Science & Technology Facilities Council

HTCondor at RAL: An Update

Andrew Lahiff

STFC Rutherford Appleton Laboratory

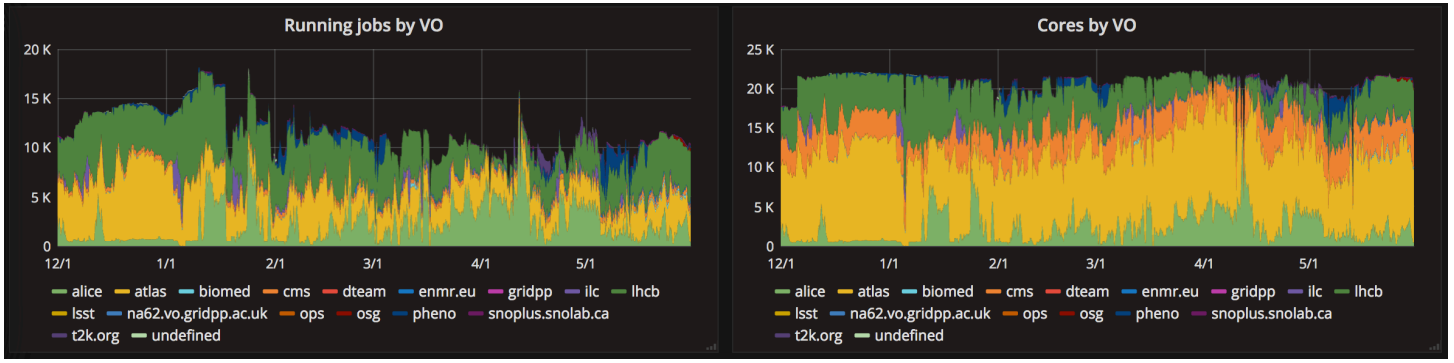
8th June, European HTCondor Workshop 2017

Overview

- Introduction
- Docker universe
- Monitoring
- Multi-core jobs
- Private cloud
- Containers
  - on-premises
  - public clouds
- Plans

Introduction

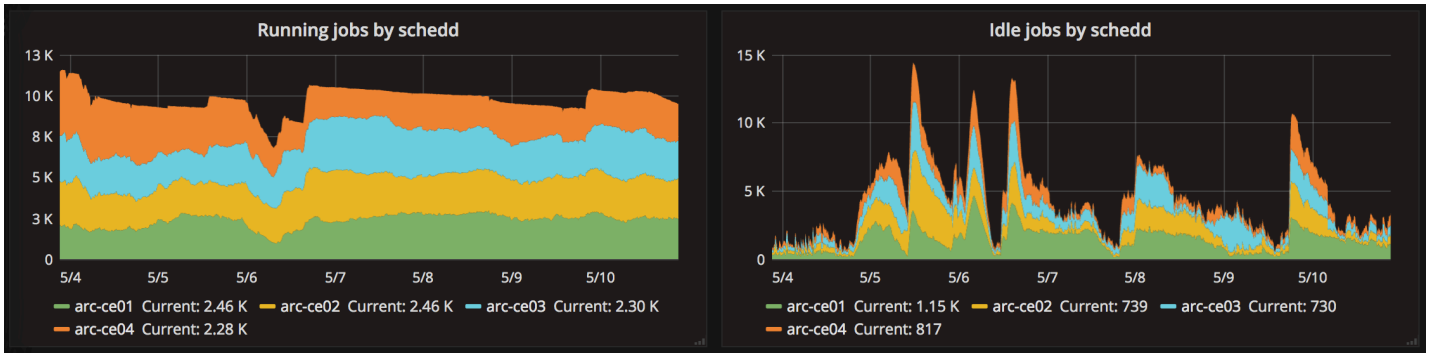
- RAL Tier-1 supports
  - ALICE, ATLAS, CMS, LHCb
  - many others, both HEP & non-HEP
- Batch system
  - 24000 cores (soon 26000)
  - HTCondor 8.6.3 (worker nodes, CMs), 8.4.8 (CEs)



Past 6 months

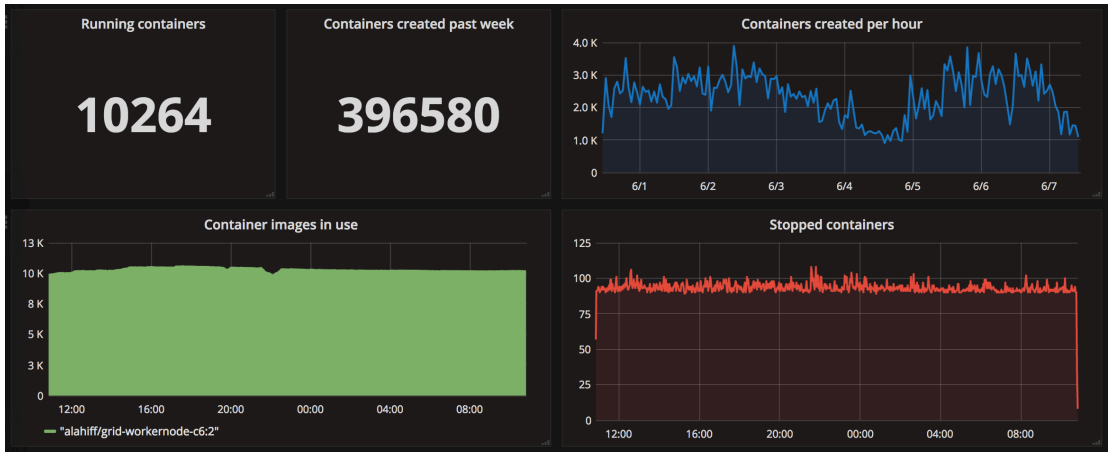
Computing elements

- Currently all job submission is via the Grid
- 4 ARC CEs
  - working fine
    - still using 5.0.5
    - generally don't upgrade very aggressively
  - ATLAS submit jobs using pre-loaded pilots
    - jobs specify their actual resource requirements
  - no plans to migrate to HTCondor CEs yet



Docker universe

- We migrated to 100% Docker universe this year
  - migration to SL7 took place over several months
  - Docker 17.03 CE using OverlayFS storage backend
  - worker node filesystem:
    - / (RAID 0, XFS)
    - /pool (RAID 0, XFS with ftype=1) for Docker, job sandboxes & CVMFS cache



Docker universe

- HTCondor config relating to Docker

```
DOCKER=/usr/local/bin/docker.py
DOCKER_DROP_ALL_CAPABILITIES=regexp("pilot",x509UserProxyFirstFQAN) =?= False
DOCKER_MOUNT_VOLUMES=GRID_SECURITY, MJF, GRIDENV, GLEXEC, LCMAPS, LCAS, PASSWD, GROUP, CVMFS,
CGROUPS, ATLAS_RECOVERY, ETC_ATLAS, ETC_CMS, ETC_ARC
DOCKER_VOLUME_DIR_ATLAS_RECOVERY=/pool/atlas/recovery:/pool/atlas/recovery
DOCKER_VOLUME_DIR_ATLAS_RECOVERY_MOUNT_IF=regexp("atl",Owner)
DOCKER_VOLUME_DIR_CGROUPS=/sys/fs/cgroup:/sys/fs/cgroup:ro
DOCKER_VOLUME_DIR_CGROUPS_MOUNT_IF=regexp("atl",Owner)
DOCKER_VOLUME_DIR_CVMFS=/cvmfs:/cvmfs:shared
DOCKER_VOLUME_DIR_ETC_ARC=/etc/arc:/etc/arc:ro
DOCKER_VOLUME_DIR_ETC_ATLAS=/etc/atlas:/etc/atlas:ro
DOCKER_VOLUME_DIR_ETC_ATLAS_MOUNT_IF=regexp("atl",Owner)
DOCKER_VOLUME_DIR_ETC_CMS=/etc/cms:/etc/cms:ro
DOCKER_VOLUME_DIR_ETC_CMS_MOUNT_IF=regexp("cms",Owner)
DOCKER_VOLUME_DIR_GLEXEC=/etc/glexec.conf:/etc/glexec.conf:ro
DOCKER_VOLUME_DIR_GRIDENV=/etc/profile.d/grid-env.sh:/etc/profile.d/grid-env.sh:ro
DOCKER_VOLUME_DIR_GRID_SECURITY=/etc/grid-security:/etc/grid-security:ro
DOCKER_VOLUME_DIR_GROUP=/etc/group:/etc/group:ro
DOCKER_VOLUME_DIR_LCAS=/etc/lcas:/etc/lcas:ro
DOCKER_VOLUME_DIR_LCMAPS=/etc/lcmaps:/etc/lcmaps:ro
DOCKER_VOLUME_DIR_MJF=/etc/machinefeatures:/etc/machinefeatures:ro
DOCKER_VOLUME_DIR_PASSWD=/etc/passwd:/etc/passwd:ro
```

Docker universe

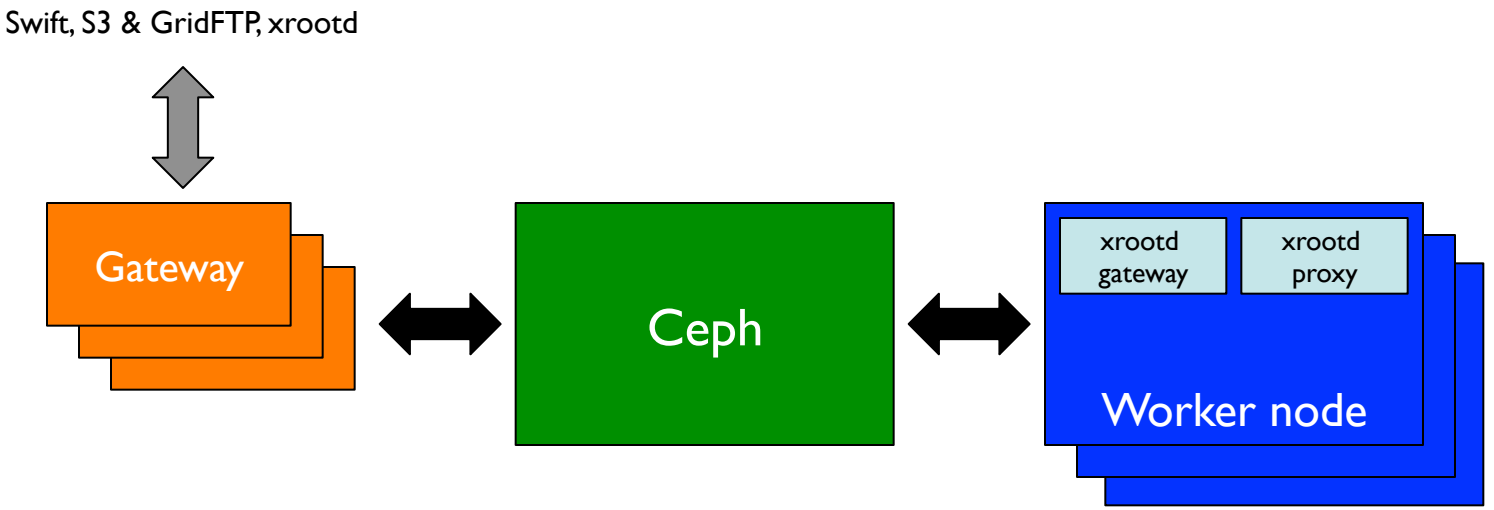
- Before the Docker universe, used
  - cgroups with soft memory limits
  - PERIODIC_SYSTEM_REMOVE to kill jobs which exceed 3x requested memory
- With the Docker universe
  - by default jobs are given a hard memory limit
    - high rate of jobs being killed (mainly ATLAS)
  - now use a script which modifies arguments to "docker run" generated by HTCondor (the "DOCKER" knob)
    - set a soft memory limit at the memory requested by job
    - hard limit at 2x requested memory

Docker universe

- Goodbye MOUNT_UNDER_SCRATCH
  - we used this for a long time
    - enables /tmp to be unique for each job
  - no need for this with the Docker universe
    - /tmp is already unique to each container
    - with OverlayFS storage backend /tmp is more than big enough
- Converting Vanilla to Docker universe
  - currently using job routers on each CE
  - will switch to schedd job transforms once we upgrade CEs to 8.6.x

Docker universe & storage

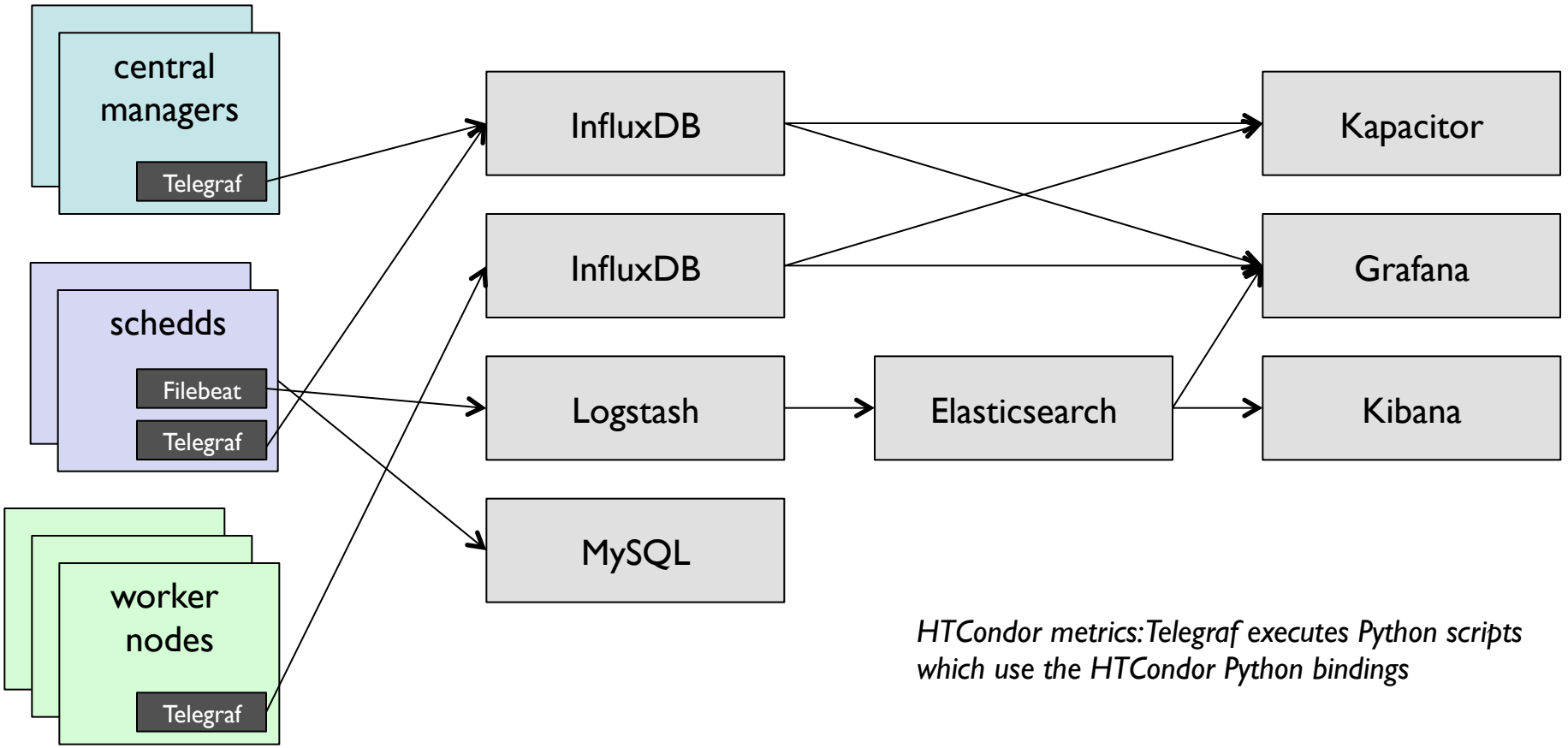
- Started rolling out xrootd Ceph gateways & proxies onto worker nodes
  - an important driver for migrating to SL7 worker nodes
  - xrootd daemons running in containers
  - jobs think they're talking to the remote gateway, but they're actually talking to the local proxy



Monitoring

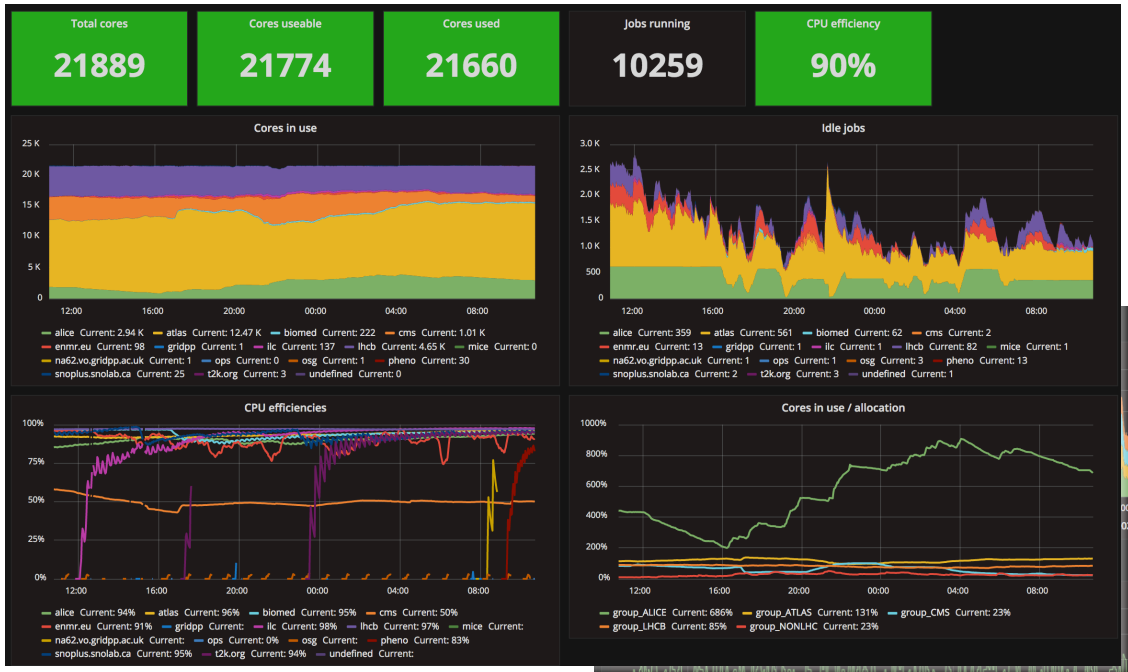
- Still have condor_gangliad running
  - but generally never look at Ganglia
  - gmond no longer being installed on some services
- Alternative in use since 2015
  - Telegraf for collecting metrics
  - InfluxDB time series database
  - Grafana for visualisation
  - Kapacitor for stream processing of metrics
- ClassAds for completed jobs
  - Filebeat, Logstash, Elasticsearch
  - MySQL (accounting)

Monitoring: overview



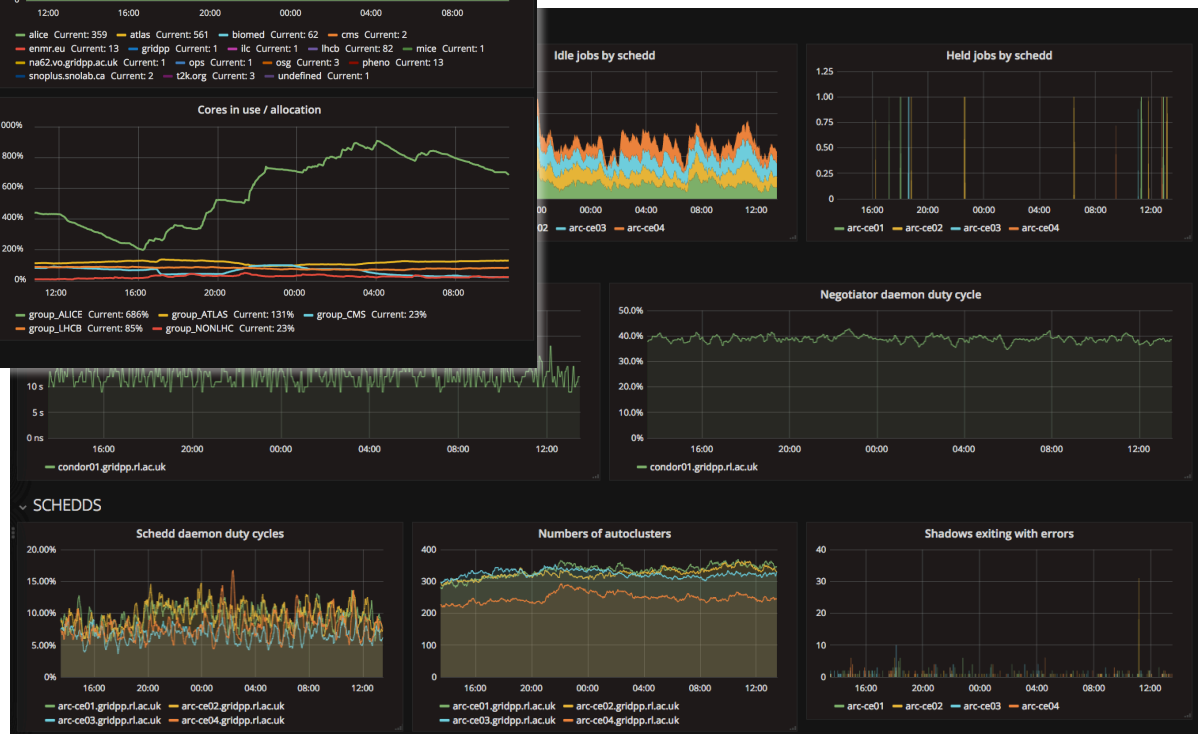
HTCondor metrics: Telegraf executes Python scripts which use the HTCondor Python bindings

Monitoring: dashboards



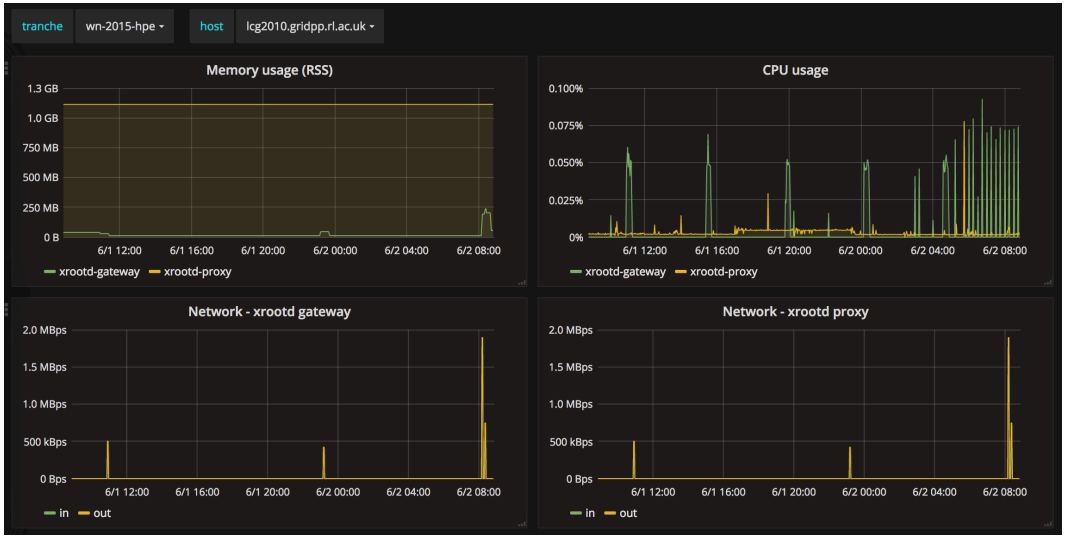
Overview dashboard

Cluster health dashboard



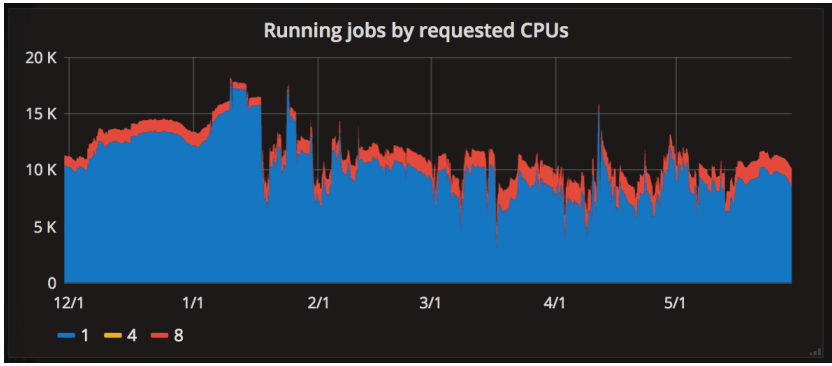
Monitoring: containers

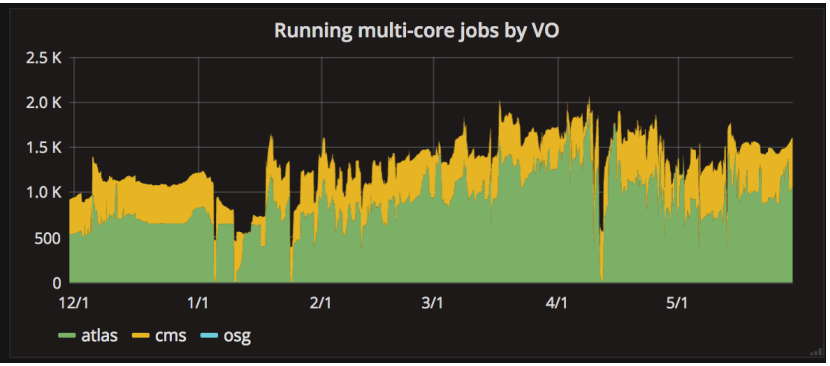
- Using Telegraf to collect container metrics from WNs
  - currently just xrootd gateway & proxy containers
- No per-job time-series metrics yet, will revisit when
  - InfluxDB 1.3 is released
    - supports high cardinality indexing
  - show tag values with a WHERE time clause



Multi-core jobs

- Multi-core jobs account for ~50% number of used cores
  - all 8-cores
  - only ATLAS & CMS
- Haven't made any changes for over a year
  - it just works





Past 6 months

Multi-core jobs

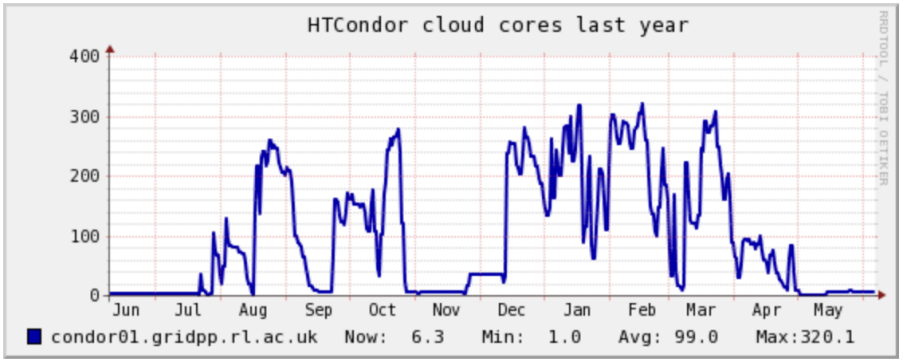
- We don't use the defrag daemon or the "Drained" state
  - why have any idle cores?
  - we want to be able to run preemptable jobs on cores which are draining
- START expression on worker nodes contains:

```
ifThenElse($(PREEMPTABLE_ONLY), isPreemptable =?= True, True)
```

- Draining sets PREEMPTABLE_ONLY to True
  - only jobs with isPreemptable set to True can run
  - currently only ATLAS Event Service jobs
- Preemptable jobs (if any) are killed when enough cores have been drained on a node

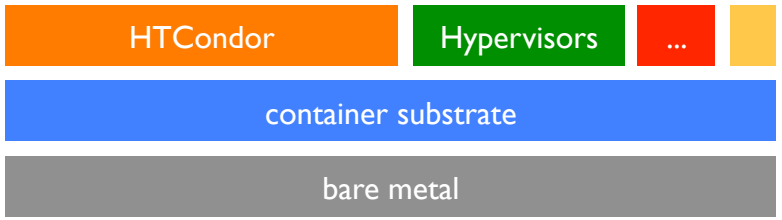
Cloud

- For several years have been making opportunistic use of our private OpenNebula cloud
  - using condor rooster for provisioning virtual worker nodes
  - only idle cloud resources used
    - backs off if cloud usage increases
- Future (short-term)
  - migrating to OpenStack
  - migrating to CernVMs



Worker nodes & containers

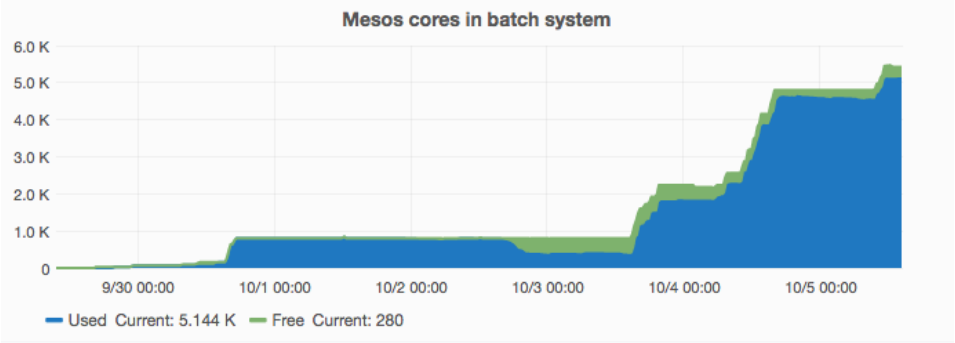
- Investigating ways of having more flexible computing infrastructure
  - need to be able to support more communities, not just LHC experiments
  - single set of resources for
    - HTCondor
    - OpenStack hypervisors
    - Other Stuff™



- Ideally
  - install nothing related to any experiment/community on any machine
  - run everything in containers, managed by schedulers (not people)

Worker nodes & containers

- Example using Apache Mesos as the cluster manager
  - last year did some tests with real jobs from all 4 LHC experiments
    - running >5000 cores
  - Mesos cluster containing 248 nodes
  - custom Mesos framework
    - creates startds in containers which join our production HTCondor pool
  - auto-scaling squids in containers for CVMFS/Frontier

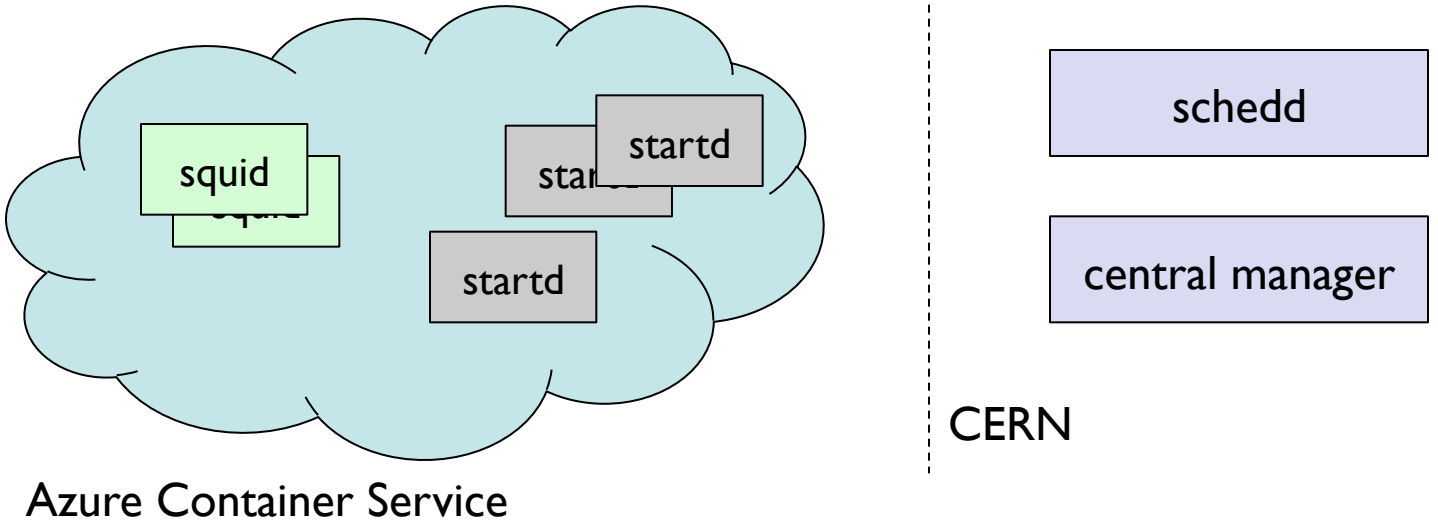


Kubernetes as an abstraction layer

- Kubernetes is an open-source container cluster manager which can be run anywhere
  - on-premises
  - "click a button" on public clouds (natively or via 3rd parties)
- Using it as an abstraction to enable portability between
  - on-premises resources
  - multiple public clouds
- Benefits
  - Don't need to worry about handling different cloud APIs
  - Run a single command to create an elastic, self-healing Grid site
    - on a single cluster
    - on multiple clusters around the world (via Kubernetes federation)

Kubernetes as an abstraction layer

- Did initial testing with CMS CRAB3 analysis jobs
  - RAL, Google (GKE), Azure (ACS), AWS (StackPointCloud)
- Now running ATLAS production jobs on Azure
  - using "vacuum" model for independently creating startds which join a HTCondor pool at CERN



Thanks to Microsoft for an Azure Research Award

Upcoming plans include

- The return of local users
  - haven't had local users for many years
  - would like to be able to support users of local facilities
    - job submission using Kerberos auth?
    - federated identity? Can INDIGO IAM help?
- SLA-aware automated rolling upgrades
  - use a key-value store to coordinate draining & reboots
    - idea from CoreOS
    - machines need to get access to a lock to drain
  - have done PoC testing but not yet tried in production
  - not HTCondor-specific, could be useful other services
- Getting rid of pool accounts!!!

Questions?