DESI GPU Hackathon Progress and Goals

March 3-6, 2020

The DESI hackathon team

Progress and current status

- DESI is in Python so this is a brave new world. Whatever we learn will be useful both to DESI and to the rest of NERSC's many Python users!
- Major restructuring of DESI's specter code to be more GPU friendly
- CPU and GPU version which currently get the same (wrong) answer. Can test for correctness with --test flag.
- CPU profiling with PySpy
- Partially GPU-ized application using Numba CUDA Kernels and CuPy
- GPU profiling using nvprof, nsight systems
- Major bottleneck appears to be CuPy eigh function
- GPU version currently runs in ~5 mins on 1 V100 (and ⅛ skylake)
- Time to beat is ~2 mins on a Haswell

Goals for this hackathon (can be done in parallel)

- Optimize overall structure of code to fully occupy GPU. Use CUDA/CuPy streams instead of computing bundles serially.
- Get rid of unnecessary HtD and DtH transfer. May need to do all memory management manually.
- Overlap data transfer (like at the end of ex2d_patch) and compute. Use pinned memory.
- GPU-ize code in ex2d (still largely on CPU).
- Fix bookkeeping issues to get the right answers.
- Goal: be as fast as CPU version. Stretch goal: be faster than CPU version!
- More info here: https://github.com/sbailey/gpu_specter/tree/hackathon

Day 1 Summary and Day 2 Goals

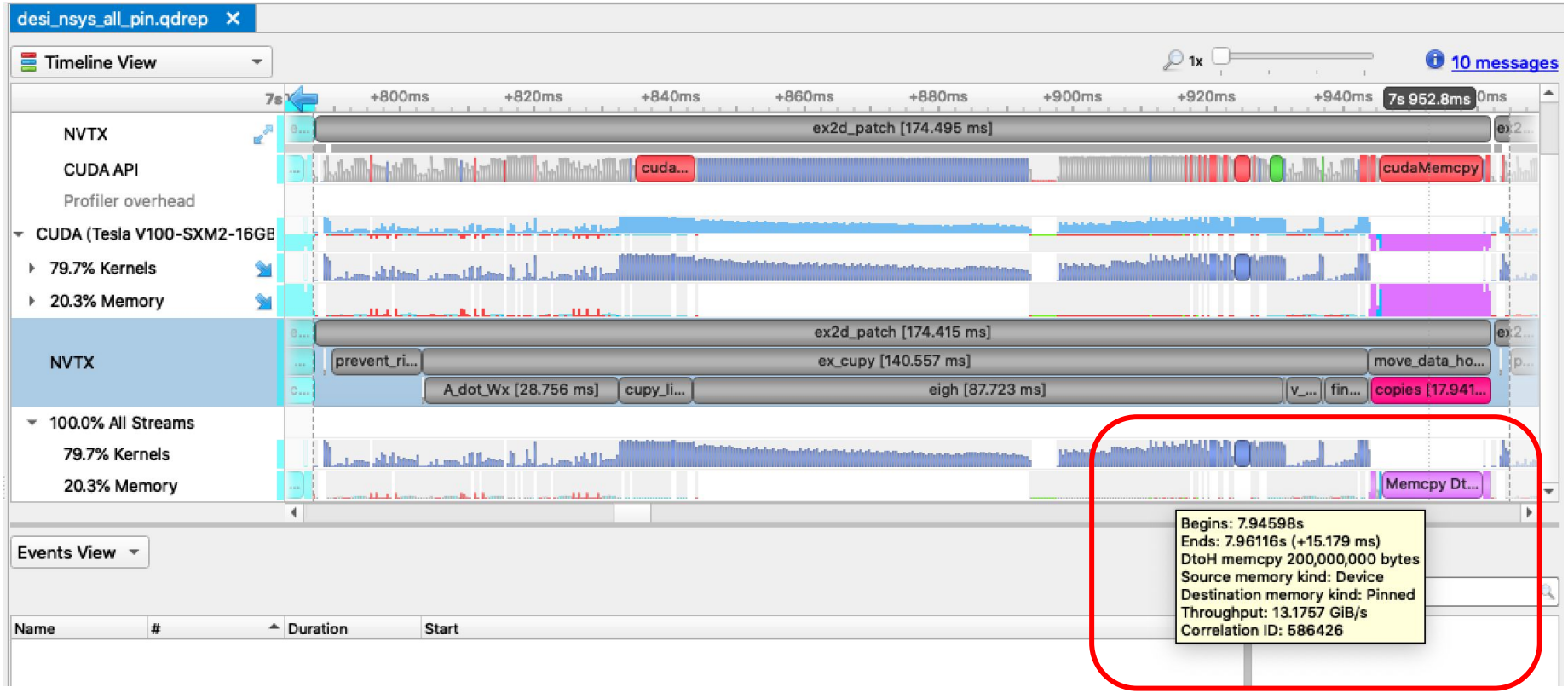
Day 1:

- Tried pinned memory, working but at the cost of bookkeeping/correctness
- Pinned memory a challenge because not all ccd patches are the same size
- Stephen working on correctness (gpu progress + correctness seem to be opposing forces, unfortunately)

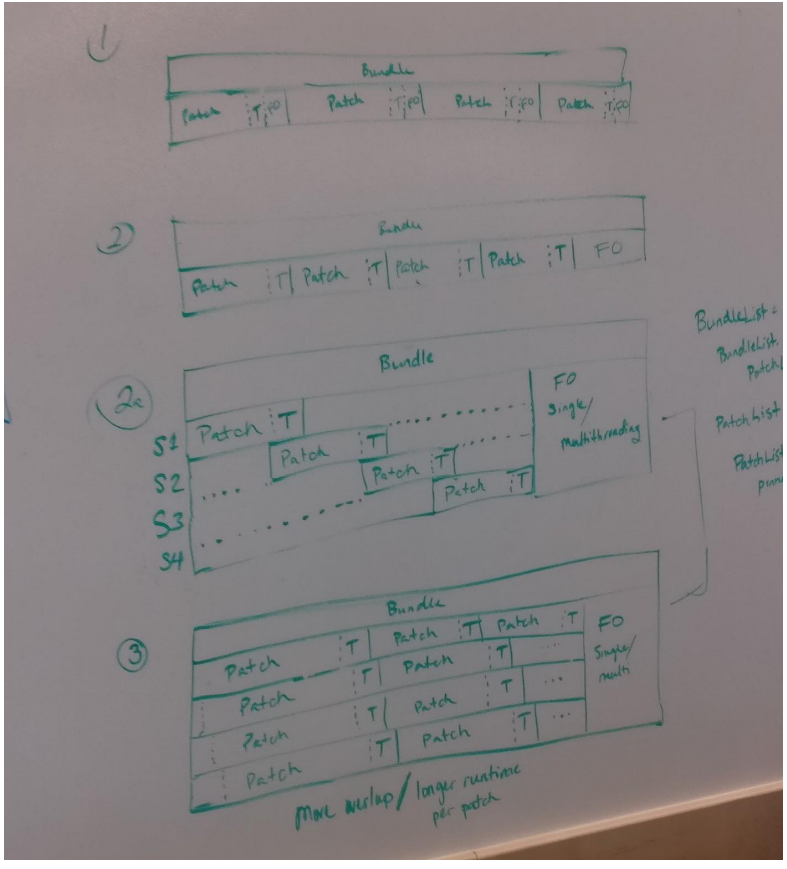
Day 2 goals:

- Try using streams + pinned memory to get some parallelism/async operations
- Optimize Numba CUDA Kernels
- Continue to work on correctness

Pinned memory!



Some Ideas (courtesy Matt Nicely)



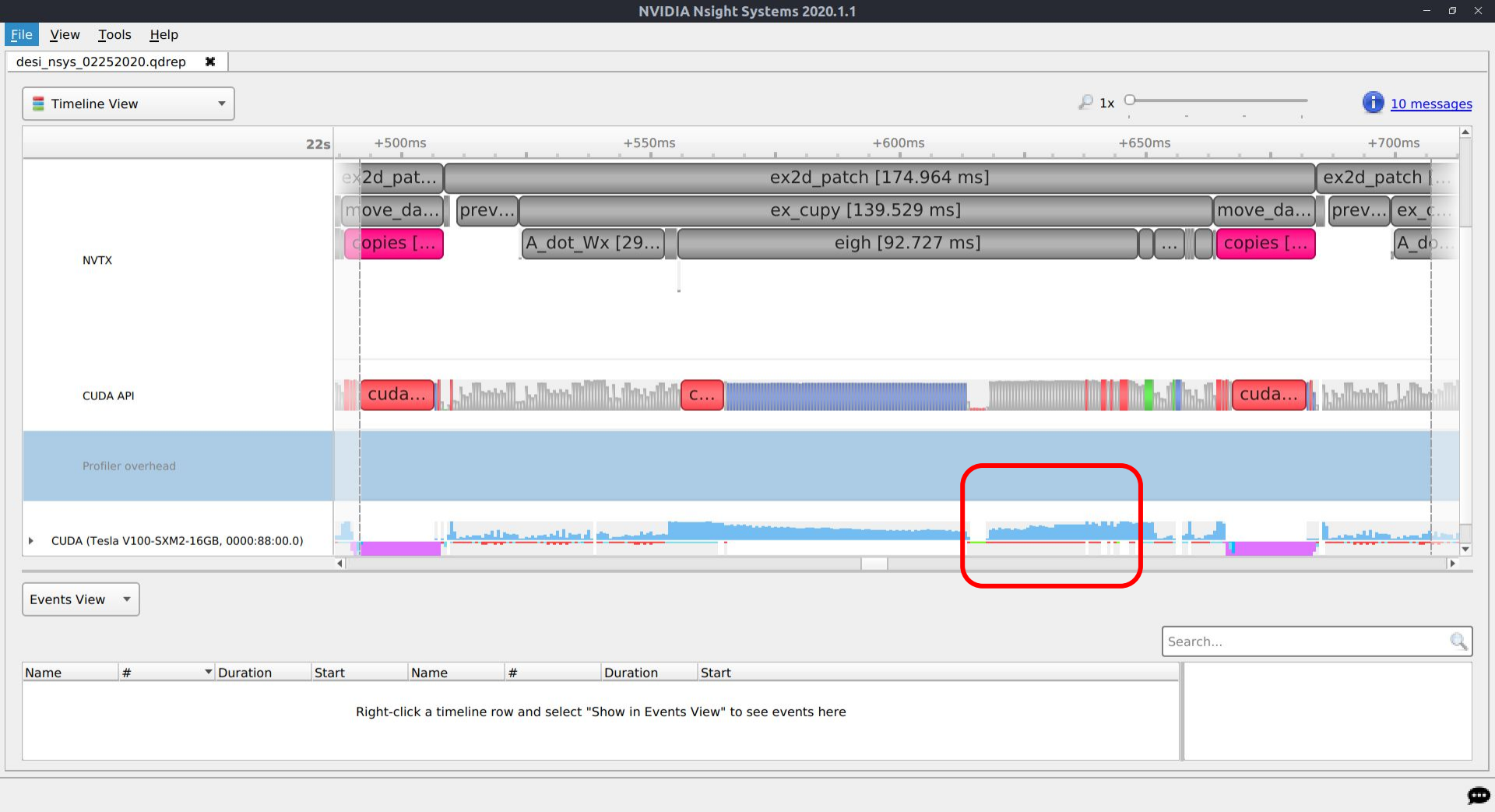
Day 2 Summary and Day 3 Goals

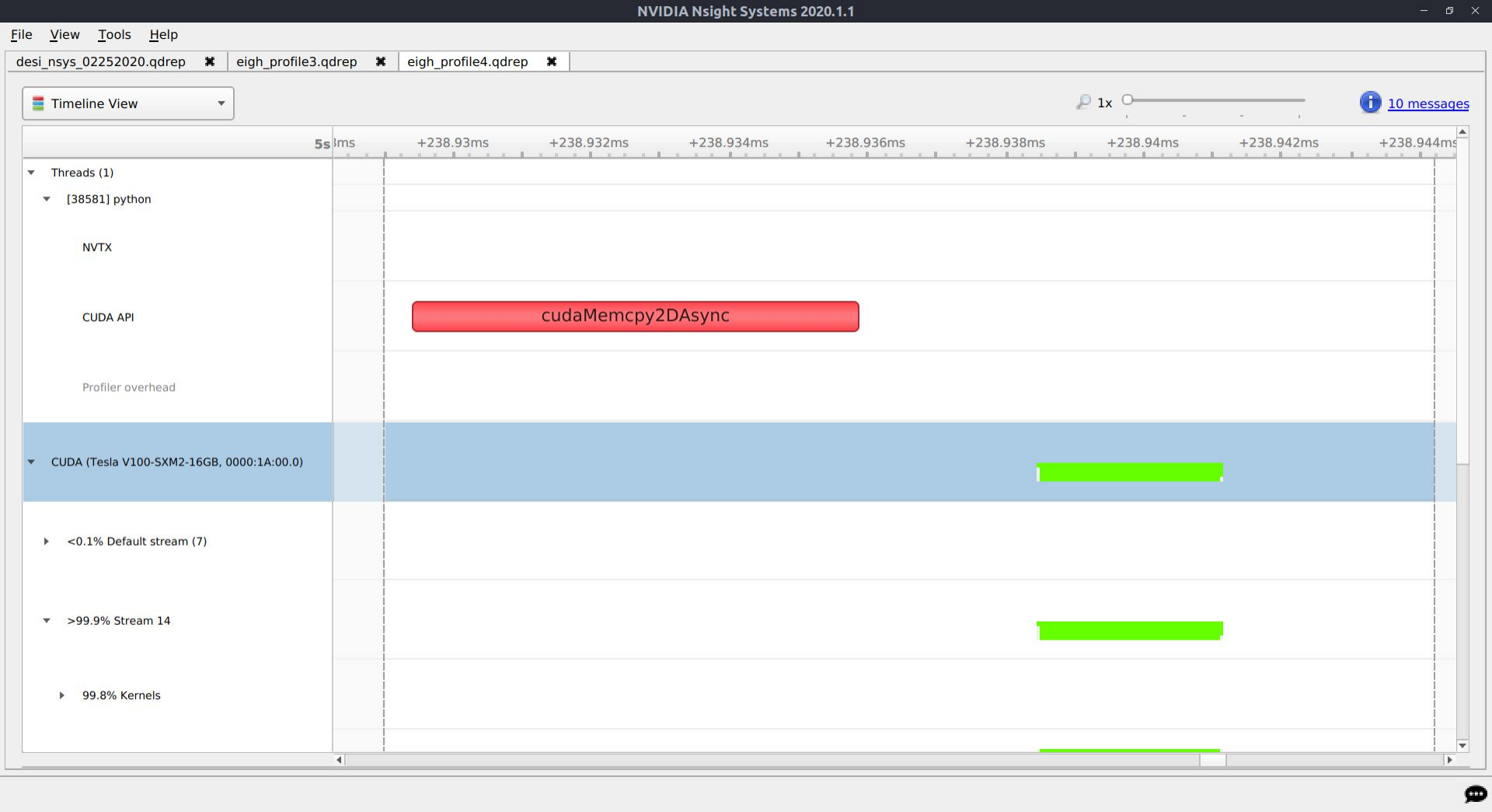
Day 2:

- Explored in detail pinned vs page memory in CuPy eigh (our bottleneck), Mark submitted github issue: https://github.com/cupy/cupy/issues/3154
- Tried but gave up on async memory transfers and compute, this is a future goal
- Made weak scaling measurements with 1 rank 1 bundle, 2 ranks 2 bundles, etc. → ok weak scaling until we OOM at 7 ranks

Day 3 goals:

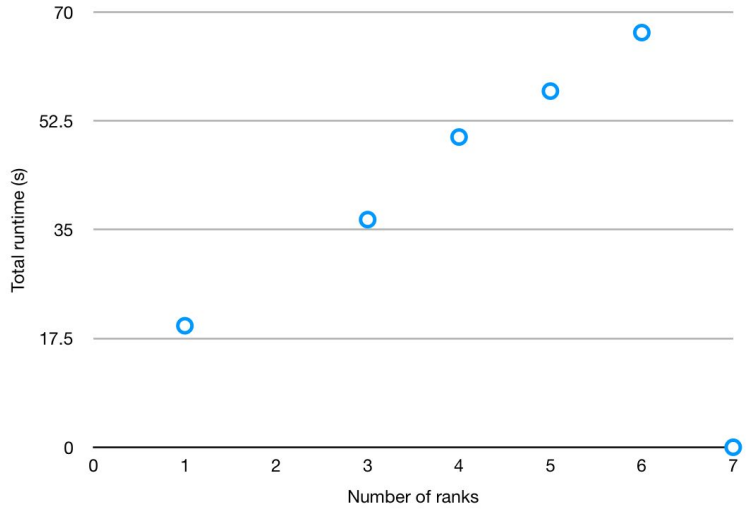
- Try out MPS while using MPI


NVIDIA Nsight Systems 2020.1.1
File
View
Tools
Help
desi_nsys_02252020.qdrep
Timeline View
1x
10 messages
22s
+500ms
+550ms
+600ms
+650ms
+700ms
ex2d_pat...
ex2d_patch [174.964 ms]
ex2d_patch [...
move_da...
prev...
ex_cupy [139.529 ms]
move_da...
prev...
ex_c...
copies [...
A_dot_Wx [29...
eigh [92.727 ms]
...
copies [...
A_do...
NVTX
cuda...
c...
cuda...
CUDA API
Profiler overhead
CUDA (Tesla V100-SXM2-16GB, 0000:88:00.0)
Events View
Search...
Name
#
Duration
Start
Name
#
Duration
Start
Right-click a timeline row and select "Show in Events View" to see events here


NVIDIA Nsight Systems 2020.1.1
File
View
Tools
Help
desi_nsys_02252020.qdrep
eigh_profile3.qdrep
eigh_profile4.qdrep
Timeline View
1x
10 messages
5s
+238.93ms
+238.932ms
+238.934ms
+238.936ms
+238.938ms
+238.94ms
+238.942ms
+238.944ms
Threads (1)
[38581] python
NVTX
CUDA API
cudaMemcpy2DAsync
Profiler overhead
CUDA (Tesla V100-SXM2-16GB, 0000:1A:00.0)
<0.1% Default stream (7)
>99.9% Stream 14
99.8% Kernels

MPI weak scaling → not bad

7 bundles, 7 ranks	OOM
6 bundles, 6 ranks	`1m 6s 687ms`
5 bundles, 5 ranks	`57s 278ms`
4 bundles, 4 ranks	`49s 890ms`
3 bundles, 3 ranks	`36s 610ms`
2 bundles, 2 ranks	`Segfault, ??`
1 bundle, 1 rank	`19s 541ms`



Day 3 Summary and Day 4 Goals

Day 3:

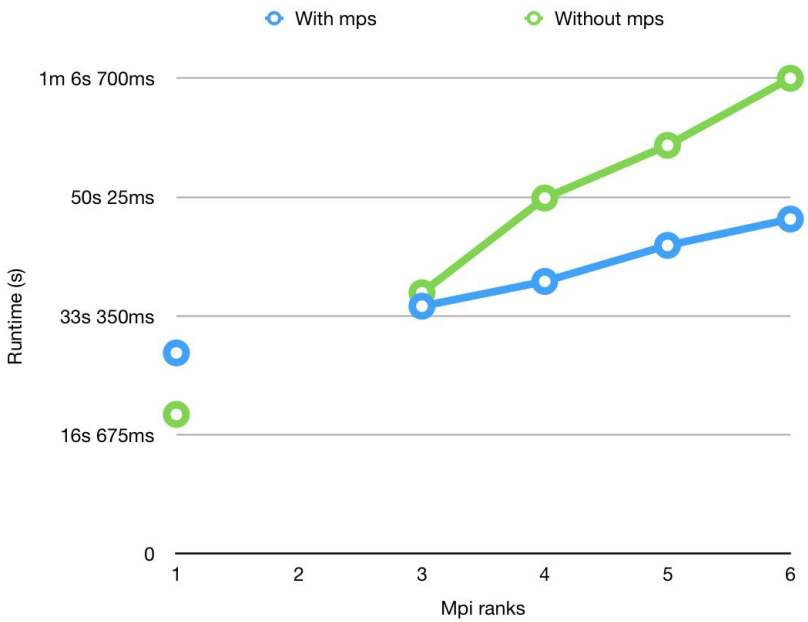
- Tested manual memory management in CuPy, Mark submitted an issue (https://github.com/cupy/cupy/issues/3154)
- Tested mps, in some cases made our code 1.75x faster
- Measured strong and weak scaling
- Used multiple GPUs for the first time (up to 6)
- Tested different patch sizes, runtime relatively insensitive to patch size
  - This is surprising to us → patch size mattered a lot on the cpu
  - This could also be a bug, will investigate further

Day 4 goals:

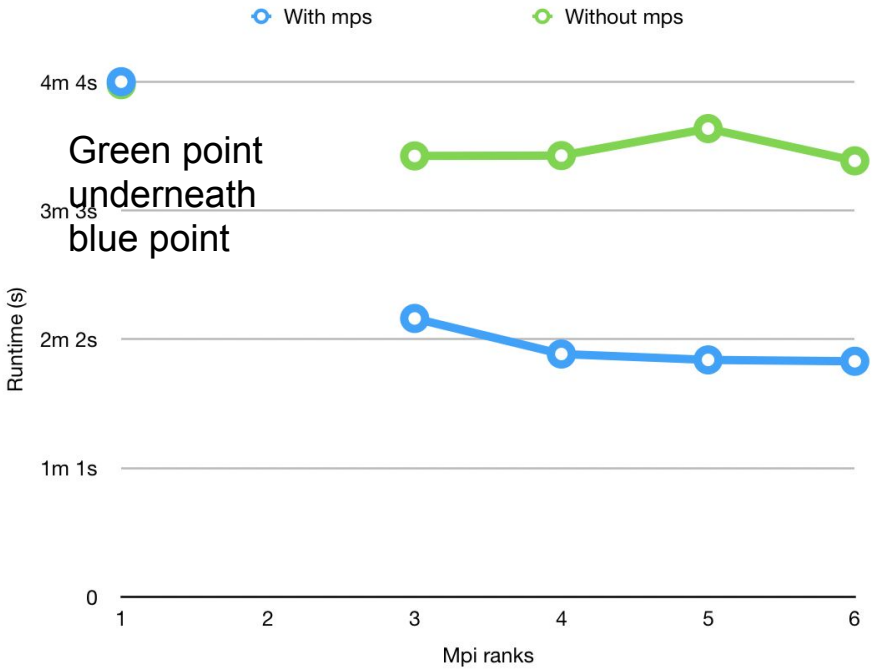
- Write down what we learned (for ourselves and our users)
- Make plans for future code development

Strong and weak scaling tests with single gpu

Weak scaling (bundles proportional to number of ranks)

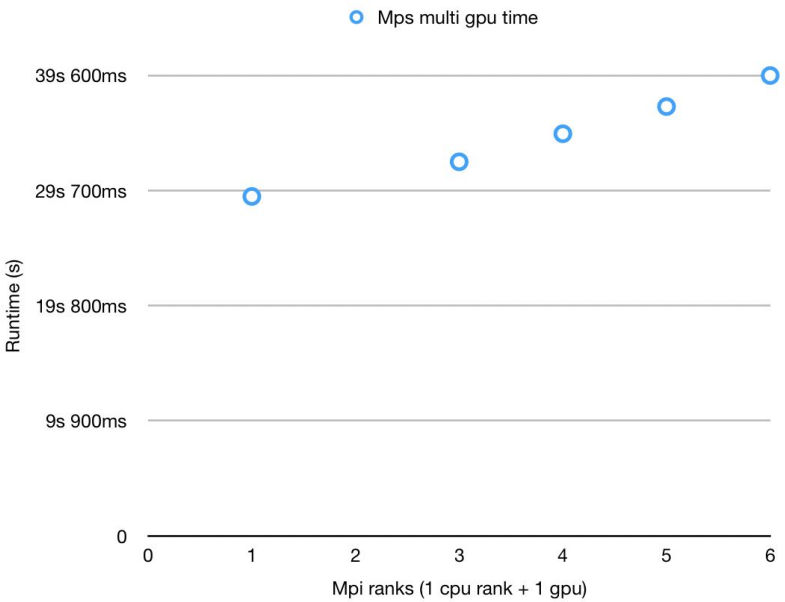


Strong scaling (20 bundles total)

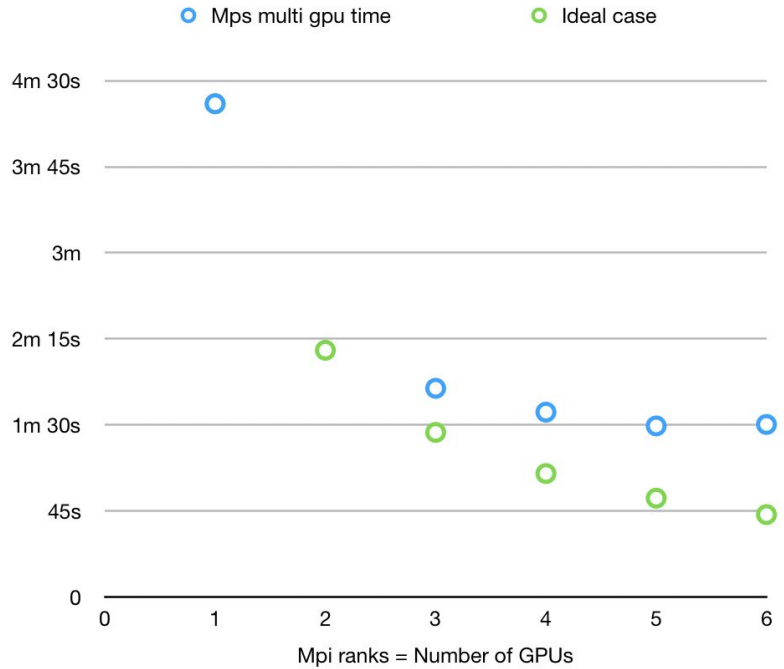


Strong and weak scaling tests on multiple gpus

Weak scaling (bundles proportional to number of ranks)



Strong scaling (20 bundles total)



Hackathon Summary

- Pinned memory is good
- Multiple MPI ranks per GPU was good for this app because it allows us to interleave CPU and GPU work
- MPI + MPS is better because it allows simultaneous GPU work
- Didn’t explore streams because we believe that MPS will bring more benefits
- Filed two CuPy issues, NVIDIA developers looking into them
  - https://github.com/cupy/cupy/issues/1299
  - https://github.com/cupy/cupy/issues/3154
- Speedup from:
  - Pinned memory at end of each patch
  - MPI + MPS
  - Increasing hardware (adding more gpus)
- BUT! We broke the app in the process

Future work

- Investigate overhead
  - Why is weak scaling not flat?
  - Why is overhead growing in strong scaling?
  - Will need more markers/profiling data to learn more
- Address memory use so we can scale beyond 6 ranks per node
- Correctness-- get answers that are actually useful
- Reducing technical debt (fixing bookkeeping, re-adding parts of the app we removed)
- Move more work to GPU (still some on CPU that is probably well-suited to GPU)
- More adjustments to overall structure
  - Accumulate data and transfer memory once at end of each bundle
  - Explore 3D arrays in cp.linalg.solve and cp.eigh (and necessary restructuring)