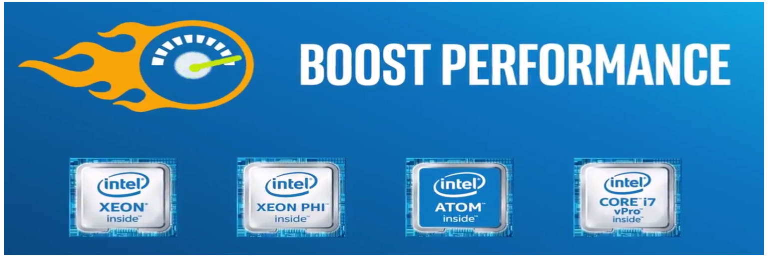
Intel® Math Kernel Library 2018 (Intel® MKL)

February, 2018

Intel® Math Kernel Library

Intel® MKL

- Speeds computations for scientific, engineering, financial and machine learning applications
- Provides key functionality for dense and sparse linear algebra (BLAS, LAPACK, PARDISO), FFTs, vector math, summary statistics, deep learning, splines and more
- Included in Intel® Parallel Studio XE and Intel® System Studio Suites
- Available at no cost and royalty free



- Optimized for single core vectorization and cache utilization
- Automatic parallelism for multi-core and many-core
- Scales from cores to clusters
- Great performance with minimal effort

Intel® MKL Optimized Mathematical Building Blocks

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

Fast Fourier Transforms
- Multidimensional
- FFTW* interfaces
- Cluster FFT

Vector Math
- Trigonometric
- Hyperbolic
- Exponential
- Log
- Power
- Root
- Vector RNGs

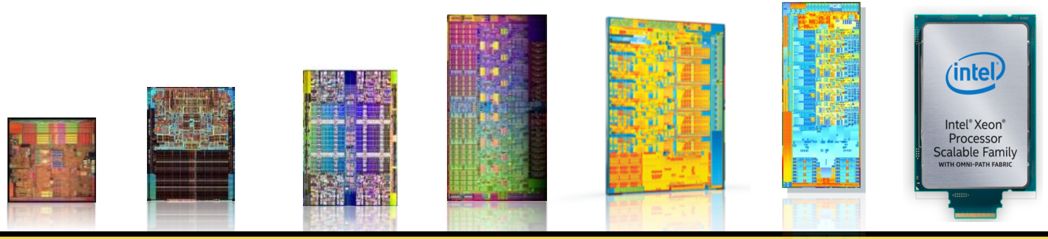
Deep Neural Networks
- Convolution
- Pooling
- Normalization
- ReLU
- Inner Product

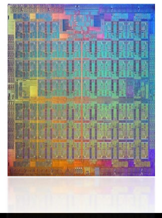
Summary Statistics
- Kurtosis
- Central moments
- Variation coefficient
- Order statistics and quantiles
- Min/max
- Variance-covariance
- Robust estimators

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

Automatic Dispatching to Tuned ISA-specific Code Paths

More cores → More Threads → Wider vectors





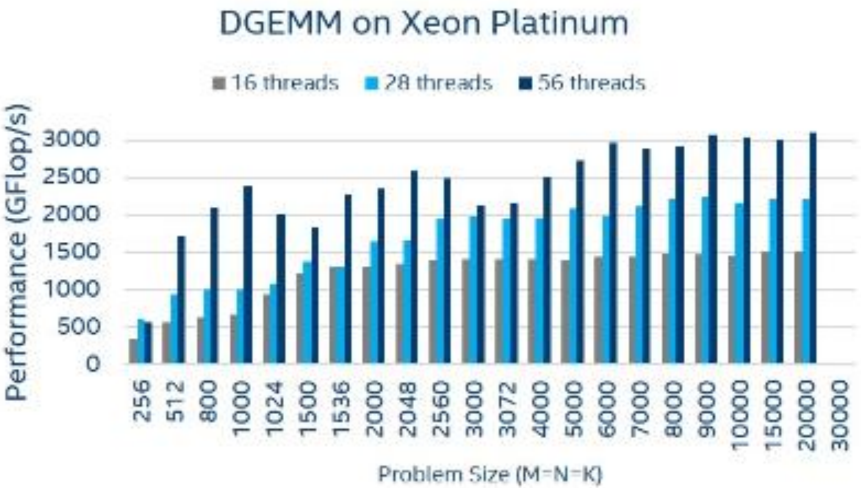
	Intel® Xeon® Processor 64-bit	Intel® Xeon® Processor 5100 series	Intel® Xeon® Processor 5500 series	Intel® Xeon® Processor 5600 series	Intel® Xeon® Processor E5-2600 v2 series	Intel® Xeon® Processor E5-2600 v3 series v4 series	Intel® Xeon® Scalable Processor[1]	Intel® Xeon Phi™ x200 Processor (KNL)
Up to Core(s)	1	2	4	6	12	18-22	28	72
Up to Threads	2	2	8	12	24	36-44	56	288
SIMD Width	128	128	128	128	256	256	512	512
Vector ISA	Intel® SSE3	Intel® SSE3	Intel® SSE4- 4.1	Intel® SSE 4.2	Intel® AVX	Intel® AVX2	Intel® AVX-512	Intel® AVX-512

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

Performance Benefits for the latest Intel Architectures

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

(formerly codenamed Skylake Server)



Configuration: Intel® Xeon® Platinum 8180, 2x28 cores, 2.5GHz, 38.5MB L3 cache, 376GB RAM, OS Ubuntu 16.04 LTS; Intel® Math Kernel Library (Intel® MKL) 2018. 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. 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 2018 New Features and Optimizations

BLAS and LAPACK	Compact BLAS and LAPACK functions Direct Call LAPACK Cholesky and QR factorizations LU factorization and Inverse without pivoting Aasen-based factorization and solve functions Bounded Bunch-Kaufman (Rook) pivoting factorizations
FFTs	Verbose Mode Support
Vector Math - 24 New Functions	v?Fmod, v?Remainder v?Powr, v?Exp2, v?Exp10 v?Cospi, v?Sinpi, v?Tanpi, and more

Intel® MKL BLAS (Basic Linear Algebra Subprograms)

De-facto Standard APIs since the 1980s	
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
BLAS-like Extensions	Direct Call, Batched, Packed and Compact
Reference Implementation	*http://netlib.org/blas/*

Intel® MKL LAPACK (Linear Algebra PACKage)

De-facto Standard APIs since the 1990s

1000s of Linear Algebra Functions	Matrix factorizations - LU, Cholesky, QR Solving systems of linear equations Condition number estimates Symmetric and non-symmetric eigenvalue problems Singular value decomposition and many more ...
Precisions Available	Real – Single and Double, Complex – Single and Double
Reference Implementation	*http://netlib.org/lapack/*

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 supported
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 DNN (Deep Neural Network) Functions

Highly optimized basic building blocks for DNNs	
Use cases	Inference and training Image recognition, semantic segmentation, object detection
Functions	Convolution, Inner Product Activation, Normalization, Pooling, Sum, Split/Concat, Data transformation
Applications	Supported in Tensorflow, MXNet, IntelCaffe and more
Open source version	*https://github.com/01org/mkl-dnn*

Intel® MKL Vector Math

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

Random Number Generators (RNGs)	Pseudorandom, quasi-random and non-deterministic random number generators with continuous and discrete distribution
Summary Statistics	Parallelized algorithms to compute basic statistical estimates for single and double precision multi-dimensional datasets
Convolution and Correlation	Linear convolution and correlation transforms for single and double precision real and complex data

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

Intel® MKL General Components

Sparse BLAS	NIST-like and inspector execute interfaces
Data Fitting	1D linear, quadratic, cubic, step-wise and user-defined splines, spline-based interpolation and extrapolation
Partial Differential Equations	Helmhotz, Poisson, and Laplace equations
Optimization	Trust-region solvers for nonlinear least square problems with and without constraints
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[1]	Intel® Distribution for LINPACK* Benchmark
	Intel® Optimized High Performance Conjugate Gradient Benchmark

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

Intel® 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/

BACK-UP

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

Compiling & Linking with Intel® MKL

In general:

- Include “mkl.h”
- Include “mkl_dfti.h” for FFT applications
- Using Intel® MKL Link Line Advisor to assist in compiling and linking applications with Intel® MKL

Compiling & Linking with Intel® MKL

Intel® Math Kernel Library (Intel® MKL) Link Line Advisor v4.7 Reset

Select Intel® product:	Intel(R) MKL 2018.0 ▼
Select OS:	Linux* ▼
Select usage model of Intel® Xeon Phi™ Coprocessor:	None ▼
Select compiler:	Intel(R) Fortran ▼
Select architecture:	Intel(R) 64 ▼
Select dynamic or static linking:	Dynamic ▼
Select interface layer:	64-bit integer ▼
Select threading layer:	OpenMP threading ▼
Select OpenMP library:	Intel(R) (libiomp5) ▼
Select cluster library:	☐ Cluster PARDISO (BLACS required) ☑ CDFT (BLACS required) ☐ ScaLAPACK (BLACS required) ☑ BLACS
Select MPI library:	Intel(R) MPI ▼
Select the Fortran 95 interfaces:	☑ BLAS95 ☑ LAPACK95
Link with Intel® MKL libraries explicitly:	☐

Compiling & Linking with Intel® MKL

Use this link line:

```
 ${MKLROOT}/lib/intel64/libmkl_blas95_ilp64.a
${MKLROOT}/lib/intel64/libmkl_lapack95_ilp64.a -L${MKLROOT}/lib/intel64 -
lmkl_cdft_core -lmkl_intel_ilp64 -lmkl_intel_thread -lmkl_core -
lmkl_blacs_intelmpi_ilp64 -liomp5 -lpthread -lm -ldl
```

Compiler options:

```
 -i8 -I${MKLROOT}/include/intel64/ilp64 -I${MKLROOT}/include
```

Some Performance Tips

- Use mkl_malloc and mkl_free for allocating and freeing aligned memory
- Maximize access to the local memory
- Use OMP_NUM_THREADS and KMP_AFFINITY to control threads
- For Apps that require high memory BW, allocate memory in MCDRAM
  - Numactl
  - Install memkind library
- More details can be found in the developer guide for Intel® MKL

Intel® MKL 11.0 – 2017 Noteworthy Enhancements

- Conditional Numerical Reproducibility (CNR)
- Intel Threading Building Blocks (TBB) Composability
- Intel® Optimized High Performance Conjugate Gradient (HPCG) Benchmark
- Small GEMM Enhancements (Direct Call) and Batch
- Sparse BLAS Inspector-Executor API
- Extended Cluster Support (MPI wrappers and macOS)
- Parallel Direct Sparse Solver for Clusters
- Extended Eigensolvers
- Deep Neural Networks Convolution, Normalization, Activation, and Pooling Primitives