

MESOS for LHCbDIRAC

Christophe HAEN
6th DIRAC Workshop
23-25 May 2016

Current situation

Host A



Host B



- Static installations
- Placement optimization problems
- Low availability
- Painful updates
- Risk of heterogeneity in the configuration

I have a dream

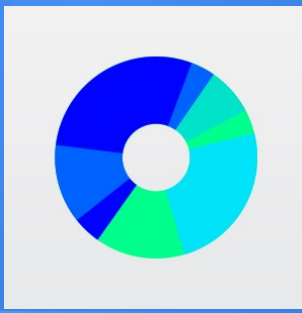


Let “something”
run it “somewhere”
for you



- Runs “tasks” on “slaves”
- “Slaves” have “resources” to offer (cpu, mem, etc)
- “Resources” are offered to “Frameworks”
- “Frameworks” contains your work description

Marathon



- Distributed init.d for long-running services
- Web + rest interface
- Placement constraints
- Easy scaling
- Rolling upgrades

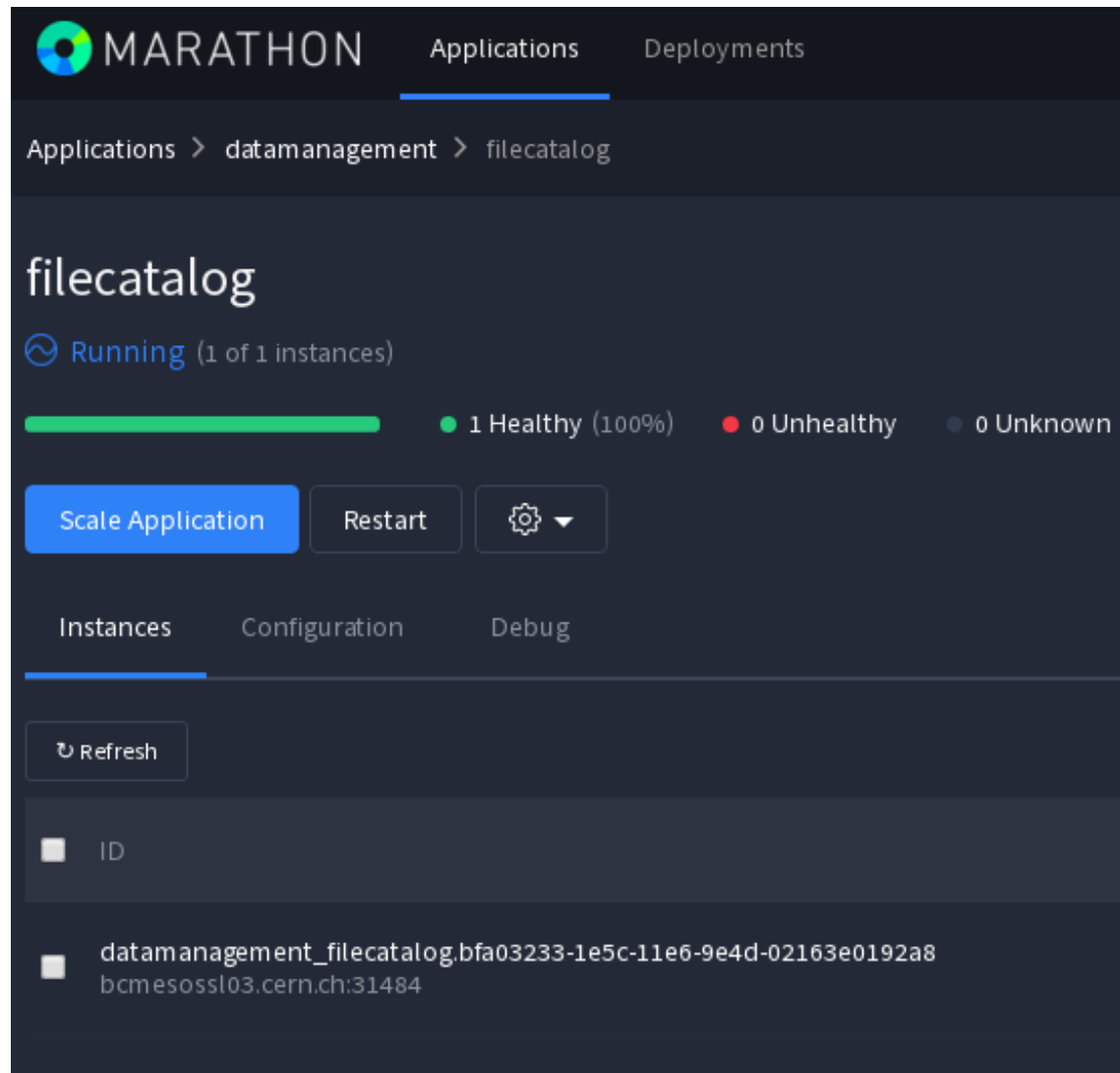
Easy !

- Make my VOBoxes mesos slaves
- Put LHCbDIRAC in a Docker image
- Describe my LHCbDIRAC installation as docker instances
- Give that to Marathon
- Have a beer

DFC in Marathon

```
{
  "id": "/datamanagement/filecatalog",
  "cpus": 0.8,
  "mem": 600,
  "instances": 1,
  "cmd": "dirac-service DataManagement/FileCatalog",
  "container": {
    "type": "DOCKER",
    "docker": {
      "image": "bcmesosms02:5000/registry/lhcbdirac:v8r2p44",
      "portMappings": [
        { "containerPort": 9197, "hostPort": 0 }
      ]
    }
  }
}
```

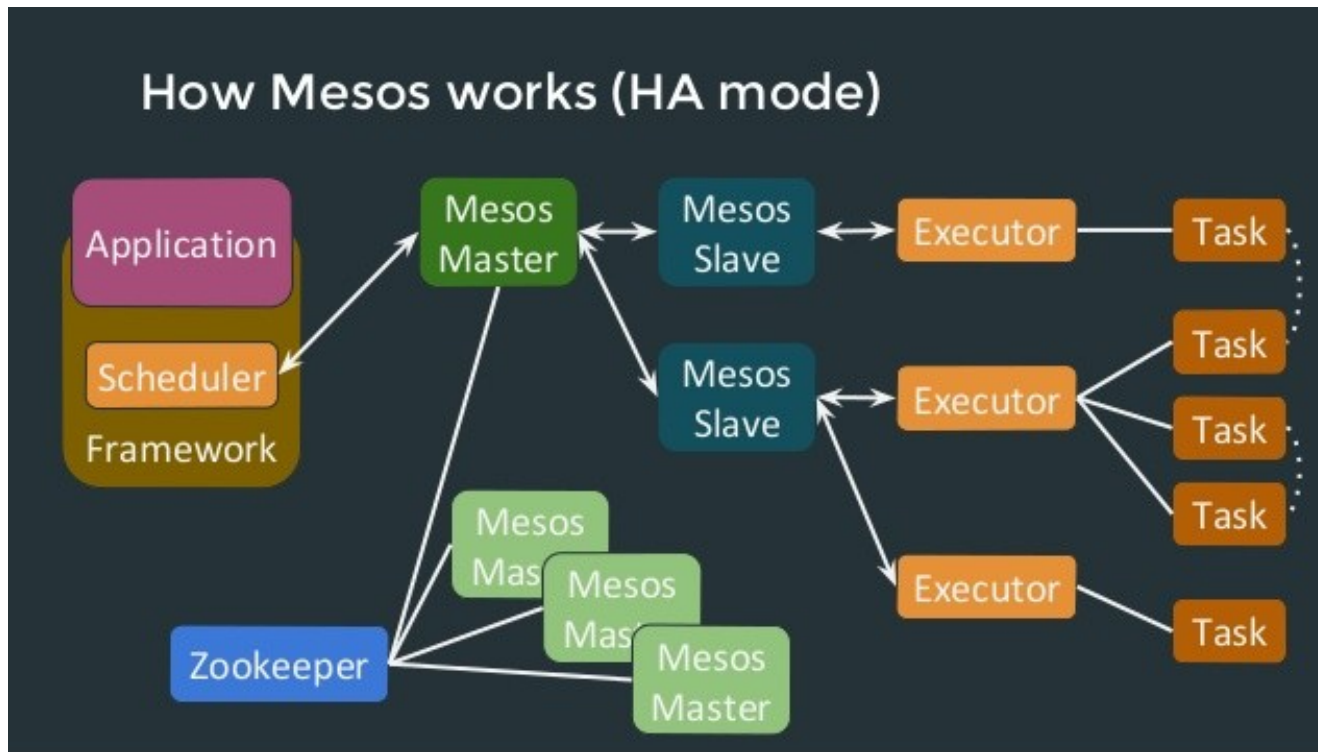
DFC in Marathon



The screenshot displays the Marathon web interface for the 'filecatalog' application. At the top, the 'MARATHON' logo is visible, along with navigation tabs for 'Applications' and 'Deployments'. The breadcrumb path is 'Applications > datamanagement > filecatalog'. The application name 'filecatalog' is prominently displayed, followed by its status 'Running (1 of 1 instances)'. A progress bar indicates 100% health, with 1 Healthy instance, 0 Unhealthy instances, and 0 Unknown instances. Below this, there are three buttons: 'Scale Application' (highlighted in blue), 'Restart', and a settings gear icon. A secondary navigation bar includes 'Instances', 'Configuration', and 'Debug' tabs, with 'Instances' being the active tab. A 'Refresh' button is located below the tabs. The main content area shows a table with one instance:

ID
datamanagement_filecatalog.bfa03233-1e5c-11e6-9e4d-02163e0192a8 bcmesoss103.cern.ch:31484

Clusterize the master



- Zookeeper
- Several masters
- Choose a leader
- Quorum decision
- Failover
- Also for Marathon !

We are done !



Well, not really yet...

Service discovery

- What is running ?
- Where is it ?
- How do I access it ?

No really satisfactory answer from Mesos/Marathon yet

Consul



- Agents running on every nodes (server or client mode)
- Health check (see later)
- DNS functionality
 - e.g. filecatalog.datamanagement.service.consul
 - Only works within the Consul cluster

Consul



Filter by name any status ▼ EXPAND

bcmesosms01.cern.ch	2 services
bcmesosms02.cern.ch	2 services
bcmesosms03.cern.ch	2 services
bcmesossl01.cern.ch	4 services
bcmesossl02.cern.ch	4 services
bcmesossl03.cern.ch	4 services

bcmesossl03.cern.ch 188.184.84.243

DEREGISTER

SERVICES

cadvisor No tags	188.184.84.243:31420
filecatalog-datamanagement No tags	188.184.84.243:31484
ftsmanager-datamanagement No tags	188.184.84.243:31993
mesos agent follower	188.184.84.243:5051

CHECKS

Serf Health Status serfHealth passing		
NOTES		
OUTPUT	Agent alive and reachable	

Service 'filecatalog-datamanagement' check service:mesos-consul:188.184.84.243:filecatalog-datamanagement:31484 passing		
NOTES		
OUTPUT	HTTP GET http://bcmesosms02:1234/ping?host=188.184.84.243&port=31484&service=DataManagement/FileCatalog	

Consul



- Need to register your services in Consul
 - Registrator
 - **Mesos-consul** (might change my mind)
- Still need to use the info !
 - Consul-template: go templating language

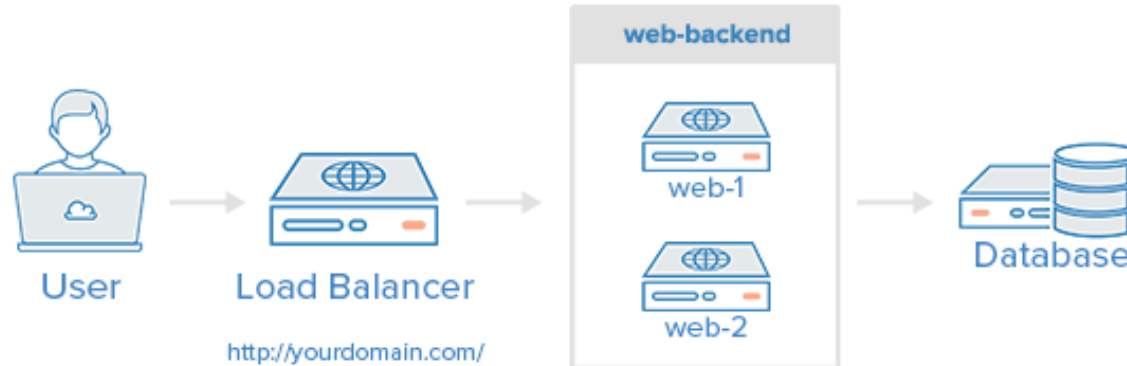


HAPROXY

Powering Your Uptime

- High availability load balancer
- Proxy server
- TCP & HTTP
- SSL, keep-alive, compression, header rewriting, etc
- Scale for ever
- Used everywhere

Layer 4 Load Balancing





HAPROXY

Powering Your Uptime

HAProxy.ctmpl

```
listen DataManagement_FileCatalog
  bind *:9197 {{range service "filecatalog-datamanagement" }}
    server {{.Node}} {{.Address}}:{{.Port}}{{end}}
```

HAProxy.cfg

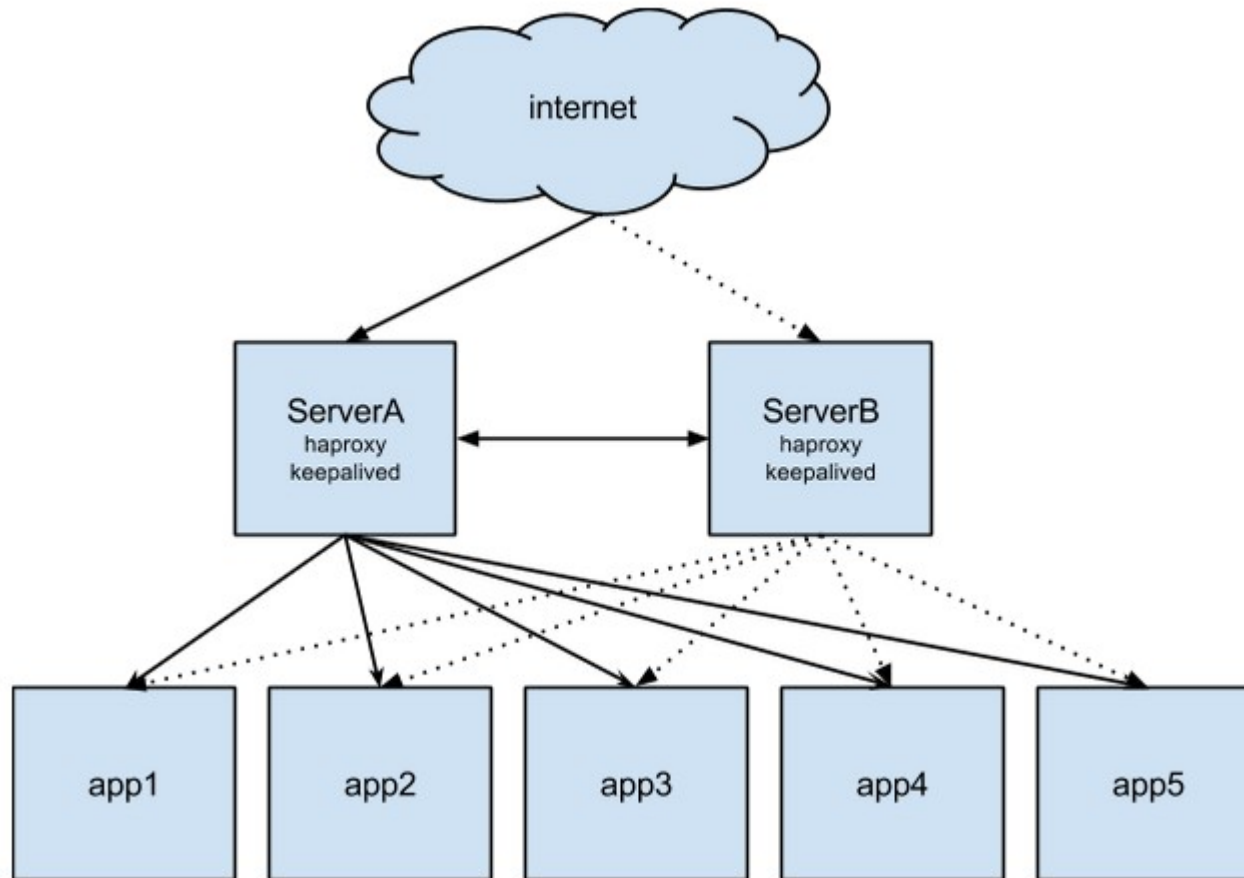
```
listen DataManagement_FileCatalog
  bind *:9197
  server bcmesossl03.cern.ch 188.184.84.243:31484
  server bcmesossl01.cern.ch 188.184.84.241:32595
```

- A single entry in the CS for URL
- Could even think of a global “default server”



HAPROXY

Powering Your Uptime



Health monitoring: Marathon

- Failing containers are watched
- Monitor the behavior of the container:
 - HTTP
 - TCP
 - Command (a bit tricky): `dirac-self-ping.py`

```
"healthChecks": [  
  {  
    "protocol": "COMMAND",  
    "command": { "value": "python /opt/dirac/dirac_self_ping.py 9197/DataManagement/FileCatalog"},  
    "intervalSeconds": 10,  
    "maxConsecutiveFailures": 2,  
    "timeoutSeconds": 10  
  }  
]
```

Health monitoring: Consul

- Unhealthy entities not returned when querying Consul
- Host monitoring:
 - Standard Nagios plugins
 - Defined on each slave
 - Generate Mesos slave whitelist with consul-template

```
{  
  "id": "check-disk",  
  "name": "disk space",  
  "notes": "Critical 5%, warning 30% free",  
  "script": "/usr/lib64/nagios/plugins/check_disk -w 30% -c 5%",  
  "interval": "1m"  
},
```

Health monitoring: Consul

- Service monitoring:
 - HTTP, TTL, TCP, Docker
 - Meet “The Pinger”: LHCbDIRAC container, listening to http, performing dirac ping on demand
 - Updates the HAProxy conf
 - Conf defined in marathon task json

```
"labels": {  
  "check_http" : "http://bcmesosms02:1234/ping?host={host}&port={port}&service=DataManagement/FileCatalog",  
  "check_interval": "10s",  
}
```

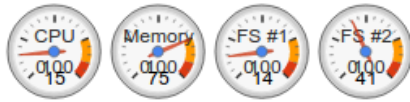
Performance monitoring

- Resource utilization
- Useful to write marathon cpu & mem entries
- Containers & hosts
- cAdvisor:
 - Made by google
 - One instance per slave (can be run with Mesos)
 - Monitor hosts and discover containers
 - Web interface and rest API

cAdvisor

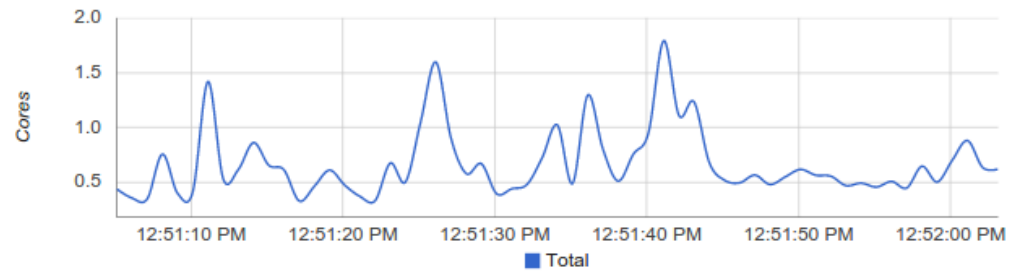


Overview

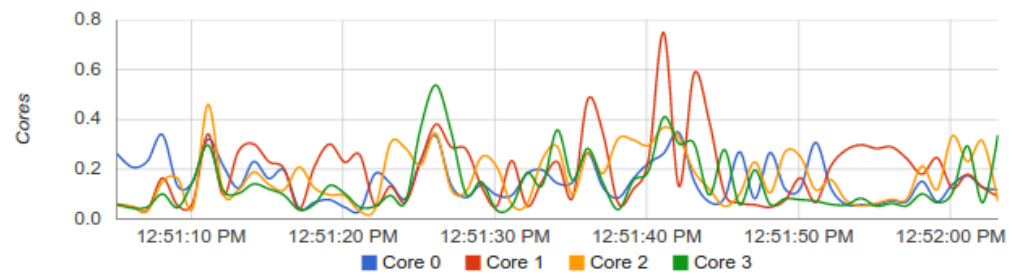


CPU

Total Usage



Usage per Core

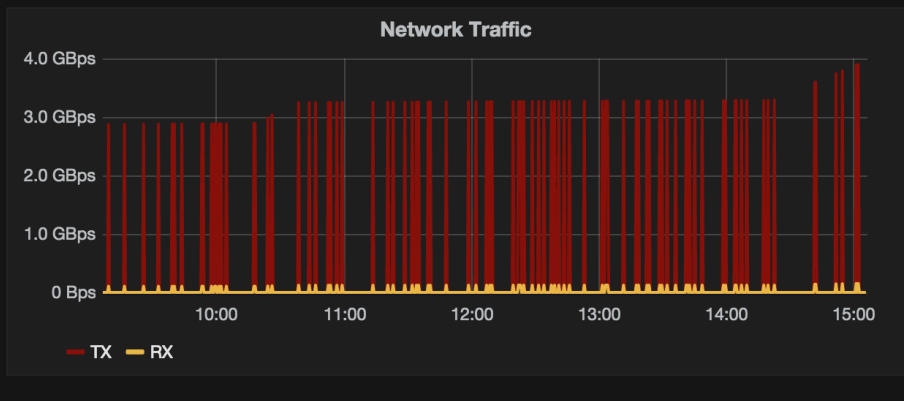
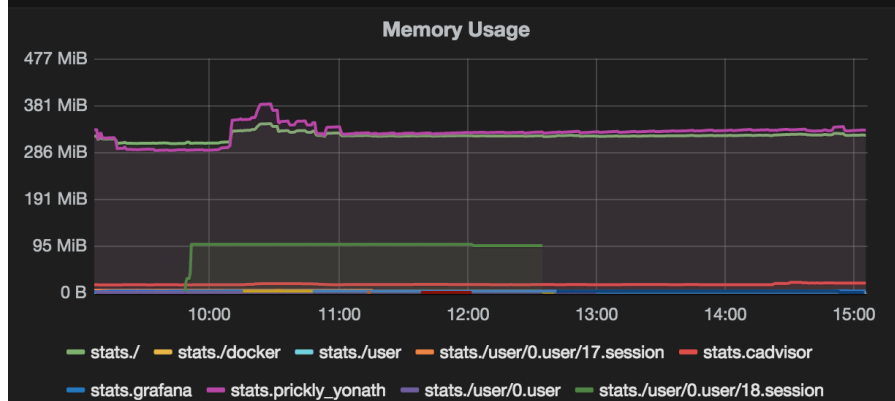
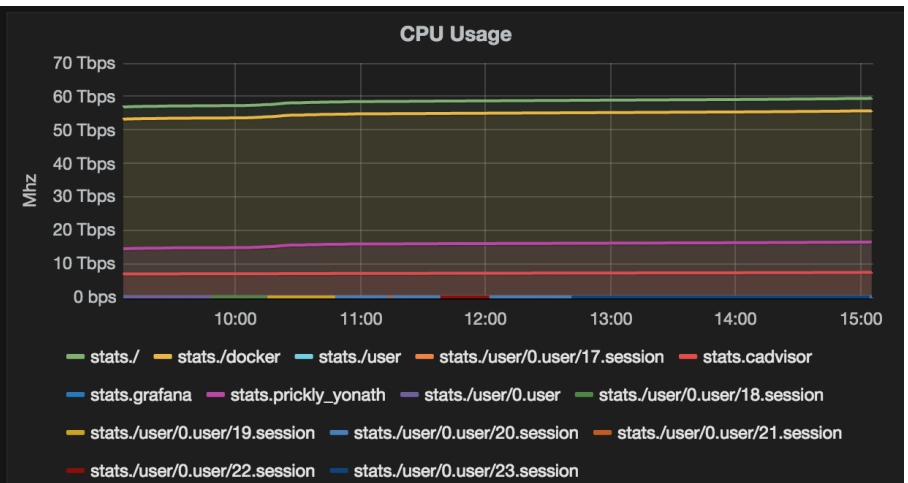
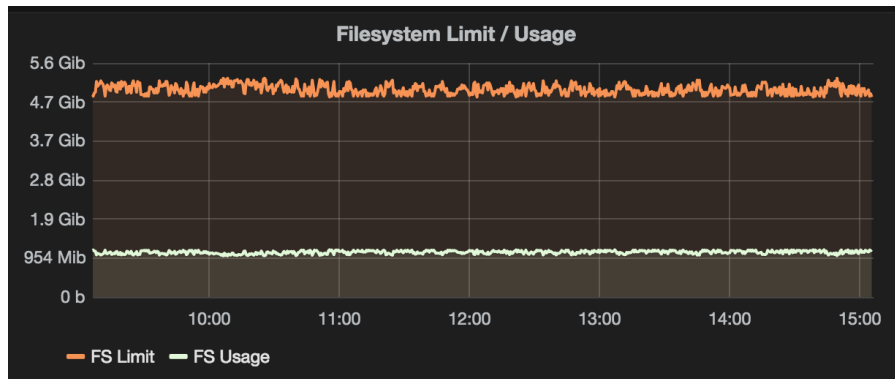


Usage Breakdown

And then...

- Aggregates all the cAdvisor
 - Heapster
 - Homemade tools
- InfluxDB
- Grafana

And then...



Autoscaling

- Existing tools:
 - Very primitive
 - Based on CPU/Mem from Mesos stats
 - Based on HAProxy counters
- Write your own from InfluxDB

Logging

- Here, it hurts... badly
- Can get stdout & stderr (mesos, marathon, docker)
- Can tail -f, grep, ... (docker)
- **We need a central logging**
 - Quickly :-)
 - Ideally, directly from the code (json + message queue + NoSQL db ?)
 - Poor's man solution for the time being

ELK stack

- ElasticSearch + LogStash + Kibana
- Forward all docker logs to logstash
 - Docker-gen: generates filebeat configuration
 - Filebeat: actually forward the messages
- We can't write logstash parses for all our logs

Logstash for services

```
DLOGLEVEL ALWAYS|NOTICE|INFO|VERB|DEBUG|WARN|ERROR|EXCEPT|FATAL
DIRAC_RET OK|ERROR
DIRAC_CALLER \[(?:(hosts:%{HOSTNAME:dirac_host})|(%{WORD:dirac_group}\:%{WORD:dirac_user}))\]
DIRAC_CALL_SRC \(\[%{IPORHOST:client_src}\]:%{NUMBER:client_port}\)%{DIRAC_CALLER}
DIRAC_LOG_START %{TIMESTAMP_ISO8601} %{WORD}
                (?<service>\w+(\w+)+).* %{DLOGLEVEL:log_level}:
SERVICE_QUERY %{DIRAC_LOG_START} Executing action
                %{DIRAC_CALL_SRC} RPC/%{WORD:dirac_method}\(<masked>\)
SERVICE_ANSWER %{DIRAC_LOG_START} Returning response %{DIRAC_CALL_SRC}
                \(%{NUMBER:response_time} secs\)
                %{DIRAC_RET:ret_status}(: %{GREEDYDATA:message})*
```

```
# 2016-04-22 11:10:32 UTC DataManagement/FileCatalog[PzMY] NOTICE: Returning
response ([::ffff:128.142.132.74]:58269)[diracAdmin:chaen] (0.01 secs) OK
# 2016-04-22 12:10:25 UTC DataManagement/FileCatalog[PzMY] NOTICE: Returning
response ([::ffff:128.142.132.74]:36970)[diracAdmin:chaen] (0.00 secs) ERROR:
Unknown method fakeMethod
```

Logstash for services

```
{
  "_source": {
    "log": "2016-05-20 17:18:05 UTC DataManagement/FileCatalog
          NOTICE: Executing action ([:ffff:128.142.132.74]:59373)
          [diracAdmin:chaen] RPC/fakeMethod(<masked>)\n",
    "time": "2016-05-20T17:18:48.985791214Z",
    "fields": {
      "dirac_component": "DataManagement/FileCatalog",
      "image_name": "registry/lhcbdirac"
    },
    "host": "bcmesoss103.cern.ch",
    "service": "DataManagement/FileCatalog",
    "log_level": "NOTICE",
    "client_src": "[:ffff:128.142.132.74]",
    "client_port": "59373",
    "dirac_group": "diracAdmin",
    "dirac_user": "chaen",
    "dirac_method": "fakeMethod"
  }
}
```

Where are we after 5 weeks ?

- Test bench
- I have “a” docker image
- 3 masters of everything with quorum
- 5 slaves
- **A running DMS and RMS**
- Some logging and monitoring

Master machines

- Mesos server
- Marathon server
- Zookeeper
- Consul server
- Consul-template
- (Haproxy)

Slave machines

- Docker
- Mesos slave
- Consul client
- Docker-gen + filebeat
- cAdvisor (can be in Mesos)
- nagios-plugins

Somewhere

- mesos-consul
- ELK stack
- Docker registry
- cAdvisor aggregator + influxDB + grafana

Pros

- System administrator → geh weg !!
- New release to rollout ? 1 click
- Scalling ? 1 click or automatic
- High availability
- No need to bother about placement
- Full control on the env
- Heterogeneity less of a problem
- Run parallel setups easily (certification, multi-VO, etc)
- Not only for LHCbDIRAC !

Does not solve everything

- You can blow up everything :-)
- Easy to jump in, but requires a bit of education
- Need to add the configuration in the CS by hand when adding new elements
- Problems inherent to docker (data storage)
 - CS
 - DIRAC SE

Still lot of work

- Logging, Logging, Logging, Logging
- Authorization and authentication of the management
- Agents ? Probably Chronos framework
- Find a proper strategy
- Squeeze the image
- Test, test, test
- ...
- Definitely the way to go, but still many degrees of freedom

LHCbDIRAC docker

- Dockerfiles are not dynamic
- Strong coupling between versions (DIRAC, externals, bundle, extensions)
- 2 options:
 - All in one dockerfile
 - Inheritance
- Currently, brutally **EVERYTHING** in one → uncool

LHCbDIRAC docker

- All in one dockerfile:
 - Easy to read and manage
 - Duplicates between extensions
 - Can contain useless things
- Inheritance:
 - Factorize
 - No multiple inheritance :-)

LHCbDIRAC docker

- Idea:
 - Externals + bundles – (web + plotting)
 - DIRAC: Externals
 - WebDIRAC: DIRAC + web + plotting
 - LHCbDIRAC: DIRAC
 - LHCbWebDIRAC: WebDIRAC
- Results in 2 images in Mesos

LHCbDIRAC docker

- CRLs and CAs ? Mounted volume !
 - Easier to manage, spot problems
- Host certificate ? Mounted volume !
 - Easier to manage, safe, sufficient
- Dirac.cfg ? Mounted volume !
 - Easier to manage, multiple setup per host