Using container orchestration to improve service management at the RAL Tier 1

Andrew Lahiff, Ian Collier

Rutherford Appleton Laboratory

10th October, CHEP 2016, San Francisco

Overview

- Motivation & goals
- Apache Mesos
- Marathon
- Image creation & storage
- Metrics & logging
- Service discovery
- Example
- Summary & future plans

Motivation

- UK Tier-1 facility at RAL
  - provides resources to all 4 LHC experiments, many non-LHC experiments
  - also provides resources to local facilities, e.g. Diamond Light Source, ISIS Neutron & Muon Source
- Changing landscape
  - LHC resource requirements increasing in size
  - broadening user base of the Tier-1
  - non-LHC experiments & local facilities becoming more important
    - allocation expected to exceed UK contribution to the LHC in coming years
  - staff effort unlikely to increase & in fact will probably decrease
- Important to investigate ways of managing existing services & potentially provide more services with less effort

How do we run services?

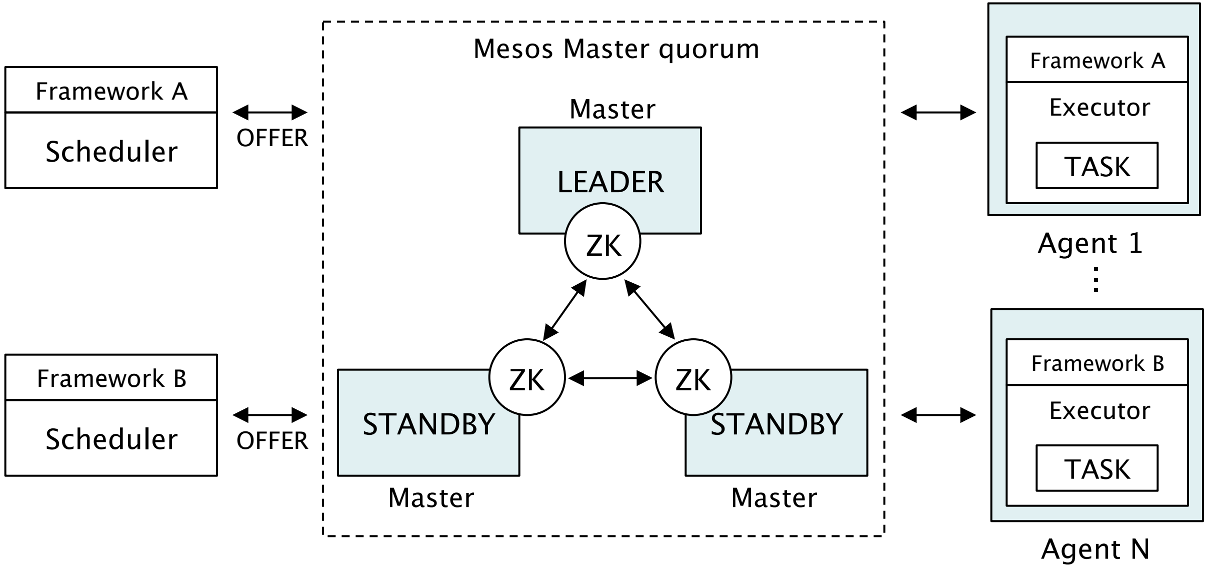
- Aquilon (Quattor) for configuration management
- Services run on enterprise virtualization platforms or in some cases bare metal
- Significant amount of manual work involved for both deployment & upgrades, also:
    - manual intervention is required to resolve problems
        - many Nagios tests but at most they result in emails or pager alarms
    - very static environment
        - many manual steps in order to scale horizontally
    - problem with VM or hypervisor results in loss of service
        - recent improvement due to use of shared storage under hypervisors
    - dead machines visible to users due to use of static DNS aliases
        - started to use load balancers to get around this (only FTS3 so far)
- Very little has changed in the past 6 years or more

Goals

- Ideally would like to have an environment where
  - The **infrastructure** itself is
    - felixible
    - fault-tolerant
    - scalable
  - **Services** are
    - quickly & easily deployable, easily updated
    - self-healing
    - elastic, auto-scaling
    - multi-tenant
- Looking at what's happening in the wider world, one way to do this is to
  - migrate to running applications in containers
  - manage the containers using schedulers, not people
- This is a very significant change in the way we manage services

Apache Mesos

- Originated in UC Berkeley in 2011 & became a Top Level Project at Apache in 2013
- Mesos is a cluster manager which
  - enables a large group of machines to appear as a single pool of resources
  - allows you to have multiple schedulers sharing the same resources



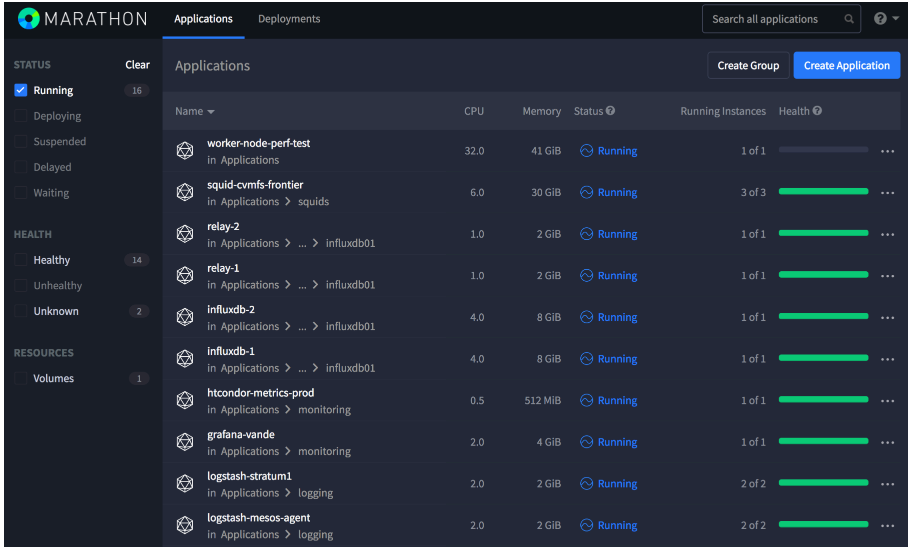
Mesos agents provide resources to the Mesos master

Mesos master offers resources to frameworks

Frameworks decide what offers to accept & what to do with them

Marathon

- A Mesos framework for long-running services
  - applications & groups of applications with dependencies
  - health checks



MARATHON   Applications   Deployments   Search all applications

STATUS   Clear
- ☑ Running 16
- □ Deploying
- □ Suspended
- □ Delayed
- □ Waiting

HEALTH
- □ Healthy 14
- □ Unhealthy
- □ Unknown 2

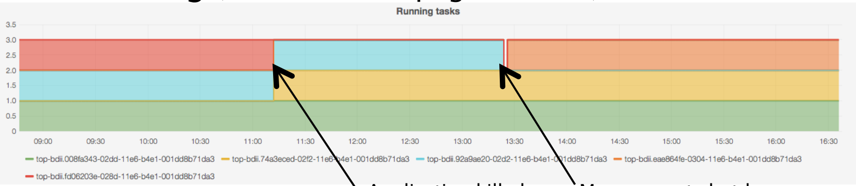
RESOURCES
- □ Volumes 1

Applications   Create Group   Create Application

Name	CPU	Memory	Status	Running Instances	Health
worker-node-perf-test in Applications	32.0	41 GiB	Running	1 of 1	
squid-cvmfs-frontier in Applications > squids	6.0	30 GiB	Running	3 of 3	
relay-2 in Applications > ... > influxdb01	1.0	2 GiB	Running	1 of 1	
relay-1 in Applications > ... > influxdb01	1.0	2 GiB	Running	1 of 1	
influxdb-2 in Applications > ... > influxdb01	4.0	8 GiB	Running	1 of 1	
influxdb-1 in Applications > ... > influxdb01	4.0	8 GiB	Running	1 of 1	
htcondor-metrics-prod in Applications > monitoring	0.5	512 MiB	Running	1 of 1	
grafana-vande in Applications > monitoring	2.0	4 GiB	Running	1 of 1	
logstash-stratum1 in Applications > logging	2.0	2 GiB	Running	2 of 2	
logstash-mesos-agent in Applications > logging	2.0	2 GiB	Running	2 of 2	

Benefits include

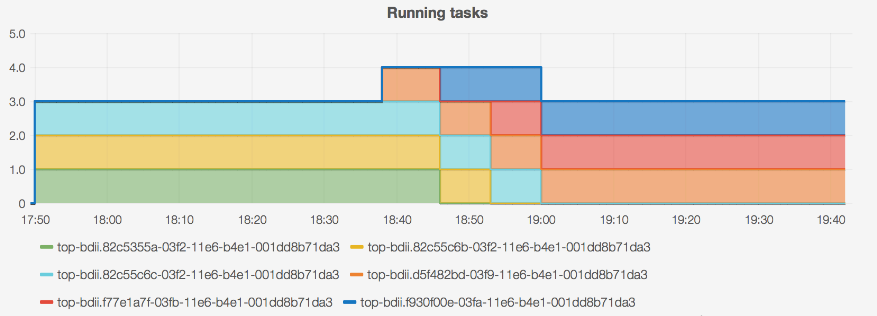
Self-healing (no more 2am pager alarms)



Application killed

Mesos agent shutdown

Rolling upgrades

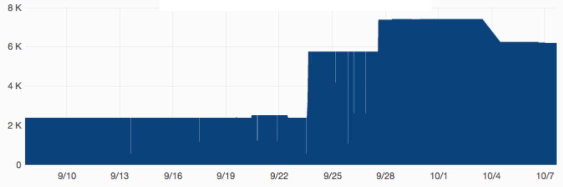


Each colour represents a task (instance of an application)

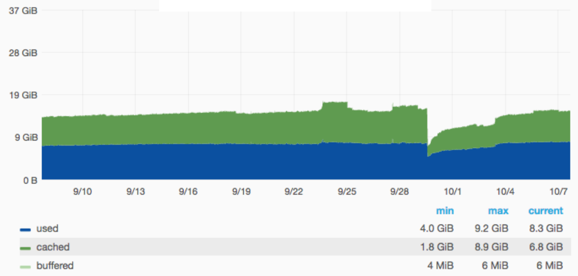
Deployment at RAL

- Mesos agents
  - 164 x 32 cores, 84 x 16 cores (all bare metal)
  - Mesos agent, Docker engine, Consul agent
- Mesos masters
  - 5 VMs (4 cores, 8 GB memory each)
    - can lose up to 2 hosts without any problems
  - Mesos master, ZooKeeper, Marathon, Consul server
- Some experiences
  - CPU & memory usage on the Mesos masters is low
    - Consul biggest user of CPU
  - Disk i/o important for ZooKeeper
    - 2 of our 3 virtualization platforms don't quite have fast enough disk storage

Number of cores in cluster

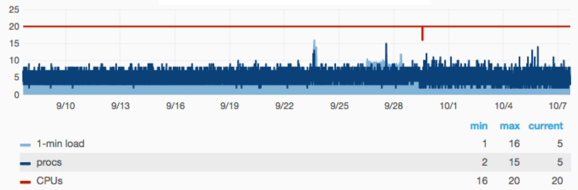


Memory (summed across masters)



	min	max	current
used	4.0 GiB	9.2 GiB	8.3 GiB
cached	1.8 GiB	8.9 GiB	6.8 GiB
buffered	4 MiB	6 MiB	6 MiB

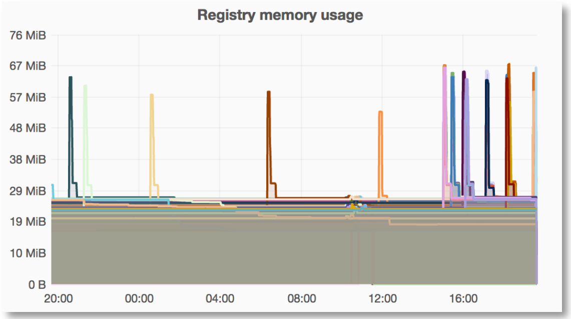
CPU (summed across masters)



	min	max	current
1-min load	1	16	5
procs	2	15	5
CPUs	16	20	20

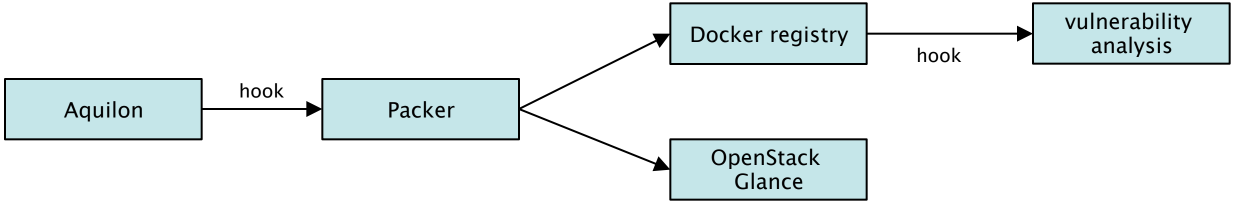
Image creation & storage

- Distributed private Docker registry
  - Ceph storage backend via Swift gateway
  - central registry instance providing write access
  - read-only registry on every Mesos agent
    - lightweight
    - avoids having a single point of failure / bottleneck



Memory usage of registry container on each agent

- Creating images
  - currently creating images “by hand” from Dockerfiles
  - work in progress on leveraging Packer to build VM & container images from our configuration management system (Aquilon)

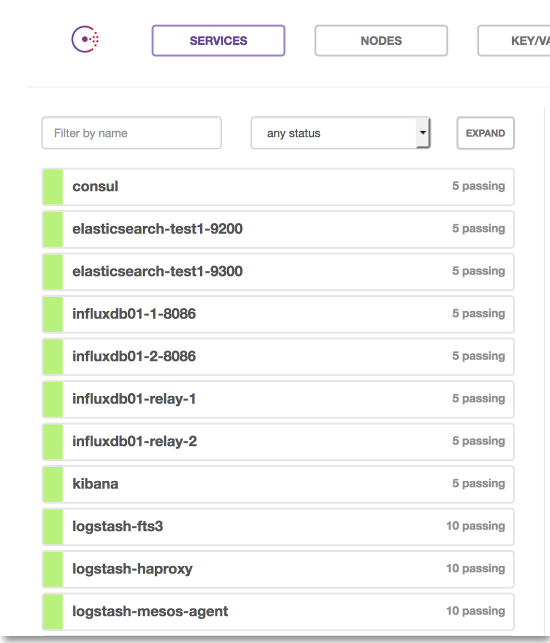


Metrics & logging

- Traditional monitoring is host-centric & assumes static infrastructure
  - not suitable for a dynamic containerized environment
  - no static hostnames or IP addresses & number of instances can change
- Use metadata to dynamically aggregate metrics & logs
  - e.g. application name rather than hostname
- Metrics
  - Telegraf: collecting infrastructure metrics (input plugins for Mesos, ZooKeeper, ...)
  - cAdvisor: collecting container metrics (resource usage, application metrics)
  - InfluxDB, Grafana
- Logging
  - centralized is logging important: applications can move around
  - Filebeat (stdout/err from containers, Mesos logs, ...)
  - Logstash, Elasticsearch, Kibana

Service discovery

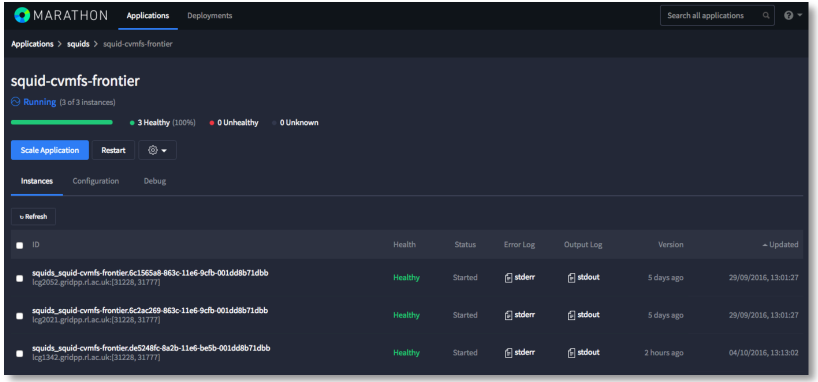
- Static hostnames in configuration files no longer make sense
- Using Consul, a distributed tool for service discovery
  - Containers exposing ports are automatically registered in Consul
- Within the Mesos cluster
  - Services can be accessed in several ways, including DNS, e.g.
    - logstash.service.consul
- External access to services in the Mesos cluster
  - Pairs of load balancers
    - HAProxy (load balancing)
    - Keepalived (HA floating IP addresses)
  - Configuration dynamically updated by Consul
    - It’s possible to setup HAProxy for zero-downtime reloads



consul	5 passing
elasticsearch-test1-9200	5 passing
elasticsearch-test1-9300	5 passing
influxdb01-1-8086	5 passing
influxdb01-2-8086	5 passing
influxdb01-relay-1	5 passing
influxdb01-relay-2	5 passing
kibana	5 passing
logstash-fts3	10 passing
logstash-haproxy	10 passing
logstash-mesos-agent	10 passing

Example: worker nodes

- Investigating the ability to run HTCondor worker nodes on Mesos
  - Existing production HTCondor central managers & ARC CEs
  - Running on Mesos
    - worker nodes
    - squids
- Container management
  - Marathon for squids
    - autoscaling based on request rate
  - A custom framework for worker nodes
    - creates worker node containers as needed
    - Why not Marathon? Need to be able to scale down & perform rolling upgrades without killing jobs
- Marathon & the custom framework registered in Mesos as different roles
  - can ensure that worker nodes can't take over entire cluster



Applications > squids > squid-cvmfs-frontier

squid-cvmfs-frontier

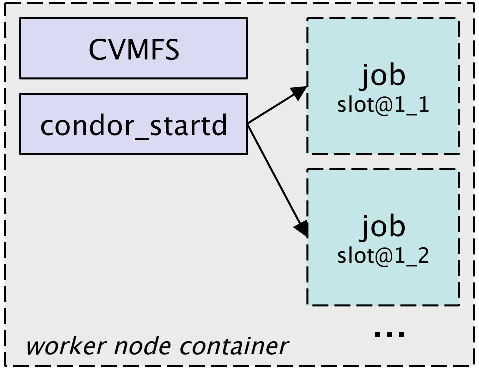
Running (3 of 3 instances)

● 3 Healthy (100%) ● 0 Unhealthy ● 0 Unknown

□	ID	Health	Status	Error Log	Output Log	Version	Updated
□	squids_squid-cvmfs-frontier.6c1565a8-863c-11e6-9cfb-001dd8b71dbb lcg2052.gridpp.rl.ac.uk:[31228, 31777]	Healthy	Started	stderr	stdout	5 days ago	29/09/2016, 13:01:27
□	squids_squid-cvmfs-frontier.6c2ac269-863c-11e6-9cfb-001dd8b71dbb lcg2021.gridpp.rl.ac.uk:[31228, 31777]	Healthy	Started	stderr	stdout	5 days ago	29/09/2016, 13:01:27
□	squids_squid-cvmfs-frontier.de5248fc-8a2b-11e6-be5b-001dd8b71dbb lcg1342.gridpp.rl.ac.uk:[31228, 31777]	Healthy	Started	stderr	stdout	2 hours ago	04/10/2016, 13:13:02

Worker nodes

- CVMFS & condor_startd inside the container
  - host doesn’t need anything at all related to worker nodes installed
  - allows us to run as many worker nodes as required without having to dedicate a set of resources configured as “WLCG worker nodes”
- Each job
  - runs in it’s own CPU & memory cgroups nested in the worker node container
  - has it’s own PID & mount namespace
- Container exits if there has been no work for a specified duration



CVMFS

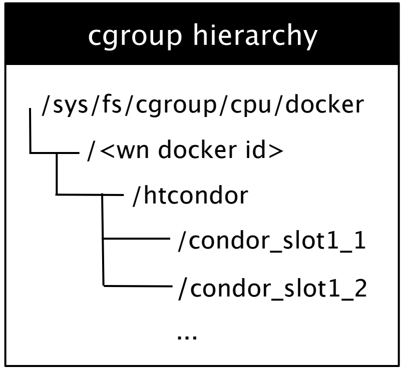
condor_startd

job
slot@1_1

job
slot@1_2

…

worker node container



cgroup hierarchy

/sys/fs/cgroup/cpu/docker

/<wn docker id>

/htcondor

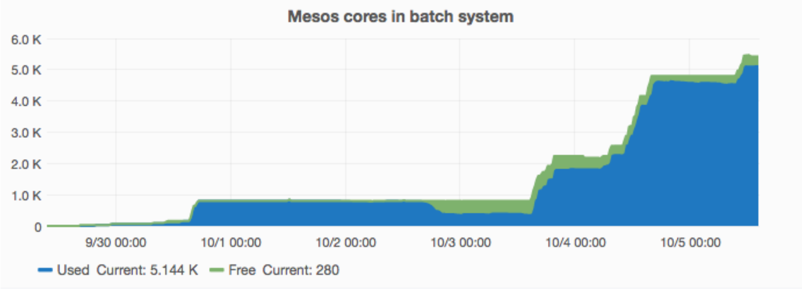
/condor_slot1_1

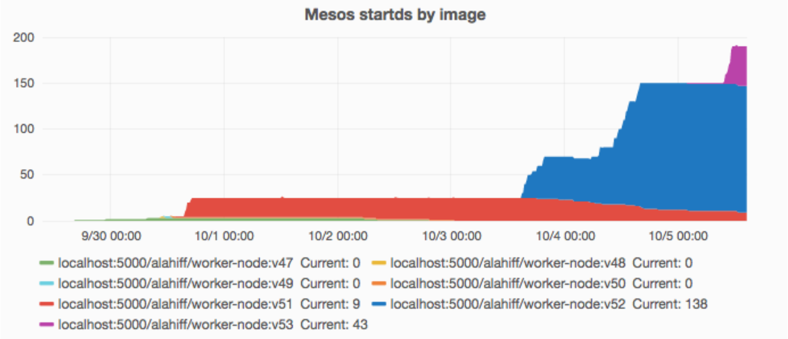
/condor_slot1_2

…

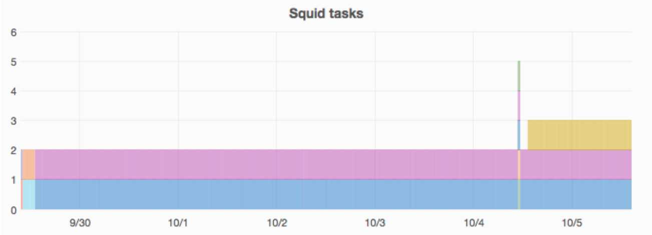
Worker nodes

- Example of recent tests with real jobs from all 4 LHC experiments

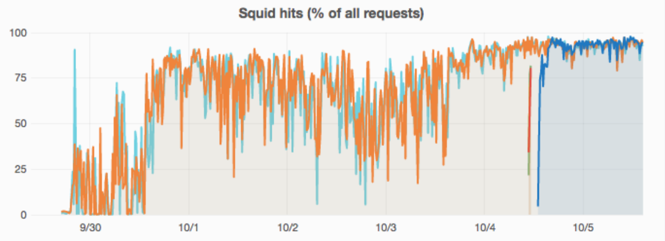




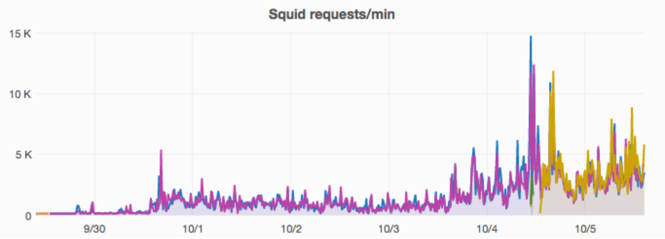
HTCondor startd ClassAds contain details such as Mesos task ID & image name which are automatically added to job ClassAds for traceability



Each colour correpsonds to a task (instance of an application)



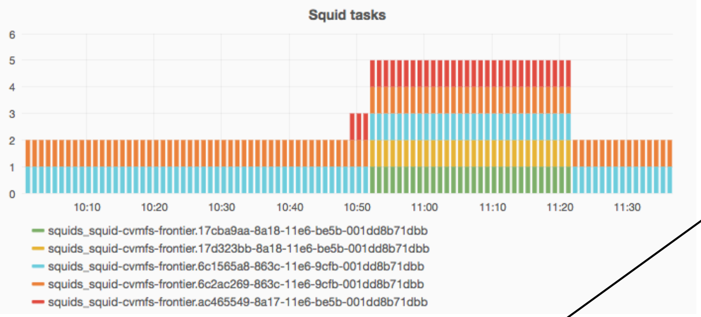
Appliication metrics collected by cAdvisor



Each new squid automatically starts receiving traffic after being created

Auto-scaling

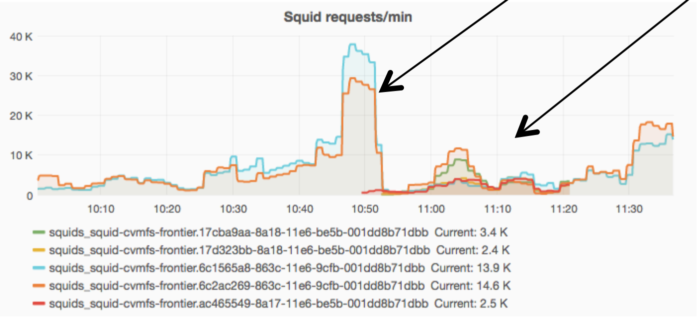
- Example of the number of squid instances changing based on load (request rate)

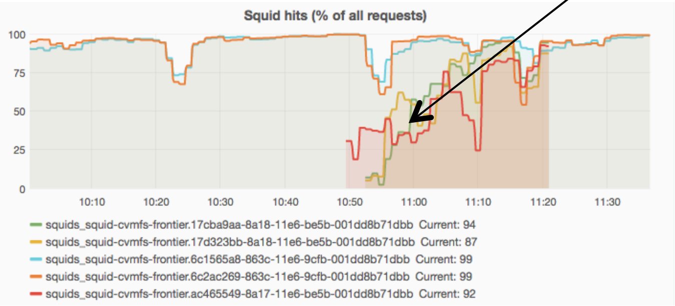


Spike in request rate triggers creation of additional squid instances

Drop in request rate therefore number of squid instances is reduced

Squid caches warming up





Summary & future plans

- The use of containers & container orchestration seems to have many benefits compared to our existing infrastructure
  - potentially higher availability with less effort & higher resource utilization
- Status of Mesos at RAL
  - used for testing & development; some non-critical internal services; a small fraction of batch jobs
  - not yet an "official" production service
    - it's a significant change in philosophy
    - the Tier-1 facility at RAL has to meet high SLAs, so moving away from a well-established infrastructure takes time
  - technologies initially looked at as part of the work on Mesos now being used more widely within the RAL Tier-1
    - Keepalived & HAProxy used in production in front of FTS3 for > 6 months
    - Telegraf, InfluxDB & Grafana now monitoring over 900 hosts

Summary & future plans

- Future plans include
  - increased integration with our configuration management system
    - move to using images created by Packer from configuration in Aquilon
  - using Ceph to allow containers to have persistent storage
  - investigate running OpenStack hypervisors in containers
    - will allow us to have cloud & batch sharing the same resources
  - contributions to INDIGO-DataCloud

Questions?