A Novel Architecture Style: Diffused Cloud for Virtual Computing Lab

Deven N. Shah
Professor
Terna College of Engg. & Technology
Nerul, Mumbai

Sukhada Bhingarkar
Assistant Professor
MIT College of Engg.
Paud Road, Pune

Bharati Ainapure
Assistant Professor
MIT College of Engg.
Paud Road, Pune

ABSTRACT
Using Eucalyptus Systems' private cloud solution, an institute can build Virtual Computing Lab (VCL) that can satisfy the requirements of an institute, but it is assumed that infinite computing resources are available on demand thereby eliminating the need for cloud computing users to plan far ahead for provisioning.. However, due to extensive usage of computational resources, cluster controller (CC) component of Eucalyptus becomes a bottleneck, hampering performance of cloud computing environment. To overcome the drawbacks of Eucalyptus, diffused cloud approach is proposed based on master-slave concept where one master and multiple slaves serve the resources to the clients. This approach improves the performance of the server and would allow cloud servers to extend their computational power by dynamic resource discovery over the network. 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, thus implementing the concept "*Cloud never Dies*". In order to demonstrate the performance of this novel architecture we provide and interpret several experimental results.

General Terms
Cloud Computing, IaaS, PaaS, SaaS, Virtual Machine Instances.

Keywords
Cloud Controller, Master, Node Controller, Private cloud, Slave, Virtual Computing Lab

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 diffused 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 diffused 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 setting up 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. Diffused 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 diffused cloud. Section 5 presents experimental results. Finally, Section 6 gives conclusion and future work.

2. RELATED WORK
There are several architectures for cloud platforms have been published in these recent years. Some of examples of these platforms are Eucalyptus, Nimbus, OpenNebula etc. Several projects and products provide virtual infrastructure management capabilities using these platforms. Not all of

them are labeled with the term cloud and cannot be used complete IaaS offerings. 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 Anomalism 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.

On the other hand, projects such as Globus Nimbus [6] can help transform existing infrastructure into an IaaS cloud with cloud-like interfaces.Nimbus exposes Amazon EC2 interfaceand offers self-configuring virtual cluster support. However, it has limited choice of preconfigured mapping policies i.e. first fit, round robin, greedy and green-it. Also, its Virtual Machine management capabilities are limited.

OpenNebula[2] is a part of EU's Reservoir project which aims to develop open source technologies to enable the deployment and management of complex IT services across different administrative domains. Initial placement is based on requirements or rank policies to prioritize those resources that are more suitable for the virtual machine using dynamic information. It supports any static or dynamic placement policy. OpenNebula is further extended to include Haizea 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.

oVirt[7] 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 [8].

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 [9].

3. PROBLEM FORMULATION

VCL architecture mainly is intended for designing and configuring a private cloud computing system that 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. VCL provides good scheduling mechanism to identify the movements and flows of campus activities. With the help of desktop and HPC utilization, VCL provides efficient utilization of the computational infrastructure in institute's labs[1][10]. Eucalyptus[3] is an open source tool that offers complete IaaS solution on the top of hypervisors. Eucalyptus was used to create VCL 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 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 [11].

However, VCL 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: DIFFUSED 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 diffused 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 Diffused Cloud.

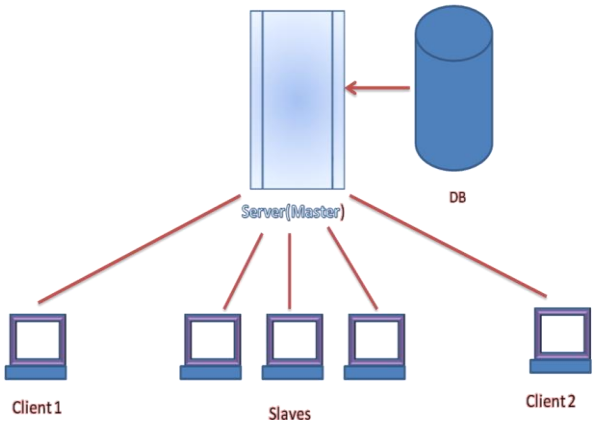

Figure 1. System Architecture

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

The flow of events happens in two parts. Part I corresponds to flow of events when Client1 requests to Master for launching VM instance. Figure 2 illustrates the flow of events of part I in the system.

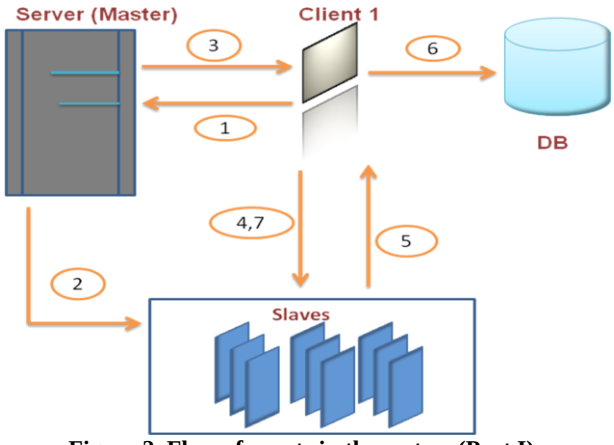

Figure 2. Flow of events in the system (Part I)

The description of flow of events in part I is as follows:
1. When the client1 wants a image, it sends a request to the master machine through a User Interface.
2. This request is evaluated by the master machine and accordingly forwards this request to the suitable slave machine.
3. The Mater machine gives the IP address of slave machine to the client1 for any further connections.
4. Once the client1 knows the IP address of the slave, it directly connects with the slave machine.
5. The slave runs a VM instance to which client1 get connected using VNC client. This instance is then used by the client1 to perform any tasks he intends to perform.
6. Client1 also sends its address to the database so that it can perform the function of slave for any future calls to the master.
7. Client1 is added to the pool of resources, and is registered as a slave with the master.

Part II corresponds to flow of events when client2 makes a request to the master for launching VM instance. Figure 3 illustrates the flow of events of part II in the system.

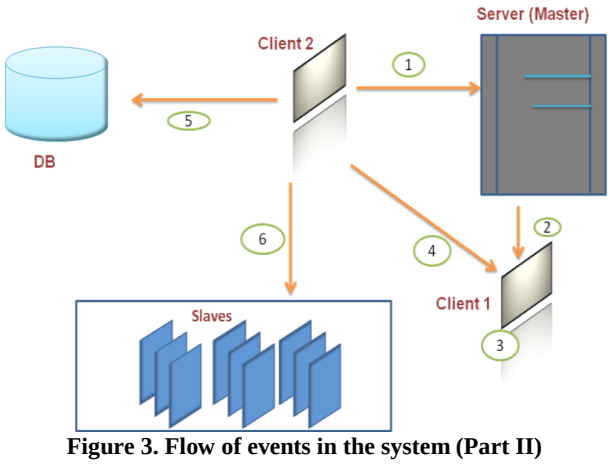

Figure 3. Flow of events in the system (Part II)

The description of flow of events in part II is as follows:
1. New Client2 sends request to the Master.
2. Master forwards request to slave (might / might not be client1)
3. Slave launches an instance.
4. Client2 connects to the instance.
5. Client2 also sends its address to the database so that it can perform the function of slave for any future calls to the master.

6. Client2 resources are 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 Mathematical Modeling

Mathematically the problem can be expressed as follows:

Let M be a Master with finite set of resources defined by following tuple: $M = (c, m, s, b)$

Where, c = CPU resources
m = memory resources
s =,Disk storage
b = Network bandwidth

Let $A = (a_1, a_2, \ldots, a_{n-1})$ be a set of clients, and let $S = (S_1, S_2, \ldots S_k, \ldots S_m)$ be a set of slaves where each slave $S_k = (S_k^{cpu}, S_k^{ram})$ specifies the CPU capacity and the memory capacity of the VM.

When client a_i makes a request for VM instance to the Master M, M forwards this request to a slave S_k. S_k allocates necessary resources for launching requested VM instance to . After running VM instance, a_i registers itself as a slave into a set S. Hence S is updated as $S = \sum S_k \cup a_i$

When another client machine a_{i+1} makes a request for VM , the resources of a_i are allocated to a_{i+1}. Then, a_{i+1} is added to slave set and in such a way, all those client machines requesting for resources are registered as slaves after their request is fulfilled. Thus S is modified to $S = \sum S_k \cup \sum a_{n-1}$.

Let $P = (p_1, p_2, \ldots p_j, \ldots p_q)$ be a set of ports on which request for VM is sent. Let p_m is a port on which M communicates. p_{ai} and p_{si} are the ports on which client and slave communicate respectively. p^{max} defines the maximum input/output requests the port can receive. Thus, $p^{max} = p_{in} + p_{out}$, where , p_{in} = Number of input requests and p_{out} = Number of output responses. Thus diffused cloud architecture ensures that sufficient resources are always available.

4.4 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 diffused cloud.

Table 1. Machine Configuration

Sr. No.	Machine Configuration	Role
1.	64-bit Ubuntu 9.10 server with 48GB RAM	Master
2.	32-bit/64-bit Ubuntu with minimum 4GB RAM having VT enabled.	Client

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 4 shows performance graph when no instances are running on any machine, where 24 CPU cores launched. In this graph we can see that about 50% of CPU is utilized along with 3.7% memory and 12 KB/s network utilization between 10 to 45 seconds of time period.

Figure 4. No instances running on any machine

Figure 5 shows performance graph when 1 instance is running on a machine. In this graph we can see that CPU utilization reaches up to 60% and 3.6 % memory utilization between 20 to 50 seconds.

Figure 5. 1 instance running on a machine

Figure 6 shows performance graph when 2 instances are running on a machine. In this graph we can see that almost 85

to 90% of CPU and 5.6% of memory utilization between 40 to 50 seconds which is the marginal increase in the utilization.

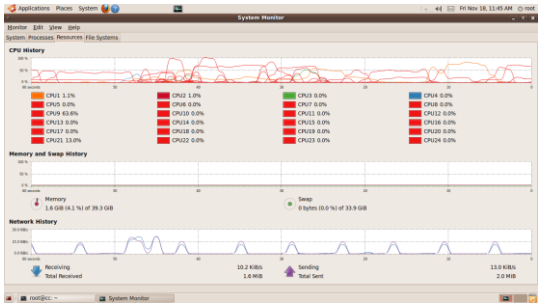

Figure 6. 2 instances running on a machine

Figure 7 shows performance graph when 5 instances are running on a machine. In the same way in this graph also we can see CPU utilization increased up to 100% with 5.6% memory, and network up to 35KB/s utilization between all tine periods.

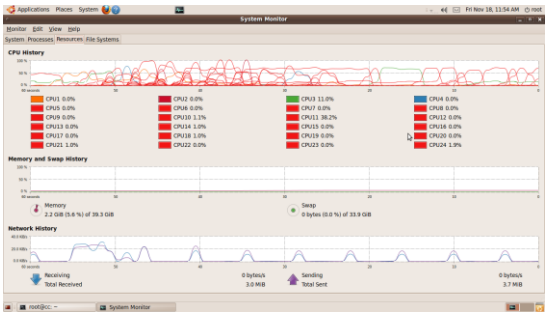

Figure 7. 5 instances running on a machine

Figure 8 shows performance graph when all instances are terminated result will be almost same as no instances are running.

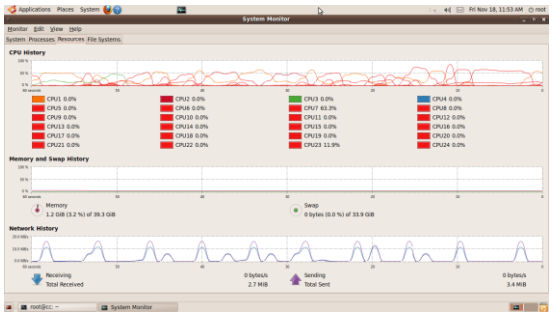

Figure 8. all instances terminating

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, diffused cloud that improves the performance of the server and allows cloud servers to extend their computational power, thus allowing increased service capability. A diffused 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 diffused 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.

7 REFERENCES

[1] Jitesh Moothoor, Vasvi Bhatt, A Cloud Computing Solution for Universities: Virtual Computing Lab, Dec. 2009

[2] Borja Sotomayor, Rub' en S. Montero, Ignacio M. Liorente, Ian Foster, An Open Source Solution for Virtual Infrastructure Management in Private and Hybrid Clouds. IEEE Internet Computing. Special Issue on Cloud Computing, 2009.

[3] Daniel Nurmi, Rich Wolski, Chris Grezgorezyk, "Eucalyptus: A Technical Report on an Elastic Utility Computing Architecture Linking your Programs to Useful System", UCSB Computer Science Technical Report, August 2008.

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

[5] Platform, "Platform VM Orchestrator. [Online]. Available: http://www.platform.com/resources/datasheets/vmov4-ds.pdf. [Accessed: July 8, 2010]."

[6] I. Foster, "Globus toolkit version 4: Software for service oriented systems," in IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, 2005.

[7] oVirt http://ovirt.org

[8] Apache Hadoop, http://hadoop.apache.org/, (accessed 20.01.2010).

[9] Lizhe Wang, Jie Tao, Marcel Kunze, Dharminder Rattu and Alvaro Canales Castellanos, "The Cumulus Project: Build a Scientific Cloud for a Data Center", CCA08, 2008

[10] Eric Jansson, Virtual Computing Labs in Liberal Education, Spring 2010

[11] Sukhada B., Bharati A., Dr. Deven Shah, "The case of private cloud for the technical institute- Virtual Computing Lab", Equinox 2011, Terna College of Engg., Mumbai