JANA2: Multi-threaded Event Reconstruction

Amber Boehnlein, Nathan Brei, David Lawrence, Dmitry Romanov
Jefferson Lab

Mar 14, 2019

ACAT 2019
Saas Fee, Switzerland
Overview of Jefferson Lab

- Department of Energy National Laboratory with research mission in Nuclear Physics

- In operation since 1995

- Managed for DOE by Jefferson Science Associates, LLC
  - Joint venture of Southeastern Universities Research Association and PAE

- Our primary research tool is CEBAF (Continuous Electron Beam Accelerator Facility) – unique in the world

Jefferson Lab by the numbers:

- 700 employees
- FY2018 Budget: $162.4M
- 169 acre site
- 1,600 Active “User Scientists”
- 27 Joint faculty
- 608 PhDs granted to-date (211 in progress)
- K-12 programs serve more than 13,000 students and 300 teachers annually
Hall-D Complex
Electron beam accelerator
- continuous-wave (1497MHz, 2ns bunch structure in halls)
- Polarized electron beam
- Upgraded to 12GeV (from 6GeV)
- 70 μA max @ 12GeV
  (200μA max @ 6GeV)

Thomas Jefferson National Accelerator Facility (JLab)
Newport News, Virginia, USA
Aerial photon taken April 6, 2012

Hall-D Complex

Electron beam accelerator
- continuous-wave
  (1497MHz, 2ns bunch structure in halls)
- Polarized electron beam
- Upgraded to 12GeV
  (from 6GeV)
- 70 μA max @ 12GeV
  (200μA max @ 6GeV)
### GlueX Computing Needs

<table>
<thead>
<tr>
<th></th>
<th>2017 (low intensity GlueX)</th>
<th>2018 (low intensity GlueX)</th>
<th>2019 (PrimEx)</th>
<th>2019 (high intensity GlueX)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real Data</td>
<td>1.2PB</td>
<td>6.3PB</td>
<td>1.3PB</td>
<td>3.1PB</td>
</tr>
<tr>
<td>MC Data</td>
<td>0.1PB</td>
<td>0.38PB</td>
<td>0.16PB</td>
<td>0.3PB</td>
</tr>
<tr>
<td><strong>Total Data</strong></td>
<td><strong>1.3PB</strong></td>
<td><strong>6.6PB</strong></td>
<td><strong>1.4PB</strong></td>
<td><strong>3.4PB</strong></td>
</tr>
<tr>
<td>Real Data CPU</td>
<td>21.3Mhr</td>
<td>67.2Mhr</td>
<td>6.4Mhr</td>
<td>39.6Mhr</td>
</tr>
<tr>
<td>MC CPU</td>
<td>3.0Mhr</td>
<td>11.3MHR</td>
<td>1.2Mhr</td>
<td>8.0Mhr</td>
</tr>
<tr>
<td><strong>Total CPU</strong></td>
<td><strong>24.3PB</strong></td>
<td><strong>78.4Mhr</strong></td>
<td><strong>7.6Mhr</strong></td>
<td><strong>47.5Mhr</strong></td>
</tr>
</tbody>
</table>

**Anticipate 2018 data will be processed by end of summer 2019**

Projection for out-years of GlueX High Intensity running at 32 weeks/year

11/27/18

*Jefferson Lab Computing Review*
Overly Simplified View of JANA’s Role

DAQ → raw data files → C++ objects (low level) → reconstruction algorithms → C++ objects (refined) → reconstructed data files
Some Goals of the JANA framework

• Provide mechanism for many physicists to contribute code to the full reconstruction program

• Implement multi-threading efficiently external to contributed code

• Provide common mechanisms for accessing job configuration parameters, calibration constants, etc...
Data on demand = Don’t do it unless you need it
Stock = Don’t do it twice

Conservation of CPU cycles!
Complete Event Reconstruction in JANA

- **Event Source**
  - HDDM File
  - EVIO File
  - ET system
  - Web Service

- **Event Processor**
  - User supplied code
  - Fill histograms
  - Write DST
  - L3 trigger

Framework has a layer that directs object requests to the factory that completes it.

Multiple algorithms (factories) may exist in the same program that produce the same type of data objects. This allows the framework to easily redirect requests to alternate algorithms specified by the user at run time.
Multi-threading

- Each thread has a complete set of factories making it capable of completely reconstructing a single event.

- Factories only work with other factories in the same thread eliminating the need for expensive mutex locking within the factories.

- All events are seen by all Event Processors (multiple processors can exist in a program).
High event rate (100kHz) requires buffering in front end leading to entangled events “Event” changes meaning.
**std::packaged_task<>** combines data and algorithm into single objects allowing threads to be generic.
What the user needs to know:

```cpp
auto tracks = jevent->Get<DTrack>();

for(auto t : tracks){
    // ... do something with a track
}
```

vector<const *DTrack> tracks
JANA process configured for full recon launch

Run 42513:
Physics Production mode Trigger: FCAL_BCAL_PS_m9.conf
setup: hd_all.tsg
0/90 PERP 90
JD70-100 58um
TPOL Be 75um
beam looks stable
“Parsing Only” Job demonstrates more stable CPU usage with single flavor of thread. Full event reconstruction using JANA1 on KNL via container at NERSC.
Improvements in JANA2

• Better use of “modern” C++ features
  – thread model via C++ language
  – lock guards
  – shared pointers
  – atomic variables
• Generalized use of threads (pool)
  – queue sets
  – subtasks (std::packaged_task<>)
• NUMA awareness
• Python API
Features maintained from JANA1

- On demand interface
- Plugin support
- Rich configuration parameter feature
- Built-in profiling features
- Automated ROOT tree generation*
Schedule for JANA2 LDRD Project at JLab

FY18
10/2017
NERSC R&D
Initial Development
4/2018
5/2018
6/2018
(NERC2018)
7/2018
9/2018
GlueX Integration R&D
Documentation

FY19
10/2018
Develop 2nd tier features
12/2018
Update documentation
3/2019
Integrate with JLEIC
6/2019
Port Hall-D recon.

FY20
10/2019
Port Hall-D recon.
2/2020
Update documentation

10/2019?*
(CHEP2019)

*Conference presentations
Welcome to JANA!

JANA is a C++ framework for multi-threaded HENP (High Energy and Nuclear Physics) event reconstruction. It is very efficient at multi-threading with a design that makes it easy for less experienced programmers to contribute pieces to the larger reconstruction project. The same JANA program can be used to easily do partial or full reconstruction, fully maximizing the available cores for the current job.

Its design strives to be extremely easy to setup when first getting started, yet have a depth of customization options that allow for more complicated reconstruction as your project grows. The intent is to make it easy to run on a laptop for local code development, but to also be highly efficient when deploying to large computing sites like NERSC.

JANA is currently undergoing a complete rewrite. The new version will be JANA 2. The code is not quite ready for actual use yet, but you are free to browse around to see how progress is going. The project is hosted on GitHub.

```c++
auto tracks = jevent->GetDTrack>(tracks);

for (auto t : tracks){
    // ... do something with a track
}
```
Backups
"Parsing Only" Job demonstrates more stable CPU usage with single flavor of thread