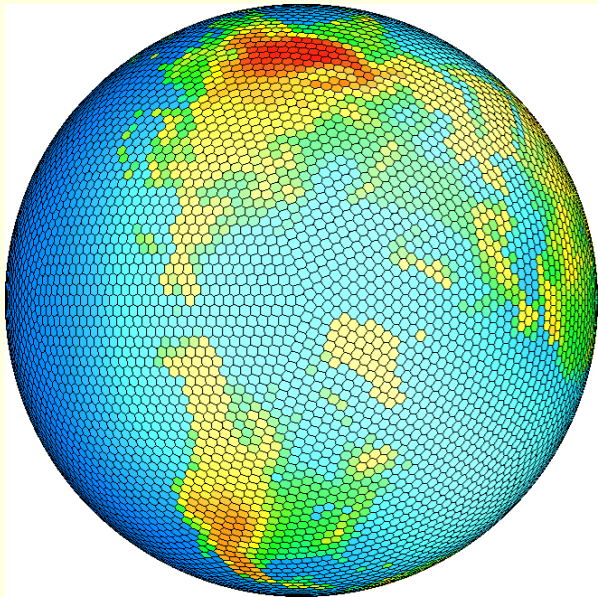
Porting and Tuning WRF Physics Packages on Intel Xeon and Xeon Phi and NVIDIA GPU

Tom Henderson
Thomas.B.Henderson@noaa.gov
Mark Govett, James Rosinski,
Jacques Middlecoff
NOAA Global Systems Division

Indraneil Gokhale, Ashish Jha,
Ruchira Sasanka
Intel Corp.



WRF Physics Packages

- WSM6
  - Microphysics parameterization used in WRF, NIM (NOAA), MPAS (NCAR), etc.
    - Water vapor, cloud water, cloud ice, rain, snow, graupel
- RRTMG-LW
  - Longwave radiation package used in too many NWP models to list here
- Double-precision in NIM & MPAS, single-precision in WRF
  - All results in this talk are double-precision

Approach

- Re-use WSM5 tuning for Xeon Phi already done by John Michalakes where possible
- Re-use RRTMG-LW experience from John Michalakes
- Diverge from John's approach in use of optional compile-time constants for vertical dimension
- Use Non-Hydrostatic Icosahedral Model (NIM) as dynamical core to test performance improvements

Source Code Requirements

- Must maintain single source code for all desired execution modes
  - Single and multiple CPU/GPU/Xeon Phi
  - Prefer Fortran + directives
    - Use F2C-ACC (Govett) and commercial OpenACC compilers for GPU
    - Use OpenMP plus Intel directives for Xeon CPU and Xeon Phi
    - Use SMS (NOAA) for distributed (MPI) parallelism
- Avoid architecture-specific code transformations
  - Unless automated

Port Validation

- Good cross-architecture bitwise-exact solutions for NIM dynamics validation
  - Xeon Phi: use slow-but-exact Intel math library to match Xeon & Xeon Phi
  - NVIDIA GPU: Optionally push rare math library calls back to CPU for testing
- Rudimentary validation for WSM6 and RRTMG-LW thus far

What Makes “Good” Code for Xeon and Xeon Phi?

- OpenMP threading
  - Minimize threading overhead
- Vectorizable
- Fixed inner dimension
  - Compile-time constants
  - Build-time-adjustable length of inner dimension
    - Optimal = vector width
- Aligned memory
  - Begin arrays on vector boundaries

What Makes “Good” Code for Xeon and Xeon Phi?

- Intel compiler warns of inefficient behavior
  - Loops that cannot be vectorized
  - “Partial”, “peel”, and “remainder” loops
  - Unaligned access
  - “Gathers” and “scatters”
  - Reasons for inefficiency in some cases

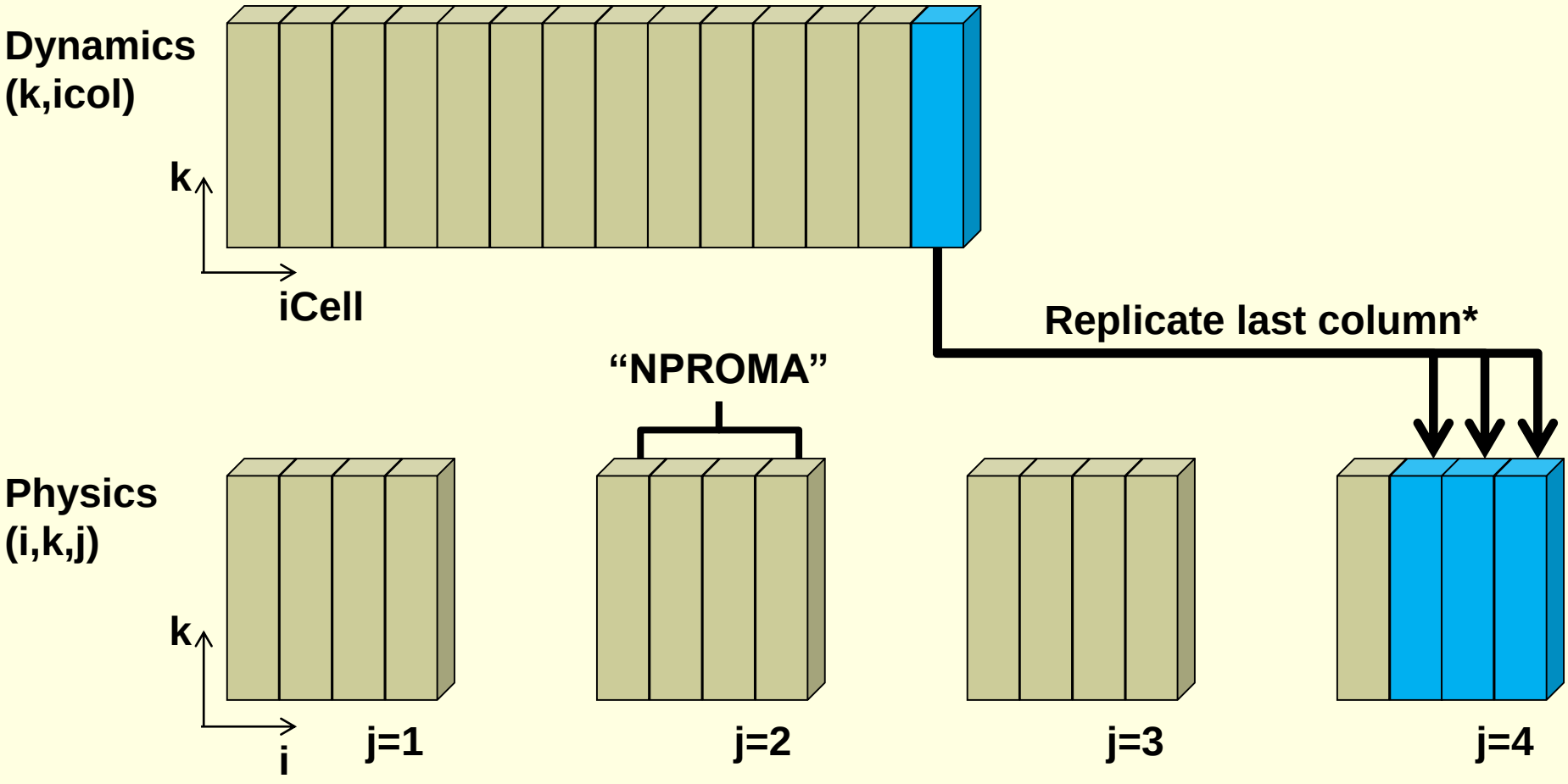
Code Modifications: Threading

- Add single OpenMP loop to NIM for **all** "physics"
  - Minimizes OpenMP overhead
- Split arrays into "chunks" with fixed inner dimension
  - Allow large chunk sizes for GPU, small for Xeon & Xeon Phi
  - Modify loops that transfer arrays between dynamics and physics to handle "chunks"
    - Very little impact on existing code
- Use Intel Inspector to find race conditions
  - It really works

Code Modifications: Threading

- NIM (and MPAS) dynamics: (k,iCell)
  - “k” = vertical index within a single column
  - “icol” = single horizontal index over all columns
- WRF Physics: (i,k,j)
  - “i” = horizontal index over columns in a single “chunk”
  - “k” = vertical index within a single column
  - “j” = index over “chunks”
- Use OpenMP to thread “j” loop

Example: Chunk Width = 4



* Replication avoids adding "if" blocks to all physics "i" loops

Code Modifications: Vectorization

- Add compiler flag for alignment
- Split/fuse loops per Intel compiler complaints
- Add Intel compiler directives
  - Alignment
    - Compiler cannot always tell if memory is aligned
  - Vectorization
    - Compiler cannot always tell if a loop can be safely vectorized
    - Intel added two of these missed by me

Compile-Time Constants

- Performance improves if compile-time constants are used for memory and loop bounds with Intel compiler
  - Also benefits GPU since sizes of arrays in GPU “shared memory” must be known at compile time
- Stride-1 loops work best
- Use Fortran parameters or literal constants
- But, hard-coding compile-time constants is too constraining for research codes…

Compile-Time Constants

- Add build-time option to use compile-time constants
  - Select “i” chunk size at build time (John M.)
  - Select “k” vertical size at build time (new)

```
real :: y(ims:ime,kms:kme)
real :: x(kms:kme)
do k=kts,kte
  do i=its,ite
```

```
real :: y(1:8,1:32)
real :: x(1:32)
do k=1,32
  do i=1,8
```

- Optional + automatic = very flexible
  - Many good ways to do this…
  - Constant “k” allows simplification of WSM5 code

NIM Test Cases

- Single-node test
  - 225km global resolution (10242 columns)
  - Time-step = 900 seconds
  - 72 time steps
  - WSM6 and RRTMG-LW called every time step
- Mimic expected number of columns per node for target resolution (~3km)
- 32-level idealized case
- 32-level and 41-level real data cases

Devices and Compilers

- SNB 2 sockets (on loan from Intel)
  - E5-2670, 2.6GHz, 16 cores/node
  - ifort 14
- IVB-EP 2 sockets (Intel endeavor)
  - E5-2697v2, 2.7GHz, 24 cores/node
  - ifort 15 beta
- HSW-EP 2 sockets (Intel endeavor)
  - E5-2697v3, 2.6 GHz, 28 cores/node
  - ifort 15 beta
- KNC 1 socket (on loan from Intel)
  - 7120A, 1.238GHz
  - ifort 14
- NVIDIA K20X GPU (Titan, ORNL)
  - Mark Govett, F2C-ACC, work in-progress

WSM6 Run Times

Device	Threads	Chunk Width (DP words)	Time	Time with Intel Optimizations
SNB	32	4	7.5	**6.7**
KNC	240	8	8.7	**5.6**
IVB-EP	48	4	3.4	**3.1**
HSW-EP	56	4	**2.6**	--
K20X GPU	--	--	**5.3**	--

- Intel optimizations reduce precision and make assumptions about padding, streaming stores, etc.
- Defensible because WSM6 uses single precision in WRF
- KNC: ~12% further speedup using ifort 15 (not beta)
- GPU preliminary result courtesy of Mark Govett

WSM6: Benefit of Compile-Time Constants for Xeon & Xeon Phi

Device	Threads	Baseline Time	Time With Constant “k”	Time With Constant “i” and “k”
KNC	240	12.5	11.6	8.7
IVB	48	4.4	4.1	3.4

- 1.4x speedup on KNC
- 1.3x speedup on IVB

WSM6: Effect of Vector Length on Xeon & Xeon Phi

Device	2 DP Words	4 DP Words	8 DP Words	16 DP Words	32 DP Words
KNC	--	--	8.68	8.82	10.10
IVB	3.76	3.38	3.51	3.68	3.71

RRTMG-LW: Benefit of Compile-Time Constants (Preliminary)

Device	Threads	Baseline Time	Time With Constant "k"
KNC	240	19.1	13.5
IVB	48	4.5	3.2

- ifort 15 (not beta)
- ~1.4x speedup on KNC
- ~1.4x speedup on IVB
- Directives not yet added, more tuning TBD

Compile-Time Constants: All Stars Must Align

- Compiler flags
- Use compile-time constants for loop *and* memory bounds
- Use ifort 14 or 15
- Use SNB, IVB, or HSW (*not* Westmere)
  - Use AVX for maximum effect
- May need directives
  - !DIR$ASSUME_ALIGNED
  - !DIR$VECTOR ALIGNED
- Pay attention to compiler output

Summary

- KNC competitive with SNB despite slower clock (WSM6)
- K20X GPU competitive with KNC
- KNL (and GPU) will need to catch up with IVB/HSW
- Optimizations sped up both Xeon and Xeon Phi
- Optional compile-time constants beneficial for Intel compiler and for GPU shared memory
- Simplified WSM5 and WSM6 code via optional compile-time *vertical* loop and memory bounds

Near-Future Directions

- Finish RRTMG-LW
- Understand use of optional compile-time constants in more detail
  - Possible future Intel compiler directives or PGO to address this optimization?
  - Test with other compilers (PGI, Cray)
- Considering solution for inclusion in NIM, WRF, MPAS, etc. (with Michalakes)
  - We've been here before, do better this time
- Target other WRF physics packages used by NOAA models
- GFS physics

Thanks to…

- Intel: Mike Greenfield, Ruchira Sasanka, Ashish Jha, Indraneil Gokhale, Richard Mills
  - Provision of “loaner” system and access to endeavor
  - Consultations regarding code optimization
  - Work-arounds for compiler issues
  - Aggressive optimization
- John Michalakes
  - Consultation regarding WSM5 work
  - Code re-use

Thank You

Compiler Options

- Xeon baseline optimization flags
  - -O3 –ftz -qopt-report-phase=loop,vec -qopt-report=4 -align array64byte -xAVX
- Xeon aggressive optimization flags
  - -fp-model fast=1 -no-prec-div -no-prec-sqrt -fimf-precision=low -fimf-domain-exclusion=15 -opt-assume-safe-padding
- Xeon Phi baseline optimization flags
  - -O3 –ftz -vec-report6 -align array64byte
- Xeon Phi aggressive optimization flags
  - -fp-model fast=1 -no-prec-div -no-prec-sqrt -fimf-precision=low -fimf-domain-exclusion=15 -opt-assume-safe-padding -opt-streaming-stores always -opt-streaming-cache-evict=0

Effect of Thread Count

Device	Max. Threads	25%	50%	75%	100%
KNC	240	14.9	10.5	--	8.7
IVB	48	--	4.4	3.8	3.4