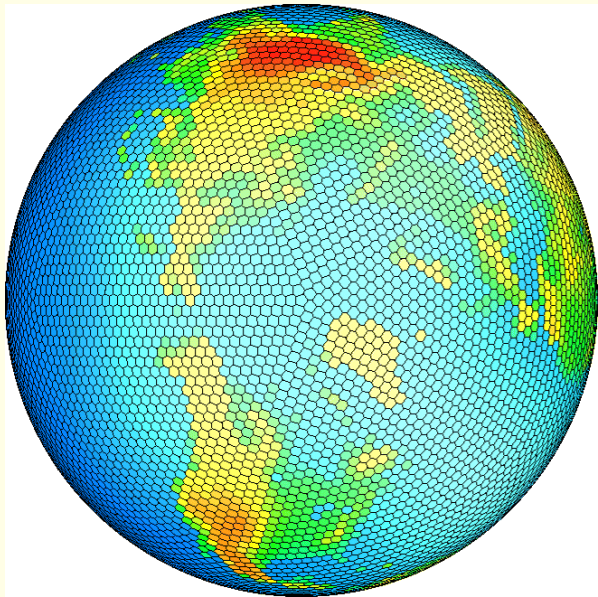
Progress Porting and Tuning NWP Physics Packages on Intel Xeon Phi

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

Indraneil Gokhale, Ashish Jha,
Ruchira Sasanka
Intel Corp.



WSM6 Microphysics

- Microphysics parameterization used in WRF
  - Water vapor, cloud water, cloud ice, rain, snow, graupel
- Also used by MPAS (NCAR), NIM (NOAA), GRIMS (YSU/KIAPS,KMA), etc.
  - More about NIM in Jim Rosinski's talk…
- Double-precision in MPAS, NIM, and GRIMS, single-precision in WRF

Why WSM6?

- Slowest single routine for expected NIM configuration
- Re-use WSM5 tuning for Xeon Phi already done by John Michalakes where possible
  - Some divergence from John's approach…
- NOTE: This is work in-progress
- NOTE: "Xeon Phi" = "KNC" = "MIC"

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 CPU and Xeon Phi
- Avoid architecture-specific code transformations
  - Automatic transformations OK

Port Validation

- Good bitwise-exact solutions for NIM dynamics debugging
  - Slow-but-exact Intel library for Xeon & Xeon Phi, Thanks Intel!
  - Optionally push rare math library calls back to CPU for NVIDIA GPU
- Rudimentary validation for WSM6 thus far
  - Idealized input data set
  - Will extend to more rigorous validation of real input data case(s) shortly

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

- Vectorizable
- Fixed inner dimension
  - Literal constants known at compile time
  - Adjustable length of inner dimension
    - Optimal = vector width
- Aligned memory
  - Begin arrays on vector boundaries
- Intel compiler warns of inefficient behavior
  - Loops that cannot be vectorized
  - “partial”, “peel”, and “remainder” loops
  - Unaligned access

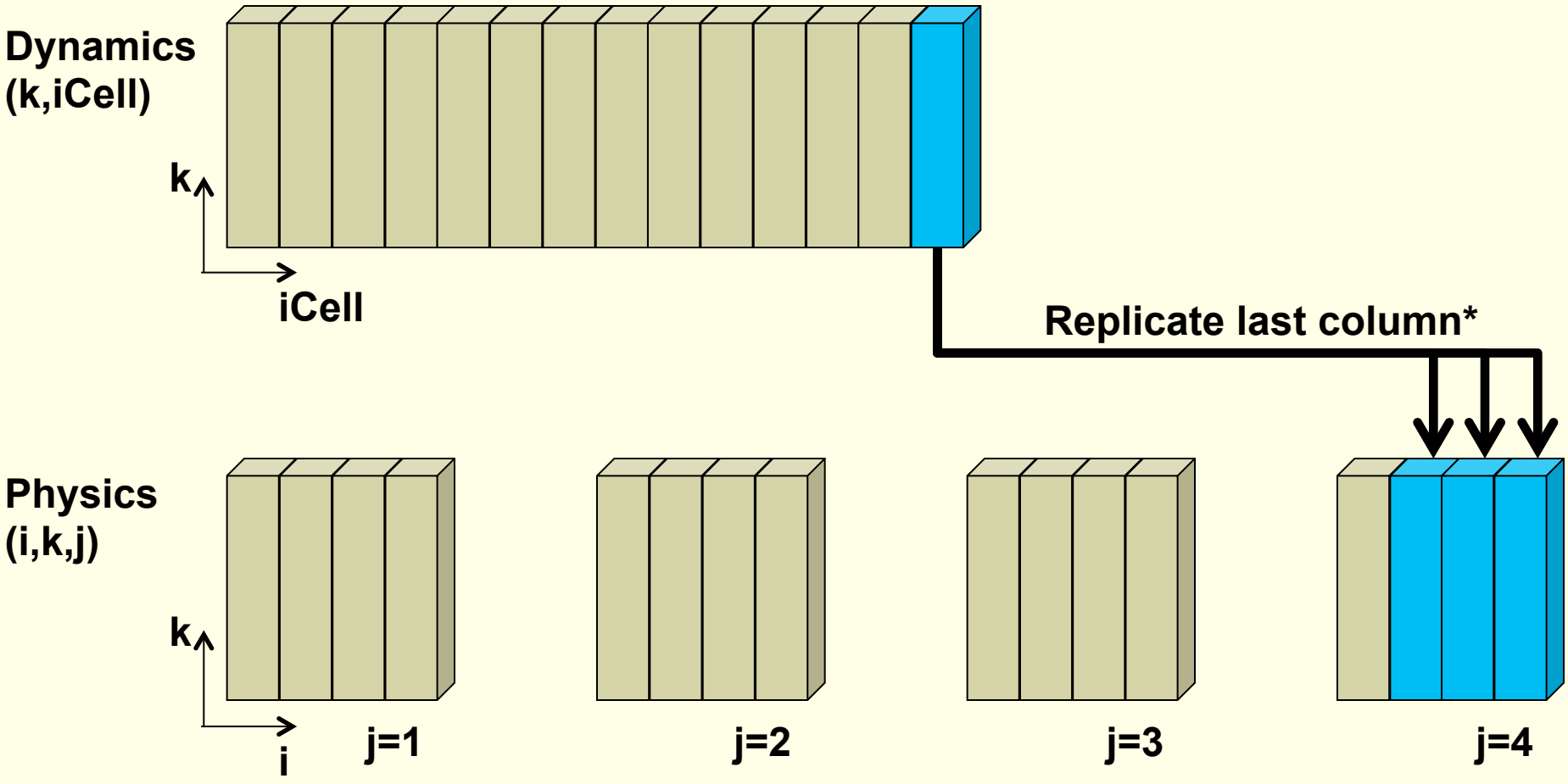
Code Modifications: Threading

- Add single OpenMP loop for **all** “physics”
  - Minimize OpenMP overhead
- Split arrays into “chunks” with fixed inner dimension
  - Pick inner chunk size at compile time
  - Allow large chunk size for GPU
  - 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

- MPAS and NIM dynamics: (k,iCell)
  - “k” = vertical index within a single column
  - “iCell” = single horizontal index over all columns
- 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

Automatic Code Modifications

- Compiler wants to see literal constants for memory and loop bounds
- WRF index conventions
  - Memory:
    - `real x(ims:ime,kms:kme,jms:jme)`
  - Loop bounds:
    - `do i=its,ite`
    - `do k=kts,kte`
  - Amenable to automatic translation
  - Manually push WRF index conventions down into a few lower-level routines

Automatic Code Modifications

- Optionally translate “i” to literal constants
  - Select chunk size at build time
- Optionally translate “k” to literal constants
  - Select vertical size at build time
  - Not done by John Michalakes

```
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

Idealized Test Case

- NIM “aqua-planet” test case
  - Single-node test
    - 225km global resolution (10242 columns)
    - 32 vertical levels
    - Time-step = 900 seconds
    - 72 time steps
    - WSM6 called every time step
  - Mimic expected per-node workload for target resolution <3km

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

WSM6 “Best” 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	48	4	3.4	3.1
HSW-EP	56	4	2.6	--

- Intel optimizations reduce precision and make assumptions about padding, streaming stores, etc.
- Defensible because WSM6 uses single precision in WRF

Effect of Automatic Code Transformations

Device	Threads	Baseline Time	Time With Literal "k"	Time With Literal "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
- Compiler directives instead of user-guided code translation?

Effect of Vector Length

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

Summary

- WSM6 performance
  - KNC competitive with SNB despite slower clock
  - KNL will need to catch up with IVB/HSW
- Optimizations sped up both Xeon and Xeon Phi

Near-Future Directions

- WSM6
  - Real-data test case
    - Add OpenMP loops (2) to MPAS physics
    - Repeat WSM6 tests in MPAS
  - More rigorous validation
  - Investigate load imbalance
  - Fractions of peak performance
    - FLOPS, memory bw & latency
- Target other WRF physics packages used by NIM

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