INTEL DISTRIBUTION FOR PYTHON – ADVANTAGES AND ACCELERATION OF MACHINE LEARNING WORKLOADS

Legal Disclaimer and Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos.

Copyright © 2017, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

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
Faster Python* with Intel® Distribution for Python 2018

Optimization Notice

High Performance Python Distribution

- Accelerated NumPy, SciPy, scikit-learn well suited for scientific computing, machine learning & data analytics
- Drop-in replacement for existing Python. No code changes required
- Highly optimized for latest Intel processors
- Take advantage of Priority Support – connect direct to Intel engineers for technical questions

What's New in 2018 version

- Updated to latest version of Python 3.6
- Optimized scikit-learn for machine learning speedups
- Conda build recipes for custom infrastructure

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

Software & workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark & MobileMark, are measured using specific computer systems, components, software, operations & functions. Any change to any of those factors may cause the results to vary. You should consult other information & performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to http://www.intel.com/performance.
**Python* Landscape**

**Challenge#1**
Domain experts are not professional software programmers

**Challenge#2**
Python performance limits migration to production systems

---

**Adoption of Python** continues to grow among domain experts & developers for its productivity benefits

**Intel's Python Tools**
- Accelerate Python performance
- Enable easy access
- Empower the community

---

**Most Popular Coding Languages of 2016**

- Python 26.7%
- Java 22.6%
- JavaScript 17.6%
- C# 9.4%
- C 7.37%
- Ruby 6.9%
- PHP 5.9%
- HTML/CSS 3.8%
- SQL 3.7%
- Objective-C 1.8%

---

**Optimization Notice**
Copyright © 2017, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.*
### What’s Inside Intel® Distribution for Python

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

<table>
<thead>
<tr>
<th>FASTER PERFORMANCE</th>
<th>GREATER PRODUCTIVITY</th>
<th>ECOSYSTEM COMPATIBILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Libraries, Parallelism, Multithreading, Language Extensions</td>
<td>Prebuilt &amp; Accelerated Packages</td>
<td>Supports Python 2.7 &amp; 3.6, conda, pip</td>
</tr>
<tr>
<td>Accelerated NumPy/SciPy/scikit-learn with Intel® MKL¹ &amp; Intel® DAAL²</td>
<td>Prebuilt &amp; optimized packages for numerical computing, machine/deep learning, HPC, &amp; data analytics</td>
<td>Compatible &amp; powered by Anaconda*, supports conda &amp; pip</td>
</tr>
<tr>
<td>Data analytics, machine learning &amp; deep learning with scikit-learn, pyDAAL</td>
<td>Drop in replacement for existing Python - No code changes required</td>
<td>Distribution &amp; individual optimized packages also available at conda &amp; Anaconda.org, YUM/APT, Docker image on DockerHub</td>
</tr>
<tr>
<td>Scale with Numba* &amp; Cython*</td>
<td>Jupyter* notebooks, Matplotlib included</td>
<td>Optimizations upstreamed to main Python trunk</td>
</tr>
<tr>
<td>Includes optimized mpi4py, works with Dask* &amp; PySpark*</td>
<td>Conda build recipes included in packages</td>
<td>Commercial support through Intel® Parallel Studio XE 2017</td>
</tr>
<tr>
<td>Optimized for latest Intel® architecture</td>
<td>Free download &amp; free for all uses including commercial deployment</td>
<td></td>
</tr>
</tbody>
</table>

**Intel® Architecture Platforms**

Operating System: Windows*, Linux*, MacOS¹

---

¹Intel® Math Kernel Library
²Intel® Data Analytics Acceleration Library

---

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

---

¹ Available only in Intel® Parallel Studio Composer Edition.
UMath Optimizations & Vectorization to Utilize Multiple Cores, Memory Management

**Intel® Core™ i7 Processor**

Intel® Distribution for Python® Performance Speedups for Select Math Functions on Intel® Core™ i7 Processors

- Speedup with Intel Python vs pip/numpy

**Up to 37X speedups**

<table>
<thead>
<tr>
<th>Math functions (Array size = 1M)</th>
<th>Speedup Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>array-array</td>
<td>1.3X</td>
</tr>
<tr>
<td>array-scalar</td>
<td>1.3X</td>
</tr>
<tr>
<td>array*array</td>
<td>1.3X</td>
</tr>
<tr>
<td>array*scalar</td>
<td>1.3X</td>
</tr>
<tr>
<td>array+array</td>
<td>1.3X</td>
</tr>
<tr>
<td>array+scalar</td>
<td>1.3X</td>
</tr>
<tr>
<td>erf</td>
<td>4.3X</td>
</tr>
<tr>
<td>exp</td>
<td>2.0X</td>
</tr>
<tr>
<td>log10</td>
<td>23.6X</td>
</tr>
</tbody>
</table>

Software: Stock: CentOS Linux release 7.3.1611 (Core), python 3.6.2, pip 9.0.1, numpy 1.13.1, scipy 0.18.1, scikit-learn 0.19.0

**Intel® Xeon® Processor**

Intel® Distribution for Python® Performance Speedups for Select Math Functions on Intel® Xeon® Processor

- Speedup with Intel Python vs pip/numpy

**Up to 440X speedups**

<table>
<thead>
<tr>
<th>Math functions (Array size = 1M)</th>
<th>Speedup Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>array-array</td>
<td>16X</td>
</tr>
<tr>
<td>array-scalar</td>
<td>17X</td>
</tr>
<tr>
<td>array*array</td>
<td>16X</td>
</tr>
<tr>
<td>array*scalar</td>
<td>17X</td>
</tr>
<tr>
<td>array+array</td>
<td>16X</td>
</tr>
<tr>
<td>array+scalar</td>
<td>17X</td>
</tr>
<tr>
<td>erf</td>
<td>51X</td>
</tr>
<tr>
<td>exp</td>
<td>77X</td>
</tr>
<tr>
<td>log10</td>
<td>442X</td>
</tr>
</tbody>
</table>

Hardware: Intel® Core™ i7-7567U CPU@3.50GHz (1 socket, 2 cores per socket, 2 threads per core), 32GB DDR4 @ 2133MHz, Intel® Xeon® CPU E5-2699 v4@2.00GHz (2 sockets, 22 cores per socket, 1 thread per core-HT is off), 256GB DDR4@2400MHz, Intel® Xeon Phi™ CPU T250@1.40GHz (1 socket, 80 cores per socket, 4 threads per core), 192GB DDR4 @ 1200MHz, 16GB MCDRAM@ 7200MHz in cache mode

Software: Stock: CentOS Linux release 7.3.1611 (Core), python 3.6.2, pip 9.0.1, numpy 1.13.1, scipy 0.18.1, scikit-learn 0.19.0

**Intel® Xeon® Phi™ Processor**

Intel® Distribution for Python® Performance Speedups for Select Math Functions on Intel® Xeon Phi™ Processor Family

- Speedup with Intel Python vs pip/numpy

**Over 3,000X speedups**

<table>
<thead>
<tr>
<th>Math functions (Array size = 1M)</th>
<th>Speedup Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>array-array</td>
<td>6X</td>
</tr>
<tr>
<td>array-scalar</td>
<td>16X</td>
</tr>
<tr>
<td>array*array</td>
<td>6X</td>
</tr>
<tr>
<td>array*scalar</td>
<td>17X</td>
</tr>
<tr>
<td>array+array</td>
<td>6X</td>
</tr>
<tr>
<td>array+scalar</td>
<td>16X</td>
</tr>
<tr>
<td>erf</td>
<td>3209X</td>
</tr>
<tr>
<td>exp</td>
<td>3283X</td>
</tr>
<tr>
<td>log10</td>
<td>3357X</td>
</tr>
</tbody>
</table>

Software & 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 & functions. Any change to any of those factors may cause the results to vary. You should consult other information & performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to http://www.intel.com/Performance.
Performance Speedups for Black Scholes Formula

**Intel® Core™ i7 Processor**

Performance Speedups for Intel® Distribution for Python* for Black Scholes* Formula on Intel® Core™ i7 Processor (Higher is Better)

- **plot/numpy**
- **Intel Python**

Hardware: Intel® Core™ i7-7567U CPU@3.50GHz (1 socket, 2 cores per socket, 2 threads per core), 32GB DDR4 @ 2133MHz. Intel® Xeon® CPU E5-2690 v4@2.20GHz (2 sockets, 22 cores per socket, 1 thread per core-HT is off), 256GB DDR4@2400MHz. Intel® Xeon Phi™ CPU 7250@1.40GHz (1 socket, 68 cores per socket, 4 threads per core), 192GB DDR4@2133MHz, 16GB MCDRAM@7200MHz in cache mode.

Software: Stock: CentOS Linux release 7.3.1611 (Core), python 3.6.2, pip 9.0.1, numpy 1.13.1, scipy 0.19.1, scikit-learn 0.19.0


Intel® Xeon® Phi™ Processor

Performance Speedups for Intel® Distribution for Python* for Black Scholes* Formula on Intel® Xeon Phi™ Processor Family (Higher is Better)

- **plot/numpy**
- **Intel Python**

Software & 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 & functions. Any change to any of those factors may cause the results to vary. You should consult other information & performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to [http://www.intel.com/ ® Performance](http://www.intel.com/®Performance).

**Optimization Notice**

Copyright © 2017, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

*Intel 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.*
Installing Intel® Distribution for Python* 2018

Standalone Installer
Download full installer from https://software.intel.com/en-us/intel-distribution-for-python

Anaconda.org
Anaconda.org/intel channel
> conda config --add channels intel
> conda install intelpython3_full
> conda install intelpython3_core

Docker Hub
docker pull intelpython/intelpython3_full

YUM/APT
Access for yum/apt:

2.7 & 3.6

Linux*  Windows*  OS X*
But Wait.....There’s More!

Outside of optimized Python*, how efficient is your Python/C/C++ application code?

Are there any non-obvious sources of performance loss?

Performance analysis gives the answer!
Tune Python* + Native Code for Better Performance

Analyze Performance with Intel® VTune™ Amplifier (available in Intel® Parallel Studio XE)

Challenge
- Single tool that profiles Python + native mixed code applications
- Detection of inefficient runtime execution

Solution
- Auto-detect mixed Python/C/C++ code & extensions
- Accurately identify performance hotspots at line-level
- Low overhead, attach/detach to running application
- Focus your tuning efforts for most impact on performance

Auto detection & performance analysis of Python & native functions
Diagnose Problem code quickly & accurately

Details Python* calling into native functions

Identifies exact line of code that is a bottleneck
Deeper Analysis for Better Insight

Call Stack Listing for Python* & Native Code

Detailed Time Analysis
A 2-prong approach for Faster Python* Performance

High Performance Python Distribution + Performance Profiling

Step 1: Use Intel® Distribution for Python

- Leverage optimized native libraries for performance
- Drop-in replacement for your current Python - no code changes required
- Optimized for multi-core and latest Intel processors

Step 2: Use Intel® VTune™ Amplifier for profiling

- Get detailed summary of entire application execution profile
- Auto-detects & profiles Python/C/C++ mixed code & extensions with low overhead
- Accurately detect hotspots - line level analysis helps you make smart optimization decisions fast!
- Available in Intel® Parallel Studio XE Professional & Cluster Edition
Intel® Data Analytics Acceleration Library (Intel® DAAL)

- Targets both data centers (Intel® Xeon® and Intel® Xeon Phi™) and edge-devices (Intel® Atom)
- Perform analysis close to data source (sensor/client/server) to optimize response latency, decrease network bandwidth utilization, and maximize security
- Offload data to server/cluster for complex and large-scale analytics

- (De-)Compression
- (De-)Serialization
- PCA
- Statistical moments
- Quantiles
- Variance matrix
- QR, SVD, Cholesky
- Apriori
- Outlier detection
- Regression
  - Linear
  - Ridge
- Classification
  - Naïve Bayes
  - SVM
  - Classifier boosting
  - kNN
- Clustering
  - Kmeans
  - EM GMM
- Collaborative filtering
  - ALS
- Neural Networks
Computational Aspects of Big Data

Volume
- Distributed across different nodes/devices
- Huge data size not fitting into node/device memory

Variety
- Non-homogeneous data
- Sparse/Missing/Noisy data

Velocity
- Data coming in time
Regression

Problems

- A company wants to define the impact of the pricing changes on the number of product sales
- A biologist wants to define the relationships between body size, shape, anatomy and behavior of the organism

Solution: Linear Regression

- A linear model for relationship between features and the response

\[
\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \ldots + \hat{\beta}_N x_N
\]

Source: Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. (2014). An Introduction to Statistical Learning. Springer
Classification

Problems

- An emailing service provider wants to build a spam filter for the customers
- A postal service wants to implement handwritten address interpretation

Solution: Support Vector Machine (SVM)

- Works well for non-linear decision boundary
- Two kernel functions are provided:
  - Linear kernel
  - Gaussian kernel (RBF)
- Multi-class classifier
  - One-vs-One

Source: Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. (2014). *An Introduction to Statistical Learning*. Springer
Cluster Analysis

Problems

- A news provider wants to group the news with similar headlines in the same section
- Humans with similar genetic pattern are grouped together to identify correlation with a specific disease

Solution: K-Means

- Pick $k$ centroids
- Repeat until converge:
  - Assign data points to the closest centroid
  - Re-calculate centroids as the mean of all points in the current cluster
  - Re-assign data points to the closest centroid

Source: Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. (2014). *An Introduction to Statistical Learning*. Springer
Dimensionality Reduction

Problems

- Data scientist wants to visualize a multi-dimensional data set
- A classifier built on the whole data set tends to overfit

Solution: Principal Component Analysis

- Compute eigen decomposition on the correlation matrix
- Apply the largest eigenvectors to compute the largest principal components that can explain most of variance in original data

Source: Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. (2014). *An Introduction to Statistical Learning*. Springer
Demo

https://github.com/IntelPython/BlackScholes_bench
https://github.com/daaltces/pydaal-tutorials
CODE THAT PERFORMS AND OUTPERFORMS

Download a free, 30-day trial of Intel® Parallel Studio XE 2018 today


AND DON’T FORGET...

To check your inbox for the evaluation survey which will be emailed after this presentation.

P.S.

Everyone who fills out the survey will receive a personalized certificate indicating completion of the training!