# An opportunistic AliEn site with Plancton on the OCCAM HPC facility in Torino

Matteo Concas (Politecnico di Torino, DET)
matteo.concas@cern.ch
Dario Berzano (CERN)
dario.berzano@cern.ch

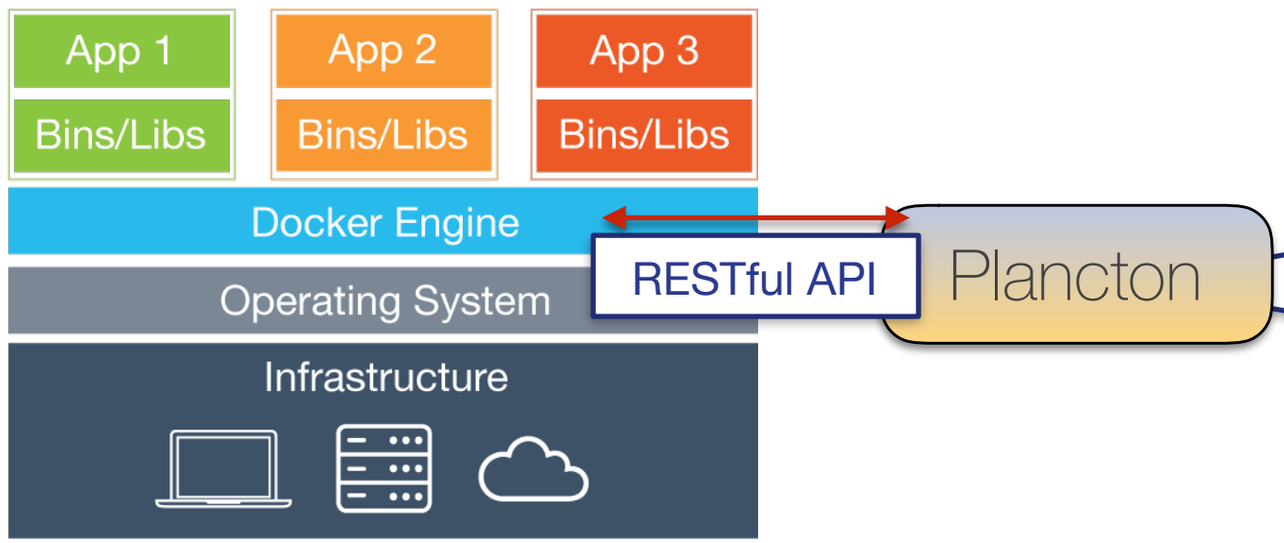
*Thursday - May 4th, 2017*

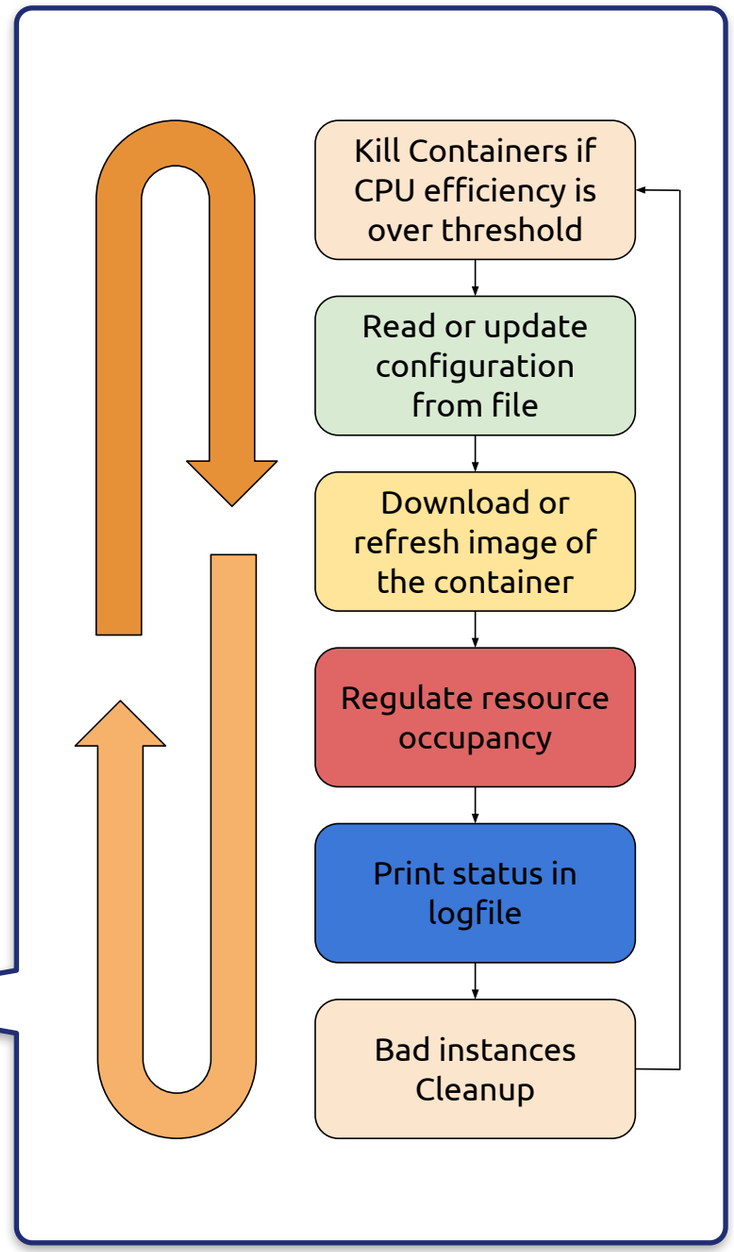
# What is Plancton?

- It works with Docker, a mainstream tool for container provisioning, easy to programmatically manage Linux containers
- A service that automatically and continuously spawns Docker containers according to the available resources (mconcas/plancton)
  - It runs as a **standalone** instance on each machine ➙ each daemon is independent from the others in a cluster
  - Plancton is **stateless**: daemon and its configuration can be updated without affecting running containers
  - **Rolling updates**: it is possible to gradually replace “old” containers with ones based on newer images as soon as they terminate
  - Effective command line **tools** to control running containers (drain mode, force stop, etc…)

# Plancton: a “simple” container scheduler

- Workflow
  - Check for **available resources** (CPU usage)
  - If enough resources available ➙ **start new container**
  - Other basic **checks** and **cleanup** (clean up exited containers, query status, etc…)
  - Check **overdue containers** (jobs beyond TTL…) and handle misbehaviours



# What Plancton is not?

- Plancton is not designed to replace more complex (and complete) orchestrators like **Kubernetes**, **Apache Mesos** or **Docker Swarm** ➙ as long as the use-case is “simple” we keep it simple

- It provides a **lightweight** and **easy** way to setup a computing cluster running "pilot" jobs without configuring a batch farm

- It is **application agnostic**: it does not care about the applications running inside, it does not interact nor manages them ➙ use cases are confined inside Docker containers

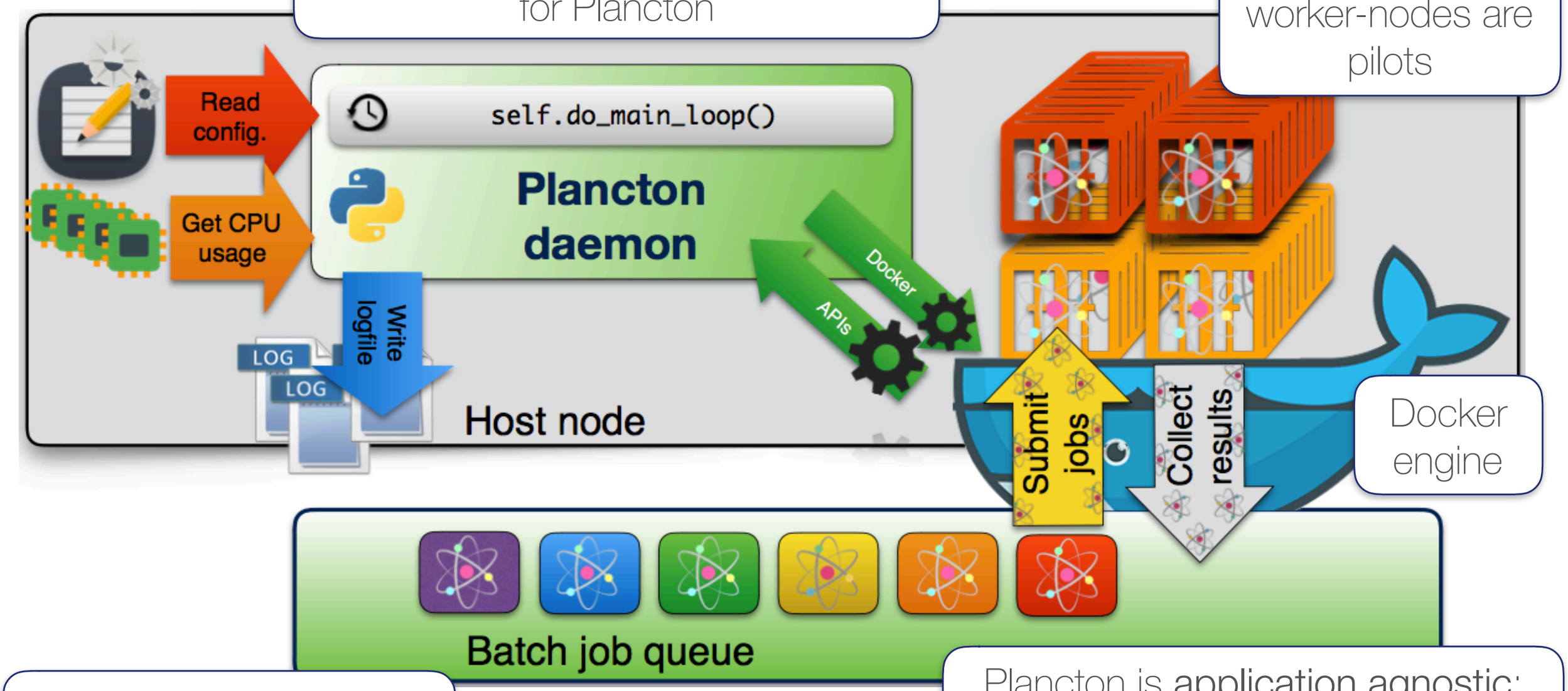
# From “pilot jobs” to “pilot containers”

- Short deployment time (~secs) and negligible overhead on launch: it is possible to **use containers as pilots** as it is inexpensive to continuously spawn them
  - The container runs a pilot **executable**: when the pilot dies, the container dies too. It starts services needed to attach the worker to the "batch" cluster (e.g. HTCondor, RabbitMQ, etc…)
  - They act like pilot jobs, the only difference is that the execution is “wrapped” into a correct environment, isolated from the rest of the process tree, that we throw away when the job is done
  - On a 24-cores machine we can run 24 single-core containers each one executing a job, with the possibility to optionally define resource limits, with a finer granularity

# Architecture

There is no centralised management for Plancton

Docker containers worker-nodes are pilots




Docker engine

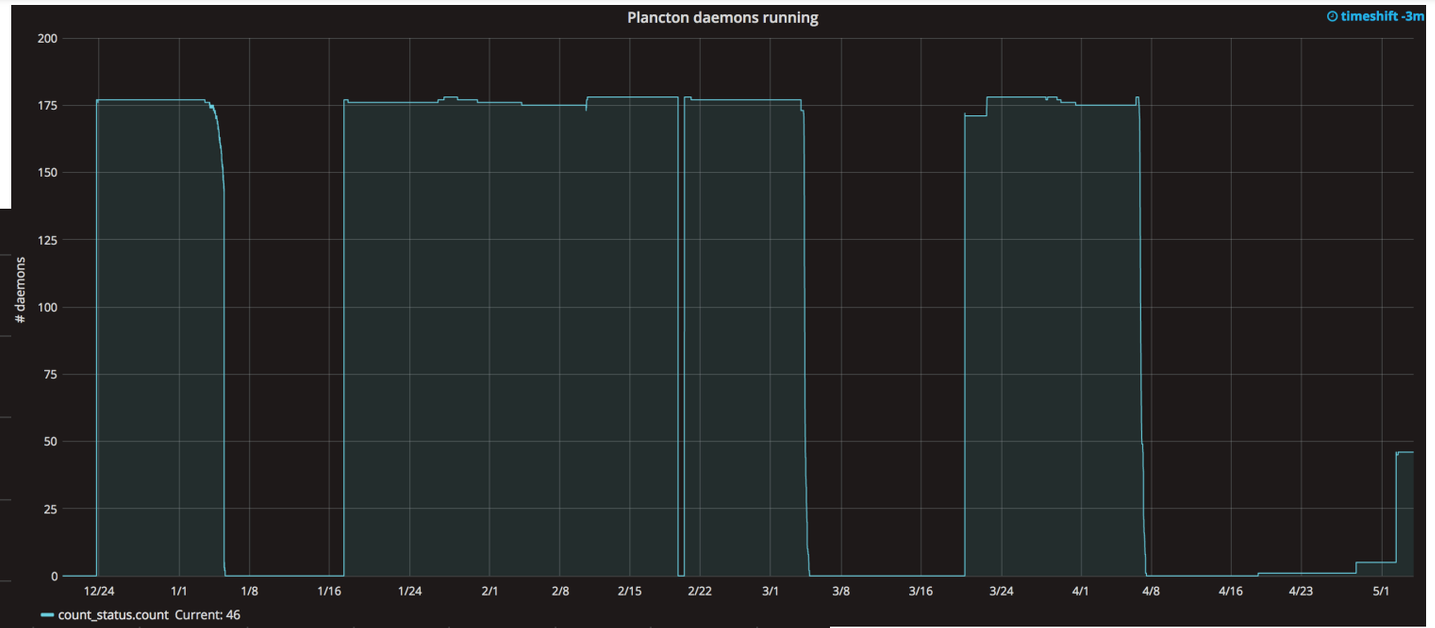
Users submit their job to local batch system: transparent to users

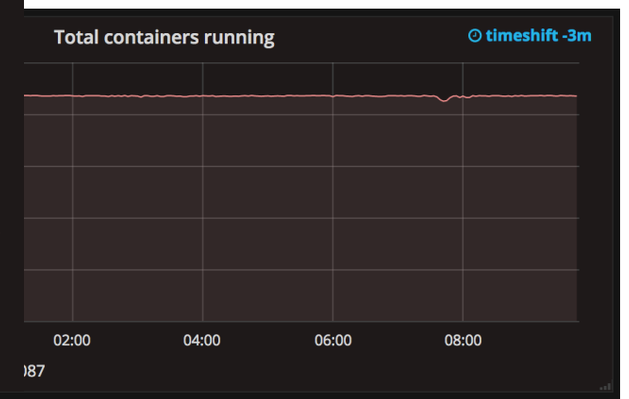
Plancton is **application agnostic**: It does not need to be aware of the job-scheduling application

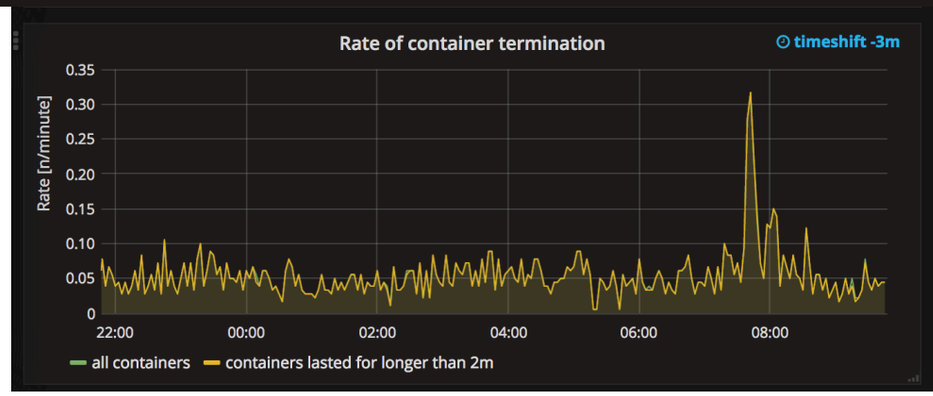
# Already existing successful implementations: HLT

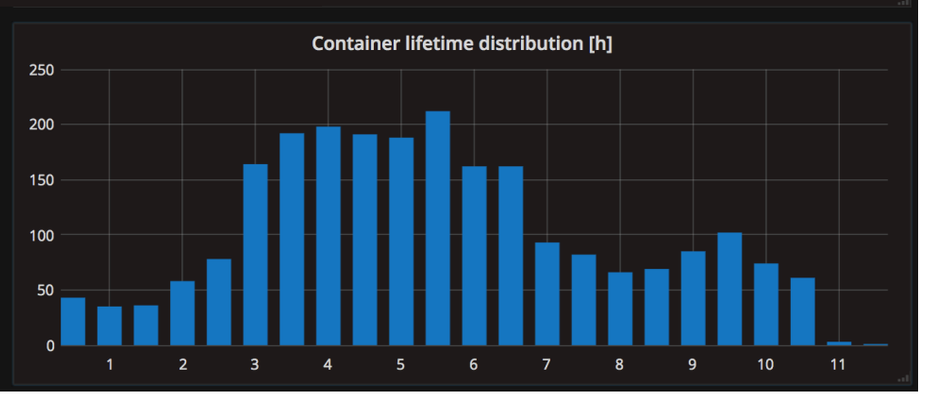
- Plancton is currently used in production on the ALICE High Level Trigger (HLT) facilities to opportunistically run Monte Carlo production jobs outside data taking
- Quickest method to deploy an AliEn site
- CVMFS and Plancton and Docker are installed on each node (only requirements)
- Using AliEn v2-19.395 (thanks Miguel and Cotstin!)
  - Configured to have 1 container = 1 job agent = 1 job
- It flawlessly scaled up to ~8000 cores
- Given the few requirements it is easy to port it elsewhere (see next section!)

# Already existing successful implementations: HLT (2)

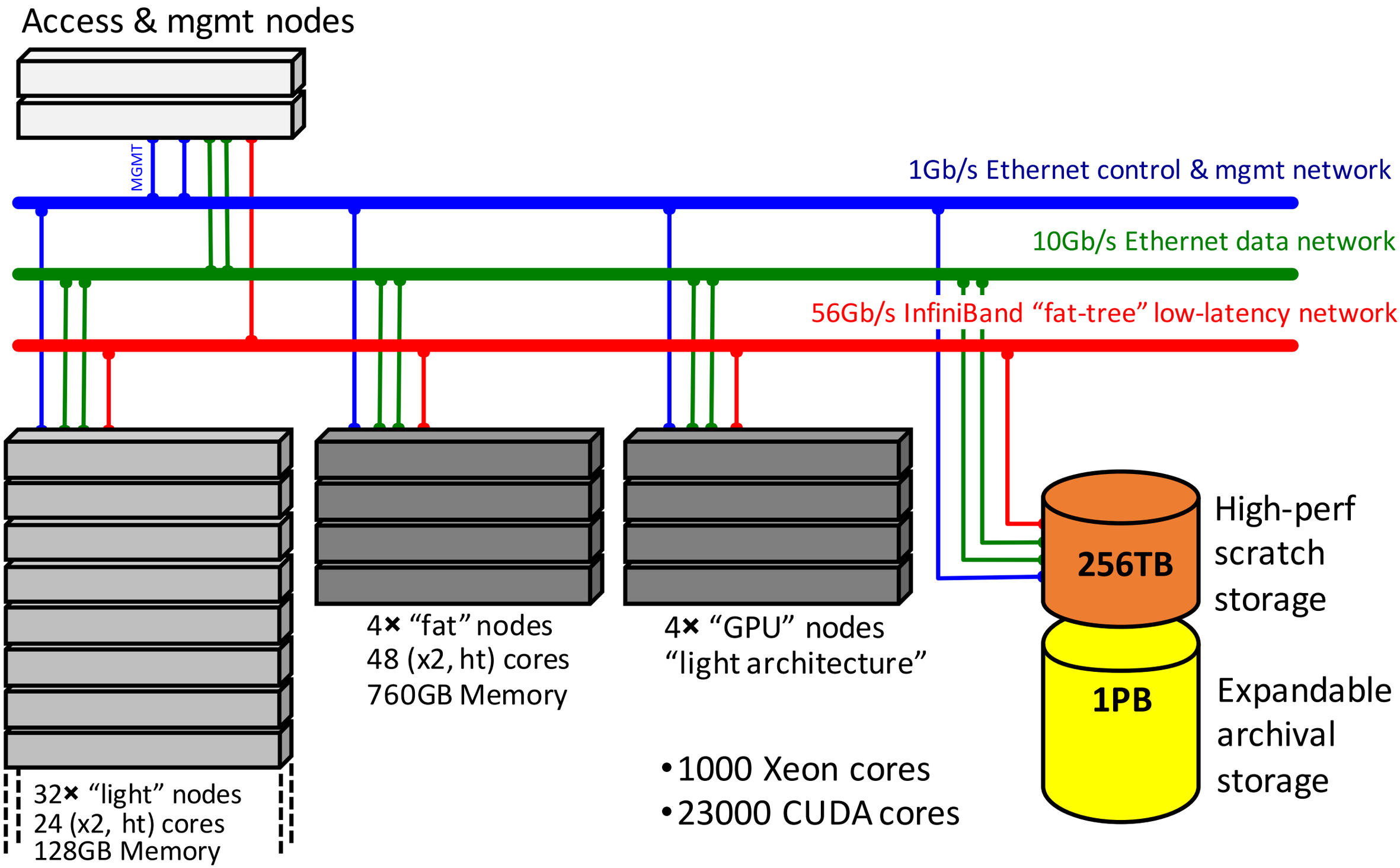




- HLT very stable with Plancton/Docker since end December 2016

# Setup of an AliEn site on the OCCAM facility using Plancton

# OCCAM architecture



# OCCAM: Multi-purpose HPC facility in Torino

- More than a traditional HPC cluster, caters to very diverse use cases
  - CPU-intensive HPC workloads
  - I/O-intensive data analysis
  - Single-image high-memory pipelines
  - GPU-intensive use cases
- Borrow some Cloud Computing concepts
  - Run “computing applications” instead of “batch jobs”
  - Cluster partitioned in isolated virtual resources

# OCCAM: Multi-purpose HPC facility in Torino

- Enabling technology: Linux containers (Docker+Mesos)
  - Isolate applications without performance overhead
  - Decouple application/infrastructure management into two administrative domains
- Opportunistic resources exploitation resources with Plancton
  - Quick deployment and scenario swap (opportunistic/ dedicated)
  - Vanilla environment for containers with only CVMFS as extra requirement
  - Possible to schedule opportunistic periods against production stops

# Implementation details: let's containerise all the things!

- We have containerised Plancton itself (mconcas/plancton)
  - Docker socket exposed inside Plancton container
  - Configuration updates automatically fetched via HTTP URL
  - Health check to verify it works
  - Ready to be deployed with Marathon (currently done with Ansible)
  - Other external components are in other containers (VOBox, RabbitMQ, InfluxDB, Grafana)

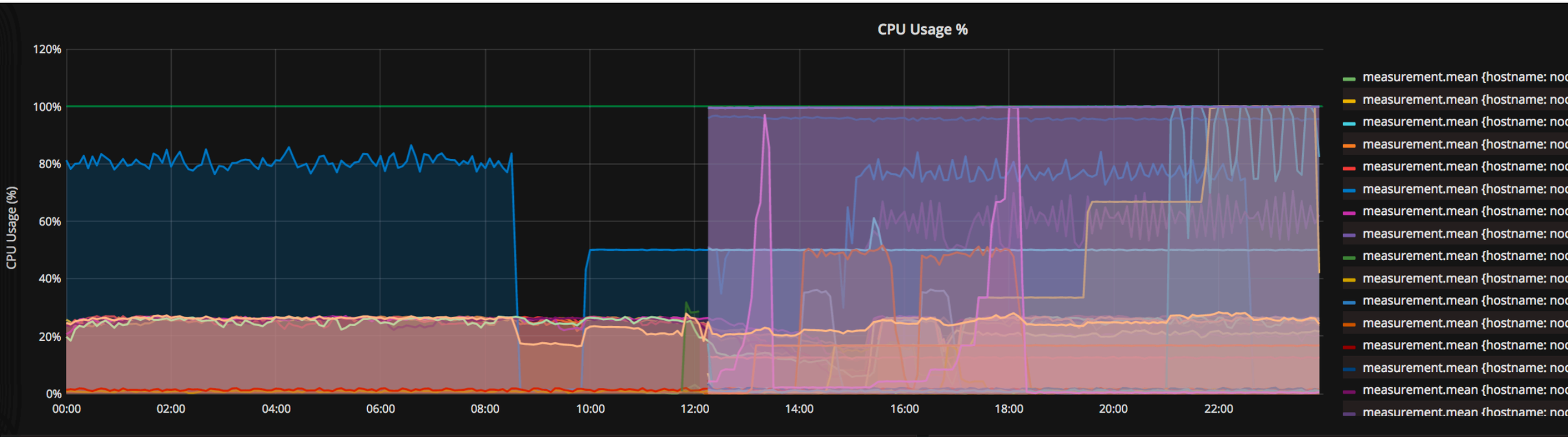
# Implementation details: VOBox and RabbitMQ

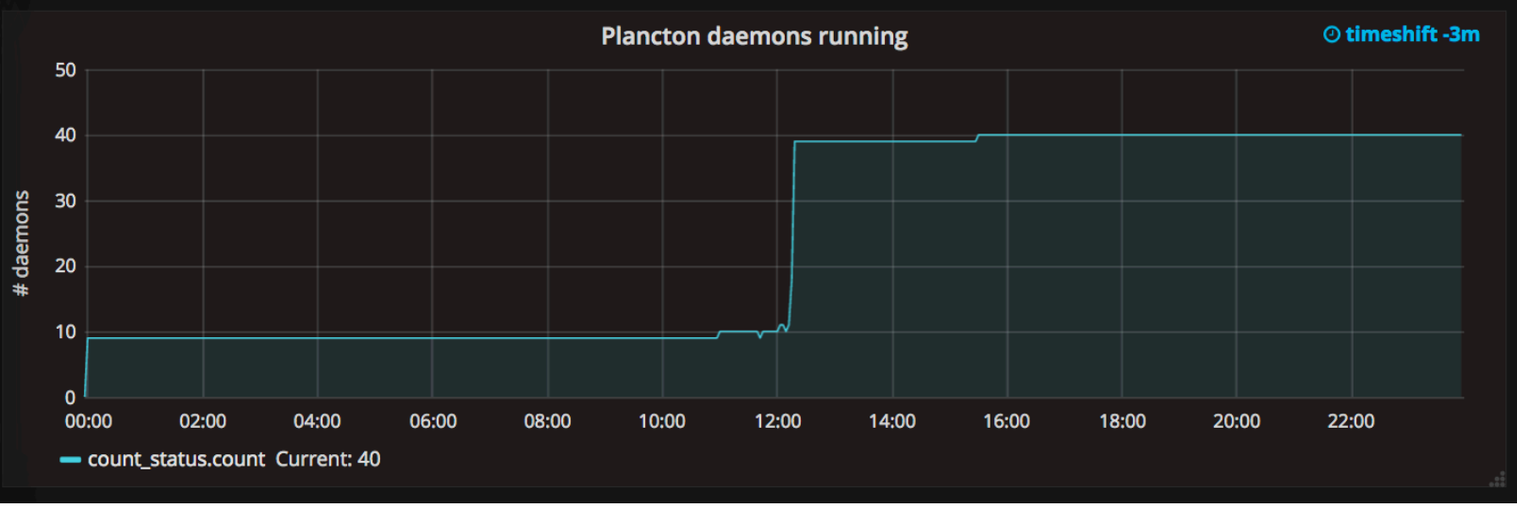
- Fully containerised AliEn VOBox!
  - X.509 credentials are not stored in the container image but exposed at startup
  - The AliEn CE submits job agents to RabbitMQ via HTCondor-like commands (**condor_{q,submit}**), see github.com/dberzano/thyme
- RabbitMQ unmodified official container works for us
- Stateless approach: if any of the service containers die (VOBox, RabbitMQ, Plancton itself!) jobs keep running - allows for rolling updates
- Worker containers: base CC7 images with HEP_OSlibs
  - Main executable **(thyme-worker)** polls RabbitMQ for some payload (AliEn JAs in our case) and dies after "some" seconds if there is nothing to do
  - AliEn JA runs a single job for increased security and better scheduling granularity (useful when draining) - possible with a new AliEn config option

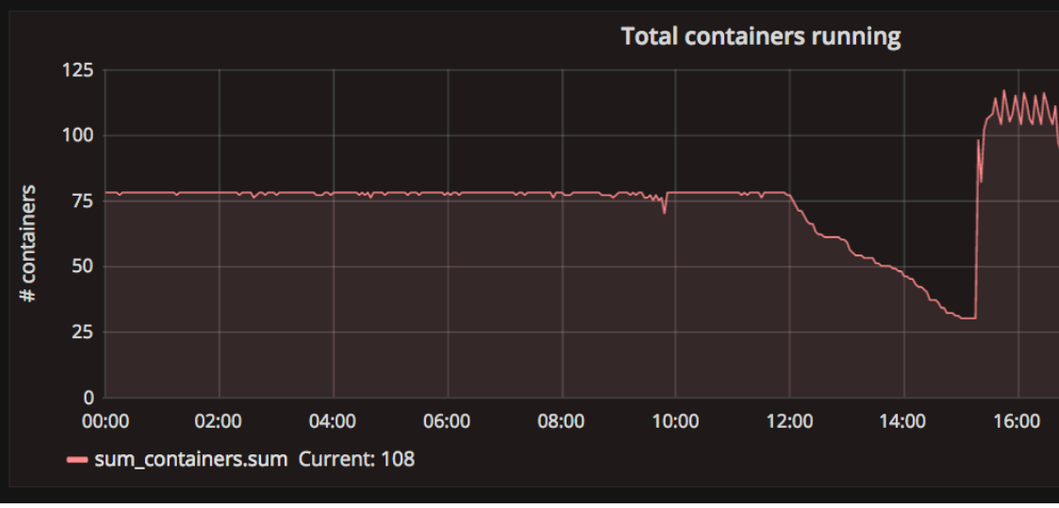
# Implementation details: monitoring the application

- Plancton instances send “metrics” to (configurable) InfluxDB targets
  - Low network pressure: “few” queries per minute
  - Database is a container with a Docker volume for data persistency
- Metric plots using a Grafana dashboard
  - Same template used on HLT
  - Integrated statistics help to diagnose possible issues

# OCCAM Dashboard



# Results on the OCCAM facility

- Plancton could be easily ported to OCCAM in a single working day
- Some ALICE productions work properly with 10 GB of RAM memory (no swap)
  - OCCAM “fat” nodes host up to 80 MC jobs (96 cores available)
  - OCCAM “light” nodes only 13 (48 cores available)
  - Increase swap is not an option on HPC
  - At the moment we cannot dedicate long running times to Monte Carlo (it is purely opportunistic)