Science & Technology Facilities Council

www.stfc.ac.uk

Experience with batch systems and clouds sharing the same physical resources

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

Goal

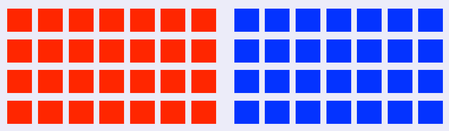
To provide both grid and cloud computing resources in an efficient way without static partitioning and without using virtualised worker nodes

Introduction

- It is becoming increasingly common for WLCG sites to provide both grid and cloud computing resources
- Having separate cloud and batch resources results is inflexible and results in poor utilisation due to static partitioning of resources – hypervisors for the cloud and worker nodes for the batch system
  - The cloud could be busy and the batch system largely idle, but there is no way the worker nodes can be used in the cloud
  - Alternatively, the batch system could be busy and the cloud idle
- The simplest and most common solution is to just use virtualised worker nodes, however this results in inefficiencies due to virtualisation overheads
- Here we present a preliminary investigation into running jobs directly on physical nodes which are also used as hypervisors for the cloud

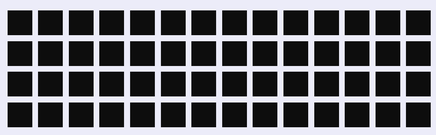
The problem

- The situation we would like to avoid is static partitioning, where our OpenNebula schedules jobs within its dedicated resources and our HTCondor batch system schedules jobs within its dedicated resources:



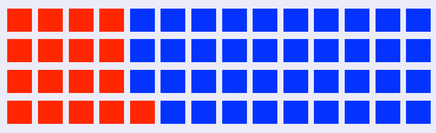
cloud *batch*

- The ideal situation is to have a common set of resources which can be divided up as necessary, in a dynamic way, between the cloud and the batch system:



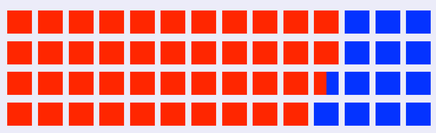
generic cloud/batch resources

- When the batch system is busy but not many cloud resources are required:



cloud *batch*

- When the cloud busy but not many batch system resources are required:

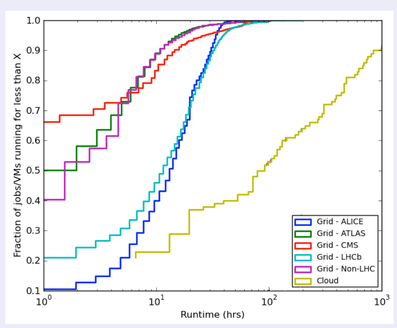


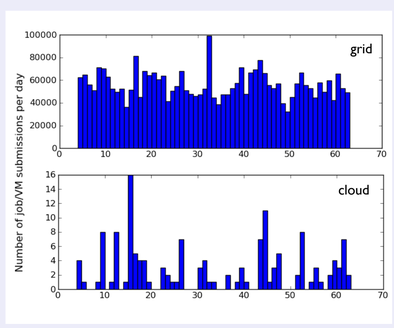
cloud *batch*

Possibly even nodes could provide resources to both the cloud and batch system simultaneously

Workload characterisations

- Batch jobs and cloud VMs have significantly different characteristics
  - Jobs running on the batch system are generally short-lived and submitted at a high rate
  - Currently the cloud is mainly used for development work, hence the rate at which VMs are created is low and the VMs are long-lived



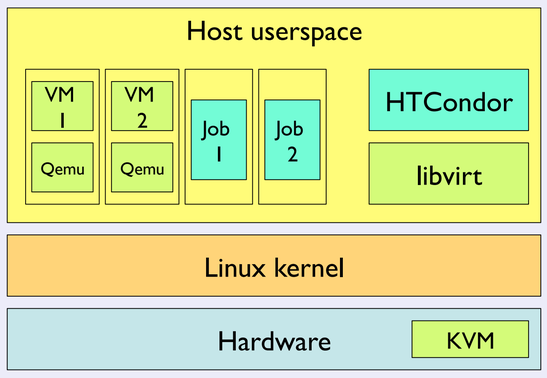


Two months of data showing the number of jobs submitted and VMs created per day

- The slow creation rate and long-lived nature of VMs on our cloud should mean it won't be too difficult to schedule the shorter-running batch jobs on the resources used by the cloud

VMs and batch jobs at the same time

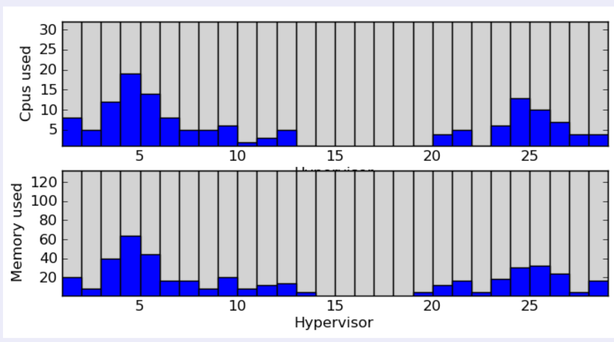
- It is essential to ensure that batch jobs cannot consume too many resources and starve VMs running on the same machine, potentially affecting performance
- Linux Control Groups (cgroups) provides a way managing resources used by groups of processes
  - Cloud VMs: our OpenNebula hypervisors currently use CPU cgroups to enforce the amount of CPU available to each VM
  - Batch jobs: our HTCondor worker nodes currently use both CPU and memory cgroups



A node acting as a hypervisor and worker node

Making the batch system aware of running VMs

- A startd cron can be used to make HTCondor aware of the CPU and memory resources which have been allocated to VMs
- The START expression can be configured to only allow jobs to start running which will fit within the remaining CPU and memory resources



Snapshot of the distribution of CPU and memory resources allocated to VMs on each hypervisor.

Blue indicates used resources & grey indicates free resources.

Both the OpenNebula scheduler and HTCondor can be configured to pack VMs/jobs tightly on as few nodes as possible rather than to distribute across as many nodes as possible. This would be expected to help with scheduling.

- We currently have OpenNebula using the actual CPU load as well as numbers of allocated CPUs in its scheduling decisions
  - Batch jobs using CPU on a hypervisor would result in OpenNebula not allocating the used CPU resources to VMs

Sharing resources

Cycle scavenging

- The simplest method would be to use HTCondor as it was originally designed, and just make use of any free resources on hypervisors
- If the resources of the hypervisor are exceeded, batch job(s) would need to be preempted
- Due to the low creation rate and long lifetime of cloud VMs, and the ability of OpenNebula minimise the number of hosts used by packing VMs, we would expect a low number of preemptions

Beyond cycle scavenging

- A simple method can be employed for switching a node between use as a hypervisor and as a worker node
  - If OpenNebula wants to run a VM on a node, a hook can change the START expression so that new batch jobs can't start
  - If HTCondor wants to run a job on an idle node, it can be configured to disable the node in OpenNebula so that VMs won't be created
- In order to ensure that the batch system doesn't completely take over the cloud, a condor_defrag daemon can be used to drain nodes, with the number of draining nodes configured to depend on the number of free hypervisors available

GridPP UK Computing for Particle Physics

GridPP Tier 1