Preparing atmospheric modeling codes for the latest generation MIC architecture (KNL)

October 25, 2016

Jim Rosinski
NOAA/ESRL

Intel Contributors

- Mike Greenfield
- Ruchira Sasanka
- Ashish Jha
- Richard Mills

Outline

- Hardware used for FV3 and NIM testing
- New features in KNL—user implications
- NIM model: thread scaling, performance on multiple generations of Intel hardware
- FV3 overview for software engineers
- Performance issues and non-issues
- Performance results: HSW vs. KNL
- Future directions

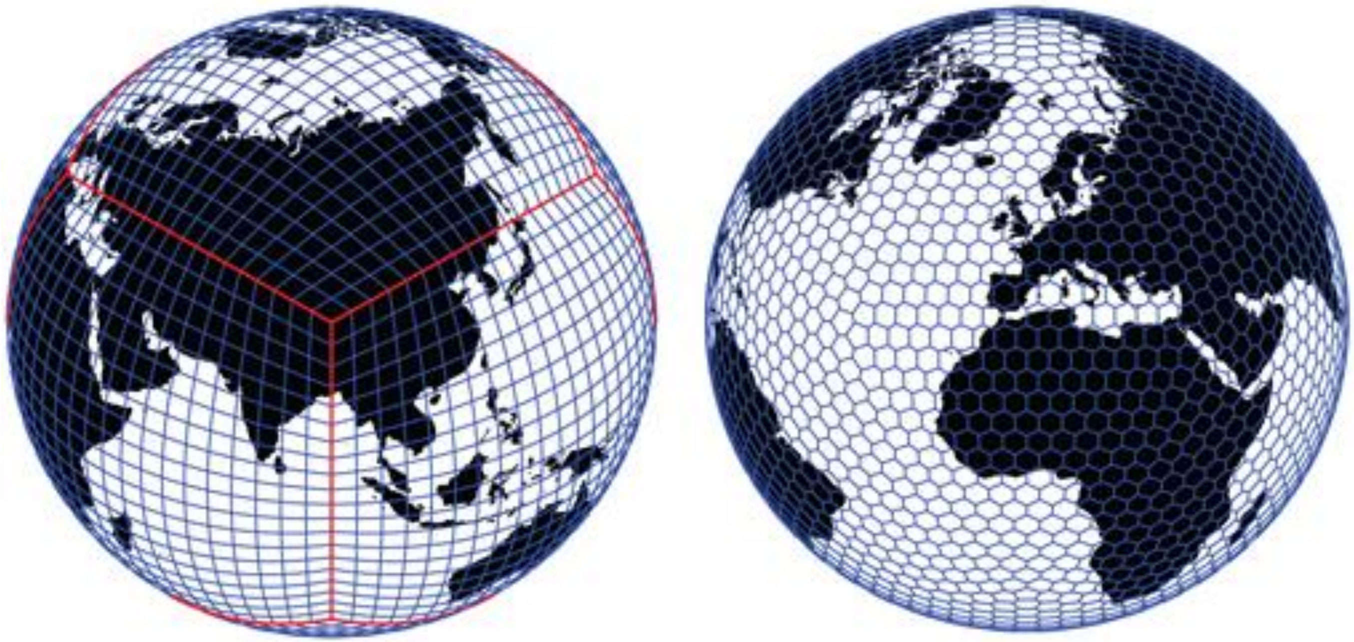
Machine specs

Machine	Arch	Cores/ node	Hyper threaded?	CPU	Compiler	MPI
Stampede 1.0	SNB/KNC	16/61	No/4-way	2.7 GHz/1.1 GHz	Ifort 15.0.2	Impi 5.0.2 (Mellanox FDR IB)
Stampede 1.5	KNL	68	Yes (4-way)	1.4 GHz	Ifort 17.0.0	Impi 17.0.0 (Omnipath)
theia	HSW-EP	24	Yes (2-way)	E5-2690@ 2.6 GHz	Ifort 15.1.133	Impi 5.0.1.035 (InfiniBand)

New features in KNL

- Up to 384 GB DDR4 main memory
- 16 GB High-bandwidth memory (MCDRAM)
- Self-booting Linux system (no attached host required)
- Binary compatible with x86
  - Exception handling works just like x86 (-fpe0 works)
- Much lower cost to –fp-model precise
- 512-bit vectors: enable with –xMIC-AVX512
- Out of order instruction execution
- Bump in core count vs. KNC (68 or 72 vs. 61)
- Clock rate increase vs. KNC (1.4 or 1.5 GHz vs. 1.1)
- 1 MPI per core allowed (OMP not mandatory)
  - 10 node 110 km NIM run only 20% slower in pure MPI mode

Cubed-sphere grid (left) and icosahedral-hexagonal grid (right)

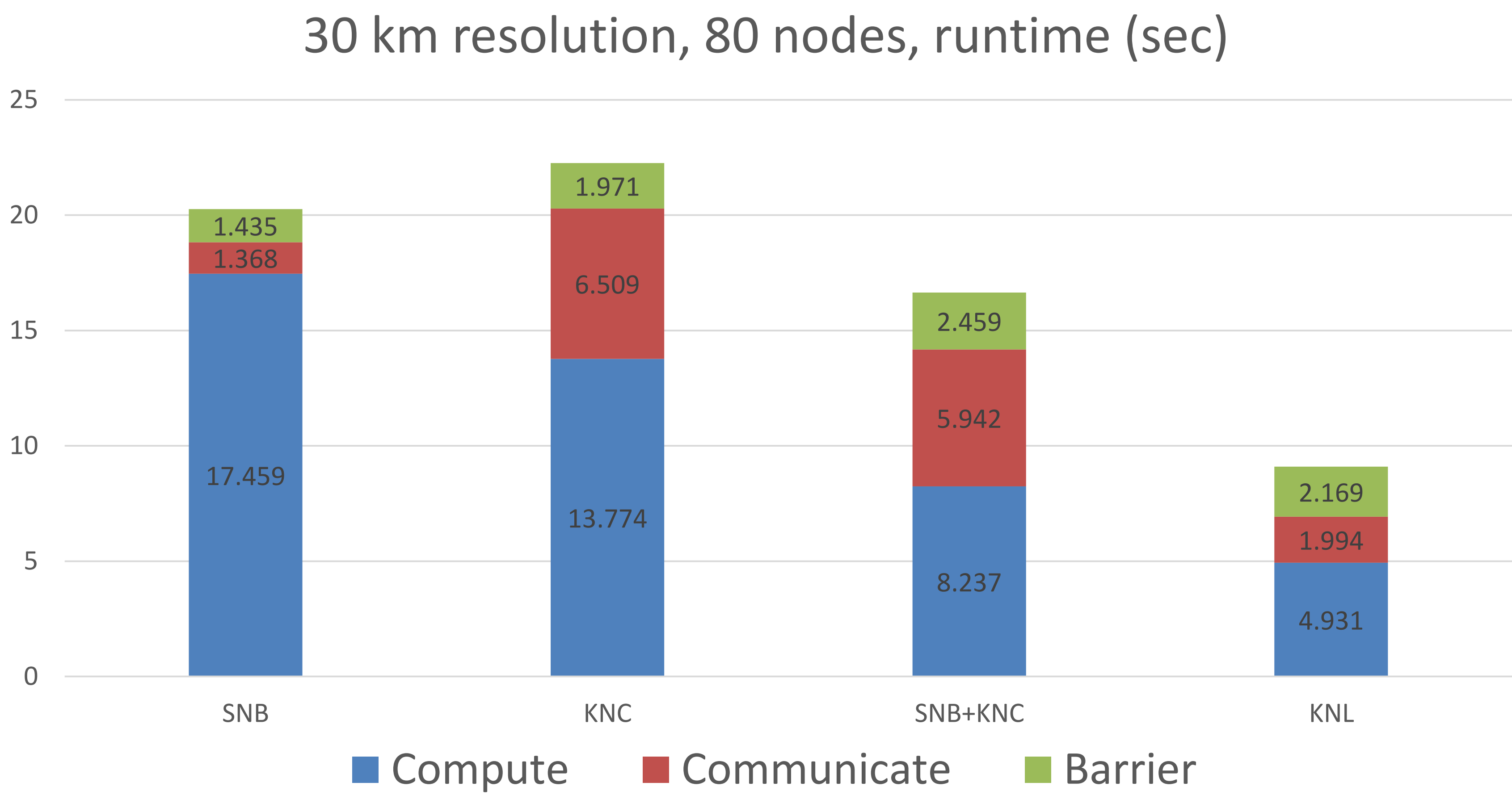


Graphic courtesy Peter Lauritzen (NCAR)

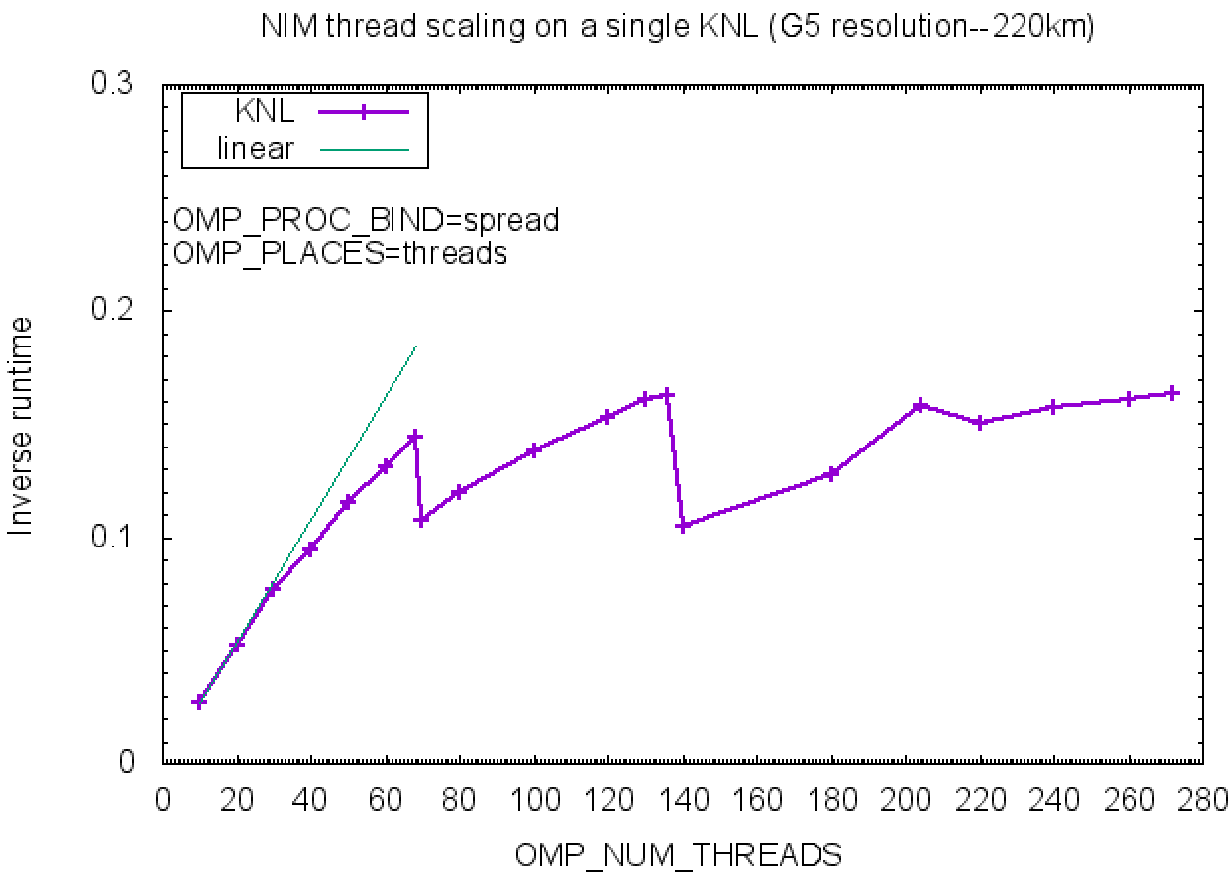
NIM, MPAS code structure

- Hybrid MPI/OMP
- Fortran, (k,i) data layout
  - Horizontal index of neighbors (i) addressed via lookup table
  - Overhead of indirect addressing amortized by “k” on the inside
- Shallow water parts of the dynamics easily vectorize over “k”
- Transpose from (k,i) to (i,k) and back for physics
  - Minimal impact on model runtime

NIM performance on 2 generations of Intel hardware



NIM thread scaling on KNL



FV3 Overview

- Selected by NWS as next-generation dynamical core to be used in forecast mode (~10 day forecasts)
- Non-hydrostatic capability required for high-resolution runs (< 10 km)
  - Hydrostatic assumption means force of gravity exactly balances vertical pressure gradient force (dp/dz = -rho*g)
- Physical parameterizations from current forecast model (GFS) have been enabled in FV3
- 2 horizontal resolutions provided to ESRL: c192 (~0.5 degree), c768 (~0.125 degree)
- 127 vertical levels

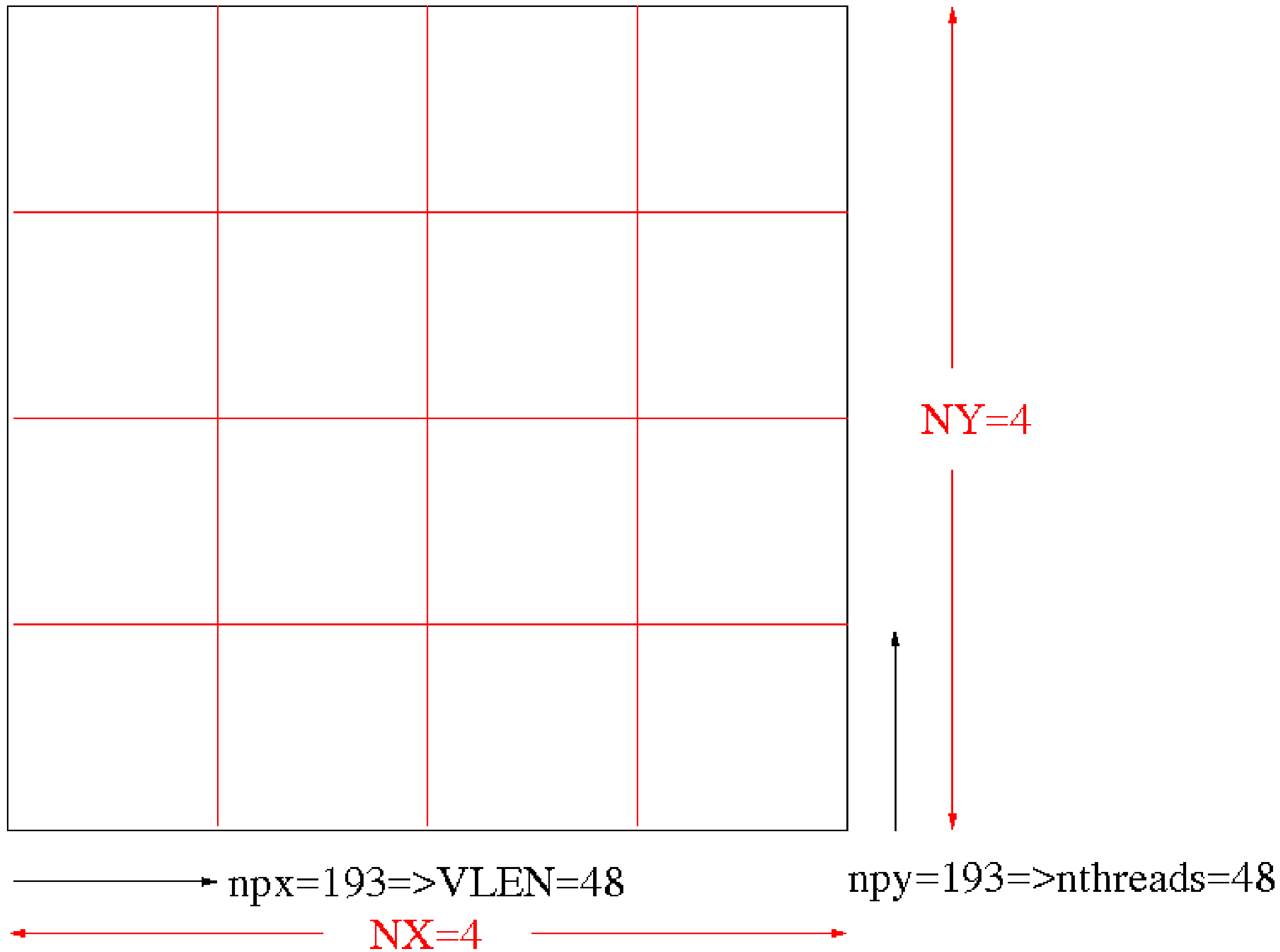
FV3 code structure

- Hybrid MPI/OpenMP dynamical core
  - Well-suited for many-core architectures
- ”Cubed sphere” means 6 separate ”faces”
  - Must have at least 1 MPI rank per face
  - Ability to run w/o MPI would be a welcome addition
- (i,j,k) data layout (Fortran ordering)
- Shallow water portions of code thread over “k” and vectorize over “i”
- Vertical dependencies: Remapping portion of code threads over “j” and vectorizes over “i”
- Special handling required for “edge” and “corner” points

FV3 code structure (cont’d)

- Highly vectorized as indicated by compiler opt report
  - Corner calculations excepted (moving ”k” inside would help)
  - Turning off vectorization slowed the code down by ~2X
- MPI communication utilizes FMS “wrapping” infrastructure
  - All messages are “packed” prior to sending, “unpacked” after receiving
  - 2-deep and sometimes 3-deep halos are exchanged
- On average, 17 OMP loops are executed each time step
  - GPTL timers indicate threading overhead is not a problem, even on KNL
  - Many “j” and “i” loops nested inside: Pushing “k” inside will be difficult

Example MPI task layout (4x4) on a single FV3 face (~0.5 degree)



ESRL mods to FV3

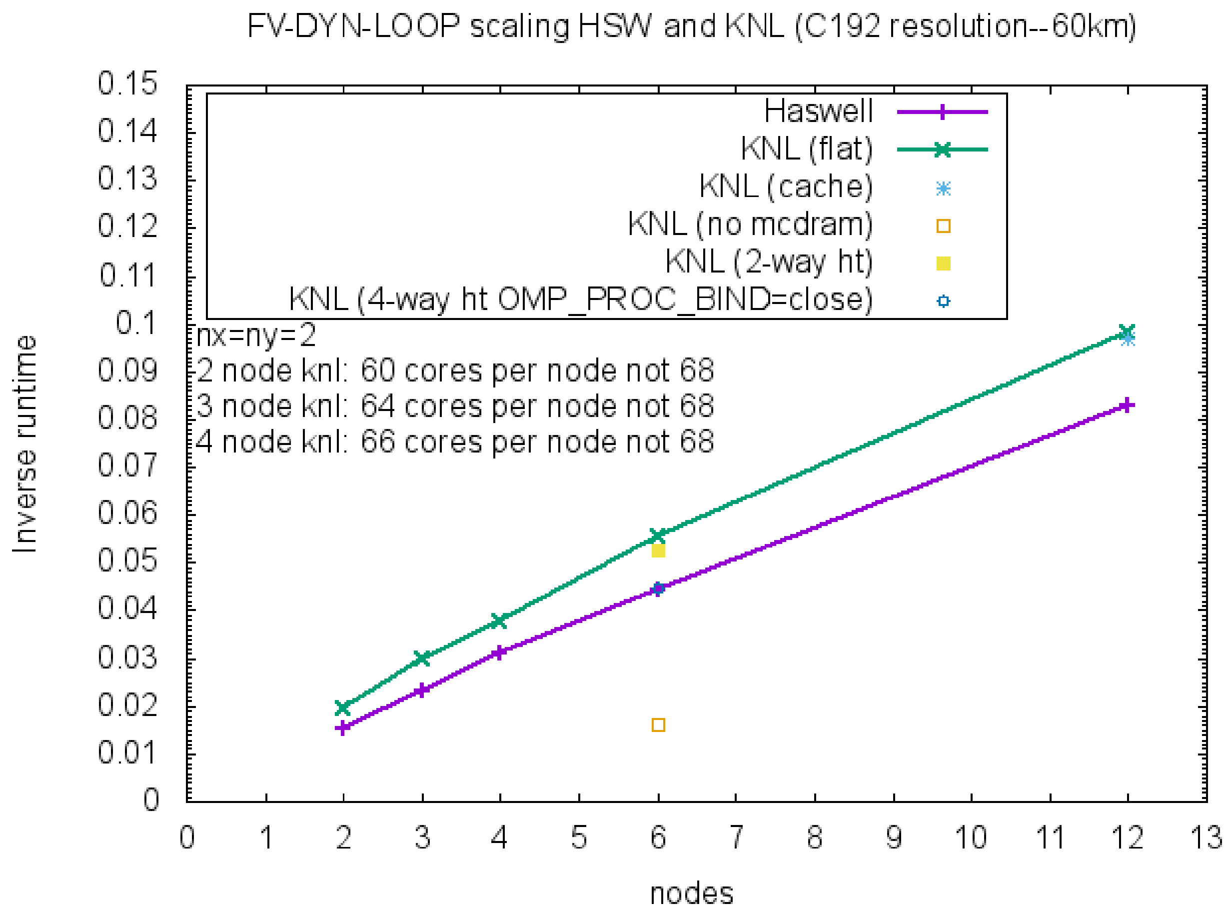
- Thread-safe timing library (GPTL)
- Fused a few OMP loops
- Updated compiler flags for HSW
- Turn off manual thread pinning on KNC/KNL

How threading overhead was measured

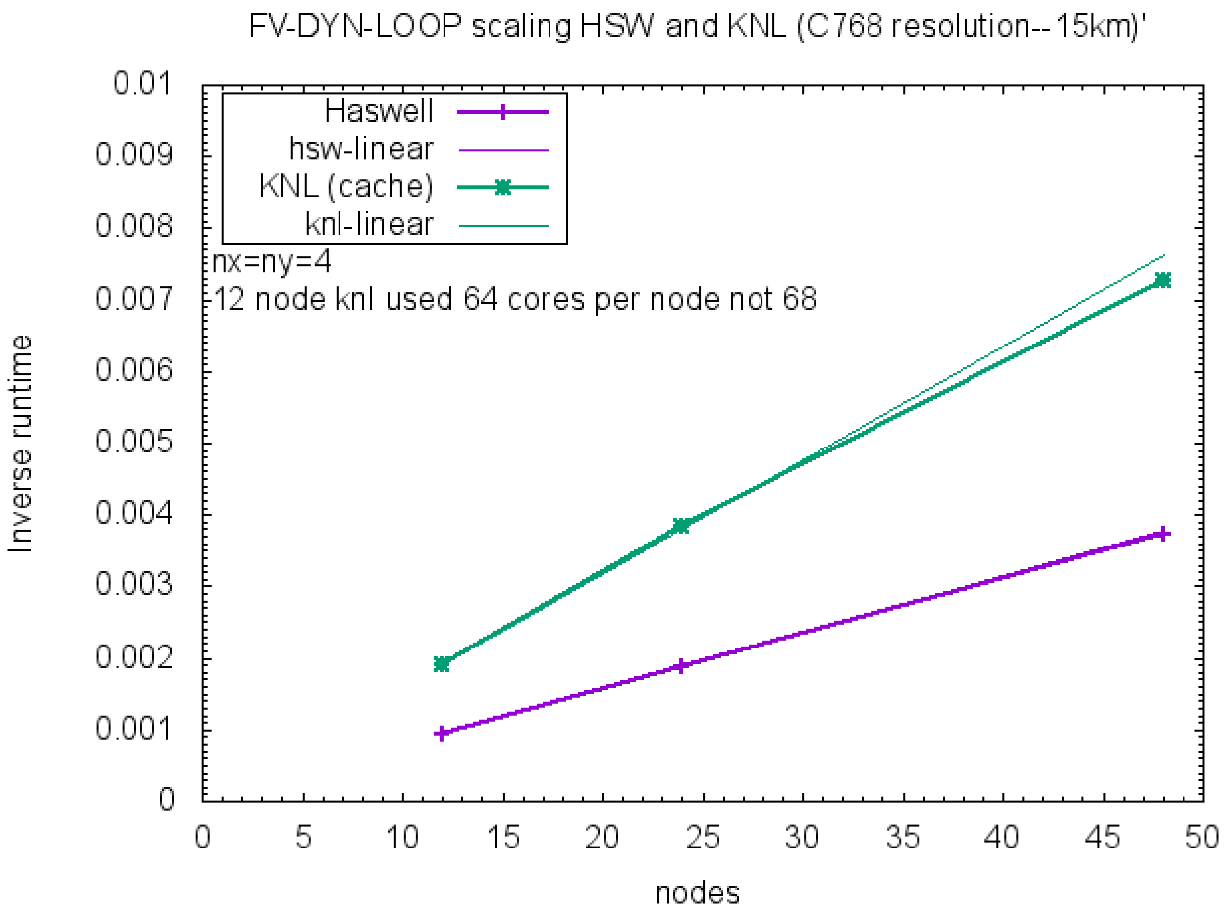
```
  ret = gptlstart_threadohd_outer ('c_sw')
!$OMP PARALLEL DO
  do k=1,npz
    ret = gptlstart_threadohd_inner ('c_sw')
    call c_sw (. . .)
    ret = gptlstop_threadohd_inner ('c_sw')
  end do
  ret = gptlstop_threadohd_outer ('c_sw')
```

- Each invocation, ”inner” timer notes time taken by slowest thread for each “inner” call
- Each invocation, “outer” timer accumulates difference between its c_sw and slowest “inner” thread
- When timers are printed, overhead is the sum of these differences

FV3 Low resolution multi-node scaling



FV3 High resolution multi-node scaling



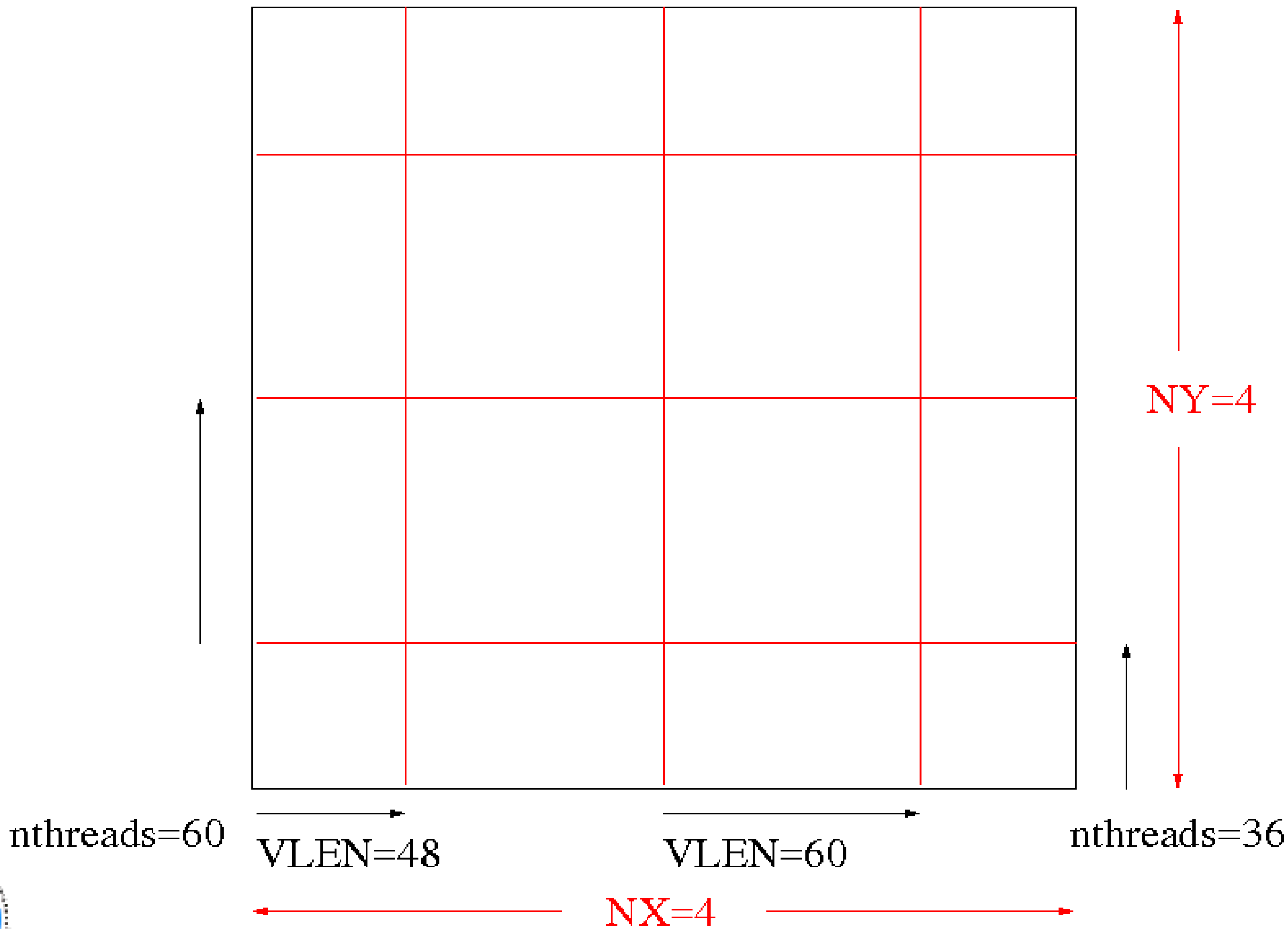
FV3 comm-time compared to run-time (hi-res)

# nodes	HSW comm time (sec)	HSW run time (sec)	KNL comm time (sec)	KNL run time (sec)
12	26.8	1053.7	26.3	523.9
24	16.9	529.5	16.1	260.5
48	11.1	267.5	10.6	137.4

A note about thread pinning…

- As received from GFDL, FV3 code manually pins threads to individual cores inside user code. This was removed for KNL, and modified for CPU
- On CPU, allowing threads to be assigned to a range of cores incurred little to no penalty, as long as they were not allowed to span sockets.
- On KNL (and KNC), threads are automatically pinned to individual cores

Proposed MPI task layout to address load imbalance from edge/corner points



Summary

- FV3 is well vectorized, and scales well in OpenMP and MPI
  - Corner and edge point calculations contribute some imbalance
  - 17 OMP loops per time step add some load-induced imbalance but little OpenMP overhead
- Cost of –fp-model precise flag (to guarantee bitwise identical results) decreased dramatically on KNL
- Pure MPI (no OMP) is possible, and works well
- MCDRAM provides a significant performance benefit
  - Cache and flat modes perform similarly when most or all memory fits in MCDRAM
- Unlike KNC, hyperthreading provides little benefit
- KNL performance exceeds HSW, particularly at higher resolution
- Attempts to push “k” loop inside will involve significant code restructuring

Where Next?

- Tweaks to thread and task pinning?
- Adjust number of points owned by MPI tasks to improve load imbalance caused by edge and corner calculations
- Improve MPI performance—can packing be eliminated in some cases?
- Enable FV3 to run without MPI (for testing)
- Can pushing “k” loop inside provide a benefit on KNL (or at least no penalty)?
- Physics optimizations

Backup slides

NIM: Overlapping communication with computation