



HOW Series: Knights Landing

Hands-On Workshop on Performance Optimization for Intel® Xeon Phi™ Processor Family x200

Colfax International — [@colfaxintl](#)

September 2016 - Rev. 3.0

About This Document

This document represents the materials of a Web-based training “Introduction to 2nd Generation Intel® Xeon Phi™ Processors: Developer’s Guide to Knights Landing” developed and run by Colfax International.

© Colfax International, 2013–2016

Parallel Programming Boot Camp (1-Day) / Workshop (4-Days)



Instructor-led 1-day or 4-days training, at your office or at Colfax facility in Sunnyvale, CA

[Click here to learn more](#)

1-Day Parallel Programming Boot Camp

For software engineers and architects, providing an overview of parallel programming frameworks and optimization guidelines for multi-core CPUs (Intel® Xeon®) and many-core coprocessors (Intel® Xeon Phi™):

- Discussions about three layers of parallelism: SIMD, Threads, Cluster environment
- Tips for quick porting/development of HPC software applications
- Real-life examples of code and optimization techniques
- Hardware solution and corresponding software implementations, APIs, and frameworks

4-Days Parallel Programming Workshop

For the developer who wants to hit the ground running with the modern multi-core CPUs (Intel® Xeon®), many-core coprocessors (Intel® Xeon Phi™) and leading software development tools:

- Hardware installation
- MPSS tools and the Linux environment on the Intel® Xeon Phi™ coprocessor
- Exploring differences in serial vs. parallel programming / processing / hardware usage
- Accelerated clusters
- Optimizations of vector arithmetics, memory traffic, thread parallelism and communication
- Using the Intel® Math Kernel Library

[Register Now!](#)

colfaxresearch.com/knl-webinar/

Disclaimer

While best efforts have been used in preparing this training, Colfax International makes no representations or warranties of any kind and assumes no liabilities of any kind with respect to the accuracy or completeness of the contents and specifically disclaims any implied warranties of merchantability or fitness of use for a particular purpose. The publisher shall not be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused, or alleged to have been caused, directly or indirectly, by the information or programs contained herein. No warranty may be created or extended by sales representatives or written sales materials.

Resources

Colfax Research

COLFAX RESEARCH
CONTRIBUTING TO INNOVATIONS IN COMPUTING

READ WATCH LEARN CONNECT JOIN

To search, type and hit enter

Popular

The Hands-On Tutorials (HOT) webinars: details an efficient programming for Intel architecture

The Hands-On Workshop (HOW) Series

Introduction to Intel DAAL, Part 1: Polynomial Regression with Batch Mode Computation

Parallel Programming Book

Introduction to parallel programming, deep discussion of optimization techniques, exercises.

Research and Educational Publications

Introduction to Intel DAAL, Part 1: Polynomial Regression with Batch Mode Computation

Optimization Techniques for the Intel MIC Architecture, Part 3 of 3: False Sharing and Padding

Software Developer's Introduction to the HGST Ultrastar Archive HaaS SMR Drives

Optimization Techniques for the Intel MIC Architecture, Part 2 of 3: Strip-Mining for Vectorization

Optimization Techniques for the Intel MIC Architecture, Part 1 of 3: Multi-Threading and Parallel Reduction

Performance to Power and Performance to Cost Ratios with Intel Xeon Phi Coprocessors (and why TX Acceleration May be Enough)

Log In/Out or Register

Consulting

Share

Colfax offers consulting services for enterprises, researchers, and developers. We help you to:

- Optimize your existing application to take advantage of parallelism, from vectors to cores to clusters and clouds
- Future-proof your application for upcoming innovations
- Accelerate your application using coprocessor technology
- Investigate the potential system configurations that satisfy your cost, power and performance requirements.
- Take a clean-slate to develop a novel approach to solve your computing problem

All Video Courses - COP 901 - Chapter 2 - Episode 1.1

Episode 2.1 - Purpose of the MIC architecture

Learn why the MIC architecture is important for high performance computing applications, software development, and system architecture.

Learn why the MIC architecture is important for high performance computing applications, software development, and system architecture.

Learn why the MIC architecture is important for high performance computing applications, software development, and system architecture.

Software Developer's Introduction to the HGST Ultrastar Archive HaaS SMR Drives

In this paper, we will discuss the HGST Ultrastar Archive HaaS SMR drives, which are the first 7.2 TB drives to offer 100 MB/s sequential read performance. These drives are ideal for use in high capacity applications, such as data backup and archival storage.

Configuration and Benchmarks of Peer-to-Peer Communication over Gigabit Ethernet and InfiniBand in a Cluster with Intel Xeon Phi Coprocessors

The Intel Xeon Phi coprocessors are designed to be used in a cluster environment. This article discusses the configuration and benchmarks of peer-to-peer communication over Gigabit Ethernet and InfiniBand in a cluster with Intel Xeon Phi coprocessors.

Fluid Dynamics with Fortran on Intel Xeon Phi coprocessors

In this article, we will discuss the use of Fortran on Intel Xeon Phi coprocessors for fluid dynamics simulations. We will show how to use the Intel Xeon Phi coprocessors to accelerate the simulation and how to use the Intel Xeon Phi coprocessors to reduce the simulation time.

Interview with James Reinders: future of Intel MIC architecture, parallel programming, education

In this interview, James Reinders discusses the future of Intel MIC architecture, parallel programming, and education. He talks about the challenges of parallel programming and the importance of education in preparing students for the future of computing.

Parallel Computing in the Search for New Physics at LHC

The Large Hadron Collider (LHC) is the world's largest particle accelerator. It produces vast amounts of data that need to be processed and analyzed. Parallel computing is used to speed up the processing and analysis of this data.

<http://colfaxresearch.com/>

HOW Series



THE "HOW" SERIES

DEEP DIVE

WITH CODE MODERNIZATION EXPERTS

*10x 2-hour sessions | 24-hour 2-weeks remote access to a system | Filling up fast, register now!

Interested? Sign-up at:

colfaxresearch.com/how-series

Developer Access Program (DAP)

Can't wait to get your hands on Knights Landing?



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

§2. Intel Architecture: Today and Tomorrow

Intel Architecture

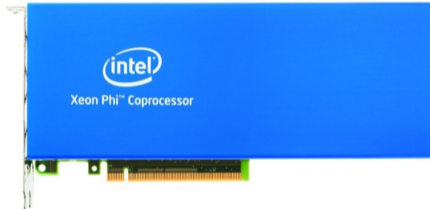
Intel Xeon Processor



Current: Broadwell
Upcoming: Skylake

Multi-Core Architecture

Intel Xeon Phi Coprocessor, 1st generation



Knights Corner (KNC)

Intel Xeon Phi Processor, 2nd generation*



* socket and coprocessor versions

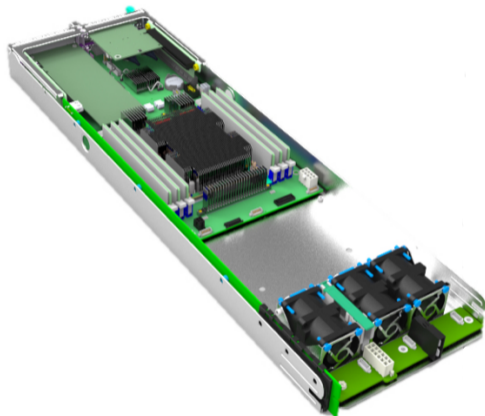
Knights Landing (KNL)

Intel Many Integrated Core (MIC) Architecture

Servers with Intel Xeon Phi Processors

- Bootable Host Processor
- RHEL/CentOS/SUSE/Win
- 64 cores \times 4 HT, 1.3 GHz
- \leq 384 GiB DDR4, $>$ 90 GB/s
- 16 GiB HBM, $>$ 400 GB/s
- PCIe bus for network
- 4 systems in 2U chassis

dap.xeonphi.com



Workstations with Intel Xeon Phi Processors

- Bootable Host Processor
- RHEL/CentOS/SUSE/Win
- 64 cores \times 4 HT, 1.3 GHz
- \leq 384 GiB DDR4, $>$ 90 GB/s
- 16 GiB HBM, $>$ 400 GB/s
- PCIe bus for add-in cards
- Liquid- or air-cooled

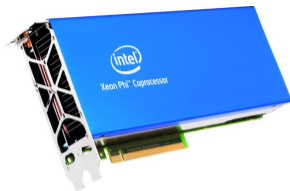
dap.xeonphi.com



Future Form-Factors

KNLF: KNL with Fabric

- Fabric integrated on CPU
 - ▶ Intel OmniPath Architecture
- Socket mount processor



*KNC image

KNL Coprocessor

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

§3. Out-of-the-Box Performance

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	Multidimensional (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 coefficient Quantiles, order statistics Min/max Variance-covariance	Splines Interpolation Cell search

DGEMM in MKL

C/C++

```
1 #include <mkl.h>  
2 ...  
3 cblas_dgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans,  
4             n, n, n, 1.0, A, n, B, n, 0.0, C, n);
```

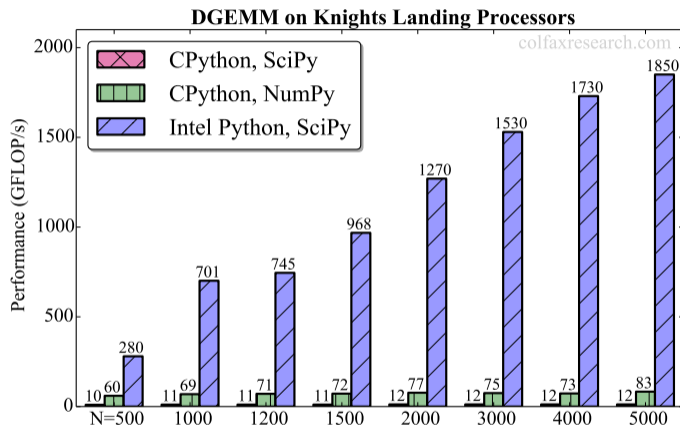
Python

```
from scipy.linalg.blas import dgemm as dgemm  
...  
C=dgemm(alpha=1.0, a=A, b=B, overwrite_c=1, trans_b=1)
```

R

```
require(Matrix)  
...  
c <- a %*% b
```

Python Benchmark

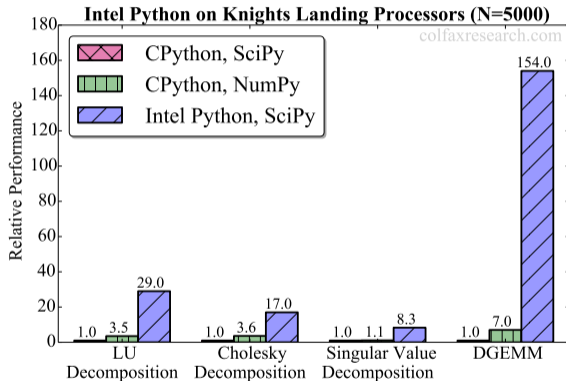


For more information on KNL with Intel Python, see [this paper](#)

§4. Importance of Code Optimization

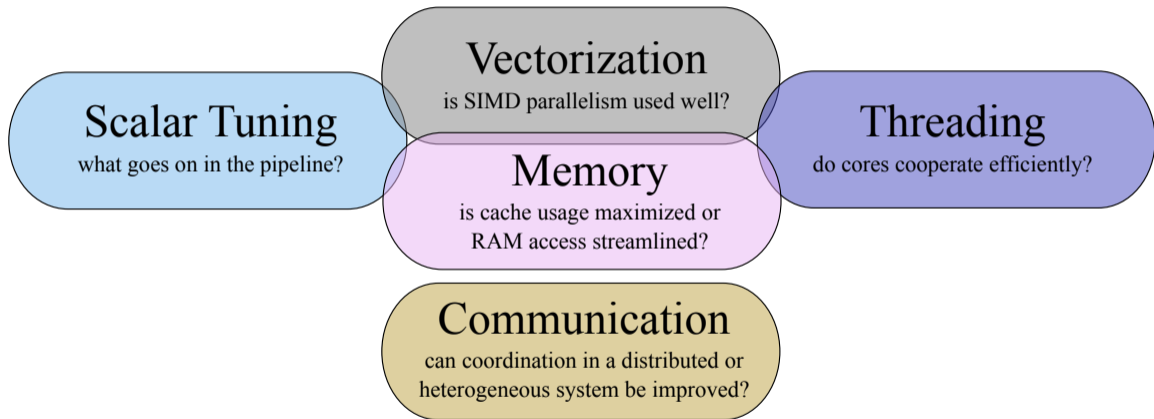
Python Benchmark

What we did not tell you in the previous slide...



It takes good software to unlock the performance of KNL!

Optimization Areas



Application

- 1 Astrophysics:
 - ▶ planetary systems
 - ▶ galaxies
 - ▶ cosmological structures
- 2 Electrostatic systems:
 - ▶ molecules
 - ▶ crystals

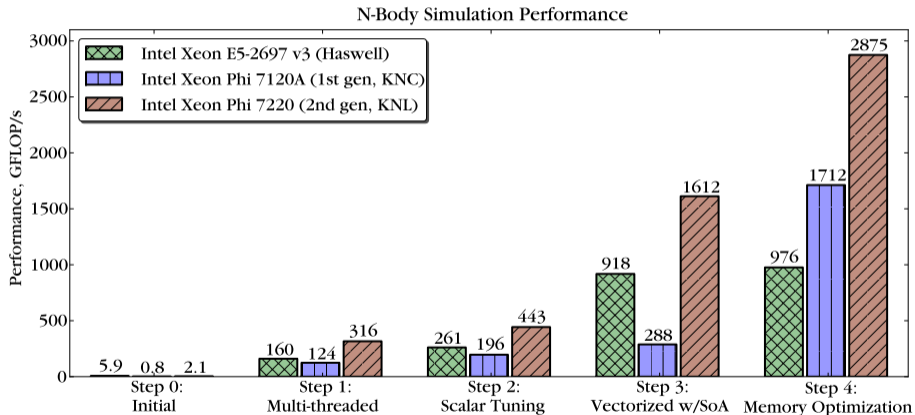
This work: “toy model” with all-to-all $O(n^2)$ algorithm. Practical N-body simulations may use tree algorithms with $O(n \log n)$ complexity.



Source: [APOD](#), credit: Debra Meloy Elmegreen (Vassar College) et al., & the Hubble Heritage Team (AURA/ STScI/ NASA)

N-body Simulation on Intel Architecture

Details in Chapter 23 of [this book](#)



The best way to prepare your code for KNL: optimize for Xeon or KNC

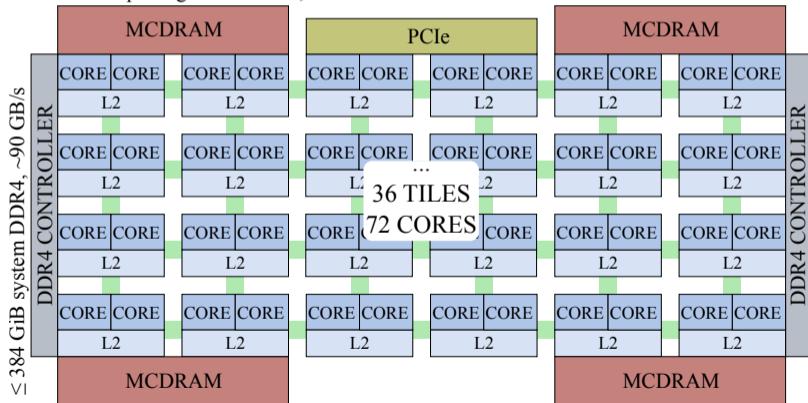
§5. Details of Architecture

Cores in Intel Xeon Phi Processors

KNL Die Organization: Tiles

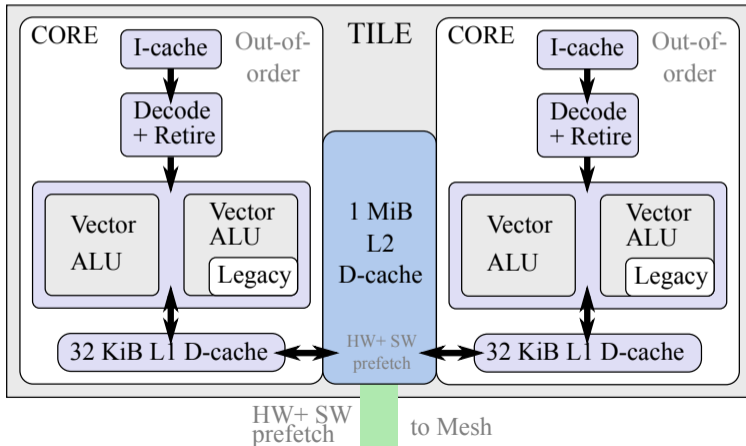
- Up to 36 tiles, each with 2 physical cores (72 total).
- Distributed L2 cache across a mesh interconnect.

≤ 16 GiB on-package MCDRAM, ~ 400 GB/s



KNL Cores

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



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

$$\begin{array}{r} 4 + 1 = 5 \\ 0 + 3 = 3 \\ -2 + 8 = 6 \\ 9 + -7 = 2 \end{array}$$

Vector Instructions

$$\begin{array}{r} 4 \\ 0 \\ -2 \\ 9 \end{array} + \begin{array}{r} 1 \\ 3 \\ 8 \\ -7 \end{array} = \begin{array}{r} 5 \\ 3 \\ 6 \\ 2 \end{array}$$

Vector Length

Dual VPU

Each core on KNL has two Vector Processing Units (VPUs).

Penalty from non-vectorized code

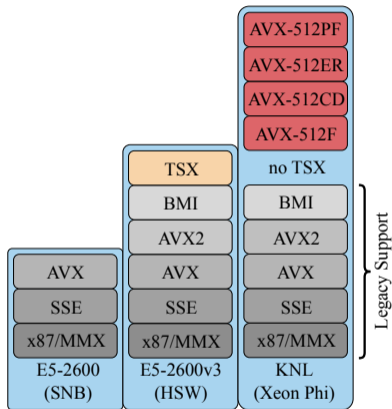
SP → 512 bit registers / 32 bits × 2 VPUs = **32** SIMD lanes

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



Supported Instruction Sets on KNL

- 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.
- \leq Intel® AVX2
 - ▶ Legacy mode operation.
 - ▶ Binary compatibility with Xeon.
 - ▶ Does *not* include IMCI (from KNC).



AVX-512 Features

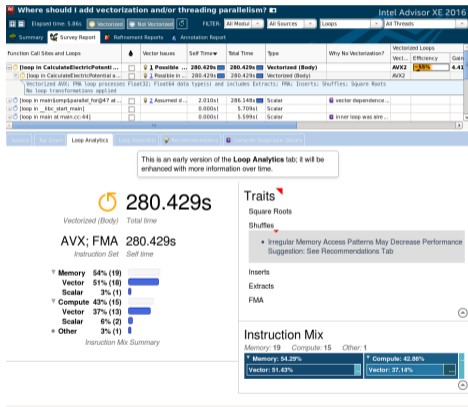
Knights Landing: first AVX-512 processor

- 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.

White Paper: <http://colfaxresearch.com/avx-512/>

Performance Considerations

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



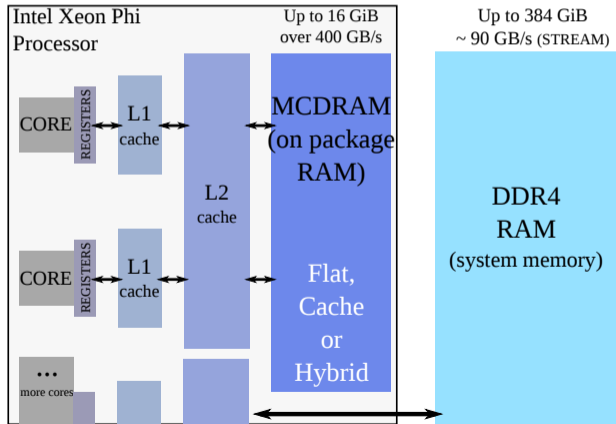
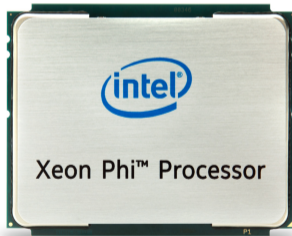
[link to paper](#)

- 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

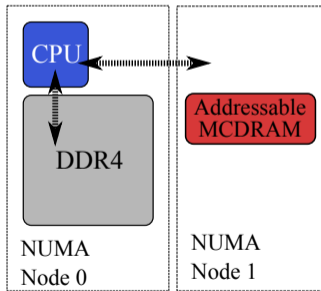
- Direct access to on-package MCDRAM *and* system DDR4 (socket)
- Use MCDRAM as cache, in flat mode, or as hybrid



MCDRAM Memory Modes

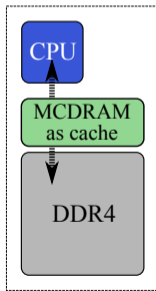
Flat Mode

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



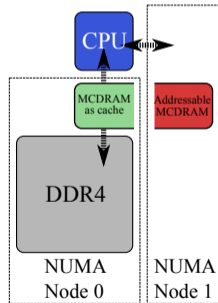
Cache Mode

- 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 with All-to-All
user@knl% numactl -H
available: 2 nodes (0-1)
node 0 cpus:  ... all cpus ...
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 MCDRAM (Flat/Hybrid)

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

Allocation in HBM with Memkind Library

Manual allocation to MCDRAM possible with hbwmalloc and Memkind Library.

```

1  #include <hbwmalloc.h>
2  const int n = 1<<10;
3  // Allocation to MCDRAM
4  double* A = (double*) hbw_malloc(sizeof(double)*n);
5  // No replacement for _mm_malloc. Use posix_memalign
6  double* B;
7  int ret = hbw_posix_memalign((void*) B, 64, sizeof(double)*n);
8  .....
9  // Free with hbw_free
10 hbw_free(A); hbw_free(b);

```

Fortran Allocations.

```

1  REAL, ALLOCATABLE :: A(:)
2  !DEC$ ATTRIBUTES FASTMEM :: A
3  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
```

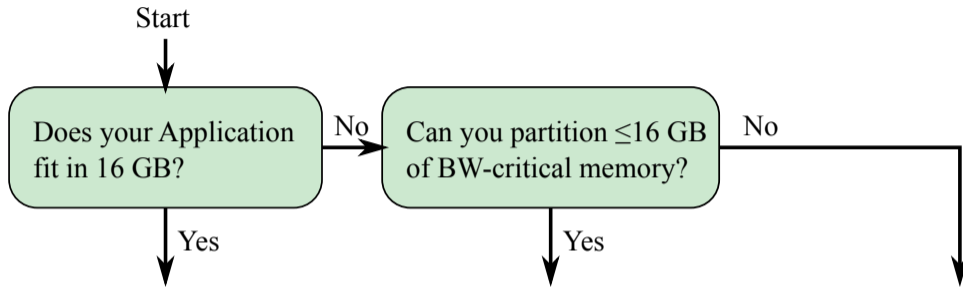
Open source distribution of Memkind library can be found at:

<http://memkind.github.io/memkind/>

Colfax Publication on using MCDRAM and Memkind Library:

<http://colfaxresearch.com/knl-mcdram/>

Flow Chart for Bandwidth-Bound Applications

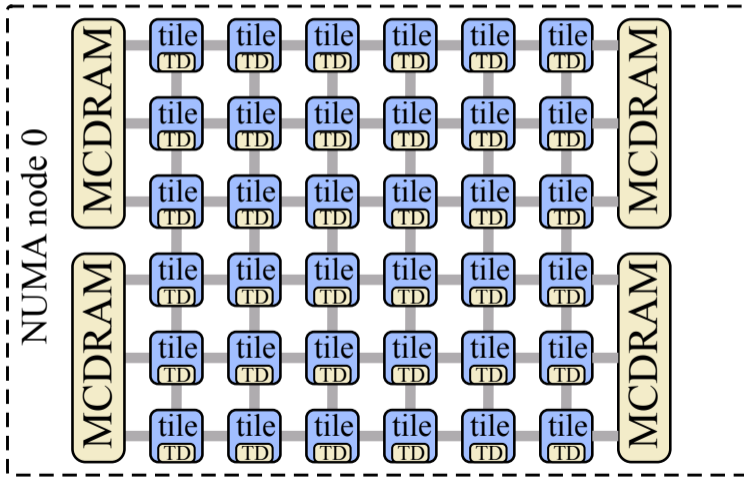


numactl	Memkind	Cache mode
<ul style="list-style-type: none"> • Simply run the whole program in MCDRAM • No code modification required 	<ul style="list-style-type: none"> • Manually allocate BW-critical memory to MCDRAM • Memkind calls need to be added. 	<ul style="list-style-type: none"> • Allow the OS to figure out how to use MCDRAM • No code modification required

Clustering Modes on KNL

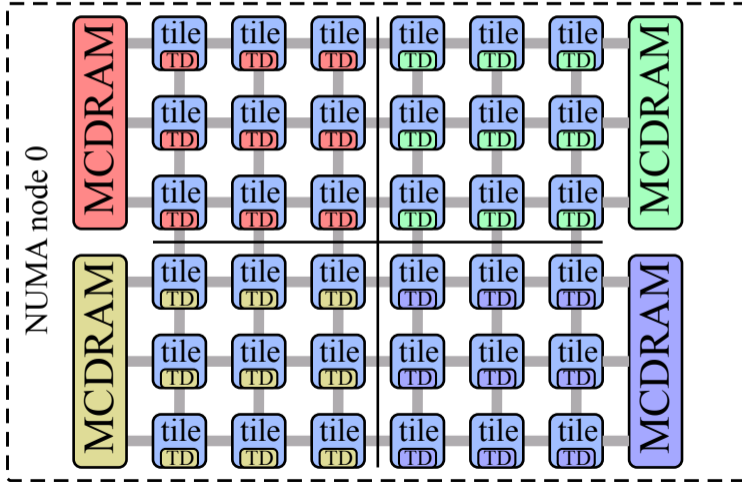
Clustering Modes: All-to-All

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



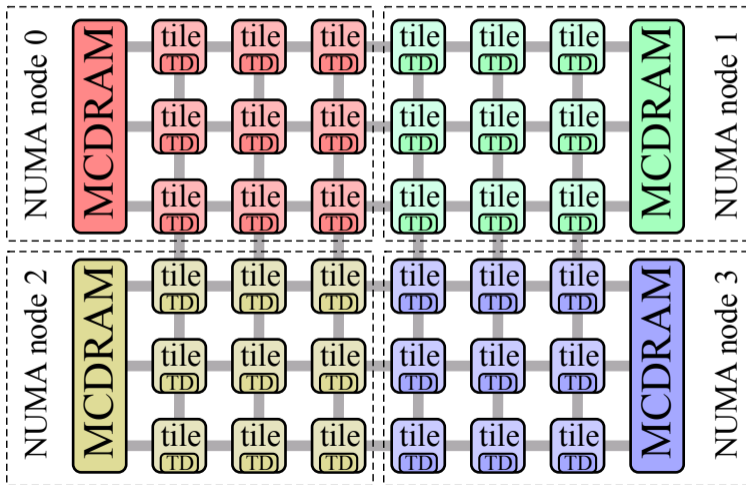
Clustering Modes: Quadrant/Hemisphere

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



Clustering Modes: SNC-4/SNC-2

Will appear as 4 NUMA nodes (similar to quad-socket system).



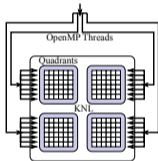
How to Use Clustering modes

Thread affinity to match the memory/directory affinity.

Nested OpenMP

```

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



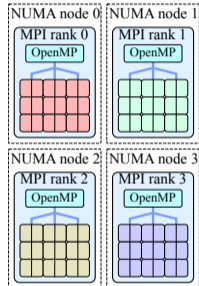
```

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
```

White Paper: <http://colfaxresearch.com/knl-numa/>

“Get Ready for Intel’s Knights Landing” Guide

Get Ready For Intel® Xeon Phi processors (Codenamed: Knights Landing)



colfaxresearch.com/knl-ready

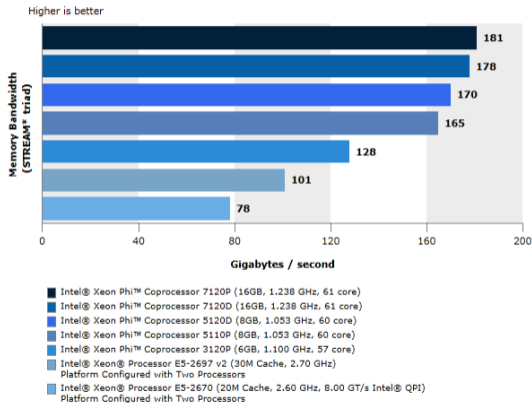
§6. Hands-On Demonstrations

Running the STREAM Benchmark

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

Expected for KNL: ≈ 400 GB/s



STREAM Benchmark on Tuning

- Set 1 thread per core on all platforms (-1 for offload)
- Set affinity “scatter”: [white paper](#) (Colfax Research)
- Tune prefetching: [white paper](#) (Intel)

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

- Parallel first touch (see NUMA discussion in Session 08)
- 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 ... //
```

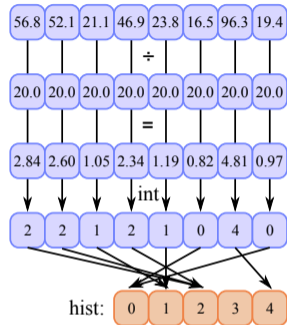
Vectorization with AVX-512

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

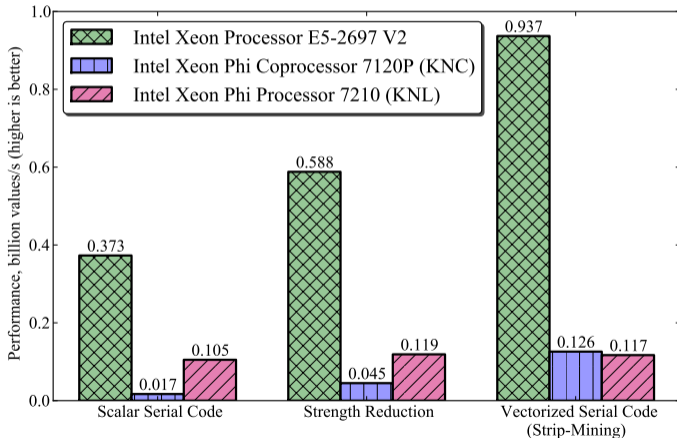
Intel Compiler versions ≥ 15.0 supports AVX-512 instruction set.

```
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
```

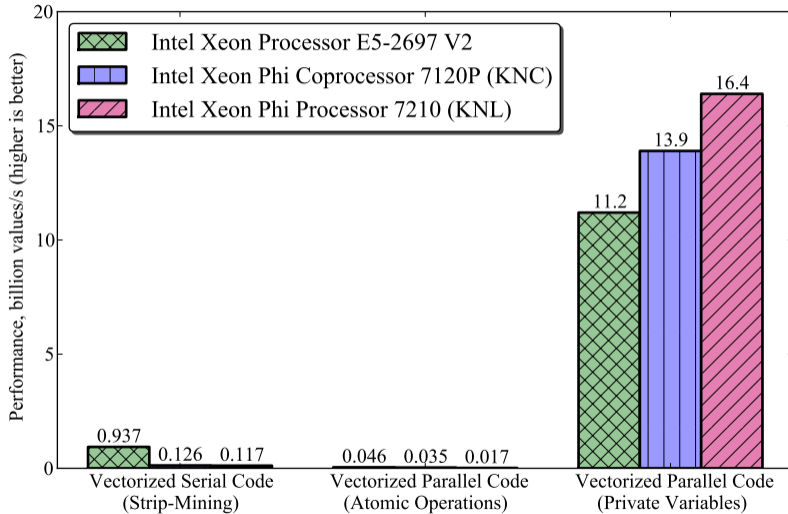
- `-xMIC-AVX512` : for KNL (supports F, CD, ER, PF)
- `-xCORE-AVX512` : for future Xeon (supports F, CD, DQ, BW, VL)
- `-xCOMMON-AVX512` : common to KNL and Xeon (supports F, CD)

Strip-Mining for Vectorization

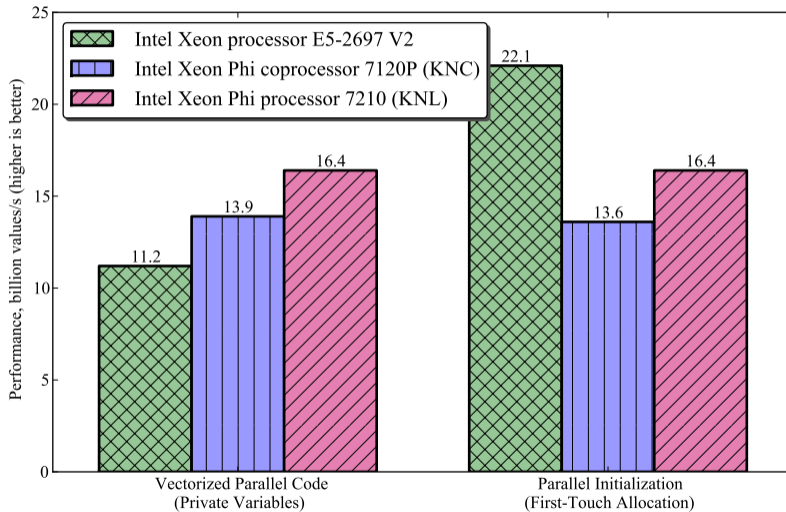
Vectorization improves performance on both platforms. However, more work is needed to take advantage of the MIC architecture. See materials on multi-threading.



Using Reduction instead of Synchronization



Memory Traffic Optimization



Tuning with Intel Math Kernel Library

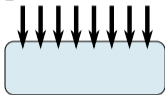
Batch Processing

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

```
1 MKL_Complex16* data =  
2     (MKL_Complex16*) malloc(sizeof(MKL_Complex16)*fft_size*num_fft);  
3  
4 DFTI_DESCRIPTOR_HANDLE handle;  
5 DftiCreateDescriptor(&handle, DFTI_DOUBLE, DFTI_COMPLEX, 1, (MKL_LONG)fft_size);  
6 DftiSetValue(handle, DFTI_NUMBER_OF_TRANSFORMS, num_fft);  
7 DftiSetValue(handle, DFTI_INPUT_DISTANCE, fft_size);  
8 DftiSetValue(handle, DFTI_OUTPUT_DISTANCE, fft_size);  
9 DftiSetValue(handle, DFTI_PLACEMENT, DFTI_INPLACE);  
10 DftiCommitDescriptor(handle);
```

Nested Parallelism

Fine-grained
parallelism



...

throwing all threads
on one MKL function

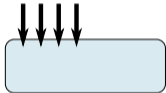
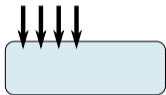
Coarse-grained
parallelism



...

using one thread
per MKL function

Nested
parallelism



...

putting teams of threads
on several MKL functions

Thread Affinity Tuning

Xeon

MKL uses 1 thread/core

OMP_NUM_THREADS=#phys. cores

KMP_AFFINITY=scatter

KMP_AFFINITY=compact (HT off)

KMP_AFFINITY=compact,1 (HT on)

Xeon Phi

MKL uses 4 threads/core

OMP_NUM_THREADS=4×#cores

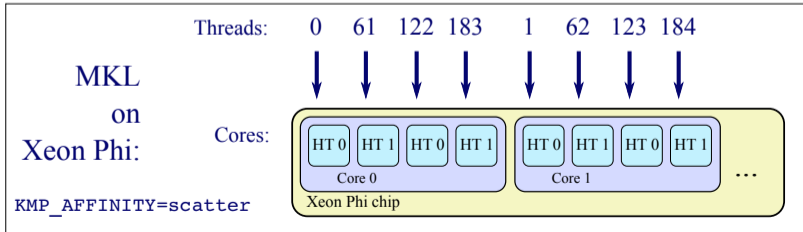
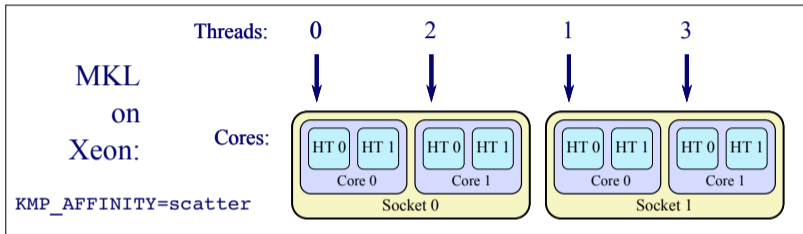
KMP_AFFINITY=scatter

KMP_AFFINITY=compact

See also HOW series [Session 8](#).

Thread Affinity: Scatter Pattern

Generally beneficial for bandwidth-bound applications.



Data Container Tweaks

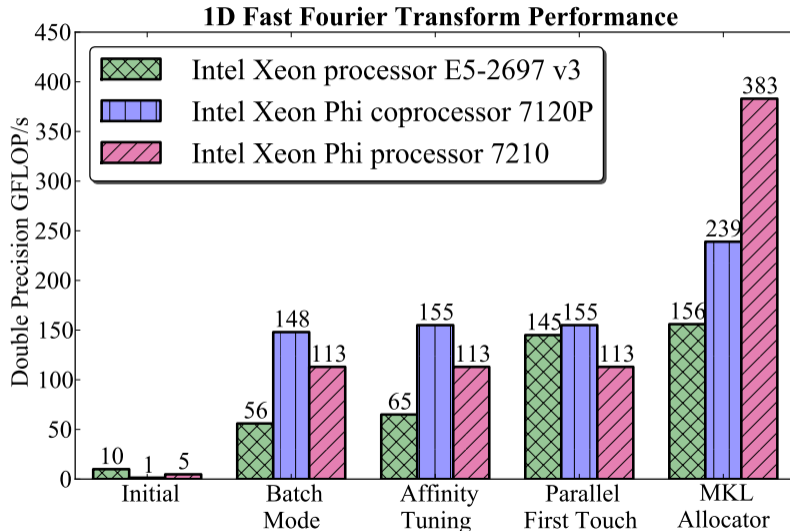
Parallel first-touch (see also HOW series [Session 8](#)):

```
1  #pragma omp parallel for
2  for(int i = 0; i < num_fft; i++)
3  for(int j = 0; j < fft_size; j++) {
4  data[i*fft_size+j].real = cos(k*j);
5  data[i*fft_size+j].imag = sin(k*j);
6  }
```

Special allocator (alignment – see also HOW series [Session 6](#) and allocation in HBM on KNL)

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

Complex-to-Complex 1D FFT Performance



Machine Learning on Intel Architecture

More Information: “HOW” Series

HOW Series

THE "HOW" SERIES

DEEP DIVE

WITH CODE MODERNIZATION EXPERTS

*10x 2-hour sessions | 24-hour 2-weeks remote access to a system | Filling up fast, register now!

Need more tutorials? Sign-up at:

colfaxresearch.com/how-series

§7. Closing Words

Bottom Line

KNL is a **highly-parallel** successor of KNC, with improvements in both **performance** and **ease-of-use**.



KEEP CALM
AND
GO PARALLEL

Get Ready For Intel® Xeon Phi processors (Codename: Knights Landing)



colfaxresearch.com/knl-ready

Colfax Research

COLFAX RESEARCH

CONTRIBUTING TO INNOVATIONS IN COMPUTING

[Log In/Out](#) or [Register](#)

READ WATCH LEARN CONNECT JOIN

To search, type and hit enter

Popular

The Hands-On Tutorials (HOT) webinars: details an efficient programming for Intel architecture

The Hands-On Workshop (HOW) Series

Introduction to Intel DAAL, Part 1: Polynomial Regression with Batch Mode Computation

Parallel Programming Book

Introduction to parallel programming, deep discussion of optimization techniques, exercises.

© 2015, Colfax International. 508 pages.

Research and Educational Publications



Introduction to Intel DAAL, Part 1: Polynomial Regression with Batch Mode Computation



Software Developer's Introduction to the HGST Ultrastar Archive HaaS SMR Drives



Optimization Techniques for the Intel MIC Architecture, Part 1 of 3: Multi-Threading and Parallel Reduction



Optimization Techniques for the Intel MIC Architecture, Part 3 of 3: Strip-Mining for Vectorization



Optimization Techniques for the Intel MIC Architecture, Part 2 of 3: Strip-Mining for Vectorization



Performance to Power and Performance to Cost Ratios with Intel Xeon Phi Coprocessors (and why TX Acceleration May be Enough)

Featured Video

[Watch Research tutorial on vectorization for a floating point](#)



<http://colfaxresearch.com/?p=704>

Presentations

[Watch Intel MIC Architecture](#)

[Watch Introduction to Intel Xeon Phi Coprocessors](#)

Consulting

Share

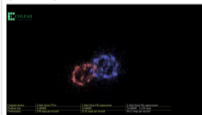


Colfax offers consulting services for enterprises, research and education. We can help you to:

- Optimize your existing application to take advantage of parallelism, from vectors to cores to clusters and clouds
- Future-proof your application for upcoming innovations
- Accelerate your application using coprocessor technologies
- Investigate the potential system configurations that satisfy your cost, power and performance requirements.
- Take a clean sheet to develop a novel approach to solve your computing problem

All Video Courses - COP 901 - Chapter 2 - Episode 1.1

Episode 2.1 - Purpose of the MIC architecture



Watch this video to learn the purpose of the MIC architecture. In this video episode 2.1 we will introduce you to the coprocessor based on the Intel Hyper-Threaded Core, an MIC architecture and explore some of the specific hardware considerations.

0:00 Introduction of MIC Architecture
01:45 Introduction of Intel Xeon Phi Coprocessors

Software Developer's Introduction to the HGST Ultrastar Archive HaaS SMR Drives



In this paper we will discuss the new HGST Ultrastar Archive HaaS SMR drives. Software developers who use applications that require large amounts of data for high storage capacity, these drives are well suited for "big data" and other applications. In a public sector application, the data is frequently read but seldom modified.

The SMR drives are best managed, meaning they are designed to be managed in the same way as the traditional hard drives. This is the only way to ensure that the data is always available and that the system is always running. The goal of this paper is to give you a better understanding of the SMR drives and how they can be used in your applications.

We will present an example, and then explain how to use the SMR drives. We will also present an example, and then explain how to use the SMR drives. We will also present an example, and then explain how to use the SMR drives.

Fluid Dynamics with Fortran on Intel Xeon Phi coprocessors

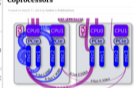
In this presentation, a **Fortran** program is used to solve the Navier-Stokes equations for fluid dynamics. The program is written in Fortran and is executed on the Intel Xeon Phi coprocessor. The code is written in Fortran and is executed on the Intel Xeon Phi coprocessor. The code is written in Fortran and is executed on the Intel Xeon Phi coprocessor.



Download the code and run it on your Intel Xeon Phi coprocessor. The code is written in Fortran and is executed on the Intel Xeon Phi coprocessor. The code is written in Fortran and is executed on the Intel Xeon Phi coprocessor.

Download the code and run it on your Intel Xeon Phi coprocessor. The code is written in Fortran and is executed on the Intel Xeon Phi coprocessor. The code is written in Fortran and is executed on the Intel Xeon Phi coprocessor.

Configuration and Benchmarks of Peer-to-Peer Communication over Gigabit Ethernet and InfiniBand in a Cluster with Intel Xeon Phi Coprocessors



In this paper we will discuss the configuration and benchmarks of peer-to-peer communication over Gigabit Ethernet and InfiniBand in a cluster with Intel Xeon Phi coprocessors. The code is written in Fortran and is executed on the Intel Xeon Phi coprocessor.

In this paper we will discuss the configuration and benchmarks of peer-to-peer communication over Gigabit Ethernet and InfiniBand in a cluster with Intel Xeon Phi coprocessors. The code is written in Fortran and is executed on the Intel Xeon Phi coprocessor.



Interview with James Reinders: future of Intel MIC architecture, parallel programming, education

In this interview, James Reinders discusses the future of Intel MIC architecture, parallel programming, and education. The code is written in Fortran and is executed on the Intel Xeon Phi coprocessor.



In this interview, James Reinders discusses the future of Intel MIC architecture, parallel programming, and education. The code is written in Fortran and is executed on the Intel Xeon Phi coprocessor.

In this interview, James Reinders discusses the future of Intel MIC architecture, parallel programming, and education. The code is written in Fortran and is executed on the Intel Xeon Phi coprocessor.

<http://colfaxresearch.com/>

Developer Access Program (DAP)

Can't wait to get your hands on Knights Landing?



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

Thank you for Tuning In!



§8. Back-up Slides

Loop Collapse: Principle

Idea: combine iterations spaces of the inner loop and the outer loop.

```
1 #pragma omp parallel for collapse(2)
2   for (int i = 0; i < m; i++)
3     for (int j = 0; j < n; j++) {
4         // ...
5         // ...
6     }
```

```
1 #pragma omp parallel for
2   for (int c = 0; c < m*n; c++) {
3       i = c / n;
4       j = c % n;
5       // ...
6   }
```

Exposing Parallelism: Strip-Mining and Loop Collapse

```
1 void sum_stripmine(const int m, const int n, long* M, long* s){
2     const int STRIP=1024;
3     assert(n%STRIP==0);
4     s[0:m]=0;
5     #pragma omp parallel
6     {
7         long total[m]; total[0:m]=0;
8         #pragma omp for collapse(2) schedule(guided)
9         for (int i=0; i<m; i++)
10            for (int jj=0; jj<n; jj+=STRIP)
11                #pragma vector aligned
12                    for (int j=jj; j<jj+STRIP; j++)
13                        total[i]+=M[i*n+j];
14        for (int i=0; i<m; i++) // Reduction
15            #pragma omp atomic
16                s[i]+=total[i];
17    } }
```

Checking for AVX-512 support

Check /proc/cpuinfo for AVX-512 flags

```
user@knl% cat /proc/cpuinfo
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm
constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc
aperfmpperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 fma
cx16 xtpr pdcm sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer xsave avx
f16c rdrand lahf_lm abm 3dnowprefetch arat epb xsaveopt pln pts dtherm
tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 avx2 smep bmi2
erms avx512f rdseed adx avx512pf avx512er avx512cd
```

Also possible through C/C++ function calls: see [this blog](#).

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 \geq 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 `-O2` or `-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)
```

HOW Series



THE "HOW" SERIES

DEEP DIVE

WITH CODE MODERNIZATION EXPERTS

STARTS MAY 23

*10x 2-hour sessions | 24-hour 2-weeks remote access to a system | Filling up fast, register now!

Interested? Sign-up at:

colfaxresearch.com/how-series

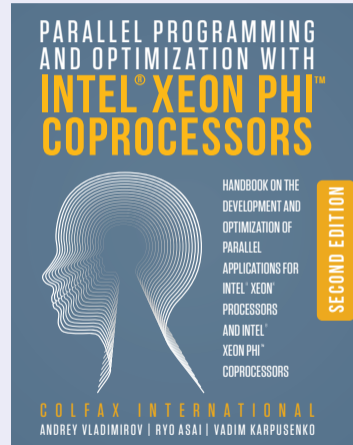
Textbook

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

Parallel Programming and Optimization with Intel® Xeon Phi™ Coproprocessors

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

© Colfax International, 2015



<http://xeonphi.com/book>