Intel® Math Kernel Library (Intel® MKL)

Naveen Gv

Intel Corporation

Third-party Tools Powered by Intel® Math Kernel Library

IMSL* Fortran Numerical Libraries (Rogue Wave)

NAG* Libraries

MATLAB* (MathWorks)

GNU Octave*

NumPy* / SciPy*

PETSc* (Portable Extensible Toolkit for Scientific Computation)

WRF* (Weather Research & Forecasting run-time environment)

The HPCC* benchmark

And more …

Motivation

How and where to optimize?

1. Appropriate algorithm
2. **Performance Library**
3. Multicore
4. SIMD

Delivered Values

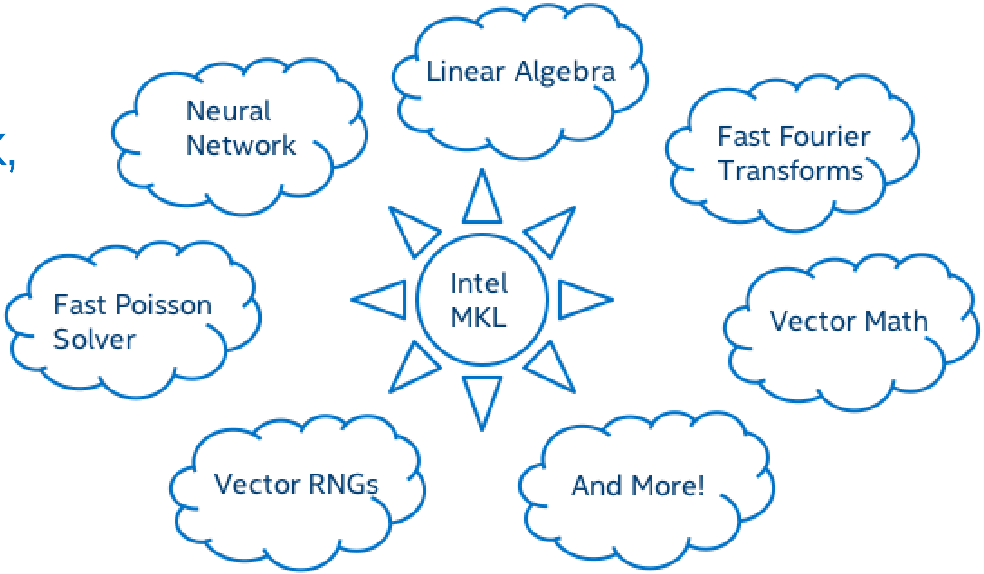
- Easy access to high perf.
- Rich functionality
- Support

```
for (int i = 0; i < M; ++i) {
 for (int j = 0; j < N; ++j) {
  c[i*K+j] = 0;
  for (int k = 0; k < K; ++k) {
   c[i*K+j] += a[i*N+k]
             * b[k*K+j];
  }
 }
}
```

Intel® Math Kernel Library

Intel® Math Kernel Library

- Speeds math processing for machine learning, scientific, engineering financial and design applications
- Includes functions for dense and sparse linear algebra (BLAS, LAPACK, PARDISO), FFTs, vector math, summary statistics and more
- De facto standard APIs for easy switching from other math libraries
- Highly optimized, threaded and vectorized to maximize processor performance



Components of Intel MKL

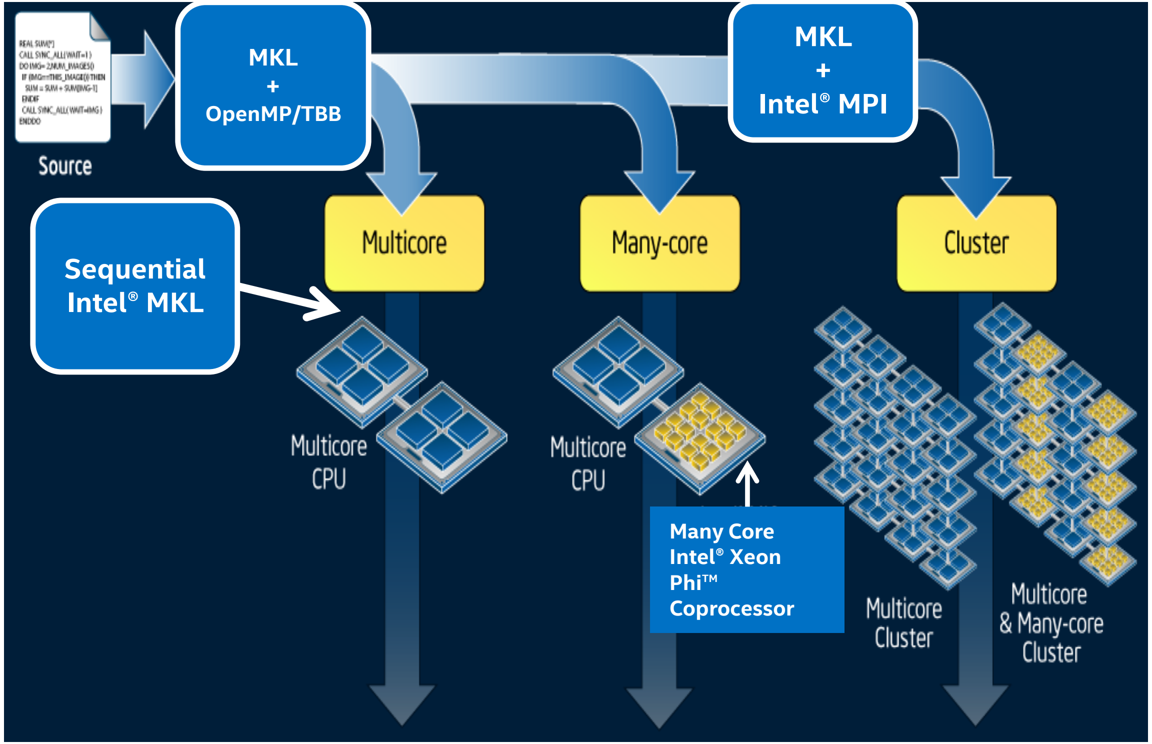
Linear Algebra	Fast Fourier Transforms	Vector Math	Summary Statistics	And More...	Deep Neural Networks
• BLAS • LAPACK • ScaLAPACK • Sparse BLAS • Sparse Solvers • Iterative • PARDISO* • Cluster Sparse Solver	• Multidimensional • FFTW interfaces • Cluster FFT	• Trigonometric • Hyperbolic • Exponential • Log • Power • Root • Vector RNGs	• Kurtosis • Variation coefficient • Order statistics • Min/max • Variance-covariance	• Splines • Interpolation • Trust Region • Fast Poisson Solver	• Convolution • Pooling • Normalization • ReLU • Softmax

Automatic Performance Scaling from the Core, to Multicore, to Many Core and Beyond

Intel® MKL

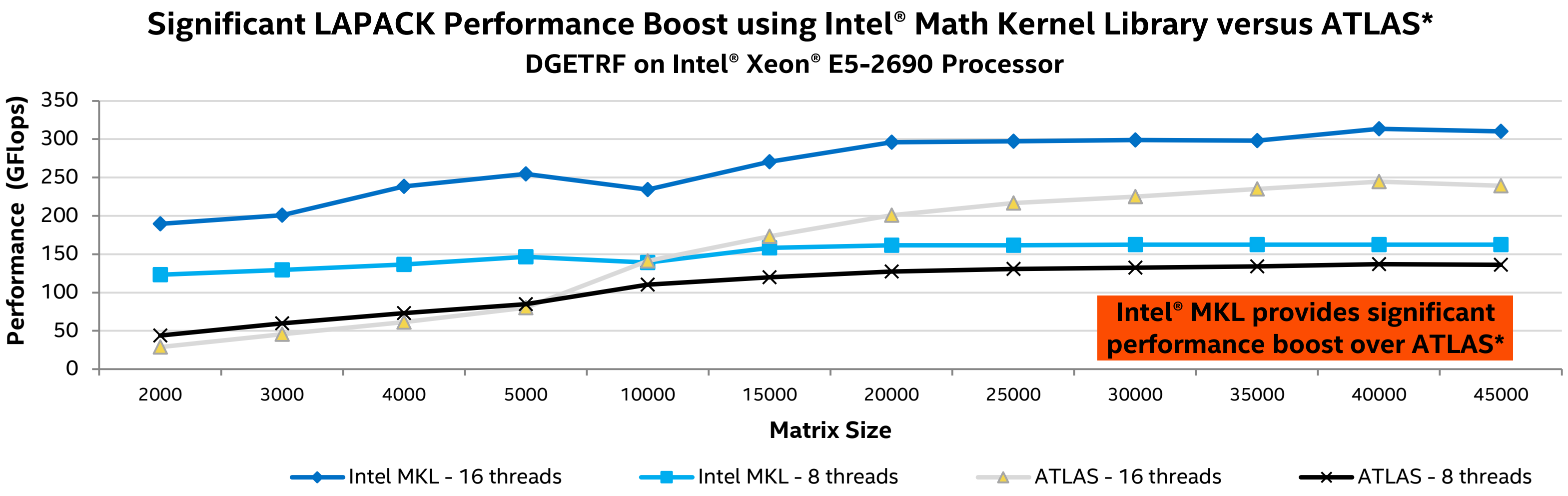
Extracting performance from the computing resources

- Core: **vectorization**, prefetching, cache utilization
- Multi-Many core (processor/socket) level **parallelization**
- Multi-socket (node) level **parallelization**
- Clusters **scaling**



Performance Benefit to Applications

Intel® MKL



Configuration: Hardware: CPU: Dual Intel® Xeon E5-2697v2@2.70Ghz; 64 GB RAM. Interconnect: Mellanox Technologies* MT27500 Family [ConnectX*-3] FDR.. Software: RedHat* RHEL 6.2; OFED 3.5-2; Intel® MPI Library 5.0 Intel® MPI Benchmarks 3.2.4 (default parameters; built with Intel® C++ Compiler XE 13.1.1 for Linux*);

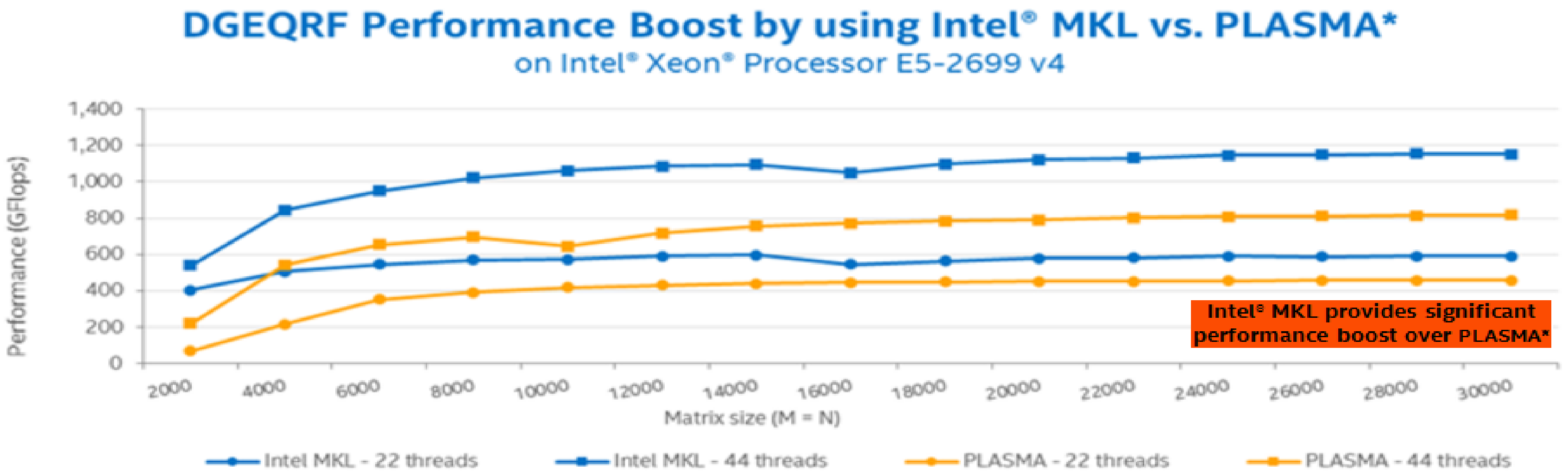
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may 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.

The latest version of Intel® MKL unleashes the performance benefits of Intel architectures

Performance Benefit to Applications

Intel® MKL



Configuration: Versions: Intel® Math Kernel Library (Intel® MKL) v.2017; Hardware: CPU: Intel® Xeon E5-2699 v4, Two Twenty-two core CPU (55 MB smart cache,2.2 Ghz, 64 GB RAM; Operation System: RedHat* RHEL 7.2 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. 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.

The latest version of Intel® MKL unleashes the performance benefits of Intel architectures

Optimization Notice

BLAS – Basic Linear Algebra Subprograms

Defacto-standard APIs since the 1980s (Fortran 77)

- Level 1 – vector-vector operations
- Level 2 – matrix-vector operations
- Level 3 – matrix-matrix operations
- Precisions: single, double, single complex, double complex

Original BLAS available at
http://netlib.org/blas/

Operation	MKL Routine "D is for double"	Example	Computational complexity (work)
Vector Vector	DAXPY	$y = y + \alpha x$	$O(N)$
Matrix Vector	DGEMV	$y = \alpha Ax + \beta y$	$O(N^2)$
Matrix Matrix	DGEMM	$C = \alpha A * B + \beta C$	$O(N^3)$

LAPACK – Linear Algebra PACKage

De-facto-standard APIs since early 1990s

1000s of linear algebra functions

4 floating point precisions supported

Breadth of coverage:

- Matrix factorizations: LU, Cholesky, QR, SVD and CSD
- Solving systems of linear equations
- Condition number estimates
- Singular value decomposition
- Symmetric and non-symmetric eigenvalue problems
- And much, much more
- Fully compatible with LAPACK version 3.6

Original LAPACK is available at:
http://netlib.org/lapack/

Fast Fourier Transform (FFT)

Support multidimensional transforms

Multiple transforms on single call

Input/output strides supported

- Allow FFT of a part of image, padding for better performance, transform combined with transposition, facilitates development of mixed-language applications.

Integrated FFTW interfaces

- Source code of FFTW3 and FFTW2 wrappers in C/C++ and Fortran are provided.
- FFTW3 wrappers are also built into the library.

Vector Math Functions

Example: y(i) = $e^{x(i)}$ for *i =1 to n*

- Arithmetic
  - add/sub/sqrt/ …
- Exponential and log
  - exp/pow/log/log10
- Trigonometric and hyperbolic
  - sin/cos/sincos/tan(h)
  - asin/acos/atan(h)
- Rounding
  - ceil, floor, round …
- And many more …

- Real and complex
- Single/double precision
- 3 accuracy modes
  - High accuracy
    - (Almost correctly rounded)
  - Low accuracy
    - (2 lowest bits in error)
  - Enhanced performance
    - (1/2 the bits correct)

Vector-based elementary functions allow developers to balance accuracy with performance

Vector Statistics

Random Number Generators (RNGs)	Psuedo-random, quasi-random, and non-deterministic generators
	Continuous and discrete distributions of various common distribution types
Summary Statistics (SS)	Parallelized algorithms for computation of statistical estimates for raw multi-dimensional datasets.
Convolution/ correlation	A set of routines intended to perform linear convolution and correlation transformations for single and double precision real and complex data.

Intel® MKL Sparse Solvers

PARDISO – Parallel Direct Sparse Solver	Support a wide range of matrix types.
	Based on BLAS level 3 update and pipelining parallelism.
	Supports out-of-core execution for huge problem sizes.
	New: **Parallel Direct Sparse Solver for Clusters**.
DSS – Direct Sparse Solver Interface for PARDISO	An alternative, simplified interface to PARDISO.
ISS – Iterative Sparse Solver	Symmetric positive definite: CG solver.
	Non-symmetric indefinite: Flexible generalized minimal residual solver.
	Based on Reverse Communication Interface (RCI).

More Intel® MKL Components

Data Fitting

1D linear, quadratic, cubic, step-wise const, and user-defined splines

Spline based interpolation/extrapolation

PDEs (Partial Differential Equations)

Solving Helmholtz, Poisson, and Laplace problems.

Optimization Solvers

Solvers for nonlinear least square problems with/without constraints

Support Functions

Memory management

Threading control

…

Conditional Numerical Reproducibility (CNR)

What causes a variation in results?

- With floating-point numbers, the order of computation matters!
- Associativity does not always hold … **(a+b)+c ≠ a+(b+c)**

$2^{-63} + 1 + -1 = \mathbf{2^{-63}}$	**(infinitely precise result)**
$(2^{-63} + 1) + -1 = \mathbf{0}$	**(correct IEEE double precision result)**
$2^{-63} + (1 + -1) = \mathbf{2^{-63}}$	**(correct IEEE double precision result)**

CNR run-time controls

MAXIMUM COMPATIBILITY

For consistent results …	Function Call mkl_cbwr_set(…)	Env. Variable MKL_CBWR =
on Intel or Intel-compatible CPUs supporting SSE2 instructions or later	MKL_CBWR_COMPATIBLE	COMPATIBLE
on Intel processors supporting SSE4.2 instructions or later	MKL_CBWR_SSE4_2	SSE4_2
on Intel processors supporting Intel® AVX or later	MKL_CBWR_AVX	AVX
on Intel processors supporting Intel® AVX2 or later	MKL_CBWR_AVX2	AVX2
on Intel processors supporting Intel® AVX512 or later	MKL_CBWR_AVX512	AVX512
from run-to-run (but not processor-to-processor)	MKL_CBWR_AUTO	AUTO

MAXIMUM PERFORMANCE

Inspector-Executor Sparse BLAS API

Inspect step – analyze matrix to choose best strategy

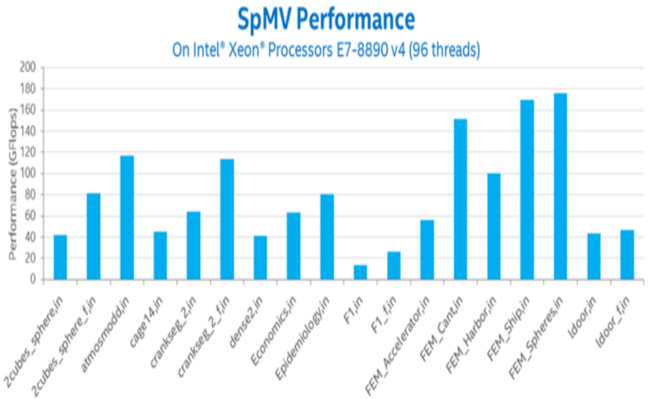
- **Computational kernels for portrait**
- **Balancing strategy for parallel execution**

Execute step – use analysis data to get better performance

- **Optimization applied to get better performance**
- **Level chosen based on expected number of iterations**

Compared to existing implementation new API provides

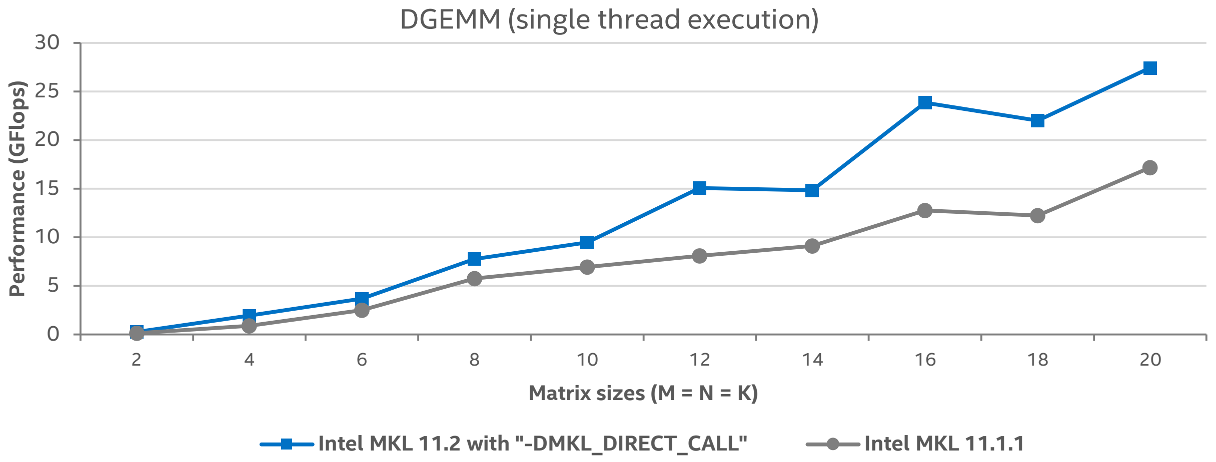
- Parallel triangular solver
- Improved sparse matrix by sparse matrix multiplication
- Both 0-based and 1-based indexing, row-major and column-major ordering
- Supporting CSR, CSC, COO, BSR formats
- Extended BSR support



Configuration: Versions: Intel® Math Kernel Library (Intel® MKL) v.2017; Hardware: CPU: Intel® Xeon E7-8890 v4,4 Twenty-four core CPU (60 MB smart cache,2.2 Ghz, 64 GB RAM; Operation System: RedHat* RHEL 7.2 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.
Benchmark Source: Intel Corporation

Improved Small Matrix Multiply Performance

(S/D/C/ZGEMM) - utilizes partial inlining, kernels, and reduced call/error checking overhead



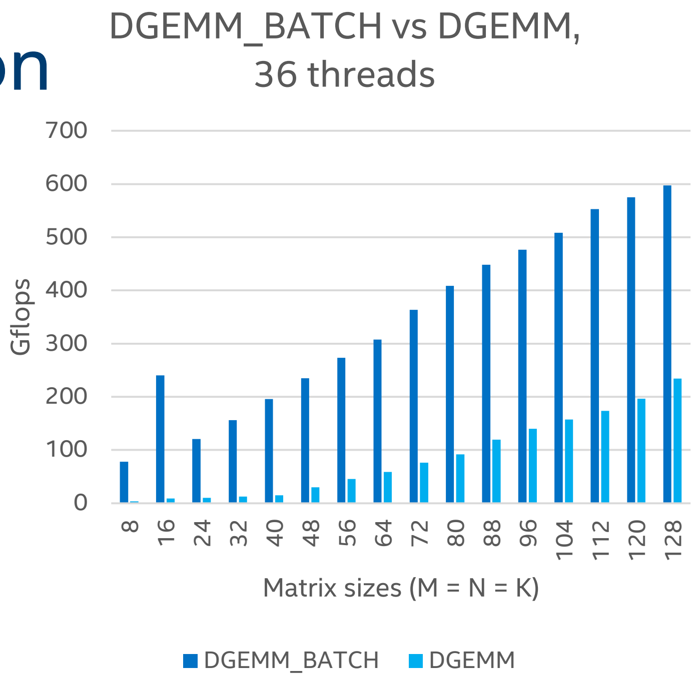
To use, define "MKL_DIRECT_CALL" or "MKL_DIRECT_CALL_SEQ" on the compile line

Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 11.1.1 and 11.2; Hardware of cluster nodes: Intel® Core™ i7-4770K, Quad-core CPU (8MB LLC, 3.50GHz), 32GB of RAM; Operating System: RHEL 6.1 GA x86_64; Benchmark Source: Intel Corporation.

Batch Matrix-Matrix Multiplication

Compute independent matrix-matrix multiplications (GEMMs) simultaneously with a single function call

- Supports all precisions of GEMM and GEMM3M
- Handles varying matrix sizes with a single function call
- Better utilizes multi/many-core processors for small sizes



Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 11.3.3; 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

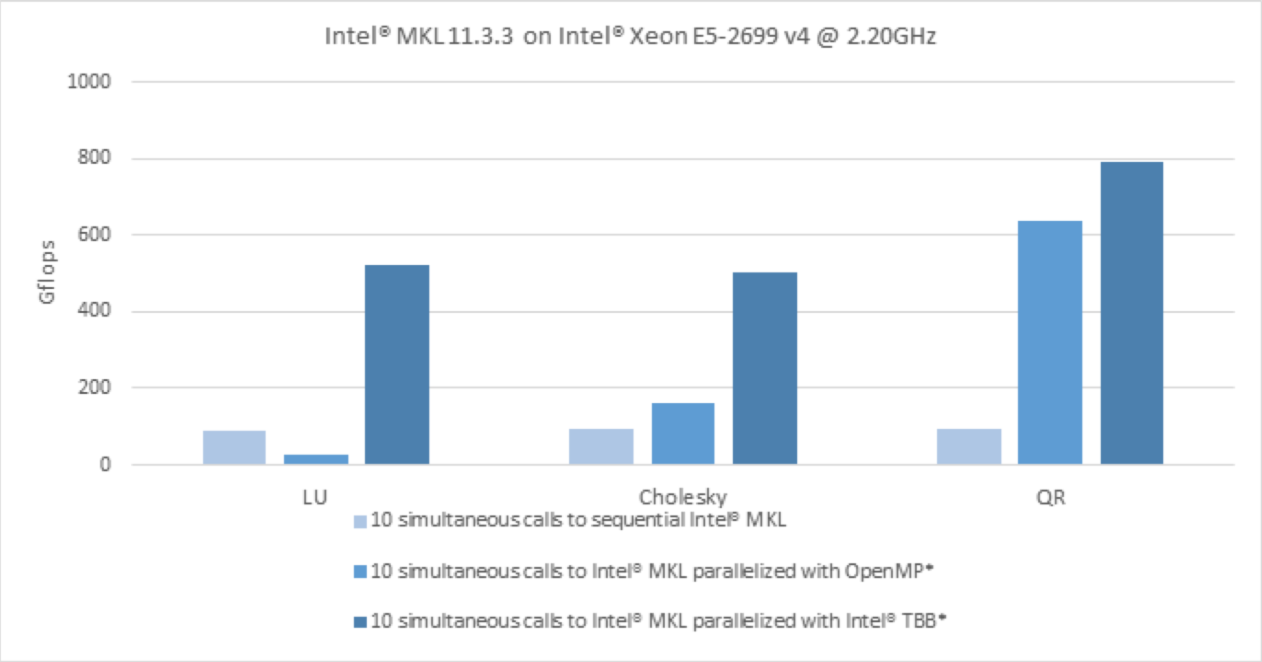
Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, 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® MKL 2017

- Optimized math functions to enable neural networks (CNN and DNN) for deep learning
- Improved ScaLAPACK performance for symmetric eigensolvers on HPC clusters
- New data fitting functions based on B-splines and monotonic splines
- Improved optimizations for newer Intel processors, especially Knight's Landing Xeon Phi
- Extended TBB threading layer support for all BLAS level-1 functions

MKL's TBB Threading Layer Option Delivers Huge Performance Gains in Busy Parallel Programs

- TBB threading is ideal for workloads when MKL runs in parallel with other threaded computation
- OpenMP threading is better when MKL can use entire threadpool



Intel® DAAL+ Intel® MKL = Complementary Big Data Libraries Solution

Intel MKL	Intel DAAL
C and Fortran API Primitive level	Python, Java & C++ API High-level
Processing of homogeneous data in single or double precision	Processing heterogeneous data (mix of integers and floating point), internal conversions are hidden in the library
Type of intermediate computations is defined by type of input data (in some library domains higher precision can be used)	Type of intermediate computations can be configured independently of the type of input data
Most of MKL supports batch computation mode only	3 computation modes: Batch, streaming and distributed
Cluster functionality uses MPI internally	Developer chooses communication method for distributed computation (e.g. Spark, MPI, etc.) Code samples provided.

"Initially, the Spark/Shark-based solution required 40 hours to compete a computation. Youku improved performance significantly by implementing Intel® Math Kernel Library (Intel® MKL) into its solution…After implementation of Intel MKL, Youku reduced the computation time to less than three hours."

Source: Youku Tudou Video Sharing Recommendation Case Study

Intel® MKL Summary

Intel MKL 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[1]

- Intel® Optimized MP LINPACK Benchmark
- Intel® Optimized Technology Preview for High Performance Conjugate Gradient Benchmark

[1] http://www.top500.org

Intel(R) MKL Resources

Intel® MKL website

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

Intel MKL forum

- https://software.intel.com/en-us/forums/intel-math-kernel-library

Intel® MKL benchmarks

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

Intel® MKL link line advisor

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

Legal Disclaimer & Optimization Notice

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