Intel® Math Kernel Library
(Intel® MKL) 10.1

In-Depth
## Contents

Intel® Math Kernel Library (Intel® MKL) 10.1: In-Depth

- **Intel® Math Kernel Library (Intel® MKL) 10.1**: 4
- **Highlights**: 4
- **Features**: 4
  - Multicore ready: 4
  - Automatic runtime processor detection: 4
  - Support for C and Fortran interfaces: 4
  - Support for all Intel® processors in one package: 4
  - Royalty-free distribution rights: 4
- **New in Intel MKL 10.1**: 4
  - Computational Layer: 5
  - PARDISO Direct Sparse Solver: 5
  - Sparse BLAS: 5
  - LAPACK: 5
  - Discrete Fourier Transform Interface (DFTI): 5
  - Iterative Solver Preconditioner: 5
  - Vector Math Functions: 5
  - User's Guide: 5
- **Performance Improvements in Intel MKL 10.1**: 5
- **Performance Improvements in Version 10.0**: 6
  - BLAS: 6
  - LAPACK: 6
  - FFTs: 6
  - VML/VSL: 6
- **Functionality**: 6
  - Linear Algebra: BLAS and LAPACK: 6
  - BLAS: 6
  - Sparse BLAS: 6
  - LAPACK: 7
  - BLAS and LAPACK Performance: 7
  - Linear Algebra: ScalAPACK: 8
  - ScalAPACK Performance*: 8
  - Raw Performance: 8
  - Block Size Robustness: 9
  - Linear Algebra: Sparse Solvers: 10
- **References**: 10
- **PARDISO*: Parallel Direct Sparse Solver**: 10
  - New Out-of-Core Support!: 10
- **Iterative Solvers**: 11
  - FGMRES Solver: 11
  - Conjugate Gradient Solver: 11
  - ILU0/ILUT Preconditioners: 11
  - Sparse BLAS: 11
- **References**: 11
  - Fast Fourier Transforms (FFT): 12
Interfaces .............................................. 12
Fortran and C ............................................. 12
Modern FFT Interface ................................. 12
FFTW Interface ......................................... 12
Performance (Shared Memory Parallel) .......... 12
1-D Transform ........................................... 12
2-D Transforms ......................................... 13
3-D Transforms ........................................... 13
Performance (Distributed Memory) .............. 14
References .................................................. 15
Vector Math Library ................................. 15
Performance ............................................. 15
Accuracy Modes ......................................... 16
References .................................................. 16
Vector Statistical Library .......................... 16
Convolution/Correlation .............................. 16
Random Numbers ....................................... 16
Random Number Generator Performance ........ 17
Comparison of Intel® MKL to Alternative Libraries ...... 17
Basic Random-Number Generators (BRNGs) ....... 17
Distribution Generator Types .................... 17
References .................................................. 19
LINPACK Benchmark .................................... 20
Ease of Use .............................................. 20
Performance ............................................. 21
Downloads .............................................. 21
Questions/Comments? .................................. 21
References .................................................. 22
Compatibility ............................................ 22
Operating Systems ..................................... 22
Development Environments ....................... 22
Processors .............................................. 22
Technical Support ...................................... 22
Intel® Premier Support .............................. 22
User Forum ............................................. 22
References .................................................. 22
Intel® Math Kernel Library (Intel® MKL) 10.1

Intel® Math Kernel Library (Intel® MKL) 10.1 offers highly optimized, extensively threaded math routines for scientific, engineering, and financial applications that require maximum performance.

Intel MKL is available with the Intel® C++ and Fortran Compilers Professional Editions and Intel® Cluster Toolkit, as well as a standalone product.

Intel MKL provides high performance, future proofing for applications and productivity for developers. Intel MKL is extremely optimized for current multicore x86 platforms and will continue to be optimized for future platforms to ensure applications benefit seamlessly from the latest architecture enhancements.


Highlights
- Version 10.1 is now available. See below for a list of the new features and performance improvements.
- Intel MKL 10.1 now optimized for Intel® Core™ i7 processors (previously codenamed Nehalem).
- See the system requirements for a full list of supported operating systems, compilers, and processors.
- Check out WhatIf.intel.com for interesting new technologies related to Intel MKL.

Features

Outstanding performance on Intel® processors
Achieve leadership performance from the math library that is highly optimized for, Intel® Xeon®, Intel® Core™, Intel® Itanium®, and Intel® Pentium® 4 processor-based systems. Special attention has been paid to optimizing multi-threaded performance for the Intel® Xeon Quad-core processors and the new Intel® Core™ i7 Quad-Core processors. Intel MKL strives for performance, competitive with that of other math software packages on non-Intel processors.

Multicore ready
- Excellent scaling on multicore and multiprocessor systems: Use the built-in parallelism of Intel MKL to automatically obtain excellent scaling on multicore and multiprocessors including Intel® Xeon® 7400 and the latest dual and quad-core systems. Intel MKL BLAS, Fast Fourier transforms, and Vector Math, among many other routines are threaded using OpenMP®.
- Thread-Safety: All Intel MKL functions are thread-safe. A non-threaded sequential version of Intel MKL is also provided.

Automatic runtime processor detection
A runtime check is performed so that processor-specific optimized code is executed, ensuring that your application achieves optimal performance on whatever system it is executing on.

Support for C and Fortran interfaces
Intel MKL includes both C and Fortran interfaces, unlike some alternative math libraries that require you to purchase multiple products.

Support for all Intel® processors in one package
Intel MKL includes support for Intel® Xeon®, Intel® Core™, Intel® Pentium 4, Intel® Itanium® architectures in a single package. Alternative math libraries require you to purchase multiple products for all supported processors.

Royalty-free distribution rights
Redistribute unlimited copies of the Intel MKL runtime libraries with your software.

New in Intel MKL 10.1
In this release of Intel Math Kernel Library (Intel MKL 10.1) provides optimized multi-threaded performance for the newest Intel® processors (Intel® Xeon® 7400 series, Intel® Core™). Intel MKL 10.0 introduced a new “layered” architecture to better support the varied usage models of our users as well as merged the standard and cluster editions so there is a single comprehensive package.

Optimizations for the new Intel® Xeon® and Intel® Core™ Processors. For more information see section “Performance Improvements in Version 10.1” below.

“Layered” Architecture introduced in Intel MKL 10.0
Intel MKL 10.0 introduced a re-architected product to provide multiple layers so that the base Intel MKL package supports numerous configurations of interfaces, compilers, and processors in a single package. Many other library vendors have specific versions that must be first found, downloaded, installed, and tested depending on the particular configuration of your development environment. This new Intel MKL architecture is intended to provide maximum support for our
varied customers' needs, while minimizing the effort it takes to obtain and utilize the great performance of Intel MKL. For more information, please refer to the "Using Intel MKL Parallelism" section of the Intel MKL User's Guide.

Computational Layer
This layer forms the heart of Intel MKL. A runtime check is performed so that processor-specific optimized code is executed. Users can build custom shared objects to include only the specific code needed and thus reduce the size of this layer if size is an issue.

PARDISO Direct Sparse Solver
- Out-of-core memory implementation for solving larger problems on SMP systems
- Support of separate backward/forward substitution for DSS/PARDISO.
- A new parameter for turning off iterative refinement for DSS interface.
- A new parameter for checking sparse matrix structure has been introduced for PARDISO interface.
- The sparse solver functionality now integrated into the core math library and it is no longer necessary to link a separate solver library.
- The sparse solver functionality can now be linked dynamically.

Sparse BLAS
- Added routines for computing the sum and product of two sparse matrices stored in compressed sparse row format
- Adder routines for converting between different sparse matrix formats.
- Added support for all data types (single precision, complex and double complex).
- Added sparse 0-based indexing.
- Added single precision support added.
- Threaded Level-3 Sparse BLAS triangular solvers.

LAPACK
- The capability to track and/or interrupt the progress of lengthy LAPACK computations has been added via a callback function mechanism. A function called mkl_progress can be defined in a user application, which will be called regularly from a subset of the MKL LAPACK routines. Refer to the specific function descriptions to see which LAPACK functions support the feature.

Discrete Fourier Transform Interface (DFTI)
- Added the DftiCopyDescriptor function for convenience when using the FFTs.
- The size of statically linked executables calling DFTI has been reduced significantly.
- Complex storage is now available for real-to-real transforms.

Iterative Solver Preconditioner
- ILUT accelerator/preconditioner for the Intel MKL RCI iterative solvers

Vector Math Functions
- New Mul, Conj, MulbyConj, CIS, Abs functions
- New “Enhanced Performance” mode EP Mode is for applications where math function inaccuracies don't dominate parameter inaccuracies (e.g. Monte Carlo simulations and Media applications)
- All VML functions are now threaded
- Optimized versions of the Cumulative Normal Distribution (CdfNorm), its inverse (CdfNormInv), and the inverse complementary error function (ErfcInv) have been added to the Vector Math Library.

User's Guide
- We have greatly improved our Intel MKL User's Guide. It is an indispensable tool for working with Intel MKL. Visit the Documentation page to download it or view it online.
- Compiler Support: Support for new compilers including the new Intel® compilers 11.0 and PGI* compilers.

Performance Improvements in Intel MKL 10.1
We improved performance in all areas of the library. Below are some specific measured performance gains. Performance improvements are illustrated for each Intel MKL product domain (BLAS/LAPACK, FFT, VML, VSL, etc.)

- BLAS
  - 32-bit improvements
    - Up to 50% improvement for (Z,C)GEMM on Quad-Core Intel® Xeon® processor 5300 series
    - 10% improvement for all (D,S,Z,C)GEMM code on Quad-Core Intel® Xeon® processor 5400 series
  - 64-bit improvements
    - 50% improvement for SGEMM on the Intel® Core™ i7 processor.
    - 30% improvement for right-side cases of DTRSM on the Intel® Core™ i7 processor.
- Direct sparse solver (DSS/PARDISO):
  - 35% performance improvement on average for out-of-core PARDISO.
• VML and VSL
• Optimizations on the Intel® Core™ i7 processor:
  - Up to 17% improvement for the following VML functions: Asin, Asinh, Acos, Acoh, Atan, Cbrt, Cls, Cos, Cosh, Conj, Div, ErfInv, Exp, Hypot, Inv, InvCbrt, InvSqrt, Ln, Log10, MulByConj, Sin, SinCos, Sinh, Sqrt, Tanh.
  - Up to 67% improvement for uniform random number generation.
  - Up to 10% improvement for VSL distribution generators based on Wichmann-Hill, Sobol, and Niederreiter BRNGs (64-bit only).

Performance Improvements in Version 10.0

BLAS
• Threading of DGEMM was improved for small and middle sizes - outer product sizes by 10%, square sizes by 80%
• DGEMM/SGEMM Large square and large outer product sizes were improved by 4-5% on 1 thread and 10-15% on 8 threads
• DTRSM, DTRMM, and DSYRK were improved by 5-30%
• Other level 3 real functions were improved by 2-4% on large sizes

LAPACK
• We dramatically improved the performance of several linear equation solvers (?spsv/?hpsv, ?gsrsv, ?gtsw, ?ptsv, ?sysv/?hesv). Banded and packed storage format and multiple right-hand sides cases see speed-ups of up to 100 times.
• All symmetric eigensolvers (?syev/?syevd, ?syevx/?heevx, ?syevr?heevr) have significantly improved, since tridiagonalization routine (?sytrd/?hetrd) has sped up to 4 times
• All symmetric eigensolvers in packed storage (?spev/?hepv, ?spevd/?hepvd, ?spevx/?hepvx) have significantly improved, since tridiagonalization routine in packed storage (?sptrd/?heptrd) has sped up to 3 times
• Up to 2 times improvement for a number of routines applying orthogonal/unitary transformations (?ormqr/?unmqr, ?ormrq/?unmrq, ?ormql/?unmlq).

FFTs
• Improved single threaded performance of up to 1.8 times on complex 1D FFTs for power-of-two sizes.
• On Intel® 64 architecture-based systems running in 64-bit mode single precision complex backward 1D FFT for data sizes greater than $2^{22}$ elements have been sped up by up to 2 times on 4 threads and up to 2.4 times on 8 threads on Intel® Itanium® processors

VML/VSL
• Performance of VSL functions is improved on non-Intel processors by approximately 2 times on average
• Performance of VML vdExp, vdSin, and vdCos functions is improved on non-Intel processors by 18% on average
• Performance of VSL functions is improved on IA-32 and Intel® 64 architecture by 7% on average

Functionality

Linear Algebra: BLAS and LAPACK
Employ BLAS and LAPACK routines that are highly optimized for Intel processors, and that provide significant performance improvements over alternative implementations. Intel MKL 10.1 is compliant with the new 3.1 release of LAPACK.

The BLAS and LAPACK libraries are time-honored standards for solving a large variety of linear algebra problems. The Intel® Math Kernel Library (Intel® MKL) contains an implementation of BLAS and LAPACK that is highly optimized for Intel® processors. Intel MKL can enable you to achieve significant performance improvements over alternative implementations of BLAS and LAPACK.

BLAS
Basic Linear Algebra Subprograms (BLAS) provide the basic vector and matrix operations underlying many linear algebra problems. Intel MKL BLAS support includes:
• BLAS Level 1 - vector-vector operations
• BLAS Level 2 - vector-matrix operations
• BLAS Level 3 - matrix-matrix operations
• Sparse BLAS - an extension of BLAS Levels 1, 2, and 3
Multiple matrix storage schemes (Full, Packed, Banded) are provided for BLAS levels 2 and 3.

Gain the performance enhancements of multiprocessing without altering your application using parallelized (threaded) BLAS routines from Intel MKL. If you wish to manage threading in your applications, all BLAS functions within Intel MKL are thread-safe.

Sparse BLAS
Achieve performance improvements and lower memory requirements with sparse BLAS routines that have been carefully optimized for data sparsity. Sparse BLAS includes a set of functions that perform common vector and matrix operations on sparse data (data where the majority of elements are zero). Sparse BLAS coverage includes
selected Level 1, 2, and 3 BLAS routines for double-precision real functions. Sparse BLAS are often used in conjunction with sparse solvers. Intel MKL supports both NIST* and SparseKit* style interfaces.

The following Matrix Types and Data Storage Formats are supported:

<table>
<thead>
<tr>
<th>Matrix Types</th>
<th>Data Storage Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>Compressed Sparse Row (CSR)</td>
</tr>
<tr>
<td>Symmetric</td>
<td>Compressed Sparse Column (CSC)</td>
</tr>
<tr>
<td>Triangular</td>
<td>Block Sparse Row (BSR)</td>
</tr>
<tr>
<td>Diagonal</td>
<td>Diagonal (DIA)</td>
</tr>
<tr>
<td>Skew-symmetric</td>
<td>Coordinate (COD)</td>
</tr>
<tr>
<td></td>
<td>Skyline</td>
</tr>
</tbody>
</table>

**LAPACK**

Intel MKL includes Linear Algebra Package (LAPACK) routines that are used for solving:

- Linear equations
- Eigenvalue problems
- Least squares problems
- Singular value problems

LAPACK routines support both real and complex data. Routines are supported for systems of equations with the following types of matrices: general, banded, symmetric or Hermitian, triangular, and tridiagonal. The LAPACK routines within Intel MKL provide multiple matrix storage schemes. LAPACK routines are available with a Fortran interface.

**BLAS and LAPACK Performance**

Double-precision general matrix-matrix multiply (DGEMM) is the workhorse routine for dense linear algebra. The charts below compare DGEMM performance of Intel MKL 10.1 to ATLAS* (Automatically Tuned Linear Algebra Software). ATLAS is a popular linear algebra software package that includes the complete BLAS API and a small subset of the LAPACK API. More information on ATLAS is available at [http://math-atlas.sourceforge.net/](http://math-atlas.sourceforge.net/).

Intel MKL has been optimized for performance and tuned to provide excellent scaling for multiple threads. The performance benefits of using Intel MKL on today's dual and quad-core processors can be 2-5 times that of alternatives. The charts below show the following:

1. Intel MKL can provide significant performance benefit over ATLAS*.
2. The multiple processor/threading performance scaling of Intel MKL BLAS and LAPACK is impressive.
The Intel MKL implementation of the ScaLAPACK library is specially tuned for Itanium®, Intel® Xeon®, and Intel® Pentium® processor-based systems. ScaLAPACK includes two areas of Linear Algebra—direct solvers and the eigenvalue problems. As such, we will look at both PDGETRF (a direct solver used for solving linear systems of equations) and PDSYEV (used for solving eigenvalue problems). PDGETRF (Parallel, Double precision, GEneral, TRiangular matrix Factorization) is a key function in the linear equations solver area because it is a general factorization routine that applies to many classes of matrices, and because the lower upper (LU) Factorization that it completes is the performance-intensive portion of linear equations solvers.

In summary, the BLAS and LAPACK functionality included in Intel MKL is highly optimized for Intel processors and can significantly increase the performance of your application compared to alternative implementations of BLAS and LAPACK.

Linear Algebra: ScaLAPACK

The Intel MKL implementation of ScaLAPACK can provide significant performance improvements over the standard NETLIB implementation.

Intel® Math Kernel Library (Intel® MKL) provides the underlying components of ScaLAPACK (Scalable Linear Algebra Package), including a distributed memory version of BLAS (PBLAS or Parallel BLAS) and a set of Basic Linear Algebra Communication Subprograms (BLACS) for inter-processor communication.

ScaLAPACK is a standard package of routines for solving linear algebra problems on distributed memory multiprocessor machines (clusters).

These charts show both the impressive scaling of Intel MKL DGEMM and LAPACK (dgetrf) when using multiple threads as well as the performance advantage over ATLAS and FFTW respectively that can be obtained.

In summary, the BLAS and LAPACK functionality included in Intel MKL is highly optimized for Intel processors and can significantly increase the performance of your application compared to alternative implementations of BLAS and LAPACK.

ScaLAPACK Performance*

The Intel MKL implementation of the ScaLAPACK library is specially tuned for Itanium®, Intel® Xeon®, and Intel® Pentium® processor-based systems.

ScaLAPACK includes two areas of Linear Algebra—direct solvers and the eigenvalue problems. As such, we will look at both PDGETRF (a direct solver used for solving linear systems of equations) and PDSYEV (used for solving eigenvalue problems). PDGETRF (Parallel, Double precision, GEneral, TRiangular matrix Factorization) is a key function in the linear equations solver area because it is a general factorization routine that applies to many classes of matrices, and because the lower upper (LU) Factorization that it completes is the performance-intensive portion of linear equations solvers.

In our tests, we compare the Intel MKL implementation of ScaLAPACK to the publicly available implementation from NETLIB. We show the performance of Netlib ScaLAPACK using BLAS from Intel MKL as well as ATLAS* BLAS. More information on the ScaLAPACK library is available at http://www.netlib.org/scalapack/*.

Raw Performance

Figure 2 shows performance on a 32-node cluster with 64 Intel Xeon processors for various problem and memory sizes. Figure 2 illustrates that:

1. Intel MKL ScaLAPACK significantly outperforms NETLIB ScaLAPACK.
2. Intel MKL is even more impressive when compared to NETLIB ScaLAPACK using ATLAS* BLAS.
Block Size Robustness

When running ScalAPACK, you must decide how to "block" your data. The process of determining how to distribute your data among nodes involves choosing an appropriate block size. The block size determines the amount of data that goes to each node. This requires effort and choosing the wrong block size can have significant adverse effects on performance.

The Intel MKL implementation of ScalAPACK is tolerant of block size differences. Figure 5 below shows how Intel MKL provides approximately the same high performance regardless of block size. The same cannot be said for NETLIB ScalAPACK.

In summary, the Intel MKL implementation of ScalAPACK is highly optimized for Intel processors and can significantly increase the performance of your application compared to other implementations of ScalAPACK.
The CCLRC (UK) has published a detailed analysis of direct sparse solvers in which PARDISO is found to perform very well compared to alternatives. View the report homepage at http://epubs.cclrc.ac.uk/work-details?w=34126 or go directly to the PDF file [401KB] at http://epubs.cclrc.ac.uk/bitstream/724/raltr-2005005.pdf.


PARDISO*: Parallel Direct Sparse Solver

New Out-of-Core Support!

In version 10.0 we added support for out-of-core memory to PARDISO. While computers have greatly increased memory capacity, there continue to be a large number of problems for which problem sizes are too great to solve with in-memory solutions. For customers who are encountering problem size limitations we encourage you to try our new out-of-core memory PARDISO solution.

The PARDISO* solver, licensed from the University of Basel, is a thread-safe, high-performance, memory-efficient software library for solving large sparse, symmetric, and asymmetric linear systems of equations on shared-memory multiprocessors.

The PARDISO solver exploits pipelining parallelism and memory hierarchies with a combination of left- and right-looking Level-3 BLAS super-node techniques. To improve sequential and parallel sparse numerical factorization performance, the algorithms are based on a Level-3 BLAS update.

For sufficiently large problem sizes, numerical experiments demonstrate that the scalability of the parallel algorithm is nearly independent of the shared-memory multiprocessing architecture, and speed-ups of up to seven times (on eight processors) have been observed. This approach to parallelism is based on OpenMP* directives.

The CCLRC (UK) has published a detailed analysis of direct sparse solvers in which PARDISO is found to perform very well compared to alternatives. View the report homepage at http://epubs.cclrc.ac.uk/work-details?w=34126 or go directly to the PDF file [401KB] at http://epubs.cclrc.ac.uk/bitstream/724/raltr-2005005.pdf.


PARDISO supports a wide range of sparse matrix types and computes the solution of real or complex; symmetric, structurally symmetric, or asymmetric; positive definite, indefinite, or Hermitian sparse linear systems of equations on shared-memory multiprocessing architectures.
Preconditioners (also referred to as “accelerators”) are used to accelerate an iterative solution process. In some cases, their use can reduce dramatically the number of iterations and thus lead to better solver performance. Intel MKL currently includes two preconditioners called ILU0 (Incomplete LU Factorization) and ILUT (Incomplete LU Factorization with Threshold). The ILU/ILUT preconditioners can be applied to any non-degenerate matrix and can be used alone or together with the MKL FGMRES solver. Both preconditioners are based on a well-known factorization of the original matrix into a product of two triangular matrices (low triangular and upper triangular). Usually this decomposition leads to some fill-in in the resulting matrix structure as compared to the original matrix however the distinctive feature of the ILU0 preconditioner is that it preserves the structure of the original matrix in the result, while ILUT preconditioner controls the number of fill-ins. ILUT preconditioner is more reliable, while computations with ILU0 can normally be performed faster and with less memory usage.

Sparse BLAS

Sparse BLAS is a set of functions that perform a number of common vector and matrix operations on sparse data. Sparse vectors and matrices are those in which the majority of elements are zeros. Intel MKL includes an implementation of sparse BLAS has been specially optimized to take advantage of data sparsity. Sparse BLAS coverage includes selected BLAS Level 1 routines for all data types and Level 2 and 3 BLAS routines for double-precision real functions. Matrix types include general matrices, symmetric matrices, triangular matrices, anti-symmetric, and diagonal matrices. Data structures supported include Compressed Sparse Row (CSR), Compressed Sparse Column (CSC), diagonal, coordinate, and skyline formats. The data below shows up to 10x performance with 8 threads, and the performance improvements between in Intel MKL 10.1 compared to a previous major version.

References


Utilization of Parallel Solver Libraries to solve Structural and Fluid Problems (PDF 187KB): Read a whitepaper by Cranes Software that analyzes the ability of the Intel MKL PARDISO sparse solver in Finite Element Analysis (FEA) applications athttp://www.intel.com/cd/software/products/asmo-na/eng/371766.htm
Fast Fourier Transforms (FFT)
Utilize our multi-dimensional FFT routines (1D up to 7D) with a modern, easy-to-use C and Fortran interface. Intel MKL supports distributed memory clusters with the same API enabling you to improve your performance by distributing the work over a large number of processors with minimal effort. Intel MKL also provides compatibility with the FFTW 2.x and 3.0 interfaces making it easy for current FFTW users to plug Intel MKL into their existing applications.

Fourier transforms are used in digital signal processing, image processing, and in partial differential equation (PDE) solvers. The Fast Fourier Transform (FFT) functionality in Intel® Math Kernel Library (Intel® MKL) has been highly optimized for Intel® architecture-based machines.

Intel MKL also provides support for distributed memory multiprocessor architecture machines (clusters).

Features of Intel MKL Fast Fourier Transforms include:
- High performance
- Outstanding multiprocessor scaling
- Multi-dimension support (1-D up to 7-D)
- Mixed-radix support
- Modern, easy-to-use interface
- FFTW 2.x and 3.x interfaces

Interfaces

Fortran and C
Both Fortran and C interfaces exist for all FFT functions.

Modern FFT Interface
Intel MKL offers a novel FFT interface that is specifically designed to be easier to use and maintain.

FFTW Interface
Intel MKL includes FFTW 2.x and 3.x compatibility APIs enabling FFTW users to integrate Intel MKL with minimal effort. Our goal is to provide FFTW users with easy access to the high performance and quality of Intel MKL Fast Fourier Transforms. FFTW is an alternative FFT package from MIT. See the Technical User Notes at http://www.intel.com/software/products/mkl/docs/fftw_mkl_user_notes_2.htm

- FFTW 3.x interface
- FFTW 2.x interface

Performance (Shared Memory Parallel)
Intel MKL Fast Fourier Transforms are highly optimized for medium and large size transforms. The charts below compare the performance of Intel MKL with FFTW, a popular FFT software package. The charts illustrate these key points:
- Intel MKL offers outstanding performance for medium and large size transforms.
- Intel MKL offers outstanding scalability for multiprocessor systems.
- Intel MKL compares favorably to FFTW for medium and large size transforms.

Note: For all tests below, a fftw_plan_dft_xd() was called using the FFTW_PATIENT flag to ensure good performance was achieved for FFTW.

1-D Transform
The one-dimensional transform charts below show how Intel MKL can provide a significant performance boost over FFTW.

See larger image at: http://cache-www.intel.com/cd/00/00/41/38/413826_413826.jpg
See larger image at: http://cache-www.intel.com/cd/00/00/41/38/413827_413827.jpg
2-D Transforms

Two-dimensional transforms are large enough that they can benefit significantly from multiprocessing. The following charts illustrate how Intel MKL provides significantly better threaded performance than FFTW. In fact, Intel MKL single thread performance often outperforms the 2, 4, and 8 thread performance of FFTW.

See larger image: [http://software.intel.com/file/10793](http://software.intel.com/file/10793)

3-D Transforms

Three-dimensional transforms show the same behavior as the two-dimensional transforms. Intel MKL excels at providing excellent performance scaling for medium and large transforms.

See larger image: [http://software.intel.com/file/10795](http://software.intel.com/file/10795)

See larger image: [http://cache-www.intel.com/cd/00/00/41/38/413830_413830.jpg](http://cache-www.intel.com/cd/00/00/41/38/413830_413830.jpg)


See larger image: [http://cache-www.intel.com/cd/00/00/41/38/413830_413830.jpg](http://cache-www.intel.com/cd/00/00/41/38/413830_413830.jpg)
Performance (Distributed Memory)

Intel MKL also provides support for distributed memory multiprocessor architecture machines (clusters). Included in this support are FFTs designed for distributed memory parallelism. As cluster systems are often configured with much higher degrees of parallelism than shared memory parallel systems, Intel MKL expands headroom for problem size and performance, particularly for large problems.

The charts illustrate these key points:

- Intel MKL offers outstanding performance across a wide range of transform sizes.
- Intel MKL offers increasing performance benefit over FFTW as the problem size increases.

In summary, the Fast Fourier Transform functionality of the Intel Math Kernel Library offers outstanding performance and scalability across a wide range of problem sizes that are of interest to the high performance computing community. Unlike FFTW, Intel MKL does not require the time-consuming process of doing pre-production machine calibration runs ("plan creation") prior to running one's software. Intel MKL provides FFT support for distributed memory parallel computing systems and provides outstanding performance headroom for larger problem sizes. Intel MKL includes a FFTW interface to enable developers to easily compare performance and switch from FFTW to Intel MKL with minimal source code changes.
References

**FFTW 2.x Wrappers**: Learn more about porting FFTW 2.x applications to Intel MKL at [http://www.intel.com/software/products/mkl/docs/fftw2xmklnotes.htm](http://www.intel.com/software/products/mkl/docs/fftw2xmklnotes.htm)

**FFTW 3.x Wrappers**: Learn more about porting FFTW 3.x applications to Intel MKL at [http://www.intel.com/software/products/mkl/docs/fftw3xmklnotes.htm](http://www.intel.com/software/products/mkl/docs/fftw3xmklnotes.htm)


Vector Math Library

Increase application performance with vectorized implementations of computationally intensive core mathematical functions (power, trigonometric, exponential, hyperbolic, logarithmic, and more).

The Vector Math Library (VML) included with Intel® Math Kernel Library (Intel® MKL) provides highly optimized vector implementations of computationally intensive core mathematical functions. The library has both Fortran and C interfaces for all VML functions. All functions have also been threaded (click on function to see threaded performance).

---

**Supported VML Functions**

<table>
<thead>
<tr>
<th>Arithmetic</th>
<th>Trigonometric</th>
<th>Hyperbolic</th>
<th>Power/Root</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Sin+</td>
<td>Sinh+</td>
<td>Pow+</td>
</tr>
<tr>
<td>Sub</td>
<td>Cos+</td>
<td>Cosh+</td>
<td>Powx+</td>
</tr>
<tr>
<td>Div</td>
<td>SinCos</td>
<td>Tanh+</td>
<td>Pow2o3</td>
</tr>
<tr>
<td>Sqr</td>
<td>CS**</td>
<td>Asinh+</td>
<td>Pow3o2</td>
</tr>
<tr>
<td>Mul</td>
<td>Tan+</td>
<td>Acosh+</td>
<td>Sqr-1</td>
</tr>
<tr>
<td>Conjugate</td>
<td>Asin+</td>
<td>Atanh+</td>
<td>Cbrt</td>
</tr>
<tr>
<td>MulByConjugate</td>
<td>Acos+</td>
<td>InvSqrt</td>
<td></td>
</tr>
<tr>
<td>Abs</td>
<td>Atan+</td>
<td>InvCbrt</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Atan2</td>
<td></td>
<td>Hypot</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rounding</th>
<th>Exponential/Logarithmic</th>
<th>Special</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floor</td>
<td>Exp+</td>
<td>Erf</td>
<td>Inv</td>
</tr>
<tr>
<td>Ceil</td>
<td>Expint</td>
<td>Erfc</td>
<td>Div</td>
</tr>
<tr>
<td>Round</td>
<td>Ln+</td>
<td>ErfInv</td>
<td></td>
</tr>
<tr>
<td>Trunc</td>
<td>Log10+</td>
<td>ErfIncv (New)</td>
<td></td>
</tr>
<tr>
<td>Rint</td>
<td>Log1p</td>
<td>CdfNorm (New)</td>
<td></td>
</tr>
<tr>
<td>NearbyInt</td>
<td></td>
<td>CdfNormInv (New)</td>
<td></td>
</tr>
<tr>
<td>Modf</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All functions are available for Real data types.

^ Indicates support for Complex data types.

^^ Indicates support for ONLY Complex data types.

Performance

The Vector Math Library in Intel MKL can provide substantial performance advantages over scalar implementations. The chart below compares Intel® MKL 10.1 VML functions to the equivalent functions implemented by LibM and SVML, the runtime libraries that support calls to math functions from the Intel® C++ and Fortran compilers. The chart shows that most VML functions provide a 2-5x performance benefit in many cases and even up to 15x-30x in a few cases.
Accuracy Modes

VML functions support Single and Double Precision and are provided with three performance/accuracy modes. Having multiple accuracy modes is a key feature that separates Intel MKL from vectorizing compilers. With Intel MKL, you can choose the precision and accuracy that best meets your needs and thus maximizes performance of your specific code.

References


Vector Statistical Library

The Intel® Math Kernel library (Intel® MKL) includes an expanding amount of statistical functionality that we aggregate into a sub-library which we call the Vector Statistical Library (VSL). The Intel MKL Vector Statistical Library today includes Convolution/Correlation and an extensive collection of Random Number Generators.

Convolution/Correlation

VSL provides a set of routines intended to perform linear convolution and correlation transformations for single and double precision data. We provide:

- Fourier and Direct Algorithms
- One or Multiple Dimensions
- Single and Double Precision
- C and Fortran Interfaces
- Modern easy-to-use "task" oriented API
- IBM ESSL* Interface for ESSL users


Random Numbers

Speed up your simulations using our vector random number generators, which can provide substantial performance improvements over scalar random number generator alternatives.

The Intel MKL Vector Statistical Library (VSL) contains a collection of random number generators for a number of probability distributions. All VSL functions are highly optimized to deliver outstanding performance on Intel® architecture.

Applications that can significantly improve performance with VSL include simulation algorithms commonly used in physics, chemistry, medical simulations, and financial analysis software. The library provides both Fortran and C interfaces for all VSL functions.

VSL provides nine basic random number generators that differ in speed and statistical qualities, as shown below.
Random Number Generator Performance

VSL functions operate on single- and double-precision real vector arguments and can provide substantial performance advantages over scalar implementations. Tables containing complete performance information for all VSL random number generators and distributions on Intel® processors are available here.

To demonstrate the optimized performance of Intel® Math Kernel Library, we compared the standard C `rand()` function to the Intel MKL vector uniform random number generator. Intel MKL VSL functions are thread-safe, so we also present the speed of the Intel MKL vector uniform random number generator after it has been parallelized using OpenMP®.

### VSL Performance vs. C Code on Intel® Xeon® Processor-based System

<table>
<thead>
<tr>
<th></th>
<th>Intel Xeon Processor</th>
<th>Running Time (seconds)</th>
<th>Speedup vs. rand() (times)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard C <code>rand()</code> function</td>
<td>40.52</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>Intel® MKL VSL random number generator</td>
<td>6.88</td>
<td>5.89</td>
<td></td>
</tr>
<tr>
<td>OpenMP® version (8 threads)</td>
<td>0.92</td>
<td>44.04</td>
<td></td>
</tr>
</tbody>
</table>

1. Two-way Quad-Core Intel® Xeon® processor-based system (8 cores total), running at 2.4 GHz with 2x8MB L2 cache and 4GB memory with Windows® 2003 Enterprise x64 Edition and Intel® C++ Compiler 10.0
2. Intel® 64 architecture version of Intel® MKL VSL was used in measurements

Comparison of Intel® MKL to Alternative Libraries

Visual Numerics Inc. and Numerical Algorithms Group Ltd. are providers of popular numerical and statistical libraries. Visual Numerics provides a collection of mathematical and statistical analysis subroutines for Fortran and C/C++ users known as the IMSL Fortran 90 MP Library (F90MP®) and the IMSL C Numerical Library®, respectively. Numerical Algorithms Group provides numerical libraries for C/C++ and Fortran users known as NAG Fortran 77 Library*, NAG Fortran 90 Library*, NAG C Library*, NAG SMP Library* and the NAG Parallel Library*.

The following table presents summary information on random number generation capabilities existing in the Intel MKL 10.0, IMSL F90MP 6.0, and NAG Fortran 77 (Mark 21) libraries.
<table>
<thead>
<tr>
<th>Feature</th>
<th>Intel MKL</th>
<th>IMSL</th>
<th>NAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRNGs</td>
<td>7 pseudorandom</td>
<td>9 pseudorandom</td>
<td>2 pseudorandom</td>
</tr>
<tr>
<td></td>
<td>2 quasi-random</td>
<td>1 quasi-random</td>
<td>3 quasi-random</td>
</tr>
<tr>
<td>User-designed BRNGs</td>
<td>Supported</td>
<td>User-designed BRNG replaces library BRNG; many library services do not work with user-designed BRNG</td>
<td>Not supported</td>
</tr>
<tr>
<td>Sequence manipulation</td>
<td>• Mechanisms of creating, copying and deleting streams</td>
<td>• Service subroutines for switching between BRNGs</td>
<td>• Subroutines for saving and restoring BRNG seed</td>
</tr>
<tr>
<td></td>
<td>• Saving/restoring stream to/from a file</td>
<td>• Subroutines for saving and restoring BRNG seed</td>
<td>• Service subroutines for switching between BRNGs</td>
</tr>
<tr>
<td></td>
<td>• Arbitrary number of random streams based on one or more BRNGs</td>
<td>• Subroutines for saving and restoring state tables for GFSR and shuffled BRNGs</td>
<td></td>
</tr>
<tr>
<td>Sub-sequence splitting</td>
<td>Limited skip-ahead method support (only skipping 100,000 elements in a sequence)</td>
<td>Not supported</td>
<td></td>
</tr>
<tr>
<td>Distribution generators and other BRNG-related functionality</td>
<td>• 20 univariate</td>
<td>• 24 univariate</td>
<td>• 24 univariate</td>
</tr>
<tr>
<td></td>
<td>• 1 multivariate</td>
<td>• 3 multivariate</td>
<td>• 2 multivariate</td>
</tr>
<tr>
<td></td>
<td>• Multiple transformation methods available by passing method ID as parameter</td>
<td>• General discrete, continuous univariate and data based multivariate generators</td>
<td>• General univariate distribution generator</td>
</tr>
<tr>
<td></td>
<td>• Random orthogonal and correlation matrices, two-way tables, order statistics, samples and permutations, stochastic processes</td>
<td>• Random orthogonal and correlation matrices, random tables</td>
<td>• Random samples, permutations, time-series models, orthogonal and correlation matrices, random tables</td>
</tr>
<tr>
<td></td>
<td>• Multiple transformation methods available as separate subroutines</td>
<td>• Multiple transformation methods are not available</td>
<td>• ORNGs are for uniform, Gaussian and lognormal distributions</td>
</tr>
<tr>
<td></td>
<td>• ORNGs are for uniform distribution only</td>
<td>• ORNGs are for uniform, Gaussian and lognormal distributions</td>
<td>• ORNGs are for uniform, Gaussian and lognormal distributions</td>
</tr>
<tr>
<td>Vector/scalar interface</td>
<td>Vector interface only</td>
<td>Some distribution generators have vector and scalar form; others are of either vector or scalar form only</td>
<td>Some distribution generators have vector and scalar form; others are of either vector or scalar form only</td>
</tr>
<tr>
<td>Programming languages interface</td>
<td>C and Fortran interface in a single package</td>
<td>Fortran, C and Java interface in separate packages</td>
<td>C and Fortran interface in separate packages</td>
</tr>
<tr>
<td>Supported hardware</td>
<td>• Intel® architectures only</td>
<td>Multiple platforms (e.g., Intel, Cray, HP, IBM, Sun)</td>
<td>Multiple platforms (e.g., Apple, Cray, HP, IBM, Sun)</td>
</tr>
<tr>
<td></td>
<td>• Highly optimized for target architecture</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Two methods for generation of normally distributed random numbers are available in the library starting from Mark 20.
The following table presents numerical and performance results on an Intel Xeon processor-based system for the three libraries with random number generation capabilities, namely for Intel MKL 10.0, IMSL F90MP 6.0, and NAG SMP Fortran Library (Mark 21). Speed-ups are measured against the slowest version.

### Performance Comparison: Black-Scholes Option-Pricing Model

<table>
<thead>
<tr>
<th>Library</th>
<th>Basic Generator</th>
<th>Option Value (Exact Value)</th>
<th>Absolute Error (Standard Error)</th>
<th>Time (seconds)</th>
<th>Speedup (times faster)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Call</td>
<td>Put</td>
<td>Call</td>
<td>Put</td>
</tr>
<tr>
<td>Intel® MKL</td>
<td>MCG31m1</td>
<td>16.7306 (16.7341)</td>
<td>7.2177 (7.2179)</td>
<td>0.0036 (0.0019)</td>
<td>0.0002 (0.0009)</td>
</tr>
<tr>
<td></td>
<td>MCG59</td>
<td>16.7364 (16.7341)</td>
<td>7.2162 (7.2179)</td>
<td>0.0023 (0.0019)</td>
<td>0.0017 (0.0009)</td>
</tr>
<tr>
<td></td>
<td>MT19937</td>
<td>16.7349 (16.7341)</td>
<td>7.2164 (7.2179)</td>
<td>0.0007 (0.0019)</td>
<td>0.0015 (0.0009)</td>
</tr>
<tr>
<td>NAG</td>
<td>Original</td>
<td>16.7339 (16.7341)</td>
<td>7.2182 (7.2179)</td>
<td>0.0002 (0.0019)</td>
<td>0.0003 (0.0009)</td>
</tr>
<tr>
<td>IMLS</td>
<td>MT19937</td>
<td>16.7324 (16.7341)</td>
<td>7.2178 (7.2179)</td>
<td>0.0017 (0.0019)</td>
<td>0.0001 (0.0009)</td>
</tr>
<tr>
<td></td>
<td>Minimal Standard</td>
<td>16.7343 (16.7341)</td>
<td>7.2177 (7.2179)</td>
<td>0.0001 (0.0019)</td>
<td>0.0009 (0.0009)</td>
</tr>
</tbody>
</table>

1. Monte Carlo is a valuable tool for performing real-time financial analysis of complex, worldwide markets. In our example, we consider the well-known Black-Scholes option-pricing model, which is a framework for thinking about option pricing and is a de facto standard in the financial world.

We used the ‘minimal standard’ and 32-bit MT19937 basic generators in the IMSL library. We used the ‘original’ basic generator in the NAG library. For Intel MKL 10.0, we used three basic generators: MCG31m1, MT19937 and MCG59. The last being identical to the ‘original’ basic generator in NAG libraries, while properties of MCG31m1 are similar to the ‘minimal standard’ basic generator from IMSL.

For more information on this performance test, please refer to the white paper Making the Monte Carlo Approach Even Easier and Faster by Sergey A. Maidanov and Andrey Naraikin at: http://www.intel.com/cd/ids/developer/asmo-na/eng/95573.htm

In summary, the Vector Statistical Library of Intel Math Kernel Library contains Convolution/Correlation and a comprehensive set of random number generators that are highly optimized for Intel® architecture and can significantly increase the performance of your application over alternative solutions.

### References

**Intel VSL Performance Data**: Find complete performance information for all Intel VSL random number generators and distributions on Intel Xeon and Intel Itanium processors at http://www.intel.com/software/products/mkl/data/vsl/vsl_performance_data.htm

**Intel VSL Notes (PDF 1.2MB)**: View detailed information on VSL BRNGs and distributions, and random number generators in general at http://www.intel.com/cd/software/products/asmo-na/eng/347649.htm

**Monte Carlo European Options Pricing (PDF 85KB)**: See a paper discussing the use of popular random number generators in derivative security pricing at http://software.intel.com/en-us/articles/monte-carlo-simulation-using-various-industry-library-solutions
LINPACK Benchmark

Intel provides free LINPACK benchmark packages built with Intel MKL to help you obtain the highest possible benchmark results for your Intel® architecture-based systems.

Intel Optimized LINPACK Benchmark packages can help you in your quest to obtain the highest possible benchmark results for Intel® architecture-based systems. These free packages are implementations of the LINPACK benchmarks which use BLAS and LAPACK software that has been highly tuned for maximum performance on Intel® Xeon® processor-based and Itanium®-based systems. These same high performance BLAS and LAPACK routines are available to software developers in the Intel® Math Kernel Library.

Intel offers two LINPACK benchmark packages:

Intel Optimized SMP LINPACK Benchmark package

For use on SMP machines. This package contains implementations of the LINPACK 1000 benchmark code* (http://www.netlib.org/benchmark/1000). See below for performance and download information.

The package includes the executables listed below, as well as shell scripts and input files.

Intel Optimized (SMP) LINPACK Benchmark 10.1

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>linpack_itanium</td>
<td>64-bit program executable for Itanium®-based systems.</td>
</tr>
<tr>
<td>linpack_xeon32</td>
<td>32-bit program executable for Intel® Xeon® processor-based systems.</td>
</tr>
<tr>
<td>linpack_xeon64</td>
<td>64-bit program executable for 64-bit Intel® Xeon® processor-based systems.</td>
</tr>
</tbody>
</table>

Intel Optimized MP LINPACK Benchmark for Clusters package

This package is an implementation of the Massively Parallel MP LINPACK* (http://www.top500.org/) benchmark for use on distributed memory computer systems (clusters). Use this package to benchmark your computer cluster for submission to the Top500 Supercomputer list.

Intel Optimized MP LINPACK Benchmark 10.1 for Clusters

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPL 1.0a</td>
<td>The complete HPL 1.0a distribution, as well as additional files to make HPL easier to use.</td>
</tr>
<tr>
<td>nodeperf.c</td>
<td>A program to test for DGEMM performance consistency on all cluster nodes.</td>
</tr>
</tbody>
</table>

Ease of Use

The Intel Optimized LINPACK Benchmark packages have been designed to save you time. Fewer file downloads, no need to compile, and less iterating are benefits that save your valuable time. The Intel Optimized SMP LINPACK benchmark package doesn't require any of the time consuming searching that HPL does, and while the Intel Optimized MP LINPACK benchmark package does do searching, it provides information early in the process so you don't have to wait until the entire run is completed before reconfiguring and beginning another run. Try the Intel Optimized LINPACK benchmark packages and see how much time you save.

See the LINPACK Benchmark chapter of the Intel MKL Users Guide for more information.

<table>
<thead>
<tr>
<th>Elapsed Time h:mm</th>
<th>Intel Optimized SMP LINPACK binary</th>
<th>Elapsed Time h:mm</th>
<th>HPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:00</td>
<td>Start</td>
<td>0:00</td>
<td>Start</td>
</tr>
<tr>
<td>0:00</td>
<td>Download Intel Optimized LINPACK package</td>
<td>0:00</td>
<td>Download HPL</td>
</tr>
<tr>
<td>0:10</td>
<td>Configure scripts</td>
<td>0:10</td>
<td>Download BLAS</td>
</tr>
<tr>
<td>0:20</td>
<td>Start Run</td>
<td>0:20</td>
<td>Download MPI</td>
</tr>
<tr>
<td>1:00</td>
<td>Collect Results</td>
<td>0:30</td>
<td>Build MPI</td>
</tr>
<tr>
<td></td>
<td>1:00</td>
<td></td>
<td>Build HPL</td>
</tr>
<tr>
<td></td>
<td>1:30</td>
<td></td>
<td>Start Run 1</td>
</tr>
<tr>
<td></td>
<td>2:30</td>
<td></td>
<td>Review Results</td>
</tr>
<tr>
<td></td>
<td>2:50</td>
<td></td>
<td>Change HPL.dat for improvements</td>
</tr>
<tr>
<td></td>
<td>3:00</td>
<td></td>
<td>Start Run 2</td>
</tr>
<tr>
<td></td>
<td>4:00</td>
<td></td>
<td>Review Results</td>
</tr>
<tr>
<td></td>
<td>8:00</td>
<td></td>
<td>Iterate changing HPL.dat and running until satisfied with results. Can take many iterations.</td>
</tr>
</tbody>
</table>

Iterate changing HPL.dat and running until satisfied with results. Can take many iterations.

8:00 Call it a day and vow to finish tomorrow…
Performance

The following charts show the impressive performance that the Intel Optimized SMP LINPACK Benchmark package has achieved:

- 91.5 percent of theoretical maximum performance on Quad-Core Intel® Xeon® processor based Systems.
- Improved performance with newer versions of Intel MKL, up to 8% better performance with MKL 10.1 compared to MKL 9.1.

Downloads

The Optimized LINPACK and MP LINPACK Benchmark packages listed below are available for free download.


<table>
<thead>
<tr>
<th>Package Contents</th>
<th>Intel Optimized LINPACK Benchmark</th>
<th>Intel Optimized MP LINPACK Benchmark for Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>Download Size</td>
<td>Source</td>
</tr>
<tr>
<td>Linux* package (tgz)</td>
<td>6.71MB</td>
<td>X</td>
</tr>
<tr>
<td>Windows* package (zip)</td>
<td>3.79MB</td>
<td>X</td>
</tr>
<tr>
<td>Mac OS* package (tgz)</td>
<td>763KB</td>
<td>X</td>
</tr>
</tbody>
</table>

Questions/Comments?


References

Compatibility

Operating Systems
Intel MKL 10.1 supports Linux*, Windows* (including HPC Server 2008) and Mac OS* X. Linux variants include: Red Hat*, Suse*, Debian*, Ubuntu*, Asianux*, and other Linux Standard Base 3.1 variants.

Development Environments
Intel MKL is easily used and integrated with popular development tools and environments, such as Microsoft Visual Studio*, Xcode*, Eclipse*, and the GNU Compiler Collection (GCC).

Processors
Intel MKL 10.1 supports all Intel Architecture compatible processors and is specifically optimized for:

- Intel® Xeon® processor family
- Intel® Core™ processor family
- Intel® Itanium processors family
- Intel® Pentium® processor family
- AMD Opteron* and Athlon* processor families

NOTE: Intel MKL for Mac OS* X is not available as a standalone product. It is only available with the Intel C++ Compiler Professional Edition and Intel Fortran Compiler Professional Edition.

Technical Support
Every purchase of an Intel® Software Development Product includes a year of support services, which provides access to Intel Premier Support and all product updates during that time. Intel Premier Support gives you online access to Intel MKL discussion forum, technical notes, application notes, and documentation. Install the product, and then register to get support and product update information.

Intel® Premier Support
Receive one year of world-class technical support with every purchase of Intel MKL. During this period, you can download product upgrades free of charge, including major version releases. For more information, visit the Intel Registration Center at https://registrationcenter.intel.com/RegCenter/Register.aspx. Additionally, the user forum is a great place to get community support.

User Forum

References

§. Performance tests and ratings are measured using specific computer systems and/or components and reflect the appropriate performance of Intel products as measured by those tests. Any difference in system 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 the performance of Intel products, go to http://www.intel.com/software/products/

© 2009, Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.
*Other names and brands may be claimed as the property of others.
0209/BLACMD/PDF 321515-001