# Batch Job Submission Tutorial

Ken Herner
FIFE Workshop
22 June 2017

*22 June 1942: Congress adopts the Pledge of Allegiance*

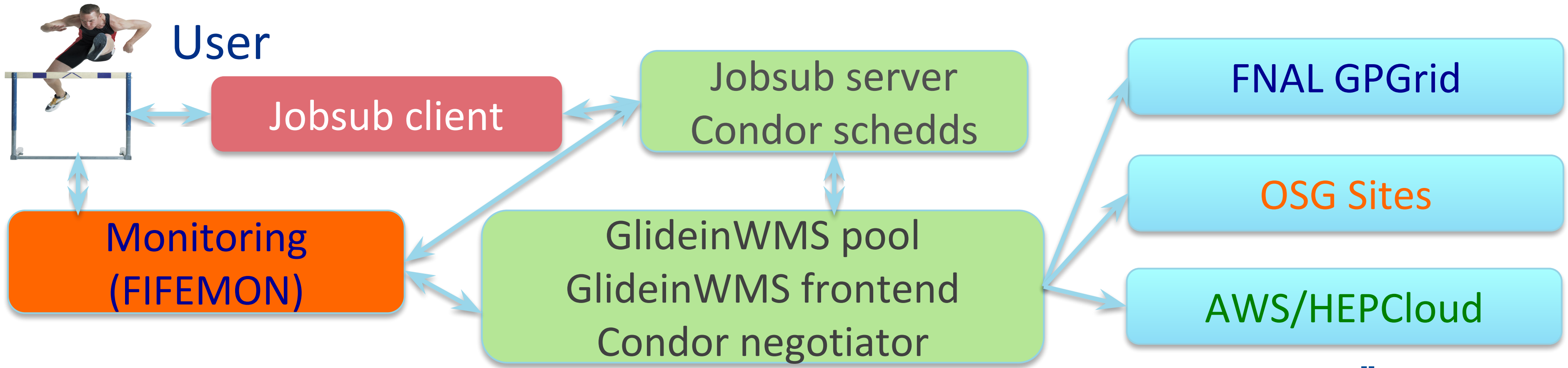
FABRIC FOR FRONTIER EXPERIMENTS

## Outline

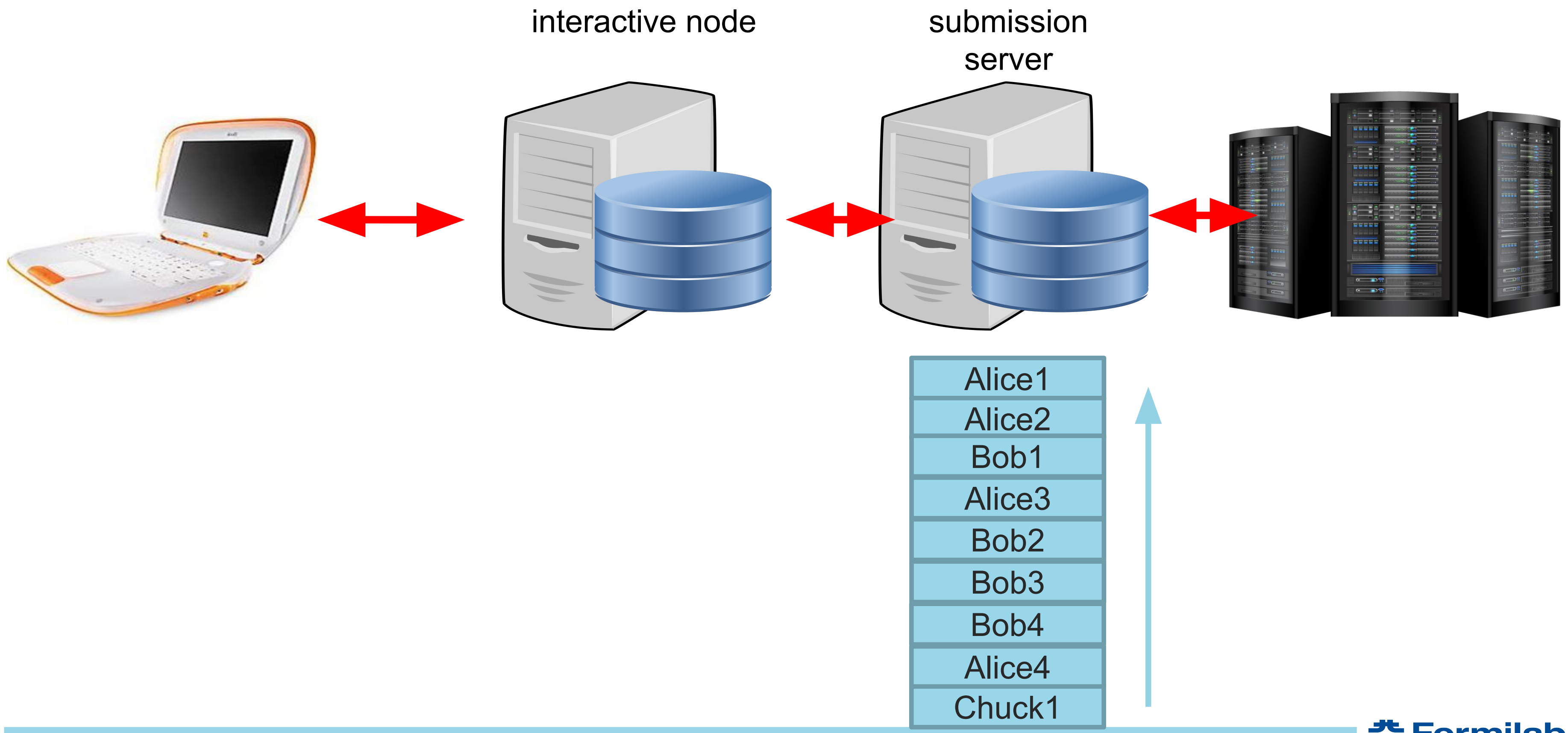
- Examples of job submission
- Monitoring
- GPU job submission

## Job Submission and management architecture

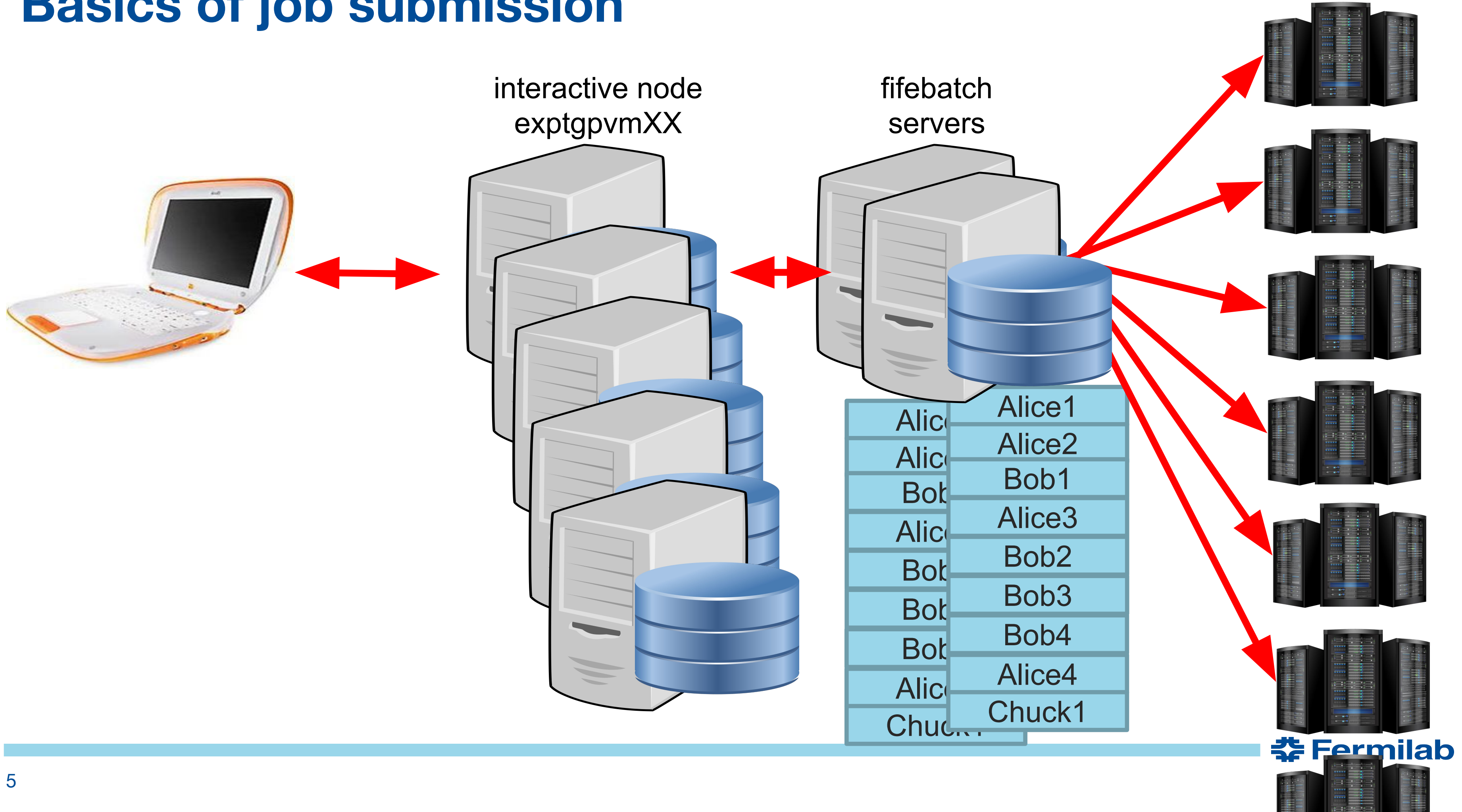
- Common infrastructure is the fifebatch system: one GlideInWMS pool, 2 schedds, frontend, collectors, etc.
- Users interface with system via “jobsub”: middleware that provides a *common tool across all experiments*; shields user from intricacies of Condor
- Common monitoring provided by FIFEMON tools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



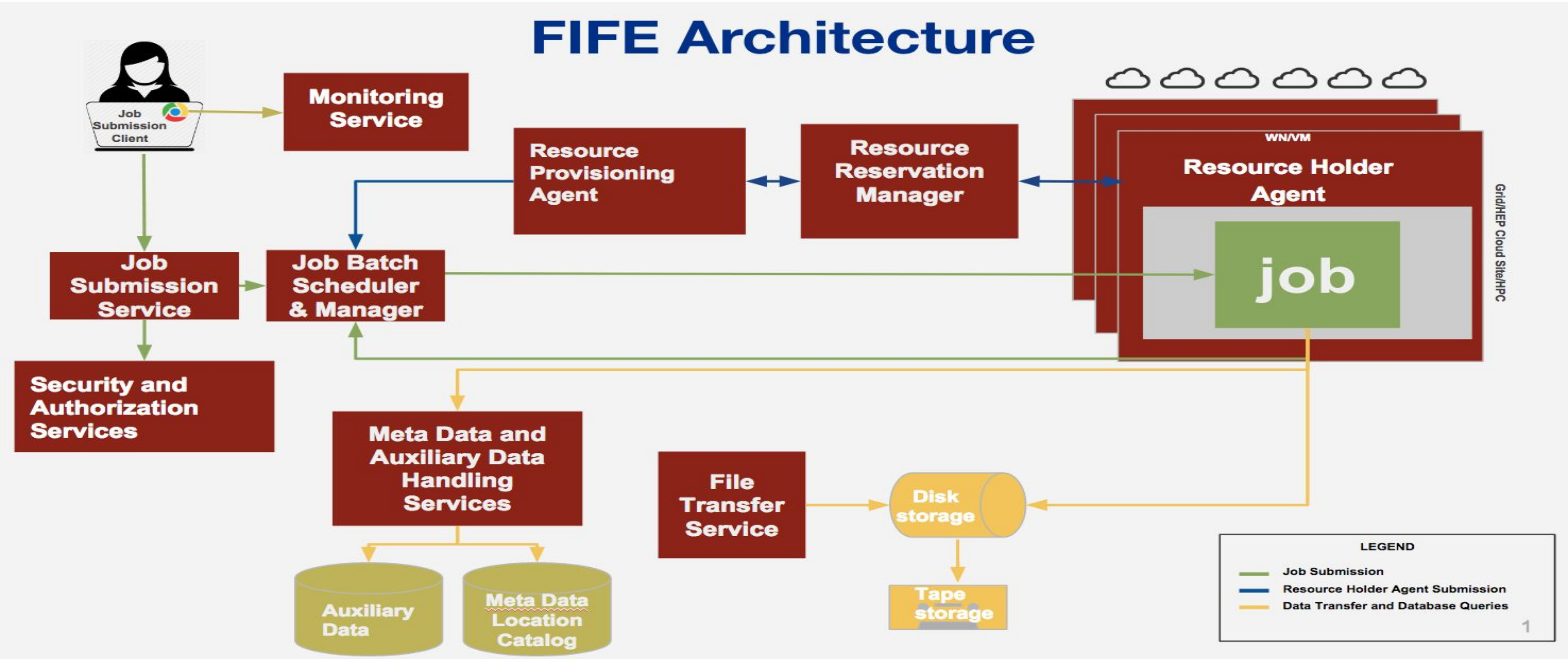
# Basics of job submission



# Basics of job submission



# More complicated picture


FIFE Architecture
Job Submission Client
Monitoring Service
Resource Provisioning Agent
Resource Reservation Manager
WN/VM
Resource Holder Agent
job
Grid/HEP Cloud Site/HPC
Job Submission Service
Job Batch Scheduler & Manager
Security and Authorization Services
Meta Data and Auxiliary Data Handling Services
File Transfer Service
Disk storage
Tape storage
Auxiliary Data
Meta Data Location Catalog
LEGEND
Job Submission
Resource Holder Agent Submission
Data Transfer and Database Queries
1

## Example script and submission command

- ➢ kinit -f
- ➢ ssh -K (your experiment gpvm)
- ➢ mkdir /pnfs/<your experiment>/scratch/users/${USER}
- ➢ ls /dune/app/users/kirby/dune_may2017_tutorial/*.sh
- ➢

Let’s take a look at basic_grid_env_test.sh

Note that this shape ➢ begins commands you can cut and paste

## Look inside the basic grid env test script

```
<dunegpvm01.fnal.gov> more basic_grid_env_test.sh
#!/bin/bash
printenv
set -x #start bash debugging at this point
echo Start  `date`
echo Site:${GLIDEIN_ResourceName}
echo "the worker node is " `hostname` "OS: "  `uname -a`
echo "the user id is " `whoami`
echo "the output of id is " `id`
set +x #stop bash debugging at this point
cd $_CONDOR_SCRATCH_DIR
echo "pwd is " `pwd`
Sleep $[ ( $RANDOM % 10 )  + 1 ]m #sleep for random integer of minutes between 1-10 inclusive
echo Stop `date`
exit 0;
```

## How do you submit that script to run on the OSG?

- ➢ source /cvmfs/fermilab.opensciencegrid.org/products/common/etc/setup

This establishes a UPS product working area (more about this later)

- ➢ setup jobsub_client #with no options, get version declared "current"
- ➢ jobsub_submit -N 2 -G **your_experiment** --expected-lifetime=1h \
  --memory=500MB --disk=2GB \
  --resource-provides=usage_model=DEDICATED,OPPORTUNISTIC,OFFSITE \
  file:///nashome/k/kherner/basic_grid_env_test.sh

https://cdcvs.fnal.gov/redmine/projects/jobsub/wiki/Jobsub_submit

N is the number of jobs in a cluster

G is the experiment group

expected-lifetime is the upper limit it of single job run time in the cluster

memory is the RAM footprint of a single job in the cluster

disk is the scratch space need for a single job in the cluster

## Things to note upon job submission

When you run the command you will get something like:

```
/fife/local/scratch/uploads/dune/kirby/2017-05-15_161406.077316_7456

/fife/local/scratch/uploads/dune/kirby/2017-05-15_161406.077316_7456/basic_grid_env_test.sh_201705
15_161407_2384341_0_1_.cmd

submitting....

Submitting job(s).

2 job(s) submitted to cluster 17067704.

JobsubJobId of first job: 17067704.0@fifebatch1.fnal.gov

Use job id 17067704.0@fifebatch1.fnal.gov to retrieve output
```

You can use this job ID string
(the entire thing, not just the number)
For manipulating the job

# How do I check up on my submitted jobs?

- jobsub_q --user=${USER}

```
JOBSUBJOBID                     OWNER     SUBMITTED     RUN_TIME   ST PRI SIZE CMD
17067704.0@fifebatch1.fnal.gov    kirby     05/15 16:14  0+00:00:00 I  0   0.0
basic_grid_env_test.sh_20170515_161407_2384341_0_1_wrap.sh
1 jobs; 0 completed, 0 removed, 1 idle, 0 running, 0 held, 0 suspended
```

- user specifies the uid you want the status of on jobsub server
- --jobid can be used to get the status of a single job
- job statuses can be the following:
  - R is running
  - I is idle (a.k.a. waiting for a slot)
  - H is held (job exceeded a resource allocation)
- -G to get the group
- --hold: for all the held jobs
- --run: for all the running jobs
- --better-analyze do condor_q -better-analyze on job (must use with --jobid) to list matching
- use better-analyze with caution! can overload the server by repeatedly trying

https://cdcvs.fnal.gov/redmine/projects/jobsub/wiki/Jobsub_q

## Additional commands to consider

Full documentation of the jobsub client here

https://cdcvs.fnal.gov/redmine/projects/jobsub/wiki/Using_the_Client

- jobsub_history – get history of submissions
- jobsub_rm – remove jobs/clusters from jobsub server
- jobsub_hold – set jobs/clusters to held status
- jobsub_release – release held jobs/clusters
- jobsub_fetchlog – get the condor logs from the server

## Fetching your logs from jobs submitted

- Need to remember the jobid for the cluster you submitted
- make sure you setup the jobsub_client UPS product
- setup jobsub_client
- mkdir basic_log; cd basic_log
- jobsub_fetchlog -G your_experiment \
  --jobid=17067704.0@fifebatch1.fnal.gov \
  --unzipdir=/dir/of/your/choice

#replace with the jobid that we highlighted earlier

Downloads the logs from to 17067704.0@fifebatch1.fnal.gov and unzips them into /dir/of/your/choice

## Inside the jobsub log tarball

➢ ls –alrt

```
total 52
-rwxr-xr-x 1 kirby dune  450 May 15 16:14 basic_grid_env_test.sh
-rwxr-xr-x 1 kirby dune 6473 May 15 16:14 basic_grid_env_test.sh_20170515_161407_2384341_0_1_wrap.sh
-rw-r--r-- 1 kirby dune 2254 May 15 16:14 basic_grid_env_test.sh_20170515_161407_2384341_0_1_.cmd
-rw-r--r-- 1 kirby dune    0 May 15 16:14 .empty_file
-rw-r--r-- 1 kirby dune 6903 May 15 16:43
basic_grid_env_test.sh_20170515_161407_2384341_0_1_cluster.17067704.0.out
-rw-r--r-- 1 kirby dune  869 May 15 16:43
basic_grid_env_test.sh_20170515_161407_2384341_0_1_cluster.17067704.0.err
-rw-r--r-- 1 kirby dune 4983 May 15 16:43 basic_grid_env_test.sh_20170515_161407_2384341_0_1_.log
```

➢ These files are in order:

- shell script sent to the jobsub server
- wrapper script created by jobsub server to set environment variables
- condor command file sent to condor to put job in queue
- an empty file
- stdout of the bash shell run on the worker node
- stderr of the bash shell run on the worker node
- condor log for the job

## More complicated script to run on the grid

- This script prints the environment
- changes to the scratch area
- based upon the $EXPERIMENT variable sets the $SCRATCH directory in dCache (a.k.a. pnfs space)
- prints information about the grid proxy
- sets up the Fermilab common UPS product area
- sets up the IFDH client UPS product
- echoes the environment to a user created log and error files
- tries to list contents of the not-mounted /dune/data volume
- determines the GRID_USER from the proxy if not set
- sleeps for random time
- copies log files to $SCRATCH directory

## Submitting a more complicated job

Now, try submitting the following:

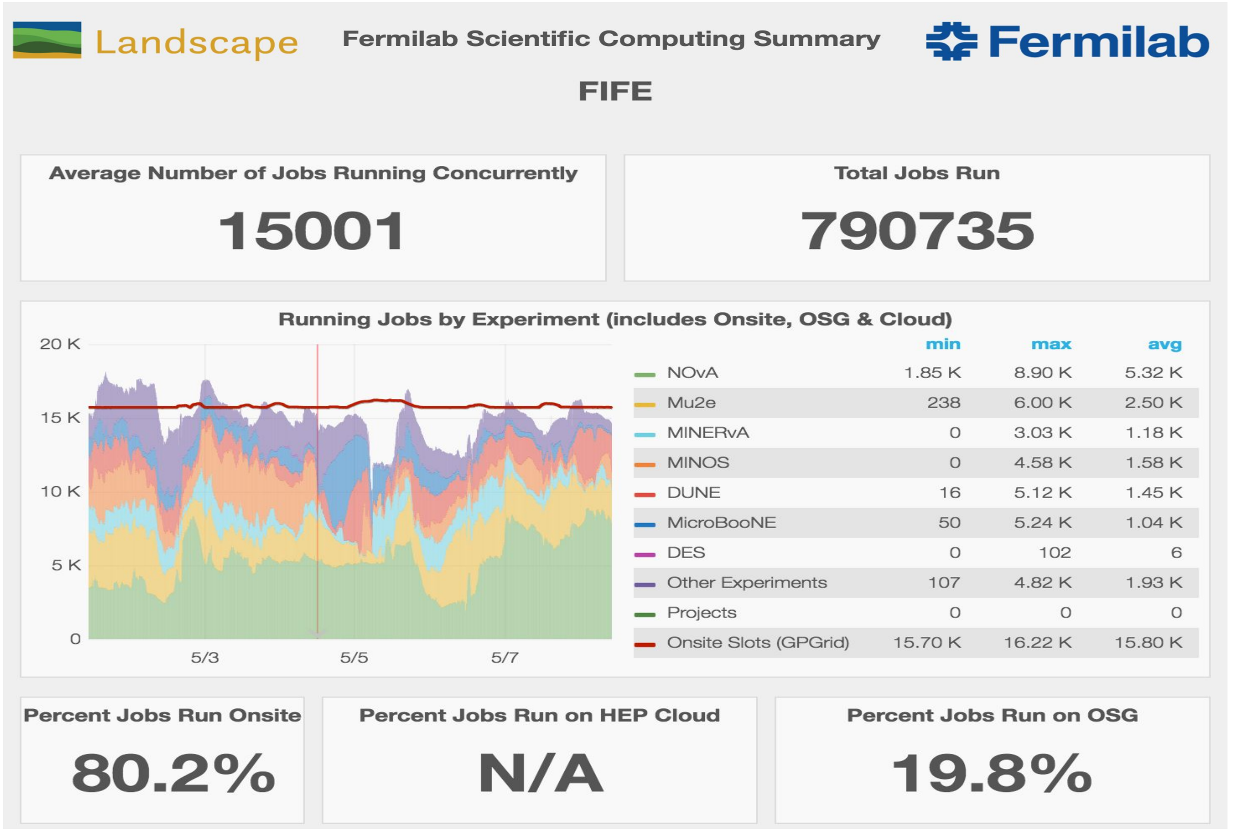
➢ 
```
jobsub_submit -N 2 -G your_experiment --expected-lifetime=1h \
--memory=500MB --disk=2GB \
--resource-provides=usage_model=DEDICATED,OPPORTUNISTIC,OFFSITE \
file:///nashome/k/kherner/submission_test.sh
```

Then you can fetch the log, check FIFEMON, etc., as you did with the previous test

If successful, you should see an output file with the cluster and process number in your /pnfs/experiment/scratch/users/username directory

# 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



Landscape   Fermilab Scientific Computing Summary   Fermilab

FIFE

Average Number of Jobs Running Concurrently: 15001

Total Jobs Run: 790735

Running Jobs by Experiment (includes Onsite, OSG & Cloud)

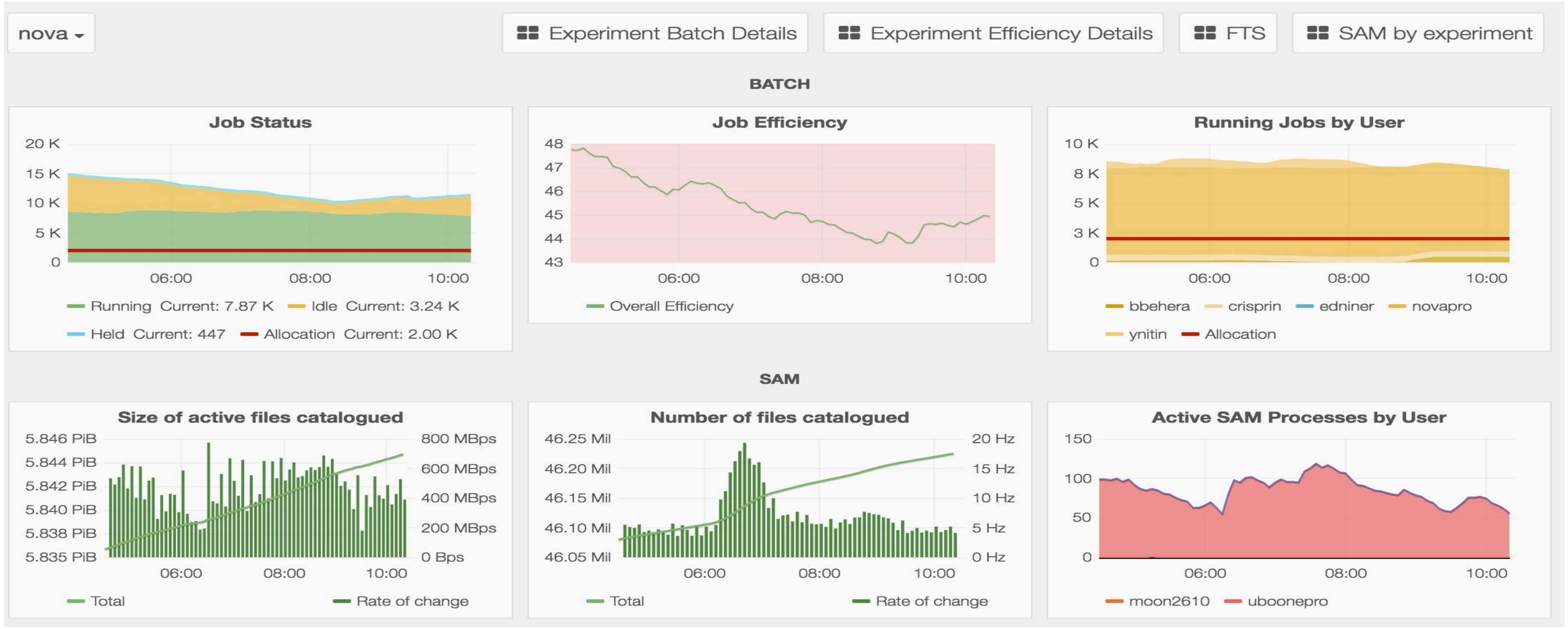
| | min | max | avg |
|---|---|---|---|
| NOvA | 1.85 K | 8.90 K | 5.32 K |
| Mu2e | 238 | 6.00 K | 2.50 K |
| MINERvA | 0 | 3.03 K | 1.18 K |
| MINOS | 0 | 4.58 K | 1.58 K |
| DUNE | 16 | 5.12 K | 1.45 K |
| MicroBooNE | 50 | 5.24 K | 1.04 K |
| DES | 0 | 102 | 6 |
| Other Experiments | 107 | 4.82 K | 1.93 K |
| Projects | 0 | 0 | 0 |
| Onsite Slots (GPGrid) | 15.70 K | 16.22 K | 15.80 K |

Percent Jobs Run Onsite: 80.2%

Percent Jobs Run on HEP Cloud: N/A

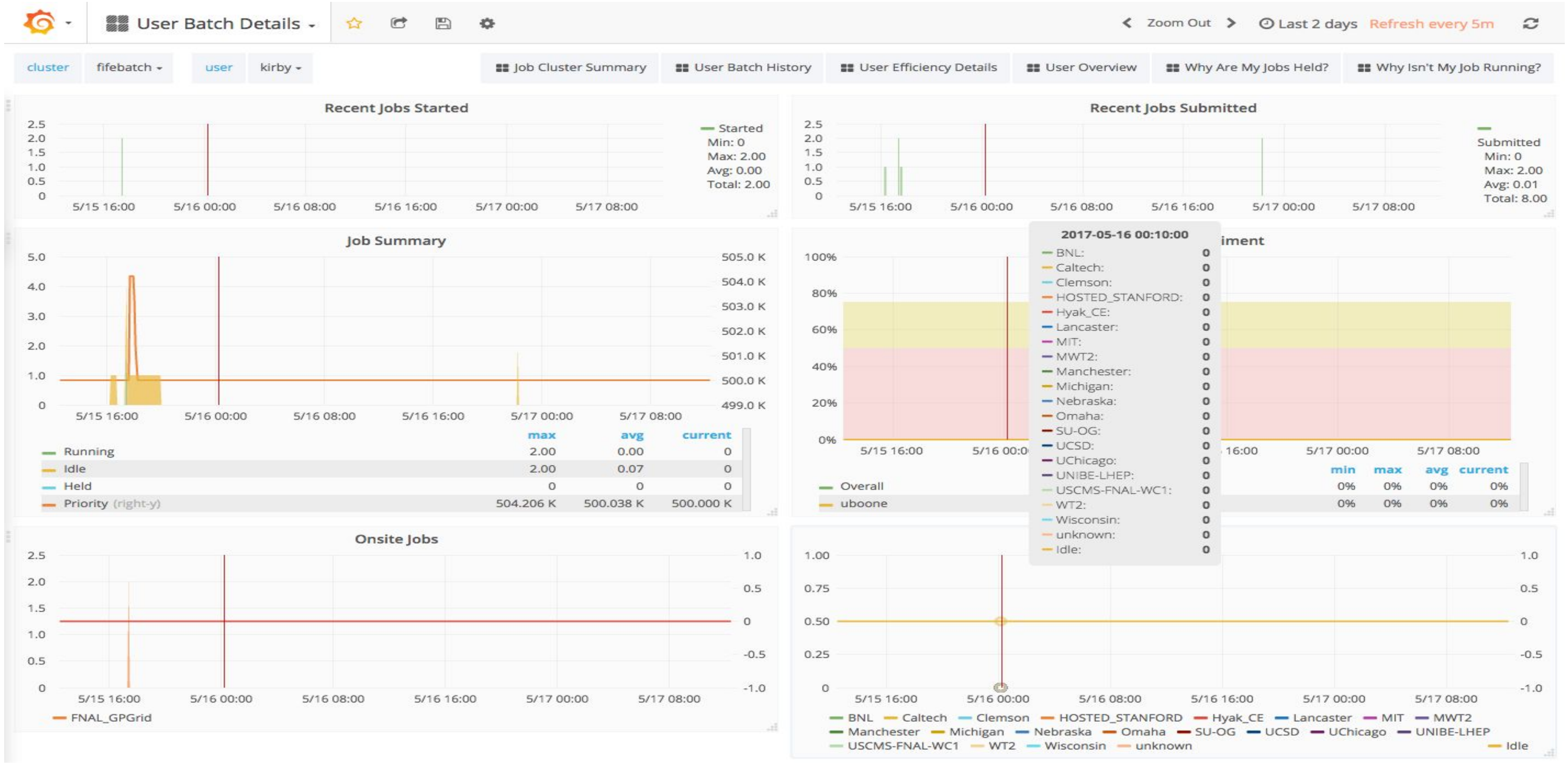
Percent Jobs Run on OSG: 19.8%

# Detailed profiling of experiment operations



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

# User Batch Details Dashboard



cluster: fifebatch; user: kirby

**Recent Jobs Started**

Started
Min: 0
Max: 2.00
Avg: 0.00
Total: 2.00

**Recent Jobs Submitted**

Submitted
Min: 0
Max: 2.00
Avg: 0.01
Total: 8.00

**Job Summary**

| | max | avg | current |
|---|---|---|---|
| Running | 2.00 | 0.00 | 0 |
| Idle | 2.00 | 0.07 | 0 |
| Held | 0 | 0 | 0 |
| Priority (right-y) | 504.206 K | 500.038 K | 500.000 K |

| | min | max | avg | current |
|---|---|---|---|---|
| Overall | 0% | 0% | 0% | 0% |
| uboone | 0% | 0% | 0% | 0% |

**2017-05-16 00:10:00**

| | |
|---|---|
| BNL: | 0 |
| Caltech: | 0 |
| Clemson: | 0 |
| HOSTED_STANFORD: | 0 |
| Hyak_CE: | 0 |
| Lancaster: | 0 |
| MIT: | 0 |
| MWT2: | 0 |
| Manchester: | 0 |
| Michigan: | 0 |
| Nebraska: | 0 |
| Omaha: | 0 |
| SU-OG: | 0 |
| UCSD: | 0 |
| UChicago: | 0 |
| UNIBE-LHEP: | 0 |
| USCMS-FNAL-WC1: | 0 |
| WT2: | 0 |
| Wisconsin: | 0 |
| unknown: | 0 |
| Idle: | 0 |

**Onsite Jobs**

FNAL_GPGrid

BNL, Caltech, Clemson, HOSTED_STANFORD, Hyak_CE, Lancaster, MIT, MWT2, Manchester, Michigan, Nebraska, Omaha, SU-OG, UCSD, UChicago, UNIBE-LHEP, USCMS-FNAL-WC1, WT2, Wisconsin, unknown, Idle

## Things to know when you’re submitting jobs

- What number of CPUs does the job need?
- How much total memory does the job need?
    - Does it depend on the input(s)?
    - Have I tested the input?
- How much scratch hard disk scratch space does the job need to use?
    - Am I staging input files from storage?
    - Am I 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.)

# A word on Production jobs

If you're submitting with a role other than the default Analysis role (typically the Production role), you'll need to add the --role option, e.g. **jobsub_submit --role=Production …** You must be authorized for this role of course. Then the job will run as the experimentpro user, and be able to access everything owned by that user (e.g. in /pnfs space)

**Note:** when doing this, all subsequent commands on such jobs (jobsub_rm, jobsub_fetchlog, jobsub_release, etc.) must also use the same --role option

## A bit on DAGs

Directed Acyclic Graphs (DAGs) are a way to run jobs that depend on other jobs (e.g. run geant4 on the output of the event generator step)

User can define structure and dependencies, but there is only a single submission (no babysitting required!) Later stage jobs start automatically after previous stage finished.

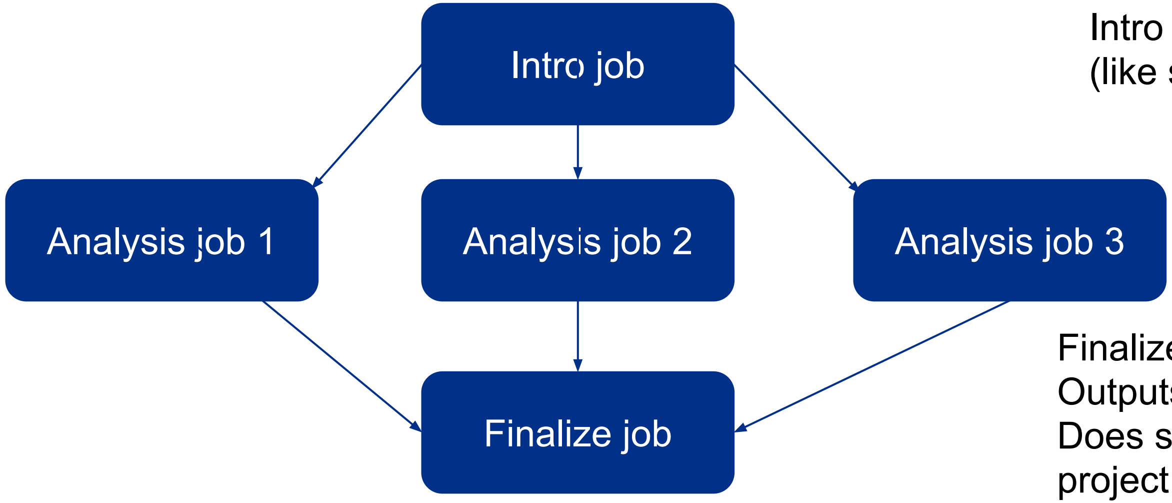
**Note: if parent job fails, dependent jobs will not run**

Possible to create DAGs for jobsub with simple xml schema, and then submit with **jobsub_submit_dag**

## XML schema

There are serial jobs, parallel jobs, and each section runs as a different stage (note ALL jobs in stage N must finish before stage N+1 starts)

Consider this simple workflow:



Intro job does prep work (like starting a SAM project)

Finalize job looks at Outputs of analysis jobs and Does something and/or ends project

# Toy workflow XML

```
$ cat mytoy.xml
<serial>
jobsub -n --memory=500MB --disk=1GB \ --expected-lifetime=1h \
   --resource-provides=usage_model=DEDICATED,OPPORTUNISTIC,OFFSITE file://init_script.sh ARGS0
</serial>
<parallel>
jobsub -n --memory=2000MB --disk=1GB \ --expected-lifetime=3h \
   --resource-provides=usage_model=DEDICATED,OPPORTUNISTIC,OFFSITE file://analysis_script.sh ARGS
jobsub -n --memory=2000MB --disk=1GB \ --expected-lifetime=3h \
   --resource-provides=usage_model=DEDICATED,OPPORTUNISTIC,OFFSITE file://analysis_script.sh ARGS
jobsub -n --memory=2000MB --disk=1GB \ --expected-lifetime=3h \
   --resource-provides=usage_model=DEDICATED,OPPORTUNISTIC,OFFSITE file://analysis_script.sh ARGS
</parallel>
<serial>
jobsub -n --memory=1000MB --disk=5GB \ --expected-lifetime=2h \
   --resource-provides=usage_model=DEDICATED,OPPORTUNISTIC,OFFSITE file://finalize_script.sh ARGS2
</serial>
```

# Toy workflow XML

```
$ cat mytoy.xml
<serial>
jobsu
  --r
</ser
<para
jobsu
  --r                                                                              S
jobsub -n --memory=2000MB --disk=1GB \ --expected-lifetime=3h \
  --resource-provides=usage_model=DEDICATED,OPPORTUNISTIC,OFFSITE file://analysis_script.sh ARGS
jobsub -n --memory=2000MB --disk=1GB \ --expected-lifetime=3h \
  --resource-provides=usage_model=DEDICATED,OPPORTUNISTIC,OFFSITE file://analysis_script.sh ARGS
</parallel>
<serial>
jobsub -n --memory=1000MB --disk=5GB \ --expected-lifetime=2h \
  --resource-provides=usage_model=DEDICATED,OPPORTUNISTIC,OFFSITE file://finalize_script.sh ARGS2
</serial>
```

Notes:
You put jobsub, **not jobsub_submit**, inside the xml. **You also need a -n** after jobsub.
Yo do not specify group and role here; that is part of jobsub_submit_dag
The arguments to each job can be different. You can also switch resource requirements and arguments around from job to job

## Submitting a toy DAG; fetching logs

In our example you would do

**jobsub_submit_dag -G myexpt file://mytoy.xml**

If you wanted to run as the production user, add --role=Production to jobsub_submit_dag (NOT inside the xml)

If any of the analyze jobs were to fail, the finalize job would not run. But: no need to monitor and submit each sage separately

Other notes:

The jobs do NOT share a cluster ID; each gets its own. There is a variable called JOBSUBPARENTJOBID (based on the control job) that is the same in all jobs in the DAG

If you do jobsub_fetchlog --jobid=<job ID of submission> you will get the logs for ALL jobs in the DAG. If you want them only for a specific job, do **jobsub_fetchlog --jobid=<job ID of particular job> --partial** (the --partial option does the trick)

## GPU Access

Lots of (justified) excitement about GPUs. Some of you may be familiar with the Wilson Cluster at Fermilab; this requires a separate account

Other GPU clusters available via OSG and reachable via jobsub, with some extra options

Today, we will likely end up at Nebraska. Syracuse and UCSD are coming very soon. Unfortunately it tends to be slow going since this is somewhat new for all involved

I don't have a specific task set up for GPUs, but will instead offer general advice

First, get a test job running...

## GPU Test job

- In general you just have to add **--lines='+RequestGPUs=1'** to your jobsub_submit command
  - The FNAL frontend will see this and steer to you an appropriate site
  - Can request >1 but no promises made about availability or fast start time. Could also add options for specific site if needed (not recommended)
- So try the following:
- **$ jobsub_submit -G des -M --memory=1000MB --disk=1GB  --expected-lifetime=1h -N 8 \ --resource-provides=usage_model=OFFSITE --lines='+RequestGPUs=1' \ --jobsub-server=https://fifebatch-dev.fnal.gov file:///home/s1/kherner/basicscript_GPU.sh**

```
$ jobsub_submit -G des -M --memory=1000MB --disk=1GB  --expected-lifetime=1h -N 8 \
--resource-provides=usage_model=OFFSITE --lines='+RequestGPUs=1' \
--jobsub-server=https://fifebatch-dev.fnal.gov file:///home/s1/kherner/basicscript_GPU.sh
```

**Notes:**

**--jobsub-server=https://fifebatch-dev.fnal.gov** necessary today because this is still technically experimental. Will not be needed once this is in production. **You should NOT send more than a few test jobs here. You also need to supply this option if doing jobsub_q, rm, fetchlog, etc. on these jobs.**

-N 8 needed on the development server. N can be anything you want in production.

## Summary

This is a lot of information in a short time, but hopefully it's also useful as reference material

Please take time to study the scripts in more detail to understand exactly what they're doing

When in doubt, please ask us. Knowing is much more than half the battle. Useful documentation:

https://cdcvs.fnal.gov/redmine/projects/jobsub/wiki/Using_the_Client

https://cdcvs.fnal.gov/redmine/projects/jobsub/wiki/Jobsub_submit_dag

https://cdcvs.fnal.gov/redmine/projects/jobsub/wiki/Jobsub_enviornment_variables

**For bugs, issues, feature requests:**

https://cdcvs.fnal.gov/redmine/projects/jobsub/issues