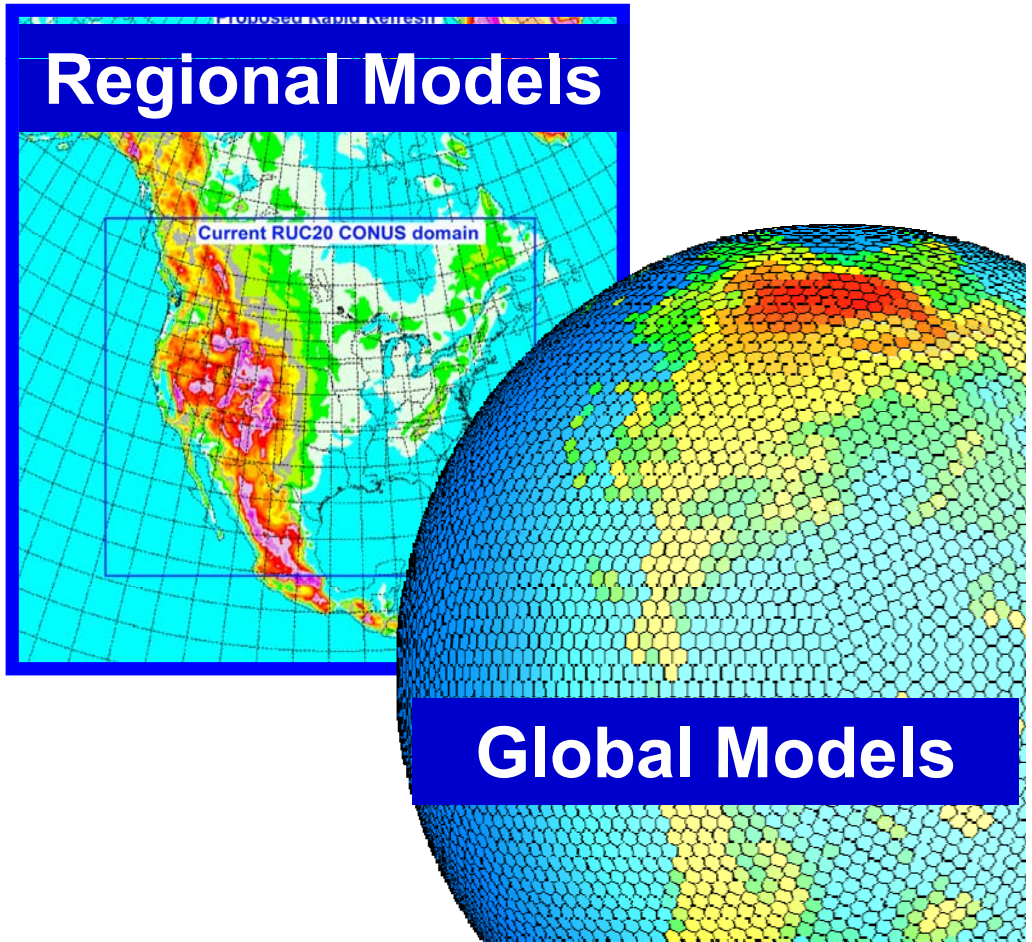
Parallelization of the NIM Dynamical Core for GPUs


Xbox 360

Mark Govett
Jacques Middlecoff, Tom Henderson,
Jim Rosinski, Craig Tierney

DOE Jaguar

- CPU
  - Bigger Systems
  - More Expensive Facilities
  - Bigger Power Bills
  - Lower System Reliability
- GPU
  - Faster
  - Less power
  - Lower cost



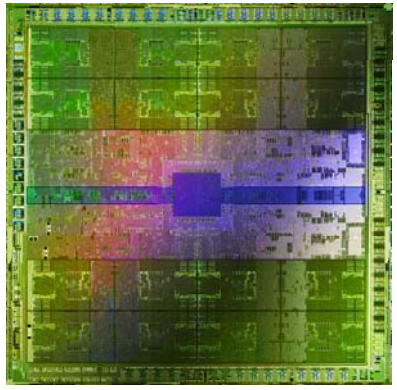
GPU & MIC Hardware

- NVIDIA: Fermi chip first to support HPC
  - Tesla (2008) 240 cores, 933 SP GFlops, CUDA
  - Fermi (2010) 512 cores, 662 DP Gflops, ECC memory
  - Kepler (2012) 28nm - - > 3-4x performance / watt
  - Maxwell (2014?) Integrated CPU & GPU
- AMD/ATI: Graphics chip primarily
  - Fusion (2011): integrates CPU & GPU
- Intel MIC: Many Integrated Core (2012), 32-64 cores
  - Large memory, cache
  - x86 => existing tools work
  - TACC plans a large cluster

NVIDIA: Fermi (2010)

3 billion transistors

448 (512) cores



Tesla (2008)

GPU: 2008
933Gflops
150W

CPU:2008
~45 Gflops
160W

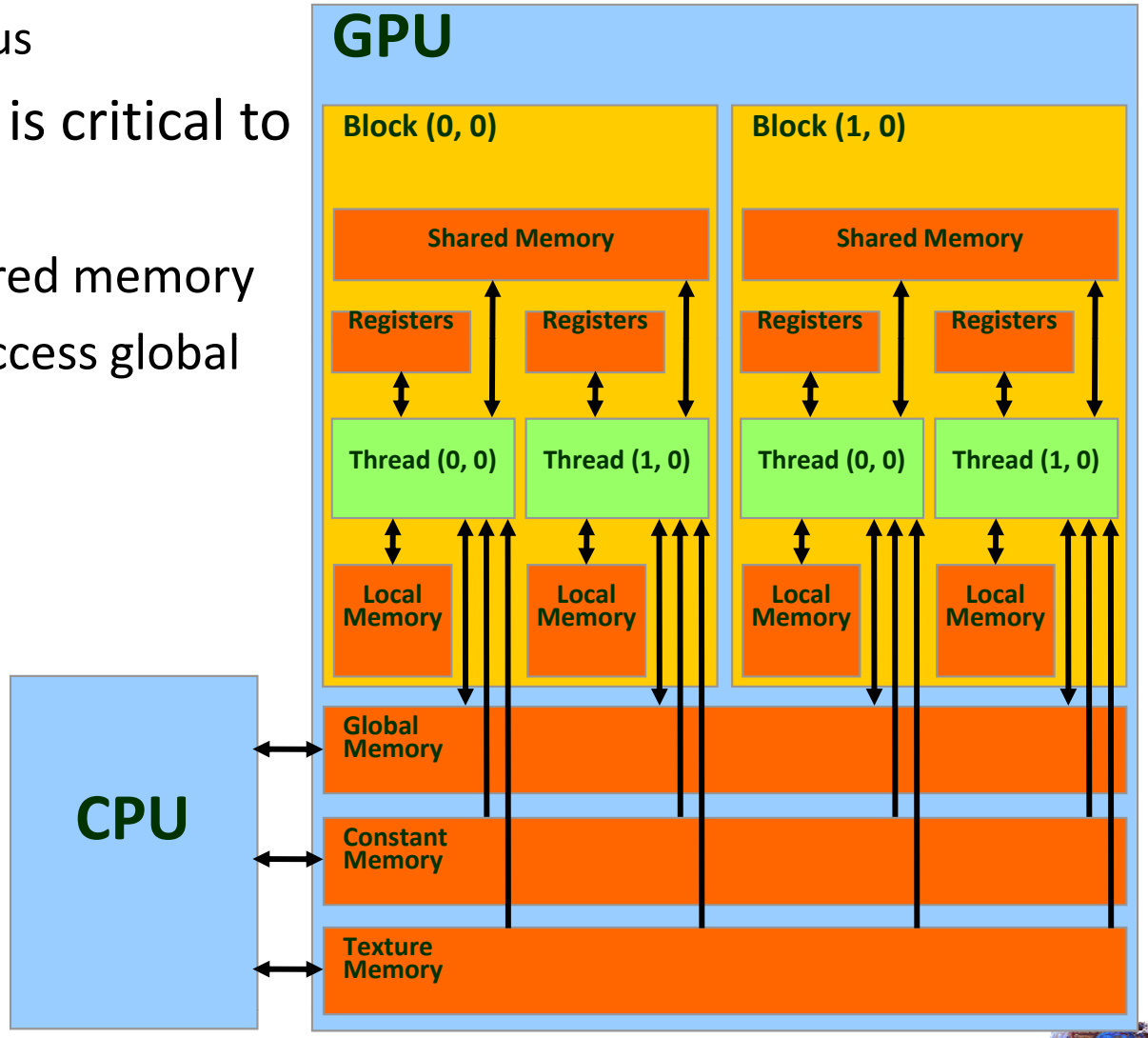
ESRL
GLOBAL SYSTEMS DIVISION

Application Performance

- NVIDIA GPU: a co-processor to CPU
  - Data moved across PCI bus
- Efficient use of memory is critical to good performance
  - 1-2 cycles to access shared memory
  - Hundreds of cycles to access global memory

Memory	Tesla	Fermi
Shared	16K	16/48K
Cache		16/48K
Global	1-2GB	4-6GB

GPU Multi-layer Memory (Tesla)



GPU Programming Approaches

- Language Approach
  - CUDA, OpenCL, CUDA Fortran, etc.
  - Less portable to different architectures
  - Requires that separate versions be maintained
    - In practice this rarely works – too costly, difficult
- Directive-based Approach
  - Appear as comments in the source
    - !ACC$DO VECTOR (1)
  - Compilers can analyze and (hopefully) generate efficient code
    - Dependent on maturity

Nonhydrostatic Icosahedral Model (NIM)

- Team of scientists, parallel programmers and computer scientists
  - Designed for GPU, fast parallel execution

NIM Development Team

- **Jin Lee**
- **A.E. MacDonald**
- Jung-Eun (Esther) Kim
- Ka Yee Wong
- Jian-Wen Bao
- Ning Wang
- Jacques Middlecoeff
- Mark Govett
- Tom Henderson
- Jim Rosinski

Collaborators/Contributors:

- **YSU:** Prof. Songyou Hong, Sueng-On Hwang(KMA)
- **CSU :** Prof. David Randall, Todd Jones
- **PSD :** Drs. George Kiladis, *Stefan* N. Tulich.
- **Purdue U:** Prof. Wen-Yih Sun
- **GFDL:** Dr. S.-J. Lin

ESRL finite-volume Icos- models (FIM/NIM)

ESMF

- **FIM (flow-following finite-volume Icosahedral model): A hydrostatic model <u>for NCEP global model ensemble.</u>**
- **A hydrostatic model consists of 2-D finite-volume SWM coupled with hybrid σ-θ vertical solver.**
- **Produce accurate medium-range weather forecasts with scores comparable to GFS.**
- **Development began in ~2000**

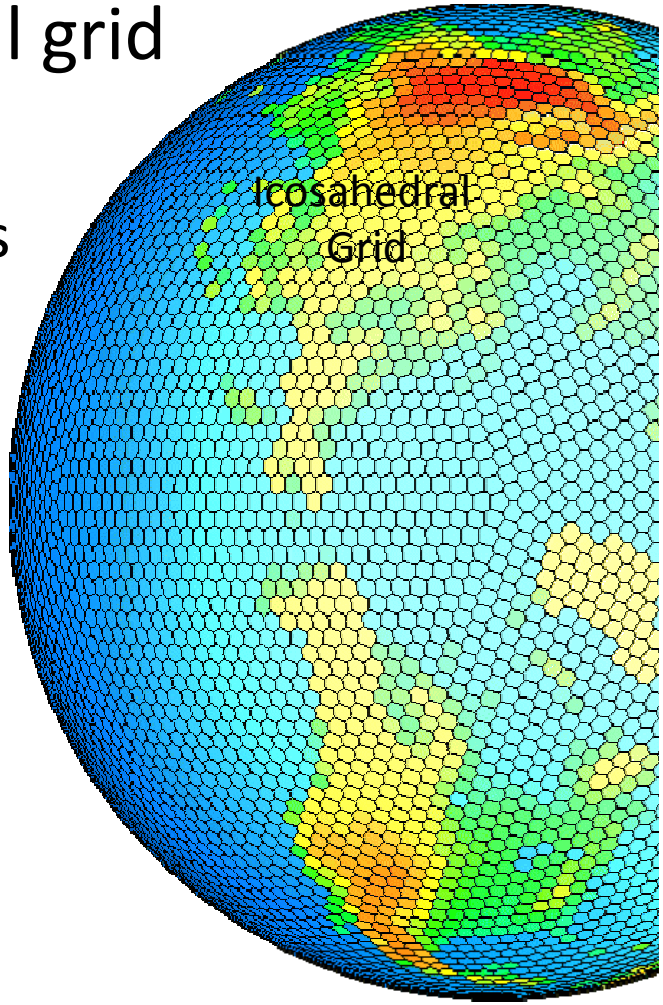
- **NIM (Nonhydrostatic Icosahedral model): A multi-scale global model <u>for weather and intra-seasonal climate predictions</u>.**
- **Extension of 2-D finite-volume integration into 3-D integration on control volume defined on the height coordinate.**
- **Use the latest GPU technology to speed up high-resolution model calculations**
- **Targeting 2KM global scale**
- **Development began in ~2005**

Novel features of FIM/NIM

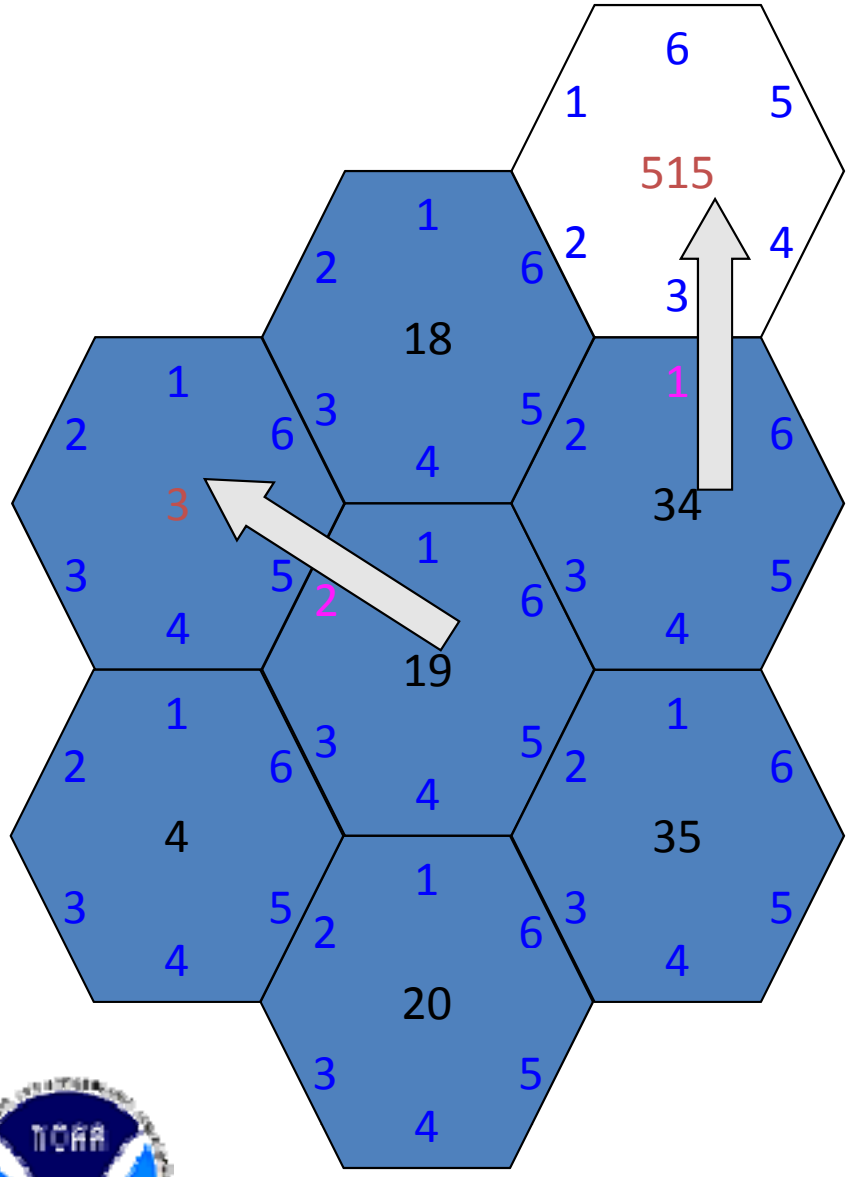
- Finite-volume Integrations on *Local Coordinate*
- Efficient Indirect Addressing Scheme on Irregular Grid
  - Adopted by MPAS icos model at NCAR
- FIM: Hybrid σ-θ Coordinate w/ GFS Physics
- Conservative and Monotonic Adams-Bashforth 3rd-order FCT Scheme
- Grid Optimization for Efficiency and Accuracy
- Novel Features of NIM
  - Three dimensional finite-volume integration
  - Runge-Kutta (RK) – 4th solver for vertically propagating acoustic waves
  - Conservative and positive definite transport scheme

NIM Code Design

- Uniform, hexagonal-based, icosahedral grid
  - 1 horizontal dimension
  - Novel indirect addressing scheme permits concise, efficient code
- Separated coarse and fine grain parallelization
  - CPU controls high level flow
    - Distributed memory parallelism (MPI)
    - Initialization, message passing, I/O
  - GPU executes dynamics routines
    - Data is resident in GPU memory
    - Data is passed to CPU only for I/O and inter GPU communications



NIM/FIM Indirect Addressing (MacDonald, Middlecoff)



- Single horizontal index
- Store number of sides (5 or 6) in "nprox" array
  - nprox(34) = 6
- Store neighbor indices in "prox" array
  - prox(1,34) = 515
  - prox(2,19) = 3
- >1% performance impact
  - Indirect reference is not the innermost dimension
- Very compact code

Simple Loop With Indirect Addressing

- Compute sum of all horizontal neighbors
  - nip = number of columns
  - nvl = number of vertical levels

```
xnsum = 0.0
do ipn=1,nip                  ! Horizontal loop
  do isn=1,nprox(ipn)         ! Loop over edges (sides, 5 or 6)
    ipp = prox(isn,ipn)       ! Index of neighbor across side "isn"
    do k=1,nvl                ! Vertical loop
      xnsum(k,ipn) = xnsum(k,ipn) + x(k,ipp)
    enddo
  enddo
enddo
```

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
```

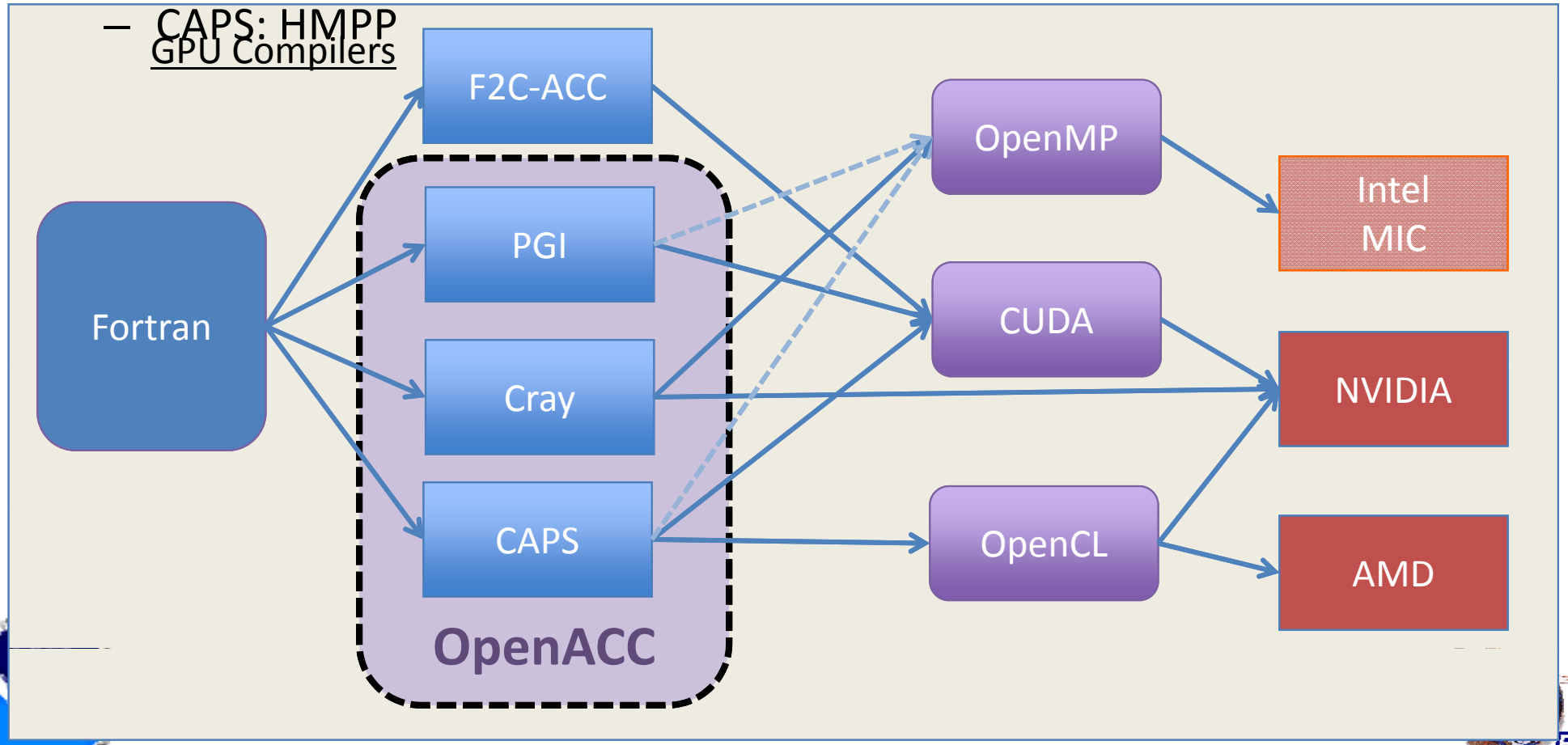
- Continues to be developed
  - Until commercial compilers are better

Directive-Based Fortran GPU

Compilers and Portability

OpenACC directives adopted by vendors

- PGI: Accel
- F2C-ACC: OpenSource
- CAPS: HMPP
- Cray: OpenMP



Fortran GPU Compiler Results (2011)

Using NIM G5 - 10242 horizontal points, 96 vertical levels

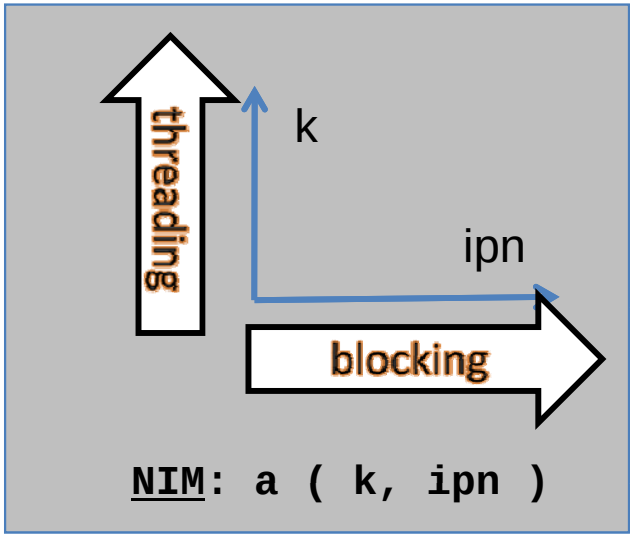
Fermi GPU vs. Intel Westmere CPU Socket

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

- Used PAPI performance counters on CPU (GPTL)
  - *Estimated* ~29% of peak (11.2 GFLOPS) on CPU

Parallelization Factors for NIM

- Code design a dominant factor in performance
  - Weather and climate codes typically have a high memory access to compute ratio
    - Memory accesses limit performance
  - 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

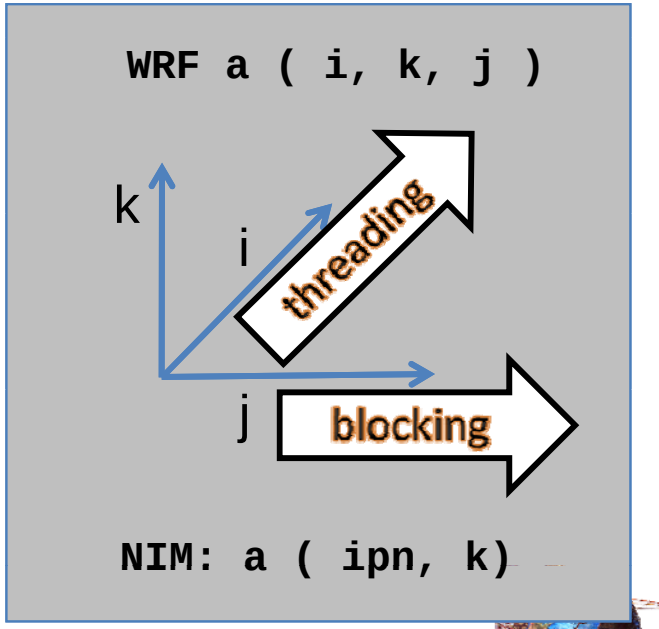


j

WRF Physics

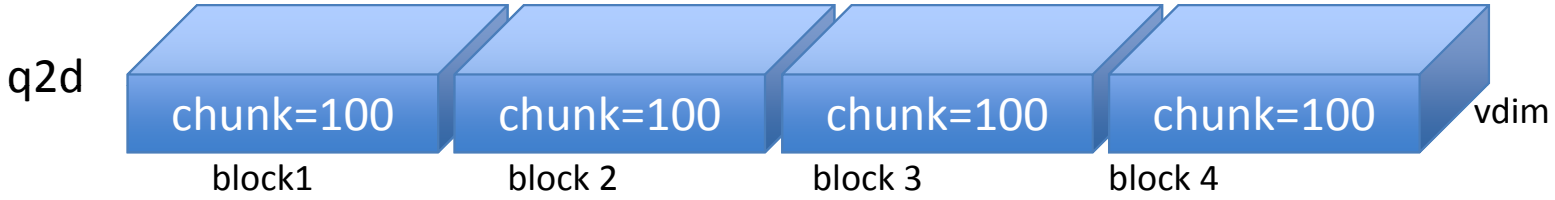
- Legacy code
  - Community Model used worldwide for more than a decade
  - WRF-ARW, WRF-NMM, WRF-RR, WRF-CHEM, HWRF
- Traditional cartesian grid
  - 3D arrays (horizontal, vertical, horizontal) = = > array3D(i,k,j)
- Designed for CPU architectures
  - Primary calculations on 2D arrays (i,k)
  - Improves cache utilitzation
- **Limited ability to change the code**

- A challenge for GPU parallelization
  - Dependencies in vertical
  - GPU: threading in horizontal dimensions
- Further challenge for NIM
  - Only 1 horizontal dimension



Application Requirements
NIM + WRF Physics

- Chunking
  - Assigning threads and blocks to the same dimension



- Approach has been successful
  - ~2-3x performance improvement anticipated
  - No changes to WRF code base
- Supported in F2C-ACC
  - Working with vendors to provide this

Application Requirements

weather and climate codes

- Promotion of variables
  - Needed for correctness & performance
  - Added dimension needed to store block or thread parallel calculations
    - 2D array + + + > 3D array
- Demotion of variables
  - 1D arrays - - -> GPU register variables
  - 2D arrays - - -> shared memory arrays
  - Results in huge performance increases

Future Work

- NIM continues to be developed
  - physics integration
  - Aqua-planet simulations
  - Testing with real data this summer

- Committed to a single source
  - Performance portable between CPU, GPU, serial, parallel
    - NVIDIA, Intel MIC, AMD
  - We anticipate some challenges for legacy codes
    - FIM, WRF, HYCOM
    - Unclear performance of Intel MIC, tools

Conclusion

- F2C-ACC has proven useful
  - For NIM parallelization
  - To evaluate commercial compilers
    - Good interactions with vendors on improvements
- NIM GPU performance results encouraging
  - 5x faster than Intel-Westmere (socket-to-socket)
  - Plan to run NIM on Intel MIC in 2012
- Challenges Remain
  - Codes take too long to port to GPUs
    - MIC may be better, but price / performance unclear
  - Performance portability a concern
  - Standards for GPU directives

Final Thoughts

- Unclear how well industry will support us
  - Climate & weather only a small piece of their business
- Do we need to develop our own tools?
  - Preserve & adapt our models
  - Enhance code portability
  - Improve performance optimizations
  - Architecture specific changes
    - Array re-declarations
    - Loop re-ordering
    - Loop unrolling
    - Promotion & demotion of variables
    - Use of special memories, vector instructions
- These are all possible with existing compiler technology