Making the Cloud Efficient: In-depth study of cloud computing and Resource Management

Vijay Pasikanti, Sneha Shankar Narayan

May 5, 2014

CS677 - Operating Systems

Faculty: Prashant Shenoy

Contents

Chapter 1

Introduction

Cloud computing platforms are complex systems with many VMs which in turn run different applications. Since applications have varying needs, different VMs are configured differently. The goal of this project is to write a simple tool that is inspired by Amazon Trusted Advisor that is used in Amazon's EC2 cloud to continuously monitor a cloud deployment and make recommendations on how to improve security, resource usage, etc for various VMs by analyzing the hypervisor. The project involves setting up OpenStack on two nodes, setting up hypervisors and virtual machines and then writing a tool that uses the OpenStack API as well as the hypervisor API to gather various statistics that are then analyzed to provide recommendations.

In this project we focus on monitoring the statistics based on which we provide resource utilization recommendations.The hypervisor that is being used here is KVM.

Chapter 2

Openstack setup

Openstack is a cloud management platform where in there is a controller node and a compute node.

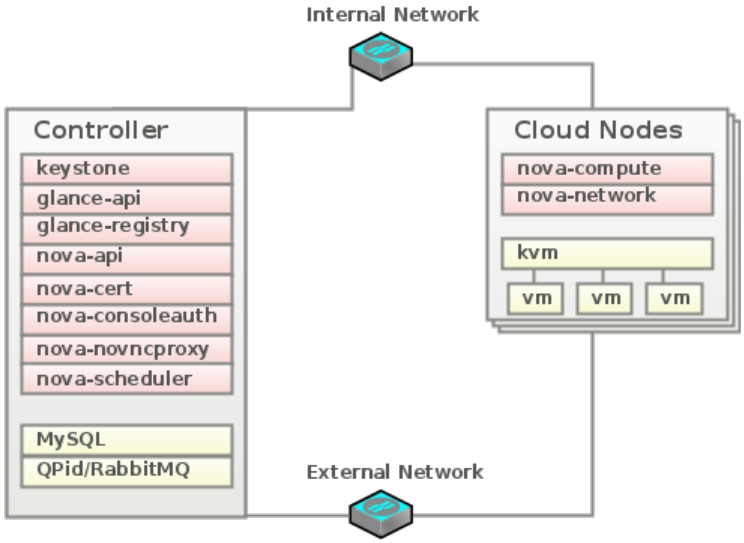
- The controller node runs the Identity Service, Image Service, dashboard, and management portion of Compute. It also contains the associated API services, MySQL databases, and messaging system.
- The compute node runs the hypervisor portion of Compute, which operates tenant virtual machines. By default, Compute uses KVM as the hypervisor. Compute also provisions and operates tenant networks and implements security groups. You can run more than one compute node.

We have one controller node and one compute node and the following open stack services have been installed

- **Identity service (Keystone):** Keystone is responsible for user management and service catalog.
- **Image service (Glance):** Glance is responsible for image discovery, retrieval, and storage, storing and retrieving metadata about images and for managing a storage repository for images.
- **Compute service (Nova):** Nova is a cloud computing fabric controller. It is used to host and manage cloud computing systems. Compute interacts with the Identity Service for authentication, Image Service for images, and the Dashboard for the user and administrative interface. Access to images is limited by project and by user; quotas are limited per project (for example, the number of instances). The Compute service scales horizontally on standard hardware, and downloads images to launch instances as required.

The messaging between the various open stack services is done using RabbitMQ.

Our open stack architecture is configured the following way:



After configuring the controller and the compute node a network was setup between them. We have launched various instances of Ubuntu on our compute node. Currently we having the following VMs running in our compute node

```
root@obelix24:~# nova list
+--------------------------------------+----------+--------+------------+-------------+-----------------+
| ID                                   | Name     | Status | Task State | Power State | Networks        |
+--------------------------------------+----------+--------+------------+-------------+-----------------+
| d7cc815a-e158-4253-b4e4-65b27615fb86 | ubuntu11 | ACTIVE | None       | Running     | vmnet=10.0.0.2  |
| d53d43ba-27e0-4b45-afe0-ca566704953a | ubuntu12 | ACTIVE | None       | Running     | vmnet=10.0.0.3  |
| 0c53289c-a089-4f89-96f1-efff70fcbd9a | ubuntu13 | ACTIVE | None       | Running     | vmnet=10.0.0.4  |
| 1a1678fb-8daa-4647-89a7-414312be46d5 | ubuntu14 | ACTIVE | None       | Running     | vmnet=10.0.0.5  |
| 33251435-11f3-4013-ad30-4b4c94d99188 | ubuntu15 | ACTIVE | None       | Running     | vmnet=10.0.0.6  |
| 2b108650-c32e-4695-ae04-00622e61d8f4 | ubuntu16 | ACTIVE | None       | Running     | vmnet=10.0.0.7  |
| 56e157c6-a06f-4a6b-b152-b58664ad0c5e | ubuntu17 | ACTIVE | None       | Running     | vmnet=10.0.0.8  |
| eb54b3e2-5c5d-4c7b-a1ec-297fa96c53e9 | ubuntu18 | ACTIVE | None       | Running     | vmnet=10.0.0.9  |
| 64311806-d774-4f3a-b2bb-aacf4e6f9360 | ubuntu19 | ACTIVE | None       | Running     | vmnet=10.0.0.10 |
+--------------------------------------+----------+--------+------------+-------------+-----------------+
```

The instances are managed by the nova compute service.

Chapter 3

Existing monitoring systems explored

- V

Chapter 4

Openstack monitor

The Openstack monitor that we have implemented is written completely in python, it has various modules to collect the statistics from the different virtual machines, store the statistics on disk, retrieve the stored data and provide recommendations on a web interface.

4.1 Performance metrics

The performance metrics that we have decided to use to provide resource utilization specific metrics are:

- Average CPU utilization
- Average memory utilization
- Average disk utilization
- Average network utilization
- Average disk input/output

Our aim was to analyze the above metrics and warn the user when say, the CPU utilization becomes greater than 90 % which in turn means he needs to move the VM elsewhere or he can increase the resource allocated to the VM.

4.2 Design and Implementation

On a high level there is daemon which runs every five seconds to query the statistics from the hypervisor and this data is stored in Round Robin databases. There is another daemon which reads the data from the round robin files, provides recommendations and stores it as JSON files which is shown on the web interface which is updated every minute.

4.2.1 Collection of statistics

The various statistics that are required by our system to provide recommendations was collected using libvirt APIs. The following libvirt modules were used

- **virt-top:** virt-top is similar to the unix functionality 'top' which provides data regarding the CPU, network and memory utilization of the processes in the system. Virt-top however looks at hypervisor level information and provides data that is specific to the virtual instances in the machine. The output of virt-top was streamed and statistics about the CPU, memory, disk-IO and network utilization of the VMs was collected.
- **virt-df:** Virt-df is similar to the unix functionality 'df' which lists the disks that are mounted and the used and available space in a disk. Virt-df looks into the various virtual machines and reports the storage space that is in use and that is available. Thus, disk utilization was collected.

A daemon runs every five seconds and captures the data from the above libvirt modules and stores the data in Round robin database files.

4.2.2 Storing in RRD files

The main motivation behind using Round robin databases was because the RRDtool can manage time-series data effectively, won't grow infinitely in size because of the round robin nature of the database and was also easily integrable with python. RRDtool can be configured to calculate the rate of change from the previous to the current value and store this information instead.

Using the python API of RRD tool the daemon that collects the various statistics also updates the RRD files associated with each of the metrics collected of each instance.

Samples of the metrics collected have been stored for the following time intervals

- 5 second averages for 1 day
- 30min averages for 5 days
- 1 hour averages for 10 days
- 2 hour averages for 20 days
- 12 hour averages for 20 days
- 12 hour averages for 20 days
- 1 week averages for 10 weeks

Also we store average maximum and minumum values for memory utilization as well for the following time intervals

- 5 second averages for 1 day
- 1 hour averages for 5 days
- 1 day averages for 10 days

4.2.3 Processing statistics and giving recommendations

Another daemon has been configured to run indefinitely and it reads the statistics from the RRD files, processes them and provides recommendations. Recommendations are given based on the thresholds defined and we have defined the following thresholds.

- Average CPU utilization
  - High: 80%, to recommend user to upgrade hardware
  - Low: 20%, to alert user about resource under utilization
- Average memory utilization
  - High: 90%, to recommend user to upgrade hardware
  - Low: 20%, to alert user about resource under utilization
- Average disk utilization
  - High: 75%, to recommend user to add more disks
- Average network utilization
  - High: 10000 bytes transferred, to recommend user to increase bandwidth allocation
  - Low: 10 bytes transferred, to recommend user to downgrade bandwidth for instance
- Average disk input/output
  - High: 10000 IOps done, Alert user about high I/O
  - Low: 50 IOps done, alert user about resource under utilization

Using the values stored in the RRD files if it is found that any of these thresholds have been hit in the past hour that fact is recorded. This daemon spits out such recommendations to JSON files which is in turn fed into the web interface when it gets updated.

4.2.4 Web interface

A simple python HTTP server is currently running in the compute node of the openstack setup and by using SSH tunneling we can look at the results on any browser. The webpage shows the recommendations provided by the Openstack monitor. The JSON files dumped by the daemon that processes the statistics is read using AJAX and the webpage is dynamically updated every minute with the latest data. The critical alerts are highlighted in red, low utilization and shown in yellow such that a user who uses the monitoring system can take an appropriate action with respect to the configuration of the virtual machines.

Chapter 5

Testing

- V

Chapter 6

Results

-V

6.0.5 Screenshots

Chapter 7

Conclusion

We were able to read the various statistics like CPU utilization, memory utilization, network bandwidth utilization, and disk utilization of the virtual machines in the openstack compute node by querying the KVM hypervisor using libvirt APIs. We were able to process the data collected and provide recommendations to the user/open stack administrator to take appropriate actions regarding the virtual machine configuration.

This monitoring system can be extended to record various other statistics of the virtual machines in order to build a full fledged open source equivalent of the Amazon Trusted Advisor.

Chapter 8

References

8.1 Openstack

1. http://docs.openstack.org/trunk/install-guide/install/apt/content/

8.2 RRDTool

1. http://oss.oetiker.ch/rrdtool/index.en.html
2. http://oss.oetiker.ch/rrdtool/doc/index.en.html
3. http://oss.oetiker.ch/rrdtool/tut/rrdtutorial.en.html

8.3 Monitoring systems explored

1. http://collectd.org/
2. http://ganglia.sourceforge.net/
3. http://www.nagios.org/
4. http://sensuapp.org/

8.4 libvirt

1. http://libvirt.org/
2. http://libvirt.org/python.html
3. http://libvirt.org/virshcmdref.html
4. http://people.redhat.com/~rjones/virt-top/
5. http://libguestfs.org/virt-df.1.html