Streaming Event Reconstruction with JANA2

Nathan Brei, David Lawrence, Amber Boehnlein

9 July 2019

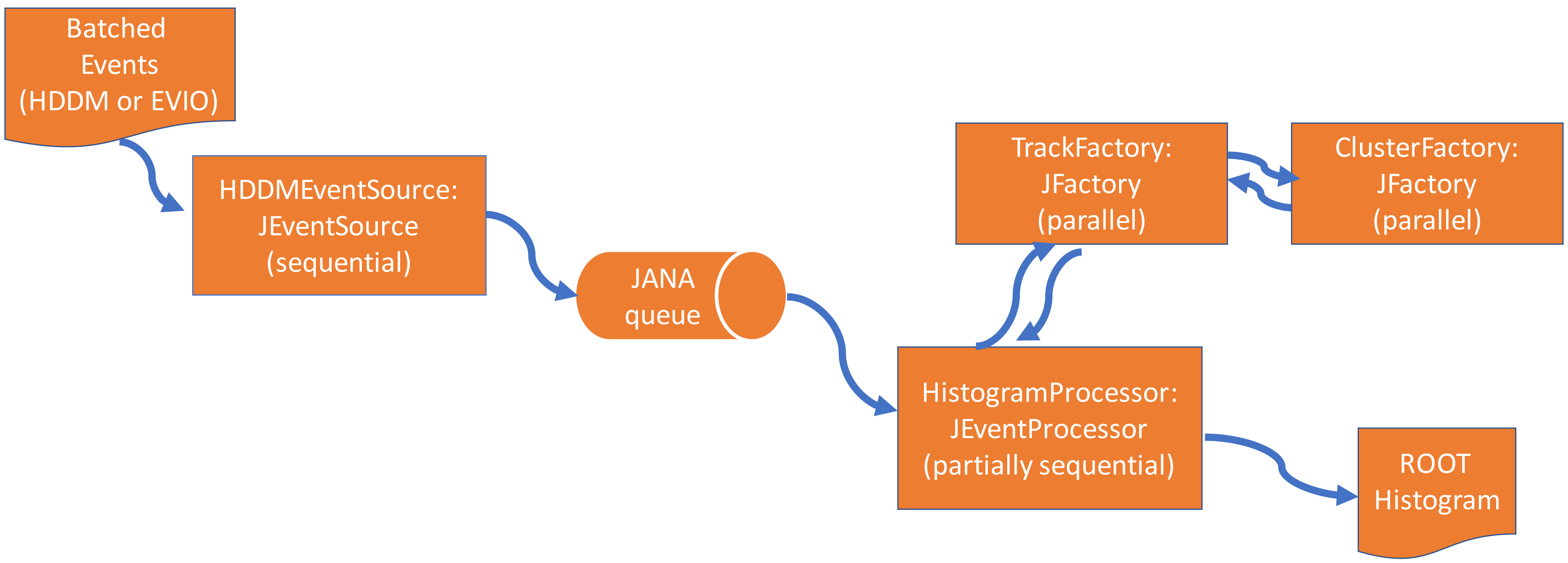
What is JANA?

- A modern C++ framework
  - Parallelizes event reconstruction across threads (single-node)
  - Provides a plugin architecture for organizing scientific code into decoupled components
  - Intermediate results get calculated at most once ("lazy + memoized")
  - Lightweight and close to the hardware

- Internally uses a non-blocking streaming model
  - Avoid waiting on locks, swap out different scheduling algorithms
  - Optimize for manycore and NUMA architectures, e.g. NERSC
  - Self-report parallel performance and bottlenecks
  - Semantics are similar to Kahn Process Networks

- Used by GlueX, EIC, BDX, etc

JANA(1+2) toy example (batch processing)



JANA(1) vs JANA(2)

In JANA(1), the fundamental unit of parallelism is a (physics) event. This is sufficient most of the time, but doesn't fit in several key areas:

- Parsing/disentangling
- Subevent-level parallelism
- Streaming data readout

- The JANA(2) engine now supports these use cases
- The next challenge is extending the API to expose this functionality
- General goal is to preserve existing semantics, and avoid making the simple use cases more complicated

Streaming Data Readout with JANA

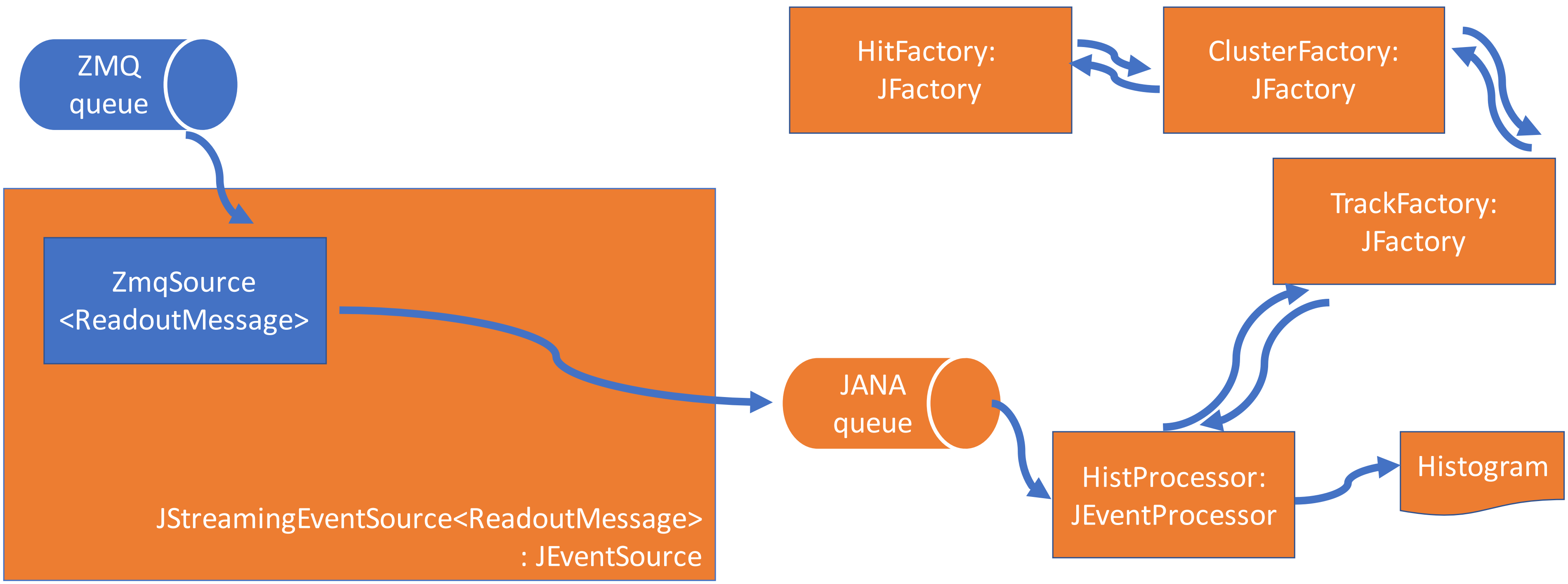
What makes this interesting:

- Detectors emit "Hits" (:= a value indexed by timestamp and detector_id), whereas JANA processes "Events" (:= a collection of values across all detector_ids within some timestamp interval)
- => We can do event building in JANA!
- In general this is called stream windowing, and it is closely related to an SQL JOIN
- The JANA engine needs some kind of stream windowing whenever two streams are merged

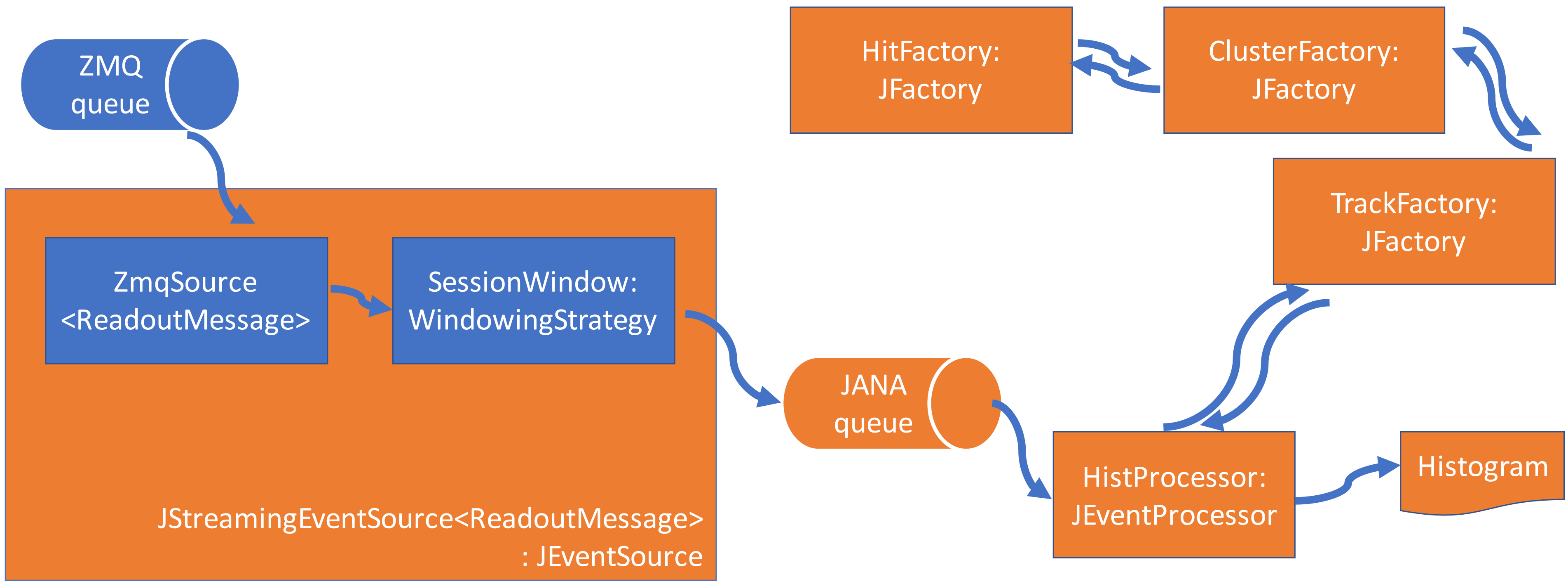
Design goals:

- Support streaming as an optional plugin, but use it to inform API improvements
- Keep deserialization, transport, and windowing orthogonal to each other
- Make these components reusable
- Keep JANA responsible for thread-level parallelism; use ZeroMQ or Kafka or xMsg for node-level parallelism

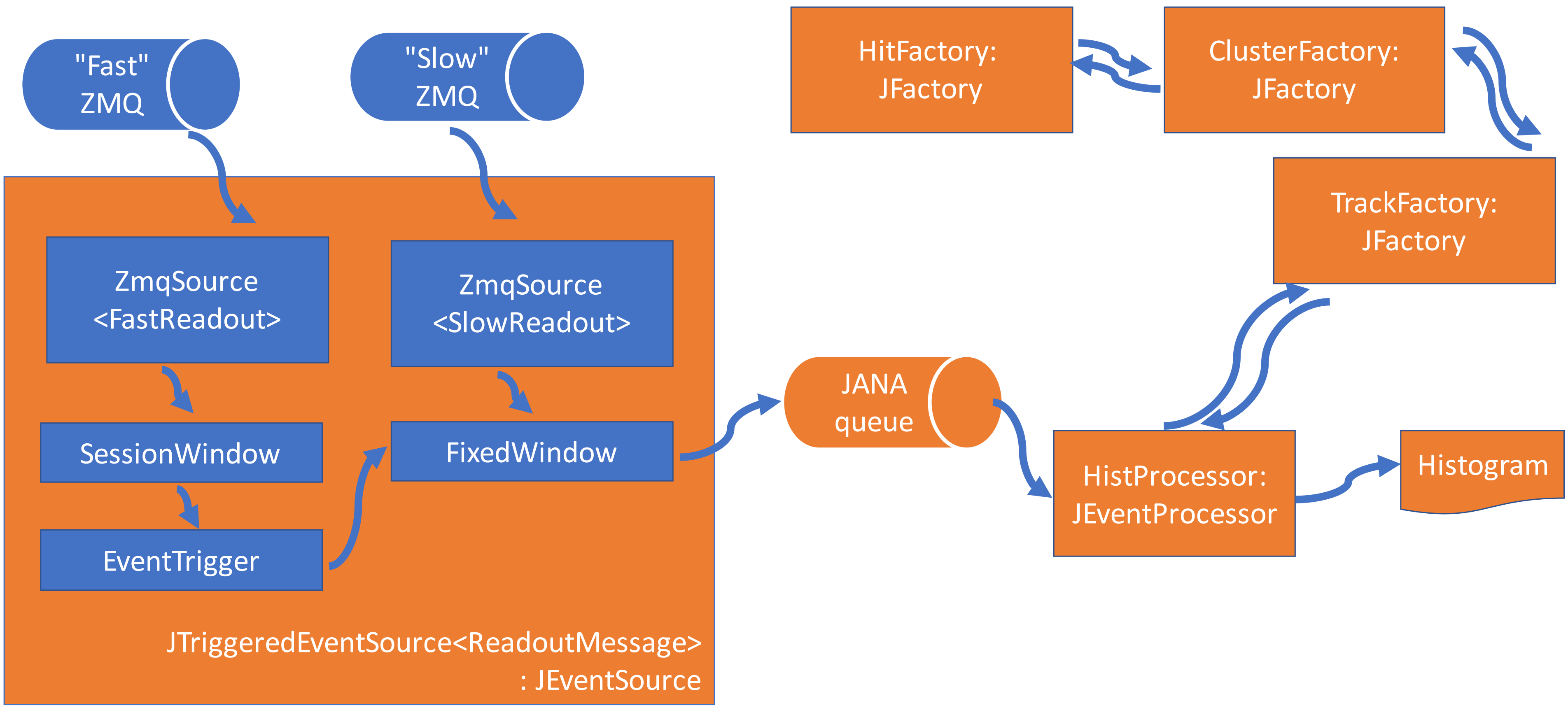
1. Streaming Data Readout, no event building



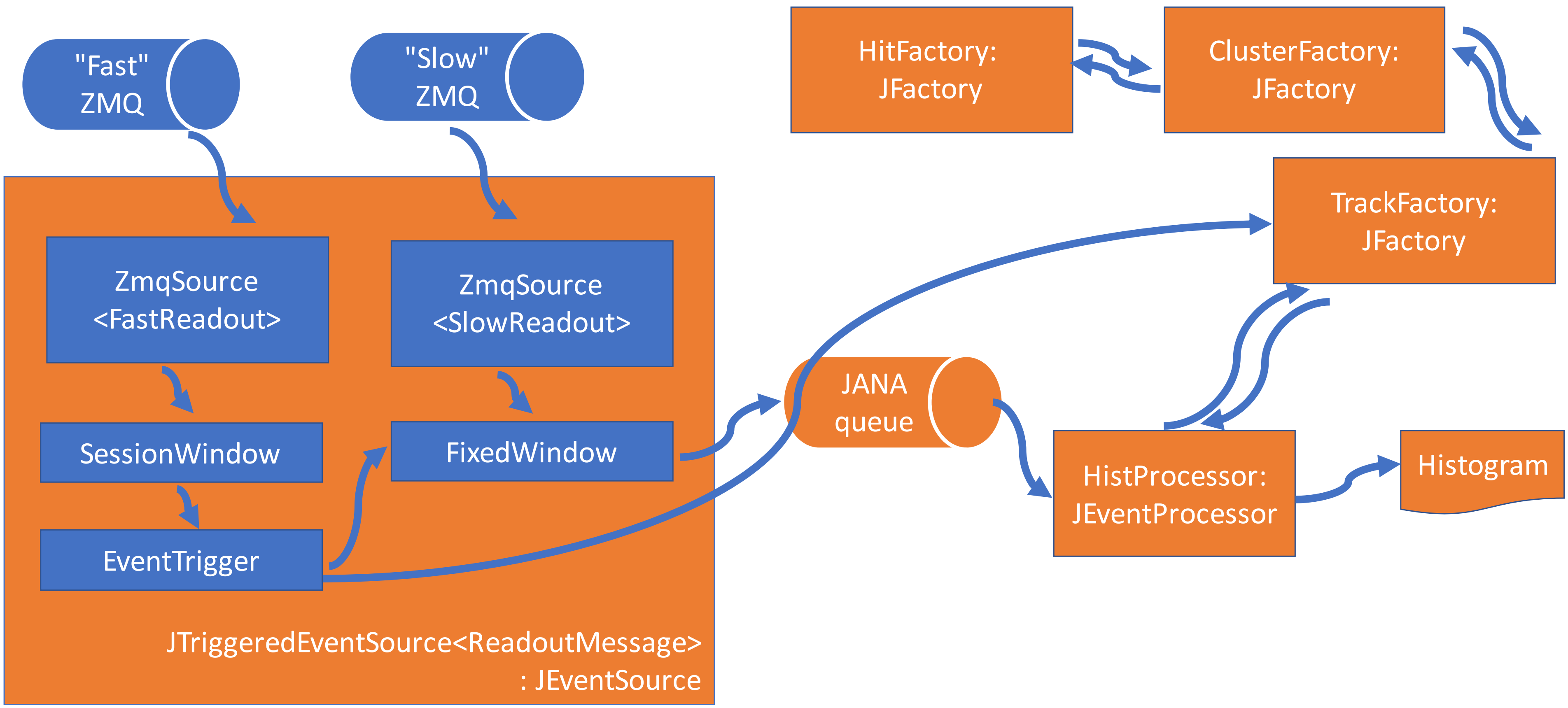
2. Streaming Data Readout, with event building



3. Streaming Data Readout, with software trigger



3. Streaming Data Readout, with software trigger



Next steps

- Short term
  - Demonstrate/perf test using JANA in a streaming context
  - Integrate control messages, e.g. change of run number
  - Develop reusable abstractions for streaming event sources
- Medium term
  - Support the INDRA-ASTRA streaming readout LDRD (Markus, Graham, & Eric)
  - Evolve JANA to support this as cleanly as possible
  - Open question: How does one ensure memory safety when working with time-indexed, memory-pooled, user-defined types?
- Long term
  - xMsg+JANA as a streaming/reactive analogue to MPI+OpenMP

Thank you!

Why do triggering inside JANA?

- Code for doing reconstruction can be used for triggering without modification.
- Any results calculated for the triggering are automatically propagated downstream to the reconstruction.
- Parallelization of triggering can coordinated with parallelization of reconstruction.
- Tradeoff between bounding latency and balancing load can be explored by tuning scheduler parameters.
- Caveat: This only scales up to a point, after which we would have to use node-level parallelism as well.

Arrows-and-Queues engine

- Directed acyclic graph of queues and arrows
- Arrows pop data from an input queue, compute something, and push new data onto an output queue
- Details:
  - Each worker thread is assigned an arrow from a scheduler and attempts to execute it
  - If the pop() fails, the arrow execution will fail rather than block
  - If the pop() succeeds, the push is guaranteed to succeed without needing to block
  - Backpressure is maintained by reserving space on the output queue before popping
  - Hybrid push-pull semantics cleanly handle critical sections
  - This is similar to a formalism called Kahn Process Networks
  - The general solution space is called 'reactive' or 'dataflow' programming