Using Intel® Math Kernel Library on Intel® Xeon Phi™ Coprocessors

Sridevi Allam - Technical Consulting Engineer
Intel Corporation (SSG|DPD)
Agenda

- Overview of Intel® MKL
- Introduction to Support on Intel® Xeon Phi Coprocessors
- Performance Charts
- Link Line Advisor
- MKL 11.1 New features
- Documentation References
Intel® MKL is industry’s leading math library

**Linear Algebra**
- BLAS
- LAPACK
- Sparse solvers
- ScalAPACK

**Fast Fourier Transforms**
- Multidimensional (up to 7D)
- FFTW interfaces
- Cluster FFT

**Vector Math**
- Trigonometric
- Hyperbolic
- Exponential, Logarithmic
- Power / Root
- Rounding

**Vector Random Number Generators**
- Congruential
- Recursive
- Wichmann-Hill
- Mersenne Twister
- Sobol
- Neiderreiter
- Non-deterministic

**Summary Statistics**
- Kurtosis
- Variation coefficient
- Quantiles, order statistics
- Min/max
- Variance-covariance
- ...  

**Data Fitting**
- Splines
- Interpolation
- Cell search

* 2011 & 2012 Evans Data N. American developer surveys
Intel® Math Kernel Library (Intel® MKL) Support for Intel® Xeon Phi™ Coprocessors

• Support for the Intel® Xeon Phi™ coprocessors is introduced starting Intel® MKL 11.0

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

• All Intel MKL functions are supported:
  - But optimized at different levels.
Highly Optimized Functions

As of Intel® Math Kernel Library 11.1:

- 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
Usage Models on Intel® Xeon Phi™ Coprocessors

- Automatic Offload

- Compiler Assisted Offload

- Native Execution
Automatic Offload (AO)

• Offloading is automatic and transparent.
  - No code changes required
  - Automatically uses both host and target

• Can take advantage of multiple coprocessors.

• By default, Intel® Math Kernel Library decides:
  - When to offload
  - Work division between host and targets
AO Contd..

• Users enjoy host and target parallelism automatically.

• Users can still specify work division between host and target.

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!
Work Division control in AO

Examples:

mkl_mic_set_Workdivision(MKL_TARGET_MIC, 0, 0.5) : Offload 50% of computation only to the 1st Card

MKL_MIC_0_WORKDIVISION=0.5 : Offload 50% of computation only to the 1st Card
Usage Models Contd ..

• Compiler Assisted Offload (CAO)
  - Explicit controls of data transfer and remote execution using compiler offload pragmas/directives
  - Can be used together with Automatic Offload
  - Offloading is explicitly controlled by compiler pragmas or directives.
  - All Intel® Math Kernel Library (Intel® MKL) functions can be offloaded in CAO.
  - Can leverage the full potential of compiler’s offloading facility.
Usage Models Contd ... 

- 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, &N, &beta, C, &N);

}
```
Usage Models Contd ...

Native Execution:

- Input data and binaries are copied to targets in advance

Ex: Build the code like : `icc -mmic -mkl mkl_dft_1d.c`

And manually upload the binary executable and dependent libraries to the target and ssh into target machine and run from there

- If MKL function call is inside an offload region, it consumes input and produces output only inside this offload region
Linking Examples

**AO:** The same way of building code on Intel® Xeon® processors!

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

**Native:** Using `-mmic`

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

**CAO:** Using `-offload-option`(example to Link MKL statically for both host and MIC)

```
icc -O3 sgemm.c -L$MKLROOT/lib/intel64 -offload-option, mic,ld,-
L$MKLROOT/lib/mic -Wl,-Bstatic, -lmkl_intel_lp64 - Wl,--start-group -lmkl_intel_thread -
lmkl_core -Wl,--end-group -Wl,-Bdynamic
```
Considerations of Using Intel® Math Kernel Library 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.

Minimize Data Transfer overhead when Offload

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

You can always run on host if offloading does not offer Better Performance
Where to Find Code Examples

$MKLROOT/examples/mic_ao/blasc/source

  sgemm.c  -- AO Example

$MKLROOT/examples/mic_offload/.../source

  sgemm.c  -- blasc
  complex_dft_1d.c  -- dftc
  sgeqrf.c, sgetrf.c, spotrf.c  -- Lapackc
  vdrnggaussian.c, vsrnggaussian.c  -- vslc
  etc etc
## Intel® Math Kernel Library Link Line Advisor

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Intel® product:</td>
<td>Intel(R) MKL 11.1</td>
</tr>
<tr>
<td>Select OS:</td>
<td>&lt;Select operating system&gt;</td>
</tr>
<tr>
<td>Select usage model of Intel® Xeon Phi™ Coprocessor:</td>
<td>&lt;Select usage model&gt;</td>
</tr>
<tr>
<td>Select compiler:</td>
<td>&lt;Select compiler&gt;</td>
</tr>
<tr>
<td>Select architecture:</td>
<td>&lt;Select architecture&gt;</td>
</tr>
<tr>
<td>Select dynamic or static linking:</td>
<td>&lt;Select linking&gt;</td>
</tr>
<tr>
<td>Select interface layer:</td>
<td>&lt;Select interface&gt;</td>
</tr>
<tr>
<td>Select sequential or multi-threaded layer:</td>
<td>&lt;Select threading&gt;</td>
</tr>
<tr>
<td>Select OpenMP library:</td>
<td>&lt;Select OpenMP&gt;</td>
</tr>
<tr>
<td>Select cluster library:</td>
<td>CDFT (BLACS required)</td>
</tr>
<tr>
<td></td>
<td>ScaLAPACK (BLACS required)</td>
</tr>
<tr>
<td></td>
<td>BLACS</td>
</tr>
<tr>
<td>Select MPI library:</td>
<td>&lt;Select MPI&gt;</td>
</tr>
<tr>
<td>Select the Fortran 95 interfaces:</td>
<td>BLAS95</td>
</tr>
<tr>
<td></td>
<td>LAPACK95</td>
</tr>
</tbody>
</table>
**Performance Charts on Intel® Xeon Phi™ coprocessors**

**HPL Linpack Benchmark on 16 Nodes with Intel® Xeon® Processors E5-2680 and Intel® Xeon Phi™ Coprocessors 7120P (N = 320k, P = 8, Q = 4)**

- **Pure Mode:** 1 CPU Each Node
  - 5.24 TFlops
  - 94.7% efficiency

- **Hybrid Mode:** 1 CPU and 1 Coprocessor Each Node
  - 18.56 TFlops
  - 74.7% efficiency

- **Hybrid Mode:** 1 CPU and 2 Coprocessors Each Node
  - 30.37 TFlops
  - 68.7% efficiency

Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 11.1, Intel® MPI 4.1.0.024, Intel® C++ Compiler 13.0, Intel® Manycore Platform Software Stack (MPSS) 2.1.6720-15; Hardware of cluster nodes: Intel® Xeon® Processor E5-2680, 2 Eight-Core CPUs (20MB LLC, 2.7GHz), 64GB of RAM; Intel® Xeon Phi™ Coprocessor 7120P, 61 cores (30.5MB total cache, 1.3GHz), 16GB GDDR5 Memory; Operating System: RHEL 6.1 GA x86_64; Benchmark Source: Intel Corporation. September 2013
Performance Charts Contd ...
Performance Charts Contd...

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 (MKL) 11.1, Intel® Manycore Platform Software Stack (MFSS) 2.1.6720-15; Hardware: Intel® Xeon® Processor E5-2697 v2, 2 Twelve-Core CPUs (30MB LLC, 2.7GHz), 32GB DDR3 RAM (1333MHz), 16GB 3D XPoint 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.


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 Tips

Problem size considerations:

- Large problems have more parallelism.
- But not too large (8GB memory on a coprocessor).
- FFT prefers power-of-2 sizes.

Data alignment consideration:

- 64-byte alignment for better vectorization.

OpenMP thread count and thread affinity:

- KMP_AFFINITY=balanced

Large (2MB) pages for memory allocation:

- Reduce TLB misses and memory allocation overhead.

MKL 11.1 Highlights

• Support for Intel® Xeon Phi™ coprocessors on Windows OS* hosts.
  - The same usage models of using MKL on Linux* hosts.

• Better installation experience:
  - A choice of components to install
  - Examples and tests are packaged as archives

• HPL support for heterogeneous clusters.

• CNR support for unaligned input data.

• Performance improvements across the board.
Better installation experience

- Introduced online Installer starting with the MKL 11.1

- Introduced Partial Installation Feature:

  By default, these components are NOT installed:
  - Cluster components (scaLAPACK, Cluster DFT)
  - Components needed by PGI* compilers (e.g. libmkl_pgi_thread.so)
  - Components needed by CVF (e.g. mkl_intel_s_dll.lib)
  - The SP2DP interface
  - Users may re-run the installer at a later time to install any of these components.
CNR support for unaligned input data

Before

- Memory alignment
  - Align memory — try Intel MKL memory allocation functions
  - 64-byte alignment for processors in the next few years

- Number of threads
  - Set the number of threads to a constant number
  - Use sequential libraries

- Deterministic task scheduling
  - Ensures that FP operations occur in order to ensure reproducible results

- Code path control
  - Maintains consistent code paths across processors
  - Will often mean lower performance on the latest processors

After

- Pre-requisite: Fixed number of threads
  - Set the number of threads to a constant number (MKL_NUM_THREADS)
  - Use sequential libraries

- Deterministic task scheduling
  - Ensures that FP operations occur in order to ensure reproducible results

- Code path control
  - Maintains consistent code paths across processors
  - Will often mean lower performance on the latest processors

• Data alignment is no longer a requirement for getting numerical reproducibility.
• But aligning input data is still a good idea for getting better performance.
## Intel® MKL 11.1 Packages

<table>
<thead>
<tr>
<th>Windows*</th>
<th>Linux*</th>
<th>Mac OS* X</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel® Parallel Studio XE Intel® C++ Studio XE Intel® Fortran Studio XE</td>
<td>Intel® Parallel Studio XE Intel® C++ Studio XE Intel® Fortran Studio XE</td>
<td>Intel® Composer XE</td>
</tr>
<tr>
<td>Intel® Composer XE</td>
<td>Intel® Composer XE</td>
<td>Intel® Composer XE</td>
</tr>
<tr>
<td>Intel® C++ Composer XE</td>
<td>Intel® C++ Composer XE</td>
<td>Intel® C++ Composer XE</td>
</tr>
<tr>
<td>Intel® Fortran Composer XE Intel® Visual Fortran Composer XE</td>
<td>Intel® Fortran Composer XE</td>
<td>Intel® Fortran Composer XE</td>
</tr>
<tr>
<td>Intel® Cluster Studio XE</td>
<td>Intel® Cluster Studio XE</td>
<td></td>
</tr>
<tr>
<td>Intel® MKL Standalone Product</td>
<td>Intel® MKL Standalone Product</td>
<td></td>
</tr>
</tbody>
</table>

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


http://software.intel.com/en-us/node/458836#2632E0AD-C8CF-427C-802B-52A06AC778F2
Online Resources

Articles, tips, case studies, hands-on lab:


Performance charts online:


The MIC developer community: http://www.intel.com/software/mic-developer

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY
ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS
DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR
IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES
RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY
PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

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.

Copyright © Intel Corporation. All rights reserved. Intel, the Intel logo, Xeon, Xeon Phi, Core,
VTune, and Cilk are trademarks of Intel Corporation in the U.S. and other countries.

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