BUILD FAST AND CROSS-PLATFORM APPLICATIONS WITH INTEL® PERFORMANCE LIBRARIES

Chao Yu, Technical Consulting Engineer,

Intel® IPP and MKL Team
Agenda

• Intel® IPP and Intel® MKL Benefits
• What’s New in Intel® MKL 11.3
• What’s New in Intel® IPP 9.0
  – New Features and Changes
  – Tips to Move Intel® IPP 9.0
Optimized Blocks for Image, Signal and Data Processing
Intel® Integrated Performance Primitives

**Image Processing**
- Geometry transformations
- Linear and non-linear filtering
- Linear transforms
- Statistics and analysis
- Color models

**Computer Vision**
- Feature detection
- Objects tracking
- Pyramids functions
- Segmentation, enhancement
- Camera functions
- And more

**Signal Processing**
- Transforms
- Convolution, Cross-Correlation
- Signal generation
- Digital filtering
- Statistical

**Data Compression**
- LZSS
- LZ77(ZLIB)
- LZO
- Bzip2

**Cryptography**
- Symmetric cryptography
- Hash functions
- Data authentication
- Public key

**String Processing**
- String Functions: Find, Insert, Remove, Compare, etc.
- Regular expression
Intel® IPP Benefits

Enhances Developer Productivity

- Optimized image, signal and data processing routines
- Thread-safe functions

Industry Leading Performance

- Instruction set-level optimizations (SIMD)
- Efficient parallelism on multicore platforms

Support for Latest Processor Architectures

- Optimized for current multi-core processors
- Applications benefit seamlessly

Cross Platform and Operating System Support

**Multi OS:**
- Windows*
- Linux*
- OSX*
- Android*
- VxWorks*

**Multi Platform:**
- Mobile and Embedded (Intel® Quark, Intel® Atom™)
- Tablet (Intel® Atom™, Intel® Core™)
- Ultrabook/PC (Intel® Core™)
- Servers and Workstations (Intel® Xeon® and Intel® Xeon® Phi™)
Optimized Mathematical Building Blocks
Intel® Math Kernel Library

Linear Algebra
- BLAS
- LAPACK
- Sparse Solvers
  - Iterative
  - Pardiso*
- ScaLAPACK

Fast Fourier Transforms
- Multidimensional
- FFTW interfaces
- Cluster FFT

Summary Statistics
- Kurtosis
- Variation coefficient
- Order statistics
- Min/max
- Variance-covariance

Vector Math
- Trigonometric
- Hyperbolic
- Exponential, Log
- Power / Root

Vector RNGs
- Congruential
- Wichmann-Hill
- Mersenne Twister
- Sobol
- Neiderreiter
- Non-deterministic

And More
- Splines
- Interpolation
- Trust Region
- Fast Poisson Solver

*Other names and brands may be claimed as the property of others.
Intel® Math Kernel Library Benefits

Mathematical problems arise in many scientific disciplines


Intel MKL provides scientific programmers and domain scientists

- Interfaces to de-facto standard APIs (BLAS,LAPACK)
- Support for Linux*, Windows* and OS* X target systems
- The ability to extract great parallel performance with minimal effort

\[-\frac{\partial u^2}{\partial x^2} - \frac{\partial u^2}{\partial y^2} + q u = f(x, y)\]

Intel® MKL can help solve your computational challenges
Agenda

• Intel® IPP and Intel® MKL Benefits

• What's New in Intel® MKL 11.3

• What’s New in Intel® IPP 9.0
  – New Features and Changes
  – Tips to Move Intel® IPP 9.0
New Features
Intel® Math Kernel Library (Intel® MKL) 11.3

- Optimized for the latest Intel® Xeon® processors and for Intel® Xeon Phi™ Coprocessor
- Sparse BLAS inspector-executor APIs
  - 2-stage APIs for Sparse BLAS (analyze and execute)
- Batch GEMM functions
  - Improved the performance of multiple, simultaneous matrix multiply operations
- New counter-based pseudorandom number generator
  - ARS-5 (based on the Intel AES-NI instruction set) and Philox4x32-10
- Intel® MKL PARDISO scalability
  - Improved Intel® MKL PARDISO and Cluster Sparse Solver scalability on Intel Xeon Phi coprocessors
- Cluster components extension
  - New MPI wrappers that allow users to build custom BLACS library for most MPIs
  - Cluster components support on OS X*
- New Intel® Threaded Building Blocks (Intel® TBB) threading layer
Two-stage APIs for Sparse BLAS functions

- \texttt{mkl\_sparse\_d\_create\_csr(\&A, SPARSE\_INDEX\_BASE\_ZERO, rows, cols, rowsStart, rowsEnd, colIndx, values)}

- \texttt{mkl\_sparse\_set\_mv\_hint(\&A, SPARSE\_OPERATION\_NON\_TRANSPOSE, SPARSE\_MATRIX\_TYPE\_SYMMETRIC, n\_iter);}

- \texttt{mkl\_sparse\_set\_memory\_hint(\&A, SPARSE\_MEMORY\_AGGRESSIVE);}  
- \texttt{mkl\_sparse\_optimize(\&A);}  

\begin{verbatim}
for (int i=0; i<n_iter; i++) {
    mkl\_sparse\_d\_mv(SPARSE\_OPERATION\_NON\_TRANSPOSE, alpha, \&A, SPARSE\_MATRIX\_TYPE\_SYMMETRIC, x, beta, y);
    . . . . . .
}
\end{verbatim}

- \texttt{mkl\_sparse\_destroy(\&A);}  

- **Inspect stage:** inspects the matrix sparse pattern and applies matrix structure changes
  - Computational kernel for portrait
  - Balancing strategy for parallel execution

- **Execution stage:** reuse the information to improve performance.
Intel® MKL Sparse BLAS Performance Using Inspector-Executor API on Intel® Xeon® Processor E5-2699 v3

Inspector-Executor API Boosts DCSRMV Performance

Inspector-executor API vs NIST(standard) API Intel® Xeon® Processor E5-2699 v3 (36 threads)

Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 11.3; Hardware: Intel® Xeon® Processor E5-2699 v3, 2 Eighteen-core CPUs (45MB LLC, 2.3GHz), 64GB of RAM; Operating System: RHEL 6.4 GA x86_64; Benchmark Source: Intel Corporation. Matrices from the University of Florida Sparse Matrix Collection (http://www.cise.ufl.edu/research/sparse/matrices/index.html).

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

Optimization Notice: Intel® compiler 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.
Batch Matrix-Matrix Multiplication

- Execute independent general matrix-matrix multiplication (GEMM) operations simultaneously with a single function call
  - Support all precisions xGEMM and xGEMM3M functions
  - Handle varying matrix sizes with a single function call
  - Better utilize multi-core processors for small sizes

\[
C_i = \alpha_i A_i \cdot B_i + \beta_i C_i
\]

```
idx = 0
for i = 0..group_count - 1
    alpha and beta in alpha_array[i] and beta_array[i]
    for j = 0..group_size[i] - 1
        A, B, and C matrix in a_array[idx], b_array[idx], and c_array[idx]
        C := alpha*op(A)*op(B) + beta*C,
        idx = idx + 1
    end for
end for
```
Faster 10,000 Small Matrix Multiplication Instances using Intel® MKL DGEMM_BATCH

DGEMM_BATCH vs DGEMM, Intel® Xeon® Processor E5-2699v3

Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 11.3; Hardware: Intel® Xeon® Processor E5-2699v3, 2 Eighteen-core CPUs (45MB LLC, 2.3GHz), 64GB of RAM; Operating System: CentOS 7.1 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

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 .
Intel® TBB Threading Layer

• Intel MKL threading with TBB

• A new threading layer:
  – Level3 BLAS
  – LAPACK function: LU factorization and solving
  – Poisson

• Linkage:
  -lmkl_intel_lp64 -lmkl_tbb_thread -lmkl_core -ltbb -lstdc++

• Dynamically selecting the threading layer
  • Environment variable:
    MKL_THREADING_LAYER =TBB
  • API:
    mkl_set_threading_layer (MKL_THREADING_TB )
Agenda

• Intel® IPP and Intel® MKL Benefits
• What's New in Intel® MKL 11.3

• What’s New in Intel® IPP 9.0
  – New Features and Changes
  – Tips to Move Intel® IPP 9.0
New Features
Intel® Integrated Performance Primitives (Intel® IPP) 9.0

• Additional optimization for Intel® Quark™, Intel® Atom™, and the processors with Intel® AVX-512 instructions support
  – Intel® Quark™: cryptography, data compression optimization
  – Intel® Atom™: computer vision, image processing optimization
  – Intel® AVX-512: computer vision, image processing optimization
• New APIs to support external threading

• Improved CPU dispatcher
  – Auto-initialization. No need for the CPU initialization call in static libraries.
  – Code dispatching based on CPU features

• Optimized cryptography functions to support SM2/SM3/SM4 algorithm

• Custom dynamic library building tool

• New APIs to support external memory allocation
New CPU Dispatching Code

- Intel IPP functions are optimized for specific processors.
- A single function has many versions, each one optimized to run on a specific processor.
- New CPU dispatching code K0 for processors with Intel® Advanced Vector Extensions 512 (Intel® AVX-512).

### Platform Identifier Optimization

<table>
<thead>
<tr>
<th>Platform</th>
<th>Identifier</th>
<th>Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>IA-32 Intel® Architecture</td>
<td>px</td>
<td>C-optimized for all IA-32 processors</td>
</tr>
<tr>
<td></td>
<td>w7</td>
<td>Optimized for processors with Intel SSE2</td>
</tr>
<tr>
<td></td>
<td>p8</td>
<td>Optimized for processors with Intel® Streaming SIMD Extensions 4.1 (Intel SSE4.1)</td>
</tr>
<tr>
<td></td>
<td>h9</td>
<td>Optimized for processors with Intel® Advanced Vector Extensions 2 (Intel® AVX2)</td>
</tr>
<tr>
<td>Intel® 64 architecture</td>
<td>mx</td>
<td>C-optimized for processors with Intel® 64 instructions set architecture</td>
</tr>
<tr>
<td></td>
<td>n8</td>
<td>Optimized for the Intel® Atom™ processor</td>
</tr>
<tr>
<td></td>
<td>y8</td>
<td>Optimized for 64-bit applications on processors with Intel® Streaming SIMD Extensions 4.1 (Intel SSE4.1)</td>
</tr>
<tr>
<td></td>
<td>e9</td>
<td>Optimized for processors that support Intel® Advanced Vector Extensions instruction set</td>
</tr>
<tr>
<td></td>
<td>l9</td>
<td>Optimized for processors with Intel® Advanced Vector Extensions 2 (Intel® AVX2)</td>
</tr>
<tr>
<td></td>
<td>k0</td>
<td>Optimized for processors with Intel® Advanced Vector Extensions 512 (Intel® AVX-512)</td>
</tr>
</tbody>
</table>

Some examples:

- `ippsCopy_8u(...)`
- `mx_ippsCopy_8u(...)`
- `l9_ippsCopy_8u(...)`
- `k0_ippsCopy_8u(...)`
External Memory Allocation

- Reduce the memory allocation for different Intel IPP function calls
  - A shared memory buffer can be used for different APIs

```
<...,>
ippAAInitAlloc (... , pStateA );
ippBBInitAlloc (... , pStateB);
ippCCInitAlloc (... , pStateC );
<...,>
ippAAProcess (... , pStateA );
ippBBProcess (... , pStateB );
ippCCProcess (... , pStateC );
<...,>
ippAAFree( pStateA );
ippBBFree( pStateB );
ippCCFree( pStateC );
<...,>
```

```
ippAAGetSize(...,&specSizeA, &bufSizeA);
ippBBGetSize(...,&specSizeB, &bufSizeB);
ippCCGetSize(...,&specSizeC, &bufSizeC);
maxSize=MAX(bufSizeA, bufSizeB, bufSizeC);
pBuffer = malloc( maxSize);
<...,>
ippAAInit (... , pSpecA );
ippAAProcess (... , pSpecA, pBuffer );
ippBBInit (... , pSpecB);
ippBBProcess (... , pSpecB , pBuffer );
ippCCInit (... , pSpecC );
ippCCProcess (... , pSpecC, pBuffer );
<...,>
```

Internal Memory Allocation APIs (Deprecated)

```
A Memory
Spec (const) part
AA Buffer part

B Memory
Spec (const) part
BB Buffer part

C Memory
Spec (const) part
CC Buffer part
```

External Memory Allocation APIs (New)

```
Memory
SpecA part
SpecB part
SpecC part
Shared memory buffer for A, B, C
```
External Memory Allocation (Cont.)

- Intel IPP 9.0 removes the internal memory allocations in single-threaded libraries
- All memory allocations should be done at the application level
  - The legacy InitAlloc/Free functions are removed from Intel IPP 9.0
  - Use the substitution GetSize/Init functions
  - Additional GetSize/Init functions added to support external memory allocation
  - New redesigned geometry functionality to remove the internal memory allocation

Optimization Notice
Copyright © 2015, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
Intel IPP Threading

• Intel® IPP library primitives are “thread-safe”

• Intel IPP internally threaded libraries are available as optional installation

• The external threading is recommended, which is more effective than the internal threading
External Threading Support

- The external threading is recommended, which is more effective than the internal threading.
- Many of Intel IPP APIs were updated to enable external threading support.
- To use the external threading, users need to handle the “border” data for some of the Intel IPP APIs.

```c
// The first threading:
// No history delay data: NULL
// The last threading:
// Use previous data for the border: pSrc-dlyLineLen
// Other threadings:
// Use previous data for the border: pSrc-dlyLineLen
// No need to output delay data: NULL

dlyLineLen = tapLen - 1;  // dlyLineLen is used to keep old data.
ippsFIRSRGetSize_32f(… & specSize, &bufSize);
ippsFIRSRInit_32f(filterTaps,tapLen,…,pSpec);
len = LEN/NUNTHREADS;  // simplified code, not consider for tail data
for(iThread=0;iThread<NUNTHREADS;iThread++)  // it means parallel for
{
    Ipp32f* pSrc = input+iThread*len;
    Ipp32f* pDst = output+iThread*len;

    if( iThread == 0)
        ippsFIRSR_32f( pSrc, pDst, len, pSpec, NULL, NULL, buffer);
    else if (iThread == NUMTHREADS - 1)
        ippsFIRSR_32f(pSrc, pDst, len, pSpec, pSrc - dlyLineLen, OutDlyLine, buffer);
    else
        ippsFIRSR_32f(pSrc,pDst, len, pSpec, pSrc - dlyLineLen, NULL, buffer);
}
```
New Custom Dynamic Library Building Tool

- A tool that can create the custom .dll or .so from Intel IPP static libraries
  - Link with Intel IPP dynamically, but no need to redistribute Intel IPP dynamic libraries.
  - Create the dynamic library containing the selected functions only.
  - Significantly reduce the size of dynamic libraries distributed with the applications.
Agenda

- Intel® IPP and Intel® MKL Benefits
- What's New in Intel® MKL 11.3
- What’s New in Intel® IPP 9.0
  - New Features and Changes
  - Tips to Move Intel® IPP 9.0
# Find Performance Benefit with the New Release

- **Performance tests tools** to measure specific functions used in your applications:
  ```
  \texttt{\ipp\tools\ia32(intel64)\perfsys}
  ```

- Run the tools at Intel IPP 9.0 and old version to learn performance difference:
  ```
  > ps_ipps.exe -f \ippiWarpAffineLinear\_32f\_C1R -B -r performance.csv
  ```

<table>
<thead>
<tr>
<th>function</th>
<th>Parm1</th>
<th>Parm2</th>
<th>Parm3</th>
<th>Parm4</th>
<th>Parm5</th>
<th>Parm6</th>
<th>Parm7</th>
<th>Commer Clocks</th>
<th>per</th>
</tr>
</thead>
<tbody>
<tr>
<td>\ippiWarpAffineLinear</td>
<td>32f</td>
<td>C1R</td>
<td>Resize + Shift</td>
<td>128x128</td>
<td>64x64</td>
<td>(2:1)</td>
<td>ippBorderRepl</td>
<td>nLps=4</td>
<td>7.56 oxch</td>
</tr>
<tr>
<td>\ippiWarpAffineLinear</td>
<td>32f</td>
<td>C1R</td>
<td>Resize + Shift</td>
<td>192x192</td>
<td>64x64</td>
<td>(3:1)</td>
<td>ippBorderRepl</td>
<td>nLps=4</td>
<td>7.74 oxch</td>
</tr>
<tr>
<td>\ippiWarpAffineLinear</td>
<td>32f</td>
<td>C1R</td>
<td>Resize + Shift</td>
<td>256x256</td>
<td>64x64</td>
<td>(4:1)</td>
<td>ippBorderRepl</td>
<td>nLps=4</td>
<td>7.82 oxch</td>
</tr>
<tr>
<td>\ippiWarpAffineLinear</td>
<td>32f</td>
<td>C1R</td>
<td>Resize + Shift</td>
<td>320x320</td>
<td>64x64</td>
<td>(5:1)</td>
<td>ippBorderRepl</td>
<td>nLps=4</td>
<td>8.27 oxch</td>
</tr>
<tr>
<td>\ippiWarpAffineLinear</td>
<td>32f</td>
<td>C1R</td>
<td>Resize + Shift</td>
<td>96x96</td>
<td>64x64</td>
<td>(3:2)</td>
<td>ippBorderRepl</td>
<td>nLps=4</td>
<td>7.61 oxch</td>
</tr>
<tr>
<td>\ippiWarpAffineLinear</td>
<td>32f</td>
<td>C1R</td>
<td>Resize + Shift</td>
<td>100x100</td>
<td>64x64</td>
<td>(5:2)</td>
<td>ippBorderRepl</td>
<td>nLps=4</td>
<td>7.65 oxch</td>
</tr>
<tr>
<td>\ippiWarpAffineLinear</td>
<td>32f</td>
<td>C1R</td>
<td>Resize + Shift</td>
<td>128x128</td>
<td>64x64</td>
<td>(2:1)</td>
<td>ippBorderRepl</td>
<td>nLps=4</td>
<td>6.04 oxch</td>
</tr>
<tr>
<td>\ippiWarpAffineLinear</td>
<td>32f</td>
<td>C1R</td>
<td>Resize + Shift</td>
<td>192x192</td>
<td>64x64</td>
<td>(3:1)</td>
<td>ippBorderRepl</td>
<td>nLps=4</td>
<td>6.19 oxch</td>
</tr>
<tr>
<td>\ippiWarpAffineLinear</td>
<td>32f</td>
<td>C1R</td>
<td>Resize + Shift</td>
<td>256x256</td>
<td>64x64</td>
<td>(4:1)</td>
<td>ippBorderRepl</td>
<td>nLps=4</td>
<td>6.15 oxch</td>
</tr>
<tr>
<td>\ippiWarpAffineLinear</td>
<td>32f</td>
<td>C1R</td>
<td>Resize + Shift</td>
<td>320x320</td>
<td>64x64</td>
<td>(5:1)</td>
<td>ippBorderRepl</td>
<td>nLps=4</td>
<td>6.57 oxch</td>
</tr>
<tr>
<td>\ippiWarpAffineLinear</td>
<td>32f</td>
<td>C1R</td>
<td>Resize + Shift</td>
<td>96x96</td>
<td>64x64</td>
<td>(3:2)</td>
<td>ippBorderRepl</td>
<td>nLps=4</td>
<td>6.1 oxch</td>
</tr>
<tr>
<td>\ippiWarpAffineLinear</td>
<td>32f</td>
<td>C1R</td>
<td>Resize + Shift</td>
<td>100x100</td>
<td>64x64</td>
<td>(5:2)</td>
<td>ippBorderRepl</td>
<td>nLps=4</td>
<td>6.68 oxch</td>
</tr>
</tbody>
</table>

*Performance is measured by CPU clocks per elements. The smaller, the better*
Intel® IPP API changes

- **Internal memory allocation functions**
  
  $ippxxxInitAlloc \rightarrow ippxxxGetSize + ippsMalloc + ippxxxInit$

- **Redesigned and the removed APIs**: find the substitution APIs in the manual

  Manual Volume 1, Appendix B, Removed Functions for Signal Processing
  Manual Volume 2, Appendix C, Removed Functions for Image and Video Processing

<table>
<thead>
<tr>
<th>Removed from 9.0</th>
<th>Substitution or Workaround</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipps10Log10_32s_ISFs</td>
<td>Use ippsConvert_32s32f+ippsLog10_32f_A24 (ippVM)</td>
</tr>
<tr>
<td>ipps10Log10_32s_SFs</td>
<td>Use ippsConvert_32s32f+ippsLog10_32f_A24 (ippVM)</td>
</tr>
<tr>
<td>ippsALawToLin_S16s</td>
<td>Use any open-source g711 implementation</td>
</tr>
<tr>
<td>ippsALawToLin_Su32f</td>
<td>Use any open-source g711 implementation</td>
</tr>
</tbody>
</table>
Intel® IPP Domain Changes

- Some Intel® IPP domains are deprecated
  Audio and Video Coding/ Speech Coding/ Image Compression/ Data Integrity/Generated Transforms/Small Matrices /Realistic Rendering

- The alternatives for the deprecated domains
  - Small Matrices (ippMX): Intel® MKL
  - Video Coding (ippVC), Audio Coding(ippAC) and JPEG Coding(ippJP): Intel® Media SDK
  - Find more on the alternative functions article

- Using Intel IPP Legacy Libraries
  - Providing the separated libraries containing the removed deprecated Intel IPP functions
  - Fully independent from the Intel IPP main package
  - The legacy domain and the new versions of Intel IPP can be linked together
  - The legacy optimization are available. No further optimization for future release
For More Information...

• Where to find?
  - Available as part of Intel fully product suites:
    • Intel® Parallel Studio XE Suites: Intel IPP, Intel MKL
    • Intel® System Studio Suite: Intel IPP, Intel MKL
    • Intel® Integrated Native Developer Experience: Intel IPP
  - Free tools for academic researcher/student/educator/open source contributor
  - Free community licenses for Intel® Performance Libraries

• More product information
  - Intel IPP: http://www.intel.com/software/products/ipp

Legal Disclaimer & Optimization Notice

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 © 2014, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

<table>
<thead>
<tr>
<th>Optimization Notice</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>

Notice revision #20110804