Data Handling with SAM and ART at the NOνA Experiment

Adam Aurisano[1], Christopher Backhouse[2], Gavin Davies[3], Robert Illingworth[4], Nathan Mayer[5], Marc Mengel[4], Andrew Norman[4], Dominick Rocco[6], Jan Zirnstein[6]

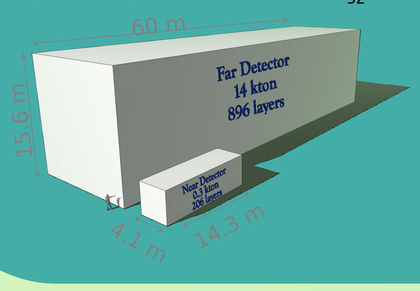
[1]University of Cincinnati, [2]California Institute of Technology, [3]Indiana University, [4]Fermilab, [5]Tufts University, [6]University of Minnesota

NOνA

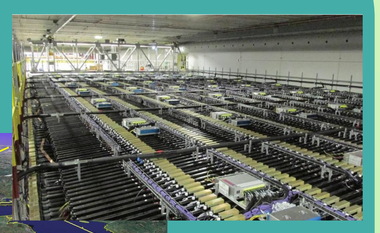
NOνA is a long-baseline neutrino oscillation experiment located 14 mrad off-axis from the mostly ν_μ NuMI beam and is designed to measure:

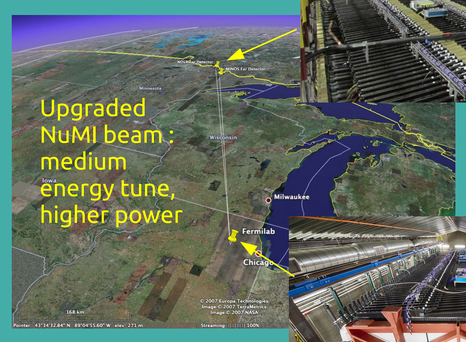
ν_e appearance:
- θ_{13}
- θ_{23} octant
- mass hierarchy
- constraints on CP violation

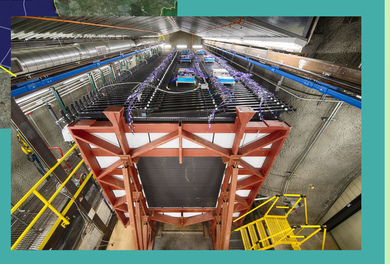
ν_μ disappearance:
- improved precision on $|\Delta m^2_{32}|$ and θ_{23}



The far detector is located 810 km from FNAL, in Ash River, MN





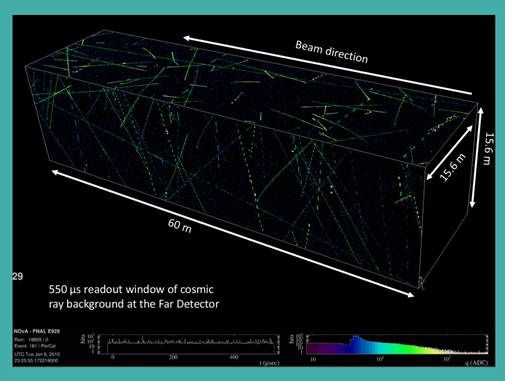


The near detector is located 1 km from the target, on the FNAL campus.

Big Data

NOνA produces a large volume of data, both in number of files and total file size.
- NOνA produces ~5,000-7,000 raw files per day with peaks in excess of 12,000.
- Raw files must be processed through several data tiers to produce fully calibrated and reconstructed analysis files.
- Many simulated neutrino interactions must be produced and processed through the same stages as data.

We accomplish this using the SAM system, developed at Fermilab. SAM provides many services, including:
- File indexing according to metadata
- Cataloging the physical location (potentially multiple) of files
- Dataset management
- Facilitating data transfer to off-site grids



A typical 500 µs event at the far detector. Since the far detector is on the surface, it sees ~40 kHz of cosmic rays. The large cosmic ray rate, combined with the large number of channels (~344,000 for the far detector) leads to very large datasets, even though neutrinos interact rarely.

Draining Datasets

To facilitate bookkeeping in production jobs, we developed the **draining dataset paradigm**.
- In SAM a *dataset* is a query based on metadata registered for each file.
- A *draining dataset* is one that includes a clause based on parentage and data tiers, where data tiers are sequential stages of data processing.
  - For instance, a draining dataset used to process raw files from our proprietary raw format into an ART-compatible format would include a clause like:
    "data_tier raw and not isparentof: (data_tier artdaq)"
- More specifications can be added to only look at files that have been processed with certain software versions.

Draining datasets have a variety of benefits:
- As files are produced, the draining dataset automatically expands.
- As regular cron jobs successfully process those files, the draining dataset automatically shrinks.
- This makes it possible to have asynchronous data processing as many tiers can be processed in cron jobs simultaneously with files disappearing from one tier and becoming ready for processing in the next tier automatically.

Draining Dataset Cycle

Submit job to the grid.
- A project starts at job submission time
- SAM projects contain a list of all files satisfying the dataset definition at creation time

A wrapper script establishes a consumer process for each grid node.
- Consumer processes request input files from SAM project
- SAM provides urls the IFDH tool set can fetch with grid protocols

A python wrapper extracts metadata from each input file.
- Constructs output file name according to metadata
- Constructs and runs the ART executable invocation

The ART job includes a NOνA specific metadata module to transfer metadata to the output file from:
- The configuration file
- Input files
- The command line invocation
- Dynamically generated metadata

Finished files are transferred to a directory watched by the File Transfer System (FTS) that:
- Parses the metadata of files
- Declares them to SAM
- Copies them to dCache and enstore (tape storage)

Due to the parentage information in the dataset, successfully processed files will not appear in subsequent submissions of the dataset.

Metadata

SAM indexes files based on metadata, so generating metadata and ensuring its correctness is of prime importance. To ensure that metadata cannot be separated from the files with which it is associated, we embed metadata inside our files:

Encode metadata in configuration files as ART parameter set

Custom NOνA metadata module

- Extract metadata from configuration file
- Fetch metadata from input ART files using IFDH calls
- Copy metadata from the command-line and environment vars
- Dynamically generate metadata like # of events and parentage

Use ART metadata service to copy to embedded sqlite database

Use metadata module method to dump metadata as a C++ map for analysis ROOT files

Use FTS plugins developed for each file type (configuration files, ART files, and analysis ROOT files) to parse embedded metadata and declare to SAM

Monte Carlo Production

In addition to facilitating metadata transfer into ART files, embedding metadata in configuration files allows for the creation of special configuration file datasets for use in Monte Carlo production where the configuration files are treated like input files.

This approach provides a number of benefits:
- Configuration files act as a data tier analogous to raw files in data, so no special handling is necessary to start a Monte Carlo generation job.
- Configuration datasets are fully indexed in SAM and saved to dCache, so it is easy to search for and download a few representative files to understand how a Monte Carlo set was generated.
- The metadata module correctly sets the parentage of the output Monte Carlo files, so one can easily download the configuration file used to generate a particular Monte Carlo file.
- Careful construction of metadata provides effective versioning for Monte Carlo runs. This makes it easy to perform validation of new Monte Carlo features.

21st International Conference on Computing in High Energy and Nuclear Physics
April 2015, Okinawa, Japan

@NOvANuZ

facebook.com/novaexperiment

www-nova.fnal.gov