Vacuum models for job execution

Andrew McNab
University of Manchester
GridPP, LHCb, DUNE

Daniele Spiga
INFN Perugia
CMS

Overview

- Vacuum model
- Vac and Vcycle
  - Vac and Vcycle architecture
  - At WLCG sites
- DODAS slides from Daniele Spiga
  - DODAS and its vacuum implementation
  - CMS workflow including DODAS
- Themes / ideas

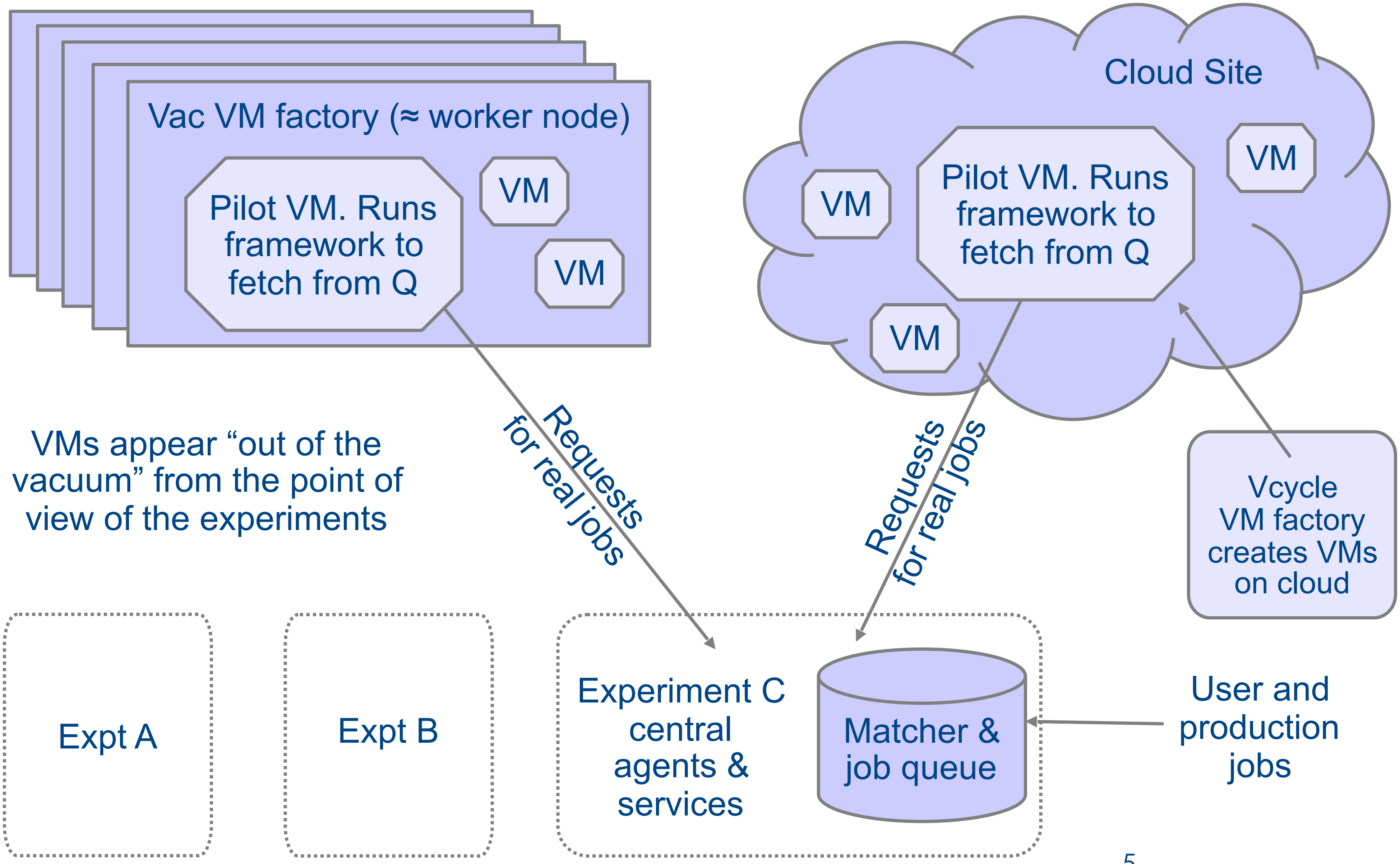
Vacuum model

- Introduced several years ago
  - *("Running jobs in the vacuum", A McNab et al 2014 J. Phys.: Conf. Ser. 513 032065)*
- Goal was that instead of experiments pushing pilot jobs into sites through a Compute Element, the sites will pull jobs in
- You can then have much simpler sites as you just need to manage the pull process: you don't need a CE or batch system
- You still need to know how to pull jobs for each experiment you support
  - Have a suitable VM or container "worker node" for each experiment
  - Have a way of creating "logical machines" like that on the hardware
- Vac does this on autonomous bare metal worker node machines
- Vcycle applied Vac ideas to use OpenStack etc for the creation of the VMs
- DODAS creates services on a cloud system or bare metal
  - Services then create workers which use the vacuum model to get work

Vac and Vcycle

- Both VM lifecycle managers for VMs running jobs, using the Vacuum model
- Vac is a standalone daemon on each worker node machine to create VMs
  - Vac worker nodes talk to each other to achieve site-wide target shares for experiments
- Vcycle manages VMs on clouds like OpenStack, Google Cloud, EC2, …
  - Can be run at the site, by the experiment, or by infrastructure projects like GridPP
- LHCb, ATLAS, ALICE, generic DIRAC VMs working in production on Vac and Vcycle
  - Same VMs for both Vac and Vcycle
  - These VM definitions are CernVM-based, but monolithic images work too
  - LHCb also provides Docker containers which Vac can run
- VM - Vac/Vcycle interface defined by HSF technical note HSF-TN-2016-04

Vac and Vcycle



Vac VM factory (≈ worker node)
Pilot VM. Runs framework to fetch from Q
VM
VM
Cloud Site
VM
Pilot VM. Runs framework to fetch from Q
VM
VM
Requests for real jobs
Requests for real jobs
Vcycle VM factory creates VMs on cloud
Expt A
Expt B
Experiment C central agents & services
Matcher & job queue
User and production jobs

VMs appear "out of the vacuum" from the point of view of the experiments

Vac/Vcycle at WLCG sites

- The site just has to run Vac or Vcycle - no grid middleware
  - If you're running Vcycle, then *you* have decided to run OpenStack etc yourself, or you get it from your IT services, or a company, etc etc
  - All this just covers compute of course; not storage
- You don't need to run a BDII for this to work, but:
  - Vac/Vcycle send monitoring to a central Ganglia-style service: VacMon
  - VacMon republishes this info in the new CRR JSON on behalf of sites
- You don't need to run an APEL server
  - Vac/Vcycle write out APEL accounting record files as VMs finish
  - Sites run ssmsend from cron to send them to directly to APEL
- Vcycle can manage a pool of Squid server VMs
- Each Vac factory has its own Squid server, which joins a peer-to-peer pool at the site

DODAS in a nutshell

DODAS: **D**ynamic **O**n-**D**emand **A**nalysis **S**ervice

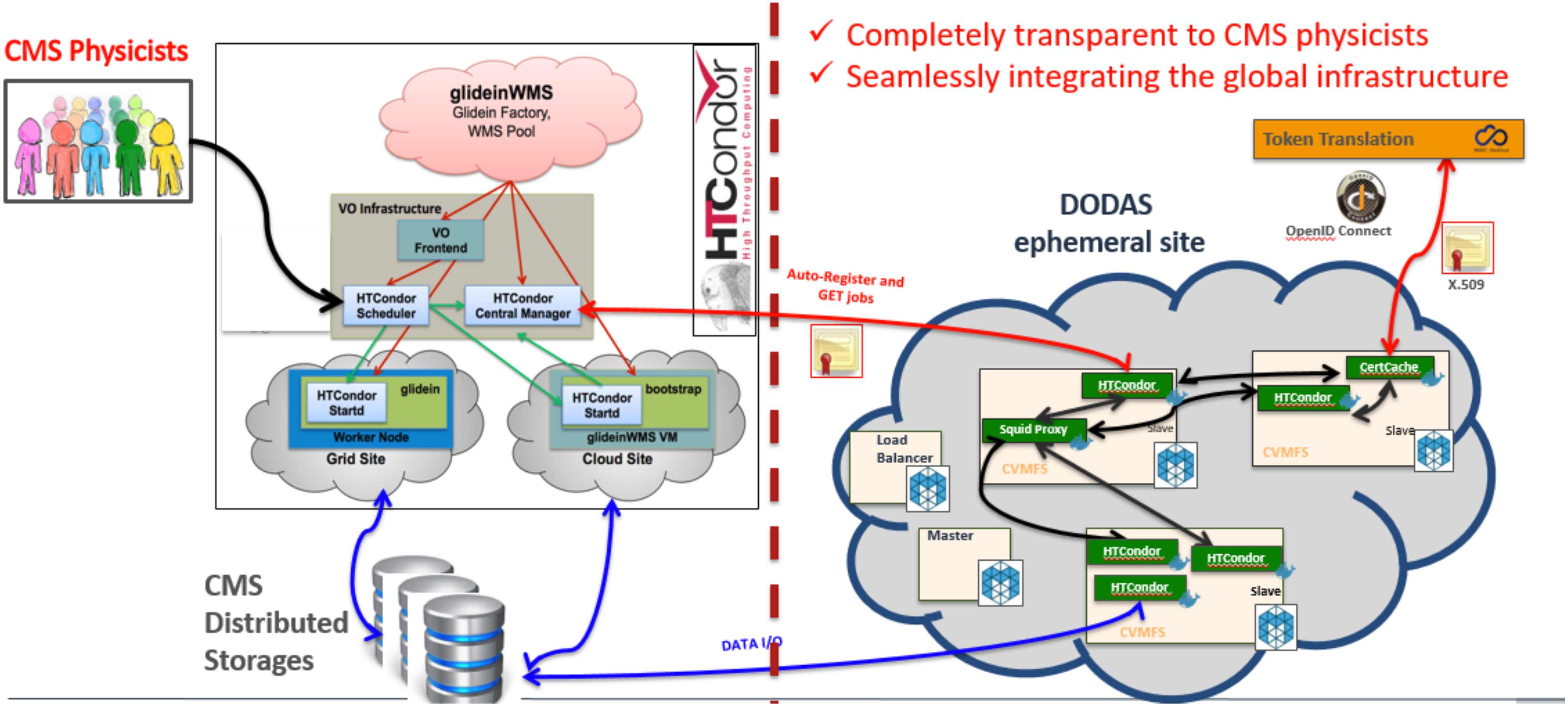
- A open source deployer manager
  - Allows on-demand creation and configuration of container based clusters for data processing with almost zero effort
  - A cluster can be a standalone set of resources, a **WLCG Tier*-like** an extension of an existing center and more
    - BigData Analytics, Batch System as a Service, Distributed processing framework for ML
- Support for hybrid clouds deployment
- High level of automation and self-healing
- Oriented to the ZeroOps model
- Supports communities-tailored (user-tailored) applications and software for data processing
- Flexible Authentication and Authorization model based on JWT from INDIGO-IAM
- Based on “industry standards” to minimise code development and maintenance

Vacuum approach with DODAS

- DODAS relies on vacuum approach to provide WLCG-like resources and it is responsible to automate:
  - Bare-hosts (e.g. VMs) instantiation based on user requirements, defined at TOSCA level
    - Virtual hardware can be scaled up/down (elasticity)
  - Services and software configurations at host leves (e.g. CVMFS, docker engine etc)
  - Container orchestrator deployment (e.g. K8s, Mesos/Marathon), and this is in form of dockers
  - Deployment and execution of services/microservices (e.g. Worker Nodes, squid proxy, x509 cache) over orchestrators
    - **this is how worker nodes are "spontaneously produced" and scaled up/down**
  - In addition DODAS provides a JWT based ecosystem for authentication and authorization: INDIGO-IAM

- Current incarnation is based on HTCondor as a means to manage (aka overlay) distributed worker nodes (startd)
  - Does not deploy Computing Element (CE) but it could be added (example of modularity)
  - DODAS Vacuum system is integrated in the **CMS Global pool** (details on backup slides)

The overall workflow at CMS



✓ Completely transparent to CMS physicists
✓ Seamlessly integrating the global infrastructure

- **The very strategy to the automation has been applied to a Xcache**
  - **DODAS generated site can incude a XCache automated setup**

In closing: themes/ideas

- Radical ways of pursuing Lightweight Sites ideas
  - Either remove need for services or “outsource” them
- What things do we need WLCG to do/change?
  - APEL was easy to outsource from the site, since the central server accepts usage records directly
  - BDII -> CRR JSON and CRIC is another good step
  - What about ETF tests? (LHCb and CMS done)
- Can the implementations interoperate?
  - eg can we have LHCb containers running on DODAS sites?
  - and CMS containers created by Vac/Vcycle?
  - And used by other implementations?

Extra Slides

Key objectives

The target is to simplify both resources provisioning, setup and services deployment & management, as such the main objectives are:

- **To abstract from the underlying infrastructure**:
  - Virtual HW provisioning is not bound to a specific resource provider, hybrid model (see later)
    - Uses TOSCA to describe resource requirements and dependencies
    - Relies on Infrastructure Manager to connect with infrastructures/providers
- **To automate services deployment and operations**:
  - Deployment and setup dependencies, services and code
  - Provide knobs for self-healing enabling restarting and reconfiguration to reduce ops effort
  - Enable elasticity and scalability based on custom metric (and/or manual)
    - Uses Docker && Ansible and container orchestrators (Mesos, K8s)

The overall implementation is **modular and flexible** in order to allow customisations and extension **accommodating use cases and requirements community-specific**

DODAS: main motivations

- To provide a “ZeroOps solution” to exploit opportunistic computing
  - Optimising the costs/benefits to exploit resources not necessarily/permanently dedicated to a specific experiment
- To allow sites with limited (or without) effort for experiment specific support to provide computing to the collaboration
  - Tier3-like, campus facilities and general purposes farming
- To ease site overflow and/or elastic site extension
  - To absorb peaks of requests at Tiers1/2 by using external providers such as public clouds
  - To accommodate workflows with special requirements

- Moreover DODAS can be a technology enabler to build:
  - Regional/national level computing infrastructure
  - Prototype of quasi interactive analysis facility to exploit different model for physics analysis
    - integrated into the data and workflow management of the experiments.

Prerequisites (from a VO perspective)

1. DODAS borns as Platflorm as a Service (PaaS) solution to abstract any cloud IaaS as such it **relies on cloud API endpoints** for resource provisioning and resource configuration
   - Supports a wide range of clouds: verified on AWS, Google, Azure, Openstack, OpenNebula, TSystem
   - However **the core system is completely "cloud free"**, it is a container based system managed through Ansible roles, this imply
     - Manual deployment works on pre-allocated bare metal
     - Moreover we are evaluating to extend automation also to this scenario

2. Community runtime environments, dependencies and specific services **must be provided**
   - Either extending existing configurations and images or developing new one

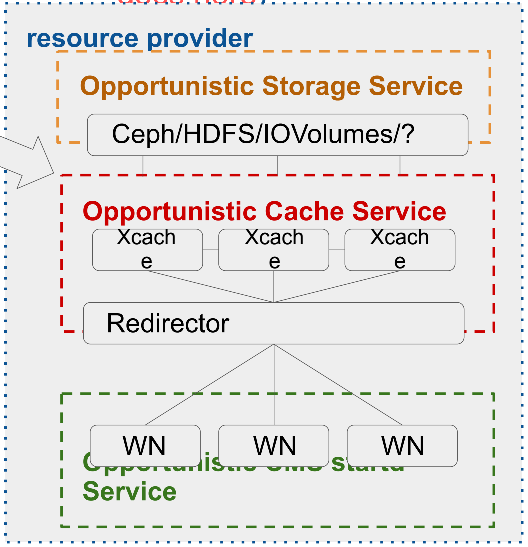
DODAS is currently integrated within CMS experiment and AMS2 analysis workflow. Also it is being evaluated within DAMPE, Virgo and Fermi analysis workflows

Not only compute… data ingestion

- The very strategy to the automation has been applied to a Xcache Docker container has been setup to allow an easy deployment
  - passing a complete xcache config file
  - or setting caching parameter as arguments/env
  - healthcheck call implemented

A Docker Compose configuration file is available to **orchestrate the deployment of a local test instance.** The stack contains a test remote server, a cache instance and cache redirector (preliminary docs here)

- A variety of recipe for orchestration tools have been created:
  - docker swarm, k8s and marathon services (redirector+caches)
  - config and scale services with compose-like recipes

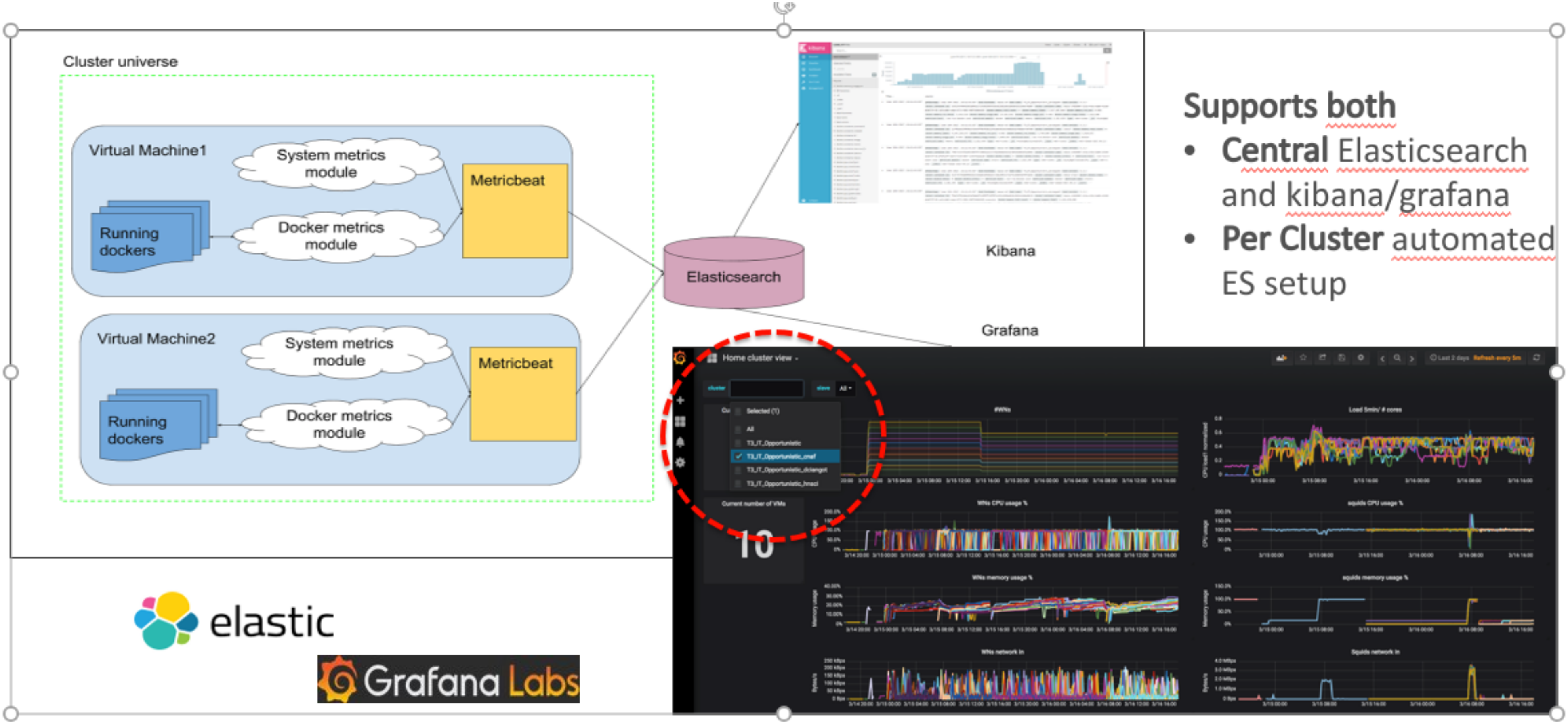

resource provider
Opportunistic Storage Service
Ceph/HDFS/IOVolumes/?
Opportunistic Cache Service
Xcache
Xcache
Xcache
Redirector
WN
WN
WN
Opportunistic CMS site Service

- Work in progress
  - Preparing to test the new HTCondor token support, together with INDIGO-IAM authZ service

Work in progress

- DODAS Monitoring: provided in beta version is continuous development driven by requirements
    - under consolidation
- SAM integration is on going
    - End to end check has been done with succes
- APEL based accounting
    - Started right now
- Data ingestions solution and optmisation
    - Continuous evolution

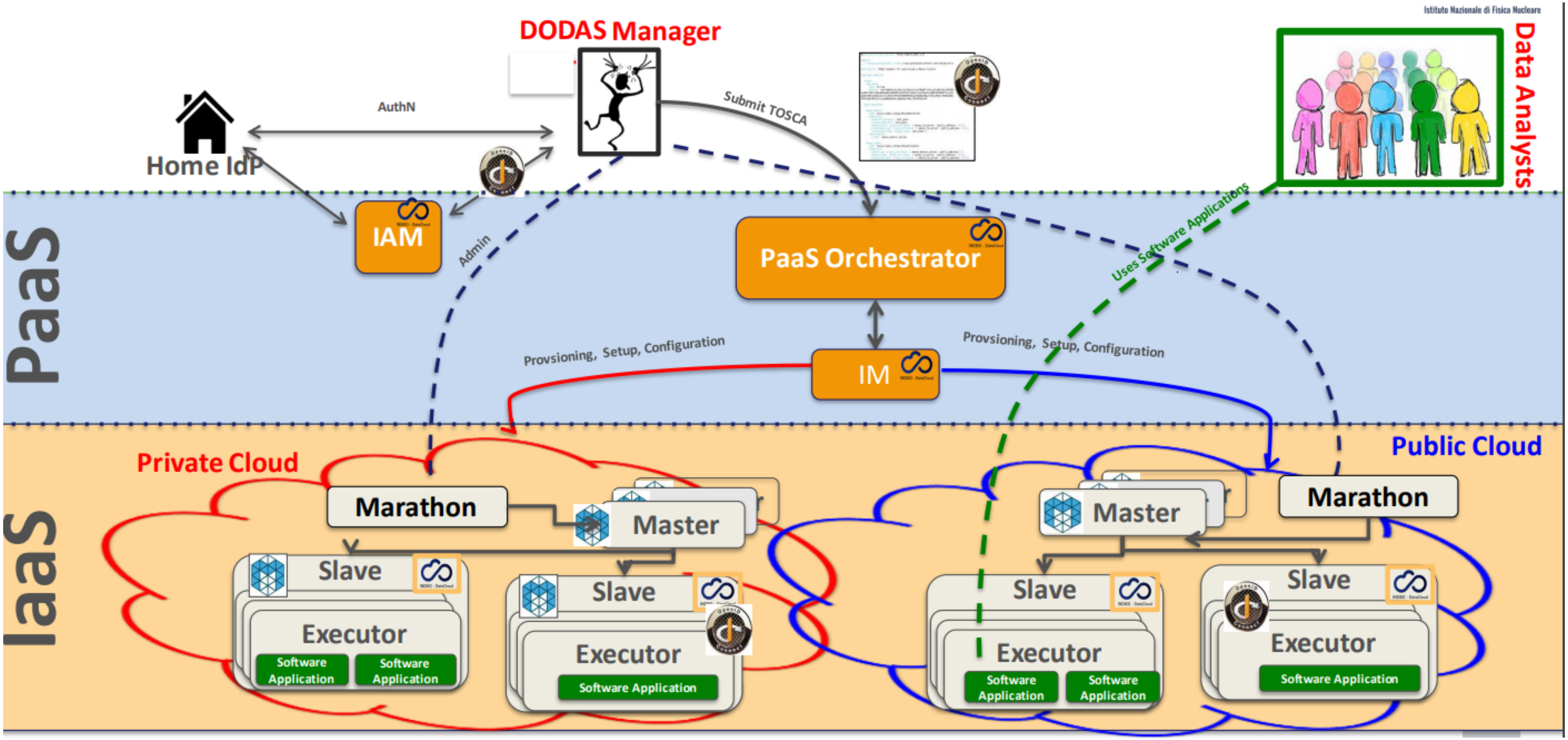
Underlying Monitoring implementation



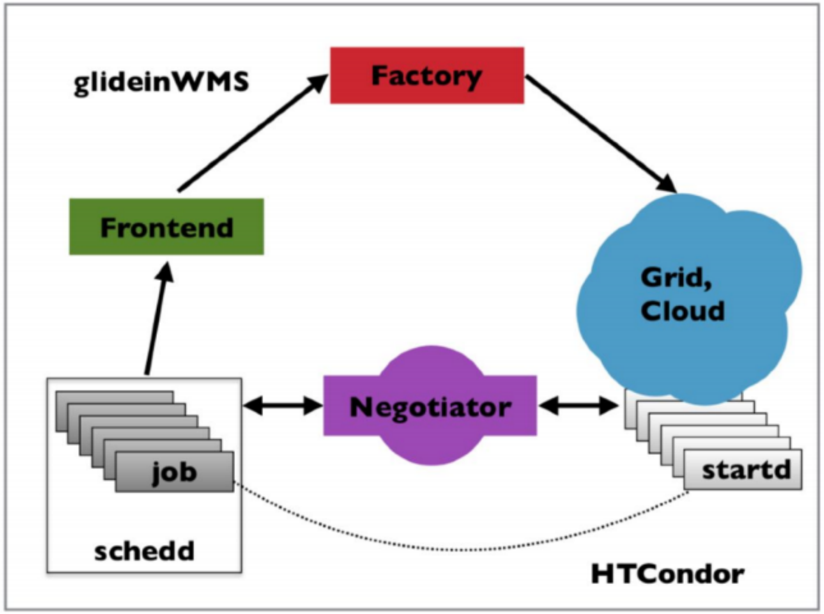
Supports both

- **Central** Elasticsearch and kibana/grafana
- **Per Cluster** automated ES setup

Architectural overview

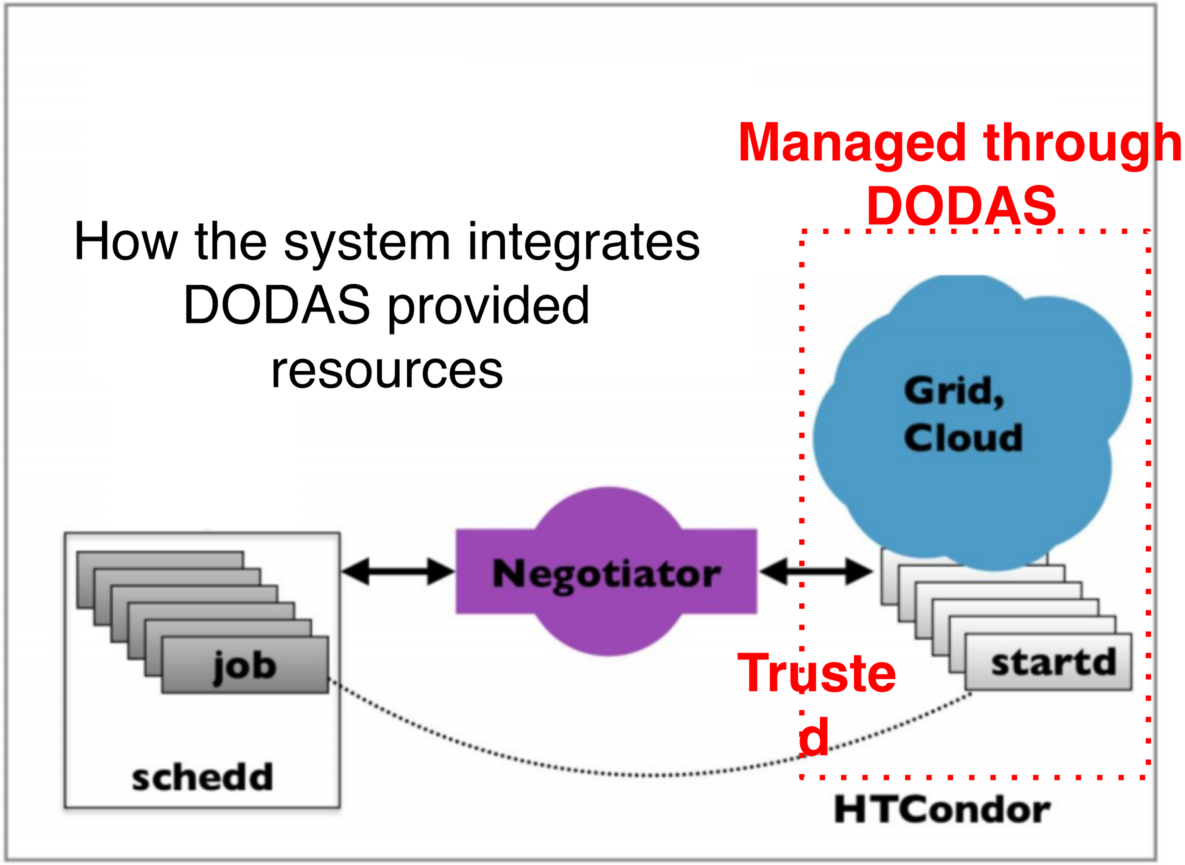


From CMS Global pool perspectives



High level view of the CMS Global pool and its main elements

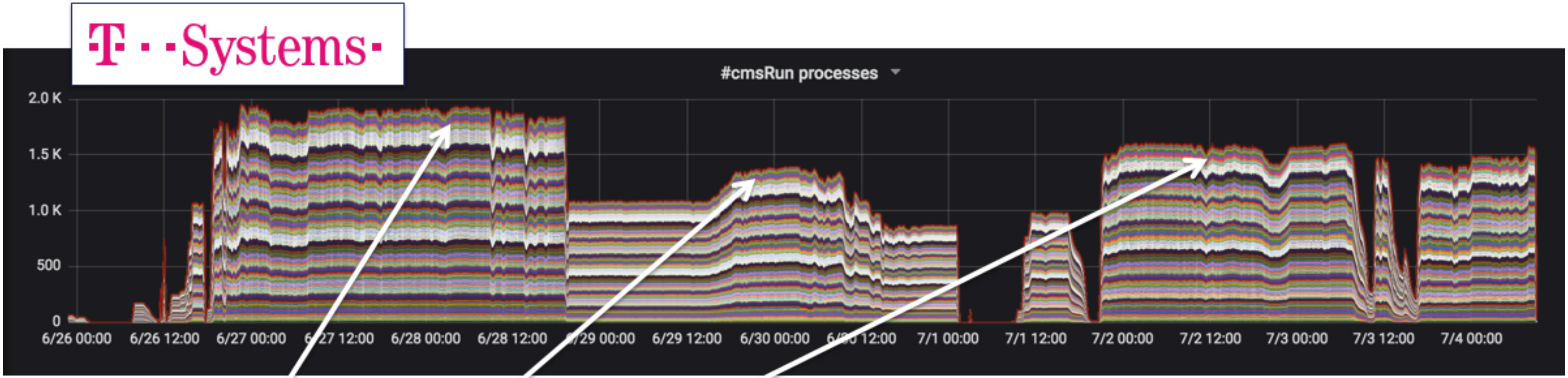
How the system integrates DODAS provided resources

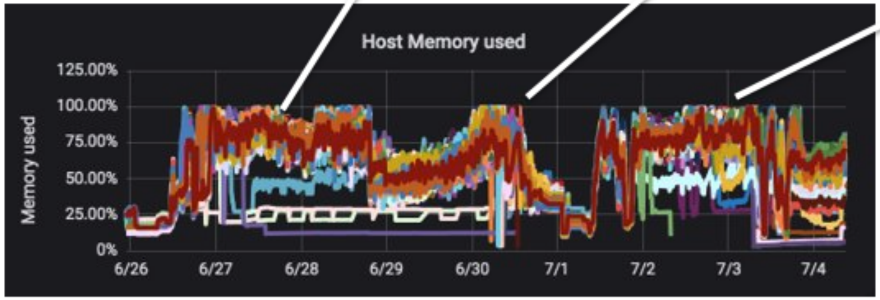


CMS Integration

- DODAS is fully integrated into the CMS computing model to create lightweight ephemeral WLCG-Tier on demand
  - Generated sites automatically deploy and manage
    - HTCondor services
    - Squids
    - ProxyCaches
    - CVMFS
- Completely transparent to the users
  - Additional resources are made available to the global pool and user/ compOps can exploit them

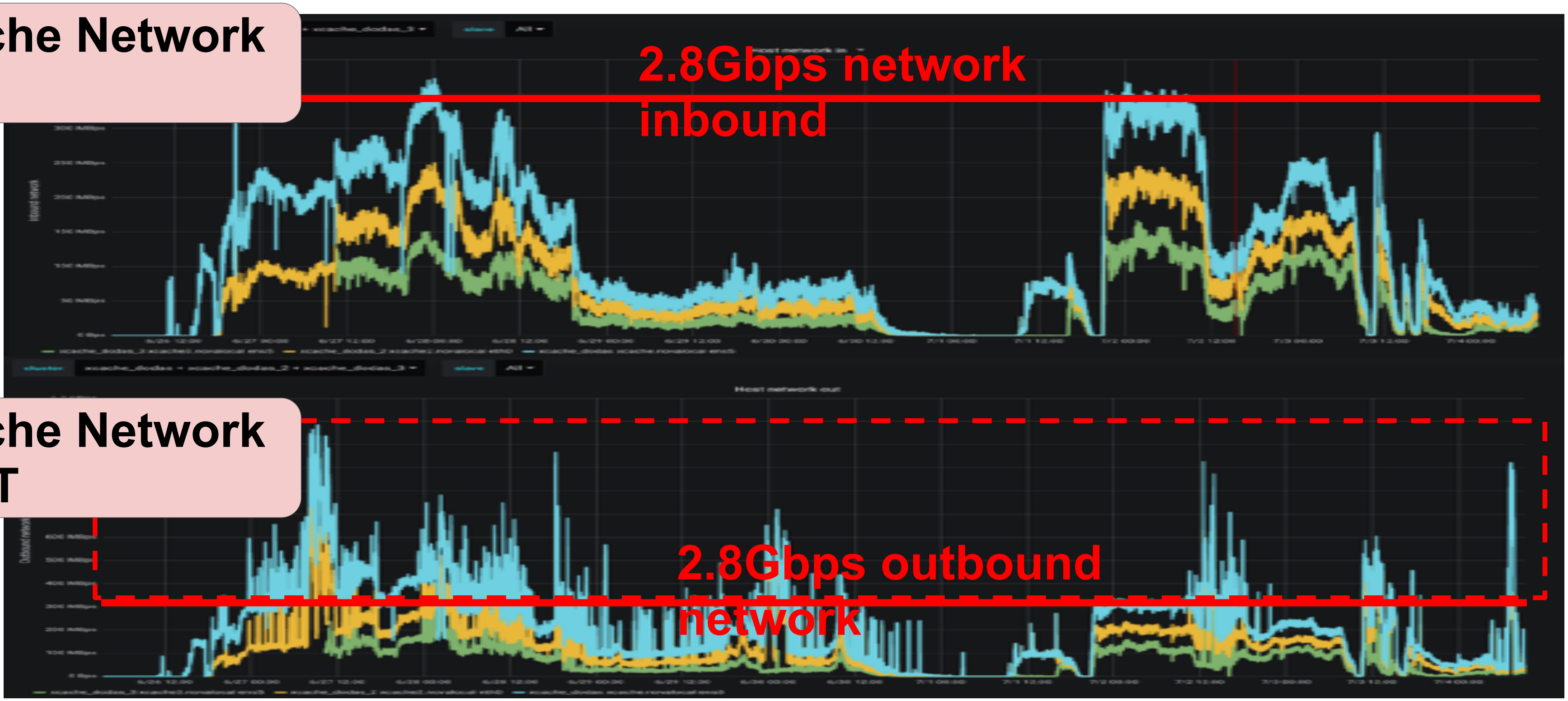
DODAS and public cloud: CMS analysis jobs





- Elasticity and self-healing
- Stability over days/weeks (120k jobs)
- Handling "special requirements"
high memory jobs

CMS analysis on public cloud: effect of cache layer



Cache Network IN

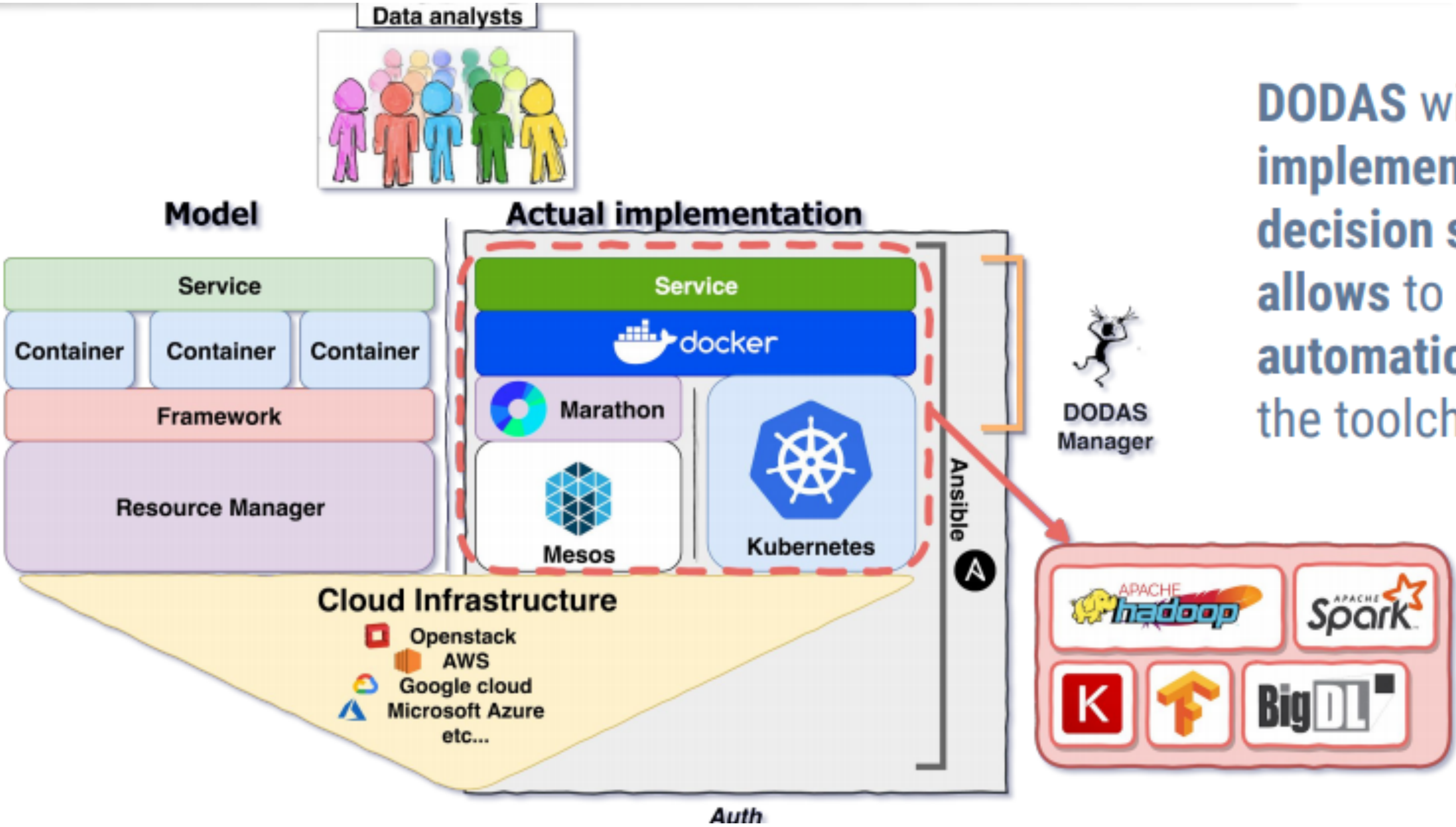
2.8Gbps network inbound

Cache Network OUT

2.8Gbps outbound network

Not negligible gain with **significant spike of direct cache disk/memory usage**

Not only HTCondor



DODAS will be used to **implement** a **smart cache decision service** because it **allows** to **compose automatically the blocks** of the toolchain.

For details see also ACAT2019: https://bit.ly/2TRZND4