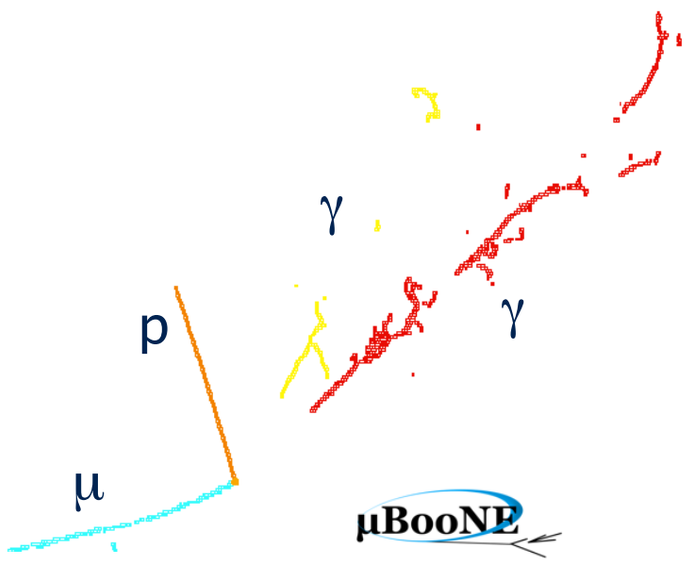
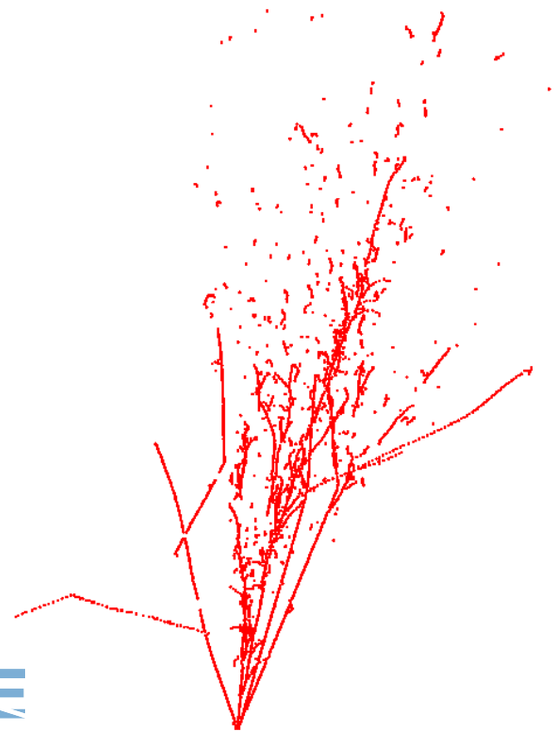


Pandora Exercise - I: Making samples or The Odyssey from GENIE to pndr

L. Escudero for the Pandora Team



Our route

Our Ithaca will be to create pndr files of big samples of events of the interaction types we are interested in (below) with the vertex in fiducial volume:

- ITHACA = PNDR
- Interactions:
  - CCQE
  - CCRESpi+
  - CCRESpi0
- In fiducial volume
- Versions
  - 4_36_00_03 (sim)
  - 5_08_00_05 (reco)
- Stages
  - GENIE
  - G4
  - DetSim

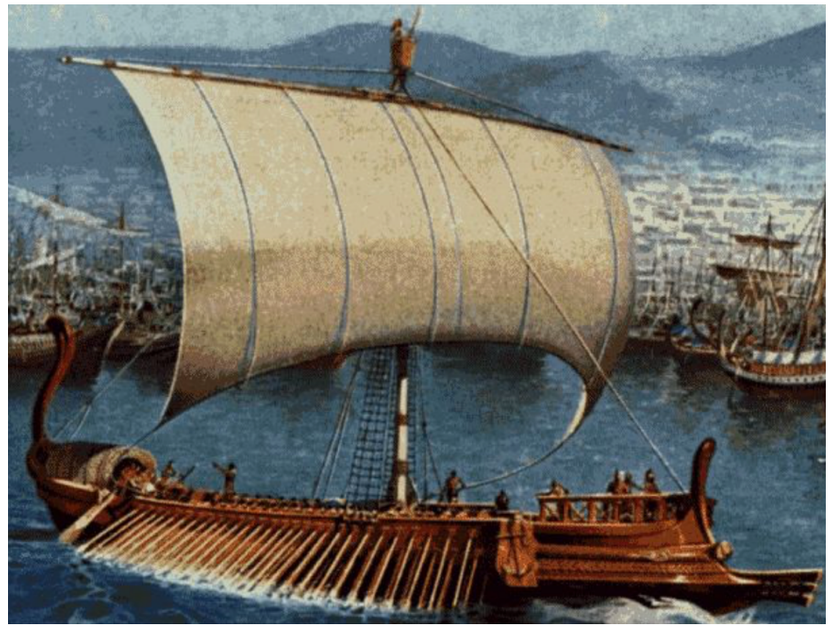
The ship

Our ship is going to be Fermigrid, our compass larbatch (uboonecode) project.py

To submit our jobs we will do something like: project.py --xml my_xml_file.xml --stage X —-submit

We will need to provide a .xml file like: (this is just the beginning, stages added in next slides)

```
<?xml version="1.0"?>
<!-- Production Project -->
<!DOCTYPE project [
<!ENTITY release "v04_36_00_03">
<!ENTITY file_type "mc">
<!ENTITY run_type "physics">
<!ENTITY name "my_prod_chain_res">
<!ENTITY user "lorena">
]>

<job>
<project name="&name;">
  <!-- Group -->
  <group>uboone</group>
  <!-- Project size -->
  <numevents>10</numevents>
  <maxfilesperjob>1</maxfilesperjob>
  <!-- Operating System -->
  <os>SL6</os>
  <!-- Batch resources -->
  <resource>DEDICATED,OPPORTUNISTIC</resource>
  <!-- Larsoft information -->
  <larsoft>
    <tag>&release;</tag>
    <qual>e9:prof</qual>
    <local>/uboone/app/users/lorena/v4_36_00_03/local.tar</local>
  </larsoft>
```



We will be using a local version of the code to use the filter (next pages)

```
tar -C $MRB_INSTALL -czf local.tar .
```

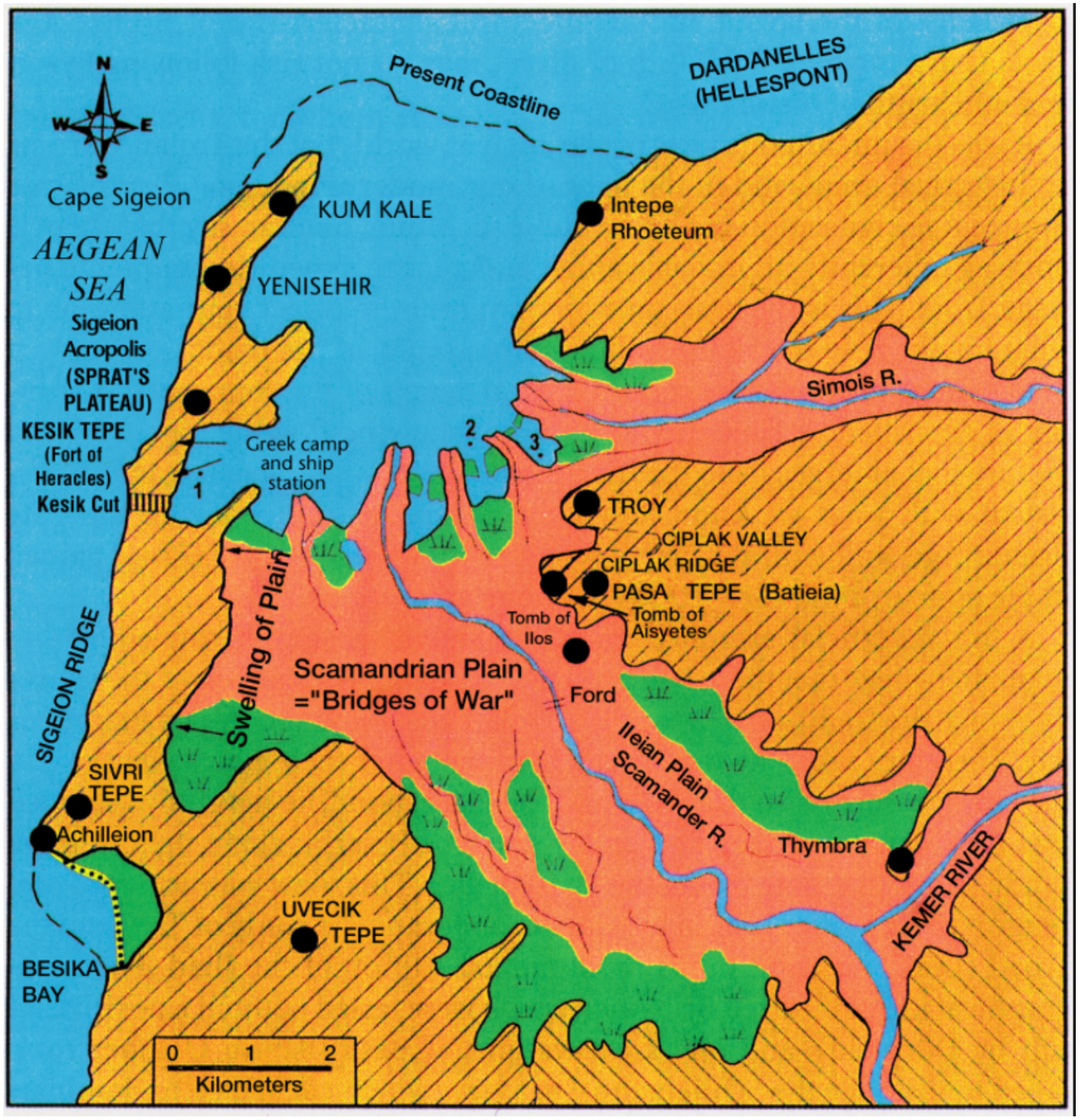
Example: /pnfs/uboone/scratch/users/lorena/prod_chain_res.xml

Before embarking

1. **Make sure you have a FNAL account!**
2. **Join uboone VO!**
3. **Install a clean "mrb newDev" of LArSoft versions we need:**
   - **v04_36_00_03 (for GENIE, G4 and DetSim)**
   - **v05_08_00_05 (for SignalProcessing and Pandora)**
4. **Get and edit /uboone/app/users/lorena/setup_gric**
5. **Make yourself a directory in**
   - **/uboone/app/users (for local LArSoft versions)**
   - **/pnfs/uboone/scratch/users (for grid outputs)**

And… breath…

Step1: Create a my_prodgenie_res.fcl (and similarly for ccqe):



Troy… lets start the war!

GENIE

Brief instructions: by giving an EventGeneratorList we can do a first selection of the kind of interaction: CCQE, CCRES, etc. (Note that this selection is broader than the samples we want, e.g. CCRES will include for pi+ and pi0 samples)

Step1: Create a my_prodgenie_res.fcl (and similarly for ccqe):

```
#include "prodgenie_bnb_nu_uboone.fcl"

physics.producers.generator.EventGeneratorList: “CCRES"
```

simple, uh?

```
physics.producers.generator.FluxSearchPaths:   "/pnfs/uboone/persistent/
uboonebeam/bnb_gsimple/bnb_gsimple_fluxes_02.28.2014_470/"
physics.producers.generator.FluxFiles:          ["gsimple_microboone-470-
onaxis_mc_nu_dummy_ntrd_*.root"]
physics.producers.generator.FluxCopyMethod: "IFDH"
```

Be careful out there, in the Fermigrid-ocean you might need to specify the flux paths and the method to copy them (IFDH = Intensity Frontier Data Handling, see https://web.fnal.gov/project/FIFE/notes/Pages/ifdh.aspx) if you are not using the latest versions LArSoft v6

Example: /uboone/app/users/lorena/v4_36_00_03/my_prodgenie_res.fcl

GENIE

We need to pay attention when submitting many GENIE jobs with a small number of events each one to the transfer of flux files in order to not to overload dCache

Flux files are 4 MB each, and the default setting is 2 GB (MaxFluxFileMB, limit of the collective size of files passed to flux driver), then you get 500 files transferred, which is an excessive amount of data transferred for jobs generating few (10ish) events This can be changed through the parameter MaxFluxFileMV (Mbytes) example:

MaxFluxFileMB limit the collective size of files passed to flux driver (Mbytes)

physics.producers.generator.EventGeneratorList: "CCQE"
physics.producers.generator.FluxSearchPaths: "/pnfs/uboone/persistent/uboonebeam/bnb_gsimple/bnb_gsimple_fluxes_02.28.2014_470/"
physics.producers.generator.FluxFiles: ["gsimple_microboone-470-onaxis_mc_nu_dummy_ntrd_*.root"]
physics.producers.generator.FluxCopyMethod: "IFDH"
physics.producers.generator.MaxFluxFileMB: 100

For reference, for production (like MCC7) flux size was limited to 500 MB for 50 events. Numbers of this order should be safe for randomness of events

How it looks like inside the .xml file: GENERATOR STAGE

```
<!-- Project stages -->

<stage name="gen">
  <fcl>/uboone/app/users/lorena/v4_36_00_03/my_prodgenie_res.fcl</fcl>
  <outdir>/pnfs/uboone/scratch/users/lorena/&name;</outdir>
  <logdir>/pnfs/uboone/scratch/users/lorena/&name;</logdir>
  <workdir>/pnfs/uboone/scratch/users/lorena/work/&name;</workdir>
  <numjobs>1</numjobs>
  <datatier>generated</datatier>
  <defname>&name;_gen</defname>
</stage>
```

The one you just created

Edit as needed to match the total number of events you want (specified in a command before)
For example, submit 1000000 events divided into 1k jobs

Not advisable to run over big number of events (judge yourself, but don't use a job with 100k events) - because GENIE might give a negative oscillation probability at some point with our flux files and the job will fail, so expect some jobs to fail

Example: /pnfs/uboone/scratch/users/lorena/prod_chain_res.xml

Lets submit this stage to the Fermigrid machines!

```
<uboonegpvm04.fnal.gov> project.py --xml prod_chain_res.xml --stage gen --submit
```

Indeed, nothing should happen after this command

```
<uboonegpvm04.fnal.gov> project.py --xml prod_chain_res.xml --stage gen --status

Project my_prod_chain_res:

Stage gen: 0 art files, 0 events, 0 analysis files, 0 errors, 1 missing files.
Stage gen batch jobs: 1 idle, 0 running, 0 held, 0 other.

Stage g4 output directory does not exist.
Stage g4 batch jobs: 0 idle, 0 running, 0 held, 0 other.

Stage detsim output directory does not exist.
Stage detsim batch jobs: 0 idle, 0 running, 0 held, 0 other.
```

You can check the status of your jobs at any time

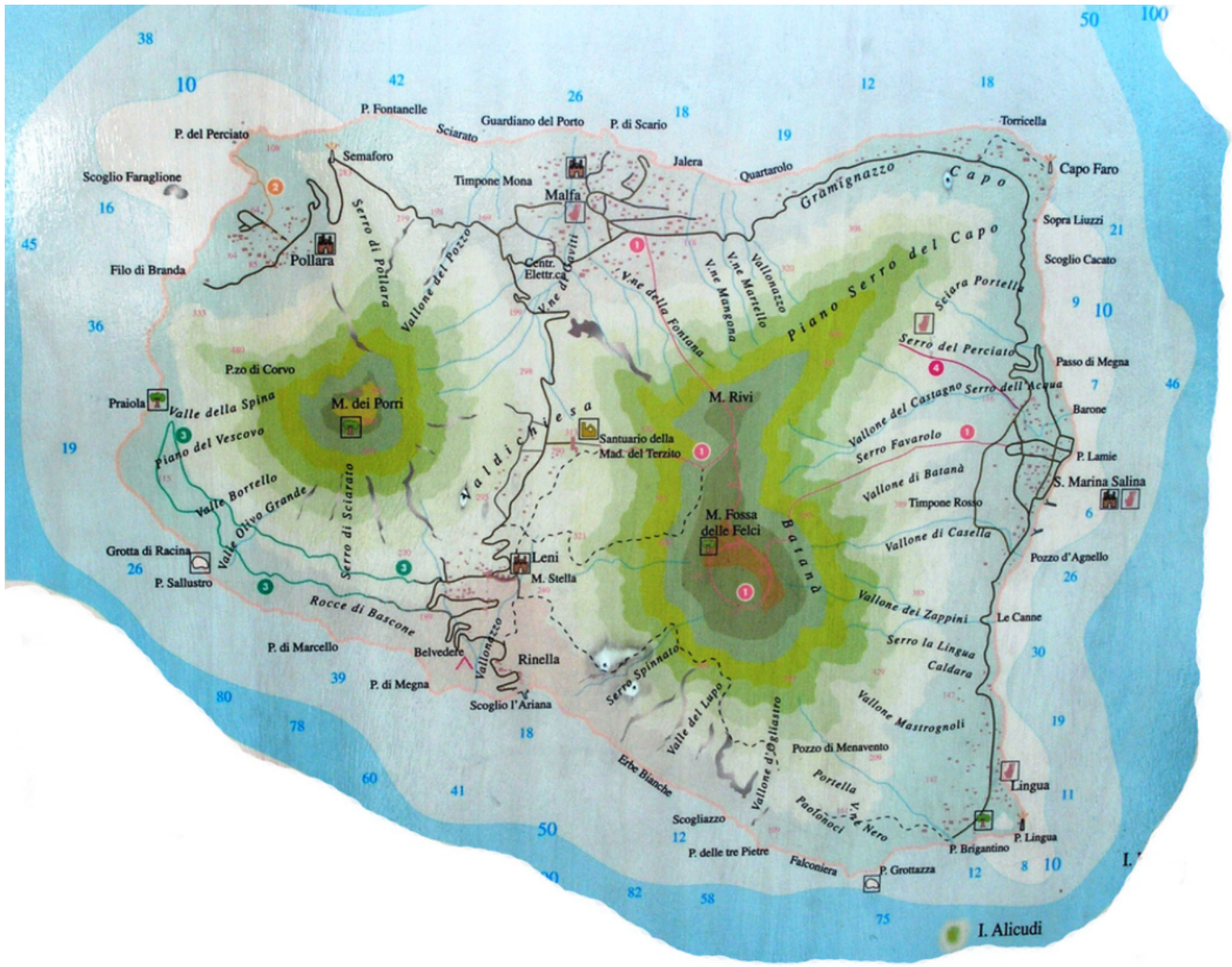
And you need to "check" to get the output files before moving to the next stage

```
<uboonegpvm04.fnal.gov> project.py --xml prod_chain_res.xml --stage gen --check

Checking directory /pnfs/uboone/scratch/users/lorena/my_prod_chain_res
Checking root files in directory /pnfs/uboone/scratch/users/lorena/my_prod_chain_res/11785124_0.
10 total good events.
1 total good root files.
1 total good histogram files.
Adding layer two for path /pnfs/uboone/scratch/users/lorena/my_prod_chain_res/bad.list.
Adding layer two for path /pnfs/uboone/scratch/users/lorena/my_prod_chain_res/missing_files.list.
Adding layer two for path /pnfs/uboone/scratch/users/lorena/my_prod_chain_res/transferred_uris.list.
Adding layer two for path /pnfs/uboone/scratch/users/lorena/my_prod_chain_res/checked.
0 processes with errors.
0 missing files.
```

Note: There is an option to re-submit the jobs failing (--makeup) but we need to be careful with random seeds (to discuss)

G4 and Filter

Step2+3: Filter integrated in G4



Aeolian islands, twin mountains

Filter

Step2: Filter (integrated with Step 3)

As it is at the moment, the filter is very simple: just selects events based on the true interaction code and asking the neutrino vertex to be within the fiducial volume defined. The option to filter depending on number of particles created is implemented but not used: some of the created particles won't leave the nucleus, so could be done in G4 (TODO?)

Instructions to get/build it

cd $MRB_SOURCE
mrb g https://github.com/loressa/myfiltermodule.git
cd myfiltermodule
git checkout v5 (see note below)
mrb uc
cd $MRB_BUILDDIR
mrbsentenv
mrb i
mrbslp (remember to do this even if it is already compiled anytime you want to use the filter, otherwise the library won't be loaded and the filter won't be recognised)

Instructions to run it (independently)

lar -c srcs/myfiltermodule/myfiltermodule/run_myfilter.fcl input_file.root

Note: there are two branches, *master* was originally created to run with LArSoft v6, and branch *v5* is updated to work with LArSoft v5, works as well with v4

Filter

Inside myfilter.fcl, select options:

```
BEGIN_PROLOG

myfilter:
{
    module_type:                    "MyFilter"

    InteractionType: 1001
    DetectorHalfLengthX: 256.35
    DetectorHalfLengthY: 233.
    DetectorHalfLengthZ: 1036.8
    CoordinateOffsetX: 0.
    CoordinateOffsetY: 116.5
    CoordinateOffsetZ: 0.
    SelectedBorderX: 10.
    SelectedBorderY: 20.
    SelectedBorderZ: 10.
}

END_PROLOG
```

Interaction type - Remember, remember…

1001 for CCQE

1003 for CCRES pi+

1004 for CCRES pi0

Fiducial volume definition

G4 and Filter

Step3: Run g4 on the output files from GENIE, filtering the events

Get a local copy of the standard .fcl and edit to introduce the filter

```
cp $UBOONECODE_DIR/job/standard_g4_uboone.fcl my_standard_g4_uboone.fcl
```

```
#include "services_microboone.fcl"
#include "largeantmodules_microboone.fcl"
#include "mcreco.fcl"
#include "time_memory_tracker_microboone.fcl"
#include "myfilter.fcl"
process_name: G4

…

source:
{
  module_type: RootInput
  maxEvents:  -1        # Number of events to create
  saveMemoryObjectThreshold: 0
}

…

physics:
{

filters:
 {
        myfilter: @local::myfilter
 }


…
simulate: [  myfilter, rns, largeant, mcreco ]
```

This was important when submitting g4+detsim after done the filtering independently, it is the way to keep using myproject.py without it complaining that the number of events in the project doesn't match the number of events g4 is going to find in the file, and to run over all the filtered events in the file without knowing how many they are

Example: /uboone/app/users/lorena/v4_36_00_03/my_standard_g4_uboone.fcl

G4 and Filter

How it looks like inside the .xml file: G4 STAGE

```
<stage name="g4">
  <fcl>/uboone/app/users/lorena/v5/fcl/my_standard_g4_uboone.fcl</fcl>
  <outdir>/pnfs/uboone/scratch/users/lorena/&name;/g4</outdir>
  <logdir>/pnfs/uboone/scratch/users/lorena/&name;/g4</logdir>
  <workdir>/pnfs/uboone/scratch/users/lorena/work/&name;/g4</workdir>
  <numjobs>1</numjobs>
  <datatier>simulated</datatier>
  <defname>&name;_g4</defname>
</stage>
```

Don't forget to edit and have the local version we will use including the filter

Example: /pnfs/uboone/scratch/users/lorena/prod_chain_res.xml

G4 and Filter

```
<uboonegpvm04.fnal.gov> project.py --xml prod_chain_res.xml --stage g4 --submit
```

We need to submit, same project, next stage

```
<uboonegpvm04.fnal.gov> project.py --xml prod_chain_res.xml --stage g4 --check

Checking directory /pnfs/uboone/scratch/users/lorena/my_prod_chain_res/g4
Checking root files in directory /pnfs/uboone/scratch/users/lorena/my_prod_chain_res/g4/11787136_0.
10 total good events.
1 total good root files.
1 total good histogram files.
Adding layer two for path /pnfs/uboone/scratch/users/lorena/my_prod_chain_res/g4/bad.list.
Adding layer two for path /pnfs/uboone/scratch/users/lorena/my_prod_chain_res/g4/missing_files.list.
Adding layer two for path /pnfs/uboone/scratch/users/lorena/my_prod_chain_res/g4/checked.
0 processes with errors.
0 missing files.
```

Don't panic… project.py does not need to understand the logic of what you are doing inside - lets have a look at the out file (below)

```
emacs -nw /pnfs/uboone/scratch/users/lorena/my_prod_chain_res/g4/11787136_0/lar.out
```

Print out to check the filter is running

Events passing the filtering

Events we checked

```
Begin processing the 10th record. run: 1 subRun: 1 event: 10 at 14-Nov-2016 11:45:41 CST
Interaction: 1003 and I want: 1004
14-Nov-2016 11:45:41 CST  Closed file
prodgenie_bnb_nu_uboone_20161114T165042_gen_aaa8ec7e-7d11-4b71-b21e-eaa98f4a622c.root

TrigReport ---------- Event  Summary ------------
TrigReport Events total = 10 passed = 1 failed = 9

TrigReport ------ Modules in End-Path: end_path ------------
TrigReport  Trig Bit#    Visited     Passed     Failed      Error Name
TrigReport     0    0         10         10          0          0 out1
```

DetSim

Step4: detsim

Ogygia (Calypso's home)… where most of the time is spent

DetSim

Step4: detsim

If everything was followed correctly up to this point, there is nothing special to do for the detsim stage, we will run the standard detsim and submit jobs as usual

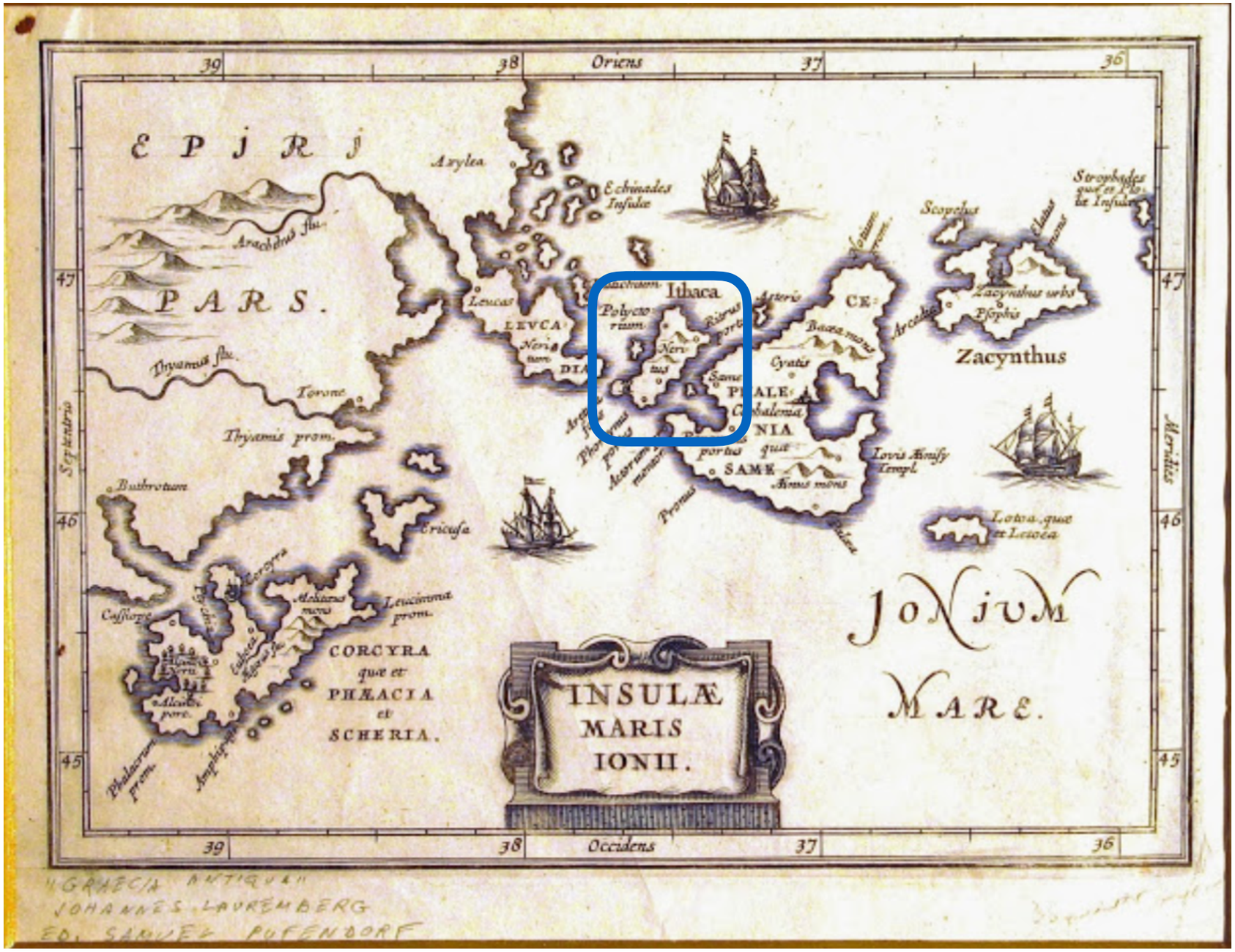
```
<stage name="detsim">
  <fcl>standard_detsim_uboone.fcl</fcl>
  <outdir>/pnfs/uboone/scratch/users/lorena/&name;/detsim</outdir>
  <logdir>/pnfs/uboone/scratch/users/lorena/&name;/detsim</logdir>
  <workdir>/pnfs/uboone/scratch/users/lorena/work/&name;/detsim</workdir>
  <numjobs>1</numjobs>
  <datatier>detector-simulated</datatier>
  <defname>&name;_detsim</defname>
</stage>
```

```
project.py --xml prod_chain_res.xml --stage detsim --submit
```

Example: /pnfs/uboone/scratch/users/lorena/prod_chain_res.xml

Pndr

Step5: create pndr files



Finally… Ithaca!

Step5: create pndr files

Remember! Now we need to work with v05_08_00_05 of LArSoft!
Following Exercise 1 in the Pandora Workshop…

cp $UBOONECODE_DIR/job/reco_uboone_mcc7_driver_stage2.fcl ./my_pandora_writer.fcl

```
#include "reco_uboone_mcc7_driver_common.fcl"

process_name: McRecoAprStage2

services.DetectorClocksService.InheritClockConfig:  false

physics.producers.pandoraWriter: @local::microboone_pandorawriter
physics.producers.pandoraWriter.HitFinderModuleLabel: "gaushit"
physics.producers.pandoraWriter.ConfigFile: "MyPandoraSettings_Write.xml"

services.TFileService.fileName: "reco_stage_2_hist.root"
physics.reco: [ @sequence::microboone_reco_mcc7_signalprocessing, pandoraWriter ]
physics.trigger_paths: [ reco ]
physics.end_paths:[]
#outputs.out1.fileName: "%ifb_%tc_reco2.root"
#outputs.out1.dataTier: "reconstructed"
source.inputCommands: ["keep *_*_*_*", "drop *_*_*_McRecoStage2" ]
```

Also: cp $LARPANDORA_DIR/scripts/PandoraSettings_Write.xml ./MyPandoraSettings_Write.xml
(edit name of output file if you want)

Example: /uboone/app/users/lorena/v5_08_00_05/my_pandora_writer.fcl

Step5: create pndr files

For a question of speed, we are going to run one my_pandora_writer.fcl job per detsim root file, and we will submit them to Fermigrid (directly, without project.py), using script files like:

```
#!/usr/bin/env bash
source /grid/fermiapp/products/uboone/setup_uboone.sh
setup uboonecode v05_08_00_05 -q e9:prof
export USRDIR=/uboone/app/users/lorena/v5_08_00_05
export INDIR=/pnfs/uboone/scratch/users/lorena/bnb_1004_v04_36_00_03/detsim/
14790601_99
export OUTDIR=/pnfs/uboone/scratch/users/lorena/PndrFiles/1004/
ifdh cp ${INDIR}/filtered_events_1004_*.root my_file.root
ifdh cp ${USRDIR}/my_pandora_writer.fcl .
ifdh cp ${USRDIR}/MyPandoraSettings_Write.xml .
lar -c my_pandora_writer.fcl -n 2000 -s my_file.root
ifdh cp Pandora_Events.pndr ${OUTDIR}/Pandora_Events_1004_1.pndr
```

Path to find my input file

Our job

We need to give copy/move commands using IFDH (remember we have seen this before?)

Where I will find the jobs

Example: /uboone/app/users/lorena/v5_08_00_05/job_grid_pndr_1.sh

Step5: create pndr files

To create a .sh file for each output file and submit it to Fermigrid, I created a bash script /uboone/app/users/lorena/v5_08_00_05/jobs_grid_submitter.sh
(python, bash experts welcome to improve this bit!)

```
ls /pnfs/uboone/scratch/users/lorena/bnb_1004_v04_36_00_03/detsim >
filtered_detsim_1004.txt

./jobs_grid_submitter.sh filtered_detsim_1004.txt
```

Grid commands:

```
jobsub_q --user escudero --group uboone
```
To check what the jobs are doing

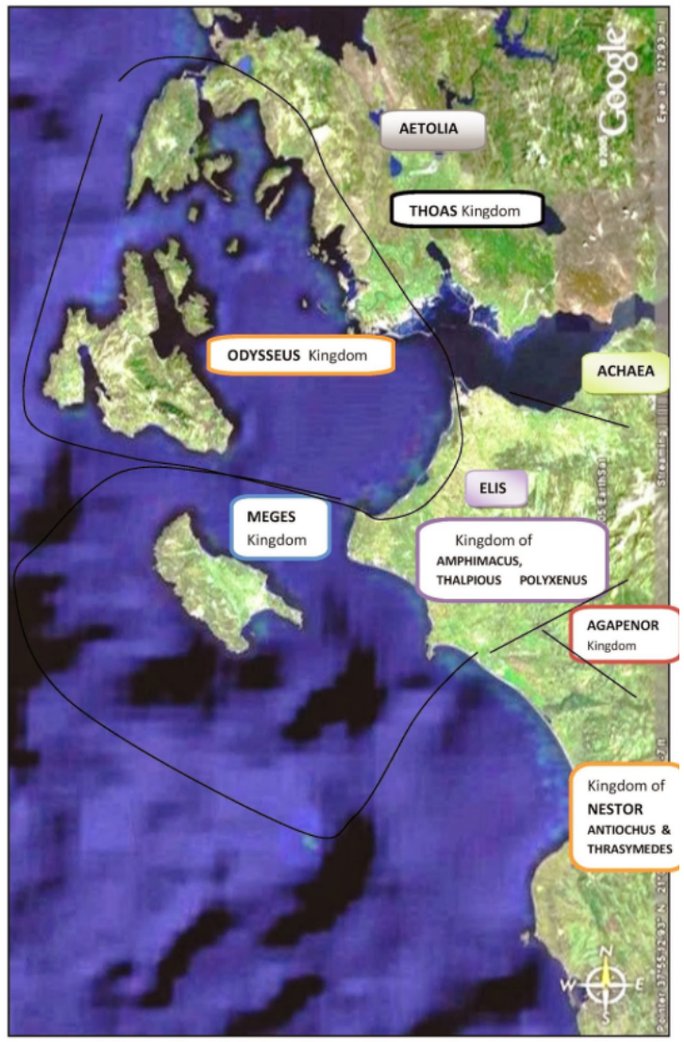
```
jobsub_fetchlog --jobid=11795247.0@fifebatch1.fnal.gov
```
To retrieve log files

http://fifemon1.fnal.gov/monitor/user/<username> Very useful (and visual!) to monitor your jobs

Alternative routes

During this Odyssey, I created other ways to run my jobs in Fermigrid without using project.py. If you want to explore alternative routes, you can submit your jobs with a bash script like:

```
#!/usr/bin/env bash

#source /cvmfs/uboone.opensciencegrid.org/products/
setup_uboone.sh
source /grid/fermiapp/products/uboone/setup_uboone.sh
setup uboonecode v04_36_00_03 -q e9:prof
#setup ifdhc

export USRDIR=/uboone/app/users/lorena/v4_36_00_03
export OUTDIR=/pnfs/uboone/scratch/users/lorena/jobsub/

ifdh cp ${USRDIR}/filtered_events_res.root
filtered_events_res.root
#lar -c eventdump.fcl filtered_events_res.root -n 100

lar -c standard_g4_uboone.fcl filtered_events_res.root -n
100

ifdh cp filtered_events_res*g4.root ${OUTDIR}/
rm filtered_events_res*g4.root
```



Example: /uboone/app/users/lorena/v4_36_00_03/test_job_grid.sh

Further Documentation

Here you can find the different options you can select from (CCQE, CCRES, CCCOH, etc):
https://cdcvs.fnal.gov/redmine/projects/nutools/wiki/GENIEHelper
And a translation to the nuance codes that we are used to see comes from here: https://userweb.jlab.org/~saw/MINERVA/nuance_defaults.cards.
(e.g. if you search for “Description of resonance reactions” you will find the different IDs for the different interaction types, like 3 for charged pions, 4 for pi0…).
To the nuance codes, GENIE adds 100, so nuance code for CCQE is 1, in GENIE appears as 1001, etc.

https://cdcvs.fnal.gov/redmine/projects/lardbt/wiki/Running_Grid_Jobs_using_projectpy/12
https://cdcvs.fnal.gov/redmine/projects/fife/wiki/Getting_Started_on_GPCF
https://cdcvs.fnal.gov/redmine/projects/larbatch/wiki/User_guide
https://cdcvs.fnal.gov/redmine/projects/uboonecode/wiki/Gridtips
http://microboone-docdb.fnal.gov/cgi-bin/RetrieveFile?docid=4020&filename=IIT_20150106.pdf&version=1
https://indico.fnal.gov/getFile.py/access?contribId=32&resId=0&materialId=slides&confId=8406
https://en.wikipedia.org/wiki/Geography_of_the_Odyssey