Cineca TRAINING High Performance Computing 2017

Domain specific libraries for Deep Learning

Riccardo Zanella – r.zanella@cineca.it

SuperComputing Applications and Innovation Department

CINECA

Table of Contents

Learning Algorithm

Definition of Learning Algorithm [Mitchell 1997][1]

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Learning Algorithm

Definition of Learning Algorithm [Mitchell 1997][1]

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

So we need to identify:

- the class of tasks T

Learning Algorithm

Definition of Learning Algorithm [Mitchell 1997][1]

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

So we need to identify:

- the class of tasks T
- the measure of performance P

Learning Algorithm

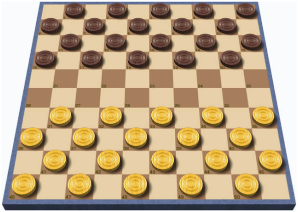
Definition of Learning Algorithm [Mitchell 1997][1]

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

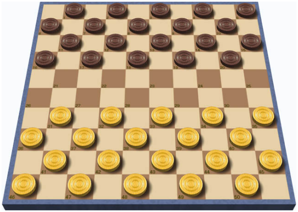
So we need to identify:

- the class of tasks T
- the measure of performance P
- the source of experience E

Example: checkers game

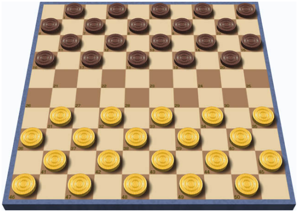


Example: checkers game



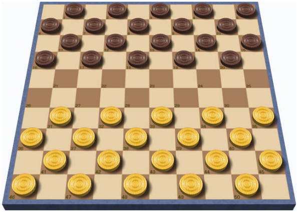
- **task class T:** playing checkers

Example: checkers game



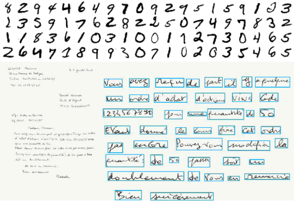
- **task class T:** playing checkers
- **performance measure P:** fraction of games won against opponents

Example: checkers game

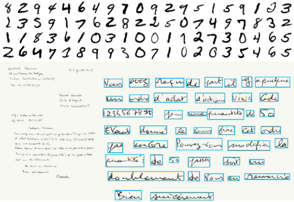


- **task class T:** playing checkers
- **performance measure P:** fraction of games won against opponents
- **training experience E:** playing practice games against itself

Example: handwritten characters recognition

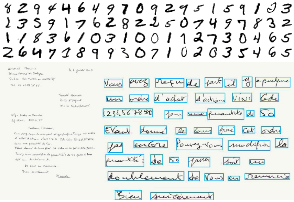


Example: handwritten characters recognition



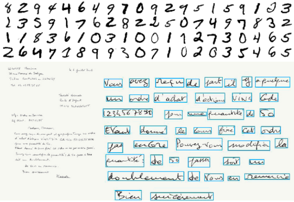
- **task class T:** recognizing and classifying handwritten characters within images

Example: handwritten characters recognition



- **task class T:** recognizing and classifying handwritten characters within images
- **performance measure P:** fraction of characters correctly classified

Example: handwritten characters recognition



- **task class T:** recognizing and classifying handwritten characters within images
- **performance measure P:** fraction of characters correctly classified
- **training experience E:** a database of handwritten characters with given classifications

Example: supervised learning

- **training experience E:** a number of traininig examples $E = \{z_1, z_2, z_3 \dots\}$
  each example is a (input,target) pair: $Z_i = (X_i, Y_i)$

Example: supervised learning

- **training experience E:** a number of traininig examples $E = \{z_1, z_2, z_3 \dots\}$
  each example is a (input,target) pair: $Z_i = (X_i, Y_i)$
- **task class T:** a decision function f able to predict unknown Y from known X

Example: supervised learning

- **training experience E:** a number of traininig examples $E = \{z_1, z_2, z_3 \ldots\}$
  each example is a (input,target) pair: $Z_i = (X_i, Y_i)$
- **task class T:** a decision function f able to predict unknown Y from known X
- **performance measure P:** a loss function L to measure the (non-symmetric) distance $L(f, Z_i) = d(f(X_i), Y_i)$

Example: supervised learning

- **training experience E:** a number of traininig examples $E = \{z_1, z_2, z_3 \dots\}$
  each example is a (input,target) pair: $Z_i = (X_i, Y_i)$
- **task class T:** a decision function f able to predict unknown Y from known X
- **performance measure P:** a loss function L to measure the (non-symmetric) distance $L(f, Z_i) = d(f(X_i), Y_i)$

Examples:

- regression
  - X is a real-valued scalar or vector
  - Y is a scalar real value
  - f is able to predict Y_i value from X_i
  - L is usually the euclidean norm

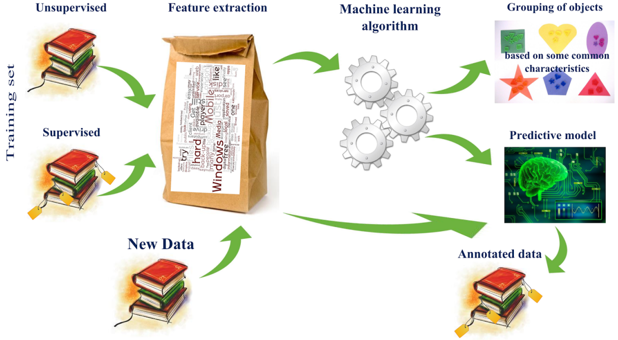
Example: supervised learning

- **training experience E:** a number of traininig examples $E = \{z_1, z_2, z_3 \ldots\}$
  each example is a (input,target) pair: $Z_i = (X_i, Y_i)$
- **task class T:** a decision function f able to predict unknown Y from known X
- **performance measure P:** a loss function L to measure the (non-symmetric) distance $L(f, Z_i) = d(f(X_i), Y_i)$

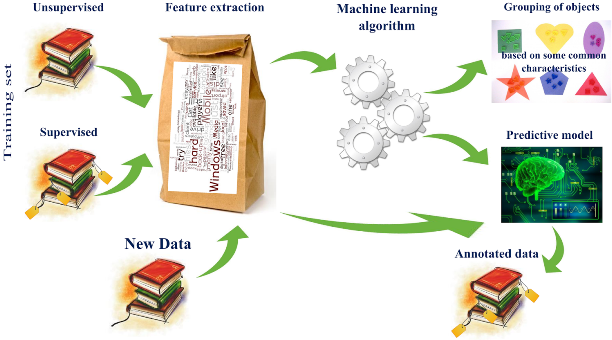
Examples:

- regression
  - X is a real-valued scalar or vector
  - Y is a scalar real value
  - f is able to predict Y_i value from X_i
  - L is usually the euclidean norm
- classification
  - X is a real-valued scalar or vector (features)
  - Y is an integer (label) corresponding to a class index
  - f is able to provide the probability of X_i being in class Y_i
  - L is usually the negative log-likelihood

Machine Learning

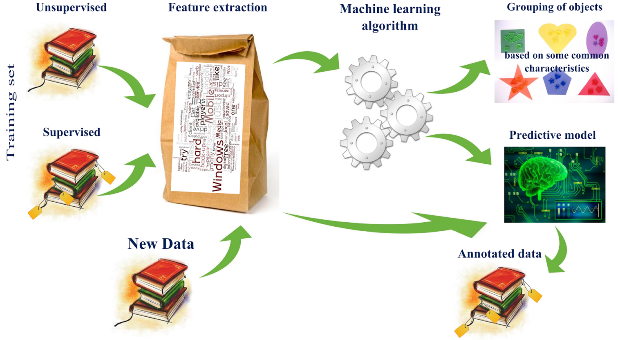


Machine Learning



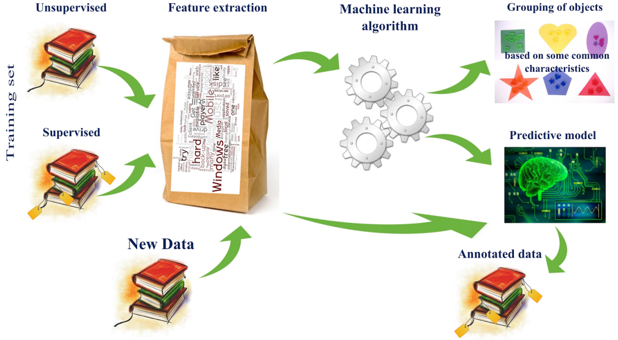
- Application of computer-enabled algorithm to a data set to find a pattern

Machine Learning



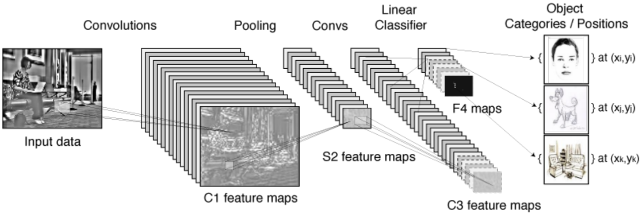
- Application of computer-enabled algorithm to a data set to find a pattern
- Wide range of tasks: segmentation. classification, clustering, supervised/unsupervised learning

Machine Learning

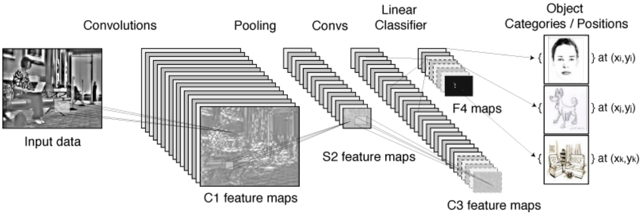


- Application of computer-enabled algorithm to a data set to find a pattern
- Wide range of tasks: segmentation. classification, clustering, supervised/unsupervised learning
- Various algorithms: association rules, decision trees, SVM

Deep Learning

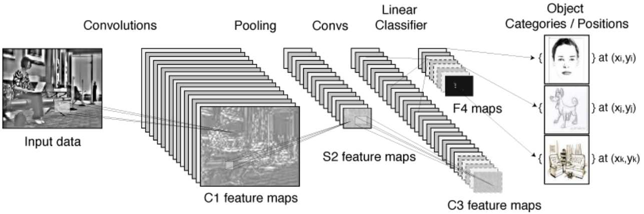


Deep Learning



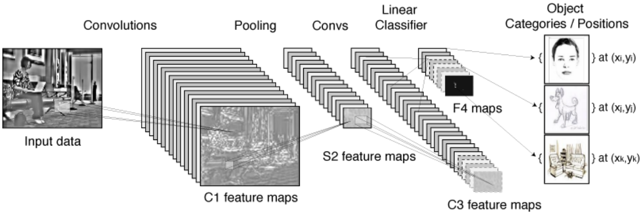
- Application of an Artificial Neural Network to a data set to find a pattern

Deep Learning



- Application of an Artificial Neural Network to a data set to find a pattern
- Multiple hidden layers (to mimic human brain processes associated to vision/hearing)

Deep Learning



- Application of an Artificial Neural Network to a data set to find a pattern
- Multiple hidden layers (to mimic human brain processes associated to vision/hearing)
- Big data sets and relevant number of variables

Framework desired features

We are interested in:

- classical machine learning algorithms
- deep learning approach (especially convolutional neural network)

Framework desired features

We are interested in:

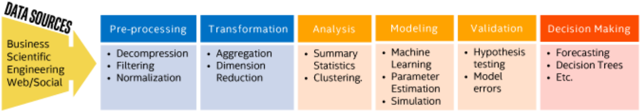
- classical machine learning algorithms
- deep learning approach (especially convolutional neural network)
- high level language (Python)
- little/no programming effort

Framework desired features

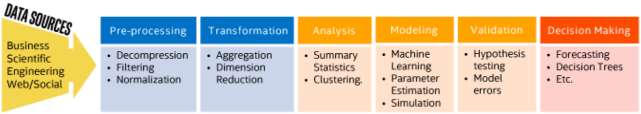
We are interested in:

- classical machine learning algorithms
- deep learning approach (especially convolutional neural network)
- high level language (Python)
- little/no programming effort
- integration with existing pipelines
- multi-core CPU and/or many-core GPU support

Intel Data Analytics Acceleration Library (DAAL)

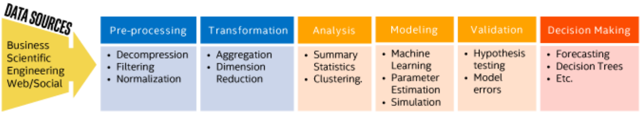


Intel Data Analytics Acceleration Library (DAAL)



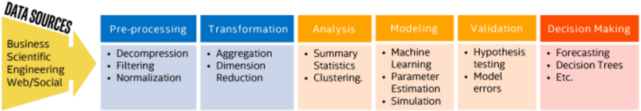
- Functions for machine learning, deep learning, data analytics

Intel Data Analytics Acceleration Library (DAAL)



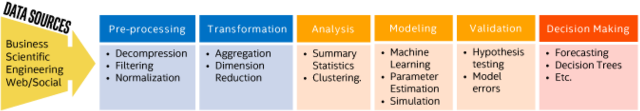
- Functions for machine learning, deep learning, data analytics
- Optimized for Intel architecture devices (processors, coprocessors, and compatibles)

Intel Data Analytics Acceleration Library (DAAL)



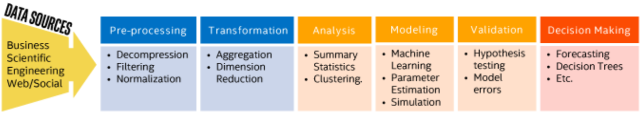
- Functions for machine learning, deep learning, data analytics
- Optimized for Intel architecture devices (processors, coprocessors, and compatibles)
- C++, Java and Python APIs

Intel Data Analytics Acceleration Library (DAAL)



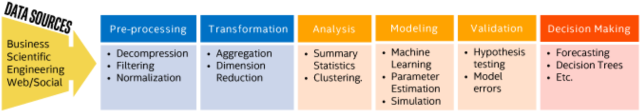
- Functions for machine learning, deep learning, data analytics
- Optimized for Intel architecture devices (processors, coprocessors, and compatibles)
- C++, Java and Python APIs
- Connectors to popular data sources including Spark and Hadoop

Intel Data Analytics Acceleration Library (DAAL)



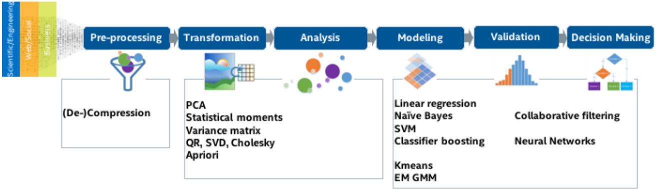
- Functions for machine learning, deep learning, data analytics
- Optimized for Intel architecture devices (processors, coprocessors, and compatibles)
- C++, Java and Python APIs
- Connectors to popular data sources including Spark and Hadoop
- Open source version under Apache 2.0 license

Intel Data Analytics Acceleration Library (DAAL)

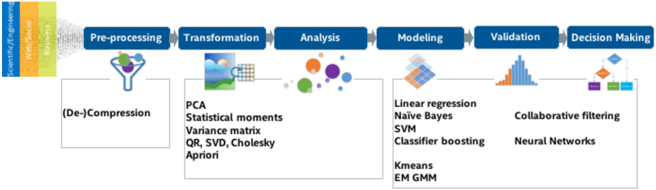


- Functions for machine learning, deep learning, data analytics
- Optimized for Intel architecture devices (processors, coprocessors, and compatibles)
- C++, Java and Python APIs
- Connectors to popular data sources including Spark and Hadoop
- Open source version under Apache 2.0 license
- Paid versions include premium support.

Algorithms

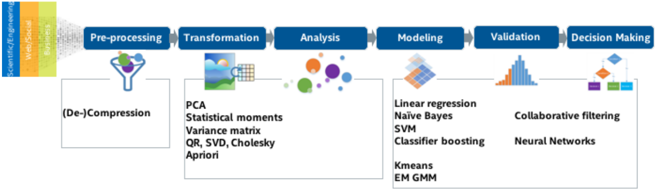

Scientific/Engineering
Web/Social
Business
Pre-processing
Transformation
Analysis
Modeling
Validation
Decision Making
(De-)Compression
PCA
Statistical moments
Variance matrix
QR, SVD, Cholesky
Apriori
Linear regression
Naïve Bayes
SVM
Classifier boosting
Kmeans
EM GMM
Collaborative filtering
Neural Networks

Algorithms



Statistics: min, max, mean, standard deviation, correlation, covariance matrix, correlation distance matrix, cosine distance matrix

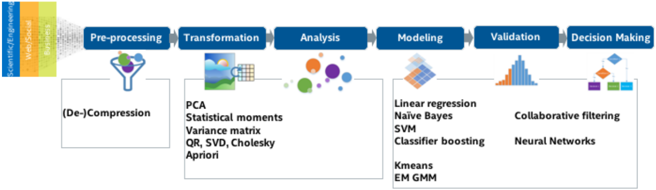
Algorithms



Statistics: min, max, mean, standard deviation, correlation, covariance matrix, correlation distance matrix, cosine distance matrix

Factorizations: Cholesky, QR, SVD

Algorithms

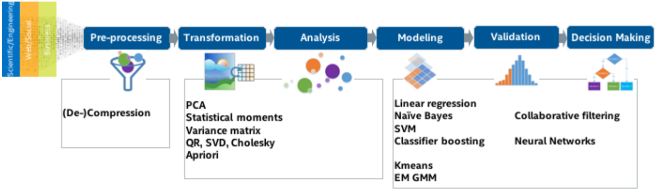


Statistics: min, max, mean, standard deviation, correlation, covariance matrix, correlation distance matrix, cosine distance matrix

Factorizations: Cholesky, QR, SVD

Dimensionality Reduction: PCA

Algorithms



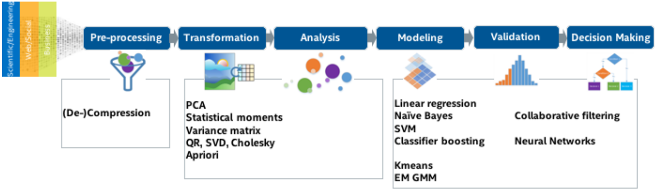
Statistics: min, max, mean, standard deviation, correlation, covariance matrix, correlation distance matrix, cosine distance matrix

Factorizations: Cholesky, QR, SVD

Dimensionality Reduction: PCA

Classification: Naive Bayes, K-Nearest Neighbors, SVM, multiclass classification

Algorithms



Statistics: min, max, mean, standard deviation, correlation, covariance matrix, correlation distance matrix, cosine distance matrix

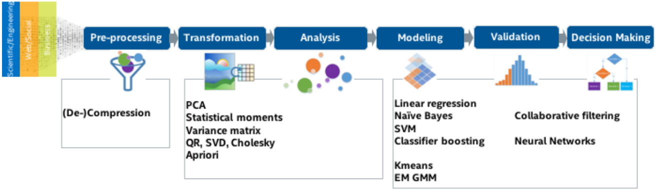
Factorizations: Cholesky, QR, SVD

Dimensionality Reduction: PCA

Classification: Naive Bayes, K-Nearest Neighbors, SVM, multiclass classification

Neural Networks: layers of type: fully-connected, activation, convolutional, normalization, concat, split, softmax, loss function

Algorithms



Statistics: min, max, mean, standard deviation, correlation, covariance matrix, correlation distance matrix, cosine distance matrix

Factorizations: Cholesky, QR, SVD

Dimensionality Reduction: PCA

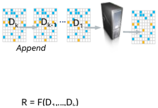
Classification: Naive Bayes, K-Nearest Neighbors, SVM, multiclass classification

Neural Networks: layers of type: fully-connected, activation, convolutional, normalization, concat, split, softmax, loss function

Clustering: K-Means, EM for GMM

Processing Modes

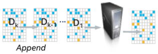
Processing Modes



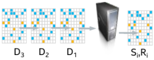
Batch processing

All data is stored in the memory of a single node. An Intel DAAL function is called to process the data all at once.

Processing Modes



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

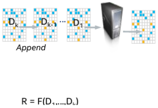


$S_{i+1} = T(S_i,D_i)$
$R_{i+1} = F(S_{i+1})$

Batch processing
All data is stored in the memory of a single node. An Intel DAAL function is called to process the data all at once.

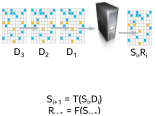
Streaming processing
All data does not fit in memory, or when data is arriving piece by piece. Intel DAAL can process data chunks individually and combine all partial results at the finalizing stage.

Processing Modes



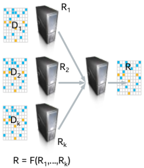
Batch processing

All data is stored in the memory of a single node. An Intel DAAL function is called to process the data all at once.



Streaming processing

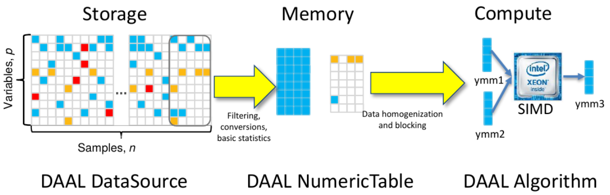
All data does not fit in memory, or when data is arriving piece by piece. Intel DAAL can process data chunks individually and combine all partial results at the finalizing stage.



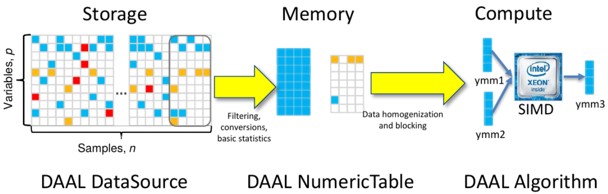
Distributed processing

Intel DAAL supports a model similar to MapReduce. Slaves in a cluster process local data (map stage), and then the master process collects and combines partial results from slaves (reduce stage).

DAAL data flow



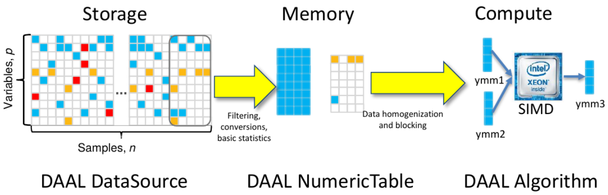
DAAL data flow



Data sources:

- file based (CSV, binary)
- database query (ODBC, SQL)
- Python: numpy array interoperability

DAAL data flow



Data sources:

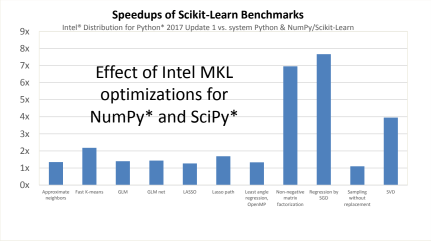
- file based (CSV, binary)
- database query (ODBC, SQL)
- Python: numpy array interoperability

Data structures:

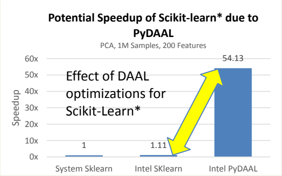
- numeric tables
  - homogeneous data: dense, sparse, packed, triangular matrix, symmetric matrix
  - heterogeneous data: SOA vs AOS
- tensors (n-dimensional matrix)

Official Intel benchmark results (I)

Skt-Learn* Optimizations With Intel® MKL… And Intel® DAAL



Intel® Distribution for Python* ships Intel® Data Analytics Acceleration Library with Python interfaces, a.k.a. pyDAAL



System info: 32x Intel® Xeon® CPU E5-2698 v3 @ 2.30GHz, disabled HT, 64GB RAM; Intel® Distribution for Python* 2017 Gold; Intel® MKL 2017.0.0; Ubuntu 14.04.4 LTS; Numpy 1.11.1; scikit-learn 0.17.1.

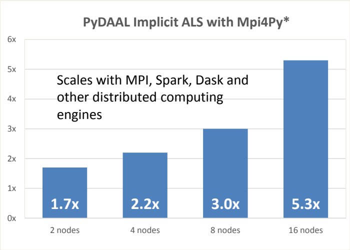
[Supercomputing 2016 (SC16), November 13-18, 2016, Salt Lake City]

Official Intel benchmark results (II)

intel SC16

Distributed Parallelism

- Intel® MPI* accelerates Intel® Distribution for Python (mpi4py*, ipyparallel*)
- Intel Distribution for Python also supports
  - PySpark* - Python* interfaces for Spark*, an engine for large-scale data processing
  - Dask* - flexible parallel computing library for numerical computing



Configuration Info: Hardware (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.

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 .

[Supercomputing 2016 (SC16), November 13-18, 2016, Salt Lake City]

pyDAAL installation

Requirements:

- Intel Math Kernel Library (MKL): for BLAS and LAPACK
- Integrated Performance Primitives (IPP) for data compression/decompression
- Threading Building Blocks (TBB) for multicore and many-core parallelism

pyDAAL installation

Requirements:

- Intel Math Kernel Library (MKL): for BLAS and LAPACK
- Integrated Performance Primitives (IPP) for data compression/decompression
- Threading Building Blocks (TBB) for multicore and many-core parallelism

Installation methods:

1. anaconda Intel channel (Linux)
2. Intel distribution (Windows, Linux, OS X)
3. build from source