

# Successes and Challenges using GPUs for Weather and Climate Models



Xbox 360



DOE Jaguar

Mark Govett

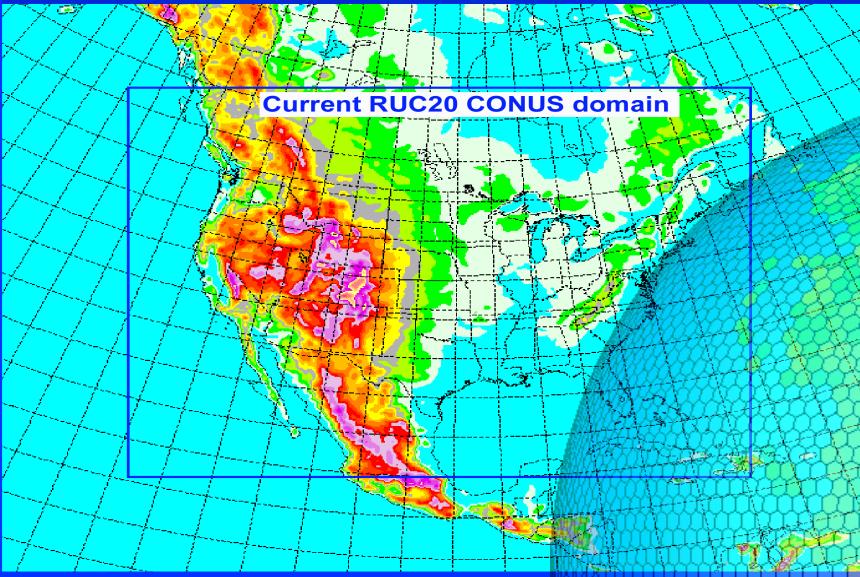
Jacques Middlecoff, Tom Henderson,  
Jim Rosinski, Craig Tierney

- CPU
  - Bigger Systems
  - More Expensive Facilities
  - Bigger Power Bills
  - Lower System Reliability

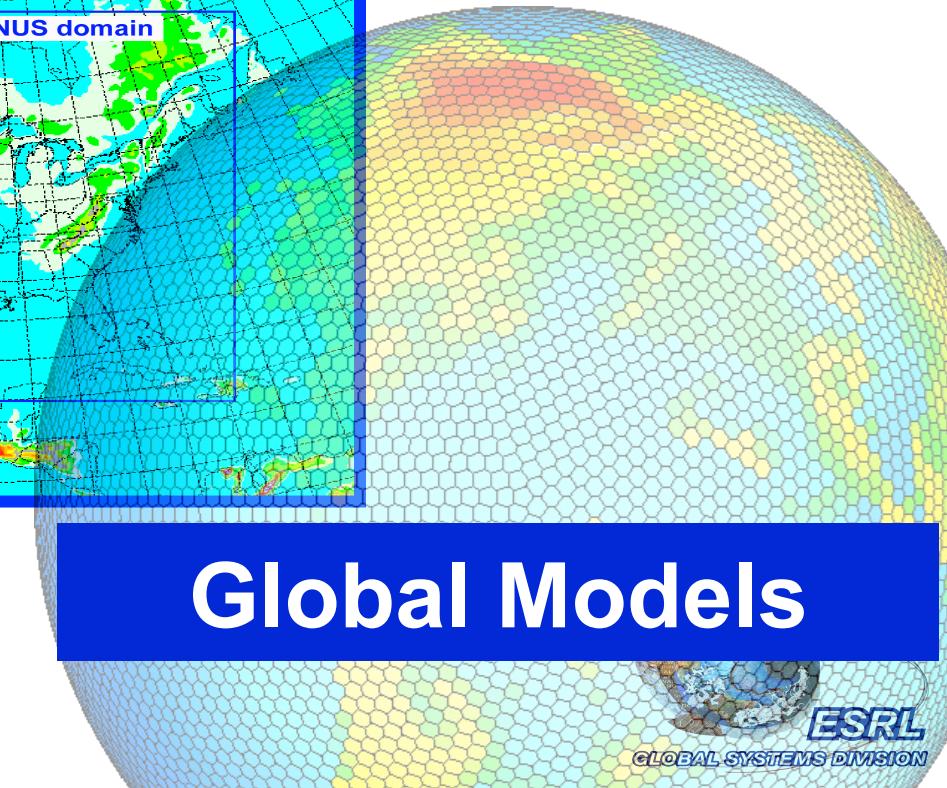


- GPU
  - Faster
  - Less power
  - Lower cost

## Regional Models



## Global Models



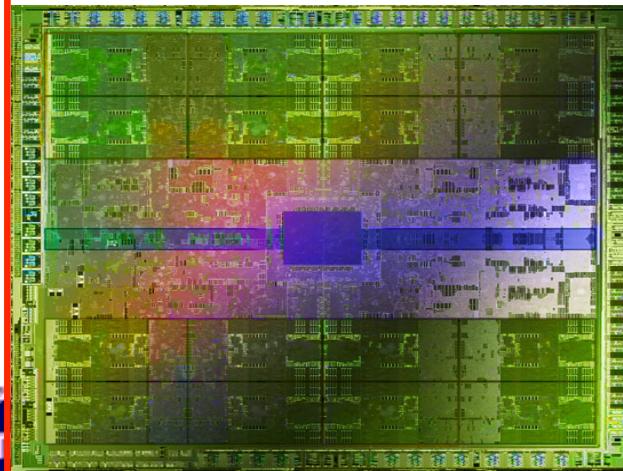
ESRL

GLOBAL SYSTEMS DIVISION

# GPU Technology

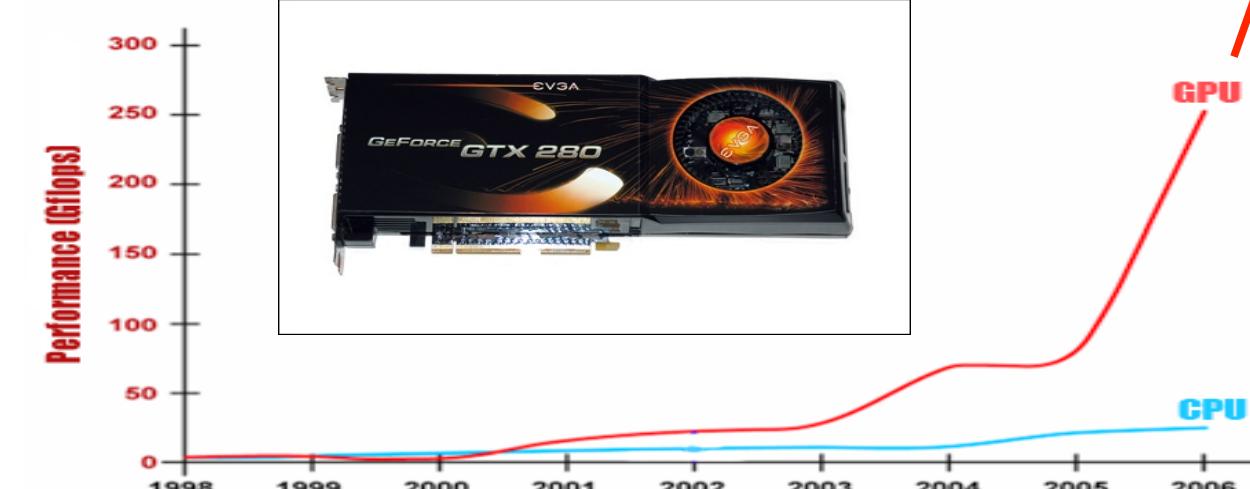
- NVIDIA: Fermi chip first to support HPC
  - Formed partnerships with Cray, IBM on HPC systems
  - #1, #3 systems on TOP500 (Fermi, China)
- AMD/ATI: Primarily graphics currently
  - #7 system on TOP500 (AMD-Radeon, China)
  - Fusion chip in 2011 (5 TeraFlops)
- Intel: Many Integrated Core (2012), 32-64 cores

NVIDIA: Fermi (2010)



- ✧ 1.1 TeraFlops
- ✧ 8x increase in double precision
- ✧ Increase in memory bandwidth
- ✧ Error correcting memory

NVIDIA: Tesla (2008)



GPU: 2008  
933Gflops  
150W

CPU:2008  
~45 Gflops  
160W

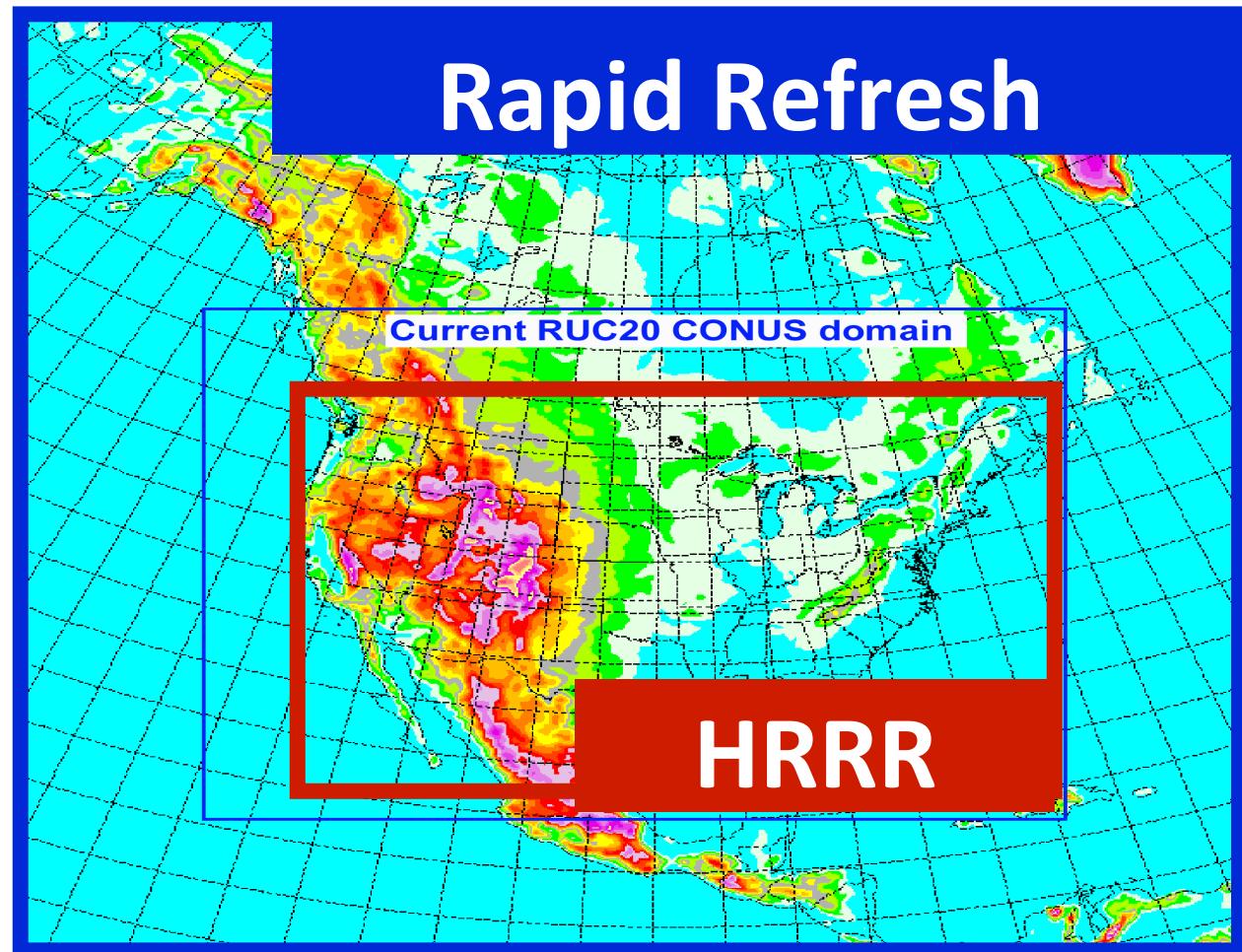
# High-Resolution Rapid Refresh

Real-time, 12 hour forecast, 3-km CONUS domain,  
updated hourly

*Explicit prediction of thunderstorms*

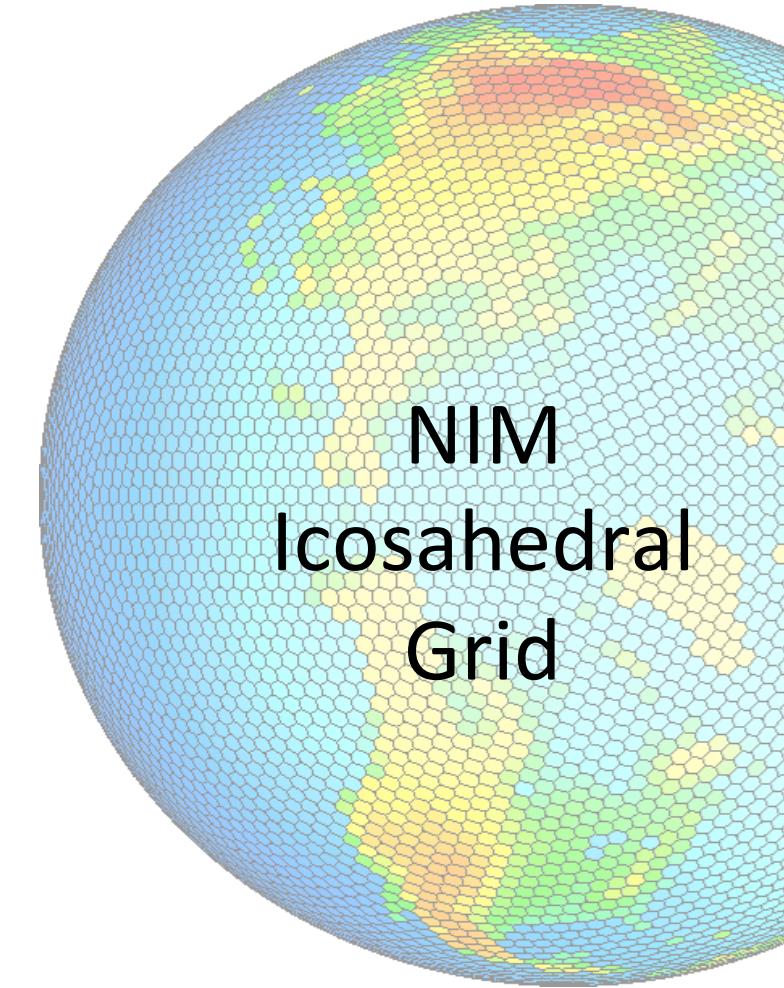
*Improved prediction of terrain  
related and other mesoscale  
features (wind, clouds, precip)*

*HRRR runs as nest within RUC or  
Rapid Refresh and benefits from  
RUC / RR data assimilation*



# Using GPUs for Global Cloud Resolving Models (GCRM)

- Benefits
  - Clouds have a major influence on weather and climate
  - Improvements in 5-100 day forecasts
  - Improved Hurricane track and intensity
- Active developments in the research community
  - NICAM: University of Tokyo
  - GCRM: Colorado State University
  - NIM: NOAA Earth System Research Laboratory
- Non-hydrostatic Icosahedral Model (NIM)
  - Targeting 2KM horizontal resolution
  - Uniform, hexagonal-based, icosahedral grid
  - Novel indirect addressing scheme used that permits concise, efficient code



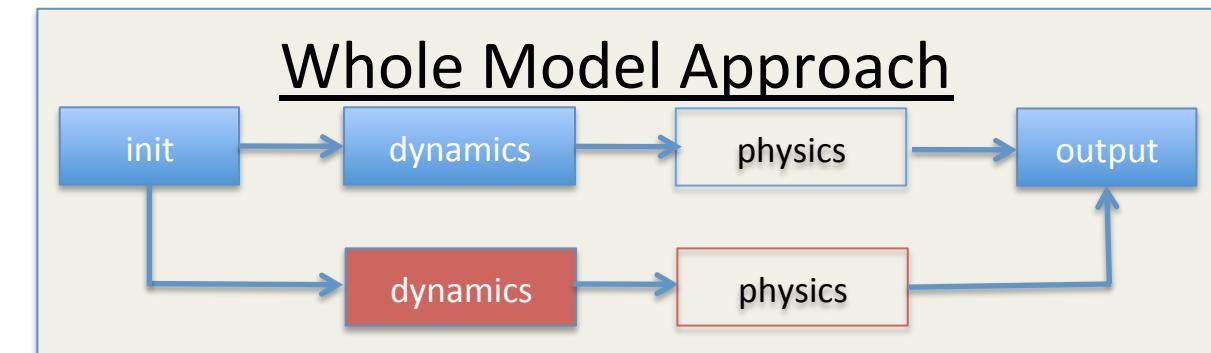
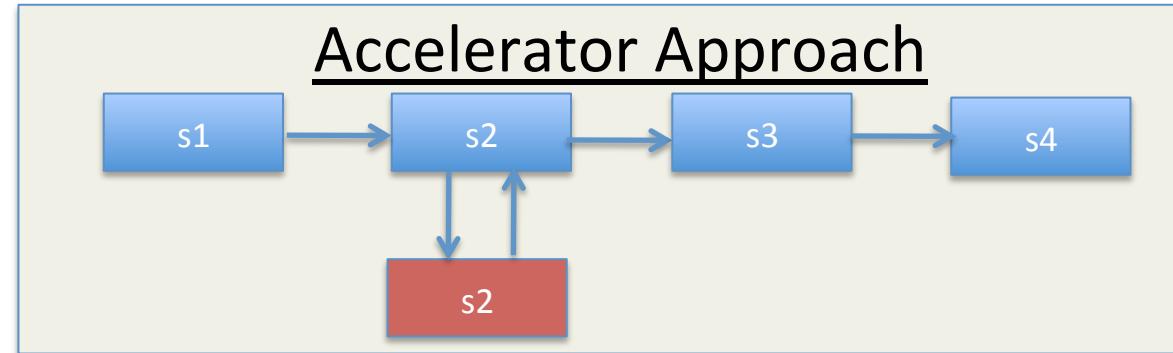
January 18, 2011

ACS Program Review



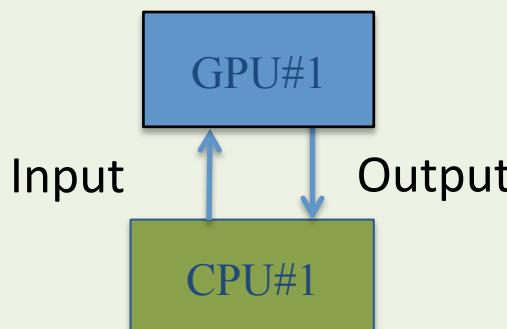
# Software Development

from CPUs to GPUs (2010s)



- NIM was designed for CPU and GPU Architectures
- Code converted to CUDA using the F2C-ACC compiler we developed
  - Commercial compilers were not available in 2008

Single GPU communications



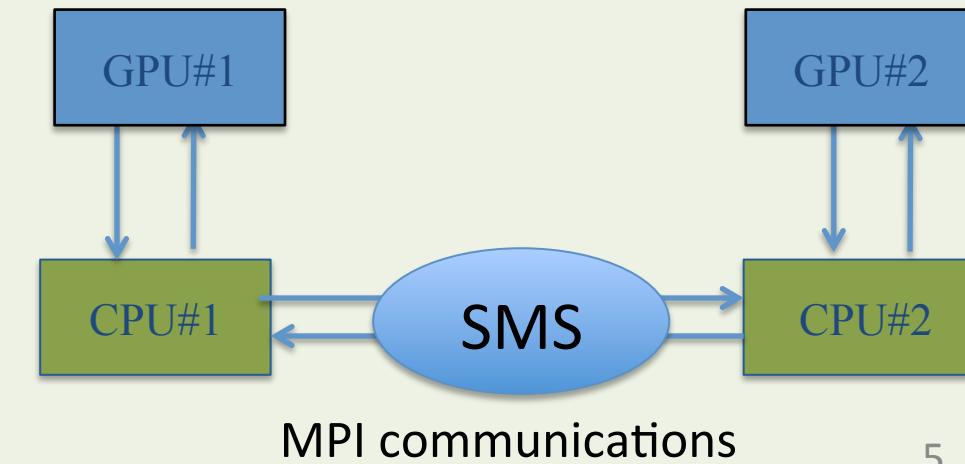
– Serial Performance

- 2009: 34x Tesla / Harpertown
- 2010: 25x Fermi / Nehalem

– Parallel Performance

- 2010: 15x with MPI communications

Multi - GPU communications



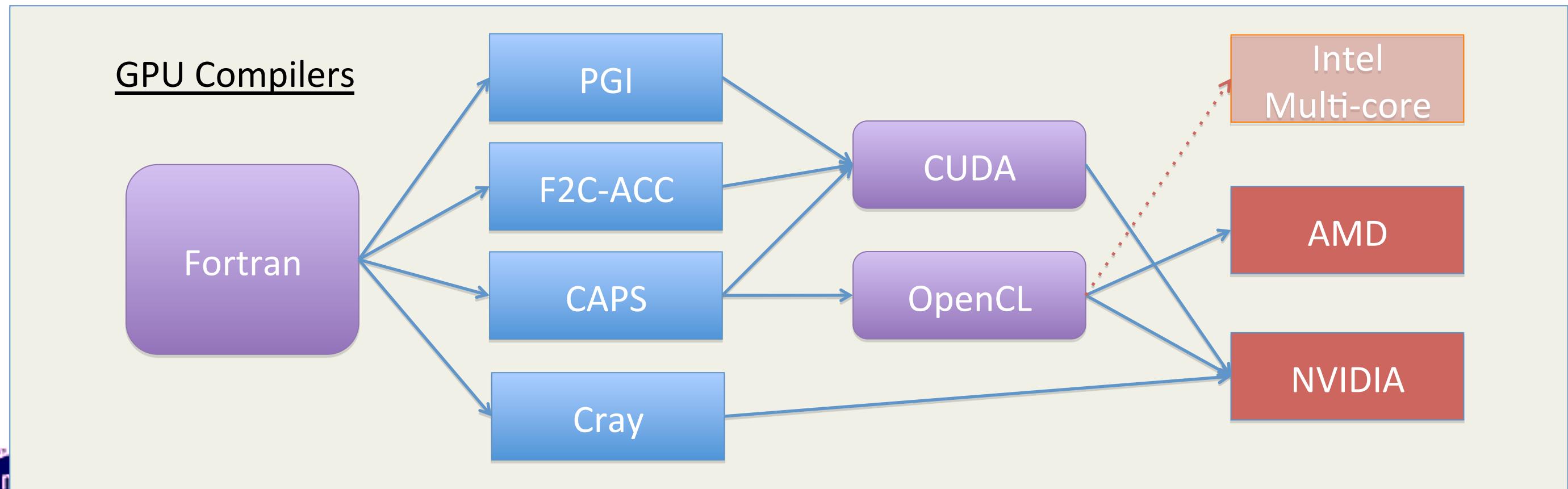
# GPU Programming Approaches

- Language Approach
  - CUDA, OpenCL, CUDA Fortran, etc.
  - User control over coding and optimizations
    - May not be portable across architectures
  - Requires that separate versions be maintained
    - In practice this rarely works – too costly, difficult
- Directive-based Approach
  - Single source for CPU, GPU, serial, parallel
  - Appear as comments in the source
    - !ACC\$DO VECTOR (1)
  - Compilers can analyze and (hopefully) generate efficient code
    - Dependent on maturity



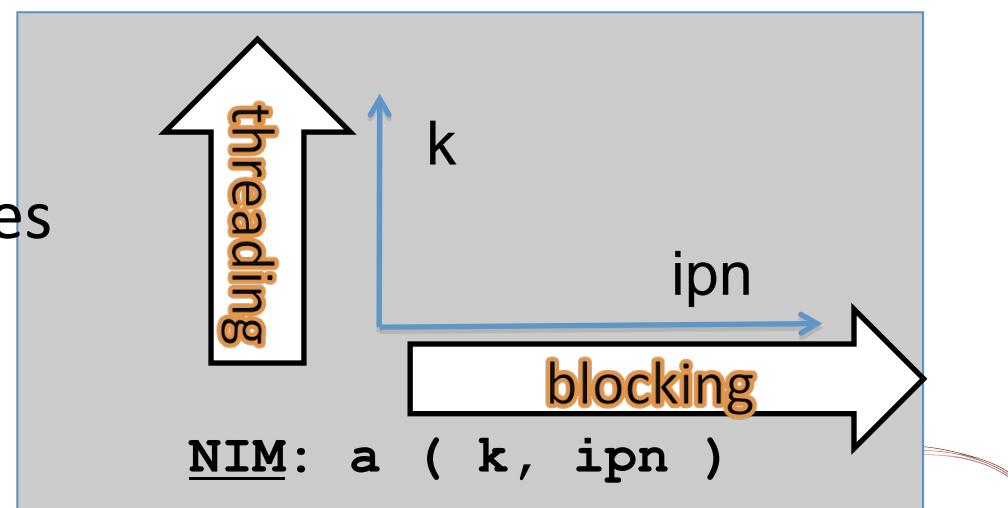
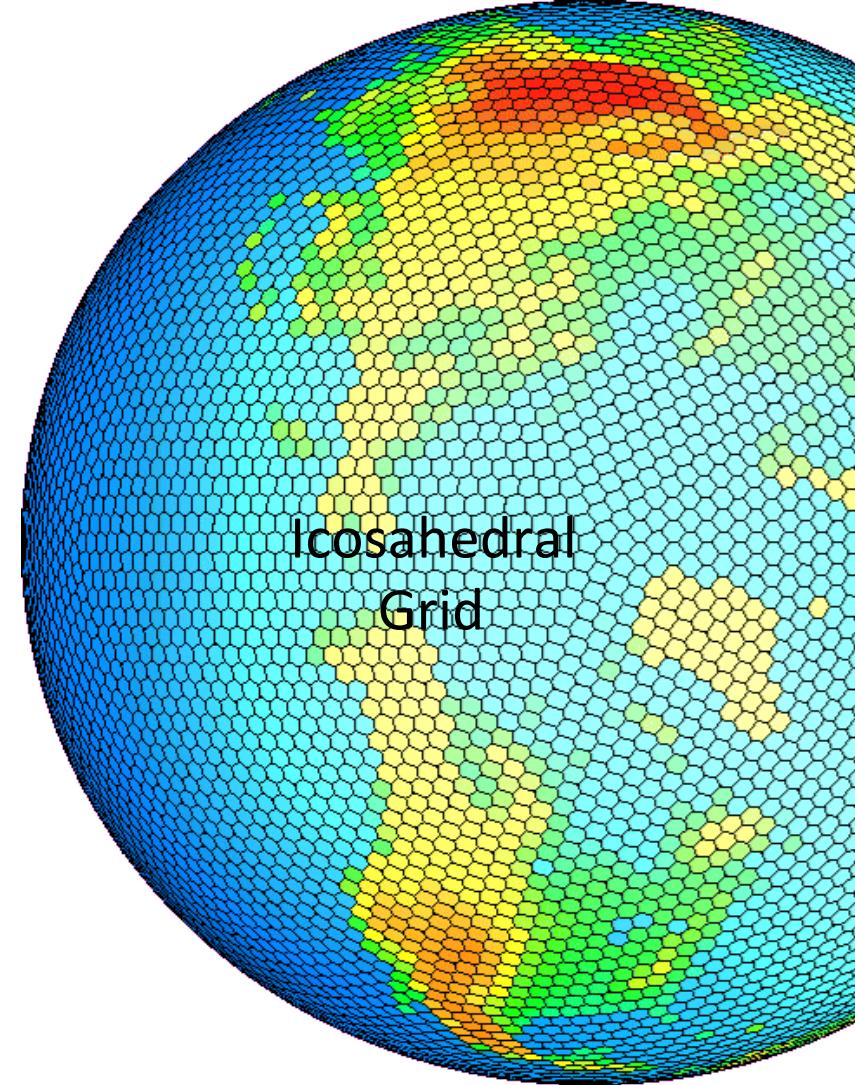
# Directive-Based Fortran GPU Compilers and Portability

- No industry standard for directives (yet)
  - PGI: Accel
  - F2C-ACC: OpenSource
  - CAPS: HMPP
  - Cray: OMP “like”



# NIM Dynamics

- Uniform, hexagonal-based, icosahedral grid
- Novel indirect addressing scheme permits concise, efficient code
- Dynamics is running entirely on GPUs
  - Horizontal data dependencies
  - 2D arrays (vertical, horizontal)
  - GPU threading across the vertical
    - 32, 96 levels increasing to 192 levels at finer scales
  - Physics (scientific) integration in progress



# F2C-ACC GPU Compiler

- Developed to speed parallelization of NIM
  - Commercial compilers were not available in 2008
- Translates Fortran to C or CUDA
  - Many (but not all) language features supported
  - Generates readable, debuggable code with original comments retained
- Ten directives for code parallelization, eg.

– ACC\$REGION	! Define GPU regions
– ACC\$DO	! Identify loop level parallelism
– ACC\$DATA	! Move data between CPU and GPU
– ACC\$INSERT, ACC\$REMOVE	! Hand insertions / deletions where translation is not available

- Available on request



# Fortran GPU Compiler Results (2011)

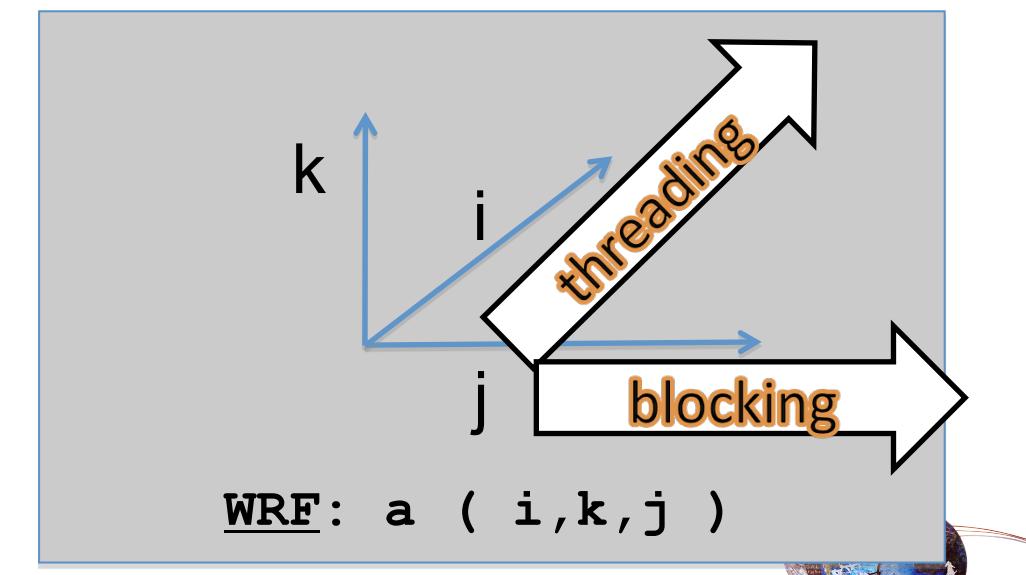
Using NIM G5 - 10242 horizontal points, 96 vertical levels  
Fermi GPU vs. Intel Westmere CPU Socket

NIM routine	CPU 1-core Time (sec)	CPU 6-core Time (sec)	F2C-ACC GPU Time (sec)	HMPP GPU Time (sec)	PGI GPU Time (sec)	F2C-ACC Speedup vs. 6-core CPU
Total	8654	2068	449	--	--	4.6
vdmints	4559	1062	196	192	197	5.4
vdmintv	2119	446	91	101	88	4.9
flux	964	175	26	24	26	6.7
vdn	131	86	18	17	18	4.8
diag	389	74	42	33	--	1.8
force	80	33	7	11	13	4.7



# WRF Physics

- Community Model used worldwide for more than a decade
  - Significant number of collaborators, contributors
- Used in WRF-ARW, WRF-NMM, WRF-RR, WRF-CHEM, HWRF, etc.
- Traditional cartesian grid
  - 3D arrays (horizontal, vertical, horizontal)  $\Rightarrow$  `array3D( i, k, j )`
- Designed for CPU architectures
- **Limited ability to change the code**
  - **Must continue to be performance portable**
- GPU parallelization
  - In progress – select routines
  - Dependencies in vertical
  - GPU: threading in horizontal dimensions
  - Collapse i, j dimensions during transpose

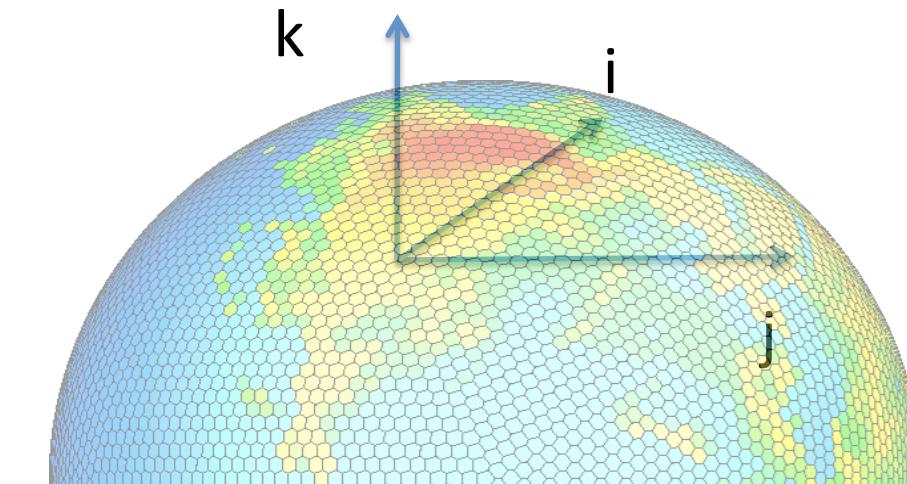


# Parallelization Factors for NIM

- Code design a dominant factor in performance
  - Weather codes typically have a high memory access to compute ratio
    - Implies lots of accesses, few computations
  - Data alignment led to a 10x improvement
- Data dependencies guide parallelization
  - Dynamics are in the horizontal
    - $a$  [ vert, horiz ]
  - Physics are in the vertical column
    - $a$  [ horiz, vert ]
  - Transpose needed to optimize memory accesses

lat-lon  $a$  (  $k$ ,  $i$ ,  $j$  )

NIM:  $a$  [  $k$ ,  $\text{indx}$  )



# Successes

- Parallelization of NIM
  - 5x speedup of NIM dynamics (socket-to-socket)
  - F2C-ACC continues to be used for NIM
- Development of F2C-ACC
  - **Useful for comparisons to commercial compilers**
    - Establish performance benchmarks
    - Ease of use: readability of generated code
    - Directives that support our weather, climate codes
    - Validate correctness of results
  - Feedback to compiler vendors
    - Communicate needs in the weather and climate community



# Challenges

- Validating results
  - dependent on the computer architecture
    - CPU, GPU, Intel, IBM, NVIDIA, AMDm etyc
  - Physics is more sensitive than dynamics
  - How do you determine acceptable results?
- Performance portability
  - Modest to extensive code changes
  - Promotion of variables for correctness
    - Demotion of variables for performance
  - Loop restructuring
  - Blocking and threading control



# Challenges

- Data management
  - How to work with high data volume
    - 15 KM, 2M horizontal points, 96 vertical
      - 2GB per variable per output time
    - 1.75KM, 167M horizontal points, 192 vertical
      - Projected 64GB per variable per output time
- Visualization
  - Exploring using gaming software
    - GPUs, progressive disclosure

**UNITY 3D**  
*game development tool*



# Conclusion

- Committed to a single source
  - Performance portable between CPU, GPU, serial, parallel
    - NVIDIA, AMD, Intel, etc
  - We anticipate significant challenges for legacy codes
- We will continue to compare compilers
  - F2C-ACC, HMPP, and PGI Accel
    - Performance, ease-of-use
- Challenges Remain
  - Codes take too long to port to GPUs
  - Performance portability a concern
  - Standards for GPU directives

