DL Clustering Pipeline

Wirecell Meeting

Avinay Bhat
08/13/2026

Why streamline the DL Production Pipeline?

- Old workflow was scientifically valid but file-heavy
- Reconstruction, labeling, conversion, and training were separated by large handoff files
- This increased I/O, storage, orchestration, and provenance risk
- Goal: run production on Polaris /lus/eagle, not through repeated sbndgpvm file handoffs
- Central principle: change data transport, not scientific content

Old vs New Architecture on Polaris

- Old: write EventGraph to disk, then Python reads it back
- New: stream completed EventGraph directly from C++ to Python
- Runs inside a PBS production job on Polaris/Lustre-Eagle
- One ART process, one persistent socket, one Python receiver, one final H5 writer
- Final output lives on /lus/eagle; no intermediate event-data handoffs

Streaming Implementation

- Implemented EventGraphIPC in larwirecell
- Protocol: SBNDIPC v1
- Messages: EVENT_GRAPH, ACK, ERROR, END_OF_STREAM
- Python receives named arrays and calls the same canonical adapter:
  - build_apa_graph()
  - StreamingH5Writer
- hdf5_output = true: build EventGraph
- hdf5_file_output = false: do not write old intermediate H5

Polaris/Lustre-Eagle Production Contract

- Production runs on Polaris-side /lus/eagle project storage

- Allowed outputs:
  - final canonical NuGraph H5
  - <output>.partial during streaming or after failure
  - socket pathname, logs, metrics, manifest, config

- Not allowed as handoffs:
  - TensorSetLabeler nugraph.h5
  - truth JSON / CellTree ROOT / NPZ
  - temporary chunk H5
  - /tmp or /dev/shm event-data substitutes

- Bee archive or final ART ROOT may be terminal diagnostics only

Validation, Forks, and Production Status

- Validation ladder passed from raw IPC to 50-event production
- 50 events → 100 APA samples
- 3,800 / 3,800 canonical fields exact
- 1,000 / 1,000 topology comparisons exact
- Streaming H5 SHA256 matched frozen file-based reference
- Final H5 was bit-for-bit identical
- Code preserved on abhatfnal personal GitHub forks:
  - WCT: abhat/wire-cell-toolkit
  - larwirecell: abhat/larwirecell
  - NuGraph streaming: abhatfnal/nugraph

Time and Memory Footprint

- Persistent no-intermediate NuGraph production — 50 physical events
- 50 events → 100 APA samples → one final canonical H5
- Wire-Cell + streaming production time: 18.0 min
  - lar wall time: 1082 s
  - ≈ 21.6 s/event
- Full PBS job wall time: 20 min 42 s
  - includes WCT build/test, larwirecell build, startup, reconstruction & finalization
- Peak reconstruction memory: ~1.8 GB
  - /usr/bin/time max RSS: 1.84 GB
  - ART VmHWM: 1.89 GB
- Python streaming receiver: ~0.56 GB
  - 563 → 578 MB high-water mark over 50 events
- Final H5: 69.5 MB
  - 100 samples, train/val/test = 76 / 14 / 10
  - No intermediate event-data files

Back Up Slides