Intel® Math Kernel Library (Intel® MKL)

Part of Intel® Parallel Studio XE Composer Edition

December 2014

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
- 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**

Optimized Mathematical Building Blocks
Intel® Math Kernel Library

Linear Algebra

- BLAS
- LAPACK
- Sparse Solvers
  - Iterative
  - Pardiso*,
  - **Cluster_sparse_solver**
  - **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

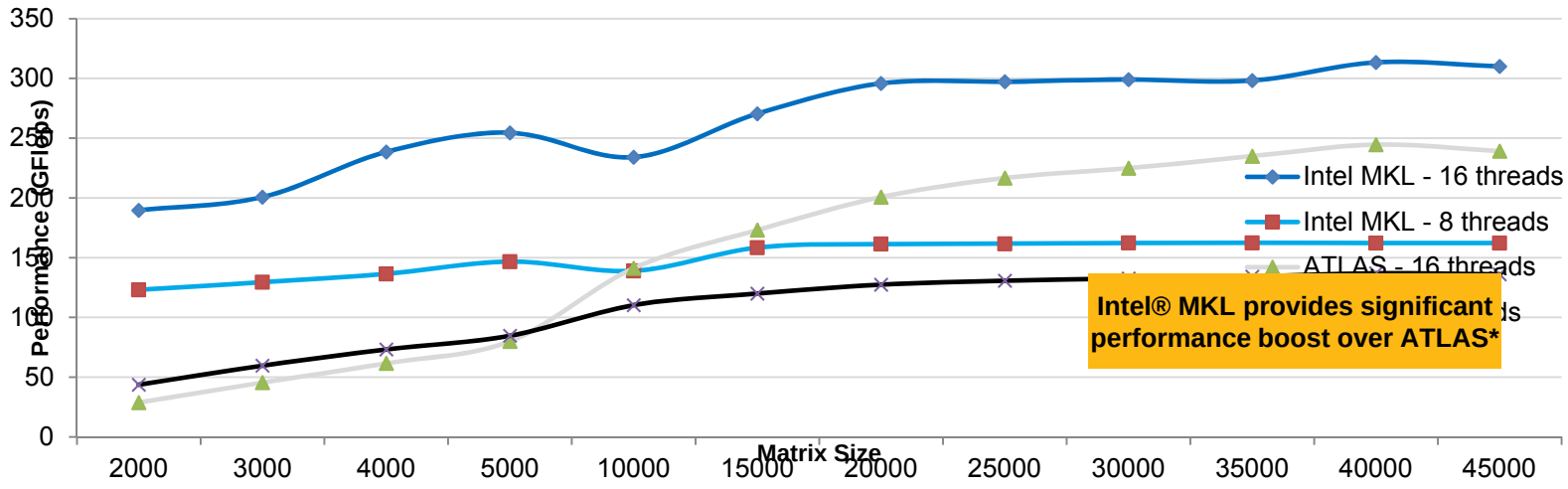
Summary Statistics

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

And More

- Splines
- Interpolation
- Trust Region
- Fast Poisson Solver

Immediate Performance Benefit to Applications



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*);

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

Intel® Xeon Phi™ Coprocessor Support by Intel® MKL

Automatic Offload	No code changes required Automatically uses both host and target Transparent data transfer and execution management
Compiler Assisted Offload	Explicit control 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, $(a+b)+c \neq a+(b+c)$
  - 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

BLAS – Basic Linear Algebra Subprograms

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

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

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

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

LAPACK – Linear Algebra PACKage

Defacto-standard APIs since early 1990s
1000s of linear algebra functions
4 floating point precisions supported
Breadth of coverage:

- Matrix factorizations:–LU, Cholesky, QR
- Solving systems of linear equations
- Condition number estimates
- Singular value decomposition
- Symmetric and non-symmetric eigenvalue problems
- And much, much more

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

Intel® MKL Vector Math Library (VML)

- Collection of vector math functions
- Real/Complex
- Double precision(DP), Single precision(SP)
- 3 accuracy modes
  - **H**igh **A**ccuracy, HA (correct rounding in >99% cases, behave according to C99; slowest, default mode)
  - **L**ow **A**ccuracy, LA (≤2 lsb incorrect, behave according to C99;30-50% faster than HA)
  - **E**nhanced **P**erformance, EP (~1/2 incorrect bits, is not guaranteed on entire domain;30-50% faster than LA)

Real functions

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

Complex functions

Arithmetic	Power and Root	Exponential & Logarithmic	Trigonometric	Hyperbolic
Add	Div	Exp	Sin	Sinh
Sub	Sqrt	Ln	Cos	Cosh
Mul	Pow	Log10	Tan	Tanh
MulByConj	Powx		Asin	Asinh
Conj			Acos	Acosh
Abs			Atan	Atanh
Arg			CIS	

New Features of Intel MKL 11.2

- Parallel Direct Sparse Solvers for Clusters
- Verbose mode for BLAS and LAPACK
- S/C/Z/DGEMM improvements on small matrix sizes
- Significant SVD and Eigen Solvers performance improvements
- Cookbook recipes
- Other features and optimizations

Parallel Direct Sparse Solvers for Clusters

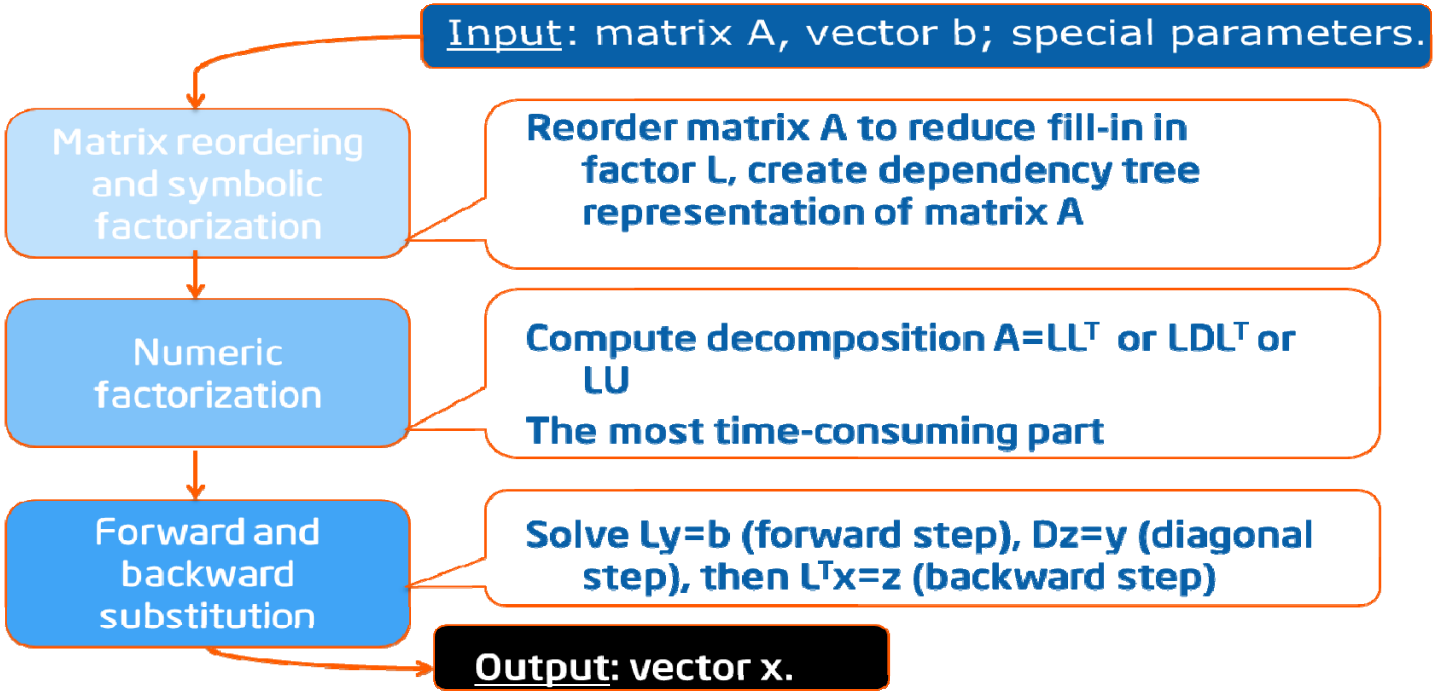
-In-core, OOC, hybrid and now – distribute version
-Similar functionality and API for SMP and Cluster versions

```
{
……
pardiso (pt, maxfct, mnum, mtype, phase, n, a, ia, j
a, perm, nrhs, iparm, msglvl, b, x, error);
……..
}
```

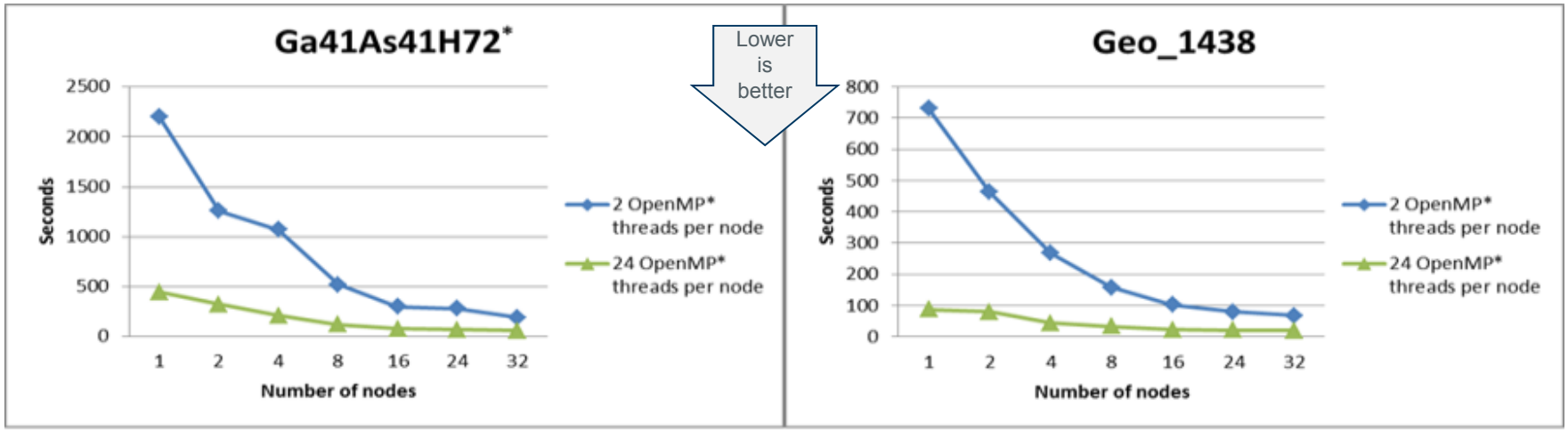
```
{
…..
cluster_sparse_solver(pt, maxfct, mnum, mtype, phase, n
, a, ia, ja, perm, nrhs, iparm, msglvl, b, x, comm, error )
……
}
```

-2 input formats: CSR, DCSR
-Hybrid threading of computations
-C and Fortran examples

Parallel Direct Sparse Solvers for Clusters



MKL *cluster_sparse_solver* - Scalability Results



- **Each node contains two Intel® Xeon® E5-2697 v2 processors (24 cores in total), 64GB RAM**
- **Intel® MKL 11.2**

*Here and further: The University of Florida Sparse Matrix Collection T. A. Davis and Y. Hu, ACM Transactions on Mathematical Software, Vol 38, Issue 1, 2011, pp 1:1 - 1:25. http://www.cise.ufl.edu/research/sparse/matrices.

Verbose Mode for BLAS & LAPACK

Motivation ? : ***" It would be worth if MKL reports called functions names with parameters, that we won't spend additional time and resource to figure this out"***

- **How to Enable Verbose Mode? :**

  - Set the environment variable MKL_VERBOSE to 1
    OR
  - Function Call mkl_verbose(1)

- **What happens when Verbose mode Enabled?**

  - Every call of a verbose-enabled function finishes with printing verbose log, including the list of version Information, the function name, the argument

Verbose mode - example

{MKL VERBOSE} **thrID:**4158465744 **func:** DGEMM (T,T,5,2,4,0.5, 0xa1481c8,4,0xa148170,2,-1.2,0xa148290,5) **time:**0.2e-2s **freq:**3.2GHz **Nthr:**8 **dyn**:0 **CPU:**AVX **CNR:**off **AO**:0 **iface**:ilp64 **plat:**lnx **mem**:1 **mstat**:10MB

where,

- **thrID**: calling thread ID
- **func**: called function name with input parameters
- **time**: time (sec) spend in function
- **freq**: current CPU frequency
- **Nthr**: Number of MKL threads
- **dyn**: {1|0} – MKL_DYNAMIC
- **CPU**: SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, AVX, AVX2 for Intel & *AMD for non Intel*
- **CNR**:OFF, AUTO, COMPATIBLE, SSE2, SSE3, SSSE3, SSE4_1, SSE4_2, AVX, AVX2
- **AO:** {1|0},
- **iface**: ilp64, lp64, cdecl, stdcall
- **plat**: lnx, osx, win, mic,
- **mem**: {1|0} - MKL_FAST_MM on/off
- **mstat:** MB used by function

?GEMM improvements on small matrix sizes

How to call?

•include "mkl_direct_call.fi OR include "mkl_direct_call.h"

•Compile with the option "-DMKL_DIRECT_CALL or "MKL_DIRECT_CALL_SEQ"

•Fortran :-fpp (/fpp on Windows)

•C/C++ : -std=c99 or /Qstd=c99

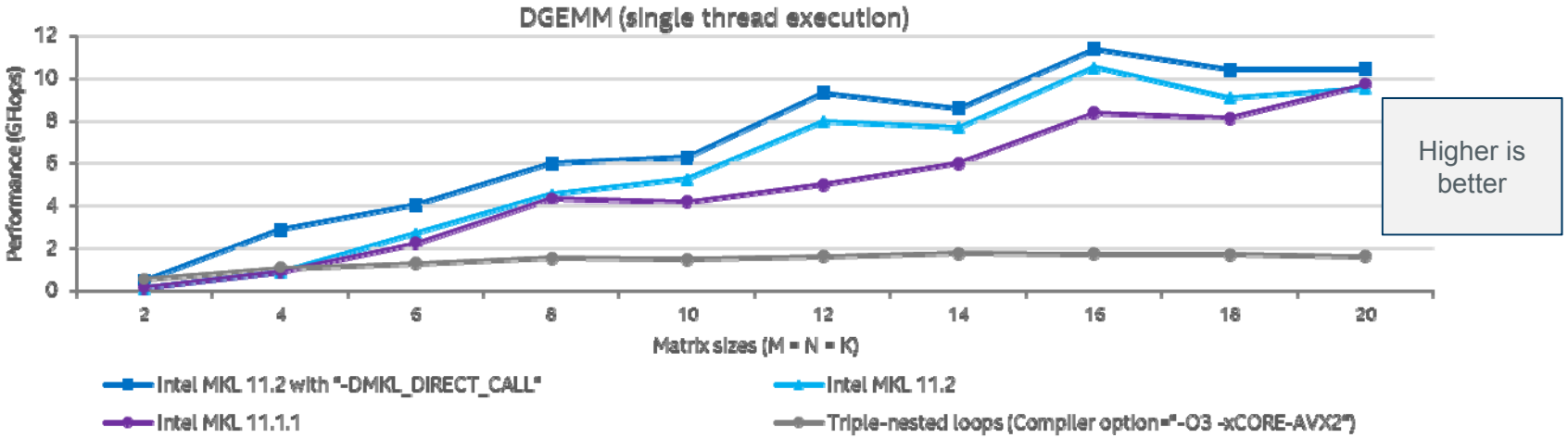
Current Limitations:

- No errors reported if incorrect parameters are passed to the function call
- CNR, GNU Fortran, CBLAS API, "verbose mode"

- *"I Just wanted to thank the MKL team for MKL_DIRECT_CALL. It works well with small matrices as well as large matrices. I am seeing a 2x performance improvement when the matrices are small. Previously I had my own _inline gemm for small matrices as calling MKL zgemm was inefficient, but the new feature has allowed me to clean up my code as well as speed it up...."* ***Andrew Cunningham. ESI.***

?GEMM improvements on small matrix sizes

- Performance improvement for small matrices (for 4x4 to 18x18 matrices) over Intel® MKL 11.1.1 for S/DGEMM
- Further performance improvement (up to 2x) over Intel® MKL 11.1.1
- through reduced call/error checking overhead and new inline functions (S/D/C/ZGEMM).



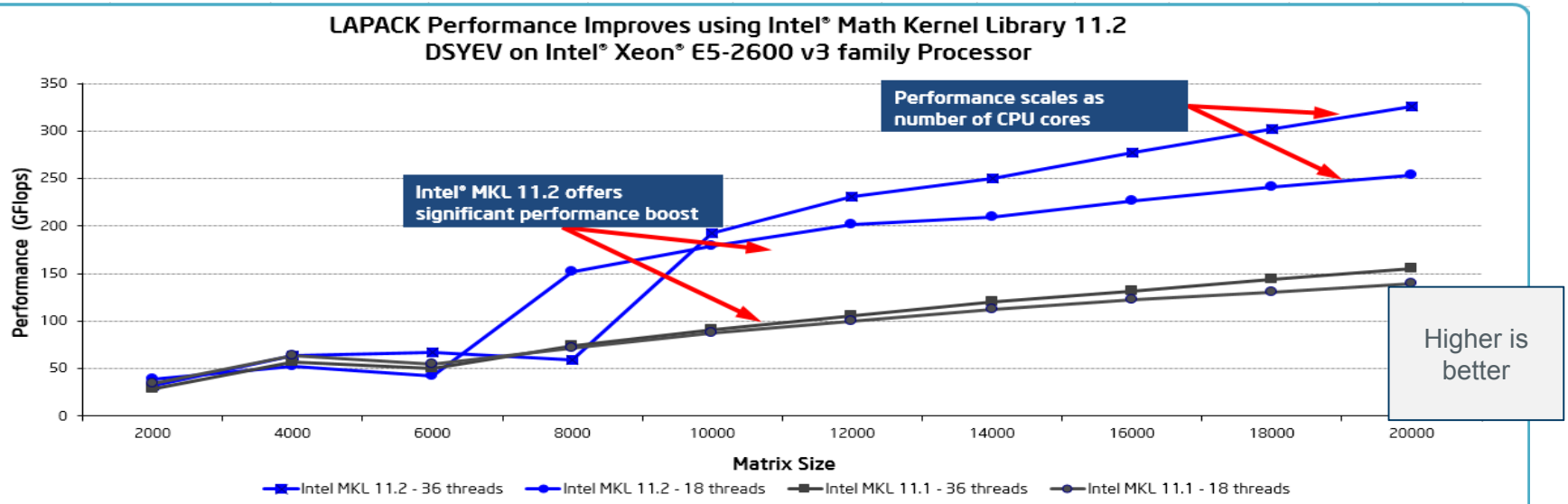
Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 11.1.1 and 11.2, Intel® C++ Compiler 15.0; Hardware: Intel® Xeon® Processor E5-2697v3, 2 Fourteen-Core CPUs (35MB LLC, 2.6GHz), 64GB of RAM; Operating System: RHEL 6.1 GA x86_64; Benchmark Source: Intel Corporation.

Significant SVD performance improvement

- Number of optimizations for Symmetric Eigen Solvers and SVD
- Mostly for large size matrices (N > 4000, 6000 and so)
- SVD speedup of > 6X for big thread count and matrix sizes
- Similar speedup for Eigen solvers

More details in the KB article
https://software.intel.com/en-us/articles/significant-performance-improvment-of-symmetric-eigensolvers-and-svd-in-intel-mkl-112

DSYEV performance improvement



Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 11.2, 11.1; Hardware: Intel® Xeon® Processor E5-2699v3, 2 18-Core CPUs (45MB LLC, 2.3GHz), 64GB of RAM; Operating System: RHEL 6.2 GA x86_64; Benchmark Source: Intel Corporation. Only eigenvalues are computed. Flops are counted by formula 8/3*n^3. numactl was used to collect MKL 11.2 data.

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, refer to www.intel.com/performance/resources/benchmark_limitations.htm.

* Other brands and names are the property of their respective owners

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

Cookbook - Recipe

Using LAPACK symmetric eigensolvers for Hermitian tridiagonal matrices

Goal

Compute eigenvalues and eigenvectors for a Hermitian tridiagonal matrix using LAPACK symmetric eigensolvers.

LAPACK provides symmetric eigensolvers for real-valued tridiagonal matrices, but no corresponding eigensolvers for complex Hermitian matrices.

Solution

A simple matrix transformation to a Hermitian tridiagonal matrix allows you to use one of the LAPACK eigensolvers.

1. Multiply the Hermitian tridiagonal matrix by a matrix calculated to eliminate the imaginary parts of the off-diagonal elements, which transforms it to a real-valued matrix.
2. Choose the LAPACK eigensolver routine according to the task you wish to perform and the algorithm you wish to use:
   - `sstev`, `dstev`: Compute all eigenvalues and, optionally, eigenvectors.
   - `sstevd`, `dstevd`: Compute all eigenvalues and, optionally, eigenvectors using a divide-and-conquer algorithm.
   - `sstevx`, `dstevx`: Compute selected eigenvalues and eigenvectors.
   - `sstevr`, `dstevr`: Compute selected eigenvalues and, optionally, eigenvectors using Relatively Robust Representations.
3. Reverse the transformation to the eigenvalues and eigenvectors returned by the LAPACK routine in order to get the eigenvalues and eigenvectors of the original matrix.

Calculating eigenvalues and eigenvectors for a Hermitian tridiagonal matrix using DSTEV

```
      PROGRAM complex_tridiagonal_hev_solver
      IMPLICIT NONE
      INTEGER N,INFO,I,J
      PARAMETER (N=5)
C Matrix
      COMPLEX*16 EC(N-1)
      REAL*8 D(N)
C Eigenvectors
```

```
C Test the vectors are eigenvectors
      DO I=1,N
          DO J=1,N
              UZ(J) = -DCMPLX(D(I),0D0)*VZ(J,I)
          END DO
          UZ(1)=UZ(1) + DC(1)*VZ(1,I) + EC(1)*VZ(2,I)
          DO J=2,N-1
             UZ(j)=UZ(j) + CONJG(EC(J-1))*VZ(J-1,I)+
     &                    DC(J)*VZ(J,I) + EC(J)*VZ(j+1,I)
          END DO
          UZ(N)=UZ(N) + CONJG(EC(N-1))*VZ(N-1,I) + DC(N)*VZ(N,I)
          PRINT *,"For ",I,"th eigenvalue ||S*U^j-lambda_j*U^j||=",
     &            DZNRM2(N,UZ,1)
      END DO

      STOP
      END
```

Discussion

In this example, *S* is a complex-valued tridiagonal Hermitian matrix:

$$S = \begin{bmatrix} d_1 & e_1 & & & & \\ c_1 & d_2 & e_2 & & & \\ & c_2 & d_3 & e_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & c_{N-2} & d_{N-1} & e_{N-1} \\ & & & & c_{N-1} & d_N \end{bmatrix}$$

$$c_j = \overline{e_j}, j = 1, 2, \ldots N-1$$

Construct a diagonal matrix *T*, where $|t_i| = 1$ for $i = 1, 2, \ldots N$:

$$T = \begin{bmatrix} t_1 & & & & \\ & t_2 & & & \\ & & \ddots & & \\ & & & t_{N-1} & \\ & & & & t_N \end{bmatrix}$$

A new matrix $S_1 = T^H S T$ has the same eigenvalues as *S*.

Intel® Numeric String Conversion Library

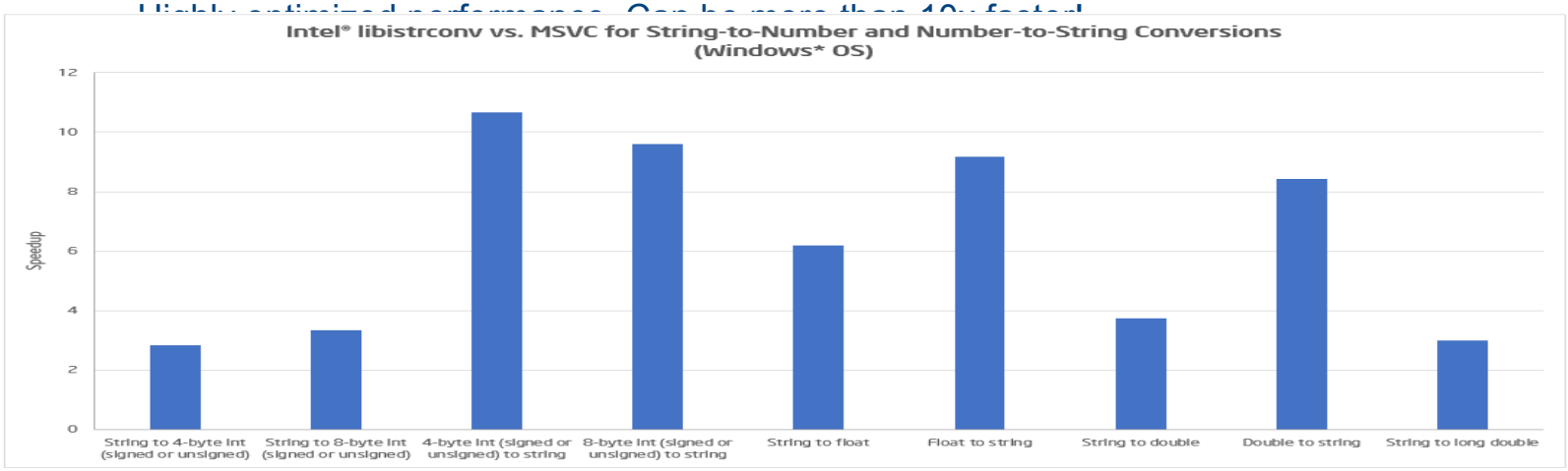
What is “Intel® Numeric String Conversion Library?”

A collection of routines that convert between ASCII strings of decimal numbers and C numeric data types.

- ASCII string (number) ↔ Floating-point number (float, double, long double)
- ASCII string (number) ↔ Integers (int32_t, uint32_t, int64_t, uint64_t)

Similar to the C functions strtof, strtod, strtol, strtoll, and snprintf, but with

- Highly optimized performance. Can be more than 10x faster!



Configuration Info - Versions: Intel® C++ Compiler XE 2015 Beta, Microsoft* C/C++ Optimizing Compiler Version 15.00.21022.08 for x64 ; Hardware: Intel® Core Processor i7-2700K, Quad-core CPU (8MB LLC, 3.5GHz), 8GB of RAM; Operating System: Windows Server 2008 R2 x64; Benchmark Source: Intel Corporation. The reported speedup for each conversion is the geomean of measurements for different value ranges.

Who Should Use It and How?

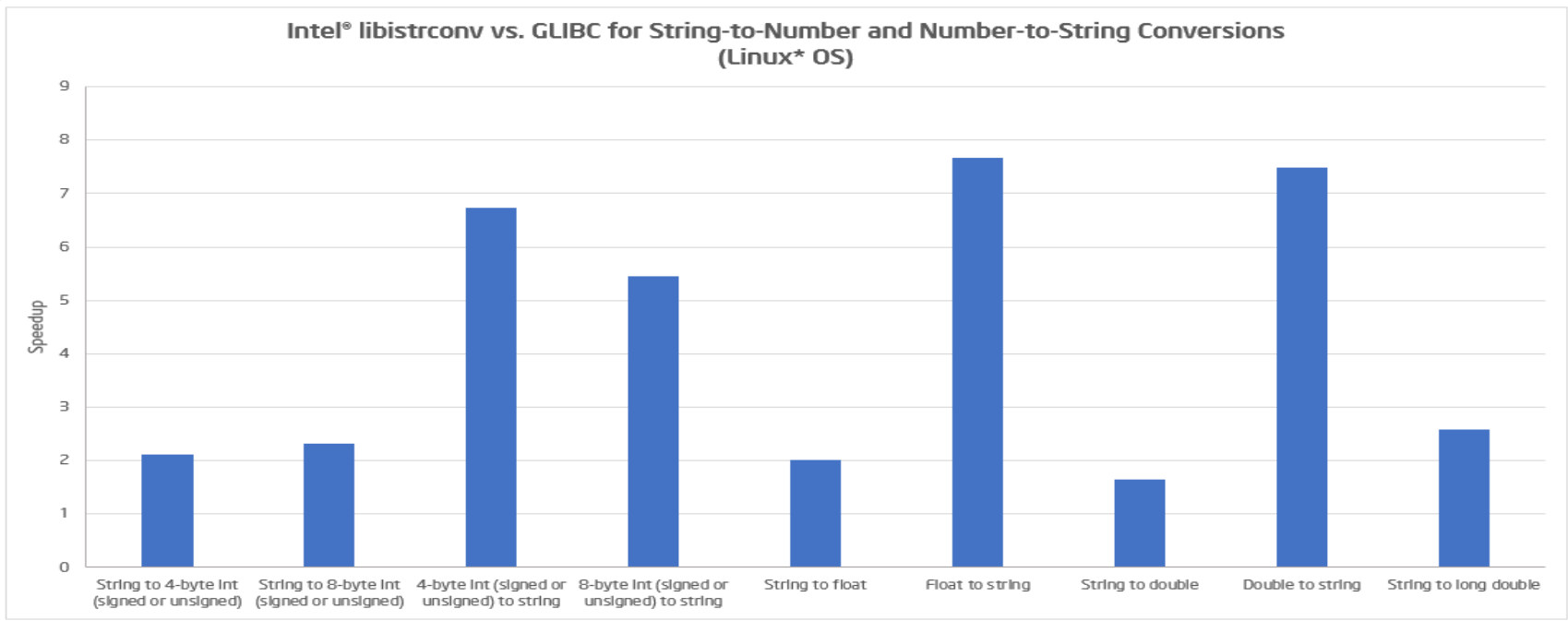
Applications that process or produce large amount of numeric values in textual formats, for example:

- Stock market tickers processing
- Reading/writing XML or JSON files that contain lots of numbers
- Reading/writing csv files that contain lots of numbers
- Web log analysis

Available as a library: libistrconv

- Bundled in Intel® C++ Compiler 14.0 update 1 and above.
- Intel compilers are not required to link with the library.
- Supported on Windows*, Linux*, and OS X*.

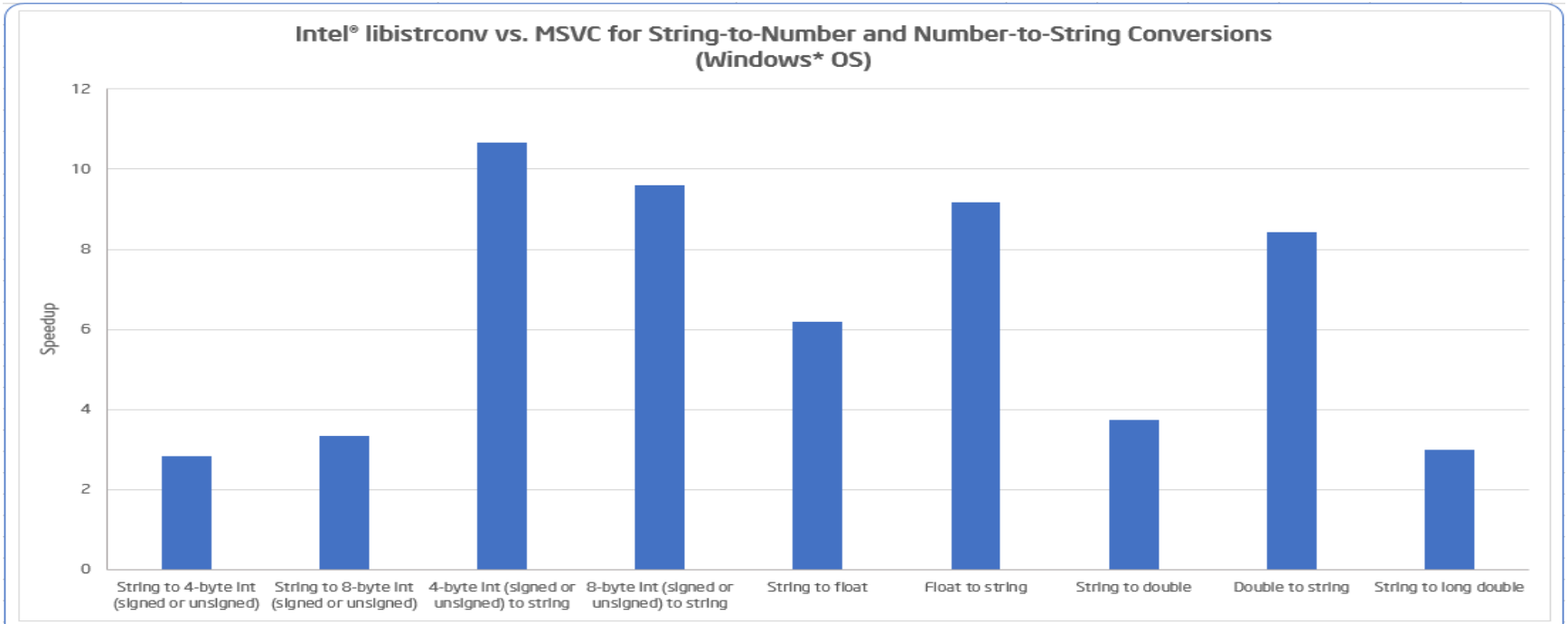
```
#include <istrconv.h>

float fx;
fx = __IML_string_to_float(pi, NULL);

char str[LENGTH];
int prec = 6;
__IML_float_to_string(s, LENGTH, prec,
fx);
```



Configuration Info - Versions: Intel® C++ Compiler XE 2015 Beta, GCC 4.4.6; Hardware: Intel® Xeon® Processor E5-2687W, 2 Eight-Core CPUs (20MB LLC, 3.1GHz), 32GB of RAM; Operating System: RHEL 6 GA x86_64; Benchmark Source: Intel Corporation. The reported speedup for each conversion is the geomean of measurements for different value ranges.

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.

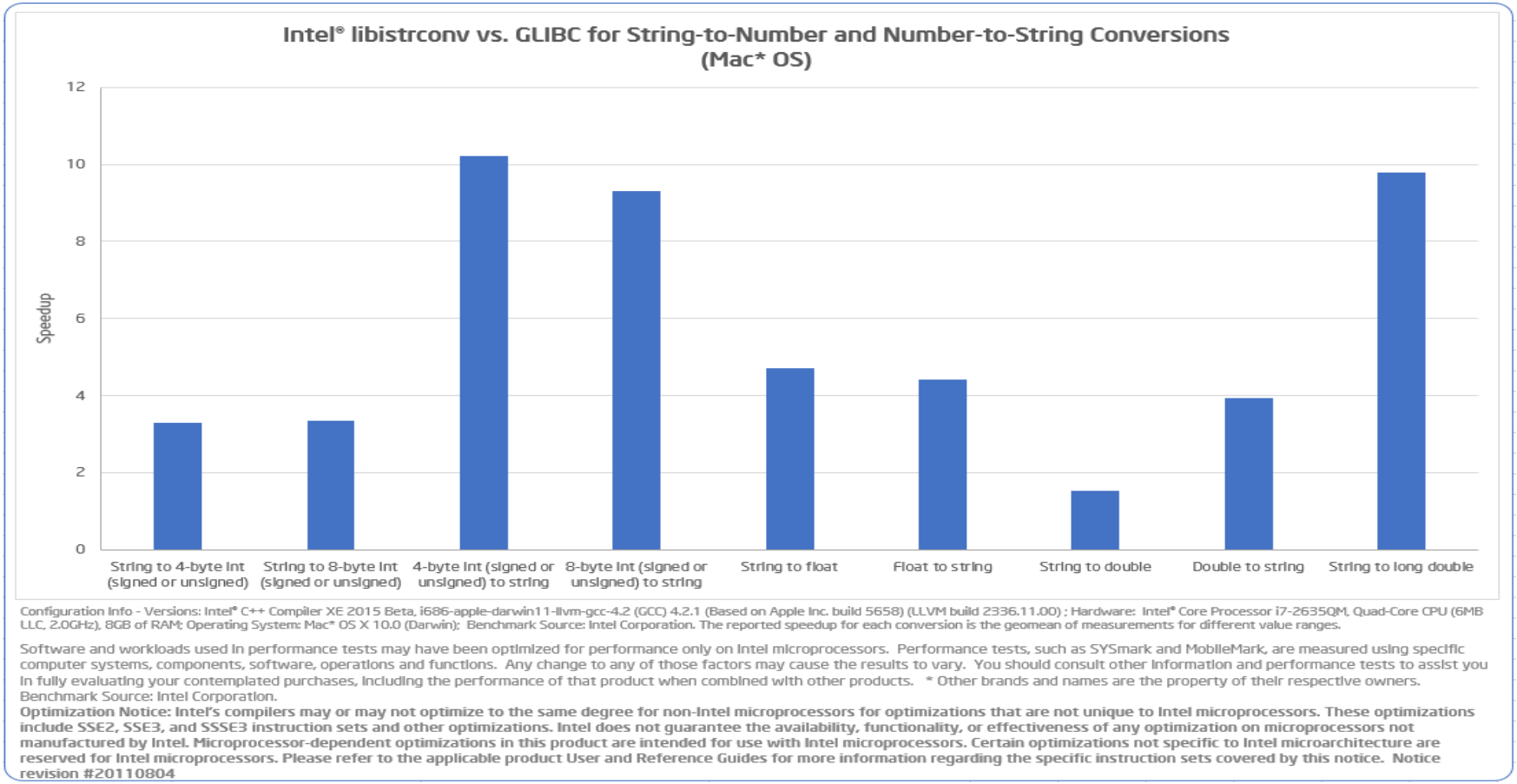
Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability,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



Configuration Info - Versions: Intel® C++ Compiler XE 2015 Beta, Microsoft* C/C++ Optimizing Compiler Version 15.00.21022.08 for x64 ; Hardware: Intel® Core Processor i7-2700K, Quad-core CPU (8MB LLC, 3.5GHz), 8GB of RAM; Operating System: Windows Server 2008 R2 x64; Benchmark Source: Intel Corporation. The reported speedup for each conversion is the geomean of measurements for different value ranges.

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



Configuration Info - Versions: Intel® C++ Compiler XE 2015 Beta, i686-apple-darwin11-llvm-gcc-4.2 (GCC) 4.2.1 (Based on Apple Inc. build 5658) (LLVM build 2336.11.00) ; Hardware: Intel® Core Processor i7-2635QM, Quad-Core CPU (6MB LLC, 2.0GHz), 8GB of RAM; Operating System: Mac* OS X 10.0 (Darwin); Benchmark Source: Intel Corporation. The reported speedup for each conversion is the geomean of measurements for different value ranges.

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

Additional Resources

Knowledge base article:

- https://software.intel.com/sites/default/files/managed/c5/51/libistrconv-KB.pdf
- See performance data on Windows*, Linux*, and OS X*.

Intel® C++ Compiler User and Reference Guide:

- https://software.intel.com/en-us/node/485141
- See a complete list of functions and code examples.

Summary:

•Parallelism + microarchitecture optimization - the key performance of your application
•Math libraries (IPP and MKL) enable design and implement parallelism without headache ☺
•Math libraries allow easily get a highly optimized code for a wide range of input data

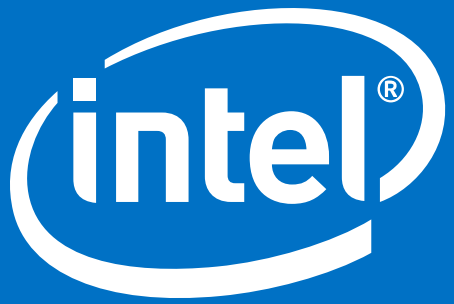
Welcome to tell us your recommendations, requests or report any issue

https://software.intel.com/ru-ru/intel-software-evaluation-center

Legal Disclaimer & 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



Software