Towards a Hybrid Cloud Platform Using Apache Mesos

Noha Xue, Hårek Haugerud(✉), and Anis Yazidi

Department of Computer Science,
Oslo and Akershus University College of Applied Sciences, Oslo, Norway
harek.haugerud@hioa.no

Abstract. Hybrid cloud technology is becoming increasingly popular as it merges private and public clouds to bring the best of two worlds together. However, due to the heterogeneous cloud installation, facilitating a hybrid cloud setup is not simple. Despite the availability of some commercial solutions to build a hybrid cloud, an open source implementation is still unavailable. In this paper, we try to bridge the gap by providing an open source implementation by leveraging the power of Apache Mesos. We build a hybrid cloud on the top of multiple cloud platforms, private and public.

Keywords: Opensource hybrid cloud · Apache Mesos · Data segmentation · Fault tolerance

1 Introduction

The use of cloud computing is becoming more common, bringing along the advantages of flexibility and abundance of available resources, but also a higher degree of complexity along with privacy and security concerns. The concepts of *multi-cloud* and *hybrid cloud* are not new and several companies explore and capitalize these concepts. Most of the available solutions are commercial. Different vendors including Dell, IBM and HP provide hybrids cloud solutions [1,2]. The *MODA-Clouds* project [3] utilizes several tools to provide an environment for utilizing multiple cloud providers. Several large companies are offering hybrid cloud solutions, often in conjunction with their existing product portfolio. VMWare is offering a hybrid cloud solution called *vRealize suite* which provides one interface to manage the entire hybrid cloud platform [4,5]. Other companies like *Cisco* [6], *IBM* [7] and *RackSpace* [8] are also offering hybrid cloud solutions. Another attempt addresses the challenges of managing heterogeneous virtual environments to create a hybrid cloud platform [9]. *PaaSage* is an interesting initiative for building a hybrid cloud solution using a defined deployment model, *Cloud Application Modeling and Execution Language* (CAMEL) [10]. However, there has been no practical demonstration of using open-source and freely available clustering technology to attempt to address the multitude of challenges

D. Tuncer et al. (Eds.): AIMS 2017, LNCS 10356, pp. 143–148, 2017.
DOI: 10.1007/978-3-319-60774-0_12

when creating a hybrid cloud platform that is available and supports data segmentation. This paper outlines an attempt to prototype such a solution in addition to facilitate cloud bursting using spot price instances. Borja et al. introduced OpenNebula in [11], which is one of the most popular open source Virtual Infrastructure Manager (VIM). OpenNebula permits to abstract resources of an existing local Grid and a cloud infrastructure. At the heart of OpenNebula we find Haizea [12] which is a VM-based Lease Manager that enhances the resource scheduling manager with advanced reservation of resources and queueing of best effort requests. Nevertheless, OpenNebula suffers from the single point failure problem [13]. In this paper, we present a lightweight solution, that is tolerant to different failure scenarios. Similarly, it is also possible to create a Hybrid Cloud With AWS and Eucalyptus.

This paper will explore and document the attempts at designing and prototyping a possible opensource solution for constructing a computer cluster built on top of private servers and external cloud providers.

2 Design and Implementation

An Apache Mesos cluster including both master nodes and slaves nodes were successfully installed and configured in Altocloud, with slave nodes correctly registering themselves to the cluster through the leading master node. However, when attempting to register a slave node running at Amazon Web Services EC2 peculiar activity was observed. The traffic from the slave node located at EC2 managed to successfully send a registration request to the leading master node, passing through multiple layers of network abstraction including two layers of NAT. Although the master node receives the registration requests, no registration acknowledge is ever sent back. Eventually, the cause was discovered to be a combination of the use of NAT and the way Mesos nodes communicates between each other. When a slave node sends a registration request, it includes information about the resources available and an IP-address. The IP-address sent along is the one that is defined on the network interface bound by the Apache Mesos process. Furthermore, in a cloud environment like Altocloud and Amazon Web Services EC2, the public IP-addresses are loosely coupled with the virtual machine and functions similarly as NAT does. Consequently, the Mesos master attempts to send the acknowledgement and other internal traffic meant for that slave node to the non-routable private IP-address. The communication flow is illustrated in Fig. 1.

By using VPN tunneling, the need for allocating public IP-addresses for each node disappears for the purpose of maintaining the cluster, as the private IP-addresses becomes routable within the hybrid cloud platform. With the exception of the extra infrastructure to maintain a VPN, the prototype is identical to the proposed proposed design. Figure 2 illustrates the final implementation of the prototype, showing how the Mesos master nodes are distributed between the different availability regions.

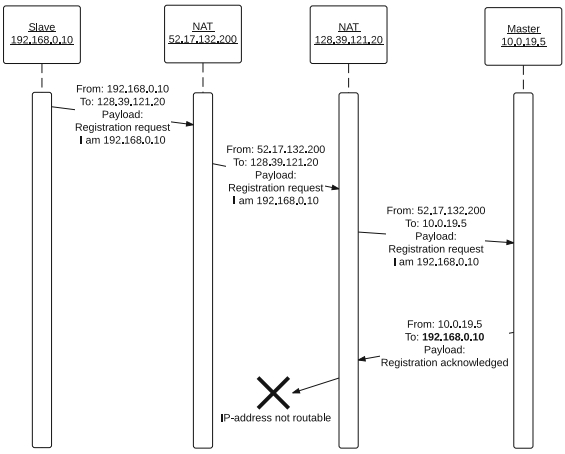

Fig. 1. Communication flow between an Apache Mesos slave node and master node with the registration attempt failing due to how public IP-addresses are handled in cloud platforms.

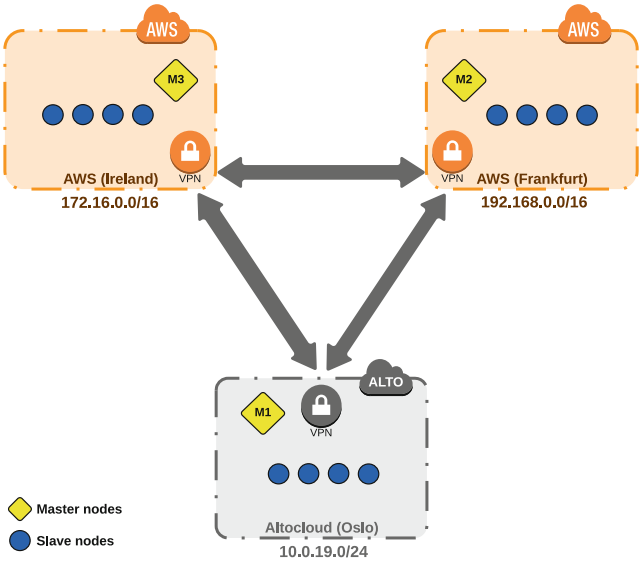

Fig. 2. Prototype 1: Maximizing availability. Distributing the master nodes and thereby the risks.

Test Scenarios

A Mesos Slave Process Becomes Unavailable. In the event of a Mesos slave node becoming unavailable for some reason, the Mesos master node allows a default timeout period of 75 s to pass before procedures for deactivating the slave node

is begun. Should the slave node start responding within this timeout period, nothing will happen and both the Mesos master node and the slave node simply ignores the temporary unavailability.

However, if the timeout period is exceeded and the slave nodes is still unavailable, the Mesos master node will attempt to deactivate the Mesos slave process on the slave node before it from the list of available slave nodes. Tasks that were lost will be rescheduled to other slave nodes with available capacity.

Should a slave node simply be temporarily disconnected from the master node, but exceed the timeout period, the Mesos master will forcibly shut the Mesos slave node down. To account for such scenarios, the official Apache Mesos documentation recommends monitoring the Mesos slave process and restart if it should be terminated for any reason. In this case, this is achieved with a simple check using Monit. In Listing 1.1 log events of such a case is listed.

The Working Mesos Master Instance Cease to Function. ZooKeeper maintains an active connection to the participants of the quorum and will after a very short timeout lasting a few seconds, will initiate a new leader electing for choosing a new leading Mesos master node. As long as the number of functional Mesos master nodes is equal or higher than the quorum size, a new leader will be elected and will replace the unresponsive Mesos master node.

```
1 17:34:23.298998 Shutting down slave ...5050-5669-S3 due to health check timeout
2 17:34:23.300134 Removing slave ...5050-5669-S3 at slave(1)@192.168.187.205:5051
  ↪ (192.168.187.205)
3 17:34:23.301009 Removed slave 20150501-230056-2407081856-5050-5669-S3
4 17:34:23.536837 Notifying framework ...5050-27030-0006 (marathon) at
  ↪ ...473b-b57a-83121a00a01c@128.39.121.140:43217 of lost slave ...5050-5669-S3
  ↪ (192.168.187.205) after recovering
5 17:34:29.017205 Slave ...5050-5669-S3 at slave(1)@192.168.187.205:5051 (192.168.187.205)
  ↪ attempted to re-register after removal; shutting it down
6 17:34:57.329751 Registering slave at slave(1)@192.168.187.205:5051 (192.168.187 .205) with
  ↪ id ...5050-5669-S4
```

Listing 1.1: Excerpt from `/var/log/mesos/mesos-master.INFO` showing the forced shut down of the Mesos slave process at 192.168.187.205 and the registration as new slave at end. Truncated for increased readability.

This scenario was tested with a simple reboot of the instance where the leading Mesos master was running. The backup Mesos masters quickly discovers the loss of connection to the leading Mesos master and promptly, with the use of ZooKeeper elects a new leading Mesos master node. The rebooted Mesos master node later joins the cluster as a backup node after coming back online.

The setup proposed in this prototype has three Mesos master nodes, with the quorum size set to two. This means that among the Mesos master nodes, one can fail without crippling the cluster, as the quorum size dictates the number of election participants that has to be able to communicate to be able to elect a new leader.

An Entire Region within the Hybrid Cloud Becomes Unavailable. If an entire region becomes unavailable, the Mesos nodes located within those regions will by extension also become unavailable. In this particular case, the loss of one single site equals the loss of one Mesos master node and four slave nodes. Each node, depending on the type, is handled as specified in the test scenarios mentioned above.

This was tested by taking down the VPN tunnels at the VPN gateway of the concerned region. This cuts all communication between the affected region and the other ones. As expected the Mesos master nodes continued without any issues, as the current leader was not the affected one. As for the affected Mesos slave nodes, after the timeout of 75 s, the leading Mesos master node determined that the slave nodes were unresponsive deactivated them.

The Hybrid Cloud Splits and Semi-isolates Part of the Platform. In the event of split in the hybrid cloud, resulting in a partly isolated availability region, the quorum mechanics will prevent inconsistencies of the cluster and avoid issues like the split-brain problem.

To test this scenario, two simple `iptables` DROP rules was added on the Mesos master node located in Frankfurt with the IP address `192.168.0.5`.

```
iptables -A INPUT -s 10.0.19.5 -j DROP
iptables -A OUTPUT -d 10.0.19.5 -j DROP
```

The leading Mesos master node at the current time was `10.0.19.5`, with nothing occurring immediately as a result of the `iptables` DROP rules. The leading master continued with no issues and other two standby Mesos masters correctly redirected to the leading master node. However, after rebooting the ZooKeeper process and Mesos master process on the master nodes, the cluster is unable to elect a new leader. Immediately after the `iptables` DROP rules were removed, a new leading Mesos master was elected and operations continued as normal.

3 Conclusion

This paper presents a prototype of a hybrid cloud platform using Apache Mesos to weave together heterogeneous clouds and geographical locations into a unified platform. The prototype proposed focuses on a specific perspective, namely, maximizing availability.

References

1. Connor, T.R., Southgate, J.: Automated cloud brokerage based upon continuous real-time benchmarking. In: 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), pp. 372–375. IEEE (2015)

2. Breiter, G., Naik, V.K.: A framework for controlling and managing hybrid cloud service integration. In: 2013 IEEE International Conference on Cloud Engineering (IC2E), pp. 217–224. IEEE (2013)
3. MODAClouds. Modaclouds. http://www.modaclouds.eu
4. VMWare, Inc. vrealize suite. http://www.vmware.com/products/vrealize-suite/features.html
5. VMWare, Inc., Cloud Computing. http://www.vmware.com/cloud-computing/hybrid-cloud.html
6. Butler, B.: Re-examining ciscos intercloud strategy, January 2015. http://www.networkworld.com/article/2864857/cloud-computing/re-examining-cisco-s-intercloud-strategy.html
7. IBM. Private and hybrid cloud. http://www.ibm.com/cloud-computing/uk/en/private-cloud.html
8. Rackspace, Inc., Hybrid cloud computing, hybrid hosting by rackspace. http://www.rackspace.com/cloud/hybrid
9. Breiter, G., Naik, V.: A framework for controlling and managing hybrid cloud service integration. In: 2013 IEEE International Conference on Cloud Engineering (IC2E), pp. 217–224, March 2013
10. PaaSage. Paasage: Model-based cloud platform upperware. http://www.paasage.eu
11. Borja, S., Ruben, M., Ignacio, M., Ian, F.: An open source solution for virtual infrastructure management in private and hybrid clouds. IEEE Internet Comput. **1**, 14–22 (2009)
12. Kovács, Á., Lencse, G.: Modelling of virtualized servers. In: 2015 38th International Conference on Telecommunications and Signal Processing (TSP), pp. 241–245. IEEE (2015)
13. Feller, E., Rilling, L., Morin, C.: Snooze: a scalable and autonomic virtual machine management framework for private clouds. In: 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID 2012), pp. 482–489, May 2012