HEP Event Processing Frameworks

A very high-level view

Marc Paterno
Christopher D. Jones
9 February 2012

Organization of this talk

- The topics we cover are very inter-connected
- We have tried to impose an organization
- This organization makes sense to the authors; we hope it makes sense to you

Facts of the problem domain

- Our units of data processing, *events*, are independent
- Event data are organized hierarchically: run, subrun, event; detector context (calibration, geometry)
- Different levels of hierarchy represent different *intervals of validity*
- Events are collections of tagged arbitrary *structured* data
  - new types are defined by physicists (not framework developers)
  - frameworks don't know about the many types
- Objects put into the event become immutable
- All events passing several possible physicist-defined *filtering criteria* must be written to the same file (by the end of the job, they should be in the same file)
- Non-event *summary data* (*e.g.* histograms) are accumulated and stored in (possibly different) files

Facts of the data flow and program organization

- Framework programs are configured by physicists at *runtime* to use arbitrary modules
- Modules typically contain physicist-specified runtime configuration parameters
- Module processing times range from milliseconds to many seconds
- Modules depend on data from other modules, but this dependency can change from event to event
- Modules communicate through only a few known channels (event, subrun, run); they *never* call another module directly
- Frameworks can be configured to call some modules only if event data meet some *filtering condition*
- Processed events are stored in external files, because
  - processing is often slow
  - we need reproducibility of and agreement on the definition of data samples

Facts of our development community

- Modules are written in C++
- Module development is done independently by physicists
- We want physicists to be able to use multithreading *within* a module (*e.g.* OpenMP, libdispatch)
- Physicists mostly do not know how to write thread-safe code; we need to give them guidelines
  - allow instances of distinct classes to be used simultaneously
  - allow distinct instances of a single module class to be active in different threads
  - allow one instance of a module to be used simultaneously from multiple threads
- Physicists frequently use libraries that are not thread-safe and that we do not control and that they can not (or will not) do without; we need to provide facilities and guidelines for their safe use

A very abstract processing diagram

- Circles represent data products
- Rectangles represent processing modules
- Any or all data products may be written to an output file