27th CESM workshop

June 15, 2022

Enabling the execution of PUMAS on GPUs

Jian Sun, John Dennis, Sheri Mickelson, Brian Vanderwende,
Andrew Gettelman, Katherine Thayer-Calder

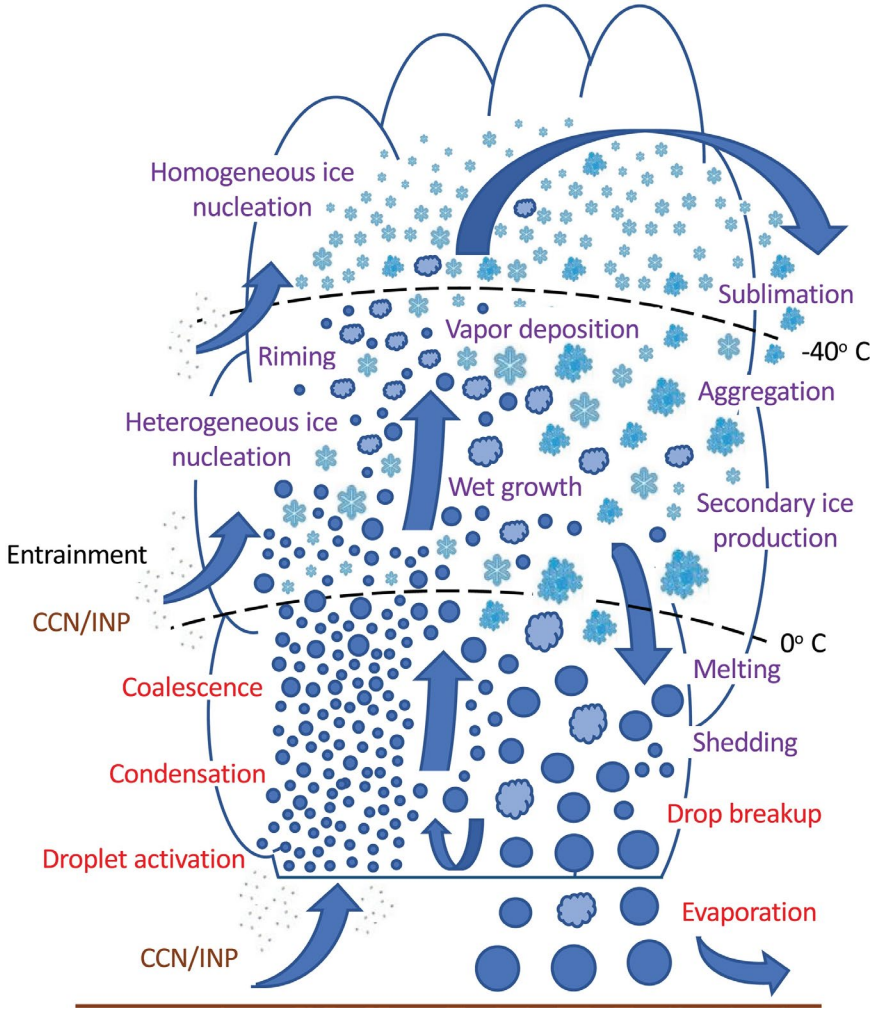
This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

Outline

- What is cloud microphysics?
- Code overview of cloud microphysics scheme in CAM (PUMAS)
- 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_pumas_v1.F90: ~2900 lines, 7 subroutines/functions
  - micro_pumas_utils.F90: ~2000 lines, 54 subroutines/functions
- CAM:
  - wv_sat_methods.F90: ~400 lines, 29 subroutines/functions
  - wv_saturation.F90: ~700 lines, 30 subroutines/functions
  - 33 additional CAM codes related to wv_* F90
- CAM physics source codes: ~410,000 lines
  - PUMAS represents ~1.2% of CAM physics codes
  - PUMAS contributes to ~8% of computational time of CAM physics

Offload to GPU

- Start from a KGen kernel (easy for code development and debug)
- Use **OpenACC** and **OpenMP offload** to port CPU codes to GPU
  - Directives-based parallel programming model → Single source code ✓
  - Convert CPU codes to GPU codes by adding pragmas (Readability ✓)
  - Intel's auto-migration tool from OpenACC to OpenMP offload (https://github.com/intel/intel-application-migration-tool-for-openacc-to-openmp)

Code size of OpenACC version (similar for OpenMP offload)

- PUMAS:
  - micro_pumas_v1.F90: ~2900 lines → ~3200 lines
  - micro_pumas_utils.F90: ~2000 lines → ~2200 lines
- CAM:
  - wv_sat_methods.F90: ~400 lines → ~460 lines
  - wv_saturation.F90: ~700 lines → ~790 lines

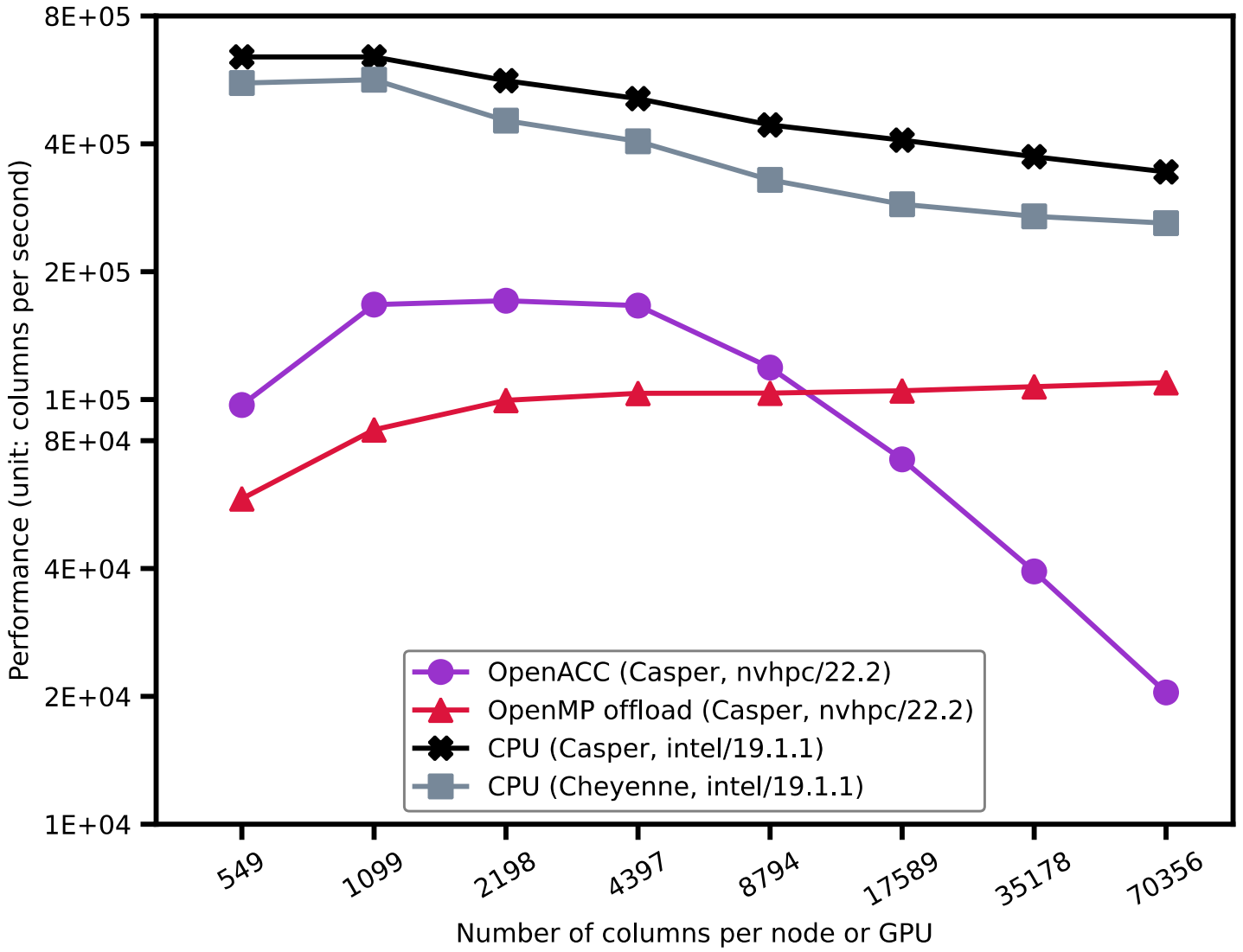
Increase the code size by **~10%**

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 verification 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.

Performance of KGen kernel

- One CPU node (CPU run) vs. one MPI rank + one GPU (GPU run)
- Log scale for X- and Y-axis
- GPU performs worse than CPU consistently



Profiling KGen kernel

- Example of computational hotspots in the GPU code: loops with dependency

Original implementation

```
do k=2,nlev
   do i=1,mgncol
      if (qc(i,k) < qsmall .and. qi(i,k) < qsmall) then
         precip_frac(i,k) = precip_frac(i,k-1)
      end if
   end do
end do
```

- Solution: reverse the loop order

New implementation

```
do i=1,mgncol
   do k=2,nlev
      if (qc(i,k) < qsmall .and. qi(i,k) < qsmall) then
         precip_frac(i,k) = precip_frac(i,k-1)
      end if
   end do
end do
```

Profiling KGen kernel (cont'd)

- Example of computational hotspots in the GPU code: sedimentation subroutines run in serial
- Solution: run them asynchronously

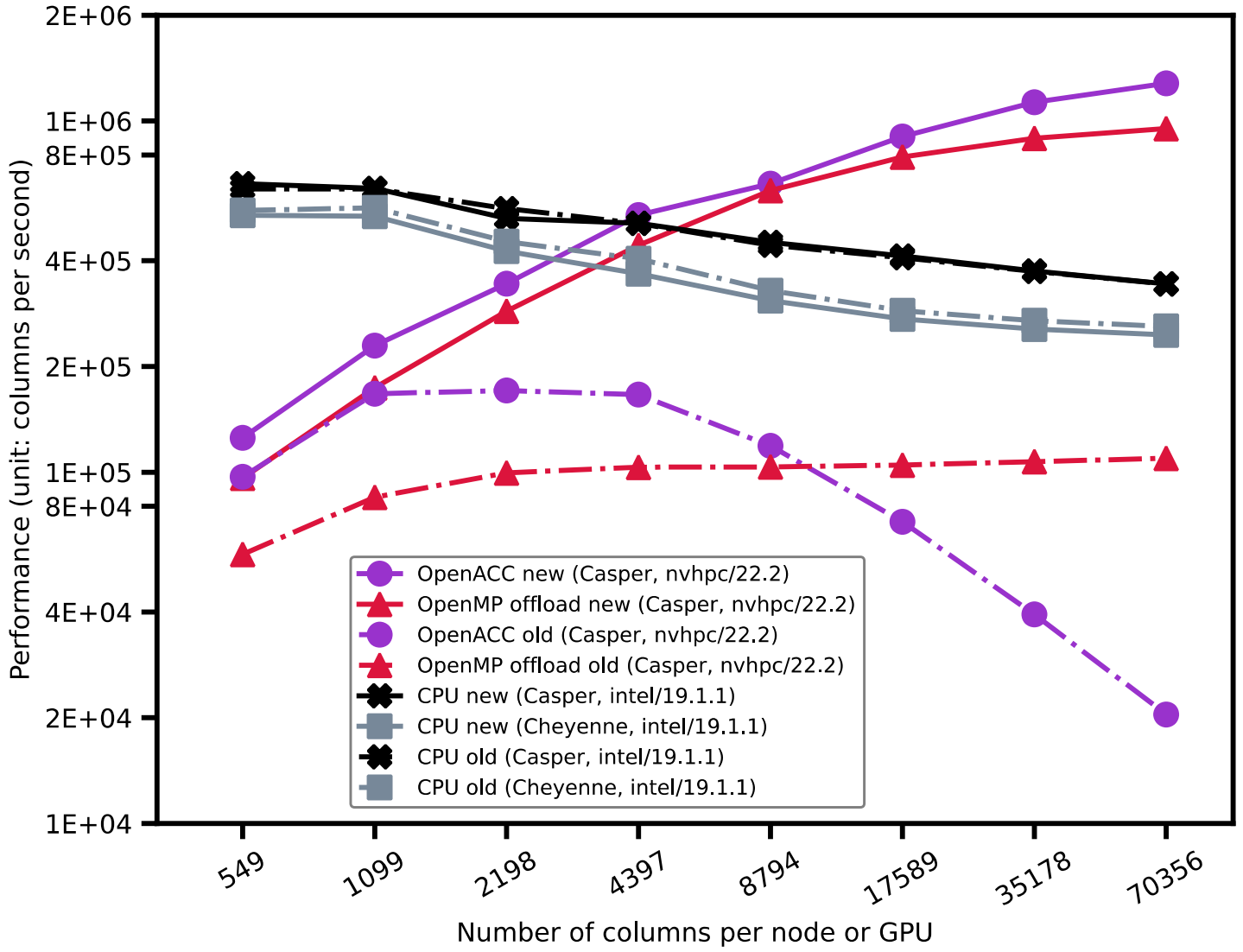
Original implementation

```
! ice
call Sedimentation(mgncol,nlev,do_cldice,deltat,fi,fni,pdel_inv, &
                   qitend,nitend,qisedten,dumi,dumni,prect,iflx, &
                   xxlx=xxls,qxsevap=qisevap,tlat=tlat,qvlat=qvlat, &
                   xcldm=icldm,preci=preci)
! liq
call Sedimentation(mgncol,nlev,.TRUE.,deltat,fc,fnc,pdel_inv, &
                   qctend,nctend,qcsedten,dumc,dumnc,prect,lflx, &
                   xxlx=xxlv,qxsevap=qcsevap,tlat=tlat,qvlat=qvlat,xcldm=lcldm)
! rain
call Sedimentation(mgncol,nlev,.TRUE.,deltat,fr,fnr,pdel_inv, &
                   qrtend,nrtend,qrsedten,dumr,dumnr,prect,rflx)
! snow
call Sedimentation(mgncol,nlev,.TRUE.,deltat,fs,fns,pdel_inv, &
                   qstend,nstend,qssedten,dums,dumns,prect,sflx,preci=preci)
! graupel
call Sedimentation(mgncol,nlev,.TRUE.,deltat,fg,fng,pdel_inv, &
                   qgtend,ngtend,qgsedten,dumg,dumng,prect,gflx,preci=preci)
```

New implementation

```
! ice
call Sedimentation(mgncol,nlev,do_cldice,deltat,fi,fni,pdel_inv, &
                   qitend,nitend,qisedten,dumi,dumni,prect_i,iflx,IQUEUE, &
                   xxlx=xxls,qxsevap=qisevap,tlat=tlat_i,qvlat=qvlat_i, &
                   xcldm=icldm,preci=preci_i)
! liq
call Sedimentation(mgncol,nlev,.TRUE.,deltat,fc,fnc,pdel_inv, &
                   qctend,nctend,qcsedten,dumc,dumnc,prect_l,lflx,LQUEUE, &
                   xxlx=xxlv,qxsevap=qcsevap,tlat=tlat_l,qvlat=qvlat_l,xcldm=lcldm)
! rain
call Sedimentation(mgncol,nlev,.TRUE.,deltat,fr,fnr,pdel_inv, &
                   qrtend,nrtend,qrsedten,dumr,dumnr,prect_r,rflx,RQUEUE
! snow
call Sedimentation(mgncol,nlev,.TRUE.,deltat,fs,fns,pdel_inv, &
                   qstend,nstend,qssedten,dums,dumns,prect_s,sflx,SQUEUE preci=preci_s)
! graupel
call Sedimentation(mgncol,nlev,.TRUE.,deltat,fg,fng,pdel_inv, &
                   qgtend,ngtend,qgsedten,dumg,dumng,prect_g,gflx,GQUEUE preci=preci_g)
```

Performance of KGen kernel (new implementations)



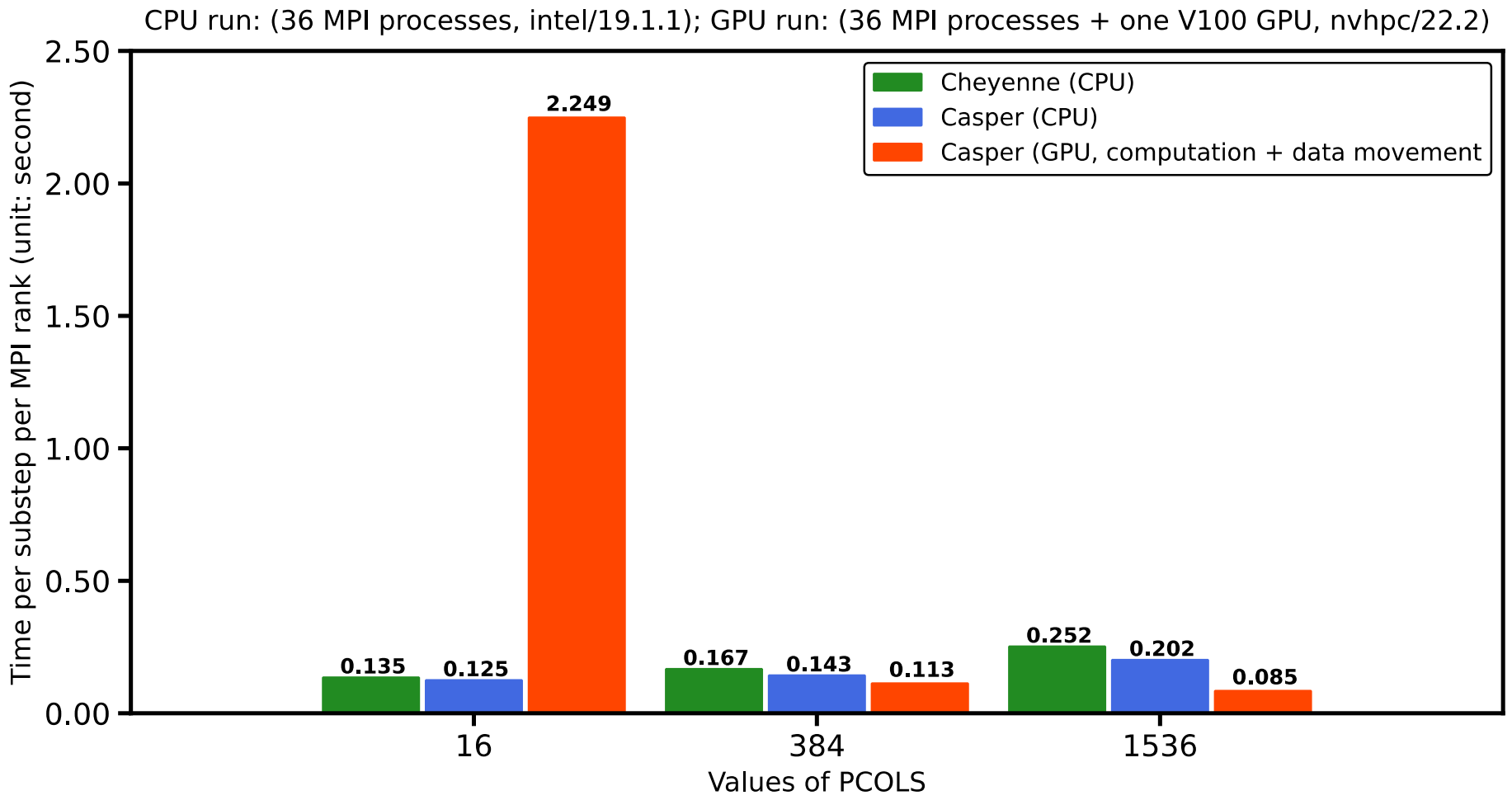
After some optimizations, we find:

- The performance of OpenACC and OpenMP offload is largely improved, which the CPU performance is marginally affected
- The performance of OpenACC and OpenMP offload is competitive, but OpenACC is better in general
- GPU outperforms CPU for large problem size

CAM configuration

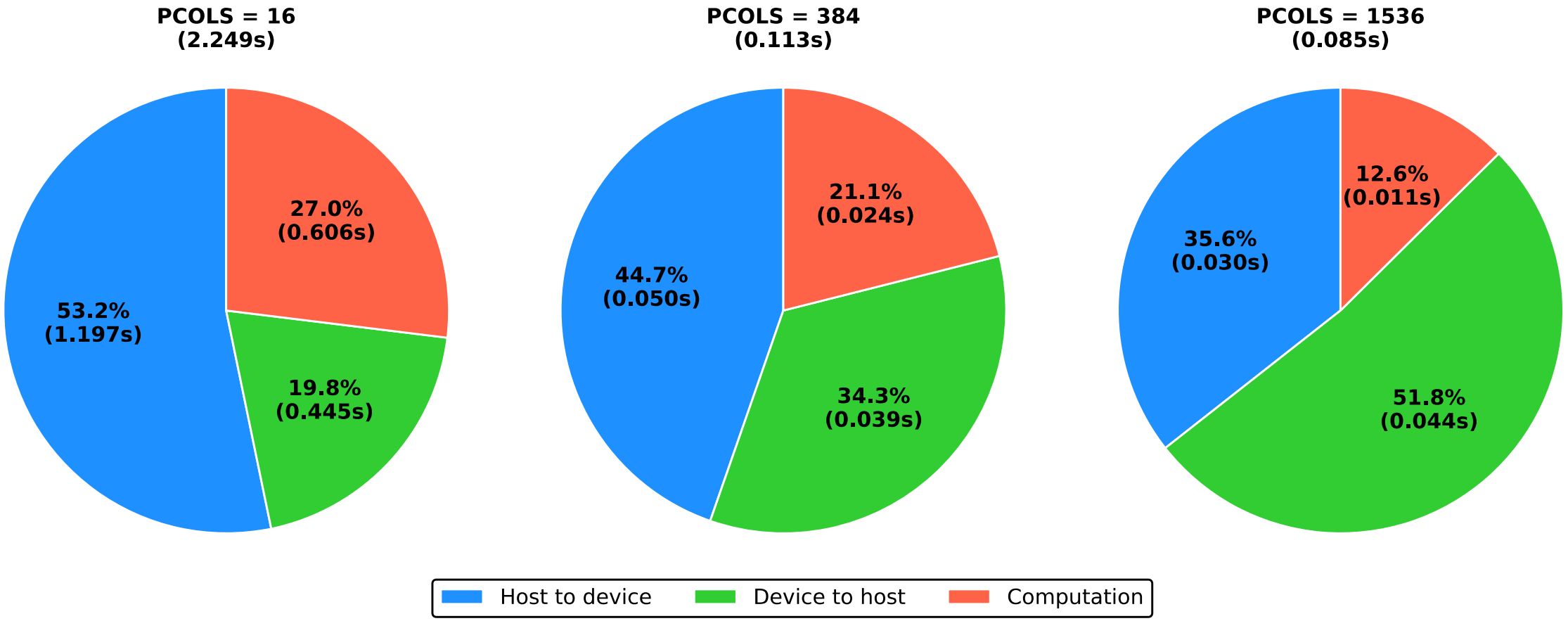
- Test configurations:
  - Compset: F2000dev
  - Cloud microphysics: MG3 configuration
  - Dycore: FV
  - Resolution: 1 degree for FV
  - Simulation length: 1 day
  - Machine: Cheyenne (CPU), Casper (CPU and GPU)
  - Resource: one compute node with 36 CPU cores, one V100 GPU per node
  - PCOLS: different data sizes offloaded to GPU per time

Performance of PUMAS in CAM



- GPU-enabled PUMAS can outperform its CPU version in a practical CAM simulation

Computation vs. data movement in a GPU run



- Data transfer could be more time-consuming than computation

Summary

- What we have done/learned:
  - Port PUMAS to GPU by OpenACC and OpenMP offload
  - Evaluate the performance on CPU and GPU
  - Even one GPU per node is showing promising results
  - OpenMP offload is relatively immature compared to OpenACC → more evaluations needed
- What is next?
  - Multiple GPUs per node
  - High-resolution CAM simulation
  - Sensitivity tests with different PUMAS configuration
  - One manuscript is under preparation

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**: Supreeth Madapur Suresh, Cena Miller, Allison Baker, Daniel Howard
  - **CSEG**: Jim Edwards
  - **CGD**: Cheryl Craig, Steve Goldhaber
  - **Areanddee**: Rich Loft
  - **NVIDIA**: Raghu Raj Prasanna Kumar
  - **INTEL**: Harald Servat
  - **ORNL**: Youngsung Kim