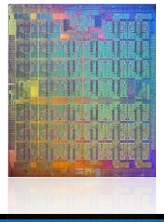
OPTIMIZED INTEL® MATHEMATICAL LIBRARIES FOR HPC AND MACHINE LEARNING

Moscow, March 01, 2018

ruslan.israfilov@intel.com

Intel® Xeon™ Processors Evolution

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 v4 series	Intel® Xeon® Scalable Processor[1]	Intel® Xeon Phi™ x200 Processor (KNL)
Up to Core(s)	1	2	4	6	12	18-22	28	72
Up to Threads	2	2	8	12	24	36-44	56	288
SIMD Width	128	128	128	128	256	256	512	512
Vector ISA	Intel® SSE3	Intel® SSE3	Intel® SSE4- 4.1	Intel® SSE 4.2	Intel® AVX	Intel® AVX2	Intel® AVX-512	Intel® AVX-512

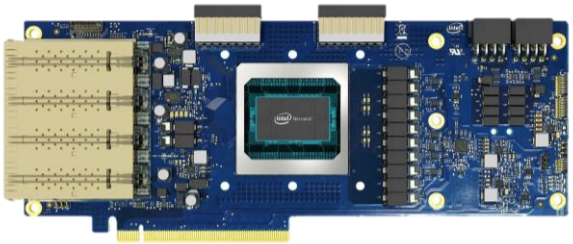
[1] Product specification for launched and shipped products available on ark.intel.com.

Hardware Becomes Heterogeneous

FPGA Based Accelerator

Intel® Programmable Acceleration Card with Intel Arria® 10 GX FPGA

Specialized Hardware for Deep Learning



Intel® Nervana™ Neural Network Processor

https://www.altera.com/products/boards_and_kits/dev-kits/altera/acceleration-card-arria-10-gx.html
https://ai.intel.com/intel-nervana-neural-network-processors-nnp-redefine-ai-silicon/

Optimized Libraries for Intel® Architectures

Optimized Frameworks
to simplify development

Spark MLlib · Caffe · neon · scikit learn · TensorFlow · torch

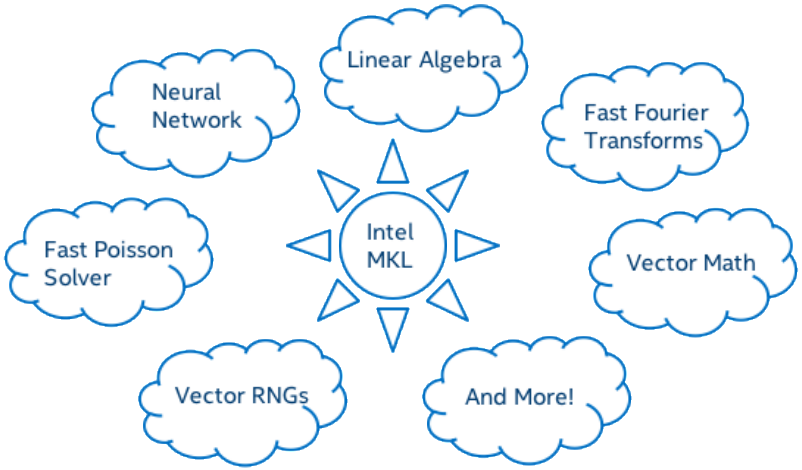
Libraries/Languages
featuring optimized building blocks

Intel® Math Kernel Library (Intel® MKL) · Intel® Data Analytics Acceleration Library (Intel® DAAL) · Intel® Integrated Performance Primitives (Intel® IPP) · Intel® Distribution for Python*

Hardware Technology
portfolio that is broad and cross-compatible

\+ Memory & Storage · Networking

Faster, Scalable Code with Intel® Math Kernel Library



- Features highly optimized, threaded, and vectorized math functions that maximize performance on each processor family
- Utilizes industry-standard C and Fortran APIs for compatibility with popular BLAS, LAPACK, and FFTW functions — no code changes required
- Dispatches optimized code for each processor automatically without the need to branch code

Learn More: software.intel.com/mkl

What's Inside Intel® MKL

Accelerate HPC, Enterprise, IoT & Cloud Applications

Linear Algebra
- BLAS
- LAPACK
- ScaLAPACK
- Sparse BLAS
- Iterative sparse solvers
- PARDISO*
- Cluster Sparse Solver

FFTs
- Multidimensional
- FFTW interfaces
- Cluster FFT

Neural Networks
- Convolution
- Pooling
- Normalization
- ReLU
- Inner Product

Vector RNGs
- Congruential
- Wichmann-Hill
- Mersenne Twister
- Sobol
- Neiderreiter
- Distributions

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

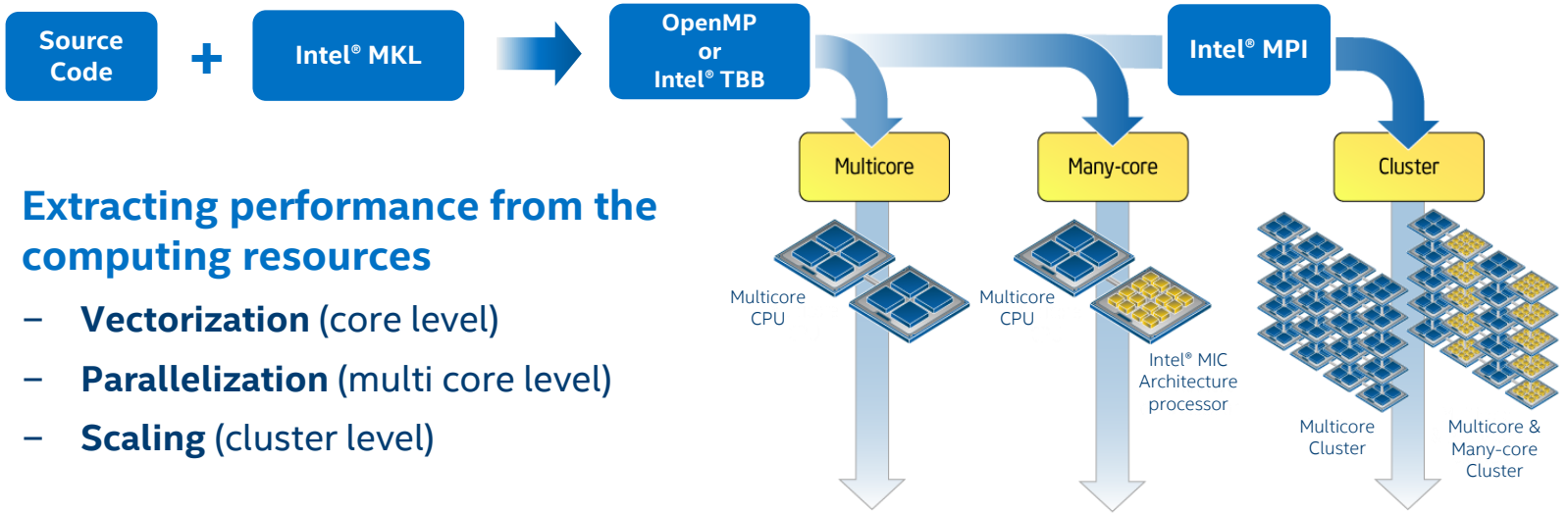
Vector Math
- Trigonometric
- Hyperbolic
- Exponential
- Log
- Power
- Root

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

Intel® Architecture Platforms

Operating System: Windows*, Linux*, MacOS*

Automatic Performance Scaling from the Core, to Multicore, to Many Core and Beyond



Extracting performance from the computing resources

- **Vectorization** (core level)
- **Parallelization** (multi core level)
- **Scaling** (cluster level)

Intel® Integrated Performance Primitives (Intel® IPP)

Image Processing

- Geometry transformations
- Linear and non-linear filtering
- Linear transforms
- Statistics and analysis
- Color models

Computer Vision

- Feature detection
- Objects tracking
- Pyramids functions
- Segmentation, enhancement
- Camera functions
- And more

Signal Processing

- Transforms
- Convolution, Cross-Correlation
- Signal generation
- Digital filtering
- Statistical

Data Compression

- LZSS
- LZ77(ZLIB)
- LZO
- Bzip2

Cryptography

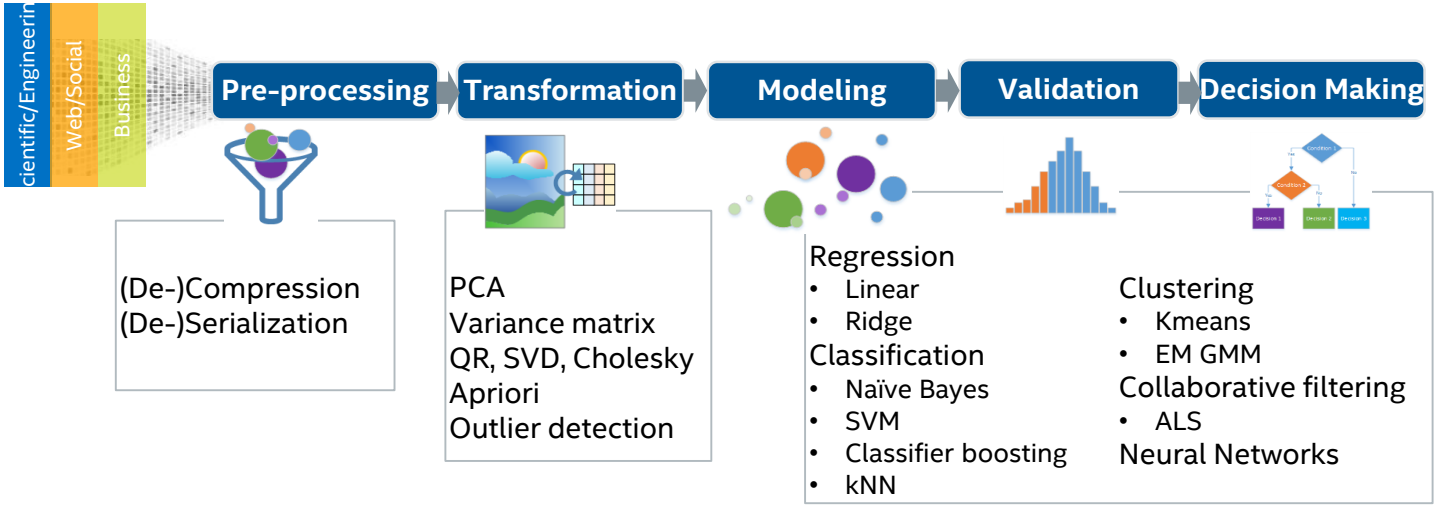
- Symmetric cryptography
- Hash functions
- Data authentication
- Public key

String Processing

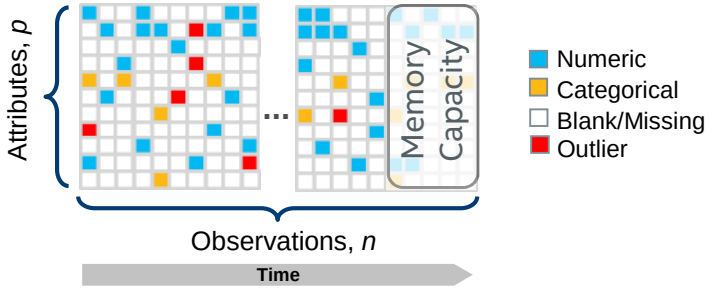
- String Functions: Find, Insert, Remove, Compare, etc.
- Regular expression

Intel® Data Analytics Acceleration Library (Intel® DAAL)

An optimized library that provides building blocks for all data analytics stages, from data preparation to data mining & machine learning



Intel® DAAL for Big Data



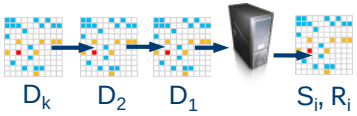
Big Data Attributes

- **Velocity** (data arriving in time)
- **Volume** (huge data not fitting into node memory)
- **Variety** (non-homogeneous/sparse/noisy data)

Batch Computing

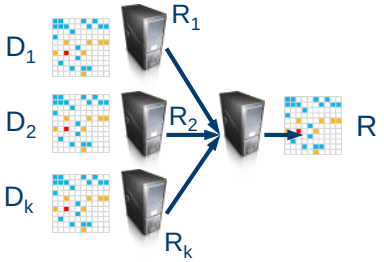
$R = F(D)$

Streaming (online) Computing



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

Distributed Computing



$R = F(R_1, \ldots, R_k)$

Intel® Data Analytics Acceleration Library (Intel® DAAL)

- **C++**, **Java**, **Python** API
- Can be used with many platforms (**Hadoop***, **Spark***, …) but not tied to any of them
- Flexible interface to connect to different data sources (CSV, SQL, …)
- **Windows***, **Linux*** and **OS X***

- IA-32 and Intel64 support
- Supports static and dynamic linking
- Developed by same team as industry-leading Intel® Math Kernel Library
- Commercial and Free Community editions

GitHub

https://github.com/intel/daal

Intel® DAAL Components

Data Management

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

- **Data Sources**
- **Numeric Tables**
- **Compression / Decompression**
- **Serialization / Deserialization**

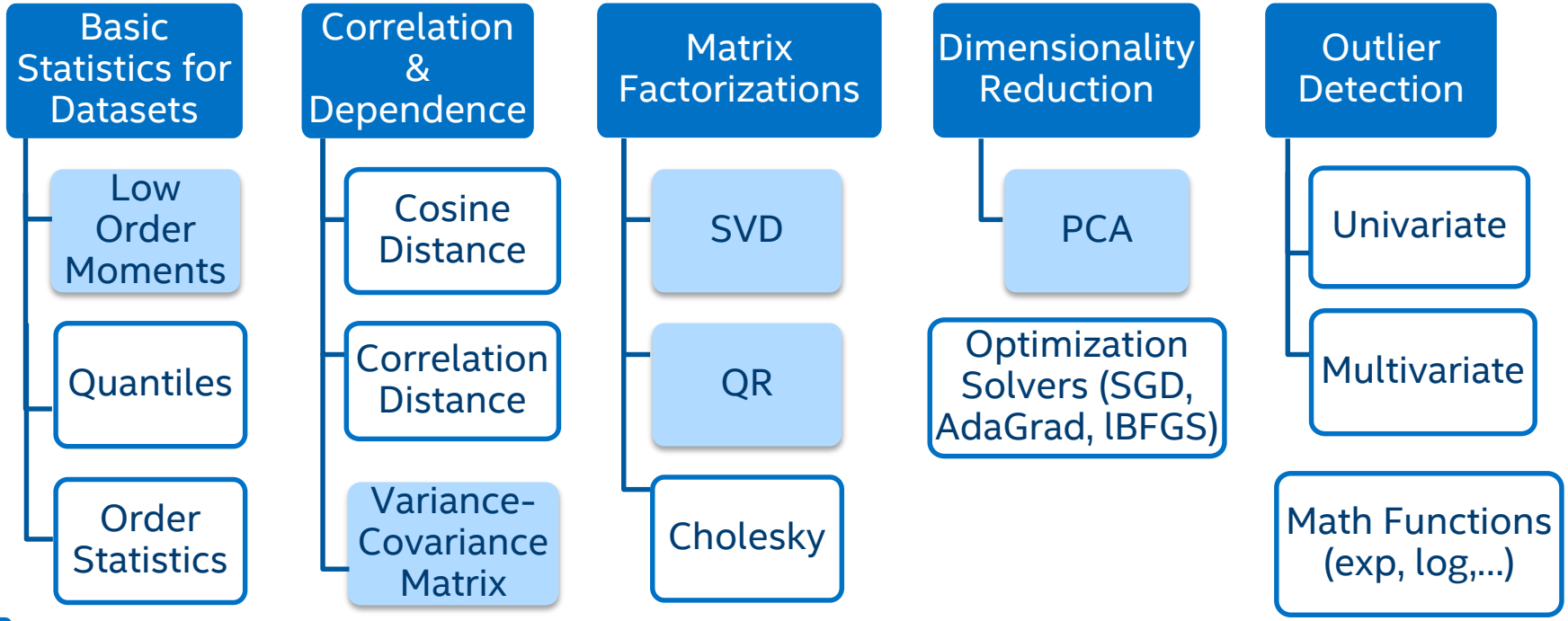
Data Processing Algorithms

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

- **Descriptive Statistics**
- **Statistical Relationships**
- **Supervised Learning**
- **Unsupervised Learning**
- **Recommender Systems**
- **Deep Learning**

Algorithms, Data Transformation & Analysis

Intel® Data Analytics Acceleration Library

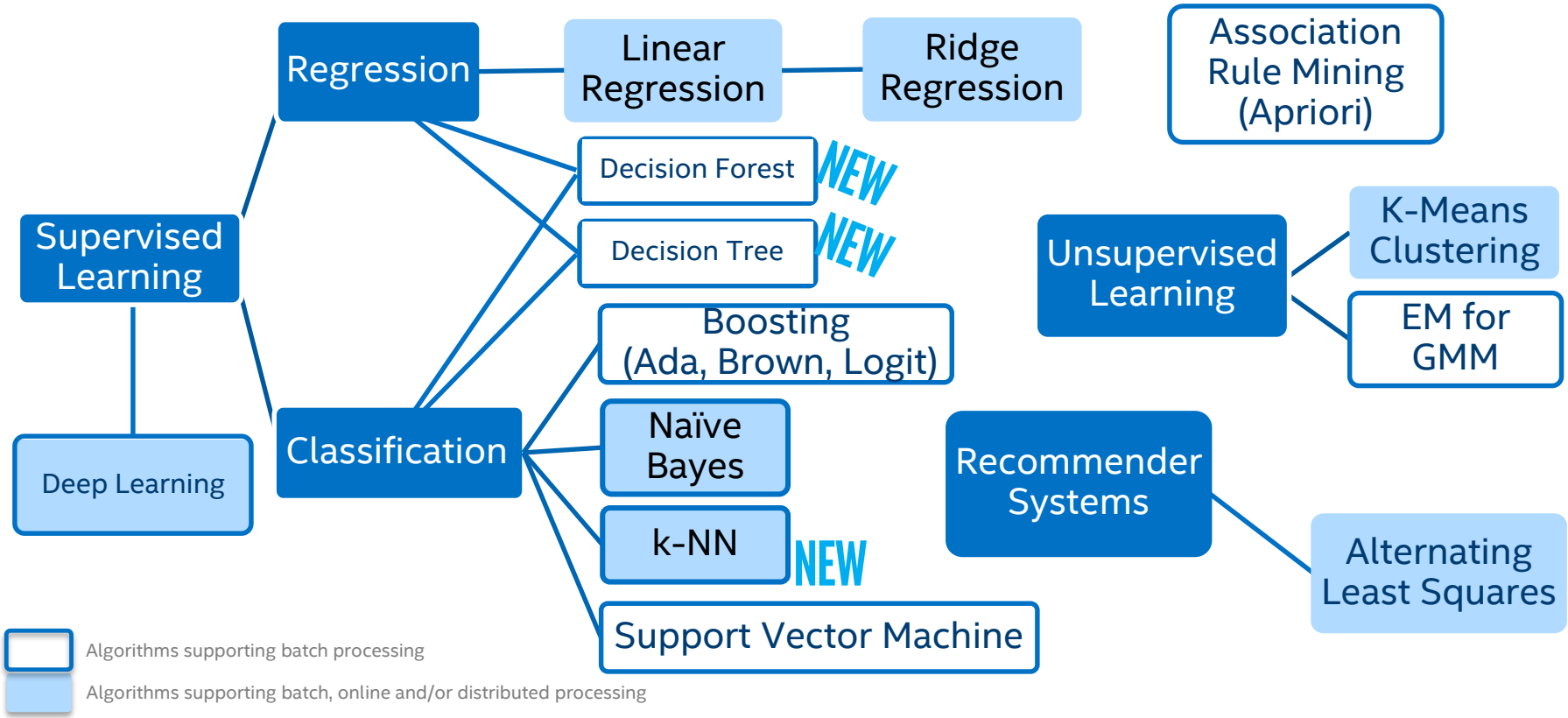


Algorithms supporting batch processing

Algorithms supporting batch, online and/or distributed processing

Algorithms & Machine Learning

Intel® Data Analytics Acceleration Library



Numeric Tables: In-Memory Data Representation

Heterogeneous – AOS (Array of Structures)

- Observations are stored in contiguous memory buffers.

Heterogeneous – SOA (Structure of Arrays)

- Features are stored in contiguous memory buffers.

Homogeneous – Dense matrix

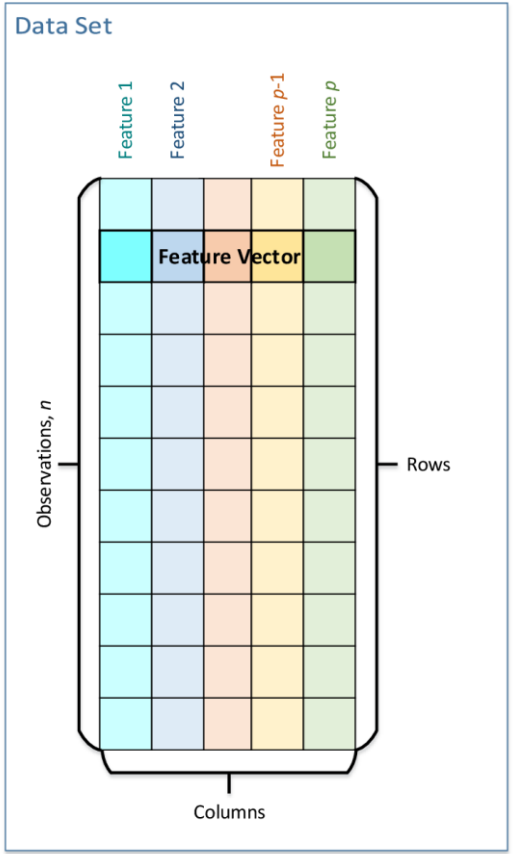
- 2D matrix: *n* rows (observations), *p* columns (features)

Homogeneous – Sparse matrix (CSR)

- Support both 0-based indexing and 1-based indexing.

Tensors

- in-memory numeric multidimensional data

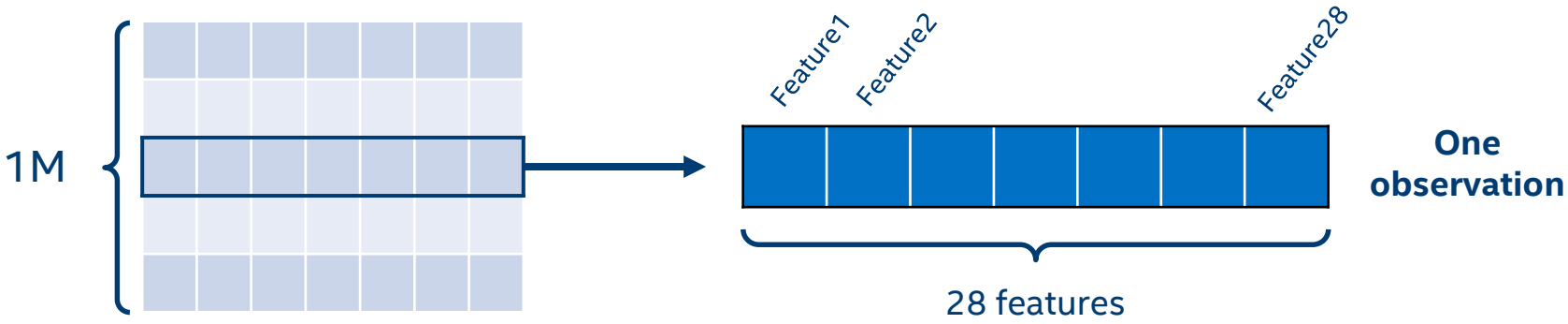


Example: Higgs Boson Classifier

Problem statament: Classify events into "tau tau decay of a Higgs boson" versus "background“ using data with features characterizing events detected by ATLAS

Dataset: HIGGS Data Set (https://archive.ics.uci.edu/ml/datasets/HIGGS), Number of observations = 1M, number of features = 28

Solution: Gradient Boosting Classifier from Intel® DAAL



Code Sample: Reading CSV Data Set

```
const std::string fileName = "higgs_train.csv";

// Define data source object
FileDataSource<CSVFeatureManager> ds(fileName, DataSource::notAllocateNumericTable,
                                                DataSource::doDictionaryFromContext);


// Load data into memory
ds.loadDataBlock();


// Retrieve the numeric table
const NumericTablePtr trainData = ds.getNumericTable();
```

Code Sample: Training Gradient Boosting Classifier

```
// Create an algorithm object to train the gradient boosted trees
gbt::classification::training::Batch<float> algorithm(2);
```

Create algorithm object

```
// Pass a training data set and dependent values to the algorithm
algorithm.input.set(classifier::training::data, trainData);
algorithm.input.set(classifier::training::labels, trainLabels);
```

Pass data to algorithm

```
// Adjust parameters of the algorithm
algorithm.parameter().maxIterations = 50;
algorithm.parameter().maxTreeDepth = 6;
```

Set parameters

```
// Build the gradient boosted trees classification model
algorithm.compute();
```

Train classifier

```
// Retrieve the trained model and write to file
ModelFileWriter writer("./model.bin");
writer.serializeToFile( algorithm.getResult()->get(classifier::training::model) );
```

Save model to file

Code Sample: Prediction Using Trained Model

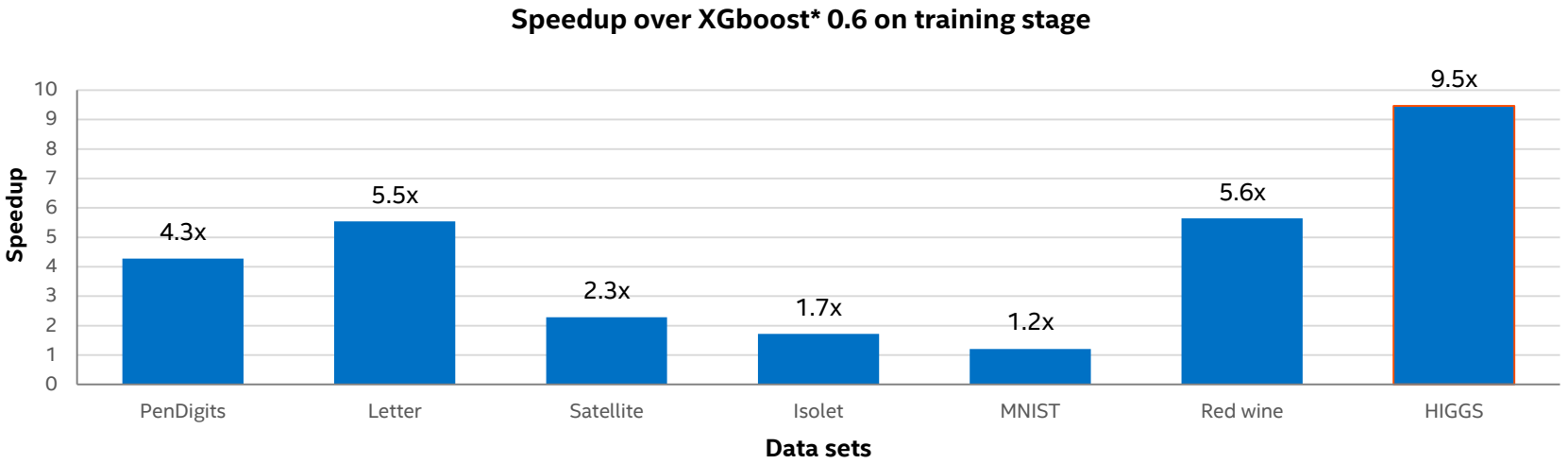
```
// Deserialize model from file
ModelFileReader reader("./model.bin");
classifier::Model trainedModel = reader.deserializeFromFile();

// Create an algorithm object to predict values
gbt::classification::prediction::Batch<float> algorithm(2);

// Pass a testing data set and the trained model to the algorithm
algorithm.input.set(classifier::prediction::data, testData);
algorithm.input.set(classifier::prediction::model, trainedModel);

// Predict values
algorithm.compute();

// Retrieve the algorithm results
classifier::prediction::ResultPtr predictionResult = algorithm.getResult();

// Retrieve predicted labels
NumericTablePtr labels = predictionResult->get(classifier::prediction::prediction);
```

Read model from file

Create algorithm object

Pass model and test data

Intel® DAAL 2018 vs XGboost* (classification)



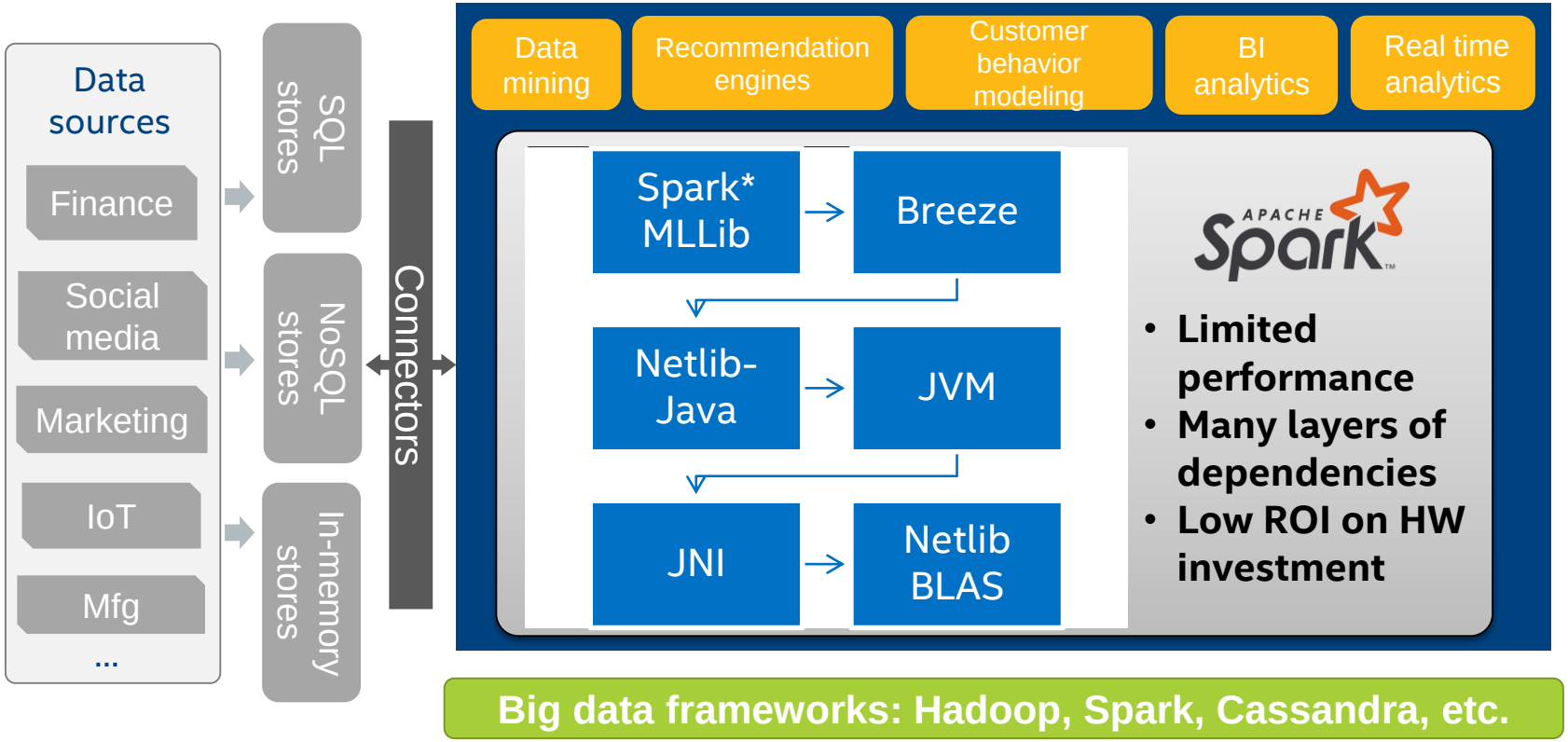
Configuration: Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz, 2x24, 63GB RAM, Intel® DAAL 2018, Xgboost* 0.6.
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific 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. For more complete information visit www.intel.com/benchmarks. Source: Intel Corporation - performance measured in Intel labs by Intel employees. 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® DAAL 2018 vs XGboost* (classification)

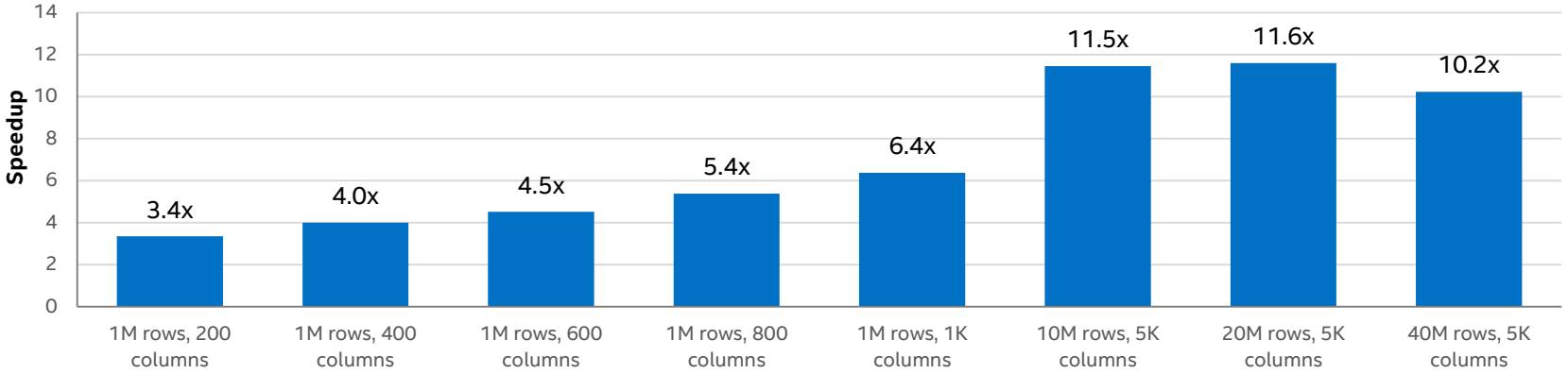
Datasets used in the analysis:

Dataset	Number of feature vectors	Number of features	Number of classes	Number of categorical features	Description
PenDigits	7494	16	10	0	Pen-Based Recognition of Handwritten Digits
Letter	16000	16	26	0	Letter Image Recognition Data
Satellite	4435	36	7	0	Satellite image recognition
Isolet	6238	617	26	0	Predict which letter-name was spoken
MNIST	60000	784	10	0	Handwritten digits recognition
Red wine	1119	11	10	0	Wine recognition data
White wine	3429	11	10	0	Wine recognition data
HIGGS	1M	28	2	0	Distinguish signal proccess: HIGGS/non HIGGS We used first 1M rows for training

Limitations of Existing Big Data Solutions



Intel® DAAL-PCA Performance Boosts Using Intel® DAAL vs. Spark* MLlib on an Eight-node Cluster



Configuration Info - Versions: Intel® Data Analytics Acceleration Library 2017, Spark 1.2; 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 .

Intel® MKL Resources

Intel® MKL website, forum, benchmarks

- https://software.intel.com/en-us/intel-mkl
- https://software.intel.com/en-us/forums/intel-math-kernel-library
- https://software.intel.com/en-us/intel-mkl/benchmarks#

Intel® MKL link line advisor

- http://software.intel.com/en-us/articles/intel-mkl-link-line-advisor/

Intel® IA optimized frameworks

- https://github.com/intel/caffe
- https://github.com/intel/theano

Intel® DAAL Resources

Intel® Machine Learning

- http://www.intel.com/content/www/us/en/analytics/machine-learning/overview.html

Intel® DAAL website

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

Intel® DAAL forum

- https://software.intel.com/en-us/forums/intel-data-analytics-acceleration-library

Intel® DAAL blogs

- https://software.intel.com/en-us/blogs/daal
- https://01.org/daal/blogs/kmoffat/2016/intel%C2%AE-daal-and-intel%C2%AE-mkl-%E2%80%93-complementary-high-performance-machine-learning

Legal Disclaimer and 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. For more complete information visit www.intel.com/benchmarks.

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®
experience
what's inside™