



# MAXIMIZING UTILIZATION FOR DATA CENTER INFERENCE WITH TENSORRT INFERENCE SERVER

David Goodwin, Soyoung Jeong

# AGENDA

Important capabilities to maximize data center utilization

TensorRT Inference Server architecture for maximum utilization

- Multi-frameworks

- Multi-models

- Model concurrency

Real-world use-case: Naver

# MAXIMIZING UTILIZATION

Often GPU is not fully utilized by a single model... increase utilization by:

- Supporting a variety of model frameworks

- Supporting concurrent model execution, one or multiple models

- Supporting many model types: CNN, RNN, “stateless”, “stateful”

- Enabling both “online” and “offline” inference use cases

- Enabling scalable, reliable deployment

# 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

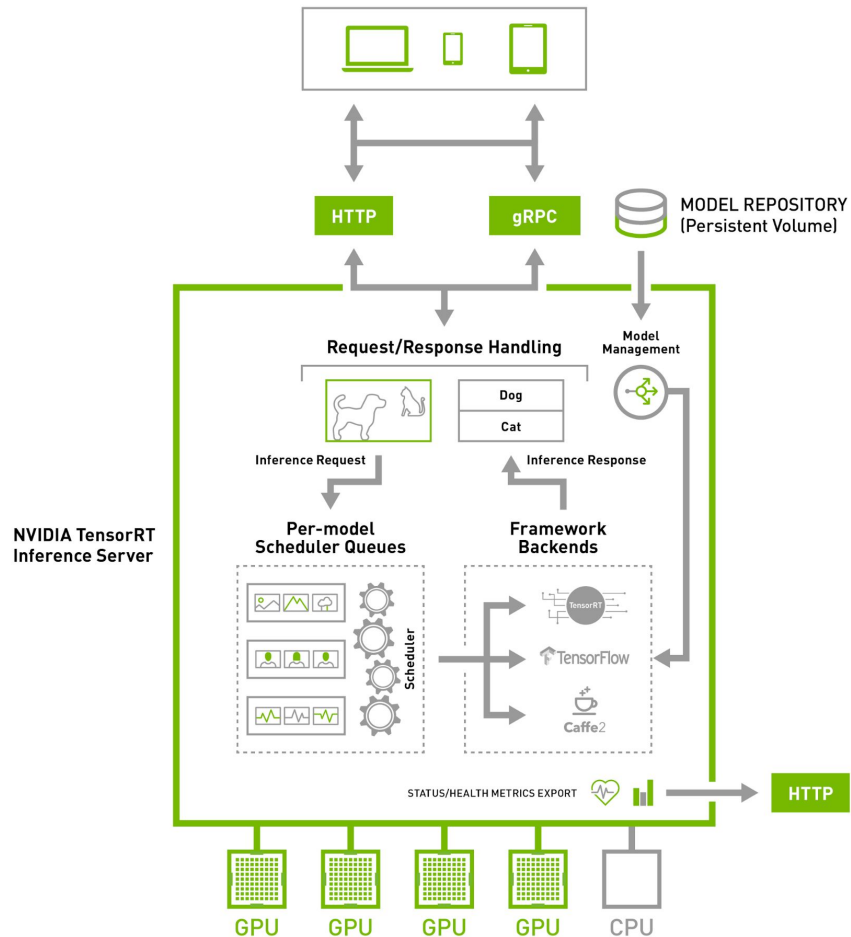
Enable both “online” and “offline” inference use cases

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

# 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: ...

ModelZ

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

ModelY

platform: TensorRT  
scheduler: dynamic-batcher  
concurrency: ...

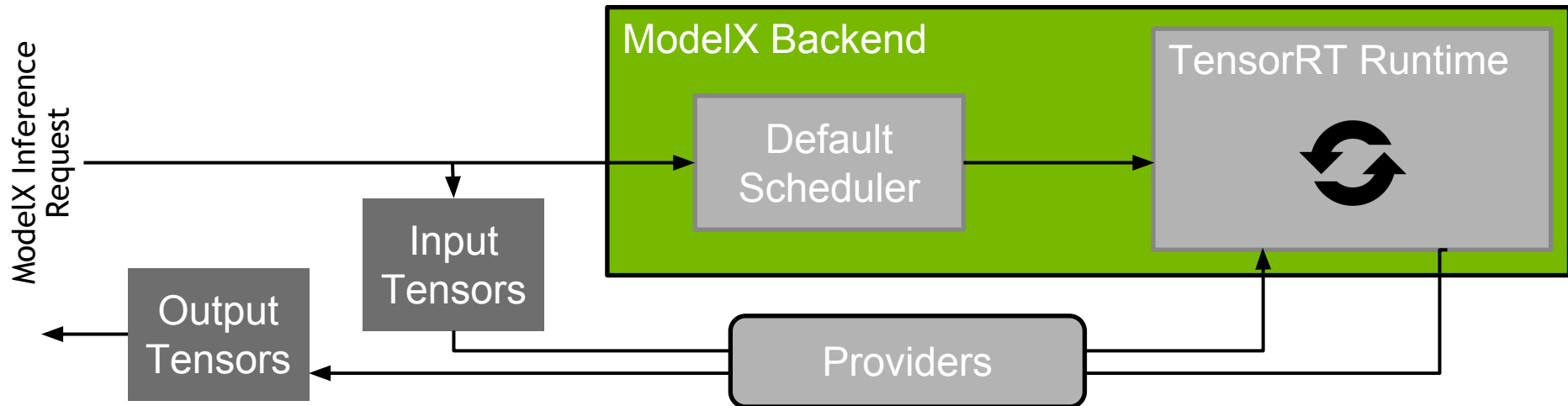
# BACKEND ARCHITECTURE

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

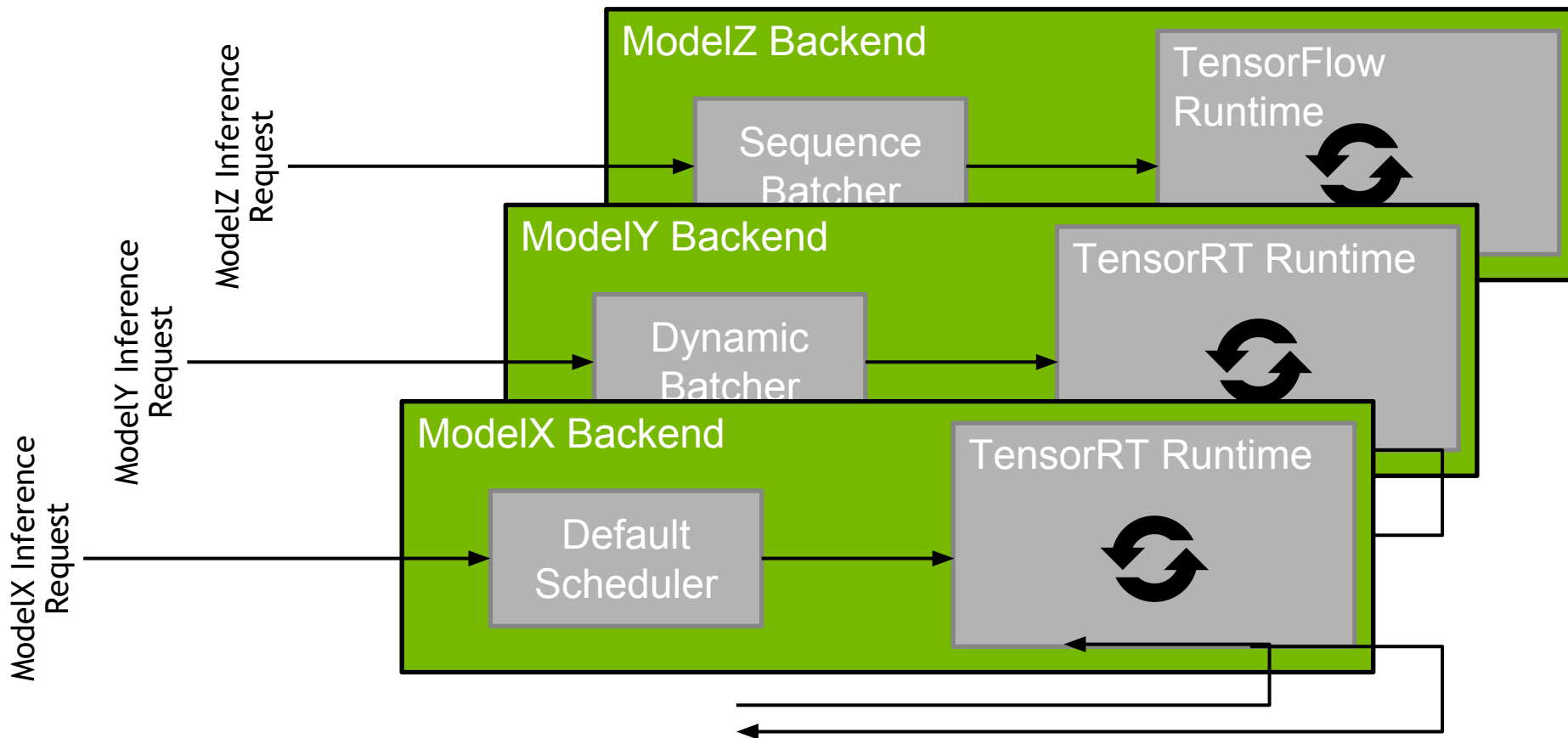
Supported standard frameworks: TensorRT, TensorFlow, Caffe2

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

Efficient data movement, no additional copies



# MULTIPLE MODELS



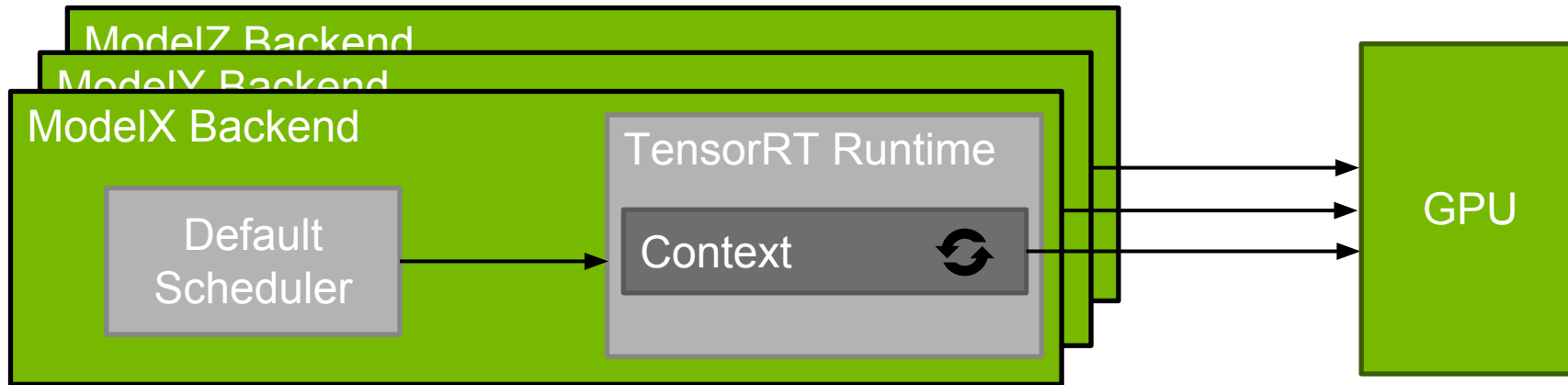


# MODEL CONCURRENCY

## 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

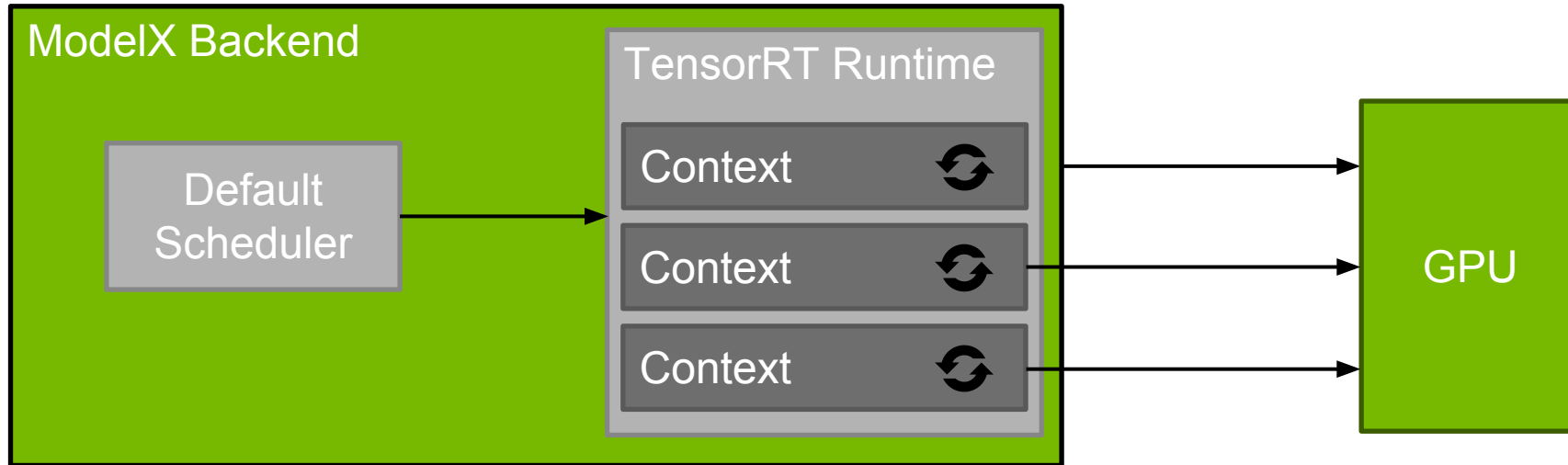


# MODEL CONCURRENCY

## Multiple Instances of the Same Model

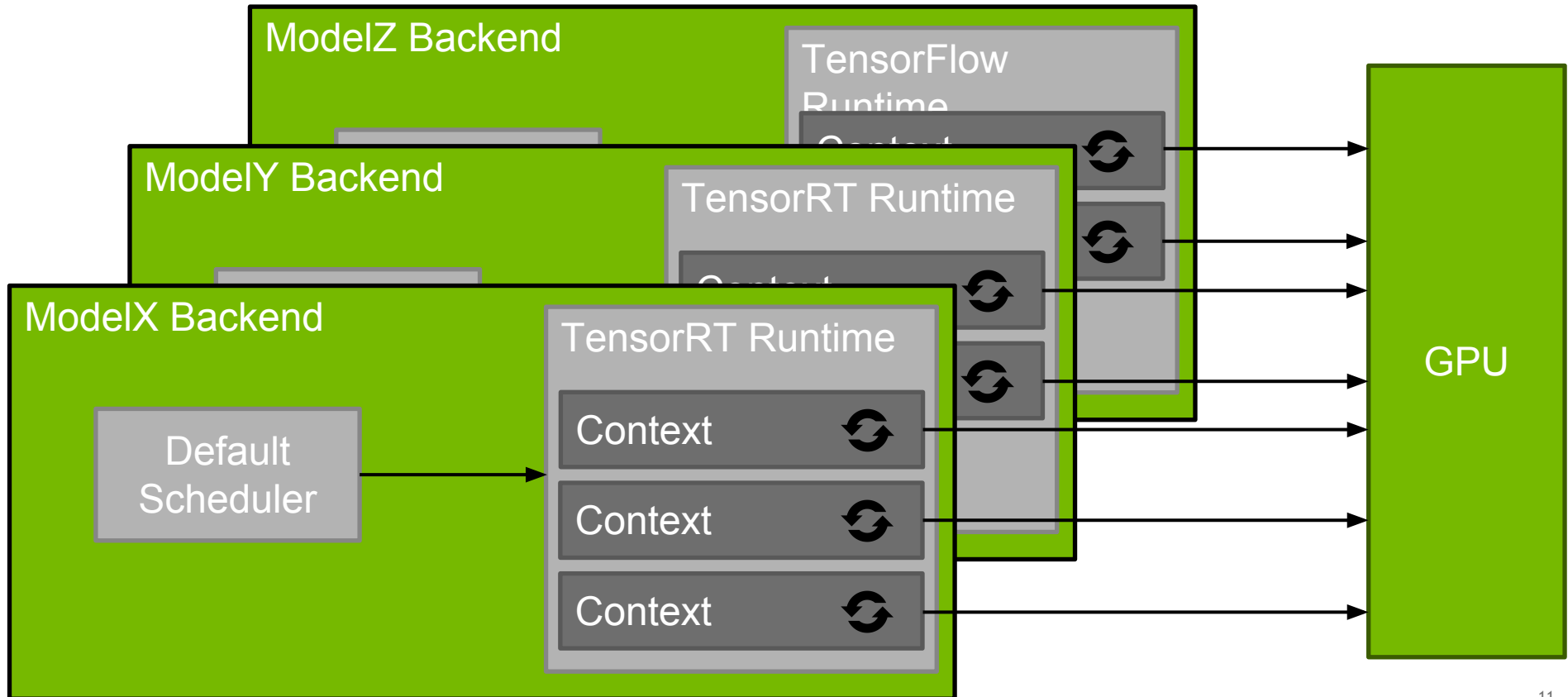
Model metadata allows multiple instances to be configured for each model

Multiple model instances allow multiple inference requests to be executed simultaneously



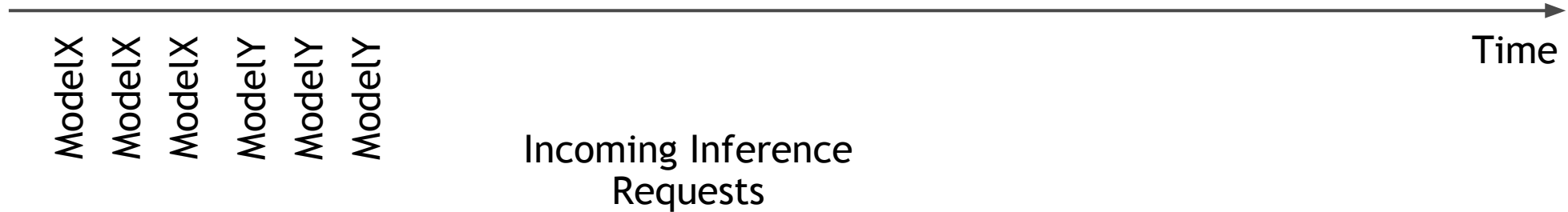
# MODEL CONCURRENCY

Multiple Instances of Multiple Models



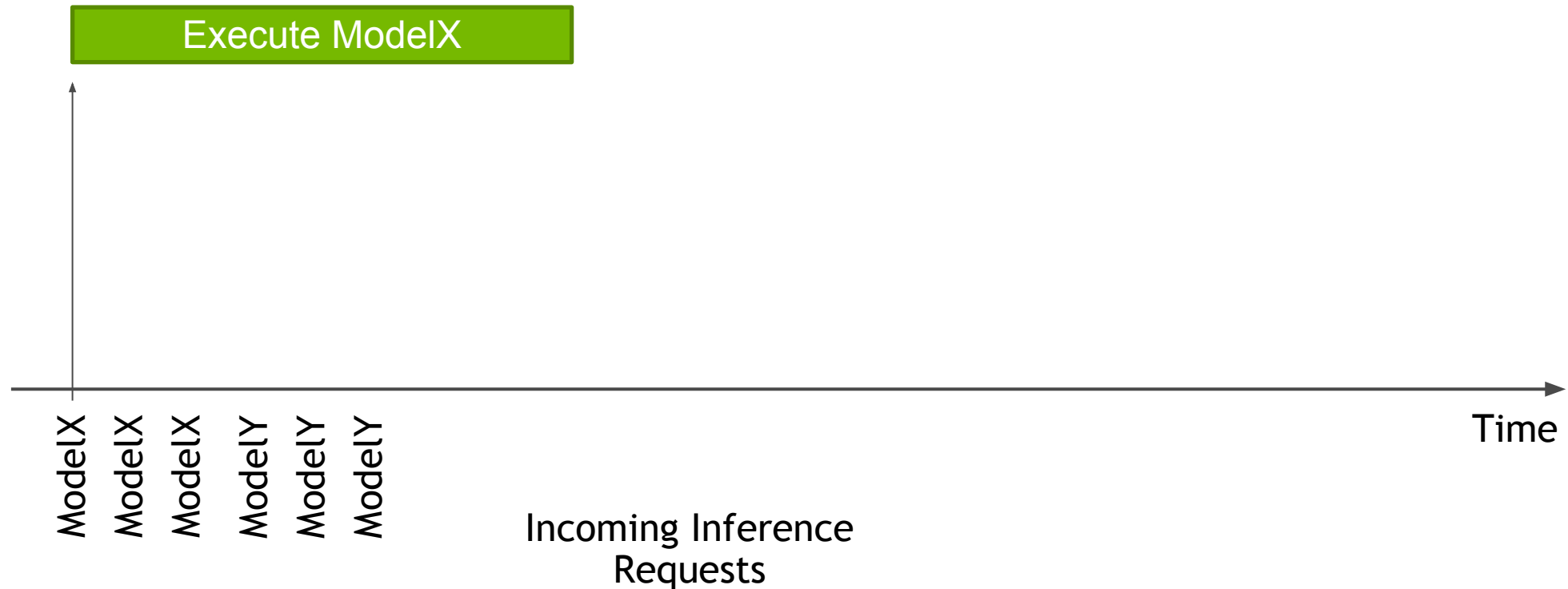
# CONCURRENT EXECUTION TIMELINE

## GPU Activity Over Time



# CONCURRENT EXECUTION TIMELINE

## GPU Activity Over Time



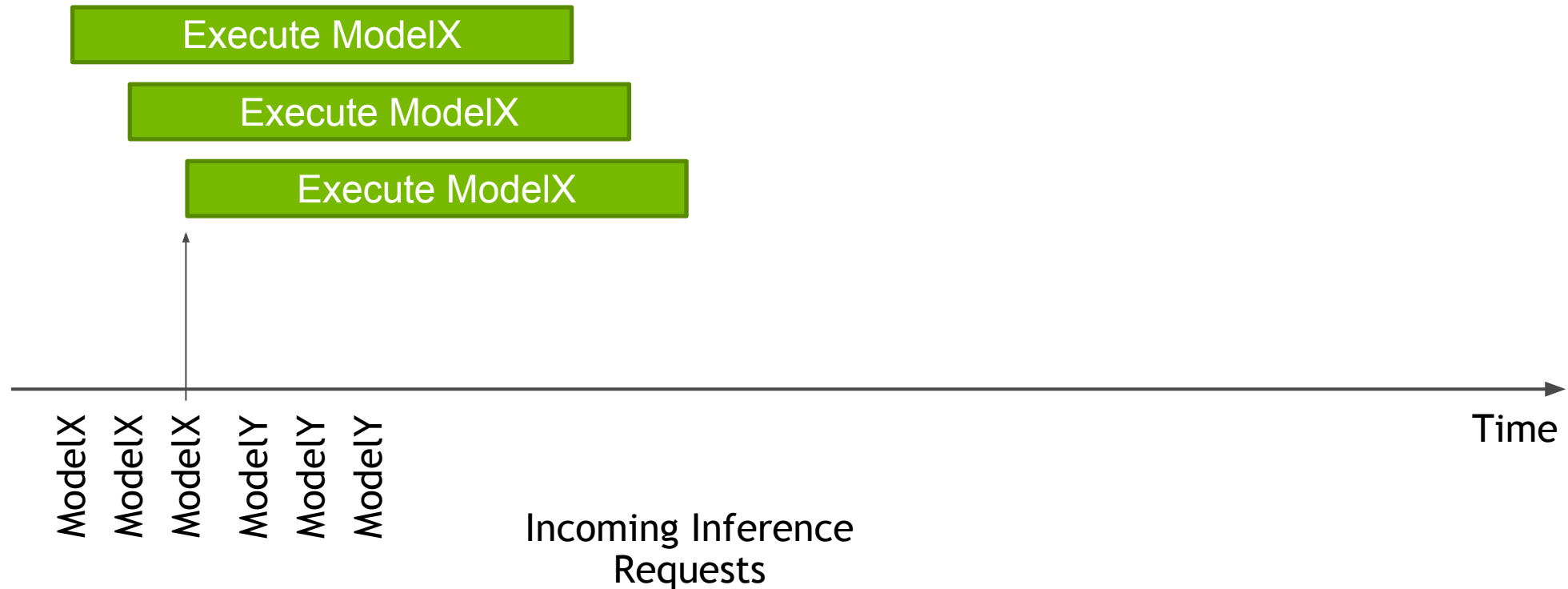
# CONCURRENT EXECUTION TIMELINE

## GPU Activity Over Time



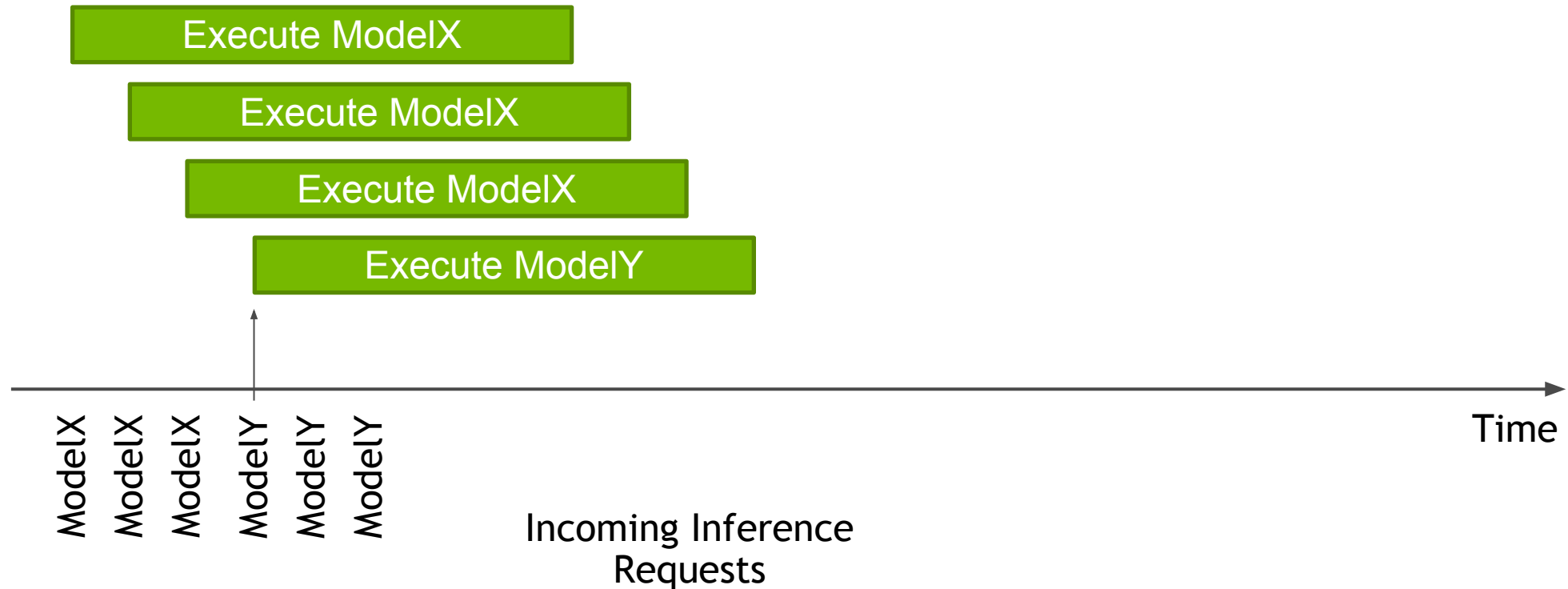
# CONCURRENT EXECUTION TIMELINE

## GPU Activity Over Time



# CONCURRENT EXECUTION TIMELINE

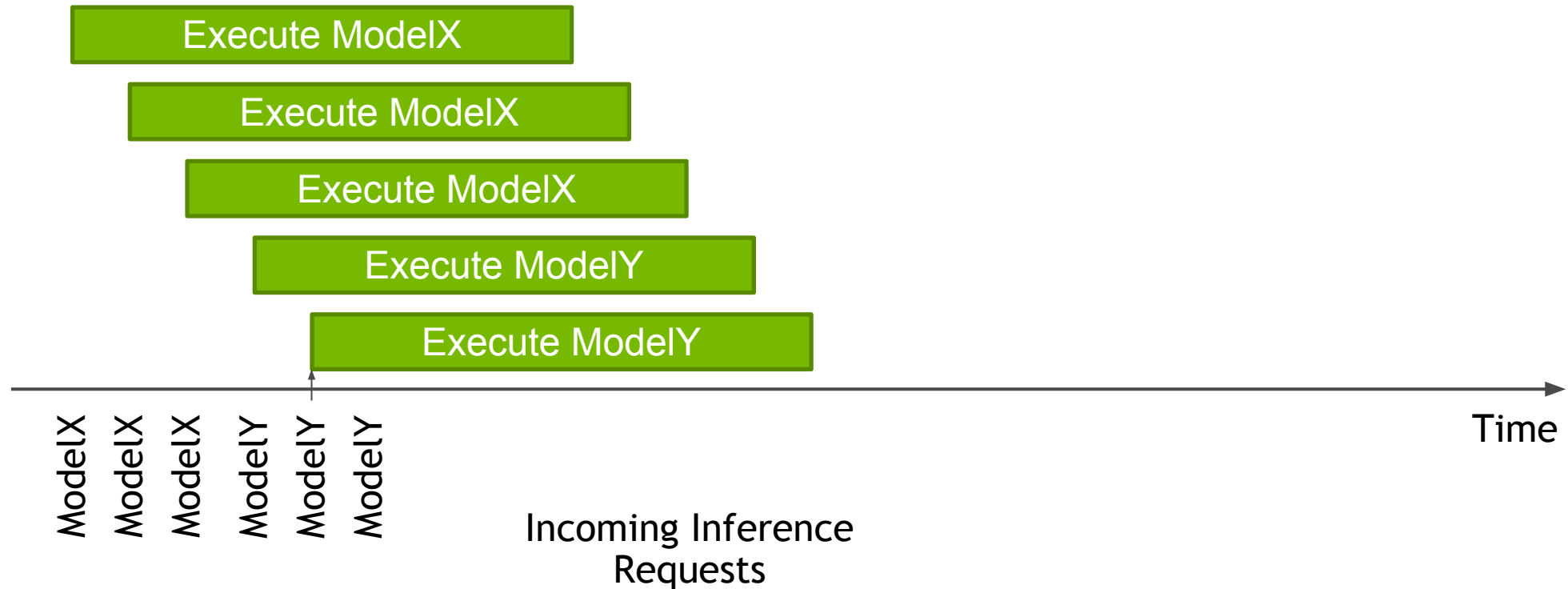
## GPU Activity Over Time





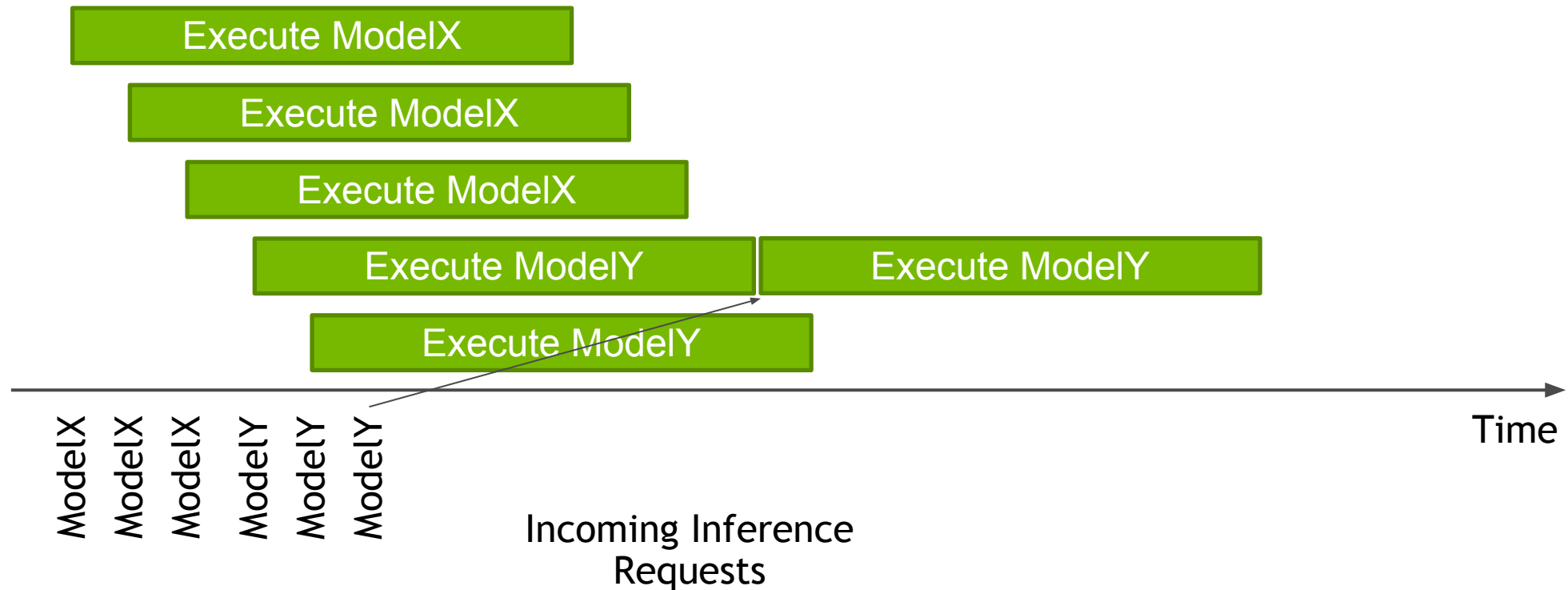
# CONCURRENT EXECUTION TIMELINE

## GPU Activity Over Time



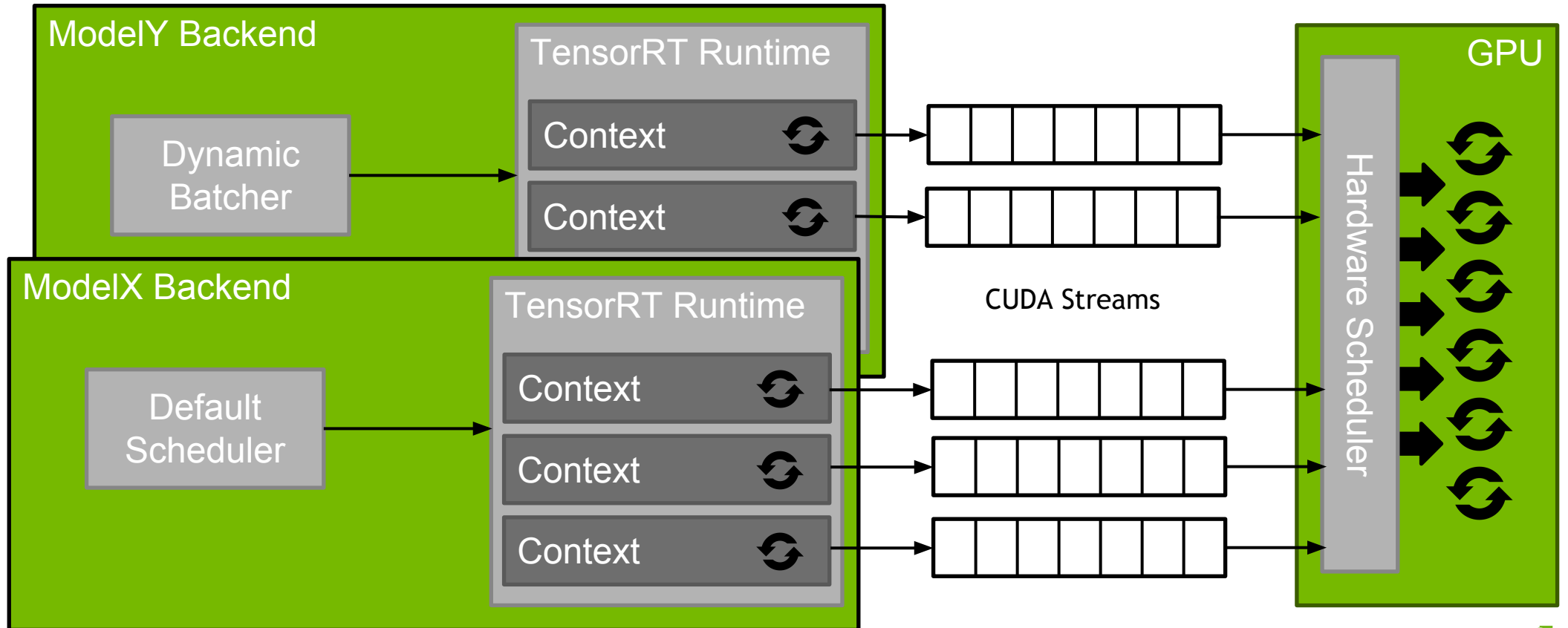
# CONCURRENT EXECUTION TIMELINE

## GPU Activity Over Time



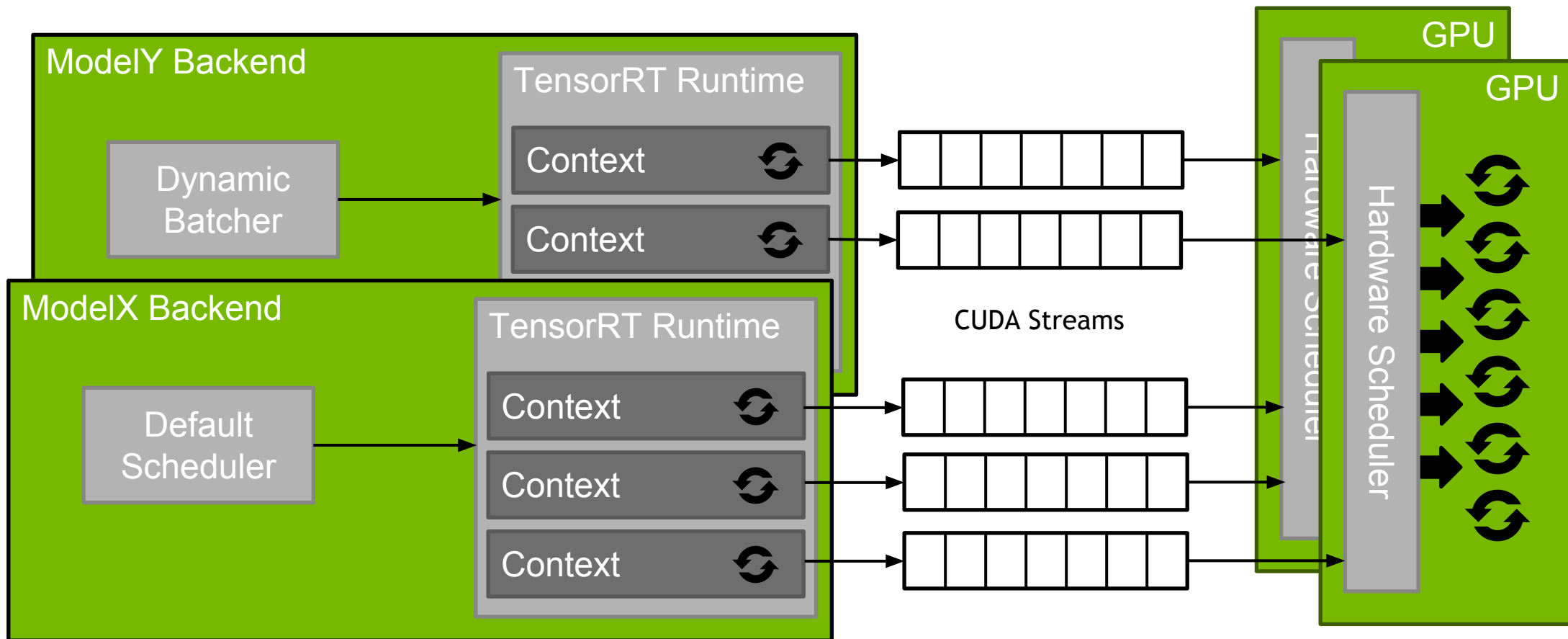
# SHARING A GPU

CUDA Enables Multiple Model Execution on a GPU



# MUTLI-GPU

Execution Contexts Can Target Multiple GPUs



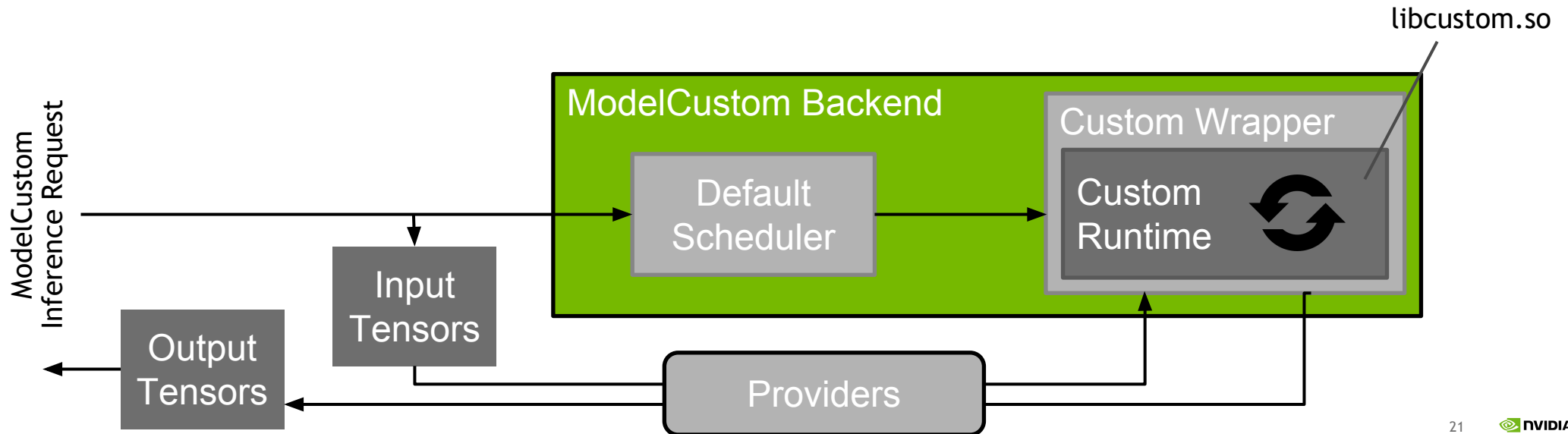
# CUSTOM FRAMEWORK

## 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.

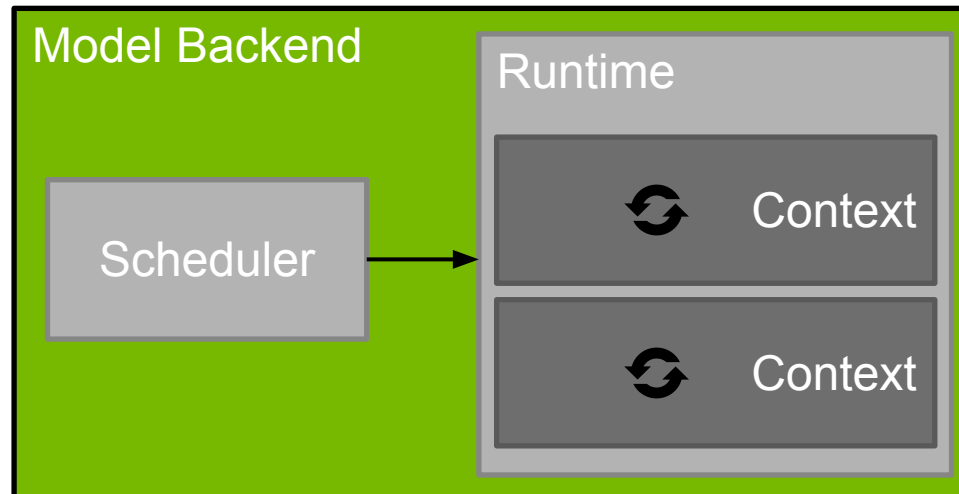


# SCHEDULER ARCHITECTURE

Scheduler responsible for managing all inference requests to a given model

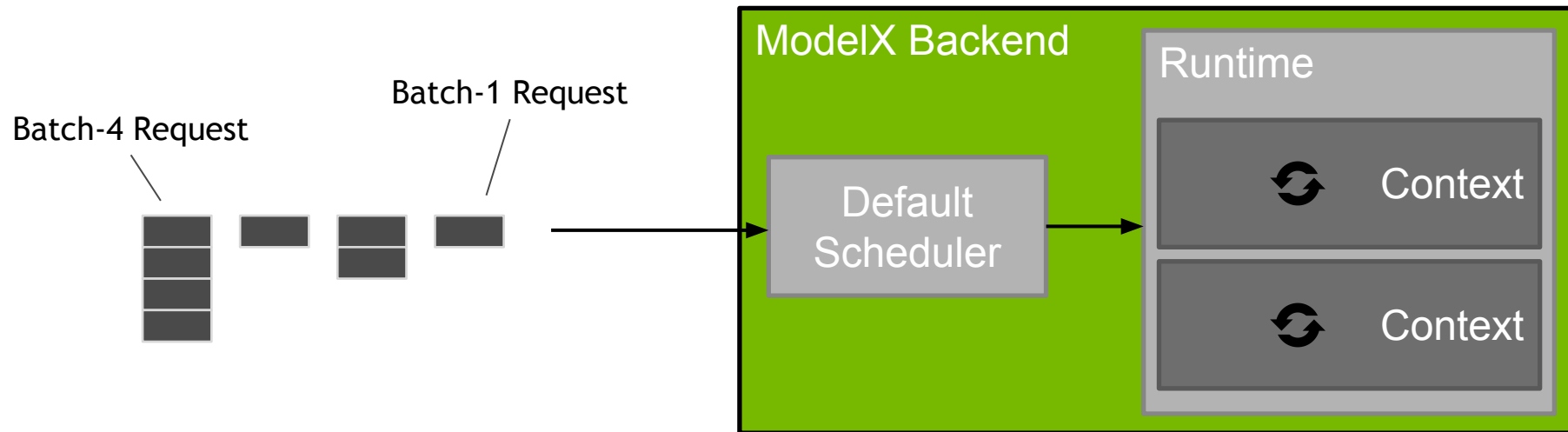
Distribute requests to the available execution contexts

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



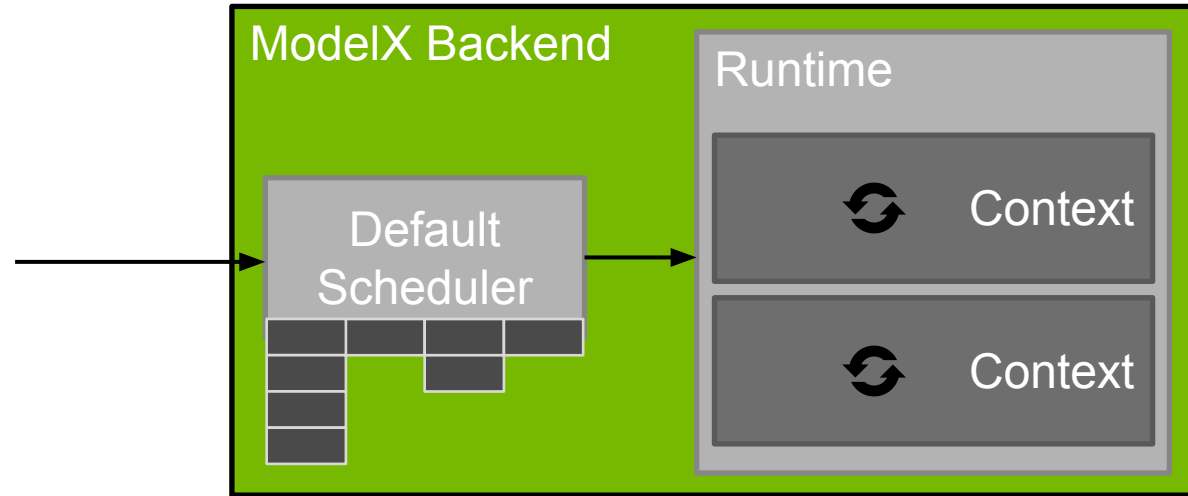
# DEFAULT SCHEDULER

Distribute Individual Requests Across Available Contexts



# DEFAULT SCHEDULER

Distribute Individual Requests Across Available Contexts



Incoming requests to ModelX  
queued in scheduler

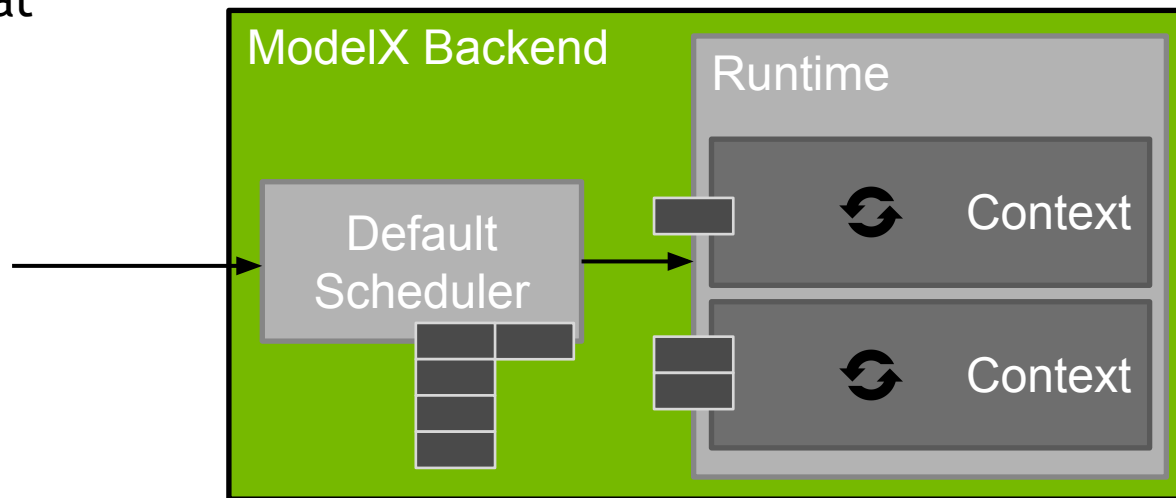


# DEFAULT SCHEDULER

## Distribute Individual Requests Across Available Contexts

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

Utilization =  $3/8 = 37.5\%$



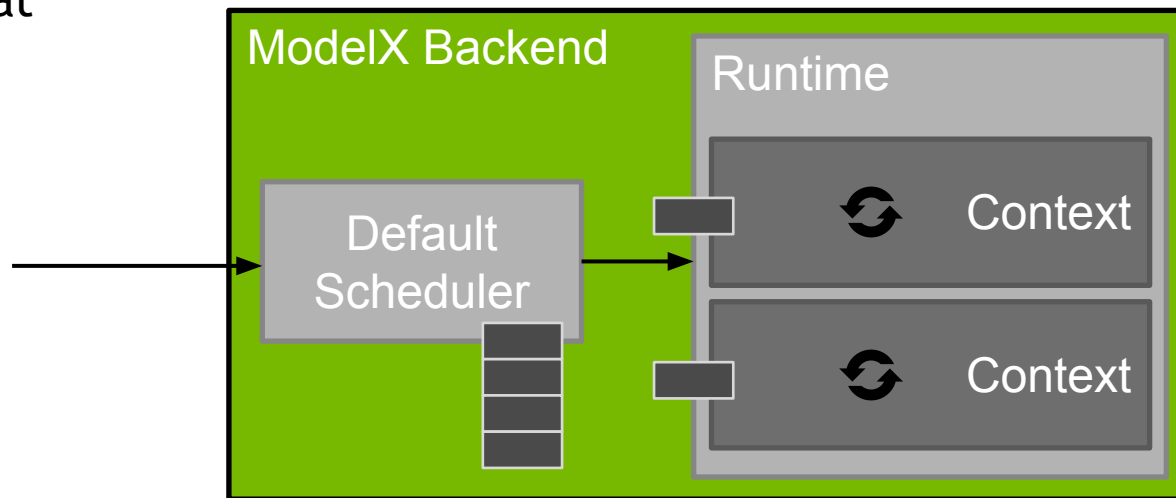
requests assigned in order  
to ready contexts

# DEFAULT SCHEDULER

## Distribute Individual Requests Across Available Contexts

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

Utilization =  $2/8 = 25\%$



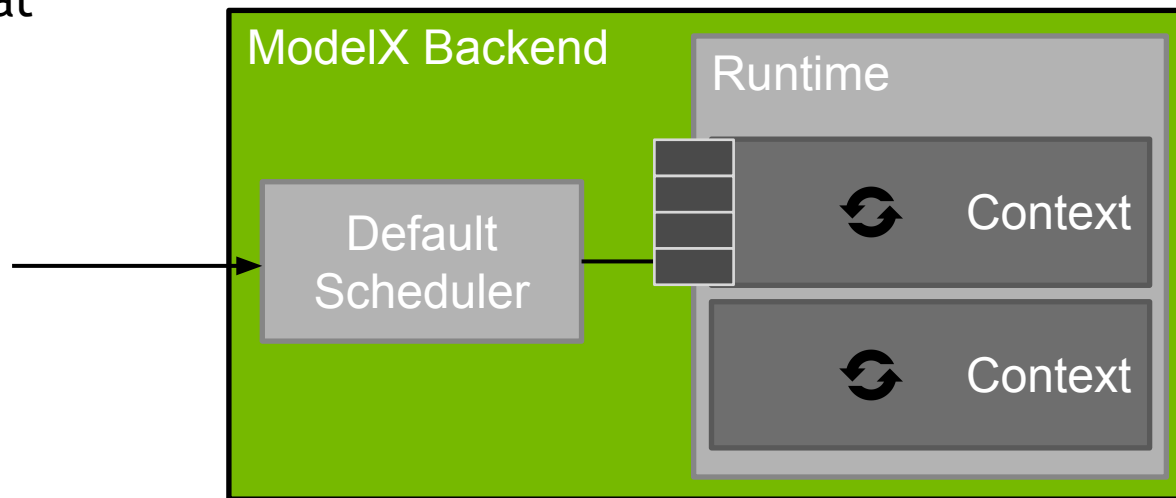
When context completes a new request is assigned

# 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

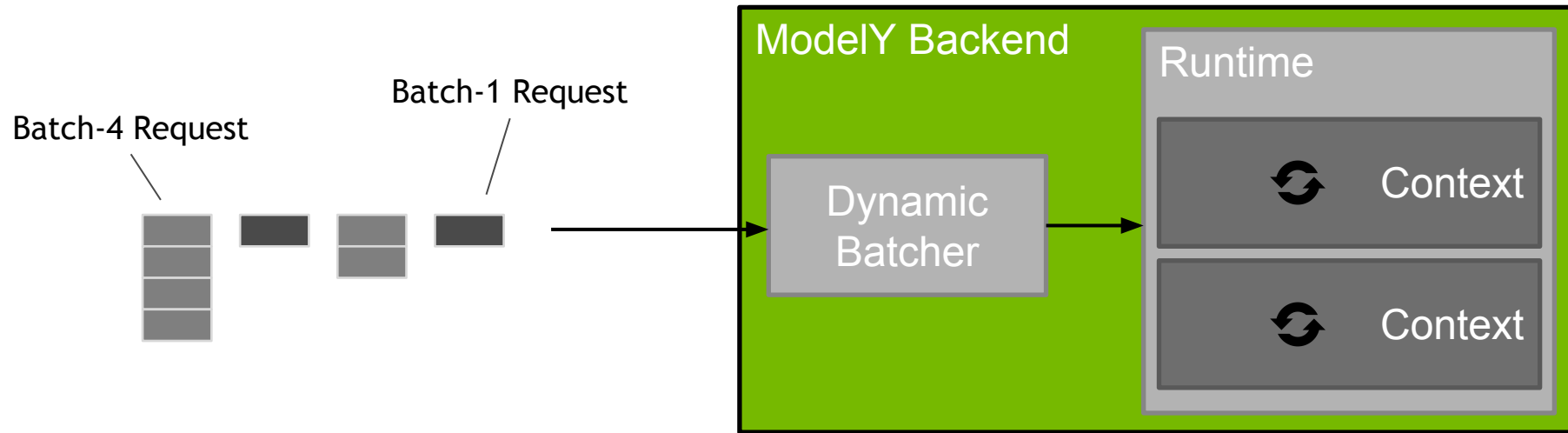
But GPU utilization dependent on the batch-size of the inference request

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

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

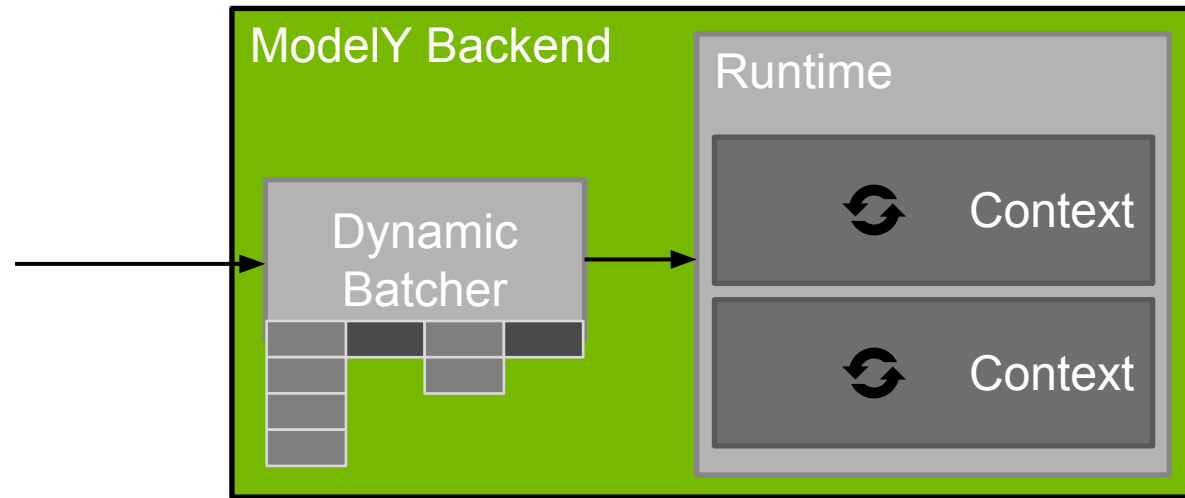
# DYNAMIC BATCHING SCHEDULER

Group Requests To Form Larger Batches, Increase GPU Utilization



# DYNAMIC BATCHING SCHEDULER

Group Requests To Form Larger Batches, Increase GPU Utilization



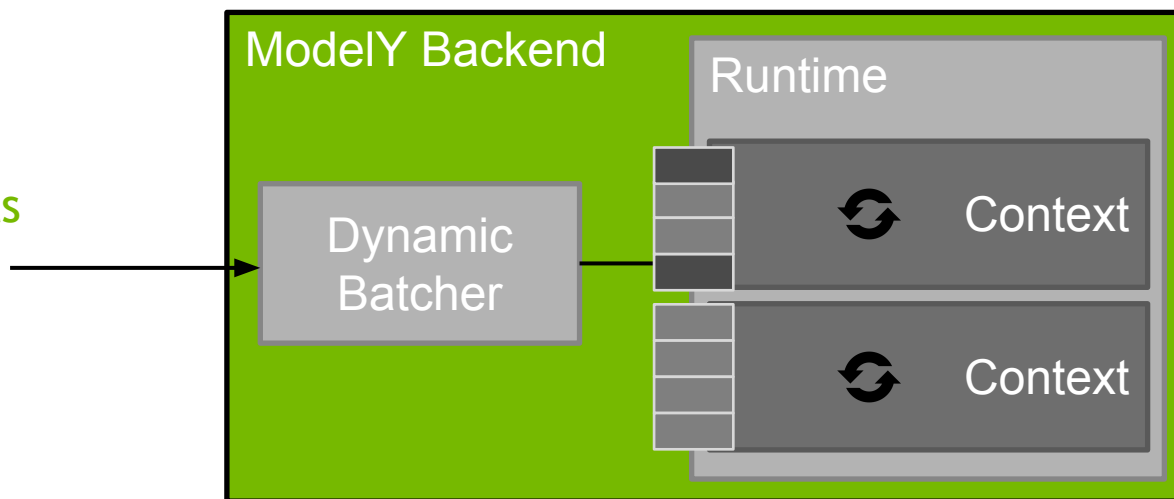
Incoming requests to ModelY  
queued in scheduler

# 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

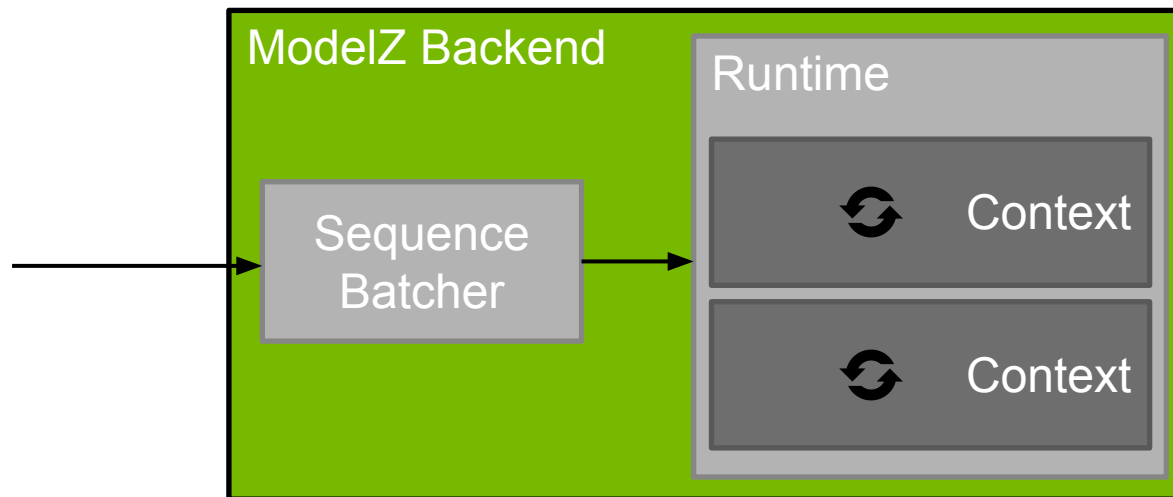
## Dynamic Batching for Stateful Models

Sequence: 3 inference requests

3 2 1

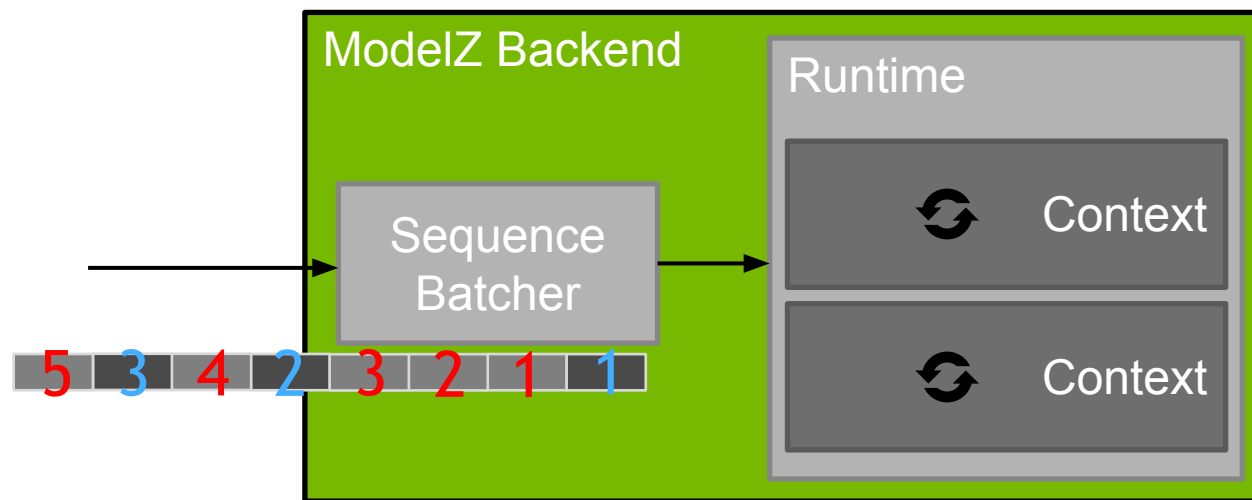
5 4 3 2 1

Sequence: 5 inference requests



# SEQUENCE BATCHING SCHEDULER

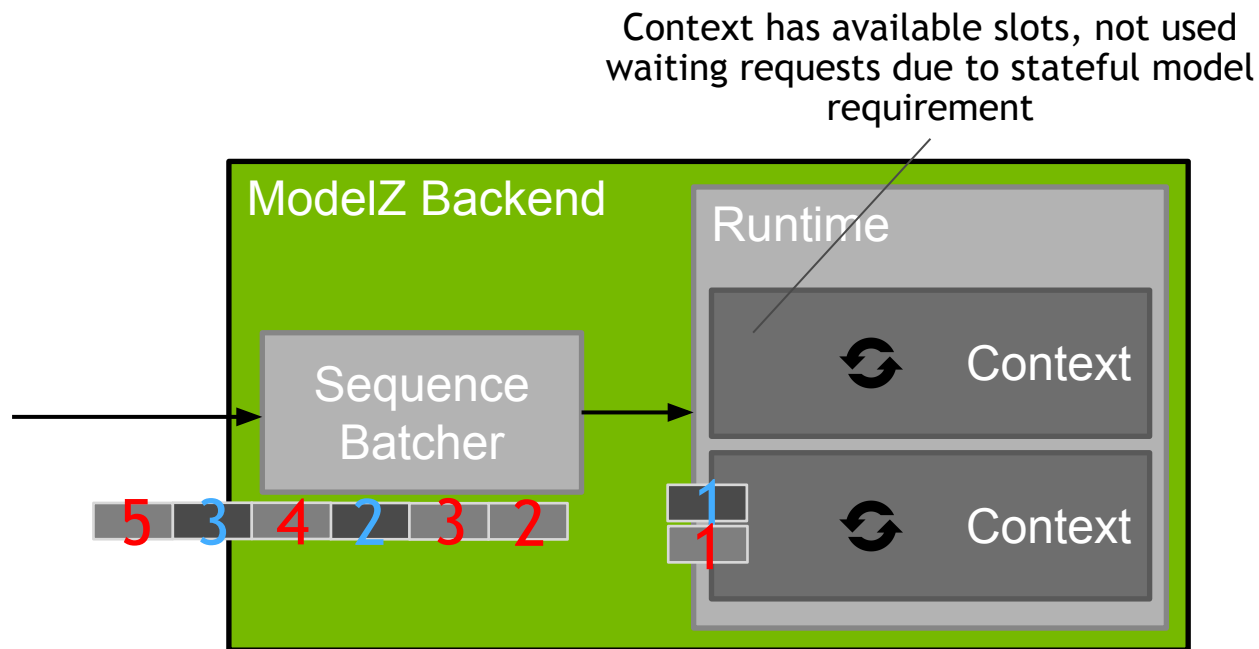
## Dynamic Batching for Stateful Models



Inference requests arrive  
in arbitrary order

# SEQUENCE BATCHING SCHEDULER

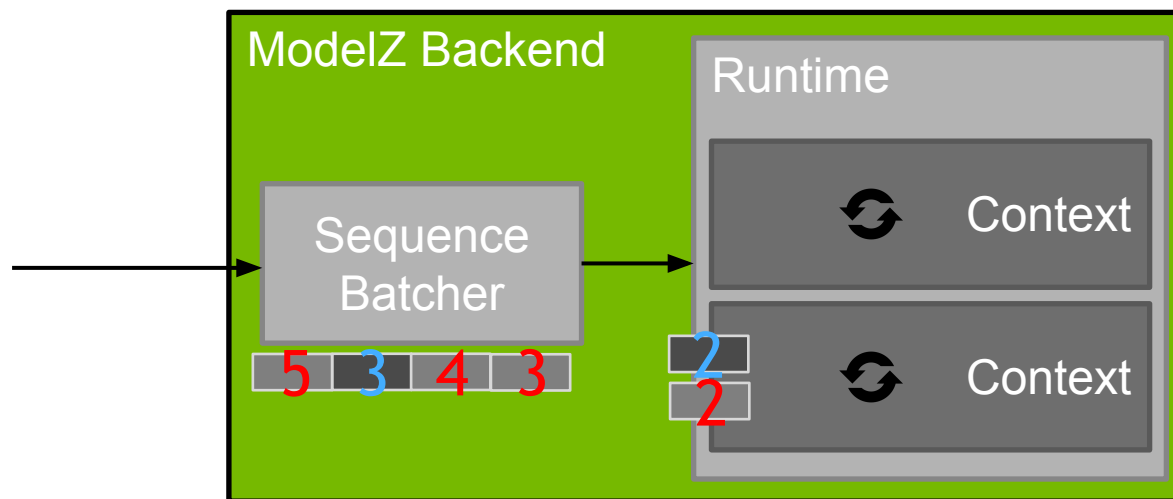
## Dynamic Batching for Stateful Models



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

# SEQUENCE BATCHING SCHEDULER

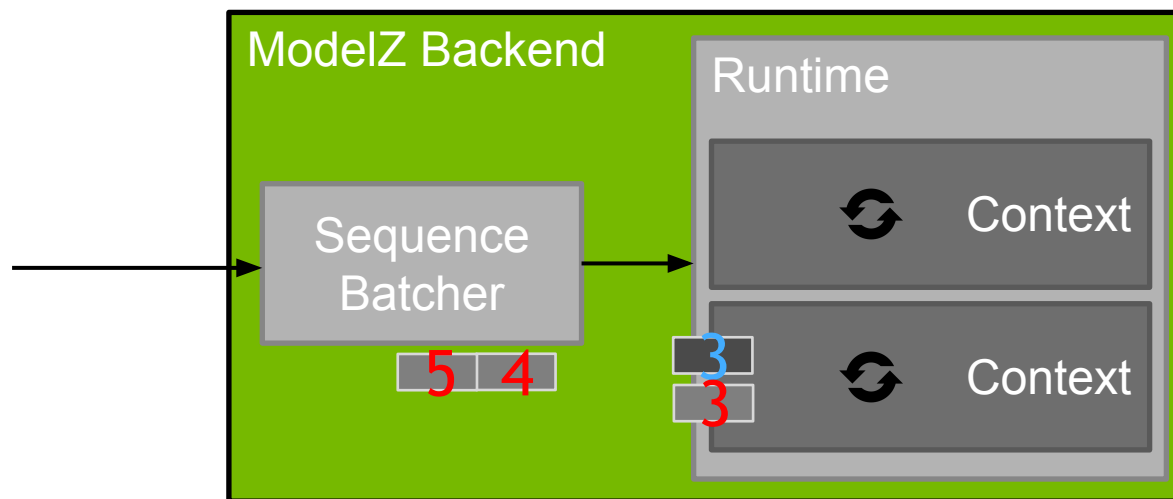
## Dynamic Batching for Stateful Models



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

# SEQUENCE BATCHING SCHEDULER

## Dynamic Batching for Stateful Models



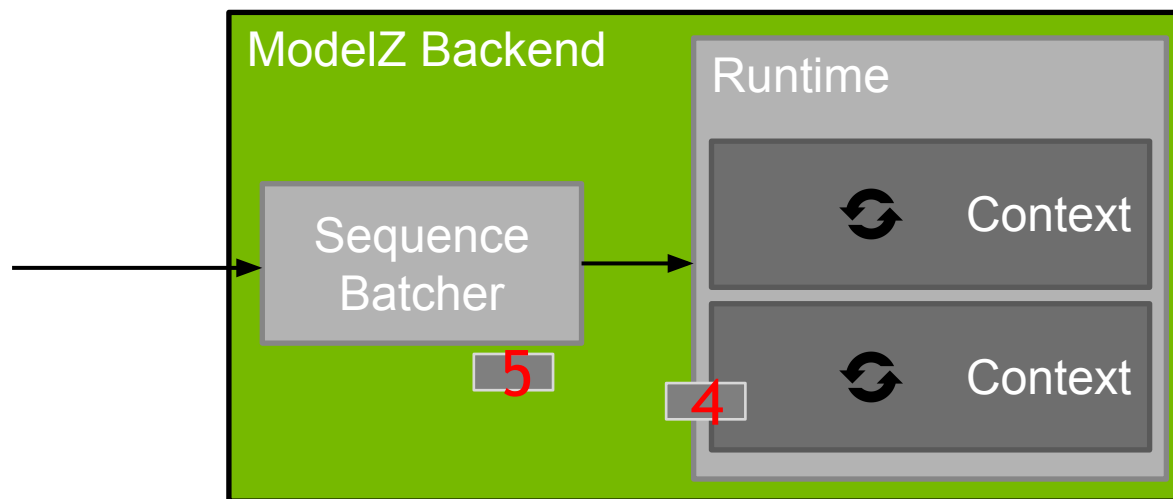
Sequence batcher allocates context slot to sequence and routes all requests to that slot

# 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.



# MAXIMIZING DATA CENTER UTILIZATION WITH TENSORRT INFERENCE SERVER

## Recap

Expand the number of models available to share the GPU

- Support a variety of model frameworks

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

Enable multiple models and multiple instances to execute concurrently on GPU

- Support multi-model and multi-instance via CUDA streams

Enable many model types to exploit large batches which have higher GPU utilization

- Provide scheduling / batching algorithms for both “stateless” and “stateful” models

The background is a dark, almost black, field with a complex network of thin, glowing green lines. These lines intersect at various points, creating a web-like structure. At many of these intersection points, there are small, bright green dots or nodes. Some of these dots are slightly larger and more intense than others. The overall effect is one of a dynamic, interconnected system, possibly representing a network or a data structure. The text 'NAVER USE-CASE' is positioned in the lower right quadrant of the image, in a bold, white, sans-serif font.

**NAVER USE-CASE**



# NAVER

## Korea No. 1 Search Engine & Internet Company

네이버를 시작페이지로 > | [휴먼어택아터](#) | [해피빈](#)



메일 카페 블로그 지식IN 쇼핑 Pay ▶TV 사진 뉴스 증권 부동산 지도 영화 뮤직 책 웹툰 더보기 ▾ 5 김새론 ▾

현대영상다이렉트

손해보험협회 상의할 제8856호(2018.10.24)

**11월내 車 보험료 할인 받는 방법**

32% : 주행거리특약(연간 3천km 이하)

13% : 자녀할인특약(태아, 부부1인한정 기준)

11.5% : 무사고(3년 이상)

4.7% : 교통법규 준수(무사고 기준)

**지금 확인**

연합뉴스 > "각성상태서 전처 폭행"...양진호, 가정폭력 의혹도 네이버뉴스 연예 스포츠 경제

뉴스스탠드 > 전체 언론사 | MY 뉴스

동아일보	MBN	OSEN	연합뉴스TV	NEWSIS	중앙일보
BLOTER	KBS	mydaily	스포츠서울	이데일리	한국일보
NEWS 24	소년한국일보	SPOTV NEWS	소비자가 만드는 신문	스포츠한국	Able news

여행+ 디자인 경제M JOB& 과학 중국 비즈니스 FARM 스쿨캠 공연전시 < >

Connect with people

**NAVER Sign In**

Forgot Username or Password? Sign up


11.08. (목) | 스포츠 3/6 < >

**배구** [보이는 V토크쇼] 1라운드 숨은 MVP는?  
**야구** 비룡의 날개 꺾을 뻔한 정의윤의 아찔한 신평위...  
**해외축구** PSG도 거부한 음바페 요구, '네이마르 이상...

00:13 01:00

쇼핑 상품 쇼핑물 MEN


12:05 72%




뉴스 연예 스포츠 자동차 여행+ 비즈니스 +

"미세먼지, 재난상황 준비 총력대응"...경유차 인센티브 폐지  
여야 '기무사 계엄문건' 국방위 청문회 실시키로  
美선거發 증시 혼풍...다음 가능자는 '시진핑'  
자율주행차용 운전면허 신설...운전허용 대상도 대폭 확대  
최근 고농도 초미세먼지 원인, 국내 요인이 더 컸다

**금상송검색어** 2 예방접종도우미사이트 ▾

 암, 뇌졸중, 급성심근경색 동시 보장 (특약)  
**암 보험, 하나면 된다** 10초 보험료 계산 >  
암보험료(1인) 2018.10.15 기준

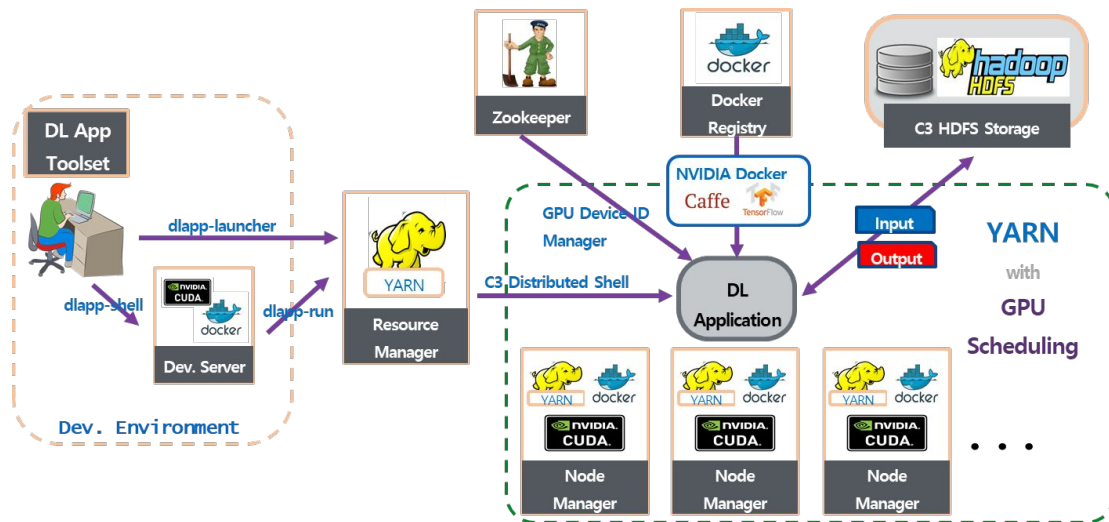
 < > < >

# DATA ENGINEERING PLATFORM



# C3DL PLATFORM

Since 2016



YARN-based DL platform for Search Division's DL R&D

CPU / GPU scheduler based on YARN (<https://github.com/naver/hadoop>)

Both training/inference supported

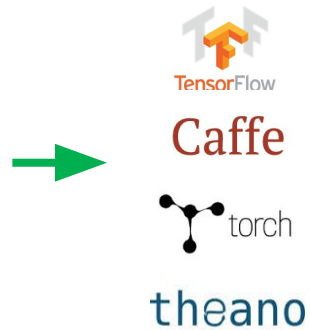
# WHY TRTIS IN C3DL?

Can be used for several types of Inference Services

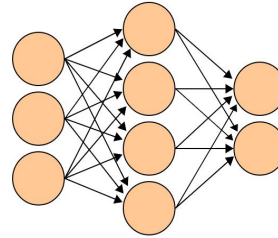
**Datasets**



**Training**



**Model**



**Inference**



Batc  
h



Serving



Streaming

**Service**



# WHY TRTIS FOR C3DL?

## Optimized for Large-Scale Inference Service

Supports HTTP / gRPC

Each Data Handling with numpy-like format

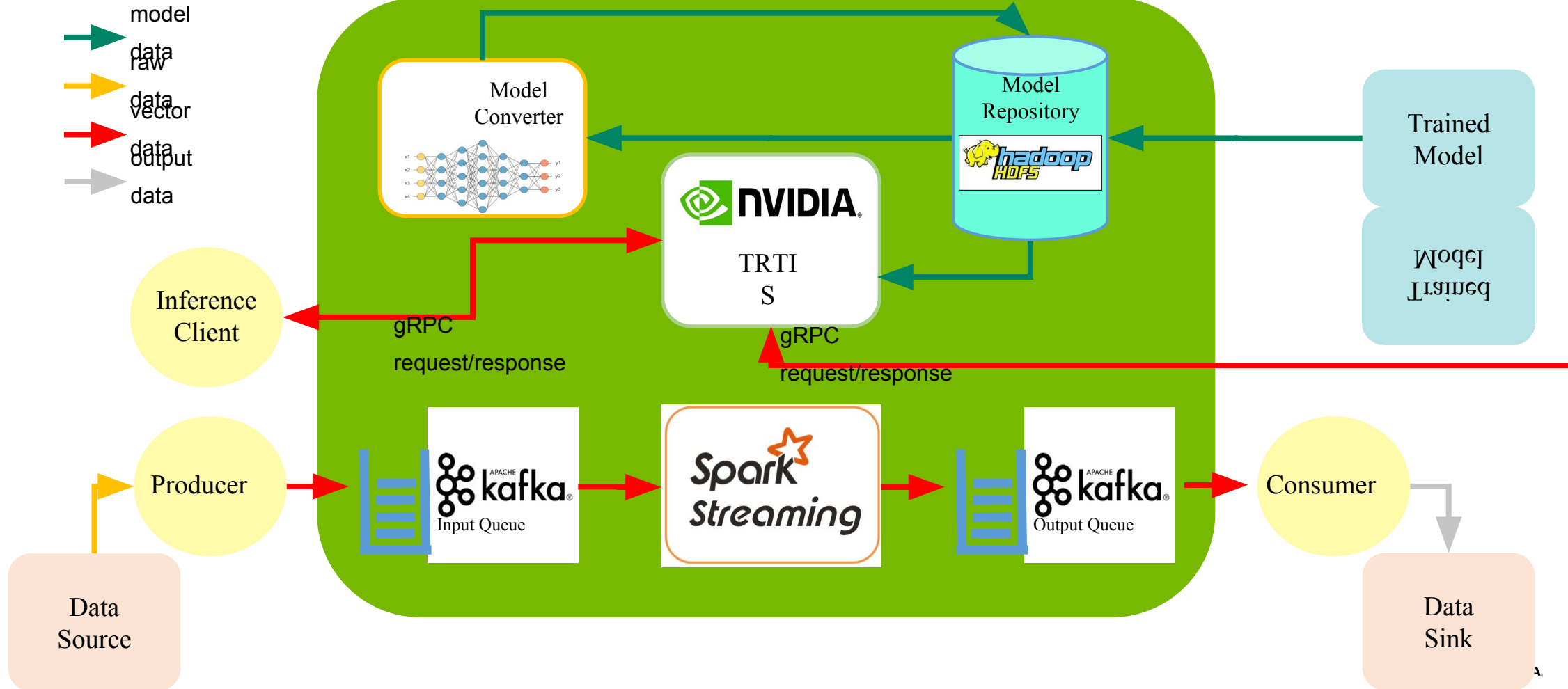
Dynamic Model Deployment with Model Store

Optimized for Container-based Provisioning

Multi-model / Multi-GPU supported

Multi Framework supported

# C3 DL INFERENCE



# FUTURE PLANS

More Use cases with TRTIS

More Inference on GPUs: Image as well as Text-based

More cost-efficient Inference : T4 adoption

More Collaboration with NVIDIA: Applying TRT for more Models

# MAXIMIZE GPU UTILIZATION WITH TENSORRT INFERENCE SERVER

Try It Today!

The TensorRT Inference Server is available as a ready-to-run Docker image on the NVIDIA Compute Cloud. <https://ngc.nvidia.com/catalog/containers/nvidia:tensorrtserver>

The TensorRT Inference Server is open-source. Read the docs, build the source, file issues, contribute pull requests! <https://github.com/NVIDIA/tensorrt-inference-server>

Questions, feedback?

Connect with the Experts: NVIDIA TensorRT Inference Server

Wednesday, 3/20/19 | 12:00 - 13:00 - SJCC Hall 3 Pod D (Concourse Level)





**nVIDIA®**