Journal of Physics: Conference Series **664** (2015) 032026 doi:10.1088/1742-6596/664/3/032026

A Review of Event Processing Frameworks used in HEP

E Sexton-Kennedy[1]
[1]Fermilab, P.O.Box 500, Batavia, IL 60510-5011, USA

E-mail: sexton@fnal.gov

Abstract. Today there are many different experimental event processing frameworks in use by running or about to be running experiments. This talk will discuss the different components of these frameworks. In the past there have been attempts at shared framework projects for example the collaborations on the BaBar framework (between BaBar, CDF, and CLEO), on the Gaudi framework (between LHCb and ATLAS), on AliROOT/FairROOT (between Alice and GSI/Fair), and in some ways on art (Fermilab based experiments) and CMS' framework. However, for reasons that will be discussed, these collaborations did not result in common frameworks shared among the intended experiments. Though importantly, two of the resulting projects have succeeded in providing frameworks that are shared among many customer experiments: Fermilab's art framework and GSI/Fair's FairROOT. Interestingly, several projects are considering remerging their frameworks after many years apart. I'll report on an investigation and analysis of these realities. With the advent of the need for multi-threaded frameworks and the scarce available manpower, it is important to collaborate in the future; however it is also important to understand why previous attempts at multi-experiment frameworks either worked or didn't work.

1. Introduction

Since there have been event processing framework reviews in the past, it is legitimate to ask, why review them now? It is generally accepted that we are in the midst of a second paradigm shift from single threaded applications to multi-threaded applications, motivated by industry trends[1]. The first shift was the move from Fortran based applications to C++ based ones. At that time a number of new event processing frameworks were developed to help experiments meet this challenge. This second shift promises to be even more disruptive then the first, which is why it is beneficial to reflect on, and learn from that past experience, as well as examine the framework software status of the field now.

2. Framework Collaborations

2.1. A Case History of Framework Collaboration: The BaBar, CDF, and Cleo Collaboration

During the 1997 CHEP conference a number of us from BaBar, CDF, and CLEO got together and decided to collaborate on our existing event processing frameworks. For CDF and BaBar this was easier since we had started from the same code base and it was really just a matter of remerging the fork between our two respective repositories. With Cleo this collaboration was more about exchanging ideas then exchanging code. The conference spawned several follow up meetings in San Francisco and FNAL that moved the collaboration forward and for BaBar and CDF included a code merge. The experience was written up and presented at the 1998 CHEP conference[2]. The lessons learned are still applicable today, and bear repeating here:

1. It is possible to assemble a group of experts interested in infrastructure development from different experiments to join forces on their work with benefits to everyone.

2. The importance of early joint design sessions between different experiments can not be stress enough. Maintaining constant communication is also paramount.
3. The most influential factor in the sharing success between CDF and BaBar was that both of these experiments use the same file organization and building system called Software Release Tools (SRT) and the same code management system, CVS.
4. In a collaboration like this it is key to be using the same foundational classes, and external dependencies.

While the paper concludes, "The impact of this multi-experiment collaboration is found to be positive." it did not last for the entire running period of the experiments lifetime. Why?
Once BaBar started taking data in 1999, CDF and BaBar stopped synchronizing their repositories. The pressure of taking data and the need for the quick fix in the middle of the night make collaboration take a back seat. The tools available in the '90s were not adequate to the task of supporting a geographically diverse multi-stakeholder project. A trusted cloud service, like GitHub that could host a shared repository and make changes instantly viewable and testable in a continuous integration system, did not exist then. Despite the fact that the collaboration went inactive the initial efforts to collaborate still payed dividends. When BaBar was forced to migrate from RogeWave to the STL they benefited from the adaptor classes written for CDF. The people who migrated from BaBar to CDF and back, appreciated the fact that they did not have to learn a new event processing framework.

2.2. Active Collaborations Today

There are a number of active collaborations today. ATLAS and LHCb collaborate on Gaudi and support a wider user community including GLAST, HARP, DayaBay, and MINERvA. ALICE and GSI/Fair collaborate on AliROOT/FairROOT and their users include Panda, Cbm R3B MPD, ASYEOS EIC. These collaborations are sharing code and participate in meetings together on a regular basis. There is also a collaboration between CMS and the supporters of *art* that is similar in nature to the older collaboration between CDF and CLEO, where ideas and designs are shared but not the code. There are many users of *art,* Mu2e, Muon g-2, NOvA, MicroBooNE, LAriAT, Darkside-50, and DUNE. Representatives from these experiments meet with the developers in a weekly "stakeholders" meeting to discuss feature requests and bug reports[3].

An interesting thing to note about these groupings is their geographical clustering. In addition, they have similar physics interests, for example, *art* for the neutrino experiments, FairROOT for heavy ion physics. These factors have contributed to the success of these collaborations, which have existed for many years. Clearly being able to meet face to face on a regular basis facilitates this success, but do the different physics domains really require different event processing frameworks? or is this clustering more the result of human factors; like people know each other because they go to the same conferences, or they work at the same experimental sites, or both?

3. The Application Domain and Framework Components

In order to help answer the question posed at the end of the last section, it is worthwhile to examine the components and facilities that these frameworks provide their users, and what application domains they are used in within the experiments.

3.1. Framework Components

The major components of event processing frameworks are: 1. an execution engine including the concept of event loops and scheduling of algorithms and filters within a single event. In some event processing frameworks, filters can both prematurely stop stop further detailed processing of the current event, and reject the event for output. 2. a configuration system implemented in some scripting

language. There are some differences about the need for a Turing complete[6] language, verses a declarative language, but all use the configuration system to specialize and modify the behaviors of their plugin components, including the execution engine. 3. tools for building an event data model and the persistence of that model to storage. 4. management of non-Event data such as detector conditions data, luminosity or beam conditions, etc. 5. provenance and meta-data creation and management 6. a service system which manages the shared resources needed by components. For example a message logging system, where the shared resource is the job log file, is a straight forward service. 7. the component architecture itself including plugin management, implemented through shared libraries, for all of the elements described in

3.2. Application Domain

The space of applications that an experiment framework must provide is widely agreed. Primary reconstruction, Event Generation and Simulation are the primary applications for experiment processing frameworks. These applications read input files, apply processing steps and write output files. (This statement holds even for event generators like Pythia, if one considers its large number of parameter settings as input data.) While experiments differ in the complexity of their software triggers, most experiments require high level online triggering and monitoring. The offline frameworks are reused online to easily move algorithms for selection and monitoring from the offline environment in which they are developed into the real-time context of data taking. The component nature of these frameworks allows online input and output to be specialized for that purpose, without the users having to be expert in their experiment's DAQ systems. Most end user analysis in HEP is done with ROOT[5]. However it is often the case that after the primary reconstruction, the resulting datasets are too large and unwieldy for any end user analysis tool. Experiment event processing frameworks are used to reduce the reconstruction datasets by means of: 1. reducing the number of events in a sample by applying event selection criteria, 2. reducing the size of collections within selected events by applying threshold cuts to the objects within a collection, for example in a TrackCollection you could require each track saved to have a pT above some value, 3. reducing the size of the objects within a collection, by restricting their information content to be only what is required for most analysis. The information set required for downstream reconstruction components is often much larger then what is required for end user analysis. Experiment event processing frameworks provide the tools necessary to carry out this reduction with the components described in the previous paragraph so that the resulting datasets are manageable and ready for user's physics analysis.

4. Is Wider Field-Wide Collaboration Possible?

The basics of event processing frameworks enumerated in section 3 have not changed much in the past decade. This can be seen from the talk I gave at the 2006 CHEP in India[4]. Note that nothing listed above is really specific to any one frontier or domain.

It is very important to be wary of implementation details masquerading as requirements, there are many ways of supporting a diversity of requirements within an existing framework. For example some may claim that a streaming detector is different because it doesn't have the concept of an event. However DayaBay uses Gaudi. DayaBay is a small experiment whose detector operates in a streaming mode. They adapted Gaudi, which was designed as an event processing framework, for all of the reasons that are listed in reference[3].

The solution space for the components of event processing frameworks, is not that large, for instance in the area of persistency, most use ROOT I/O, in the area of configuration, most use python (including the very popular pyROOT interface to ROOT). In 2006 the diversity of solutions was much larger. Now is the time to take advantage of this convergence within the field. The communication tools for collaboration are much better. Most of us use Indico for meeting agendas,

and are very familiar with remote participation tools like ReadyTalk or Vidyo. Code sharing repository tools have been greatly improved by developers outside of HEP, and we are seeing a wide adoption of git, which has much better support for distributed development. The movement within the field to form a HEP software foundation is gaining traction, and has support from the experiments as well as the funding agencies. All of these drivers lead me to conclude that the answer to the question in the title of this paragraph is yes. Whether that collaboration is like the CDF-BaBar collaboration described in 2.1, where code is shared, or more like the Cleo-CDF/BaBar collaboration, where ideas and designs are shared, remains to be seen.

5. Conclusions and Outlook

Putting it all together there is a lot of motivation for wider collaboration on event processing frameworks within the field of HEP. The challenge of the coming paradigm shift to parallel execution of algorithms is large and it will require an expertise that few within the field possess. In the previous shift from Fortran to C++, many of the more senior contributors to experiment software did not make the transition to the new language and were therefore left behind. The need for a concurrent event processing framework, which can shield algorithm writers from the complexities of multi-threaded programming by devising simple rules to follow (like don't use non-const global statics) and minimizing what they need to know, is large, and required in order to not lose another large group of scientist-programmers. With the advent of the need for multi-threaded event processing frameworks, we should exploit all the expertise available in our experiments, and outside of them, to meet these challenges. There are overheads in wide collaborations but we gain more by collaboration in the long run than it costs initially. If successful the building of communities and shared experience as seen in the intensity frontier and GSI/Fair can be extended to the whole community. At CHEP2015 CMS and ATLAS core software groups arranged for a first meeting to discuss our multi-threaded frameworks. For the next CHEP, with the multi-threading frameworks planned to be well established, it will be interesting to see how much overlap and collaboration there is between HEP experiments for this technology transition.

6. References

1. Sverre Jarp *et al* 2011 *J. Phys.: Conf. Ser.* **331** 052009, "Evaluating the scalability of HEP software and multi-core hardware" doi:10.1088/1742-6596/331/5/052009
2. R Jacobsen, Marc Turcotte, Elizabeth Sexton-Kennedy, Christopher D. Jones, Martin Lohner, Simon Patton, "Engineering a Shared Software Base Between High Energy Physics Experiments" http://www.lns.cornell.edu/~cdj/publications/conferences/CHEP98/JointDesign.pdf
3. C Group 2015 *J. Phys.: Conf. Ser.* ***CHEP2015*** "Computing at the Intensity Frontier" http://indico.cern.ch/event/304944/session/15/contribution/554/material/slides/0.pdf
4. E Sexton-Kennedy 2006, "Event Processing Frameworks a Social and Technical Challenge" http://indico.cern.ch/event/048/session/0/contribution/436/material/slides/0.ppt
5. https://root.cern.ch/drupal/
6. http://en.wikipedia.org/wiki/Turing_completeness

Acknowledgments

Fermilab: Operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the United States Department of Energy.