International Journal of Computer Science and Informatics

Volume 2 | Issue 2 Article 6

October 2012

Cloud Computing in Distributed System

M.H. Nerkar
GCOE, Jalgaon, nerakar.mh24@gmail.com

Follow this and additional works at: https://www.interscience.in/ijcsi

 Part of the Computer Engineering Commons, Information Security Commons, and the Systems and Communications Commons

Recommended Citation
Nerkar, M.H. (2012) "Cloud Computing in Distributed System," *International Journal of Computer Science and Informatics*: Vol. 2 : Iss. 2 , Article 6.
DOI: 10.47893/IJCSI.2012.1072
Available at: https://www.interscience.in/ijcsi/vol2/iss2/6

This Article is brought to you for free and open access by the Interscience Journals at Interscience Research Network. It has been accepted for inclusion in International Journal of Computer Science and Informatics by an authorized editor of Interscience Research Network. For more information, please contact sritampatnaik@gmail.com.

Cloud Computing in Distributed System

M.H.Nerkar & Sonali Vijay Shinkar

GCOE, Jalgaon

Abstract **-** Cloud Computing as an Internet-based computing; where resources, software and information are provided to computers on-demand, like a public utility; is emerging as a platform for sharing resources like infrastructure, software and various applications. The majority of cloud computing infrastructure consists of reliable services delivered through data centers and built on servers. Clouds often appear as single points of access for all consumers' computing needs. Commercial offerings of the cloud are expected to meet quality of service guarantees for customer satisfaction and typically offer service level agreements. The deployment of cloud computing can be easily observed while working on Internet, be it Google Docs or Google Apps, YouTube Video sharing or Picassa Image sharing, Amazon's Shopping Cart or eBay's PayPal, the examples are numerous. This paper does a literature survey on some of the prominent applications of Cloud Com- puting, and how they meet the requirements of reliability, availability of data, scalability of software and hardware systems and overall customer satisfaction.

Keywords **-** *EUCALYPTUS, QoS, Network Flow Model, AOCAPI.*

I. INTRODUCTION

Cloud computing includes hosting several services over the Internet, divided into three categories: Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a- Service (IaaS). In SaaS is a model of software deployment where a provider licenses an application to users for use as a service on demand. The vendors may host the application on web servers or download the application to the consumer device; and after the on-demand contract expires, disabling it. Google Apps, Google Docs [7], Acrobat.com and Salesforce.com are major SaaS providers.In PaaS deployment of applications is provided without the cost and complexity of buying and managing the underlying hardware and software layers, providing all of the facilities required to support the complete life cycle of the web applications. Amazon Web Services [2], Azure Services Platform, Rackspace Cloud and Google App Engine are some examples of this category. IaaS is the delivery of computer infrastructure, usually a platform virtualization; where instead of purchasing servers, software, data center space or network equipment, clients instead buy the resources as a fully outsourced service. IaaS like Amazon Web Services provides virtual server instances with unique IP addresses and blocks of storage on demand. Amazon Elastic Compute Cloud [2] and Eucalyptus [1] are prominent examples of IaaS.

This paper looks into the design considerations and system architecture of some of the well-known applications of Cloud Computing. The use of core Distributed System techniques is highlighted in such renderings. Further addressed are the issues faced by the developers before, during and after the design implementation. A comprehensive report of diverse Cloud renderings having different requirements for their system due to varying expectations of the customers is presented. Following this is a chapter on Eucalyptus that is an IaaS rendering developed for research purposes and research environment.

Next is a review of design suggestions focused on customer satisfaction by using Service Oriented Architecture or Quality of Service guarantees to the customers while optimizing the profit of the cloud vendors. The google file system discussed next focuses on the requirements of only the Google like dominance of append operations in contrast to random writes. Chapter on MapReduce that follows aims to improve performance while keeping the design as simple as using just Map and Reduce functions of functional programming. Lastly Amazons Dynamo looks into the issues of high reliability and availability while trading of consistency for achieving it.

EUCALYPTUS [1] is an open-source cloud-computing framework for research purposes that uses storage and computational infrastructure. It is composed of several hierarchical components, viz. Cloud Controller, Cluster Controller and Node Controller which interact with each other while supplying facilities to the cloud client. Cloud computing systems delivering Infrastructure as a Service dynamically provision Virtual Machine instances to the client for hosting software services. Scheduling of the VM instances is

one of the crucial questions in cloud computing. Eucalyptus attempts to solve the issues of VM scheduling, storage of data, network between the nodes of cloud and definition of user interfaces.

II. DESIGN

The four components of Eucalyptus that have their own Web-service interface for communication with other components are described as follows:

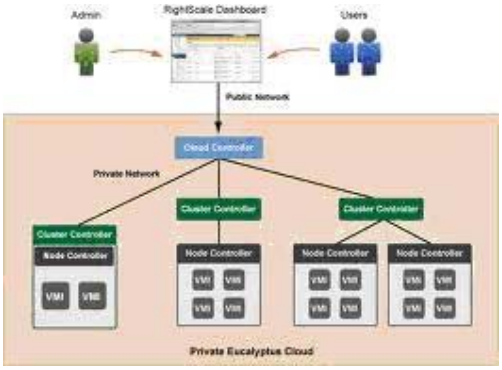

Figure: Design of Eucalyptus

Node Controller: Every node that runs Virtual Machine instances has an execution of Node Controller, NC. An NC is expected to reply to the describe Resource and describe Instance queries from Cluster Controller (CC) about the node's number of cores, memory size or disk space available; and handle its subsequent control requests of run Instance by creating virtual network's endpoint and instructing hypervisor to run instance, and terminate Instance by instructing hypervisor to end VM, rupturing network end-point and cleaning the local data.

• **Cluster Controller**: CC is the head of many NCs forming a cluster. It has the job of connecting the Cloud Controller, CLC to the NCs. It distributes the general requests of CLC to all nodes in the cluster and also trickles down the specific requests of CLC to a set of nodes in the cluster.

•**Cloud Controller**:CLC issues run Instances, describeInstances, terminateInstances and describeResources commands to a CC or a set of CCs. It manages all this information and being the only entry point to the cloud it schedules the VM instances. CLC also gives user visible interface to the cloud, for them to sign up and query the system; as well as cloud administrator interface for inspecting system component's availability.

• **Walrus**: It is a data storage device which streams data in and out of the cloud, and also stores the VM images uploaded to Walrus and accessed from the nodes. It supports concurrent and serial data transfer. Apart from these high-level components, an essential part of Eucalyptus is the **Virtual Overlay Network** which is VLAN implementation running over the top of Virtual Machines. Users attach a VM instance to a "Network" at the boot time. There is a unique VLAN tag for each such network which helps connect VMs to the public Internet and at the same time separate VMs belonging to different cloud allocations.

Associated Issues:

Designed for the academic and research purposes, Eucalyptus deploys an infrastructure for VM creation controlled by the user. During the design the main issue was use of resources found within research environment. Hence the design of Eucalyptus uses hardware commonly found in existing laboratories, including Linux clusters and server farms. The networking used is simple and flat Virtual Networking which addresses three issues.

• Connectivity:

Virtual overlay network provides connectivity of nodes to public Internet and to other nodes running VM instances scheduled by the same cloud allocation. Connectivity can be partial too, so that at least one of the VM instance from a set of instances has connectivity to Internet, using which user can log in and access all the instances.

• Isolation:

The overlay network isolates of the network of the nodes of one cloud allocation from that of the nodes of some other cloud allocation for security issues. This prevents VM instance of one cloud allocation to acquire MAC address of physical resource and interfere with VM instances of other cloud allocations on the same resource.

• Performance:

Owing to reduced performance overheads of Virtual Networking in the recent years, the use of such a network design is favored. Research is further facilitated by the modular nature of the design, helping researchers replace one component for enhancement without the need to interfere with others.

Eucalyptus' simple design is such that it just offers the basic requirement of provisioning of services [4]. It suffers from huge internal network traffic due to frequent access to data centers by the nodes. The cloud systems are configured having the peak traffic in to consideration. So most of the nodes and hence the resources are left idle most of the time [5].

In AOCAPI [5] when CC gives a *remove Instance* command to the NC, it will neither remove the disk image from the machine nor disturb the file system. It will just mark it as disabled, treated same as removing the image. Hence, it can be again be marked as enabled and can run when the image has to be reloaded. This would eliminate the overhead in fetching the disk image from data center hence reducing network overheads.

For this purpose of smart scheduling the address controller is used which decides to the address where each user request must be forwarded. The address controller consults the usage register and recent index. The usage register monitors usage of all nodes by recording the information like CPU load on a node exerted by each virtual machine instance running on it. The recent index which records recent set of nodes used by each user. It stores the address of virtual machine instance to which a request was sent last time for a user, along with time stamp to find the most recent one.

III. SERVICE ORIENTATION

The aim of a cloud computing platform is to deliver services to the cloud clients, yet most of the platforms have not yet adopted service oriented architecture (SOA) to guarantee Quality of Service guarantees to the clients. At the same time, there should be run time optimization of the cloud so as to attain maximum profit in the cloud constrained by those QoS guarantees and Service Level Agreements, SLAs between the cloud vendor and clients.

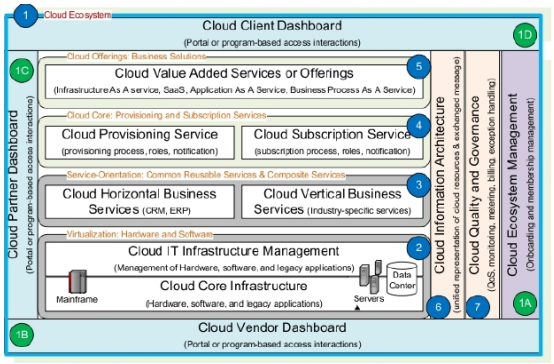

Figure: CCOA Overview[6]

The Cloud Computing Open Architecture, CCOA presented in [6] amalgamates the service oriented architecture with the virtualization techniques by its seven architectural principles and ten interconnected modules. This architecture meets the end objectives of creating scalable provisioning platform for cloud computing which can be configured based on the customer requirements, proposing shared services to provide cloud offerings to business consumers in a unified way, and maximizing business value through monetization of computing.

The first of the seven principles, illustrated along with the ten modules in Fig 3.1 also from [6], the **Integrated Cloud Ecosystem Management** includes four modules which are interdependent and give and take services from one another. Cloud Vendor Dashboard is used for managing internal operations of the cloud, Cloud Partners *Dashboard* for Cloud partners who collaborate with Cloud

Vendors to leverage services to the Cloud Client while also providing them components through its interface to the rest of the cloud. Cloud Client Dashboard is the centre of the unified framework clients use to access services, like Web portals or program based business to enterprise users channel or phone based individual customer representative channel. Cloud Vendors and Cloud Clients also interface the Cloud Ecosystem Management module which supervises cloud activities while managing memberships.

•The second principle of **Virtualization of Infrastructure** is met by using Hardware components in plug and play mode for hardware virtualization and managing software images, codes, sharing etc. for software virtualization. The module used is the Core Infrastructure of the Cloud.

•Third principle is **Service Orientation** which is provided by the Cloud Horizontal Business sub-module, which are the platform services share by a range a customers; and Cloud Vertical Business sub module which are more domain or industry specific.

• The fourth principle of **Extensible Provisioning and Subscription** for cloud segregates Cloud Provisioning Services from Cloud Subscription Services which share role defining framework and notification framework but operate provisioning process and subscription process separately.

• The fifth one about **Configurable Cloud Offerings** are the cloud business solutions in the form of IaaS like storage cloud Google Docs, SaaS like software leveraged by PayPal for customers, Application as a Service like web based development tools and Business process as a service like software testing platforms.

• The next module of Cloud Information Architecture is responsible for the effective communication of various modules with each other and helps meet the sixth principle of **Unified Information Exchange**.

• Lastly the Cloud Quality Governance module is identifies quality indicators and governs their state to use the Quality of Service parameters for defining reliability, response time and security. With this module we attain the most important principle of **Cloud Quality and Governance**.

This architecture successfully amalgamates the power of service oriented architecture missing in many cloud offerings and the existing use of virtualization technology missing in pure service oriented architectures.

IV. PERFORMANCE MODEL DRIVEN QoS GUARANTEES AND OPTIMIZATION

While the previous work focuses on how to provide services to the customers in a unified way while making the best out of the resources, this Performance Model Driven Cloud in [4] monitors the performance delivered by the cloud, ensures QoS guarantees to customers as well as optimizes the profits in the cloud constrained by these QoS and SLAs.

Performance model predicts and makes optimal decisions about many decision variables to foresee interactions among these decisions and hence optimizing decisions in autonomic control. A performance model like LQM has good correspondence to layered resource behavior. The performance parameters of LQM are external services, CPU demands of entries and requests within entries. LQM is an extended queuing network model which predicts throughputs, queueing delays, service delays and utilization of resources.

Quality of Service is a goal of Cloud management which is treated as a constraint on the resource **optimization** that is seeking maximum profit out of minimum number of resources. For a service of class c, the associated price customer pays to the application is P_C, and response time of the service is assumed to be the measure of its QoS.

The workload given to a class of service c describes the intensity of the streams of user requests for the service, in terms either of a throughput f_C for user class c, or the number N_C of users that are interacting and their think time Z_C which represents the users mean delay between receiving a response, and issuing its next request. For each service class c there is a required throughput $f_{C,min}$ or a required user response time $R_{C,max}$. $R_{C,max}$ can be expressed as a minimum user throughput requirement using Little's result:

$$f_C \geq f_{C,min} = N_C/(R_{C,max} + Z_C)$$

Now original delay requirements are changed to throughput requirements and optimization will consider only the throughput.

Network Flow Model, NFM, depicting the flow of execution commands at the processors is used for the purpose of optimization. The nodes of NFM are the entities and the arcs arcs with their weights represent the flow of demand and CPU-sec of execution per second.

Each host h has a price of CPU execution of C_h per CPU-sec, including unused CPU-sec allocated in order to reduce contention delays, In the NFM results; each task t has a reservation α_{ht} in CPU-sec per sec, on some host h. If ζ_{App} and τ_{App} are the sets of user classes and tasks involved with App.

$$PROFIT_{App} = \Sigma_c \zeta_{App} - \Sigma_{(h,t)} \tau_{App} C_h \alpha_{ht}$$

The cloud optimization is to maximize the total profits:

$$TOTAL = \Sigma_{App} PROFIT_{App}$$

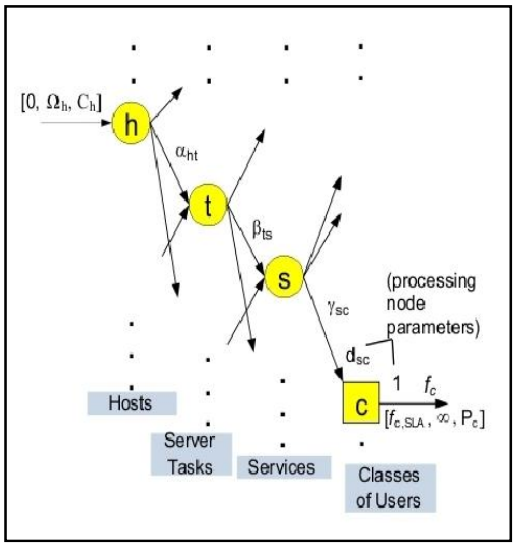

Figure: Network Flow Model[4]

This approach of optimization for profit maximization is effective as well as scalable to meet new challenges of Cloud Computing. The scalability for very large cloud shall come from scaling the performance model calculations by partitioning them to subsets of processors. These subsets can be very large though, as observed during the implementations of [4], so as to accommodate many applications. Further work is being done to account for VM overhead costs, memory allocation, communication delays and licensing costs of software replicas.

V. CONCLUSION

The design of several diverse platforms deploying cloud computing are studied in details and their issues have been highlighted for easy deployment of similar solutions for such issues. For instance simply using Linux clusters and server farms with easy and modular design of Eucalyptus developed for research environments. Applications that need to focus on delivery of best services to the customers and

maximizing profit have the aforementioned architecture to easily deliver services and satisfy the customers by use of service orientation or guarantee them QoS at the same time as optimizing the vendors profit. Some applications requiring high availability can use design similar to Dynamos, to trade of between availability and consistency. For those focusing on performance can deploy a simple MapReduce architecture of Google. Google file system apart from detailing a system that is scalable, fault tolerant and delivers high performance; also teaches to observe the customer behavior and requirements closely and optimizing to deliver them best services possible.

BIBLIOGRAPHY

[1] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Yousef, and Dmitrii Zagorodnov. The Eucalyptus Open-source Cloud-computing System. In Proceedings of Cloud Computing and Its Applications [online], October 2008.

[2] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels. Dynamo: Amazons Highly Available Key-value Store. SOSP07, October 1417, 2007, Stevenson, Washington, USA.

[3] Jefrey Dean, Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. OSDI 2004. [4] Jim (Zhanwen) Li, John Chinneck, Murray Woodside, Marin Litoiu, Gabriel Iszlai. Performance Model Driven QoS Guarantees and Optimization. In CLOUD 09: Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, Pages 1522.

[5] Karthik R, Diganta Goswami. An Open Cloud Architecture for Provision of IaaS. [Accepted]

[6] Liang-Jie Zhang and Qun Zhou. CCOA: Cloud Computing Open Architecture.IEEE International Conference on Web Services, 2009.

[7] Sanjay Ghemawat, Howard Gobiof, and Shun-Tak Leung The Google File System. SOSP'03, October1922,2003,BoltonLanding,NewYork, USA.

◈◈◈