art outlook

Kyle J. Knoepfel

Mu2e-II Workshop at Northwestern University

29 August 2018

art's approach

- *art* is intended to be used for analysis jobs and large-scale production jobs.

- Much of our approach is forward-looking. We aim to (e.g.):
  - run *art* on HPC machines
  - deliver cutting-edge software tools (e.g. TensorFlow)
  - enable experiments to benefit from modern language features (*art* supports C++17)

- We have users that prefer to develop on macOS systems
  - We support open-source Clang builds on macOS and Linux

- We actively contribute to the code bases of *art*-using experiments/projects:
  - As of May 2018, the SciSoft team supports and develops code for LArSoft

art's approach

- We support the offline (and some online) processing for 11 projects/experiments.

- Our development efforts are guided by:
  - Current and future needs of *art*-using experiments (represented by stakeholders).
  - Current and future software and hardware technology, in and outside of HEP.
  - Feedback from individual users, and our own estimation of what features would make *art* simpler to use.

- Experiment support:
  - Design guidance and code reviews at the request of experiments
  - Small-scale profiling efforts at the request of experiments

Who uses *art*?

ArgoNeuT

artdaq project

DUNE
DEEP UNDERGROUND NEUTRINO EXPERIMENT

Imaging Cosmic And Rare Underground Signals

LArIAT experiment

LAr Soft

μBooNE

Mu2e

μ g−2

NOvA

SHORT-BASELINE NEAR DETECTOR
SBND

Previous and potential users

DARKSIDE

next

supernemo collaboration

Fermilab

art concepts

- Hierarchical data processing ($run \supset subrun \supset event$)
- **Experiments decide** how to define the processing levels (e.g. event)
- All processing elements are plugins, loaded at run-time via user configuration
  - Input source
  - Data-processing modules
  - Output modules
  - Other utilities that facilitate data-processing
- *art* provides various input sources and output modules, but all processing elements can be user-defined
- Workflows are assembled by a configuration file loaded at run-time
  - Adjustments to workflows do not require recompilation of C++ source code

art concepts

- Hierarchical data processing ($run \supset subrun \supset event$)
- **Experiments decide** how to define the processing levels (e.g. event)
- All processing elements are plugins, loaded at run-time via user configuration
  - Input source
  - Data-processing modules
  - Output modules
  - Other utilities that facilitate data-processing
- *art* provides various input sources and output modules, but all processing elements can be user-defined
- Workflows are assembled by a configuration file loaded at run-time
  - Adjustments to workflows do not require recompilation of C++ source code

These concepts are agnostic to hardware and any specific I/O format.

What are frameworks targeting for 2025-2030?

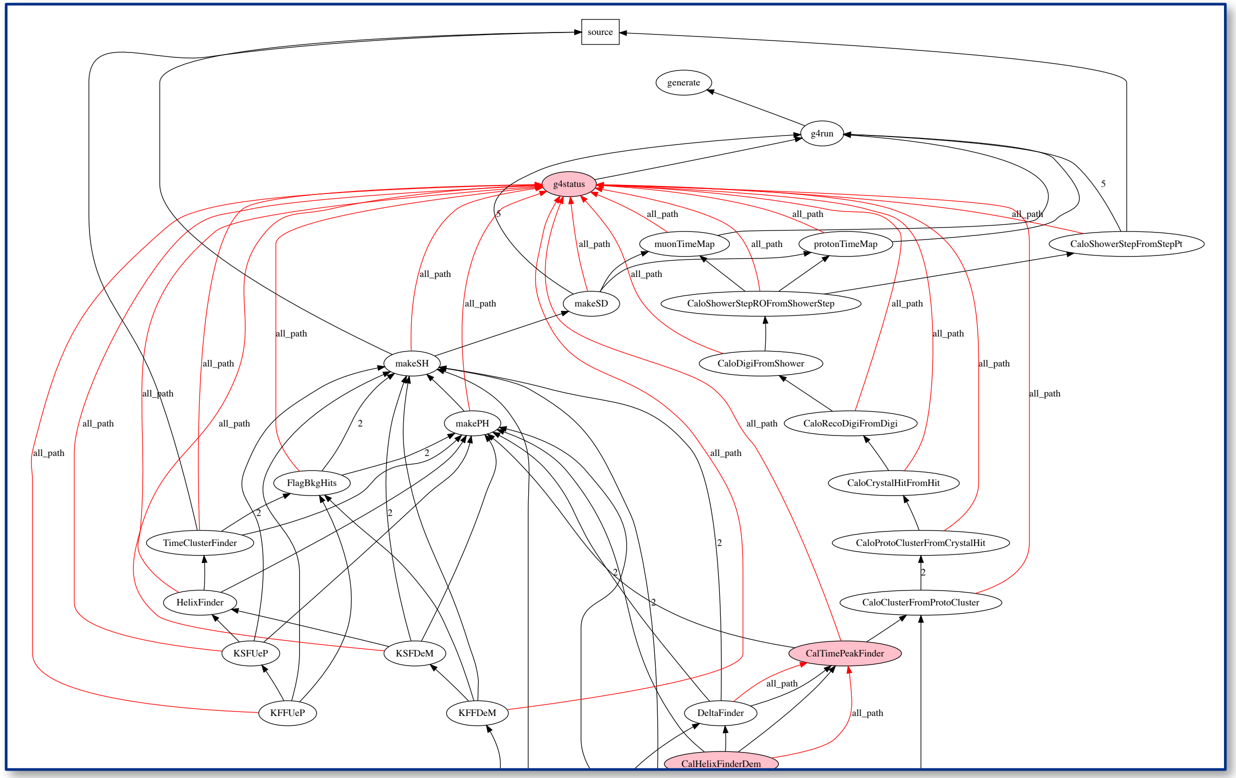
- Large data rates/sizes from the LHC experiments and DUNE have required rethinking how data is processed, stored, distributed, etc.
- Hardware is moving to more cores, but less RAM per core.
  - Multi-threading introduced to share memory.
- HEP computing is strongly being pushed toward external computing resources
  - For Fermilab-related experiments, access to all resources are intended to happen through HEPCloud.
- HPC platforms
  - Think more in terms of parallel processes, and not necessarily multi-threading.
  - *art*, and projects related to it, have been exploring this for a few years (e.g.):
    - Exploring alternatives to ROOT
    - Moving away from UPS to Spack, which was designed for HPC systems

Near-term/imminent plans (2018-2019)

- Help experiments upgrade to *art* 3 to realize multi-threading benefits.
- Move toward non-event level parallelism.
- Continue to prune library dependencies from core framework infrastructure:
  - ROOT is now no longer part of the core framework
  - Various Boost dependencies can be removed when C++17 is adopted
- Continue research with HDF5 file format:
  - HDF is the primary data-storage formats on HPC machines
  - Work done in collaboration with the HDF group over the last couple years
  - HDF5 analysis-format alternative for ROOT TTree (Handed over to NOvA functioning software they want to use now — July 9)
    - 42 TB data read/decompressed in under 20 seconds on Cori (1200 KNL nodes) in Python.
- Continue research of HEPnOS distributed file system:
  - Work done in collaboration with Argonne National Laboratory

Improvements to module scheduling (2019-2020)

- *art* assembles the graph of data dependencies between modules.
- Data-dependency errors are caught at job start-up time, just after module constructors have been called.
- We do not yet use the graph to optimize event processing.
  - We intend to do so.



Middle-term R&D (2020-2022)

- Working toward the ability to use HDF5/HEPnOS as an alternative I/O system in *art*
- Extend *art* concepts to better support intra-module parallelism
  - Experiments like DUNE may not be able to afford multiple events in flight at the same time
  - Concurrently process independent portions of an event
- Is the event the right abstraction?
  - We are considering how to support user-defined levels of data aggregation.
- Work done during this time informs long-term R&D

Long-term R&D (2023-)

- Exploring in what ways our event model is compatible for single-node, grid, and HPC jobs.
- The intent is that the module authors' views of data would not change; the workflow orchestration, however, may be largely different.
  - Think solely in terms of datasets instead of files
  - Workflow details are subsumed by the framework
- SciDAC work is directly looking at an implementation for HPC jobs:
  - http://computing.fnal.gov/hep-on-hpc/
  - Addresses data-store aspect of distributed, hierarchical data

A few more remarks

- Many of the *art*'s concepts have been refined over the last few years
- In developing the framework, our *modus operandi* has been:

  No framework code is so precious that it must be saved; nothing is untouchable.

  Our users' code is precious, and we should break as little of it as possible.
- Much of the code internal to the framework has been replaced, without imposing many breaking changes on users.
- Perhaps surprisingly, though, many users *are* willing to move to different ways of thinking.
- It is this flexibility that makes it possible to explore **and use** new technologies.