AWS re:Invent

NOV. 27 – DEC. 1, 2023 | LAS VEGAS, NV

AIM209-S

SPONSORED BY NVIDIA

Simplifying the adoption of generative AI for enterprises

Jiahong Liu
Senior Solution Architect
NVIDIA

Peter Dykas
Senior Solution Architect
NVIDIA

Speakers

Jiahong Liu

Peter Dykas

Agenda

01 NVIDIA and AWS collaboration

02 NVIDIA AI on AWS

03 At-scale training on AWS

04 Deployment with Triton

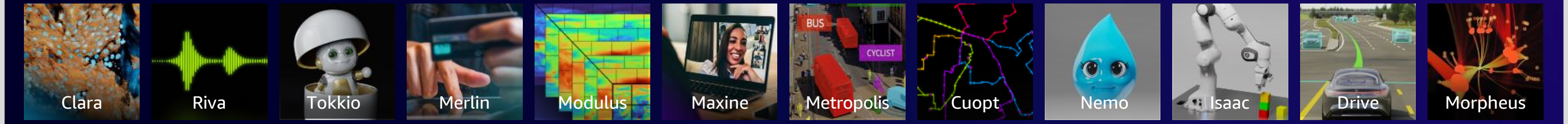
05 Acceleration with TensorRT

06 Call to action and conclusion

NVIDIA AI

END-TO-END OPEN PLATFORM FOR PRODUCTION AI

Application Workflows



Clara	Riva	Tokkio	Merlin	Modulus	Maxine	Metropolis	Cuopt	Nemo	Isaac	Drive	Morpheus
Medical imaging	Speech AI	Customer service	Recommenders	Physics Ml	Video	Video Analytics	Logistics	Conversational AI	Robotics	Autonomous Vehicles	Cybersecurity

NVIDIA AI Enterprise

- AI and data science development and deployment tools
- Cloud-native management and orchestration
- Infrastructure optimization

Accelerated infrastructure

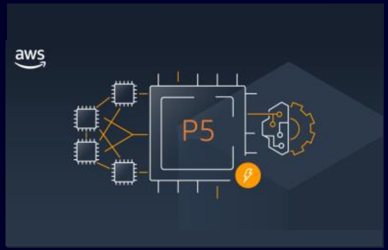
Cloud | Data center | Edge | Embedded

NVIDIA Launchpad

Hands-on labs

NVIDIA and AWS collaboration

GPU power from the cloud to the edge



The highest performance instance for ML training and HPC applications powered by NVIDIA H100 GPUs

High-performance instances for graphics-intensive applications and ML inference powered by NVIDIA A10G GPUs

The best price performance in Amazon EC2 for graphics workloads powered by NVIDIA T4G GPUs

Deploy fast and scalable AI with NVIDIA Triton Inference Server in Amazon SageMaker

Improve your operations with computer vision at the edge powered by NVIDIA Jetson

Spot defects with automated quality inspection powered by NVIDIA Jetson

NVIDIA GPU-optimized software available for free in the AWS Marketplace

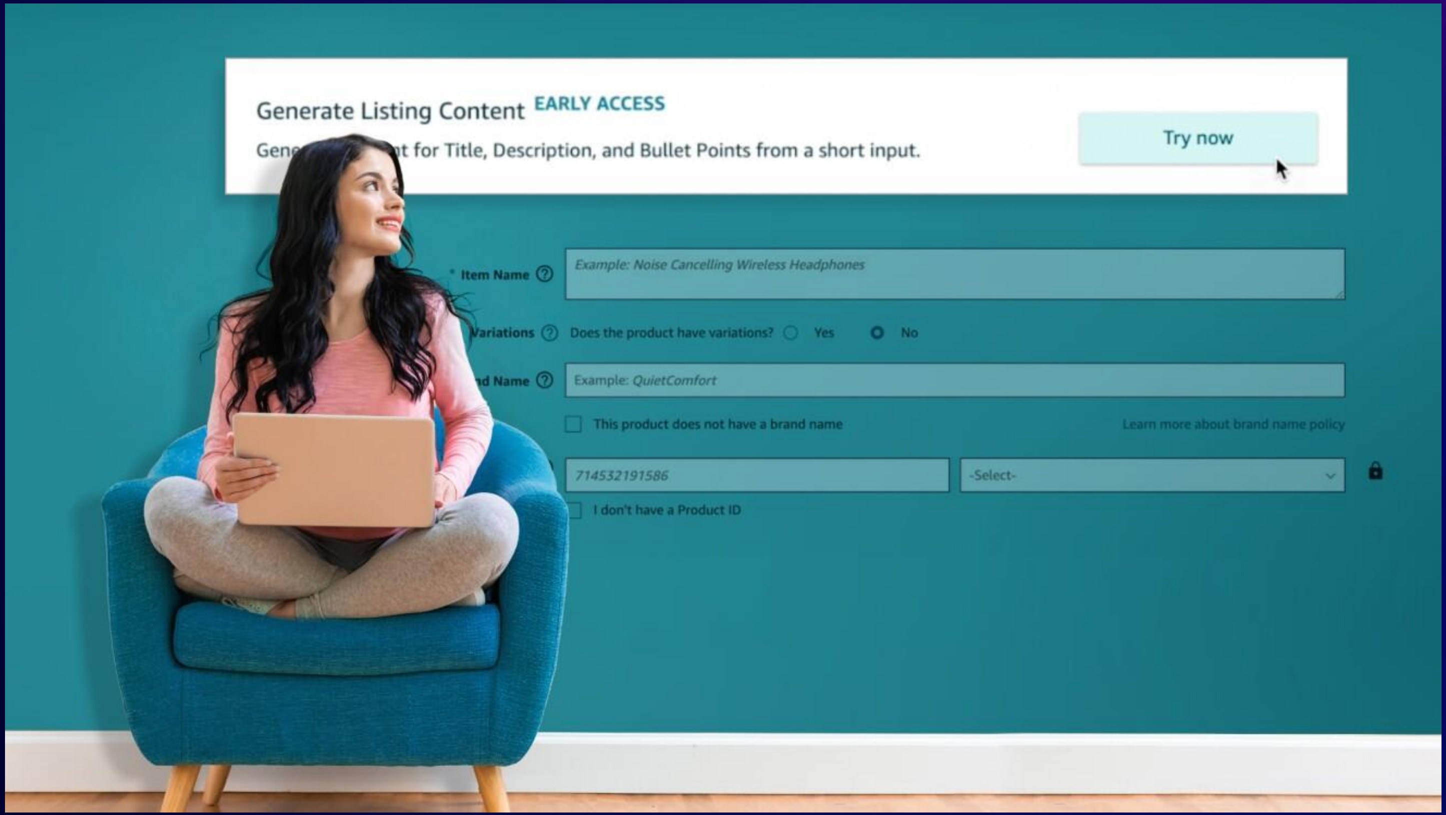
Amazon Catalog

Automatically generate product listings for sellers

Using NVIDIA Tensor Core GPUs, NVIDIA Triton Inference Server, and NVIDIA TensorRT, the Amazon Catalog team achieved

- 50% CO2 reduction
- 3x latency improvement
- 2x throughput improvement

toward generative AI for the product listing program

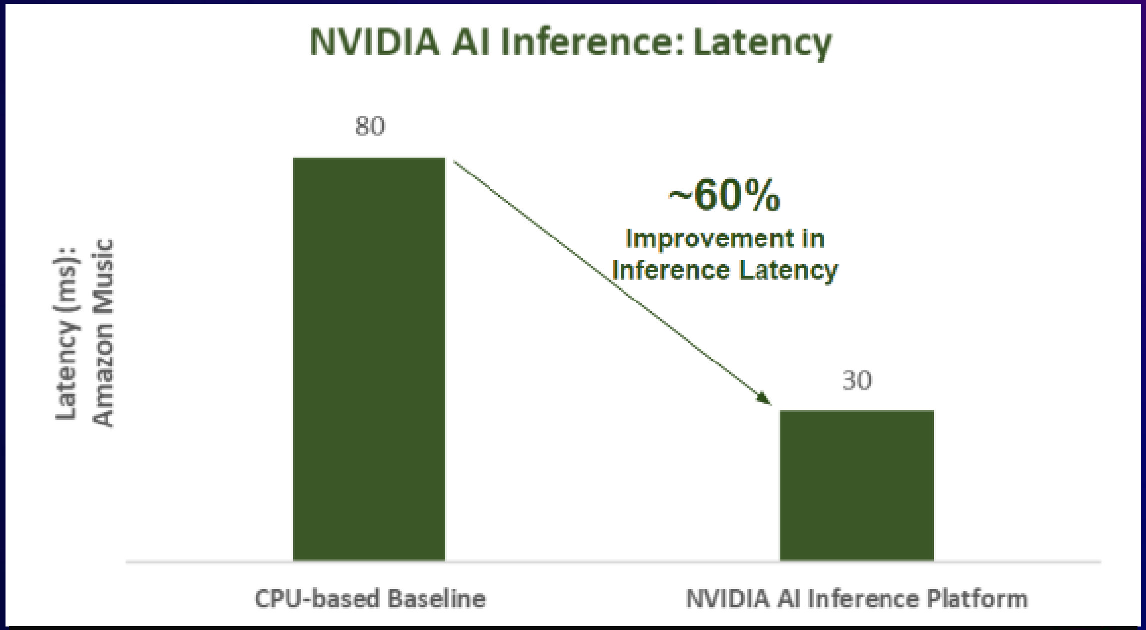


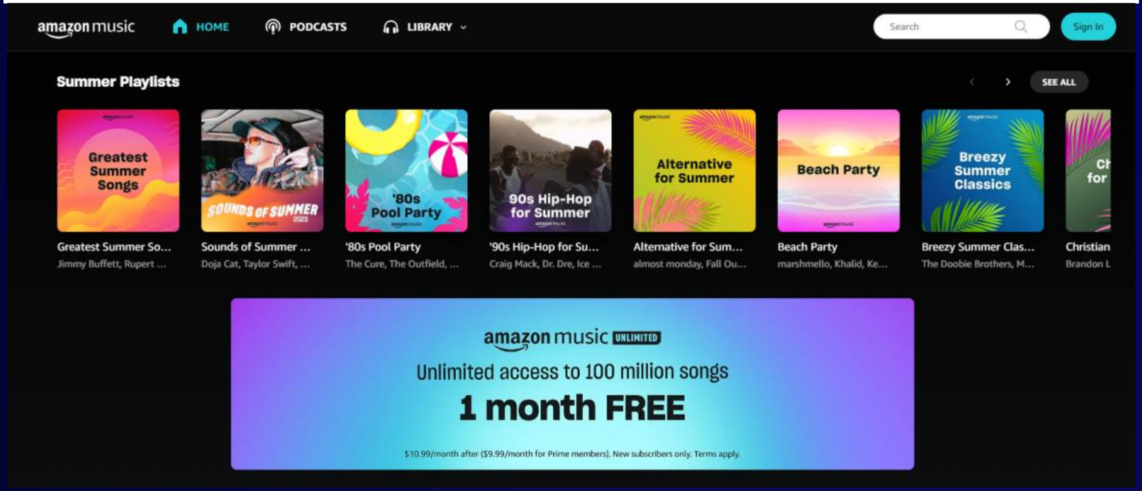
Amazon Music

Corrects misspelled words in music search

Using NVIDIA Tensor Core GPUs, NVIDIA Triton Inference Server, and NVIDIA TensorRT, Amazon Music achieved:

- 12x reduction in training time by optimizing GPU utilization
- 63% reduction in inference latency vs. CPUs
- 73% reduction in inference cost vs. CPUs
- 25 ms end-to-end latency at peak traffic





Amazon Search

Real-time spell check for product search

- One of the most visited e-commerce websites
- Deep learning (DL) AI model for automatic spell correction to search effortlessly
- Triton and TensorRT meets sub-50 ms latency target and delivers 5X throughput for DL model on GPUs on AWS
- Triton Model Analyzer reduced time to find optimal configuration from weeks to hours

https://aws.amazon.com/blogs/machine-learning/how-amazon-search-achieves-low-latency-high-throughput-t5-inference-with-nvidia-triton-on-aws/

NVIDIA AI on AWS

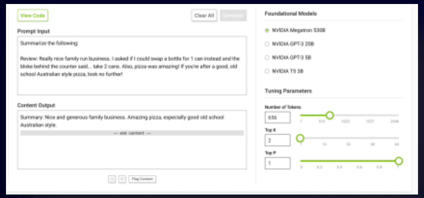
NGC

PORTAL TO AI SERVICES, SOFTWARE, AND SUPPORT

NGC Catalog

Cloud services

Tested across GPU-accelerated platforms

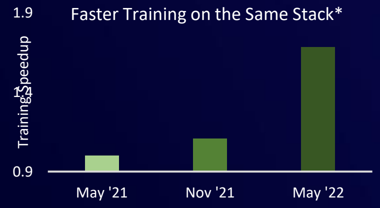


AI Services for NLP, biology, speech

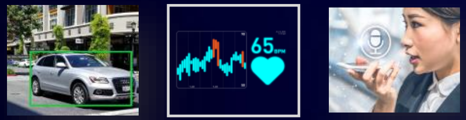
AI workflow management and support

Performance-optimized

End-to-end AI development



Monthly sw container updates



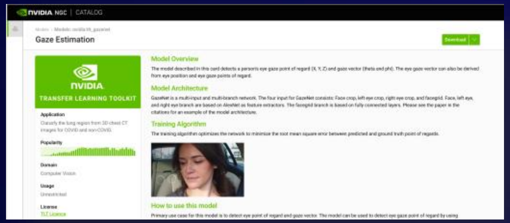
SOTA models

Fully transparent

Quickly find/deploy the right software

vulnerabilities	OS package	Medium	(CVE-2021-3995) libmount1
vulnerabilities	OS package	Medium	(CVE-2021-3995) fdisk
vulnerabilities	OS package	Medium	(CVE-2019-9152) hdf5-helpers
vulnerabilities	OS package	Medium	(CVE-2018-17233) hdf5-helpers

Detailed security scan reports



Model resumes

Accelerates development

Focus on building, not setup

One-click deploy from NGC

Develop once; deploy anywhere with NVIDIA VMI

ngc.nvidia.com

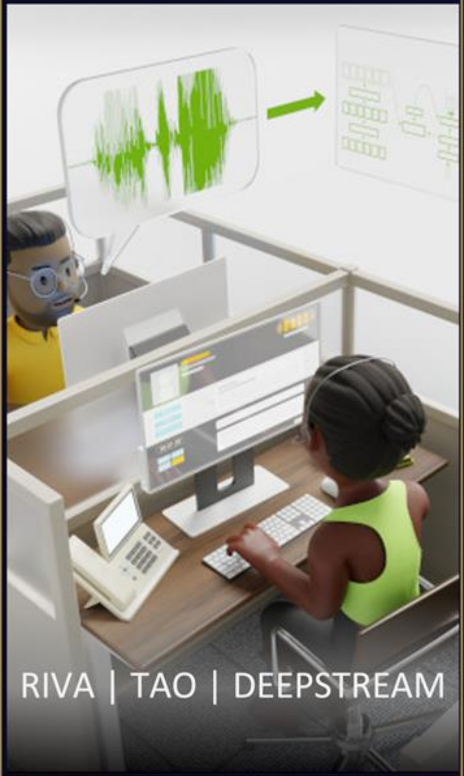
Accelerating the next wave of AI

AI PLATFORM UPDATES

DATA SCIENCE
Analytics, ML, Visualization

RECOMMENDER
Personalization Engine, Simplified

SPEECH & VISION
Conv AI, Video Analytics

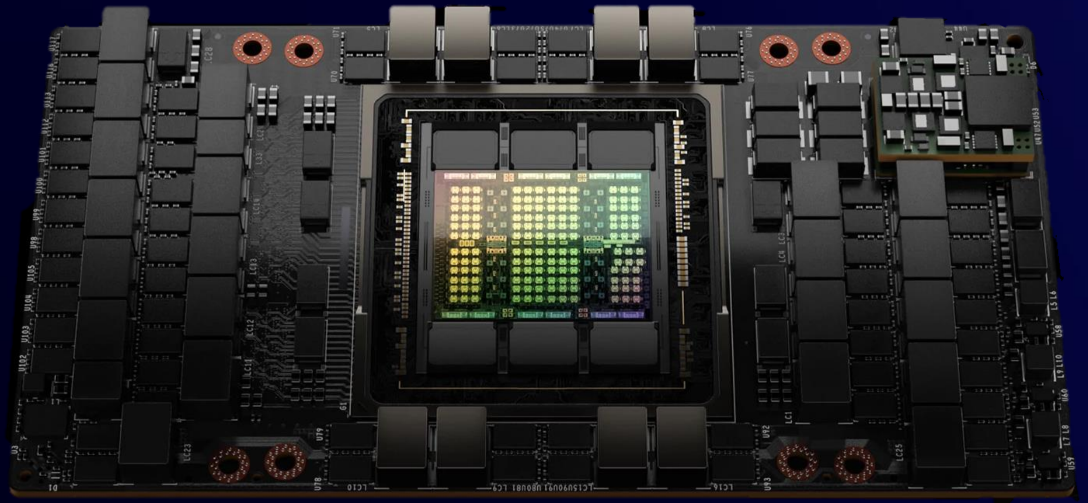


INFERENCE
Fast, Scalable Predictions

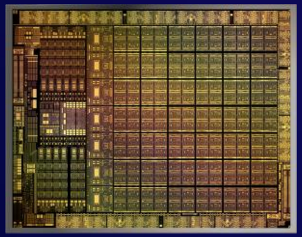
NLP
Large Language Model

NVIDIA H100

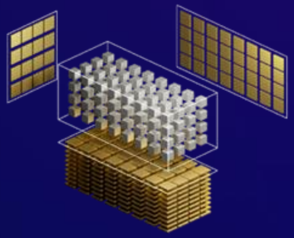
THE NEW ENGINE OF THE WORLD’S AI INFRASTRUCTURE



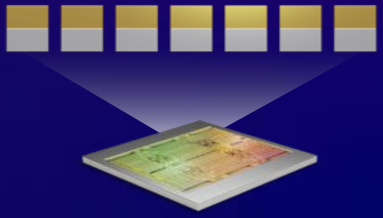
Powering next generation of GPU systems in AWS



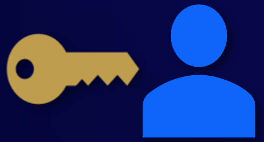
World’s most advanced chip



Transformer engine

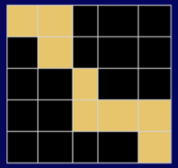


2nd-gen MIG



Confidential computing

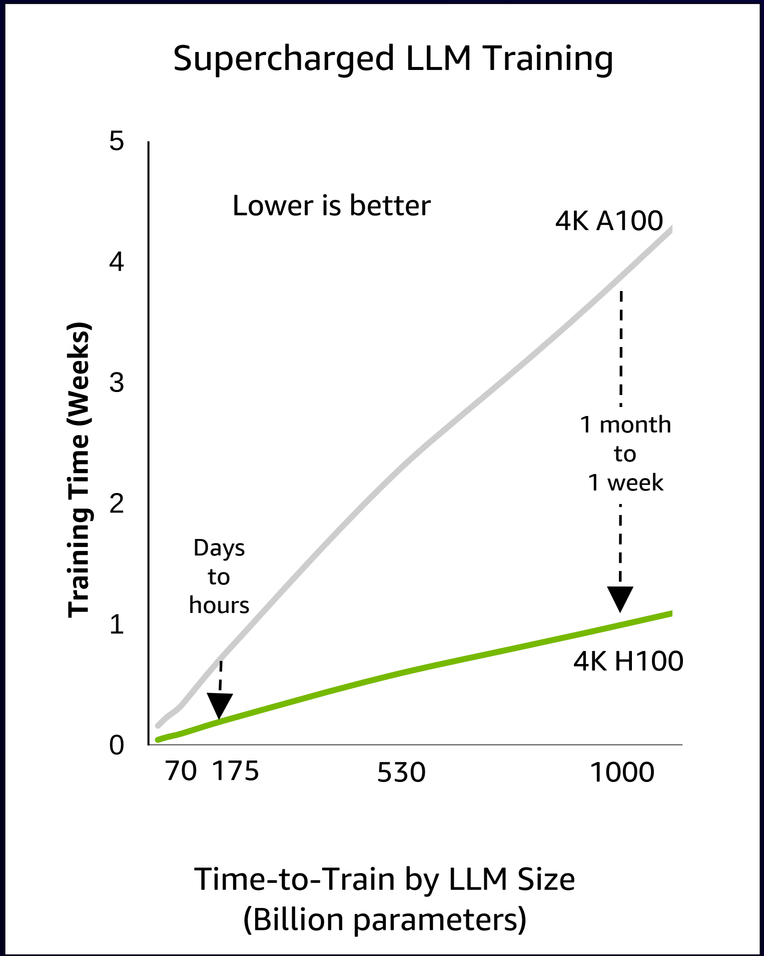
4th-gen NVLink

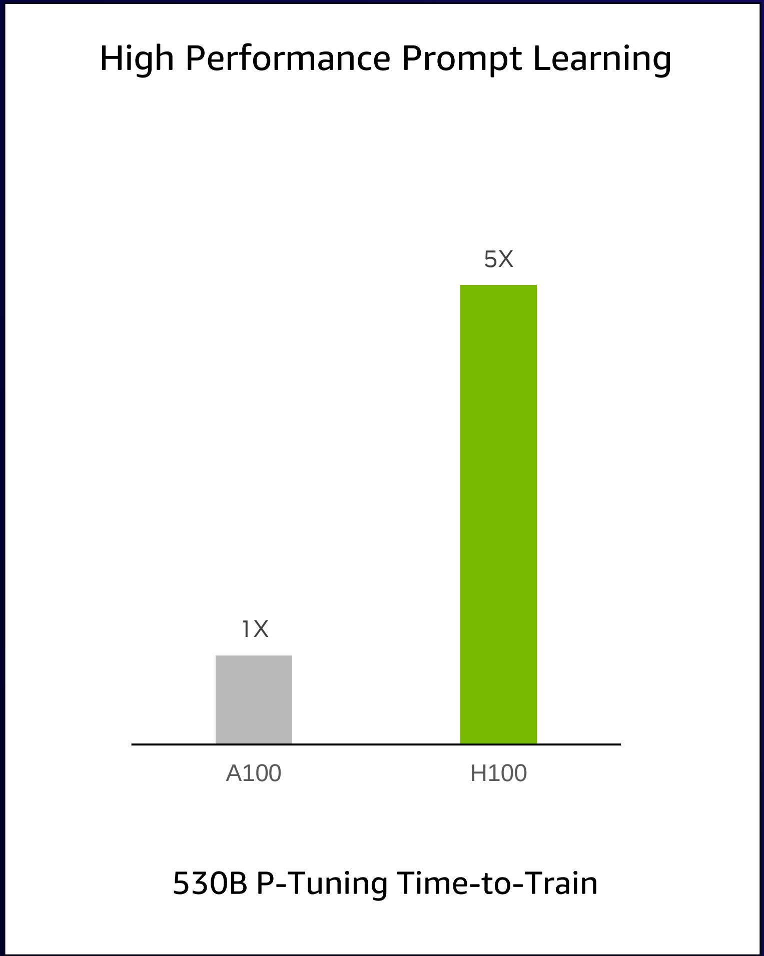


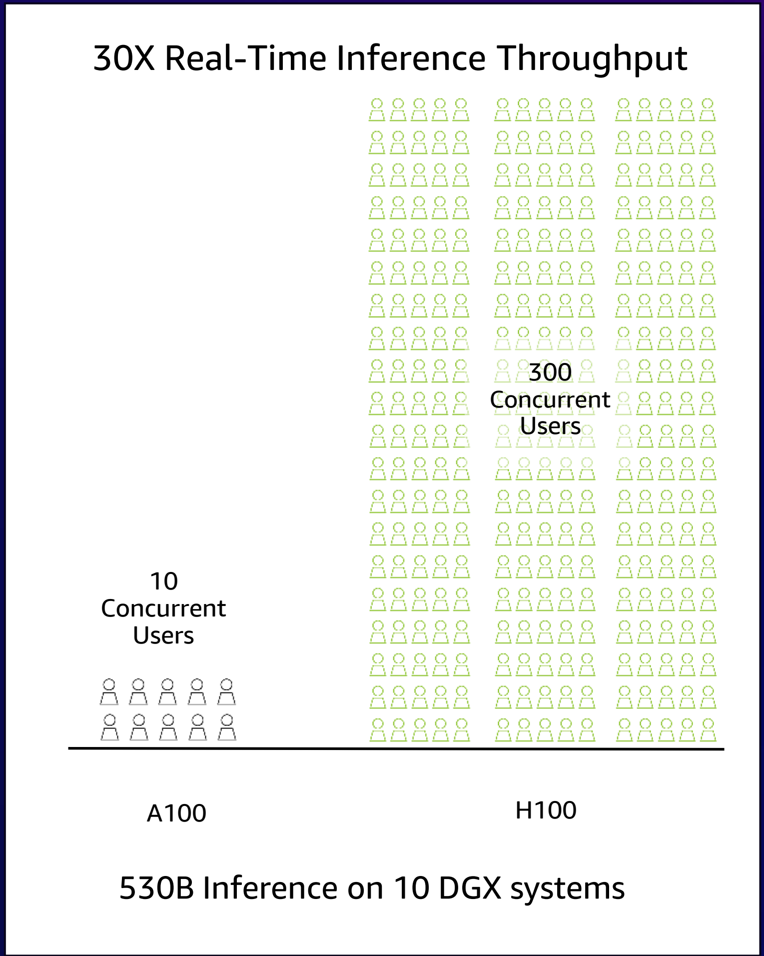
DPX instructions

NVIDIA H100 supercharges LLMs

HOPPER ARCHITECTURE ADDRESSES LLM NEEDS AT SCALE





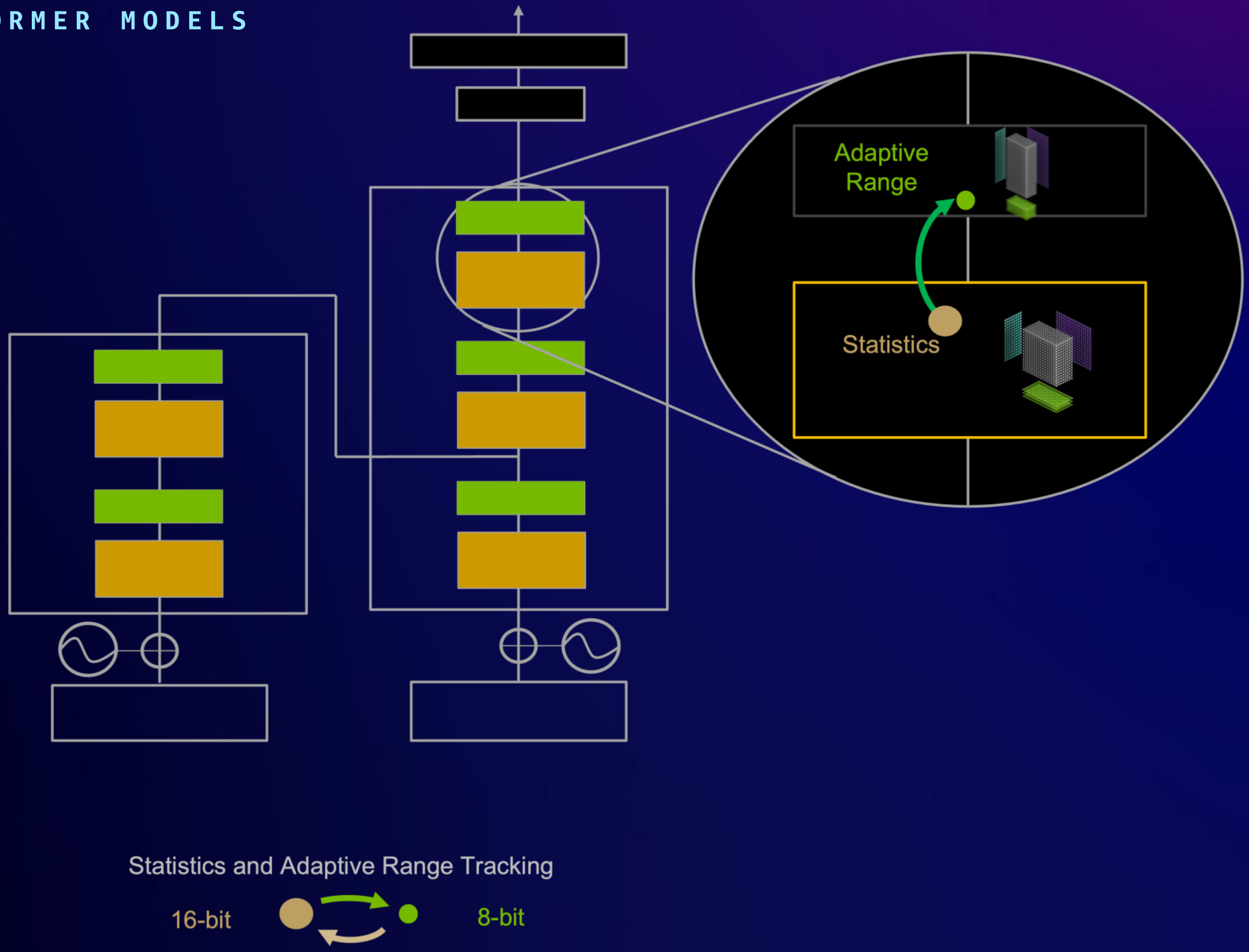


LLM Training | 4096 GPUs | H100 NDR IB | A100 HDR IB | 300 Billion tokens.
P-Tuning | DGX H100 | DGX A100 | 530B Q&A tuning using SQuAD dataset
Inference | chatbot| 10 DGX H100 NDR IB | 10 DGX A100 HDR IB | <1 sec latency | 1 inference/second/user.
H100 data center projected workload performance, subject to change

Transformer engine

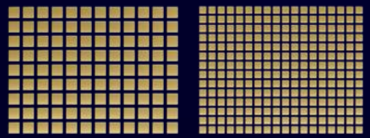
TENSOR CORE OPTIMIZED FOR TRANSFORMER MODELS

- 6x faster training and inference of transformer models
- NVIDIA tuned adaptive range optimization across 16-bit and 8-bit math
- Configurable macro blocks deliver performance without accuracy loss
- **Why is this important for your model training?**

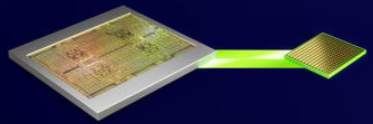


NVIDIA A100

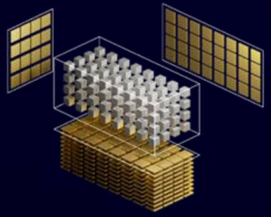
SUPERCHARGING THE WORLD'S HIGHEST PERFORMING AI SUPERCOMPUTING GPU



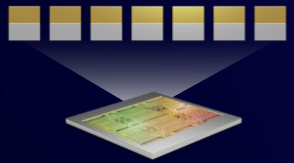
80GB HBM2e
For largest datasets
and models



2 TB/s +
World's highest memory bandwidth
To feed the world's fastest GPU

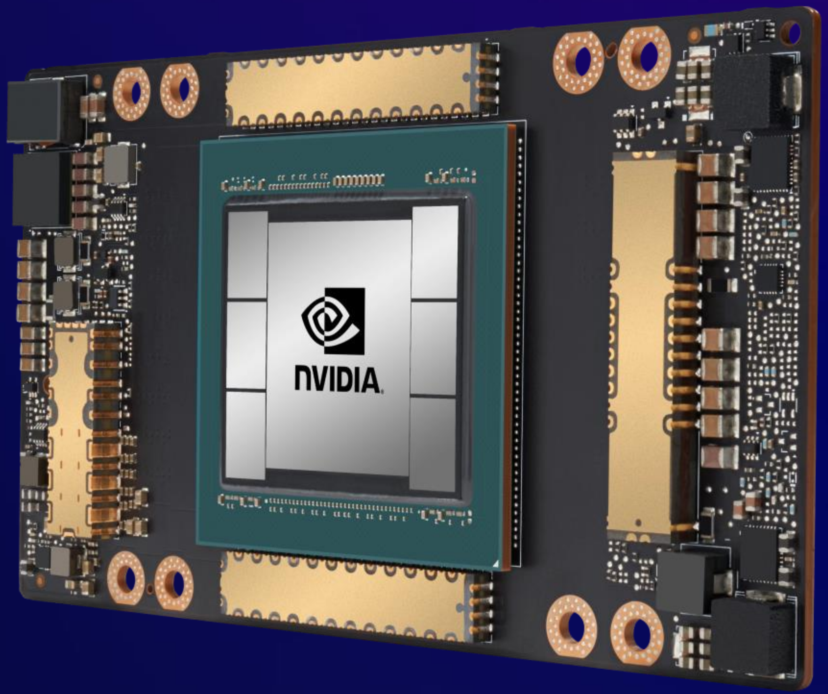


3rd-gen Tensor Core



Multi-instance GPU

3rd-gen NVLink



*Powering AWS P4d/P4de Instances

Amazon EC2 instance powered by NVIDIA GPUs

ACCESSIBLE VIA AWS, AWS MARKETPLACE, AND AWS SERVICES

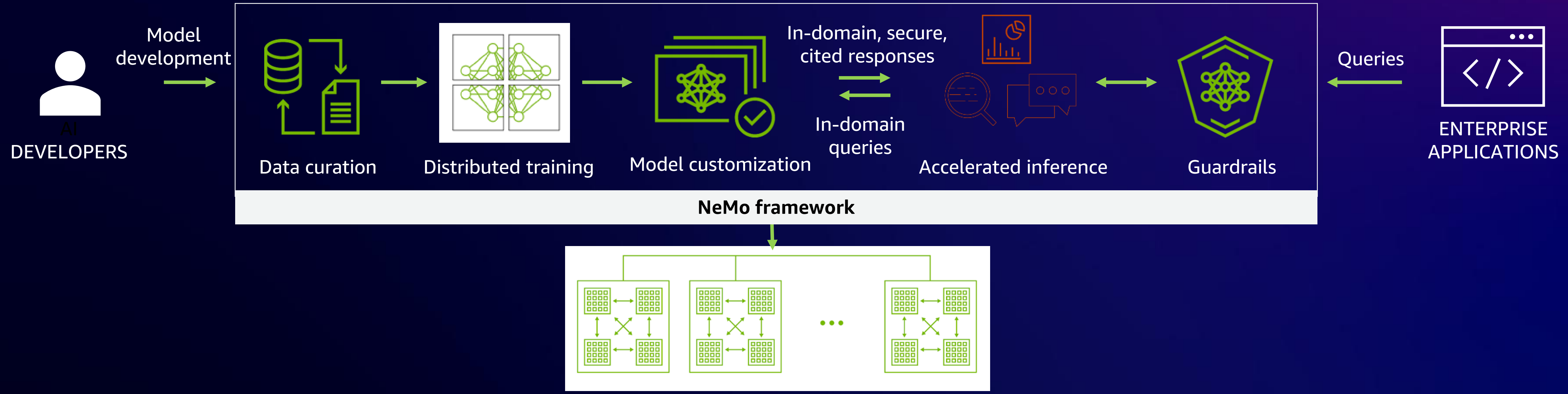
NVIDIA GPU	AWS Instance	GA	Use Case Recommendations	Regions	GPU Memory	GPUs	On-Demand Price/Hour
T4	G4	9/2019	The universal GPU ML inference, training, remote visualization workstations, rendering, video transcoding * Includes Quadro Virtual Workstation	20+	16 GB	1, 4, 8	$0.52 - $7.82
T4G	G5g	11/2021	Graphic workloads such as Android game streaming, ML inference, graphics rendering, and AV simulation	5	16 GB	1, 2	$0.42
A10G	G5	11/2021	Best performance for graphics, HPC and cost-effective ML inference	3	24 GB	1, 4, 8	$1.00
A100	P4d, P4de	11/2020	ML training, HPC across industries	8	40, 80 GB	8	$32.77
H100	P5	7/26/2023	Best performance, ML training, HPC across industries	2	80 GB	8	$98.32

EC2 G5g is now available in US East (N. Virginia), US West (Oregon), and Asia Pacific (Tokyo ,Seoul and Singapore) regions; on-demand, reserved and spot pricing available
EC2 G5 is now available in US East (N. Virginia), US West (Oregon), and Europe (Ireland) regions; on-demand, reserved, spot or as part of savings plans
EC2 P4d is now available in US East (N. Virginia and Ohio), US West (Oregon), Europe (Ireland and Frankfurt), and Asia Pacific (Tokyo and Seoul) regions; on-demand, reserved, spot, dedicated host or savings plan availability

Training

NeMo Framework

END-TO-END, CLOUD-NATIVE FRAMEWORK TO BUILD, CUSTOMIZE, AND DEPLOY GENERATIVE AI MODELS



Multi-modality support

Build language, image, generative AI models

Data Curation @ Scale

Extract, deduplicate, and filter info from large unstructured data at scale

Optimized Training

Accelerate training and throughput by parallelizing the model and the training data across 1,000s of nodes

Model Customization

Easily customize with P-tuning, SFT, Adapters, RLHF, AliBi

Deploy at-scale Anywhere

Run optimized inference at scale anywhere

Guardrails

Keep applications aligned with safety and security requirements using NeMo Guardrails

Support

NVIDIA AI Enterprise and experts by your side to keep projects on track

Now in general availability with NVIDIA AI Enterprise

Multi-modal available via early access now

Model architecture support across modalities

NEMO FRAMEWORK NOW SUPPORTS TRAINING AND DEPLOYMENT OF POPULAR MODEL ARCHITECTURES

Language models

GPT
T5, mT5, T5-MoE
BERT

Text-to-image models

Prompt: A 'sks' dog mecha robot

Stable Diffusion v1.5/v2.0
Imagen
Vision Transformers
CLIP

Image-to-image models

Instruction: Make it on a beach

Dreambooth
InstructPix2Pix

Building generative AI foundation models

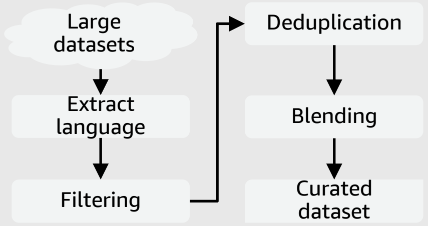
EFFICIENTLY AND QUICKLY TRAINING MODELS USING NEMO

Challenges of building foundation models

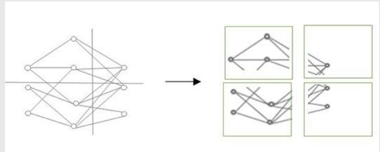
- Mountains of training data
- Large-scale compute infrastructure for training and inferencing, costing $10M+ in just cloud costs
- Deep technical expertise
- Complex algorithms to build on large-scale infrastructure

Accelerated training with NeMo

Data curation tool

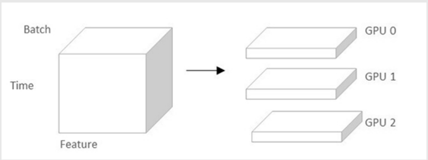


Accelerated training



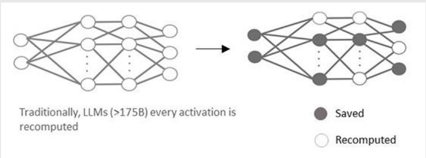
Tensor and pipeline parallelism

Accelerated training



Sequence parallelism

Accelerated training



Selective activation recomputation

Solving pain points across the stack

UNMET NEEDS		NEMO ADDRESSING NEEDS
Large-scale data processing	→	Data curation & preprocessing tools
Multilingual data processing & training	→	Relative positional embedding (RPE) – multilingual support
Finding optimal hyperparameters	→	Hyperparameter tool
Convergence of models	→	Verified recipes for large GPT and T5-style models
Scaling on clouds	→	Scripts/configs to run on Azure, OCI, and AWS
Deploying for inference	→	Model navigator and export to FT functionalities
Deployment at scale	→	Quantization to accelerate inferencing
Evaluating models in industry standard benchmarks	→	Productization evaluation harness
Differing infrastructure setups	→	Full-stack support with FP8 and Hopper support
Lack of expertise	→	Documentation

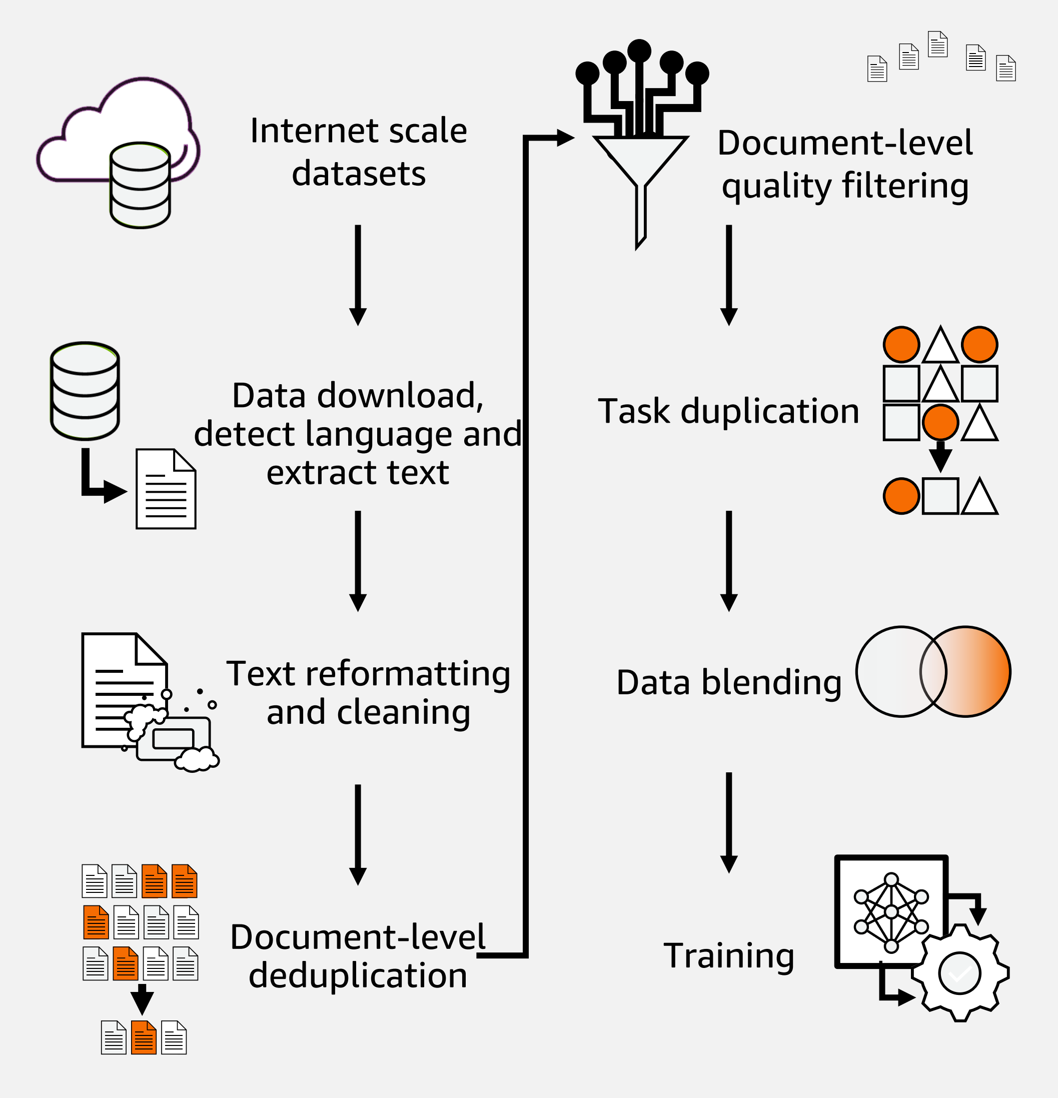
Data curation improves model performance

NEMO DATA CURATOR ENABLING LARGE-SCALE HIGH-QUALITY DATASETS FOR LLMS

- Reduce the burden of combing through unstructured data sources
- Download data and extract, clean, deduplicate, and filter documents at scale

NeMo Data Curator steps:

1. Data download, language detection and text extraction – HTML and LaTeX files
2. Text reformatting and cleaning – Bad Unicode, newline, repetition
3. GPU-accelerated document-level deduplication
   - Fuzzy deduplication
   - Exact deduplication
4. Document-level quality filtering
   - Classifier-based filtering
   - Multilingual heuristic-based filtering
5. Task Deduplication - Performs intra-document deduplication



Megatron core

TRANSFORMER PARALLELISM

PyTorch-based library for scaling transformer training

Core building blocks needed for a LLM FW

Tensor and pipeline parallelism

Distributed optimizer

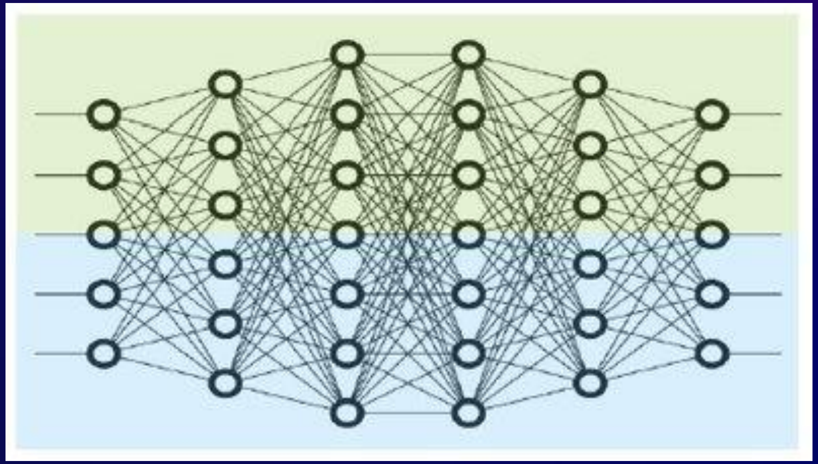
Expert parallelism (MoE)

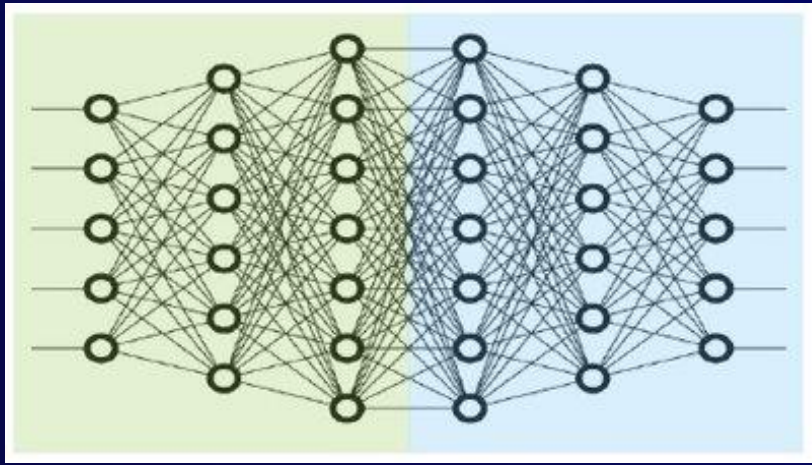
Distributed checkpointing

Modular transformer layer

GPT, BERT, T5, Models

Datasets



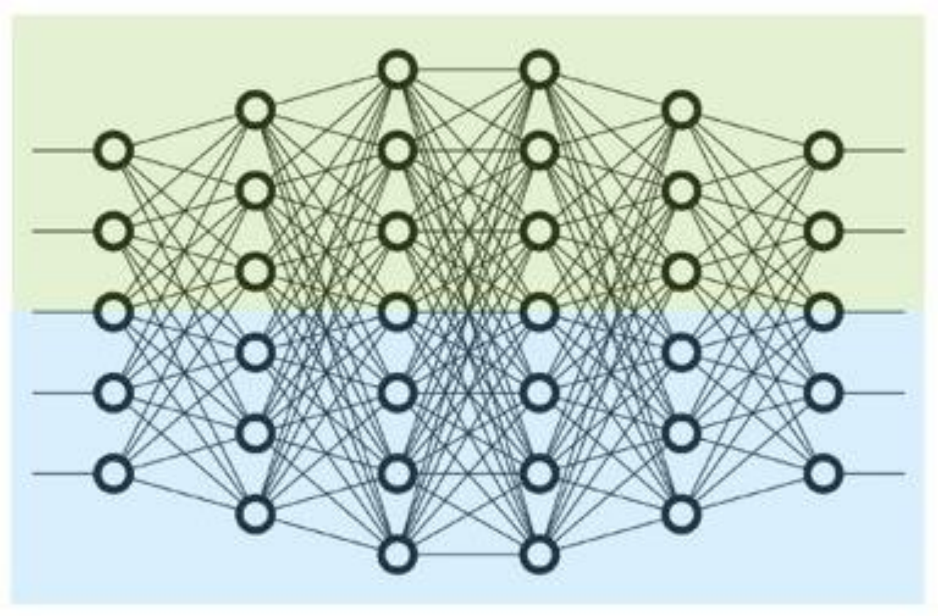


Tensor parallelism for training

TRAINING MODELS AT SCALE

Tensor (intra-layer) parallelism

- Split individual layers across multiple GPUs
- Devices compute different parts of Layers 0,1,2,3,4,5
- Reduces amount of computation on each worker
- Allows scaling of layer size, for bigger models, weak-scale, potential perf improvements with strong-scaling

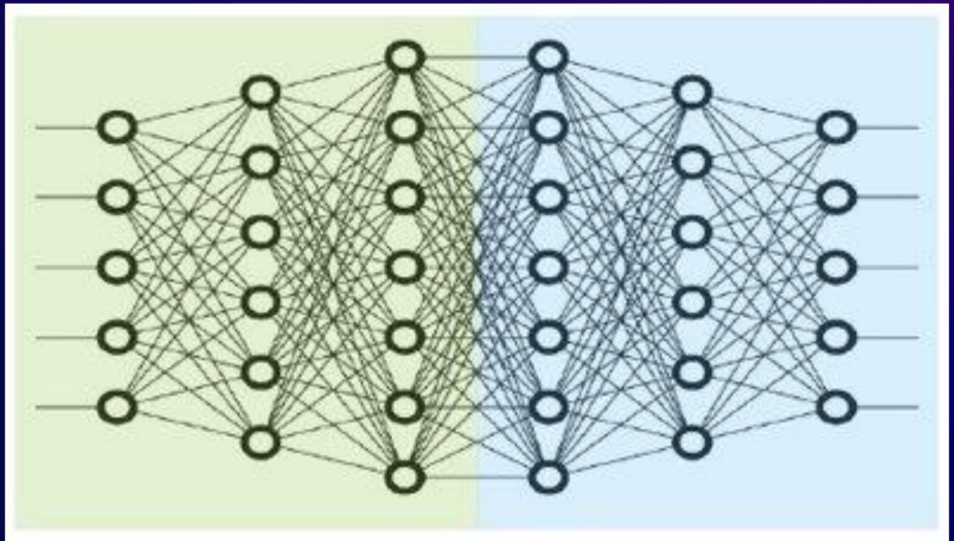


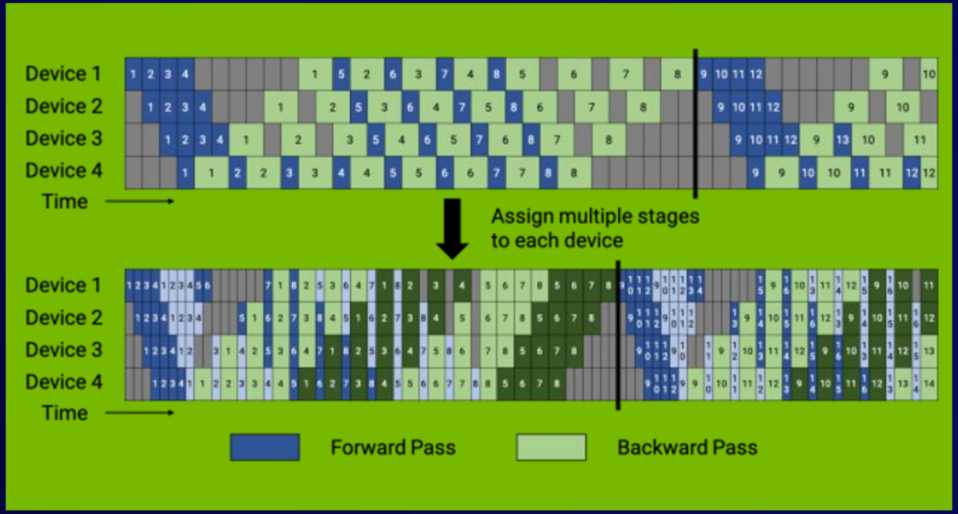
Pipeline parallelism for training

TRAINING MODELS AT SCALE

Pipeline (inter-layer) parallelism

- Split contiguous sets of layers across multiple GPUs
- Layers 0,1,2 and layers 3,4,5 are on different GPUs
- Support for interleaved scheduling to reduce pipeline bubbles
- More difficult to implement for wide range of models then tensor parallelism





Sequence parallelism for training

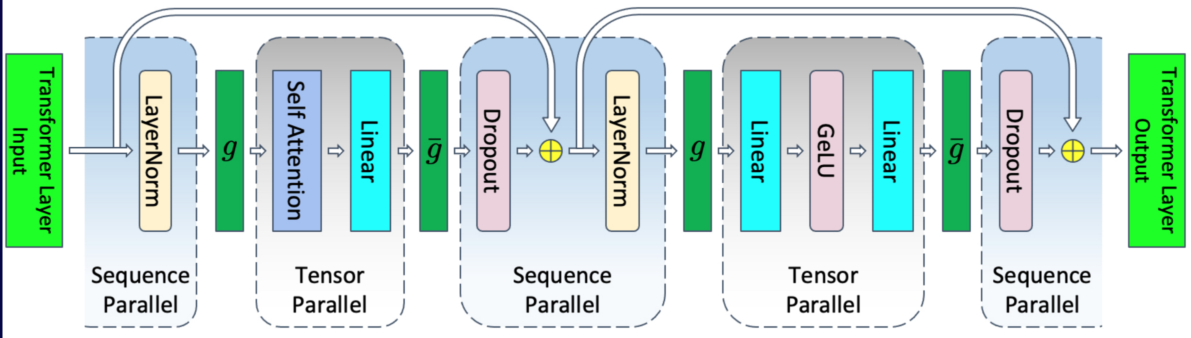
TRAINING MODELS AT SCALE

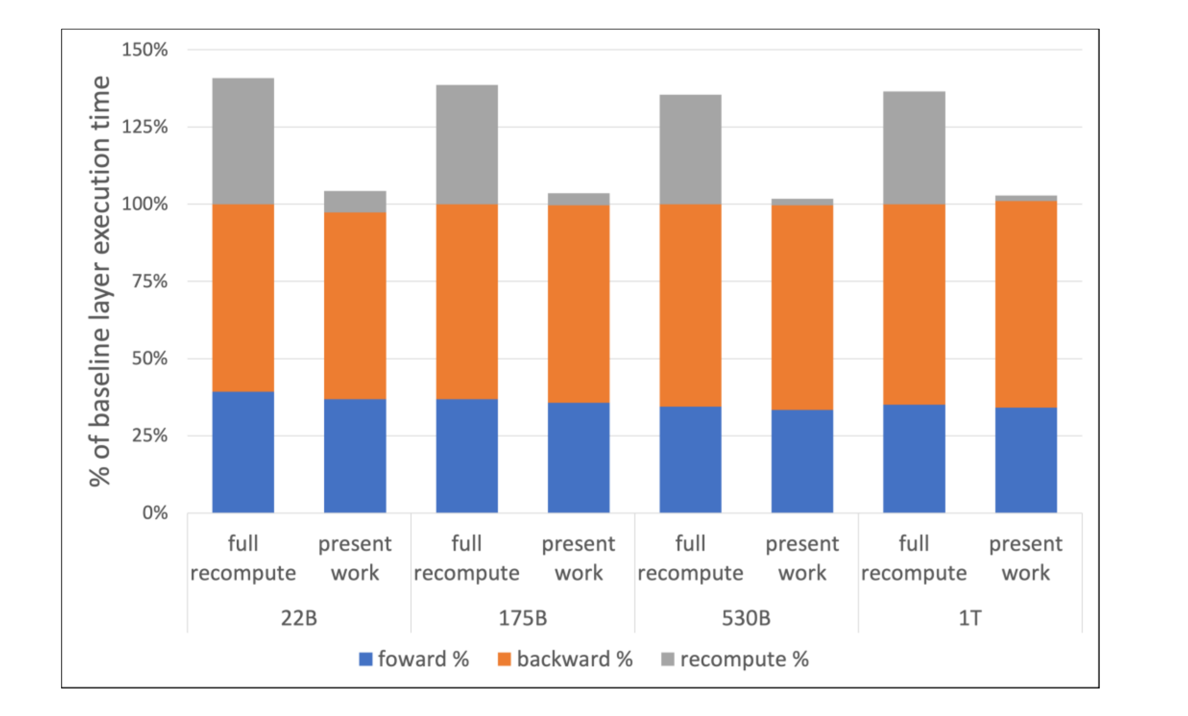
SEQUENCE PARALLELISM

- Splits tensors across the time/sequence dimension
- Reduce memory consumption of activation tensors to reduce re-computation of activations during back-prop

SELECTIVE ACTIVATION RECOMPUTATION

- Choose activations to calculate based on compute-memory tradeoff
- Lower memory footprint of activations and increase throughput of network





Impact of high cost of training LLMs

NEED FOR FASTER AND MORE EFFICIENT WAYS OF TRAINING LARGE MODELS

Economic impact

- Billions of parameters may cost millions of dollars if models are trained inefficiently
- Source: The Cost Of Training NLP Models

- $2.5k - $50k (110 million parameter model)
- $10k - $200k (340 million parameter model)
- $80k - $1.6m (1.5 billion parameter model)

- The cost of one training run, and a typical fully-loaded cost with hyper-parameter tuning and multiple runs per setting
- The following figures are based on internal AI21 Labs data The figures also assume the use of cloud solutions such as GCP or AWS, and on-premise implementations are sometimes cheaper. Still, the figures provide a general sense of the costs.

Environmental impact

- Significant CO2 emissions resulting per kilowatt of compute energy used
- Source: Energy and Policy Considerations for Deep Learning in NLP

Consumption	CO_2e (lbs)
Air travel, 1 passenger, NY↔SF	1984
Human life, avg, 1 year	11,023
American life, avg, 1 year	36,156
Car, avg incl. fuel, 1 lifetime	126,000
Training one model (GPU)	
NLP pipeline (parsing, SRL)	39
w/ tuning & experimentation	78,468
Transformer (big)	192
w/ neural architecture search	626,155

Table 1: Estimated CO_2 emissions from training common NLP models, compared to familiar consumption.[1]

Barrier to entry

- Startups, non-profits, and small colleges have limited accessibility to SOTA models due to unaffordable costs

Resource contention on large clusters

- Limited compute budget with multiple teams and projects on a shared cluster
- Limited scope of making corrections to a trained model

Other concerns

- Language inclusion, toxicity, profanity, etc. (ongoing separate research– will not be addressed in this talk)

What drives the cost of training LLMs

EXTREMELY HIGH COST OF EXPERIMENTATION

Hidden cost of training LLMs

- Besides the direct cost drivers such as dataset size, model size and training volume, there are inefficiencies in how we approach experiments that escalate the overall cost of experimentation

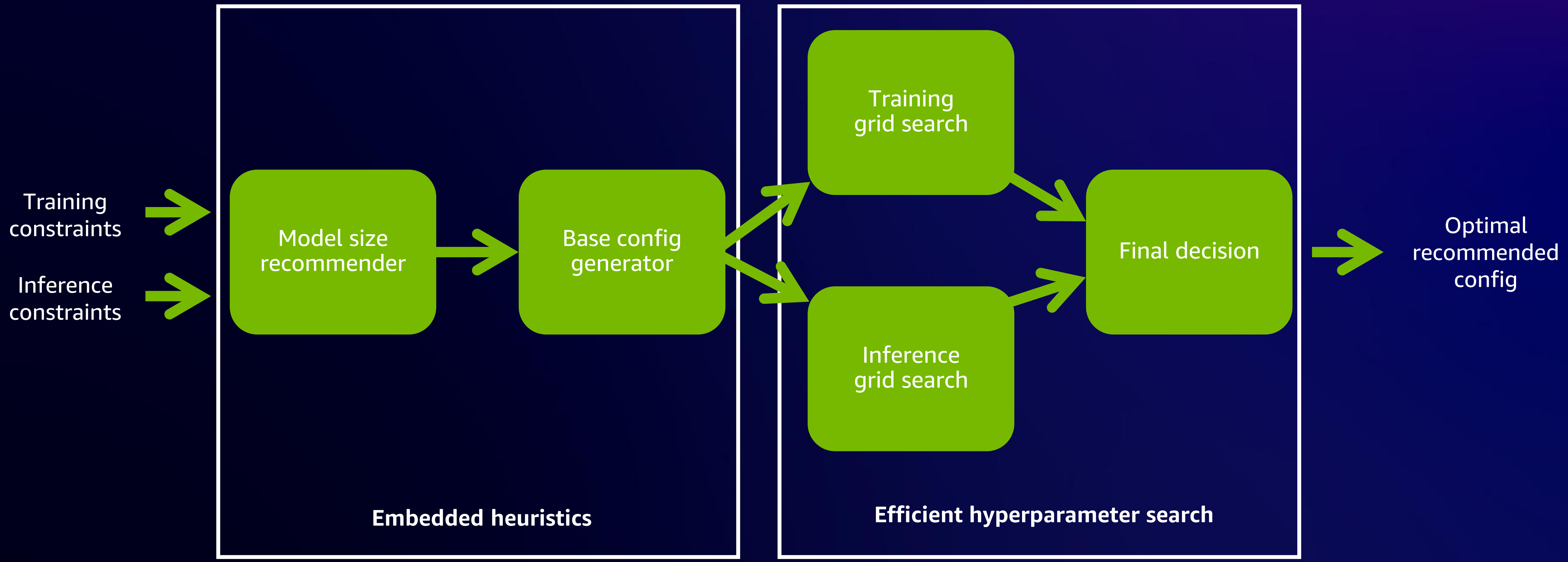
Multiple runs to train meaningful models

- Multiple runs to arrive at stable hyperparameter configuration
- Multiple runs to achieve high scaling efficiency during training
- Multiple inference tests to achieve high throughput and low latency
- Repeat the above steps multiple times for different model sizes
- This reduces or eliminates the scope of making corrections or changes to an already trained model

- What is the right model size for my hardware?
- Which hyperparameters are the most sensitive to convergence?
- How should I improve the throughput of my training runs?
- Which hyperparameters should I change when we add more GPUs?
- How can I use my existing compute optimally?

AutoConfigurator tool

EFFICIENT HYPERPARAMETER SEARCH WITH EMBEDDED HEURISTICS



AutoConfigurator inputs

MAKING NEMO FRAMEWORK FAST AND EASY TO USE

- The user only needs to set some minimal parameters:
  - Model type and optionally size (i.e., GPT-3, 20B params)
  - Number of GPUs to use (i.e., 32 nodes with 8 GPUs each)
  - Vocab size (i.e., 51200 for GPT-3)
  - Number of tokens to run for (i.e., 300B tokens for GPT-3)
  - Grid search training run parameters such as max steps to train

- The tool also allows the user to provide the training constraints instead of the model size for users who don't know what size to use:
  - GPT-3 with 16 nodes for 6 days → 5B parameter model

NVIDIA NeMo works with powerful generative foundation models

SUITE OF GENERATIVE FOUNDATION LANGUAGE MODELS BUILT FOR ENTERPRISE HYPER-PERSONALIZATION

Fastest Responses

GPT-8

Balance of accuracy – latency

GPT-22

For complex tasks

GPT-43

Information retrieval

Community-built models

Falcon LLM

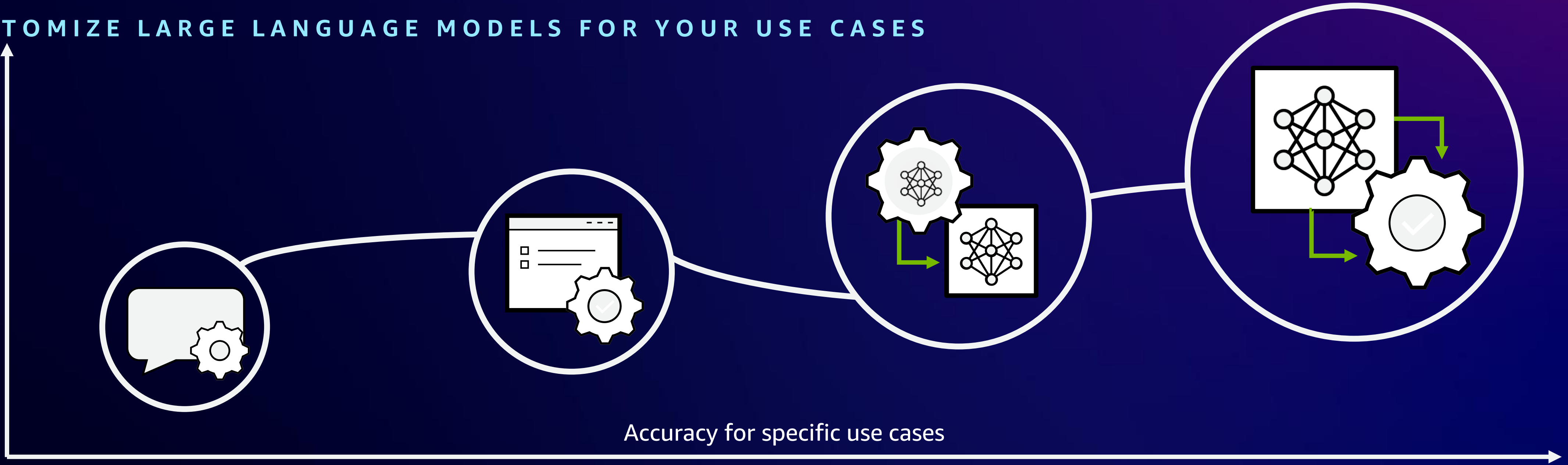
Llama

MPT

StarCoder

Suite of model customization tools in NeMo

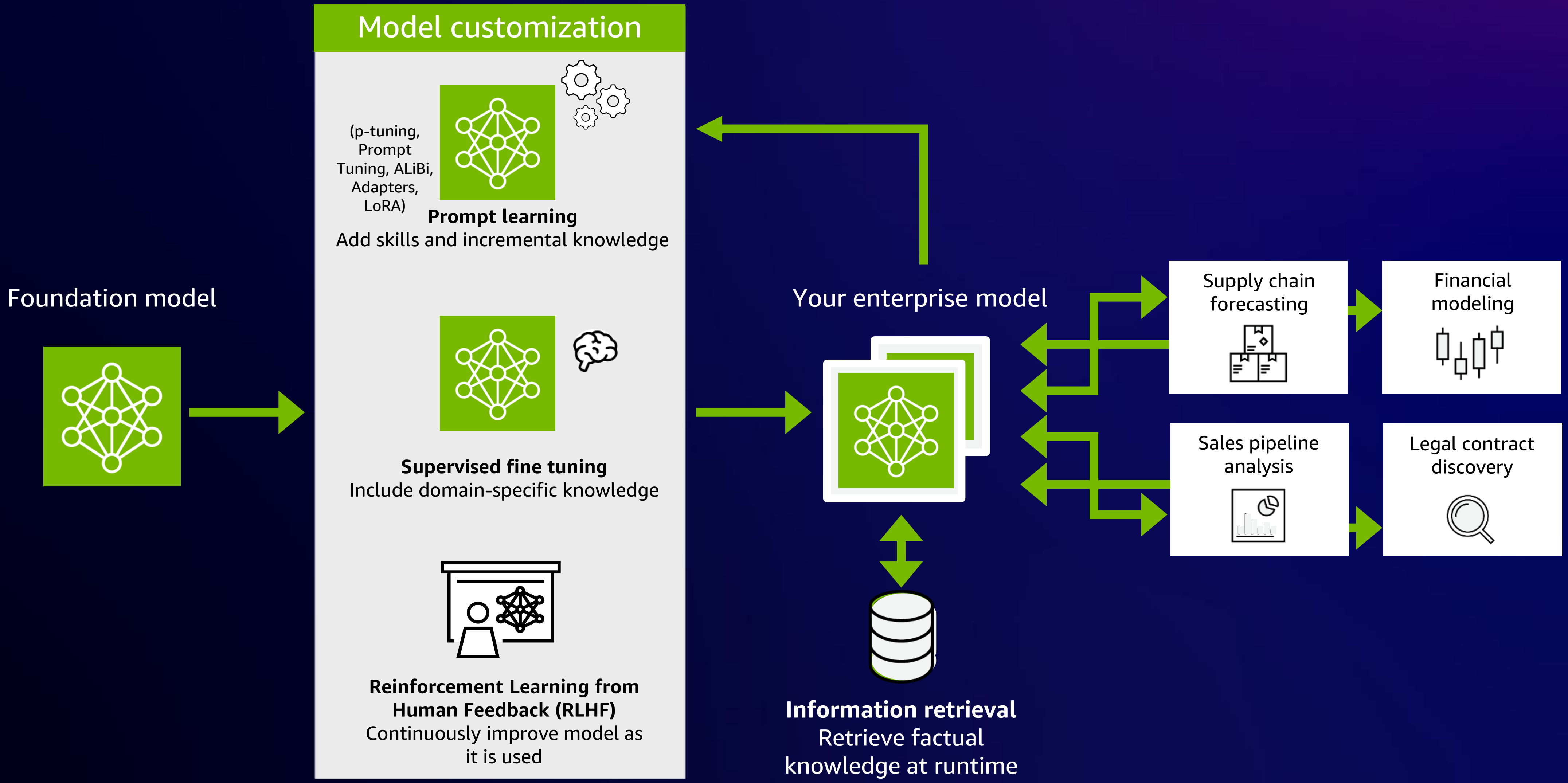
WAYS TO CUSTOMIZE LARGE LANGUAGE MODELS FOR YOUR USE CASES



	PROMPT ENGINEERING	PROMPT LEARNING	PARAMETER EFFICIENT FINE-TUNING	INSTRUCTION TUNING
Techniques	• Few-shot learning • Chain-of-thought reasoning • System prompting	• Prompt tuning • P-tuning	• Adapters • LoRA • IA3	• SFT • RLHF
Benefits	• Good results leveraging pre-trained LLMs • Lowest investment • Least expertise	• Better results leveraging pre-trained LLMs • Lower investment • Will not forget old skills	• Best results leveraging pre-trained LLMs • Will not forget old skills	• Best results leveraging pre-trained LLMs • Change all model parameters
Challenges	• Cannot add as many skills or domain specific data to pre-trained LLM	• Less comprehensive ability to change all model parameters	• Medium investment • Takes longer to train • More expertise needed	• May forget old skills • Large investment • Most expertise needed

Model customization for enterprise-ready LLMs

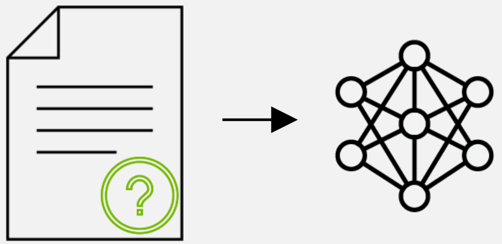
CUSTOMIZATION TECHNIQUES TO OVERCOME THE CHALLENGES OF USING FOUNDATION MODELS



Reinforcement from human feedback

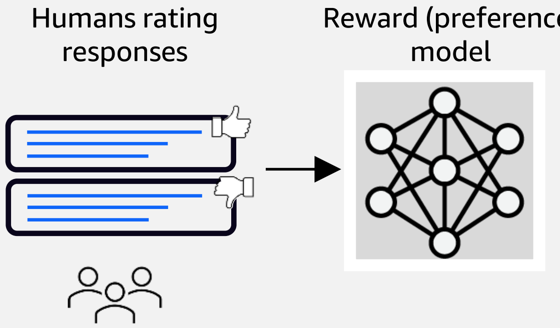
OPEN-SOURCE, SCALABLE, AND DISTRIBUTED LIBRARY TO FINE-TUNE LLMS OF ANY SIZE USING RLHF

1 Supervised fine tuning of LLM



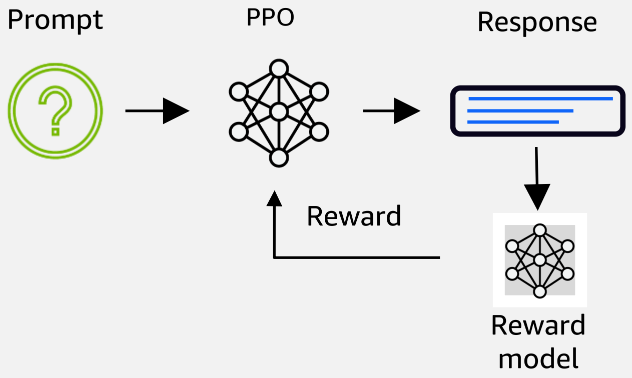
~10K-100K prompt-responses as input
Fine-tune LLM using prompt and responses

2 Train reward model with human feedback



100K-1M responses ranked and rated reward model – trained to mimic human feedback of model generated responses to prompts

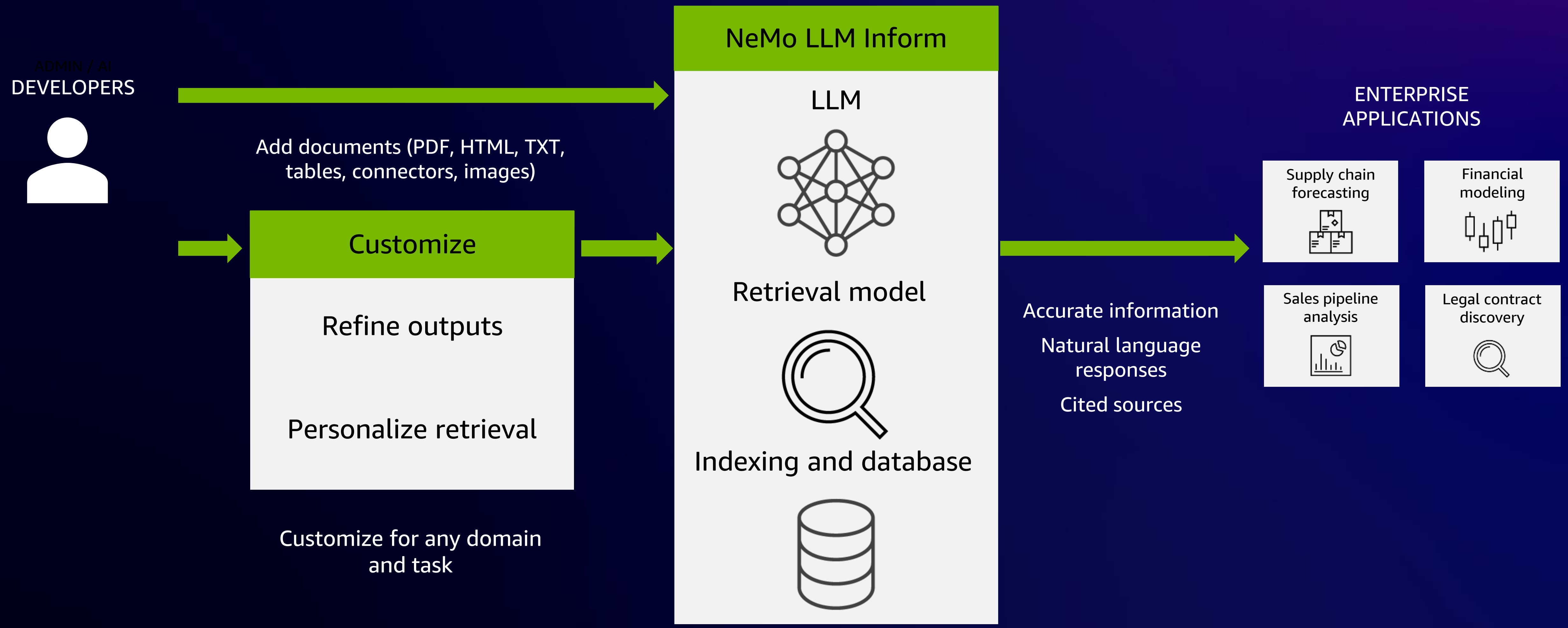
3 Reinforcement learning pipeline with human feedback



Build pipeline with RLHF to continuously improve model over time

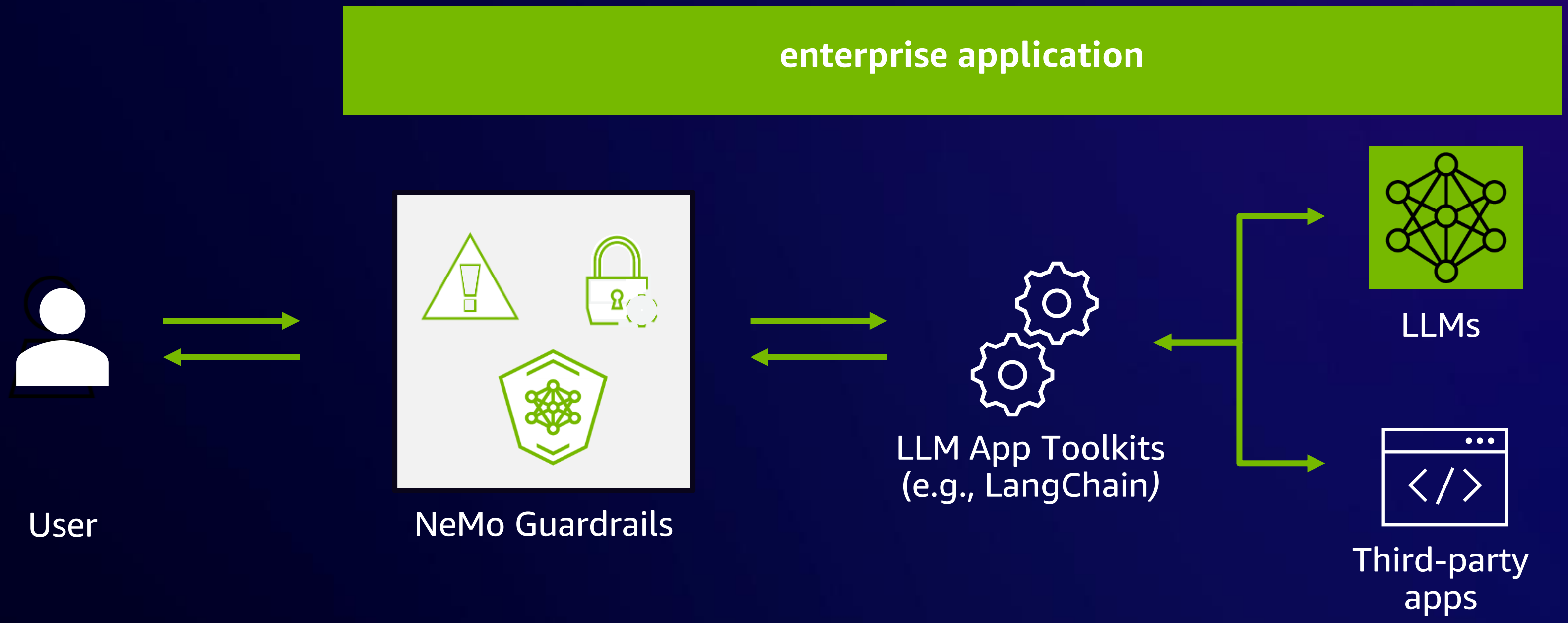
4 models: PPO Policy Network, PPO Value Network, Reward Model, Initial Policy

Retrieval augmented models



Guardrails can keep LLMs on track

ENSURE ACCURACY, APPROPRIATENESS, AND SECURITY IN LLMS



Topical guardrails

Focus interactions within a specific domain

Safety guardrails

Prevent hallucinations, toxic, or misinformative content

Security guardrails

Prevent executing malicious calls and handing power to a third-party app

Get started with NeMo framework

Download now - language

Apply now - multimodal

Web pages

- NVIDIA Generative AI Solutions
- NVIDIA NeMo Framework
- NeMo Guardrails TechBlog

Blogs

- What are Large Language Models?
- What Are Large Language Models Used For?
- What are Foundation Models?
- How To Create A Custom Language Model?
- Adapting P-Tuning to Solve Non-English Downstream Tasks
- NVIDIA AI Platform Delivers Big Gains for Large Language Models
- The King's Swedish: AI Rewrites the Book in Scandinavia
- eBook Asset
- No Hang Ups With Hangul: KT Trains Smart Speakers, Customer Call Centers With NVIDIA AI

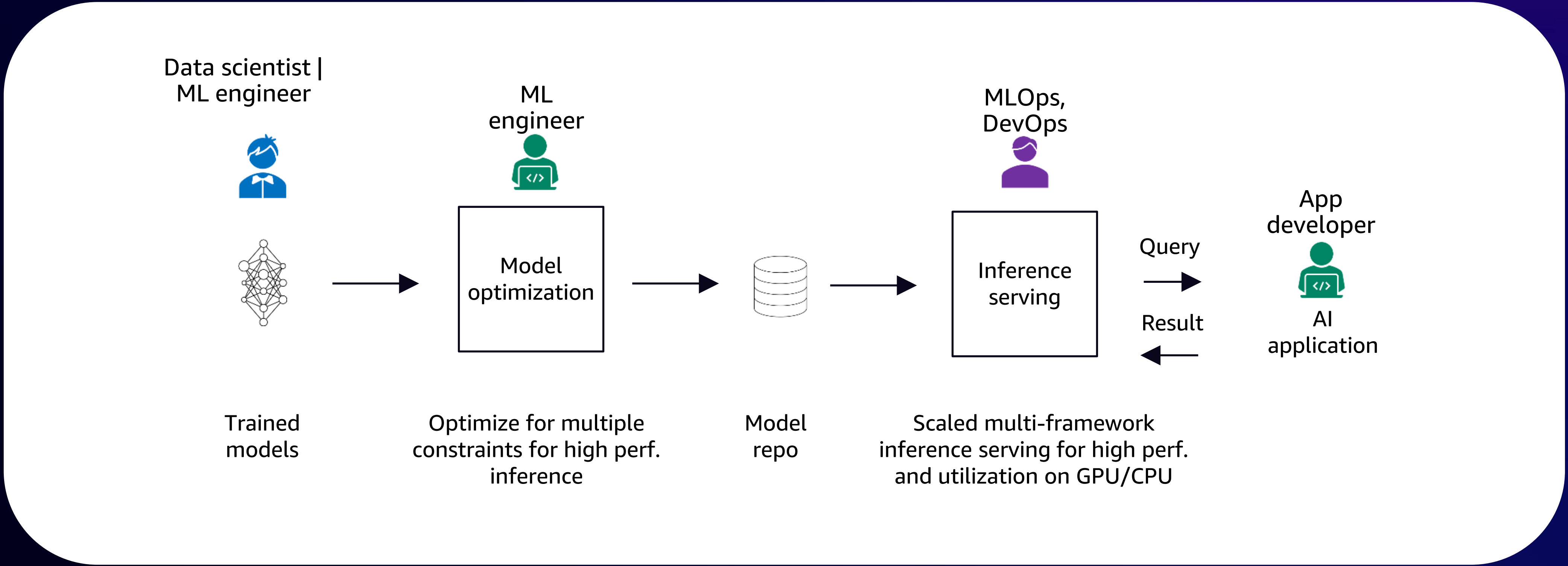
GTC sessions

- How to Build Generative AI for Enterprise Use-cases
- Leveraging Large Language Models for Generating Content
- Power Of Large Language Models: The Current State and Future Potential
- Generative AI Demystified
- Efficient At-Scale Training and Deployment of Large Language Models – GTC Session
- Hyperparameter Tool GTC Session

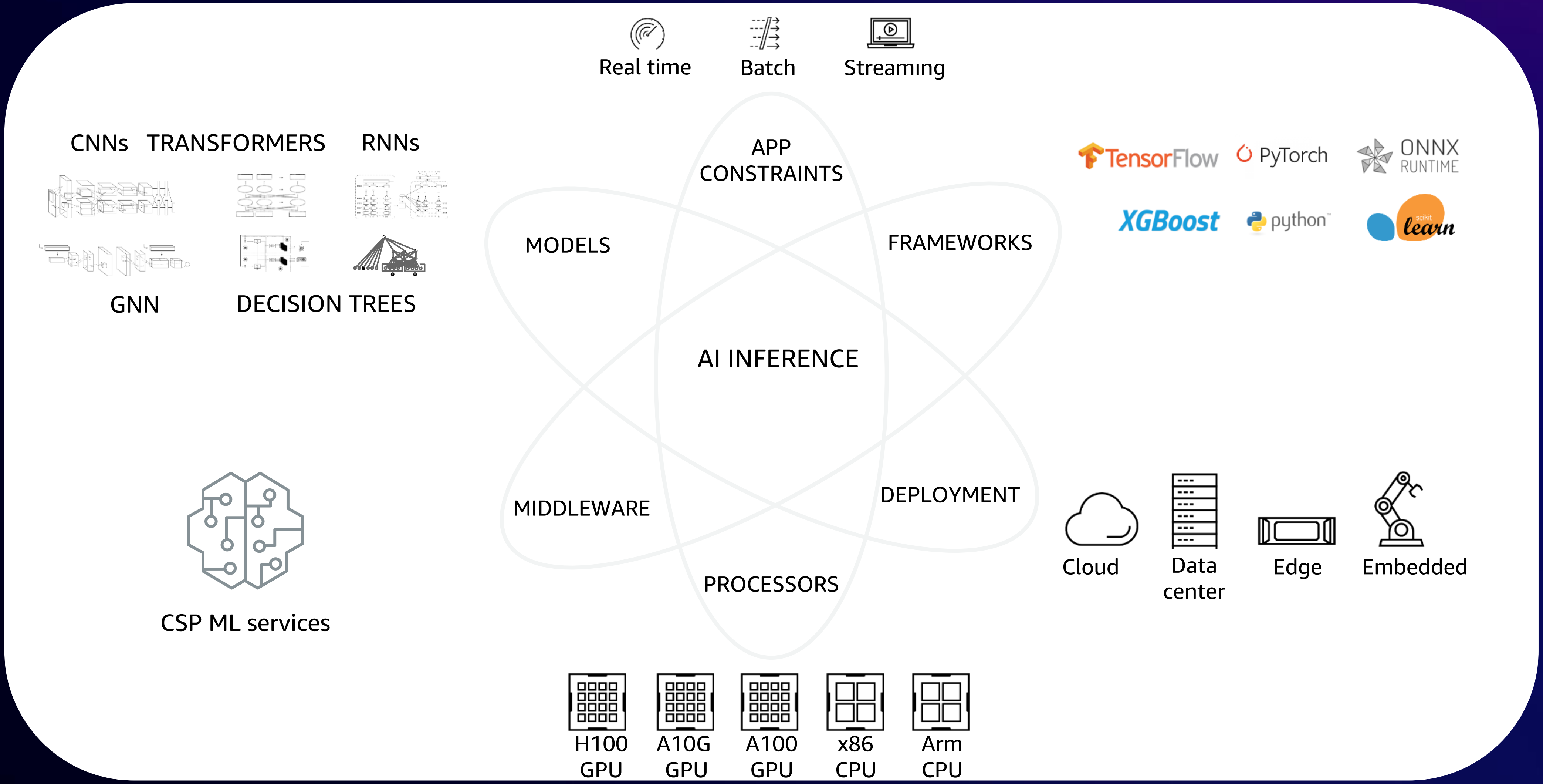
Deployment with Triton Inference Server

AI inference workflow

TWO-PART PROCESS IMPLEMENTED BY MULTIPLE PERSONAS



Challenges In AI inference



NVIDIA Triton Inference Server

OPEN-SOURCE SOFTWARE FOR FAST, SCALABLE, SIMPLIFIED INFERENCE SERVING

Any framework

Supports multiple framework backends natively e.g., TensorFlow, PyTorch, TensorRT, XGBoost, ONNX, Python, and more

Any query type

Optimized for real time, batch, streaming, ensemble inferencing

Any platform

X86 CPU | Arm CPU | NVIDIA GPUs | MIG

Linux | Windows | Virtualization

Public cloud, data center and edge/embedded (Jetson)

DevOps and MLOps

Integration with Kubernetes, KServe, Prometheus, and Grafana

Available across all major cloud AI platforms

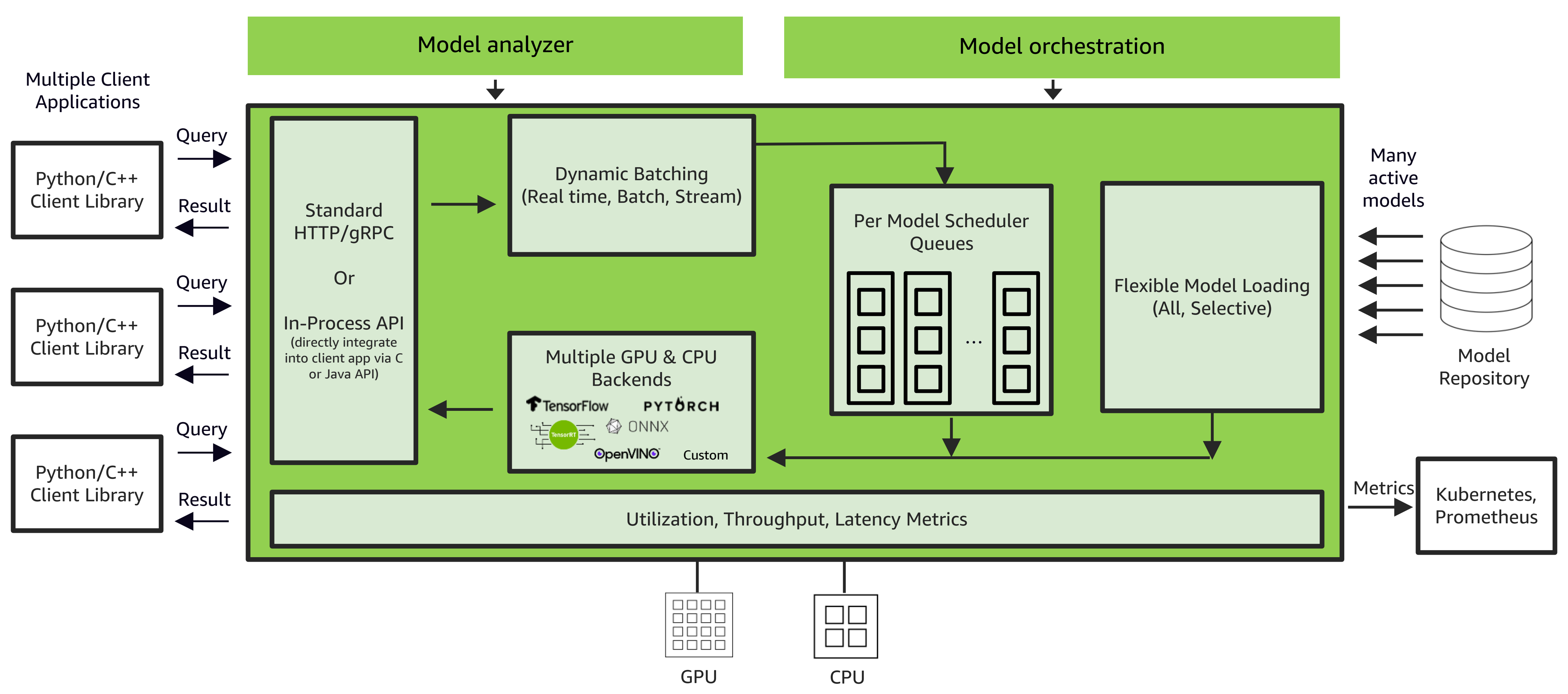
Performance and utilization

Model analyzer for optimal configuration

Optimized for high GPU/CPU utilization, high throughput and low latency

Triton's architecture

DELIVERING HIGH PERFORMANCE ACROSS FRAMEWORKS



Inference with many natively supported backends

TensorFlow 1.x/2.x
Any Model
SavedModel | GraphDef

PyTorch
Any model
JIT/Torchscript | Python

TensorRT
All TensorRT optimized models

TF-TensorRT & TorchTRT
Any TensorFlow and PyTorch model

Forest Inference Library (FIL)
Tree-based models (e.g., XgBoost, Scikit-learn RandomForest)

ONNX RT
ONNX converted models

Python
Custom code in Python e.g. pre/post processing, any Python model.

TensorRT-LLM
Multi-GPU, multi-node inferencing for large language models (LLM)

OpenVINO
OpenVINO optimized models on Intel architecture

Custom C++ Backend
Custom framework in C++

DALI
Pre-processing logic using DALI operators

NVTabular
Feature engineering and preprocessing library for tabular data

HugeCTR
Recommender model with large embeddings

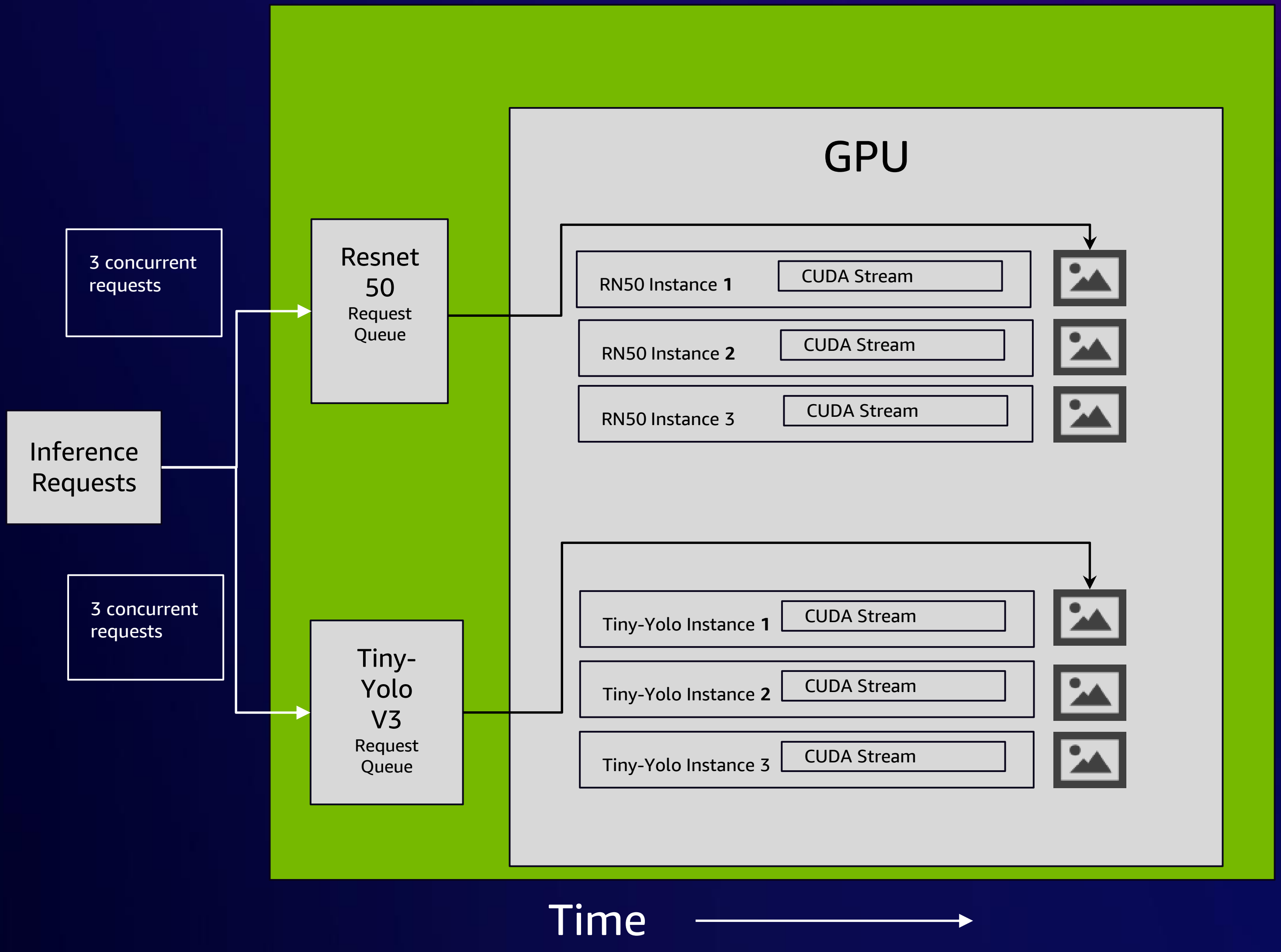
Concurrent model execution

INCREASE THROUGHPUT AND UTILIZATION

Triton can run concurrent inference on:

1) Multiple different models
2) And/or multiple copies of the same model in parallel on the same system

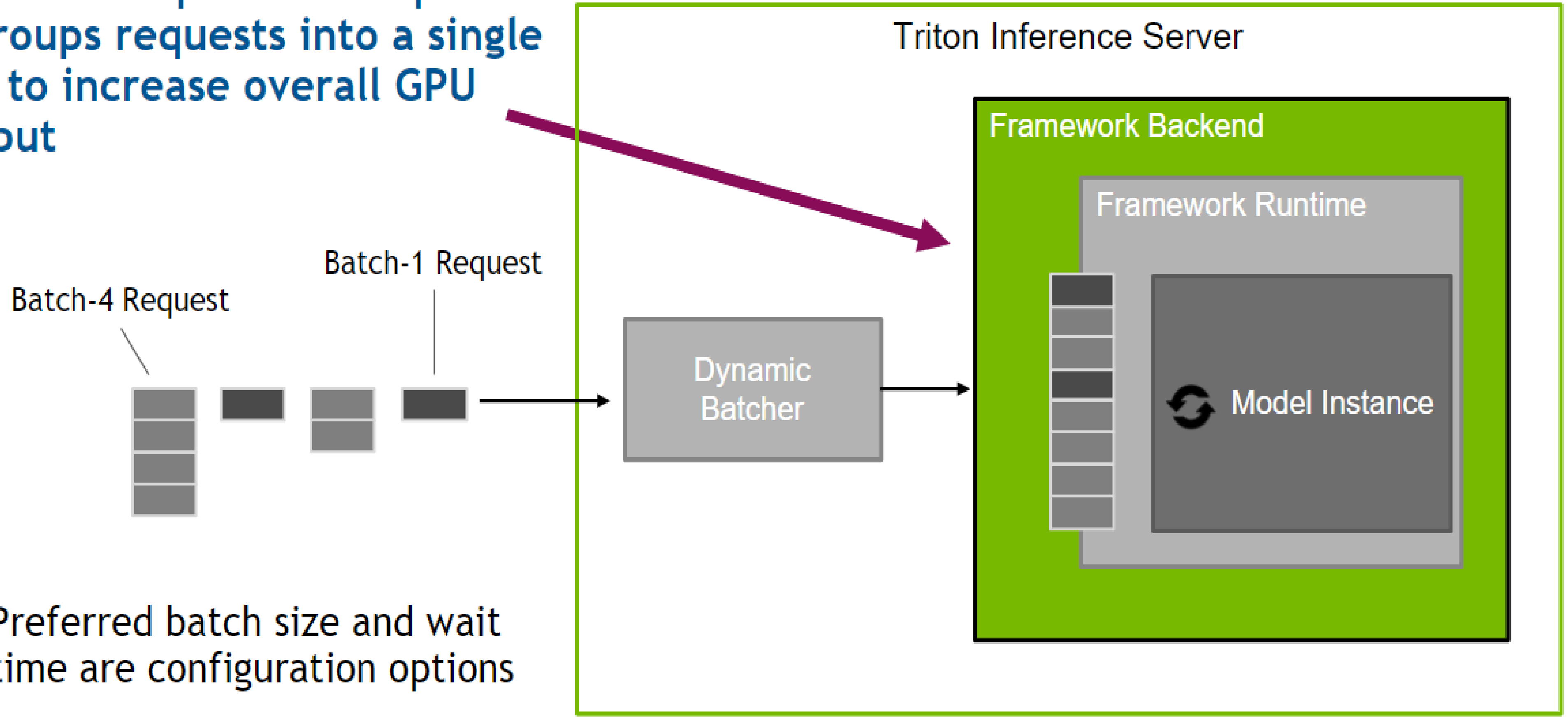
Maximizes GPU utilization, enabling better performance and lowering the cost of inference



Dynamic batching scheduler

GROUP REQUESTS TO FORM LARGER BATCHES AND INCREASE GPU UTILIZATION

Client sends independent requests. Triton groups requests into a single "batch" to increase overall GPU throughput



Preferred batch size and wait time are configuration options

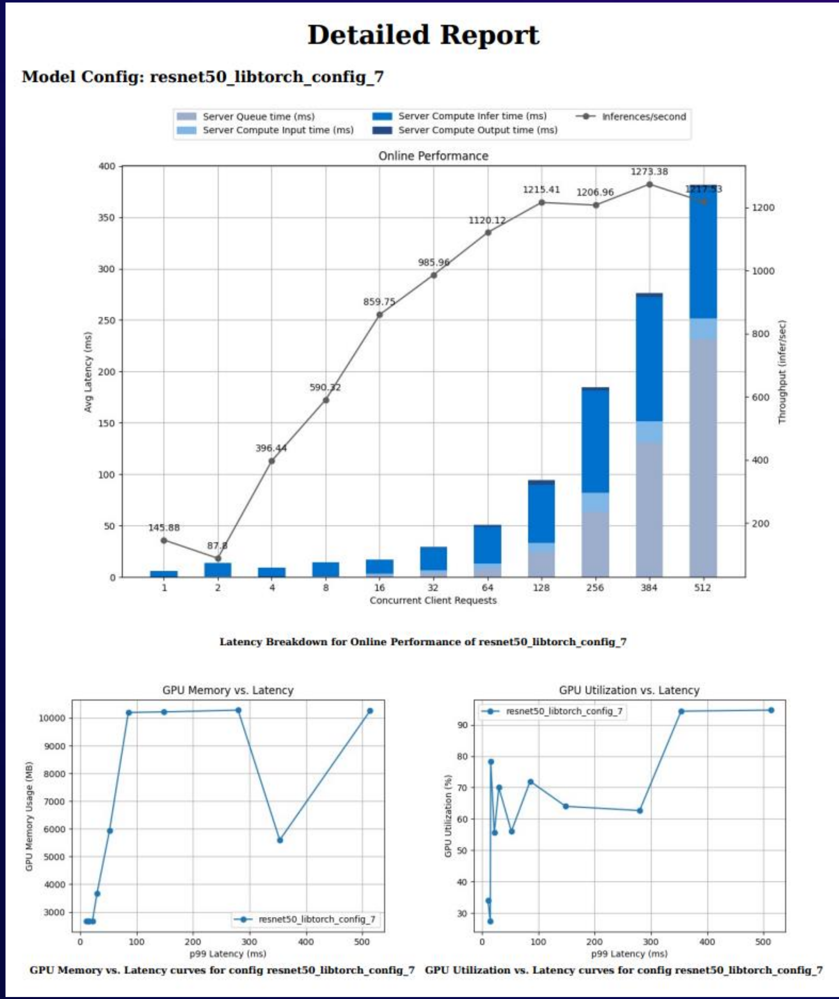
Optimal model configuration

USING THE MODEL ANALYZER CAPABILITY

Model analyzer

QoS constraints (Throughput, latency, memory) → **Evaluate multiple configs** (Precision X Batch Size X Concurrent instances) → Optimal model configuration

↑ Models

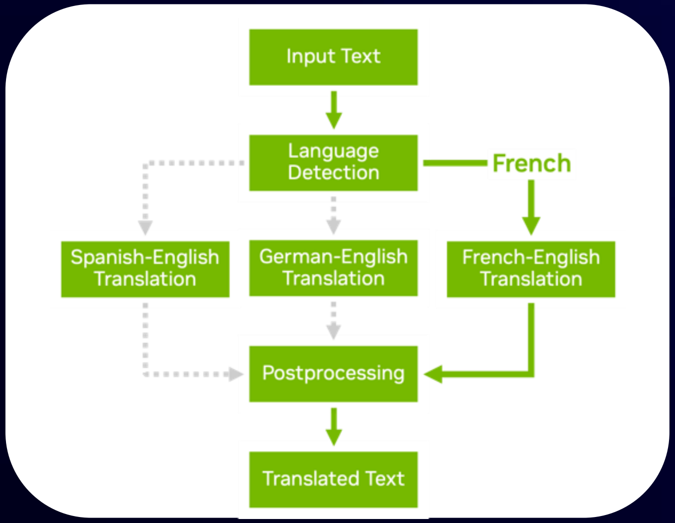


GitHub repo and docs: https://github.com/triton-inference-server/model_analyzer

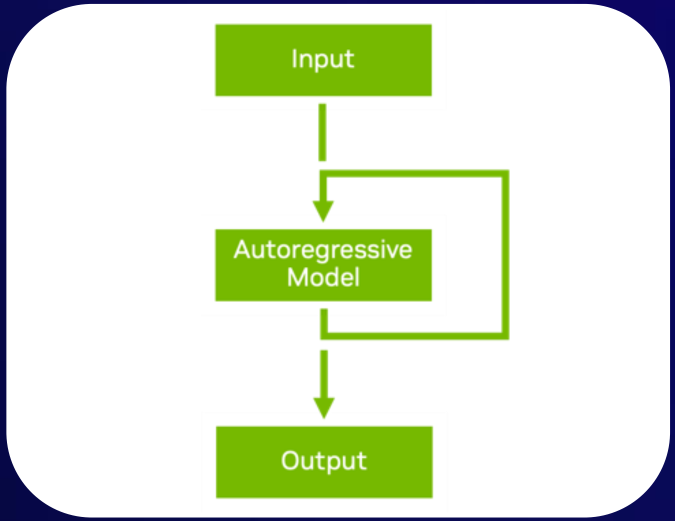
Model pipelines with business logic scripting

CONTROL FLOW AND LOOPS IN MODEL ENSEMBLES

- **Model ensembles beyond simple pipelines:** Enables arbitrary ordering of models with conditionals, loops, and other custom control flow
- **Call any other backend from the Python or C++ backend:** Triton will efficiently pass data to and from the other backend



Conditional model execution

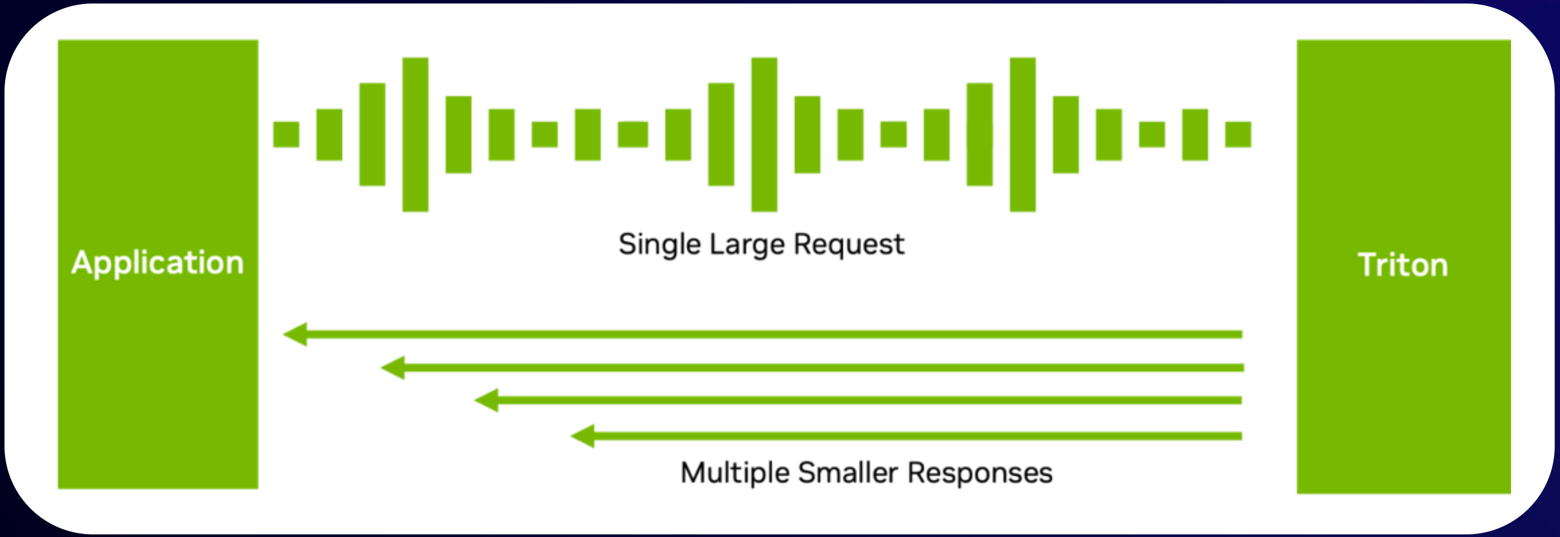


Looping execution for autoregressive models

Decoupled models

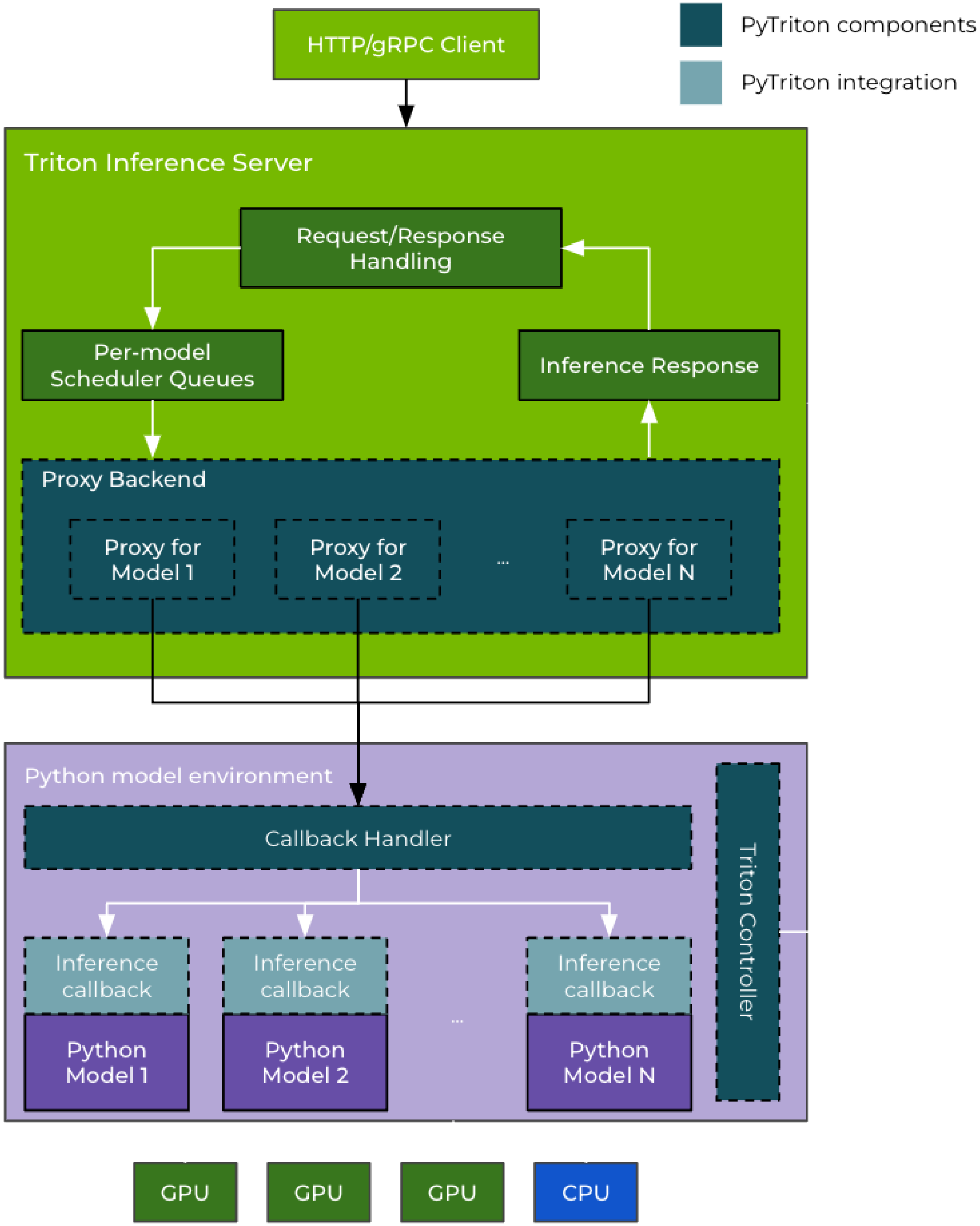
ALLOWS 0, 1, OR 1+ RESPONSES PER REQUEST

- For situations where requests and responses are not strictly 1-to-1
  - e.g., automated speech recognition use cases
- Supported in C++ and Python back-end
- Requires use of gRPC bi-directional streaming API or in-process C/Java API



PyTriton

- **Provides a more “Flask-like experience” for users**
  - No need for model repo and config.pbxt file
  - Enables faster prototyping
- **Ease of integration into existing codebase with a python interface**
  - Integrate Triton directly into training container/environment
  - Requires a simple pip install rather installing large container
  - All Triton dependencies included
- **Unlocks new use cases**
  - e.g., online training
- **Easily deploy any Python model/function**
  - JAX, PyTorch models that don't export (e.g. DGL), RAPIDS, Numpy
- **Enables rich set of preprocessing capabilities without need for custom backend**
  - e.g., emulate dynamic batching for models that don't support it



Delivering value across industries

amazon
Search & ads

American Express
Payment fraud detection

Ant Group
Digital copyright management

GE Healthcare
Medical imaging

智能互联 INTELLIGENT CONNECTIVITY
Alibaba
Text to speech for smart speaker

NIO
Image processing for autonomous driving

Office 365
Grammar check

Teams
Meeting transcription

Microsoft
Document translation

NAVER
Image classification and recommendation

SIEMENS energy
Preventive maintenance

BPIL
Defect detection

Tencent 腾讯
Multiple use cases

airtel
Contact center speech analytics

United States Postal Service
Package analytics

火山引擎
ByteDance Volcengine ML Platform

Wealthsimple
Fraud, fintech

YAHOO! JAPAN
Similar image search

TRACXPOINT
Retail product Identification

Intelligent Voice®
Financial fraud detection

Clinical notes analytics

co:here
NLP Services

tabnine
Coding assistant

RINNA
AI character generation

Learn more and download

For more information:
https://developer.nvidia.com/nvidia-triton-inference-server

Get the ready-to-deploy container with monthly updates from the NGC catalog:
https://catalog.ngc.nvidia.com/orgs/nvidia/containers/tritonserver

Open-source GitHub repository:
https://github.com/NVIDIA/triton-inference-server

Latest release information:
https://github.com/triton-inference-server/server/releases

Quick start guide:
https://github.com/triton-inference-server/server/blob/main/docs/quickstart.md

Triton Inference Server on Amazon SageMaker

A Triton Inference Server Container developed with NVIDIA, includes NVIDIA Triton Inference Server along with useful environment variables to tune performance (e.g., set thread count) on SageMaker

Use with Amazon SageMaker Python SDK to deploy your models on scalable, cost-effective SageMaker endpoints without worrying about Docker

Code examples to find readily usable code samples using Triton Inference Server with popular machine learning frameworks on Amazon SageMaker

Benefits of Triton Inference Server on Amazon SageMaker

Cost-effective – SageMaker optimizes scale and performance to reduce costs

MLOps-ready – Includes metadata persistence, model management, logging metrics to Amazon CloudWatch, and monitoring data drift and performance

Flexible – Ability to run real-time inference for low latency, offline inference on batch data, and asynchronous inference for longer inference times

Secure – High-bar on security, with available mechanisms including encryption at rest and in transit, VPC connectivity, and fine-grained IAM permissions

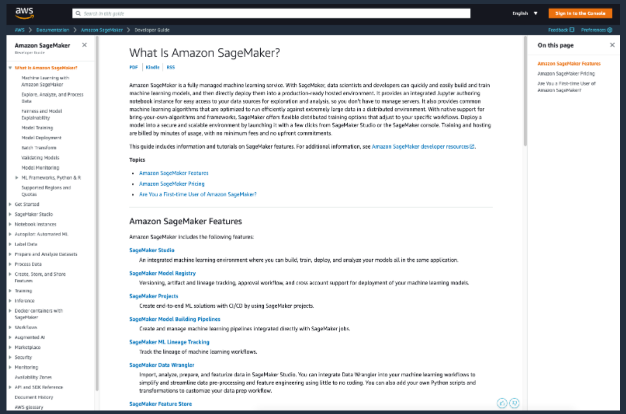
Less heavy lifting – No container registry management, no custom installation of libraries, no extra SDK configuration

Get started with Triton Inference Server on Amazon SageMaker

Sample notebooks

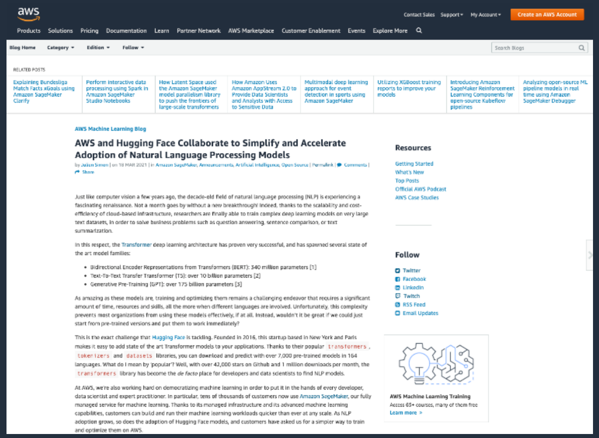
[Access GitHub]

Documentation



[AWS Documentation]

Launch blog



[Read the blog]

Amazon SageMaker and Triton technical resources

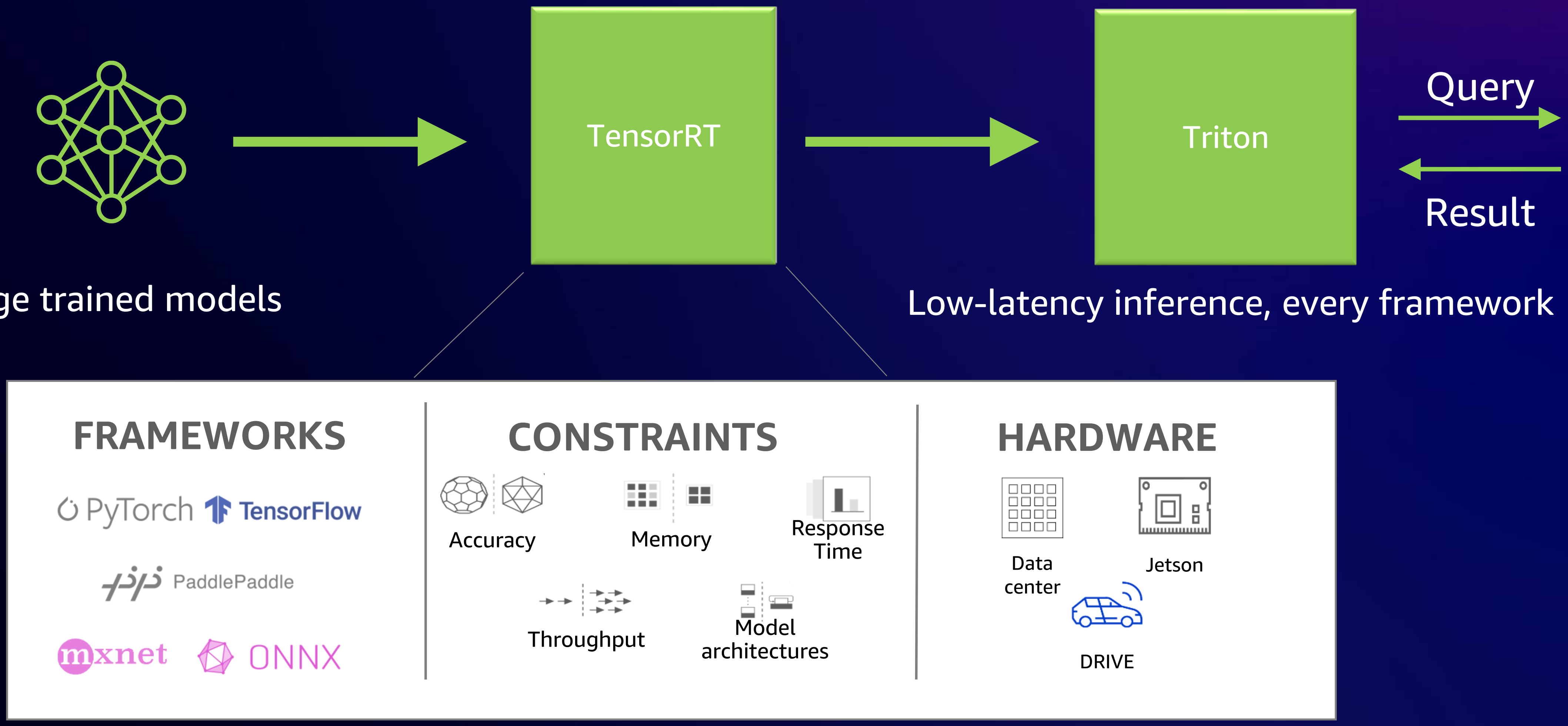
Triton on SageMaker:

- Amazon announces new NVIDIA Triton Inference Server on Amazon SageMaker
- Use Triton Inference Server with Amazon SageMaker
- Host ML models on Amazon SageMaker using Triton: Python backend
- Host ML models on Amazon SageMaker using Triton: TensorRT models
- Hosting ML Models on Amazon SageMaker using Triton: XGBoost, LightGBM, and Treelite Models
- Run multiple deep learning models on GPU with Amazon SageMaker multi-model endpoints
- Achieve low-latency hosting for decision tree-based ML models on NVIDIA Triton Inference Server on Amazon SageMaker
- Achieve hyperscale performance for model serving using NVIDIA Triton Inference Server on Amazon SageMaker
- Deploy fast and scalable AI with NVIDIA Triton Inference Server in Amazon SageMaker
- Getting the Most Out of NVIDIA T4 on AWS G4 Instances
- How Amazon Search achieves low-latency, high-throughput T5 inference with NVIDIA Triton on AWS

Acceleration with TensorRT

Inference is complex

REAL TIME | COMPETING CONSTRAINTS | RAPID UPDATES



World-leading inference performance

TENSORRT ACCELERATES EVERY WORKLOAD

BEST-IN-CLASS RESPONSE TIME AND THROUGHPUT **VS. CPUs**

36X Computer vision < 7ms	**583X** Speech recognition < 100ms	**21X** NLP < 50ms
10X Reinforcement learning	**178X** Text-to-speech < 100ms	**12X** Recommenders < 1 sec

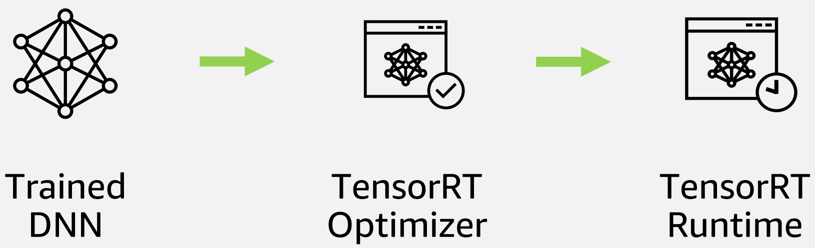
NVIDIA TensorRT

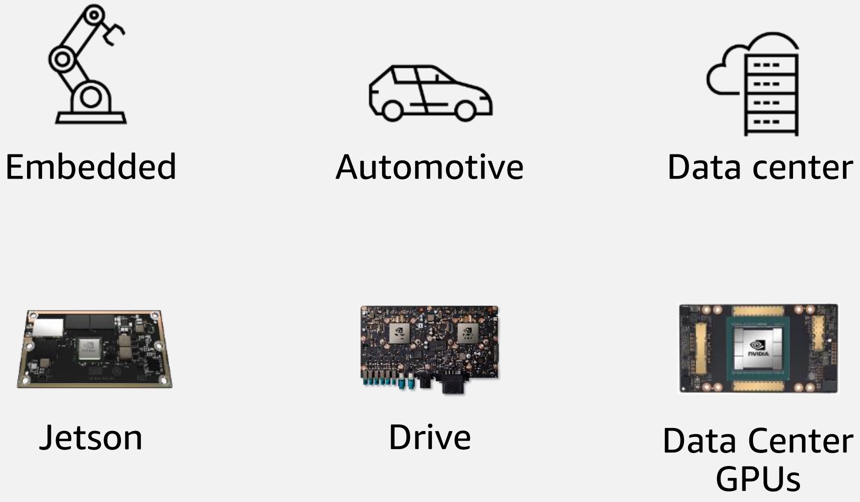
SDK FOR HIGH-PERFORMANCE DEEP LEARNING INFERENCE

Optimize and deploy neural networks in production.

Maximize throughput for latency-critical apps with compiler and runtime; optimize every network, including CNNs, RNNs, and Transformers

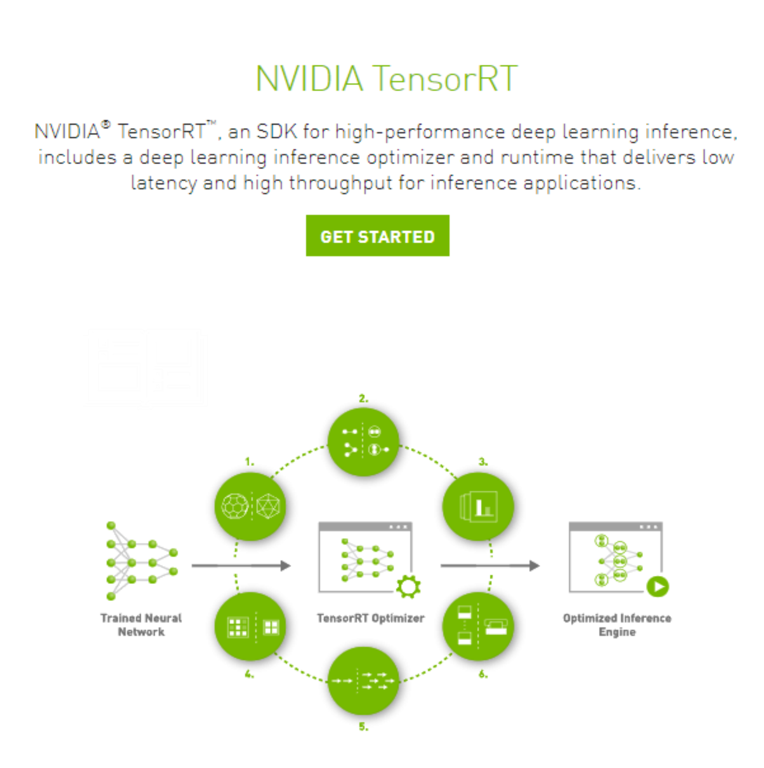
1. Reduced mixed precision: FP32, TF32, FP16, and INT8
2. Layer and tensor fusion: Optimizes use of GPU memory bandwidth
3. Kernel auto-tuning: Select best algorithm on target GPU
4. Dynamic tensor memory: Deploy memory-efficient apps
5. Multi-stream execution: Scalable design to process multiple streams
6. Time fusion: Optimizes RNN over time steps



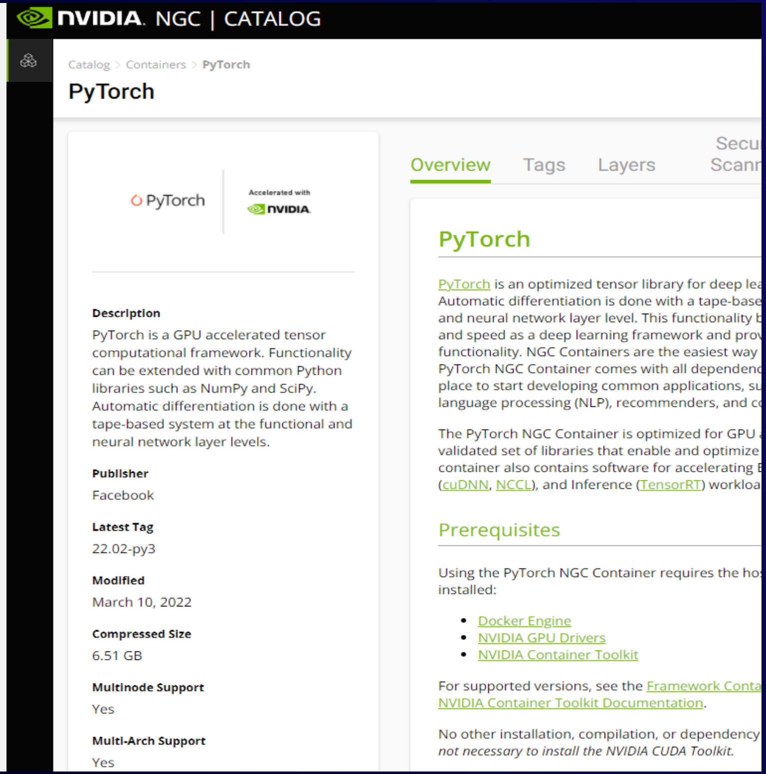


Download TensorRT today

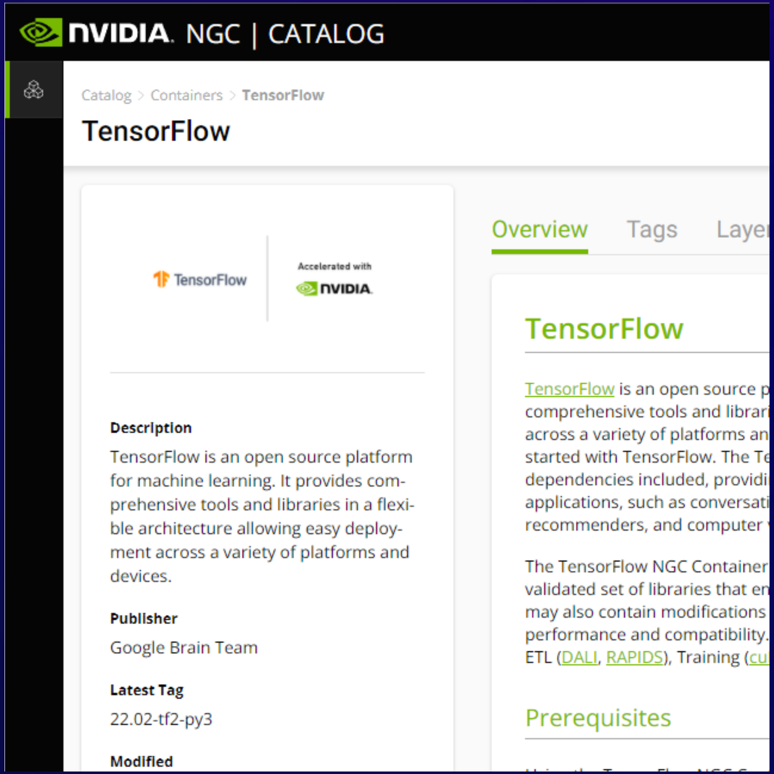
TensorRT



Torch-TensorRT



TensorFlow-TensorRT



TensorRT 8.6 GA is available for free to the members of the NVIDIA Developer Program:
developer.nvidia.com/tensorrt

Large language model ecosystem

RAPID EXPANSION IN LLMS, INFERENCE IMPORTANCE INCREASING

LLaMa | Dolly | Falcon | Starcoder | ChatGLM | MPT | ...

LLM ecosystem expanding rapidly

- Increase in the rate of performant community LLMs
  - LLaMa, Dolly, Falcon, Starcoder, ChatGLM, MPT
- Many new operators, fine tuning, and quantization
- More and more companies deploying LLMs

Inference importance increasing

- Need good inference deployments for production
- Performance decreases costs and improves user experience
- Large model sizes drives costs and complexity of deployment

Need **performant, extensible, and robust** solution

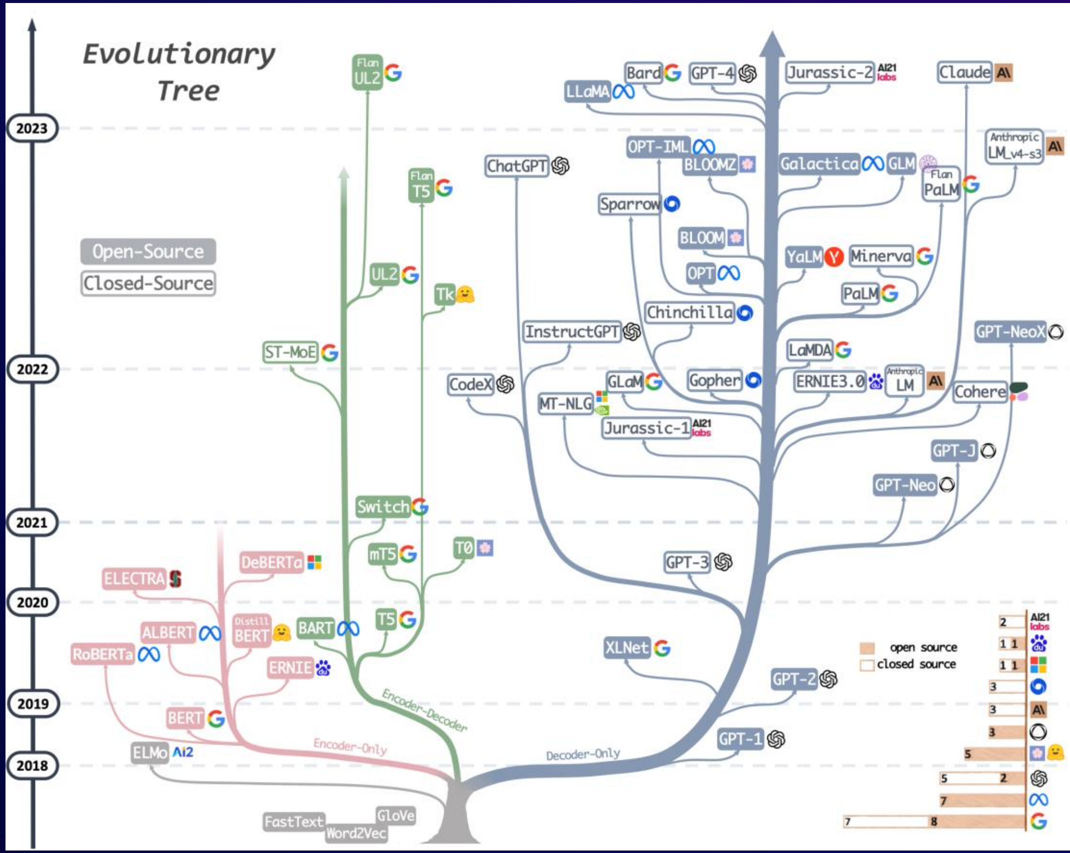


Image from Mooler0410/LLMsPracticalGuide

Yang, J., Jin, H., Tang, R., Han, X., Feng, Q., Jiang, H., … Hu, X. (2023). Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond. arXiv [Cs.CL]. Retrieved from http://arxiv.org/abs/2304.13712

Current ecosystem for LLMs

IMPROVING ON NV LLM INFERENCE OPTIMIZATION TO MEET ECOSYSTEM DEMAND

FasterTransformer

- **33x speedup** over CPU, **6x** over PyTorch GPU
  - Multi-gpu/node inference
  - **Open-source** kernels and models on GitHub
- Extension is difficult – C++ & CUDA w/ no concrete API
- Lacks the support of a formal product
  - Docs, QA, CI/CD, release cadence, etc.

Goals for LLM inference

- Preserve **performance**, **MGMN**, and **open-source**
- Developers **enabled to extend and** support models directly
- Confidence to **deploy in production** as a full NV product


Performant


MGMN


OSS

Extendable

NVIDIA

Full NVIDA product


Robust

TensorRT-LLM optimizing LLM inference

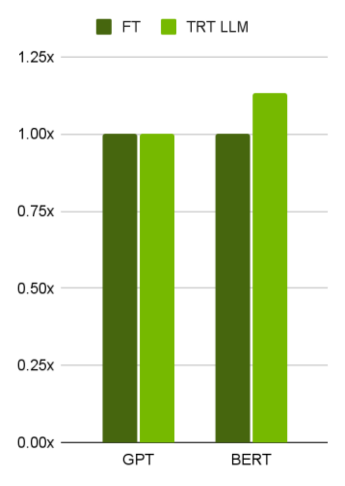
SOTA PERFORMANCE FOR LARGE LANGUAGE MODELS FOR PRODUCTION DEPLOYMENTS

- TensorRT-LLM is an **open-source** library for **optimal performance** on the latest **Large Language Models** for **inference** on NVIDIA GPUs

- TensorRT-LLM wraps TensorRT's Deep Learning Compiler, optimized kernels from FasterTransformer, pre/post processing, and MGMN communication in a simple open-source Python API for defining, optimizing, and executing LLMs for inference in production

SoTA performance

Leverage TensorRT compilation & kernels from FasterTransformer, CUTLASS, OAI Triton, ++



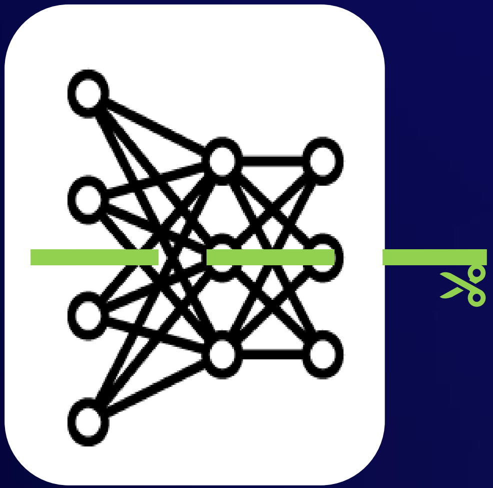
Ease extension

Add new operators or models in Python to quickly support new models & operators in LLMs with optimized performance

```
# define a new activation
def silu(input: Tensor) → Tensor:
    return input * sigmoid(input)

#implement models like in DL FWs
class BertModel(Module)
def __init__(…)
self.layers = ModuleList([…])

def forward (…)
hidden = self.embedding(…)

for layer in self.layers:
hidden_states = layer(hidden)

return hidden
```

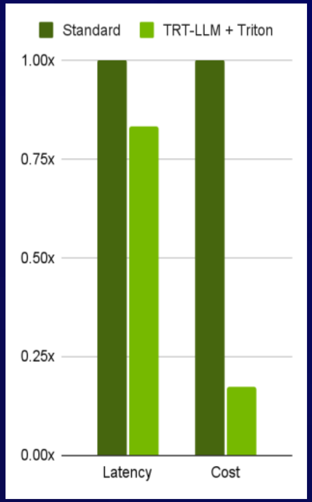
MultiGPU-MultiNode

Seamlessly run pre-partition models across multiple GPUs or multiple nodes. Manually or auto-partition new models



LLM batching with Triton

Maximize throughput & GPU utilization through new scheduling techniques for LLMs



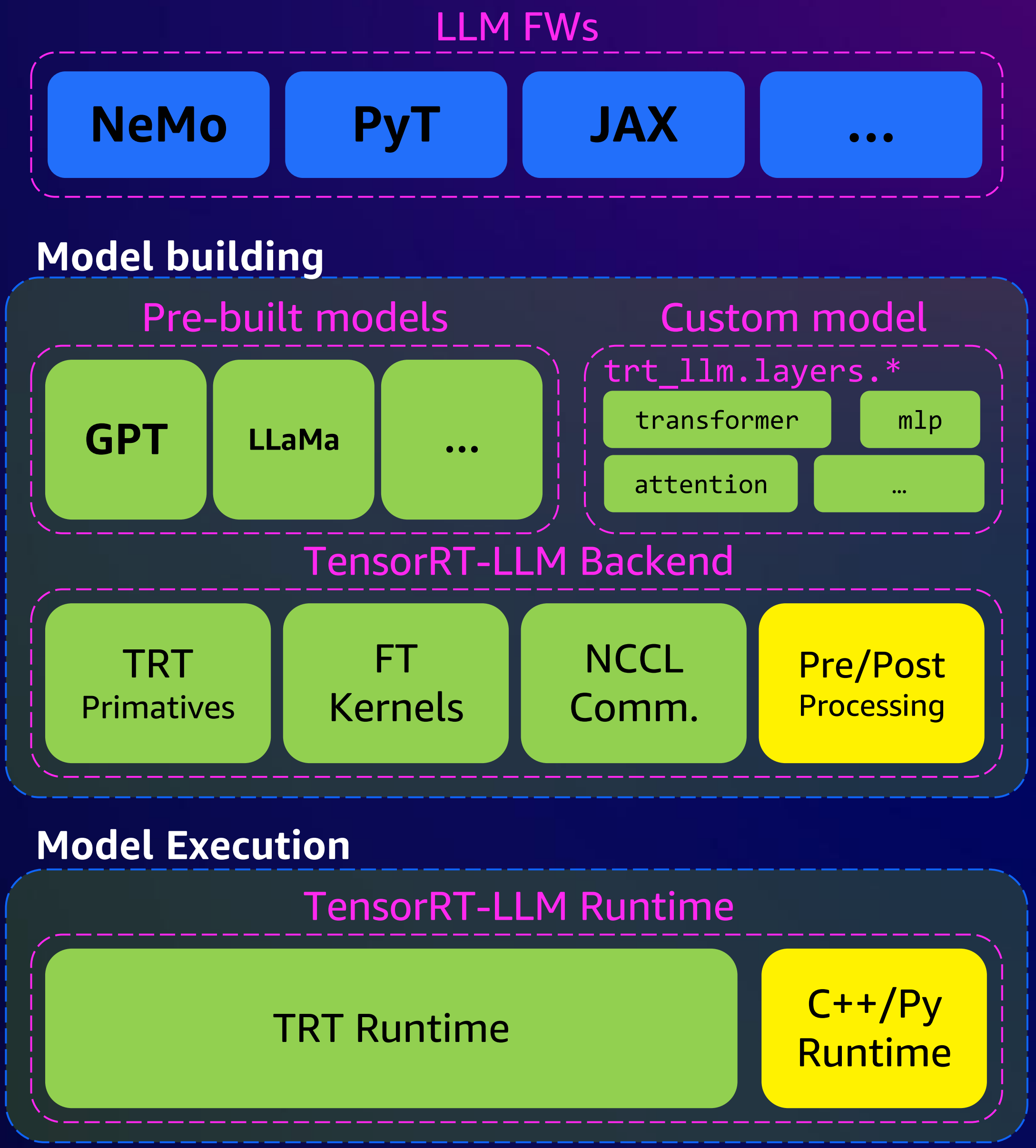
Numbers are preliminary based on internal evaluation

TensorRT-LLM overview

USAGE AND IMPLEMENTATION

TensorRT-LLM Inference

- PyTorch/TE-like building blocks for transformers
  - e.g., fMHA, layerNorm, activations, etc.
  - Built on top of [TensorRT Python API]
- Build arbitrary LLM or deploy pre-built implementations
  - e.g., GPT, LLaMa, BERT, etc.
- MGMN inference
  - Leverages NCCL plugins for multi-device communication
    - Long term will be OOTB in TRT
  - Pre-segmented graphs in pre-built models
  - User can manually segment for custom models
  - Future will allow automatic segmentation across GPUs
- Combines TRT layers, NCCL plugins, perf plugins, & pre/post processing ops into a single object
  - Include tokenization and sampling (e.g., Beam search)



TensorRT-LLM usage

CREATE, BUILD, EXECUTE

- Instantiate model and load the weights
  - Load pre-built models or define via TRT-LLM Python APIs
- Build and serialize the engines
  - Compile to optimized implementations via TensorRT
  - Saved as a serialized engine
- Load the engines and run optimized inference
  - Execute in Python, C++, or Triton

0. Trained model in FW
NeMo, HuggingFace, or from DL Frameworks

1. Model initialization
Load example model, or create one via python APIs

2. Engine building
Optimized model via TRT and custom kernels

TRT-LLM Engine | **TRT Engine** | **Kernel plugins**

3. Execution
Load & execute engines in Python, C++, or Triton

LLM specific serving via Triton

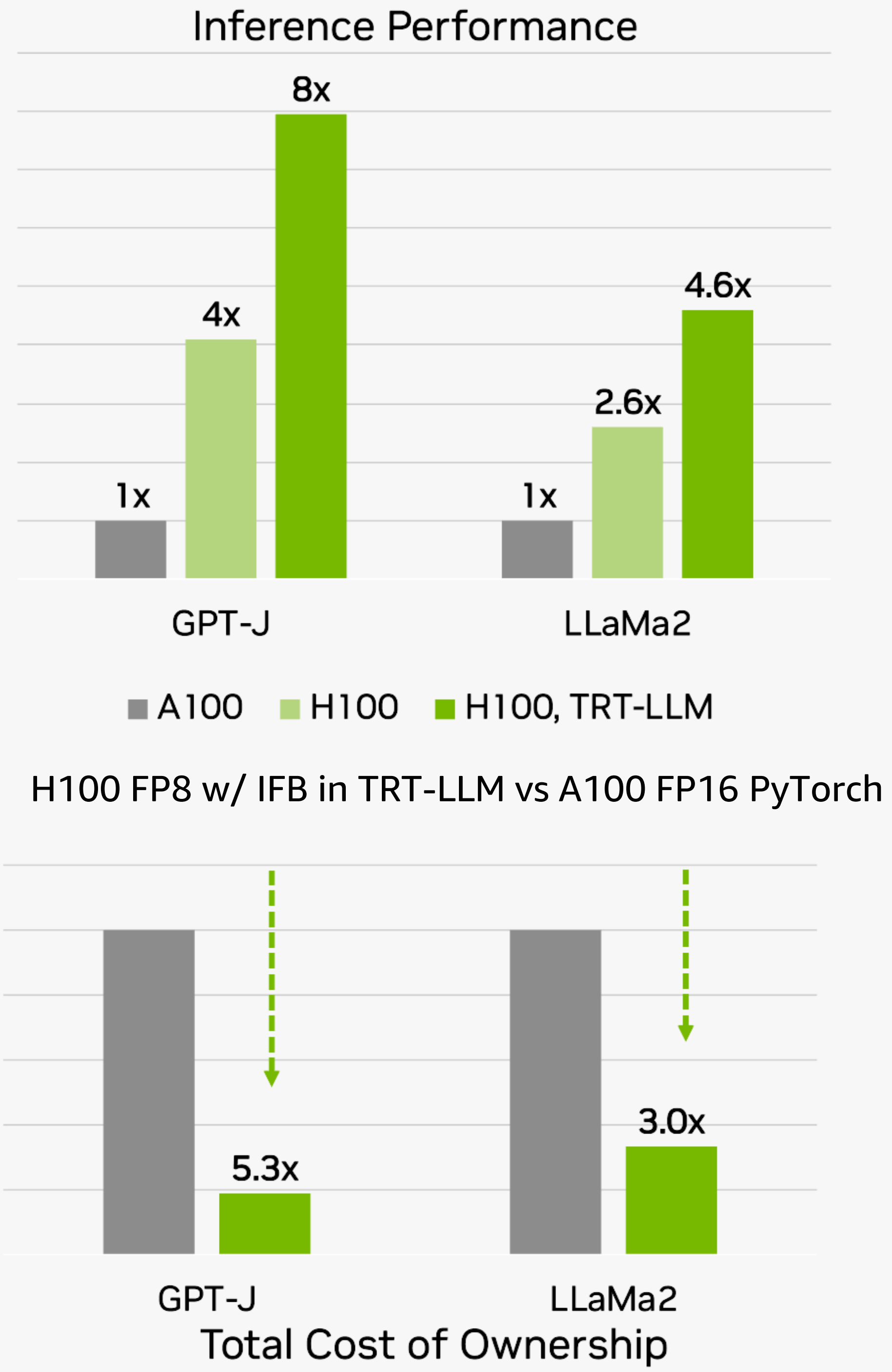
MAXIMIZE GPU UTILIZATION & THROUGHPUT

- TensorRT LLM and Triton together will allow for new LLM specific batch scheduling to optimize throughput and GPU utilization
- Triton backend for TRT LLM
- Inflight batch updates – Orca
  - Replace elements as completed with new requests
  - Significantly improves GPU utilitizionat and throughput for heterogeneous sequence length batches
- Up to **50% faster** and **6x lower cost**
- Stream output tokens for better UX

Inference performance

TENSORRT-LLM HAS SPEED-OF-LIGHT PERFORMANCE

- **8x** faster performance
  - Custom MHAs, Inflight-batching, paged attention, quantized KV cache, and more drive inference performance
- **5x** reduction in TCO
  - FP8 and Inflight-batching performance allows for H100 to improve TCO significantly
- **5x** reduction in energy
  - Performance allows for reduction of energy/inference, allowing for more efficient use of datacenters

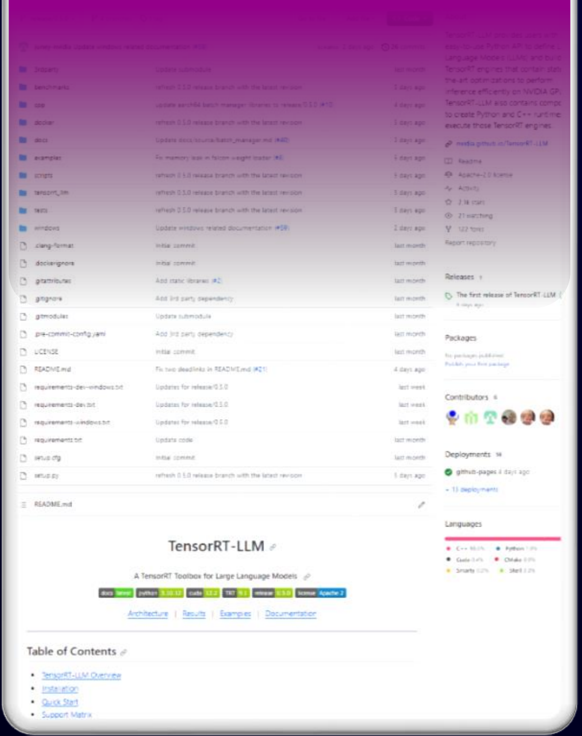


https://developer.nvidia.com/blog/nvidia-tensorrt-llm-supercharges-large-language-model-inference-on-nvidia-h100-gpus/

TensorRT-LLM Available Now!

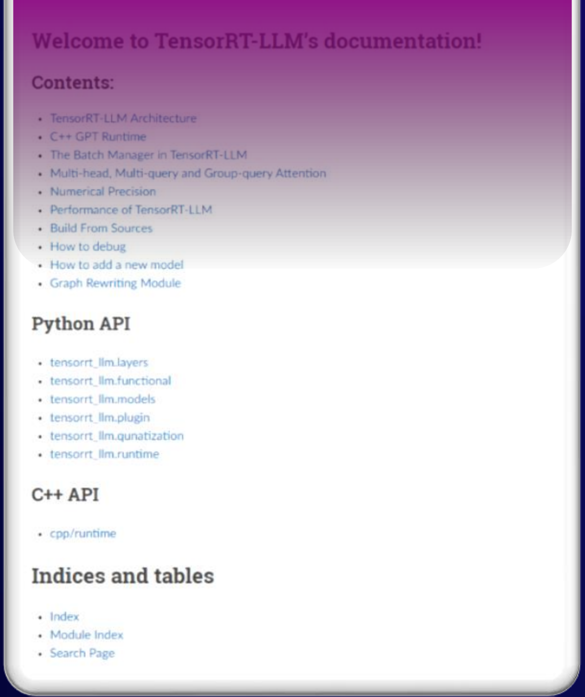
KEY RESOURCES FOR TENSORRT-LM

- TensorRT-LLM Github
- Source for TensorRT-LLM

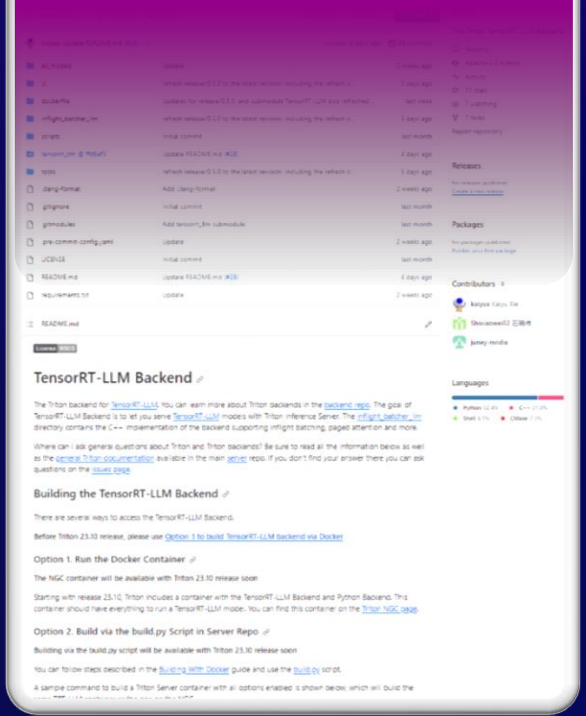


- Getting Started Blog
- Learn to optimize and deploy
- TensorRT-LLM with Triton Server

- TensorRT-LLM documentation
- API docs, arch overviews, & perf data



- NVIDIA Triton Backend
- Source for TensorRT-LLM Triton Backend



Coming Soon to NeMo Framework & NVAIE

Call to action and conclusions

Conclusions

WHAT DID WE LEARN TODAY?

- NVIDIA GPUs power the most compute intensive workloads from computer vision to speech to language and many more
- NVIDIA NeMo Framework is a source open toolkit for large language model training and deployment
- NVIDIA Triton is an inference server for deploying your models
- NVIDIA TensorRT is an SDK for optimizing deep learning models
- NVIDIA TensorRT-LLM supercharges large language model inference

NVIDIA AI enterprise support

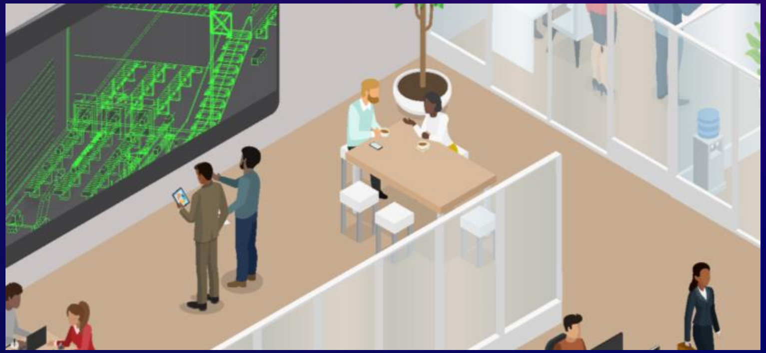
NVIDIA EXPERTS SUPPORTING YOUR AI POWERED BUSINESS

Starting from scratch

Expert guidance

Prioritizing with NVIDIA

Close connection with product and engineering teams



Expanding a proof of concept

Deep AI knowledge

Using best practices to deploy GPUs

Real world experience across industries

Keeping ecommerce up and running

Fast response time

Thank you!

Please complete the session survey in the mobile app

Jiahong Liu
Jiahongl@nvidia.com

Peter Dykas
wdykas@nvidia.com