PERFORMANCE OPTIMIZATION FOR INTEL® XEON PHI™ X200 PRODUCT FAMILY

The Hands-On Workshop (HOW) Series

Colfax International — *colfaxresearch.com*

September 2016

LIST OF TOPICS

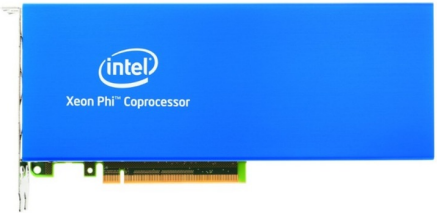
- **Part 1**: Meet Intel Xeon Phi Processors
  - Intel Architecture: Today and Tomorrow
  - Cores in Intel Xeon Phi Processors
  - Vector Instruction Support
  - High-Bandwidth Memory
  - Clustering Modes
- **Part 2**: Hands-On Demonstrations
  - Memory Bandwidth Optimization
  - Vectorization with AVX-512
  - Tuning with Intel Math Kernel Library
  - Machine Learning on Intel Architecture
  - Where to Learn More

*formerly Knights Landing

§1. MEET INTEL XEON PHI PROCESSORS

INTEL ARCHITECTURE: TODAY AND TOMORROW

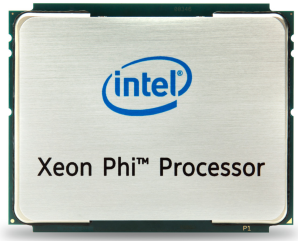
INTEL ARCHITECTURE

Intel Xeon Processor	Intel Xeon Phi Coprocessor, 1st generation	Intel Xeon Phi Processor, 2nd generation*
		
Current: Broadwell Upcoming: Skylake	Knights Corner (KNC)	* socket and coprocessor versions Knights Landing (KNL)
Multi-Core Architecture	Intel Many Integrated Core (MIC) Architecture	

INTEL XEON PHI PROCESSORS (2ND GEN)

Specialized platform for demanding computing applications.

- Socket version or coprocessor
- 64-72 cores × 4 HT at 1.3-1.5 GHz
- 3+ TFLOP/s in DP (FMA)
- 6+ TFLOP/s in SP (FMA)
- ≤ 384 GiB DDR4 (> 90 GB/s)
- 16 GiB HBM (MCDRAM, > 400 GB/s)
- Binary-compatible with Xeon
- Common OS (RHEL/CentOS/Fedora/SUSE/Windows)



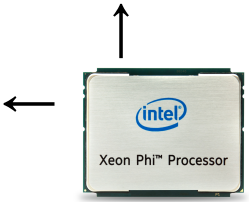
BOOTABLE INTEL XEON PHI PROCESSORS

- Bootable Host Processor
- RHEL/CentOS/SUSE/Win
- 64 cores × 4 HT, 1.3 GHz
- ≤ 384 GiB DDR4, > 90 GB/s
- 16 GiB HBM, > 400 GB/s
- PCIe bus for networking

dap.xeonphi.com

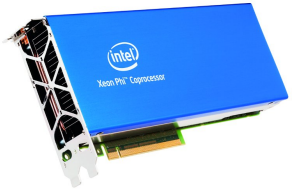
Servers:

Workstations:



KNLF: KNL with Fabric

- Fabric integrated on CPU
  - Intel® Omni-Path Architecture
- Socket mount processor

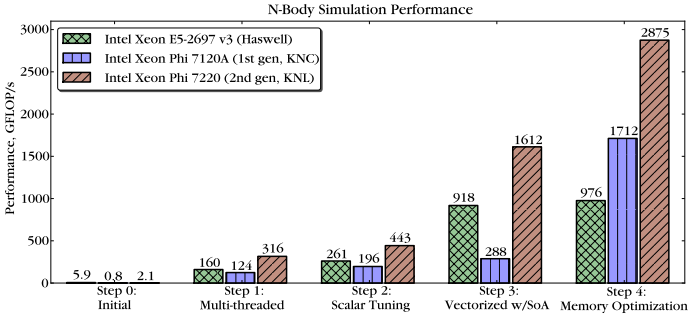


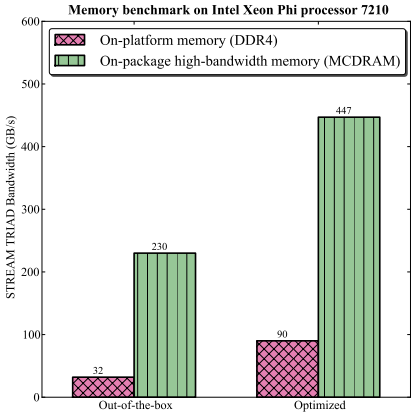
*KNC image

KNL Coprocessor

- PCIe add-in card
  - Requires host
- Multiple KNLs in a system

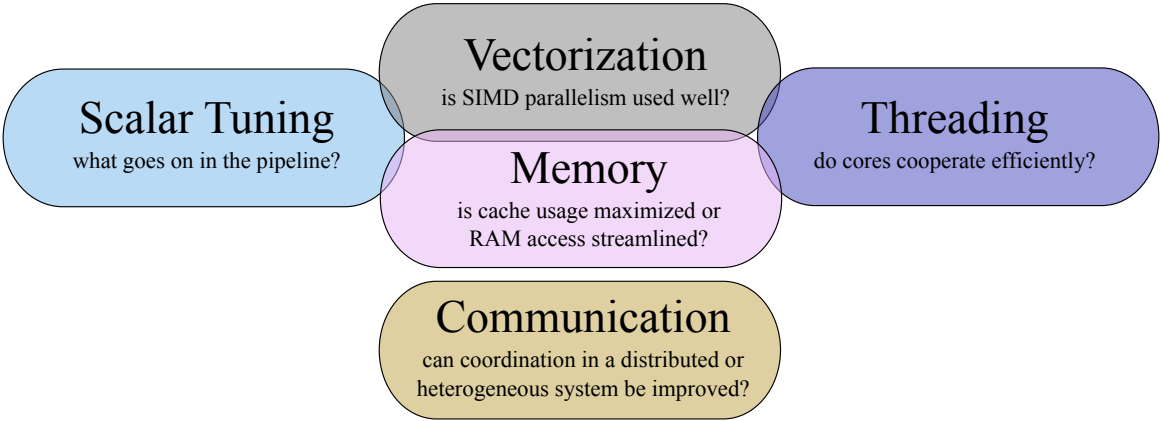
IT TAKES GOOD SOFTWARE TO UNLOCK THE PERFORMANCE!





Details on N-body simulation in Chapter 23 of this book: lotsofcores.com/KNLbook

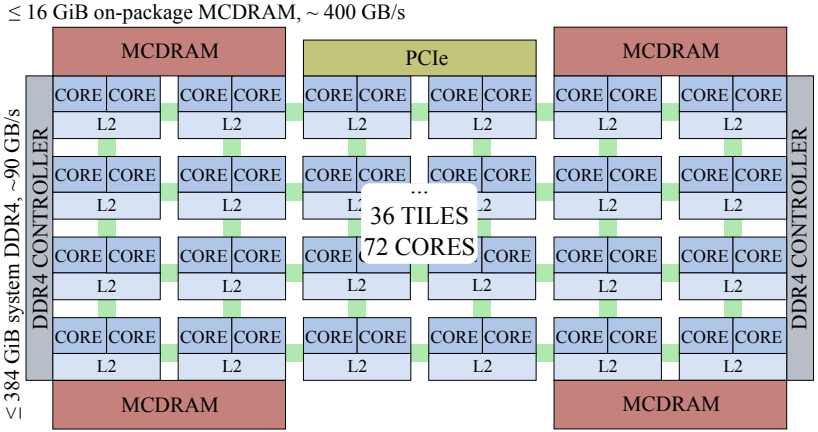
OPTIMIZATION AREAS



CORES IN INTEL XEON PHI PROCESSORS

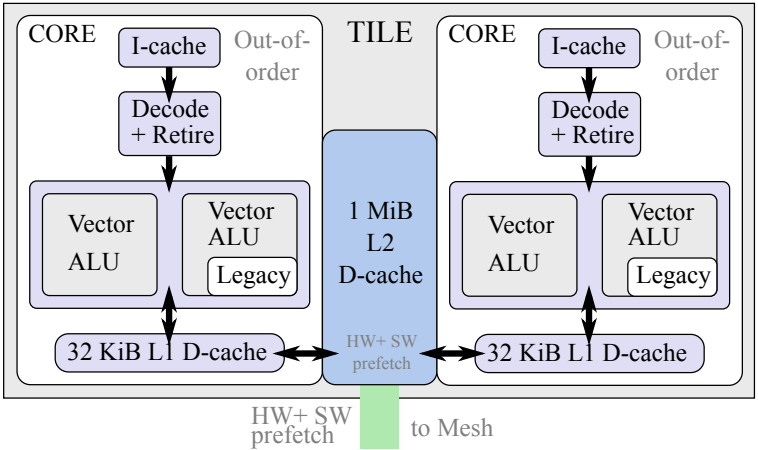
KNL DIE ORGANIZATION: TILES

- Up to 36 tiles, each with 2 physical cores (72 total).
- Distributed L2 cache across a mesh interconnect.

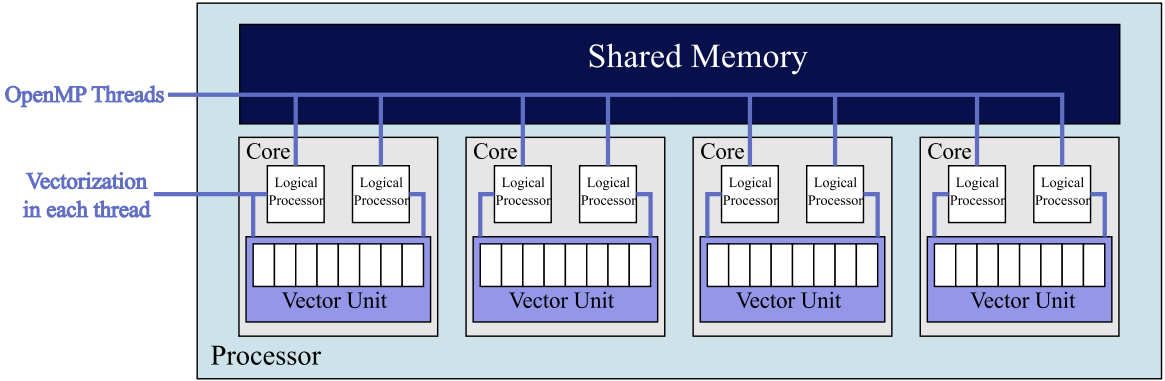


KNL CORES

- 4-way hyper-threading (up to $4 \times 72 = 288$ logical processors)
- L2 cache shared by 2 cores on a tile.



CO-EXISTENCE WITH VECTORS



Utilize cores: run multiple threads/processes (MIMD)

Utilize vectors: each thread (process) issues vector instructions (SIMD)

MORE FORGIVING CORES THAN FIRST GENERATION (KNC)

Based on Intel® Atom cores (Silvermont microarchitecture)

- **Out-of-order cores:**
  Better latency masking from long latency operations
- **Back-to-back instructions from single thread:**
  Thread count requirement reduced to ≈ 70 (from ≈ 120 for KNC)
- **Advanced branch prediction:**
  Fewer cycles wasted to branch misprediction

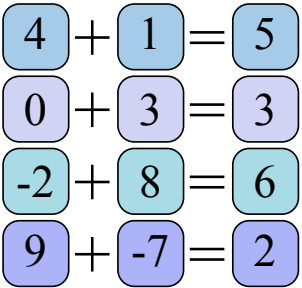
Generally more forgiving to non-optimized code.

VECTOR INSTRUCTION SUPPORT

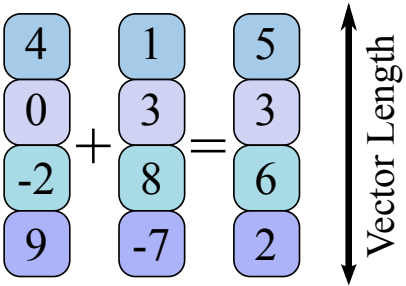
SHORT VECTOR SUPPORT

Vector instructions – one of the implementations of SIMD (Single Instruction Multiple Data) parallelism.

Scalar Instructions



Vector Instructions



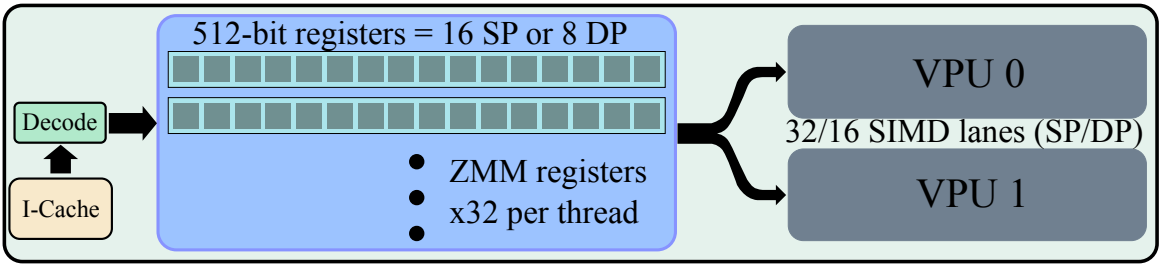
DUAL VPU

Each core has two Vector Processing Units (VPUs).

Penalty from non-vectorized code

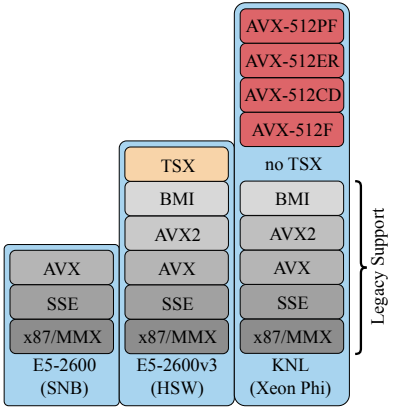
SP → 512 bit registers / 32 bits × 2 VPUs = **32** SIMD lanes

DP → 512 bit registers / 64 bits × 2 VPUs = **16** SIMD lanes



SUPPORTED INSTRUCTION SETS

- Intel® Advanced Vector Extensions 512 (AVX-512)
  - 512-bit vector registers.
  - Hardware gather/scatter, DP transcendental functions support and more.
  - Supported by non-Intel compilers like GCC.
- ≤ Intel® AVX2
  - Legacy mode operation.
  - Binary compatibility with Xeon.
  - Does *not* include IMCI (from KNC).



AVX-512 FEATURES

- AVX-512F (Fundamentals)
  - Extension of most AVX2 instructions to 512-bit vector registers.
- AVX-512CD (Conflict Detection)
  - Efficient conflict detection (application: binning).
- AVX-512ER (Exponential and Reciprocal)
  - Transcendental function (`exp`, `rcp` and `rsqrt`) support.
- AVX-512PF (Prefetch)
  - Prefetch for scatter and gather.

Learn more: colfaxresearch.com/avx-512

USING AVX-512: TWO APPROACHES

Automatic Vectorization:

- Vectorization w/ compiler.
- Portable: just recompile.
- Tuning with directives.

```
1 double A[vec_width], B[vec_width];
2 // ...
3
4
5 // This loop will be auto-vectorized
6 for(int i = 0; i < vec_width; i++)
7   A[i]+=B[i];
```

```
1 double A[vec_width], B[vec_width];
2 // ...
3 // This is explicitly vectorized
4 __m512d A_vec = _mm512_load_pd(A);
5 __m512d B_vec = _mm512_load_pd(B);
6 A_vec = _mm512_add_pd(A_vec,B_vec);
7 _mm512_store_pd(A,A_vec);
```

Explicit Vectorization:

- Vectorization w/ intrinsics
- Full control over instructions
- Limited portability

GCC SUPPORT FOR AVX-512

GCC ≥ 4.9.1 supports AVX-512 instruction set.

```
user@knl% g++ -v
gcc version 4.9.2 (GCC)
user@knl% g++ foo.cc -mavx512f -mavx512er -mavx512cd -mavx512pf
```

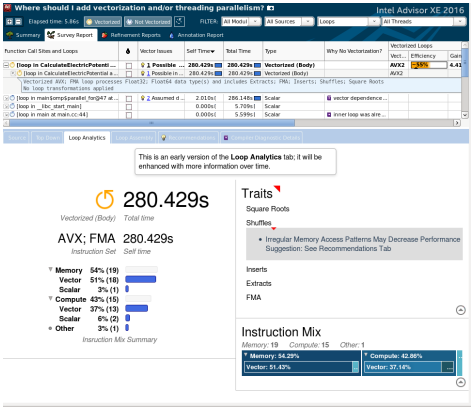
Basic automatic vectorization support: add -O3.

```
1 // ... foo.cc ... //
2 for(int i = 0; i < n; i++)
3   B[i] = A[i] + B[i];
```

```
user@knl% g++ -s foo.cc -mavx512f -O3
user@knl% cat foo.s
...
  vmovapd -16432(%rbp,%rax), %zmm0
  vaddpd -8240(%rbp,%rax), %zmm0, %zmm0
  vmovapd %zmm0, -8240(%rbp,%rax)
```

PERFORMANCE CONSIDERATIONS

Even if your code is vectorized, tuning may unlock more performance.



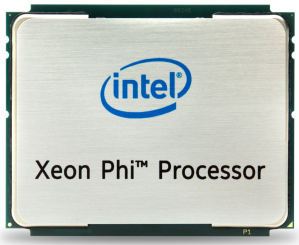
download paper to learn more

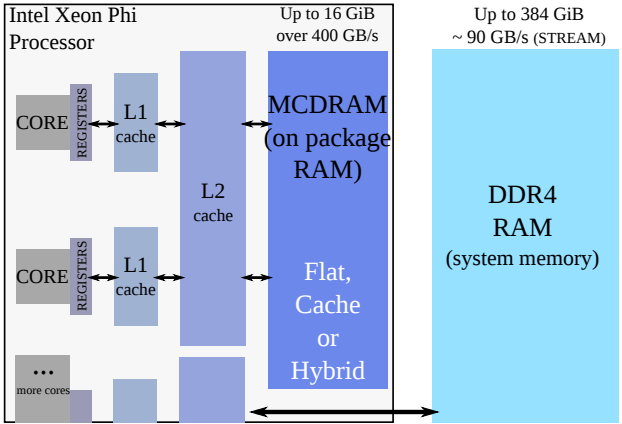
- Providing enough parallelism.
  - More consecutive vector operations required to overcome vectorization latency.
- Loop pipelining and unrolling.
  - Double the pipeline stages to populate.
- Better vectorization patterns.
  - Avoid long latency operations with unit-stride and unmasked operations.

HIGH-BANDWIDTH MEMORY

KNL MEMORY ORGANIZATION

- Direct access to on-package MCDRAM *and* system DDR4 (socket)
- Use MCDRAM as cache, in flat mode, or as hybrid

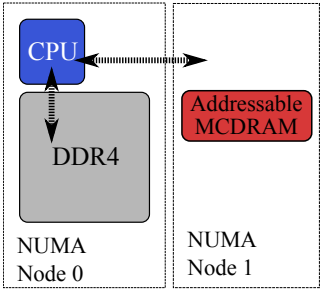




MCDRAM MEMORY MODES

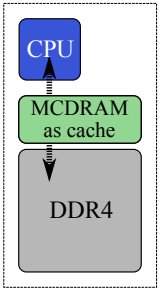
Flat Mode

- MCDRAM treated as a NUMA node
- Users control what goes to MCDRAM



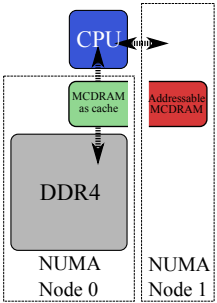
Cache Mode

- MCDRAM treated as a Last Level Cache (LLC)
- MCDRAM is used automatically



Hybrid Mode

- Combination of Flat and Cache
- Ratio can be chosen in the BIOS



RUNNING APPLICATIONS IN HBM WITH NUMACTL

- Finding information about the NUMA nodes in the system.

```
user@knl% # In Flat mode of MCDRAM
user@knl% numactl -H
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 ... 254 255
node 0 size: 98207 MB
node 0 free: 94798 MB
node 1 cpus:
node 1 size: 16384 MB
node 1 free: 15991 MB
```

- Binding the application to HBM (Flat/Hybrid)

```
user@knl% gcc myapp.c -o runme -mavx512f -O2
user@knl% numactl --membind 1 ./runme
// ... Application running in HBM ... //
```

ALLOCATION IN HBM WITH MEMKIND LIBRARY

Manual allocation to HBM possible with hbwmalloc and Memkind Library.

```
#include <hbwmalloc.h>
const int n = 1<<10;
// Allocation to MCDRAM
double* A = (double*) hbw_malloc(sizeof(double)*n);
// No replacement for _mm_malloc. Use posix_memalign
double* B;
int ret = hbw_posix_memalign((void*) B, 64, sizeof(double)*n);
.....
// Free with hbw_free
hbw_free(A); hbw_free(b);
```

Fortran Allocations.

```
REAL, ALLOCATABLE :: A(:)
!DEC$ ATTRIBUTES FASTMEM :: A
ALLOCATE (A(1:1024))
```

COMPILATION WITH MEMKIND LIBRARY AND HBWMALLOC

To compile C/C++ applications:

```
user@knl%  icpc -lmemkind foo.cc -o runme
user@knl%  g++ -lmemkind foo.cc -o runme
```

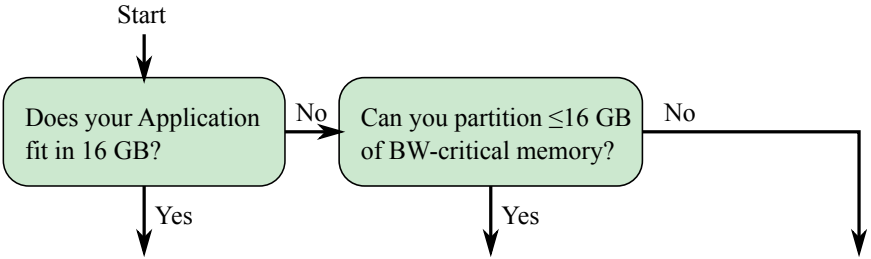
To compile Fortran applications:

```
user@knl%  ifort -lmemkind foo.f90 -o runme
user@knl%  gfortran -lmemkind foo.f90 -o runme
```

Open source distribution of Memkind library can be found at:
memkind.github.io/memkind

Learn more:
colfaxresearch.com/knl-mcdram

FLOW CHART FOR BANDWIDTH-BOUND APPLICATIONS

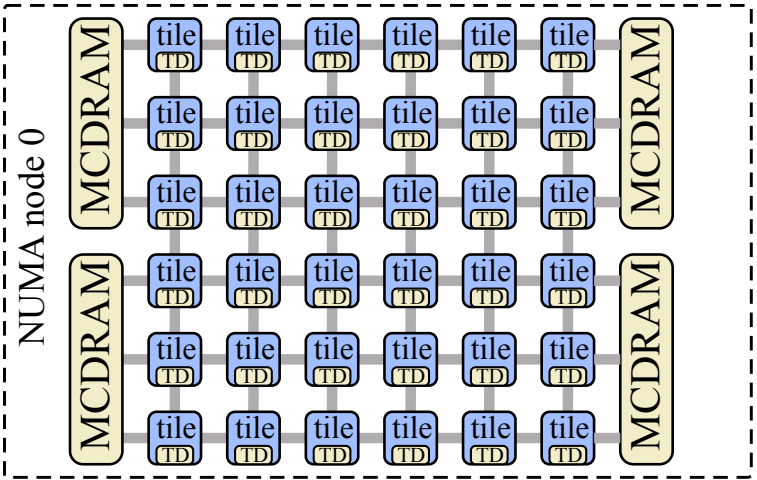


numactl	Memkind	Cache mode
▷ Simply run the whole program in MCDRAM ▷ No code modification required	▷ Manually allocate BW-critical memory to MCDRAM ▷ Memkind calls need to be added.	▷ Allow the OS to figure out how to use MCDRAM ▷ No code modification required

CLUSTERING MODES

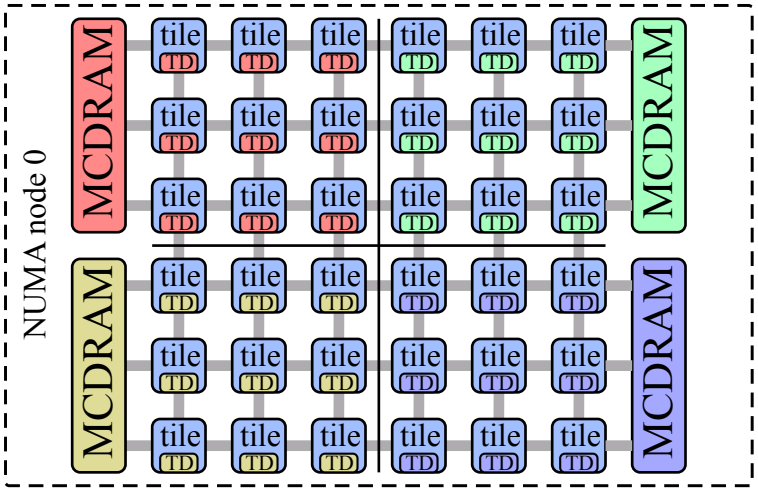
CLUSTERING MODES: ALL-TO-ALL

No affinity between the distributed Tag Directory (TD) and memory.



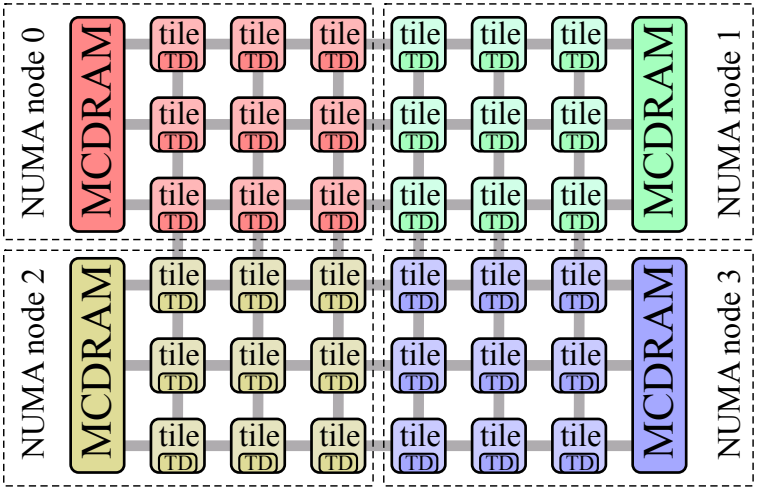
CLUSTERING MODES: QUADRANT/HEMISPHERE

Tag Directory (TD) and memory reside in the same quadrant.



CLUSTERING MODES: SNC-4/SNC-2

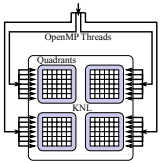
Cores appear as 4 (or 2) NUMA nodes.



HOW TO USE CLUSTERING MODES

Nested OpenMP

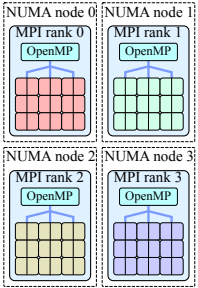
```
1  #pragma omp parallel
2  {
3    // ...
4  #pragma omp parallel
5    {
6      // ...
7    }
8  }
```




```
user@knl% OMP_NUM_THREADS=4,72
user@knl% OMP_NESTED=1
```

MPI+OpenMP

```
1  stat = MPI_Init();
2  // ...
3  #pragma omp parallel
4  {
5      // ...
6  }
7  // ...
8  MPI_Finalize();
```




```
user@knl% mpirun -host knl \
>  -np 4 ./myparallel_app
```

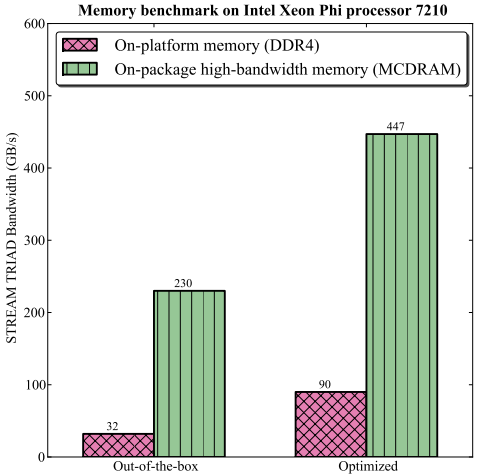
Learn more: colfaxresearch.com/knl-numa

§2. HANDS-ON DEMONSTRATIONS

MEMORY BANDWIDTH OPTIMIZATION

STREAM BENCHMARK

- Industry-standard tool for memory bandwidth measurement
- 4 tests: COPY, ADD, SCALE and TRIAD
- Download from Dr. John McCalpin's site: www.cs.virginia.edu/stream/



STREAM BENCHMARK TUNING

- KNL: Compile with `-xMIC-AVX512`
- Set large enough array size: `-DSTREAM_ARRAY_SIZE=64000000`
- Set 1 thread per core (-1 for offload)
- Xeon CPU: set affinity “scatter” (default on Xeon Phi)
- KNC: Tune prefetching (learn more)

In addition, secret sauce for your own STREAM-like application:

- Parallel first touch (see Session 8 of the HOW Series)
- Essential element – streaming stores: discussion

HBM IN KNIGHTS LANDING

- Finding HBM (MCDRAM) in an Intel Xeon Phi processor x200 (KNL):

```
user@knl% # In Flat mode with All-to-All or Quadrant
user@knl% numactl -H
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 ... 249 250 251 252 253 254 255
node 0 size: 98207 MB
node 0 free: 94798 MB
node 1 cpus:
node 1 size: 16384 MB
node 1 free: 15991 MB
```

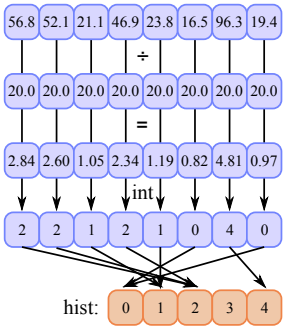
- Binding the application to HBM (Flat/Hybrid)

```
user@knl% numactl --membind 1 ./runme
// ... Application running in HBM ... //
```

VECTORIZATION WITH AVX-512

DEMO: AVX-512CD AND HISTOGRAMS

```
1  // worker.cc from Colfax lab 4.04
2  void Histogram(const float* age, int* const hist,
3                 const int n, const float group_width,
4                 const int m) {
5
6    for (int i = 0; i < n; i++) {
7      const int j = (int) ( age[i] / group_width );
8      hist[j]++;
9    }
10 }
```




```
user@knl% cat worker.optrpt
....
remark ...: vectorization support: scatter was generated for the variable hist:
remark ...: vectorization support: gather was generated for the variable hist:
remark #15300: LOOP WAS VECTORIZED
```

INTEL COMPILER SUPPORT FOR AVX-512

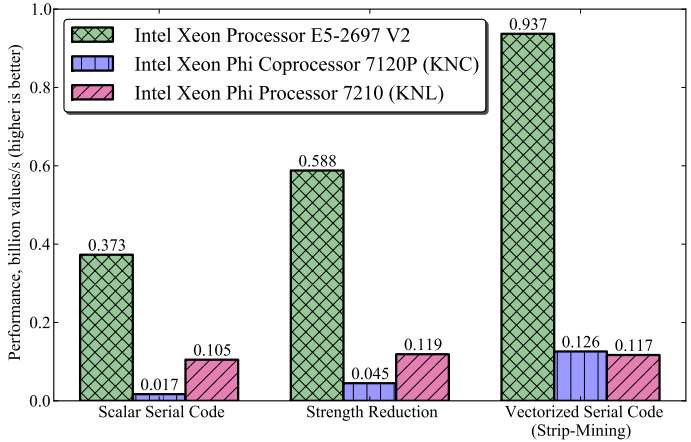
Intel Compiler versions ≥ 15.0 supports AVX-512 instruction set.

```
user@knl% icc -v
icc version 16.0.1 (gcc version 4.8.5 compatibility)
user@knl% icc -help
// ... truncated output ... //
-x<code>
     ...
     MIC-AVX512
     CORE-AVX512
     COMMON-AVX512
```

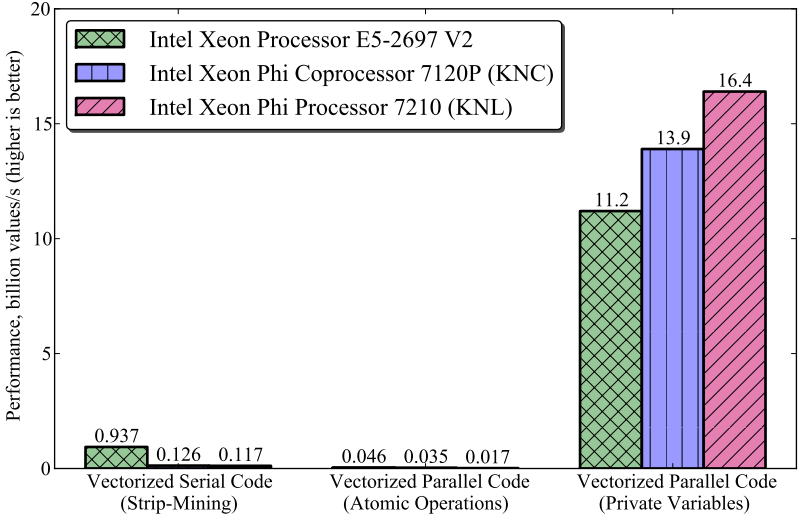
- -xMIC-AVX512 : for KNL (supports F, CD, ER, PF)
- -xCORE-AVX512 : for future Xeon (supports F, CD, DQ, BW, VL)
- -xCOMMON-AVX512 : common to KNL and Xeon (supports F, CD)

STRIP-MINING FOR VECTORIZATION

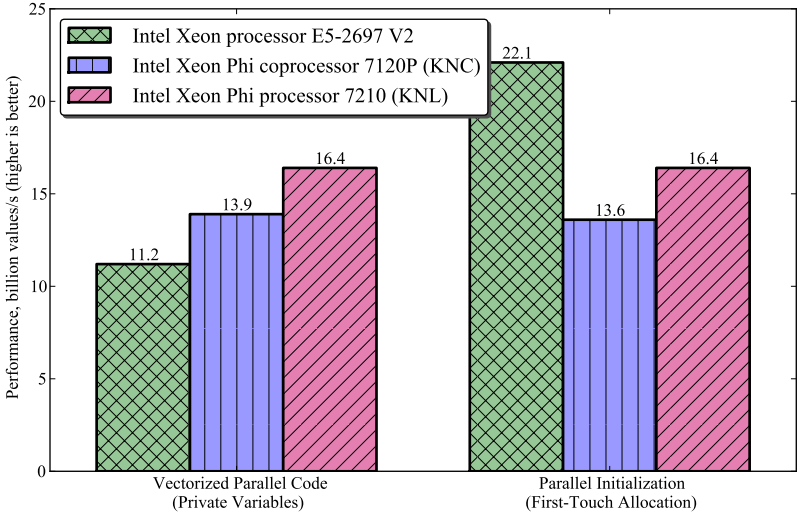
Vectorization improves performance on both platforms. However, more work is needed to take advantage of the MIC architecture. See materials on multi-threading.



USING REDUCTION INSTEAD OF SYNCHRONIZATION



MEMORY TRAFFIC OPTIMIZATION



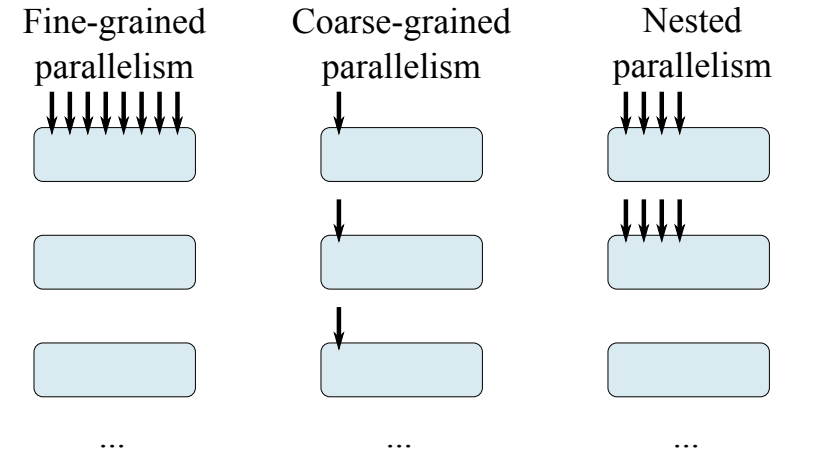
TUNING WITH INTEL MATH KERNEL LIBRARY

Intel® Math Kernel Library

Intel® Math Kernel Library (MKL) — standard mathematical functions optimized for Intel architecture.

Linear Algebra	Fast Fourier Transform	Vector Math	Vector Random Number Generators	Summary Statistics	Data Fitting
BLAS LAPACK Sparse solvers ScaLAPACK	Multidimentional (up to 7D) FFTW interfaces Cluster FFT	Trigonometric Hyperbolic Exponential Logarithmic Power/Root Rounding	Congruential Recursive Wichmann-Hill Mersenne Twister Sobol Neiderreiter Non-deterministic	Kurtosis Variation coefficitent Quantiles, order statistics Min/max Variance-covariance	Splines Interpolation Cell search

NESTED PARALLELISM



BATCH PROCESSING

- Feed multiple signals to MKL
- Let the library decide on the parallel strategy.

```
MKL_Complex16* data =
                (MKL_Complex16*) malloc(sizeof(MKL_Complex16)*fft_size*num_fft);

DFTI_DESCRIPTOR_HANDLE handle;
DftiCreateDescriptor(&handle, DFTI_DOUBLE, DFTI_COMPLEX, 1, (MKL_LONG)fft_size);
DftiSetValue(handle, DFTI_NUMBER_OF_TRANSFORMS, num_fft);
DftiSetValue(handle, DFTI_INPUT_DISTANCE, fft_size);
DftiSetValue(handle, DFTI_OUTPUT_DISTANCE, fft_size);
DftiSetValue(handle, DFTI_PLACEMENT, DFTI_INPLACE);
DftiCommitDescriptor(handle);
```

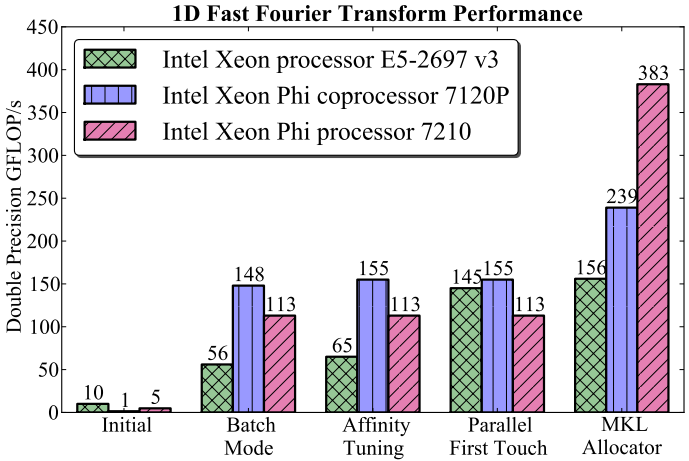
DATA CONTAINER TWEAKS

Special allocator (alignment – see also HOW series Session 6) + and allocation in HBM on Intel Xeon Phi processors

```
MKL_Complex16* data =
        (MKL_Complex16*) mkl_malloc(sizeof(MKL_Complex16)*fft_size*num_fft, 64);
```

COMPLEX-TO-COMPLEX 1D FFT PERFORMANCE

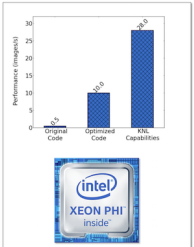
$2 \cdot 10^5$ FFTs of size 2048.



MACHINE LEARNING ON INTEL ARCHITECTURE

MACHINE LEARNING: OPTIMIZED MIDDLEWARE

INTEL® XEON PHI™ PROCESSORS — MACHINE LEARNING



NEURALTALK2 — OPEN SOURCE IMAGE TAGGING CODE (KARPATHY & FEI-FEI, STANFORD)

COLFAX
Customized Solutions

colfaxresearch.com/isc16-neuraltalk

WHERE TO LEARN MORE

GET READY FOR INTEL® XEON PHI PROCESSORS (CODENAMED: KNIGHTS LANDING)

colfaxresearch.com/knl-ready

HOW SERIES

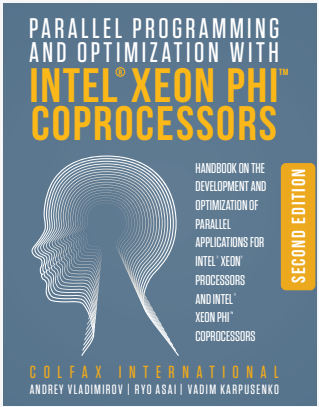
colfaxresearch.com/how-series

TEXTBOOK

ISBN: 978-0-9885234-0-1 (508 pages, Electronic or Print)

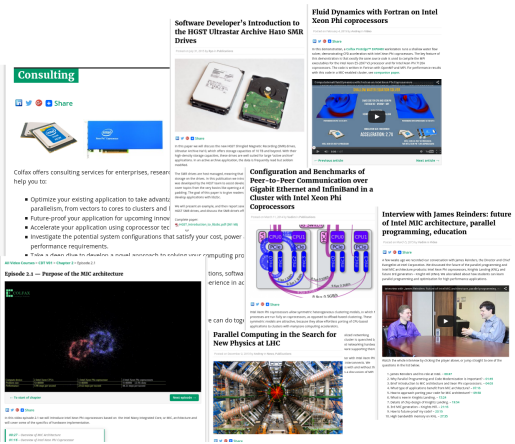
Parallel Programming and Optimization with Intel® Xeon Phi™ Coprocessors

Handbook on the Development and Optimization of Parallel Applications for Intel® Xeon® Processors and Intel® Xeon Phi™ Coprocessors



http://xeonphi.com/book

COLFAX RESEARCH



http://colfaxresearch.com/

DEVELOPER ACCESS PROGRAM (DAP)

Can't wait to get your hands on an Intel Xeon Phi processor?

Find out more at dap.xeonphi.com
or contact us at dap@colfax-intl.com.