



U.S. DEPARTMENT OF  
**ENERGY**

Office of  
Science



**Korea Institute of  
Science and Technology Information**

# Demo of Federated Cloud Job Management

Stuart Fuess (Fermilab), for KISTI-Fermilab CRADA collaboration project

Supercomputing 2015

November, 2015

# Purpose of the demo – a few preparatory slides

---

- Fermilab has embarked on a project to provide our High Energy Physics (HEP) users with a unified interface for job submission to multiple resources, including:
  - Local dedicated High Throughput Computing (HTC) clusters
  - Shared opportunistic grid resources, particularly the Open Science Grid (OSG)
  - Cloud resources, including:
    - Partners, such as KISTI
    - Commercial, such as Amazon Web Services (AWS)
  - High Performance Computing (HPC) centers
- Multiple resources may be available at any time
  - The challenge is to effectively and economically distribute data intensive workflows among the diverse resources

# Fermilab HEP Cloud Facility Project

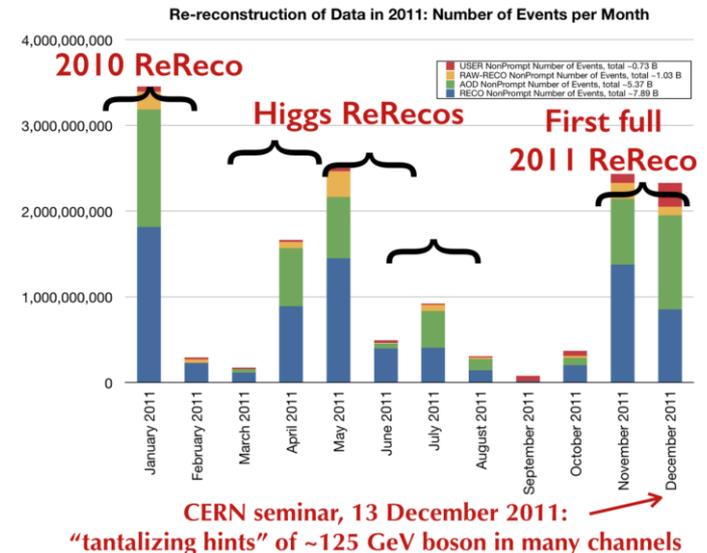
---

- The project is building the infrastructure to:
  - Manage 10s of thousands of simultaneous HTC jobs
  - Match job resource requirements to potential computing resources
  - Decide when and where to effectively place jobs
    - This **Decision Engine** is the key intellectual component of the project
      - Includes the economic factors essential for deciding if and how to utilize commercial resources
  - Provide the tools for input and output data management
- This is a project in progress, and the demo will utilize development infrastructure

# KISTI and Fermilab joint development

- KISTI and Fermilab have worked with a CRADA for joint development of a federated cloud infrastructure
  - The work has naturally evolved into a component of HEP Cloud
- The collaboration has contributed essential components of:
  - Cost-sensitive provisioning algorithms for AWS spot market
  - Image management, code and data movement techniques

A focus has been on workflows for the CMS experiment at the LHC, which shows typical “bursty” resource needs:



# Elements of the architecture

---

- Virtual Organization (VO) resources
  - Job submission
  - Job management infrastructure
  - Code, database, and other remotely accessible storage elements
- Compute resources
  - The demo will access:
    - KISTI's GCloud OpenStack cloud
    - AWS
  - Potentially numerous other resources are available
- The HEP Cloud Facility infrastructure
  - To tie the resources together

## What the demo will show

---

- With animated slides and occasional cutovers to actual job submission and monitoring sites:
  - Submission of a set of physics simulation jobs for the CMS experiment at the LHC, utilizing the CMS VO job submission infrastructure
  - Distribution of these jobs among resources recognized by the HEP Cloud Facility, including:
    - KISTI GCloud OpenStack cloud
    - AWS
- Start with an overview of the components

VO  
Infrastructure

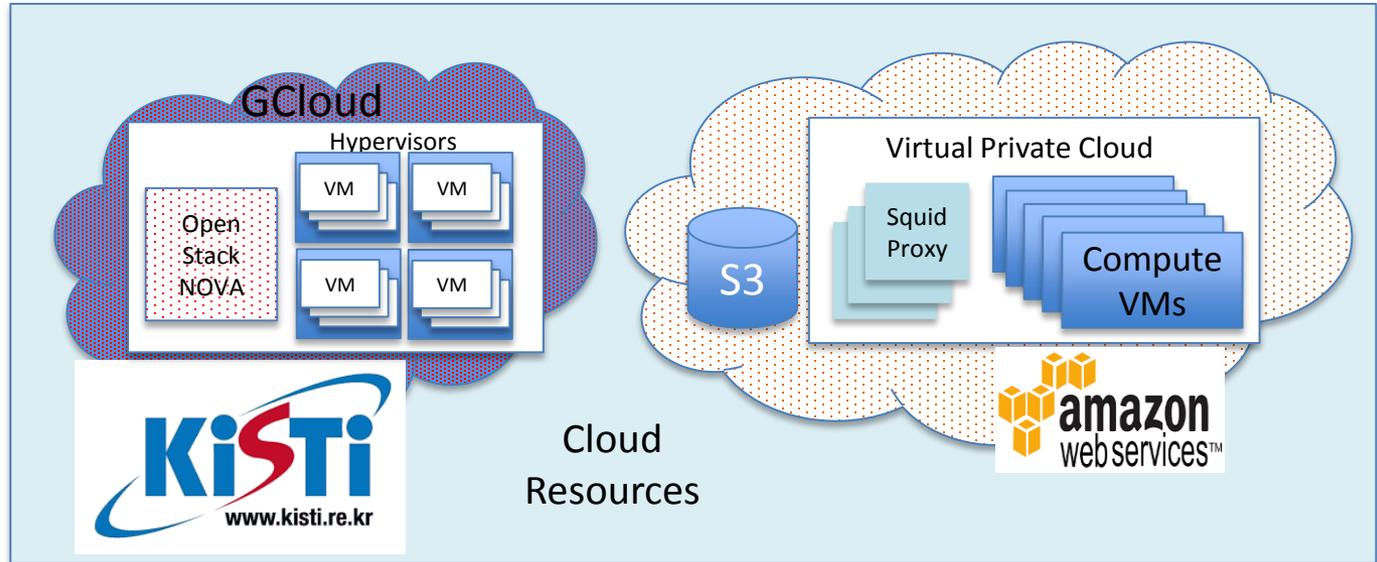
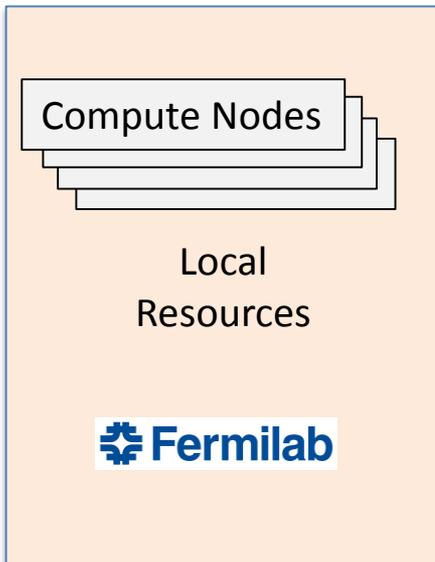
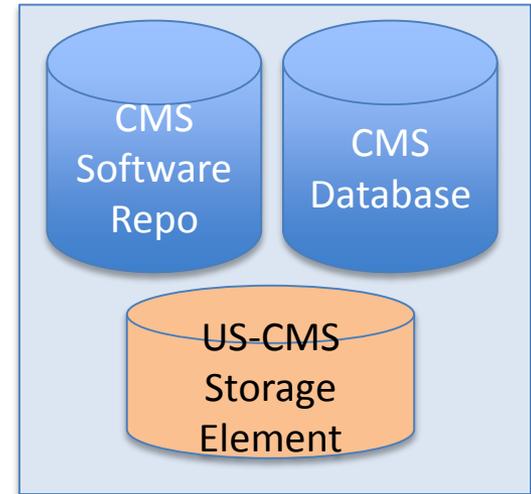
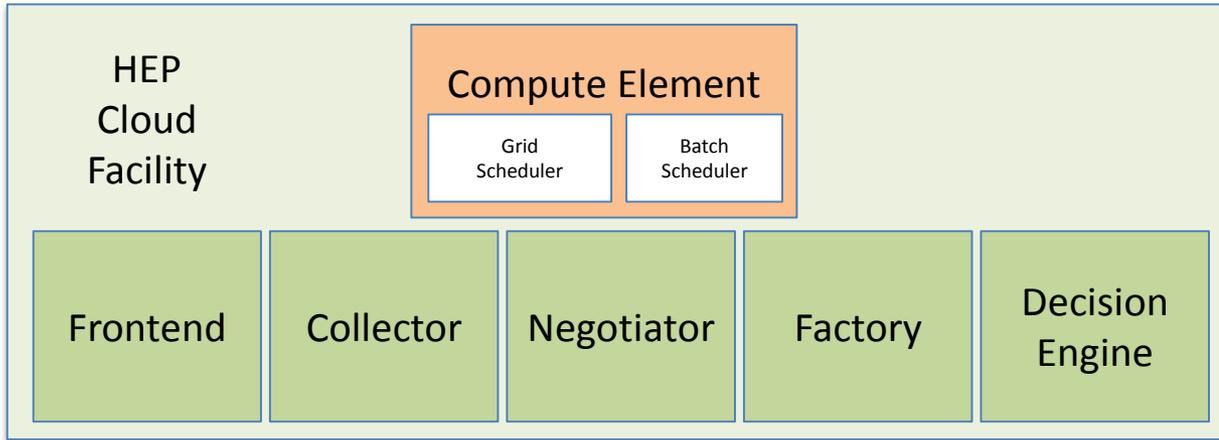


CMS  
Workflow  
Agent

Jobsub  
Server



VO  
Resources



# Now the animation...

---

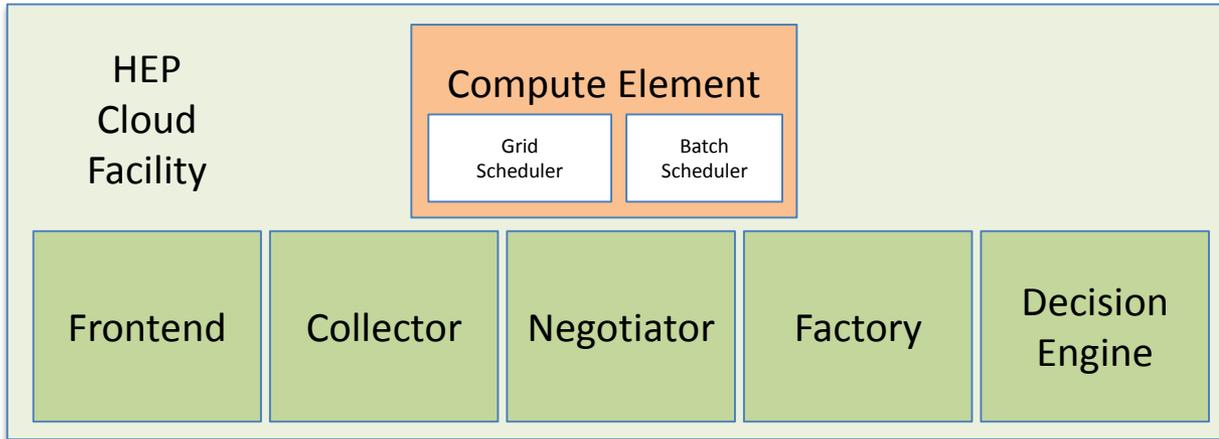
VO  
Infrastructure



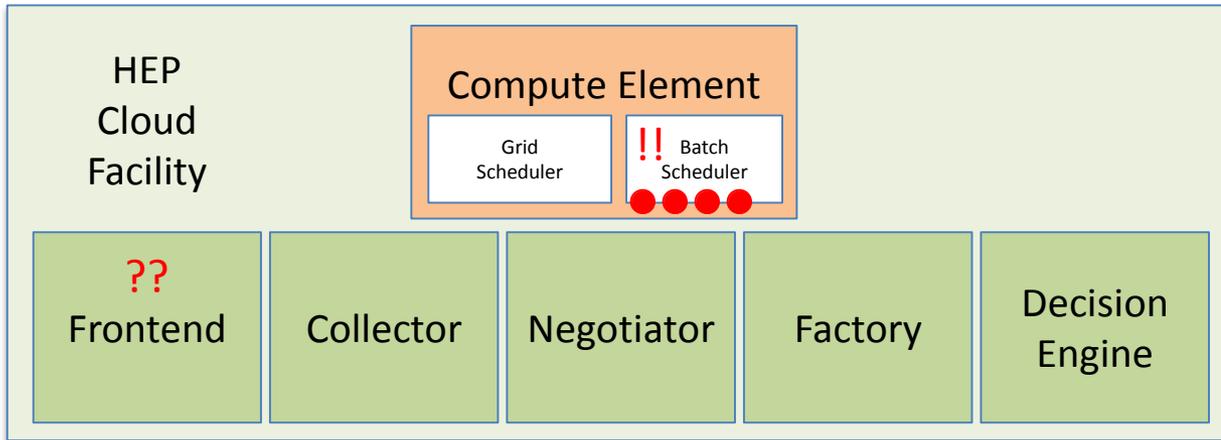
CMS  
Workflow  
Agent

1. User submits set of jobs to VO-specific job management agent  
(In our example the VO is the CMS Experiment at the LHC).

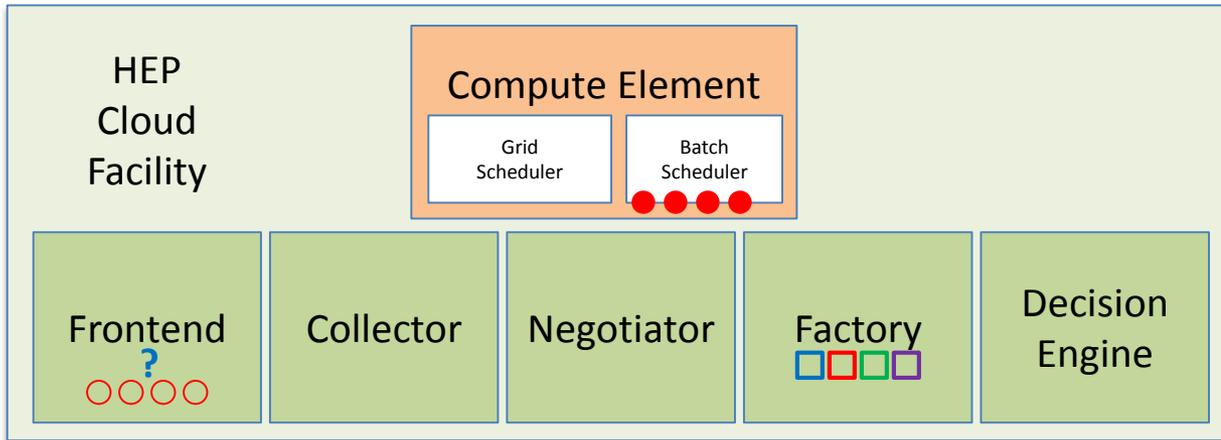
VO  
Infrastructure



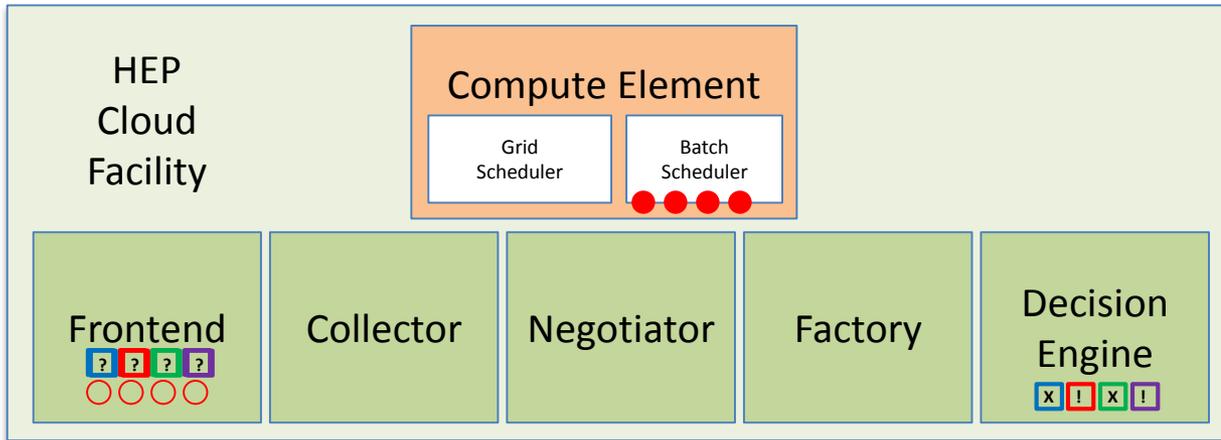
2. Job management agent releases jobs to Compute Element Grid Scheduler.
3. Compute Element checks authentication and authorization of jobs, then passes jobs to Compute Element Batch Scheduler.



4. Frontend polls Compute Element Batch Scheduler for jobs.
5. Compute Element sends Frontend a set of job identifiers with their corresponding resource requirements.

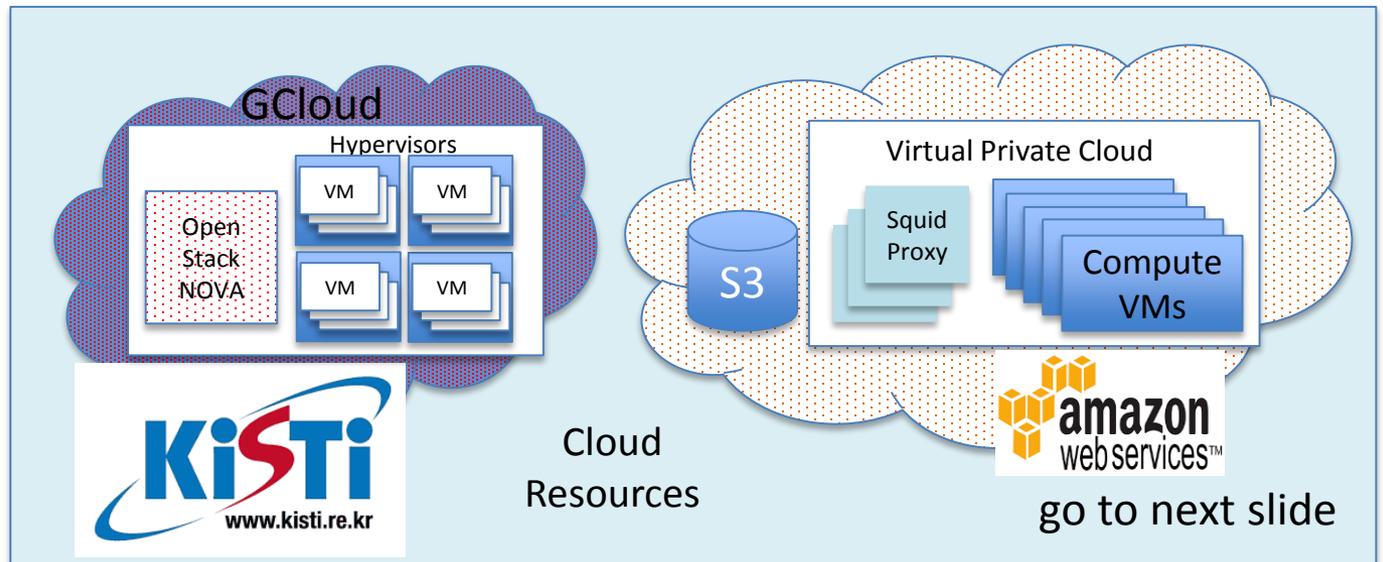
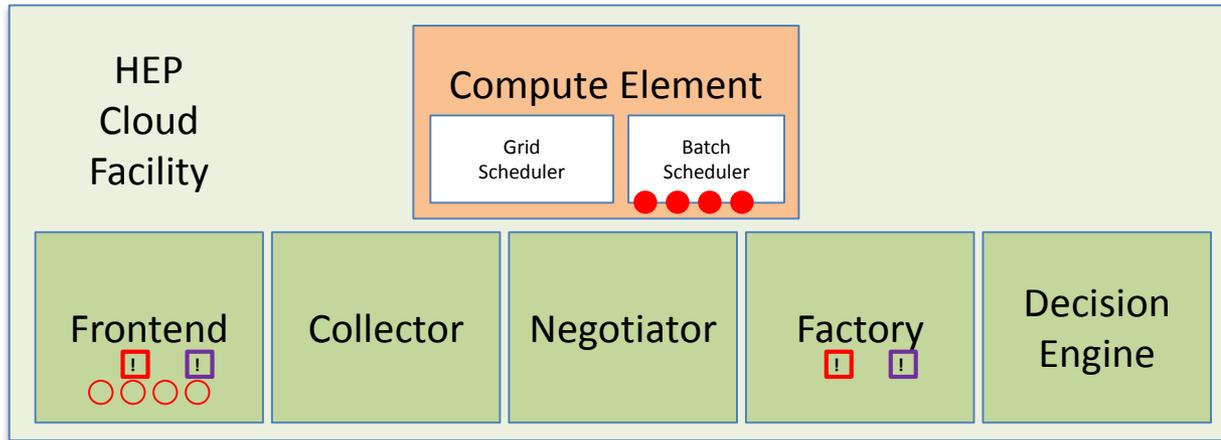


6. Frontend asks Factory for list of entries with description of acceptable resources.
7. Factory returns list of entries with appropriate resources.



8. Frontend asks Decision Engine whether it should request pilot jobs (VMs) from each potential resource.
9. Decision Engine answers Yes or No for each resource.

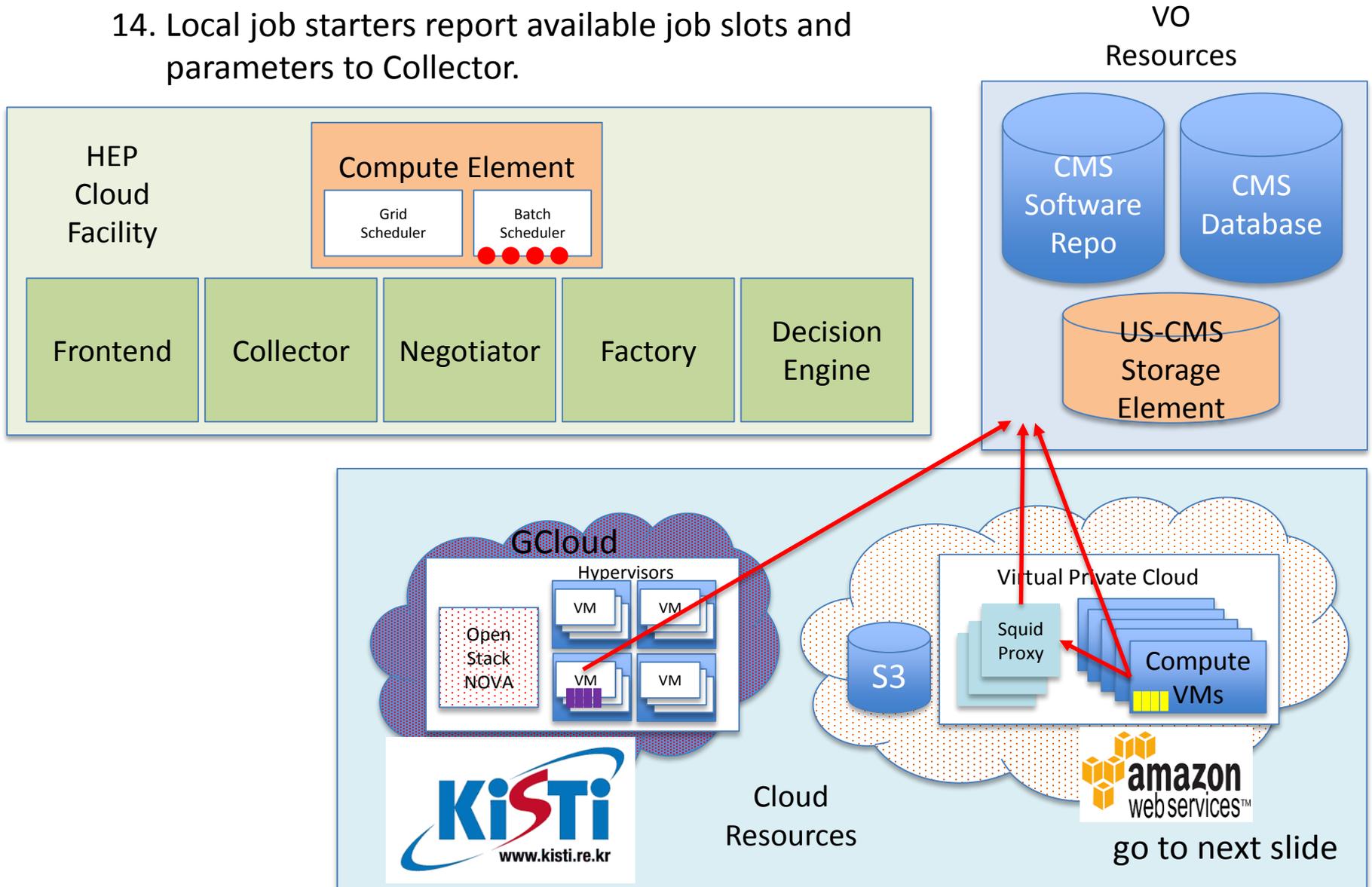
10. Frontend asks Factory to provision resources.
11. Factory requests pilot jobs (VMs) from selected resources.
12. Resource instantiates pilot jobs (VMs).



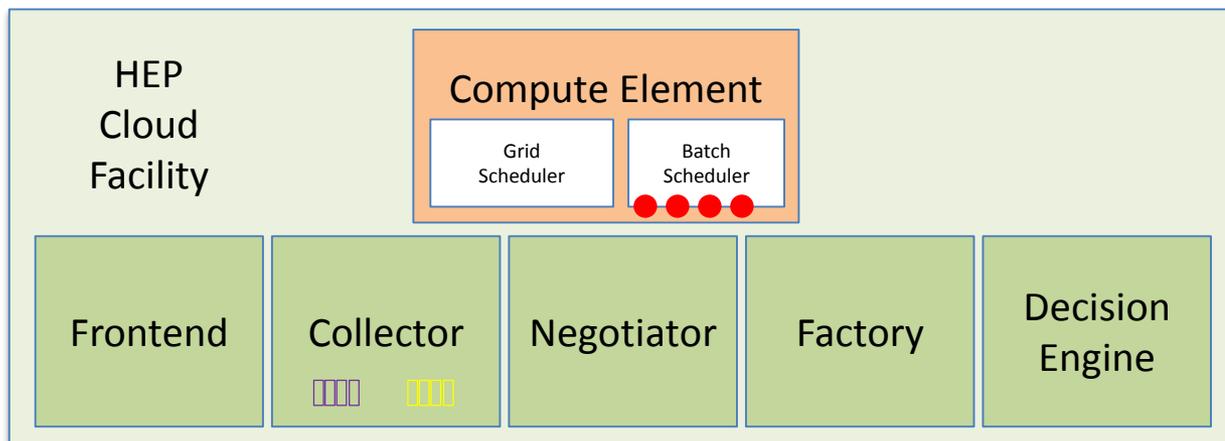
13. At pilot job (VM) creation:

- Location of squid servers, VO resources set
- Local job starter created.

14. Local job starters report available job slots and parameters to Collector.



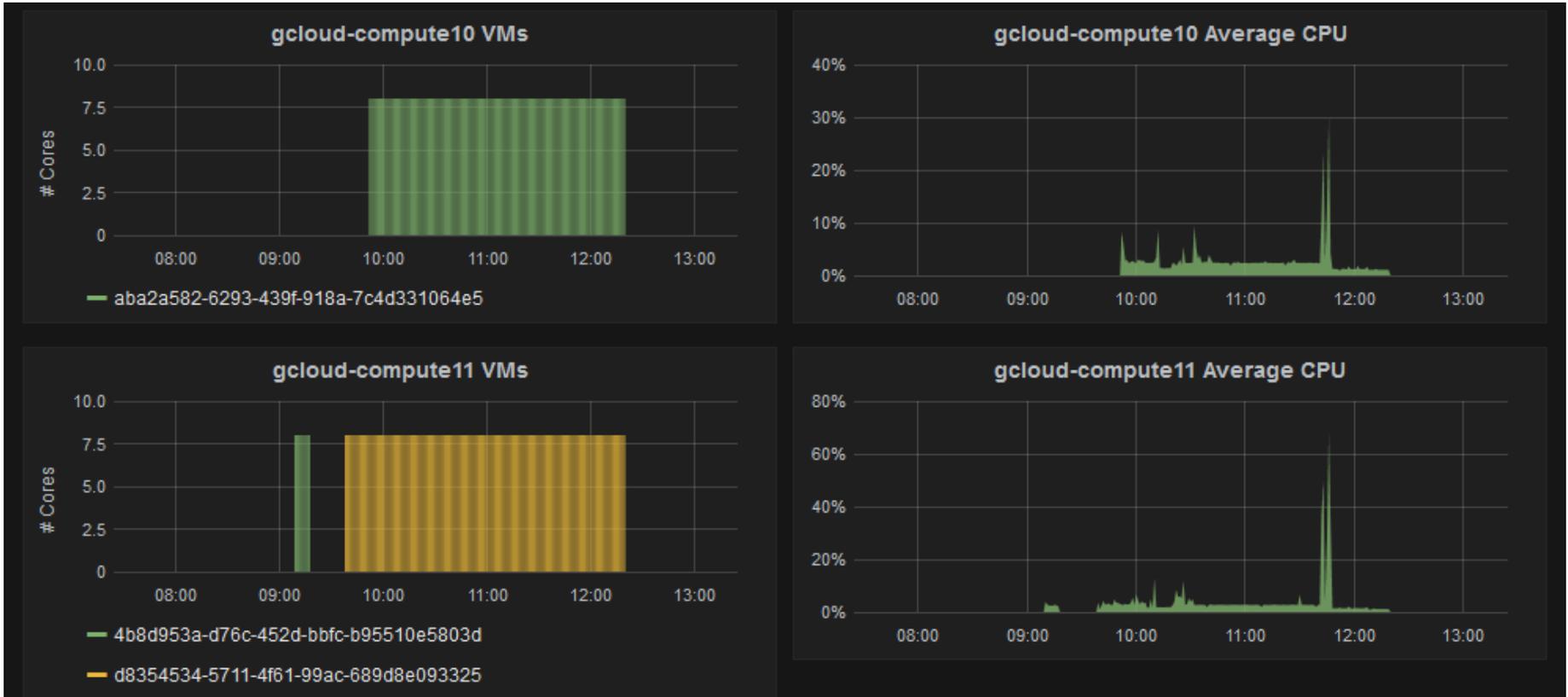
Pause for a moment: what is the current status?



- Batch Scheduler on Compute Element has a list of jobs to be run
- Decision Engine has determined appropriate resources
- Factory has provisioned pilot jobs (VMs) on those resources
- Pilot jobs (VMs) have created local job starters and reported available job slots to Collector

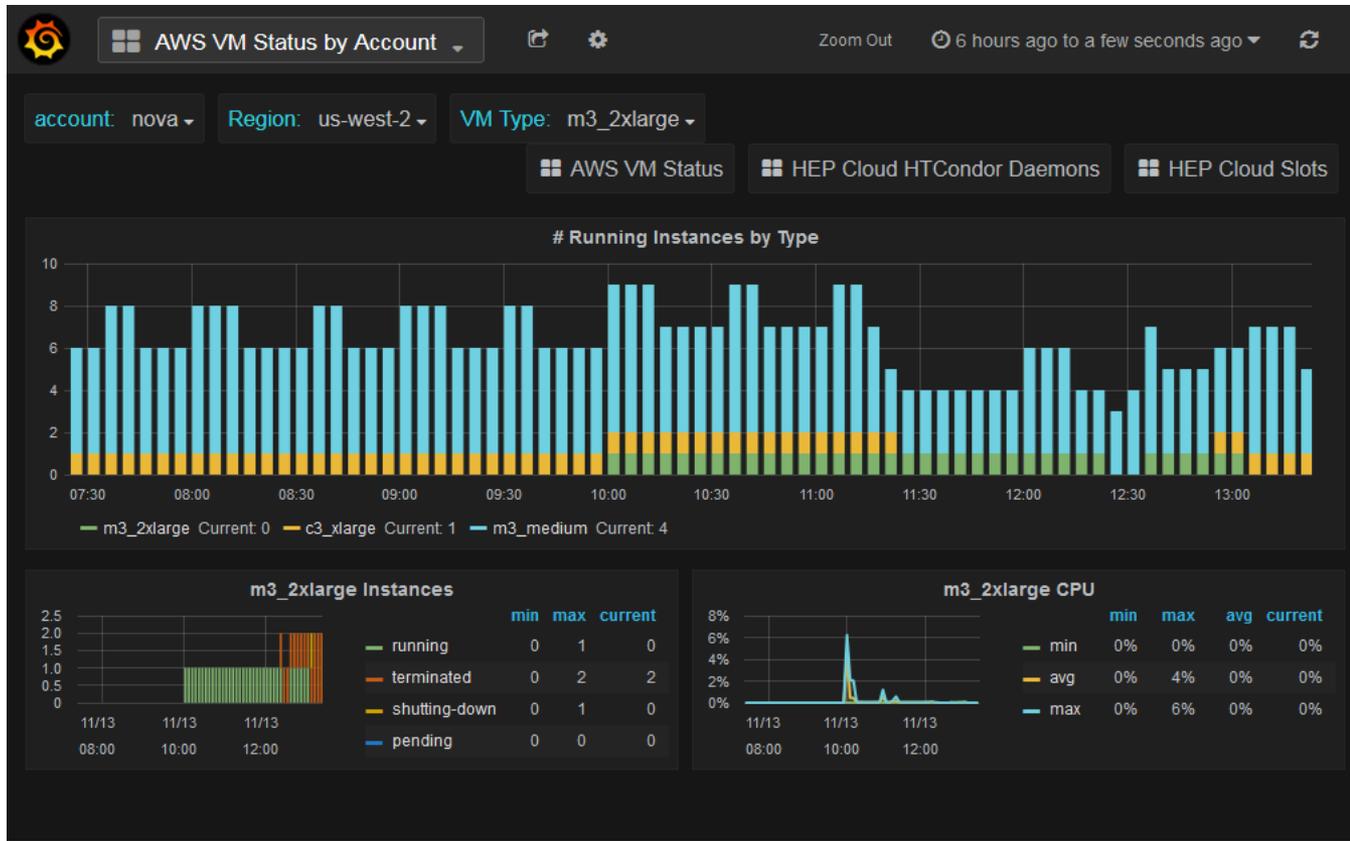
# At this time can use monitoring to view VM creation (1/2)

This shows a 8-core VMs created on two separate GCloud hosts:



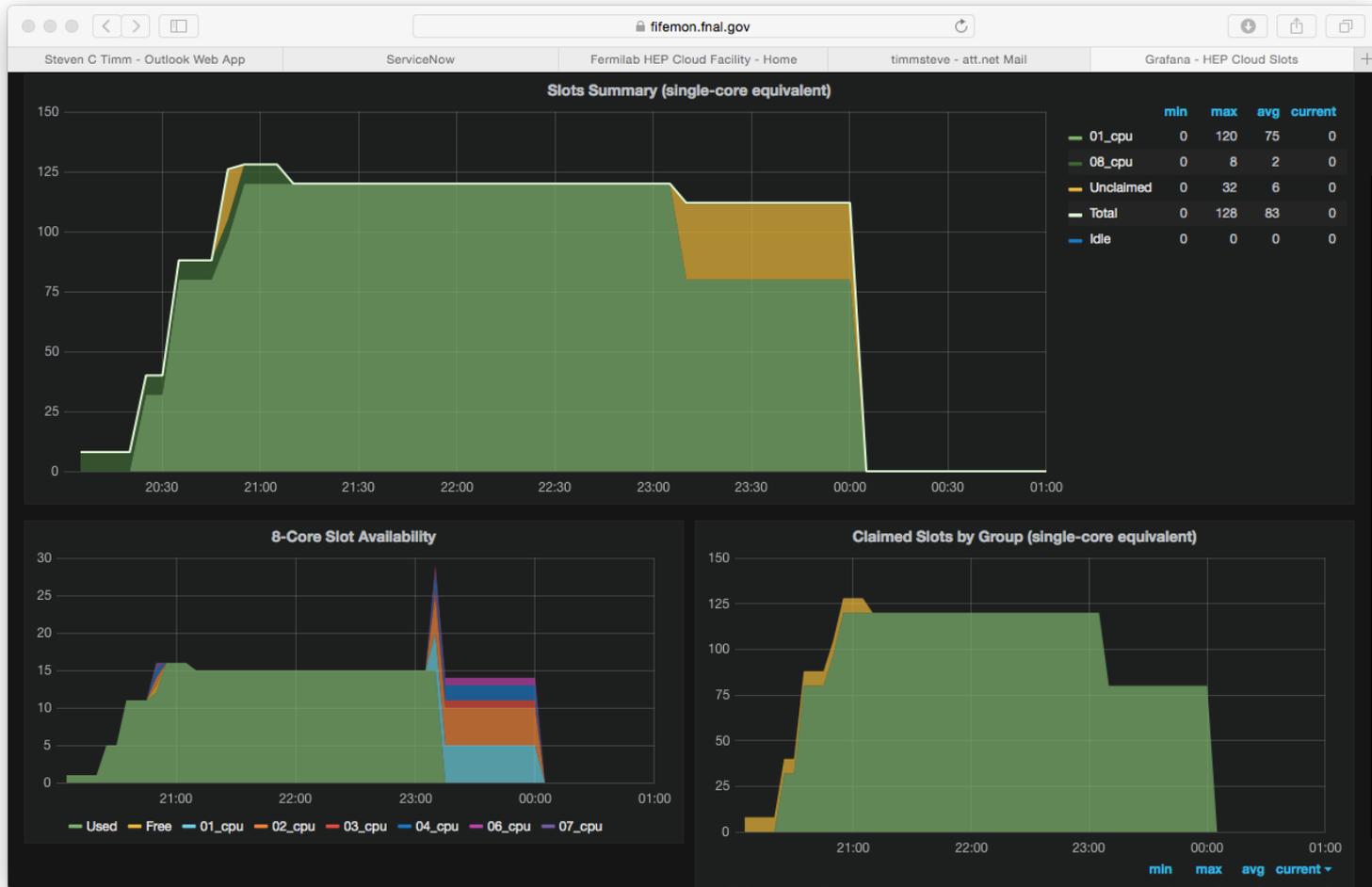
# At this time can use monitoring to view VM creation (2/2)

This shows VMs created on AWS us-west-2 of type m3\_2xlarge:



# Monitoring shows status of jobs

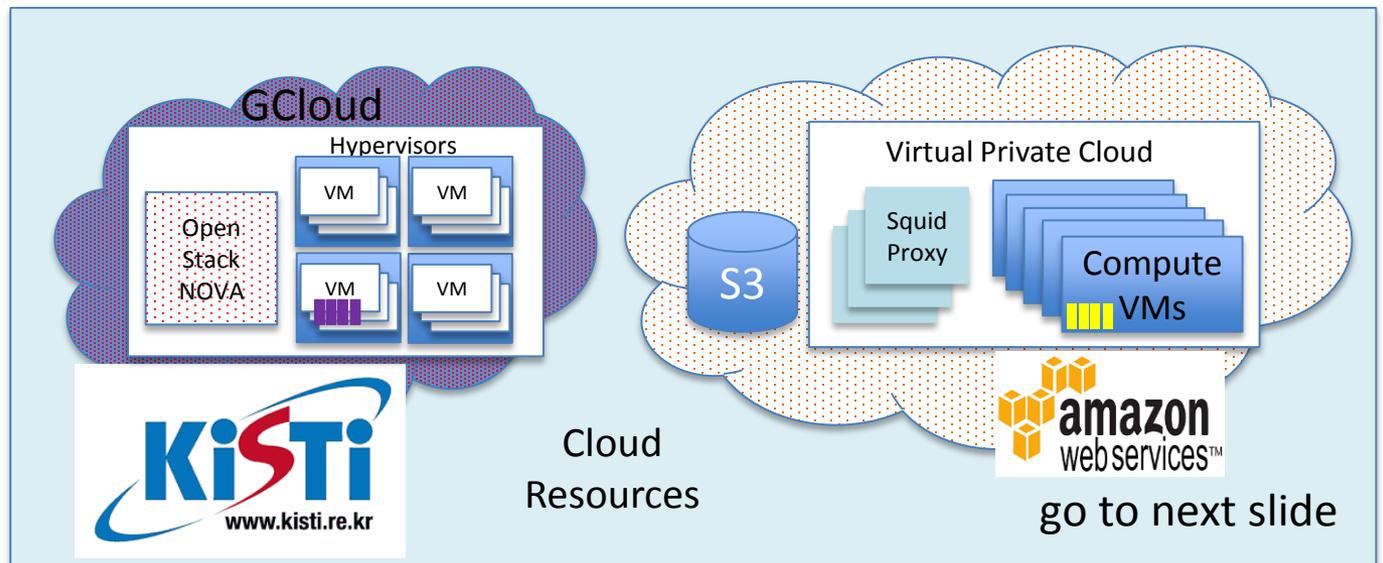
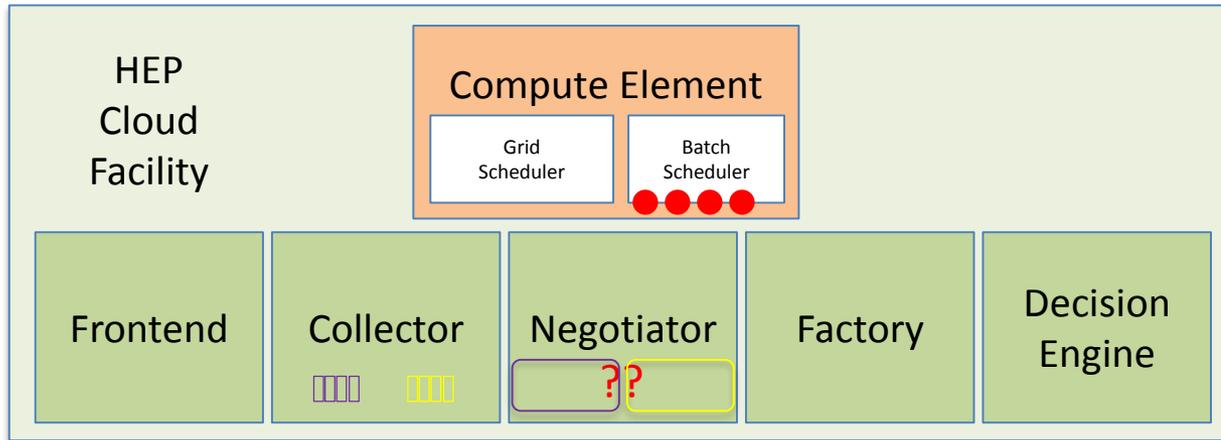
“Available” are open job slots, “Claimed” are running jobs



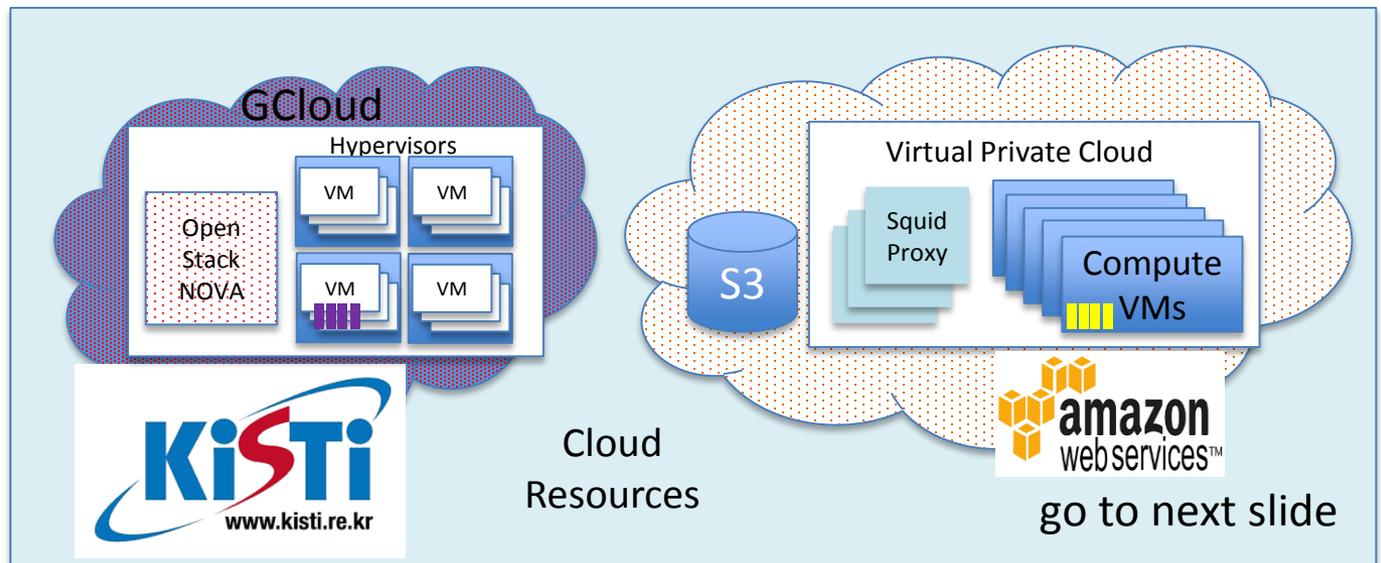
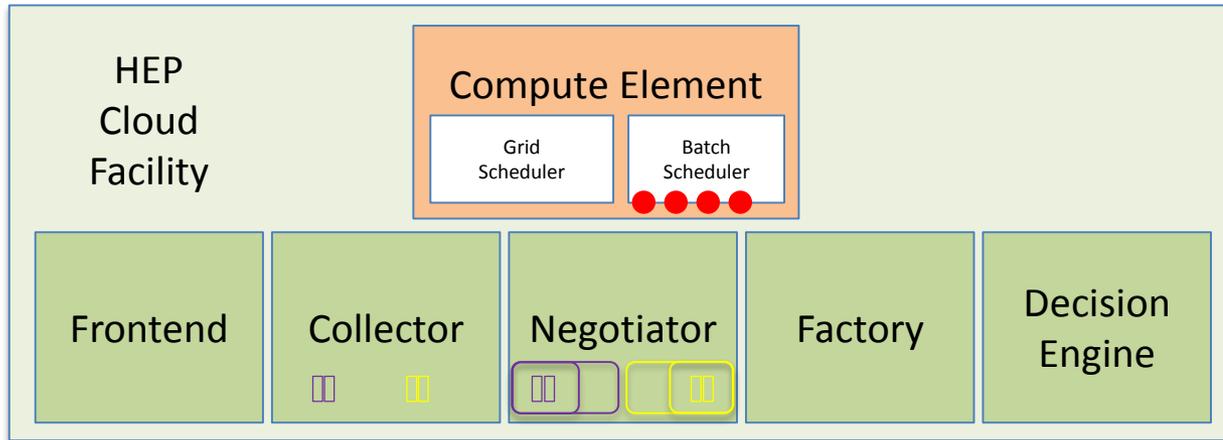
# Back to the animation...

---

15. Negotiator polls Compute Element to look for jobs.
16. Negotiator asks Collector for available job slots and matches job requirements to job slot parameters.

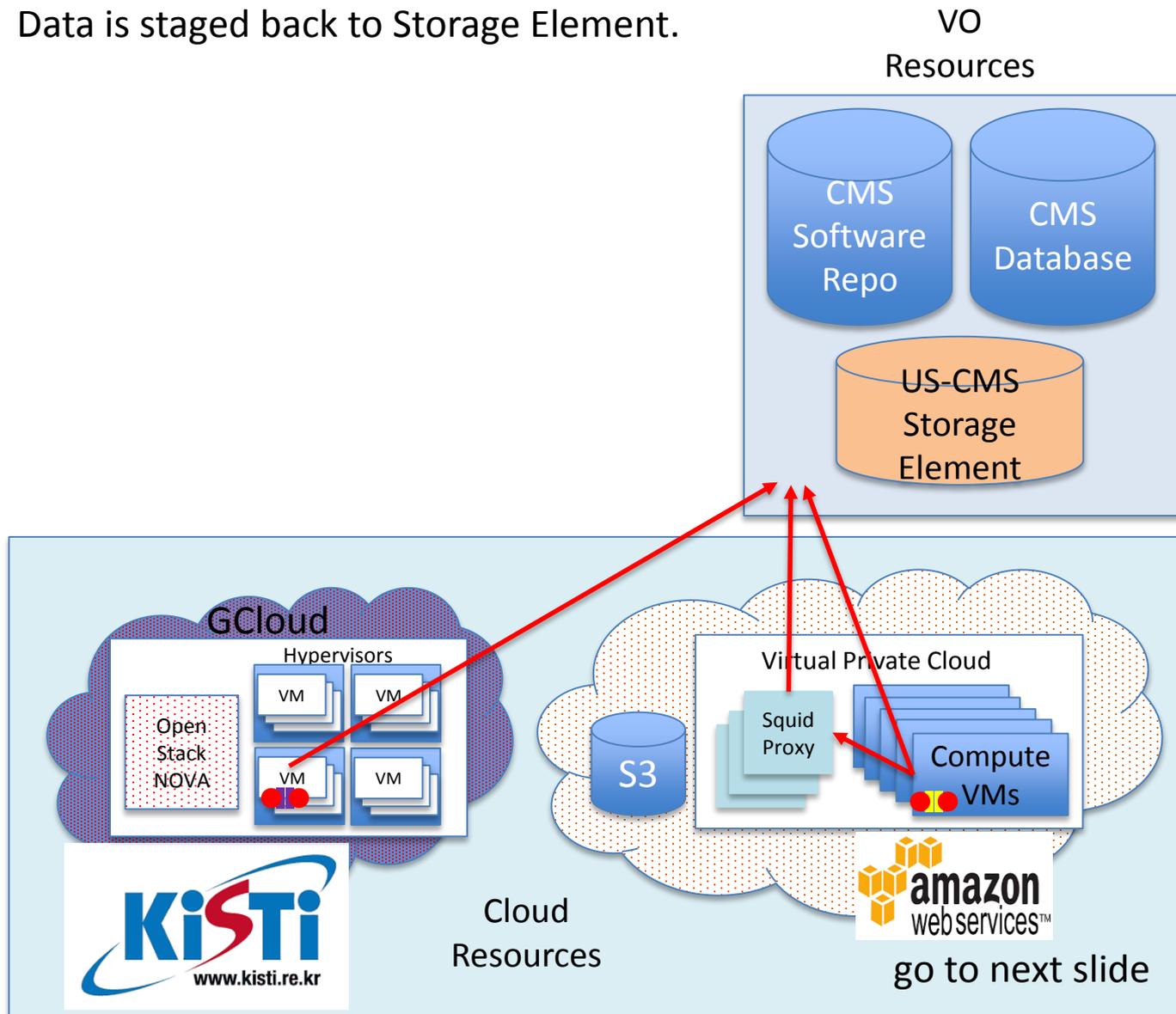


17. Negotiator informs Compute Element of resource matches.
18. Compute Element communicates with job starter on matched VMs to start matched jobs. Jobs start!
19. VM updates Collector with reduced job slot count.



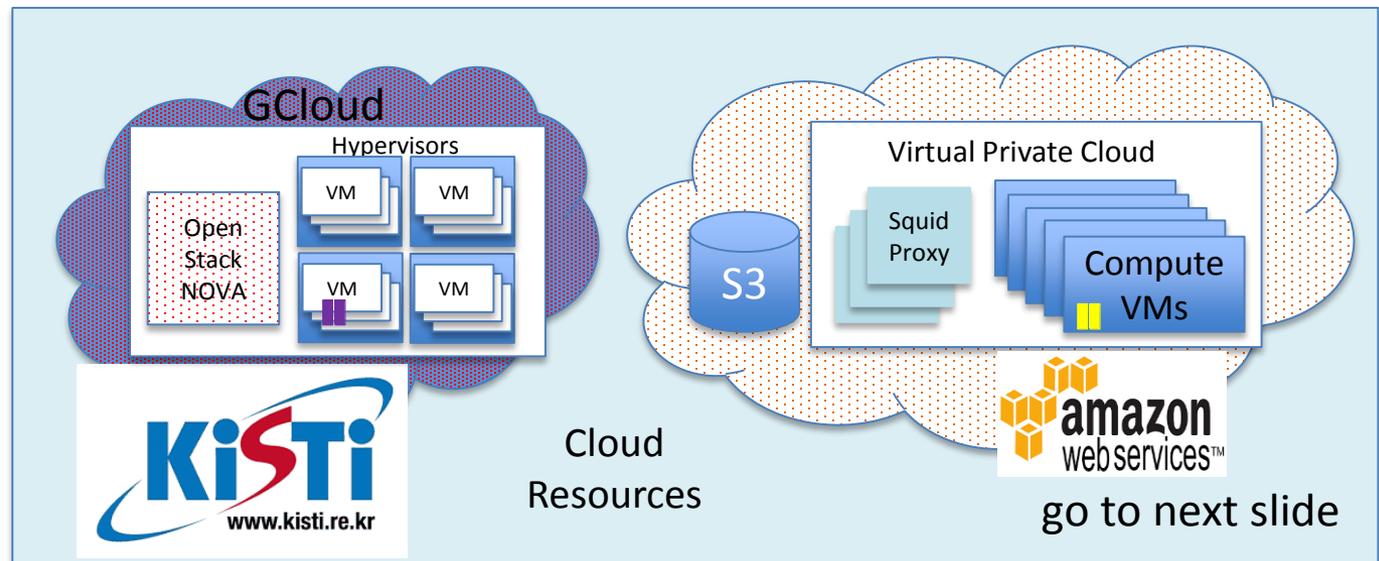
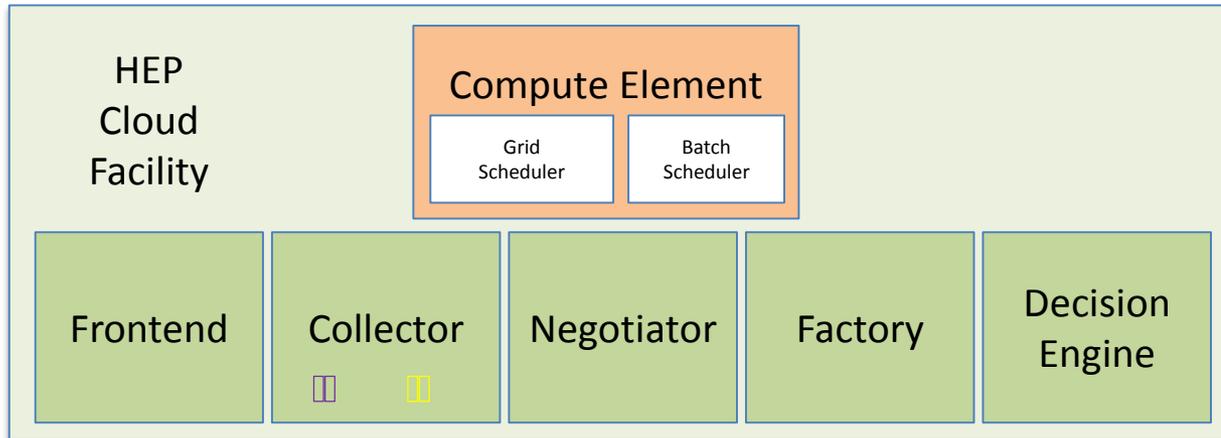
20. Jobs run! Communicate with VO Resources as necessary and as configured at VM start.

21. Jobs finish. Data is staged back to Storage Element.



22. VM updates Collector with increased job slot count.

23. VM expires after idle period, job slots decreased.



VO  
Infrastructure

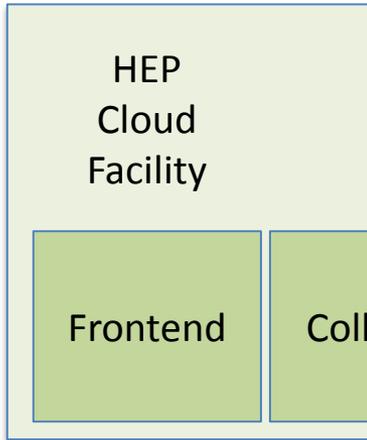


CMS  
Workflow  
Agent

Jobsub  
Server



VO  
Resources



System is ready for more jobs!

Thanks for watching this demo.

