Cloud Interoperability:
Extending On-Demand Services to Public Clouds

Alessio Balsini

Scientific Computing Division

Grid & Cloud Services Department

Final Presentation, Oct. XX 2014

Training Program

- **Brief Description of Trainee/Intern’s Role for this Phase**
  - The Intern will assist with extending the Cloud computing platforms available to Fermilab and in this way, will help increase the amount of computing resources available for scientific data processing. Cloud computing is a form of Internet-based computing, whereby shared servers provide resources, software, and data to computer and other devices on demand. It is an important new paradigm of distributed computation and is highly popular with public and commercial institutions.
- **Specific Goals and Objectives for this Phase**
  - The intern will learn how to access remote resources at a variety of commercial and community Cloud providers (such as Google Cloud, RackSpace, and Microsoft Azure). He will evaluate programmatic and web interfaces to provision resources at these providers, and will document the procedures and lessons learned from performing these tasks. By the end of this phase, the intern will be able to use his knowledge on Cloud computing to enhance existing or new computational programs in the industry or academia.
- **Final Objectives in Detail**
  - The intern will learn how to provision remote resources at both commercial and community Cloud providers. This will enable the student to develop skills in Cloud management software and tools, learn techniques on interacting with Cloud management systems, and be able to generalize the skills learned to acquire a higher level of knowledge on distributed computing paradigms. Major Equipment to be used: FermiCloud, and Infrastructure as a Service Computing platform dedicated to scientific computing at Fermilab. Major Computer Software to be used: Linux OS, Shell, Ruby, OpenNebula, and Condor.

Outline

- Problem
  - *Fermilab* requirements
- Solution
  - Fermilab infrastructure
  - Solutions in the market
- My contribution

Problem
Main Drivers

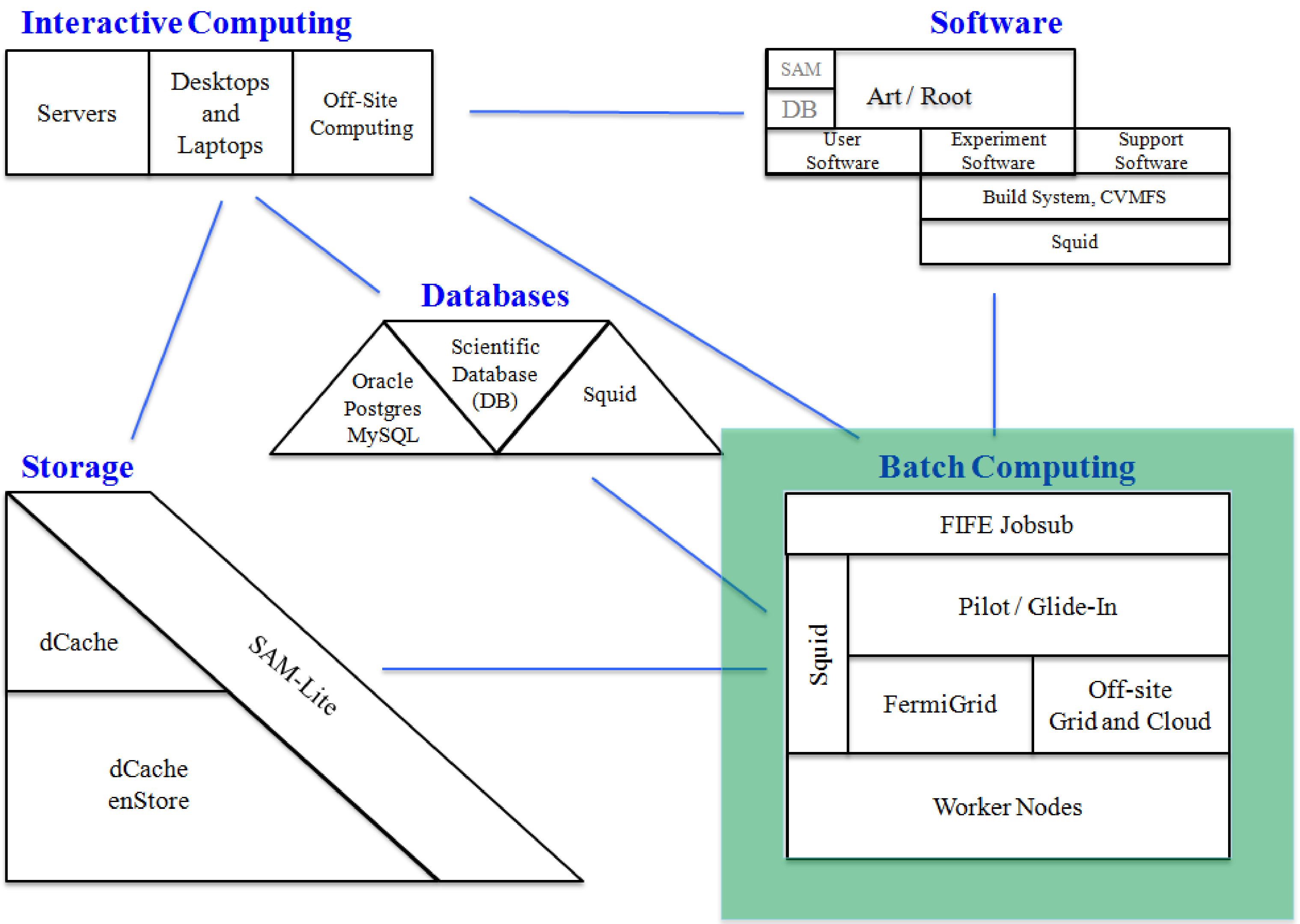
- User
  - Single researcher
    - Simple, but effective way of submitting jobs
  - Experimental communities
    - Resources independent (in performances and costs) from other experimental communities
  - Fermilab institution
    - Needs mechanisms to manage the needs listed above
- Computational requirements
  - Constantly Increasing
  - Not uniform during time
  - Experiments are changing
    - From few, big experiments: *CDF*, *D0*
    - To many, small experiments: *ArgoNeuT*, *Muon g-2*, *LBNE*, *MARS*, *MicroBooNE*, *Minerva*, *Minos*, *Mu2e*, *Nova*, …
- Computational infrastructure must be
  - Flexible
  - Agile

Solution
Fermilab Scientific Computing Architecture

- Scientific Computing Division aims at integrating experiments applications into a common architecture of services
  - Easy to manage
  - Powerful
  - Scalable

- The result is a big, distributed, architecture
  - I focused on a small part of it

Solution
Fermilab Scientific Computing Architecture



Solution
Fermilab Scientific Computing Architecture

- This requires a flexible management software
  - Job submission
    - Job submission client: *HTCondor,* or *JobSub*
  - Jobs execution
    - *Job Management System*: *HTCondor*
  - Virtual Machines dynamic creation and deletion
    - *Workload Management System*: *HTCondor* + *GlideinWMS*
  - Virtual Machines initialization
    - *Configuration Management System*: *Puppet*

Solution

Cloud Computing: an Integral Component of the Architecture

- Here comes the Cloud
  - Much more flexible architecture
  - Fewer loss of resources
  - External providers pricings
    - Constantly decreasing and now competitive with current internal resources maintenance costs

- Fermilab already has
  - Grid
  - Internal Cloud (*Fermicloud*)
  - External Cloud provider (*AWS*, *Amazon Web Services*)

Solution
New Perspectives with New Providers

- Risk of vendor lock-in
  - Pricing
  - *QoS* (*Quality of Service*)
  - System requirements may change
  - New Cloud technologies

- So, exploring solutions by other providers
  - *Google Cloud* (*Google Compute Engine*)
  - *Microsoft* (*Windows Azure*)

My Contribution
Overview

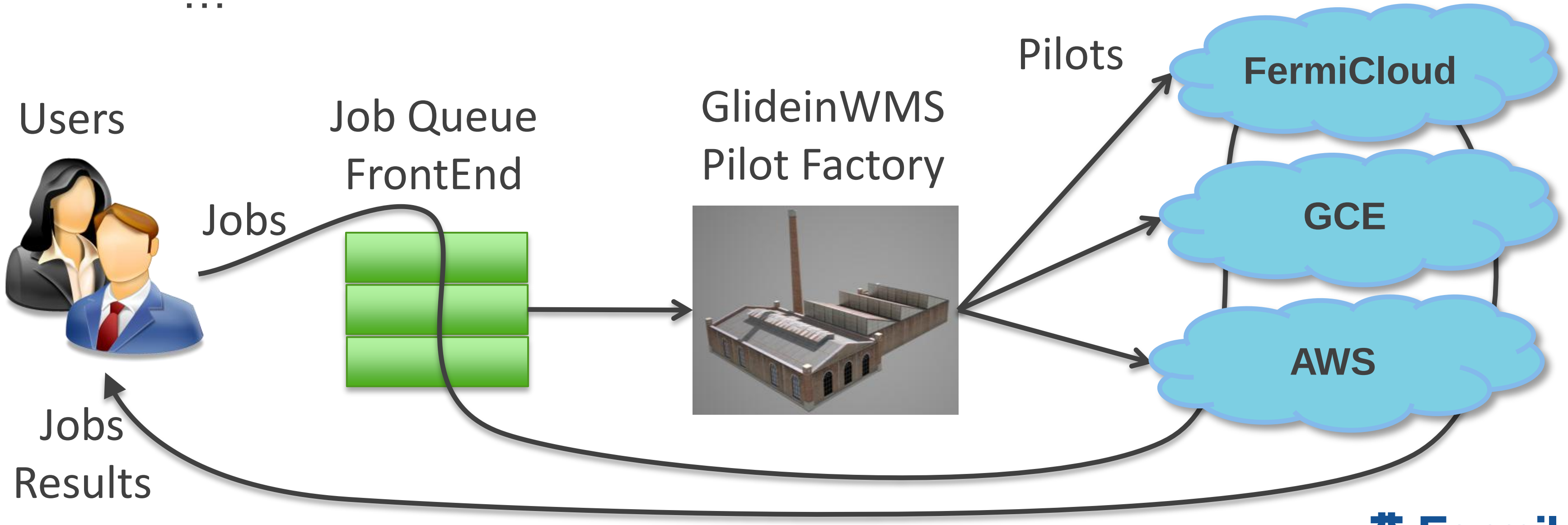
- Exploring, Testing and Reporting
  - Virtual Machines management through: *web*, *command-line* and *REST APIs* (*Application Programming Interfaces*) for
    - *Amazon Web Services*
    - *Microsoft Windows Azure*
    - *GCE* (*Google Compute Engine*)
  - Jub submission tools used by researchers and their extensions for the cloud

- Extending Workload Management
  - *HTCondor* / *GlideinWMS* for *GCE*

- Improving Computational Environment Contextualization
  - Virtual Machines initialization script

My Contribution
Use Case Result

- Previous architecture: users were able to run scientific workflows only on internal resources and *Amazon Web Services*
- New improvements I contributed to: freely choose different cloud providers
  - Switching based on preferred effectiveness metrics
    - Efficiency
    - Cost
    - Human involvement
    - …



Thank you!

Alessio Balsini

alessio.balsini@gmail.com

Appendix: What I Did in Detail
Exploring and Reporting

- Virtual Machines management with
  - *AWS*
  - *GCE*
  - *Azure*
- And their *APIs*
  - *REST*
  - *Command-line*
  - *Web*
- For
  - disk customization
  - instances management
  - parameter passing: *user metadata* and *custom data*
- *Google Cloud*'s specific:
  - *OAuth2.0* authentication
  - Replica-pools
  - Autoscaler

- *Fermilab*'s infrastructure
  - Job Submission Tools clients
    - *HTCondor*
    - *GlideinWMS*
    - *Jobsub v0.4*
      - *Grid*
      - *FermiCloud*
      - AWSResources
    - *FemiCloud*
    - *Fermilab*'s Grid
- Useful stuff
  - *Ruby*
  - *Puppet*

Appendix: What I Did in Detail
Extending and Testing

- Extending
  - *HTCondor*'s OAuth2.0 authentication with *Google Cloud*
  - Supported development of *Puppet* script for Virtual Machines initialization in
    - *FermiCloud*
    - *AWS*
  - Supported development of shell scripts for Virtual Machines initialization in *AWS*
- Found and notified bugs
  - *jobsub* parameter acceptance: solved
  - *FermiCloud* jobs starving: solved
  - *AWS* jobs starving: solved
  - *HTCondor*'s *GCE* authentication procedure through *gcutil*: pending
  - *GCE*'s authentication failure through *REST API*: pending
    - But found a workaround: Python API authentication