CODE THAT OUTPERFORMS
INTEL® PARALLEL STUDIO XE 2019
PERFORMANCE LIBRARIES

Accelerate Parallel Code,
Transform Enterprise to Cloud & HPC to AI Applications
SPEED UP NUMERICAL APPLICATION PERFORMANCE WITH INTEL® MATH KERNEL LIBRARY (INTEL® MKL) 2019

Fastest and most used math library for Intel®-based systems¹

¹ Data from Evans Data Software Developer surveys, 2011-2016
Faster, Scalable Code with Intel® Math Kernel Library

- Speeds computations for scientific, engineering, financial and machine learning applications by providing highly optimized, threaded, and vectorized math functions
- Provides key functionality for dense and sparse linear algebra (BLAS, LAPACK, PARDISO), FFTs, vector math, summary statistics, deep learning, splines and more
- Dispatches optimized code for each processor automatically without the need to branch code
- Optimized for single core vectorization and cache utilization
- Automatic parallelism for multi-core and many-core
- Scales from core to clusters
- Available at no cost and royalty free
- Great performance with minimal effort!

Available as standalone or as a part of Intel® Parallel Studio XE and Intel® System Studio

Operating System: Windows*, Linux*, MacOS1*
What’s Inside Intel® MKL

LINEAR ALGEBRA
- BLAS
- LAPACK
- ScaLAPACK
- Sparse BLAS
- Iterative sparse solvers
- PARDISO
- Cluster Sparse Solver

FFT
- Multidimensional
- FFTW interfaces
- Cluster FFT

VECTOR RNGS
- Congruential
- Wichmann-Hill
- Mersenne Twister
- Sobol
- Neiderreiter
- Non-deterministic

SUMMARY STATISTICS
- Kurtosis
- Variation coefficient
- Order statistics
- Min/max
- Variance-covariance

VECTOR MATH
- Trigonometric
- Hyperbolic
- Exponential
- Log
- Power
- Root

AND MORE
- Splines
- Interpolation
- Trust Region
- Fast Poisson Solver

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
## Automatic Dispatching to Tuned ISA-specific Code Paths

More cores → More Threads → Wider vectors

<table>
<thead>
<tr>
<th></th>
<th>Intel® Xeon® Processor 64-bit</th>
<th>Intel® Xeon® Processor 5100 series</th>
<th>Intel® Xeon® Processor 5500 series</th>
<th>Intel® Xeon® Processor 5600 series</th>
<th>Intel® Xeon® Processor E5-2600 v2 series</th>
<th>Intel® Xeon® Processor E5-2600 v3 series v4 series</th>
<th>Intel® Xeon® Scalable Processor¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up to Core(s)</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>12</td>
<td>18-22</td>
<td>28</td>
</tr>
<tr>
<td>Up to Threads</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>12</td>
<td>24</td>
<td>36-44</td>
<td>56</td>
</tr>
<tr>
<td>SIMD Width</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>256</td>
<td>256</td>
<td>512</td>
</tr>
</tbody>
</table>

1. Product specification for launched and shipped products available on ark.intel.com.

---

**Optimization Notice**

Copyright © 2019, Intel Corporation. All rights reserved.

*Other names and brands may be claimed as the property of others.
What’s New for Intel® MKL 2019?

Just-In-Time Fast Small Matrix Multiplication

• Improved speed of S/DGEMM for Intel® AVX2 and Intel® AVX-512 with JIT capabilities

Sparse QR Solvers

• Solve sparse linear systems, sparse linear least squares problems, eigenvalue problems, rank and null-space determination, and others

Generate Random Numbers for Multinomial Experiments

• Highly optimized multinomial random number generator for finance, geological and biological applications
Performance Benefits for the latest Intel Architectures

DGEMM, SGEMM Optimized by Intel® Math Kernel Library 2019 Gold for Intel® Xeon® Platinum Processor

The benchmark results reported above may need to be revised as additional testing is conducted. The results depend on the specific platform configurations and workloads utilized in the testing, and may not be applicable to any particular user's components, computer system or workloads. The results are not necessarily representative of other benchmarks and other benchmark results may show greater or lesser impact from mitigations.

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 complete information visit www.intel.com/benchmarks.

Configuration: Intel® Xeon® Platinum 8180 H0 205W 2x28@2.5GHz 192GB DDR4-2666

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® MKL 11.0 - 2018 Noteworthy Enhancements

- Conditional Numerical Reproducibility (CNR)
- Intel® Threading Building Blocks (TBB) Composability
- Intel® Optimized High Performance Conjugate Gradient (HPCD) Benchmark
- Small GEMM Enhancements (Direct Call) and Batch
- Compact GEMM and LAPACK Support
- Sparse BLAS Inspector-Executor API
- Extended Cluster Support (MPI wrappers and macOS*)
- Parallel Direct Sparse Solver for Clusters
- Extended Eigensolvers
Intel® MKL BLAS (Basic Linear Algebra Subprograms)

<table>
<thead>
<tr>
<th>De-facto Standard APIs since the 1980s</th>
<th></th>
</tr>
</thead>
</table>
| **100s of Basic Linear Algebra Functions** | **Level 1** – vector vector operations, $O(N)$  
**Level 2** – matrix vector operations, $O(N^2)$  
**Level 3** – matrix matrix operations, $O(N^3)$  |
| **Precisions Available** | Real – Single and Double  
Complex - Single and Double  |
<p>| <strong>BLAS-like Extensions</strong> | Direct Call, Batched, Packed and Compact  |
| <strong>Reference Implementation</strong> | <a href="http://netlib.org/blas/">http://netlib.org/blas/</a>  |</p>
<table>
<thead>
<tr>
<th><strong>Intel® MKL LAPACK (Linear Algebra PACKage)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>De-facto Standard APIs since the 1990s</strong></td>
</tr>
<tr>
<td>1000s of Linear Algebra Functions</td>
</tr>
<tr>
<td>Matrix factorizations - LU, Cholesky, QR</td>
</tr>
<tr>
<td>Solving systems of linear equations</td>
</tr>
<tr>
<td>Condition number estimates</td>
</tr>
<tr>
<td>Symmetric and non-symmetric eigenvalue problems</td>
</tr>
<tr>
<td>Singular value decomposition</td>
</tr>
<tr>
<td>and many more ...</td>
</tr>
<tr>
<td>Precisions Available</td>
</tr>
<tr>
<td>Real – Single and Double,</td>
</tr>
<tr>
<td>Complex – Single and Double</td>
</tr>
<tr>
<td>Reference Implementation</td>
</tr>
<tr>
<td><a href="http://netlib.org/lapack/">http://netlib.org/lapack/</a></td>
</tr>
</tbody>
</table>
### Intel® MKL Fast Fourier Transforms (FFTs)

#### FFTW Interfaces
- Support: C, C++, and FORTRAN source code wrappers provided for FFTW2 and FFTW3. FFTW3 wrappers are already built into the library.

#### Cluster FFT
- Perform Fast Fourier Transforms on a cluster
- Interface similar to DFTI
- Multiple MPIs supported

#### Parallelization
- Thread safe with automatic thread selection

#### Storage Formats
- Multiple storage formats such as CCS, PACK, and Perm

#### Batch support
- Perform multiple transforms in a single call

#### Additional Features
- Perform FFTs on partial images
- Padding added for better performance
- Transform combined with transposition
- Mixed-language usage supported
# Intel® MKL Vector Math

## Example:

\[ y(i) = e^{x(i)} \text{ for } i = 1 \text{ to } n \]

## Broad Function Support

- Basic Operations – add, sub, mult, div, sqrt
- Trigonometric – sin, cos, tan, asin, acos, atan
- Exponential – exp, pow, log, log10, log2
- Hyperbolic – sinh, cosh, tanh
- Rounding – ceil, floor, round
- And many more

## Precisions Available

- Real – Single and Double
- Complex - Single and Double

## Accuracy Modes

- High - almost correctly rounded
- Low - last 2 bits in error
- Enhanced Performance - 1/2 the bits correct
## Intel® MKL Vector Statistics

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Random Number Generators (RNGs)</strong></td>
<td>Pseudorandom, quasi-random and non-deterministic random number generators with continuous and discrete distribution</td>
</tr>
<tr>
<td><strong>Summary Statistics</strong></td>
<td>Parallelized algorithms to compute basic statistical estimates for single and double precision multi-dimensional datasets</td>
</tr>
<tr>
<td><strong>Convolution and Correlation</strong></td>
<td>Linear convolution and correlation transforms for single and double precision real and complex data</td>
</tr>
</tbody>
</table>
## Intel® MKL Sparse Solvers

| **PARDISO - Parallel Direct Sparse Solver** | Factor and solve $Ax = b$ using a parallel shared memory $LU$, $LDL$, or $LL^T$ factorization  
Supports a wide variety of matrix types including real, complex, symmetric, indefinite, ...  
Includes out-of-core support for very large matrix sizes |
| **Parallel Direct Sparse Solver for Clusters** | Factor and solve $Ax = b$ using a parallel distributed memory $LU$, $LDL$, or $LL^T$ factorization  
Supports a wide variety of matrix types (real, complex, symmetric, indefinite, ... )  
Supports $A$ stored in 3-array CSR3 or BCSR3 formats |
| **DSS – Simplified PARDISO Interface** | An alternative, simplified interface to PARDISO |
| **ISS – Iterative Sparse Solvers** | Conjugate Gradient (CG) solver for symmetric positive definite systems  
Generalized Minimal Residual (GMRes) for non-symmetric indefinite systems  
Rely on Reverse Communication Interface (RCI) for matrix vector multiply |
<table>
<thead>
<tr>
<th>Intel® MKL General Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparse BLAS</td>
</tr>
<tr>
<td>Data Fitting</td>
</tr>
<tr>
<td>Partial Differential Equations</td>
</tr>
<tr>
<td>Optimization</td>
</tr>
</tbody>
</table>
| Service Functions           | Threading controls  
                               | Memory management  
                               | Numerical reproducibility |
## Intel® MKL Summary

| Boosts application performance with minimal effort | feature set is robust and growing  
|                                                   | provides scaling from the core, to multicore, to manycore, and to clusters  
|                                                   | automatic dispatching matches the executed code to the underlying processor  
|                                                   | future processor optimizations included well before processors ship  

| Showcases the world’s fastest supercomputers   | Intel® Distribution for LINPACK* Benchmark  
|                                               | Intel® Optimized High Performance Conjugate Gradient Benchmark  

1[http://www.top500.org](http://www.top500.org)
# Intel® MKL Resources

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
HIGHLY OPTIMIZED IMAGE, SIGNAL AND DATA PROCESSING FUNCTIONS WITH INTEL® INTEGRATED PERFORMANCE PRIMITIVES 2019

High Performance, easy to use and production ready API’s

September 2019
Part of Intel® Parallel Studio XE and Intel® System Studio
<<Or available individually as applicable>>
Intel® Integrated Performance Primitives 2019

Highly Optimized Image, Signal & Data Processing Functions

Intel® Integrated Performance Primitives provides developers with ready-to-use, processor optimized functions to accelerate *Image, Signal, Data Processing & Cryptography computation tasks*

- Multi-core, multi-OS and multi-platform ready, computationally intensive and highly optimized functions
- Plug in and use APIs to quickly improve application performance
- Reduced cost and time-to-market on software development and maintenance
- Access Priority Support, which connects you direct to Intel engineers for technical questions (paid versions only)

**What's New in 2019 version**

- Open source distribution of Intel® Integrated Performance Primitives Cryptography Library
- Added Threading Layer with OpenMP and TBB tool support for various image processing functions
- Added new functions to support ZFP floating-point data compression
- Improved LZ4 compression and decompression performance on high entropy data
- New color conversion functions for converting RBG images to CIE Lab color models, and vice versa
- Extended optimization for Intel® AVX-512 set and Intel® AVX2 instruction set

Learn More: software.intel.com

**Optimization Notice**

Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

**Certain technical specifications and select processors/skus apply. See product site for more details.**

Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.
Challenges faced by developers

**Performance optimization** is a never ending task.

Completing key processing tasks within designated **time constraints** is a critical issue.

**Hand optimizing code** for one platform makes code performance worse on another platform.

With manual optimization, code becomes more **complex and difficult** to maintain.

Code should run fast as possible **without spending extra effort**.
Intel® IPP Your Building Blocks for Image, Signal & Data Processing Applications

**What is Intel® IPP?**
Intel IPP provides developers with ready-to-use, processor-optimized functions to accelerate *Image & Signal processing, Data Compression & Cryptography computation tasks.*

**Why should you use Intel® IPP?**
- High Performance
- Easy to use API’s
- Faster Time To Market (TTM)
- Production Ready
- Cross-platform API

**How to get Intel® IPP?**
- Intel Parallel Studio XE
- Intel System Studio
- Free Tools Program

**Optimized for**
- Intel® Atom®
- Intel® Core™
- Intel® Xeon® Phi™
- Intel® Xeon®

**Supports**
- Windows
- Linux
- Android
- MacOS

**Addresses**
- Data Center
- Internet of Things
- Embedded Systems
- Cloud Computing

**Image Processing Uses**
- Medical Imaging
- Computer Vision
- Digital Surveillance
- ADAS
- Automated Sorting
- Biometric Identification
- Visual Search

**Signal Processing Uses**
- Games (sophisticated audio content or effects)
- Echo cancellation
- Telecommunications
- Energy

**Data Compression & Cryptography Uses**
- Data centers
- Enterprise data management
- ID verification
- Smart Cards/wallets
- Electronic Signature
- Information security/cybersecurity

Find out more at: [http://software.intel.com/intel-ipp](http://software.intel.com/intel-ipp)
What’s Inside Intel® Integrated Performance Primitives

High Performance, Easy-to-Use & Production Ready APIs

Image Processing

Computer Vision

Color Conversion

Image Domain

Signal Processing

Vector Math

Signal Domain

Data Compression

Cryptography

String Processing

Data Domain

Intel® Architecture Platforms


1 Available only in Intel® Parallel Studio Composer Edition.
Intel® IPP Benefits to Applications

Cloud and Server application
- Web image processing (resize, filtering, etc.)
- Web data compression and transferring, data encryption/decryption

Medical Images
- CT, MRI signal processing
- Medical image processing

Storage
- Storage data compression
- Storage data encryption/decryption

Print Imaging
- Image enhancement and correction
- Data compression

Digital Surveillance
- Computer vision
- Image recognition

Signal Processing
- Seismic data analysis, radar and sonar signal processing.

Machine Vision
- Image filtering, segmentation
- Edge detection, pattern recognition

In-Vehicle Infotainment
- Image and audio data processing

Biometric Identification
- Biometric image and signal processing

Visual Search
- Examining image content (color, shape, texture...)

Communication
- Wireless communication single processing
- CRC and MIMO functions for communication.

And More
- Digital media, security, mobile......
Gets Good Performance with Intel® IPP

In popular apps like WeChat*, QQ*, and QQ Album* the volume of newly generated images reach about 100 petabytes. Some users may try to upload illegal images (e.g., porn). The system has to run a check on each image to try to block them. Imagine trying to search through 100 petabytes of data.

IPP filter function (ipp_filter2D) took 9ms to perform the operation when compared to 143ms with openCV. The IPP filter2D is 15x faster than the OpenCV* plain code.

JD.com business has grown rapidly, from offering approximately 1.5 million SKUs in 2011 to approximately 25.7 million in 2013. Today, JD.com must handle petabytes of data, which takes an efficient, robust, distributed file system.

JD.com speeds up its image processing 17x – handling 300,000 images in 162 seconds instead of 2800 seconds.

Tencent doubled the speed of its image filter System

JD.com sped image processing with Intel® IPP

More Case Studies
INTEL® DATA ANALYTICS ACCELERATION LIBRARY
Boost Machine Learning & Data Analytics Performance

- Helps applications deliver better predictions faster
- Optimizes data ingestion & algorithmic compute together for highest performance
- Supports offline, streaming & distributed usage models to meet a range of application needs
- Split analytics workloads between edge devices and cloud to optimize overall application throughput

Learn More: software.intel.com/daal

What’s New in the 2019 Release

New Algorithms

- **High performance Logistic Regression**, most widely-used classification algorithm
- **Extended Gradient Boosting Functionality** provides inexact split calculations & algorithm-level computation canceling by user-defined callback for greater flexibility
- **User-defined Data Modification Procedure in CSV & IDBC data sources to implement** a wide range of feature extraction & transformation techniques

Pre-processing
- Decompression, Filtering, Normalization

Transformation
- Aggregation, Dimension Reduction

Analysis
- Summary Statistics, Clustering, etc.

Modeling
- Machine Learning (Training) Parameter Estimation Simulation

Validation
- Hypothesis Testing, Model Errors

Decision Making
- Forecasting, Decision Trees, etc.
Performance Scaling with Intel® Data Analytics Acceleration Library (Intel® DAAL)

Within a CPU Core
- SIMD vectorization: optimized for the latest instruction sets, Intel® AVX2, AVX512...
- Internally relies on sequential Math Kernel Library

Scale to Multicores or Many Cores
- Threading Building Blocks threading

Scale to Cluster
- Distributed processing done by user application (MPI, MapReduce, etc.)
- Intel® DAAL provides
  - Data structures for partial and intermediate results
  - Functions to combine partial or intermediate results into global result
Processing Modes

Batch Processing

\[ R = F(D_1, ..., D_k) \]

Online Processing

\[ S_{i+1} = T(S_i, D_i) \]
\[ R_{i+1} = F(S_{i+1}) \]

Distributed Processing

\[ R = F(R_1, ..., R_k) \]
Machine Learning Algorithms
Intel® Data Analytics Acceleration Library

Supervised Learning
- Neural Networks

Classification
- Regression
  - Logistic Regression
  - Ridge Regression
  - Linear Regression
- Unsupervised Learning
  - K-Means Clustering
  - EM for GMM

Regression
- Decision Forest
- Decision Tree
- Boosting (Ada, Brown, Logit)
- Naïve Bayes
- k-NN
- Support Vector Machine

Unsupervised Learning
- Collaborative Filtering
- Alternating Least Squares

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
Data Transformation & Analysis Algorithms

Intel® Data Analytics Acceleration Library

**Basic Statistics for Datasets**
- Low Order Moments
  - Quantiles
- Order Statistics

**Correlation & Dependence**
- Cosine Distance
- Correlation Distance
- Variance-Covariance Matrix

**Matrix Factorizations**
- SVD
- QR
- Cholesky

**Dimensionality Reduction**
- PCA
- Association Rule Mining (Apriori)
- Optimization Solvers (SGD, AdaGrad, lBFGS)

**Outlier Detection**
- Univariate
- Multivariate

Algorithms supporting batch processing
Algorithms supporting batch, online and/or distributed processing

*Other names and brands may be claimed as the property of others.*
Example Performance Benchmark: Speedup over XGBoost*  
Intel® Data Analytics Acceleration Library (Intel® DAAL)  

Intel® DAAL is 2-12x Faster  

![Graph showing speedup factors for Higgs, MLSR, Slice localization, and Year prediction.](image)
Get the Benefits of Advanced Threading with Threading Building Blocks

Use Threading to Leverage Multicore Performance & Heterogeneous Computing

- Parallelize computationally intensive work across CPUs, GPUs & FPGAs,—deliver higher-level & simpler solutions using C++
- Most feature-rich & comprehensive solution for parallel programming
- Highly portable, composable, affordable, approachable, future-proof scalability

What’s New in 2019 Release

- New capabilities in Flow Graph improve concurrency & heterogeneity through improved task analyzer & OpenCL* device selection
- New templates to optimize C++11 multidimensional arrays
- C++17 Parallel STL, OpenCL*, & Python* Conda language support
- Expanded Windows*, Linux*, Android*, MacOS* support

Learn More: software.intel.com/intel-tbb
What’s Inside Threading Building Blocks

Parallel Execution Interfaces
- Flow Graph
- Generic Parallel Patterns
- Parallel STL

Low-Level Interfaces
- Tasks
- Task arenas
- Global Control

Interfaces Independent of Execution Model
- Concurrent Containers: Hash Tables, Queues, Vectors
- Memory Allocation: Scalable Allocator, Cache Aligned Allocator
- Primitives and Utilities: Synchronization Primitives, Thread Local Storage
Heterogeneous Support
Threading Building Blocks (TBB)

TBB flow graph as a coordination layer for heterogeneity—retains optimization opportunities & composes with existing models

CPU, integrated GPUs, etc.

- Threading Building Blocks
  - OpenVX*
  - OpenCL*
  - COI/SCIF
  - ...

TBB as a *composability layer* for library implementations
- One threading engine *underneath* all CPU-side work

TBB flow graph as a *coordination layer*
- Be the glue that connects heterogeneous hardware & software together
- Expose parallelism between blocks—simplify integration
Advantages of Using Threading Building Blocks over other Threading Models

- Specify tasks instead of manipulating threads. Threading Building Blocks (TBB) maps your logical tasks onto threads with full support for nested parallelism.

- TBB uses proven, efficient parallel patterns.

- TBB uses work stealing to support the load balance of unknown execution time for tasks. This has the advantage of low-overhead polymorphism.

- Flow graph feature in TBB allows developers to easily express dependency and data flow graphs.

- Has high level parallel algorithms, concurrent containers, and low level building blocks like scalable memory allocator, locks and atomic operations.
Excellent Performance Scalability with Threading Building Blocks on Intel® Xeon® Processor

Performance results are based on testing as of July 31, 2018 and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks. 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 complete information visit www.intel.com/benchmarks. Configuration: Testing by Intel as of July 31, 2018. Software versions: Intel® C++ Intel® 64 Compiler, Version 18.0, Threading Building Blocks (TBB) 2019; Hardware: 2x Intel® Xeon® Gold 6152 CPU @ 2.10GHz, 192GB Main Memory; Operating System: CentOS Linux* release 7.4 1708 (Core), kernel 3.10.0-693.e17.x86_64; Note: sudoku, primes and tachyon are included with TBB.

Intels 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. For more complete information about compiler optimizations, see our Optimization Notice.
Performance results are based on testing as of August 2017 to September 2018 and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure.

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 complete information visit www.intel.com/benchmarks.

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.

Copyright © 2018, Intel Corporation. All rights reserved. Intel, Xeon, Core, VTune, and the Intel logo 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
BACKUP