HEP Event Reconstruction
with Cutting Edge Computing Architectures

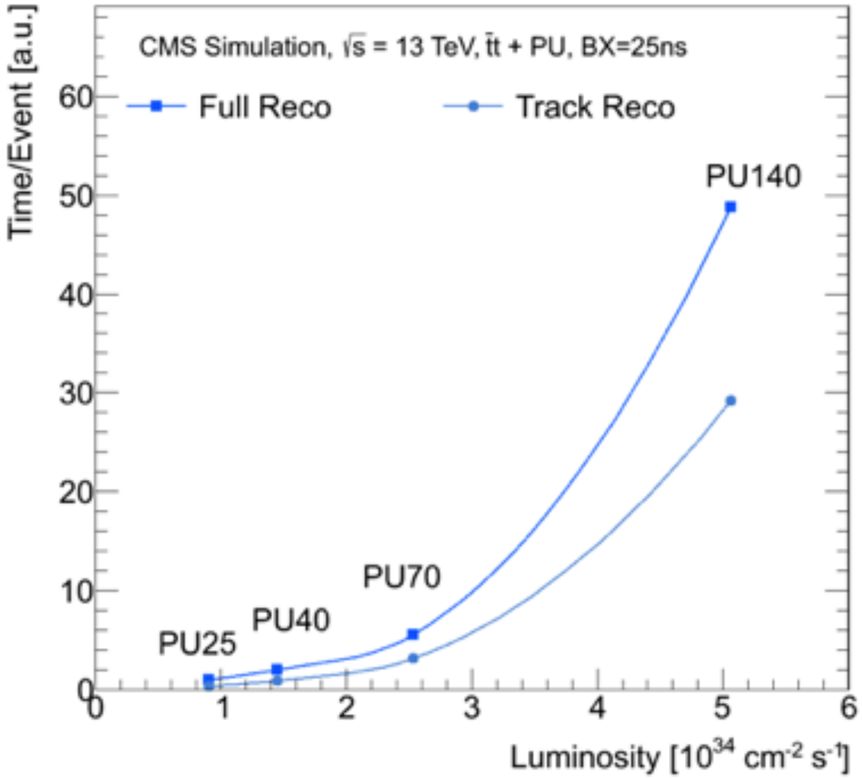
Sophie Berkman, Giuseppe Cerati, Allison Hall (FNAL)
Brian Gravelle, Boyana Norris (UOregon)

SciDAC-4 PI meeting, July 17, 2019

Introduction

- Science goals: Higgs as a tool for discovery, physics of ν mass
  - HL-LHC, LArTPC program (SBN, DUNE)
- Reconstruction: processing of detector signals to extract information about the particles that produced them
  - Challenge for complicated detectors and busy data
- Future experiments even more challenging:
  - Larger sizes or more granular: more detector channels
  - Higher beam intensities: more data to process
- Reconstruction CPU time does not scale well

- **Need large speedups** in reconstruction to reach design detector sensitivity and enable discoveries!



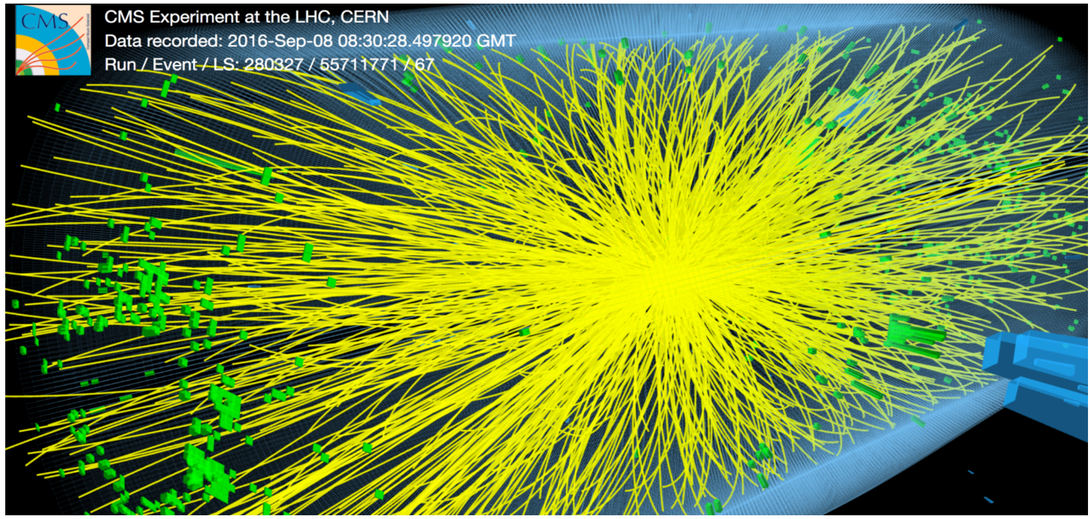
Project goals

- Accelerate HEP event reconstruction using modern parallel architectures.

- Focus on two areas:
  - Novel parallel algorithm for charged particle **tracking in CMS**
    - Cornell/FNAL/UCSD/UOregon/Princeton collaboration
    - Non-SciDAC funding from NSF IRIS-HEP and from USCMS
  - Pioneer similar techniques for **reconstruction in LArTPC detectors**

- Goals of the project are the following:
  1. Identify key algorithms for the physics outcomes of each experiment, and that also are dominant contributors to their reconstruction workflows
  2. Characterize and re-design the algorithms to make efficient usage of parallelism, both at data- and instruction-level
  3. Deploy the new code in the experiments' framework
  4. Explore execution on different architectures and platforms

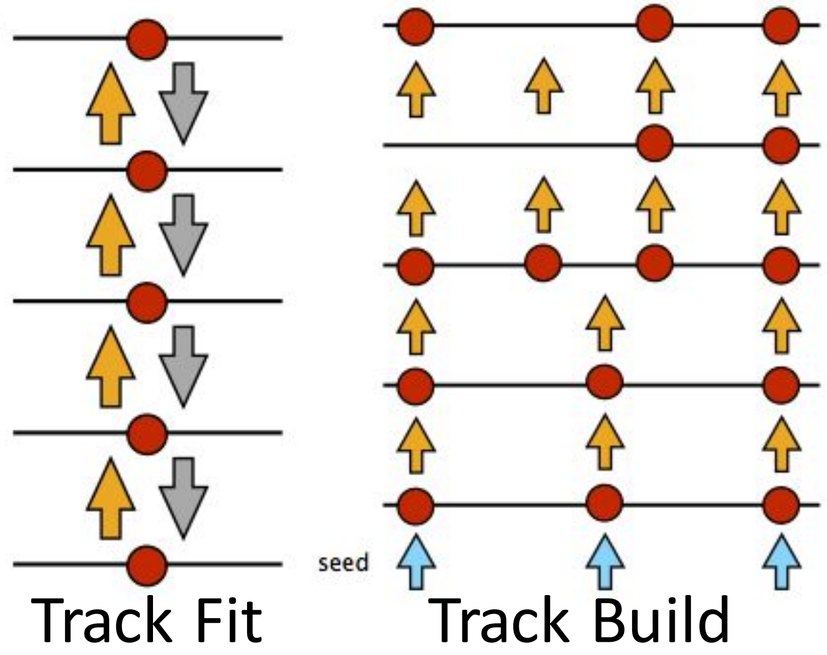
CMS tracking prototype



- Reconstruct trajectories of charged particles through thin silicon layers
  - image: PileUp (PU) 50 collisions; HL-LHC: PU~200

- Building: **combinatorial search** for compatible hits along the track based on Kalman filter
  - drives increase in processing time, scales poorly with N_{hits}
  - Inspired by CMS version but with large differences, e.g. avoid resolving fine grained geometry structures

- Computational challenges: **branching points, low arithmetic intensity**
  - quick processing of many small objects



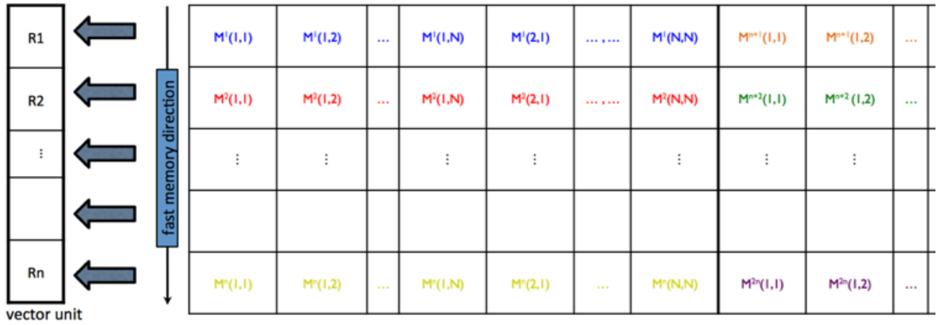
Fermilab

Standalone Performance

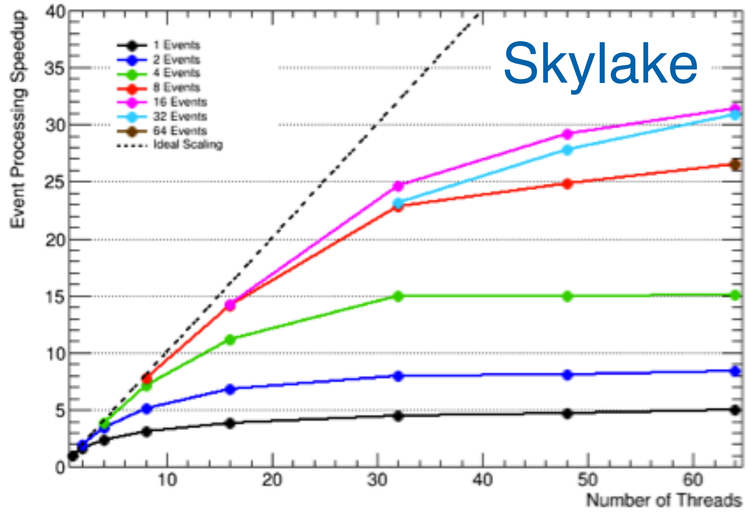
- Vectorized using **Matriplex**: SIMD processing of multiple candidates, achieve **2-3x speedup**

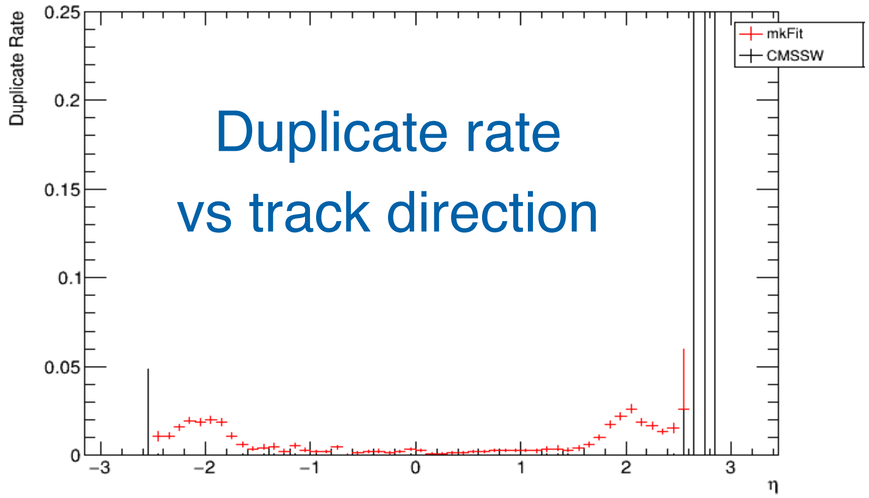
- **Thread-level parallelization** at multiple stages: events, detector regions, bunches of seeds
  - using TBB, up to 30x speedup on Skylake

- Physics performance continues to improve
  - Efficiency same or better than CMSSW for long tracks
  - Huge **reduction of duplicate tracks** thanks to dedicated step: now <2% across all detector regions
    - “serial” CMSSW duplicate removal cannot be applied



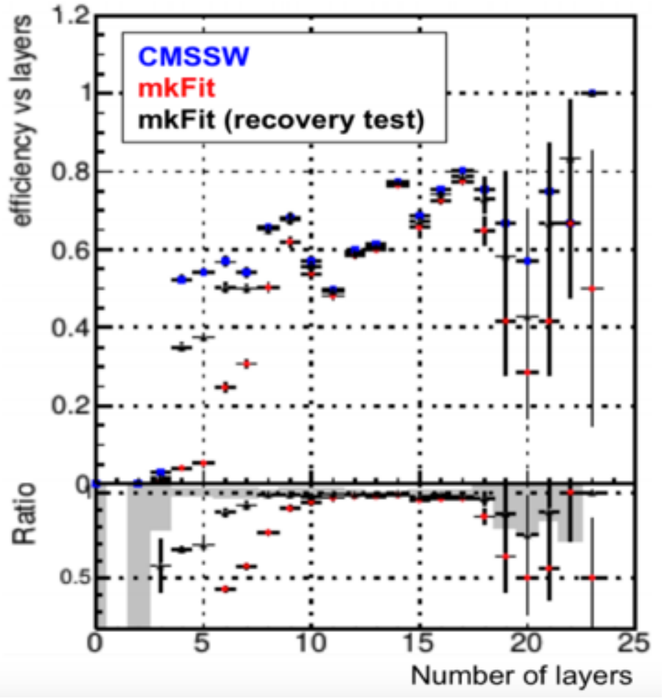
Matrix Size *NxN*, vector unit size *n*

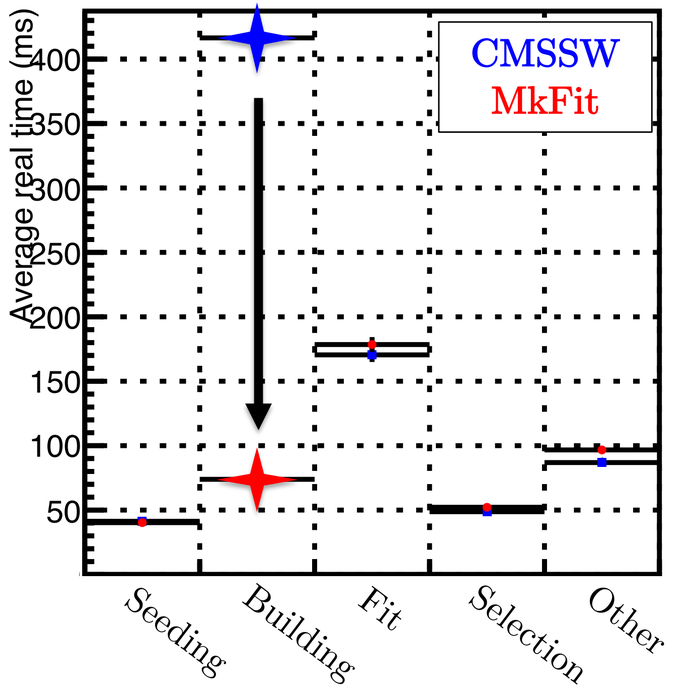




CMSSW Integration

- Successfully integrated in CMSSW as **external library**
  - mkFit compiled with icc and AVX512, CMSSW gcc and SSE
  - CMSSW validation tools revealed **efficiency loss** for short tracks
    - causes identified, work underway to recover efficiency
- Timing improvements: **>6x faster** single threaded
  - includes data format conversions (would be 8x without)
- Integration being finalized:
  - release code for central builds, discuss data formats compatibility
- Next step: **integration in High-Level Trigger (HLT)**
  - additional challenges: on-demand input, different constraints
  - Run3 HLT hardware: Skylake, possibly with GPU acceleration
  - Paper in preparation!



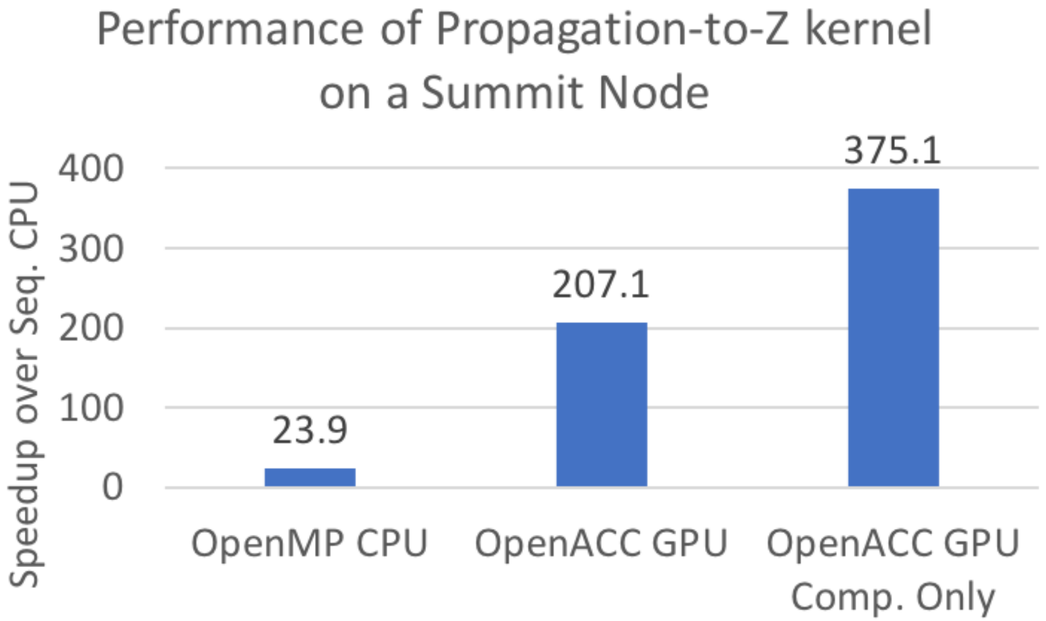


Explore Portable Implementations

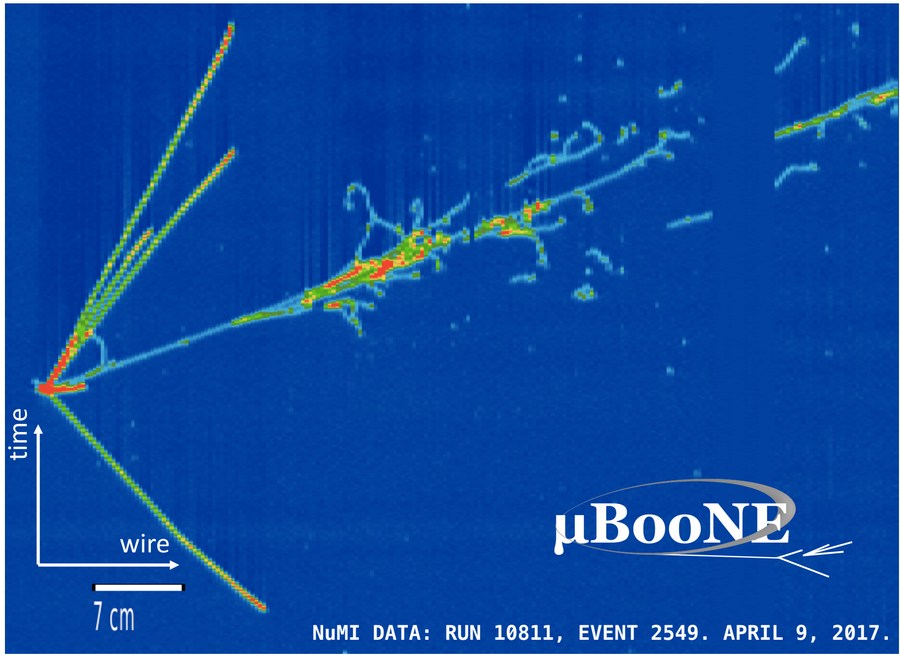
- Exploration of GPU-compatible, portable implementations is becoming a priority
  - maintainable, minimal diffs between CPU and GPU code

- Started collaboration with **RAPIDS/ORNL** to explore usage of portable **compiler directives**
  - test single function (out of ~100) from full code, get 375x speedup on GPU (excluding data transfer)
  - **challenges ahead**: data transfer, CMSSW interface

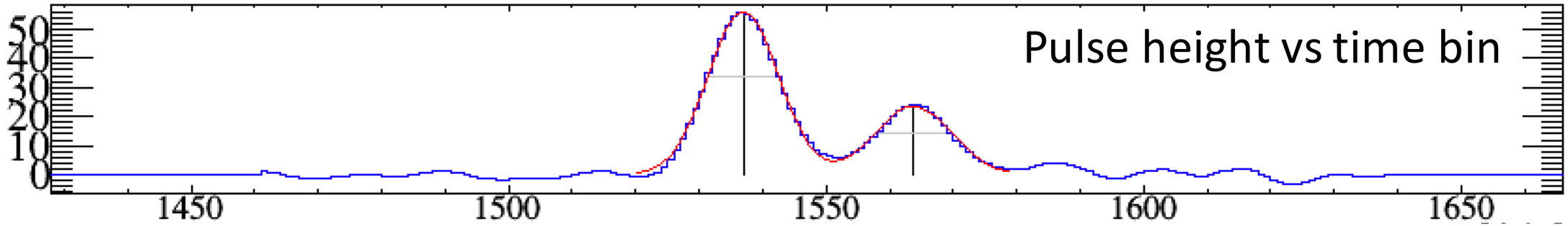
- Other tests towards portable implementations:
  - write algorithm in terms of array programming
  - performance of low level operations in CUDA



Hit Finding in LArTPC

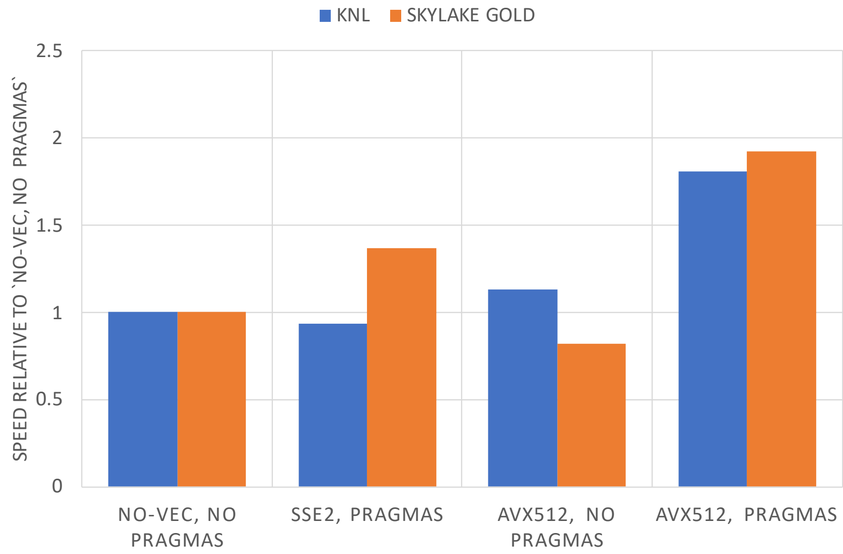


- Reconstruction in LArTPC neutrino experiments is **challenging** due to many possible neutrino topologies, noise, contamination of cosmic rays
  - Takes O(minutes)/event in MicroBooNE
  - ICARUS >5x bigger, DUNE FD ~200x bigger
  - LArTPC detectors are modular in nature ➔ parallelism!
- **Feasibility study**: hit finding
  - MicroBooNE TPC: ~8k wires readout at 2 MHz, wire signals are Gaussian pulses
  - Hit finding is the process of identifying pulses and determining **peak position and width**

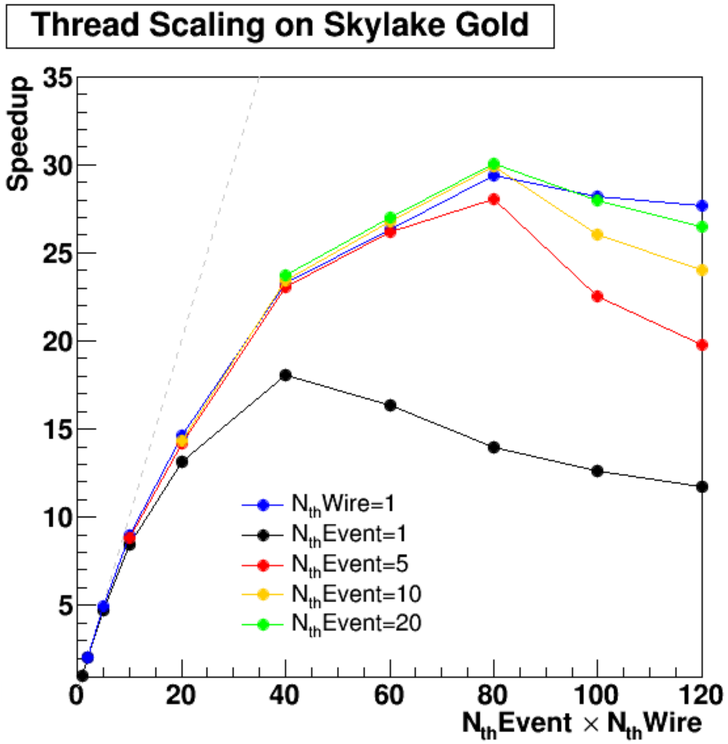


Standalone Implementation

- Replicated LArSoft hit finder as **standalone code** for easier testing and optimization
  - Replaced Gaussian fit based on Minuit+ROOT with a local implementation of Levenberg-Marquardt minimization

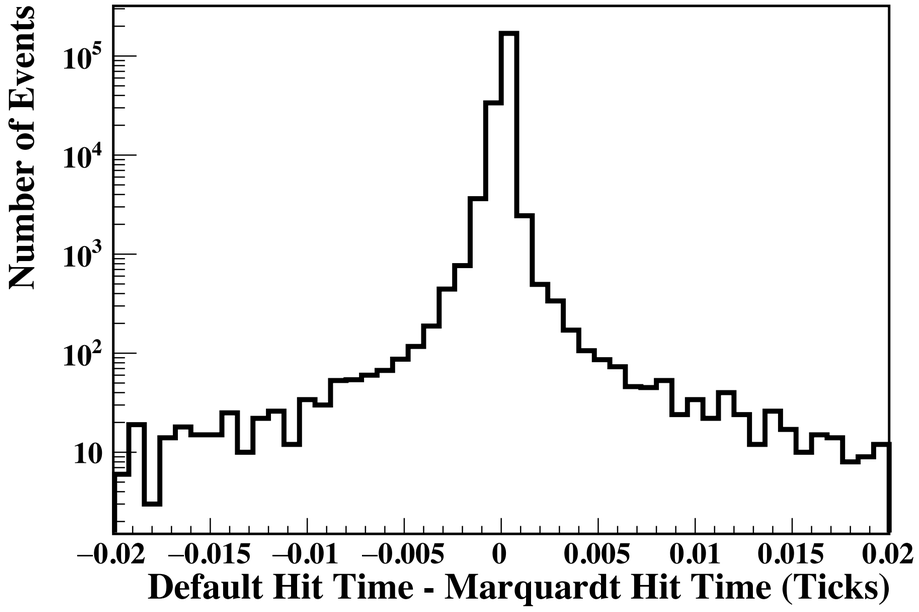


- **Vectorized loops** within the minimization algorithm, typically across data bins
  - Close to **2x speedups**, both on Skylake Gold and KNL
  - Comparison with Intel Math Kernel Library, our fitter is faster
- Nested OpenMP **parallelization** over events and wires
  - Near-ideal scaling at low thread counts
  - Speedup up to 30x (95x) for 80 (240) threads on Skylake (KNL)



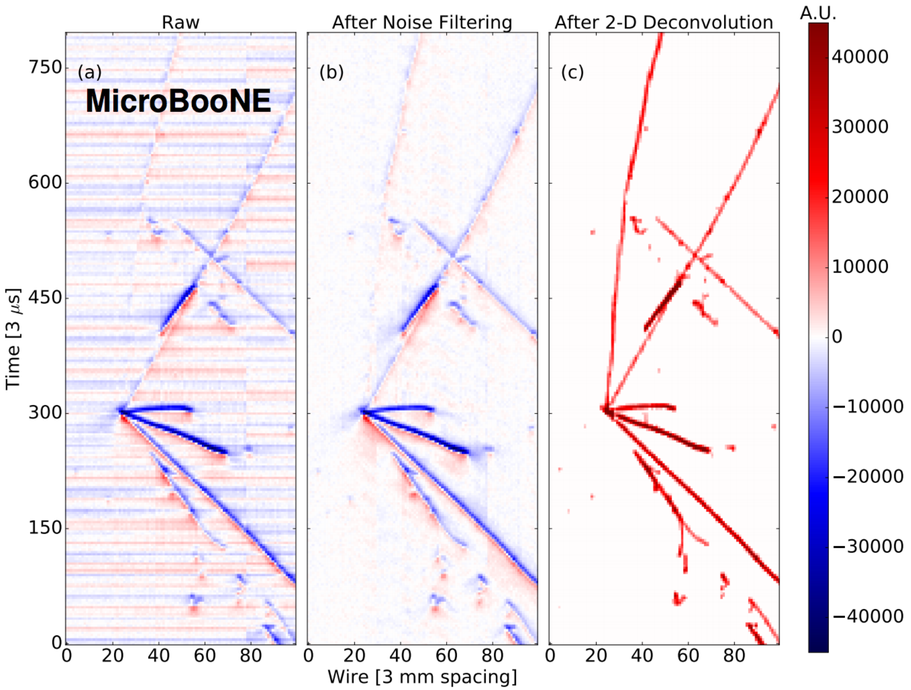
LArSoft Integration and Next Steps

- Minimization algorithm used as a **plugin in LArSoft**
- Hit finder on MicroBooNE neutrino+cosmic events runs **12x faster** than the default version (both single thread)
- Differences in algorithm output are negligible
- Preliminary results for ICARUS:
  - Hit finder takes ~40% of reconstruction time, get 5-7x speedup



- First vectorized and multi-threaded algorithm for LArTPC
  - push to set higher standards, LArSoft still compiled with gcc/SSE
  - Collaborating with LArSoft team to modernize software

- Next will focus on **signal processing**
  - Most time consuming step in MicroBooNE
  - Scales with number of wires, relevant for DUNE!



Thank you!

- Working on key algorithms for CMS and LArTPC experiments
- Obtain large speedups with modern computing architectures
- Algorithms are ported back to the experiments' frameworks
- Exploring portable solutions (collaboration with RAPIDS)
- Next steps: CMS HLT, more algorithms for LArTPC

http://computing.fnal.gov/hepreco-scidac4/