Practical Usage of the Intel Math Kernel Library (MKL)

The Hands-On Tutorial (HOT) Series

Andrey Vladimirov, PhD — Colfax International
colfaxresearch.com

 © Colfax International, 2013–2015

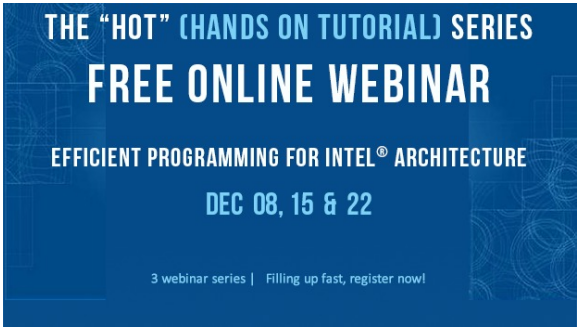
Disclaimer

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.

© Colfax International, 2013–2015

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-1512

© Colfax International, 2013–2015

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.

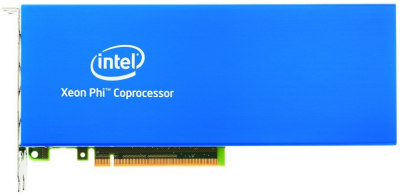
© Colfax International, 2013–2015

§2. Intel Architecture

© Colfax International, 2013–2015

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

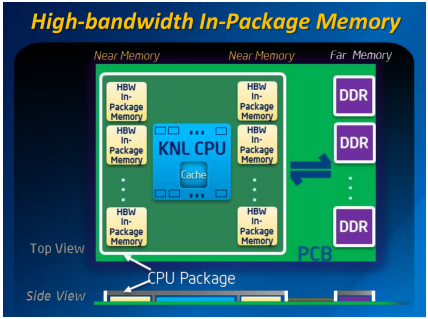


- C/C++/Fortran; OpenMP/MPI
- Special Linux distribution
- 6–16 GB cached GDDR5 RAM
- 57 to 61 cores at ≈1 GHz
- 4 hardware threads per core
- 512-bit IMCI vectors

© Colfax International, 2013–2015

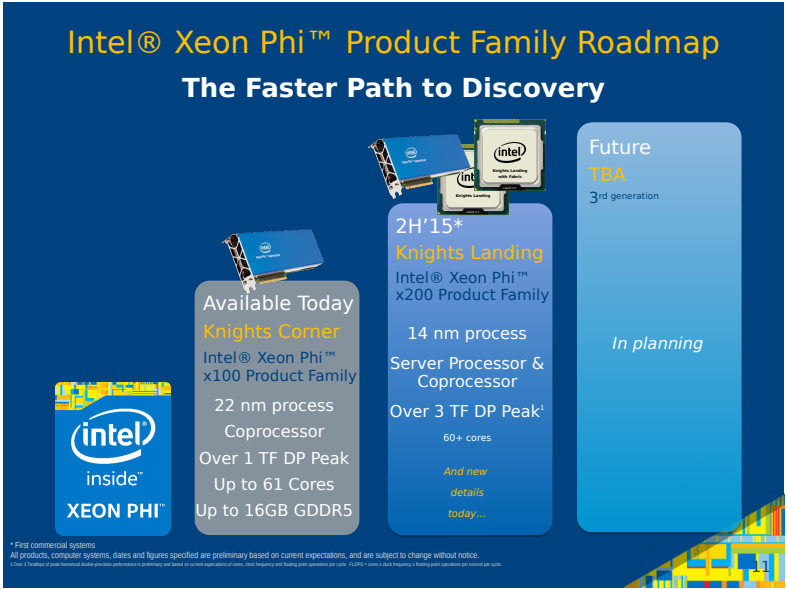
Next Generation of the MIC Architecture

- 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

 © Colfax International, 2013–2015



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

§3. Intel Math Kernel Library

© Colfax International, 2013–2015

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.

© Colfax International, 2013–2015

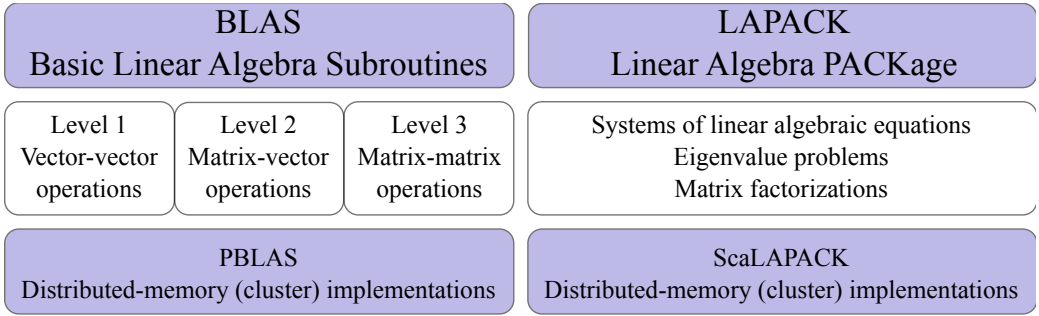
Scope of MKL

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	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

© Colfax International, 2013–2015

Linear Algebra Support



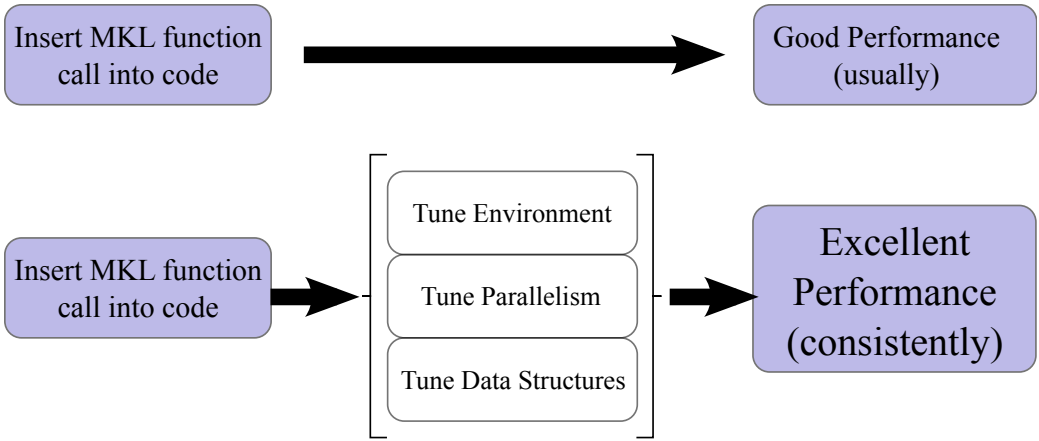
 © Colfax International, 2013–2015

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

© Colfax International, 2013–2015

Performance Tuning with MKL

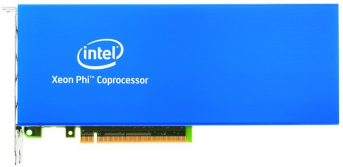


 © Colfax International, 2013–2015

§4. Hands-On Labs

© Colfax International, 2013–2015

“Standard Candle” Testbench



vs.

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

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

See also “Intel Xeon Product Family: Performance Brief”

© Colfax International, 2013–2015

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);
```


© Colfax International, 2013–2015

Thread Affinity Tuning

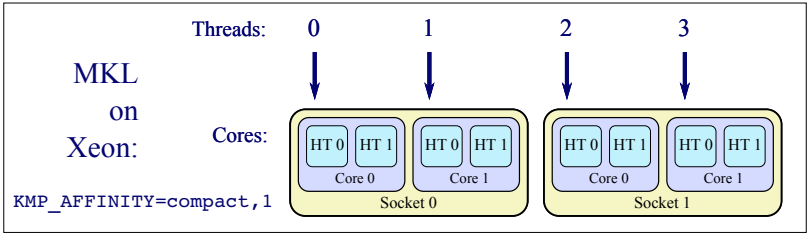
Xeon	**Xeon Phi**
MKL uses 1 thread/core	MKL uses 4 threads/core
`OMP_NUM_THREADS`=#phys. cores	`OMP_NUM_THREADS`=4×#cores
`KMP_AFFINITY`=scatter	`KMP_AFFINITY`=scatter
`KMP_AFFINITY`=compact (HT off)	`KMP_AFFINITY`=compact
`KMP_AFFINITY`=compact,1 (HT on)	

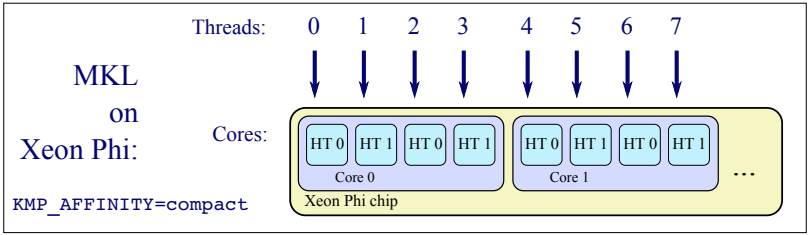
See also HOW series Episode 3.4.

© Colfax International, 2013–2015

Thread Affinity: Compact Pattern

Generally beneficial for compute-bound applications.

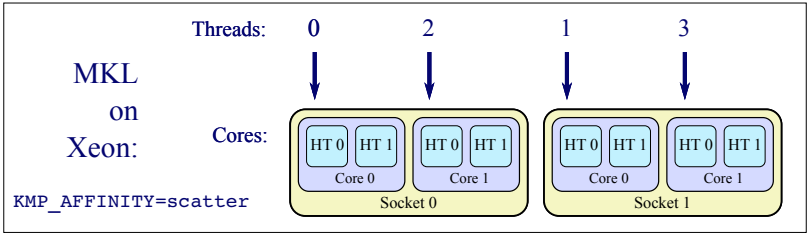


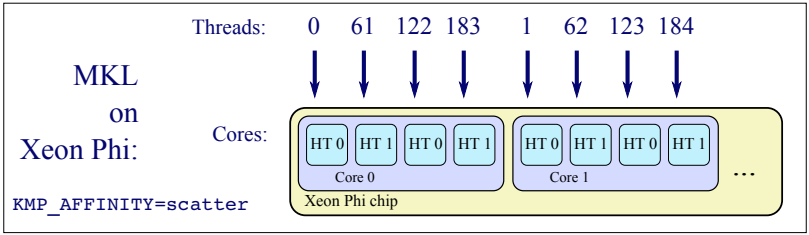


 © Colfax International, 2013–2015

Thread Affinity: Scatter Pattern

Generally beneficial for bandwidth-bound applications.





 © Colfax International, 2013–2015

Data Container Tweaks

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

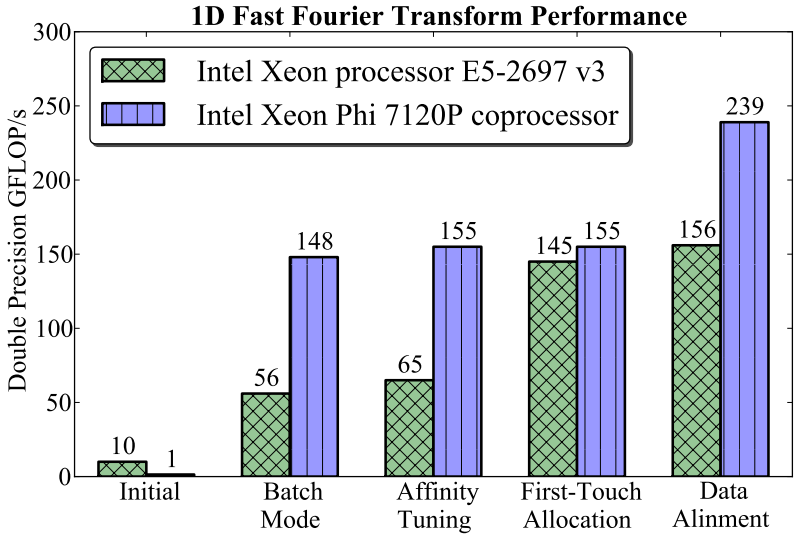
```
#pragma omp parallel for
  for(int i = 0; i < num_fft; i++)
    for(int j = 0; j < fft_size; j++) {
      data[i*fft_size+j].real = cos(k*j);
      data[i*fft_size+j].imag = sin(k*j);
    }
```

Data alignment (see also HOW series Episode 3.2)

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


© Colfax International, 2013–2015

Complex-to-Complex 1D FFT Performance



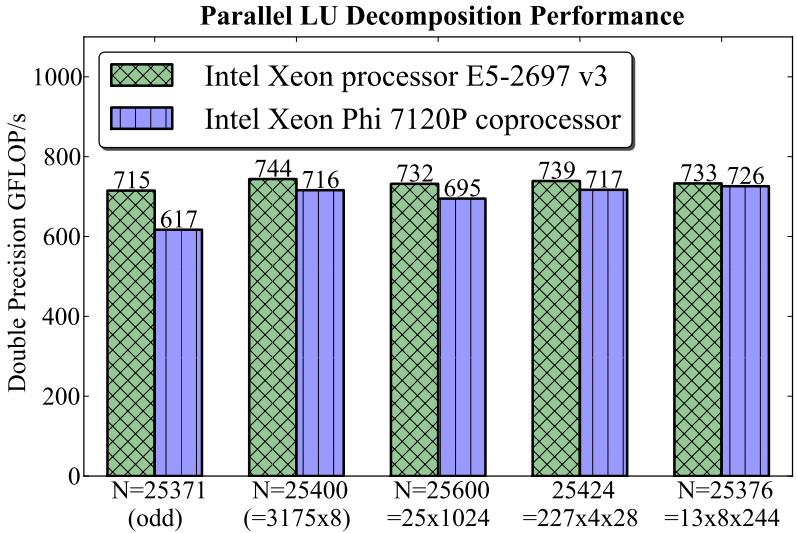
© Colfax International, 2013–2015

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

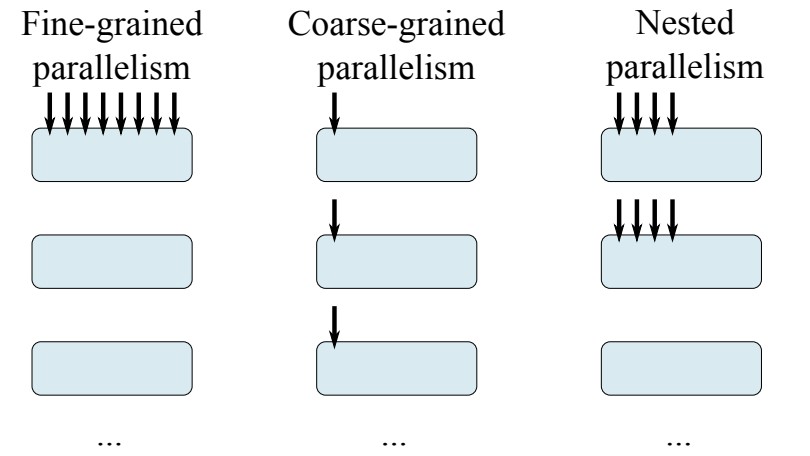
© Colfax International, 2013–2015

LU Decomposition Size Tuning



 © Colfax International, 2013–2015

Nested Parallelism



 © Colfax International, 2013–2015

OpenMP Hot Teams

Xeon	Xeon Phi
`OMP_NUM_THREADS=2,14`	`OMP_NUM_THREADS=60,4`

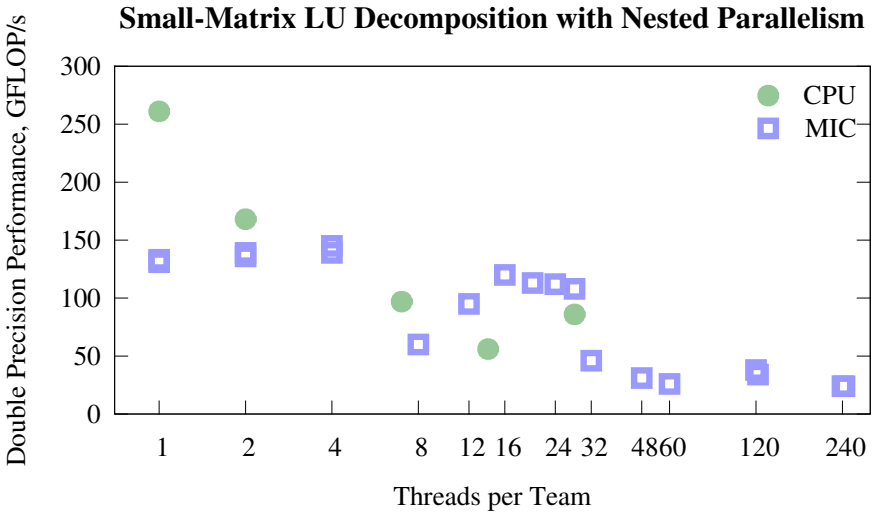
`OMP_NESTED=1`
`OMP_PROC_BIND=spread,close`
`OMP_PLACES=cores`
	`OMP_NESTED=1`
	`OMP_PROC_BIND=spread,close`
	`OMP_PLACES=threads`

`KMP_HOT_TEAMS_MODE=1`
`KMP_HOT_TEAMS_MAX_LEVEL=2`
`OMP_MAX_ACTIVE_LEVELS=2`
	`KMP_HOT_TEAMS_MODE=1`
	`KMP_HOT_TEAMS_MAX_LEVEL=2`
	`OMP_MAX_ACTIVE_LEVELS=2`

`MKL_DYNAMIC=false`	`MKL_DYNAMIC=false`

© Colfax International, 2013–2015

Small Matrix LU Decomposition with Nested Parallelism



 © Colfax International, 2013–2015

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
```

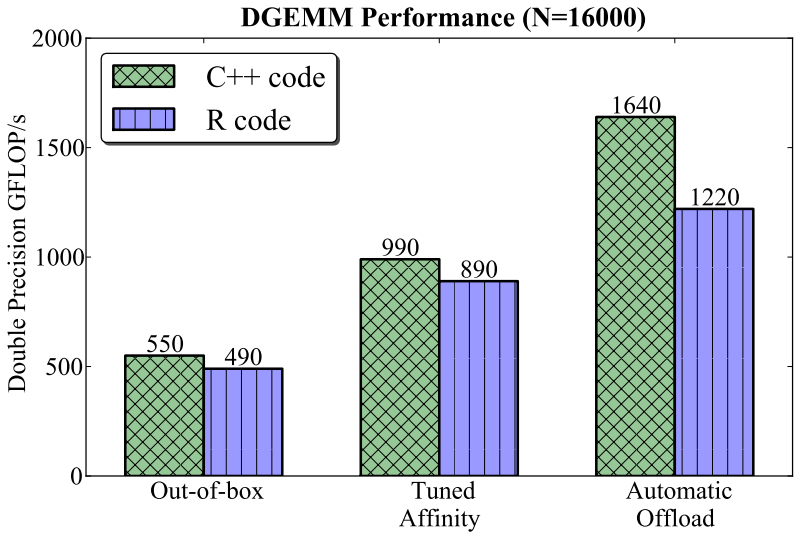

© Colfax International, 2013–2015

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
```


© Colfax International, 2013–2015

Performance with C++ Code and in R



 © Colfax International, 2013–2015

§5. Additional Information

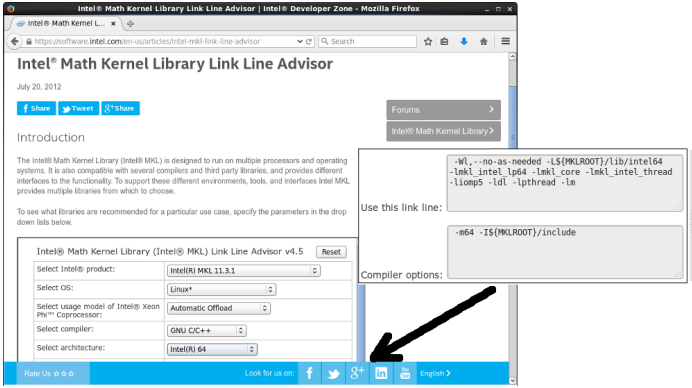
© Colfax International, 2013–2015

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

© Colfax International, 2013–2015

Compilation with MKL



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

 © Colfax International, 2013–2015

§6. Resources

© Colfax International, 2013–2015

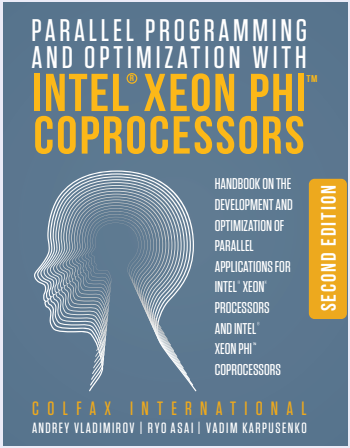
Supplementary Materials: Textbook

ISBN: 978-0-9885234-0-1 (2nd edition, 508 pages, Electronic or Print)

Parallel Programming
and Optimization with
Intel® Xeon Phi™
Coprocessors

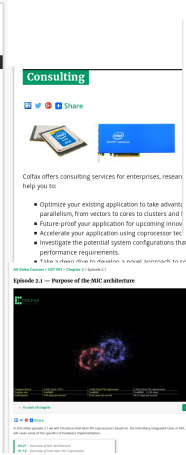
Handbook on the Development and
Optimization of Parallel Applications
for Intel® Xeon® Processors
and Intel® Xeon Phi™ Coprocessors

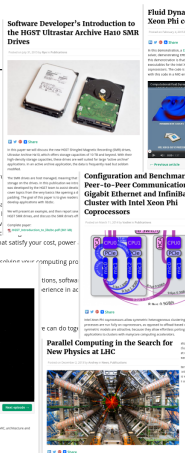
© Colfax International, 2015

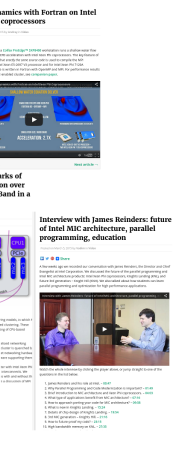


http://xeonphi.com/book

Colfax Research







http://colfaxresearch.com/

 © Colfax International, 2013–2015

Supplementary Materials: Video Courses

colfaxresearch.com/video-courses

© Colfax International, 2013–2015

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.

© Colfax International, 2013–2015