Science & Technology Facilities Council

Deploying services with Mesos at a WLCG Tier 1

Andrew Lahiff, Ian Collier

HEPIX Spring 2016 Workshop, DESY Zeuthen

Overview

- Services at RAL
- Mesos
- Example migration of a service from physical hosts to Mesos
- Migrating other services
- Some impressions
- What's next

Services at RAL

- Long-running production services are deployed on multiple platforms
  - Bare metal, 2008 Hyper-V, 2012 Hyper-V (soon), cloud (soon)
- Configuration management greatly simplifies deployment & management of services, but there's a lack of automation
  - Manually decide on which platform & host to deploy
  - Deployment & upgrades still involve many manual steps
  - Very static environment
    - e.g. many manual steps required to scale horizontally
  - In case of problems with individual machines
    - manual intervention required to restore services

Services at RAL

- Changing landscape
  - More & more communities (non-LHC) as well as local facilities becoming important
    - likely to need to run additional services
  - Staff effort more likely to decrease than increase
- Important to
  - Reduce the amount of effort required to manage services without affecting availability (ideally improving it)
  - Reduce number of out-of-hours interventions
  - Be more agile and adaptive to changing conditions
  - Maximize utilization of resources
- Others must have similar problems – what's happening in the wider world?

Services at RAL

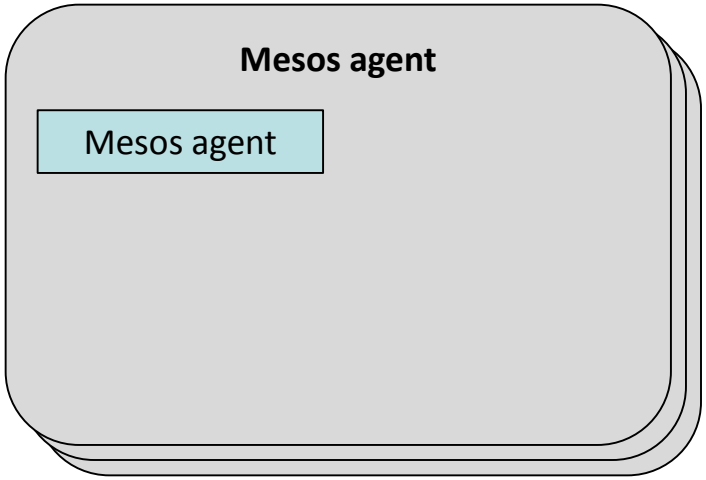
- One solution is to make significant & fundamental changes to the way we run services
  - **Manage applications using a scheduler**
    - would allow us to automate most of what we do manually today *(see later)*
    - possibility for application-aware scheduling, e.g. a scheduler which knows how an Elasticsearch cluster should be managed
  - **Run applications in containers**
    - removes the dependency between applications & hosts
    - enables applications to be quickly started anywhere
    - allows for isolation between different applications

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
  - abstracts away the concept of individual machines
- Used by an increasing number of both small & large organisations for reasons including
  - improving resource utilisation: have multiple distributed systems sharing the same resources
  - providing a self-healing fault-tolerant infrastructure
  - scalable to 10,000's nodes

Architecture at RAL

5 x SL7 VMs

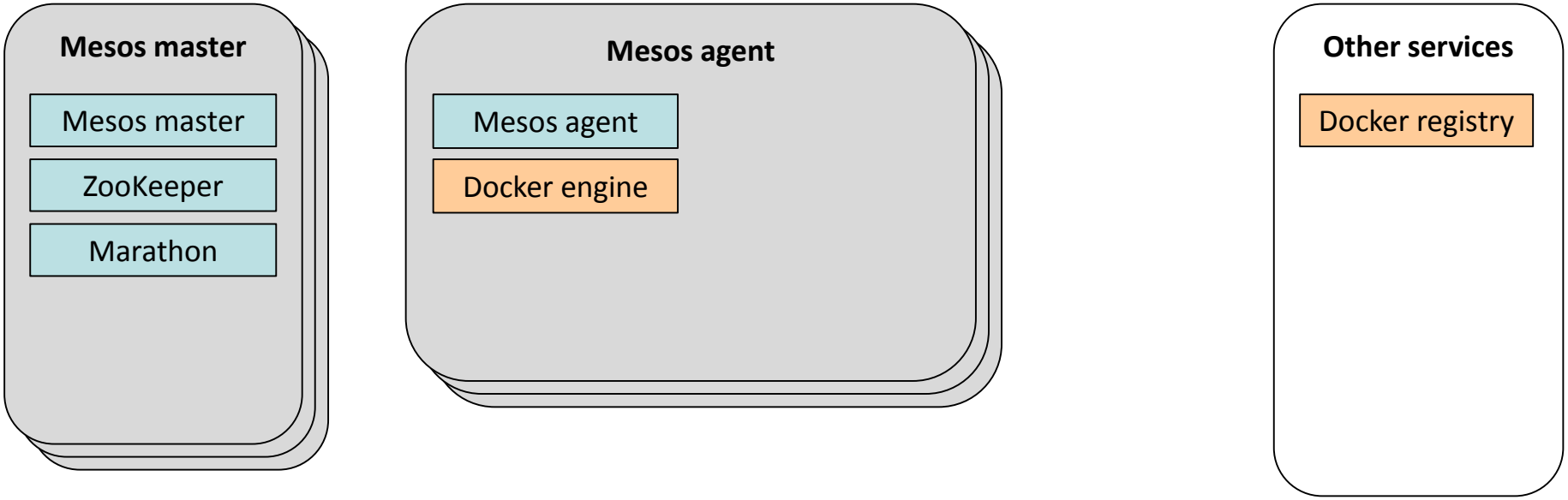


16 x bare metal: SL7, 16 cores, 24 GB RAM

- **ZooKeeper**: used for leader election & distributed coordination
- **Mesos masters**: in control of the cluster; offer resources to schedulers
- **Marathon**: distributed “init” for long-running services (a Mesos framework)
- **Mesos agents**: provide resources & run tasks

orchestration

Architecture at RAL

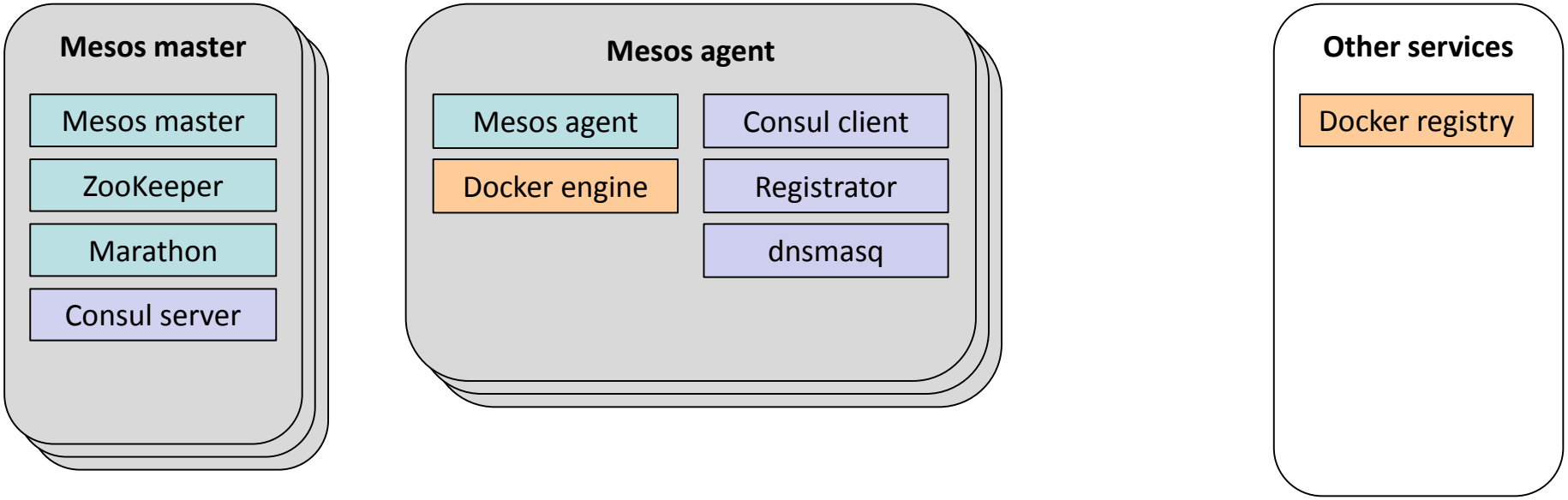


- **Docker engine**: allows each Mesos agent to run Docker containers
- **Docker registry**: local (private) image store

orchestration

containers

Architecture at RAL



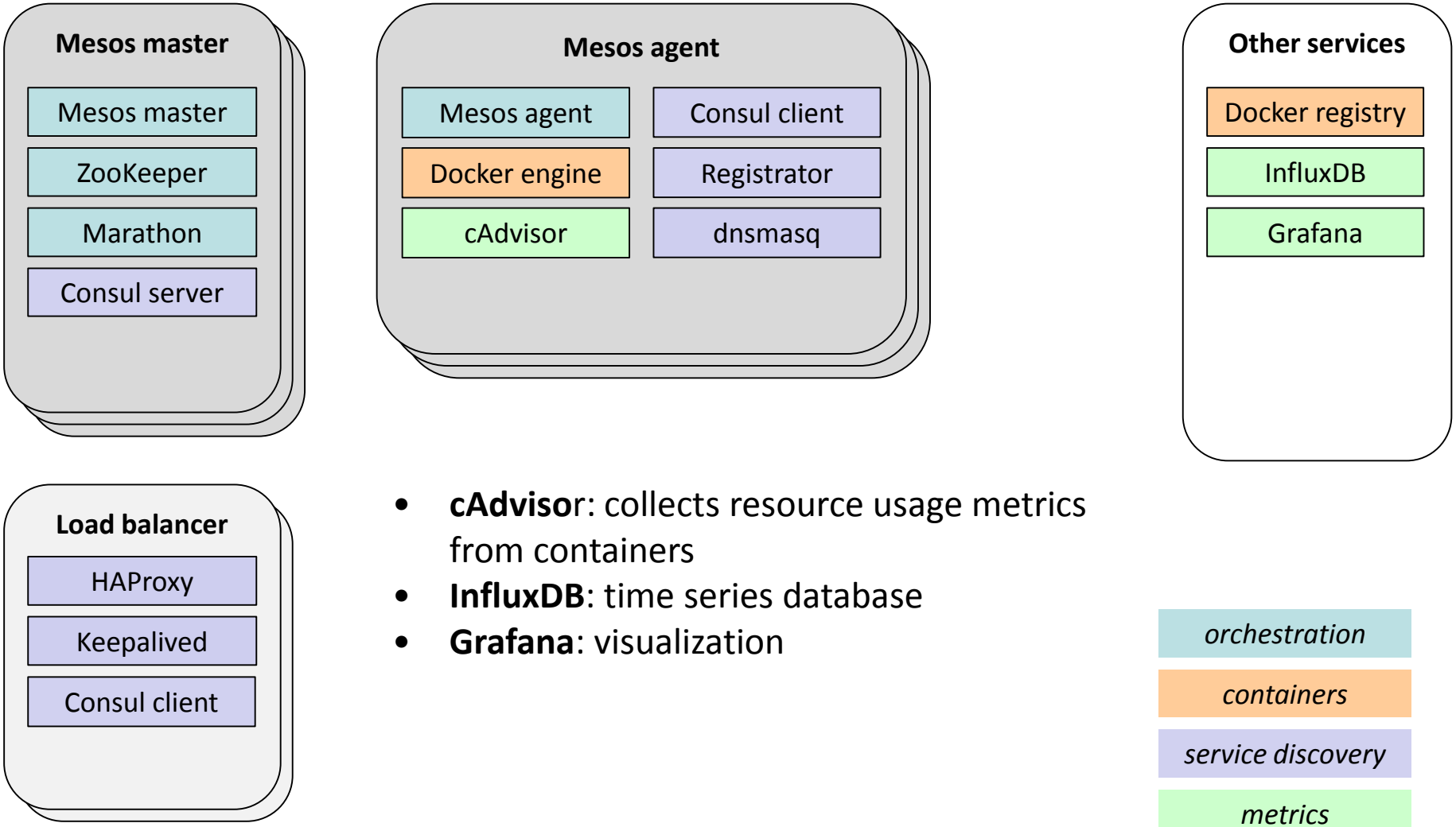
- **Consul**: distributed tool for service discovery
- **Registrator**: registers services provided by Docker containers with Consul
- **dnsmasq**: allows containers to access Consul's DNS interface
- **HAProxy**: load balancing, dynamically updated by Consul & made highly available by **Keepalived**

orchestration

containers

service discovery

Architecture at RAL



- **cAdviso**r: collects resource usage metrics from containers
- **InfluxDB**: time series database
- **Grafana**: visualization

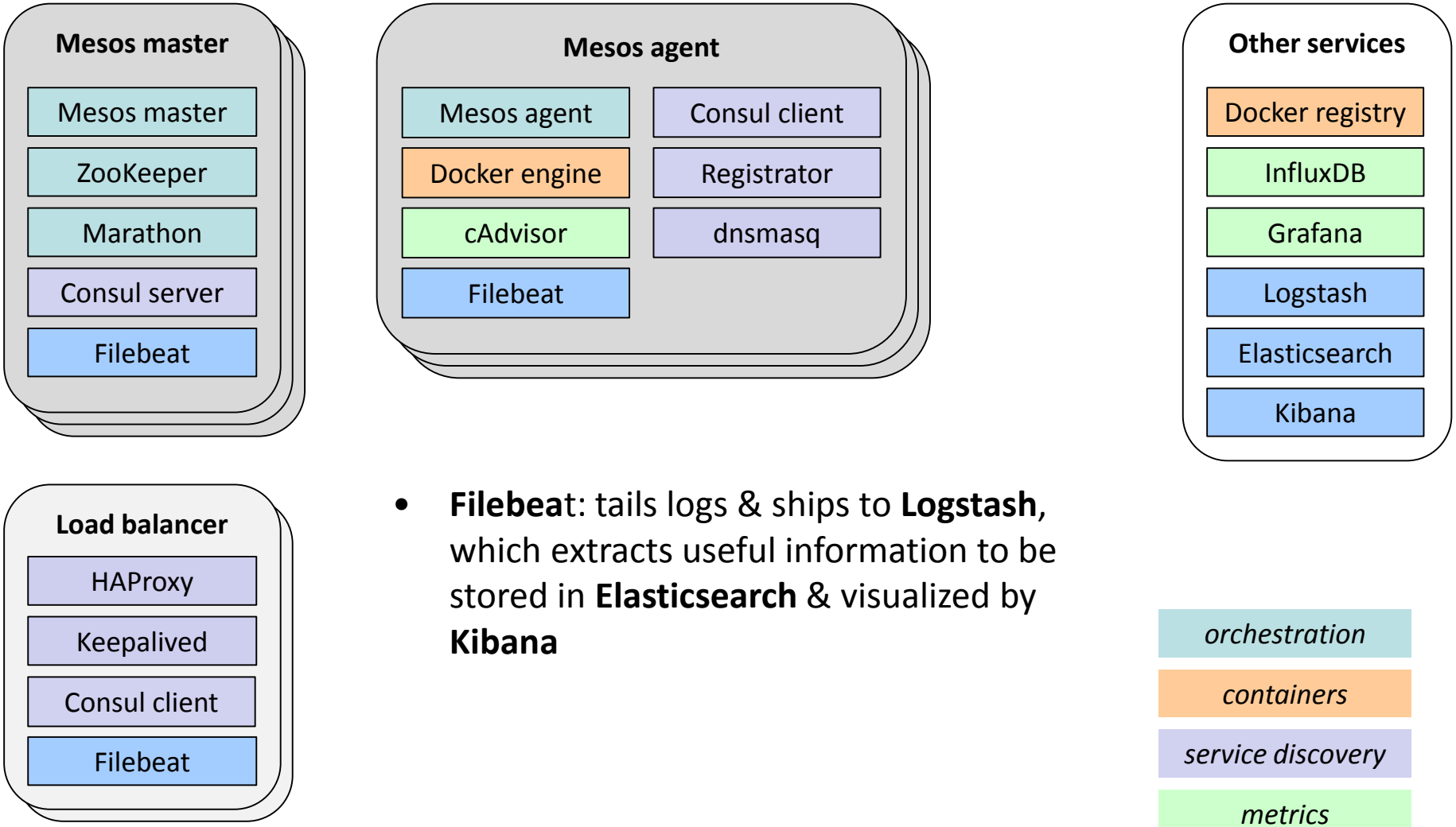
orchestration

containers

service discovery

metrics

Architecture at RAL



- **Filebeat**: tails logs & ships to **Logstash**, which extracts useful information to be stored in **Elasticsearch** & visualized by **Kibana**

orchestration

containers

service discovery

metrics

logging

Configuration management vs scheduling

- Could deploy parts of the infrastructure itself using Marathon
  - install just Mesos and Docker engine on each Mesos agent
  - all other services needed (metrics, logging, service discovery) run via Marathon instead of installing RPMs
- Some benefits
  - e.g. Marathon would ensure everything is running, could do rolling upgrades, etc.
- Some potential problems
  - core services start to become dependent on scheduling
- Currently have separation between infrastructure & applications
  - Infrastructure (orchestration, service discovery, monitoring, logging)
    - Entirely configured & deployed using Aquilon (Quattor)
  - Applications run on top of this infrastructure
    - They are managed by Marathon, for example
  - We're currently cautious & unlikely to move from this any time soon

Example service: top BDII

- Simple example of grid middleware
  - no host certificate required
- Very similar to all other services at RAL in terms of
  - deployment
  - configuration
  - alerts
  - monitoring
  - how external “users” access it
  - how failures are handled
  - how upgrades are handled
  - …

Current production service

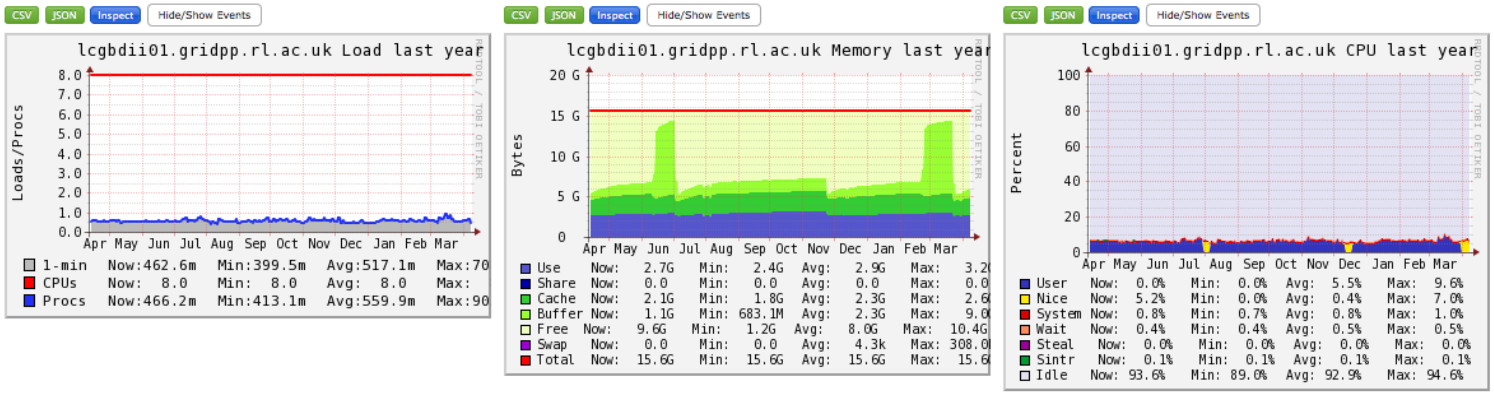
- Current top BDII production service at RAL
  - 3 physical machines: 8 cores, 16 GB
  - Site firewall hole for each machine
  - Round-robin DNS alias
- Nagios tests
  - 24 hour pager alarms for each host
    - top BDII service check
    - host checks (load, disk space, read-only filesystem, host down)
- Custom restarter script on each host
- Metrics
  - standard Ganglia metrics only

Limitations & issues

- If a machine dies or has problems overnight
  - still in DNS alias, so some % of requests will fail until fixed
  - pager alarm triggered, someone will try to fix it
- What if the 3 existing machines can no longer cope with the load?
  - there are a number of manual steps
    - request IP addresses for appropriate hostname(s)
    - deploy machine(s)
    - request site firewall hole(s) be added
    - request change to DNS alias
  - this is a very slow response

Limitations & issues

- Upgrades
  - software/OS upgrades done manually on a rolling basis
    - we use a configuration management system, but there's no facility for orchestration
  - when machines rebooted, some % of requests will fail
    - due to use of simple DNS alias
- Low utilization of resources



Load, memory & CPU usage over past year for one of the 3 top BDII hosts

Migration to Mesos

- What needs to be done to migrate a top BDII to Mesos?
  - Need a top BDII Docker image
  - JSON config for Marathon, specifies:
    - image, resources required, port(s), health checks
    - any contstraints, e.g. each instance should be on a different node or rack
  - Health checks
    - ideally should be built into the image & written by the developers of the application
    - either via response code from a http endpoint (e.g. /health) or exit code of a script inside the container
  - Site firewall holes
    - Needed for floating IP addresses only

Self-healing & fault tolerance

- Once running under Marathon, will have improved service availability with less effort
  - if application dies, it will be restarted
  - if a machine dies, applications running on it will be restarted somewhere else
- Loss of Mesos masters has no affect (provided quorum is maintained)

A Mesos agent host shutdown – task restarted automatically on another host

Task died – restarted automatically (not necessarily on the same host)

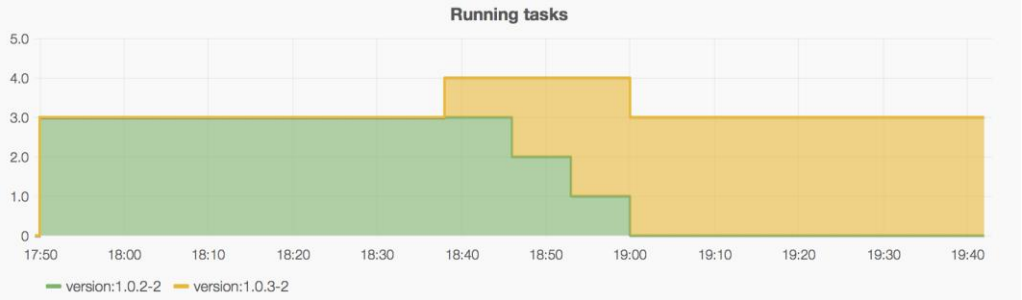
A Mesos master shutdown – no effect

Rolling upgrades

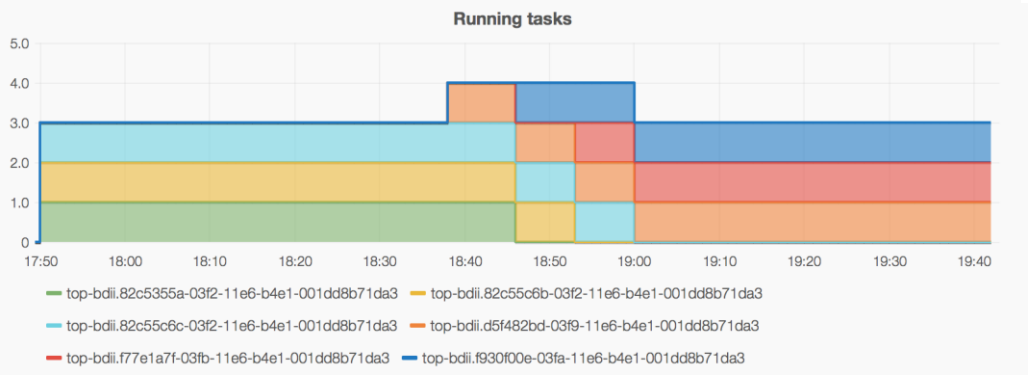
- Example of an automated rolling upgrade
  - running containers are not upgraded, they are replaced
  - old instances killed only when new ones become healthy
  - configurable upgrade policies in Marathon

In this example Marathon ensures that at least 100% of the capacity is maintained during the upgrade.

Marathon waits for health checks to become successful before continuing the rolling upgrade.



Number of containers running each version



Each colour represents a different container

Alerts – applications

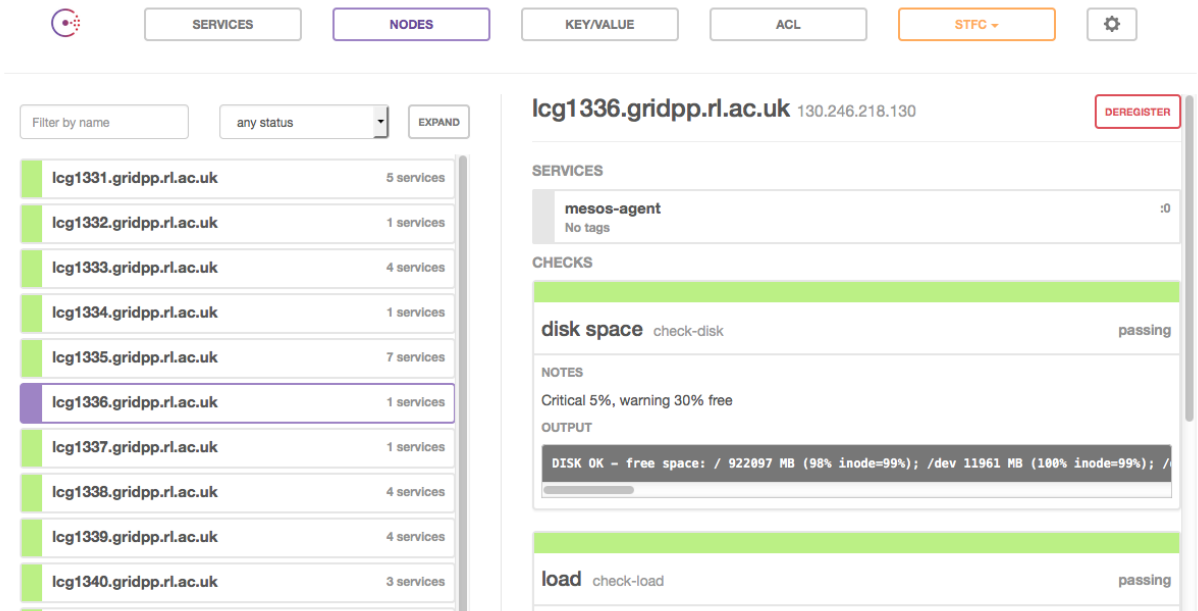
- General ideas
  - no longer need to worry about individual hosts or instances of applications
  - only need to worry if any problems will be visible to users
- In practice
  - On a Nagios server
    - Check that the floating IP address is alive (basic TCP test)
  - Load balancer hosts
    - Nagios check that the number of healthy backend servers is above a minimum threshold
    - (Assumes that the health check used does provide a good indication whether the application is working or not!)

Alerts – infrastructure

- Standard Nagios tests
  - Mesos masters
    - minimum number of healthy ZooKeeper servers, Mesos masters & Consul servers (i.e. quorum is maintained)
    - Marathon is functional & has a leader
  - Mesos agents
    - minimum number of healthy agents
    - maximum percentage of resources usage
  - Load balancer
    - checks for HAProxy, Keepalived
- No callouts on any individual Mesos agents
  - Nothing should depend on a specific Mesos agent being alive or even be aware of the hostnames

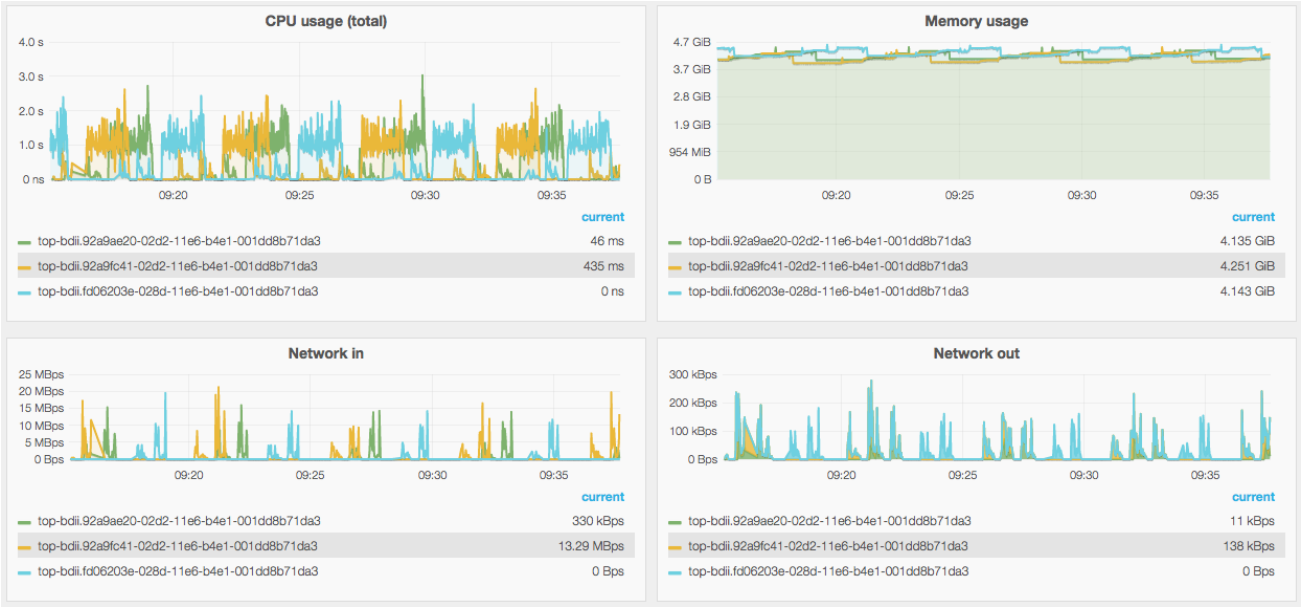
Infrastructure monitoring

- Consul on each Mesos agent can run standard Nagios checks
  - Services running on them will automatically be unhealthy & therefore not accessible
  - Consul can also provide Mesos masters with a white list of healthy Mesos agents



Metrics

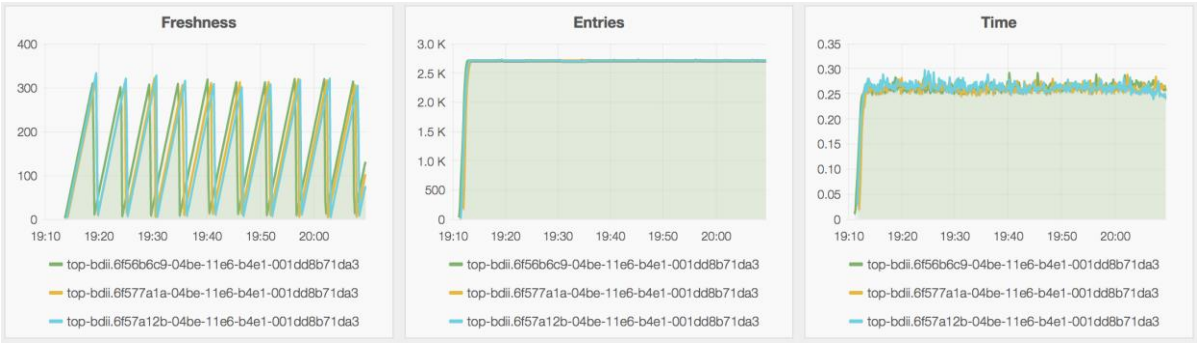
- Can easily view all instances of a particular application
  - irrespective of how many instances there are or what hosts they're running on
  - note that it wouldn't be possible to do this with Ganglia



Top BDII resource usage metrics collected by cAdvisor, stored in InfluxDB & visualized by Grafana

Metrics

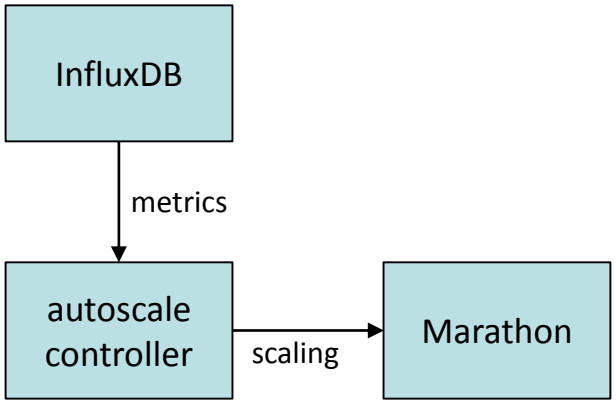
- What about application metrics?
  - A process in the container which sends metrics somewhere?
    - Problem: site specific, assumptions about monitoring technology used
  - A process in the container which makes metrics available on a http endpoint (e.g. /metrics)?
    - Probably better: metrics can be ‘scraped’



Top BDII application metrics collected by cAdvisor

Auto scaling

- Automatically scale capacity according to demand
  - VMs take minutes to start
    - this can be too long
  - Containers take seconds to start
    - can more quickly respond to spikes in demand
- Scaling based on metrics collected by cAdvisor
  - Could use resource usage, e.g. CPU, load, memory, network
  - And/or application metrics, e.g. request rate



Traceability & logging

- For traceability reasons, we keep for a short period of time container images & Mesos sandboxes
  - would allow us to investigate any issues, no matter how short-lived the container was
- Mesos & Marathon logs to central loggers & Elasticsearch
  - information about all containers
- External (e.g. user) access to services
  - HAProxy logs (to central loggers & Elasticsearch) include source IP, service accessed, host & container name

```
2016-04-16T14:45:54.281063+01:00 vm135 haproxy[14678]: 130.246.180.41:23937
[16/Apr/2016:14:45:48.624] top-bdii-test top-bdii-test/lcg1331.gridpp.rl.ac.uk:mesos-
a94a0d23-deb9-407f-8876-0c77f01a7fdf 1/0/5657 0 -- 1/0/0/0/0 0/0
```

Migrating other services

- More complex applications
  - typically have multiple sub services running in the same VM
    - problems with one can affect others in the same VM
    - frequently have multiple instances of all services, even if not needed
  - can be split into multiple containers
    - each container has a single purpose
    - container orchestration combined with dynamic service discovery makes this both possible & straightforward
- Stateful applications
  - Marathon supports persistent storage volumes (new)
    - external disk (using a Docker volume plugin)
    - local disk on each Mesos agent (tasks & their data are pinned to the node they first run on)

Some impressions

- Ideally there would be official releases of grid middleware as container images
  - even without orchestration, would make deployment easier for sites
- Getting ZooKeeper, Mesos & Marathon to work is relatively straightforward & works reliably
- Service discovery is where things get more complicated
  - many options available, no “perfect” solution (yet)
    - some use DNS, some use Consul, some involve HAProxy on every Mesos agent, ...
    - all have their pros & cons
- Similarly, there’s a variety of monitoring & logging options

What's next

- Mesos currently going through the internal RAL change control process in order to make it a production service
  - Prerequisite before running production services on Mesos
  - Making the case for benefits of investing in a very different way of doing things
- Looking into additional use cases, e.g.
  - dynamic Jenkins slaves for another part of the Scientific Computing Dept at RAL
  - INDIGO DataCloud pilot deployment
- Simplify creating & managing images
  - Later this year a graduate trainee will work on setting up a VM & container "image factory"
- Host certificates & other secrets
  - Need to be able to securely insert secrets into containers (& VMs)