Fermilab LDRD Proposal

Project Title: Adaptable data-processing for HEP computing frameworks

Principal Investigator: Kyle Knoepfel
Lead Division/Sector/Section: Scientific Computing
Co-Investigators (w/institutions): N/A

Project Description

The goal of this project is to enable a HEP computing framework to adapt its processing *in situ* to varying data-collection windows, thus addressing significant data-processing challenges faced by DUNE and other HEP experiments.

Instead of localized accelerator bunch crossings, neutrino physicists analyze interaction regions and data-collection time windows that sometimes overlap or vary in size—flexible constructs poorly supported by today's collider-based frameworks that assume fixed, rigidly defined data groupings. Neutrino physicists work around these limitations by manually transforming collider-friendly data groupings into ones more relevant for neutrino analysis. Such transformations are a source of potential error and expensive to implement as neutrino physicists must repackage the data to analyze it.

To support the physics program of DUNE, Fermilab's flagship experiment, a new framework paradigm is thus required: one that leverages HEP framework successes over the past 20 years yet allows for adaptable processing based on the data under examination, allowing neutrino and collider physicists alike to concentrate on the physics. Such a paradigm would also reduce the need to develop and maintain multiple frameworks, resulting in cost savings for the HEP community.
This proposed project addresses these issues by meshing state-of-the-art capabilities of the C++ programming language with an existing technology that can adjust to the data groupings necessary for analyzing neutrino or collider physics.

Significance

Figure 1 illustrates a DUNE far-detector module whose alternating anode and cathode planes create an electric field. This electric field drives ionized electrons from the neutrino interaction to an anode, which records them as electrical signals in one of the anode plane assemblies (APAs). Upon detecting a possible neutrino interaction, DUNE's data acquisition system will open a 5.4 millisecond time window to collect the electrical signals from nearby APAs. The computing framework will then process a *trigger record*, which is a full readout of all 150 APAs of a DUNE far-detector module for a given set of time windows.

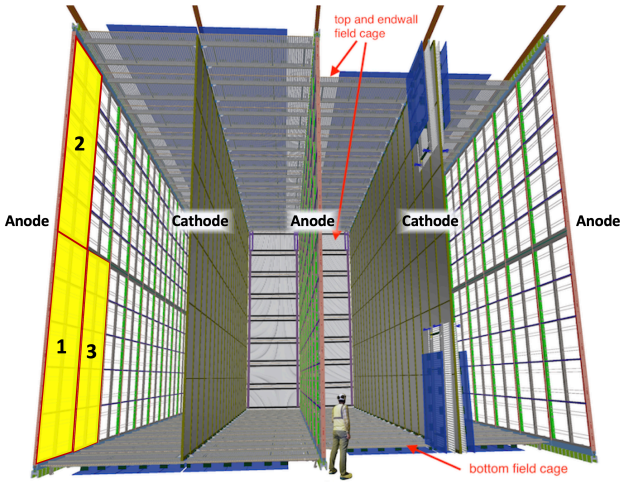

***Figure 1**. Diagram of one DUNE far-detector module. Each anode plane comprises 50 APAs, three of which are labeled "1", "2", and "3" above.*

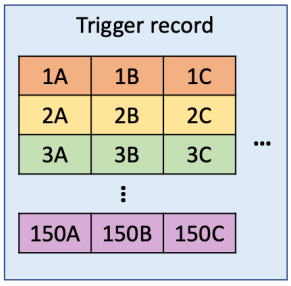

***Figure 2.** Trigger record as a data matrix. One axis is the APA (1-150), the other axis is the time window (A-C).*

Figure 2 shows a trigger record, which is analogous to a matrix of data grouped according to APA in one axis and time window in the other axis. Depending on the particular data-processing step, the trigger record may need to be processed along the APA axis, the time-window axis, as a whole unit, or as a different data-grouping altogether. An additional complication is that the data stored for one time window may overlap with the data stored for another time window, requiring careful bookkeeping to avoid analysis errors.

Regardless of how a framework job processes a trigger record, subsequent framework jobs must be able to interpret the data correctly while processing the trigger record using different data groupings that may even be distributed *across* trigger-record boundaries. As stated by the DUNE Software Framework Requirements Taskforce [1]:

> In neutrino physics we often need to make contextual switches between what the unit of interest is, where examples are the subsetting of extended accelerator spill structures into smaller time windows, or the subsetting of time windows into disconnected regions of concentrated activity, or the subsetting of activity regions into shower and particle track trajectories and objects. We need our framework to be able to operate over these units in a native manner, instead of having everything below the highest organizational unit be looped over manually by a user module.

To illustrate the types of "context switches" required of the framework, consider Figure 3, which shows a supernova data-processing workflow. The framework first runs its physics-reconstruction steps by processing the data according to each APA. A subsequent framework job may then analyze the output of the reconstruction steps according to the time window, enabling a time-dependent analysis of the neutrinos detected from a supernova. At the end of each framework job, the trigger record and any physics information calculated during the framework jobs must remain intact for further physics analysis.

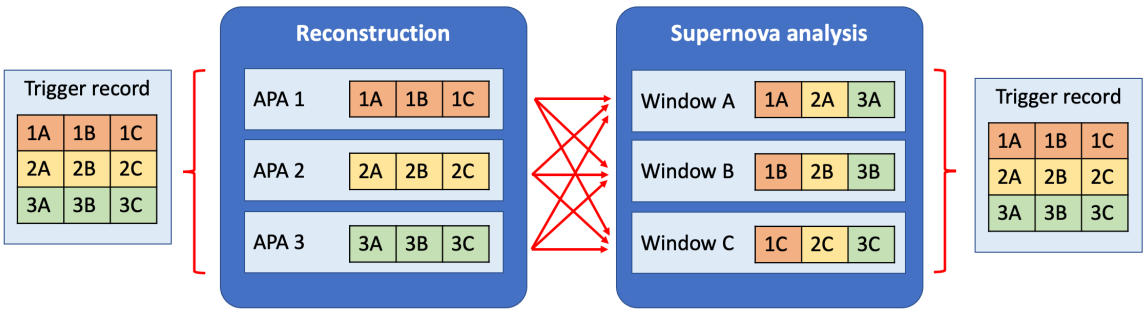

Figure 3. *Sample processing workflow of a supernova trigger record. For the first framework job, each APA is processed separately; for the second framework job, the data is analyzed according to time window. For illustration reasons, we use only 3 APAs and 3 time windows.*

A recent review of DUNE's framework needs states: "Although extant frameworks may be able to accommodate such processing, there is no single framework that we know of that readily provides such processing behavior" [2]. Specifically:

- The LHCb, ATLAS, Daya Bay, and LZ experiments have used the Gaudi framework [3], which supports collider-based concepts and not the sophisticated data hierarchies needed by DUNE.
- The CMS experiment and FNAL-based neutrino experiments use the CMSSW [4] and art [5] frameworks, respectively. Both frameworks originate from the same LHC-motivated code and neither one supports the flexible processing DUNE requires.
- Other HEP frameworks exist (e.g. [6-8]), but they also are largely collider-based and suffer from similar deficiencies as those described above.

With the proposed work, a framework will be able to process dynamically defined data organizations by automatically adapting to the needs of the processing step without requiring awkward interventions from the framework user. In addition to directly benefiting DUNE and other non-collider experiments, a technology capable of such data-processing would also support existing collider-based frameworks, further enabling the sharing of valuable, time-tested software between the collider and neutrino communities.

Research Plan

To achieve framework adaptability while retaining efficient computation, multiple technologies will need to work in concert: (a) modern C++ programming for robust data access; (b) a bookkeeping system to prevent analysis errors when using flexible data groupings; (c) a parallel-processing library that executes tasks concurrently on one or more computers, including supercomputing centers; and (d) a file-handling system that can accommodate the framework's parallel-processing behavior. Items (a) and (b) are the focus of year 1, and items (c) and (d) are the focus of year 2, building on knowledge learned and technology developed in the first year.

This project has three deliverables:

1. An adaptable prototype framework,
2. A hierarchical data-processing design, and
3. A granular and asynchronous file-writing approach.

Even if all three deliverables are not achieved, the experiences gained working toward each one are suitable for a journal article. This proposal thus assumes that up to three journal articles will be written over the course of the project. If necessary, deliverables 2 and 3 can be adjusted to accommodate challenges encountered in year 1.

A detailed breakdown of this project's objectives and the corresponding deliverables is below. Time estimates are expressed in calendar months assuming a 40% FTE commitment from myself and a 5% FTE commitment from a framework consultant at FNAL, which has a strong history in developing frameworks (CDF, DØ, CMS, and intensity-frontier). In addition, I will be applying for a summer student or intern, who will be externally funded (see Diversity and Inclusion Plan).

Year 1 objective: Support extensible and flexible data groupings

The *de facto* computing language for HEP frameworks is C++ due to its robust programming model and efficient processing. Its strengths, however, pose challenges when supporting flexible and dynamic data groupings as required by DUNE. The main thrust of the first year is to address these challenges by embedding within the C++-based framework program an adaptive technology that keeps track of the data groupings. **Deliverable 1** is thus a prototype framework program that can adapt its processing to the data-groupings specified by the individual running the program. This goal is achieved by accomplishing three tasks described below.

Task A (2 calendar months) is to understand the data groupings used in reconstruction and analysis programs of various neutrino experiments (e.g. DUNE, NOvA, and MicroBooNE). Starting with data from the detector, I will study how it is subdivided, processed, and grouped together for final physics analysis. Neutrino physicists and data analysts will be consulted to better understand current framework usage patterns.

Task B (4 calendar months) is to create a prototype C++ program to try different groupings of data for an existing neutrino framework. This task is composed of two subtasks:

1. Establish a simple coding pattern for physicists to use so the framework can discover what types of data groupings are supported by a given algorithm used in a framework job. To create a system that is not burdensome for the algorithm writer, experimental C++ facilities may be necessary, requiring interactions with international C++ experts for guidance.
2. Create a framework configuration system where the physicist can specify which data groupings to process for the job. For example, in Figure 3, the physicist must be able to specify whether the data should be processed according to APA (the first framework job) or according to time window (the second).

Task B is ideal for having input from a summer student or intern—a computing non-expert who, like a neutrino physicist, could provide feedback on what types of coding patterns are simple and intuitive to use.

Task C (5 calendar months) is perhaps the most difficult in achieving deliverable 1—the task is to *record* which data groupings were used for that job so that subsequent jobs understand the data groupings used by the stored data and calculated physics quantities. These data organization and relationship details are termed *metadata*, and they are crucial ingredients in creating a bookkeeping system that is extensible yet subject to the constraint that data cannot be misused (e.g. double-counted) when analyzing it. A technology such as a relational [9] or graph database [10] is well-suited for this kind of task. However, there are two complications:

- The database technology must integrate well with a C++-based framework.
- It is not yet known whether the required metadata can be expressed in a database format that can be stored in the same file as the physics data itself.

The art framework [5] has successfully used a relational database for many of its metadata needs. DUNE's needs, however, will likely require more expansive use of databases both during the job and in the framework file itself.

A year-end post-mortem (1 calendar month) is devoted to documenting the findings of deliverable 1 in a journal article. If it is not yet achieved, a determination will be made whether deliverable 1 should be decoupled from deliverables 2 and 3 and whether work on deliverable 1 should be suspended.

Year 2 objective: Efficient processing of data groupings

Year 1 focuses on building a framework technology that can process a trigger record according to different groupings of data. The focus for year 2 is to process such groupings in an efficient manner. A common way of doing this is to process multiple data groupings in parallel, where the framework is responsible for scheduling the execution of algorithms that physicists want to use in their framework job. This phase of the project will explore

two challenges inherent in the parallel processing of data groupings—these challenges are addressed in year 2 with the two deliverables listed below.

Deliverable 2 (4 calendar months) is to create a design for processing hierarchical data groupings in parallel. In real data-taking, data groupings such as trigger records do not exist by themselves but reside within an often complex data hierarchy. To illustrate, see Figure 4, which depicts the simplest hierarchy by grouping trigger records within runs.

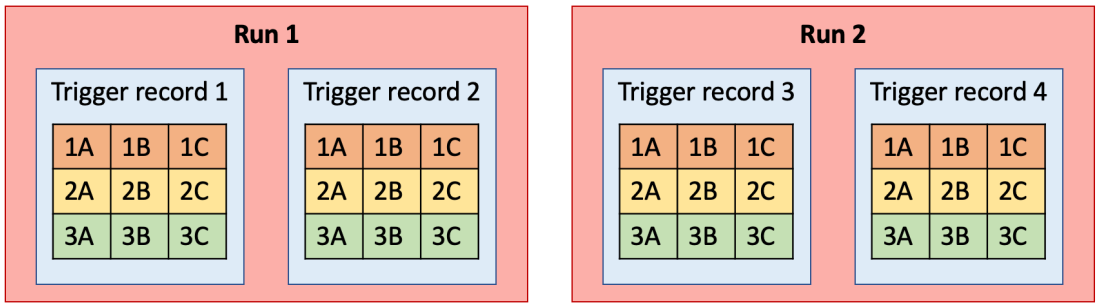

Figure 4. *Trigger records are grouped into runs, each of which may have calibrations to be applied to the data.*

Boundaries between runs are used to signify physical changes in the experiment such as different beam conditions or detector configurations. When analyzing data from these runs, physicists then apply adjustments or *calibrations* to the collected data to account for detector artifacts. Processing data from different runs in parallel is therefore a challenge because the framework must store and provide calibration information at a coarser level than just the trigger record. Deliverable 2 thus concentrates on creating a design pattern for parallel-processing a hierarchy of data groupings.

Although well-vetted, parallel-processing software exists (e.g. Intel's TBB [11], MPI [12], or HPX [13]), such software does not strongly support efficient processing of data hierarchies. The deliverable will require adopting one of those software options and interacting with parallel-processing software vendors to take full advantage of the parallel-processing capabilities of the computing system. Preference will be given to a software library that works well with high-performance computing systems supported by the DOE.

Deliverable 3 (6 calendar months) is to support granular and asynchronous writing to framework files. A difficulty of processing data in parallel is determining how to efficiently write that data to a file. With an art- or CMSSW-based framework, trigger record data would be written in a synchronized way to a file at trigger-record boundaries, which is illustrated in the left diagram of Figure 5. With this type of processing, the framework must store all of a trigger record's data in the program's memory, which grows until all trigger record components (e.g. "1A" through "3C") have been fully processed and that data is written to the file. The amount of DUNE data from a trigger record can easily exhaust the computer memory allowed for a framework program. It is therefore desirable to write data to a file once that data is no longer required so that the memory usage of the program can be reduced. The right diagram of Figure 5 shows how the memory usage of the framework program can be reclaimed while processing the trigger record and not at the end of processing it.

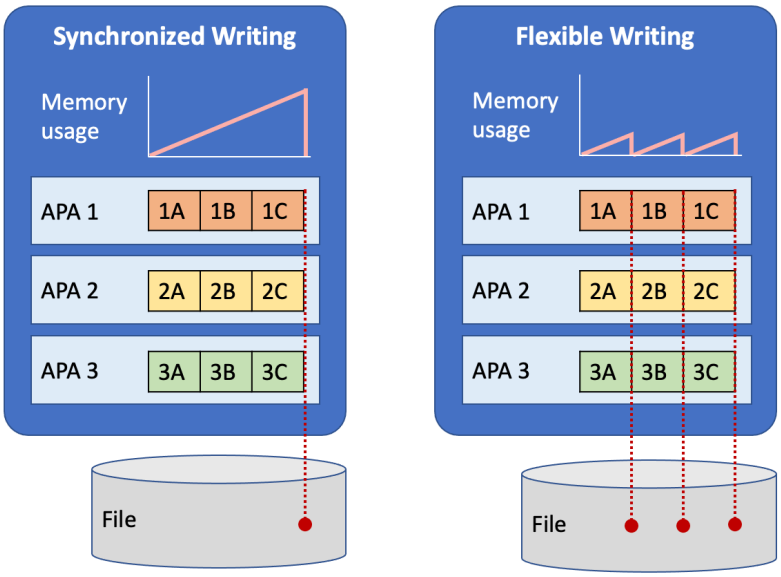

Figure 5. *Illustration showing how writing data to a file after processing a full trigger record (left) can lead to substantially more memory usage than writing to a file throughout the processing of one trigger record (right).*

A year-end post-mortem (2 calendar months) is used to document in one or more journal articles the experience gained when pursuing deliverables 2 and 3. If deliverable 1 was not achieved, these final 2 months may be used to summarize the entire project in one journal article. Regardless of whether the deliverables are entirely successful, a journal article of the knowledge learned in this project will be of interest to the HEP community and to DUNE in particular.

References

1. DUNE Software Framework Requirements Taskforce Report, Annotated for Non-DUNE Review [https://docs.google.com/document/d/11UnMis5yiokeM1W9wwWI1Uaa4gG38RxV3dD8S7Tt7ps]
2. DUNE Framework Requirements Review – HSF Findings (2021) [https://indico.cern.ch/event/1038551/attachments/2243908/3874756/DUNE-framework-requirements-HSF-findings.pdf].
3. See https://lhcb.github.io/developkit-lessons/first-development-steps/03a-gaudi.html and https://gitlab.cern.ch/gaudi/Gaudi
4. E Sexton-Kennedy *et al* J.Phys.Conf.Ser. 608 (2015) **1**, 012034 [https://inspirehep.net/literature/1372982, doi:10.1088/1742-6596/608/1/012034]
5. See https://art.fnal.gov; C. Green *et al* J.Phys.Conf.Ser. 396 (2012) 022020 [https://inspirehep.net/literature/1211020, doi:10.1088/1742-6596/396/2/022020]
6. ALICE O2 Analysis Framework [https://aliceo2group.github.io/analysis-framework/]
7. basf2 software framework [https://software.belle2.org]
8. MARLIN framework [https://github.com/iLCSoft/Marlin]
9. E. F. Codd. "A Relational Model of Data for Large Shared Data Banks" (1970). *Communications of the ACM*. **13** (6): 377–387. [doi:10.1145/362384.362685]
10. R. Angles, C. Gutierrez. "Survey of graph database models" (2008). *ACM Computing Surveys*. **40**(1): 1–39. [doi:10.1145/1322432.1322433]

11. See https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/onetbb.html#gs.8vm3sw
12. See https://www.mcs.anl.gov/research/projects/mpi/index.html
13. See https://hpx.stellar-group.org