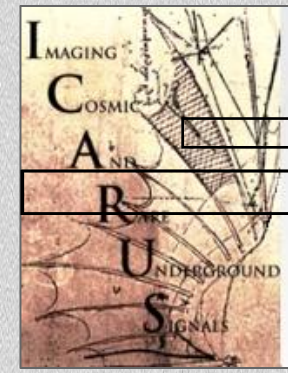


ICARUS Software

Imaging Cosmic And Rare Underground Signals

ICARUS Multi-threading Production Workflow

Tracy Usher (SLAC)

LArSoft Multi-threading and Acceleration Workshop

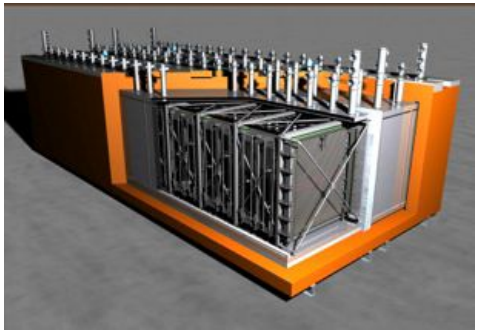
March 2-3, 2023

Overview

- Brief introduction to the ICARUS detector
- Quick review of ICARUS Processing Chain
  - Focus on TPC signal processing
- Early Standalone Tests of Multi-Threading of TPC Signal Processing Chain
- Is ICARUS TPC signal processing a candidate for multi-threading?
  - Echoes of Scrooge: Bah! Humbug!
- Path Forward

Brief Introduction to the ICARUS Detector

- ICARUS largest of currently operational LArTPCS:
  - 760 tons LAr total volume
  - Active volume 476 tons
- ICARUS consists of 2 Cryostats situated side-by-side
  - Each Cryostat contains 2 TPCs arranged back-to-back (mirror reflections)
- Each TPC has:
  - 1.5 m drift
  - 3 wire planes
  - ~55000 channels
- In Addition:
  - 360 8” PMTs total (90/TPC)
  - Cosmic Ray Tagger (~95% coverage)
  - Overburden (2.7m concrete, 6.5 mwe)



- Main ICARUS TPC differences:
  - First induction plane consists of horizontal wires - split at the longitudinal TPC center
  - Middle induction and collection planes consist of U/V oriented wires depending on TPC
  - Total uncompressed data volume for reading all four TPCs: ~440MB/event
  - Readout electronics are warm

General ICARUS Processing Overview

There are essentially 3 stages to the ICARUS event processing:

1. Signal Processing (“Stage 0”)
    a. Converts raw signals from detector to LArSoft data products
    b. Final level is “hit” information as the basic input to reconstruction

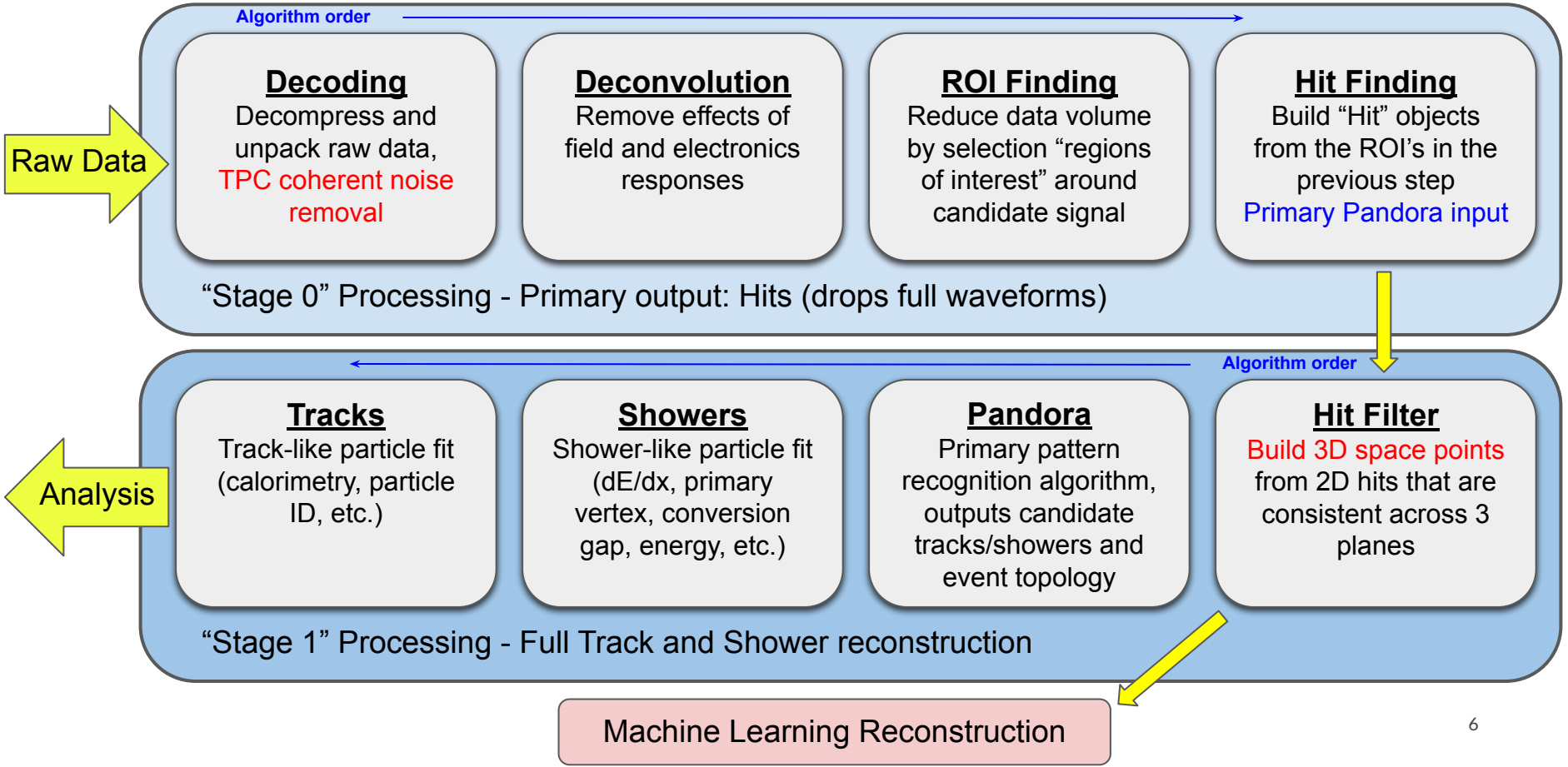
2. Event Reconstruction (“Stage 1”)
    a. Reconstruction of tracks and showers based on input hit information
    b. Candidate event topology, timing corrections, calorimetry, particle ID, etc.
    c. Everything needed to do final analysis output as LArSoft data objects

3. Summary Output
    a. Central Analysis Framework (CAF) format output (root trees)
    b. Assume all final CR rejection, candidate neutrino selection and high level analysis done here

Caveats For What Follows

- Focus in this presentation is on the TPC Signal Processing stage
  - Physics simulation - Geant4
    - Need to move from the legacy Geant4 in LArSoft
    - See Soon's presentation yesterday on g4
  - Detector simulation
    - Moving to using the WireCell "2D" TPC simulation
    - See Brett's presentation earlier today on Wire Cell
  - Higher level reconstruction
    - For Pandora path see Ryan's presentation earlier today
    - For ML path already using HPC
- At this point PMT and CRT signal processing and reconstruction times are in the noise level for both CPU time and memory usage
- There is no way I could be confused with a true multi–threading expert…

ICARUS Processing Chain - TPC Reconstruction

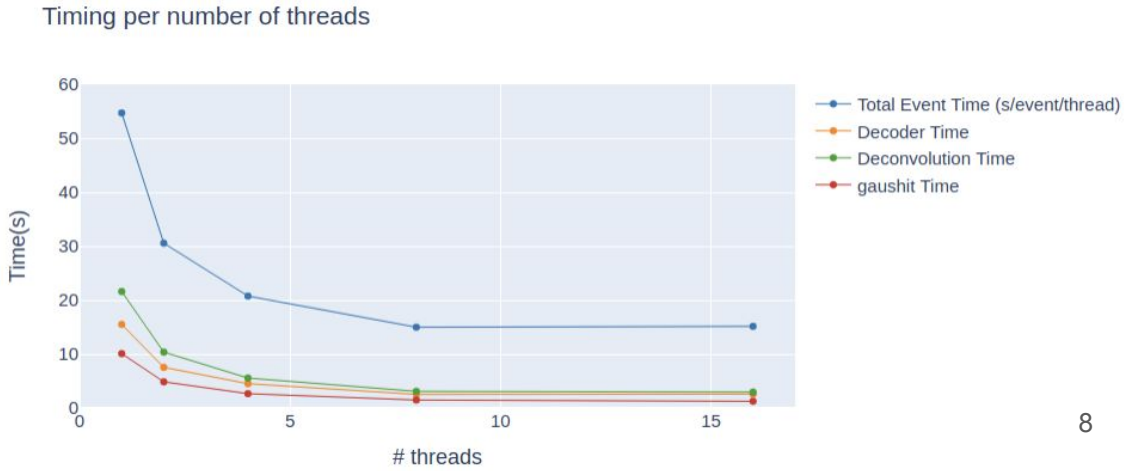


ICARUS Signal Processing General Considerations

- Data is presented to Stage 0 in units of TPCs
  - At this point in time each step in Stage 0 processes all four TPCs before moving to next step
- TPC Data handled differently from “decoding” to other steps
  - Each TPC presents 13824 channels of raw data to the decoder.
    - The decoder handles this data in hardware defined blocks of 512 channels
  - The deconvolution, ROI handling and hit finding handle each TPC’s data by channel
- Imagine two ways one might organize for multi-threading:
  - Can arrange modules to operate one TPC at a time, then have four schedules to handle flow through the signal processing modules for each TPC
  - Can arrange internal loops in each module to run multi-threaded
    - This allows us to increase past four threads if necessary?

Earlier Work On Multi-threading At Stage 0

- Initial work on multi-threading done ~2 years ago - BEFORE production running
- Approach to allow for parallel loops within the modules, not to parallelize the module execution
  - TBB loops in the decoder and the 1D deconvolution/ROI finding
  - Use the multi-threaded GausHit finder (see Giuseppe's talk)
- Results were encouraging!
- But… at the time, services were not thread safe, etc..
  - And early production running implied job time dominated by other factors
- Using a Dell Precision 7540 for testing
  - 16 cores
  - 64 GB RAM
- Containerized…
  - Docker container of SL7 running in Singularity accessing LArSoft code stack through cvmfs



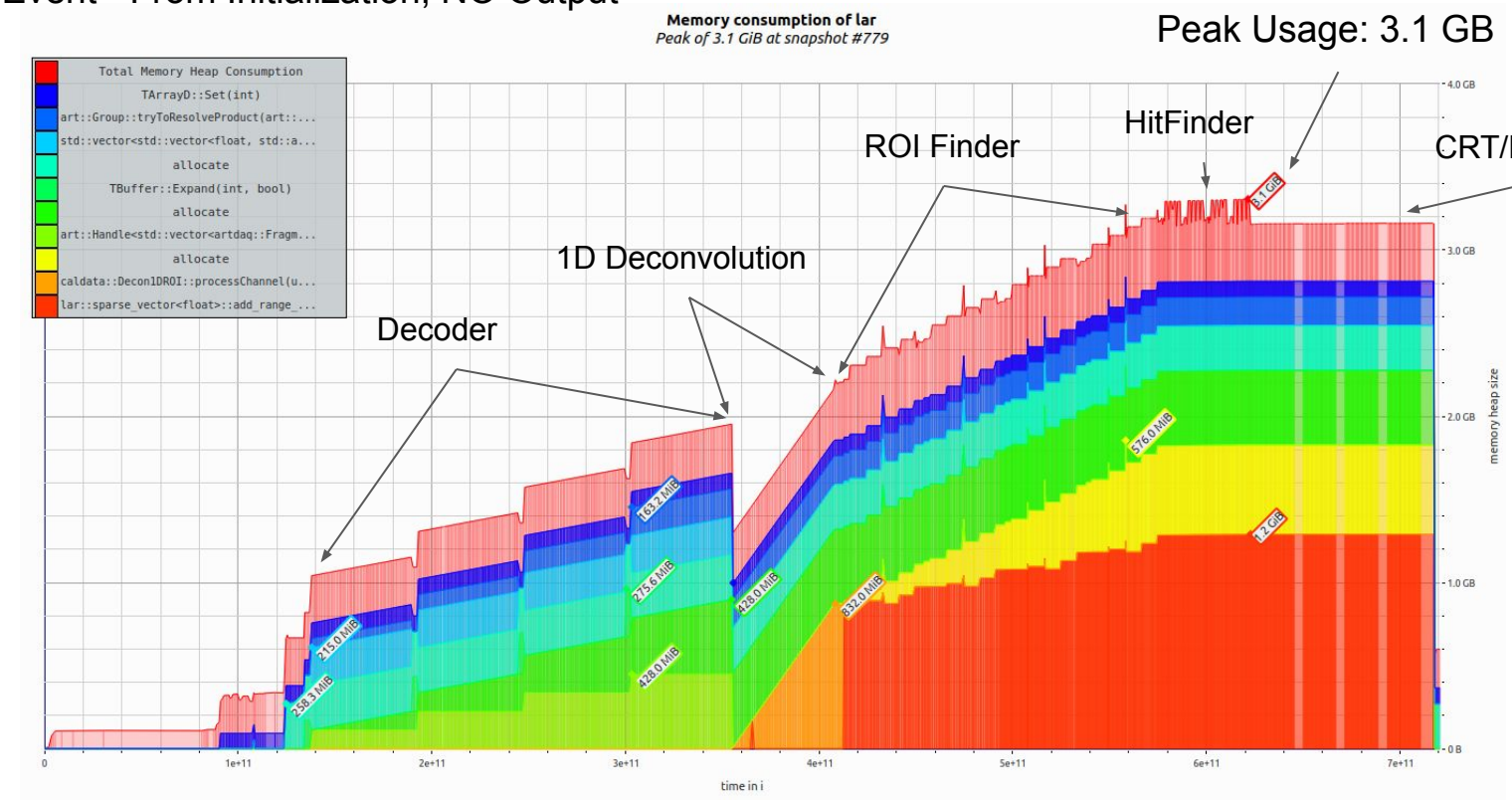
Does Running Multi-threaded Make Sense?

- Production running current done on the grid
  - Mantra: 1 core = 2 GB of memory and this defines 1 grid slot
  - Jobs submitted asking for grid slots

- Issues to understand:
  - How many grid slots do our jobs need?
    - Or, does our memory usage balance against using extra cores?
  - What is driving the total job time
    - Meaning: if the cpu time for running a job is not the major time driver for a job then are we really gaining anything?
  - Does our code have other thread safety issues we need to deal with?
    - It seems when you get failures do to thread issues it can be hard to understand exactly what the issue is

ICARUS TPC Signal Processing Memory Usage

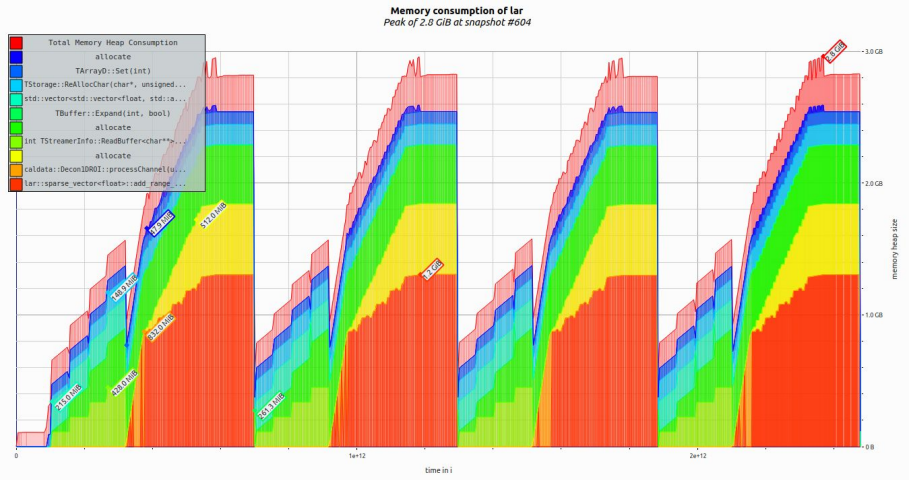
Single Event - From Initialization, NO Output



Source: valgrind massif (Thanks to Gianluca Petrillo)

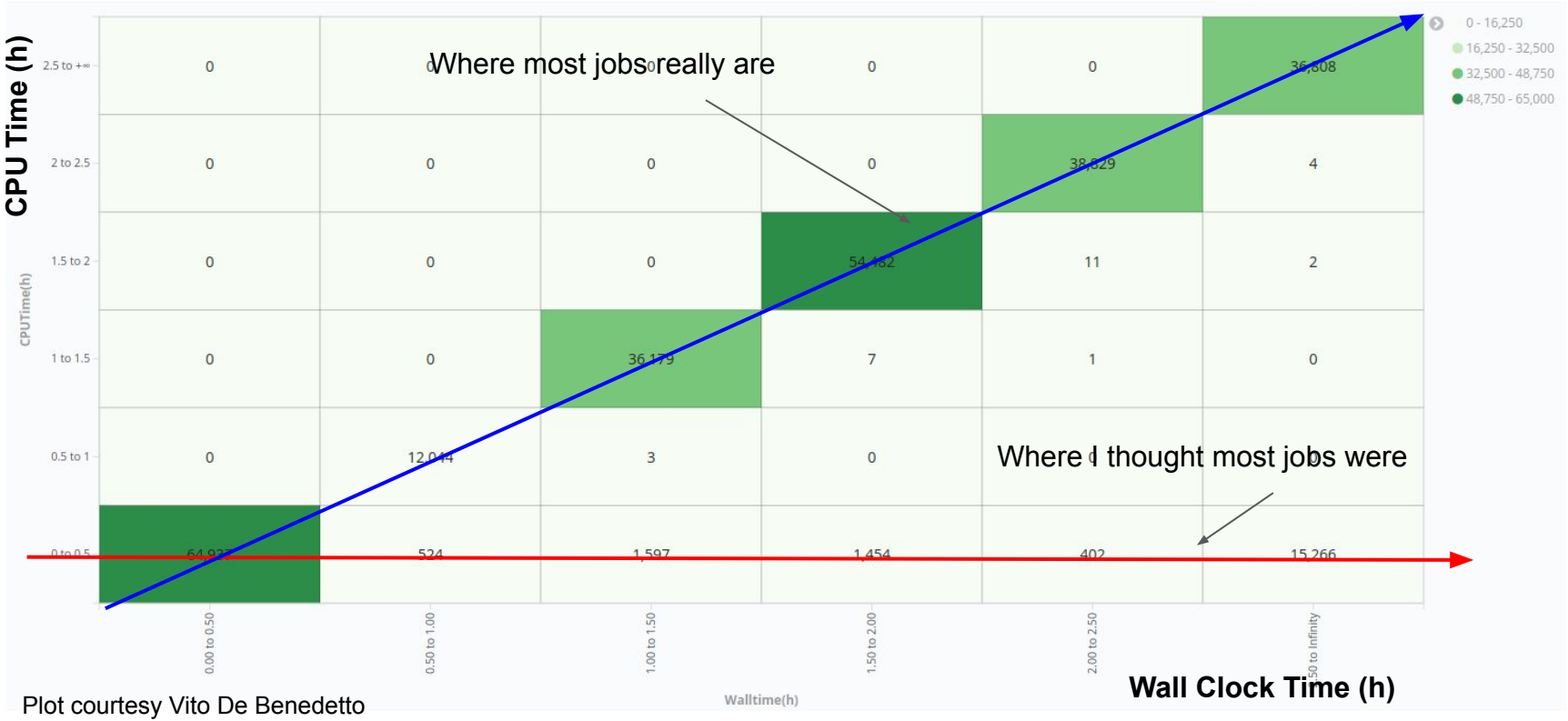
ICARUS TPC Signal Processing Memory Usage

- Generally, the decoding and deconvolution stages use the most memory
  - In both cases full waveform data is converted from short ints to floats
  - In the decoder it is converted back to short before placing in the data store.
- Memory usage is very regular from event to event
- Have not included the impact of root output to art root format files
- Peak memory usage reported by valgrind is 3.1 GB, jobs typically execute in grid slots under 4 GB



- Generally, because of total memory usage we are requesting 2 grid slots for stage 0 processing.
  - ”Grace” if we go over 4 GB, jobs automatically resubmitted with 8 GB

ICARUS Stage 0 Job Timing: CPU vs Wall Clock



Plot courtesy Vito De Benedetto

Generally, CPU time is dominant component of job processing time

Remnant Potential Thread Safety Issues?

- Database accessing services
  - See Kyle's presentation yesterday, should be good to go now?

- The Decoder, 1D deconvolution and ROI finding steps make use of code in the icarus_signal_processing package
  - Is meant to be dual use - LArSoft package but also independent of LArSoft
    - No LArSoft package dependencies
  - This package makes heavy use of the standard template library
    - In particular, stl vectors
  - Are stl allocations thread safe in our usage?
    - Yes: We're good to go
    - No: We need to do some work in this package to insure thread safety

Path Forward

- I learned a lot in getting ready for this presentation!

- ICARUS production data processing at the stage 0 level would clearly benefit from moving to a multi-threaded processing model
  - Current jobs are leaving idle 3 CPU's in standard job submissions
  - For most jobs processing time dominated by CPU time
  - Potential to 2, possibly 3 times faster?

- PR's exist to enable this processing
  - Still would like to make sure we do not have an issue with stl usage in our bottom level signal processing package
  -
- ICARUS will prioritize trying to make this happen!