Introduction to DUNE Computing

Eileen Berman (stealing from many people)

DUNE Physics Week

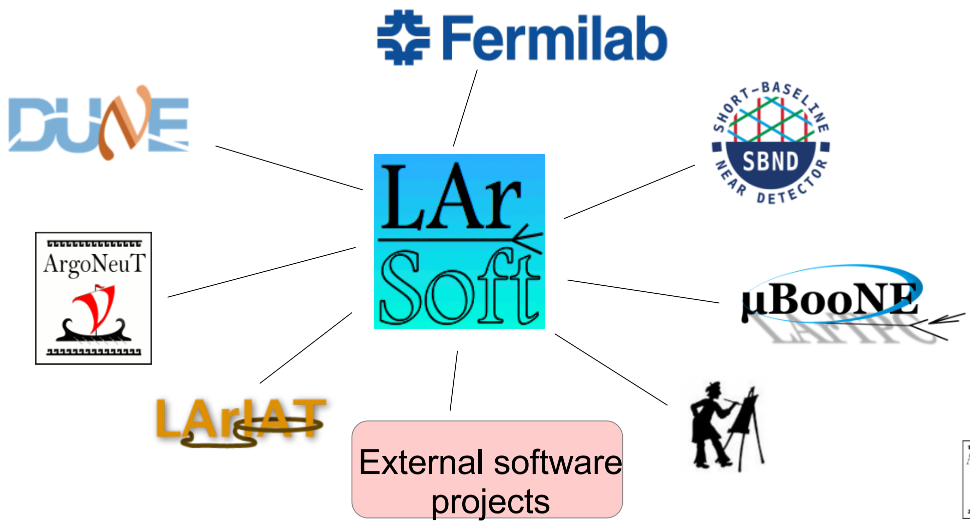
Nov 14, 2017

What Does This Include?

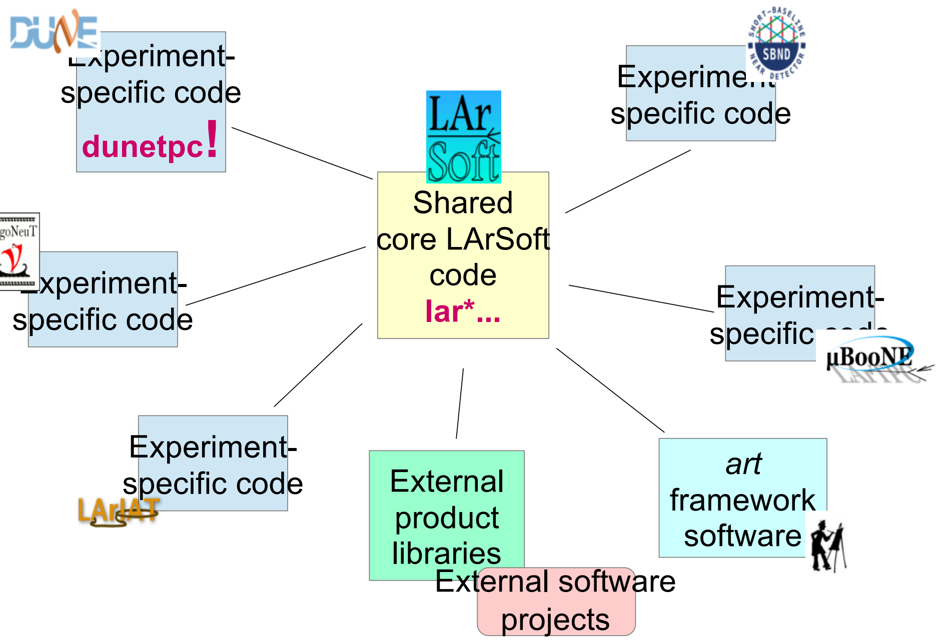
- **LArSoft** (thanks to Erica Snider)
- **Gallery** (thanks to Marc Paterno)
- **Data Management (Storage)** (thanks to Pengfei Ding and Marc Mengel)
- **FIFE Tools (Grid Submission)** (thanks to Mike Kirby)
- **Best Practices** (thanks to Ken Herner)

What is LArSoft?

- LArSoft is a collaboration of experiments



- LArSoft is a body of code



LArSoft Code

- The code for each product lives in a set of git repositories at Fermilab

larcore	Low level utilities
larcoreobj	Low level data products
larcorealg	Low level utilities
lardata	Data products
lardataobj	Data products
lartoolobj	Low level art tool interfaces (new!)
larsimtool	Low level simulation tool implementations (new!)
lardataalg	Low level algorithms
larevt	Low level algorithms that use data products
larsim	Simulation code
larreco	Primary reconstruction code
larana	Secondary reconstruction and analysis code
lareventdisplay	LArSoft-based event display
larpandora	LArSoft interface to Pandora
larexamples	Placeholder for examples

LArSoft Code

- The code for each product lives in a set of git repositories

larcore	Low level utilities
larcoreobj	Low level data products
larcorealg	Low level utilities
lardata	Data products
lardataobj	Data products

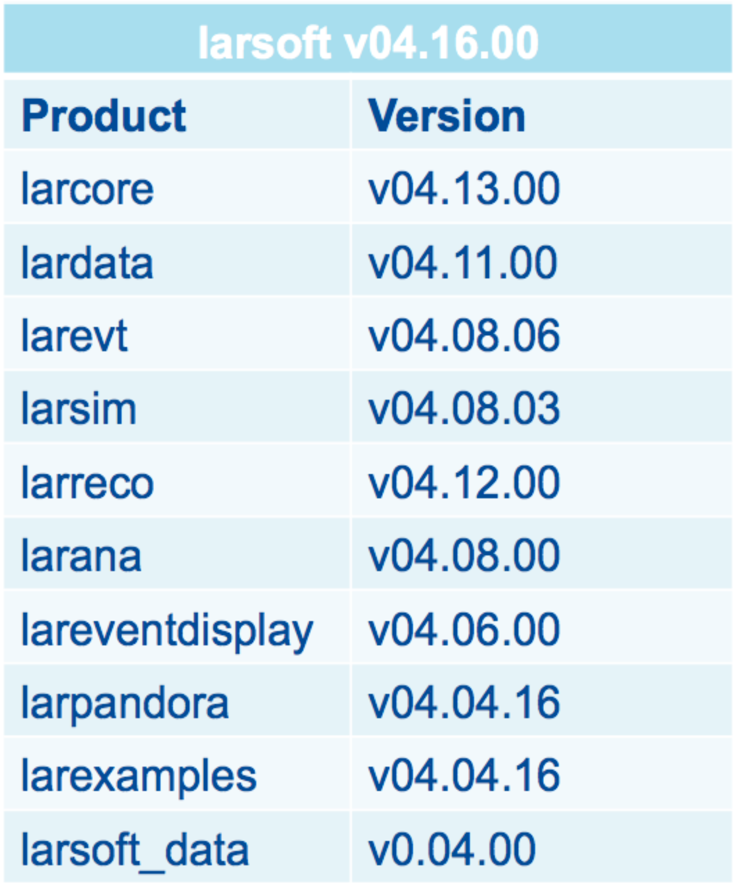
1) All publicly accessible at http://cdcvs.fnal.gov/projects/<repository name>

2) For read/write access:
   ssh://p-<repository name>@cdcvs.fnal.gov/cvs/projects/<repository name>
   (requires valid kerberos ticket)

larana	Secondary reconstruction and analysis code
lareventdisplay	LArSoft-based event display
larpandora	LArSoft interface to Pandora
larexamples	Placeholder for examples

What is a LArSoft Release?

- A LArSoft release is a **consistent set of LArSoft products** built from **tagged versions of code** in the repositories
  - Implicitly includes corresponding versions of all external dependencies used to build it
  - Each release of LArSoft has a **release notes** page on scisoft.fnal.gov
    - http://scisoft.fnal.gov/scisoft/bundles/larsoft/<version>/larsoft-<version>.html
- larsoft
  - An umbrella ups product that binds it all together under one version, one setup command
    - setup larsoft v06_06_00 -q …
- larsoft_data
  - A ups product with large configuration files (photon propagation lookup libraries, radiological decay spectra, supernova spectra)



larsoft v04.16.00

Product	Version
larcore	v04.13.00
lardata	v04.11.00
larevt	v04.08.06
larsim	v04.08.03
larreco	v04.12.00
larana	v04.08.00
lareventdisplay	v04.06.00
larpandora	v04.04.16
larexamples	v04.04.16
larsoft_data	v0.04.00

UPS is a tool that allows you to switch between using different versions of a product

What is a LArSoft Release?

- A LArSoft release is a **consistent set of LArSoft products** built from **tagged versions of code** in the repositories
  - Implicitly includes corresponding versions of all external dependencies used to build it
  - Each release of LArSoft has a **release notes** page on scisoft.fnal.gov

larsoft v04.16.00	
Product	**Version**
larcore	v04.13.00
lardata	v04.11.00
larevt	v04.08.06

1) dunetpc is DUNE's experiment software built using LArSoft/art

2) A dunetpc release (and UPS product) is bound to a particular release of LArSoft

3) By convention, the version numbering is kept in sync, aside from possible patching of production releases

- `larsoft_data`
  - A ups product with large configuration files (photon propagation lookup libraries, radiological decay spectra, supernova spectra)

UPS is a tool that allows you to switch between using different versions of a product

LArSoft and the art Framework

- LArSoft is built on top of the *art* event processing framework
- The *art* framework
  - Reads events from user-specified input sources
  - Invokes user-specified modules to perform reconstruction, simulation analysis, event-filtering tasks
  - May write results to one or more output files
- Modules
  - Configurable, dynamically loaded, user-written units with entry points called at specific times within the event loop
  - Three types
    - Producer: may modify the event
    - Filter: like a Producer, but may alter flow of module processing within an event
    - Analyzer: may read information from an event, but not change it

LArSoft and the art Framework

- Services
  - Configurable global utilities registered with framework, with entry points to event loop transitions and whose methods may be accessed within modules
- Tools
  - Configurable, local utilities callable inside modules
  - See this talk at LArSoft Coordination Meeting for details on tools
- The run-time configuration of art, modules, services and tools specified in FHiCL (.fcl files)
  - See art workbook and FHiCL quick-start guide for more information on using FHiCL to configure *art* jobs
  - See https://cdcvs.fnal.gov/redmine/projects/fhicl-cpp/wiki/Wiki for C++ bindings and using FHiCL parameters inside programs

Running LArSoft

- (From the homework) Don't need to build code, use DUNE's code

```
# setup the dunetpc  environment
source /cvmfs/dune.opensciencegrid.org/products/dune/setup_dune.sh
setup dunetpc v06_34_00 -q e14:prof
lar -n 1 -c prod_muminus_0.1-5.0GeV_isotropic_dune10kt_1x2x6.fcl
```

- The '*source*' line sets up versions of the software ups products and the environment needed to run the DUNE-specific code using LArSoft
- The '*setup*' line says to use version 06_34_00 of the dunetpc software ups product. This release is bound to a particular release of LArSoft
- The '*lar*' line runs the art framework using a DUNE 'fcl' file as input, which defines what the software is supposed to do

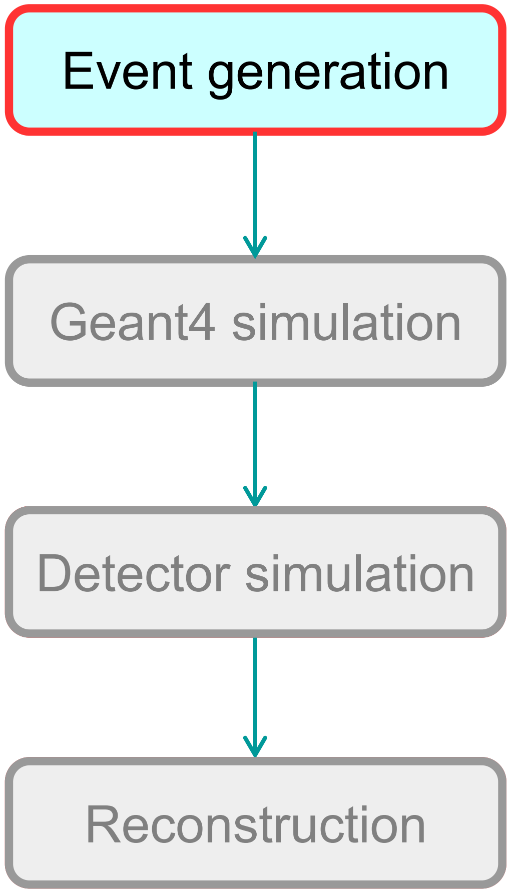
Running LArSoft – fcl Files

- How does ***art*** find the fcl file?
  - FHICL_FILE_PATH environment variable
  - Defined by setup of dunetpc and other software products
- How do I examine final parameter values for a given fcl file?
  - `fhicl-expand`
    - Performs all “#include” directives, creates a single output with the result
  - `fhicl-dump`
    - Parses the entire file hierarchy, **prints the final state** all FHiCL parameters
    - **Using the “--annotate” option, also lists the fcl file + line number** at which each parameter takes its final value
    - Requires FHICL_FILE_PATH to be defined
- How can I tell what the FHiCL parameter values are for a processed file?
  - `config_dumper`
    - Prints the full configuration for the processes that created the file

LArSoft – Processing Chain

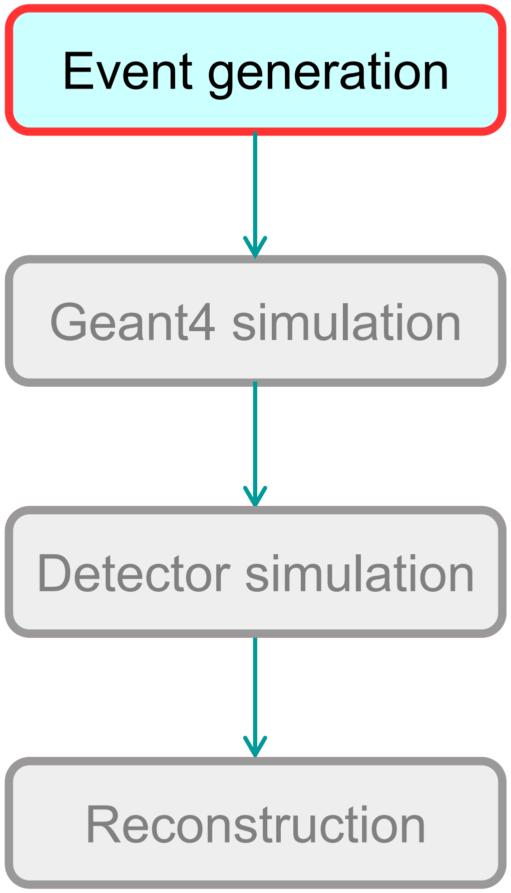
- Major processing steps are in a set of pre-defined fcl files

- First example was SingleGen Module
  - In larsim/larsim/EventGenerator
  - fcl was in dunetpc/fcl/dunefd/gen/single/



LArSoft – Processing Chain

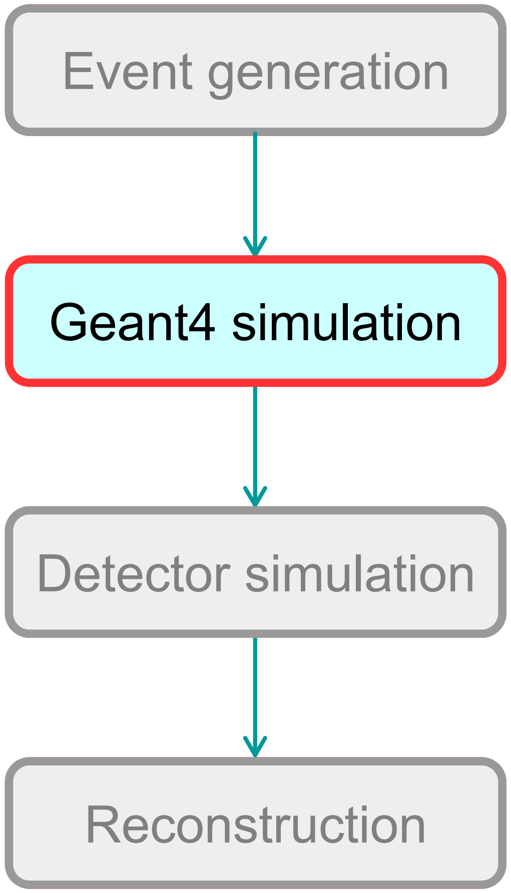
- Major processing steps are in a set of pre-defined fcl files



- Other event generation options
  - GENIE: `GENIEGen` module
  - NuWro: `NuWroGen` module
  - CORSIKA: `CORSIKAGen` module
  - CRY: `CosmicsGen` module
  - NDk: `NDKGen` module
  - `TextFileGen` module
  - When all else fails...reads a text file, produces simb::MCTruth
- larsim/larsim/EventGenerator/
  - Others in larsim/larsim/EventGenerator

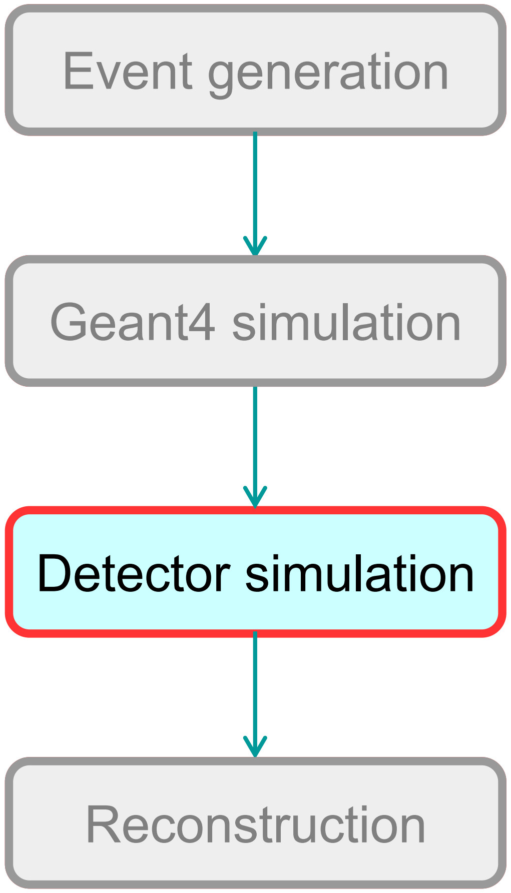
LArSoft – Processing Chain

- Geant4 simulation
  - Traces energy deposition, secondary interactions within LAr
  - Also performs electron / photon transport
  - `LArG4` module in `larsim/larsim/LArG4`
  - Note:
    - Many generator / simulation interfaces are defined in `nutools` product.
  - Homework fcl:
    - `standard_g4_dune10kt_1x2x6.fcl`
    - In `dunetpc/fcl/dunefd/g4/`



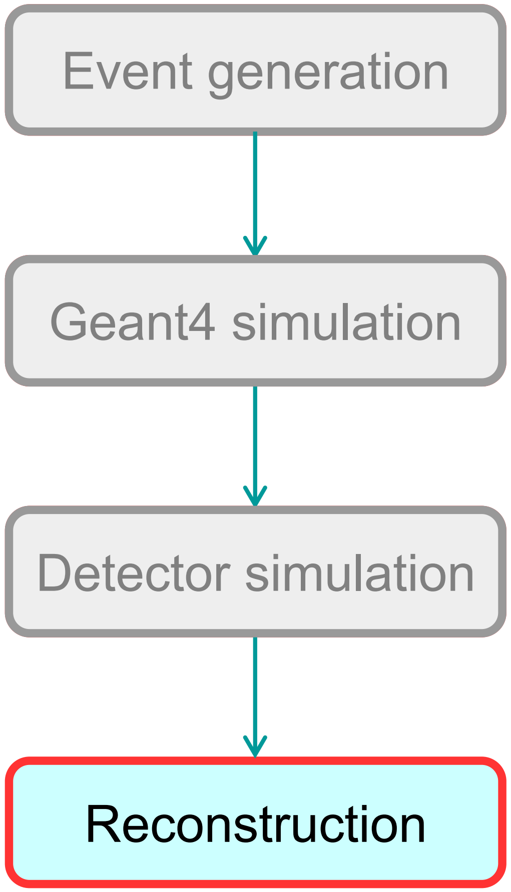
LArSoft – Processing Chain

- Detector simulation
  - Detector and readout effects
  - Field response, electronics response, digitization
  - Historically, most of this code is experiment-specific
    - `dunetpc`
    - More recently, the active development is part of wire-cell project with interfaces to LArSoft
  - Homework fcl:
    - `standard_detsim_dune10kt_1x2x6.fcl`
    - In `dunetpc/fcl/dunefd/detsim/`



LArSoft – Processing Chain

- Reconstruction
  - Performs pattern recognition, extracts information about physical objects and processes in the event
  - May include signal processing, hit-finding, clustering of hits, view matching, track and shower finding, particle ID
    - 2D and 3D algorithms
    - External RP interfaces for Pandora and Wire-cell
  - Homework fcl:
    - standard_reco_dune10kt_1x2x6.fcl
    - In dunetpc/fcl/dunfd/reco/



LArSoft – Modify Config of a Job

- Suppose you need to modify a parameter in a pre-defined job
- Several options. Here are two.
  - Option 1
    - Copy the fcl file that *defines* the parameter to the "pwd" for the `lar` command
    - Modify the parameter
    - Run lar -c … as before
      - The modified version will get picked because "." is always first in FHICL_FILE_PATH
  - Option 2
    - Copy the top-level fcl file to the "pwd" for the `lar` command
    - Add an override line to the top-level fcl file
      - E.g., in the homework generator job, all those lines at the bottom:

```
...
services.Geometry: @local::dune10kt_1x2x6_geo
source.firstRun: 20000014

physics.producers.generator.PDG: [ 13 ]              # mu-
physics.producers.generator.PosDist: 0               # Flat position dist.
...
```

LArSoft – Modify Code of a Job

In cases where configuration changes will not be sufficient, you will need to modify, build, then run code:

- Create a new working area from a fresh login + DUNE set-up

```
mkdir <working_dir>
cd <working_dir>
mrb newDev -v <version> -q <qualifiers>
```

(Note, if dunetpc/larsoft is already set up, then only need "mrb newDev")

- – This creates the three following directories inside <working_dir>

```
<working_dir>/localProducts_<MRB_PROJECT>_<version>_<qualifiers>
             /build_<os flavor>
             /srcs
```

Local products directory

Source directory

Build directory

LArSoft – Modify Code of a Job

In

An aside:

- `mrb` : multi-repository build system
- Purpose is to simplify building of multiple products pulled from separate repositories
- `setup mrb` command executed in experiment setup
- Most commonly used commands
  - `mrb --help` #prints list of all commands with brief descriptions
  - `mrb <command> --help` #displays help for that command
  - `mrb gitCheckout` #clone a repository into working area
  - `mrbsetenv` #set up build environment
  - `mrb build / install -jN` #build/install local code with N cores
  - `mrbslp` #set up all products in `localProducts`...
  - `mrb z` #get rid of everything in build area

LArSoft – Modify Code of a Job

- Set up local products and development environment
  - `source localProducts_<MRB_PROJECT>_<version>_<qualifiers>/setup`
  - Creates a number of new environment variables, including
    - `MRB_SOURCE` points to the srcs directory
    - `MRB_BUILDDIR` points to the build_... directory
    - Modifies `PRODUCTS` to include `localProducts`... as the first entry
- Check out the repository to be modified (and maybe others that depend on any header files to be modified)
  - cd $MRB_SOURCE
  - mrb g dunetpc # g is short for gitCheckout
  - Clones dunetpc from current head of “develop” branch
  - Adds the repository to top-level build configuration file (CMakeLists.txt)

LArSoft – Modify Code of a Job

- Make changes to the code…
  - Look in <working_dir>/srcs/<repository-name>
- Go to the build dir and setup the development environment
  - cd $MRB_BUILDDIR
  - mrbsetenv
- Build the code
  - mrb b   # b is short for build
- Install local ups products from the code you just built
  - mrb i   # i is short for install. This will do a build also.
  - Files are re-organized and moved into localProducts... directory
    - All fcl files are put into a top-level “job” directory with no sub-structure
    - All header files are put into a top-level “include” directory with sub-directories
    - Other files are moved to various places, including source files, while some, such as build configuration files, are ignored and not put anywhere in the ups product

LArSoft – Modify Code of a Job

- Now set-up the local versions of the products just installed
  - cd $MRB_TOP
  - mrbslp
- Run the code you just built
  - lar -c <whatever fcl file you were using> …
- Another useful command: get rid of the code you just built so you can start over from a clean build
  - cd $MRB_BUILDER
  - mrb z

LArSoft – Navigating art/root files

- lar -c eventdump.fcl -s <file>
  - Uses the FileDumperOutput module to produce this:

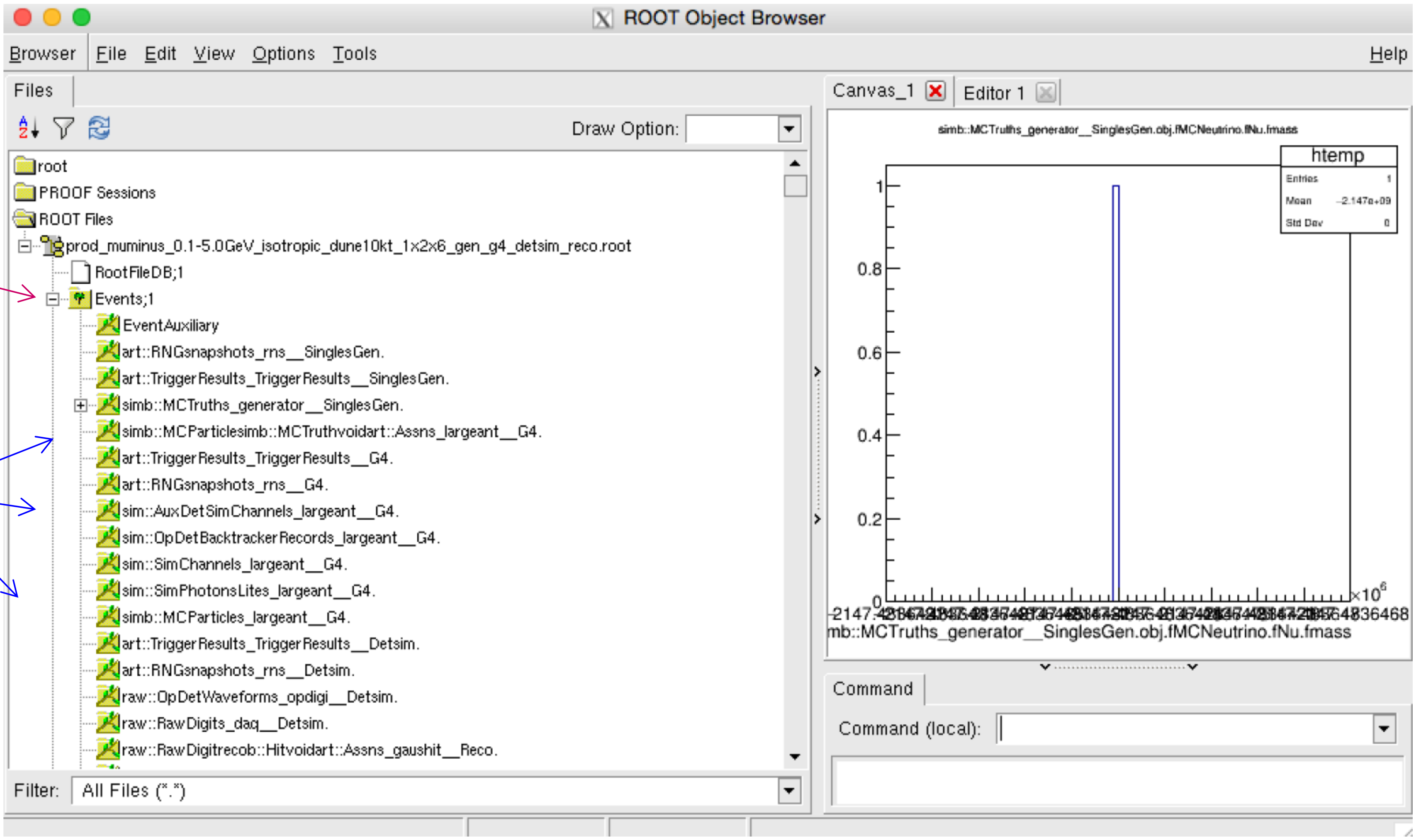
```
Begin processing the 1st record. run: 20000014 subRun: 0 event: 1 at 17-May-2017 01:59:11 CDT
PRINCIPAL TYPE: Event
PROCESS NAME | MODULE_LABEL.. | PRODUCT INSTANCE NAME | DATA PRODUCT TYPE.......................................... | SIZE
SinglesGen.. | generator..... | ..................... | std::vector<simb::MCTruth>................................. | ...1
SinglesGen.. | rns........... | ..................... | std::vector<art::RNGsnapshot>.............................. | ...1
SinglesGen.. | TriggerResults | ..................... | art::TriggerResults........................................ | ...-
G4.......... | largeant...... | ..................... | std::vector<sim::OpDetBacktrackerRecord>................... | ..99
G4.......... | rns........... | ..................... | std::vector<art::RNGsnapshot>.............................. | ...2
G4.......... | TriggerResults | ..................... | art::TriggerResults........................................ | ...-
G4.......... | largeant...... | ..................... | std::vector<simb::MCParticle>.............................. | ...8
G4.......... | largeant...... | ..................... | std::vector<sim::AuxDetSimChannel>......................... | ...0
G4.......... | largeant...... | ..................... | art::Assns<simb::MCTruth,simb::MCParticle,void>............ | ...8
G4.......... | largeant...... | ..................... | std::vector<sim::SimChannel>............................... | .684
G4.......... | largeant...... | ..................... | std::vector<sim::SimPhotonsLite>........................... | ..99
Detsim...... | TriggerResults | ..................... | art::TriggerResults........................................ | ...-
Detsim...... | opdigi........ | ..................... | std::vector<raw::OpDetWaveform>............................ | .582
Detsim...... | daq........... | ..................... | std::vector<raw::RawDigit>................................. | 4148
Detsim...... | rns........... | ..................... | std::vector<art::RNGsnapshot>.............................. | ...1
Reco........ | TriggerResults | ..................... | art::TriggerResults........................................ | ...-
Reco........ | trajcluster... | ..................... | std::vector<recob::Vertex>................................. | ...2
Reco........ | pmtrajfit..... | kink................. | std::vector<recob::Vertex>................................. | ...0
Reco........ | pandora....... | ..................... | std::vector<recob::PCAxis>................................. | ...0
Reco........ | pmtrack....... | ..................... | std::vector<recob::Vertex>................................. | ...2
Reco........ | pandoracalo... | ..................... | art::Assns<recob::Track,anab::Calorimetry,void>............ | ...3
Reco........ | pandora....... | ..................... | art::Assns<recob::PFParticle,recob::SpacePoint,void>....... | .581
...
...
```

LArSoft – Navigating art/root files

- Examine the file within a Tbrowser (in root)

The event TTree

Data product branches



```
root
PROOF Sessions
ROOT Files
  prod_muminus_0.1-5.0GeV_isotropic_dune10kt_1x2x6_gen_g4_detsim_reco.root
    RootFileDB;1
    Events;1
      EventAuxiliary
      art::RNGsnapshots_rns__SinglesGen.
      art::TriggerResults_TriggerResults__SinglesGen.
      simb::MCTruths_generator__SinglesGen.
      simb::MCParticlesimb::MCTruthvoidart::Assns_largeant__G4.
      art::TriggerResults_TriggerResults__G4.
      art::RNGsnapshots_rns__G4.
      sim::AuxDetSimChannels_largeant__G4.
      sim::OpDetBacktrackerRecords_largeant__G4.
      sim::SimChannels_largeant__G4.
      sim::SimPhotonsLites_largeant__G4.
      simb::MCParticles_largeant__G4.
      art::TriggerResults_TriggerResults__Detsim.
      art::RNGsnapshots_rns__Detsim.
      raw::OpDetWaveforms_opdigi__Detsim.
      raw::RawDigits_daq__Detsim.
      raw::RawDigitrecob::Hitvoidart::Assns_gaushit__Reco.
```

simb::MCTruths_generator__SinglesGen.obj.fMCNeutrino.fNu.fmass

htemp	
Entries	1
Mean	-2.147e+09
Std Dev	0

LArSoft – Navigating art/root files

- Dumping individual data products
  - ls $LARDATA_DIR/source/lardata/ArtDataHelper/Dumpers
  - ls $LARSIM_DIR/source/larsim/MCDumpers
  - Dedicated modules named "Dump<data product>" produce formatted dump of contents of that data product
  - Run then with fcl files in those same directories: `dump_<data type>.fcl`
  - E.g.: `lar -c dump_clusters.fcl -s <file>`
    - General fcl files are in $LARDATA_DIR/job

Gallery – Reading Event Data Outside of art

- *gallery* is a (UPS) product that provides libraries that support the reading of event data from *art*/ROOT data files outside of the *art* event-processing framework executable.
- *gallery* comes as a binary install; you are not building it.
- *art* is a framework, *gallery* is a library.
- When using *art*, you write libraries that "plug into" the framework. When using *gallery*, you write a main program that uses libraries.
- When using *art*, the framework provides the event loop. When using *gallery*, you write your own event loop.
- *art* comes with a powerful and safe (but complex) build system. With *gallery*, you provide your own build system.

Gallery – What does it do?

- gallery provides access to event data in *art*/ROOT files outside the *art* event processing framework executable:
  - without the use of EDProducers, EDAnalyzers, *etc.*, thus
  - without the facilities of the framework (*e.g.* callbacks for runs and subruns, *art* services, writing of *art*/ROOT files, access to non-event data).
- You can use *gallery* to write:
  - compiled C++ programs,
  - ROOT macros,
  - (Using PyROOT) Python scripts.
- You can invoke any code you want to compile against and link to. Be careful to avoid introducing binary incompatibilities.

Gallery – When should I Use It?

- If you want to use either Python or interactive ROOT to access *art*/ROOT data files.
- If you do not want to use framework facilities, because you do not need the abilities they provide, and only need to access event data.
- If you want to create an interactive program that allows random navigation between events in an *art*/ROOT data file (e.g., an event display).

Gallery – When Should I NOT Use IT?

- When you need to use framework facilities (run data, subrun data, metadata, services, *etc.*)
- When you want to put something into the Event. For the *gallery* Event, you can not do so. For the *art* Event, you do so to communicate the product to another module, or to write it to a file. In *gallery*, there are no (framework!) modules, and *gallery* can not write an *art*/ROOT file.
- If your only goal is an ability to build a smaller system than your experiment's infrastructure provides, you might be interested instead in using the build system *studio*: https://cdcvs.fnal.gov/redmine/projects/studio/wiki.
  You can use *studio* to write an *art* module, and compile and link it, without (re)building any other code.

Data Management

- Storage volumes:

Storage systems	Path on GPVMs
BlueArc App	/dune/app/users/${USER}
BlueArc Data	/dune/data/users/${USER}; /dune/data2/users/${USER}
Scratch dCache	/pnfs/dune/scratch/users/${USER}
Persistent dCache	/pnfs/dune/persistent/users/${USER}
Tape-backed dCache	/pnfs/dune/tape_backed/users/${USER}

  - More volumes to be added (EOS at CERN, /pnfs at BNL etc.)
- Data handling tools:
  - *IFDH*
  - *SAM and SAM4Users*"

Data Management - BlueArc

- a Network Attached Storage (NAS) system;
- App area, /dune/app
  - used primarily for code and script development;
  - **should not be used to store data**;
  - slightly lower latency;
  - smaller total storage (200 GB/user).
- Data area, /dune/data or /dune/data2
  - used primarily for storing ntuples and small datasets (200 GB/user);
  - higher latency than the app volumes;
  - full POSIX access (read/write/modify);
  - not mounted on any of the GPGrid or OSG worker nodes;
  - throttled to have a maximum of 5 transfers at any given time.
  - Will not be able to copy to/from /dune/data areas in a grid job come January 2018.

Data Management - BlueArc

- a Network Attached Storage (NAS) system;
- App area, /dune/app
  - used primarily for code and script development;

DON'T USE BlueArc volumes in grid jobs!
DON'T code NEW jobs using BlueArc!
Access to them is going away in Jan 2018!!

  - not mounted on any of the GPGrid or OSG worker node;
  - throttled to have a maximum of 5 transfers at any given time.
  - Will not be able to copy to/from /dune/data areas in a grid job come January 2018.

Data Management - dCache

- A lot of data distributed among a large number of heterogeneous server nodes.
- Although the data is highly distributed, dCache provides a file system tree view of its data repository.
- dCache separates the namespace of its data repository (pnfs) from the actual physical location of the files;
- the minimum data unit handled by dCache is a file.
- files in dCache become *immutable*
  - Opening an existing file for write or update or append fails;
  - Opening an existing file for read works;
  - Opens can be queued until a dCache door (I/O protocols provided by I/O servers) is available (good for batch throughput but annoying for interactive use).

Data Management - dCache

Areas	Location	Storage type	Space	File lifetime	When disk/tape is full
Scratch	/pnfs/dune/scratch	Disk	No hard limit. Scratch area is shared by all experiments (>1PB as of today).	refer to the scratch lifetime plot: http://fndca.fnal.gov/dcache/lifetime/PublicScratchPools.jpg	LRU eviction policy, new files will overwrite LRU files.
Persistent	/pnfs/dune/persistent	Disk	190 TB	> 5 years, Managed by DUNE	No more data can be written when quota is reached.
Tape-backed	/pnfs/dune/tape_backed	Tape	Pseudo-infinite	>10 years, Permanent storage.	New tape will be added.

Data Management – Scratch dCache

- Copy needed files to scratch, and have jobs fetch from there, **rather than from BlueArc**
- Least Recently Used (LRU) eviction policy applies in scratch dCache
- Scratch lifetime: http://fndca.fnal.gov/dcache/lifetime/PublicScratchPools.jpg
- NFS access is not as reliable as using ifdh, xrootd
- **Don't put thousands of files into one directory in dCache;**

Note: Do not use "rsync" with any dCache volumes.

Data Management – Persistent/Tape-backed dCache

- Storing files into persistent or tape-backed area is only recommended with “sam_clone_dataset” tool, or other tools that automatically declare locations to SAM.

- Grid output files should be written to the scratch area first. If finding those files is valuable for longer term storage, they can be put into the persistent or tape-backed area with SAM4users tool:
  - sam_add_dataset, create a SAM dataset for files in the scratch area;
  - sam_clone_dataset , clone the dataset to the persistent or tape-backed area;
  - sam_unclone_dataset , delete the replicas of the dataset files in the scratch area.
  - NOTE: SAM4users will change your filename to insure it is unique.

Data Management – Best Practices

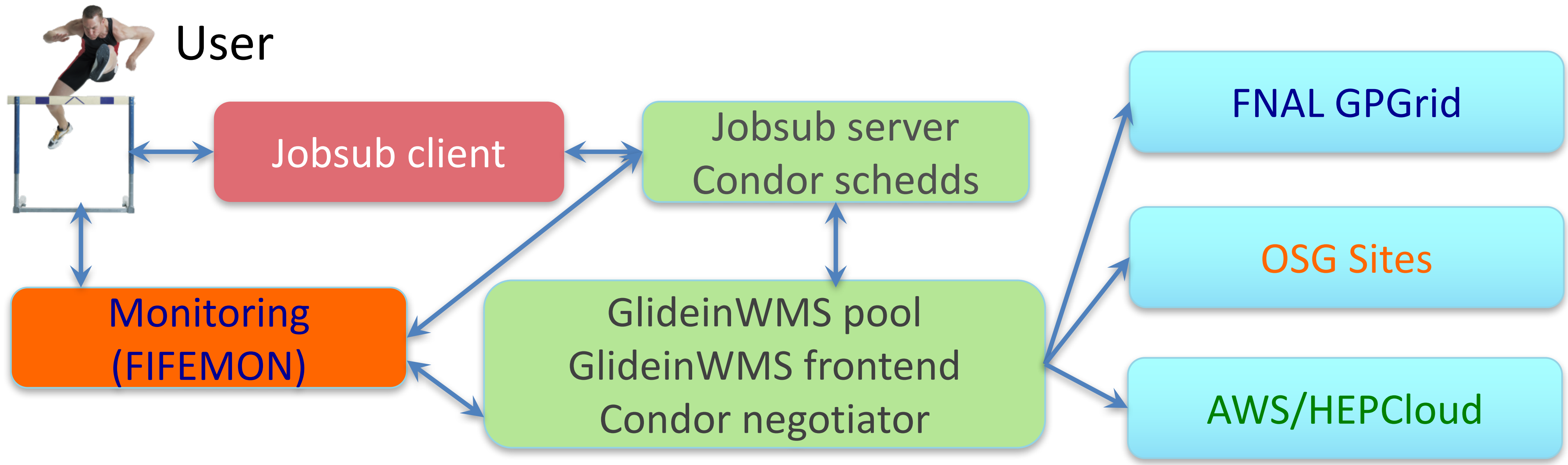
- **DO NOT use BlueArc areas for grid jobs**; Access is going away in January 2018.
  - /dune/data and /dune/data2 were never mounted on grid nodes
  - /dune/app is going away in January from grid nodes
- Avoid using "rsync" on any dCache volumes;
- Store files into dCache scratch area first;
- Always use SAM to do bookkeeping for files under persistent or tape-backed areas;
- For higher reliability, use "ifdh" or "xrootd" in preference to NFS for accessing files in dCache.

FIFE Tools (Grid Job Submission)

- The **F**abr**I**c for **F**rontier **E**xperiments centralized services includes:
  - 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

FIFE Tools – Job Submission

- Users interface with the batch system via “jobsub” tool
- Common monitoring provided by FIFEMON tools



FIFE Tools – Job Submission

- What happens when you submit jobs to the grid?
  - You are authenticated and authorized to submit
  - 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
  - Server distributes your executable to the worker nodes
  - Executable runs on remote cluster and NOT as your user id – no home area, no NFS volume mounts, etc.

FIFE Tools – Job Submission

- kinit
- ssh -K dunegpvm01.fnal.gov #don't everyone use duneand 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`
- source /cvmfs/fermilab.opensciencegrid.org/products/common/etc/setup
- setup jobsub_client
- 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

FIFE Tools – Job Submission (jobsub)

- 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
  - *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
- jobsub command outputs jobid needed to retrieve job output
  - **EX**: JobsubJobId of first job: 17067704.0@jobsub01.fnal.gov

FIFE Tools – Job Submission

- What do I need to know to submit a job?
  - 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.)

FIFE Tools – Submitting Production Jobs

- To submit Production jobs you need to add to the jobsub_submit command line –
  - --role=Production
- And you must be authorized for this role in DUNE
- All subsequent jobsub commands you issue must also use the
  - --role=Production option.

FIFE Tools – Check on Jobs

- jobsub_q --user=${USER}
  - *USER* specifies the uid whose jobs you want the status of.
  - Job status 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)
  - With the *--held* parameter, held reason codes are not printed out. Need to use FIFEMON.
- Additional commands
  - 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

FIFE Tools – Fetching Job Logs

- Need your jobid
- jobsub_fetchlog -G dune --jobid=nnnnnn
- Returns a tarball with the following in it –
  - shell script sent to the jobsub server (.sh)
  - wrapper script created by jobsub server to set environment variables (.sh)
  - condor command file sent to condor to put job in queue (.cmd)
  - an empty file
  - stdout of the bash shell run on the worker node (.out)
  - stderr of the bash shell run on the worker node (.err)
  - condor log for the job (.log)
  - the original fetchlog tarball

FIFE Tools – Accessing Software/Libraries

- 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:

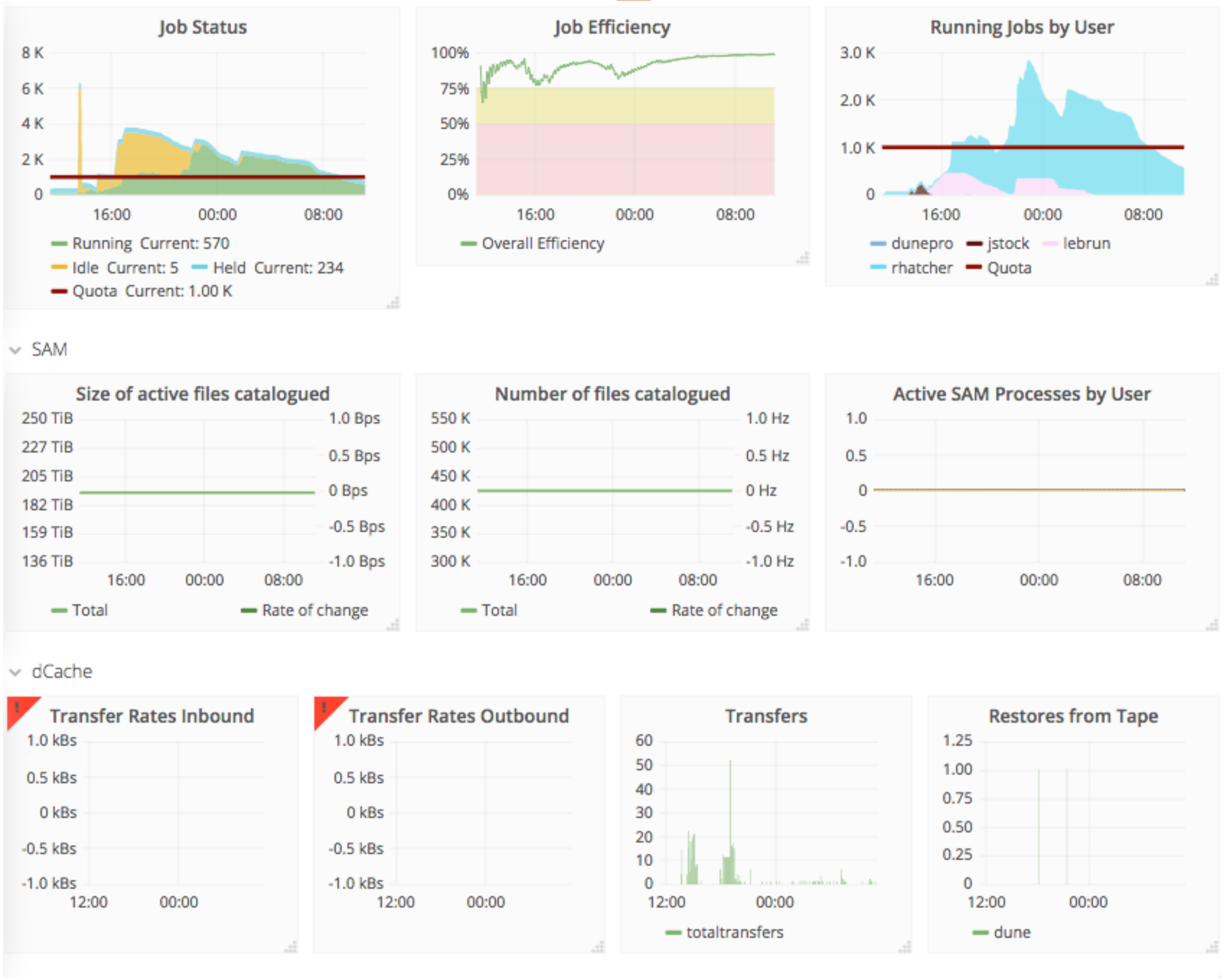
https://cdcvs.fnal.gov/redmine/projects/jobsub/wiki/Jobsub_submit

Accessing CVMFS

- LArSoft and dunetpc software versions are accessible at FNAL and remotely using CVMFS.
  - https://cdcvs.fnal.gov/redmine/projects/dune/wiki/Access_files_in_CVMFS
  - Don't miss the Quick Start guide at the bottom of the page
  - This tells you how to install CVMFS on your computer.
- You will need two repositories –
  - /cvmfs/dune.opensciencegrid.org. (to get DUNE software)
  - /cvmfs/fermilab.opensciencegrid.org (to get LArSoft and dependencies)
- Adding files to CVMFS – you will need permission from the DUNE S&C coordinators first.

FIFE Tools - Monitoring

- FIFEMON for DUNE – need services account to login
- Some of what is monitored ->



Best Practices – Common Complaints/Problems

1. Jobs are taking too long
2. Fewer jobs run simultaneously than expected
3. Jobs fail due to missing mount points
4. Jobs run longer than expected or get held after exceeding resource requests (memory, local disk, run time)

Best Practices -

Jobs are taking too long
Fewer jobs run simultaneously than expected

- **Most Likely Cause** – there are no available slots that match your request.

- Have you submitted to all available resources, including offsite?
  - There are potentially 1000's of cores beyond FermiGrid.
  - Make your scripts OSG-ready from the beginning.
  - Recommend NOT choosing specific sites.
  - **https://cdcvs.fnal.gov/redmine/projects/fife/wiki/Information_about_job_submission_to_OSG_sites**

Best Practices

Jobs are taking too long
Fewer jobs run simultaneously than expected

- Have you accurately specified your needed resources? The more you request, the harder it is to match.
  - Check memory, disk and run time
  - Request only what you need. Don't just stick with the default, you can gain by requesting **less** than the default.
  - Jobs are matched to slots based on the resource requests.
  - Requesting too little isn't good either.
    - Your job will be automatically **held** if it goes above the requested memory, local disk, or run time request.
    - Held = stopped and will restart from the beginning when manually released.
  - Memory/disk usage checks run every two minutes.

Best Practices – Requesting Resources

- Jobsub has --memory, --disk, --cpu, --expected-lifetime opts
- --memory and --disk will take units of KB, MB, GB, TB
  - 1 GB = 1024 MB, not 1000 MB
- Example of a well-formed request
  - --cpu=1 –memory=1800MB –disk=15GB –expected-lifetime=6h
- You might not be using jobsub directly, consult the documentation of whatever you are using to pass resource requests to jobsub_submit

Best Practices – Requesting Resources

	Accepted Units	Default Units	Default Request	Limit (FermiGrid)
--memory	KB, MB, GB, TB	MB	2000MB	16,000MB
--cpu	integer	integer	1	8
--disk	KB, MB, GB, TB	KB	35,000,000 KB	
--expected-lifetime	h (hours), m (minutes), s (seconds) Can also use - short (3h), medium (8h), long (24h)	s	8 hours max run time	4 days

Best Practices

Jobs fail due to missing mount points

- Grid nodes do not have /pnfs directly mounted.
- Non-FermiGrid nodes do not have any of the following mounted
  - /grid/fermiapp/anything (going away on FermiGrid in Jan)
  - /experiment/anything (going away on FermiGrid in Jan)
  - /pnfs/anything
  - Jobs fail due to missing mount points
- Write scripts without using these mounts.
  - Get experiment software from CVMFS and/or tarballs copied from dCache
  - Do not hardcode file paths (CVMFS is ok to hardcode)
- You can test to make sure you don't need these mounts by running your job script on fermicloud168.fnal.gov. Follow these instructions –
  - https://cdcvs.fnal.gov/redmine/projects/fife/wiki/Using_'grid-like'_nodes

Best Practices

Jobs run longer than expected

- Your job could be stuck copying inputs/outputs
  - Did you use a /grid/… or /dune/… (bluearc) disk?
  - Did you allow enough time for file transfer from remote sites?
  - If you are running with a SAM dataset, did you pre-stage it? This is important as files may have to be fetched from tape which can take a while.
  - Did you spell your filename correct on an 'ifdh cp' line?
    - Pass *–e IFDH_CP_MAXRETRIES=2* to jobsub
    - Use the *–timeout self-destruct* timer option to jobsub_submit so the job self-destructs before being held
  - Was your job held for some reason? Check here –
    - https://fifemon.fnal.gov/monitor/dashboard/db/why-are-my-jobs-held?orgId=1 (choose your username from the drop-down menu in the upper left corner)

Best Practices - A Common Denominator

- All of following problems –
  - Inability to use offsite resources
  - Slow run times and slow file transfer rates
  - Job failures due to missing mount points
  - Stuck in idle waiting for jobs to copy

Can be caused by using BlueArc areas in your job

It is OK to do a *jobsub_submit <options> file:///dune/app/foo*
But, /dune/app/foo should not use BlueArc inside of it.

!/bin/bash

Best Practices - A Common Denominator

- Get rid of BlueArc - Change from

```
#!/bin/bash
# setup SW
. /grid/fermiapp/products/dune/setup_dune.sh
setup some_packages
ifdh cp -D /pnfs/dune/scratch/users/${GRID_USER}/my_input_file ./
/dune/app/users/${GRID_USER}/my_custom_code/mycode -i my_input_file -o my_output_file
ifdh cp -D my_output_file /pnfs/dune/scratch/users/${GRID_USER}/some_dir/
```

Best Practices - A Common Denominator

- Change to

```
#!/bin/bash
# setup SW
. /cvmfs/dune.opensciencegrid.org/products/dune/setup_dune.sh
setup some_packages
ifdh cp -D /pnfs/dune/scratch/users/${GRID_USER}/my_input_file ./
ifdh cp -D /pnfs/dune/scratch/users/${GRID_USER}/my_custom_code.tar.gz ./
tar zmfx my_custom_code.tar.gz
./my_custom_code/mycode -i my_input_file -o my_output_file
ifdh cp -D my_output_file /pnfs/dune/scratch/users/${GRID_USER}/some_dir/
```

Still needs error checking, dependency checking, …

Best Practices

- Submit test jobs before any large submissions after changes
- Test interactively (gpgtest, fermicloud168)
- Consult with the DUNE S&C Coordinators before submitting large jobs
- If you have specific requirements the worker node must meet, put them in the job requirements via –
  - --*append_condor_requirements* jobsub option
  - These might include, veto of a specific site, specific version of CVMFS you need, …
- Prestage your input files
- If you need to pass environment variables to your script to be evaluated on the worker node, preface them with a '\' (variable will be expanded in the job, not during submission)
  - --source=\$CONDOR_DIR_INPUT/script.sh

Best Practices

- How many jobs can I submit at once?
  - Max in a single submission is **10K**
  - For multiple submissions, do not exceed 1000 jobs/minute
- How many jobs can I have queued?
  - No hard limit, but no more than you could run in a week using DUNE's FermiGrid quota
- Don't forget to account for slot weight! For 4 GB memory jobs, again divide by 2.

Getting Help

- https://cdcvs.fnal.gov/redmine/projects/dune/wiki/Getting_Started_with_DUNE_Computing
- https://cdcvs.fnal.gov/redmine/projects/dune/wiki/Computing_How-To_Documentation
- Use the DUNE Service Portal. (services username/password)
- Issues with job submission, data movement, trouble with SAM, …
  - Open a service desk ticket.

Get More Info On LArSoft

- These slides were taken from – DUNE S&C Tutorial from Aug
- Main public LARSoft web page : https://larsoft.org
- LArSoft wiki - https://cdcvs.fnal.gov/redmine/projects/larsoft/wiki
- Quick page with links to quick-start guides by experiment
- Forum to discuss LArTPC software - http://www.larforum.org/
- LArSoft email list - larsoft@fnal.gov
- Bi-weekly meeting 9am Central in WH3NE - LArSoft Indico site
- LArSoft issue tracker - https://cdcvs.fnal.gov/redmine/projects/larsoft/issues/new
- 2015 LArSoft course material
- Best practices for writing fcl files explained by Kyle Knoepfel
  - Not needed for typical user. Required for user who writes modules or production workflows.

Get More Info on Gallery

- These slides were taken from – DUNE S&C Tutorial from Aug
- Gallery main web page - http://art.fnal.gov/gallery/
- Gallery demo - https://github.com/marcpaterno/gallery-demo

Get More Info on Data Management

- These slides were taken from – DUNE S&C Tutorial from Aug
- Using DUNE's dCache - https://cdcvs.fnal.gov/redmine/projects/dune/wiki/Using_DUNE's_dCache_Scratch_and_Persistent_Space_at_Fermilab
- Understanding storage volumes https://cdcvs.fnal.gov/redmine/projects/fife/wiki/Understanding_storage_volumes
- SAM4Users wiki https://cdcvs.fnal.gov/redmine/projects/sam/wiki/SAMLite_Guide
- SAM wiki https://cdcvs.fnal.gov/redmine/projects/sam/wiki/User_Guide_for_SAM

Get More Info on FIFE Tools

- These slides were taken from – DUNE S&C Tutorial from Aug
- https://web.fnal.gov/project/FIFE/SitePages/Home.aspx
- https://cdcvs.fnal.gov/redmine/projects/jobsub/wiki/Jobsub_submit
- List of environment variables on worker nodes:
  - https://cdcvs.fnal.gov/redmine/projects/jobsub/wiki/Jobsub_enviornment_variables
- Full documentation of the jobsub client here
  - https://cdcvs.fnal.gov/redmine/projects/jobsub/wiki/Using_the_Client
- https://cdcvs.fnal.gov/redmine/projects/fife/wiki/Wiki
- https://cdcvs.fnal.gov/redmine/projects/fife/wiki/Introduction_to_FIFE_and_Component_Services
- https://cdcvs.fnal.gov/redmine/projects/fife/wiki/Advanced_Computing
- https://cdcvs.fnal.gov/redmine/projects/jobsub/wiki#Client-User-Guide
- https://cdcvs.fnal.gov/redmine/projects/ifdhc/wiki

Learn More About Running DAGs

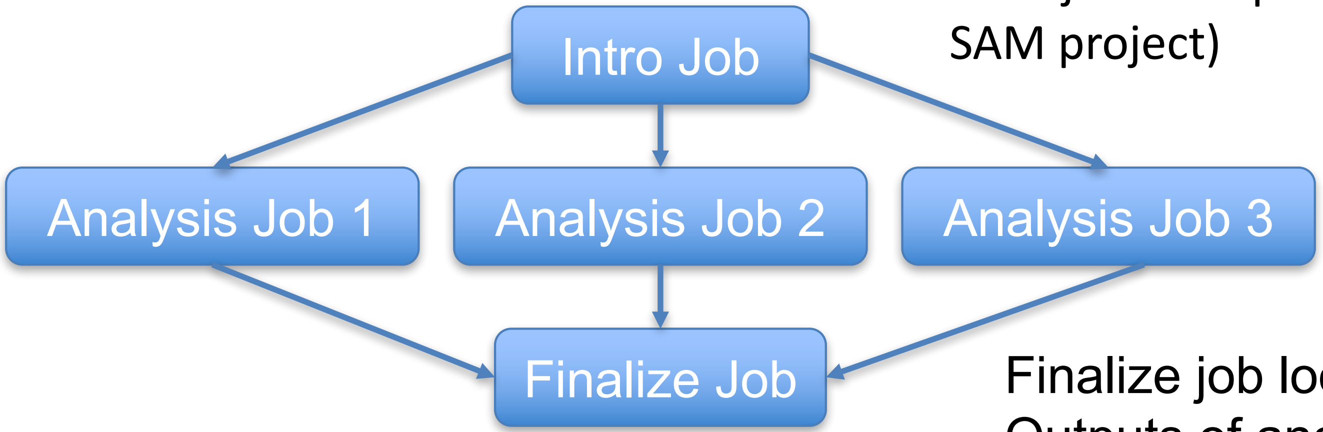
A bit on DAGs

- 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>

jobsub -n --memory=500MB --disk=1GB \ --expected-lifetime=1h \ --resource-
provides=usage_model=DEDICATED,OPPORTUNISTIC,OFFSITE file://init_script.sh ARGS0
```

Notes:
You can put jobsub and jobsub_submit, inside the xml. **You also need a -n** after jobsub.

You 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

```
</parallel>

<serial>

jobsub -n --memory=1000MB --disk=5GB \ --expected-lifetime=2h \ --resource-
provides=usage_model=DEDICATED,OPPORTUNISTIC,OFFSITE file://finalize_script.sh ARGS2

</serial>
```

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)

Learn More About GPUs

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
- 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' \ file:///home/s1/kherner/basicscript_GPU.sh**
  - -N 8 needed on the development server. N can be anything you want in production.

😀