Optimizing the Weather Research and Forecasting Model with OpenMP Offload and Codee

Chayanon (Namo) Wichitrnithed[1], Woo-Sun Yang[2], Helen He[2], Brad Richardson[2], Koichi Sakaguchi[3], Manuel Arenaz[4], William I. Gustafson[3], Jacob Shpund[5], Ulises Costi Blanco[4], Alvaro Goldar Dieste[4]
[1]Oden Institute, UT Austin, [2]Lawrence Berkeley National Laboratory,
[3]Pacific Northwest National Laboratory, [4]Appentra Solutions S.L., [5]The Hebrew University of Jerusalem

ABSTRACT

Currently, the Weather Research and Forecasting model (WRF) utilizes shared memory (OpenMP) and distributed memory (MPI) parallelisms. To take advantage of GPU resources on Perlmutter, we port parts of the computationally expensive routine Fast Spectral Bin Microphysics (FSBM) to GPUs using OpenMP device offloading. The optimization process is guided by a combination of profilers and a static code inspection tool Codee, resulting in a 1.88x overall speedup for the CONUS-12km winter storm test case.

BACKGROUND

- **The Weather Research and Forecasting Model (WRF)** is a numerical weather prediction system written in Fortran and is widely used for both research and operational work
- Solves the 3D Euler equations using finite differences, and parallelized through domain decomposition (MPI) and shared work within each domain (OpenMP)
- Computationally expensive routine: **Fast Spectral-Bin Microphysics (FSBM)** scheme computes discrete 33 particle sizes for each grid point (Figure 1)

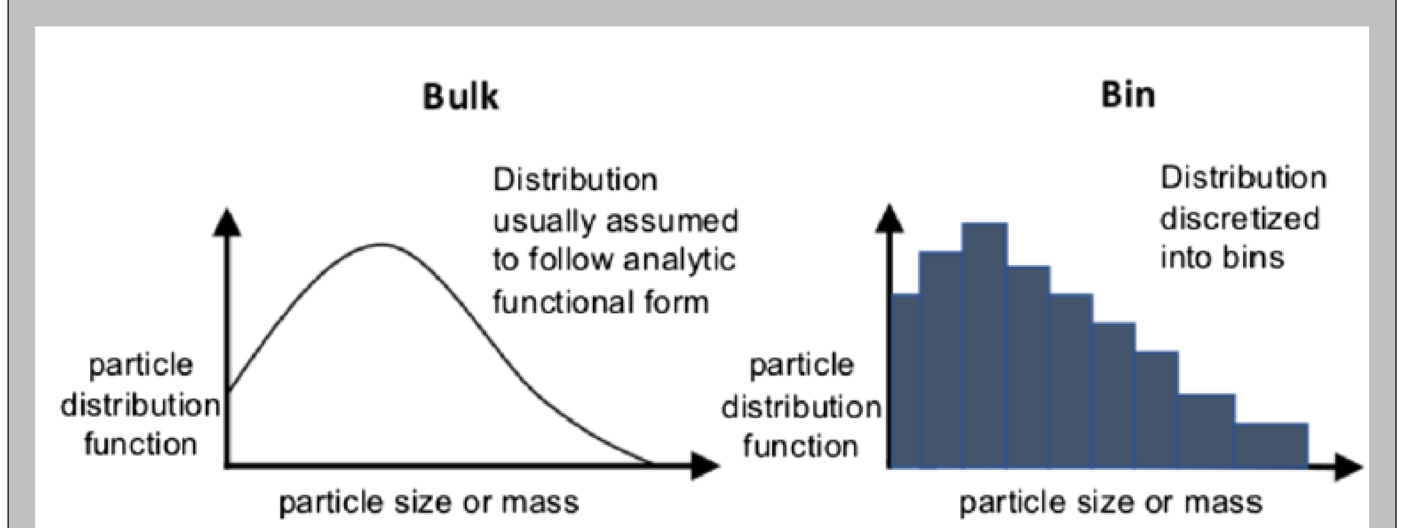

Figure 1: Comparison between bulk microphysics schemes (left) and the more expensive bin schemes (right). Image from Morrison et al,. 2020.

- **Codee** is a command-line static code inspection tool for C, C++ and Fortran that analyzes the source code for modernization and performance-related issues

OBJECTIVE

Explore the use of both profilers and static code analyzer (Codee) to offload expensive routines of WRF to GPUs on Perlmutter through OpenMP offload capabilities

METHODOLOGY

1. A hotspot subroutine within FSBM is identified by gprof, but contains **shared arrays between grid points**

```
do j = jts:jte
  do k = kts:kte
    do i = its:ite
      ! Collision-Coalesence process
      call COAL_BOTT_NEW(...)
```

2. Get dependency information from Codee and **replace shared lookup arrays with independent function calls**

```
! Codee: Loop modified
!$omp target teams distribute parallel do &
!$omp private(n) map(from: cwlg, cwls, ...) ...
do n = 1,33
  ! Codee: Loop modified
  !$omp simd
  do m = 1,33
    ckern_1 = ...
    ckern_2 = ...
    cwlg(m,n) = (ckern_2 + (ckern1-ckern_2* ..)) * ...
```

Codee detects no loop-carried dependencies

3. Offload only outer 2 loops (j,k) due to **large stack usage by automatic arrays inside the subroutine**

4. Replace automatic arrays with slices of global arrays, allowing a **full collapse of the 3 loops**, increasing thread occupancy

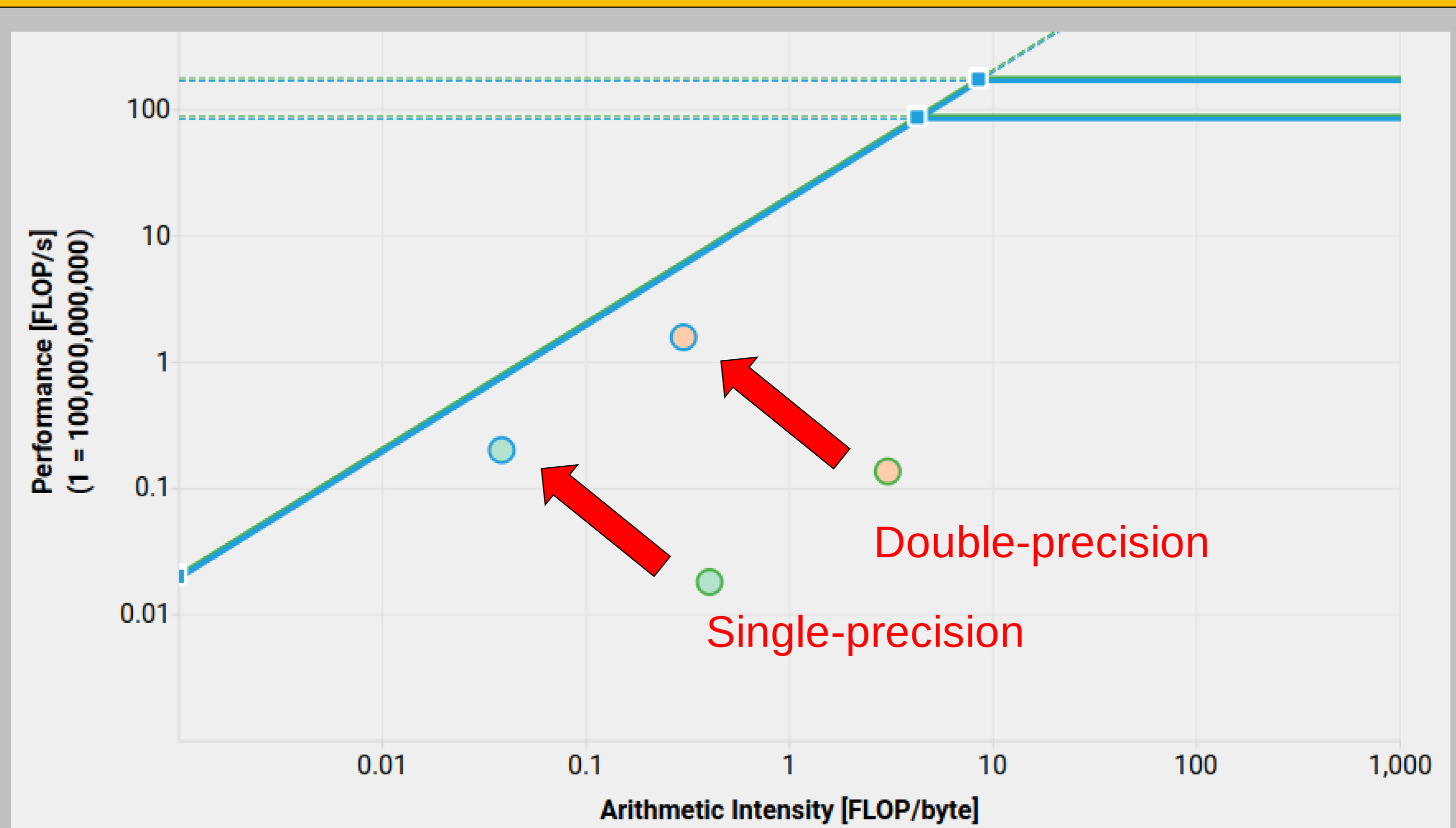

Figure 2: GPU roofline plot comparison of step 3 and 4, showing an increase in FLOP/s from higher occupancy.

RESULTS

- Test case: 10-min simulation of CONUS-12km (2019 Winter Storm) with 325 x 400 x 50 grid points.
- Compiled using PrgEnv-nvidia/8.5.0
- Timings done using 32 MPI ranks and 1 GPU per rank

	Time (s)	Speedup	Cumulative Speedup
Baseline	733.7	-	-
Lookup Optimization	506.2	1.44x	1.44x
Offloading outer 2 loops	423.3	1.19x	1.73x
Offloading all 3 loops	388.8	1.08x	1.88x

Table 1: Wall time and speedup comparison of different steps of optimization

CONCLUSION

- Achieved a total of 1.88x speedup for the CONUS-12km case through porting major FSBM calculations to GPUs with OpenMP
- Explored a workflow for optimization: identifying hotspots with profilers and using Codee to gain more detailed insights of the source code
- Next steps: offload remaining parts of FSBM and reduce host-device data transfers

ACKNOWLEDGMENTS

This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

National Energy Research Scientific Computing Center