Volume 5, Issue 4, 2015 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Memory Intensive Technique for Load Balancing in the Cloud

Ankita J. Betai[*]
Dept. of CSE, DBIT
Bangalore, India

Bhaskar R
Dept. of CSE, DBIT
Bangalore, India

Poornanand P. Naik
Foal Technologies
Bangalore, India

Abstract— A Load Balancer provides techniques by which instances of applications, storage units, processing units and hardware configuration can be provisioned and de-provisioned (released) automatically, without requiring changes to the Cloud configuration. It automatically checks and handles the increases and decreases in capacity and adapts its distribution decisions dynamically based on the capacity available at the time a request is made, that is, when a job arrives or a file is uploaded. This paper focuses on the need to consider each individual node's capacity and allocate the jobs based on the capacity which is left-out in the node, after comparison with the threshold value. The Node Status: Idle, Normal and Overloaded, is determined based on the comparison between Left-out capacity and Threshold value. The Left-out capacity indicates the memory which is available after deducting the total used capacity (Total size occupied) from the Total memory which is available for storage. The aim here is to ensure that the incoming job requests are evenly distributed to the resources based on their handling capacity and none of the systems crash. It ensures that the new incoming job is dispatched to the node with the highest Left-out capacity, thus refreshing the node status, at each iteration and resulting in the workload being evenly distributed based on the individual differing capacities of the nodes, so that one common implementation mechanism does not overload any of the nodes. It provides fault tolerance when coupled with a failover mechanism. This dynamic approach ensures high performance, less complexity, high throughput, increase in operational efficiency and high availability computing.

Keywords— Load Balancer, Node Status, Left-out Capacity, Total-Capacity, Threshold value

I. INTRODUCTION

Cloud Computing is based on the availability, usage, provisioning, virtualization and consumption of computing resources. It involves deploying and provisioning, clusters of remote as well as local servers, applications, hardware units, storage units and software networks that allow centralized as well as distributed data storage and online access to computer services or resources, on demand, on a 'pay-per-use' basis. It relies on dynamic sharing of resources and multi-tenancy to achieve coherence and economies of scale. Its foundation is based on the concept of consolidation of application licenses, converged infrastructure, shared services and resource automation. Cloud computing, or just in brief "the cloud", also focuses on providing visibility, control, data security, integration, interoperability and automation across all the business and IT assets to deliver higher value services [11] by maximizing the usage and effectiveness of the shared resources. Cloud resources are shared by multiple users, available on heterogeneous thin or thick client platforms, scale according to the usage and are dynamically reallocated or reprovisioned as per demand. This serves as the foundation stone for allocating resources to users. For example, a cloud facility that serves Indian economy during Indian business hours with a specific service or application (e.g., email, transaction) may reallocate by reprovisioning the same resources to serve British economy during British business hours with a different set of applications (e.g., a web server, online simulation tool). This approach aims to maximize the use of computing power as well as available resources, thus reducing environmental damage as well as the load on particular specific servers which remain in the working state throughout. With the advent of cloud computing, multiple users can access a single server to retrieve, modify and work upon their data without purchasing licenses for different applications by just using the concept of plug-in and play. Cloud computing means "a type of Internet-based computing," where different services such as hardware, servers, storage and applications are delivered to an organization's computers and devices through the Internet. Almost every Cloud has core characteristics like, Virtualization, Flexibility, Scalability, Security, Affordability, Agility, Device and Location Independence, Maintenance, Multi-tenancy, Reliability, Self-service Provisioning, Elasticity, Pay per use, On-demand self-service, broad network access, Resource pooling, measured service and Increased Productivity.

A Cloud consists of numerous computing resources like computers, a computer cluster, hardware devices, network links, central processing units and disk drives. In computing, Load Balancing distributes workloads across these multiple computing resources. Formally, Load Balancing can be defined as, dividing the amount of work that a computer has to do, between two or more computers so that more work gets done in the same amount of time and all users get served faster.[9], [10] Load Balancing can be implemented with hardware, software or a combination of both[9], [10], [13]. Load Balancing is the main reason for computer server clustering[9], [10]. On the Internet, companies whose web sites get a great deal of traffic, usually use Load Balancing. For Web Serving, one approach is to route each request in turn to a different server host address in a domain name system (DNS) table, [9], [10], [12], [13] usually referred to as the Round Robin Technique. Usually, if there are several servers which would be used to balance a work load, another server also known as a controller is needed to determine which server to assign the work to. Since Load Balancing requires multiple servers, it is usually combined with failover and backup servers [10], [12], [13]. In some schemes, the servers or the nodes are haphazardly distributed over different geographic locations.

Load balancing has become a necessity because multiple servers servicing one application can quickly be overwhelmed, overloaded and would crash if the workload is not split up or broken down. This avoids a situation where in which some of the servers are over utilized beyond their capacity and some of them are underutilized. Load Balancing is agent less and platform- and protocol-independent. Load Balancers are present in front of the servers (could be nodes or physical systems) to divide carefully the incoming traffic load. Companies whose servers are constantly reaching at the maximum usage level during peak usage times are good candidates for load balancing.

II. Proposed Method

Parameters such as Processing Speed of the CPU, Available Memory, Bandwidth, CPU Utilization ratio, Memory Utilization Ratio, Time Quantum etc. can be considered for balancing the incoming load in the Cloud.[1] Here, the available memory of the participant nodes is taken into consideration. The memory which is available during Cloud setup is a static parameter, which becomes dynamic in nature as and when there is workload incoming in the Cloud and the jobs get processed. Here, the idea is to allocate the jobs to the nodes based on the storage capacity of each of the nodes. The examples of the jobs being given to the Cloud are the files being uploaded. As the files are uploaded, the load goes on increasing and the available memory which is left goes on decreasing. To denote the intake capacity of the nodes, they are differentiated based on their status, that is, idle, normal and overloaded. A Threshold value is maintained for each of the individual nodes which serves as the most important parameter for the Job Dispatch Strategy. The Status is refreshed periodically and the best candidate node is chosen for the incoming file, based on the Left-Out Capacity of the node.

III. Implementation

Consider there is a huge Public Cloud. This public cloud consists of numerous nodes with different capacities spread haphazardly in different geographical locations. The nodes are nothing but individual physical systems with differences in OS, processing speed, memory, cores etc. The load which is incoming in the cloud is of different nature and this workload arrival pattern cannot be predicted. The types of jobs could be simple file storage, which solely depends on memory and others could be compute-intensive tasks where some processing and computation is involved and the result must be returned back to the user. Hence the different types of jobs incoming in the cloud are:-[6], [7]

i) Memory Intensive (Data Intensive)
ii) Processor Intensive (Computation Intensive)
iii) Combination of both

It is the task of the service provider (owner of the cloud) to consider the varying capacities of all the nodes spread in all directions and distribute the incoming workload evenly based on the capacities of the node, so that a situation does not arise where few of the nodes are under loaded and few others are heavily loaded. Without the load being properly balanced, a situation arises where few of the nodes get overloaded and still the jobs (incoming workload) are being given to them, and the result is that, the system crashes. This is because; periodic checks are not done to evaluate their current capacity. Load Balancing, which means even distribution of workload among all the distributed nodes present in the cloud, is a main and crucial task of the service provider. It must be ensured that the system works correctly so that-

- Performance is increased
- User satisfaction is improved
- Delay is reduced by obtaining faster and better response times during task execution
- Waiting time in the queue is reduced
- Most important-priority tasks are given importance and they are not made to wait[2]
- Optimal utilization of available resources
- One system is not under loaded or heavily loaded

Thus it can be concluded that Load Balancing is purely a server side programming concept.

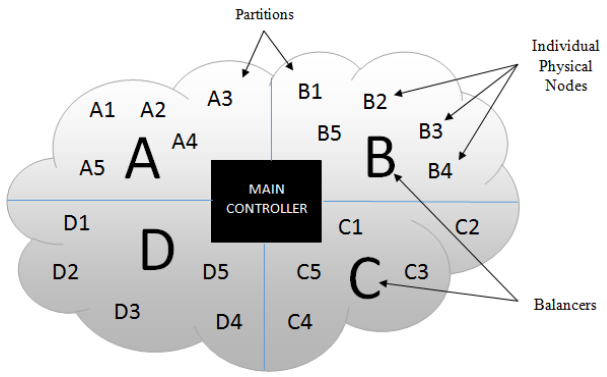

Fig. 1 An example of a Public Cloud with Partitions containing Balancers and Individual Physical Nodes

In the Public Cloud, as depicted in Fig. 1, consider there are 4 partitions based on geographical locations. There is a centrally located Main Controller, which is the heart of the system. Jobs, which the users submit onto the cloud, arrive at the Main Controller.[1] Now, the Main Controller is ready with a queue of jobs and it must check and decide, to which partition, the jobs must be given.[2] These partitions are named as Partition A, Partition B, Partition C and Partition D respectively. There is a separate Balancer maintained for each partition. This Balancer communicates with all the nodes (physical systems) present in the cloud partition. For example, in Partition A, there is a Balancer called Balancer A which communicates at regular intervals with all the nodes present A1, A2, A3 and so on.

A. Partition Selection

Workload acceptance state for a particular partition can be described as, the balancer being in the ON state. When a particular Balancer is in the ON state, the rest of the Balancers are in the OFF state. Here, there are 4 Balancers. Each of the Balancer will be in the ON state for a particular duration of time, which is called time quantum, for example 50 seconds. Initially, starting with Balancer A, when it is ready to accept incoming jobs, it is in the ON state, the rest of the Balancers are OFF, and the jobs from the Main Controller will be given to Balancer A for a time duration of 50 seconds. As soon as 50 seconds are over, Balancer A will be turned OFF and Balancer B will be in the ON state for the next 50 seconds. This continues in a Round Robin fashion for all the Balancers. At a particular instant of time, say t, when a job arrives at the Main Controller, it checks, which Balancer is in the ON state, and gives the job to that particular Balancer. This is how a particular partition or a Balancer is selected. Here, the controller checks for a condition, from which partition or Balancer, it is getting the best response. It gets best response from the balancer which is ready to accept the incoming job that is, whichever Balancer is in the ON state. This is called Best Reply method and it is a functionality of Game Theory.

B. Node Selection

Consider a situation where Partition A is selected. This partition consists of a Balancer A and say there are 6 nodes namely A1, A2, A3 and so on till A6. The Balancer maintains a Load Status table, where information about all the nodes is contained and it can be referred from time to time to allocate the jobs to respective nodes.

TABLE I LOAD STATUS TABLE

Node	**Total Capacity (in KB)**	**Threshold (in KB)**	**No. of Files processed (uploaded)**	**Total size occupied (in KB)**	**Left-out Capacity (in KB)**	**Status**
A1	13000	12000	25	5000	7000	Normal
A2	4000	2000	50	1000	1000	Normal
A3	27000	25000	60	25000	0	Overloaded
A4	20000	19000	5	1000	18000	Normal
A5	1000	500	2	1000	0	Overloaded
A6	5000	4000	0	0	4000	Idle

Here, the Total Capacity of each of the nodes is known by the administrator or the Cloud Provider, because it is a static parameter and has to be determined while setting up the Cloud.

The Threshold value has to be determined by the administrator itself, based on the capacity of each of the nodes. Since it is a heterogeneous environment and since the capacities of each node in the Cloud differ,[1] a common threshold value cannot be taken into consideration, and different threshold values for each of the nodes are determined in the beginning while setting up the Cloud.

Node Selection is the most important phase of Load balancing where the individual physical system has to be pointed out, to which the incoming jobs will be given. It consists of three steps:-

- Calculation of Left-out Capacity
- Determining Status of Nodes
- Job Dispatch Strategy

1) Calculation of Left-out Capacity:

Algorithm 1:-

```
while (nodes)
{
    If (Total Size Occupied >= Threshold)
        update Left-out Capacity = 0
    else
        update Left out Capacity = Threshold – Total Size occupied
}
```

2) Determining Status of Nodes:

Algorithm 2:-

```
while (nodes)
{
    Check for the Left-out Capacity
    if (Left-out capacity = = 0)
        update the status to "Overloaded"
    while (status != "Overloaded")
    {
        if  (Left-out capacity = = Threshold)
            update the status to "Idle", arrange the nodes in the decreasing order of left-out capacity and
            assign id from 01 onwards (01, 02, 03, 04 and so on)
        else
            update the status to "Normal" and arrange the nodes in the decreasing order of left-out
            capacity and assign id from 1 onwards (1, 2, 3, 4 and so on)
    }
}
```

TABLE II LOAD STATUS TABLE WITH ID

Id	Node	Left-out Capacity (in KB)	Threshold (in Kb)	Status
01	A6	4000	4000	Idle
1	A4	18000	19000	Normal
2	A1	7000	12000	Normal
3	A2	1000	2000	Normal

The most important advantage here is that, the overloaded nodes are automatically eliminated and a situation does not arise where more number of jobs or greater load will be given to the overloaded nodes, since here we are considering only those nodes for load balancing which are idle or normal status nodes. As and when more load is given to the nodes, the available memory decreases that is, the left-out capacity decreases and once the threshold value is reached, that node is said to be overloaded. Here, a noteworthy point is that, overloaded nodes are not assigned any id. So, the number of comparisons to be made is thus reduced and hence the complexity of the algorithm is also reduced. The result is that, time is saved and faster results are obtained. Thus, workload is evenly distributed among the nodes, to give better performance and comparatively lesser time for execution.

3) Job Dispatch Strategy:

There are jobs waiting at the Balancer A. The Job Dispatch strategy for Balancer A is described here.

TABLE III QUEUE OF JOBS AT BALANCER A

Jobs	Total Size (in KB)
J1	600
J2	1000
J3	2000
J4	500
J5	50000
J6	8000
J7	7000
J8	10000
J9	4000
J10	2500

Algorithm 3:-

```
Step 1:     Comparison between the Total Size of the incoming job and the Left-out capacity of the nodes in the
            respective partition
            Start the comparison of the Total Size of the first job, from the beginning of Table III, with the lowest
            id node considered first in Table II
            while (jobs in the Queue at the Balancer)
            {
                If (Total Size <= Left-out Capacity)
```

```
                    assign the Job to that particular node
          }
          Return the job back to controller,                 // in case if Total Size of the incoming job > Left-
          which in turn searches for another empty partition  // out Capacity of all the nodes in the partition
Step 2:   If the job is assigned to a particular node, execute the Job
Step 3:   Update the Load Status Table with id
          Left-out capacity = Left-out capacity – Total Size
Step 4:   Apply Algorithm 2
```

After the job is assigned to a particular node, it gets executed and the Balancer updates the Load Status table with id information. Here, Job J1 arrives at the Balancer A. Starting with id=01, the comparison starts. Here the Total Size required by Job J1 is 600 KB. Here Total Size < Left-out capacity (600 < 4000) so the job J1 is given to node A6. Job J1 gets executed on Node A6, and then the information in the load status table with id is updated. The left-out capacity becomes 3400. Algorithm 2 is applied and the Load Status table with id is updated as follows:-

TABLE IV LOAD STATUS TABLE WITH ID AFTER THE 1ST ITERATION

Id	Node	Left-out Capacity (in KB)	Threshold (in Kb)	Status
1	A4	18000	19000	Normal
2	A1	7000	12000	Normal
3	A6	3400	4000	Normal
4	A2	1000	2000	Normal

Now, for Job J2, its total size is 1000 KB, Algorithm 3 is applied, comparisons are done, starting from the lowest id of the node, and the job is assigned to a particular node. After the job is executed, the left-out capacity is calculated and the nodes are again arranged in the decreasing order of Left-out capacities and ids are assigned. This process goes on, for as many jobs the Balancer has received during a particular time quantum.

This process continues until a refresh period, till a particular time quantum. Say for example the time quantum is 50 seconds. Till 50 seconds, the Balancer A continues to accept the job from the Main Controller, processes the jobs and gives back the desired results back to the main Controller. Here, it might so happen that during the 25th second itself, the load becomes too heavy to handle, for the nodes in a particular partition. The solution to this is described in the algorithm itself. After all the comparisons are made, if the incoming job does not fit anywhere, that means if the Total Size of the file is greater than the Left-out capacity of all the nodes, return the job back to the Main Controller and there is no need to repeat the comparisons till the time quantum gets over if the same situation occurs for all the pending jobs. It might so happen that this critical situation arises only for one large job (say job J5 here of Total Size 50000 KB) then after Job J5 is given back successfully to the Main Controller, the comparison again continues for the remaining Jobs in the queue until the time quantum of 50 seconds.

TABLE V LOAD STATUS TABLE WITH ID AFTER THE LAST ITERATION WHEN THE TIME QUANTUM OF 50 SECONDS IS OVER

Id	Node	Left-out Capacity (in KB)	Threshold (in Kb)	Status
-	A1	0	12000	Overloaded
1	A4	2500	19000	Normal
2	A2	1000	2000	Normal
3	A6	900	4000	Normal

In the Load status table with id, the left-out capacity and status, after 10 jobs are assigned to respective nodes during the time quantum of 50 seconds is shown in Table V.

IV. CONCLUSIONS

The research work focuses on scheduling the incoming job to a node which has highest Left-out capacity. All the regions of the Cloud i.e. all the Cloud partitions come into picture periodically and a time quantum is maintained to schedule the jobs to a particular partition. As the job is given to a particular node, after it is executed, the node status is refreshed and a newer order is obtained with the highest Left-out capacity node appearing first. An attempt is made to utilize the resources available at all the nodes efficiently, so that over utilization or under utilization does not become a concern. A conventional Round Robin technique of allocating jobs to next nodes in the queue is evaded here and this technique proves to be much better, since the next nodes in the queue where Round Robin was being deployed, might be overloaded ones, or there might arise a situation where, even though the nodes are not overloaded, they are not capable enough to handle or execute a larger sized incoming job and a lot more comparisons are needed which ultimately increases the complexity of the algorithm. So, the proposed technique automatically arranges the nodes in the descending order of Left-out Capacity, and if the first comparison itself fails, then the job is given back to the Main Controller, thus obtaining faster results.

REFERENCES

[1] Gaochao Xu, Junjie Pang, and Xiaodong Fu, *A Load Balancing Model Based on Cloud Partitioning for the Public Cloud*, IEEE Transactions on Cloud Computing Year 2013, Tsinghua Science & Technology, 02/2013, 18(1):34-39. DOI: 10.1109/TST.2013.6449405, ISSN 1007-0214 04/12 pp34-39, Volume 18, Number 1, February 2013.

[2] Latha C A, *Job Scheduling in the Grid Computing Using Criteria*, Prof., Dept. of CS & E, D B I T Bangalore, India: International Journal of Computer Applications, Volume 90- No 6, March 2014.

[3] http://en.wikipedia.org/wiki/Cloud_computing, accessed on April 9, 2015.

[4] http://searchnetworking.techtarget.com/definition/load-balancing, accessed on April 7, 2015.

[5] https://coderwall.com/p/8czknq/memory-or-cpu-intensive-which-do-you-make, accessed on April 8, 2015.

[6] https://coderwall.com/p/-qtvcq/identifying-memory-and-cpu-intensive-code, accessed on April 8, 2015.

[7] http://www.webopedia.com/TERM/C/cloud_computing.html, accessed on April 8, 2015.

[8] http://www.rackspace.com/cloud/what_is_cloud_computing, accessed on April 9, 2015.

[9] http://searchnetworking.techtarget.com/definition/load-balancing, accessed on April 23, 2015.

[10] http://www.bitpipe.com/tlist/Load-Balancing.html, accessed on April 23, 2015.

[11] http://www.slideshare.net/IBMAsean/smarter-planet-dynamic-infrastructure, accessed on April 23, 2015

[12] http://searchnetworking.techtarget.com/definition/load-balancing, accessed on April 23, 2015

[13] http://searchwindowsserver.techtarget.com/news/761804/Know-IT-All-Question, accessed on April 24, 2015