



Practical Usage of the Intel Math Kernel Library (MKL)

The Hands-On Tutorial (HOT) Series


Andrey Vladimirov, PhD — Colfax International
colfaxresearch.com

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.

About the Series

Hands-On Tutorial (HOT) series: webinars on efficient programming for the Intel architecture. Select topics of parallel programming and performance optimization with detailed practical demonstrations.

A blue rectangular graphic with white and light blue text. The background features faint, light blue circuit-like patterns. The text is centered and reads: 'THE "HOT" (HANDS ON TUTORIAL) SERIES' in light blue, 'FREE ONLINE WEBINAR' in large white letters, 'EFFICIENT PROGRAMMING FOR INTEL® ARCHITECTURE' in white, and 'DEC 08, 15 & 22' in light blue. At the bottom, it says '3 webinar series | Filling up fast, register now!' in white.

THE "HOT" (HANDS ON TUTORIAL) SERIES
FREE ONLINE WEBINAR
EFFICIENT PROGRAMMING FOR INTEL® ARCHITECTURE
DEC 08, 15 & 22
3 webinar series | Filling up fast, register now!

colfaxresearch.com/hot-1512

Slides, Code, Video

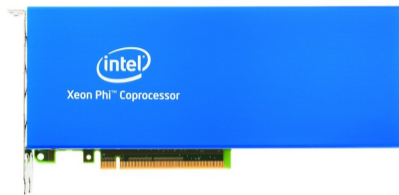
You can download slides, code and watch the video recording of this webinar here (requires registration for a free Colfax Research account):

colfaxresearch.com/hot-1512

Past webinars (December 8 & 15, 2015: “From Scalar to Serial...” and “Finding the Low-Hanging Fruit for Performance Optimization”) — recordings on same page.

§2. Intel Architecture

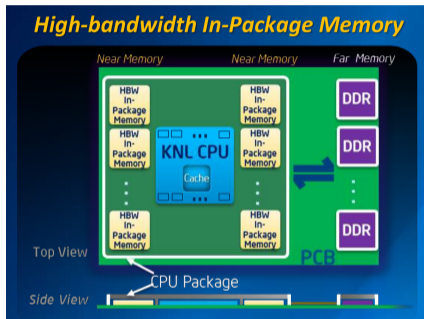
Intel Xeon Phi Coprocessors and the MIC Architecture



- C/C++/Fortran; OpenMP/MPI
 - Standard Linux OS
 - Up to 768 GB of DDR3 RAM
 - ≤ 18 cores/socket ≈ 3 GHz
 - 2-way hyper-threading
 - 256-bit AVX vectors
- C/C++/Fortran; OpenMP/MPI
 - Special Linux distribution
 - 6–16 GB cached GDDR5 RAM
 - 57 to 61 cores at ≈ 1 GHz
 - 4 hardware threads per core
 - 512-bit IMCI vectors

Next Generation of the MIC Architecture

- 2nd generation MIC product: code name Knights Landing (KNL)
- Intel's 14 nm manufacturing process
- A processor (running the OS) or a coprocessor (PCIe device)
- On-package high-bandwidth memory w/ flexible memory models: flat, cache, & hybrid
- Intel Advanced Vector Extensions AVX-512 (public)



Source: [Intel Newsroom](#)

Intel® Xeon Phi™ Product Family Roadmap

The Faster Path to Discovery



Available Today
Knights Corner
Intel® Xeon Phi™
x100 Product Family
22 nm process
Coprorocessor
Over 1 TF DP Peak
Up to 61 Cores
Up to 16GB GDDR5



2H'15*
Knights Landing
Intel® Xeon Phi™
x200 Product Family
14 nm process
Server Processor &
Coprorocessor
Over 3 TF DP Peak¹
60+ cores
*And new
details
today...*

Future

TBA

3rd generation

In planning

* First commercial systems

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

¹ Over 3 Teraflops of peak theoretical double precision performance is preliminary and based on current expectations of cores, clock frequency and floating point operations per cycle. FLOPS = cores x clock frequency x floating point operations per second per cycle.

Source: <https://www.brighttalk.com/webcast/10773/116329>

§3. Intel Math Kernel Library

Performance Tuning Methodology

- **Scalar optimization** (compiler-friendly practices)
- **Vectorization** (must use 16- or 8-wide vectors)
- **Multi-threading** (must scale to 100+ threads)
- **Memory access** (streaming access or tiling)
- **Communication** (offload, MPI traffic control)

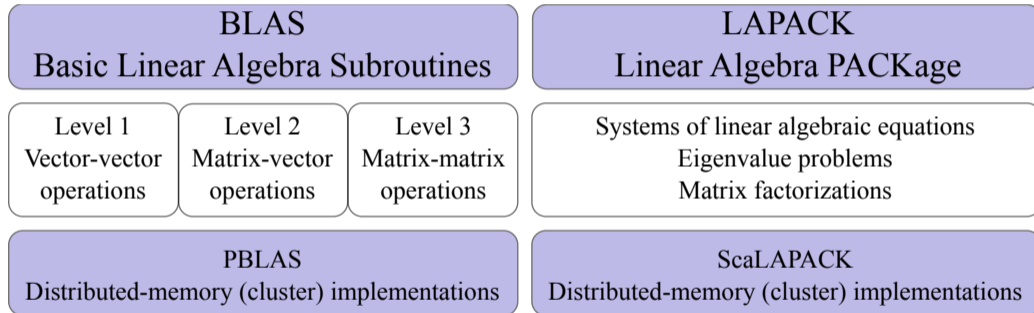
Details in the [HOW Series](#) and [our book](#).

Scope of MKL

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 coefficent Quantiles, order statistics Min/max Variance-covariance	Splines Interpolation Cell search

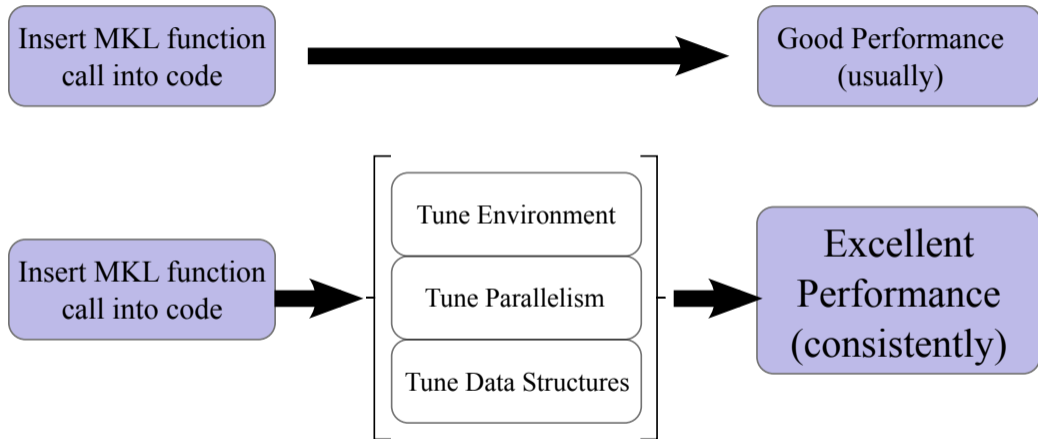
Linear Algebra Support



Using MKL with Intel Xeon Phi Coprocessors

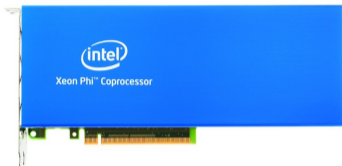
	Native	Compiler-Assisted Offload (CAO)	Automatic Offload (AO)
Programming:	Call MKL functions like from CPU code	Call MKL functions from #pragma offload	Call MKL functions from CPU code
Compilation:	With argument -mmic	Conventional for CPU	Conventional for CPUs
Running	On Xeon Phi	On host CPU	On host CPU; MKL_MIC_ENABLE=1
Data movement	Data already on coprocessor	Managed by programmer	Managed by MKL
Functions	All	All	Some; problem size > threshold
Multiple Copr.	Use cluster functions	DIY	Supported out-of-box

Performance Tuning with MKL



§4. Hands-On Labs

“Standard Candle” Testbench



One Intel Xeon Phi 7120P
coprocessor (2012)
TDP: 300 W, RCP: \$4129

vs.



Two Intel Xeon E5-2697 v3
CPUs (2014)
TDP: 290 W, RCP: \$5404

See also [“Intel Xeon Product Family: Performance Brief”](#)

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);
```

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

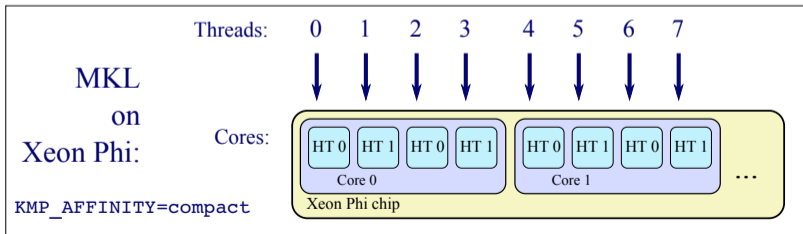
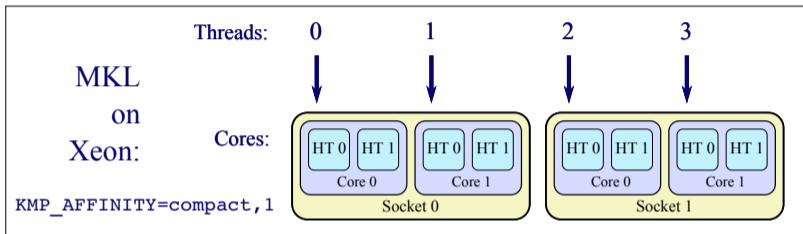
KMP_AFFINITY=scatter

KMP_AFFINITY=compact

See also HOW series [Episode 3.4](#).

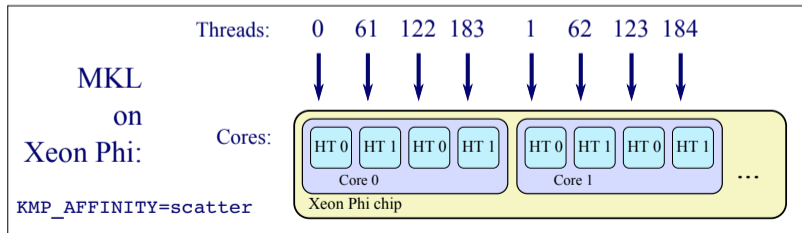
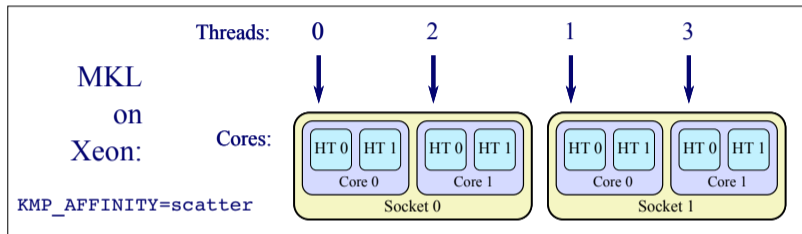
Thread Affinity: Compact Pattern

Generally beneficial for compute-bound applications.



Thread Affinity: Scatter Pattern

Generally beneficial for bandwidth-bound applications.



Data Container Tweaks

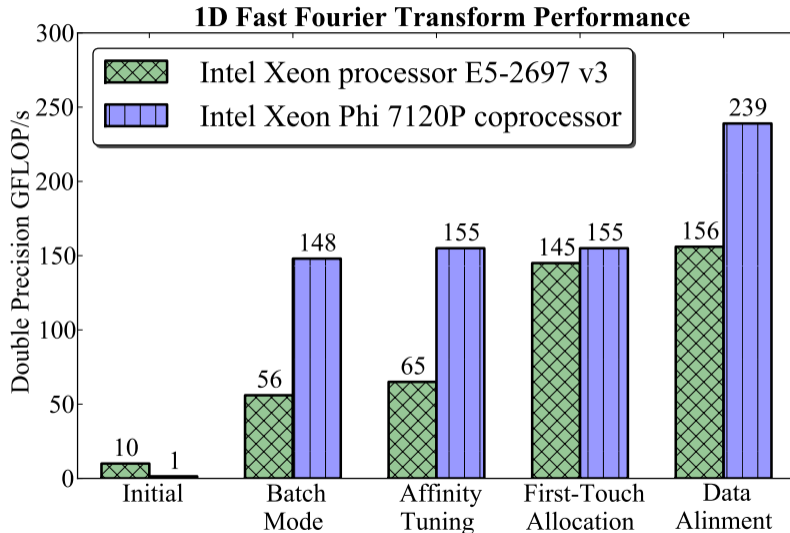
Parallel first-touch (see also HOW series [Episode 3.5](#)):

```
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     }
```

Data alignment (see also HOW series [Episode 3.2](#))

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

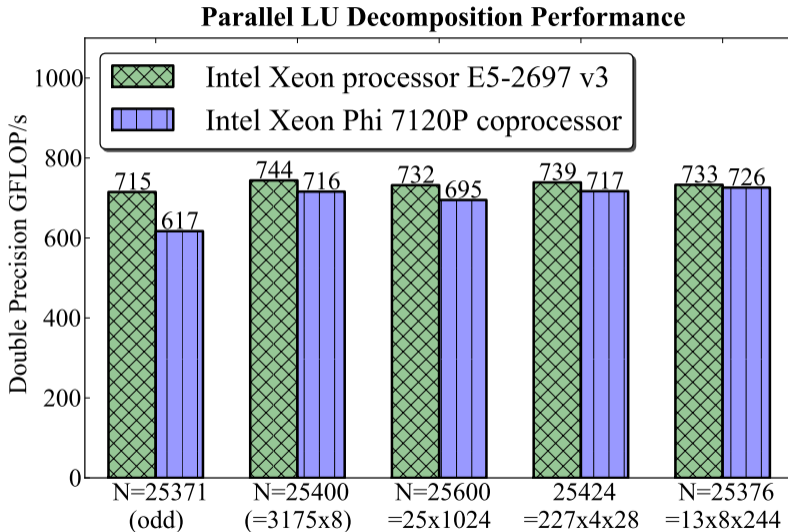
Complex-to-Complex 1D FFT Performance



Problem Size Tuning

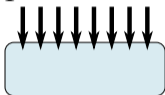
- Some functions require “good” problem sizes
- Good practice #1: array length a multiple of vector length
- Good practice #2: array length a multiple of number of threads
- Good practice #3: array length a multiple of a power of 2
- BLAS and LAPACK have “leading array dimension” for padding rows

LU Decomposition Size Tuning



Nested Parallelism

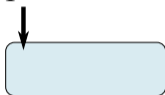
Fine-grained
parallelism



...

throwing all threads
on one MKL function

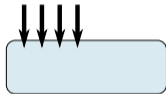
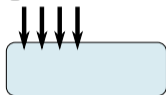
Coarse-grained
parallelism



...

using one thread
per MKL function

Nested
parallelism



...

putting teams of threads
on several MKL functions

OpenMP Hot Teams

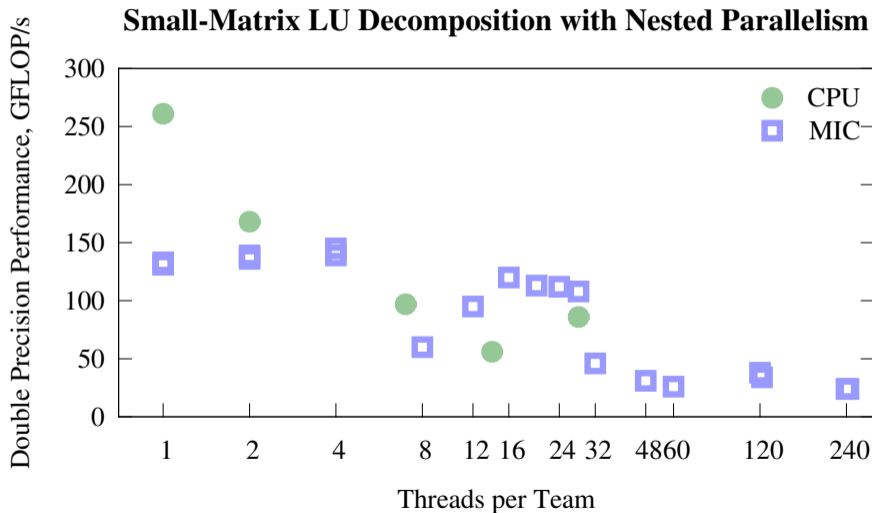
Xeon

Xeon Phi

- `OMP_NUM_THREADS=2,14`
- `OMP_NESTED=1`
`OMP_PROC_BIND=spread,close`
`OMP_PLACES=cores`
- `KMP_HOT_TEAMS_MODE=1`
`KMP_HOT_TEAMS_MAX_LEVEL=2`
`OMP_MAX_ACTIVE_LEVELS=2`
- `MKL_DYNAMIC=false`

- `OMP_NUM_THREADS=60,4`
- `OMP_NESTED=1`
`OMP_PROC_BIND=spread,close`
`OMP_PLACES=threads`
- `KMP_HOT_TEAMS_MODE=1`
`KMP_HOT_TEAMS_MAX_LEVEL=2`
`OMP_MAX_ACTIVE_LEVELS=2`
- `MKL_DYNAMIC=false`

Small Matrix LU Decomposition with Nested Parallelism



Automatic Offload

```
MKL_MIC_ENABLE=1

OMP_NUM_THREADS=... # CPU thread setting
KMP_AFFINITY=... # CPU affinity setting

MIC_KMP_PLACE_THREADS=4t # MIC thread setting
MIC_KMP_AFFINITY=... # MIC affinity setting
```

Compilation of R with MKL

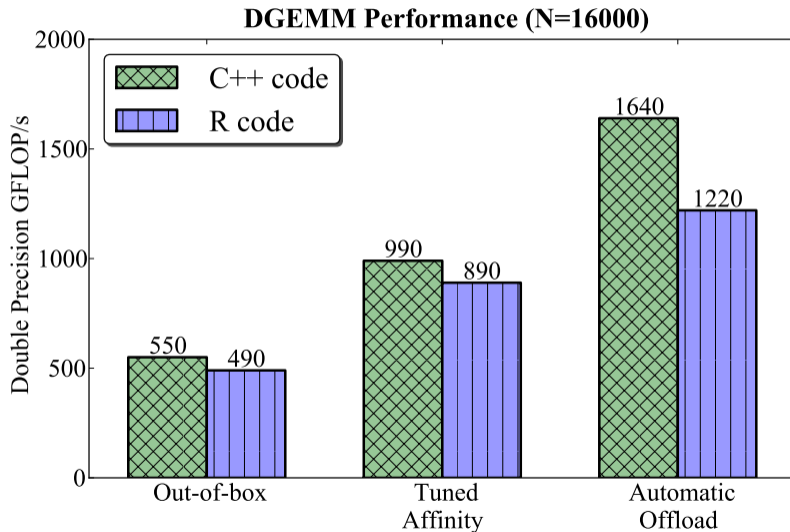
```
./configure \  
  --with-blas="-L/opt/intel/mkl/lib/intel64 \  
              -lmkl_intel_lp64 -lmkl_core -lmkl_intel_thread -lpthread -lm" \  
  --with-lapack \  
  CC=icc CFLAGS="-O2 -qopenmp -I/opt/intel/mkl/include" \  
  CXX=icpc CXXFLAGS="-O2 -qopenmp -I/opt/intel/mkl/include" \  
  F77=ifort FFLAGS="-O2 -qopenmp -I/opt/intel/mkl/include" \  
  FC=ifort FCFLAGS="-O2 -qopenmp -I/opt/intel/mkl/include" \  
  --prefix=/opt/R
```

```
user@host% make
```

```
user@host% su
```

```
root@host% make install
```

Performance with C++ Code and in R



§5. Additional Information

Acquiring MKL

	Community License	Commercial License
Cost	Free	Per developer
Support	Intel Premier Support	Community forum
Use in products	Yes	Yes
Royalty-free	Yes	Yes

Compilation with MKL

Intel® Math Kernel Library Link Line Advisor | Intel® Developer Zone - Mozilla Firefox

Intel® Math Kernel Library Link Line Advisor

July 20, 2012

Share Tweet +Share

Forums
Intel® Math Kernel Library

Introduction

The Intel® Math Kernel Library (Intel® MKL) is designed to run on multiple processors and operating systems. It is also compatible with several compilers and third party libraries, and provides different interfaces to the functionality. To support these different environments, tools, and interfaces Intel MKL provides multiple libraries from which to choose.

To see what libraries are recommended for a particular use case, specify the parameters in the drop down lists below.

Intel® Math Kernel Library (Intel® MKL) Link Line Advisor v4.5

Select Intel® product: Intel(R) MKL 11.3.1

Select OS: Linux*

Select usage model of Intel® Xeon Phi™ Coprocessor: Automatic Offload

Select compiler: GNU C/C++

Select architecture: Intel(R) 64

Use this link line:
-Wl,--no-as-needed -L\${MKLROOT}/lib/intel64 -lmkl_intel_lp64 -lmkl_core -lmkl_intel_thread -liomp5 -ldl -lpthread -ln

Compiler options:
-m64 -I\${MKLROOT}/include

Rate Us: ⭐ ⭐ ⭐ ⭐ ⭐

Look for us on: f t g+ y

English >

<https://software.intel.com/en-us/articles/intel-mkl-link-line-advisor>

§6. Resources

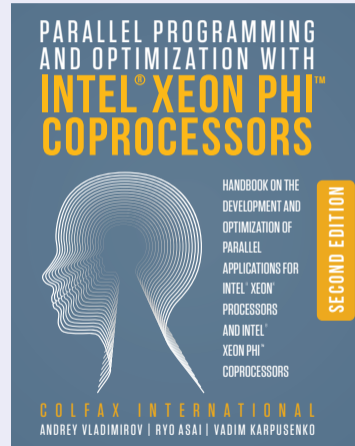
Supplementary Materials: Textbook

ISBN: 978-0-9885234-0-1 (2nd edition, 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>

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 on efficient programming for Intel architecture

The Hands-On Workshop (HOW) Series

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

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 H800 SMR Drives

Optimization Techniques for the Intel MIC Architecture, Part 3 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 1x Acceleration May Be Enough)

Parallel Programming Book

Introduction to parallel programming, deep discussion of optimization techniques, exercises. © 2015, Colfax International. 508 pages.

Featured Video

Research material on vectorization in a streaming code

Events

Presentations

Conferences

Consulting

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

- Optimize your existing application to take advantage of parallelism, from vectors to cores to clusters and beyond.
- Future-proof your application for upcoming Intel architectures.
- Accelerate your application using coprocessor technology.
- Investigate the potential system configurations that satisfy your cost, power, and performance requirements.
- Take a clean sheet to evaluate a novel approach to collaborate with computing partners, software vendors, and hardware manufacturers.

Episode 2.1 - Purpose of the MIC architecture

Software Developer's Introduction to the HGST Ultrastar Archive H800 SMR Drives

The paper will discuss the new HGST Ultrastar Archive H800 SMR drive, software developer tips to obtain optimal storage utilization of 10 TB per segment, 500 TB high density storage capacity, these drives are well suited for large "data archive" applications. In other archive applications, the data is frequently read but seldom modified.

The H800 drives are high capacity, meaning the storage on the drive is in parallel across many cores. Improving the read/write access speed, reducing the read/write latency, and reducing the cost of the drive are the primary goals of the paper. It is given as a guide for the user to optimize the performance of the existing applications with H800.

We will present an example, and then report our test results on the H800 drive, and describe the drive architecture.

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

The paper describes peer-to-peer communication over Gigabit Ethernet and InfiniBand in a cluster with Intel Xeon Phi coprocessors. The paper reports on the performance of the communication over Gigabit Ethernet and InfiniBand in a cluster with Intel Xeon Phi coprocessors. The paper reports on the performance of the communication over Gigabit Ethernet and InfiniBand in a cluster with Intel Xeon Phi coprocessors.

Fluid Dynamics with Fortran on Intel Xeon Phi coprocessors

The Intel Xeon Phi coprocessor is a parallel vector processor designed for high performance computing. This paper describes the use of Fortran on the Intel Xeon Phi coprocessor for fluid dynamics simulations. The paper reports on the performance of the simulations on the Intel Xeon Phi coprocessor.

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

James Reinders is a Senior Software Architect at Intel. He is the author of the book "Intel MIC Architecture: Parallel Programming and Performance". He is also the author of the book "Intel MIC Architecture: Parallel Programming and Performance".

<http://colfaxresearch.com/>

Supplementary Materials: Video Courses



colfaxresearch.com/video-courses

Slides, Code, Video

You can download slides, code and watch the video recording of this webinar here (requires registration for a free Colfax Research account):

colfaxresearch.com/hot-1512

Past webinars (December 8 & 15, 2015: “From Scalar to Serial...” and “Finding the Low-Hanging Fruit for Performance Optimization”) — recordings on same page.