared parameters, reporter</td>
</tr>
</tbody>
</table>

### 3.2 THE PLAN OF HSIM

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

#### 3.2.1 Architecture of HSim

The data flow of HSim shown in Figure 3.1. To achieve a simulation, the Cluster Reader component reads the cluster parameters from the Cluster Spec to create a simulated Hadoop cluster environment. A specified number of nodes are initialized and arranged using a certain type of topology. After the cluster is configured, the node parameters will be processed by the Cluster Reader as well to specify the types of nodes including processors, hard disk, memory, Master node, Slave nodes, Map instances and Reduce instances. This initialization process can create both homogeneous nodes and heterogeneous nodes. Then the simulated cluster is ready for incoming jobs retrieved from the job queue using different job schedulers. The Job Spec will be processed by the Job Reader component and jobs will be submitted to HSim for simulation.
HSim tracks a master-slave mode. The simulated Map instances (MapperSim), Reduce instances (ReducerSim), JobTracker, and TaskTrackers are located on these nodes. The Master node is the Name node of Hadoop framework which comprises JobTracker to correspond and schedule the tasks. The Slave nodes are the Data nodes of Hadoop frameworks which comprise TaskTrackers. On Slave nodes Map instances and Reduce instances perform data processing tasks. From Figure 3.1 it can be observed that when a job is submitted to a simulated Hadoop cluster, the JobTracker splits the job into several tasks. Then TaskTracker and JobTracker 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, then the Reduce instances will be notified to be ready for merging.
phase. Moreover, if the JobTracker finds that all Reduce tasks have been finished, then the job will be considered as finished. 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.

3.2.2 MapperSim

When a Hadoop application is presented to HSim, the input data will be divided into a number of data chunks and each chunk is associated with a Map instance by Liu et al (2013). 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 the data which are 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 are 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 produced. To advance 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 the data are being written into the buffer, if a threshold is reached, a background thread will start spilling the data to the 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. This step means that the processor involved in MapperSim does not simply keep processing, it may be interrupted by the current states of memory buffer. For each spilled chunk of the output, before it is written to the hard disk the background thread will split the chunk into partitions which are associated with the Reduce instances. During this step, the in-memory pre-sorting happens.
And if a Combine function is needed, the combiner will be involved in this step after sorting. After the task is finished, the partitions will be combined into a single file which contains sorted data to be copied to the Reduce instances.

Figure 3.2 Data flowing in the MapperSim component

Figure 3.2 shows the working mechanism of MapperSim. Figure 3.3 shows a sequential diagram describing the interactions of MapperSim with other components in HSim.

3.2.3 ReducerSim

The ReducerSim component simulates the Reduce instances in Hadoop framework. It is used to gather the outputs from MapperSim and reduce the final outputs into HDFS. Figure 3.4 shows the data flowing in
ReducerSim. The output files of the MapperSim component are kept in the local hard disk. The ReducerSim component needs the output from several MapperSim components for its particular partition.

![Diagram of data flow](image)

**Figure 3.4 Data flowing in the ReducerSim component**

The ReducerSim starts copying the 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 files copied, these outputs will be merged and spilled into hard disk. As the number of files increases, a background thread combines 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 combines 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 HSim supports this feature. Figure 3.5 shows its sequence diagram.
3.2.4 Trackers

JobTracker is mainly used to track a simulated job and TaskTracker is used to run separate 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 state if the current task is finished and ready to run a new task. Figure 3.6 shows the work flow of the components in HSim.

Figure 3.5 Hardware interactions in ReducerSim

Figure 3.6 The workflow of HSim
3.3 VALIDATIONS OF HSIM

To confirm HSim, a number of tests have been conducted. The performances of HSim against published yardstick results have been compared. Also an experimental environment of a Hadoop cluster was set up and the simulator HSim was evaluated with our Hadoop applications.

3.3.1 Confirming HSim with Benchmarks

HSim is validated first with 3 standard results presented in (Kyong-Ha et al 2011, Stonebraker et al 2010) Grep Task, Selection Task, and UDF Aggregation Task.

3.3.1.1 Grep task

This task simulated exactly what (Kyong-Ha et al 2011, Stonebraker et al 2010) prepared in their benchmarking work. HSim simulated the cluster using 1 node, 10 nodes, 25 nodes, 50 nodes and 100 nodes respectively. Two different situations were tested, one was that each node was assigned 525MB data to process, and the other was that 1TB data is submitted to the cluster. Each situation was evaluated 3 to 5 times. The simulation results are plotted in Figure 3.7 and Figure 3.8 respectively which are near to the standard results. Both the simulation results and benchmark results are in the same scale. Regarding the complex physical environments, the simulation results can give acceptable accuracy. The gaps between simulation results and benchmark results can be disregarded. The confidence intervals of the results are small in both situations (in the range of 0 and 2.5 seconds in the first situation and in the range of 4.0 and 7.3 seconds in the second situation) showing a solid performance of HSim.
3.3.1.2 Selection task

The Selection Task was designed to discover the performances of Hadoop framework dealing with complex tasks. Each node processed 1GB ranking table to retrieve the target page URLs with a user defined threshold. This task was simulated and the results are shown in Figure 3.9. Regarding the
multiple factors and complexity of the physical standard environments, the simulation results are highly close to the benchmark results. The simulation results show that considering the complex working mechanisms and parameters of Hadoop framework, the simulator HSim can give sufficiently very close results compared with the standard results.

3.3.1.3 UDF Aggregation task

The UDF Aggregation Task read the generated document files and searches for all the URLs that appeared in the contents. And then for each unique URL, HSim counted the number of unique pages that referred to that particular URL across the entire set of files. The simulation results are shown in Figure 3.10 which again is close to the standard results considering the complexities of the simulations. The simulation results show a great stability of HSim for the task.

![Figure 3.9 Selection task evaluation](image-url)
3.3.2 Assessing HSim with Customized Hadoop Applications

Two customized Hadoop applications were involved for validation: one was for information retrieval and the other one was for content-based image annotation. The two applications were evaluated in both Hadoop experimental cluster and HSim. This section offers the assessment results.

3.3.2.1 The experimental and simulated situation

The Hadoop experimental cluster consisted of 4 nodes. Three nodes were used as Datanodes with CPU Q5004@2.2G; RAM 4GB, 120GB Seagate Hard Disk, and running OS Fedora 12. One node was used Namenode with CPU C2D5004@2.26G, 2GB RAM and running OS Fedora 12. Each Datanode employed 4 mappers and 1 reducer with default cluster configurations. The network bandwidth was 1Gbps. We used HSim to simulate a Hadoop cluster with the same configurations as those of the experimental cluster. This experimental environment is totally differs with the previous work carried out by other researchers.
3.3.2.2 MR-LSI

MR-LSI was designed and implemented using the Hadoop framework. MR-LSI (Hang et al. 2009 & Hammoud et al. 2010) is a MapReduce based distributed LSI algorithm for information retrieval. The details will be described in the next chapter. It involved both Map and Reduce functions and contained a number of IO operations. MR-LSI was evaluated in both experimental environment and HSim, and the results are shown as graph in Figure 3.11.

![Figure 3.11 Assessing HSim with MR-LSI](image)

It can be noticed that the overall performance of HSim is substantially close to that of the real Hadoop cluster, especially for scenarios dealing with MapReduce jobs with larger sizes of datasets and involving an increased number of mappers. One thing should be pointed out that HSim is designed to simulate a large scale Hadoop cluster so that if only one node is in the cluster the errors may occur due to inaccuracy of simulating a cluster consisting of a single node. (In this case one machine employs four mappers so when the number of mappers is less than 5, there is only one node in the
cluster). For comparison purpose, the performance of the MRPerf simulator is also tested using the same configurations as that of HSim. From the results presented in Figure 3.12 it can be seen that HSim significantly outperforms MRPerf in comparison with the performance of the real Hadoop cluster. As discussed in Section 2, using too much estimation on the values of Hadoop parameters limits MRPerf in simulating MapReduce behavior accurately.

3.3.2.3 MR-SMO

MR-SMO (Alham et al 2011) is a MapReduce based distributed SMO algorithm for content based image annotation. MR-SMO was built on Hadoop framework, and also involved both Map and Reduce functions. MR-SMO was evaluated in the experimental Hadoop cluster as well as in HSim. The MRPerf simulator was also employed to evaluate the performance of MR-SMO. From the results presented in Figure 3.12 it can be observed that the performance of the simulated cluster using HSim is noticeably close to that of the real Hadoop cluster. Again MRPerf does not produce precise simulation results.

![Figure 3.12 Evaluating HSim with MR-SMO](image-url)
3.3.3 Debates

The Hadoop framework is a composite system involving a number of components. HSim is designed and implemented to simulate such components and collaborations. It works in the way the Hadoop framework works. However, we cannot simply conclude that HSim can perfectly simulate Hadoop without any restrictions. The precision of HSim can be affected by a number of factors such as the time of job propagations, cold starts of Map instances, key distributions, system communications, shared hardware resources and dynamic IO loads. Enabling the Combiner feature of Hadoop also can affect the accuracy of HSim. However, the combiner instance has not been fully implemented in HSim. A combiner can be considered as an in-memory sort process. The output of mappers will be combined and written into an intermediate file by a combiner. And then the file will be sent to a reducer. So when the number of mappers is small, the benefits gained from using combiners are not significant. However, when the number of mappers gets large, system IO operations comprises hard disk reading, writing, and network utilities will benefit a lot from combiners. Though HSim does not work that well with simulated combiners in large clusters, it still performs well in a simulated cluster with up to 100 nodes.

3.4 SUMMARY

This section presents HSim, a Hadoop simulator for simulating data intensive MapReduce applications. HSim was validated with established benchmark results and also with experimental environments which have shown that HSim can perfectly simulate MapReduce performance. It can be used to study the scalability of MapReduce applications which might comprise hundreds of nodes. HSim can also be used to explore the impacts of the large number of Hadoop parameters by changing their values.