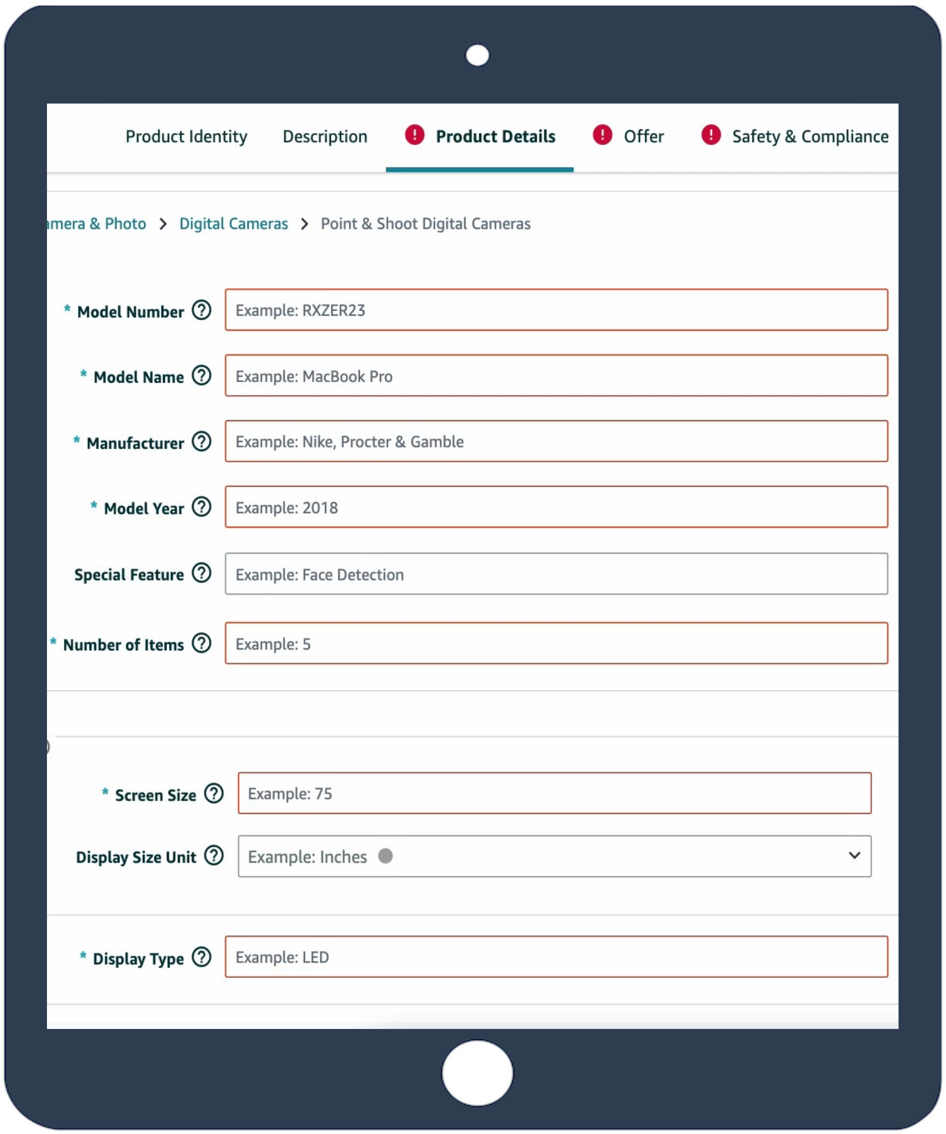
S62390 | March 19, 2024, GTC

Accelerating Generative AI with TensorRT-LLM To Enhance Seller Experience at Amazon

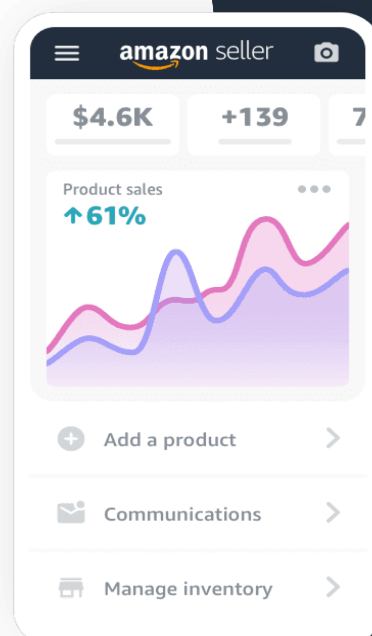
Vishwanath Kumaraswamy
Sr. SDE, Amazon Catalog

Haohang Huang
Sr. AI Engineer, NVIDIA DevTech

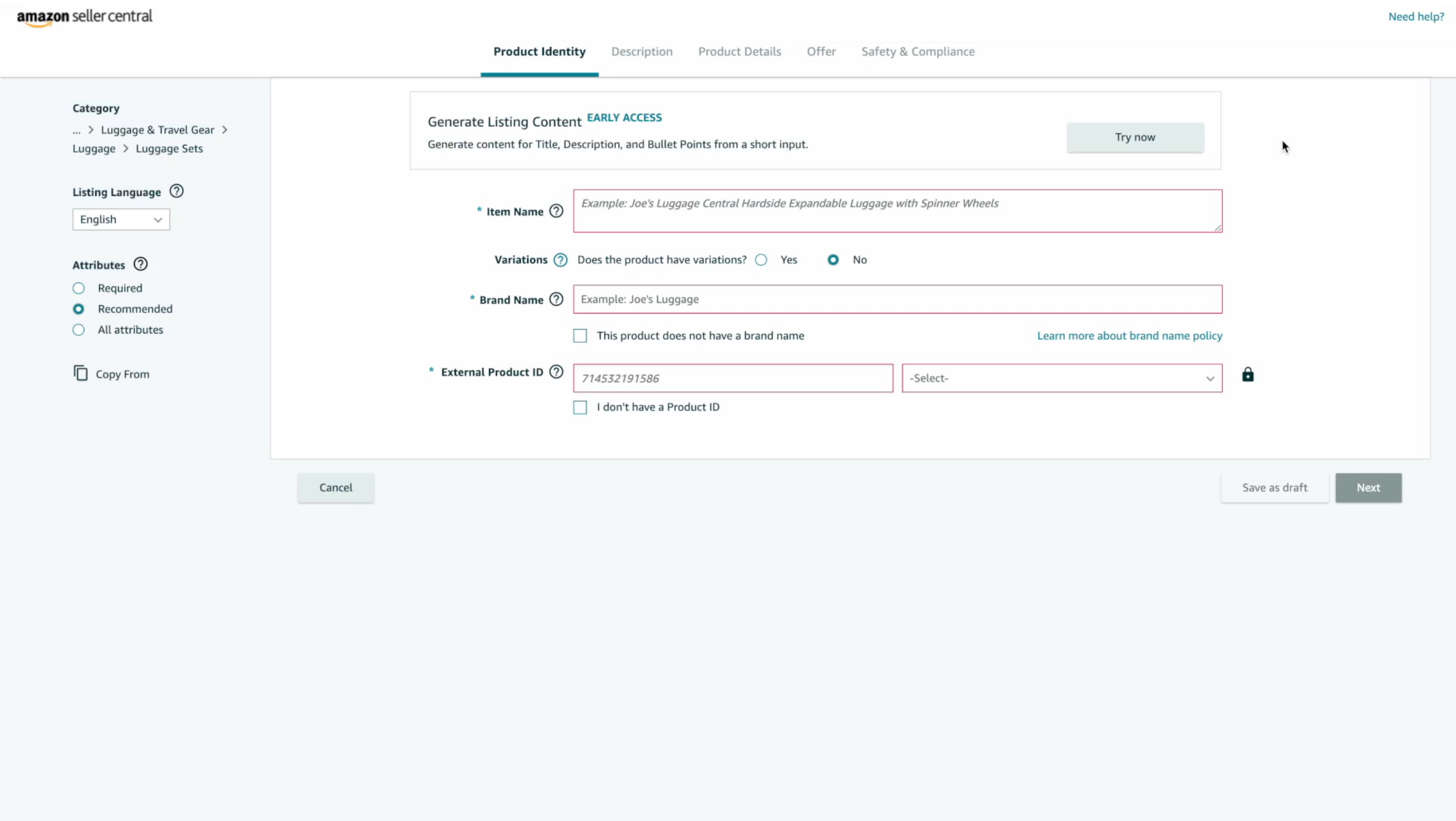
Product Title
4.5 61
Nikon Z 5 Mirrorless Camera with 24-200mm f/4-6.3 VR Lens, 24MP, 4K UHD, Dual SD Slots, Bluetooth & Wi-Fi
Customers
know what they want.
Structured Product Details
Product information
Imaging
Lens
Exposure
Display
Features
Warranty
Item details
Measurements
Product comparison widget
Nikon Z 5 with Telephoto Zoom Lens | Our most compact full-frame mirrorless stills/video...
Nikon Z 30 with Two Lenses | Our most compact, lightweight mirrorless stills/video camera...
Nikon Z 50 with Two Lenses • Compact mirrorless stills/video camera with wide-angle and...
Add to Cart
Price -23% $1,696.95 List: $2,199.95 | $946.95 | -7% $1,246.95 List: $1,346.95
Delivery Get it as soon as Tuesday, Feb 20
Customer Ratings 4.4 66 | 4.5 198 | 4.7 597
Picture Quality 4.5 | 4.6 | 4.6
Auto Focus 4.3 | 4.4 | 5.0
Touch Screen 4.3 | — | 4.6
Product Variations
Style: Camera + 24-200mm + FTZ II Mount Adapter
Camera + 24-200mm + FTZ II Mount Adapter
Camera + 24-50mm + FTZ II Mount Adapter
Camera Body Only

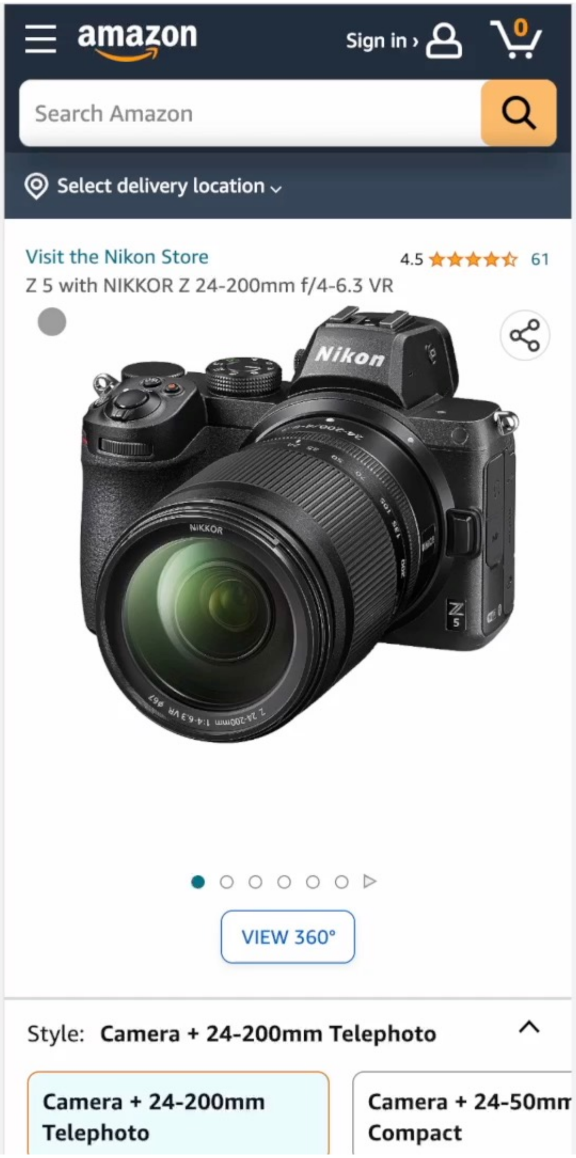


Product details boosts seller's business

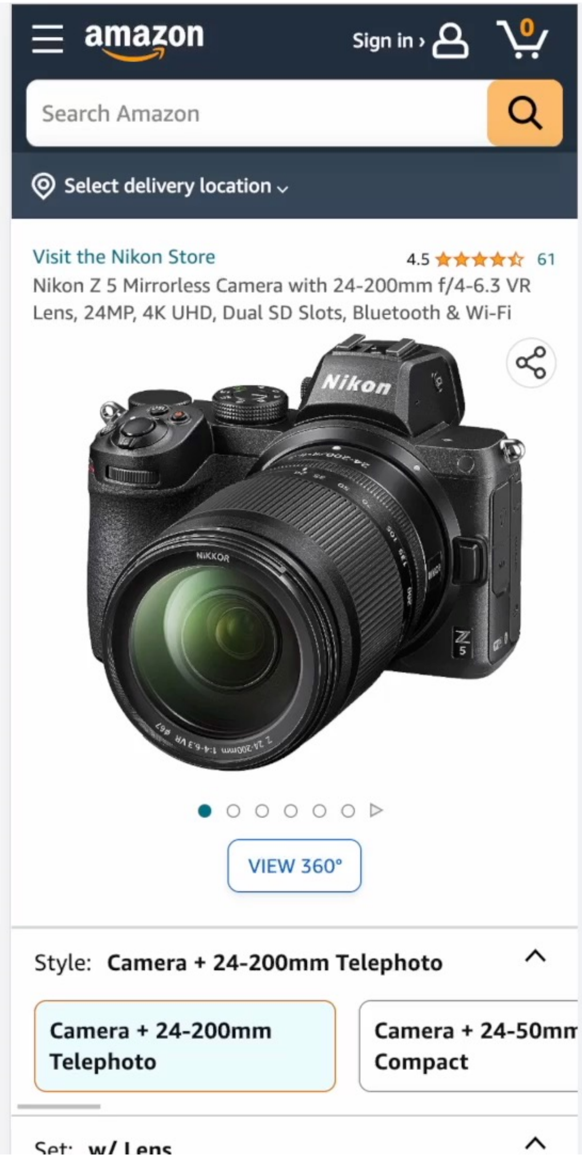


Empowering Sellers and Delighting Buyers:
The Generative AI Approach


amazon seller central
Need help?
Product Identity
Description
Product Details
Offer
Safety & Compliance
Category
... > Luggage & Travel Gear >
Luggage > Luggage Sets
Listing Language
English
Attributes
Required
Recommended
All attributes
Copy From
Generate Listing Content EARLY ACCESS
Generate content for Title, Description, and Bullet Points from a short input.
Try now
* Item Name
Example: Joe's Luggage Central Hardside Expandable Luggage with Spinner Wheels
Variations
Does the product have variations?
Yes
No
* Brand Name
Example: Joe's Luggage
This product does not have a brand name
Learn more about brand name policy
* External Product ID
714532191586
-Select-
I don't have a Product ID
Cancel
Save as draft
Next

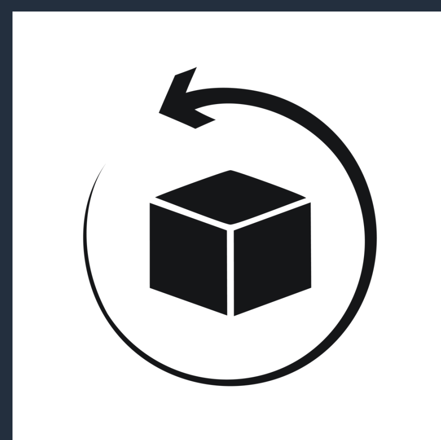


Before

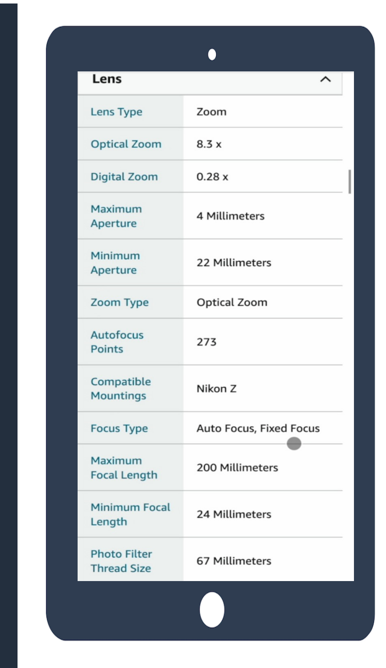


After

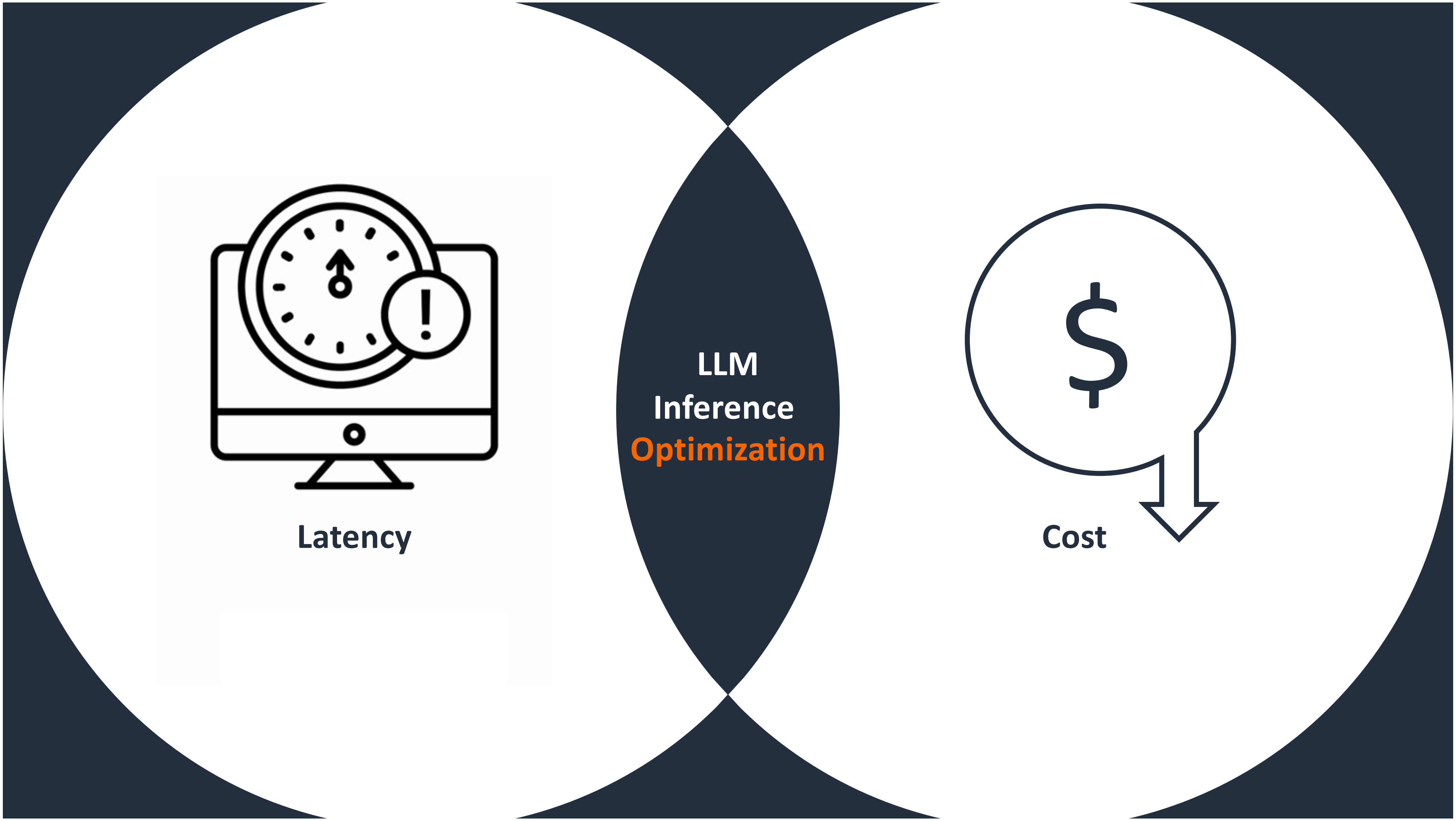
Reduced Seller Effort



Fewer Returns

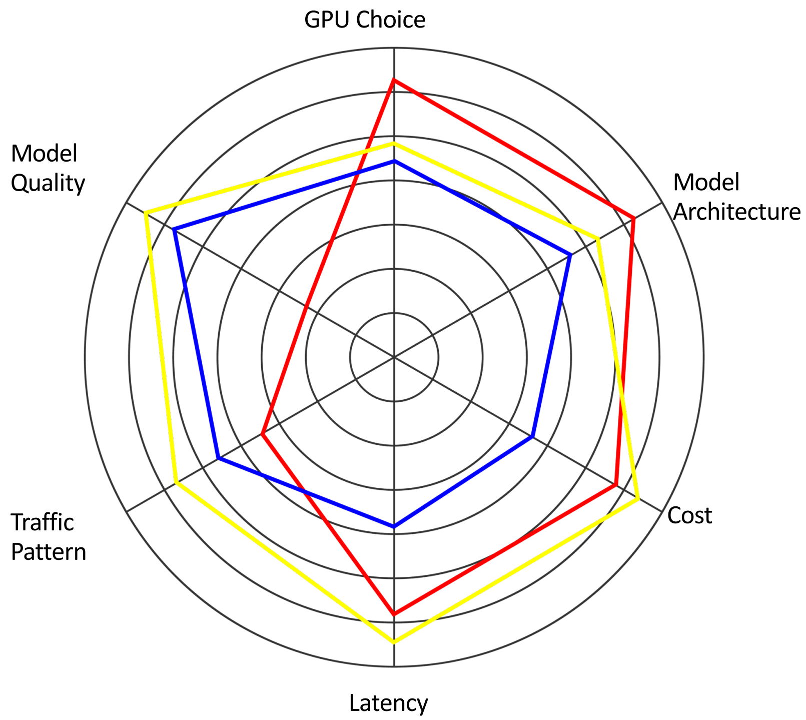


Detailed information for customers


LLM
Inference
Optimization
Latency
Cost

Inference Optimization is

Multi-dimensional



What's in Our Arsenal ?

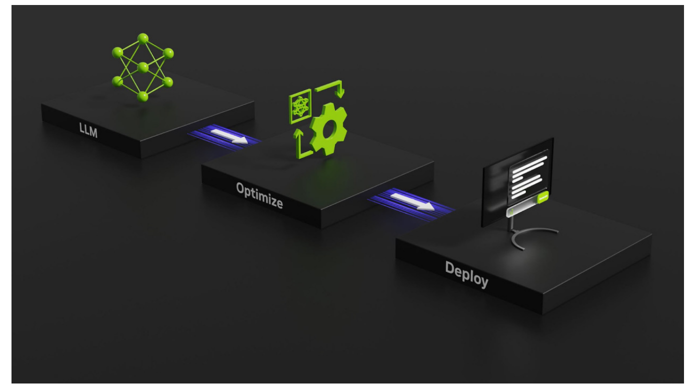
AWS G4 (T4)

AWS G5 (A10G)

AWS P5 (H100)

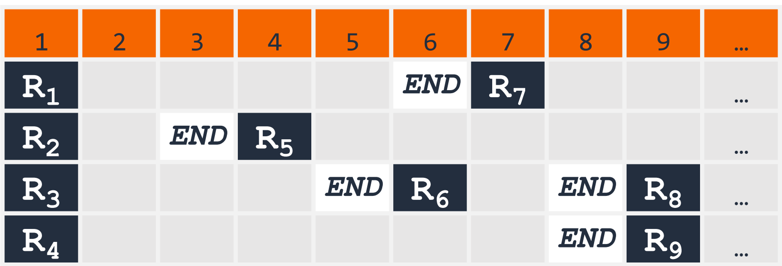
AWS P4 (A100)

TensorRT- LLM



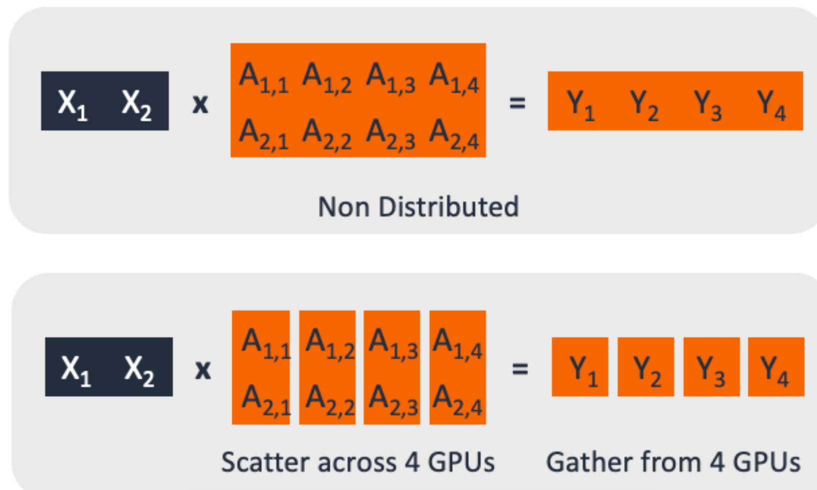
Triton Inference Server

Features of TensorRT-LLM

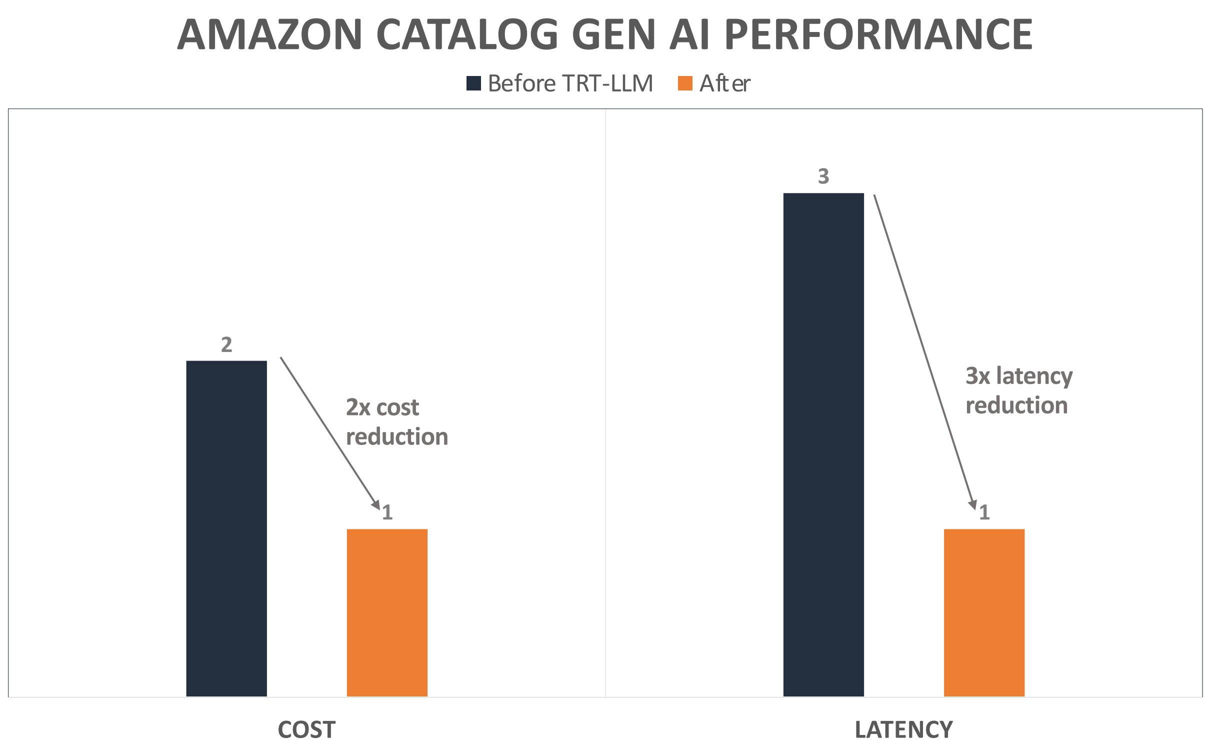


In-flight batching

Quantization Precisions



Tensor Parallelization



Check out our Science paper
https://arxiv.org/abs/2309.05920

What's Next

- INT4 Activation-aware Weight Quantization (AWQ)
- Speculative Decoding
- Continued Collaboration

Dive deep into TensorRT-LLM

TensorRT-LLM Design and Workflow

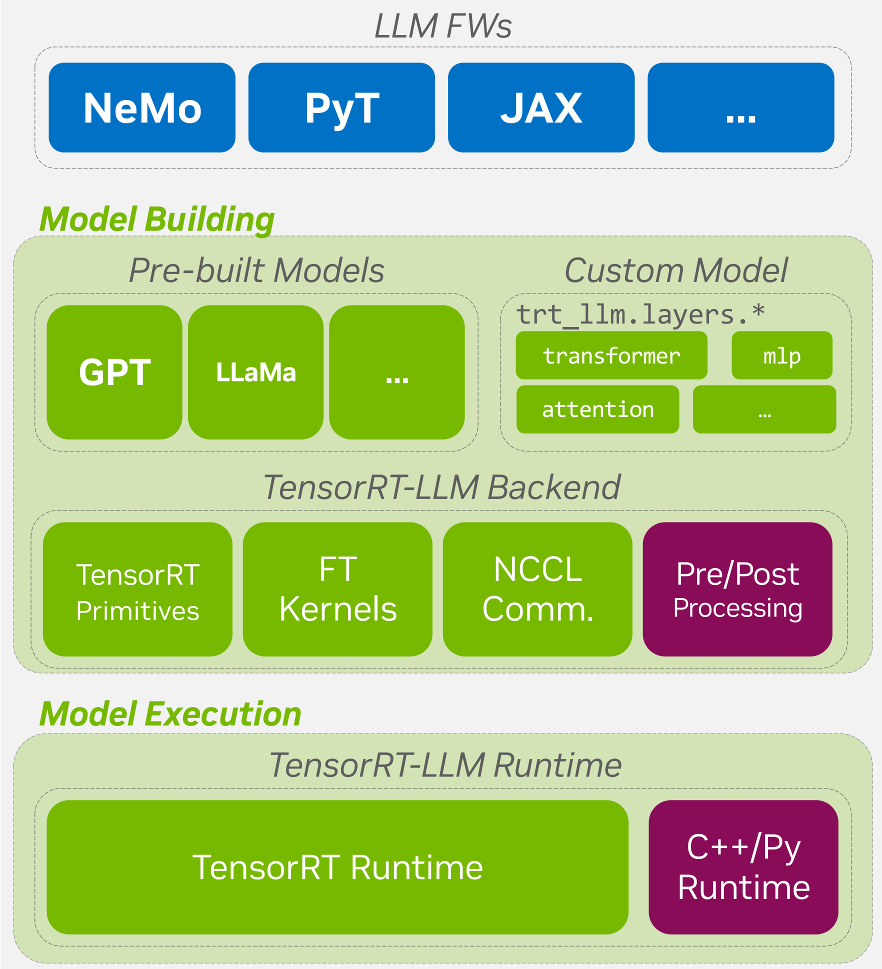
TensorRT-LLM Design

LLM Inference

NVIDIA solution for scalable LLM Inference, built on top of

- **TensorRT** to leverage its deep learning compiler for building and executing LLM models/graphs
- **FasterTransformer** to leverage its optimized kernels for performance
- **NCCL** for scalable inference with for multi-node, multi-GPU communication
- Other components for the customizations of LLM inference, such as CUTLASS

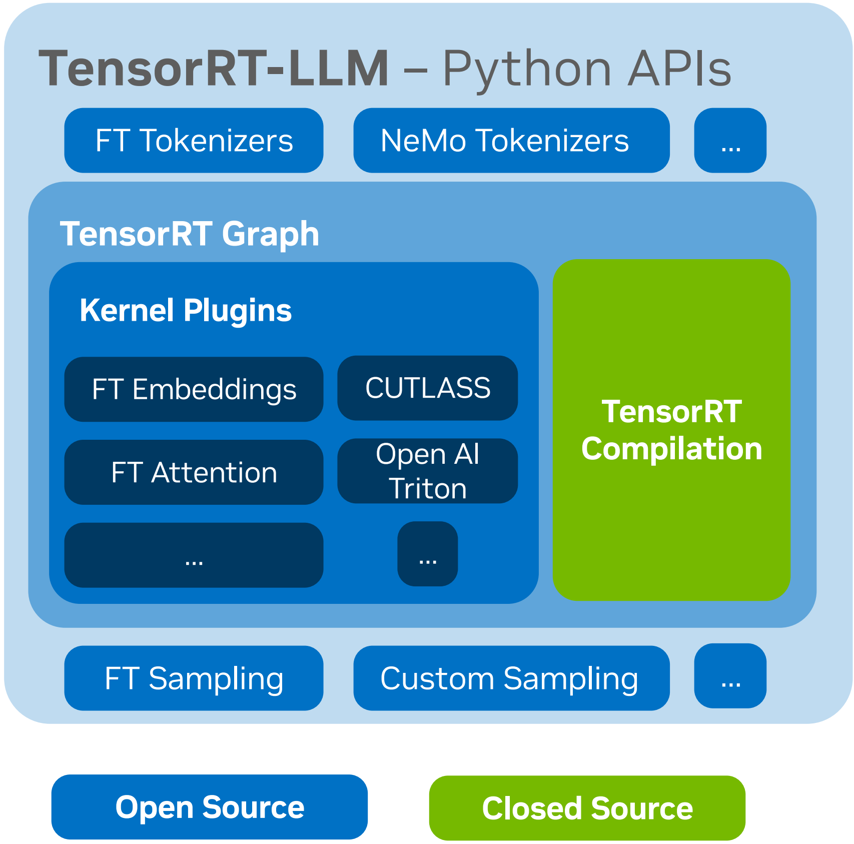
TensorRT-LLM aims to provide the best LLM inference performance on NVIDIA GPU with great flexibility & usability



Open-Source Access

Maintaining access and “hackability” in TensorRT-LLM

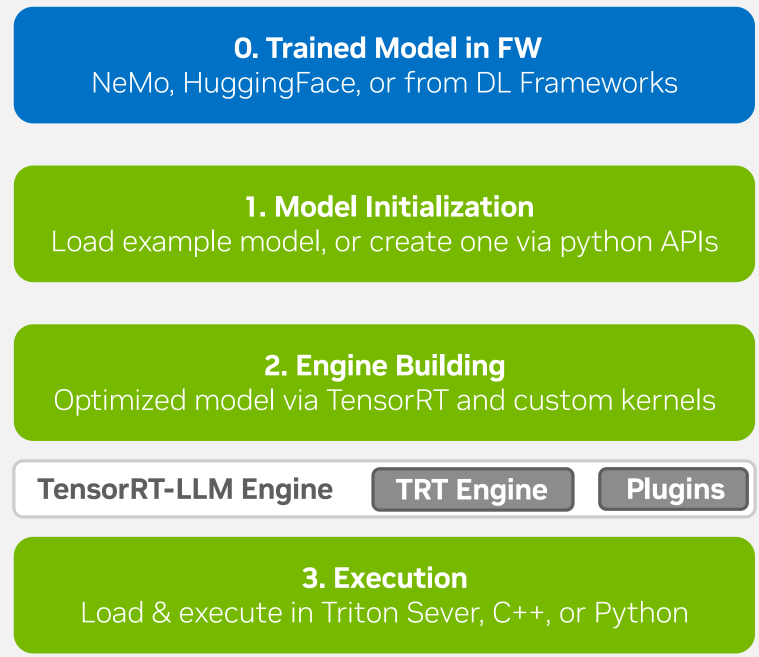
- TensorRT-LLM is open source under Apache2
  - Python APIs for model definition & weight loading
  - All plugins/kernels except those with a risk of exposing HW information, e.g. FMHA, batch manager
  - Pre/post-processing tokenizers & samplers are OSS
  - Etc.
- Construct models with entirely OSS APIs & kernels
- TensorRT Compilation will remain closed source to the public



TensorRT-LLM Workflow

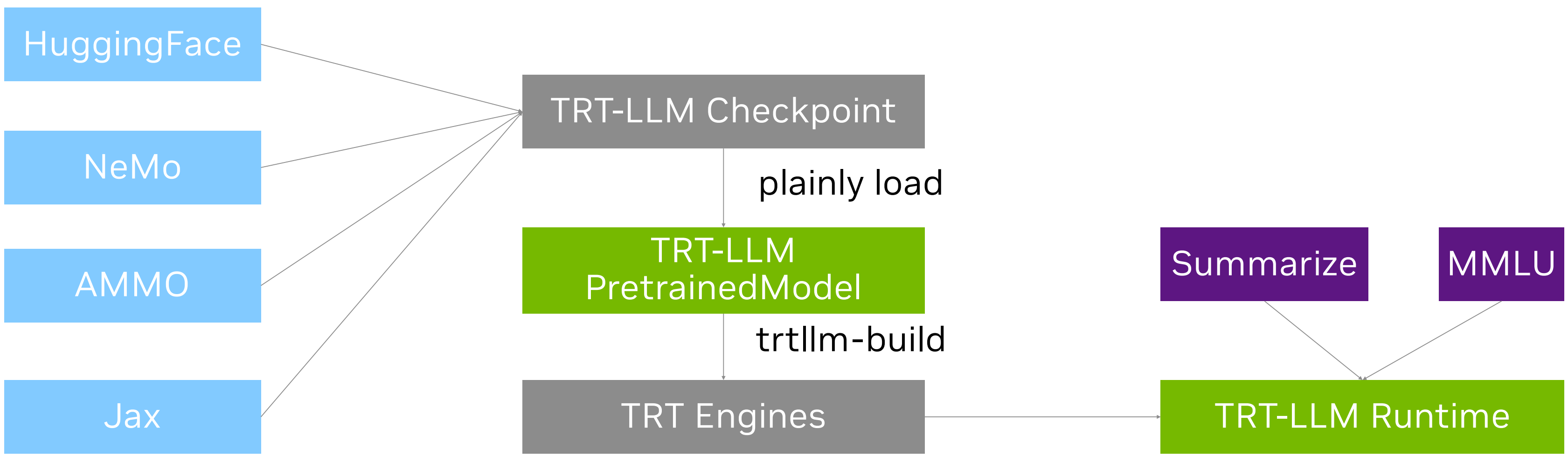
Create, Build, Execute

- Instantiate model and load the weights
  - Load pre-built models or define via TensorRT-LLM Python APIs
- Build & 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



Unified Workflow

- Unified TensorRT-LLM checkpoint format



Model Coverage

Rapid evolution in Gen AI era

- Encoder-only models
  - BERT, RoBERTa
- Decoder-only models
  - GPT, LLaMa, Mixtral, Gemma, Starcoder, ChatGLM, & more
- Encoder-Decoder Models
  - T5 family, BART family, Fairseq NMT, Whisper
- State Space Model (SSM)
  - Mamba
- Multimodal Models
  - BLIP2 w/ OPT, BLIP2 w/ T5, LLaVA, VILA, Nougat, Qwen-VL

TensorRT-LLM aims at a **performant**, **robust**, & **extensible solution** for LLM deployments

LLaMa | GPT | Falcon | Starcoder | ChatGLM | MPT | ...

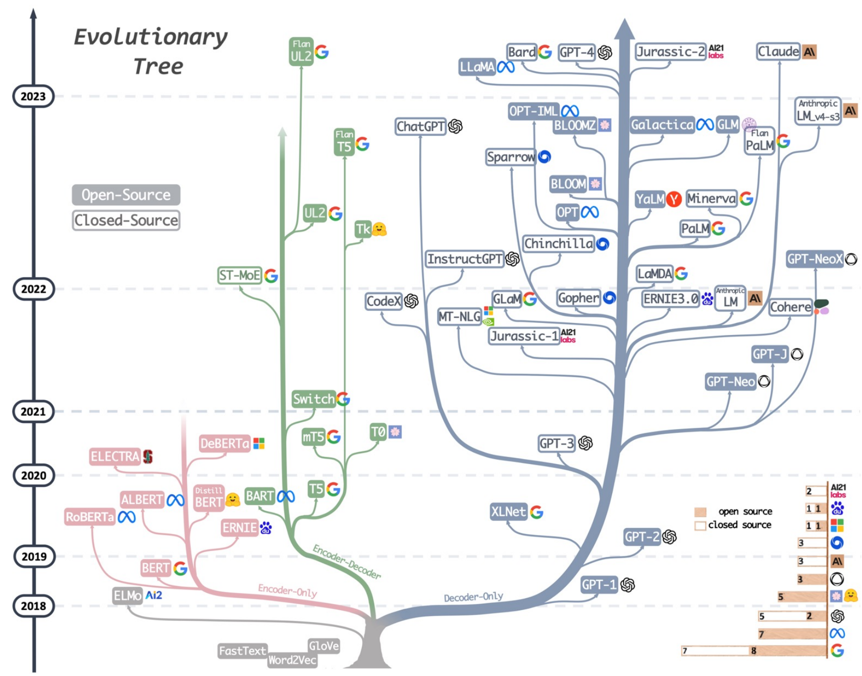


Image from Mooler0410/LLMsPracticalGuide

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

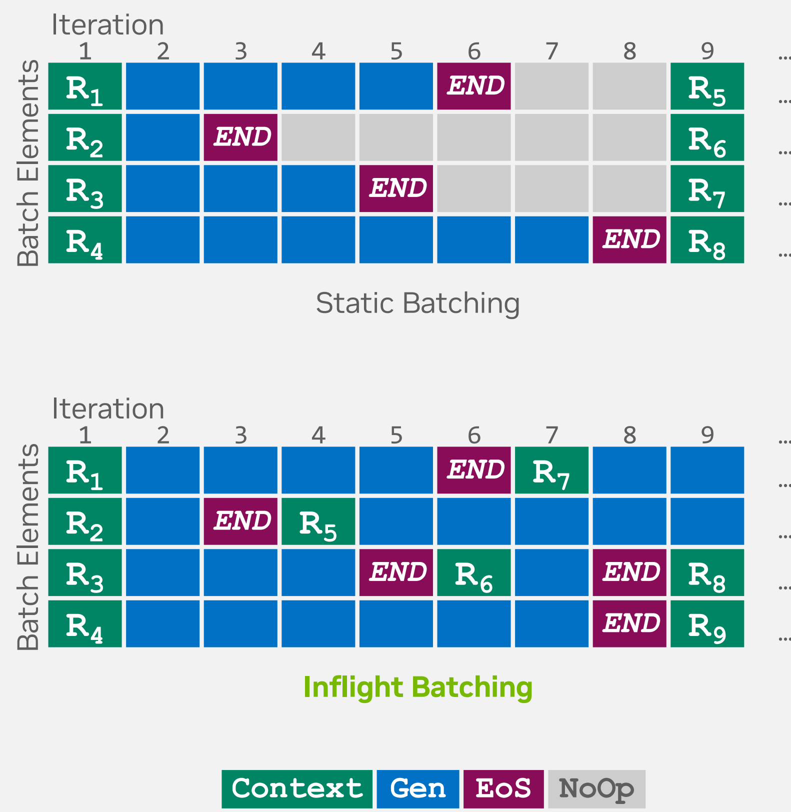
TensorRT-LLM Key Features

Inflight Batching

Maximizing GPU Utilization during LLM Serving

TensorRT-LLM provides custom Inflight Batching to optimize GPU utilization during LLM Serving

- Replaces completed requests in the batch
    - Evicts requests after EoS & inserts a new request
- Improves throughput, time to first token, & GPU utilization
- Integrated directly into the TensorRT-LLM Triton backend
- Accessible though the TensorRT-LLM Batch Manager



KV Cache Optimizations

Paged & Quantized KV Cache

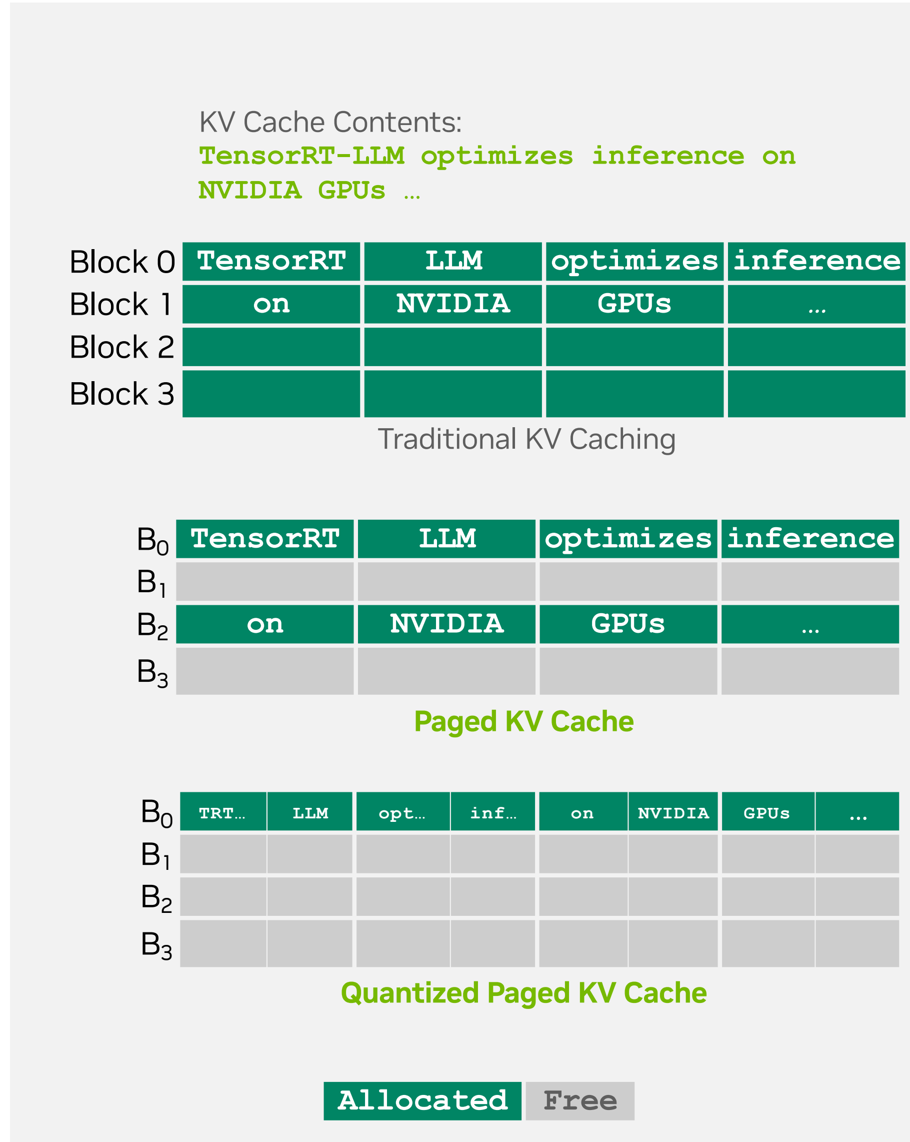
Paged KV Cache improves memory consumption & utilization

- Stores keys & values in non-contiguous memory space
- Allows for reduced memory consumption of KV cache
- Allocates memory on demand

Quantized KV Cache improves memory consumption & perf

- Reduces KV Cache elements from 16b to 8b (or less!)
- Reduces memory transfer improving performance
- Supports INT8 / FP8 KV Caches

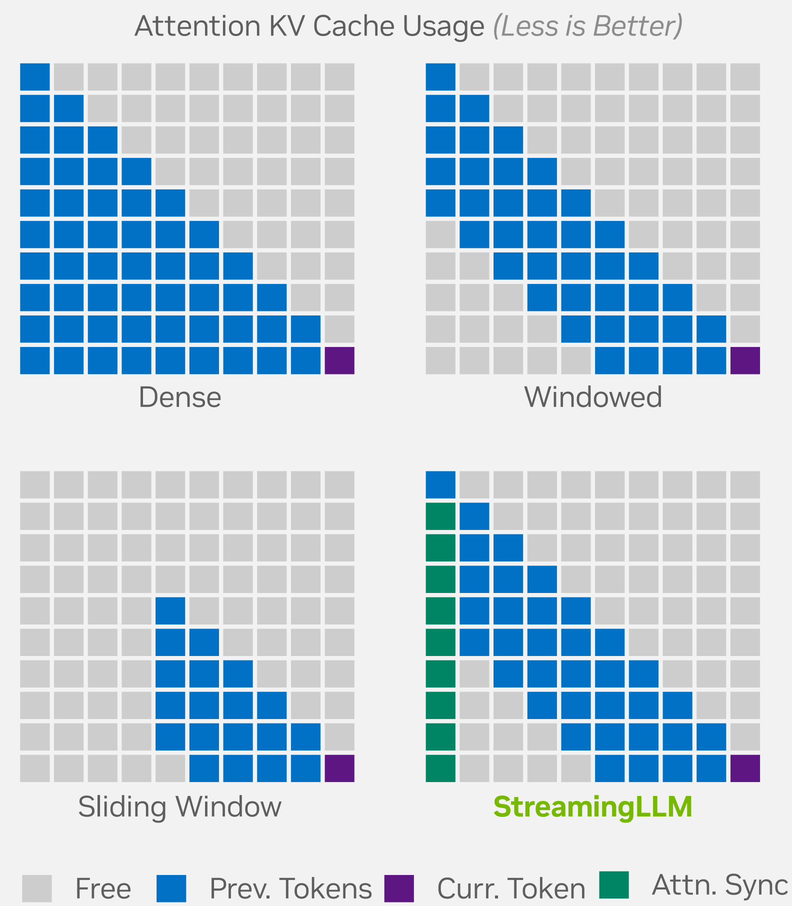
Both allow for increased peak performance



KV Cache & Attention Techniques

(Sliding) Window Attention, & Streaming LLM

- Allow for longer (sometimes unlimited) sequence length
  - Reduces KV Cache Memory usage
  - Avoids OOM Errors
- (Sliding) Windowed Attention evict tokens based on arrival
  - Significantly reduces memory usage
  - Can negatively impact accuracy or require recomputing KV
- Streaming-LLM allows for unlimited sequence length
  - Does not evict Attention Sinks (important elements)
  - KV Cache stays constant size
  - Does not require recompute & does not impact accuracy
  - Particularly beneficial for multi-turn (i.e. chat) use cases

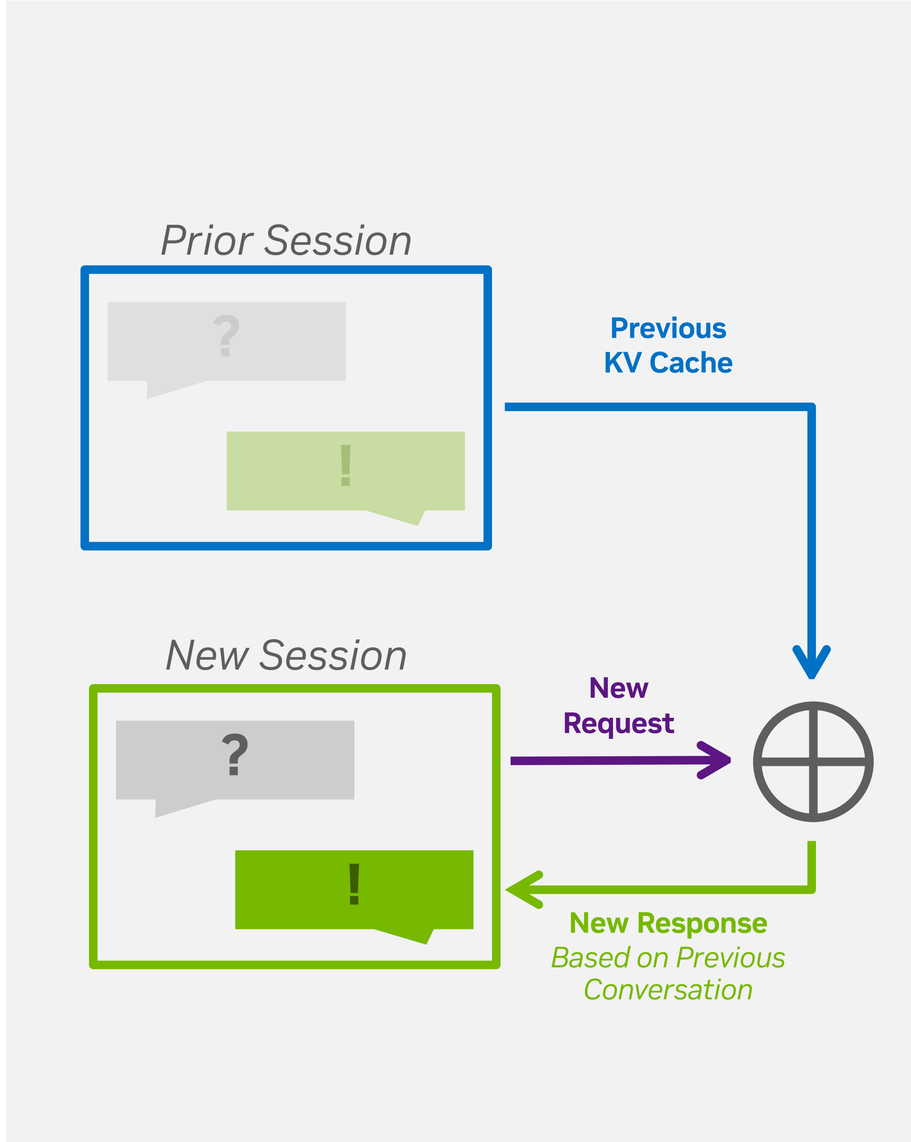


KV Cache Reusage

System Prompt Caching & Block reusage

Allows for interactive/ turn based systems & System Prompts

- Load prior KV cachce blocks to avoid recomupation
  - Saves significant compute
  - Reduces Start-up time
- Block resuage allows for turn-based (chat) applications
  - Allows for additional options for intelligently reusing blocks
- System prompts allows for a preset KV cache for the LLM
  - E.g. to give rules, personality, or prior knowledge



Quantization

How to Chose a Precision

- Weight quantization reduces memory footprint & traffic
  - Reduces latency
  - Can fit larger models
  - Costs compute time to unpack the weights
- Activation quantization saves on compute
  - Improves throughput
  - Can run larger batch sizes
- W*X*A*Y* = weights quantized to *X* bits, and activations to *Y*
- Quantization documentation. Currently, all PTQ (post-training quantization) methods

Method	Performance Improvement small batch BS <=4	large batch BS>=16	Accuracy impact	Calibration time
FP8 (W8A8)	Medium	Medium	**Very low / None**	**O(1min)**
INT8 SQ (W8A8)	Medium	Medium	Medium	O(1min)
INT8 WO (W8A16)	Medium	***None***	**Low**	*None*
INT4 WO (W4A16)	**High**	***None***	**High**	***None***
INT4 AWQ (W4A16)	**High**	***None***	**Low**	O(10min)
INT4 GPTQ (W4A16)	**High**	***None***	**Low**	O(10min)
INT4-FP8 AWQ (W4A8)	**High**	Medium	**Low**	O(10min)

SQ = Smooth Quant
WO = Weight Only
AWQ = Activation Aware Quantization

INT4/INT8 weight-only

INT8 weight-only, for example

- Quantize only weights to INT8
- Storage in INT8, compute in fp16/bf16, i.e. W8A16
- 2x reduction in weight storage
  - Few memory traffics
  - High throughput with large batch sizes
- Straightforward and easy to be implemented
- Good generalization ability

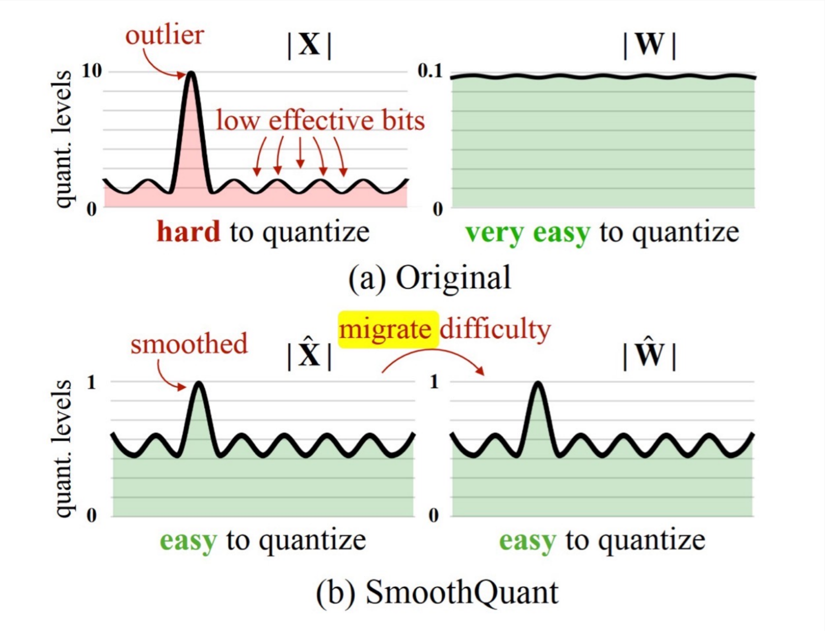
SQ (SmoothQuant)

"Smooth" $$\mathbf{Y} = (\mathbf{X}\mathrm{diag}(\mathbf{s})^{-1}) \cdot (\mathrm{diag}(\mathbf{s})\mathbf{W}) = \hat{\mathbf{X}}\hat{\mathbf{W}}$$

- Easy to quantize weights vs. Hard to quantize activations w/ outliers
- Smooth is performed **offline**, migrating the quantization difficulty from activations to weights

"Quantization"

- Per-tensor or per-channel for weights
- Per-tensor or per-token for activation
- SmoothQuant is a typical algorithm of HW&SW co-design
- Efficient to be implemented on GPU

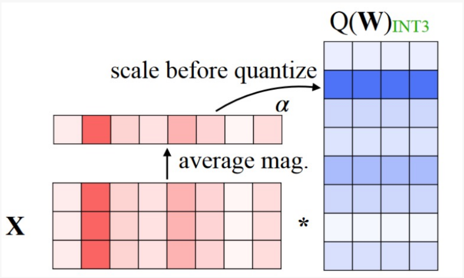


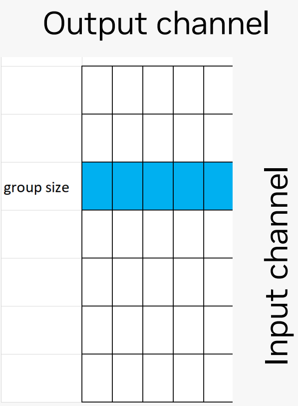
GPTQ

- A layer-wise quantization algorithm minimizing a layer-wise reconstruction loss
  - Loss function $||WX - W_qX||$
- Quantization
  - Quantize weights only to INT4, compute in fp16, W4A16
  - Compatible with essentially any choice of quantization grid, such as grouping
- Co-design for efficient deployment with GPU
  - Lazy batch-updates to improve the compute intensity

AWQ (Activation-aware Weight Quantization)

- Weights are not equally important for LLM's performance. 0.1%~1% salient weights
- Salient weights are determined by activation distribution instead of weight distribution – "Activation-aware"
- Scale up the salient weight before quantization
  - Scalers are determined solely by activation
  - Per-(output) channel scaling
- Grouped quantization along the input channel direction
  - Similar as the GPTQ but with no weight reconstruction





Quantization

How to Chose a Precision

- Best precision varies by application
  - FP8 activations generally provides best performance → Hopper Transformer Engine
- Small batch: bandwidth bound
  - WO is preferred
- Large batch: bandwidth and compute bound,
  - Activation quantization is needed. Try **FP8 → INT8 SQ** → AWQ → GPTQ
- KV cache quantization is also recommended, FP8 / INT8

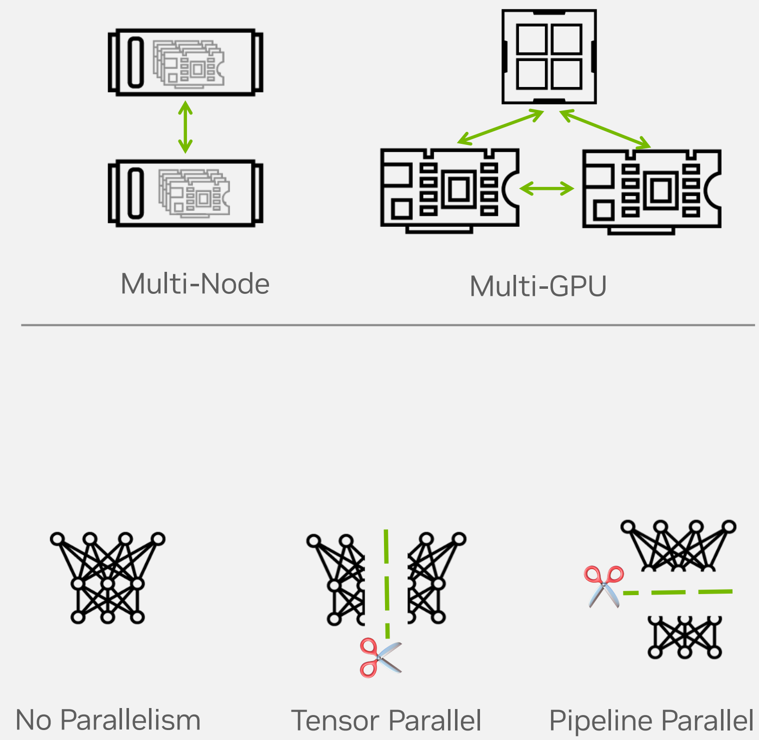
Method	Performance Improvement small batch BS <=4	Performance Improvement large batch BS>=16	Accuracy impact	Calibration time
FP8 (W8A8)	Medium	Medium	**Very low / None**	**O(1min)**
INT8 SQ (W8A8)	Medium	Medium	Medium	O(1min)
INT8 WO (W8A16)	Medium	***None***	**Low**	*None*
INT4 WO (W4A16)	**High**	***None***	**High**	***None***
INT4 AWQ (W4A16)	**High**	***None***	**Low**	O(10min)
INT4 GPTQ (W4A16)	**High**	***None***	**Low**	O(10min)
INT4-FP8 AWQ (W4A8)	**High**	Medium	**Low**	O(10min)

SQ = Smooth Quant
WO = Weight Only
AWQ = Activation Aware Quantization

Multi-GPU Multi-Node

Sharding Models across GPUs

- Supports Tensor & Pipeline parallelism
- Allows for running very large models (tested up to 530B)
- Supports multi-GPU (single node) & multi-node
- TensorRT-LLM handles communication between GPUs
- Examples are parametrized for sharding across GPUs

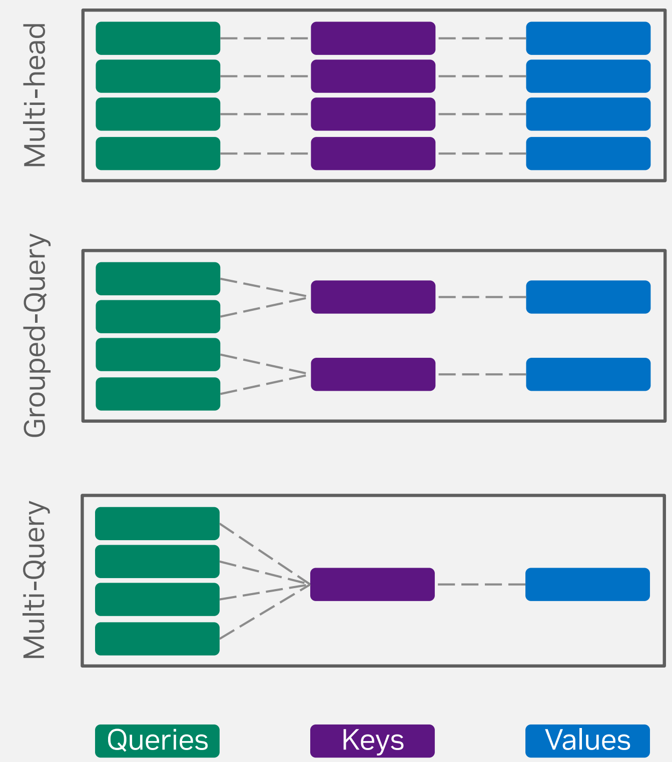


NVIDIA

Optimized Attention

Custom Implementations for Attention

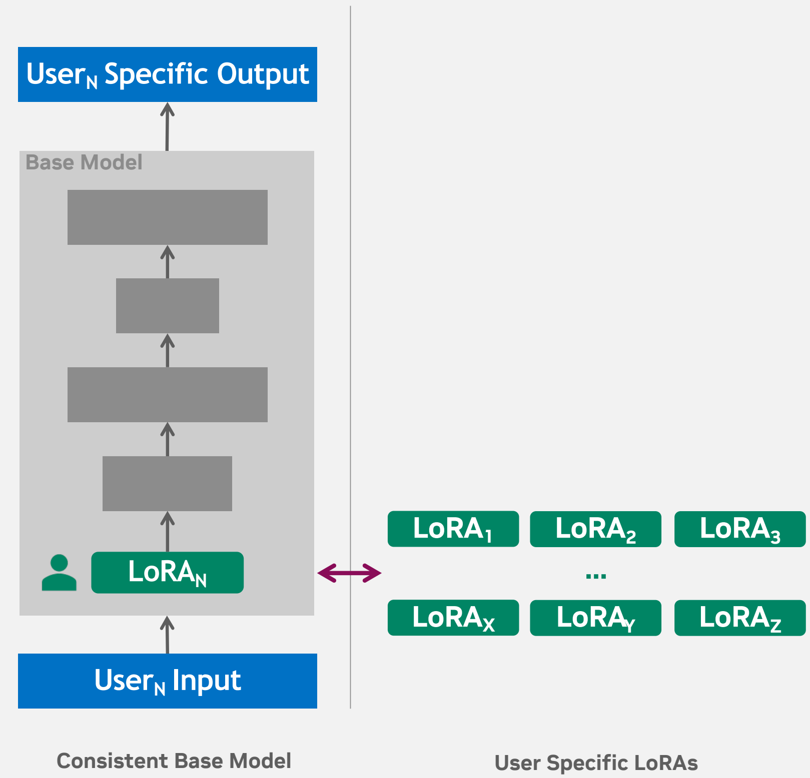
- Custom optimized CUDA kernels for Attention
  - Similar to FlashAttentionV2
- Optimized for A100 & H100
- Kernels for Encoder & Decoder, as well as context
- Supports MHA, MQA, GQA



LoRA & Customization

Efficiently Supporting Customer User Experience

- Low-Rank Adaptation (LoRA) & Prompt tuned models are support in TensorRT-LLM
- Support multiple customers with a single model
- Dynamically swap LoRAs at runtime



Dynamically Swap LoRAs based on User

And Much More

- Speculative decoding
- Chunked context
- SLoRA, QLoRA
- ...
- Continued collaboration with Amazon team for all the exciting features

Acknowledgements

Robert Tekiela
VP, Selection & Catalog Systems, Amazon Selection & Catalog Sys

Mary Beth Westmoreland
VP, WW Seller Partner Experiences

&

Amazon Catalog System Services Organization

NVIDIA Amazon Account Team
Kshitiz Gupta, Jiahong Liu, Eliuth Triana, Anish Mohan

NVIDIA TensorRT-LLM Team
Nick Comly, June Yang

NVIDIA DevTech Team
Julien Demouth, Chandler Zhou, Murat Guney