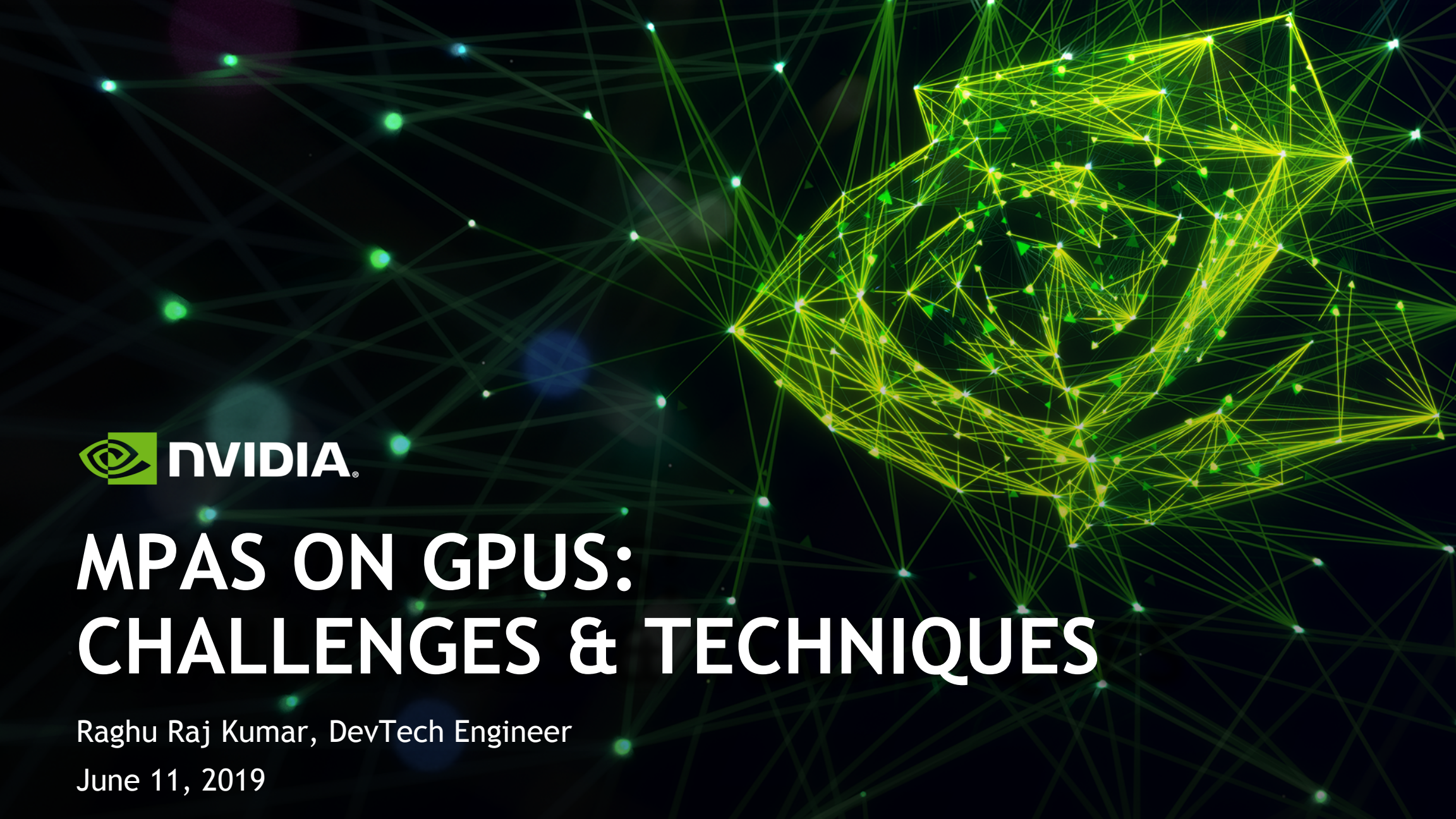


MPAS ON GPUS: CHALLENGES & TECHNIQUES

Raghu Raj Kumar, DevTech Engineer

June 11, 2019

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

```
DO 320 I=its,ite
      IF(BR(I).LT.0.)GOTO 310
      IF(BR(I).LT.0.2)GOTO 270
      REGIME(I)=1
      PSIM(I)=-10.*GZ1OZ0(I)
      PSIM(I)=AMAX1(PSIM(I),-10.)
      IF(UST(I).LT.0.01)THEN
         RMOL(I)=BR(I)*GZ1OZ0(I) !ZA/L
      ELSE
         RMOL(I)=KARMAN*GOVRTH(I)*ZA(I)*MOL(I)/(UST(I)*UST(I)) !ZA/L
      ENDIF
      RMOL(I)=AMIN1(RMOL(I),9.999) ! ZA/L
      RMOL(I) = RMOL(I)/ZA(I) !1.0/L

      GOTO 320
```

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(:,:),zxu(:,:),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(:,:),invdcedge(:),nedgesoncell(:),rho_pp(:,:),ruavg(:,:),rw_p(:,:),tend_
ru(:,:),tend_rw(:,:),theta_m(:,:))
```

PHASE 3 : PORTING & OPTIMIZATION

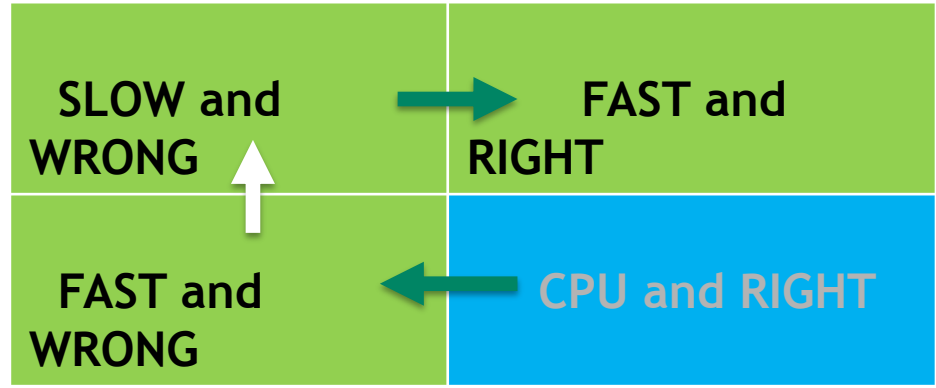
```
do iCell=cellSolveStart,cellSolveEnd
   do i=1,nEdgesOnCell(iCell)
      iEdge = edgesOnCell(i,iCell)
      !DIR$ IVDEP
      do k = 2, nVertLevels
         flux = edgesOnCell_sign(i,iCell) * fzm(k) * u_tend(k,iEdge)
         w_tend(k,iCell) = w_tend(k,iCell) - zb_cell(k,i,iCell)
      end do
   end do
   !DIR$ IVDEP
   do k = 2, nVertLevels
      w_tend(k,iCell) = ( fzm(k) * zz(k,iCell) + fzp(k) * zz(k-1,iCell))
   end do
end do
```

```
!$acc data copy(w_tend, &
!$acc edgesoncell, edgesoncell_sign, fzm, fzp,nedgesoncell, u_tend, &
!$acc zb3_cell, zb_cell, zz)
!$acc kernel
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

   !DIR$ IVDEP
   do k = 2, nVertLevels
      w_tend(k,iCell) = ( fzm(k) * zz(k,iCell) + fzp(k) * zz(k-1,iCell))
   end do
end do
!$acc end kernel
!$acc end data
```

```
!$acc data present(w_tend, &
!$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

!DIR$ IVDEP
   do k = 2, nVertLevels
      w_tend(k,iCell) = ( fzm(k) * zz(k,iCell) + fzp(k) * zz(k-1,iCell))
   end do
end do
!$acc end parallel
!$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

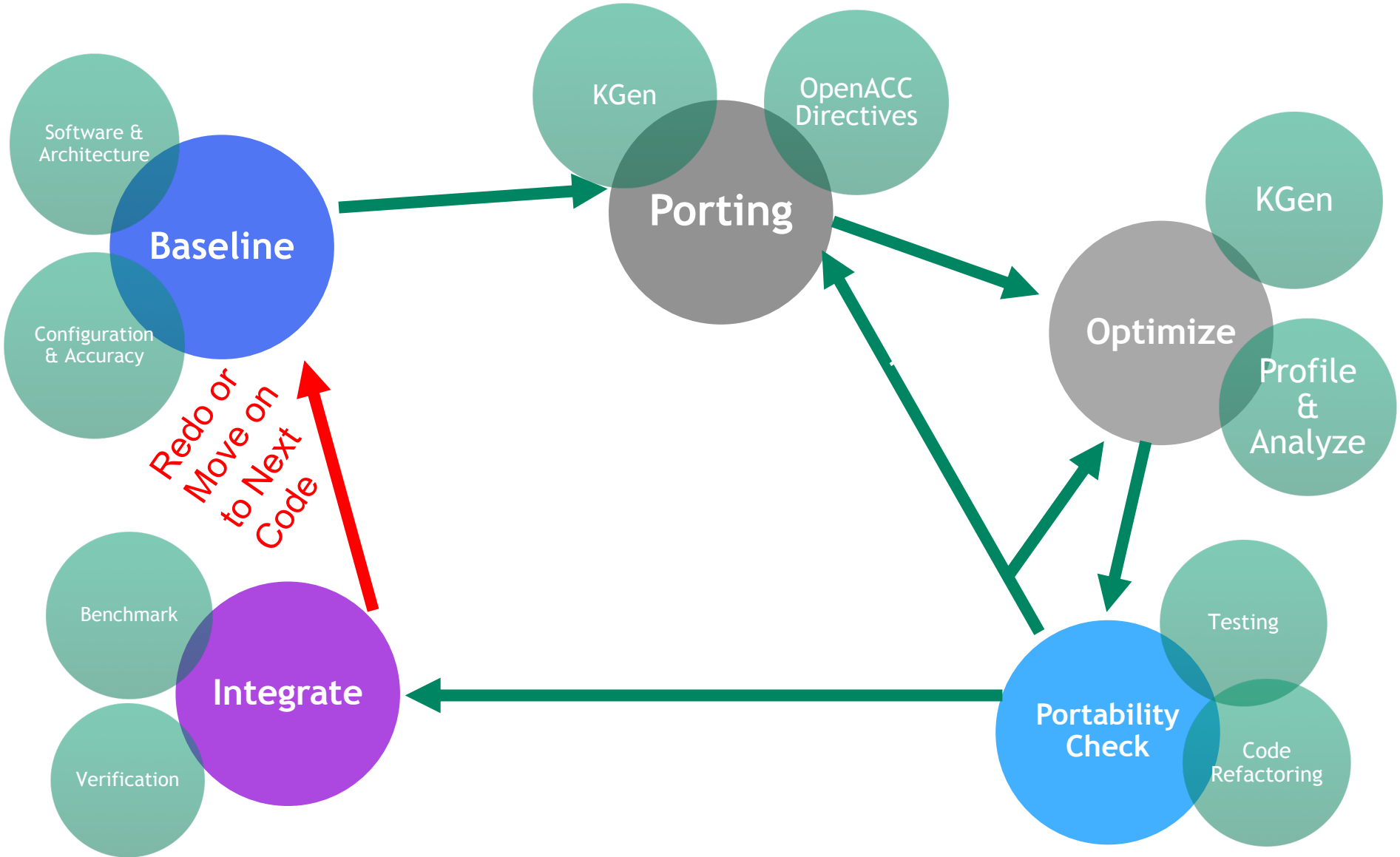


- PGI Compiler and Profiler
  - Compiler generated or profiler output
    - Guides optimization on GPUs
  - Indicates warnings/issues
  - Helps debug performance issues

- 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

DEVELOPER'S PERSPECTIVE




THANK YOU!
QUESTIONS?
NVIDIA