Clusters in the Cloud

Shunde Zhang[1], Paul Coddington[2]

1 eResearch SA, Adelaide, Australia, shunde.zhang@ersa.edu.au
2 eResearch SA, Adelaide, Australia, paul.coddington@ersa.edu.au

INTRODUCTION

Researchers who require a lot of compute power are good candidates to take advantage of the large, elastic compute resources provided by the Australian Research Cloud. Many of these researchers currently use shared or dedicated compute clusters and submit jobs to these resources using job management (or queueing) systems such as Torque, PBS, Sun Grid Engine (SGE) and Condor. An easy way for these researchers to take advantage of cloud resources is to provide a compute cluster that is made up of virtual machines in the cloud and enables jobs to be submitted to the cluster using a familiar queueing system.

For over a year we have been exploring, developing and testing different approaches to running compute clusters in the cloud, and have implemented some solutions that are now in production use by several research groups.

CLUSTERS FOR HIGH-THROUGHPUT COMPUTING

We initially worked with the Centre of Excellence in Particle Physics at the Terascale (CoEPP) to develop high-throughput computing (HTC) clusters to process data collected by the ATLAS experiment at the Large Hadron Collider at CERN as part of the worldwide LHC Computing Grid. We set up a dynamic cluster using Condor as the job management system and Cloud Scheduler [1], which works alongside Condor and dynamically starts up new cloud VMs and adds them to the Condor pool, and then shuts them down when they are idle.

For other types of jobs, researchers preferred to use the familiar Torque queuing system. We therefore developed an alternative approach by setting up a standard, static Torque cluster using cloud VMs. However this had some limitations, in particular it could not elastically provision compute resources based on the number of jobs in the queue, and could not easily be set up by a research group without assistance from experienced systems administrators.

STARCLUSTER

StarCluster [2] is an open source cluster toolkit developed by MIT for Amazon's EC2 cloud. It has been designed to automate and simplify the process of building, configuring and managing clusters of virtual machines for various purposes by using different plugins. It provides plugins for resource management systems for computing clusters, such as SGE and Condor, as well as plugins for Hadoop and MySQL clusters. StarCluster builds a series VMs and configures them as a cluster, including one head node and a number of worker nodes. The head node runs a shared file system for all other worker nodes to access. The chosen plugin installs the corresponding resource management system on all these VMs and make it ready for the user to use. The user only needs to log in to the head node to submit batch jobs, which will be distributed to all other worker nodes for execution. It enables users to create private clusters in the cloud.

However, because StarCluster was designed for EC2, to get it to work in the NeCTAR cloud we had to make several modifications [3] to avoid the use of EC2 features that are not supported by OpenStack. We also developed a plugin for Torque (used on eRSA clusters and many others in Australia) since StarCluster only supported SGE. Using StarCluster, research groups can set up their own private cluster in the cloud. However installing and running StarCluster is challenging for users with limited Unix skills. We therefore set up a customized instance of StarCluster at eResearch SA (eRSA), so users did not have to install the software and could use it relatively easily to set up a private cluster, and log in using their eRSA account. We also created a VM image that gave access to all the software applications installed on other eRSA clusters using CVMFS [4], which allows efficient read-only access to remote file systems over http. StarCluster has been successfully used by several research groups to set up private clusters in the Australian Research Cloud, some with over 100 cores.

DYNAMIC TORQUE

A disadvantage of StarCluster and the static Torque cluster is that compute nodes could only be manually added or removed from the cluster. We therefore developed a system called Dynamic Torque [5] that can dynamically scale up and down the number of worker nodes in the cloud according to the number of jobs in the Torque queue. The system is shown in Figure 1.

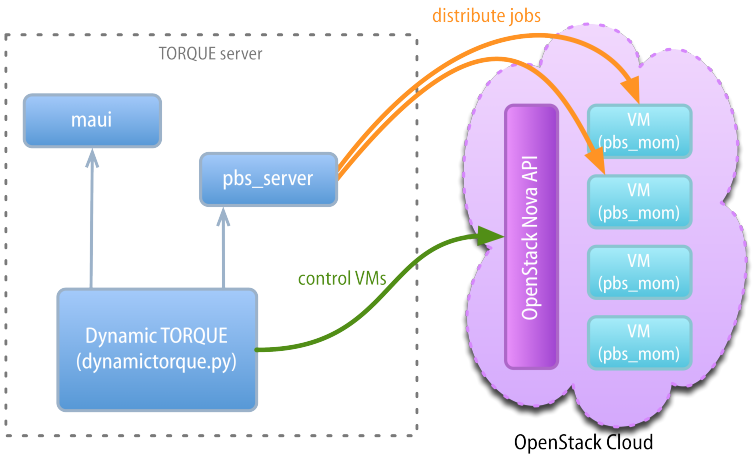

Figure 1: Dynamic Torque

Dynamic Torque works by actively querying the job queue at fixed intervals. If all existing worker nodes are busy but there are still jobs waiting, it will launch new VMs in the cloud based on the job priorities gathered from the Maui job scheduler and the resource requirements of these jobs. Dynamic Torque then queries the states of the newly-launched VMs to determine if they are ready or not. Typically it takes around 5 minutes for the OpenStack cloud to provision a VM and extra time to execute a server customization script. When a VM is ready Dynamic Torque adds it to the Torque server as a normal worker node. Torque/Maui can then distribute queued jobs to the new worker node.

As VMs are added to Torque as normal worker nodes, they need to be set up like other physical worker nodes. Firstly, a PBS MOM process needs to run on the VM in order to communicate with the Torque server. Secondly, the VM needs to know all user information of the system; generally it should be configured with a central LDAP server. Lastly, the VM needs to be able to access a shared file system, so the jobs run on this VM can see the input data and write output data. These can be built into the image that is used to launch the VM, or done in the VM customization step.

CoEPP's static Torque cluster was replaced with Dynamic Torque in mid-2013, and it has worked well, across multiple availability zones, dynamically changing the size of the cluster between 20 and 200 cores based on the number of jobs submitted. eRSA has recently set up a shared cluster in the cloud on the SA cloud node using Dynamic Torque. eRSA users can run jobs on this cluster in a very similar way to the physical clusters managed by eRSA. We have also developed a Dynamic Torque plugin for StarCluster to allow VMs to be created and shut down dynamically based on load rather than manually.

SHARED CLUSTER WITH MULTIPLE TENANCIES

Recently we have added the capability for Dynamic Torque to allow users to use the resource allocation of their own tenancy to launch new worker nodes. These worker nodes are exclusive to users of that tenancy only, so users effectively get a private cluster (with their own VM image if required), but without having to set up and manage their own cluster using StarCluster or Dynamic Torque, which can be complicated, particularly if there are problems with Torque or OpenStack, which can be difficult for users to debug and fix.

This is done via the account string property in Torque and MAUI. When a user submits a new job, an account string will be put into the job script to specify the tenancy to run this job. Then a new worker node is fired up using that tenant's quota and reserved in MAUI using the same account string. Thus MAUI can match the job with that account string to this worker node. If all worker nodes of this tenant are busy, some shared resources (worker nodes without reservation, contributed from an eRSA tenancy) will be used to run the job; this is a default MAUI behaviour.

REFERENCES

1. Cloud Scheduler, http://cloudscheduler.org/
2. Star Cluster, http://star.mit.edu/cluster/
3. StarCluster for OpenStack, https://github.com/shundezhang/StarCluster/
4. CVMFS, http://cernvm.cern.ch/portal/filesystem
5. Dynamic Torque, https://github.com/shundezhang/dynamictorque.git
6. ViBatch, https://ekptrac.physik.uni-karlsruhe.de/trac/BatchVirt/wiki/ViBatch/

ABOUT THE AUTHORS

Shunde Zhang is a systems and software developer at eResearch SA, who is working on eResearch SA's migrating applications to the cloud program and previously on a NeCTAR eResearch Tools project to do high-throughput computing on the NeCTAR cloud. He has extensive experience on grid computing and distributed data storage as well as Java/Python development.

Paul Coddington is Deputy Director of eResearch SA, where he has managed eResearch projects, infrastructure and user support since 2002. He has a background in high-performance computing, distributed computing, computational science and research data management.