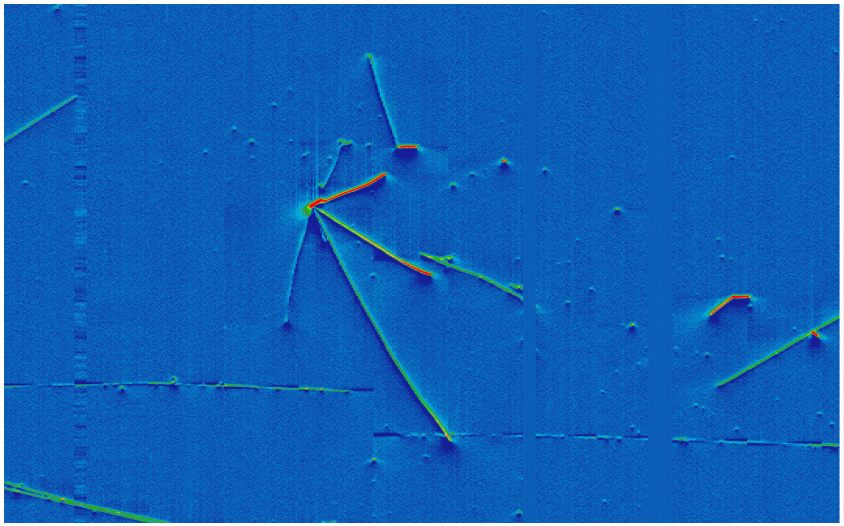
Vectorizing and Parallelizing the Gaus-Hit Finder

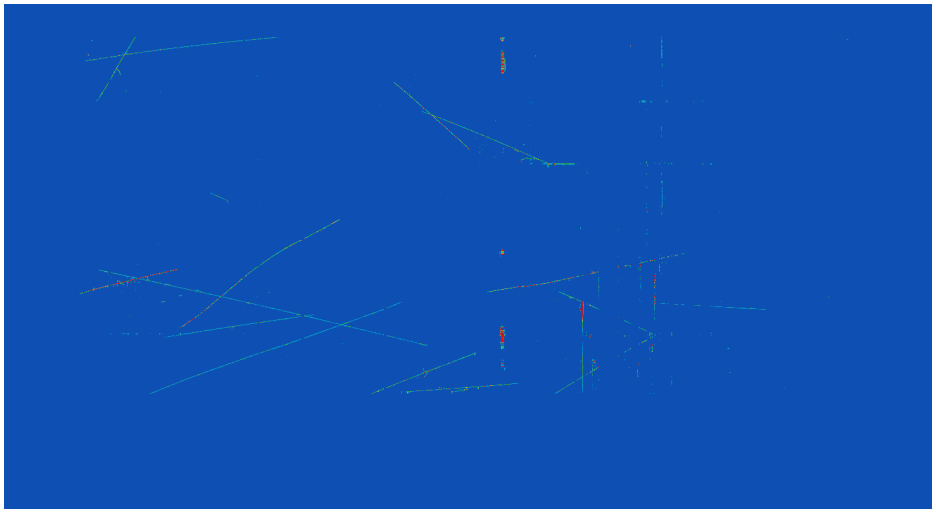


Sophie Berkman
for the SciDAC HEP Reco Group
(G. Cerati, B. Gravelle, A. Hall, B. Norris, M. Wang)

LArSoft Meeting
June 18, 2019

SciDaC Project: HEP Event Reconstruction

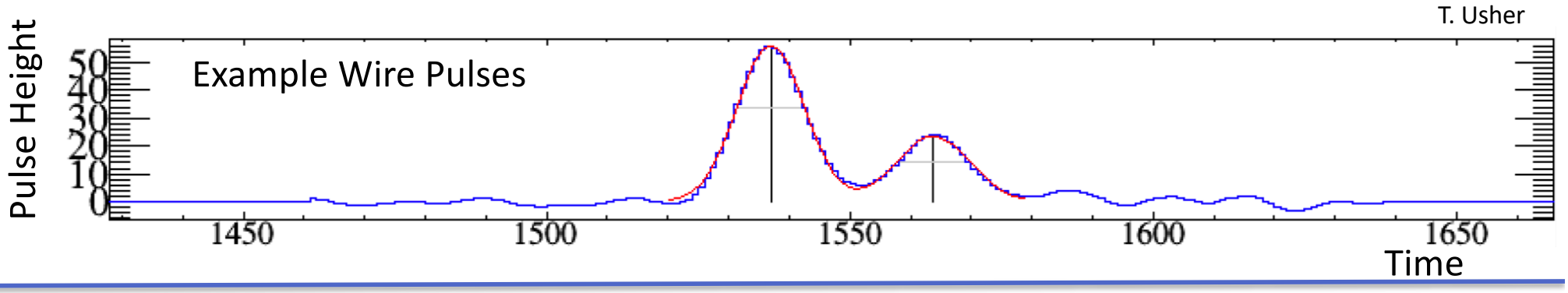
- Study improvements to HEP event reconstruction using vectorization and modern computing architectures
- Liquid Argon:
  - Took O(100 s) to process a μBooNE event (8,256 wires)
    - MCC8 rerconstruction
  - Improvements necessary for a larger scale experiment like DUNE (384,000 wires/ 10 kTon cryostat)
  - Focus on vectorizing and parallelizing low level signal processing and event reconstruction
- CMS: vectorize and parallelize tracking code



Fermilab UNIVERSITY OF OREGON

Feasibility study: GausHitFinder

- Feasibility study: GausHitFinder
  - Charged particles produce pulses on wires. Identify and extract parameters associated with pulses (position, amplitude, width).
  - Wires are independent; can be processed independently
  - ~15% of µBooNE work flow time (in MCC8)
- Vectorization and parallelization developments were done within a stand-alone version of the GausHitFinder developed by M. Wang, G. Cerati, B. Norris
  - Implements the Levenberg-Marquardt algorithm to do the fitting
  - ROOT/ Minuit not suitable for parallelization - global memory management
  - Stand-alone code is 8 times faster than the ROOT version even before vectorization and parallelization.
  - Will discuss results on stand-alone code, and then LArSoft integration
    - All results are on single muon events simulated in µBooNE
    - Stand alone code compiled with icc



Vectorization of Stand-Alone GausHitFinder

- Vectorization challenges:
  - Minimization difficult because fits converge in different numbers of iterations
  - Cannot fit multiple hits at the same time
  - Vectorize the most time consuming loop, but this is not all of the code
- Vectorization Strategies:
  - Compiler vectorization: use avx512
  - Explicit vectorization on the most time consuming loops:
  - Loops determined by profiling the code
  - #pragma omp simd, #pragma ivdep
- **Speed increases**
  - **Explicit vectorization:** ~65% faster on KNL, ~50% faster on Skylake
  - **Compiler and explicit vectorization:** 2 times faster on KNL than with no vectorization

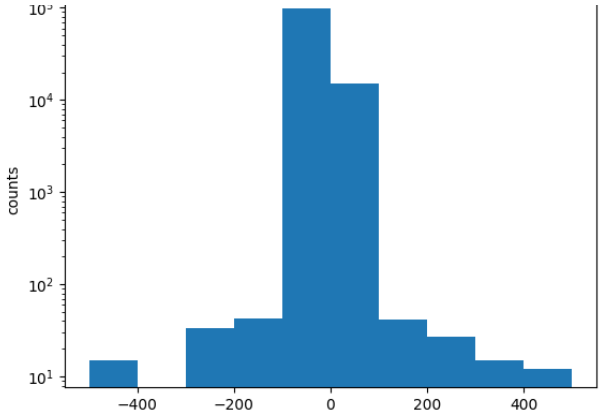
Vectorization Compiler Option	Speed-Up relative to no vectorization
no-vec, no pragmas	1
sse, pragmas	1.2
avx512, no pragmas	1.3
avx512, pragmas	2.0

Vectorization Using Intel MKL

- Intel Math Kernel Libraries (MKL)
  - State of the art vectorized math library, so it is an important point of comparison.
- Do the fitting in a way that is well vectorized
- Implemented MKL as another fitting option within stand-alone framework
- **Results:**
  - Physics performance consistent with Marquardt fitter
  - ~5 times slower than Marquardt fitter.

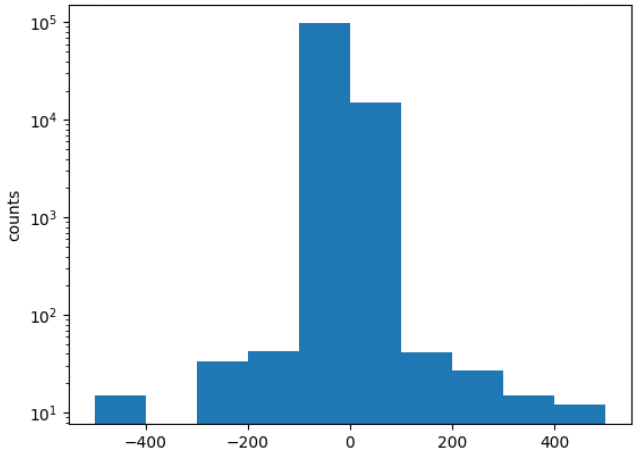
Work by B. Gravelle (U. Oregon)

Marquardt Fit Performance



Simulated-Reconstructed Hit Time

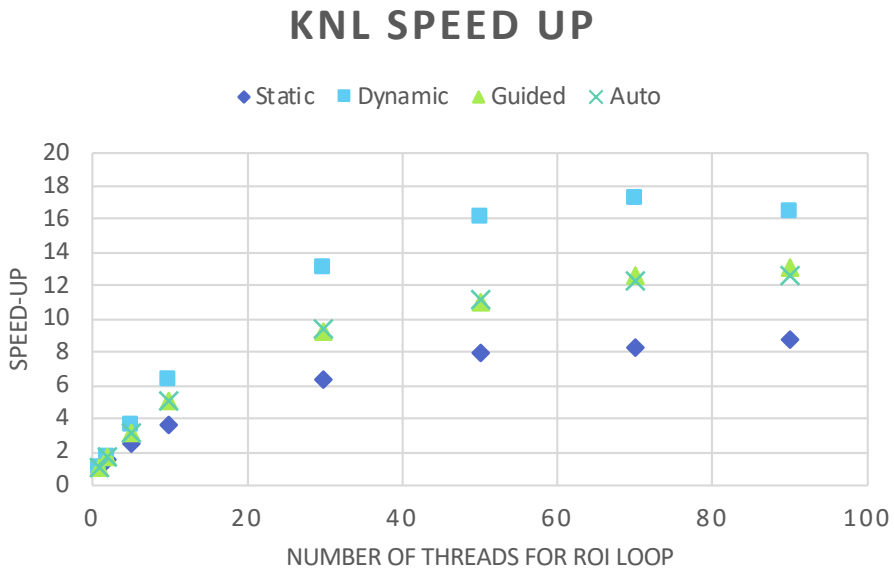
MKL Fit Performance

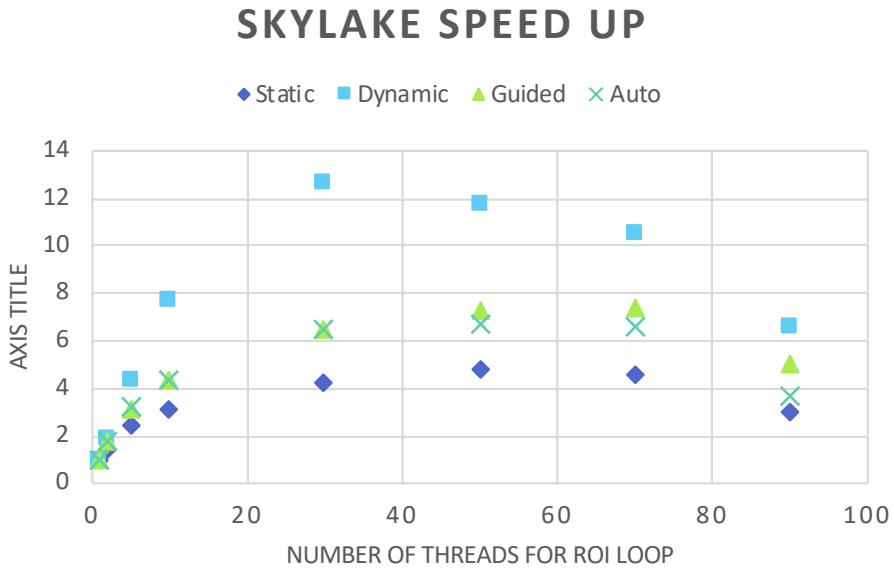


Simulated-Reconstructed Hit Time

Parallelization of Stand-Alone GausHitFinder

- Using OpenMP parallel for loop over regions of interest (ROI) on the wires
  - Fastest with “dynamic” thread scheduling
- Parallelization challenges:
  - Algorithm has a relatively small amount of work. Single muon events have less less work to do than the average neutrino event.
  - Thread overhead may limit speed up
- **Speed increases with parallelization:**
  - KNL: 17 times faster
  - Skylake: 12 times faster
- The speed improvements from parallelization are not yet included in LArSoft



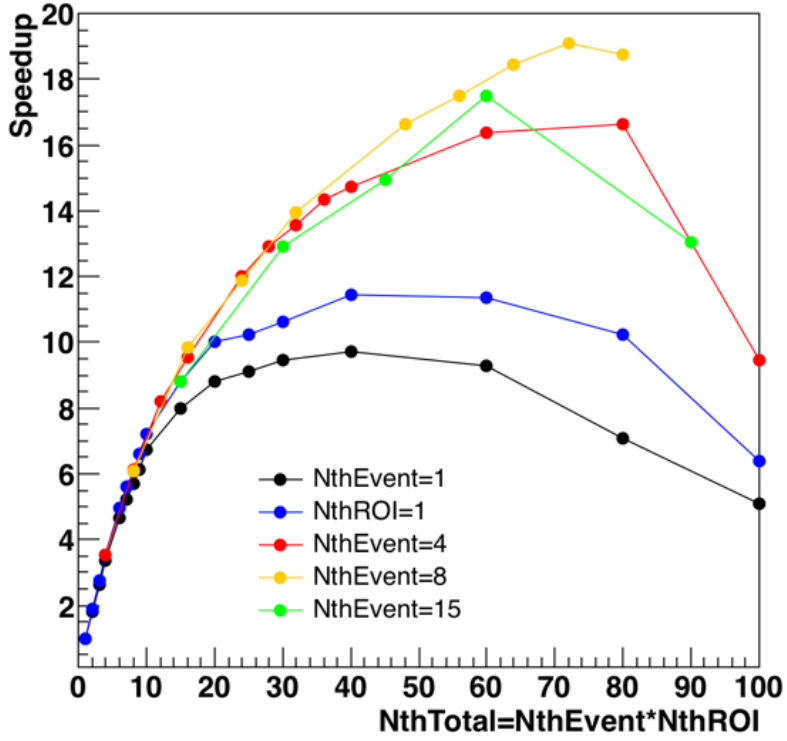


Parallelizing over Events

Work by G. Cerati

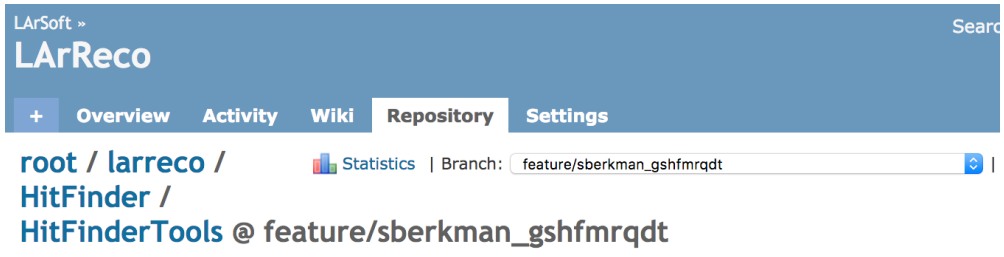
- Additional speed increases by parallelizing over events as well as regions of interest
  - Additional OMP parallel for loop with dynamic scheduling
- Speed increases on skylake with additional parallelization:
  - ~9 times faster when parallelizing only over region of interest (Event threads fixed to 1)
  - ~11 times faster when parallelizing only over events (ROI threads fixed to 1)
  - ~20 times faster when parallelizing over both

Speed Up Parallelizing over Events and ROIs



LArSoft Integration

- Integrated a version of the stand-alone code with the Marquardt fitter into LArSoft
  - **Branch of larreco:** *feature/sberkman_gshfmrqdt*
- Marquardt fitting is implemented as a class called MarqFitAlg
  - Does not depend on any external libraries
- New tool "PeakFitterMrqdt_tool.cc" does the fit using the same Marquardt fitter as implemented in the stand alone code.
- Can call this new tool instead of the default "PeakFitterGaussian_tool.cc" in the GausHitFinder_module.cc
  - Does the fitting in "findPeakParameters" function
- None of the current functionality was changed in this branch, just has the option to use the new fitter
- Mike is also using this Levenberg-Marquardt fitter in LarSoft.

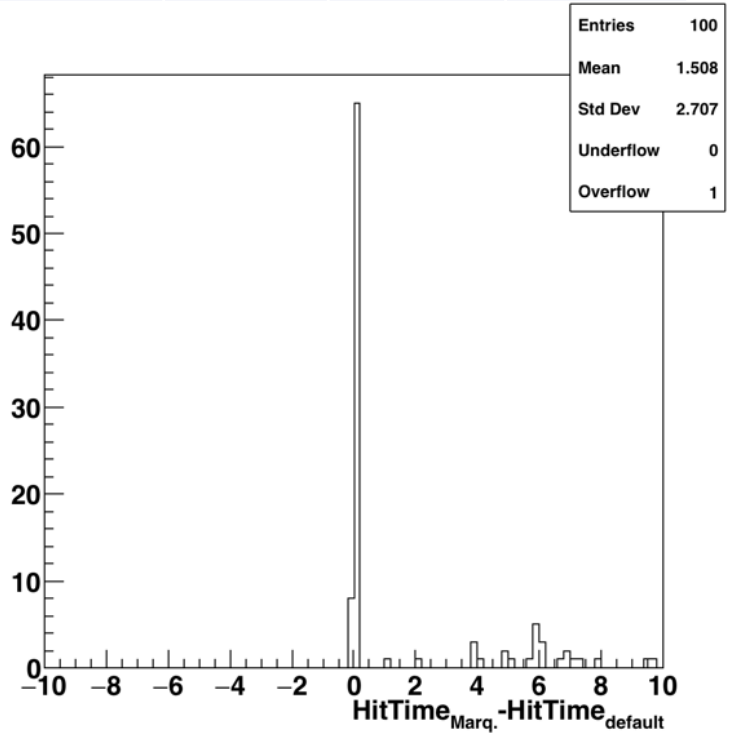


LArSoft Validation

- Initial validation done on uboonebuild01.fnal.gov, with 20 single muon events
- **Results:**
  - Hit finder is **9.6 times** faster on average than the current LArSoft version.
  - Physics results are comparable.
    - Will look into ~25% of cases where results are different.
- Does not yet include all of the vectorization and parallelization improvements.
  - No parallelization
  - Uses sse instead of avx512

Validation by G. Cerati

Fitter	Avg Time (s)	Min Time (s)	Max Time (s)
ROOT	0.674	0.146	1.78
Marquardt	0.070	0.034	0.151
Speed Increase	9.6	4.3	11.8



Conclusions & Future Work

- GausHitFinder has been vectorized and parallelized:
  - Up to 20 times faster with parallelization
  - Up to 2 times faster with vectorization
- Levenberg-Marquardt algorithm has been implemented to do the fitting in the GausHitFinder algorithm instead of ROOT
  - Fitter implementation performs well when compared to MKL
- New version of the GausHitFinder integrated into LArSoft:
  - 9.6 times faster than the current implementation
  - Results are reasonable, some additional validation may be needed to understand any differences between new and current version.
  - Not yet taking advantage of all of the potential vectorization and parallelization improvements, which are further independent speed-ups.
- Future directions:
  - GPUs: work has started on the CMS side of the SciDAC project and plan to test similar techniques with liquid argon code.
  - Plan to start working with other signal processing algorithms next.