OVERVIEW

HOW CAN ONE PORT A METEOROLOGICAL APPLICATION ON GPUs?
10 Minute Bird’s Eye View

- Designing the port/optimization
- Tools to help guide the process
- Complexity of the process
- Developers perspective

Thanks:
Richard Loft, Director of TDD, NCAR
Supreeth Suresh, Software Engineer, NCAR
Students of University of Wyoming
PHASE 1: ASSESS GPU SUITABILITY

- Code Review: Markers
  - F77 code snippets
  - Creation of data on CPU
  - Halo exchange calls
  - Where is the parallelism?

- CPU Execution Profiling
  - Execution time
    - Dycore, Halo and Physics
  - Source code line count

- Outcome
  - Use OpenACC to port
  - Order of porting- Dycore, Haloexchange and Physics
  - Categorize CPU routines and GPU routines
  - Prepare testcases/benchmarks
PHASE 2: DESIGNING THE DIRECTIVES

- **OpenACC directives - Code**
  - Kernel directives for automatic parallelization
  - Parallel directives for user control and efficient parallelization

- **OpenACC directives - Data**
  - PGI compiler lists the variables needed to be copied/created
  - Module variables - declare create
  - Local variables - create
  - MPAS variables
    - CPU variables and respective GPU copies are created simultaneously

- **Halo exchange directives**
  - Send/Recv buffers & MPI book keeping on GPUs
  - GPU-GPU MPI

---

2346, Generating data copyin(rho_zz(;;),rtheta_pp(;;),rtheta_pp_old(;;),ru_p(;;),rdzw(;;),rw(;;),w(;;),zz(;;),zux(;;),alpha_tri(;;),cofwz(;;),cqu(;;),edgesoncell_sign(;;),invareacell(;;),tend_rt(;;),wwavg(;;),tend_rho(;;),rw_save(;;),a_tri(;;),cellsonedge(;;),cofrz(;;),coftz(;;),cofwr(;;),cofwt(;;),dcedge(;;),dss(;;),dvedge(;;),edgesoncell(;;),exner(;;),fzm(;;),fzp(;;),gamma_tri(;;),invdedge(;;),edgesoncell(;;),rho_pp(;;),ruavg(;;),rw_p(;;),tend_ru(;;),trend_rw(;;),theta_m(;;))
PHASE 3 : PORTING & OPTIMIZATION

```
$acc data copy(w_tend, &
$acc edgesOnCell, edgesOnCell_sign, fzm, fzp, nedgesOnCell, u_tend, &
$acc zb3_cell, zb_cell, zz)
$acc parallel num_workers(8) vector_length(32)
$acc loop gang worker private(iEdge, flux)
do iCell=cellSolveStart, cellSolveEnd
  do i=1, nEdgesOnCell(iCell)
    iEdge = edgesOnCell(i, iCell)
    !$DIR$ IVDEP
    do k = 2, nVertLevels
      flux = edgesOnCell_sign(i, iCell) * fzm(k) * u_tend(k, iEdge)
      w_tend(k, iCell) = w_tend(k, iCell) - zb_cell(k, i, iCell)
    end do
  end do
end do

$acc end kernel
$acc end data
```
PHASE 4: TOOLS TO DEBUG, VERIFY & VALIDATE

• KGen Tool
  • Code cutter by NCAR
  • Verifies PGI compiled output for each kernel- CPU or GPU
  • Helps verify OpenACC directives for a small kernel

• PGI Compiler and Profiler
  • Compiler generated or profiler output
    • Guides optimization on GPUs
  • Indicates warnings/issues
  • Helps debug performance issues

• PGI Compiler Assisted Software Testing (PCAST)
  • Using CPU execution as reference, compares GPU (or CPU) results
  • Any variable and any location- but needs host updates, hence slow
  • Helps in code integration

• MPAS Validation Tool
  • Developed by MMM, checks if the “Science” is right!
  • Helps validate the final output

<table>
<thead>
<tr>
<th>SLOW and WRONG</th>
<th>FAST and RIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAST and WRONG</td>
<td>CPU and RIGHT</td>
</tr>
</tbody>
</table>
DEVELOPER’S PERSPECTIVE

Baseline

Porting

Optimize

Integrate

Software & Architecture

Configuration & Accuracy

Benchmark

Verification

KGen

OpenACC Directives

Profile & Analyze

Testing

Code Refactoring

Redo or Move on to Next Code

>10% compromise?
THANK YOU! QUESTIONS?