

artdaq:
*An Event Filtering Framework
for Fermilab Experiments*

Marc Paterno
Scientific Software
Infrastructure Group
RT 2012
June 12, 2012



Fermi National Accelerator Laboratory

 Office of Science / U.S. Department of Energy

Managed by Fermi Research Alliance, LLC

What is *artdaq*, and why did we make it?

artdaq is a **prototype toolkit** for creating the **event building and filtering** portions of a DAQ.

- Traditionally, HEP experiments have developed custom event building and filtering software mostly **from scratch**; there has been little sharing of effort.
- The TeVatron experiments (CDF and DØ) were large enough to afford the level of effort needed.
- Currently planned, and future, experiments have fewer collaborators and **smaller budgets**.
- Our group has had success in solving a similar problem with **offline** software for these experiments (**art**): we wish to try extending our recent success.



Goals for *artdaq*

- Allow use of **commodity computers** as close to the data as possible.
- Make efficient use of **multi-core computers**.
- Take advantage of **high-speed networking** and hardware buses.
- Support modular algorithms, enable use of GPGPUs.
- Enable more collaborators to contribute to online code development.
- Produce an environment for our own R&D tasks.



Maximize single-node event-building capacity

- **Affordable and common modern networking** supports high data rates (*e.g.*, the supercomputing community).
- Modern commodity computing hardware is **multicore**, and accelerators (*e.g.*, GPGPUs) are cheap.
- Shared-memory parallelism (multi-threading) is increasingly important, but hard. Distributed parallelism is also important, and hard.
 - Most physicists will not devote time to developing expertise in parallel programming.
 - Tools to help simplify use of both of these things are needed.

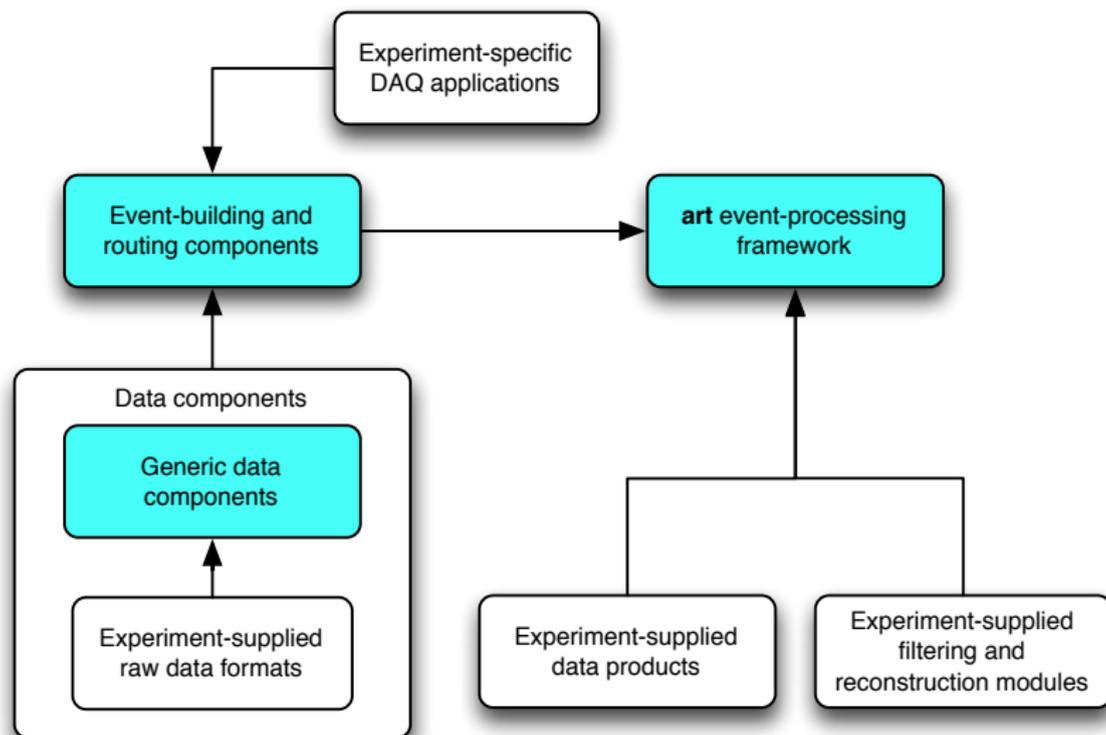
Simplify multi-node event building

- Most HEP detectors are read out through multiple DAQ front-ends, each responsible for a segment of the full detector.
- **Event building** is the process of assembling the many pieces of readout into a single whole.
- Event building often requires the coordinated work of several computing nodes.
- It is imperative to **communicate data efficiently and reliably** from data collection nodes to wherever the triggering or compression algorithms are run.
- Flexible reconfiguration of where processing is done greatly eases development.

Trigger algorithm issues

- Trigger algorithms often consist of many parts: unpack raw data, convert from electronics addresses to physics coordinates, apply calibrations, find hits, find tracks ...
- Experiments often want to run multiple trigger algorithms: looking for cosmic rays, neutrino interactions, magnetic monopoles, event supernova explosions ...
- Trigger algorithms should be flexible, *e.g.*, we want to adjust thresholds without recompilation.
- Development should be done in a familiar environment to most collaborators: share as much as possible between online and offline.
- Experiments need to verify algorithms in offline simulation.

The architecture of *artdaq*



- Only a few classes are needed to represent a wide variety of data: primary are `Fragment` and `RawEvent`.
- Fragments carry experiment data, as a **contiguous series of bytes**, as well as some routing information. Data can be of **arbitrary size**.
- `RawEvent` aggregates an **arbitrary number** of `Fragment`s.
- Different `Fragment`s can carry **different types** of data (e.g. detector readout, trigger blocks), and are clearly distinguishable.
- We take advantage of features of C++ 2011 to assure no unnecessary copying of data is done.

artdaq event building components

- **artdaq** event-building programs have **fragment receiver**, **event builder** and **event processor** layers.
- MPI used for coordination and communication in a multi-process distributed program; also uses multi-threading.
- `SHandles` and `RHandles` classes provided to simplifying sending and receiving of `Fragments` via MPI.
- `EventStore` class assembles `Fragments` into events, and passes them to another thread to be processed by the **art** framework: no copying of data involved.
- Orderly program shutdown when correct end-of-data markers are seen.



- **art** is used offline by several Fermilab experiments.
- Derived from the CMS framework, used in trigger.
- Executes **experiment-supplied** modules.
- Choice of modules is **runtime-configurable**, without recompilation.
- Modules create data products (*e.g.*, set of hits, set of tracks).
 - Provenance information is recorded for each created product (configuration information of the module that created the product, what other products were used for the construction).
 - Data products are persistable to **art** data files.
- Framework has monitoring points around module execution to allow event-by-event timing of each module, to identify uncontrolled memory usage, and to **identify where processing errors occur**.

What the experiment provides

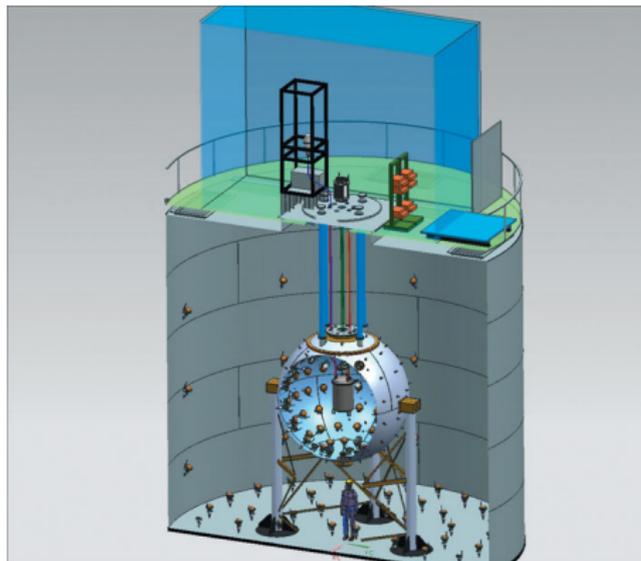
- Top-level DAQ application: **artdaq** is only a **toolkit**.
- Modules that inherit from **art** base classes, for reconstruction and triggering.
- Definitions of their own data product types, which have to conform some restrictions imposed by **art** (for persistency purposes—restrictions are relaxed for products that can not be persisted).



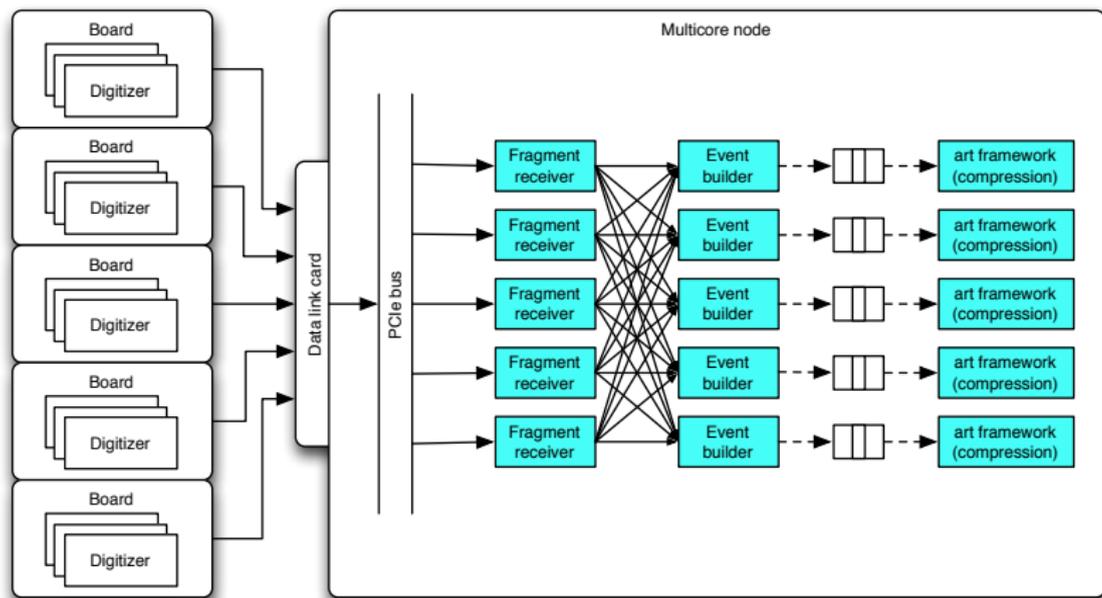
- Fast compression and high data rate at DarkSide-50.
- Mu2e multi-node event building.
- The NO ν A prototype Data Driven Trigger.

The DarkSide-50 experiment

- LAr time-projection chamber.
- Underground at Gran Sasso.
- Search for Dark Matter.
- 38 channels of 250MHz digitizers.
- Looking for ~ 10 events/year signal.
- Continuous data output of 300MB/s; practical considerations require reducing the data to 30MB/s.
- How inexpensively can we construct the event builder?

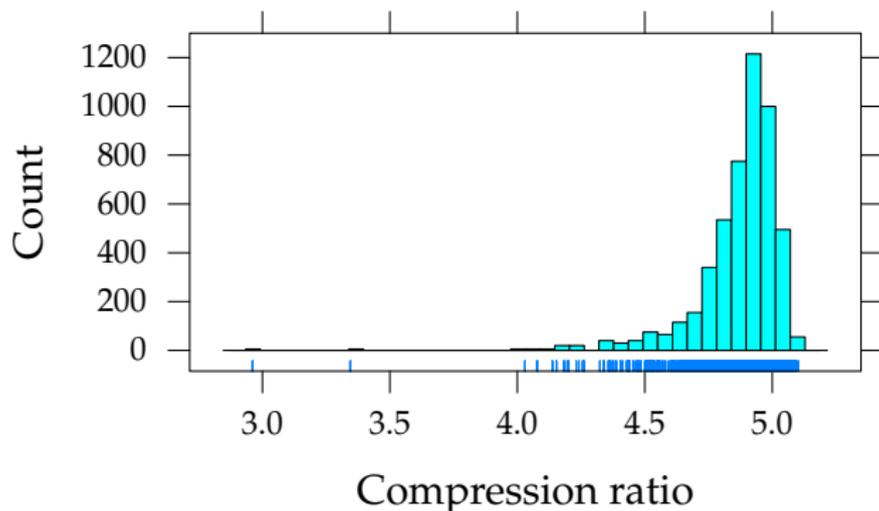


Prototype event builder for DarkSide-50



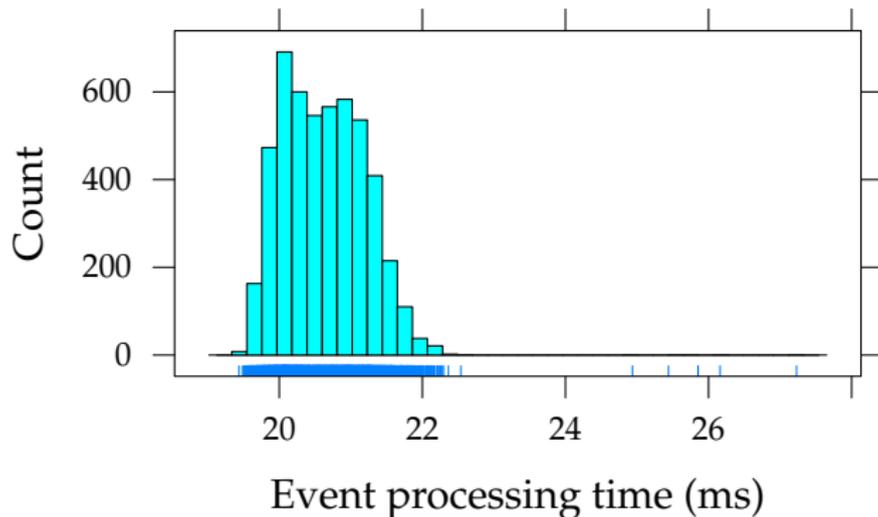
- Data link emulated by IB (\$6000, including NICs; use only 6/18 ports); detectors simulated by processes on 3 32-core nodes, communication through MPI over IB.
- **All event-building is run on 1 32-core node.**

Preliminary compression results



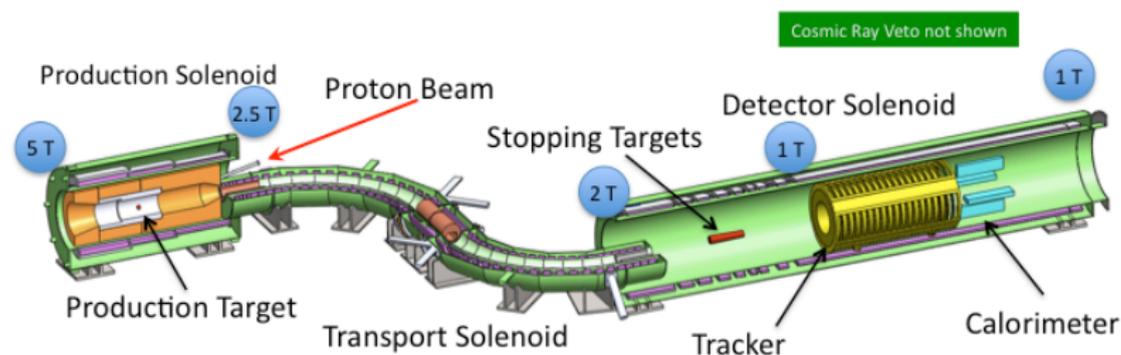
- Used Huffman coding, with 5 threads per **art**
- Other algorithms may do better.
- Identifying **signal regions** may do much better.

Preliminary throughput results



- Initial throughput is ~ 250 events/s, ~ 1.5 GB/s.
- There is **much more available parallelism** in the problem!

The Mu2e experiment



- Search for neutrinoless decay of muon to electron.
- ~ 275 front-end sources, $\sim 30\text{GB/s}$ of waveform data.
- **Filtering software must reduce this** to $\sim 30\text{MB/s}$.
- **How many commodity nodes** are required to handle this?
- Using **artdaq** of 5 32-core nodes and IB network, we have achieved $\sim 3\text{GB/s}$ throughput.
- We are working on understanding the scaling of the system.

The NO ν A experiment



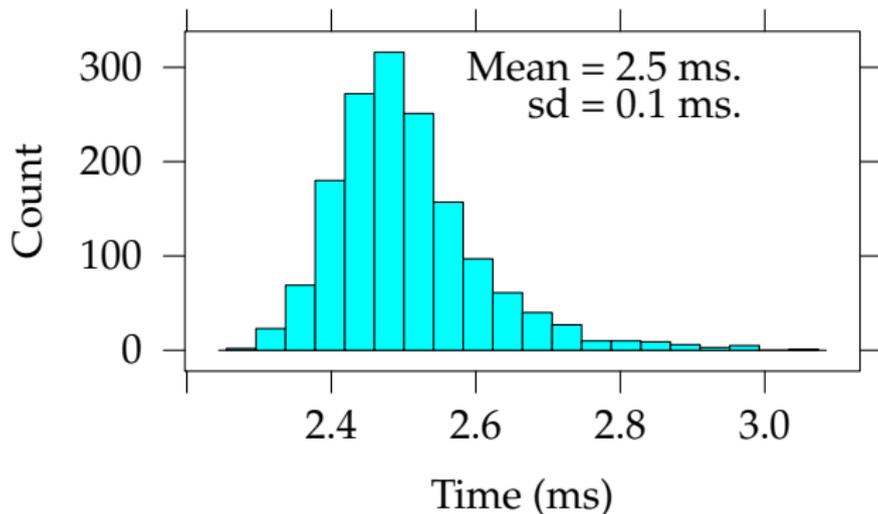
NO ν A Far Detector is 810km from Fermilab, at Ash River, MN.

NO ν A event building

- The Far Detector 385,000 readout channels, read through 180 data collection nodes, **custom designed upstream hardware** (the DCMs) delivers fragments of data in 5ms time slices to 180 multicore commodity buffering nodes using **gigabit Ethernet**.
- The system can sustain a raw data input rate of greater than 2GB/s, writing to shared-memory buffers from which triggered events are selected.
- We used **art** to read the live NO ν A data stream, to **demonstrate the portability of offline algorithms** to the online system.
- First physics algorithm implemented is a 2D Hough transform (not yet parallelized).
- **art** framework allows collection of some **performance metrics** without needing to instrument the algorithm.

Measured framework overhead in $NO\nu A$ DDT

- There is sometimes a concern that a “framework” will incur unacceptable overhead.
- We can measure the software and determine the results.
- The $NO\nu A$ time budget is 55ms; the overhead is $< 5\%$.



Future plans

Short term:

- tune of MPI use, study scaling for Mu2e event building
- finer-grained parallelism for DarkSide-50.
- investigate parallelization of NO ν A's trigger algorithm

Longer term:

- We are introducing event-level and sub-event-level multi-threading into **art**.
- Implementation of some algorithms (*e.g.*, Hough transform) in CUDA for GPGPU.
- Development of FPGA/PCIe based data generator for better simulation of experimental data.
- We are starting to talk with additional experiments (μ BooNE, muon g-2) to evaluate their needs.

- We have demonstrated that we can build, using commodity hardware, an event-building system for a modern experiment (DarkSide-50).
- Using tools (MPI) from the HPC community, we have been able to quickly move forward to building fully-configurable distributed multi-process programs, without having to write any low-level code.
- We have demonstrated portability between offline and online software for testing and debuggability.
- We have established an environment in which we can carry on with our R&D tasks.