“Intel MKL is indispensable for any high-performance computer user on x86 platforms.”
Prof. Jack Dongarra, Innovative Computing Lab, University of Tennessee, Knoxville
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
Legal Disclaimer

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.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, reference www.intel.com/software/products.


*Other names and brands may be claimed as the property of others.

Copyright © 2011. Intel Corporation.

http://intel.com/software/products
Agenda

• What is Intel® MKL?
• Intel® MKL components
• Performance advantages
• Basic product info:
  – Standalone and bundled products
  – Environment and language interfaces
  – How to link?
• Intel® MKL Who use Intel® MKL
• The latest release and more info
What is a math library?

Start with your problem in a scientific discipline

These problems might 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 is industry’s leading math library

**Linear Algebra**
- BLAS
- LAPACK
- Sparse solvers
- ScaLAPACK

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

**Vector Math**
- Trigonometric
- Hyperbolic
- Exponential, Logarithmic
- Power / Root
- Rounding

**Random Number Generators**
- Congruential
- Recursive
- Wichmann-Hill
- Mersenne Twister
- Sobol
- Niederreiter
- RDRAND-based

**Summary Statistics**
- Kurtosis
- Variation coefficient
- Quantiles, order statistics
- Min/max
- Variance-covariance
- ...

**Others**
- Data fitting (splines)
- PDE solvers
- Nonlinear optimization
- Convolution, Correlation

Diagram shows Intel MKL connected to various computing architectures including:
- Multicore CPU
- Many-core
- Clusters with Multicore and Many-core

Source: Clusters with Multicore and Many-core
# Intel® MKL BLAS

## Basic Linear Algebra Subroutines (BLAS)

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Level 1:</strong></td>
<td>Vector operations</td>
<td>Dot products, swap, min, max, scaling, rotation, etc.</td>
</tr>
<tr>
<td><strong>Level 2:</strong></td>
<td>Matrix-vector operations</td>
<td>Matrix-vector products, rank-1/rank-2 updates, triangular solvers, etc.</td>
</tr>
<tr>
<td><strong>Level 3:</strong></td>
<td>Matrix-matrix operations</td>
<td>Matrix-matrix products, rank-k/rank-2k updates, triangular solvers, etc.</td>
</tr>
<tr>
<td>Sparse BLAS</td>
<td></td>
<td>BLAS level 1, 2, &amp; 3 for sparse vectors and matrices</td>
</tr>
</tbody>
</table>

## Matrix storage schemes

- BLAS: Full, packed, and banded storage
- Sparse BLAS: CSR, CSC, coordinate, diagonal, skyline, BSR, etc.
## Intel® MKL LAPACK

**De facto industry standard API**

<table>
<thead>
<tr>
<th>Linear Algebra Package (LAPACK)</th>
<th>Solving systems of linear equations, factoring and inverting matrices.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Solving linear least squares problems, Eigenvalues, singular value problems, and Sylvester’s equations.</td>
</tr>
<tr>
<td></td>
<td>Many auxiliary and utility functions.</td>
</tr>
<tr>
<td></td>
<td>LAPACK driver routines: Combines several routines in one call to solve a particular problem.</td>
</tr>
</tbody>
</table>

**scaLAPACK**

LAPACK for distributed memory architectures.

- Using MPI, BLACS (basic linear algebra communication subprograms), and BLAS.
Intel® MKL FFT Routines

- Mixed radix, multidimensional FFTs
- Supports user-defined scaling and transform sign
- Multiple transforms in a single call
- Supports data stride in input
- Supports FFTW* interfaces
- Cluster FFTs
  - FFTs for distributed memory systems
  - Works with MPI
  - FFTW* support

De facto industry standard API
# Intel® MKL Sparse Solvers

<table>
<thead>
<tr>
<th>Solver</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DSS</strong> – Direct Sparse Solver Interface for PARDISO</td>
<td>An alternative, simplified interface to PARDISO.</td>
</tr>
</tbody>
</table>
Intel® MKL Vector Math Library (VML)

- Collection of vector math functions
- Real/Complex
- Double precision (DP), Single precision (SP)
- 3 accuracy modes
  - **High Accuracy**, HA (correct rounding in >99% cases, behave according to C99; slowest, default mode)
  - **Low Accuracy**, LA (≤2 lsb incorrect, behave according to C99; 30-50% faster than HA)
  - **Enhanced Performance**, EP (~1/2 incorrect bits, is not guaranteed on entire domain; 30-50% faster than LA)

### Real functions

<table>
<thead>
<tr>
<th>Arithmetic</th>
<th>Power and Root</th>
<th>Exponential &amp; Logarithmic</th>
<th>Trigonometric</th>
<th>Hyperbolic</th>
<th>Special</th>
<th>Rounding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Inv</td>
<td>Exp</td>
<td>Sin</td>
<td>Sinh</td>
<td>Erf</td>
<td>Floor</td>
</tr>
<tr>
<td>Sub</td>
<td>Div</td>
<td>Expm1</td>
<td>Cos</td>
<td>Cosh</td>
<td>Erfc</td>
<td>Ceil</td>
</tr>
<tr>
<td>Sqr</td>
<td>Sqrt</td>
<td>Ln</td>
<td>Tan</td>
<td>Tanh</td>
<td>ErfInv</td>
<td>Trunc</td>
</tr>
<tr>
<td>Mul</td>
<td>InvSqrt</td>
<td>Log10</td>
<td>Asin</td>
<td>Asinh</td>
<td>ErfcInv</td>
<td>Round</td>
</tr>
<tr>
<td>Abs</td>
<td>Cbrt</td>
<td>Log1p</td>
<td>Acos</td>
<td>Acosh</td>
<td>CdfNorm</td>
<td>NearbyInt</td>
</tr>
<tr>
<td>LinearFrac</td>
<td>InvCbrt</td>
<td>Atan</td>
<td>Atanh</td>
<td>CdfNormInv</td>
<td>Rint</td>
<td></td>
</tr>
<tr>
<td>Pow2o3</td>
<td></td>
<td>Atan2</td>
<td></td>
<td>LGamma</td>
<td>Modf</td>
<td></td>
</tr>
<tr>
<td>Pow3o2</td>
<td></td>
<td>SinCos</td>
<td></td>
<td>TGamma</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Powx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hypot</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Complex functions

<table>
<thead>
<tr>
<th>Arithmetic</th>
<th>Power and Root</th>
<th>Exponential &amp; Logarithmic</th>
<th>Trigonometric</th>
<th>Hyperbolic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Div</td>
<td>Exp</td>
<td>Sin</td>
<td>Sinh</td>
</tr>
<tr>
<td>Sub</td>
<td>Sqrt</td>
<td>Ln</td>
<td>Cos</td>
<td>Cosh</td>
</tr>
<tr>
<td>Mul</td>
<td>Pow</td>
<td>Log10</td>
<td>Tan</td>
<td>Tanh</td>
</tr>
<tr>
<td>MulByConj</td>
<td>Powx</td>
<td></td>
<td>Asin</td>
<td>Asinh</td>
</tr>
<tr>
<td>Conj</td>
<td></td>
<td></td>
<td>Acos</td>
<td>Acosh</td>
</tr>
<tr>
<td>Abs</td>
<td></td>
<td></td>
<td>Atan</td>
<td>Atanh</td>
</tr>
<tr>
<td>Arg</td>
<td></td>
<td></td>
<td>CIS</td>
<td></td>
</tr>
</tbody>
</table>
### Intel® Vector Statistical Library (VSL)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Number Generators (RNGs)</td>
<td>Psuedo-random, quasi-random, and non-deterministic generators</td>
</tr>
<tr>
<td></td>
<td>Continuous and discrete distributions of various common distribution types</td>
</tr>
<tr>
<td>Summary Statistics (SS)</td>
<td>Parallelized algorithms for computation of statistical estimates for raw multi-dimensional datasets.</td>
</tr>
<tr>
<td>Convolution/correlation</td>
<td>A set of routines intended to perform linear convolution and correlation transformations for single and double precision real and complex data.</td>
</tr>
</tbody>
</table>
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
- ...
Why is Intel® MKL faster?

Optimization for maximum speed.

Resource limited optimization – exhaust one or more resource of system:

- **CPU**: Vectorization, Register use, FP units.
- **Cache**: Keep data in cache as long as possible; deal with cache interleaving.
- **TLB**: Maximally use data on each page.
- **Memory**: Minimally access memory.
- **Computer**: Use all the processor cores available using threading.
- **System**: Use all the nodes available.

Leverage on Intel® instruction set architecture performance features:

- Intel® SSE2, Intel® SSE3, Intel® SSSE3, Intel® SSE4, Intel® AVX, Intel® AVX2, ...
# How is Intel® MKL Parallelized

<table>
<thead>
<tr>
<th>Domain</th>
<th>SIMD</th>
<th>Open MP</th>
<th>MPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLAS 1, 2, 3</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>FFTs</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>LAPACK (dense LA solvers)</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(relies on BLAS 3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ScaLAPACK (cluster dense LA solvers)</td>
<td>X</td>
<td>(hybrid)</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>(hybrid)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PARDISO (sparse solver)</td>
<td>X</td>
<td>X</td>
<td>Coming soon ...</td>
</tr>
<tr>
<td></td>
<td>(relies on BLAS 3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VML/VSL</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Cluster FFT</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
Matrix Multiply Performance (Intel® MKL vs. ATLAS*)

Significant Performance Boost using Intel® Math Kernel Library versus ATLAS*
DGEMM on both Intel® Xeon® and Intel® Core™ Processors

See the latest performance charts for MKL key components:
## Intel® MKL Environment

### Language Support

<table>
<thead>
<tr>
<th>Domain</th>
<th>Fortran 77</th>
<th>Fortran 95/99</th>
<th>C/C++</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLAS</td>
<td>X</td>
<td>X</td>
<td>Via CBLAS</td>
</tr>
<tr>
<td>Sparse BLAS Level 1</td>
<td>X</td>
<td>X</td>
<td>Via CBLAS</td>
</tr>
<tr>
<td>Sparse BLAS level 1&amp;2</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>LAPACK</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ScaLAPACK</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PARDISO</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>DSS &amp; ISS</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>VML/VSL/DF</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FFT/Cluster FFT</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>PDEs</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Optimization (TR) Solvers</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SSL</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Intel® MKL a Key Ingredient of the Technical Computing SW Product Offerings

And is also available standalone

How to Link with Intel® MKL

Static or dynamic linking

Custom Dynamic linking
• Links only with the components that are needed in your applications.

Single dynamic linking
• Links only with a single MKL runtime library.
Who Use Intel® MKL

Open Source Projects
- NumPy/SciPy
- PETSc
- Trilinos
- Eigen
- Gromacs
- Octave
- WRF

Library Vendors
- nag®
- Rogue Wave
- CenterSpace®

Major ISVs
- The MathWorks
- WOLFRAM
- SAS
- REvolution

Strategic End User Accounts from segments:
- Financial Segment
- Energy
- Manufacturing
- Academic/Research
- Animation
- Life Sciences

A Very Good Kitty, Indeed
DreamWorks Animation's Puss in Boots Uses Intel® Math Kernel Library to Help Create Dazzling Special Effects
Top Features in Latest Release

Support for Intel® Xeon Phi™ coprocessor:

- Host: Linux* OS or Windows* OS
- Three usage models:
  - Automatic Offload
  - Compiler Assisted Offload
  - Native Execution

Conditional Numerical Reproducibility (CNR):

- Balance reproducibility and performance.
- Run-to-run, and processor-to-processor numerical reproducibility
Intel® MKL Supports for Intel® Xeon Phi™ Coprocessors

• Intel® MKL 11.0 and above supports the Intel® Xeon Phi™ coprocessors.

• Heterogeneous computing
  - Takes advantage of both multicore host and many-core coprocessors.

• Optimized for wider (512-bit) SIMD instructions and threaded for many cores.

• All Intel MKL functions are supported:
  - But optimized at different levels.

Pairing highly parallel software with highly parallel hardware.
Highly Optimized Functions

BLAS Level 3, and much of Level 1 & 2

Sparse BLAS: ?CSRMV, ?CSRMM

Some important LAPACK routines (LU, QR, Cholesky)

Fast Fourier transforms

Vector Math Library

Random number generators in the Vector Statistical Library

Broader functionality to be optimized in future update releases.
Xeon vs. KNC vs. Automatic Offload

LU Factorization Performance using Intel® Math Kernel Library on Intel® Xeon Phi™ Coprocessors 7120P and Intel® Xeon® Processor E5-2697 v2

Configuration Info - Software Versions: Intel® Math Kernel Library (Intel® MKL) 11.1, Intel® Manycore Platform Software Stack (MPSS) 2.1.6720-15; Hardware: Intel® Xeon® Processor E5-2697 v2, 2 Twelve-Core CPUs (30MB LLC, 2.7GHz), 32GB DDR3 RAM (1333MHz); Intel® Xeon Phi™ Coprocessor 7120P, 61 cores (30.5MB total cache, 1.23GHz), 16GB DDR5 Memory; 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, September 2013.

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

Copyright © 2012, Intel Corporation. All rights reserved.
*Other brands and names are the property of their respective owners.
Improvement to the CNR Feature

Before

- Memory alignment
  - Align memory — try Intel MKL memory allocation functions
  - 64-byte alignment for processors in the next few years

- Number of threads
  - Set the number of threads to a constant number
  - Use sequential libraries

- Deterministic task scheduling
  - Ensures that FP operations occur in order to ensure reproducible results

- Code path control
  - Maintains consistent code paths across processors
  - Will often mean lower performance on the latest processors

After

- Pre-requisite: Fixed number of threads
  - Set the number of threads to a constant number (MKL_NUM_THREADS)
  - Use sequential libraries

- Deterministic task scheduling
  - Ensures that FP operations occur in order to ensure reproducible results

- Code path control
  - Maintains consistent code paths across processors
  - Will often mean lower performance on the latest processors

Data alignment is no longer a requirement for getting numerical reproducibility.
But aligning input data is still a good idea for getting better performance.
Intel® Composer XE 2015 Beta (Intel® MKL 11.2 Beta) New Features

- Better Matrix Multiply performance for small matrices
  - Reduced call/error checking overhead for S/DGEMM calls

- Verbose mode for debug/identifying call characteristics

- Improved performance for SVD and symmetric eigensolvers

- Intel MKL Cluster PARDISO* (direct sparse solver)
  - Enjoy the features of SMP PARDISO* on a cluster
  - Competitive with the MUMPS* offering

- Automatic offload batch FFT support for Xeon Phi

- AVX-512 support for early KNL enabling

- Optimizations for Intel® Atom processors

- Android* support
More Information about Intel® MKL

Product web site:
- Performance charts
- Documentation
- User forum

Latest release: Intel® MKL 11.1 update 2
Extra
Intel’s Numeric String Conversion Functions

- Converting between ASCII strings and C floating point data types.

- Available since Intel® C++ Compiler 14.0:
  - User code links with `libistrconv`.
  - Provides functionality similar to GLIBC functions: snprintf and strtof/strtod.
  - Provides optimized performance.
libistrconv Function List

- Convert floating-point numbers to strings:
  
  int __IML_float_to_string(char * str, size_t n, int prec, float x);
  int __IML_double_to_string(char * str, size_t n, int prec, double x);

- Convert strings to floating-point numbers:
  
  float __IML_string_to_float(const char * nptr, char ** endptr);
  double __IML_string_to_double(const char * nptr, char ** endptr);