ISSN (e): 2250 – 3005 || Volume, 05 || Issue, 10 ||October – 2015 ||
International Journal of Computational Engineering Research (IJCER)

Efficient Resource Allocation to Virtual Machine in Cloud Computing Using an Advance Algorithm

Rajeev Kumar [1], Aditya Sharma[2]
[1] Deptt. of C.S.E.,Arni University, Kangra,India
[2] Deptt. of C.S.E, Arni University, Kangra, India

ABSTRACT:
The focus of the paper is to generate an advance algorithm of resource allocation and load balancing that can deduced and avoid the dead lock while allocating the processes to virtual machine. In VM while processes are allocate they executes in queue , the first process get resources , other remains in waiting state .As rest of VM remains idle . To utilize the resources, we have analyze the algorithm with the help of First-Come, First-Served (FCFS) Scheduling, Shortest-Job-First (SJR) Scheduling, Priority Scheduling, Round Robin (RR) and CloudSIM Simulator.

KEYWORDS: *VM(Virtual machine)*

I. INTRODUCTION

Cloud computing has attracted attention as an important platform for software deployment, with perceived benefits such as elasticity to fluctuating load, and reduced operational costs compared to running in enterprise data centers. While some software is written from scratch especially for the cloud, many organizations also wish to migrate existing applications to a cloud platform. A cloud environment is one of the most shareable environments where multiple clients are connected to the common environment to access the services and the products. A cloud environment can be public or the private cloud. In such environment, all the resources are available on an integrated environment where multiple users can perform the request at same time. In such case , some approach is required to perform the effective scheduling and the resource allocation.

II. RESOURCE ALLOCATION

There are different algorithm that defines the load balancing to provide resources on the criteria as following

2.1 Token Routing: The main objective of the algorithm is to minimize the system cost by moving the tokens around the system. But in a scalable cloud system agents cannot have the enough information of distributing the work load due to communication bottleneck. So the workload distribution among the agents is not fixed. The drawback of the token routing algorithm can be removed with the help of heuristic approach of token based load balancing. This algorithm provides the fast and efficient routing decision. In this algorithm agent does not need to have an idea of the complete knowledge of their global state and neighbor's working load. To make their decision where to pass the token they actually build their own knowledge base. This knowledge base is actually derived from the previously received tokens. So in this approach no communication overhead is generated.

2.2 Round Robin: In this algorithm, the processes are divided between all processors. Each process is assigned to the processor in a round robin order. The process allocation order is maintained locally independent of the allocations from remote processors. Though the work load distributions between processors are equal but the job processing times for different processes are not same. So at any point of time some nodes may be heavily loaded and others remain idle. This algorithm is mostly used in web servers where Http requests are of similar nature and distributed equally.

2.3 Randomized: Randomized algorithm is of type static in nature. In this algorithm process can be handled by a particular node n with a probability p. The process allocation order is maintained for each processor independent of allocation from remote processor.

This algorithm works well in case of processes are of equal loaded. [10]. However, problem arises when loads are of different computational complexities. Randomized algorithm does not maintain deterministic approach. It works well when Round Robin algorithms generate solver head for process queue.

2.4 Central queuing: This algorithm works on the principal of dynamic distribution. Each new activity arriving at the queue manager is inserted into the queue. When request for an activity is received by the queue manager it removes the first activity from the queue and sends it to the requester. If no ready activity is present in the queue the request is buffered, until a new activity is available. But in case new activity comes to the queue while there are unanswered requests in the queue the first such request is removed from the queue and new activity is assigned to it. When a processor load falls under the threshold then the local load manager sends a request for the new activity to the central load manager.

2.6 Connection mechanism: Load balancing algorithm can also be based on least connection mechanism which is a part of dynamic scheduling algorithm. It needs to count the number of connections for each server dynamically to estimate the load. The load balancer records the connection number of each server. The number of connection increases when a new connection is dispatched to it, and decreases the number when connection finishes or timeout happens.

III. ALGORITHM

The algorithm provide parallel processes to each virtual machine rather than serial processes one by one.

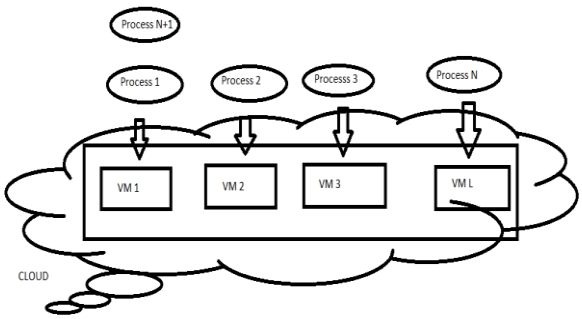

Figure 1. VM Function

*a. *Input the M number of Clouds with L1, L2, L3……., Ln (n tends to no. of last virtual machine))number of Virtual Machines associated with each cloud.
*b. *Define the available memory and load for each virtual machine.
*c. *Assign the priority to each cloud.
*d. *Input n number of user process request with some parameters specifications like arrival time, process time, required memory etc.
*e. *Arrange the process requests in order of memory requirement
*f. *For i=1 to n
*g. *{
*h. *Identify the priority Cloud and Associated VM having Available Memory(L1,L2,L3……Ln)>Required Memory(i)
*i. *Perform the initial allocation of process to that particular VM and the Cloud
*j. *}
*k. *For i=1 to n
*l. *{
*m. *Identify the Free Time slot on priority cloud to perform the allocation. As the free slot identify, record the start time, process time, turnaround time and the deadline of the process.
*n. *}
*o. *fori=1 to n
*p. *(
q. start queue Q1.
*r. *{
*s. Process i1allocate to VM L1.
*t. *Print "Migration Done"
*u. Process i2, i3……in allocate to VM L2, L3……,Ln respectively.
w. Q1, i1, p1 ends till i(n+1) allots to L1 again.
*X. * start new Queue Q2 [i (n+1)],
Q3 {I (2n+1)},……. Respectively.
*y. *}
*z. *}

IV. EXPERIMENTAL REVIEW

Larger waiting time and Response time

In round robin architecture the time the process spends in the ready queue waiting for the processor to get executed is known as waiting time and the time [13] the process completes its

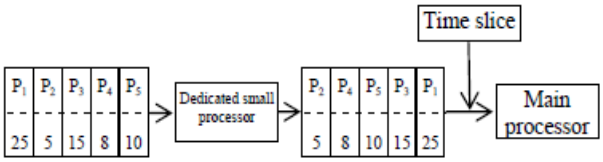

Figure: Process scheduling in shortest round robin

Intelligent time slice generation

A new way of intelligent time slice calculation has been proposed which allocates the frame exclusively for each task based on priority, shortest CPU burst time and context switch avoidance time.

Let the original time slice (OTS) is the time slice to be given to any process if it deserves no special consideration

Intelligent time slice = Original Time Slice(OTS)+ Priority Component (PC)+ Shortness Component for CPU burst time (SC)+ Context Switch Component(CSC)

The intelligent time slice of process P1 is same as the original time slice of four milliseconds and time slice of four milliseconds is assigned to process P1. After the execution of four milliseconds time slice the CPU is allocated to process P2. Since the CPU burst of process P2 is lesser than the assumed CPU burst (ATS), one milliseconds of SC has been included. The process P3 has the highest priority, so priority component is added and the total of five milliseconds is allocated to process P3. The Balanced CPU burst for process P4 is leaser than OTS, context switch component is added and a total of eight millisecond time slice is given to process P4. Process P5 is given a total of five milliseconds with one millisecond of priority component is added to original time slice. After executing a cycle the processor will again be allocated to process P1 for the next cycle and continuously schedules in the same manner.

$$\sum_{i=1}^{n} \frac{wt_i}{n}$$

wt → waiting time of process.

n → No. of process.

$$\sum_{i=1}^{n} \frac{tt_i}{n}$$

Steps for scheduling are as follows

Step 1:

Master system (VMM) receives information regarding virtual machine from slave (VM-1….n). If the master node capability doesn't catch the data, it will determine the virtual machine to be dead. This study proposed by parameter W.

- If W=0 is set up, it will define the virtual machine to be working and still alive now.
- If W=1 then node is dead.
- If W=2 then node is in previous state.

Step 2: If Master node receives the data from slave, then it gets the information's regarding data (memory used, CPU time etc...)

Step 3: Then Master node builds the weighted table containing the details which is collected from step 2.

Step 4: Then the master node sorts (Round-robin method) all the virtual machines according to their performance. Which is 1≦i≦N.Where N is the number of the virtual machines.

Step 5: The scheduling capability generates the weighted table.

Step 6: The virtual machine control capability receives the weighted table from the Step 5, and distributes the task to the virtual machines according to the weighted value.

V. RESULT

The proposed algorithm and existing round robin algorithm implemented like graphical simulation. Java language is used for implementing VM load balancing algorithm. Assuming the application is deployed in one data centre having virtual machines (with 2048Mb of memory in each VM running on physical processors capable of speeds of 100 MIPS) and Parameter Values are as under Table discuss the Parameter value's which are used for Experiment.

Parameter	Value
Data Center OS	Linux
VM Memory	2048mb
Data Center Architecture	X86
Service Broker Policy	Optimize Response Time
VM Bandwidth	1000

Table 1. Parameter values used for Experiment

These experimental results shows that weighted round robin method improves the performance by consuming less time for scheduling virtual machines.

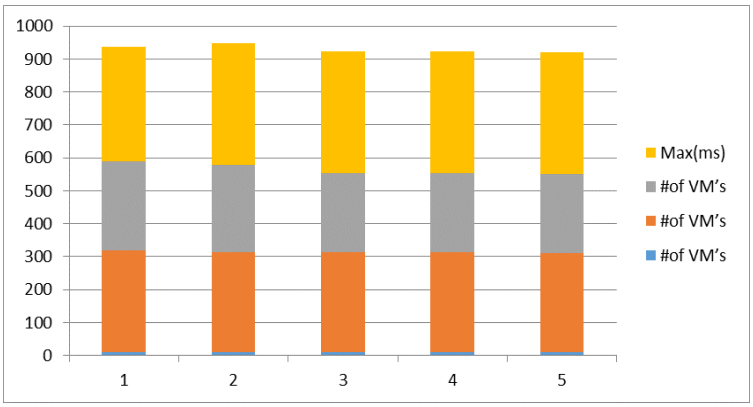

Figure 2. The results based on Round robin algorithm

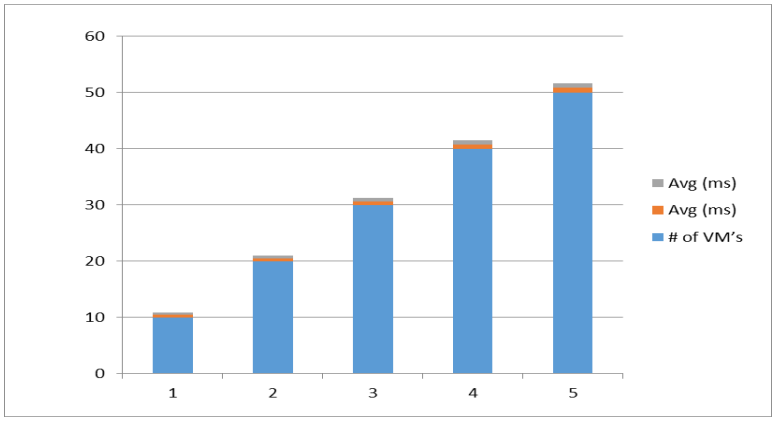

Figure 3. The Data Centre for Round robin algorithm

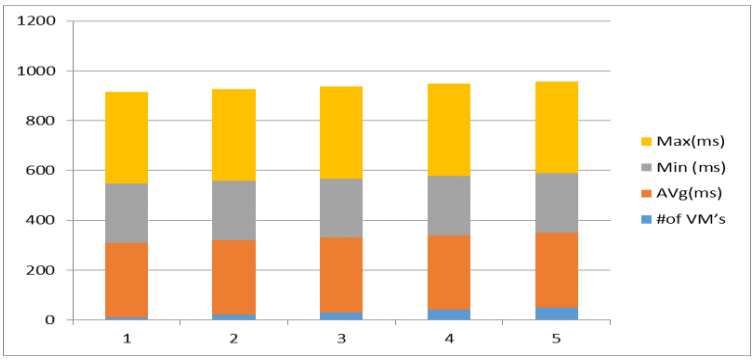

Figure 4. The results based on Weighted Round robin table

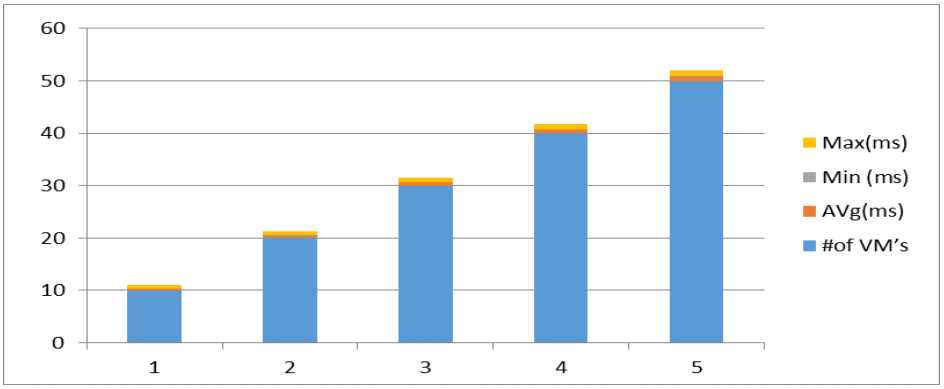

Figure 5. The Data Centre for Weighted Round robin table

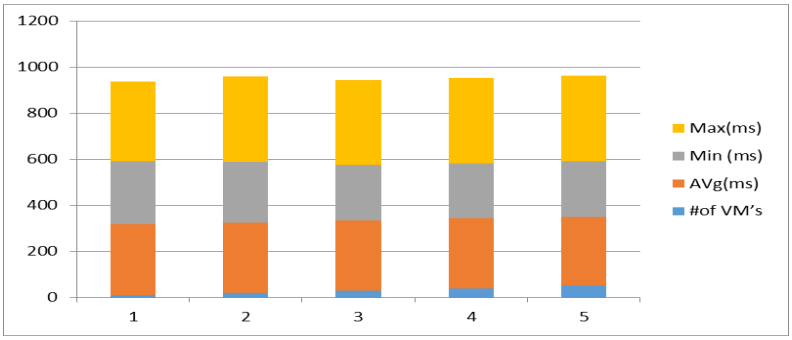

Figure 6. Comparison of results between Round Robin and weighted round robin For Overall response time

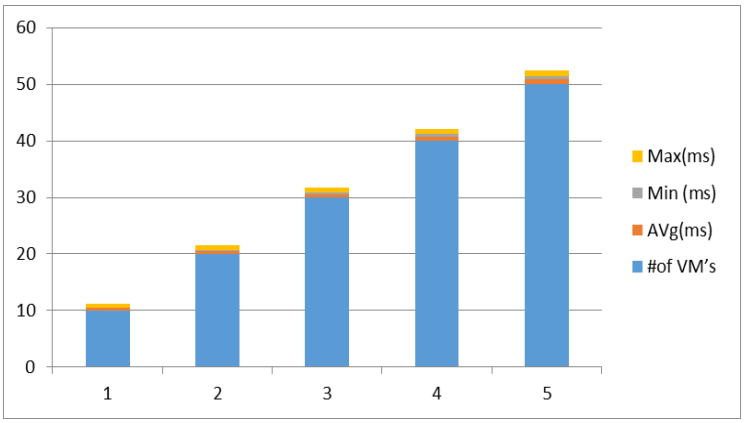

Figure 7. Comparison of results between Round Robin and Weighted Round Robin for Data Center processing time.

VI. CONCLUSION

The above algorithm distributes the process allocation in such a way that process does not concede each other and the waiting state time for process is very much less .as well as all recourses (VMs, memory) are using efficiently. That means the dead lock accruing chances are very much lees. .if the processes may allocate to virtual machine at time. Processes may execute fast and chances of deadlock accruing is less. So we need an algorithm that can describe how process execution can be done on virtual machine fast. A comparative study of round robin architecture shortest round robin and intelligent time slice for round robin architecture is made. It is concluded that the proposed architectures are superior as it has less waiting, response times, usually less preemption and context switching thereby reducing the overhead and saving of memory space. Future work can be based on these architectures modified and implemented for hard real time system where hard deadline systems require partial outputs to prevent catastrophic events.

REFFERENCES:

[1] Anupama Prasanth, "Cloud Computing Services: A Survey*", International Journal of Computer Applications,* Vol. 46, No.3, May 2012, PP.0975 – 8887.

[2] Flavio Lombardi a, RobertoDiPietro, "Secure Virtualization For Cloud Computing", *Secure virtualization for cloud computing,* Journal of Network.

[3] Amazon cloud computing, virtualization and virtual machine, 2002.

[4] Thomas Weigold, Thorsten Kramp and Peter Buhler, *"*ePVM- An Embeddable Process Virtual Machine*"Annual International Computer Software and Applications Conference(COMPSAC 2007),* IEEE.

[5] Rajeev Kumar, Rajiv Ranjan " virtual Machine Scheduling To Avoid Deadlocks", International Journal of Computer Science and Information Technology Research ,Vol.2, Issue 2, pp:(369-372),June,2014.

[6] Robert Blazer, "Process Virtual Machine", *IEEE, 1993,* PP.37-40.

[7] Loris Degioanni, Mario Baldi, Diego Buffa, FulvioRisso, Federico Stirano, GianlucaVarenni, "Network Virtual Machine (NETVM): A New Architecture for Efficient and Portable Packet Processing Applications", 8th *International Conference on Telecommunications,* Zagreb, Croatia, June 15 - 17, 2005,PP.163-168.

[8] DongyaoWu,JunWei,ChushuGao,WenshenDou**, "**A Highly Concurrent Process Virtual Machine Based on Event-driven Process Execution Model"**,** *2012 Ninth IEEE International Conference on e-Business Engineering,*PP.61 – 69.

[9] Vue Hu, YueDongWang, "Process-Level Virtual Machine Embedded Chain", *2011 International Conference on Computer Science and Network Technology,* 1EEE, December 24-26, 2011,PP.302 – 305.

[10] Yosuke Kuno, Kenichi Nii, SaneyasuYamaguchi, "A Study on Performance of Processes in Migrating Virtual Machines", *2011 Tenth International Symposium on Autonomous Decentralized Systems,*IEEE,2011, PP.568-572.

[11] GeetaJangra, PardeepVashist, ArtiDhounchak, " Effective Scheduling in Cloud Computing is a Risk? ",*IJARCSSE,*Volume 3, Issue 8, August 2013,PP.148 – 152.

[12] Ghao G, Liu J, Tang Y, Sun W, Zhang F, Ye X, Tang N (2009) Cloud Computing: A Statistics Aspect of Users. *In:First International Conference on Cloud Computing (CloudCom),* Beijing, China. Heidelberg: Springer Berlin. PP.347-358.

[13] Zhang S, Zhang S, Chen X, Huo X (2010) Cloud Computing Research and Development Trend. *In: Second International Conference on Future Networks (ICFN'10),* Sanya, and Hainan, China. Washington, DC, USA: IEEE Computer Society. PP.93-97

[14] Cloud Security Alliance (2011) Security guidance for critical areas of focus in Cloud ComputingV3.0.Available:https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdfweb cite

[15] MarinosA, Briscoe G (2009) Community Cloud Computing. *In: 1st International Conference on Cloud Computing (CloudCom),* Beijing, China. Heidelberg: Springer-Verlag Berlin.

[16] Khalid A (2010) Cloud Computing: applying issues in Small Business. *International Conference on Signal Acquisition and Processing (ICSAP'10)*PP. 278 – 281.