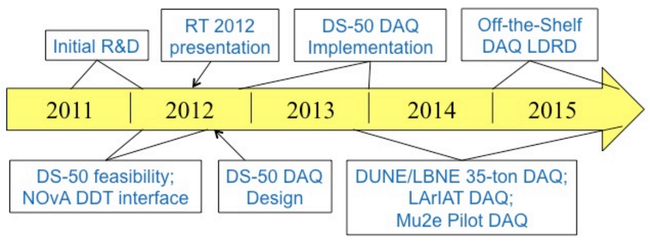
Recent Developments in the Infrastructure and Use of *artdaq*

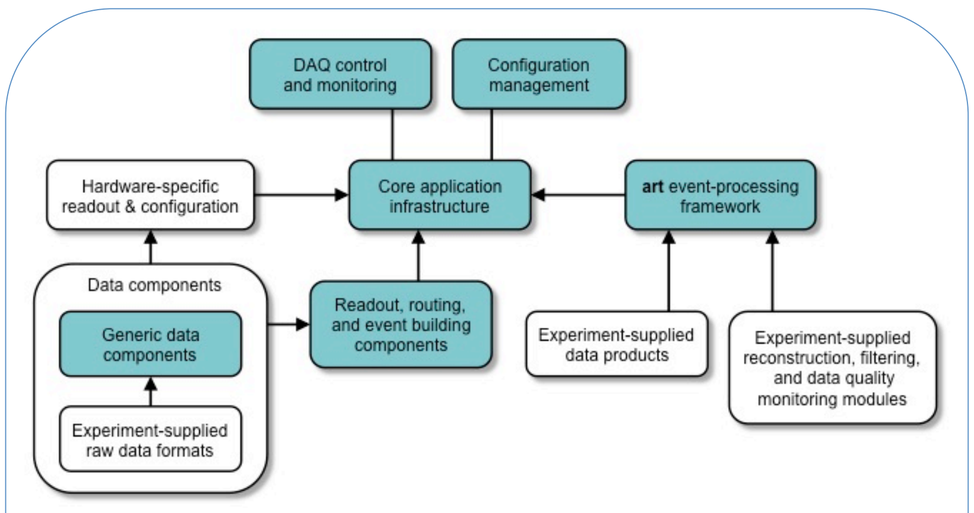
K.Biery, M.Bowden, E.Flumerfelt, J.Freeman, A.Prosser, R.Rechenmacher, R.Rivera
Fermi National Accelerator Laboratory

The *artdaq* Data Acquisition Toolkit

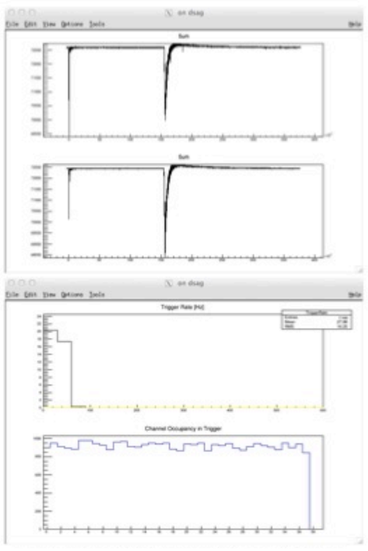
artdaq is a data acquisition software toolkit that has been developed at Fermilab, and it is being used by a growing number of high-energy and cosmology experiments. It currently provides data transfer, event building, run control, and event analysis functionality. The event analysis functionality is provided by the *art* framework which is used for offline reconstruction and simulation by many experiments at Fermilab.



Timeline of *artdaq* development and use.



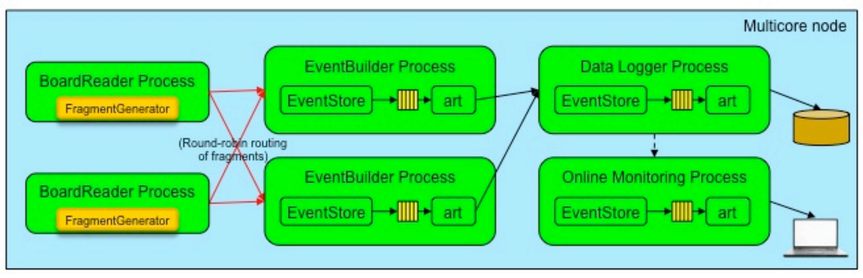
artdaq architecture: Core functionality is provided by the framework, and experiment-specific components are developed by experimenters. Common functionality is shown in boxes with blue background and experiment-specific components are shown in boxes with white background.



artdaq online monitoring: Sample DarkSide-50 data quality plots.

artdaq demo:

- An easy way to try out *artdaq* and learn more about it
- https://cdcvs.fnal.gov/redmine/projects/artdaq-demo/wiki
- The Wiki pages contain instructions for downloading, building, and running a sample system

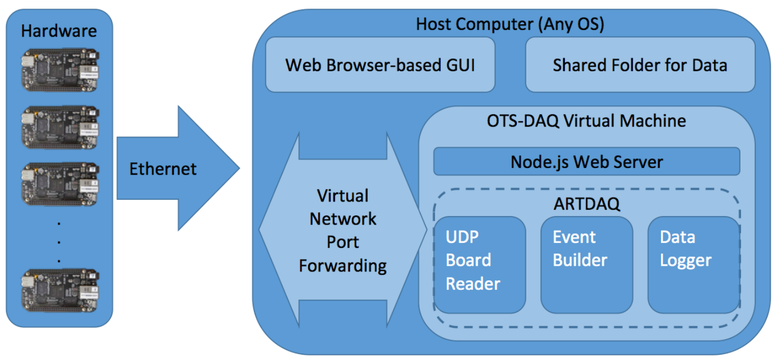


artdaq components that are included in the sample *artdaq-demo* system.

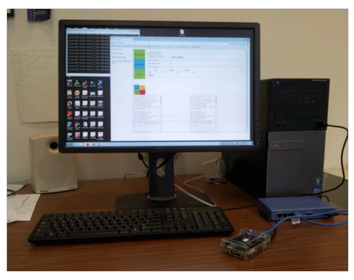
Off-the-Shelf Hardware and Software

A Fermilab LDRD (lab-directed R&D) project is underway to evaluate a low-cost, scalable data acquisition system architecture based on commercial technology developed for the "Internet of Things". This approach connects intelligent front-end digitizers directly to a standard network without additional layers of custom readout electronics. The software will be based on *artdaq* and *art*, and the goal is to provide an off-the-shelf solution that provides basic DAQ hardware and software functions. The system will be scalable from a few MBytes/sec to hundreds of GBytes/sec using inexpensive commodity networking equipment and interface modules.

Commercial hardware modules are currently being evaluated. The first integration milestone is to transfer UDP packets from a BeagleBone Black board to an *artdaq* system running in a VirtualBox virtual machine on a Windows PC.

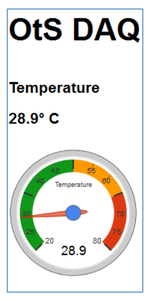


Block diagram of the integration test setup.



Integration test setup: BeagleBone Black board, PC, and *artdaq* software.

Mock-up of web interface for users to select DAQ components.

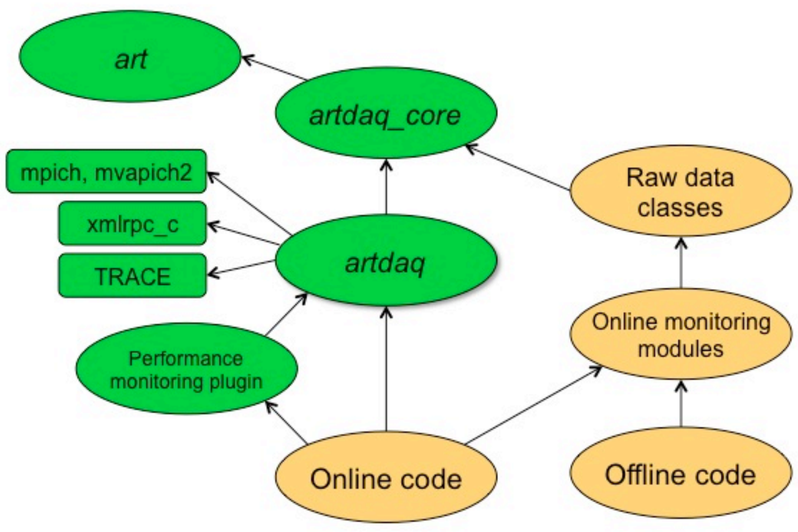


Sample web display of BeagleBone Black data.

Software Organization

artdaq-based (and *art*-based) software packages are distributed as tar files. The versions of each of the packages that are used during data taking, and the dependencies between packages, are managed with the UPS (Unix Product Support) tool developed at Fermilab.

To avoid unnecessary dependencies between online and offline software, code that is used by both is best organized in separate packages that can be used by each of them. To support this, the C++ classes in *artdaq* that are needed in both online and offline environments are distributed in a special package (*artdaq-core*). With this separation, only an experiment's online code needs to depend on the *artdaq* software package.



This diagram shows a representative set of dependencies between software packages. The direction of each arrow indicates the dependence of the source package on the target. (e.g. *artdaq* depends on *artdaq_core*.) Packages that are shown with a green background are part of the *artdaq* software suite, ones with orange background would be developed by an experiment.

The organization that is shown here avoids a situation in which an experiment's offline code needs to depend on its online code, or vice versa.

Software Version Tracking

In order to provide a convenient way for experimenters to determine which versions of various software packages were used during the data taking for a particular data set, *artdaq* provides the ability to record this information in the raw data files. To enable this feature, experiments create a few simple C++ classes based on examples provided in *artdaq*, and configure their system to make use of these classes.

An *art* module is available to print out the information that was stored in a raw data file. Here is sample output from this module:

```
Run 411 begin time = 2015-Apr-06 15:20:30 UTC, end time = 2015-Apr-06 15:34:01 UTC
----------------------------------------------------------------
Package             |Version            |Timestamp (package build time)
artdaq-core         |v1_04_10           |27-Feb-2015 16:15:54 UTC
artdaq              |v1_12_08           |06-Apr-2015 14:21:10 UTC
darksidecore        |v1_00_03           |04-Mar-2015 19:15:47 UTC
darkmon             |v1_00_05           |04-Mar-2015 19:16:34 UTC
ds50daq             |v1_04_00           |29-Mar-2015 11:34:07 UTC
----------------------------------------------------------------
Subrun 3 begin time = 2015-Apr-06 15:29:30 UTC, end time = 2015-Apr-06 15:34:01 UTC
```

Debugging with TRACE

To support system-level, process-level, and thread-level debugging of *artdaq*-based systems with minimal impact on the performance of the system, the TRACE software package is now distributed as part of the *artdaq* software suite. TRACE was developed at Fermilab, and it allows developers to add log messages to their code, disable some fraction of the messages at runtime, and configure message destinations at runtime. Supported destinations include a shared memory buffer (minimal performance impact), a log file, and the console.

Each TRACE message contains a microsecond-level timestamp, information about which system thread generated the message, and information about where in the system the process was executing (e.g. which CPU core).

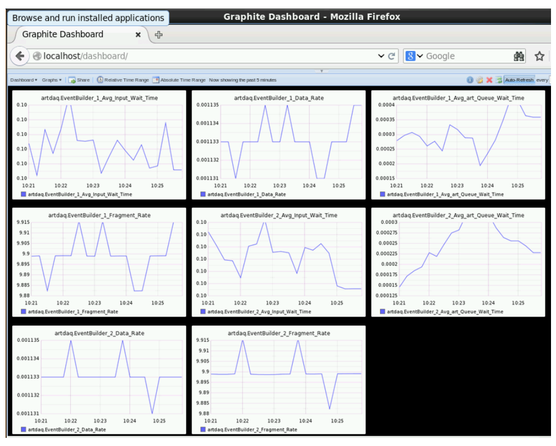
DAQ Performance Monitoring

In order to ensure that each *artdaq*-based DAQ system is performing well, the toolkit provides infrastructure for reporting performance metrics. In addition, it includes the reporting of many basic parameters that demonstrate the performance of *artdaq* itself.

This functionality is implemented as a generic "metric reporting" interface in the *artdaq* code, and a set of modules that can be "plugged into" the metric reporting framework to provide the visualization of the performance data.

In addition to a plug-in for reporting metrics to log files, interfaces to Ganglia and Graphite have been developed.

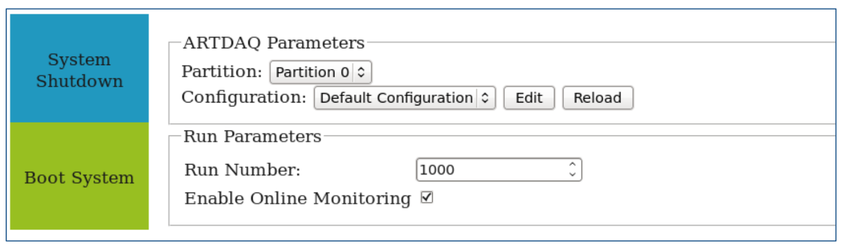
Sample Ganglia plots showing performance of an *artdaq*-based DAQ system. The plots shown contain performance metrics for the process that writes the data to disk. Additional plots are available for the processes that read out the front-end hardware and that build the events. Each of the six data-taking runs shown in these plots used a different ADC gate width, and this accounts for the changes in the event size and event rate.



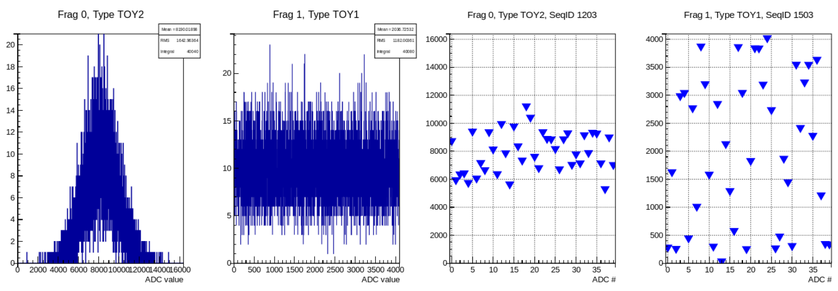
Sample Graphite plots showing performance of an *artdaq* demo system. The plots shown contain performance metrics for all of the process in the small test system.

Web-based Control and Monitoring

Prototype web-based tools are being developed using a Node.js web server, Javascript in the browser, and third-party Javascript libraries.



Web-based graphical Run Control interface. JavaScript in the browser interacts with the Node.js web server to send control messages to the *artdaq* processes and receive status updates in return.



Web-based online monitoring plots. JavaScript in the browser interacts with the Node.js web server to display the ROOT histograms that are generated in the *artdaq* online monitor process.