Fast Python* Analytics and Deep Learning Frameworks on CPU

Intel® Accelerations for AI

Nathan Greeneltch, PhD

Consulting Engineer, Intel Corporation

Intel® AI Framework Accelerations
Intel® Python* Accelerations
Intel® DAAL for Python*
Analytics

Software

Get Deep Learning Framework Performance on Intel® Architecture

Intel® Optimized AI Frameworks

Nathan Greeneltch, PhD

Consulting Engineer, Intel Corporation

Topics Covered

Define The Problem
Solutions: Intel® Hardware
Solutions: Intel® Software
How To Get the Frameworks
Resources Available

Topics Covered

Define The Problem

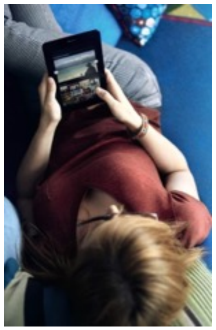
Solutions: Intel® Hardware

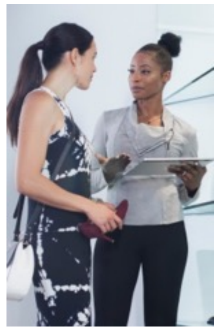
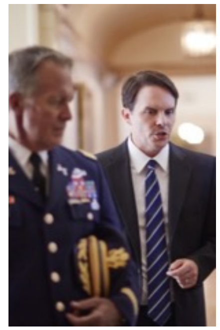
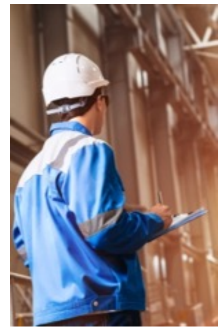
Solutions: Intel® Software

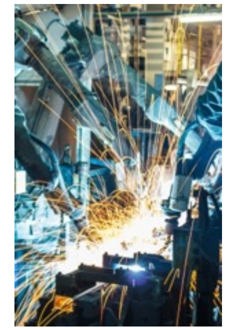
How To Get the Frameworks

Resources Available

Artificial Intelligence Will Transform…



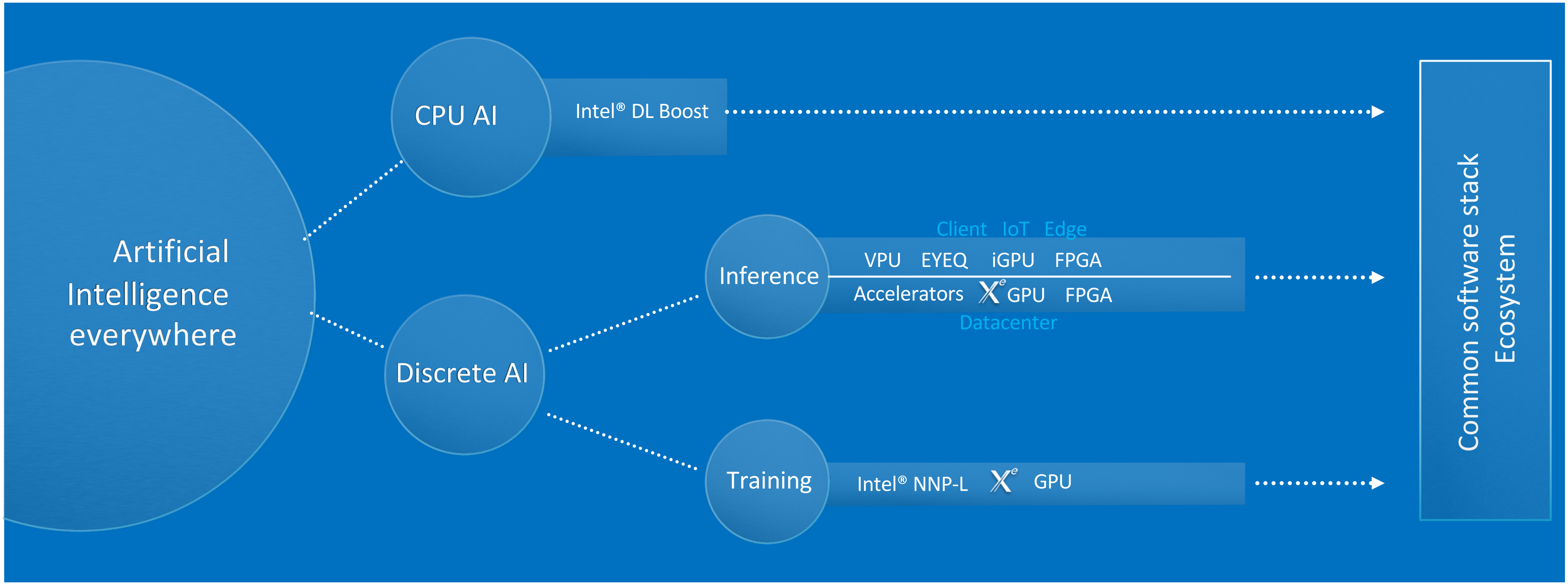



Consumer	Health	Finance	Retail	Government	Energy	Transport	Industrial	Other
Smart Assistants Chatbots Search Personalization Augmented Reality Robots	Enhanced Diagnostics Drug Discovery Patient Care Research Sensory Aids	Algorithmic Trading Fraud Detection Research Personal Finance Risk Mitigation	Support Experience Marketing Merchandising Loyalty Supply Chain Security	Defense Data Insights Safety & Security Resident Engagement Smarter Cities	Oil & Gas Exploration Smart Grid Operational Improvement Conservation	In-Vehicle Experience Automated Driving Aerospace Shipping Search & Rescue	Factory Automation Predictive Maintenance Precision Agriculture Field Automation	Advertising Education Gaming Professional & IT Services Telco/Media Sports

Source: Intel forecast

World's data will grow
10X in 10 years
Yet less than 1% of
all data
is ever analyzed
and used
Source: Seagate “Data Age 2025” July 2017
Forbes Magazine: 20 Mind-Boggling Facts Every Business Leader Must Reflect On Now (Nov 1, 2015); Inside Big Data: Exponential Growth of Data (Feb 16, 2017)

Intel AI Strategy: “Artificial Intelligence Everywhere”



AI (ML & DL) Software Stack for Intel® Processors

Deep learning and AI ecosystem includes edge and datacenter applications.

- Open source frameworks (TensorFlow*, MXNet*, PyTorch*, PaddlePaddle*)
- Intel deep learning products (BigDL, OpenVINO™ toolkit)
- In-house user applications

Intel® MKL and Intel® MKL-DNN optimize deep learning and machine learning applications for Intel® processors :

- Through the collaboration with framework maintainers to upstream changes (Tensorflow*, MXNet*, PyTorch, PaddlePaddle*)
- Through Intel-optimized forks (Caffe*)
- By partnering to enable proprietary solutions

Intel® Math Kernel Library for Deep Neural Networks (Intel® MKL-DNN) is an open source performance library for deep learning applications (available at https://github.com/intel/mkl-dnn)

- Fast open source implementations for wide range of DNN functions
- Early access to new and experimental functionality
- Open for community contributions

Intel® Math Kernel Library (Intel® MKL) is a proprietary performance library for wide range of math and science applications

Distribution: Intel Registration Center, package repositories (apt, yum, conda, pip), Intel® Parallel Studio XE, Intel® System Studio

Intel-Optimized AI Frameworks

Popular DL Frameworks are now optimized for CPU!

choose your favorite framework

TensorFlow™ * | Caffe* | mxnet* | BigDL FOR APACHE Spark* | PYTORCH

See installation guides at ai.intel.com/framework-optimizations/

More under optimization: Caffe2* PaddlePaddle*

SEE ALSO: Machine Learning Libraries for Python (Scikit-learn, Pandas, NumPy), R (Cart, randomForest, e1071), Distributed (MlLib on Spark, Mahout)
**Limited availability today*
Other names and brands may be claimed as the property of others.

intel Software

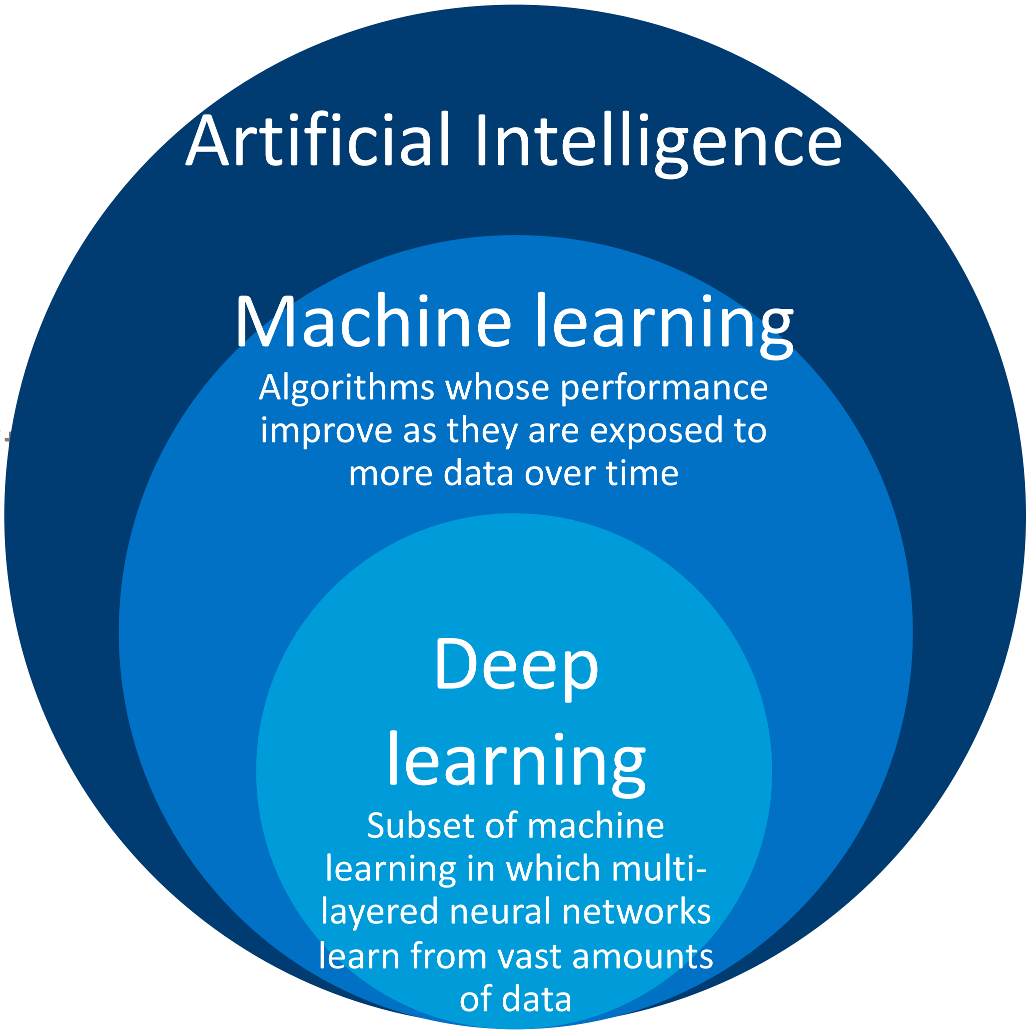
Topics Covered

Define The Problem
Solutions: Intel® Hardware
Solutions: Intel® Software
How To Get the Frameworks
Resources Available

Software

Artificial Intelligence

is the ability of machines to
learn from experience without explicit programming, in order
to perform cognitive functions
associated with the human mind



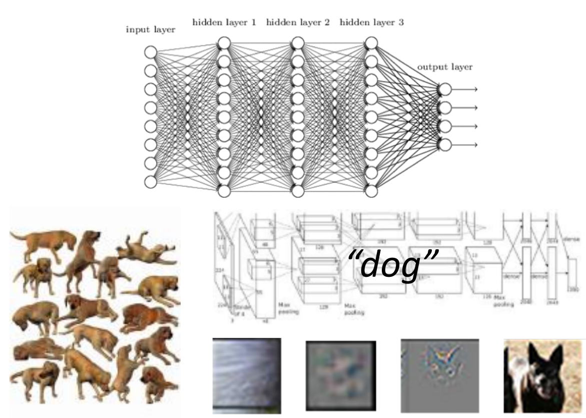
Machine Learning Technology Breakdown

Machine Learning

Autonomous computation methods that learn from experience (data)

Deep Learning

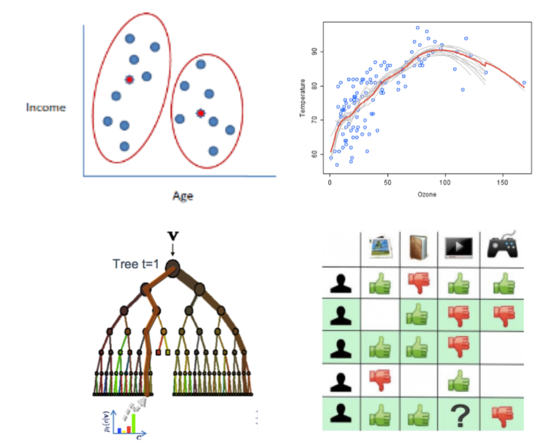
Hierarchical approach with many hidden layers - gaining fame from accurately classifying data-like images, speech, and natural language. Features are learned.



Typical customers: CSP, HPC

Other (or classic) ML

Traditional ML techniques for clustering, regression, and classification using very few (one or two) hidden layers. Requires feature engineering.



Typical customers: Enterprise, HPC

Intel® DAAL Focus

Training

Train an algorithm to build a model

- *Time-to-model is critical*

Inference

Deploy models for classification, prediction, recognition

- *Easily distributed*
- *Criteria: Throughput, TCO @ scale*

Intel Software

Machine Learning Technology Breakdown

Training

Train an algorithm to build a model

- *Time-to-model is critical*

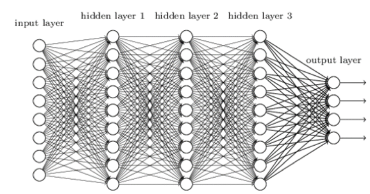
Inference

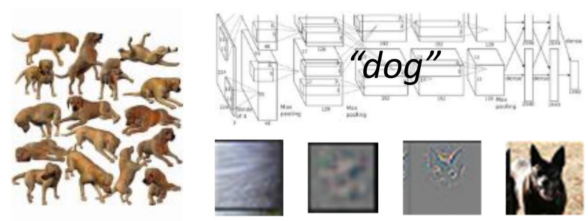
Deploy models for classification, prediction, recognition

- *Easily distributed*
- *Criteria: Throughput, TCO @ scale*

Deep Learning

Hierarchical approach with many hidden layers - gaining fame from accurately classifying data-like images, speech, and natural language. Features are learned.



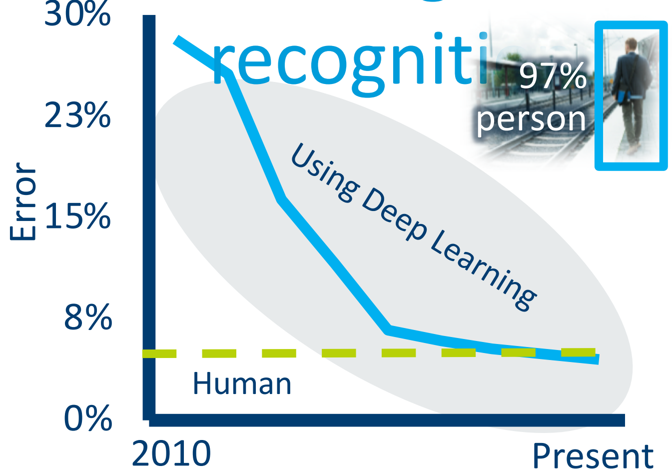


Typical customers: CSP, HPC

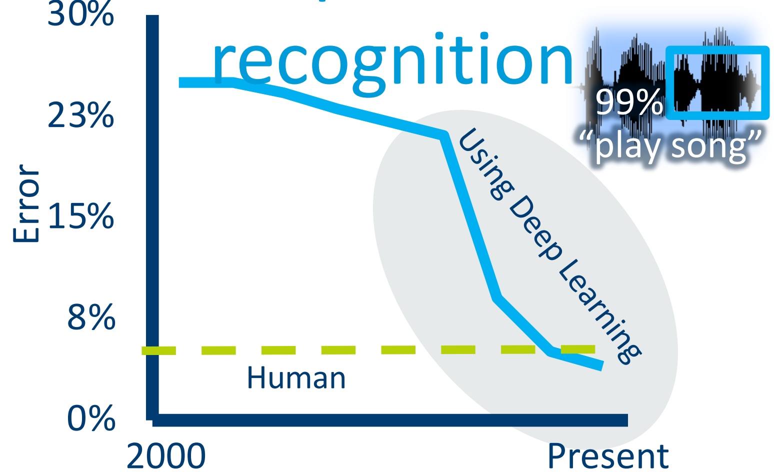
Deep Learning Breakthroughs

Machines able to meet or exceed human image & speech recognition

Image recognition



Speech recognition

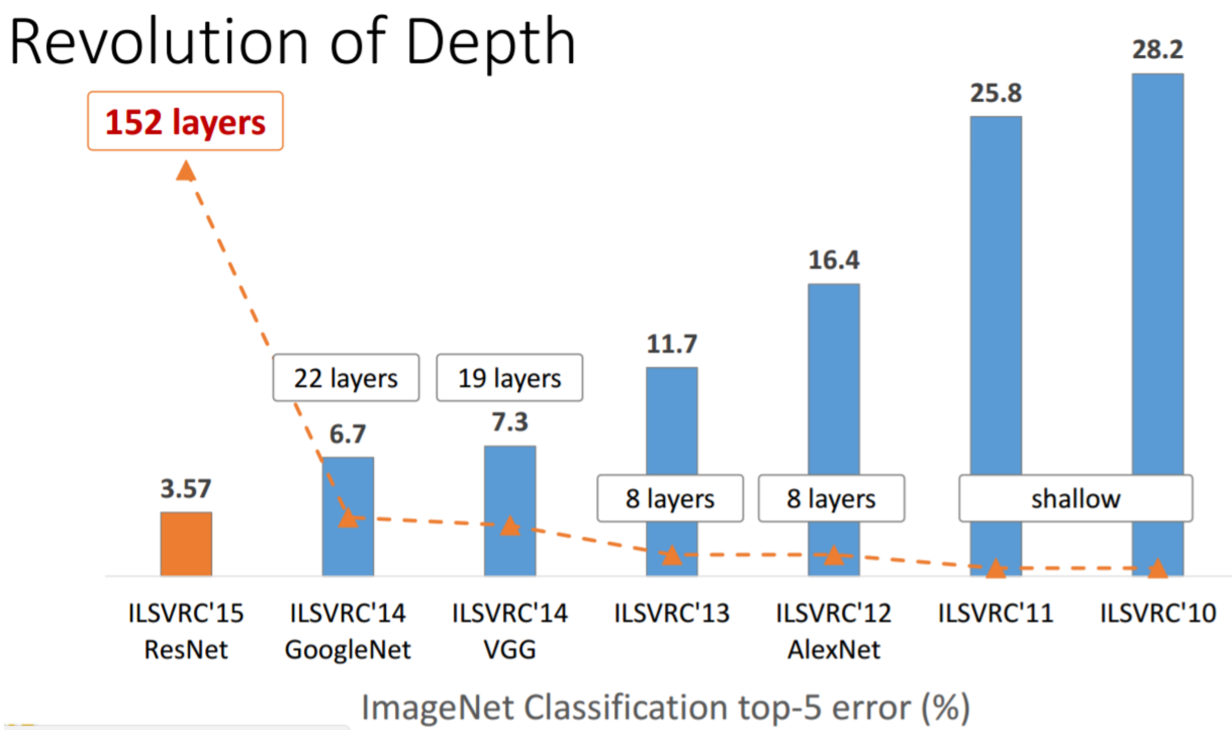


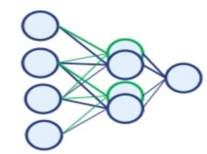
e.g.

Source: ILSVRC ImageNet winning entry classification error rate each year 2010-2016 (Left), https://www.microsoft.com/en-us/research/blog/microsoft-researchers-achieve-new-conversational-speech-recognition-milestone/ (Right)

Depth of Networks

ImageNet Large Scale Visual Recognition Competition (ILSVRC)





http://image-net.org/challenges/talks/ilsvrc2015_deep_residual_learning_kaiminghe.pdf

Intel® Xeon® Processor Scalable Family

Now build the AI you want on the CPU you know

your
FOUNDATION
for AI

Get maximum utilization
running data center and AI workloads side-by-side

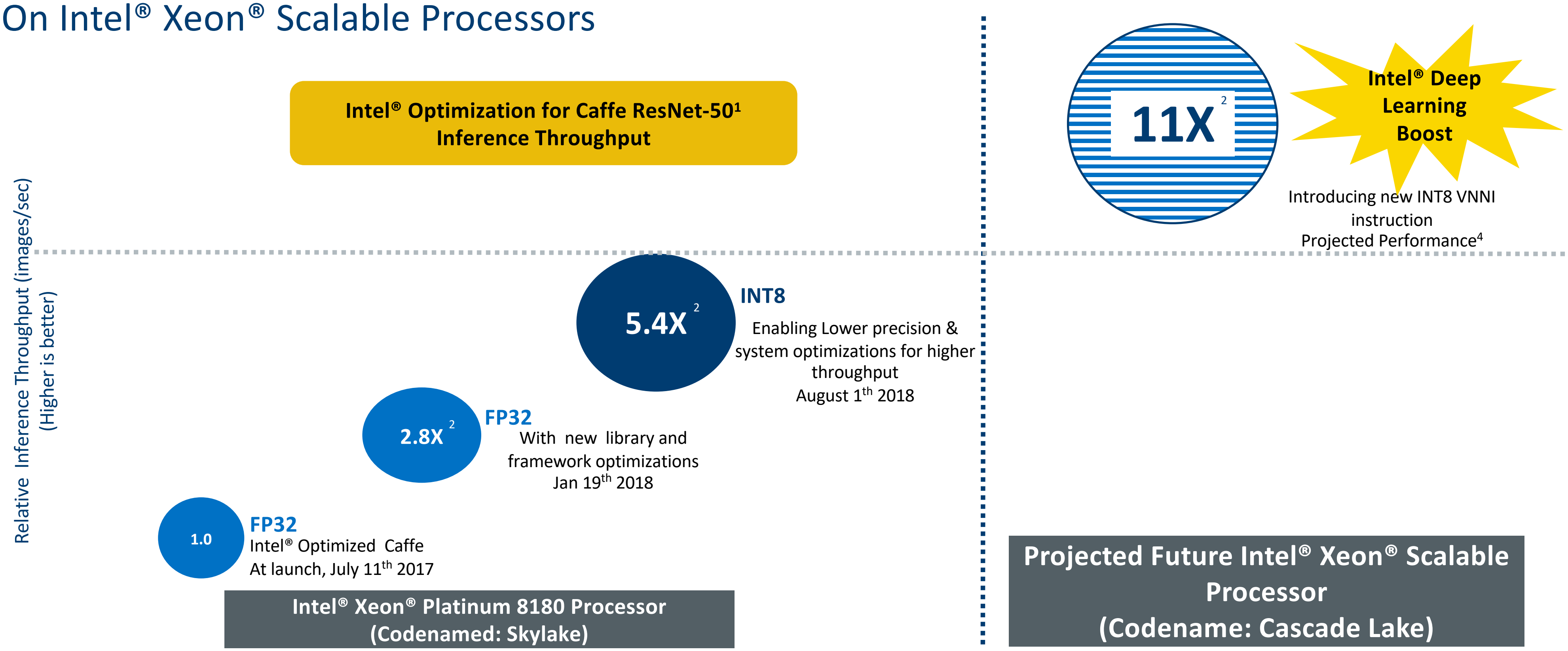
Break memory barriers
to apply AI to large data sets and models

Train models at scale
through efficient scaling to many nodes

Access optimized tools
including continuous performance gains for TensorFlow, MXNet*, more*

Run in the cloud
including AWS, Microsoft, Alibaba, TenCent, Google, Baidu, more

Continued Innovation Driving Deep Learning Inference Performance On Intel® Xeon® Scalable Processors



[4]Inference projections assume 100% socket to socket scaling

[1] Intel® Optimization for Caffe Resnet-50 performance does not necessarily represent other Framework performance.

[2] Based on Intel internal testing: 1X (7/11/2017), 2.8X (1/19/2018) and 5.4X (7/26/2018) performance improvement based on Intel® Optimization for Café Resnet-50 inference throughput performance on Intel® Xeon® Scalable Processor.

[3] 11X (7/25/2018) Results have been estimated using internal Intel analysis, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Performance results are based on testing as of 7/11/2017(1x), 1/19/2018(2.8x) & 7/26/2018(5.4) and may not reflect all publically available security update. No product can be absolutely. See configuration disclosure for details. No product can be absolutely secure .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.

Configurations for Performance Growth- Inference throughput

1x inference throughput improvement in July 2017:

Tested by Intel as of July 11th 2017: Platform: 2S Intel® Xeon® Platinum 8180 CPU @ 2.50GHz (28 cores), HT disabled, turbo disabled, scaling governor set to "performance" via intel_pstate driver, 384GB DDR4-2666 ECC RAM. CentOS Linux release 7.3.1611 (Core), Linux kernel 3.10.0-514.10.2.el7.x86_64. SSD: Intel® SSD DC S3700 Series (800GB, 2.5in SATA 6Gb/s, 25nm, MLC).**Performance measured with**: Environment variables: KMP_AFFINITY='granularity=fine, compact', OMP_NUM_THREADS=56, CPU Freq set with cpupower frequency-set -d 2.5G -u 3.8G -g performance. Caffe: (http://github.com/intel/caffe/), revision f96b759f71b2281835f690af267158b82b150b5c. Inference measured with "caffe time --forward_only" command, training measured with "caffe time" command. For "ConvNet" topologies, dummy dataset was used. For other topologies, data was stored on local storage and cached in memory before training. Topology specs from https://github.com/intel/caffe/tree/master/models/intel_optimized_models (ResNet-50),and https://github.com/soumith/convnet-benchmarks/tree/master/caffe/imagenet_winners (ConvNet benchmarks; files were updated to use newer Caffe prototxt format but are functionally equivalent). Intel C++ compiler ver. 17.0.2 20170213, Intel MKL small libraries version 2018.0.20170425. Caffe run with "numactl -l".

2.8x inference throughput improvement in January 2018:

Tested by Intel as of Jan 19th 2018 Processor :2 socket Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz / 28 cores HT ON , Turbo ON Total Memory 376.46GB (12slots / 32 GB / 2666 MHz). CentOS Linux-7.3.1611-Core, SSD sda RS3WC080 HDD 744.1GB,sdb RS3WC080 HDD 1.5TB,sdc RS3WC080 HDD 5.5TB , Deep Learning Framework Intel® Optimization for caffe version:f6d01efbe93f70726ea3796a4b89c612365a6341 Topology::resnet_50_v1 BIOS:SE5C620.86B.00.01.0009.101920170742 MKLDNN: version: ae00102be506ed0fe2099c6557df2aa88ad57ec1 NoDataLayer. . Datatype:FP32 Batchsize=64 Measured: 652.68 imgs/sec vs Tested by Intel as of July 11th 2017: Platform: 2S Intel® Xeon® Platinum 8180 CPU @ 2.50GHz (28 cores), HT disabled, turbo disabled, scaling governor set to "performance" via intel_pstate driver, 384GB DDR4-2666 ECC RAM. CentOS Linux release 7.3.1611 (Core), Linux kernel 3.10.0-514.10.2.el7.x86_64. SSD: Intel® SSD DC S3700 Series (800GB, 2.5in SATA 6Gb/s, 25nm, MLC).**Performance measured with**: Environment variables: KMP_AFFINITY='granularity=fine, compact', OMP_NUM_THREADS=56, CPU Freq set with cpupower frequency-set -d 2.5G -u 3.8G -g performance. Caffe: (http://github.com/intel/caffe/), revision f96b759f71b2281835f690af267158b82b150b5c. Inference measured with "caffe time --forward_only" command, training measured with "caffe time" command. For "ConvNet" topologies, dummy dataset was used. For other topologies, data was stored on local storage and cached in memory before training. Topology specs from https://github.com/intel/caffe/tree/master/models/intel_optimized_models (ResNet-50),and https://github.com/soumith/convnet-benchmarks/tree/master/caffe/imagenet_winners (ConvNet benchmarks; files were updated to use newer Caffe prototxt format but are functionally equivalent). Intel C++ compiler ver. 17.0.2 20170213, Intel MKL small libraries version 2018.0.20170425. Caffe run with "numactl -l".

5.4x inference throughput improvement in August 2018:

Tested by Intel as of measured July 26th 2018 :2 socket Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz / 28 cores HT ON , Turbo ON Total Memory 376.46GB (12slots / 32 GB / 2666 MHz). CentOS Linux-7.3.1611-Core, kernel: 3.10.0-862.3.3.el7.x86_64, SSD sda RS3WC080 HDD 744.1GB,sdb RS3WC080 HDD 1.5TB,sdc RS3WC080 HDD 5.5TB , Deep Learning Framework Intel® Optimization for caffe version:a3d5b022fe026e9092fc7abc7654b1162ab9940d Topology::resnet_50_v1 BIOS:SE5C620.86B.00.01.0013.030920180427 MKLDNN: version:464c268e544bae26f9b85a2acb9122c766a4c396 instances: 2 instances socket:2 (Results on Intel® Xeon® Scalable Processor were measured running multiple instances of the framework. Methodology described here: https://software.intel.com/en-us/articles/boosting-deep-learning-training-inference-performance-on-xeon-and-xeon-phi) NoDataLayer. Datatype: INT8 Batchsize=64 Measured: 1233.39 imgs/sec vs Tested by Intel as of July 11th 2017:2S Intel® Xeon® Platinum 8180 CPU @ 2.50GHz (28 cores), HT disabled, turbo disabled, scaling governor set to "performance" via intel_pstate driver, 384GB DDR4-2666 ECC RAM. CentOS Linux release 7.3.1611 (Core), Linux kernel 3.10.0-514.10.2.el7.x86_64. SSD: Intel® SSD DC S3700 Series (800GB, 2.5in SATA 6Gb/s, 25nm, MLC).**Performance measured with**: Environment variables: KMP_AFFINITY='granularity=fine, compact', OMP_NUM_THREADS=56, CPU Freq set with cpupower frequency-set -d 2.5G -u 3.8G -g performance. Caffe: (http://github.com/intel/caffe/), revision f96b759f71b2281835f690af267158b82b150b5c. Inference measured with "caffe time --forward_only" command, training measured with "caffe time" command. For "ConvNet" topologies, dummy dataset was used. For other topologies, data was stored on local storage and cached in memory before training. Topology specs from https://github.com/intel/caffe/tree/master/models/intel_optimized_models (ResNet-50). Intel C++ compiler ver. 17.0.2 20170213, Intel MKL small libraries version 2018.0.20170425. Caffe run with "numactl -l".

11X inference throughput improvement with CascadeLake:

Future Intel Xeon Scalable processor (codename Cascade Lake) results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance vs Tested by Intel as of July 11th 2017: 2S Intel® Xeon® Platinum 8180 CPU @ 2.50GHz (28 cores), HT disabled, turbo disabled, scaling governor set to "performance" via intel_pstate driver, 384GB DDR4-2666 ECC RAM. CentOS Linux release 7.3.1611 (Core), Linux kernel 3.10.0-514.10.2.el7.x86_64. SSD: Intel® SSD DC S3700 Series (800GB, 2.5in SATA 6Gb/s, 25nm, MLC).**Performance measured with**: Environment variables: KMP_AFFINITY='granularity=fine, compact', OMP_NUM_THREADS=56, CPU Freq set with cpupower frequency-set -d 2.5G -u 3.8G -g performance. Caffe: (http://github.com/intel/caffe/), revision f96b759f71b2281835f690af267158b82b150b5c. Inference measured with "caffe time --forward_only" command, training measured with "caffe time" command. For "ConvNet" topologies, dummy dataset was used. For other topologies, data was stored on local storage and cached in memory before training. Topology specs from https://github.com/intel/caffe/tree/master/models/intel_optimized_models (ResNet-50),. Intel C++ compiler ver. 17.0.2 20170213, Intel MKL small libraries version 2018.0.20170425. Caffe run with "numactl -l".

Topics Covered

Define The Problem

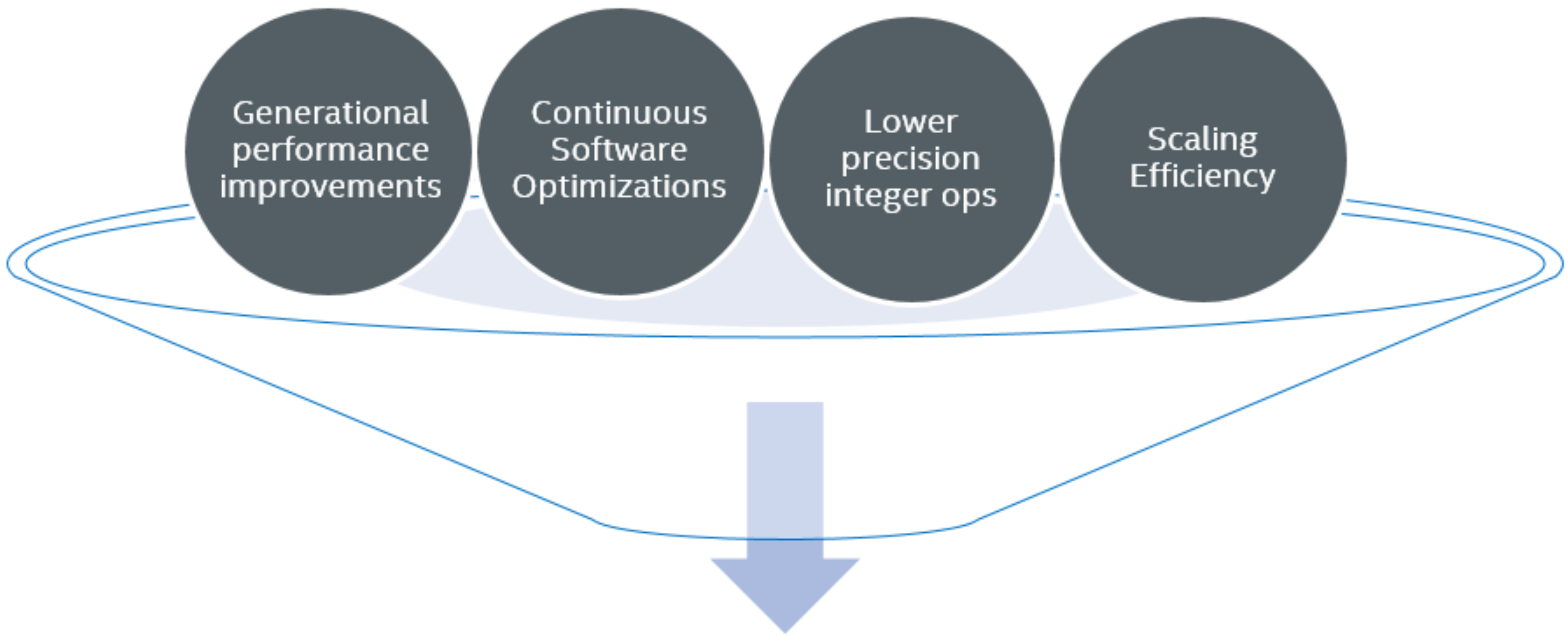
Solutions: Intel® Hardware

Solutions: Intel® Software

How To Get the Frameworks

Resources Available

Intel® Xeon® Scalable Processors for AI



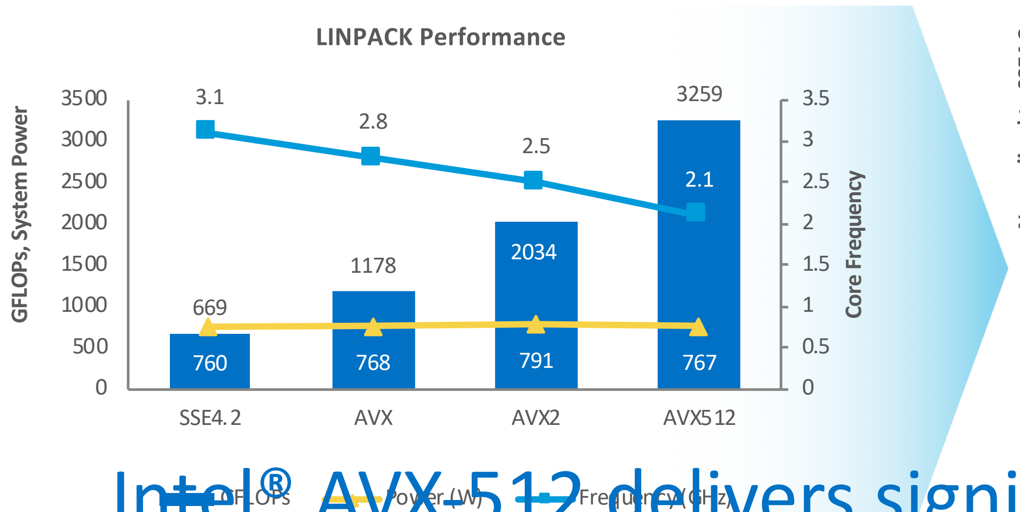
Artificial Intelligence

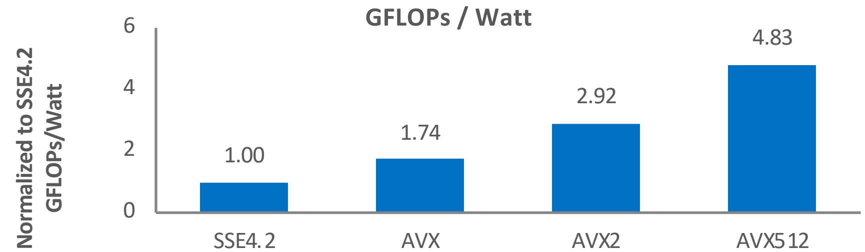
With Intel® Xeon® Scalable Processors

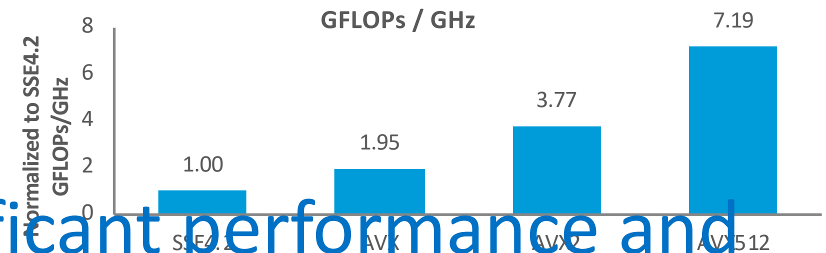
Deep Learning INFERENCE & Deep Learning TRAINING

Intel® Advanced Vector Extensions 512 (Intel® AVX-512)

512-bit wide vectors, 32 operand registers, 8 64b mask registers, Embedded broadcast & rounding







Intel® AVX-512 delivers significant performance and efficiency gains

Intel internal measurements. 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. Configuration Summary: 1-Node, 2 x Intel® Xeon® Platinum 8180 Processor on Purley-EP (Lewisburg) (S2600WF) with 384 GB (12x32GB DDR4-2666) Total Memory, Intel S3610 800GB SSD, BIOS: SE5C620.86B.01.00.0471.040720170924, 04/07/2017, RHEL Kernel: 3.10.0-514.16.1.el7.x86_64 x86_64, Benchmark: Intel® Optimized MP LINPACK

Int8 for Inference on Intel® Xeon® Scalable Processors

FP32

Sign	Exponent								Mantissa																						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

Lower precision integer ops

Typical Intel® AVX-512 instruction to perform FP32 convolutions: **vfmadd231ps**

64 Ops/Cycle

Increase Operations/cycle to improve throughput performance

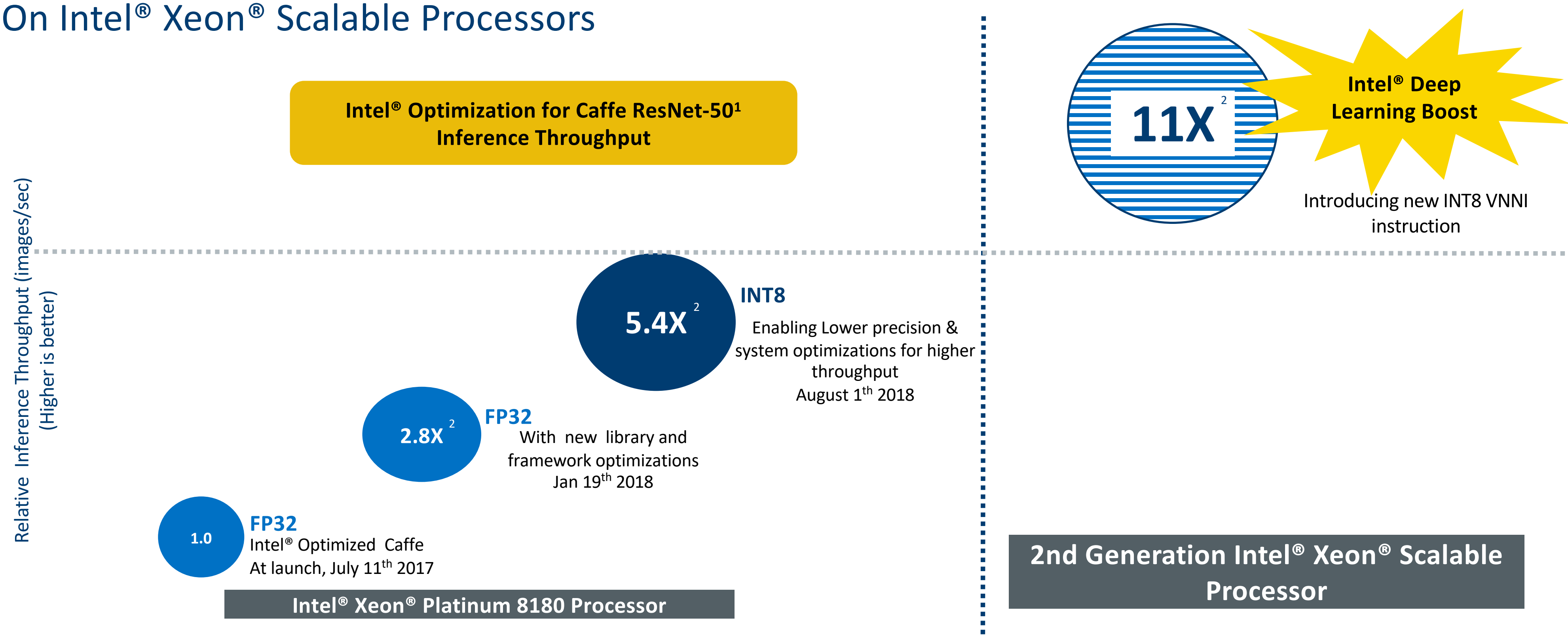
INT8

Sign	Mantissa						
07	06	05	04	03	02	01	00

Typical Intel® AVX-512 instructions to perform INT8 convolutions: **vpmaddubsw, vpmaddwd, vpaddd**

85 Ops/Cycle

Continued Innovation Driving Deep Learning Inference Performance On Intel® Xeon® Scalable Processors



[4]Inference projections assume 100% socket to socket scaling

[1] Intel® Optimization for Caffe Resnet-50 performance does not necessarily represent other Framework performance.

[2] Based on Intel internal testing: 1X (7/11/2017), 2.8X (1/19/2018) and 5.4X (7/26/2018) performance improvement based on Intel® Optimization for Café Resnet-50 inference throughput performance on Intel® Xeon® Scalable Processor. See Configuration Details 53

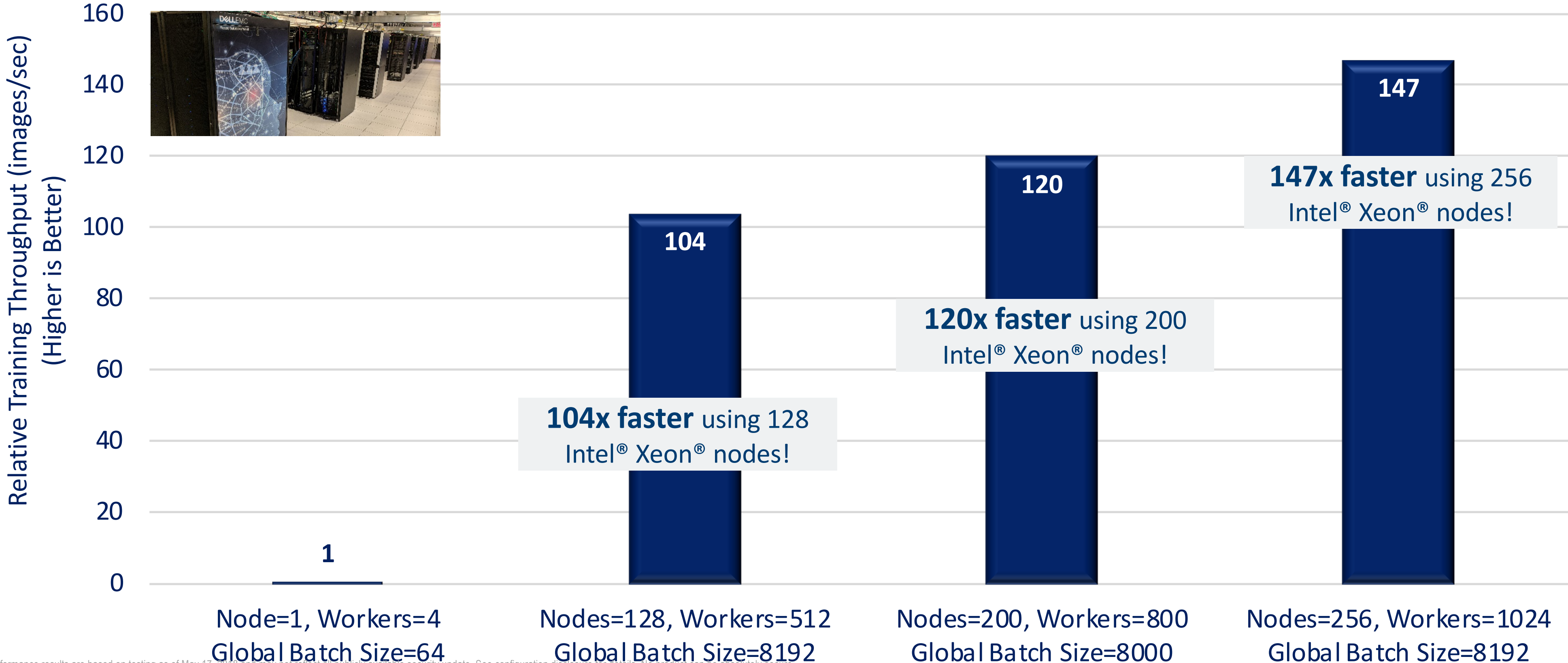
[3] 11X (7/25/2018) Results have been estimated using internal Intel analysis, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Performance results are based on testing as of 7/11/2017(1x), 1/19/2018(2.8x) & 7/26/2018(5.4) and may not reflect all publically available security update. No product can be absolutely. See configuration disclosure for details. No product can be absolutely secure .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.

Training Performance: ResNet-50/ChestXRay14

Intel® 2S Xeon® Gold 6148F processor based DellEMC* PowerEdge C6420 Zenith* Cluster on OPA™ Fabric

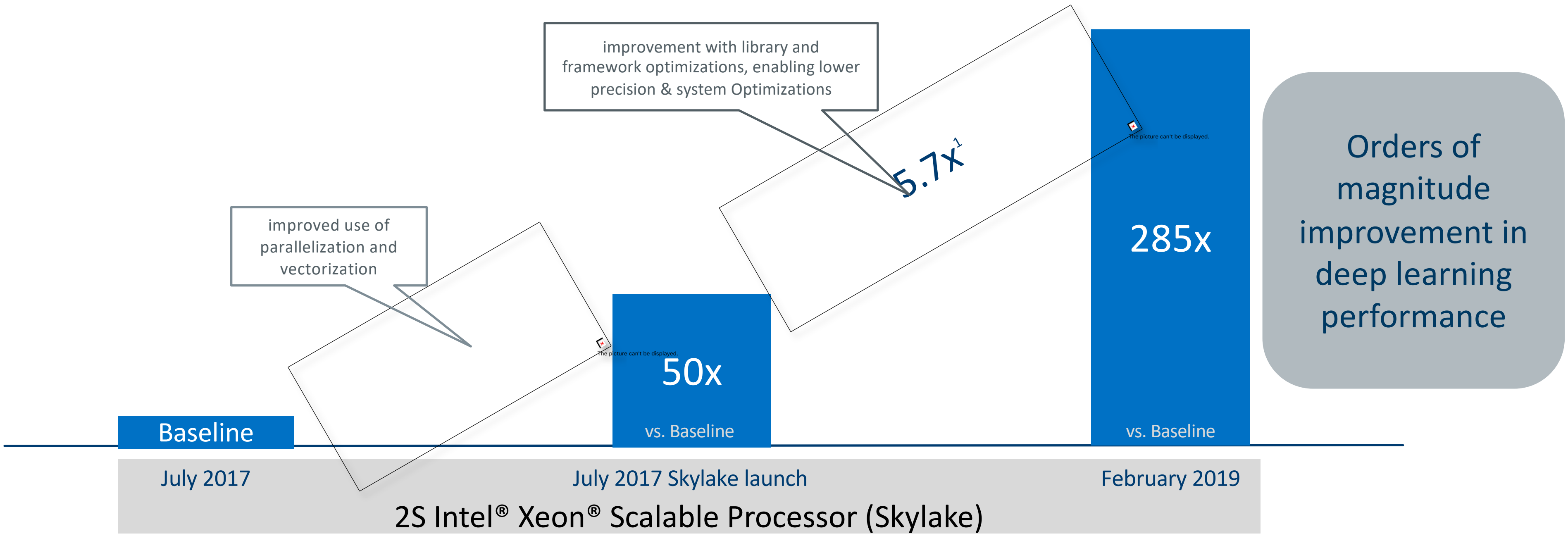
TensorFlow* 1.6 + horovod*, IMPI



intel Software

AI Performance Growth on Intel® Xeon® Processors

Software Optimizations and Hardware features driving Deep Learning Performance on Intel® Xeon® Scalable Processors



[1] 5.7x inference throughput improvement with Intel® Optimizations for Caffe ResNet-50 on Intel® Xeon® Platinum 8180 Processor in Feb 2019 compared to performance at launch in July 2017. See configuration details on Config 1

Optimized Deep Learning Frameworks and Toolkits

Gen on Gen Performance gains for ResNet-50 with Intel® DL Boost

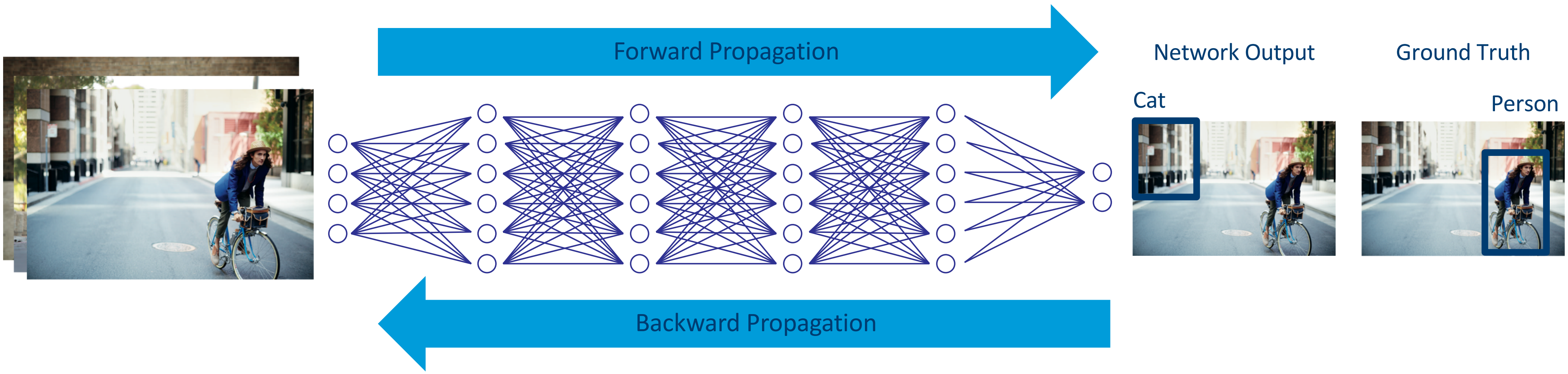
2S Intel® Xeon® Platinum 8280 Processor vs 2S Intel® Xeon® Platinum 8180 Processor

Intel® Xeon® Scalable Processor	2nd Gen Intel® Xeon® Scalable Processor	mxnet	PyTorch	TensorFlow	Caffe	OpenVINO
FP32	INT8 w/ Intel® DL Boost	3.0x	3.7x	3.9x	4.0x	3.9x
INT8	INT8 w/ Intel® DL Boost	1.8x	2.1x	1.8x	2.3x	1.9x

See Configuration Details 5

Performance results are based on testing as of dates shown in configuration and may not reflect all publicly available security updates. No product can be absolutely secure. See configuration disclosure for details. 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. 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: http://www.intel.com/performance

Deep Learning Training



Complex Networks with billions of parameters can take days to train on a modern processor

Hence, the need to reduce time-to-train. Maybe using a cluster of processing nodes?

intel Software

Supercomputer in a CPU Box

143 GFLOPS[1]

Paragon
#1 in 1993

1 TFLOPS[2]

ASCI Red
#1 in 1997

8 TFLOPS[3]

Xeon Server
2017

PFLOPS-EFLOPS?

Data Center and
Cloud 2019

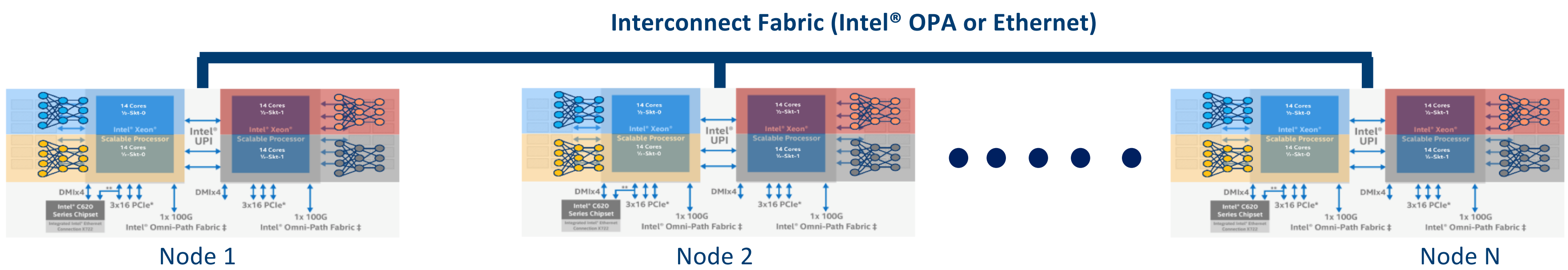
https://www.top500.org/featured/systems/intel-xps-140-paragon-sandia-national-labs/

https://en.wikipedia.org/wiki/Advanced_Simulation_and_Computing_Program

https://ark.intel.com/products/120496/Intel-Xeon-Platinum-8180-Processor-38_5M-Cache-2_50-GHz

intel Software

Scaleout Training: Multi-Workers & Multi-Nodes



Distributed Deep Learning Training Across Multiple nodes

Each node running multiple workers/node

Uses optimized MPI Library for gradient updates over network fabric

Caffe – Use Optimized Intel® MPI ML Scaling Library (Intel® MLSL)

TensorFlow* – Uber Horovod MPI Library

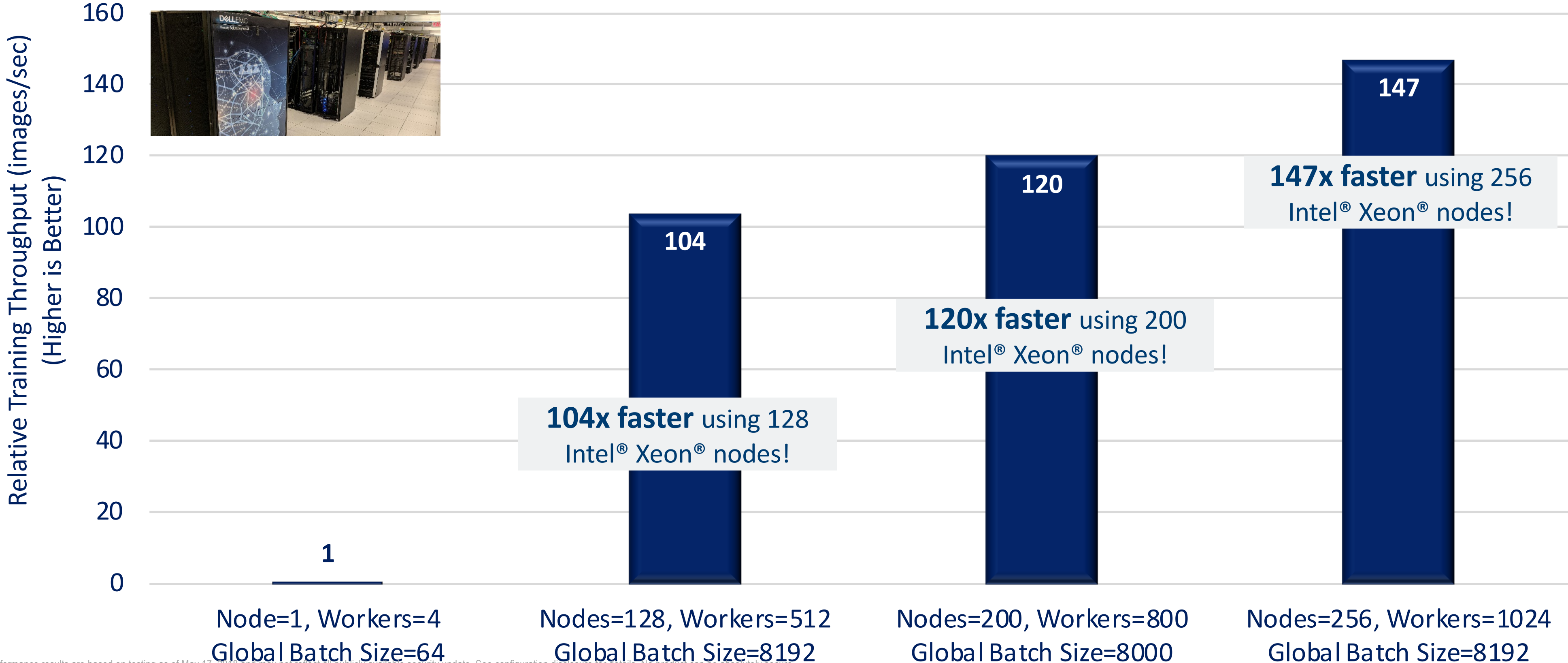
Intel Best Known Methods: https://ai.intel.com/accelerating-deep-learning-training-inference-system-level-optimizations/
https://www.intel.ai/using-intel-xeon-for-multi-node-scaling-of-tensorflow-with-horovod

Software

Training Performance: ResNet-50/ChestXRay14

Intel® 2S Xeon® Gold 6148F processor based DellEMC* PowerEdge C6420 Zenith* Cluster on OPA™ Fabric

TensorFlow* 1.6 + horovod*, IMPI



Topics Covered

Define The Problem
Solutions: Intel® Hardware
Solutions: Intel® Software
How To Get the Frameworks
Resources Available

What's Happening Under The Hood? Intel® MKL-DNN Functionality

Features:

- Training (float32) and inference (float32, int8)
- CNNs (1D, 2D and 3D), RNNs (plain, LSTM, GRU)
- Optimized for Intel processors

Portability:

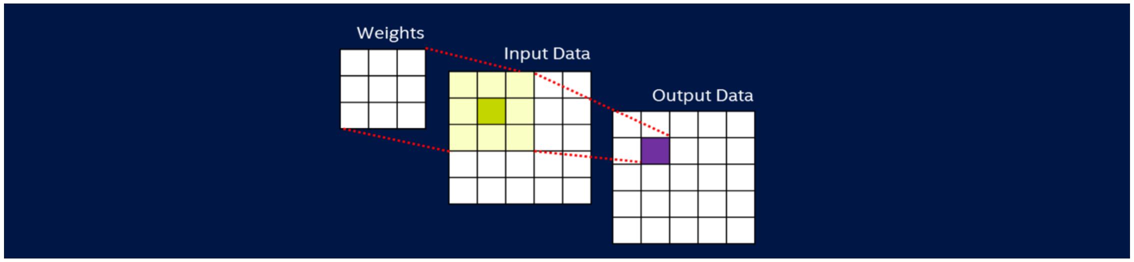
- Compilers: Intel® C++ Compiler/Clang/GCC/MSVC*
- OSes: Linux*, Windows*, Mac*
- Threading: OpenMP*, TBB

Frameworks that use Intel ® MKL-DNN:

Caffe*, TensorFlow*, MxNet*, PaddlePaddle*, Pytorch*, …

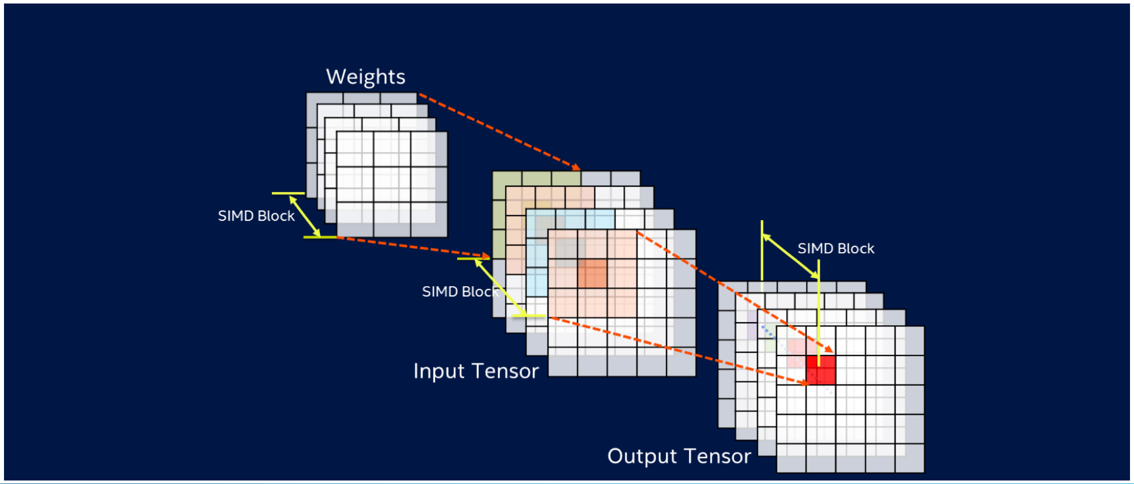
	Intel® MKL-DNN v0.16
Convolution	Direct 3D, Depthwise separable convolution Winograd convolution Deconvolution
Fully Connected Layer	Inner Product
Pooling	Maximum Average (include/exclude padding)
Normalization	LRN across/within channel, Batch normalization
Eltwise (Loss/activation)	ReLU(bounded/soft), ELU, Tanh; Softmax, Logistic, linear; square, sqrt, abs
Data manipulation	Reorder, sum, concat, View
RNN cell	RNN cell, LSTM cell, GRU cell
Fused primitive	Conv+ReLU+sum, BatchNorm+ReLU
Data type	f32, s32, s16, s8, u8

Intel® MKL-DNN Optimization Vectorization



Lower precision integer ops

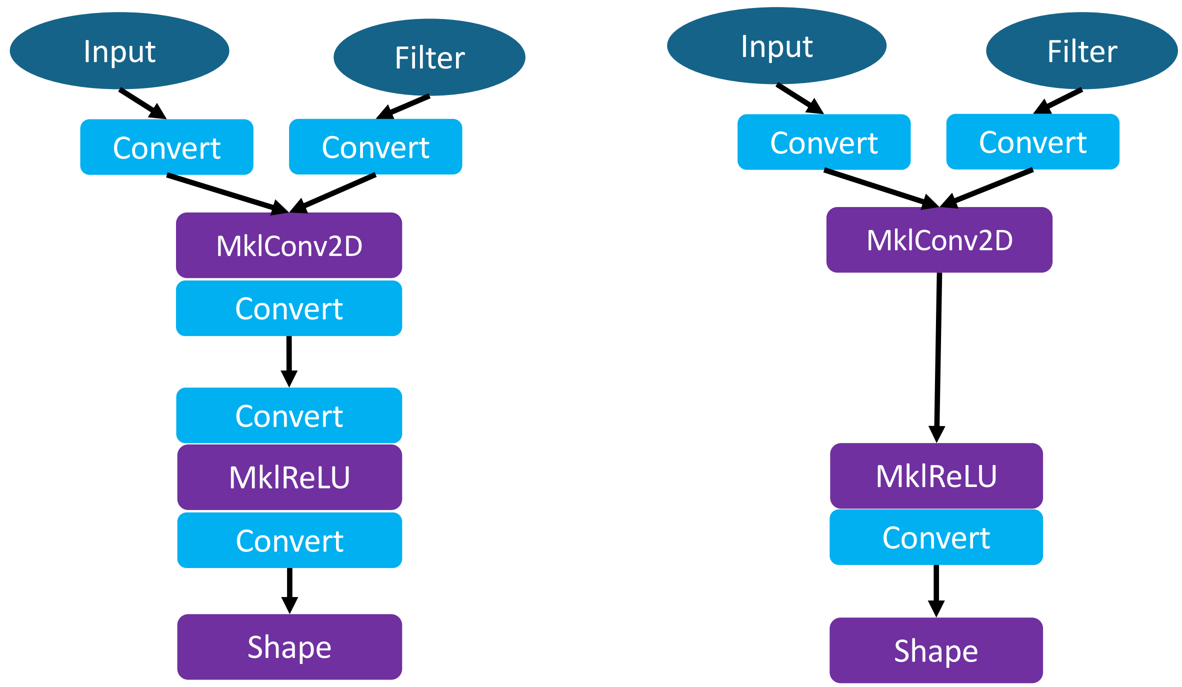
Optimizations: Intel® AVX-512 vectorization, data reuse, parallelization



AI Framework Software Optimizations
Layout Propagation

Converting to/from optimized layout can be less expensive than operating on un-optimized layout.

CPU Friendly Layout is preferred by most MKL-DNN primitives

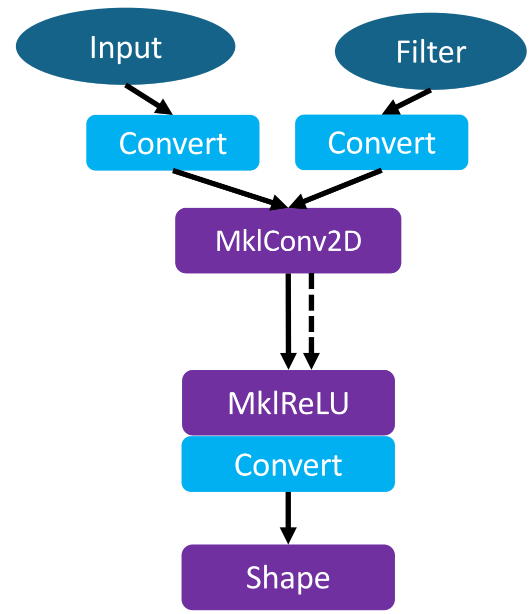


After Layout Conversion

After Layout Propagation

AI Framework Software Optimizations

Load Balancing



TensorFlow* graphs offer opportunities for parallel execution.

Threading model, Tune your Intel® MKL w/

1. **`inter_op_parallelism_threads`** = max number of operators that can be executed in parallel
2. **`intra_op_parallelism_threads`** = max number of threads to use for executing an operator
3. **`OMP_NUM_THREADS`** = MKL-DNN equivalent of **`intra_op_parallelism_threads`**

```
>>> config = tf.ConfigProto()
>>> config.intra_op_parallelism_threads = 56
>>> config.inter_op_parallelism_threads = 2
>>> tf.Session(config=config)
```

```
os.environ["KMP_BLOCKTIME"] = "1"
os.environ["KMP_AFFINITY"] = "granularity=fine,compact,1,0"
os.environ["KMP_SETTINGS"] = "0"
os.environ["OMP_NUM_THREADS"] = "56"
```

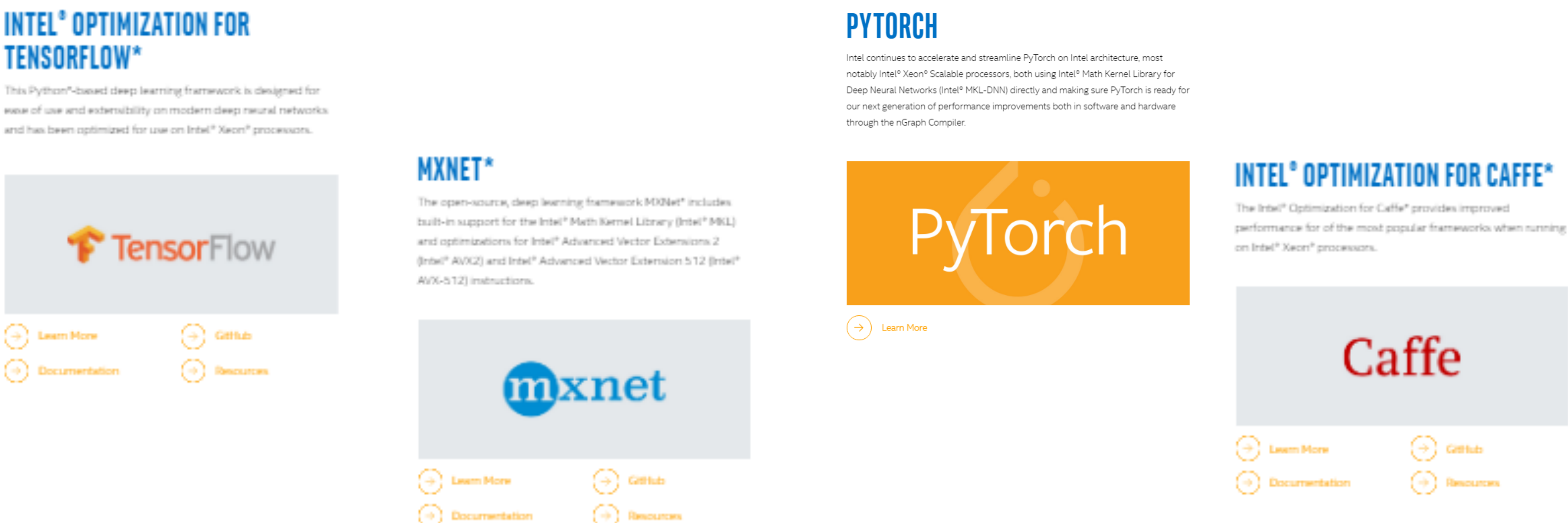
More details: https://www.tensorflow.org/performance/performance_guide

Topics Covered

Define The Problem
Solutions: Intel® Hardware
Solutions: Intel® Software
How To Get the Frameworks
Resources Available

Intel-Optimized Frameworks: How To Get?

Check out our intel.ai for the framework optimizations page



https://www.intel.ai/framework-optimizations

Intel® Optimization of TensorFlow*: How To Get?

Intel TensorFlow* install guide is available → https://software.intel.com/en-us/articles/intel-optimization-for-tensorflow-installation-guide

On Theta

- Refer to Corey Adams talk

Intel® Optimization of PyTorch*: How To Get?

Intel PyTorch* getting started guide is available → https://software.intel.com/en-us/articles/getting-started-with-intel-optimization-of-pytorch

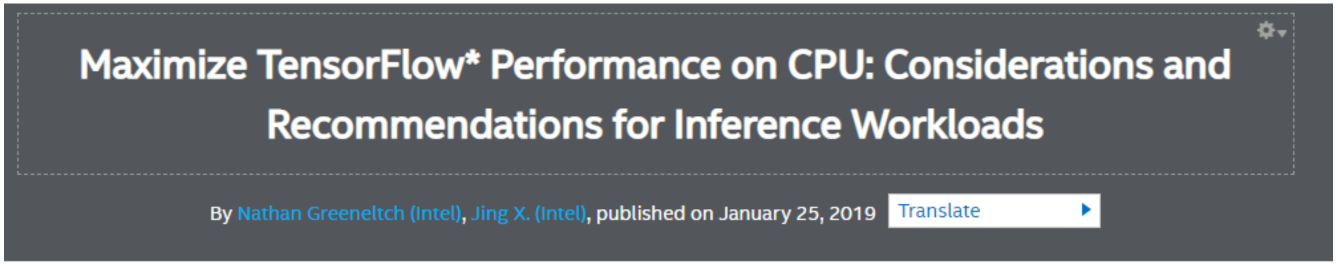
On Theta	• Refer to Corey Adams talk

Topics Covered

Define The Problem
Solutions: Intel® Hardware
Solutions: Intel® Software
How To Get the Frameworks
Resources Available

Article Plug

Intel–Optimized TensorFlow* Performance Considerations



To fully utilize the power of Intel® architecture (IA) and thus yield high performance, TensorFlow* can be powered by Intel's highly optimized math routines for deep learning tasks. This primitives library is called Intel® Math Kernel Library for Deep Neural Networks (Intel® MKL-DNN). Intel MKL-DNN includes convolution, normalization, activation, inner product, and other primitives.

TensorFlow Runtime Options Affecting Performance

Runtime options heavily effect TensorFlow performance. Understanding them will help get the best performance out of the Intel Optimization of TensorFlow.

- intra_/inter_op_parallelism_threads
- Data layout

https://software.intel.com/en-us/articles/maximize-tensorflow-performance-on-cpu-considerations-and-recommendations-for-inference

```
python tf_cnn_benchmarks.py --num_intra_threads=cores --num_inter_threads=2
```

intra_op_parallelism_threads and **inter_op_parallelism_threads** are runtime variables defined in TensorFlow. ConfigProto. The ConfigProto is used for configuration when creating a session. These two variables control number of cores to use.

- intra_op_parallelism_threads

This runtime setting controls parallelism inside an operation. For instance, if matrix multiplication or reduction is intended to be executed in several threads, this variable should be set. TensorFlow will schedule tasks in a thread pool which contains intra_op_parallelism_threads threads. As illustrated later in figure 3, OpenMP* threads are bound to thread context as close as possible on different core, setting this environment variable to the number of available physical cores is recommended.

Self-Help: Intel® AI Developer Program

For developers, students, instructors, and startups

Get smarter using online tutorials, webinars, student kits and support forums

Educate others using available course materials, hands-on labs, and more

Get 4-weeks FREE access to the Intel® AI DevCloud, use your existing Intel® Xeon® Processor-based cluster, or use a public cloud service

Showcase your innovation at industry & academic events and online via the Intel AI community forum

software.intel.com/ai

Free Support: Intel® AI Frameworks Forum

https://forums.intel.com

INTEL® OPTIMIZED AI FRAMEWORKS

Support for key Deep Learning Frameworks
and Libraries optimized for Intel Hardware.

Intel TensorFlow Installation and Performance

Intel® Optimized AI Frameworks · NathanG_intel · February 6, 2019 at 2:01 PM

155 1 0

Intel® AI Builders: Ecosystem

100+ AI Partners

CROSS VERTICAL

oem

System integrators

VERTICAL

HEALTHCARE

FINANCIAL SERVICES

RETAIL

TRANSPORTATION

NEWS, MEDIA & ENTERTAINMENT

AGRICULTURE

LEGAL & HR

ROBOTIC PROCESS AUTOMATION

HORIZONTAL

BUSINESS INTELLIGENCE & ANALYTICS

VISION

CONVERSATIONAL BOTS

AI TOOLS & CONSULTING

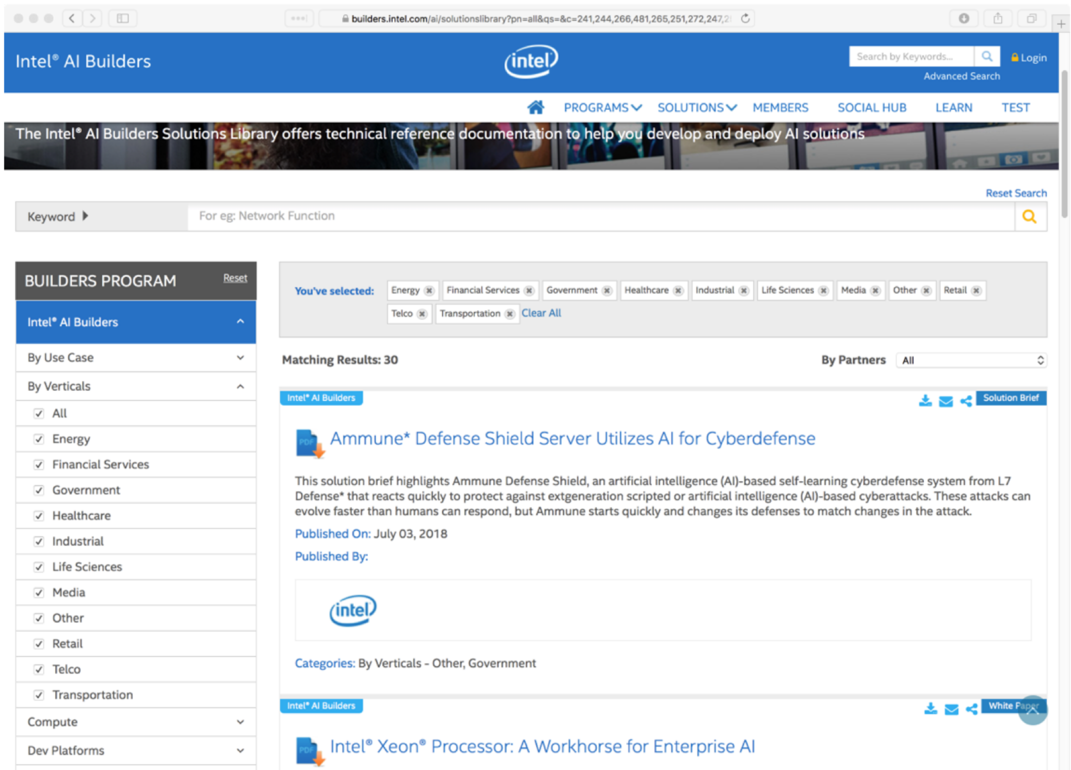
AI

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

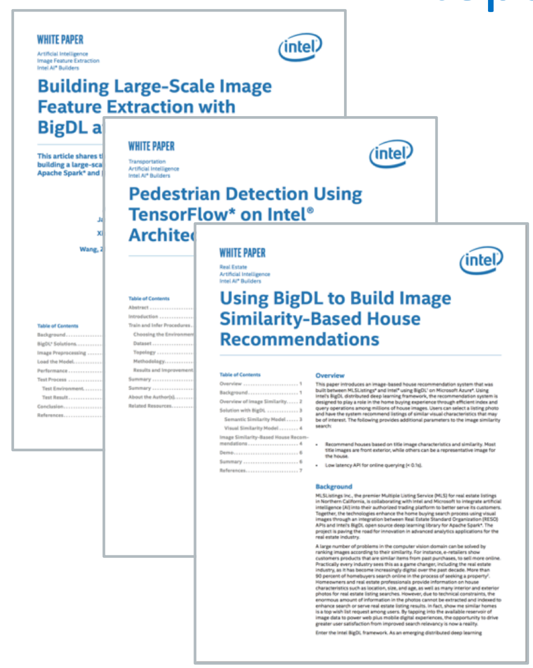
Builders.intel.com/ai

intel Software

Intel® AI Builders: Solutions Library



30+ Public Whitepapers



Builders.intel.com/ai/solutionslibrary

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

Software

CALL TO ACTION

LEARN

More information at
www.intel.ai/framework-optimizations/

EXPLORE

Use Intel's performance-optimized libraries & frameworks

ENGAGE

Use Our Forums for Free Support
forums.intel.com
Choose "Intel Optimized AI Frameworks" from list

Intel® Optimized AI Frameworks

https://anaconda.org/intel
https://software.intel.com/en-us/distribution-for-python
https://intelpython.github.io/daal4py
https://github.com/IntelLabs/hpat

Questions?

Get fast python* execution

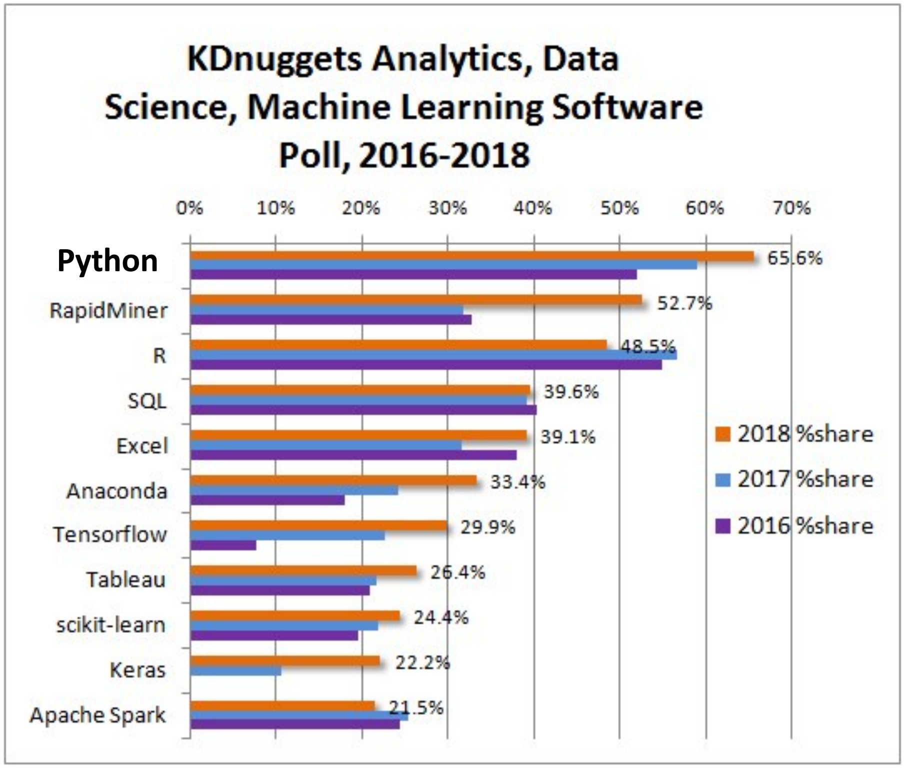
Intel® Distribution of Python 2019

Nathan Greeneltch, PhD

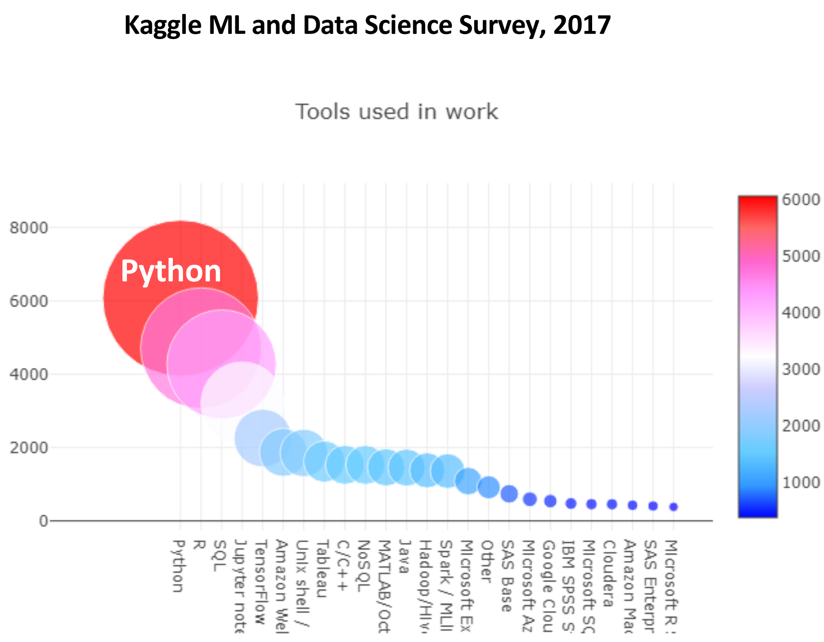
Consulting Engineer, Intel Corporation

Software

Python: Lingua Franca of Data Science



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



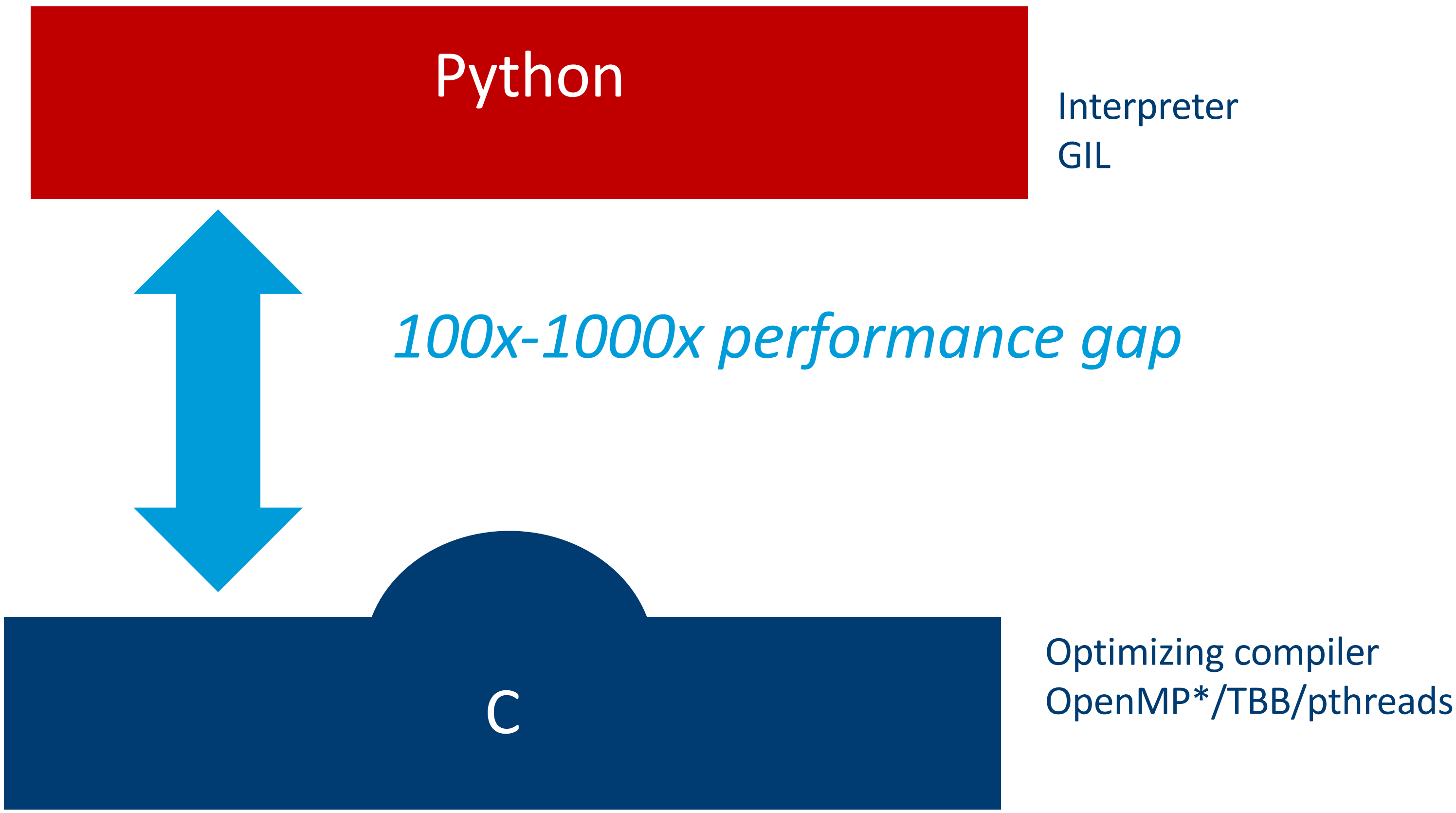
https://www.kaggle.com/sudalairajkumar/an-interactive-deep-dive-into-survey-results/data

The Reality of "Data Centric Computing"

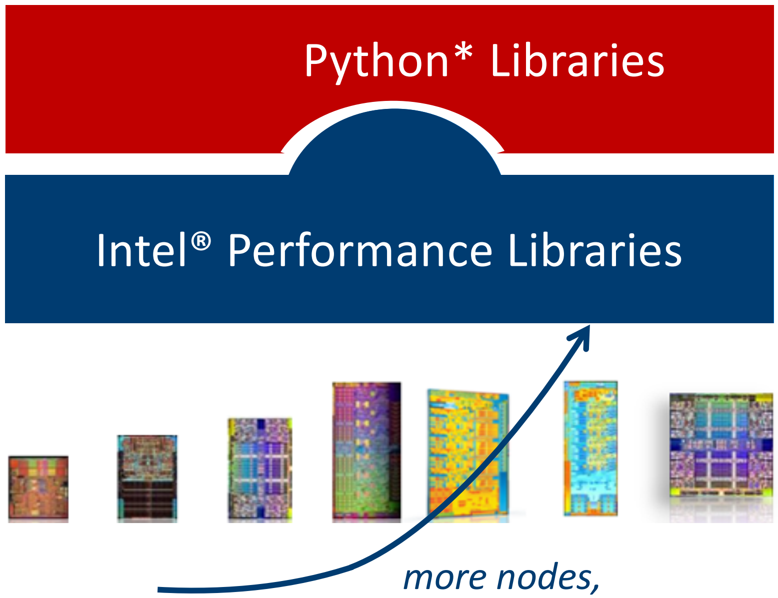
Software Challenges:

Performance Limited	• Software is slow and single-node for many organizations • Only sample a small portion of the data
Productivity Limited	• More performant/scalable implementations require significantly more development & deployment skills & time
Compute Limited	• Performance bottleneck often in compute, not storage/memory

Performance of Python*



High Performance Python



Thin layer in Python or Cython

Native highly optimized libraries (Intel MKL, Intel DAAL, Intel IPP)

(generations of processors)

more nodes,
more cores,
more threads,
wider vectors, …

Productivity with Performance via Intel® Distribution for Python*

Intel® Distribution for Python*

Easy, out-of-the-box access to high performance Python

- Prebuilt accelerated solutions for data analytics, numerical computing, etc.
- Drop in replacement for your existing Python. No code changes required.

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

Intel® Distribution for Python*

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

conda create –c intel intelpython3_full

docker pull intelpython/intelpython3_full

pip install intel-numpy intel-scipy intel-scikit-learn

Accelerated NumPy, SciPy
- Intel® MKL
- Intel® C and Fortran compilers
- Linear algebra, universal functions, FFT

Accelerated Scikit-Learn
- Intel® MKL
- Intel® C and Fortran compilers

  } via NumPy/Scipy
- Intel® Data Analytics Acceleration Library (DAAL)

Solutions for efficient parallelism
- TBB4py
- github.com/IntelPython/smp
- Intel® MPI library

Python APIs for Intel® MKL functions
- github.com/IntelPython/mkl_fft
- github.com/IntelPython/mkl_random
- github.com/IntelPython/mkl-service [*]

Python APIs for Intel® DAAL
- github.com/IntelPython/daal4py

Numba with upstreamed Intel contributions
- Parallel Accelerator
- support for SVML
- support for TBB/OpenMP threading runtimes

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

New Features for 2019

Daal4py: Accelerated Analytics tools for Data Scientists

- Package created to address the needs of **Data Scientists and Framework Designers** to harness the **Intel® Data Analytics Acceleration Library (DAAL)** with a **Pythonic API**
- Pandas compatible, **one-liner API** for accessing many **hardware accelerated Machine Learning and Analytics functions**
- Powers our ***Scikit-Learn* accelerations*** in our shipped version of the package
- **Extends capabilities** past Scikit-learn by providing **scaling and distributed modes**

HPAT: A compiler-based framework to speed up Pandas/NumPy

- Used to accelerate the popular ***Pandas*** framework, specifically for the **Dataframe** construct used in analytics and machine learning
- Accelerates **previously unoptimizable portion of end-to-end workflows by accelerating the dataframe and preprocessing steps** of production-level machine learning
- Extends capabilities utilizing pandas instead of migrating to another production solution **with little to no code changes**
- Takes advantage of additional compute nodes via **MPI** for **distributed scaling of compute kernels**

Scaling analytics workloads end-to-end

- Many solutions in the industry have been focused *solely on performance of training or inference*—**but in practice this is only 10% of the actual time**
- The **majority of the time spent** is from the **data ingress and preprocessing** steps
- Identifying the methods to speed up a data analytics tasks from end-to-end **includes both preprocessing and scale out to complete the performance picture**
- Creating both the initial prototype or discovery process with ML and **extending the code to production with the same tools and increased performance** is *the desired workflow* for any **Data Scientist**

Python* Data Analytics that scales

Intel® DAAL and HPAT

Nathan Greeneltch, PhD

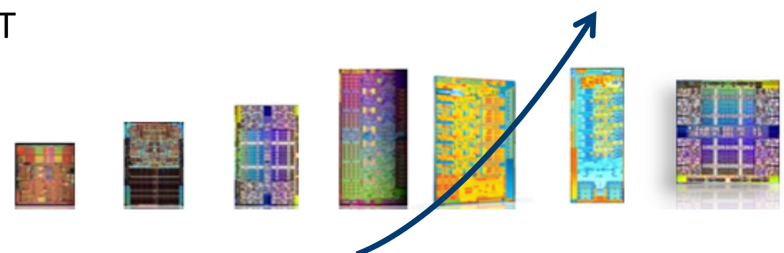
Consulting Engineer, Intel Corporation

Scaling analytics workloads end-to-end

Data Input → Data Preprocessing → Model Creation → Prediction

Pandas
Spark
HPAT

Scikit-learn
Spark
DL-frameworks
daal4py



more nodes, more cores, more threads, wider vectors, …

Scaling analytics workloads end-to-end

HPAT

Drop-in acceleration of Python analytics
(Pandas, Numpy & select custom Python)

- Statically compiles analytics code to binary
- Simply annotate with *@hpat.jit*
- Built on Anaconda Numba compiler

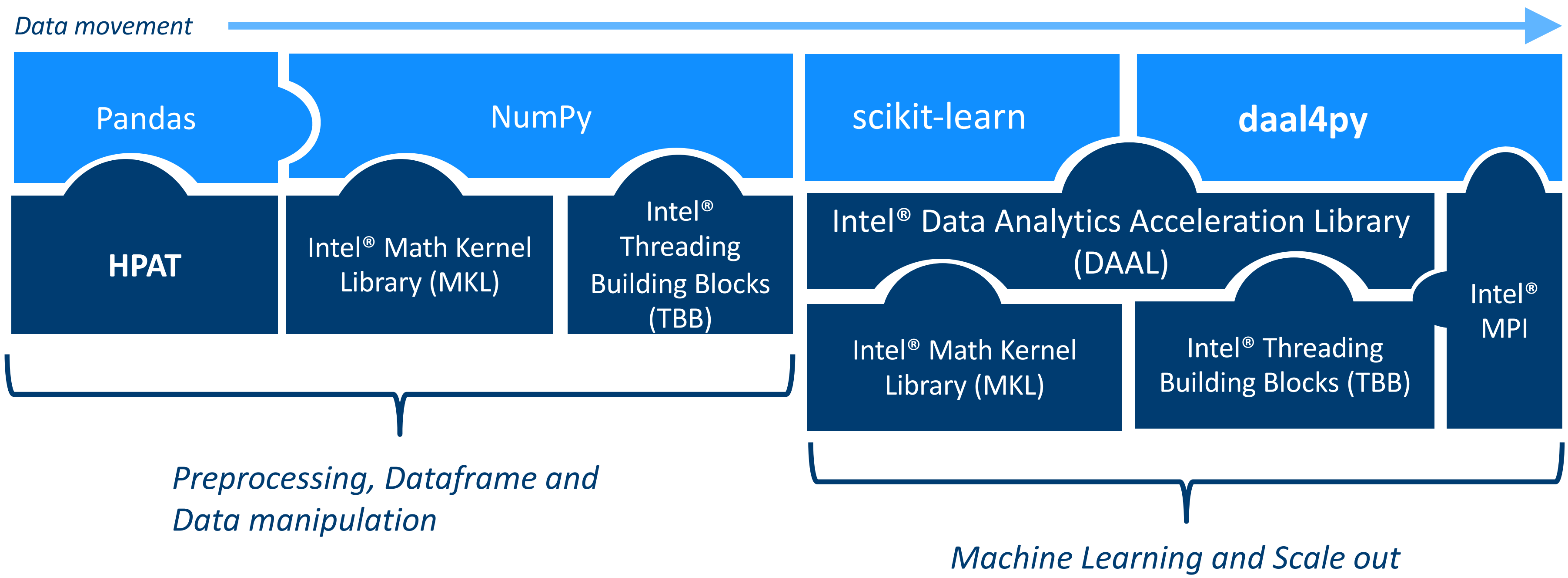
daal4py

Ease-of-use of scikit-learn
+ Performance of DAAL

- High-level Python API for DAAL
- 10x fewer LOC wrt DAAL for single node,
  100x fewer LOC wrt DAAL for multi-node

Automatically scales to multi-node with MPI

End-to-end performance and scale out of analytics



Intel's libraries, tools, and runtimes help **accelerate** the **entire analytics process** from *preprocessing through machine learning and scale out*

Accelerating Pandas using HPAT

```
import pandas as pd
import hpat

@hpat.jit
def process_times():
    df = pq.read_table('data.parquet').to_pandas();
    df['event_time'] = pd.DatetimeIndex(df['event_time'])
    df['hr'] = df.event_time.map(lambda x: x.hour)
    df['minute'] = df.event_time.map(lambda x: x.minute)
    df['second'] = df.event_time.map(lambda x: x.second)
    df['minute_day'] = df.apply(lambda row: row.hr*60 + row.minute, axis = 1)
    df['event_date'] = df.event_time.map(lambda x: x.date())
    df['indicator_cleaned'] = df.indicator.map(lambda x: -1 if x == 'na' else int(x))
```

```
$ aprun -n # -N # python ./process_times.py
```

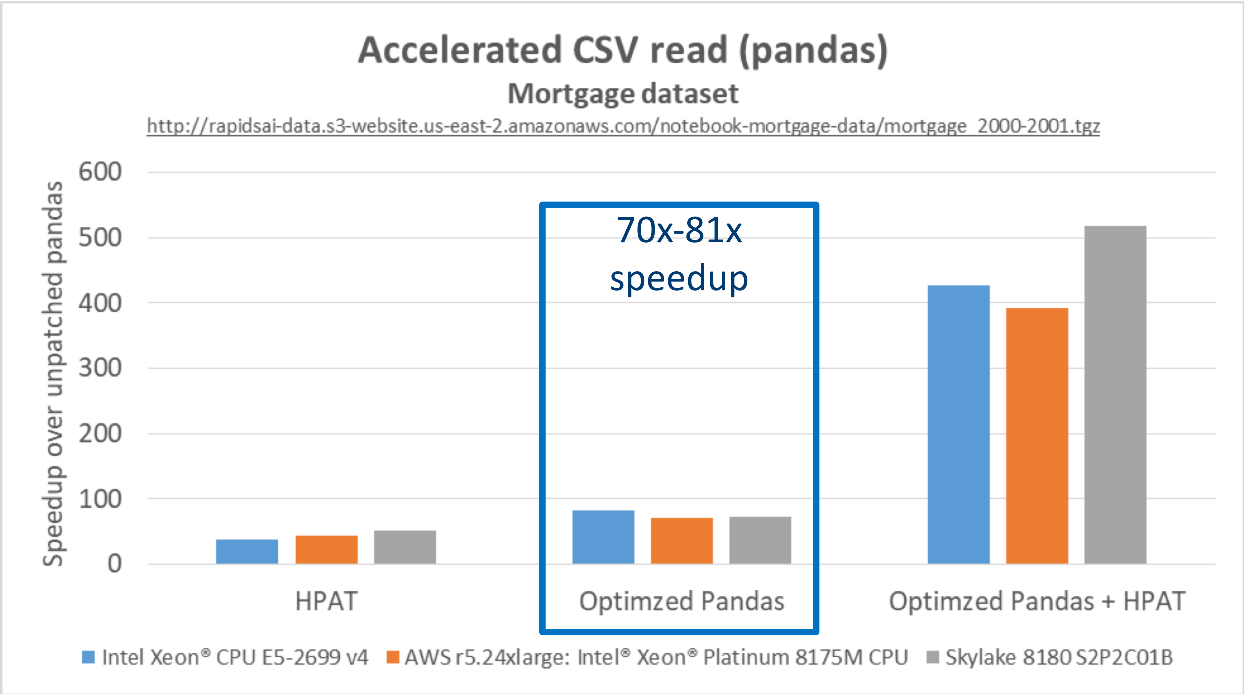
HPAT's Scope of Functionalities (Early Preview)

Operations	• Python/Numpy basics • Statistical operations (mean, std, var, …) • Relational operations (filter, join, groupby) • Custom Python functions (apply, map)
Data	• Missing values • Time series, dates • Strings, unicode • Dictionaries • Pandas Extend Numba to support
Interoperability	• I/O integration (CSV, Parquet, HDF5, Xenon) • Daal4py integration

Accelerating pandas CSV read

Patches merged to pandas mainline:

https://github.com/pandas-dev/pandas/pull/25804
https://github.com/pandas-dev/pandas/pull/25784

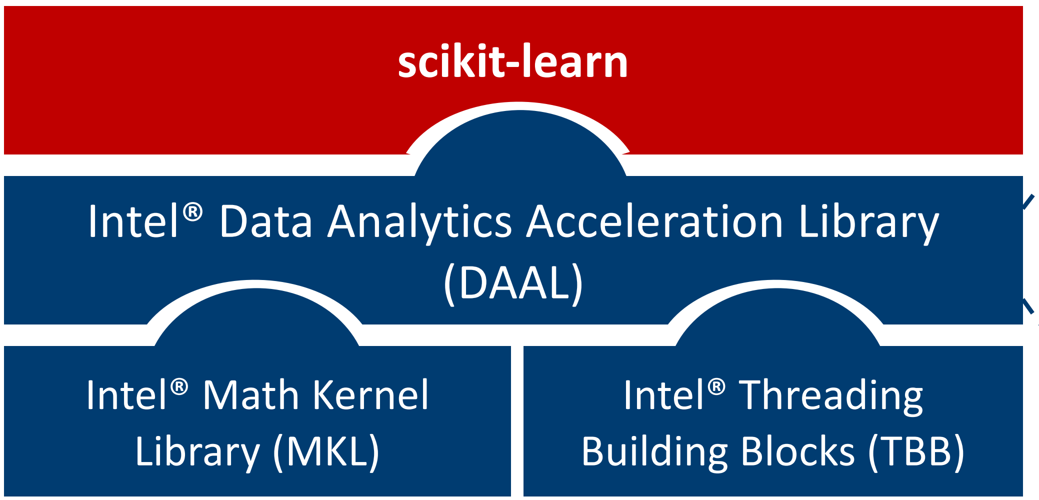


Intel(R) Xeon(R) CPU E5-2699 v4: 2.20GHz; 1chreads per core; 22 cores per socket; 2 sockets
Intel(R) Xeon(R) Platinum 8175M CPU: 2.50GHz; 2 threads per core; 24 cores per socket; 2 sockets
Skylake 8180 S2P2C01B: 2.5GHz
1 thread per core; 28 cores per socket; 2 sockets

HPAT Details

- **Open Source**: https://github.com/IntelLabs/hpat
- **BSD Licensed**
- Built on top of **Numba**, leverages many of Intel's vectorization optimizations
- Little to no code changes required (only @hpat.jit decorator)
- Optimizes the **pandas** framework and **numpy** code together to accelerate preprocessing code and tasks
- Major release later this year

Accelerating Machine Learning



- Efficient memory layout via `Numeric Tables`
- Blocking for optimal cache performance
- Computation mapped to most efficient matrix operations (in MKL)
- Parallelization via TBB
- Vectorization

Try it out! `conda install -c intel scikit-learn`

intel Software

Accelerating scikit-learn through daal4py

```
> python -m daal4py <your-scikit-learn-script>
```

Monkey-patch any scikit-learn on the command-line

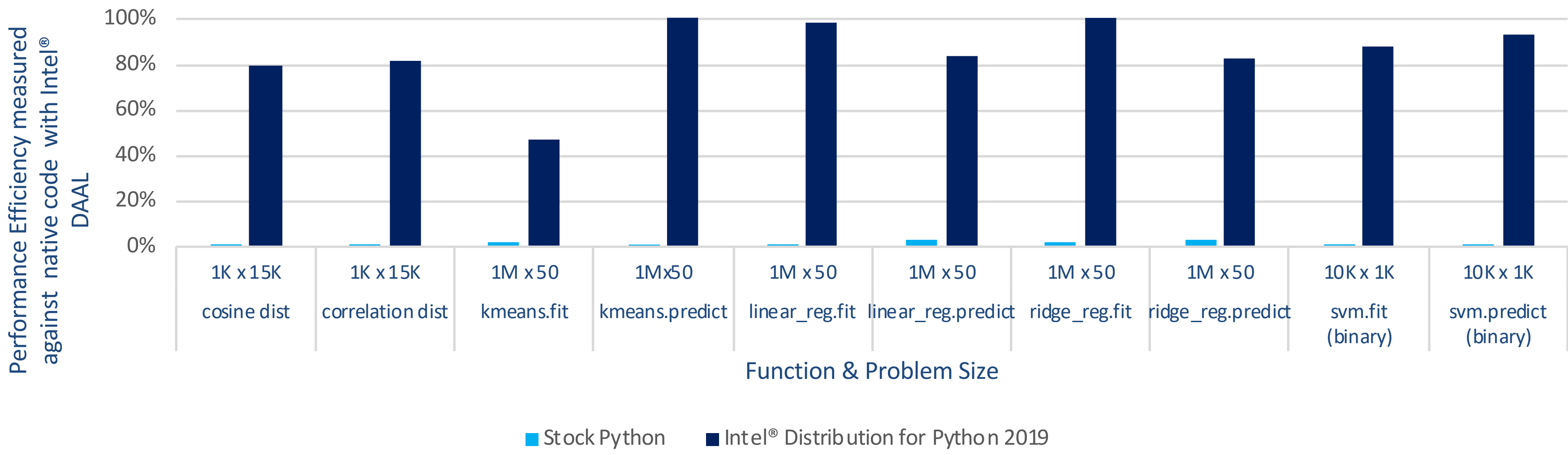
```
import daal4py.sklearn
daal4py.sklearn.patch_sklearn()
```

Monkey-patch any scikit-learn programmatically

Scikit-learn with daal4py patches applied passes scikit-learn test-suite

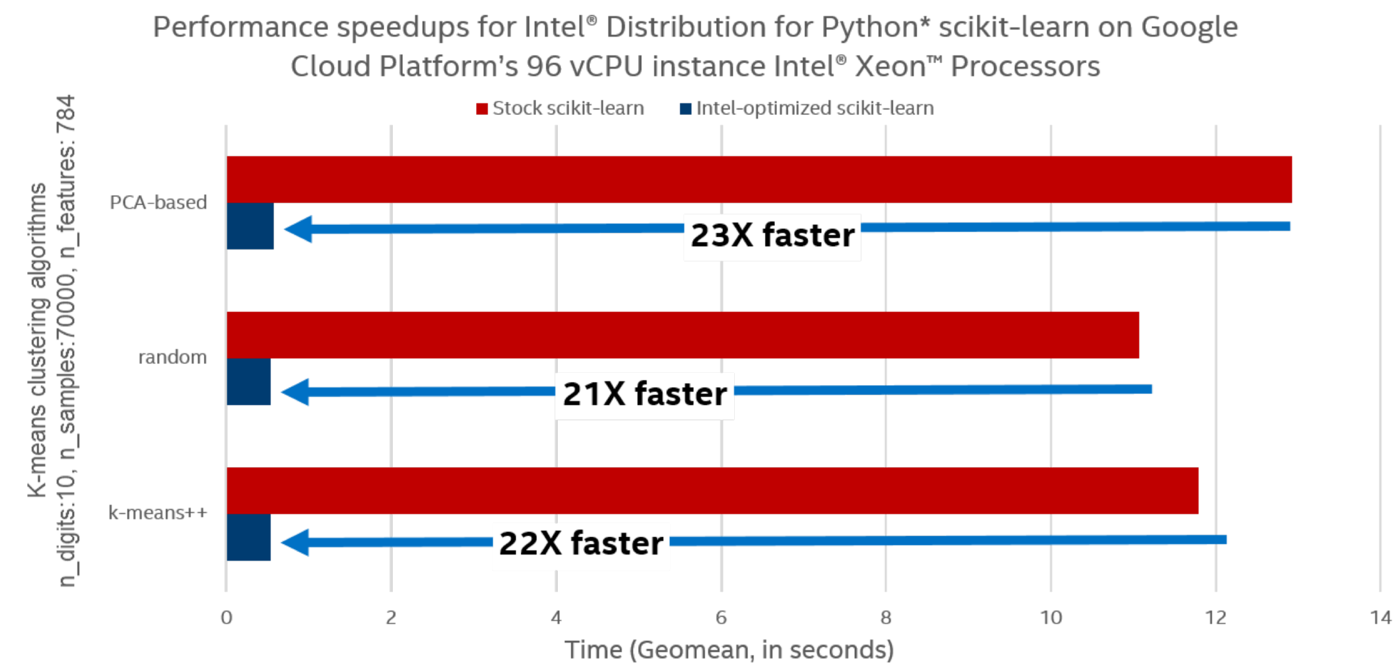
Close to native code Scikit-learn* Performance with Intel® Distribution of Python

Compared to Stock Python packages on Intel® Xeon® processors



Configuration: Stock Python: 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 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.

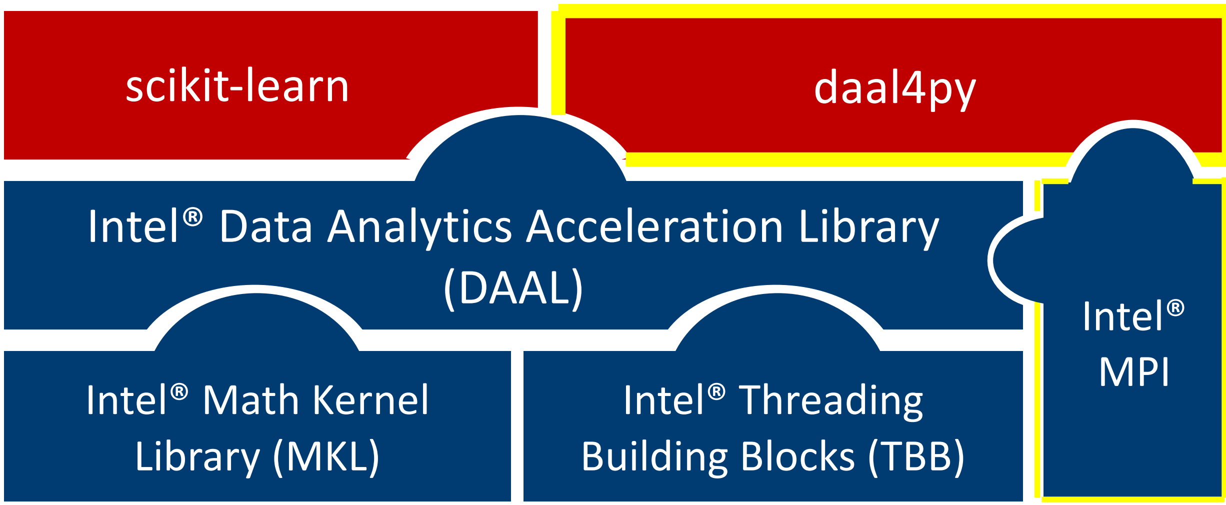
Accelerating K-Means



System Configuration: GCP VM, zone us-central1-c; 96 vCPU, Intel Skylake; 360 GB memory. Ubuntu 16.04.3 LTS; Linux instance-1 4.10.0-38-generic #42~16.04.1-Ubuntu SMP Tue Oct 10 16:32:20 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux; Intel® Distribution for Python* from Docker image intelpython/intelpython3_full:latest (created 2017-09-12T20:10:42.862965559Z); Stock Python*: pip install scikit-learn

https://cloudplatform.googleblog.com/2017/11/Intel-performance-libraries-and-python-distribution-enhance-performance-and-scaling-of-Intel-Xeon-Scalable-processors-on-GCP.html

Scaling Machine Learning Beyond a Single Node



Simple Python API
Powers scikit-learn

Powered by DAAL

Scalable to multiple nodes

Try it out! `conda install -c intel daal4py`

K-Means using daal4py

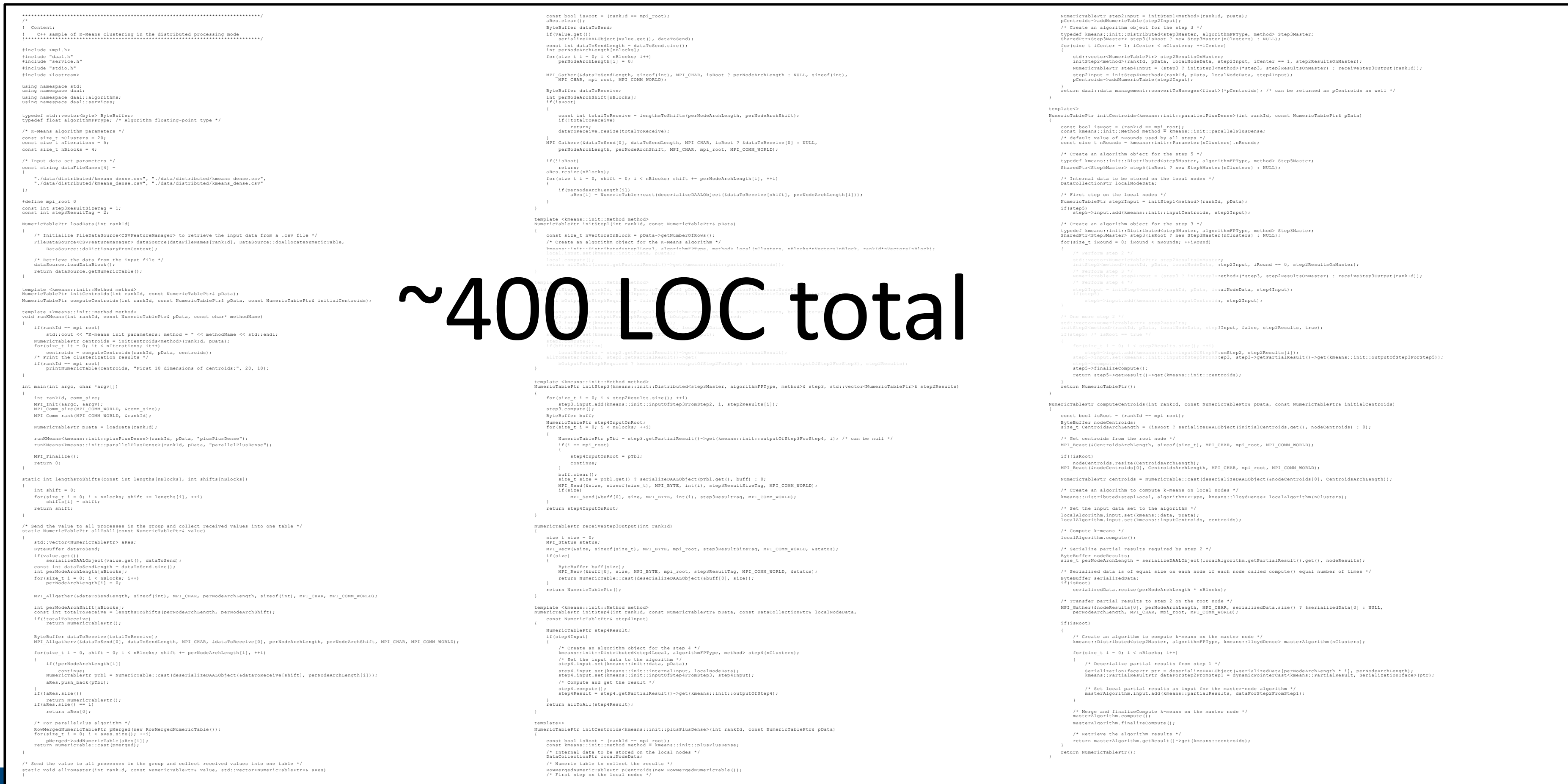
```
import daal4py as d4p

# daal4py accepts data as CSV files, numpy arrays or pandas dataframes
# here we let daal4py load process-local data from csv files
data = "kmeans_dense.csv"

# Create algo object to compute initial centers
init = d4p.kmeans_init(10, method="plusPlusDense")
# compute initial centers
ires = init.compute(data)
# results can have multiple attributes, we need centroids
Centroids = iris.centroids
# compute initial centroids & kmeans clustering
result = d4p.kmeans(10).compute(data, centroids)
```

Distributed K-Means using daal4py

```
import daal4py as d4p

# initialize distributed execution environment
d4p.daalinit()

# daal4py accepts data as CSV files, numpy arrays or pandas dataframes
# here we let daal4py load process-local data from csv files
data = "kmeans_dense_{}.csv".format(d4p.my_procid())

# compute initial centroids & kmeans clustering
init = d4p.kmeans_init(10, method="plusPlusDense", distributed=True)
centroids = init.compute(data).centroids
result = d4p.kmeans(10, distributed=True).compute(data, centroids)
```

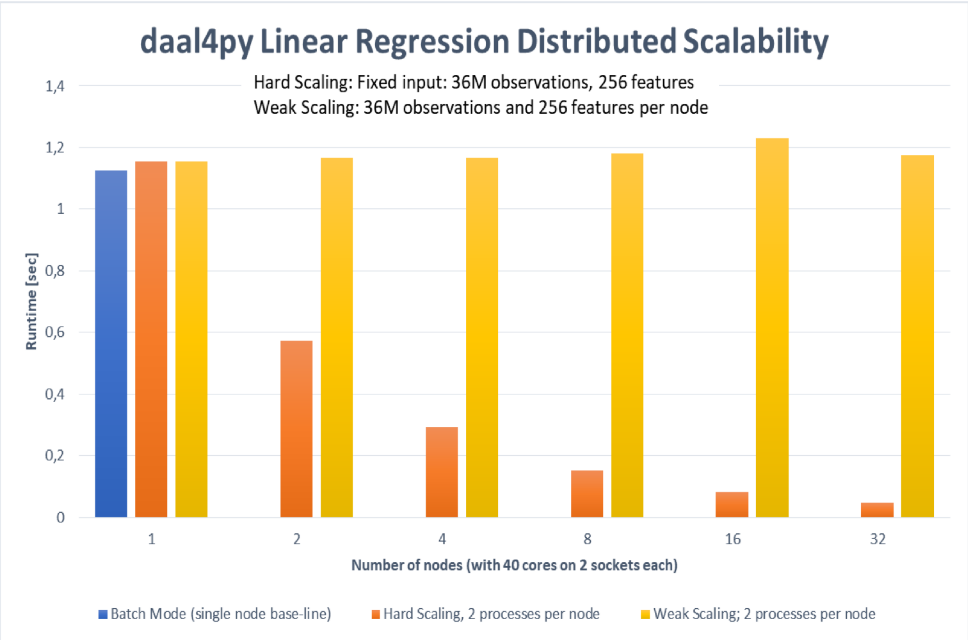
```
aprun -n # -N # python ./kmeans.py
```

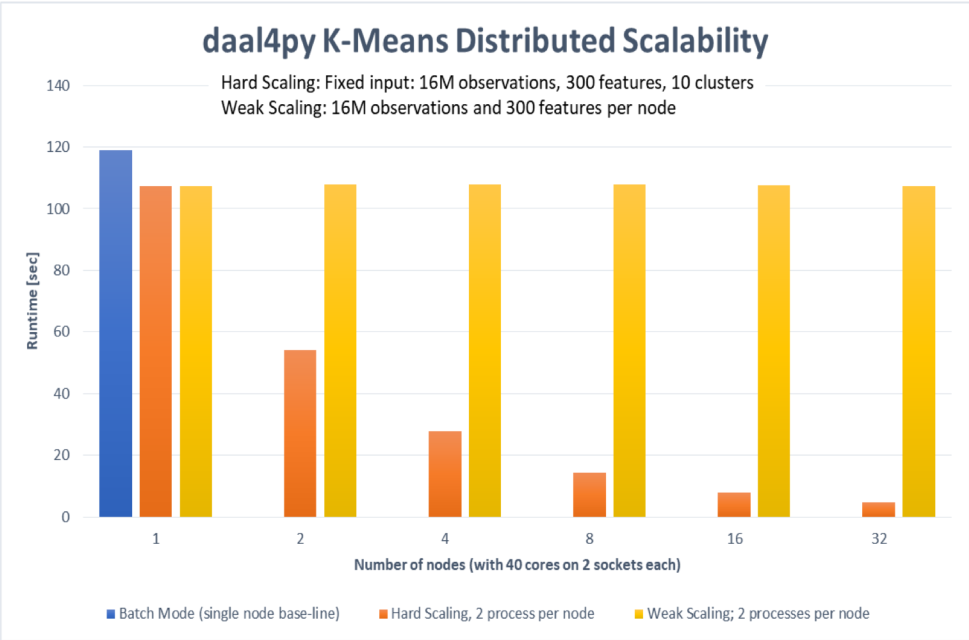
Distributed K-Means Using DAAL (C++ API)



Strong & Weak Scaling via daal4py

Hardware	Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz, EIST/Turbo on
	2 sockets, 20 Cores per socket
	192 GB RAM
	16 nodes connected with Infiniband
Operating System	Oracle Linux Server release 7.4
Data Type	double





On a 32-node cluster (1280 cores) daal4py computed linear regression of 2.15 TB of data in 1.18 seconds and 68.66 GB of data in less than 48 milliseconds.

On a 32-node cluster (1280 cores) daal4py computed K-Means (10 clusters) of 1.12 TB of data in 107.4 seconds and 35.76 GB of data in 4.8 seconds.

Streaming data (linear regression) using daal4py

```
import daal4py as d4p

# Configure a Linear regression training object for streaming
train_algo = d4p.linear_regression_training(interceptFlag=True, streaming=True)

# assume we have a generator returning blocks of (X,y)...
rn = read_next(infile)

# on which we iterate
for chunk in rn:
    algo.compute(chunk.X. chunk.y)

# finalize computation
result = algo.finalize()
```

Intel® Data Analytics Acceleration (Intel® DAAL) Algorithms supported by daal4py Data Transformation and Analysis

Basic statistics for datasets
- Low order moments
- Quantiles
- Order statistics

Correlation and dependence
- Cosine distance
- Correlation distance
- Variance-Covariance matrix

Matrix factorizations
- SVD
- QR
- Cholesky

Dimensionality reduction
- PCA

Outlier detection
- Univariate
- Multivariate

Association rule mining (Apriori)

Optimization solvers (SGD, AdaGrad, lBFGS)

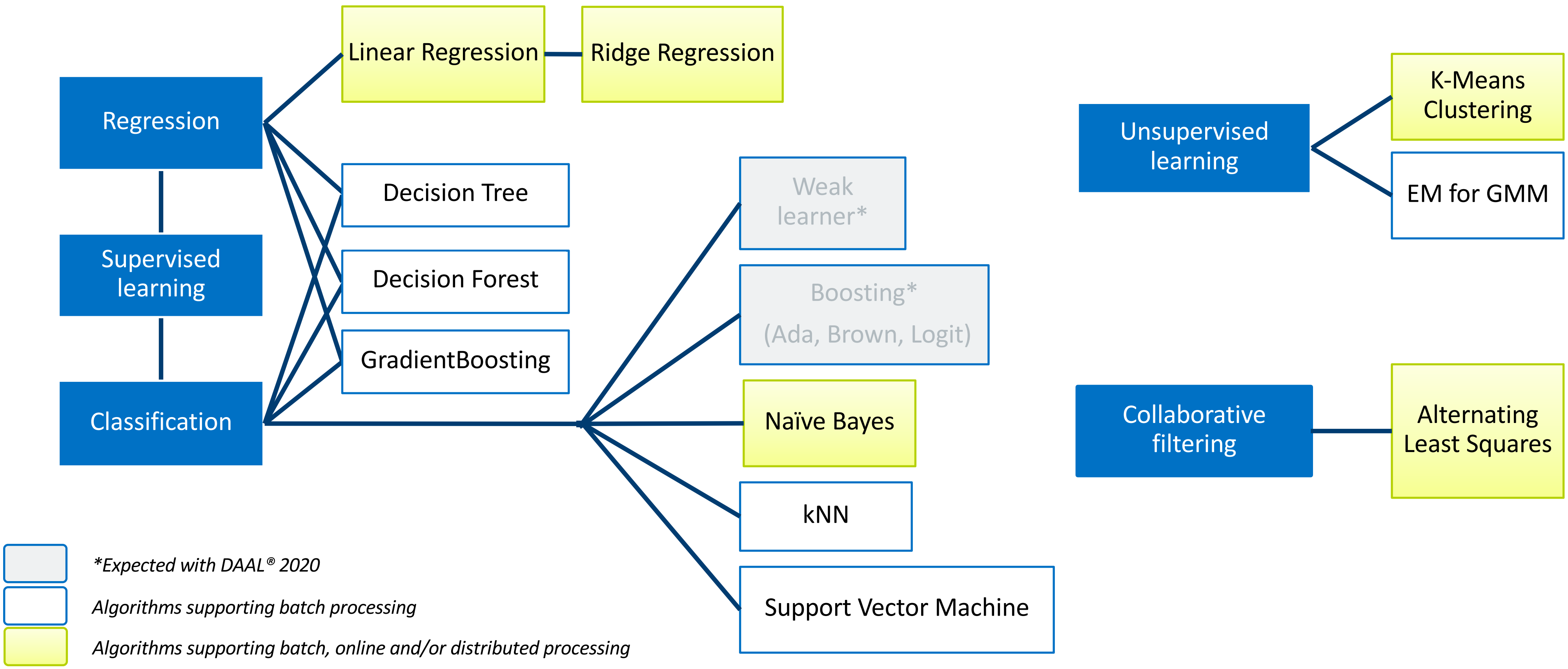
Math functions (exp, log,...)

Algorithms supporting batch processing

Algorithms supporting batch, online and/or distributed processing

Intel® DAAL Algorithms supported by daal4py

Machine Learning



Intel® DAAL for Python*: Summary

Fast & Scalable	•Close to native performance through Intel® DAAL •Efficient MPI scale-out •Streaming
Easy to use	•Known usage model •Picklable
Flexible	•Object model separating concerns •Plugs into scikit-learn •Plugs into HPAT
Open	•Open source: https://github.com/IntelPython/daal4py

https://intelpython.github.io/daal4py/

Intel® HPAT: How to get (check github)

Conda

- conda create -n HPAT -c ehsantn -c anaconda -c conda-forge hpat

https://intellabs.github.io/hpat-doc/dev/index.html

CALL TO ACTION

LEARN

More information at
https://software.intel.com/en-us/distribution-for-python

EXPLORE

Use Intel's accelerated Python* libraries

ENGAGE

Use Our Forums for Free Support
forums.intel.com

Intel® Distribution for Python*

https://anaconda.org/intel
https://software.intel.com/en-us/distribution-for-python
https://intelpython.github.io/daal4py
https://github.com/IntelLabs/hpat

Questions?