Tier-1 experience with provisioning virtualised worker nodes on demand

Andrew Lahiff, Ian Collier
STFC Rutherford Appleton Laboratory, Harwell Oxford, UK

Introduction

- Interest in cloud computing by the major experiments has been gaining increasing momentum, in particular as a way to make use of opportunistic resources using direct submission to a cloud interface
- Nevertheless, grid submission to traditional batch systems is currently still the primary method of running jobs at WLCG sites
- The ability to use virtualised worker nodes from a cloud in a traditional batch system is potentially very useful, as it allows a site to:
  1. Take advantage of the benefits of virtualisation
  2. Provide both cloud and grid resources without partitioning
  3. Make use of a local private cloud when there are idle jobs in the batch system and there are free resources in the cloud
- Here we present recent work carried out at the RAL Tier-1 where we address point 3 above

SCD Cloud

- Prototype built to gain practical experience and test potential use cases for a private cloud
- Available to SCD staff on a self service basis, and has around 30 active users
- Using StratusLab
  - Based on OpenNebula
  - Very easy to set up – Quattor configurations provided
- Using iSCSI & LVM based persistent disk storage
  - Caches images
  - Instantiation very fast ~20-30 seconds, sometimes less
  - In the future will likely use parallel object store such as Ceph
- Cloud front end is on a VM (hosted on Hyper-V)
- Persistent disk store is on a 18TB retired disk server
- Compute resource ~100 retired batch workers with 8 cores and 16GB RAM

Images

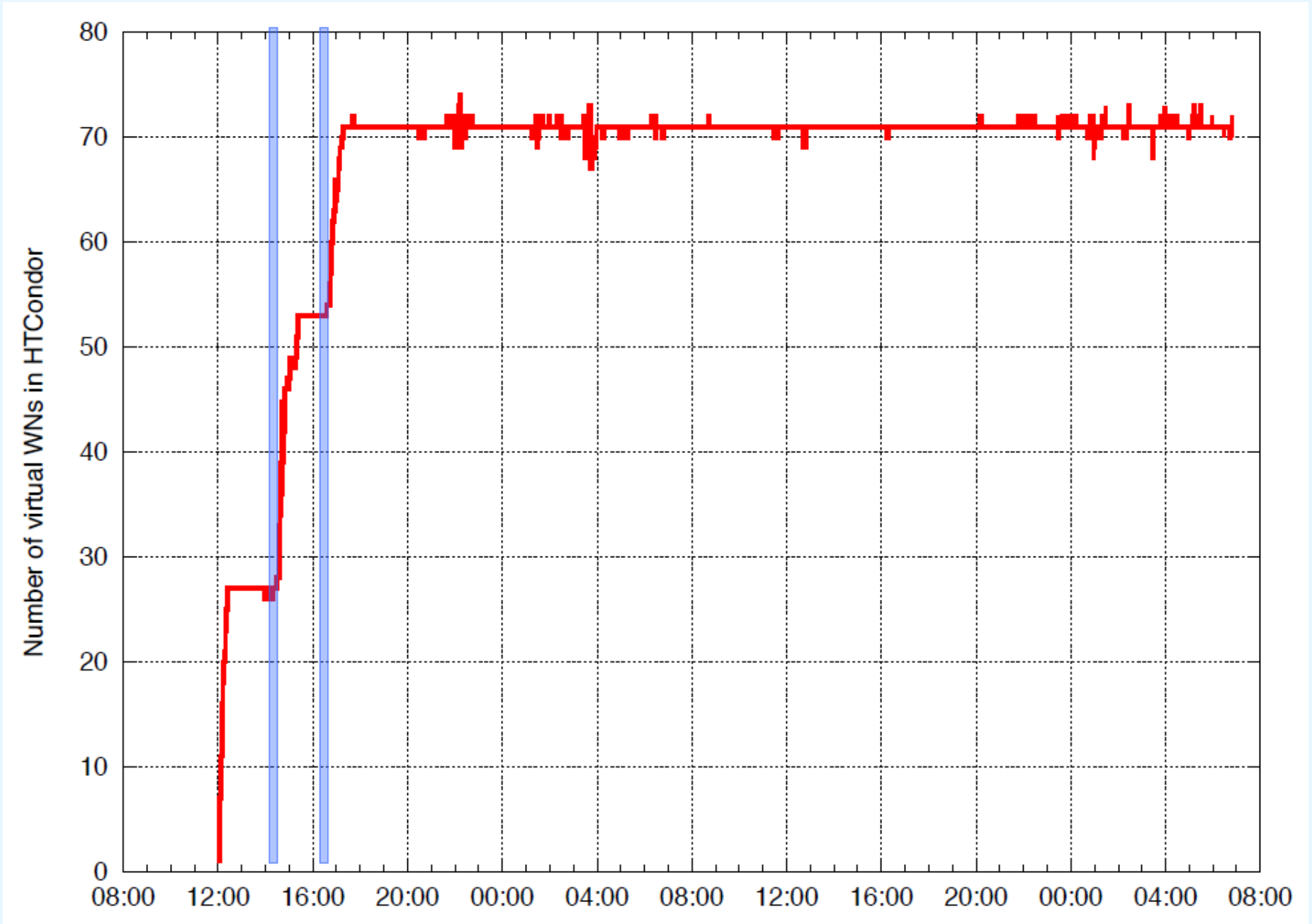
- Only images created by RAL sys admins are used, and are therefore trusted
- Images prepared using libvirt, KVM and Qemu
- EMI 2 SL6 worker node image identical to our physical worker nodes
  - Same privileges as physical worker nodes, including access to CASTOR

Testing and performance

- Preliminary investigations into the virtualisation overhead have been done by running the following on 8-core physical and virtual worker nodes:
  1. Benchmarking
  2. Copying files from CASTOR to the worker node using xrootd
  3. Typical CMS MC re-reconstruction workflow
- Identical hardware was used for the physical worker nodes and hypervisors
- No attempt has been made yet to improve performance

	HEPSEC06[1]	Read rate[2]	CPU efficiency[3]	Time per event[3]
Physical	69.45	79 MB/s	99.2%	14.7s
Virtual	64.82	22 MB/s	97.3%	16.1s

- Testing dynamically provisioned worker nodes



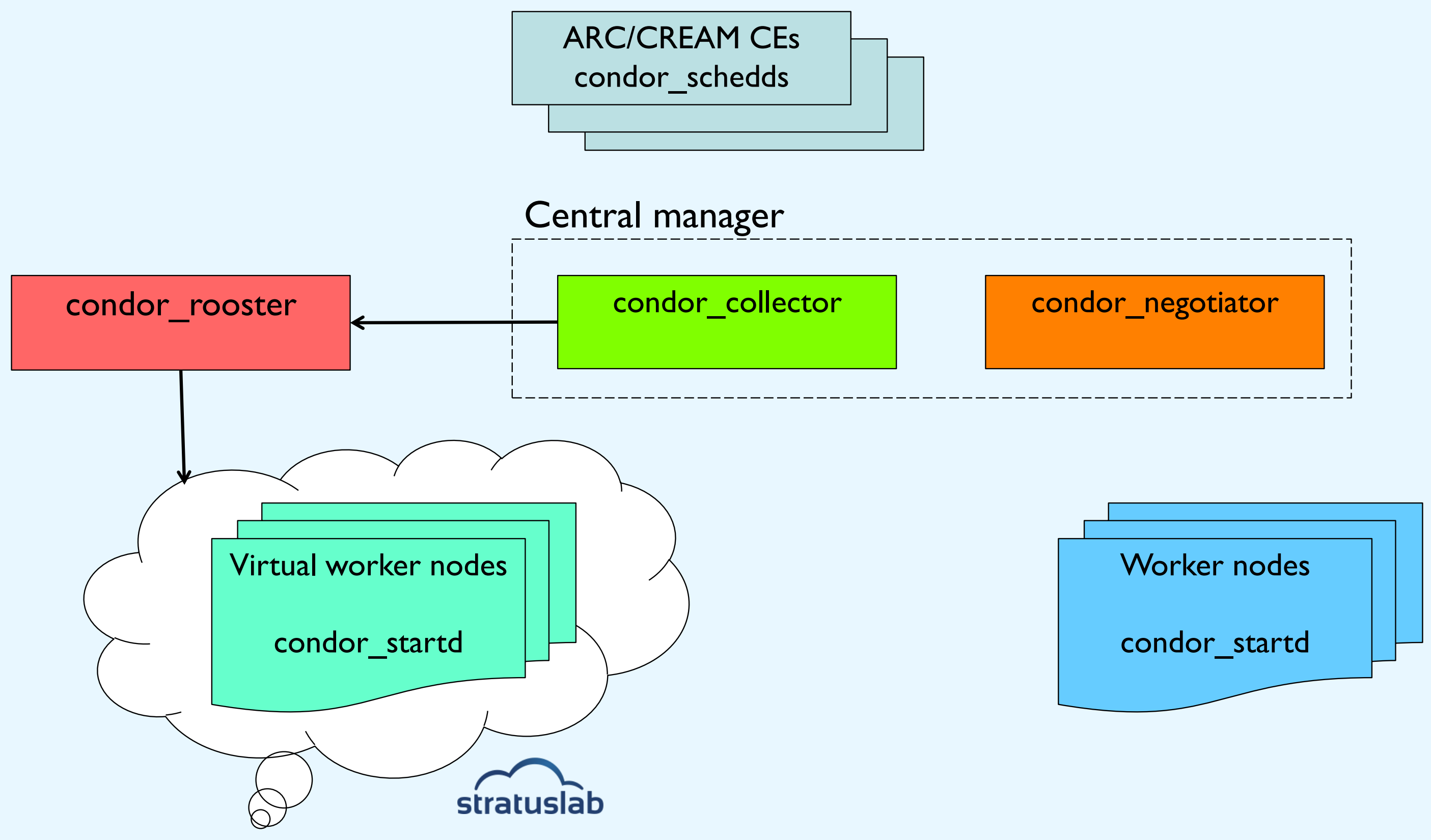
- System enabled at 12:00
- Blue lines indicate additional hypervisors enabled
- VMs have 4 cores, 8 GB RAM
- Available cloud resources are quickly used
- Number of running virtual worker nodes remains constant
- Networking, CPU and memory usage of the persistent disk storage remained low

Batch system at the RAL Tier-1

- The RAL batch system consists of 656 worker nodes and over 9300 job slots
- Torque/Maui was used for many years, but scalability and reliability problems lead us to investigate alternative technologies, resulting in HTCondor being selected
- One advantage of HTCondor over other batch systems is that it was designed to make use of opportunistic resources easily
  - Complex solutions have been developed to enable Torque to work with dynamically provisioned resources
  - SLURM has features which simplify the use of dynamic resources, but we rejected SLURM based on scalability testing
- HTCondor has recently gone into production at RAL, currently with 50% of the total CPU capacity

Integrating virtualised worker nodes

- Based on existing power management features of HTCondor
- **Virtual machine instantiation**
  - ClassAds for offline machines are sent to the collector
  - Negotiator can match idle jobs to the offline machines
  - Rooster daemon detects these matches, triggering the creation of VMs
  - Pool password inserted into VM using contextualisation with CloudInit
  - Volatile disks for the job scratch area, CVMFS cache and /tmp are created on the hypervisor's local disk
- **Virtual machine lifetime**
  - Managed entirely by HTCondor on the VM itself. Configured to:
    - Only start jobs when a worker node health-check script is successful
    - Only start new jobs for a specified time period
    - Shut the machine down once draining has completed



Discussion

- We have successfully demonstrated a simple method for provisioning virtual worker nodes and using them for running jobs in the production batch system
  - No modifications required to existing batch system or cloud infrastructure
  - Job failure rates no worse than physical worker nodes
- The RAL Tier-1 batch system almost always has idle jobs
  - How to prevent it from completely taking over the private cloud?
  - A fixed quota on the cloud isn't enough, really need to use fairshares
  - Current method: choose cores/memory per VM in such a way that there are always some resources leftover per hypervisor for other users
- Virtualisation overhead
  - Even without making any attempt to optimise performance, the current system is able to perform useful work using resources which would otherwise be idle
  - The main areas where improvement would be most beneficial are network and disk I/O
- Constant draining of VMs results in inefficiencies
  - Has benefits, however, e.g. rolling upgrades of kernel errata
  - Could be resolved by using single-core VMs, or perhaps running short jobs while long jobs are draining
  - Alternatively run long-lived VMs
- Monitoring
  - Existing monitoring infrastructure doesn't handle dynamic environments where nodes are coming and going: need to find solutions for this

GridPP UK Computing for Particle Physics
GridPP TIER1
stratuslab