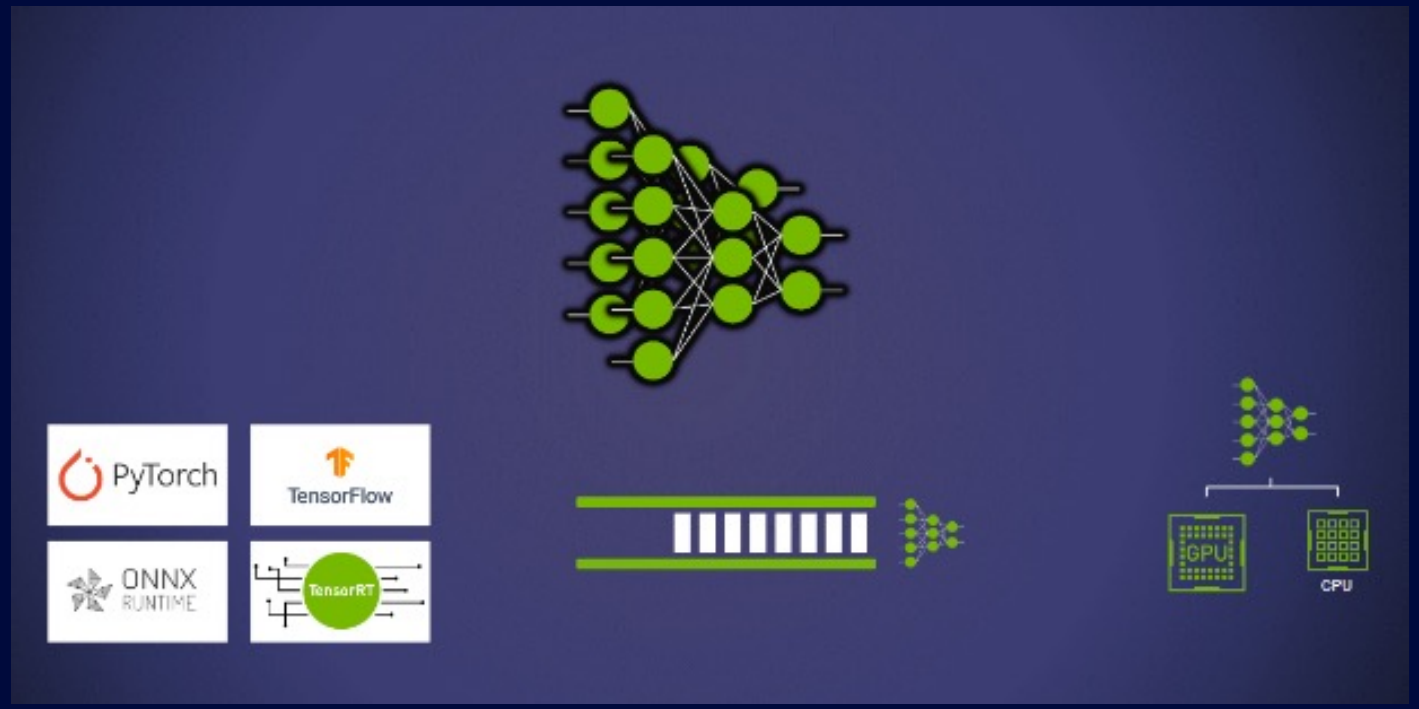
ASO LAB Seminar
3
Triton Server

엄소은

CONTENTS

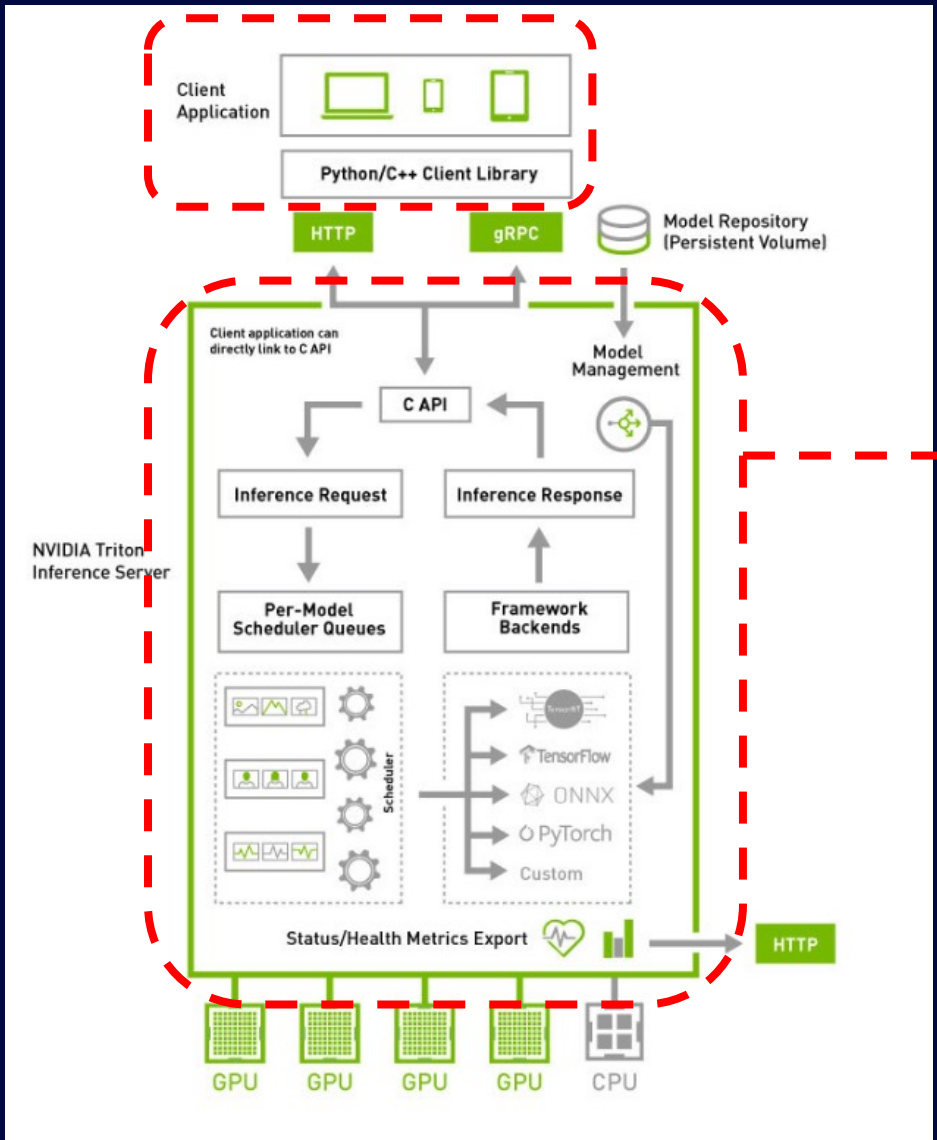
1. Triton Architecture

Triton Server



- Deploy pytorch, tensorflow, onnx model at the same server
- Different loads for the models
- Run on different hardware devices
- Independently manage serving configuration

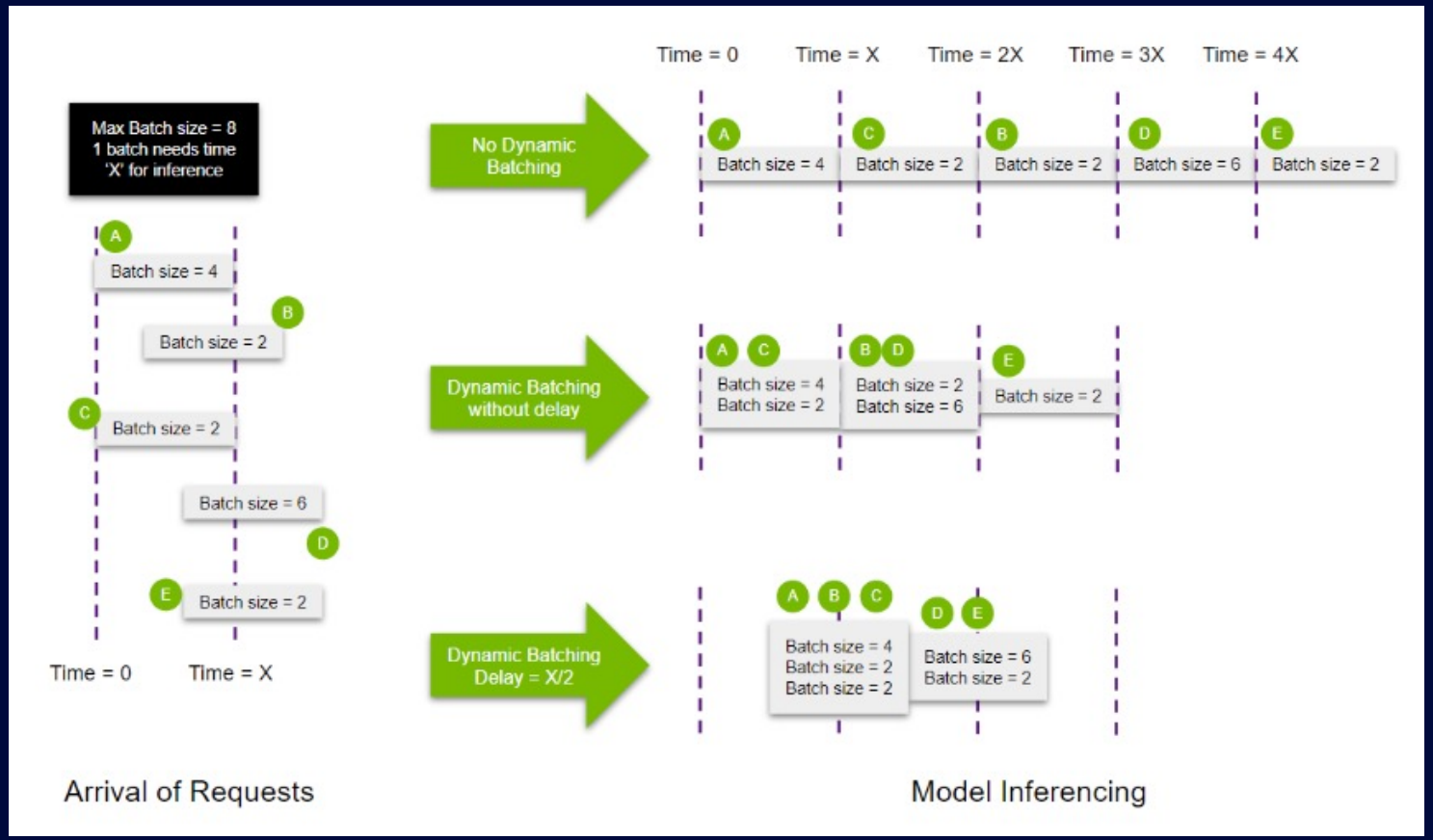
1. Triton Architecture



- Work as backend server
- Scheduling & batching algorithms

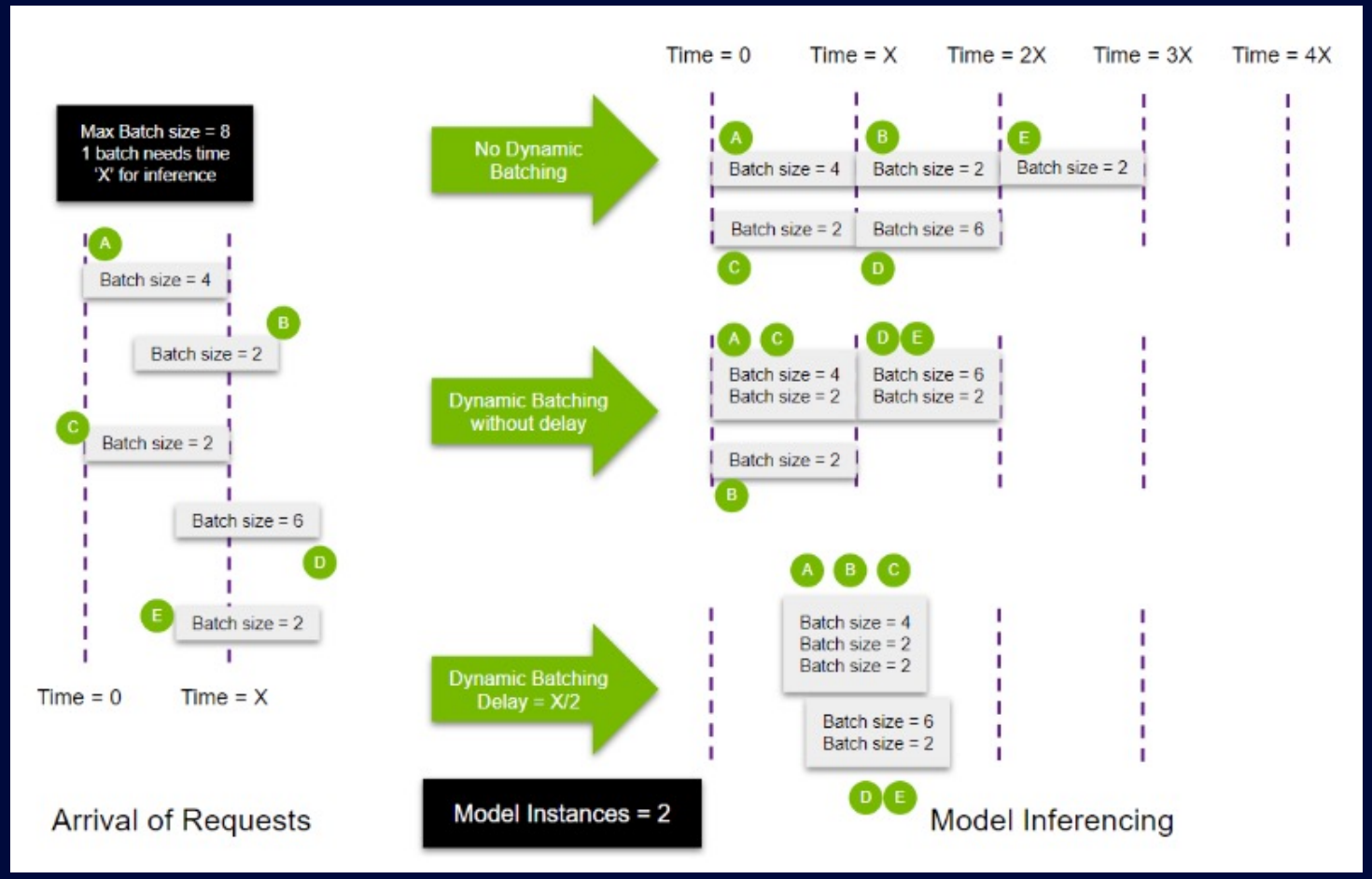
2. Optimization Techniques

Dynamic Batching



2. Optimization Techniques

Concurrent Model Execution



2. Optimization Techniques

Latency Analyzing

Model : OpenCV's EAST model

```
Client:
  Request count: 11240
  Throughput: 624.115 infer/sec
  p50 latency: 25614 usec
  p90 latency: 26465 usec
  p95 latency: 26521 usec
  p99 latency: 26582 usec
  Avg HTTP time: 25613 usec (send/recv 133 usec + response wait 25480 usec)
Server:
  Inference count: 11240
  Execution count: 11240
  Successful request count: 11240
  Avg request latency: 25022 usec (overhead 21 usec + queue 21834 usec + compute input 18 usec + compute infer 3137 usec + compute output 11 usec)
```

- Latency = Client + Server latency
- Client : send/recv(133 usec) + response wait(25400 usec)
- Server : overhead(21 usec) + queue(21834 usec) + compute input(18usec) + compute infer(3137 usec) + compute outut (11 usec)

2. Optimization Techniques

Latency Analyzing

Three major contributors of latency:

- Network Latency
  - Case by case...
- Inference Compute time
  - Solved by optimizing the network graphs by fusing layers, reducing models precision, fusing kernels
- Latency caused due to wait time in the model's queue
  - Solved by adding more instances of the model

2. Optimization Techniques

Experiment using Perf Analyzer

Model : OpenCV's EAST model

- P95 Batch Throughput for concurrency = 10

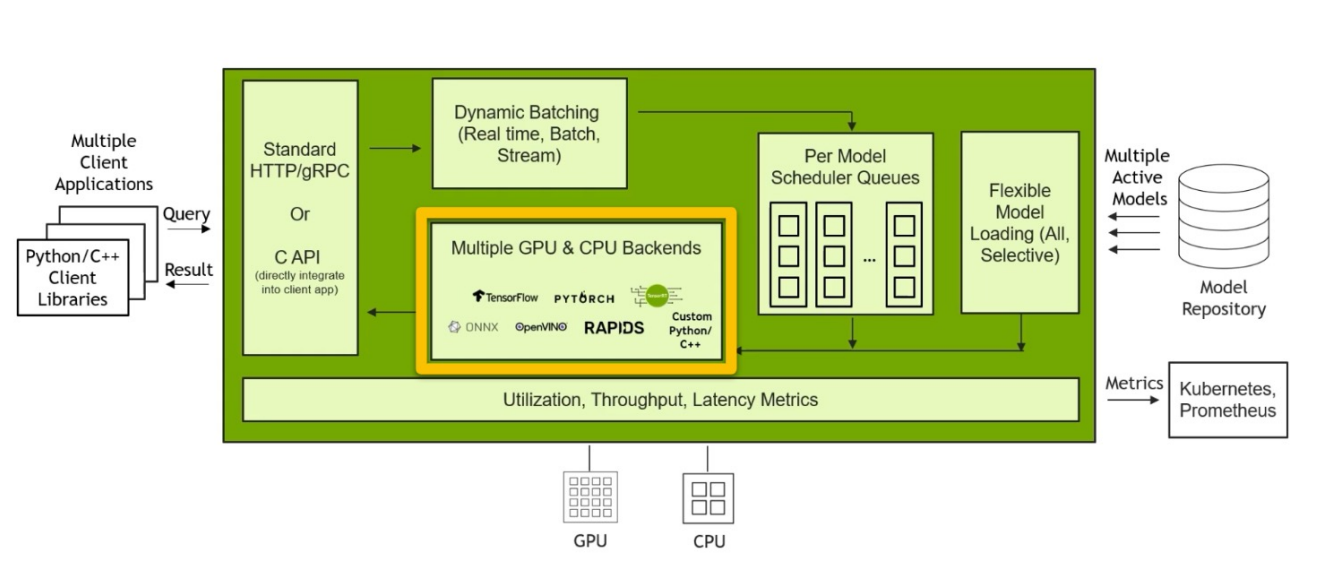
Model Instance / Dynamic Batching	1	2
X	212.849 infer/sec	398.294 infer/sec
O	408.811 infer/sec	400.537 infer/sec

- P95 Batch Latency for concurrency = 10

Model Instance / Dynamic Batching	1	2
X	95259 usec	57832 usec
O	59648 usec	54306 usec

2. Optimization Techniques

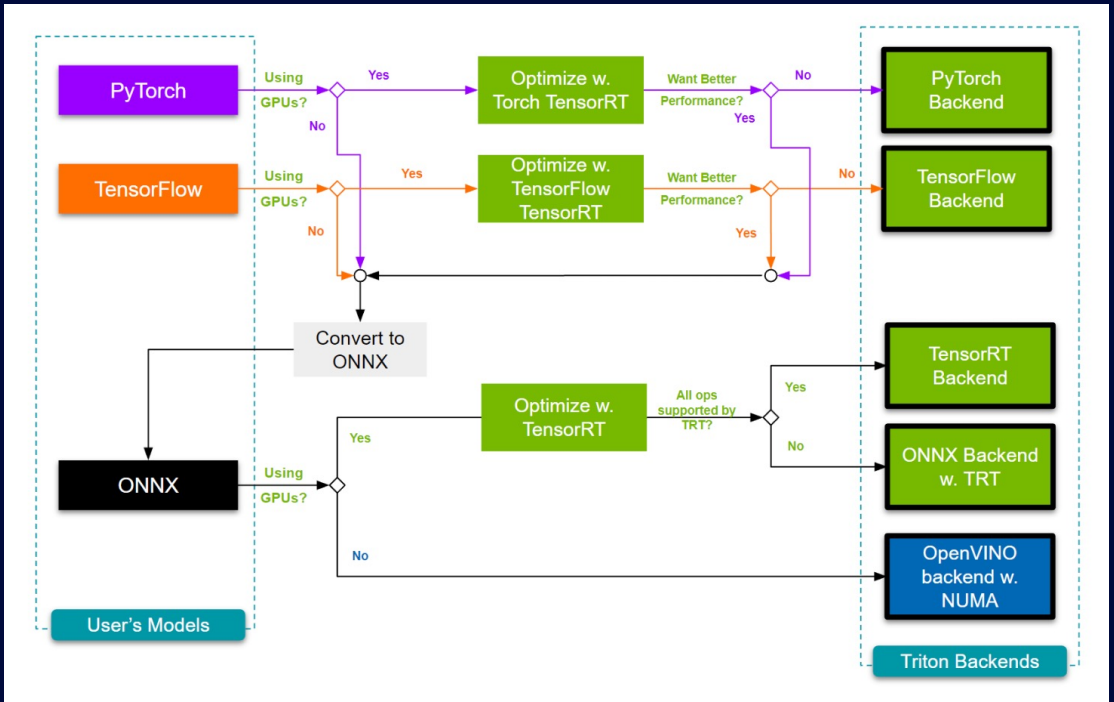
Accelerating Inference



- Triton Inference Server = "Backend" that executes the model
- Acceleration Recommendations
  - Type of Hardware : GPU / CPU
  - Type of model : shallow model (e.g. Random Forests), Neural Networks(e.g. BERT, CNN), Large Transformer Models

2. Optimization Techniques

Accelerating Inference



- Acceleration Recommendations
  - Type of Hardware : GPU / CPU
  - Type of model : shallow model (e.g. Random Forests), Neural Networks(e.g. BERT, CNN), Large Transformer Models

2. Optimization Techniques

Accelerating Inference (for onnx model)

1. With TensorRT and CUDA execution providers for GPU
2. With OpenVINO for CPU.

2. Optimization Techniques

Accelerating Inference (no dynamic batching , instance = 1)

1. ONNX RT execution on GPU w. CUDA execution provide
   - Concurrency : 10, throughput : 738.091 infer/sec, latency :13540 usec

```
parameters { key: "cudnn_conv_algo_search" value: { string_value: "0" } }
parameters { key: "gpu_mem_limit" value: { string_value: "4294967200" } }
```

```
Inferences/Second vs. Client Average Batch Latency
Concurrency: 10, throughput: 738.091 infer/sec, latency 13540 usec
```

2. ONNX RT execution on CPU w. OpenVINO acceleration
   - Concurrency : 10, throughput : 675.62 infer/sec, latency : 14797 usec

```
optimization { execution_accelerators {
  cpu_execution_accelerator : [ {
    name : "openvino"
  } ]
}}
```

```
Inferences/Second vs. Client Average Batch Latency
Concurrency: 10, throughput: 675.62 infer/sec, latency 14797 usec
```

3. ONNX RT execution on GPU w. TRT acceleration
   - Concurrency : 10, throughput : 2772.06 infer/sec, latency 3604 usec

   (TensorRT doesn't natively support INT64, so was casted down to INT32)

```
optimization {
  graph : {
    level : 1
  }
 execution_accelerators {
    gpu_execution_accelerator : [ {
      name : "tensorrt",
      parameters { key: "precision_mode" value: "FP16" },
      parameters { key: "max_workspace_size_bytes" value: "1073741824" }
    }]
  }
}
```

```
Inferences/Second vs. Client Average Batch Latency
Concurrency: 10, throughput: 2772.06 infer/sec, latency 3604 usec
```

3. Plans

To-Do

1. config 를 바꾸었을 때 실제 server 코드에서 어떤 일이 일어나는지 파악해보기
2. 실험에서 throughput, latency 만 봤었는데, GPU/CPU 사용량도 함께 관찰하기
3. client side latency 분석하기