IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 5, Issue 4, April 2016*

# Smart Load Balancing Algorithm towards Effective Cloud Computing

**Vibha M B[1], Dr.R.RajuGondkar[2]**

Assistant Professor, Department of MCA, Dayananda Sagar College of Engineering, Bangalore, India [1]

Professor, Department of MCA, Brindavan College of Engineering, Bangalore, India [2]

**Abstract**: Cloud Computing has dominated the current era. Ithas been associated with varied areas and has grown tremendously. Resource sharing and effective utilisation of resources is the key to its success. In its journey, cloud has faced several challenges and has overcome many. Load balancing is one such challenge still faced by cloud.The work here depicts methods to achieve load balancing. The objective of this paper is to highlight the role of Hadoop and MapReduce to attain load balancing. The partition strategy is used to achieve load balancing. Hadoop, a Java based open source frame work is capable of storing and processing large data. MapReduce programming model is used to obtain parallel processing.

**Keywords**: Load Balancing, Partitioning, Hadoop, MapReduce, Scheduling.

## I. INTRODUCTION

Distributed systems have seen tremendous advancement in performing various operations and cloud has been efficacious to provide a flexible and scalable approach. Cloud computing is an extensive distributed computing standard and is widely adopted by organisations. The abstracted, virtualized, computing power, storage, platforms and services of cloud are delivered on demand to the users over the Internet. Google, Amazon, IBM and many others have developed their own cloud platforms which manage multiple nodes and heterogeneous applications. With the massive growth in recent years, cloud computing still poses challenges to researchers. Virtual machines create many logical resources and workstations/servers. These apart from the physical nodes are managed by Datacentres. The Datacentres hosts network's most critical systems and are vigorous to the continuity of daily tasks. The scalable property of cloud has forced these data centres to adapt optimal techniques to manage/balance the load. The objective of load balancing is to optimize resource utility, to provide high throughput, maximize response time and elude overloading of resources.

Load balancing is dividing the amount of work that a computer has to perform between two or more computers, a cluster, network links, CPU's or disk units so that more tasks are processed in the same amount of time and provide faster service [16][1]. Load balancing is the most effective way to solve the above problem in a cloud computing infrastructure, which ensures that services are delivered transparently regardless of the physical implementation and location within the "cloud" [2]. Load balancing in cloud computing provides an efficient solution to various issues pertaining to cloud computing environment set-up and usage. Load balancing must take into account two major tasks, one is the resource provisioning or resource allocation and other is task scheduling in distributed environment. Efficient allocation of resources and scheduling of resources as well as tasks will ensure resource availability on demand, effective utilization of resources, energy conservation and cost efficiency [3].

The load balancing can be achieved at local operating system by scheduling the tasks, thus improving the performance of the system. However global scheduling is required to decide the multiprocessor execution. They are categorised into static, dynamic and adaptive. Decisions taken in static load balancing algorithms use prior knowledge of the system while dynamic algorithms use system state information to take decisions [4]. Adaptive load balancing algorithms are also a type of dynamic algorithms. They dynamically change their parameters or policies that suit the changing system state [4]. An efficient load balancing algorithm will have to ascertain that every node in the system performs equal or similar task. These algorithms play a vital role in providing effective utilisation of the resources in cloud.

Enormous data is being generated across the globe every minute. The data if not handled properly creates data explosion. Social media like Facebook, Twitter, Youtube, etc. alone contributes huge mass of data, thus posing a challenge to the researchers for data handling.

The objective of this paper is to show how Hadoop can be implemented to balance the load. The paper is structured as follows in the following Sections. In section 2, a model for load balancing is put forth. Section 3, outlines theHadoop Distributed File System [HDFS] and related work for load balancing. The MapReduce programming model is presented in section 4. Section 5 discusses the

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 5, Issue 4, April 2016*

various scheduling algorithms with a comparative analysis. Section 6 discusses conclusions with future work.

## II. MODEL FOR LOAD BALANCING

Cloud comprises of nodes spanned across various geographical locations. Partitioning the cloud will help to manage its functioning. Cloud partition is a subarea of the cloud with divisions based on the geographic locations [6]. Partitions help in managing the load. The job to be processed arrives and the Partition Controller decides the partition to which the job has to be sent for processing.

The Partition Controller also finds if there is a need for any further partition. If so it is done. Then the job is assigned to the Job Balancer. The Job Balancer uses strategies to assign job. The Job Balancer has to choose appropriate partition based on the partition status, which are of three categories:

a. Idle: Degree-Load(M)=0
b. Normal:0<Degree_Load(M)<$\text{Degree\_Load}_{\text{upperlimit}}$
c. Overload: Degree_Load(M)>$\text{Degree\_Load}_{\text{upperlimit}}$
d. Breakdown of Partition.

where parameters are set by Job balancer, M=No. of CPU cycles and process execution time involved, Degree_Load= Threshold limit [6].

The Partition Controller dispatches the jobs to the partitions. In case of idle and normal situations, the job is handled in the same partition. In case of overload, a different partition in looked for and the tasks are distributed to them. The fourth state is handled at the time of physical failure of partition.

The jobs are assigned by gathering status of information from each node. The degree of the load is based on various static and dynamic parameters. The physical entities like CPU speed and utilisation, disk memory, network bandwidth etc. constitutes static parameters.

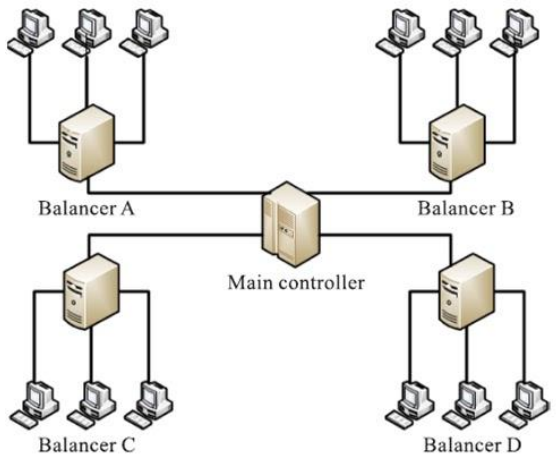


Fig1: Relation between Partition Controller and Job Balancer.

The Partition Controller and Job Balancer are in communication to determine the suitable partition. The Job Balancer here uses MapReduce framework to break the tasks and perform parallel processing.

## III.HADOOP FRAMEWORK

Hadoop is a Java based open source framework. Hadoop is used by many companies like Yahoo, Facebook and Amazon to process their big data generated everyday [17]. Hadoop is employed to deal with large amounts of data, called Big Data, as it provides users a reliable storage with the help of Hadoop Distributed File System and adopts the MapReduce model to process the data in the Hadoop cluster [5]. The three major classifications of roles in a Hadoop deployment are Client, Master nodes and Slave nodes. The two core parts of Hadoop are Hadoop Distributed File System (HDFS) and MapReduce framework. The HDFS is used for storing voluminous data, which the normal RDBMS has failed to and MapReduce is used for parallel processing of data. The two key phases HDFS and MapReduce, uses Master nodes to coordinate with their respective slave nodes. The Name Node coordinates with data storage (HDFS), while the Job Tracker Master allows parallel computations on all the data (MapReduce). Slave nodes are Data Node coordinating with Name Node and Task Tracker coordinating with Job Tracker. They execute the task designated by their Masters [18].

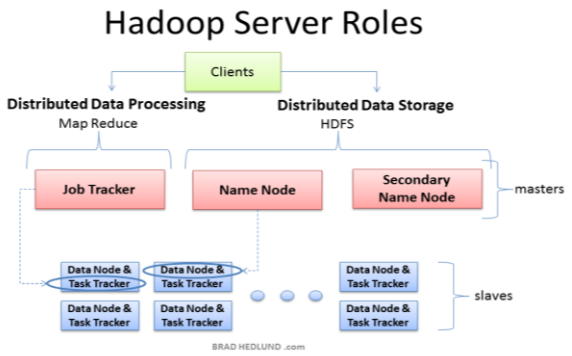


Fig 2: Hadoop server Roles

The data blocks are located on data nodes and Hadoop divides data nodes on different racks. Hadoop installed on a very large cluster can span many data centres, which consists of multiple racks. The clusters are designed for storing and analysing huge amount of unstructured data in distributed environment. The data is loaded to the cluster (HDFS writes), analysed (MapReduce), Store the results in cluster (HDFS writes) and read from the cluster [18]. The client breaks the data into three blocks and consults the Name Node to send the data. It receives a list of three data nodes that having the replica of this block. The client writes the block to the Data Node, which is located on the local rack. That Data Node in turn replicates two other Data Nodes located on another rack. The intention of having multiple copies is that, in case there is a failure in one rack, other racks can resume its task. The Name Node

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 5, Issue 4, April 2016*

provides the map of where data is located and where the data should go in the cluster. The data blocks of Hadoop are large and default block size is 64 megabytes. The data node sends periodical report to the Name Node about the available blocks, its location, storage capacity and transfer latency. This is known as Heartbeat from a data node. Receiving the heartbeat, the Name Node instructs the operations. Currently Hadoop is being used to develop and schedule programs. The advantage of having Hadoop for load balancing is that, it is capable of performing parallel processing using MapReduce function. Hadoop is capable of providing stable, scalable and reliable interface to the users. It can run applications in the cluster that consists of a large number of inexpensive hardware. The key scheduling of Hadoop is Task scheduling, which will control the tasks and resources thus improving the performance of the system.

TABLE I: COMPONENTS OF HADOOP SYSTEM

| Node Type | Task |
|---|---|
| Name Node | It provides the meta data and consists of the information about location/address of the data blocks. |
| Data Node | Sends heart beat to the Name Node in a ordered time interval |
| Job Tracker | Responsible to schedule, allocate and monitor job execution on slave Task Tracker |
| Task Tracker | Follow the instructions of Job Tracker using MapReduce functions. |

## IV. MAPREDUCE PROGRAMMING MODEL

Voluminous data has to be managed by dividing the work into set of independent tasks, thus making it possible for parallel computation. MapReduce is a programming model designed for processing large volumes of data. It is a style of parallel programming that is supported by some capacity-on-demand-style [19].A MapReduce breaksthe input data-set into independent pieces which are processed by the *map tasks* in a parallel manner [20]. The first function Map picks up a chunk with a key/value pair. It in turn generates intermediate key/value pair and then this is sorted based on the key/value and delivers onto Reduce function. The Reduce function combines these values to produce a set of values. The value generated per Reduce phase is usually one or zero output. Map phase is meant to divide the job to tasks and Reduce function assembles the results of tasks to produce a final outcome. There are three steps in MapReduce[20]

Step1: Mapper - Input to Mapper is set in the Driver program of a particular Input Format type and file(s) on which the Mapper process has to run. The output of Mapper will be a map <*key*, *value*>, key and value set in Mapper output is not saved in HDFS, but an intermediate file is created in the OS space path and that file is read and shuffle and sorting takes place.

Step 2: Shuffling and Sorting- Shuffle and sort are intermediate steps in MapReduce between Mapper and Reducer, which is handled by Hadoop and can be overridden if required. The Shuffle process aggregates all the Mapper output by grouping key values of the Mapper output and the value will be appended in a list of values. So, the Shuffle output format will be a map <key, *List<list of values>>* [20].

Step3: Reducer- Reducer is the aggregator process where data after shuffle and sort, is sent to Reducer where we have <*key*, *List<list of values>>*, and Reducer will process on the list of values. Each key could be sent to a different Reducer. Reducer can set the value, and that will be consolidated in the final output of a MapReduce job and the value will be saved in HDFS as the final output [20].

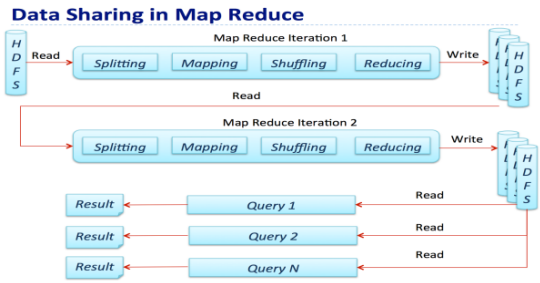


Fig 3: MapReduce Phases

The framework sorts the outputs of the maps, which are then input to the *reduce tasks*. Typically both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks. The typical MapReduce program will contain three different components namely: a driver class, a mapper class and a reducer class. The pseudo code is shown below[21]:

**map(*InputKeyType* inputKey, *InputValueType* inputVal ue):**

//do processing on the *inputKey* and *inputValue* and form *intermediateKey* , *intermediateValue* pair
Emit(intermediateKey, intermediateValue);
// map can emit more than one intermediate key-value pairs

**reduce(*IntermediateKeyType* intermediateKey, *Iterator* values):**

//the values associated with a particular intermediateKey is iterated and a user-specified operation is performed over the values
// multiple reducers can run in parallel and the number of reducers is specified by the user
//*outputKey* will contain the key output from the reducer and *outputValue* will contain the value that is output for that particular key
Emit(outputKey, outputValue);
// reduce method can emit more than one output key-value pairs

To optimize the processing capacity of the **map** phase, MapReduce can run several identical mappers in parallel.

Since every mapper is the same, they produce the same result as running one map function [21].

## V. COMPARATIVE ANALYSIS OF VARIOUS ALGORITHMS

To compare the performance of algorithms, we have to consider three basic factors namely locality, fairness and synchronization. Locality is the distance between input data node and task designated node. Fairness is the trade-offs between the locality and dependency between the maps and reduce phases. Synchronization is the process of transmitting the intermediate output of the map processes to the reduce processes as input [7]. The task scheduling activities is directly affecting the system optimization of Hadoop and system resources utilization. Several approaches have been tried to provide solutions. Each algorithm will have its own pros and cons.

### *A. FIFO Scheduling*

This is the simplest algorithm used by Hadoop which operates on the principle of a queue. This is a straight forward and non-pre-emptive algorithm. It can be used on small clusters, where prior knowledge of the system is known. In this method, the actual job is partitioned into individual jobs and loaded to the queue. The Task Tracker assigns the jobs to the free blocks that are available. The jobs dominate the cluster, complete the task assigned and then passes the control to the next job. The subsequent jobs in the queue have to wait until the previous job passes the control. In other words it is sequential execution. This is a disadvantage because, the priority based and short jobs have to wait for their turn. The main advantage of this scheduling is its implementation is simple.

### *B. Fair Scheduling*

This algorithm was developed by Facebook to manage the access to their Hadoop cluster [8]. Fair scheduling is a method of assigning resources to jobs such that all jobs get, on average, an equal share of resources over time. When there is a single job running, that job uses the entire cluster. When other jobs are submitted, tasks slots that free up are assigned to the new jobs, so that each job gets roughly the same amount of CPU time. It is also an easy way to share a cluster between multiple of users. Fair sharing can also work with job priorities - the priorities are used as weights to determine the fraction of total compute time that each job gets. The fair scheduler organizes jobs into *pools*, and divides resources fairly between these pools. This can be useful when jobs have a dependency on an external service like a database or web service that could be overloaded if too many map or reduce tasks are run at once. [22]. It is pre-emptive algorithm. Short jobs are given more attention to complete the task quickly and longer jobs are also taken care fairly.

### *C. Capacity Scheduling*

Capacity planning is the process of determining the production capacity needed by an organization to meet changing demands for its products. [23]. This has been the traditional and vertical way of scaling up web applications, however IT capacity planning has been developed with the goal of forecasting the requirements for this vertical scaling approach[24] This algorithm's aim is to manage a fair scheduling among huge mass of users.

The Capacity Scheduler allocates jobs based on the submitting user to queues with configurable numbers of Map and Reduce slots [9]. When a slot is free, the Task Tracker selects the lowest load, through which the oldest job was designated. This scheduling enforces cluster's capacity shared among users than between the jobs.

### *D. Longest Appropriate Time to End(LATE)*

LATE was proposed by Zaharia et al[10].This functions MapReduce in heterogeneous environment. It has robustly improved the system performance when compared to classical algorithms.

LATE scheduler has made an attempt to improve Hadoop by identifying slow tasks. It rates the jobs by estimated time remaining and starts a copy of the highest ranked task that has a progress rate lower than the Slow Task Threshold[11]. LATE's methodology is based on prioritizing tasks to speculate, Locating fast nodes to execute and overlaying speculative tasks to avoid thrashing[7]. The advantage of this algorithm is robustness and heterogeneity.

### *E. Self Adaptive Map Reduce[SAMR]*

LATE fails to identify real slow tasks. SAMR improvises MapReduce and effectively uses system resources. SAMR can be used to improve accountability for data locality. SAMR decreases the total execution time up to 25% compared with Hadoop's scheduler and 14% compared to LATE scheduler[12].

### *F. Delay Scheduling*

Delay scheduling is used to improve data locality by asking jobs to wait for its turn for scheduling on a node with local data. In this scheduling, if the head-of-line cannot launch a local task, it is skipped and looks out for subsequent tasks. Non-local jobs are allowed to launch in case the local node has missed the jobs, to avoid starvation.

### *G. Ant Colony Optimization(ACO)*

The algorithm works based on the heuristic approach. Whenever a request is initiated, the ants and pheromones are initiated. The ants move towards the head node.

The forward movement indicates the ant is moving from the overloaded node to the other by verifying whether the other node is overloaded or not. If the Ant finds under loaded node, it will continue forward movement towards head, else it will take a backward movement to get to the previous node. The Ant commits suicide, when it reaches the Target node to avoid further backward movements.

A comparison of the algorithms discussed above is summarised below:

TABLE III: COMPARISON OF ALGORITHM

| Scheduling Method/Factors | FIFO | Fair Scheduling | Capacity | Delay Scheduling | ACO | LATE | SAMR |
|---|---|---|---|---|---|---|---|
| Implementation | Simple. Schedule the jobs based on the arrival | Less Complex. Equal distribution of resources among users | Complex. Maximum users, high throughput among Multi-tenant | Simple Conflict resolution between locality and fairness | Complex. Heuristic. Colony based approach | Less Complex Robust, heterogeneous | Improved MapReduce |
| Taxonomy | Non-Adaptive | Adaptive | Adaptive | Adaptive | Heuristic | Adaptive | Adaptive |
| Utilization of Resource | Low | High | High | High | High | High for slow tasks | High |
| System performance | High for small clusters | High for both small and large clusters | High for large clusters only | High for small clusters | High for small clusters | High for slow tasks in small and large clusters | High for both small and large clusters |
| Fairness | No | Yes | Yes | Yes | Yes | Yes | Yes |
| Load Balancing | No | Yes | Yes | Yes | Yes | Yes | Yes |
| Type of Job Processing | Single | Multiple Jobs | Multiple Jobs | Multiple Jobs | Multiple Jobs | Multiple Jobs | Multiple jobs |

## VI.CONCLUSION

The above work, addresses the complexity of load balancing by partitioning the cloud. The partitioned cloud is managed by the Partition Controller and Job Balancer. The work envisages the open source Hadoop framework to handle and control data. It uses MapReduce programming model to handle parallel processing of data. Importance of Hadoop's HDFS is also highlighted in order to store data. A comparative study of MapReduce algorithms is made. The choice of the algorithm depends on the locality, fairness and the size of the cluster. The future work will be to concentrate in creating a cluster and schedule the tasks using the appropriate MapReduce algorithm.

## ACKNOWLEDGMENT

I acknowledge Bharathiar University, Coimbatore for providing me an opportunity to carry out research in the field of computer science.

## REFERENCES

1. Suguna R, DivyaMohandass, Ranjani R, "A Novel Approach For Dynamic Cloud Partitioning And Load Balancing In Cloud Computing Environment", Department of CSE,SKR Engineering college, ISSN: 1992-8645,
2. RenGao and et al, "Dynamic Load Balancing Strategy for Cloud Computing with Ant Colony Optimization" School of Information Engineering, Hubei University of Economics, China, Future Internet 2015, 7(4), Page No. 465-483
3. MayankaKatyal*, Atul Mishra , "A Comparative Study of Load Balancing Algorithms in Cloud Computing Environment", Published online in www.publishingindia.com
4. Niranjan G. Shivaratri, Phillip Krueger, and MukeshSinghal"Load Distributing for Locally Distributed Systems", volume: 25, Issue:12 ,ISSN:0018-9162,Ohio State University.
5. XiaofeiHou, Ashwin Kumar T K, Johnson P Thomas "Dynamic Workload Balancing for Hadoop MapReduce", Computer Science Department Oklahoma State University Stillwater, OK, USA.
6. GaochaoXu, Junjie Pang, and Xiaodong Fu, "A Load Balancing Model Based on Cloud Partitioning for the Public Cloud", Tsinghua Science And Technology ISSN ll1 0 07 - 0 214ll0 4/12llp p 4-3 9 Volume 18, Number 1, February 2013.
7. Seyed Reza Pakize,"A Comprehensive View Of Hadoop Map Reduce Scheduling Algorithms",International Journal of computer networks and communications security,Vol. 2,Issue 9,pp.308-317,September 2014
8. B. ThirmalaRao, N. V. Sridevei, V. Krishna Reddy, LSS.Reddy, "Performance Issues of Heterogeneous Hadoop Clusters in Cloud Computing", Global Journal Computer Science & Technology Vol. 11, Issue 8, pp.81-87,May 2011
9. "Terms & Definitions - Supply Chain Management". North Carolina State University. 2006. Retrieved 2008-10-2
10. wikipedia.org/wiki/Capacity_planning
11. Dean, J. and Ghemawat, S., "MapReduce: a flexible data processing tool", communication of ACM,Vol. 53,Issue 1,pp72-77,January 2010
12. M. Zaharia, A. Konwinski, A.D. Joseph, R. Katz and I. Stoica, "Improving MapReduce performance in heterogeneous environments " In: OSDI 2008: 8th USENIX Symposium on Operating Systems Design and Implementation 2008. [24]Q. Chen, D. Zhang, M. Guo, Q. Deng Q and S.
13. B.P Andrews and A. Binu, " Survey on Job Schedulers in Hadoop Cluster ", IOSR Journal of Computer Engineering, Vol.15, NO. 1, Sep. - Oct. 2013, pp. 46-50
14. S. Khalil, S.A. Salem, S. Nassar and E.M. Saad, "Mapreduce Performance in Heterogeneous Environments: A Review", International Journal of Scientific & Engineering Research, Vol. 4, NO. 4, April - 2013, pp. 410-416
15. Vijay Vardan, Dept. of Computing, Macquarie University, Sydney, Australia, proceedings 2014 IEEE Fourth International Conference on Big Data and Cloud Computing
16. K. Shvachko, H. Kuang, S. Radia, R. Chansler, "The Hadoop Distributed File System", in Proceedings of IEEE Conference on Mass Storage Systems and Technologies (MSST), 2010.
17. Radojevic, B. and M. Zagar, "Analysis of issues with load balancing algorithms in hosted (cloud) environments." in proc.34th International Convention on MIPRO, IEEE, 2011.
18. Searchnetworking.techtarget.com/definition/load-balancing
19. T. White. Hadoop: "The Definitive Guide O'Reilly Media, Inc, 2010
20. hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html#MapReduce
21. www.ibm.com/developerworks/cloud/library/cl-mapreduce/
22. howtodoinjava.com/big-data/hadoop/hadoop-mapreduce-tutorial-for-beginners/
23. https://hadooptutorial.wikispaces.com/MapReduce
24. https://hadoop.apache.org/docs/r1.2.1/fair_scheduler.html
25. www.hadooptutorial.wikispaces.com/MapReduce

## BIOGRAPHY

**Mrs. Vibha M B** is an academician with over 11 years of experience. She is currently working in Dayananda Sagar College of Engineering, Bangalore and pursuing Ph.D from Bharathiar University Coimbatore, India.


DOI 10.17148/IJARCCE.2016.54135