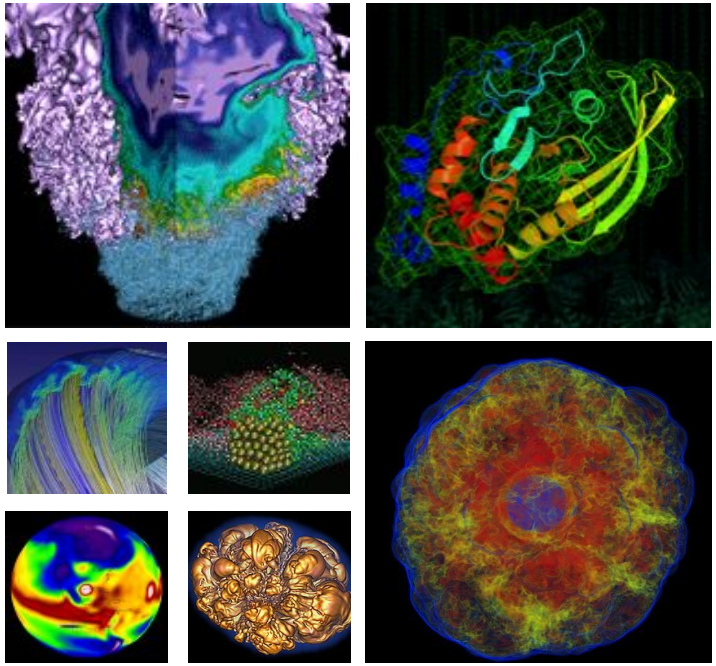
Accelerating LArTPC Simulations: Enhancing larnd-sim with GPU Optimization Techniques



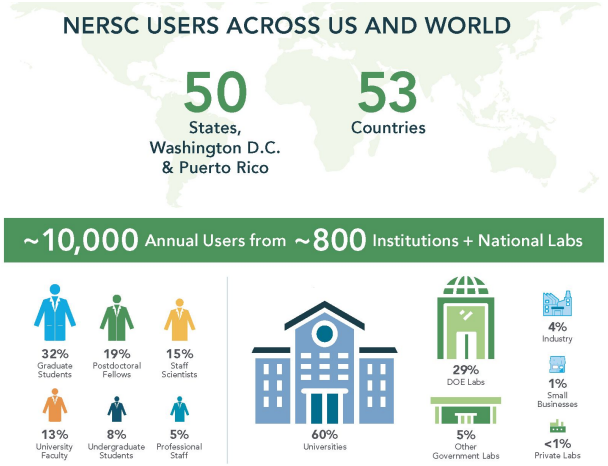
CUG Conference
May 4-8 2025
Jersey City

Madan Timalsina
NERSC/NESAP Postdoc
Data & AI Services
LBNL

On the behalf of NERSC-NESAP and DUNE team:
Madan Timalsina, Matt Kramer, Pengfei Ding, Ronan Doherty, Rishabh Dave, Nicholas Tyler, Urjoshi Sinha, William Arndt, Callum Wilkinson

NERSC

- NERSC is the DOE Office of Science's premier facility for high-performance computing and data analysis
- Perlmutter, NERSC's flagship supercomputer, accelerates AI, data analysis, and simulations with its hybrid CPU-GPU architecture
- One of the largest supporters of physical science research in the U.S



Acknowledged in ~ 6000 refereed scientific publications & high profile journals since 2020

U.S. DEPARTMENT OF ENERGY | Office of Science

DUNE DEEP UNDERGROUND NEUTRINO EXPERIMENT

BERKELEY LAB

NERSC Science Acceleration Program (NESAP)

NERSC Science Acceleration Program (NESAP) facilitates collaboration with code teams, vendors, and developers to optimize scientific applications for cutting-edge computational architectures and next-generation supercomputing systems

Why NESAP?

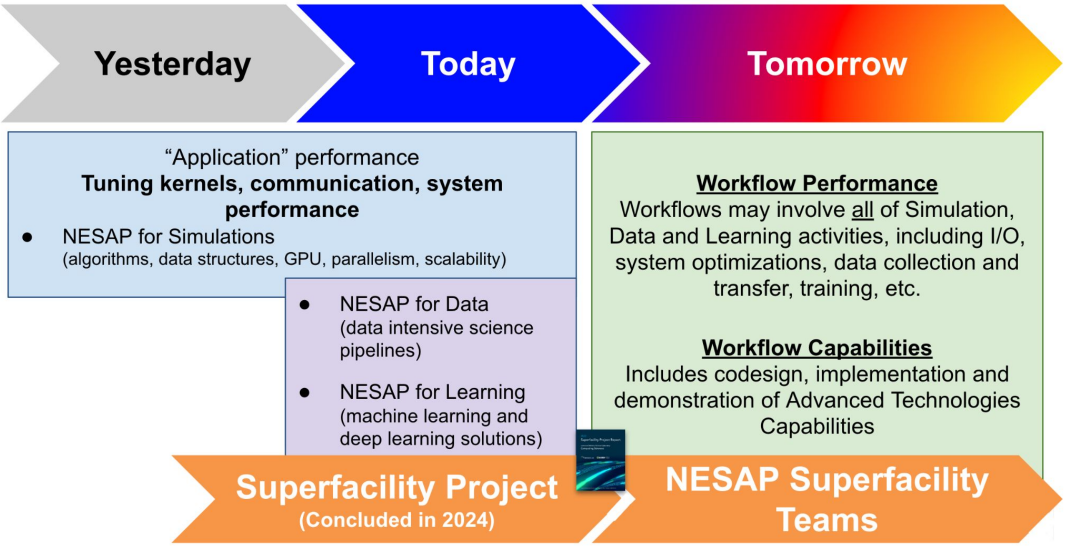
Step 1: HPC is hard !

Step 2: We work (in-depth) with ~40 teams (the maximum number we have enough staff for) for about 3 years, to figure out how they can make the best use of our new HPC systems

- We help them improve their code (e.g. CPU => GPU)
- Lessons learned are used to improve the data center (e.g. new GPU programming models)

Step 3: Now HPC is slightly less hard !

NESAP is Evolving

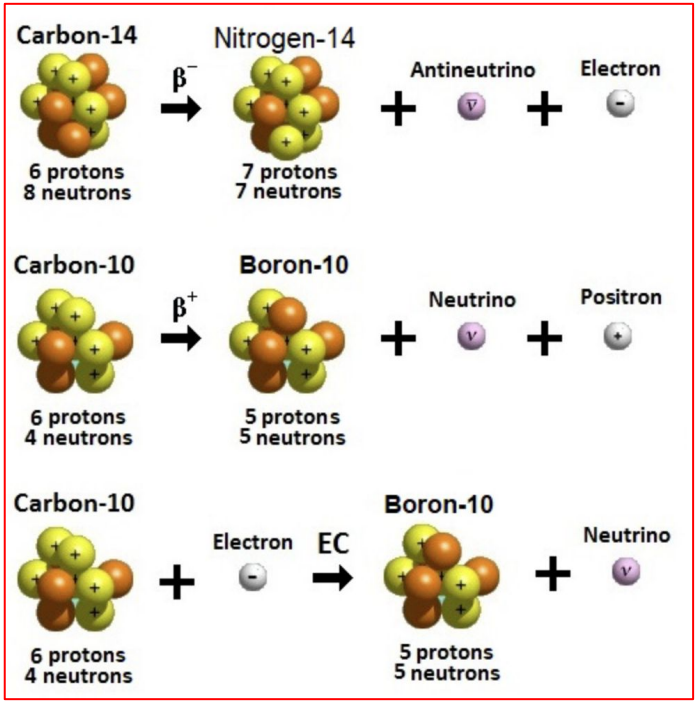


Deep Underground Neutrino Experiment (DUNE)

The **D**eep **U**nderground **N**eutrino **E**xperiment (**DUNE**) aims to measure neutrino oscillations between three flavors: **electron neutrino, muon neutrino, and tau neutrino**, as well as study **supernova neutrinos** and search for **proton decay**!

What is a neutrino?

- A **fundamental particle**, was hypothesized to **conserve energy in beta (β) decays**.
- Observed spectrum didn't match a two-body decay, so a **third particle was introduced to explain the continuous β-spectrum** by ensuring the sum of energies of the proton and electron remains constant.



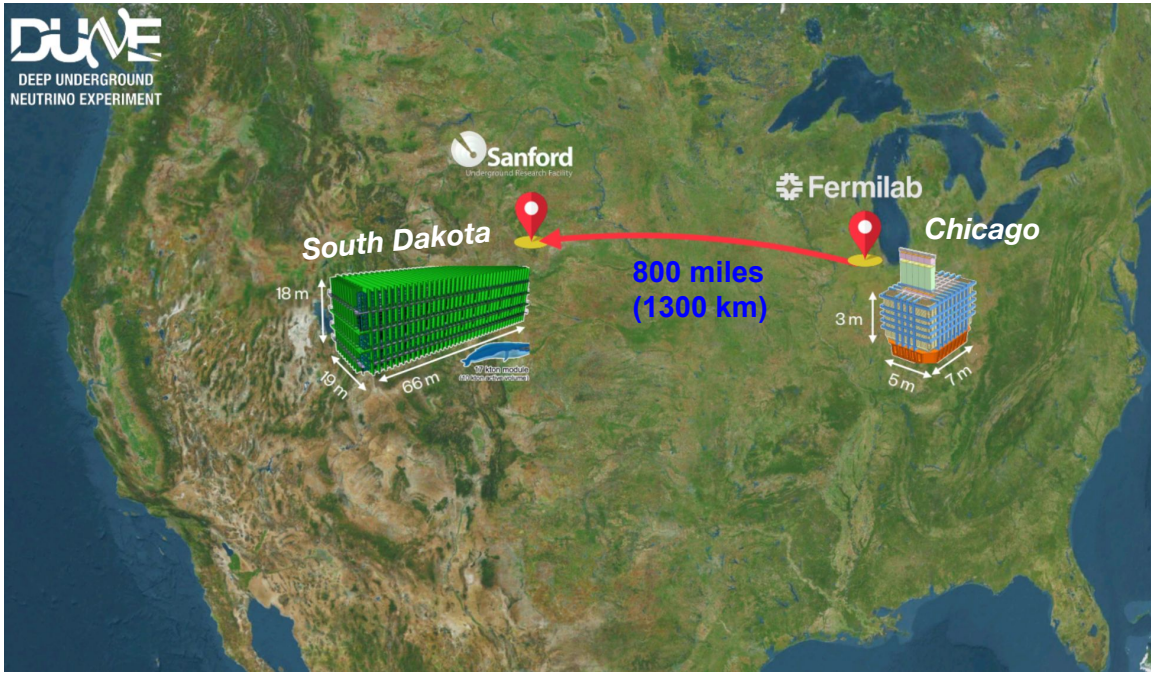
Why neutrinos?

- Neutrinos could explain why the universe is made of **matter rather than antimatter**.
- Detecting supernova-produced neutrinos can aid in understanding the formation of **neutron stars or black holes**.
- The same detector can check for **proton decay**, potentially indicating a sign that three of the four fundamental forces we know merge into a single **Grand Unified Theory**.

DOI:10.13140/RG.2.2.27436.00644

Deep Underground Neutrino Experiment (DUNE)

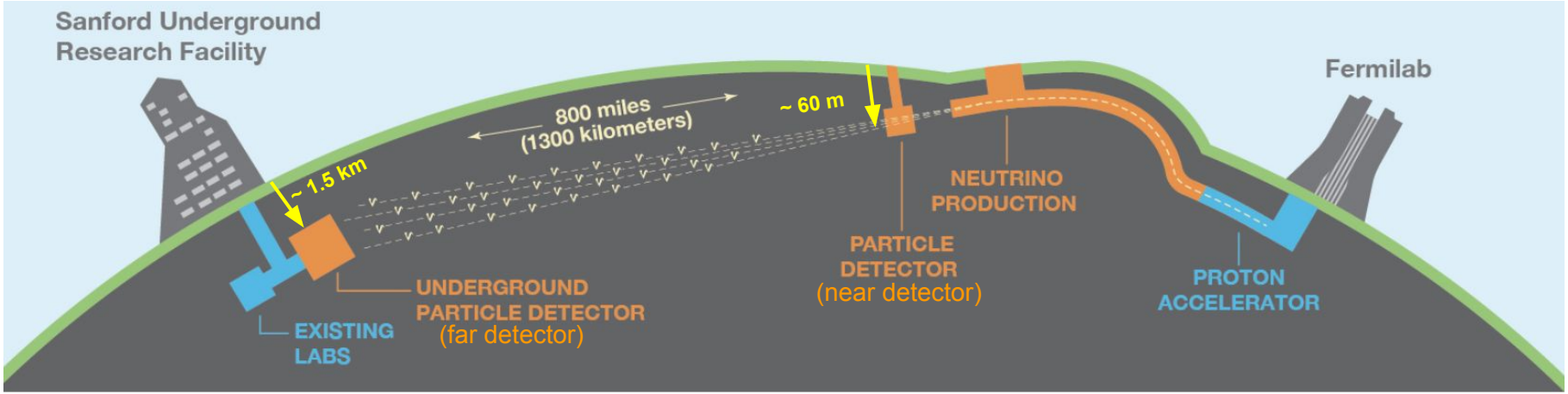
- The DUNE neutrino experiment involves over 1,500 scientists and engineers from more than 200 institutions in 36 countries
- **Next generation long-baseline accelerator neutrino oscillation experiment** in U.S (would be largest experiment on U.S. soil)
- Physics program: **Leptonic CP violation, neutrino mass hierarchy, supernova burst neutrino detection** & much more!
- **Very intense beam and very large far detectors** (each module have 62.0m x 15.1m × 14.0m containing a total LAr mass of about 17.5 kt)



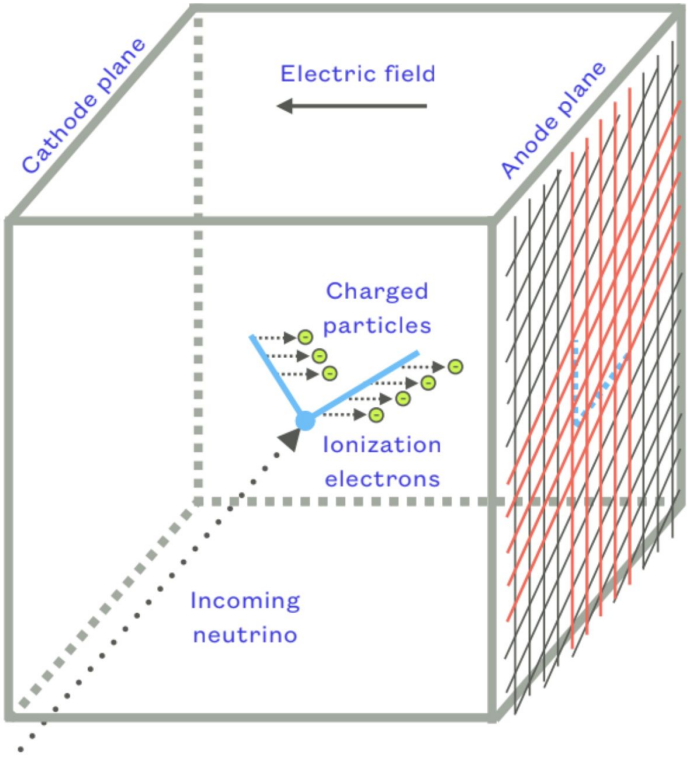
Deep Underground Neutrino Experiment (DUNE)

How is the DUNE beam produced?

- High-energy protons (120 GeV) slam into a target
- Outgoing π+ (π-) are focused by magnetic horns
- Pions in decay tunnel produce muon and neutrinos (or anti-neutrinos)
- Neutrinos travel 800 miles through the earth to reach the far detectors
- **Very long baseline;** matter effect enhances mass ordering sensitivity

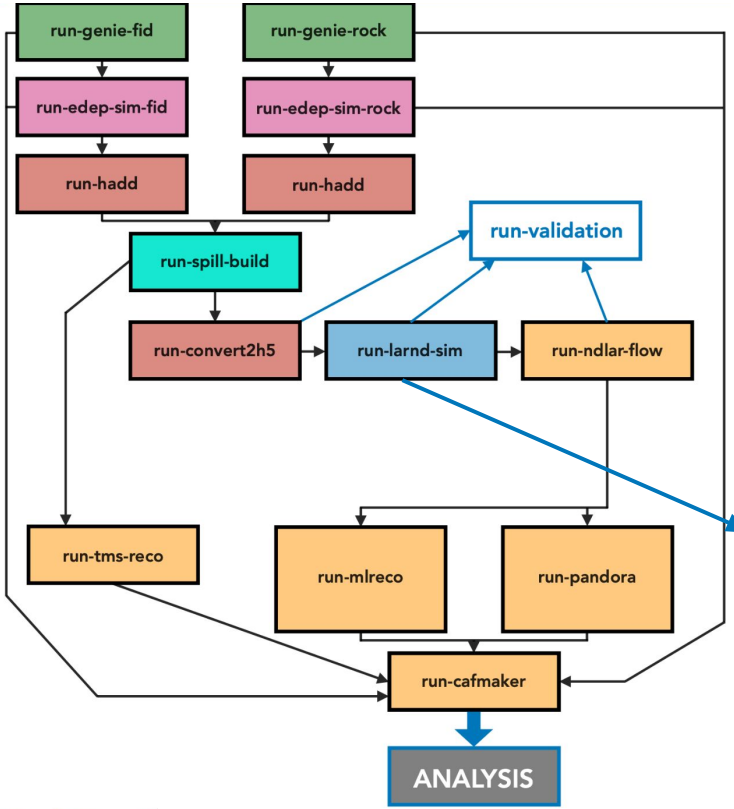


Time Projection Chamber (TPC)



- A **L**iquid **Ar**gon **T**ime **P**rojection **C**hamber (**LArTPC**) consists of a structure immersed in liquid argon with a constant electric field applied across it.
- Neutrinos (infrequently) interact with atomic nuclei and generate charged particles (e.g., $\boldsymbol{\nu_\mu + n \rightarrow \mu^- + p}$).
- The charged particles ionize the liquid argon, causing ionization electrons to move towards the anode plane at a steady speed.
- The DUNE near detector will be the first large-scale LArTPC to use a truly 2D *pixelated* anode, and **larnd-sim** was developed specifically for pixelated LArTPCs.

Simulation and Reconstruction Workflow



Neutrino Generation

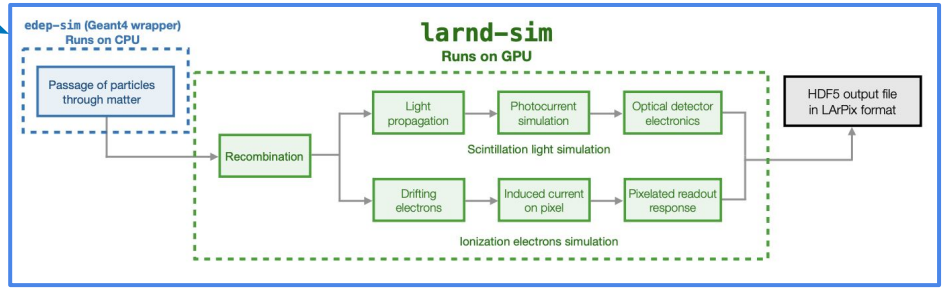
Particle Propagation

File Conversion

Spill Building

Detector Simulation

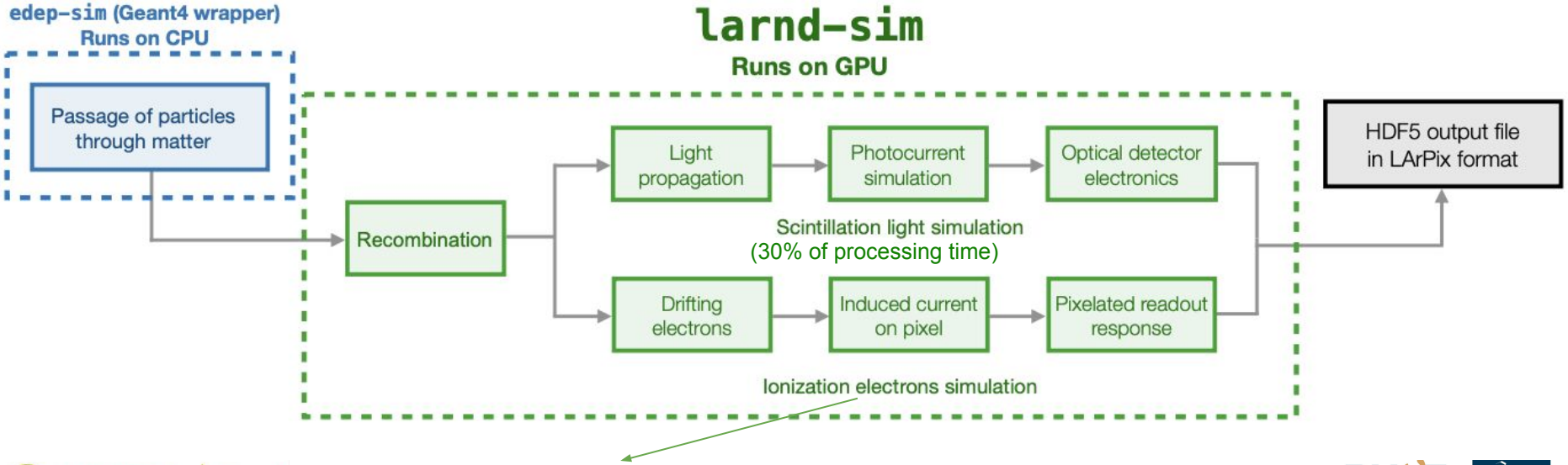
Reconstruction



DUNE: LArND Simulation

- Novel detector simulation for pixelated LArTPCs:
  - Charge drifting, analog front end, digitization
  - Scintillation, light detection
- Input: HDF5 file with energy depositions from Geant4
- Output: Simulated data stream, plus MC truth information

- Idiomatic Python, Numba JIT-compiled to CUDA:
  - Just apply `@numba.cuda.jit` decorator
  - `cupy`: NumPy on the GPU
  - ~ 15 kernels



https://arxiv.org/abs/2212.09807

DUNE: LArND Simulation

Example: CUDA kernel for charge quenching

CUDA JIT kernel for simulating charge recombination on Geant4 tracks

Retrieve thread index and check array bounds

Energy loss and total energy deposition

Compute recombination factor for two modes: Box and Birks
&
Error handling

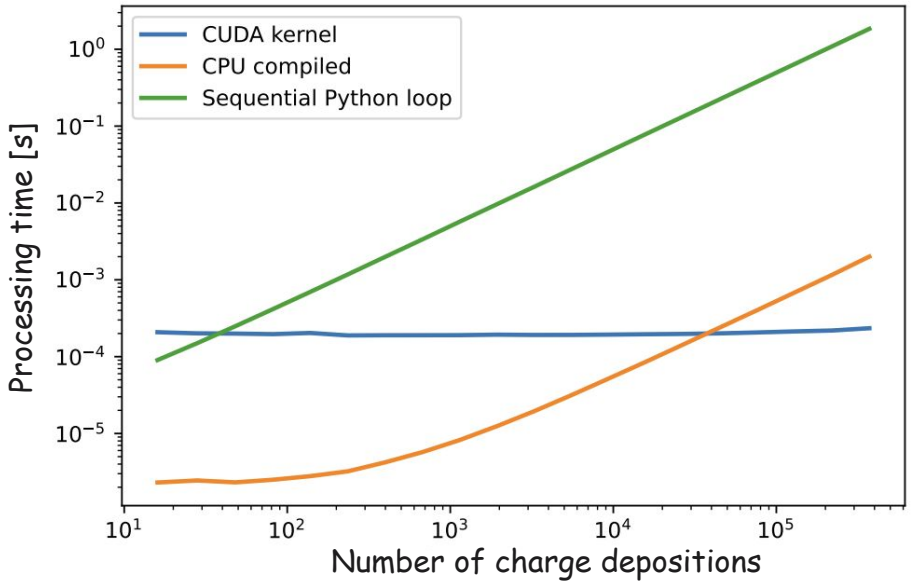
Compute the number of electrons & photons

```
@cuda.jit
def quench(tracks, mode):
    itrk = cuda.grid(1)

    if itrk < tracks.shape[0]:
        dEdx = tracks[itrk]["dEdx"]
        dE = tracks[itrk]["dE"]

        recomb = 0
        if mode == physics.BOX:
            # Baller, 2013 JINST 8 P08005
            csi = physics.BOX_BETA * dEdx / (detector.E_FIELD * detector.LAR_DENSITY)
            recomb = max(0, log(physics.BOX_ALPHA + csi)/csi)
        elif mode == physics.BIRKS:
            # Amoruso, et al NIM A 523 (2004) 275
            recomb = physics.BIRKS_Ab / (1 + physics.BIRKS_kb * dEdx / (detector.E_FIELD * detector.LAR_DENSITY))
        else:
            raise ValueError("Invalid recombination mode: must be 'physics.BOX' or 'physics.BIRKS'")

        if isnan(recomb):
            raise RuntimeError("Invalid recombination value")

        tracks[itrk]["n_electrons"] = recomb * dE / physics.W_ION
        tracks[itrk]["n_photons"] = (dE/light.W_PH - tracks[itrk]["n_electrons"]) * light.SCINT_PRESCALE
```

DUNE: LArND Simulation

Example: CUDA kernel for charge quenching

- **The GPU implementation is exponentially faster than the CPU version (for charge deposition > $5x10^4$), leveraging massive parallelization**
- Typical neutrino beam spill in ND-LAr averages **10^5 number of charge deposition** (segments)
- **Right plot: Processing time for calculating recombination factor** as a function of number of charge depositions (GEANT4 segments), comparing sequential Python loop, CPU and GPU
- **Implemented using JIT-compiled Numba** for both CPU and GPU. The segments array is allocated on the GPU using CuPy, a NumPy-compatible library accelerated by CUDA



Credit: Roberto Soleti [1]

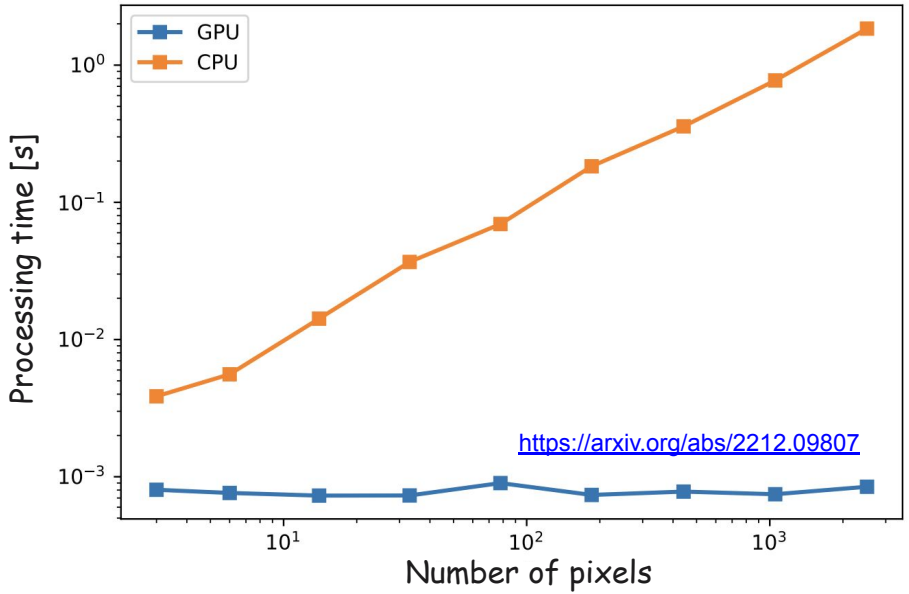
Note: Tested with NVIDIA® Tesla® V100 (Volta) GPUs

[1] DUNE collab. (2023), "Highly-parallelized simulation of a pixelated LArTPC on a GPU", J. Inst. 18 (P04034)

DUNE: LArND Simulation

Example: Electronics response simulation

- **CUDA kernel is exponentially faster than CPU, over three orders of magnitude improvement for simulations with O(10^3) pixels**
- Pixel pads are individually read out by application-specific integrated circuits (ASICs) that include an amplifier, discriminator, and digitizer for each pixel
- larnd-sim sums the currents induced by different tracks on the same pixel, before simulating the digitization process
- CUDA kernel simulates trigger logic and noise sources
  - Three types of noise included: reset noise, discriminator noise, and uncorrelated noise
- Threads organized in a one-dimensional grid for parallel processing of pixel data



Credit: Roberto Soleti [1]

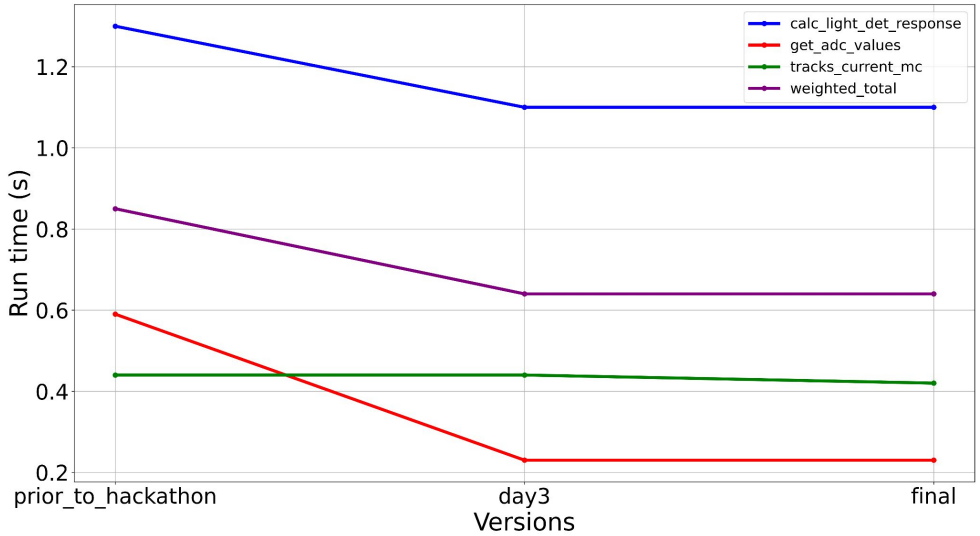
Note: Tested with NVIDIA® Tesla® V100 (Volta) GPUs

[1] DUNE collab. (2023), "Highly-parallelized simulation of a pixelated LArTPC on a GPU", J. Inst. 18 (P04034)

LArND Simulation: Hackathon with OPEN HACKATHONS NERSC

Hackathon Preparation

- Developed **"miniapps"** for the three most computationally expensive GPU kernels
- Implemented a script for configuring, executing, and profiling the full simulation using **Nsight Systems / Compute**



Hackathon Achievements

- Optimized GPU performance through:
  - **Register tuning**: Adjusted maximum register count to balance thread occupancy and register availability
  - **Memory & execution tuning**: Improved grid/block dimensions, optimized memory transfers
  - **Floating-point optimization**: Fixed precision issues and enabled **"fastmath"** for transcendental functions
  - And more!
- **Achieved:**
  - 10–500% kernel speedup (3 different kernels)
  - ~20% overall simulation speedup

Weighted total =

Light waveform simulation *x 50%*
+ Charge digitization *x 25%*
+ Charge-induced analog current *x 10%*

U.S. DEPARTMENT OF ENERGY | Office of Science

OPEN HACKATHONS https://www.openhackathons.org/s/

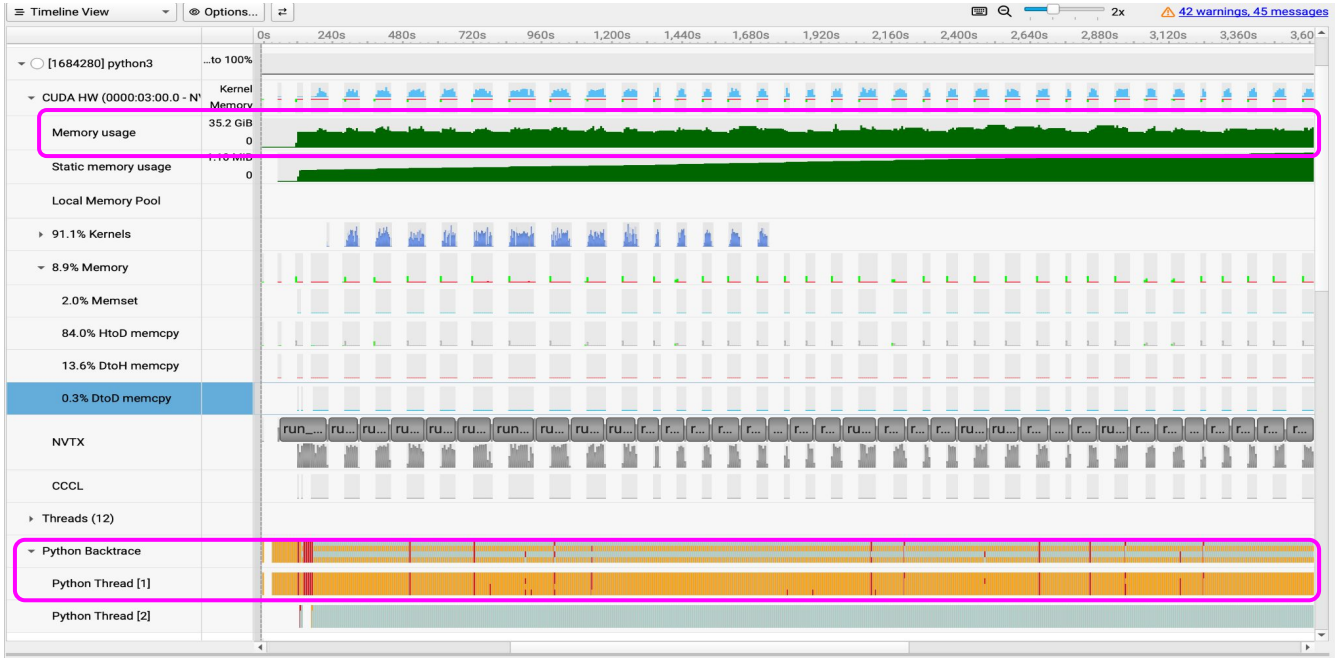
DUNE DEEP UNDERGROUND NEUTRINO EXPERIMENT

BERKELEY LAB

LArND Simulation: Optimization

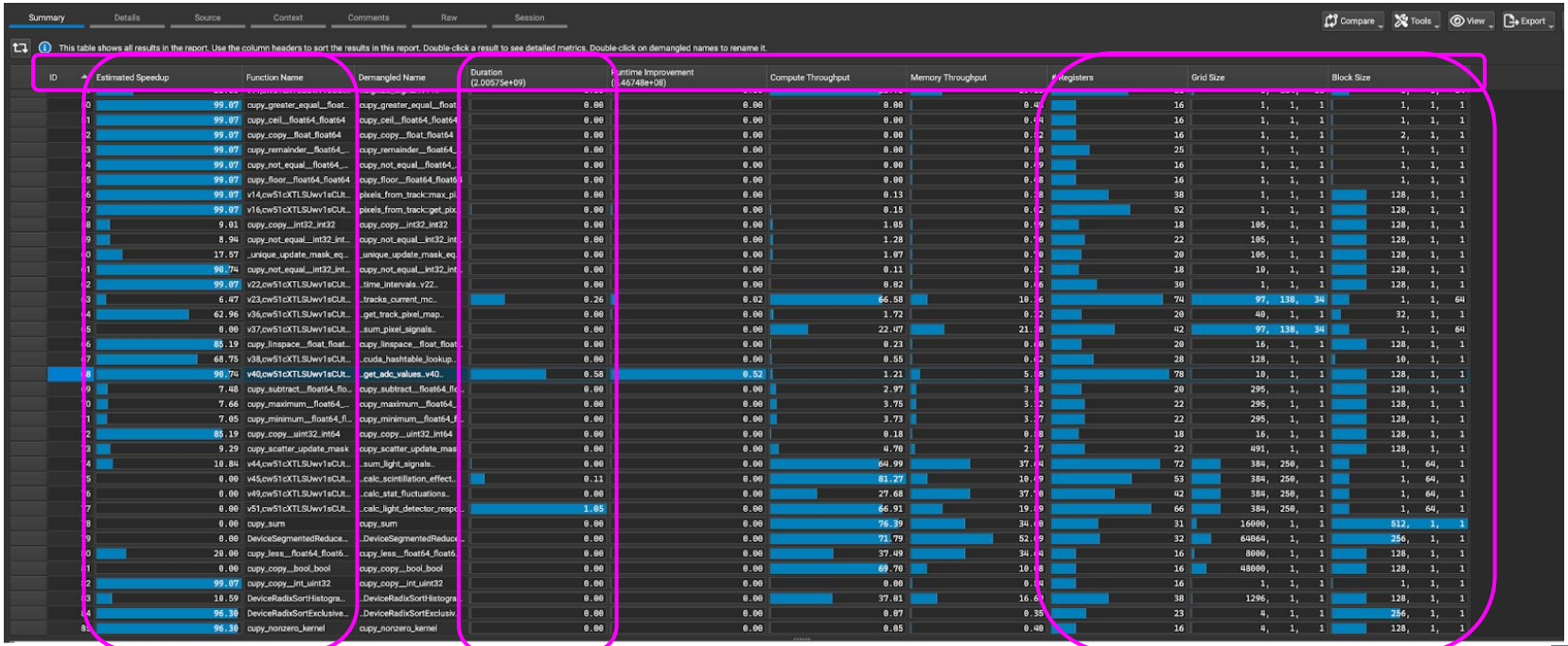
- Used Nsight Systems and Nsight Compute
- Identified origins of high memory usage using timeline graph and Python stacktrace
- Pinpointed specific lines of code for large memory allocations
- Converted a regular 3D array to an irregular (2+1)D "jagged" array
- Significant achievement in the peak memory uses and simulation run time

Nsight Systems



LArND Simulation: Optimization

Nsight Compute helps optimize kernel performance by providing insights into grid/block usage, register usage, Streaming Multiprocessor (SM) utilization, occupancy etc.

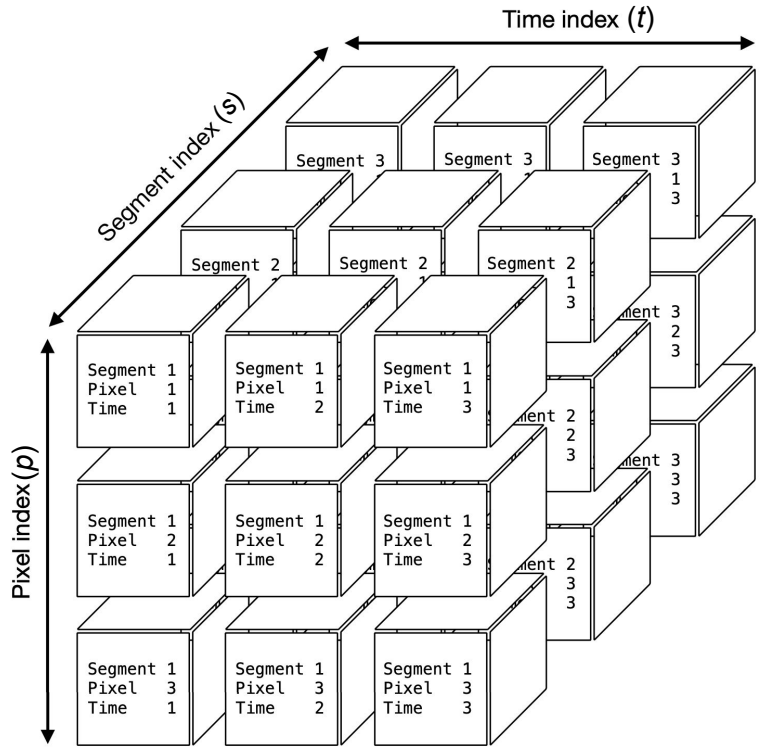


DUNE: LArND Simulation

Induced current simulation:

Example use of jagged array

- **Using jagged array significantly reduces runtime and peak memory usage**
- Converted this regular 3D array to an irregular (2+1)D “jagged” array
- Kernel calculates the current induced by track segment *s* on pixel *p* at time *t*
  - Kernel executes on 3D CUDA grid over (*s, p, t*)
- The # of contributing segments varies from pixel to pixel
- Originally used fixed-dimensional 3D array to store the output, with lots of wasted padding for those pixels that only “see” a handful of segments

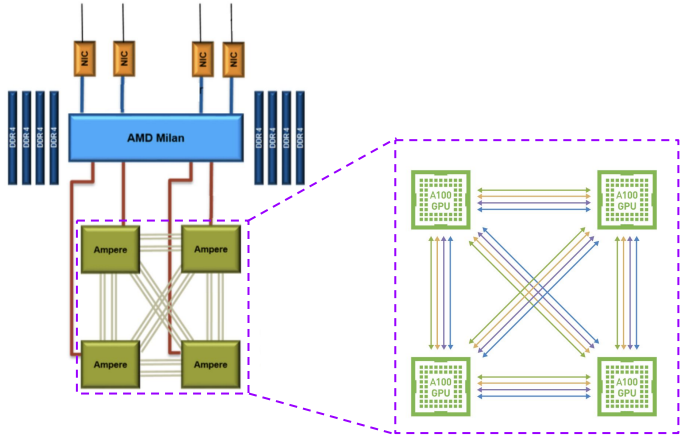


LArND Simulation: Benchmarking

Partition	# of nodes	CPU	GPU	NIC
GPU	1536	1x AMD EPYC 7763	4x NVIDIA A100 (40GB)	4x HPE Slingshot 11
	256	1x AMD EPYC 7763	4x NVIDIA A100 (80GB)	4x HPE Slingshot 11
CPU	3072	2x AMD EPYC 7763	-	1x HPE Slingshot 11
Login	40	2x AMD EPYC 7713	1x NVIDIA A100 (40GB)	-

Benchmarking and Simulation:

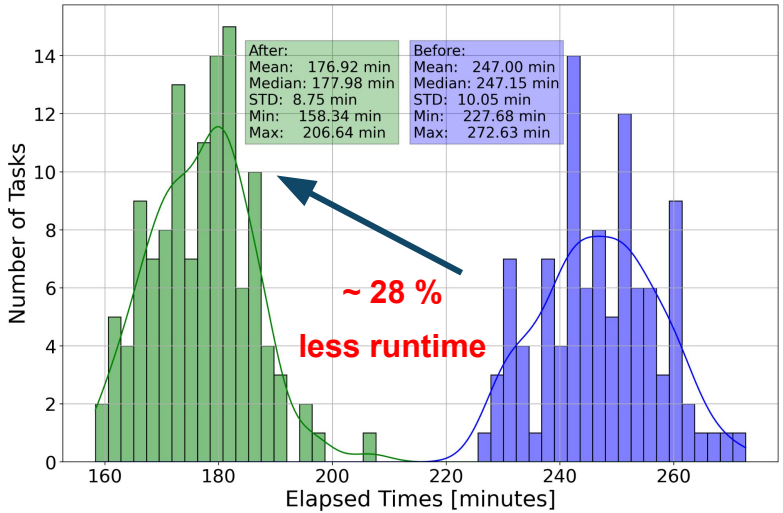
- Benchmarked the runs on both flavors of GPU nodes (A100 with 40GB and 80GB HBMs)
- Developed workflow to control the simulation and submit jobs on Perlmutter using SLURM
- Tested various CUDA versions (11.7, 12.2, 12.4)
- Simulations included different configuration:
  - *2x2_variation* (4 reduced-scale detector modules)
  - *ndlar* (35 full-scale detector modules)



LArND Simulation: Benchmarking

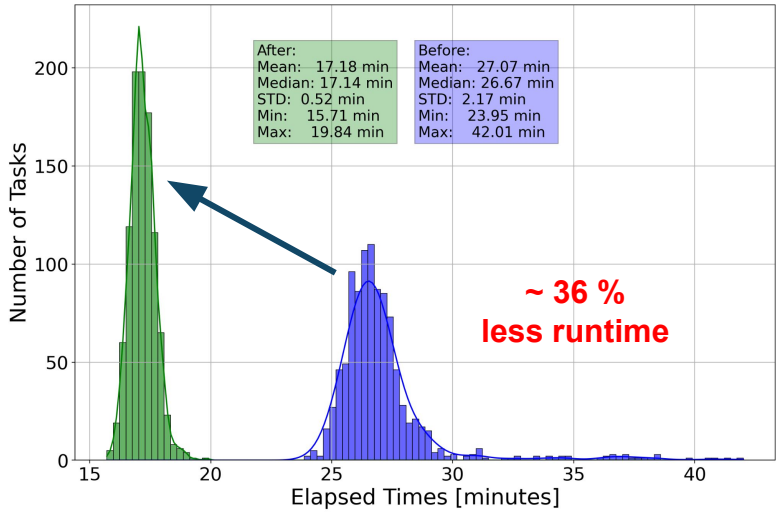
Simulation Runtime

Ndlar
(35 full-scale detector modules)



- 124 input files using 80GB/GPUs, 122 task completed, 2 failed due to out of GPU memory
- Submitted jobs on 5 nodes, using 7 arrays, each with 20 GPU tasks (one gpu per task)
- ~ 28 % reduction in runtime (mean to mean) after optimization

2x2_variation
(4 reduced-scale detector modules)

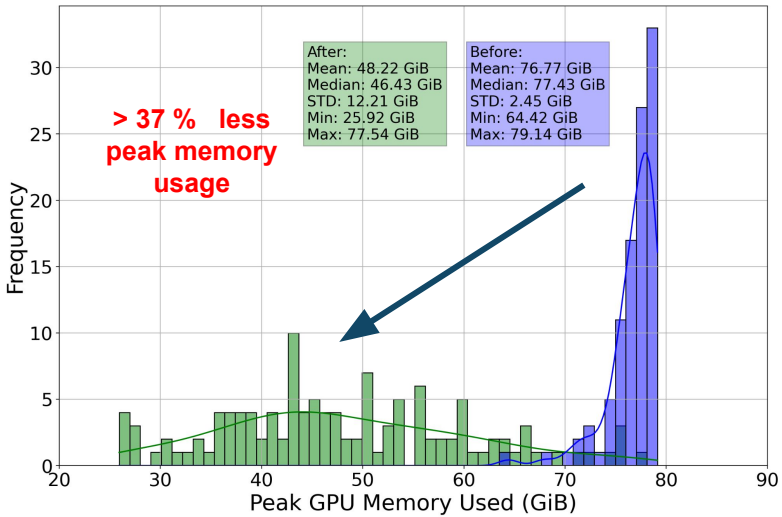


- 1000 input files using 80GB/GPUs, all completed tasks
- Submitted jobs on 5 nodes, using 5 arrays, each with 20 GPU tasks (one gpu per task)
- 20 tasks submitted simultaneously on each array, required 10 submission cycles to complete the jobs
- ~ 36% reduction in runtime (mean to mean) after optimization

LArND Simulation: Benchmarking

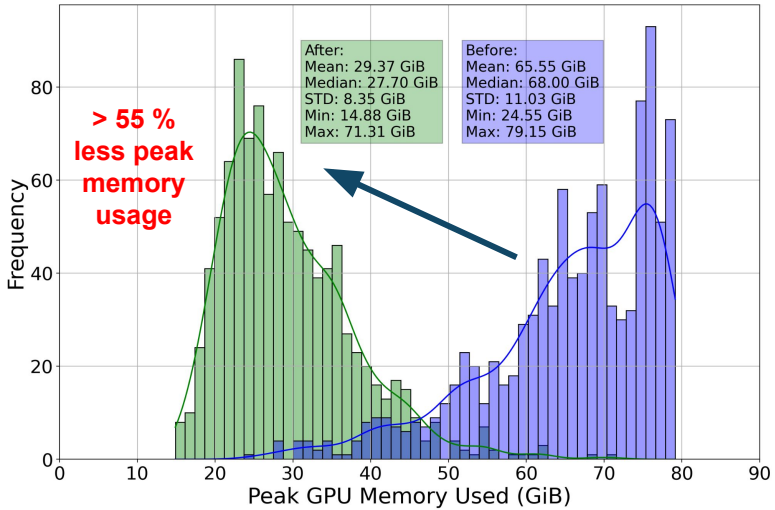
Peak Memory Uses

`Ndlar`
(35 full-scale detector modules)



- 124 input files using 80GB/GPUs, 122 task completed, 2 failed due to out of GPU memory
- Submitted jobs on 5 nodes, using 7 arrays, each with 20 GPU tasks (one gpu per task)
- More than 37 % reduction in peak memory use (mean to mean) after optimization

`2x2_variation`
(4 reduced-scale detector modules)

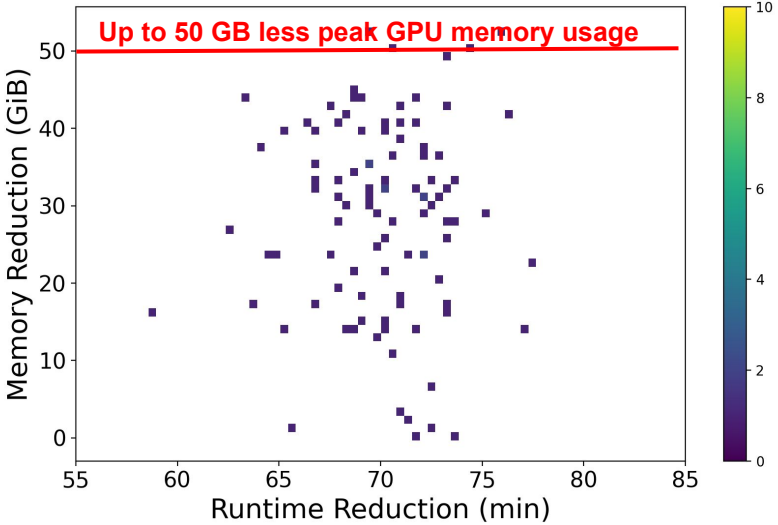


- 1000 input files using 80GB/GPUs, all completed tasks
- Submitted jobs on 5 nodes, using 5 arrays, each with 20 GPU tasks (one gpu per task)
- 20 tasks submitted simultaneously on each array, required 10 submission cycles to complete the jobs
- More than 55% reduction in peak memory use (mean to mean) after optimization

LArND Simulation: Benchmarking

Ndlar

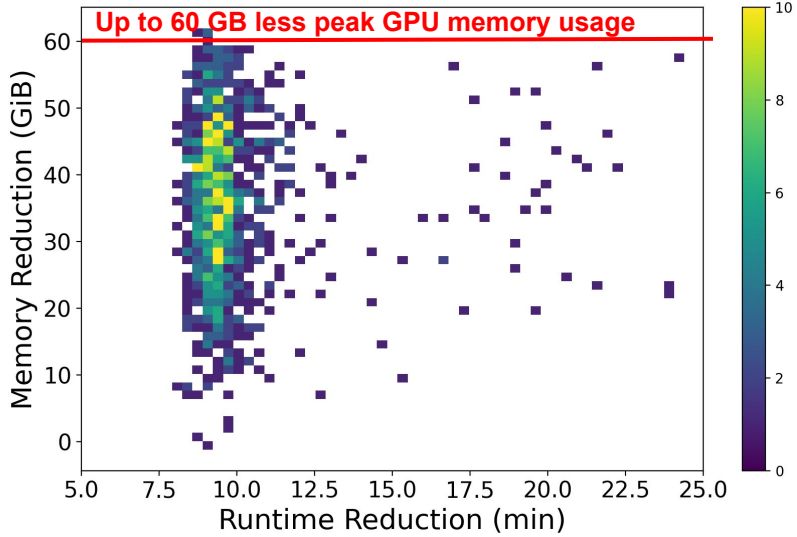
(35 full-scale detector modules)



- 124 input files using 80GB/GPUs, 122 task completed, 2 failed due to out of GPU memory
- Submitted jobs on 5 nodes, using 7 arrays, each with 20 GPU tasks (one gpu per task)
- Up to ~50 GB reduction in peak memory use after optimization

2x2_variation

(4 reduced-scale detector modules)



- 1000 input files using 80GB/GPUs, all completed tasks
- Submitted jobs on 5 nodes, using 5 arrays, each with 20 GPU tasks (one GPU per task)
- 20 tasks submitted simultaneously on each array, required 10 submission cycles to complete the jobs
- Up to ~60 GB or reduction in peak memory use after optimization

LArND Simulation: Ongoing/future works

- In addition to charge simulation, re-enable full ND-LAr light simulation and continue optimizing and benchmarking its memory use and runtime
- Continuously addressed bottlenecks by converting large arrays to jagged structures, significantly improving memory usage and performance
- Refining pixel sub-batching strategies to lower peak GPU memory usage as well as runtime usage
- Parameterizing/compressing per-voxel light-emission time distributions to save gigabytes and accelerate simulation
- Planning containerization for easy, fast, portable deployment
- In addition to NERSC Perlmutter, planning to run on ALCF (Argonne Leadership Computing Facility)

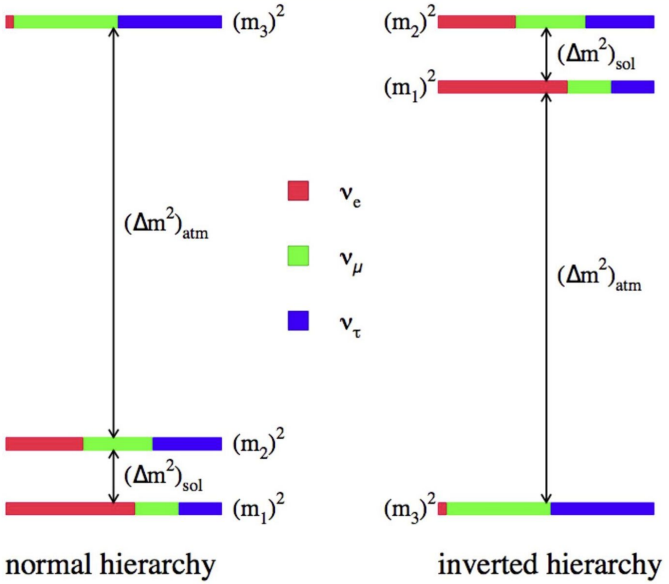
LArND Simulation: Summary

- NESAP partners with science teams to optimize codes on NERSC's hybrid CPU-GPU system
- **larnd-sim:** Python/Numba+CuPy GPU simulator streamlining PLArTPC modeling and exploiting massive parallelism
- **larnd-sim:** One of the few GPU-based simulation techniques in high-energy physics (HEP) deployed in practice
- **Through detailed optimization, we achieved:**
  - **>50% less peak GPU memory usage.**
  - **30% less runtime.**
  - **Kernel speedups 10% - 500%.**
- We continue refining larnd-sim, aiming to extend these methods to other noble-liquid detector technologies
- For more details and to follow our progress, visit the larnd-sim GitHub repository:
  - https://github.com/DUNE/larnd-sim/tree/develop

Thank You

Neutrino Mass Hierarchy

- **Problem**
  - Current neutrino oscillation data support two possible mass orderings:
- **Normal Hierarchy (NH) / Normal Ordering (NO)**
  - Two lightest neutrinos are close in mass (~10 meV difference)
  - Third neutrino is about 50 meV heavier
- **Inverted Hierarchy (IH) / Inverted Ordering (IO)**
  - Lightest neutrino is alone
  - The other two are heavier by ~50 meV, with ~10 meV between them
- **Current status**
  - Data slightly favor Normal Ordering (NO) over Inverted Ordering (IO)

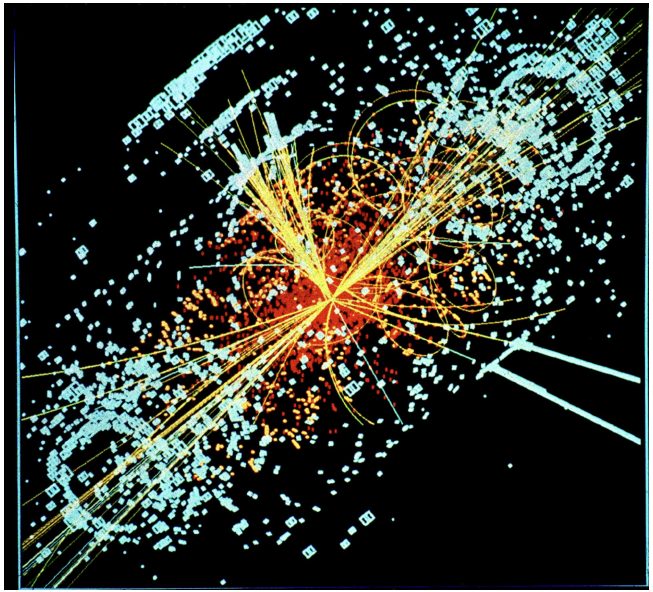


Leptonic CP Violation

- **What is CP Symmetry?**
  - C (Charge conjugation): Swap particles with their antiparticles
  - P (Parity): Mirror the spatial coordinates
  - CP Symmetry: Physics should stay the same under both transformations
  - Discovered in 1964 in meason (neutral kaon decays), awarded the 1980 Nobel Prize (Cronin & Fitch)

- **Why it matters**
  - Helps explain why the universe is made of more matter than antimatter
  - Key in understanding weak interactions

- **Leptonic CP Violation**
  - If too small, can't explain matter-antimatter asymmetry
  - Implies new physics beyond the Standard Model may be needed
  - New particles or interactions could bring additional CP violation



Simulated data from the CMS detector at the LHC showing a Higgs boson created in proton-proton collisions, decaying into hadron jets and electrons.

DUNE: LArND Simulation

Kernel Profiling

- Code profiled using NVIDIA Nsight Systems and Compute tools.
- ***Induced current*** CUDA kernel accounts for 97.5% of processing time.
- Kernel utilizes 99.9% of its time for computing operations.
- Memory transfer time is negligible in overall processing.
- Roofline analysis indicates the algorithm is compute-bound.