Get Ready for Intel® MKL on Intel® Xeon Phi™ Coprocessors

Zhang Zhang
Technical Consulting Engineer
Intel® Math Kernel Library
Legal Disclaimer

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL’S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL® PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Intel may make changes to specifications and product descriptions at any time, without notice. All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice. Intel, processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Sandy Bridge and other code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to http://www.intel.com/performance

Intel, Core, Xeon, VTune, Cilk, Intel and Intel Sponsors of Tomorrow, and Intel Sponsors of Tomorrow logo, and the Intel logo are trademarks of Intel Corporation in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Copyright ©2011 Intel Corporation.

Hyper-Threading Technology: Requires an Intel® HT Technology enabled system, check with your PC manufacturer. Performance will vary depending on the specific hardware and software used. Not available on all Intel® Core™ processors. For more information including details on which processors support HT Technology, visit http://www.intel.com/info/hyperthreading

Intel® 64 architecture: Requires a system with a 64-bit enabled processor, chipset, BIOS and software. Performance will vary depending on the specific hardware and software you use. Consult your PC manufacturer for more information. For more information, visit http://www.intel.com/info/em64t

Intel® Turbo Boost Technology: Requires a system with Intel® Turbo Boost Technology capability. Consult your PC manufacturer. Performance varies depending on hardware, software and system configuration. For more information, visit http://www.intel.com/technology/turboboost

*Other brands and names are the property of their respective owners.
Agenda

• Intel® MKL supports Intel® Xeon Phi™ Coprocessor

• Intel® MKL usage models on Intel® Xeon Phi™
  - Automatic Offload
  - Compiler Assisted Offload
  - Native Execution

• Performance

• Where to find more information?
Using Intel® MKL on Intel® Xeon Phi™ Coprocessors
Intel® MKL is industry’s leading math library *

<table>
<thead>
<tr>
<th>Linear Algebra</th>
<th>Fast Fourier Transforms</th>
<th>Vector Math</th>
<th>Vector Random Number Generators</th>
<th>Summary Statistics</th>
<th>Data Fitting</th>
</tr>
</thead>
<tbody>
<tr>
<td>• BLAS</td>
<td>• Multidimensional (up to 7D)</td>
<td>• Trigonometric</td>
<td>• Congruential</td>
<td>• Kurtosis</td>
<td>• Splines</td>
</tr>
<tr>
<td>• LAPACK</td>
<td>• FFTW interfaces</td>
<td>• Hyperbolic</td>
<td>• Recursive</td>
<td>• Variation</td>
<td>• Interpolation</td>
</tr>
<tr>
<td>• Sparse solvers</td>
<td>• Cluster FFT</td>
<td>• Exponential, Logarithmic</td>
<td>• Wichmann-Hill</td>
<td>• Quantiles, order</td>
<td>• Cell search</td>
</tr>
<tr>
<td>• ScalAPACK</td>
<td></td>
<td>• Power / Root</td>
<td>• Mersenne Twister</td>
<td>statistics</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Rounding</td>
<td>• Sobol</td>
<td>• Min/max</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Neiderreiter</td>
<td>• Variance-</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>covariance</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* 2011 & 2012 Evans Data N. American developer surveys

* Other brands and names are the property of their respective owners.
Intel® MKL Supports for Intel® Xeon Phi™ Coprocessors

- Intel® MKL 11.0 and above supports the Intel® Xeon Phi™ coprocessors.

- Heterogeneous computing
  - Takes advantage of both multicore host and many-core coprocessors.

- Optimized for wider (512-bit) SIMD instructions and threaded for many cores.

- All Intel MKL functions are supported:
  - But optimized at different levels.

Pairing *highly parallel* software with *highly parallel* hardware.
Highly Optimized Functions

BLAS Level 3, and much of Level 1 & 2

Sparse BLAS: ?CSRMV, ?CSRMM

Some important LAPACK routines (LU, QR, Cholesky)

Fast Fourier transforms

Vector Math Library

Random number generators in the Vector Statistical Library

Broader functionality to be optimized in future update releases.
Usage Models on Intel® Xeon Phi™ Coprocessors

Automatic Offload
- No code changes required
- Automatically uses both host and target
- Transparent data transfer and execution management

Compiler Assisted Offload
- Explicit controls of data transfer and remote execution using compiler offload pragmas/directives
- Can be used together with Automatic Offload

Native Execution
- Uses the coprocessors as independent nodes
- Input data and binaries are copied to targets in advance
Automatic Offload (AO)

- Offloading is automatic and transparent.
- Can take advantage of multiple coprocessors.
- By default, Intel MKL decides:
  - When to offload
  - Work division between host and targets
- Users enjoy host and target parallelism automatically.
- Users can still specify work division between host and target. (for BLAS only)
How to Use Automatic Offload

- Using Automatic Offload is easy:

  Call a function: `mkl_mic_enable()`

  Set an env variable: `MKL_MIC_ENABLE=1`

- What if there doesn’t exist a coprocessor in the system?
  - Runs on the host as usual **without penalty**!

- The context of Automatic Offload is a single function.
Automatic Offload Enabled Functions

• A selected set of MKL functions are AO enabled.
  - Only functions with sufficient computation to offset data transfer overhead are subject to AO
    - LAPACK 3 amigos: LU, QR, Cholesky

• Offloading happens only when matrix sizes are right. The following are dimension sizes in numbers of elements.
  - ?GEMM: M, N > 2048, K > 256
  - ?SYMM: M, N > 2048
  - ?TRSM/?TRMM: M, N > 3072
  - LU: M, N > 8192
# Work Division Control in Automatic Offload

<table>
<thead>
<tr>
<th>Examples</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>mkl_mic_set_Workdivision(MKL_TARGET_MIC, 0, 0.5)</td>
<td>Offload 50% of computation only to the 1st card.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Examples</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MKL_MIC_0_WORKDIVISION=0.5</td>
<td>Offload 50% of computation only to the 1st card.</td>
</tr>
</tbody>
</table>

Work division settings have no effects for LAPACK functions.
Compiler Assisted Offload (CAO)

Offloading is explicitly controlled by compiler pragmas or directives.

All MKL functions can be offloaded in CAO.
- In comparison, only a subset of MKL is subject to AO.

Can leverage the full potential of compiler’s offloading facility.

Can offload multiple MKL functions using one offload region.

More flexibility in data transfer and remote execution management.
- A big advantage is data persistence: Reusing transferred data for multiple operations.
How to Use Compiler Assisted Offload

• The same way you would offload any function call to the coprocessor.

• An example in C:

```c
#pragma offload target(mic) \
    in(transa, transb, N, alpha, beta) \
    in(A:length(matrix_elements)) \
    in(B:length(matrix_elements)) \
    in(C:length(matrix_elements)) \
    out(C:length(matrix_elements) alloc_if(0))
{
    sgemm(&transa, &transb, &N, &N, &N, &alpha, A, &N, B, &N, &beta, C, &N);
}
```
How to Use Compiler Assisted Offload

• An example in Fortran:

```fortran
!DEC$ ATTRIBUTES OFFLOAD : TARGET( MIC ) :: SGEMM
!DEC$ OMP OFFLOAD TARGET( MIC ) &
!DEC$ IN( TRANSA, TRANSB, M, N, K, ALPHA, BETA, LDA, LDB, LDC ), &
!DEC$ IN( A: LENGTH( NCOLA * LDA ) ), &
!DEC$ IN( B: LENGTH( NCOLB * LDB ) ), &
!DEC$ INOUT( C: LENGTH( N * LDC ) )
!$OMP PARALLEL SECTIONS
!$OMP SECTION
   CALL SGEMM( TRANSA, TRANSB, M, N, K, ALPHA, &
               A, LDA, B, LDB BETA, C, LDC )
!$OMP END PARALLEL SECTIONS
```
Using AO and CAO in the Same Program

- Users can use AO for some MKL calls and use CAO for others in the same program.
  - Only supported by Intel compilers.
  - Work division must be set explicitly for AO.
  - Otherwise, all MKL AO calls are executed on the host.

- Set ‘OFFLOAD_ENABLE_ORSL=1’ for better resource synchronization.

- Can be done simultaneously from different threads.
  - For example, one thread doing AO, the other doing CAO.
Native Execution

• Use the coprocessor as an independent compute node.
  - Programs can be built to run only on the coprocessor by using the
    "-mmic" build option.

• MKL function calls inside an offloaded code region executes natively.
  - Better performance if input data is already available on the
    coprocessor, and output is not immediately needed on the host side.
Considerations of Using Intel® MKL on Intel® Xeon Phi™ Coprocessors

High level parallelism is critical in maximizing performance.

- BLAS (Level 3) and LAPACK with large problem size get the most benefit.
- Scaling beyond 100’s threads, vectorized, good data locality

Minimize data transfer overhead when offload.

- Offset data transfer overhead with enough computation.
- Exploit data persistence: CAO to help!

You can always run on the host if offloading does not offer better performance.
Performance

Matrix Multiply Performance using Intel® Math Kernel Library
on Intel® Xeon Phi™ Coprocessor 7120P and Intel® Xeon® Processor E5-2697 v2

DGEMM

Configuration Info: Software Versions: Intel® Math Kernel Library (Intel® MKL) 11.1, Intel® Manycore Platform Software Stack (MPSS) 2.1.6720-15; Hardware: Intel® Xeon® Processor E5-2697 v2, 2 Twelve-Core CPUs (30MB LLC, 2.7GHz), 32GB DDR3 RAM (1333MHz); vs. one Intel® Xeon Phi™ Coprocessor 7120P, 61 cores (30.5MB total cache, 1.239GHz), 16GB DDR5 Memory; Operating System: RHEL 6.1 GA x86_64.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. * Other brands and names are the property of their respective owners. Benchmark Source: Intel Corporation, September 2013.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations are not specific to Intel microarchitecture and are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804
LU Factorization Performance using Intel® Math Kernel Library on Intel® Xeon Phi™ Coprocessors 7120P and Intel® Xeon® Processor E5-2697 v2

Configuration Info - Software Versions: Intel® Math Kernel Library (Intel® MKL) 11.1, Intel® Manycore Platform Software Stack (MPSS) 2.1.6b20-15; Hardware: Intel® Xeon® Processor E5-2697 v2, 2 Twelve-Core CPUs (30MB LLC, 2.7GHz), 32GB DDR3 RAM (1333MHz); Intel® Xeon Phi™ Coprocessor 7120P, 61 cores (30.5MB total cache, 1.23GHz), 16GB DDR5 Memory; Operating System: RHEL 6.1 GA x86_64.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. * Other brands and names are the property of their respective owners. Benchmark Source: Intel Corporation, September 2013.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804
QR Factorization Performance using Intel® Math Kernel Library on Intel® Xeon Phi™ Coprocessors 7120P and Intel® Xeon® Processor E5-2697 v2

Configuration Info: Software Versions: Intel® Math Kernel Library (Intel® MKL) 11.1, Intel® Manycore Platform Software Stack (MPSS) 2.1.6720-15; Hardware: Intel® Xeon® Processor E5-2697 v2, 2 Twelve-Core CPUs (30MB LLC, 2.7GHz), 32GB DDR3 RAM (1333MHz); Intel® Xeon Phi™ Coprocessor 7120P, 61 cores (30.5MB total cache, 1.239GHz), 16GB DDR3 Memory; Operating System: RHEL 6.1 GA x86_64.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYsmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. * Other brands and names are the property of their respective owners. Benchmark Source: Intel Corporation. September 2013.

Optimization Notice: Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804.
Cholesky Factorization Performance using Intel® Math Kernel Library on Intel® Xeon Phi™ Coprocessor 7120P and Intel® Xeon® Processor E5-2697 v2

Performance (Gflops)

Matrix Size

- Native Execution on Intel® Xeon Phi™ Coprocessor 7120P
- Automatic Offload with 1 Intel® Xeon Phi™ Coprocessor 7120P
- Intel® Xeon® Processor E5-2697 v2

Configuration Info - Software Versions: Intel® Math Kernel Library (Intel® MKL) 11.1, Intel® Manycore Platform Software Stack (MPSS) 2.1, 6720-15; Hardware: Intel® Xeon® Processor E5-2697 v2.2 Twelve-Core CPUs (30MB LLC. 2.7GHz), 32GB DDR3 RAM (1333MHz); Intel® Xeon Phi™ Coprocessor 7120P, 61 cores (30.5MB total cache, 1.23GHz), 16GB DDR5 Memory; Operating System: RHEL 6.1 GA x86_64.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. * Other brands and names are the property of their respective owners. * Benchmarks Source: Intel Corporation, September 2013.

Optimization Notice: Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimization techniques. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804.
Batch Complex 1D FFT Performance using Intel® Math Kernel Library on Intel® Xeon Phi™ Coprocessor 7120P and Intel® Xeon® Processor E5-2697 v2

![Graphs showing performance comparison between Single Precision and Double Precision for different transform lengths/batch sizes.](image-url)

Configuration Info - Software Versions: Intel® Math Kernel Library (Intel® MKL) 11.1, Intel® Manycore Platform Software Stack (MPSS) 2.1.6720-15: Hardware: Intel® Xeon® Processor E5-2697 v2, 2 Twelve-Core CPUs (30MB LLC, 2.7GHz), 32GB DDR3 RAM (1333MHz); vs. one Intel® Xeon Phi™ Coprocessor 7120P, 61 cores (30.5MB total cache, 1.238 GHz), 16GB DDR5 Memory; Operating System: RHEL 6.1 GA x86_64.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. * Other brands and names are the property of their respective owners. Benchmark Source: Intel Corporation. September 2013.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804
Performance Tuning Hints

Problem size considerations:
- Large problems: more parallelism (limited by avail. memory)
- FFT: prefer power-of-2 (POT) sizes

Data alignment consideration:
- 64-byte alignment for better vectorization
- FFT: additional considerations for 2d+

OpenMP* thread count and thread affinity:
- Avoid thread migration for better data locality

Large (2 MB) pages for memory allocation:
- Reduce TLB misses and memory allocation overhead
- Libhugetlbf, \texttt{mmap()}, MIC\_USE\_2MB\_BUFFERS=<threshold>
- Automatic use is incorporated into newer Intel MPSS (THP)

Tips are documented and maintained here:
Performance Tuning: Alignment

Compiler-assisted offload

• Memory alignment is inherited from host!
• Fast DMA transfers: align with page-granularity

General memory alignment (SIMD vectorization)

• Align buffers (leading dimension) to a multiple of vector width (64 Byte)
• Use* mkl_malloc, _mm_malloc (_aligned_malloc), or tbb::scalable_aligned_malloc

* Remember to call the corresponding free-function.
Performance Tuning: FFT

Memory alignment for 2d (and higher) FFTs

- Single-precision (SP): strides divisible by 8 but not divisible by 16
- Double-precision (DP): strides divisible by 4 but not divisible by 8

Consider single call interface in case of parallelizing a series of individual 1d FFTs!
Performance Tuning: Affinity

Intel MKL threading runtime is OpenMP*

Environment variables OMP_* (similar MKL_* variables take precedence)
- Coprocessor (CAO): MIC_ENV_PREFIX=MIC MIC_OMP_NUM_THREADS=…

Intel OpenMP thread affinity

KMP_AFFINITY=<see below>
- Host: e.g., compact,1
- Coprocessor: balanced

MIC_ENV_PREFIX=MIC  MIC_KMP_AFFINITY=<see below>
- Coprocessor (CAO): balanced

KMP_PLACE_THREADS
- New! Note: does not replace KMP_AFFINITY
- Helps to set/achieve pinning on e.g., 60 cores with 3 threads each
  kmp_* (or mkl_*) functions take precedence over corresponding env. variables

Intel MPI process affinity

I_MPI_* variables
More information:
- Linking on Intel® Xeon Phi™
- Online resources
Intel® MKL Link Line Advisor

A web tool to help users to choose correct link line options.


Also available offline in the MKL product package.
Linking Examples

AO: The same way of building code on Xeon!

```bash
icc -O3 -mkl sgemm.c -o sgemm.exe
```

Native: Using `-mmic`

```bash
icc -O3 -mmic -mkl sgemm.c -o sgemm.exe
```

CAO: Using `-offload-option`

```bash
icc -O3 -openmp -mkl \
   -offload-option,mic,ld, "-L$MKLROOT/lib/mic -Wl,\ 
   --start-group -lmkl_intel_lp64 -lmkl_intel_thread \ 
   -lmkl_core -Wl,--end-group" sgemm.c -o sgemm.exe
```
Where to Find Code Examples

$MKLROOT/examples/mic_ao
- sgemm AO example

$MKLROOT/examples/mic_offload
- dexp VML example (vdExp)
- dgaussian double precision Gaussian RNG
- fft complex-to-complex 1D FFT
- sexp VML example (vsExp)
- sgaussian single precision Gaussian RNG
- sgemm SGEMM example
- sgemm_f SGEMM example (Fortran 90)
- sgemm_reuse SGEMM with data persistence
- sgeqrf QR factorization
- sgetrf LU factorization
- spotrf Cholesky
- solverc PARDISO examples
Online Resources

How-to articles, tips, case studies, hands-on lab:


Performance charts online:


The MIC developer community:


Intel® MKL forum:
