



Introduction to *artdaq*

SBN Electronics/DAQ Meeting
09-Dec-2014

Kurt Biery

(On behalf of the SCD Real-time Software Infrastructure group: Ron Rechenmacher, Gennadiy Lukhanin, John Freeman, Eric Flumerfelt)

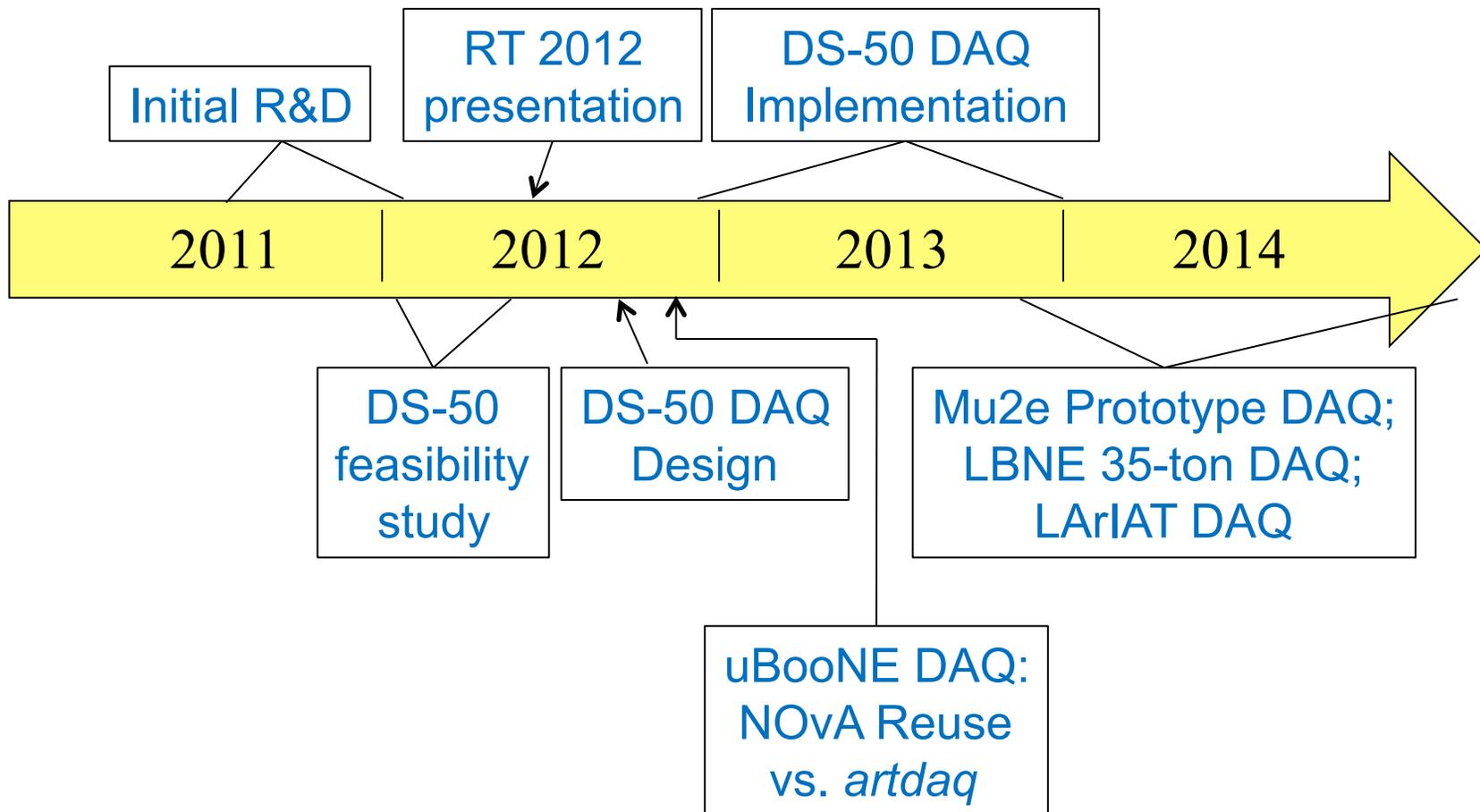
What is *artdaq*?



artdaq is a **toolkit** for creating **data acquisition systems**

- Fermilab Scientific Computing Division's core DAQ software
 - Provides common, reusable components
 - Based on a data-streaming architecture with software event filtering
- Integrated with the *art* framework
 - Same algorithms online and offline
- It provides data transfer, event building, process management, system and process state behavior, control messaging, message logging, infrastructure for DAQ process and *art* module configuration, and writing of data to disk in ROOT format.
- The goal is to provide the common, reusable components of a DAQ system and allow experimenters to focus on the experiment-specific parts of the system. These are the software that reads out and configures the experiment-specific front-end hardware, the analysis modules that run inside of *art*, and the online data quality monitoring modules.

Timeline



Current Users



DarkSide-50

- First production use of *artdaq*
- CAEN VME modules, commodity computers
- DAQ running reliably; stable operation

LBNE

- 35t prototype detector running early 2015
- Vertical slice testing of HW and SW now
- Strong physicist involvement on *artdaq* customizations
- We're developing the *artdaq* interface to Run Control and contributing to configuration management

Mu2e

- Pilot system under development now
- Commercial PCIe cards, commodity computers
- 30 GB/sec, filtering factor of ~1000

LArIAT

- *artdaq* used on top of original DAQ code
- Expected to resume data taking soon

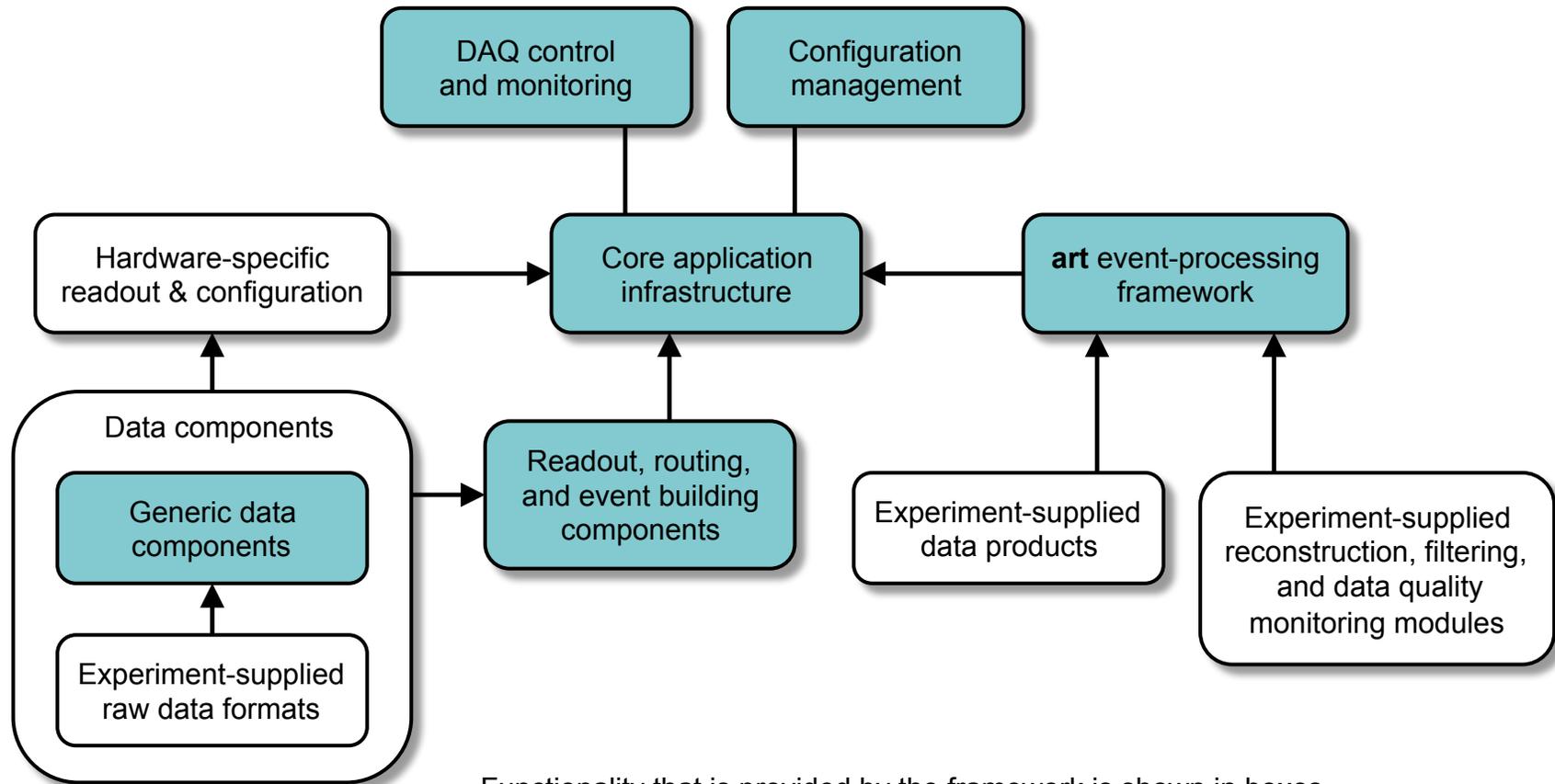
NOvA

- *art* for enhanced triggering (data-driven triggers)
- A few pieces of *artdaq* for *art* input

uBooNE

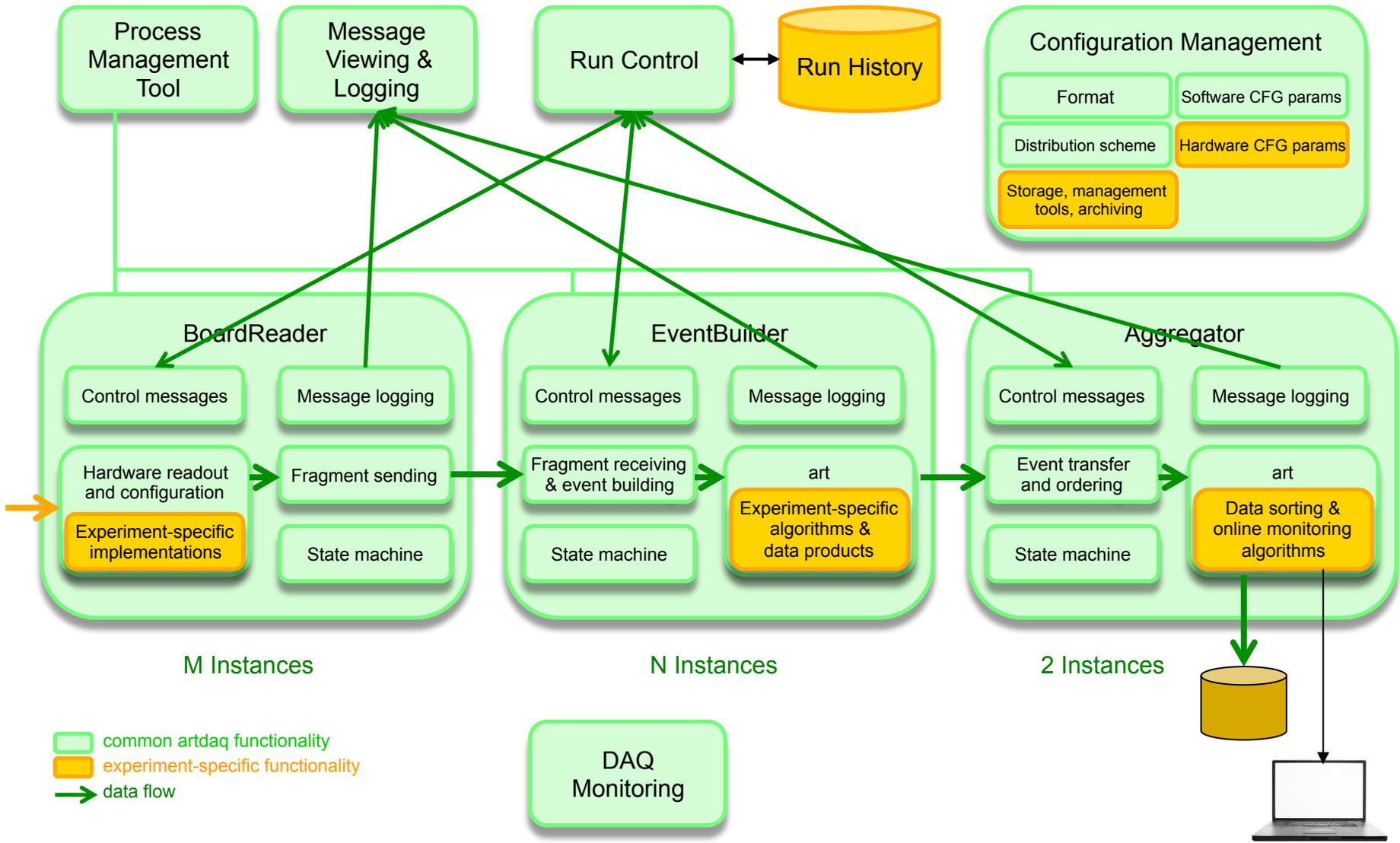
- Uses a number of *artdaq* utility classes

Architecture



Functionality that is provided by the framework is shown in boxes with shaded (blue) background. Functionality that is provided by the experiment is shown in boxes with white background.

Software Components & Functions



Some Technical Details



Uses C++11 features

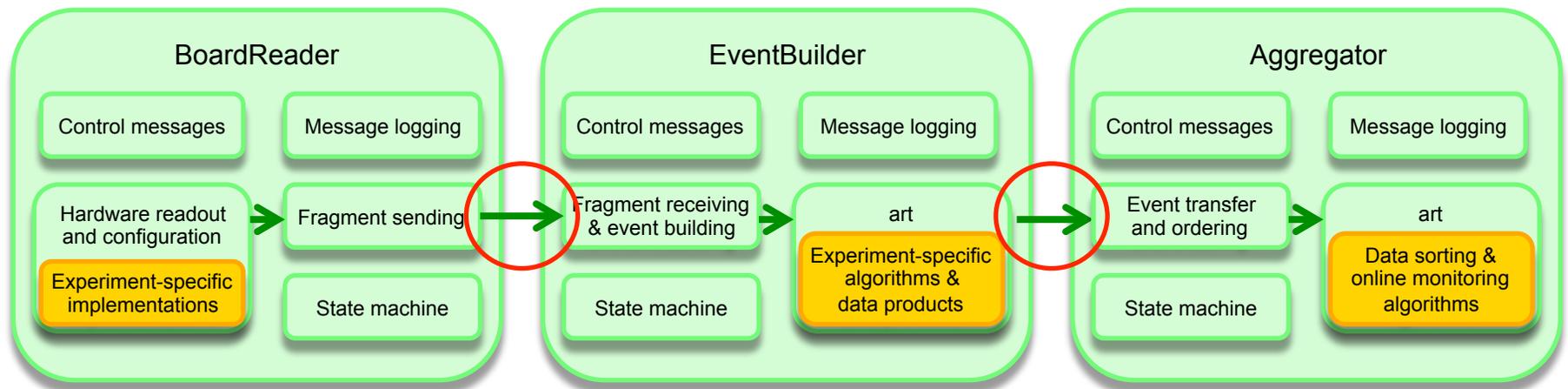
- e.g. move semantics to minimize data copies

MPI for data transfer

- Wrapper classes for sending and receiving MPI buffers

Process management

- Wrapper script around *mpirun* command



State Behavior & Control Messaging

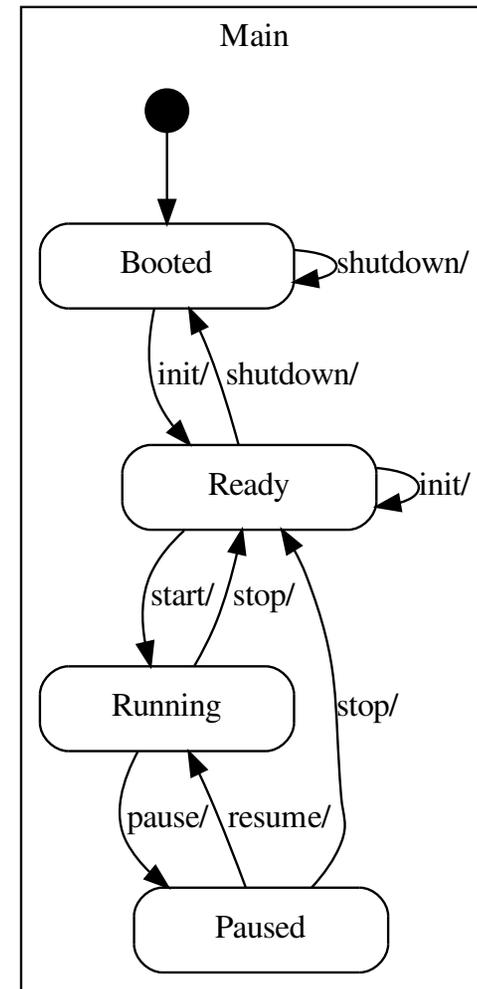


State behavior part of *artdaq* core

- Standard states:
 - Booted, Ready (configured), Running, Paused
- Standard transitions:
 - Init(cfgString), start(runNum), stop, pause, resume, shutdown
- State Machine Compiler tools
- Overall system state managed by Run Control

Control messaging infrastructure also included

- Standard commands:
 - State transitions & status queries
- XMLRPC
- Command-line Run Control scripts provided



Simplified *artdaq* state diagram

Configuration Management



Configuration infrastructure part of core *artdaq*

- Single structured text string sent at “init” transition
 - hardware, software, system, *art* configuration
- Interpretation of software and system parameters provided in core *artdaq*
- Hardware parameters handled by experiment-specific code
- Archiving and management of configuration data – not yet part of core *artdaq* – but may be in the future

```
daq: { # red = group identifiers
  max_fragment_size_words: 2097152
  fragment_receiver: {
    mpi_buffer_count: 40
    generator_ds50: {
      fragment_id: 2
    } # green = software params
    generator: V1495Driver
    first_event_builder_rank: 3
    event_builder_count: 5
    rt_priority: 2 # system params
    link_type: PCIE
    link_number: 0 # hardware params
    vme_base: 0x01000000
    pulser_frequency: 0.0
    laser_frequency: 0.0
    random_triggers: false
    acquisition_gate_us: 400.0
    low_threshold: 5
  }
}
```

Sample *artdaq* BoardReader configuration (FHiCL)

Typical Analysis Tasks (*art*)



Data compression

- DarkSide-50 – factor of 5-6 reduction

Online monitoring

- DarkSide-50

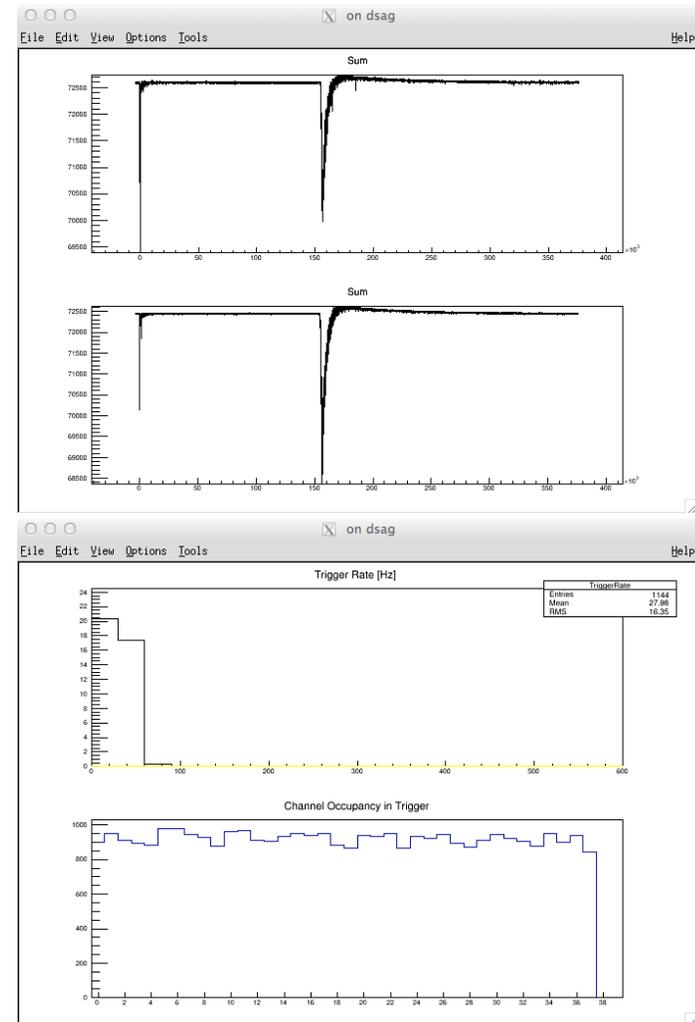
Event filtering

- Mu2e – reduction by factor ~ 1000

Event selection

- NOvA – data driven trigger

Same code online as offline



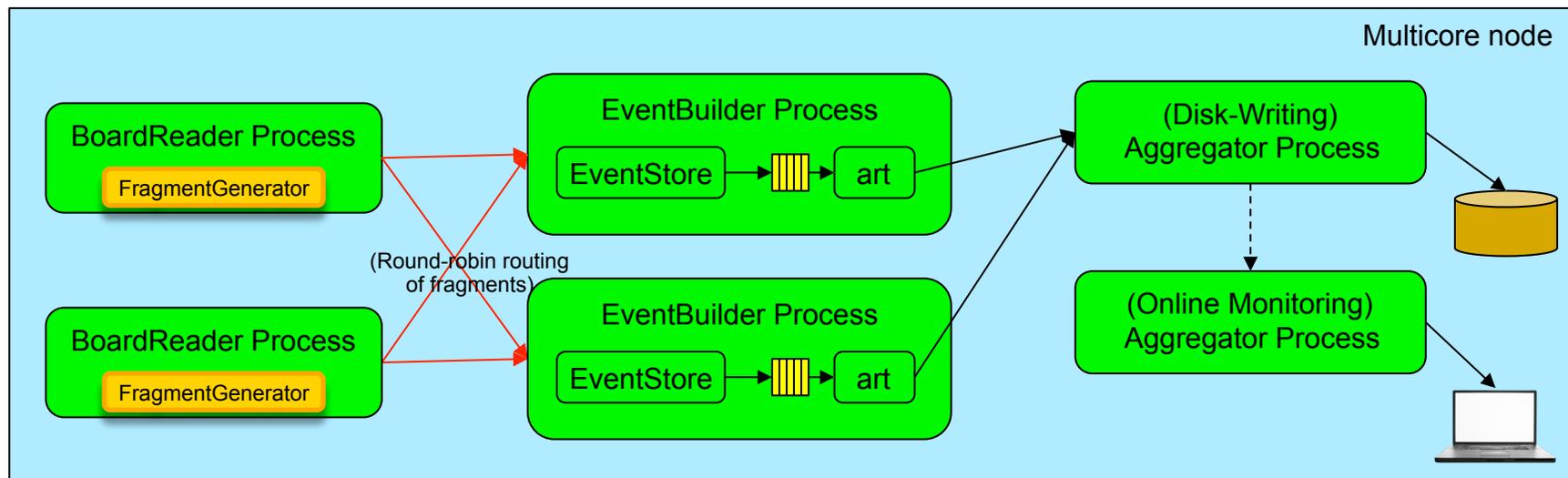
Sample DS-50 online monitoring histograms

The *artdaq*-demo



Demo package to illustrate *artdaq* use

- Instructions for downloading, building, and running a sample system
- More information here:
 - <https://cdcv.s.fnl.gov/redmine/projects/artdaq-demo/wiki>
- An easy way to try out *artdaq* and learn more about it

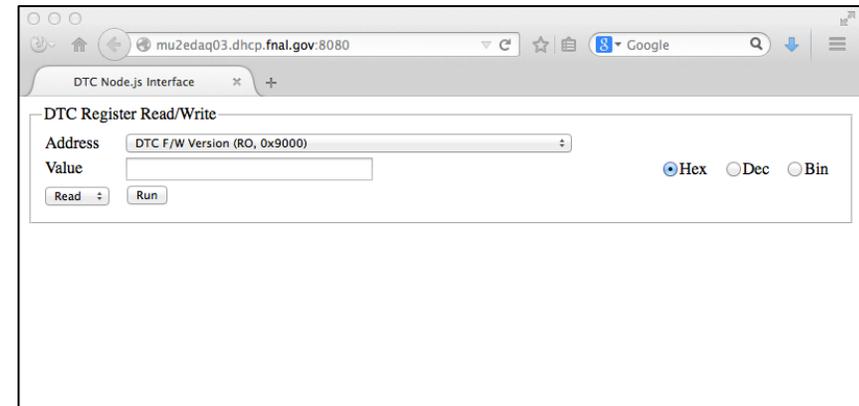


Current and Future Work



In collaboration with SCD engineers

- Investigate the use of IoT hardware for back-end DAQ
- Develop user interfaces for electronics hardware and firmware diagnostics and testing



Core *artdaq* enhancements

- DAQ monitoring
- Run Control GUI
- Improved online monitoring scheme
- Multi-level system for software triggering (compared to filtering)

Continued partnerships with experiment DAQ groups

Backup slides



artdaq for DarkSide-50



DAQ

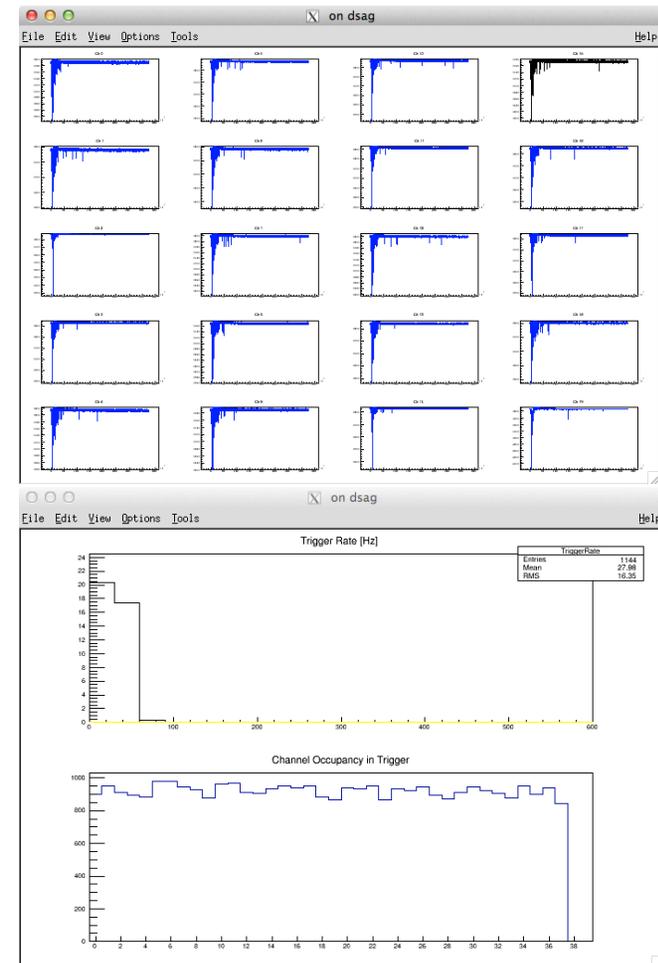
- Commercial VME modules (CAEN)
- Commodity servers and networking
- *artdaq* software (first production use)
- Typical event size ~ 10 MB
- Event rates up to 80 Hz

Customizations for Dark-Side

- Configuration and readout of VME modules
- Loss-less data compression
- Online monitoring

Status

- Reliable operation
- Meeting performance needs



artdaq for LBNE



Timeline

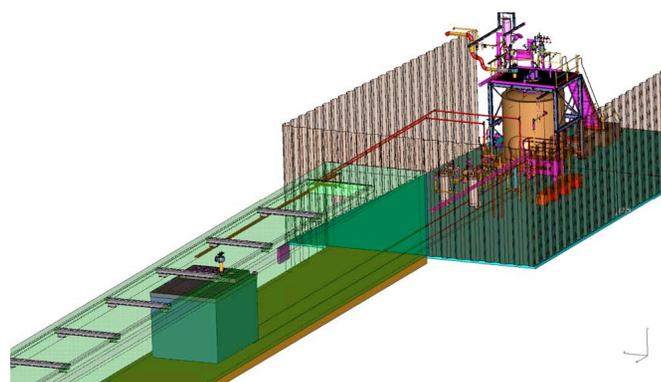
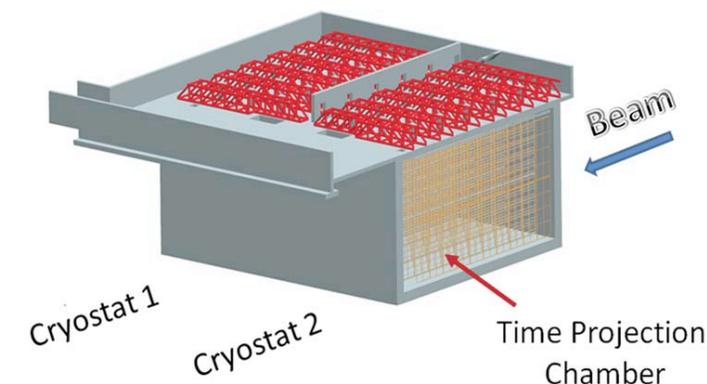
- 35 ton prototype – early 2015
- Full detector – 202x

DAQ

- Custom front-end electronics
- SLAC RCEs for front-end readout and possible zero-suppression
- *artdaq* software
- Data rate ~4 GB/s
- 2400 connections to the front-end electronics

Integration with existing components

- Run Control from IceCube
- Configuration management from NOvA



artdaq for Mu2e



Timeline

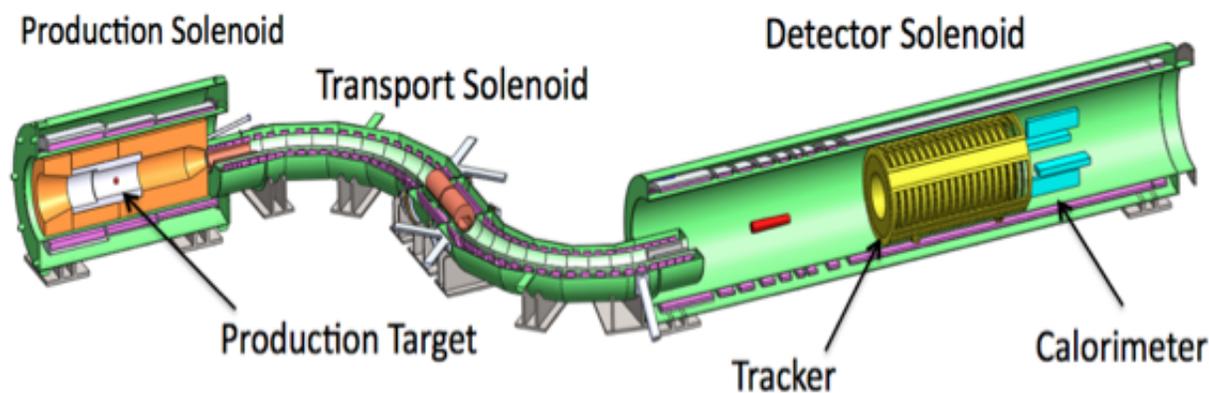
- CD-2/3 approval July 2014
- Commissioning data in 2019

Mu2e-specific development

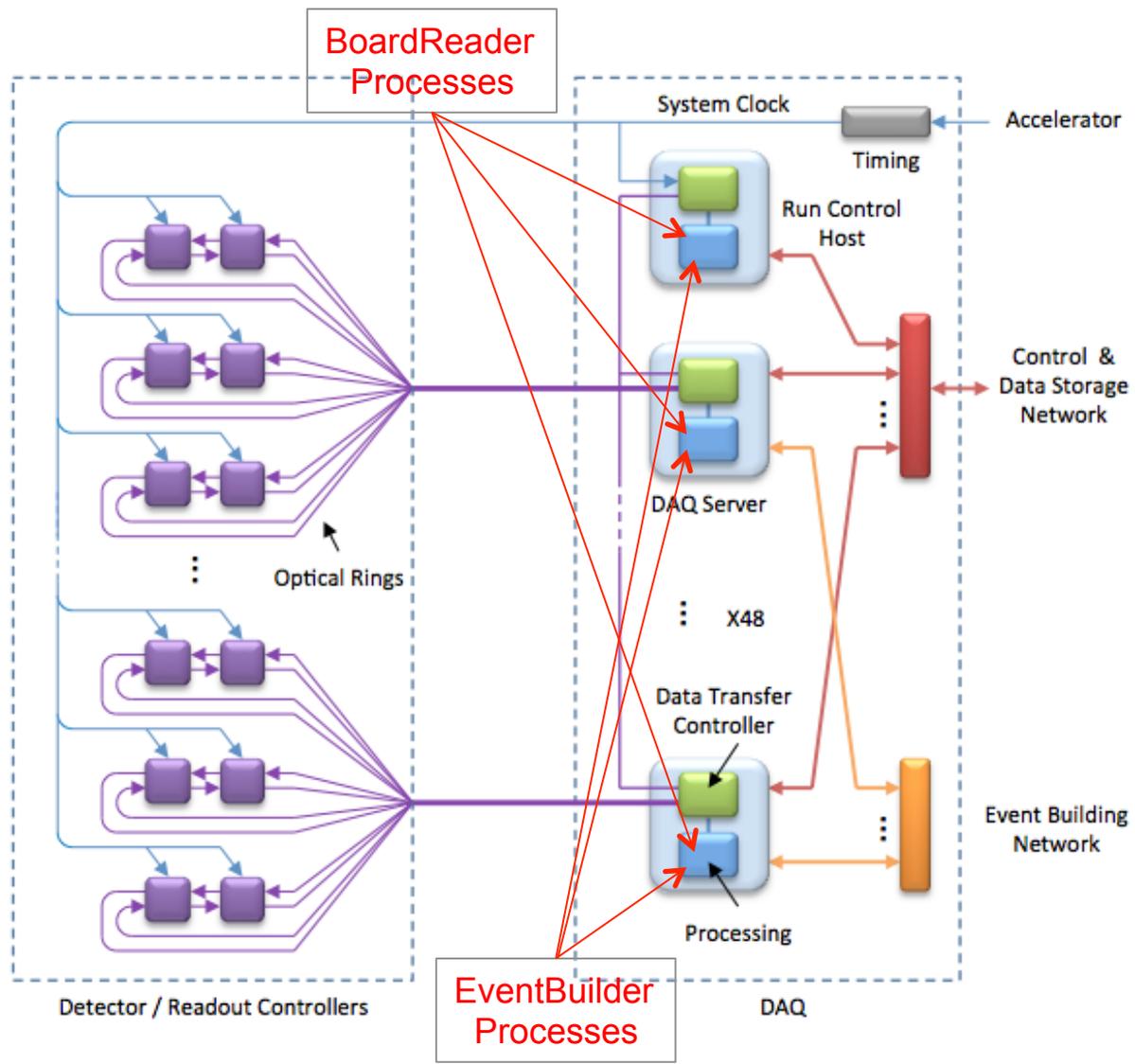
- Configuration and readout of hardware
- Reconstruction algorithm development and testing

DAQ

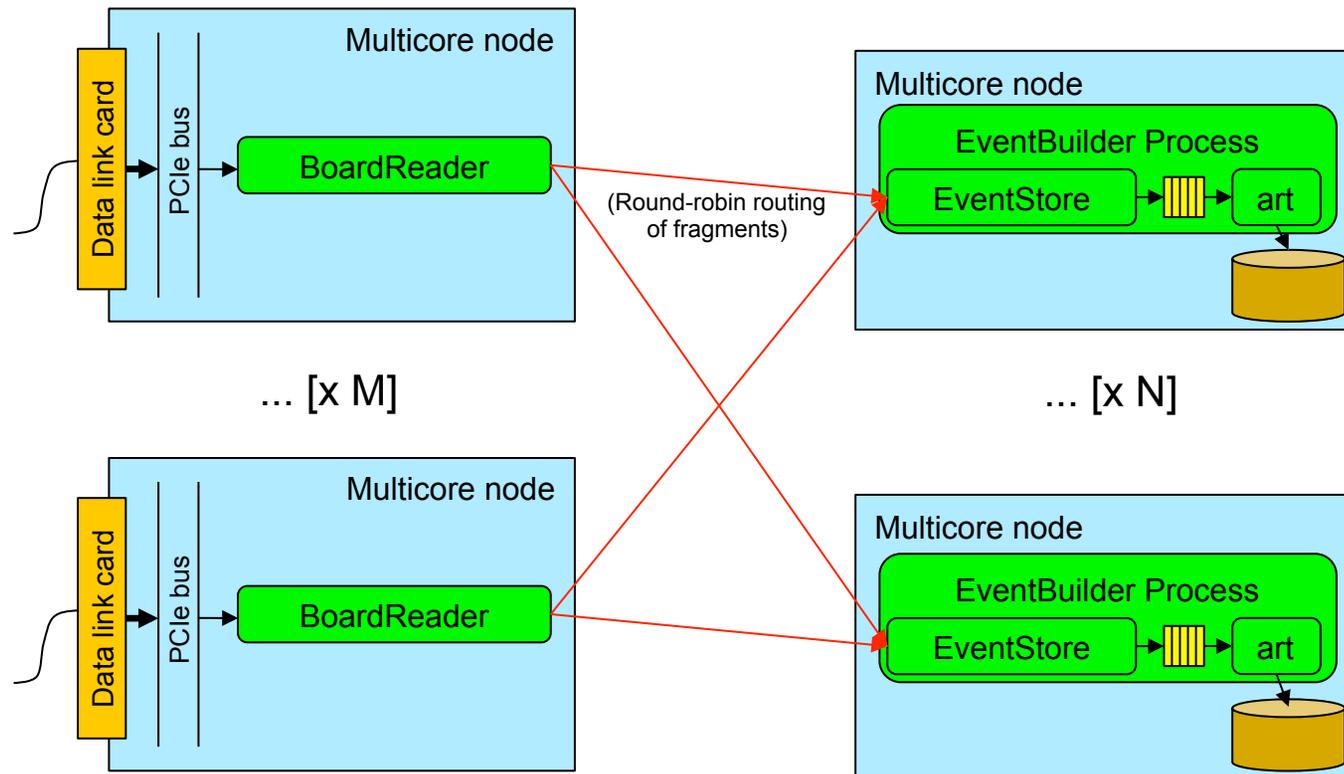
- Custom readout electronics
- Commodity servers and networking
- *artdaq* software & custom firmware
- Expected data rate: 30 GB/s
- Software rejection factor: 1000



artdaq for Mu2e



Generic DAQ



Lots of variations:

- multiple fragment receivers per front-end node
- multiple event builder/**art** process pairs per reconstruction node
- (multiple **art** processes per event builder)
- everything run on a single node

A flexible configuration process makes testing and deployment easier.

MPI and mpirun



In *artdaq*, MPI (Message Passing Interface) is used to transfer data between the distributed processes and to manage the processes.

- MPI is “a library specification for message passing, designed for high performance on parallel machines and workstation clusters.”
- It supports point-to-point and collective messaging. It also supports parallel execution features such as synchronization between processes (e.g. all process wait until they have all reached a certain point in their execution).
- An MPI “program” contains all of the cooperating processes, and startup is handled by an agent (*mpirun*) that runs the program on a configurable set of nodes.
- In MPI-1, the same executable binary is used for all processes, and the different processes know which role to perform based on their “rank”.
- For the initial *artdaq* test system, we wrote a straightforward Python script to automate the configuration of the nodes and handle the invocation of *mpirun*.

Sample DS-50 *artdaq* Deployment

