Intel® Performance Libraries

Energy | Science & Research | Engineering Design | Financial Analytics | Signal Processing | Digital Content Creation

Powerful Mathematical Library

Intel® Math Kernel Library (Intel® MKL)

- Speeds math processing in scientific, engineering and financial applications
- Functionality for dense and sparse linear algebra (BLAS, LAPACK, PARDISO), FFTs, vector math, summary statistics and more
- Provides scientific programmers and domain scientists
  - Interfaces to de-facto standard APIs from C++, Fortran, C#, Python and more
  - Support for Linux*, Windows* and OS X* operating systems
  - The ability to extract great parallel performance with minimal effort
- Unleash the performance of Intel® Core, Intel® Xeon and Intel® Xeon Phi™ product families
  - Optimized for single core vectorization and cache utilization
  - Coupled with automatic OpenMP*-based parallelism for multi-core, manycore and coprocessors
  - Scales to PetaFlop (1015 floating-point operations/second) clusters and beyond
- Included in Intel® Parallel Studio XE Suites

**http://www.top500.org

Used on the World's Fastest Supercomputers**

Intel® Math Kernel Library is a Computational Math Library

Mathematical problems arise in many scientific disciplines

Energy

Signal Processing

Financial Analytics

Engineering Design

Digital Content Creation

Science & Research

These scientific applications areas typically involve mathematics …

- Differential equations
- Linear algebra
- Fourier transforms
- Statistics

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

Intel® MKL can help solve your computational challenges

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

Vector Math

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

Vector RNGs

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

Summary Statistics

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

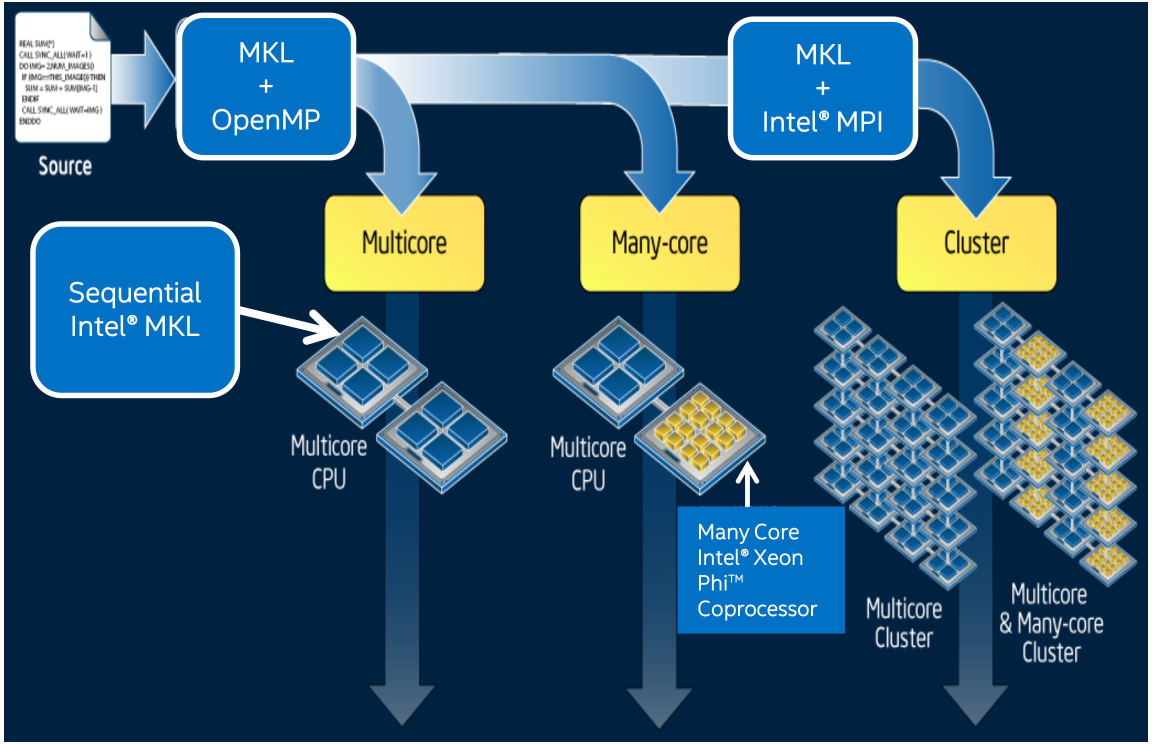
And More

- Splines
- Interpolation
- Trust Region
- Fast Poisson Solver

Intel® Math Kernel Library is a Performance Library

We go to extremes to get the most performance from the available resources

- Core: vectorization, prefetching, cache utilization
- Multicore (processor/socket) level parallelization
- Multi-socket (node) level parallelization
  - Cluster scaling
  - Data locality is key

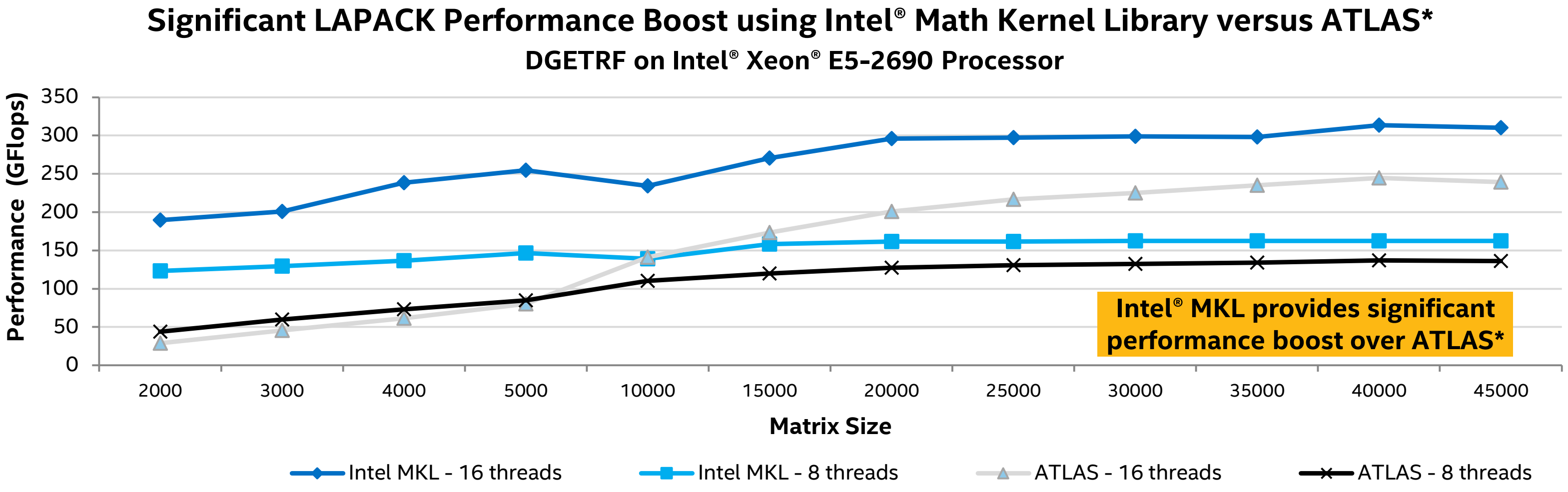


Automatic scaling from multicore to manycore and beyond

Optimization Notice

Immediate Performance Benefit to Applications

Intel® Math Kernel Library



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

Notable New Features

Intel® Math Kernel Library

As always, optimized for the latest Intel® Xeon® processors

Intel® Xeon Phi™ Coprocessor support

- native
- compiler assisted offload
- automatic offload

Conditional Numerical Reproducibility

- consistent results from run-to-run
- controls provided to ensure consistency across processor families

LAPACKE Interfaces

- extends the de-factor Fortran LAPACK APIs
- callable from C and with support for row-major storage

Extended Eigensolvers

- solves standard and generalized sparse symmetric/hermitian Eigenproblems
- based on and compatible with FEAST* http://www.ecs.umass.edu/~polizzi/feast/

Available since Intel® MKL 11.0

Intel® Xeon Phi™ Coprocessor Support

Intel® Math Kernel Library (Intel® MKL)

Automatic Offload	• No code changes required • Automatically uses both host and target • Transparent data transfer and execution management
Compiler Assisted Offload	• Explicit controls for data transfer and remote execution using compiler pragma offload • Can be used together with Automatic Offload
Native Execution	• Uses the coprocessors as independent nodes • Input data and binaries are copied to targets in advance

Conditional Numerical Reproducibility

Intel® Math Kernel Library (Intel® MKL)

The cause for a variation in results

- With floating-point numbers, the order of computation matters!
- Remember that associativity does not always hold, that is, (a+b)+c ≠ a+(b+c)
  - 2-63 + 1 + -1= 2-63 (infinitely precise result)
  - (2-63 + 1) + -1= 0 (correct IEEE double precision result)
  - 2-63 + (1 + -1) = 2-63 (correct IEEE double precision result)

Intel® MKL 11.0 brought fixed code path options for aligned data and deterministic scheduling

- Example: attain identical results on every Intel processor supporting AVX instructions or later
  - function call: mkl_cbwr_set(MKL_CBWR_AVX)
  - environment variable: set MKL_CBWR_BRANCH=“AVX”

Intel MKL 11.1 removed the data alignment restriction

New Features

Intel® Math Kernel Library (Intel® MKL) 11.2

- The Cluster Direct Sparse Solver extends the capabilities of Intel® MKL PARDISO, enabling users to solve large distributed sparse systems of equations on clusters
- Small Matrix Multiply performance improvements deliver performance boosts of 1.3 times on average for small problem sizes (less than 20x20)
- Support for the next generation Intel® Advanced Vector Extensions 512 (Intel® AVX-512) instruction set with optimizations in BLAS, DFT and VML
- A Intel MKL cookbook that provides step-by-step recipes to solve common mathematical problems using existing library functions
- Verbose mode allows users to understand how Intel MKL is used in your program
  - provides detailed Intel MKL versioning information
  - identifies the library functions called and the parameters passed to them
  - returns the amount of time spent in each function call

Futures

Intel® Math Kernel Library (Intel® MKL)

Technology Previews Available

Intel® MKL Sparse Matrix Vector Multiply Prototype Package

- improved sparse matrix-vector multiply for Intel® Xeon Phi™ coprocessor
- 2-stage API for sparse BLAS (analyze and execute)
- support for a new Ellpack Sparse Block (ESB) format

Intel® Optimized Technology Preview for High Performance Conjugate Gradient Benchmark

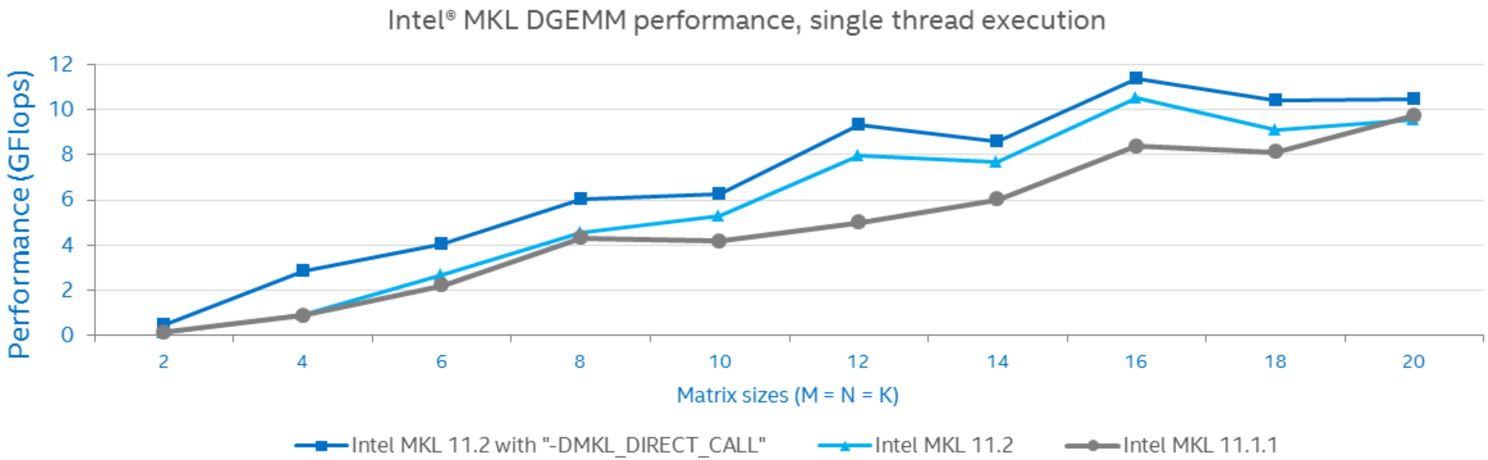
- proposed to supplement the current High Performance Linpack Benchmark
- designed to be more representative of common application workloads

Packages available – contact intel.mkl@intel.com

We seek your insights into defining new Intel MKL features

DGEMM Small Matrix Improvements

Intel® Math Kernel Library 11.2



Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 11.1.1 and 11.2; Hardware of cluster nodes: Intel® Xeon® Processor E5-2697v2, 2 Twelve-core CPUs (30MB LLC, 2.7GHz), 64GB of RAM; Operating System: RHEL 6.1 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 .

Future Directions: SpMV Prototype Format Package

Intel® Math Kernel Library (Intel® MKL)

Improved sparse matrix-vector multiply for Intel® Xeon Phi™

- 2-stage API for Sparse BLAS (analyze and execute)
- Ellpack Sparse Block (ESB) format

Package available – contact intel.mkl@intel.com

Seek industry feedback on API and performance