intel ai summit
英特爾AI科技論壇

# Bringing AI Everywhere

## Optimizing AI Inference on Intel Core Ultra through ONNX Runtime OpenVINO Execution Provider

John Feng, Mayuresh Varerkar, Sahar Fatima

Mar 27th , 2024

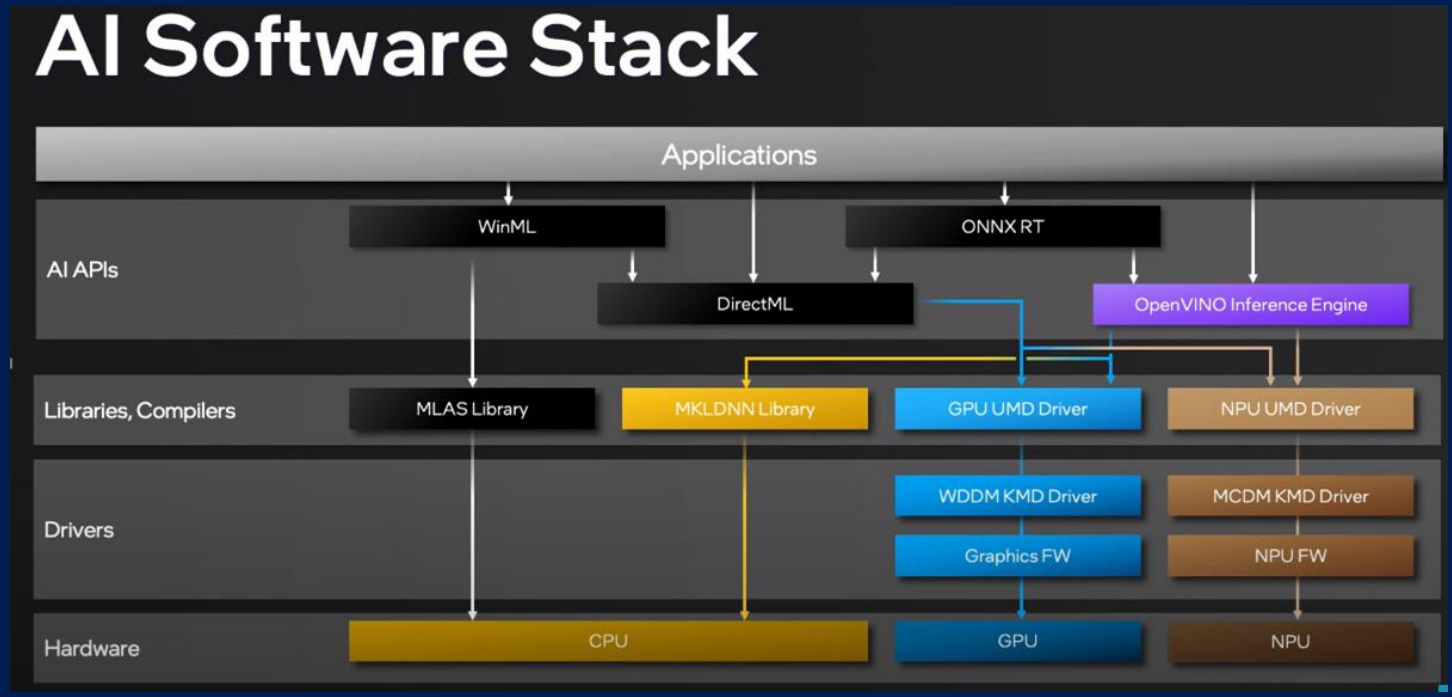
# Agenda

This session will focus on running inference on Intel accelerators using ONNX Runtime OpenVINO Execution Provider APIs and will cover:

- Role of ONNX Runtime in Intel AI Inference Strategy
- Why ONNX RT OV EP ?
- Inference stack for ONNX RT OV EP
- Demo
- OpenVINO EP Provider options
- Deployment – (Packaging & Distribution)
- Pointer to samples/ how to get started

# Role of ONNX Runtime in Intel AI Inference Strategy

- Client AI Inference Frameworks for Apps
  - OpenVINO
  - ONNX Runtime (ONNX RT)
  - Windows ML (WinML)
- ONNX RT is Microsoft backed industry framework
  - X-IHV, X-OS support
  - Embraces Open Neural Network eXchange (ONNX) as standard model format
- ONNX RT APIs supports inferencing on Intel devices through Execution Providers (EPs)
  - Default EP (Math Linear Algebra Library)
  - DirectML Execution Provider
  - OpenVINO Execution Provider




intel AI summit

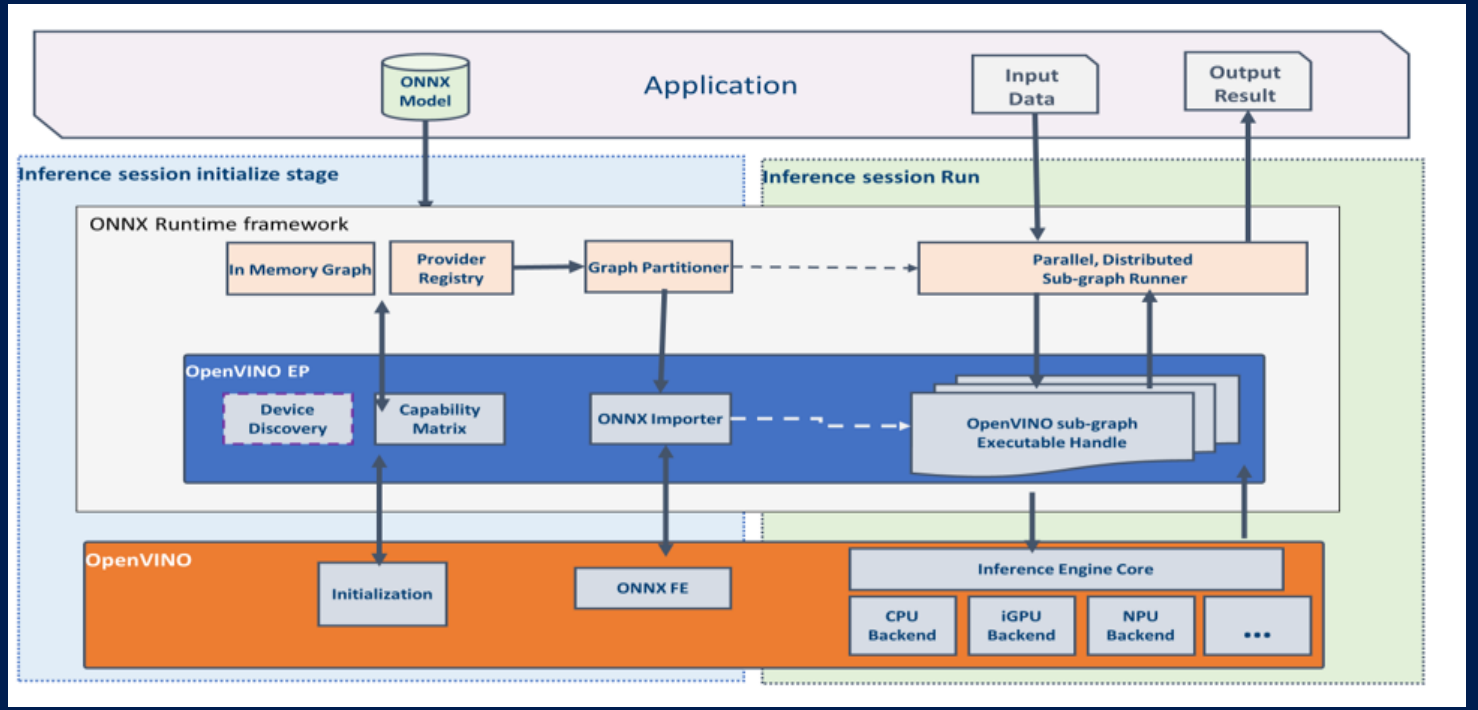
# Why ONNX RT OpenVINO EP ?

- EP helps expose Intel specific device capabilities and OpenVINO features for ONNX RT app developers
- Graph Partitioning: OpenVINO™ unsupported operators fallback to MLAS on CPU
- Close to native OpenVINO performance for FP16/FP32 precision, Support for ONNX QDQ Models
- Highly desired OpenVINO features can be exposed through OV EP
  - Dynamic shape support on CPU, GPU
  - Model Caching support to improve First Inference Latency
- Simple API change to switch between EPs (DML -> OpenVINO EP)
  - SessionOptionsAppendExecutionProvider_OpenVINO(session_options, &options);

Optimized Performance on Intel devices using standard ONNX RT APIs

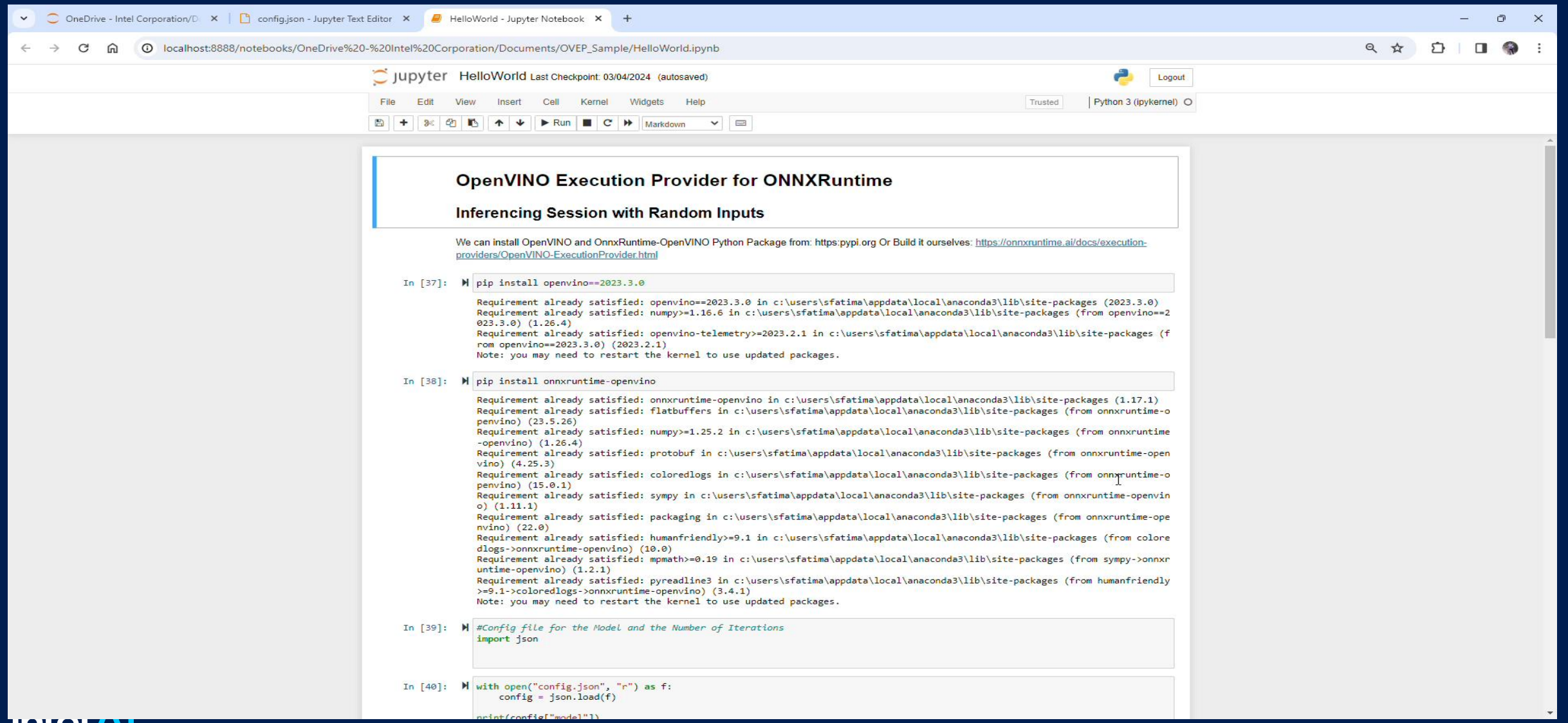
# Inference stack for ONNX RT OpenVINO EP

- EP helps expose Intel specific device capabilities, OpenVINO features and PnP KPIs for ORT app developers
- Session Initialization (ONNX Model Input, Provider Options)
  - Loads ONNX Model and converts to in memory representation
  - Queries capability matrix inside EP to trigger graph partitioning if needed
  - Supported graph is converted to OpenVINO model using ONNX FE
  - OpenVINO model is compiled for the specific device and cached for subsequent inferences
- Session Run (Input Data)
  - Fills input tensors based on input data from application
  - Computes inference on targeted device
  - Returns output tensor  to application




Application interfaces with high level APIs: Session Init and Run

# Demo


OneDrive - Intel Corporation/D
config.json - Jupyter Text Editor
HelloWorld - Jupyter Notebook
localhost:8888/notebooks/OneDrive%20-%20Intel%20Corporation/Documents/OVEP_Sample/HelloWorld.ipynb
jupyter HelloWorld Last Checkpoint: 03/04/2024 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Markdown
OpenVINO Execution Provider for ONNXRuntime
Inferencing Session with Random Inputs
We can install OpenVINO and OnnxRuntime-OpenVINO Python Package from: https:pypi.org Or Build it ourselves: https://onnxruntime.ai/docs/execution-providers/OpenVINO-ExecutionProvider.html
In [37]:
pip install openvino==2023.3.0
Requirement already satisfied: openvino==2023.3.0 in c:\users\sfatima\appdata\local\anaconda3\lib\site-packages (2023.3.0)
Requirement already satisfied: numpy>=1.16.6 in c:\users\sfatima\appdata\local\anaconda3\lib\site-packages (from openvino==2023.3.0) (1.26.4)
Requirement already satisfied: openvino-telemetry>=2023.2.1 in c:\users\sfatima\appdata\local\anaconda3\lib\site-packages (from openvino==2023.3.0) (2023.2.1)
Note: you may need to restart the kernel to use updated packages.
In [38]:
pip install onnxruntime-openvino
Requirement already satisfied: onnxruntime-openvino in c:\users\sfatima\appdata\local\anaconda3\lib\site-packages (1.17.1)
Requirement already satisfied: flatbuffers in c:\users\sfatima\appdata\local\anaconda3\lib\site-packages (from onnxruntime-openvino) (23.5.26)
Requirement already satisfied: numpy>=1.25.2 in c:\users\sfatima\appdata\local\anaconda3\lib\site-packages (from onnxruntime-openvino) (1.26.4)
Requirement already satisfied: protobuf in c:\users\sfatima\appdata\local\anaconda3\lib\site-packages (from onnxruntime-openvino) (4.25.3)
Requirement already satisfied: coloredlogs in c:\users\sfatima\appdata\local\anaconda3\lib\site-packages (from onnxruntime-openvino) (15.0.1)
Requirement already satisfied: sympy in c:\users\sfatima\appdata\local\anaconda3\lib\site-packages (from onnxruntime-openvino) (1.11.1)
Requirement already satisfied: packaging in c:\users\sfatima\appdata\local\anaconda3\lib\site-packages (from onnxruntime-openvino) (22.0)
Requirement already satisfied: humanfriendly>=9.1 in c:\users\sfatima\appdata\local\anaconda3\lib\site-packages (from coloredlogs->onnxruntime-openvino) (10.0)
Requirement already satisfied: mpmath>=0.19 in c:\users\sfatima\appdata\local\anaconda3\lib\site-packages (from sympy->onnxruntime-openvino) (1.2.1)
Requirement already satisfied: pyreadline3 in c:\users\sfatima\appdata\local\anaconda3\lib\site-packages (from humanfriendly>=9.1->coloredlogs->onnxruntime-openvino) (3.4.1)
Note: you may need to restart the kernel to use updated packages.
In [39]:
#Config file for the Model and the Number of Iterations
import json
In [40]:
with open("config.json", "r") as f:
config = json.load(f)
print(config["model"])

# OpenVINO EP Provider options

- ORT OpenVINO EP Provider Options are passed as input to ORT Session options
  - Device and Precision Selection through OpenVINO
    - Physical devices/ accelerators: CPU, GPU, NPU
    - Virtual devices: AUTO, MULTI, HETERO
    - Inference precision tied to the physical device: e.g.: GPU_FP16
  - OpenVINO Model Caching
    - Specifying the path to cache location
  - CPU Programmable Parameters
    - Number of threads
    - Number of streams
  - GPU Programmable Parameters
    - OpenCL Context
- Choosing the target accelerator through OpenVINO EP for ONNX Model Execution

Ability to configure OpenVINO EPsettings through ORT Provider Options

# Deployment – (Packaging & Distribution)

- Packaging ONNX RT OpenVINO EP into Applications
  - ONNX Runtime libraries: ONNXruntime.dll, onnxruntime_providers_shared.dll
  - OpenVINO Execution provider library: onnxruntime_providers_openvino.dll
  - OpenVINO libraries: Openvino.dll, openvino_onnx_frontend.dll, <device plugins based on target device: OpenVINO_Intel_CPU_Plugin.dll, OpenVINO_Intel_GPU_Plugin.dll, OpenVINO_Intel_NPU_Plugin.dll>

- Distribution Mechanisms for ONNX RT OV EP and OpenVINO Dlls
  - Building from source
  - Precompiled runtime libraries
  - PyPi packages

# Pointer to samples/ how to get started

- **ONNX Runtime Docs:** https://onnxruntime.ai/docs/
- **OpenVINO Execution Provider:** https://onnxruntime.ai/docs/reference/execution-providers/OpenVINO-ExecutionProvider.html

- **Intel ONNX Runtime Inference Examples:** https://github.com/intel/onnxruntime-inference-examples
- Yolov8 Object Detection Sample: https://github.com/microsoft/onnxruntime-inference-examples/tree/main/python/OpenVINO_EP/yolov8_object_detection [Use OpenVINO 2023.3.0- > pip install openvino=2023.3.0]

# Notices and Disclaimers

For notices, disclaimers, and details about performance claims, visit www.intel.com/PerformanceIndex or scan the QR code:


Thank You!