CloudTop: An elastic and scalable paradigm for building cloud solutions

December 24, 2012

Abstract

Cloud computing solutions have gained popularity over the last few years due to a numerous reasons like ease of use, lesser initial cost and the company not having to focus on maintaining systems. One of the ways of delivering cloud service to users is by delivering it through infrastructure as a service, which is essentially providing raw desktops of desired configuration to the user. This work aims to expolore a new paradigm in building and maintaining infrastructure for providing such a service. The proposed paradigm is inspired by cellular biology,where every cell can perform all roles with varying degrees of efficiency. However, depending on certain characteristics and their competencies, they autonomically assign roles to orchestrate a bigger task. Such a system is characterized by the absence of a central controller and the cells communicate among themselves to assign roles. This mechanism gives the system a true distributed and elastic nature as opposed to the other solutions currently published. The proposed work is partially implemented called CloudTop.

0.1 Introduction

A lot of enterprises are migrating their operations towards cloud based solutions which are increasingly becoming ubiquituous in the computing sector. At a very broad level, cloud computing can be classified into:

- Software as a service
- Platform as a service
- Infrastructure as a service

When a software which is hosted by the cloud provider can be accessed by a client through a thin client and the user pays only for his usage, it is usually referred to as software as a service. When a user is provided with a fully working software library or API which the user can use to program custom software. This provides more granulity in terms of control over the software that the enterprise uses compared to software as a service. When a computer system is provided through a virtual machine, it is often referred to as infrastructure as a service. Amazon Elastic is one of the popular infrastructure as a service providers commercially.The proposed paradigm is used to build and maintain infrastructure required to provide infrastructure as a service.However, such a design can be extended to other types of cloud services and any distributed system in general. This would necessiate some changes to the current design and is a topic not explored in the current work. The current work will focus solely on building an infrastructure to provide infrastructure as a service.

Our design is inspired by the cellular model found in biological systems.Initally, all the cells are identical to each other. After a point of time, however these cells assign themselves different roles and morph into muscles, nerves, tissues etc. thereby performing specific roles so that the human body functions as it is supposed to. In the proposed architecture, we argue that different systems which make up a cloud are like cells in the human body. Each of these systems is capable of performing any role in the cloud system. Thus, all the systems in a cloud are perfectly identical to each other in terms of the roles that it *can* perform. However, not all system's efficiency to perform a given task would be equal.Some systems would perform a task better than some other system. Thus, it's only the efficiency in performing a role that the different systems differ in and not the ability to perform a role. As an example, in the unavailability of doctor, our mother can perform the role of a doctor by prescribing some medicines. Thus, our mother performs the role of a doctor even though the efficiency might probably be lesser than a professional doctor. Even though the systems assign roles autonomously, the assignment is based on global constraints.

In conventional distributed systems, the underlying theme behind building all distributed systems has been assigning specific roles to systems.An example to illustrate it is STORM. STORM, is a distributed real time computational platform used by Twitter .STORM [1] is primarily used for processing real time processing of stream data. Twitter uses STORM to compute trends and figures out the reach of its tweets. Reach of a tweet is a measure to determine the popularity of a tweet. In this platform, there is a central controller called Nimbus.

[1] https://github.com/nathanmarz/storm

Apart from this, there are two types of nodes called Zookeepers and Supervisors. The ZooKeeper nodes are responsible for cluster management while the Supervisor nodes are where the actual computations take place. Any node can act as either a Zookeeper or a Supervisor. Node responsibilities are assigned by the central controller, Nimbus. Distributed solutions are required in cases when roles are short lived and nodes need to take on different roles quite frequently, or when the distributed system is too large or has a lot of churn, making the central controller the bottleneck.

0.2 Related Literature

In this section, an attempt is made to compare CloudTop with the similar work. We have discussed ConPaas and Eucalyptus in detail in this section and tried to compare and contrast the work with CloudTop. Eucalyptus [1] is an open source project which provides a framework for building a cloud offering IaaS. The Eucalyptus system comprises the following components.

- Cloud Controller
  Cloud Controller (CLC) is the entry-point into the cloud for administrators, developers, project managers, and end-users. The CLC is responsible for querying the node managers for information about resources, making high-level scheduling decisions, and implementing them by making requests to cluster controllers.

- Cluster Controller
  Cluster controller runs on any machine which acts as the front end of a cluster. This machine should necessarily be connected to all the node controllers [discussed in next item] and the cloud controller. The main function of the cluster controller is monitoring and collecting information about all the nodes in the cluster and scheduling the execution of VMs on the nodes. In addition, the cluster controller is also responsible in the enforcement of SLAs directed by Cloud Controller.

- Node Controller
  A Node Controller (NC) executes on every node which hosts VM instances. NCs control the execution, inspection and termination of VM instances on the host where it runs, fetches and cleans up local copies of instance images and queries and controls the system software running on its node in response to queries and control requests from the cluster controller.

- Management Platform
  Management platform provides an interface to Eucalyptus services and modules.These features can include VM management, storage management user/group management, accounting, monitoring, SLA definition and enforcement, cloud-bursting, provisioning, etc.

However, our design differs from Eucalyptus and is more scalable compared to it.In Eucalyptus, nodes are assigned statically during start-up or during execution manually with a role viz, cloud controller ,cluster controller etc. that it shall play. In our approach,i.e. in CloudTop, any node would be able to play any role and depending upon various factors like load, requests, number

of running machines etc., these machines will orchestrate among themeslves to complete the request. It is worth noting that Eucalyptus also has a single point of failure or a bottleneck in the form of cloud controller. Since, in CloudTop any node can play any role, such a bottleneck is avoided.
In this case, scalability of Eucalyptus, technically refers to the ease of use of Eucalyptus for a system admin. In conventional terms,Eucalyptus can be scaled to 256 systems. However, it is to be realized that, assignning role to such a huge number of systems would not be a trivial task for the system admin. In the architecture that we propose, it would not be necessary to assign roles to systems individually and instead, the distribution of roles is specified and the servers are dynamically assigned to the respective roles.

Another architecture for building a cloud architecture is ConPaas[2].ConPaaS is an elastic architecture for building applications on the cloud. The block diagram for ConPaaS is shown in figure 1.

In the ConPaaS architecture, a collection of services are offered to the user.Thus, every application that needs to be executed on the ConPaaS system is a collection of services. Each service is self managed and elastic.The services offered by ConPaaS currently are: 1. PhP and JSP 2. MySQL and NoSQL. 3. MapReduce etc.

Each application is submitted to the ConPaaS system through a front end. The front end communicates with the service manager about the requirements of the application. The service manager is responsible for provisioning of resources. Each service has a manager and other agent VMs. The manager nodes do not execute any of the actual application code. They only perform initialization of new nodes, destroying VMs etc.. Every service has a number of agent or worker VMs. These VMs perform the actual computational logic.

The services in ConPaaS is similar to the roles in CloudTop. However, the elastic claim of ConPaaS is doubtful since every service requires a manager node. Another problem with ConPaaS is, it requires a few front end VMs for the jobs to be submitted. This requirement is avoided in CloudTop since any request can be directed to any node in the system. If the node does not perform the task, it would send the request to some other node which performs it.

0.3 Controllers of CloudTop

Different roles in CloudTop is represented as the controllers in it. CloudTop comprises of the following Controllers:

- Node
- Space
- Role
- Map

These terms are explained below:

- Node
  A node is the most basic element in CloudTop. Any server or computer which forms a part of infrastructure for the cloud is a node.All the nodes

are connected to each other through the internet backbone. In the functioning of CloudTop, the properties which determine the efficiency of a node in performing a certain task are RAM(R), CPU frequency(C) and disk space(D). Each node is assigned a node identification number(NId). $N_i : R \times C \times \times D \rightarrow NID$.

- Dimensions
  The dimensions are defined with respect to a node. Dimensions represent the properties of a node which are of interest to the designer of CloudTop. RAM, CPU frequency and disk space are the dimensions of a cloud.

- Space
  One of the essential requirements of a cloud system is the abstraction of the specific node on which a given user's task is performed. The space is the lowest level of abstraction for a user. A space is thus a logical collection of nodes which perform a role. Within a space, a particular node might perform a task independently or a subset of nodes within a space might perform a task by co-operating with each other. In CloudTop, we have defined space to be a collection of a nodes which are similar to each other in terms of their dimensions. Every space in a cloud has a maximum and a minimum number of nodes. One of the ways of partitioning the nodes into different spaces that was discussed during the semester was to partition a given space whenever the the number of nodes in a space exceeded the maximum limit. In this approach, if the maximum number of nodes in a space were $Node_{max}$ and minimum number of nodes were $Node_{min}$, then $Node_{max} \geq 2 * Node_{min}$. It is assumed that all the nodes within a subspace know the IP address of all the nodes in the space.

- Role
  The CloudTop system needs to provide a number of services to the user. Some of them include login,providing VMs to the users, allocating a space to an incoming node into the cloud etc..The roles are broad classifications of the different services that the system has to provide. ex. In the role login, the services include sign up,login, deactivate account etc. Some of the roles and their associated services are listed below. However, not all of them are proposed to be implemented in the initial version of CloudTop.

Roles and related services	
Login	signup deactivate-account login logout
Authorization	share-documents revoke-permissions request-permissions
VM management	create-VM pause-VM start-VM stop-VM checkpoint-VM restore-VM
Space	$assign_space(node)$ $create_space(cloud)$ $return_nodes_in_space$
Map	$create_map(cloud)$ $assign_roles(space)$
Network management	create-NATting assign-publicIP
Metering	getCostForSession viewRatesForOtherConfigurations

Cloud	checkIfSeedNode(Node)
	joinCloud(Node)
	resumeNode(Node)

- Map A node in the system might not perform all the possible roles in the system. However, it might receive a request to play a certain role which it currently is not allowed to. In such cases, the node should return the address of the nodes which are currently playing the desired role. It contacts the map file to find the nodes which are performing a given role. A map file provides a mapping between a role and the the spaces which perform the role.

  If role is represented by R_i and spaces by S_i, $R_i \rightarrow S_1, S_2, S_i$.

In the next section, the design of the CloudTop system is described.

0.4 Design and working of cloud

In CloudTop, every node can play any role in the system. However, depending on system characteristics, every node ends up playing a subset of all the roles. Thus, the roles played by a system could be divided into:

- Active Roles
- Inactive Roles

Active roles are the ones which are currently played by the system. Inactive roles are the those roles which can by played by the system but are currently not performed by the node. Every role in a CloudTop system is either active or inactive for a particular node. If a request for a role which is inactive in the current node is encountered, the node queries the map file and returns the address of the node for which the current role is active.

An example in this regard would make the working more clear. Consider a scenario where a user wants to create a new account for himself with the CloudTop system. The user has to call the $create_account$ service in the role *login*. The user can call the method through the web interface. This call would be directed to any of the nodes randomly in the CloudTop system. Two cases emerge in such a situation

- If the request is executed on a node which has the role in which the requested service is present is active.
- If the request is executed on a node whose role in which the requested service is present is inactive.

If the $create_account$ request is directed to any of the nodes in which the *login* role is active, then the request is either executed on the same node itself or the node contacts some other node within the same subspace to execute the request.

If the $create_account$ request was directed to a node in which *login* rule was inactive, then the node queries the *map* file to find the subspaces in which the given role *login* is active. These subspaces for which the role is active is returned to the user.

0.5 Initial prototype

The current implementation of the CloudTop comprises of the init module which performs the initialization of the system. Each role in CloudTop is implemented as a controller in CloudTop. Each node will have all the controllers present with it. However, all the controllers may not be active in every node. The following controllers are implemented (only to the extent of init function) in this version of CloudTop.

- Node Controller The node controller is used for implementing functions which are only related to the node. These functions are typically performed by a node during the bootstrap process before they join a cloud.
  - init() - init() is the first function called by CloudTop and is used to check if the current node is resuming after a brief shutdown from the cloud system or is entering the system afresh. This is done by checking for the existense of a file, “node.yml”.The file “node.yml” is created only when a node becomes a part of a cloud and the cloud system assigns the node a space id and the set of roles that it is to perform.If the node is already a part of the cloud, it has to carry on performing the roles that it was assigned prior to its shutdown. Else it calls the new function in the node controller.
  - - checkifseedNode() This function is used for checking if the given node is a seed node.The seed node has a different execution sequence compared to other nodes and is used only for the init sequence. This is necessary because for the first node, it has to create a space and

assign all the roles to itself. For the other nodes, they can contact a node in the system to assign itself a space id. Therefore, the first node in a cloud is called the seed node. To check if a node is the first node in the system, the administrator is expected to enter a seed node IP address. If the IP address of the given system matches the entered address, then it is a seed node and. If it is a seed node, then the init cloud() method in the cloud controller is called which essentially creates a space and assigns all the roles to itself. If the node is not a seed node, then the node calls the join cloud() method in the cloud controller of the seed node along with the necessary parameters like hard disk space, RAM and CPU speed.

- Cloud Controller - The cloud controller hosts functions which orchestrate the functioning of cloud by being the first point of contact for nodes which want to join a cloud. The initc loud() function in the cloud controller allows a node to start a cloud service by being the first node in the cloud. The cloud controller contains functions which pertain to setting up a new cloud, adding a new node to the system, destroying a node from the cloud etc.. Some of the important functions implemented in the cloud controller are:

  – init cloud() - The init cloud() function is called by the seed node to set up the cloud. It first creates a space by setting the maximum and minimum values for hard disk space, RAM space and CPU speed.

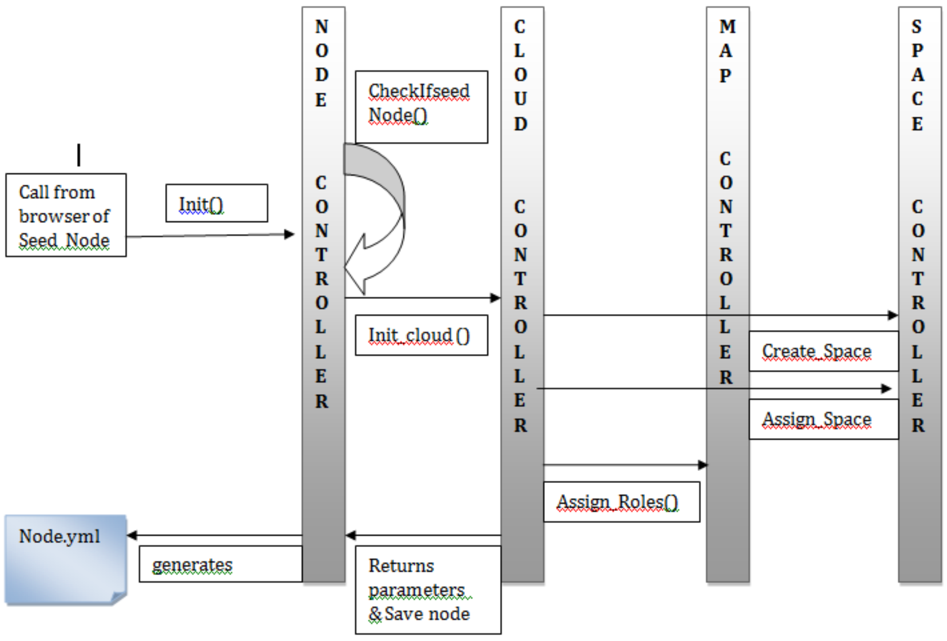

Figure 1: Init process of seed node

It also creates the possible roles by reading the entries from an administrator created file. It then assigns itself(the current node) to the space and assigns all the roles to itself. After the node is saved to the database, a file with the details of the node is generated "node.yml". Figure 1 depicts the initialization of seed node.

– join cloud() - The join cloud() function is called by nodes wishing to join the cloud. In an ideal scenario, this function can be executed in any active node in the system. However, since the database is not replicated to all the nodes in the system in the current implementation, this function will necessarily have to be called in the seed node.The join cloud() functionin the new node calls the 'getSpacefromNode' function in the cloud controller of the seed node along with passing the node's parameters to the function. From the seed node, the assign space method is called and new space id is generated. The seed node then returns this id to the new node by calling the method 'recieveSpacefromNode' in the new node. After the space id is received, the new node is saved with the new parameters by the save node() function in the new node. As in the After saving the node a file with the details of the "node.yml". Figure 2 shows the init process of a node other than a seed node.

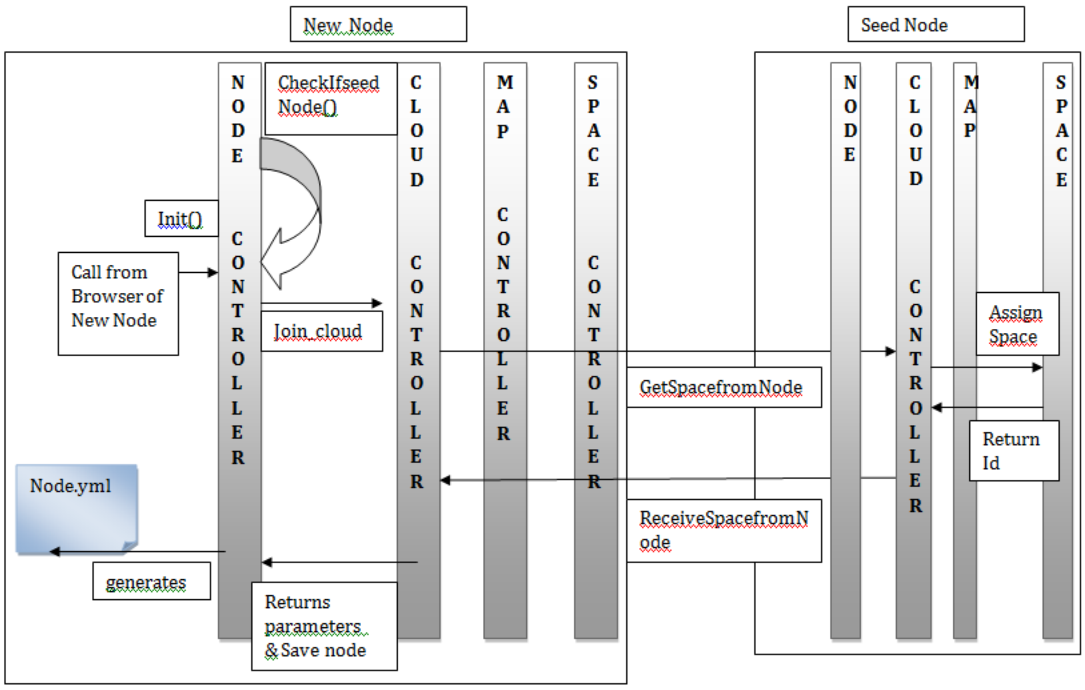

Figure 2: Init process of a node other than seed node

- Map controller The map is used to maintain a mapping between a space and the roles played by each of them.The functions pertaining to assigning roles to a space, changing of roles when a new node enters the system is performed by the map controller. Although the role assignment engine has to be coded in this controller, currently all the roles are assigned to all the nodes, since only the init() process is underway. The function which performs the same is assign roles().

- Space Controller The space is the lowest abstraction in CloudTop to perform roles/tasks. Creation of spaces, changing assignment of nodes to spaces, assigning a space id to a new node in the cloud are the functions performed by the space controller. Some of the functions in space controller are:

  - assign space() The node object is passed as a parameter to this method and a space ID is returned when the function is executed in it. In this function, a more sophisticated algorithm to assign nodes to a space and policies to determine when a space has to split has to be determined. In the current version, a node is assigned to a space if the parameters of the given node lie between the minimum and maximum value for the same parameter.
  - create space() In the current implementation, the create space would work only for the first space created, since only the init functionality has been the objective. Creating a space essentially entails setting the maximum and minimum values for hard disk space, speed of the CPU and the RAM.

Bibliography

[1]] D.Nurmi, R.Wolski, C.Grzegorczyk, G.Obertelli, S. Soman, L.Youseff, and D.Zagorodnov *The Eucalyptus Open-source Cloud Computing System.* Proc. 9 th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2009), Shanghai, China: May 2009, pp. 124–131.

[2]] G Pierre, C Stratan ,Ana Oprescu,Thilo Kielmann *TConPaaS: an integrated runtime environment for elastic cloud applications.* · Proceeding PDT '11 Proceedings of the Workshop on Posters and Demos Track Article No. 5