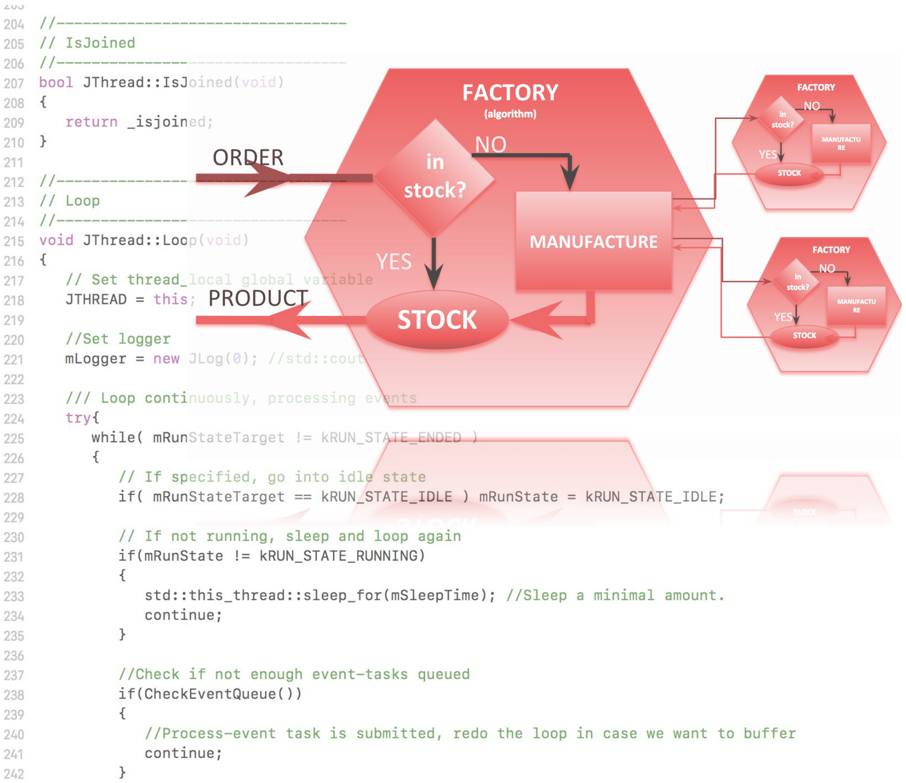
JANA2: Multi-threaded Event Reconstruction

Nathan Brei, **David Lawrence**
Jefferson Lab

April 1, 2020

HSF Framework Working Group



```
//---------------------------------
// 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;
         }
```

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

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**

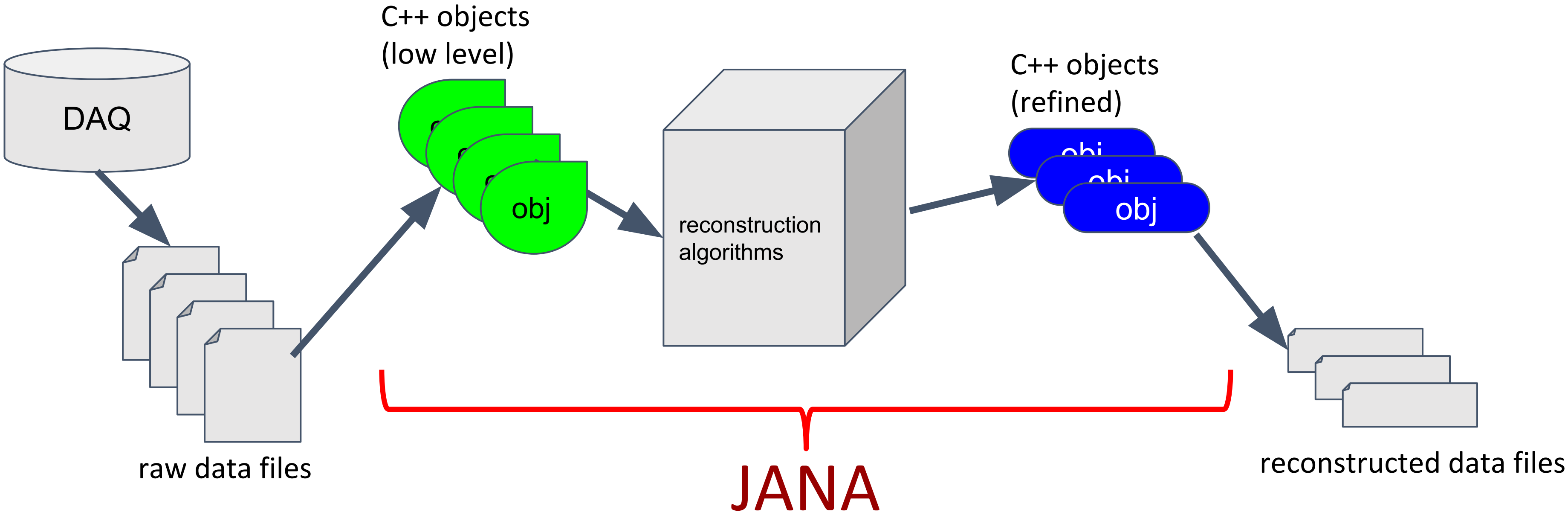
Anticipate 2018 data will be processed by end of summer 2019

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**

Event size:
12-13kB

JANA's Role in Data Processing



Some Goals of the JANA framework

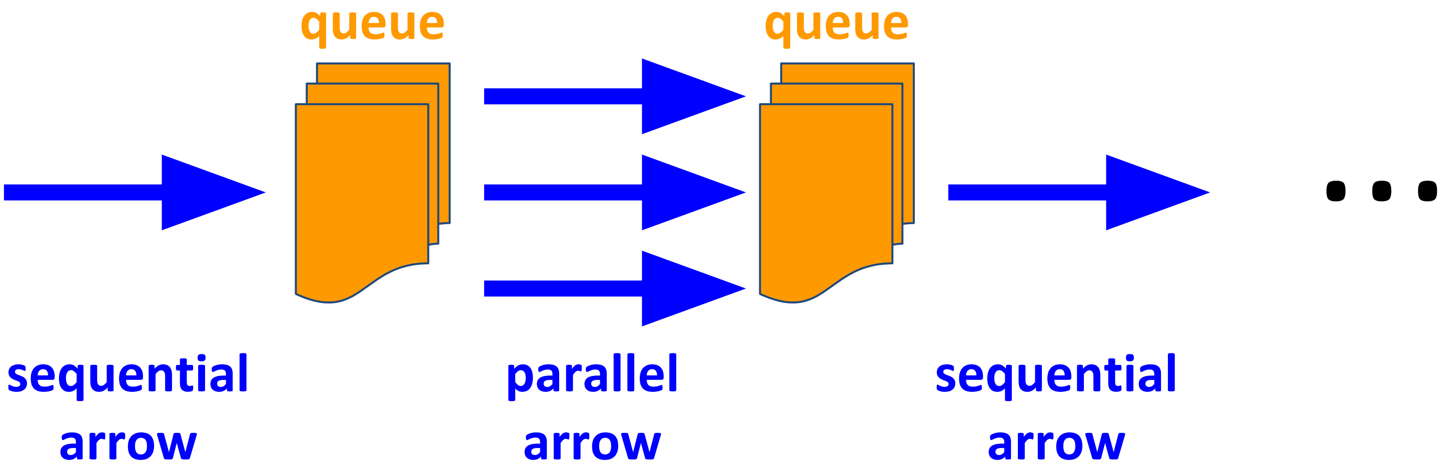
- Provide mechanism for many physicists to contribute code to the full reconstruction program

- Implement multi-threading efficiently external to contributed code

- Provide common mechanisms for accessing job configuration parameters, calibration constants, etc...

JANA2 arrows separate sequential and parallel tasks

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



queue

queue

. . .

sequential arrow

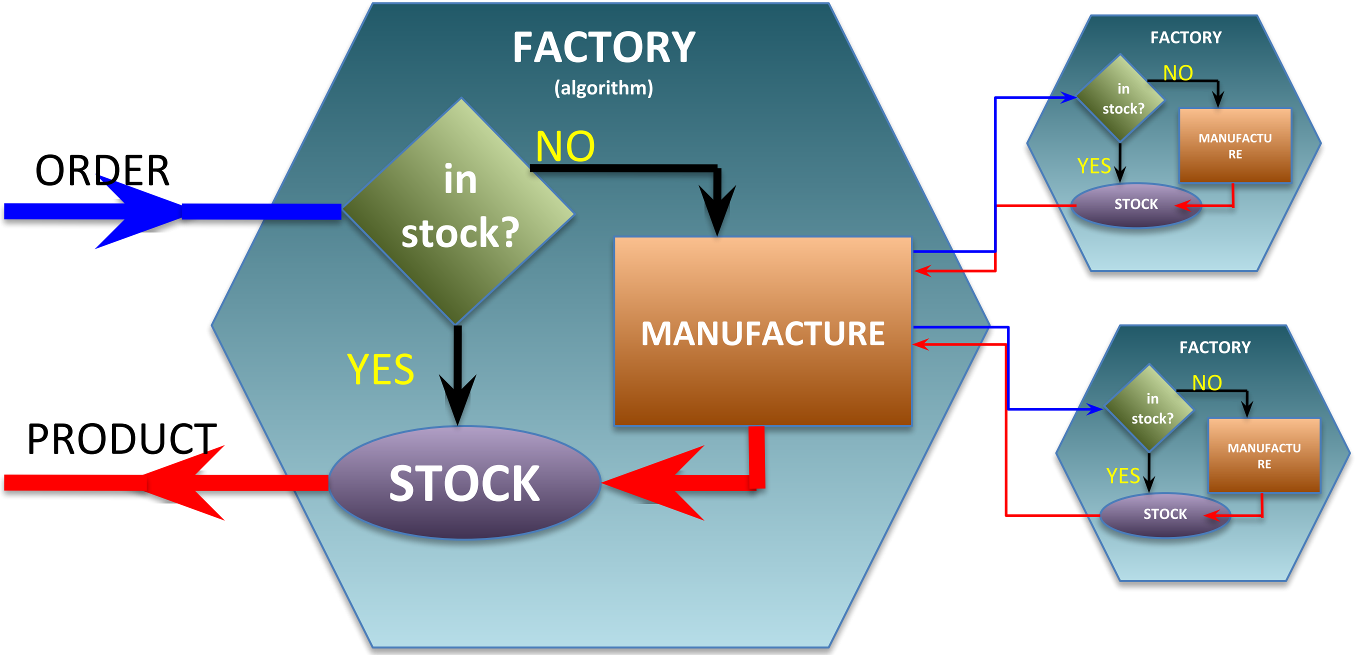
parallel arrow

sequential arrow

Reactive/Dataflow Programming

- Data is presented to arrow in the form of a queue
- Arrow transforms data and places it in downstream queue
- Minimal synchronization time spent in accessing queues
- Course tasks within arrow can eliminate most or all other synchronization points

Factory Model

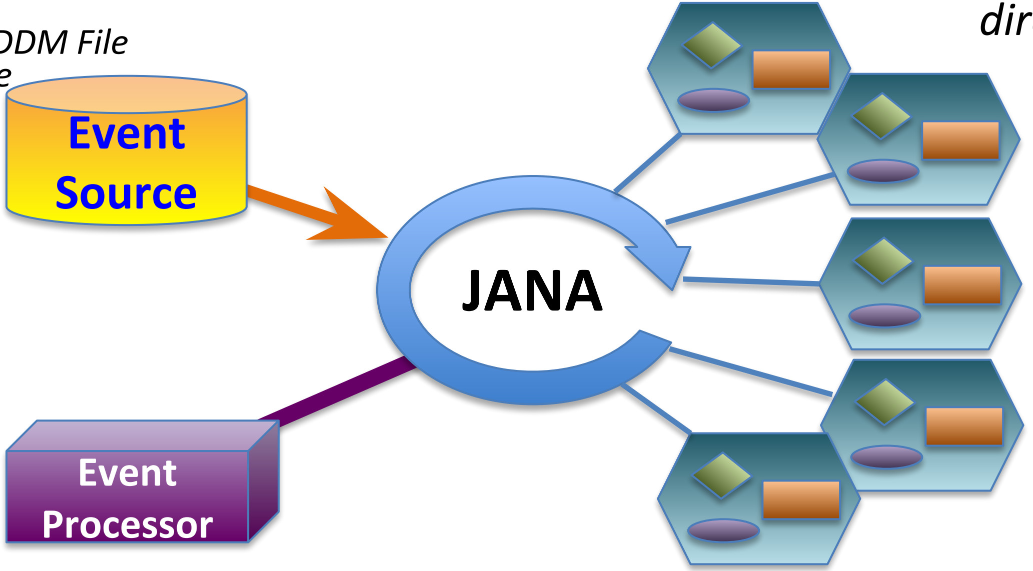


Data on demand = Don't do it unless you need it
Stock = Don't do it twice

Conservation of CPU cycles!

Complete Event Reconstruction in JANA

HDDM File
EVIO File
ET system
Web Service



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

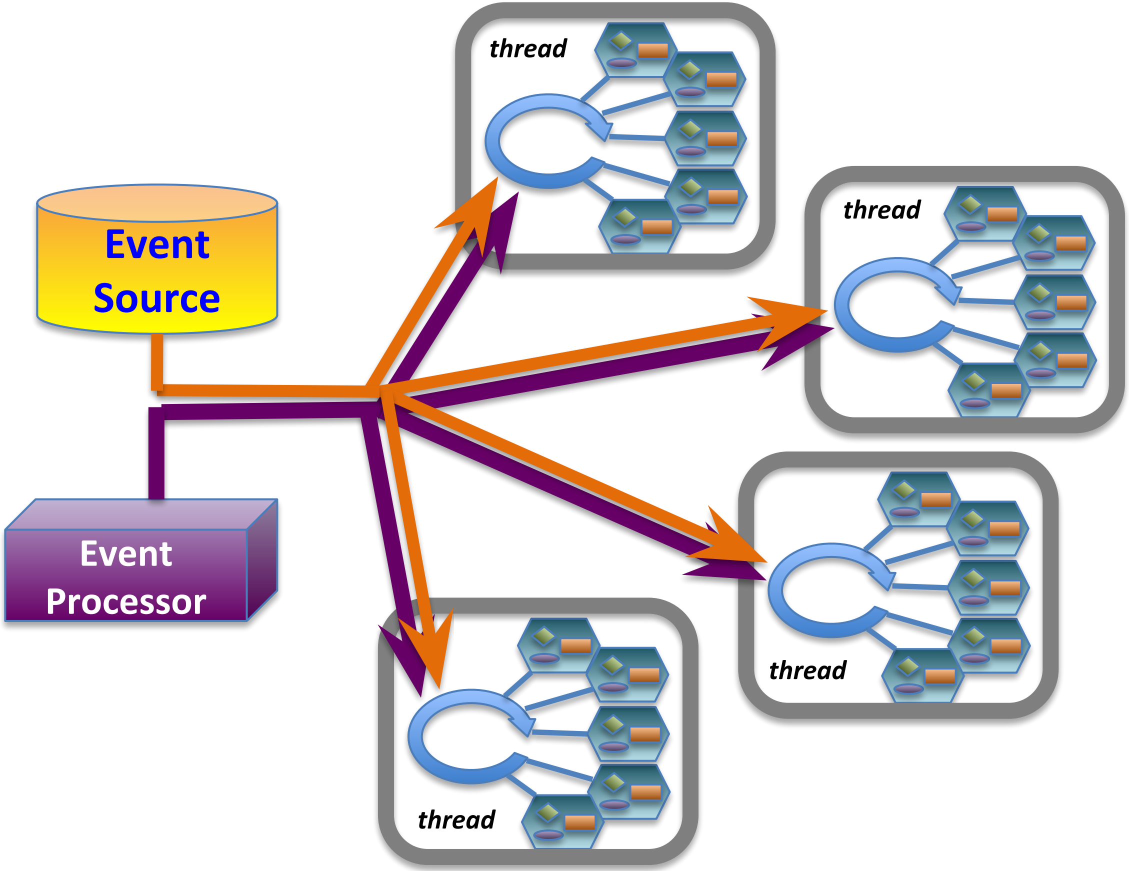
This allows the framework to easily redirect requests to alternate algorithms specified by the user at run time

Multi-threading

- *A complete set of factories is assigned to an event giving it exclusive use while that event is processed*

- *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)*



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 (introduced in c++11)
  - 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)

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*

Data on Demand => Software Trigger

Event by event decision on whether to activate a factory:

Software triggers may have multiple "keep" or "discard" conditions that may be probed in order of CPU cost

```
// Getting hit objects is cheap so we check that first
auto NcaloHits = jevent->Get<CaloHit>().size();
if( NcaloHits>minCaloHits ){

    keep_event = true;

// Tracks factory only activated if not already keeping event
}else if( jevent->Get<Tracks>().size() > minTrackHits ) {

    keep_event = true;

}
```

If an alternate factory is desired:

(i.e. algorithm)

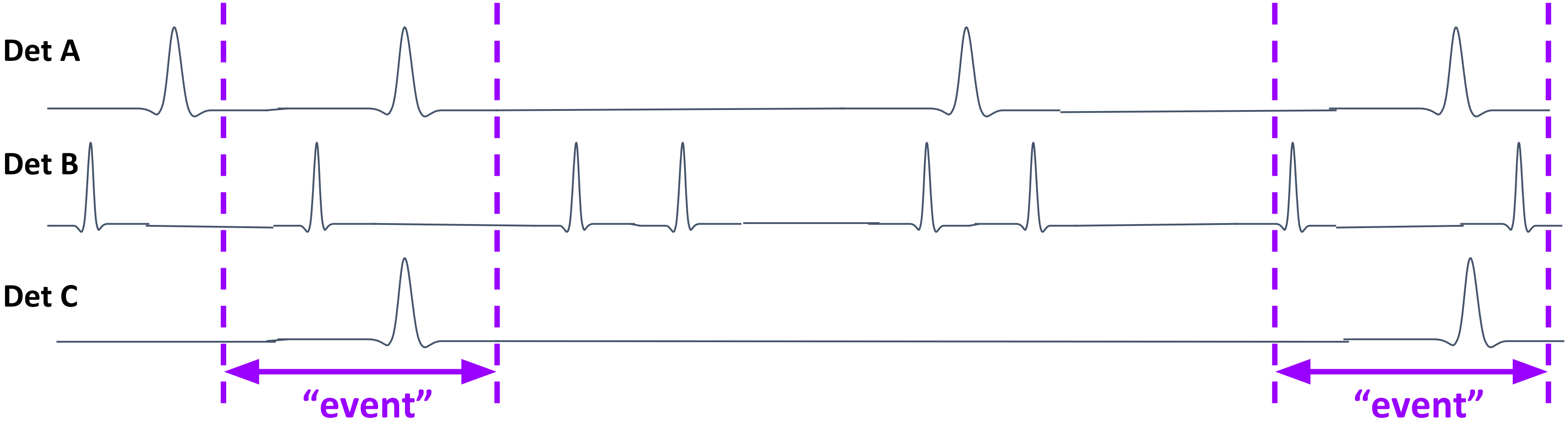
```
auto tracks = jevent->Get<DTrack>(“MyTest”);
```

or, even better

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

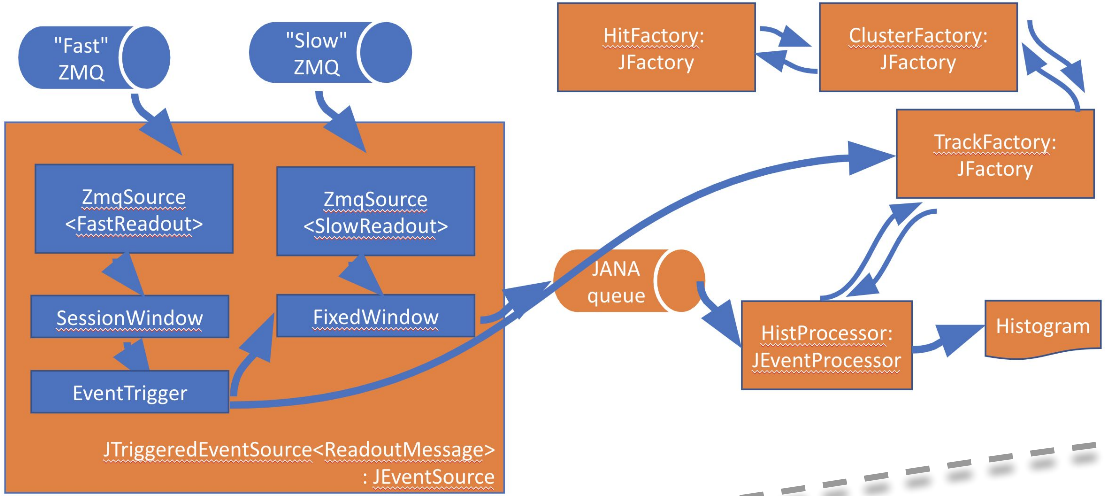
- Configuration parameters are set at run time
- NAME:DEFTAG is special and tells JANA to re-route ALL requests for objects of type NAME to the specified factory.

“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



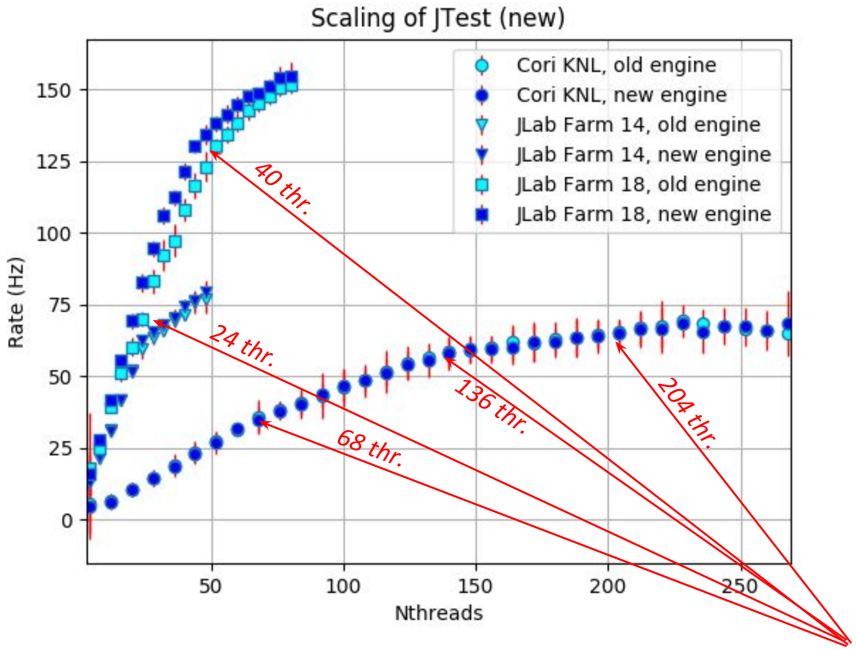
INDRA-ASTRA initiative:

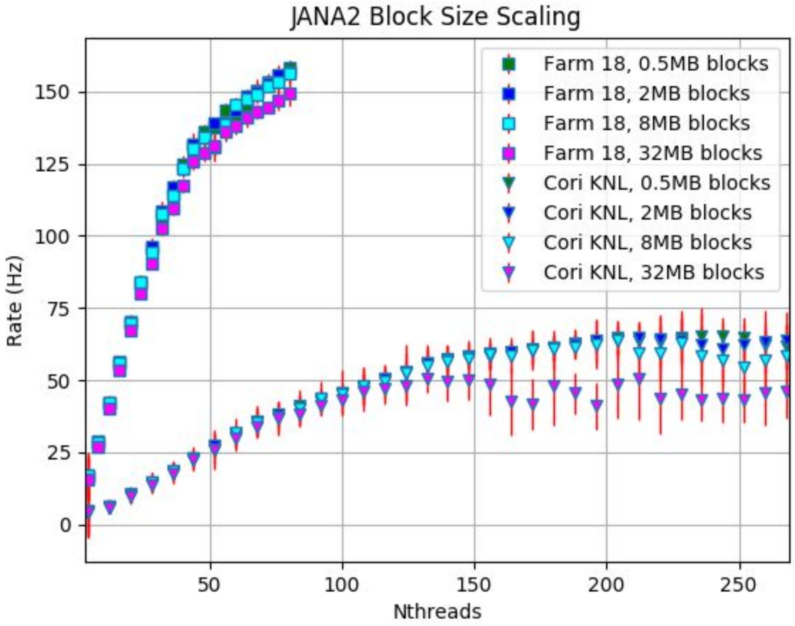
- Software trigger
- Multi-flavored stream merging
- Event building

Support for Heterogeneous Hardware

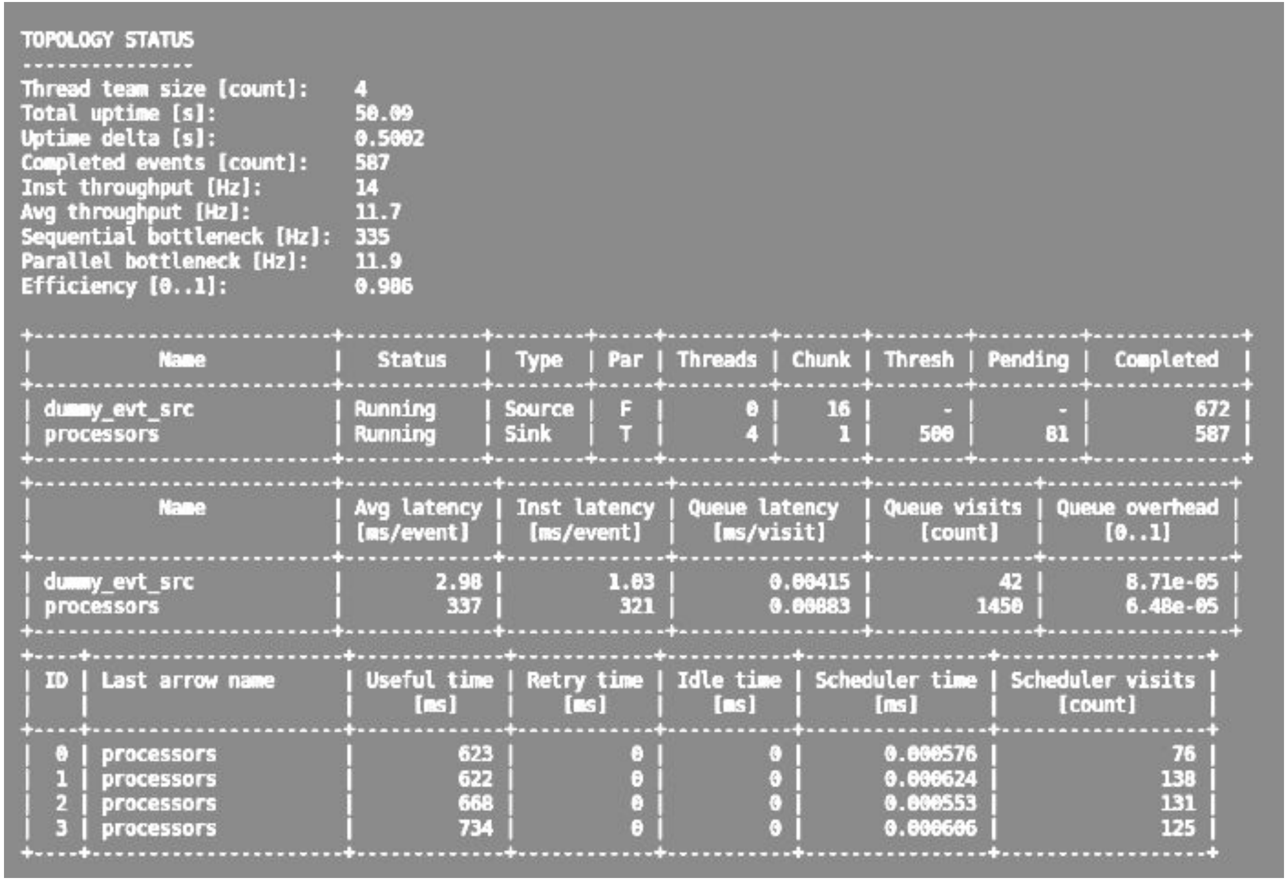
- Sub-event level parallelism
  - Run ML on GPU or TPU

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 compared to the original JANA.

This helps pinpoint bottlenecks, especially in more complex systems

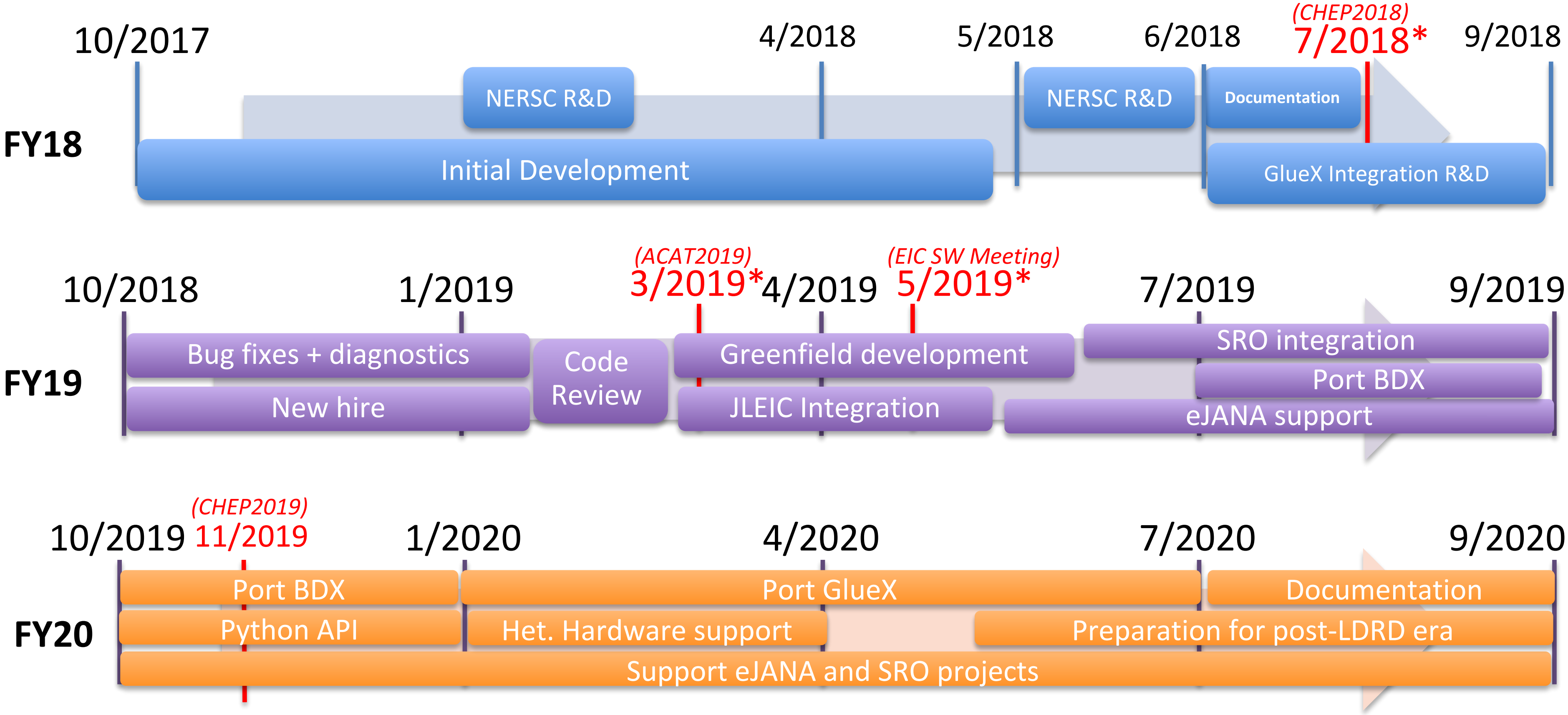
Summary

- JANA2 is:
  - C++ multi-threaded event processing framework
  - Reactive/dataflow programming model
    - arrow/queue architecture
  - On Demand algorithm activation
    - factory model *(lockless!)*
    - software trigger
  - builds on >10 years experience with JANA1
  - Python interface (embedded and extension)
- Follow project on github:

https://github.com/JeffersonLab/JANA2

Backups

Schedule



FY18

10/2017 4/2018 5/2018 6/2018 (CHEP2018) 7/2018* 9/2018

NERSC R&D

Initial Development

NERSC R&D

Documentation

GlueX Integration R&D

FY19

10/2018 1/2019 (ACAT2019) 3/2019* 4/2019 (EIC SW Meeting) 5/2019* 7/2019 9/2019

Bug fixes + diagnostics

New hire

Code Review

Greenfield development

JLEIC Integration

SRO integration

Port BDX

eJANA support

FY20

10/2019 (CHEP2019) 11/2019 1/2020 4/2020 7/2020 9/2020

Port BDX

Python API

Port GlueX

Het. Hardware support

Documentation

Preparation for post-LDRD era

Support eJANA and SRO projects

*Conference/Workshop presentations

GlueX Reconstruction Software

Automatic call graph generation using janadot plugin