Getting Reproducible Results with Intel® MKL
Why do results vary?

Root cause for variations in results

• Floating-point numbers → order of computation matters!

• Single precision example where \((a+b)+c \neq a+(b+c)\)

\[
\begin{align*}
2^{-63} + 1 + -1 &= 2^{-63} & \text{(infinitely precise result)} \\
(2^{-63} + 1) + -1 &\approx 0 & \text{(correct IEEE single precision result)} \\
2^{-63} + (1 + -1) &\approx 2^{-63} & \text{(correct IEEE single precision result)}
\end{align*}
\]

Order matters when doing floating point arithmetic.
Why are reproducible results important for users?

**Technical/legacy**
Software correctness is determined by comparison to previous ‘gold’ results.

**Debugging**
When developing and debugging, a higher degree of run-to-run stability is required to find potential problems.

**Legal**
Accreditation or approval of software might require exact reproduction of previously defined results.

**Customer perception**
Developers may understand the technical issues with reproducibility but still require reproducible results since end users or customers will be disconcerted by the inconsistencies.

Why does the order of operations change?

<table>
<thead>
<tr>
<th>Optimizations</th>
<th>Why does it change?</th>
</tr>
</thead>
<tbody>
<tr>
<td>instruction sets</td>
<td><strong>Code path</strong> affects grouping of data in registers</td>
</tr>
<tr>
<td>multiple cores / multiple processors</td>
<td>most functions are <strong>threaded</strong> to use as many cores as will give good scalability</td>
</tr>
<tr>
<td>Non-deterministic task scheduling</td>
<td>some algorithms use <strong>asynchronous task scheduling</strong> for optimal performance</td>
</tr>
</tbody>
</table>

Many optimizations require a change in order of operations.
How to get numerical reproducibility?
Conditional Numerical Reproducibility with Intel® MKL

MKL 11.0 (2012 September release)

- 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

MKL 11.1 (2013 September release)

- 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.
Consider $m=n=k=999$, $M_{\text{kernel\_unroll}}=8$

For CNR, pad the matrix chunks to be multiples of $M_{\text{kernel\_unroll}}$ such that the last thread executes the border code-paths
DGEMM Transa='N' and Transb='N' on Intel(R) Xeon(R) Processor E5-4650 (2.7 GHz "Sandy Bridge")

Geomean of Speed-up = 1.0

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
## CNR Code Path Controls

<table>
<thead>
<tr>
<th>Maximum Compatibility</th>
<th>For consistent results ...</th>
<th>Function Call</th>
<th>Environment Variable MKL_CBWR=</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>on Intel® or Intel®-compatible CPUs supporting SSE2 instructions or later</td>
<td>mkl_cbwr_set( ... )</td>
<td>MKL_CBWR_COMPATIBLE</td>
</tr>
<tr>
<td></td>
<td>on Intel® processors supporting SSE2 instructions or later</td>
<td></td>
<td>MKL_CBWR_SSE2</td>
</tr>
<tr>
<td></td>
<td>on Intel processors supporting SSE4.2 instructions or later</td>
<td></td>
<td>MKL_CBWR_SSE4_2</td>
</tr>
<tr>
<td></td>
<td>on Intel processors supporting Intel® AVX or later</td>
<td></td>
<td>MKL_CBWR_AVX</td>
</tr>
<tr>
<td></td>
<td>from run to run (but not processor-to-processor)</td>
<td></td>
<td>MKL_CBWR_AUTO</td>
</tr>
</tbody>
</table>

These are run time controls.
Balancing reproducibility and performance

CNR Impact on Performance of Intel® Optimized LINPACK Benchmark

- **CNR Off**: Maximum performance with CNB off
- **AUTO**: AUTO - Deterministic task scheduling
- **AVX**: AVX - Best performing code path on Sandy Bridge
- **SSE4_2**: SSE4_2 - Code path supported on both Sandy Bridge and Westmere
- **COMPATIBLE**: COMPATIBLE - Getting reproducible results on IA and IA-compatible processors

GFlops (Peak performance)

- **Sandy Bridge-EP (Intel® Xeon® E5-2690)**
- **Westmere-EP (Intel® Xeon® X5680)**

Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 11.0; Hardware: Intel® Xeon® Processor E5-2690, 2 Eight-Core CPUs (20MB LLC, 2.9GHz), 32GB of RA and Intel® Xeon® Processor X5680, 2 Six-Core CPUs (12MB LLC, 3.33GHz), 48GB of RAM; Operating System: RHEL 6 GA x86_64; Benchmark Source: Intel Corporation.

Test environment: 64-bit executable, Matrix 40k x 40k, OMP_NUM_THREADS=12

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, refer to [www.intel.com/performance/resources/benchmark_limitations.htm](http://www.intel.com/performance/resources/benchmark_limitations.htm).

* 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
Why “Conditional”?

- Reproducibility is currently available under certain conditions:
  - Within single operating systems / architecture
    - Reproducibility only applies within the blue boxes, not between them...
  - Reproducibility only with the same number of threads
  - Reproducibility on supported servers and workstations
    - No support yet for Intel® Xeon Phi™ coprocessors
  - Within a particular version of Intel® MKL
    - Results in version 11.1 update 1 may differ from results in version 11.1
  - Reproducibility controls in Intel MKL only affect Intel MKL functions
What about floating point computations outside Intel® MKL?

Intel compilers provide switches to improve floating point results reproducibility:

• Run-to-run
  - Compiler options “-fp-model precise -fp-model source”

• Processor-to-processor
  - Compiler option “-fimf-arch-consistency=true”

These are compile time controls. You have to build your code using these switches.
How to do deterministic reductions?

For deterministic parallel reduction in OpenMP:
- KMP_DETERMINISTIC_REDUCTION=1
- Only available in the Intel implementation of OpenMP

For deterministic parallel reduction in Intel® TBB:
- Function “parallel_deterministic_reduce()”
Negligible impacts on performance

- Intel MKL’s LAPACK had two operations that were not “CNR-friendly”
  - Parallel reduces that were nondeterministic
  - Variable block sizes
- Removing these did not seem to have a significant impact on performance
Further resources

• Knowledgebase articles
  - Intel MKL conditional numerical reproducibility
  - Consistency of FP results using the Intel compilers

• Support
  - Intel MKL user forum
  - Intel Premier support