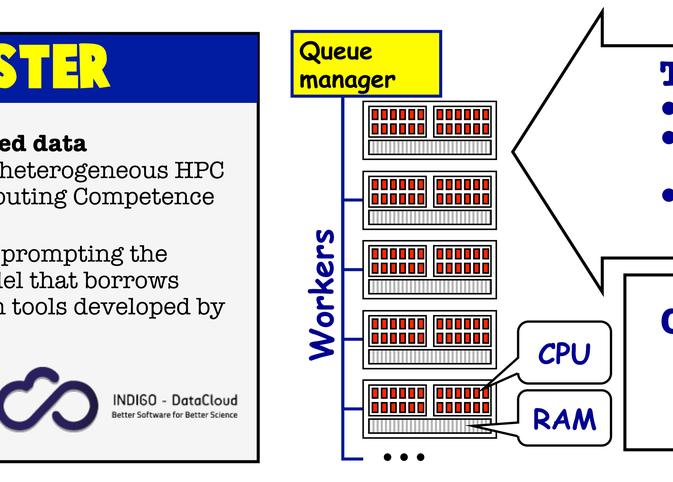
MANAGING AN HETEROGENEOUS SCIENTIFIC COMPUTING CLUSTER WITH CLOUD-LIKE TOOLS: IDEAS AND EXPERIENCE

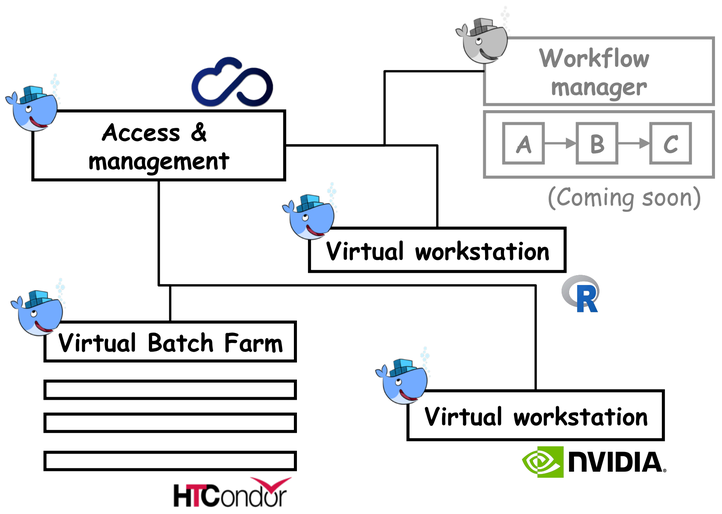
Marco Aldinucci[1], Stefano Bagnasco[2], Matteo Concas[3], Stefano Lusso[2], Sergio Rabellino[1], Sara Vallero[2]

[1]C3S and Computer Science Department, University of Torino
[2]C3S and Istituto Nazionale di Fisica Nucleare, Torino
[3]C3S, Politecnico di Torino and Istituto Nazionale di Fisica Nucleare, Torino

THE OCCAM CLUSTER

- The **Open Computing Cluster for Advanced data Manipulation** (OCCAM) is a multi-purpose heterogeneous HPC cluster operated by C3S, the Scientific Computing Competence Centre of the University of Torino.
- It caters to a very diverse user community, prompting the adoption of an innovative management model that borrows several Cloud Computing concepts, based on tools developed by the INDIGO-DataCloud EU project.





OVERALL OCCAM ARCHITECTURE

Core concept: Scientific Computing Application, defined by:

- **Runtime environment** (application software, libraries,configurations,...)

We use Linux containers to package the full application runtime environment and provide a virtualization layer free from performance penalties, leveraging on the Docker ecosystem to allow users to re-use off-the-shelf images and decouple application support from infrastructure support.

- **Resource requirements** (CPU, memory, GPU, low-latency network, storage,...)

Resource requests are granted by a reservation-allocation mechanism; in the final architecture, Apache Mesos frameworks will manage resources and manage all inter-application scheduling.

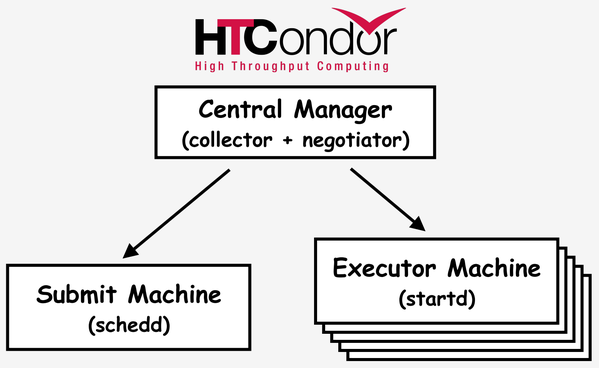
- **Execution** model (batch-like, single image interactive, pipeline,...)

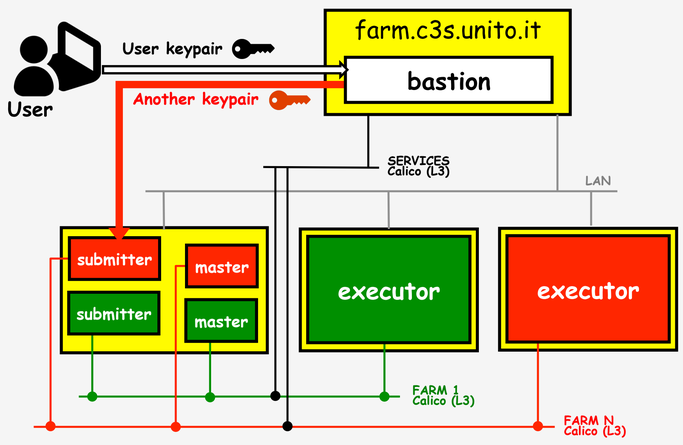
Different execution models are supported by deploying compositions of the relevant management and execution services. Most complete example: **Batch System as-a-service** (see below).

BATCH SYSTEM AS-A-SERVICE

Dockerized HTCondor-based batch farm

- All farm components are packaged in separate Docker containers: they can be started as Mesos tasks or with the usual 'docker run' command.
- Configuration parameters passed as 'docker run' arguments end-up as parameters to the entrypoint script
- Tini (https://github.com/krallin/tini) and Supervisord (http://supervisord.org/) to start required services
- Expose health status and custom metrics (HTTP endpoint on port 5000)
- Access through "bastion service" containers running on the OCCAM front-end





Orchestration: automatically deploy a scalable batch system virtual cluster

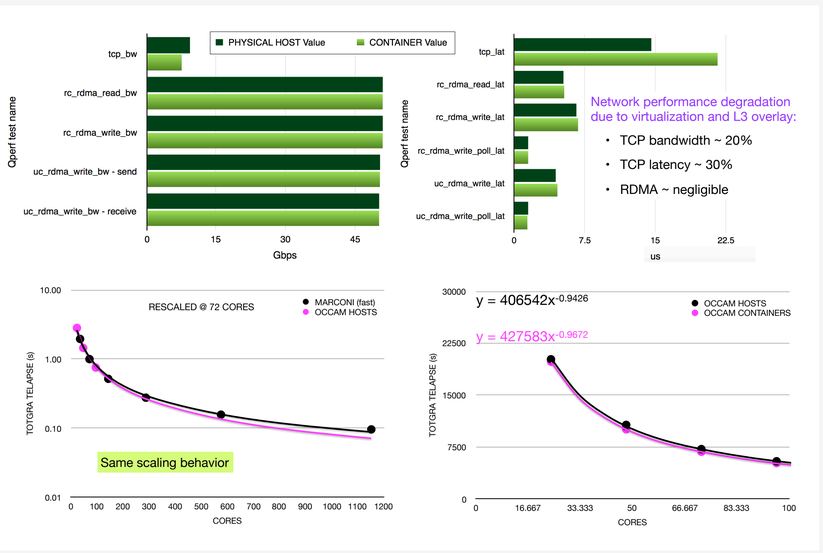
- **Apache Mesos** for resource abstraction and management
- **Marathon** to schedule cluster components as long-running services
- Alternatively, **HTMFrame**: a custom Scala implementation of the Mesos Scheduler Driver github.com/svallero/HTMframe
  - Implements custom policies on Mesos resource offers to instantiate roles in the correct sequence, health-check them and reinstantiate upon failure
  - **Auto-scaling of the cluster** according to HTCondor metrics

Networking: isolation via private L3 overlay networks, managed by **Calico**

- ACLs to manage access from the bastion service to different networks
- Mesos-DNS for service discovery

PROJECT STATUS AND PERFORMANCE STUDIES

- The OCCAM cluster is a production facility that had to run user workloads since the very beginning, so there is very small room for R&D activities and the adoption of the final architecture is gradual, using a DevOps approach.
- For example, single-node applications, i.e. "virtual workstations" with access to high-memory 4-way nodes or GPUs, and pipeline-like applications are not yet managed by Mesos but by custom tools allowing users to run their Docker containers on the cluster in a secure way.
- However, the user workflow is defined and we plan to gradually reach the final configuration without changing the user interface semantics.



Networking performance:

- As measured with qperf
- TCP performance penalty due to the Calico networking between containers
- However, no latency or bandwidth degradation on InfiniBand (directly exposed as a device in the container)

Scalability:

- Using CRYSTAL, a widely used MPI computational chemistry software
- No performance penalty between containers and running the same directly on hosts
- No difference in compoarison with a conventional HPC cluster (CINECA's Marconi)

CONCLUSIONS

- We have been operating the OCCAM cluster for more than one year using an innovative cloud-like management model based on deploying virtual clusters tailored for specific Scientific Computing Applications
- We adopted a two-level scheduling mechanism, with in-application scheduling managed by HTCondor and inter-application scheduling managed by Mesos frameworks
- By directly using Docker instead of Singularity or Shifter we simplified the software stack and were able to exploit a wider software ecosystem
- Several Computing Applications have been run on the system, generally with a reasonable learning curve for the adoption of containers, and remarkable stability in production mode
- Even though the system is not yet running in its final architectural configuration, the first operational experience and performance tests imply that the model is viable and, indeed, provides access to resources to communities usually excluded from larger conventional HPC facilities.