H2S HACK2SKILL

intel

oneAPI <HACK>ATHON

BUILD SOLUTIONS TO UNLOCK THE POTENTIAL OF HETEROGENEOUS COMPUTING

REGISTER NOW

Problem Statements

- Object Detection For Autonomous Vehicles
- Medical Image Processing
- Open Innovation in Education

Hackathon Phases

PHASE - 1

Idea Submission - In a PPT format, converted into PDF

This will be the ideation phase of the hackathon where participants shall brainstorm their ideas for the given problem statements and submit innovative solutions to their chosen problem statements using Intel® AI Analytics Toolkits, its Libraries and SYCL/DPC++ Libraries

PHASE - 2

Prototype Development - Creation of Prototype & Code submissions in GitHub repository

In this phase, the participants will be working towards building their prototype and will be attending mentor hours taken by industry experts and professionals from Intel®. By the end of this phase, they will have to submit their prototype codes in the Final Submission.

PHASE - 3

30 Hour Offline Hackathon

The shortlisted participants after making their final submissions, will be invited to fine tune their prototype with the help of the hackathon mentors and present it in front of an esteemed Jury Panel on the hackathon days of 1st & 2nd of July

oneAPI

Deliverables

Deliverables - Idea Submission Phase

- Participants must strictly engage in the use of Intel® AI Analytics Toolkits, its Libraries & SYCL/DPC++ Libraries while framing their ideations.
- The ideations of the participants must be compiled in the form of a Powerpoint Presentation (PPT) converted into a PDF format for submission
- A sample template for idea submission will be given to the participants in their hackathon dashboard, all the points mentioned in the template should strictly be covered in their submission.
- Link of the GitHub repository will be given to the participants in their hackathon dashboard
- Participants will be required to fork this repository and update the README file, filling in the required details.
- To avoid disqualification, participants must submit their ideas in the hackathon dashboard before the deadline i.e. 7th May 2023

Deliverables

Deliverables - Prototype Development Phase

- The prototype submission should be made on a GitHub public repository.
- The entire code base needs to be present on the team's GitHub repository.
- The prototype submission must lay down equal emphasis on the deployment/inference benchmarking for both, with and without Intel® oneAPI
- Pull requests of the forked repository need to be generated

Note : Participants can update the README file in their forked repository, in the prototype development phase as well in order to reflect their changes. The GitHub repositories must be retained throughout the process, even after the winner announcement.

Judging Criteria

Code Quality - 33.33%

- Code is easy to understand and is reproducible
- Code is well tested, and functions without errors
- Code is well documented

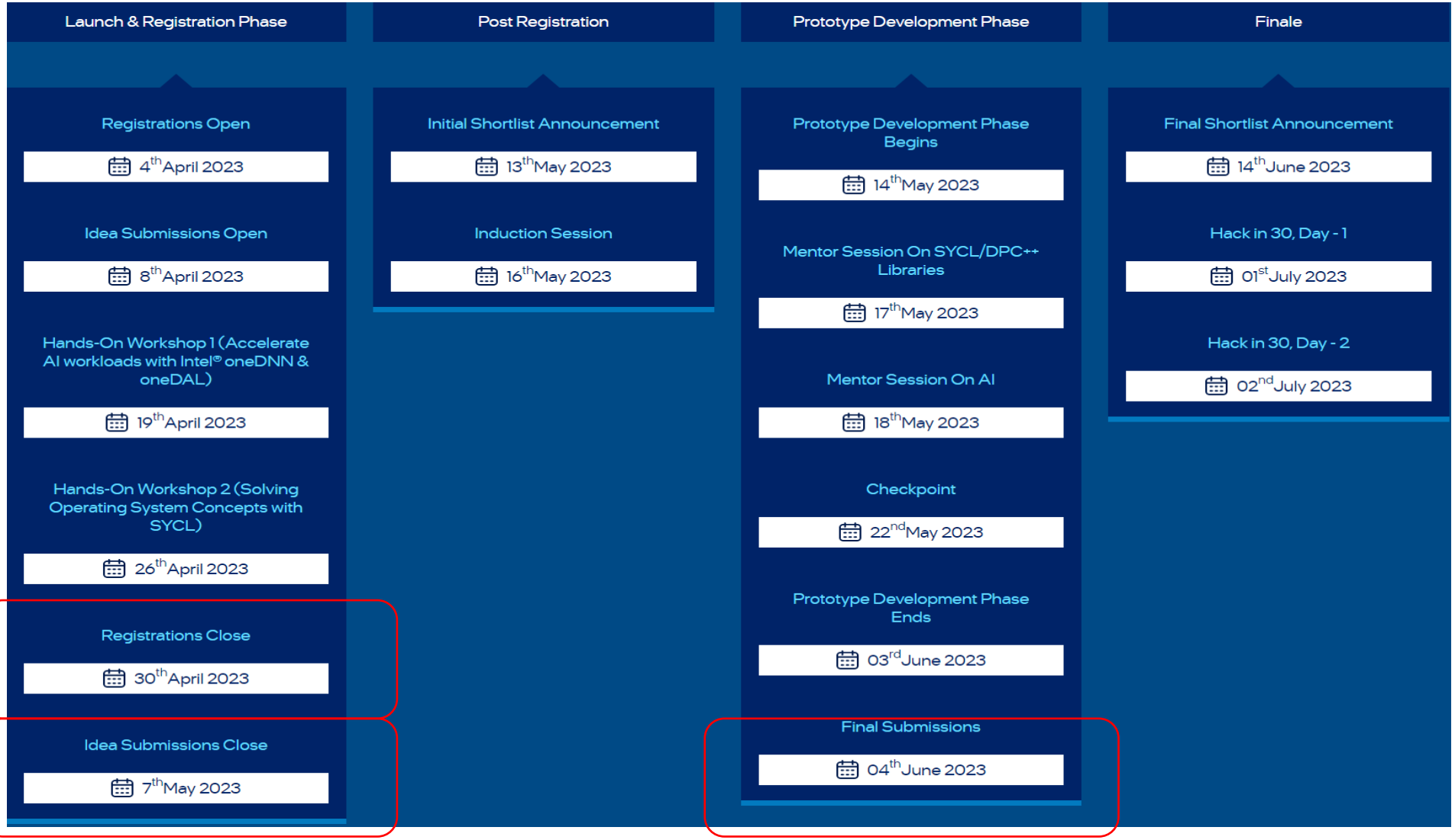
Code Quality - 33.33%

- Intel® AI Analytics Toolkits, its Libraries and SYCL/DPC++ Libraries are leveraged and used appropriately by developer
- Data Preprocessing and Exploratory Data Analysis (EDA)
- Model Training and Inference
- Implementation of Intel® oneAPI is a must

Creativity & Originality - 33.33%

- Solution is original and clearly differentiates itself from submissions in the same category
- Solution creates clear additional value beyond competition
- Scale and novelty of technology

Timelines



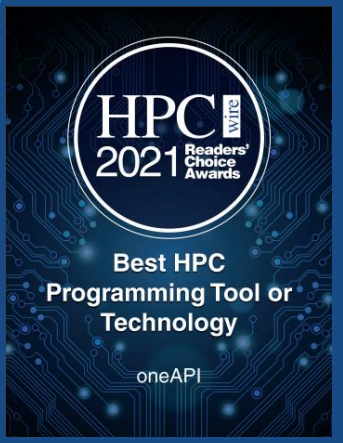
oneAPI

Poll Question 1

oneAPI

Accelerate AI workloads with Intel® oneDNN & oneDAL

Aditya Sirvaiya
AI Software Solutions Engineer
AI & Analytics, Intel Corporation



intel

Agenda

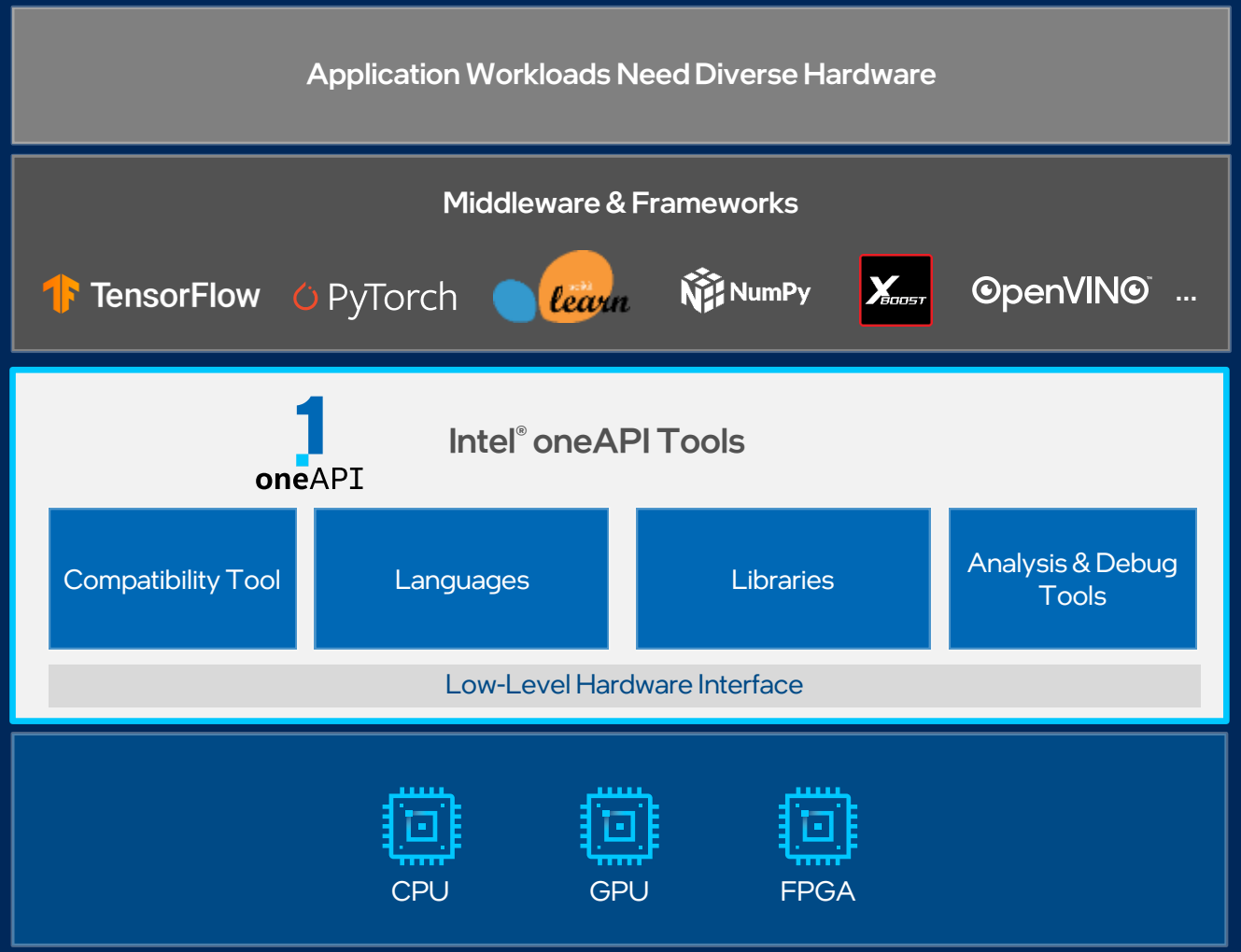
- Overview of Intel® AI Analytics Toolkit[oneAPI Toolkit]
- Intel® Extension for Scikit-learn* [oneDAL]
- Intel® Distribution of Modin* [oneDAL]
- Intel® Optimization for PyTorch*(PyTorch+IPEX) [oneDNN]
- Intel® Optimization for TensorFlow* [oneDNN]
- Intel® Neural Compressor[Inference Optimization Tool]

Intel® oneAPI Tools

Built on Intel's Rich Foundation of CPU Tools Expanded to Accelerators

A complete set of advanced compilers, libraries, and porting, analysis and debugger tools

- Accelerates compute by exploiting cutting-edge hardware features
- Interoperable with existing programming models and code bases (C++, Fortran, Python, OpenMP, etc.), developers can be confident that existing applications work seamlessly with oneAPI
- Eases transitions to new systems and accelerators using a single code base frees developers to invest more time on innovation



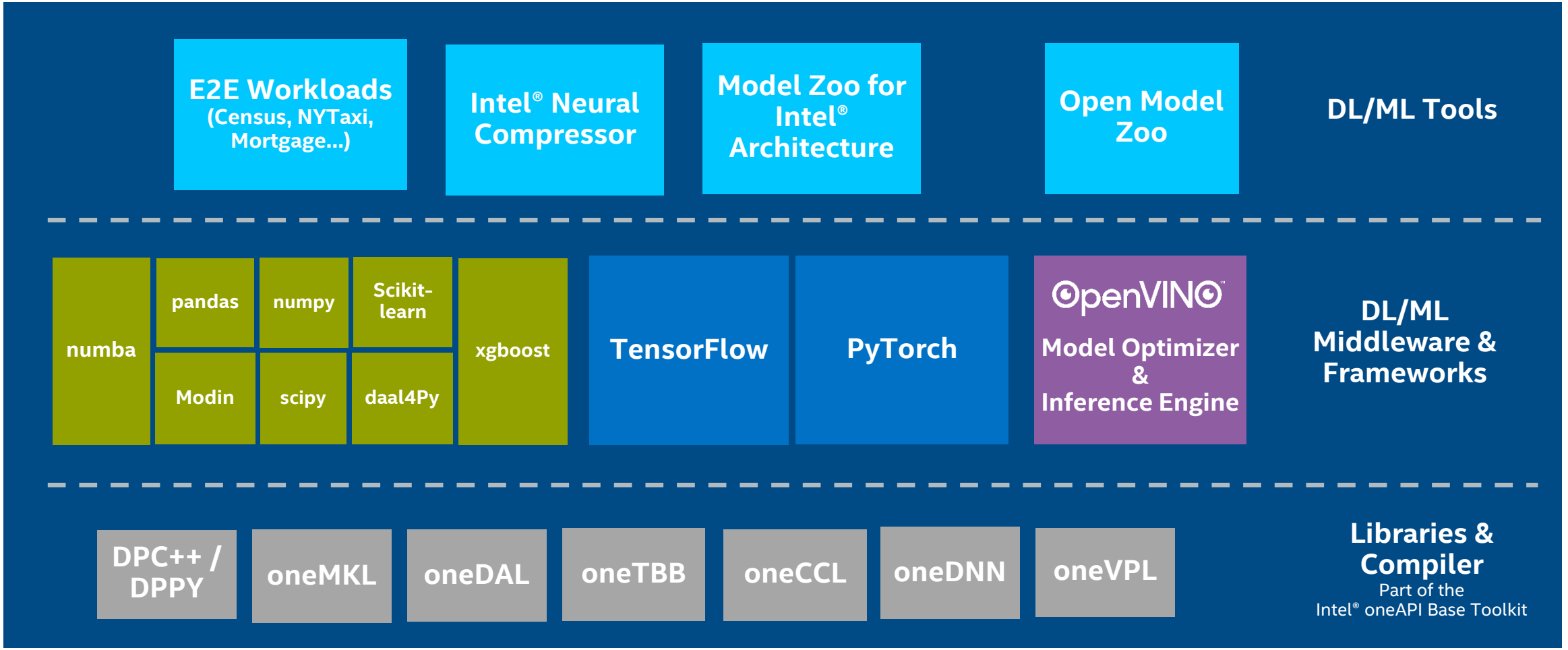
Available Now

Latest version is 2022.3.1

Visit software.intel.com/oneapi for more details
Some capabilities may differ per architecture and custom-tuning will still be required. Other accelerators to be supported in the future.

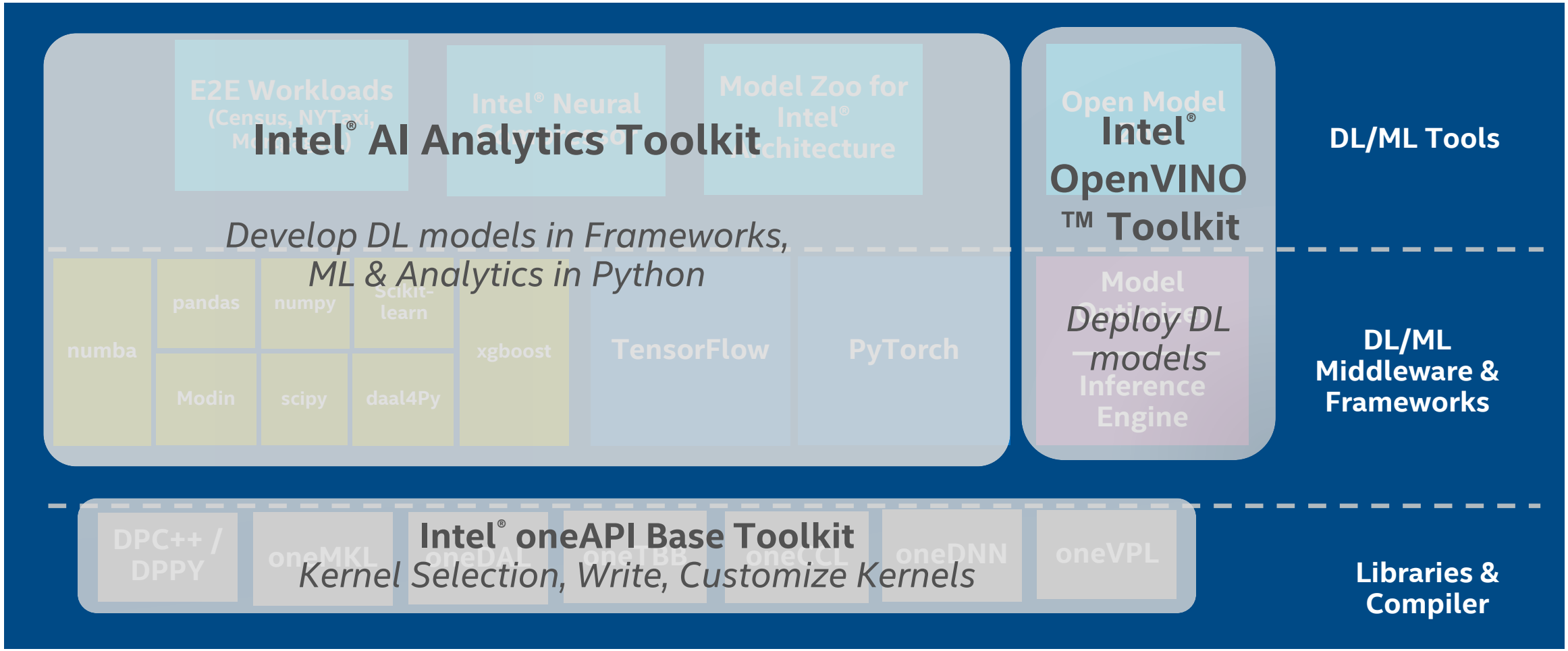
AI Software Stack for Intel® Architecture

Intel offers a Robust Software Stack to Maximize Performance of Diverse Workloads



AI Software Stack for Intel® Architecture

Intel offers a Robust Software Stack to Maximize Performance of Diverse Workloads



Full Set of Intel oneAPI cross-architecture AI ML & DL Software Solutions

For details on Intel® Distribution of OpenVINO™ Toolkit, see 30-3-30 (internal) & product site

Intel® AI Analytics Toolkit

Intel® AI Analytics Toolkit

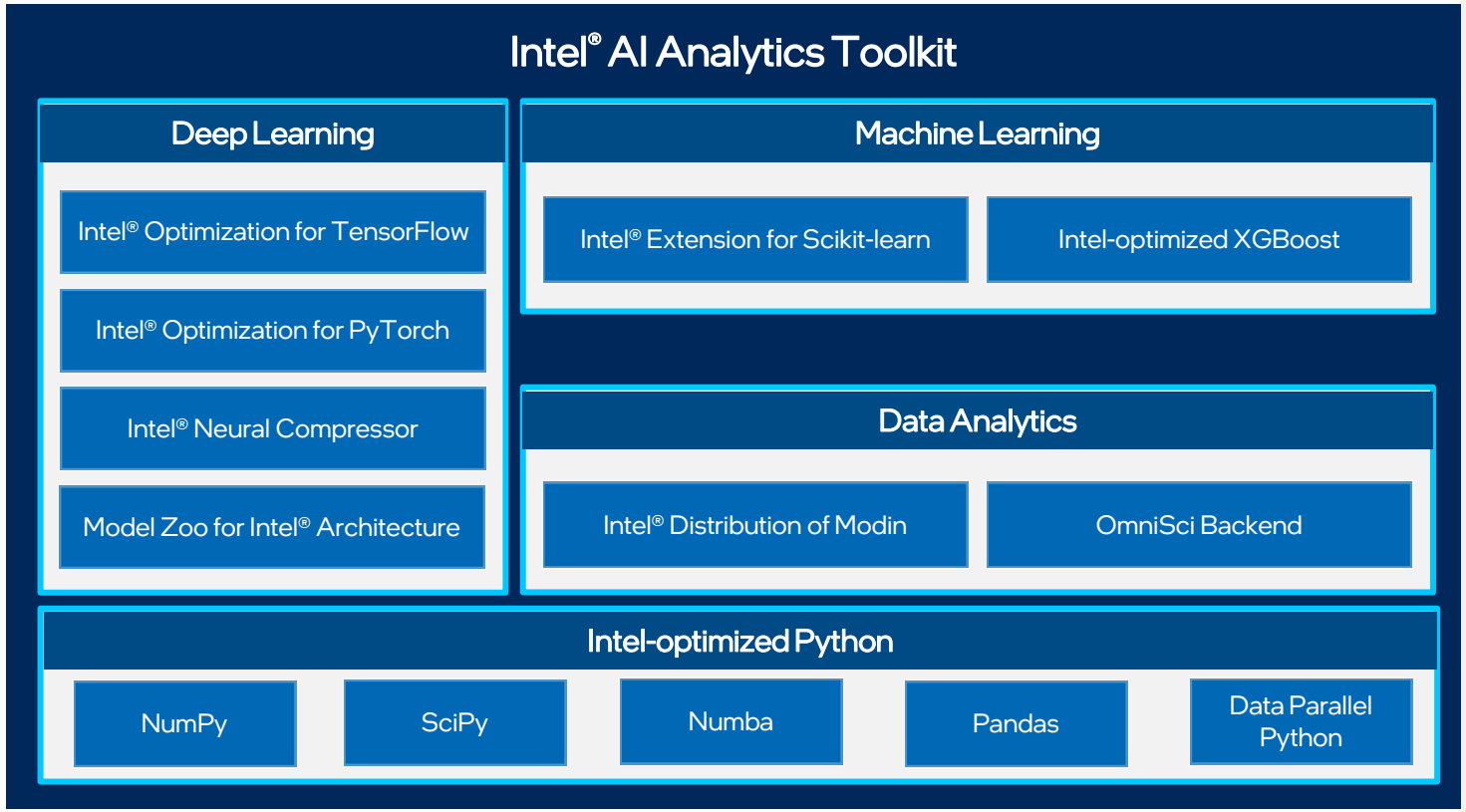
Accelerate end-to-end AI and data analytics pipelines with libraries optimized for Intel® architectures

Who Uses It?

Data scientists, AI researchers, ML and DL developers, AI application developers

Top Features/Benefits

- Deep learning performance for training and inference with Intel optimized DL frameworks and tools
- Drop-in acceleration for data analytics and machine learning workflows with compute-intensive Python packages



CPU GPU

Hardware support varies by individual tool. Architecture support will be expanded over time.

Get the Toolkit HERE or via these locations

Intel Installer | Docker | Apt, Yum | Conda | Intel® DevCloud

Learn More: software.intel.com/oneapi/ai-kit

Back to Domain-specific Toolkits for Specialized Workloads

How Intel delivers AI optimizations

Upstream

Integrated acceleration to popular open source software

Modin, XGBoost, TF, PT, PDPD, MxNet, more …

Intel Extension

Easily pluggable extensions to open source software

Scikit-Learn Extension, Optimized Analytics Package, Intel Extension for Pytorch, more …

Intel Distro

Intel Optimized Distributions of open source software

Modin, Intel TF, Intel Distribution of Python

Intel Tools

Tools / Kits which improve productivity and perf on Intel HW

AIKIT, OpenVINO™, BigDL, oneContainer Portal, Cnvrg.io, Intel Neural Compressor, SigOpt,

Across major software channels (PyPI, Anaconda, Intel, Apt, Yum, Docker)

Intel AI Software Value Approach Map

Categories	SW Product	Optimizations Upstreamed	Intel Extension
DL Frameworks	TensorFlow	Differentiation upstreamed to Open Source	+ Differentiation in Intel package
	PyTorch		
ML Frameworks	Scikit-Learn		
Data Prep	Modin		

See link below for workloads and configurations. Results may vary

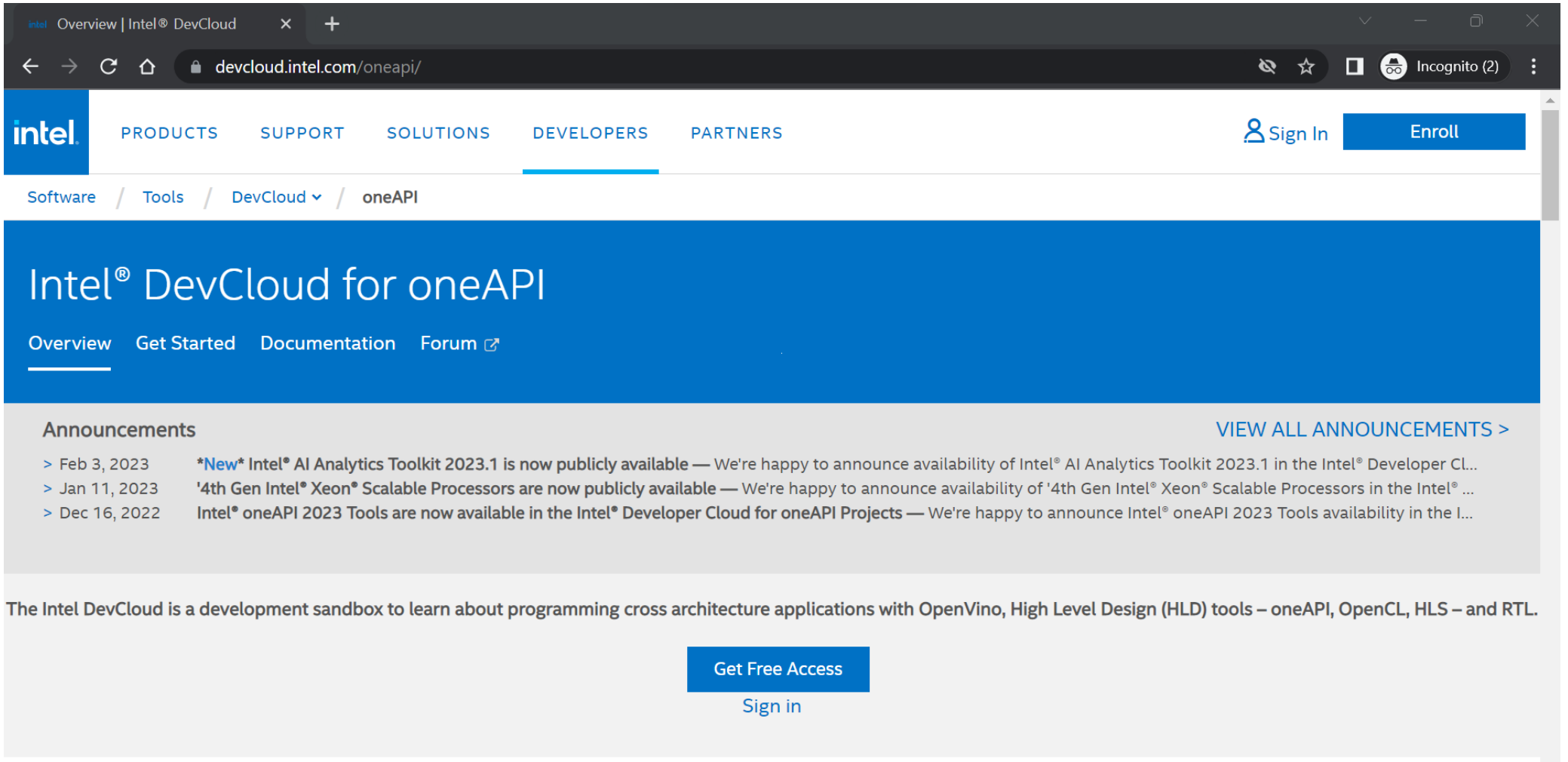
https://techdecoded.intel.io/resources/one-line-code-changes-to-boost-pandas-scikit-learn-and-tensorflow-performance/#gs.bzkn2n

DevCloud Registration Process

https://devcloud.intel.com/oneapi/

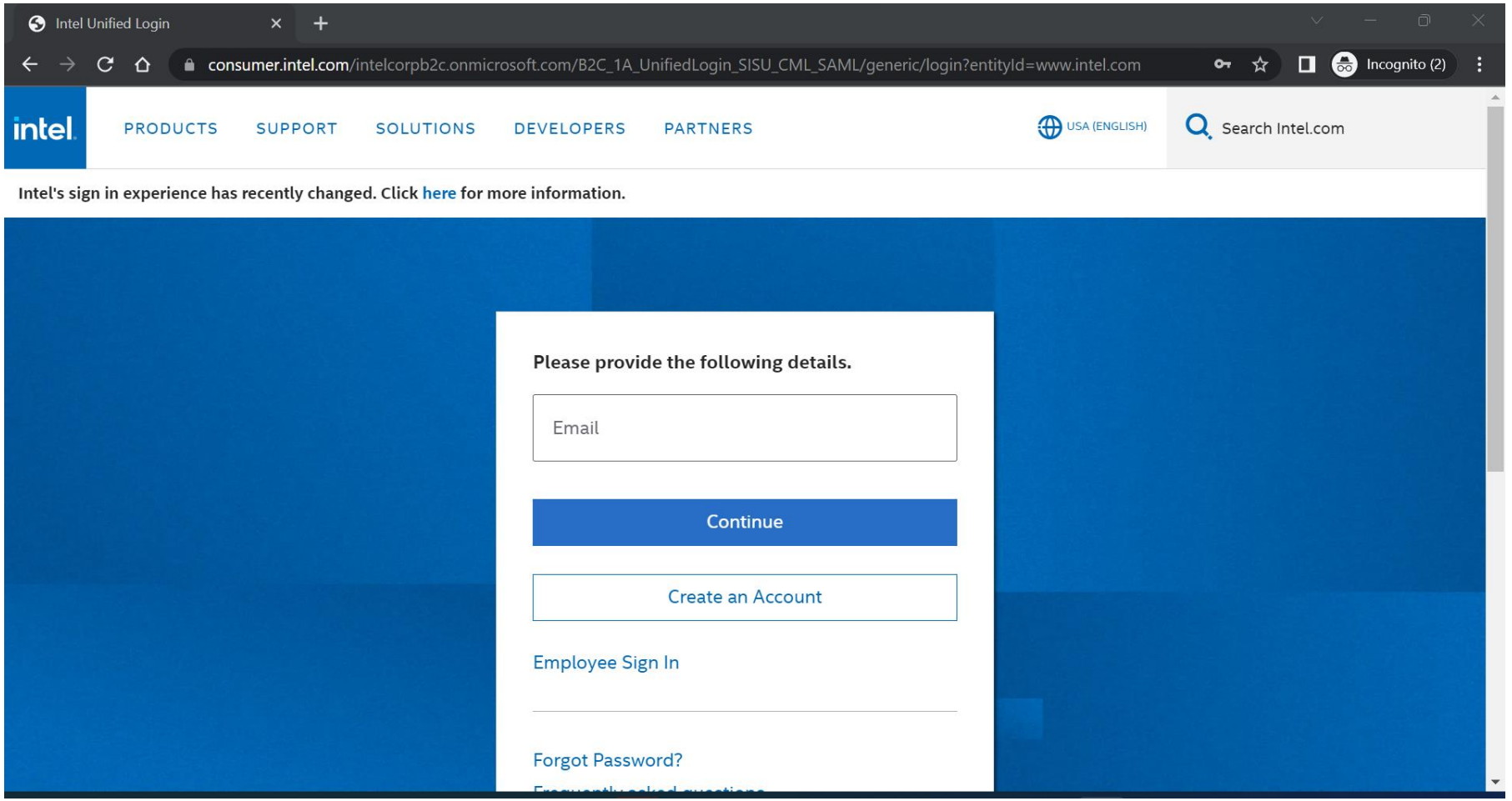
Poll Question 2

Step 1:

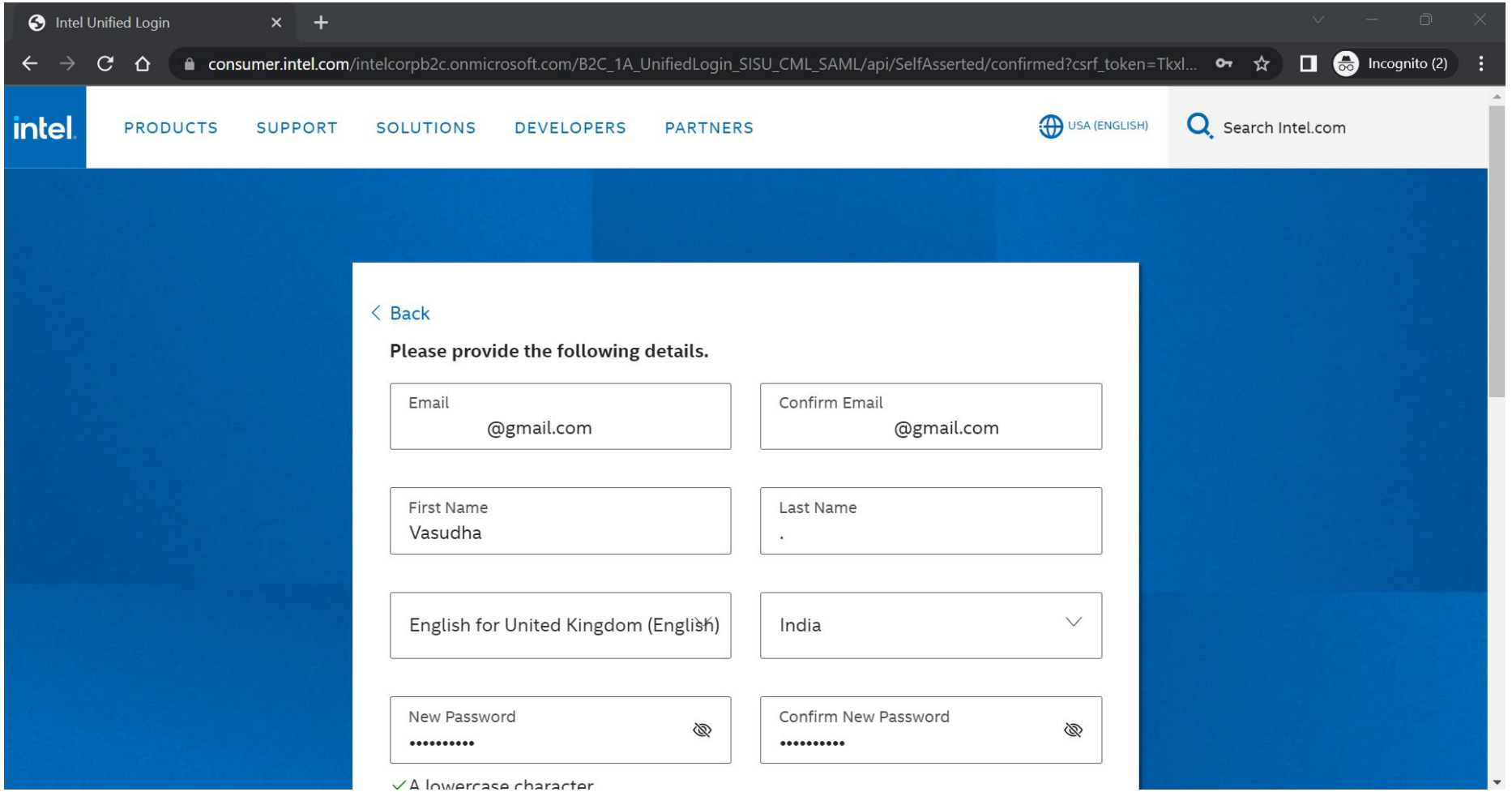


https://devcloud.intel.com/oneapi/

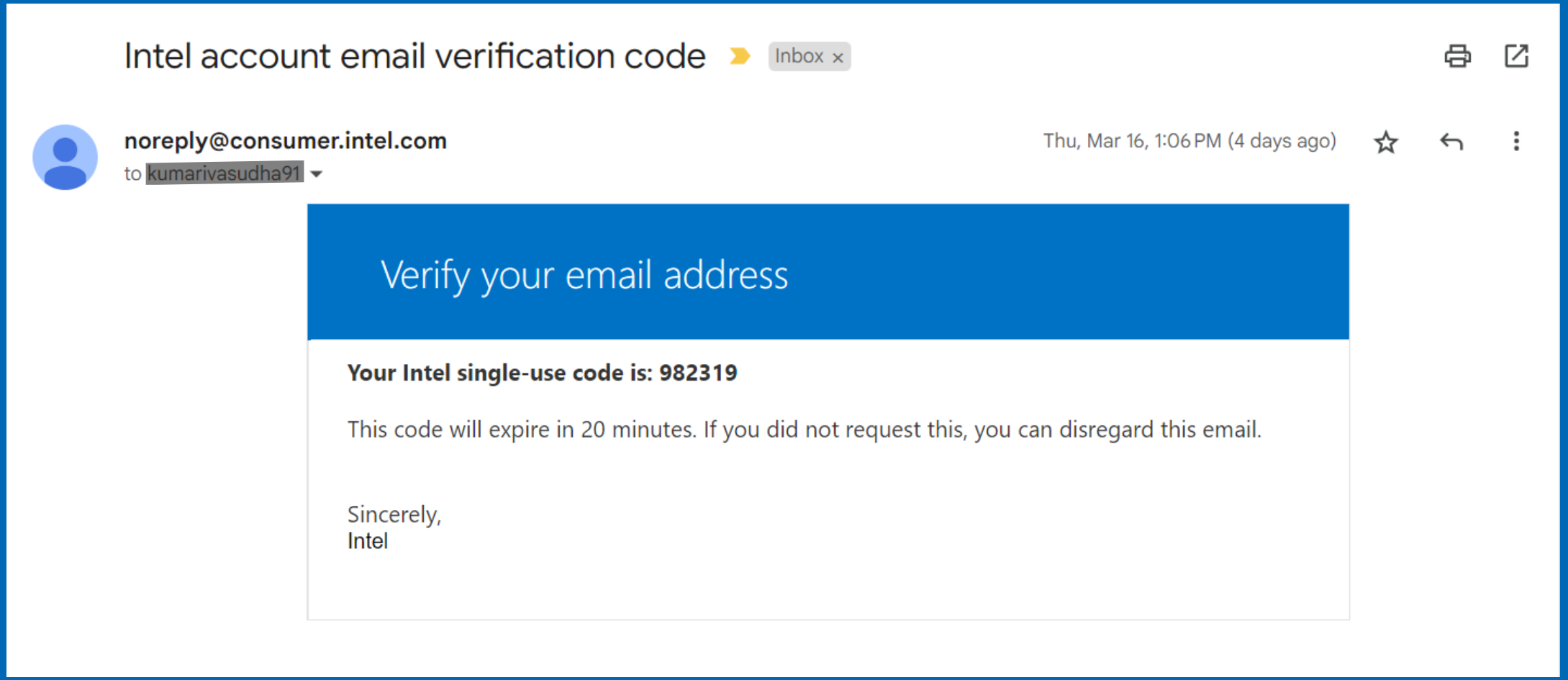
Step:2


Intel Unified Login
consumer.intel.com/intelcorpb2c.onmicrosoft.com/B2C_1A_UnifiedLogin_SISU_CML_SAML/generic/login?entityId=www.intel.com
Incognito (2)
intel
PRODUCTS
SUPPORT
SOLUTIONS
DEVELOPERS
PARTNERS
USA (ENGLISH)
Search Intel.com
Intel's sign in experience has recently changed. Click here for more information.
Please provide the following details.
Email
Continue
Create an Account
Employee Sign In
Forgot Password?

Step:3

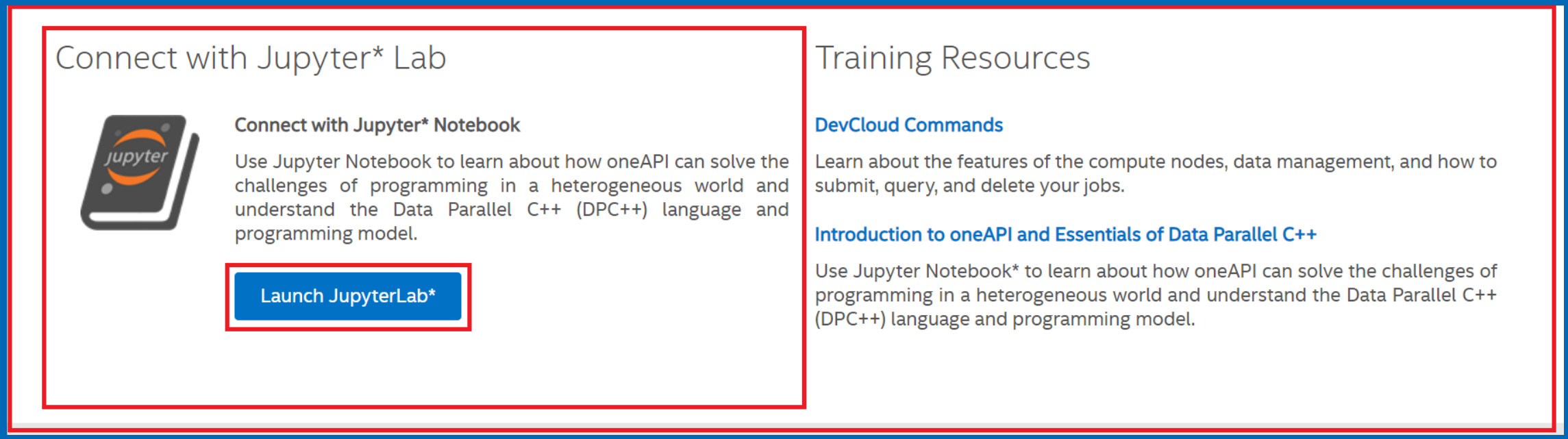

Intel Unified Login
consumer.intel.com/intelcorpb2c.onmicrosoft.com/B2C_1A_UnifiedLogin_SISU_CML_SAML/api/SelfAsserted/confirmed?csrf_token=Tkxl...
Incognito (2)
intel
PRODUCTS
SUPPORT
SOLUTIONS
DEVELOPERS
PARTNERS
USA (ENGLISH)
Search Intel.com
Back
Please provide the following details.
Email
@gmail.com
Confirm Email
@gmail.com
First Name
Vasudha
Last Name
.
English for United Kingdom (English)
India
New Password
Confirm New Password
A lowercase character

Step:4


Intel account email verification code
Inbox ×
noreply@consumer.intel.com
to kumarivasudha91
Thu, Mar 16, 1:06 PM (4 days ago)
Verify your email address
Your Intel single-use code is: 982319
This code will expire in 20 minutes. If you did not request this, you can disregard this email.
Sincerely,
Intel

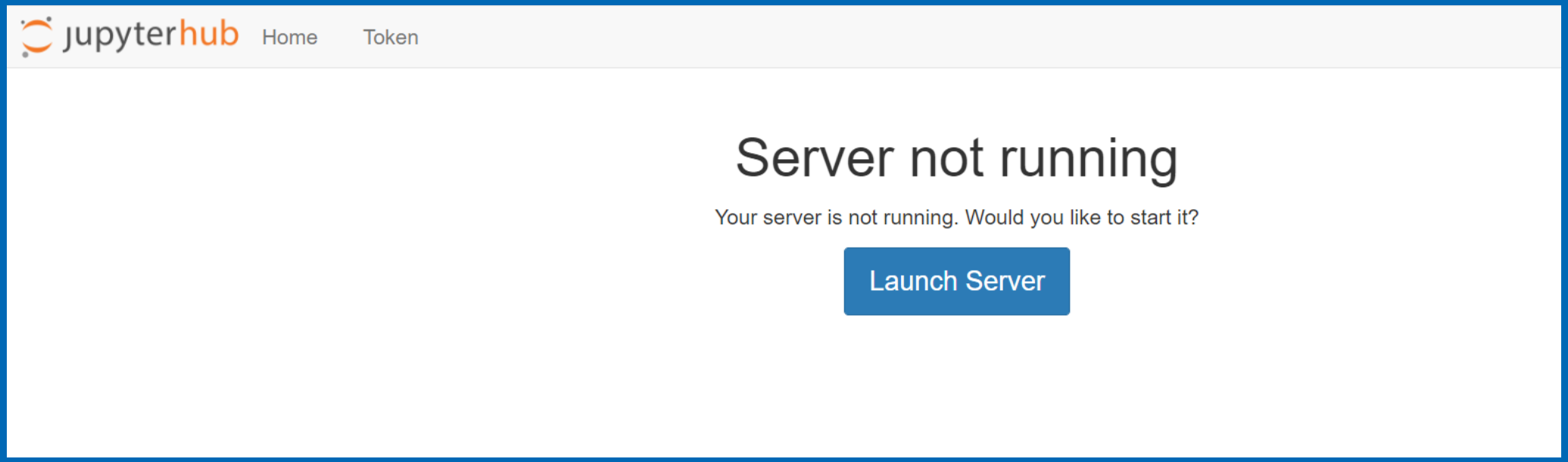
Step:5

Scroll down the page to connect with JupyterLab*



Step:6

Launch Server



Step:7

File Edit View Run Kernel Tabs Settings Help

/

Name

- Hough_Transform_on...
- Intel_oneAPI_MKL_Tr...
- MPI_with_OpenMP_o...
- oneAPI_Essentials
- OpenMP_Offload
- tmp
- xpublog
- Welcome.ipynb

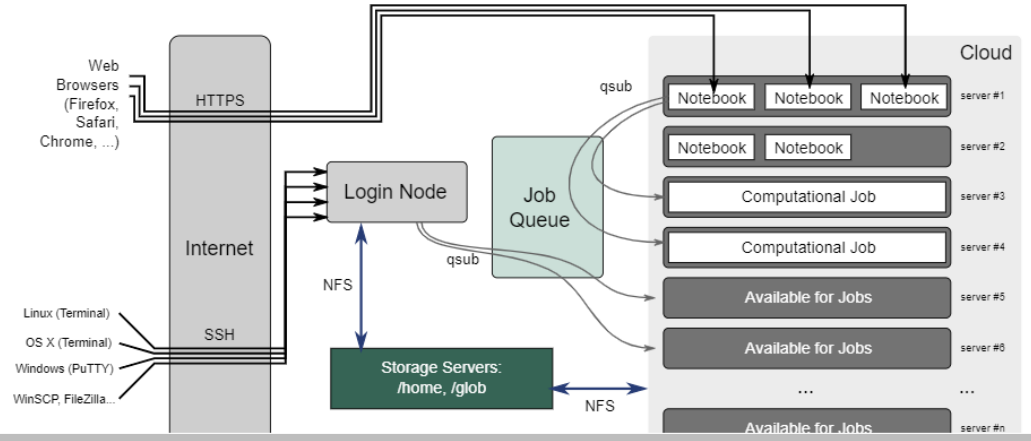
Launcher | Welcome.ipynb

Markdown

Welcome to Jupyter Notebooks on the Intel DevCloud for oneAPI Projects!

This document covers the basics of the JupyterLab access to the Intel DevCloud for oneAPI Projects. It is not a tutorial on the JupyterLab itself. Rather, we will run through a few examples of how to use the computational resources available on the DevCloud *beyond* the notebook.

The diagram below illustrates the high-level organization of the DevCloud. This tutorial explains how to navigate this organization.



Explore Resources- https://devcloud.intel.com/oneapi/

Join us on Discord for LIVE Q&A at the Hands-on Workshop

https://discord.gg/ycwqTP6

Clone one oneAPI sample from Github repo

https://github.com/oneapi-src/oneAPI-samples.git

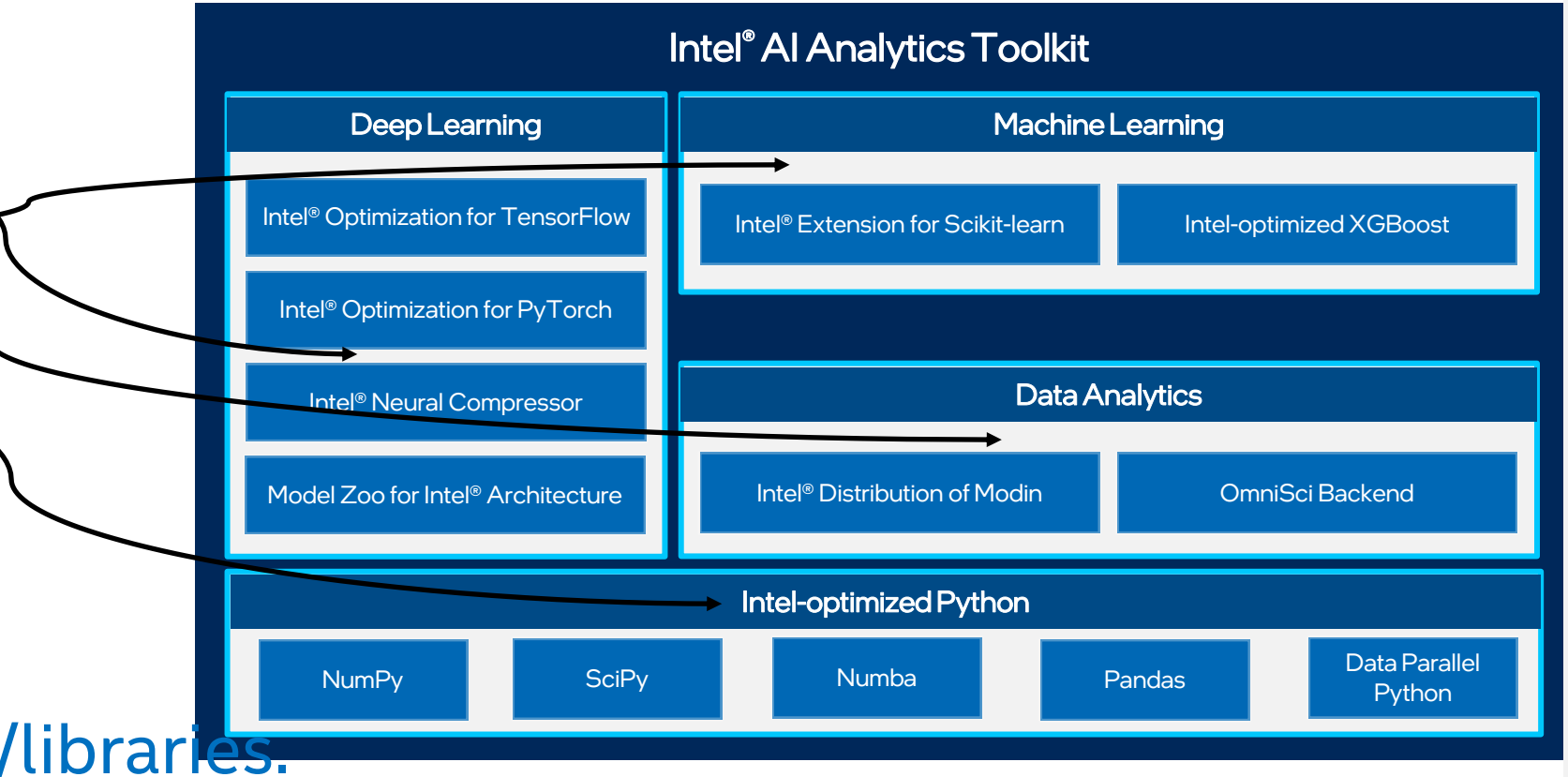
Intel® AI Analytics Toolkit Components

Maximize Hardware Value with Intel-optimized Software

Intel® AI Analytics Toolkit

- oneDNN
- oneDAL
- oneMKL

- You can install AI Analytics toolkit for all the packages/libraries. Or can install standalone package.



CPU GPU

Hardware support varies by individual tool. Architecture support will be expanded over time.

Get the Toolkit HERE or via these locations

Intel Installer | Docker | Apt, Yum | Conda | Intel® DevCloud

Learn More: software.intel.com/oneapi/ai-kit

Back to Domain-specific Toolkits for Specialized Workloads

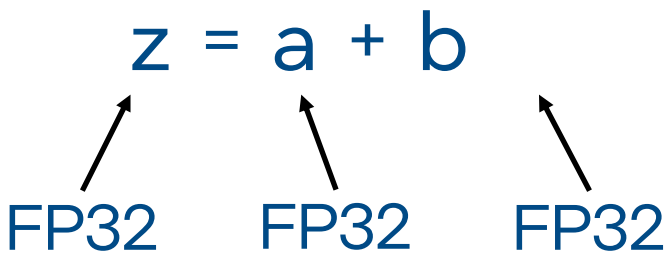
Getting Started with Intel® oneAPI AI Analytics Toolkit

Overview	Installation	Hands on	Learning	Support
▪ Visit Intel® oneAPI AI Analytics Toolkit (AI Kit) for more details and up-to-date product information ▪ Release Notes	▪ Download the AI Kit from Intel, Anaconda or any of your favorite package managers ▪ Get started quickly with the AI Kit Docker Container ▪ Installation Guide ▪ Utilize the Getting Started Guide	▪ Code Samples ▪ Build, test and remotely run workloads on the Intel® DevCloud for free. No software downloads. No configuration steps. No installations.	▪ Machine Learning & Analytics Blogs ▪ Intel AI Blog site ▪ Webinars & articles	▪ Ask questions and share information with others through the Community Forum ▪ Discuss with experts at AI Frameworks Forum

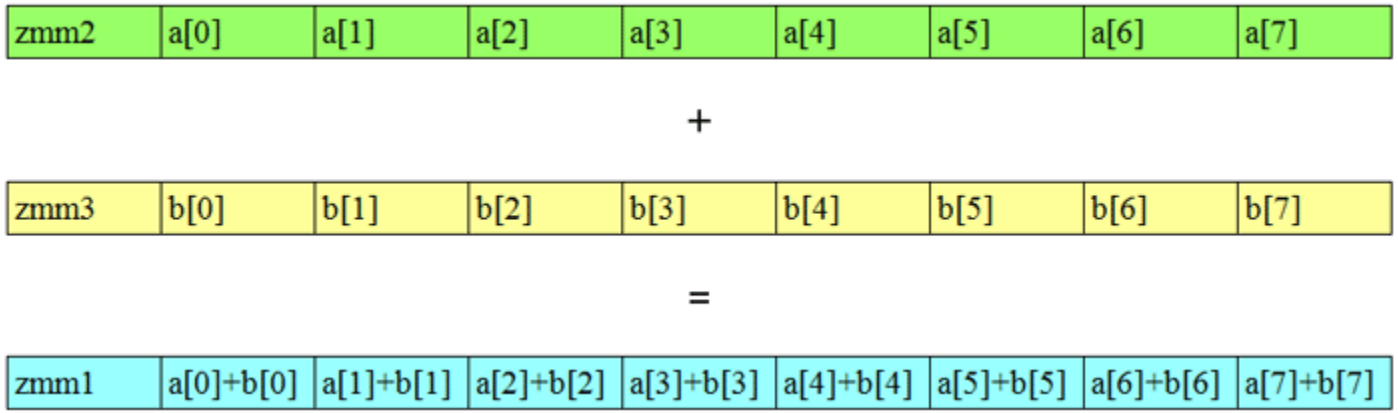
Download Now

Intel® oneAPI Data Analytics Library(oneDAL)

Vectorization

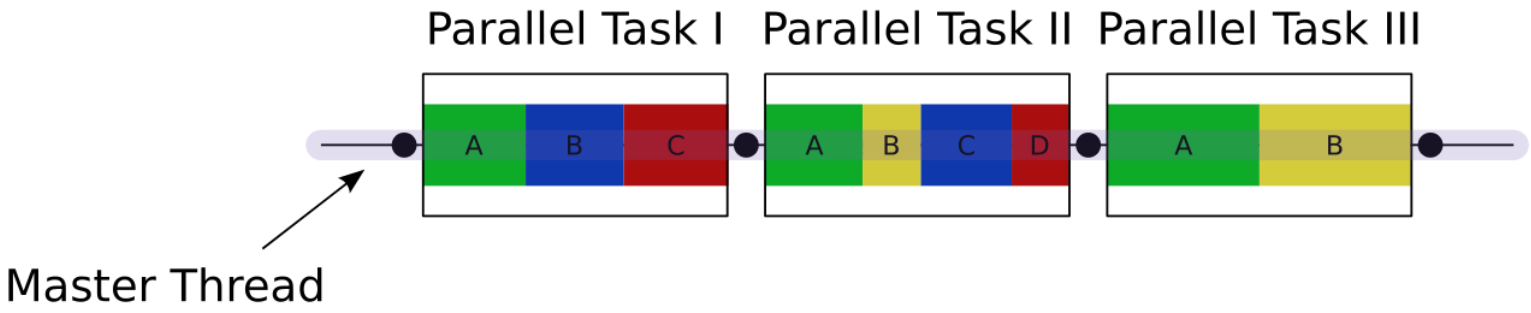


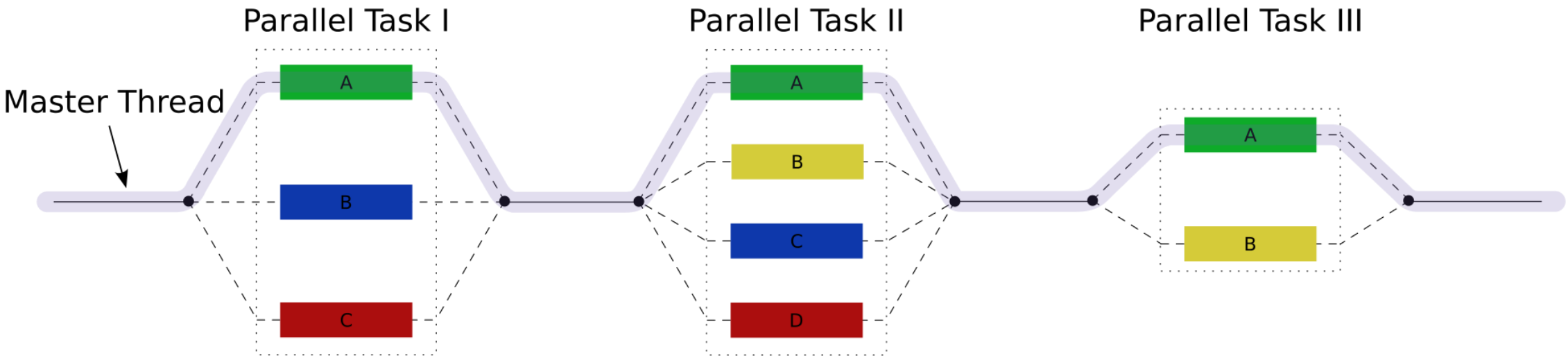
ISA	Length	Num of FP32
AVX	128 bits	4
AVX2	256 bits	8
AVX512	512 bits	16



https://www.intel.com/content/www/us/en/developer/articles/technical/improve-performance-with-vectorization.html

Parallelization



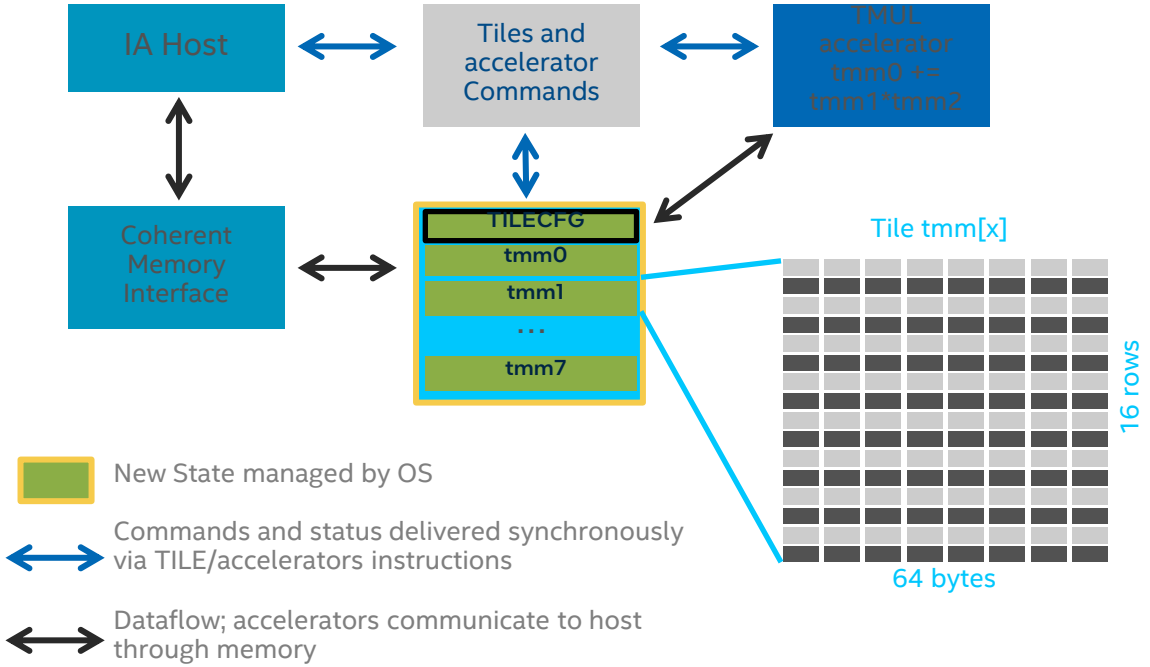


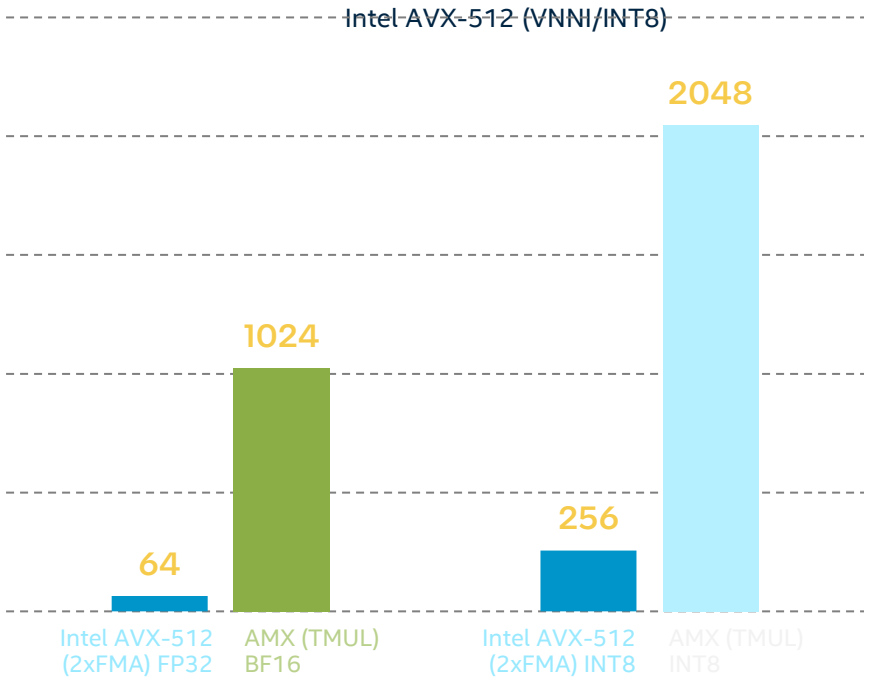
https://en.wikipedia.org/wiki/OpenMP

Intel® Deep Learning

Built-In AI Acceleration Engine

2nd Gen Intel® Xeon® Scalable Processors	3rd Gen Intel® Xeon® Scalable Processors	Sapphire Rapids
Intel Deep Learning Boost (Intro)	**Intel Deep Learning Boost**	**Intel Deep Learning Boost**
Intel® AVX-512 **(VNNI/INT8)**	Intel AVX-512: VNNI/INT8 (CPX/ICX) & **BFloat16 (CPX)**	**Intel AMX – INT8 and BFloat16 support**





Intel AMX programing reference: https://software.intel.com/content/dam/develop/external/us/en/documents-tps/architecture-instruction-set-extensions-programming-reference.pdf

Results have been simulated. For workloads and configurations visit www.intel.com/ArchDay21claims. Results may vary.

Intel® Extension for Scikit Learn*(oneDAL)

Get Faster scikit-learn for Accelerated Data Analytics & Machine Learning

- Intel® Extension for Scikit-learn* offers you a way to accelerate existing scikit-learn code.
- The software acceleration provided by Intel® Extension for Scikit-learn* is achieved through the use of vector instructions, IA hardware-specific memory optimizations, threading, and optimizations for all upcoming Intel platforms at launch time.
- The acceleration is achieved through patching: replacing the stock scikit-learn algorithms with their optimized versions provided by the extension.
- Command line: `python –m sklearnex my_code.py`
- Inside script/jupyter notebook:
```
from sklearnex import patch_sklearn
patch_sklearn()
```
- The extension is part of the Intel® AI Analytics Toolkit (AI Kit) that provides flexibility to use machine learning tools with your existing AI packages.
- The top benefits are:
  - No up-front cost to learning a new API
  - Integration with the Python* ecosystem
  - Up to 100x better performance and accuracy than stock scikit-learn

Reference: https://www.intel.com/content/www/us/en/developer/tools/oneapi/scikit-learn.html#gs.iyytz7
Installation Guide: https://intel.github.io/scikit-learn-intelex/installation.html

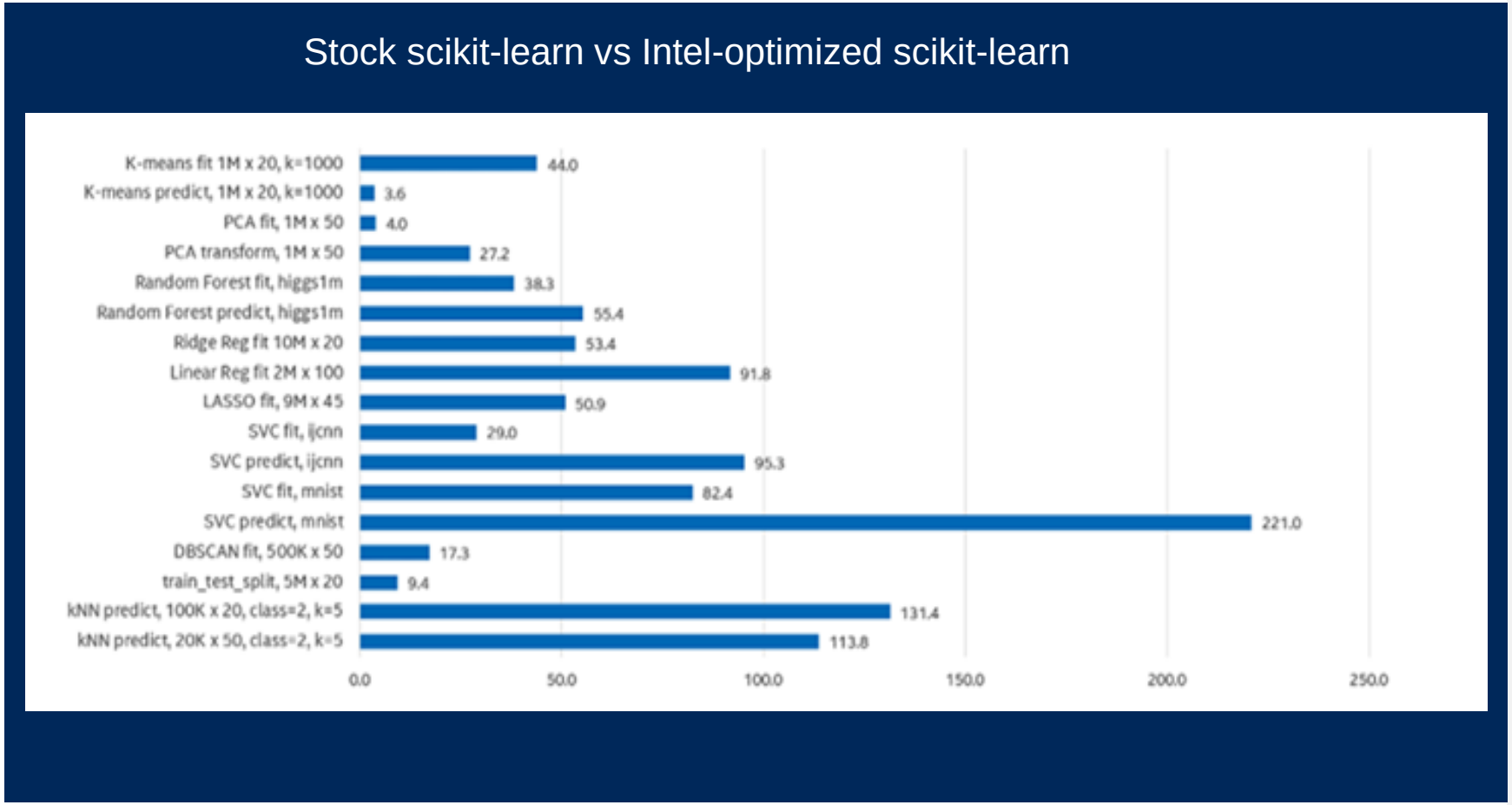
ML Performance with Intel-optimized scikit-learn *

```
from sklearn.svm import SVC
X, Y = get_dataset()

clf = SVC().fit(X, y)
res = clf.predict(X)
```

Common Scikit-learn (mainline)

```
from sklearnex import
patch_sklearn
patch_sklearn()
from sklearn.svm import SVC
X, Y = get_dataset()

clf = SVC().fit(X, y)
res = clf.predict(X)
```

Scikit-learn on Intel CPU optimized by Intel® AI Analytics Toolkit



Easy as adding two lines of code

*Measured March 2021

For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.
See backup for configuration details.

Intel® Extension for scikit-learn*

- Install scikit-learn-intelex:

  pip install scikit-learn-intelex

- Code Change: Add the patch

  ```
  from sklearnex import patch_sklearn
  patch_sklearn()
  ```

- Code Change: Remove the patch

  ```
  from sklearnex import unpatch_sklearn
  unpatch_sklearn()
  ```

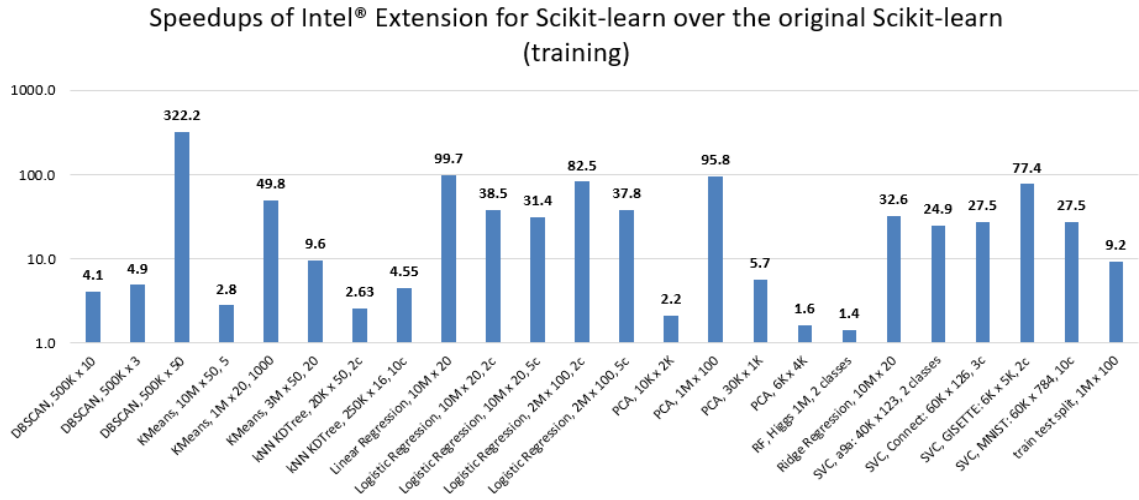
Links: intel-sklearnex Docs Code Sample

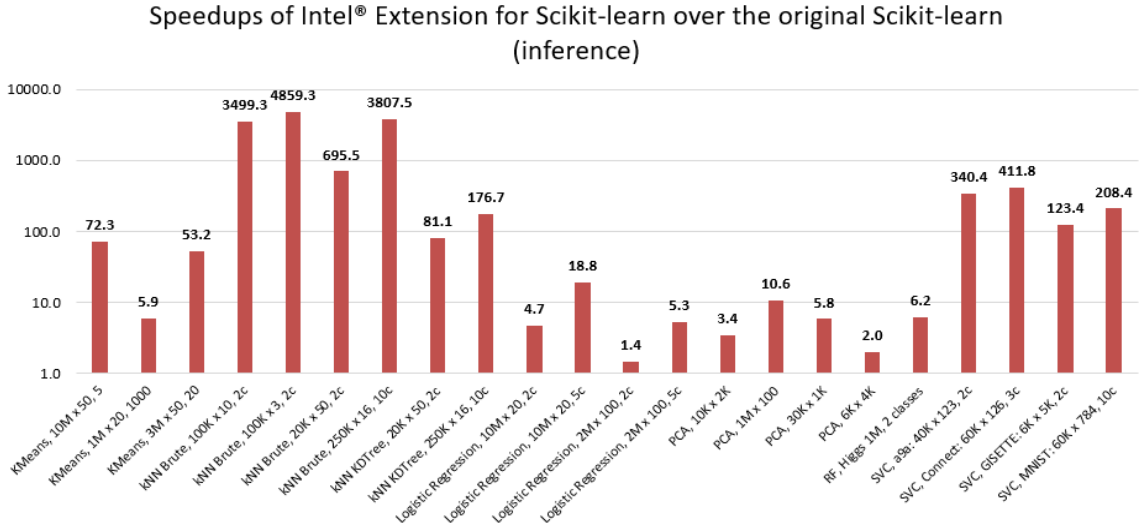
Intel® Extension for Scikit Learn*

Get Faster scikit-learn for Accelerated Data Analytics & Machine Learning

HW: c5.24xlarge AWS EC2 Instance using an Intel Xeon Platinum 8275CL with 2 sockets and 24 cores per socket

SW: scikit-learn version 0.24.2, scikit-learn-intelex version 2021.2.3, Python 3.8





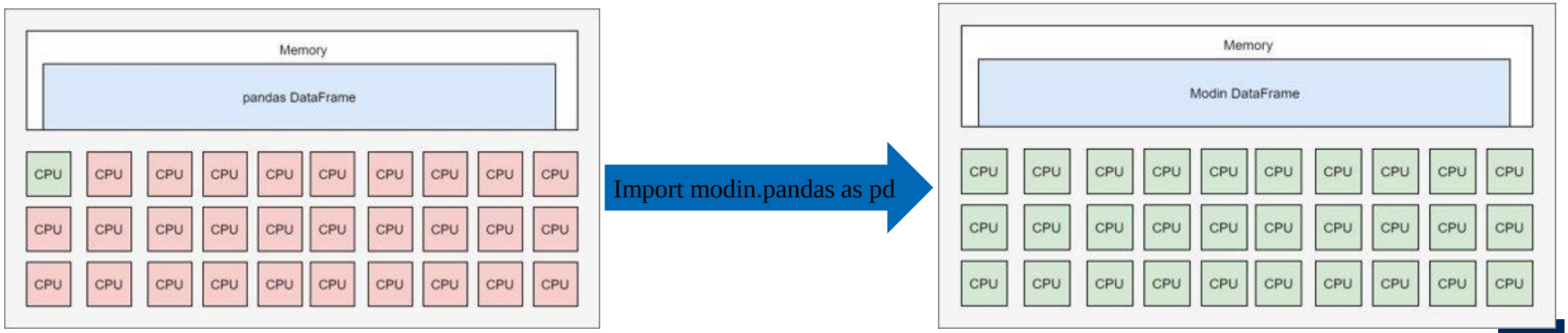
Details https://github.com/intel/scikit-learn-intelex#-acceleration

Intel® Distribution of Modin*

Scale your pandas workflows by changing a single line of code.

Features

- Modin* is a drop-in replacement for pandas, enabling data scientists to scale to distributed DataFrame processing without having to change API code.
- Accelerated DataFrame Processing
  - Modin transparently **distributes the data and computation across available cores**, unlike Pandas which only uses one core at a time
- Compatible with Existing APIs and Engines
  - As of 0.9 version, **Modin supports 100% of Pandas API**
  - Use Dask*, Ray, or HEAVY.AI compute engines to distribute the data without having to write code.
  - Continue to use the rest of your Python ecosystem code, such as NumPy, XGBoost, and scikit-learn*



Pandas* on Big Machine

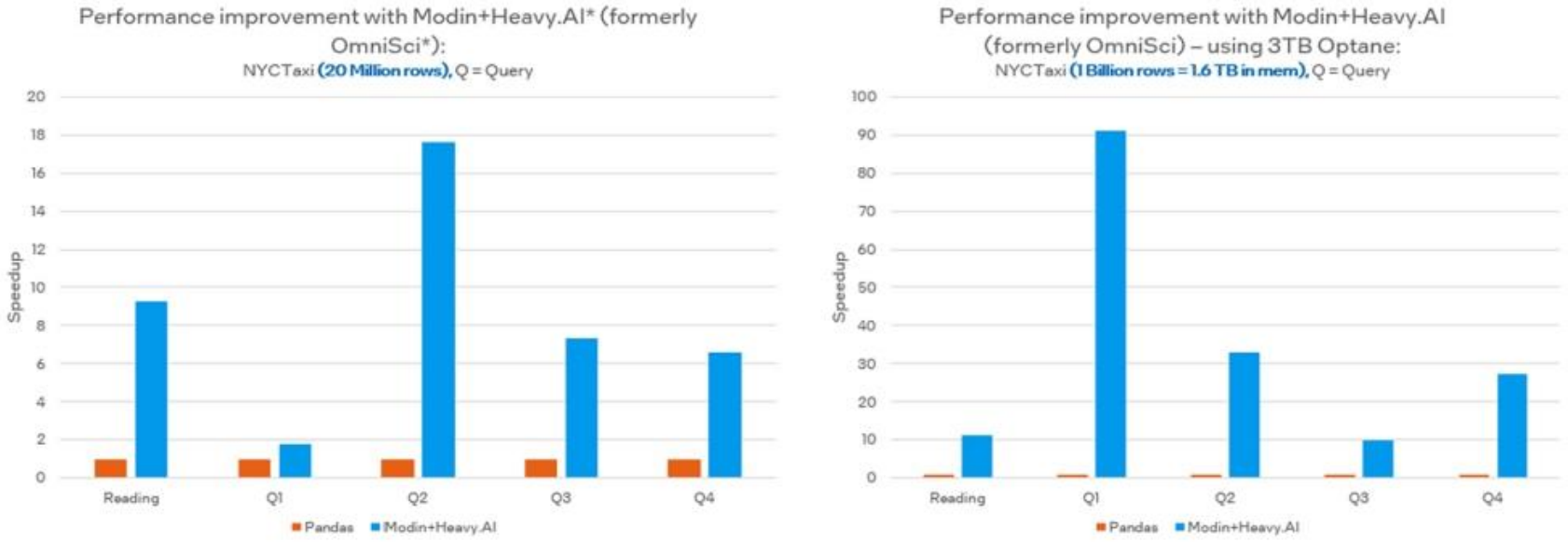
Modin on Big Machine

Reference :https://www.intel.com/content/www/us/en/developer/tools/oneapi/distribution-of-modin.html#gs.iyyyql
Installation Guide: https://www.intel.com/content/www/us/en/developer/articles/guide/distribution-of-modin-getting-started-guide.html

Intel® Distribution of Modin*

Scale your pandas workflows by changing a single line of code.

Accelerate Pandas Workflows with Intel® Distribution of Modin*



Testing Date: Performance results are based on testing by Intel as of **February 19, 2021** and may not reflect all publicly available security updates.

Q=query. Dataset source: https://github.com/toddwschneider/nyc-taxi-data

This dataset consists of up to 1.1 billion individual taxi trips in the city of New York from January 2009 through June 2015, covering both yellow and green taxis. The NYC taxi workload ingests the large dataset into a dataframe and performs queries over them.

Configuration Details and Workload Setup: For 20 million rows: Dual-socket Intel® Xeon® Platinum 8280L CPUs (S2600SFT platform), 28 cores per socket, hyperthreading-enabled , turbo modo-enabled, NUMA nodes per socket =2, BIOS: SE5C620.86B.02.01.0013.121520200651, kernel: 5.4.0-65-generic, microcode: 0x4003003, OS: Ubuntu 20.04.1 LTS, CPU governor: performance, transparent huge pages: enabled, System DDR Mem Config: slots/cap/speed: 12 slots/32GB/2933MHz, total memory per node: 384 GB DDR RAM, boot drive: INTEL SSDSC2BB800G7. 1 billion rows: Dual socket Intel Xeon Platinum 8260M CPU, 24 cores per socket, 2.40GHz base frequency, DRAM memory: 384 GB 12x32GB DDR4 Samsung @ 2666 MT/s 1.2V, Optane memory: 3TB 12x256GB Intel® Optane™ technology @ 2666MT/s, kernel: 4.15.0-91-generic, OS: Ubuntu 20.04.4.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.

Performance varies by use, configuration, and other factors. Learn more at www.Intel.com/PerformanceIndex. Your costs and results may vary.

There will still be a need to tune for each architecture.

Modin: Drop-in replacement for Pandas

- Modin Installation:

```
pip install modin[all]
```

- Code change:

```
import modin.pandas as pd
df = pd.read_csv('my_data.csv')
```

Links: intel-modin Docs Code Sample

Intel® oneAPI Deep Neural Network Library(oneDNN)

Intel® oneDNN overview

Feature:

- Highly vectorized and threaded building blocks
- Performance critical functions
- Training (float32, bfloat16) and inference (float32, int8)
- CNNs (1D, 2D and 3D), RNNs (plain, LSTM, GRU)

Portability:

- Compilers: Intel C++ compiler/Clang/GCC/MSVC*
- OS: Linux*, Windows*, Mac*
- Threading: OpenMP*, TBB

	Intel® oneDNN
Convolution	2D/3D Direct Convolution/Deconvolution, Depthwise separable convolution 2D Winograd convolution
Inner Product	2D/3D Inner Production
Pooling	2D/3D Maximum 2D/3D Average (include/exclude padding)
Normalization	2D/3D LRN across/within channel, 2D/3D Batch normalization
Eltwise (Loss/activation)	ReLU(bounded/soft), ELU, Tanh; Softmax, Logistic, linear; square, sqrt, abs, exp, gelu, swish
Data manipulation	Reorder, sum, concat, View
RNN cell	RNN cell, LSTM cell, GRU cell
Fused primitive	Conv+ReLU+sum, BatchNorm+ReLU
Data type	f32, bfloat16, s8, u8

Intel® Optimization for PyTorch*

- Intel Extension for PyTorch* extends PyTorch with optimizations for an extra performance boost on Intel hardware.
- It delivers up-to-date features and optimizations for PyTorch on Intel hardware, examples include AVX-512 Vector Neural Network Instructions (AVX512 VNNI) and Intel® Advanced Matrix Extensions (Intel® AMX).
- The optimizations are built using oneDNN to provide cross-platform support and acceleration.

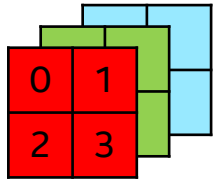
```python
import torch
import torchvision.models as models

model = models.resnet50(pretrained=True)
model.eval()
data = torch.rand(1, 3, 224, 224)

#################### code changes ####################
import intel_extension_for_pytorch as ipex
model = ipex.optimize(model)
######################################################

with torch.no_grad():
  d = torch.rand(1, 3, 224, 224)
  model = torch.jit.trace(model, d)
  model = torch.jit.freeze(model)

  model(data)
```

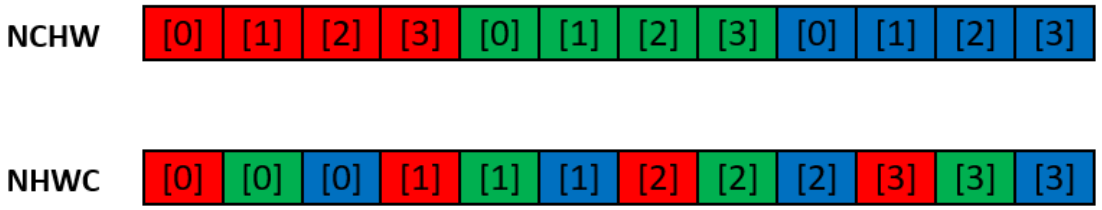
Features

- Easy-to-use Python* API
- Operator Optimizations
- Quantization
- Graph Optimization
- Auto Mixed Precision
- Channel last

Reference: https://intel.github.io/intel-extension-for-pytorch/cpu/latest/
Installation Guide: https://intel.github.io/intel-extension-for-pytorch/cpu/latest/tutorials/installation.html

Memory layout





- Used in Vision workloads
- NCHW
  - Default format
  - *torch.contiguous_format*
- **NHWC**
  - A working-in-progress feature of PyTorch
  - *torch.channels_last*
  - NHWC format yields higher performance

```
## NB: internally blocked format will still be used.
##   aka. we do 'reorder' for 'input', 'weight' and 'output',
##   and believe me this is expensive, roughly 50% perf loss...
input = torch.randn(1, 10, 32, 32)
model = torch.nn.Conv2d(10, 20, 1, 1)
output = model(input)
```

```
input = torch.randn(1, 10, 32, 32)
model = torch.nn.Conv2d(10, 20, 1, 1)
## NB: convert to Channels Last memory format.
##   oneDNN supports NHWC for feature maps (input, output),
##   but weight still needs to be of blocked format.
##   Still we can save reorders for feature maps.
input = input.to(memory_format=torch.channels_last)
model = model.to(memory_format=torch.channels_last)
output = model(input)
```

Intel® Extension for PyTorch*(IPEX)

- Install PyTorch CPU:

  pip install torch==1.13.1+cpu torchvision==0.14.1+cpu -f https://download.pytorch.org/whl/torch_stable.html

- Install Intel Extension for PyTorch(IPEX): need same version as torch

  pip install intel_extension_for_pytorch==1.13.100 -f https://developer.intel.com/ipex-whl-stable-cpu

- Code changes:

- TorchScript is preferred for Inference with IPEX:

```python
import torch
import torchvision.models as models

model = models.resnet50(pretrained=True)
model.eval()
data = torch.rand(1, 3, 224, 224)

#################### code changes ####################
import intel_extension_for_pytorch as ipex
model = ipex.optimize(model)
######################################################

with torch.no_grad():
  d = torch.rand(1, 3, 224, 224)
  model = torch.jit.trace(model, d)
  model = torch.jit.freeze(model)

  model(data)
```

Links: IPEX Docs Code Sample

Intel® Optimization for TensorFlow

- TensorFlow* is a widely used deep-learning framework that's based on Python*. It's designed for flexible implementation and extensibility on modern deep neural networks.
- The platforms use the Intel® oneAPI Deep Neural Network Library (oneDNN), an open-source, cross-platform performance library for deep-learning applications.
- Enable those Intel **oneDNN** CPU optimizations by setting the the environment variable **TF_ENABLE_ONEDNN_OPTS=1** for the **official x86-64 TensorFlow after v2.5**.
- Since TensorFlow v2.9, the oneAPI Deep Neural Network Library (oneDNN) optimizations are enabled by default.

Features

- Operator Optimization
- Graph Optimization
- Advanced Auto Mixed Precision
- Ease-of-use Python API
- GPU Profiler
- CPU Launcher [experimental]
- INT8 Quantization

Reference: https://github.com/intel/intel-extension-for-tensorflow
Installation guide: https://www.intel.com/content/www/us/en/developer/articles/guide/optimization-for-tensorflow-installation-guide.html

Intel® Optimization for TensorFlow*

- Install TensorFlow:

  pip install tensorflow

  - Enable or Disable oneDNN using env var: *TF_ENABLE_ONEDNN_OPTS=1 or 0*
- Install Intel Optimization for TensorFlow:

  pip install intel-tensorflow
- Code changes: Nothing
- Intel® Extension for TensorFlow* is also available for TensorFlow support on Intel CPUs and discrete GPUs.

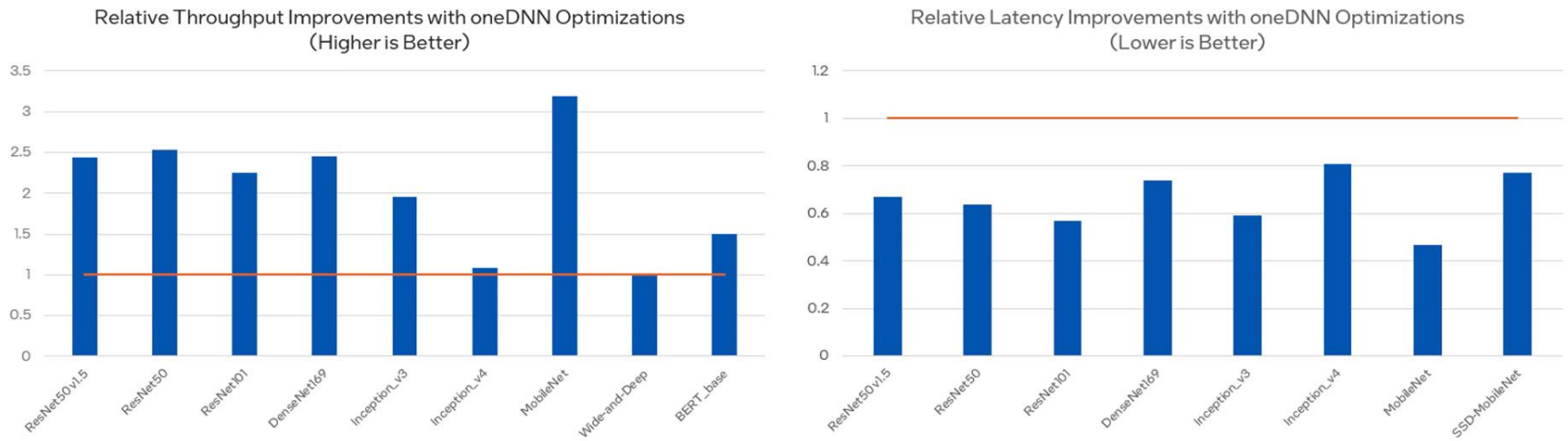
  pip install tensorflow==2.12.0
  pip install --upgrade intel-extension-for-tensorflow[cpu]
  pip install --upgrade intel-extension-for-tensorflow[gpu] For dGPU

Links: Intel-TF Docs ITEX

Intel® Optimization for TensorFlow

Performance Benefits of Intel® oneAPI Deep Neural Network Library (oneDNN) with TensorFlow* 2.8

FP32 Inference Workloads



Testing Date: Performance results are based on testing by Intel as of **March 5, 2022** and may not reflect all publicly available security updates.

Configuration Details and Workload Setup: Amazon Web Services* (AWS) EC2 C6i and instances powered by 3rd Generation Intel Xeon Scalable processors (code named Ice Lake) with an all-core turbo frequency of 3.5 GHz. Kernel: 5.11.0-1019-aws x86_64. OS: Ubuntu* 20.04.2. TensorFlow* version 2.8. Workload: Model Zoo for Intel® Architecture v2.5. Data type: FP32.

Throughput measurements: AWS c6i.12xlarge instance type with 24 physical CPU cores and 96 GB memory on a single socket. Latency measurements: AWS c6i.2xlarge instance type with 4 physical CPU cores and 8 GB memory on a single socket.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.

Performance varies by use, configuration, and other factors. Learn more at www.Intel.com/PerformanceIndex. Your costs and results may vary.

Intel® OpenMP for multithreading

- Intel OpenMP is available with IPEX and Intel-TensorFlow. It is to further optimize the workload.
- OMP_NUM_THREADS=<number of physical cores per socket> OpenMP version of intra_op_parallelism_threads. For running a single operation/function on multiple threads. And these threads are bound to Physical cores. Sometimes keeping this number at n is not optimal due to overhead. So try with smaller values too.
- KMP_AFFINITY=granularity=fine,compact,1,0 Tell, how to bound the threads to cores.
- KMP_BLOCKTIME=0 or 1 0 for CNN based model
- KMP_SETTINGS=1
- One example is :

```
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"
```

  KMP_BLOCKTIME=1 KMP_SETTINGS=1 OMP_NUM_THREADS=8 KMP_AFFINITY=granularity=fine,verbose,compact,1,0 python infer.py

- Or we can also export all these in env like this: export OMP_NUM_THREADS=8
- NUMACTL can also be used if we are having multi-node machine.
- Example: KMP_BLOCKTIME=1 KMP_SETTINGS=1 OMP_NUM_THREADS=12 KMP_AFFINITY=granularity=fine,verbose,compact,1,0 numactl –m 0,1 –N 0,1 –C 0-11 python infer.py

Links: TensorFlow optimization Guide PyTorch

Intel® Neural Compressor

EASY QUANTIZATION

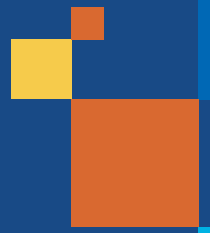
Quantization & Optimization by Simple API

Support TensorFlow, PyTorch, ONNX, MXNet Model

Automatic Accuracy-driven

Open-sourced Python Tool

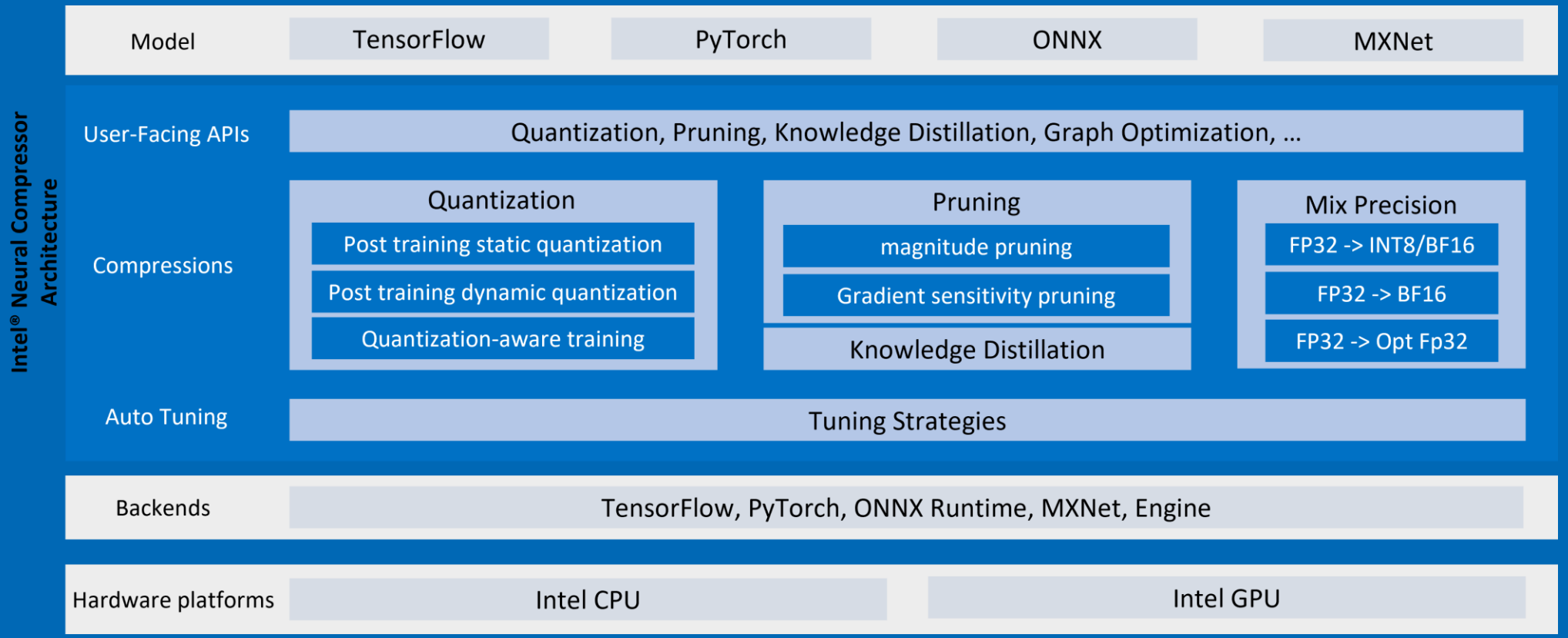
Used by **Alibaba, Tencent, ByteDance, Kuaishou, MSFT and HuggingFace & ONNX**



intel®

Architecture

- Technology
  - Quantization
  - Pruning
  - Knowledge distillation
  - Graph Optimization
  - Mix Precision



Intel® Neural Compressor(INC)

- Install Intel Neural Compressor:

  pip install neural-compressor

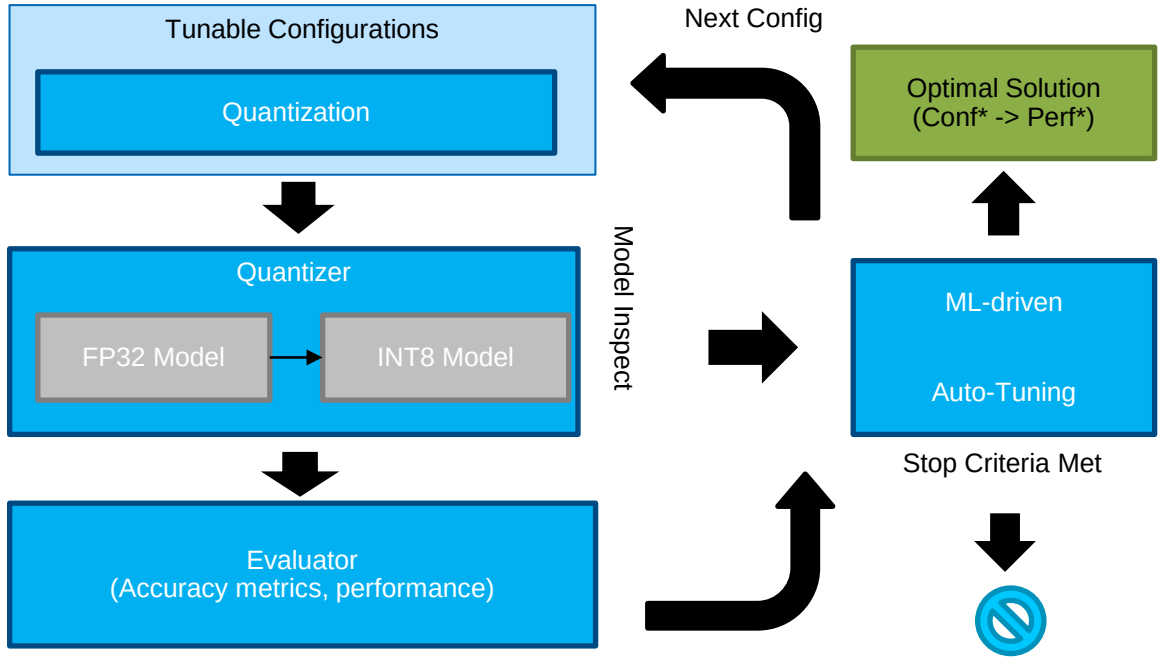
- Quantization & Optimization by Simple API
- Support TensorFlow, PyTorch, ONNX, MXNet Model
- Automatic Accuracy-driven
- Open-sourced Python Tool

```
from neural_compressor.config import PostTrainingQuantConfig
from neural_compressor.data import DataLoader
from neural_compressor.data import Datasets

dataset = Datasets('tensorflow')['dummy'](shape=(1, 224, 224, 3))
dataloader = DataLoader(framework='tensorflow', dataset=dataset)

from neural_compressor.quantization import fit
config = PostTrainingQuantConfig()
q_model = fit(
    model="./mobilenet_v1_1.0_224_frozen.pb",
    conf=config,
    calib_dataloader=dataloader,
    eval_dataloader=dataloader)
```

Links: INC Docs Code Sample

Intel® Neural Compressor

Support Multiple Deep Learning Frameworks,
One Click to Create High Performance Compressed Model



2.2X Performance based on VNNI, Accuracy Loss <1%

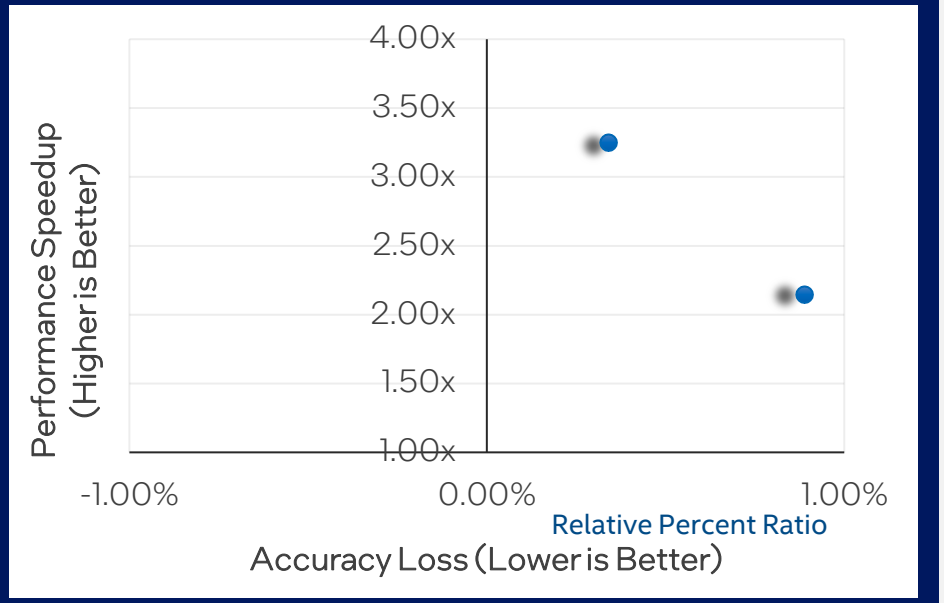
Performance

Stock TF on ICX (FP32)	oneDNN Enabled (FP32)	INC Quantization (INT8)	AMX OPTs on SPR (INT8)
Baseline	1.5x	3.9x	4.8x
13.06	20.54	81.66	394

30x Total Perf Gain

Customer/LZ model: SSD-ResNet-34
(BS=1)

Model Accuracy & Performance

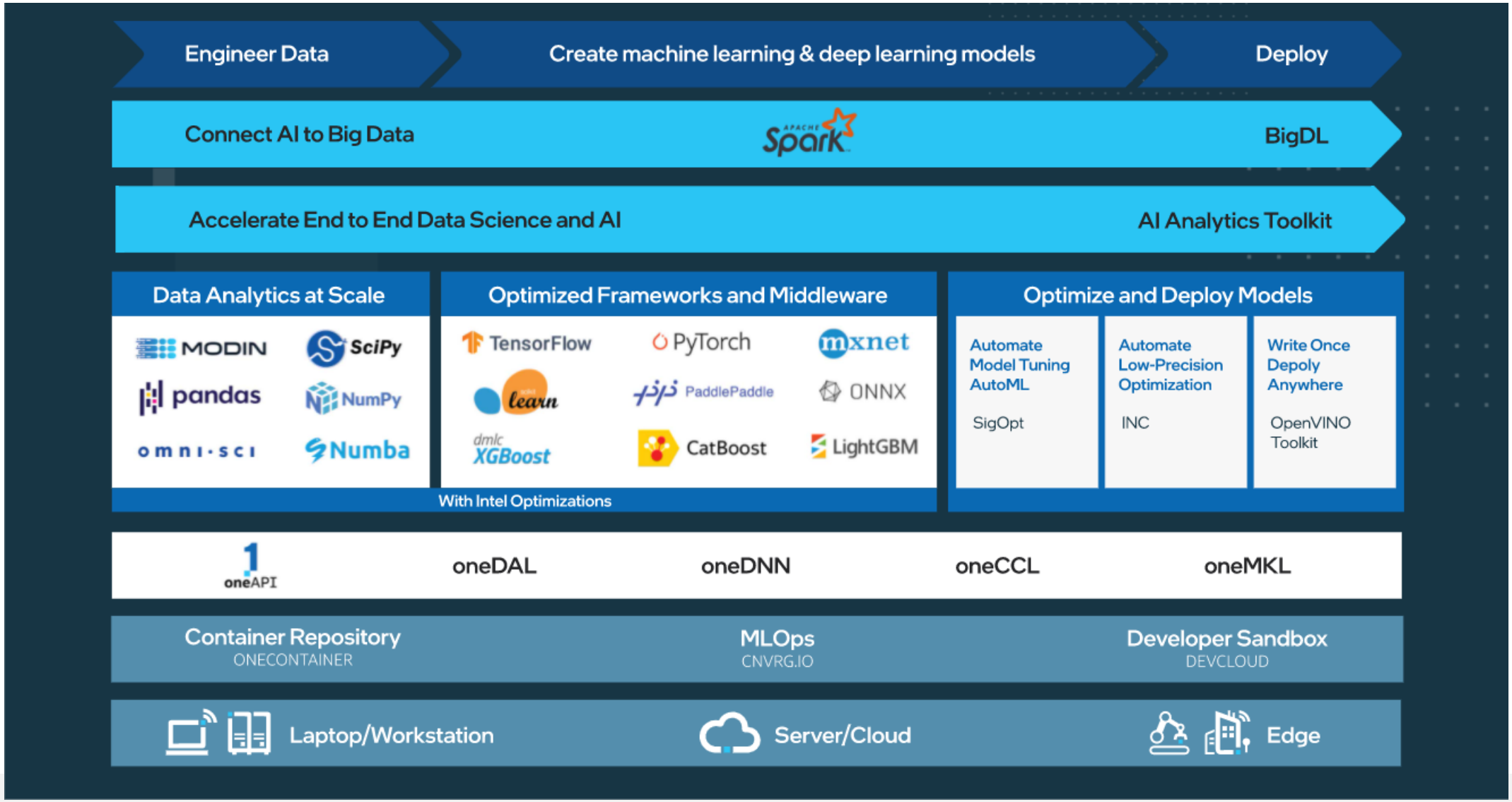


OOB Random Models
(w/ VNNI example)

2.2x geomean and up to 4x performance

AI Software Stack for Intel® XPUs

Intel offers a Robust Software Stack to Maximize Performance of Diverse Workloads



Notices & Disclaimers

Intel is committed to respecting human rights and avoiding complicity in human rights abuses. See Intel's Global Human Rights Principles. Intel® products and software are intended only to be used in applications that do not cause or contribute to a violation of an internationally recognized human right.

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex. Results may vary.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details.
No product or component can be absolutely secure.

Your costs and results may vary.

Intel technologies may require enabled hardware, software or service activation.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

Intel® Extension for PyTorch* is licensed under Apache License Version 2.0. This software includes components that have separate copyright notices and licensing terms. Your use of the source code for these components is subject to the terms and conditions of the following licenses.

Poll Question 3


intel