AN ADAPTIVE ALGORITHM FOR DYNAMIC PRIORITY BASED VIRTUAL MACHINE SCHEDULING IN CLOUD

Subramanian S[1], Nitish Krishna G[2], Kiran Kumar M[3], Sreesh P[4] and G R Karpagam[5]

[1] Department of Computer Science and Engineering,
P.S.G College of Technology,
Coimbatore, Tamil Nadu , 641004, India

[2] Department of Computer Science and Engineering,
P.S.G College of Technology,
Coimbatore, Tamil Nadu , 641004, India

[3] Department of Computer Science and Engineering,
P.S.G College of Technology,
Coimbatore, Tamil Nadu , 641004, India

[4]Department of Computer Science and Engineering,
P.S.G College of Technology,
Coimbatore, Tamil Nadu , 641004, India

[5]Department of Computer Science and Engineering,
P.S.G College of Technology,
Coimbatore, Tamil Nadu , 641004, India

Abstract

Cloud computing, a relatively new technology, has been gaining immense popularity over the last few years. The number of cloud users has been growing exponentially and apparently scheduling of virtual machines in the cloud becomes an important issue to analyze. This paper throws light on the various scheduling algorithms present for scheduling virtual machines and also proposes a new algorithm that combines the advantages of all the existing algorithms and overcomes their disadvantages.

Keywords: *Scheduling, Eucalyptus, Dynamic priority, Cloud.*

1. Introduction

Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The name comes from the use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software and computation [1]. The scheduling of virtual machines in a cloud computing environment has become crucial due to the increase in the number of users. This is usually done to load balance a system effectively or to achieve a target quality of service [2]. The scheduling algorithm used contributes to the performance of the entire system and the throughput. Eucalyptus is **open source** software for building AWS-compatible private and hybrid clouds [3]. There are several scheduling policies already available in Eucalyptus. On analysis of these algorithms, each is found to have its own drawback and hence a new algorithm which overcomes these disadvantages is preferable. The newly proposed algorithm is explained and its working under various circumstances has been illustrated. Also, it is compared with the other existing algorithms.

2. EXISTING SCHEDULING ALGORITHMS IN EUCALYPTUS

Scheduling in Eucalyptus determines the method by which Virtual Machines are allocated to the nodes. This is done to balance the load on all the nodes effectively and to achieve a target quality of service. The need for a good scheduling algorithm arises from the requirement for it to perform multitasking and multiplexing.

The scheduling algorithm in Eucalyptus is concerned mainly with:

Throughput - number of VMs that are successfully allocated per time unit.

Response time - amount of time it takes from when a request was submitted until the first response is produced.

Fairness / Waiting Time – All the requests for an allocation of a node should be treated in the same manner without any bias.

In practice, these goals often conflict and thus a scheduler will implement a suitable compromise. Preference is given to any one of the above mentioned concerns depending upon the user's needs and objectives [4] [5]. A scheduling policy can be chosen by changing the value of SCHEDPOLICY in eucalyptus.conf file [6]. The various existing algorithms along with their limitations are listed below.

2.1 GREEDY ALGORITHM

The Greedy algorithm is the default algorithm used for scheduling of Virtual Machines in Eucalyptus. The Greedy algorithm is very simple and straight forward. As a matter of fact, it was the only scheduling policy which was in use for a long time. Only after the cloud started evolving, more complex scheduling policies came into effect.
The greedy algorithm uses the first node that it finds with suitable resources for running the VM that is to be allocated. The first node that is identified is allocated the VM. This means that the greedy algorithm exhausts a node before it goes on to the next node.
As an example, if there are 3 nodes and the first node's usage is 40% while the other two are under loaded and if there are two VMs to be allocated, then both are allocated to the first node which might result in the increase of its usage to 90% while the other two nodes will still remain under loaded.
As obviously seen, the main advantage of the Greedy algorithm is its simplicity. It is both simple to implement and also the allocation of VMs do not require any complex processing. The major drawback would be the low utilization of the available resources. As illustrated in the example above, even if there are under loaded nodes, an overloading of a node might result.

2.2 ROUND ROBIN ALGORITHM

The Round Robin algorithm mainly focuses on distributing the load equally to all the nodes. Using this algorithm, the scheduler allocates one VM to a node in a cyclic manner. The round robin scheduling in the cloud is very similar to the round robin scheduling used in the process scheduling.
The scheduler starts with a node and moves on to the next node, after a VM is assigned to that node. This is repeated until all the nodes have been allocated at least one VM and then the scheduler returns to the first node again. Hence, in this case, the scheduler does not wait for the exhaustion of the resources of a node before moving on to the next.
As an example, if there are three nodes and three VMs are to be scheduled, each node would be allocated one VM, provided all the nodes have enough available resources to run the VMs.
The main advantage of this algorithm is that it utilizes all the resources in a balanced order. An equal number of VMs are allocated to all the nodes which ensure fairness. However, the major drawback of using this algorithm is that the power consumption will be high as many nodes will be kept turned on for a long time. If three resources can be run on a single node, all the three nodes will be turned on when Round Robin is used which will consume a significant amount of power.

2.3 POWER SAVE ALGORITHM

The Power Save algorithm optimizes the power consumption by turning off the nodes which are not currently used. Instead of keeping all the nodes turned on, resulting in a lot of power consumption, this algorithm aims at turning the unused nodes off which will reduce the power consumed to a reasonable extent.
The scheduler allocates a VM to the node and then traverses through the list of nodes to check if the node is unused and if found to be so, turns it off. If the resource of a node which has been turned off is required for the allocation of a VM, the scheduler turns it on again and then allocates the VM to that node.
As an example, consider the scenario in which there are three nodes, two of which are unused. When a new VM is to be scheduled, the scheduler may allocate it to the node which is already being used and would turn off the two nodes which are unused.
The Power Save algorithm results in the reduction of power consumption but this is at the expense of lower utilization of resources. This algorithm is used only at places where there is an extreme need for reducing the consumption of power.
A new scheduling algorithm has been proposed in this paper which is based on the assignment of dynamic priority to the nodes. It overcomes the stated limitations of the existing scheduling algorithms and works effectively under all circumstances.

3. PROPOSED DYNAMIC PRIORITY BASED SCHEDULING ALGORITHM

The need for a new algorithm can be easily realized from the fact that all the existing algorithms suffer from serious drawbacks. The proposed algorithm uses dynamic priority for the nodes based on which the virtual machines are

scheduled. It schedules the VMs to the nodes depending upon their priority value, which varies dynamically based on their load factor. This dynamic priority concept leads to better utilization of the resources. Priority of a node is assigned depending upon its capacity and the load factor. This algorithm strikes the right balance between performance and power efficiency.

3.1 PSEUDO CODE

Algorithm triggered when a request for a new instance arrives.
Input: None
Output: None
Algorithm sched_priority
{
Flag=0;
If(P1 is not set)
P1=max available resource node
If(P1 is turned OFF)
Turn P1 ON
If(load factor of P1<0.8)
Assign VM to P1;
Flag=1;
if(P2 is set AND load factor of P2<0.8 AND Flag=0)
Swap P1 and P2;
Assign VM to P1;
Else if(Flag=0)
P2=P1
P1=current max available resource node

If(P1 is turned OFF)
Turn P1 ON
Assign VM to P1;
Turn OFF all unused nodes;
}

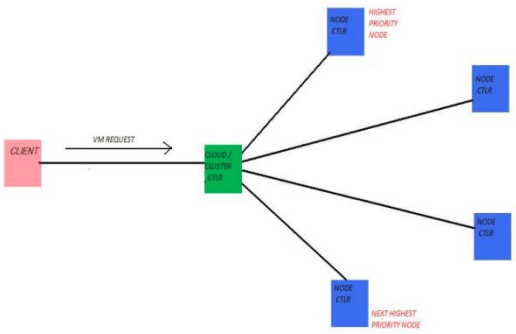

Fig 3.1 Priority based dynamic scheduling algorithm

As and when the virtual machines are assigned to the nodes, recalculation of their priorities takes place. Table 3.1 shows the statistics of the available resources along with priorities when 4 nodes were used. Table 3.2 explains how these priorities got changed when a request for two large instances having high memory requirements arrived.

Node	Priority	Resource available
1	1	24%
2	3	100%
3	4	100%
4	2	100%

Table 3.1 Initial priority value

Node	Priority	Resource available
1	2	10%
2	3	100%
3	4	100%
4	1	100%

Table 3.2 Priority values after arrival of a new instance

3.2 TRACING ALGORITHM WITH VARIOUS SCENARIOS

When a request to schedule a Virtual Machine is received, it checks if the maximum resource node has been identified and its priority has been assigned. If the node has not been identified, then it is identified first. The node with the maximum resource is determined and then it is checked whether the node has a load factor less than 80%. This is done to prevent a particular node from being overloaded. Once the suitable node is identified, then it is assigned the highest priority and the VM is allocated to it as shown in Fig 3.2.

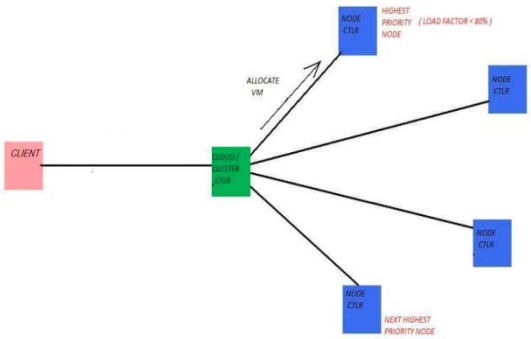

Fig 3.2 VM allocated to the highest priority node

If the highest priority has a load factor above 80%, then it checks if the next maximum resource node has been identified. If it has been identified and if its load factor is less than 80%, then the VM is scheduled to that node and the search for the next maximum resource node with the load factor less than 80% takes place. Before assigning the

new node as the highest priority node, the current maximum resource node is assigned to previous maximum resource node and the previous maximum resource node is assigned the next highest priority.

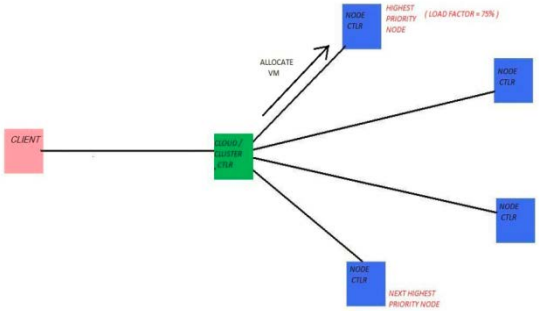

Fig 3.3 Highest priority node has a load factor of 75%

The purpose of keeping track of the previous maximum resource node is to prevent the fluctuations above and below the load factor in extreme cases, when VMs are assigned to it and removed from it. Consider the scenario in which, the maximum resource node has been identified and it has been assigned the highest priority. The VMs are allocated to it till the load factor is less than 80%. Assume that the current load factor is 75% as shown in Fig 3.3. After allocating a VM to it, the load factor becomes 85% as shown in Fig 3.4. On completion of the work, the VM is shut down and the load factor goes to 75%. The load factor fluctuates above and below 80% frequently and will initiate a new search request for finding the new node. To overcome this problem, the previous maximum resource node is kept track off i.e. the next highest priority node.

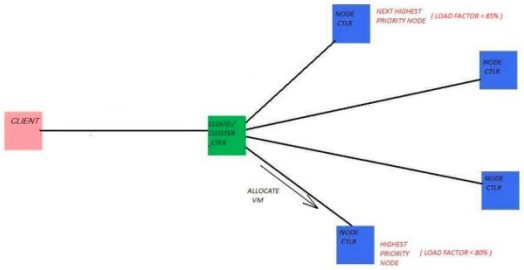

Fig 3.4 Highest priority node is changed

If the highest priority node has the load factor more than 80%, then it keeps allocating the VMs to the next highest priority node until its load factor becomes greater than 80%. This gives the highest priority node some time to finish off the work of VM and shut it down so that it would be well below the load factor. If the load factor is still above 80%, then its priority is decreased and the other node is assigned higher priority.

This algorithm shows remarkable speed in terms of the time taken to schedule a Virtual Machine to a node, once these nodes have been identified. The scheduling would be done to these nodes directly and the other nodes are not taken into consideration.

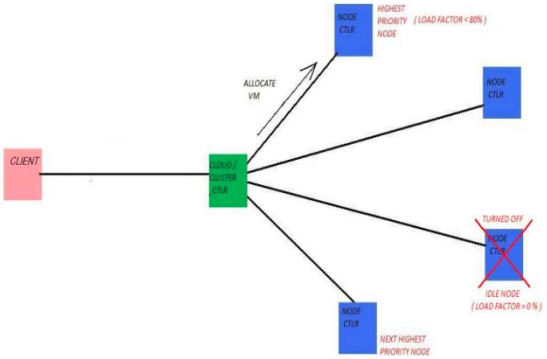

Fig 3.5 Idle node is shut down

When it comes to power efficiency, every time the request for scheduling is received, it also checks for the idle nodes. The idle nodes are one to which, no VM is allocated to it. These nodes are turned off to save power as shown in Fig 3.5.

3.3 HIGHLIGHTS OF THE ALGORITHM

1. Once the highest and next highest priority nodes have been identified, then the scheduling is very quick.
2. It prevents a particular node from being overloaded by considering the load factor.
3. The idle nodes are turned off. Hence it is power efficient.
4. It prevents fluctuations around the load factor of 80% in most cases. Fluctuation occurs only under extreme cases, when all the nodes have load factor which are approximately 80%.

4. COMPARISON OF ALGORITHMS

On comparison with the other algorithms, the proposed dynamic priority based scheduling algorithm has been found to work more effectively in most of the cases.

4.1 COMPARISON WITH GREEDY ALGORITHM

The proposed algorithm does not exhaust a particular node like Greedy algorithm. Therefore the proposed algorithm makes better utilization of the available resources.

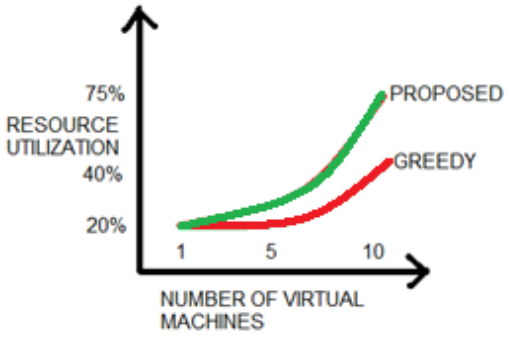

Fig 4.1 Comparison with greedy algorithm

As shown in Fig. 4.1, the resource utilization of the proposed algorithm is better than that of the greedy algorithm, given that the number of nodes is fixed.

4.2 COMPARISON WITH ROUND ROBIN ALGORITHM

The proposed algorithm takes a constant time to assign a node once the priorities have been assigned, unlike the Round Robin algorithm and the proposed algorithm also consumes lesser power than the round robin algorithm.

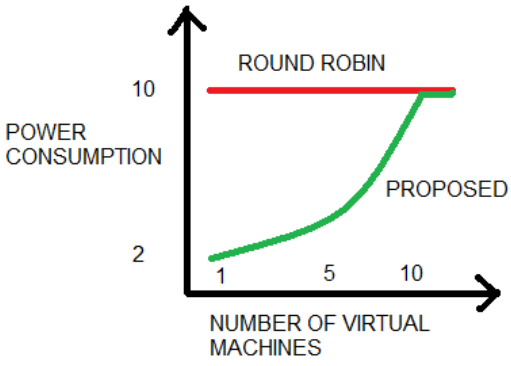

Fig 4.2 Comparison with round robin algorithm

As shown in Fig. 4.2, keeping the number of nodes fixed, the power consumption in the proposed algorithm is lesser than that of round robin algorithm until the number of virtual machines increases such that all the nodes are turned on. From that point onwards, both the algorithm consume the same power until any one of the nodes gets turned off according to the proposed algorithm.

4.3 COMPARISON WITH POWER SAVE ALGORITHM

The proposed algorithm does not turn off the nodes very frequently unlike the power save algorithm and hence faster response times are seen than that of power save algorithm.

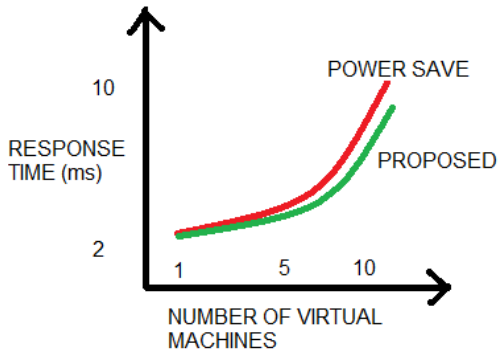

Fig 4.3 Comparison with power save algorithm

As shown in Fig.4.3, even though the difference is not very vast, the response time of the proposed algorithm is lesser than that of power save algorithm as the number of virtual machines increase, keeping the number of nodes fixed.

5. CONCLUSION

Thus a new algorithm for the scheduling of virtual machines in Eucalyptus platform was proposed along with the explanation of how the algorithm would work under various circumstances. A comparative study was done comparing the various existing algorithms with the proposed algorithm and the results were illustrated in the form of graphs. However, the proposed algorithm does not handle certain cases like the failure of nodes. Also, the uptime and downtime of nodes have not been measured.

This scheduling algorithm can be extended to suit other cloud platforms also. Here, the load factor above which the highest priority node changes is kept constant. Further extension to this algorithm can be done by varying the maximum load factor, above which the priority of a node decreases, dynamically by setting it to an optimum value based on the present conditions.

Acknowledgments

We would like to thank Cordys for introducing the concept of cloud computing to us and inspiring us to take up issues in cloud computing as our research project.

References

[1] Cloud Computing:
http://en.wikipedia.org/wiki/Cloud_computing
[2] Scheduling:
http://en.wikipedia.org/wiki/Scheduling_(computing)

[3] Eucalyptus:
http://en.wikipedia.org/wiki/Eucalyptus_(computing)
[4] Brief discussion of Job Scheduling algorithms-
http://www.cs.sunysb.edu/~algorith/files/scheduling.shtml
[5] Blazewicz, J., Ecker, K.H., Pesch, E., Schmidt, G. und J. Weglarz, Scheduling Computer and Manufacturing Processes, Berlin (Springer) 2001
[6] Eucalyptus Scheduling Algorithms:
http://open.eucalyptus.com/

Subramanian S is currently the B.E. degree candidate In Department of Computer Science, P.S.G College of Technology, Coimbatore, India. His research interests include Cloud computing and its applications.

Nitish Krishna G is currently the B.E. degree candidate In Department of Computer Science, P.S.G College of Technology, Coimbatore, India. His research interests include Cloud computing and its applications.

Kiran Kumar M is currently the B.E. degree candidate In Department of Computer Science, P.S.G College of Technology, Coimbatore, India. His research interests include Cloud computing and its applications.

Sreesh P is currently the B.E. degree candidate In Department of Computer Science, P.S.G College of Technology, Coimbatore, India. His research interests include Cloud computing and its applications.

G R Karpagam is a professor at the Department of Computer Science, P.S.G College of Technology, Coimbatore, India. Her specialization areas include database, service oriented architecture and cloud computing.