Indian Journal of Science and Technology, Vol 8(23), DOI: 10.17485/ijst/2015/v8i23/51867, September 2015
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

A Quantitative Study on the Performance of Cloud Data Centre: Dynamic Maintenance Schedules with Effective Resource Provisioning Schema

R. K. Nadesh[1*], Meduri Jagadeesh[1] and M. Aramudhan[2]

[1]SITE, VIT University, Vellore - 632014, Tamil Nadu, India; rknadesh@vit.ac.in, meduri.jagadeesh@msn.com
[2]Department of IT, PKIET, Puducherry - 609603, India; aranagai@yahoo.co.in

Abstract

The demand for cloud based resources has been increasing, catering to the need for enormous growth in Cloud data centers to serve its customers by providing different types of VMs and providing support too many business applications is always challenging. Cloud data center is the collection of physical resources, which provide services to its customers in a virtual manner. There is need for effective scheduling algorithms along with scheduling maintenance activity of the servers. To provide continuous and uninterrupted services to its customers, data centers is to be maintained properly by looking at the issues like scheduling the server's resources to requests, queuing the requests to the servers and proper maintenance time allocated to each server. Devising the algorithm and implementing is done in algorithmic perspective with Open Nebula toolkit and the results are evaluated using the sunstone GUI server. To enable resources in Open Nebula toolkit, ruby scripts are coded with embedded C. Open Nebula CLI commands are used to check the performance of the proposed design.

Keywords: Command Line Interface (CLI), Virtual Machine (VM)

1. Introduction

Data centers are the collection of physical servers which are networked to provide uninterrupted services to its users. To serve users requests from these data centers, VMs are created with features that provide services to cloud users, give him feasibility to add different applications on these VMs as per his requirement. Different VMs are created on the same physical servers which provide resources to VMs to function properly. These resources include requirements like CPU, memory. Data centers consist of these resources in a large scale where a lot of nodes connected to each other with a central controller which controls them. These resources will be allocated to the VM based on its requirements or features required by the user depending on the cloud service provider, where pricing for the utilization of these resources is done either on time based or VM size depends on the service provider's policy.

Resource provisioning is the process of scheduling the resources to VMs from data center. Resource provisioning is done by central controller which provides resources to the requests of the user in the form of VMs. In order to schedule these resources from the date centers scheduling algorithms are designed which should effectively schedule the data centers resources.

The physical resources of the data center are managed with the cloud management tool kits which should provide all the features for improving the functionality of them towards the data centers to add enhanced features. Some of the toolkits are Nimbus, Eucalyptus and Open Nebula. These toolkits are used by the service providers in

*Author for correspondence

order to serve their customer needs and service provider can re-model them as they are open source and utilize them to improve their data centers performance.

Designing solutions to the above problems; a very few research scholars are working by providing different algorithms for maintenance. The solutions consist of fixed maintenance time for each and every server, based on the maintenance time; the tasks of providing resources of a server are migrated to another server.

Earlier algorithms will use a predictive resource provisioning schema with fixed maintenance schedules. Still now many web sites with a fixed maintenance schedules and some dynamic provisioning schemas are designed but they are not co-coordinating with the maintenance activities of the data centers. This shows the need for a dynamic provisioning along with the dynamic maintenance scheduling schema.

Earlier algorithms shift the VMs of a particular server to another by assigning the resources to VM from another server. The shifting should be done such that the VMs applications should not be affected and don't show any degrade in the performance of applications of the VM. Algorithms are mostly fixed time maintenance and have some problems which are discussed in the problem description. There are several performance issues that rise while dealing with these algorithms.

2. Problem Definition

In data centers every server needs time for maintenance activities, during these maintenance activities several performance issues of the server are checked and improved if necessary. The maintenance activities consist of physical maintenance like connections checking and other required activities. Software updates are also necessary when there is any update done by the Cloud data centre maintainer. Approach to maintenance activity raises questions like when to maintain the data centre server and how much time required for maintaining the server. Maintenance not only deals with the setting of the scheduling time for maintenance. It also has the issues that will rise while working with the datacenters like failure of hosts, allowing the user requests dynamically at any time without dropping the datacenters performance, along with maintenance and scheduling of the resources to the customer requests. The related works are summarized.

Resource allocation using policy based schemas[1], the model uses Haizea, a resource lease manager which allocates resources based on whether the request is reserved, has a deadline, need a best effort and immediate. With these features a dynamic planning scheduling algorithm is used which improved the results. Failure aware resources provisioning algorithm[2] which suits hybrid cloud environment where the failures are inevitable. The model uses scheduling algorithms to provide well brokering strategies (size, time and area used for the provisioning. This model gives the idea for moving the VMs from public and private clouds which needs more research work to avoid failures.

Deadline based algorithm[3], which is designed to provide application dependent resource provisioning. VMs applications deadlines are considered by dynamic resource provisioning model[4]. Their work not only provides provisioning but also scheduling of those requests with the new resources. A grid based simulation environment is created and the results are effective in terms with the First Come First Serve scheduling schema. The performance of this model is degraded with other scheduling mechanisms. Due to requirements of the customer, hybrid clouds are designed which require special concern for provision and maintenance.

A grid based provisioning schema improving the QOS by the data centers using QStorMan toolkit[5], designed model uses 3 layer approach called cluster, resource monitoring and management. This approach improves extensive evaluation and execution time of the application. The problem with predictive resource provisioning in data centers[6] is with Single Exponential Smoothing algorithm as example. To improve the efficiency their proposed model uses Vector projection strategy and designs a dynamic resource scheduling schema in two steps.

At certain levels the bulk data that will harm the provisioning activity. This will overcome the time dependent resource allocation problem and provides solution using a scheduling algorithm. High level Meta scheduler and provides data placement in the form of a service[7].

Next priority is also one of the tasks that should be allocated to the data centers while dealing with the requests from the clients. Till now only a few research scholars deal with this problem[8], gives a node based priority scheduling to the servers.

QOS needed by the users and the measures that should be followed by data centers to improve the QOS for its customers[8]. To provide effective provisioning the

model uses Aneka platform for deployment of scalable resources. The drawback with this approach is it will not consider the time while provisioning of the resource[9].

Error correction neural network and linear regression based prediction schema experiments are contacted with using sliding window and without using it, the results shows that sliding window based neural network prototype is more accurate[10].

The Ant Colony optimization and Multi-Objective Genetic Algorithm are used for the virtual machine placement problem[11]. It is an algorithm that combines both multi-objective algorithm and ant colony optimization.

Data centers in India need to incorporate innovative designs for energy efficiency and embrace the concept of "Green IT" for sustained growth. Existing Datacenters need to adopt the best practices in design, operation and maintenance to achieve operational excellence[12]. An optimization technique called Bacterial Foraging[13] has been used in order to continuously optimize the allocation of resources thereby improving the energy efficiency of the data centre.

From related work, it is understood, the problem of fixed maintenance activity though they have no updates. This will have bad effect on performance considerations. All the models concerned about different ways of provisioning but not on reduction of service activity due to the laps in maintenance scheduling activity. This shows need for an enhanced maintenance time scheduling algorithm.

3. Challenges

The algorithms designed should work dynamically without manual intervention, synchronization between maintenance and provisioning algorithms should be availed. Performance issues should be monitored while doing critical operations like:

- VM failures should be avoided while pre-empting or migrating from host to host or host failures.
- Avoiding resource wastages and VM pending state problem.
- Providing updates to hosts dynamically at any time.

Priority scheduling is another task where server can choose the requests based on its resources. Effective priority algorithm is needed to achieve this task, where earlier priority assigning algorithms have been developed based on shortest server assigning schema which has defects like assigning high rated servers to small tasks which causes problem of showing busy to priority and high loaded requests. There is need for an algorithm which gives priority to the requests based on their loads. Existing algorithms though provide resources to the users requests, many CPU cycles are wasted which were not utilized by the users shows the necessarily of an advanced provisioning algorithm which reduces the wasted cycles and provide services more effectively.

4. Proposed System

The central theme of this paper is to design a model for dynamic, load-balanced resource scheduling as well as maintenance and those co-ordinates with each other. To achieve proposed model algorithms are designed, implemented. Simulation is performed on the proposed design. In order to design the algorithms we have to consider the following conditions:

- Avoid the resource wastage before and after scheduling the resources to the customer's request.
- Proper maintenance time should be given to each and every server in data centre with the help of scheduling schemas.
- If needed priority should be given to data centers in maintenance when they had utilized more compared to others.
- Reduce the losses and delays while provisioning of the VMs in data centers.

To achieve these tasks the algorithms are designed in the following categories:

- Algorithm for scheduling the resources of data centers.
- Algorithm for Scheduling time for maintenance of data centers.

From the different toolkits that used for data center activities Open Nebula is used in this design which provides much flexibility and support to management and scheduling activities of the data centers.

5. Design

The proposed design is categorized in to three algorithms.

- Resource scheduling algorithm.
- Maintenance scheduling algorithm.
- Host Priority scheduling algorithm.

5.1 Algorithm 1: Resource Scheduling Algorithm

This Algorithm is used for allocating the resources for a VM that is created:

Input: Resources are CPU, memory from host, images and files.

Output: CPU cycles, Memory, VCPUs allocated to a VM.

Interface between resources and VMs is created by Template.

Algorithm:

Inputs: Requests RQ, priorities P and resources (RS), Server S

Outputs: Allocated VMs and remaining resources (VM)

while RQ

count = count + 1

End while.

If, count >0

Send S to algorithm 3

Receive S_p from algorithm 3

Get RQ (count).

for VM_i belongs to VM do

for $RQ_{i,p}$ belongs to RQ do

if RQ_i.src = $S_{j,P}$

End if

End for

5.2 Algorithm 2: Maintenance Scheduling Algorithm

In the proposed design we are creating a data server maintenance algorithm which will be scheduled on the data centre servers dynamically checking them periodically. Maintenance activity also will be scheduled when any server is set free from VMs. There is no particular period of time designed for the maintenance i.e. it is completely depends on the number of updates. One more feature of this algorithm is in order to serve the VMs at heavy load times the servers in the maintenance will be sent for serving VMs based on the previous updated check point history.

Algorithm 2:

Input: server's S, maintenance M, Updates (U)

Output: S assigned to M or algorithm 1

While S

If U_l equals to S_u

S_u assigned to algorithm 1

Else

Assign U_l time

S_u belongs to U_l

End else

End if

End while

while U_l

Call S

S_i update with U_l

End while

5.3 Algorithm 3: Host Priority Scheduling Algorithm

At any time when the allocated resources are set to free then they will be checked for maintenance. The resources that completed maintenance activity will come under this section and will be ready to serve the VMs. The inputs in this algorithm are resources from data centers and the outputs are the allocation of resources to the resource scheduling algorithm and presenting the particular data center to maintenance activity. The idea of this algorithm is to give priority to the resources in such a manner that these resources will select the requests depends on the requirements of the requests. The requirements consist of CPU utilization, time duration, and processor. Based on these aspects each server is server is given a name like High loaded server, Medium loaded server and Low loaded server.

Algorithm 3:

Input: Servers (S) with HL, ML and LL, Request (RQ), L1, L2, L3 (limits)

Output: Servers

While (RQ)

Check server type

If rq.cpu< L3

Assign LL

End if

If rq.cpu>L3 and <L2

Assign ML

End if

Else

Assign HL
End else
End While.

The maintenance of data centers is achieved by monitoring the hosts and allocating the available resources in the form of VM's. This is achieved by having an interface between the infrastructure and the users. Data centers need an interface which has the task of scheduling and maintenance of the resources. The data flow diagram of the design is shown in Figure 1 followed by the Open Nebula architecture.

The algorithms are implemented in both driver and scheduler locations shown in Figure 2 i.e. Algorithm 1 at scheduler and Algorithm 2 at drivers. Template CLI will take care of Algorithm 3.

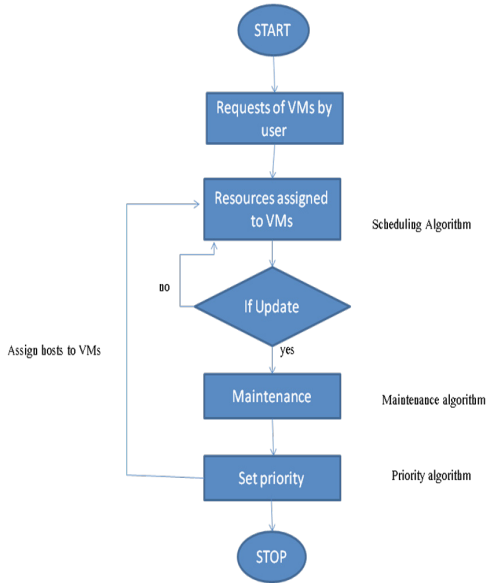

Figure 1. Data flow Diagram.

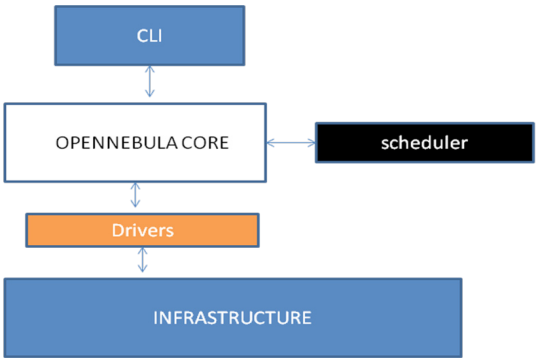

Figure 2. Architectural design.

To implement above modules Open Nebula toolkit is used which support management features of the data centers, as well as provides the following to implement the service provider's designs.

- VMs.
- Templates.
- Images.
- Hosts.
- Clusters.

5.3.1 Virtual Machine

A VM is a virtual resource that is given to the cloud customers with CPU and memory. This resource is used by the customer for his needs.

5.3.2 Template

A template is a supportive and explains how to describe the wanted-to-be-ran Virtual Machine, and how users typically interact with the system.

5.3.3 Image

Administrators and users are to set up images, which can be operative systems or data, to be used in virtual machines easily. These images can be used by several virtual machines simultaneously, and also shared with other users. OpenNebula provides flexibility to users even to add his own images to the VMs. By enabling the images to the hosts the users will use those images and work with their VMs.

5.3.4 Hosts and Clusters

A host is a server that has the ability to run Virtual Machines and that is connected to Open Nebula's Frontend server. Open Nebula can work with Hosts with a heterogeneous configuration. Hosts are created coding with Ruby scripts with Open Nebula. A cluster is a group of hosts; data center consists of number of clusters based on its size.

5.4 Scheduling

Implementing ruby scripts in OpenNebula for resource scheduling are to be written in "/lib/one/remotes/scripts-common.sh" and it should be configured with "/etc/sched.conf".

5.4.1 One-by-One and Distributed Approach

In this approach VMs are allocated to the host's one after another, hosts are created in OpenNebula and managed

by oneadmin account of CLI and ruby scripts. In the distributed approach VMs are distributed equally to all the hosts.

5.4.2 *Resource Wastage and Pending State Problem*

In above scheduling approaches after allocating of the full capacity of the hosts when new VM comes in to queue they will remain in pending state waiting for resources. Wastage of resources i.e. customer may not utilize all the resources that he has provided and there may be some requests in waiting state as shown in Figure 3.

5.5 Maintenance

The maintenance is the important section and this should work parallel with the resource scheduling algorithm. Here the maintenance activity of the data centre is scheduled dynamically. The ruby scripts for the maintenance is set in "/var/lib/one/remotes/im/balanced-probes.d" and they are to be registered in "/etc/oned.conf" file.

The present work sets the update time to the hosts in a dynamic manner i.e. no need for service provider to register updates manually. Present design has set update activity for the hosts one after the other depends on the number of hosts. If there are a lot of hosts they are divided in to further groups and the update activity is scheduled as per the group.

A group may be a cluster or large clusters are further divided in to groups. Each host in a group is set for update dynamically. In the present design, it consist of default cluster of OpenNebula as a group and sets maintenance time. OpenNebula also provides the checking the update facility through logs and helps us to observe the how updates are going on in the data centre in detail for each second.

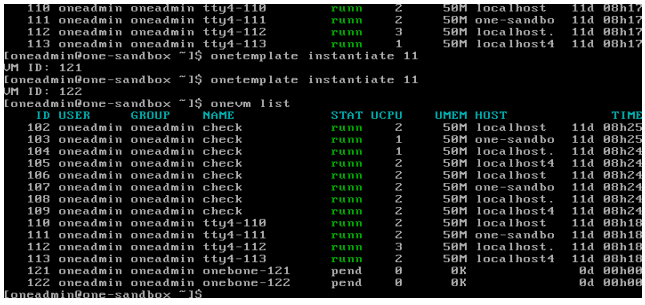

Figure 3. Pending state and resource wastage problem.

5.6 Distributed Load Aware Scheduling with Maintenance Approach

Here with ruby scripts both maintenance and scheduling for resources is achieved parallel by creating additional resources in OpenNebula toolkit to check efficiency of proposed model and this will solves the problem of pending state and resource wastage.

As shown in Figure 4 when all the resources of CPU is finished and instead of keeping the waiting VMs in pending state they are assigned to those hosts those which resources are not completely utilized.

6. Performance

As specified in the proposed system, Sunstone server is used in the design which provides graphical interface for the OpenNebula infrastructure. Sunstone server collects all the working status information about the hosts, VMs, templates and shows in a graphical format. This helps to calculate the results accurately and estimate the performance of the existing system and compare it with the proposed design.

Sunstone-server works in background collect all infrastructure information from OpenNebula. Proposed design enable port option in sunstone-server.conf file which makes "http://localhost:9869" option enable GUI of sunstone-server.

6.1 Metrics

Performance of proposed algorithm is measured by following metrics

- CPU cycles.
- Memory.

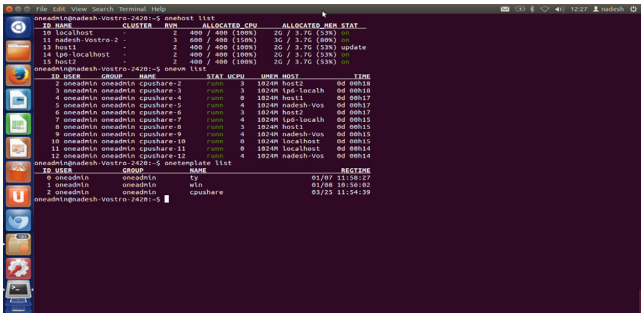

Figure 4. Distributed load aware scheduling and maintenance.

6.1.1 CPU cycles

Each host has CPU cycles measured in 100 percent and after creating of VM they will be utilized, experiment is conducted on CPU cycles of 200 assigning to the VMs. Then the existing works as well as proposed algorithms are simulated and the results are noted.

6.1.2 Memory

Each host is created with a memory that is available from the node. Different hosts of 3 sizes 480MB, 1.8GB and 3.1GB derived as per systems availability derived from ruby script "host.cc"

6.2 Parameters under Study

CPU utility, memory utility, extra allocation and host performance is shared and maintained. These four parameters are graphically evaluated for the proof of the proposed model and are explained as follows:

- CPU utility: How fast hosts allocate CPU to VMs.
- Memory utility: How fast host allocate memory to VMs.
- Extra allocation: Weather scheduling algorithm support extra allocation.
- Host Performance: How host is performing at the time of updates, CPU usage and memory utilization.

6.3 Case Study

After designing Scheduling as well as maintenance algorithms in parallel the following cases are obtained.

Case 1: Greedy scheduling model with fixed maintenance activity.
Case 2: Load balanced One-by-One scheduling with dynamic maintenance activity.
Case 3: Load balanced Distributed scheduling with dynamic maintenance activity.

For each case we have enabled the same "tty-linux image" for the users to use in their VMs.

6.4 Results

Sunstone graphs for greedy approach shown in Figure 5 proves there is always downfall in host performance and there are 0 states in host resource utilization.

In dynamic distributed-load aware maintenance approach host performance is monitored in sunstone-server as per case 3 the graphs shown are shown in the Figure 6, it is observed that number of cycles allocated is consistent and no failures to 0 states. One-by-One approach has good graphs but it has the entire load on particular servers while the others left alone.

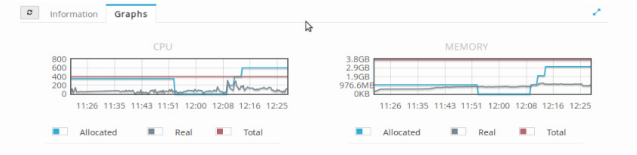

Figure 5. Greedy approach.

Figure 6. Dynamic distributed-load aware maintenance approach host performance.

Table 1. Performance measurement table

	Greedy Approach	One-By-One Approach	Distributed Approach
CPU Utilization	Weak	Fast	Consistent
Memory Utility	Moderate	Good	Good
Host Performance	No	Moderate	Fast
Update and recovery	No	Yes	Yes

From sunstone-servers GUI, the performance of OpenNebula infrastructure in the three cases is measured. Based on the performance rating is issued to the approach as weak, moderate, good and consistent.

From the results of performance table distributed load sharing as well as dynamic maintenance have good rating in all the parameters. Though CPU utilization not occurred instantly but it has consistent performance after scheduling.

7. Conclusion

The paper tackles greedy scheduling problem with load-aware distributed and one-by-one approach. The maintenance algorithm is designed in dynamic approach which will co-ordinate with scheduling approaches, the metrics performance compared with earlier related work proved efficient and consistent. Proposed model reduces the maintenance activity time and provides a dynamic

mechanism. These algorithms are completely adaptive to situations and provide dynamic provisioning and time maintenance which is the main theme of this paper.

This work also makes research scholars to think more about maintenance time of data centers and improve the existing algorithms to work in a more effective way. Further the idea of migrating VM from host to host of different clusters is raised which leads effect on VMs performance and avoided in this design.

8. Future Work

Designed algorithms overcome the issue of pre-emptive schedule indirectly by migration which does not decrease VM performance. Pre-emptive in case of data center resource provisioning means the VM working at one host of a cluster. It should be migrated to a host of other cluster without degrading the performance.

One-by-One approach will have more CPU utility. Approaching in this area will increase the performance factors still more as better CPU utility will increase resource allocation to number of VMs in very less time, mean while the unused hosts can be set to rest.

9. References

1. Nathani A, Chaudhary S, Somani G. Policy based resource allocation in IaaS cloud. Future Generat Comput Syst. 2012 Jan; 28(1):94–103.
2. Javadi B, Abawajy J, Buyya R. Failure-aware resource provisioning for hybrid Cloud infrastructure. J Parallel Distr Comput. 2012 Oct; 72(10):1318–31.
3. Vecchiola C, Calheiros RN. Karunamoorthy D, Buyya R. Deadline-driven provisioning of resources for scientific applications in hybrid clouds with Aneka. Future Generation Computer Systems. 2012 Jan; 28(1):58–65.
4. Le G, Xu K, Song J. Dynamic resource provisioning and scheduling with deadline constraint in elastic cloud. International Conference on Service Science; Beijing University of Posts and Telecommunication. 2013 Apr 11–13. p. 113–7.
5. Skalkowski K, Slota R, Krol D, Kitowski J. QoS-based storage resources provisioning for grid applications. Future Generation Computer Systems. 2012 Mar; 29(3):713–27.
6. Xu L, Chen W, Wang Z, Yang S. Smart-DRS: A strategy of dynamic resource scheduling in cloud data center. IEEE International Conference on Cluster Computing Workshops; Beijing. 2012 Sep 24–28. p. 120–7.
7. Balman M. Advance resource provisioning in bulk data scheduling. 27th International Conference on Advanced Information Networking and Applications; Barcelona. 2013 Mar 25–28. p. 984–92.
8. Abu Sharkh M, Ouda A, Shami A. A resource scheduling model for cloud computing data centers. Wireless Communications and Mobile Computing Conference; Sardinia. 2013 Jul 1–5. p. 213–8.
9. Calheiros RN, Vecchiola C, Karunamoorthy D, Buyya R. The Aneka platform and QoS-driven resource provisioning for elastic applications on hybrid clouds. Future Generat Comput Syst. 2012 Jun; 28(6):861–70.
10. Islam S, Keung J, Lee K, Liu A. Empirical prediction models for adaptive resource provisioning in the cloud. Future Generat Comput Syst. 2012 Jan; 28(1):155–62.
11. Sarma VAK, Rajendra R, Dheepan P, Kumar KSS. An optimal ant colony algorithm for efficient VM placement. Indian Journal of Science and Technology. 2015 Jan; 8(S2):156–9.
12. Aslekar A, Damle P. Improving efficiency of data centres in India: A review. Indian Journal of Science and Technology. 2015 Feb; 8(S4):44–9.
13. Dhingra A, Paul S. Green cloud: Heuristic based BFO technique to optimize resource allocation. Indian Journal of Science and Technology. 2014 May; 7(5):685–91.