



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.



colfaxresearch.com/hot-16-03

Learn More

Comprehensive Hands-On Workshop (HOW) series begins April 18, 2016. Free remote access to training servers (space is limited!):

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 "HOW" (HANDS ON WORKSHOP) SERIES' in light blue, 'FREE ONLINE TRAINING' in large white letters, 'PARALLEL PROGRAMMING AND OPTIMIZATION' in white, 'FOR INTEL® ARCHITECTURE' in white, and 'STARTS APR 18' in light blue. At the bottom, in smaller white text, it says '*10 2-hour sessions | 24-hour 3-week access to a system | Filling up fast, register now!'.

THE "HOW" (HANDS ON WORKSHOP) SERIES

FREE ONLINE TRAINING

PARALLEL PROGRAMMING AND OPTIMIZATION

FOR INTEL® ARCHITECTURE

STARTS APR 18

*10 2-hour sessions | 24-hour 3-week access to a system | Filling up fast, register now!

colfaxresearch.com/how-16-04

§2. Intel Architecture

Computing Platforms

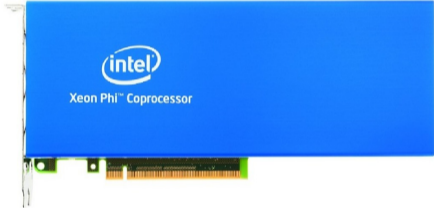
Intel Xeon Processor



Current: Haswell
Upcoming: Broadwell

Multi-Core Architecture

Intel Xeon Phi Coprocessor, 1st generation



Current: Knights Corner (KNC)

Intel Many Integrated Core (MIC) Architecture

Intel Xeon Phi Processor, 2nd generation*



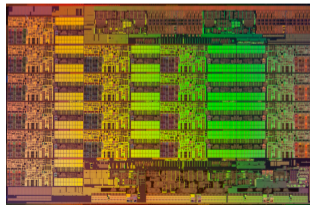
* socket and coprocessor versions

Upcoming: Knights Landing (KNL)

Intel Xeon CPU: Purpose and Specifications

General-purpose platform for demanding computing applications.

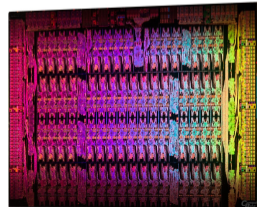
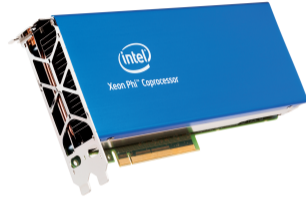
- Up to ~ 1 TFLOP/s in DP
- Up to ~ 2 TFLOP/s in SP
- Up to 768 GiB DDR4 RAM
- ~ 126 GB/s bandwidth
- Hardware-rich: forgiving of sub-optimal code



Intel Xeon Phi Processors (1st Gen)

Specialized platform for demanding computing applications.

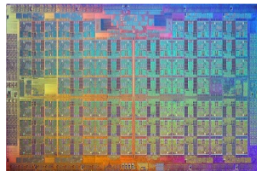
- PCIe end-point device
- ~ 1.2 TFLOP/s in DP
- ~ 2.4 TFLOP/s in SP
- Up to 16 GiB GDDR5 RAM
- ~ 176 GB/s bandwidth
- Heterogeneous clustering
- Runs special Linux distribution



Intel Xeon Phi Processors (2nd Gen)

Specialized platform for demanding computing applications.

- Socket version or coprocessor
- 3+ TFLOP/s in DP
- 6+ TFLOP/s in SP
- Up to 16 GiB MCDRAM
- ~ 400 GB/s MCDRAM bandwidth
- Up to 384 GiB DDR4 RAM
- ~ 90 GB/s DDR4 bandwidth
- Supports common OS
- **Public disclosures**



§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)

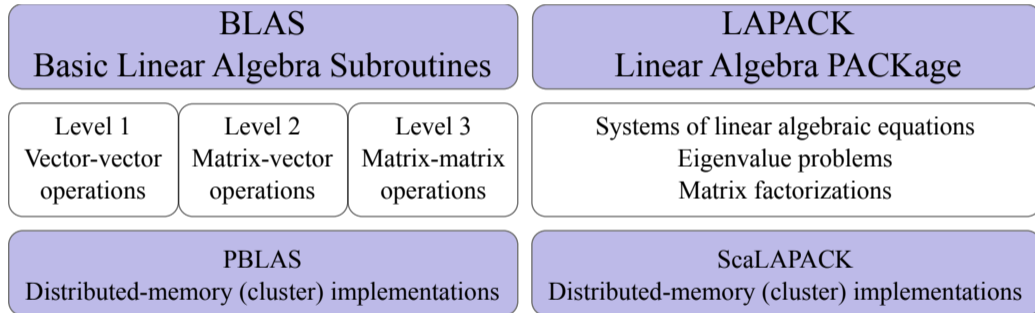
Details in the [HOW Series](#) and [our book](#).

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

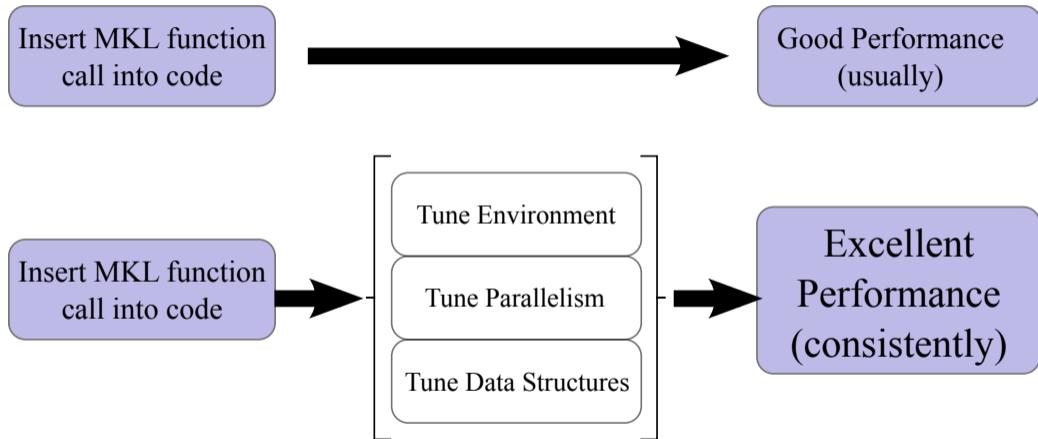
Linear Algebra Support



Using MKL with Intel Xeon Phi Coprocessors

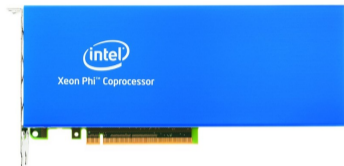
	Native	Compiler-Assisted Offload (CAO)	Automatic Offload (AO)
Programming:	Call MKL functions like from CPU code	Call MKL functions from #pragma offload	Call MKL functions from CPU code
Compilation:	With argument -mmic	Conventional for CPU	Conventional for CPUs
Running	On Xeon Phi	On host CPU	On host 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

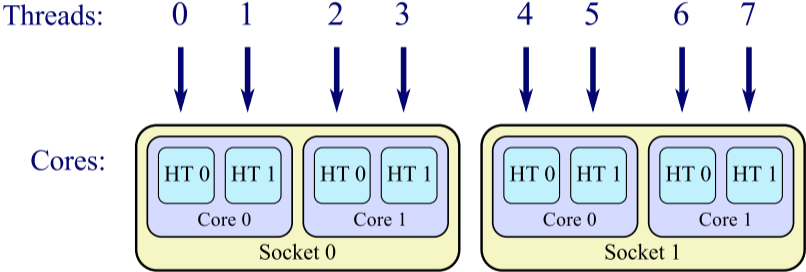
KMP_AFFINITY=scatter

KMP_AFFINITY=compact

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

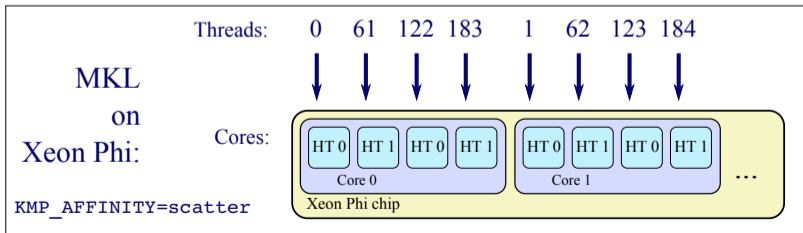
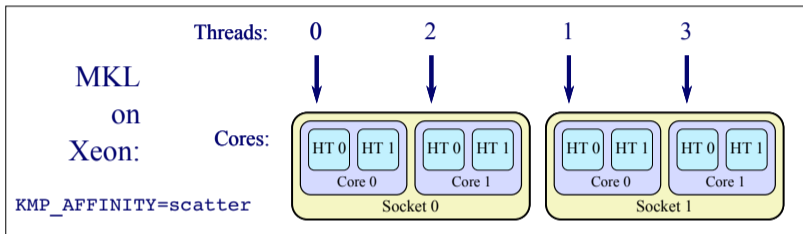
Thread Affinity: Compact Pattern

Generally beneficial for compute-bound applications.



Thread Affinity: Scatter Pattern

Generally beneficial for bandwidth-bound applications.



Data Container Tweaks

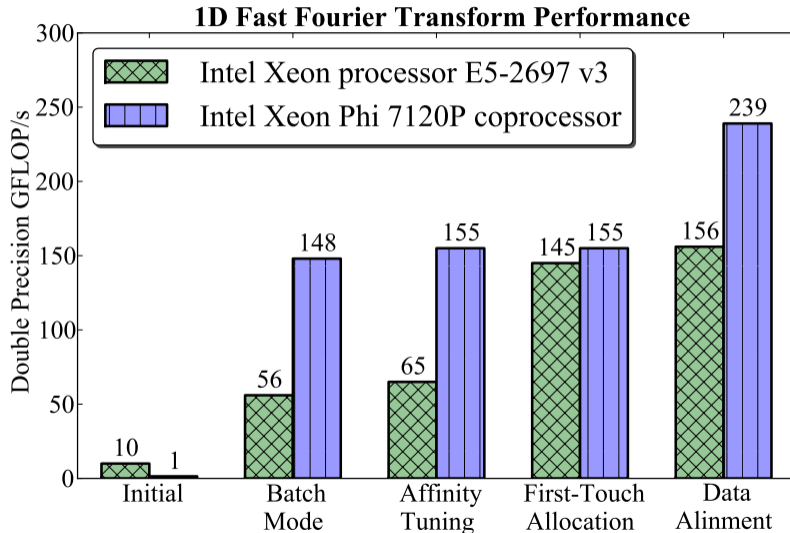
Parallel first-touch (see also HOW series [Session 8](#)):

```
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     }
```

Data alignment (see also HOW series [Session 6](#))

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

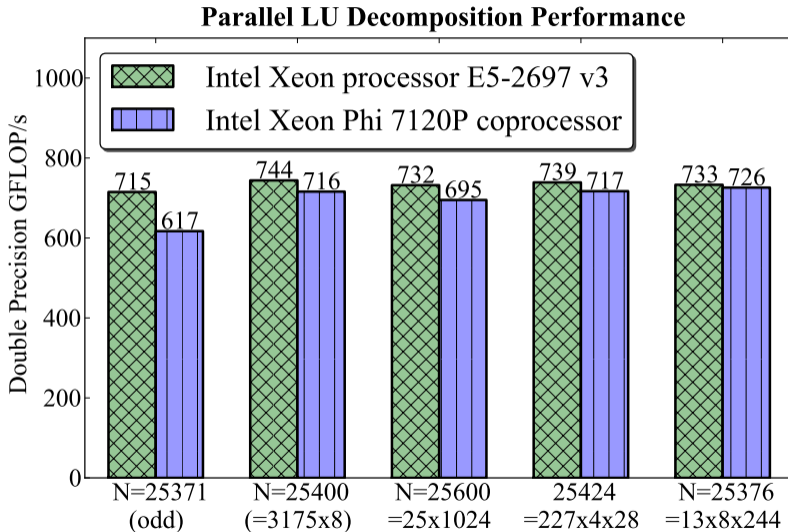
Complex-to-Complex 1D FFT Performance



Problem Size Tuning

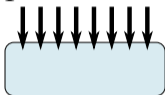
- Some functions require “good” problem sizes
- Good practice #1: array length a multiple of vector length
- Good practice #2: array length a multiple of number of threads
- Good practice #3: array length a multiple of a power of 2
- BLAS and LAPACK have “leading array dimension” for padding rows

LU Decomposition Size Tuning



Nested Parallelism

Fine-grained
parallelism



...

throwing all threads
on one MKL function

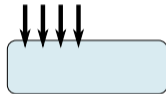
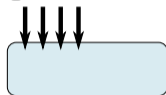
Coarse-grained
parallelism



...

using one thread
per MKL function

Nested
parallelism



...

putting teams of threads
on several MKL functions

OpenMP Hot Teams

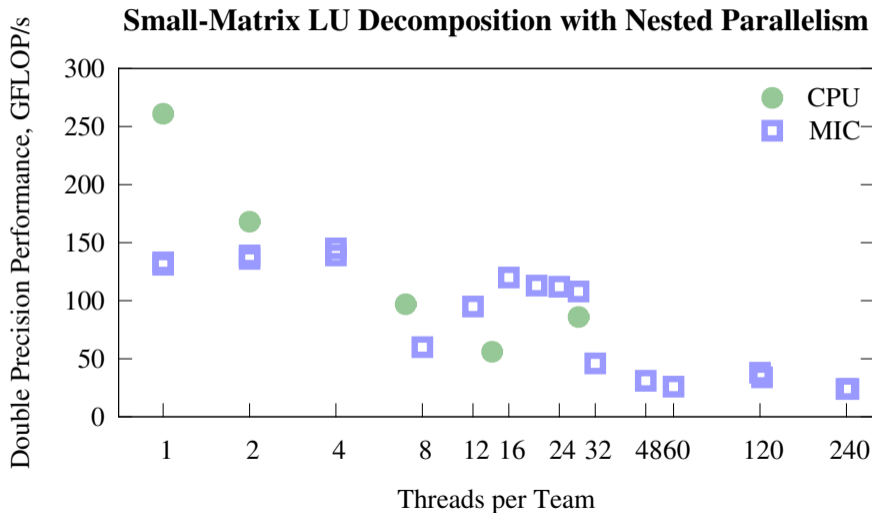
Xeon

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

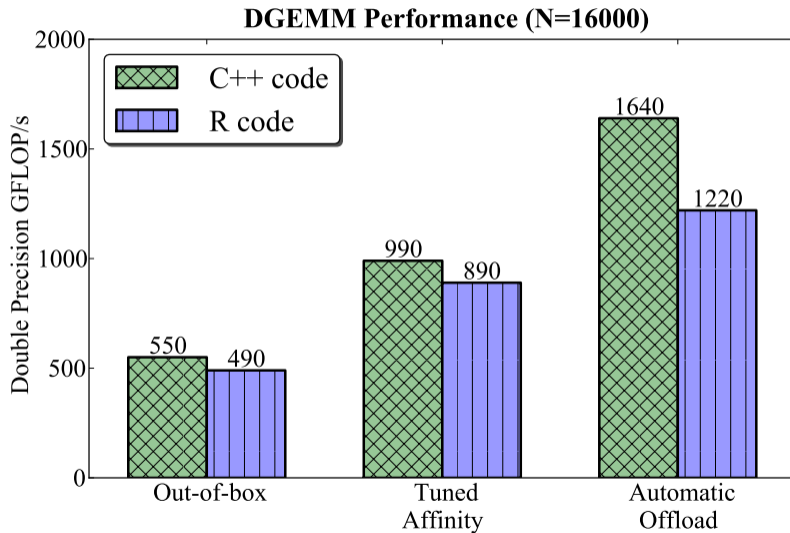
```
./configure \  
  --with-blas="-L/opt/intel/mkl/lib/intel64 \  
              -lmkl_intel_lp64 -lmkl_core -lmkl_intel_thread -lpthread -lm" \  
  --with-lapack \  
  CC=icc CFLAGS="-O2 -qopenmp -I/opt/intel/mkl/include" \  
  CXX=icpc CXXFLAGS="-O2 -qopenmp -I/opt/intel/mkl/include" \  
  F77=ifort 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	Community forum	Intel Premier Support
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

Rate Us:

Look for us on:

Use this link line:

```
-Wl,-no-as-needed -L${MKLR00T}/lib/intel64 -lmkl_intel_lp64 -lmkl_core -lmkl_intel_thread -liomp5 -ldl -lpthread -ln
```

Compiler options:

```
-m64 -I${MKLR00T}/include
```

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

§6. Resources

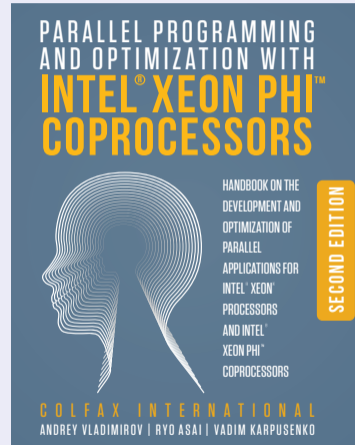
Supplementary Materials: Textbook

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 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)

Parallel Programming Book

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

Featured Video

generated Additional Reading

See Research material on vectorization in a streaming code

Events

Presentations

Conferences

Consulting

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

- 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 across computing platforms, software, and hardware

Episode 2.1 – Purpose of the MIC architecture

Intel® Xeon™ Coprocessor - QOP 9M1 - Chapter 2 - Episode 2.1

Intel® Xeon™ Coprocessor - QOP 9M1 - Chapter 2 - Episode 2.1

Intel® Xeon™ Coprocessor - QOP 9M1 - Chapter 2 - Episode 2.1

Fluid Dynamics with Fortran on Intel Xeon Phi coprocessors

Intel® Xeon™ Phi coprocessors are a platform where Fortran can be used to solve complex problems. The performance of Fortran on Intel Xeon Phi coprocessors is compared to Intel Xeon E5-2680 v2. The Intel Xeon Phi coprocessor is a many-core processor with 37.7 billion transistors and 60 cores. The Intel Xeon Phi coprocessor is a many-core processor with 37.7 billion transistors and 60 cores.

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

Intel® Xeon™ Phi coprocessors are a platform where Fortran can be used to solve complex problems. The performance of Fortran on Intel Xeon Phi coprocessors is compared to Intel Xeon E5-2680 v2. The Intel Xeon Phi coprocessor is a many-core processor with 37.7 billion transistors and 60 cores. The Intel Xeon Phi coprocessor is a many-core processor with 37.7 billion transistors and 60 cores.

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

Intel® Xeon™ Phi coprocessors are a platform where Fortran can be used to solve complex problems. The performance of Fortran on Intel Xeon Phi coprocessors is compared to Intel Xeon E5-2680 v2. The Intel Xeon Phi coprocessor is a many-core processor with 37.7 billion transistors and 60 cores. The Intel Xeon Phi coprocessor is a many-core processor with 37.7 billion transistors and 60 cores.

<http://colfaxresearch.com/>

Learn More

Comprehensive Hands-On Workshop (HOW) series begins April 18, 2016. Free remote access to training servers (space is limited!):



THE "HOW" (HANDS ON WORKSHOP) SERIES

FREE ONLINE TRAINING

PARALLEL PROGRAMMING AND OPTIMIZATION

FOR INTEL® ARCHITECTURE

STARTS APR 18

*10 2-hour sessions | 24-hour 3-week access to a system | Filling up fast, register now!

colfaxresearch.com/how-16-04

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-16-03

Past webinars (March 21 & 23, 2016: “From Scalar to Serial...” and “Finding the Low-Hanging Fruit for Performance Optimization”) — recordings on same page.