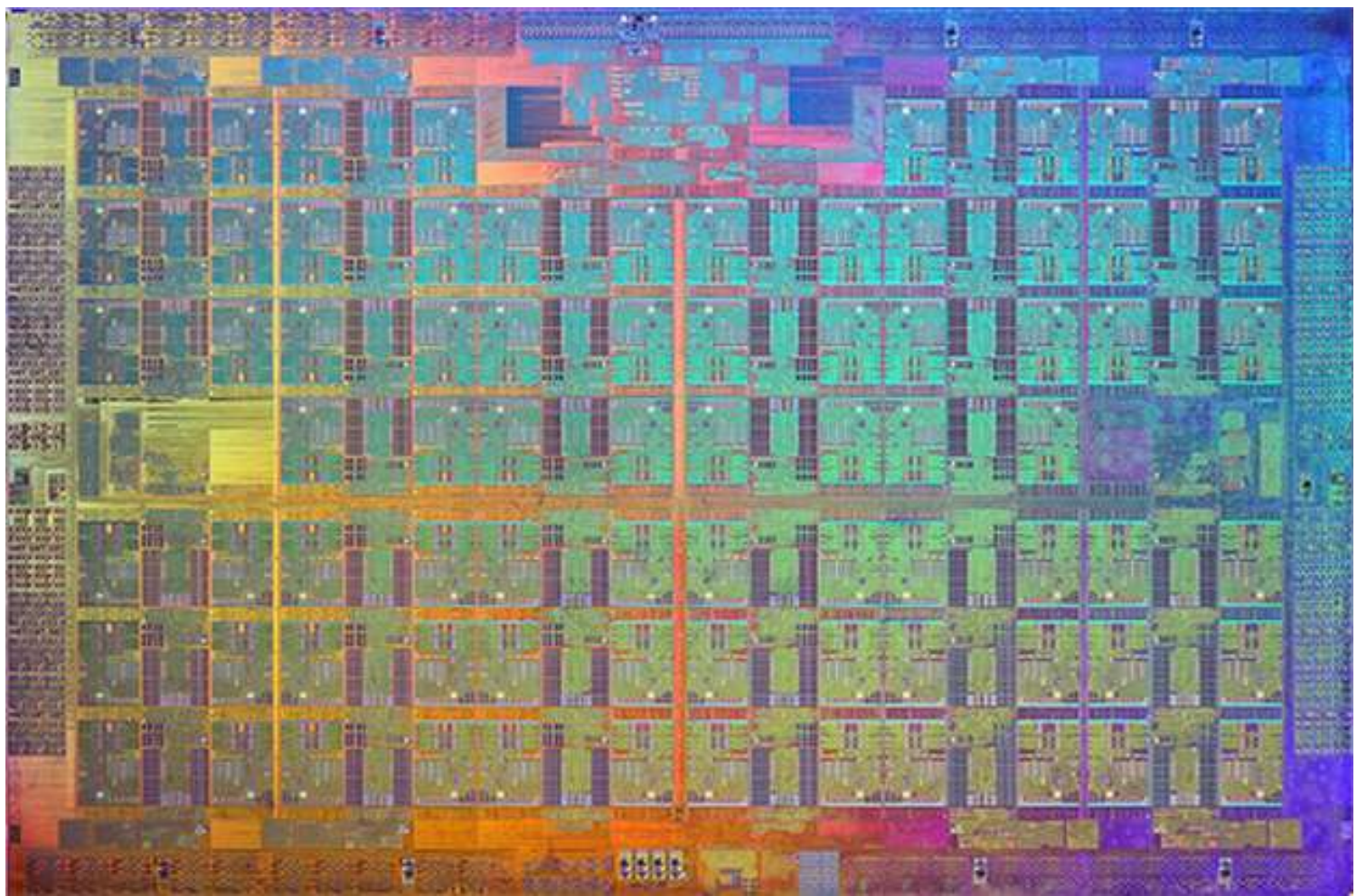




HIGH PERFORMANCE PYTHON ON INTEL[®] ARCHITECTURE

Ralph de Wargny

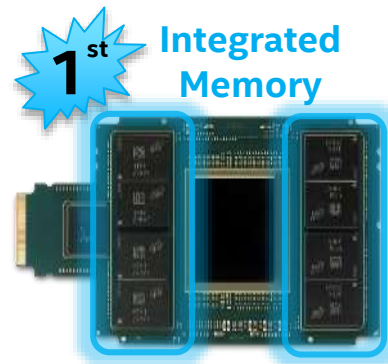
Intel Corp. / Software & Services Group



Optimization Notice

Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Introducing the Intel® Xeon Phi™ Processor



LEADERSHIP PERFORMANCE ... WITH ALL THE BENEFITS OF A CPU

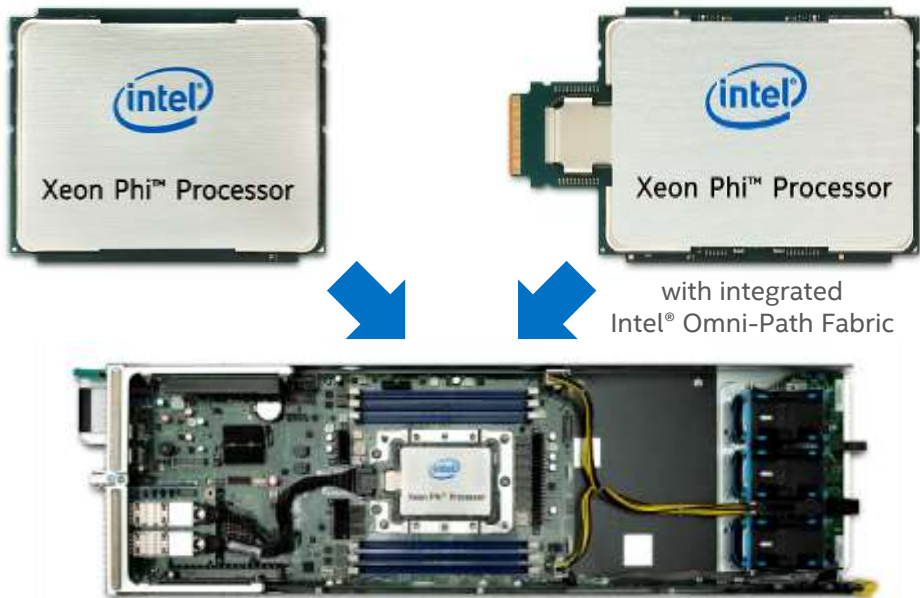
- ✓ Run Any Workload
- ✓ Programmability
- ✓ Power Efficient
- ✓ No PCIe Bottleneck
- ✓ Large Memory Footprint
- ✓ Scalability & Future-Ready

Optimization Notice

Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Intel® Xeon Phi™ Product Family x200

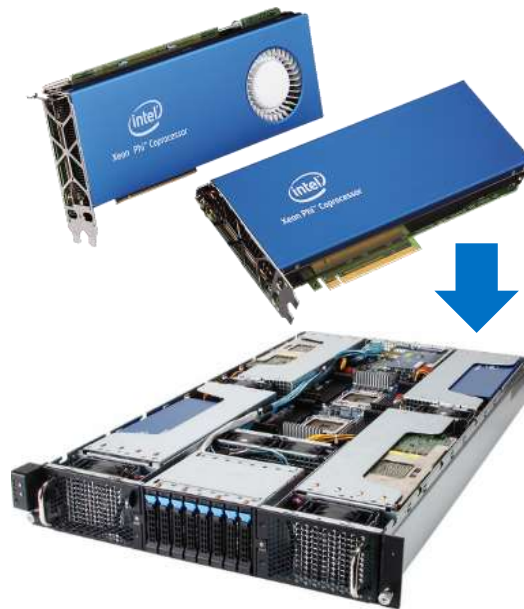
Intel® Xeon Phi™ Processor



Host Processor in Groveport Platform

Self-boot Intel® Xeon Phi™ processor

Intel® Xeon Phi™ Coprocessor x200



Ingredient of Grantley Platforms

Requires Intel® Xeon® processor host

Optimization Notice

Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

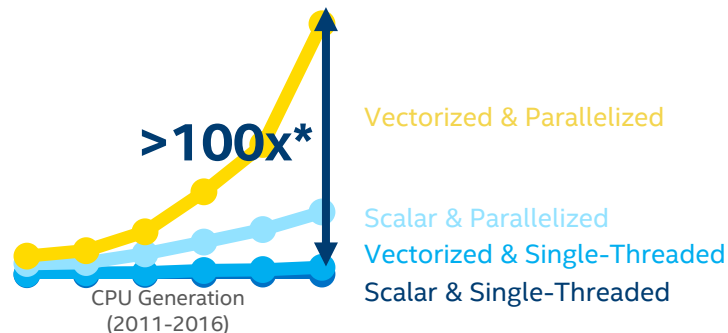




Solve Biggest Challenges Faster

Highly-Parallel

Intel® Xeon® processors are increasingly parallel and require modern code



Intel® Xeon Phi™ processors are extremely parallel and use general purpose programming



Up to 72 cores (288 threads)



Intel® Advanced Vector Extensions 512 (AVX-512)

*Binomial Options DP simulation performed on Intel® Xeon® processor X5570 (formerly codenamed Nehalem), Intel® Xeon® processor x5680 (formerly codenamed Westmere), and Intel® Xeon® processor E5 2600 families v1 through v4 for 4 sets of code with varying levels of vectorization and threading optimization

Optimization Notice

Copyright © 2016, Intel Corporation. All rights reserved.

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



Parallel is the Path Forward

More cores → More Threads → Wider vectors



| | Intel® Xeon® processor 64-bit | Intel® Xeon® processor 5100 series | Intel® Xeon® processor 5500 series | Intel® Xeon® processor 5600 series | Intel® Xeon® processor 5600 v2 series | Intel® Xeon® processor 5600 v3 series | BDW | Intel® Xeon Phi™ x100 coprocessor | Intel® Xeon Phi™ x200 processor & coprocessor |
|---------------|-------------------------------|------------------------------------|------------------------------------|------------------------------------|---------------------------------------|---------------------------------------|-------------|-----------------------------------|---|
| Up to Core(s) | 1 | 2 | 4 | 6 | 12 | 18 | 22 | 57-61 | Up to 72 |
| Up to Threads | 2 | 2 | 8 | 12 | 24 | 36 | 44 | 228-244 | Up to 288 |
| SIMD Width | 128 | 128 | 128 | 128 | 256 | 256 | 256 | 512 | 512 |
| Vector ISA | Intel® SSE3 | Intel® SSE3 | Intel® SSE4.2 | Intel® AVX | Intel® AVX | Intel® AVX2 | Intel® AVX2 | IMCI 512 | Intel® AVX-512 |

Optimization Notice

Copyright © 2016, Intel Corporation. All rights reserved.
 *Other names and brands may be claimed as the property of others.



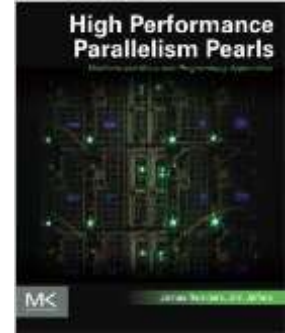
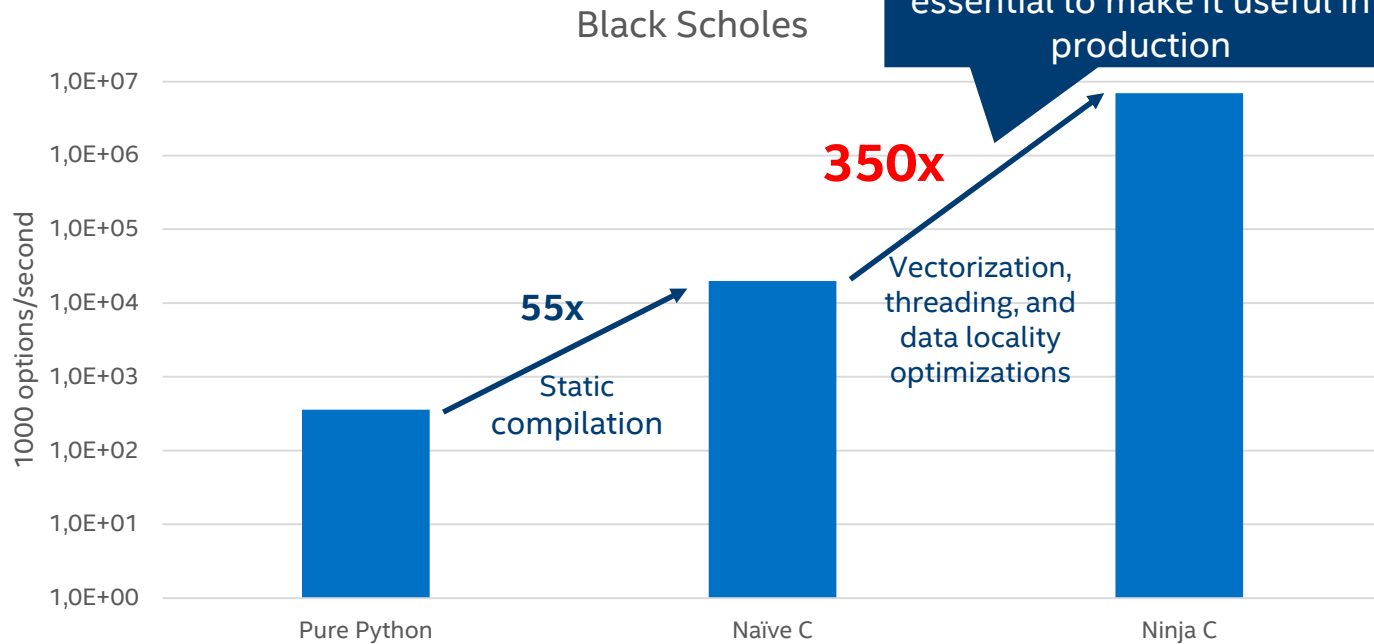
**Out-of-the-box performance is
not good enough for production**

**Use of
high-performance extensions
is hard**

Intel® Distribution for Python*

**Gives easy access to high-
performance in Python***

What is required for making Python performance closer to native code?



Chapter 19. Performance Optimization of Black Scholes Pricing

$$V_{call} = S_0 \cdot CDF(d_1) - e^{-rt} \cdot X \cdot CDF(d_2)$$

$$V_{put} = e^{-rt} \cdot X \cdot CDF(-d_2) - S_0 \cdot CDF(-d_1)$$

$$d_1 = \frac{\ln\left(\frac{S_0}{X}\right) + \left(r + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}$$

$$d_2 = \frac{\ln\left(\frac{S_0}{X}\right) - \left(r - \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}$$

Optimization Notice

Copyright © 2016, Intel Corporation. All rights reserved.
 *Other names and brands may be claimed as the property of others.

Configuration info: - Versions: Intel® Distribution for Python 2.7.10 Technical Preview 1 (Aug 03, 2015), gcc 15.0; Hardware: Intel® Xeon® CPU E5-2698 v3 @ 2.30GHz (2 sockets, 16 cores each, HT=OFF), 64 GB of RAM, 8 DIMMS of 8GB@2133MHz; Operating System: Ubuntu 14.04 LTS.



Performance-productivity technological choices

Numerical packages
acceleration with Intel®
performance libraries
(MKL, DAAL, IPP)

Better parallelism
and composable
multi-threading
(TBB, MPI)

Profiling Python and
mixed language codes
(VTune)

Language extensions for
vectorization and multi-
threading
(Cython, Numba, Pyston, etc)

Integration with Big Data and
Machine Learning platforms and
frameworks
(Spark, Hadoop, Theano, etc)

Optimization Notice

Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.



Intel® and Python

1. Enable hooks in Python packages
 - Intel® MKL, Intel® DAAL, Intel® IPP
 - Most popular numerical/data processing packages
 - NumPy, SciPy, Scikit-Learn, PyTables, Scikit-Image, ...
2. Provide Python interfaces for Intel® DAAL (a.k.a PyDAAL)
3. Available through Intel® Distribution for Python* and as Conda packages
4. Most optimizations eventually upstreamed to open source projects

Optimized mathematical building blocks

Intel® Math Kernel Library (Intel MKL)

Linear Algebra

- BLAS
- LAPACK
- ScaLAPACK
- Sparse BLAS
- Sparse Solvers
 - Iterative
- PARDISO* SMP & Cluster

Up to
100x
faster

Fast Fourier Transforms

- **Multidimensional**
- FFTW interfaces
- Cluster FFT

Up to
10x
faster!

Vector Math

- Trigonometric
- Hyperbolic
- Exponential
- Log
- Power
- Root

Up to
10x
faster!

Vector RNGs

- **Multiple BRNG**
- **Support methods for independent streams creation**
- **Support all key probability distributions**

Up to
60x
faster!

Summary Statistics

- Kurtosis
- Variation coefficient
- Order statistics
- Min/max
- Variance-covariance

And More

- Splines
- Interpolation
- Trust Region
- Fast Poisson Solver

Optimization Notice

Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Functional domain in **this color** accelerate respective NumPy, SciPy, etc. domain

Configuration info: - Versions: Intel® Distribution for Python 2017 Beta, icc 15.0; Hardware: Intel® Xeon® CPU E5-2698 v3 @ 2.30GHz (2 sockets, 16 cores each, HT=OFF), 64 GB of RAM, 8 DIMMS of 8GB@2133MHz; Operating System: Ubuntu 14.04 LTS.



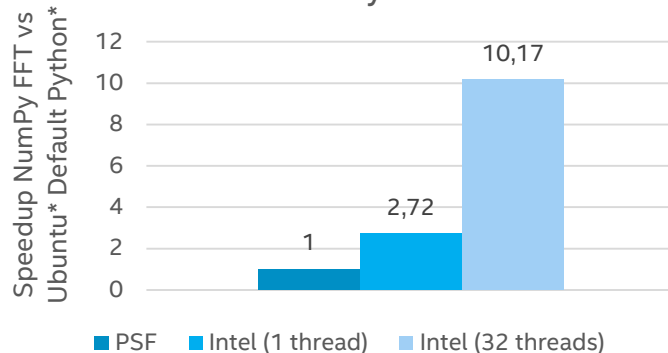
Optimized FFT show case

Intel® Math Kernel Library (Intel MKL)

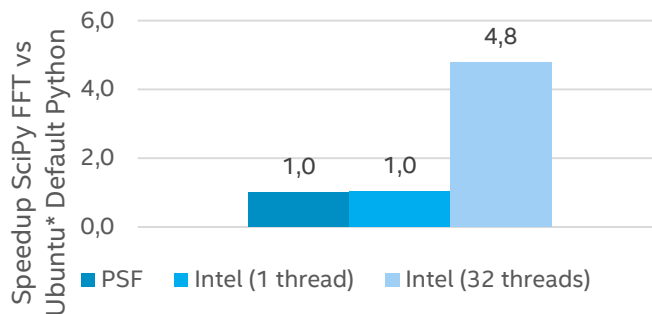
- Original SciPy FFT implementation is about 2x faster than original NumPy FFT
- Intel engineers bridged NumPy and SciPy implementations via common layer and embedded MKL FFT calls, what measurably accelerates both NumPy and SciPy
 - NumPy and SciPy are computationally compatible
 - FFT descriptors caching applied for enhanced performance in repetitive and multidimensional FFT calculations

Available starting Intel® Distribution for Python* 2017 Beta

NumPy FFT NumPy vs Ubuntu*
Default Python*



SciPy FFT Intel SciPy vs Ubuntu*
Vanilla Python*



Optimization Notice

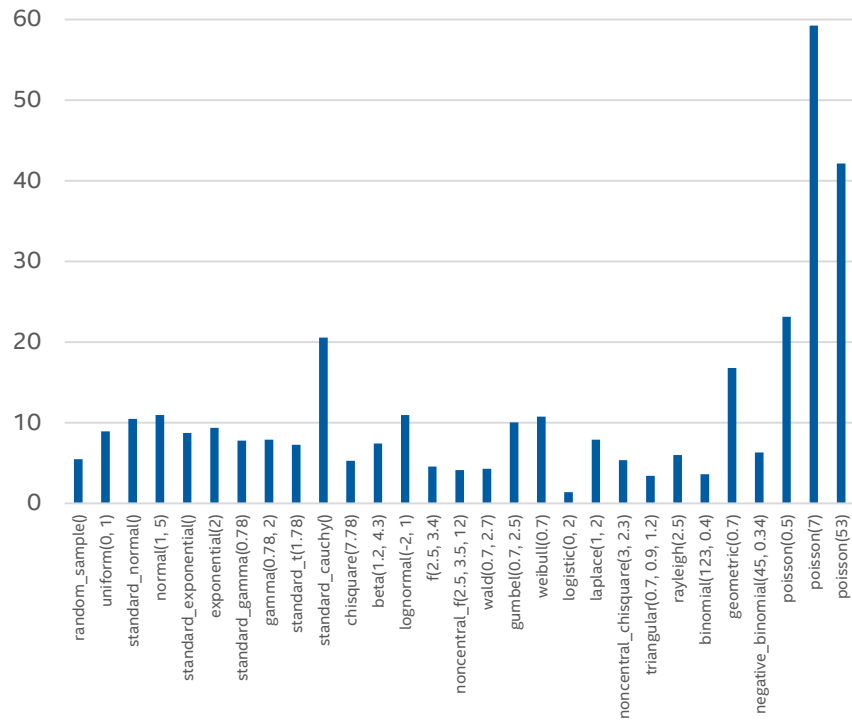
Optimized RNG show case

Intel® Math Kernel Library (Intel MKL)

- Implemented numpy.random in vector fashion to enable vector MKL RNG and VML calls
- Enabled multiple BRNG
- Enabled multiple distribution transformation methods

Initial data. Final data to be available in the update for Intel® Distribution for Python* 2017 Beta

numpy.random speedup due to MKL RNG



Optimization Notice

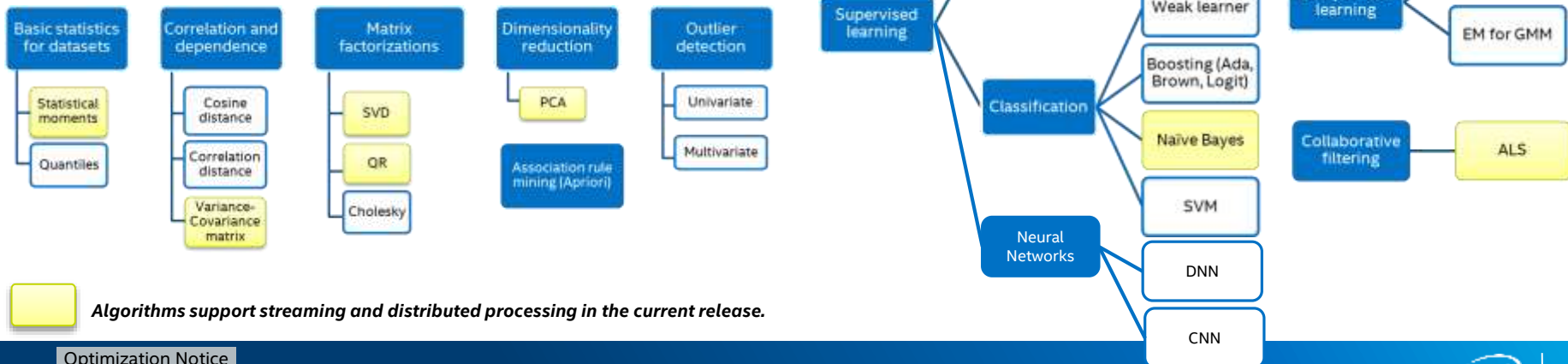
Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Configuration info: - Versions: Intel® Distribution for Python (internal build 28.4.2016), gcc 15.0; Hardware: Intel® Xeon® CPU E5-2630 v3 @ 2.40GHz (16 cores), 32 GB; Operating System: Ubuntu 14.04 LTS.



Optimized blocks for data analytics pipelines

Intel® Data Analytics Acceleration Library (Intel DAAL)



Optimization Notice

Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

PROFILING PYTHON APPLICATIONS WITH INTEL[®] VTUNE[™] AMPLIFIER



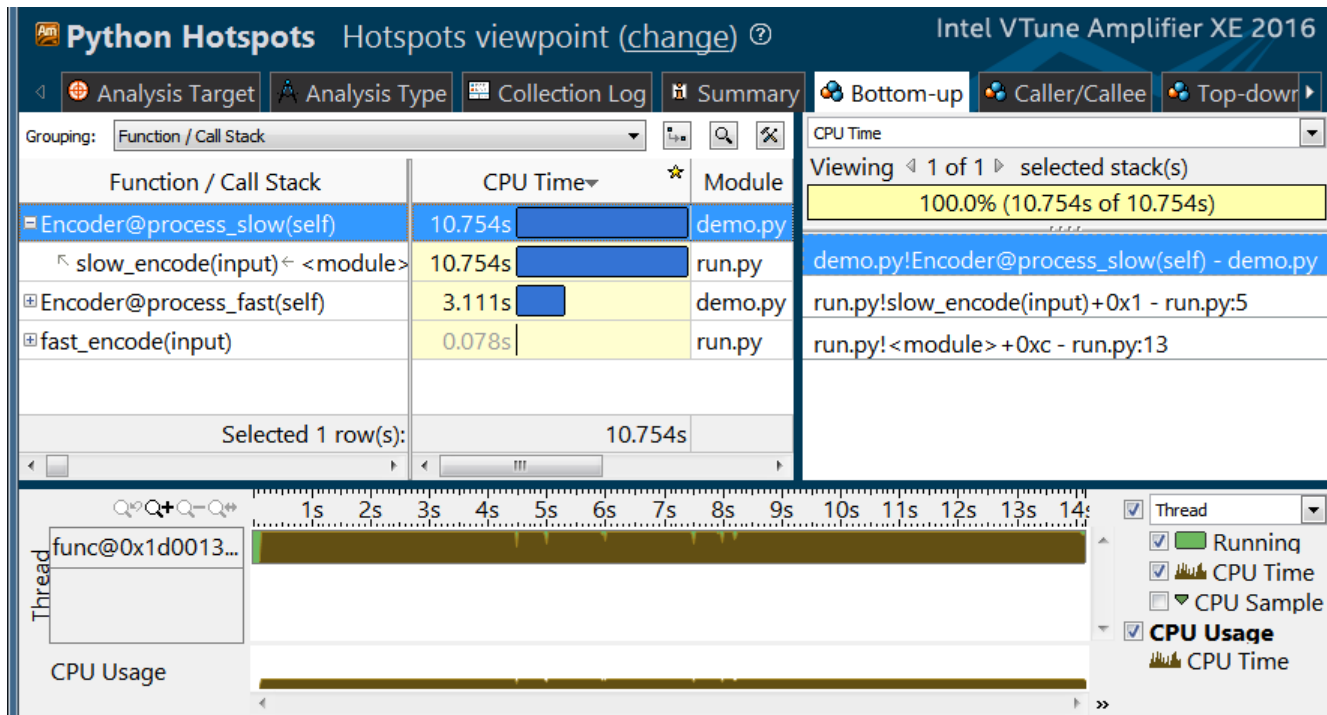
Simple Python profiling example

```
class Encoder:
    CHAR_MAP = {'a': 'b', 'b': 'c'}
    def __init__(self, input):
        self.input = input

    def process_slow(self):
        result = ''
        for ch in self.input:
            result += self.CHAR_MAP.get(ch, ch)
        return result

    def process_fast(self):
        result = []
        for ch in self.input:
            result.append(self.CHAR_MAP.get(ch, ch))
        return ''.join(result)
```

Intel® VTune™ Amplifier example



Optimization Notice

Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.



Intel® VTune™ Amplifier – source view

The screenshot shows the Intel VTune Amplifier XE 2016 interface. The main window displays the source code for a Python class named `Encoder`. The code is as follows:

```
1 class Encoder:
2     CHAR_MAP = {'a': 'b', 'b': 'c'}
3     def __init__(self, input):
4         self.input = input
5
6     def process_slow(self):
7         result = ''
8         for ch in self.input:
9             result += self.CHAR_MAP.
10        return result
11
12    def process_fast(self):
13        result = []
```

The CPU Time column shows that the `process_slow` method (lines 6-10) is the most time-consuming, taking 10.754s. A blue bar highlights this value. The right-hand pane shows the CPU Time profile, indicating that 100.0% (10.754s of 10.754s) of the time is spent in the `demo.py!Encod...lf` function.

| S. ▲ | Source | CPU Time |
|------|---|----------|
| 1 | <code>class Encoder:</code> | |
| 2 | <code> CHAR_MAP = {'a': 'b', 'b': 'c'}</code> | |
| 3 | <code> def __init__(self, input):</code> | |
| 4 | <code> self.input = input</code> | |
| 5 | | |
| 6 | <code> def process_slow(self):</code> | |
| 7 | <code> result = ''</code> | |
| 8 | <code> for ch in self.input:</code> | |
| 9 | <code> result += self.CHAR_MAP.</code> | 10.754s |
| 10 | <code> return result</code> | |
| 11 | | |
| 12 | <code> def process_fast(self):</code> | |
| 13 | <code> result = []</code> | |

Optimization Notice

Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.



Intel® VTune™ Amplifier: Accurate & Easy

Line-level profiling details:

- Uses sampling profiling technique
- Average overhead ~1.1x-1.6x (on certain benchmarks)

Cross-platform:

- Windows and Linux
- Python 32- and 64-bit; 2.7.x, 3.4.x, 3.5.0 versions

Rich Graphical UI

Supported workflows:

- Start application, wait for it to finish
- Attach to application, profile for a bit, detach

Boost Your Python Performance with Intel® Distribution for Python

Easy access through full installer and conda

Full scipy-stack + selected HPC/Big-Data packages

numpy, scipy, matplotlib, ipython/jupyter, sympy, pandas, pyDAAL, scikit-*, mpi4py,...

Windows, Linux, MacOS (all 64bit)

Sign up to beta at <https://software.intel.com/en-us/python-distribution>

Get answers, help&support from

<https://software.intel.com/en-us/forums/intel-distribution-for-python>



Legal Disclaimer & 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.

Copyright © 2016, 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