Intel® Math Kernel Library

Software & Services Group

Intel Corporation

Agenda

- Intel® MKL system requirements and installation
- Why Intel MKL?
- Overview of Intel MKL
- Intel MKL environment
- The Library Components in details
- Linking with Intel MKL
- Threading in Intel MKL
- Lab

Agenda

- Intel® MKL system requirements and installation
- Why Intel MKL?
- Overview of Intel MKL
- Intel MKL environment
- The Library Components in details
- Linking with Intel MKL
- Threading in Intel MKL
- Lab

Intel MKL Supported Operating Systems

Intel MKL has been tested with the following Operating Systems

Windows* versions

- Windows* 7 (IA-32/Intel®64) : SP1 is required for use of Intel® AVX instructions
- Windows Server* 2008 (IA-32/Intel®64)
- Windows* Vista* (IA-32/Intel®64)
- Windows Server* 2003 (IA-32/Intel®64)
- Windows* XP* (IA-32)
- Windows* XP* Pro x64 Edition (Intel®64)
- Windows Compute Cluster Server* 2003 (Intel® 64)

The following Linux* distributions are supported

- Red Hat* Enterprise Linux* 4, 5, 6 (IA-32/Intel®64)
- SUSE LINUX Enterprise Server* 9, 10, 11 (IA-32/Intel®64)
- SGI ProPack* for Linux 5, 6 (Intel®64)
- Red Hat Fedora* core 12, 13, 15 (IA-32/Intel®64)
- Debian* GNU/Linux 4, 5, 6 (IA-32/Intel® 64)
- Ubuntu* 9.10.04, 11.04 (IA-32/Intel®64)
- Asianux* Server 3 (IA-32/Intel®64)
- Turbolinux* 11 (IA-32/Intel®64)

Mac OS* support

- OS X 10.7 (IA-32/Intel®64) with Xcode 4.3/4.2/4.1
- OS X 10.6.8 (IA-32/Intel®64) with Xcode 4.2/4.0/3.2.5

Note: Intel MKL is expected to work on many more Linux* distributions as well. Let us know if you have trouble with the distribution you use.

Optimization Notice

Intel MKL installation – Linux*

The default top-level installation folder for this product is

/opt/intel/ComposerXE-2011/mkl

This product installs into an arrangement of folders as shown below

	/benchmarks
/opt/intel/ComposerXE-2011/mkl	/bin
	/examples
	/include
	/interfaces
	/lib
	/tests
	/tools

Optimization Notice

Agenda

- Intel® MKL system requirements and installation
- Why Intel MKL?
- Overview of Intel MKL
- Intel MKL environment
- The Library Components in details
- Linking with Intel MKL
- Threading in Intel MKL
- Lab

Why Intel® Math Kernel Library?

Performance, Performance, Performance!

Intel’s engineering, scientific, and financial math library

Addresses:

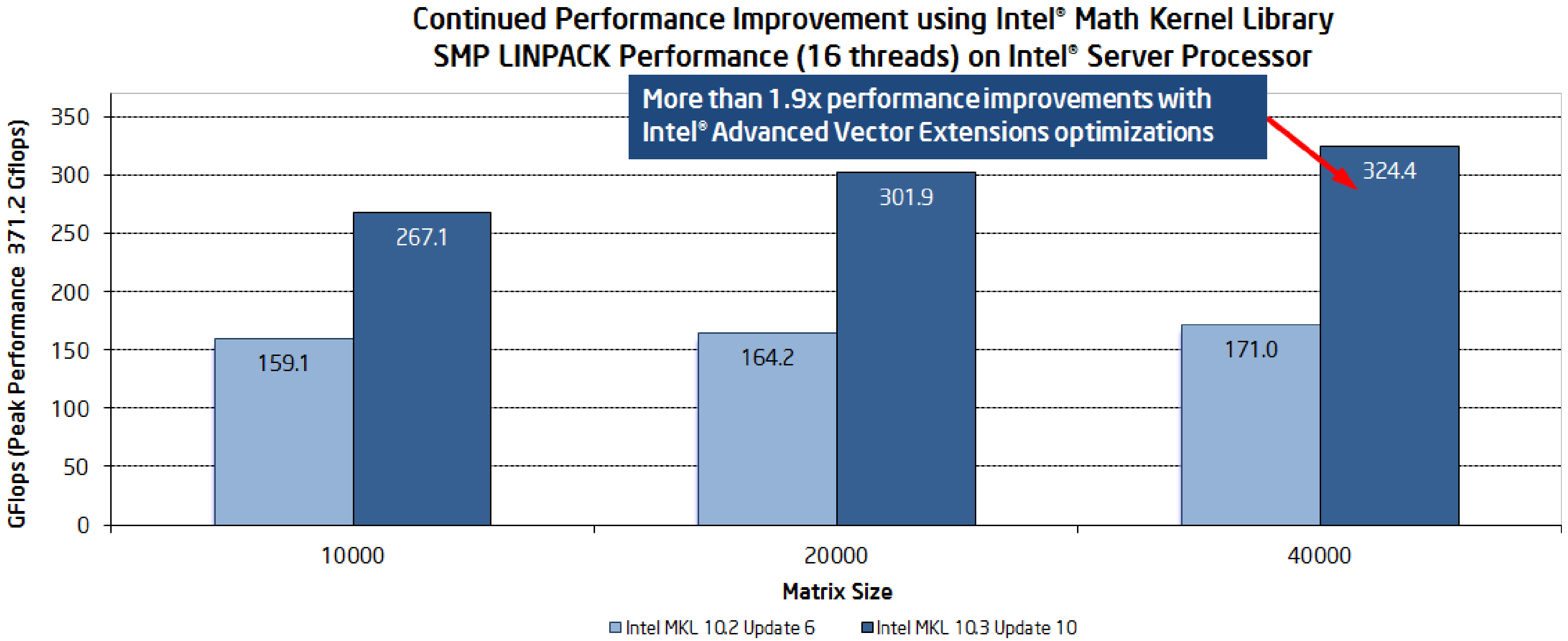
- Basic matrix-vector operations (BLAS)
- Linear equation solvers (LAPACK, PARDISO, ISS)
- Eigenvector/eigenvalue solvers (LAPACK)
- Some quantum chemistry needs (GEMM from BLAS)
- PDEs, signal processing, seismic, solid-state physics (FFTs)
- General scientific, financial - vector transcendental functions (VML) and vector random number generators (VSL)

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

Why Intel® Math Kernel Library?



Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 10.2.6 & 10.3.10; Hardware: Intel® Xeon® Processor E5-2690, 2 Eight-Core CPUs (20Mb L3 Cache, 2.9GHz), 32GB of RAM; Operating System: RHEL 6 GA x86_64; Benchmark Source: Intel Corporation.

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

Intel® MKL: Application Areas

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

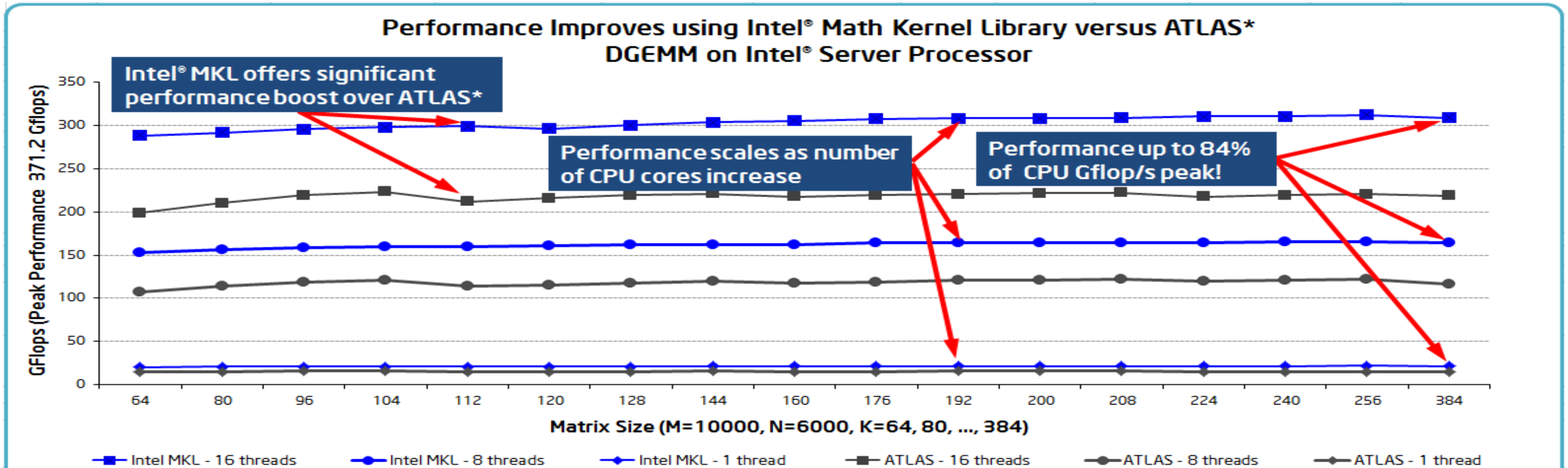
Why Intel® Math Kernel Library?

Resource limited optimization – exhaust one or more resources of a system

- **CPU**: Register, FP units utilization
- **Cache**: Keep data in cache as long as possible, deal with cache interleaving
- **TLB**: Max use of data on each page
- **Memory bandwidth**: Min memory acceses
- **Computer**: Utilize all cores available using threading
- **System**: Utilize all nodes available

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 Intel processors



Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 10.3.10 ATLAS 3.8.4; Hardware: Intel® Xeon® Processor E5-2690, 2 Eight-Core CPUs (20Mb L3 Cache, 2.9GHz), 32GB of RAM; Operating System: RHEL 6 GA x86_64; Benchmark Source: Intel Corporation.

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.

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

Agenda

- Intel® MKL system requirements and installation
- Why Intel MKL?
- Overview of Intel MKL
- Intel MKL environment
- The Library Components in details
- Linking with Intel MKL
- Threading in Intel MKL
- Lab

Intel MKL Components

BLAS (Basic Linear Algebra Subroutines)
- 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

ScaLAPACK
- Computational, driver and auxiliary routines for distributed-memory architectures

DFT/FFT
- Mixed radix, multi-dimensional transforms

Cluster DFT/FFT
- DFTs for distributed memory systems

Sparse Solvers (PARDISO, DSS and ISS)
- Direct for symmetric, structurally symmetric or non-symmetric, positive definite, indefinite or Hermitian sparse linear system of equations
- Out-Of-Core (OOC) version and iterative solvers for huge problem sizes

Intel MKL Components

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

DFL (Data Fitting Library)

- Spline construction
- Spline based interpolation and computation of derivatives, integration
- Cell research

PDEs (Partial Differential Equations)

- Trigonometric transform and Poisson solvers

Optimization Solvers

- Solvers for nonlinear least square problems with/without boundary condition

Support Functions

Intel MKL Contents

Data types supported

- Single precision Real and Complex
- Double precision Real and Complex

Examples

C/C++, Fortran

Well-documented (documentation available online)

Online articles and examples on how to use it with C#, Java, Python, etc.

Intel® MKL Knowledge Base

- http://software.intel.com/en-us/articles/intel-mkl-kb-home/

Intel® MKL Documentation

- http://software.intel.com/en-us/articles/intel-math-kernel-library-documentation/

Agenda

- Intel® MKL system requirements and installation
- Why Intel MKL?
- Overview of Intel MKL
- Intel MKL environment
- The Library Sections
- Linking with Intel MKL
- Threading in Intel MKL
- Lab

Intel MKL Environment

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

32-bit 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	+	+	Via CBLAS
Sparse BLAS Level 1	+	+	Via CBLAS
Sparse BLAS level 1&2	+	+	+
LAPACK	+	+	+
ScaLAPACK	+		
PARDISO	+	+	+
DSS & ISS	+	+	+
VML/VSL	+	+	+
FFT/Cluster FFT		+	+
PDEs		+	+
Optimization (TR) Solvers	+	+	+
SSL	+	+	+

F77 support will be removed from Intel® MKL 11.0 onwards, except for Sparse Solvers

Optimization Notice

Agenda

- Intel® MKL system requirements, installation and environment
- Why Intel MKL?
- Overview of Intel MKL
- Intel MKL environment
- The Library Components in details
- Linking with Intel MKL
- Threading in Intel MKL
- Lab

Optimization Notice

Intel MKL BLAS

BLAS (Basic Linear Algebra Subroutines)

- **Level 1 BLAS** (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 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

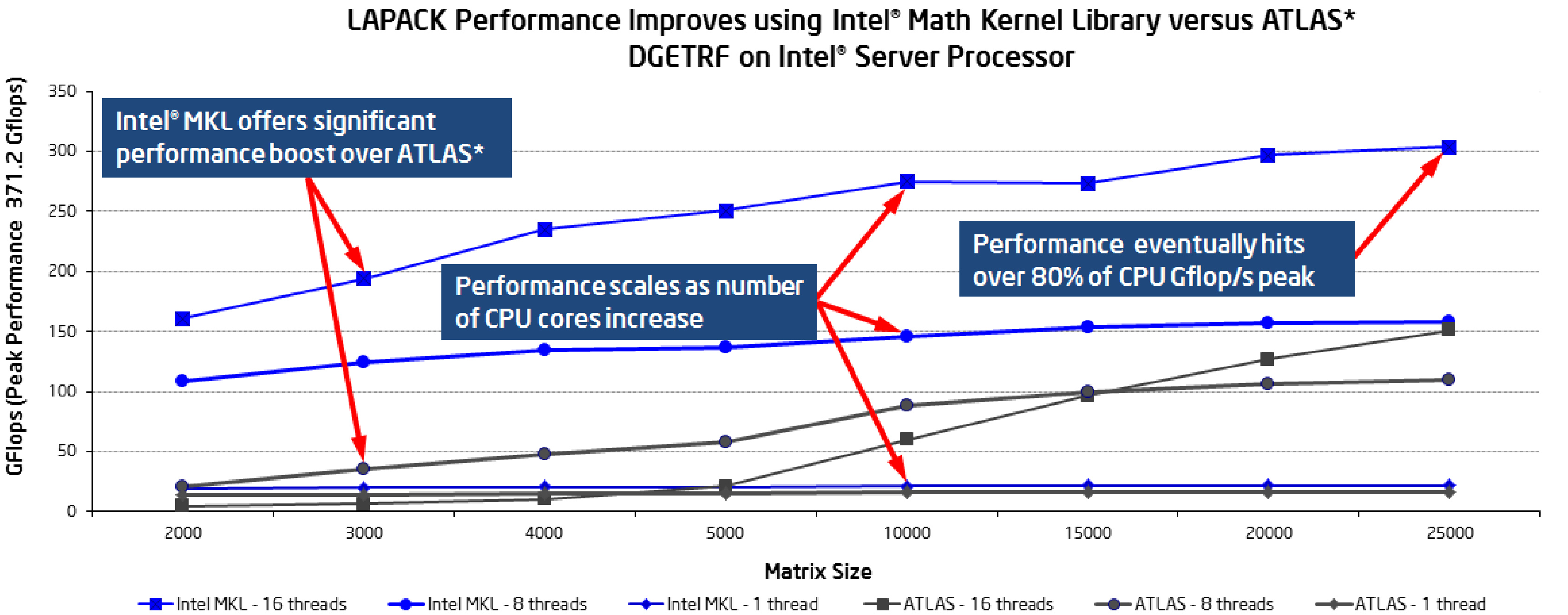
Driver Routines

- 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

Intel MKL LAPACK



Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 10.3.10 ATLAS 3.8.4; Hardware: Intel® Xeon® Processor E5-2690, 2 Eight-Core CPUs (20Mb L3 Cache, 2.9GHz), 32GB of RAM; Operating System: RHEL 6 GA x86_64; Benchmark Source: Intel Corporation.

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

C interface to LAPACK

- 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

- Both LP64/ILP64 supported
  - ILP64 libraries use the 64-bit integer type (necessary for indexing large arrays, with more than 2^{31}-1 elements), whereas the LP64 libraries index arrays with the 32-bit integer type

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
  - Achieve better work balance between available processes
  - Use BLAS level 3 for optimized local computations

Intel MKL BLACS

The BLACS routines implemented in Intel MKL are of four categories

- Combines
- Point-to-Point Communication
- Broadcast
- Support

Intel MKL Sparse Solvers

PARDISO – Parallel Direct Sparse Solver

- For Symmetric Multiprocessing systems
- High performance, robust and memory efficient
- Based on Level 3 BLAS update and pipelining parallelism
- Out-of-Core version for huge problem sizes
- C-style 0-based indexing option

DSS – Direct Sparse Solver Interface to PARDISO

- Alternative to PARDISO
- **Steps**: create ->define Array Struct->reorder->factor->solve->delete

ISS – Iterative Sparse Solver

- Reverse Communication Interface (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 standard library

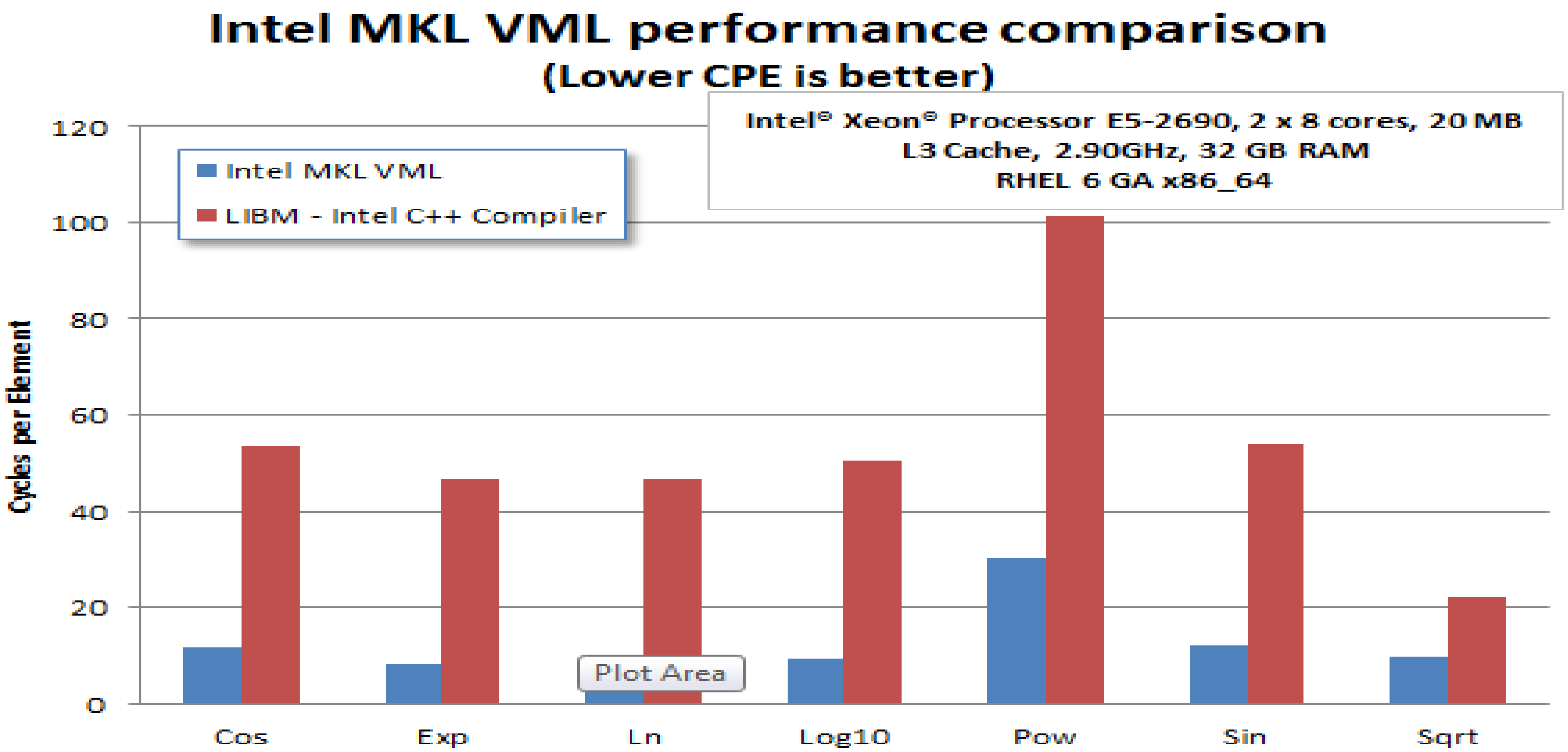
Multiple accuracy modes

- High accuracy (HA) >52 bits accurate for double precision and >23 bits for single preicison
- Lower accuracy (LA), faster >50 bits accurate for double precision >21 bits for single precision
- Enhanced Performance (EP) >25 bits accurate for double precision and >11-12 bits for single precision
- 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 computations applications

Optimization Notice

Intel MKL VML



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.

Refer to our Optimization Notice for more information regarding performance and optimization choices in Intel software products at : http://software.intel.com/en-us/articles/optimization-notice/.

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

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
MCG31m1, GFSR250, MRG32, MCG59, WH, MT19937, MT2203, SFMT19937	Sobol-quasi, Niederreiter-quasi

ABSTRACT: pseudorandom or quasi-random, depending on the user provided settings
NON-DETERMINISTIC: Available in the latest CPUs such as Intel® AVX.

	Running Time (seconds)	Speedup vs. rand() (times)
Standard C rand() function	24.03	1.00
Intel® MKL VSL random number generator	4.16	5.78
OpenMP* version (16 threads)	0.28	85.82

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

Intel® Xeon Processor E5-2690 processor-based system (8 cores, 16 threads total), running at 2.9 GHz with 20MB L3 cache and 32GB memory with Linux* and Intel® C++ Compiler 12.0.
Intel® MKL 10.3.10 VSL Intel® 64 architecture version was used in measurements

Excellent Multi-core Scaling

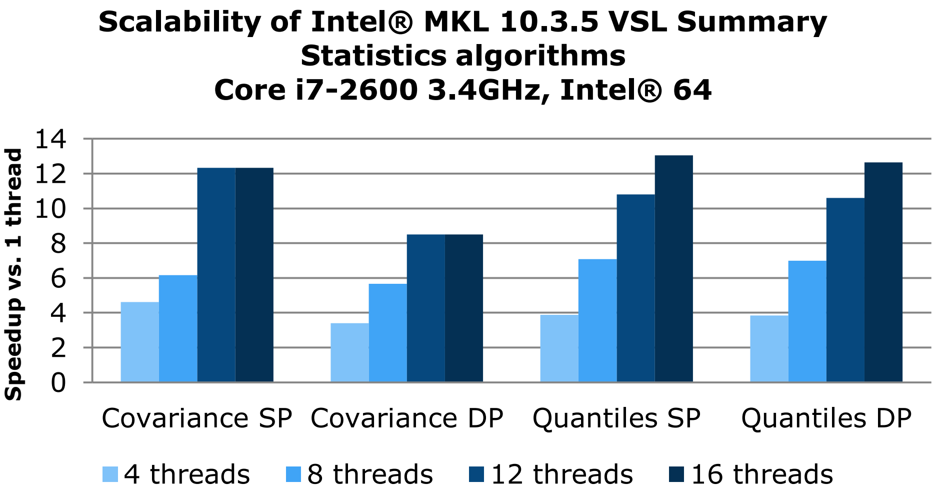
Using Intel MKL VSL

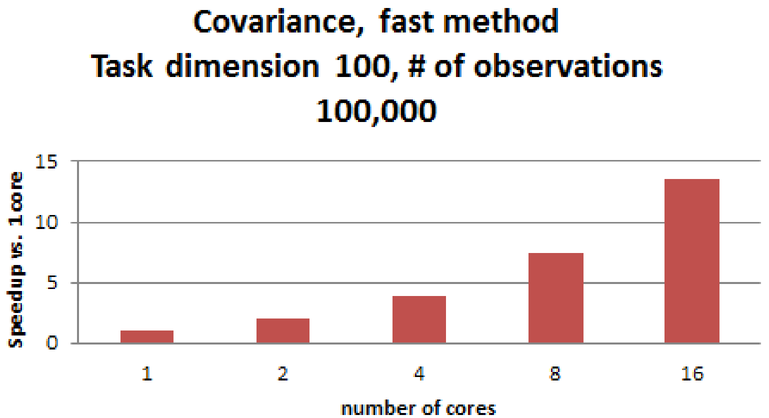
3-step main process and an optional 4th step

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);
```

Intel MKL Summary Statistics: Estimation of Dependencies

- Variance-covariance/correlation matrix
  - Close to linear scaling of covariance estimator
  - Fast method
    - ~40 times faster than R on 8 core Harpertown 2.8 GHz
  - One pass method
    - ~17 times faster then R on 8 core Harpertown 2.8 GHz





Intel® Xeon® Processor E5-2690, 2 x 8 cores, 20 MB L3 Cache, 2.90GHz, 32 GB RAM. RHEL 6 GA x86_64

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.

Refer to our Optimization Notice for more information regarding performance and optimization choices in Intel software products at : http://software.intel.com/en-us/articles/optimization-notice/.

Intel MKL Data Fitting Component

Intel MKL Data Fitting – SW solution for

- Spline construction
- Spline based interpolation and computation of derivatives
- Spline based integration
- Cell Search

Application Areas

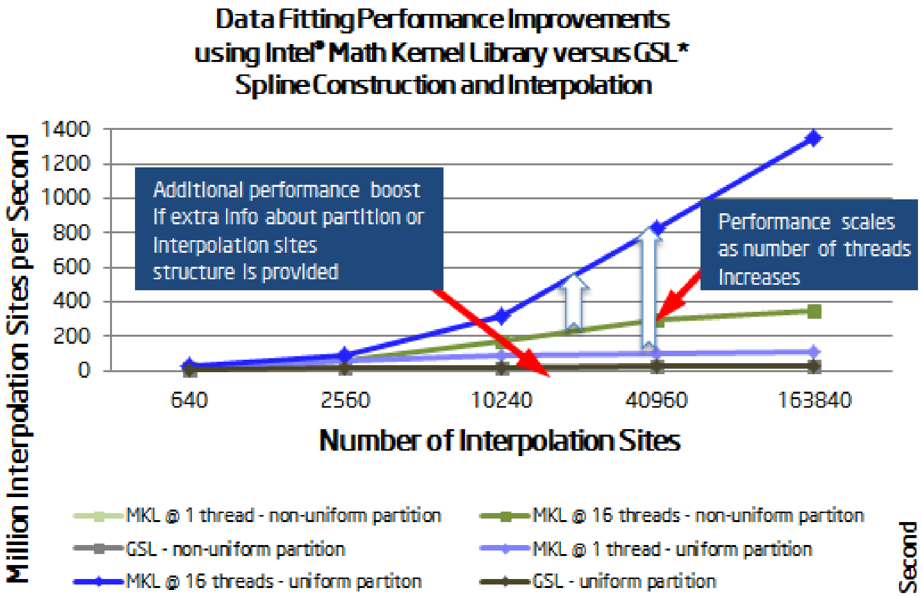
- **Data analysis and analytics :** Approximation of statistical estimates like histogram
- **Manufacturing :** Geometrical modeling
  - *"B-spline recurrence relations … were used at Boeing, …, five hundred million times a day"* Carl de Boor, On Wings of Splines Newsletter of Institute for Mathematical Sciences, ISSUE 5 2004
- **Energy :** Surface approximation
- **ISV :** SW libraries

Data Fitting API and usage model

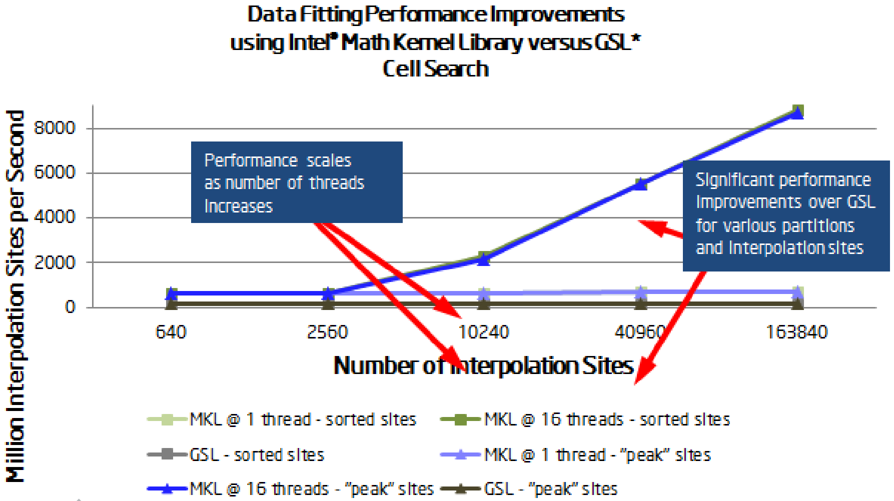
Step	Code example
Create a task	`status = dfdNewTask1D( &task, nx, x, xhint, ny, y, yhint );`
Modify the task parameters.	`status = dfdEditPPSpline1D( task, s_order, c_type, bc_type, bc, ic_type, ic, scoeff, scoeffhint );`
Perform Data Fitting spline-based computations	`status = dfdInterpolate1D(task, estimate, method, nsite, site, sitehint, ndorder, dorder, datahint, r, rhint, cell );`
Destroy the task or tasks	`status = dfDeleteTask( &task );`

API and usage model similar to that in Vector Statistical component, Fourier Transforms in Intel MKL

Intel MKL Data Fitting Performance



Construction of natural cubic spline with free end boundary conditions for function defined on uniform and non-uniform partitions. Partition size is 1280. Spline-based values and first derivatives are computed.



Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 10.3.10 GSL 1.15; Hardware: Intel® Xeon® Processor E5-2690, 2 Eight-Core CPUs (20 MB L3 Cache, 2.90GHz), 32 GB of RAM; Operating System: RHEL 6 GA x86_64; Benchmark Source: Intel Corporation.

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.

Refer to our Optimization Notice for more information regarding performance and optimization choices in Intel software products at : http://software.intel.com/en-us/articles/optimization-notice/.

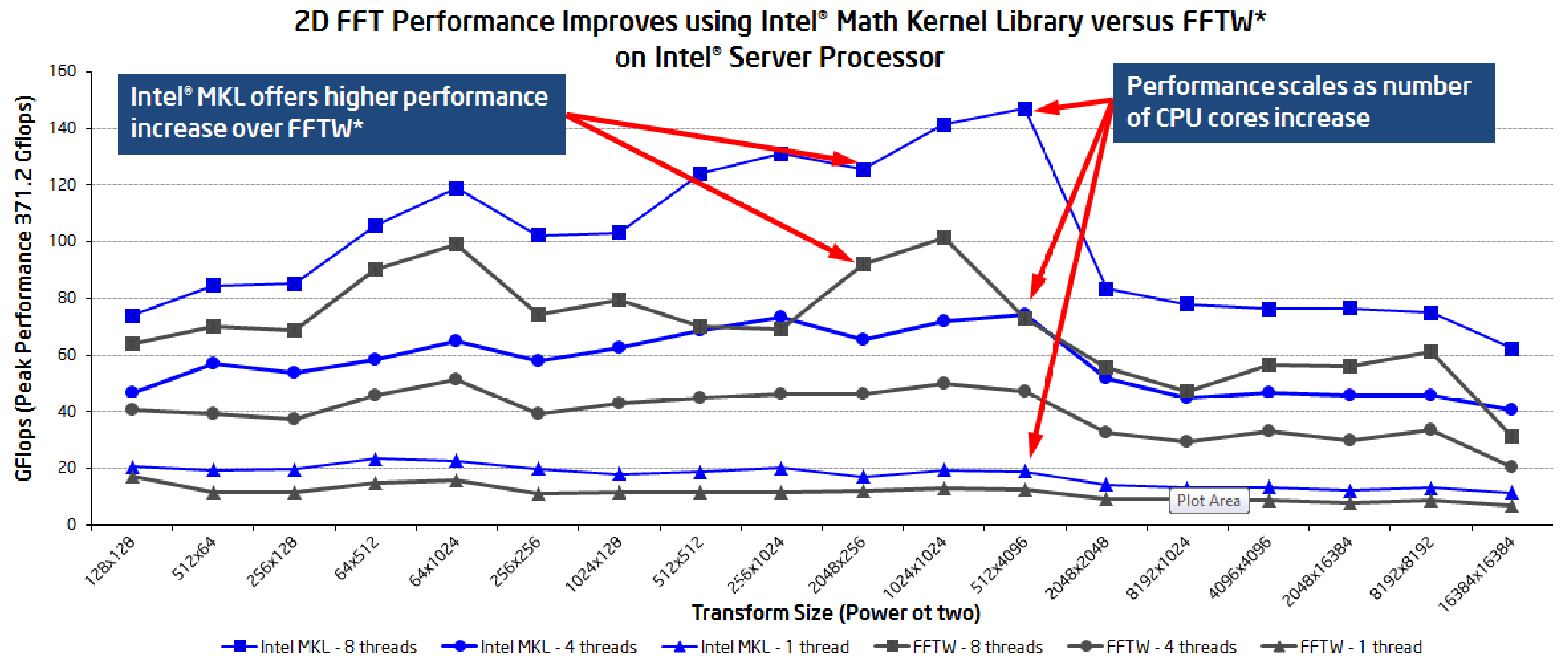
* Other brands and names are the property of their respective owners

Performing cells search on non-uniform partition. Partition size is 1280. Sorted sites - interpolation sites are sorted ; "peak" sites - distribution of interpolation sites has a clear peak.

Intel MKL Fast Fourier Transform (FFT)

- Multidimensional (1, 2, 3, …, 7)
- Multithreaded
- Mixed radix
- User-specified scaling, transform sign
- Multiple one-dimensional transforms in a single call
- Stride support
- Supports FFTW* interface through wrappers
- Split-complex (real real) support for 2D/3D FFTs

Intel MKL FFT



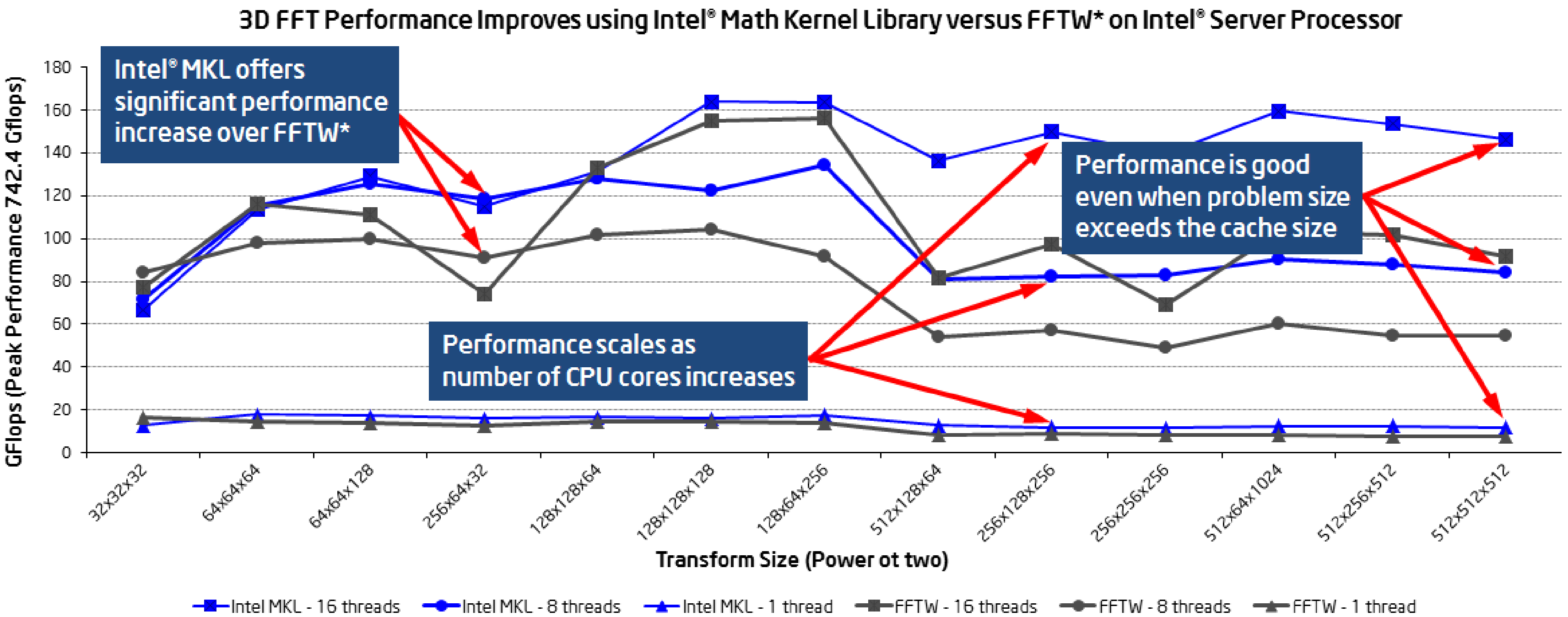
Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 10.3.10 FFTW 3.3; Hardware: Intel® Xeon® Processor E5-2690, 2 Eight-Core CPUs (20Mb L3 Cache, 2.9GHz), 32GB of RAM; Operating System: RHEL 6 GA x86_64: Intel Corporation.

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

Intel MKL FFT



Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 10.3.10 FFTW 3.3; Hardware: Intel® Xeon® Processor E5-2690, 2 Eight-Core CPUs (20Mb L3 Cache, 2.9GHz), 32GB of RAM; Operating System: RHEL 6 GA x86_64: Intel Corporation.

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.

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

Using the Intel MKL FFT

Main 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

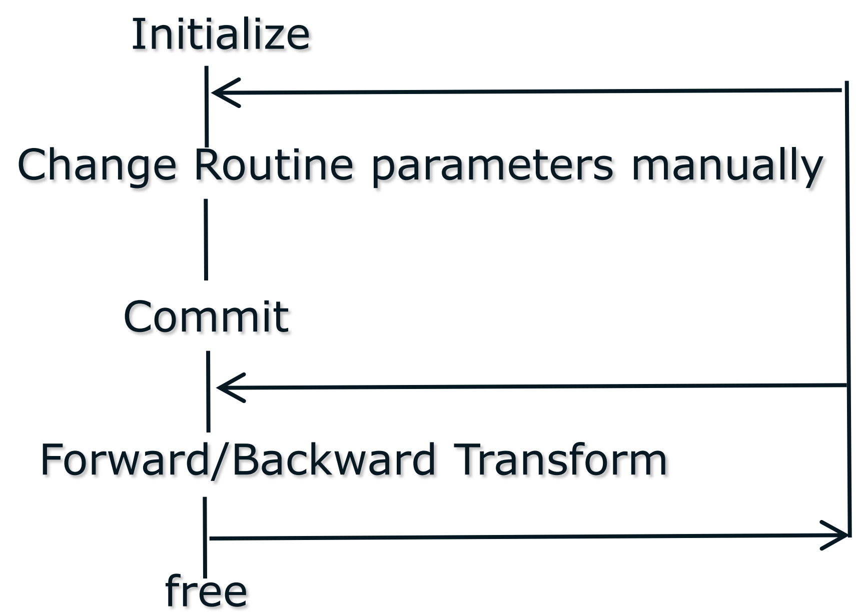
Intel MKL Cluster FFT

- FFT for distributed memory systems(clusters)
- Works with MPI using BLACS
- Open MP Support since Intel MKL 10.3
- 1, 2, 3 and multidimensional
- Requires basic MPI programming skills
- Same interface as the FFT (Fast Fourier Transforms)
- FFTW* support

Intel MKL Partial Differential Equations

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

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



Intel MKL Optimization Solvers

Optimization solver routines for

- Solving nonlinear least squares problems without constraints
- Solving nonlinear least squares problems with boundary constraints
- Computing the Jacobi matrix by central differences for solving nonlinear least squares problem

Based on Trust Region (TR) Methods

- **TR strength**: global and super linear convergence which differ them from the first order methods and unmodified Newton methods

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
- measure CPU frequency
- allocate and free memory allocated by Intel MKL memory management software

Agenda

- Intel® MKL system requirements, installation and environment
- Why Intel MKL?
- Overview of Intel MKL
- Intel MKL environment
- The Library Sections
- Linking with Intel MKL
- Threading in Intel MKL
- Lab

Linking with Intel MKL

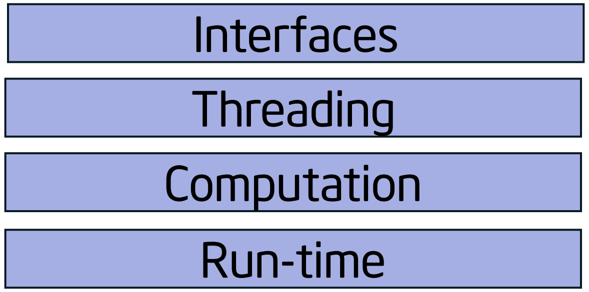
- Static Linking
- Dynamic linking
- Custom Dynamic Linking
- Single Dynamic Library (SDL): enables dynamic selection of interfaces and threading layer

Quick Comparison of Intel® MKL Linkage Models				
Feature	Dynamic Linkage	Static Linkage	Custom Dynamic Linkage	Single Dynamic Library
Processor Dispatches	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 Windows* - mkl_rt.lib 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

Linking with Intel MKL

Layered model approach for better control

- Interface Layer
  - LP64 / ILP64
- Threading Layer
  - Parallel based on Intel / alternate OpenMP implementation
  - Sequential
- Computational Layer
- Run-time Layer



Note: Users are strongly encouraged to link run-time layer library dynamically

Ex 1: Static linking using Intel® Fortran Compiler, 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, Intel®64 processor on Windows*

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

In this example, _dll.lib are the dispatcher libraries to the dynamic libraries (dll's).

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

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

All examples presume that correct paths to Intel MKL and libiomp5 libs are in place

If you are using Intel Compiler, use **–mkl[=lib]** flag on Linux*/Mac OS* and **/Qmkl[=lib]** on Windows* and it automatically picks up the Intel MKL libs, where lib can be parallel (default), sequential or cluster

Intel MKL Link Line Advisor tool

Intel® Math Kernel Library (MKL) Link Line Advisor

Reset

Select Intel® product:	Intel(R) MKL 10.3
Select OS:	Linux*
Select processor architecture:	Intel(R) 64
Select compiler:	Intel(R) C/C++
Select dynamic or static linking:	Dynamic
Select interface layer:	ILP64 (64-bit integer)
Select sequential or multi-threaded layer:	Multi-threaded
Select OpenMP library:	Intel(R) (libiomp5)
Select cluster library:	☑ CDFT (BLACS required) ☐ ScaLAPACK (BLACS required) ☑ BLACS
Select MPI library:	Intel(R) MPI
Select the Fortran 95 interfaces:	☐ BLAS95 ☐ LAPACK95
Link with Intel® MKL libraries explicitly:	☐

Use this link line:

```
-L$(MKLROOT)/lib/intel64 -lmkl_cdft_core -lmkl_intel_ilp64 -lmkl_intel_thread
-lmkl_core -lmkl_blacs_intelmpi_ilp64 -openmp -lpthread -lm
```

Compiler options:

```
-DMKL_ILP64  -I$(MKLROOT)/include
```

http://software.intel.com/en-us/articles/intel-mkl-link-line-advisor/

Agenda

- Intel® MKL system requirements, installation and environment
- Why Intel MKL?
- Overview of Intel MKL
- Intel MKL environment
- The Library Sections
- Linking with Intel MKL
- Threading in Intel MKL
- Lab

Optimization Notice

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
MKL_DYNAMIC – Intel MKL decides on the optimal number of threads to use

Call OpenMP or Intel MKL using

```
omp_set_num_threads()
mkl_set_num_threads()
mkl_domain_set_num_threads()
mkl_set_dynamic()
```

Example: Configure Intel MKL to run 4 threads for BLAS, sequentially in service functions and using default number of threads in all other parts of the library

- Environment variable
  MS Windows*: `set MKL_DOMAIN_NUM_THREADS="MKL_ALL=1, MKL_BLAS=4"`
  Use `export` on Linux*/Mac OS* to set the environment variables

- Function calls

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

In either case MKL BLAS runs on 4 threads, MKL service routines runs on 1 thread and rest of the libs like LAPACK, FFT, PARDISO, etc. run on default number of threads (see more in MKL documentation)

Agenda

- Intel® MKL system requirements, installation and environment
- Why Intel MKL?
- Overview of Intel MKL
- Intel MKL environment
- The Library Sections
- Linking with Intel MKL
- Threading in Intel MKL
- Lab

References

Intel® MKL product Information

– www.intel.com/software/products/mkl

Intel® MKL Knowledge Base

– http://software.intel.com/en-us/articles/intel-mkl-kb-home/

Intel® MKL User Discussion Forum

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

Technical Issues/Questions/Feedback

– http://premier.intel.com/

Intel® MKL Documentation

– http://software.intel.com/en-us/articles/intel-math-kernel-library-documentation/

Legal Disclaimer & Optimization Notice

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

intel®