

JOBSUB ARCHITECTURE

Parag Mhashilkar

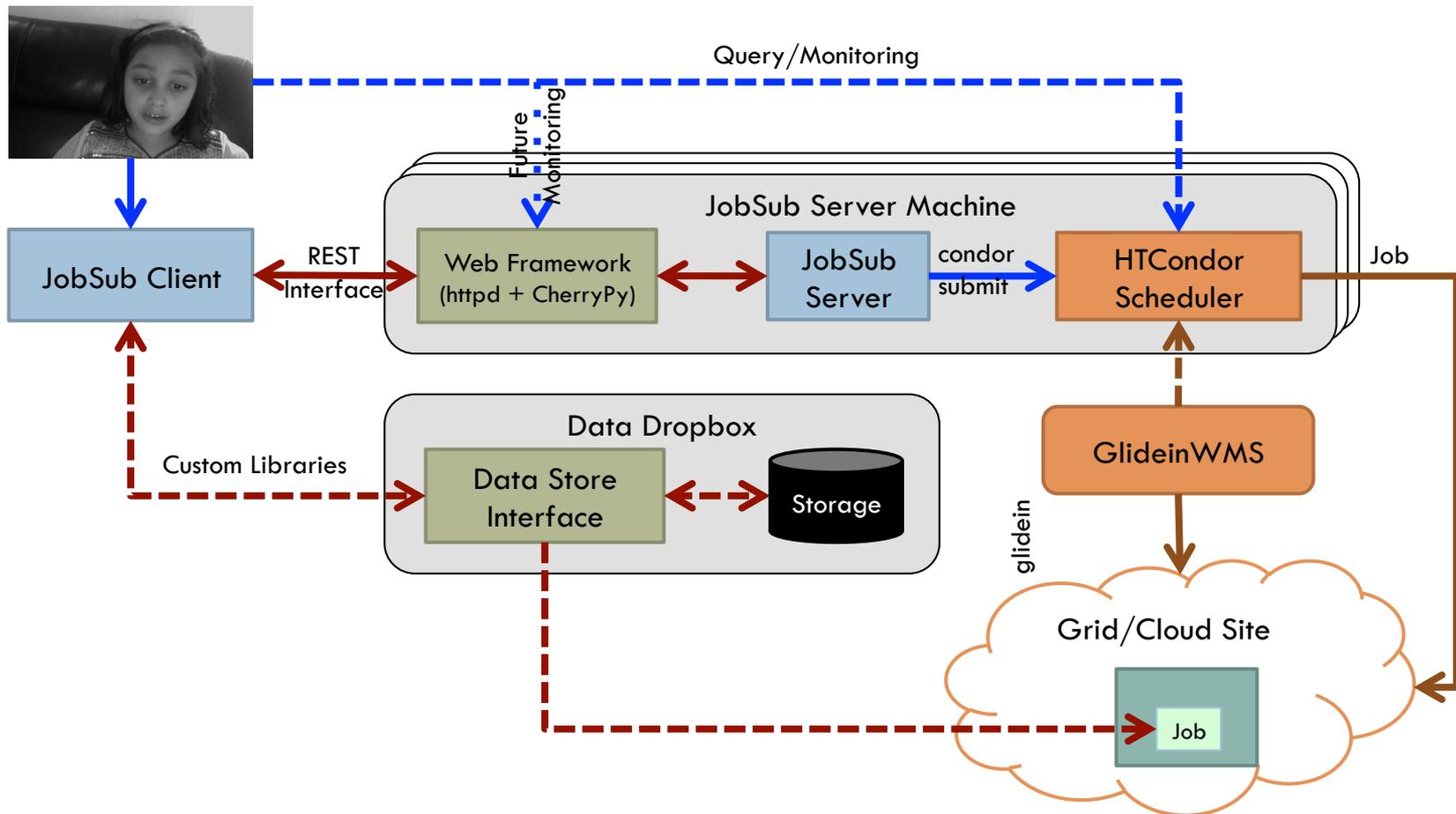
JobSub Architecture

2

- Based on the input from -
 - FIFE Architecture Committee Report: CS-doc-5180
 - Meetings/Conversations
 - Authentication: Kevin Hill & Mine Altunay
 - CMS CRAB Framework: Eric Vaandering
 - Possible Cloud/Future Authentication mechanisms: Igor Sfiligoi
 - Ssh & sshfs
 - <https://indico.cern.ch/getFile.py/access?contribId=322&sessionId=9&resId=0&materialId=poster&confId=214784>
 - New CRAB Analysis Framework
 - <https://indico.cern.ch/contributionDisplay.py?contribId=113&sessionId=5&confId=214784>
 - Non-Kerberos sshd not allowed (Old policy, still valid?)
 - <http://www.fnal.gov/docs/strongauth/policy.html>

JobSub: Client-Server Architecture

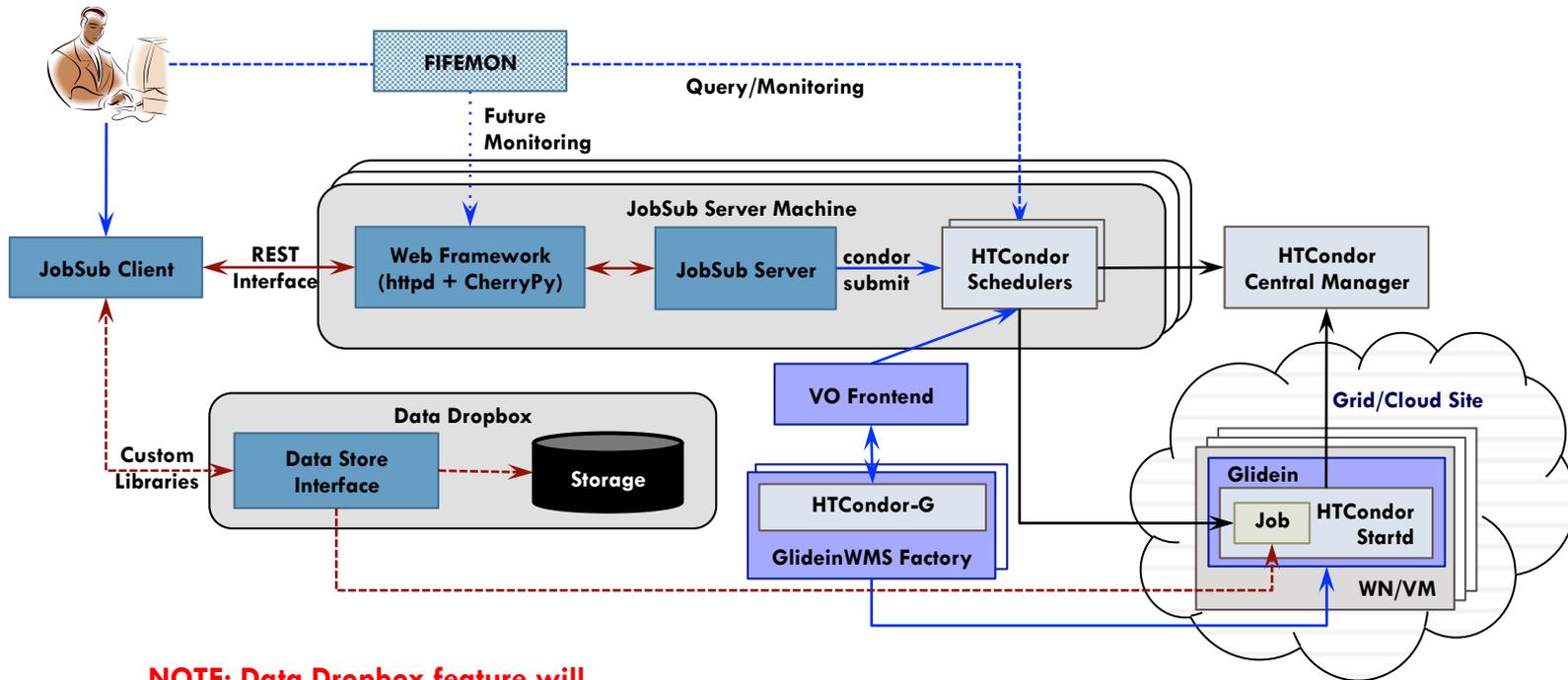
3



Last Updated: November 14, 2013

JobSub: Client-Server Architecture with GlideinWMS Services

4



NOTE: Data Dropbox feature will be implemented in future releases.

Architecture Highlights

5

- Modular
 - ▣ Components can be easily replaced/upgraded
 - ▣ Central JobSub server:
 - Accepts requests using a well defined REST-Like API
 - Support for multiple client types
 - Command line (Only client supported by default in the beginning)
 - Browser
 - Portlet/App clients
 - Clients get changes faster with minimal need to update software
- Fault Tolerant
 - ▣ Minimal dependency between the components

Architecture Highlights ...

6

- Network Centric
 - ▣ Thin Client
 - Requires minimal software on the client except python, curl, ssl (cilogon client if used)
 - REST APIs for communicating with the services
 - Can be installed on-site and off-site
- Scalable
 - ▣ Services can be deployed in HA mode
 - LVS
 - HTTPD
 - HTCondor
 - Stateless JobSub Server
 - Storage interface with SQUID

Architecture: Alternatives

7

- Alternative 1: Thick client & Thin Server using curl+https
 - Cons
 - Operational overhead with more services to install on the client
 - Upgrading services requires more coordination
 - Deployment may get complicated
 - Increases inter-dependency between services
 - Pros
 - HTTPS is a industry standard
 - Forward looking and supports several authentication mechanisms
 - Does not require direct user accounts/login into the server machines (Operations group request)
- Alternative 2: Thick client & Thin Server using gsissh+gssshd
 - Cons
 - Same as Alternative 1
 - Client - Server communication locked into using x509 credentials
 - Requires direct user accounts/login into the server machines
 - Server requires gssshd and to accept x509 credentials - Against computing policy (sshd must support Kerberos only authentication)

Architecture: Alternatives ...

8

- Alternative 3: Thin client & Thick Server using gsissh+gssisshd
 - Cons
 - Client - Server communication locked into using x509 credentials
 - Server requires gssisshd and to accept x509 credentials - Against computing policy (sshd must support Kerberos only authentication)
 - Requires direct user accounts/login into the server machines
 - Pros
 - HTTPS is a industry standard
 - Forward looking and supports several authentication mechanisms
 - Does not require direct user accounts/login into the server machines
- Selected Design
 - Pros of following schemes
 - Thin client
 - curl+https

High Level Tasks

9

High Level Tasks	Phase	Status
Prototype using Web framework (Using Django & CherryPy)	1	DONE
Design REST APIs & client-server communication protocol	1	In-Progress
Create plumbing code <ul style="list-style-type: none">Thin Client using curlWeb framework - server plumbing	1	In-Progress
Data Dropbox: To allow for custom input data & libraries <ul style="list-style-type: none">Understand the requirementsDesign, develop & deploy	2	Not Started
Improve JobSub code maintainability <ul style="list-style-type: none">Change the server to make experiment settings configurableMove to condor python bindingsImplement monitoring through web framework	1 & 2	In-Progress