GPU Acceleration of the WRF Model with OpenACC

Carl Ponder, cponder@nvidia.com
Stan Posey, sposey@nvidia.com
NVIDIA, Santa Clara, CA, USA

Various Models with OpenACC Collaborations

Model	Domain	GPU Approach	Collaborators
WRF	NWP/Climate	OpenACC (OACC)	NCAR, Cray, NVIDIA
COSMO	NWP/Climate	CUDA C, OACC	CSCS, SCS, NVIDIA
CAM-SE	Climate	CUDA Ftn, OACC (future)	ORNL, Cray, NVIDIA
NIM/FIM	NWP/Climate	Dirs, OACC	NOAA, NVIDIA
GEOS-5	Climate	CUDA Ftn, PGI Dirs	NASA, NVIDIA
NEMO	Ocean Model	OACC	NVIDIA, STFC (future)

GPU Status of WRF Developments

- **Several non-trunk efforts at various stages:**
  - Dynamics and some physics by Thomas Nipen at UBC – source at NVIDIA
  - KernelGen project: www.kernelgen.org update at NCAR 2012 workshop
  - Cray and OpenACC (Pete Johnsen) with results at 2012 NCAR workshop
  - C-DAC and HPC-FTE group working with NVIDIA India (Priyanka)
  - Shortwave radiation model by NV software group and PGI (G. Ruetsch)
  - Physics kernels by John Michalakes: www.mmm.ucar.edu/wrf/WG2/GPU/
  - NIM to include WRF physics using PGI and/or HMPP, OpenACC
  - Several physics schemes by Space Science and Engineering Center, WI, USA
- **Trunk efforts at various stages:**
  - WSM5 physics model (15% - 25%) in release 3.2 from 2009
  - WRF 3.5 with OpenACC –NVIDIA and NCAR (MMM – Dave Gill) collaboration

OpenACC Developments for WRF 3.4/3.5

Programming weather, climate, and earth-system models on heterogeneous multi-core platforms

September 12-13, 2012 at the National Center for Atmospheric Research in Boulder, Colorado

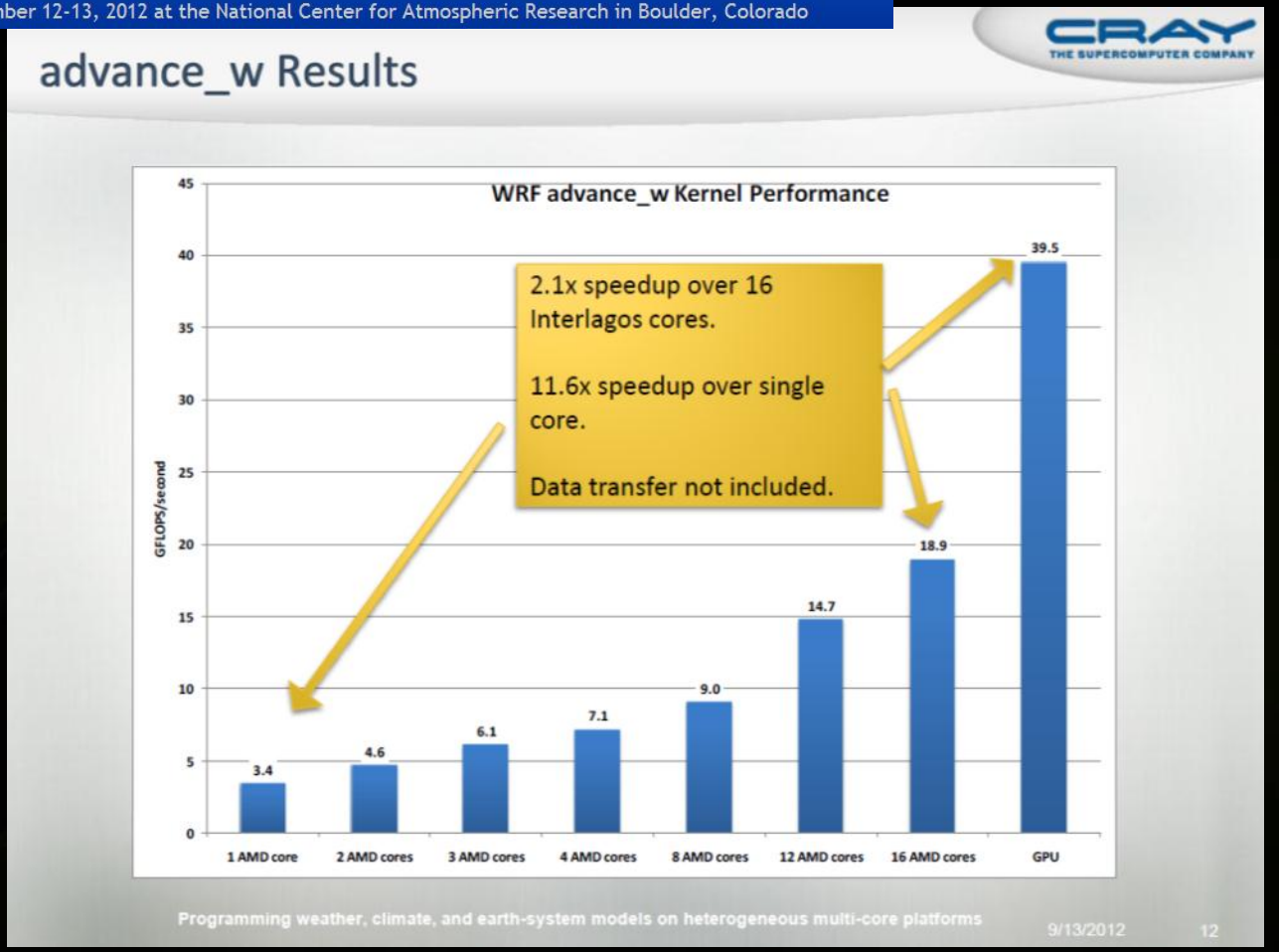
WRF Experiments on GPU Accelerators using OpenACC

- Pete Johnsen, Cray, Inc.

WRF routine `advance_w`

- Dynamics routine to advance vertical velocity
- Standard Fortran use with OpenACC directives
- 2.1x speedup for 16 cores

Source: http://data1.gfdl.noaa.gov/multi-core/



FIM and NIM

Running the FIM and NIM Weather Models on GPUs

- Mark Govett (NOAA Earth System Research Laboratory)

Source: http://on-demand.gputechconf.com/gtc/2013/presentations/S3429-FIM-NIM-Weather-Models-on-GPUs.pdf

FIM Performance – Single Socket

- No changes to the FIM source code
- GPU timings used F2C-ACC compiler
- Optimized for Fermi GPU, further optimizations for Kepler needed
  - Code changes need to improve hybgen performance (currently >50% of GPU runtime)

FIM Dynamics Routines	NVIDIA Fermi GPU 1 socket	Intel CPU SandyBridge 1 socket	Intel Xeon Phi – KNC 1 socket	NVIDIA Kepler GPU Early Results
trcadv	1.28	2.10	1.09 (1.9)	0.99 (2.0)
cnuity	1.04	1.13	0.49 (2.3)	0.68 (1.8)
momtum	0.41	0.71	0.34 (2.1)	0.35 (2.1)
hybgen	4.13	4.09	3.10 (1.3)	3.43 (1.2)
TOTAL	8.01	8.87	5.86 (1.5)	6.30 (1.4)

NIM Serial Performance (2013)

- No changes to the source code
- Single Socket Performance
  - 10K horizontal points, 96 vertical levels
- Very efficient CPU performance
  - Measured 29% of peak performance (Intel Westmere)

NIM	Opteron	Westmere	SandyBridge	Fermi	K20x
runtime	143.0	86.8	60.0	25.0	20.7

- Parallel performance
  - Being run on up to 160 GPUs
  - Working on optimizing inter-GPU communications

NIM Parallel Performance

- Weak Scaling with Communications Optimization
  - Moved collective operation to the CPU instead of doing it on the GPU using GPU MappedMemory
    - Too many small writes across the PCIe bus
  - Resulted in a 5-17x speedup for the Pack Operation

GPUs	GPU to GPU Comm Time	Total Time Time (sec)
10	232 (22%)	1034
40	247 (23%)	1054
160	266 (24%)	1076

Operation	Time
Iniitilalization	3 (1%)
Pack Data on GPU	45 (17%)
CPU – GPU Copy	59 (22%)
MPI Comms	82 (31%)
UnPack on GPU	77 (29%)
Total	266

COSMO

Towards GPU-accelerated Operational Weather Forecasting

- Oliver Fuhrer (MeteoSwiss)

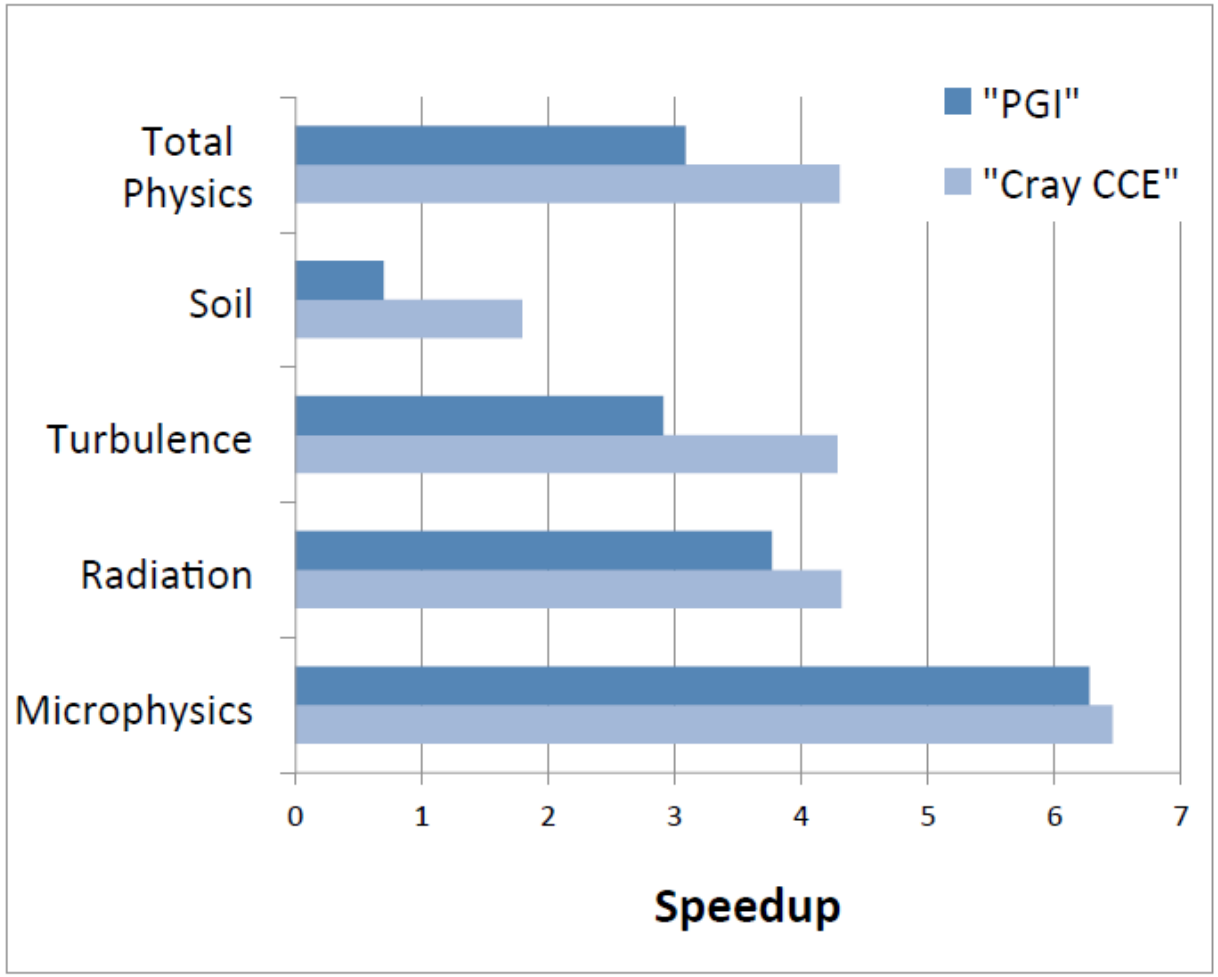
Source: http://on-demand.gputechconf.com/gtc/2013/presentations/S3417-GPU-Accelerated-Operational-Weather-Forecasting.pdf

Requirements

- Performance portability
- Single source code
- Code understandable / maintainable by domain scientists
- Possibility to exchange code

Performance: Physics

- AMD Interlagos vs. Tesla K20x
- Test: 128x128x60
- 90% of physics time in turbulence, radiation and microphysics
- Speedup scales with effort

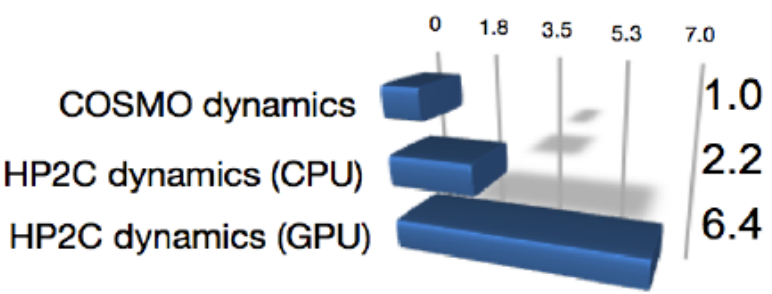


Performance: Dynamics

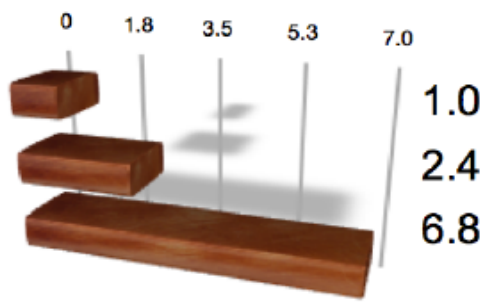
A single switch in order to compile for GPU

- Test domain: 128 x 128 x 60 on a single CPU / GPU
- **CPU** (OpenMP, kji-storage)
  - Factor 1.6x – 1.8x faster than Fortran version
  - No explicit use of vector instructions (up to 30% improvement)
- **GPU** (CUDA, ijk-storage)
  - Factor 2.8x faster than CPU version
  - Ongoing performance optimization

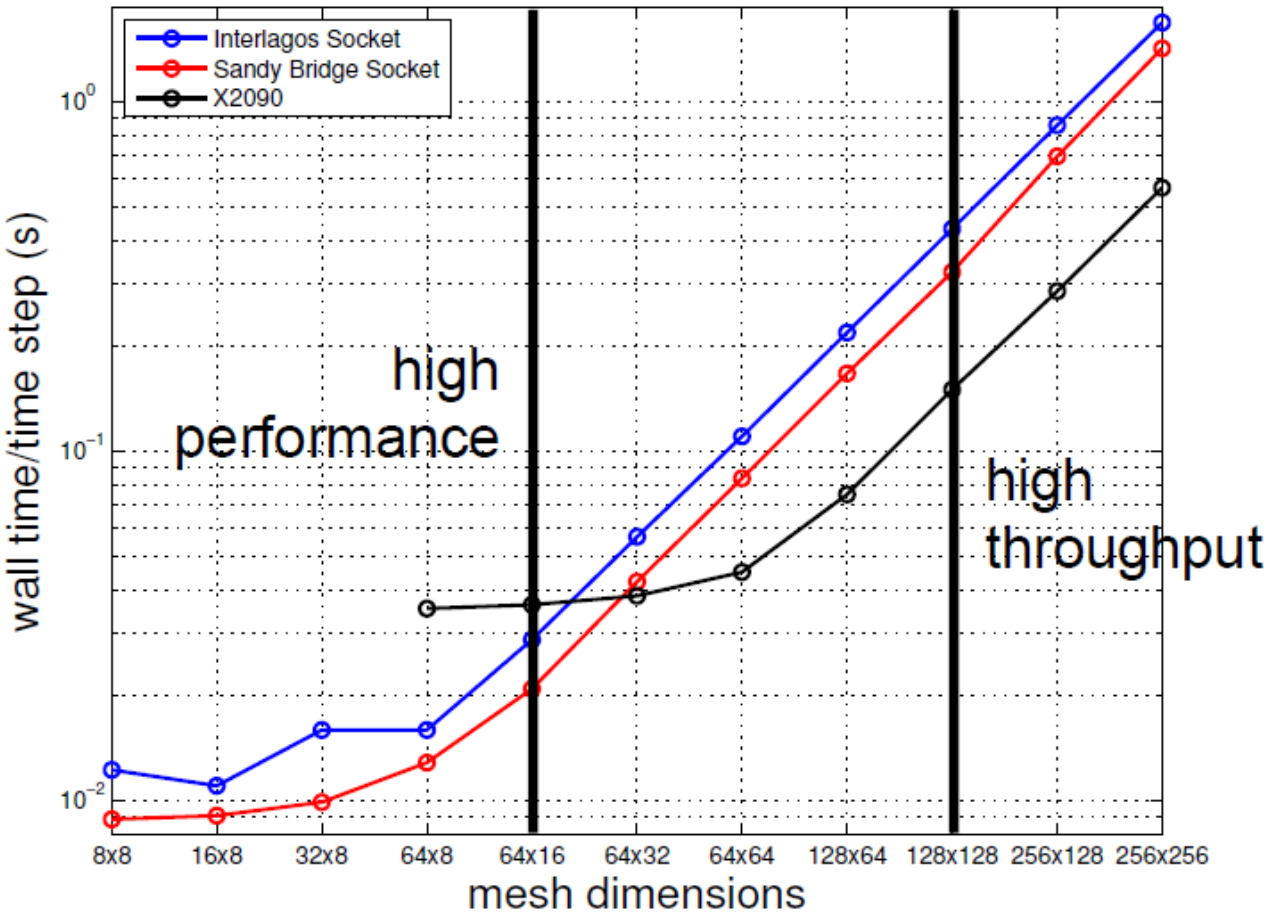
Interlagos vs. Fermi (M2090)



SandyBridge vs. Kepler



Optimal hardware for different use cases



OpenACC Experience

- Retain existing user code (e.g. Fortran)
- Relatively easy to get code running & validating
- Useful for large code bases

- Performance may require significant restructuring → single source?
- Data placement can be tricky
- Compiler support still improving → thanks for the help!
- No fine grain control (e.g. data placement, register allocation)

NEMO

Accelerating NEMO with OpenACC

- Maxim Milakov (NVIDIA)

Source: http://on-demand.gputechconf.com/gtc/2013/presentations/S3209-Accelerating-NEMO-with-OpenACC.pdf

Kernels directive

```
!$acc kernels
```
<- Implicit *present_or_copy* clause

```
! 3. monotonic flux in the i & j direction (paa & pbb)
! ---------------------------------------
DO jk = 1, jpkm1
   DO jj = 2, jpjm1
      DO ji = fs_2, fs_jpim1   ! vector opt.
         zau = MIN( 1.e0, zbetdo(ji,jj,jk), zbetup(ji+1,jj,jk) )
         zbu = MIN( 1.e0, zbetup(ji,jj,jk), zbetdo(ji+1,jj,jk) )
         zcu =       ( 0.5  + SIGN( 0.5 , paa(ji,jj,jk) ) )
         paa(ji,jj,jk) = paa(ji,jj,jk) * ( zcu * zau + ( 1.e0 - zcu) * zbu )

         zav = MIN( 1.e0, zbetdo(ji,jj,jk), zbetup(ji,jj+1,jk) )
         zbv = MIN( 1.e0, zbetup(ji,jj,jk), zbetdo(ji,jj+1,jk) )
         zcv =       ( 0.5  + SIGN( 0.5 , pbb(ji,jj,jk) ) )
         pbb(ji,jj,jk) = pbb(ji,jj,jk) * ( zcv * zav + ( 1.e0 - zcv) * zbv )

! monotonic flux in the k direction, i.e. pcc
! -------------------------------------------
         za = MIN( 1., zbetdo(ji,jj,jk+1), zbetup(ji,jj,jk) )
         zb = MIN( 1., zbetup(ji,jj,jk+1), zbetdo(ji,jj,jk) )
         zc =       ( 0.5  + SIGN( 0.5 , pcc(ji,jj,jk+1) ) )
         pcc(ji,jj,jk+1) = pcc(ji,jj,jk+1) * ( zc * za + ( 1.e0 - zc) * zb )
      END DO
   END DO
END DO
!$acc end kernels
CALL lbc_lnk( paa, 'U', -1., pbb, 'V', -1. )   ! lateral boundary condition (changed sign)
```

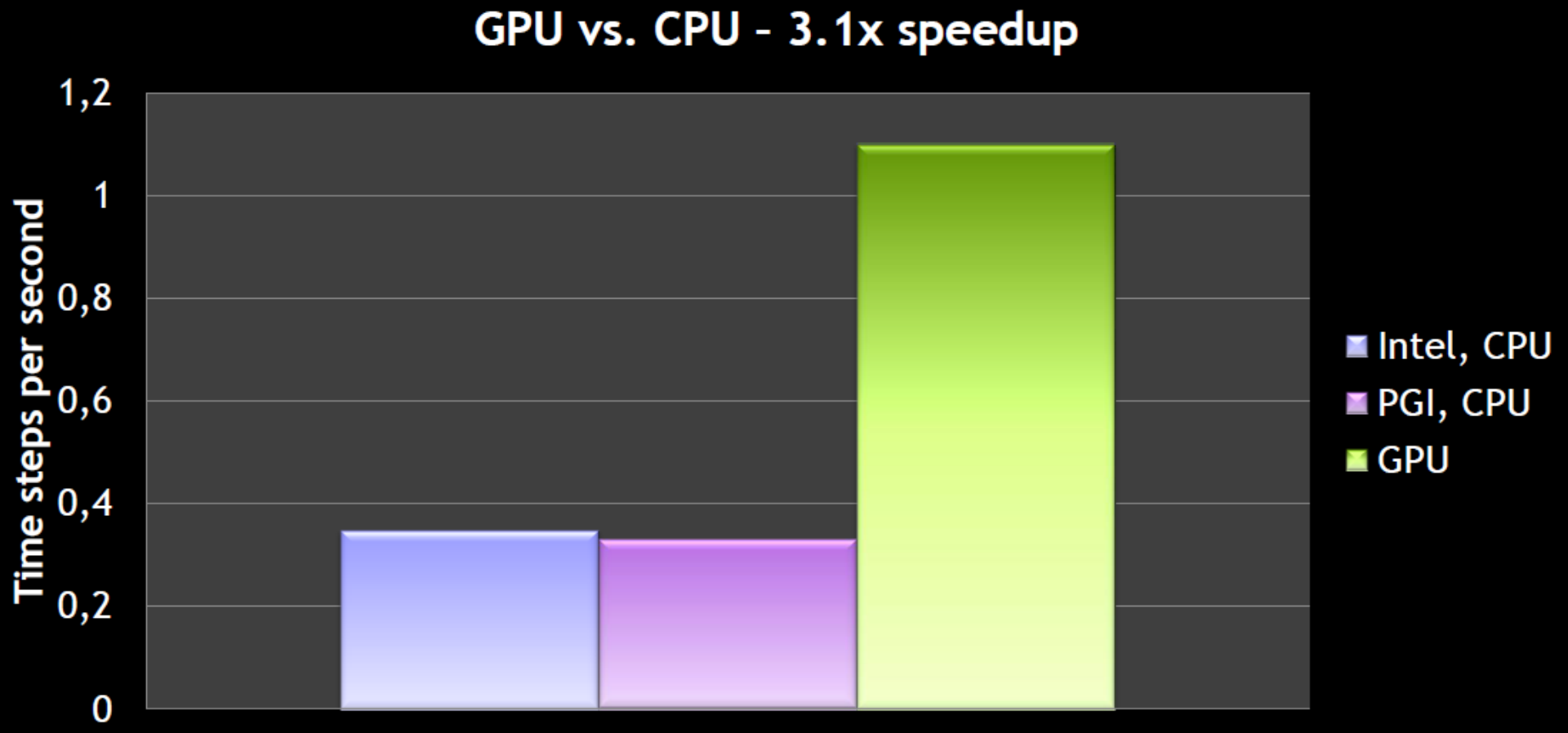
Data directive

```
!$acc data pcopy(umask, vmask, tmask, tmask_i, tsn, un, vn, wn, avt, avmu) &
!$acc pcopy(e1u, e1v, e1t, e1f, e2u, e2v, e2t, e2f, e3u, e3v, e3t, e3f, e3w, e3uw, e3vw) &
!$acc pcopy(fmask, hdivn, hdivb, rotb, rotn, ua, va, ub, vb, ff, mbkt, rhop, hmld) &
!$acc pcopy(spgu, spgv, gcdprc, gcb, gcx, gcxb, sshn, bmask, gcdmat) &
!$acc pcopy(avmv, bfrua, bfrva, mbku, mbkv, utau, utau_b, vtau, vtau_b) &
!$acc pcopy(tsa, tsb, rhd, gdept, gdepw, rn2, gru, grv, hmlp, hmlpt) &
!$acc pcopy(nmln, omlmask, uslpml, vslpml, wslpiml, wslpjml, uslp, vslp, wslpi, wslpj) &
!$acc pcopy(emp, emps, qns, qsr, gphiu, gphiv, gphit, taum, wndm, rn2b) &
!$acc pcopy(qns_b, emp_b, emps_b, fr_i, rnf, hu, hv, gcp, gccd, gcdes, gcr) &
!$acc pcopy(ssu_m, ssv_m, sst_m, sss_m, ssh_m, gtsu, gtsv, ahtu, ahtv, ahtw) &
!$acc pcopy(ssha, sshb, etot3, qsr_hc_b, qsr_hc, ah_wslp2, bfrcoef2d) &
!$acc pcopy(r2dtra, rdttra, dissl, htau, en, avmb, avtb, avtb_2d, avm) &
!$acc pcopy(sbc_tsc, sbc_tsc_b, rnf_tsc, rnf_tsc_b, nk_rnf, h_rnf, rnfmsk)

#if defined key_mpp_mpi
      !$acc data pcreate(t3ew, t3we, t3ns, t3sn, t2ew, t2we, t2ns, t2sn, pt2d_lbcnfd, tr2ew, tr2we, tr2ns, tr2sn)
#endif

      DO WHILE ( istp <= nitend .AND. nstop == 0 )
#if defined key_agrif
         CALL Agrif_Step( stp )           ! AGRIF: time stepping
#else
         CALL stp( istp )                 ! standard time stepping
#endif
         istp = istp + 1
         IF( lk_mpp )   CALL mpp_max( nstop )
      END DO

#if defined key_mpp_mpi
      !$acc end data
#endif

      !$acc end data
```

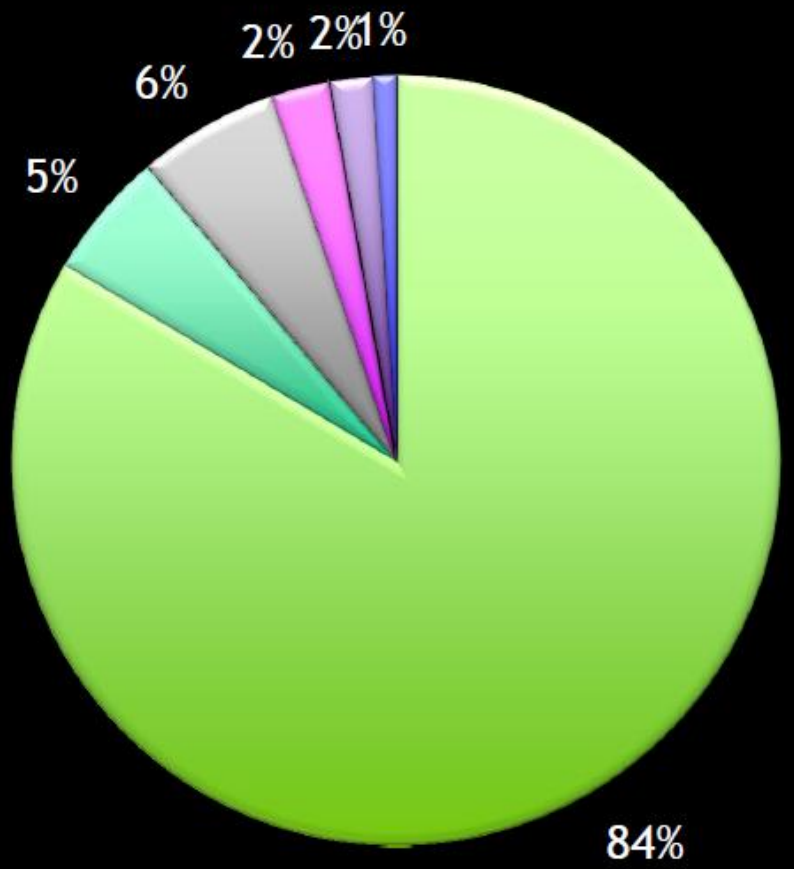
Benchmarking - hardware

- "Sandy Bridge + Kepler" nodes, each having:
  - CPU: 2 sockets * Xeon E5-2670 (Sandybridge), 2.6GHz (3.3GHz Turbo Boost), 8 cores, 64 GB RAM
  - GPU: 2x Tesla K20X, ECC off, 6GB RAM each
  - 4x FDR Infiniband (56 Gb/s)
- Running configuration is GYRE_50 (1/4 degree), requires about 23GB of total RAM, fits 4 K20X
- The code is running on 2 nodes
- The performance is measured by running 1000 time steps, startup and shutdown overheads are not included in figures

Benchmarking - results



Vampir - analysis



Review of CWO Models and GPU Progress

Climate

GEOS-5 (US)
CESM (US)
- CAM-SE (HOMME)
- POP (US)
CFSv2 (US)
- GFS
- MOM4

Weather

WRF (US)
FIM/NIM (US)
COAMPS (US)
COSMO (EU)
ICON (DE)
AROME (FR)
HARMONIE (EU)
- Hirlam + Aladin
ASUCA (JP)
GRAPES (CN)
OLAM (BR)

Ocean

MITgcm (US)
HYCOM (US)
ROMS (US)
MOM4 (US)
GOLD (US)
POP (US)
NEMO (EU)

- Substantial GPU Development
- Some GPU Development
- GPU Evaluation Not Started

Next-Gen CWO Models and GPU Status

Climate

GEOS-5 (US) → GEOS-6 (US)
CESM (US)
- CAM-SE (HOMME)
- POP (US) → MPAS-O
CFSv2 (US)
- GFS
- MOM4 (US) → MOMn

Weather

WRF (US) → MPAS-A or NIM
ICON (DE)
UM (UK) → GungHo
IFS (UK) → OOPS

Ocean

MOM4 (US) → MOMn
POP (US) → MPAS-O

- Substantial GPU Development
- Some GPU Development
- GPU Evaluation Not Started

Questions ?

Carl Ponder, cponder@nvidia.com
Stan Posey, sposey@nvidia.com
NVIDIA, Santa Clara, CA, USA