Intel® AI Analytics Toolkit Classical ML Optimizations

Aaryan Kothapalli, AI Software Solutions Engineer

Yuning Qiu, AI Software Solutions Engineer

intel®

Agenda

- Intel® AI Analytics Toolkit Overview
- Intel Distribution for Python Overview
- Intel® Distribution of Modin
- Intel® Extension for Scikit-Learn* and XGBoost
- Exercises

Intel® AI Analytics Toolkit

Accelerates end-to-end Machine Learning and Data Analytics pipelines with frameworks and libraries optimized for Intel® architectures

Who Uses It?

Data scientists, AI Researchers, Machine and Deep Learning developers, AI application developers

Learn More: intel.com/oneAPI-AIKit

Intel® AI Analytics Toolkit

Deep Learning

- Intel® Optimization for TensorFlow
- Intel® Optimization for PyTorch
- Model Zoo for Intel® Architecture
- Intel® Neural Compressor

Machine learning

- Intel® Extension for Scikit-learn
- Intel® optimizations for XGBoost

Data Analytics

- Intel® Distribution of Modin
- OmniSci Backend

Core Python

Intel®-optimized

- NumPy
- SciPy
- Numba
- Pandas

DPPY

A Brief Overview of Intel® AI Python Offerings

For larger scale and increased performance in data science workloads:

Engineer Data | Create Machine Learning and Deep Learning Models | Deploy

Containers
Intel® Developer Catalog

MLOps
cnvrg.io

Developer Sandbox
Intel® Developer Cloud

Connect AI to Big Data — Spark — BigDL

Accelerate End-to-End Data Science and AI — Intel® AI Analytics Toolkit

Accelerated Python — Python Data API Standard/ Data Parallel Extensions for Python language — Intel® Distribution for Python

Write Once Deploy Anywhere

Data Analytics at Scale

MODIN
NumPy
pandas
SciPy
Numba

Optimized Frameworks and Middleware

TensorFlow
PyTorch
PaddlePaddle
ONNX RUNTIME
scikit learn
mxnet
tvm
dmlc XGBoost
Spark MLlib The Machine Learning Library

With Intel Optimizations

Optimize and Deploy Models

Automate Model Tuning AutoML
SigOpt

Automate Low-Precision Optimization
Intel® Neural Compressor

OpenVINO™ Toolkit

oneAPI
oneDAL
oneDNN
oneCCL
oneMKL

intel. XEON
intel. CORE
Xe
Accelerators
intel. habana
SynapseAI®

Key Features & Benefits

Intel® AI Analytics Toolkit

- Accelerate end-to-end AI and Data Science pipelines, achieve drop-in acceleration with optimized Python tools built using oneAPI libraries (i.e. oneMKL, oneDNN, oneCCL, oneDAL, and more)
- Achieve high-performance deep learning training and inference with Intel-optimized TensorFlow and PyTorch versions, and low-precision optimization with support for fp16, int8 and bfloat16
- Expedite development using open-source Intel-optimized pre-trained deep learning models for best performance via Model Zoo for Intel® Architecture (IA)
- Enable distributed training through Torch-CCL, and support of standards-based Horovod library
- Seamlessly scale Pandas workflows across multi-node dataframes with Intel® Distribution of Modin, accelerate analytics with performant backends such as OmniSci
- Increase machine learning model accuracy and performance with algorithms in Scikit-learn and XGBoost optimized for IA
- Supports cross-architecture development (Intel® CPUs/GPUs) and compute

Getting Started with Intel® AI Analytics Toolkit

Overview	Installation	Hands on	Learning	Support
▪ Visit Intel® AI Analytics Toolkit (AI Kit) for more details and up-to-date product information ▪ Release Notes	▪ Download the AI Kit from Intel, Anaconda or any of your favorite package managers ▪ Get started quickly with the AI Kit Docker Container ▪ Installation Guide ▪ Utilize the Getting Started Guide	▪ Code Samples ▪ Build, test and remotely run workloads on the Intel® DevCloud for free. No software downloads. No configuration steps. No installations.	▪ Machine Learning & Analytics Blogs ▪ Intel AI Blog site ▪ Webinars & articles	▪ Ask questions and share information with others through the Community Forum ▪ Discuss with experts at AI Frameworks Forum

Download Now

Intel Distribution For Python

Intel® Distribution for Python

Developer Benefits

Maximize Performance

Performance Libraries, Parallelism, Multithreading, Language Extensions

- Near-native performance comes through acceleration of core Python numerical packages
- Accelerated NumPy/SciPy/scikit-learn with oneMKL & oneDAL
- Data analytics, machine learning & deep learning with scikit-learn, XGBoost, Modin, daal4py
- Scale with Numba*, Cython*, tbb4py, mpi4py, SDC
- Optimized for latest Intel® architectures

Minimize Development Cost

Drop-in Python Replacement

- Prebuilt optimized packages for numerical computing, machine/deep learning, HPC, & data analytics
- Data-Parallel Python provides cross-architecture XPU support
- Conda build recipes included in packages
- Free download & free for all uses including commercial deployment

Vast Ecosystem

Familiar usage and compatibility

- Supports Python 3
- Supports conda & pip package managers
- Packages available via conda, pip YUM/APT, Docker image on DockerHub
- Commercial support through the Intel® oneAPI Base Toolkit

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

Intel® Architecture Platforms: CPU | GPU | Other accel.

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*

NumPy* and SciPy* Optimizations

Scope

BLAS/LAPACK using oneMKL

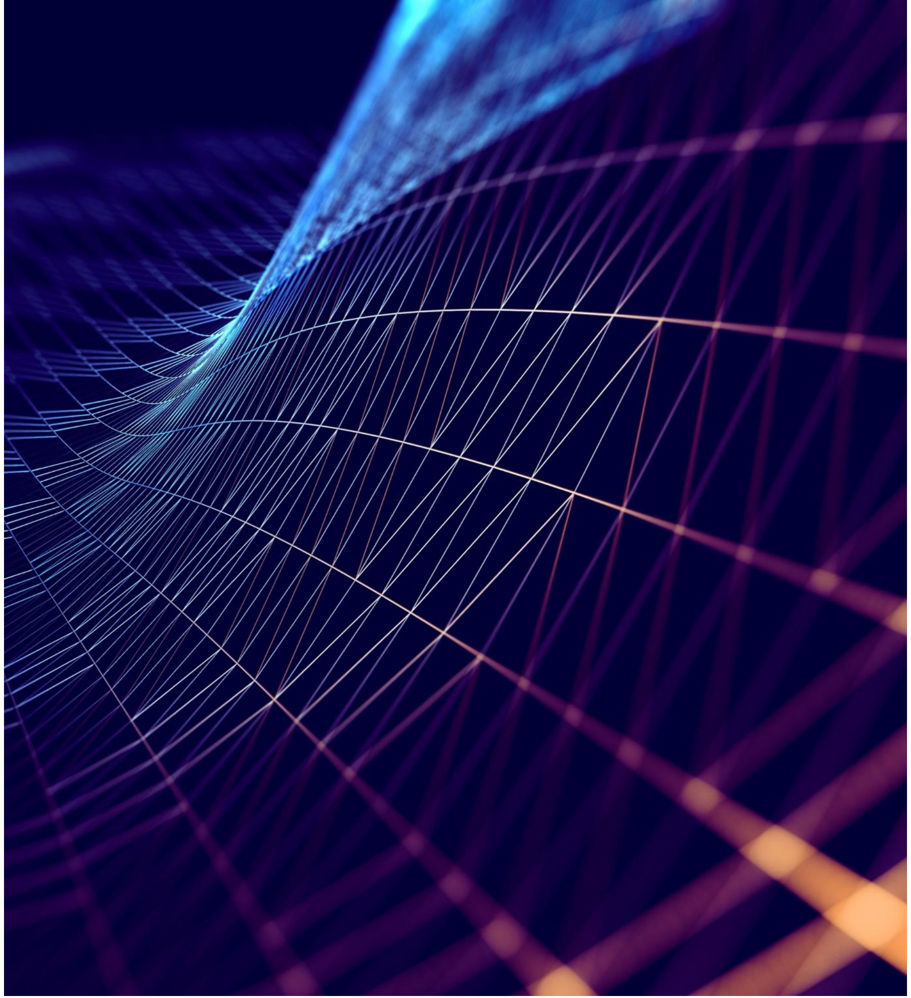
oneMKL-based FFT functionality

Vectorized, threaded universal functions

Use of Intel® C Compiler, and Intel® Fortran Compiler

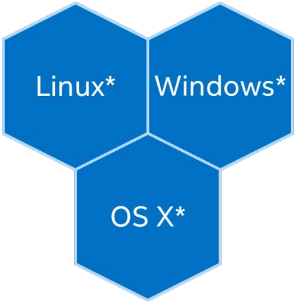
Aligned memory allocation

Threaded memory copying



Choose Your Download Option

Python Solutions	Download Options
Tools and frameworks to accelerate end-to-end data science and analytics pipelines	Intel® AI Analytics Toolkit
Develop fast, performant Python code with essential computational packages	Intel® Distribution for Python
Optimized Python packages from package managers and containers	Conda | YUM | APT | Docker
Develop in the Cloud	Intel® DevCloud



* Also available in the Intel® oneAPI Base Toolkit

1 Line of Code. Infinite Scalability.

Intel Distribution of Modin

intel®

Current Data Loading & ETL Landscape

After a certain data size, need to change your API to handle more data

100 MB+ of Data

Easy to use,
difficult to scale

Easy to scale,
difficult to use

Increasing data size

pandas

APACHE Spark

With Modin, use the same API no matter the scale

Spend the time that would be used to change the workload's API, and
use it to improve your workload and analysis

Easy to use,
Easy to scale

Increasing data size

0-1TB+ of Data

Single Line Code Change for Infinite Scalability

No need to learn a new API to use Modin

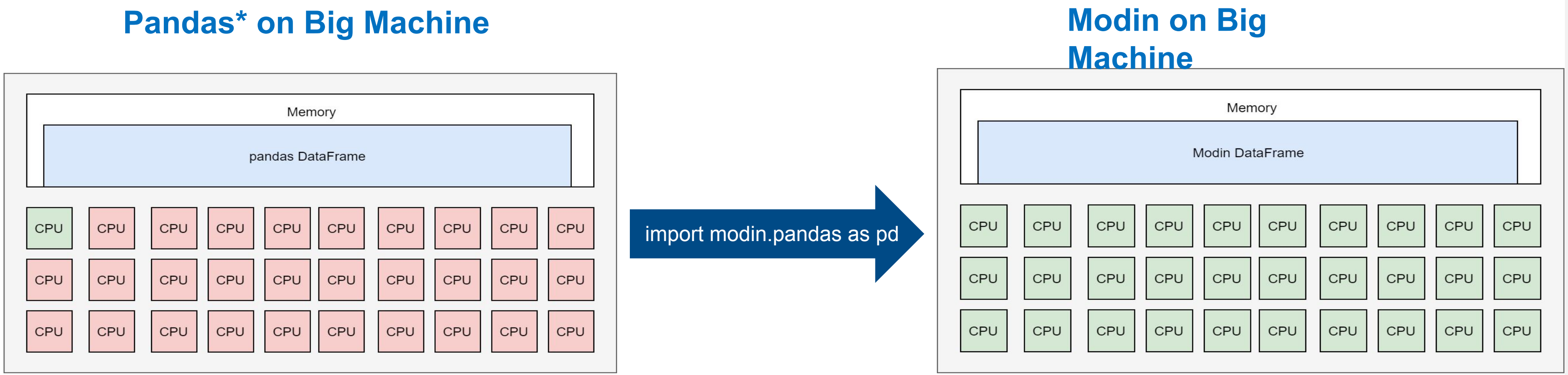
MODIN

```
import pandas as pd
```

- Accelerate your Pandas* workloads across multiple cores and multiple nodes
- No upfront cost to learning a new API
  - import modin.pandas as pd
- Integration with the Python* ecosystem
- Integration with Ray/Dask clusters (run on what you have, even on a laptop!)
- Integration with Intel-built oneAPI Heterogeneous Data Kernels (oneHDK) backend
  - New experimental Modin backend based on HeavyDB* technology

Modin: How it Works

- Modin transparently **distributes the data and computation across available cores**, unlike Pandas which only uses one core at a time
- To use Modin, you **do not need to know how many cores your system has**, and you do not need to specify how to distribute the data



Modin – Layered API view

MODIN

Pandas | SQL | DSL(Ibis) | User Defined DSL

User level API

Query Compiler API

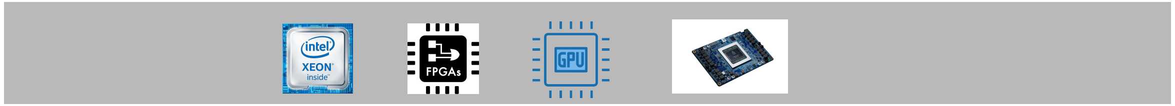
Dataframe Algebra API

Storage API | Execution API

System level API

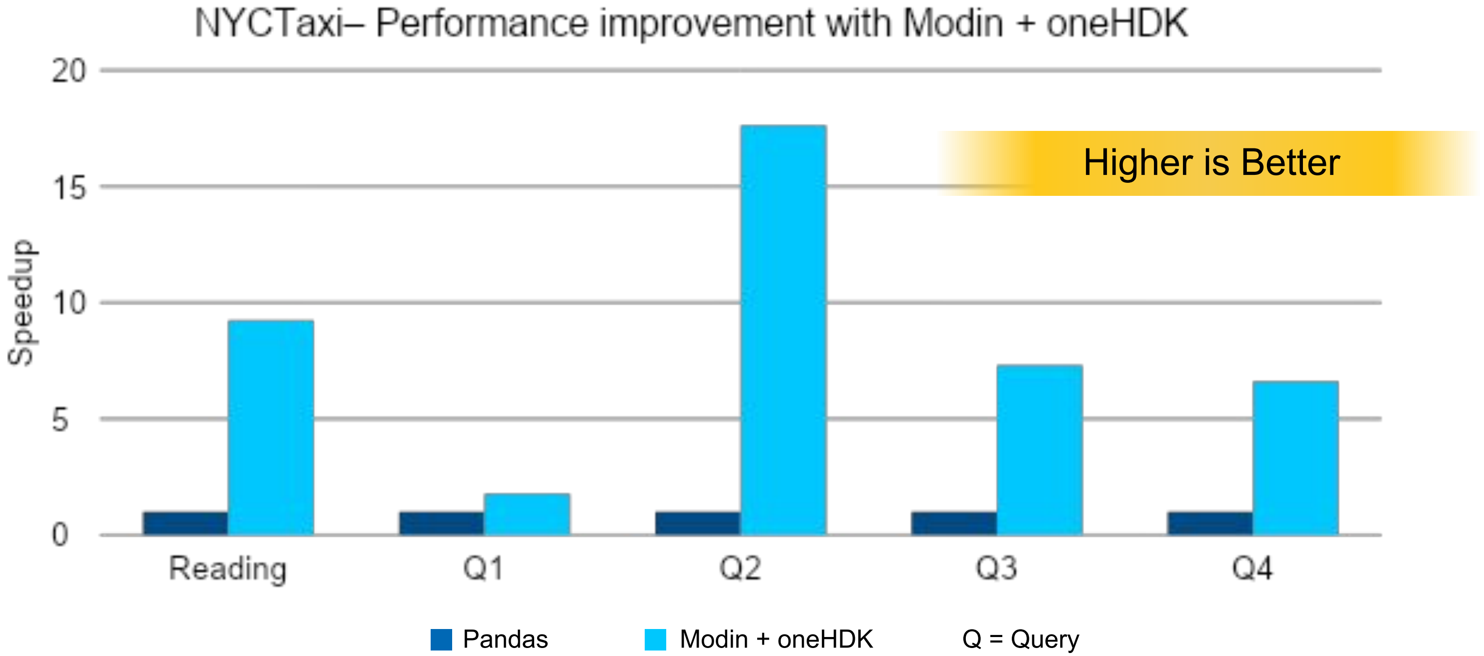
Dask | Ray | MPI | Multiprocessing

Middleware



NYCTaxi Workload Performance

Pandas* vs. Modin*



Dataset source: https://github.com/toddwschneider/nyc-taxi-data

Configurations: For 20 million rows: Dual socket Intel(R) Xeon(R) Platinum 8280L CPUs (S2600WFT platform), 28 cores per socket, hyperthreading enabled, turbo mode enabled, NUMA nodes per socket=2, BIOS: SE5C620.86B.02.01.0013.121520200651, kernel: 5.4.0-65-generic, microcode: 0x4003003, OS: Ubuntu 20.04.1 LTS, CPU governor: performance, transparent huge pages: enabled, System DDR Mem Config: slots / cap / speed: 12 slots / 32GB / 2933MHz, total memory per node: 384 GB DDR RAM, boot drive: INTEL SSDSC2BB800G7. For 1 billion rows: Dual socket Intel Xeon Platinum 8260M CPU, 24 cores per socket, 2.40GHz base frequency, DRAM memory: 384 GB 12x32GB DDR4 Samsung @ 2666 MT/s 1.2V, Optane memory: 3TB 12x256GB Intel Optane @ 2666MT/s, kernel: 4.15.0-91-generic, OS: Ubuntu 20.04.4

A Closer Look:

Intel Extension for Scikit-Learn* and XGBoost Optimizations

intel®

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/oneAPI/oneDAL

What's New in the oneDAL Release

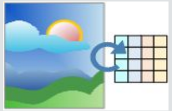
New GPU support for the following Algorithms:

- **Statistical:** Correlation, Low-order moments*
- **Classification:** Linear Regression*, Logistic Regression*, KNN, SVM
- **Unsupervised Learning:** K-means clustering, DBSCAN
- **Classification & Regression:** Random Forest
- **Dimensionality Reduction:** PCA

Pre-processi
ng

Decompression,
Filtering,
Normalization

Transformati
on



Aggregation,
Dimension Reduction

Analysis

Summary
Statistics
Clustering, etc.

Modeling

Machine Learning (Training)
Parameter Estimation
Simulation

Validation

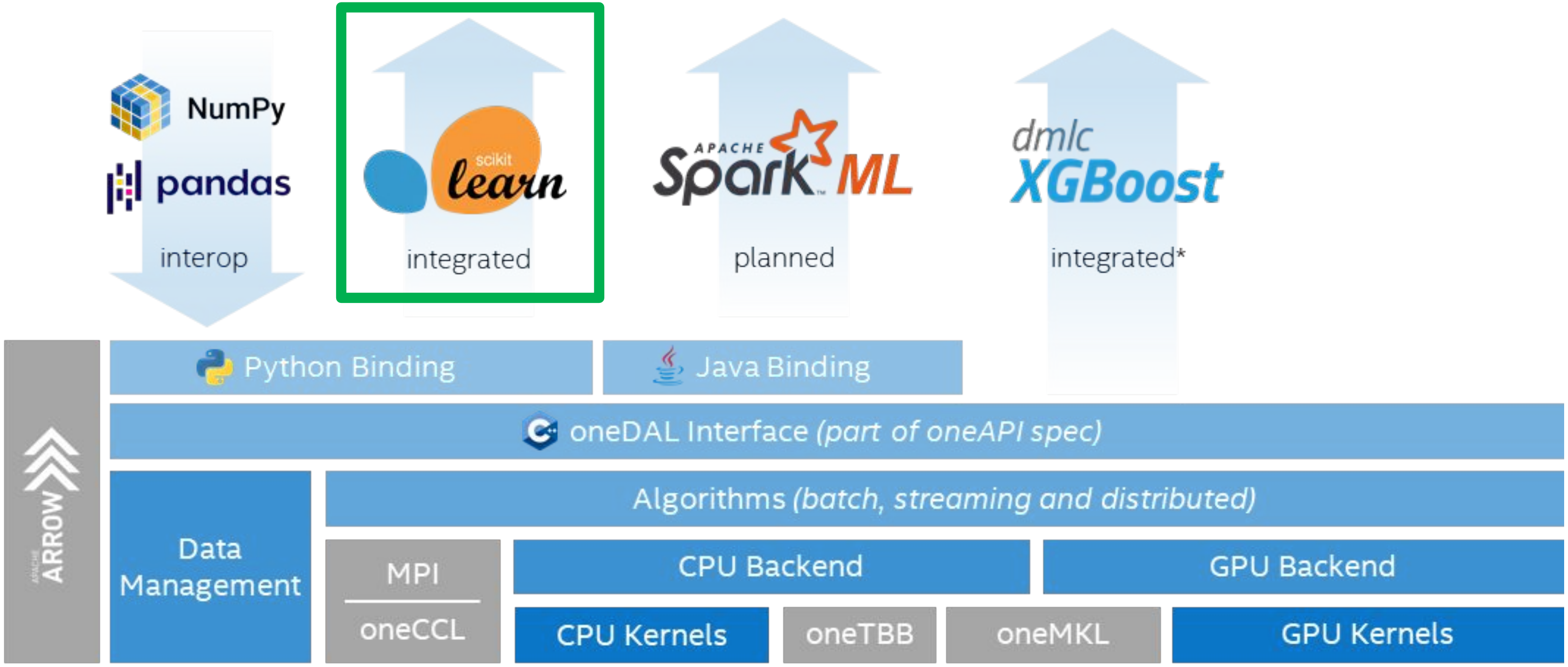
Hypothesis Testing
Model Errors

Decision
Making

Forecasting
Decision Trees, etc.

oneAPI Data Analytics Library (oneDAL)

Optimized building blocks for all stages of data analytics on Intel Architecture



GitHub: https://github.com/oneapi-src/oneDAL

Intel Extension for Scikit-learn*

Common Scikit-learn

```
from sklearn.svm import SVC

X, Y = get_dataset()


clf = SVC().fit(X, y)
res = clf.predict(X)
```

Scikit-learn mainline

Scikit-learn with Intel CPU opts

```
from sklearnex import patch_sklearn
patch_sklearn()

from sklearn.svm import SVC

X, Y = get_dataset()


clf = SVC().fit(X, y)
res = clf.predict(X)
```

Available through:

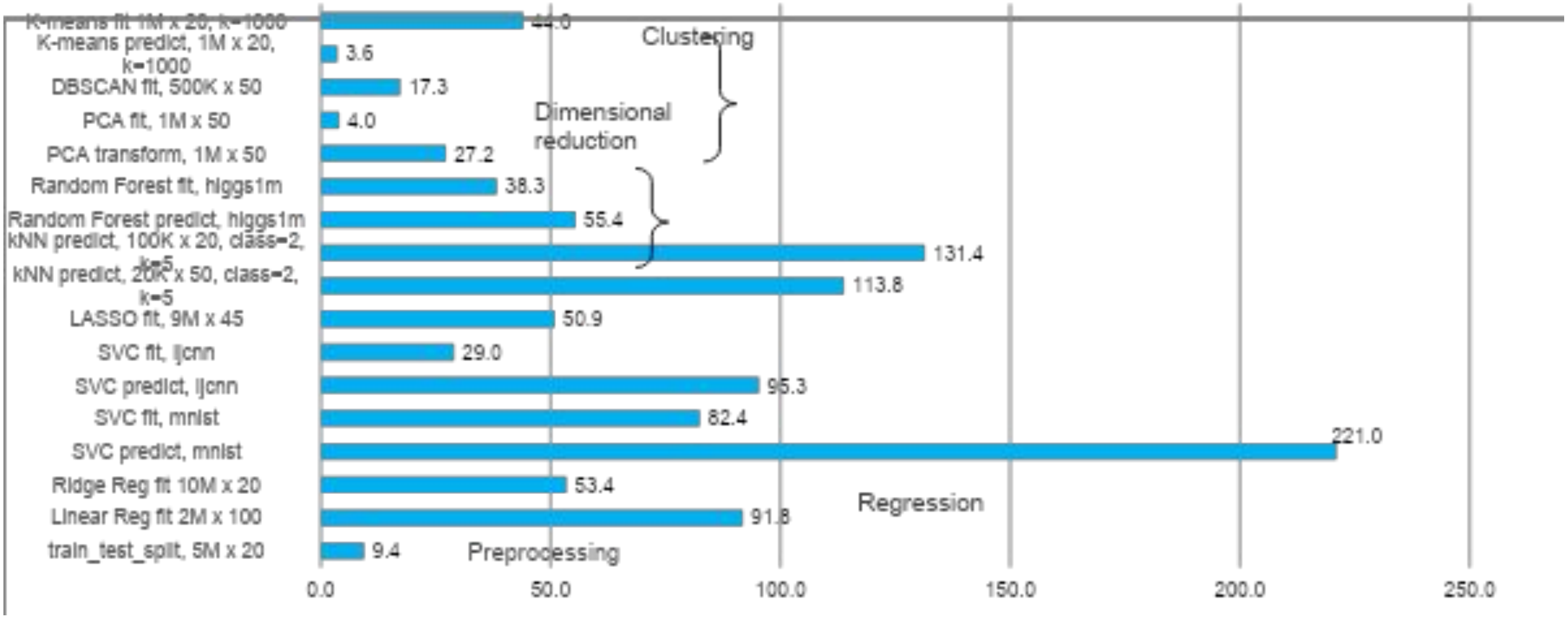
- conda install scikit-learn-intelex
- conda install –c intel scikit-learn-intelex
- conda install –c conda-forge scikit-learn-intelex
- pip install scikit-learn-intelex

Same Code, Same Behavior

PASSED

- Scikit-learn, not scikit-learn-*like*
- Scikit-learn conformance (mathematical equivalence) defined by Scikit-learn Consortium, continuously vetted by public CI

Speedup with oneDAL-powered Scikit-learn over stock Scikit-learn – higher is better



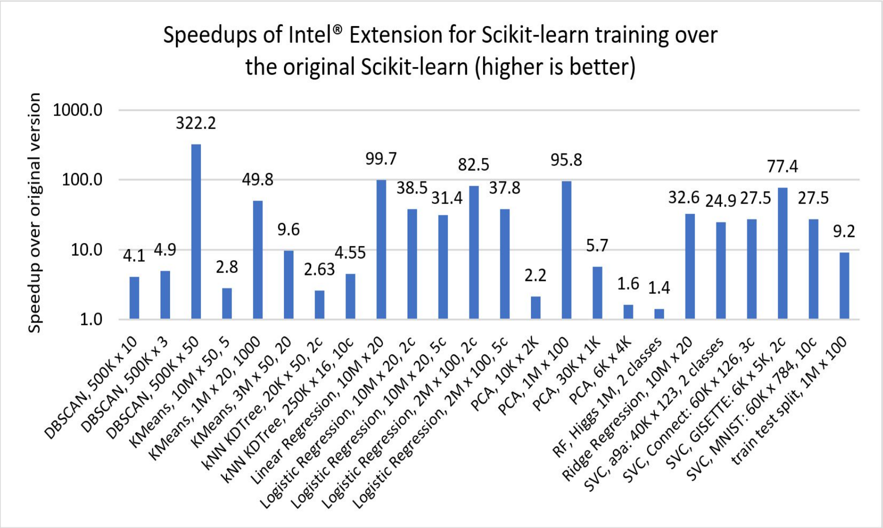
Testing Data: Performance results are based on testing by Intel as of October 23, 2020 and may not reflect all publicly available security updates

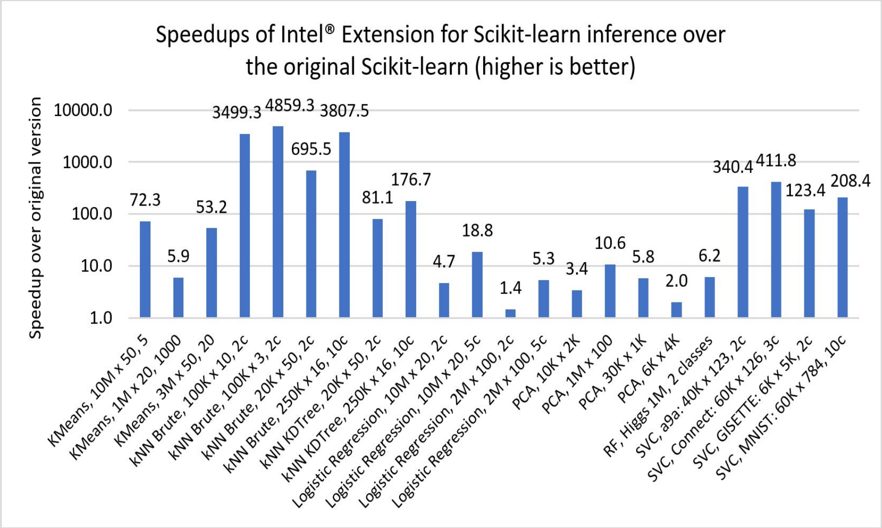
Configurations Details and Workload Setup: Intel® oneAPI Data Analytics Libraru 2021.1(oneDAL). Scikit-Learn* 0.23.1, Intel® Distribution for Python* 3.8; Intel® Xeon® Platinum 8280LCPU@2.70GHz, 2 sockets, 28 cores per socket, 10M samples, 10 features, 100 clusters, 100 iterations, float32

Intel technologies may require enabled hardware, software or service activation. No product or component can be absolutely secure.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available options. Learn more at www.Intel.com/PerformanceIndex.

Intel® Extension for Scikit-Learn Performance on CLX compared to original Scikit-Learn (Training & Inference)

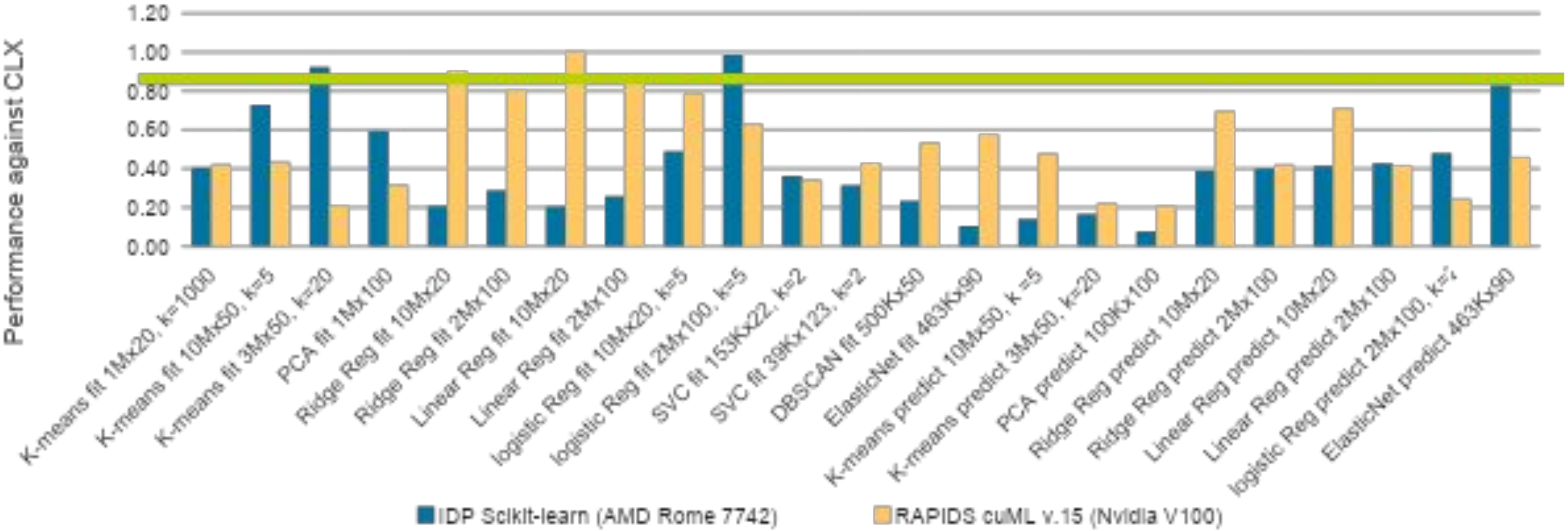




Testing Date: Performance results are based on testing by Intel as of June 8, 2021 and may not reflect all publicly available security updates.
Configuration Details and Workload Setup: c5.24xlarge AWS EC2 (3.0 GHz Intel Xeon Platinum 8275CL, two sockets, 24 cores per socket) Python 3.8, scikit-learn 0.24.2, scikit-learn-intelex 2021.2.3.
Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See configuration disclosure for details. Not product or component can be absolutely secure.
Performance varies by use, configuration, and other factors. Learn more at www.intel.com/PerformanceIndex. Your costs and results may vary

Competitor's Relative Performance vs. Intel® Distribution for Python (IDP) with Scikit-learn from the Intel® AI Analytics Toolkit

(Intel = 1)



Testing Date: Performance results are **based on testing by Intel as of** October 23, 2020 and may not reflect all publicly available security updates.

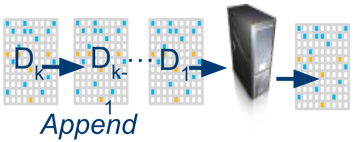
Configuration Details and Workload Setup: Intel® oneDAL beta10, Scikit-learn 0.23.1, Intel® Distribution for Python 3.7, Intel® AI Analytics Toolkit 2021.1, Intel(R) Xeon(R) Platinum 8280 CPU @ 2.70GHz, 2 sockets, 28 cores per socket, microcode: 0x4003003, total available memory 376 GB, 12X32GB modules, DDR4. AMD Configuration: AMD Rome 7742 @2.25 GHz, 2 sockets, 64 cores per socket, microcode: 0x8301038, total available memory 512 GB, 16X32GB modules, DDR4, Intel® oneDAL beta10, Scikit-learn 0.23.1, Intel® Distribution for Python 3.7. NVIDIA Configuration: Nvidia Tesla V100-16Gb, total available memory 376 GB, 12X32GB modules, DDR4, Intel(R) Xeon(R) Platinum 8280 CPU @ 2.70GHz, 2 sockets, 28 cores per socket, microcode: 0x5003003, cuDF 0.15, cuML 0.15, CUDA 10.2.89, driver 440.33.01, Operation System: CentOS Linux 7 (Core), Linux 4.19.36 kernel.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.

Performance varies by use, configuration, and other factors. Learn more at www.Intel.com/PerformanceIndex. Your costs and results may vary.

Processing Modes

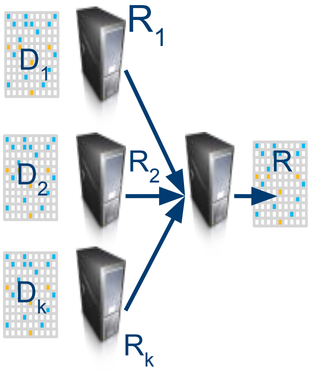
Batch Processing



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

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

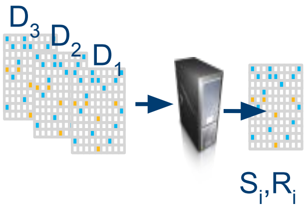
Distributed Processing



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

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

Online Processing

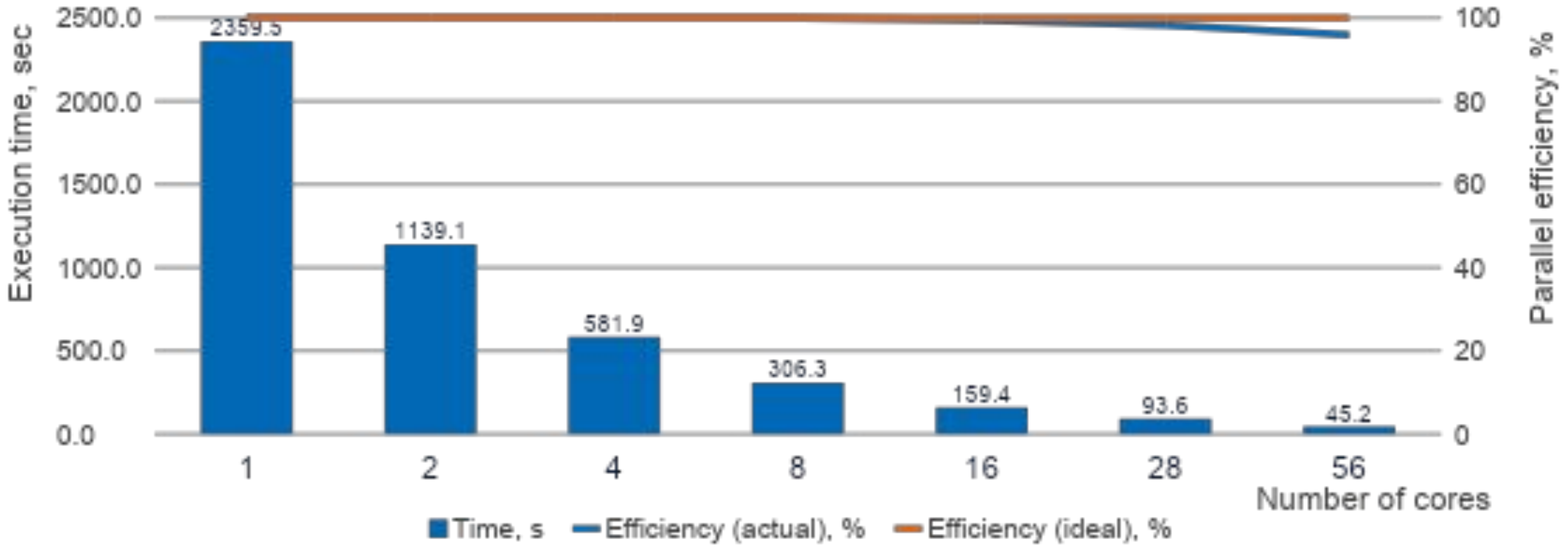


$S_{i+1} = T(S_i,D_i)$
$R_{i+1} = F(S_{i+1})$

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

oneDAL K-Means Fit, Cores Scaling

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



Performance varies by use, configuration, and other factors. Learn more at www.intel.com/PerformanceIndex.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.

Your costs and results may vary. Intel technologies may require enabled hardware, software, or service activation.

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 10/23/2020. Intel® oneAPI Data Analytics Library 2021.1 (oneDAL); Intel® Xeon® Platinum 8280LCPU @ 2.70GHz, 2 sockets, 28 cores per socket, 10M samples, 10 features, 100 clusters, 100 iterations, float32.

Gradient Boosting Acceleration – gain sources

Pseudocode for XGBoost* (0.81) implementation

```
def ComputeHist(node):
  hist = []
  for i in samples:
    for f in features:
      bin = bin_matrix[i][f]
      hist[bin].g += g[i]
      hist[bin].h += h[i]
  return hist

def BuildLvl:
  for node in nodes:
    ComputeHist(node)

  for node in nodes:
    for f in features:
      FindBestSplit(node, f)

  for node in nodes:
    SamplePartition(node)
```

Memory prefetching to mitigate

irregular memory access

Usage uint8 instead of uint32

SIMD instructions instead of scalar code

Nested parallelism

Advanced parallelism, reducing seq loops

Usage of AVX-512, vcompress instruction (from Skylake)

Pseudocode for Intel® oneDAL implementation

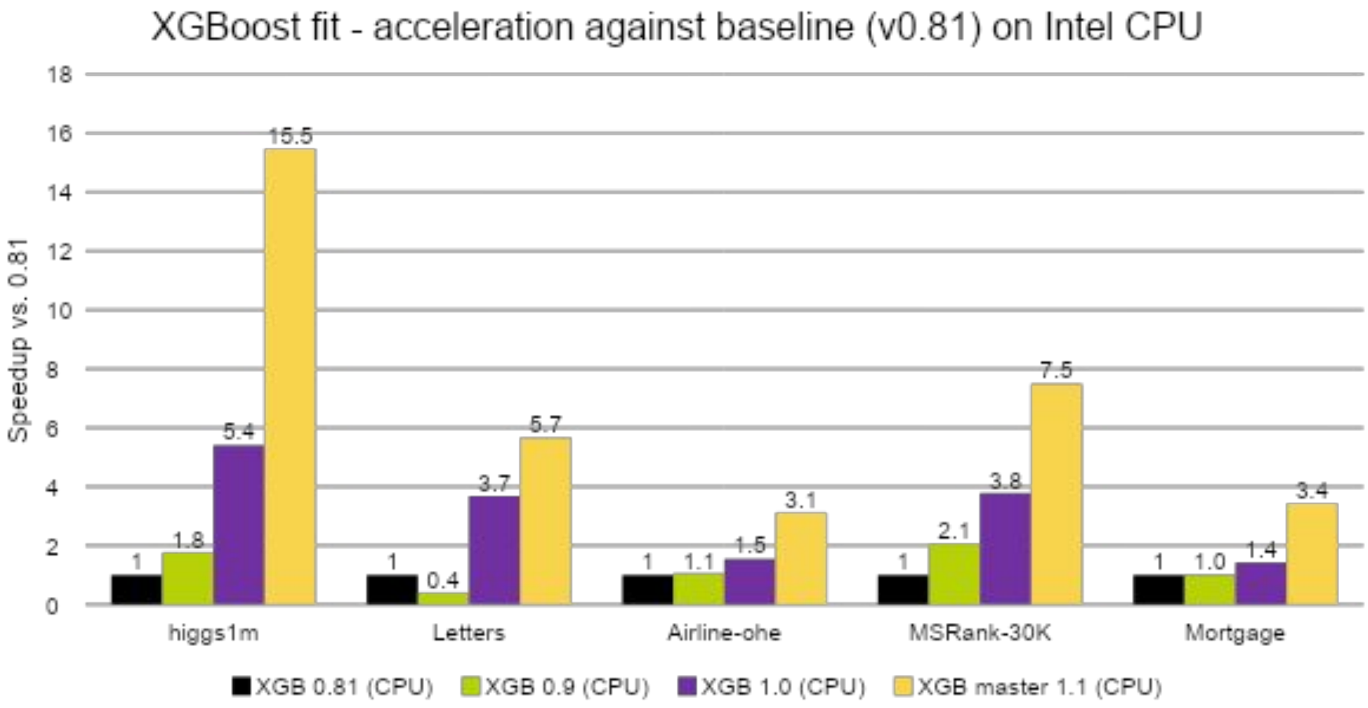
```
def ComputeHist(node):
  hist = []
  for i in samples:
    prefetch(bin_matrix[i + 10])
    for f in features:
      bin = bin_matrix[i][f]
      bin_value = load(hist[2*bin])
      bin_value = add(bin_value, gh[i])
      store(hist[2*bin], bin_value)
  return hist

def BuildLvl:
  parallel_for node in nodes:
    ComputeHist(node)

  parallel_for node in nodes:
    for f in features:
      FindBestSplit(node, f)

  parallel_for node in nodes:
    SamplePartition(node)
```

Training stage

Legend:

Moved from Intel® oneDAL to XGBoost (v1.3)	Already available in Intel® DAAL, potential optimizations for XGBoost*

XGBoost* fit CPU acceleration ("hist" method)

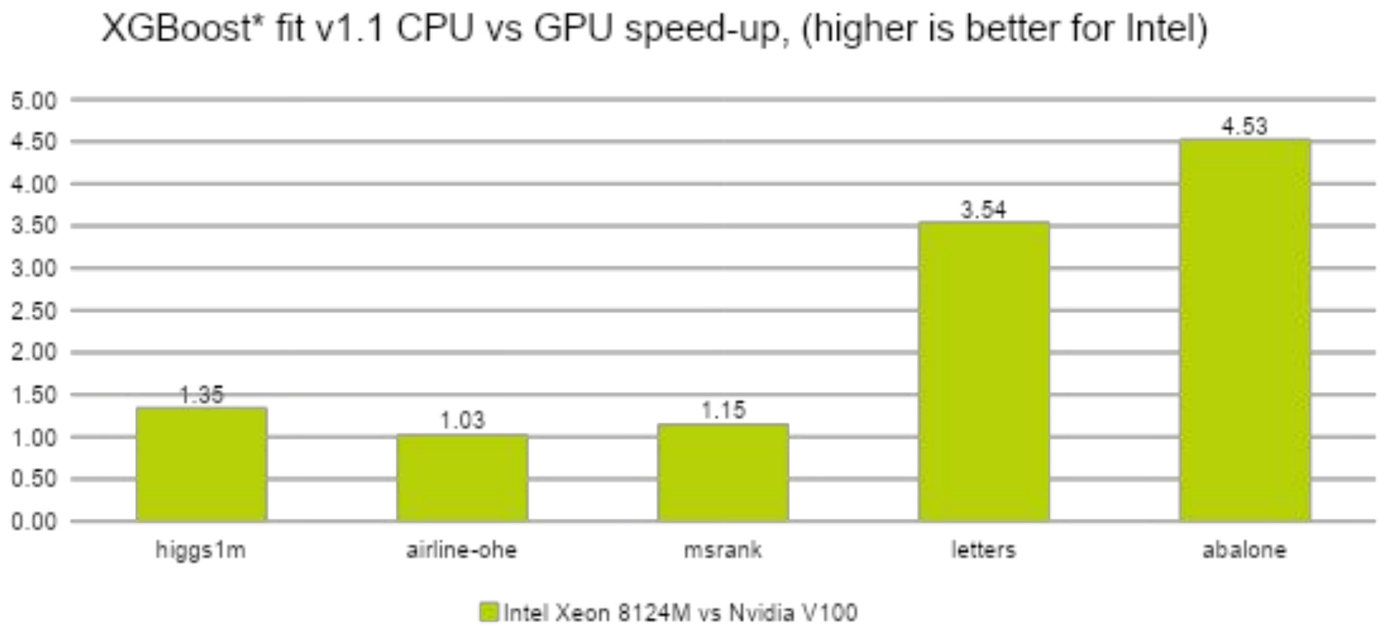


+ Reducing memory consumption

memory, Kb	Airline	Higgs1m
Before	28311860	1907812
#5334	16218404	1155156
reduced:	1.75	1.65

CPU configuration: c5.24xlarge AWS Instance, CLX 8275 @ 3.0GHz, 2 sockets, 24 cores per socket, HT:on, DRAM (12 slots / 32GB / 2933 MHz)

XGBoost* CPU vs. GPU



Details: https://medium.com/intel-analytics-software/new-optimizations-for-cpu-in-xgboost-1-1-81144ea21115
CPU: c5.18xlarge AWS Instance (2 x Intel® Xeon Platinum 8124M @ 18 cores, OS: Ubuntu 20.04.2 LTS, 193 GB RAM.
GPU: p3.2xlarge AWS Instance (GPU: NVIDIA Tesla V100 16GB, 8 vCPUs), OS: Ubuntu 18.04.2 LTS, 61 GB RAM.
SW: XGBoost 1.1:build from sources. compiler – G++ 7.4, nvcc 9.1. Intel DAAL: 2019.4 version, downloaded from conda. Python env: Python 3.6, Numpy 1.16.4, Pandas 0.25, Scikit-lean 0.21.2.
Testing Date: 5/18/2020

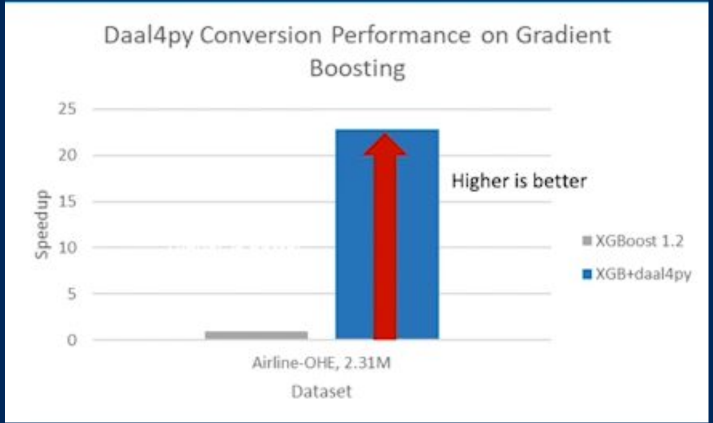
XGBoost* and LightGBM* Prediction Acceleration with Daal4Py

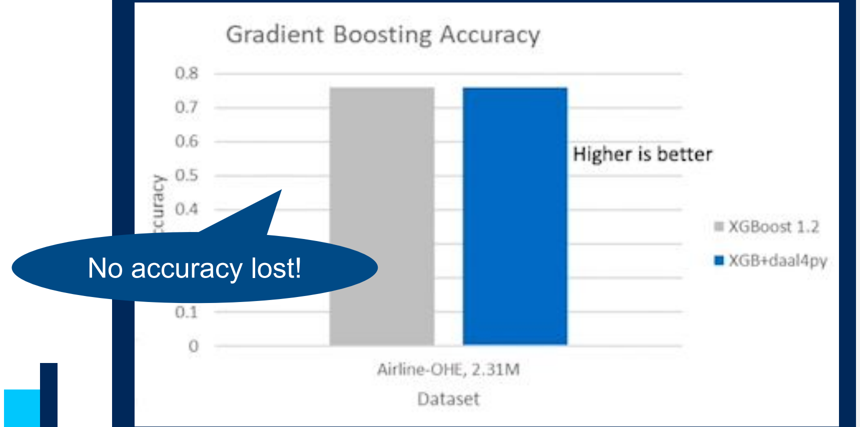
- Custom-trained XGBoost* and LightGBM* Models utilize Gradient Boosting Tree (GBT) from Daal4Py library for performance on CPUs

- No accuracy loss; 23x performance boost by simple model conversion into daal4py GBT:

```
# Train common XGBoost model as usual
xgb_model = xgb.train(params, X_train)

import daal4py as d4p

# XGBoost model to DAAL model
daal_model = d4p.get_gbt_model_from_xgboost(xgb_model)

# make fast prediction with DAAL
daal_prediction = d4p.gbt_classification_prediction(…).compute(X_test, daal_model)
```

- Advantages of daal4py GBT model:
  - More efficient model representation in memory
  - Avx512 instruction set usage
  - Better L1/L2 caches locality





For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.
See backup for configuration details.

Call to Action

Download oneAPI Toolkits for free

[Intel® AI Analytics Toolkit]

For more details on Intel oneAPI, visit

software.intel.com/oneapi

https://devcloud.intel.com/oneapi/

AI Analytics Toolkit Support Forum

For more details on specific AI Kit optimizations, visit

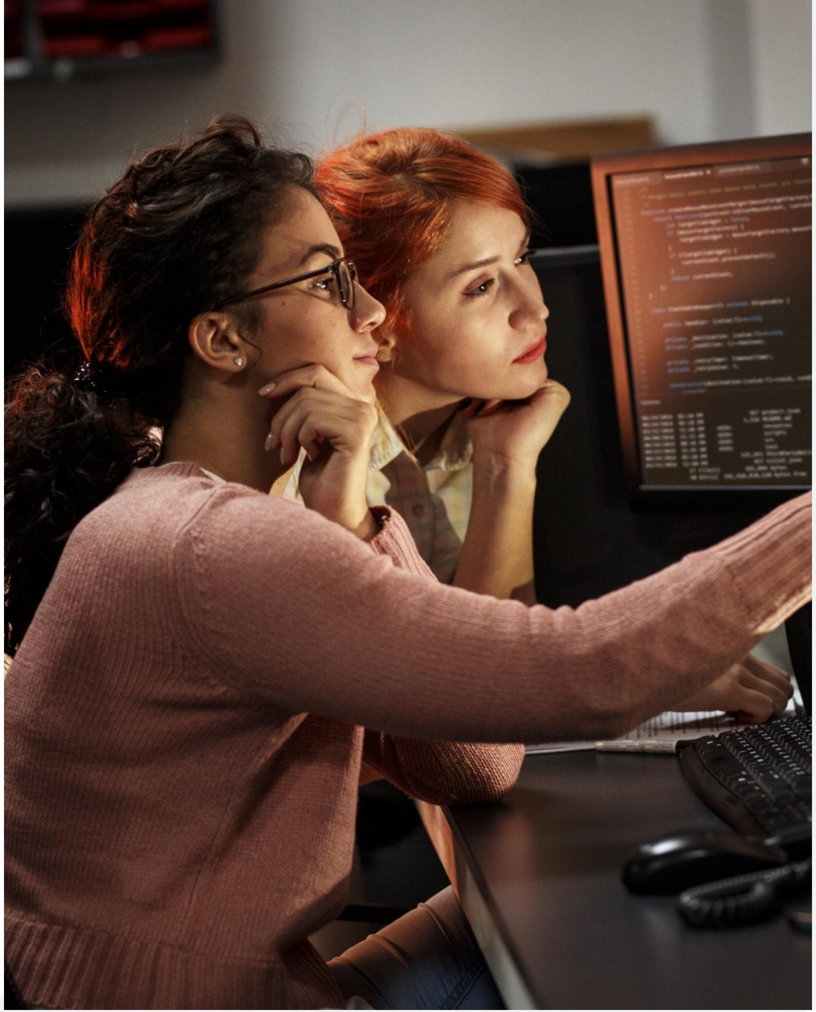
Intel oneContainer Portal

Intel® AWS Containers

Intel® AI Analytics Toolkit Code Samples

Intel® Distribution for Python Support Forum

Machine Learning and Data Analytics Support Forum



Exercises

- Intel Modin - Getting Started
- Intel Extension for SKLearn - Getting Started
- Intel XGBoost - Getting Started

Notices and Disclaimers

- Performance varies by use, configuration and other factors.
  Learn more at www.Intel.com/PerformanceIndex.
- Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.
- Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.
- Your costs and results may vary.
- Intel technologies may require enabled hardware, software or service activation.
-

Workloads and Configurations

See all benchmarks and configurations: https://software.intel.com/content/www/us/en/develop/articles/blazing-fast-python-data-science-ai-performance.html. Each performance claim and configuration data is available in the body of the article listed under sections 1, 2, 3, 4, and 5. Please also visit this page for more details on all scores, and measurements derived.

Testing Date: Performance results are based on testing by Intel as of October 16, 2020 and may not reflect all publicly available updates. **Configurations details and Workload Setup**: 2 x Intel® Xeon® Platinum 8280 @ 28 cores, OS: Ubuntu 19.10.5.3.0-64-generic Mitigated 384GB RAM (192 GB RAM (12x 32GB 2933). SW: Modin 0.81. Scikit-learn 0.22.2. Pandas 1.01, Python 3.8.5, DAL(DAAL4Py) 2020.2, Census Data, (21721922.45) Dataset is from IPUMS USA, University of Minnesota, www.ipums.org [Steven Ruggles, Sarah Flood, Ronald Goeken, Josiah Grover, Erin Meyer, Jose Pacas and Matthew Sobek. IPUMS USA: Version 10.0 [dataset], Minneapolis, MN. IPUMS, 2020. https//doc.org/10.18128/D010.V10.0]

Testing Date: Performance results are based on testing by Intel® as of October 23, 2020 and may not reflect all publicly available updates. **Configuration Details and Workload Setup**: Intel® oneAPI Data Analytics Library 2021.1 (oneDAL). Scikit-learn 0.23.1, Intel® Distribution for Python 3.8; Intel® Xeon® Platinum 8280LCPU @ 270GHz, 2 sockets, 28 cores per socket, 10M samples, 10 features, 100 clusters, 100 iterations, float32.

Testing Date: Performance results are based on testing by Intel® as of October 23, 2020 and may not reflect all publicly available updates. **Configuration Details and Workload Setup**: Intel® AI Analytics Toolkit v2021.1; Intel® oneAPI Data Analytics Library (oneDAL) beta10, Scikit-learn 0.23.1, Intel® Distribution for Python 3.7, Intel® Xeon® Platinum 8280 CPU @ 2.70GHz, 2 sockets, 28 cores per socket, microcode: 0x4003003, total available memory 376 GB, 12X32GB modules, DDR4. **AMD Configuration**: AMD Rome 7742 @2.25 GHz, 2 sockets, 64 cores per socket, microcode: 0x8301038, total available memory 512 GB, 16X32GB modules, DDR4, oneDAL beta10, Scikit-learn 0.23.1, Intel® Distribution for Python 3.7. **NVIDIA Configuration**: NVIDIA Tesla V100 – 16 Gb, total available memory 376 GB, 12X32GB modules, DDR4, Intel® Xeon Platinum 8280 CPU @ 2.70GHz, 2 sockets, 28 cores per socket, microcode: 0x5003003, cuDF 0.15, cuML 0.15, CUDA 10.2.89, driver 440.33.01, Operation System: CentOS Linux 7 (Core), Linux 4.19.36 kernel.

Testing Date: Performance results are based on testing by Intel® as of October 13, 2020 and may not reflect all publicly available updates. **Configurations details and Workload Setup**: CPU: c5.18xlarge AWS Instance (2 x Intel® Xeon® Platinum 8124M @ 18 cores. OS: Ubuntu 20.04.2 LTS, 193 GB RAM. GPU: p3.2xlarge AWS Instance (GPU: NVIDIA Tesla V100 16GB, 8 vCPUs, OS: Ubuntu 18.04.2LTS, 61 GB RAM. SW: XGBoost 1.1: build from sources compiler – G++ 7.4, nvcc 9.1 Intel® DAAL: 2019.4 version: Python env: Python 3.6, Numpy 1.16.4, Pandas 0.25 Scikit-learn 0.21.2.

Workloads and Configurations

Testing Date: Performance results are based on testing by Intel® as of October 26, 2020 and may not reflect all publicly available updates. **Configuration Details and Workload Setup**: Intel® Optimization for Tensorflow v2.2.0; oneDNN v1.2.0; Intel® Low Precision Optimization Tool v1.0; Platform; Intel® Xeon® Platinum 8280 CPU; #Nodes 1; #Sockets: 2; Cores/socket: 28; Threads/socket: 56; HT: On; Turbo: On; BIOS version:SE5C620.86B.02.01.0010.010620200716; System DDR Mem Config: 12 slots/16GB/2933; OS: CentOS Linux 7.8; Kernel: 4.4.240-1.el7.elrepo x86_64.

Testing Date: Performance results are based on testing by Intel® as of February 3, 2021 and may not reflect all publicly available updates. **Configuration Details and Workload Setup:** Intel® Optimization for PyTorch v1.5.0; Intel® Extension for PyTorch (IPEX) 1.1.0; oneDNN version: v1.5; DLRM: Training batch size (FP32/BF16): 2K/instance, 1 instance; DLRM dataset (FP32/BF16): Criteo Terabyte Dataset; BERT-Large: Training batch size (FP32/BF16): 24/Instance. 1 Instance on a CPU socket. Dataset (FP32/BF16): WikiText-2 [https://www.salesforce.com/products/einstein/ai-research/the-wiktext-dependency-language-modeling-dataset/]: ResNext101-32x4d: Training batch size (FP32/BF16): 128/Instance, 1 instance on a CPU socket, Dataset (FP32/BF16): ILSVRC2012; DLRM: Inference batch size (INT8): 16/instance, 28 instances, dummy data. Intel® Xeon® Platinum 8380H Processor, 4 socket, 28 cores HT On Turbo ON Total memory 768 GB (24 slots/32GB/3200 MHz), BIOS; WLYDCRBLSYS.0015.P96.2005070242 (ucode: OX 700001b), Ubuntu 20.04 LTS, kernel 5.4.0-29-genen: ResNet50: [https://github.com/Intel/optimized-models/tree/master/pytorch/ResNet50]: ResNext101 32x4d: [https://github.com/intel/optimized-models/tree/master/pytorch/ResNext101_32x4ct: DLRM: https//github.com/intel/optimized-models/tree/master/pytorch/dlrm].

Configuration Details

NYCTaxi Workload performance:

For 20 million rows: Dual socket Intel(R) Xeon(R) Platinum 8280L CPUs (S2600WFT platform), 28 cores per socket, hyperthreading enabled, turbo mode enabled, NUMA nodes per socket=2, BIOS: SE5C620.86B.02.01.0013.121520200651, kernel: 5.4.0-65-generic, microcode: 0x4003003, OS: Ubuntu 20.04.1 LTS, CPU governor: performance, transparent huge pages: enabled, System DDR Mem Config: slots / cap / speed: 12 slots / 32GB / 2933MHz, total memory per node: 384 GB DDR RAM, boot drive: INTEL SSDSC2BB800G7. For 1 billion rows: Dual socket Intel Xeon Platinum 8260M CPU, 24 cores per socket, 2.40GHz base frequency, DRAM memory: 384 GB 12x32GB DDR4 Samsung @ 2666 MT/s 1.2V, Optane memory: 3TB 12x256GB Intel Optane @ 2666MT/s, kernel: 4.15.0-91-generic, OS: Ubuntu 20.04.4

End-to-End Census Workload performance (Stock):

Tested by Intel as of 2/19/2021. 2 x Intel® Xeon Platinum 8280L @ 28 cores, OS: Ubuntu 20.04.1 LTS Mitigated, 384GB RAM (384GB RAM: 12x 32GB 2933MHz), kernel: 5.4.0-65-generic, microcode: 0x4003003, CPU governor: performance. SW: Scikit-learn 0.24.1, Pandas 1.2.2, Python 3.9.7, Census Data, (21721922, 45) Dataset is from IPUMS USA, University of Minnesota, www.ipums.org [*Steven Ruggles, Sarah Flood, Ronald Goeken, Josiah Grover, Erin Meyer, Jose Pacas and Matthew Sobek. IPUMS USA: Version 10.0 [dataset]. Minneapolis, MN: IPUMS, 2020. https://doi.org/10.18128/D010.V10.0]*

End-to-End Census Workload performance (Optimized):

Tested by Intel as of 2/19/2021. 2 x Intel® Xeon Platinum 8280L @ 28 cores, OS: Ubuntu 20.04.1 LTS Mitigated, 384GB RAM (384GB RAM: 12x 32GB 2933MHz), kernel: 5.4.0-65-generic, microcode: 0x4003003, CPU governor: performance. SW: Scikit-learn 0.24.1 accelerated by daal4py (now sklearnex) 2021.2, modin 0.8.3, omniscidbe v5.4.1, Python 3.9.7, Census Data, (21721922, 45) Dataset is from IPUMS USA, University of Minnesota, www.ipums.org [*Steven Ruggles, Sarah Flood, Ronald Goeken, Josiah Grover, Erin Meyer, Jose Pacas and Matthew Sobek. IPUMS USA: Version 10.0 [dataset]. Minneapolis, MN: IPUMS, 2020. https://doi.org/10.18128/D010.V10.0]*