ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 1, Issue 7, September 2013

MART in Load Balancing Algorithm for Distributed Applications

R.Palson Kennedy[1], Dr.T.V.Gopal[2]

Research Scholar, Faculty of I & C , Anna University Chennai 25 & Professor, PMREC, Chennai,India[1]

Professor, Dept. Of CSE, Faculty of I & C, Anna university, Chennai, India[2]

ABSTRACT: The distributed computations are widely used in the modern world for processing large scale jobs. Hadoop framework which is based on Google Map Reduce model becomes popular due to its great processing power and ease to use. However, due to lack of load management, especially in a heterogeneous computing environment, the performance of Hadoop framework may be deteriorated. Therefore this paper presents a load balancing algorithm which aims to balance the load among heterogeneous nodes. And also, the Hadoop simulator HaSim is involved to evaluate the performance of the load balancing algorithm. The results indicate that the performance of the cluster has been significantly enhanced.

Keywords: Distributed computing; MapReduce; Hadoop; Load balancing; HaSim
.

I. INTRODUCTION

Hadoop framework [3] [6] [7] is designed to process large scale data in a distributed computing environment based on MapReduce computing model [4] [5]. As being claimed by Hadoop, the framework facilitates the developments of distributed computing based applications. These kinds of facilities are based on the interactions among three important components mainly which are named HDFS, Map instances (mappers) and Reduce instances (reducers)Technique(MRT).Though the overall structure of the Hadoop framework simplifies the processing, the components hide a lot of complex low-layer details including hardware and software aspects at the background. The framework also supplies a simple job scheduler FIFO (First In First Out) which serves the jobs in order of their submissions. The sequential scheduler could ease the management of job to some extent and sometimes it is efficient when the framework deals with the job queue. However, some important heterogeneous factors have not been considered by the job scheduler yet. And also it is well know that at present, many clusters are running in highly homogenous environments. For example Hadoop at Yahoo [1] employs a homogenous cluster with 4000 processors, 2TB RAM and 1.5PB storage capacity. A large number of benchmarks and sorting competitions have been tested based on the environment. The results have been published to show the powerful of Hadoop framework. In these distributed computing tasks, people focus on the extreme performances using homogenous environment which can ideally avoid the unbalanced load issues. Therefore, behind these highlighted results, the load balancing issue is quite considerable of which has been hidden deeply by the homogeneous environments. Normally, it is extremely hard to build up a homogenous cluster with a number of nodes up to several thousands. As a result, a number of Hadoop clusters with heterogeneous nodes are quite common. The architecture of Hadoop framework has been designed quite flexible to adapt to heterogeneous resources. Thus, it can be seen clearly that the heterogeneities of the resources will affect the performance of the cluster. Devaraj Das [2], the engineering manager of Yahoo Bangalore Grid Computing Group concludes the load issue from four aspects - imbalance in input splits, imbalance in computations, imbalance in partition sizes, and imbalance in heterogeneous hardware.

In this paper, we present a load balancing algorithm in heterogeneous MapReduce environments with considering the interactions of a number of factors including Hadoop parameters. We designed and implemented the algorithm using the Hadoop simulator HaSim. And then we evaluated the performances of the proposed algorithm and

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 1, Issue 7, September 2013

comparison is made with the other load balancing strategies of MapReduce. The results show a great improvement in performance in terms of the efficiency of the simulated cluster.

II. REVIEW OF EXISTING WORKS

At present there are a few researches focusing on studying load balancing for MapReduce. One research contributed by Sven Groot [8] pointed out that due to the overheads of data copying, network transferring, local hard disk reading and writing, a mapper may limit the job execution time. To show the impacts of unbalanced load issue, the author use Jumbo based on Google Distributed File System [9]. In the author's scenario he implemented two algorithms of which one is a single algorithm called 'word count' [3] and the other one is a complex one called 'Parallel FP-Growth frequent item set mining algorithm' [10]. The experimental results show that the slower nodes delay the processing which causes that the faster nodes are not fully utilized. Based on the results, the author claimed that both mapper and reducer impact the performance of Hadoop framework. However, firstly in this paper only a number of experiments have been done without any solution on solving the unbalanced load issue. Secondly, the impacts brought by reducer should be considered. For theoretical algorithm experiments, multiple reducers may be involved in terms of efficiency. Contrarily, for a practical algorithm, reducer is normally involved to collect the final output which should be regarded as a whole data set without any segmentation. Thus for the data integrity, single reducer is better than multiple reducers which needs another job to collect parts from different reducers to form a whole data set.

Therefore it is regarded that in the data processing, the load issues among multiple mappers are more critical. One group of researchers realized the importance of load balancing issue in Hadoop as well. Sadasivam et al. [11] try to optimize the performance of the Hadoop cluster so that they proposed an approach called Parallel Hybrid PSO-GA using MapReduce based on genetic algorithm. In their algorithm they use Hadoop framework itself to deal with the genetic algorithm [12] which aims to solve the unbalanced load issue in Hadoop. Their algorithm mainly aims on achieving an optimized scheduler for multiple users based on the different resource capacities. During the processing, they made the number of iterations maximally 30 times to guarantee the efficiency. Their results show that the PSO-GA algorithm outperforms Max MIPS, typical PSO and typical GA. However, several points can be criticized from their design and implementation. The first point is that the overhead of Hadoop is quite considerable. When the framework is involved to compute a Hadoop job scheduler for Hadoop itself, though the overhead of following jobs may be reduced, the overhead of the scheduler computation definitely cannot be avoided. The second point in their design is they just simply consider the capacity of a resource in terms of utilization of processor. This simply idea is lack of accuracy to describe the real Hadoop system. As studied in paper [13], there are a number of factors which may impact the performances of the framework including processing features, IO features and Hadoop working mechanisms. Therefore the fitness function based on pure utilizations of processors in Parallel Hybrid PSO-GA cannot compute the scheduler accurately. The third point is they considered balancing the load among multiple users but they do not consider the load among mappers for one job. Thus the unbalanced load will make certain number of mappers unutilized, which delays one job. Moreover the total number of jobs will be affected.

III. HADOOP SIMULATOR -HASIM

The Hadoop simulator HaSim follows a master-slave mode in its design. Parameters related to a simulated cluster include the number of Hadoop nodes, the topologies of these nodes (currently only supporting simple racks), the number of mappers and reducers, the CPU speed, memory size, the average reading and writing speeds of hard disk and network bandwidth of each node. HaSim supports one processor per node and each processor can have one or more processor cores.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering
(An ISO 3297: 2007 Certified Organization)
Vol. 1, Issue 7, September 2013

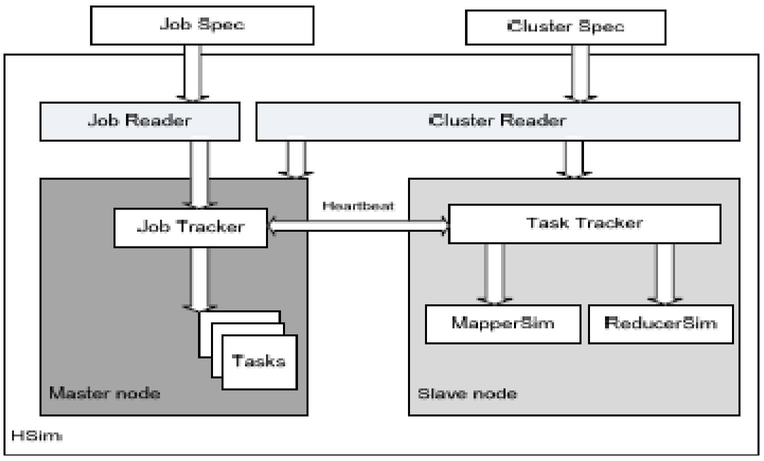

Figure 1: HaSim architecture.

The values of some parameters such as CPU speed and the writing and reading speeds of hard disk can be assigned based on measurements from real-world experiments. The NumberOfReducers specifies the number of reduce instances. Figure 1 shows the software architecture of HaSim.
The validation is based on the published benchmark results [14] using algorithms Grep, Selection and UDF Aggregation. Figure 2 and 3 show the simulator can simulate the framework with stable and accurate performance.

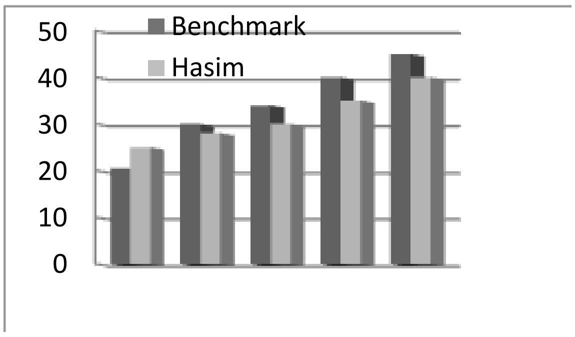

Figure2:Grep(535MB/node). (X-No of Nodes, Y-overheads)

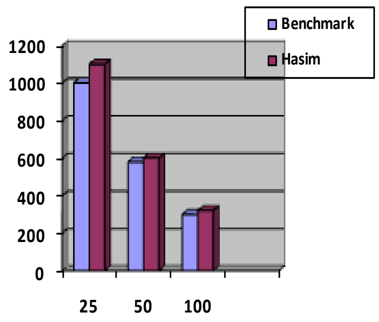

Figure 3: (1TB/cluster). (X- No of Nodes, Y-overheads)

IV. ALGORITHM DESIGN

To solve an optimization problem, genetic algorithm solutions need to be represented as chromosomes encoded as a set of strings which are normally binary strings. However, a binary representation is not feasible as the number of mappers in a Hadoop cluster environment is normally large which will result in long binary strings. A decimal string to represent a chromosome in which the data chunk assigned to a mapper is represented as a gene is employed. In Hadoop, the total time (T) of a mapper in processing a data chunk consists of the following four parts:
Data copying time (..Tc..) in copying a data chunk from Hadoop distributed file system to local hard disk.
Processor running time (Tp..) in processing a data chunk.
Intermediate data merging time (.Tm.) in combining the output files of the mapper into one file.
Buffer spilling time (Tb.) in emptying filled buffers. T=Tc+Tb+Tm+Tb..........................
Let
Dm.. be the size of the data chunk.
Hd.. be the writing speed of hard disk in MB/second.
Bw. be the network bandwidth in MB/second.
Pr.be the speed of the processor running the mapper process in MB/second.
Bf. be the size of the buffer of the mapper.
Ra. be the ratio of the size of the intermediate data to the size of the data chunk.
Nf be the number of frequencies in processing intermediate data.
Nb be the number of times that buffer is filled up.
Vb be the volume of data processed by the processor when the buffer is filled up.
S be the sort factor of Hadoop.
Therefore Tc =.Dm/min (Hd,Bw)

Here Tc.. depends on the available resources of hard disk and network bandwidth. The slower one of the two factors will be the bottleneck in copying data chunks from Hadoop distributed file system to the local hard disk of the mapper.Tp=Dm/Pr. When a buffer is filling, the processor keeps writing intermediate data into the buffer and in the mean time the spilling process keeps writing the sorted data from the buffer to hard disk. Therefore the filling speed of a buffer can be represented by
Pr x Ra -Hd.... Thus the time to fill up a buffer can be computed by Bf/. Pr x Ra -Hd. As a result, for a buffer to be filled up, the processor will generate a volume of intermediate data with the size of .... which can be computed using equation below:
Vb = Pr x.Ra x Bf/. Pr x Ra -Hd.
The total amount of intermediate data generated from the original data chunk with a size of .Dm..isDm x Ra.. Therefore the number of times for a buffer to be filled up can be computed using equation: Nb = Dm x Ra/Vb.
The time for a buffer to be spilled once is Bf/Hd., therefore the time for a buffer to be spilled for Nb.. times is Nb x Bf / Hd. Then we have
Tb = Nb x Bf / Hd
The frequencies in processing intermediate data Nf. can be computed using equation : .Nf =.(Nb/s)-1
When the merging occurs once, the whole volume of intermediate data will be written into the hard disk causing an overhead of Dm x Ra/Hd..

Thus if the merging occurs Nf times, the time consumed by hard disk IO operations can be represented by Dm x Ra x Nf/Hd...we have
Tm = Dm x Ra x Nf/Hd
The total time .Ttotal to process data chunks in one processing wave in MapReduce Hadoop is the maximum time consumed by k participating mappers, where Ttotal= max (T1,T2,T3,.....,Tk). According to divisible load theory [15], to achieve a minimum Ttotal., it is expected that all the mappers to complete data processing at the same time: T1=T2=T3=.....=,Tk . Let
Ti be the processing time for the .ith mapper.
T^{-} be the average time of the k mappers in data processing, at the same time:T^{-} =ΣTi/k

The fitness function can be defined using equation:

$$f(T) = \sqrt{\sum_{i=1}^{k} (\bar{T} - T_i)^2}$$

IV. SIMULATED RESULTS

The heterogeneity of the cluster is defined using the equation:

$$Heterogeneity = \sqrt{\sum_{i=1}^{k} (\bar{p} - p_i)^2} \text{ where } \bar{p} = \frac{\sum_{i=1}^{k} p_i}{k}, \sum_{i=1}^{k} p_i = P$$

whereP. represent the total processing speed of the cluster.
Pi. represent the processing speed of the ith processor.
p¯ . represent the average processing speed of the cluster.
k. represent the number of processor employed in the cluster.
Table 1 shows the simulated environment in detail, and Figures 6 ,7and 8 show the simulation results.

Simulation Environment	
Simulated algorithm	MR-LSI[16]
Number of simulated nodes:	20
Number of processors in each n¯ode	1
Number of cores in each processor	2
Size of data Test 1	10GB to 100GB
Reading speed of Hard disk:	80MB/s
Writing speed of Hard disk	40Gb/s
Sort factor	100

Table 1: The simulated environment.

The processing speeds of processors: Depending on heterogeneities
Heterogeneities: from 0 to 2.28
Number of hard disk in each node: 1
Number of Map instances 2
Number of Reduce instances 1

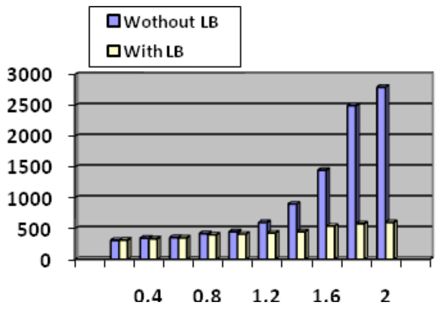

Figure 4: The performance of the load balancing scheme. (X-length of Heterogenity, Y-overheads)

Firstly 10GB data has been tested in the simulated cluster with different levels of heterogeneity. From Figure 4 it can be observed that when the level of heterogeneity is less than 1.08 which indicates a nearly homogeneous environment, the load balancing scheme does not make any difference to the performance of MR-LSI. However, the load balancing scheme reduces the overhead of MR-LSI significantly with an increasing level of heterogeneity.

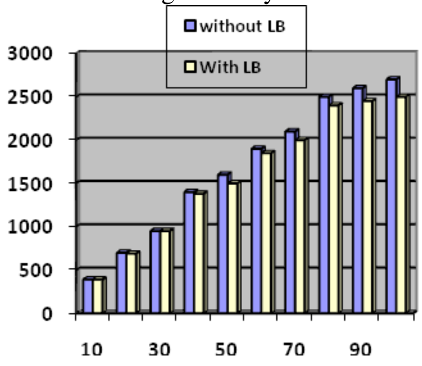

Figure5: The performance of the MR-LSI with difference sizes of data.(X-Data Size , Y-Overhead)

The levels of heterogeneity are keeping the same in the tests but varied the size of data from 1GB to 10GB. This set of tests was used to evaluate how the load balancing scheme performs with different sizes of datasets. Figure 5 shows that the load balancing scheme can always reduce the overhead of MR-LSI.

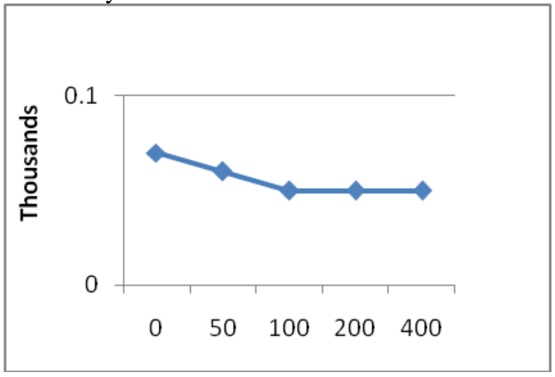

Figure 6: Convergence of the genetic algorithm. (X-No of Nodes, Y-overheads

The load balancing scheme builds on a genetic algorithm whose convergence affects the efficiency of MR-LSI. To analyze the convergence of the genetic algorithm, the number of generations is varied and the overhead of MR-LSI in processing a 10GB dataset in the simulated Hadoop environment is measured. Figure 6 shows that MR-LSI reaches a stable performance when the number of generations in the genetic algorithm reaches 300 indicating a quick convergence of the genetic algorithm.

VI. CONCLUSION

This paper presents a genetic algorithm based load balancing algorithm for Map Reduce environments in support of data intensive distributed applications. Simulation results have shown the effectiveness of the algorithm in balancing workload among Map Reduce nodes .The MR-LSI algorithm speeds up the computation process of SVD and maintains the high level of accuracy in information retrieval .

REFERENCES

[1] Yahoo. Hadoop at Yahoo! Available at: http://developer.yahoo.com/hadoop/. (Lasted accessed: 20-July-2013).
[2] Devaraj Das. How to Hadoop. Available at: http://trac.nchc.org.tw/cloud/raw-attachment/Fwiki/HadoopWorkshop/h adoop-assembled.pdf (Last accessed: 03-Sept-2012)
[3] Apache Hadoop! Available at: http://hadoop.apache.org/ [Accessed November 2, 2012].
[4] Dean, J., and Ghemawat, S. (2011). MapReduce: Simplified Data Processing on Large Clusters. In Proc. of OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA.
[5] Lämmel, R. (2007). Google's MapReduce programming model — Revisited. Sci. Comput. Program. vol. 68.
[6] Venner, J. (2011). Pro Hadoop (1st ed). New York: Springer.
[7] White, T. (2012). Hadoop: The Definitive Guide (2nd ed.). CA : O'Reilly Media.
[8] Groot, S. (2010). Jumbo: Beyond MapReduce for Workload Balancing. VLDB 2010 , 36th International Conference on Very Large Data BasesSingapore.
[9] Gobioff, H., and Leung, S.T. (2003). The google file system. In SOSP '03, pp 29-43, New York, NY, USA.
[10] Li, H., Wang, Y., Zhang, D., Zhang, M., and Chang, E. Y. (2008). Pfp: parallel fp-growth for query recommendation. In RecSys '08, pp 107-114, New York, NY, USA.
[11] Sadasivam, G. S., and Selvaraj, D. (2012). A Novel Parallel Hybrid PSO-GA using MapReduce to Schedule Jobs in Hadoop Data Grids. 2012 Four World Congress on Nature and Biologically Inspired Computing, Kitakyushu, Fukuoka, Japan.
[12] Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, Mass.: Addison-Wesley, 1989.
[13] Goda, K., Tamura, T., Oguchi, M., and Kitsuregawa, M. (2002). Run-time load balancing system on san-connected pc cluster for dynamic injection of cpu and disk resource - a case study of data mining application.
[14] Paulson, E., Rasin, A., Abadi, D. J., DeWitt, D. J., Madden, and S., Stonebraker, M. (2009). A comparison of approaches to large-scale data analysis. In: Proceedings of the 35th SIGMOD international conference on Management of data, New York, NY, USA.
[15] Shokripour, A., and Othman, M. (2011). Survey on Divisible Load Theory and its Applications. International Conference on Information Management and Engineering, pp 300-304.
[16] Li, M., Hammoud, S., Alham, N. K., and Ponraj, M. (2012). A MapReduce based distributed LSI. In: Proceedings of the 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), YanTai, China. Pp 206-212

BIOGRAPHY

R.Palson Kennedy is Professor in the department of Computer Science & Engineering in P.M.R. Engineering College, Chennai-95 who enjoys challenges of creativity and attention in detail. He is a passionate researcher in the field of information retrieval, Data mining & data warehouse, Distributed & cloud computing as well. He has dedicated past seven years in the area of Distributed & Cloud Computing. His professional career covered 15 years in Teaching & 7 years Industrial sector.,