FAST DATA ANALYTICS WITH PYTHON* AND INTEL® DAAL

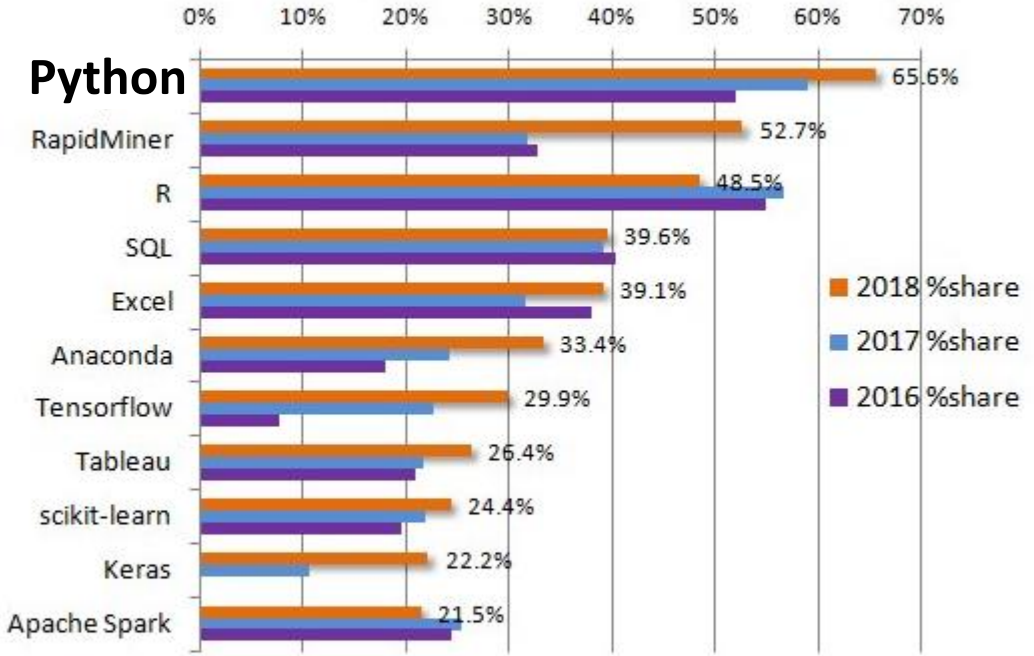
Ruslan Israfilov

Data Analytics Software Engineer, Intel

Python: lingua franca of data science

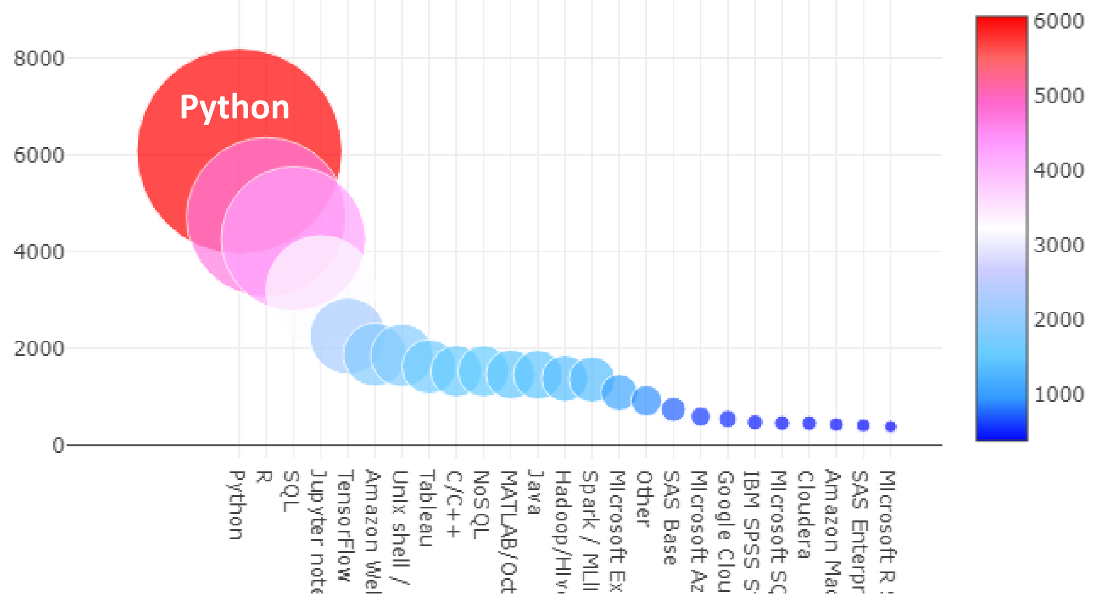
python™

KDnuggets Analytics, Data Science & Machine Learning Software Pool, 2016-2018



https://www.kdnuggets.com/2018/05/poll-tools-analytics-data-science-machine-learning-results.html

Kaggle, Machine Learning & Data Science Survey, 2017



https://www.kaggle.com/sudalairajkumar/an-interactive-deep-dive-into-survey-results/data

Python is not about performance

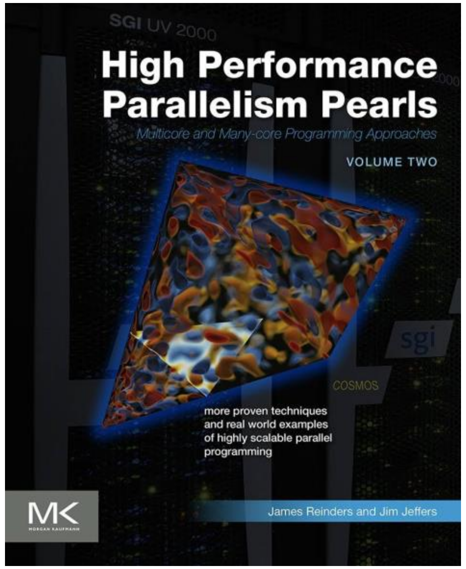
Black—Scholes Formula

Problem: Evaluate fair European call- and put-option price, V_{call} and V_{put}, for underlying stock

PARSEC
['pär-"sek] A unit of measure

good performance benchmark

```python
def black_scholes ( nopt, price, strike, t, rate, vol ):
        mr = -rate
        sig_sig_two = vol * vol * 2

        P = price
        S = strike
        T = t

        a = log(P / S)
        b = T * mr

        z = T * sig_sig_two
        c = 0.25 * z
        y = invsqrt(z)

        w1 = (a - b + c) * y
        w2 = (a - b - c) * y

        d1 = 0.5 + 0.5 * erf(w1)
        d2 = 0.5 + 0.5 * erf(w2)

        Se = exp(b) * S

        call = P * d1 - Se * d2
        put = call - P + Se

        return call, put
```



Chapter 19: Performance Optimization of **Black—Scholes** Pricing

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

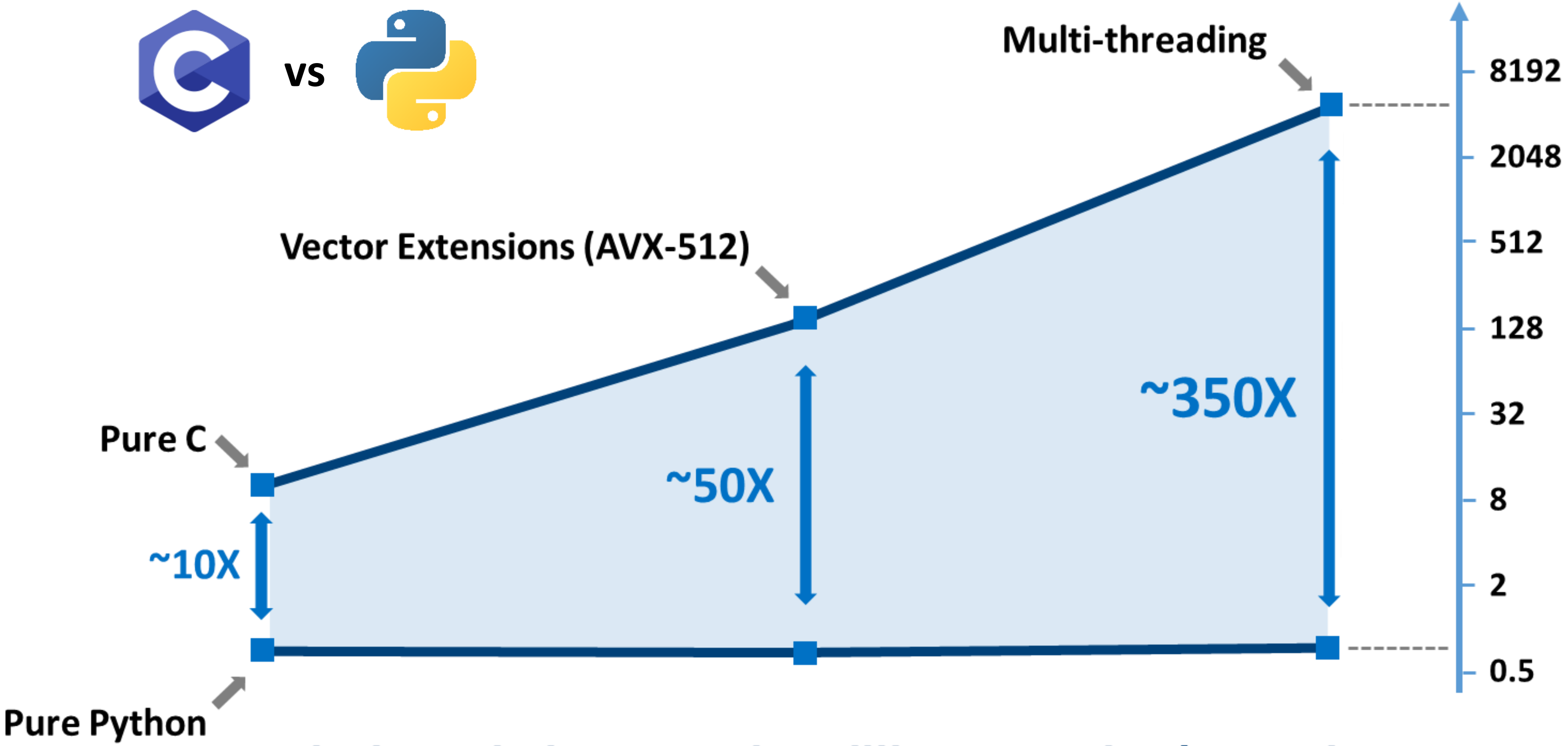
$$V_{put} = e^{-rT} \cdot X \cdot \mathrm{CDF}(-d_2) - S_0 \cdot \mathrm{CDF}(-d_1)$$

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

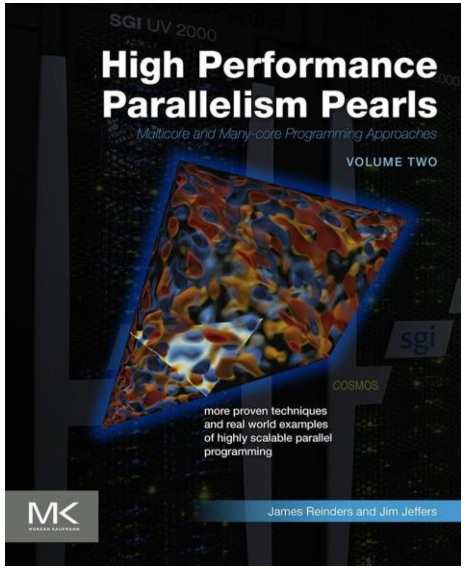
$$d_2 = \frac{\ln\left(S_0/X\right) + \left(r - \sigma^2/2\right)T}{\sigma\sqrt{T}}$$

Benchmark: http://parsec.cs.princeton.edu/overview.htm

Python is not about performance



Black—Scholes Formula: Million operation/second



Chapter 19: Performance Optimization of **Black—Scholes** Pricing

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

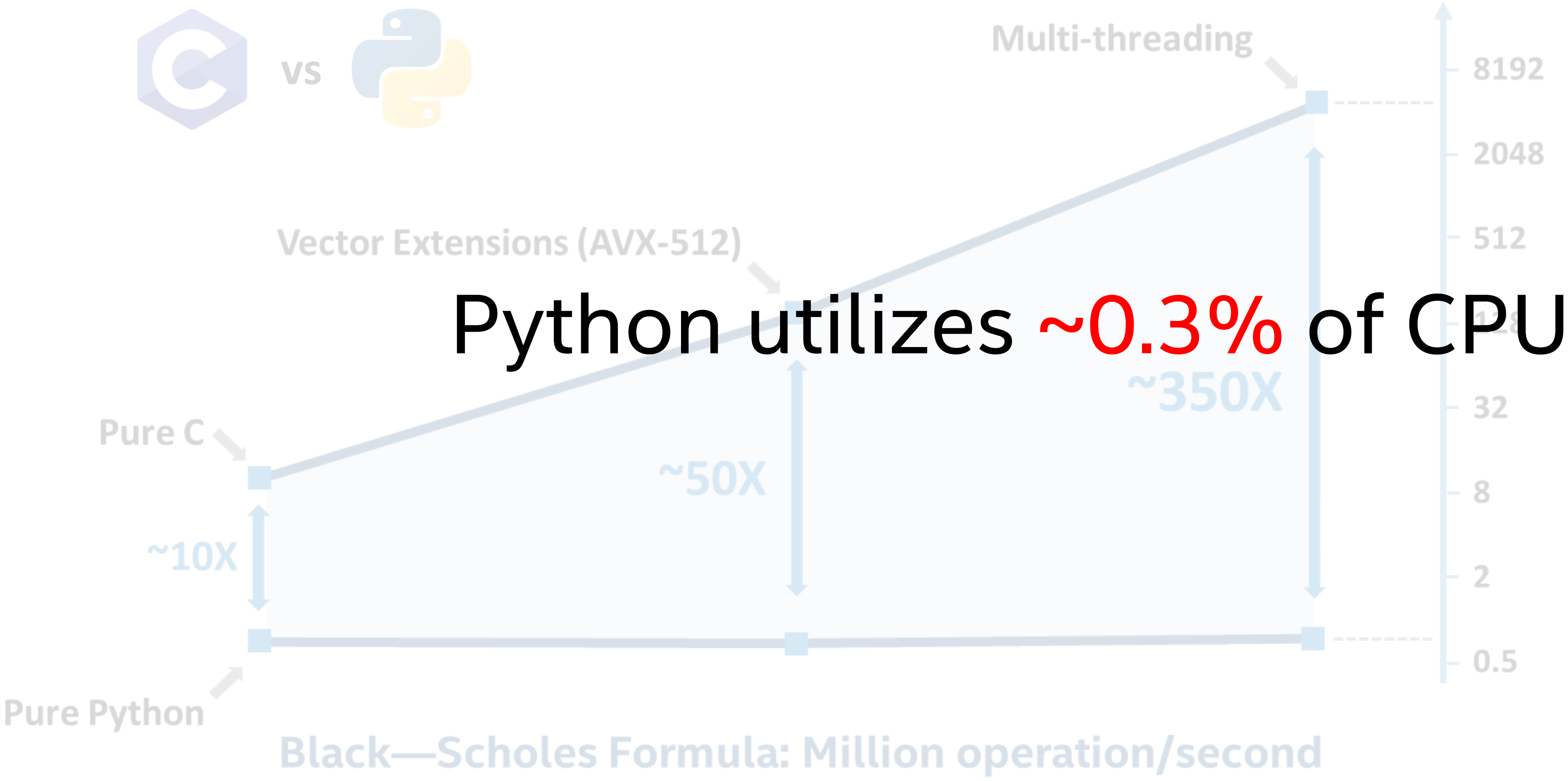
$$V_{put} = e^{-rT} \cdot X \cdot \mathrm{CDF}(-d_2) - S_0 \cdot \mathrm{CDF}(-d_1)$$

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

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

Source Code: github.com/IntelPython/BlackScholes_bench

Python is not about performance



Python utilizes ~0.3% of CPU

Chapter 19: Performance Optimization of Black—Scholes Pricing

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

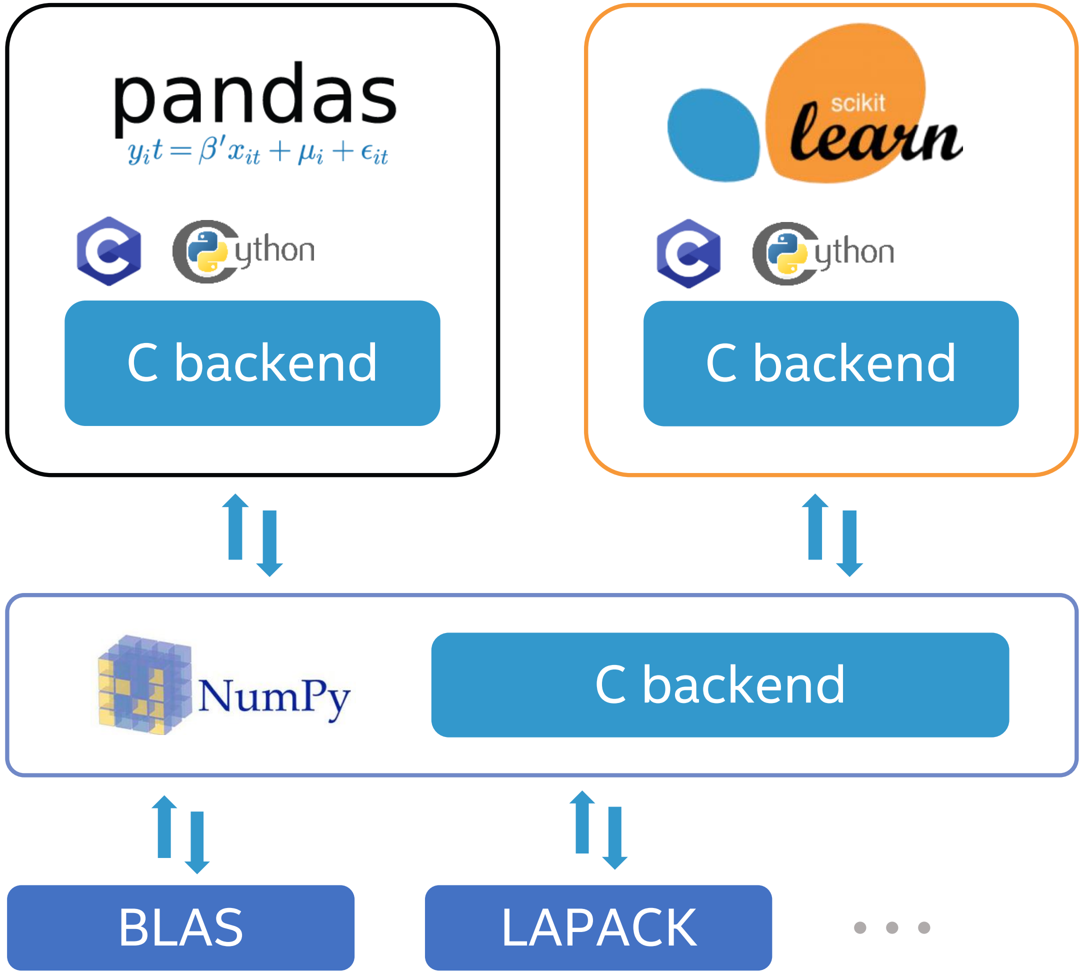
$$V_{put} = e^{-rT} \cdot X \cdot \mathrm{CDF}(-d_2) - S_0 \cdot \mathrm{CDF}(-d_1)$$

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

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

Source Code: github.com/IntelPython/BlackScholes_bench

Performance Issues in Data Analytics



- No hardware-specific optimizations:
  - No (or inefficient) vectorization
  - Bad cache-memory utilization
- No (or inefficient) threading
- Greater part is still written in Python
- Global Interpreter Lock (GIL)
- Poorly optimized low-level math libraries

INTEL® DISTRIBUTION FOR PYTHON*

Drop in replacement for your existing Python. No code changes required.

scikit learn · NumPy · SciPy · pandas · TensorFlow · and more …

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

OPTIMIZED WITH	Intel® DAAL	Intel® MKL	Intel® MPI
BUILT WITH	Intel® C/C++ Compilers		

optimized for Intel hardware

intel ATOM x7 inside · intel CORE i7 8th Gen · intel XEON PLATINUM inside

try it now via Conda* or Docker*

```
conda create -c intel intelpython3_full
```

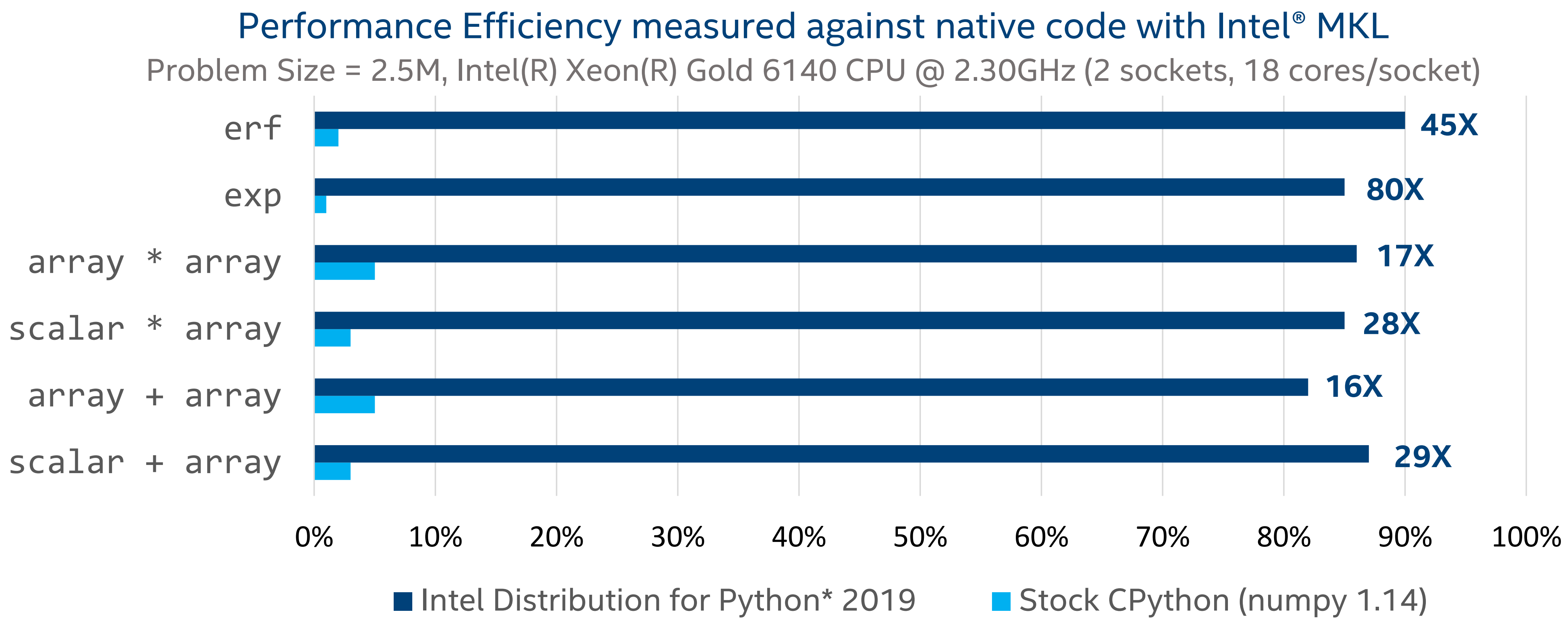
```
docker pull intelpython/intelpython3_full
```

Multiple OSs

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

Close to native code vector math Performance with Intel Python* 2019

Compared to Stock Python packages on Intel® Xeon processors



See hardware & software configuration at the end

More Benchmarks: software.intel.com/en-us/distribution-for-python/benchmarks

Accelerating Data Analytics

Optimized scikit-learn* package
Part of Intel Distribution for Python*

scikit-learn* frontend API

Intel® Data Analytics Acceleration Library (DAAL)

Intel® Math Kernel Library (MKL)

Intel® Threading Building Blocks (TBB)

- Efficient memory layout
- Vectorization (SSE, AVX)
- Data chunking for optimal cache-memory access
- Parallelization via Intel® TBB

```
conda install -c intel scikit-learn
```

```
pip install intel-scikit-learn
```

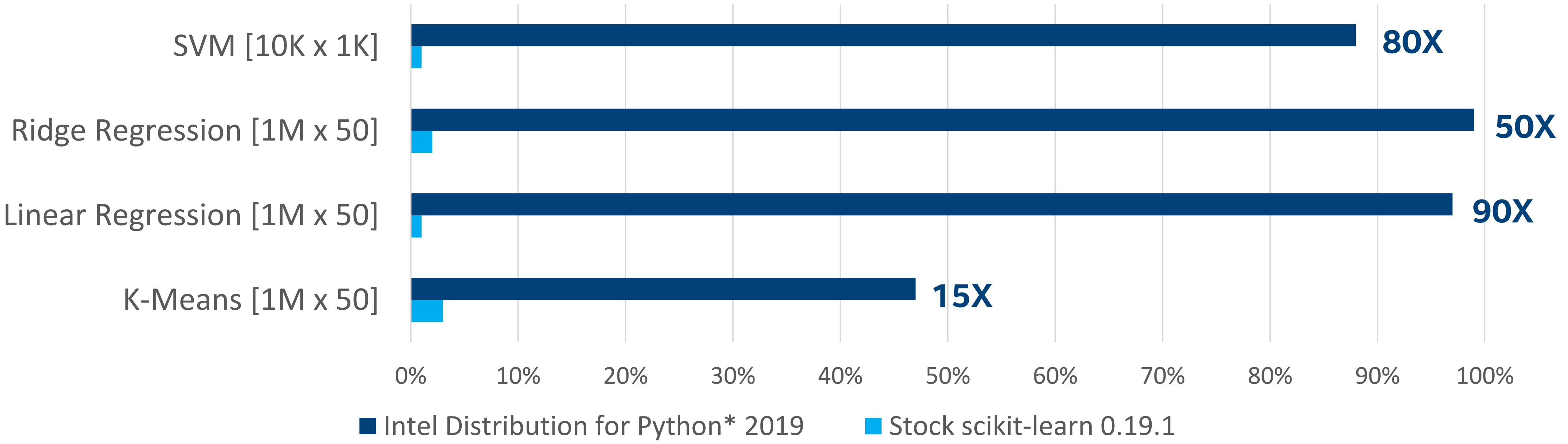

Intel® DAAL: software.intel.com/intel-daal

Close to native code scikit-learn Performance with Intel Python* 2019

Compared to Stock Python packages on Intel® Xeon processors

Performance Efficiency measured against native code with Intel® DAAL

Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz (2 sockets, 18 cores/socket)



See hardware & software configuration at the end

CASE STUDY Optimal Design with **5x** Performance Boost

DATADVANCE

“We tested different version combinations and distributions of Python and NumPy for estimation of Sobol indices using pSeven Core, since it’s one of the common problems our customers solve. **For older versions of Python, for example 2.6, the boost reached even 10x, but for the newer ones it stayed around 3x to 5x”**

Dmitry Vetrov, chief developer at DATADVANCE

Installing Intel® Distribution for Python

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:
https://software.intel.com/en-us/articles/installing-intel-free-libs-and-python

scikit learn
pandas $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$
NumPy
SciPy
Numba
TensorFlow

2.7 & 3.6

intel®

Hardware & Software Configuration

Close to native code vector math Performance with Intel Python* 2019 &
Close to native code scikit-learn Performance with Intel Python* 2019

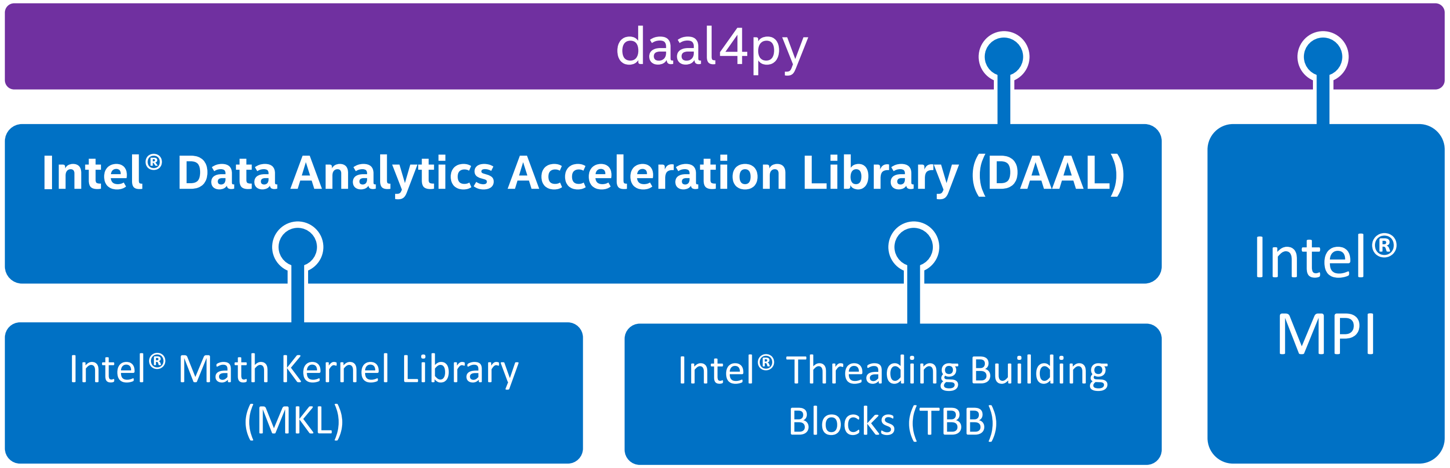
python 3.6.6 hc3d631a_0 installed from conda, numpy 1.15, numba 0.39.0, llvmlite 0.24.0, scipy 1.1.0, scikit-learn 0.19.2 installed from pip;Intel Python: Intel Distribution for Python 2019 Gold: python 3.6.5 intel_11, numpy 1.14.3 intel_py36_5, mkl 2019.0 intel_101, mkl_fft 1.0.2 intel_np114py36_6,mkl_random 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

Distributed K-Means Scalability with Intel® DAAL and Intel® MPI

Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz, 24 cores per node, 786GB RAM per node; Infiniband 100 Gb/sec (4X EDR); Intel(R) MPI 2018 U3, Intel(R) DAAL 2019 C++, Intel(R) C++ Compiler 2018

Scaling Machine Learning Beyond a Single Node

New API for scaling machine learning with Intel® DAAL & Intel® MPI



Simple Python API similar to scikit-learn*

Powered by Intel® DAAL

Scalable to multiple nodes

```
conda install -c intel daal4py
```

Example: Distributed K-Means with daal4py

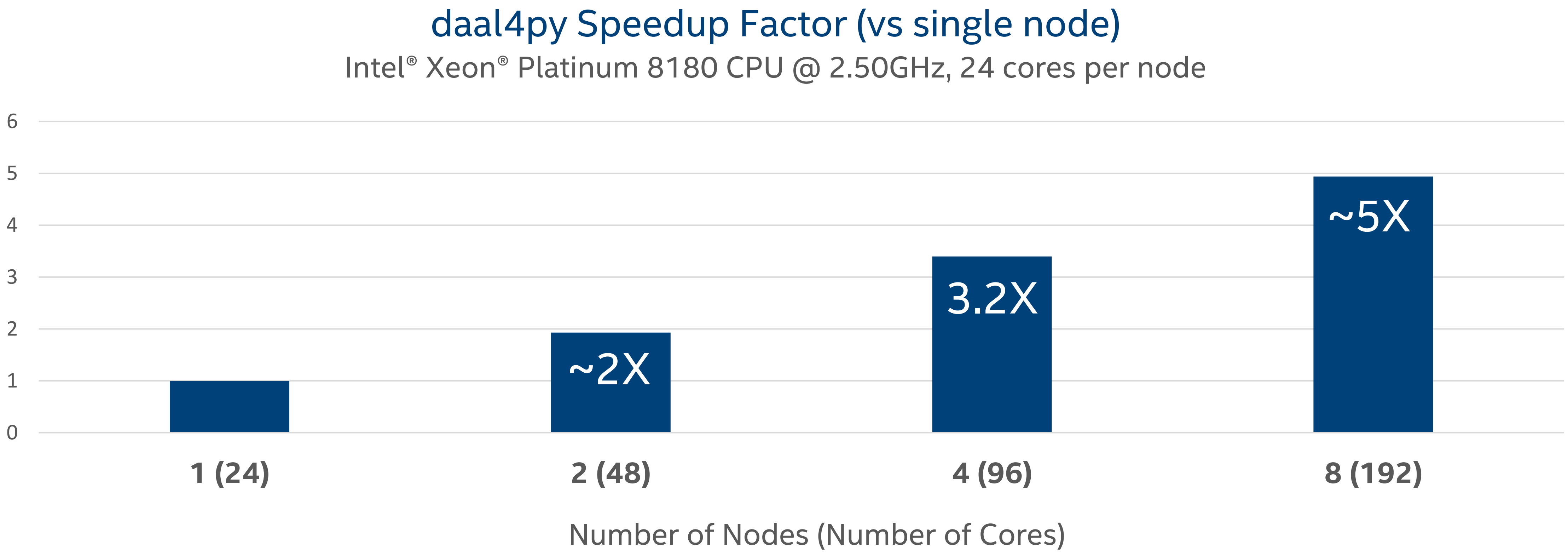
kmeans.py:

```
import daal4py as d4p

# initialize distributed execution environment
d4p.daalinit()

# load data from csv file into numpy array
data = pd.read_csv("path_to_data.csv").values

# compute initial centroids
centroids = d4p.kmeans_init(10, distributed=True).compute(data)

# compute centroids and assignments
result = d4p.kmeans(10, distributed=True).compute(data, centroids)
```

```
mpirun -n 4 -genv DIST_CNC=MPI python kmeans.py
```

Distributed K-Means Scalability with Intel® DAAL and Intel® MPI

Measured on InfiniBand* cluster on Intel® Xeon processors



See hardware & software configuration at the end

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. Source: Intel Corporation - performance measured in Intel labs by Intel employees. 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.

Legal Disclaimer & Optimization Notice

The benchmark results reported above may need to be revised as additional testing is conducted. The results depend on the specific platform configurations and workloads utilized in the testing, and may not be applicable to any particular user's components, computer system or workloads. The results are not necessarily representative of other benchmarks and other benchmark results may show greater or lesser impact from mitigations.

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.

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