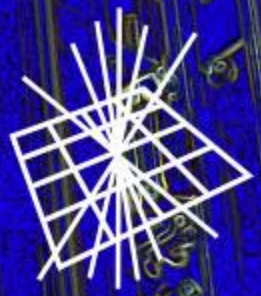


GridPP

UK Computing for Particle Physics

Experience with batch systems & clouds sharing the same physical resources

Andrew Lahiff, Alex Dibbo, George Ryall, Frazer Barnsley, Ian Collier

STFC Rutherford Appleton Laboratory

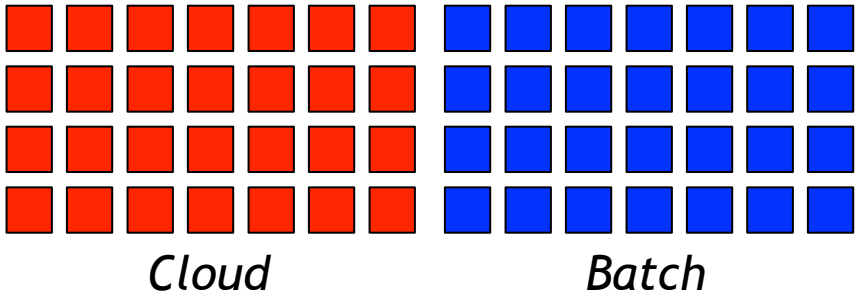
CHEP 2015

Okinawa, Japan

Science & Technology Facilities Council

Introduction

- Common situation: separate batch & cloud resources
  - RAL Tier-1 batch system
    - HTCondor
    - 560 worker nodes, 12000 cores
  - STFC Scientific Computing cloud
    - OpenNebula with Ceph storage backend
    - 28 hypervisors, 892 cores, 3.4 TB RAM, 750 TB raw storage
- Problem
  - Static partitioning

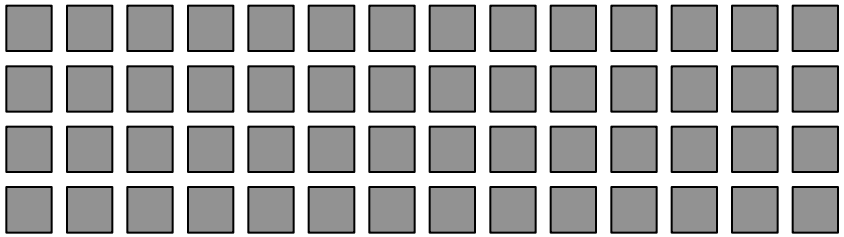


Issues / Common solution

- Issues
  - Cloud largely idle, not enough capacity in the batch system
  - Batch system could be idle, not enough capacity in the cloud
    - Less likely!
- Simplest solution
  - Cloud manages all resources
  - Worker nodes created in the cloud as needed
  - Resource provisioning
    - condor_rooster *(we've been using this)*
    - GlideinWMS
    - Elastiq
    - Cloud Scheduler
    - …
  - Worker nodes
    - Usually cloud VMs

Question

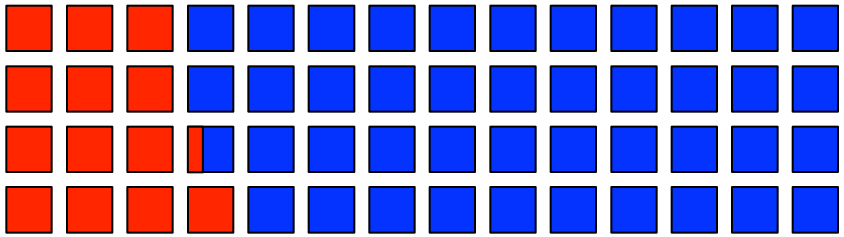
- Could the cloud & batch system share the same resources?
  - Machines setup as both hypervisors & worker nodes:



Generic cloud+batch resources

Question

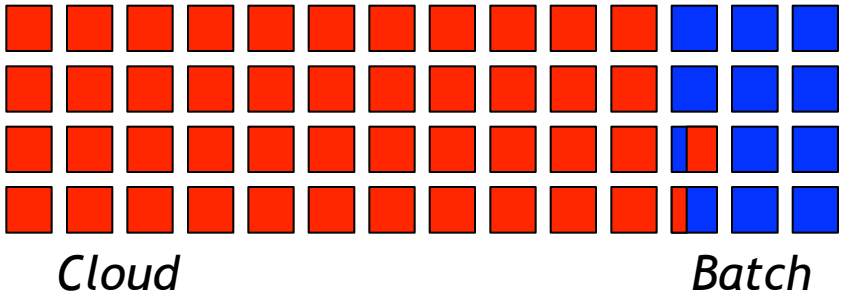
- Could the cloud & batch system share the same resources?
  - When the batch system is busy:



Possibly with VMs & jobs on the same machines at the same time

Question

- Could the cloud & batch system share the same resources?
  - When the cloud is busy:



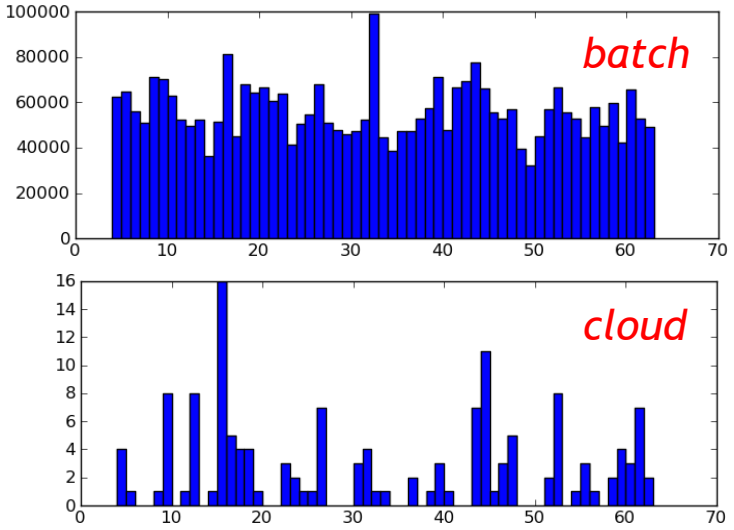
Possibly with VMs & jobs on the same machines at the same time

- Could OpenNebula & HTCondor independently schedule VMs & jobs on the same resources?

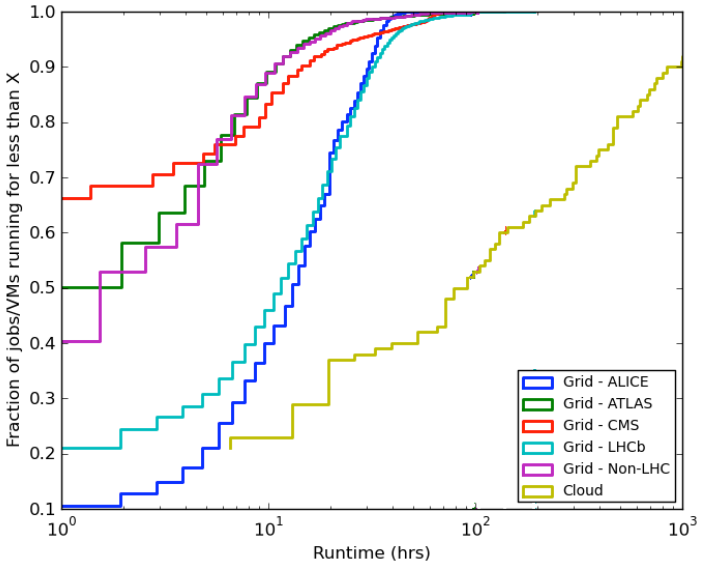
Workload characteristics

- Our batch jobs & cloud VMs have significantly different characteristics

Jobs submitted/VMs created per day (over a 2 month period)



Lifetimes of jobs & VMs



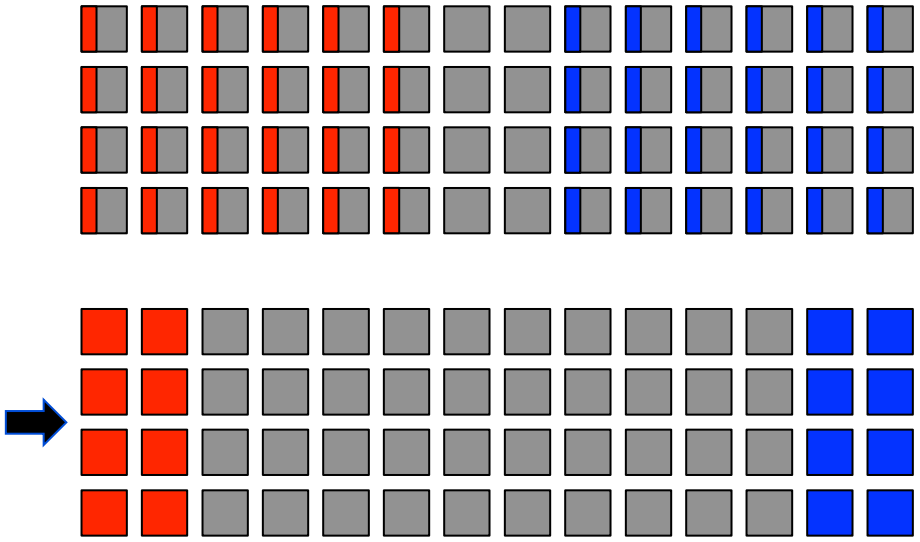
- The cloud is quite static compared to the batch system
  - Cloud currently only used by staff for development, …

VMs & batch jobs co-existing

- It is essential that batch jobs can't consume too many resources & starve VMs, potentially affecting performance
  - Or vice-versa
- Linux Control Groups (cgroups) provide a way of managing resources used by groups of processes
  - Batch jobs
    - Our HTCondor worker nodes have CPU & memory limits enforced via cgroups
  - Cloud VMs
    - Our OpenNebula hypervisors have CPU limits enforced via cgroups
      - Also possible to enforce memory limits
- Isolating jobs from each other, VMs & the host
    - Our HTCondor worker nodes use PID & filesystem namespaces

Sharing resources

- Keep jobs & VMs separated
  - Configure OpenNebula to
    - Pack VMs tightly
  - Configure HTCondor to
    - Pack jobs tightly
    - Prefer worker nodes which are not running VMs



- HTCondor awareness of VMs
  - Startd cron
  - START expression
    - Ensure CPU & memory used by VMs is taken into account
- OpenNebula awareness of jobs
  - CPU load used in scheduling decisions

Sharing resources

- Two ideas being considered
  - Simplest method
    - HTCondor originally used for “scavenging cycles” from idle desktops
    - Scavenge cycles from hypervisors
      - Expect to have low rate of job preemptions
  - More complex method
    - Switch nodes between use as hypervisor and worker node
      - Without using a central agent
    - OpenNebula
      - Setup a hook to prevents HTCondor from starting jobs on node
    - HTCondor
      - Configure to disable node in OpenNebula
      - Trigger live migration of VMs
      - Defrag daemon used to drain nodes if free resources become too scarce