INTRODUCTION TO INTEL® DATA ANALYTICS ACCELERATION LIBRARY AND INTEL® DISTRIBUTION OF PYTHON

Lakshmi Narasimhan/Jing Xu

Intel Architecture Graphics and Software (IAGS)

SPEED UP DEVELOPMENT

WITH OPEN AI SOFTWARE

MODEL

MACHINE LEARNING | DEEP LEARNING

TOOLKITS
App Developers

ANALYTICS ZOO

MODEL ZOO

OpenVINO

LIBRARIES
Data Scientists

Intel® Data Analytics Acceleration Library (DAAL)

Intel® Distribution for Python* (Sklearn*, Pandas*)

R (Cart, Random Forest, e1071)

Distributed (MlLib on Spark, Mahout)

Intel Optimized Frameworks

TensorFlow*, BigDL, Caffe, ONNX, mxnet, PyTorch

More framework optimizations in progress…

KERNELS
Library Developers

Intel® Math Kernel Library (Intel® MKL)

Intel® oneAPI Collective Communication Library (Intel® oneCCL)

Deep Neural Networks Library (Intel® oneDNN)

CPU ▪ GPU

Visit: www.intel.ai/technology

1 An open source version is available at: 01.org/openvinotoolkit

*Other names and brands may be claimed as the property of others.

Developer personas show above represent the primary user base for each row, but are not mutually-exclusive

All products, computer systems, dates, and figures are preliminary based on current expectations, and are subject to change without notice.

SPEED UP DEVELOPMENT

WITH OPEN AI SOFTWARE

MODEL

MACHINE LEARNING

DEEP LEARNING

TOOLKITS
App Developers

ANALYTICS ZOO
MODEL ZOO
OpenVINO

LIBRARIES
Data Scientists

Intel® Data Analytics Acceleration Library (DAAL)

Intel® Distribution for Python* (Sklearn*, Pandas*)

R (Cart, Random Forest, e1071)

Distributed (MlLib on Spark, Mahout)

Intel Optimized Frameworks

TensorFlow* BigDL Caffe ONNX mxnet PyTorch

More framework optimizations in progress…

KERNELS
Library Developers

Intel® Math Kernel Library (Intel® MKL)

Intel® oneAPI Collective Communication Library (Intel® oneCCL)

Deep Neural Networks Library (Intel® oneDNN)

CPU ▪ GPU

Visit: www.intel.ai/technology

1 An open source version is available at: 01.org/openvinotoolkit
*Other names and brands may be claimed as the property of others.
Developer personas show above represent the primary user base for each row, but are not mutually-exclusive
All products, computer systems, dates, and figures are preliminary based on current expectations, and are subject to change without notice.

Speed-up Machine Learning and Analytics with Intel® oneAPI Data Analytics Library (oneDAL)

Boost Machine Learning & Data Analytics Performance

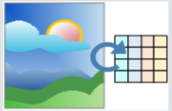
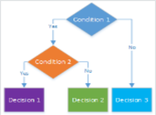
- Helps applications deliver better predictions faster
- Optimizes data ingestion & algorithmic compute together for highest performance
- Supports offline, streaming & distributed usage models to meet a range of application needs
- Split analytics workloads between edge devices and cloud to optimize overall application throughput

Learn More: software.intel.com/daal

What's New in the 2020 Release

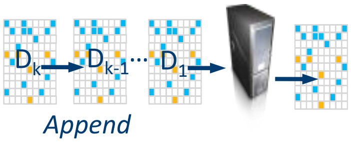
New Algorithms:

- Probabilistic classification and variable importance computation for gradient boosted trees
- Classification stump with information gain and Gini index split methods
- Regression stump with the Mean Squared Error (MSE) algorithm split method

Pre-processing	Transformation	Analysis	Modeling	Validation	Decision Making
					
Decompression, Filtering, Normalization	Aggregation, Dimension Reduction	Summary Statistics Clustering, etc.	Machine Learning (Training) Parameter Estimation Simulation	Hypothesis Testing Model Errors	Forecasting Decision Trees, etc.

Processing Modes

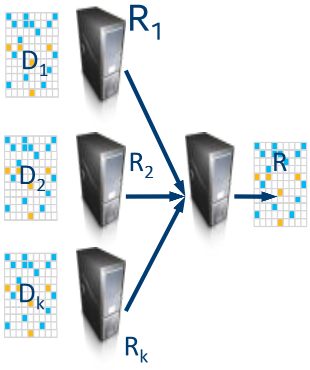
Batch Processing



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

```
d4p.kmeans_init(10, method="plusPlusDense")
```

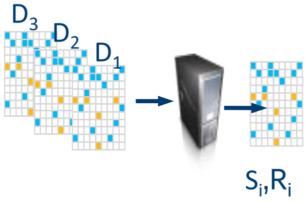
Distributed Processing



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

```
d4p.kmeans_init(10, method="plusPlusDense“, distributed=“True”)
```

Online Processing

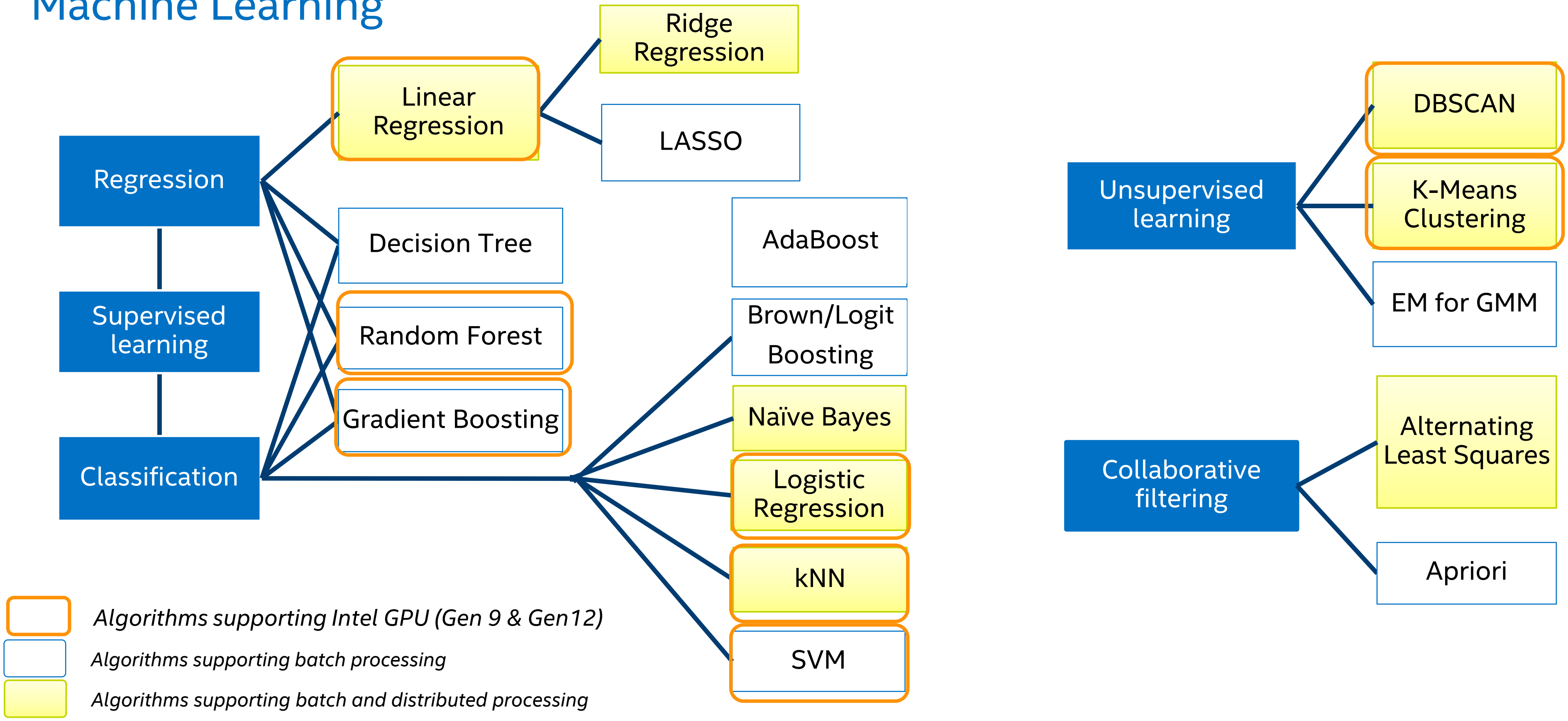


$S_{i+1} = T(S_i,D_i)$
$R_{i+1} = F(S_{i+1})$

```
d4p.kmeans_init(10, method="plusPlusDense“, streaming=“True”)
```

Intel® oneAPI Data Analytics Library(beta) (oneDAL) Algorithms

Machine Learning



Intel® oneAPI Data Analytics Library(beta) (oneDAL) algorithms
Data Transformation and Analysis

Basic statistics for datasets
- Low order moments
- Quantiles
- Order statistics

Correlation and dependence
- Cosine distance
- Correlation distance
- Variance-Covariance matrix

Matrix factorizations
- SVD
- QR
- Cholesky
- tSVD

Dimensionality reduction
- PCA

Association rule mining (Apriori)

Optimization solvers (SGD, AdaGrad, lBFGS, CD)

Outlier detection
- Univariate
- Multivariate

Math functions (exp, log,…)

Algorithms supporting batch processing Intel GPU (Gen 9 & Gen12)
Algorithms supporting batch processing
Algorithms supporting batch, online and/or distributed processing

oneAPI Data Analytics Library (oneDAL)

PCA
KMeans
LinearRegression
Ridge
SVC
pairwise_distances
logistic_regression_path

Scikit-Learn* Equivalents

USE_DAAL4PY_SKLEARN=YES

Scikit-Learn* API Compatible

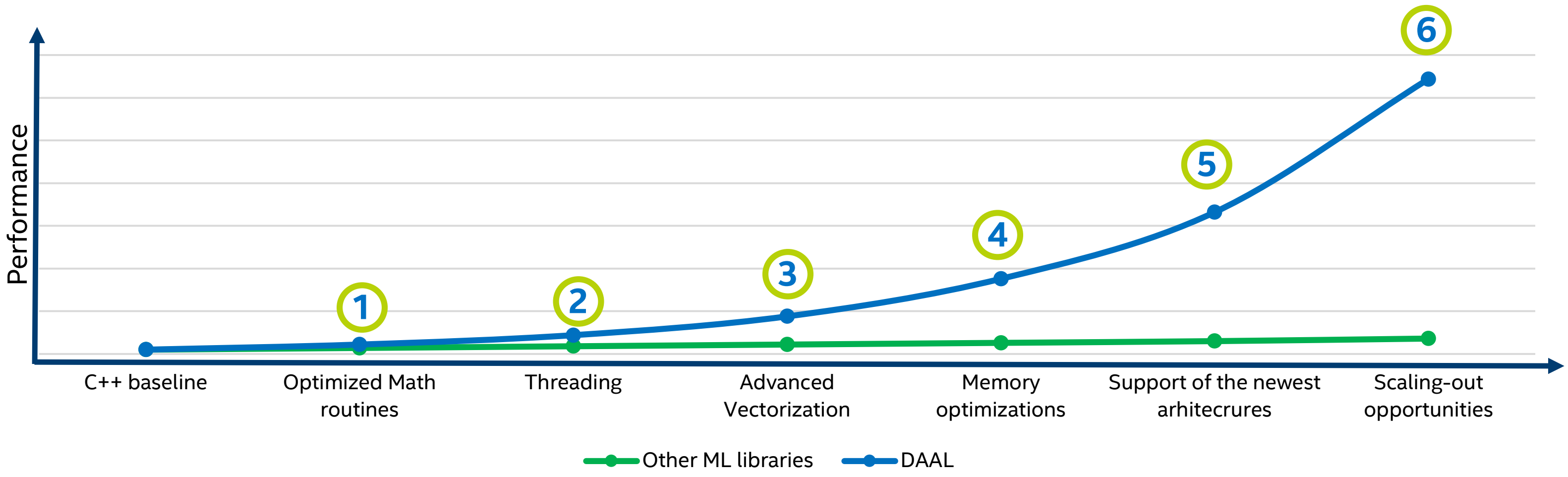
KNeighborsClassifier
RandomForestClassifier
RandomForestRegressor

Use directly for
- Scaling to multiple nodes
- Streaming data
- Non-homogeneous dataframes

daal4py

Intel® oneDAL

What makes Intel® oneDAL faster?



1 The best performance on Intel Architectures with Intel® MKL vs. less performance OS BLAS/LAPACK libs

2 targets to many-core systems to achieve the best scalability on Intel® Xeon, other libs mostly target to client versions with small amount of cores

3 uses the latest available vector-instructions on each architecture, enables them by compiler options, intrinsics. Usually other ML libs build application without vector-instructions support or sse4.2 only.

4 uses the most efficient memory optimization practices: minimally access memory, cache access optimizations, SW memory prefetching. Usually Other ML libs don't make low-level optimizations.

5 enables new instruction sets and other HW features even before official HW lunch. Usually other ML libs do this with long delay.

6 provides distributed algorithms which scale on many nodes

Accelerate libraries with Intel® Distribution for Python*

High Performance Python* for Scientific Computing, Data Analytics, Machine Learning

FASTER PERFORMANCE

Performance Libraries, Parallelism, Multithreading, Language Extensions

Accelerated NumPy*/SciPy*/scikit-learn* with oneMKL[1] & oneDAL[2]

Data analytics, machine learning with scikit-learn, daal4py

Optimized run-times Intel MPI®, Intel® TBB

Scale with Numba* & Cython*

Includes optimized mpi4py, works with Dask* & PySpark*

Optimized for latest Intel® architecture

GREATER PRODUCTIVITY

Prebuilt & Accelerated Packages

Prebuilt & optimized packages for numerical computing, machine/deep learning, HPC & data analytics

Drop-in replacement for existing Python*

Usually NO code changes required!

Conda build recipes included in packages

Free download & free for all uses including commercial deployment

ECOSYSTEM COMPATIBILITY

Supports Python* 2.7 & 3.6, & 3.7 conda, pip

Compatible & powered by Anaconda*, supports conda & pip

Distribution & individual optimized

oneMKL accelerated NumPy*, and SciPy now in Anaconda*!

Optimizations upstreamed to main Python* trunk

Commercial support through Intel® Parallel Studio XE

Intel® Architecture Platforms

Operating System: Windows*, Linux*, MacOS[1]*

[1]Intel® oneAPI Math Kernel Library
[2]Intel® oneAPI Data Analytics Library

Performance Optimization:

Introduction to Python Performance, cont.*

The layers of quantitative Python*

- The Python* language is interpreted and has many type checks to make it flexible
- Each level has various tradeoffs; *NumPy** value proposition is immediately seen
- For best performance, escaping the Python* layer early is best method

Python*

Enforces *Global Interpreter Lock* (GIL) and is single-threaded, abstraction overhead. No advanced types.

NumPy*

Gets around the GIL (multi-thread and multi-core) *BLAS API* can be the bottleneck

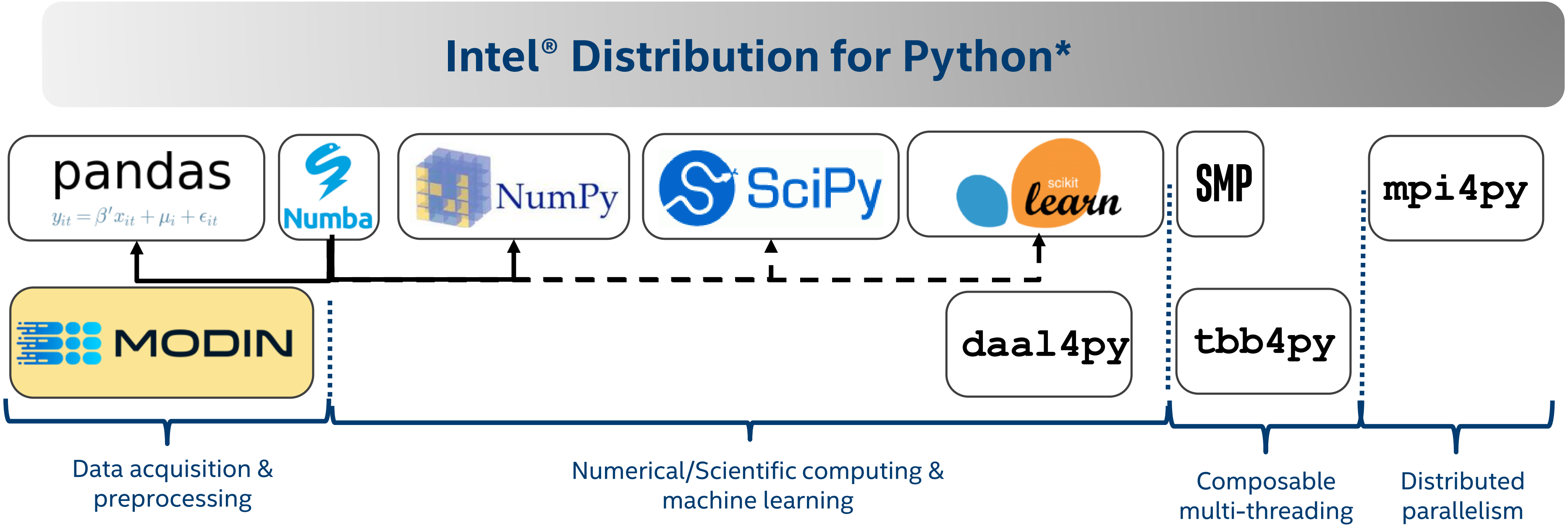
*Basic Linear Algebra Subprograms (BLAS) [CBLAS]

Intel® oneAPI Math Kernel Library (oneMKL)

Gets around BLAS API bottleneck
Much stricter typing
Fastest performance level
Dispatches to hardware vectorization

Intel® oneMKL included with Anaconda* standard bundle; is Free for Python*

Productivity with Performance via Intel® Distribution for Python*

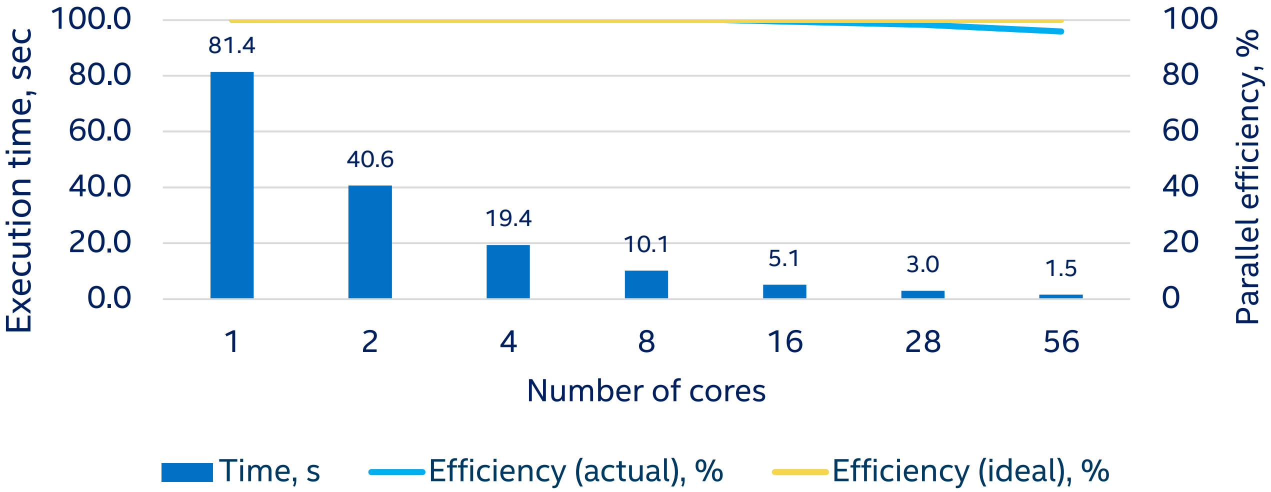


Learn More: software.intel.com/distribution-for-python

https://www.anaconda.com/blog/developer-blog/parallel-python-with-numba-and-parallelaccelerator/

Intel® DAAL 2020 K-means fit, cores scaling

(10M samples, 10 features, 100 clusters, 100 iterations, float32)



Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer. Performance results are based on testing as of **11/11/2019** and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure.

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.

Configuration: Testing by Intel as of **11/11/2019**. Intel® Data Analytics Acceleration Library 2019.3 (Intel® DAAL); Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz, 2 sockets, 28 cores per socket, 10M samples, 10 features, 100 clusters, 100 iterations, float32

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® Distribution for Python* Scikit-learn* Optimizations

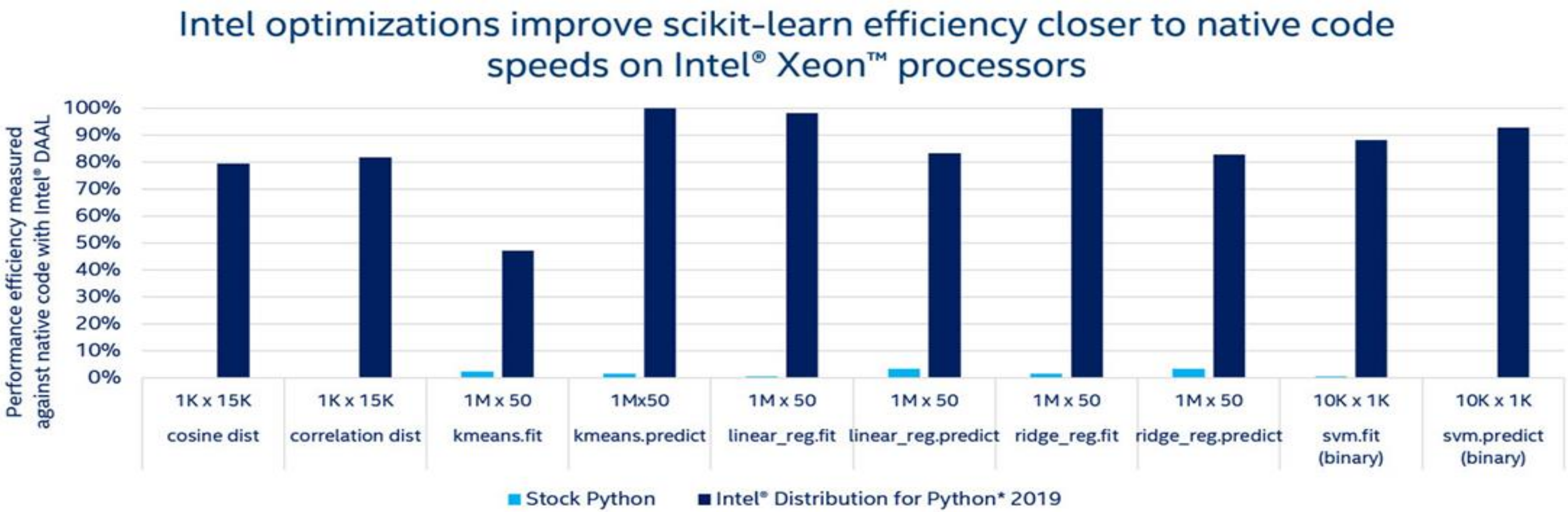


Figure 1**

Figure 1

Performance results are based on testing as of **July 9, 2018** and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure. 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, see Performance Benchmark Test Disclosure.

Testing by Intel as of July 9, 2018. Configuration: Stock Python: python 3.6.6 hc3d631a_0 installed from conda, numpy 1.15, numba 0.39.0, llvmlite 0.24.0, scipy 1.1.0, scikit-learn 0.19.2 installed from pip; Intel Python: Intel® Distribution for Python* 2019 Gold: python 3.6.5 intel_11, numpy 1.14.3 intel_py36_5, mkl 2019.0 intel_101, mkl_fft 1.0.2 intel_np114py36_6,mkl_random 1.0.1 intel_np114py36_6, numba 0.39.0 intel_np114py36_0, llvmlite 0.24.0 intel_py36_0, scipy 1.1.0 intel_np114py36_6, scikit-learn 0.19.1 intel_np114py36_35; OS: CentOS Linux 7.3.1611, kernel 3.10.0-514.el7.x86_64; Hardware: Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz (2 sockets, 18 cores/socket, HT:off), 256 GB of DDR4 RAM, 16 DIMMs of 16 GB@2666MHz

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. For more complete information about compiler optimizations, see our Optimization Notice.

Strong & Weak Scaling via daal4py

Hardware	Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz, EIST/Turbo on
	2 sockets, 20 Cores per socket
	192 GB RAM
	16 nodes connected with Infiniband
Operating System	Oracle Linux Server release 7.4
Data Type	double

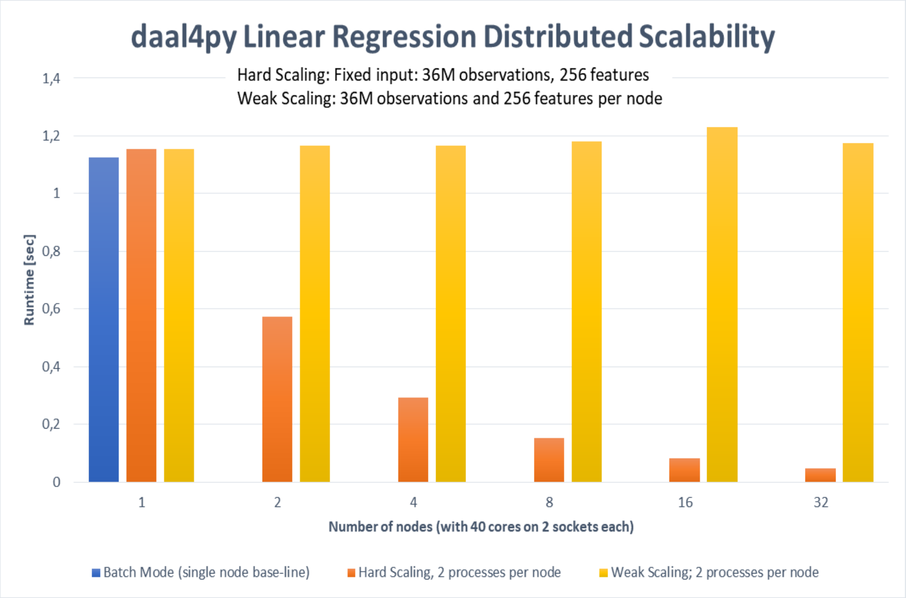


Figure 2**

On a 32-node cluster (1280 cores) daal4py computed linear regression of 2.15 TB of data in 1.18 seconds and 68.66 GB of data in less than 48 milliseconds.

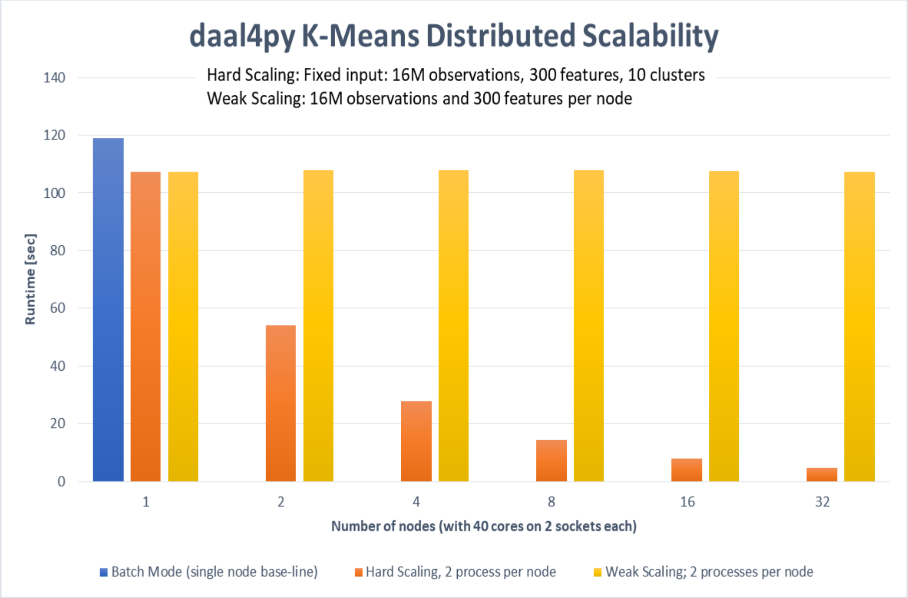
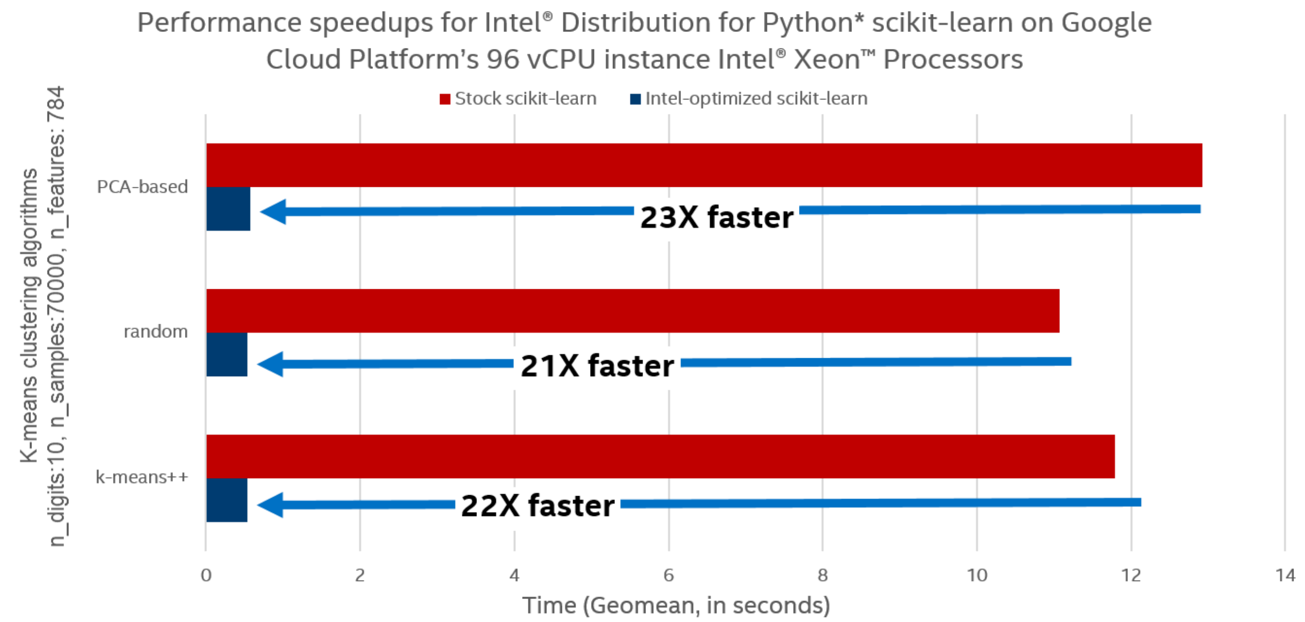


Figure 3**

On a 32-node cluster (1280 cores) daal4py computed K-Means (10 clusters) of 1.12 TB of data in 107.4 seconds and 35.76 GB of data in 4.8 seconds.

Accelerating K-Means



System Configuration: GCP VM, zone us-central1-c; 96 vCPU, Intel Skylake; 360 GB memory. Ubuntu 16.04.3 LTS; Linux instance-1 4.10.0-38-generic #42~16.04.1-Ubuntu SMP Tue Oct 10 16:32:20 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux; Intel® Distribution for Python* from Docker image intelpython/intelpython3_full:latest (created 2017-09-12T20:10:42.862965559Z); Stock Python*: pip install scikit-learn

https://cloudplatform.googleblog.com/2017/11/Intel-performance-libraries-and-python-distribution-enhance-performance-and-scaling-of-Intel-Xeon-Scalable-processors-on-GCP.html

K-Means using daal4py

```
import daal4py as d4p

# daal4py accepts data as CSV files, numpy arrays or pandas dataframes
# here we let daal4py load process-local data from csv files
data = "kmeans_dense.csv"

# Create algob object to compute initial centers
init = d4p.kmeans_init(10, method="plusPlusDense")
# compute initial centers
ires = init.compute(data)
# results can have multiple attributes, we need centroids
centroids = ires.centroids
# compute initial centroids & kmeans clustering
result = d4p.kmeans(10).compute(data, centroids)
```

Distributed K-Means using daal4py

```
import daal4py as d4p

# initialize distributed execution environment
d4p.daalinit()

# daal4py accepts data as CSV files, numpy arrays or pandas dataframes
# here we let daal4py load process-local data from csv files
data = "kmeans_dense_{}.csv".format(d4p.my_procid())

# compute initial centroids & kmeans clustering
init = d4p.kmeans_init(10, method="plusPlusDense", distributed=True)
centroids = init.compute(data).centroids
result = d4p.kmeans(10, distributed=True).compute(data, centroids)
```

```
mpirun -n 4 python ./kmeans.py
```

Streaming data (linear regression) using daal4py

```
import daal4py as d4p

# Configure a Linear regression training object for streaming
train_algo = d4p.linear_regression_training(interceptFlag=True, streaming=True)

# assume we have a generator returning blocks of (X,y)...
rn = read_next(infile)

# on which we iterate
for chunk in rn:
    algo.compute(chunk.X. chunk.y)

# finalize computation
result = algo.finalize()
```

Intel-optimized XGBoost*

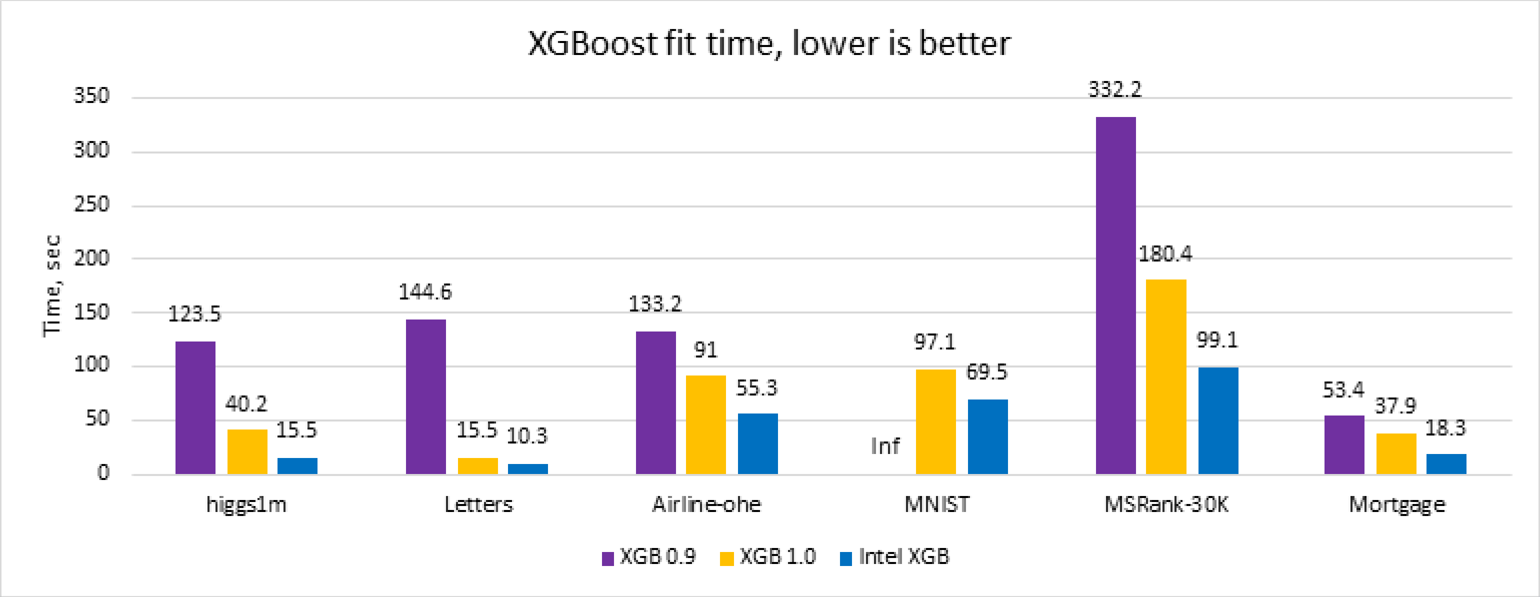


Figure 4**

1) XGBoost* 0.9 – w/ no Intel optimizations
2) XGBoost* 1.0 – the latest official XGBoost
3) XGBoost* from Intel channel

(we expect that XGBoost* 1.1 official will have similar performance).

Intel XGB 0.9

conda install xgboost –c intel

Demo

Scikit-Learn Sample with oneDAL

Original image (96,615 colors)

Quantized image (64 colors, K-Means)

More Resources

Intel® Distribution for Python

- Product page – overview, features, FAQs…
- Training materials – movies, tech briefs, documentation, evaluation guides…
- Support – forums, secure support…

Footnotes and Disclaimers