JANA2: Mutli-threaded Event Reconstruction

David Lawrence,

Amber Boehnlein Jefferson Lab

July 10, 2018

CHEP 2018

Sofia, Bulgaria

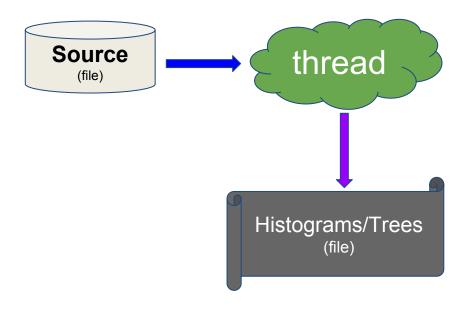


```
205 // IsJoined
   bool JThread::IsJoined(void)
                                                    FACTORY
      return _isjoined;
210 }
213 // Loop
215 void JThread::Loop(void)
216 {
      // Set thread_local global v
       JTHREAD = this; PRODUCT
                                            STOCK
       mLogger = new JLog(0); //std::cou
      /// Loop continuously, processing
         while( mRunStateTarget != kRUN_STATE_ENDED )
            // If specified, go into idle state
            if( mRunStateTarget == kRUN STATE IDLE ) mRunState = kRUN STATE IDLE;
            // If not running, sleep and loop again
            if(mRunState != kRUN STATE RUNNING)
               std::this_thread::sleep_for(mSleepTime); //Sleep a minimal amount.
                continue:
236
             //Check if not enough event-tasks queued
            if(CheckEventQueue())
238
                //Process-event task is submitted, redo the loop in case we want to buffer
                continue;
```

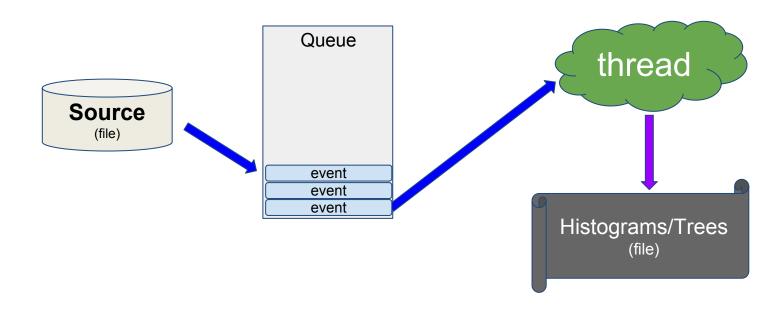




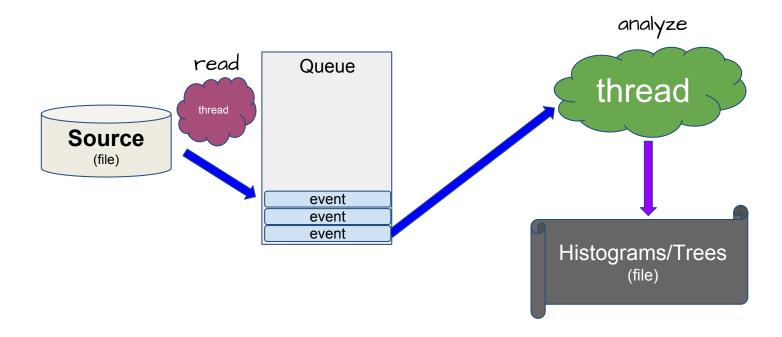
basic model for single thread event reconstruction



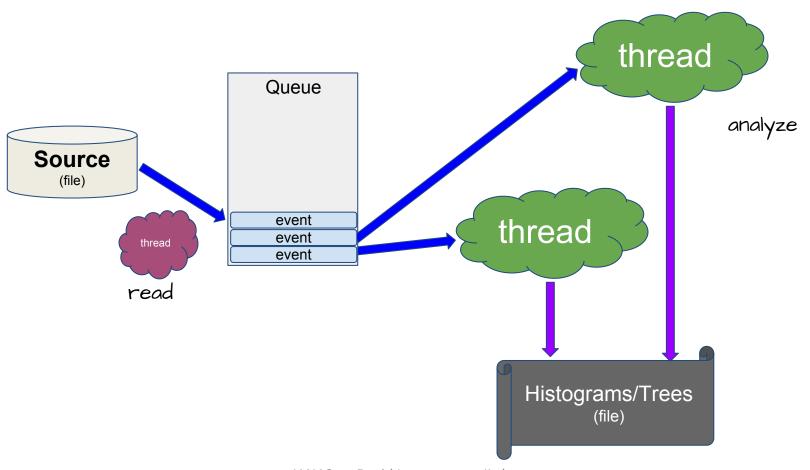
Add queue to move towards multi-thread



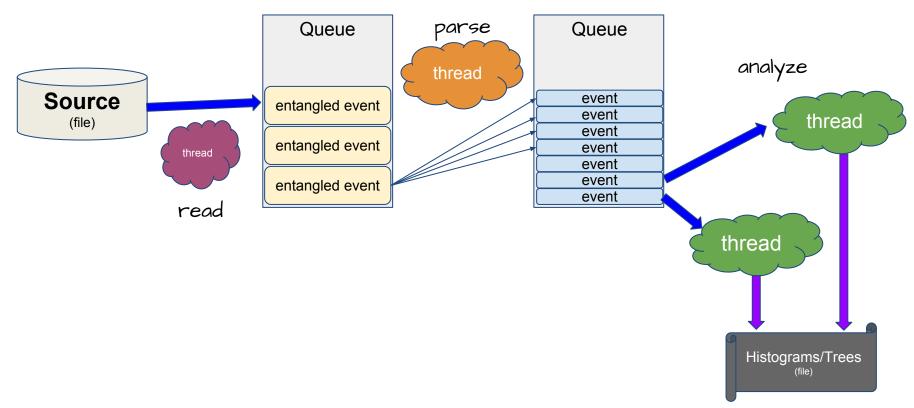
Two flavors of thread required



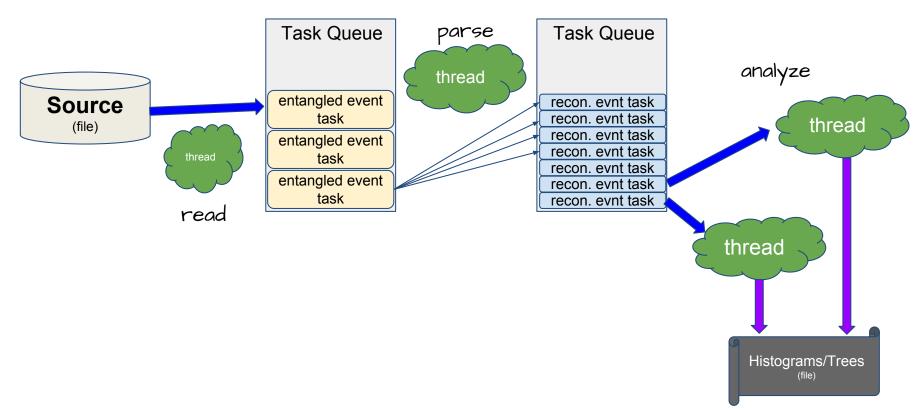
basic model for multi-threaded event reconstruction

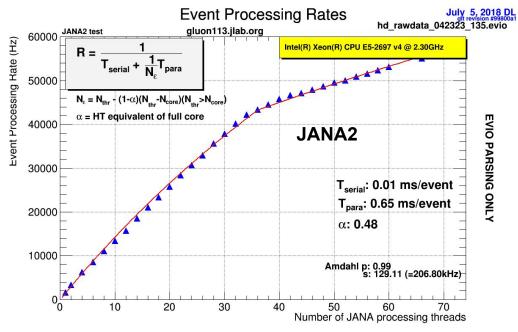


High event rate (100kHz) requires buffering in front end leading to entangled events "Event" changes meaning.

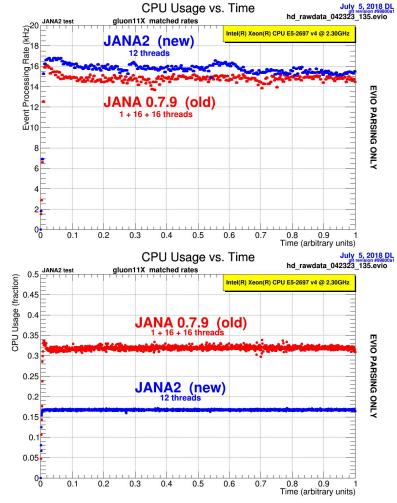


std::packaged_task combines data and algorithm into single objects allowing threads to be generic

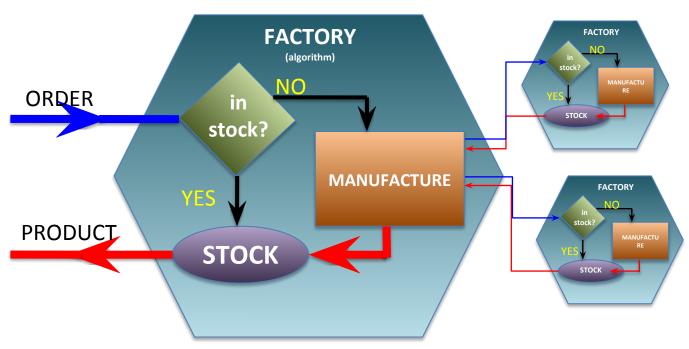




"Parsing Only" Job demonstrates more stable CPU usage with single flavor of thread



Factory Model



Data on demand = Don't do it unless you need it Stock = Don't do it twice _____

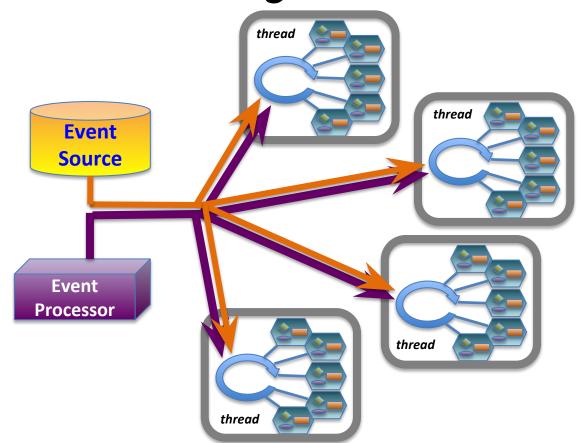
Conservation of CPU cycles!

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)



What the reconstruction algorithm developer needs to know

```
auto tracks = jevent->Get<DTrack>();
for(auto t : tracks){
   // ... do something with a track
}
```

GlueX Reconstruction Software

Automatic call graph generation using janadot plugin

JANA2 Website on GitHub





Multi-threaded HENP Event Reconstruction

Getting Started



// 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.

Download

FAQ

It's 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 efficent 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

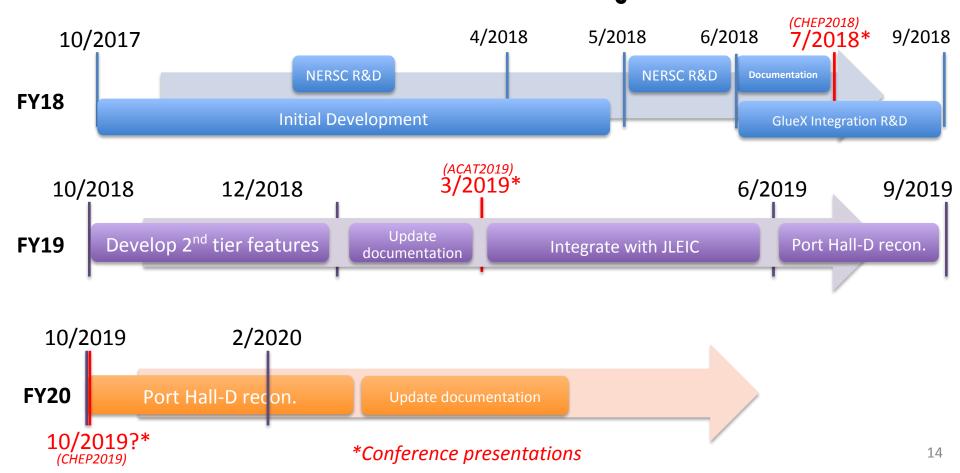
```
auto tracks = jevent->Get<DTrack>(tracks);
for(auto t : tracks){
   // ... do something with a track
}
```

JeffersonLab.

JANA2 is maintained by

This page was generated by GitHub Pages.

Schedule for JANA2 LDRD Project at JLab



Backups



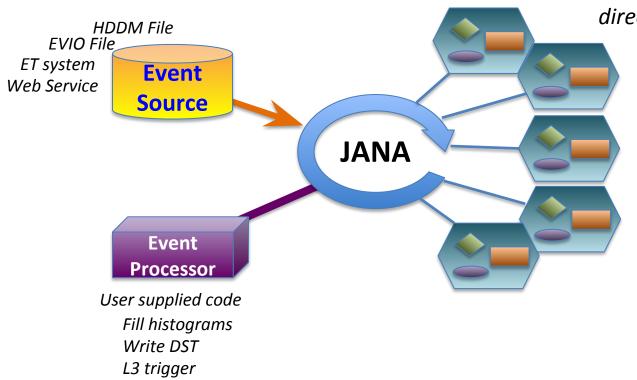
GlueX (JANA1) Jobs on NERSC Cori (I & II)

Job ID	Run	Rate	Wall Hours	CPU Hours	Threads
9662111	30279.002	248Hz	1.325	42.41	64 (Haswell)
9654879	30279.002	104Hz	3.230	103.36	256 (KNL)
9654892	31034.002	229Hz	1.991	63.72	64 (Haswell)
9667013	31034.002	95Hz	4.862	155.58	256 (KNL)

Run 30279: 150nA, JD70-100 58um 0/90 PARA 1.2M events (single file) 2/4/2017 Run 31034: 150nA, JD70-100 58um 45/135 PERP 1.6M events (single file) 3/8/2017

KNL jobs run about 2.4 times slower = cost **2.4** times as much from NERSC allocation

Complete Event Reconstruction in JANA



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