# Maximizing performance and scalability using Intel performance libraries

Roger Philp

Intel HPC Software Workshop Series 2016

HPC Code Modernization for Intel® Xeon and Xeon Phi™

February 17th 2016, Barcelona

# Intel® Parallel Studio XE 2016 components

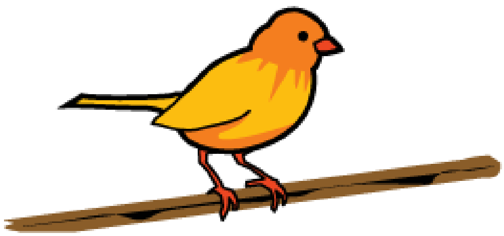
| Component | Full Licensing (including Intel® Premier Support) | | | Free Licensing | | | |
|---|---|---|---|---|---|---|---|
| | Composer Edition | Professional Edition | Cluster Edition | Student/ Educator | Open Source Contributor | Academic Researcher | Community (Everyone!) |
| Intel® C/C++ Compiler (including Intel® Cilk™ Plus) | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| Intel® Fortran Compiler | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| OpenMP 4.0 | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| Intel® Threading Building Blocks (C++ only) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Intel® IPP Library (C/C++ only) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Intel® Math Kernel Library | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Intel® Data Analytics Acceleration Library | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Intel® MPI Library | | | ✓ | ✓ | | ✓ | |
| Rogue Wave IMSL Library (Fortran only) | Bundled and Add-on | Add-on | Add-on | | | | |
| Intel® Advisor XE | | ✓ | ✓ | ✓ | ✓ | | |
| Intel® Inspector XE | | ✓ | ✓ | ✓ | ✓ | | |
| Intel® VTune™ Amplifier XE | | ✓ | ✓ | ✓ | ✓ | | |
| Intel® ITAC + MPI Performance Snapshot | | | ✓ | ✓ | | | |

BAYNCORE

# Intel® Parallel Studio XE: performance libraries

| | |
|---|---|
| Intel Threading Building Blocks (Intel® TBB) | • C++ template library for task parallelism<br>• Rich set of components for scalable parallel applications |
| Intel® Integrated Performance Primitives (Intel® IPP) | • Collection of high performance routines<br>• Broad range of functionality on different domains |
| Intel® Math Kernel Library (Intel® MKL) | • Highly optimized C/Fortran computing math library<br>• Sequential/parallel/cluster implementations |
| Rogue Wave IMSL Fortran Numerical Library | • Mathematical and statistical library for HPC<br>• Available as a bundled or add-on package to Intel PSXE |
| Intel® Data Analytics Acceleration Library (Intel® DAAL) | • Optimized building blocks library for data analytics<br>• New in Intel Parallel Studio XE 2016 |

BAYNCORE



# Intel® Threading Building Blocks Library

Intel® Parallel Studio XE Suite

Intel® System Studio Suite

# Intel® Threading Building Blocks (TBB)

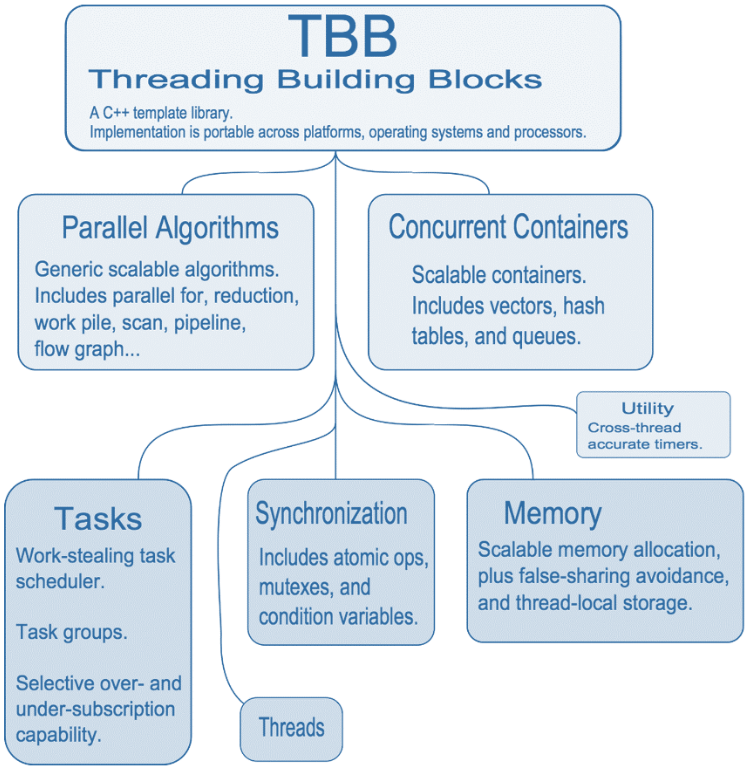
C++ template library for task parallelism

- Open specification, portable across platforms, OSs and processors
- Intel and open source versions available

Philosophy behind Intel® TBB

- Rich feature set for task based parallelism
  - Known parallel patterns easily mapped
- Logical tasks are transparently mapped to threads
  - Full support for nested parallelism
- Work-stealing scheduler to favour load balancing

Check release notes for news on latest 4.4 and earlier versions



# tbb example

**Standard C++ example**

```
void SerialApplyFoo( float a[], size_t n ) {
    for( size_t i=0; i!=n; ++i )
        Foo(a[i]);
}
```

**tbb version – with work stealing**

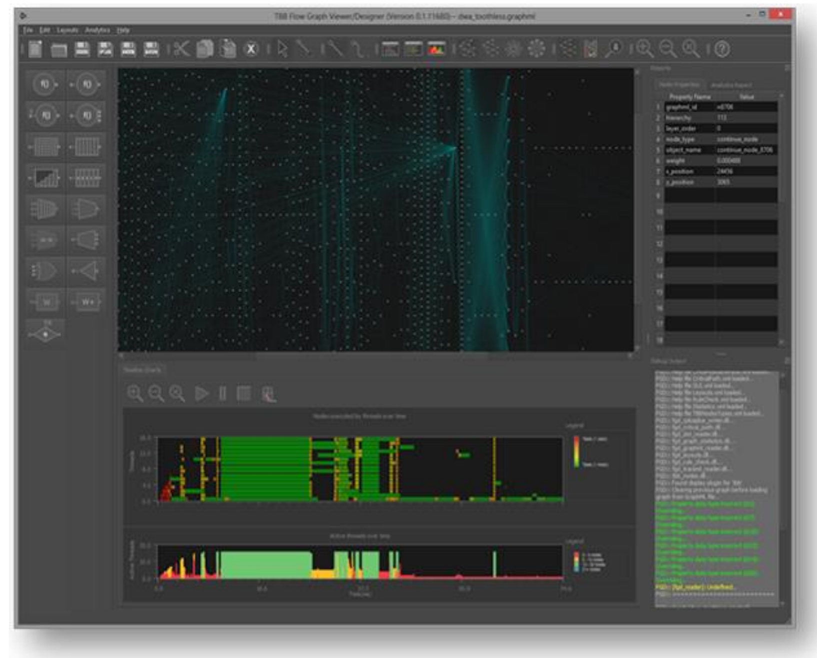
```
#include "tbb/tbb.h"
using namespace tbb;

void ParallelApplyFoo( float a[], size_t n ) {
    parallel_for(0, n, [&](int i) {
        Foo(a[i]);
    } );
}
```

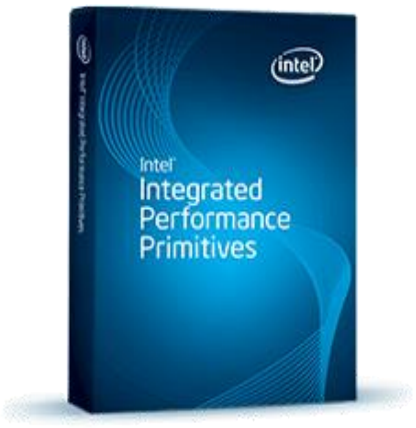
# Flow graph designer

(Alpha) Tool to assist developers in creating and tuning TBB applications

- ***As an analyzer***, it provides capabilities to collect and visualize execution traces from TBB flow graph applications. Users can explore the topology of their graphs, interact with a timeline of node executions, and project statistics of their graphs

- ***As a designer***, it provides the ability to visually create Intel TBB flow graph diagrams and generate C++ stubs for further development






# Intel® Integrated Performance Primitives

Intel® Parallel Studio XE Suite

Intel® System Studio Suite

# Intel® Integrated Performance Primitives

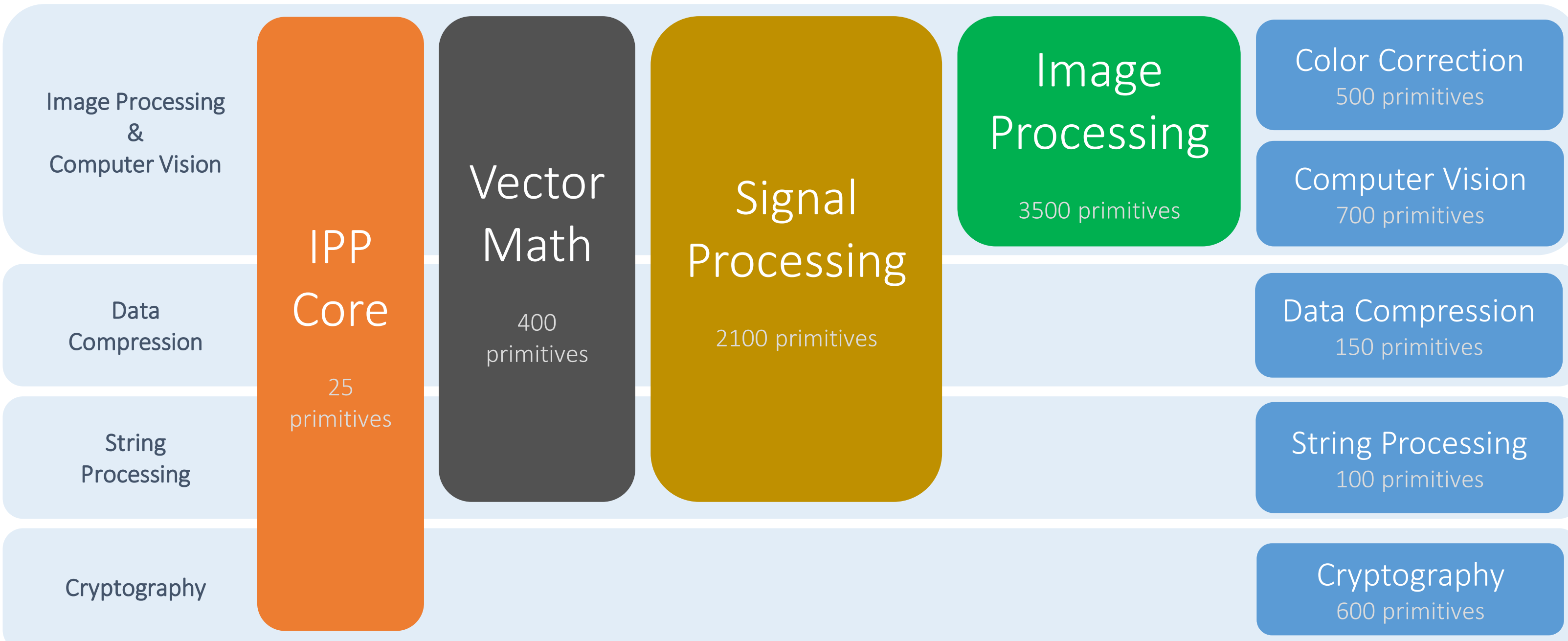
| Signal processing | Image processing | Computer vision | String processing | Data compression | Cryptography |
|---|---|---|---|---|---|
| One dimensional input data processing | 2D/3D input data processing (e.g., color conversion) | Optimization to accelerate common OpenCV functions | String manipulation and regular expression functionality | Huffman, VLC, and dictionary compression techniques | Support for standard cryptographic algorithms (not available in all countries) |

Intel® IPP: Extensive C/C++ performance library for multiple domains

- Multi-core-ready, computationally intensive optimized functions

Available on a wide variety of Intel platforms and OSs

# Primitives and supporting domains




## Function naming convention

- *ipp<data-domain><name>_<datatype>[_<descriptor>](<parameters>)*

# Intel® IPP and OpenCV 3.0

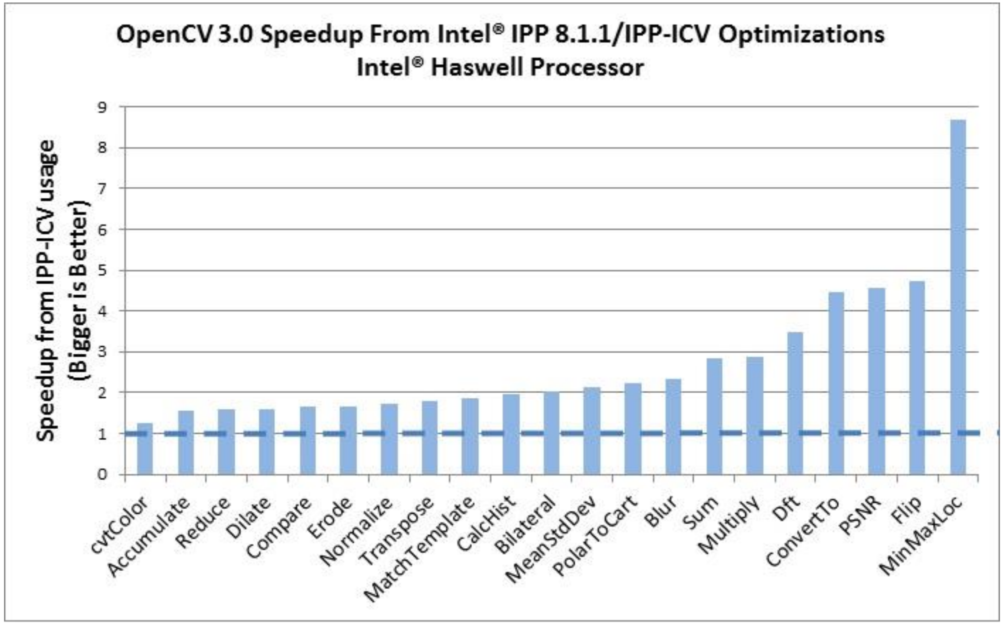
## OpenCV 3.0 (Open Source Computer Vision)

- Computer vision and machine learning software library
- Supports optimization on Intel platforms with Intel® IPP

## Intel® IPP for OpenCV (ICV)

- Subset of Intel® IPP, ~750 functions fully integrated into OpenCV 3.0
- Integration enabled by default on x86/intel64 configurations

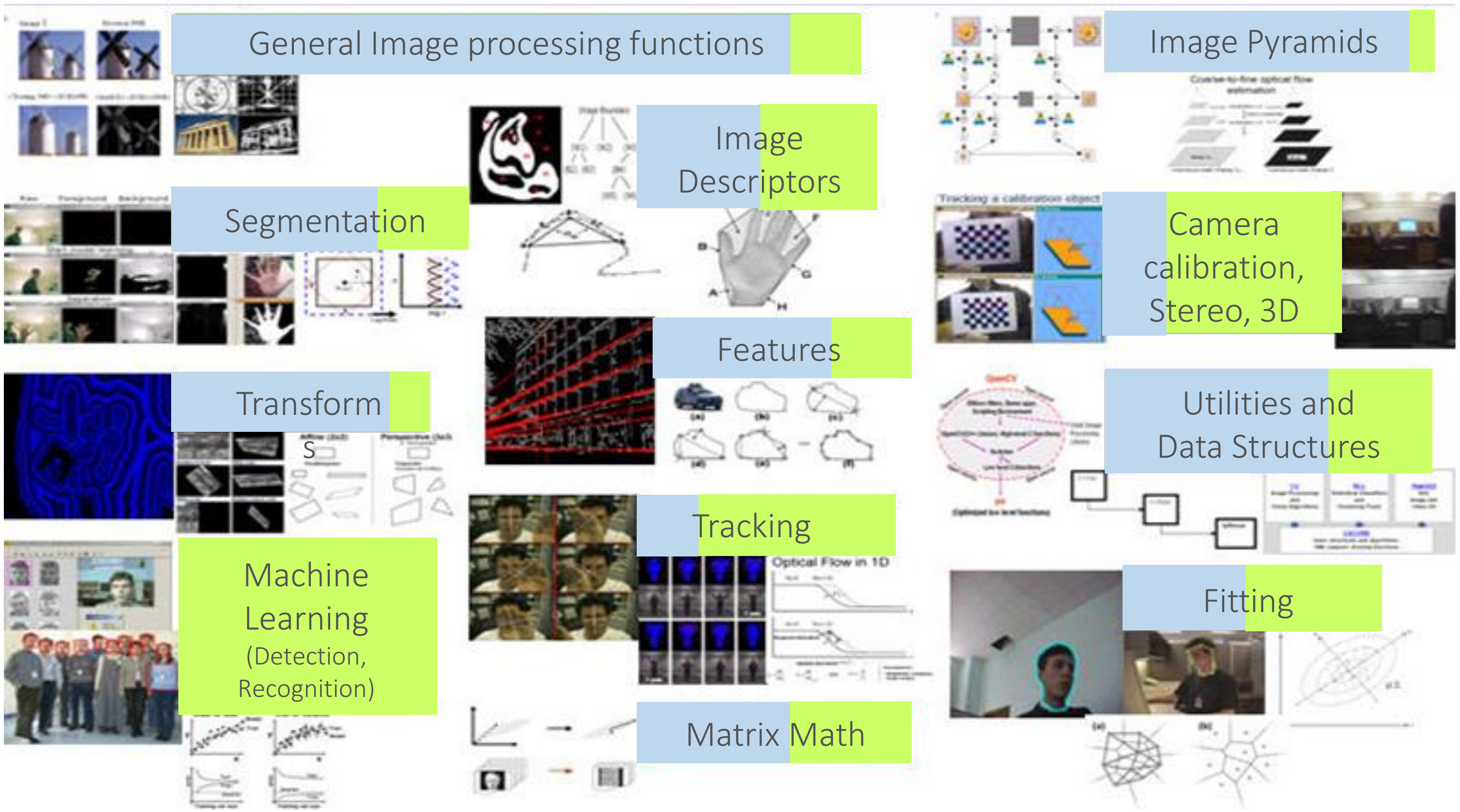
## ICV provides ~40% performance gains



# OpenCV 3.0 at glance

More than 500 algorithms (~60% benefit from Intel® IPP)

*Covered by IPP* *OpenCV only*



# Intel® IPP 9.0 main features

## Optimized for performance, throughput and power efficiency

- Dynamic dispatching of best host-based function version (including SIMD capabilities)
- Extensive support of latest Intel® processors/coprocessors

## Other feature highlights

- Integration of IPP subset (ICV) into OpenCV 3.0
- Static/dynamic, PIC/no-PIC library versions
- No internal memory allocation or threading

## What’s new in latest 9.0 release?

- New API for external threading
- Improved CPU dispatcher (including auto-initialization)
- Optimized cryptography functions to support SM2/SM3/SM4 algorithm
- Custom dynamic library building tool
- Additional optimizations for new Intel® processors/coprocessors


# Intel® Math Kernel Library

Intel® Parallel Studio XE Suite

Intel® System Studio Suite

# Intel® Math Kernel Library (Intel® MKL)


Intel® MKL: Collection of C/Fortran high-performance math routines for science, engineering and financial applications

- Extract great parallel performance with minimal effort
- De-facto industry standard APIs (C/Fortran)
- Additional API and environment variables for runtime configuration
- Support for Windows, Linux, and OS X

Optimized for performance on Intel® processors/coprocessors

- Dynamic dispatching of best host-based function version
  - Extensive use of SIMD extensions and optimal cache blocking factors
- Highly optimized sequential/parallel/cluster implementations

# Mathematical building blocks on Intel® MKL

## Linear Algebra
- BLAS
- LAPACK
- Sparse Solvers
  - Iterative
  - PARDISO (SMP)
  - PARDISO (cluster)
  - ScaLAPACK

## Fast Fourier Transforms
- Multi-dimensional
- FFTW interfaces
- Cluster FFT

## Vector Math
- Trigonometric
- Hyperbolic
- Exponential, Log
- Power/Root

## Vector RNGs
- Congruential
- Wichmann-Hill
- Mersenne Twister
- Sobol
- Neiderreiter
- Non-deterministic

## Summary Statistics
- Kurtosis
- Variation coefficient
- Order statistics
- Min/max
- Variance-covariance

## And More
- Splines
- Interpolation
- Trust Region
- Fast Poisson Solver

# Intel® MKL on Intel® Xeon Phi™

| | |
|---|---|
| **Automatic Offload** | • No code changes required<br>• Automatically uses both host and target (`MKL_MIC_ENABLE=1`)<br>• Transparent data transfer and execution management |
| **Compiler Assisted Offload** | • Explicit control for data transfer and remote execution<br>• Invoked with compiler `offload` or OpenMP `target` pragmas<br>• Can be used together with Automatic Offload |
| **Native Execution** | • Uses the coprocessors as independent nodes<br>• Input data and binaries are copied to targets in advance |

Intel® MKL provides full support for Intel® Xeon Phi™ coprocessor

BAYNCORE

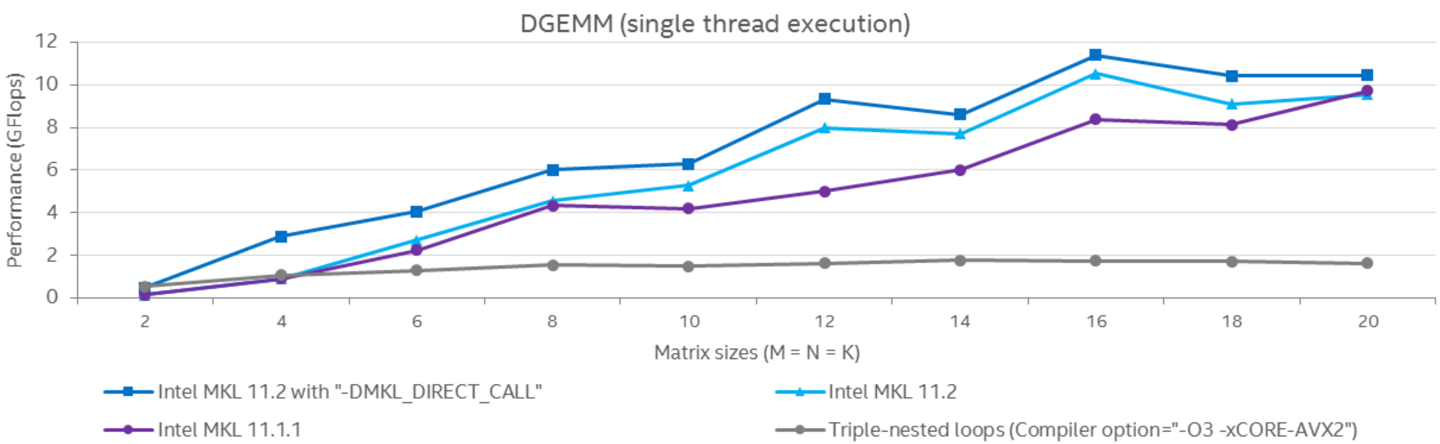
# *GEMM improvements on small matrices

Significant performance improvements for square sizes smaller than 20

- Applicable to all small sizes and input parameters
- No errors reported when incorrect parameters are passed to the function call

How to enable [small matrices optimization]

- Include `mkl_direct_call.fi`/`mkl_direct_call.h` module
- Compile with the option `MKL_DIRECT_CALL`/`MKL_DIRECT_CALL_SEQ` symbol



# What's new in Intel® MKL 11.3

Main new features an improvements

- Additional two-stage API for sparse BLAS2/3 routines
- MKL MPI wrappers
- Support for batched small *GEMM independent operations
- Support for Philox4x35 and ARS5 RNGs ($2^{128}$ period)
- Sparse solver SMP scalability improvements

Many other features and optimizations (check MKL 11.3 release notes)

- HBM support for 2nd generation of Intel® Xeon Phi™
- Improved MKL composability with Intel® TBB applications
- Cluster components now available for OS X
- Many BLAS/(Sca)LAPACK/PARDISO improvements
- Many improvements on latest AVX2/IMCI and future AVX-512 hardware

# Intel® MKL Cookbook

## Detailed [recipes](#) for solving complex problems with Intel® MKL

### Using LAPACK symmetric Eigen solvers for Hermitian tri-diagonal 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:

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

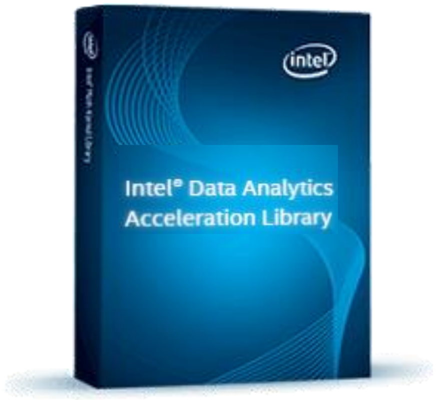
$c_j = \overline{e_j}, j = 1, 2, \ldots N-1$

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

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

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

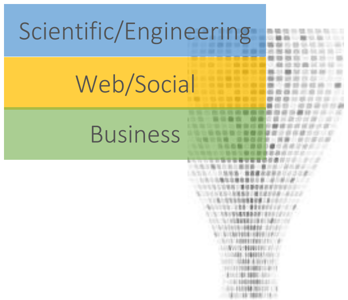
BAYNCORE




# Intel® Data Analytics Acceleration Library

Intel® Parallel Studio XE Suite 2016

# Intel® Data Analytics Acceleration Library




Intel® DAAL: An industry leading end-to-end IA-based data analytics acceleration library of fundamental algorithms covering all big data analysis stages

More information later in “*Coding high performance big data analytics applications*” session

| Pre-processing | Transformation | Analysis | Modeling | Validation | Decision Making |
|---|---|---|---|---|---|
| (De-)Compression<br>Filtering<br>Normalization | Aggregation<br>Dimension<br>Reduction | Summary<br>Statistics<br>Clustering | Classification<br>Regression<br>Association rules<br>Simulation | Hypothesis<br>Testing<br>Metrics | Prediction |

BAYNCORE

# Summary

**Intel® Parallel Studio XE 2016 tool suite to boost performance of parallel applications on Intel® processors/coprocessors**

Tool suite components

- High-performance C/C++ and Fortran compilers
- **Performance and parallel libraries**
- Design, tune, and verification tools

What's new in 16.0?

- **Free licensing for selected communities**
- Support for latest C/C++/Fortran standards
- **Improved performance and compatibility with new/future Intel hardware**
- **Intel® DAAL: new library for big data analytics**
- Intel® Vectorization Advisor: new design/analysis tool for vectorising your code

BAYNCORE

# Online resources

Intel® software development tools, performance tuning, etc.

| | |
|---|---|
| • Documentation library | All available documentation about Intel software |
| • HPC webinars | Free technical webinars about HPC on Intel platforms |
| • Modern code | Intel resources about code modernization |
| • Forums | Public discussions about Intel SIMD, threading, ISAs, etc. |

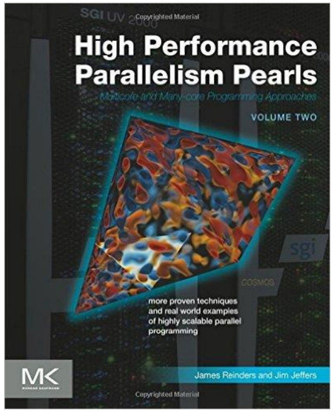
Intel® Xeon Phi™ resources

| | |
|---|---|
| • Developer portal | Programming guides, tools, trainings, case studies, etc. |
| • Solutions catalog | Existing Intel® Xeon Phi™ solutions for known codes |

Other resources (white papers, benchmarks, case studies, etc.)

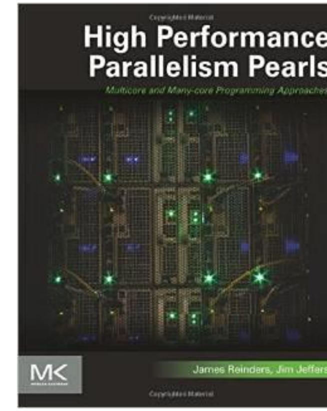
| | |
|---|---|
| • Go parallel | BKMs for Intel multi- and many-core architectures |
| • Colfax research | Publications and material on parallel programming |
| • Bayncore labs | Research and development activities (WIP) |

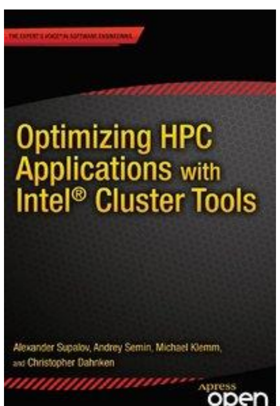
# Recommended books




*High performance parallelism pearls: multi-core and many-core approaches (Vol. 2)*, by James Reinders and Jim Jeffers, Morgan Kaufmann, 2015

*High performance parallelism pearls: multi-core and many-core approaches*, by James Reinders and Jim Jeffers, Morgan Kaufmann, 2014




*Optimizing HPC applications with Intel® cluster tools*, by Alexander Supalov et al, Apress, 2014

*Parallel programming with Intel® Parallel Studio XE*, by Stephen Blair-Chappell and Andrew Stokes, Wrox press, 2012