# A Novel Load-Balancing Algorithm for Distributed Systems

Parvati Rajendran
*School of Computing Science and Engineering, VIT University, Vellore, India*

Shalinee Singh
*School of Computing Science and Engineering, VIT University, Vellore, India*
s

K Ramesh Babu
*School of Computing Science and Engineering, VIT University, Vellore, India*

## Abstract

*Distributed file systems are key building blocks for cloud computing applications based on the Map Reduce programming paradigm. Load balance among storage nodes is a critical function in clouds. In a load-balanced cloud, the resources can be well utilized and provisioned, maximizing the performance of Map Reduce-based applications. In such a distributed file system, the load of a node is typically proportional to the number of file chunks the node possesses. In this paper, a fully distributed load rebalancing algorithm is presented to cope with the load imbalance problem. The proposed algorithm is compared against a centralized approach in a production system and strives to balance the loads of nodes and reduce the demanded movement cost as much as possible, while taking advantage of physical network locality and node heterogeneity.*

## 1. Introduction

Cloud Computing became very popular in the last few years. As part of its services, it provides a flexible and easy way to keep and retrieve data and files. Especially for making large data sets and files available for the spreading number of users around the world. Handling such large data sets require several techniques to optimize and streamline operations and provide satisfactory levels of performance for the users. Therefore, it is important to research some areas in the Cloud to improve the storage utilization and the download performance for the users. One important issue associated with this field is dynamic load balancing or task scheduling. Load balancing algorithms were investigated heavily in various environments; however, with Cloud environments, some additional challenges are present and must be addressed. In Cloud Computing the main concerns involve efficiently assigning tasks to the Cloud nodes such that the effort and request processing is done as efficiently as possible, while being able to tolerate the various affecting constraints such as heterogeneity and high communication delays.

Load balancing algorithms are classified as static and dynamic algorithms. Static algorithms are mostly suitable for homogeneous and stable environments and can produce very good results in these environments. However, they are usually not flexible and cannot match the dynamic changes to the attributes during the execution time. Dynamic algorithms are more flexible and take into consideration different types of attributes in the system both prior to and during run-time. These algorithms can adapt to changes and provide better results in heterogeneous and dynamic environments. However, as the distribution attributes become more complex and dynamic. As a result some of these algorithms could become inefficient and cause more overhead than necessary resulting in an overall degradation of the services performance.

## 2. Load Balancing in Cloud computing

It is a process of reassigning the total load to the individual nodes of the collective system to make resource utilization effective and to improve the response time of the job, simultaneously removing a condition in which some of the nodes are over loaded while some others are under loaded. A load balancing algorithm which is dynamic in nature

does not consider the previous state or behaviour of the system, that is, it depends on the present behaviour of the system. The important things to consider while developing such technique are : estimation of load, comparison of load, stability of different system, performance of system, interaction between the nodes, nature of work to be transferred, selecting of nodes and many other ones. This load considered can be in terms of CPU load, amount of memory used, delay or Network load.

Goals of Load balancing:

- To improve the performance substantially
- To have a backup plan in case the system fails even partially
- To maintain the system stability
- To accommodate future modification in the system

## 3. Existing Load Balancing Techniques In Clouds

Following load balancing techniques are currently prevalent in clouds

**3.1 VectorDot-** A. Singh et al. proposed a novel load balancing algorithm called VectorDot. It handles the hierarchical complexity of the data-center and multidimensionality of resource loads across servers, network switches, and storage in an agile data center that has integrated server and storage virtualization technologies. VectorDot uses dot product to distinguish nodes based on the item requirements and helps in removing overloads on servers, switches and storage nodes.

**3.2 CARTON-** R. Stanojevic et al. proposed a mechanism CARTON for cloud control that unifies the use of LB and DRL. LB (Load Balancing) is used to equally distribute the jobs to different servers so that the associated costs can be minimized and DRL (Distributed Rate Limiting) is used to make sure that the resources are distributed in a way to keep a fair resource allocation. DRL also adapts to server capacities for the dynamic workloads so that performance levels at all servers are equal. With very low computation and communication overhead, this algorithm is simple and easy to implement.

**3.3 Compare and Balance-** Y. Zhao et al. addressed the problem of intra-cloud load balancing amongst physical hosts by adaptive live migration of virtual machines. A load balancing model is designed and implemented to reduce virtual machines' migration time by shared storage, to balance load amongst servers according to their processor or IO usage, etc. and to keep virtual machines' zero-downtime in the process. A distributed load balancing algorithm COMPARE AND BAL-ANCE is also proposed that is based on sampling and reaches equilibrium very fast. This algorithm assures that the migration of VMs is always from high-cost physical hosts to low-cost host but assumes that each physical host has enough memory which is a weak assumption.

**3.4 Event-driven-** V. Nae et al. presented an event-driven load balancing algorithm for real-time Massively Multiplayer Online Games (MMOG). This algorithm after receiving capacity events as input, analyzes its components in context of the resources and the global state of the game session, thereby generating the game session load balancing actions. It is capable of scaling up and down a game session on multiple resources according to the variable user load but has occasional QoS breaches.

**3.5 Scheduling strategy of VM resources –**

J. Hu et al. proposed a scheduling strategy on load balancing of VM resources that uses historical data and current state of the system. This strategy achieves the best load balancing and reduced dynamic migration by using a genetic algorithm. It helps in resolving the issue of load imbalance and high cost of migration thus achieving better resource utilization.

## 4. Types of Load balancing algorithms

There are many load balancing algorithms, such as Round Robin, Equally Spread Current Execution Algorithm, and Ant Colony algorithm. Nishant et al. used the ant colony optimization method in nodes load balancing. Randles et al. gave a compared analysis of some algorithms in cloud computing by checking the performance time and cost. They concluded that the ESCE algorithm and throttled algorithm are better than the Round Robin algorithm. Some of the classical load balancing methods are similar to the allocation method in the

operating system, for example, the Round Robin algorithm and the First Come First Served (FCFS) rules. The Round Robin algorithm is used here because it is fairly simple. Depending on who initiated the process, load balancing algorithms can be of three catagories as given in :

**4.1.1 Sender Initiated:** If the load balancing algorithm is initialised by the sender

**4.1.2 Receiver Initiated:** If the load balancing algorithm is initiated by the receiver

**4.1.3 Symmetric**: It is the combination of both sender initiated and receiver initiated

Depending on the current state of the system, load balancing algorithms can be divided into 2 categories as given in :

**4.2.1 Static:** It doesn't depend on the current state of the system. Prior knowledge of the system is needed

**4.2.2 Dynamic**: Decisions on load balancing are based on current state of the system. No prior knowledge is needed. So it is better than static approach.

## 5. Algorithm for Dynamic Load balancing in Distributed Systems

$N_i$: List of primary nodes
$SN_j$: List of supporting nodes, j<i
PI: Priority index maintained by supporting node
$P_k$: List of processes in Process Queue

Assumption
1.Each node N is having some load,
2. Some supporting nodes SN may have load
$N_i \leftarrow$ Load, $S_j \leftarrow$ Load

Procedure: Main ( )
{
1. Suppose a node $N_t$ is heavily loaded with load £, where 0< t< i
// Two situations are possible as in step 2
2. If ( Search_node ) // Search_node will find out whether a light weighted primary node is available
{
Available_Node ←£ // Overload is assigned to Available node given by Search_node ()
Load ($N_t$) =Load ($N_t$) – £
}
Else
{
Call Search_S_node ( ) // Search_node will find out a light weighted or minimum weighted supporting node with index as Available_S_node
Available_s_node ← £
Load ($N_t$) =Load ($N_t$) – £
IN_S (Available_s_node, £)
}
}

Procedure: IN_S (SN_node, £)
{
1. Priority is assigned to SN_node as PI (SN_node) = t
2. If t > PI (RP) // RP is the currently process on selected supporting node.
{
// P_List is list of pending process maintained according to priority.
P_List ← RP // RP is added to Pending List
Now RP ← £ // £ is now allotted to the selected Supporting node.
}
Else
P_List ← £ // £ is added to Pending List
}.
}

Procedure: Search _node ()
{
1. for each $N_i$ except node initiating the search_node procedure
{
Check the node with minimum load
//Minimum load includes the number of processes as well as structure or configuration of node and node should be able to accept node
}
2. If Desired Node available
{
return (index of available node) // index defines the property to identify the Node
}
}

Procedure: Search _S_node ()
{
1. for each $SN_j$ except node initiating the search_node procedure
{

Check the Supporting node with minimum load// Minimum load includes the number of processes as well as structure or configuration of node
}
2. return (index of available Supporting node) // index defines the property to identify the Node
}

## 6. Distributed Load Balancing for the Clouds

### 6.1 Honeybee Foraging Algorithm

In case of load balancing, as the web servers demand increases or decreases, the services are assigned dynamically to regulate the changing demands of the user. The servers are grouped under virtual servers (VS), each VS having its own virtual service queues. Each server processing a request from its queue calculates a profit or reward, which is analogous to the quality that the bees show in their waggle dance. One measure of this reward can be the amount of time that the CPU spends on the processing of a request. The dance floor in case of honey bees is analogous to an advert board here. This board is also used to advertise
the profit of the entire colony.

### 6.2 Biased Random Sampling

Here a virtual graph is constructed, with the connectivity of each node (a server is treated as a node) representing the load on the server. Each server is symbolized as a node in the graph, with each indegree directed to the free resources of the server. Regarding job execution and completion, whenever a node does or executes a job, it deletes an incoming edge, which indicates reduction in the availability of free resource. After completion of a job, the node creates an incoming edge, which indicates an increase in the availability of free resource.

### 6.3 Active Clustering

Active Clustering works on the principle of grouping similar nodes together and working on these groups. The process involved is:

- A node initiates the process and selects another node called the matchmaker node from its neighbours satisfying the criteria that it should be of a different type than the former one.
- The so called matchmaker node then forms a connection between a neighbour of it which is of the same type as the initial node.
- The matchmaker node then detaches the connection between itself and the initial node.

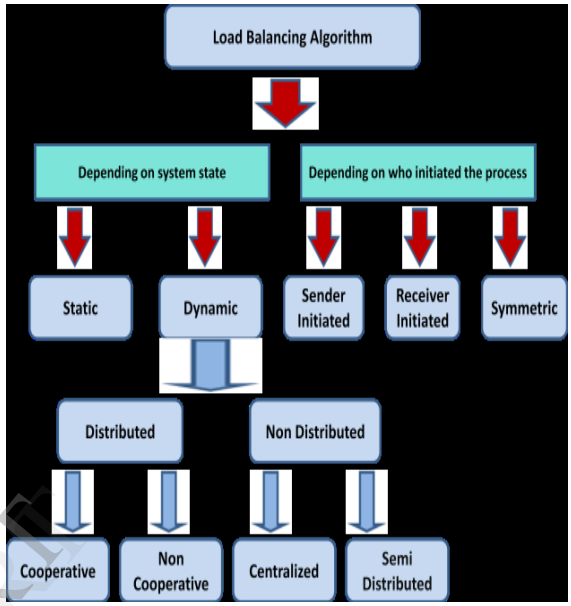


**Fig.1: Classification of Load balancing algorithms**

## 7. Cloud partitioning

There are several cloud computing categories with this work focused on a public cloud. A public cloud is based on the standard cloud computing model, with service provided by a service provider. A large public cloud will include many nodes and the nodes in different geographical locations. Cloud partitioning is used to manage this large cloud. A cloud partition is a subarea of the public cloud with divisions based on the geographic locations. The architecture is shown in Fig. The load balancing strategy is based on the cloud partitioning concept. After creating the cloud partitions, the load balancing then starts: when a job arrives at the system, with the main controller deciding which cloud partition should receive the job. The partition load balancer then decides how to assign the jobs to the nodes. When the load status of a cloud partition is normal, this partitioning can be accomplished locally. If the cloud partition load status is not normal, this job should be transferred to another partition.

## 8. Main controller and balancers

The load balance solution is done by the main controller and the balancers. The main controller first assigns jobs to the suitable cloud partition and then communicates with the balancers in each partition to refresh this status information. Since the main controller deals with information for each partition, smaller data sets will lead to the higher processing rates. The balancers in each partition gather the status information from every node and then choose the right strategy to distribute the jobs. The relationship between the balancers and the main controller is shown in Fig.1

### 8.1 Assigning jobs to the cloud partition

When a job arrives at the public cloud, the first step is to choose the right partition. The cloud partition status can be divided into three types:

**8.1.1 Idle**: When the percentage of idle nodes exceeds and change to idle status.

**8.1.2 Normal**: When the percentage of the normal nodes exceeds and changes to normal load status.

**8.1.3 Overload**: When the percentage of the overloaded nodes exceeds, change to overloaded status.

The parameters are set by the cloud partition balancers. The main controller has to communicate with the balancers frequently to refresh the status information. The main controller then dispatches the jobs using the following strategy: When job i arrives at the system, the main controller queries the cloud partition where job is located. If this location's status is idle or normal, the job is handled locally. If not, another cloud partition is found that is not overloaded.

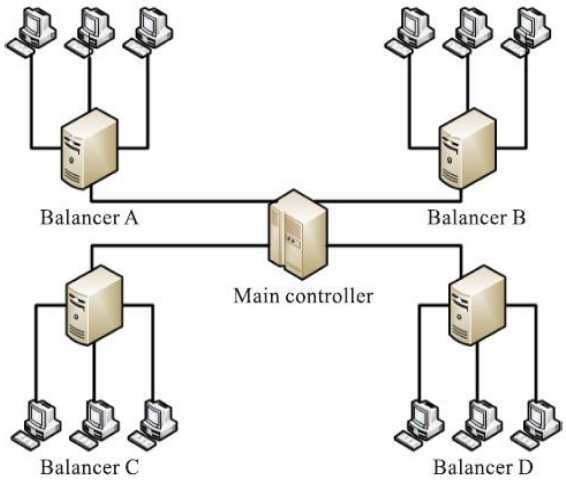


**Fig.2 Controllers and Balancers**

## 9. Conclusion

In order to balance the load uniformly over a cluster system, one has to choose a mix of centralized, decentralized, approach. This communication overhead and load balancing time depends upon the approach selected in the algorithm. The simulator gives better result for new algorithm. In future, this work can be extended to develop a new algorithm to modify dynamic decentralized approach so as to reduce the communication overhead as well as to reduce migration time and also make it scalable.

## 10. References:

[1] Sun Nian1, Liang Guangmin2 "Dynamic Load Balancing Algorithm for MPI Parallel Computing" 2010 Pages 95 to 99

[2] Janhavi B,Sunil Surve ,Sapna Prabhu "Comparison of load balancing algorithms in a Grid" 2010 International Conference on Data Storage and Data Engineering Pages from 20 to 23.

[3] Yongzhi Zhu Jing Guo Yanling Wang "Study on Dynamic Load Balancing Algorithm Based on MPICH" 2009 MPI_COMM_RANK: World Congress on Software Engineering. Pages from 103 to 107.

[4] Yanyong Zhang, Anand Sivasubramaniam, JoseÂ Moreira, and Hubertus Franke" Impact of Workload and System Parameters on Next Generation Cluster Scheduling Mechanisms" 2001 IEEE transactions on parallel and distributed systems Pages from 967 to 985.

[5] Erik D. Demaine, Ian Foster,Carl Kesselman, and Marc Snir. "Generalized Communicators in the Message Passing Interface" 2001 IEEE transactions on parallel and distributed systems pages from 610 to 616.

[6] Bernd F reisleben Dieter Hartmann Thilo Kielmann "Parallel Raytracing A Case Study on Partitioning and Scheduling on Workstation Clusters" 1997 Thirtieth Annual Hawwaii International Conference on System Sciences.

[7] Sharada Patil,Arpita Gopal "Comparison of Cluster Scheduling Mechanism using Workload and System Parameters" 2011 ISSN 0974- 0767 International journal of Computer Science and Application.

[8] Parimah Mohammadpour,Mohsen Sharifi, Ali Paikan," A Self-Training Algorithm for Load Balancing in Cluster Computing", 2008 IEEE Fourth International Conference on Networked Computing and Advanced Information Management , Pages from 104 to 110.

[9] Marta Beltr´an and Antonio Guzm´an " Designing load balancing algorithms capable of dealing with workload variability ", 2008 International Symposium on Parallel and Distributed Computing pages from 107 to 115.

[10] Paul Werstein, Hailing Situ and Zhiyi Huang „Load Balancing in a Cluster Computer" 2006 Proceedings of the Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies.

[11] Hau Yee Sit Kei Shiu Ho Hong Va Leong Robert W. P. Luk Lai Kuen Ho " An Adaptive Clustering Approach to Dynamic Load Balancing∗ ",2004 IEEE 7th International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'04)

[12] Marta Beltr´an and Antonio Guzm´an "Designing load balancing algorithms capable of dealing with workload variability" 2008 International Symposium on Parallel and Distributed Computing Pages from 107 to 114.

[13] Sharada Patil,Arpita Gopal "Comparison of Cluster Scheduling Mechanism using Workload and System Parameters" 2010 International Journal of Computer Science and Application (IJCSA ISSN: 0974-0767) Pune in Dec 2010

[14] Sharada Patil,Arpita Gopal "Study of dynamic load balancing algorithms for linux clustered system using simulator" 2011 ISSN 0974-3588 International journal of Computer Applications in Engineering Technology and Sciences.

[15] Mrs. Sharada Patil,Dr Arpita Gopal,2012 "Need Of New Load Balancing Algorithms For Linux Clustered System" – International Conference on Computational techniques And Artificial intelligence (ICCTAI'2012) (ISBN 978-81-922428-5-9) Penang Maleshia in year Jan 2012