OpenVINO
DEVCON中国
系列工作坊 2023

# oneAPI & OpenVINO™ 联合黑客松技术培训


武卓

英特尔 OpenVINO 布道师

杨亦诚

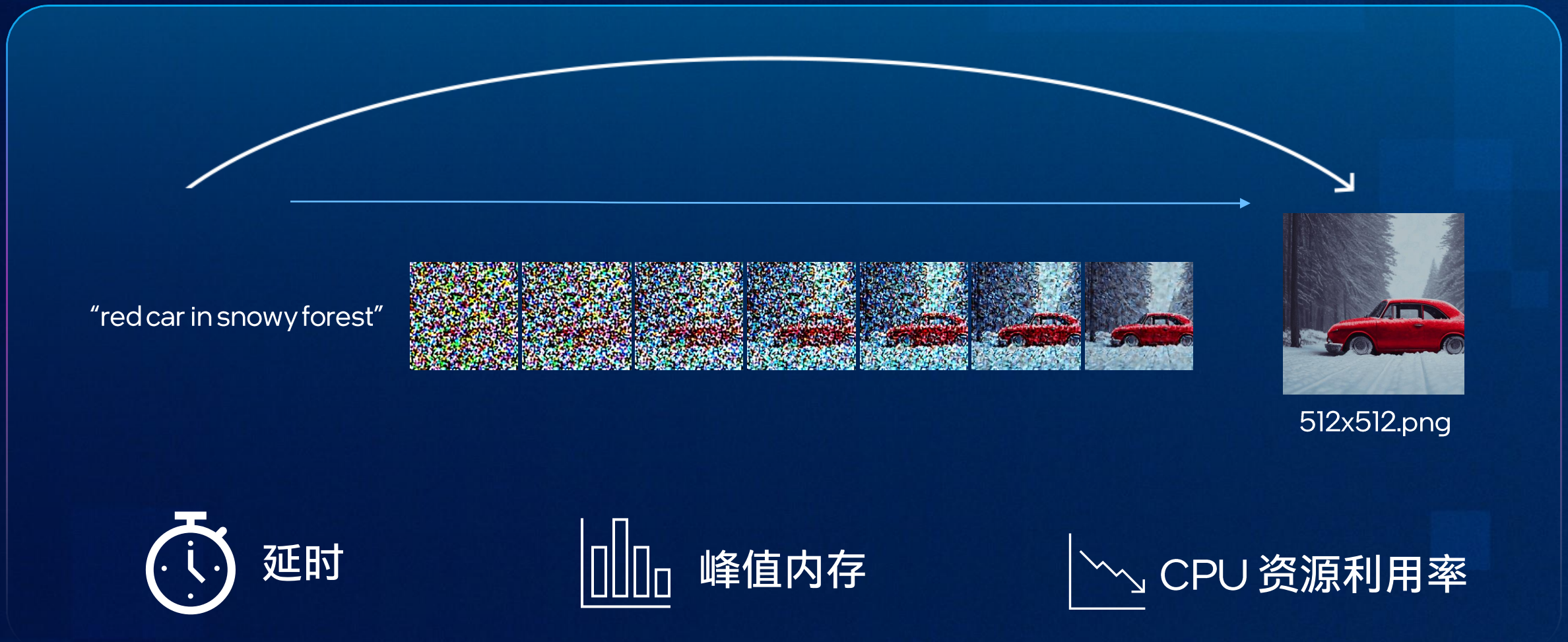
英特尔 AI 软件工程师

# 1.任务介绍

# 什么是文生图?

Prompt: “red car in snowy forest”

# 任务目标：在边缘设备上优化端到端性能



# 参赛流程

## 初赛

提交项目 Proposal，主办方将根据Proposal 中描述的方案，评估其可行性与预计效果，并以此筛选入围决赛队伍。

Proposal

## 决赛

向入围队伍统一发放指定硬件平台，选手须围绕该硬件平台进行方案开发，并展示最终成果。



Solution

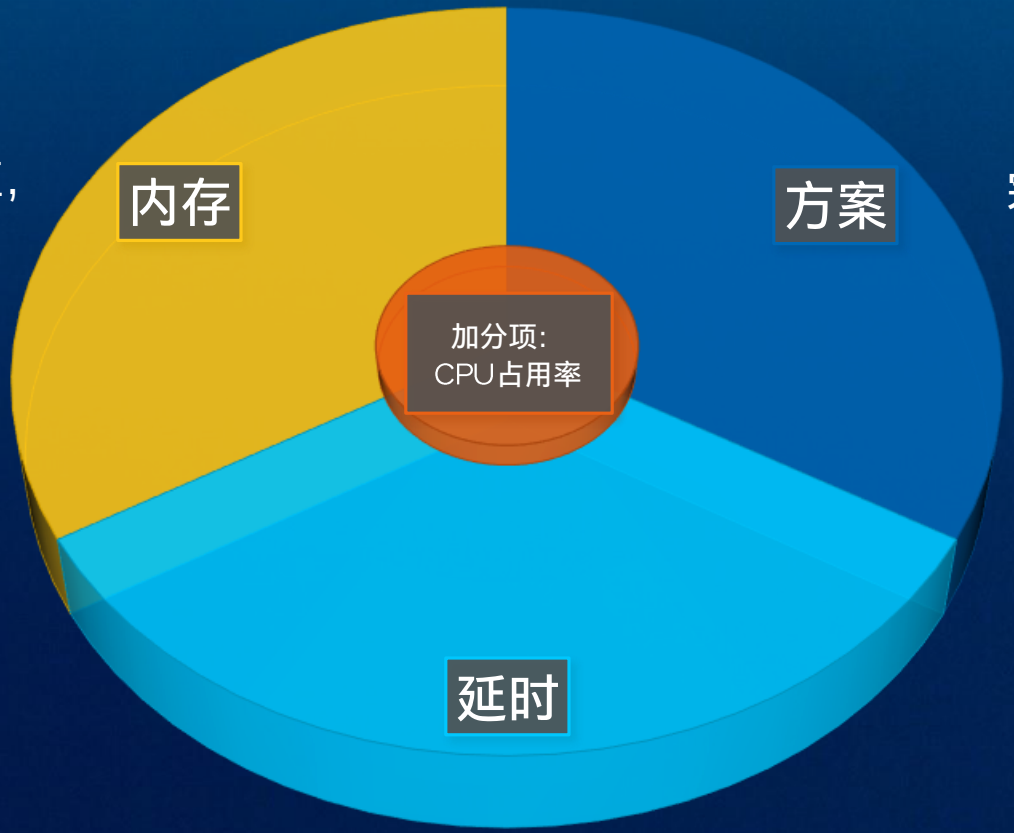
Codes

Dependency

https://github.com/openvinotoolkit/openvino/blob/master/docs/dev/build.md

# 评价标准

20次采样迭代后，内存占用峰值，越低越好




完整性，可行性与创新性

20次采样迭代后，端到端延迟（文本输入-RGB图像输出）越低越好

# 2.基础概念介绍

# OpenVINO™ 工具套件

## 1 模型

PyTorch | TensorFlow | TensorFlow Lite | PaddlePaddle | ONNX | Keras | Caffe | mxnet | KALDI

## 2 优化

OpenVINO™

性能优化

CPU: intel Atom | intel Core | intel Xeon | arm (NEW IN 2023)

GPU: intel Iris Xe Graphics | intel Arc Graphics | intel Data Center GPU Flex Series

NPU: intel Movidius

FPGA: intel FPGA AI Suite

## 3 部署

Windows | Linux | macOS

Powered by oneAPI

The productive, smart path to freedom for accelerated computing from the economic and technical burdens of proprietary alternatives.

# OpenVINO runtime API

```
compiled_model = core.compile_model(model=model, device_name="CPU")
```

```
core = ov.Core()
```

1. Create OpenVINO™ Runtime Core → 2. Compile Model → 3. Create Inference Request

```
infer_request = compiled_model.create_infer_request()
```

4. Set Inputs → 5. Start Inference → 6. Process Inference Results

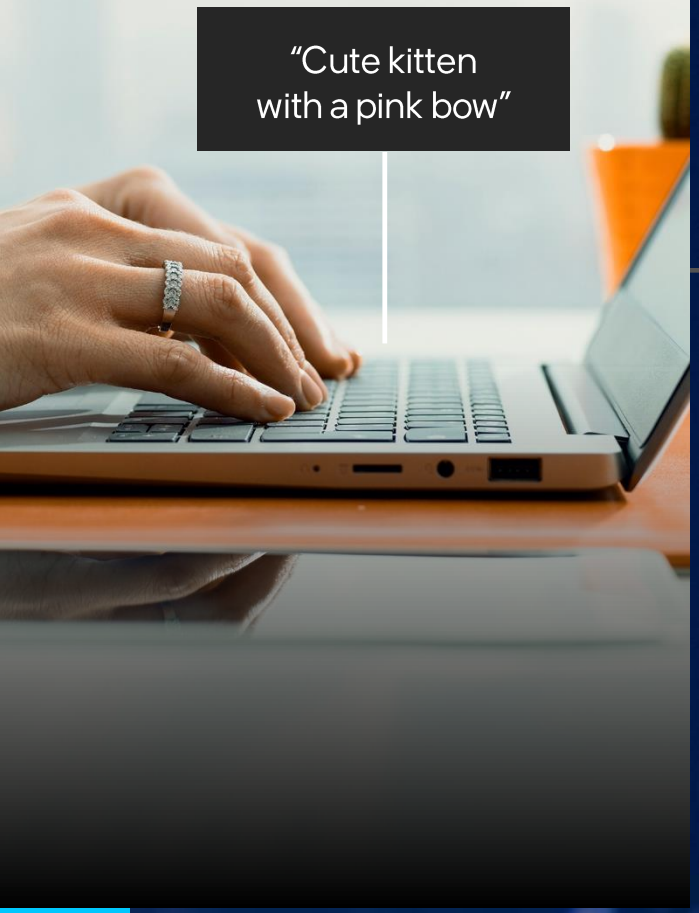
Inference Loop

```
infer_request.set_input_tensor(input_tensor)
```

```
infer_request.start_async()
infer_request.wait()
```

```
output = infer_request.get_output_tensor()
output_buffer = output.data
```

# Stable Diffusion



"Cute kitten with a pink bow"

Text Prompt Understanding

Text Encoder

Image Creation "Diffusion"

Unet+ Unet-

Image Decoder

VAE

Output: Image

Four Models
Multiple Iterations of Unet in Diffusion Stage

Text Prompt Understanding

Text Encoder

Image Creation "Diffusion"

Unet+ Unet-

Image Decoder

VAE

UNet Step 1

UNet Step 2

...

UNet Step 4

UNet Step 5

...

UNet Step 20

# 3.优化方向

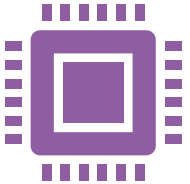
# OpenVINO™ Stable Diffusion

## 高性能推理

减少内存占用

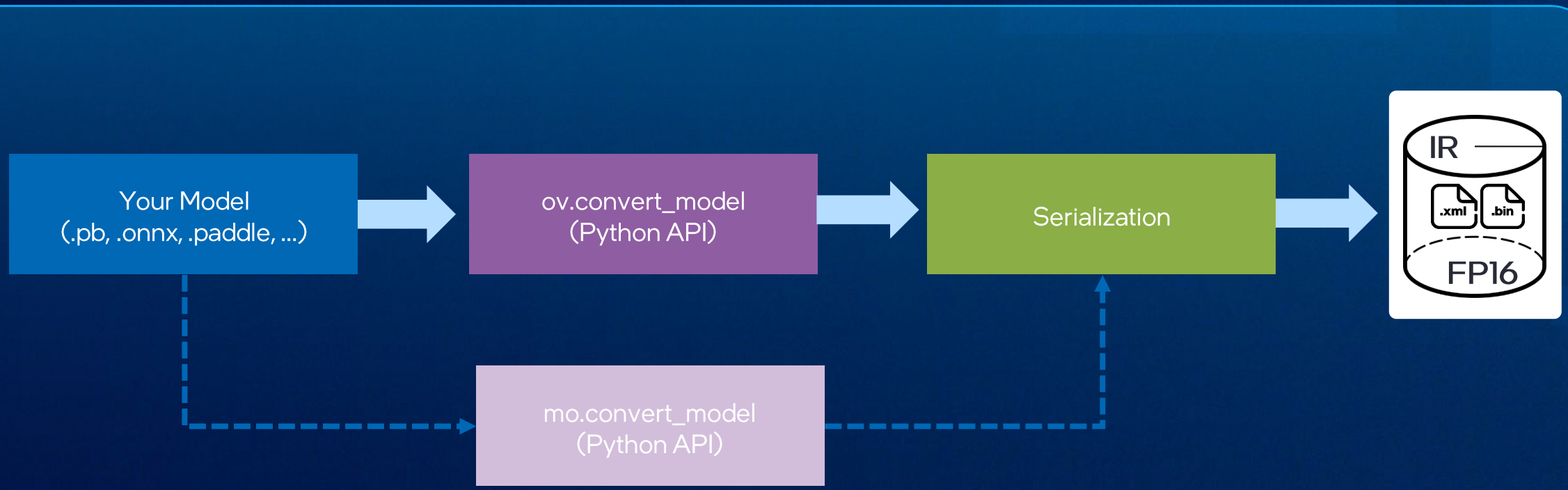
推理性能优化

流水线优化



灵活地在 CPU 和英特尔 GPU上运行工作负载

# 模型转换工具 API： ov.convert_model




```
ov_model = ov.convert_model("model.onnx", compress_to_fp16=True)

ov.save_model(ov_model, "converted_model.xml")
```

OpenVIN

```
8.0K    stabilityai_cpu/feature_extractor
1.3G    stabilityai_cpu/text_encoder
8.0K    stabilityai_cpu/scheduler
1.6M    stabilityai_cpu/tokenizer
3.3G    stabilityai_cpu/unet
[illegible]  stabilityai_cpu/vae
4.9G    stabilityai_cpu/
```

FP32 Native Model

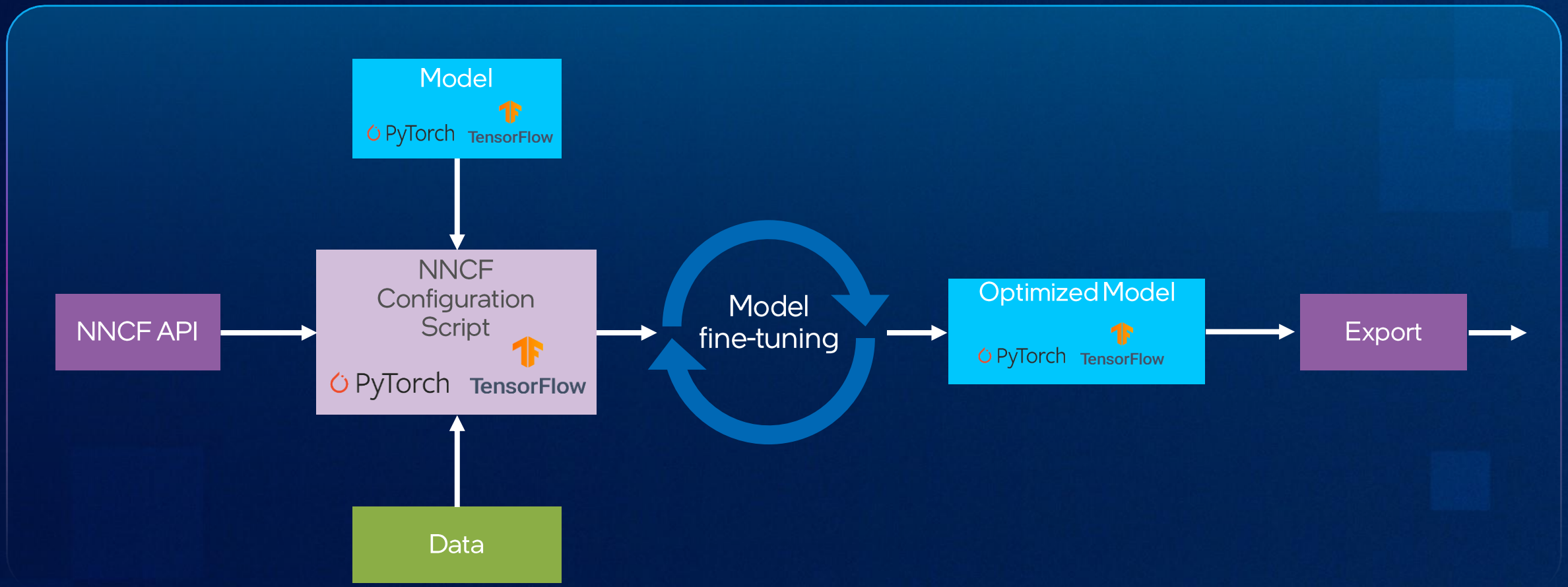
```
8.0K    openvino_ir/feature_extractor
1.3G    openvino_ir/text_encoder
131M    openvino_ir/vae_encoder
8.0K    openvino_ir/scheduler
1.6M    openvino_ir/tokenizer
3.3G    openvino_ir/unet
[illegible]  openvino_ir/vae_decoder
4.9G    openvino_ir/
```

FP32 OpenVINO Model

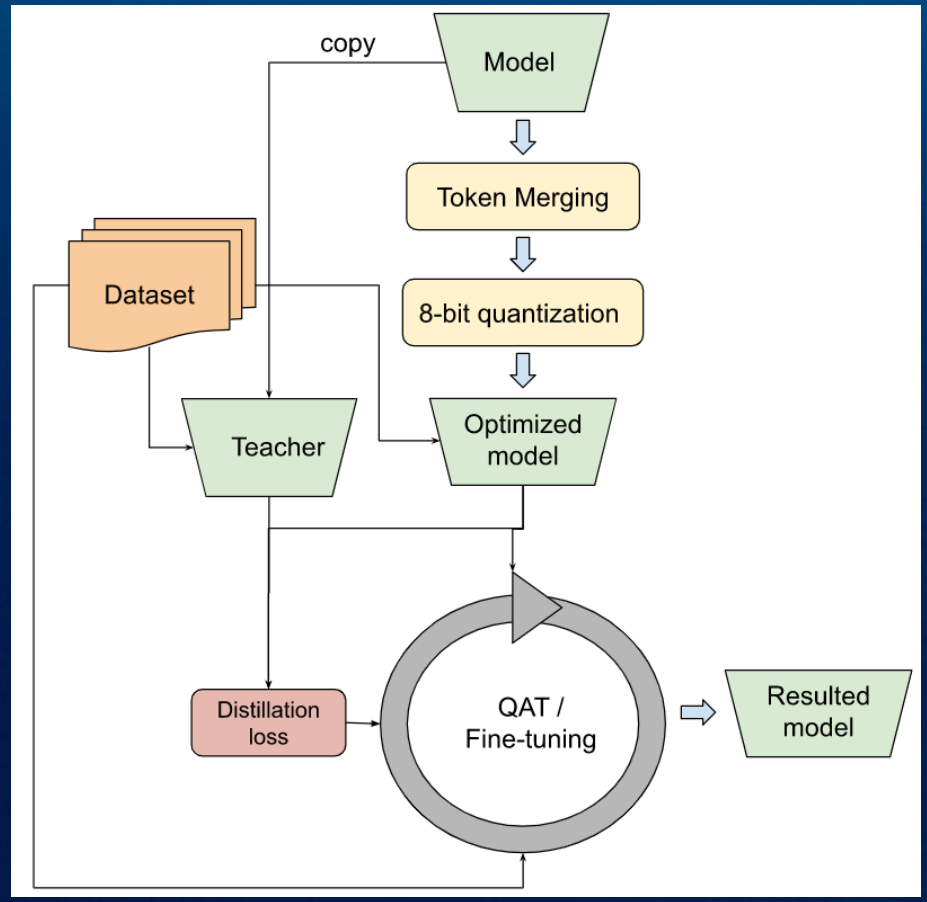
```
8.0K    modelSD21_dGPU_OV/feature_extractor
652M    modelSD21_dGPU_OV/text_encoder
8.0K    modelSD21_dGPU_OV/scheduler
1.6M    modelSD21_dGPU_OV/tokenizer
1.7G    modelSD21_dGPU_OV/unet
[illegible]  modelSD21_dGPU_OV/vae_decoder
2.4G    modelSD21_dGPU_OV/
```

FP16 OpenVINO model

# Quantization-Aware Training (NNCF)



# Quantization-Aware Training (NNCF) for Stable Diffusion



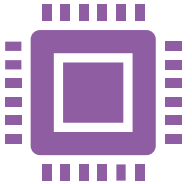
# OpenVINO™ Stable Diffusion

## 高性能推理

减少内存占用

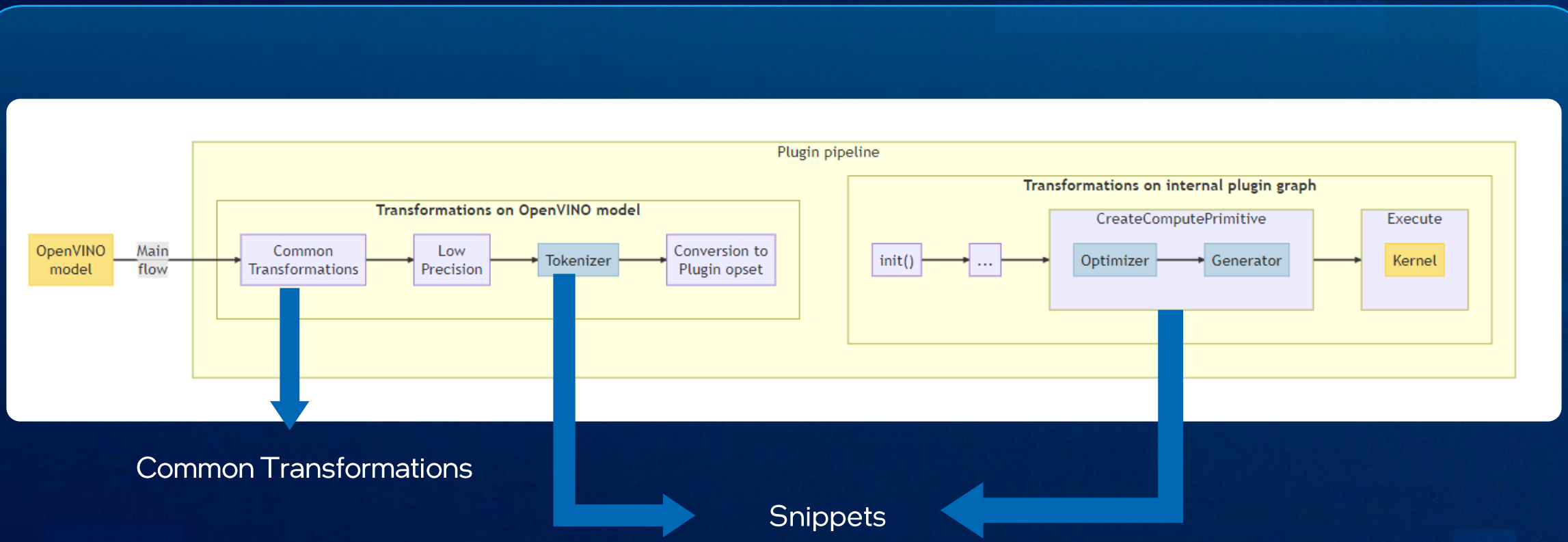
推理性能优化

流水线优化

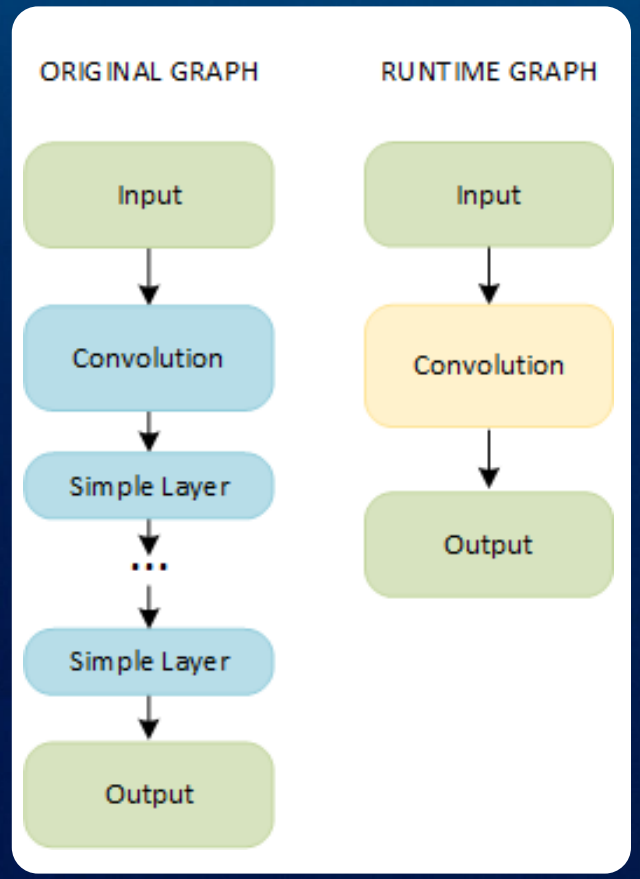


灵活地在 CPU 和英特尔 GPU上运行工作负载

# 运行时模型结构优化


Plugin pipeline
Transformations on OpenVINO model
Transformations on internal plugin graph
OpenVINO model
Main flow
Common Transformations
Low Precision
Tokenizer
Conversion to Plugin opset
init()
...
CreateComputePrimitive
Optimizer
Generator
Execute
Kernel
Common Transformations
Snippets

# 运行时模型结构优化: Common Transformations




1.创建模板 Pattern

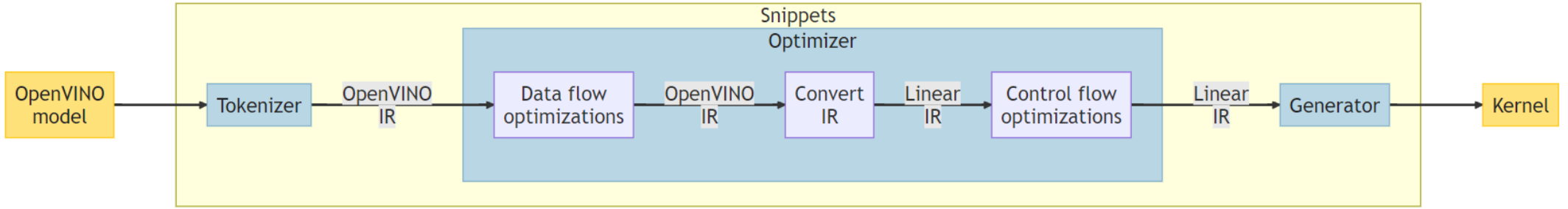
2.实现 Matcher callback

3.注册 Pattern and Matcher

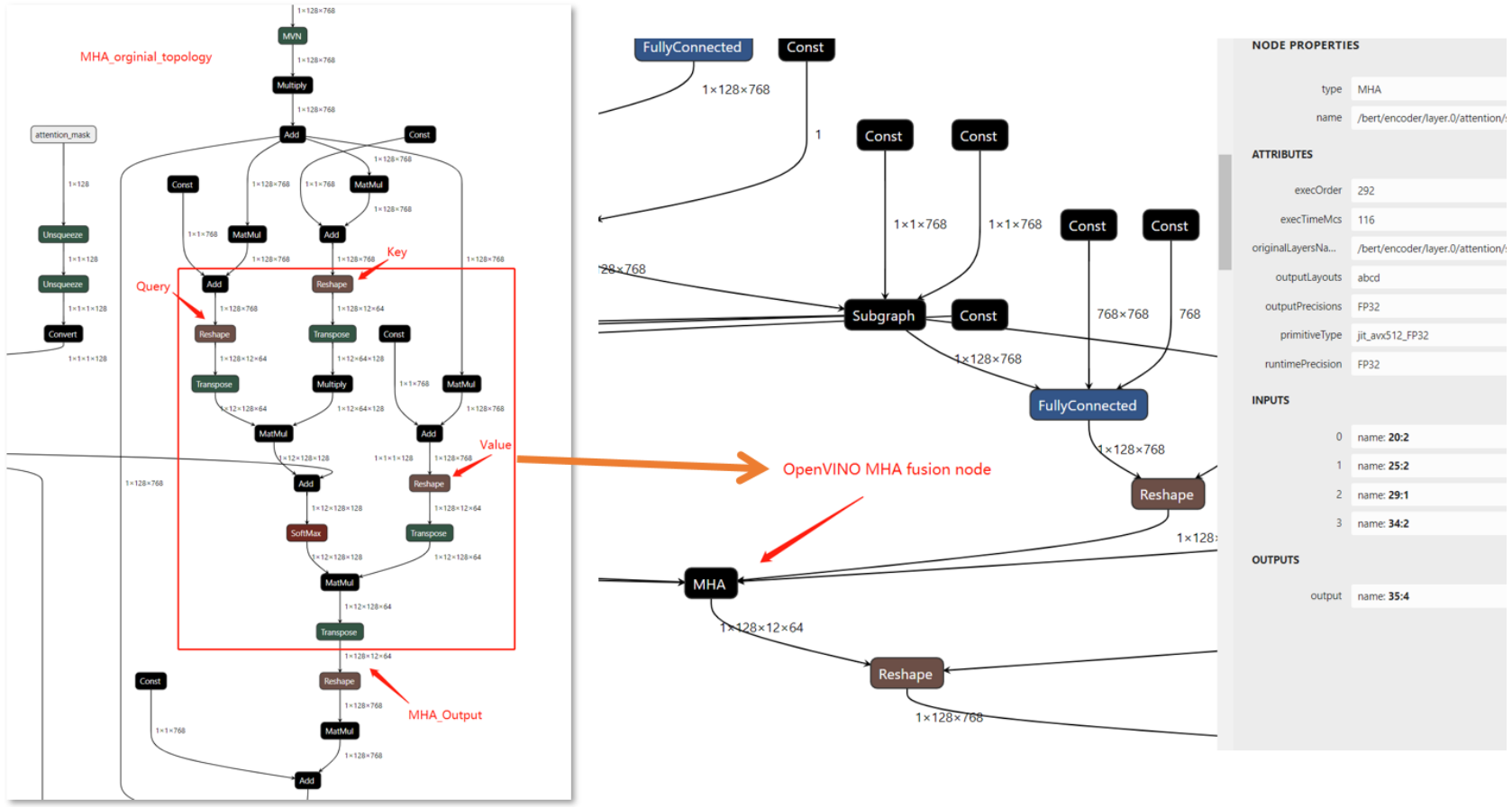
4.执行 MatcherPass

https://docs.openvino.ai/2023.1/openvino_docs_transformations.html

# 运行时模型结构优化: Snippets



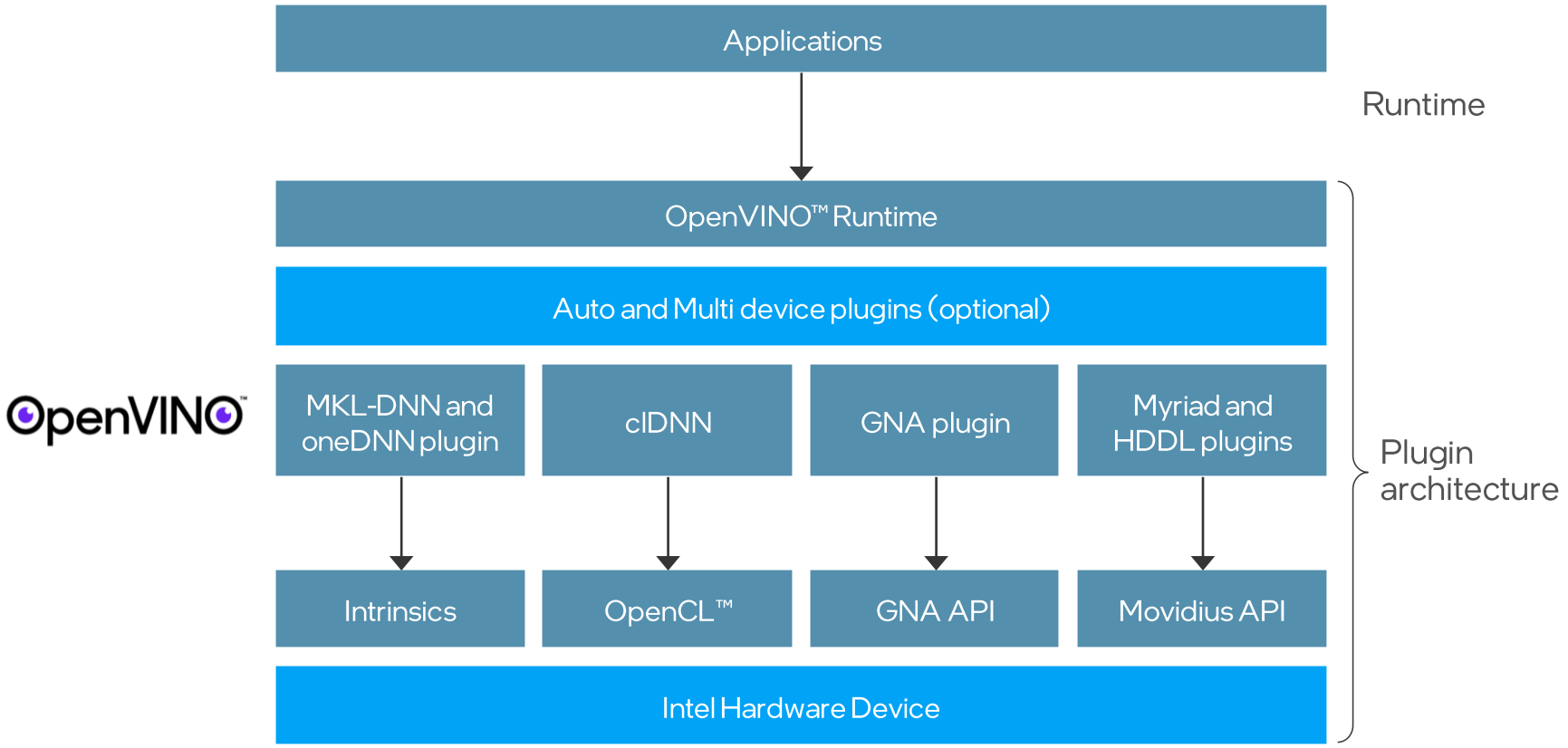
# 运行时模型结构优化: Snippets




https://blog.openvino.ai/blog-posts/openvino-tm-model-transformation-mha-subgraph-optimization

# 算子优化

```
compiled_model = core.compile_model(model=model, device_name="CPU")
```



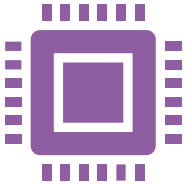
# OpenVINO™ Stable Diffusion

## 高性能推理

减少内存占用

推理性能优化

流水线优化



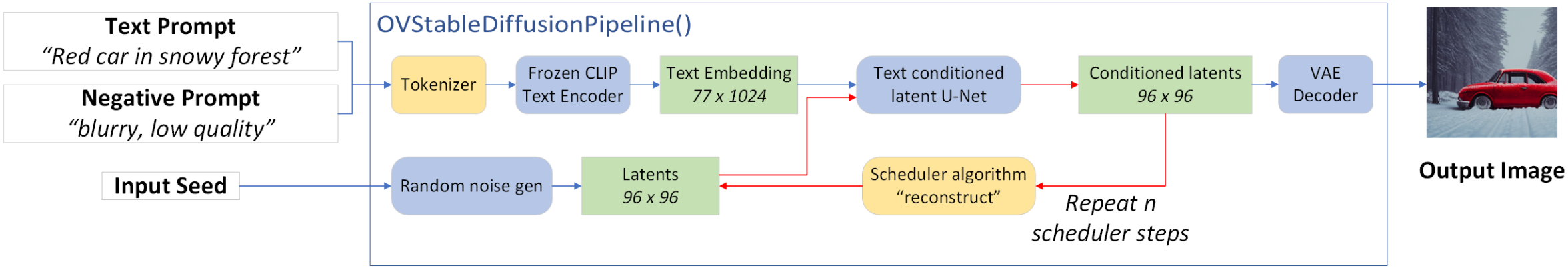
灵活地在 CPU 和英特尔 GPU上运行工作负载

# OpenVINO™ Stable Diffusion

## 高性能推理

### SD流水线

### 推理流程




基于C++的pipeline实现：https://blog.openvino.ai/blog-posts/cpp-pipeline-for-stable-diffusion-v1-5-with-pybind-for-lora-enabling

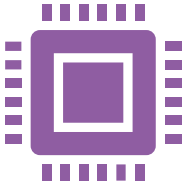
# OpenVINO™ Stable Diffusion

## 高性能推理

减少内存占用

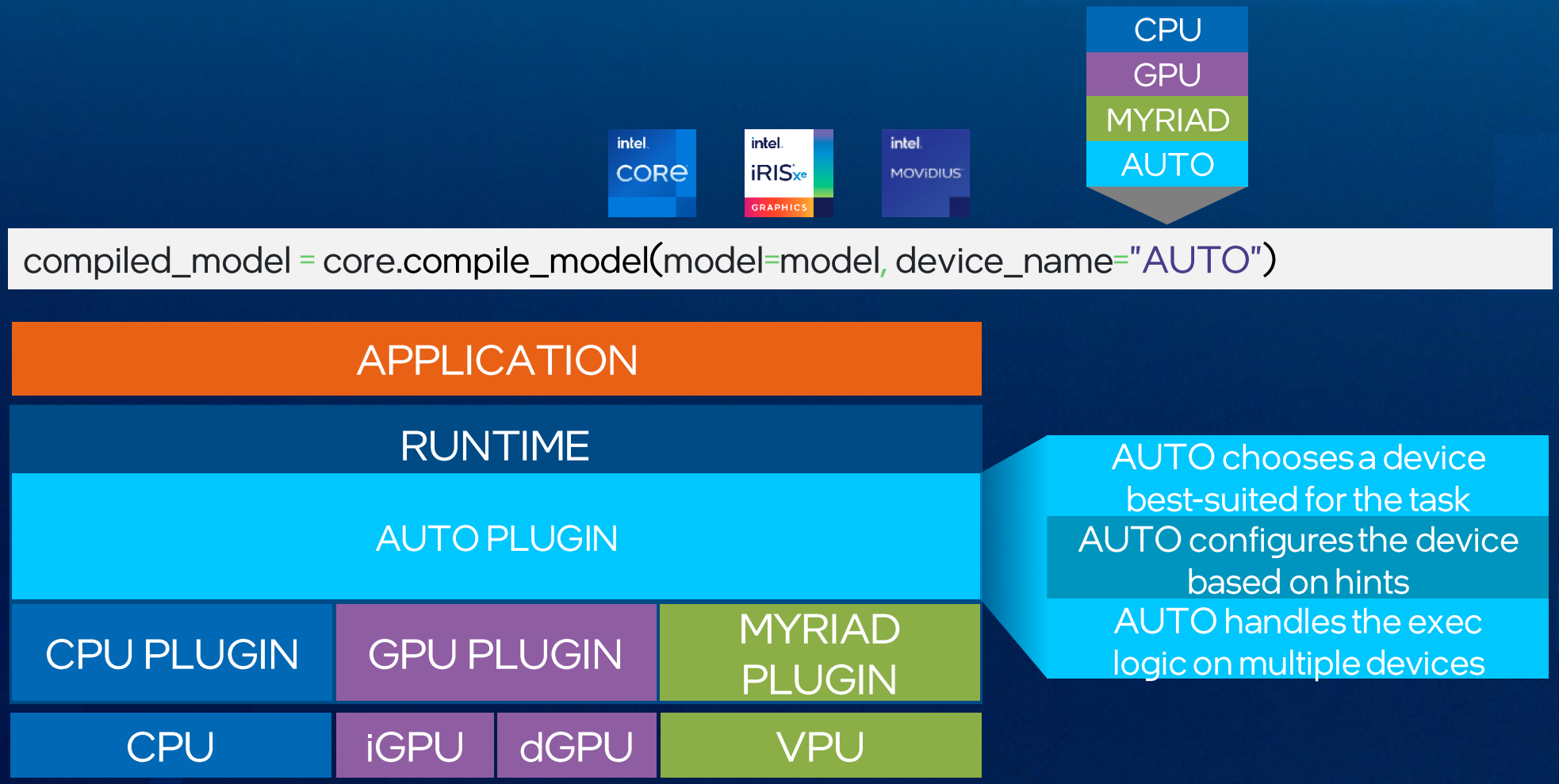
推理性能优化

流水线优化



灵活地在 CPU 和英特尔 GPU上运行工作负载

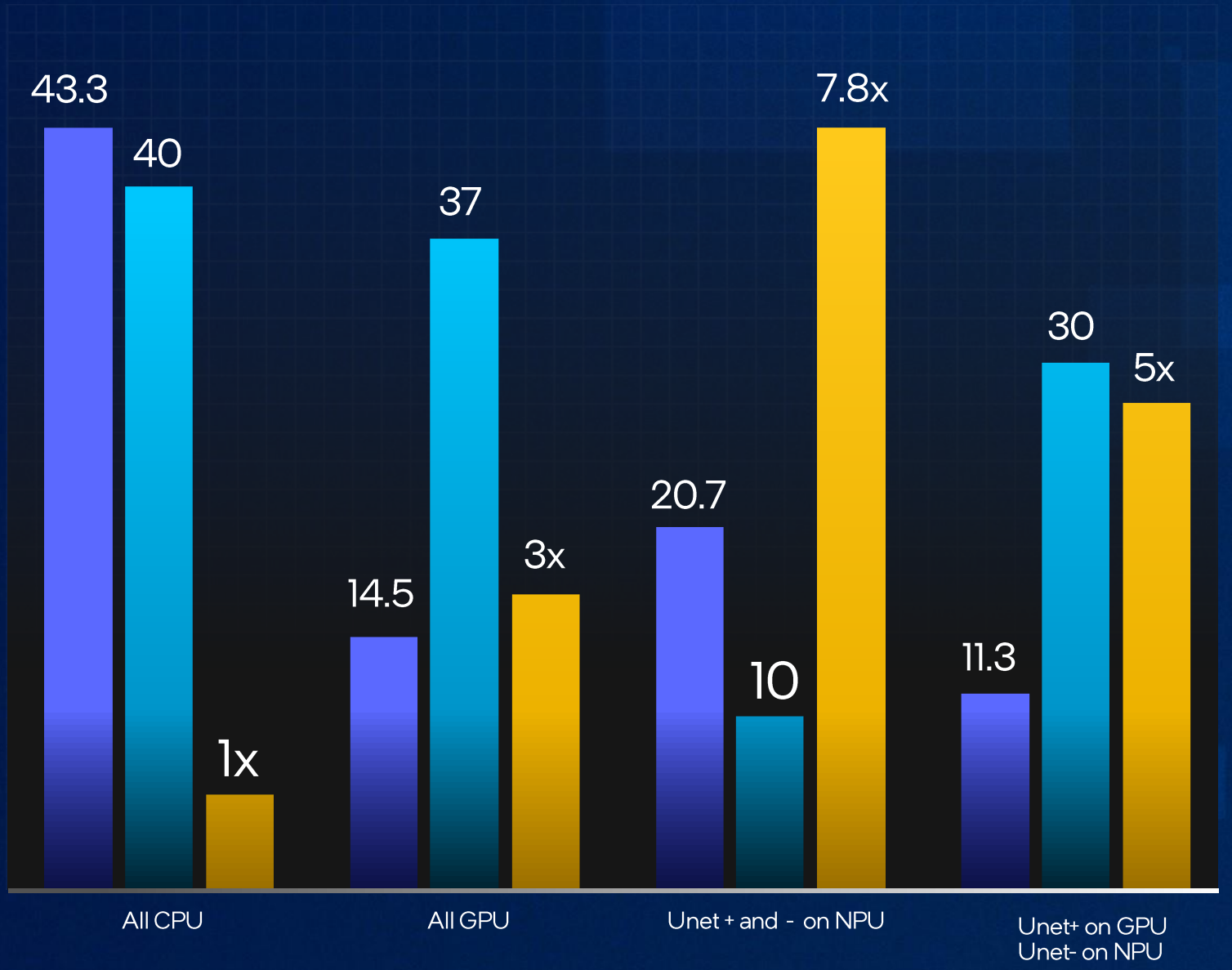
# 硬件支持


CPU
GPU
MYRIAD
AUTO
compiled_model = core.compile_model(model=model, device_name="AUTO")
APPLICATION
RUNTIME
AUTO PLUGIN
CPU PLUGIN
GPU PLUGIN
MYRIAD PLUGIN
CPU
iGPU
dGPU
VPU
AUTO chooses a device best-suited for the task
AUTO configures the device based on hints
AUTO handles the exec logic on multiple devices

# Stable Diffusion v1.5

20 Iterations

42 Inferences: Text Encoder (1) + Unet+ (20) + Unet– (20) + VAE Decoder (1)




Time: 20 Iterations (Sec)

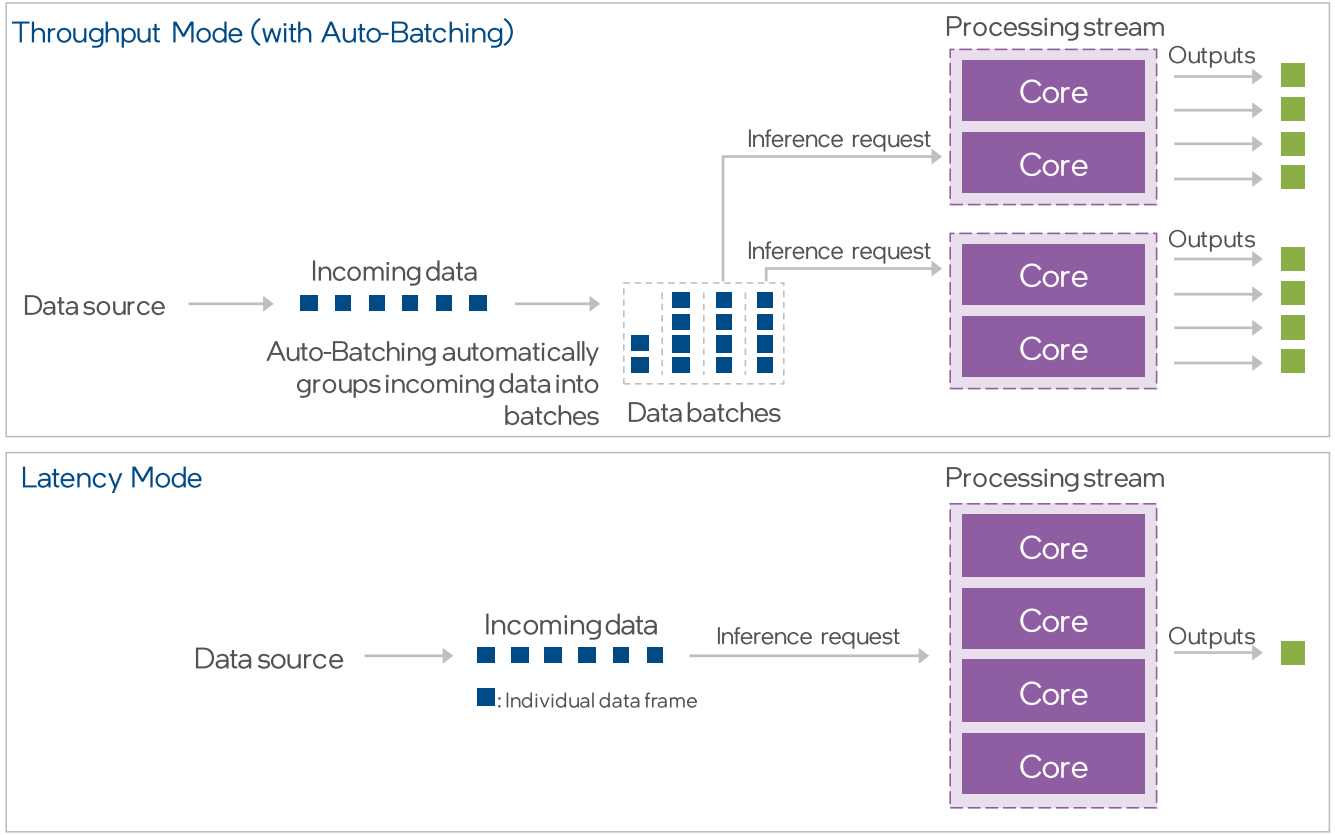
Power (W)

Efficiency (relative)

*Based on internal estimates. Learn more at www.intel.com/PerformanceIndex. Results may vary.

# Performance Hint

```
compiled_model = core.compile_model(model=model, device_name="AUTO",
config={"PERFORMANCE_HINT":"LATENCY"})
```

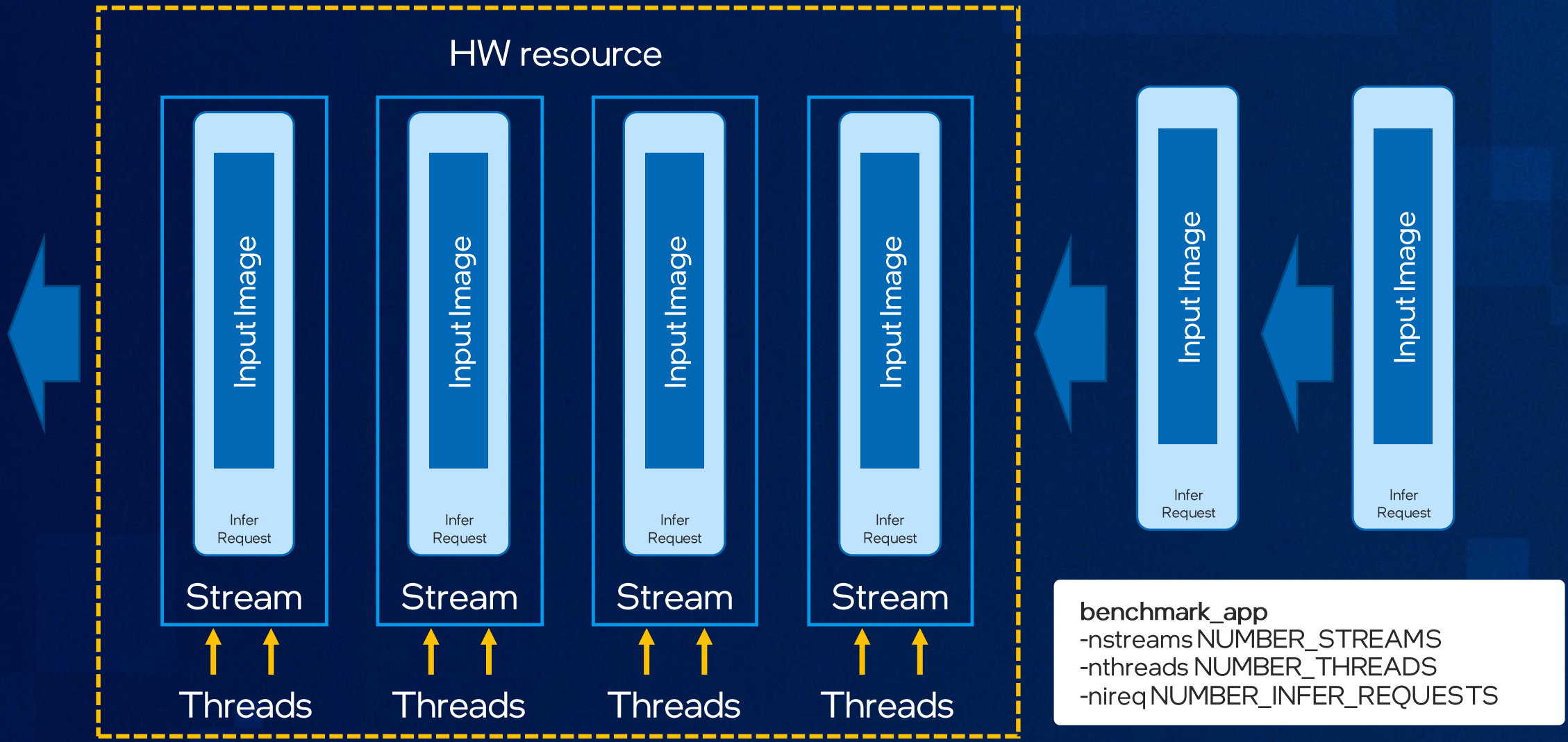


# 手动配置

```
import os
from openvino import runtime as ov

core = ov.Core()

# more inference threads
num_cores = os.cpu_count()
config = {"INFERENCE_NUM_THREADS": num_cores}

# number of inference requests
config = {"PERFORMANCE_HINT_NUM_REQUESTS": 2}

# number of executor logical partitions
config = {"NUM_STREAMS": 2}

ov_cpu_config_model = core.compile_model(ov_model, device_name="CPU", config=config)
```

# Thread, Stream, Infer Request,

self.core.set_property(device, config[device])



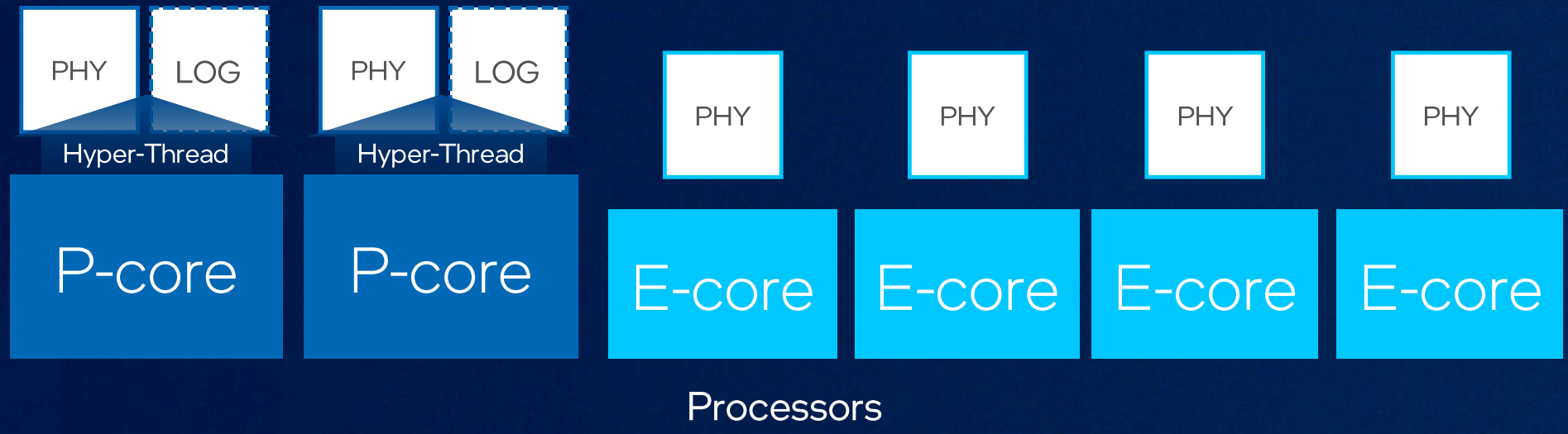

**benchmark_app**
-nstreams NUMBER_STREAMS
-nthreads NUMBER_THREADS
-nireq NUMBER_INFER_REQUESTS

# CPU 部署：性能核 or 能效核？

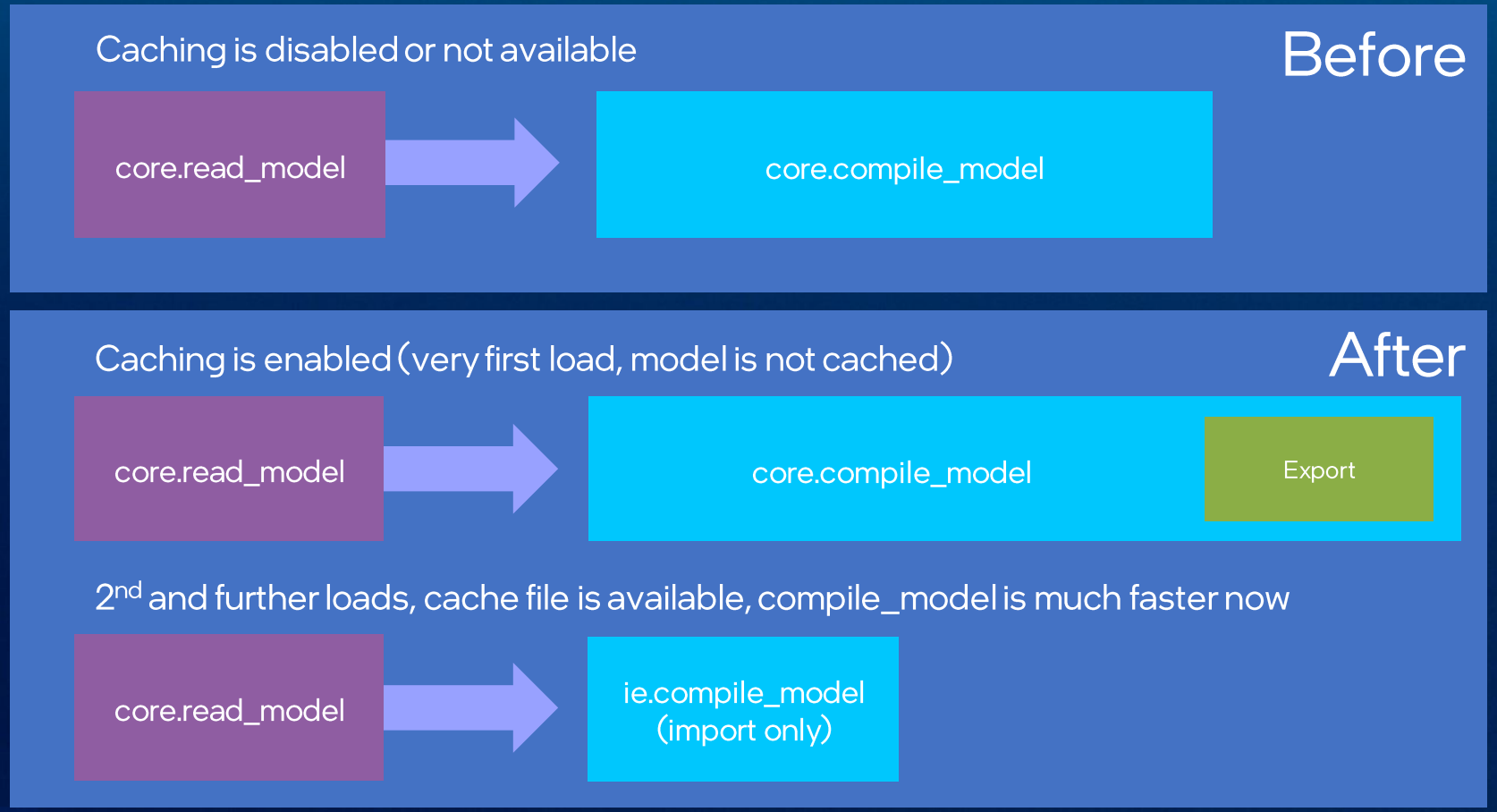
- Performance-core (P-core)
- Efficient-core (E-core)
- 12th Gen Intel® Core and up

选择在性能核、能效核还是所有核上运行推理

```
config["SCHEDULING_CORE_TYPE"] = "ANY_CORE" # ANY_CORE, PCORE_ONLY, ECORE_ONLY
```



# GPU 部署：模型缓存


Caching is disabled or not available
Before
core.read_model
core.compile_model
Caching is enabled (very first load, model is not cached)
After
core.read_model
core.compile_model
Export
2nd and further loads, cache file is available, compile_model is much faster now
core.read_model
ie.compile_model
(import only)

# 模型缓存

```
from openvino import runtime as ov

core = ov.Core()
core.set_property({"CACHE_DIR": "/path/to/cache/dir"})

model = core.read_model(model=xml_path)
compiled_model = core.compile_model(model=model, device_name=device_name)
```

# 模型性能测试工具 Benchmark_app

benchmark_app –d GPU –m MODEL_DIR -pc

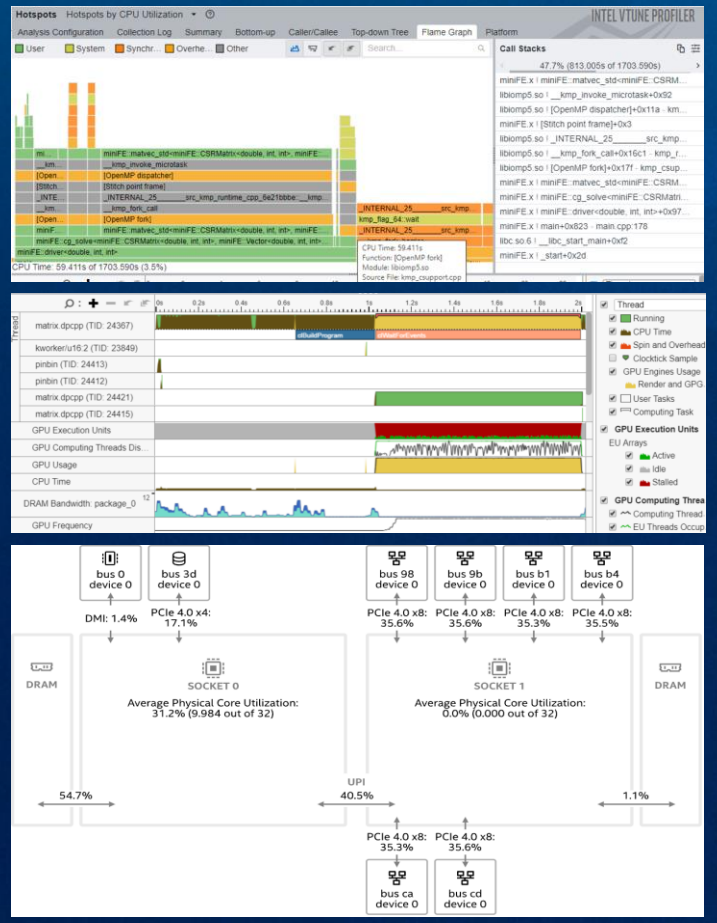
```
[ INFO ] Read model took 86.28 ms
[Step 5/11] Resizing network to match image sizes and given batch
[ INFO ] Network batch size: 1
[Step 6/11] Configuring input of the model
[ INFO ] Model input 'Func/StatefulPartitionedCall/input/_0:0' precision u8, dimensions ([N,H,W,C]): 1 224 224 3
[ INFO ] Model output 'StatefulPartitionedCall/MobilenetV3small/Predictions/Softmax' precision f32, dimensions ([...]): 1 1000
[Step 7/11] Loading the model to the device
[ INFO ] Compile model took 488.14 ms
[Step 8/11] Querying optimal runtime parameters
[ INFO ] DEVICE: CPU
[ INFO ]   AVAILABLE_DEVICES  , ['']
[ INFO ]   RANGE_FOR_ASYNC_INFER_REQUESTS  , (1, 1, 1)
[ INFO ]   RANGE_FOR_STREAMS  , (1, 8)
[ INFO ]   FULL_DEVICE_NAME  , Intel(R) Core(TM) i7-10610U CPU @ 1.80GHz
[ INFO ]   OPTIMIZATION_CAPABILITIES  , ['FP32', 'FP16', 'INT8', 'BIN', 'EXPORT_IMPORT']
[ INFO ]   CACHE_DIR  ,
[ INFO ]   NUM_STREAMS  , 4
[ INFO ]   AFFINITY  , Affinity.NONE
[ INFO ]   INFERENCE_NUM_THREADS  , 0
[ INFO ]   PERF_COUNT  , False
[ INFO ]   INFERENCE_PRECISION_HINT  , <Type: 'float32'>
[ INFO ]   PERFORMANCE_HINT  , PerformanceMode.THROUGHPUT
[ INFO ]   PERFORMANCE_HINT_NUM_REQUESTS  , 0
[Step 9/11] Creating infer requests and preparing input data
[ INFO ] Create 4 infer requests took 1.00 ms
[ WARNING ] No input files were given for input 'Func/StatefulPartitionedCall/input/_0:0'!. This input will be filled with random values!
[ INFO ] Fill input 'Func/StatefulPartitionedCall/input/_0:0' with random values
[Step 10/11] Measuring performance (Start inference asynchronously, 4 inference requests using 4 streams for CPU, inference only: True, limits: 60000 ms duration)
[ INFO ] Benchmarking in inference only mode (inputs filling are not included in measurement loop).
[ INFO ] First inference took 3.65 ms
[Step 11/11] Dumping statistics report
Count:          47704 iterations
Duration:       60009.09 ms
Latency:
   Median:      4.85 ms
   AVG:         4.97 ms
   MIN:         2.56 ms
   MAX:         64.13 ms
Throughput: 794.95 FPS
```

```
Add_131                 Status.NOT_RUN      layerType: Add              execType: undef                          realTime (ms): 0.000     cpuTime (ms): 0.000
Multiply_9074           Status.EXECUTED     layerType: Convolution      execType: brgconv_avx512_1_1_FP32 realTime (ms): 0.088     cpuTime (ms): 0.088
Clip_134                Status.NOT_RUN      layerType: Clamp            execType: undef                          realTime (ms): 0.000     cpuTime (ms): 0.000
Multiply_9081           Status.EXECUTED     layerType: GroupConvolution execType: jit_avx512_dw_FP32 realTime (ms): 0.029     cpuTime (ms): 0.029
Clip_137                Status.NOT_RUN      layerType: Clamp            execType: undef                          realTime (ms): 0.000     cpuTime (ms): 0.000
Multiply_9088           Status.EXECUTED     layerType: Convolution      execType: brgconv_avx512_1_1_FP32 realTime (ms): 0.091     cpuTime (ms): 0.091
Add_140                 Status.NOT_RUN      layerType: Add              execType: undef                          realTime (ms): 0.000     cpuTime (ms): 0.000
Multiply_9095           Status.EXECUTED     layerType: Convolution      execType: brgconv_avx512_1_1_FP32 realTime (ms): 0.085     cpuTime (ms): 0.085
Clip_143                Status.NOT_RUN      layerType: Clamp            execType: undef                          realTime (ms): 0.000     cpuTime (ms): 0.000
Multiply_9102           Status.EXECUTED     layerType: GroupConvolution execType: jit_avx512_dw_FP32 realTime (ms): 0.032     cpuTime (ms): 0.032
Clip_146                Status.NOT_RUN      layerType: Clamp            execType: undef                          realTime (ms): 0.000     cpuTime (ms): 0.000
Multiply_9109           Status.EXECUTED     layerType: Convolution      execType: brgconv_avx512_1_1_FP32 realTime (ms): 0.160     cpuTime (ms): 0.160
Multiply_9139           Status.EXECUTED     layerType: GroupConvolution execType: jit_avx512_dw_FP32 realTime (ms): 0.018     cpuTime (ms): 0.018
Relu_159                Status.NOT_RUN      layerType: Relu             execType: undef                          realTime (ms): 0.000     cpuTime (ms): 0.000
Conv_160/Without...     Status.EXECUTED     layerType: Convolution      execType: brgconv_avx512_1_1_FP32 realTime (ms): 0.012     cpuTime (ms): 0.012
Conv_160/Without...     Status.NOT_RUN      layerType: Reorder          execType: reorder_FP32                   realTime (ms): 0.000     cpuTime (ms): 0.000
Reshape_181             Status.NOT_RUN      layerType: Reshape          execType: unknown_FP32                   realTime (ms): 0.000     cpuTime (ms): 0.000
Concat_182              Status.NOT_RUN      layerType: Concat           execType: unknown_FP32                   realTime (ms): 0.000     cpuTime (ms): 0.000
Reshape_1715            Status.NOT_RUN      layerType: Reshape          execType: unknown_FP32                   realTime (ms): 0.000     cpuTime (ms): 0.000
Softmax_1746            Status.EXECUTED     layerType: Softmax          execType: jit_avx512_FP32                realTime (ms): 0.035     cpuTime (ms): 0.035
Reshape_195             Status.NOT_RUN      layerType: Reshape          execType: unknown_FP32                   realTime (ms): 0.000     cpuTime (ms): 0.000
Multiply_9119           Status.EXECUTED     layerType: GroupConvolution execType: jit_avx512_dw_FP32 realTime (ms): 0.010     cpuTime (ms): 0.010
Relu_163                Status.NOT_RUN      layerType: Relu             execType: undef                          realTime (ms): 0.000     cpuTime (ms): 0.000
Conv_164/Without...     Status.EXECUTED     layerType: Convolution      execType: brgconv_avx512_1_1_FP32 realTime (ms): 0.016     cpuTime (ms): 0.016
Conv_164/Without...     Status.NOT_RUN      layerType: Reorder          execType: reorder_FP32                   realTime (ms): 0.000     cpuTime (ms): 0.000
Reshape_209             Status.NOT_RUN      layerType: Reshape          execType: unknown_FP32                   realTime (ms): 0.000     cpuTime (ms): 0.000
Multiply_9025           Status.EXECUTED     layerType: GroupConvolution execType: jit_avx512_dw_FP32 realTime (ms): 0.014     cpuTime (ms): 0.014
Relu_155                Status.NOT_RUN      layerType: Relu             execType: undef                          realTime (ms): 0.000     cpuTime (ms): 0.000
Conv_156/Without...     Status.EXECUTED     layerType: Convolution      execType: brgconv_avx512_1_1_FP32 realTime (ms): 0.012     cpuTime (ms): 0.012
```

pip install openvino-dev

# 程序性能优化与监测工具 Intel® VTune™ Profiler

- 快速找到性能瓶颈
  - A suite of profiling for CPU, GPU, FPGA, threading, memory, cache, storage, offload, power...
  - Application or system-wide analysis
  - DPC++, C, C++, Fortran, Python*, Go*, Java*, or a mix
  - Linux, Windows, FreeBSD, Android, Yocto and more
  - Containers and VMs




https://www.intel.cn/content/www/cn/zh/developer/tools/oneapi/vtune-profiler-download.html

# Baseline 示例

https://github.com/openvinotoolkit/openvino_notebooks/blob/main/notebooks/236-stable-diffusion-v2/236-stable-diffusion-v2-text-to-image.ipynb