Integrating grid and cloud resources at the RAL Tier-1

Andrew Lahiff, Alex Dibbo, George Ryall, Frazer Barnsley, Ian Collier
STFC Rutherford Appleton Laboratory, Harwell Oxford, UK

Introduction

- Grid submission to traditional batch systems remains by far the primary method of running work at WLCG sites
- The ability to use virtualised worker nodes running on a cloud in a traditional batch system is potentially very useful, as it allows a site to:
  1. Provide both cloud and grid computing resources without partitioning
  2. Make use of a local private cloud when there are idle jobs in the batch system and there are free resources in the cloud
- There are two aspects to this for the situation where the cloud is for opportunistic use only:
  - Expanding the batch system into the cloud when the cloud has free resources
  - Reducing the amount of cloud resources used in the batch system when the cloud becomes busy
- Here we present work carried out at the RAL Tier-1 where we investigated including resources from our OpenNebula cloud into our HTCondor batch system

SCD Cloud

- Initial use case is to provide a self-service portal for members of the Scientific Computing Department to obtain VMs for development work
  - Eventually expect to be able offer access to the LHC and other experiments via cloud APIs
- OpenNebula based cloud with a Ceph storage backend
- 28 hypervisors consisting of 892 cores and 3.4 TB RAM
- 750 TB raw storage, 10 Gb/s networking
- Headnode and Galera MariaDB database cluster are on VMs in Hyper-V production virtualisation infrastructure

Monitoring

- Virtualised worker nodes have standard Ganglia monitoring
- Historically our bare metal worker nodes have always had Nagios monitoring
  - Nagios doesn't handle dynamic resources well
  - Decided not to use Nagios at all on virtualised worker nodes
- All worker nodes (virtualised or bare metal) should only run jobs if they are healthy
  - A health-check script runs on each worker node as HTCondor startd cron
    - Checks for read-only or problematic disks, CVMFS, …
  - START expression configured so that new jobs will only start if node is healthy

Security and traceability

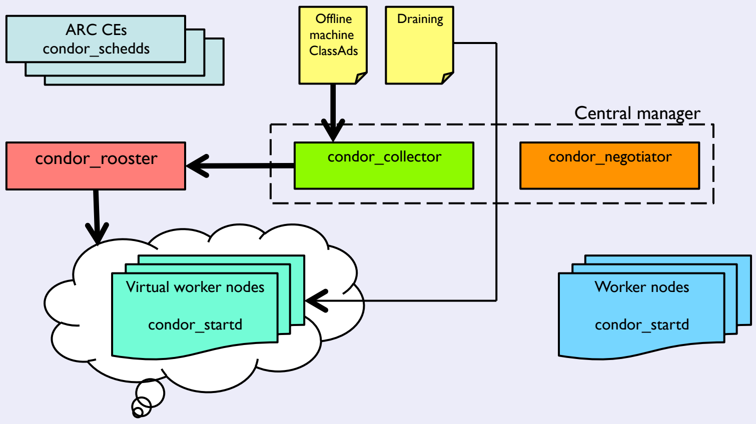
- Quarantining of disk images
  - Snapshots of images are kept for short periods of time in order to allow us to investigate potential user abuse of short-lived VMs
  - At VM instantiation, an OpenNebula hook creates a deferred shap-shot to be executed when the machine is shutdown
  - A cron job runs daily to delete any images older than a specified age
- Worker node image configured to log to our central loggers using syslog
- Open Nebula VM IDs are made available in job ClassAds so that we can easily find out what VM a job ran on
  - HTCondor in the worker node image is configured to advertise the unique VM ID
  - Schedds are configured to insert this ID into job ClassAds
  - Independently of this, log files can be used to easily determine what VM a particular job ran on

Batch system at the RAL Tier-1

- The RAL batch system consists of 560 worker nodes and over 12000 cores
- During 2013 we migrated from Torque/Maui to HCondor due to increased reliability, scalability, flexibility and ability to handle dynamic resources
  - One significant advantage of HTCondor over alternatives is that it was designed to make use of opportunistic resources (e.g. idle desktops)
  - This makes HTCondor perfectly suited for dynamic environments where the amount of resources available is constantly changing (e.g. opportunistic expansion into a cloud)

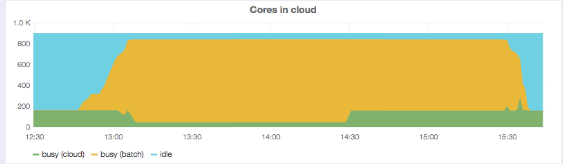
Integrating virtualised worker nodes

- Based on existing power management features of HTCondor
- **Virtual machine instantiation**
  - ClassAds for offline machines are sent to the collector when there are free resources in the cloud
  - Negotiator can match idle jobs to the offline machines
  - Rooster daemon detects these matches and triggers the creation of VMs
- **Virtual machine lifetime**
  - Managed by HTCondor on the VM itself; configured to:
    - Only start jobs when the worker node health-check script is successful
    - Only start new jobs for a specified time period
    - Shuts the machine down after being idle for a specified time period
  - Virtual worker nodes are drained when resources on the cloud become scarce
    - Once machines have drained the resources are returned to the cloud
    - Ensures that the batch system doesn't completely take over the cloud
- The OpenNebula XML-RPC API is used instead of EC2 as it was found to be more reliable and flexible



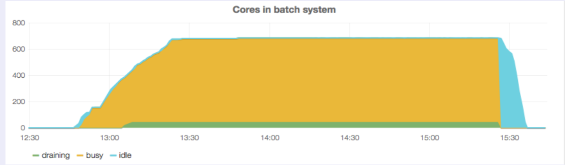
Results

Cores in the cloud



The batch system uses up cloud resources when needed but ensures there are always some free resources (cores & DHCP leases)

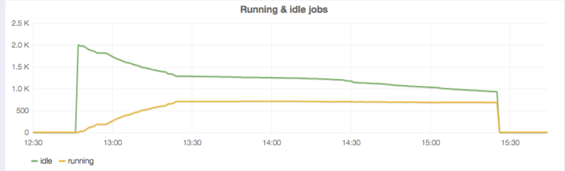
Cores in the batch system



Number of cores in the batch system increases when there are idle jobs, and decreases when there are no idle jobs

Draining is used to free-up cloud resources used by the batch system

Running and idle jobs



Conclusion

- We have demonstrated a simple method for allowing a HTCondor pool to make opportunistic usage of resources from a private cloud
  - The condor_rooster daemon is used to provision cloud resources
  - Virtualised worker nodes are drained and resources returned to the cloud when the cloud becomes busy
- Our next step will be to integrate the system described here with our production batch system and make use of the available cloud resources on a daily basis