Fermilab

Managed by Fermi Research Alliance, LLC for the U.S. Department of Energy Office of Science

A Review of Event Processing Frameworks used in HEP

Elizabeth Sexton-Kennedy

21st International Conference on Computing in High Energy and Nuclear Physics

13 April 2015

Why review Event Processing Frameworks now?

- The field is on the brink of a second paradigm shift from serial to parallel execution.
- I believe it’s beneficial to reflect on what was going on during the first shift from Fortran to C++ in the mid-‘90s
- Frameworks are what I know best so that’s what I’ll talk about as an example of software collaboration and the issues involved in collaborating across experiments.

A History of Framework Collaborations

- Not long after the first HEP paradigm shift, a Framework collaboration between BaBar, CDF, and Cleo formed at the ’97 CHEP in Berlin.
  - http://www.lns.cornell.edu/~cdj/publications/conferences/CHEP98/JointDesign.pdf Authors: Robert Jacobsen, Marc Turcotte, Elizabeth Sexton-Kennedy, Christopher D. Jones, Martin Lohner, Simon Patton
  - “The impact of this multi-experiment collaboration is found to be positive.”
- Other co-developer pairings also formed later:
  - ATLAS and LHCb on Gaudi
    - users include GLAST, HARP, DayaBay, MINERvA
  - Alice and GSI/Fair on AliROOT/FairROOT
    - users include Panda, Cbm R3B MPD, ASYEOS EIC
  - and the derivation of the intensity frontier, “art” from CMS
    - users: Mu2e, Muon g-2, NOvA, MicroBooNE, LAriAT, Darkside-50, and DUNE

Lessons Learned: quotes from the previously referenced paper

- The BaBar, CDF, CLEOII experience:
  - No single experiment can usually afford a large infrastructure development team. But this team can be assembled by joining forces between experiments with benefits to everyone.
  - We cannot stress enough the importance of early joint design sessions between different experiments. Maintaining constant communication is also paramount.
  - The most influential factor in the sharing success between CDF and BaBar is that both these experiments use the same file organization and building system called Software Release Tools (SRT) and the same code management system, CVS.
    - In 1998: Some effort in the area of careful exploration into perhaps more modern and possibly more powerful tools is indicated.
  - It's key to be using the same foundational classes, and external dependencies

Why do Framework projects diverge?

- Going back to that first collaboration:
  1. The tools available in the ‘90s were not adequate to the task of supporting a geographically diverse multi-stakeholder project.
     1. This is not longer true, git was born from a desire to support the world-wide collaboration on the Linux kernel.
     2. There are distributed repositories and no one instance has more information in it then any other.
  2. The pressure of taking data and the need for the quick fix in the middle of the night make collaboration take a back seat.
     1. Once BaBar started taking data we stopped synchronizing our repositories.
     2. However the common code base continued to be beneficial. When BaBar was forced to migrate from RogeWave to the STL they benefited from the adaptor classes written for CDF.

Is wider collaboration feasible?

- The space of applications that an experiment framework must provide is widely agreed.
  - Online triggering and monitoring, Primary reconstruction, Event Generation and Simulation, Analysis dataset creation
- The community seems to agree about what is required of Frameworks:
  - Execution engine: concept of event loops and scheduling of algorithms
  - Configuration
  - Plugin management for our component architectures
  - Persistency, event data model, and noneEvent data management
  - Provenance and meta-data creation
  - Services, including messaging and error logging/ handling
- The above has been around for more then a decade, see my ’06 CHEP talk
  - Event Processing Frameworks a Social and Technical Challenge
    - http://indico.cern.ch/event/048/session/0/contribution/436/material/slides/0.ppt

Is wider collaboration feasible?

- The solution space is not that large, for instance:
  - Configuration: Python or custom configuration language, Turing complete or declarative language
  - Persistency: most (maybe all) use Root IO
- In 2006 the diversity of solutions was larger.
- Now is the time to take advantage of this convergence.
- The communication and code sharing tools for collaboration are much better.
- Motivation for wider collaboration:
  - the coming paradigm shift to parallel execution of algorithms
  - the building of communities and shared experience as seen in the intensity frontier and GSI/Fair

Outlook

- With the advent of the need for multi-threaded frameworks and the scarce available manpower, it is important to collaborate in the future.
- It does seem that experiments within the same problem domain and on the same timescales should be able to collaborate on the solutions in many of the agreed required elements.
- A quote from Vincenzo Innocente that I strongly agree with:
  - We gain more by collaboration in the long run than it costs initially.
  - We need to be wary of implementation details masquerading as requirements.
- There is never a good time to start over, but the beginning of a new paradigm shift maybe a better time then any other. Reuse and adaption of existing solutions, should be possible.