A Tour of the Sparse Linear Algebra Functionality in Intel® Math Kernel Library

Zhang Zhang

Technical Consulting Engineer

Software & Services Group

Intel Corp.

Agenda

Intel® MKL Sparse BLAS

Intel® MKL PARDISO

Intel® MKL Parallel Direct Sparse Solver for Clusters

Intel® MKL Iterative Sparse Solvers

Intel® MKL Extended Eigensolver

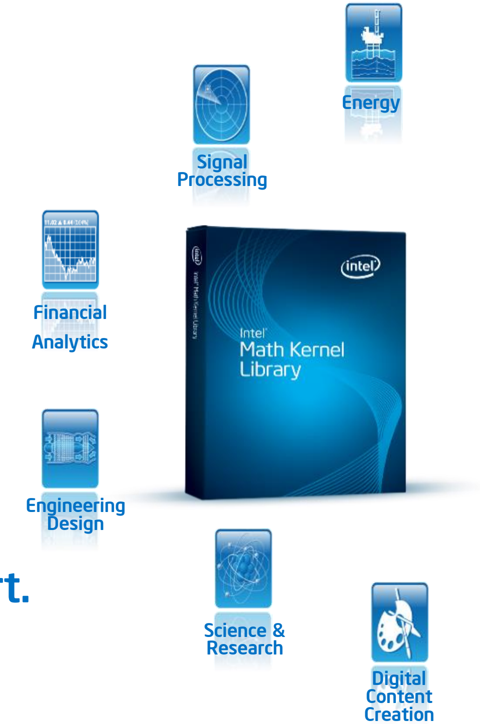
Intel® Math Kernel Library (Intel® MKL)

A feature-rich mathematical library designed with scientific, engineering and financial applications in mind.

Always optimized for the latest Intel® Xeon® and Intel® Xeon Phi™ product families.

Provides scientific programmers and domain scientists:

- Interfaces to de-facto standard APIs.
- Support for Linux*, Windows* and OS X* operating systems.
- The ability to extract great parallel performance with minimal effort.



Intel® MKL Used on the World's Fastest Supercomputers*

*http://www.top500.org

Optimization Notice

Intel® MKL Sparse BLAS

Optimization Notice

Functionality

x, y –vectors

A - sparse matrix

B, C – dense matrices

alpha, beta – scalars

op(A)=A or AT or conjg(AT)

Level 1

- y=a*x+y
- Dot products
- Rotation of points
- Gather/scatter

Level 2

- y=A*x
- y=alpha*op(A)*x+beta*y
- y= alpha*inv(op(A))*x

Level 3

- C=alpha*op(A)*B +beta*C
- Operations with parts L, D, U of matrix A decomposed as A=L+D+U
- C= alpha*inv(op(A))*B

  only for triangular matrix A

Optimization Notice

Supported Sparse Matrix Storage Formats

CSR – Compressed Sparse Row storage format

CSC - Compressed Sparse Column storage format

DIA - Diagonal storage format

SKY - Skyline storage format

BSR - Block compressed sparse row storage format

COO - Coordinate storage format

Intel MKL provides subroutines to convert from one format to another.

Optimization Notice

Important Features

Provides C and FORTRAN interface.

Supports 0-based and 1-based indexing.

Supports operations with partial matrices – *L, U, D*

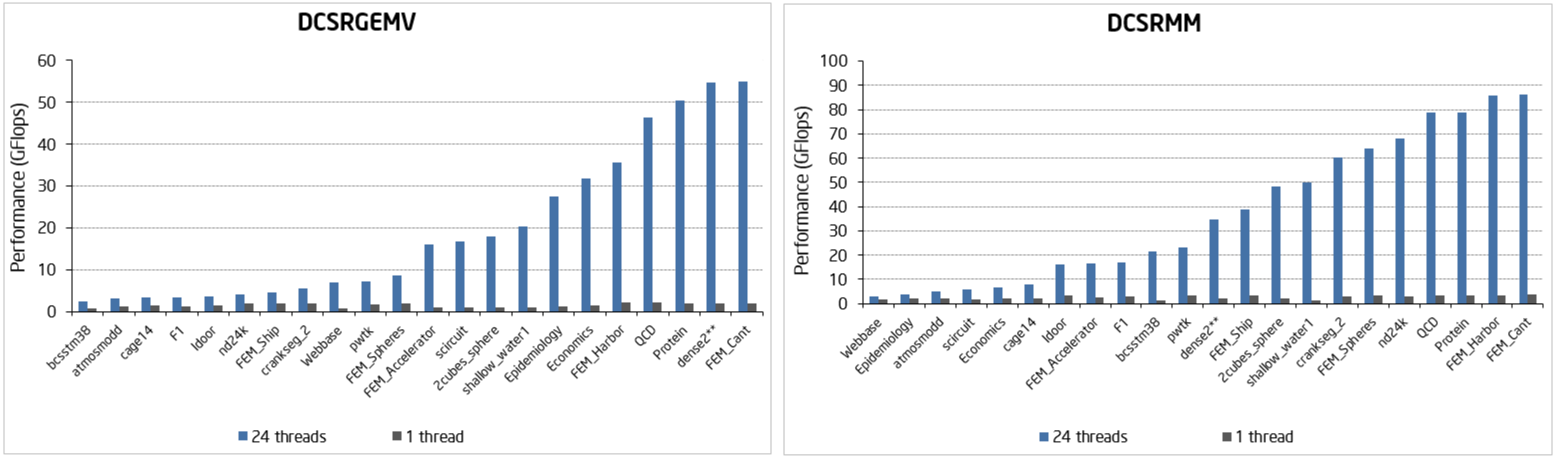
```
char transa = 't';
char matdescra[6];
matdescra[0] = 't';  /* triangular */
matdescra[1] = 'l';  /* lower */
matdescra[2] = 'n';  /* non-unit diagonal*/
matdescra[3] = 'c';  /* 0-based indexing */

mkl_dcsrmm( &transa, &m, &n, &m,
            &alpha, matdescra,
            values, cols, rowIdx,
            &b, &n, &beta, &res, &n);
```

Optimization Notice

Performance

Excellent Performance and Scalability Using Intel® Math Kernel Library Sparse BLAS

DCSRGEMV and CDSRMM on Intel® Xeon® Processor E5-2697 v2



Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 11.1; Hardware: Intel® Xeon® Processor E5-2697 v2, 2 Twelve-Core CPUs (30MB LLC, 2.7GHz), 32GB of RAM; Operating System: RHEL 6 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. ** All matrices except dense2 are from "The University of Florida Sparse Matrix Collection"(http://www.cise.ufl.edu/research/sparse/matrices/index.html)

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

Performance greatly depends on matrix structure.

On Intel® Xeon Phi coprocessors, performance is optimized for CSR format.

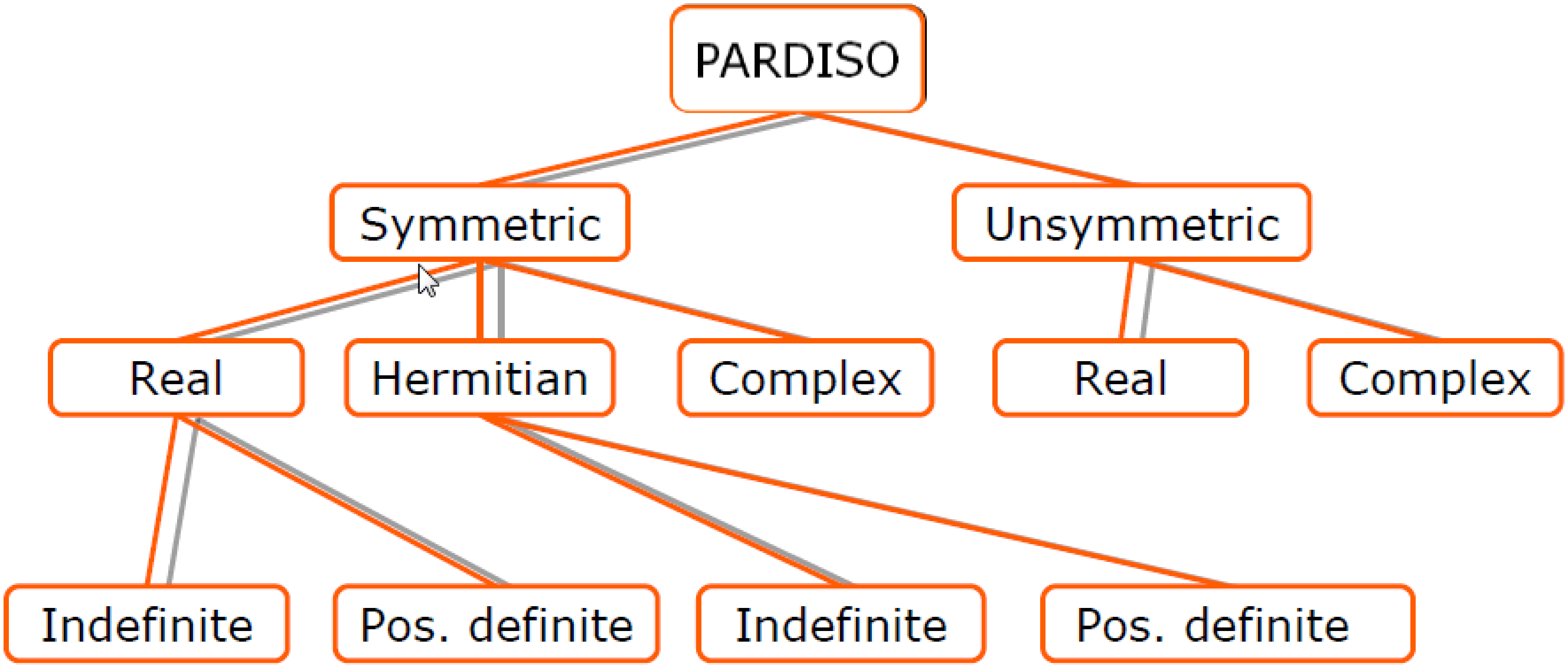
- But, see the "Intel MKL SpMV Format Technology Preview Package"
- https://software.intel.com/en-us/articles/the-intel-math-kernel-library-sparse-matrix-vector-multiply-format-prototype-package

Optimization Notice

Intel® MKL PARDISO

Optimization Notice

Intel® MKL PARDISO



$A = LDL^T$ $A = LL^T$ $A = LDL^H$ $A = LL^H$ $A = LU$

Optimization Notice

Main Features

FORTRAN and C interfaces.

Supports 0-based and 1-based indexing.

CSR format only.

Comes with a matrix checker.

Solves multiple matrices at the same time.

- Matrices with the same portrait.

Solves system with multiple RHS at the same time.

- Effectively using multiple threads

Supports direct-iterative preconditioning.

- Many matrices to be solved with identical sparsity pattern but gradually changing of the nonzero coefficients.

OOC – Out-of-Core PARDISO.

Optimization Notice

Usage Model

pardiso (pt, maxfct, mnum, mtype, phase, n, a, ia, ja, perm, nrhs, iparm, msglvl, b, x, error);

Initialization and parameters setting. Most important parameters:

- mtype – matrix type.
- phase – solution phases.
- iparm[64] – control various aspects of the solver.

Phase 1: Fill-in reduction analysis & reordering.

Phase 2: Numerical factorization.

Phase 3: Forward and Backward solve + Iterative refinement.

Termination and memory release.

User code can further control and fine tune each step.

Optimization Notice

PARDISO Example

```
iparm[0] = 0; iparm[1] = 2; …… iparm[9] = 13; iparm[10] = 1; ……
iparm[34] = 1; …… iparm[59] = 0;
phase = 12;
pardiso( pt,                  → opaque handle
        &maxfct,              } number of matrices and the current matrix to solve
        &mnum,
        &mtype,               → matrix type
        &phase,               → solution phase
        &n,                   } number of rows, and the matrix in the CSR format
        a, ia, ja,
        &perm,                → permutation vector
        &nrhs                 → number of right-hand sides
        iparm;                → the iparm array
        &msglvl,              → message level
        &b, &x, &error);
phase = 33; iparm[7] = 2;
pardiso( pt, ……,
        &b,                   → right-hand side
        &x,                   → solution vector
        &error);
phase = -1;
pardiso(……);
```

Optimization Notice

Direct Sparse Solver (DSS) Interface

DSS is a simplified interface of Intel® MKL PARDISO.

- Simpler and easier to use.
- Not as many fine tuning options.

A group of routines used in step-by-step solving process. Six steps:

- Initialization (dss_create)
- Matrix structure definition (dss_define_structure)
- Reordering (dss_reorder)
- Factorization (dss_factor_real, dss_factor_complex)
- Solve (dss_solve_real, dss_solve_complex)
- Release resource (dss_delete)

Optimization Notice

Intel® MKL Parallel Direct Sparse Solver for Clusters

Optimization Notice

Parallel Direct Sparse Solver for Clusters

A new feature in MKL 11.2 (coming out Q3'2014).

- Available for evaluation now in MKL 11.2 Beta.

Extends Intel® MKL PARDISOT to distributed memory systems.

Very competitive performance, comparing to MUMPS*, VSS*, and PETSc*.

Scales up to 1000 cores.

Familiar API to PARDISO users → Easy migration!

```
{
…
PARDISO (pt, &maxfct, &mnum, &mtype,
   &phase, &n, a, ia, ja, &idum,
   &nrhs, iparm, &msglvl, b, x,
   &error);
…
}
```

```
{
…
int comm =
   MPI_Comm_c2f(MPI_COMM_WORLD);
cluster_sparse_solver (pt, &maxfct,
   &mnum, &mtype, &phase, &n, a, ia,
   ja, &idum, &nrhs, iparm, &msglvl, b,
   x, &comm, &error);
…
}
```

Optimization Notice

Distribution of Input Matrix (Example)

Subset1 =

6	-1	*	-3	*
-1	5	*	*	*
*	*	3	*	2

Rows from 1 to 3

Values	6	-1	-3	5	3	2
Column	1	2	4	2	3	5
rowInd	1	4	5	7		

Stored on MPI rank 1

Subset2 =

*	*	8	6	2
-3	*	6	10	*
*	*	4	*	5

Rows from 3 to 5

Values	8	6	2	10	5
Column	3	4	5	4	5
rowInd	1	4	5	6	

Stored on MPI rank 2

A=

6	-1	*	-3	*
-1	5	*	*	*
*	*	11	6	4
-3	*	6	10	*
*	*	4	*	5

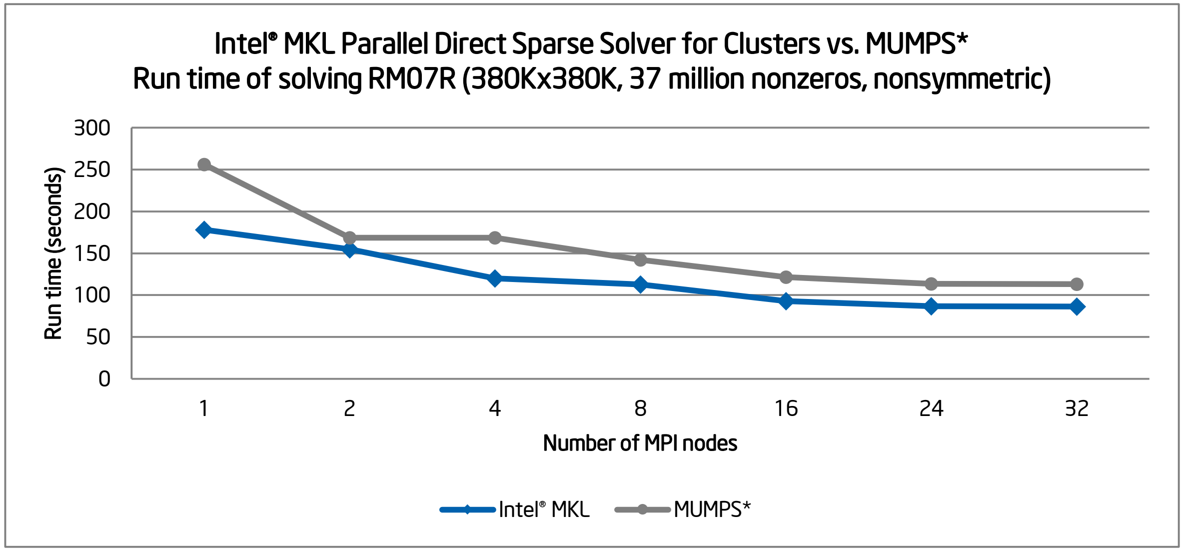
The resulted matrix to be solved

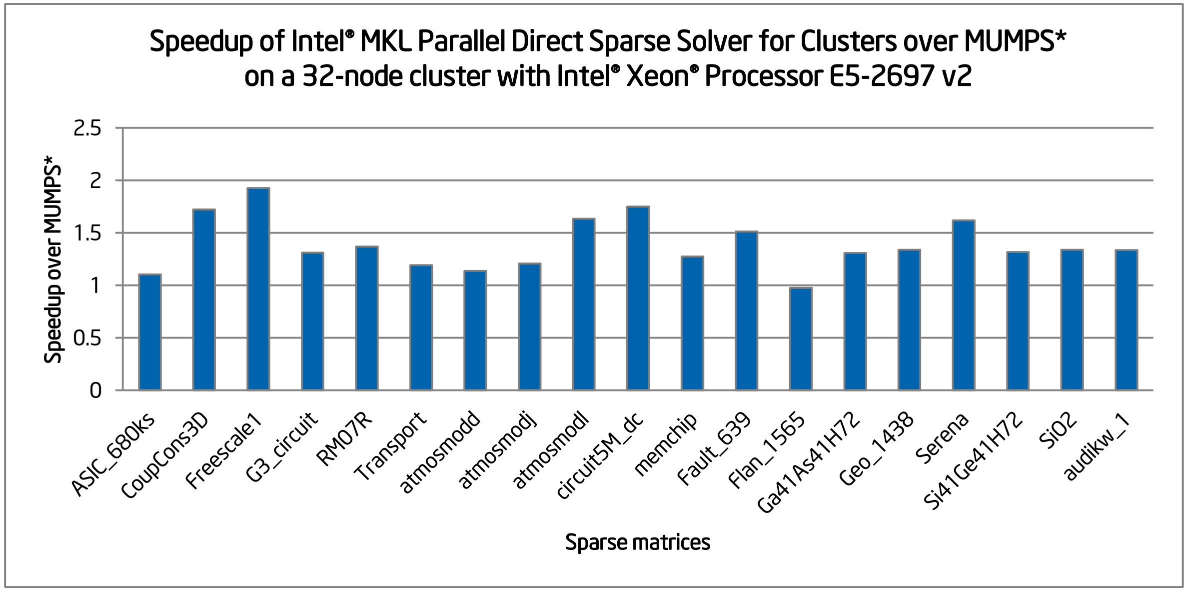
Values	6	-1	-3	5	11	6	4	10	5
Column	1	2	4	2	3	4	5	4	5
rowInd	1	4	5	8	9	10			

Note: A is symmetric, only upper-triangular part is stored.

Optimization Notice

Performance Comparison





Versions: Intel® Math Kernel Library (Intel® MKL) 11.2 beta; Intel® MPI 4.1.

Hardware of cluster nodes: Intel® Xeon® Processor E5-2697 v2, 2 Twelve-Core CPUs (30MB LLC, 2.7GHz), 64GB of RAM. Interconnect: FDR Infiniband.

Operating System: RHEL 6.1 GA x86_64

Benchmark Source: Intel Corporation.

Theses matrices are from "The University of Florida Sparse Matrix Collection" (http://www.cise.ufl.edu/research/sparse/matrices/index.html).

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

Optimization Notice

Intel® MKL Iterative Sparse Solvers

Optimization Notice

Intel® MKL Iterative Sparse Solvers

MKL provides two iterative solvers:

- RCI Conjugate Gradient Solver - Symmetric Positive Definite matrices.
- RCI Flexible Generalized Minimal Residual Solver (FGMRES) – Nonsymmetric indefinite.
- Supports only real-valued matrices.

Preconditioners:

- ILU0: Less effective, less computation
- ILUT: More effective, more computation

Optimization Notice

Reverse Communication Interface (RCI)

```
    initialize
        |
        |
        |
change parameters (manually)
        |
        |
        |
      check <--+
        |      |
        |      |
        |      |
      solve <--+
        |      |
        |      |
        |      |
       get <---+
```

Optimization Notice

Intel® MKL Extended Eigensolver for Sparse Symmetric/Hermitian Eigen Problems

Optimization Notice

An Innovative Method for Eigen Problems

Intel® MKL Extended Eigensolver is based on the FEAST algorithm:

- http://www.ecs.umass.edu/~polizzi/feast/
- Inspired by the contour integration technique in quantum mechanics.
- Subspace iteration method + Approximate spectral projection.
- Given a search interval [λ_min, λ_max], computes ALL eigenvalues and corresponding eigenvectors.

Symmetric/Hermitian standard Eigenproblem

$$Au = \lambda u,\ \lambda \in [\lambda_{\min}, \lambda_{\max}], A = A^*$$

Symmetric/Hermitian generalized Eigenproblem

$$Au = \lambda Bu, \lambda \in [\lambda_{\min}, \lambda_{\max}], A = A^*, B = B^* > 0$$

Optimization Notice

Functionality

Predefined driver routines for ease-of-use:

- Supports only CSR format and 1-based indexing.
- Internally depends on PARDISO for solving sparse linear systems (90% of computation).

RCI based routines for customizability:

- User to provide linear system solver and matrix multiplication routine.
- More flexibility for specific application needs.

Optimization Notice

Advantages

Efficiently solve Eigen problems for sparse matrices (symmetric or Hermitian).

Capture all multiplicities of Eigen values.

Highly parallelized algorithm:

- Great performance on multicore systems.
- Scalability depends on the internal linear sparse system solver.

Customizable:

- The RCI interface allows flexibility in solving special and challenging systems.
- E.g. users can use specialized iterative sparse solver and preconditioners.

Optimization Notice

Predefined Driver Routine Usage Model

dfeast_scsrev (uplo, n, a, ia, ja, fpm, epsout, loop, emin, emax, m0, e, x, m, res, info);

User to provide:

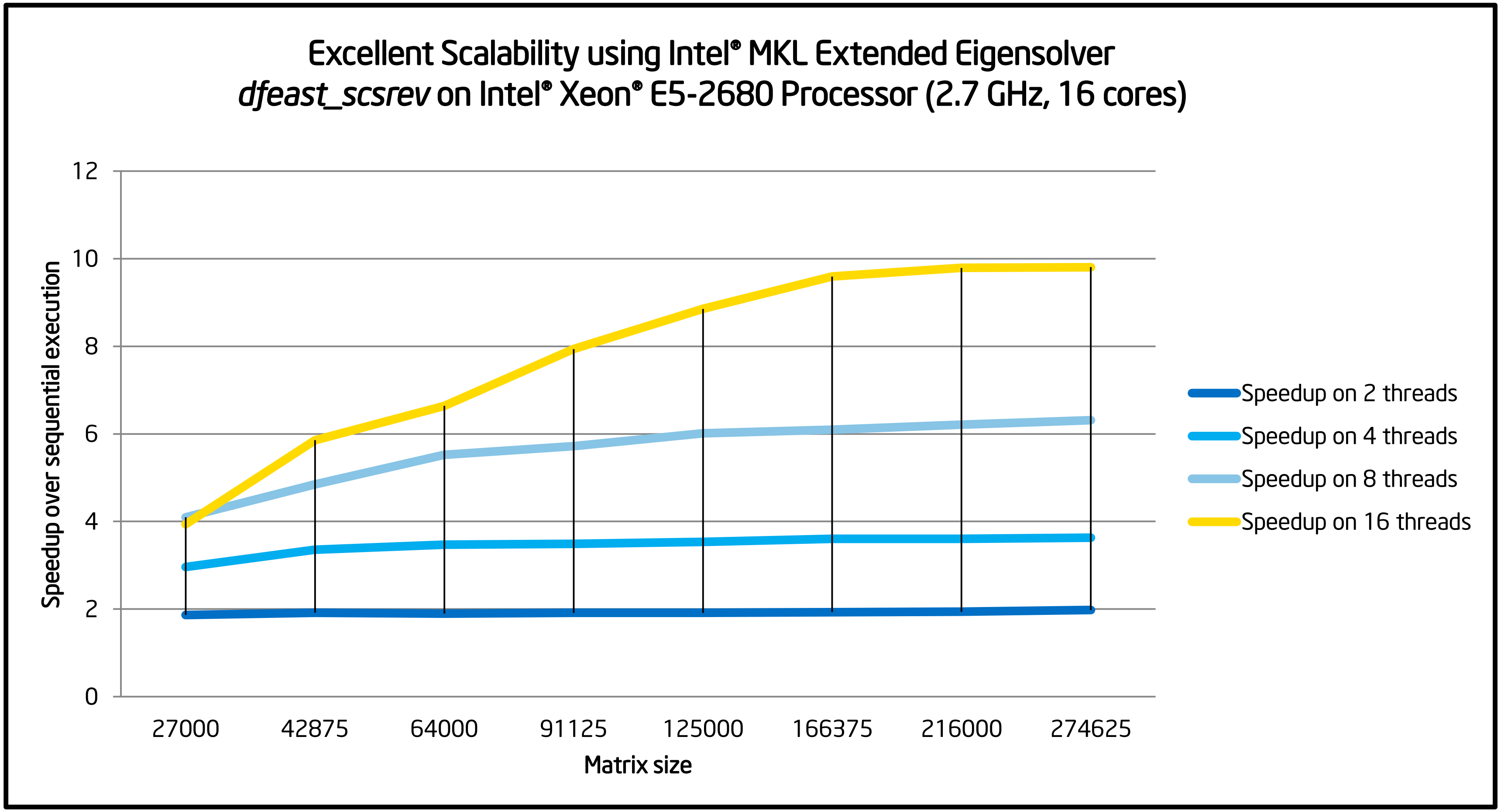
- Search interval ***(emin, emax)*** and an estimation of the number of eigenvalues within a given search interval ***m0***
- The input matrix stored in the CSR format ***(uplo, n, a, ia, ja)***
- ***fpm*** – Controls number of contour points, stopping criteria, refinement loops, etc.

On output:

- ***epsout*** – Error on trace.
- ***loop*** – Number of refinement loops executed.
- ***e*** – Eigenvalues found in the search interval.
- ***x*** – Eigenvectors.
- ***m*** – Number of eigenvalues found.
- ***res*** – relative residuals.

Optimization Notice

Performance



Optimization Notice

Intel® MKL Provides Optimized Mathematical Building Blocks

Linear Algebra

- BLAS
- LAPACK
- Sparse Solvers
  - Iterative
  - Pardiso*
- ScaLAPACK

Fast Fourier Transforms

- Multidimensional
- FFTW interfaces
- Cluster FFT

Vector Math

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

Vector RNGs

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

Summary Statistics

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

And More

- Splines
- Interpolation
- Trust Region
- Fast Poisson Solver

Optimization Notice

Additional Resources

See PARDISO and Sparse BLAS in action:

- Code sample: Finding an approximate solution to a stationary nonlinear heat equation
  http://pages.cs.wisc.edu/~holzer/cs412/lecture03.pdf

Intel® MKL Cookbook:

- https://software.intel.com/en-us/mkl_cookbook

Intel® MKL User Forum:

- https://software.intel.com/en-us/forums/intel-math-kernel-library

Intel® MKL product site:

- https://software.intel.com/en-us/intel-mkl

Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Optimization Notice

intel®