Wire-Cell Software

Brett Viren

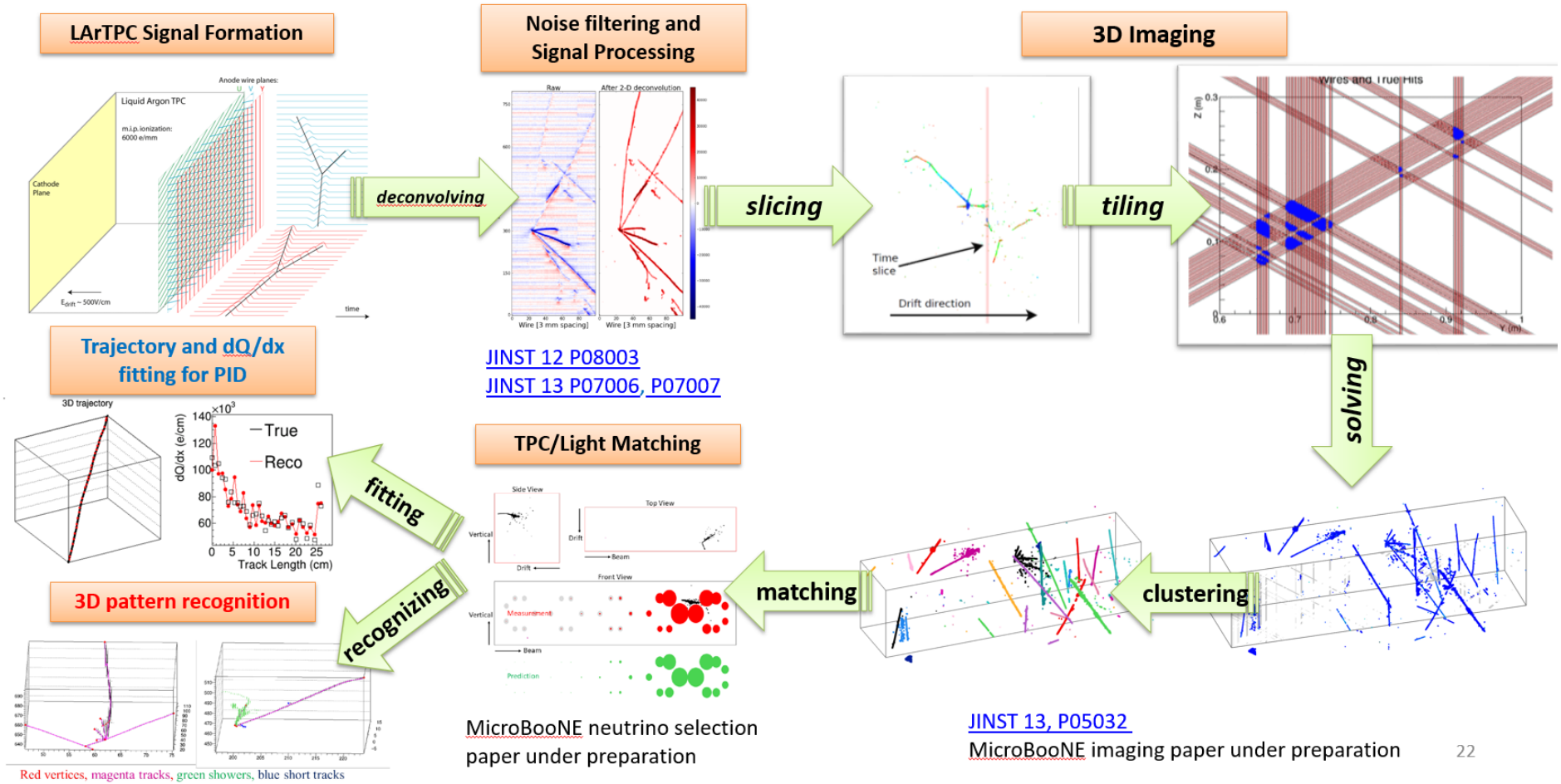
Physics Department

BROOKHAVEN
NATIONAL LABORATORY

DUNE S&C @ BNL – Jan 2019

Outline

Wire-Cell Algorithms and Processing Chain



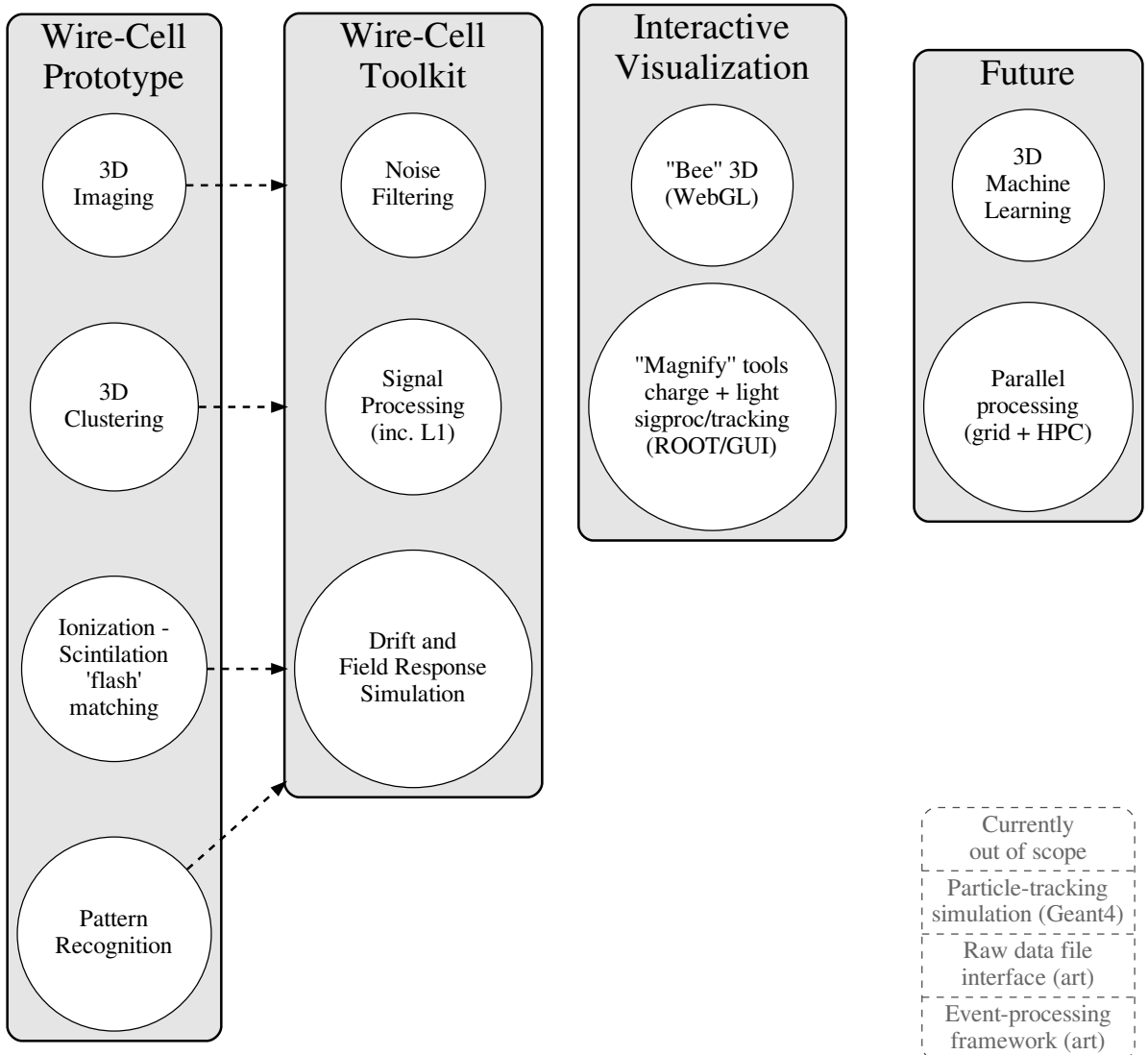
JINST 12 P08003
JINST 13 P07006, P07007

MicroBooNE neutrino selection paper under preparation

JINST 13, P05032
MicroBooNE imaging paper under preparation

Xin Qian

Wire-Cell Software Algorithm Scope



Prototype and Toolkit Code Bases

Common:

- Similar build systems and external dependencies.
- Open source repositories on GitHub.

Prototype:

- Lightly structured code base, various `main()` programs, freedom to experiment without many "rules", comes with "no" user support, no releases. Initial proving ground for eventual toolkit code.
- Used by MicroBooNE results.
- Top package: `https://github.com/BNLIF/wire-cell`

Toolkit:

- Structured/designed code base, careful dependency control, toolkit-style integration to user's app, shared library plugins, configuration subsystem, optimized code, production releases and support.
- Used by MicroBooNE and ProtoDUNE.
- Top package: `https://github.com/wirecell/wire-cell-build`

Wire-Cell Software Overview

Wire-Cell Toolkit

WC/LS/*art* Integration

Strategy and Discussion

Wire-Cell Toolkit Packages

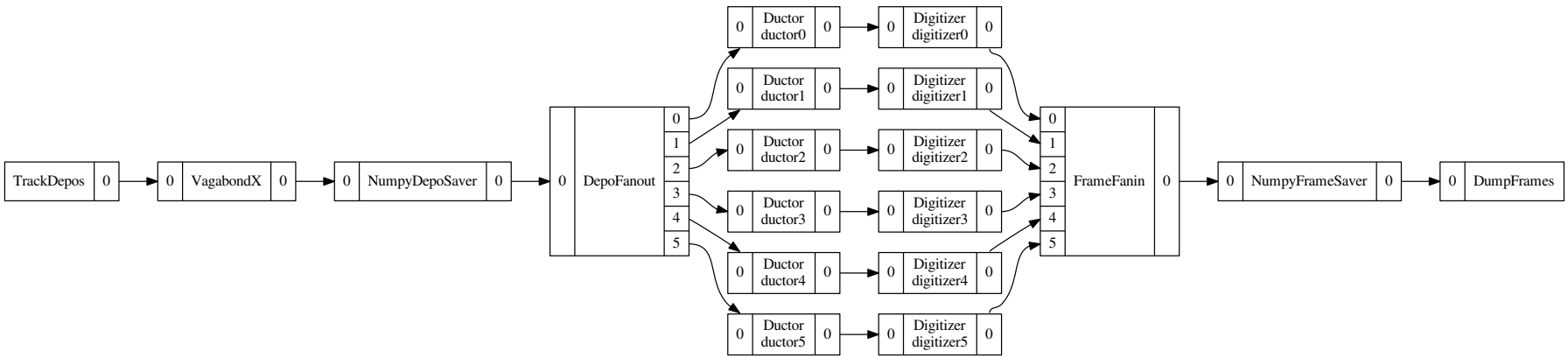
`util`	fundamental data types, operations, toolkit infrastructure.
`iface`	abstract "interface" base classes for WCT components.
`cfg`	reference configuration files (`.jsonnet` and `.fcl`)
`data`	larger config "data" files (`.json.bz2`)
`gen`	components for electron drift and field response signal and noise simulation.
`sigproc`	components for noise filtering and signal processing (field response deconvolution and L1 regularization)
`sio`	components to provide various I/O (depends on ROOT).
`python`	utility, debugging, analysis, config data file prep.
`pgraph`	single-thread, low-memory execution model implementation.
`tbb`	experimental multi-threaded execution model implementation.
`sing`	Singularity image creation and user scripts.
`docs`	news blog, manual, presentations and other documentations.
`tests`	larger-than-unit tests
`waftools`	build system support.

Layered stack of user-code entry points

`wire-cell` CLI	`art` CLI Wire-Cell/LArSoft integration
Wire-Cell plugin/shared libraries	
construct data flow processing graph via user configuration	
use abstract component interfaces via factory lookup	
`#include "implementation.h"` / use concrete WC classes	
implement new concrete component classes	
design new interface base classes	

Aside: Data Flow Programming Paradigm

Programming = drawing: construct a **directed graph** of processing **nodes** with labeled **ports** connected by **edges** which transfer data objects.

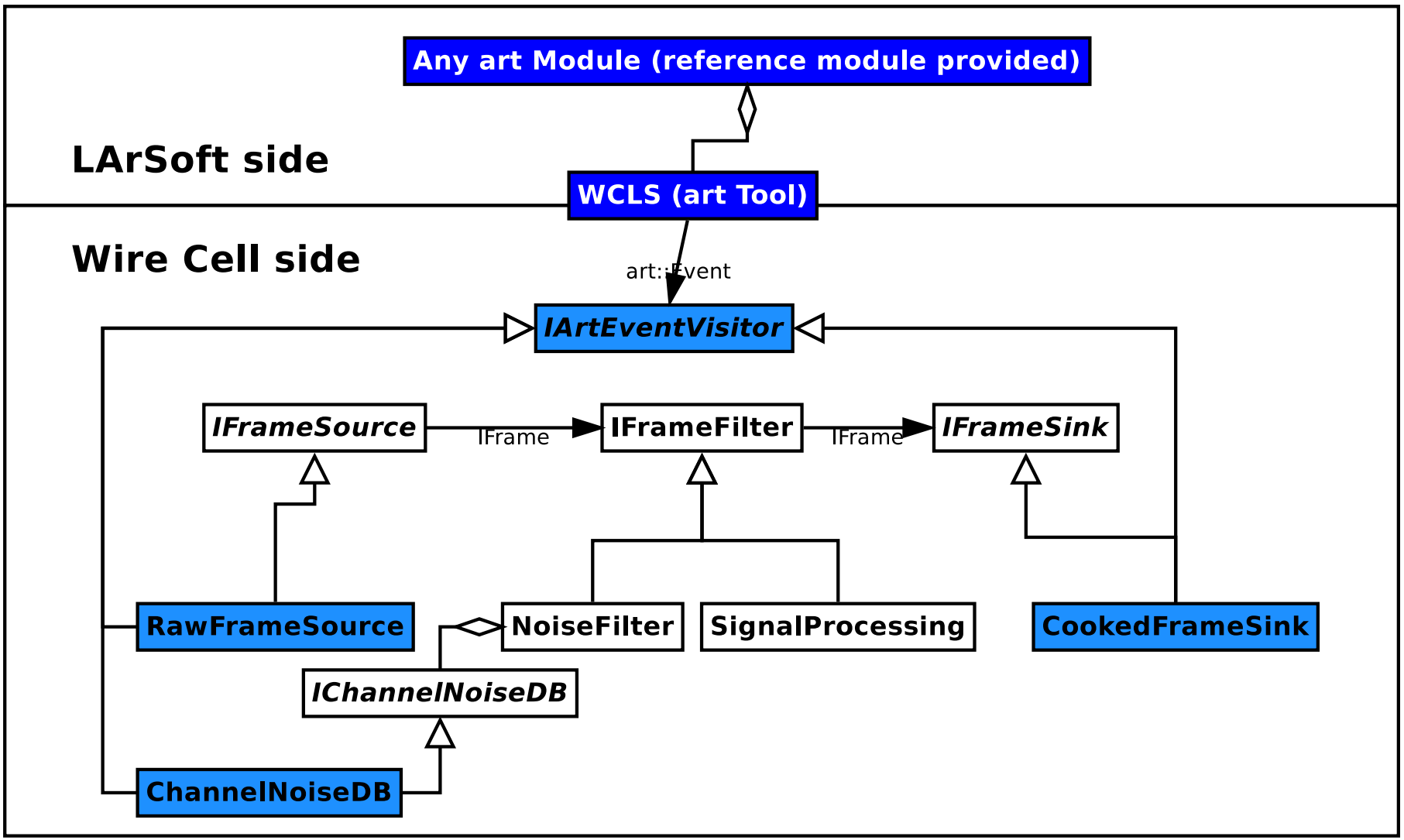


- Nodes may contain state, edges may buffer intermediate results.
- Stateless nodes and thread-safe edges allow for multi-threading.
- Edges may be generalized to allow multi-node communication.
- Edge data may be fine-grained (ie, objects smaller than one "event").
- Different graph execution strategies may be employed.

WCT Implements DFP Paradigm

- Technically optional but all "real" jobs defined in terms of a DFP graph.
- WCT Jsonnet configuration supports **scale invariant** graph construction.
  - Complex subgraph description can be encapsulated into a single node object, parameterized and re-used.
- Abstracted graph execution supports different execution strategies:
  - Primary engine: single-thread with memory-minimizing.
  - Experimental: TBB-based multi-threaded, CPU-maximizing.
  - Possible future: multi-thread+multi-node with ZeroMQ or MPI?

WC/LS Integration Design



WC/LS Design In Words

- `WireCellToolkit_module` provides reference *art* module.
- `WCLS_tool` provides *art* Tool interface almost exactly like `wire-cell` command line interface.
- A pure-WCT DFP graph may be rewritten to a WC/LS graph simply by replacing its data sources/sinks with corresponding WC/LS **converter components**.
  - "visit" the `art::Event` before or after module execution.
  - convert data or provide interface to *art* services.
- High-level WCT configuration specified in FHiCL
  - Corresponds to what may otherwise be specified to `wire-cell` CLI
  - Typically, a few select "external variable" parameters set in FHiCL
  - Additionally, must specify list of converter components.

WC/LS Status

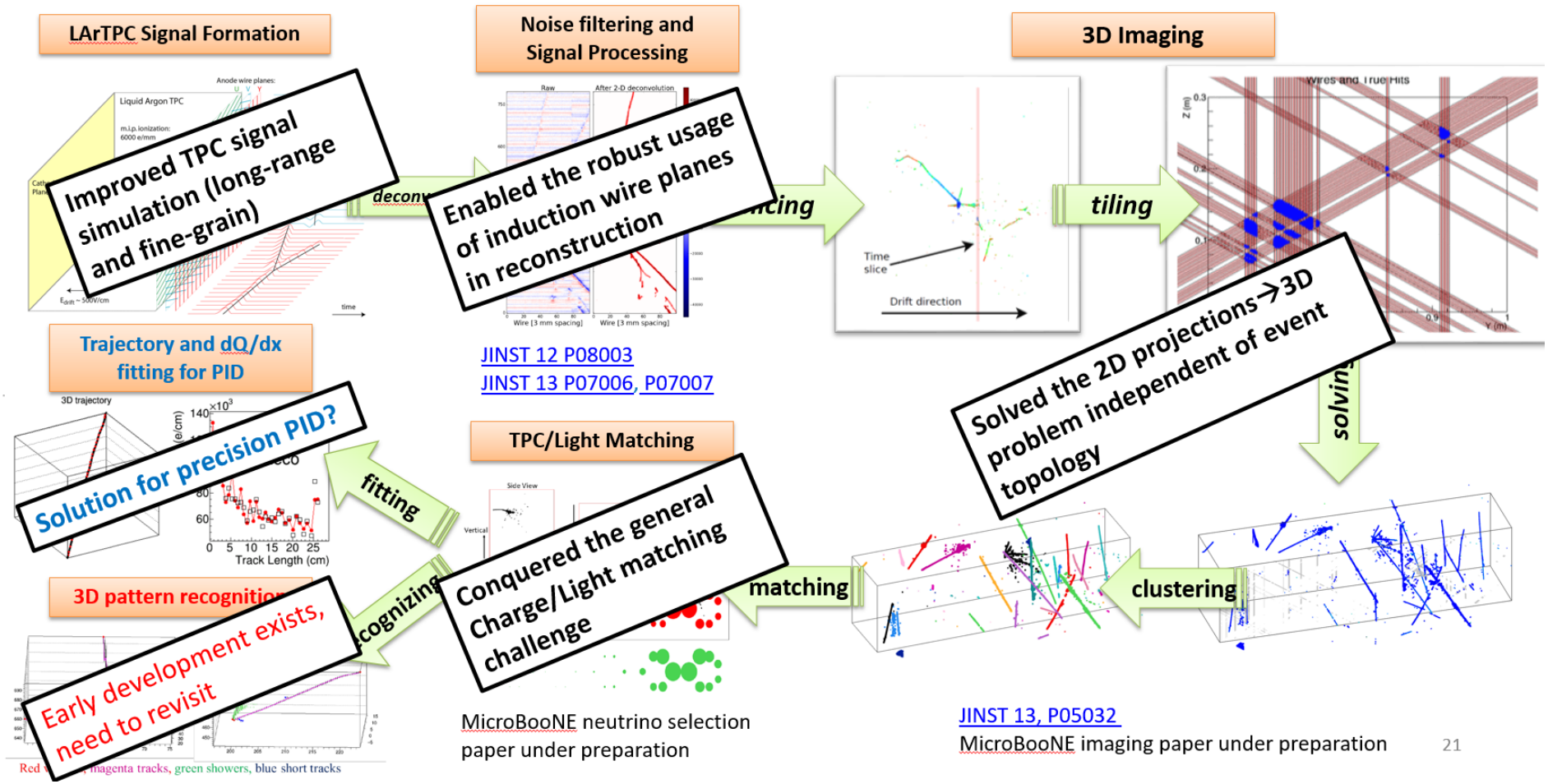
- WCT is now standard for MicroBooNE noise filtering and signal processing.
- WCT simulation integrated and tested to consume LArG4 energy depositions, produce raw ADC waveforms.
- Multi-APA support added to above to support ProtoDUNE-SP.

WCT Strategy For DUNE

1. Continue to advance algorithms and port to toolkit
2. Add support for per-APA processing
   - Relatively straight-forward in WCT
   - But, need a per-APA "loop" all the way to the input file (ie, *art* support).
3. Leverage WCT design to exploit multi-core platforms and reduce RAM/core.
4. Understand if multi-core GRID jobs are sufficient for DUNE production data and simulation processing and push into HPC space if not.

Some more on each point on following slides.

Wire-Cell Algorithm Advancement



JINST 12 P08003
JINST 13 P07006, P07007

MicroBooNE neutrino selection paper under preparation

JINST 13, P05032
MicroBooNE imaging paper under preparation

Xin Qian

Porting Prototype Algorithms - 3D Imaging

- Existing 3D prototype code needs understanding by others (started).
- New, more general data model needed (conceptual).
- Develop optimized primitive operations (conceptual).
- Initiate new eg `wire-cell-img` package.
- Initial port of algorithms should generically support MB, PDSP and others.
- Benchmarking, validation, tuning....

Per-APA execution model

- Full "event" in memory already requires substantial RAM
  - MicroBooNE/ProtoDUNE-SP easily break the 2GB/core Grid limit.
  - WCT recently reduced its footprint but still processes whole-event
  - art's ROOT I/O overhead tends to dominate
  - 150 DUNE APAs in memory at once is untenable
- SigProc output is sparse, expect $\approx 10^4$ reduction for DUNE
- Input is dense: break processing down to per-APA units.
  - WCT can define per-APA pipelines, some work needed.
  - Ultimately, only matters if *art* loads in $\mathcal{N}$ APAs at once

Discussion:

→ DUNE needs to discuss with *art*/LArSoft experts how to achieve a per-APA "event" loop.

→ DUNE should configure production jobs to be more selective as to what data types are "shunted" input→output.

- Don't carry forward "dense" `RawDigits`.

protoDUNE WC/LS SigProc memory usage

6-APA event

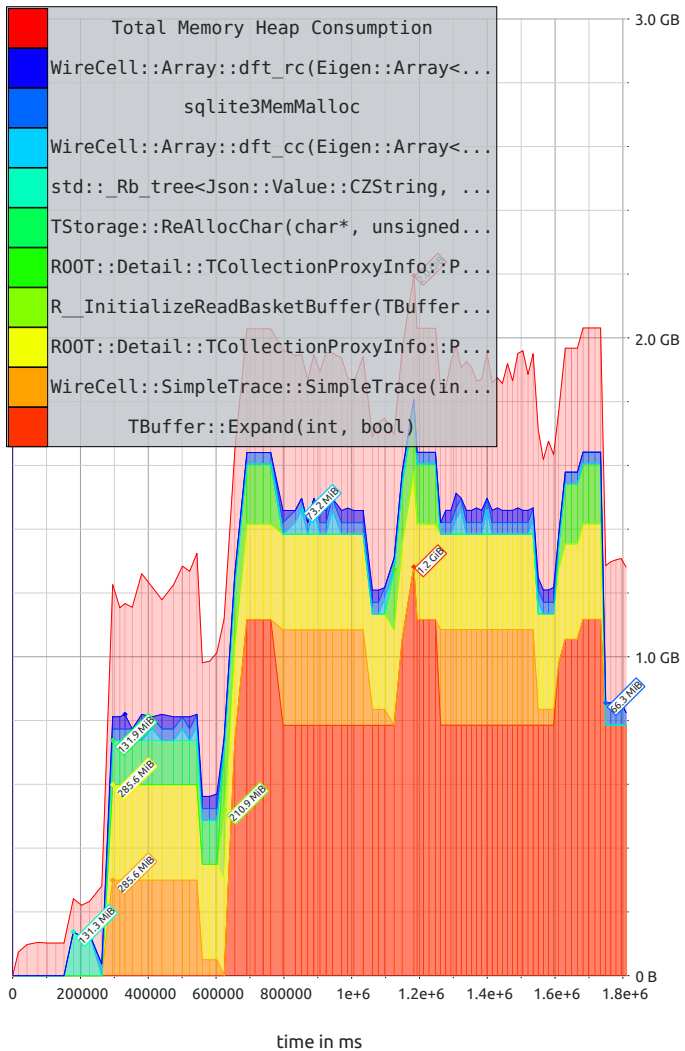
After recent ≈ 500 MB reduction inside WCT

ROOT output

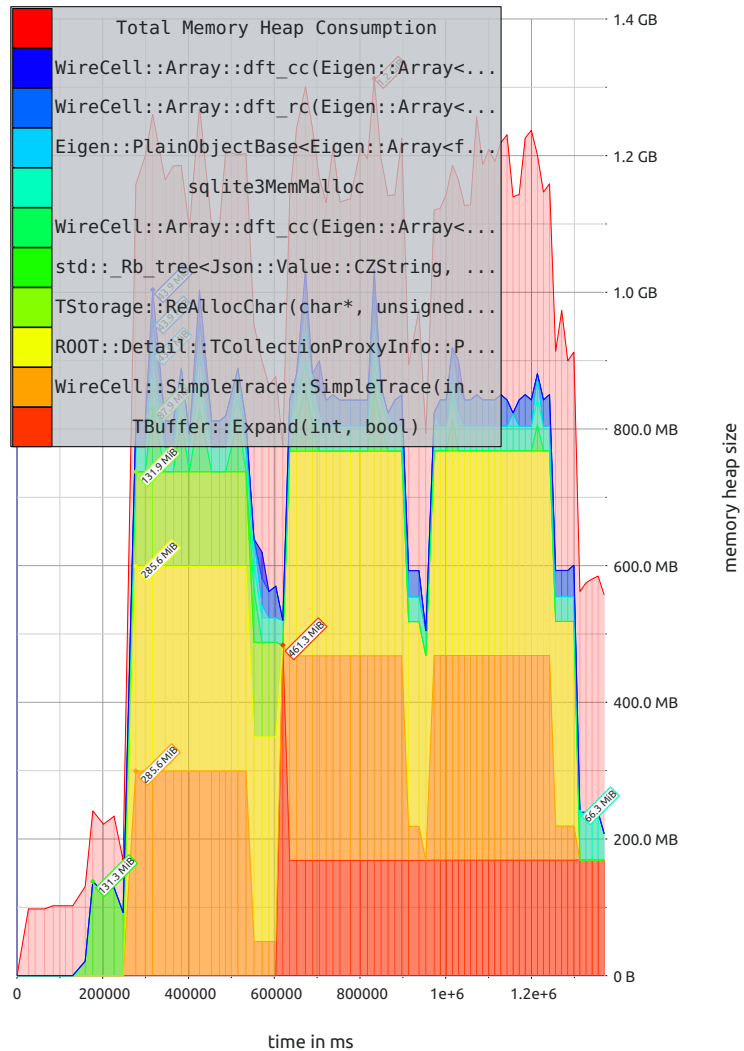
WCT sigproc

ROOT input

Copy all input to output (2GB, 30min).



No unneeded I/O copy (1.2GB, 23min).



Multi-threading

- WCT has existing TBB-based multi-threaded DFP graph execution engine.
  - o May be "trivial" or maybe surprises.
  - \+ No mutable "globals", `const` data objects so should be in good shape.
  - ? But, many node components exist now and with little care for thread-safety.
- Evaluate TBB execution performance and memory usage
- Consider adoption/development of alternative engines.
- Understand how far we can go on multi-core grid allocation alone.
- Test the HPC waters.

Thoughts on *art* Multi-threading

- *art* is multi-threaded but "only" at the module/path level,
  - This useful feature seems not yet exploited by the experiments.
  - → DUNE should try it!
  - ? Only useful to the extent that module paths are actually parallel.
- MT success and per-APA execution are linked.
  - Ideally, we want concurrent, per-APA pipelines.
  - These will require "event" level synchronization at least at job output.
- Must treat parallelism "holistically"
  - A highly parallel module (eg WCT) in an otherwise roughly serial *art* job wastes CPU.
  - → See if *art* team can add pipelining (multiple events "in flight").
  - WCT components may be pipelined "for free" depending on the execution engine.
  - BNL PAS group's "event server" technique may also be useful?

Multi-Core Grid vs HPC

It's not yet clear to me if DUNE **really** needs HPC.

? Would achieving good CPU efficiency on Grid be enough?
- How much CPU-years does DUNE need?

\- Supporting HPC will take work:
- must greatly reduce RAM/core, support fine(ish) grain MT,
- understand and obey special software environments (I'm told: "no ROOT")
- infrastructure, data ingest/egress, databases, security issues.
- in some cases deal with "unusual" CPU architecture and OS.

+/- We would be minor player on HPC

++ HPC power could open up new algorithms

? do we make the support effort just "in case"?