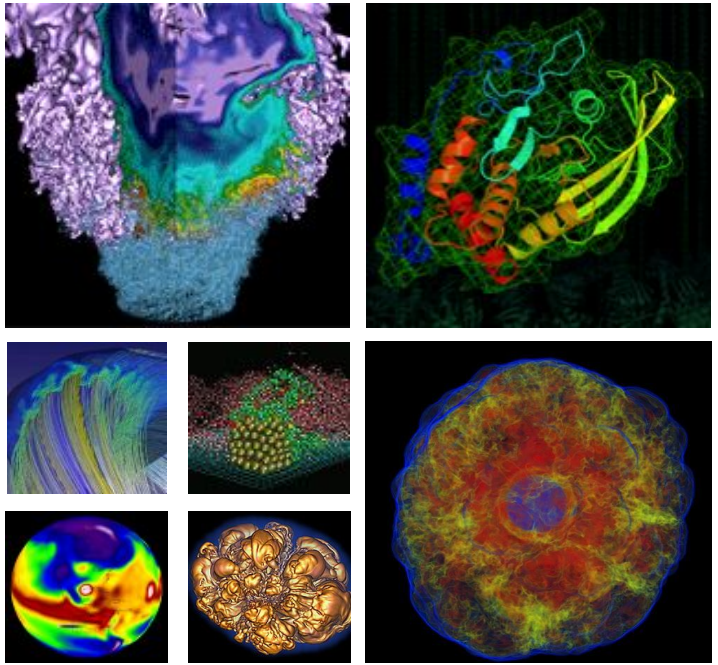
DUNE
ND-LAr



Madan Timalsina

NERSC/NESAP Postdoc
Data & AI Services

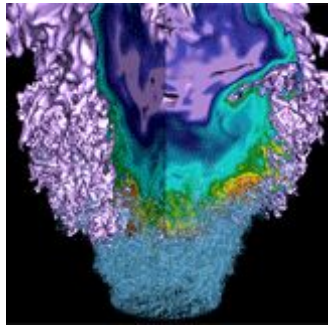
Sep 13, 2024

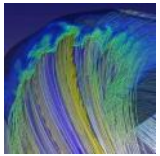
NERSC

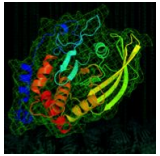
U.S. DEPARTMENT OF ENERGY | Office of Science

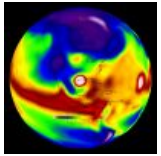
BERKELEY LAB

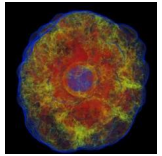
Benchmarking

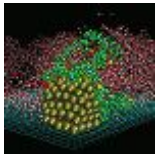












NERSC

DUNE larnd-sim Benchmarking

- **Repositories and descriptions:**
  - contains code for simulation, written in python, using CUDA-11.7 and GPU
    - https://github.com/DUNE/larnd-sim
  - Benchmarked branches
    - `develop` (commit `46187d51`) and `post-hackathon2024`
  - example to use the larnd-sim
    - https://github.com/lbl-neutrino/larnd-sim-example
  - Benchmark results:
    - https://github.com/lbl-neutrino/larnd-sim-example/tree/pre-post-hackathon-MT
      - https://github.com/lbl-neutrino/larnd-sim-example/tree/pre-post-hackathon-MT/develop
      - https://github.com/lbl-neutrino/larnd-sim-example/tree/pre-post-hackathon-MT/post-hackathon2024

DUNE larnd-sim Benchmarking-80GB GPUs

- Benchmarking 1000 input files on GPUs, all completed tasks
- 80GB bandwidth per GPU.
- Batch script submits jobs on 5 nodes.
- 5 arrays, each with 20 GPU tasks.
- Each array processes 200 tasks.
- 20 tasks submitted simultaneously per array.
- 10 submission cycles per array.
- Different random seed for each task

  rand_seed=$(((RANDOM % 10000) + 1))

Dataframe (csv file) contains following informations

```
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   file_index            1000 non-null   int64
 1   input_file            1000 non-null   object
 2   output_file           1000 non-null   object
 3   log_file              1000 non-null   object
 4   gpu_log_file          1000 non-null   object
 5   sim_start_time        1000 non-null   object
 6   task_start_time       1000 non-null   object
 7   elapsed_time(s)       1000 non-null   float64
 8   task_end_time         1000 non-null   object
 9   random_seed           1000 non-null   int64
 10  job_id                1000 non-null   int64
 11  host_name             1000 non-null   object
 12  gpu_device_id         1000 non-null   object
 13  peak_gpu_memory_used  1000 non-null   int64
 14  gpu_memory_start      1000 non-null   int64
 15  gpu_memory_end        1000 non-null   int64
 16  job_completed         1000 non-null   bool
 17  job_error_msg         0 non-null      float64
 18  input_file_size_mb    1000 non-null   float64
 19  output_file_size_mb   1000 non-null   float64
dtypes: bool(1), float64(4), int64(6), object(9)
memory usage: 149.5+ KB
```

Input files (1000)

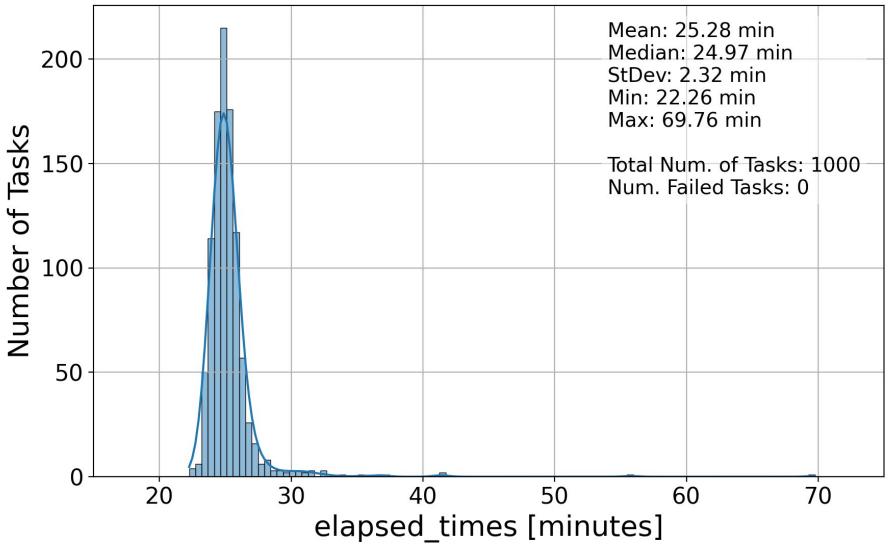
/dvs_ro/cfs/cdirs/dune/www/data/2x2/simulation/productions/MiniRun5_1E19_RHC/MiniRun5_1E19_RHC.convert2h5/EDEPSIM_H5/0000000

Output, analysis file and batch script can also be found here:

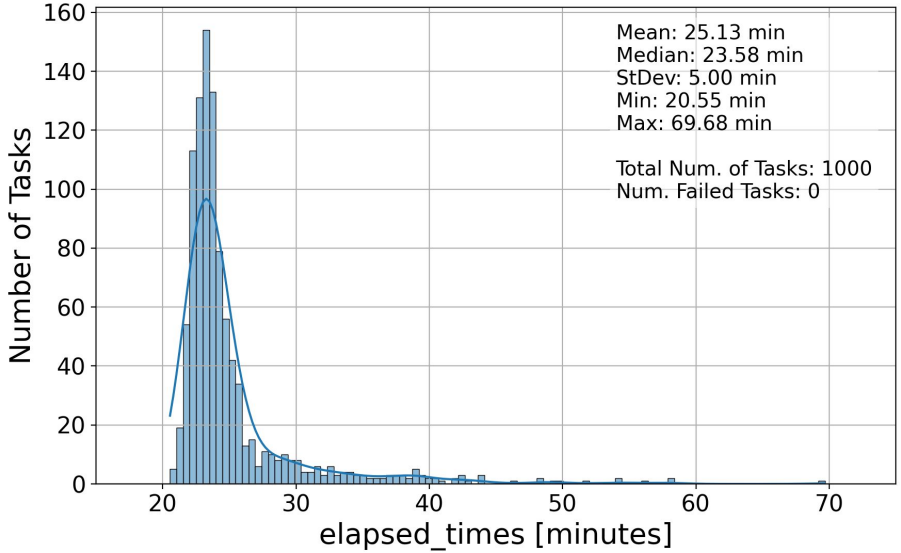
/global/cfs/cdirs/dune/users/madan12/DUNE_nesap/pre-post-hackathon

Elapsed Time [minutes]

Develop



post-hackathon2024

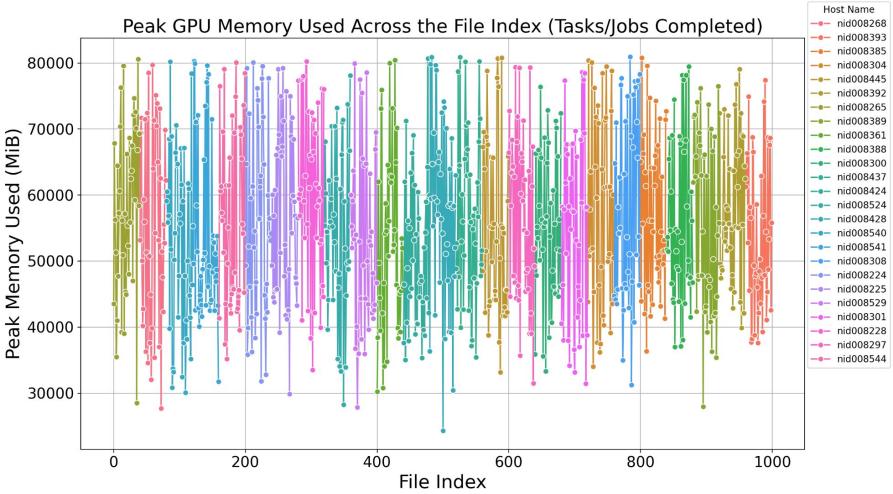


Post-hackathon:

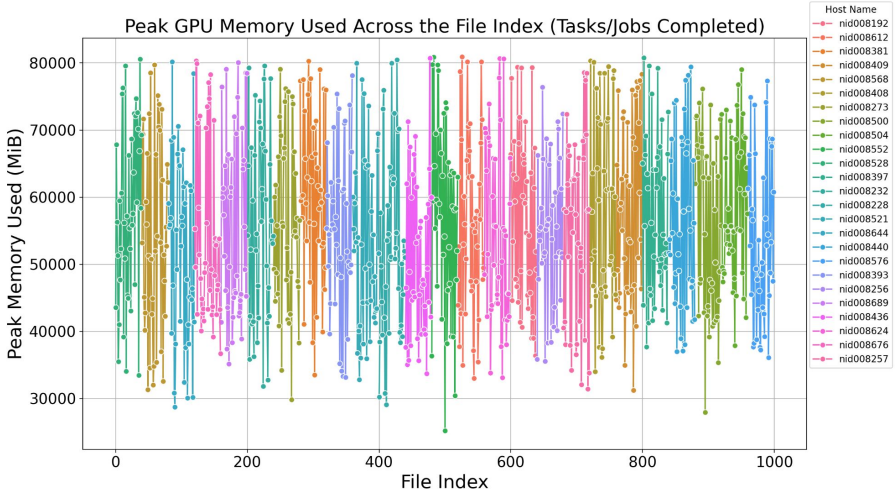
~6% reduction in median elapsed times, though plot shows a wider spread of task times

Peak Memory Used [MiB]

Develop



post-hackathon2024



Conclusion:

Comparison between the "develop" and "post-hackathon2024" branches plots:

Simulation Speed up: ~6% reduction in median elapsed time (24.97 min to 23.58 min).

Increased Variability: Standard deviation increased from 2.32 min (Develop) to 5.00 min (post-hackathon2024), indicating more variability.

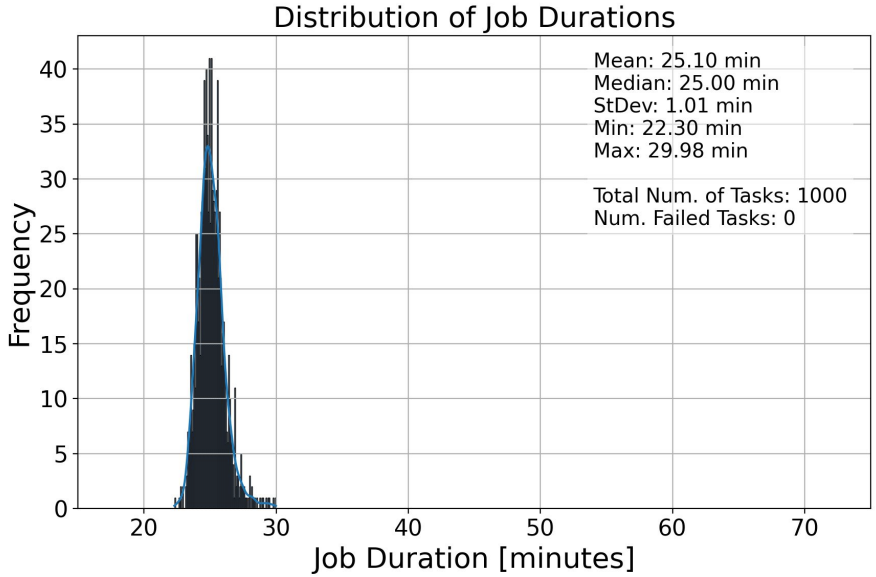
Task Distribution: Both versions peak around the same range, but the post-hackathon plot shows a wider spread of task times.

Overall Performance: Faster median times post-hackathon, though some tasks now take longer, reflecting greater performance variability.

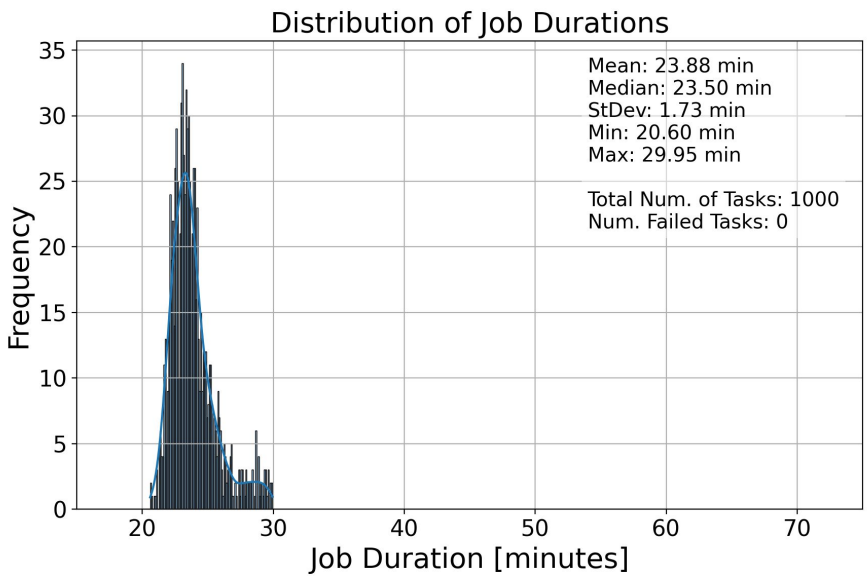
Back-up

Job Duration [minutes]

Develop

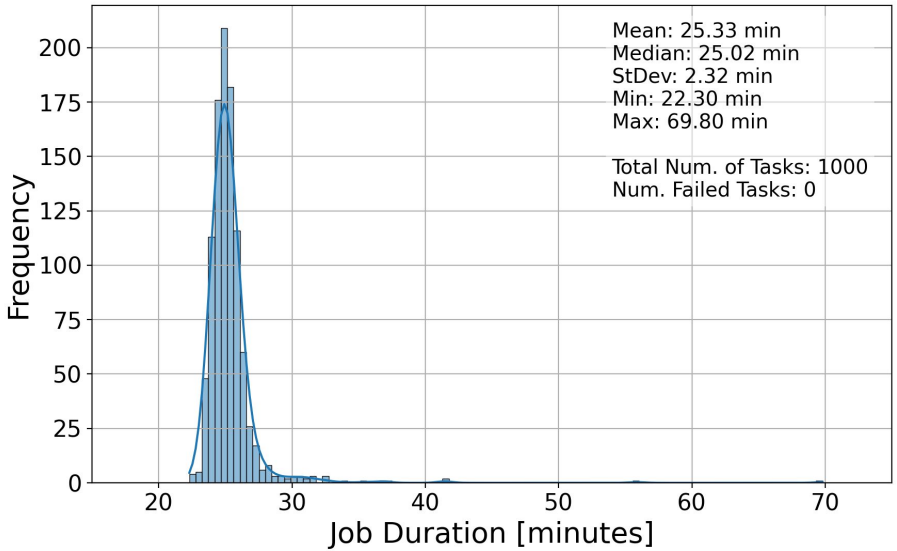


post-hackathon2024



Job Duration [minutes]

Develop



post-hackathon2024