GPU-lized cloud microphysics scheme in CAM

-- How we did it and what we learned?

Jian Sun, John Dennis

ASAP/CISL

June 01, 2021

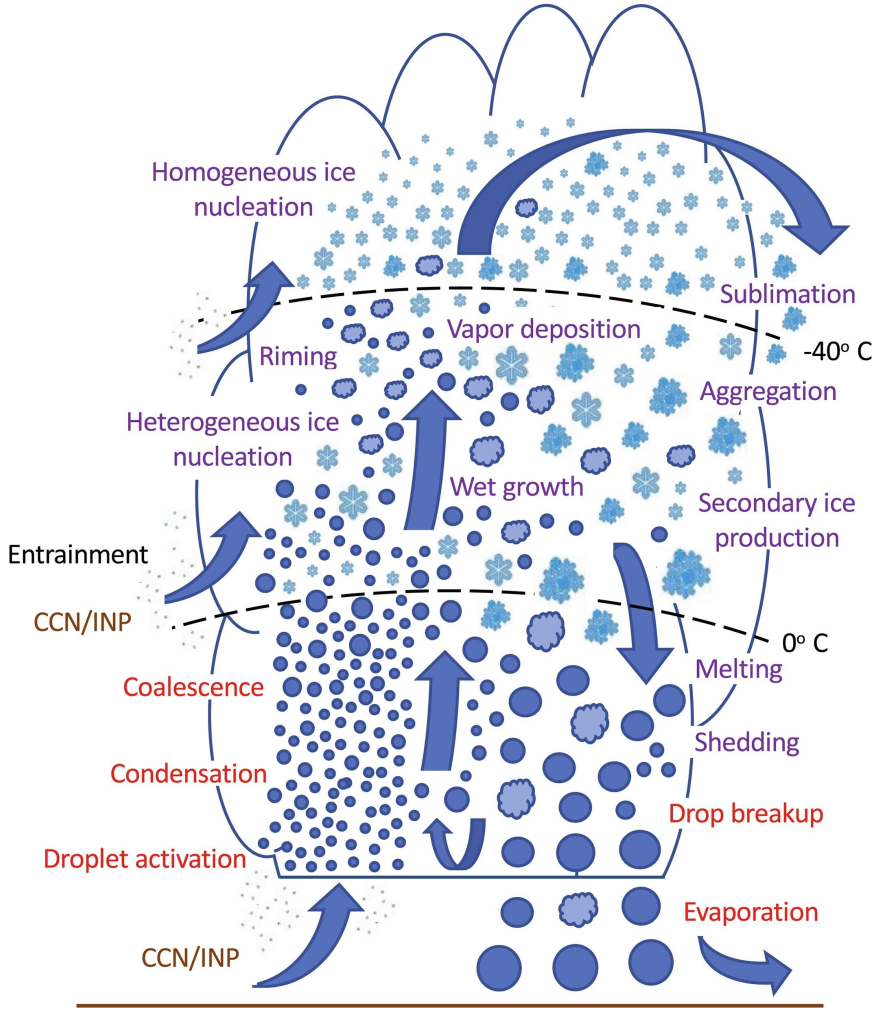
This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

Outline

- What is cloud microphysics?
- Code overview of cloud microphysics scheme in CAM
- Methodology
- Preliminary results & discussion
- Summary & Future work

What is cloud microphysics?

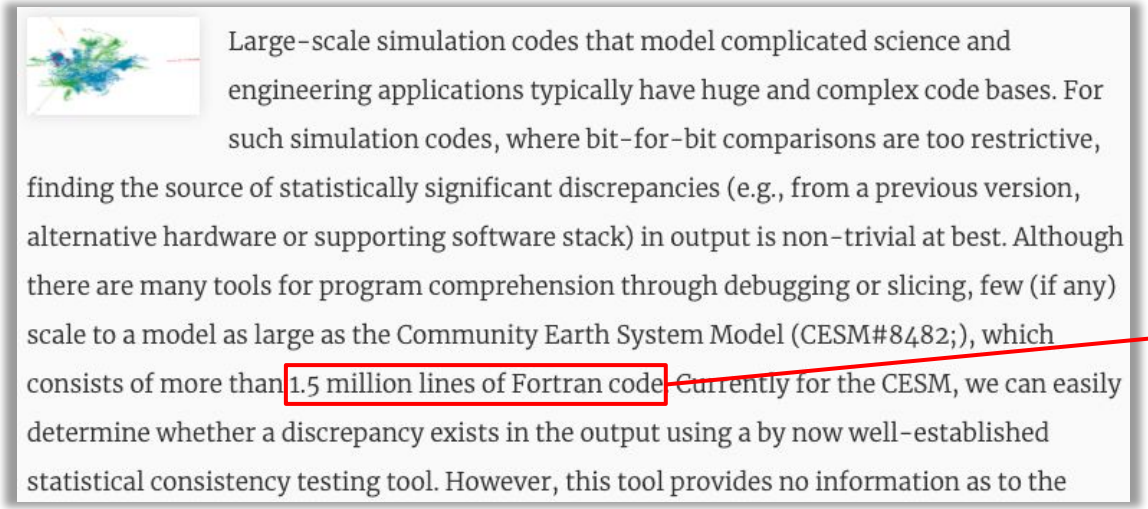
"…small-scale (from sub-micron to cm) processes driving the formation and evolution of cloud and precipitation particles…"



(Morrison et al., 2020)

Code review of cloud microphysics scheme: PUMAS

- PUMAS:
  - micro_mg3_0.F90: 3782 lines, 3 subroutines/functions
  - micro_pumas_utils.F90: 3151 lines, 55 subroutines/functions
- CAM:
  - wv_sat_methods.F90: 767 lines, 32 subroutines/functions
  - wv_saturation.F90: 1350 lines, 30 subroutines/functions
  - 27 additional CAM codes related to wv_* F90



Large-scale simulation codes that model complicated science and engineering applications typically have huge and complex code bases. For such simulation codes, where bit-for-bit comparisons are too restrictive, finding the source of statistically significant discrepancies (e.g., from a previous version, alternative hardware or supporting software stack) in output is non-trivial at best. Although there are many tools for program comprehension through debugging or slicing, few (if any) scale to a model as large as the Community Earth System Model (CESM#8482;), which consists of more than 1.5 million lines of Fortran code. Currently for the CESM, we can easily determine whether a discrepancy exists in the output using a by now well-established statistical consistency testing tool. However, this tool provides no information as to the

(Milroy et al., 2019)

PUMAS takes about ~5% of computational time of CAM

We need to change ~0.6% of CESM codes

PUMAS Github Repo: https://github.com/ESCOMP/PUMAS
CAM Github Repo: https://github.com/PUMASDevelopment/CAM

Offload to GPU

- Start from the existing efforts on MG2 porting
- Use OpenACC to port CPU codes to GPU
  - Directives-based parallel programming model □ Single source code ✓
  - Convert CPU codes to GPU codes by adding pragmas (Readability ✓)
  - Users can explicitly manage the compute kernels, data movement, etc

- PUMAS:
  - micro_mg3_0.F90: 3782 lines □ 4098 lines
  - micro_pumas_utils.F90: 3151 lines □ 3535 lines
- CAM:
  - wv_sat_methods.F90: 767 lines □ 870 lines
  - wv_saturation.F90: 1350 lines □ 1484 lines

Preliminary result: Correctness

- Does the GPU version of PUMAS/CAM codes return the bit-for-bit results compared with the CPU version of codes?
  - If yes, that is great!
  - If no, we need to ask ourselves "Is this difference expected?"
    - If yes, run a validation test (e.g., ECT, AMWG diagnostics package, etc)
    - If no, it could be something we do not understand fully (likely) or a code bug (more likely)
- Always check the correctness before looking at the performance.

Preliminary result: Performance

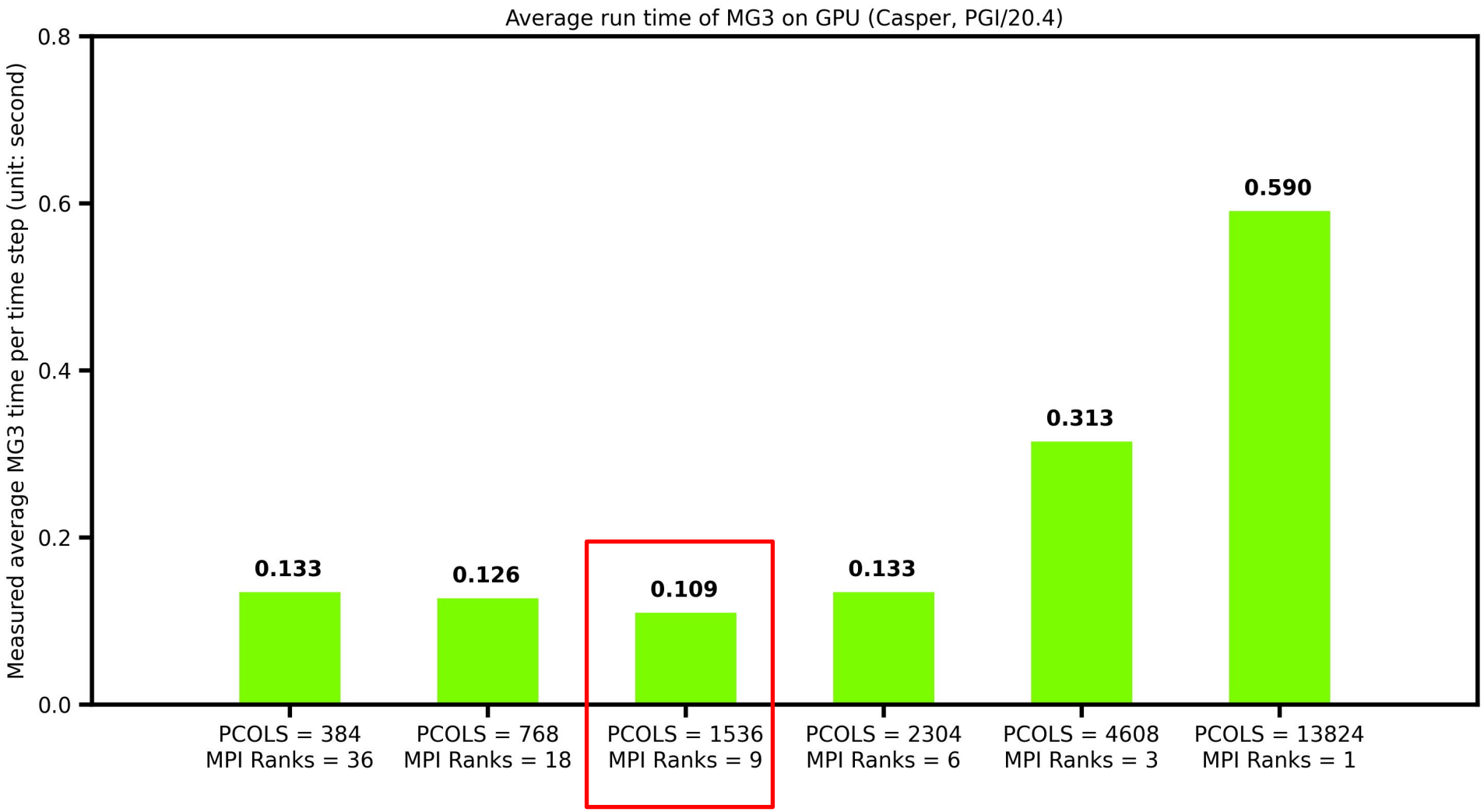
- Test configurations:
  - Compset: F2000climo
  - Resolution: f19 (96 nlat x 144 nlon = 13,824 columns, 32 layers)
  - Simulation length: 9 time steps
  - Machine: Casper, Cheyenne
  - Resource: 1 node with 36 CPU cores, 1 V100 GPUs
  - PCOLS: 16 to 384 □ different data sizes on GPU
  - MPI tasks: 1 to 36
- We focus on the computational time of MG tendency subroutine

1 Node: 1 GPU vs. 36 CPU cores

Case	wallmax time for MG3 per time step per rank (unit: second)
1 Cheyenne node (36 MPI ranks, PCOLS=16, Intel/19.0.5)	**0.120**
1 Casper node w/o GPU (36 MPI ranks, PCOLS=16, Intel/19.0.5)	**0.087**
1 Casper node w/o GPU (36 MPI ranks, PCOLS=16, PGI/20.4)	**0.229**
1 Casper node w/o GPU (36 MPI ranks, PCOLS=384, PGI/20.4)	**0.213**
1 Casper node w/ 1 GPU (36 MPI ranks, PCOLS=384, PGI/20.4)	**0.133**

1 Casper node w/ 1 V100 ≈ **0.9** Cheyenne node
≈ **0.65** Casper node w/o GPU using Intel
≈ **1.6** Casper node w/o GPU using PGI

1 GPU + each MPI rank has only 1 chunk



Faster than one Cheyenne node

Summary

❖What we has done/learned:

- Offload the cloud microphysics scheme (i.e., PUMAS) in CAM to GPU
- Check the correctness through ensemble consistency test (ECT)
- Evaluate/compare the performance on CPU and GPU
- Even one GPU per node is showing promising results

❖What is next?

- Do the profiling and look for further optimizations
  - Data movement
  - Kernel optimization
- Collaborate with more people to (1) resolve issues and (2) improve functionalities

Acknowledge

❖This work is funded by the following project:

- NSF CSSI EarthWorks (Award number: 2005137)
- NSF NCAR-base funding

❖Many thanks to contributions/helps from different labs/organizations:

- **CISL**: Rich Loft, Supreeth Madapur Suresh, Cena Miller, Brian Vanderwende, Daniel Howard, Allison Baker
- **CGD**: Andrew Gettelman, Katherine Thayer-Calder, Jim Edwards
- **NVIDIA**: Raghu Raj Prasanna Kumar


THANK YOU FOR LISTENING
ANY QUESTION ???