GPU Enablement of MICM Chemistry Solver

Qina Tan (Colorado School of Mines)

Mentors: Jian Sun, John Dennis, Matthew Dawson

SIParCS Project 9

Presentation Date: August 1st , 2023

Background & Primary Project

Model-Independent Chemistry Module (MICM)

- Software package known as a chemistry solver, being developed in C++
- About 2000 lines of codes with 96% automatic testing coverage
- Computationally expensive part of an atmospheric model

Developed a GPU version of MICM via CUDA programming

GPU Architecture

Illustration of main difference between CPU and GPU:

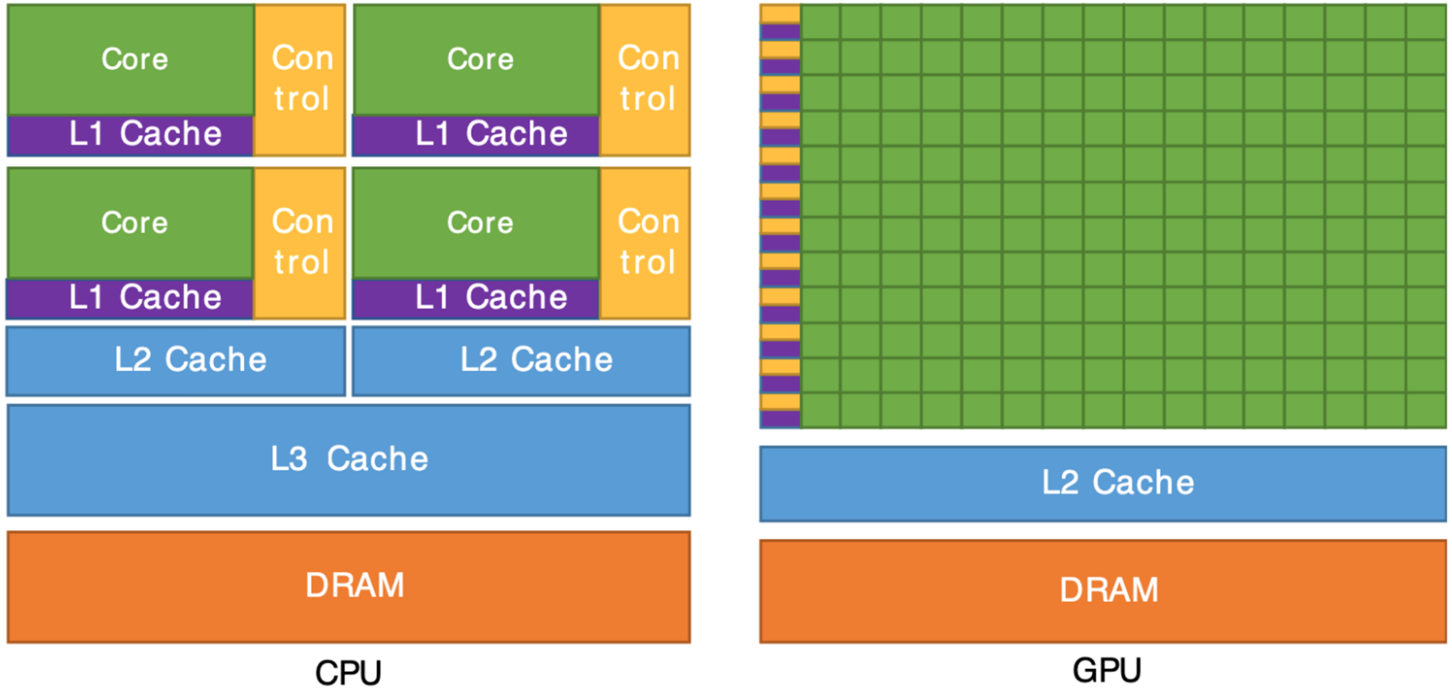
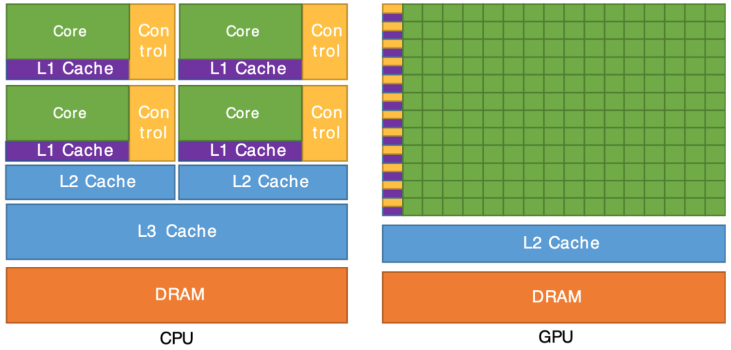


Image source: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

GPU Characteristics



Handle less complex workflow

Smaller cache memory

Many more arithmetic logic units and floating point units

Parallelize computation of large set of independent data

CUDA Programming

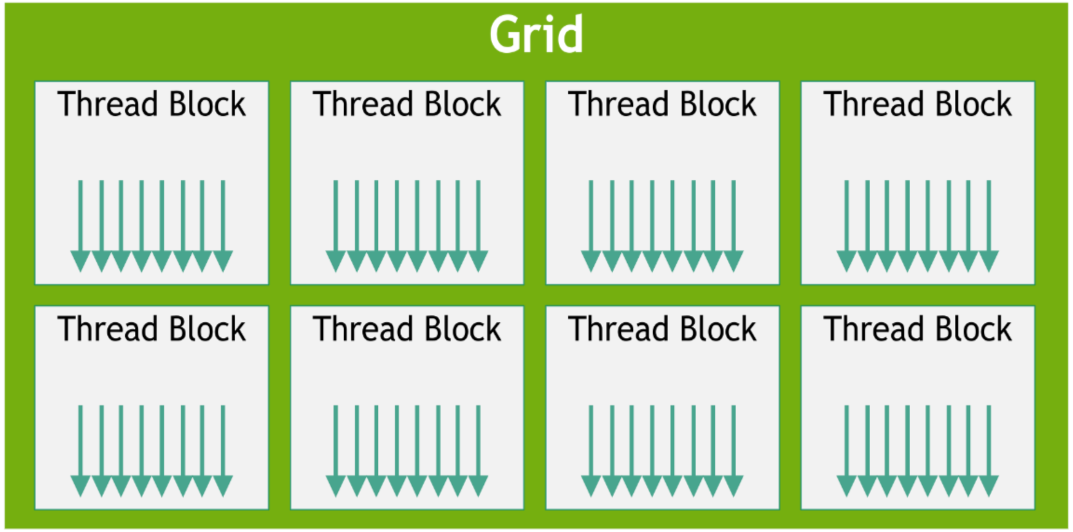


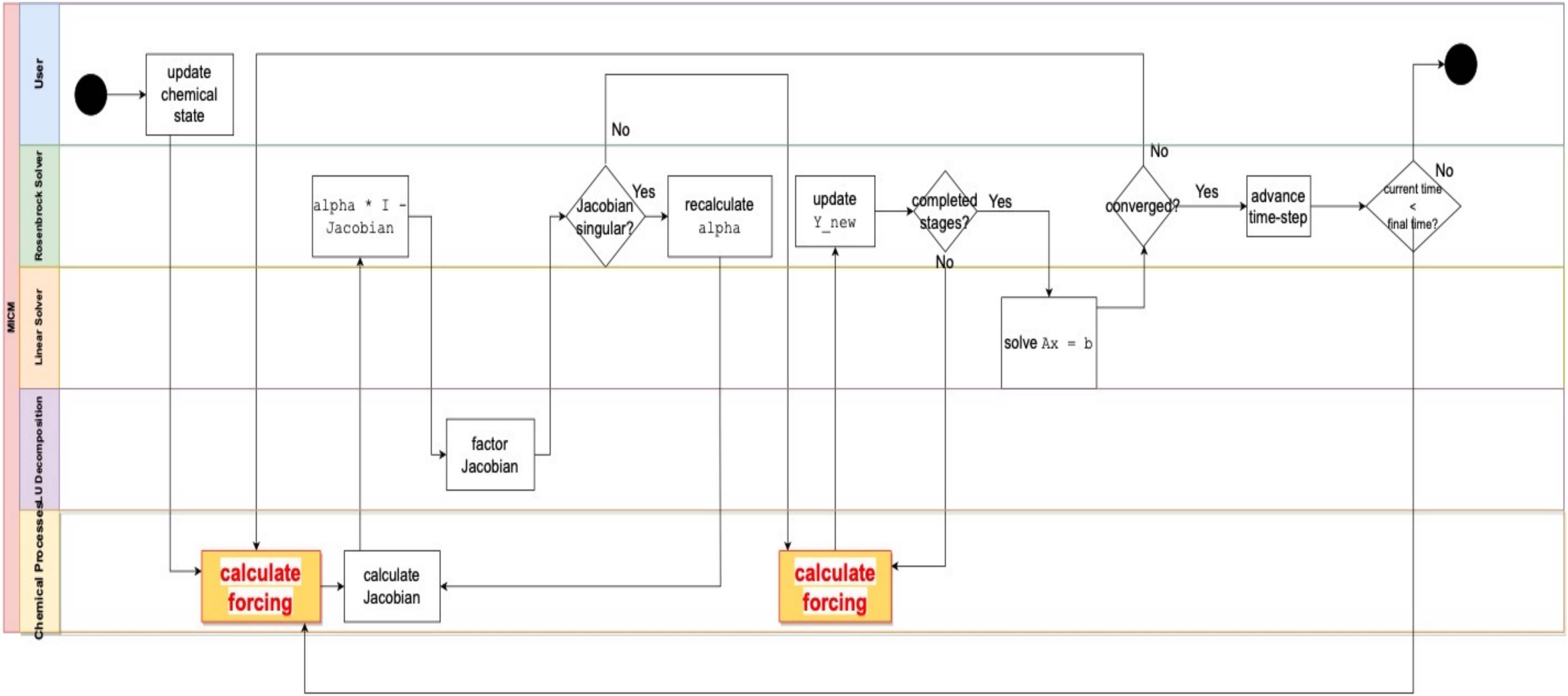
Image source: https://docs.nvidia.com/

Thread: a stream of instructions and data assigned to one process unit

Thread block: constructed by multiple threads

Grid: constructed by multiple thread blocks

MICM Flowchart



Forcing Calculation

- Calculates the **rate of change** in atmospheric composition **associated with rate constants** and **reactant concentrations** of a set of chemical reactions that occurs in the atmosphere
- Data (rate constants, composition concentrations, etc.) are organized into matrices

Matrix Computation

Columns: rate constants for each reaction in the chemical mechanism

R[0,0]	R[0,1]	R[0,2]	R[0,3]
R[1,0]	R[1,1]	R[1,2]	R[1,3]
R[2,0]	R[2,1]	R[2,2]	R[2,3]
R[3,0]	R[3,1]	R[3,2]	R[3,3]

Rows:
grids boxes in a 3D
atmosphere model

Matrix as Linear Vector

Row-major order:

R[0,0]	R[0,1]	R[0,2]	R[0,3]	R[1,0]	R[1,1]	R[1,2]	R[1,3]	...

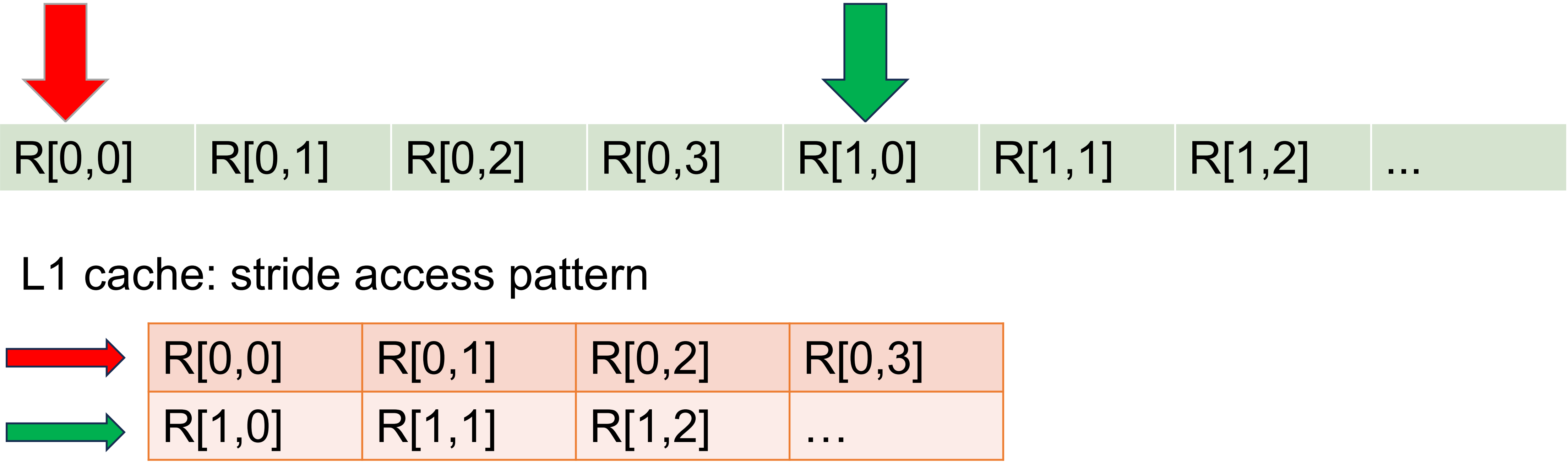
Column-major order:

R[0,0]	R[1,0]	R[2,0]	R[3,0]	R[0,1]	R[1,1]	R[2,1]	R[3,1]	…

Parallelism at Grid Level: Row-Major Order

🟥 thread 1

🟩 thread 2

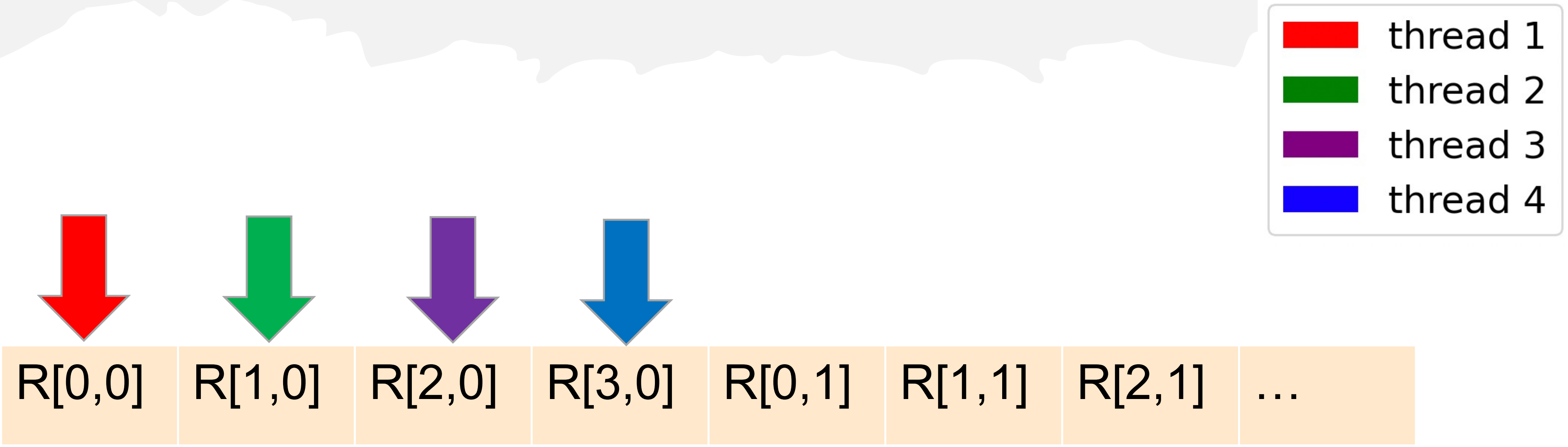


R[0,0]	R[0,1]	R[0,2]	R[0,3]	R[1,0]	R[1,1]	R[1,2]	...

L1 cache: stride access pattern

R[0,0]	R[0,1]	R[0,2]	R[0,3]
R[1,0]	R[1,1]	R[1,2]	…

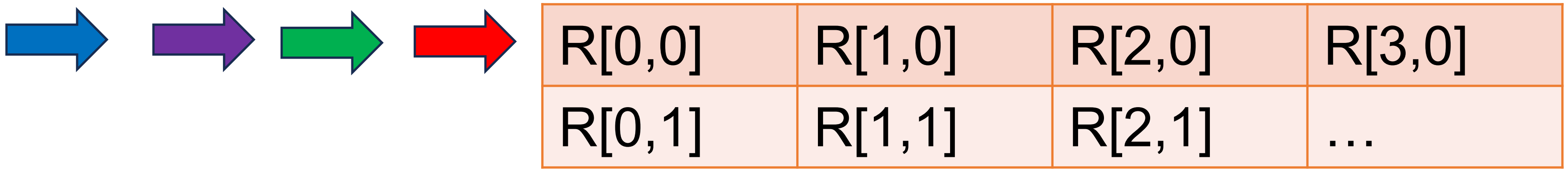
Parallelism at Grid-Level: Column-Major Order



R[0,0]	R[1,0]	R[2,0]	R[3,0]	R[0,1]	R[1,1]	R[2,1]	…

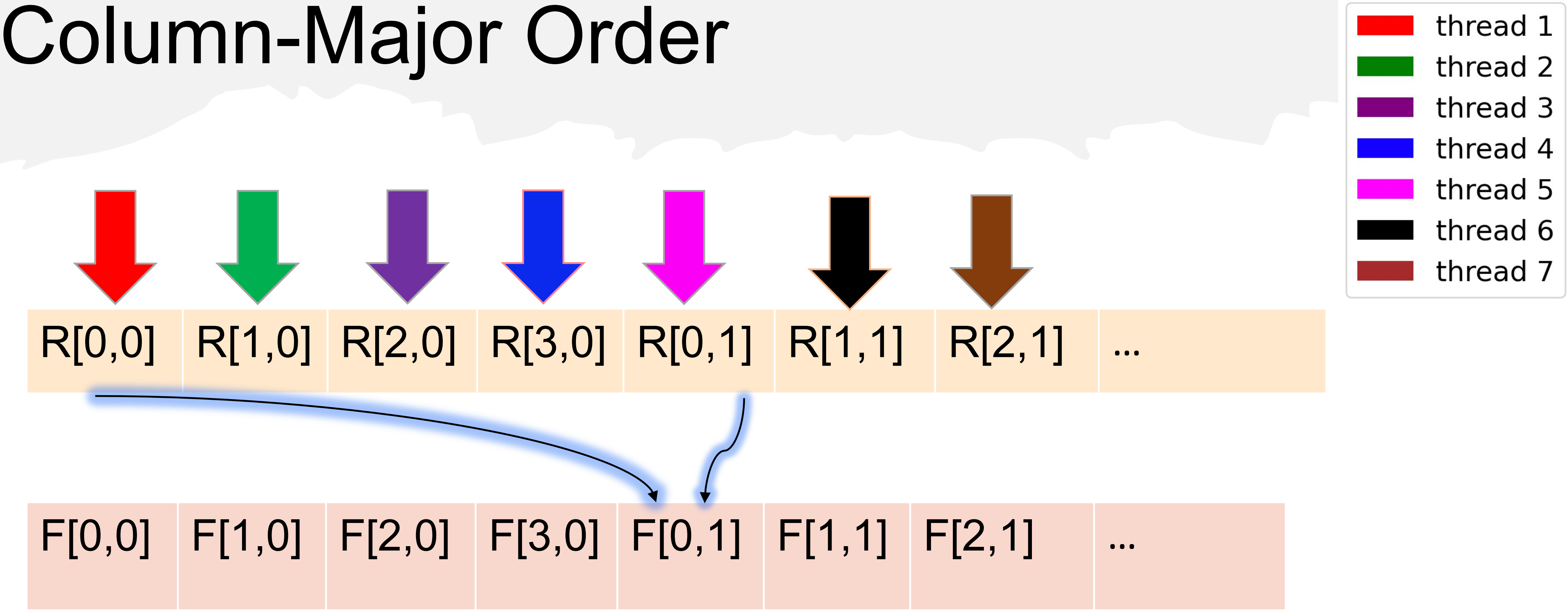
- 🟥 thread 1
- 🟩 thread 2
- 🟪 thread 3
- 🟦 thread 4

L1 cache: contiguous access pattern in parallel



R[0,0]	R[1,0]	R[2,0]	R[3,0]
R[0,1]	R[1,1]	R[2,1]	…

Parallelism at Grid/Reaction Level: Column-Major Order



🟥 thread 1
🟩 thread 2
🟪 thread 3
🟦 thread 4
🟪 thread 5
⬛ thread 6
🟫 thread 7

R[0,0]	R[1,0]	R[2,0]	R[3,0]	R[0,1]	R[1,1]	R[2,1]	...
F[0,0]	F[1,0]	F[2,0]	F[3,0]	F[0,1]	F[1,1]	F[2,1]	...

Problem: data race may happen!

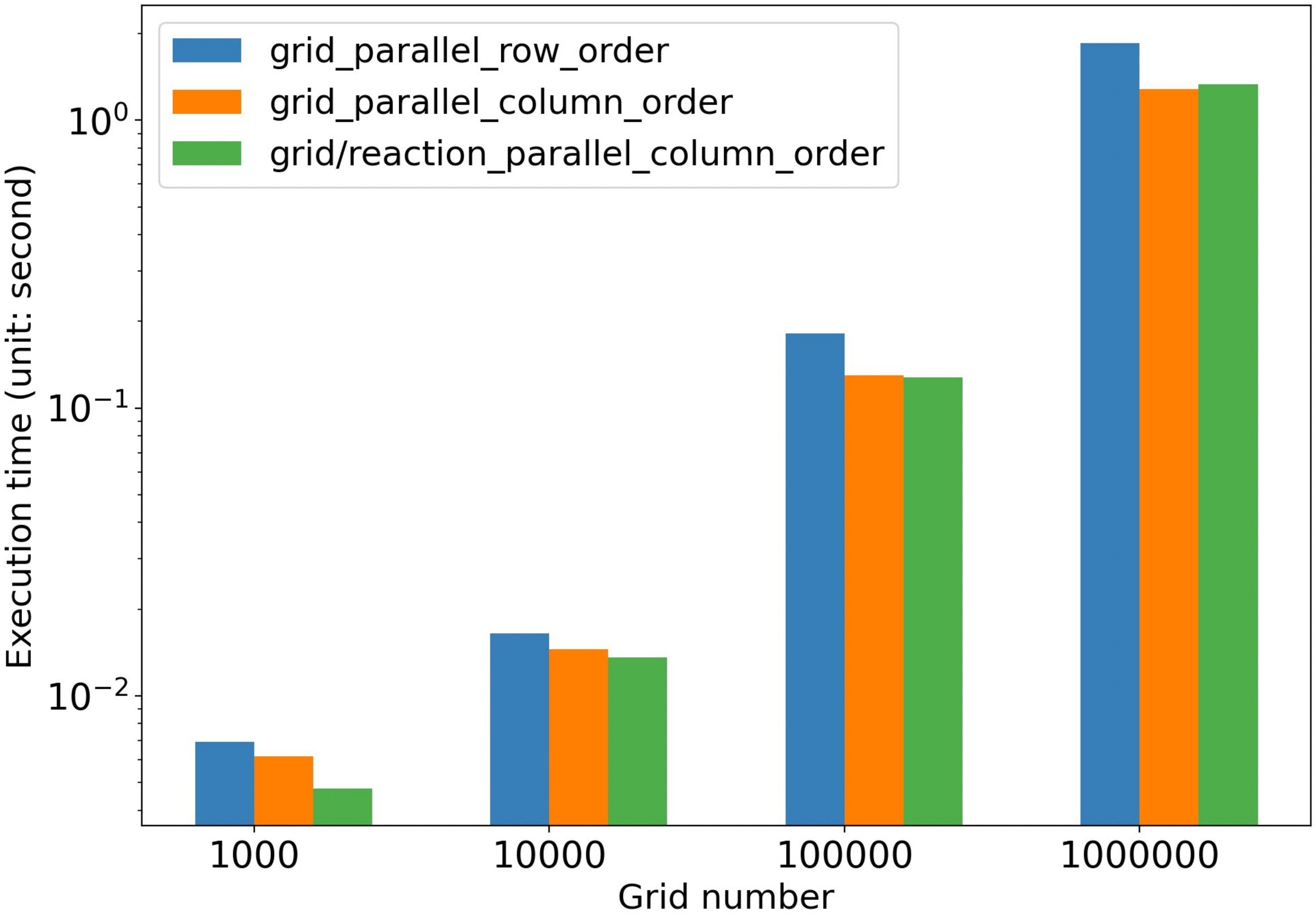
Solution with trade-off: Atomic operations → atomicAdd()

Experiments

- Machine: Gust
- Compiler: nvhpc/23.5
- Bit for Bit Accuracy of CPU code against GPU code
- CPU performance: 1 CPU core
- GPU performance: 1 NVIDIA A100 GPU (w/ and w/o data transfer time)
- 3 CUDA versions

Time Performances: GPU Implementations

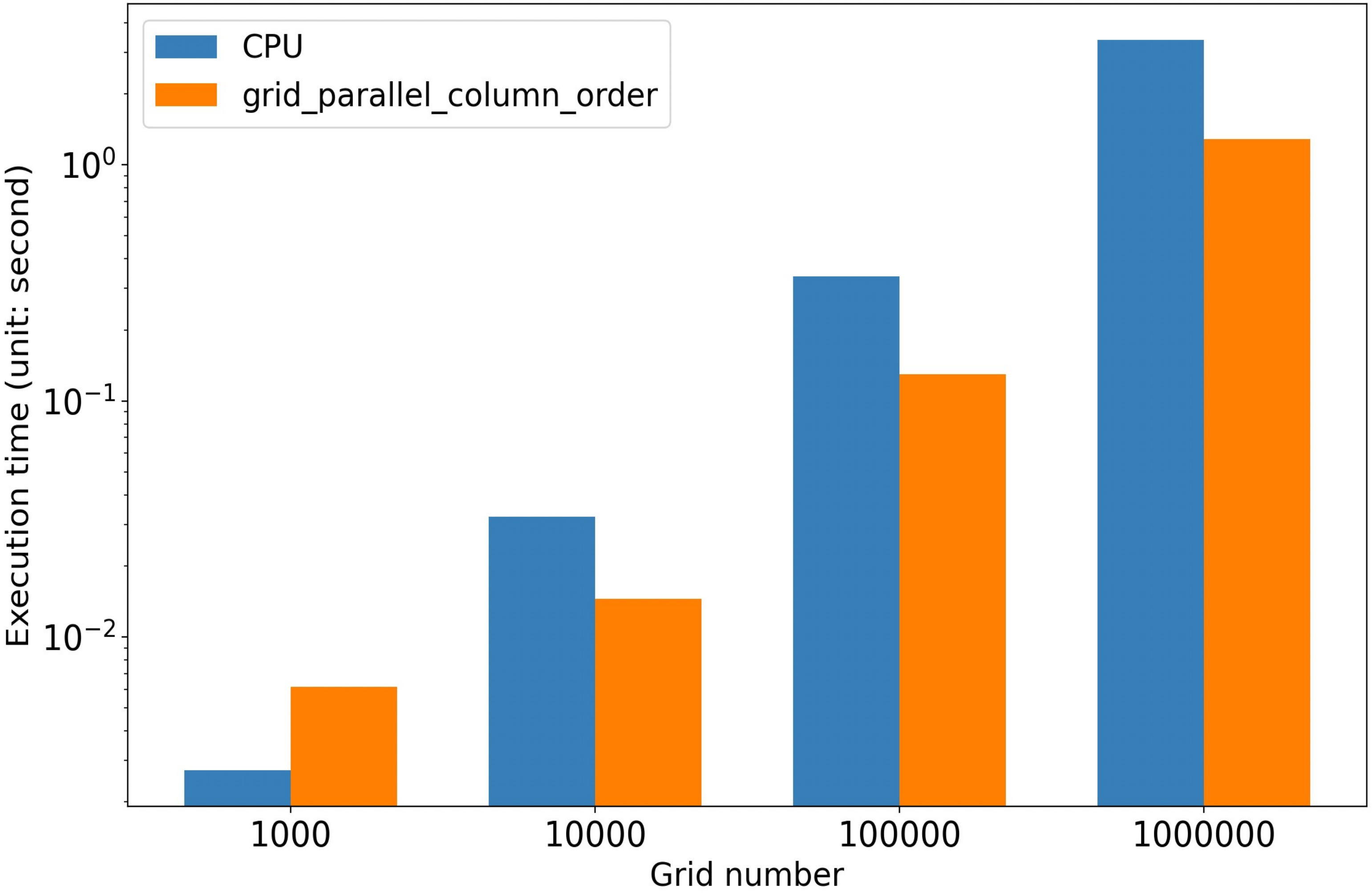
Constant inputs: 500 reactions, 400 chemical species



- **Base: Grid_parallel_row_order**
- **Avg. speedup: 1.44x (orange)**
- **Avg. speedup: 1.40x (green)**
- **(included data transfer time)**

Time Performances: CPU vs GPU

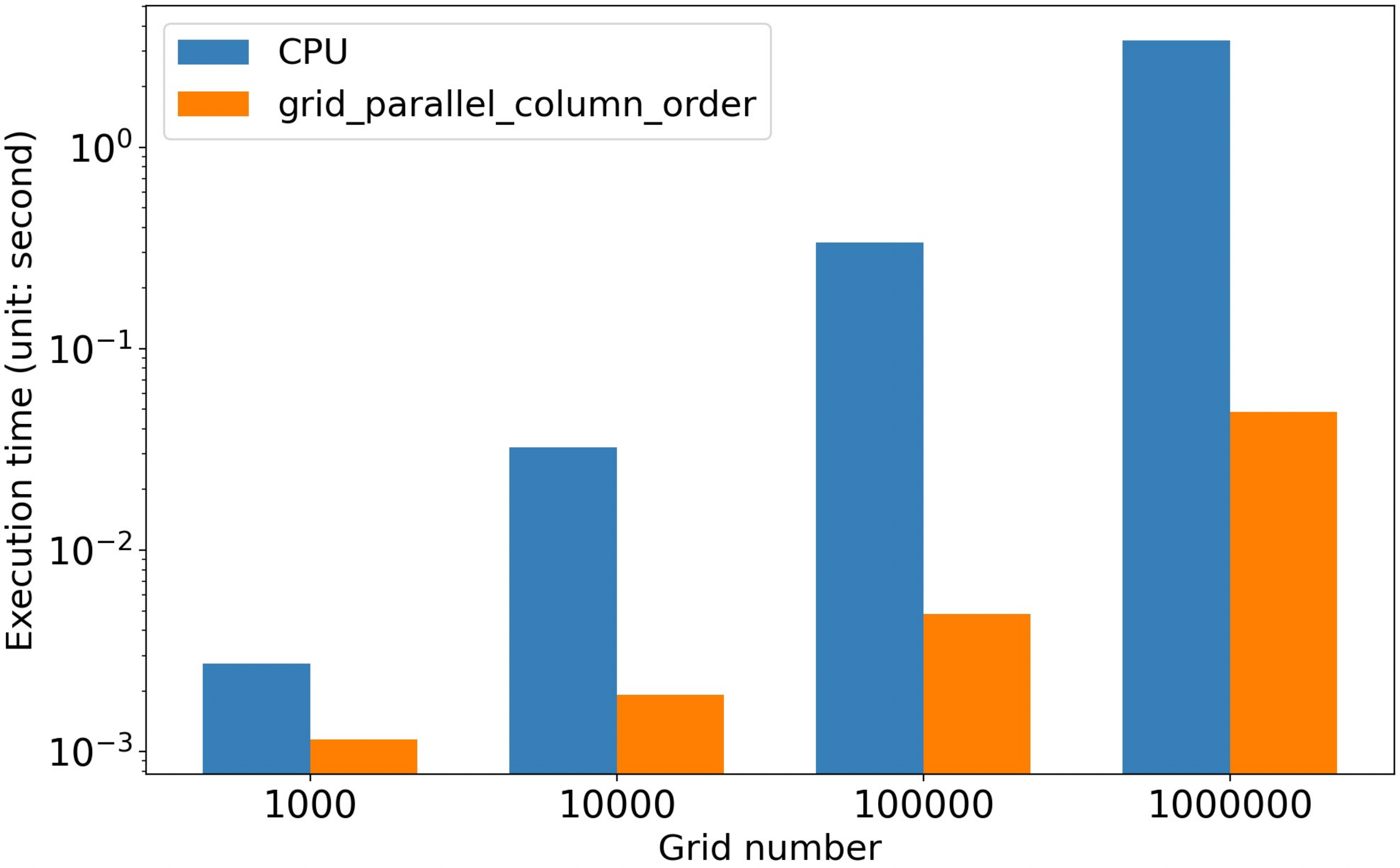
Constant inputs: 500 reactions, 400 chemical species



- **Avg. speedup: 2.62x**
- **w/ data transfer time**

Time Performances: CPU vs GPU

Constant inputs: 500 reactions, 400 chemical species



Avg. Speedup: 66.30x w/o data transfer time

Conclusion

- We ported AddForcingTerms() function to GPU via CUDA
- We evaluated different CUDA implementations
  - Different memory layouts
  - Different levels of parallelism
- Performances show increasing speedups with increasing problem size
- Future work:
  - Port more functions to GPU using similar approach
  - Explore just-in-time compilation GPU code

Acknowledgement

- **NCAR**: for providing funding
- **Jian Sun**, **John Dennis** and **Matthew Dawson**:

  my mentors, who provided invaluable guidance and

  insightful feedbacks throughout the project
- **Kyle Shores** (ACOM): for his expert assistance in making possible for my CUDA codes to compile and run
- **SIParCS Cohort:** for an unforgettable summer in Boulder!

Summer Internships in Parallel Computational Science

SIParCS

BOULDER • CO