Fermilab | U.S. DEPARTMENT OF ENERGY | Office of Science

Grid submission tutorial

Mike Kirby

FIFE Workshop

June 22, 2017

FABRIC FOR FRONTIER EXPERIMENTS

Fermilab

Head to the Open Science Grid???

Centralized Services from FIFE

FABRIC FOR FRONTIER EXPERIMENTS

- Submission to distributed computing – JobSub, GlideinWMS frontend
- Processing Monitors, Alarms, and Automated Submission
- Data Handling and Distribution
  - Sequential Access Via Metadata (SAM) File Transfer Service
  - interface to dCache/Enstore/storage services
  - Intensity Frontier Data Handling Client
- Software stack distribution – CERN Virtual Machine File System (CVMFS)
- User Authentication, Proxy generation, and security
- Electronic Logbooks, Databases, and Beam information
- Integration with future projects, e.g. HEPCloud

Centralized Services from FIFE

FABRIC FOR FRONTIER EXPERIMENTS

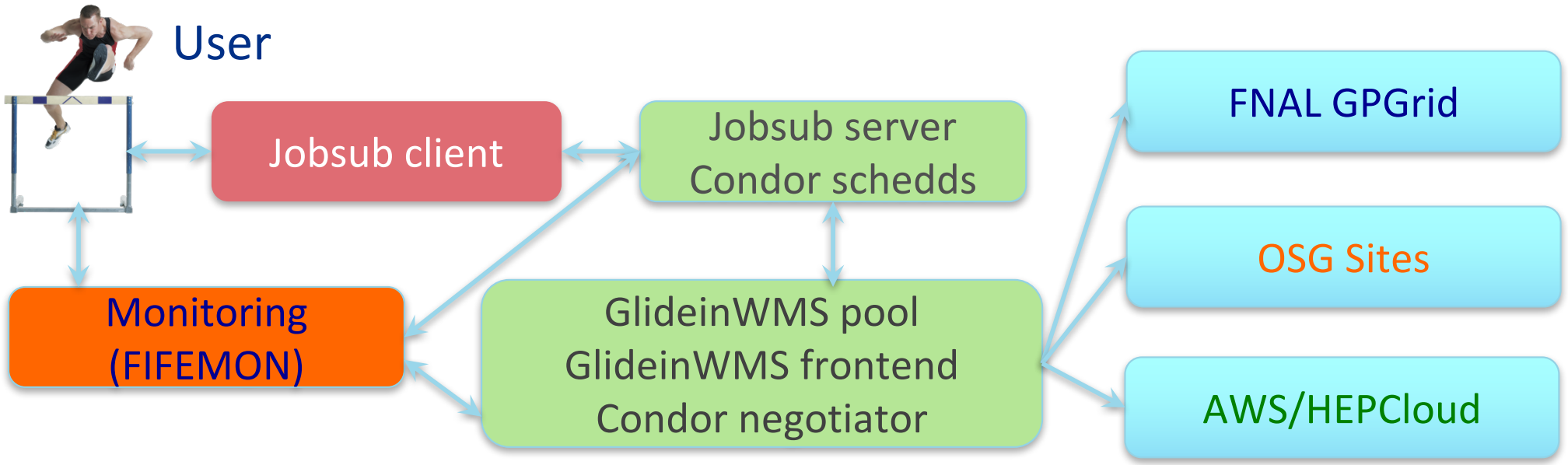
- Submission to distributed computing – JobSub, GlideinWMS frontend
- Processing Monitors, Alarms, and A
- Data Handli
  - S
  - i
  - Ir
- So
  - Syst
- Use
  - , Proxy generation, and security
- Electronic Logbooks, Databases, and Beam information
- Integration with future projects, e.g. HEPCloud

Most of this is already setup for you…

- So what do you really need to know about before you submit your jobs?
- How do you use that information?

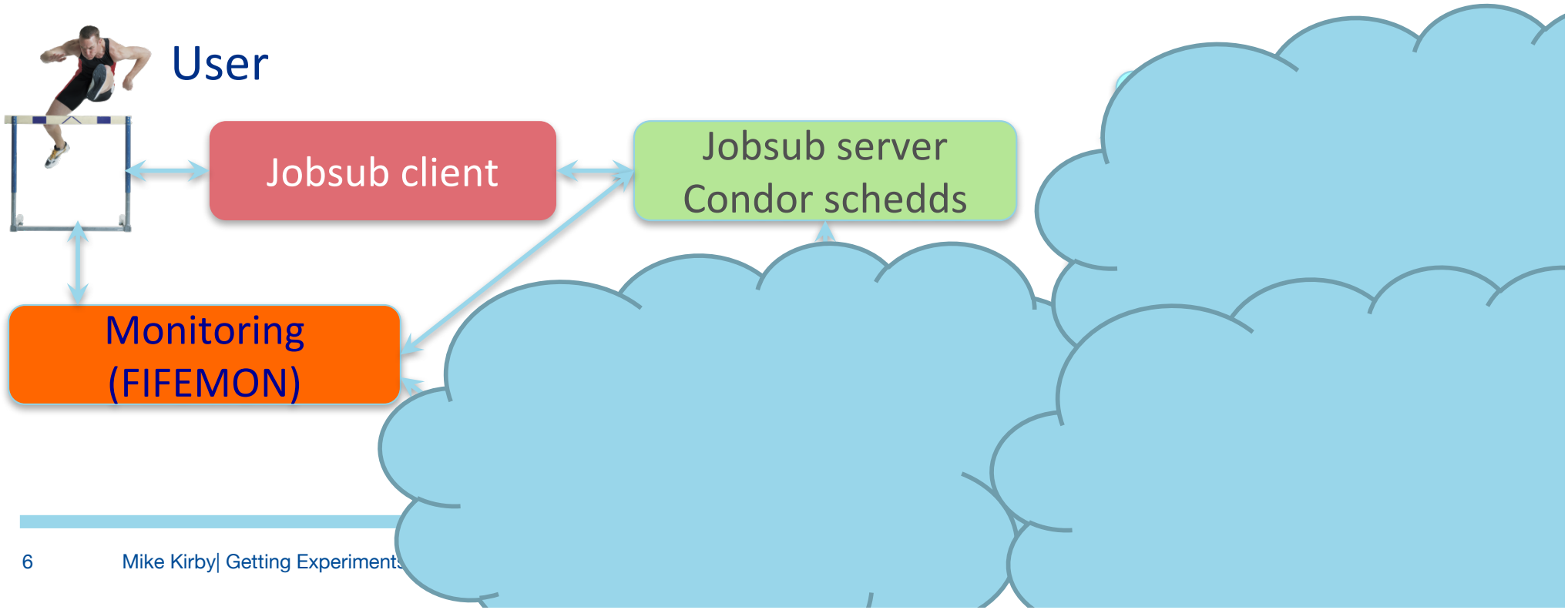
Job Submission and management architecture

- Common infrastructure is the **fifebatch** system: one GlideInWMS pool, 2 schedds, frontend, collectors, etc.
- Users interface with system via “jobsub”: middleware that provides a *common tool across all experiments*; shields user from intricacies of Condor
  - Simple matter of a command-line option to steer jobs to different sites
- Common monitoring provided by FIFEMON tools
  - Now also helps users to understand why jobs aren’t running



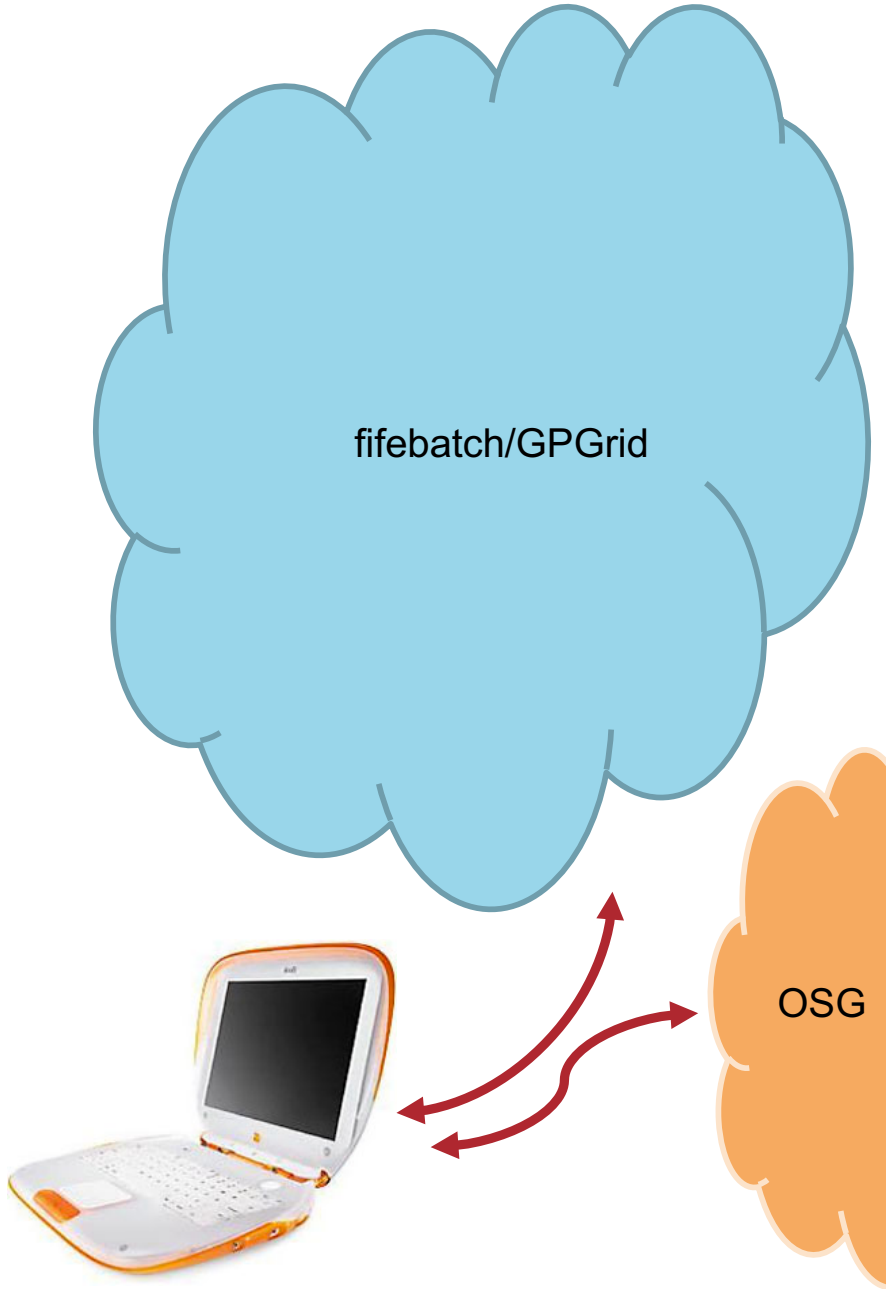
Job Submission and management architecture

- Common infrastructure is the **fifebatch** system: one GlideInWMS pool, 2 schedds, frontend, collectors, etc.
- Users interface with system via “jobsub”: middleware that provides a *common tool across all experiments*; shields user from intricacies of Condor
  - Simple matter of a command-line option to steer jobs to different sites
- Common monitoring provided by FIFEMON tools
  - Now also helps users to understand why jobs aren’t running

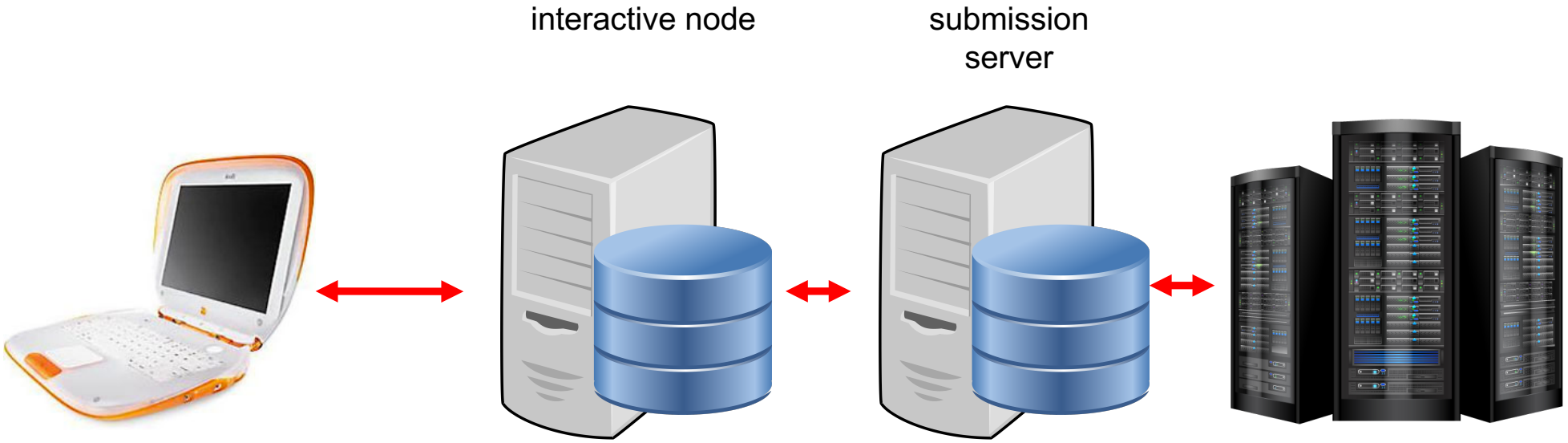


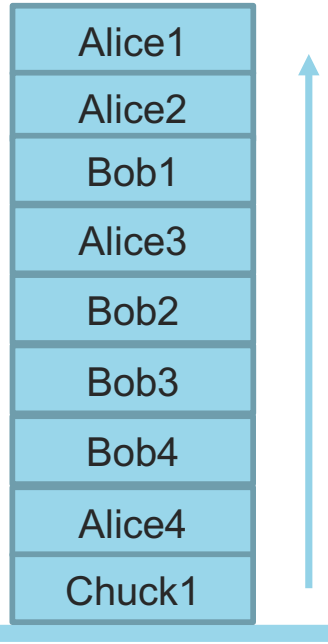
Lets start with the basics

- What happens when you submit jobs to the grid?
- You are authenticated and authorized to submit – discussed later
- Submission goes into batch queue (HTCondor) and waits in line
- You (or your script) hand to jobsub an executable (script or binary)
- Jobs are matched to a worker node – what does this mean?
- Server distributes your executable to the worker nodes
- Executable running on remote cluster and NOT as your user id – no home area, no NFS volume mounts, etc.

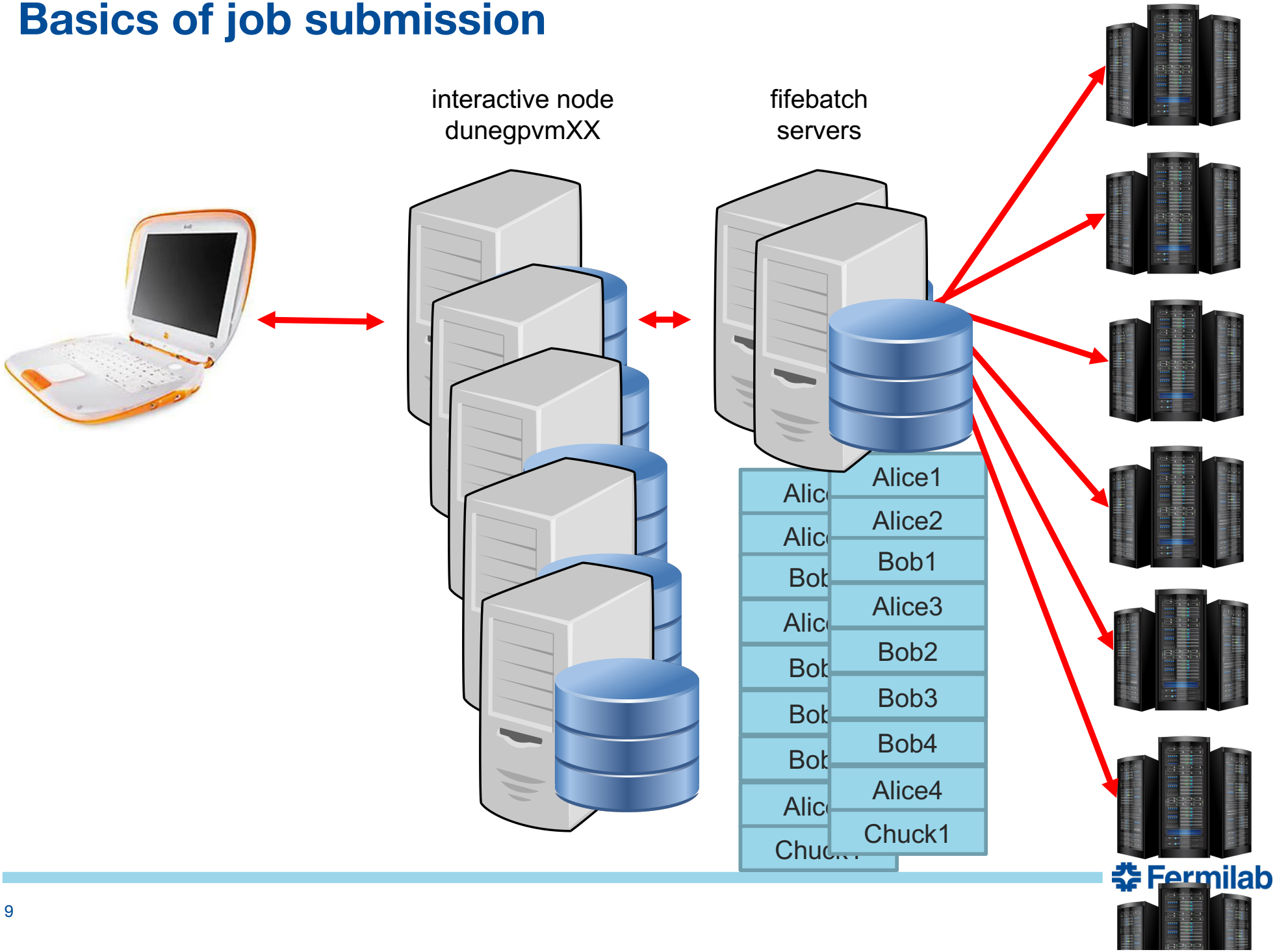


Basics of job submission





Basics of job submission



Fermilab

What do I need to know when I launch a script?

- What are the resources that my workflow needs?
  - –how many jobs to submit? how many concurrent?
  - –what allocation does my experiment have?
- What are the resources each job needs?
  - –remember that you're submitting to a worker node out on the Open Science Grid, it has limitations
  - –exceeding those limitations will cause your job to fail
- Where is the software that your workflow running coming from?
  - –is there code from your experiment?
  - –do you have custom compiled code?
- What data is required as an input to the workflow?
  - –how is the data going to get there?
  - –is the data readily available?

What resources does my workflow need?

- How many jobs to submit?
  - –if you have 1000 files to process, should you submit 1000 job sections? need to think about it!
  - –ideally jobs run for no less than 1 hour, but 2-8 hours should be target
  - –of course, need to balance input and output file size
- What is my experiment allocation? (some ground rules)
  - –shouldn't expect to submit 1,000,000 hours of processing in a week if you experiment only has an allocation of 100 slots (1M hours -> 6000 cores for a week)
  - –don't submit more than 5000 sections in one command
  - –don't submit more than 1 week of allocation at any one time
- Can the database handle NNN connections? Limit the number of concurrent jobs

What resources does each job need?

- What number of CPUs does the job need?
  - –are you running a multi-threaded job? don't do this accidentally!
- How much total memory does the job need? does it depend on the input? have I tested the input?
  - –profiling tools are available – valgrind, allinea, etc
  - –are has memory tracking modules as well
- How much scratch hard disk scratch space does the workflow use?
  - –staging input files from storage? writing output files before transferring back to storage?
- How much wall time for completion of each section? Note that wall time includes transferring input files, transferring output files, and connecting to remote resources (Databases, websites, etc.)

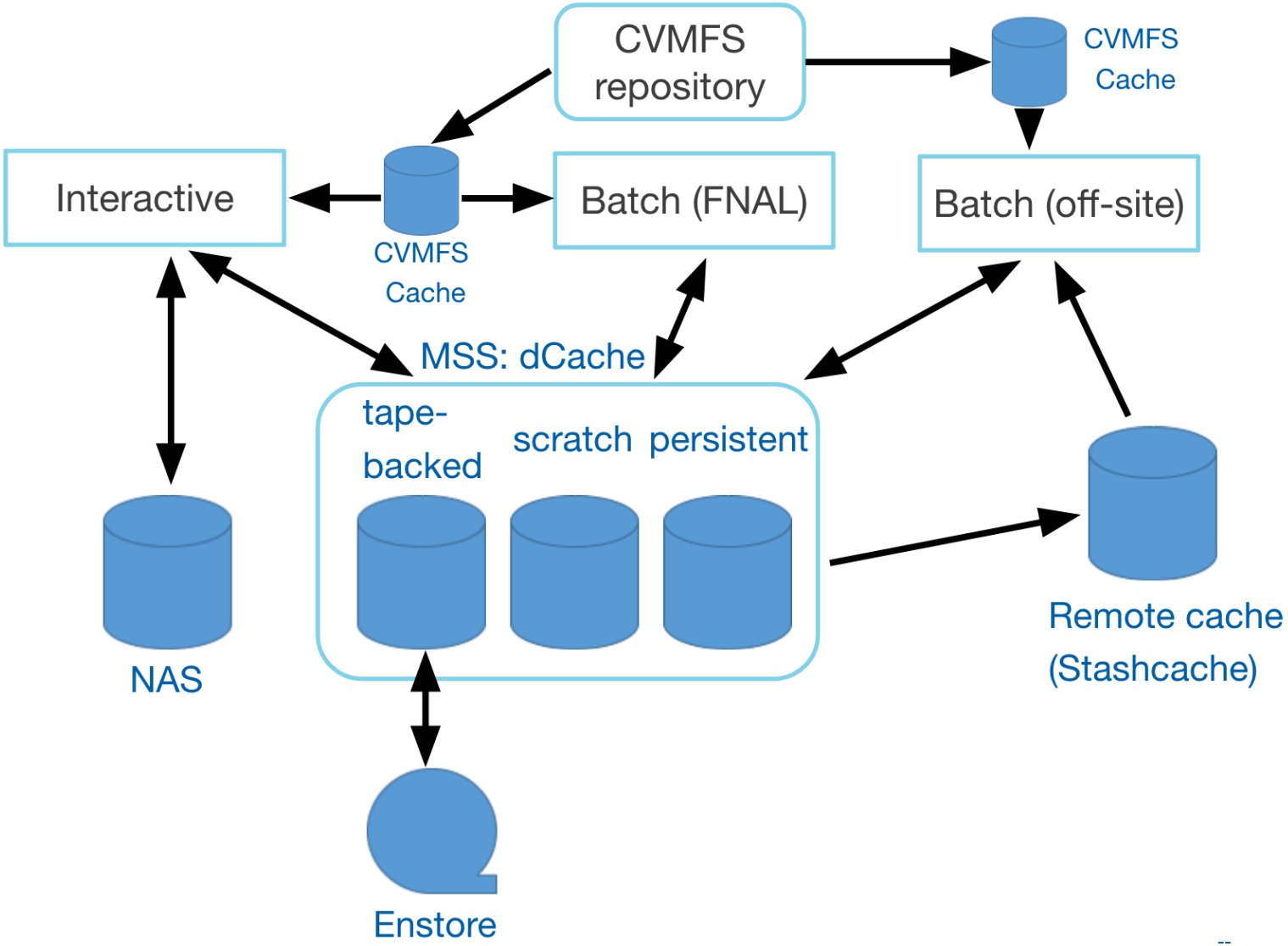
What software and libraries does my workflow need?

- The standard repository for accessing software and libraries is CVMFS (CERN Virtual Machine File System)
  - –mounted on all worker nodes
  - –mounted on all interactive nodes
  - –can be mounted on your laptop
  - –used for centralized distribution of packaged releases of experiment software – not your personal dev area
  - –not to be used for distribution of data or reference files
- locally built development code should be placed in a tarball on dCache, transferred to the worker nodes through dropbox, and then unwound into the scratch area
- can test that your software is grid friendly using gpgtest.fnal.gov as a test node.

What data is required as an input for the workflow?

- data files should be transferred in from dCache
  - –BlueArc transfers are going away (and they are extremely inefficient)
  - –can utilize multiple transfer protocols to get data
  - –XRootD, gridftp, etc – large number of doors to dCache
- but need to think about volume of data being transferred
  - –how large is the dataset? number of files and volume
  - –tune the number of job sections in a cluster to match the timing of the workflow and the size of output file
  - –1 – 10 GB is a good target for output
- Is the data ready to be processed?

Where is the data actually located?



Make sure to prestage your data!

- on an interactive node

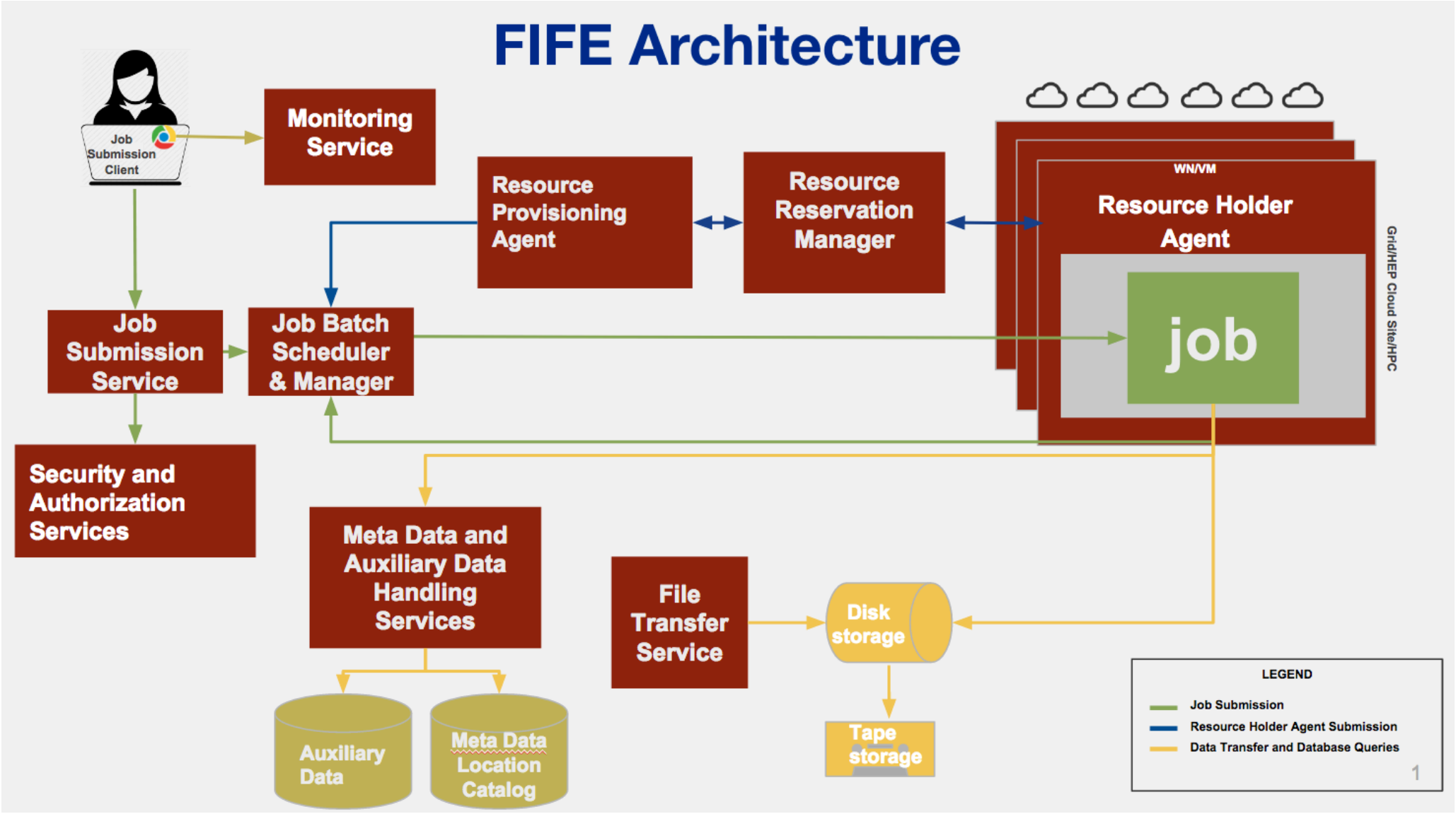
```
> source /cvmfs/fermilab.opensciencegrid.org/products/common/etc/setup
> setup sam_web_client
> setup kx509
> kx509
> samweb prestage-dataset –e <your_experiment_here>
\--defname=<your_dataset_here>
```

- When that command completes, your data should be located on disk
- Will greatly improve your efficiency knowing that the data is located ONLINE and not waiting for tape access
- cat /pnfs/<exp>/data/<full_path>/".(get)(<file_name_here>)(locality)"
  - –ONLINE – staged to disk
  - –NEARLINE – copy stored on tape

What to know before you submit you jobs?

- Think about what resources your workflow will occupy while processing?
- is it equitable for importance of submission?
- will it overload any of the services?
- will the efficiency be as high as possible?
- have you matched the resource needs with the resource request?
- have you made sure that all of the software needed is accessed via Grid-friendly tools?
- are you being a good steward of the resources being provided to you?

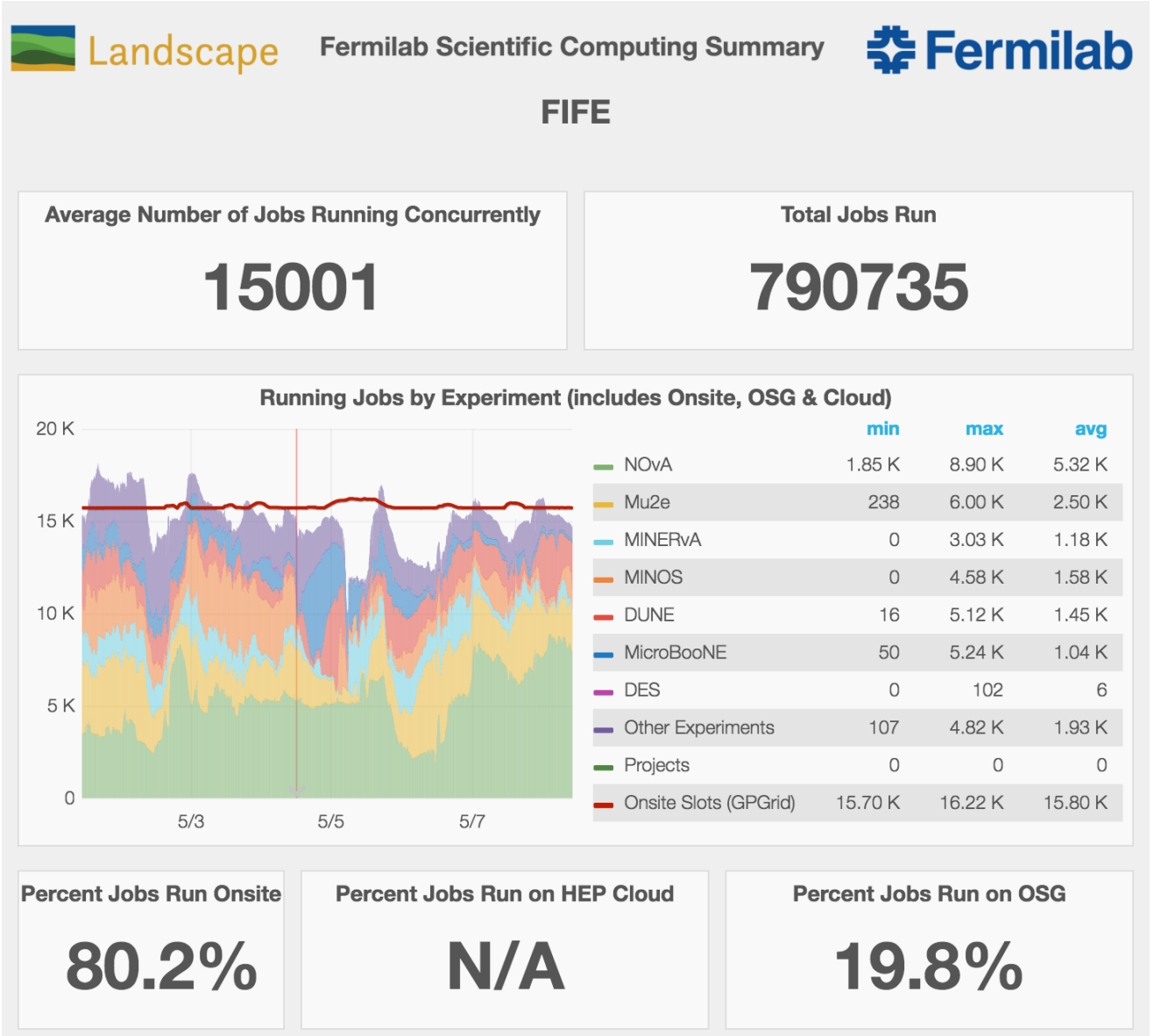
More complicated picture



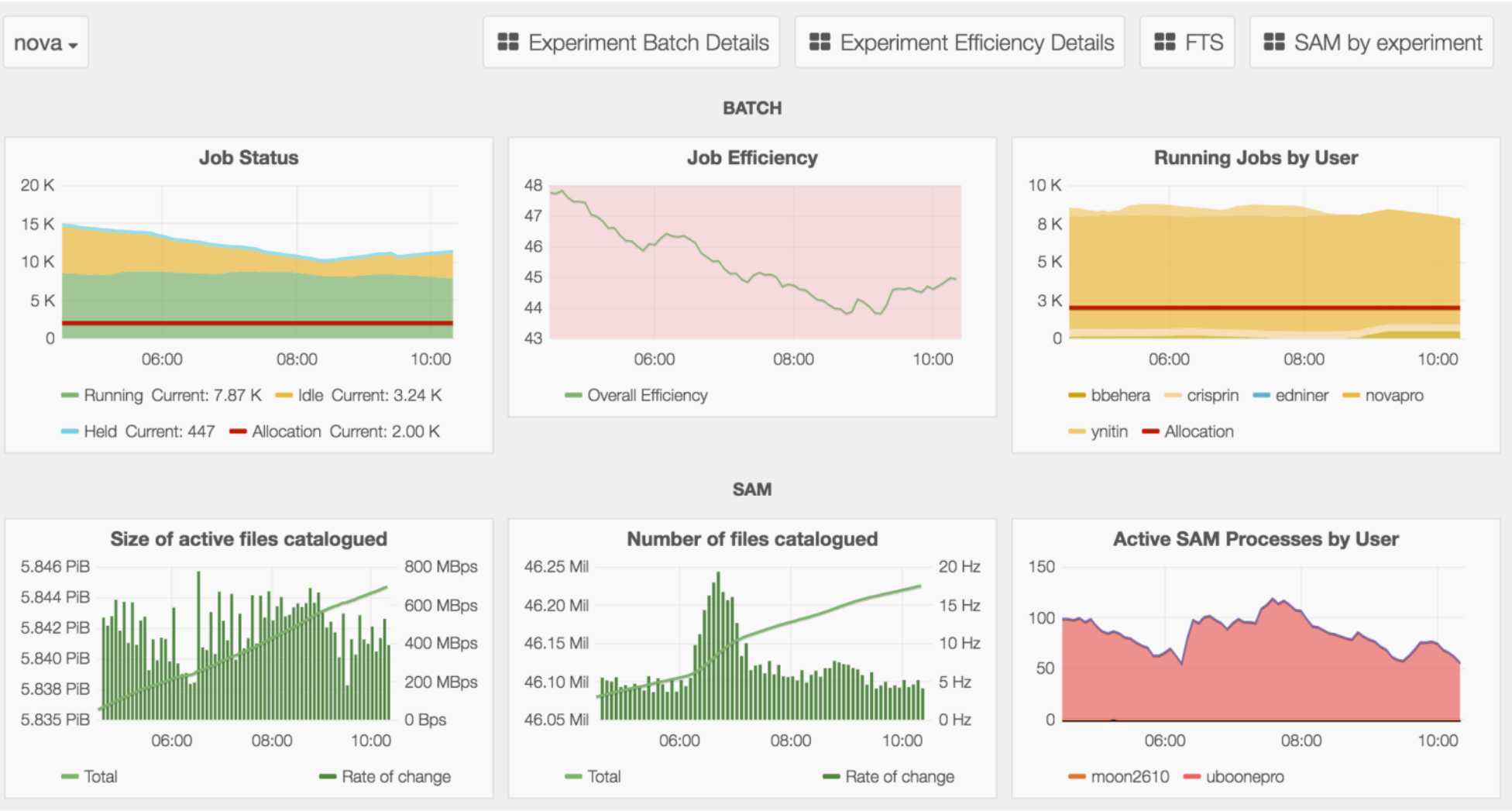
Fermilab

FIFE Monitoring of resource utilization

- Extremely important to understand performance of system
- Critical for responding to downtimes and identifying inefficiencies
- Focused on improving the real time monitoring of distributed jobs, services, and user experience
- fifemon.fnal.gov

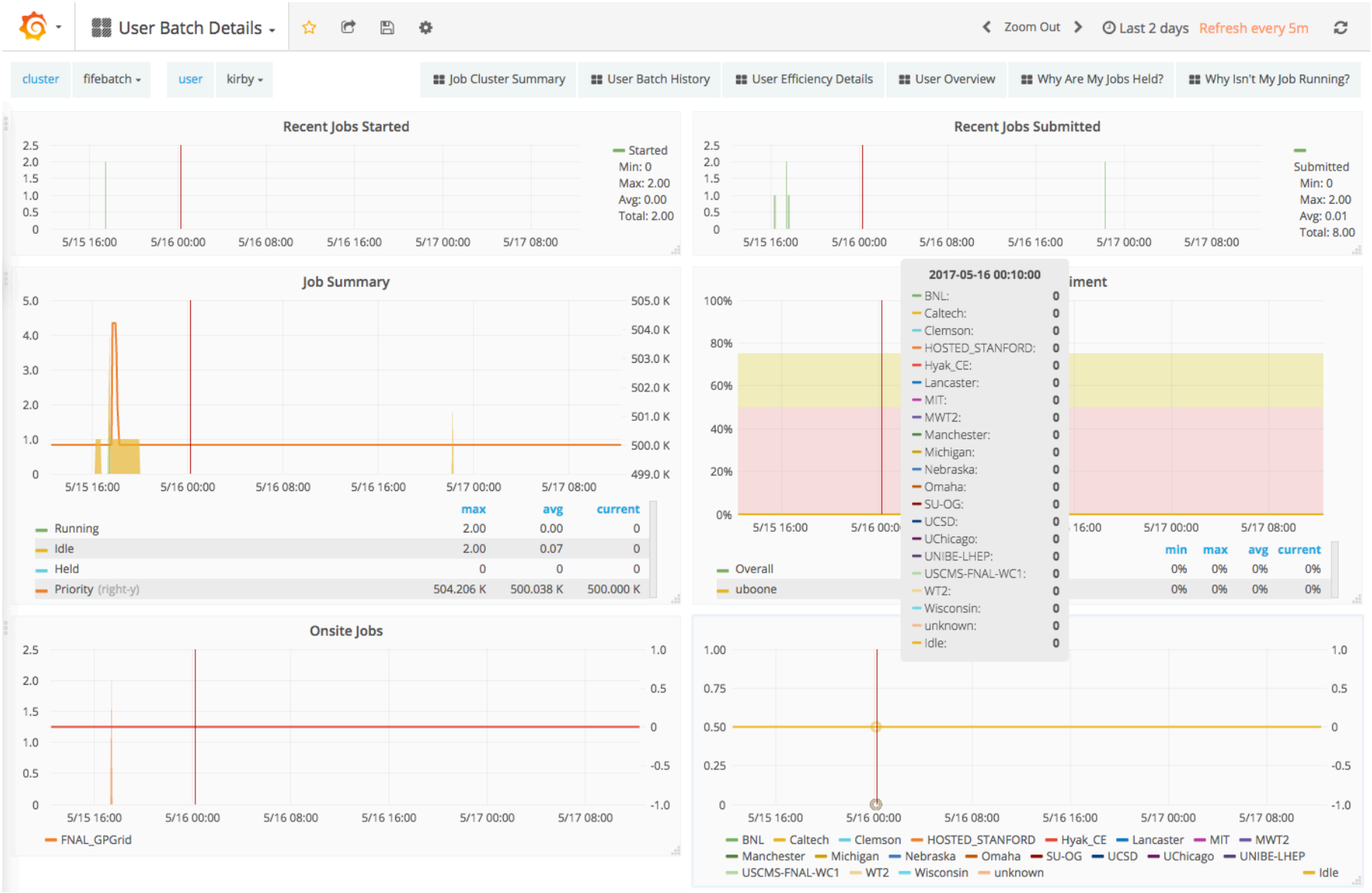


Detailed profiling of experiment operations



Allows identification for inefficiencies, potential slow downs, or blocking conditions in workflows

User Batch Details Dashboard



Fermilab

Backup