Intel® Math Kernel Library (Intel® MKL)

Optimization Notice

Legal Disclaimer

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

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

BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino Atom, Centrino Atom Inside, Centrino Inside, Centrino logo, Cilk, Core Inside, FlashFile, i960, InstantIP, Intel, the Intel logo, Intel386, Intel486, IntelDX2, IntelDX4, IntelSX2, Intel Atom, Intel Atom Inside, Intel Core, Intel Inside, Intel Inside logo, Intel. Leap ahead., Intel. Leap ahead. logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel Viiv, Intel vPro, Intel XScale, Itanium, Itanium Inside, MCS, MMX, Oplus, OverDrive, PDCharm, Pentium, Pentium Inside, skoool, Sound Mark, The Journey Inside, Viiv Inside, vPro Inside, VTune, Xeon, and Xeon Inside are trademarks of Intel Corporation in the U.S. and other countries.

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

http://intel.com/software/products

Optimization Notice

Optimization Notice

Intel compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the "Intel Compiler User and Reference Guides" under "Compiler Options." Many library routines that are part of Intel compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel compilers, associated libraries and associated development tools 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 Intel Streaming SIMD Extensions 2 (Intel SSE2), Intel Streaming SIMD Extensions 3 (Intel SSE3), and Supplemental Streaming SIMD Extensions 3 (Intel 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.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us know if you find we do not.

Notice revision #20110228

Agenda

– Why Intel® MKL and Why is it Faster?
– Overview of MKL
– MKL environment
– The Library Sections
– Linking with Intel® MKL
– Threading in Intel® MKL

Agenda

– Why Intel® MKL and Why is it Faster?
– Overview of MKL
– MKL environment
– The Library Sections
– Linking with Intel® MKL
– Threading in Intel® MKL

Why Intel® Math Kernel Library?

- Performance, Performance, Performance!
- Intel's engineering, scientific, and financial math library
- Addresses:
  - Linear equation Solvers (BLAS, LAPACK)
  - Eigenvector/eigenvalue solvers (BLAS, LAPACK)
  - Some quantum chemistry needs (dgemm from BLAS)
  - PDEs, signal processing, seismic, solid-state physics (FFTs)
  - General scientific, financial [vector transcendental functions (VML) and vector random number generators (VSL)]
  - Sparse Solvers (PARDISO - DSS and ISS)
- Tuned for Intel® processors – current and future
- Intel® AVX optimizations
  - All BLAS level 3 functions, LU/Cholesky/QR & eigensolvers in LAPACK, FFTs of lengths 2^n, Mixed Radix FFTs (3, 5, 7), VML/VSL

Application Areas which could use MKL

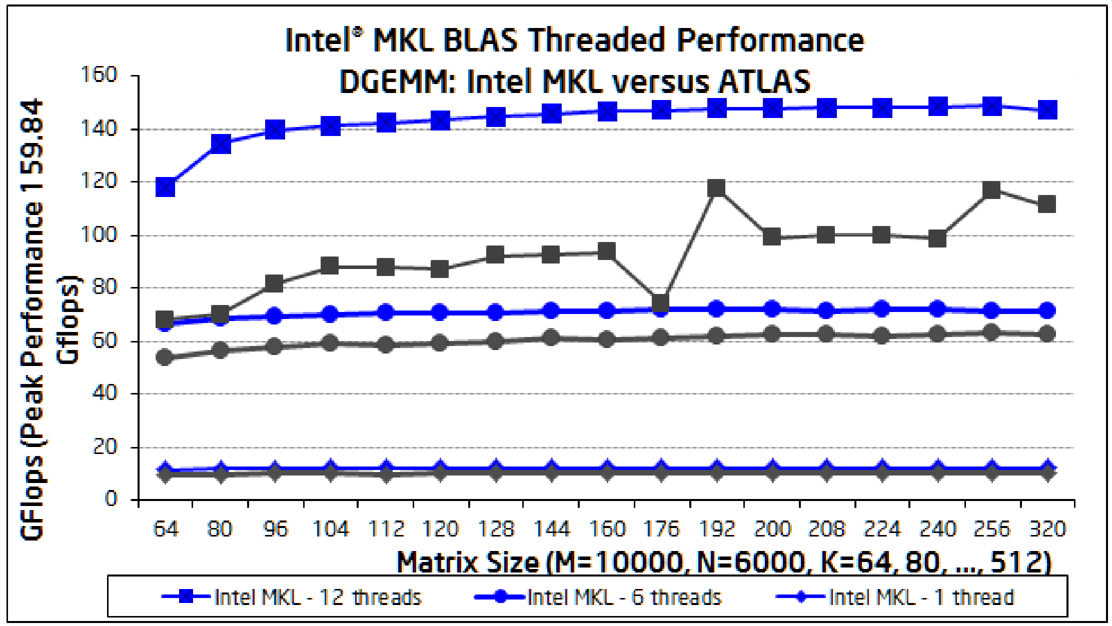
- **Energy** - Reservoir simulation, Seismic, Electromagnetics, etc.
- **Finance** - Options pricing, Mortgage pricing, financial portfolio management etc.
- **Manufacturing** - CAD, FEA etc.
- **Applied mathematics**
  - Linear programming, Quadratic programming, Boundary value problems, Nonlinear parameter estimation, Homotopy calculations, Curve and surface fitting, Numerical integration, Fixed-point methods, Partial and ordinary differential equations, Statistics, Optimal control and system theory
- **Physics & Computer science**
  - Spectroscopy, Fluid dynamics, Optics, Geophysics, seismology, and hydrology, Electromagnetism, Neural network training, Computer vision, Motion estimation and robotics
- **Chemistry**
  - Physical chemistry, Chemical engineering, Study of transition states, Chemical kinetics, Molecular modeling, Crystallography, Mass transfer, Speciation
- **Engineering**
  - Structural engineering, Transportation analysis, Energy distribution networks, Radar applications, Modeling and mechanical design, Circuit design
- **Biology and medicine**
  - Magnetic resonance applications, Rheology, Pharmacokinetics, Computer-aided diagnostics, Optical tomography
- **Economics and sociology**
  - Random utility models, Game theory and international negotiations, Financial portfolio management

Why is Intel® MKL faster?

- Optimization done for maximum speed.
- Resource limited optimization – exhaust one or more resources of the system:
  - **CPU**: Register use, FP units.
  - **Cache**: Keep data in cache as long as possible; deal with cache interleaving.
  - **TLBs**: Maximally use data on each page.
  - **Memory bandwidth**: Minimally access memory.
  - **Computer**: Use all the processor cores available using threading.
  - **System**: Use all the nodes available.
- Intel tuning experts optimize MKL for latest platforms

BLAS Performance – multiple threads

- Performance (DGEMM function)
- Excellent scaling on multiprocessors
- Intel® MKL performs far better than ATLAS* on multi-core
- Optimized for next-gen processors (Intel® AVX support)



Configuration Info
- Versions: Intel® MKL 10.3 ATLAS 3.8.3
- Hardware: 2 Socket 6-Core Intel® Xeon® Processor X5680 3.33Ghz, 12MB L2 cache, 18GB Memory
- OS: Fedora 9 x86_64

Agenda

– Intel® MKL System requirements, Installation and environment
– Why Intel® MKL and Why is it Faster?
– Overview of MKL
– MKL environment
– The Library Sections
– Linking with Intel® MKL
– Threading in Intel® MKL

Intel® MKL Contents

- BLAS
  - Basic vector-vector/matrix-vector/matrix-matrix computation routines.
- Sparse BLAS
  - BLAS for sparse vectors/matrices
- LAPACK (Linear algebra package)
  - Solvers and eigensolvers. Many hundreds of routines total!
  - C interface to LAPACK New
- ScaLAPACK
  - Computational, driver and auxiliary routines for distributed-memory architectures
- DFTs (General FFTs)
  - Mixed radix, multi-dimensional transforms
- Cluster DFT
  - For Distributed Memory systems
- Sparse Solvers (PARDISO, DSS and ISS)
  - Direct and Iterative sparse solvers for symmetric, structurally symmetric or non-symmetric, positive definite, indefinite or Hermitian sparse linear system of equations
  - Out-Of-Core (OOC) version for huge problem sizes

Intel® MKL Contents

- VML (Vector Math Library)
  - Set of vectorized transcendental functions, most of libm functions, but faster
- VSL (Vector Statistical Library)
  - Set of vectorized random number generators
  - SSL (Summary Statistical Library) : Computationally intensive core/building blocks for statistical analysis
- PDEs (Partial Differential Equations)
  - Trigonometric transform and Poisson solvers.
- Optimization Solvers
  - Solvers for nonlinear least square problems with/without boundary condition
- Support Functions

New

Agenda

– Intel® MKL System requirements, Installation and environment
– Why Intel® MKL and Why is it Faster?
– Overview of MKL
– MKL environment
– The Library Sections
– Linking with Intel® MKL
– Threading in Intel® MKL

Intel® Math Kernel Library Contents

- Data types supported:
  - Single precision Real and Complex
  - Double precision Real and Complex
- Examples
- C/C++, Fortran and now a few Java examples
- Well documented

- Note: Finding the correct link line can be a bit of a pain.

Intel® MKL Environment

	Windows*	Linux*	Mac OS*
Compiler	Intel, CVF, Microsoft	Intel, Gnu	Intel, Gnu
Libraries	.lib, .dll	.a, .so	.a, .dylib

- 32bit and 64 bit libraries to support 32-bit and 64-bit Intel® processors
- Static and Runtime dynamic libraries

Language Support			
Domain	Fortran 77	Fortran 95/99	C/C++
BLAS	X	X	Via CBLAS
Sparse BLAS Level 1	X	X	Via CBLAS
Sparse BLAS level 1&2	X	X	X
LAPACK	X	X	X
ScaLAPACK	X		
PARDISO	X	X	X
DSS & ISS	X	X	X
VML/VSL	X	X	X
FFT/Cluster FFT		X	X
PDEs		X	X
Optimization (TR) Solvers	X	X	X
SSL	X	X	X

X - supported

Agenda

– Intel® MKL System requirements, Installation and environment
– Why Intel® MKL and Why is it Faster?
– Overview of MKL
– MKL environment
– The Library Sections
– Linking with Intel® MKL
– Threading in Intel® MKL

Intel® MKL: BLAS

- BLAS (Basic Linear Algebra Subroutines)
  - Level 1 BLAS
    - Vector-vector operations
    - Dot products, swap, min, max, scaling, rotation etc.
  - Level 2 BLAS
    - Matrix-vector operations
    - Matrix-vector products, Rank 1, 2 updates, Triangular solvers etc.
    - *?GEM2V - New functionality that performs a matrix-vector product with a symmetric matrix in blocked storage*
  - Level 3 BLAS
    - Matrix-matrix operations
    - Matrix-matrix products, Rank-k, 2k updates, Triangular solvers etc.
  - Sparse BLAS
    - BLAS Level 1, 2 & 3 for sparse vectors and matrices

- Matrix Storage Schemes:
  - **BLAS:** Full, Packed and Banded Storage
  - **Sparse BLAS:** CSR and its variations, CSC, *coordinate, diagonal, skyline* storage, formats, BSR and its variations.

Matrix Multiplication

Roll Your Own

```
for (i = 0; i < N; i++) {
  for (j=0; j<N; j++) {
    for (k=0; k<N; k++) {
      c[N*i+j] += a[N*i+k] * b[N*k+j];
    }
  }
}
```

Matrix Multiplication

ddot from BLAS Level 1

```
for (i = 0; i < N; i++) {
  for (j=0; j<N; j++) {
    c[N*i+j] =cblas_ddot(N,&a[N*i],incx,&b[j],incy);
  }
}
```

Matrix Multiplication

dgemv from BLAS Level 2

```
for (i = 0; i < N; i++) {
  cblas_dgemv(CblasRowMajor, CblasNoTrans, N, N,
              alpha, a, N, &b[i],N,beta,&c[i],N);
}
```

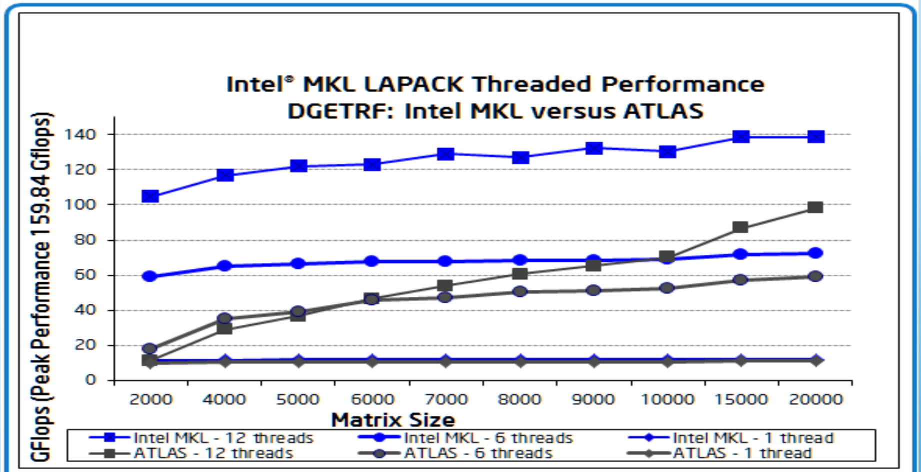
Matrix Multiplication

dgemm from BLAS Level 3

```
cblas_dgemm(CblasRowMajor, CblasNoTrans,
            CblasNoTrans, N, N, N, alpha, b, N, a,
                                    N, beta, c, N);
```

Intel® MKL: LAPACK

- **Routines for:**
  - Solving systems of linear equations, factoring and inverting matrices, and estimating condition numbers.
  - Solving least squares, eigenvalue and singular value problems, and Sylvester's equations.
  - Auxiliary and utility tasks.
  - Callback functions
- Driver Routines: To solve a particular problem, call two or more computational routines or call a driver routine that combines several tasks in one call
- **Most important LAPACK optimizations:**
  - Recursive factorization
    - Reduces scalar time (Amdahl's law: t = tscalar + tparallel/p)
    - Extends blocking further into the code
- No runtime library support required



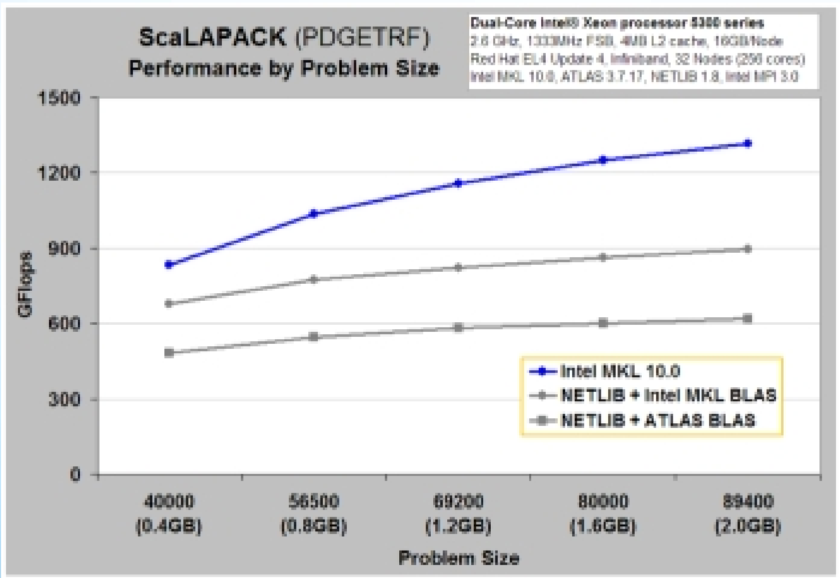
Configuration Info
- Versions: Intel® MKL 10.3 ATLAS 3.8.3
- Hardware: 2 Socket 6-Core Intel® Xeon® Processor X5680 3.33Ghz, 12MB L2 cache, 18GB Memory
- OS: Fedora 9 x86_64

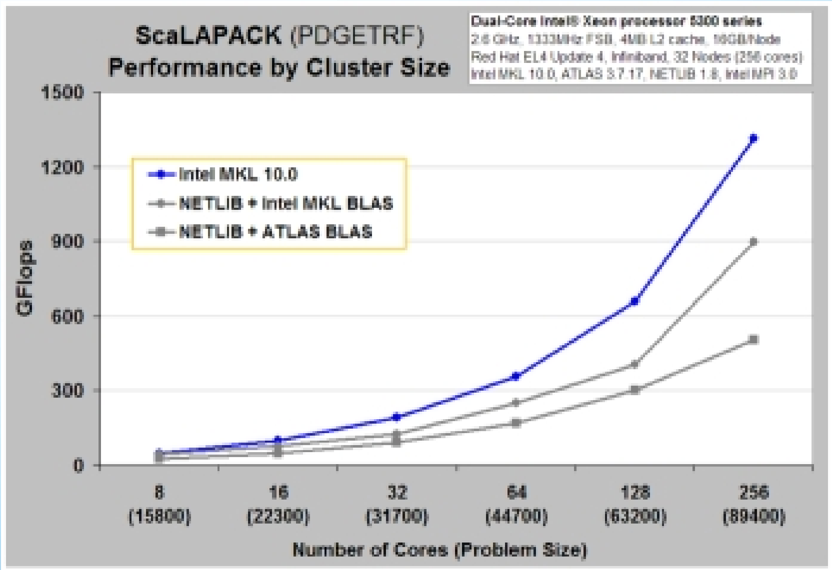
C interface to LAPACK - Functionality

- Covers all LAPACK functionality – driver and computational routines
  - LAPACK: dgetrf
  - C interface: CLAPACK_dgetrf (High-level interface)
- Both Row-major and Column-major layout supported, chosen by first argument
- Native C interface - input scalars are passed by value
- Integer return value corresponds to 'info' parameter in LAPACK
- Both LP64/ILP64 supported

Intel® MKL: ScaLAPACK

- LAPACK for distributed memory architectures
- Using MPI, BLACS and a set of BLAS
- Uses 2D block cyclic data distribution for dense matrix computations which helps
  - Better work balance between available processors
  - Use BLAS level 3 for optimized local computations



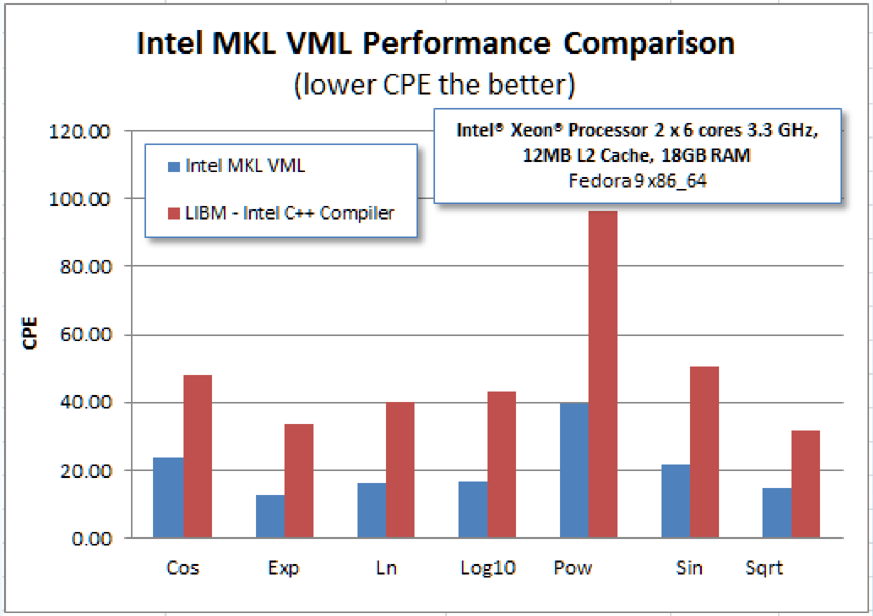


Intel® MKL: Sparse Solvers

- **PARDISO –** Parallel Direct Sparse Solver
  - For SMP systems
  - High performance, robust and memory efficient
  - Based on Level-3 BLAS update and pipelining parallelism
  - OOC version for huge problem sizes
  - C-style 0 based indexing for PARDISO
- **DSS –** Direct Sparse Solver Interface to PARDISO
  - Alternative to PARDISO
  - Steps: Create ->Define Array Struct->reorder->factor->solve->Delete
- **ISS –** Iterative Sparse Solver
  - RCI based
  - For symmetric positive definite and for non-symmetric indefinite systems

Intel® MKL: Vector Math Library (VML)

- Highly optimized implementations of computationally expensive core mathematical functions (power, trigonometric, exponential, hyperbolic etc.)
- Operates on a vector unlike libm.
- Multiple accuracy modes
  - High accuracy (HA) ~53 bits accurate
  - Lower accuracy (LA), faster ~51 bits accurate
  - Enhanced Performance (EP) ~26 bits accurate
  - Routine-level mode controls
- New VML overflow reporting feature
- Denormal paths speedup via VML FTZ/DAZ setting



- Special value handling √(-a), sin(0), and so on
- Can improve performance of non-linear programming and integrals in applications.

Intel® MKL: Vector Statistical Library (VSL)

- Functions for:
  - Generating vectors of pseudorandom and quasi-random numbers
  - Convolution & Correlation
- Parallel computation support – some functions
- User can supply own BRNG or transformations

Basic RNGs	
Pseudo RNGs	Quasi RNGs
MCG31, GFSR250, MRG32, MCG59, WH, MT19937, MT2203	Sobol-quasi, Niederreiter quasi

Distribution Generators	
Continuous	Discrete
Uniform, Gaussian (two methods), Exponential, Laplace, Weibull, Cauchy, Rayleigh, Lognormal, Gumbel, Gamma, Beta	Uniform, UniformBits, Bernoulli, Geometric, Binomial, Hypergeometric, Poission, PoissonV, NegBinomial

Performance Comparison of Random Number Generator		
Intel Xeon Processor	Running time (seconds)	Speedup vs. rand() (times)
Standard C rand() function	25.28	1.00
Intel* MKL VSL random number generator	5.60	4.51
Intel® MKL + OpenMP version (12 threads)	0.73	34.63

Configuration Info:
- Intel® Xeon® processor 2x6 Cores, 3.3 GHz
- 12MB L2 cache, 18 GB memory
- Fedora 9 X86_64
- Intel® MKL 10.3

Using VSL

3-step Process and an optional 4th Step

1. Create a stream pointer.
   ```
   VSLStreamStatePtr stream;
   ```
2. Create a stream.
   ```
   vslNewStream(&stream,VSL_BRNG_MC_G31,seed);
   ```
3. Generate a set of RNGs.
   ```
   vslRngUniform(0,&stream,size,out,start,end);
   ```
4. Delete a stream (optional).
   ```
   vslDeleteStream(&stream);
   ```

Summary Statistics: Key Functionality & Features

Functionality

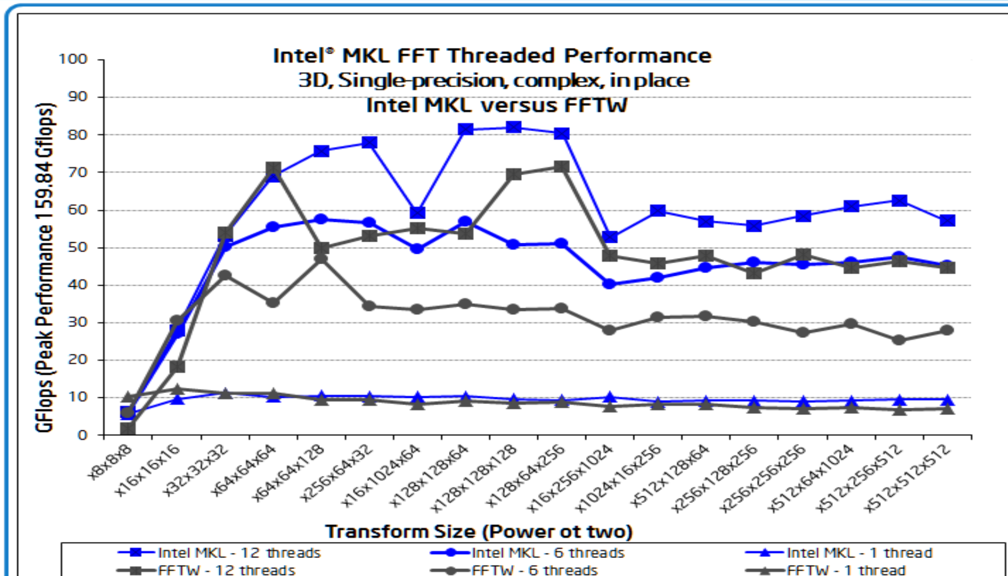
- Basic statistics
  - Moments, skewness, kurtosis, variation coefficient, quantiles and order statistics.
- Estimation of Dependencies
  - Variance-covariance/correlation matrix, partial variance-covariance/correlation matrix, pooled/group variance-covariance/correlation matrix.
- Data with Outliers
  - Detection of outliers in "noised" data, robust (to noise) estimates of the covariance matrix and mean
- Missing Values
  - Restoring statistical characteristics in presence of missed observations

Features

- Out-of-Memory Datasets
  - Addresses cases when data comes in portions or when whole dataset doesn't fit into memory
- Various Data Storage Formats
  - Flexibility for users, in-row/in-column packed, full/packed matrix
- Enhanced accuracy and performance due to modern algorithms

Intel® MKL: Fast Fourier Transform (FFT)

- 1, 2 & 3 dimensional
- Multithreaded
- Mixed radix
- User-specified scaling, transform sign
- Multiple one-dimensional transforms on single call
- Strides
- Supports FFTW interface through wrappers
- Split Complex (real real) support



Configuration Info
- Versions: Intel® MKL 10.3
- Hardware: 2 Socket 6-Core Intel® Xeon® Processor X5680 3.33Ghz, 12MB L2 cache, 18GB Memory
- OS: Fedora 9 x86_64

Using the Intel® MKL DFTs

Basically a 3-step Process

1. Create a descriptor.

```
Status = DftiCreateDescriptor(MDH, …)
```

2. Commit the descriptor (instantiates it).

```
Status = DftiCommitDescriptor(MDH)
```

3. Perform the transform.

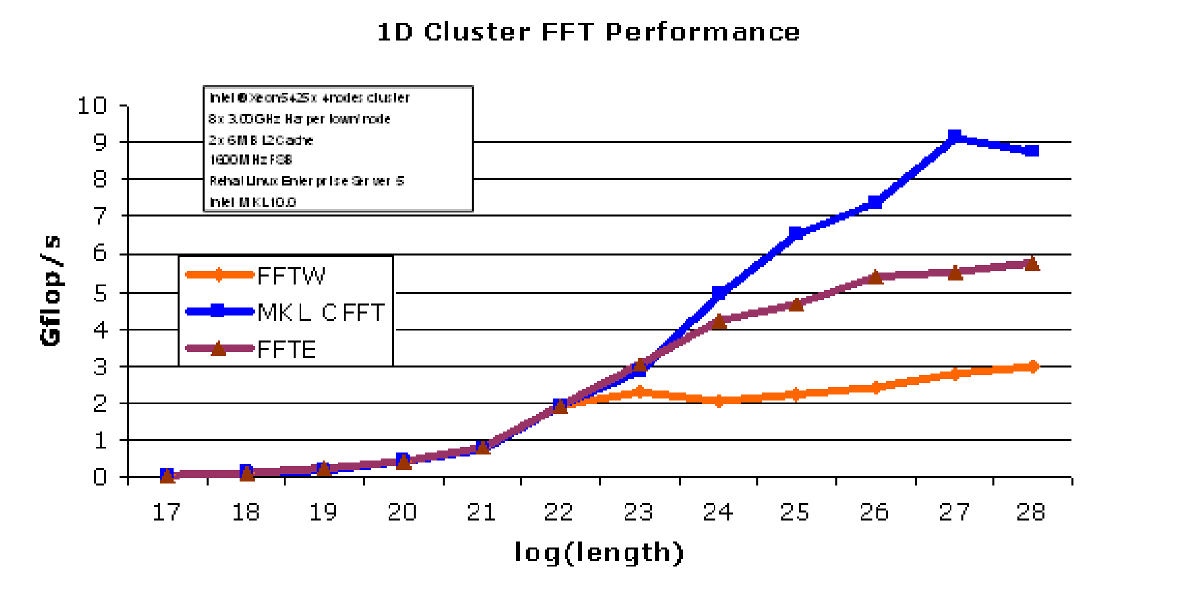
```
Status = DftiComputeForward(MDH, X)
```

- Optionally free the descriptor.

MDH: MyDescriptorHandle

Intel® MKL: Cluster FFT

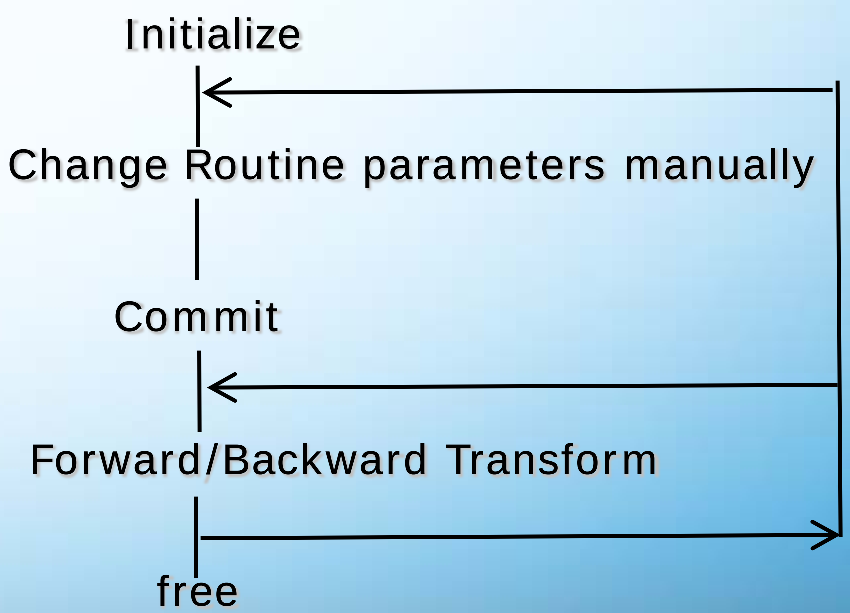
- FFT for Distributed memory systems/clusters
- Works with MPI using BLACS
- OpenMP Support since 10.3
- 1, 2, 3 and multidimensional
- Requires basic MPI programming skills
- Same interface as the DFT from standard MKL
- FFTW Support



Intel® MKL: Partial Differential Equations

- **Poisson Library**
  - for fast solving of simple Helmholtz, Poisson, and Laplace problems
- **Trigonometric Transform interface routines**

```
?_init_trig_transform
?_commit_trig_transform
?_forward_trig_transform
?_backward_trig_transform
free_trig_transform
```



Intel® MKL: Support Functions

- Intel® MKL support functions are used to:
  - retrieve information about the current Intel® MKL version
  - additionally control the number of threads
  - handle errors
  - test characters and character strings for equality
  - measure user time for a process and elapsed CPU time
  - set and measure CPU frequency
  - free memory allocated by Intel® MKL memory management software

Agenda

- Intel® MKL System requirements, Installation and environment
- Why Intel® MKL and Why is it Faster?
- Overview of MKL
- MKL environment
- The Library Sections
- Linking with Intel® MKL
- Threading in Intel® MKL

Linking with Intel® MKL

- Static Linking
- Dynamic linking
- Custom Dynamic Linking
- Dynamic Libraries

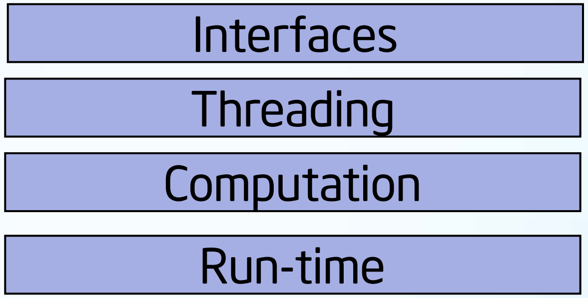
Quick Comparison of Intel® MKL Linkage Models				
Feature	Dynamic Linkage	Static Linkage	Custom Dynamic Linkage	Dynamic Libraries
Processor Dispaches	Automatic	Automatic	Recompile and redistribute	Automatic
Optimization	All Processors	All Processors	All Processors	All Processors
Build	Link to import libraries	Link to static libraries	Build separate import libraries, which are created automatically	Link only to mkl_rt library (Linux – libmkl_rt.so* Mac OS – libmkl_rt.dylib)
Calling	Regular Names	Regular Names	Regular Names	Regular Names
Total Binary Size	Large	Small	Small	Largest
Executable Size	Smallest	Small	Smallest	Smallest
Multi-threaded/ thread safe	Yes	Yes	Yes	Yes

- *In Dynamic libraries linking for Linux, correspondent OMP libs should also be linked except in sequential case.

Linking with Intel® MKL contd..

Layered model approach for better control

- Interface Layer
  - LP64 / ILP64
- Threading Layer
  - Threaded / alternate OpenMP
  - Sequential
- Computational Layer
- Run-time Layer



Ex 1: Static linking using Intel® Fortran Compiler, BLAS, Intel® 64 processor on Linux

```
$ifort myprog.f libmkl_intel_lp64.a libmkl_intel_thread.a libmkl_core.a  libiomp5.so
```

Ex 2: Dynamic linking with Intel® C++ compiler on Windows*

```
c:\>icl myprog.c mkl_intel_lp64_dll.lib mkl_intel_thread_dll.lib mkl_core_dll.lib
libiomp5md.dll
```

Ex 3: Using MKL Dynamic Interface with Intel® C++ compiler on Mac*

```
$icc myprog.c libmkl_rt.dylib
```

Note: Strongly recommended to link Run-time layer library dynamically

A link line advisor tool is available at : http://software.intel.com/en-us/articles/intel-mkl-link-line-advisor/

MKL Link Line Advisor

Intel® Math Kernel Library Link Line Advisor — Reset

Select Intel MKL/Intel Compiler version:	Intel(R) MKL 10.3
Select OS:	Linux*
Select processor architecture:	Intel(R) 64
Select compiler:	Intel C/C++
Select dynamic or static linking:	Dynamic
Select interface layer:	LP64 (32-bit integer)
Select sequential or multi-threaded version of Intel MKL:	Sequential
Select OpenMP library:	<Select OpenMP>

Use this link line:

```
-L$(MKLROOT)/lib/intel64 -lmkl_intel_lp64 -lmkl_sequential -lmkl_core -lpthread
```

Compiler options:

Agenda

– Intel® MKL System requirements, Installation and environment
– Why Intel® MKL and Why is it Faster?
– Overview of MKL
– MKL environment
– The Library Sections
– Linking with Intel® MKL
– Threading in Intel® MKL

Threading in Intel MKL - Domains and Parallelism

	Where's the Parallelism?		
Domain	**SIMD**	**Open MP**	**MPI**
BLAS 1, 2, 3	X	X	
FFTs	X	X	
LAPACK (dense LA solvers)	X (relies on BLAS 3)	X	
ScaLAPACK (cluster dense LA solvers)		X (hybrid)	X
PARDISO (sparse solver)	X (relies on BLAS 3)	X	
VML/VSL	X	X	
Cluster FFT		X	X

Threading Control in Intel® MKL

- Set OpenMP or Intel® MKL environment variable:
  - `OMP_NUM_THREADS`
  - `MKL_NUM_THREADS`
  - `MKL_DOMAIN_NUM_THREADS`
- Call OpenMP or Intel® MKL using
  - `omp_set_num_threads()`
  - `mkl_set_num_threads()`
  - `mkl_domain_set_num_threads()`

  `MKL_DYNAMIC/mkl_set_dynamic()`: Intel® MKL decides the number of threads.
- Example: You could configure Intel® MKL to run 4 threads for BLAS, but sequentially in all other parts of the library
  - Environment variable

    ```
    set MKL_DOMAIN_NUM_THREADS="MKL_ALL=1, MKL_BLAS=4"
    ```
  - Function calls

    ```
    mkl_domain_set_num_threads( 1, MKL_ALL);
    mkl_domain_set_num_threads( 4, MKL_BLAS);
    ```

References

- Intel® MKL product Information
  - www.intel.com/software/products/mkl
- MKL Knowledge Base
  - http://software.intel.com/en-us/articles/intel-mkl-kb/all
- User Discussion Forum
  - http://software.intel.com/en-us/forums/intel-math-kernel-library/
- Technical Issues/Questions/Feedback
  - http://premier.intel.com/