International Journal of Research in Computer and Communication Technology ISSN (O) 2278-5841 ISSN (P) 2320-5156

Para-Scheduling Algorithm for Architectural Framework of Virtual Computing Lab

Vishal Gaurav, Aniket Trivedi, Amit Pandit, Nishant Gupta,
Prof. Sukhada Bhingarkar, Prof. Bharati Ainapuri
MIT College of Engineering
Pune, India
vishalgaurav610@gmail.com, aniketttrivedi04@gmail.com, amitpandit09@gmail.com
nishant442201@gmail.com, Sukhada.bhingarkar@gmail.com , bharti.ainapuri@gmail.com

Abstract— **Our main concern in private cloud computing is to provide sufficient computing resources on demand, however in current architectures like eucalyptus, cluster controller is the sole provider of virtual machines to the user but due to extensive usage of computational resources, Cluster Controller becomes a bottleneck, hampering the performance. We are working on a Para-Scheduling Algorithm on an already proposed architecture which is based on a Master-Slave concept having one Master and multiple slaves having a provision that the requesting client can register itself as a slave. Our Para-Scheduling algorithm focuses on prioritizing client requests, handling the different conflicts that may arise, handles the race in conditions and also removes the bottleneck problem. The proposed Para-scheduling algorithm consists of certain parameters that will be deciding the priorities and thereby scheduling the requests for the resources. Thus, this algorithm ensures that, conflicts (racing conditions) being handled and taken care of and no bottleneck problem will arise.**

Keywords – Master slave architecture, Private Cloud, Para-Scheduling, Parameters.

I. INTRODUCTION

In the current field of Information Technology in recent years the concept of cloud computing and virtualization has gained an important status. These technologies help reducing cost via machine utilization, reduced administration time and infrastructure cost, because of these factors many organizations have started implementing this technology. In the market, already many cloud toolkits are available like Amazon's EC2 [1][6], Eucalyptus[2][7]. Main problems faced by the Eucalyptus are lack of certain features like virtual machine reservation and portability of cloud Master functionality. The main problem with Eucalyptus is that it lacks certain features such as virtual machine reservation, bottleneck and portability of cloud server functionality. In Ideal scenario the requesting client has no need to plan far ahead for provisioning because ideally infinite computing resources are available on demand, but in reality there is only limited number of resources.

To overcome the drawbacks of the Eucalyptus, an algorithm is proposed for Para-scheduling along with the master-slave architecture [3][4][5]. This paper presents a prioritizing and Para-scheduling algorithm for the master-slave architecture which allows increase service capability through dynamic discovery of resources across the network and thus reducing the probability of occurrences of network bottleneck. This paper is intended to make the reader informed about the proposed scheduling algorithm.

Primarily there are four types of cloud:

- Public
- Private
- Hybrid
- Community

Each having their advantages and disadvantages with significant applicability for any specific organization researching or actively considering a cloud deployment.

Public Cloud:

The cloud computing model having various services such as application and storage are available over the Internet for general usage. These Public cloud services may be offered on a pay-per –usage mode or other purchasing models for example IBM's Blue Cloud [8][12].

Private Cloud:

A private cloud is a virtualized data center that operates within a firewall. Private clouds are highly virtualized, joined together by mass quantities of IT infrastructure into resource pools, and privately owned and managed [8][12].

Hybrid Cloud:

Mix of both the public and private clouds is hybrid cloud Community Cloud Community cloud is an infrastructure shared by several organizations supporting a specific community according to its purpose [8][12].

Community Cloud:

A community cloud is an infrastructure shared by several organizations which supports a specific community [8][12].

II.RELATED WORK

Over the past few years many architecture have been implemented, such as Eucalyptus [2][7], Nimbus [7][9] and Open Stack [11] etc. Scheduling of clients for the purpose of resource reservation is an important criteria because there are various parameters associated with client which are needed to be considered along with the availability of resources. Thus, there are various drawbacks with scheduling of clients with different architectures.
VMware's vSphere [13] and Platform's VM Orchestrator [14] are embedded hypervisors that does not require operating system. Initially the scheduling of VM's is done based on CPU load of each physical host and afterwards dynamic scheduling is performed by monitoring utilization of available hosts. Thus, the user can define the priorities and other parameters for the scheduling purpose, but it is not possible to change the scheduling policies.
In Nimbus architecture [7] there are limited pre-configured scheduling policies i.e. first fit, round robin, greedy and green it. Also, its Virtual Management capabilities are limited. As in Open Nebula [6][7][10], the initial scheduling part is based on requirements or rank policies to prioritize those resources. It supports any static or dynamic placement policy. Open Nebula is further extended by adding functionality of leases, which are extended user request for computational reservations. These can be used to define a period of time for which the respective resources should be available. Ovirt [15] is a free Red Hat/ Fedora virtualization management system specific to it in which scheduling is manual.
Eucalyptus being the only platform/architecture offering the hierarchical structure having three pre-defined scheduling algorithm: round robin, greedy and green-it. But, these policies being static i.e. scheduling decisions made only for the new requests. This becomes one of the major drawbacks of eucalyptus [2] [7].

III. MASTER SLAVE ARCHITECTURE

The architecture is based on master-slave approach which consists of one master and one slave at the beginning. First of all, the slave has to register itself to the master to be part of cloud. After the registration with the master, the master queries the slave about the available resources. The slave replies to the master and master simultaneously updates its database about the number of cores it has. All the images are bundled and uploaded to the master so that they can be launched on a slave. This architecture grows dynamically when new clients are added to a set of slaves.

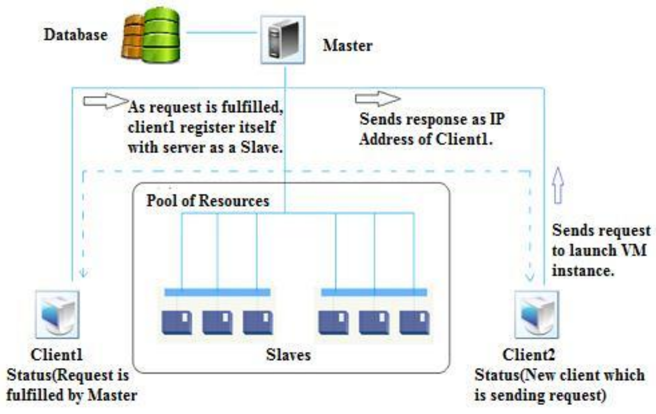
Database
Master
As request is fulfilled, client1 register itself with server as a Slave.
Sends response as IP Address of Client1.
Pool of Resources
Sends request to launch VM instance.
Client1
Status(Request is fulfilled by Master
Slaves
Client2
Status(New client which is sending request)

Figure 1: Master Slave Architecture

This architecture allows new clients to request virtual machines, and the master makes the choice of running the requested virtual machine either on previously available slaves, or on the clients who are recently registered into a set of slaves. It allows setting up node controller functionalities on machines over network that increases the capability to run more virtual instances.
The architecture also provides basic cloud computing features like launching virtual instances, keeping track of status of virtual machines, which are running on node controller, addressing of these virtual machines, NATing and mapping of virtual machines [3][4][5].

IV. PROPOSED PARA-SCHEDULING ALGORITHM

To solve the problem of scalability and limitation in resources, this paper proposes a Para-Scheduling algorithm which would not only utilize the resources of the cloud master, but would also provide an efficient way to schedule the request having greater priority. Every request has various parameters like type of resource demanded, amount of resource required, time for which the resource is required. Considering all the parameters, it requires an algorithm, which should provide the resources to the client in an efficient manner. Also, the need to maintain at least resource available in the pool of slaves should be satisfied by the algorithm.
If such efficiency is not achieved, then there could be situation where even the resources is available but cannot be allocated to the client. In order to overcome this efficient resource utilization problem, the algorithm is to be used in the architecture.
Here,
R_i : is request id.
A_i : is i'th client.
S: is set of slaves.

1. Client sends a request to master with the required information.

2. Master have list of requests R_i of client A_i .

3.1.i) If there's only one request then master checks the availability of resources.

ii) If the resource is available then a connection is established between current client and the slave.

3.2. i) If the resource is not available then the master checks the priority of a client using the resource and the client demanding the resource.
ii) One having greater priority get access to resource.
iii) If both have same priority then the demanding client has to wait until the resource is free.

4. i) If there are multiple requests, then

ii) First of all priority of the entire request is calculated.
iii) If two or more requests have same priority then they are scheduled on first come first serve basis.
iv) Next step is checking the availability of resources which is same as for one request.

5. After permission checking, availability of requested resource is checked

i) If the resource is available then it is assigned to the client

ii) Else given resource is installed and then assigned to the client.

6. When clients request is fulfilled, the resource is released and database is updated.

7. Now if the client wants to register itself as a slave it is registered and S is updated as

$$S = \sum Sk \cup ai$$

8. While there are unprocessed request go to step 3.

9. End

The representation of algorithm by flowchart:

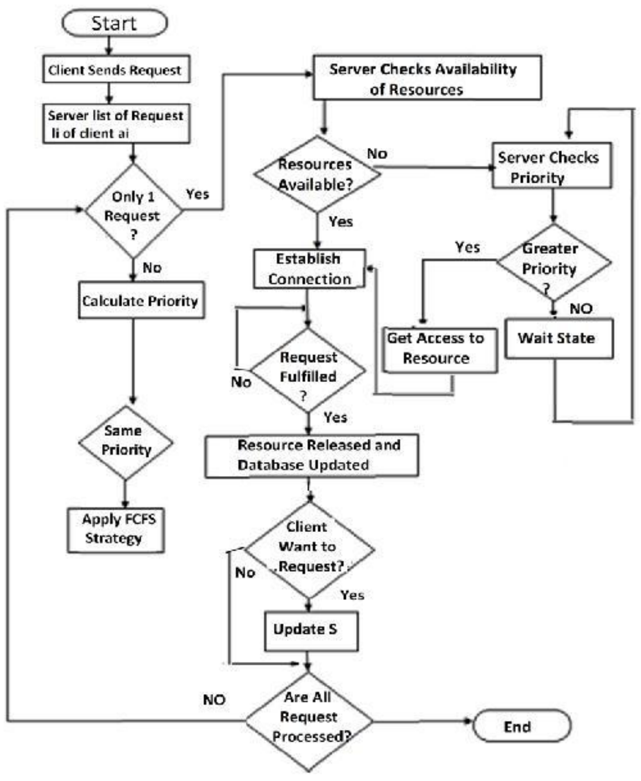

Figure 2: Flow chart of algorithm

V. MATHEMATICAL MODELLING

Mathematically the problem can be expressed as follows,
Let M be a master with finite set of resources defined by following tuples

M = (c, m, s, b)

Where,

C = CPU resources
M = memory resources
S = disk storage
B = network bandwidth

Let A = (a1 , a2 , a3 , -------------, an) be a set of clients , and let S = (s1 , s2 , s3 , --------------, tin) be a set of slaves , where each slave $Sk = (Sk^{CPU}, Sk^{RAM})$
Specifies the CPU capacity and the memory capacity of the Virtual Machine.
When client AI makes a request or VM instance to the master M,
M checks the priority of request s and then the request with maximum priority is forwarded to a slave Sk. Sk allocates necessary resources for launching requested VM instance.
After running VM instance, if AI wishes then it is registered as a slave into set S.
Hence S is updated as,

$$S = \sum Sk \cup ai$$

Calculation OF Priority

$$PF = PfV + RAS - \log_2 R - \ln E + MON$$

Where,

PfV = Platform value
RAS = Re as a Slave
R = RAM required in MB
E = Execution time in minute
MON = Member or Not

Actual Priority

$$Pi = 10 + 0.20PF$$

Based upon the Pi value master will decide the scheduling of request

If two requests have same Pi value, then check which of them is ready to register as slave, one with positive value is scheduled first.

If both have same value of RAS, then schedule it on first come first serve basis.

VI.BENEFITS OF PARA-SCHEDULING ALGORITHM

On applying our Para-Scheduling Algorithm on master slave architecture, we get following benefits:

1. Sufficient number of resources available on demand: When a new client sends a request to the master, master asks the client whether it wants to register itself as a slave or not. If the client registers himself as a slave, then the amount of resources available increases dynamically, ensuring that sufficient number of resources will be available on demand.

2. Proper and efficient use of resources: In this scheduling algorithm, we are calculating priorities and based upon the priority, we are scheduling the clients. So, always a client having greater priority and need of resource will be scheduled, ensuring that resources are used properly and efficiently.

3. Conflicts and Race conditions handled are addressed: Conflicts come when two requests have same priority. This leads to a race condition of utilization of resources. Our algorithm has provisions to solve such conflicts by scheduling them efficiently.

VII. CONCLUSION AND FUTURE WORK Using the

proposed priority algorithm we can provide per-emptive services to a new client having greater priority. This reduces the problem of utilization of resources by providing the resource for each new request by considering various parameters. The Para-scheduling algorithm decides the client to which the available resource is to be granted along with, which client is to be given first priority.

Thus, a more reliable and efficient Virtual lab can be developed by proposing the above scheduling algorithm in the already implemented Master-Slave architecture. It will prove a noteworthy solution in today's exponentially increasing demand of "Cloud Services" having various priority parameters and facing conflicts.

In future, the problem of resource per-emptive service can be solved by considering various other parameters in master-slave architecture. If the requesting client and the slaves have same priority, then from which slave, the resource is to be taken away can be decided by evaluating number of other parameters.

REFERENCES

[1] Amazon EC2 Spot Instances. "http://aws.amazon.com/ec2/spot-instances/."

[2]Eucalyptus Home Page. "http://www.eucalyptus.com/."

[3] Borja Sotomayor, Rub n S. Montero, Ignacio M. Llorente, and Ian Foster "An Open Source Solution for Virtual Infrastructure Management in Private and Hybrid Clouds" "IEEE INTERNET COMPUTING, SPECIAL ISSUE ON CLOUD COMPUTING July 7, 2009"

[4] Peter Sempolinski and Douglas Thain University of Notre Dame "A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus" "2nd IEEE International Conference on Cloud Computing Technology and Science DOI 10.1109/CloudCom.2010.42 '".

[5] Jithesh Moothoor and Vasvi Bhatt "A Cloud Computing Solution for Universities: Virtual Computing Lab" "Case study of North Carolina State University's Virtual Computing Lab 15 Dec 2009".

[6] Nimbus Home Page. "http://www.nimbusproject.org/. "

[7] Open Nebula Home Page. "http://www.opennebula.org/. "

[8] Open Stack and its technology: "HTTP://anticommunism/cloud/tools/open stack/reference-architecture".

[9] Services provided by cloud: "HTTP://blithesome/what-types-of-cloud-services-are-available/"

[10]VMware and Cloud computing http://www.vmware.com/files/pdf/cloud/Vmware-and-Cloud-Computing-BR-EN.pdf

[11]Platform "Platform VM Orchestrator.[Online].Available : http://www.platform.com/resources/datasheets/vmov4-ds.pdf.[Accessed:July8,2010]."

[12] oVirt http://ovirt.org