Accelerating MURaM on GPUs using OpenACC

2019 Multicore9 Workshop

UDEL: Eric Wright, Sunita Chandrasekaran
NCAR: Shiquan Su, Cena Miller, Supreeth Suresh, Matthias Rempel, Rich Loft
Max Planck Institute for Solar System Research: Damien Przybylski

Contact: efwright@udel.edu

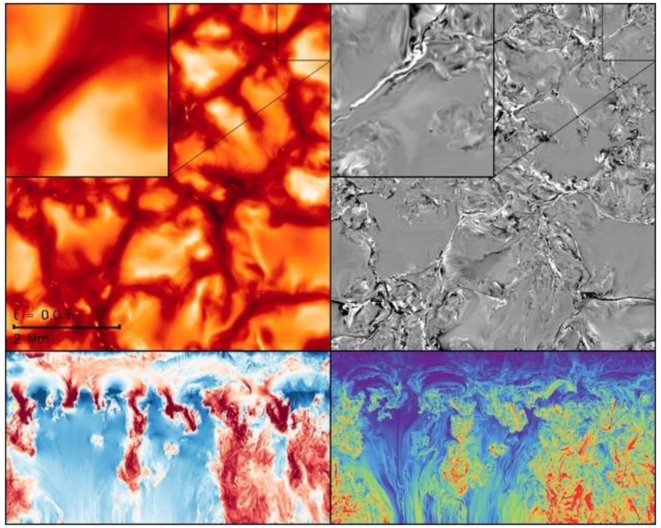
September 25th & 26th, 2019, National Center for Atmospheric Research (NCAR) Mesa Lab in Boulder, Colorado

Outline

- MURaM Introduction
- OpenACC Introduction
- Development Tools
- Development Roadblocks
- Results

MURaM (Max Planck University of Chicago Radiative MHD)

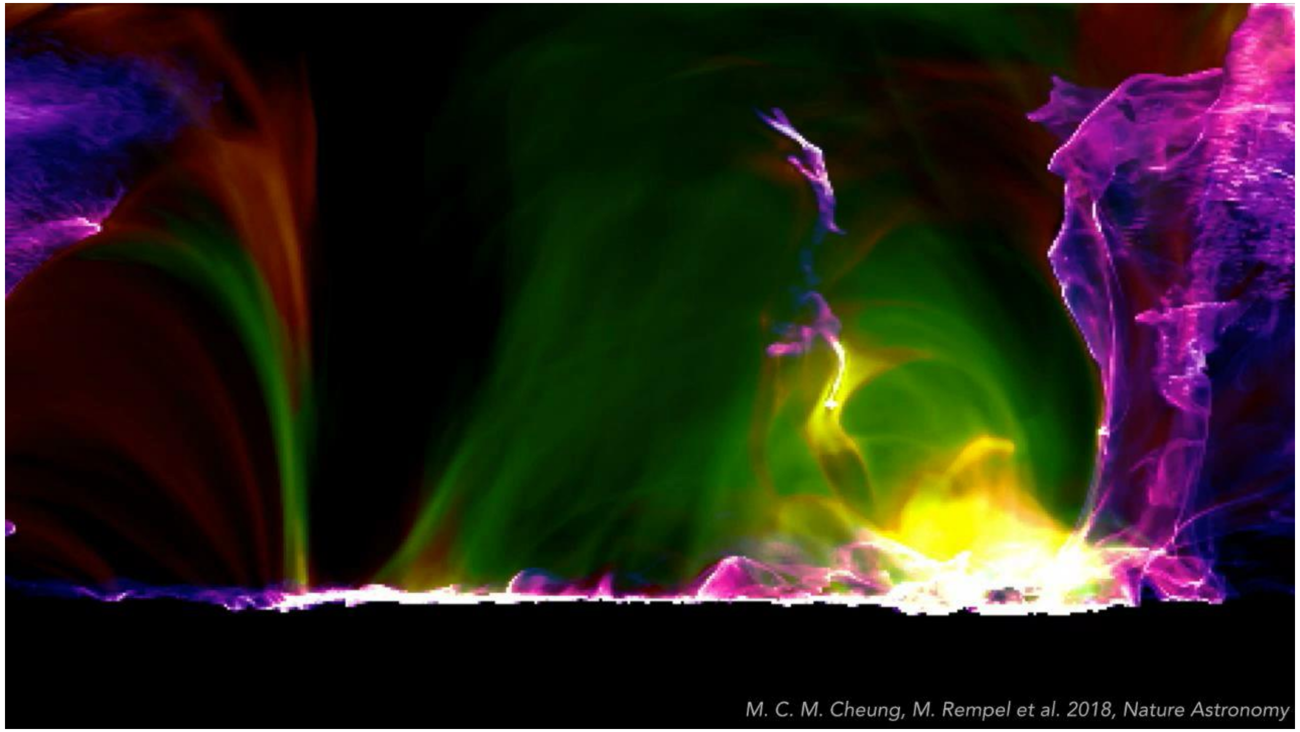
- The primary solar model used for simulations of the upper convection zone, photosphere and corona
- Jointly developed and used by HAO, the Max Planck Institute for Solar System Research (MPS) and the Lockheed Martin Solar and Astrophysics Laboratory (LMSAL)
- The Daniel K. Inouye Solar Telescope (DKIST), a ~$300M NSF investment, is expected to advance the resolution of ground based observational solar physics by an order of magnitude
- Requires at least 10-100x increase in computing power compared to current baseline



MURaM simulation of solar granulation

Physics of the MURaM Code

- **Science target**
  - Realistic simulations of the coupled solar atmosphere
  - Detailed comparison with available observations through forward modeling of synthetic observables
- **Implemented Physics**
  - Single fluid MHD
  - 3D radiative transfer, multi-band + scattering
  - Partial ionization equation of state
  - Heat conduction
  - Optically thin radiative loss
  - Ambipolar diffusion
- **Under development**
  - Non-equilibrium ionization of hydrogen



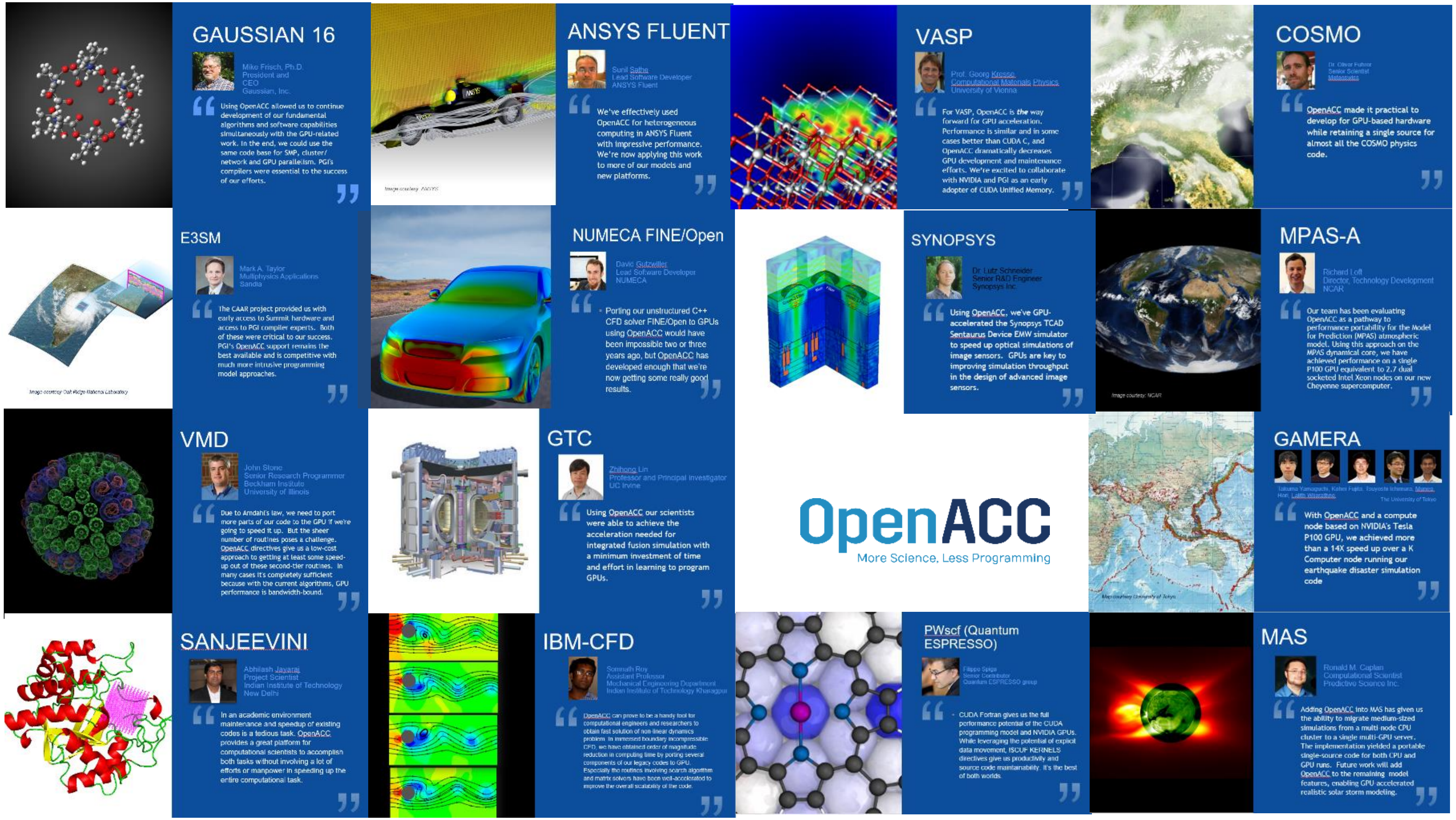
Comprehensive model of entire life cycle of a solar prominence (Cheung et al 2018)

Why OpenACC?

3 Ways to program CPU-GPU Architectures

Applications		
Libraries	OpenACC, OpenMP Directives	Programming Languages (CUDA, OpenCL)
"Drop-in" Acceleration	Incremental, Enhanced Portability	Maximum Flexibility

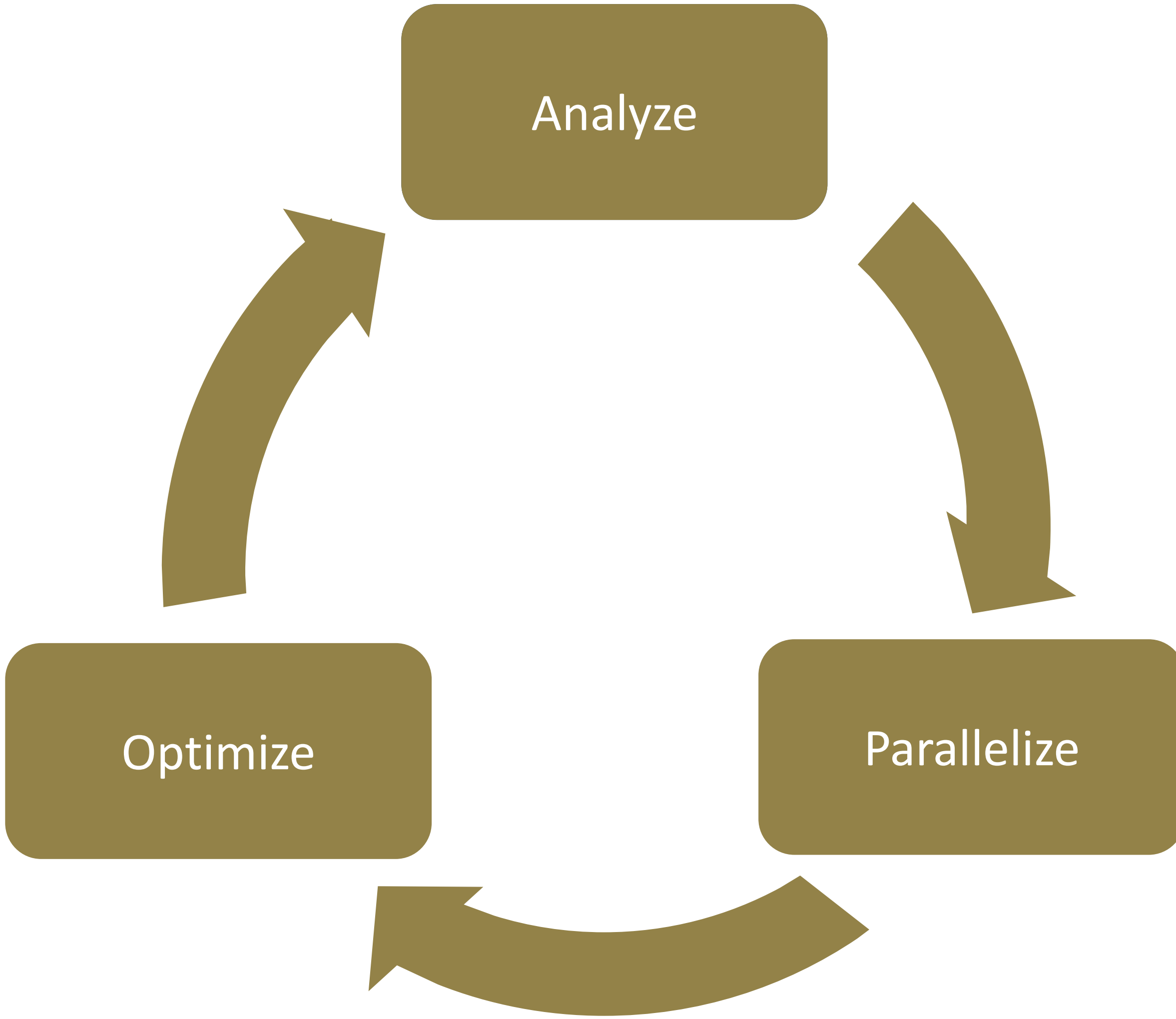
University of Delaware



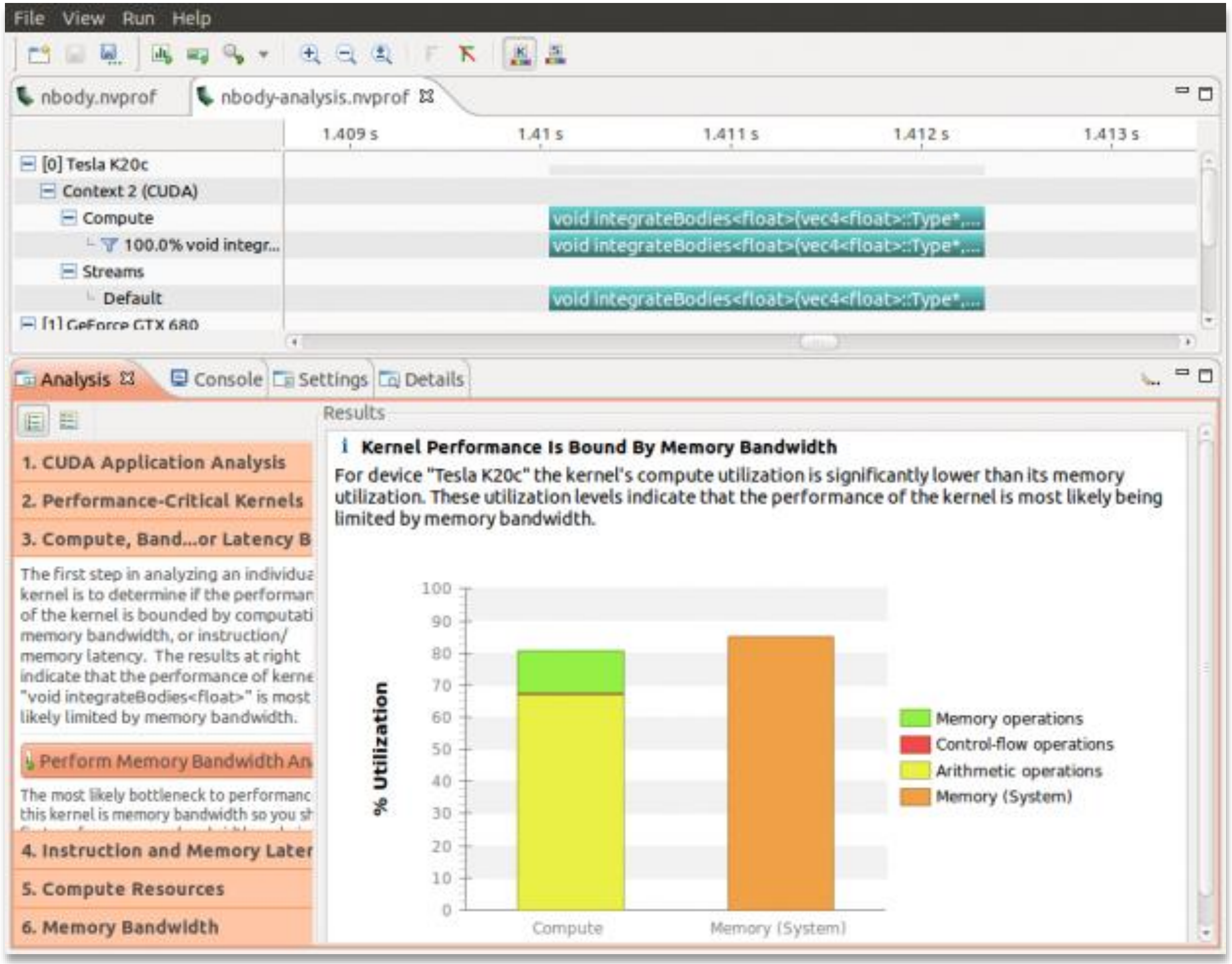
GPU Development and Tools

Development Cycle

Name	Routine Summary:	Broadwell (v4) core: (sec)
TVD Diffusion	Update diffusion scheme - using TVD slope + flux limiting.	7.36812
Magnetohydrodynamics	Calculate right hand side of MHD equations.	6.26662
Radiation Transport	Calculate radiation field and determine heating term (Qtot) required in MHD.	5.55416
Equation of State	Calculate primitive variables from conservative variables. Interpolate the equation of state tables.	2.26398
Time Integration	Performs one time integration.	1.47858
DivB Cleaner	Clean any errors due to non-zero div(B).	0.279718
Boundary Conditions	Update vertical boundary conditions.	0.0855162
Grid Exchange	Grid exchanges (only those in Solver)	0.0667914
Alfven Speed Limiter	Limit Maximum Alfven Velocity	0.0394724
Synchronize timestep	Pick minimum of the radiation, MHD and diffusive timesteps.	4.48E-05



NVPROF: NVIDIA GPU Profiler

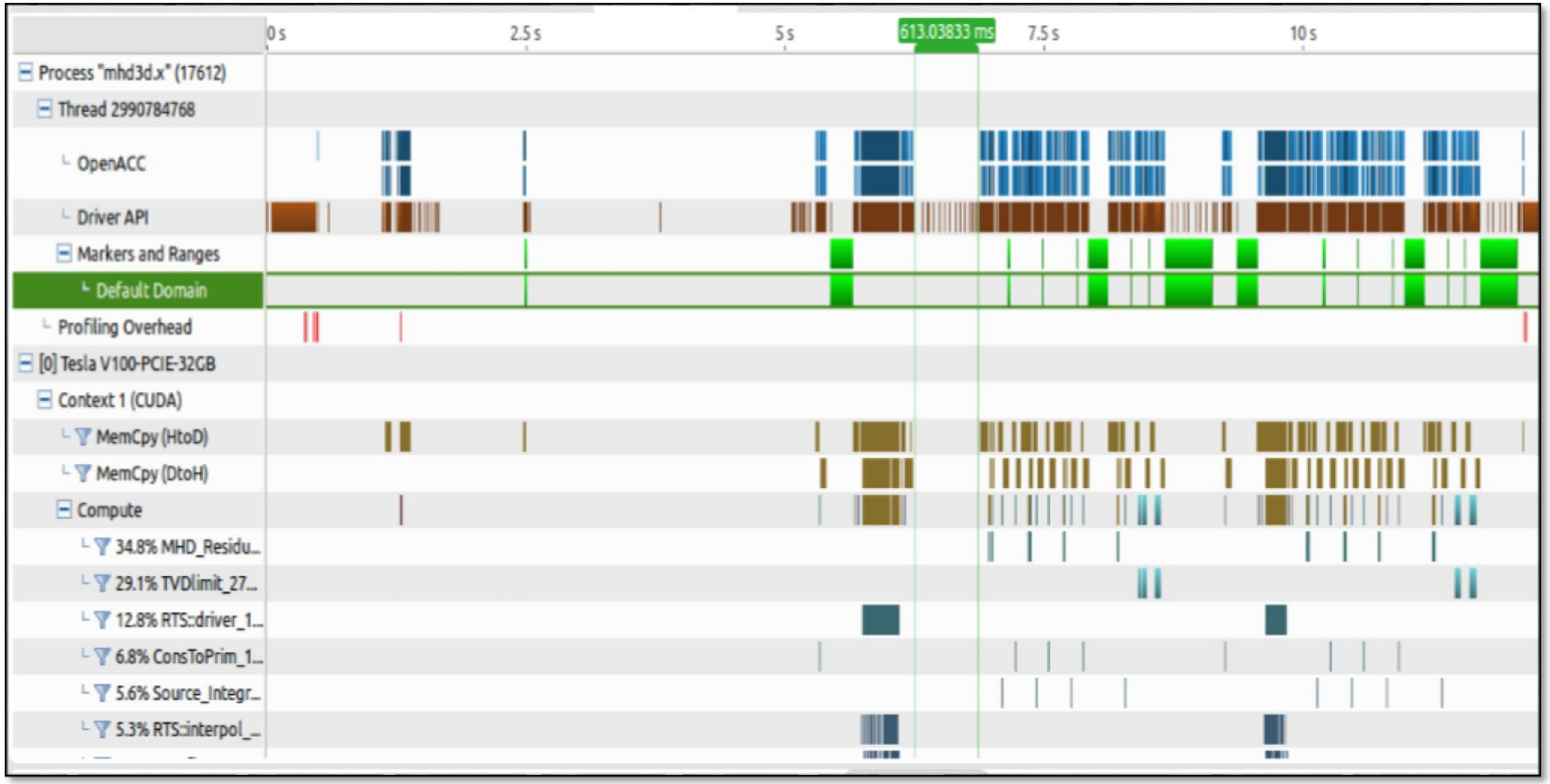


- Profilers give detailed information/feedback about code execution
- For this work, we used NVIDIA's GPU enabled profiler too: NVPROF

https://devblogs.nvidia.com/cuda-pro-tip-nvprof-your-handy-universal-gpu-profiler/

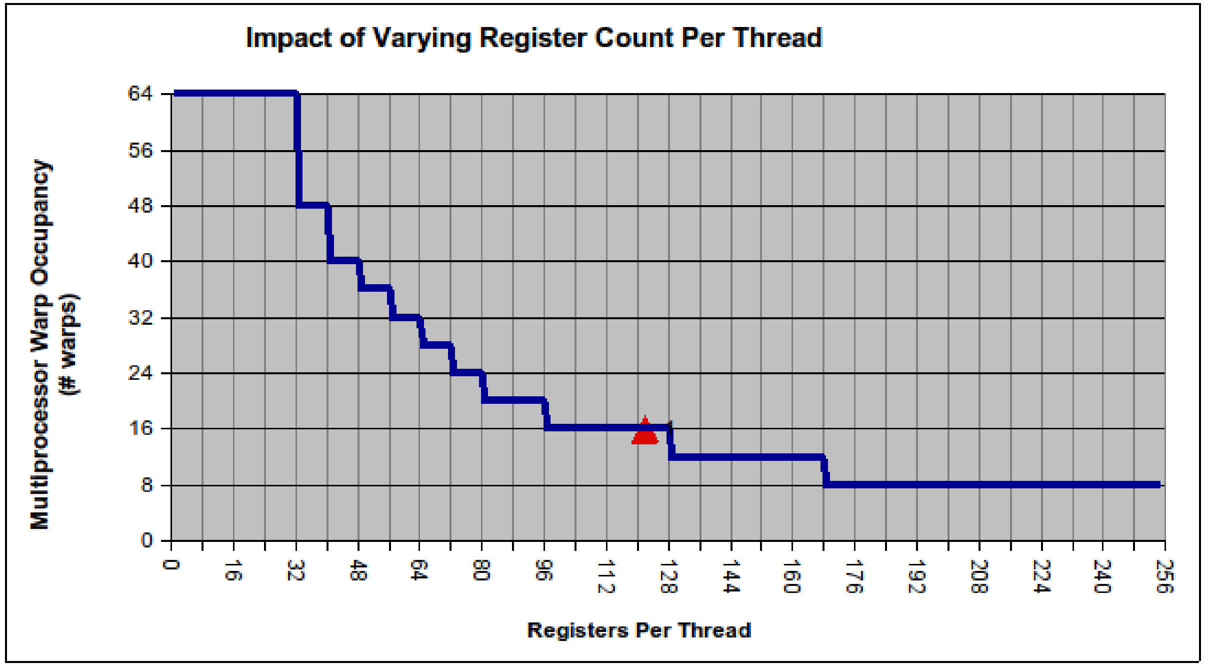
https://devblogs.nvidia.com/cuda-pro-tip-nvprof-your-handy-universal-gpu-profiler/

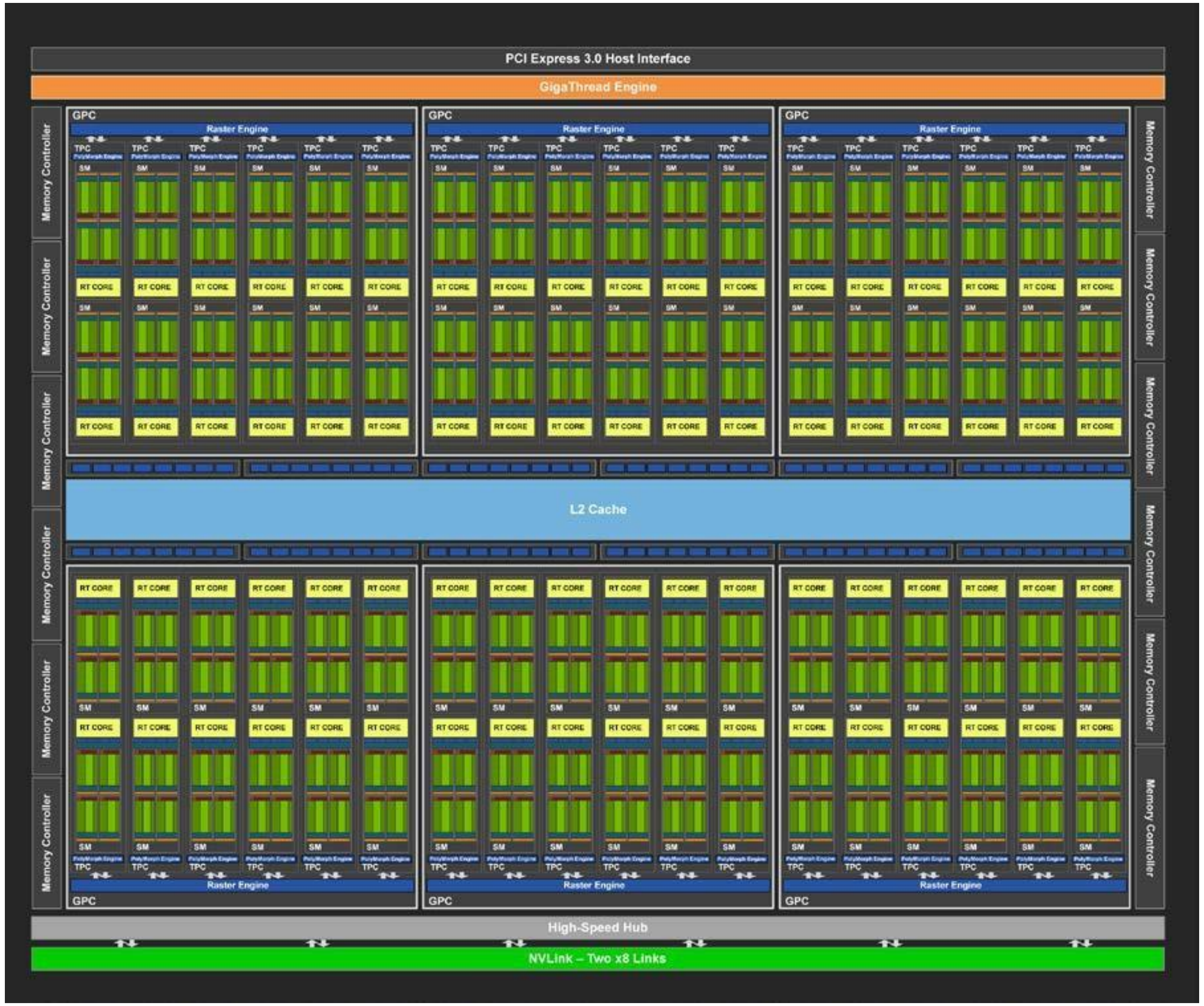
CUPTI (CUDA Profiling Tools Interface)



- Annotate code to give additional profiler feedback

CUDA Occupancy Calculator





https://devblogs.nvidia.com/nvidia-turing-architecture-in-depth/

PCAST (PGI Compiler Assisted Software Testing)

- Automated testing features for PGI compiler
- Able to do autocompare (sometimes) to make kernel debugging much easier
- In our case, we used API calls to do some checking manually, but allowed for easy code testing after

```
$ pgcc -ta=tesla:autocompare -o a.out example.c

$ PGI_COMPARE=summary,compare,abs=1 ./a.out
PCAST a1 comparison-label:0 Float
   idx: 0 FAIL ABS  act: 8.40187728e-01 exp: 1.00000000e+00 tol: 1.00000001e-01
   idx: 1 FAIL ABS  act: 3.94382924e-01 exp: 1.00000000e+00 tol: 1.00000001e-01
   idx: 2 FAIL ABS  act: 7.83099234e-01 exp: 1.00000000e+00 tol: 1.00000001e-01
   idx: 3 FAIL ABS  act: 7.98440039e-01 exp: 1.00000000e+00 tol: 1.00000001e-01
```

Roadblocks

CUDA Occupancy Report

240x160x160 Dataset

Kernel Name	Theoretical Occupancy	Achieved Occupancy
MHD	25%	24.9%
TVD	31%	31.2%
CONS	25%	24.9%
Source_Tcheck	25%	24.9%
Radiation Transport		
Driver	100%	10.2%
Interpol	56%	59.9%
Flux	100%	79%

RTS Data Dependency Along Rays

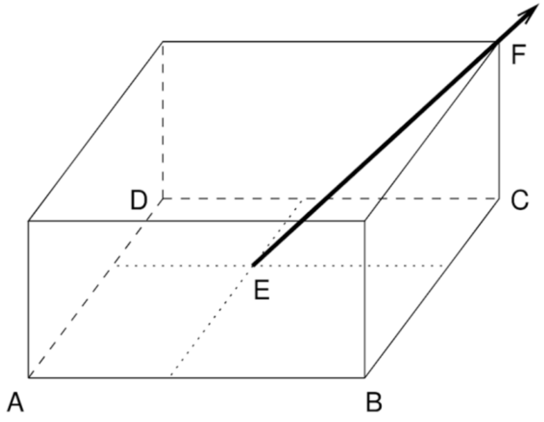


Figure 4.1: *The intensity at gridpoint F is obtained by solving the transfer equation along the short characteristic $\overline{EF}$. The intensity at the upwind point E is interpolated from the (already known) intensity values at the surrounding gridpoints, A to D.*

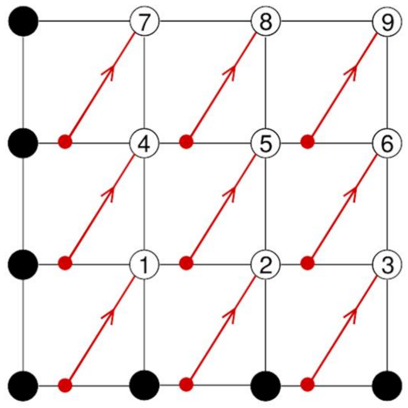


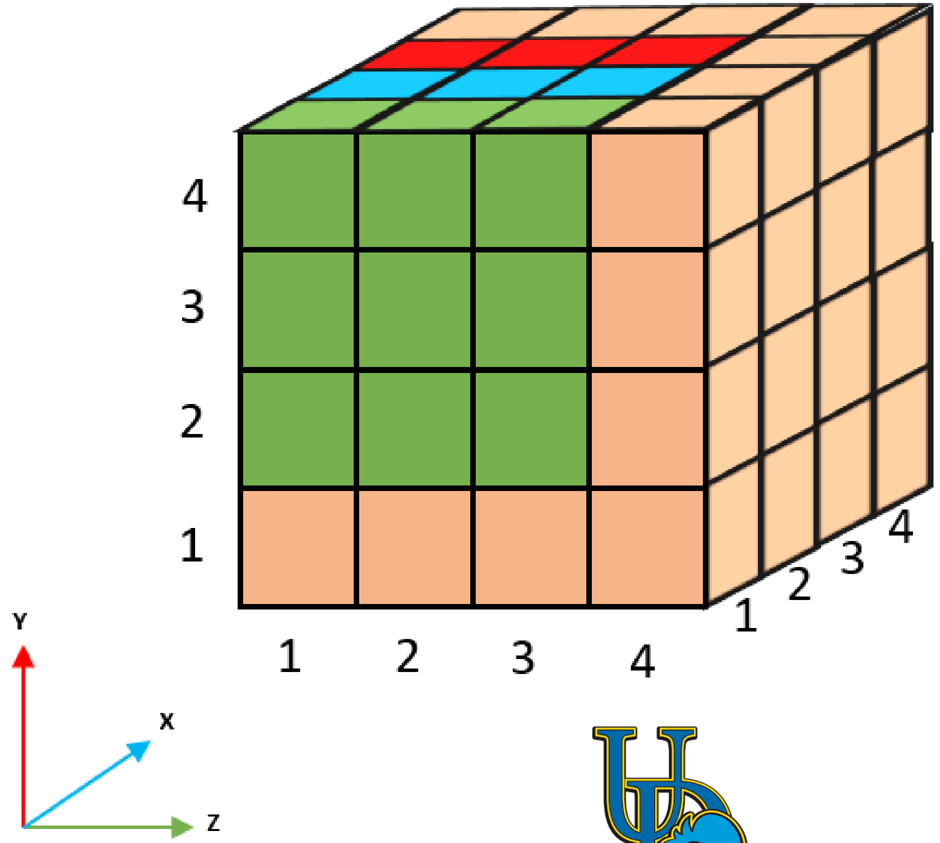
Figure 4.2: *The walking order of the Short Characteristics method in a 2D grid for a ray direction pointing into the upper right quadrant. Black circles represent gridpoints on the upwind boundaries, where the intensity values are assumed to be known.*

- Data dependency is along a plane for each octant,angle combo.
- Depends on resolution ratio, not known until run-time.
- Number of rays per plane can vary.

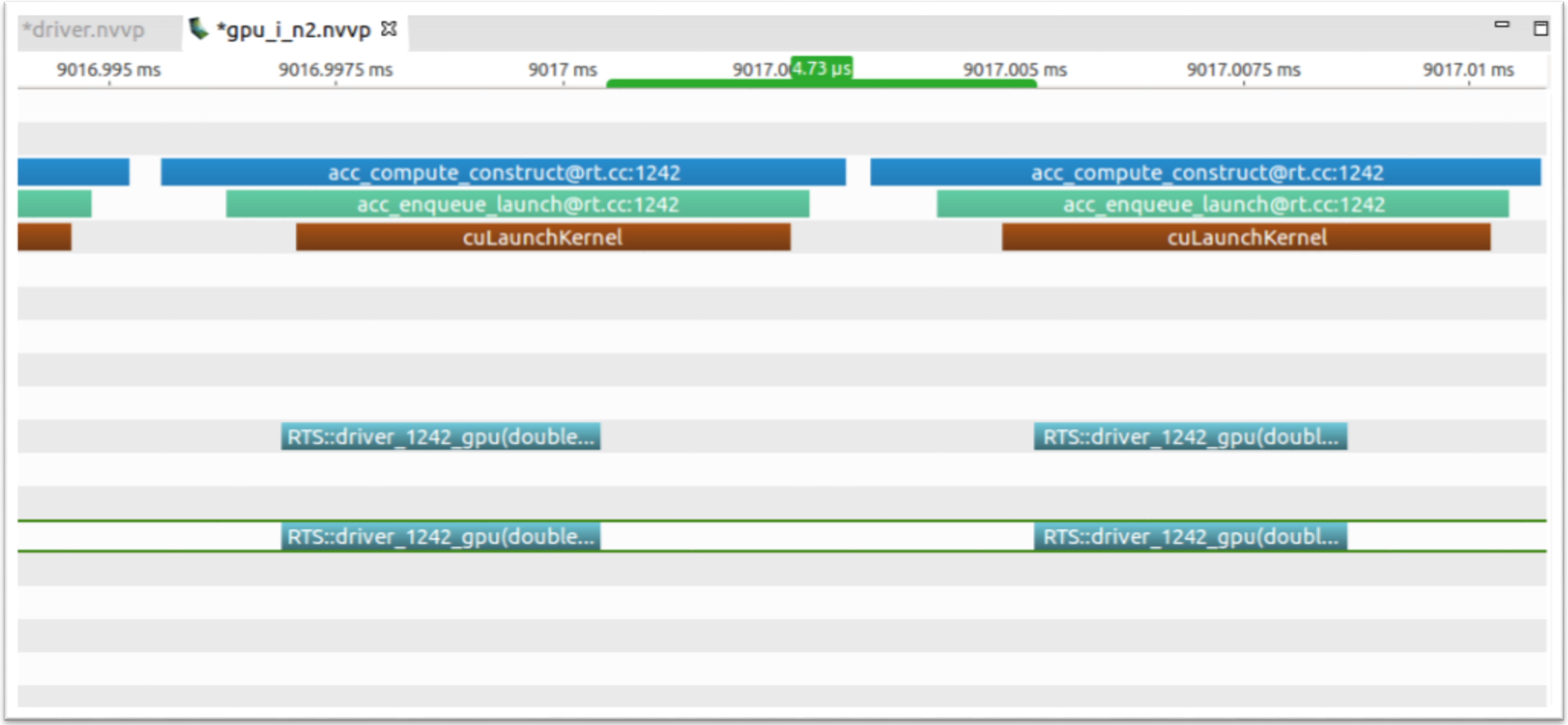
Vögler, Alexander, et al. "Simulations of magneto-convection in the solar photosphere-Equations, methods, and results of the MURaM code." Astronomy & Astrophysics 429.1 (2005): 335-351.

Solving RTS Data Dependency

- We can deconstruct the 3D grid into a series of 2D slices
- The direction of the slices is dependent on the X,Y,Z direction of the ray
- Parallelize within the slice, but run the slices themselves serially in predetermined order



Profiler driven optimizations



Results

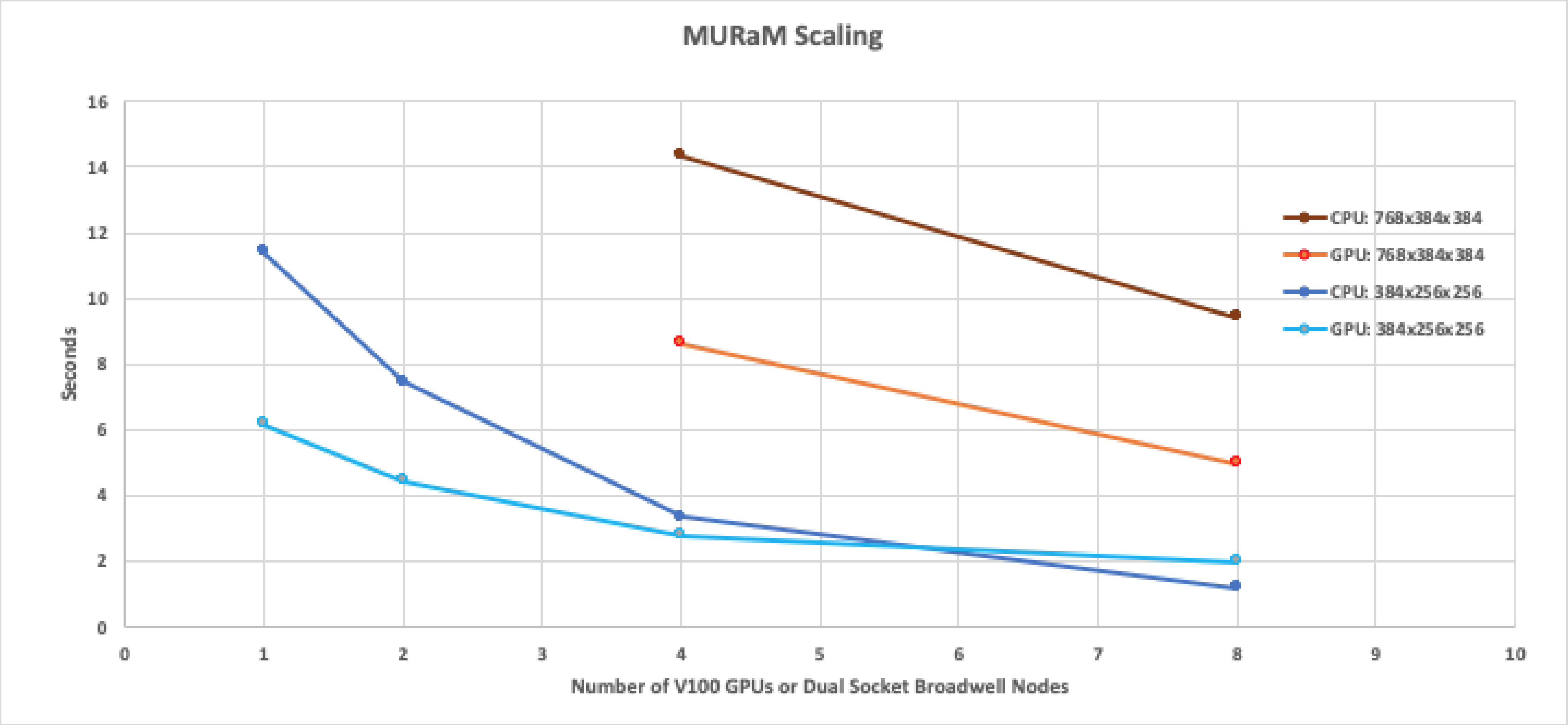
Experimental Setup

- NCAR Casper system
  - 28 Supermicro nodes featuring Intel Skylake processors
  - 36 cores/node
  - 384GB memory/node
  - 4/8 NVIDIA V100 GPUs/node
  - PGI 19.4, CUDA 10

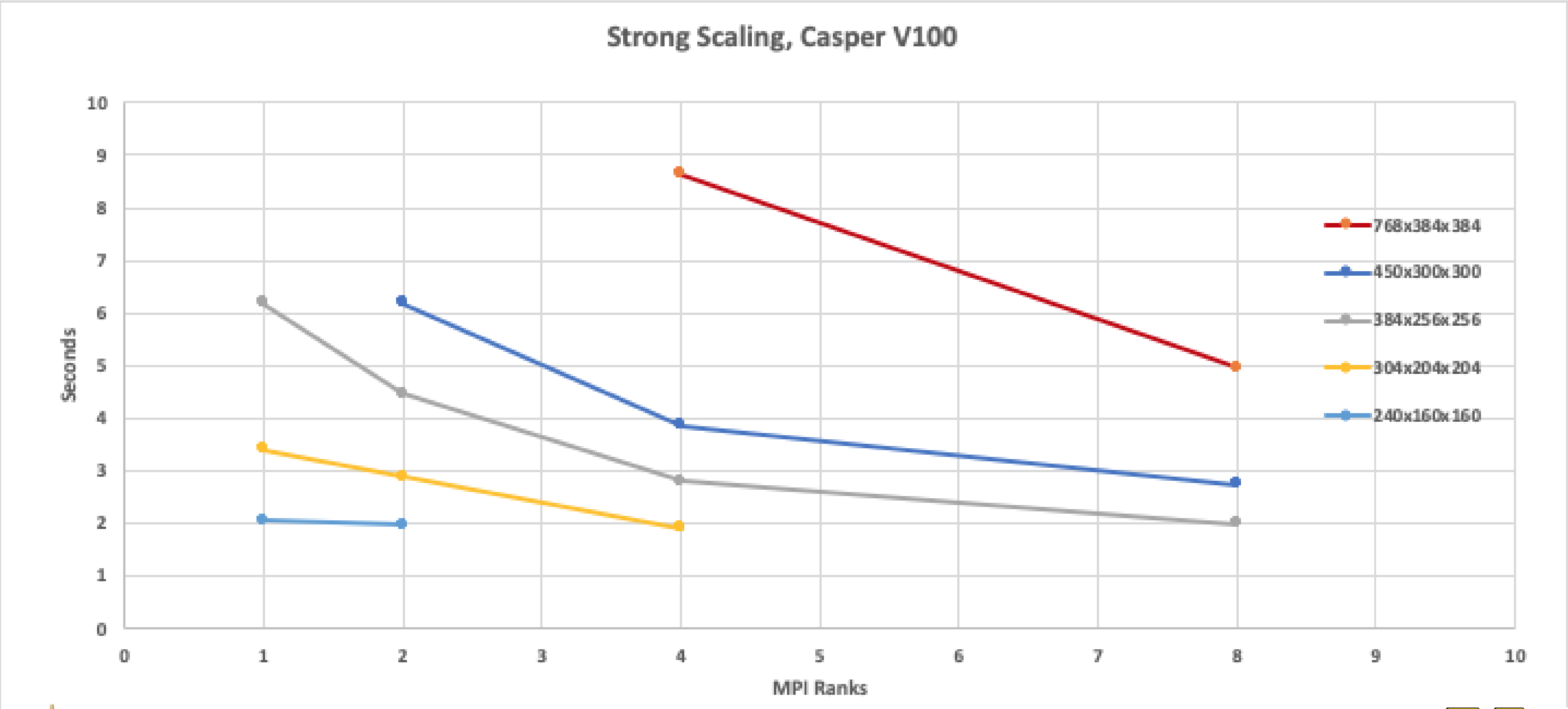
Results: CPU vs GPU

Routine	GPU time	CPU time	Speedup
RTS	0.361	0.230	0.637
MHD	0.108	0.160	1.48x
TVD	0.056	0.066	1.17x
EOS	0.031	0.071	2.29x
BND	0.004	0.007	1.75x
INT	0.050	0.071	1.42x
DST	0.163	0.031	0.19x
DIVB	0.076	0.029	0.38x
TOTAL	0.853	0.701	0.82x

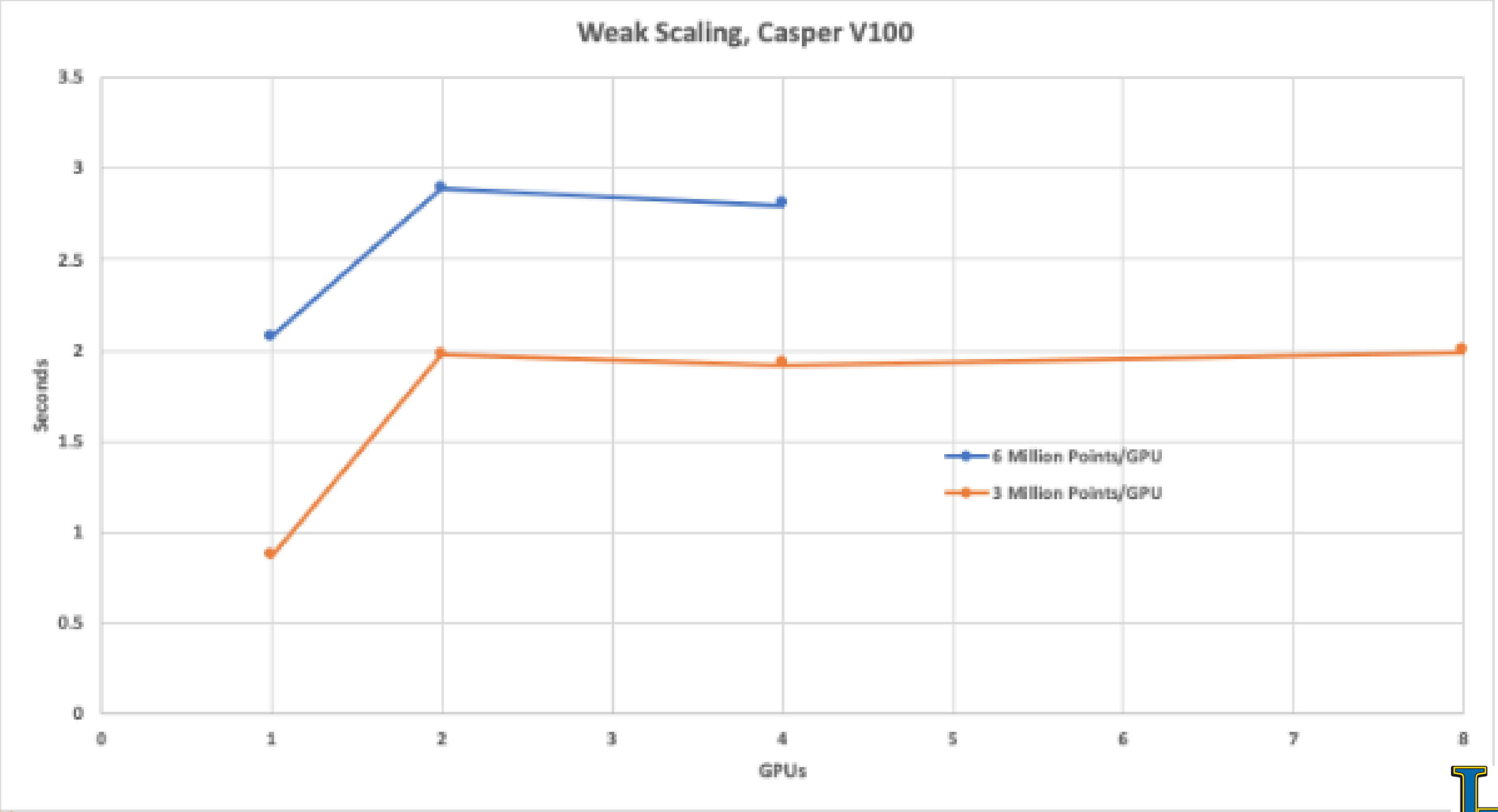
- Single NVIDIA V100 GPU
- Dual Socket Intel Skylake CPU (36 core)
- Measuring time taken for average timestep with no file I/O
- 192x128x128 sized dataset


MURaM Scaling
Seconds
Number of V100 GPUs or Dual Socket Broadwell Nodes
CPU: 768x384x384
GPU: 768x384x384
CPU: 384x256x256
GPU: 384x256x256
0
2
4
6
8
10
12
14
16
0
1
2
3
4
5
6
7
8
9
10

Strong Scaling



Weak Scaling



Summary

- MURaM
  - Single fluid MHD
  - 3D radiative transfer, multi-band + scattering
  - Partial ionization equation of state
  - Heat conduction
  - Optically thin radiative loss
  - Ambipolar diffusion
- Use OpenACC to port to GPU with directives
  - Incremental changes
  - Maintain single C++ source code
- Tools: NVPROF, CUPTI, CUDA Occupancy Calculator, PGI PCAST

Contact: efwright@udel.edu