Summer School at Fermilab

Final Report

Event reconstruction in ProtoDUNE

Roberto Schiattarella

supervised by
Kenneth Herner

October 7, 2019

Contents

DUNE is probably the most important experiment about neutrino physic never thought. It will start taking data in 2025.
Now there is a prototype experiment at CERN, in Switzerland, called protoDUNE, that is already taking data and it is testing the technologies will use by DUNE when it will ready.
During my time at FermiLab, I collaborated with DUNE production group. For now, its goals are the production of data for protoDUNE and of montecarlo simulations for DUNE and protoDUNE experiment.
I worked on several aspects: the POMS platform improvement (the interface to use computer grid in a very easy way), trying to implement an autorelease code for held jobs on the grid; the submission of a montecarlo simulation for Supernova samples of elastic scattering events; the reprocessing of protoDUNE single phase data; and the validation of data coming from that reprocessing campaign.
This document is a brief recap of what I did during my time with the production group.

1 DUNE experiment

The Deep Underground Neutrino Experiment (DUNE) is a leading-edge, international experiment for neutrino science and proton decay studies. Discoveries over the past half-century have put neutrinos, the most abundant matter particles in the universe, in the spotlight for further research into several fundamental questions about the nature of matter and the evolution of the universe — questions that DUNE will seek to answer.
DUNE will consist of two neutrino detectors placed in the world's most intense neutrino beam. One detector will record particle interactions near the source of the beam, at the Fermi National Accelerator Laboratory, to measure the unoscillated neutrino spectrum and flux. A second, much larger, detector will be installed more than a kilometer underground at the Sanford Underground Research Laboratory in Lead, South Dakota — 1,300 kilometers downstream of the source. It will try to measure oscillations, SN burst neutrinos,nucleon decay, atmospheric neutrinos.

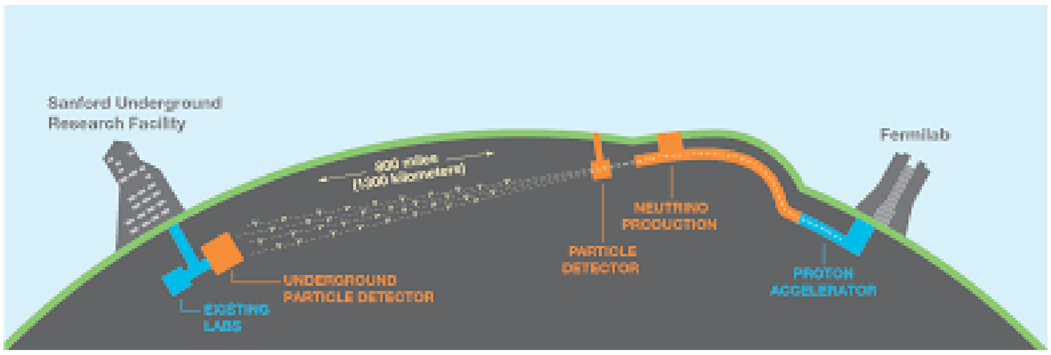

Sanford Underground Research Facility
800 miles (1300 kilometers)
Fermilab
NEUTRINO PRODUCTION
PARTICLE DETECTOR
PROTON ACCELERATOR
UNDERGROUND PARTICLE DETECTOR
EXISTING LABS

Figure 1.1: Dune Experiment project.

The detectors in South Dakota will be 4 x 10 kton LArTPC modules (single and dual-phase LArTPC).
The difference between single and dual-phase modules is that in the first one the detectors are fully submerged in liquid argon, and so when a ionising particle creates ionisation electrons they are directly collected by the anodic wires grid. Instead for dual-phase modules the charge extracted from liquid argon goes into an Argon gas phase, where the charge is amplified before the collection. As show from figure 1.2 there is a Large Electron Multiplier that is able to amplify signal.

DUNE Experiment shouldn't be able to take data before 2025. [1]

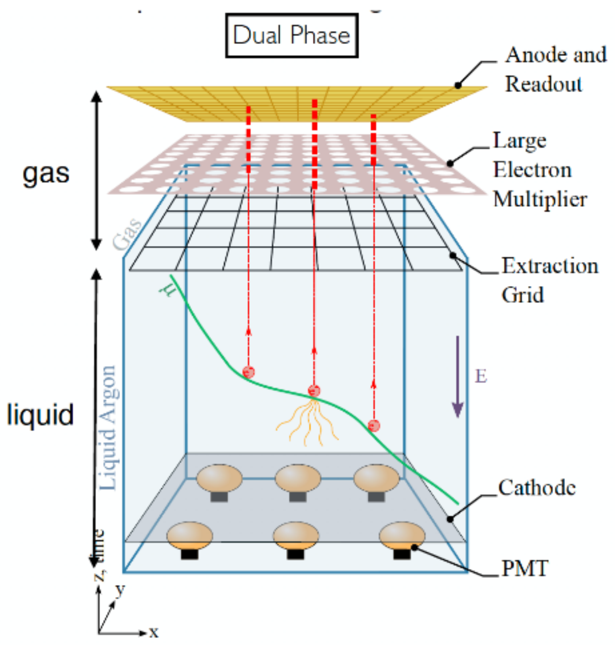


Figure 1.2: Dual Phase LArTPC module

1.0.1 protoDUNE detectors at CERN

To validate the LarTPC technologies, to demonstrate long-term performance and stability and to characterize detector response with particle energies in the region of interest for DUNE (500 Mev - 7 Gev), at CERN were built two detectors called protoDUNE single-phase and protoDUNE dual-phase.

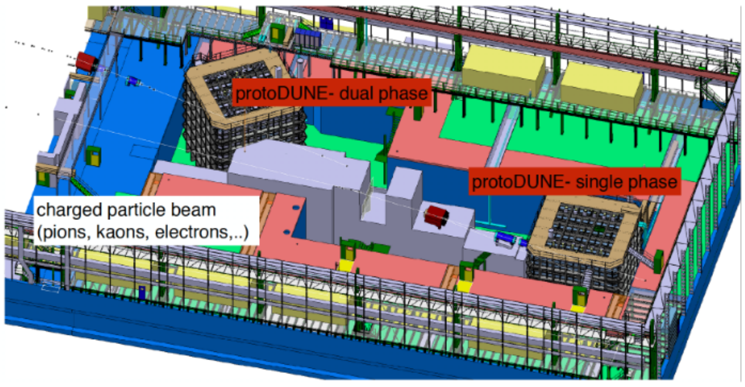


Figure 1.3: protoDUNE experiment

Each protoDUNE contains 800t of LAr. They are biggest ever to date! The dual-phase protoDUNE is still being perfected, while we have already data from the single-phase one. In particular last year some data were taken with particle beams, and at the end of this year other should come from cosmic rays.

2 Production Group

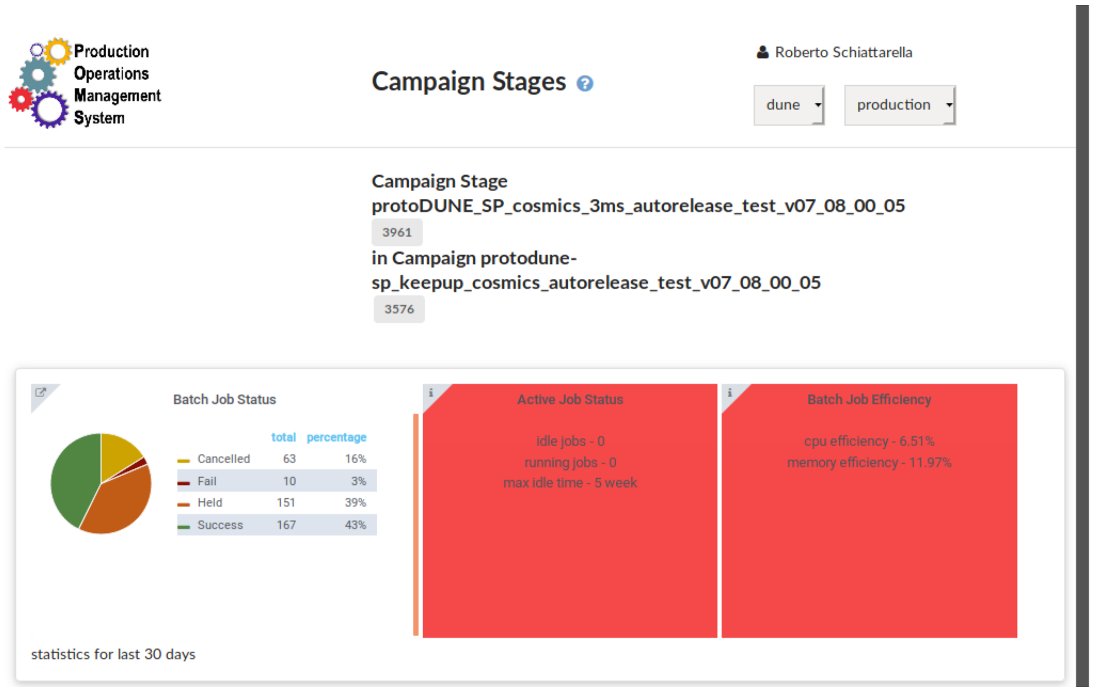
2.0.1 The computing problem

An experiment of such importance is able to produce a very big amount of data. The forecasts are that DUNE will write 10-30 PB/year from the far detectors and probably more than that from the near detector. So data management and analysis is a crucial problem for the DUNE experiment.
Even now the challenge of protoDUNE data management is very interesting: protoDUNE can write up to 2-3 GB/sec of data when running at 25 Hz beam. For now we have something like 3.2 PB of raw data from protoDUNE Single-Phase, and more will come from the future protoDUNE DualPhase measure and also to these numbers must be added all the data that come from MonteCarlo simulation.
For these reasons the production group rule in the DUNE (and protoDUNE) experiment is very important. Basically the production group main goals are processing raw data to obtain physical information that can be easily analysable in a second moment, and also to produce several MonteCarlo simulation that can be useful for the analysis groups.
Last year they processed all the Single-Phase protoDUNE raw data, and during my time at Fermilab working with the production group, we worked on the reprocessing of these data, but we will discuss that soon.

2.0.2 POMS

As the quantity of data originated by the running experiments greatly increases, the ability of simplifying the steps in data processing and management has become more and more appealing to the users. So the production group is usual to use a platform that is an interface between the users and the computer grid in which jobs are submitted. This platform is called POMS (*Production Operations Management System*).
It is a web service interface, enabling automated jobs submission on distributed resources according to customers' requests and subsequent monitoring and recovery of failed submissions, debugging and record keeping. Its ultimate goal is the most efficient utilization of all computing resources available to experiments.
The strength of POMS is the fact that it is very intuitive to use.
From the web service interface we can submit grid jobs and track these submissions through Landscape. We can also organize these into Campaigns with stages for which POMS gives a graphical editor to check the different status. POMS assists in analyzing jobs failures with plots, charts and easy access to

log files.[2]



Campaign Stages

Campaign Stage
protoDUNE_SP_cosmics_3ms_autorelease_test_v07_08_00_05
3961
in Campaign protodune-sp_keepup_cosmics_autorelease_test_v07_08_00_05
3576

Batch Job Status

	total	percentage
Cancelled	63	16%
Fail	10	3%
Held	151	39%
Success	167	43%

statistics for last 30 days

Active Job Status

idle jobs - 0
running jobs - 0
max idle time - 5 week

Batch Job Efficiency

cpu efficiency - 6.51%
memory efficiency - 11.97%

Figure 2.1: POMS Landscape

Figure 2.1 shows a typical campaign landscape page. As you can see is very easy to check job status: There is a cake graph that shows us the Batch job status. Basically the status of a job on the computer grid can be of five different kinds: we can have a Idle job (when a job is waiting to be processed), a Running job (when the job is still in a processing phase), a Removed or Completed job, and also we can have a Held job, (we will talk in detail about this kind of status in the next sections).

2.0.3 Held Jobs on the computer grid: Autorelease

Basically all the useful informations for a job submission into the computer grid by POMS, are given in the *configuration file*. In those files there are a serious of bash command useful for jobs submission. POMS offers also a GUI editor menu, from which is possible to change all the parameters present in the configuration file, so as to allow use same configuration files for different campaigns, only changing parameters on the POMS GUI Editor web menu.
This is very useful thing, because write a configuration file from zero, is not the most easy thing to do.
In particular in the configuration file there is written the amount of memory that we think a job will need on the computer grid, and also the amount of run time that we expect the job will take on the same grid. These two parameters are fundamental. In fact if one of these is exceeded the job goes to be held, and we have to relaunch jobs with bigger values of memory or run time to get

some results.
So the releases for held jobs were commands to do manually, but it is easy to understand that for campaigns with an huge amount of jobs this fact is a very big waste of time.
For these reasons in the first weeks at Fermilab, my job was to find a way to put some code lines in the configuration file that were able to autorelease jobs if those were held for exceeding the declared run time or memory values.
These are the lines that I implemented:

```
#autorelease for memory
lines_1 = +DUNE_OriginalMemory=10

lines_2 = +DUNE_GraceMemory=3000

lines_3 = +DUNE_IncreaseReqMem=(NumJobStarts>0)&&(!isUndefined
(LastHoldReasonCode))&&((LastHoldReasonCode=?=26&&
LastHoldReasonSubCode=?=1)||(LastHoldReasonCode=?=34))

lines_4 = +DUNE_ShouldRelease=(JobStatus=?=5)&&
((HoldReasonCode=?=26&&HoldReasonSubCode=?=1)||
(HoldReasonCode=?=34))&&(!isUndefined(MachineAttrMemory0))&&
(MemoryUsage<(DUNE_OriginalMemory+DUNE_GraceMemory))

lines_5 = request_memory=ifthenelse(DUNE_IncreaseReqMem,
DUNE_OriginalMemory+DUNE_GraceMemory,DUNE_OriginalMemory)

lines_6 = periodic_release=DUNE_ShouldRelease

lines_7 = job_machine_attrs=Memory


#autorelease for run time
lines_1 = +DUNE_OriginalRunTime=300

lines_2 = +DUNE_GraceRunTime=18000

lines_3 = +DUNE_IncreaseJobLifeTime=(NumJobStarts>0)&&
(!isUndefined(LastHoldReasonCode))&&
(LastHoldReasonCode=?=26&&LastHoldReasonSubCode=?=8)

lines_4 = +DUNE_ShouldRelease=(JobStatus=?=5)&&
(HoldReasonCode=?=26&&HoldReasonSubCode=?=8)&&
((EnteredCurrentStatus-JobStartDate)<
(DUNE_OriginalRunTime+DUNE_GraceRunTime))

lines_5 = +JOB_EXPECTED_MAX_LIFETIME=ifthenelse
(DUNE_IncreaseJobLifeTime,
DUNE_OriginalRunTime+DUNE_GraceRunTime,DUNE_OriginalRunTime)

lines_6 = periodic_release=DUNE_ShouldRelease
```

Let's describe these lines. Starting from the auotrelease for memory, in the first line there is the value in Kb of the memory available for the job on the grid computer. It is set at a very low value because we want that the job goes to be held for exceeding Original Memory to test the autorelease. In line 2 there is the value of memory that can be added to the original one, if job is held for exceeding the same. It is also in Kb.The following lines are the heart of the code. Basically each hold reason has a own code and in *IncreaseReqMem* and *ShouldRelease* there are the condition for which the job has to be released with the adding memory. If they are evaluated as true then request memory is set equal to DUNE_OriginalMemory+DUNE_GraceMemory, if not request_memory remains equal to the initial OrginalMemory. Then if the ShouldRelease is evaluated to true, the periodic_release turns on and the job is automatically released.
The same structure is also set for the autorelease for run time. Basically in the first two lines there are the values of the Original Run Time and the Grace one (in second). Also in this case OriginalRunTime is set to a low value to force the job to be held for exceeding that value. The following lines are always the condition for which the autorelease should be active and the *ifthenelse* method works as describe above. So if IncreseJobLifeTime is evaluated to true, the run time available for the job is the sum of the Original Run Time and the Grace run time, if not it remains equal to the Original one. And if the ShouldRelease expression is true, the periodic release turns on also in this case.
In Figures 2.2 and 2.3 you can see the results of what I did. There are two typical job log files with the information about each job release. And after a while the job exceeded the value of original memory in the first case, and of run time in the second one, both the autorelease are well done and the jobs are released again, and they were able to be completed.
It is still to define a code that combines both the autorelease in a single expression. When I left the situation was under the control of the Fermi Service Desk who was trying to fix some limitations that were present during writing of a configuration files, in which the number of condition that you can add for the autorelease was not enough to combine the two expressions. After this issue will be fixed, it will be possible to think to one single expression.

```
...
028 (22405182.000.000) 08/27 15:35:51 Job ad information event triggered.
JOB_Site = "SGridECDF"
JOB_GLIDEIN_Name = "gfactory_instance"
Size = 2088152
Proc = 0
JOB_GLIDEIN_Entry_Name = "DUNE_UK_SGridECDF_ce1"
EventTime = "2019-08-27T15:35:51"
TriggerEventTypeName = "ULOG_IMAGE_SIZE"
JOB_GLIDEIN_SiteWMS_Queue = "eddie"
MemoryUsage = 936
TriggerEventTypeNumber = 6
JOB_GLIDEIN_Site = "SGridECDF"
JOB_GLIDEIN_SiteWMS_JobId = "2917475"
MyType = "JobImageSizeEvent"
JOB_GLIDEIN_ProcId = "0"
JOB_GLIDEIN_Schedd = "schedd_glideins7@gfactory-2.opensciencegrid.org"
JOB_GLIDEIN_ClusterId = "596963"
Cluster = 22405182
JOB_GLIDEIN_Factory = "OSG"
JOB_GLIDEIN_SiteWMS_Slot = "Unknown"
Subproc = 0
ResidentSetSize = 957560
EventTypeNumber = 28
JOB_GLIDEIN_SiteWMS = "SGE"
...
012 (22405182.000.000) 08/27 15:35:51 Job was held.
         SYSTEM_PERIODIC_HOLD  Memory/limit 936/1.000000000000000E+01
        Code 26 Subcode 1
...
028 (22405182.000.000) 08/27 15:35:51 Job ad information event triggered.
JOB_Site = "$$(GLIDEIN_Site:Unknown)"
JOB_GLIDEIN_Name = "$$(GLIDEIN_Name:Unknown)"
Proc = 0
JOB_GLIDEIN_Entry_Name = "$$(GLIDEIN_Entry_Name:Unknown)"
EventTime = "2019-08-27T15:35:51"
TriggerEventTypeName = "ULOG_JOB_HELD"
HoldReasonCode = 26
JOB_GLIDEIN_SiteWMS_Queue = "$$(GLIDEIN_SiteWMS_Queue:Unknown)"
TriggerEventTypeNumber = 12
HoldReason = " SYSTEM_PERIODIC_HOLD  Memory/limit 936/1.000000000000000E+01"
JOB_GLIDEIN_Site = "$$(GLIDEIN_Site:Unknown)"
JOB_GLIDEIN_SiteWMS_JobId = "$$(GLIDEIN_SiteWMS_JobId:Unknown)"
MyType = "JobHeldEvent"
HoldReasonSubCode = 1
JOB_GLIDEIN_ProcId = "$$(GLIDEIN_ProcId:Unknown)"
JOB_GLIDEIN_Schedd = "$$(GLIDEIN_Schedd:Unknown)"
JOB_GLIDEIN_ClusterId = "$$(GLIDEIN_ClusterId:Unknown)"
Cluster = 22405182
JOB_GLIDEIN_Factory = "$$(GLIDEIN_Factory:Unknown)"
JOB_GLIDEIN_SiteWMS_Slot = "$$(GLIDEIN_SiteWMS_Slot:Unknown)"
Subproc = 0
EventTypeNumber = 28
JOB_GLIDEIN_SiteWMS = "$$(GLIDEIN_SiteWMS:Unknown)"
...
013 (22405182.000.000) 08/27 15:36:22 Job was released.
        The job attribute PeriodicRelease expression 'DUNE_ShouldRelease' evaluated to TRUE
...
028 (22405182.000.000) 08/27 15:36:22 Job ad information event triggered.
JOB_GLIDEIN_Name = "$$(GLIDEIN_Name:Unknown)"
JOB_Site = "$$(GLIDEIN_Site:Unknown)"
Proc = 0
JOB_GLIDEIN_Entry_Name = "$$(GLIDEIN_Entry_Name:Unknown)"
EventTime = "2019-08-27T15:36:22"
TriggerEventTypeName = "ULOG_JOB_RELEASED"
JOB_GLIDEIN_SiteWMS_Queue = "$$(GLIDEIN_SiteWMS_Queue:Unknown)"
TriggerEventTypeNumber = 13
JOB_GLIDEIN_Site = "$$(GLIDEIN_Site:Unknown)"
JOB_GLIDEIN_SiteWMS_JobId = "$$(GLIDEIN_SiteWMS_JobId:Unknown)"
MyType = "JobReleaseEvent"
JOB_GLIDEIN_ProcId = "$$(GLIDEIN_ProcId:Unknown)"
JOB_GLIDEIN_Schedd = "$$(GLIDEIN_Schedd:Unknown)"
JOB_GLIDEIN_ClusterId = "$$(GLIDEIN_ClusterId:Unknown)"
Cluster = 22405182
JOB_GLIDEIN_Factory = "$$(GLIDEIN_Factory:Unknown)"
Reason = "The job attribute PeriodicRelease expression 'DUNE_ShouldRelease' evaluated to TRUE"
JOB_GLIDEIN_SiteWMS_Slot = "$$(GLIDEIN_SiteWMS_Slot:Unknown)"
Subproc = 0
EventTypeNumber = 28
JOB_GLIDEIN_SiteWMS = "$$(GLIDEIN_SiteWMS:Unknown)"
...
```

Figure 2.2: Job log with autorelease for memory

```
...
028 (22206470.000.000) 08/21 12:02:33 Job ad information event triggered.
JOB_Site = "CCIN2P3"
JOB_GLIDEIN_Name = "gfactory_instance"
Size = 1695632
Proc = 0
JOB_GLIDEIN_Entry_Name = "CDF_FR_CCIN2P3_cccreamceli09_long"
EventTime = "2019-08-21T12:02:33"
TriggerEventTypeName = "ULOG_IMAGE_SIZE"
JOB_GLIDEIN_SiteWMS_Queue = "long"
MemoryUsage = 24
TriggerEventTypeNumber = 6
JOB_GLIDEIN_Site = "CCIN2P3"
JOB_GLIDEIN_SiteWMS_JobId = "14886317"
MyType = "JobImageSizeEvent"
JOB_GLIDEIN_ProcId = "0"
JOB_GLIDEIN_Schedd = "schedd_glideins8@gfactory-2.opensciencegrid.org"
JOB_GLIDEIN_ClusterId = "732814"
Cluster = 22206470
JOB_GLIDEIN_Factory = "OSG"
JOB_GLIDEIN_SiteWMS_Slot = "Unknown"
Subproc = 0
ResidentSetSize = 23560
EventTypeNumber = 28
JOB_GLIDEIN_SiteWMS = "SGE"
...
012 (22206470.000.000) 08/21 12:09:34 Job was held.
         SYSTEM_PERIODIC_HOLD  Run Time/limit 436/300
        Code 26 Subcode 8
...
028 (22206470.000.000) 08/21 12:09:34 Job ad information event triggered.
JOB_Site = "$$(GLIDEIN_Site:Unknown)"
JOB_GLIDEIN_Name = "$$(GLIDEIN_Name:Unknown)"
Proc = 0
JOB_GLIDEIN_Entry_Name = "$$(GLIDEIN_Entry_Name:Unknown)"
EventTime = "2019-08-21T12:09:34"
TriggerEventTypeName = "ULOG_JOB_HELD"
HoldReasonCode = 26
JOB_GLIDEIN_SiteWMS_Queue = "$$(GLIDEIN_SiteWMS_Queue:Unknown)"
TriggerEventTypeNumber = 12
HoldReason = " SYSTEM_PERIODIC_HOLD  Run Time/limit 436/300"
JOB_GLIDEIN_Site = "$$(GLIDEIN_Site:Unknown)"
JOB_GLIDEIN_SiteWMS_JobId = "$$(GLIDEIN_SiteWMS_JobId:Unknown)"
MyType = "JobHeldEvent"
HoldReasonSubCode = 8
JOB_GLIDEIN_ProcId = "$$(GLIDEIN_ProcId:Unknown)"
JOB_GLIDEIN_Schedd = "$$(GLIDEIN_Schedd:Unknown)"
JOB_GLIDEIN_ClusterId = "$$(GLIDEIN_ClusterId:Unknown)"
Cluster = 22206470
JOB_GLIDEIN_Factory = "$$(GLIDEIN_Factory:Unknown)"
JOB_GLIDEIN_SiteWMS_Slot = "$$(GLIDEIN_SiteWMS_Slot:Unknown)"
Subproc = 0
EventTypeNumber = 28
JOB_GLIDEIN_SiteWMS = "$$(GLIDEIN_SiteWMS:Unknown)"
...
013 (22206470.000.000) 08/21 12:10:07 Job was released.
        The job attribute PeriodicRelease expression 'ShouldRelease' evaluated to TRUE
...
028 (22206470.000.000) 08/21 12:10:07 Job ad information event triggered.
JOB_GLIDEIN_Name = "$$(GLIDEIN_Name:Unknown)"
JOB_Site = "$$(GLIDEIN_Site:Unknown)"
Proc = 0
JOB_GLIDEIN_Entry_Name = "$$(GLIDEIN_Entry_Name:Unknown)"
EventTime = "2019-08-21T12:10:07"
TriggerEventTypeName = "ULOG_JOB_RELEASED"
JOB_GLIDEIN_SiteWMS_Queue = "$$(GLIDEIN_SiteWMS_Queue:Unknown)"
TriggerEventTypeNumber = 13
JOB_GLIDEIN_Site = "$$(GLIDEIN_Site:Unknown)"
JOB_GLIDEIN_SiteWMS_JobId = "$$(GLIDEIN_SiteWMS_JobId:Unknown)"
MyType = "JobReleaseEvent"
JOB_GLIDEIN_ProcId = "$$(GLIDEIN_ProcId:Unknown)"
JOB_GLIDEIN_Schedd = "$$(GLIDEIN_Schedd:Unknown)"
JOB_GLIDEIN_ClusterId = "$$(GLIDEIN_ClusterId:Unknown)"
Cluster = 22206470
JOB_GLIDEIN_Factory = "$$(GLIDEIN_Factory:Unknown)"
Reason = "The job attribute PeriodicRelease expression 'ShouldRelease' evaluated to TR
JOB_GLIDEIN_SiteWMS_Slot = "$$(GLIDEIN_SiteWMS_Slot:Unknown)"
Subproc = 0
EventTypeNumber = 28
JOB_GLIDEIN_SiteWMS = "$$(GLIDEIN_SiteWMS:Unknown)"
...
```

Figure 2.3: Job log with autorelease for run time

3 LArSoft

The Liquid Argon Software (LArSoft) Collaboration develops and supports a shared base of physics software across Liquid Argon (LAr) Time Projection Chamber (TPC) experiments.
The LArSoft software is designed to work for all planned and running liquid argon experiments at Fermilab. It is written in C++ and built on the ROOT data analysis software and the art analysis framework supported by the Fermilab Scientific Computing Division for intensity frontier experiments.
The core LArSoft code includes [3]

- A set of experiment-independent "detector interfaces" capable of representing the geometry, detector response, and material properties of the relevant LAr detectors to the reconstruction and simulation
- The data structures (art "data products") that represent the input data to, and the output objects from the various reconstruction and simulation algorithms
- The reconstruction algorithms that rely only on the detector interfaces and input data products to extract the physics content from event data
- The simulation algorithms that rely only on the detector interfaces and input data products to produce simulated data

3.0.1 Supernova samples of elastic scattering events campaign

My work on the autorelease code is only a part of what I did in the production group. Another part of my work has consisted in the development of a campaign in which we produced Supernova samples of elastic scattering events.
Generally all the MonteCarlo works are constituted from four parts. There is an *Event generation* (in which we decide the particles we are interested to study), then there is a *Geant4 simulation* (in which we simulate interaction between particles and matter), from that starts the *Detector Simulation* (where we can simulate detector response) and last but not least the *reconstruction* (where we obtain physical analyzable objects as tracks or electromagnetic showers).

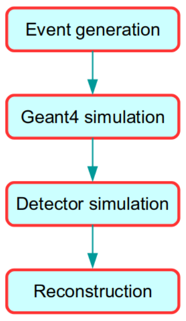
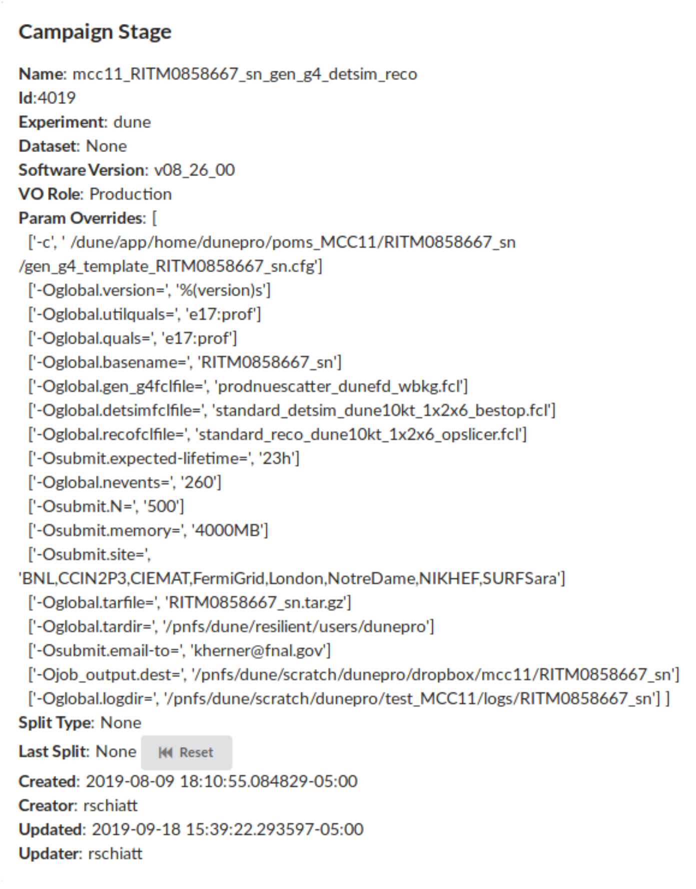


Figure 3.1: Four steps in a production job

All these steps are usually already implemented on LArSoft configuration files that basically have just to be launched.
So my work was to create a campaign on POMS and to put in the relative GUI editor the correct fcl files to use for the MonteCarlo simulations.



Campaign Stage

Name: mcc11_RITM0858667_sn_gen_g4_detsim_reco
Id:4019
Experiment: dune
Dataset: None
Software Version: v08_26_00
VO Role: Production
Param Overrides: [

```
  ['-c', ' /dune/app/home/dunepro/poms_MCC11/RITM0858667_sn
/gen_g4_template_RITM0858667_sn.cfg']
  ['-Oglobal.version=', '%(version)s']
  ['-Oglobal.utilquals=', 'e17:prof']
  ['-Oglobal.quals=', 'e17:prof']
  ['-Oglobal.basename=', 'RITM0858667_sn']
  ['-Oglobal.gen_g4fclfile=', 'prodnuescatter_dunefd_wbkg.fcl']
  ['-Oglobal.detsimfclfile=', 'standard_detsim_dune10kt_1x2x6_bestop.fcl']
  ['-Oglobal.recofclfile=', 'standard_reco_dune10kt_1x2x6_opslicer.fcl']
  ['-Osubmit.expected-lifetime=', '23h']
  ['-Oglobal.nevents=', '260']
  ['-Osubmit.N=', '500']
  ['-Osubmit.memory=', '4000MB']
  ['-Osubmit.site=',
'BNL,CCIN2P3,CIEMAT,FermiGrid,London,NotreDame,NIKHEF,SURFSara']
  ['-Oglobal.tarfile=', 'RITM0858667_sn.tar.gz']
  ['-Oglobal.tardir=', '/pnfs/dune/resilient/users/dunepro']
  ['-Osubmit.email-to=', 'kherner@fnal.gov']
  ['-Ojob_output.dest=', '/pnfs/dune/scratch/dunepro/dropbox/mcc11/RITM0858667_sn']
  ['-Oglobal.logdir=', '/pnfs/dune/scratch/dunepro/test_MCC11/logs/RITM0858667_sn'] ]
```

Split Type: None
Last Split: None
Created: 2019-08-09 18:10:55.084829-05:00
Creator: rschiatt
Updated: 2019-09-18 15:39:22.293597-05:00
Updater: rschiatt

Figure 3.2: Supernova samples campaign

In the Figure 3.2 there is the main page of the Supernova Samples Campaign. In the parameters Overrides section there are all the information that through POMS are given to the computer grid on which the jobs have been submitted. There are obviously all the fcl files the project needs. In this case the event generation and the G4 simulation are in a single fcl file. Other important

parameters are the expected lifetime of these jobs and the amount of memory they will take on the grid, and also things like the site the jobs can use and the output and log files destination.
This campaign produced a sample of 500 jobs with 260 events per job.
These supernova samples of elastic scattering events will be combined with supernova samples of charged current events to make supernova samples, and those will be used to find DUNE's pointing resolution for supernovae when there are backgrounds and detector noise. In this way it will be possible to know how accurately we can find the direction of supernova neutrinos and thus also the location of supernovae.

4 Reprocessing Campaigns for protoDUNE SP data

In the last part of my work with the production group, I started working with them on the campaign for the new reprocessing of protoDUNE Single Phase data. We are reprocessing good beam runs with dunetpc version v08_27_01 and also we are generating new montecarlo simulations for several beam momentum conditions (in particular 1,2,3,6,7 GeV). We started working on that two weeks before I left, and for the full campaign we expected 4-6 weeks to be completed.

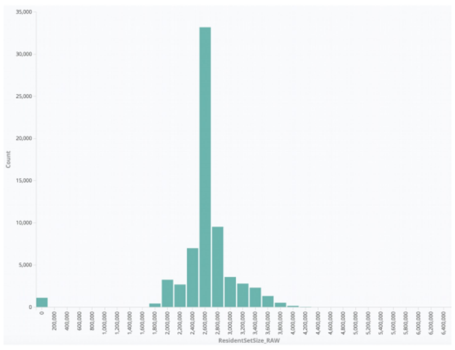


Figure 4.1: Usage Memory for data production in Kb

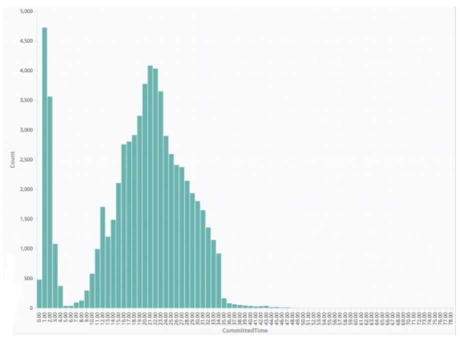


Figure 4.2: Usage Run Time for data production in h

In Figures 4.1 and 4.2 you can see the values of Memory and Run Time for each job that we launched. For the data we have a peak of the usage memory at 2.6 Gb and a run time peak at 22 h. In Figures 4.3 and 4.4 there are the same plots for the Montecarlo simulation jobs. Here we have a peak of usage memory at 3.4 Gb and a run time peak at 4 h. [3]

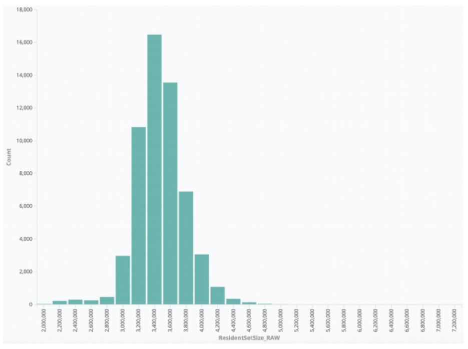


Figure 4.3: Usage Memory for Montecarlo Simulation in Kb

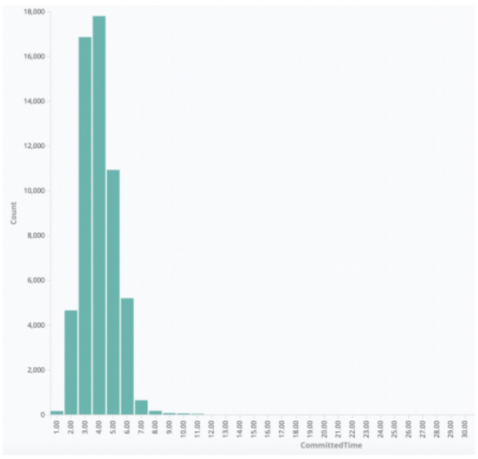


Figure 4.4: Usage Run Time for Montecarlo Simulation in h

4.0.1 Data Validation

The last step of my work with the production group consisted in the validation of the reprocessing data. Basically I compared the latest production files with the previous production sample, obtained with dunetpc version $v07\_08\_00\%$.
I started with data run 5387, that was almost completed before my last week at Fermilab.
Writing a LArsoft module I was able to get some important information about analyzable quantities for each event in the data. So we get information about hits, PFParticles (Particle Flow Particles), tracks and electromagnetic showers.
The following histograms are about these quantities. In red we have the distributions coming from the old production data, while in blue the one form the new production data.
Starting with the hits, you can see the distribution about the number of hits per event(Figure 4.5), and the distribution of charge integral and peak time

for each hit (Figure 4.6).

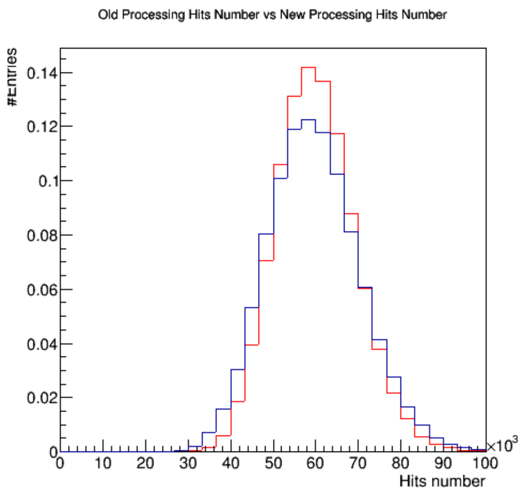


Figure 4.5: Number of hits for each event in the production data. In red the one coming from the old production data, and in blue the one coming from the new production data.

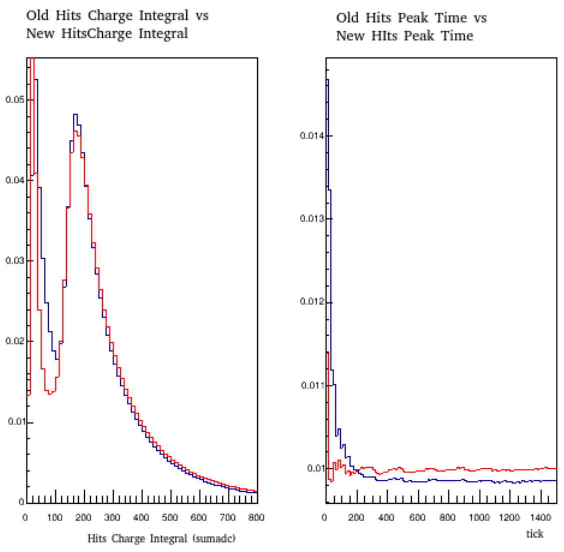


Figure 4.6: Hits charge integral and hits peak time distribution. In red the one coming from the old production data, and in blue the one coming from the new production data.

By the module that I wrote, we get also distributions about tracks in the events.
In Figure 4.7 you can see the number of tracks for each event, while the other distributions are about start track position(Figure 4.8), end track position (Figure 4.9) and angular distributions at track start and end point (Figure 4.10).

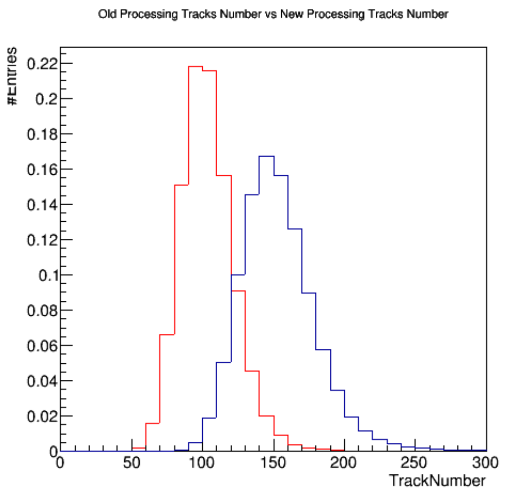


Figure 4.7: Number of Tracks for each event in the production data. In red the one coming from the old production data, and in blue the one coming from the new production data.

Other important tracks related quantities about which we have information are Number trajectory points, momentum at track start point and at end start point and tracks length.

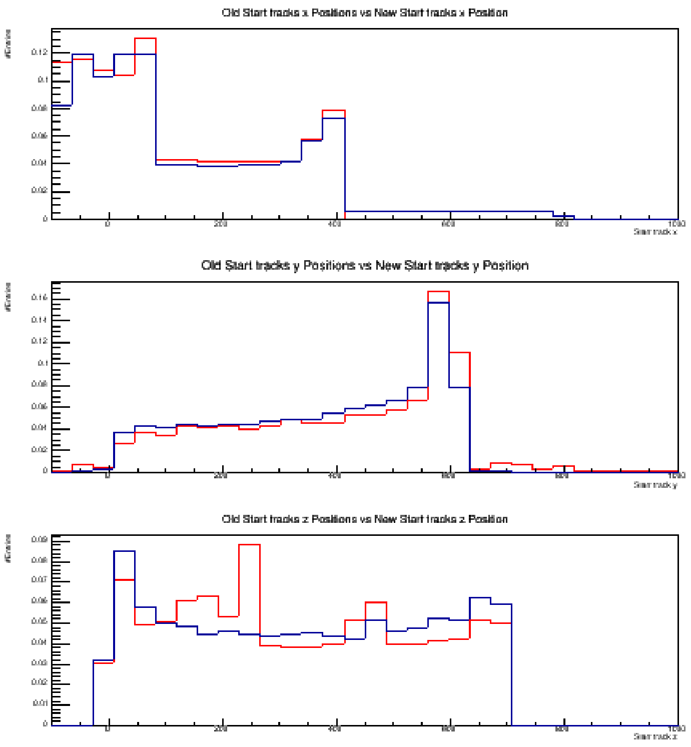


Figure 4.8: Track start point in cm. In red the one coming from the old production data, and in blue the one coming from the new production data.

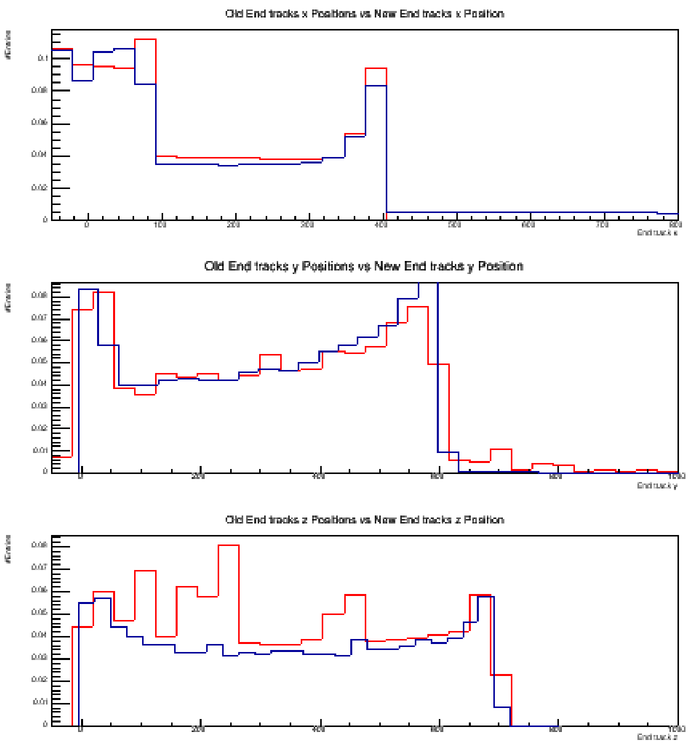


Figure 4.9: Track end point in cm. In red the one coming from the old production data, and in blue the one coming from the new production data.

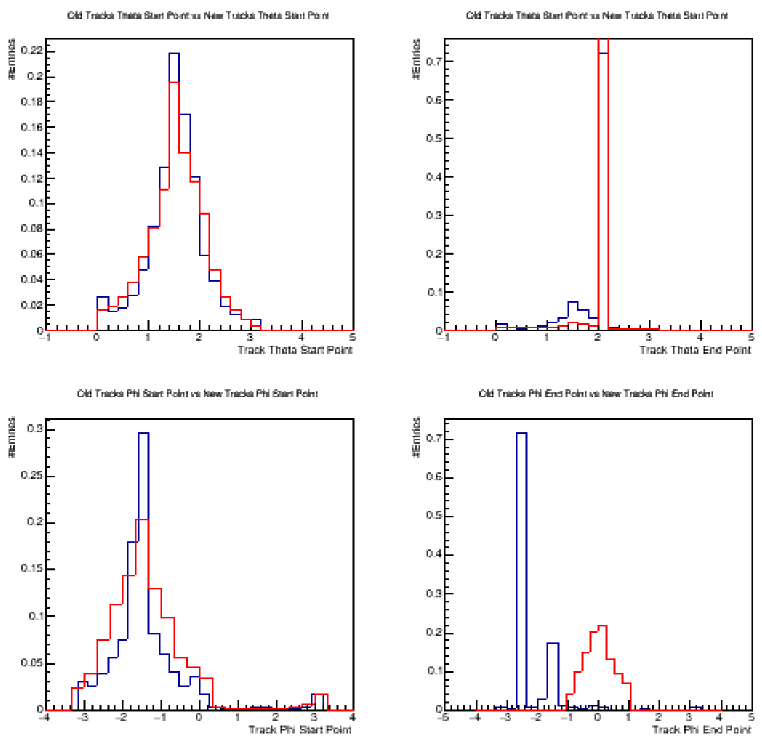


Figure 4.10: Angular distributions for each track (rad). In red the one coming from the old production data, and in blue the one coming from the new production data.

The module I wrote, was looking for information about the PFParticle in the events, and in Figure (4.11) there is the number of PFParticle reconstructed for each event.

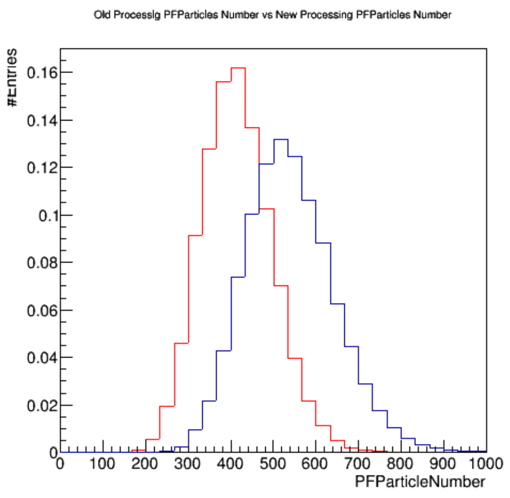


Figure 4.11: Number of Particle Flow Particles. In red the one coming from the old production data, and in blue the one coming from the new production data.

As I said we have information also about electromagnetic showers. In Figure

(4.12) there is the showers number for each event, while in Figure (4.13) there are the distributions about showers length and showers open angle.

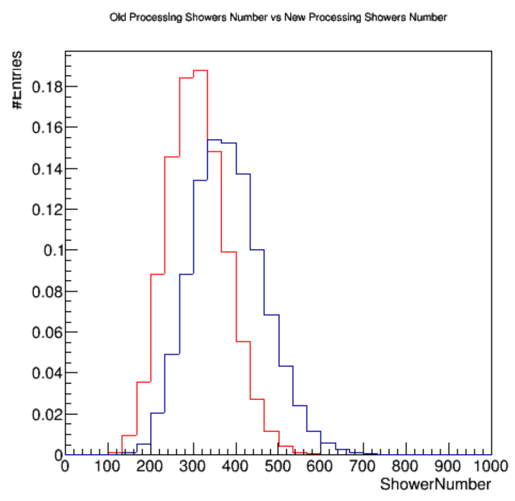


Figure 4.12: Number of electromagnetic showers. In red the one coming from the old production data, and in blue the one coming from the new production data.

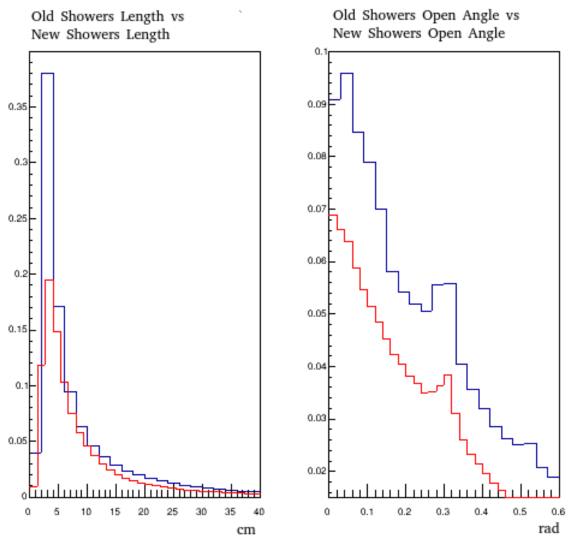


Figure 4.13: Showers Length and open angle distributions. In red the one coming from the old production data, and in blue the one coming from the new production data.

As you can see from the previous plots, there are some differences between the distributions coming from the old production data and the distributions that coming from the new production data.
So the natural future step for the production and the analysis group is to understand the reasons of these differences. Also they can use my module to

test all the other production runs, trying to understand if all the data have been reprocessed in a correct way.

Conclusion

During my time with the production group I worked on several fronts. First of all I developed a code for autorelease of held jobs, both if the jobs are held for exceeding the run time declared value if the jobs are held for exceeding the memory declared value. I noticed an important limitation about the possibility to combine two autorelease codes into a single one, and in collaboration with Fermilab Service Desk we are trying to fix this issue, to implement a definitive code to use in all future production group jobs.
At the same time I worked on the Supernova samples of elastic scattering events campaign, producing 500 jobs with 260 events per job. Those simulations will be useful to improve DUNE pointing resolution for neutrinos coming from Supernovae bursts.
In a second moment I collaborated with the production group to the reprocessing protoDUNE single phase data, that should be completed during the next week. Also I started to work on data validation for data coming from the reprocessing campaign.
Basically I wrote a LArSoft module to obtain several histograms to compare with the same histograms obtained with old reprocessing data.
We noticed some differences between new and the old data production plots and starting from my work it will be possible to understand where these differences coming from. In this way it will be possible start to understand what are differences between the two dunetpc version used.

Bibliography

[1] https://www.dunescience.org

[2] https://cdcvs.fnal.gov/redmine/projects/prod_mgmt_db/wiki/POMS_User_Documentat

[3] https://cdcvs.fnal.gov/redmine/projects/larsoft/wiki/Introduction_to_LArSoft

[4] https://indico.fnal.gov/event/21445/session/19/contribution/14/material/slides/0.pdf