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

**http://www.top500.org

Used on the World's Fastest Supercomputers**

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

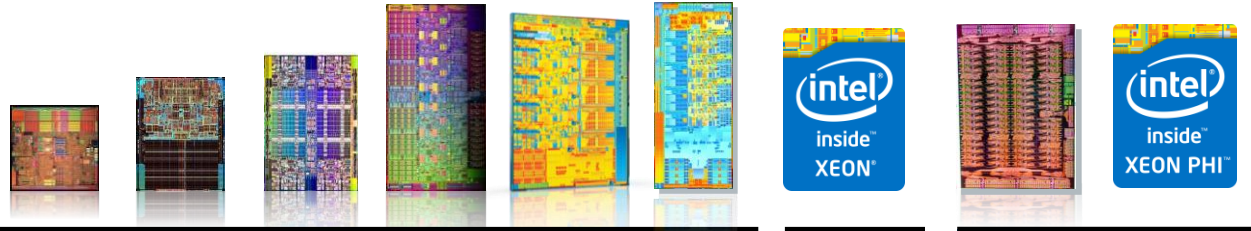
- Kurtosis
- Variation coefficient
- Order statistics
- Min/max
- Variance-covariance

And More

- Splines
- Interpolation
- Trust Region
- Fast Poisson Solver

Automatic Dispatching to Tuned ISA-specific Code Paths

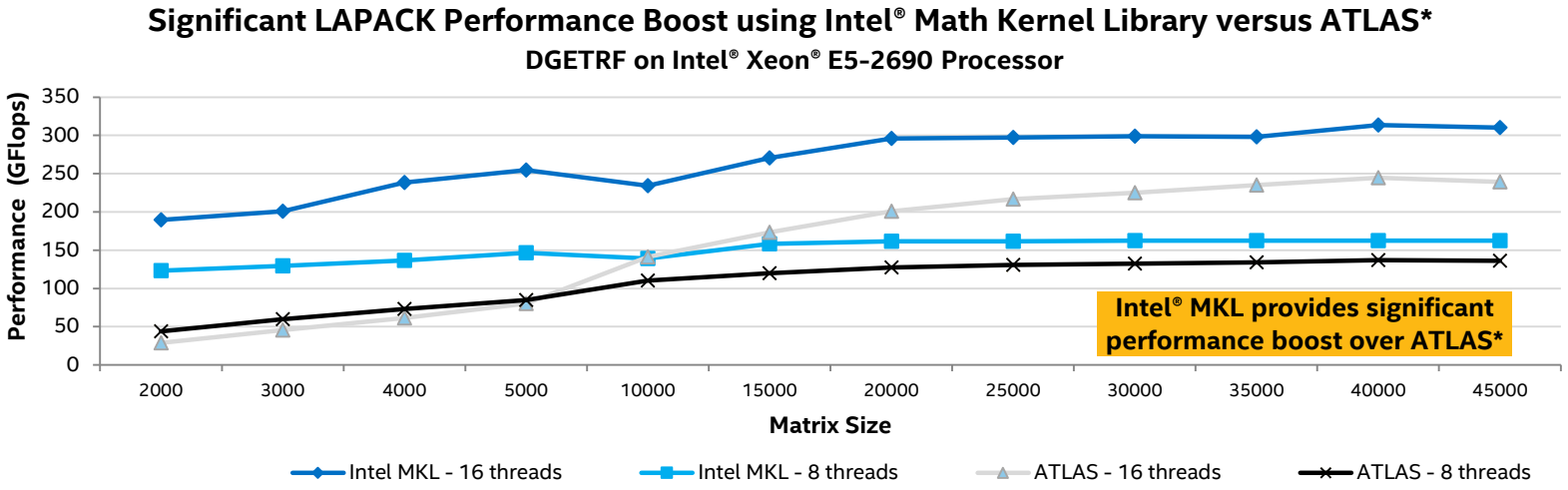
More cores → More Threads → Wider vectors



	Intel® Xeon® Processor 64-bit	Intel® Xeon® Processor 5100 series	Intel® Xeon® Processor 5500 series	Intel® Xeon® Processor 5600 series	Intel® Xeon® Processor E5-2600 v2 series	Intel® Xeon® Processor E5-2600 v3 series	~	Future Intel® Xeon® Processor[1]	Intel® Xeon Phi™ x100 Coprocessor	Intel® Xeon Phi™ x200 Processor & Coprocessor
Up to Core(s)	1	2	4	6	12	18	~	*Tbd*	57-61	TBD
Up to Threads	2	2	8	12	24	36		*tbd*	228-244	TBD
SIMD Width	128	128	128	128	256	256	~	512	512	512
Vector ISA	Intel® SSE3	Intel® SSE3	Intel® SSE4.2	Intel® AVX	Intel® AVX	Intel® AVX2		Intel® AVX-512	IMCI 512	Intel® AVX-512

Product specification for launched and shipped products available on ark.intel.com. 1. Not launched or in planning.
All dates and products specified are for planning purposes only and are subject to change without notice

Performance Benefit to Applications



Configuration: Hardware: CPU: Dual Intel® Xeon E5-2697v2@2.70Ghz; 64 GB RAM. Interconnect: Mellanox Technologies* MT27500 Family [ConnectX*-3] FDR. Software: RedHat* RHEL 6.2; OFED 3.5-2; Intel® MPI Library 5.0 Intel® MPI Benchmarks 3.2.4 (default parameters; built with Intel® C++ Compiler XE 13.1.1 for Linux*);

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.

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 FEAST[1]
- 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

GEMMT 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 trie 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

?GEMMT Functionality

Calculates C = A * S * A^T, where S is symmetric and/or diagonal

?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 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

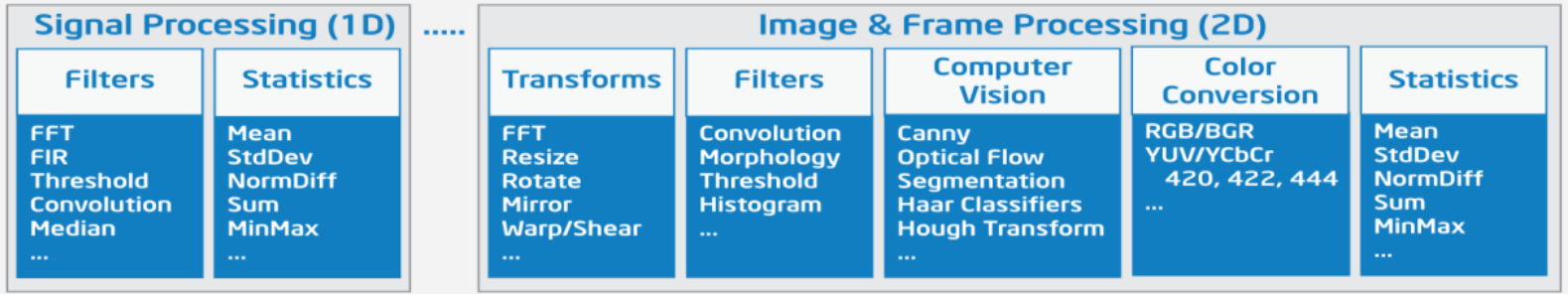
- Highly tuned routines
- Highly optimized using SSSE3, SSE, and AVX, AVX2 instruction sets
- Performance beyond what an optimized compiler produces alone

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



Performance building blocks for image, signal and data processing applications

Intel® IPP Domains

Image Processing and Computer Vision

Domain	Functions and Algorithms
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

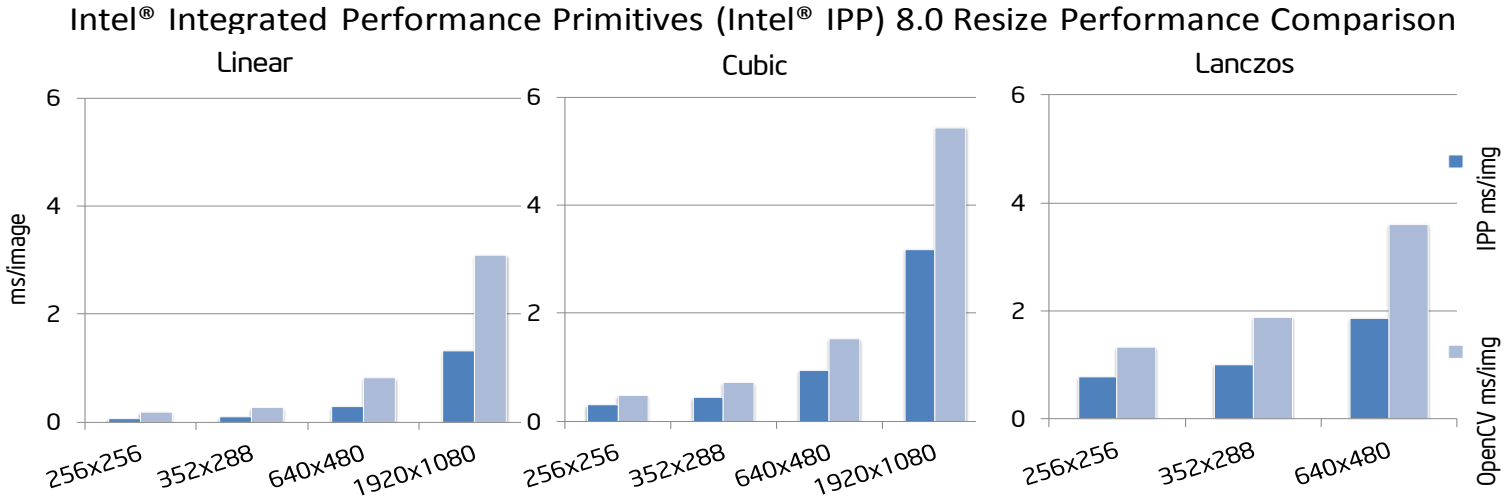
Domain	Functions and Algorithms
Signal Processing	• FFT, DFT, DCT, MDCT, Wavelet, Hilbert, Hartley and Walsh-Hadamard Transforms • Convolution, Cross-Correlation, Auto-Correlation, Conjugate • Windowing, Jaehne/Tone/Triangle signal generation
Digital Filtering	• Finite Impulse Response (FIR), Infinite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

Intel® IPP Domains

Data Processing and Compression

Domain	Functions and Algorithms
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

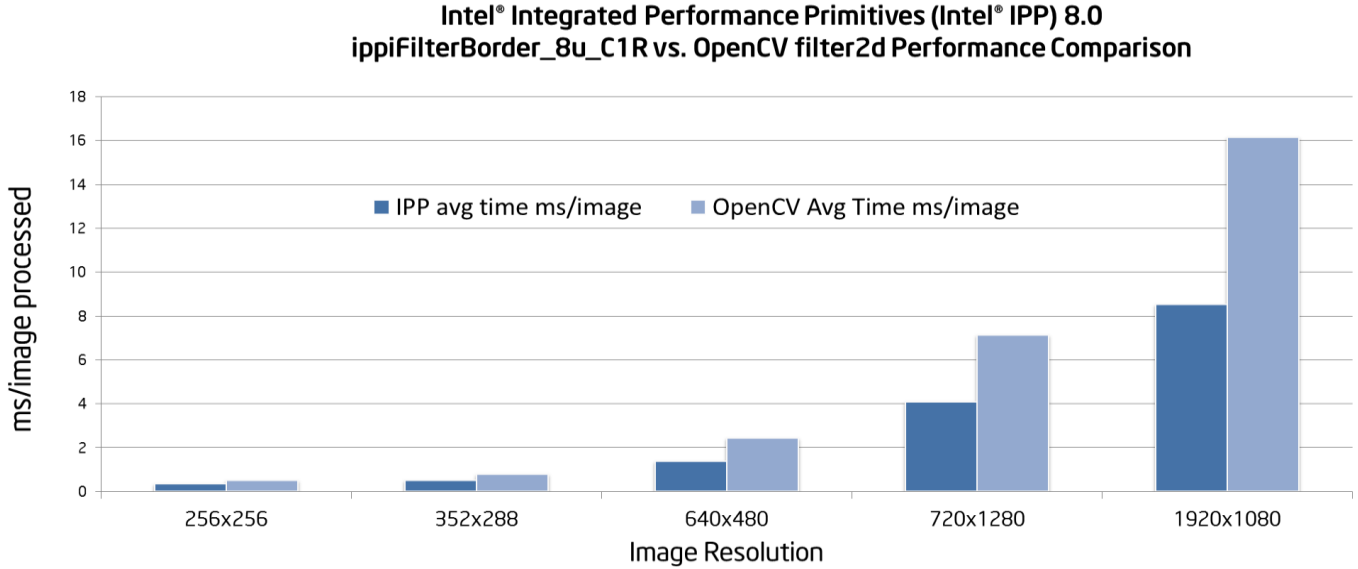
Intel® Integrated Performance Primitives Image Resize Performance vs. OpenCV



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® Integrated Performance Primitives Pixel Convolution Performance vs. OpenCV



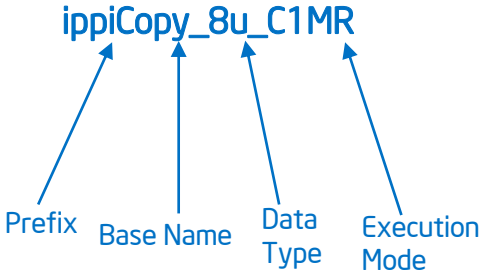
System configuration: Intel® IPP 8.0 (build 83); Precompiled OpenCV 2.4.6. Hardware: Intel® Core™ i7 4770s Processor, 3.10 GHz, 256 KB L2 Cache, 8 GB RAM; Operating System: Windows* 8 64 bit; Visual Studio 2012; Single Threaded; Benchmark source: Intel® Corporation; location: Folsom, CA, 02/07/2013; Notes: , 1 channel (greyscale) image, 7x7 nonseparable kernel.

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



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

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.

ippsCopy_8u(…)

Some examples:

px_ippsCopy_8u(…)

s8_ippsCopy_8u(…)

p8_ippsCopy_8u(…)

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

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

Intel® IPP Linkage

Linkage Models Features	1. Dynamic Linkage	2. Static Linkage with Dispatching	3. Custom Dynamic Linkage	4. Static Linkage without Dispatching
Optimization	All processors	All processors	All processors	One processor
Build	Link to stub libraries	Link to static libraries and stubs	Build separate DLL	Link to processor-specific merged libraries
Calling	Regular names	Regular names	Modified names	Processor-specific names
Total Binary Size	Large	Small	Small	Smallest
Executable Size	Smallest	Small	Smallest	Small
Kernel Mode	No	Yes	No	Yes

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

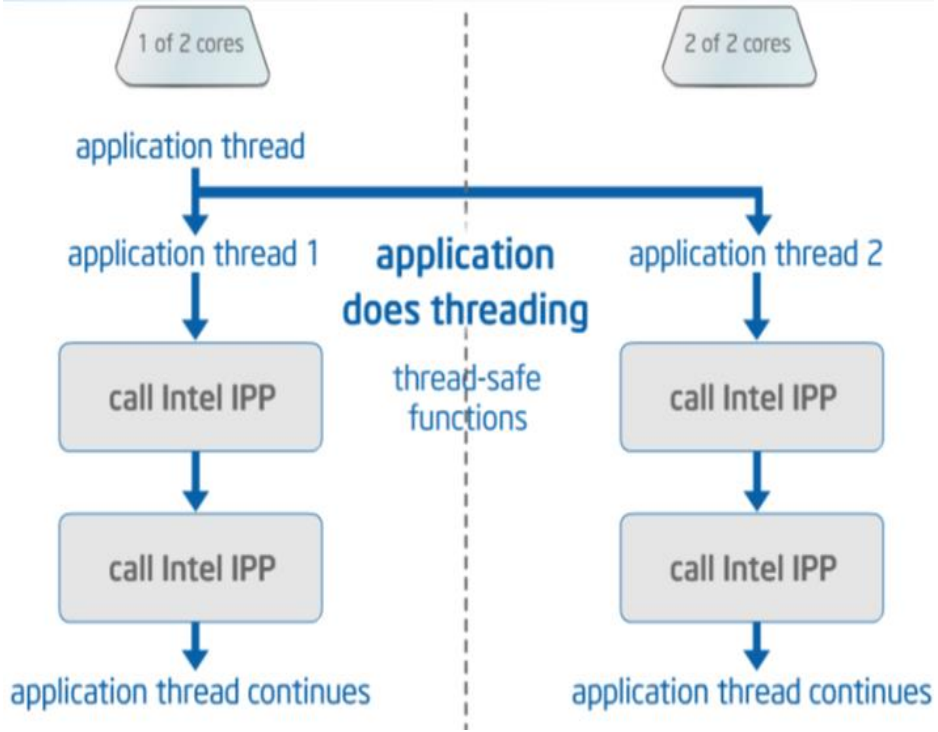
Optimized cryptography functions to support SM2/SM3/SM4 algorithm

Custom dynamic library building tool

New APIs to support external memory allocation

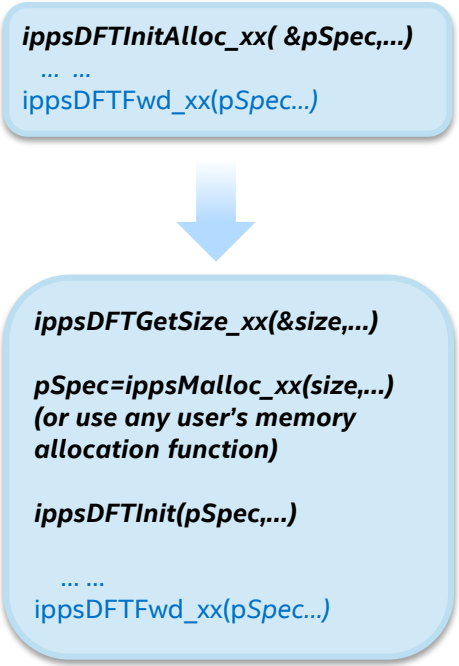
Intel IPP Threading

- 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



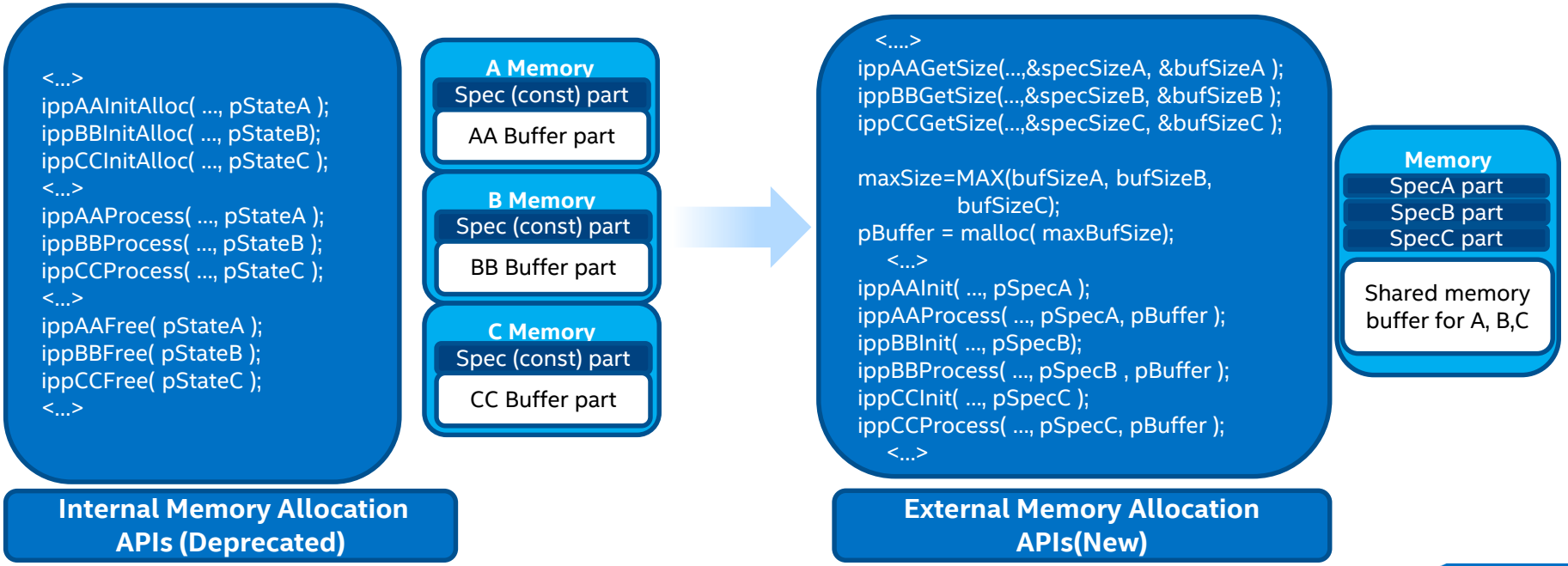
External Memory Allocation

- 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



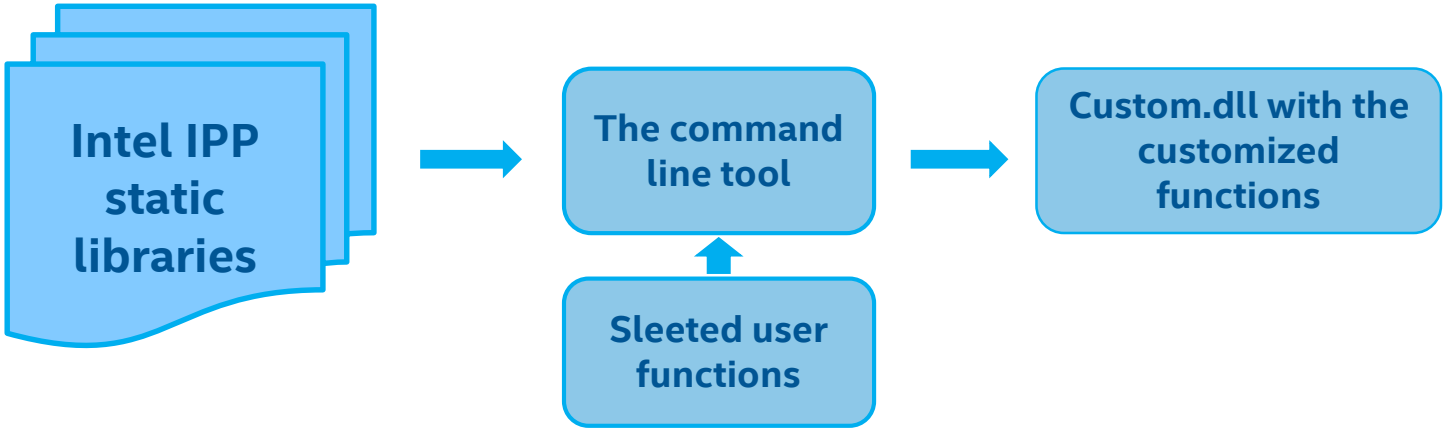
External Memory Allocation(Cont.)

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



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

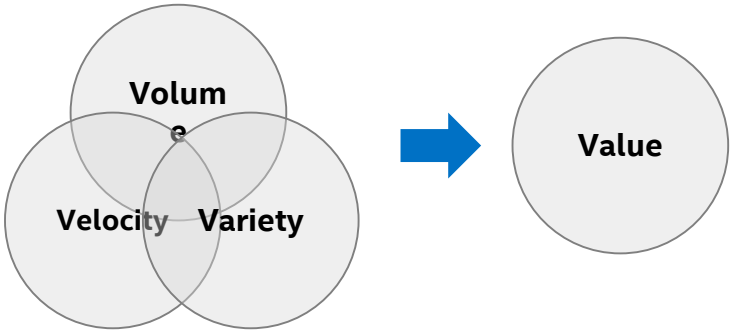
```
dlyLineLen = tapLen -1;//the delay line is used to keep old data.

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 = ouput+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



	Intel® Xeon® processor 64-bit	Intel® Xeon® processor 5100 series	Intel® Xeon® processor 5500 series	Intel® Xeon® processor 5600 series	Intel® Xeon® processor code-named Sandy Bridge EP	Intel® Xeon® processor code-named Ivy Bridge EP	Intel® Xeon® processor code-named Haswell EP	Intel® Xeon Phi™ coprocessor Knights Corner	Intel® Xeon Phi™ processor & coprocessor Knights Landing[1]
Core(s)	1	2	4	6	8	12	18	61	61+
Threads	2	2	8	12	16	24	36	244	244+
SIMD Width	128	128	128	128	256	256	256	512	512

*Product specification for launched and shipped products available on ark.intel.com.
1. Not launched or in planning.

More cores → More Threads → Wider vectors

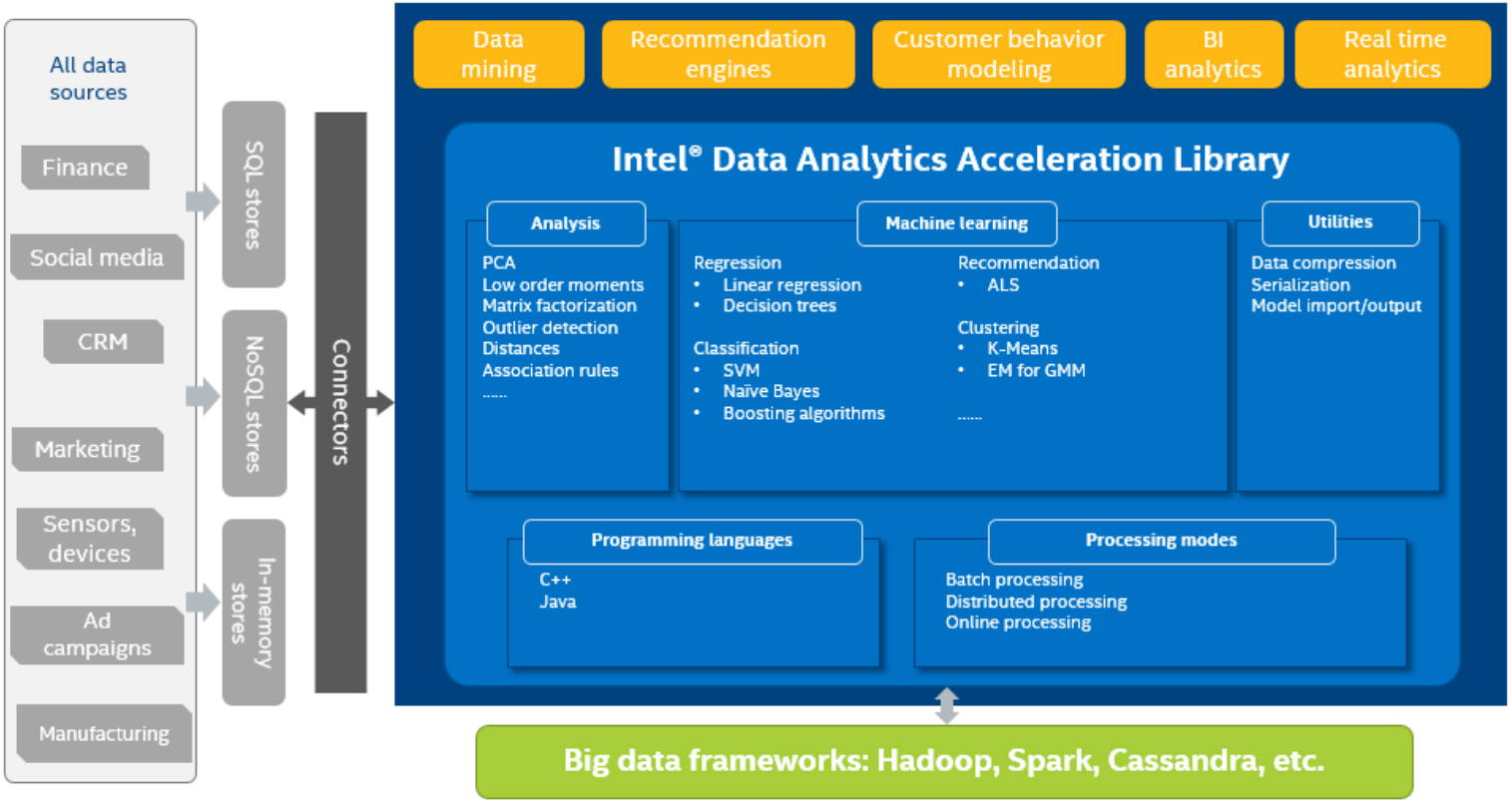
Problem:

- 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?

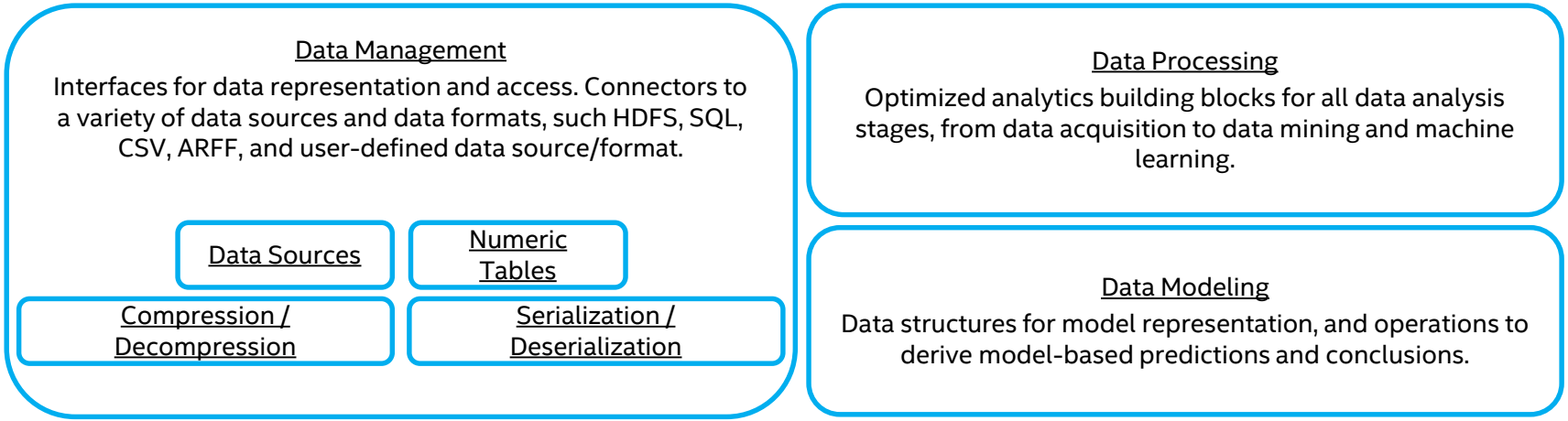


Intel® Data Analytics Acceleration Library 2016

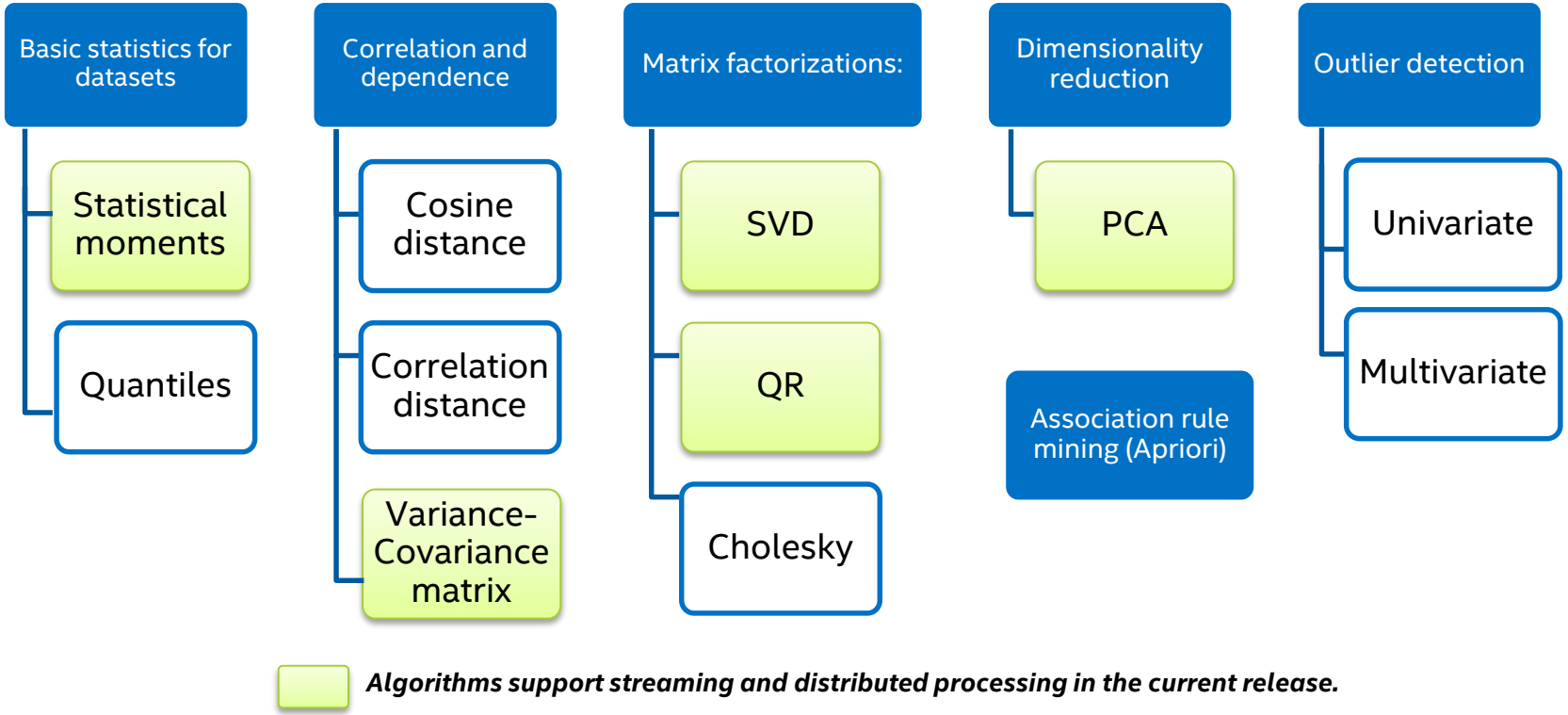
A library optimized for Intel® Architectures that provides building blocks for all data analytics stages, from data preparation to data mining and machine learning.

- 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

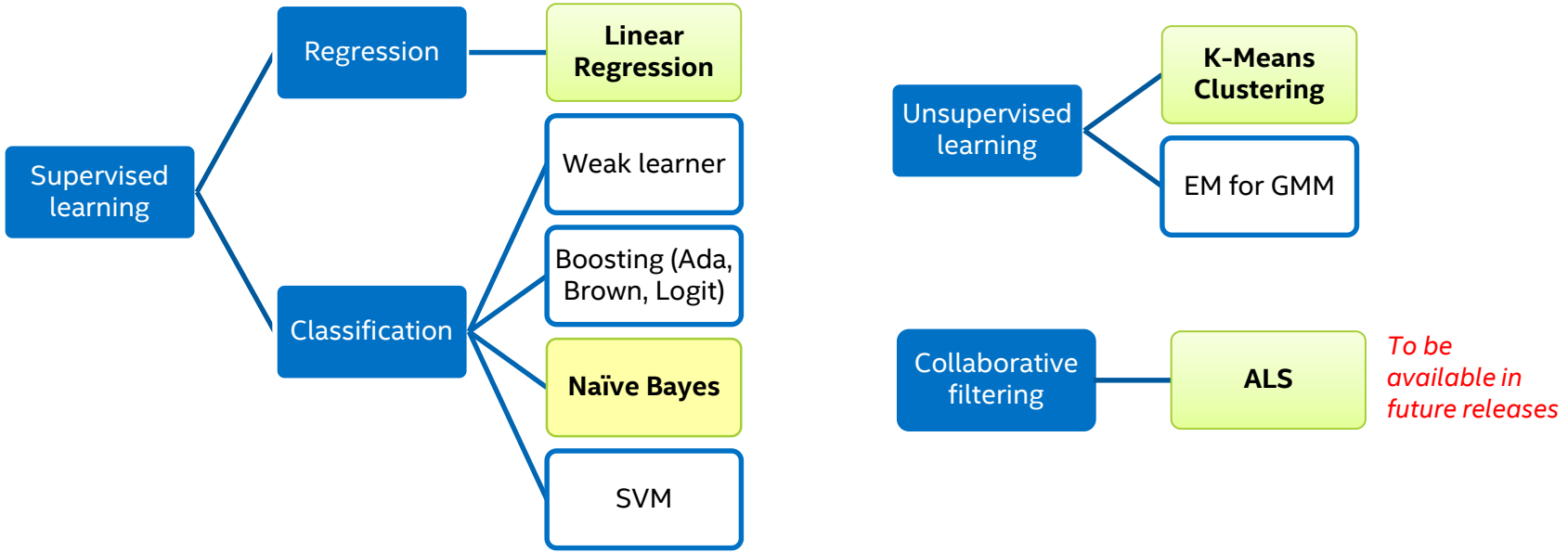


Data Transformation and Analysis (Intel® DAAL)



Algorithms support streaming and distributed processing in the current release.

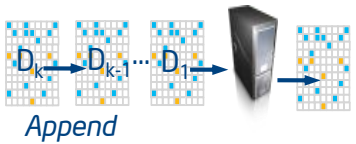
Machine Learning (Intel® DAAL)



Algorithms support streaming and distributed processing in the current release.

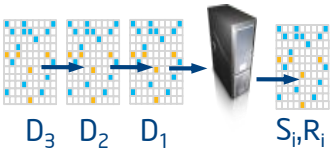
Processing modes

Batch Processing



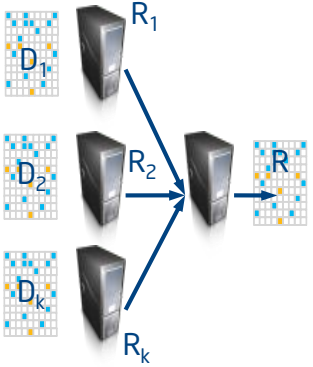
$R = F(D_1,...,D_k)$

Streaming Processing



$S_{i+1} = T(S_i,D_i)$
$R_{i+1} = F(S_{i+1})$

Distributed Processing

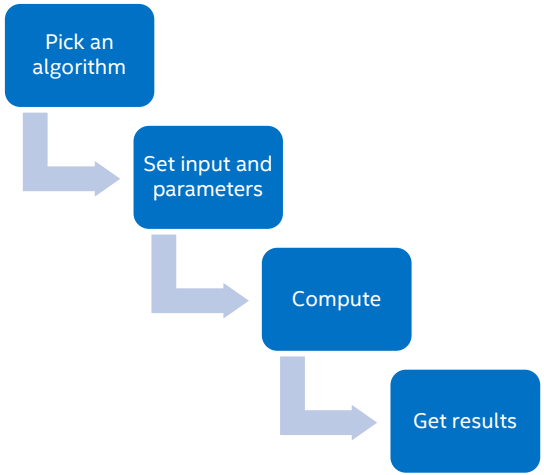


$R = F(R_1,...,R_k)$

Batch Processing

All data fits in memory on a single node.

All DAAL algorithms support batch processing.

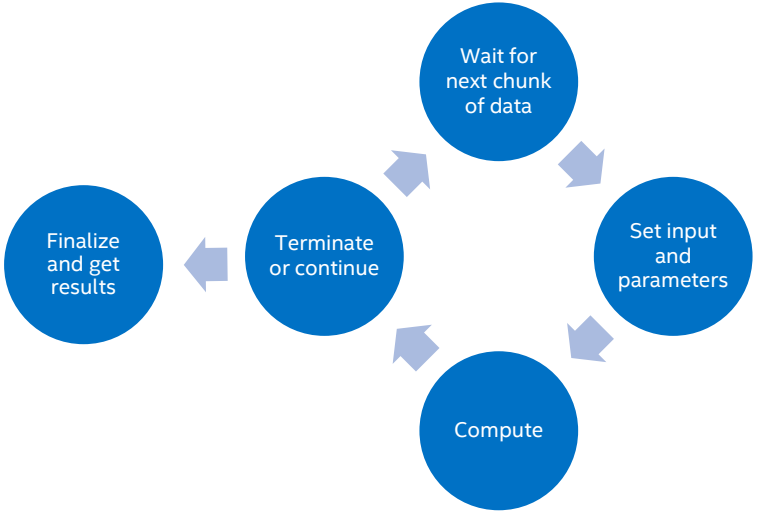


Online Processing

Data being chunked into memory piece by piece.

Work on one piece at a time, combine results at the end.

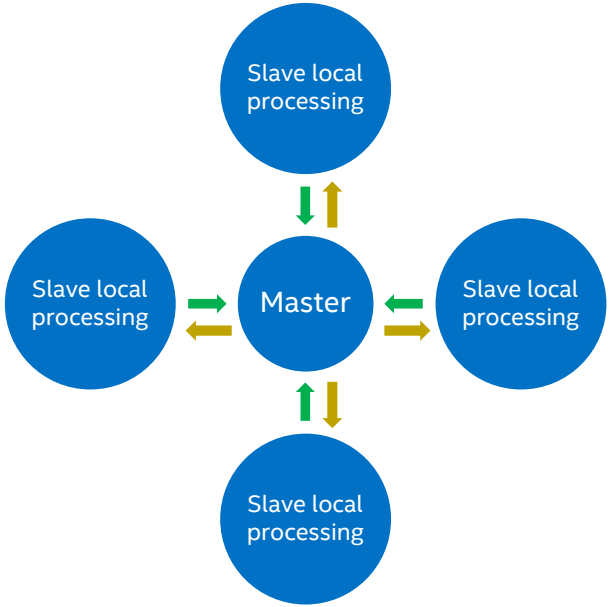
Not all DAAL algorithms support online processing



Distributed Processing

We have a cluster …

Not all DAAL algorithms support distributed processing.



MapReduce

MapReduceMap

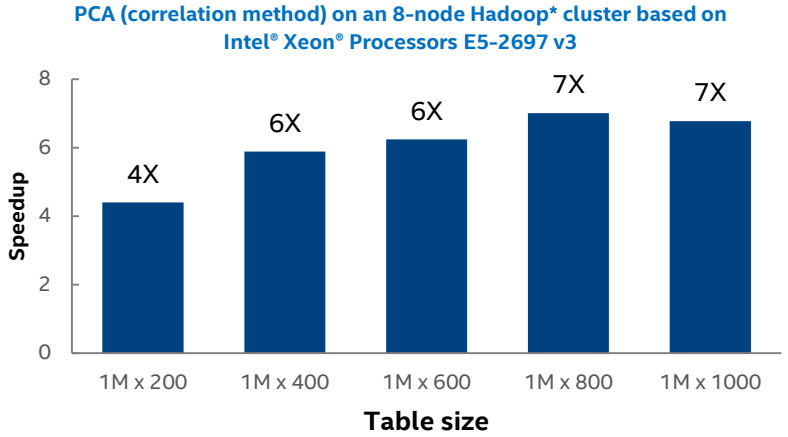
Distributed Processing: Communication

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



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:

- https://software.intel.com/en-us/intel-mkl
- https://software.intel.com/en-us/intel-ipp
- https://software.intel.com/en-us/intel-daal

Community licensing program for Intel performance libraries:

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

Legal Disclaimer & Optimization Notice

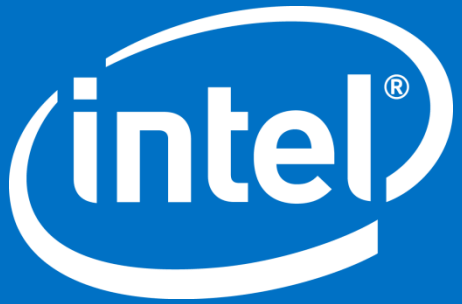
INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING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

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®