INTEL® MATH KERNEL LIBRARY - INTRODUCTION AND GENERAL TIPS

Gennady Fedorov - Technical Consulting Engineer

Intel Architecture, Graphics and Software (IAGS)

PRACE workshop, June 2020

Gennady.Fedorov@intel.com

Agenda

- Introduction: Intel IPP, DAAL, **MKL**
- MKL BLAS, Lab exercises
- MKL Sparse BLAS, Lab exercises
- MKL FFT, Lab exercises
- MKL RNG, Lab exercises

Intel® Integrated Performance Primitives

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

Image Domain	Signal Domain	Data Domain
Image Processing	Signal Processing	Data Compression
Computer Vision	Vector Math	Cryptography
Color Conversion		String Processing

Intel® Architecture Platforms

Operating System: Windows*, Linux*, Android*, MacOS[1]*

[1] Available only in Intel® Parallel Studio Composer Edition.

Intel® Data Analytics Acceleration Library (Intel® DAAL)

Boost Machine Learning & Data Analytics Performance

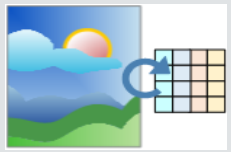
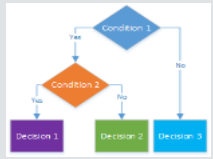
- Helps applications deliver better predictions faster
- Optimizes data ingestion & algorithmic compute together for highest performance
- Supports offline, streaming & distributed usage models to meet a range of application needs
- Split analytics workloads between edge devices and cloud to optimize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

Learn More: software.intel.com/daal

What's New in the 2020 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	Transformation	Analysis	Modeling	Validation	Decision Making
					
Decompression, Filtering, Normalization	Aggregation, Dimension Reduction	Summary Statistics Clustering, etc.	Machine Learning (Training) Parameter Estimation Simulation	Hypothesis Testing Model Errors	Forecasting Decision Trees, etc.

Intel® Math Kernel Library

Linear Algebra
- BLAS
- LAPACK
- ScaLAPACK
- Sparse BLAS
- Iterative sparse solvers
- PARDISO*
- Cluster Sparse Solver

FFTs
- Multidimensional
- FFTW interfaces
- Cluster FFT

Neural Networks
- Convolution
- Pooling
- Normalization
- ReLU
- Inner Product

Removed since MKL v.2020

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

Benchmarks
- Intel(R) Distribution for LINPACK* Benchmark
- High Performance Computing Linpack Benchmark
- High Performance Conjugate radient Benchmark

Intel® Architecture Platforms

Operating System: Windows*, Linux*, MacOS[1]*

[1] Available only in Intel® Parallel Studio Composer Edition.

What's New for Intel® MKL v.2019?

- Just-In-Time Fast Small Matrix Multiplication : Improved speed of S/DGEMM for Intel® AVX2 and Intel® AVX-512 with JIT capabilities
- CNR mode: independent of the number of threads (strict mode, BLAS)
- New sparseQR Solvers : for 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
  - Great for finance, geological and biological applications

What's New in Intel® MKL 2020

BLAS

- *Improved performance of small to medium size iGEMM when C-offset is non-zero on Intel® AVX2 and Intel® AVX 512 architecture sets.*
- *Improved SGEMM performance for TN case for small N.*
- *Added 8-bit/16-bit iGEMM optimizations for Intel® AVX and SSE4.2 architecture sets*

ScaLAPACK:

- *Introduced P{D,S}TREVC functions for computing some or all of the right and/or left eigenvectors of a real upper quasi-triangular matrix.*

Random number generators(RNGs):

- *Introduced an advanced SkipAhead method for parallel random number generation by MRG32k3a/Philox4x32-10/ARS-5 basic random number generator. (up to 19x speedup)*
- *Improved performance of ARS-5 basic random number generator for Intel® AVX 512 systems.*

Summary statistics:

- *Improved performance of fast calculation method for raw/central moments/sums, variance-covariance/correlation/cross-product matrix on Intel® AVX2 and Intel® AVX 512 architecture sets.*

Library Engineering:

- *Introduced module file support.*

Graph – since version MKL v.2020 update 1

Graph algorithms which can already be faster with MKL kernels:

- PageRank
- TriangleCount

Next targets:

- Breadth-First Search, Connected Components, Single-Source Shortest Paths, Betweenness Centrality (already have GraphBLAS-based implementation w/o MKL)
- Jaccard coefficients, Maximal Independent Set, …
- Clustering algorithms

What's New for oneAPI Math Kernel Library (oneMKL) Beta

- Support for Intel® Processor Graphics Gen9
- Support for Intel® Data Parallel C++ language bindings:
  - **BLAS** – full support CPU & Gen9
  - **LAPACK** – CPU: Select dense linear solvers, select dense eigensolvers and select batched LAPACK functions; Gen9: Batched & non-batched: LU factorization/solve/inverse, Cholesky factorization/solve, QR factorization; Non-batched: triangular matrix solve, symmetric eigensolver
  - **FFT** – CPU: 1D, 2D, 3D, C2C; Gen9: 1D, C2C
  - **RNG** – CPU: full support of pseudo-random and quasi-random Engines, continues (except gaussian_mv) and Discrete Distributions; Gen9: Philox4x32-10 and Mrg32k3a Engines, Uniform/Gaussian/Log-normal/Discrete Uniform/Unifrom Bits distributions.
  - **VectorMath** – CPU & Gen9
- Limited support for OpenMP variant of Intel® Processor Graphics Gen9 offload for C/C++
  - BLAS - S/GEMM

Distribution, packaging

- Intel® Parallel Studio XE
- Intel® System Studio
- YUM - https://software.intel.com/en-us/articles/installing-intel-free-libs-and-python-yum-repo
- APT - https://software.intel.com/en-us/articles/installing-intel-free-libs-and-python-apt-repo
- Conda - https://software.intel.com/en-us/articles/using-intel-distribution-for-python-with-anaconda
- PIP - https://software.intel.com/en-us/articles/installing-the-intel-distribution-for-python-and-intel-performance-libraries-with-pip-and
- UWD(Universal Windows* Driver) - https://software.intel.com/en-us/articles/using-intel-performance-libraries-in-universal-windows-drivers

Supported Operating Systems

Linux Distributions:

- Red Hat* Enterprise Linux* 6, 7, 7.5 (IA-32 / Intel® 64)
- Red Hat Fedora* core 27, 28 (IA-32 / Intel® 64)
- SUSE Linux Enterprise Server* 11, 12
- CentOS 6.0, 7.1, 7.2
- Debian* 8, 9 (IA-32 / Intel® 64)
- Ubuntu* 16.04, 17.04, 18.04 LTS (IA-32/Intel® 64)
- WindRiver Linux 8, 9 and 10

Windows* versions:

- Windows 10 (IA-32 / Intel® 64)
- Windows 8.1* (IA-32 / Intel® 64)
- Windows 7* SP1 (IA-32 / Intel® 64)
- Windows HPC Server 2016 (Intel® 64)
- Windows HPC Server 2012 (Intel® 64)
- Windows HPC Server 2008 R2 (Intel® 64)

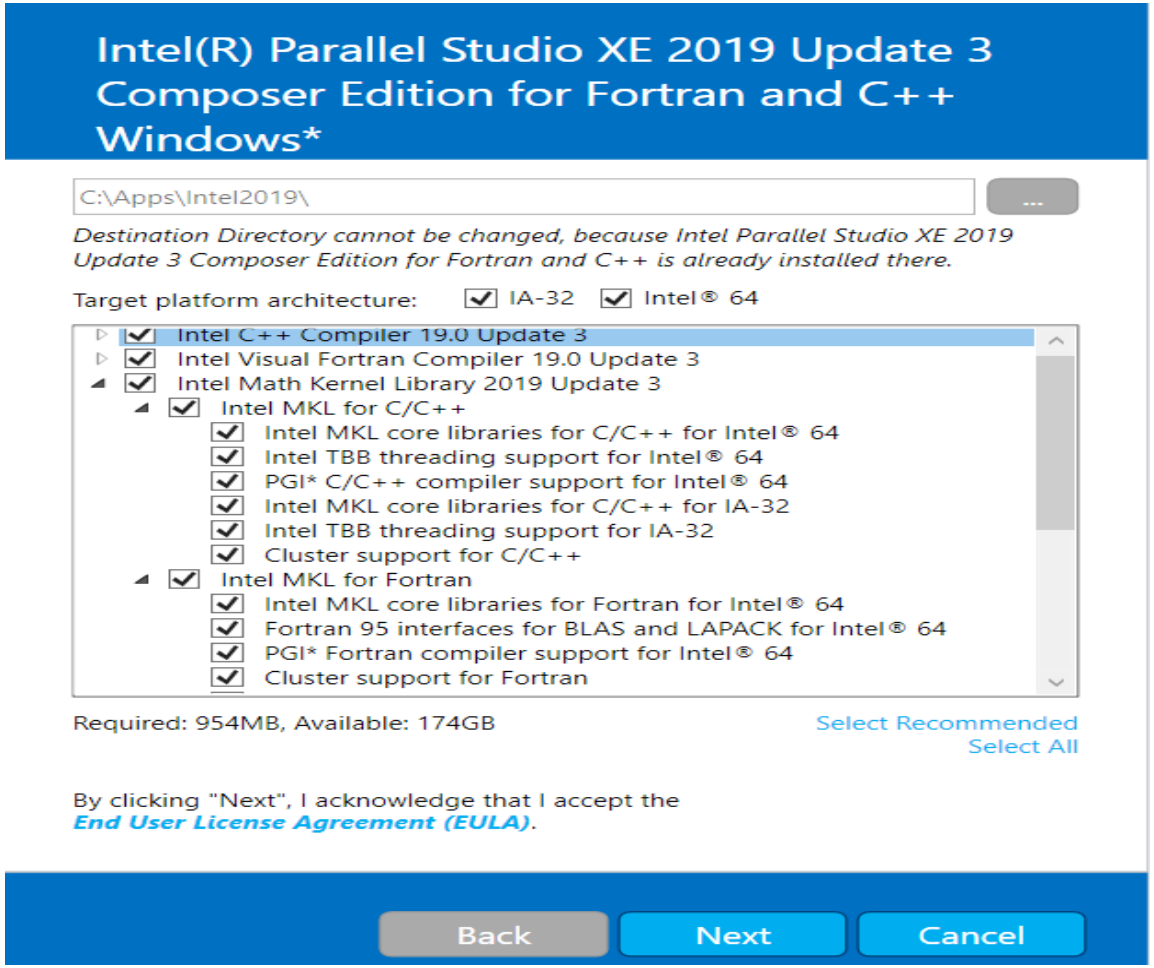
OS* support

Note: Intel MKL is expected to work on many more Linux* distributions as well. Let us know if you have trouble with the distribution you use.

https://software.intel.com/en-us/articles/intel-math-kernel-library-intel-mkl-2020-system-requirements

Installation

- Download the Intel® Parallel Studio XE 2019 Composer Edition product from Intel Registration Center
- Choose destination directory
- Choose your target platform architecture
- Refer to the Install Guide KB Article: https://software.intel.com/en-us/articles/intel-math-kernel-library-intel-mkl-2020-install-guide



Performance Benchmarks

https://software.intel.com/en-us/mkl/features/benchmarks

PERFORMANCE BENCHMARKS

This comprehensive table will help you make informed decisions about which routines to use in your applications, including performance for each major function domain in Intel® Math Kernel Library (Intel® MKL) by processor family. Each link displays a chart. Some benchmark charts only include absolute performance measurements for specific problem sizes. Other charts compare previous versions, popular alternate open-source libraries, and other functions for Intel MKL.

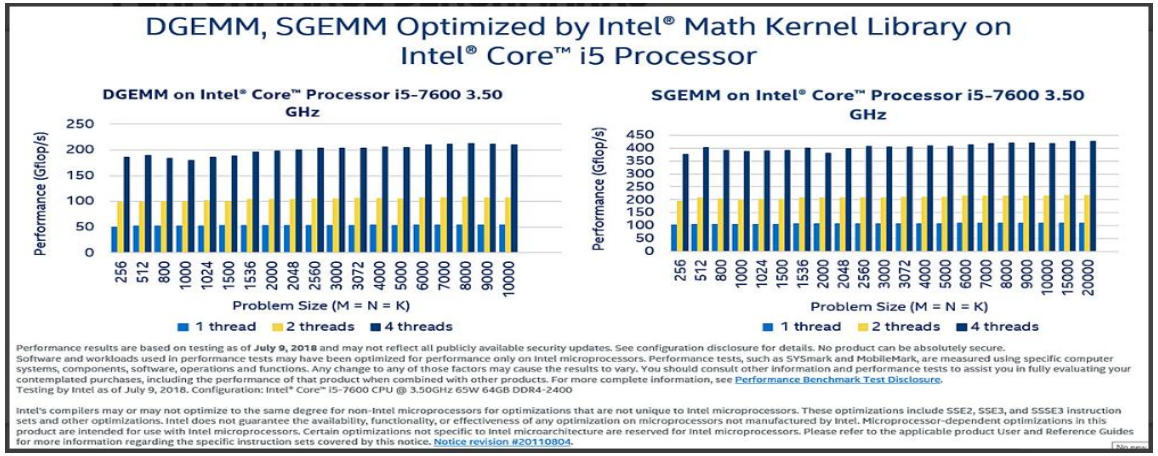
Compare selected:

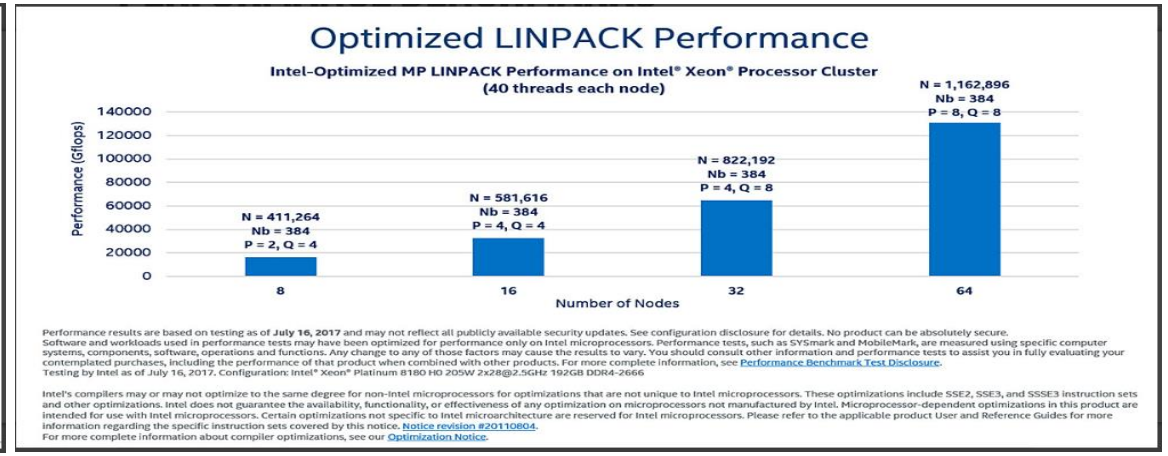
- ☑ Intel® Core™ i5 Processor Benchmarks
- ☑ Intel® Xeon® Processor Benchmarks

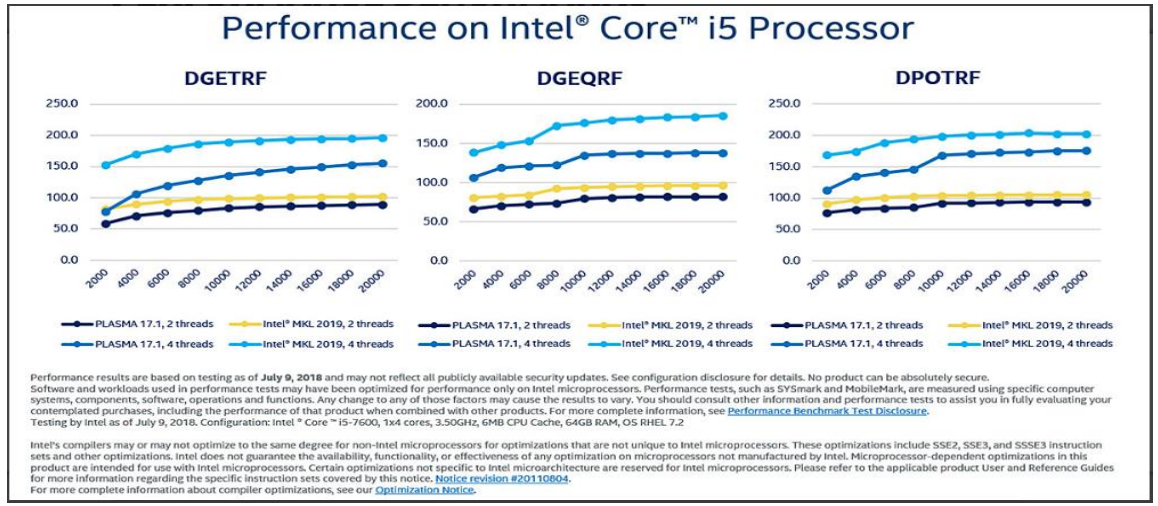
Component	Intel® Core™ i5 Processor Benchmarks	Intel® Xeon® Processor Benchmarks
BLAS	DGEMM and SGEMM INTEL MKL BLAS vs BLIS vs OpenBLAS JIT DGEMM and SGEMM	DGEMM and SGEMM INTEL MKL BLAS vs BLIS vs OpenBLAS JIT DGEMM and SGEMM
LAPACK	DGETRF, DGEQRF, DPOTRF	DGETRF, DGEQRF, DPOTRF
ScaLAPACK	NONE	PDSYEVD, PDSYEVXM
LINPACK	NONE	LINPACK
Fast Fourier Transform (FFT)	2D FFT vs FFTW 3D FFT vs FFTW	2D FFT vs FFTW 3D FFT vs FFTW
Sparse Matrix-Vector Multiplication (SpMV) and Sparse Solvers	SpMV	SpMV PARDISO vs. Multifunctional Massively Parallel Sparse (MUMPS) Cluster Sparse Solver vs. MUMPS

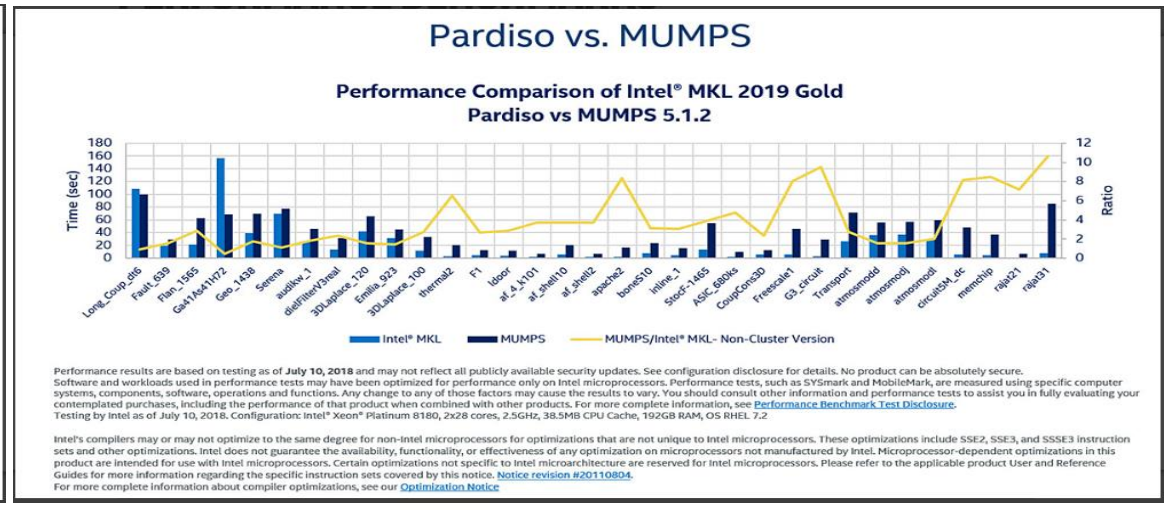
Performance Benchmarks

https://software.intel.com/en-us/mkl/features/benchmarks





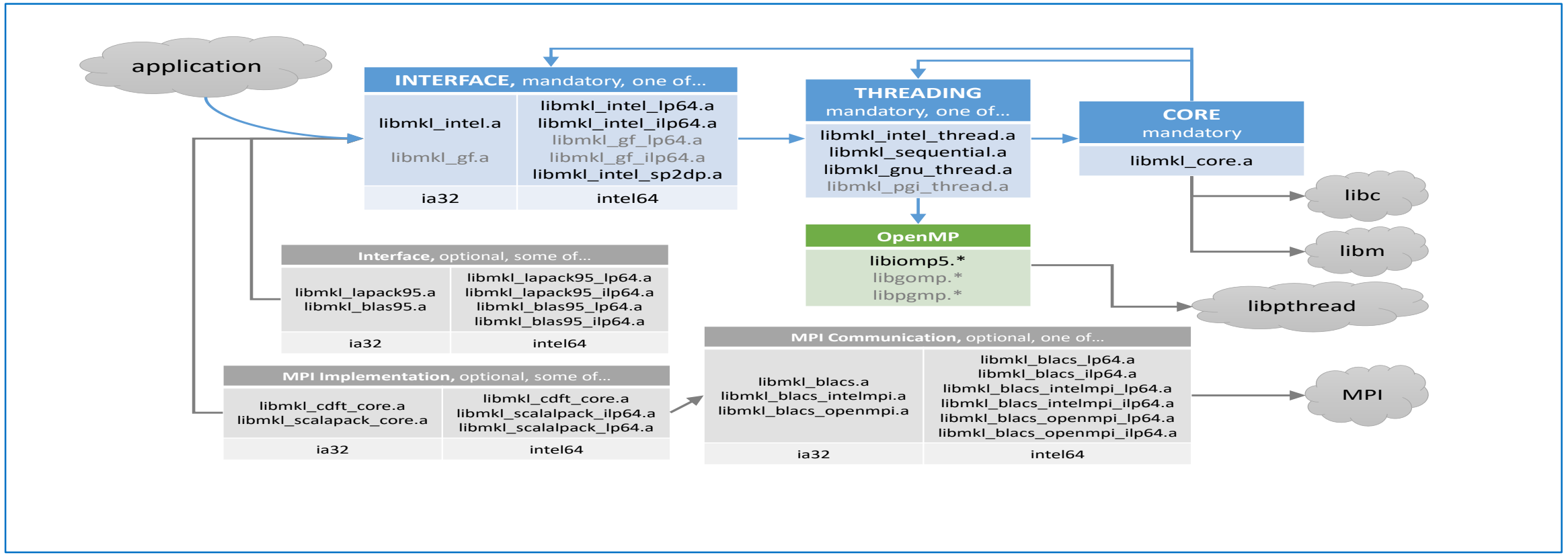




Agenda

- Introduction
- **MKL usage modes, tips**
- Known problems & Deprecations

Usage modes - Layer model



+TBB threading Layer

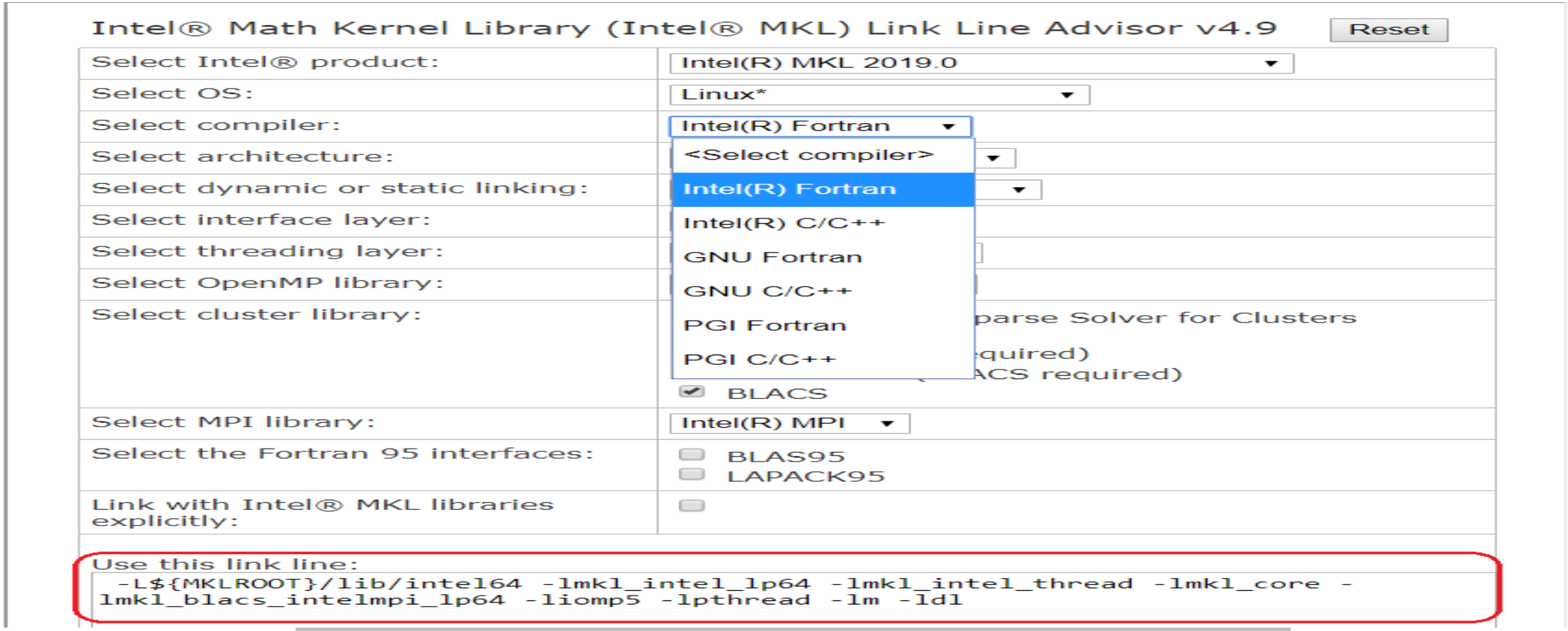
Usage Modes – Linking Quick Start

- Using a command line tool: (..\mkl\bin**mkl_link_tool.exe**)
- Using an interactive interface (Linking Adviser)
- Explicitly listing libraries on your link line (see User's Guide)
- Using the /Qmkl compiler options:

/Qmkl (-mkl)	
/Qmkl:parallel	to link with standard threaded Intel MKL.
/Qmkl:sequential	to link with sequential version of Intel MKL.
/Qmkl:cluster	to link with Intel MKL cluster components (sequential) that use Intel MPI.

- Automatically Linking a Project in the Visual Studio* Integrated Development Environment with Intel® MKL:
  - Automatically Linking Your Microsoft Visual C/C++* Project with Intel® MKL
  - Automatically Linking Your Intel® Visual Fortran Project with Intel® MKL

Usage Modes – Link Line Advisor tool



http://software.intel.com/en-us/articles/intel-mkl-link-line-advisor/

Usage Model – Threading in Intel MKL

MKL's routines are threaded via OpenMP or TBB (libiomp - <complier_install>/lib, tbb - <tbb_install>/lib/)

Techniques to Set the Number of Threads:

OMP

```
Set one of the OpenMP or Intel MKL environment variables:
    OMP_NUM_THREADS
    MKL_NUM_THREADS
    MKL_DOMAIN_NUM_THREADS
Call one of the OpenMP or Intel MKL functions:
   omp_set_num_threads()
   mkl_set_num_threads()
   mkl_domain_set_num_threads()
   mkl_set_num_threads_local()
```

TBB

```
int nThreads = tbb::task_scheduler_init::default_num_threads();
tbb::task_scheduler_init init(nThreads);
```

Intel MKL is *thread-safe. 1 exception - LAPACK deprecated routine ?lacon*

Usage Model – Threading in Intel MKL, OpenMP, MPI

	Where's the Parallelism?		
Domain	**SIMD**	**Open MP**	**MPI**
BLAS 1, 2, 3, SpBLAS	X	X	
FFTs	X	X	
LAPACK	X (relies on BLAS 3)	X	
ScaLAPACK **(Cluster dense LA solvers)**		X (hybrid)	X
Direct Sparse Solver	X (relies on BLAS 3)	X	
VML/VSL	X	X	
Cluster FFT		X	X

Usage Model - Threading in Intel MKL - TBB

- **BLAS** Level 3 routines (gemm, hemm, herk,symm, syrk, trmm,trsm)
- **LAPACK**:
  ?geqrf, ?gelqf, ?getrf, ?potrf, ?unmqr*, ?ormqr*, ?unmrq*, ?ormrq*, ?unmlq*, ?ormlq*, ?unmql*, ?ormql*, ?sytrd, ?hetrd, ?syev, ?heev, and ?latrd.
- **Parallel Direct Sparse Solver** (intel® MKL Pardiso)
- **Sparse BLAS**: **mkl_sparse_?_mv** (CSR, BSR matrix formats) and **mkl_sparse_?_mm** (CSR sparse matrix format)
- **VML** functions

Usage Model – Memory Management in MKL

MKL MM allocates and deallocates internal buffers to facilitate better performance.

mkl_malloc

mkl_calloc

mkl_realloc

mkl_free

mkl_mem_stat

mkl_peak_mem_usage

mkl_free_buffers

mkl_thread_free_buffers

mkl_disable_fast_mm

Usage Model – Custom DLL Builder

Using the Custom Dynamic-link Library Builder in the Command-line Mode

nmake <target> <options> target == libia32, libintel64, dllia32, dllintel64, help

Options:

Interface
- IA-32 : {cdecl|stdcall}. The default value is cdecl.
- Intel 64 : lp64, ilp64

Threading = {parallel|sequential}
Export = *<file name> :* the list of entry-point functions to be included in the DLL
Name = *<dll name>*

Example:

nmake ia32 interface=stdcall export=my_func_list.txt name= **my_mkl_small**

cat my_function_list.txt:

DGEMM

DGETRF

*****

Usage Model – Conditional Numerical Reproducibility

- make sure your application uses a fixed number of threads and call

int mkl_cbwr_set(int CNR branch);

or Environment Variables

set/export MKL_CWBR ==

/* branch specific values */

MKL_CBWR_OFF	0
MKL_CBWR_BRANCH_OFF	1
MKL_CBWR_AUTO	2
MKL_CBWR_COMPATIBLE	3
MKL_CBWR_SSE2	4
MKL_CBWR_SSSE3	6
MKL_CBWR_SSE4_1	7
MKL_CBWR_SSE4_2	8
MKL_CBWR_AVX	9
MKL_CBWR_AVX2	10
MKL_CBWR_AVX512_MIC	11
MKL_CBWR_AVX512	12
MKL_CBWR_AVX512_MIC_E1	13
MKL_CBWR_AVX512_E1	14

Usage Model – CNR, STRICT mode

strict CNR mode -- the identical results even if the #of threads varies

Available Since MKL 2019 u3:

- ?gemm, ?trsm, ?symm, ?hemm and their CBLAS equivalents
- IA: must be set to AVX2 or later
- OpenMP and TBB based threads
- The 64-bit Intel MKL libraries are used
- To enable strict CNR, add the new MKL_CBWR_STRICT flag to the CNR code-path:
  - mkl_cbwr_set(MKL_CBWR_AVX512 | MKL_CBWR_STRICT) or
  - append ",STRICT" to the MKL_CBWR env variable: **set/export MKL_CBWR = AVX2,STRICT**

Usage Model – Dispatching in MKL

- MKL_ENABLE_INSTRUCTIONS environment variable
- int mkl_enable_instructions (int *isa*);

  1 - Intel MKL dispatches the code path for the specified ISA by default.

  0 - The request is rejected.

  Usually this occurs if mkl_enable_instructions was called:

  - After another Intel MKL function
  - On a non-Intel architecture
  - With an incompatible ISA specified

Usage Model – Dispatching in MKL, cont

```
_MKL_API(int,MKL_Enable_Instructions,(int))
#define  mkl_enable_instructions
MKL_Enable_Instructions
#define  MKL_ENABLE_SSE4_2              0
#define  MKL_ENABLE_AVX                 1
#define  MKL_ENABLE_AVX2                2
#define  MKL_ENABLE_AVX512_MIC          3
#define  MKL_ENABLE_AVX512              4
#define  MKL_ENABLE_AVX512_MIC_E1       5
#define  MKL_ENABLE_AVX512_E1           6
```

MKL_ENABLE_AVX512
Intel® Advanced Vector Extensions 512 (Intel® AVX-512) on Intel® Xeon® processors.

MKL_ENABLE_AVX512_E1
Intel® Advanced Vector Extensions 512 (Intel® AVX-512) with support for Vector Neural Network Instructions.

MKL_ENABLE_AVX512_MIC
Intel AVX-512 on Intel® Xeon Phi™ processors.

MKL_ENABLE_AVX2
Intel® Advanced Vector Extensions 2 (Intel® AVX2).

MKL_ENABLE_AVX
Intel® Advanced Vector Extensions (Intel® AVX).

MKL_ENABLE_SSE4_2
Intel® Streaming SIMD Extensions 4-2 (Intel® SSE4-2).

Timing in MKL

ssecnd/dsecnd - Returns elapsed time in seconds. Use to estimate real time between two calls to this function.

Example:

```
double time_st = dsecnd();
    for (size_t i=0; i<LOOP_COUNT; ++i) {
        vsMul(409600, a, b, c);
    }
double time_end = dsecnd();
 double time_avg = (time_end - time_st)/LOOP_COUNT;
```

MKL MPI wrappers

Supported MPI: Intel MPI (v. 2017,2018 and 2019) MPICH(v. 2.14, 3.1, 3.3), OpenMPI 1.8x, MS MPI

While different MPI libraries are compatible on the application programming interface (API) level, they are often incompatible at the application binary interface (ABI) level. So Intel MKL provides some different libraries to support the different MPIs. For example, one should link with **libmkl_blacs_lp64.a** to use application with MPICH*, **libmkl_blacs_openmpi_lp64.a** to use Open MPI*. If users link Intel MKL cluster functions with the customized MPI libraries, which is not supported by Intel MKL, and may create some unexpected result

Affected: Cluster Sparse Solver, Cluster FFT, Scala PACK, BLACS components

Solution - *Using Intel MKL MPI wrapper code:*

- *source code available* MKLROOT/interfaces/mklmpi
- building the Intel MKL wrapper code (custom BLACS)
- linking with custom MKL BLACS library
- *KB Article :* https://software.intel.com/en-us/articles/using-intel-mkl-mpi-wrapper-with-the-intel-mkl-cluster-functions

Agenda

- Introduction
- MKL usage modes, tips
- **Known problems & Deprecation**

Known problems & Limitations - BLAS

BLAS: Small ?GEMM improving to eliminate the existing Limitations: CNR, GNU* Fortran, Verbose mode and BLAS95 API are not supported

CNR mode: #arbitrary #of threads, different OS…

FFT: All Intel MKL function domains support ILP64 programming but FFTW interfaces to Intel MKL:

- FFTW 2.x wrappers do not support ILP64
- FFTW 3.x wrappers support ILP64 by a dedicated set of functions plan_guru64
- 3D case API only – available only (built in). 2D cases – need to compile the wrapper library source fftw2xc fftw2xf fftw2x_cdft

Solvers: Preconditioners (ILUT, ILUO): are not threaded, don't support of complex data types. Iterative Solvers(CG, FGMRES): don't support of complex data types

LINPACK Benchmark: Intel Optimized LINPACK Benchmark supports only OpenMP threading. The best performance will be obtained with the Intel® Hyper-Threading Technology turned off

Bug Fix list: https://software.intel.com/en-us/articles/intel-math-kernel-library-2020-bug-fixes

Intel® MKL - Deprecation

- **Deep Neural Network (DNN)**
  - DNN is deprecated and will be removed in the next Intel MKL release. We will continue to provide optimized functions for deep neural networks in Intel Math Kernel Library for Deep Neural Networks (Intel MKL-DNN)
- **Removed support for 32 bit applications on macOS***
  - If users require 32-bit support on macOS*, they should use MKL 2018 or early versions
- **SpBLAS (NIST) API**
  - Sparse BLAS API is deprecated and will be removed in the next Intel MKL release. Please use sparse BLAS IE API instead of.
- **FFTW2 ?**

Legal Disclaimer & Optimization Notice

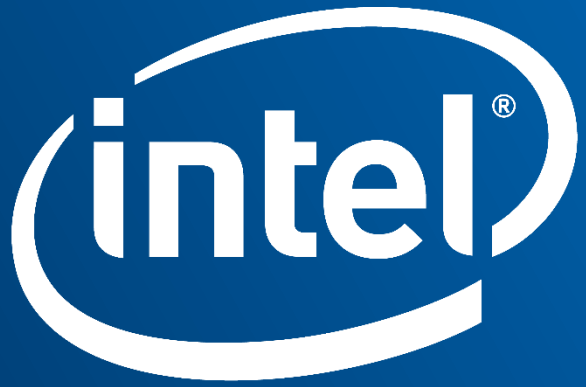
INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING 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.

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



Software