
# Early Experiences and Future Directions with the 2nd Generation ("Knights Landing") Intel® Xeon Phi™ Processor and PETSc

**Richard Tran Mills**

**June 29, 2016**

**PETSc User Meeting, Vienna, Austria**

# Notice and Disclaimers

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Intel, Intel Xeon, Intel Xeon Phi™ are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries.

*Other brands and names may be claimed as the property of others.

# Optimization Notice

**Optimization Notice**

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel.

Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

# Acknowledgements

This presentation includes contributions from several Intel colleagues:

- Ruchira Sasanka, Karthik Raman, Chris Cantalupo, Jeff Hammond, Avinash Sodani

And is informed by helpful discussions with core PETSc developers:

- Barry Smith, Matt Knepley, Mark Adams, Jed Brown, Karl Rupp

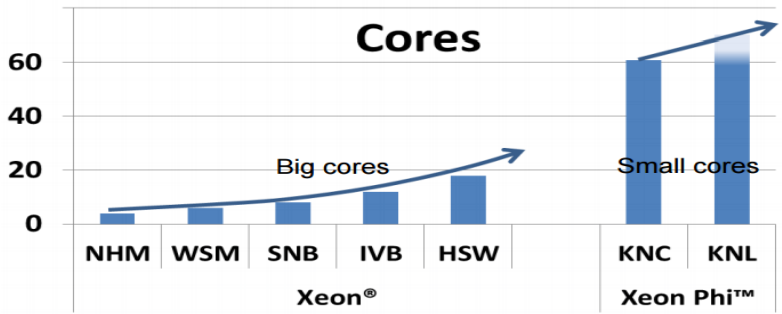
# Outline

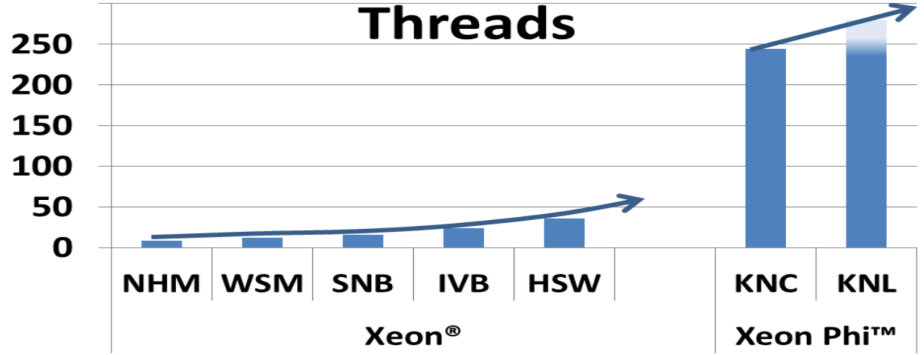
- Overview of “Knights Landing” (KNL) architecture
- Early experiences with PETSc on KNL
  - True “out-of-box” numbers: **No KNL-specific tuning yet**.
- Discussion of possible changes to PETSc for better manycore support
  - How to support multiple kinds of user-addressable memory?
    - Performance counters to assist good decision making?
  - Using new MPI-3?
  - More Intel® MKL support?

# Trend: More parallelism, deeper hierarchies

**Intel® Xeon Phi™** processors amplify importance of fine-grained parallelism, but this direction holds for machines based on "conventional" CPUs as well:

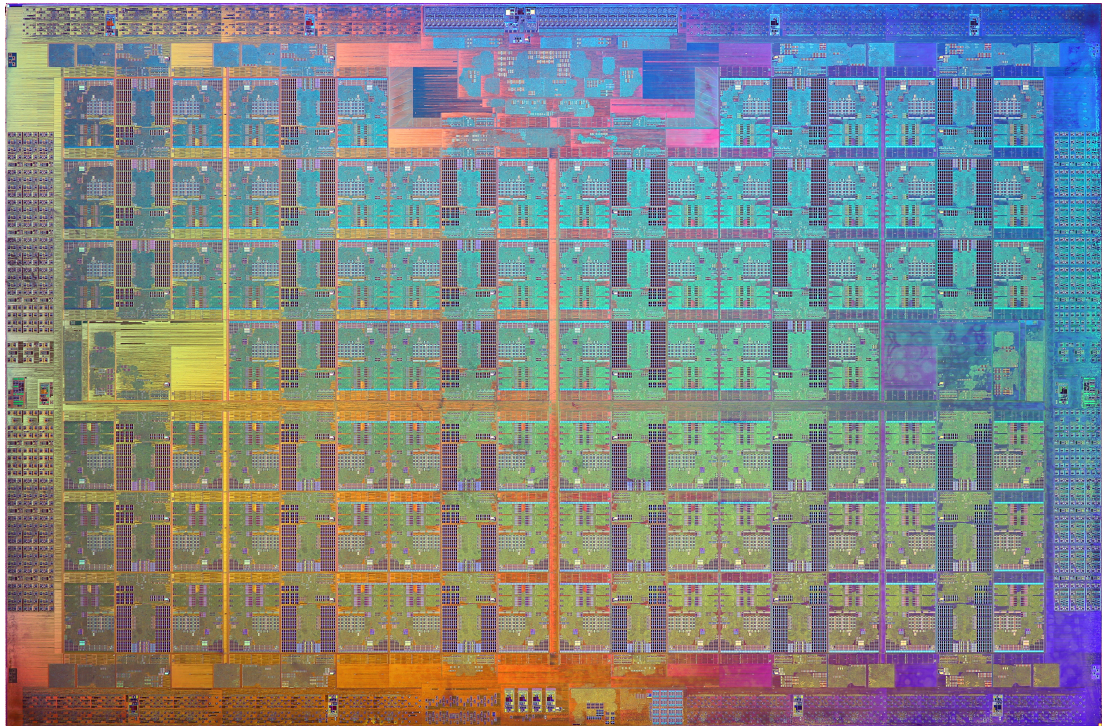
- More cores/threads in socket and across machines
  (*on per-node basis, core counts becoming roughly equivalent*)




- More data parallelism/fine-grained parallelism AVX512 (512 bit vectors w/ FMA) coming in both **Intel® Xeon Phi™** and Xeon® **processors.**
- More NUMA domains/level of storage hierarchy (DRAM NUMA domains, MCDRAM, NVRAM, IO subsystem)

# Knights Landing: Next Intel® Xeon Phi™ Processor



First **self-boot** Intel® Xeon Phi™ processor that is **binary compatible** with main line IA. Boots standard OS.

**Significant improvement in scalar** and **vector** performance

Integration of **Memory on package**: innovative memory architecture for high bandwidth and high capacity

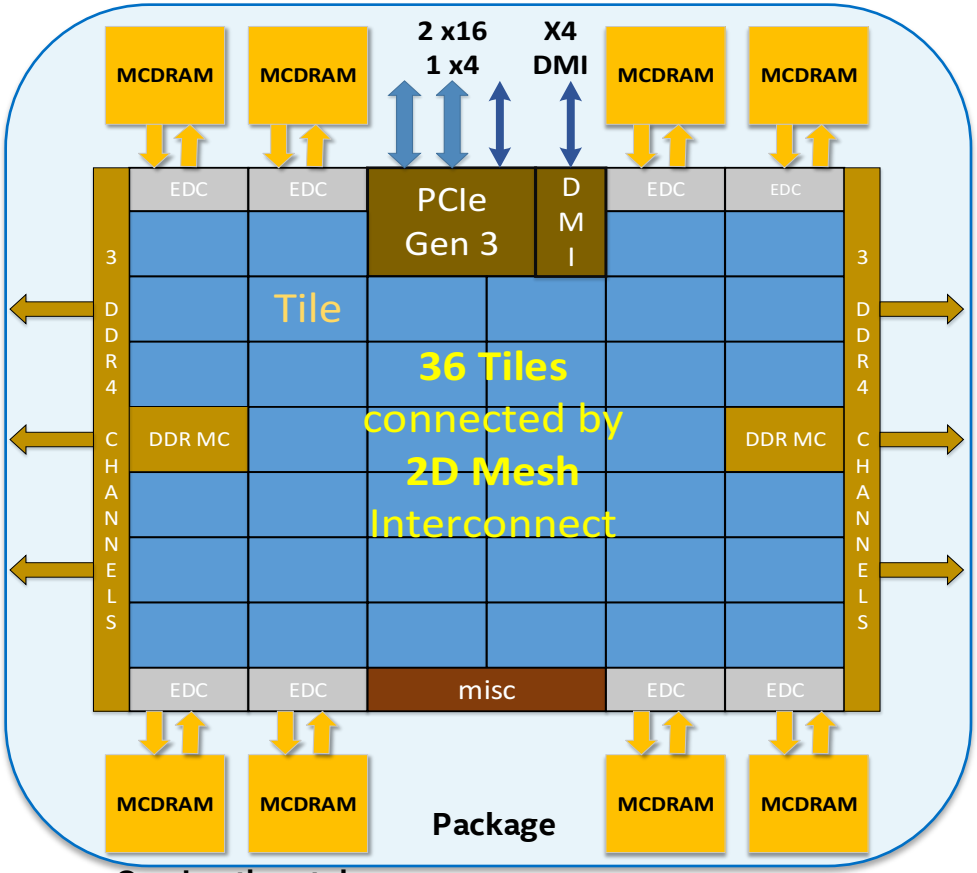
Integration of **Fabric on package**

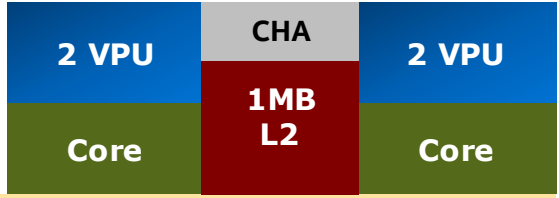
Potential future options subject to change without notice.
All timeframes, features, products and dates are preliminary forecasts and subject to change without further notification.

# What Knights Landing is NOT

- Not an accelerator
  - Bootable CPU. No PCIe offload bottleneck.
- Not a GPU
  - No CUDA vs. regular CPU source code divergence
  - Easy to run **complete application** on KNL
- Not Knight's Corner (previous generation Intel® Xeon Phi™)
  - Fully out-of-order execution
  - Mesh-on-die replaces ring bus
  - KNL is dual-issue and can saturate both VPUs from a single thread.
  - Greatly improved memory bandwidth, serial execution speed

# Knights Landing Overview




**Chip: 36 Tiles** interconnected by **2D Mesh**

**Tile**: 2 Cores + 2 VPU/core + 1 MB L2

**Memory: MCDRAM:** 16 GB on-package; High BW

**DDR4**: 6 channels @ 2400 up to 384GB

**IO:** 36 lanes PCIe* Gen3. 4 lanes of DMI for chipset

**Node**: 1-Socket only

**Fabric:** Omni-Path on-package (not shown)

**Vector**[1]: up to 2 TF/s Linpack/DGEMM; 4.6 TF/s SGEMM

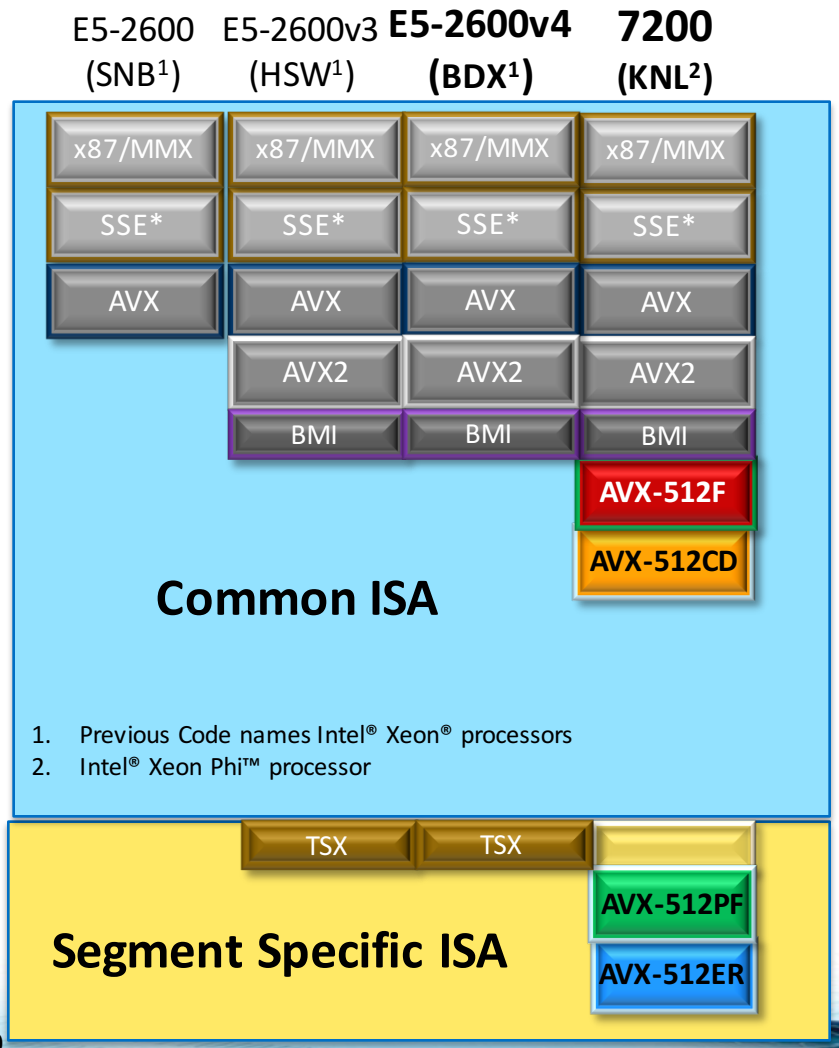
**STREAM Triad**[1]: MCDRAM up to 490 GB/s; DDR4 90 GB/s

**Scalar**[2]: Up to ~3x over current Intel® Xeon Phi™ co-processor 7120 ("Knights Corner")

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit http://www.intel.com/performance. Configurations:

1. Intel Xeon Phi processor 7250 (16GB, 1.4 GHz, 68-cores) running LINPACK (score 2000 GFLOPS), DGEMM (score 2070 GFLOPS), SGEMM (4605 GFLOPS), STREAM (DDR4 = 90 GB/s and MCDRAM = 490 GB/s), 96 GB DDR4-2133 memory, BIOS R00.RC085, Cluster Mode = Quad, MCDRAM Flat or Cache, RHEL* 7.0, MPSP 1.2.2, Intel MKL 11.3.2, Intel MPI 5.1.2, DGEMM 20K x 20K, LINPACK 100K x 100K size
2. Intel estimates based on estimated 1-user SPECint*_rate_base2006 comparing configuration 1 to Intel Xeon Phi co-processor 7120A hosted on 2x Intel Xeon processor E5-2697 v3.

# Intel ISA




## KNL implements all legacy instructions

## AVX-512 Extensions

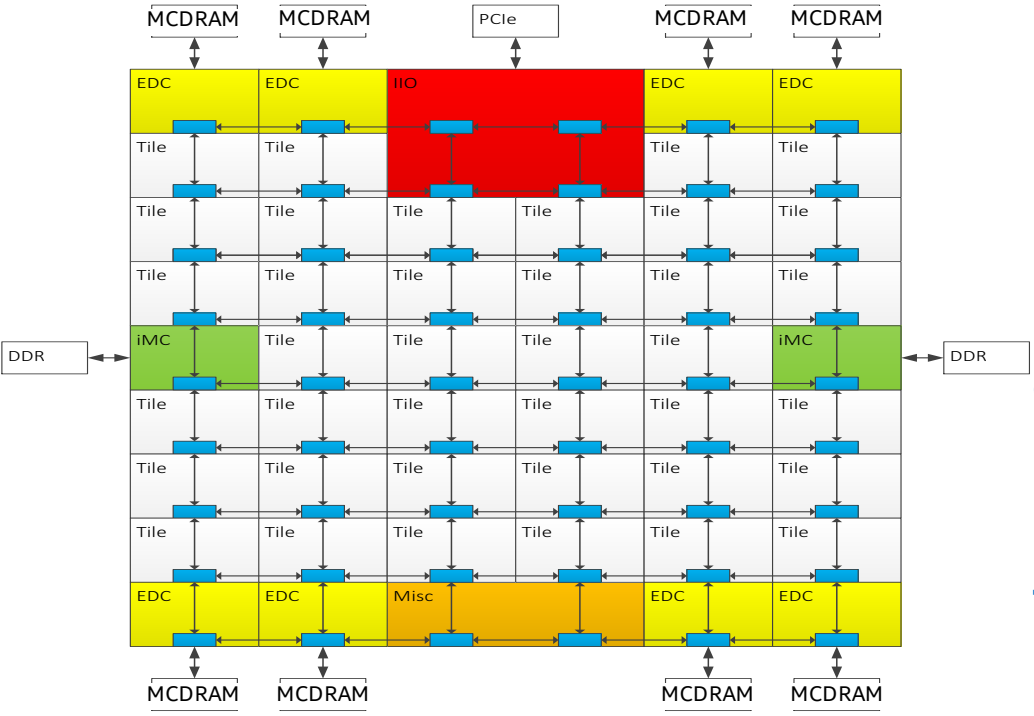
- 512-bit FP/Integer Vectors
- 32 regs, & 8 mask regs
- Gather/Scatter

**Conflict Detection**: Improves Vectorization

**Prefetch**: Gather and Scatter Prefetch

**Exponential and Reciprocal** Instructions

# KNL Mesh Interconnect




## Mesh of Rings

- Every row and column is a (half) ring
- YX routing: Go in Y → Turn → Go in X
- Messages arbitrate at injection and on turn

## Cache Coherent Interconnect

MESIF protocol (F = Forward)

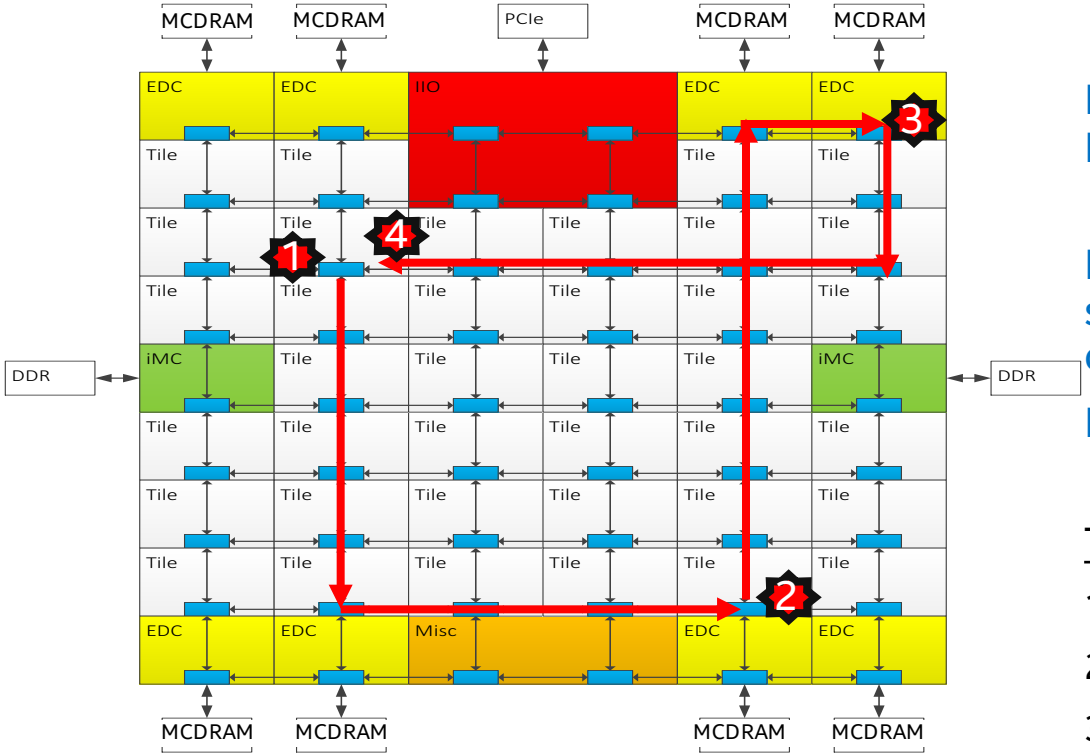
**Core in F state can provide cached shared data directly to requesting core without going to main memory.**

Distributed directory to filter snoops

## Three Cluster Modes

(1) All-to-All (2) Quadrant (3) Sub-NUMA Clustering

# Cluster Mode: All-to-All




**Address uniformly hashed across all distributed directories**

No affinity between Tile, Directory and Memory

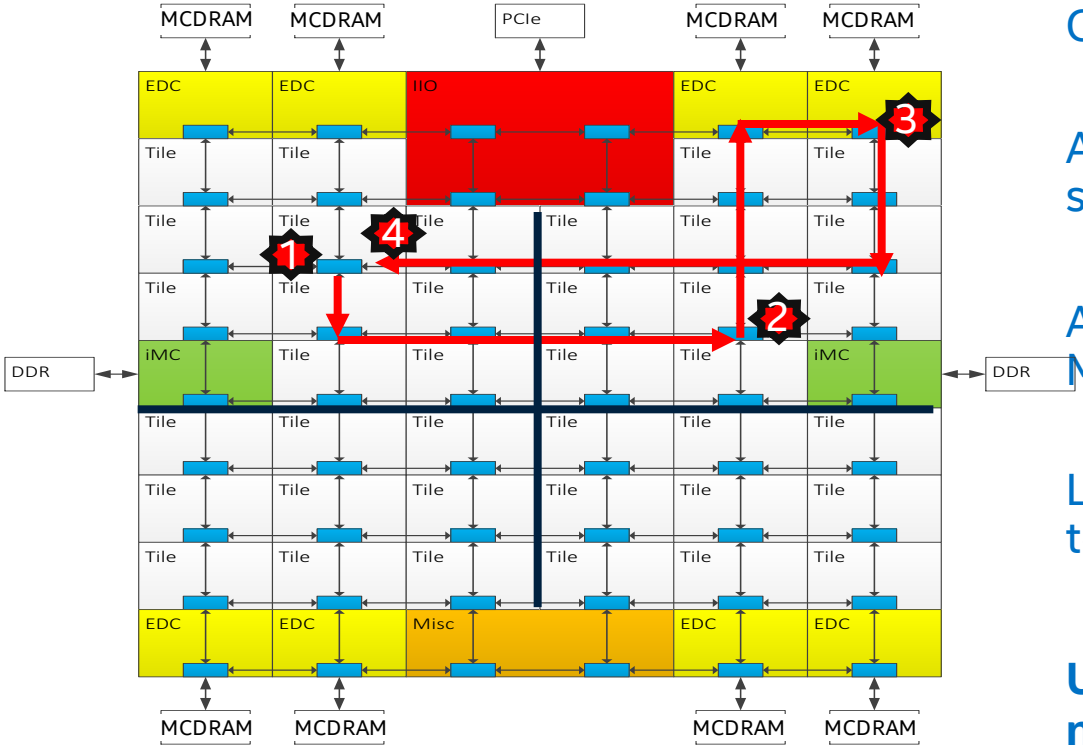
Most general mode. (Does not require same memory capacity on both DDR controllers.)

Lower performance than other modes.

Typical Read L2 miss

1. L2 miss encountered
2. Send request to the distributed directory
3. Miss in the directory. Forward to memory
4. Memory sends the data to the requestor

# Cluster Mode: Quadrant




Chip divided into four virtual Quadrants

Address hashed to a Directory in the same quadrant as the Memory

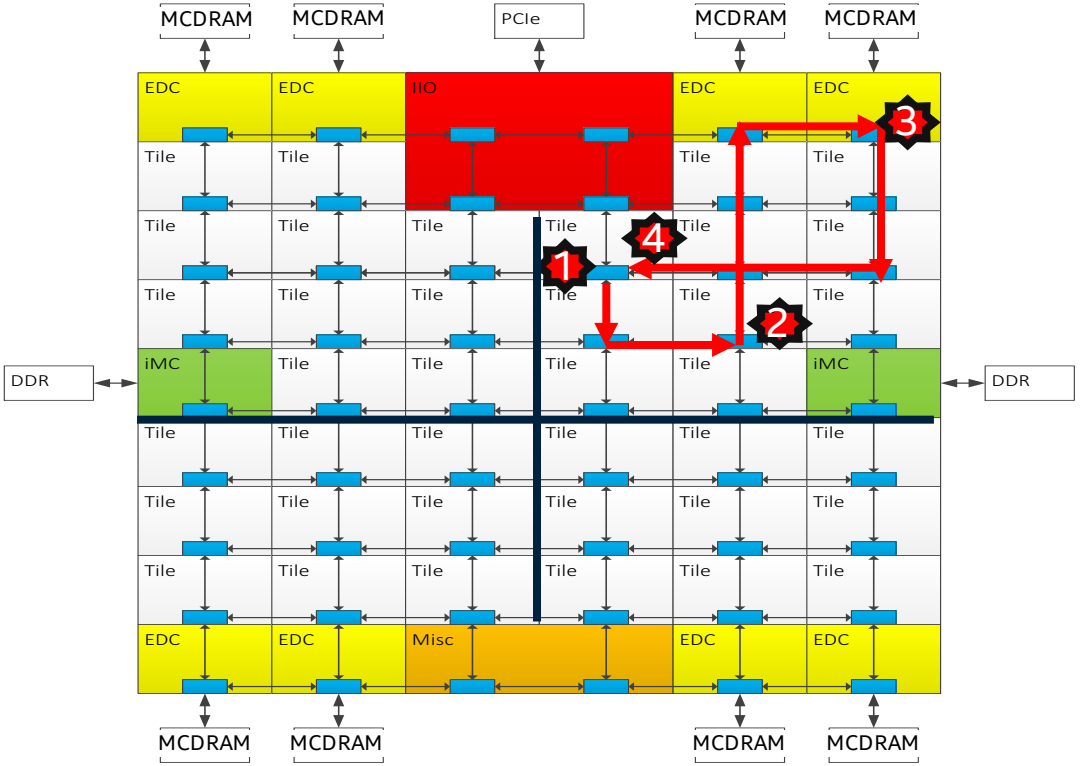
Affinity between the Directory and Memory

Lower latency and higher BW than all-to-all. SW Transparent.

**Used for all PETSc performance numbers in this presentation.**

1) L2 miss, 2) Directory access, 3) Memory access, 4) Data return

# Cluster Mode: Sub-NUMA Clustering (SNC)




Each Quadrant (Cluster) exposed as a separate NUMA domain to OS.

Looks analogous to 4-Socket Xeon

Affinity between Tile, Directory and Memory

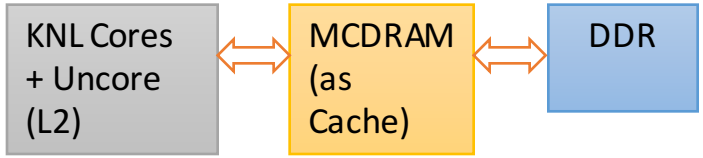
Local communication. Lowest latency of all modes.

SW needs to NUMA optimize to get benefit.

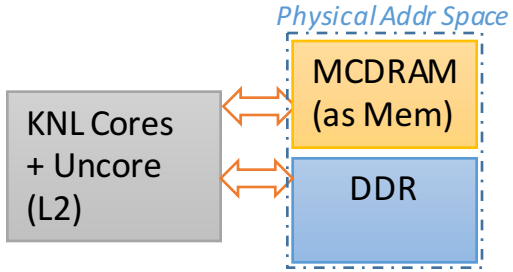
1) L2 miss, 2) Directory access, 3) Memory access, 4) Data return

# MCDRAM Modes

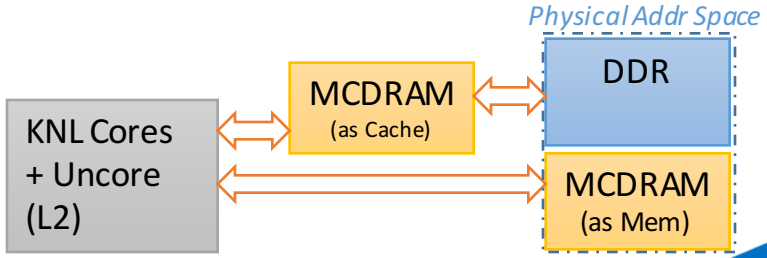
- Cache mode
  - No source changes needed to use
  - Misses are expensive (higher latency)
    - Needs MCDRAM access + DDR access




- Flat mode
  - MCDRAM mapped to physical address space
  - Exposed as a NUMA node
    - Use numactl --hardware, lscpu to display configuration
  - Accessed through memkind library or numactl



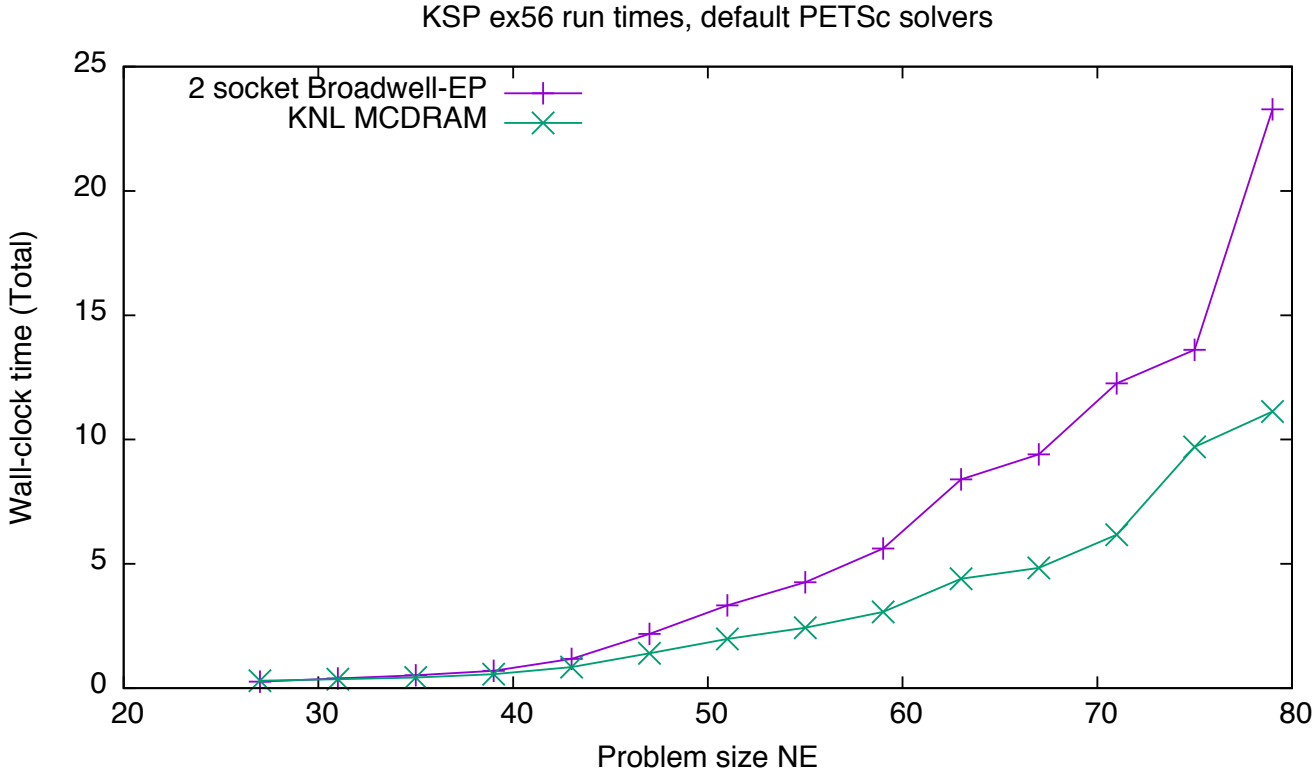

- Hybrid
  - Combination of the above two
    - E.g., 8 GB in cache + 8 GB in Flat Mode



# Outline

- Overview of “Knights Landing” (KNL) architecture
- Early experiences with PETSc on KNL
  - True “out-of-box” numbers: **No KNL-specific tuning yet**.
- Discussion of possible changes to PETSc for better manycore support
  - How to support multiple kinds of user-addressable memory?
    - Performance counters to assist good decision making?
  - Using new MPI-3?
  - More Intel® MKL support?

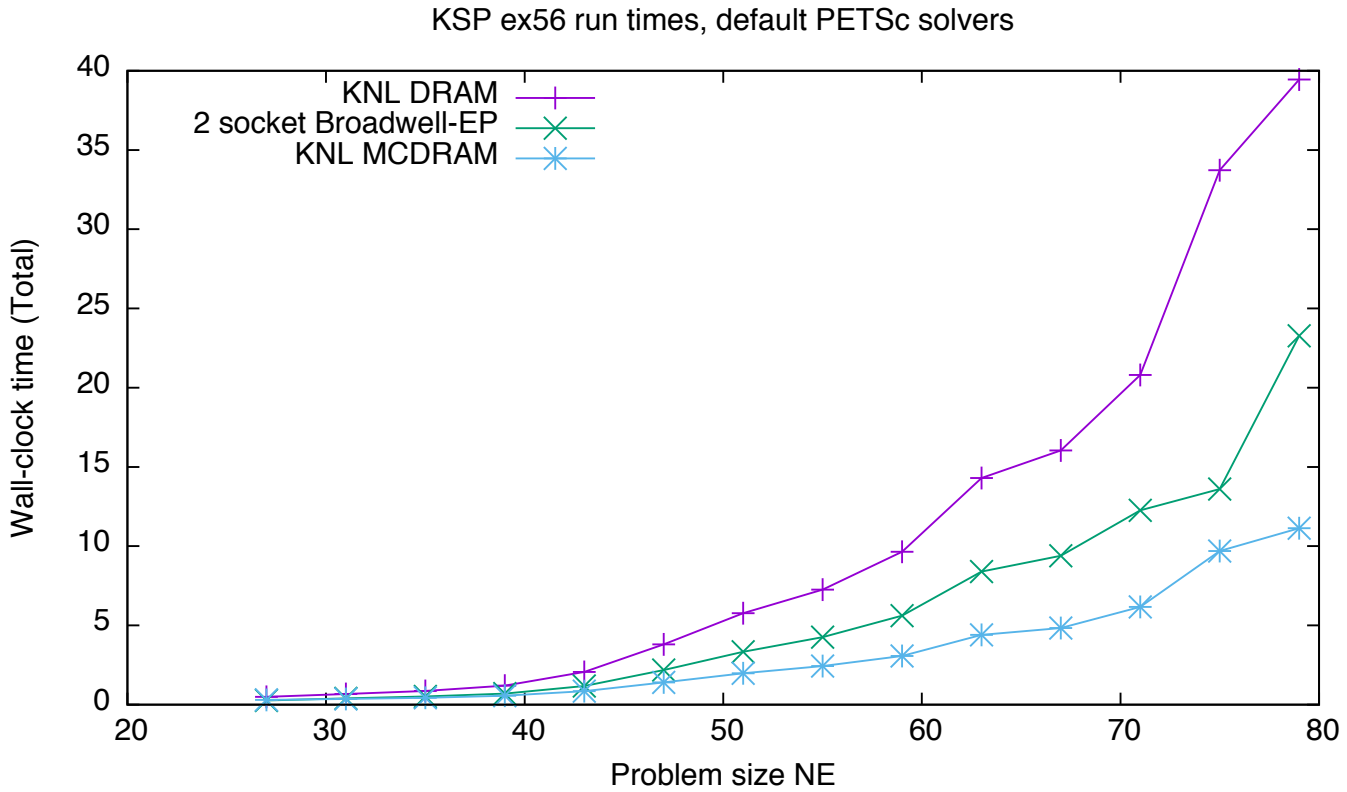
# KSP ex56 (linear elasticity), default solvers




KNL over 2X faster than dual-socket Intel® Xeon® processor E5-2697 v4 (“Broadwell-EP”) at largest problem size.

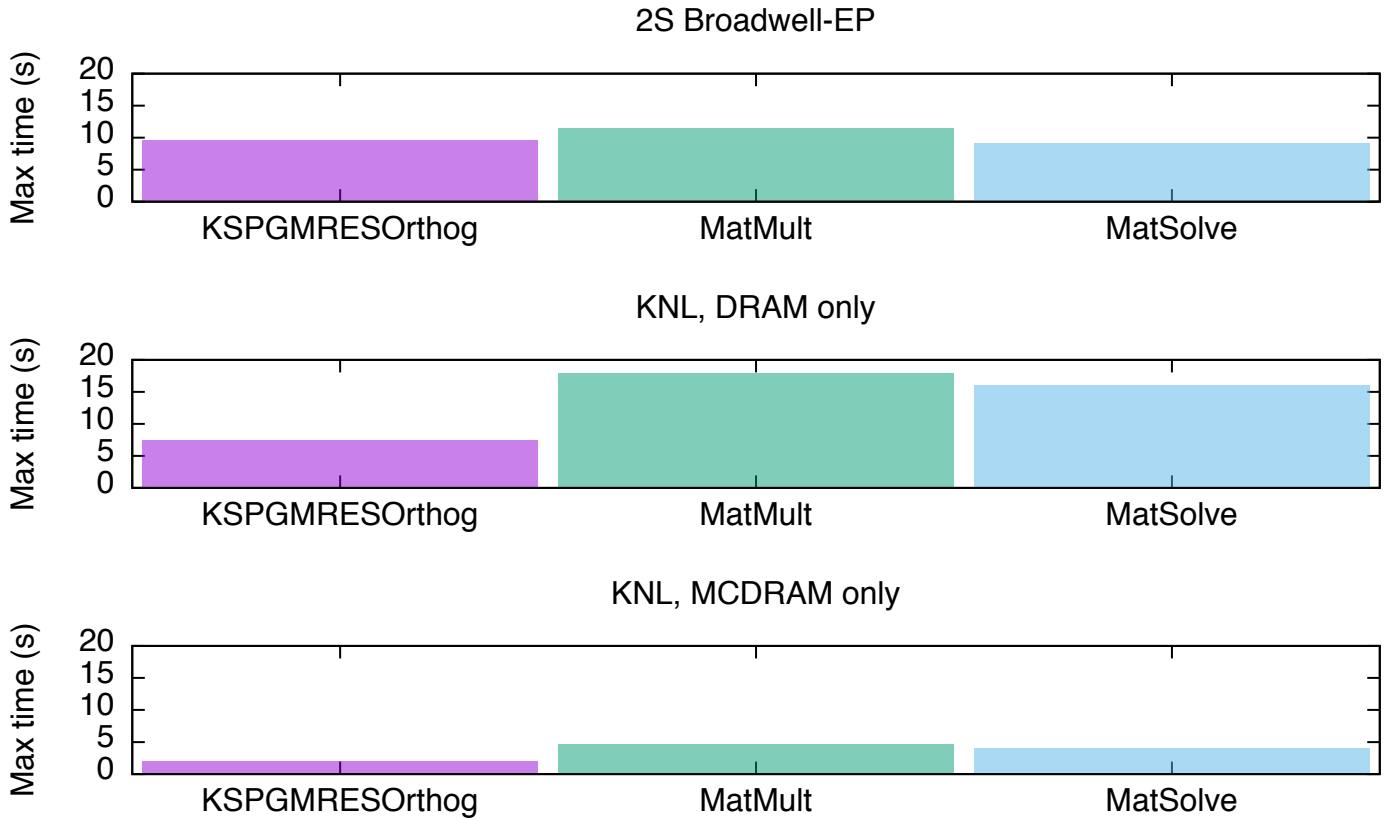
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you purchases, including the performance of that product when combined with other products. Any difference in system hardware or software design or configuration may affect actual performance. For more information go to http://www.intel.com/performance Configurations: Intel® Xeon Phi™ processor 7250 68 core, 272 threads, 1400 MHz core freq. Turbo mode ON, 1700 MHz uncore freq., MCDRAM 16 GB 7.2 GT/s, BIOS 10R00, DDR4 96GB 2400 MHz, Red Hat 7.2, quad cluster mode, MCDRAM flat memory mode; Dual Socket ® processor E5-2697 v4 2.3 GHz (Turbo OFF) , 18 Cores/Socket, 36 Cores, 72 Threads (HT on), DDR4 128GB, 2400 MHz, Red Hat 7.2

# KSP ex56 (linear elasticity), default solvers



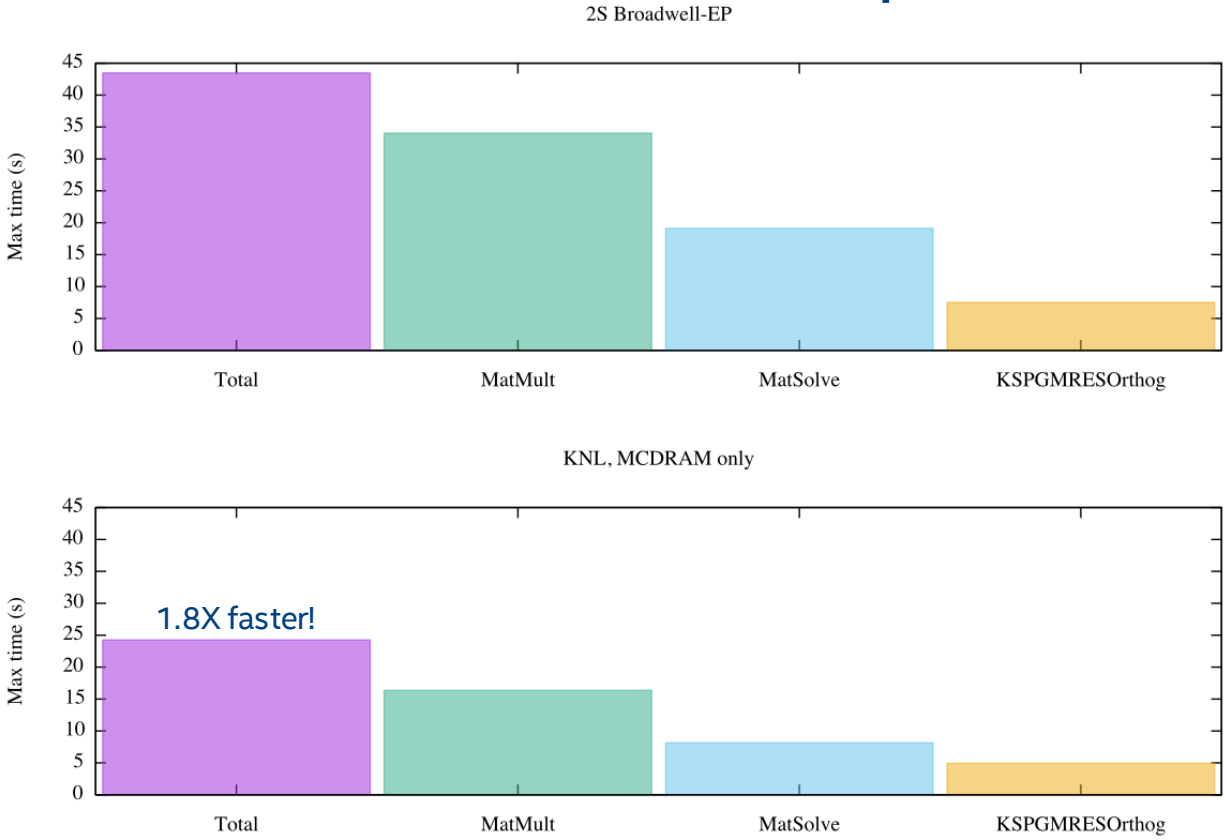

Much faster KNL solve times are (mostly) due to very high bandwidth of MCDRAM

# KSP ex56 (linear elasticity), default solvers




Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you purchases, including the performance of that product when combined with other products. Any difference in system hardware or software design or configuration may affect actual performance. For more information go to http://www.intel.com/performance Configurations: Intel® Xeon Phi™ processor 7250 68 core, 272 threads, 1400 MHz core freq. Turbo mode ON, 1700 MHz uncore freq., MCDRAM 16 GB 7.2 GT/s, BIOS 10R00, DDR4 96GB 2400 MHz, Red Hat 7.2, quad cluster mode, MCDRAM flat memory mode; Dual Socket ® processor E5-2697 v4 2.3 GHz (Turbo OFF) , 18 Cores/Socket, 36 Cores, 72 Threads (HT on), DDR4 128GB, 2400 MHz, Red Hat 7.2

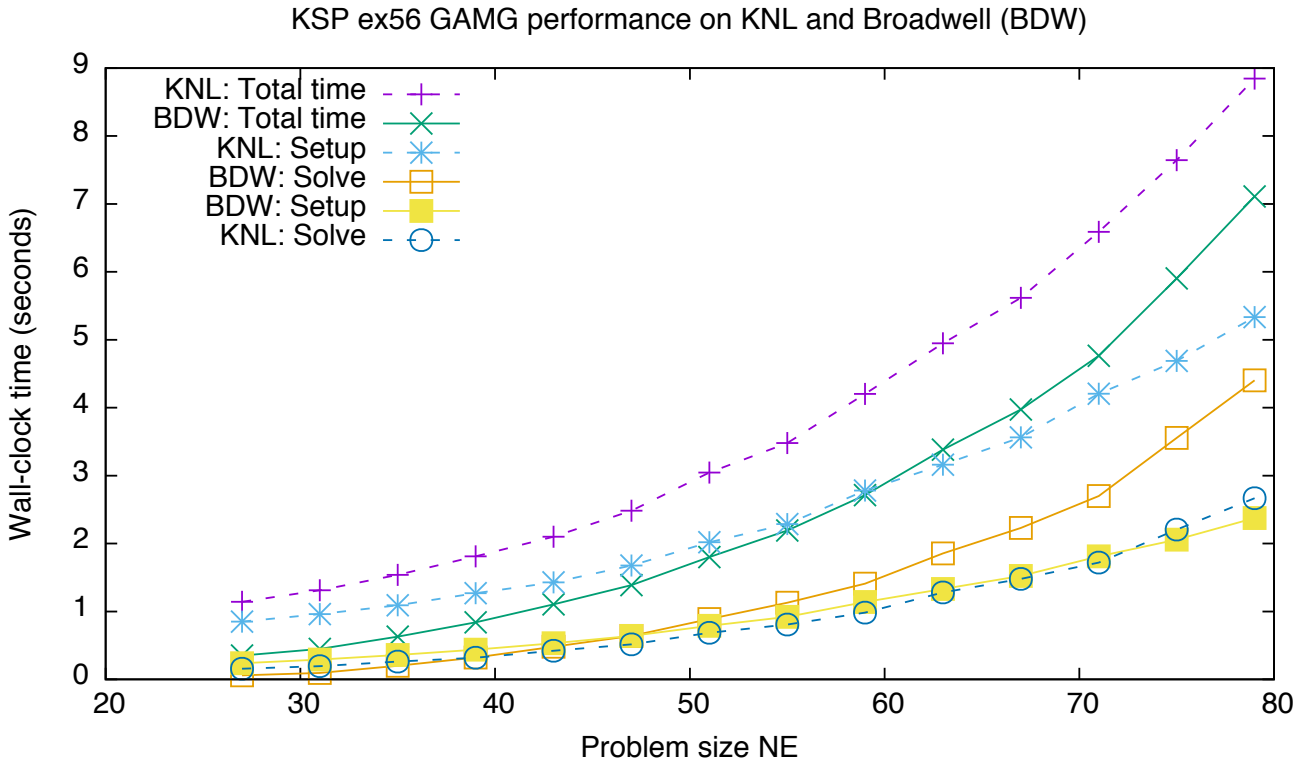
# KSP ex42 (Stokes flow), fieldsplit Schur




KNL run command: `mpirun -n 64 numactl --membind=1 ./ex42-intel2016_knl_fast -stokes_pc_type fieldsplit -stokes_pc_fieldsplit_type schur -log_view -mx 60`

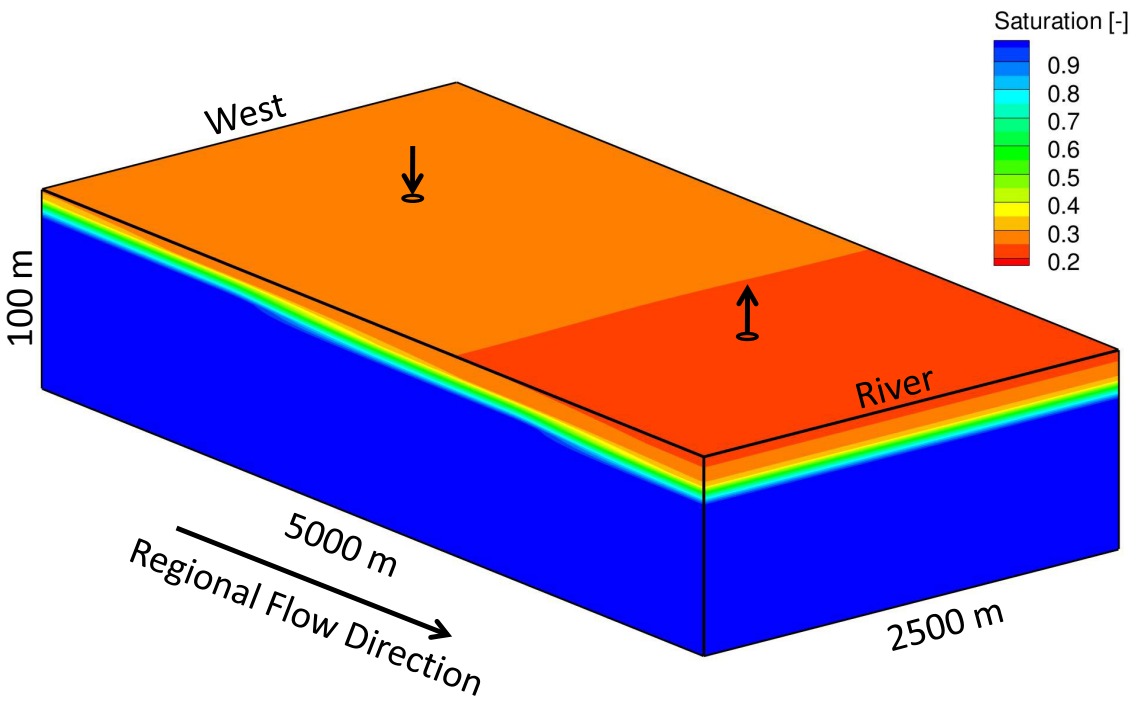
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you purchases, including the performance of that product when combined with other products. Any difference in system hardware or software design or configuration may affect actual performance. For more information go to http://www.intel.com/performance Configurations: Intel® Xeon Phi™ processor 7250 68 core, 272 threads, 1400 MHz core freq. Turbo mode ON, 1700 MHz uncore freq., MCDRAM 16 GB 7.2 GT/s, BIOS 10R00, DDR4 96GB 2400 MHz, Red Hat 7.2, quad cluster mode, MCDRAM flat memory mode; Dual Socket ® processor E5-2697 v4 2.3 GHz (Turbo OFF) , 18 Cores/Socket, 36 Cores, 72 Threads (HT on), DDR4 128GB, 2400 MHz, Red Hat 7.2

# KSP ex56, GMRES + GAMG



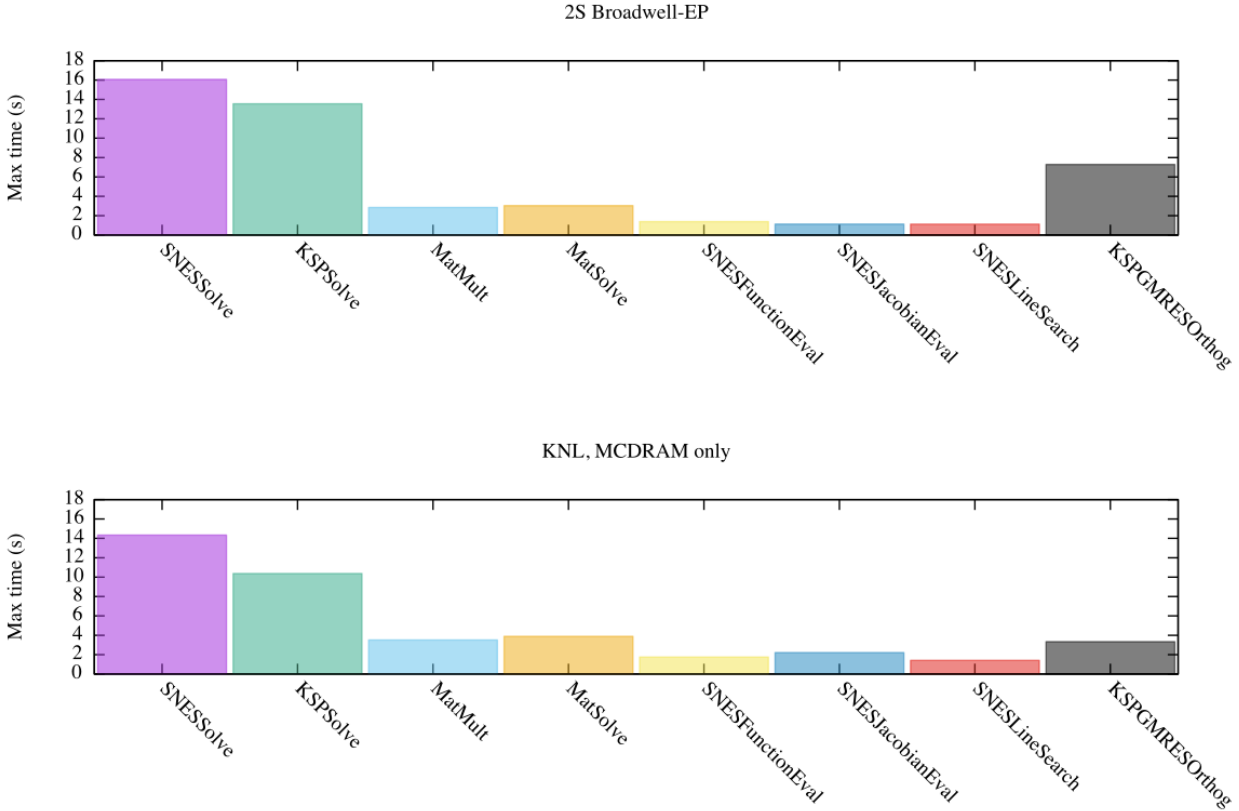

- "Solve" phase is quite fast on KNL
- Unoptimized setup is comparatively slow on KNL

# PFLOTRAN Regional Doublet




- PFLOTRAN problem analyzed in 2014 *WRR* paper (doi: 10.1002/2012WR013483)
- Variably saturated regional groundwater flow
- First order FV in space, backward Euler in time.
- Used inexact Newton with GMRES(30), block Jacobi, ILU(0) on blocks
- Used 200 x 200 x 100 grid (4 million total degrees of freedom)

# PFLOTRAN Regional Doublet




- BDW vs KNL total times comparable
- Orthogonalizations much faster on KNL
- But MatMult and MatSolve faster on BDW!
- Small work per row (~ 7 nonzeros; compare to ~80 in KSP ex56) probably unable to mask latency of gathering x vector; reordering may help.
- Jacobian formation faster on BDW; vectorizaton work on KNL probably needed

# Outline

- Overview of “Knights Landing” (KNL) architecture
- Early experiences with PETSc on KNL
  - True “out-of-box” numbers: **No KNL-specific tuning yet**.
- Discussion of possible changes to PETSc for better manycore support
  - How to support multiple kinds of user-addressable memory?
    - Performance counters to assist good decision making?
  - Using new MPI-3?
  - More Intel® MKL support?

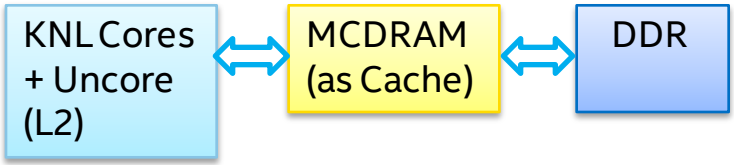
# MCDRAM as Cache

## Upside

- No software modifications required to get bandwidth benefit (over DDR)

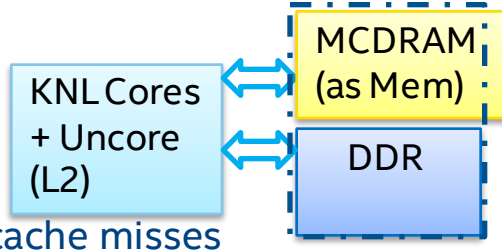
## Downside

- Less addressable memory
- All memory is transferred as:
  - DDR -> MCDRAM -> L2
- Misses need MCDRAM + DRAM access
- Unpredictable performance due to conflict misses when physical memory becomes fragmented.




# MCDRAM as Flat Mode

## Upside

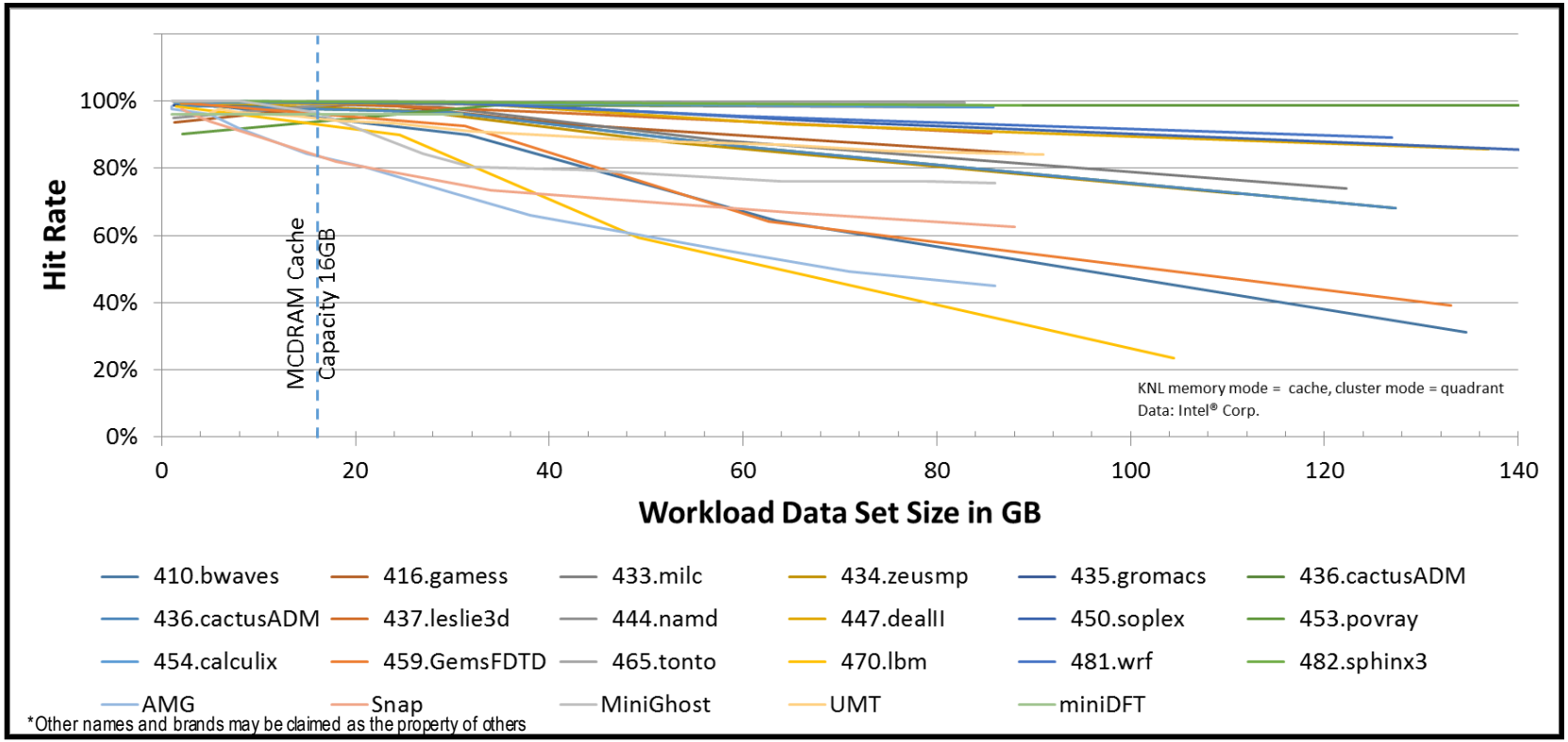



- Maximum BW
- Lower latency
  - i.e., no MCDRAM cache misses
- Maximum addressable memory
- Isolation of MCDRAM for high-performance application use only

## Downside

- Software modifications (or interposer library) required
  - to use DDR and MCDRAM in the same app
- Which data structures should go where?
- MCDRAM is a finite resource and tracking it adds complexity

# MCDRAM Cache Hit Rate



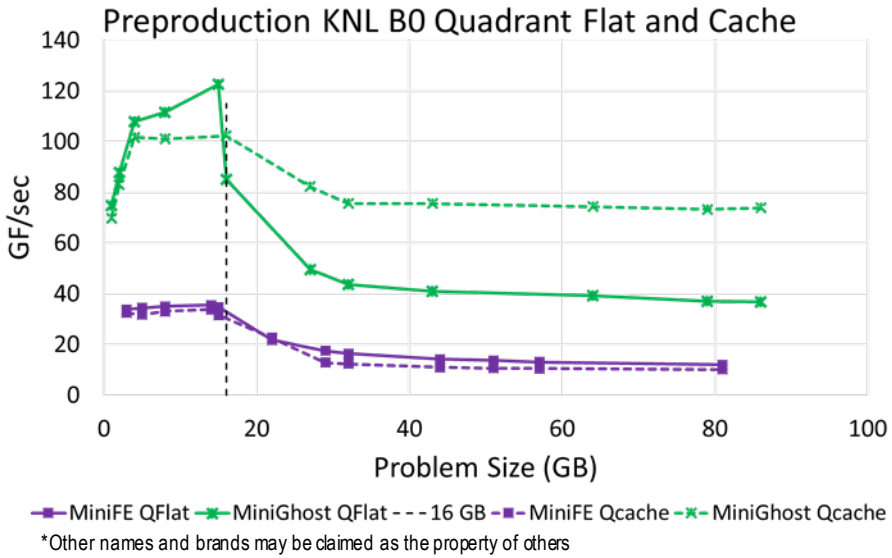

*Other names and brands may be claimed as the property of others

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you purchases, including the performance of that product when combined with other products. Any difference in system hardware or software design or configuration may affect actual performance. For more information go to http://www.intel.com/performance Configurations: Intel SPECrate* and Trinity* tests running 1P Xeon Phi 7250 (KNL 68C), Quad-Cache, default P*ratios (P0=1.6, P1=1.4, PN=1.0), ICC-16.0 internal compiler release, AVX-512, 09.D03 BIOS

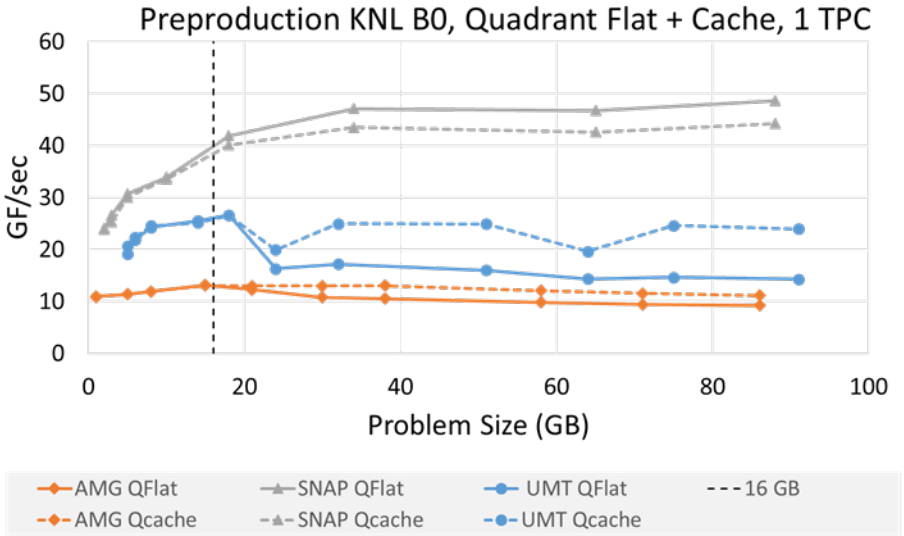
MCDRAM performs well as cache for many workloads

Enables good out-of-box performance without memory tuning

# Performance for Flat vs. Cache Mode




*Other names and brands may be claimed as the property of others




Source: Antonio Valles

## Flat vs. Cache performance depends on workloads. Up to +60% performance, but in some cases lower performance (HW does better job managing MCDRAM)

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you purchases, including the performance of that product when combined with other products. Any difference in system hardware or software design or configuration may affect actual performance. For more information go to http://www.intel.com/performance Configurations: Intel SPECrate* and Trinity* tests running 1P Xeon Phi 7250 (KNL 68C), Quad-Cache, default P*ratios (P0=1.6, P1=1.4, PN=1.0), ICC-16.0 internal compiler release, AVX-512, 09.D03 BIOS

# Using user-addressable high-bandwidth memory

In flat mode, MCDRAM is exposed as a separate NUMA node:
(libnuma, mmap() work just as for any NUMA node.)


- If footprint fits, can place entire application in MCDRAM using numactl(8).
- Can use AutoHBW interposer library to do automatic size threshold-based placement in MCDRAM.
- Can do explicit placement:
  - Fortran: !DEC$ ATTRIBUTES, FASTMEM :: A
  - C: `fv = (float *)hbw_malloc(sizeof(float) * 100),`
    or use underlying memkind (https://github.com/memkind) library:
    `a = (float *)memkind_malloc(MEMKIND_HBW_PREFERRED, size);`

# Supporting multiple kinds of memory in PETSc

- Immediate concern is supporting user-addressable DRAM and MCDRAM. But additional types of memory (e.g., 3D Xpoint™ NVRAM) will add further complications.
- Simple option: Use size thresholds for automatic greedy allocation of high-bandwidth memory.
  - AutoHBW interposer (part of memkind library) does this, but we may want ability to restrict such placement decisions to inside PETSc.
  - Largest data structures tend to be most bandwidth-intensive; small ones can fit in cache

# Supporting multiple kinds of memory in PETSc

- More complicated: Add PetscAdvMalloc() that accepts an advisory context, provide way to tag objects (Vec, Mat) to be used w/ associated malloc()s.
  - Problem: Making the right placement decisions based on a priori reasoning may be impossible
  - Do placement/migration based on measured importance.
  - Barry suggests counting VecGetArray[Read]()s.
    - Perhaps also use ancillary data from hardware counters (e.g., memory bandwidth measurements via uncore counters)?
    - I propose adding some hardware counter support in the PETSc logging framework
  - In many cases, decisions need to account for what is going on external to PETSc as well. Optimal approach may need OS and/or middleware support.
    - Linux provides move_pages(2), but this is not asynchronous

# KNL and successors will be deployed in large systems

## NERSC Cori

- ~ 1400 dual socket nodes w/ **Intel®** **Xeon®** v3 (“Haswell”) Processors, 16 cores per socket
- Over 9,300 single socket nodes w/ 2nd gen Intel Xeon Phi Processors (“Knights Landing”—KNL), w/ up to 16GB on-package, high-bandwidth memory
- Cray Aries dragonfly topology interconnect

## ALCF Aurora

- Over 50,000 nodes with 3rd gen **Intel®** **Xeon Phi™** Processors
- Over 8 PB aggregate on-package high-bandwith memory and persistent memory
- 2nd gen **Intel®** Omni-Path Architecture with silicon photonics
- Intel® Lustre* filesystem, > 1 TB/s throughput


# Exposing Concurrency on Many Levels

On prior leadership-class machines, near-exclusive focus on flat MPI optimizations was often sufficient.

Machines like Cori and Aurora require attention to many levels of concurrency:

1. Within single threads (vectorization)
   - *Strategies similar to those on AVX2 CPUs apply.*
2. Across shared memory on a node (usually OpenMP threads)
   - *Codes often use OpenMP, but PETSc has returned to MPI only.*
3. And across nodes on the interconnect (usually MPI)
   - *MPI-3 introduces several new features that can help.*
   - *At scale, may ultimately need to re-think algorithms to reduce global communications.*

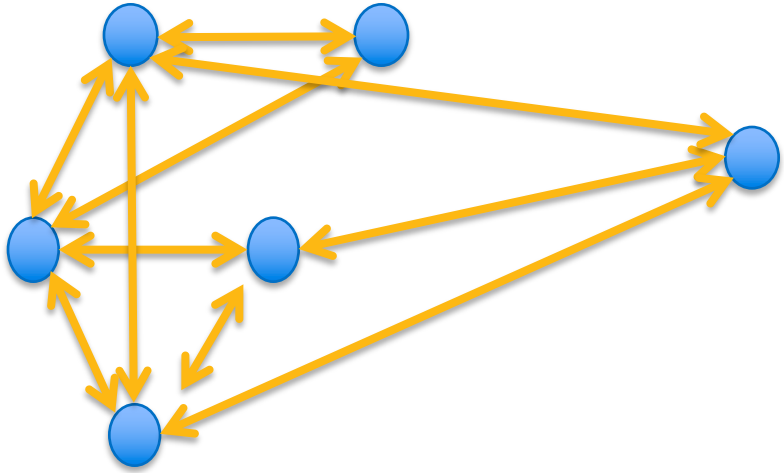
     (E.g., McInnes et al. 2014, Hierarchical Krylov and Nested Krylov Methods for Extreme-Scale Computing, http://www.mcs.anl.gov/papers/P2097-0612.pdf)

Important: Must consider #2 with #3! MPI communications a big source of serialization.

# Leveraging MPI-3 features for manycore MPPs

- MPI-3 introduces several features well-suited to manycore MPPs. Two especially useful features (supported in Intel MPI 5.x):
  - Shared memory windows
  - Neighborhood collectives
- Can be used in conjunction with OpenMP, or used to develop "MPI+MPI" hybrid shared/distributed memory MPI-only applications.

# Graph Topologies and Neighborhood Collectives



MPI-3 introduces distributed graph topologies to allow expression (in a scalable way) of any communication pattern to the runtime.

(Boundary element exchange as N Isend-Irecv + Waitall is perhaps the most common messaging pattern)

**Neighborhood collectives** perform communications specified on graph topologies. Knowing pattern in advance enables several optimizations, e.g.,

- Persistent allocation of network resources
- Intelligent scheduling (accounting for factors like **transport over shared memory vs. off-node network**)

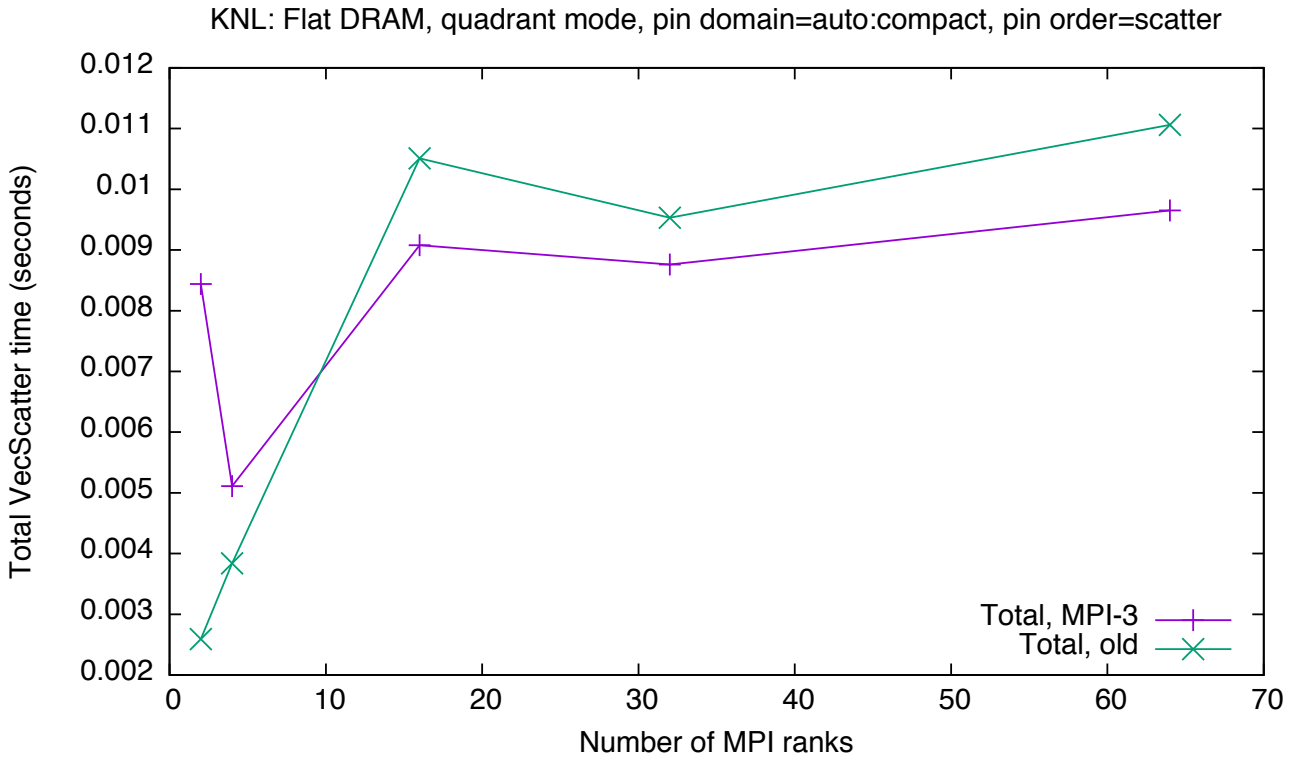
While simplifying code by expressing communications with a single call.

# Leveraging MPI-3 in PETSc

- Use neighborhood collectives.
  - Add neighborhood collectives implementation for PetscSF.
  - Add VecScatter implementation on top of PetscSF.
- Support MPI_Win_allocate_shared() in a DM.
  - Extend concept of a “local” vector from PETSC_COMM_SELF to a shared-memory communicator from MPI_Comm_split_type()?
  - Or, rather, abstract to several levels of “local”: Private to a rank, shared within a NUMA domain, shared within a node, …
  - If user desires to avoid duplicating halos inside shared memory regions, can we do so while preserving nice indexing of ghost points?
- Support direct load-store for “local” portions VecScatter
  - Barry has prototyped this in barry/utilize-hwloc.

# Leveraging MPI-3 in PETSc

In branch barry/utilize-hwloc: $PETSC_DIR/src/ksp/ksp/examples/tests/benchmarkscatters/ex1.c



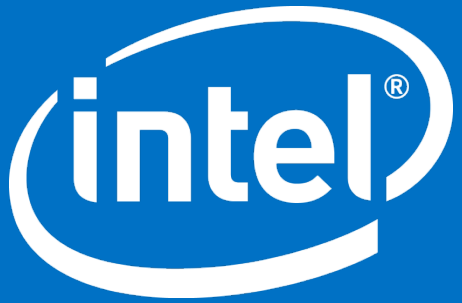

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you purchases, including the performance of that product when combined with other products. Any difference in system hardware or software design or configuration may affect actual performance. For more information go to http://www.intel.com/performance
Configurations: Intel® Xeon Phi™ processor 7250 68 core, 272 threads, 1400 MHz core freq. Turbo mode ON, 1700 MHz uncore freq., MCDRAM 16 GB 72 GT/s, BIOS 10R00, DDR4 96GB 2400 MHz, Red Hat 7.2, quad cluster mode, MCDRAM flat memory mode; Dual Socket ® processor E5-2697 v4 2.3 GHz (Turbo OFF) , 18 Cores/Socket, 36 Cores, 72 Threads (HT on), DDR4 128GB, 2400 MHz, Red Hat 7.2

# Low-level optimizations for KNL

- Good use of vectorization is critical for getting best performance on KNL.
- No PETSc developer is going to want the PETSc source code full of AVX512 intrinsics or even a lot of #pragmas.
  - Don't have manpower for such low-level optimizations anyway.
- Proposal: Add new AIJ matrix subclass ("MATAIJMKL")
  - Use Intel® MKL implementations inside MatMult(), MatSolve(), MatMatMult(), MatPtAP(), etc.
  - Support new sparse inspector-executor routines in Intel® MKL.

# Summary and Future Directions

- KNL provides “accelerator”-like manycore parallelism and energy efficiency without the “accelerator”.
- True “out-of-box” good performance with many PETSc applications.
- High-bandwidth on-package memory (MCDRAM) provides big boost to PETSc applications using assembled matrices.
- Performance should improve as we explore KNL-targeted optimizations in PETSc.
- Recruiting “fellow travelers”: Collaborators welcome!


intel®