ACCELERATING MACHINE LEARNING AND DEEP LEARNING ON INTEL® IA

Andrey Nikolaev

Principal Engineer, Architect for Intel® Data Analytics Acceleration Library and components of Intel® Math Kernel Library

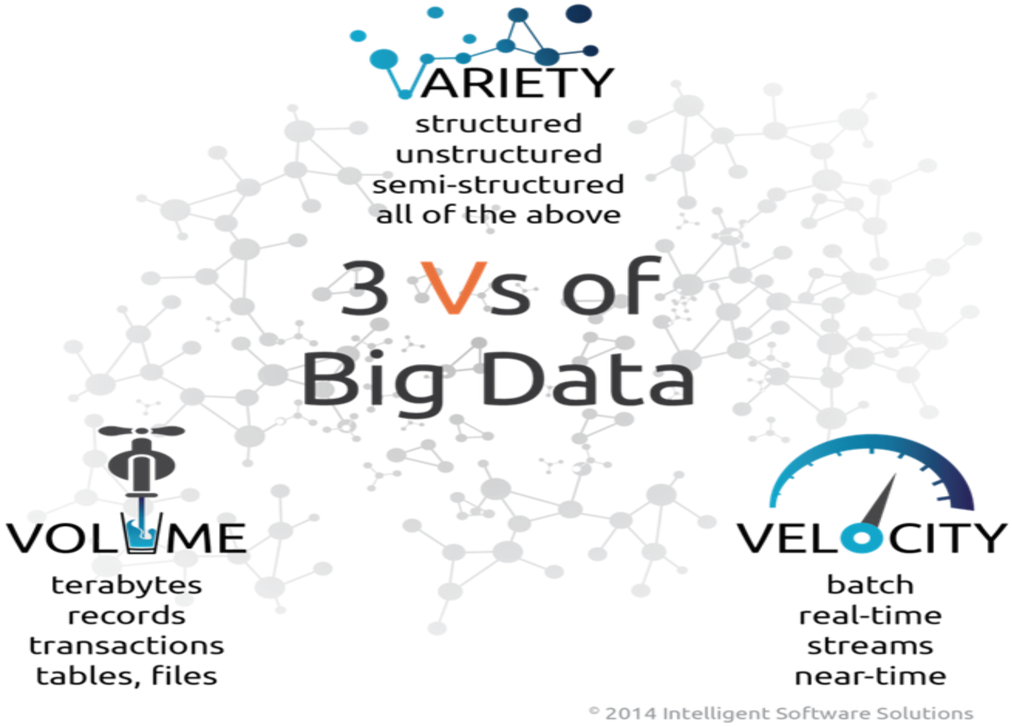
December 19, 2016

Agenda

- Introduction
- Accelerating Machine Learning on Intel® IA
- Accelerating Deep Learning on Intel® IA
- Conclusions and resources

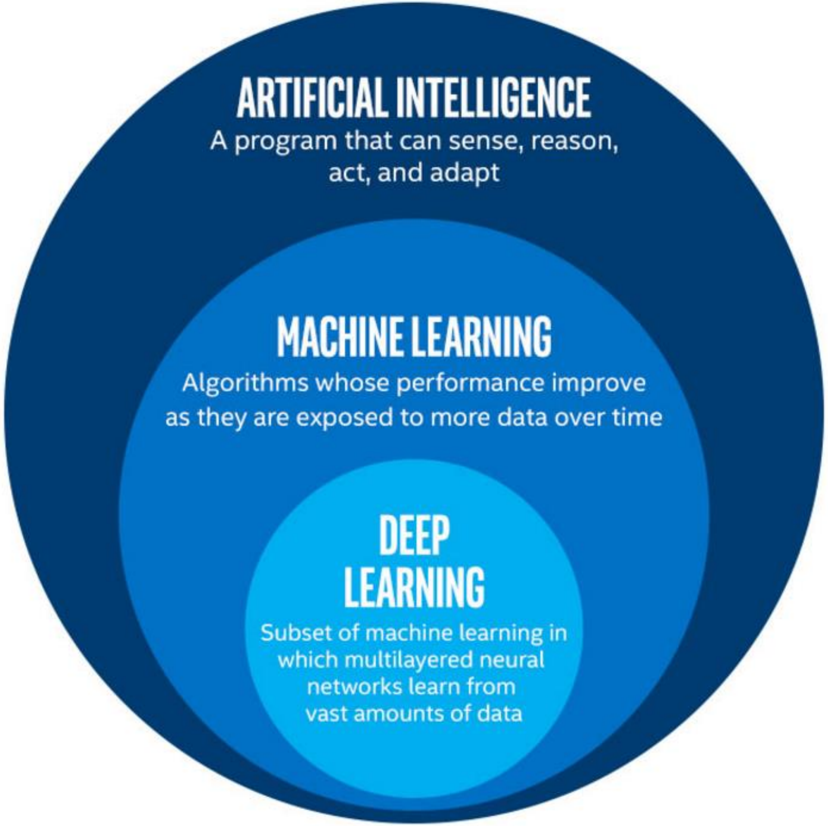
INTRODUCTION

Big Data Vs



https://www.issinc.com/three-vs-big-data-get-fourth-value/

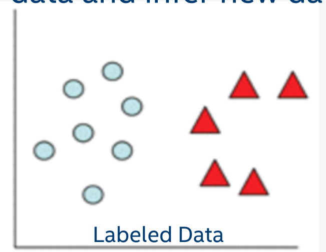
Artificial Intelligence, Machine Learning, Deep Learning



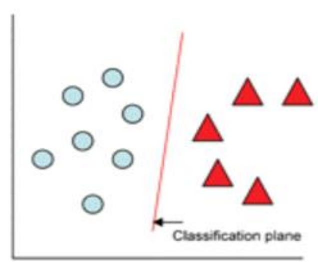
Core methods of Machine Learning

Supervised

Teach desired behavior with labeled data and infer new data



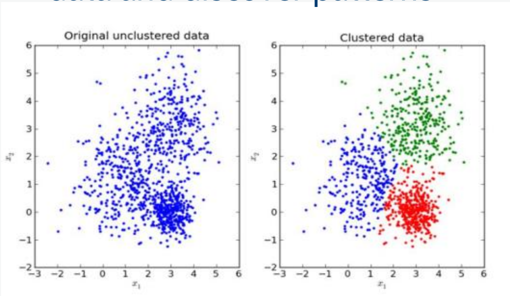
Labeled Data



Classified Data

Unsupervised

Make inferences with unlabeled data and discover patterns



Unlabeled Data Clustered Data

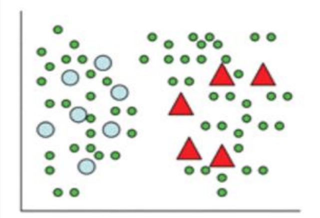
Reinforcement

Act in a environment to maximize reward

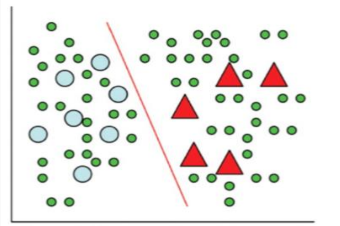
Build autonomous agents that learn

Semi-supervised

A combination of supervised and unsupervised learning



Labeled and Unlabeled Data



Classified Data

http://www.frankichamaki.com/data-driven-market-segmentation-more-effective-marketing-to-segments-using-ai/

Rich set of methodologies to work with Big Data on advanced HW

Machine Learning: Your Path to Deeper Insight

Driving increasing innovation and competitive advantage across industries

intel strategy provides the foundation for success using AI

Solutions
for reference across industries

Tools/Platforms
to accelerate deployment

TAP
Trusted Analytics Platform

Intel® Deep Learning SDK for Training & Deployment

nervana

Optimized Frameworks
to simplify development

Spark, Caffe, theano, torch, TensorFlow, neon

Libraries/Languages
featuring optimized building blocks

Intel® Math Kernel Library (Intel® MKL & MKL-DNN)

Intel® Data Analytics Acceleration Library (Intel® DAAL)

Intel® Distribution for Python*

Hardware Technology
portfolio that is broad and cross-compatible

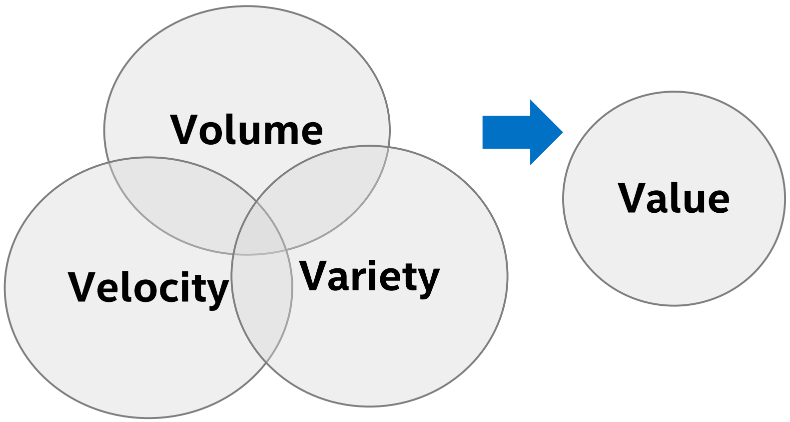
XEON PHI, XEON, Arria 10, CORE i3, CORE i5, CORE i7, ATOM

Datacenter ⟷ *Endpoint*

+Network
+Memory
+Storage

ACCELERATING MACHINE LEARNING ON INTEL® IA

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

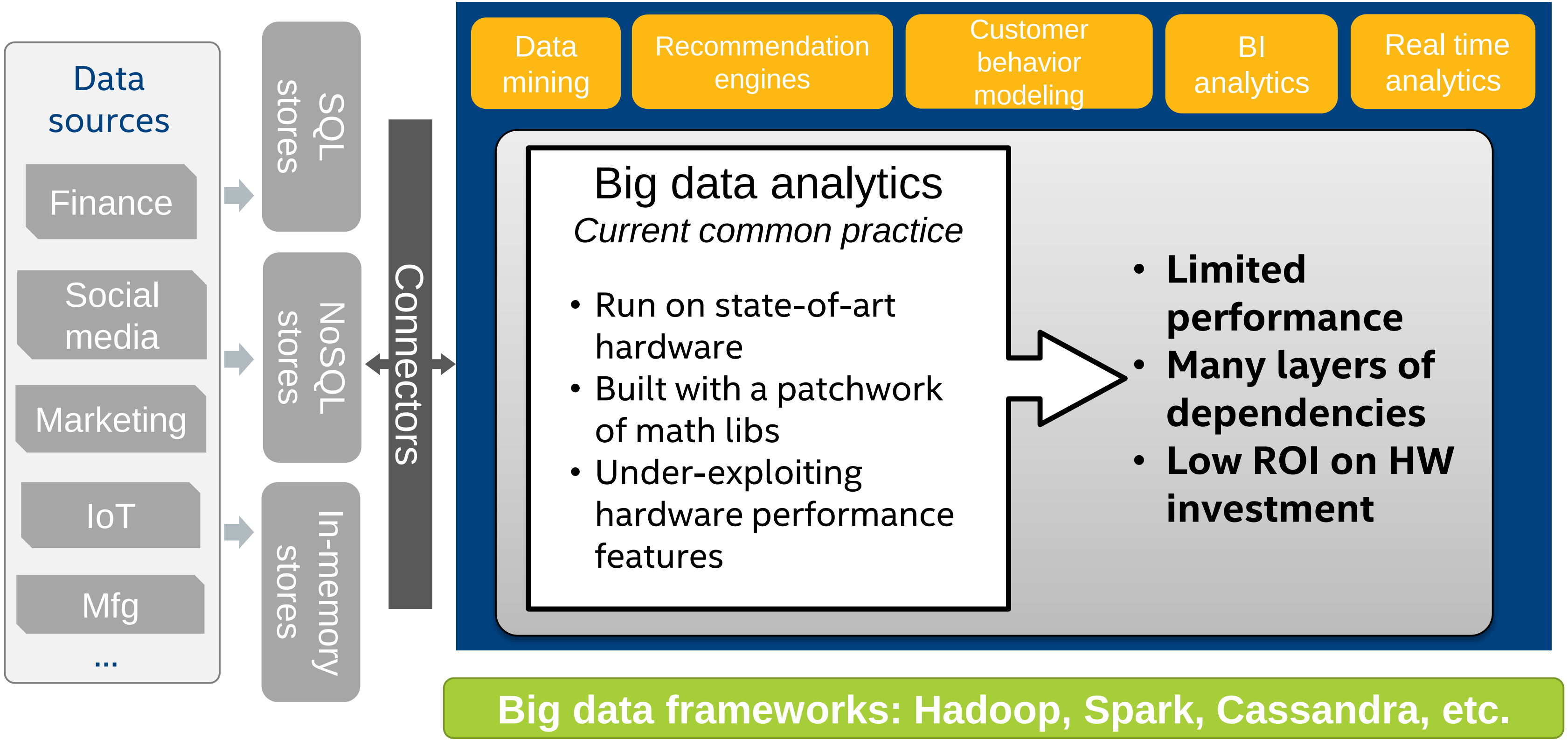
Problem:

- Big data needs high performance computing
- Many big data applications are not optimized for underlying hardware

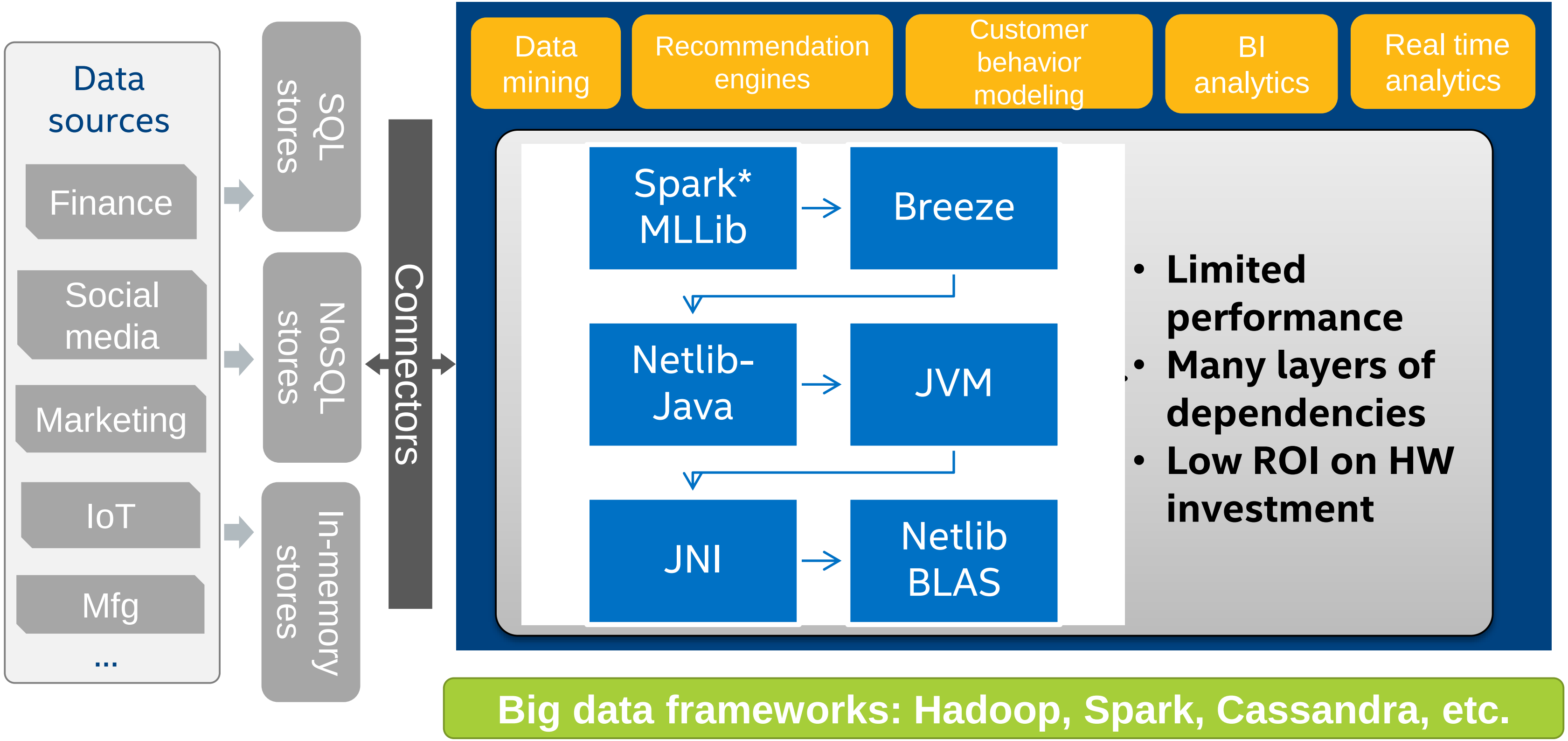
Solution:

- A performance library of building blocks to easily integrate into big data analytics workflows

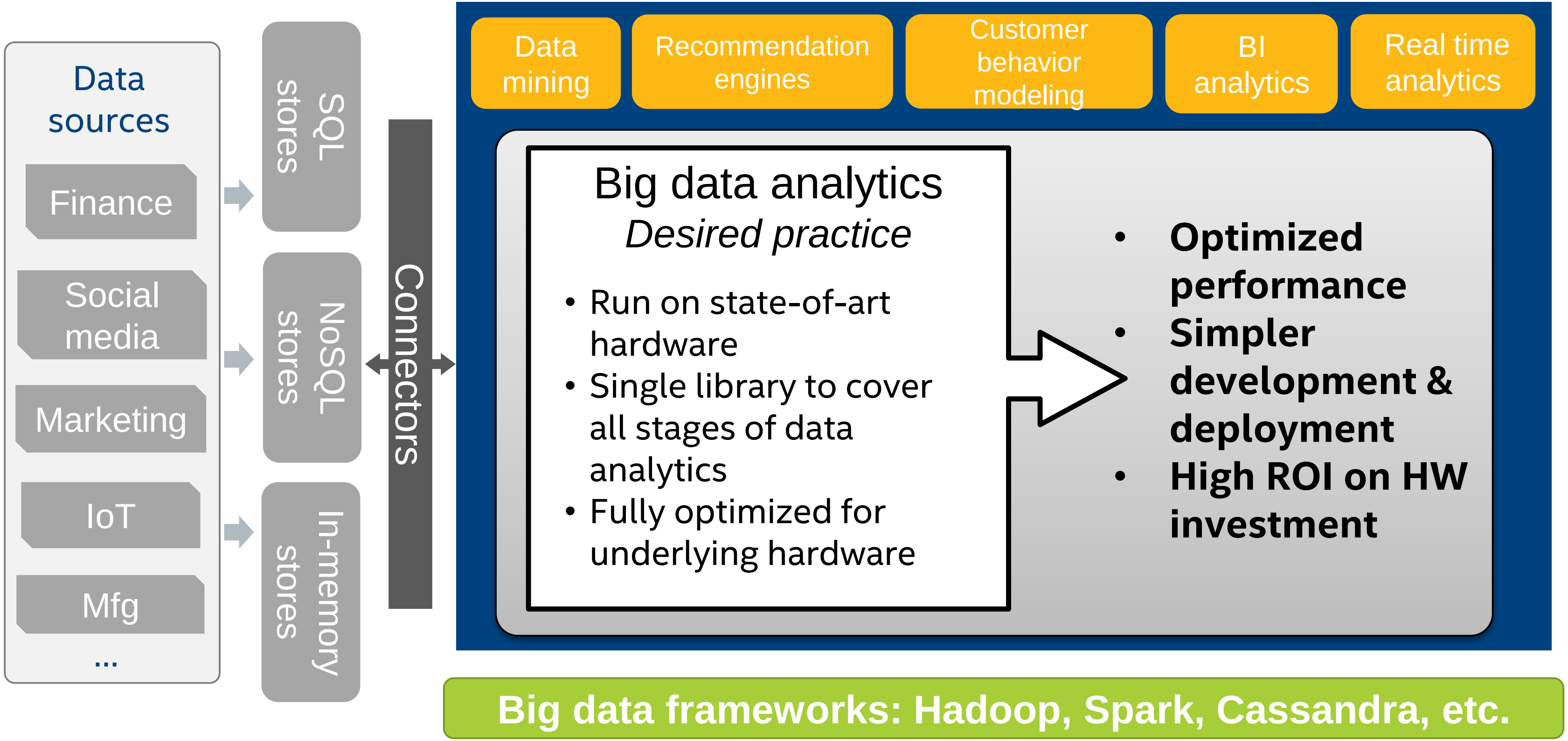
Problem Statement



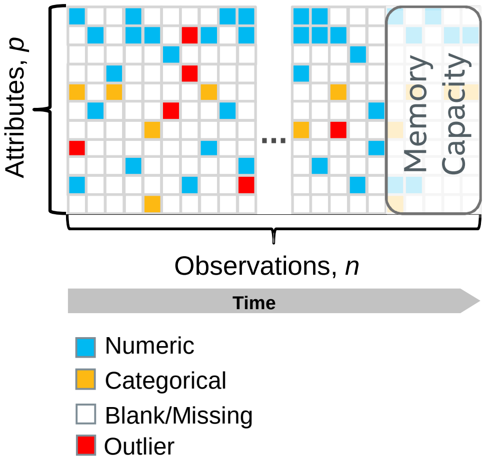
Problem Statement



Desired Solution

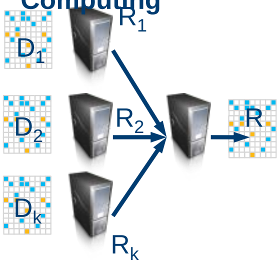


Computational Aspects of Big Data



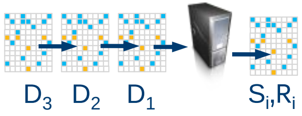
Big Data Attributes	Computational Solution
Distributed across different nodes/devices	•Distributed computing, e.g. comm-avoiding algorithms
Huge data size not fitting into node/device memory	•Distributed computing •Streaming algorithms
Data coming in time	•Data buffering •Streaming algorithms
Non-homogeneous data	•Categorical→Numeric (counters, histograms, etc) •Homogeneous numeric data kernels • Conversions, Indexing, Repacking
Sparse/Missing/Noisy data	•Sparse data algorithms •Recovery methods (bootstrapping, outlier correction)

Distributed Computing



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

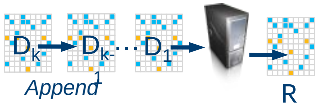
Streaming Computing



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

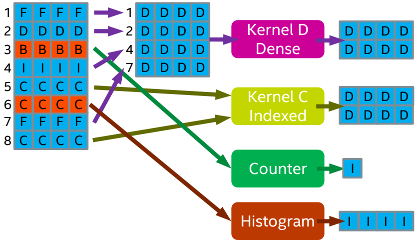
$R_{i+1} = F(S_{i+1})$

Offline Computing

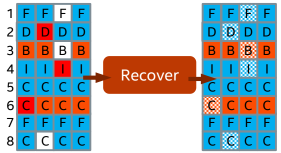


$R = F(D_1,\ldots,D_k)$

Converts, Indexing, Repacking



Data Recovery



Intel® Data Analytics Acceleration Library (Intel® DAAL)

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

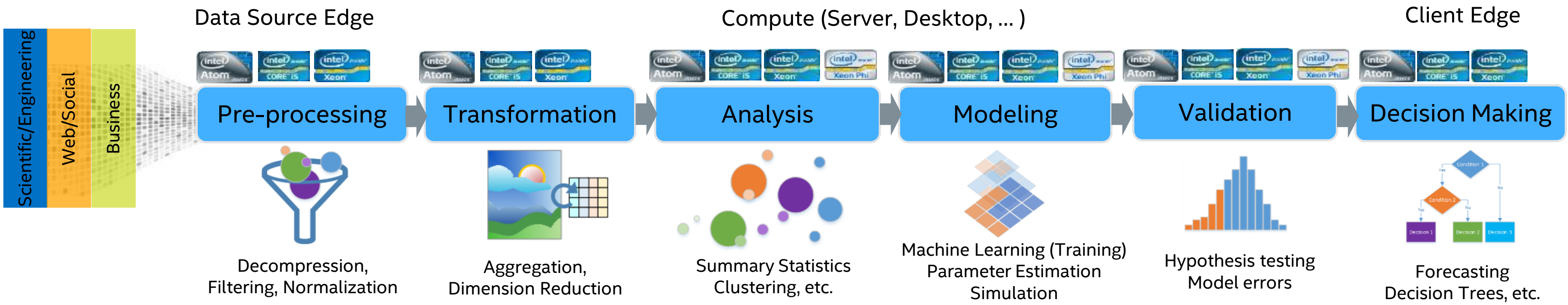
- C++, Java, and Python APIs
- Can be used with many platforms (Hadoop*, Spark*, R*, Matlab*, …) but not tied to any of them
- Flexible interface to connect to different data sources (CSV, SQL, HDFS, KDB)
- Windows*, Linux*, OS X*
- IA-32 & Intel64, static an dynamic linking
- Product launch: 2015
- Open source, Free Community, and Commercial premium supported options
- Also Included in Parallel Studio XE suites

Intel® DAAL: High Level View

Data is different, data analytics pipeline is the same

Data transfer between devices is costly, protocols are different

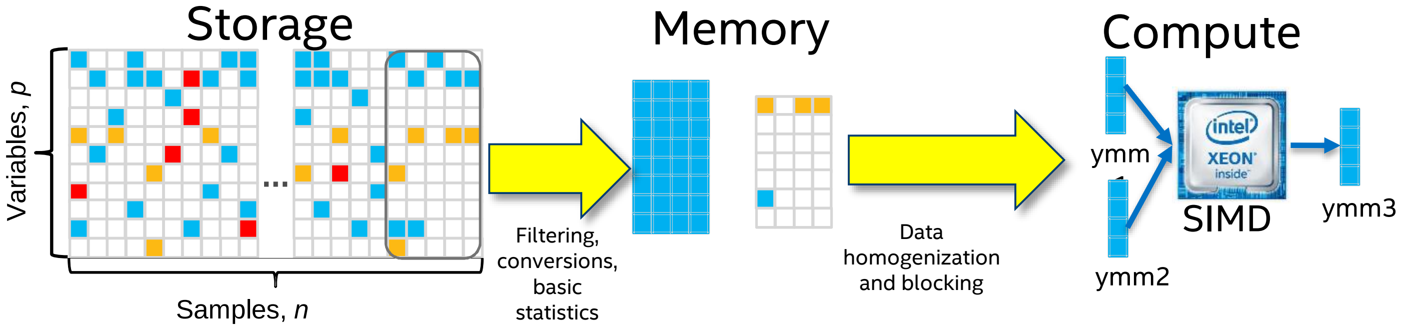
- Need data analysis proximity to Data Source
- Need data analysis proximity to Client
- Data Source device ≠ Client device
- Requires abstraction from communication protocols



Intel® DAAL: High Level View

Optimizing storage ≠ optimizing compute

- Storage: efficient non-homogeneous data encoding for smaller footprint and faster retrieval
- Compute: efficient memory layout, homogeneous data, contiguous access
- Easier manageable for traditional HPC, much more challenging for Big Data



DAAL DataSource | DAAL NumericTable | DAAL Algorithm

Intel® DAAL: High Level View

Intel® DAAL has multiple programming language bindings

C++ – ultimate performance for real-time analytics with Intel® DAAL

Java*/Scala* – easy integration with Big Data platforms (Hadoop*, Spark*, KDB*)

Python* – advanced analytics for data scientist

Why Intel® DAAL?

Automatic performance scaling

- Scale-up: from core to multicore to multi-socket
- Scale-out: from in-memory analysis to clusters to cloud

Algorithms and Data Connectors

- Widely applicable to most ML workloads
- Connectors to popular data sources

Leverages decades of work in IA optimization

- By the same team behind Intel® Math Kernel Library

Who should use Intel® DAAL?

Software developers

- Need optimized ML algorithms in their apps
- No resources/time/expertise to manually optimize themselves

Data Scientists

- Build and executes math models for domain specific knowledge discovery
- Need to speed up the performance critical parts of their models

Data Analytics ISVs

- Want competitive advantages by making their solutions run faster on IA

Big Data Integrators

- Want to beef up their product portfolio by providing performance-enhanced alternatives to popular open-source analytics tools

Intel® DAAL Components

Data Management

Interfaces for data representation and access. Connectors to a variety of data sources and data formats, such HDFS, SQL, CSV, KDB, 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

Data Modeling Algorithms

Data structures for model representation, and operations to derive model-based predictions and conclusions

Intel® DAAL computing modes

Batch

- Data fits into memory of a single node

Online

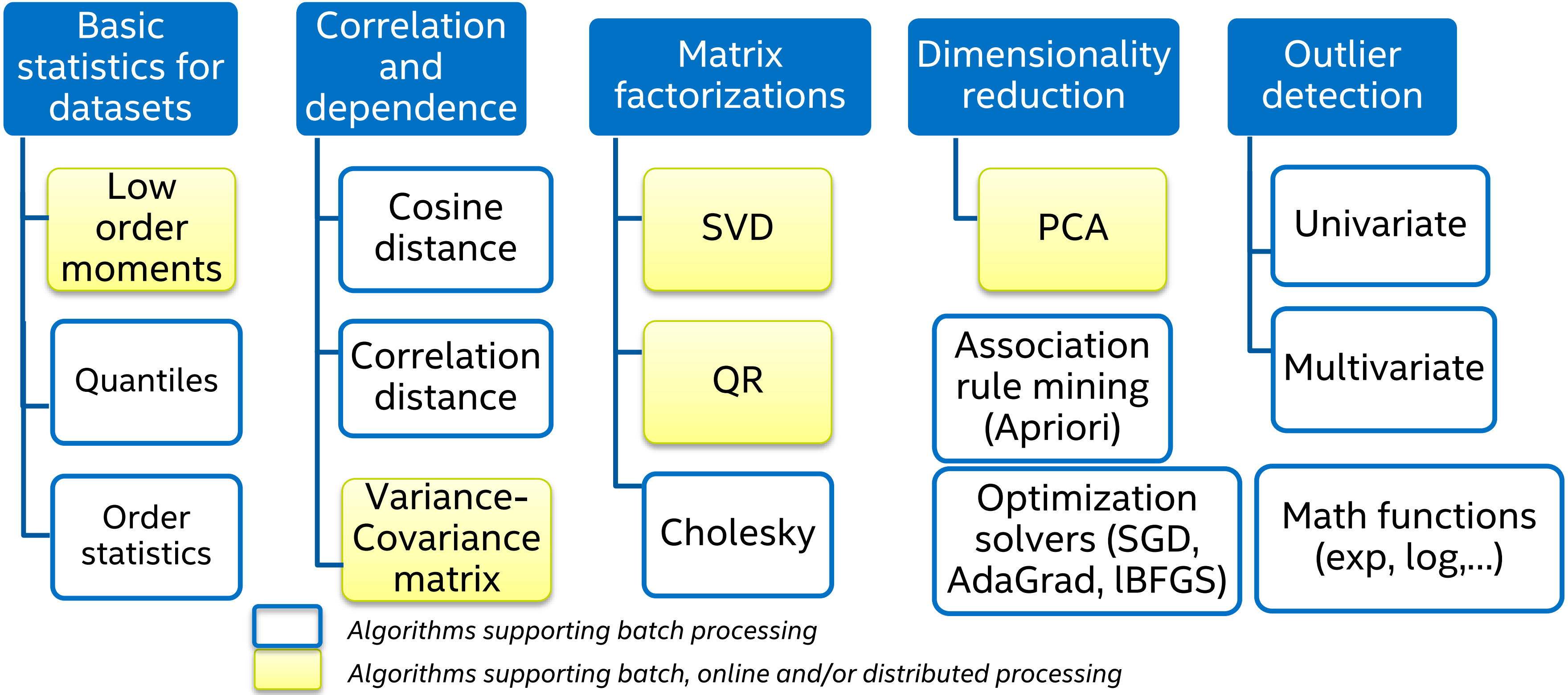
- Data arrives by blocks
- Update partial model using the latest block

Distributed

- Data is split across nodes
- Communication technology agnostic

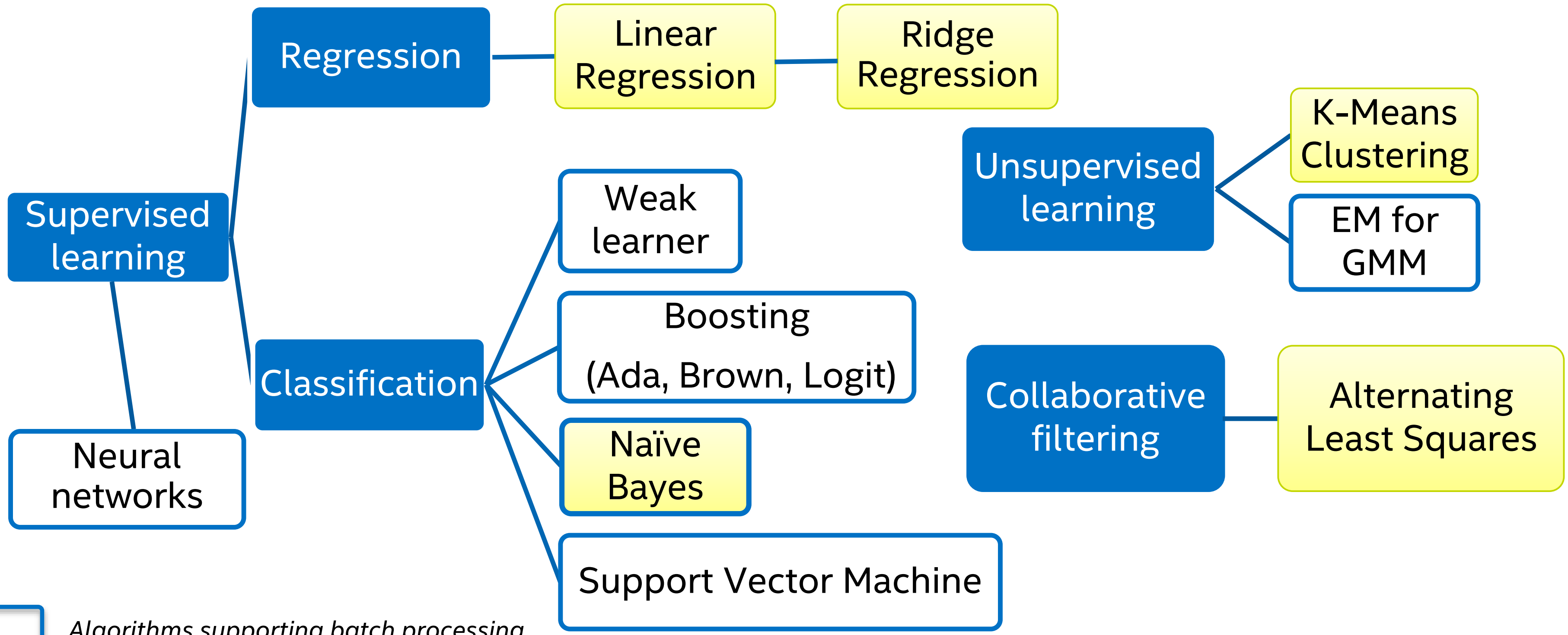
Intel® DAAL Algorithms

Data Transformation and Analysis in Intel® DAAL



Intel® DAAL Algorithms

Machine Learning in Intel® DAAL



Algorithms supporting batch processing

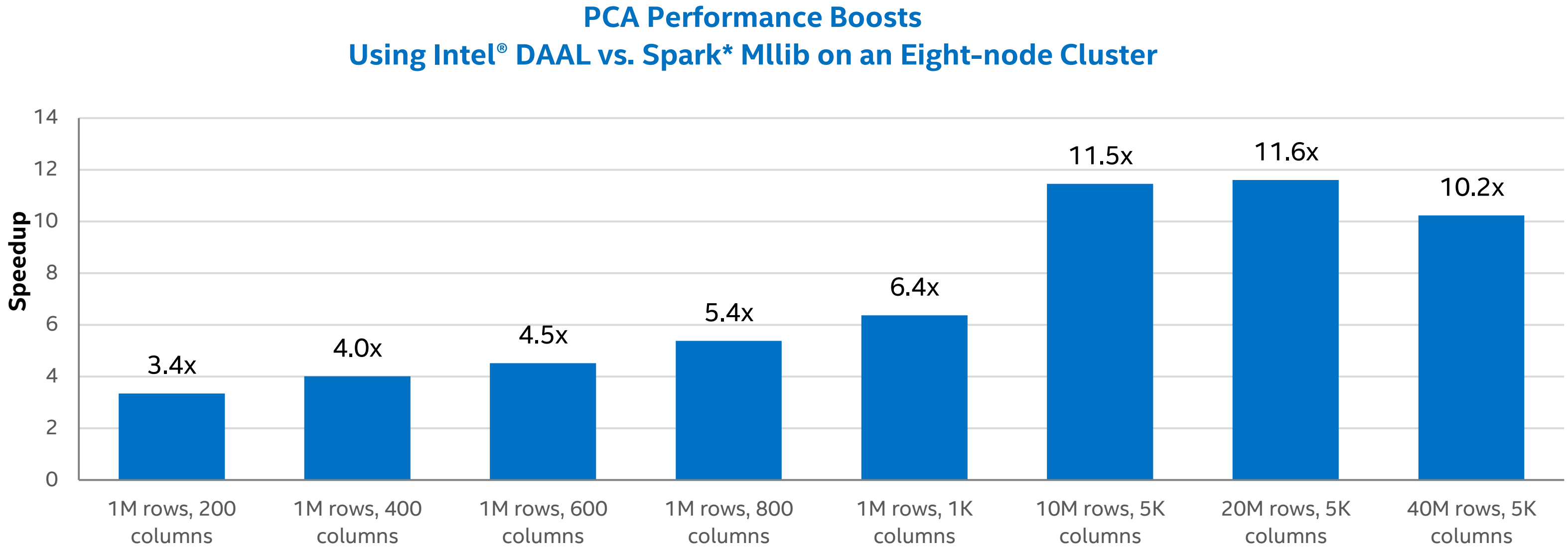
Algorithms supporting batch, online and/or distributed processing

Intel® DAAL Algorithms

C++ API example (Principal Component Analysis in batch computing mode)

```
const size_t nVectors = 1000;

int main()
{
    /* Initialize csv data source to retrieve the input data */
    FileDataSource<CSVFeatureManager> dataSource(dataFileName,
                DataSource::doAllocateNumericTable,
                DataSource::doDictionaryFromContext);

    /* Retrieve the data from the input file */
    dataSource.loadDataBlock(nVectors);

    /* Create correlation method of PCA algorithm */
    pca::Batch<> algorithm;

    /* Set input data */
    algorithm.input.set(pca::data, dataSource.getNumericTable());

    /* Run the algorithm */
    algorithm.compute();

    /* Get the access to the results */
    services::SharedPtr<pca::Result> result = algorithm.getResult();
    result->get(pca::eigenvalues);
    result->get(pca::eigenvectors);

    return 0;
}
```

Intel® DAAL Algorithms



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.

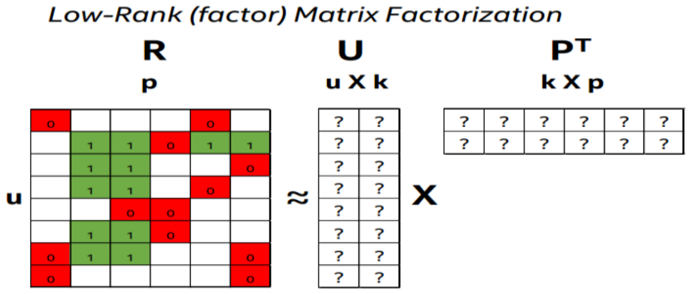
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may 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® DAAL Algorithms

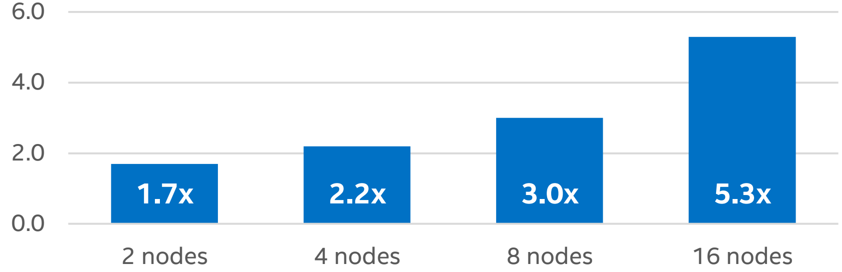
Alternating Least Square algorithm

- Moves to low-dimensional latent feature space and factorize: R = U x P^T
  - U - matrix of size u x k, association (user, feature)
  - P – matrix of size p x k, association (product, feature)
- Iterative algorithm minimizing least square error
  - Initializes U with random data, calculates P
  - Fixes P and calculates U
  - Matrix decompositions and linear solvers
- Recommendations – matrix multiply of U and P (“all items for all users” mode)



$$min \sum_{ij} (r_{ij} - u_i p_j^T)^2 + \lambda \left(\sum_i \| r_i \|^2 + \sum_j \| p_j \|^2 \right)$$

Intel® DAAL ALS in distributed computing mode. Speedup vs 1 node



Configuration Info:
HW (each node): Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz, 2x18 cores, HT is ON, RAM 128GB;
Versions: Oracle Linux Server 6.6, Intel® DAAL 2017 Gold, Intel® MPI 5.1.3; Interconnect: 1 GB Ethernet.
10M users, 10M items, 100M ratings, 10 factors 15 iterations

Intel® DAAL Algorithms

MeritData Inc use-case

"Through close collaboration with Intel engineers, we adopted the Intel® Data Analytics Acceleration Library and Intel® Math Kernel Library for algorithm optimization in our big data analysis platform (Tempo). The performance— and customers' experience— is improved significantly. We really appreciate the collaboration with Intel, and are looking forward to more collaboration."*
Jin Qiang Data Mining Algorithm Architect MeritData Inc.

https://software.intel.com/sites/default/files/managed/32/71/meritdata-case-study.pdf

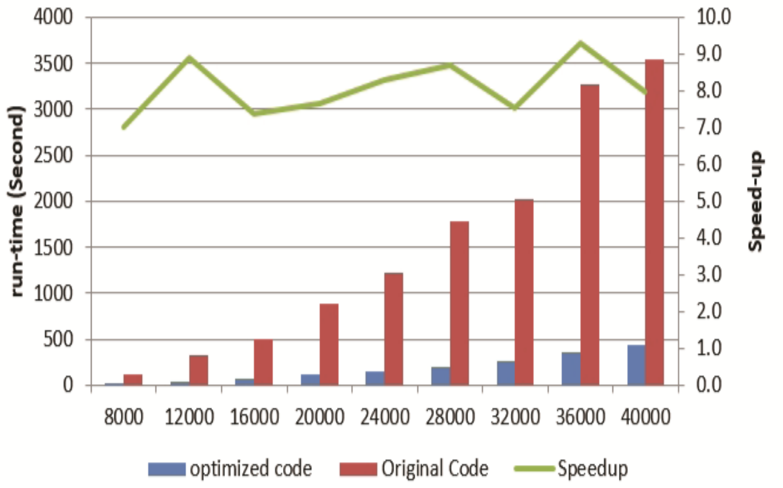
Original Code

```
public Matrix getKernelMatrix()
throws Exception {
Matrix result = new
Matrix(m_data.numInstances(),
m_data.numInstances(), 0);
for (int i = 0; i <
m_data.numInstances() - 1; i++) {
for (int j = i + 1; j <
m_data.numInstances(); j++) {
result.set(i, j, evaluate(i, j,
m_data.instance(0)));
}
}
result = result.plus(
result.transpose().plus(
Matrix.identity(m_data.numInstances()
,
m_data.numInstances()))).copy();
return result;
}
```

Optimized by Intel® DAAL

```
jobject getKernelMatrix(JNIEnv*
env,jobject,jdouble param,jint rows,jint
cols,jobject byteBuffer,jobject dstBuffer){
…
kernel_function::linear::Batch<>
linearKernel;
/* Set the kernel algorithm parameter */
linearKernel.parameter.k = 1.0;
linearKernel.parameter.b = 1.0;
linearKernel.parameter.computationMode =
kernel_function::matrixMatrix;
/* Set an input data table for the algorithm
*/
linearKernel.input.set(kernel_function::X,
data);
linearKernel.input.set(kernel_function::Y,
data);
/* Compute the linear kernel function */
linearKernel.compute();
/* Get the computed results */
services::SharedPtr<kernel_function::Result>
lkResult = linearKernel.getResult();
/* Get the results */
services::SharedPtr<NumericTable> lkMat =
lkResult->get(kernel_function::values);
BlockDescriptor<double> block;
lkMat->getBlockOfRows(0, rows, readOnly,
block);
...
}
```

Table 2. L1/2 sparse code before and after optimization with the iteration algorithm

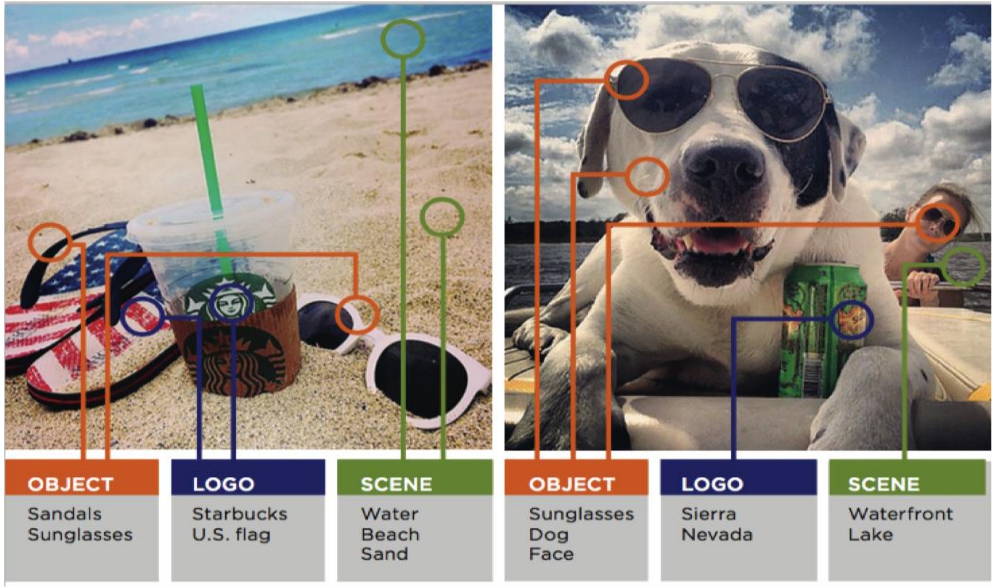


Configuration Info - Versions: Intel Data Analysis Acceleration library from Parallel_studio_xe_2017_beta; Hardware: Intel® Xeon CPU E5-2699 V3 2.30GHz, 2 sockets x 18 cores, AVX 2.0Supported. 45MB Cache, 128 GB Memory; Operating System: Centos6.7; Benchmark Source: MeritData test code and test data set

Up to 9x performance gain when doing Machine Learning with Intel® DAAL

ACCELERATING DEEP LEARNING ON INTEL® IA

Introduction



Learn multiple levels of representation and abstraction in deep fashion to make sense of data (e.g., text, images)

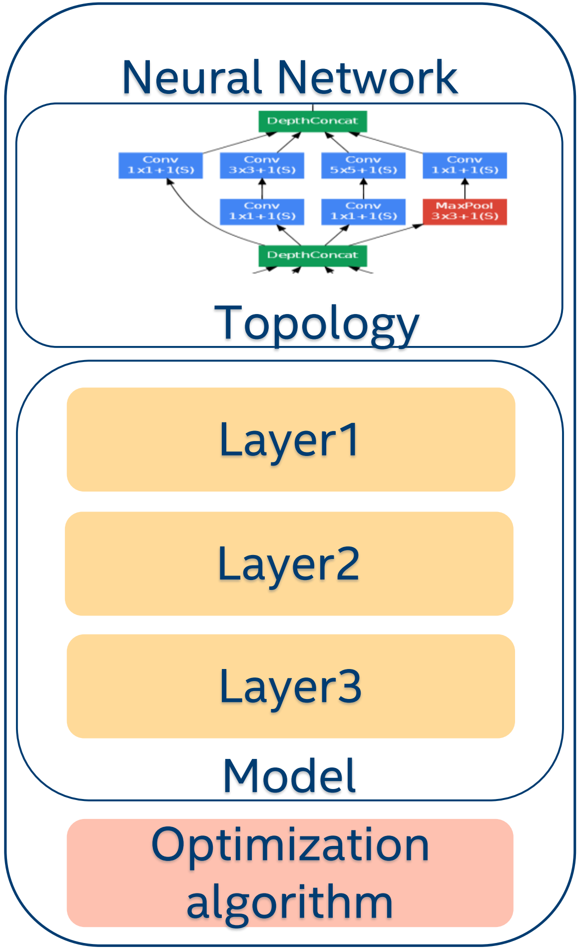
Deep learning with Intel® DAAL

- Use of combination of ML and DL algorithms with the same set of APIs and data structures
- Have all building blocks (layers, model, optimization solver) in one library to support ability to construct the whole topology or use layers and optimization solvers independently
- Optimization of the whole analytical flow
- Support of computations in single- and multi-node modes
- Rely on use of Intel® MKL primitives

Deep learning with Intel® DAAL

NN components in DAAL

- **Layer**: “NN building block”, forward & backward computations for single layer
- **Model:** a set of layers, weights and biases, service
- **NN Configuration**: the structure to describe and register NN topology
- **Optimization solver**: updates weights and biases after forward-backward pass according to specified objective function
- **NN**: topology, the model and the optimization algorithm. Executes forward and backward pass followed by optimization step
- **Multi-dimensional data structure (tensor):** structure used to represent complex data (e.g., stream of images, etc)



Deep learning with Intel® DAAL

Create algorithm for NN training
Initialize NN using its configuration

Set input data (tensors) and training parameters including parameters of optimization solver

Train network
Get NN model

```
training::Batch<> trainingNet;

Collection<LayerDescriptor> layersConf = configureNet();
trainingNet.initialize(trainingData->getDimensions(), layersConf);

trainingNet.input.set(training::data, trainingData);
trainingNet.input.set(groundTruth, trainingGroundTruth);
trainingNet.parameter.optimizationSolver-> parameter.learningRateSequence
= SharedPtr<NumericTable>(new HomogenNumericTable<>(1,1,doAllocate,0.01));
trainingNet.parameter.nIterations = 6000;

trainingNet.compute();

services::SharedPtr<prediction::Model> predictionModel =
trainingNet.getResult()->get(model)->getPredictionModel();
```

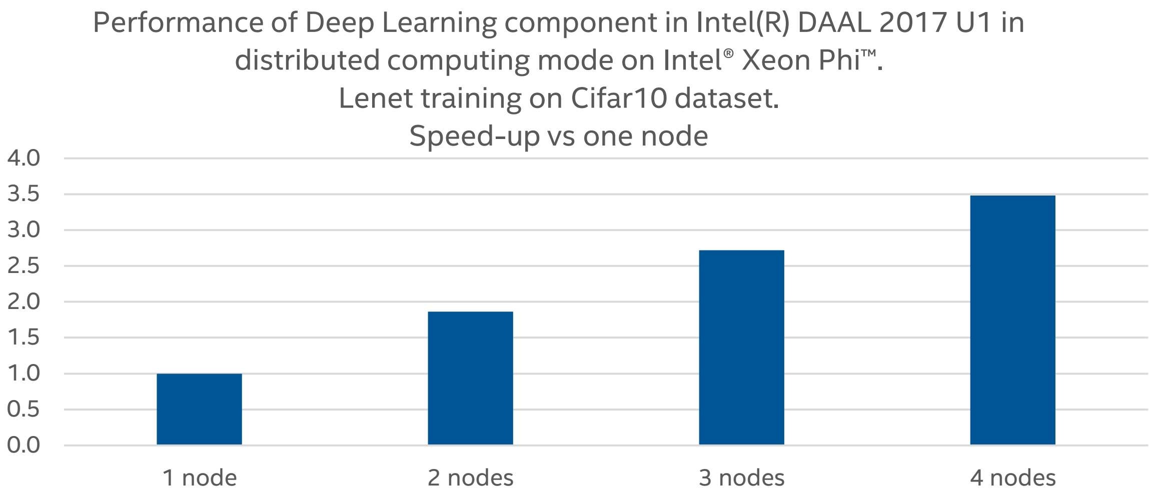
Create algorithm for NN inference

Set input data and model

Do inference
Get inferenceresults in the form of tensor

```
prediction::Batch<> predictionNet;

predictionNet.input.set(prediction::model, predictionModel);
predictionNet.input.set(prediction::data, predictionData);

predictionNet.compute();

SharedPtr<Tensor> predictionResults =
predictionNet.getResult()->get(prediction::prediction);
```

Deep learning with Intel® DAAL Distributed training



Each node: Intel® Xeon Phi™ 7250 68 cores @ 1.4GHz, RAM 16 GB, MCDRAM in cache mode
Intel® OmniPath Architecture 100 Gb/sec
Data is provided by Colfax company

More optimizations in Intel® DAAL DL training – in future releases

CONCLUSION AND RESOURCES

Conclusions

- Intel® DAAL optimizes the whole analytical flow from data acquisition till ML model training and inference
  - ~11 times faster than alternative when computing distributed PCA
  - ~9x performance gain when computing kernel function with Intel® DAAL
- Use Intel® DAAL, if you need to a mix of ML & DL computations
- Close to linear scalability of distributed DL training with Intel® DAAL for selected topologies and datasets

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 & Optimization Notice

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™