Intel® Math Kernel Library. Intel® Integrated Performance Primitives, the latest Updates, upcoming features

May 2014

Gennady Fedorov

Outline

- **Intel® Cluster Studio XE**
- Intel® MKL : Overview
- MKL 11.2 beta - new Features
- Intel® IPP, Overview
- IPP <-> MKL
- IPP, MKL : Beyond current versions
- References

Intel® Cluster Studio XE 2013 SP1

Tools to Scale Forward, Scale Faster – for HPC Clusters

Phase	Product	Feature	Benefit
Build	**Intel® MPI Library**	High Performance Message Passing (MPI) Library	• Enabling High Performance Scalability, Interconnect Independence, Runtime Fabric Selection, and Application Tuning Capability
	Intel® Composer XE SP1	C/C++ and Fortran compilers and performance libraries • Intel® Threading Building Blocks • Intel® Cilk™ Plus • Intel® Integrated Performance Primitives • Intel® Math Kernel Library	• Enabling solution to achieve the application performance and scalability benefits of multicore and forward scale to many-core
Verify	**Intel® Inspector XE**	Memory & threading dynamic analysis for code quality Static Security Analysis for code quality	• Increased productivity, code quality, and lowers cost, finds memory, threading , and security defects before they happen • Now MPI enabled at every cluster node
Verify & Tune	**Intel® Trace Analyzer & Collector**	MPI Performance Profiler for understanding application correctness & behavior	• Analyze performance of MPI programs and visualize parallel application behavior and communications patterns to identify hotspots
Tune	**Intel® VTune™ Amplifier XE**	Performance Profiler for optimizing application performance and scalability	• Remove guesswork, saves time, makes it easier to find performance and scalability bottlenecks • Now MPI enabled at every cluster node

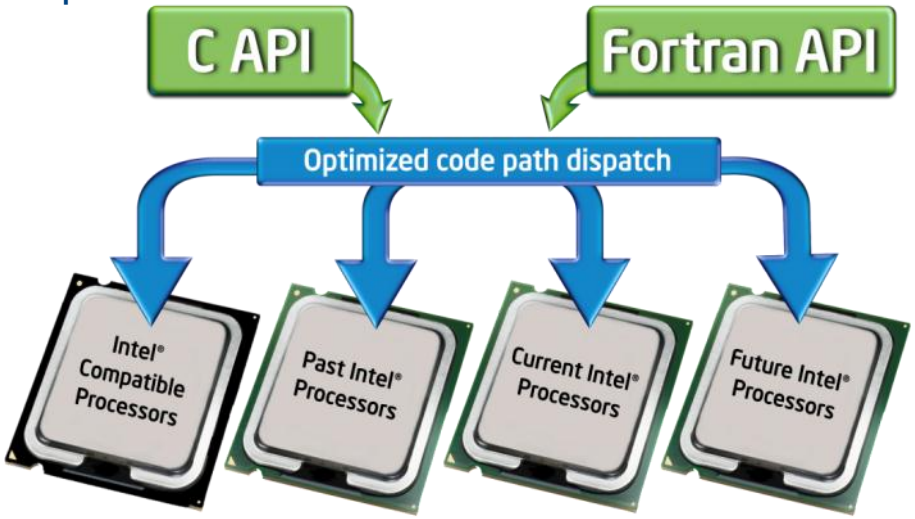
Outline

- Intel® Cluster Studio XE
- **Intel® MKL : Overview**
- MKL 11.2 beta – new Features
- Intel® IPP, Overview
- MKL : Beyond current versions
- References

Intel® MKL Overview

Intel® Math Kernel Library

- Highly optimized threaded math routines
- Applications in science, engineering, finance
- Use Intel® MKL on Windows*, Linux*, Mac OS*
- Use Intel® MKL with Intel compiler, gcc, MSFT*, PGI
- Component of Intel® Parallel Studio XE and Intel® Composer XE



HPC wire

View online

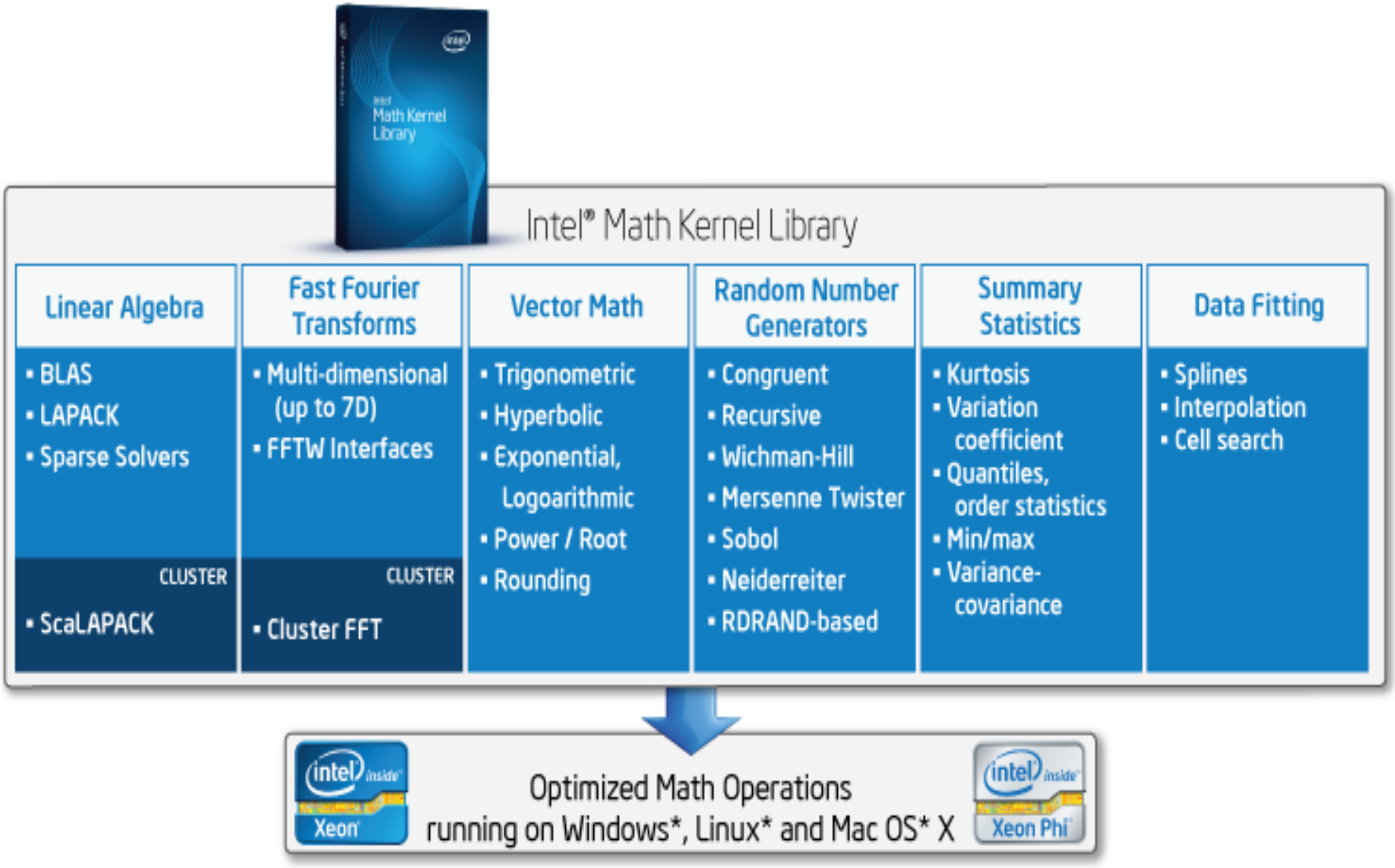
Editors' Choice Awards 2011
Best HPC Performance Product or Technoogy
Intel
Parallel Studio XE 2011

Evans Data Corporation EDC

EDC North America Development Survey 2011, Volume II

33% of math libraries users rely on Intel's Math Kernel Library

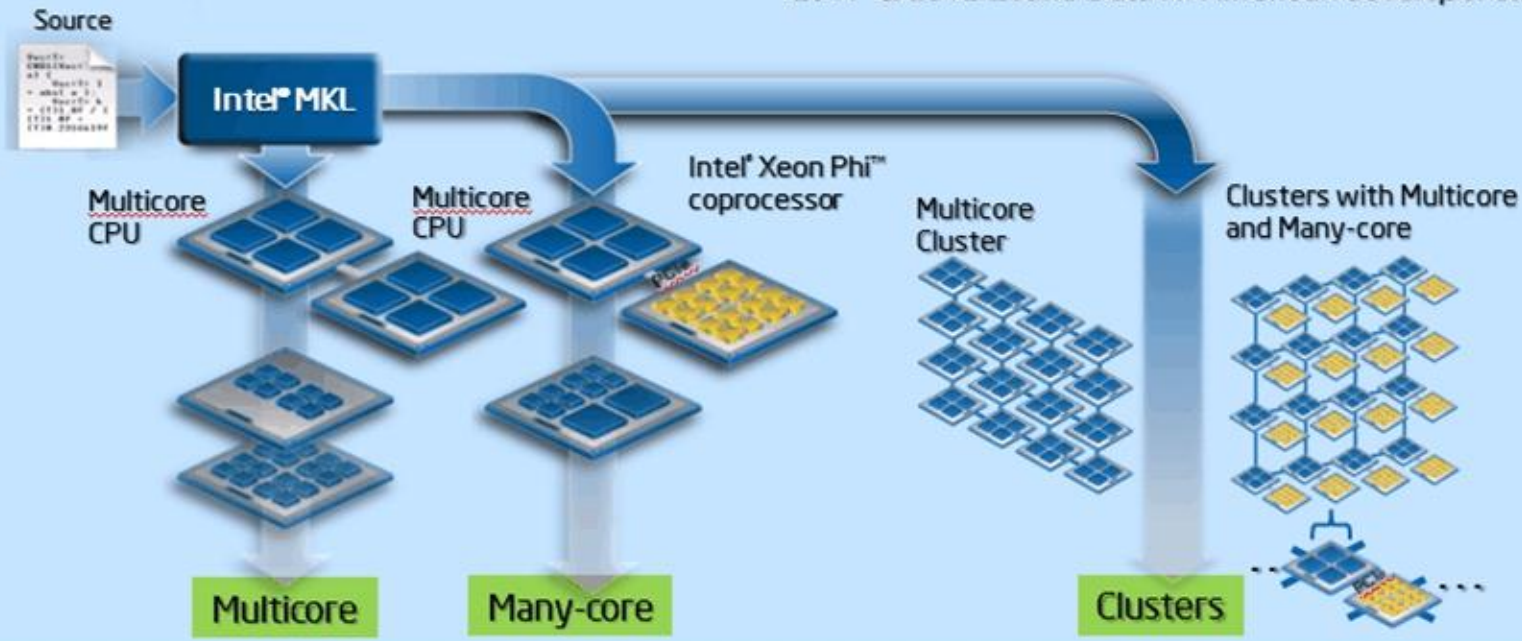
Intel® MKL Domains & Functions



Intel® MKL is industry's leading math library *

Linear Algebra	Fast Fourier Transforms	Vector Math	Vector Random Number Generators	Summary Statistics	Data Fitting
•BLAS •LAPACK •Sparse solvers •ScaLAPACK	•Multidimensional (up to 7D) •FFTW interfaces •Cluster FFT	•Trigonometric •Hyperbolic •Exponential, Logarithmic •Power / Root •Rounding	•Congruential •Recursive •Wichmann-Hill •Mersenne Twister •Sobol •Neiderreiter •Non-deterministic	•Kurtosis •Variation coefficient •Quantiles, order statistics •Min/max •Variance-covariance •…	•Splines •Interpolation •Cell search

* 2011 & 2012 Evans Data N. American developer surveys



Usage Models on Intel® Xeon Phi™ Coprocessors

- Automatic Offload

- Compiler Assisted Offload

- Native Execution

Usage models Continued ..

- **Automatic Offload**:

  - No code changes required

  - Automatically uses both host and target

  - Transparent data transfer and execution management

  Ex: **Export MKL_MIC_ENABLE=1**

Article for the List of AO Enabled Functions:
http://software.intel.com/en-us/articles/intel-mkl-automatic-offload-enabled-functions-for-intel-xeon-phi-coprocessors

How to Use Automatic Offload

- Using Automatic Offload is easy

```
Call a function:

mkl_mic_enable()
```

or

```
Set an env variable:

MKL_MIC_ENABLE=1
```

- What if there doesn't exist a MIC card in the system?
  - Runs on the host as usual **without any penalty**!
- Reporting of Automatic Offload profiling:
  - int mkl_mic_set_offload_report (int *enabled*);
    - **off** by the Default
    - **on** if "enabled" == 1 or 2
  - Env Variable: OFFLOAD_REPORT

Work Division Control in Automatic Offload

- Using support functions

Examples	Notes
MKL_MIC_Set_Workdivision(MKL_TARGET_MIC, 0, 0.5)	Offload 50% of computation only to the 1st card.

- Using environment variables

Examples	Notes
MKL_MIC0_WORKDIVISION=50	Offload 50% of computation only to the 1st card.

- The support functions take precedence over environment variables

Tips for Using Automatic Offload

- AO works only when matrix sizes are right
  - **?**GEMM: Offloading only when M, N > 2048, K > 256
  - **?**TRSM/TRMM: Offloading only when M, N > 3072
  - **?**GETRF: M,N > 8192
  - **?**POTRF:N>=6144
  - **?**GEQRF:M=N>=8192
  - *Square matrices may give better performance*
- Work division settings are just hints to MKL runtime
- How to disable AO after it is enabled?
  - *mkl_mic_set_workdivision(MIC_TARGET_HOST, 0, **1.0**)*
  - MKL_HOST_WORKDIVISION=100

Tips for Using Automatic Offload

- Threading control tips:
  - Avoid using the OS core of the target, which is handling data transfer and housekeeping tasks. Example (on a 60-core card):

```
MIC_KMP_AFFINITY=explicit,granularity=fine,proclist=[1-236:1]
```

  - Also, prevent thread migration on the host:

```
KMP_AFFINITY=granularity=fine,compact,1,0
```

  - Note: Different env-variables on host and MIC:

```
OMP_NUM_THREADS          MIC_OMP_NUM_THREADS
KMP_AFFINITY             MIC_KMP_AFFINITY
```

Intel MKL : Conditional Numerical Reproducibility (CNR) - motivation

Ever seen something like this?

```
C:\Users\me>test.exe
4.012345678901111

C:\Users\me>test.exe
4.012345678902222

C:\Users\me>test.exe
4.012345678902222

C:\Users\me>test.exe
4.012345678901111

C:\Users\me>test.exe
4.012345678902222
```

...or this?

```
C:\Users\me>test.exe
4.012345678901111

C:\Users\me>test.exe
4.012345678901111

C:\Users\me>test.exe
4.012345678901111

C:\Users\me>test.exe
4.012345678901111

C:\Users\me>test.exe
4.012345678901111
```

Intel® Xeon® Processor E5540

```
C:\Users\me>test.exe
4.012345678902222

C:\Users\me>test.exe
4.012345678902222

C:\Users\me>test.exe
4.012345678902222

C:\Users\me>test.exe
4.012345678902222

C:\Users\me>test.exe
4.012345678902222
```

Intel® Xeon® Processor E3-1275

Intel MKL : CNR – Why do results vary?

Root cause for variations in results

- floating-point numbers ➔ order of computation matters!
- double precision arithmetic example $(a+b)+c \neq a+(b+c)$

$2^{-63} + 1 + -1 = 2^{-63}$ (infinitely precise result)

$(2^{-63} + 1) + -1 = 0$ (correct IEEE single precision result)

$2^{-63} + (1 + -1) = 2^{-63}$ (correct IEEE single precision result)

- **Why does the order of operations change in Intel MKL?**
- Memory alignment affects grouping of data in registers
- Number of threads most functions are threaded to use all available cores
- Dynamic task scheduling some algorithms use asynchronous task

Order matters when doing floating point arithmetic.
Many optimizations require a change in order of operations.

Conditional Numerical Reproducibility

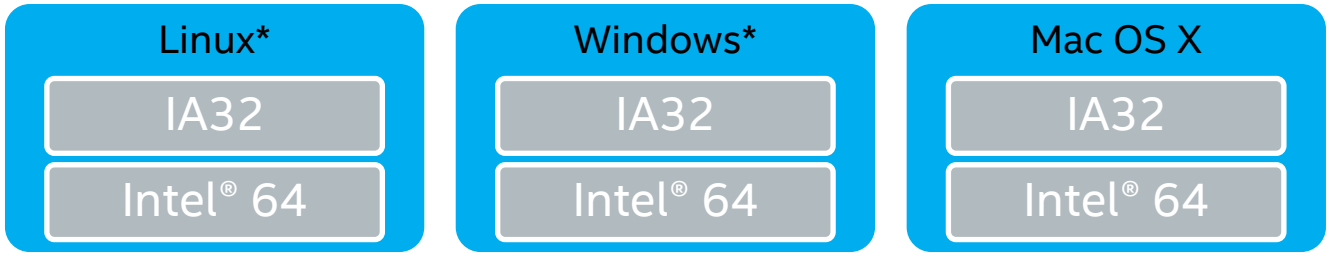
Challenges *Means of control*

Challenges	Means of control
Memory alignment	• Align memory — try Intel MKL memory allocation functions • 64-bytes alignment for processors in the next few years
Number of threads	• Set the number of threads to a constant number • Use sequential libraries
New! Dynamic task scheduling	• New control to specify static scheduling • *Previous sequential operation was required*
New! Code branches	• New controls to maintain consistent code branches across different processors (`mkl_cbwr_set(`**MKL_CBWR_AVX**`)`)

Intel MKL 11.0 introduces functions to broaden the cases where reproducibility is possible.

Why “Conditional”?

- Reproducibility is not guaranteed across…
  - operating systems
  - architectures



- Reproducibility across non-Intel processors is limited
- Reproducibility will not cover all future Intel processors
- Reproducibility will not be maintained from one version of Intel MKL to the next
- Reproducibility controls in Intel MKL only affect Intel MKL functions; other tools, and other parts of your code can still cause variation

New functionality expands the scope for reproducibility, but not to every case.

Extended Eigensolver Routines(FEAST)

- Solution of standard generalized EV problems (Feast 2.0)
  - standard, $\boldsymbol{Ax = \lambda x}$, *where are*
    - *for A* complex Hermitian and *A* real symmetric
  - generalized, $\boldsymbol{Ax = \lambda Bx}$
    - *where A* complex Hermitian, *B* Hermitian positive definite (hpd)
    - *A* real symmetric and *B* real symmetric positive definite (spd)
- Real/Complex and Single/Double precisions
- Compatible with FORTRAN 77, Fortran 90, and C
- matrix storage: Full, Band storages, Sparse format (SCR)
- Reverse communication interfaces (RCI)
- Predefined driver interfaces

MKL 11.1 Highlights

- Intel® Xeon Phi™ support from Windows* host
- Better installation experience
  - A choice of components to install
  - Examples and tests packaged as archives
- HPL support for heterogeneous clusters
- Performance improvements
- CNR support for unaligned input data
- CNR support for arbitrary #threads - delayed

Outline

- Intel® Cluster Studio XE
- Intel® MKL : Overview
- **MKL 11.2 beta – new Features**
- Intel® IPP, Overview
- MKL : Beyond current versions
- References

MKL 11.2 beta - new Features

- Verbose mode
- Small Size Problem
- Cluster Pardiso
- CookBook
- Android
- Deprecations

Verbose mode

- Motivation:
  MKL users : "It would be nice if MKL reports called functions names with parameters, that we won't spend additional time and resource to figure this out"
- BLAS, LAPACK, FFT
- The same standard MKL interface
- To enable verbose use:
  - environment variable MKL_VERBOSE=1 or
  - function mkl_verbose(1)
  - to disable - mkl_verbose(0)
- Disabled by the default

Verbose Mode Output

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

thrID: calling thread ID

func: called function name with input parameters

time: time (sec) spend in function

freq: current CPU frequency

Nthr: Number of MKL threads

dyn: {1|0} – MKL_DYNAMIC

CPU: SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, AVX, AVX2 for Intel & *AMD for non Intel*

CNR:OFF, AUTO, COMPATIBLE, SSE2, SSE3, SSSE3, SSE4_1, SSE4_2, AVX, AVX2

AO: {1|0},

iface: ilp64, lp64, cdecl, stdcall

plat: lnx, osx, win, mic,

mem: {1|0} - MKL_FAST_MM on/off

mstat: MB used by function

Small Size Problem

- Significant overhead (>100%) for small matrices
  - Function call overhead
  - Error checking
  - Architecture specific dispatching
  - Matrix size specific dispatching
  - Buffer allocation

- Unrolled C/Fortran codes may outperform MKL for small matrix multiplication

Small Size Problem

To activate the feature:

- Compile your C or Fortran code with the macro and options:

Intel MKL Mode	Macro	Compiler Option
Threaded	MKL_INLINE	-DMKL_INLINE
Sequential	MKL_INLINE_SEQ	-DMKL_INLINE_SEQ

- Fortran application:
  - -fpp compiler option
  - Include files: mkl_inline_pp.fiand mkl_inline.fi
- Link statically with MKL
- See examples into mkl_user's guide

Small Size Problem

Limitations of the Feature:

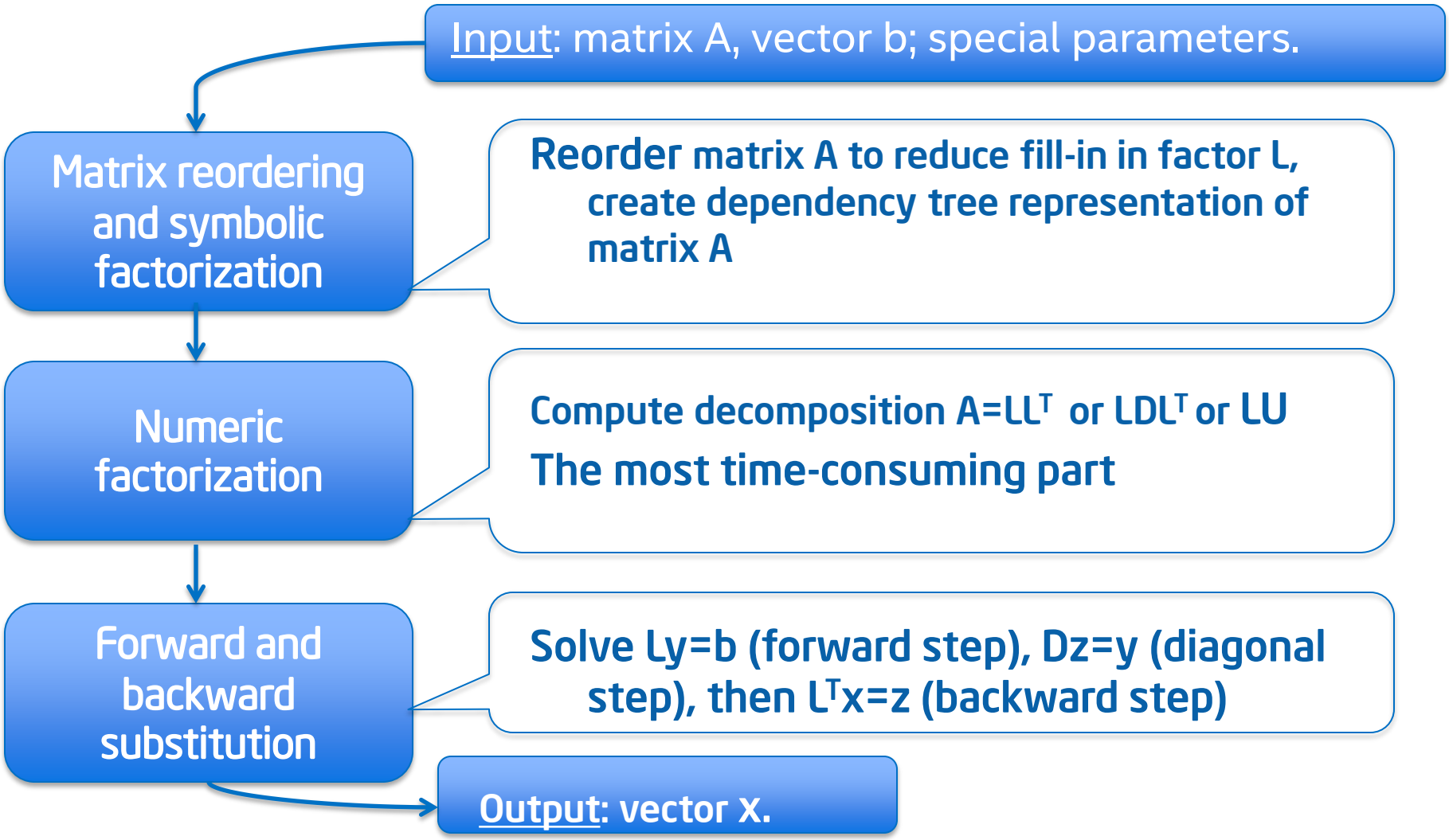
- if the MKL_INLINE or MKL_INLINE_SEQ macro is used, MKL may skip error checking

IMPORTANT

With a limited error checking, you are responsible for checking the correctness of function parameters to avoid unsafe and incorrect code execution.

- Available for the Intel® 64 architecture only
- Only Intel® Compiler 14.0 or higher is supported
- You must link with Intel MKL statically
- Available for Intel® Xeon® processors and *gemm functions
- Verbose mode, CNR and CBLAS interfaces are not supported

Cluster Pardiso



Cluster Pardiso. Interface/migration

Supported since Intel® MKL 11.2 beta for :

Linux* & Microsoft Windows*, 64-bit only

C:

```
{
….

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

Fortran:

```
….

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

```
…
INTEGER*4 comm
call CPARDISO(pt, maxfct, mnum, mtype, phase,
    n, a, ia, ja, idum, nrhs,
    iparm, msglvl, b, x, &error , comm)
…
```

Cluster Pardiso. Features

- Zero - one-based switch of Cluster Pardiso
- Support of distributed input data
- Correct treatment of internal errors
- MPI calls via MKL BLACS
- Single precision version both real & complex case
- LP64/ILP64 interfaces
- Matrix checker of CSR format
- Iterative refinement
- Reordering step is paralleled only by OpenMP
- Work with matrices with empty rows

Documentation - Cookbook

- **Motivation**: Requests from customers to show the “example from the life”
- **Steps**: Goal / Solution / Code Examples / Discussion / Resources

- **MKL v. 11.2 beta – 5 examples:**

1) *Finding an approximate solution to a nonlinear equation demonstrates a method of finding a solution to a nonlinear equation using Intel MKL PARDISO, BLAS, and Sparse BLAS routines*

2) *Factoring a block tridiagonal matrix uses Intel MKL implementations of BLAS and LAPACK routines*

•3,4) *Using Fast Fourier Transforms for computer tomography image reconstruction uses the Fast Fourier Transforms to reconstruct an image from computer tomography data*

5) *Noise filtering in financial market data streams uses Intel MKL summary statistics routines for computing a correlation matrix for streaming data*

Note:

- +12 recipes would be added in the 11.2 Gold version
- Code examples in the cookbook are provided in Fortran for some recipes and in C for other recipes.

Cookbook - Recipe

Using LAPACK symmetric eigensolvers for Hermitian tridiagonal matrices

Goal

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

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

Solution

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

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

Calculating eigenvalues and eigenvectors for a Hermitian tridiagonal matrix using DSTEV

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

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

       STOP
       END
```

Discussion

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

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

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

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

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

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

MKL for Android

- Android 32 bit
- Static linking
- Sequential version (1 thread)
- No message catalog
- Available since 11.1. **update3** & 11.2 **beta Update 1**
- 64 bit will be available later – TBD
- Dynamic Linking (Custom dll) – not available - TBD

Outline

- Intel® Cluster Studio XE
- Intel® MKL : Overview
- MKL 11.2 beta - new Features
- **Intel® IPP, Overview**
- MKL : Beyond current versions
- References

Intel® IPP Functions and Code Samples: Image Processing/Coding

Domain	Functions	Samples
Image Processing	* Geometry transformations, such as resize/rotate * Linear and non-linear filtering operation on an image for edge detection, blurring, noise removal and etc for filter effect. * Linear transforms for 2D FFTs, DFTs, DCT. * image statistics and analysis	* Tiled Image Processing / 2D Wavelet Transform /C++ Image Processing Classes/Image Processing functions Demo
Computer Vision	* Background differencing, Feature Detection (Corner Detection, Canny Edge detection), Distance Transforms, Image Gradients, Flood fill, Motion analysis and Object Tracking, Pyramids, Pattern recognition, Camera Calibration	* Face Detection
Color Models	* Convert image/video color space formats: RGB, HSV, YUV, YCbCr * Up/Down sampling * Brightness and contrast adjustments	
JPEG Coding	* High-level JPEG and JPEG2000 compression and decompression functions * JPEG/JPEG2000 support functions: DCT, Wavelet transforms, color conversion, downsampling	* Integration with the Intel® JPEG Library (IJL) / Integration with the Independent JPEG Group (IJG) library /JPEG2000 encoder/decoder /JPEG viewer
Realistic Rendering	* Acceleration Structures, Ray-Scene Intersection and Ray Tracing * Surface properties, shader support, tone mapping	* Ray Tracing

Intel® IPP Functions and Code Samples: Video/Audio/Speech

Domain	Functions	Samples
Video Coding	* VC-1, H.264, MPEG-2, MPEG-4, H.261, H.263 and DV codec support functions	* Simple Media Player/ Video Encoder / h.264/DV decoding/video transcoder /Reverberation Demo/ Virtual Multi-Channel Audio Player/
Audio Coding	* Echo cancellation and audio transcoding, BlockFiltering, Spectral Data prequantization.	* Audio Codec Console application
Speech Coding	* Adaptive/Fixed Codebook functions, Autocorrelation, Convolution, Levinson-Durbin recursion, Linear Prediction Analysis & Quantization, Echo Cancellation, Companding	* G.168, G.167, G.711, G.722, G.722.1, G.722.2, AMRWB, Extended AMRWB (AMRWB+), G.723.1, G.726, G.728, G.729, RT-Audio, GSM AMR, GSM FR
Speech Recognition	* Feature Processing, Model Evaluation/Estimation/Adaptation, Vector Quantization, Polyphase Resampling, Advanced Aurora, Ephraim-Malah Noise Supression, AEC, Voice Detection	* Aurora, Advanced Aurora, Audio Processing, Gaussian Mixture, Speech Processing
Signal Processing	* Transforms: DCT, DFT, MDCT, Wavelet (both Haar and user-defined filter banks), Hilbert * Convolution, Cross-Correlation, Auto-Correlation, Conjugate * Filtering: IIR/FIR/Median filtering, Single/Multi-Rate FIR LMS filters * Other: Windowing, Jaehne/Tone/Triangle signal generation, Thresholding	* Signal Processing Function Demo

Intel® IPP Functions and Code Samples: Data Processing/Compression

Domain	Functions	Samples
Data Compression	* Entropy-coding compression: Huffman, VLC * Dictionary-based compression: LZSS, LZ77 * Burrows-Wheeler Transform (BWT), MoveToFront (MTF), Run-Length-Encoding (RLE), Generalized Interval Transformation (GIT) * Compatible feature support for zlib and bzip2	* zlib, bzip2, gzip-compatible /General data compression examples
Cryptography	* Big-Number Arithmetic / Rijndael, DES, TDES, SHA1, MD5, RSA, DSA, Montgomery, prime number generation and pseudo-random number generation (PRNG) functions	* Intel IPP crypto usage in Open SSL*
String Processing	* Compare, Insert, change case, Trim, Find, Regexp, Hash	* "ippgrep" – regular expression matching
Vector Math	* Logical, Shift, Conversion,Power, Root, Exponential, Logarithmic, Trigonometric, Hyperbolic, Erf, Erfc	
Matrix Math	* Addition, Multiplication, Decomposition, Eigenvalues, Cross-product, transposition	
Common Functions	* CPUTypes, Thread number control, Memory Allocation	* Linkages/Different language support

Outline

- Intel® Cluster Studio XE
- Intel® MKL : Overview
- MKL 11.2 beta - new Features
- Intel® IPP, Overview
- **MKL : Beyond current versions**
- References

MKL : Beyond 11.2 beta

- AO - SVD
- AO - Eigensolvers
- AO – threshold mode for debugging
- AO - *int mkl_mic_set_resource_limit(double fraction);*
- *SpMV – Ellpack format*
- *HPCG benchmarks*

MKL -- Deprecations since 11.2 beta

- The libraries are no longer provided:
  - IA32: mkl_p4p andmkl_vml_p4p
  - Intel® 64 : mkl_p4n and mkl_vml_p4n libraries
- Support for the following OS is Deprecated and will be removed
  - Windows XP*
  - Windows Server 2003*
  - Windows Vista*
  - Red Hat* Enterprise Linux* 5
  - SUSE LINUX Enterprise Server* 10
- Old MKL domain names used for threading settings will be deleted
  - MKL_ALL, MKL_BLAS, MKL_FFT, MKL_VML
  - MKL_DOMAIN_ALL, MKL_DOMAIN_BLAS, MKL_DOMAIN_FFT, MKL_DOMAIN_VML, MKL_DOMAIN_PARDISO

IPP : beyond version 8.1

- IPP on Xeon Phi. Native, Assistant Offload
- Android Support
- Threading Strategy
- In-Place operations
- Deprecation Policy :

https://software.intel.com/sites/products/ipp-deprecated-features-feedback/

- UIC, UMC codec -- moving to legacy archive
- Audio and Video Codecs -- Media SDK
- IPP Preview – GPU support (CV and IPPI)

References

Intel® MKL page

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

Intel MKL Forum:

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

Intel IPP Forum:

- https://software.intel.com/en-us/forums/intel-integrated-performance-primitives/

Intel Experimental Software

- http://software.intel.com/en-us/whatif

Legal Disclaimer & Optimization Notice

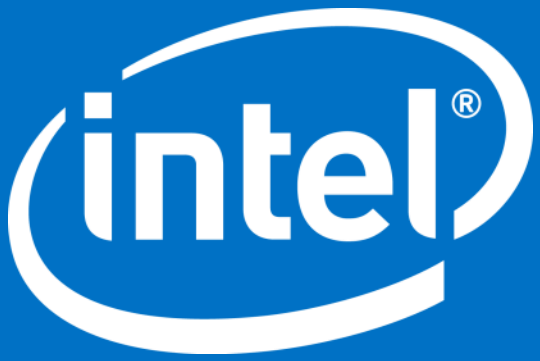
INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING 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


intel®