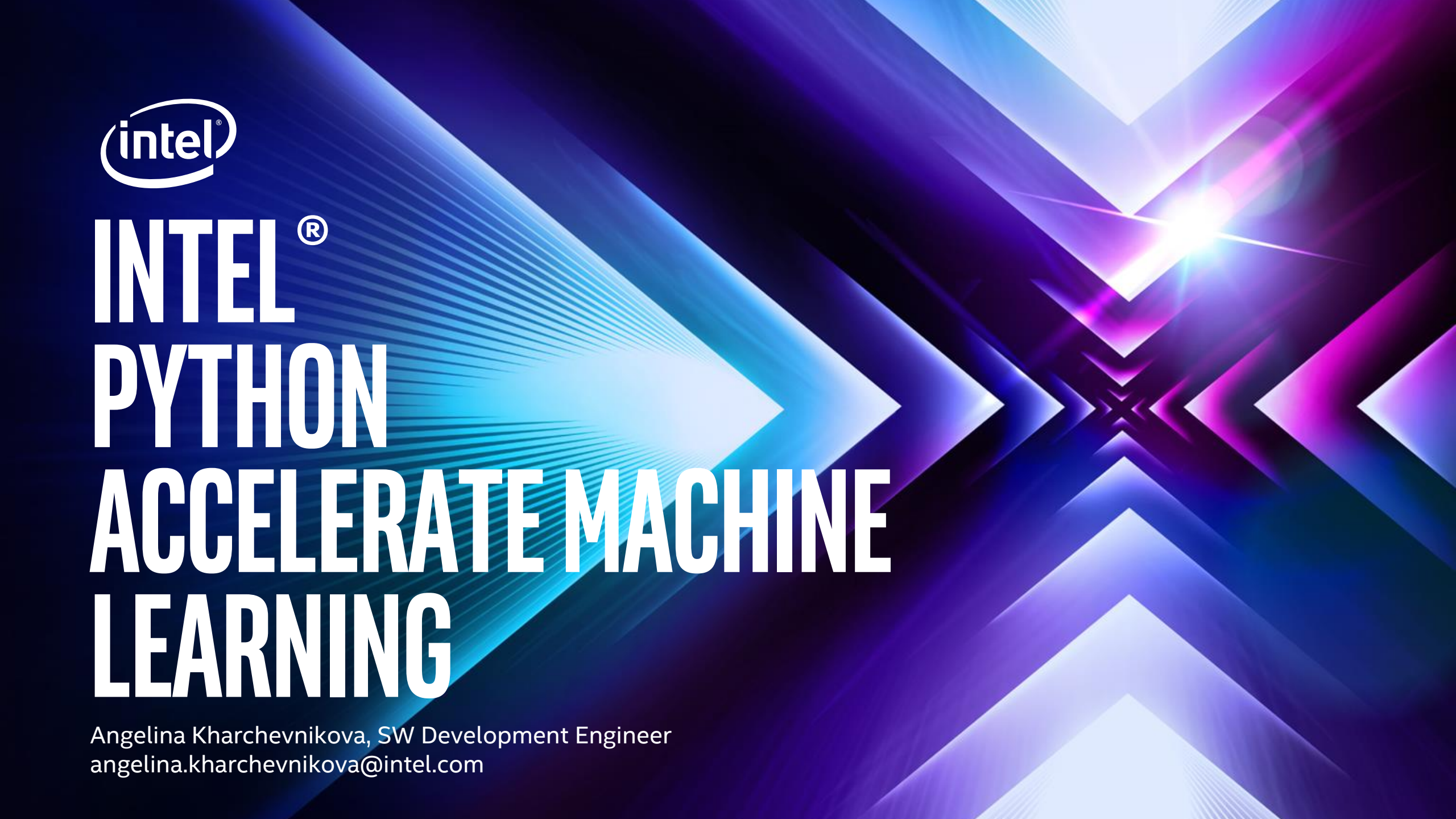


Angelina Kharchevnikova, SW Development Engineer
angelina.kharchevnikova@intel.com

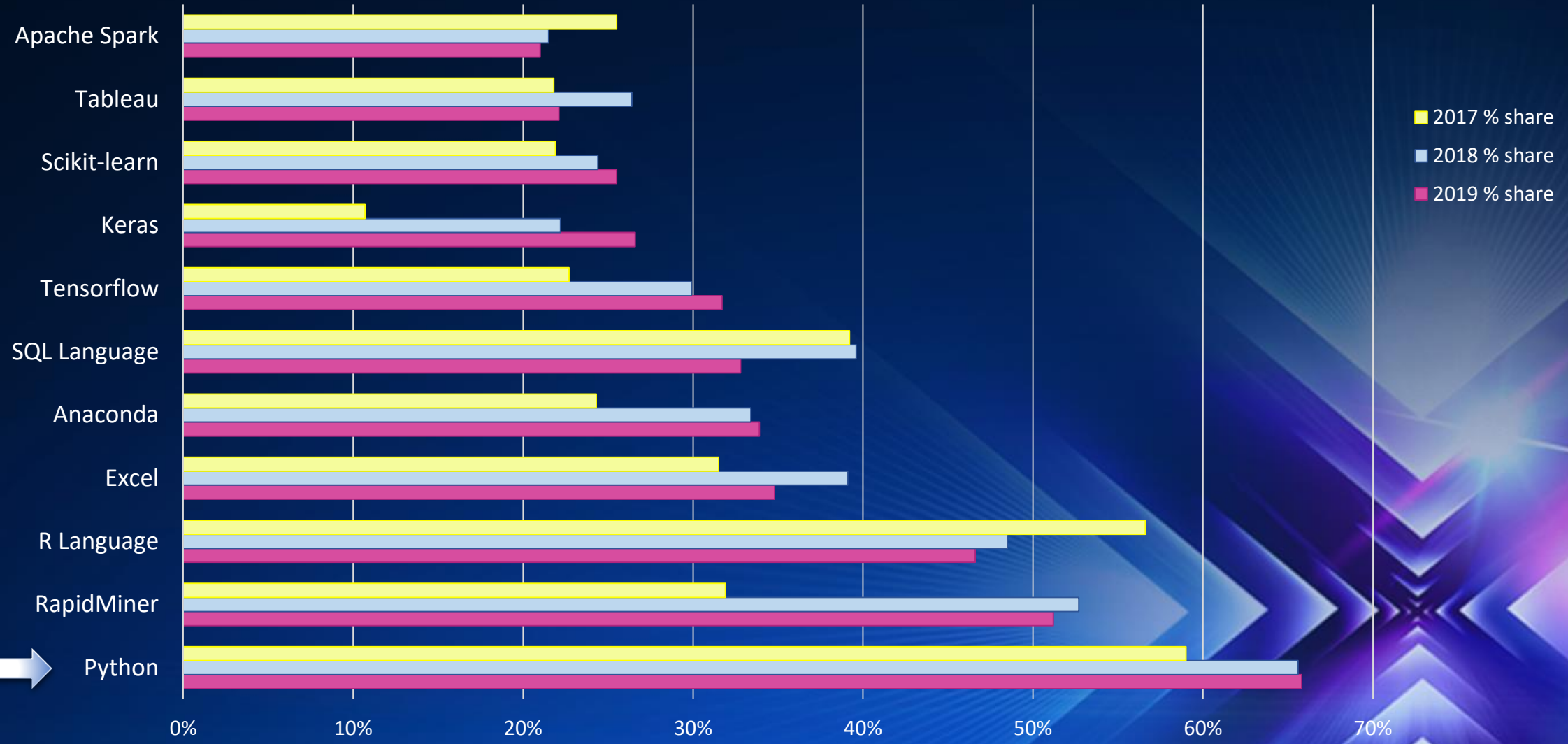
DISCLOSURES

Intel Technology and Manufacturing Day 2017 occurs during Intel's "Quiet Period," before Intel announces its 2017 first quarter financial and operating results. Therefore, presenters will not be addressing first quarter information during this year's program.

Statements in this presentation that refer to forecasts, future plans and expectations are forward-looking statements that involve a number of risks and uncertainties. Words such as "anticipates," "expects," "intends," "goals," "plans," "believes," "seeks," "estimates," "continues," "may," "will," "would," "should," "could," and variations of such words and similar expressions are intended to identify such forward-looking statements. Statements that refer to or are based on projections, uncertain events or assumptions also identify forward-looking statements. Such statements are based on management's expectations as of March 28, 2017, and involve many risks and uncertainties that could cause actual results to differ materially from those expressed or implied in these forward-looking statements. Important factors that could cause actual results to differ materially from the company's expectations are set forth in Intel's earnings release dated January 26, 2017, which is included as an exhibit to Intel's Form 8-K furnished to the SEC on such date. Additional information regarding these and other factors that could affect Intel's results is included in Intel's SEC filings, including the company's most recent reports on Forms 10-K, 10-Q and 8-K reports may be obtained by visiting our Investor Relations website at www.intc.com or the SEC's website at www.sec.gov.

PYTHON

Top Analytics, Data Science, Machine Learning Software



https://www.kdnuggets.com/2019/05/poll-top-data-science-machine-learning-platforms.html

ACCELERATE PYTHON PERFORMANCE

Analysts
Data Scientists
Machine Learning Developers

python™

scikit learn

pandas

$y_i t = \beta' x_{it} + \mu_i + \epsilon_{it}$

- Achieve faster Python application performance - right out of the box - with minimal or no changes to the code
- Accelerate NumPy, SciPy, Scikit-learn and Pandas with integrated Intel® Performance Libraries such as Intel® Data Analytics Acceleration Library

https://github.com/IntelPython

MACHINE LEARNING PIPELINE

Pandas
SDC

Scikit-learn
Daal4py
Intel® Data Analytics Acceleration Library (DAAL)

SCALABLE DATAFRAME COMPILER

Data Input

Data Preprocessing

A JIT compiler-based framework to speed up Pandas

Decorator
@hpat.jit

Parallel/Distributed Analysis

Compile

```
@hpat.jit
def get_stats():
    …
    df['latency'].sum()
    df['latency'].mean()
    …
```

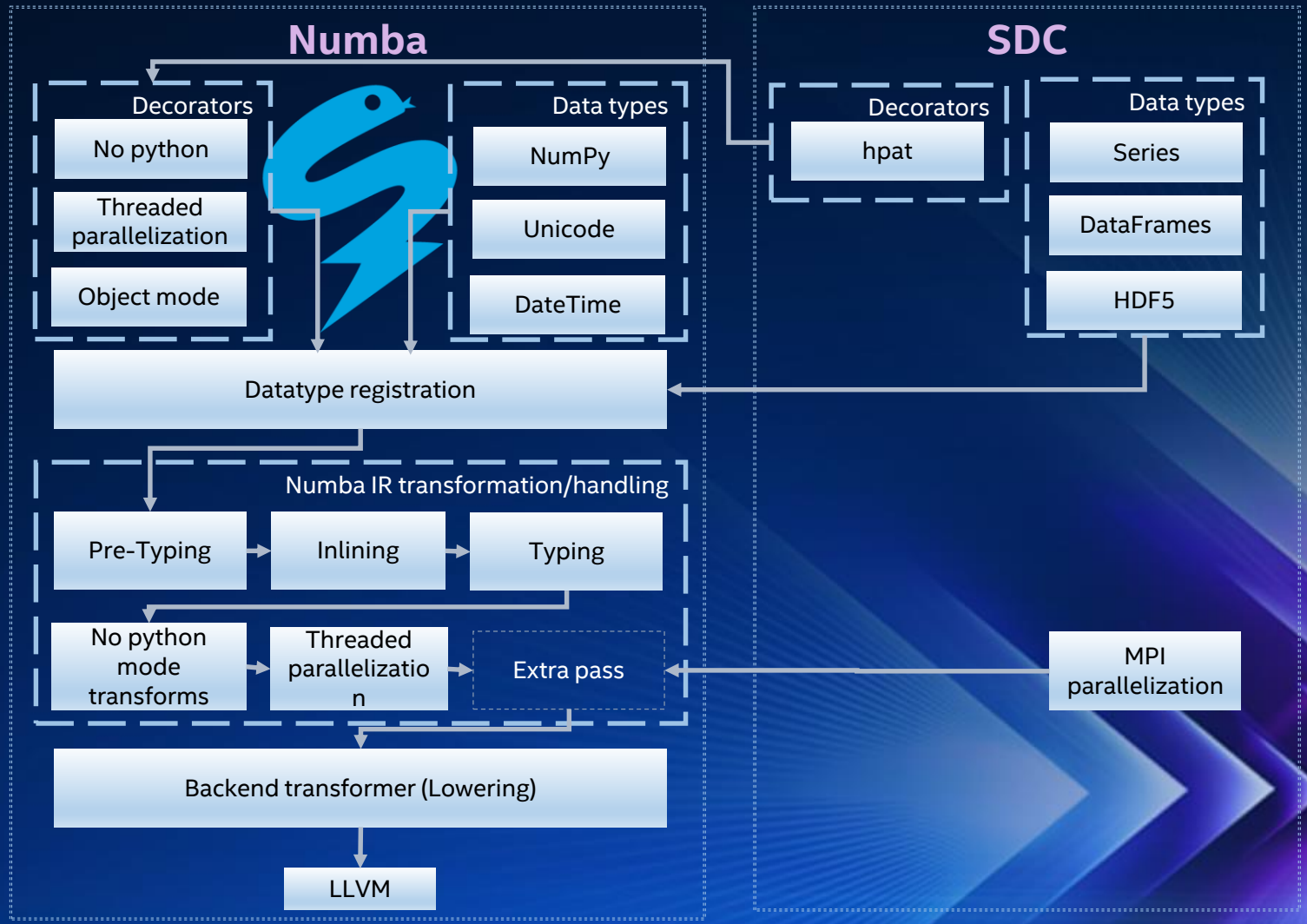
```
                vucomisd    %xmm0, %xmm0
                setnp       %dl
                jp          .LBB0_11
                vaddsd      %xmm0, %xmm2, %xmm2
.LBB0_11:
                vaddsd      %xmm0, %xmm3, %xmm1
                vcmpunordsd %xmm0, %xmm0, %xmm0
                vblendvpd   %xmm0, %xmm3, %xmm1
```

SCALABLE DATAFRAME COMPILER

Initialization:

Pipeline transform:

Execution:



SCALABLE DATAFRAME COMPILER

```
import numba
import numpy as np
import hpat
import time

@jit(nopython=True, parallel=True)
def calc_pi(n):
    x = 2 * np.random.ranf(n) - 1
    y = 2 * np.random.ranf(n) - 1
    pi = 4 * np.sum(x**2 + y**2 < 1) / n
return pi

n = int(0.1e9)
pi = calc_pi(n)
start_time = time.time()
pi = calc_pi(n)
print("time: ", time.time() - start_time)
print("PI:", pi)
```

```
import numba
import numpy as np
import hpat
import time

@jit(nopython=True, parallel=True)
def calc_pi(A, B, n):
    x = 2 * A - 1
    y = 2 * B - 1
    pi = 4 * np.sum(x**2 + y**2 < 1) / n
return pi

n = int(0.1e9)
A = np.random.ranf(n)
B = np.random.ranf(n)

pi = calc_pi(A, B, n)
start_time = time.time()
pi = calc_pi(A, B, n)
print("time: ", time.time() - start_time)
print("PI:", pi)
```

```
import numba
import numpy as np
import hpat
import time

@jit(nopython=True, parallel=True)
def calc_pi(A, B, n):
    x = 2 * A - 1
    y = 2 * B - 1
    pi = 4 * np.sum(x**2 + y**2 < 1) / n
return pi

n = int(0.1e9)
A = pd.Series(np.random.ranf(n))
B = pd.Series(np.random.ranf(n))

pi = calc_pi(A, B, n)
start_time = time.time()
pi = calc_pi(A, B, n)
print("time: ", time.time() - start_time)
print("PI:", pi)
```

#	Python	Numba		HPAT (default)
		nopython=True	nopython=True, parallel=True	
1	2.81	1.99	0.22	0.95
2	n/a	n/a	n/a	0.47
4	n/a	n/a	n/a	0.24
8	n/a	n/a	n/a	0.22

#	Python	Numba		HPAT (default)
		nopython=True	nopython=True, parallel=True	
1	0.75	0.64	0.06	0.08
2	n/a	n/a	n/a	0.12
4	n/a	n/a	n/a	0.23
8	n/a	n/a	n/a	0.52

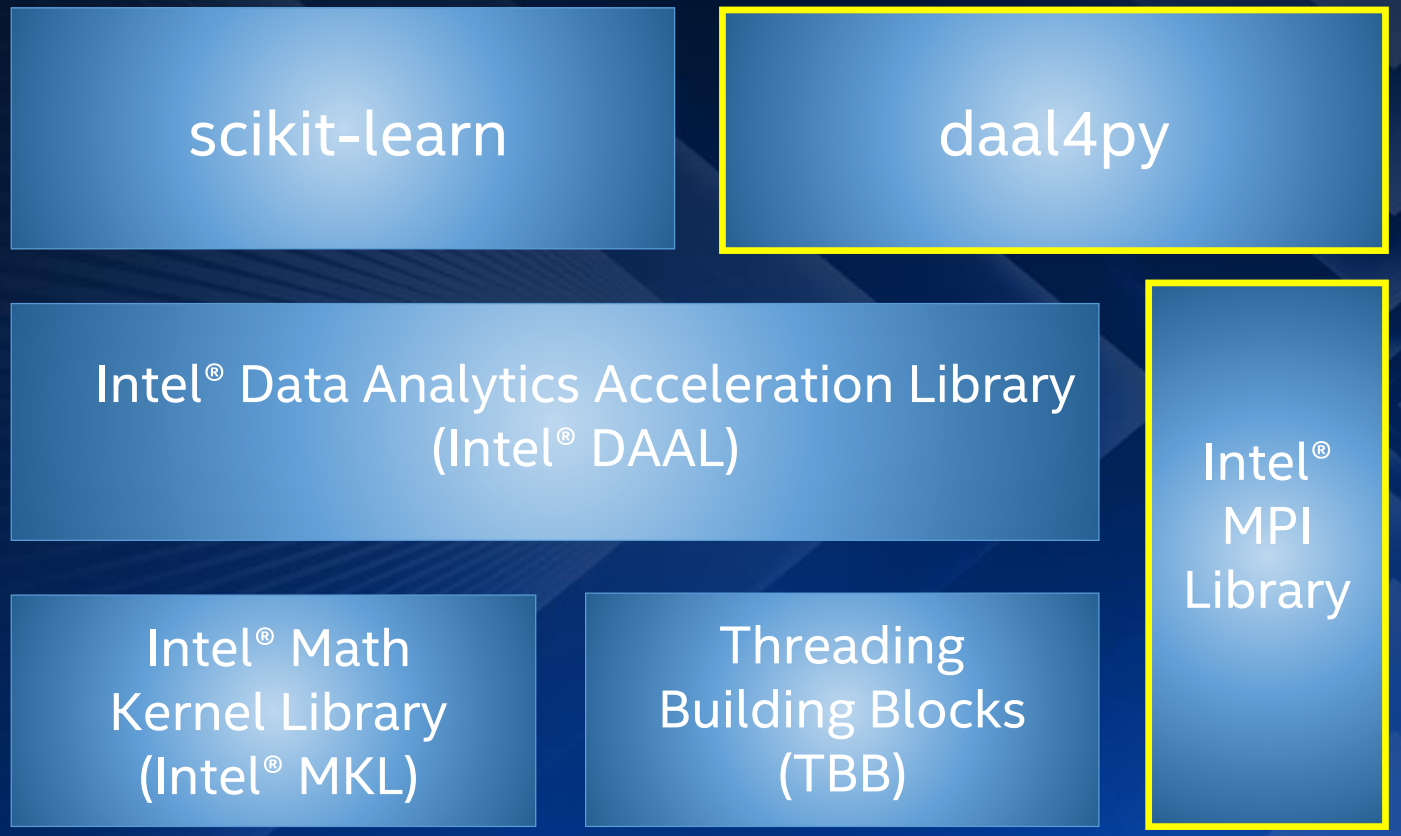
#	Python	Numba		HPAT (default)
		nopython=True	nopython=True, parallel=True	
1	3.01			0.33
2	n/a	n/a	n/a	0.40
4	n/a	n/a	n/a	0.60
8	n/a	n/a	n/a	

- Numbers are in seconds
- Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz, HT=ON (8 cores) 32GB mem

DAAL4PY

Simple Python API
Powers scikit-learn

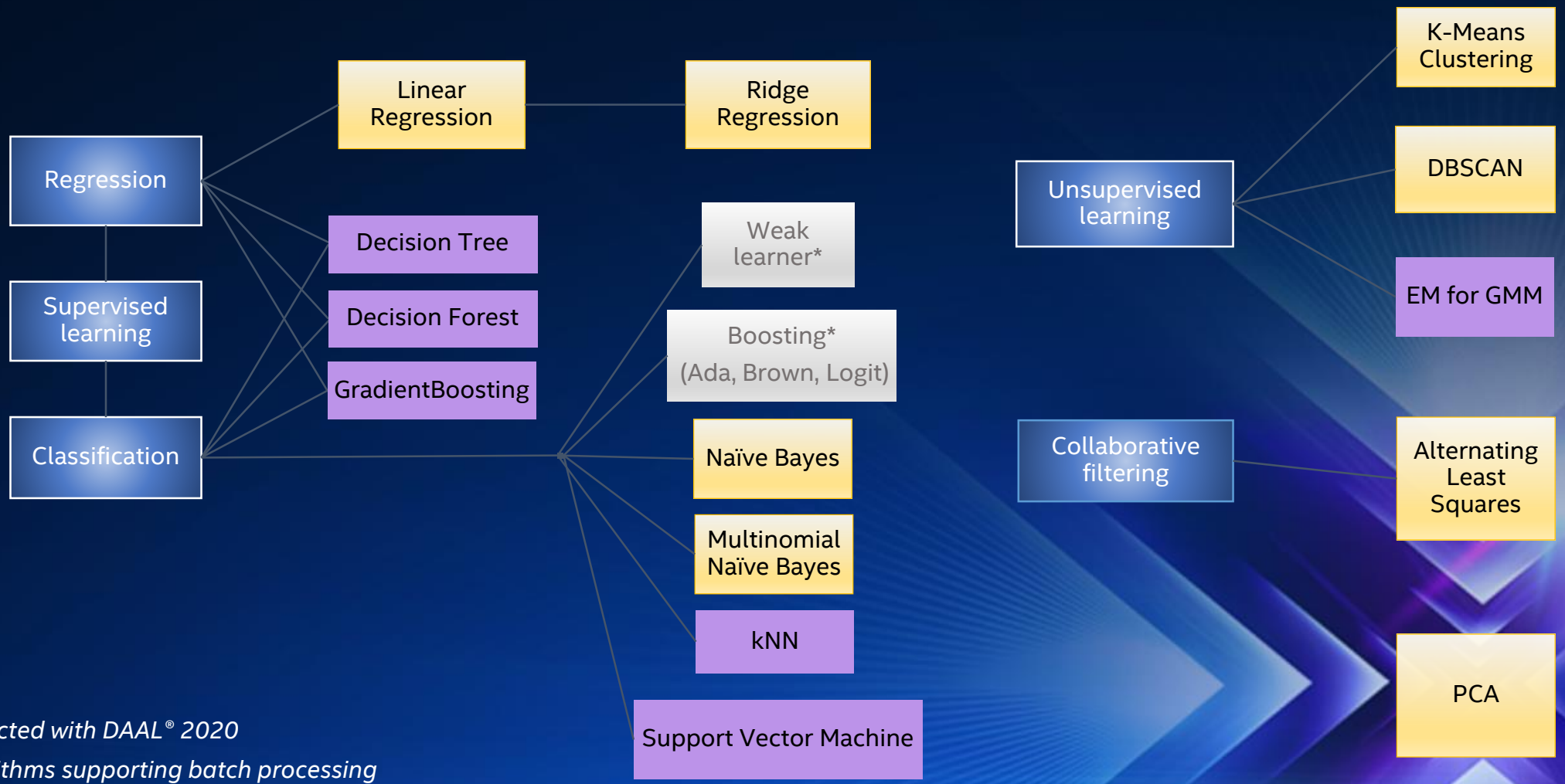
Powered by DAAl



Scalable to multiple nodes

Try it out! conda install -c intel daal4py

ML ALGORITHMS SUPPORTED BY DAAL4PY

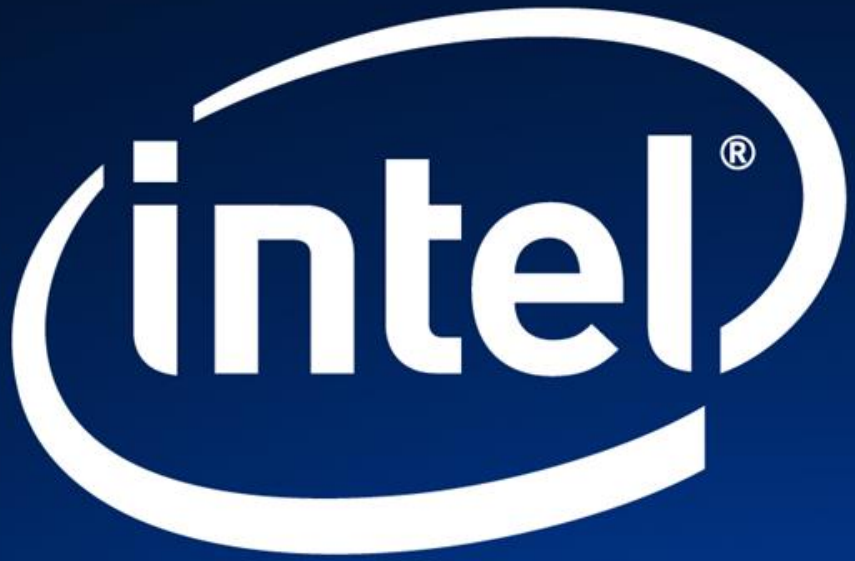


*Expected with DAAL® 2020

Algorithms supporting batch processing

Algorithms supporting batch, online and/or distributed processing

DEMO


intel