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

Introduction to FIFE
Grid submission tutorial

Mike Kirby

DUNE Software Tutorials

Aug 14, 2017

FABRIC FOR FRONTIER EXPERIMENTS

Introduction to FIFE

FABRIC FOR FRONTIER EXPERIMENTS

- The **F**abr**I**c for **F**rontier **E**xperiments aims to:
- Lead the development of the computing model for non-LHC experiments
- Provide a robust, common, modular set of tools for experiments, including
  - Job submission, monitoring, and management software
  - Data management and transfer tools
  - Database and conditions monitoring
  - Collaboration tools such as electronic logbooks, shift schedulers
- Work closely with experiment contacts during all phases of development and testing
- https://web.fnal.gov/project/FIFE/SitePages/Home.aspx

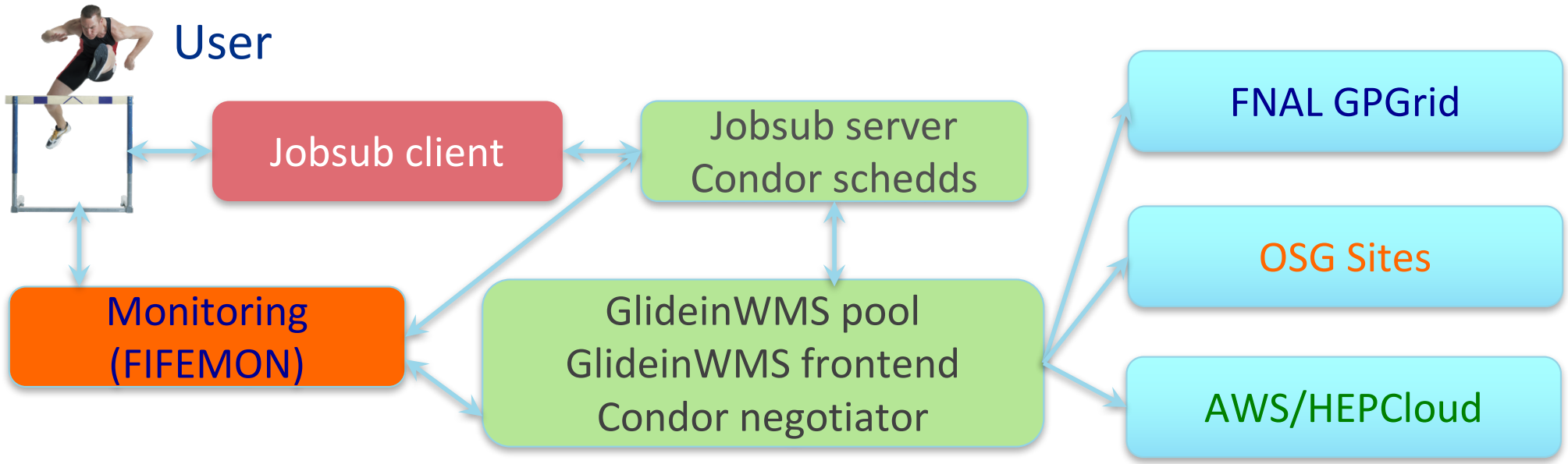
Centralized Services from FIFE

FABRIC FOR FRONTIER EXPERIMENTS

- Submission to distributed computing – JobSub, GlideinWMS
- 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 (IFDHC)
- 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

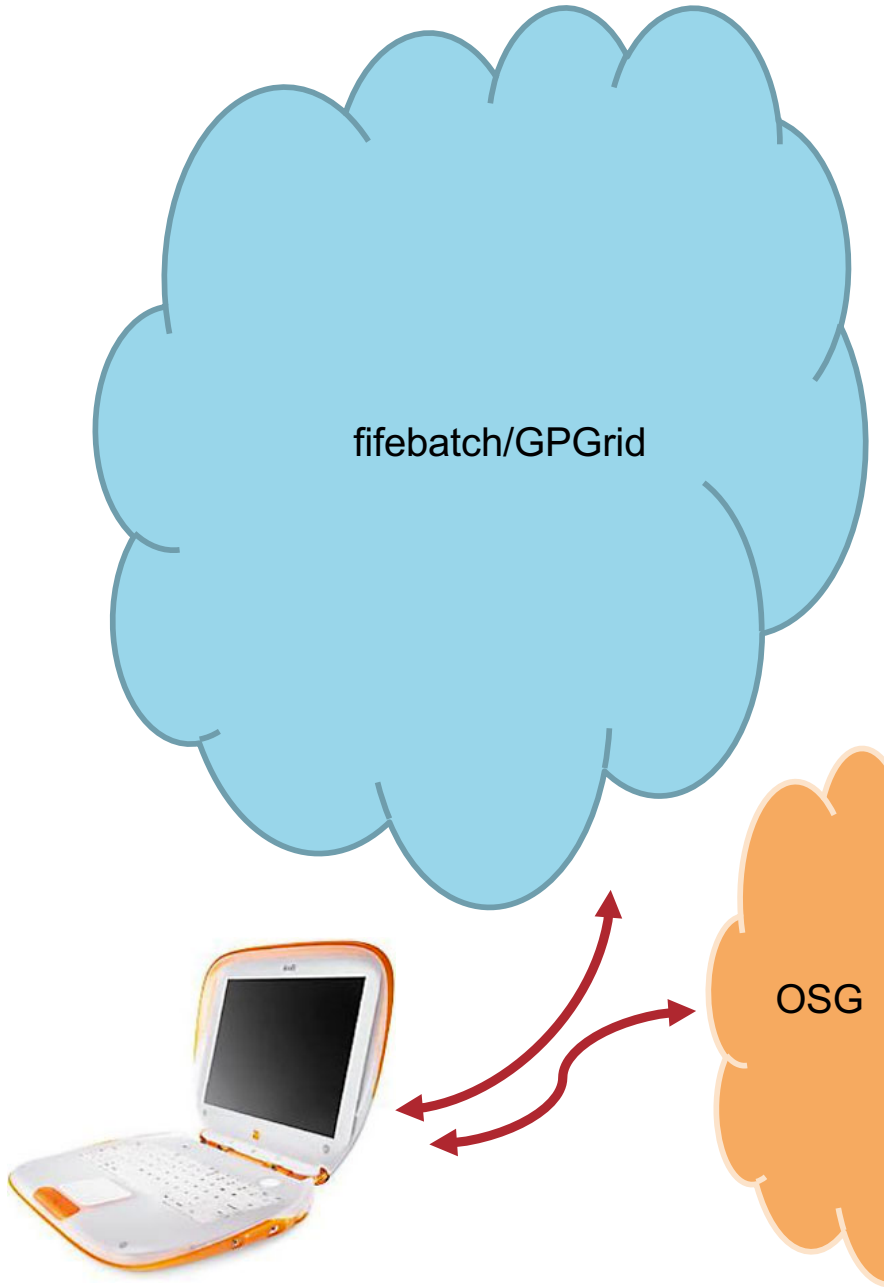
Job Submission and management architecture

- 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

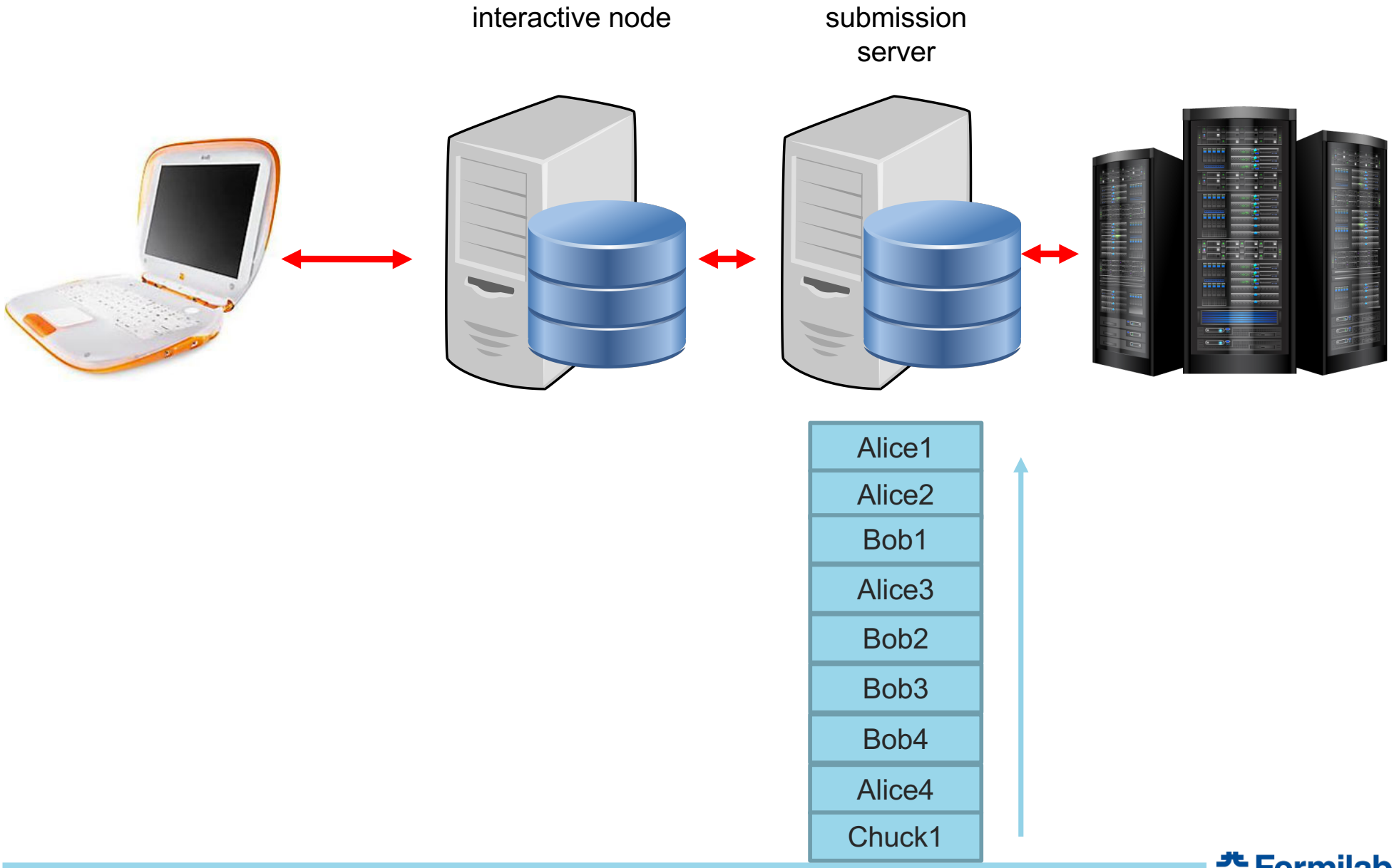


Lets start with the basics

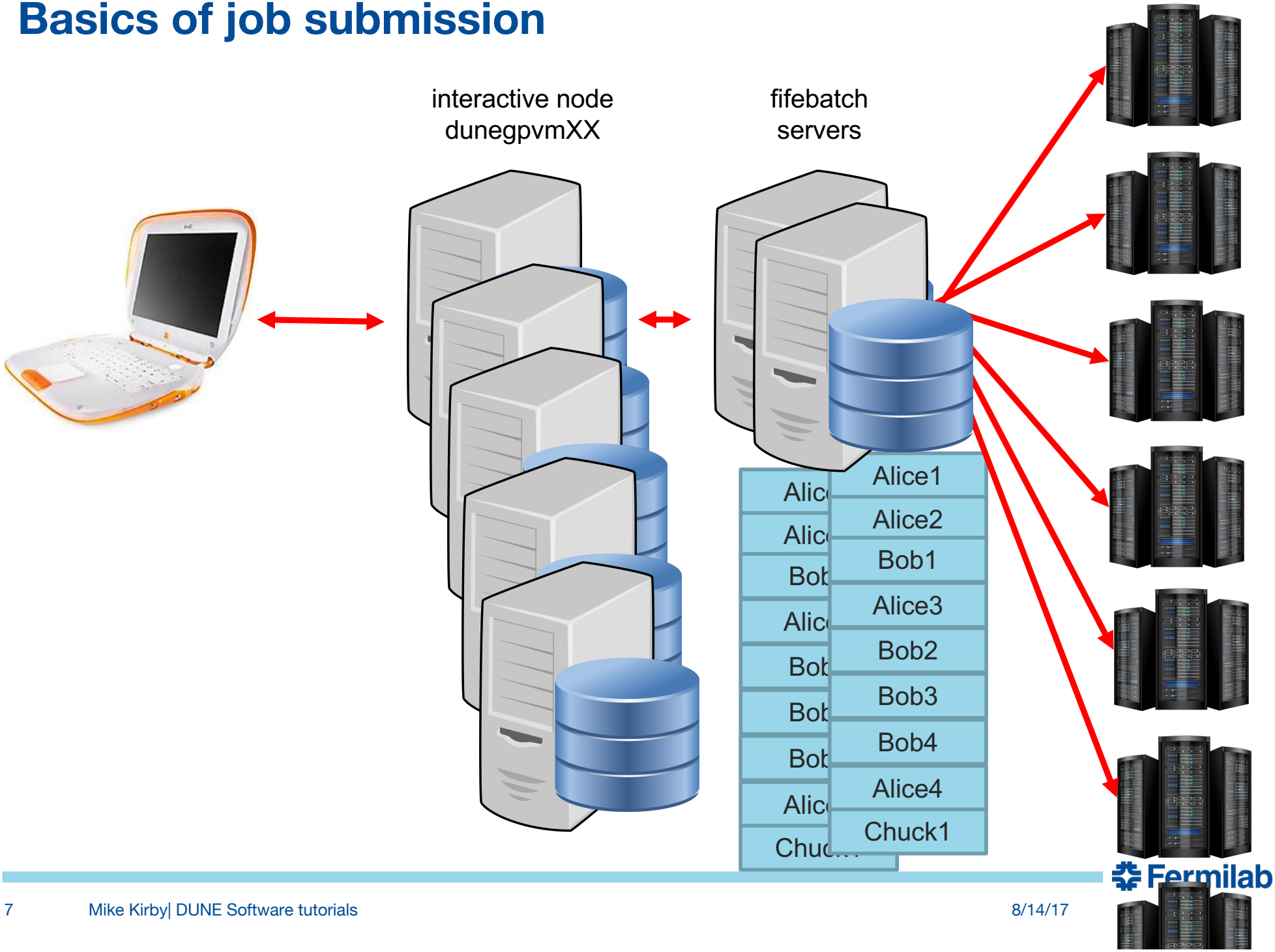
- What happens when you submit jobs to the grid?
- You are authenticated and authorized to submit – discussed later
- Submission goes into batch queue (HTCondor) and waits in line
- You (or your script) hand to jobsub an executable (script or binary)
- Jobs are matched to a worker node – what does this 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.



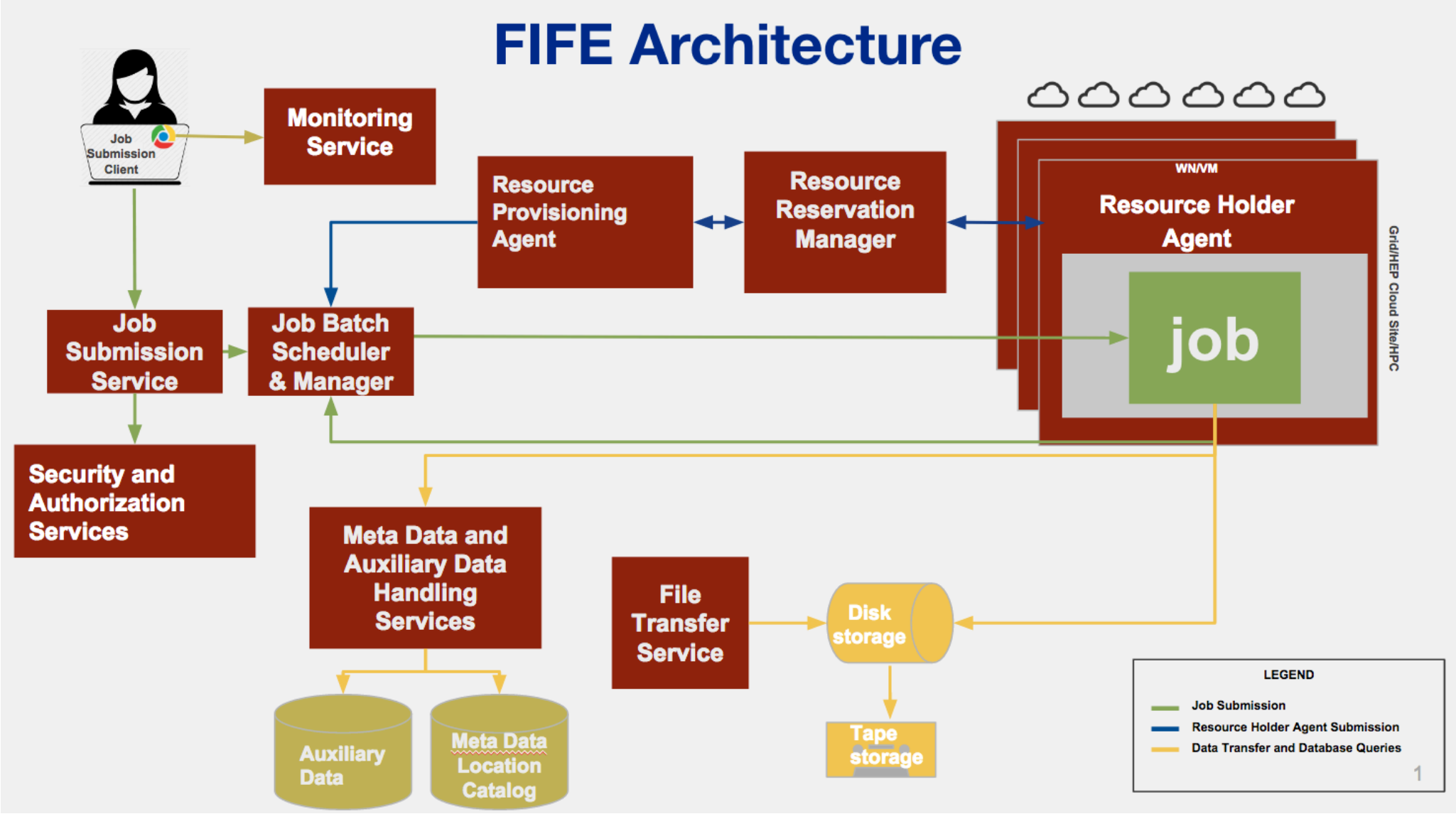
Basics of job submission



Basics of job submission



More complicated picture



Example script and submission command

- kinit
- ssh -K dunegpvm01.fnal.gov #don't everyone use dunegpvm01, spread out and use 02-10

Now that you've logged into DUNE interactive node, create a working area and copy over some example scripts

- cd /dune/app/users/${USER}
- mkdir dune_jobsub_tutorial
- cd dune_jobsub_tutorial
- cp /dune/app/users/kirby/dune_may2017_tutorial/*.sh `pwd`
- ls

There should be three scripts located in your current directory: basic_grid_env_test.sh, lar_grid_test.sh , and second_grid_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 dune --expected-lifetime=1h --memory=100MB --disk=2GB --resource-provides=usage_model=DEDICATED,OPPORTUNISTIC,OFFSITE file://`pwd`/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 how long it will take to run a single job 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

<dunegpvm01.fnal.gov> jobsub_submit -N 2 -G dune file://`pwd`/basic_grid_env_test.sh

/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_20170515_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

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_
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 dune --
jobid=17067704.0@fifebatch1.fnal.gov

#replace with the jobid that we highlighted earlier

Downloaded to 17067704.0@fifebatch1.fnal.gov.tgz

➢ ls -1

17067704.0@fifebatch1.fnal.gov.tgz

➢ tar xzf 17067704.0\@fifebatch1.fnal.gov.tgz

#replace with output of the ls -1 command (tab complete)

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
-rw-r--r-- 1 kirby dune 7048 May 15 17:00 17067704.0@fifebatch1.fnal.gov.tgz
```

- 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
  - the original fetchlog tarball

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

second grid submission script – part 1

```
<dunegpvm01.fnal.gov> more second_grid_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

umask 002 #set the read/write permission of files created to be 775

cd $_CONDOR_SCRATCH_DIR #change the working directory to be the HTCondor scratch area

echo "pwd is " `pwd`

GROUP=$EXPERIMENT

if [ -z $GROUP ]; then #bash statement asking if the variable GROUP has a non-zero value
# since GROUP doesn't have a non-zero value
# try to figure out what group the user is in
GROUP=`id -gn`
fi

case $GROUP in

dune)
SCRATCH_DIR=/pnfs/dune/scratch/users
;;
fermilab)
SCRATCH_DIR="/pnfs/fermilab/volatile"
;;
esac
```

second grid submission script – part 2

```
voms-proxy-info --all #print the grid proxy information to see values like user, VO, group, role, etc.

source /cvmfs/oasis.opensciencegrid.org/fermilab/products/common/etc/setup
#access the common UPS product area and configure environment to look for products in that area

setup ifdhc #set up environment to use the default Intensity Frontier Data Handling Client

echo "Here is the your environment in this job: " > job_output_${CLUSTER}.${PROCESS}.log 2> job_output_${CLUSTER}.${PROCESS}.err #Creates file for loggin
g information, note that this is stdout only
env >> job_output_${CLUSTER}.${PROCESS}.log 2>> job_output_${CLUSTER}.${PROCESS}.err #since file already exists use ">>" instead of ">"
ls /dune/data >> job_output_${CLUSTER}.${PROCESS}.log 2>> job_output_${CLUSTER}.${PROCESS}.err #since file already exists use ">>" instead of ">"

echo "group = $GROUP"

if [ -z ${GRID_USER} ]; then
GRID_USER=`basename $X509_USER_PROXY | cut -d "_" -f 2`
fi

echo "GRID_USER = `echo $GRID_USER`"

sleep $[ ( $RANDOM % 10 )  + 1 ]m

if [ -z "$SCRATCH_DIR" ]; then
    echo "Invalid dCache scratch directory, not copying back"
    echo "I am going to dump the log file to the main job stdout in this case."
    cat job_output_${CLUSTER}.${PROCESS}.log
    cat job_output_${CLUSTER}.${PROCESS}.err
else
```

second grid submission script – part 3

```
export IFDH_DEBUG=1

    # first do lfdh ls to check if directory exists
    ifdh ls ${SCRATCH_DIR}/$GRID_USER
    # A non-zero exit value probably means it doesn't, so create it
    if [ $? -ne 0 && -z "$IFDH_OPTION" ]; then

        echo "Unable to read ${SCRATCH_DIR}/$GRID_USER. Make sure that you have created this directory and given it group write permission (chmod g+w ${S
CRATCH_DIR}/$GRID_USER)."
        exit 74
    else
        # directory already exists, so let's copy
        ifdh cp -D $IFDH_OPTION job_output_${CLUSTER}.${PROCESS}.log ${SCRATCH_DIR}/${GRID_USER}
        if [ $? -ne 0 ]; then
            echo "Error $? when copying to dCache scratch area!"
            echo "If you created ${SCRATCH_DIR}/${GRID_USER} yourself,"
            echo " make sure that it has group write permission."
            exit 73
        fi
        ifdh cp -D $IFDH_OPTION job_output_${CLUSTER}.${PROCESS}.err ${SCRATCH_DIR}/${GRID_USER}
        if [ $? -ne 0 ]; then
            echo "Error $? when copying to dCache scratch area!"
            echo "If you created ${SCRATCH_DIR}/${GRID_USER} yourself,"
            echo " make sure that it has group write permission."
            exit 72
        fi
    fi
fi

echo "End " `date`

exit 0
```

Notes about accessing software and libraries on OSG

- 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 from dCache, and then unwound into the scratch area
- details about tarball transfers available here:

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

Submitting the second script

- cd /dune/app/users/${USER}/dune_jobsub_tutorial
- jobsub_submit -N 2 -G dune --expected-lifetime=1h --memory=100MB --disk=2GB --resource-provides=usage_model=DEDICATED,OPPORTUNISTIC,OFFSITE file://`pwd`/second_grid_test.sh

/fife/local/scratch/uploads/dune/kirby/2017-05-15_171818.710430_8037

/fife/local/scratch/uploads/dune/kirby/2017-05-15_171818.710430_8037/second_grid_test.sh_20170515_171820_3877085_0_1_.cmd

submitting....

Submitting job(s)..

2 job(s) submitted to cluster 19874357.

JobsubJobId of first job: 19874357.0@fifebatch2.fnal.gov

Use job id 19874357.0@fifebatch2.fnal.gov to retrieve output

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

- What number of CPUs does the job need?
- How much total memory does the job need? does it depend on the input? have I tested the input?
- How much scratch hard disk scratch space does the job need to 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.)

Submitting a dunetpc job

- mkdir -p /pnfs/dune/persistent/users/${USER}/
- mkdir -p /pnfs/dune/scratch/users/${USER}/
- cp /pnfs/dune/persistent/users/kirby/tutorial_prodsingle_dune35t.fcl /pnfs/dune/persistent/users/${USER}/tutorial_prodsingle_dune35t.fcl
- cd /dune/app/users/${USER}/dune_jobsub_tutorial/
- jobsub_submit -N 1 -G dune --expected-lifetime=10m --memory=2000MB --disk=4GB --resource-provides=usage_model=DEDICATED,OPPORTUNISTIC,OFFSITE file://`pwd`/lar_grid_test.sh

/fife/local/scratch/uploads/dune/kirby/2017-05-15_173103.937045_4879

/fife/local/scratch/uploads/dune/kirby/2017-05-15_173103.937045_4879/lar_grid_test.sh_20170515_173105_4053202_0_1_.cmd

submitting....

Submitting job(s).

1 job(s) submitted to cluster 19874534.

JobsubJobId of first job: 19874534.0@fifebatch2.fnal.gov

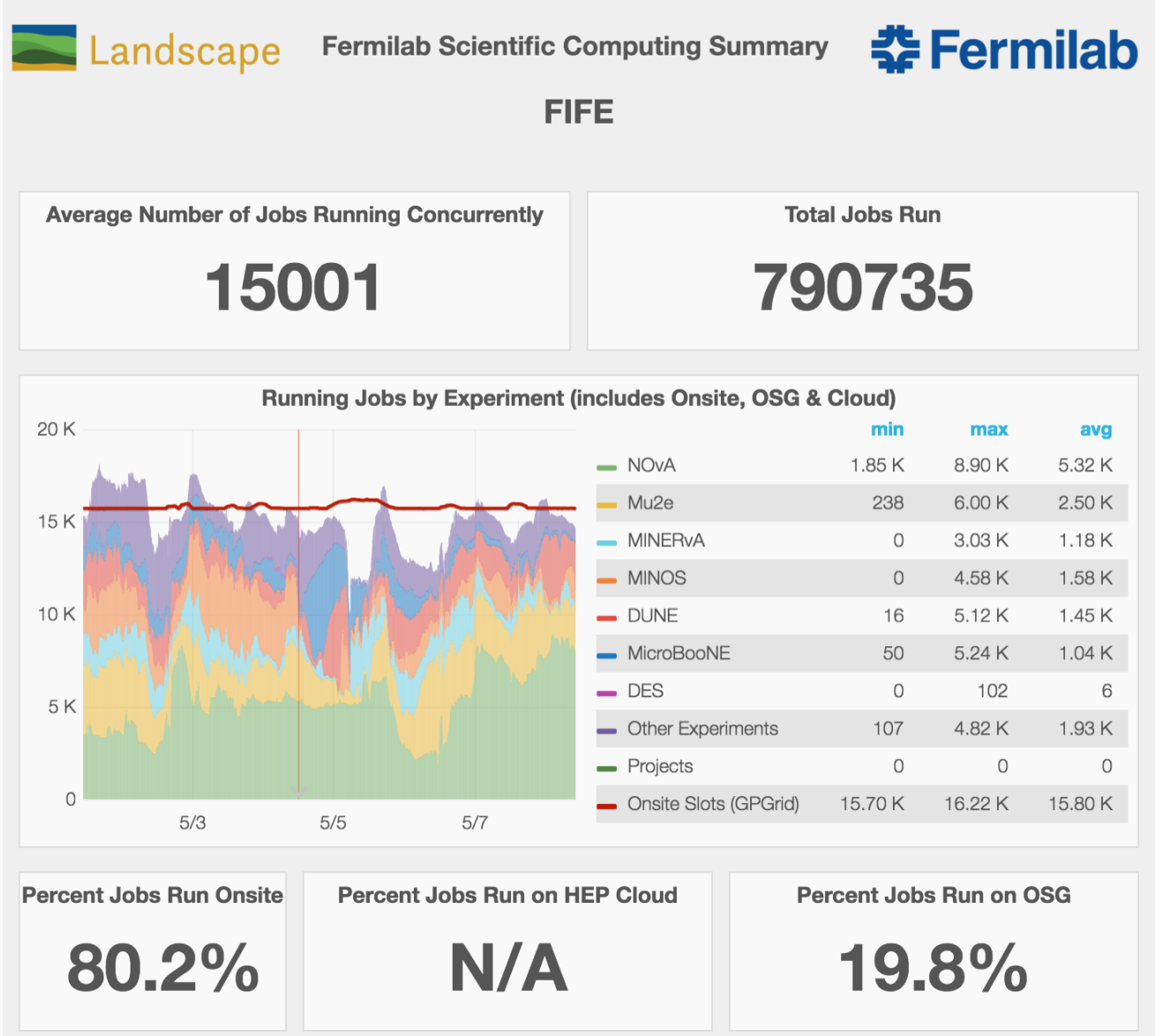
Use job id 19874534.0@fifebatch2.fnal.gov to retrieve output

Some additional links useful for writing scripts

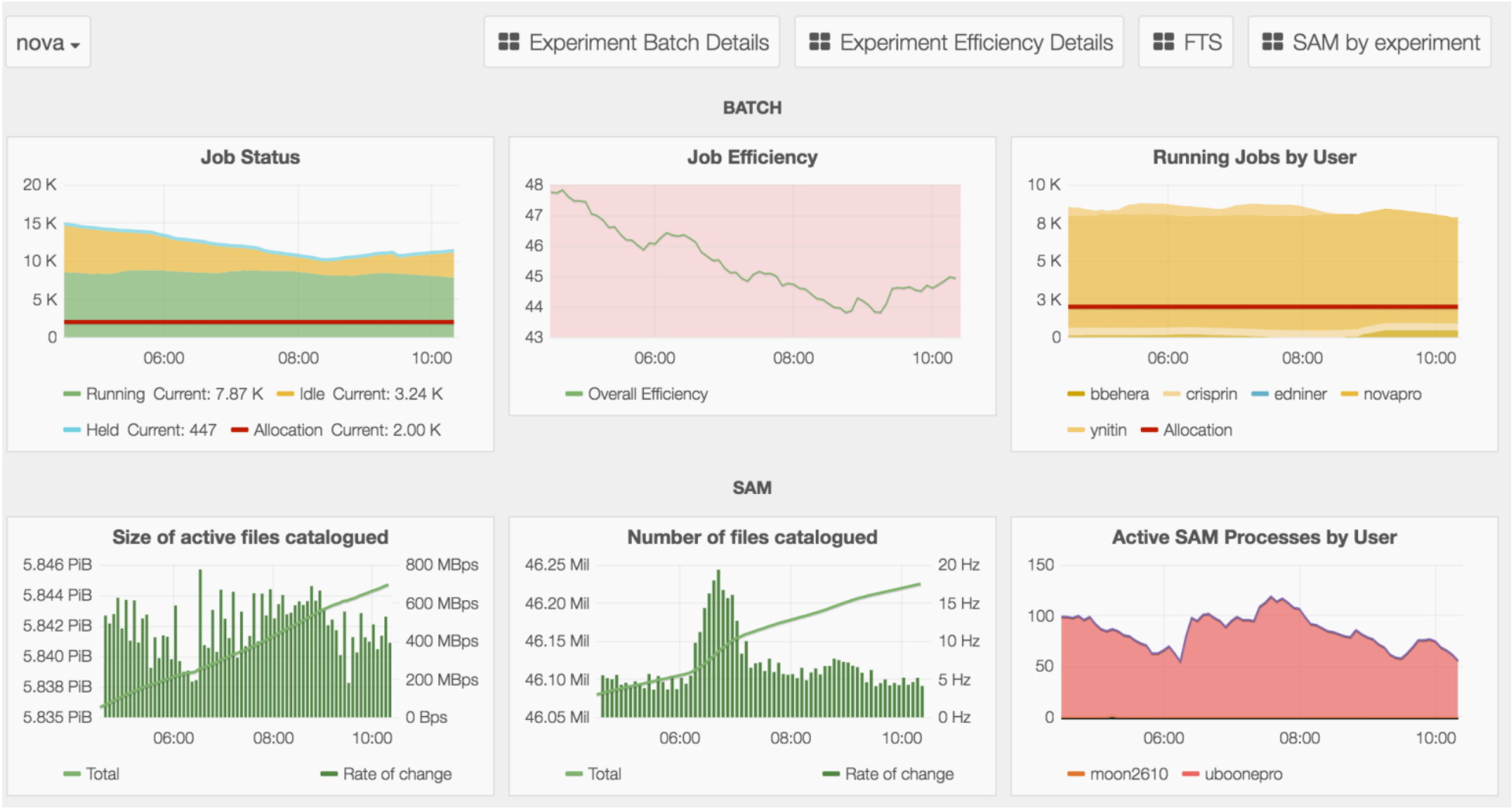
- List of environment variables on worker nodes:
  - https://cdcvs.fnal.gov/redmine/projects/jobsub/wiki/Jobsub_enviornment_variables

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

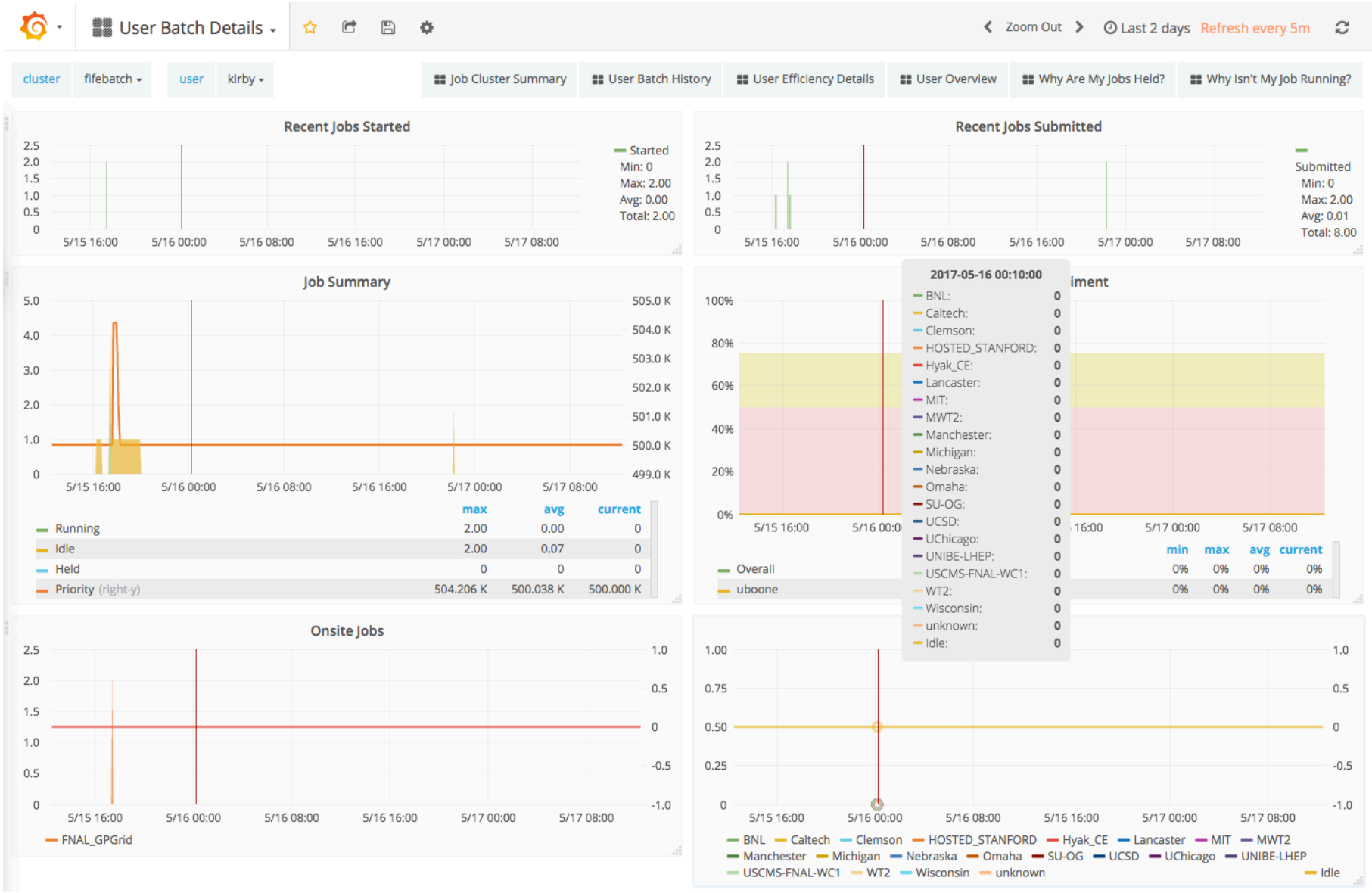


Detailed profiling of experiment operations



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

User Batch Details Dashboard



Backup

What if I want to do this in larbatch?

https://redmine.fnal.gov/redmine/projects/larbatch/wiki/User_guide

- the configuration file for larbatch is an xml file
- examples is found here:

https://cdcvs.fnal.gov/redmine/projects/dunetpc/wiki/Using_project_python

- <fcl>/pnfs/dune/persistent/users/kirby/tutorial_prodsingle_dune35t.fcl</fcl>
- <outdir>/pnfs/dune/scratch/users/kirby<outdir>
- <logdir>/pnfs/dune/scratch/users/kirby/log<logdir>
- <workdir>/pnfs/dune/scratch/users/kirby/work<workdir>
- <numjobs>2</numjobs>
- <numevents>1</numevnts>
- <group>dune</dune>

➢ project.py --xml /dune/app/users/kirby/dune_may2017_tutorial/dune_tutorial.xml --stage gen --submit