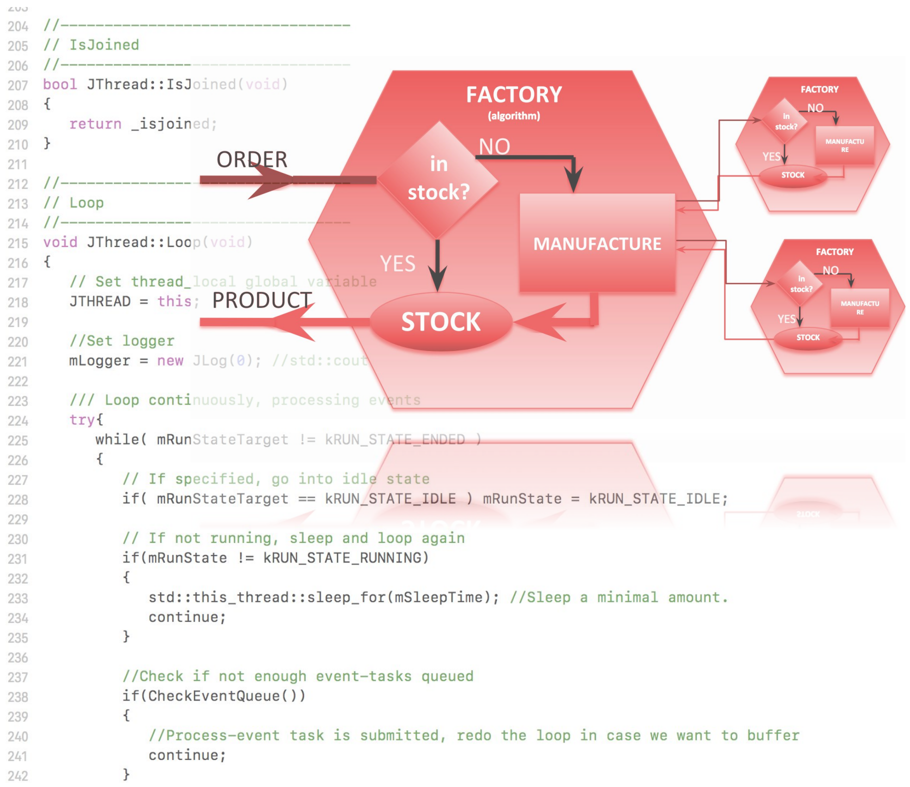
JANA2: Multi-threaded Event Reconstruction

Amber Boehnlein, Nathan Brei, **David Lawrence**
Jefferson Lab

Nov 5, 2019

CHEP 2019

Adelaide, Australia



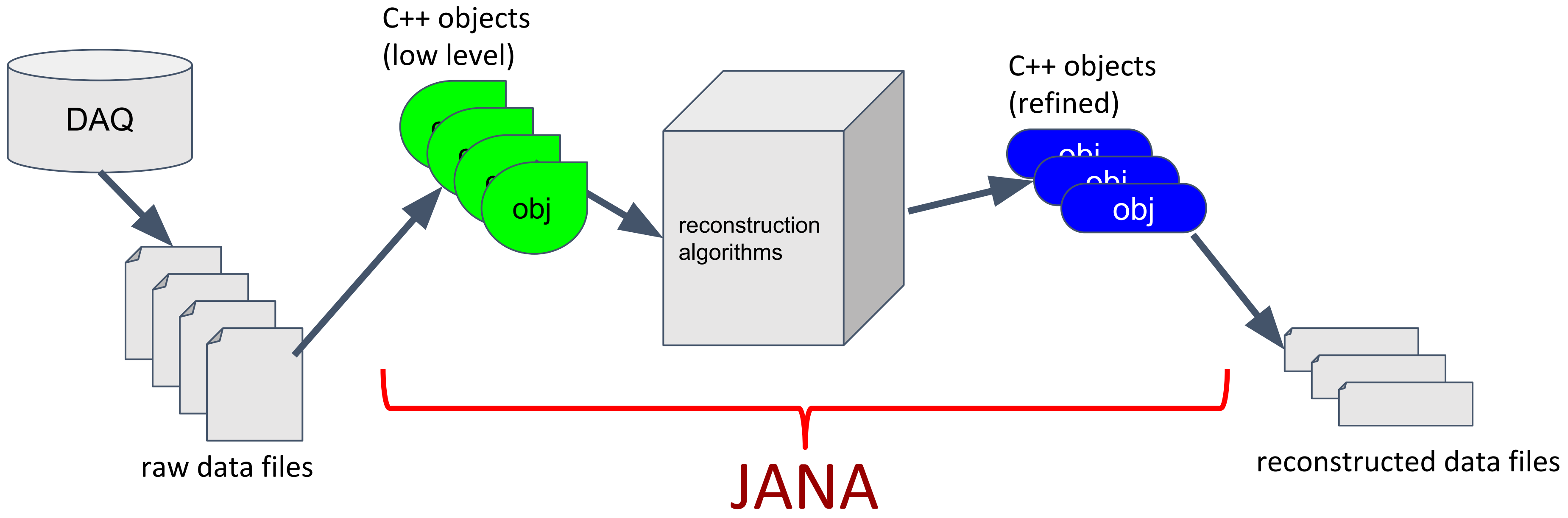
```
//-----------------------------------
// IsJoined
//-----------------------------------
bool JThread::IsJoined(void)
{
   return _isjoined;
}

//-----------------------------------
// Loop
//-----------------------------------
void JThread::Loop(void)
{
   // Set thread_local global variable
   JTHREAD = this;

   //Set logger
   mLogger = new JLog(0); //std::cout

   /// Loop continuously, processing events
   try{
      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;
         }

         //Check if not enough event-tasks queued
         if(CheckEventQueue())
         {
            //Process-event task is submitted, redo the loop in case we want to buffer
            continue;
         }
```

Jefferson Lab

U.S. DEPARTMENT OF ENERGY | Office of Science

JSA

Overly Simplified View of JANA's Role



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...

Features maintained from JANA1

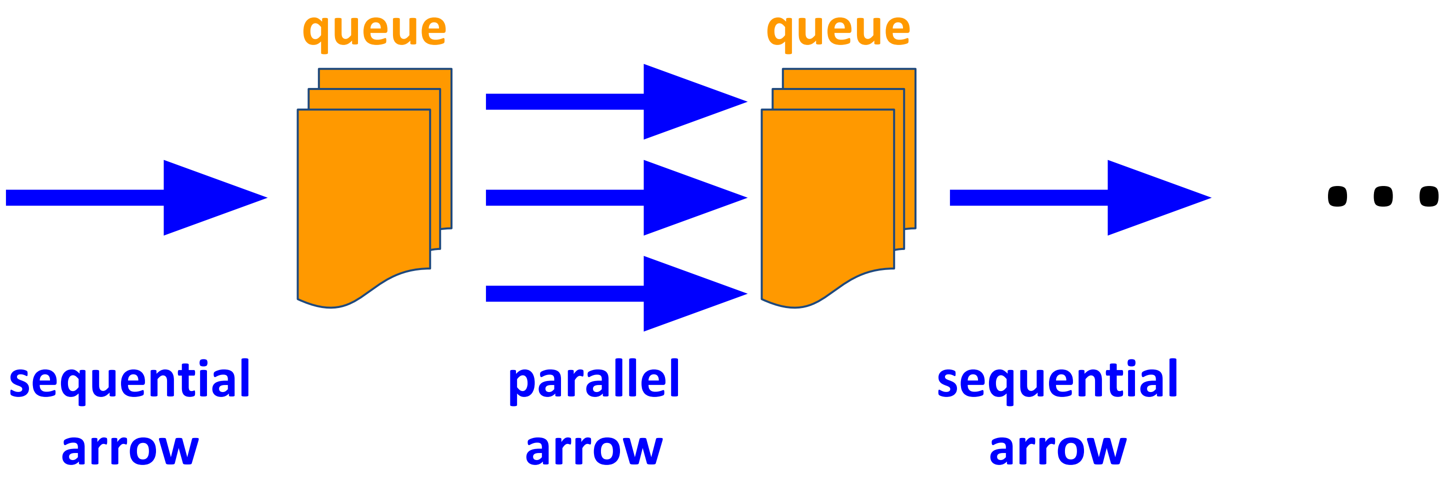
- On demand interface
- Plugin support
- Rich configuration parameter feature
- Built-in profiling features
- Automated ROOT tree generation*

Features Added in JANA2

- Better use of “modern” C++ features
  - thread model via C++ language
  - lock guards
  - shared pointers
  - lambda functions
- Generalized use of threads (pool)
  - multiple queues
  - arrows (sequential or parallel)
- NUMA awareness
- Python API (both embedded and as an extension)

JANA2 arrows separate sequential and parallel tasks

- CPU intensive event reconstruction will be done as a parallel arrow
- Other tasks (e.g. histogram filling) can be done as a sequential arrow
- Fewer locks in user code allows framework to better optimize workflow



What the user needs to know:

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

for(auto t : tracks){

  // … do something with const DTrack* t

}
```

*vector<const *DTrack> tracks*

If an alternate factory is desired:
(i.e. algorithm)

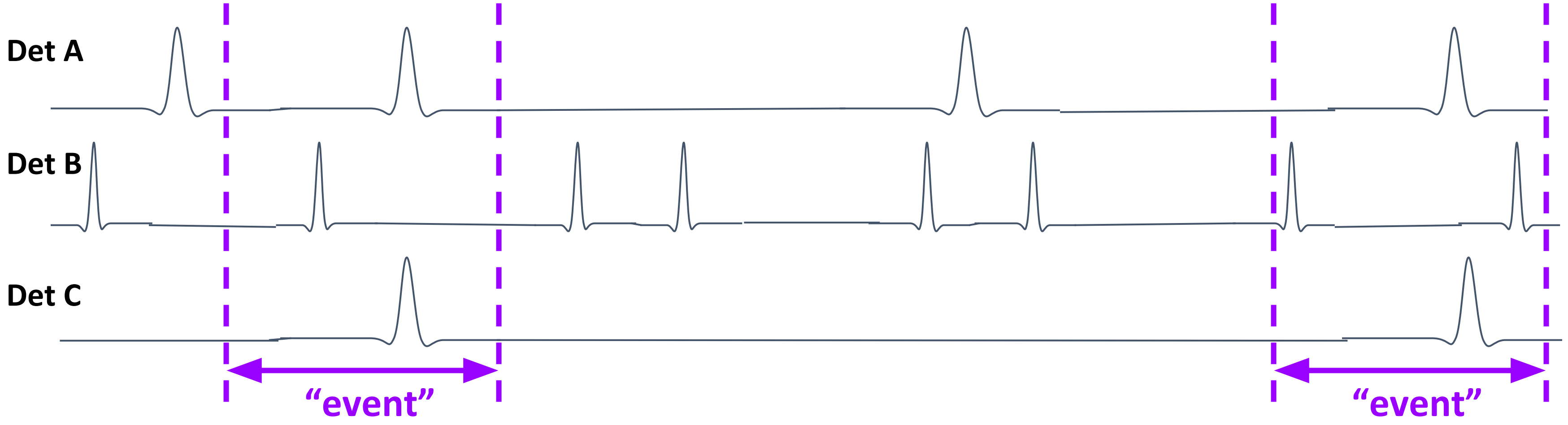
```
auto tracks = jevent->Get<DTrack>("MyTest");
```

or, even better

set configuration parameter: **DTrack:DEFTAG=MyTest**

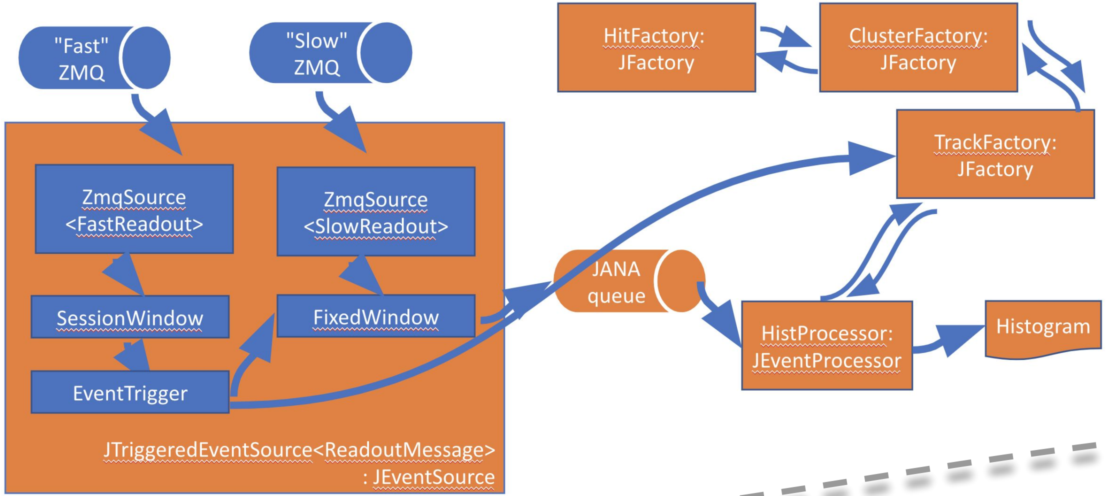
- Configuration parameters are set at run time
- NAME:DEFTAG is special and tells JANA to re-route ALL requests for objects of type NAME to the specified factory.

“Event” Reconstruction



- Physics requires studying a single reaction at a time
- High speed (=high statistics) leads to overlapping reactions in time
- “Event” here really means a slice of time
  - Traditional electronic trigger = single reaction
  - Streaming readout = potentially many reactions

Streaming Readout



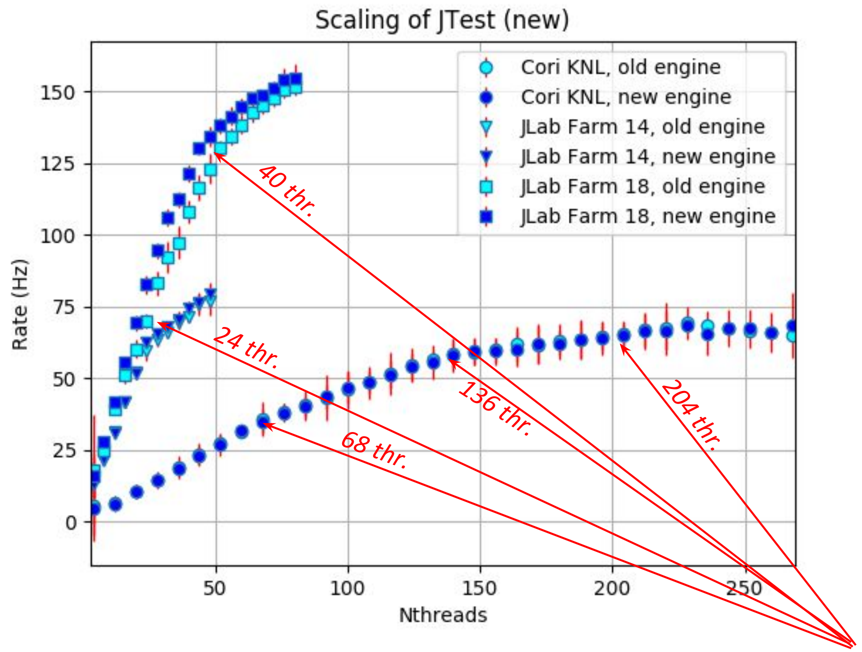
INDRA-ASTRA initiative:

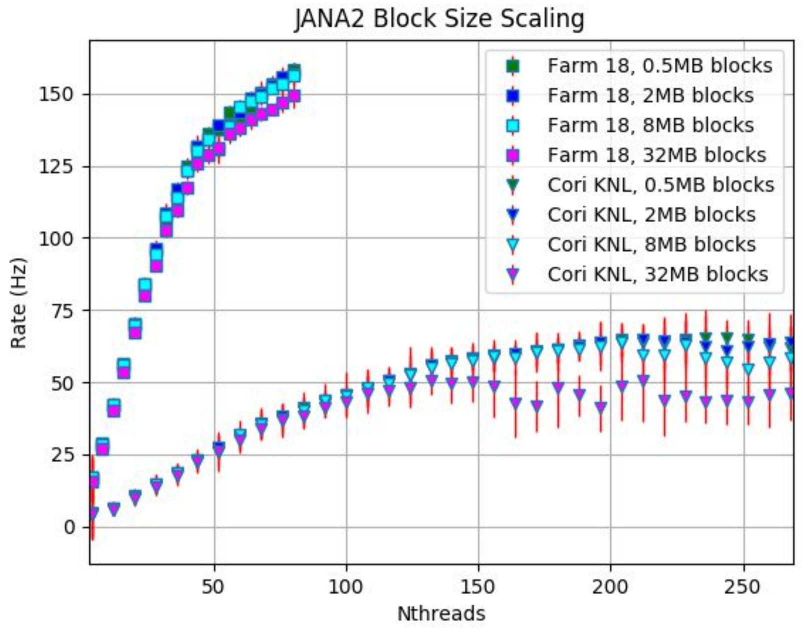
- Software trigger
- Multi-flavored stream merging
- Event building

Support for Heterogeneous Hardware

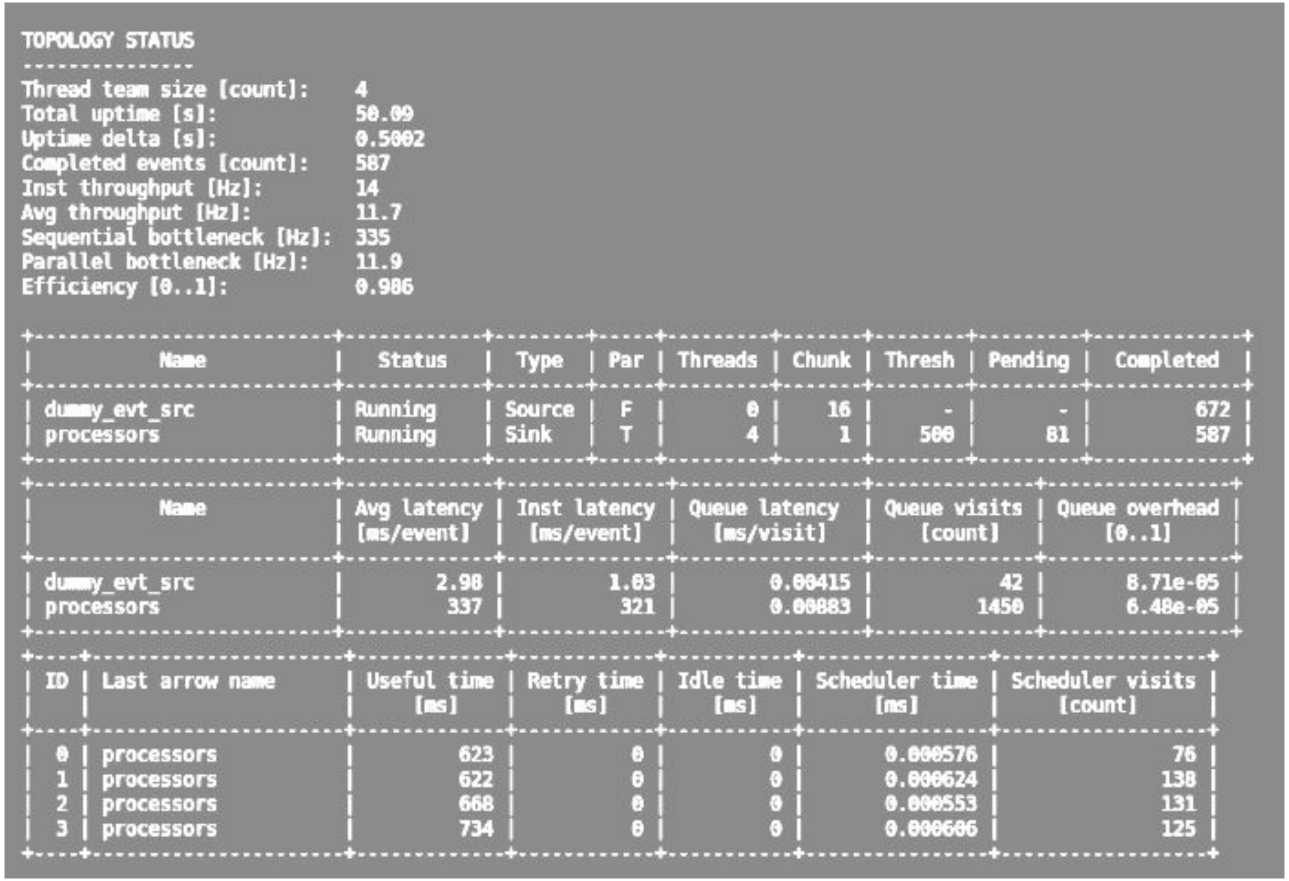
- Sub-event level parallelism
  - Run ML on GPU or TPU

JANA2 Scaling Tests (JLab + NERSC)





kinks indicate hardware boundaries



```
TOPOLOGY STATUS
---------------
Thread team size [count]:    4
Total uptime [s]:            50.09
Uptime delta [s]:            0.5002
Completed events [count]:    587
Inst throughput [Hz]:        14
Avg throughput [Hz]:         11.7
Sequential bottleneck [Hz]:  335
Parallel bottleneck [Hz]:    11.9
Efficiency [0..1]:           0.986

+--------------------------+------------+--------+-----+---------+-------+--------+---------+------------+
|           Name           |   Status   |  Type  | Par | Threads | Chunk | Thresh | Pending |  Completed |
+--------------------------+------------+--------+-----+---------+-------+--------+---------+------------+
| dummy_evt_src            | Running    | Source |  F  |       0 |    16 |      - |       - |        672 |
| processors               | Running    | Sink   |  T  |       4 |     1 |    500 |      81 |        587 |
+--------------------------+------------+--------+-----+---------+-------+--------+---------+------------+
+--------------------------+-------------+-------------+---------------+---------------+----------------+
|           Name           | Avg latency | Inst latency| Queue latency | Queue visits  | Queue overhead |
|                          | [ms/event]  | [ms/event]  |  [ms/visit]   |    [count]    |     [0..1]     |
+--------------------------+-------------+-------------+---------------+---------------+----------------+
| dummy_evt_src            |        2.98 |        1.03 |       0.00415 |            42 |       8.71e-05 |
| processors               |         337 |         321 |       0.00883 |          1450 |       6.48e-05 |
+--------------------------+-------------+-------------+---------------+---------------+----------------+
+----+-----------------------+-------------+------------+------------+----------------+-----------------+
| ID | Last arrow name       | Useful time | Retry time | Idle time  | Scheduler time | Scheduler visits |
|    |                       |    [ms]     |    [ms]    |    [ms]    |      [ms]      |     [count]     |
+----+-----------------------+-------------+------------+------------+----------------+-----------------+
|  0 | processors            |         623 |          0 |          0 |       0.000576 |              76 |
|  1 | processors            |         622 |          0 |          0 |       0.000624 |             138 |
|  2 | processors            |         668 |          0 |          0 |       0.000553 |             131 |
|  3 | processors            |         734 |          0 |          0 |       0.000606 |             125 |
+----+-----------------------+-------------+------------+------------+----------------+-----------------+
```

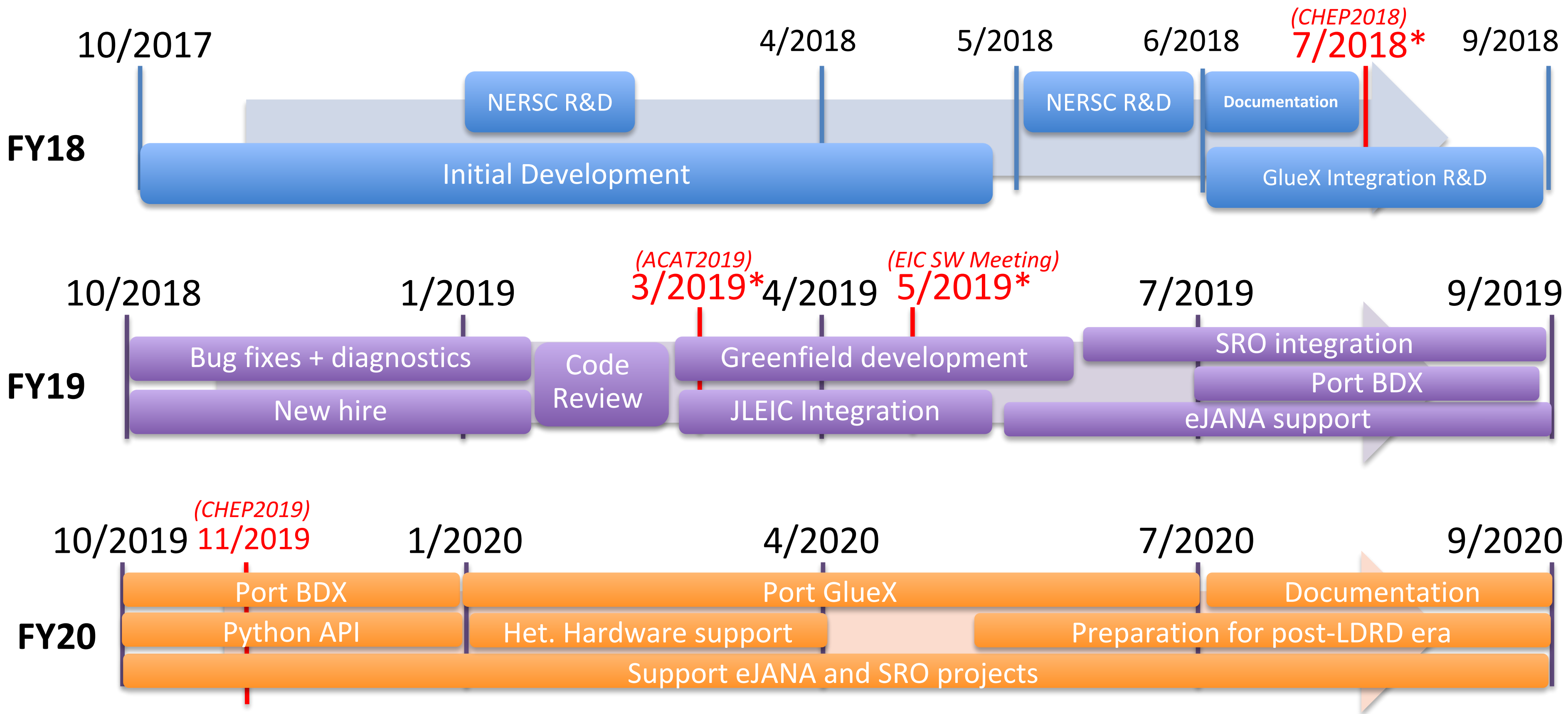
JANA2 now has much better built-in diagnostics. These will help pinpoint bottlenecks, especially in more complex systems

Summary

- JANA2 is:
  - C++ event processing framework
  - multi-threaded
  - currently being written with >10 years experience with JANA1
  - Bigger, better, badder (gooder?)
- Python interface (embedded and extension)
- Follow project on github:

https://github.com/JeffersonLab/JANA2

Schedule



**Conference/Workshop presentations*

Backups

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

Jefferson Lab

Jefferson Lab
Thomas Jefferson National Accelerator

Aerial p
taken Apri

Hall-D
Comple

Electron beam

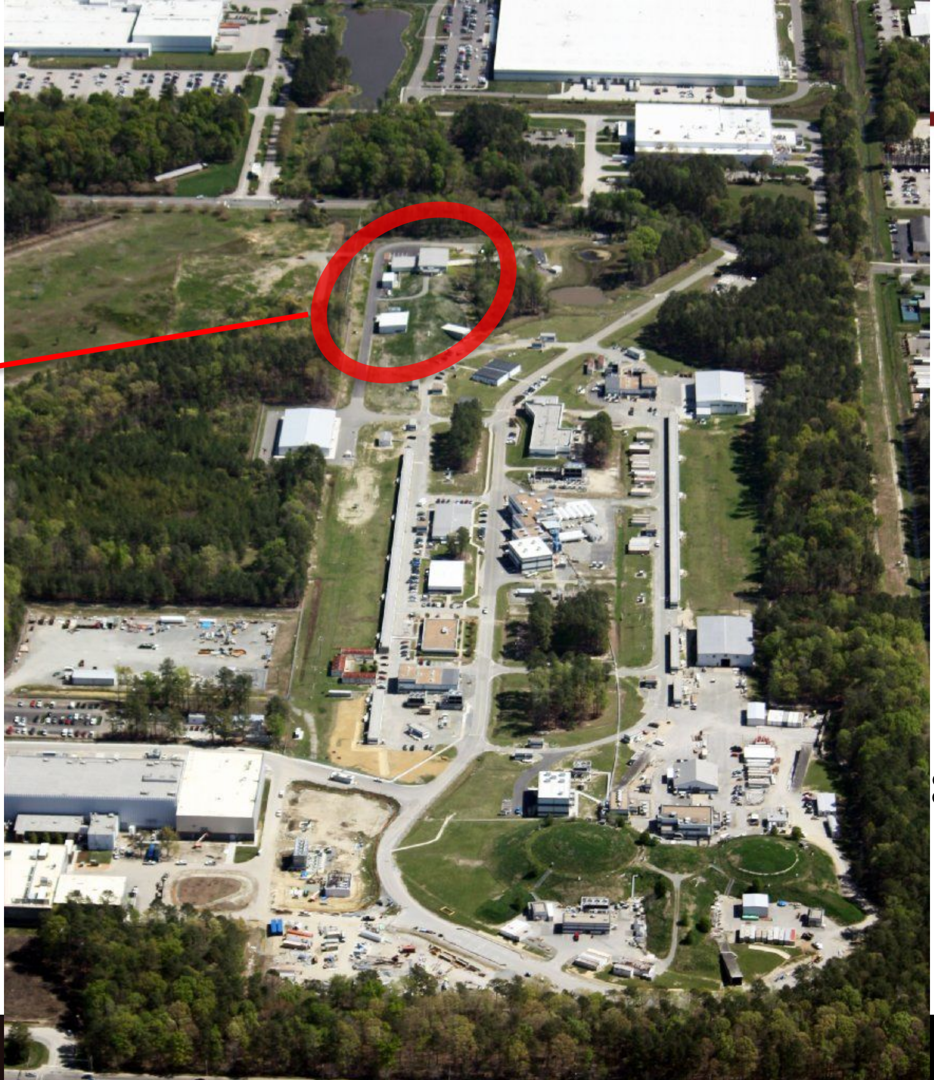
- continuous
  (1497MHz,
  structure in
- Polarized e
- Upgraded t
  (from 6GeV
- 70 µA max
  (200µA max @ 6GeV)

New York
Washington

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

	2017 (low intensity GlueX)	2018 (low intensity GlueX)	2019 (PrimEx)	2019 (high intensity GlueX)
Real Data	1.2PB	6.3PB	1.3PB	3.1PB
MC Data	0.1PB	0.38PB	0.16PB	0.3PB
Total Data	**1.3PB**	**6.6PB**	**1.4PB**	**3.4PB**
Real Data CPU	21.3Mhr	67.2Mhr	6.4Mhr	39.6Mhr
MC CPU	3.0Mhr	11.3MHr	1.2Mhr	8.0Mhr
Total CPU	**24.3PB**	**78.4Mhr**	**7.6Mhr**	**47.5Mhr**

Anticipate 2018 data will be processed by end of summer 2019

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

	Out - years (high intensity GlueX)
Real Data	16.2PB
MC Data	1.4PB
Total Data	**17.6PB**
Real Data CPU	125.6Mhr
MC CPU	36.5Mhr
Total CPU	**162.1Mhr**

Project Goals

- Address the feedback from the JLab FY 2016 ALP feedback report on developing **Core Competency**❖ **in Advanced Computer Science**.
- Position JLab to **export software for use at ASCR facilities** such as NERSC by the wider Laboratory community.
- Updated JANA framework suitable for use in multiple JLab experiments.
  - Support the **GlueX** experiment by modernizing the JANA framework. This will make the GlueX software more robust, easier to expand, and a **better training tool for students** working with the software.
  - Support future experiments. Specifically, **JLEIC** and **BDX**. JLEIC's eJANA software is already being developed synergistically with JANA2 and we wish to continue this.
  - Support **streaming readout technology** at JLab. This feature could also make JANA2 attractive as an export to other labs employing streaming readout systems.

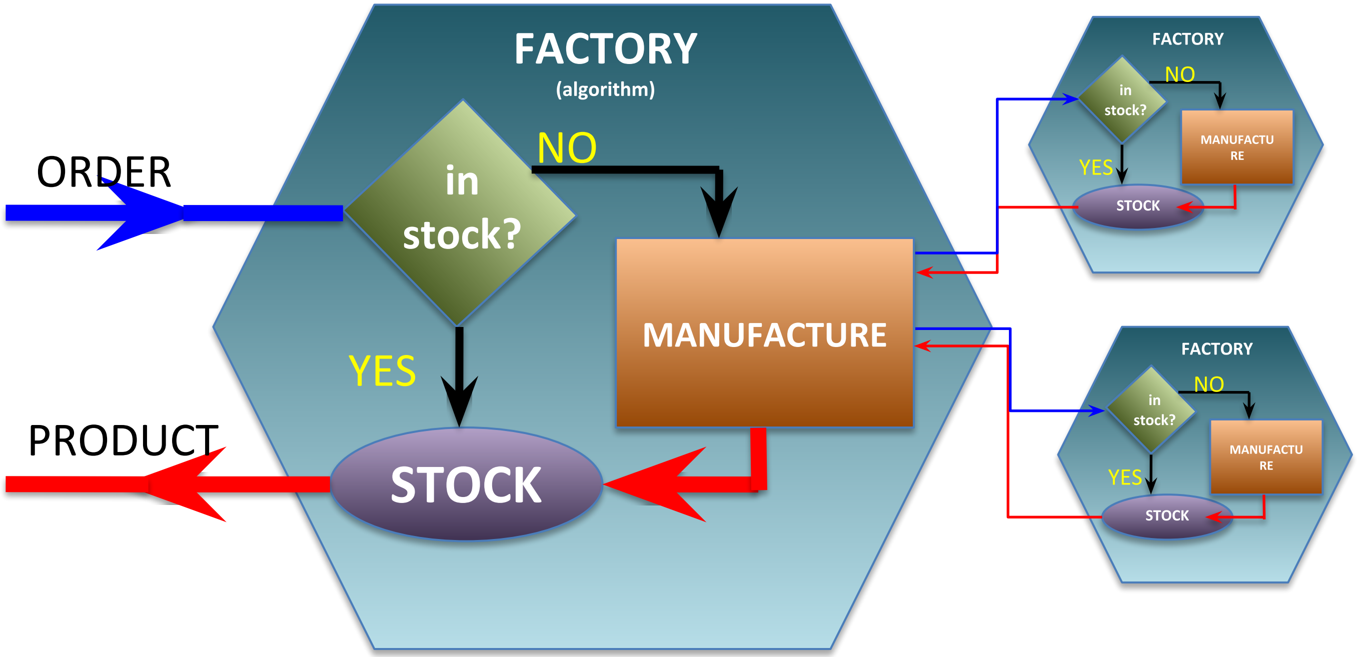
❖https://science.energy.gov/sbir/applicant-and-awardee-resources/national-labs-profiles-and-contacts/

Accomplishments in FY19

- Hired full time CS (Nathan Brei)
  - Reviewed goals
  - Redesigned workflow core (“greenfield”)
  - Improved integrated diagnostics and logging
  - NUMA awareness
- Integrated into JLEIC’s eJANA project
- Began integration into Streaming Readout (SRO) system
- Presented JANA2 at multiple international conferences and workshops
- R&D into Python interface*
- Begin port of BDX software*

**work targeted for the upcoming final quarter*

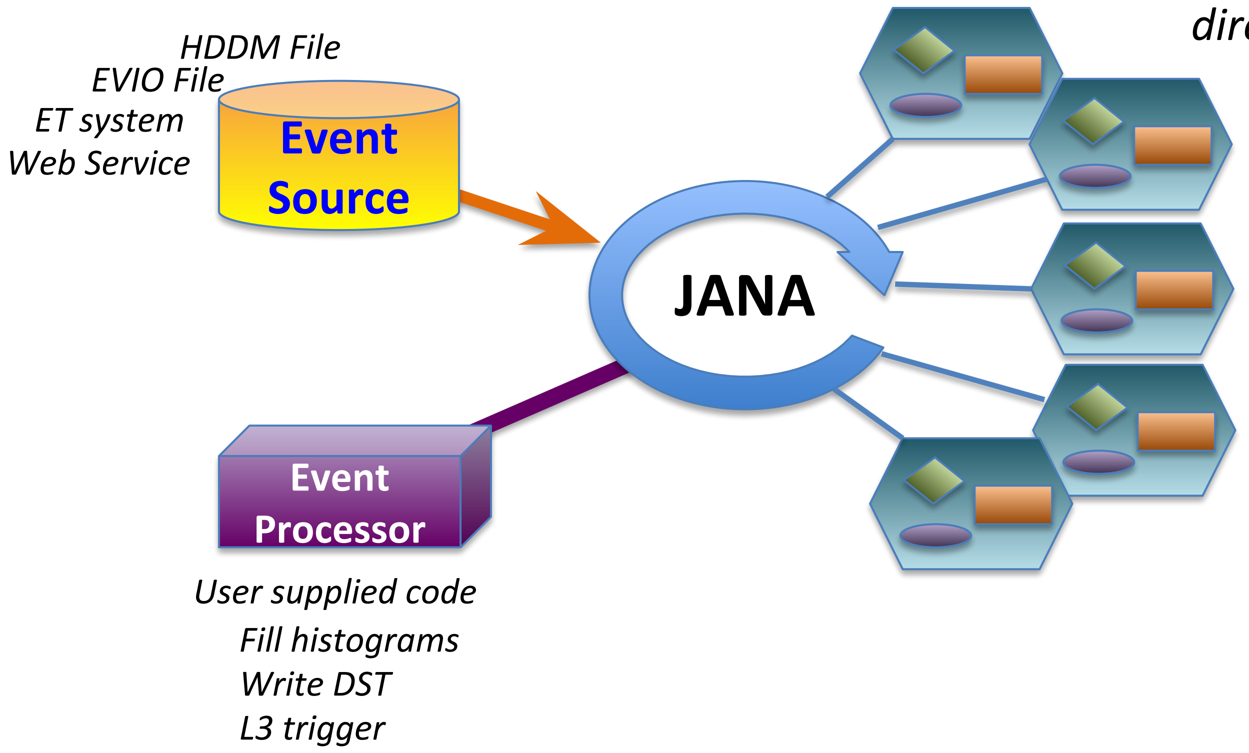
Factory Model



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



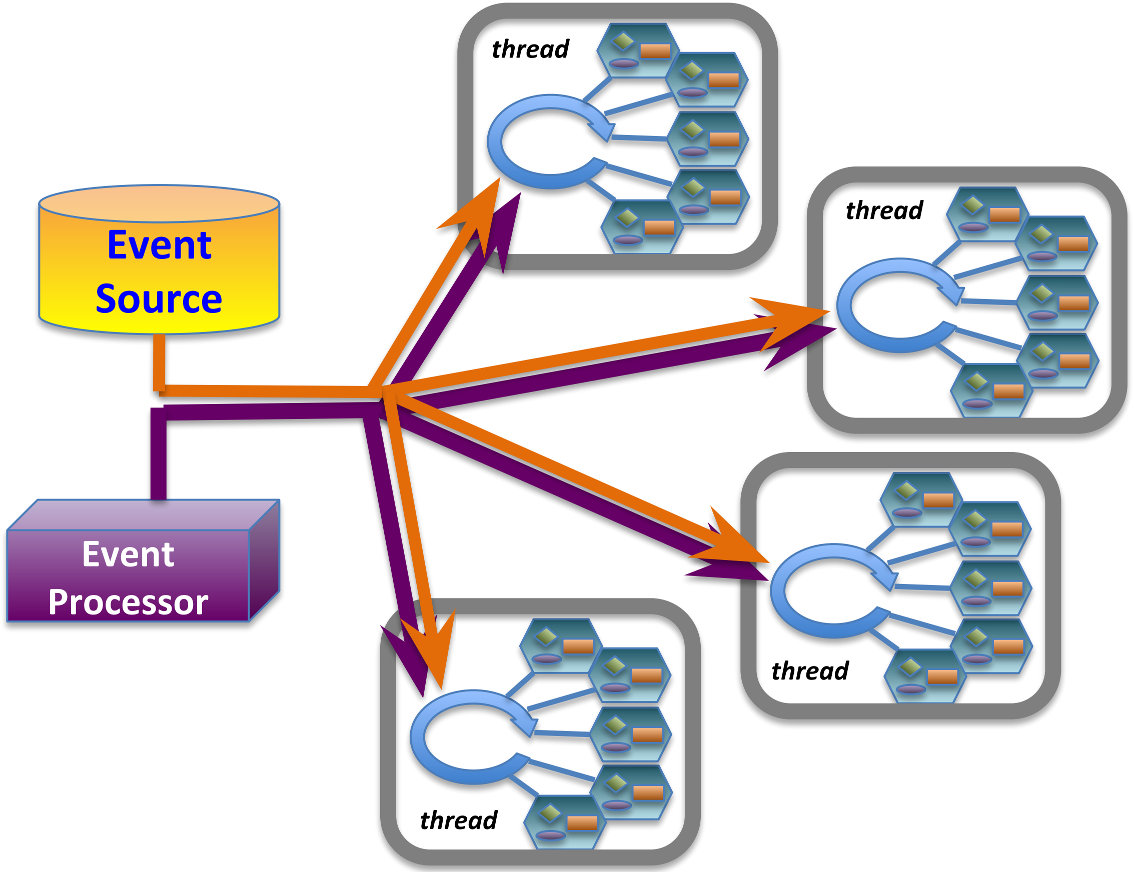
Framework has a layer that directs object requests to the factory that completes it

Multiple algorithms (factories) may exist in the same program that produce the same type of data objects

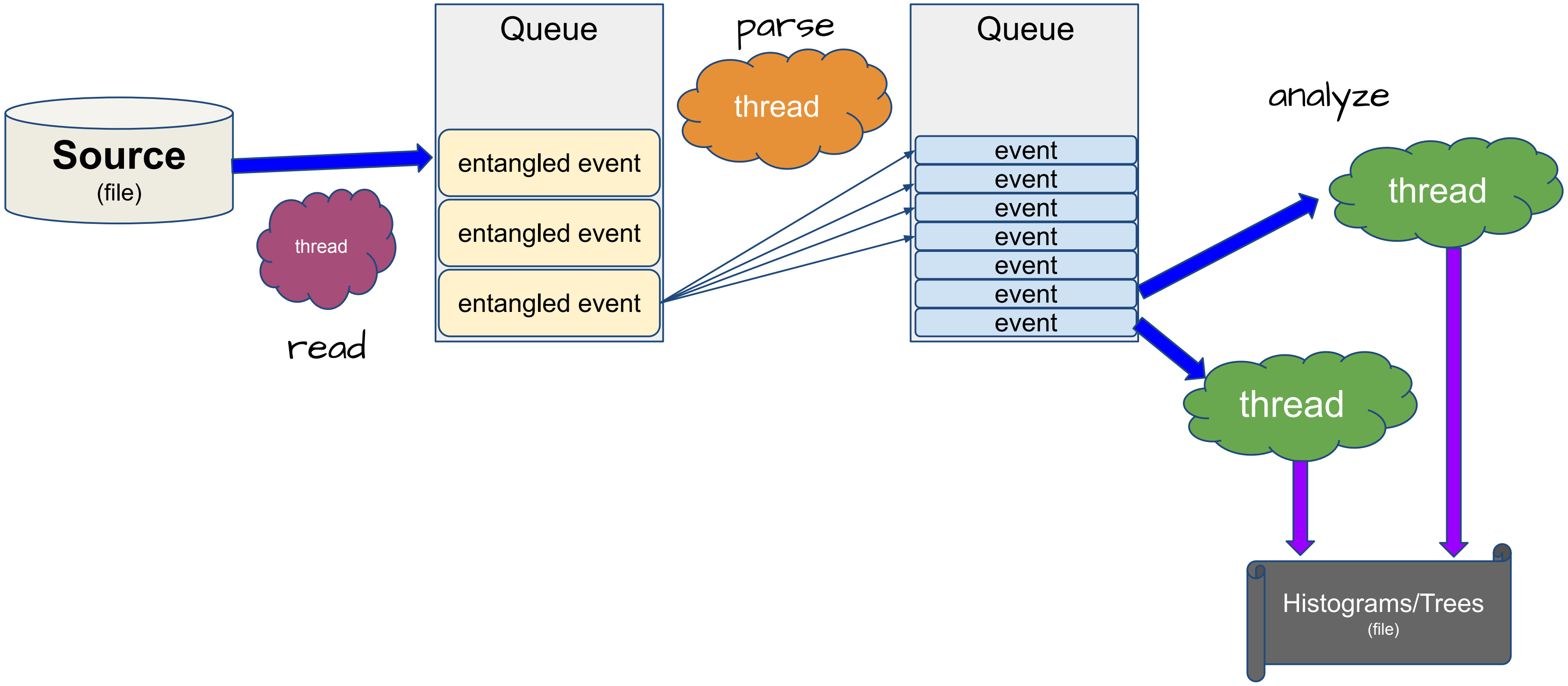
This allows the framework to easily redirect requests to alternate algorithms specified by the user at run time

Multi-threading

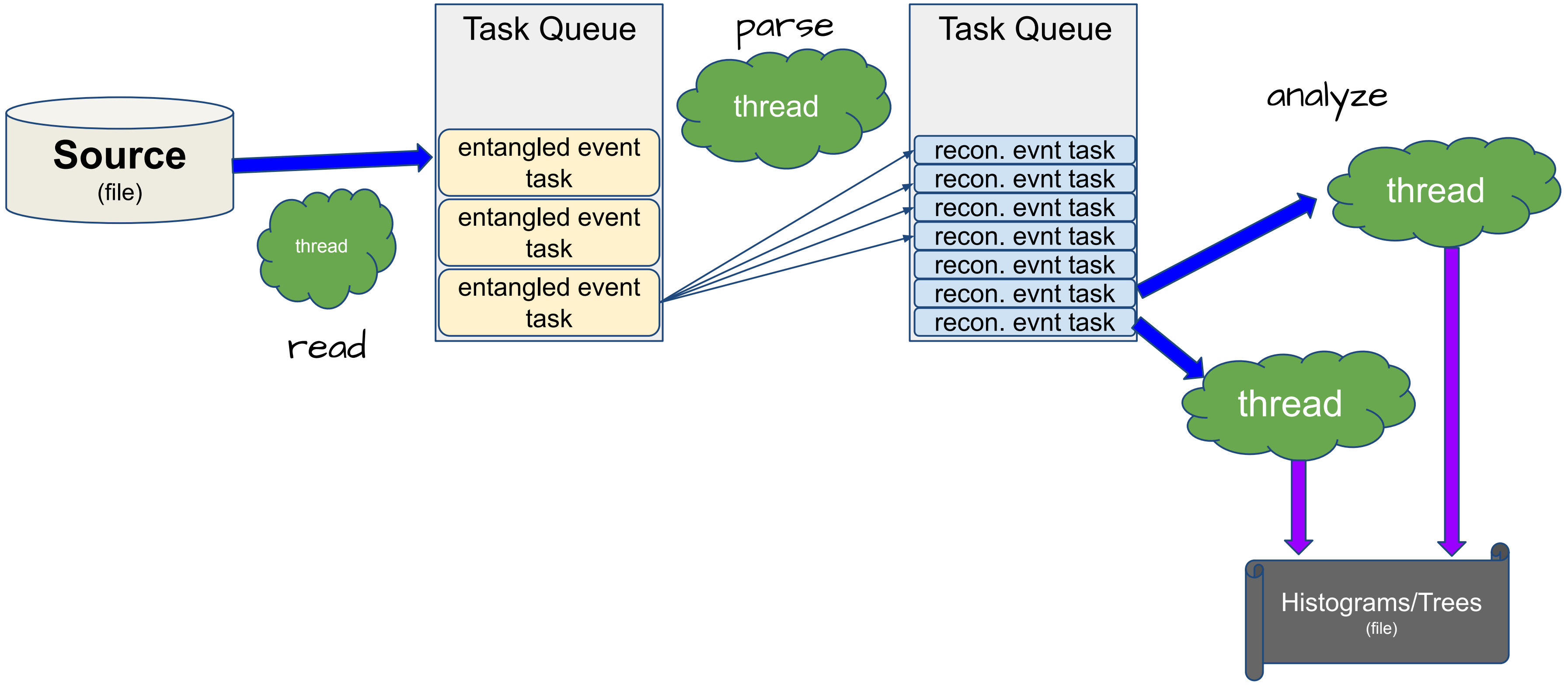
- *Each thread has a complete set of factories making it capable of completely reconstructing a single event*
- *Factories only work with other factories in the same thread eliminating the need for expensive mutex locking within the factories*
- *All events are seen by all Event Processors (multiple 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



GlueX Reconstruction Software

Automatic call graph generation using janadot plugin