Session:

**Intel® Performance Libraries:**

**Intel® Math Kernel Library (MKL)**
Agenda

- Intel MKL purpose
- Why is Intel MKL faster?
- Overview of MKL
- Intel MKL environment
- The Library Sections
- Linking with Intel MKL
- Threading in Intel MKL
Intel® Math Kernel Library Purpose

Performance, Performance, Performance!
Intel’s engineering, scientific, and financial math library
Addresses:

- Solvers (BLAS, LAPACK)
- Eigenvector/eigenvalue solvers (BLAS, LAPACK)
- Some quantum chemistry needs (dgemm)
- 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 & ISS)

Tuned for Intel® processors – current and future
Application Areas which could use MKL

**Energy** - Reservoir simulation, Seismics, 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 resource of 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.
BLAS Performance – multiple threads

- Performance (DGEMM function)
- Excellent scaling on multiprocessors
- Intel MKL performs far better than ATLAS* on multi-core

100% better than Atlas
200% better for larger matrices

Intel may make changes to specification, product descriptions, and plans at any time, without notice. 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 https://www.intel.com/software/products or call (U.S.) 1-800-628-8686 or 1-916-356-3104.
Intel® Math Kernel Library 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!

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

DFTs (General FFTs)
- Mixed radix, multi-dimensional transforms
- Multi threaded

Cluster DFT
- For SMP systems
**Intel® Math Kernel Library Contents**

**Sparse Solvers (PARDISO, DSS and and ISS)**
- For symmetric, structurally symmetric or non-symmetric, positive definite, indefinite or Hermitian sparse linear system of equations
- OOC version for huge problem sizes

**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

**PDEs (Partial Differential Equations)**
- Trigonometric transform and Poisson solvers.

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

**GMP**
- arbitrary precision arithmetic operations on integer numbers

**Support Functions**
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
## Intel® MKL Environment

<table>
<thead>
<tr>
<th>Domain</th>
<th>Fortran 77</th>
<th>Fortran 95/99</th>
<th>C/C++</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLAS</td>
<td>*</td>
<td>*</td>
<td>Via CBLAS</td>
</tr>
<tr>
<td>Sparse BLAS Level 1</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Sparse BLAS level 1&amp;2</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>LAPACK</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>ScaLAPACK</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PARDISO</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>DSS &amp; ISS</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>VML/VSL</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFT/Cluster FFT</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDEs</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimization (TR) Solvers</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

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

- **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

```c
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);
    }
}
```
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);
**Activity 1: Matrix multiplication**

Compare the performance of matrix multiply as implemented by C source code, DDOT, DGEMG and DGEMM.

Exercise control of the threading capabilities in MKL/BLAS.
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.

**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: $t = t_{\text{scalar}} + \frac{t_{\text{parallel}}}{p}$)
  - Extends blocking further into the code

No runtime library support required
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

Software and Services

*Other brands and names are the property of their respective owners.*
The BLACS routines implemented in Intel MKL are of four categories:

- Combines
- Point to Point Communication
- Broadcast
- Support.
Intel® MKL: Sparse Solvers

User callable Linear 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

**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

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

- Generating vectors of pseudorandom and quasi-random numbers
- Convolution & Correlation

Parallel computation support – some functions

User can supply own BRNG or transformations

### Performance Comparison of Random Number Generator

<table>
<thead>
<tr>
<th></th>
<th>Running Time (s)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard rand()</td>
<td>40.52</td>
<td>1</td>
</tr>
<tr>
<td>Intel MKL VSL RNG</td>
<td>6.88</td>
<td>5.89</td>
</tr>
<tr>
<td>MKL + OpenMP* version (8threads)</td>
<td>0.92</td>
<td>44.04</td>
</tr>
</tbody>
</table>

### Basic RNGs

<table>
<thead>
<tr>
<th>Pseudo RNGs</th>
<th>Quasi RNGs</th>
</tr>
</thead>
<tbody>
<tr>
<td>G31, GFSR250, MRG32, MCG59, WH, MT19937, MT2203</td>
<td>Sobol-quasi, Niederreiter quasi</td>
</tr>
</tbody>
</table>

### Distribution Generators

<table>
<thead>
<tr>
<th>Continuous</th>
<th>Discrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform, Gaussian (two methods), Exponential, Laplace, Weibull, Cauchy, Rayleigh, Lognormal, Gumbel, Gamma, Beta</td>
<td>Uniform, UniformBits, Bernoulli, Geometric, Binomial, Hypergeometric, Poisson, PoissonV, NegBinomial</td>
</tr>
</tbody>
</table>

**Configuration Info:**

- Quad-Core Intel® Xeon® processor 5300 series
- 2.4 GHz, 2x8MB L2 cache, 4 GB memory
- Windows Server® 2003 Enterprise x64 Edition
- Test run on a vector of 1000 elements
- Intel MKL 10.0 and Intel® C++ Compiler 10.1

**Excellent Multi-core Scaling**
Using VSL

Basically a 3-step Process
1. Create a stream pointer.
   ```c
   VSLStreamStatePtr stream;
   ```
2. Create a stream.
   ```c
   vslNewStream(&stream,VSL_BRNG_MC_G31,seed);
   ```
3. Generate a set of RNGs.
   ```c
   vslRngUniform(0,&stream,size,out,start,end);
   ```
4. Delete a stream (optional).
   ```c
   vslDeleteStream(&stream);
   ```
Activity 2: Calculating Pi using a Monte Carlo method

Compare the performance of C source code (RAND function) and VSL. Exercise control of the threading capabilities in MKL/VSL.
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
Using the Intel® MKL DFTs

Basically a 3-step Process

1. Create a descriptor.
   
   \[
   \text{Status} = \text{DftiCreateDescriptor}(\text{MDH}, \ldots)
   \]

2. Commit the descriptor (instantiates it).
   
   \[
   \text{Status} = \text{DftiCommitDescriptor}(\text{MDH})
   \]

3. Perform the transform.
   
   \[
   \text{Status} = \text{DftiComputeForward}(\text{MDH}, X)
   \]

Optionally free the descriptor.

MDH: MyDescriptorHandle
Intel® MKL: Cluster FFT

- FFT for SMP systems/clusters
- Works with MPI using BLACS
- 1, 2, 3 and multidimensional
- Require basic MPI programming skills
- Same interface as the DFT from standard MKL

Intel may make changes to specification, product descriptions, and plans at any time, without notice. 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 or call (U.S.) 1-800-628-8686 or 1-916-356-3104.
Intel® MKL: Partial Differential Equations

- **Poisson Library**
  - for fast solving of simple Helmholtz, Poisson, and Laplace problems

- **Trigonometric Transform interface routines**

  Initialize
  ```
  _init_trig_transform
  _commit_trig_transform
  _forward_trig_transform
  _backward_trig_transform
  free_trig_transform
  ```

  Commit
  ```
  _forward_trig_transform
  _backward_trig_transform
  ```

  Forward/Backward Transform
  ```
  free
  ```
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 differentiates them from the first order methods and unmodified Newton methods.
Intel® MKL: GMP

- Arbitrary precision arithmetic routines on integers
- Interface fully match with Gnu Multi Precision lib
- If your application uses GMP functions, link with the MKL and libm libraries.
  - For example on IA32: 
    $CC$ `prog.c` -L$MKL_LIB_PATH -lmkl_intel -lmkl_core -liomp5 -lpthread -lm

- Optimized for Intel processors
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
## Quick Comparison of Intel MKL Linkage Models

<table>
<thead>
<tr>
<th>Feature</th>
<th>Dynamic Linkage</th>
<th>Static Linkage</th>
<th>Custom Dynamic Linkage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor Updates</td>
<td>Automatic</td>
<td>Automatic</td>
<td>Recompile and redistribute</td>
</tr>
<tr>
<td>Optimization</td>
<td>All Processors</td>
<td>All Processors</td>
<td>All Processors</td>
</tr>
<tr>
<td>Build</td>
<td>Link to import libraries</td>
<td>Link to static libraries</td>
<td>Build separate import libraries, which are created automatically</td>
</tr>
<tr>
<td>Calling</td>
<td>Regular Names</td>
<td>Regular Names</td>
<td>Modified Names</td>
</tr>
<tr>
<td>Total Binary Size</td>
<td>Large</td>
<td>Small</td>
<td>Small</td>
</tr>
<tr>
<td>Executable Size</td>
<td>Smallest</td>
<td>Small</td>
<td>Smallest</td>
</tr>
<tr>
<td>Multi-threaded/thread safe</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Linking with Intel® MKL contd..

Layered model approach for better control

- Interface Layer
  - Compiler: Intel / GNU
  - LP64 / ILP64
- Threading Layer
  - Threaded / alternate OpenMP
  - Sequential
- Computational Layer
- Run-time Layer

Choose the libs from each layer for linking.

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 mkl_intel_lp64_dll.lib mkl_intel_thread_dll.lib mkl_core_dll.lib libiomp5md.dll
```

Note: Strongly recommended to link Run-time layer library dynamically
Intel® MKL Threading

There are numerous opportunities for threading:
- Level 3 BLAS (O(n^3))
- LAPACK* (O(n^3))
- FFTs (O(n log(n))
- VML, VSL ? depends on processor and function

Not threaded for some routines due to:
- Limited resource is memory bandwidth.
- Threading level 1 and level 2 BLAS are mostly ineffective (O(n))

- Threaded using OpenMP*
  - With support for GCC* and Microsoft* OpenMP*

- ScalAPACK and Cluster FFT are SMP Parallel
- All Intel® MKL is thread-safe
Thread Control in Intel® MKL

Set OpenMP or Intel MKL environment variable:

\[
\begin{align*}
OMP\_NUM\_THREADS \\
MKL\_NUM\_THREADS \\
MKL\_DOMAIN\_NUM\_THREADS
\end{align*}
\]

Call OpenMP or Intel MKL using

\[
\begin{align*}
\text{omp\_set\_num\_threads()} \\
\text{mkl\_set\_num\_threads()} \\
\text{mkl\_domain\_set\_num\_threads()} \\
\text{MKL\_DYNAMIC/mkl\_set\_dynamic(): Intel® MKL decides the number of threads.}
\end{align*}
\]

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);
  ```

  

Copyright © 2014, Intel Corporation. All rights reserved.
Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States or other countries. * Other brands and names are the property of their respective owners.
Performance Libraries: Intel® MKL
What’s Been Covered

- Intel® Math Kernel Library is a broad scientific/engineering math library.
- It is optimized for Intel® processors.
- It is threaded for effective use on multi-core and SMP machines.
References

Intel® MKL product Information
- [www.intel.com/software/products/mkl](http://www.intel.com/software/products/mkl)

Technical Issues/Questions/Feedback
- [http://premier.intel.com/](http://premier.intel.com/)

Self-help
  (Click “Support Resources” tab)

User Discussion Forum

What are the new software tools?
- [http://whatif.intel.com](http://whatif.intel.com)