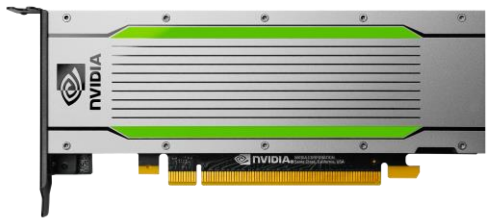
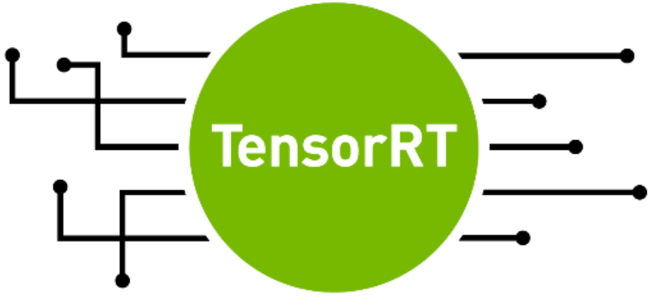
MAXIMIZING UTILIZATION FOR DATA CENTER INFERENCE WITH TENSORRT INFERENCE SERVER

정소영 상무 (soyoungj@nvidia.com) / 2019년 7월 2일

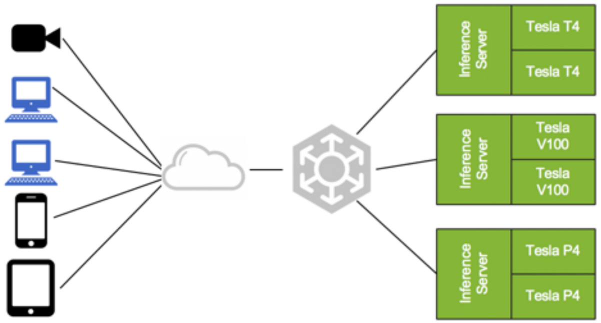
TENSORRT HYPERSCALE INFERENCE PLATFORM



WORLD'S MOST ADVANCED
SCALE-OUT GPU



INTEGRATED INTO TENSORFLOW &
ONNX SUPPORT



TENSORRT
INFERENCE SERVER

TENSORRT INFERENCE SERVER

A Software Application for Deploying AI Models At Scale

- Maximum GPU Utilization
- Mechanisms for Large-Scale Inference Service
- Optimized for Management & Monitoring
- GitHub: https://github.com/NVIDIA/tensorrt-inference-server

TENSORRT INFERENCE SERVER

Architected for Maximum Datacenter Utilization

Support a variety of model frameworks

TensorRT, TensorFlow, Caffe2, custom

Support concurrent model execution, one or multiple models

Multi-model, multi-GPU and asynchronous HTTP and GRPC request handling

Support many model types: CNN, RNN, “stateless”, “stateful”

Multiple scheduling and batching algorithms

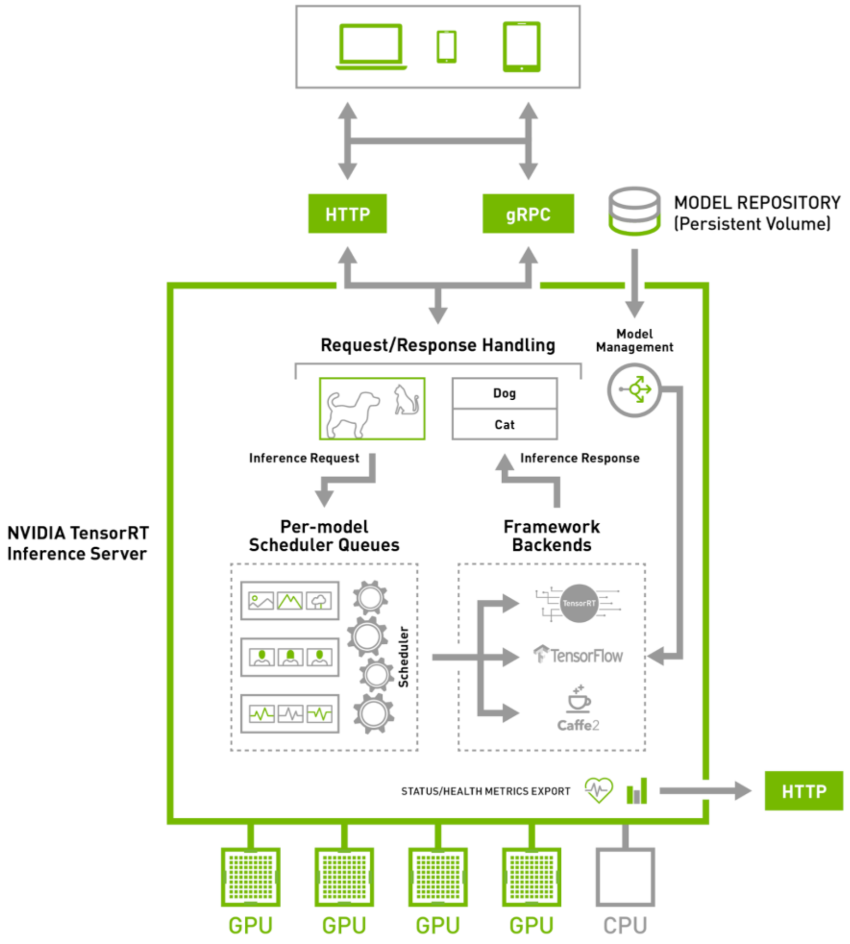
Enable both “online” and “offline” inference use cases

Batch 1, batch n, dynamic batching

Enable scalable, reliable deployment

Prometheus metrics, live/ready endpoints, Kubernetes integration

EXTENSIBLE ARCHITECTURE



Extensible backend architecture allows multiple framework and custom support

Extensible scheduler architecture allows support for different model types and different batching strategies

Leverage CUDA to support model concurrency and multi-GPU

AVAILABLE METRICS

Category	Name	Use Case	Granularity	Frequency
GPU Utilization	Power usage	Proxy for load on the GPU	Per GPU	Per second
	Power limit	Maximum GPU power limit	Per GPU	Per second
	GPU utilization	GPU utilization rate [0.0 - 1.0)	Per GPU	Per second
GPU Memory	GPU Total Memory	Total GPU memory, in bytes	Per GPU	Per second
	GPU Used Memory	Used GPU memory, in bytes	Per GPU	Per second
Count GPU & CPU	Request count	Number of inference requests	Per model	Per request
	Execution count	Number of model inference executions Request count / execution count = avg dynamic request batching	Per model	Per request
	Inference count	Number of inferences performed (one request counts as “batch size” inferences)	Per model	Per request
Latency GPU & CPU	Latency: request time	End-to-end inference request handling time	Per model	Per request
	Latency: compute time	Time a request spends executing the inference model (in the appropriate framework)	Per model	Per request
	Latency: queue time	Time a request spends waiting in the queue before being executed	Per model	Per request

MODEL REPOSITORY

File-system based repository of the models loaded and served by the inference server

Model metadata describes framework, scheduling, batching, concurrency and other aspects of each model

ModelX
- platform: TensorRT
- scheduler: default
- concurrency: …

ModelY
- platform: TensorRT
- scheduler: dynamic-batcher
- concurrency: …

ModelZ
- platform: TensorFlow
- scheduler: sequence-batcher
- concurrency: ...

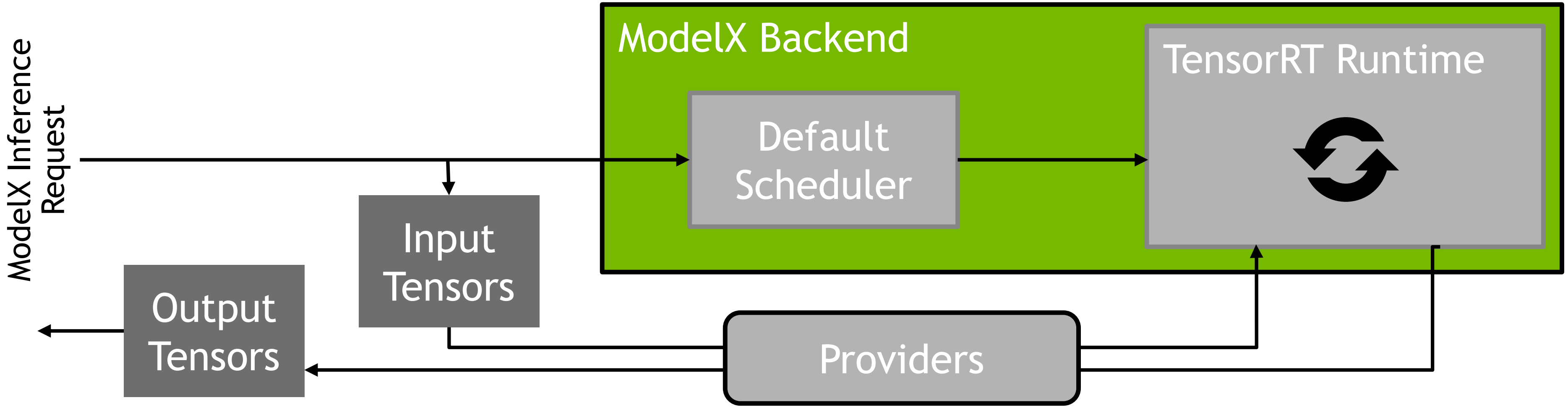
BACKEND ARCHITECTURE

Backend acts as interface between inference requests and a standard or custom framework

- Supported standard frameworks: TensorRT, TensorFlow, Caffe2

Providers efficiently communicate inference request inputs and outputs (HTTP or GRPC)

- Efficient data movement, no additional copies



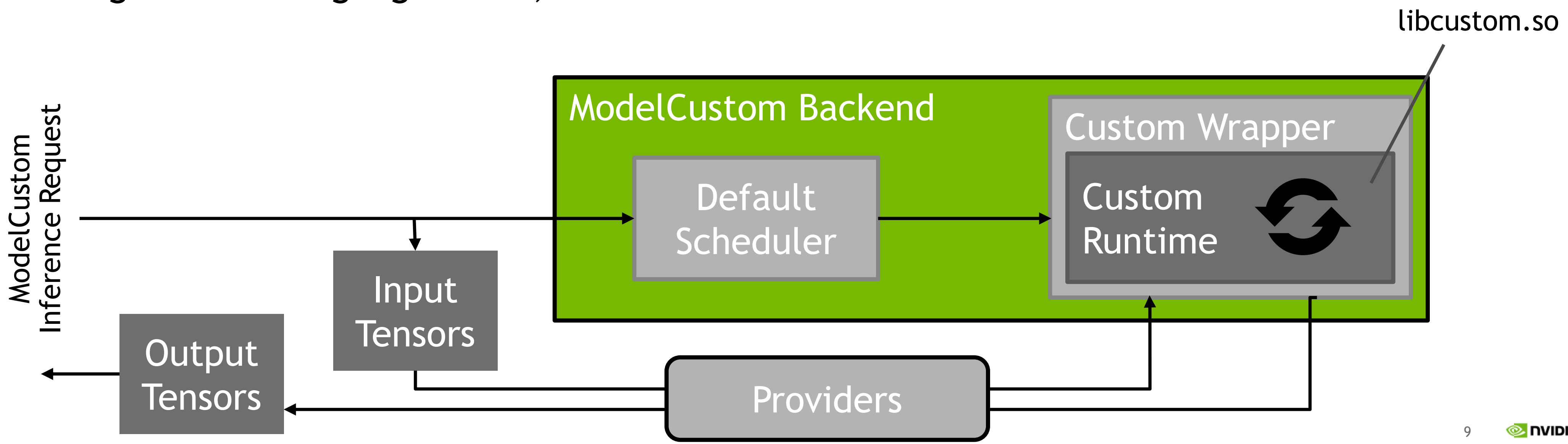
CUSTOM FRAMEWORK

Integrate Custom Logic Into Inference Server

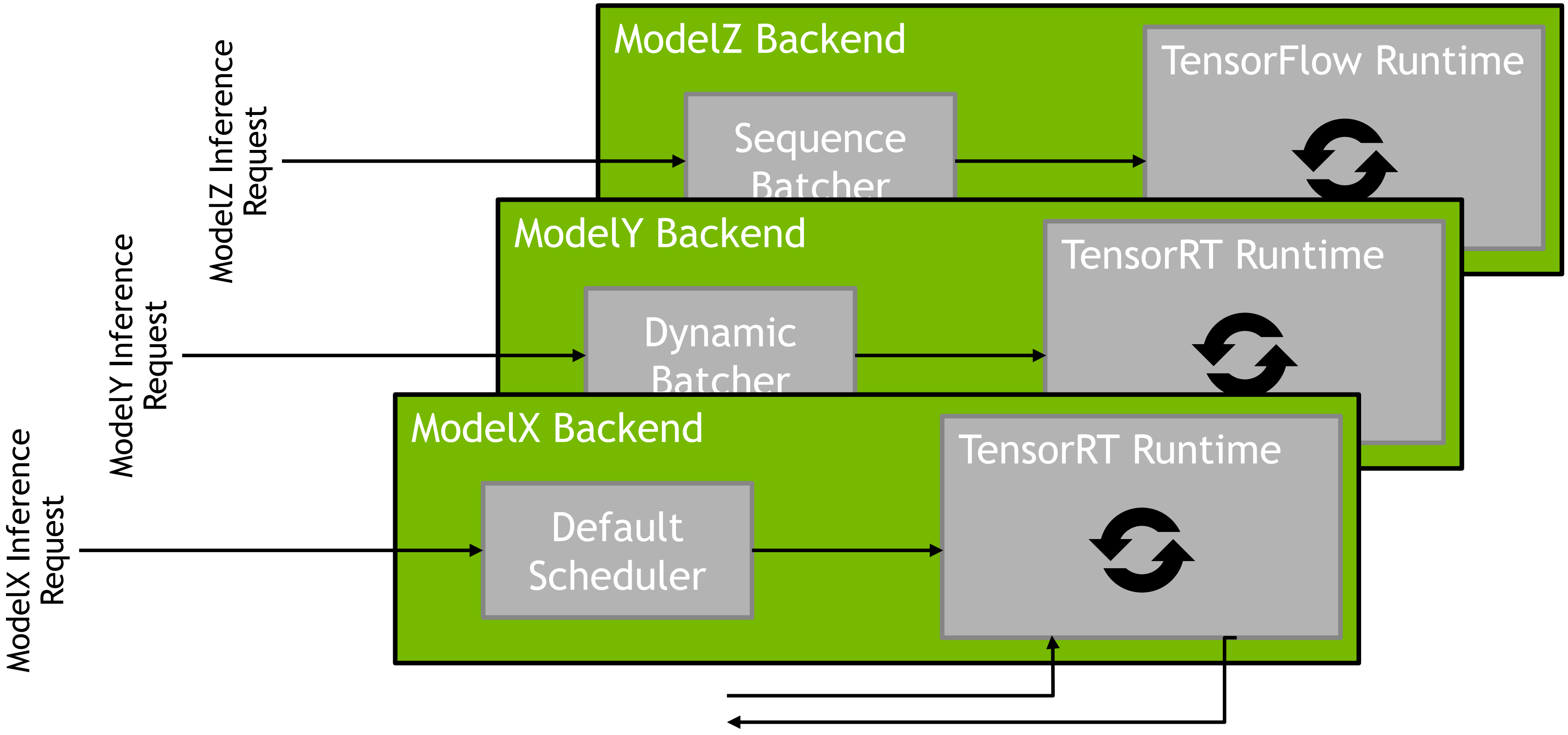
Provide implementation of your “framework”/”runtime” as shared library

Implement simple API: Initialize, Finalize, Execute

All inference server features are available: multi-model, multi-GPU, concurrent execution, scheduling and batching algorithms, etc.



MULTIPLE MODELS

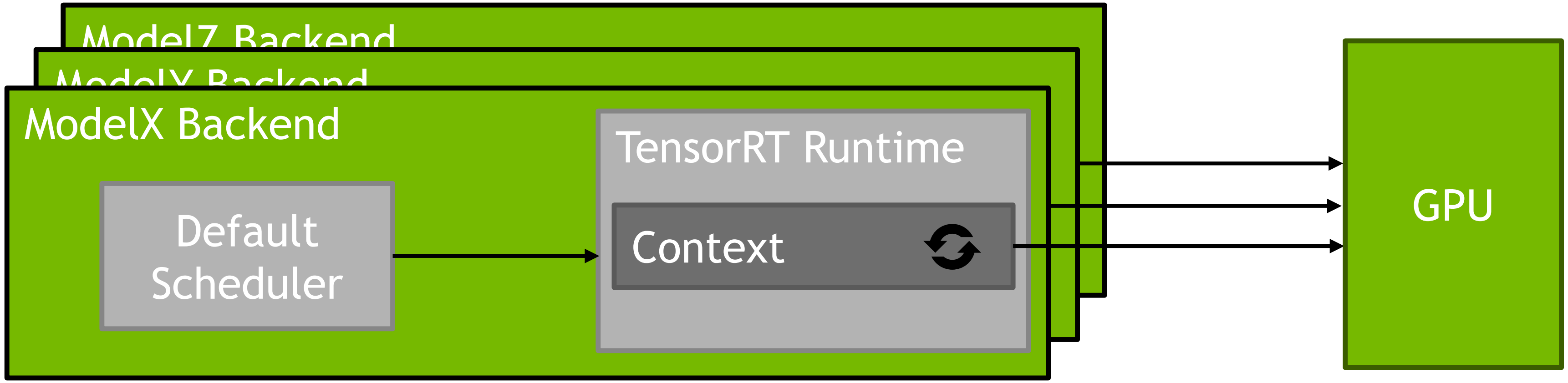


MODEL CONCURRENCY

Multiple Models Sharing a GPU

By default each model gets one *instance* on each available GPU (or 1 CPU instance if no GPUs)

Each instance has an *execution context* that encapsulates the state needed by the runtime to execute the model

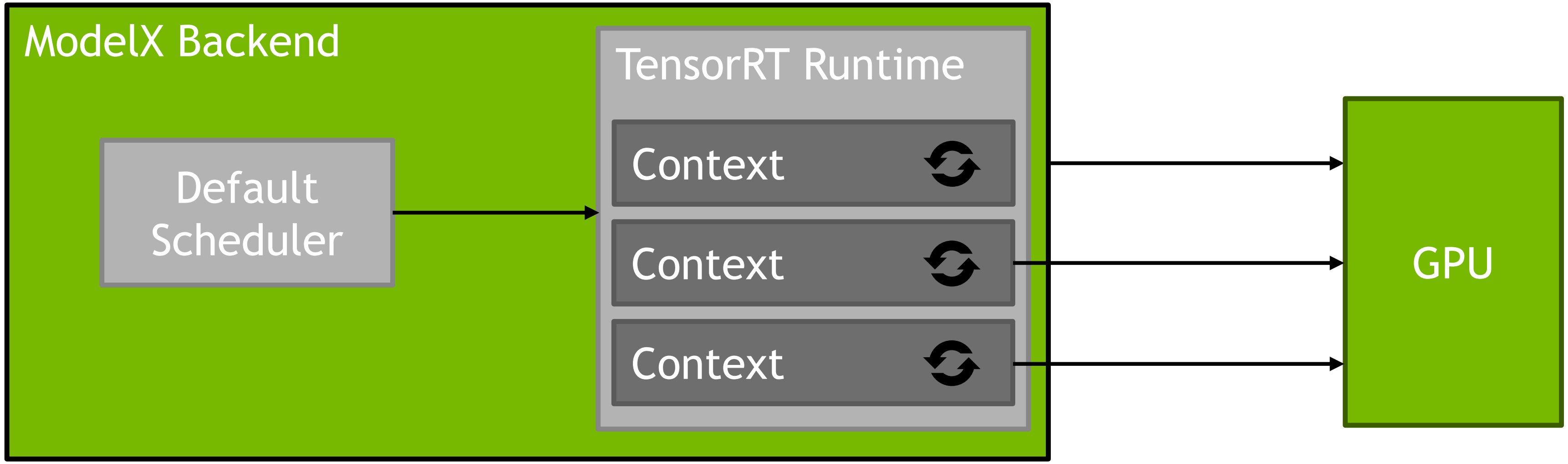


MODEL CONCURRENCY

Multiple Instances of the Same Model

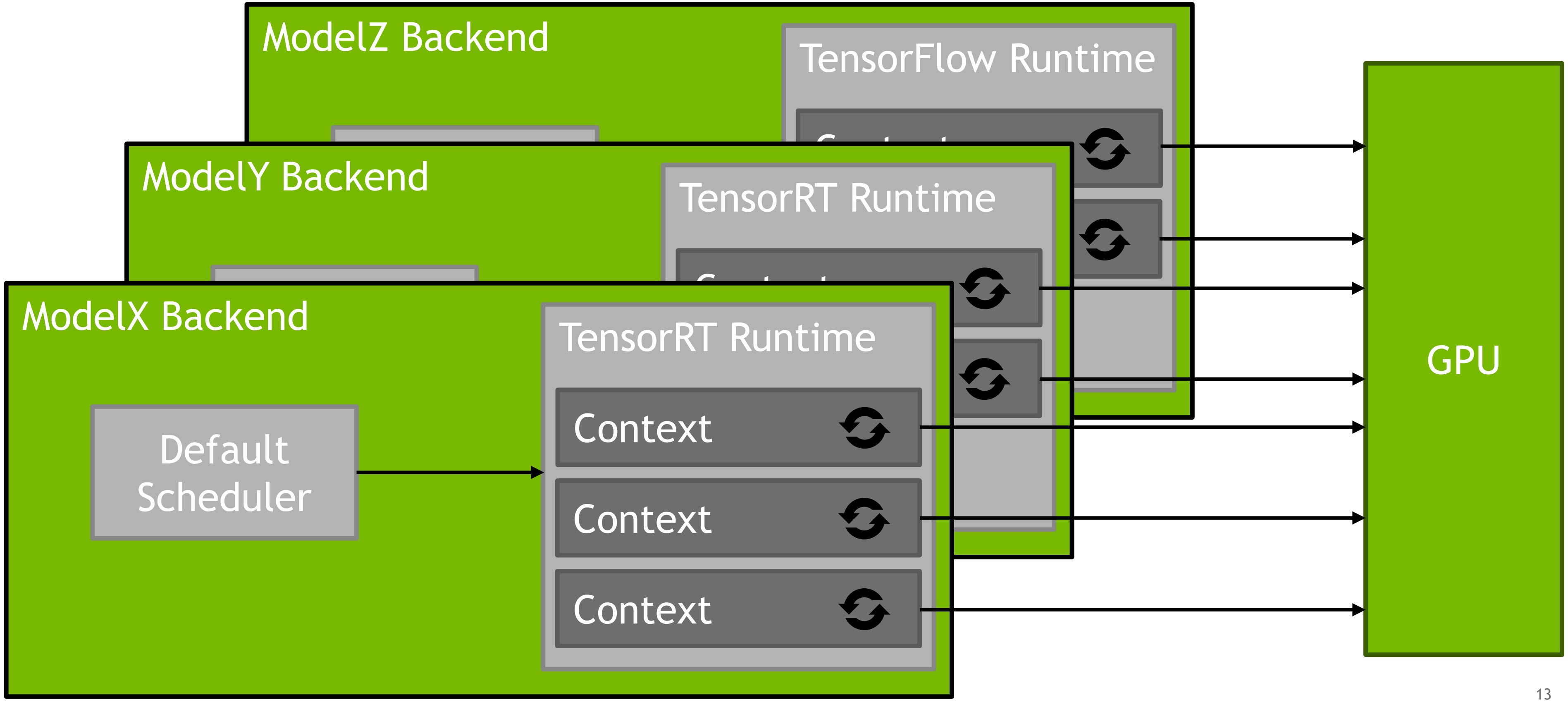
Model metadata allows multiple instances to be configured for each model

Multiple model instances allow multiple inference requests to be executed simultaneously



MODEL CONCURRENCY

Multiple Instances of Multiple Models



CONCURRENT EXECUTION TIMELINE

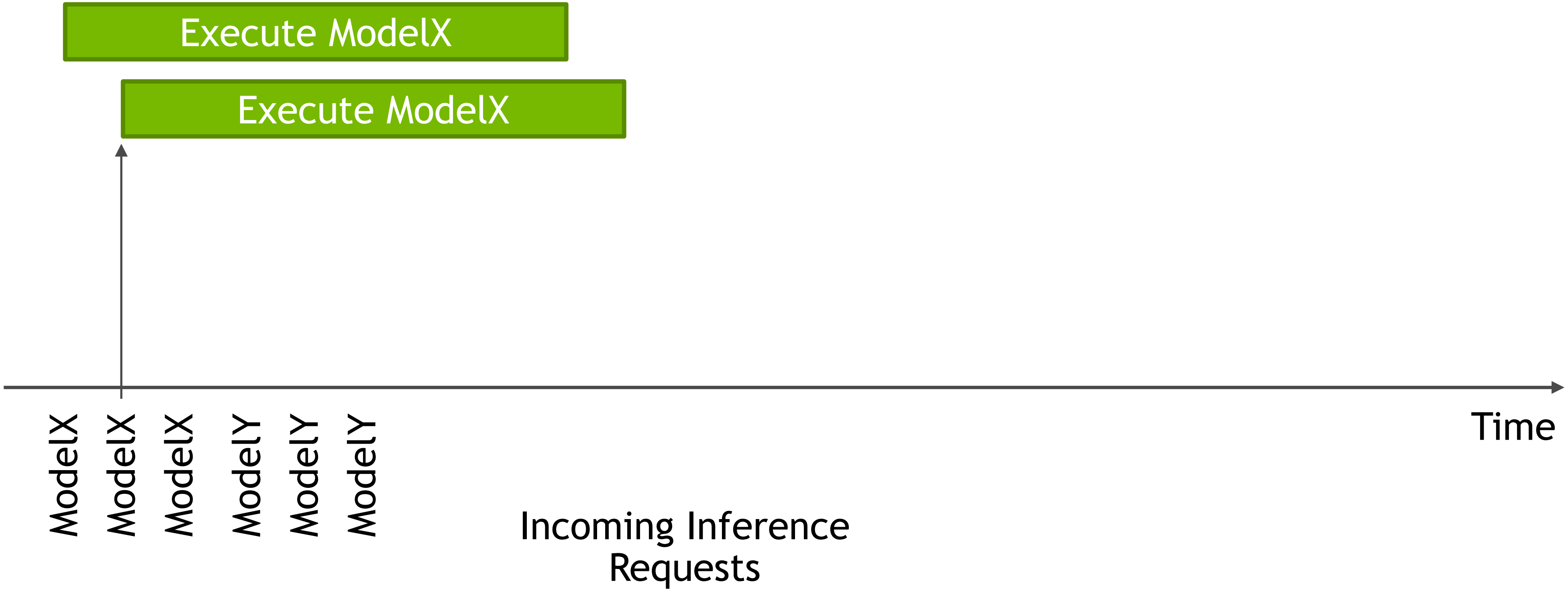
GPU Activity Over Time

CONCURRENT EXECUTION TIMELINE

GPU Activity Over Time

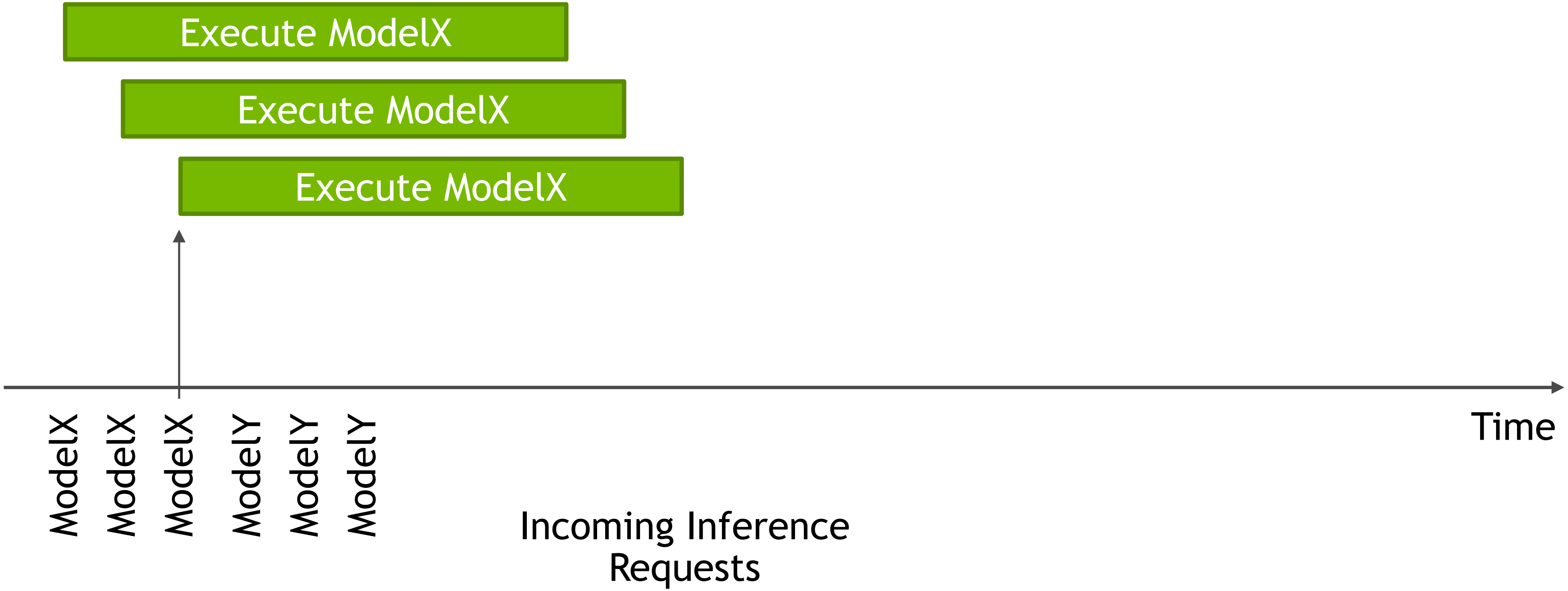
CONCURRENT EXECUTION TIMELINE

GPU Activity Over Time



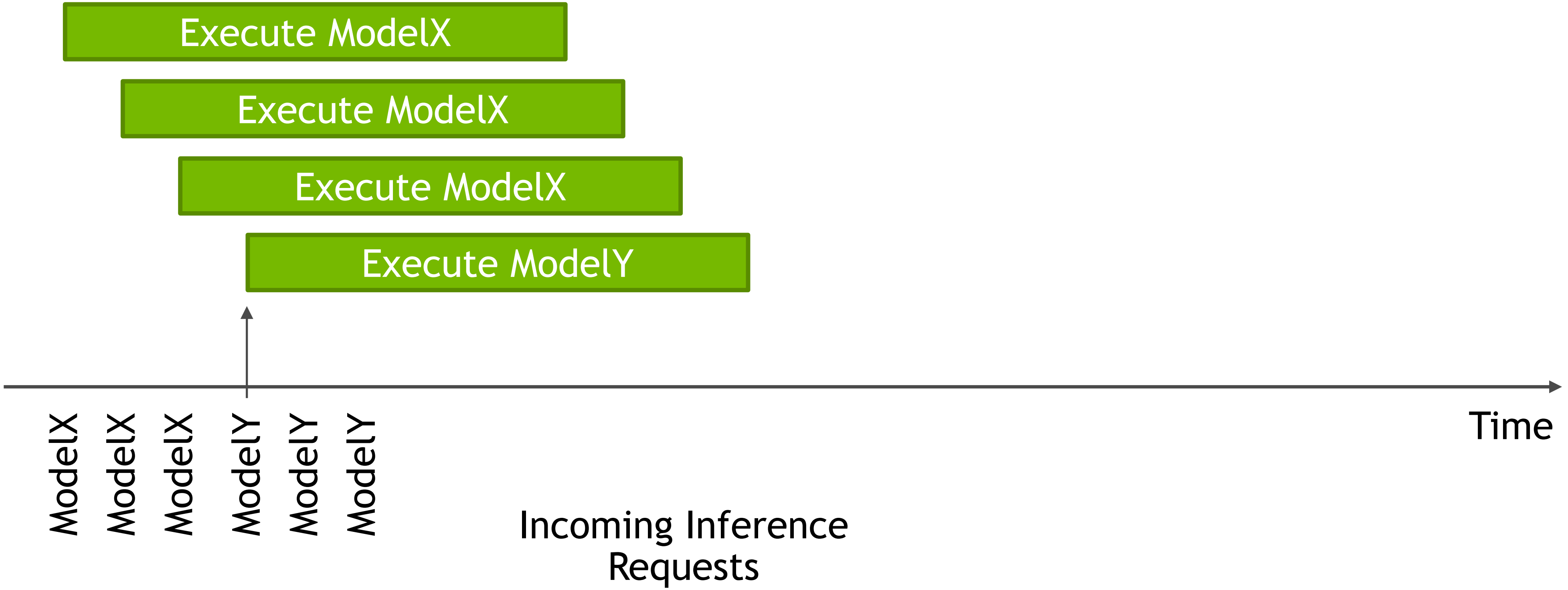
CONCURRENT EXECUTION TIMELINE

GPU Activity Over Time



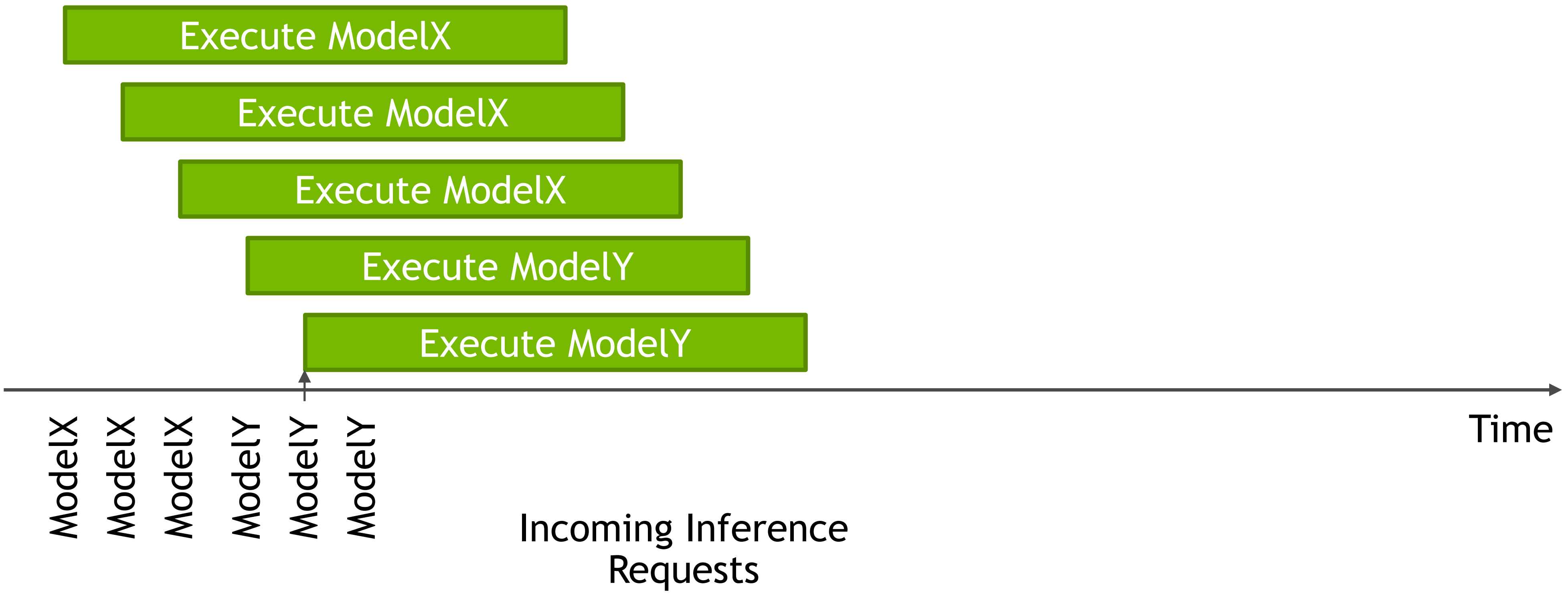
CONCURRENT EXECUTION TIMELINE

GPU Activity Over Time



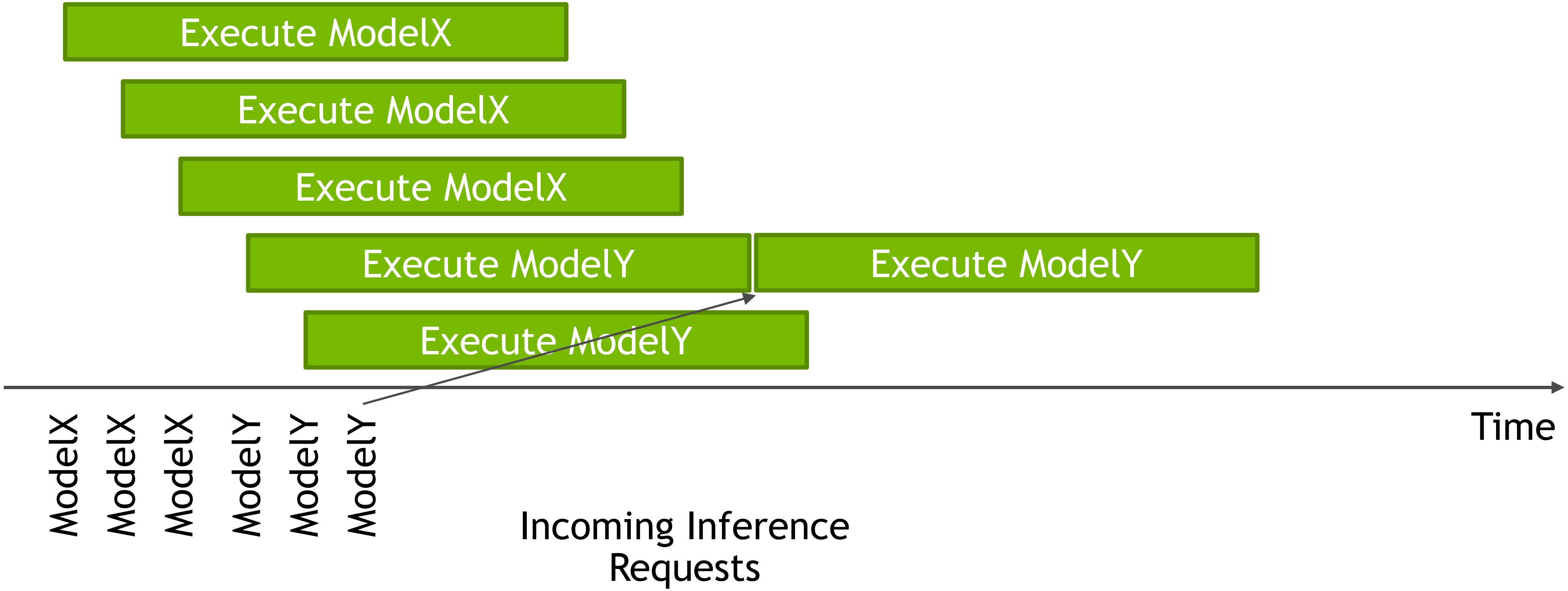
CONCURRENT EXECUTION TIMELINE

GPU Activity Over Time



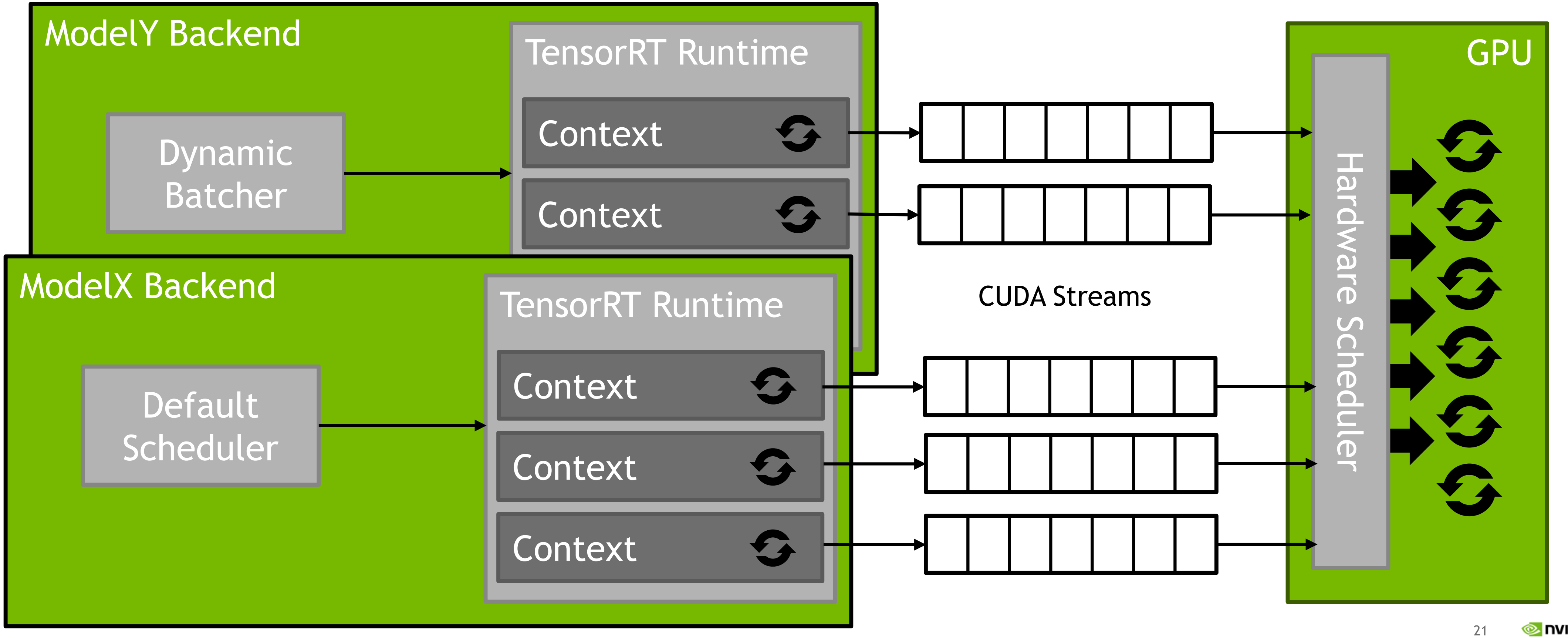
CONCURRENT EXECUTION TIMELINE

GPU Activity Over Time



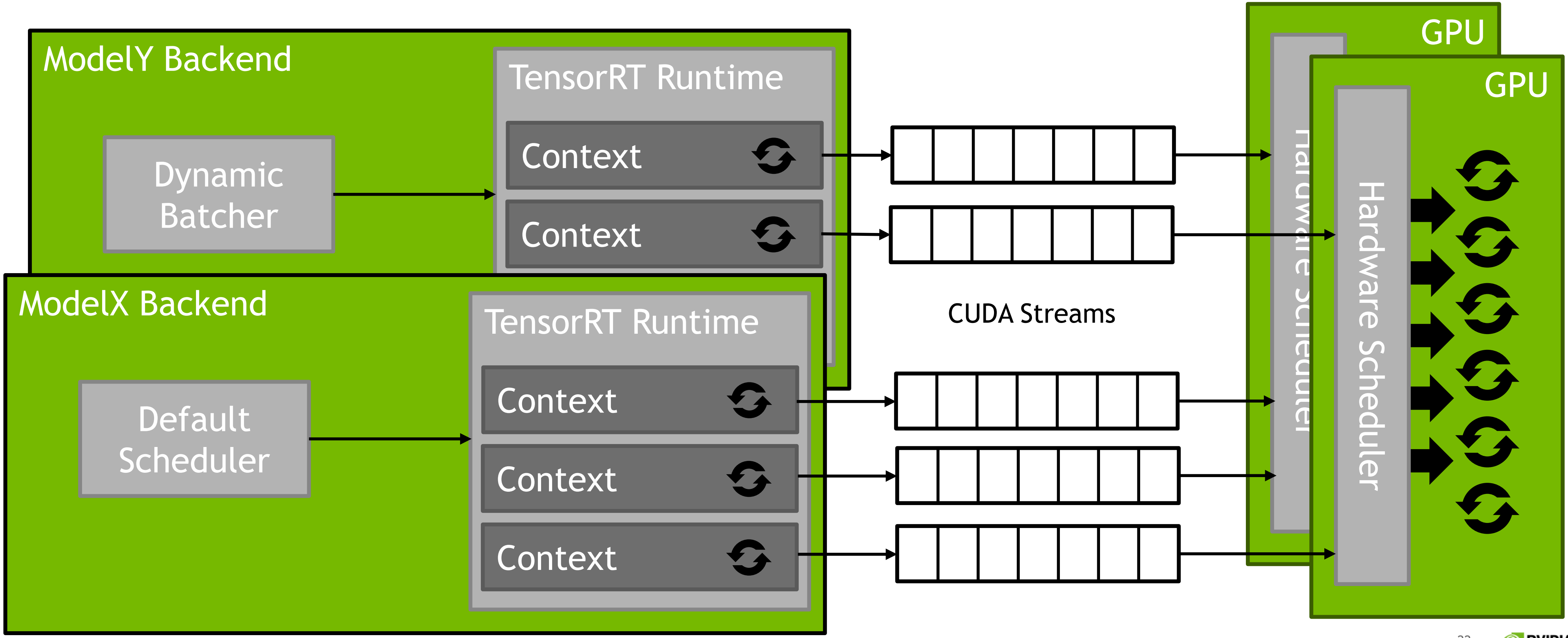
SHARING A GPU

CUDA Enables Multiple Model Execution on a GPU



MUTLI-GPU

Execution Contexts Can Target Multiple GPUs



BATCHING VS NON-BATCHING

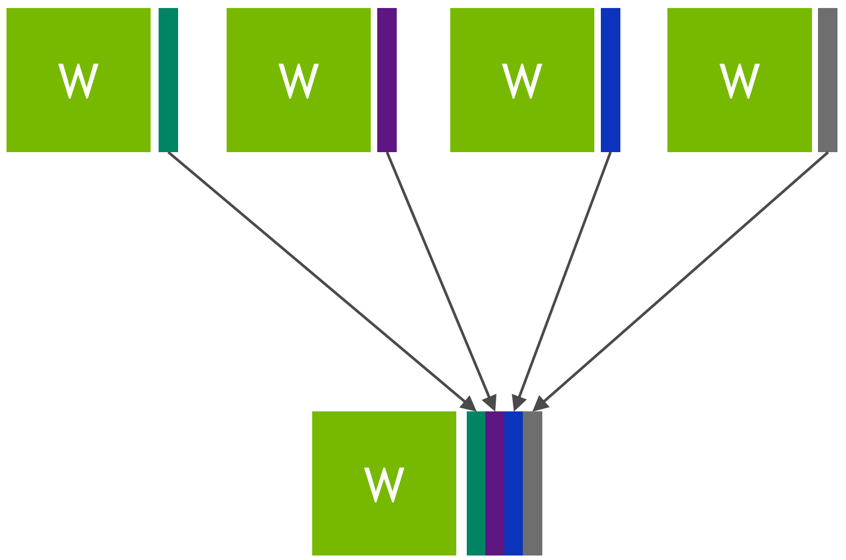
Batching: Grouping Inference Requests Together

Batch size = 1

- Run a single inference task on a GPU
- Low-latency, but the GPU is underutilized

Batch size = N

- Group inference instances together
- High throughput and GPU utilization
- Allows employing Tensor Cores in Volta and Turing

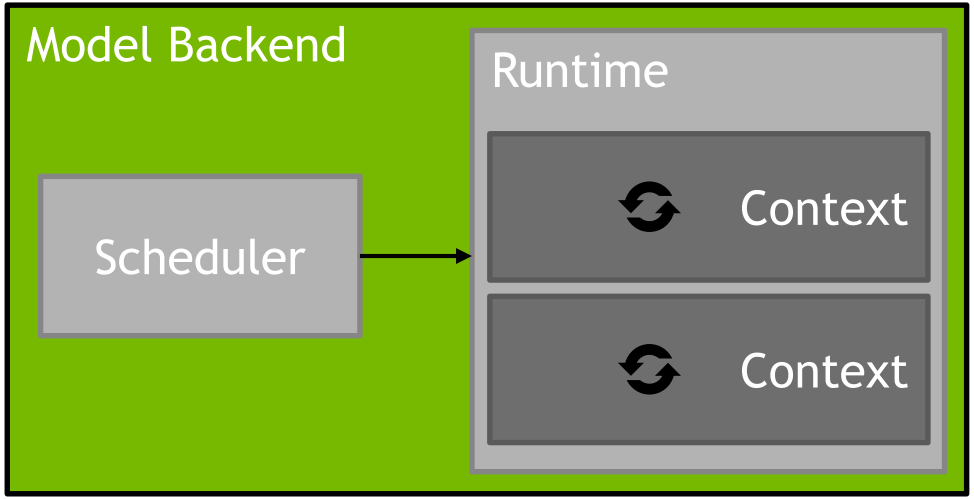


SCHEDULER ARCHITECTURE

Scheduler responsible for managing all inference requests to a given model

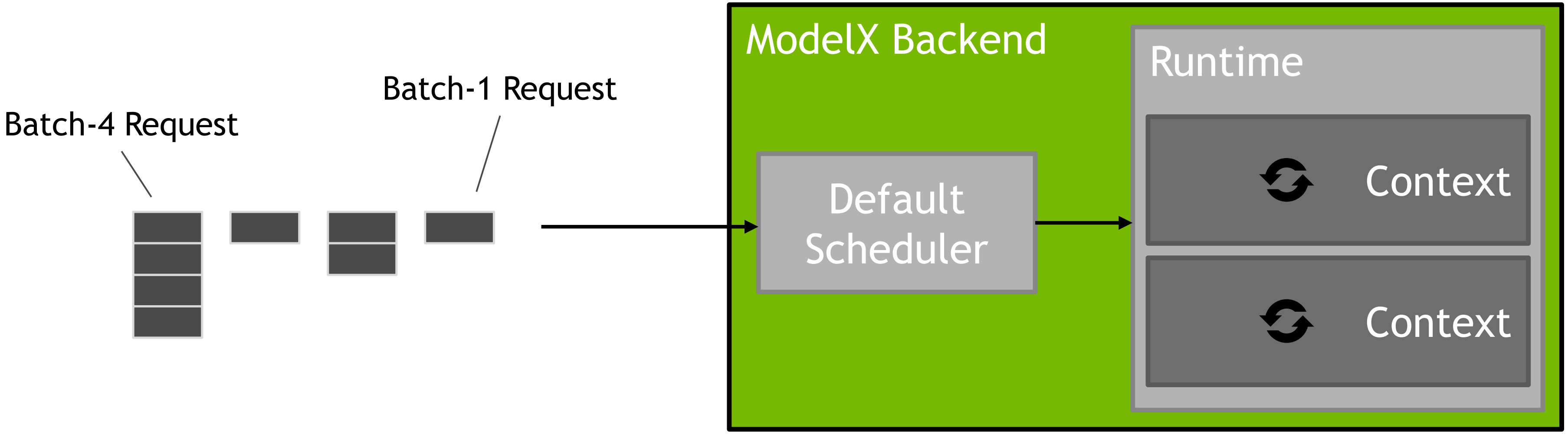
Distribute requests to the available execution contexts

Each model can configure the type of scheduler appropriate for the model



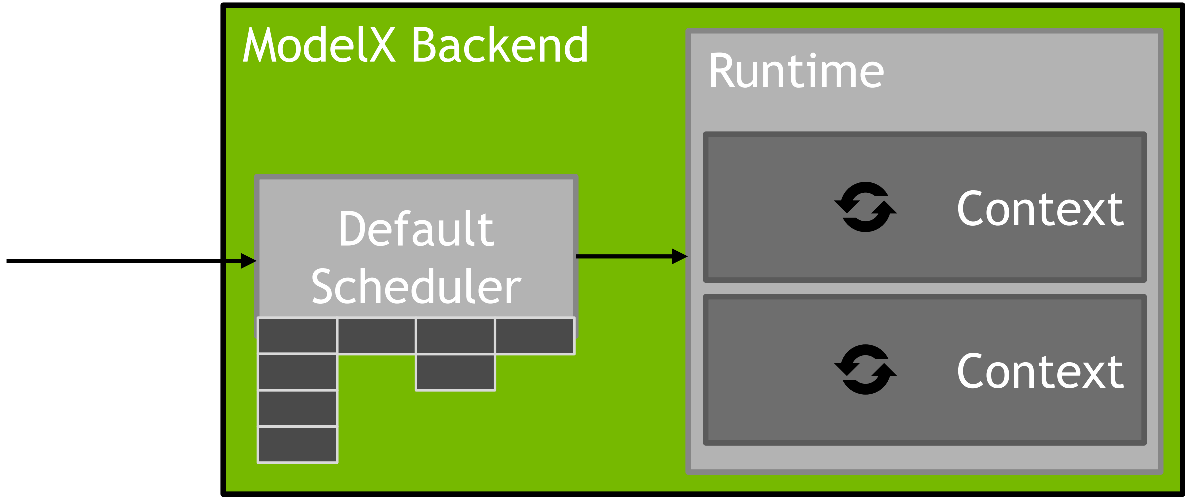
DEFAULT SCHEDULER

Distribute Individual Requests Across Available Contexts



DEFAULT SCHEDULER

Distribute Individual Requests Across Available Contexts



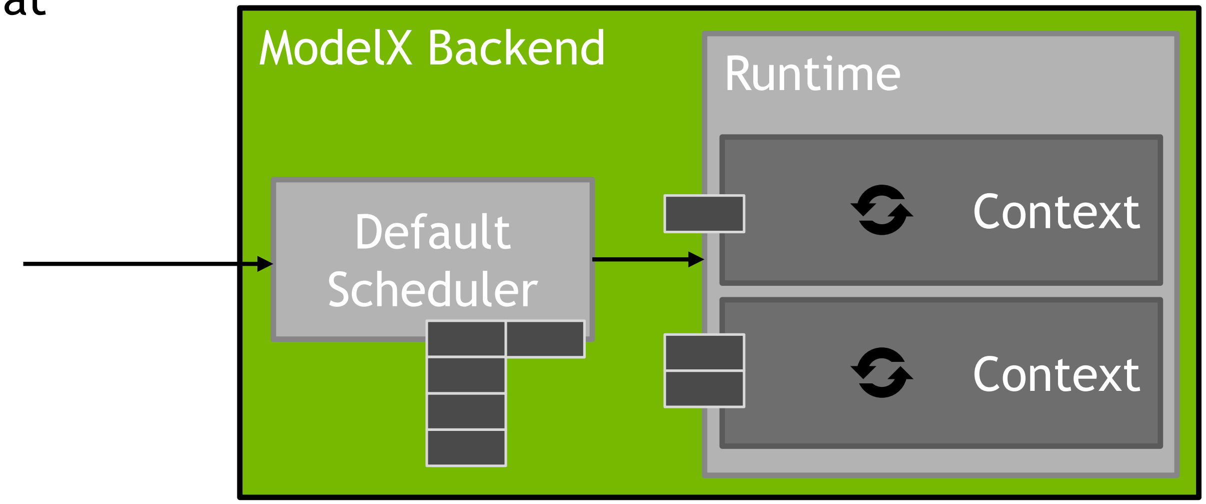
Incoming requests to ModelX queued in scheduler

DEFAULT SCHEDULER

Distribute Individual Requests Across Available Contexts

Assuming GPU is fully utilized by executing 2 batch-4 inferences at the same time.

Utilization = 3/8 = 37.5%



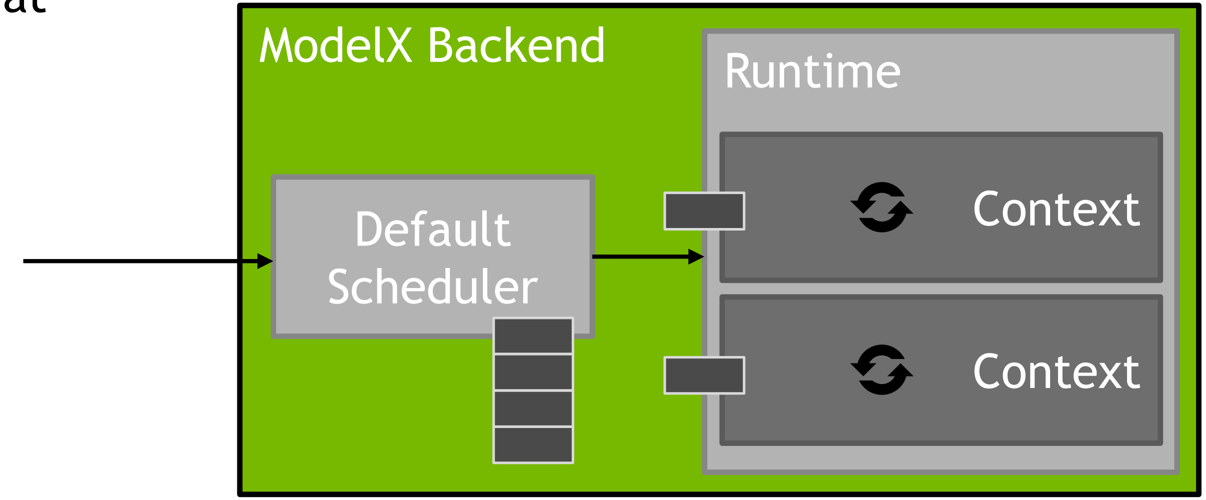
requests assigned in order
to ready contexts

DEFAULT SCHEDULER

Distribute Individual Requests Across Available Contexts

Assuming GPU is fully utilized by executing 2 batch-4 inferences at the same time.

Utilization = 2/8 = 25%



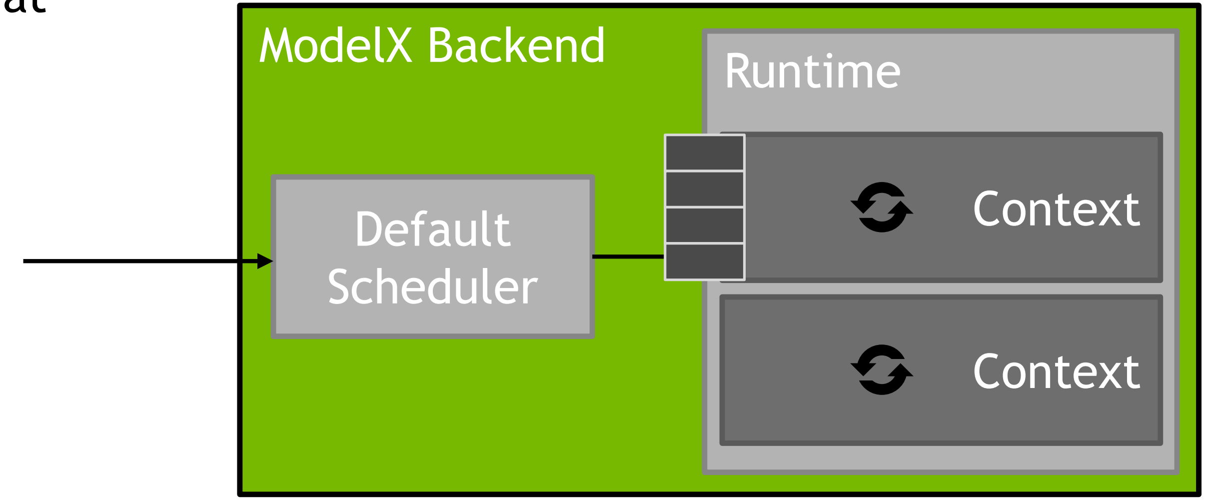
When context completes a new request is assigned

DEFAULT SCHEDULER

Distribute Individual Requests Across Available Contexts

Assuming GPU is fully utilized by executing 2 batch-4 inferences at the same time.

Utilization = 4/8 = 50%



When context completes a new request is assigned

DYNAMIC BATCHING SCHEDULER

Group Requests To Form Larger Batches, Increase GPU Utilization

Default scheduler takes advantage of multiple model instances

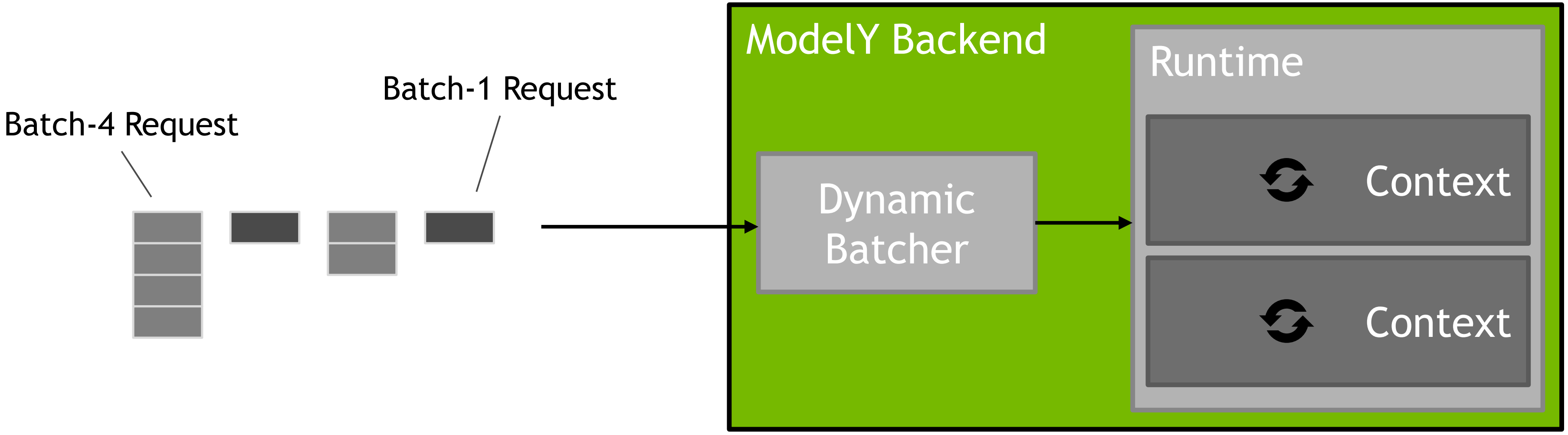
But GPU utilization dependent on the batch-size of the inference request

Batching is often on of the best ways to increase GPU utilization

Dynamic batch scheduler (aka dynamic batcher) forms larger batches by combining multiple inference request

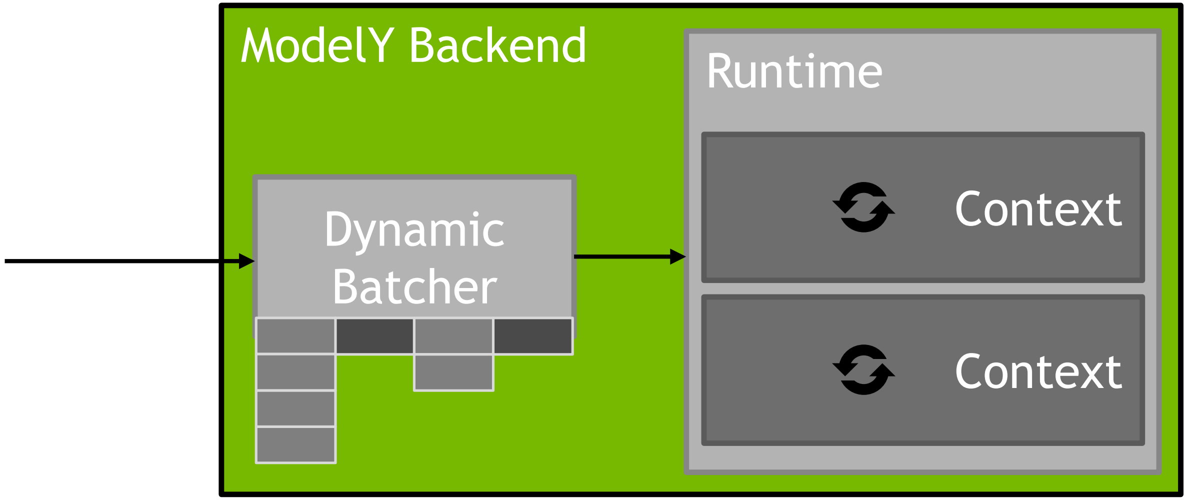
DYNAMIC BATCHING SCHEDULER

Group Requests To Form Larger Batches, Increase GPU Utilization



DYNAMIC BATCHING SCHEDULER

Group Requests To Form Larger Batches, Increase GPU Utilization



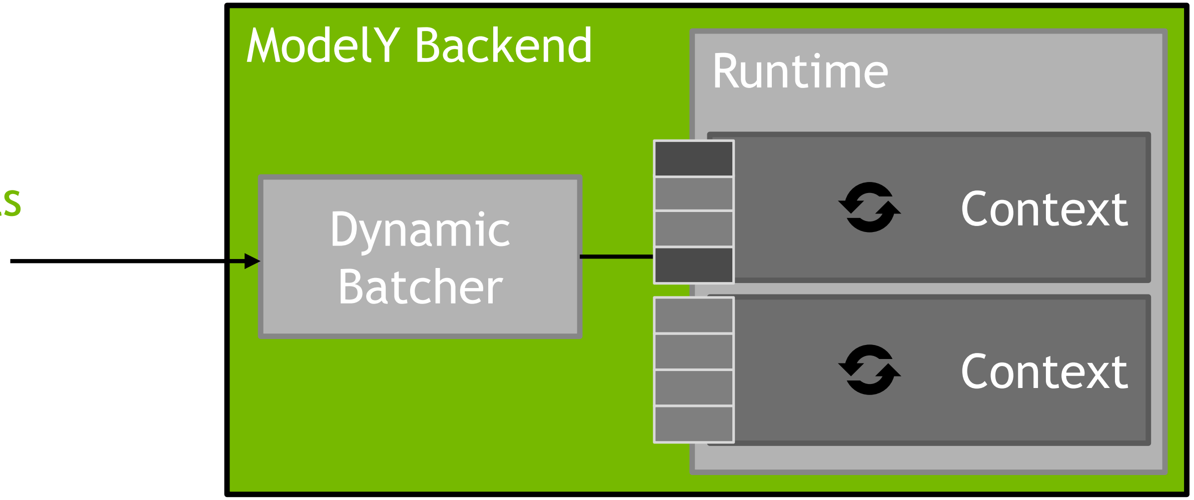
Incoming requests to ModelY queued in scheduler

DYNAMIC BATCHING SCHEDULER

Group Requests To Form Larger Batches, Increase GPU Utilization

Dynamic batcher configuration for ModelY can specify preferred batch-size. Assume 4 gives best utilization.

Dynamic batcher groups requests to give 100% utilization



SEQUENCE BATCHING SCHEDULER

Dynamic Batching for Stateful Models

Default and dynamic-batching schedulers work with stateless models; each request is scheduled and executed independently

Some models are stateful, a sequence of inference requests must be routed to the same model instance

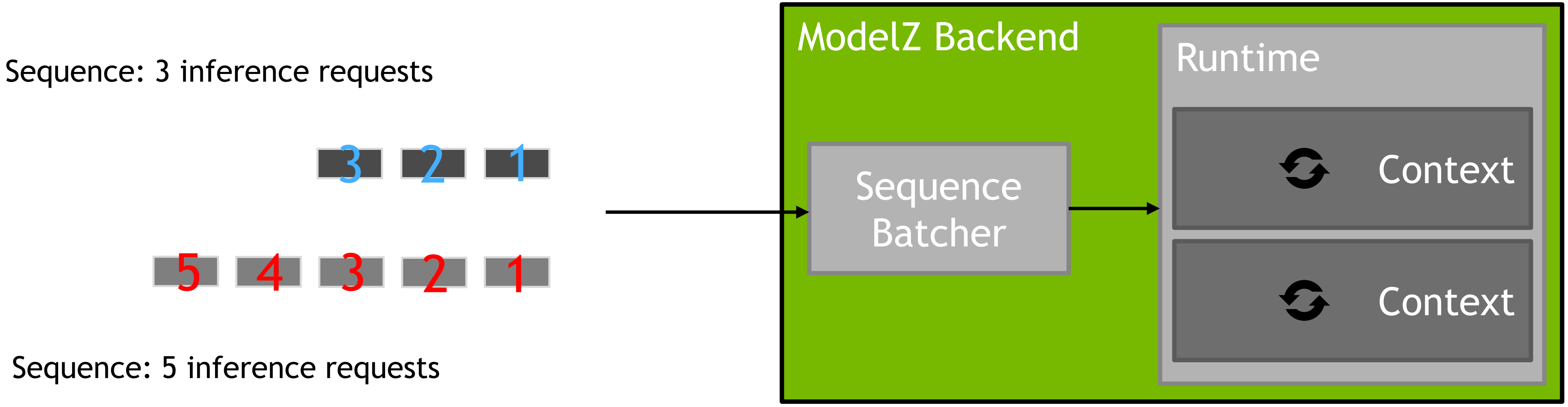
- "Online" ASR, TTS, and similar models
- Models that use LSTM, GRU, etc. to maintain state across inference requests

Multi-instance and batching required by these models to maximum GPU utilization

Sequence-batching scheduler provides dynamically batching for stateful models

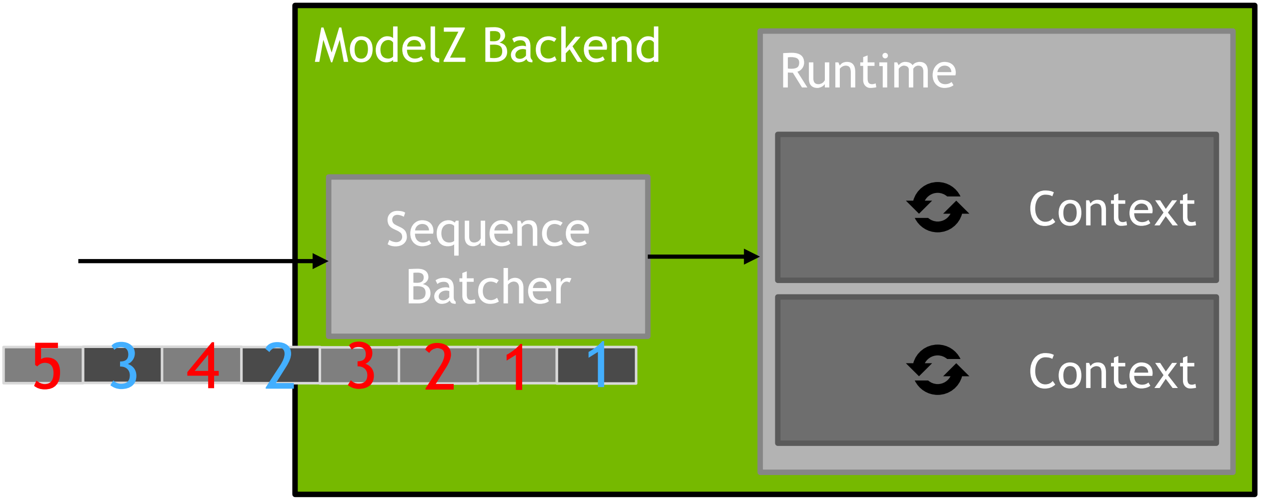
SEQUENCE BATCHING SCHEDULER

Dynamic Batching for Stateful Models



SEQUENCE BATCHING SCHEDULER

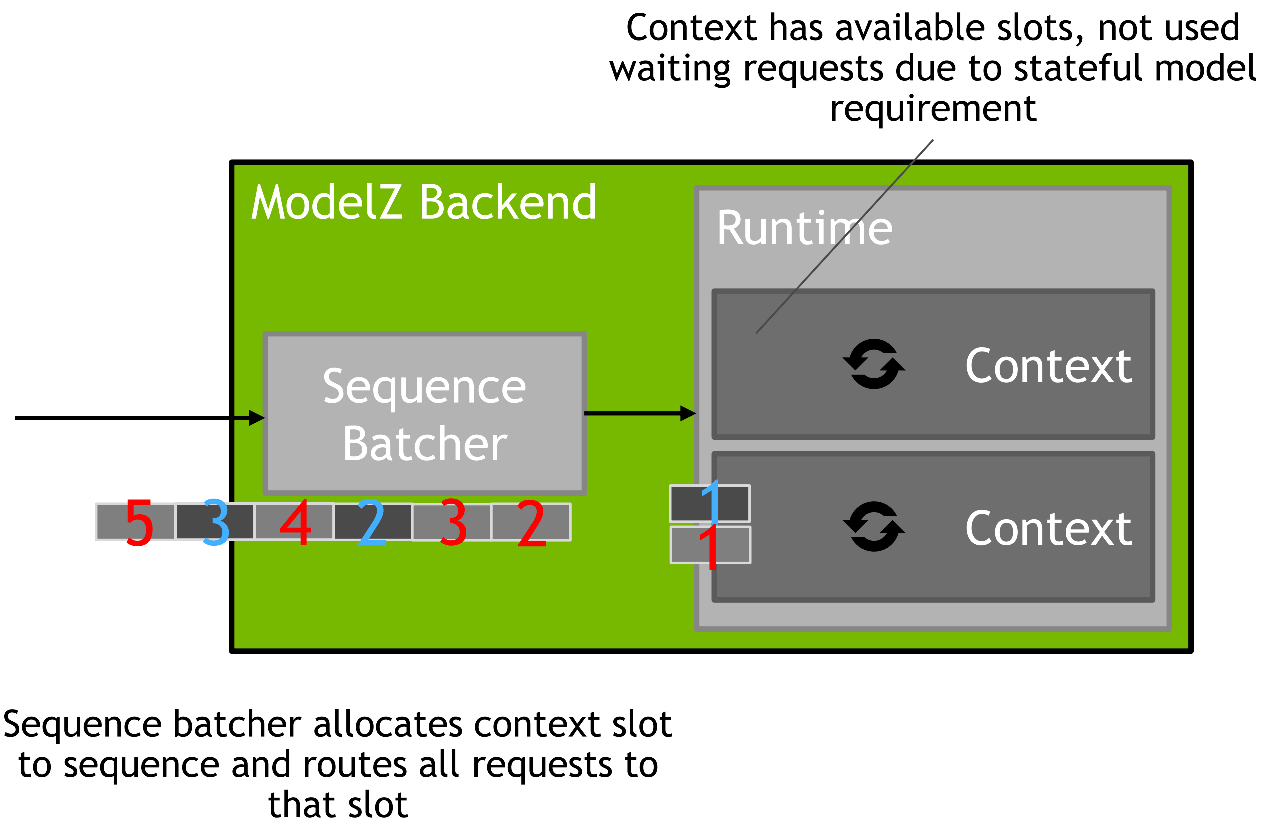
Dynamic Batching for Stateful Models



Inference requests arrive
in arbitrary order

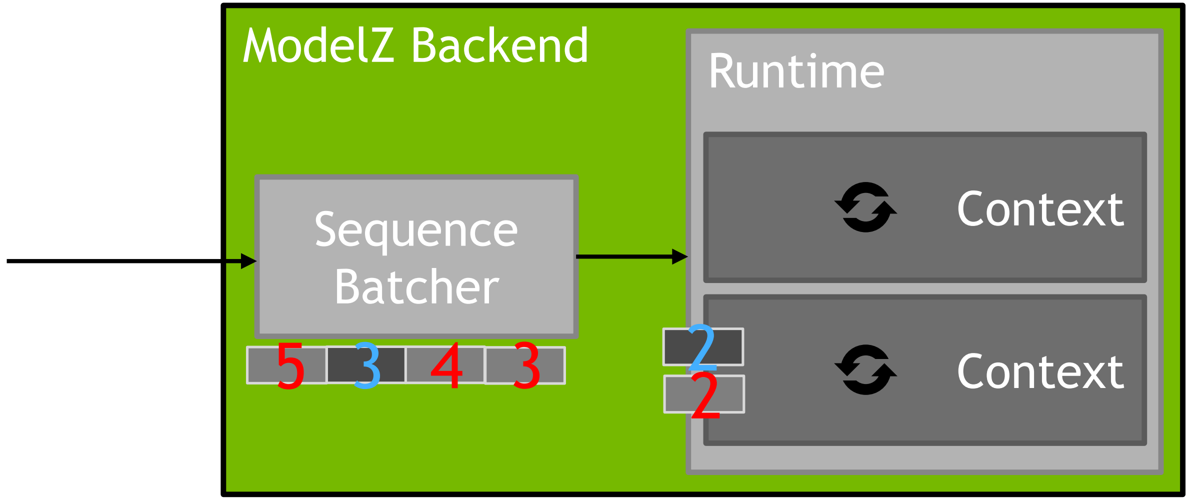
SEQUENCE BATCHING SCHEDULER

Dynamic Batching for Stateful Models



SEQUENCE BATCHING SCHEDULER

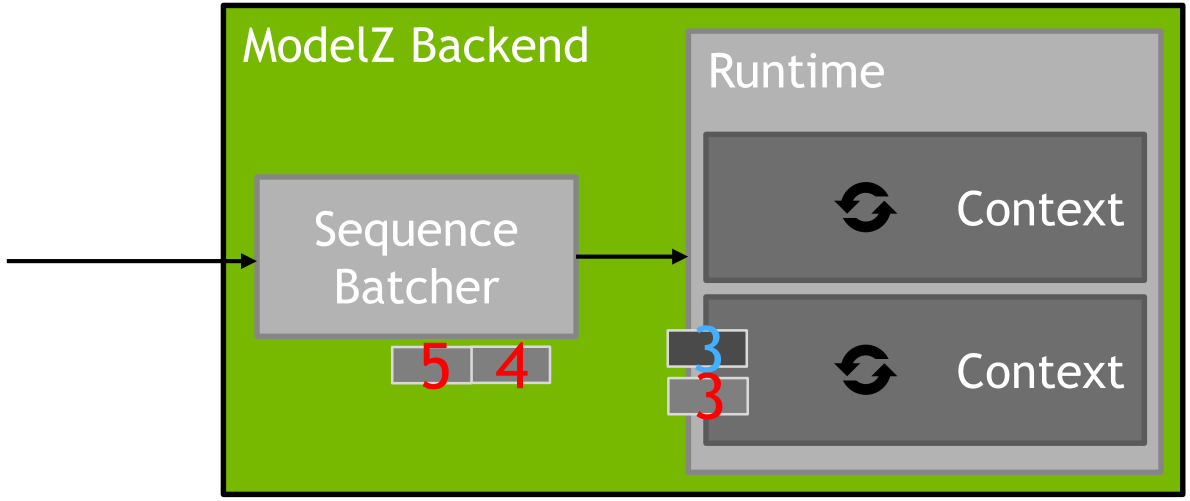
Dynamic Batching for Stateful Models



Sequence batcher allocates context slot to sequence and routes all requests to that slot

SEQUENCE BATCHING SCHEDULER

Dynamic Batching for Stateful Models



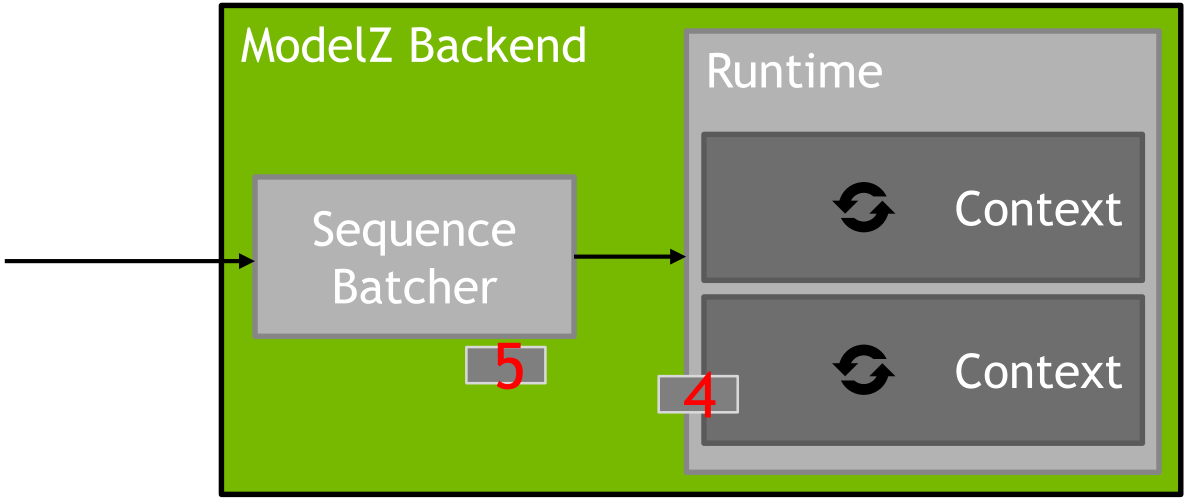
Sequence batcher allocates context slot to sequence and routes all requests to that slot

SEQUENCE BATCHING SCHEDULER

Dynamic Batching for Stateful Models

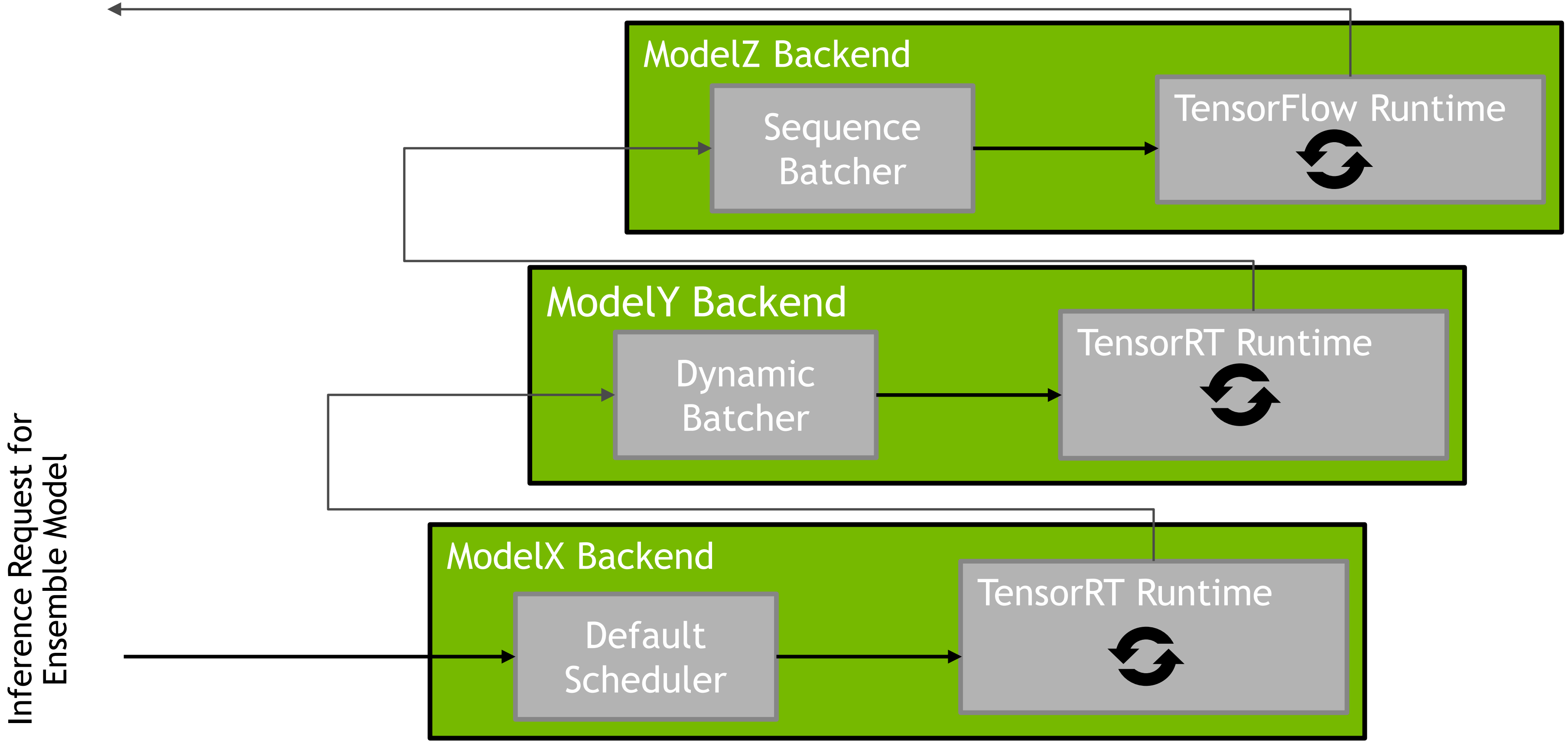
On a fully-loaded server, all context slots would be occupied by sequences.

As soon as one sequence ends another is allocated to the slot.



ENSEMBLE MODELS

A way of pipelining models in TRTIS



Recap

Concurrent Model Execution
Multiple models (or multiple instances of same model) may execute on GPU simultaneously

CPU Model Inference Execution
Framework native models can execute inference requests on the CPU

Metrics
Utilization, count, and latency

Custom Backend
Custom backend allows the user more flexibility by providing their own implementation of an execution engine through the use of a shared library

Stateless / Stateful Inference
Supports many model types including CNN, RNN, etc

Dynamic Batching
Inference requests can be batched up by the inference server to 1) the model-allowed maximum or 2) the user-defined latency SLA

Multiple Model Format Support
TensorFlow GraphDef/SavedModel
TensorFlow and TensorRT GraphDef
TensorRT Plans
Caffe2 NetDef (ONNX import path)

Ensemble Model Support
An Ensemble represents a pipeline of one or more models and the connection of input and output tensors between those models

Multi-GPU support
The server can distribute inferencing across all system GPUs

Caffe2 TensorFlow

TensorRT

ONNX
Chainer Microsoft CNTK
mxnet PYTORCH

soyoungj@nvidia.com