Agenda

- Intel® Math Kernel Library
- Intel® Integrated Performance Primitives
- Intel® Data Analytics Acceleration Library
Intel® Math Kernel Library (Intel® MKL)
Powered by the Intel® Math Kernel Library (Intel® MKL)

- Speeds math processing in scientific, engineering and financial applications
- Functionality for dense and sparse linear algebra (BLAS, LAPACK, PARDISO), FFTs, vector math, summary statistics and more
- Provides scientific programmers and domain scientists
  - Interfaces to de-facto standard APIs from C++, Fortran, C#, Python and more
  - Support for Linux*, Windows* and OS X* operating systems
  - Extract great performance with minimal effort
- Unleash the performance of Intel® Core, Intel® Xeon and Intel® Xeon Phi™ product families
  - Optimized for single core vectorization and cache utilization
  - Coupled with automatic OpenMP*-based parallelism for multi-core, manycore and coprocessors
  - Scales to PetaFlop (10^{15} floating-point operations/second) clusters and beyond
- Included in Intel® Parallel Studio XE and Intel® System Studio Suites

Used on the World’s Fastest Supercomputers**

**http://www.top500.org
Optimized Mathematical Building Blocks
Intel MKL

**Linear Algebra**
- BLAS
- LAPACK
- ScaLAPACK
- Sparse BLAS
- Sparse Solvers
  - Iterative
  - PARDISO* SMP & Cluster

**Fast Fourier Transforms**
- Multidimensional
- 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
Automatic Dispatching to Tuned ISA-specific Code Paths

More cores → More Threads → Wider vectors

<table>
<thead>
<tr>
<th></th>
<th>Intel® Xeon® Processor 64-bit</th>
<th>Intel® Xeon® Processor 5100 series</th>
<th>Intel® Xeon® Processor 5500 series</th>
<th>Intel® Xeon® Processor 5600 series</th>
<th>Intel® Xeon® Processor E5-2600 v2 series</th>
<th>Intel® Xeon® Processor E5-2600 v3 series</th>
<th>~ Future Intel® Xeon® Processor</th>
<th>Intel® Xeon Phi™ x100 Coprocessor</th>
<th>Intel® Xeon Phi™ x200 Processor &amp; Coprocessor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up to Core(s)</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>12</td>
<td>18</td>
<td>~</td>
<td>57-61</td>
<td>TBD</td>
</tr>
<tr>
<td>Up to Threads</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>12</td>
<td>24</td>
<td>36</td>
<td>tbd</td>
<td>228-244</td>
<td>TBD</td>
</tr>
<tr>
<td>SIMD Width</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>256</td>
<td>256</td>
<td>~</td>
<td>512</td>
<td>512</td>
</tr>
<tr>
<td>Vector ISA</td>
<td>Intel® SSE3</td>
<td>Intel® SSE3</td>
<td>Intel® SSE4.2</td>
<td>Intel® AVX</td>
<td>Intel® AVX</td>
<td>Intel® AVX2</td>
<td>Intel® AVX-512</td>
<td>IMCI 512</td>
<td>Intel® AVX-512</td>
</tr>
</tbody>
</table>

Product specification for launched and shipped products available on ark.intel.com.

All dates and products specified are for planning purposes only and are subject to change without notice.

1. Not launched or in planning.
The latest version of Intel® MKL unleashes the performance benefits of Intel architectures
Notable Enhancements in Intel® MKL 11.0-11.2

Optimizations – current and future hardware support
- Always highly tuned for the latest Intel® Xeon® processor
- Tuned for Intel® Xeon Phi™ Coprocessor x100 (KNC)

Features
- Conditional Numerical Reproducibility
- Extended Eigensolvers based on and compatible with FEAST¹
- Parallel Direct Sparse Solver for Clusters
- Small Matrix Multiply enhancements

Notes:
¹ http://www.ecs.umass.edu/~polizzi/feast/
New Enhancements in Intel® MKL 11.3 (Part 1)

Optimized for the latest Intel® Xeon® processors and for Intel® Xeon Phi™ x200 Coprocessor (KNL)
- Support MCDRM on KNL

Batch GEMM functions
- Improve the performance of multiple, simultaneous matrix multiply operations
- Provides grouping (the same sizes and leading dimensions) and batching across groups

GEMMMT functions calculate \( C = A * S * A^T \), where \( S \) is symmetric and/or diagonal

Sparse BLAS inspector-executor API
- Matrix structure analysis brings performance benefit for relevant applications (i.e. iterative solvers)
- Parallel triangular solver
- Both 0-based and 1-based indexing, row-major and column-major ordering
- Extended BSR support

Counter-based pseudorandom number generators
- ARS-5 based on the Intel AES-NI instruction set
- Philox4x32-10
New Enhancements in Intel® MKL 11.3 (Part 2)

Intel MKL PARDISO scalability
- Improved Intel MKL PARDISO and Cluster Sparse Solver scalability on Intel Xeon Phi coprocessors

Cluster components extension
- MPI wrappers provide compatibility with most MPI implementations including custom ones
- Cluster components support on OS X

Intel® Threaded Building Blocks(Intel® TBB) threading layer
- Provide composability with TBB-based applications
- BLAS level-3, some LAPACK functions, and the Poisson solver
Using MCDRM on KNL

MKL uses memkind library to allocate memory to MCDRM.

- Fail over to regular system allocator
- mkl_malloc, mkl_calloc, and mkl_realloc also try to allocate memory to MCDRM
- Additional service functions and env-variables to control amount of MCDRM available to MKL
Batch Matrix-Matrix Multiplication

Compute independent matrix-matrix multiplications (GEMMs) simultaneously with a single function call
- Supports all precisions of GEMM and GEMM3D
- Handles varying matrix sizes with a single function call
- Better utilizes multi/many-core processors for small sizes
GEMMMT Functionality

Calculates $C = A \times S \times A^T$, where $S$ is symmetric and/or diagonal

```c
?gemmt(uplo, transa, transb, n, k, alpha, a, lda, b, ldb, beta, c, ldc)
```

Naming
- S/D/C/Z: supported precisions
- GE: general matrix
- MM: matrix-matrix Product
- T: triangular part of result matrix updated

Parameters
- uplo: specifies whether upper or lower triangular part of the array $c$ is used
- transa, transb: specifies form of $\text{op}(A,B)$ used in the matrix multiplication
- n, k: sizes of the input matrices
- lda, ldb, ldc: leading dimensions of the input matrices
- a, b: arrays containing matrices to be multiplied
- c: array containing triangular part of result matrix
Inspector-Executor Sparse BLAS API

Two-step API provides advanced sparse optimizations

1. Inspect step – analyze matrix to choose best strategy
   - Computational kernels for portrait
   - Balancing strategy for parallel execution

2. Execute step – use analysis data to get better performance
   - Optimization applied to get better performance
   - Level chosen based on expected number of iterations
Intel® TBB Threading Layer

Scope

- BLAS: Level-3 functions, gemv, dot
- LAPACK: getrf, getrs, gesv potrs, potrf, geqrf, gels
- Poisson solver

On roadmap

- More LAPACK: pstrf, syev, gels, gelsy
- Sparse components: csrmv, bsrmv, PARDISO

Sequential execution for the rest of MKL functions

Performance

- 80% performance of OpenMP-based MKL on free system.
Intel® MKL Futures & Technology Previews

Intel® Optimized Technology Preview for High Performance Conjugate Gradient (HPCG) Benchmark

- Proposed to supplement the current High Performance Linpack Benchmark
- Designed to be more representative of common application workloads

Availability – contact intel.mkl@intel.com

We seek your insights into defining new Intel MKL features
Intel® Integrated Performance Primitives (Intel® IPP)
Intel® Integrated Performance Primitives

**Optimized for Performance & Power Efficiency**

- Highly tuned routines
- Highly optimized using SSSE3, SSE, and AVX, AVX2 instruction sets
- Performance beyond what an optimized compiler produces alone

**Intel Engineered & Future Proofed to Shorten Development Time**

- Fully optimized for current and past processors
- Save development, debug, and maintenance time
- Code once now, receive future optimizations later

**Wide Range of Cross Platform & OS Functionality**

- Thousands of highly optimized signal, data and media functions
- Broad domain support
- Supports Intel® Quark, Intel® Atom™, Core™, Xeon® and Xeon® Phi processors

### Signal Processing (1D)

<table>
<thead>
<tr>
<th>Filters</th>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT</td>
<td>Mean</td>
</tr>
<tr>
<td>FIR</td>
<td>StdDev</td>
</tr>
<tr>
<td>Threshold</td>
<td>NormDiff</td>
</tr>
<tr>
<td>Convolution</td>
<td>Sum</td>
</tr>
<tr>
<td>Median</td>
<td>MinMax</td>
</tr>
</tbody>
</table>

### Image & Frame Processing (2D)

<table>
<thead>
<tr>
<th>Transforms</th>
<th>Filters</th>
<th>Computer Vision</th>
<th>Color Conversion</th>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT</td>
<td>Convolution</td>
<td>Canny</td>
<td>RGB/BGR</td>
<td>Mean</td>
</tr>
<tr>
<td>Resize</td>
<td>Morphology</td>
<td>Optical Flow</td>
<td>YUV/YCbCr</td>
<td>StdDev</td>
</tr>
<tr>
<td>Rotate</td>
<td>Threshold</td>
<td>Segmentation</td>
<td>420, 422, 444</td>
<td>NormDiff</td>
</tr>
<tr>
<td>Mirror</td>
<td>Histogram</td>
<td>Haar Classifiers</td>
<td></td>
<td>Sum</td>
</tr>
<tr>
<td>Warp/Shear</td>
<td></td>
<td>Hough Transform</td>
<td></td>
<td>MinMax</td>
</tr>
</tbody>
</table>

Performance building blocks for image, signal and data processing applications
# Intel® IPP Domains

## Image Processing and Computer Vision

<table>
<thead>
<tr>
<th>Domain</th>
<th>Functions and Algorithms</th>
</tr>
</thead>
</table>
| **Image Processing** | • Geometry transformations, such as resize/rotate.  
  • Linear and non-linear filtering operation on an image for edge detection, blurring, noise removal, etc.  
  • Linear transforms for 2D FFTs, DFTs, DCT.  
  • Image statistics and analysis. |
| **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 |
| **Color Models**     | • Convert image/video color space formats: RGB, HSV, YUV, YCbCr,  
  • Up/Down sampling,  
  • Brightness and contrast adjustments |
## Intel® IPP Domains

**Signal Processing and Digital Filters**

<table>
<thead>
<tr>
<th>Domain</th>
<th>Functions and Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Signal Processing</strong></td>
<td>• FFT, DFT, DCT, MDCT, Wavelet, Hilbert, Hartley and Walsh-Hadamard Transforms</td>
</tr>
<tr>
<td></td>
<td>• Convolution, Cross-Correlation, Auto-Correlation, Conjugate</td>
</tr>
<tr>
<td></td>
<td>• Windowing, Jaehne/Tone/Triangle signal generation</td>
</tr>
<tr>
<td><strong>Digital Filtering</strong></td>
<td>• Finite Impulse Response (FIR), Infinite Impulse Response (IIR), Single-Rate Adaptive FIR, Multi-Rate Adaptive FIR, Median Filter, Convolution and Correlation,</td>
</tr>
<tr>
<td></td>
<td>• Coordinate Conversions (polar↔cartesian), Numeric Conversion (real↔complex), Emphasize,</td>
</tr>
<tr>
<td></td>
<td>Nearest Neighbor, Threshold, etc.</td>
</tr>
<tr>
<td><strong>Statistical</strong></td>
<td>• Sum, Max, Min, Mean, Standard Deviation, Norm, Dot Product, Zero Cross, Count</td>
</tr>
</tbody>
</table>
# Intel® IPP Domains

## Data Processing and Compression

<table>
<thead>
<tr>
<th>Domain</th>
<th>Functions and Algorithms</th>
</tr>
</thead>
</table>
| **Data Compression** | • Entropy-coding compression: Huffman and VLC  
• Dictionary-based compression: LZSS, LZ77 (ZLIB) and LZO  
• Burrows-Wheeler Transform (BWT)  
  Generalized Interval Transformation (GIT)  
  MoveToFront (MTF)  
  Run-Length-Encoding (RLE)  
• bzip2 compatible functions                                                                                                                                   |
| **Cryptography**  | • Symmetric Cryptography: DES/TDES, Rijndael, AES-CCM, AES-GCM, Blowfish, Twofish, RC5*, ARCFour  
• Data Integrity Hash Functions: MD5, SHA, Reed-Solomon  
• Data Authentication: Keyed Hash, CMAC, AES-XCBC and DAA  
• Public Key: Big Number Arithmetic, Montgomery Reduction Scheme, Pseudorandom Number, Prime Number, RSA,  
  Discrete-Logarithm-Based, Elliptic Curve, Finite Field Arithmetic, Elliptic Curve Points, Tate Pairing |
| **String Processing** | • Find, Insert, Remove, Compare, Trim, Replace, Upper, Lower, Hash, Concatenate, Split and Regular Expression  
  Find/Replace                                                                                                                                                    |
Intel® Integrated Performance Primitives Image Resize Performance vs. OpenCV

Intel® Integrated Performance Primitives (Intel® IPP) 8.0 Resize Performance Comparison

System configuration: Intel® IPP 8.0 (build 83); Precompiled OpenCV 2.4.6. Hardware: Intel® Core™ i7 4770s Processor, 3.10 GHz, 256 KB L2 Cache, 8 GB RAM; Operating System: Windows® 8 64 bit; Visual Studio 2012; Single Threaded; Benchmark source: Intel® Corporation; location: Folsom, CA, 02/07/2013; Notes: Linear, Cubic (Catmul-Rom), and Lanczos(3 node) interpolation, 1 channel (greyscale) 1280x720 source image.

Intel® IPP provides faster resize
Intel® IPP provides faster filter operations
Intel® IPP Function Naming Convention and Usage

Function names

- are easy to understand
- directly indicate the purpose of the function via distinct elements
- each element has a fixed number of pre-defined values

<table>
<thead>
<tr>
<th>Name Elements</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Prefix</strong></td>
<td>Indicates the functional data type in 1D, 2D and Matrix</td>
<td>ipps, ippi, ippm</td>
</tr>
<tr>
<td><strong>Base Name</strong></td>
<td>Abbreviation for the core operation</td>
<td>Add, FFTFwd, LuDecomp</td>
</tr>
<tr>
<td><strong>Data Type</strong></td>
<td>Describes bit depth and sign</td>
<td>8u, 32f, 64f</td>
</tr>
<tr>
<td><strong>Execution mode</strong></td>
<td>Indicates data layout and scaling</td>
<td>ISfs, C1R, P</td>
</tr>
</tbody>
</table>

Description: The image (ippi) copy (Copy) operation operating on data of type 8u (8u) in one-channel image (C1), operation with mask (M) in the region of interest (R).

Each function performs a particular operation on a known type of data in a specific mode.
Function Implementation

- Intel IPP functions are optimized for a specific processor.
- A single function has many version, each one optimized to run on a specific processor.
- The name of each version is decorated with a prefix that denotes its target processor, see table below for IA-32 prefixes full list.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Identifier</th>
<th>Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>IA-32 Intel® Architecture</td>
<td>px</td>
<td>C-optimized for all IA-32 processors</td>
</tr>
<tr>
<td></td>
<td>v8</td>
<td>Optimized for processors with Intel® Supplemental Streaming SIMD Extensions 3 (Intel SSSE3)</td>
</tr>
<tr>
<td></td>
<td>p8</td>
<td>Optimized for processors with Intel® Streaming SIMD Extensions 4.1 (Intel SSE4.1)</td>
</tr>
<tr>
<td></td>
<td>s8</td>
<td>Optimized for the Intel® Atom™ processor</td>
</tr>
<tr>
<td>Intel® 64 (Intel® EM64T) architecture</td>
<td>mx</td>
<td>C-optimized for processors with Intel® 64 instructions set architecture</td>
</tr>
<tr>
<td></td>
<td>u8</td>
<td>Optimized for 64-bit applications on processors with Intel® Supplemental Streaming SIMD Extensions 3 (Intel SSE3)</td>
</tr>
<tr>
<td></td>
<td>y8</td>
<td>Optimized for 64-bit applications on processors with Intel® Streaming SIMD Extensions 4.1 (Intel SSE4.1)</td>
</tr>
<tr>
<td></td>
<td>n8</td>
<td>Optimized for the Intel® Atom™ processor</td>
</tr>
<tr>
<td></td>
<td>e9</td>
<td>Optimized for processors that support Intel® Advanced Vector Extensions instruction set</td>
</tr>
</tbody>
</table>

ippsCopy_8u(…)

Some examples:
- px_ippsCopy_8u(…)
- s8_ippsCopy_8u(…)
- p8_ippsCopy_8u(…)

Intel ® IPP gets updated with these libs to match the latest CPU features
## Intel® IPP Linkage

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Optimization</strong></td>
<td>All processors</td>
<td>All processors</td>
<td>All processors</td>
<td>One processor</td>
</tr>
<tr>
<td><strong>Build</strong></td>
<td>Link to stub libraries</td>
<td>Link to static libraries and stubs</td>
<td>Build separate DLL</td>
<td>Link to processor-specific merged libraries</td>
</tr>
<tr>
<td><strong>Calling</strong></td>
<td>Regular names</td>
<td>Regular names</td>
<td>Modified names</td>
<td>Processor-specific names</td>
</tr>
<tr>
<td><strong>Total Binary Size</strong></td>
<td>Large</td>
<td>Small</td>
<td>Small</td>
<td>Smallest</td>
</tr>
<tr>
<td><strong>Executable Size</strong></td>
<td>Smallest</td>
<td>Small</td>
<td>Smallest</td>
<td>Small</td>
</tr>
<tr>
<td><strong>Kernel Mode</strong></td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*Intel® IPP provides a lot of flexibility*
Intel® IPP 9.0 New Features

Additional optimization for Intel® Quark™, Intel® Atom™, and the processors with Intel® AVX2 instructions support

- Intel® Quark™: cryptography optimization
- Intel® Atom™: computer vision, image processing optimization
- Intel® AVX2: computer vision, image processing optimization

New APIs to support external threading

Improved CPU dispatcher

- Auto-initialization. No need for the CPU initialization call in static libraries.
- Code dispatching based on CPU features

Optimized cryptography functions to support SM2/SM3/SM4 algorithm

Custom dynamic library building tool

New APIs to support external memory allocation
Intel IPP Threading

• Intel® IPP library primitives are “thread-safe”

• Intel IPP internal threaded library are available as optional installation.

• The external threading is recommended, which is more effective than the internal threading
External Memory Allocation

- Intel IPP 9.0 removes the internal memory allocations in single-threaded libraries
- All memory allocations should be done at the application level
  - The legacy InitAlloc/Free functions are removed from Intel IPP 9.0
  - Use the substitution GetSize/Init functions
  - Additional GetSize/Init functions added to support external memory allocation
  - New redesigned geometry functionality to remove the internal memory allocation

```
ippsDFTInitAlloc_xx( &pSpec, ...)  
... ...
ippsDFTFwd_xx(pSpec)  

ippsDFTGetSize_xx(&size, ...)  
pSpec=ippsMalloc_xx(size, ...)  
(or use any user's memory allocation function)  
ippsDFTInit(pSpec, ...)  
... ...
ippsDFTFwd_xx(pSpec)  
```
Optimization Notice

Copyright © 2014, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

External Memory Allocation (Cont.)

• Reduce the memory allocation for different IPP function calls
  - A shared memory buffer can be used for different APIs

Internal Memory Allocation APIs (Deprecated)

A Memory
  Spec (const) part
  AA Buffer part

B Memory
  Spec (const) part
  BB Buffer part

C Memory
  Spec (const) part
  CC Buffer part

External Memory Allocation APIs (New)

<...>
  ippAAInitAlloc(..., pStateA);
  ippBBInitAlloc(..., pStateB);
  ippCCInitAlloc(..., pStateC);
<...>
  ippAAProcess(..., pStateA);
  ippBBProcess(..., pStateB);
  ippCCProcess(..., pStateC);
<...>
  ippAAFree(pStateA);
  ippBBFree(pStateB);
  ippCCFree(pStateC);
<...>

A Memory Spec (const) part
AA Buffer part

B Memory Spec (const) part
BB Buffer part

C Memory Spec (const) part
CC Buffer part

Memory
SpecA part
SpecB part
SpecC part

Shared memory buffer for A, B, C

maxSize = MAX(bufSizeA, bufSizeB, bufSizeC);
pBuffer = malloc(maxBufSize);
<...>
  ippAAInit(..., pSpecA);
  ippAAProcess(..., pSpecA, pBuffer);
  ippBBInit(..., pSpecB);
  ippBBProcess(..., pSpecB, pBuffer);
  ippCCInit(..., pSpecC);
  ippCCProcess(..., pSpecC, pBuffer);
<...>

External Memory Allocation APIs (New)
New custom Dynamic Library Building Tool

- A tool that can create the custom .dll or .so from Intel IPP static libraries:
  - Link with Intel IPP dynamically, but no need to redistribute IPP dynamic libraries.
  - Create the dynamic library containing the selected functions only.
  - Significantly reduce the size of dynamic libraries distributed with the applications.
External Threading Support

- The external threading is recommended, which is more effective than the internal threading
- Many of Intel IPP APIs were updated to enable external threading support
- To use the external threading, users need to handle the “border” data for some of the IPP APIs

```c
# The delay line is used to keep old data.
dlyLineLen = tapLen -1;

ippsFIRSRGetSize_32f(... &specSize,&bufSize);
ippsFIRSRInit_32f(filterTaps,tapLen,...,pSpec);

len = LEN/NUNTHREADS;  // simplified code, not consider for tail data

for(iThread=0;iThread< NUNTHREADS;iThread++)  // it means parallel for
{
    Ipp32f* pSrc = input+iThread*len;
    Ipp32f* pDst = output+iThread*len;

    if( iThread == 0)
        ippsFIR_32f( pSrc, pDst, len, pSpec, NULL, NULL , buffer);

    else if (iThread == NUMTHREADS - 1)
        ippsFIR_32f(pSrc, pDst, len, pSpec, pSrc-dlyLineLen, OutDlyLine, buffer);

    else
        ippsFIR_32f(pSrc,pDst, len, pSpec, pSrc-dlyLineLen, NULL , buffer);
}```
Legacy Domain

• A new domain to help move to the new versions of Intel IPP
• Providing the separated static and dynamic libraries containing the removed deprecated Intel IPP functions
• Fully independent from the Intel IPP main package
  – Use different function prefix
  – Contains internal CPU dispatcher
  – The legacy optimization are available
  – The legacy domain and the new versions of Intel IPP can be linked together
Intel® Data Analytics Acceleration Library (Intel® DAAL)
Data Analytics in the Age of Big Data

Problem:
- Big data needs high performance computing.
- Many big data applications leave performance at the table – Not optimized for underlying hardware.

Solution:
- A performance library provides building blocks to be easily integrated into big data analytics workflow.
Where Intel DAAL Fits?

Intel® Data Analytics Acceleration Library

Analysis
- PCA
- Low order moments
- Matrix factorization
- Outlier detection
- Distances
- Association rules

Machine learning
- Regression
  - Linear regression
  - Decision trees
- Classification
  - SVM
  - Naïve Bayes
  - Boosting algorithms
- Recommendation
  - ALS
- Clustering
  - K-Means
  - EM for GMM

Programming languages
- C++
- Java

Processing modes
- Batch processing
- Distributed processing
- Online processing

Data mining

Recommendation engines

Customer behavior modeling

BI analytics

Real time analytics

All data sources
- Finance
- Social media
- CRM
- Marketing
- Sensors, devices
- Ad campaigns
- Manufacturing

SQL stores

NoSQL stores

In-memory stores

Connectors

Big data frameworks: Hadoop, Spark, Cassandra, etc.
Intel® Data Analytics Acceleration Library 2016

A library optimized for Intel® Architectures that provides building blocks for all data analytics stages, from data preparation to data mining and machine learning.

- C++ and Java APIs in the initial release
- Windows*, Linux*, and OS X*.
- IA-32 and Intel64 support.
- Static and dynamic linking.
- Can be used in many platforms (Hadoop*, Spark*, R*, Matlab*, …) but not tied to any of them
- Flexible interface to connect to different data sources (CSV, SQL, HDFS, …)
Intel DAAL Components

Data Management
Interfaces for data representation and access. Connectors to a variety of data sources and data formats, such as HDFS, SQL, CSV, ARFF, and user-defined data source/format.

Data Sources
Numeric Tables
Compression / Decompression
Serialization / Deserialization

Data Processing
Optimized analytics building blocks for all data analysis stages, from data acquisition to data mining and machine learning.

Data Modeling
Data structures for model representation, and operations to derive model-based predictions and conclusions.
Data Transformation and Analysis (Intel® DAAL)

- Basic statistics for datasets
  - Statistical moments
  - Quantiles
- Correlation and dependence
  - Cosine distance
  - Correlation distance
  - Variance-Covariance matrix
- Matrix factorizations:
  - SVD
  - QR
  - Cholesky
- Dimensionality reduction
  - PCA
  - Association rule mining (Apriori)
- Outlier detection
  - Univariate
  - Multivariate

*Algorithms support streaming and distributed processing in the current release.*
Supervised learning

Regression

- Linear Regression
- Weak learner
- Boosting (Ada, Brown, Logit)
- Naïve Bayes
- SVM

Classification

Unsupervised learning

- K-Means Clustering
- EM for GMM

Collaborative filtering

- ALS

To be available in future releases

Algorithms support streaming and distributed processing in the current release.
Processing modes

**Batch Processing**

\[ R = F(D_1, \ldots, D_k) \]

**Streaming Processing**

\[ S_{i+1} = T(S_i, D_i) \]
\[ R_{i+1} = F(S_{i+1}) \]

**Distributed Processing**

\[ R = F(R_1, \ldots, R_k) \]
Batch Processing

All data fits in memory on a single node.
All DAAL algorithms support batch processing.
Online Processing

Data being chunked into memory piece by piece.
Work on one piece at a time, combine results at the end.
Not all DAAL algorithms support online processing
Distributed Processing

We have a cluster ...

Not all DAAL algorithms support distributed processing.
Distributed Processing: Communication

DAAL does not provide communication functions.

How data communication is achieved depends on the cluster platform.

- MPI clusters – Explicit communication paradigm
  - User application calls MPI_Send() and MPI_Recv()
- Hadoop, Spark – Implicit communication paradigm
  - No explicit send/recv calls needed.
  - Communication taken care of by Hadoop or Spark.
PCA Performance Boosts Using Intel® DAAL vs. Spark* MLLib on Intel® Architectures

![Graph of PCA (correlation method) on an 8-node Hadoop* cluster based on Intel® Xeon® Processors E5-2697 v3](image)

Table size

<table>
<thead>
<tr>
<th>Table size</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1M x 200</td>
<td>4X</td>
</tr>
<tr>
<td>1M x 400</td>
<td>6X</td>
</tr>
<tr>
<td>1M x 600</td>
<td>6X</td>
</tr>
<tr>
<td>1M x 800</td>
<td>7X</td>
</tr>
<tr>
<td>1M x 1000</td>
<td>7X</td>
</tr>
</tbody>
</table>

Configuration Info - Versions: Intel® Data Analytics Acceleration Library 2016, CDH v5.3.1, Apache Spark* v1.2.0; Hardware: Intel® Xeon® Processor E5-2699 v3, 2 Eighteen-core CPUs (45MB LLC, 2.3GHz), 128GB of RAM per node; Operating System: CentOS 6.6 x86_64.

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. * Other brands and names are the property of their respective owners. Benchmark Source: Intel Corporation

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.
Call to Action

Check out Intel performance libraries product pages:


Community licensing program for Intel performance libraries:

- [https://software.intel.com/sites/campaigns/nest/](https://software.intel.com/sites/campaigns/nest/)
- MKL, IPP, DAAL, and TBB
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.

Copyright © 2014, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

<table>
<thead>
<tr>
<th>Optimization Notice</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>

Notice revision #20110804