

Parallelization for HEP Event Reconstruction

ICHEP 2020, Computing and Data Handling session

July 28, 2020

S.Lantz, K.McDermott, M.Reid, D.Riley, P.Wittich (Cornell)

email: cerati AT fnal.gov



S. Berkman, G.Cerati, A.Reinsvold Hall, M.Kortelainen, M.Wang (Fermilab)

P.Elmer, B.Wang (Princeton)

L.Giannini, S.Krutelyov, M.Masciovecchio, M.Tadel, F.Würthwein, A.Yagil (UCSD)

B.Gravelle, B.Norris (UOregon)

mkFit and SciDAC4-HepReco projects





Recent Changes in the Computing Landscape

- Due to power limits, the CPU frequency is no longer growing exponentially:
 - nothing for free anymore
 - since 2005, most of the gains in single-thread **performance** come from vector operations
 - growth in number of transistor leads to an increase in **logical core** count: multithreading
- Changes can become opportunities:
 - exploit both levels of parallelization to avoid sacrificing physics performance!
 - new technologies dominating the market, e.g. GPU
 - exa-scale machines are being built at HPC centers



and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten New plot and data collected for 2010-2015 by K. Rupp



Preparing for the Next Generation of Experiments

- Next generation of HEP experiments will feature unprecedented detector
- These lead to an increase in data volume and corresponding increase in
- This talk focuses on computing R&D for reconstruction in CMS and at **HL-LHC** and **DUNE**



granularity or size, and will be exposed to beams of unprecedented intensity processing time, so exploiting new computing technologies is a vital need LArTPC experiments, with the long term goal of improving the physics reach of





LHC Reconstruction Challenges

- Increasing pile-up (PU) drives exponential increase in event reconstruction processing time
- **Tracking** is the dominant contributor to the reconstruction time, and it's vital for the physics output of HEP experiments
 - reducing the tracking phase space implies significantly worse physics sensitivity
- Kalman filter-based tracking is the "standard" tracking method in HEP
 - demonstrated high efficiency physics performance, robust handling of material effects
 - track building is a combinatorial algorithm traditionally implemented in serial fashion, challenging to parallelize



Event display, CMS 2018 high PU run (PU 136)



Predicted track state **Detector** measurement Updated track state





Parallelizing Tracking: the mkFit Approach

- **mkFit** mission: parallelize Kalman filter track building
- Matriplex library designed for SIMD processing of track candidates
 - bunches of small matrices operating in lock-step, auto-generated vectorized code is aware of matrix sparsity
- mkFit is **multithreaded** at multiple levels with **TBB** tasks: events, detector regions, bunches of seeds • Reduce memory footprint with lightweight detector description (geometry, material, magnetic field) • Minimize memory operations (number and size) within combinatorial branching
- - bookkeeping of explored candidates, clone only best ranking ones at each layer (with per seed cap)

R1		M ¹ (I,I)	M ¹ (1,2)	 M ¹ (1,N)	M ¹ (2,1)	,	M ¹ (N,N)	M ⁿ⁺¹ (1,1)	M ⁿ⁺¹ (1,2)	 M
R2	ection	M²(1,1)	M²(1,2)	 M²(1,N)	M²(2,1)	,	M²(N,N)	M ⁿ⁺² (1,1)	M ⁿ⁺² (1,2)	 M
:	imory air	÷	:	:	:		:	:	:	
Rn	,	M ⁿ (1,1)	M ^₀ (1,2)	 M ⁿ (1,N)	Mº(2,1)		M ^₀ (N,N)	M ²ⁿ (I,I)	M ²ⁿ (1,2)	 м

Matriplex memory layout





mkFit Physics and Computing Results

- Standalone application achieves speedups of up to 3x from vectorization and up to 35x from multithreading across multiple collision events
 - vectorization speedup computed on main track building function only, evaluated on Intel Skylake Gold processor
- mkFit is integrated in CMS framework (CMSSW), with a single threaded application it is 6x faster
 - mkFit compiled using icc and AVX-512 extensions
 - track building with mkFit is faster than CMSSW track fitting!
 - integration of mkFit in CMS workflows is currently under investigation



• mkFit track building achieves comparable physics performance as standard CMS tracking (first iteration)

CMS tracking, first iteration



MkFit



mkFit paper — hot off the press!

 More details about the mkFit algorithm and results in our paper - arXiv:2006.00071, recently accepted for publication in JINST

Speeding up Particle Track Reconstruction using a Parallel Kalman Filter Algorithm

Steven Lantz,^a Kevin McDermott,^a Michael Reid,^a Daniel Riley,^a Peter Wittich,^a Sophie Berkman,^b Giuseppe Cerati,^b Matti Kortelainen,^b Allison Reinsvold Hall,^b Peter Elmer,^c Bei Wang,^c Leonardo Giannini,^d Vyacheslav Krutelyov,^d Mario Masciovecchio,^d Matevž Tadel,^d Frank Würthwein,^d Avraham Yagil,^d Brian Gravelle,^e Boyana Norris.^e

^aCornell University, Ithaca, NY, USA 14853 ^b Fermi National Accelerator Laboratory, Batavia, IL, USA 60510 ^cPrinceton University, Princeton, NJ, USA 08544 ^dUC San Diego, La Jolla, CA, USA 92093 e University of Oregon, Eugene, OR, USA 97403

E-mail: mic-trk-rd@cern.ch

ABSTRACT: One of the most computationally challenging problems expected for the High-Luminosity Large Hadron Collider (HL-LHC) is determining the trajectory of charged particles during event reconstruction. Algorithms used at the LHC today rely on Kalman filtering, which builds physical trajectories incrementally while incorporating material effects and error estimation. Recognizing the need for faster computational throughput, we have adapted Kalman-filter-based methods for highly parallel, many-core SIMD architectures that are now prevalent in high-performance hardware. In this paper, we discuss the design and performance of the improved tracking algorithm, referred to as MKFIT. A key piece of the algorithm is the MATRIPLEX library, containing dedicated code to optimally vectorize operations on small matrices. The physics performance of the мкFit algorithm is comparable to the nominal CMS tracking algorithm when reconstructing tracks from simulated proton-proton collisions within the CMS detector. We study the scaling of the algorithm as a function of the parallel resources utilized and find large speedups both from vectorization and multi-threading. MKFIT achieves a speedup of a factor of 6 compared to the nominal algorithm when run in a single-threaded application within the CMS software framework.

KEYWORDS: Data processing methods; Pattern recognition, cluster finding, calibration and fitting methods; Performance of High Energy Physics Detectors.





Next Steps: Strip Unpacking and Clustering on GPU

- Silicon Strip Hits are the main input to the mkFit algorithm
 - producing them can be a limiting factor to using mkFit in CMS trigger
- GPUs can help to speed up the processing of Si-Strip data
 - Unpacking of raw data from CMS Silicon Strip with SoA format
 - Parallelized implementation of "three threshold" clustering algorithm
 - Initial standalone version: multi-event processing using **OpenMP** with nested parallelism in CPU, and **CUDA** streams in GPU
- Recently developed a CMSSW producer, with CUDA application as "ExternalWorker"
 - time split roughly evenly between data transfers and computations
 - workload is not enough to saturate a GPU, multi-processing is being studied to achieve better utilization
 - achieve an aggregate throughput ~1.15 KHz, projected to be 50% faster than CPU-only equivalent process
- Next steps: evaluate hit positions on the GPU, produce hits in mkFit data format

- e mkFit algorithm mkFit in CMS trigger of Si-Strip data ith SoA format clustering algorithm g using **OpenMP** with n GPU
- and computations
 -processing is being







Next Steps: Portable Code Explorations

- NVIDIA and CUDA are the current leaders in the GPU market, but the future will bring GPUs from a variety of vendors and possibly different heterogeneous solutions
- Need to explore solutions for code portability across platforms to avoid re-writing software for each platform
- many different options are emerging, mainly compiler directives or libraries • Started an effort to **test** different **portable implementations** using a single function from the mkFit code

 - function is the propagation of track parameters to a given position along the z axis - representative of full code in terms of math operations types; main limitation is that it does not include combinatorics
- Early tests using OpenACC and Kokkos give promising results - on a Summit Node OpenACC GPU application almost 10x faster than OpenMP
 - application fully loading the CPU
 - performance of Kokkos CPU version is similar to nominal implementation with dedicated Matriplex library
- Ongoing work towards a more completed suite of measurements, including OpenMP, OpenACC, Kokkos, Alpaka, Eigen, as well as different compilers







Reconstruction for LArTPC v experiments

- Charged particles produced in neutrino interactions ionize the argon, ionization electrons drift in electric field towards anode planes
- Sense wires detect the incoming charge, producing beautiful detector data images
- Reconstruction in LArTPC experiments is challenging due to unknown interaction point, many possible topologies, noise, contamination of cosmic rays
 - Takes O(minutes)/event in MicroBooNE
 - ICARUS ~5x bigger, DUNE Far Detector O(100)x bigger
- LArTPC detectors are modular in nature → parallelism!



Y wire plane waveform



lane waveforms
\sim
V —
<u></u>

Parallelization of HitFinder Algorithm

- Hit finding: identify pulses and determine their peak position and width
- Wires can be independently processed: suitable to demonstrate speedup potential by parallelizing LArTPC reconstruction
- Hit finder algorithm has been **successfully parallelized**:
 - replaced Gaussian fit based on Minuit+ROOT with a local implementation of Levenberg-Marquardt minimization early tests showed this lead to ~8x speedup in the minimization code already
- - vectorize specific loops within the minimization algorithm, typically across data bins
 - main limitations: only a subset of the code is vectorized, number of bins is same order as vector unit size
 - two-level nested multithreading parallelization over events and regions of interests in wires



2020/07/28 Parallelization for HEP Event Reconstruction - ICHEP2020

- takes a significant fraction of the reconstruction workflow (few to few tens of %, depending on the experiment)





	•	
		_
		_

Parallel HitFinder Results

- up to 30x (95x) for 80 (240) threads on Skylake Gold (KNL)
- Physics output nearly identical to original algorithm
- New version integrated in the experiment codebase (larsoft) and adopted by ICARUS and DUNE - Single-threaded application up to 10x faster than previous version, significant impact on reco time of experiments!



• Vectorization gives ~2x speedup when compiling with icc+AVX512 (both Skylake Gold and KNL) OpenMP multi-threading shows near ideal scaling at low thread counts, with speedups increasing





Studies of FFT libraries

- Fast Fourier Transform (FFT) is a core component of signal processing algorithms for LArTPC Comparing state of the art FFT libraries: FFTW and MKL
- MKL is precompiled with AVX-512; FFTW compiled with gcc and different vector extensions
- Test single fwd+inverse FFT on wire waveform data:
 - Up to ~3.5x speedup for FFTW with AVX2 and AVX-512; MKL ~10% faster than FFTW
- Also starting to look into FFT implementations for GPU







LArTPC Reconstruction on HPC

- Work is ongoing to develop a reconstruction workflow for HPC centers. Initial target experiment and center are ICARUS and **Theta**@ALCF, respectively.
- Goal is an efficient utilization of parallel resources of HPC centers, which requires building algorithm code with custom compilers and vector extensions
- the workflow is designed to be part of a central campaign Building experiment's software with Spack would allow us to use custom compilation options
 - icc and AVX-512 are needed for optimal vectorization speedups
 - initial tests of Spack build are complete
 - next steps: customization for specific packages, building on Theta
- Workflow being defined in collaboration with HEP-on-HPC SciDAC project
 - <u>https://computing.fnal.gov/hep-on-hpc/</u>
 - use HDF5+HEPnOS to organize the data: <u>https://xgitlab.cels.anl.gov/sds/hep/HEPnOS</u>
 - framework uses DIY to distribute the workload across nodes: https://www.anl.gov/mcs/diy-diff







Snowmass advertisement

- future directions and projects: <u>https://snowmass21.org/</u>
- Modernizing the reconstruction software is crucial for the future of HEP experiments. Join the Computational Frontier and the "Experimental topics of this talk
 - Computational Frontier Workshop: <u>https://indico.fnal.gov/event/43829/</u>
 - Experimental Algorithm Parallelization WG: https://snowmass21.org/computational/algorithms



Join the Snowmass process to plan for the next years and prioritize possible

Algorithm Parallelization" Working Group, which is particularly relevant for the









Summary

- up reconstruction of current and future HEP experiments
- towards deployment in workflows and exploring GPU applications.
- usage of HPC centers

• Projects are actively working to exploit new computing architectures to speed

 mkFit: Kalman filter track building with large speedups from parallelization. Integrated in CMSSW, 6x faster single thread application. Work ongoing

 LArTPC reconstruction: vectorized and multi-threaded hit finder O(10)x faster and adopted by experiments. Work in progress to define workflow for efficient







Acknowledgments

- Project websites:
 - https://trackreco.github.io/
 - https://computing.fnal.gov/hepreco-scidac4/
- Funding agencies and collaborating institutions:









Cornell University.





UNIVERSITY OF OREGON







Fermilab





PRINCETON UNIVERSITY

