A New Adaptive Private Cloud Architecture

Prof. Bharati Ainapure, Prof. Sukhada Bhingarkar, Prof. Devan Shah

Assistant Professor, MIT College of Engg, Paud Road, Pune

Professor, Terna College of Engg. & Technology, Nerul, Mumbai

Nerul, Mumbai

Abstract: Cloud computing is the delivery of computing as a service rather than a product, whereby shared resources, software, and information are provided to the users and other devices as a metered service over a network. Public cloud has the property of infinite computing resources available on demand. Private cloud is infrastructure operated solely for a single organization. The architecture of Private cloud does not differ from public cloud except elasticity and unlimited resources. An open-source framework like Eucalyptus can be used to build private cloud in college environment. However, in Eucalyptus architecture, extensive usage of computational resources results into bottleneck, hampering performance of private cloud computing environment. To overcome these issues of Eucalyptus, this paper proposes adaptive cloud approach based on master-slave concept where one master and multiple slaves serve the resources to the clients. It allows setting up node controller functionalities over the machine on the network thus increasing capability of the private cloud. This architecture allows new clients to request virtual machines, and the server 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. Thus this architecture reduces the probability of occurrence of network bottlenecks and ensures that sufficient resources are always available to the end users.

Keywords: Cloud Controller, Master, Node Controller, Private cloud, Slave

1. INTRODUCTION

Over the past few years, the concept of cloud computing and virtualization has gained much momentum and has become a more popular phrase in information technology. Many organizations have started implementing this new technology to further reduce cost through improved machine utilization, reduced administration time and infrastructure costs [1]. There are many cloud toolkits that can be used to transform existing infrastructure into an Infrastructure as a Service (IaaS) cloud with cloud-like interfaces such as Eucalyptus. It is compatible with Amazon’s EC2 interface and is designed to support additional client-side interfaces [2][3]. However, the main hindrance in Eucalyptus is that it lacks in certain features such as virtual machine reservation and portability of cloud server functionality. While providing private cloud-computing environment, according to user perception, infinite computing resources are available on demand, thereby eliminating the need for Cloud Computing users to plan far ahead for provisioning. But in reality only finite set of computational resources are available at server end and details of implementation is masked to the cloud user by levels of abstraction over these resources. Also, as cloud services are needed to be transferred across the network to the clients, we need to consider data transfer cost also. To overcome drawbacks of Eucalyptus, there is a need to provide an architecture that solves the problem of paucity of resources and resolve the issues related to portability and network bottleneck. This paper presents architecture called as adaptive cloud for private cloud computing which would allow cloud computing environment to dynamically extend its computational power, thus allowing increased service capability through dynamic discovery of resources across network and thus reducing the probability of occurrence of network bottlenecks. The adaptive cloud approach consists of one master and multiple slave machines serving to the users and clients are added dynamically and the instances are running on the slaves. It allows setup node controller functionalities over the machine on the network thus increasing capability of the cloud. This architecture is capable of running lesser priority virtual machines (VM) on client machines running our specialized client Operating System (OS). This 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, NATting and mapping of virtual machines on network to real world machine. Adaptive Cloud also provides Cloud server facilities to be easily made available at client end. This involves provision of graphical interface to check status of request and allow client to easily launch virtual machine. This approach provides vast improvements over existing architecture of cloud by extending server resources by including client hardware and ensures that sufficient resources are always available.
The rest of the paper is structured as follows: Section 2 discusses related work. In section 3, the statement is made about the problem that is addressed in this paper. Section 4 elaborates on proposed architecture of adaptive cloud. Section 5 presents experimental results. Finally, Section 6 gives conclusion and future work.

2. RELATED WORK

There are several architectures for cloud platforms that have been published in recent years. Some of examples of these platforms are Eucalyptus, Nimbus, OpenNebula etc. Although most existing cloud computing implementations share the common high-level notion of flexible, scalable and dynamic computational provisioning, there is significant variation in exactly how that power is presented to the end user. Some systems such as Amazon’s Elastic Compute Cloud (EC2) and Enomalism allow users to allocate entire virtual machines on demand.

VMware’s vSphere[4] and Platform’s VM Orchestrator[5] are embedded hypervisors that run directly on server hardware without requiring an additional underlying operating system. The initial mapping of virtual machines is done based on CPU load of each physical host. Then dynamic mapping is performed by monitoring utilization of available hosts. The priorities and other parameters for the mapping can be defined by the user, but it is not possible to change the mapping policies. These tools also lack other features that are relevant for building IaaS clouds, such as public-cloud like interfaces, mechanisms for adding such interfaces easily and the ability to deploy virtual machines on external clouds.

OpenNebula[6] is a part of EU’s Reservoir project that uses advance reservation, preemption scheduling and VM placement constraints for balance of workload process. The Matchmaking Scheduler[7] (MMS) is the OpenNebula build-in scheduler module. The MMS first filters out the hosts that do not meet the virtual machine requirements and do not have enough resources available (CPU, memory) to run the VM. Based on the information gathered by the monitor drivers, a rank is assigned for each resource in the resulting list. The resources with a higher rank are used first to allocate virtual machines. OpenNebula is further extended to include Haizea [6] that adds the functionality of leases. The leases are extended user requests for computational reservations. They can be used to define a period of time, when the resources should be available. The virtual machine placement strategies support queues and priorities. Currently, Haizea supports three type of leases and uses several optimizations techniques, such as reusing disk images across leases in order to minimize the time required for the VMs preparation.

oVirt[8] is a free Red Hat / Fedora specific virtualization management system that allows to manage virtual machines through a web interface using libvirt. The libvirt library allows oVirt to manage virtual machines hosted on KVM virtual machine servers. However, it is unable to scale to external clouds. It has monolithic and closed architecture that is hard to extend or interface with other software, not allowing seamless integration with existing storage and network management solutions deployed in data centers. Its placement policies are also manual.

Apache Hadoop is a framework for running applications on large clusters built of commodity hardware. The Hadoop framework transparently provides applications both reliability and data motion. Hadoop implements a computational paradigm named Map/Reduce, where the application is divided into many small fragments of work, each of which may be executed or reexecuted on any node in the cluster. In addition, it provides a distributed file system (HDFS) that stores data on the compute nodes, providing very high aggregate bandwidth across the cluster. Both Map/Reduce and the distributed file system are designed so that node failures are automatically handled by the framework [9].

The Cumulus project is an on-going Cloud computing project at the recently established Steinbuch Centre for Computing (SCC) at the Karlsruhe Institute of Technology (KIT). It intends to provide virtual machines, virtual applications and virtual computing platforms for scientific computing applications. The Cumulus project currently is running on high performance HP and IBM blade servers with Linux and the Xen hypervisor [10].
3. PROBLEM FORMULATION

Private cloud can be designed and configured to serve both the educational and research missions of an institute in a very economical and cost efficient manner. In an institute the usage of resources will vary depending on the academic calendar. Private cloud provides good scheduling mechanism to identify the movements and flows of campus activities. With the help of desktop and HPC utilization, it provides efficient utilization of the computational infrastructure in institute’s labs[1][11]. Eucalyptus[3] is an open source tool that offers complete IaaS solution on the top of hypervisors. Eucalyptus was used to create private cloud that helped in college environment. It had following features:

1. Some of the existing desktops were clustered to create private cloud. Thus we did not need to invest in additional capacity.
2. The stakeholders of a college; students and staff had their own storage space in cloud to store their daily data. The final year projects were also stored on this cloud to have them available for later reference.
3. This private cloud allowed students to launch virtual machine instances based on various practical experiments.
4. The student pen drive can’t be detected in VM. Hence students will not be able to use pen drive. Though virus gets downloaded within VM, it will die down along with VM, hence it will not affect other machine.
5. The students cannot misuse the internet connectivity as each virtual machine instance is allotted a fixed amount of Internet bandwidth [12].

However, private cloud environment provided by Eucalyptus suffers from certain drawbacks. e.g. to serve 48 students in a laboratory simultaneously, we had setup a Server with 2 Xeon 6 core each processor and 48 GB RAM. But in a college, out of 700 machines, at least 250 machines are required to run virtual machine instances for performing Local Area Network (LAN). This limits the number of resources and thus the perception of infinite resources in a private cloud fails. This leads to design an architecture which ensures that sufficient resources are always available and the cloud never dies.

4. PROPOSED SOLUTION: ADAPTIVE CLOUD

4.1 System Architecture

To solve the problem of scalability and limitation in resources, this paper proposes an architecture which would not only utilize the resources of the cloud server, but would also be capable of running lesser priority virtual machines on client machines running our specialized client OS. The adaptive cloud 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.

This architecture allows new clients to request virtual machines, and the server 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 proposed solution 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. Figure 1 shows system architecture of Adaptive Cloud.
The architecture consists of following components:

**Client1:** It is a machine that wants to launch VM instance using cloud service in the network, it sends request to server machine along with the parameters specifying ram, VM image type. Whenever it utilizes cloud service, it can publish its own resources to the cloud architecture so that client1 become a slave for other clients and other client machines can make use of its resources and launch VM instances.

**Client2:** It is other client machine that wants to launch VM instance.

**Server:** This machine acts as a master. It is responsible to fulfill the requests of clients for launching the VM instances. Whenever any client machine sends request to the server machine for VM instance, corresponding response is sent back to the client specified by status of VM instance and IP address of requested VM instance so as to connect with the VM.

**Slaves:** It is a pool of resources. When client request is fulfilled, client registers itself with server machine. Client machine then announces its resources with server machine and server adds client machine in the resource pool and monitors its status.

**4.2 Flow of Events**

Figure 2 illustrates the flow of events in the system.
The description of flow of events is as follows:
1. Client1 requests an instance to the Master.
2. This request is evaluated by the master machine and accordingly forwards this request to the suitable slave machine.
3. The Master machine gives the IP address of slave machine to the client1 for any further connections. Once the client1 knows the IP address of the slave, it directly connects with the slave machine.
4. Client1 sends its address to the database so that it can perform the function of slave for any future calls to the master.
5. Client1 is added to the pool of resources, and is registered as a slave with the master.
6. New Client2 sends request to the Master.
7. Master forwards request to client1 provided that sufficient resources are available with client1.
8. If Client1 does not have enough resources, the master forwards the request to the set of slaves.
9. Client2 sends its address to the database so that it can perform the function of slave for any future calls to the master.
10. Client2 is added to the pool of resources, and is registered as a slave with the master.

Thus, this architecture ensures that there are always sufficient resources available at the disposal of the master, and thus can accept any number of incoming connections.

4.3 Experimental Setup

The experimental setup is as follows:
1. Software interface for user on the client machine will be his/her computer having ubuntu as its operating system.
2. The hardware should provide Virtualization Technology and other necessary compatibility.
3. The ubuntu machine should be running TightVNCViewer.
4. Web Interface is provided for users to launch a VM.
5. Tight VNC Viewer manages all UI related with running the VM.
6. The Master is connected to a DHCP Server, which may or may not be on a different machine.

Table 1 presents machine configuration for the adaptive cloud.

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Machine Configuration</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>64-bit Ubuntu 9.10 server with 48GB RAM</td>
<td>Master</td>
</tr>
<tr>
<td>2.</td>
<td>32-bit/64-bit Ubuntu with minimum 4GB RAM having VT enabled.</td>
<td>Client</td>
</tr>
</tbody>
</table>

The assumptions for this architecture are as follows:
1. This architecture solely focuses on to check the possibility whether cloud solution can extend their capability or not.
2. There are no security considerations while providing this architecture.
3. Architecture does not address scalability issues of the cloud.
4. The clients are manually registered as a slave.
5. EXPERIMENTAL RESULTS

The proposed architecture is evaluated by running 5 instances on client machines. The results are drawn with the help of performance graphs showing CPU, Memory and network utilization.

Figure 3 presents CPU utilization where 24 CPU cores are launched. In this graph we can see that about 12% of CPU is utilized at the time of no instances but as the instances are increased upto 5, the CPU utilization gets incremented upto 86% between 10 to 60 seconds of time period.

Figure 3. CPU Utilization

Figure 4 shows Memory utilization. As the instances are increased, the memory utilization reaches upto 5.6% over a period of time.

Figure 4. Memory Utilization

Figure 5 shows performance graph of Network Bandwidth utilization. When there are no instances, 13KB/s is the network utilization whereas it grows upto 40 KB/s as the number of instances reaches to 5.

Figure 5. Network Utilization
Hence performance graphs show that as the number of instances is increased, there is increased utilization of CPU, Memory and Network resources. But though the load on the system it does not result into bottleneck.

6. CONCLUSION AND FUTURE WORK
The current architectures available for private cloud services provide Virtual Machines to clients using the resources of the cloud server. In a situation of high demand, the server gets saturated, and is forced to refuse further requests. Hence this paper aims to correct these issues by implementing a novel architecture, adaptive cloud that improves the performance of the server and allows cloud servers to extend their computational power, thus allowing increased service capability. An adaptive cloud starts with one master and one slave but grows dynamically as new clients are added to a set of slaves. It ensures availability of sufficient resources.

In adaptive cloud architecture, master does caching of virtual machine instances. When new request comes to the master, master needs to check for available resources in the pool of slaves as well as with previous client. This leads to more time required to launch an instance for a new client. Hence future work can be extended to the automatic management for registration and allocation of computational resource to the cloud users. The future work also includes a work to be done to reduce time required for launching an instance for new request.

REFERENCES