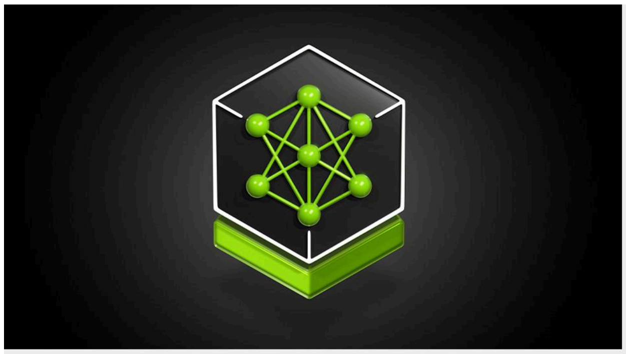
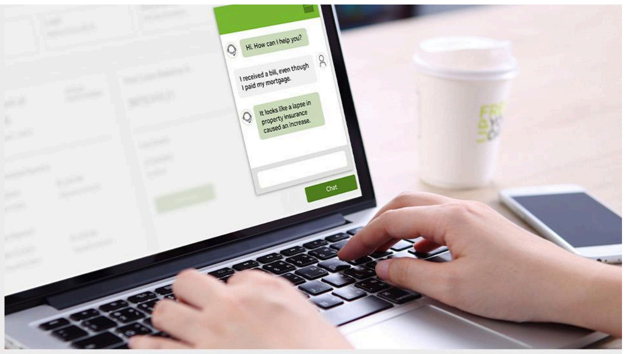
The IT Leader’s Guide to AI Inference and Performance

E-book




Hi. How can I help you?
I received a bill, even though I paid my mortgage.
It looks like a lapse in property insurance caused an increase.

```
cv_meta_sim > sim2d > main.py > ...
from simulator2D import Simulator2D
from visualizer2D import Visualizer
from analytics2D import Analytics
from scene2D import *
from utils import state_from_f
from time import sleep
    datetime
```

This e-book is your essential roadmap to navigating the complexities of AI infrastructure in today's rapidly evolving technology landscape. Tailored for IT leaders embarking on their AI journey, this guide demystifies how AI use cases directly shape performance measurement and infrastructure optimization.

AI workloads demand infrastructure that delivers high-performance, reliability, and efficiency. But how do you determine whether your infrastructure can meet the demands of your specific AI use case?

This e-book explores:

- **The Role of Use Cases in Performance Measurement:** Understand how different AI applications drive unique infrastructure requirements.
- **Key Metrics for AI Inference:** Learn what to measure — latency, throughput, energy efficiency, and more — to ensure success.
- **Best Practices for Optimizing Infrastructure:** Get actionable strategies to align your technology stack with your business goals.

With insights, frameworks, and examples, *The IT Leader's Guide to AI Inference and Performance* equips you with the knowledge to evaluate, deploy, and scale AI solutions effectively — making it a must-read for decision-makers who want to lead with confidence in the AI era.

Table of Contents

What is Inference Performance and Why It Matters

Inference is what brings AI to the real world, solving advanced deployment challenges to unlock applications — from writing code to summarizing long documents, and even generating images and short videos. The potential for inference to automate work ows, create new business models, and power groundbreaking products is transformative — especially for CIOs and IT leaders tasked with driving AI innovation within their enterprise.

However, with this opportunity comes a key challenge: managing the recurring costs associated with scaling inference workloads while balancing latency with performance.

Tackling the Cost Per Token Challenge

When maintaining on-premises AI infrastructure, every inference request served must generate su cient revenue to cover the costs of hardware accelerators, inference software, ongoing inference optimization and management, and more. For those utilizing infrastructure-as-a-service, the nancial burden shifts to monthly or annual accelerated compute instances and inference Software-as-a-Service (SaaS) fees. Regardless of the deployment model, ensuring that the ROI from AI inference justi es these costs is a critical concern for IT leaders.

Our mission is to help organizations address these challenges by optimizing AI performance and driving down inference cost. By enhancing the e ciency of AI inference systems, NVIDIA enables businesses to reduce costs while maintaining the scalability, performance, and user experience required to meet evolving AI deployment demands.

Inference costs are tightly correlated with two key factors. The rst is the performance, or throughput, of the AI platform itself. The more requests a system can handle e ciently, the better organizations can spread costs across incoming requests, lowering the cost per request. However, as AI models generate tokens in response to requests, whose complexity and type can vary signi cantly, *cost per token* becomes an essential metric for assessing both system performance and overall cost e ciency.

However, lowering cost per token must be balanced with maintaining a high-quality user experience. Maximizing request volume at the expense of user experience can reduce adoption of the AI-enabled application or service. A poor user experience can lead to customer churn, which, in turn, erodes revenue. Therefore, it's essential to measure both cost per token and system *latency*.

Latency is typically measured across two dimensions — *Time to First Token* (TTFT) and *Time Per Output Token* (TPOT). As we will explore later in more detail, optimizing these dimensions often presents trade-o s. Increasing one can lead to the deterioration of another, so it's crucial for organizations to strike the right balance.

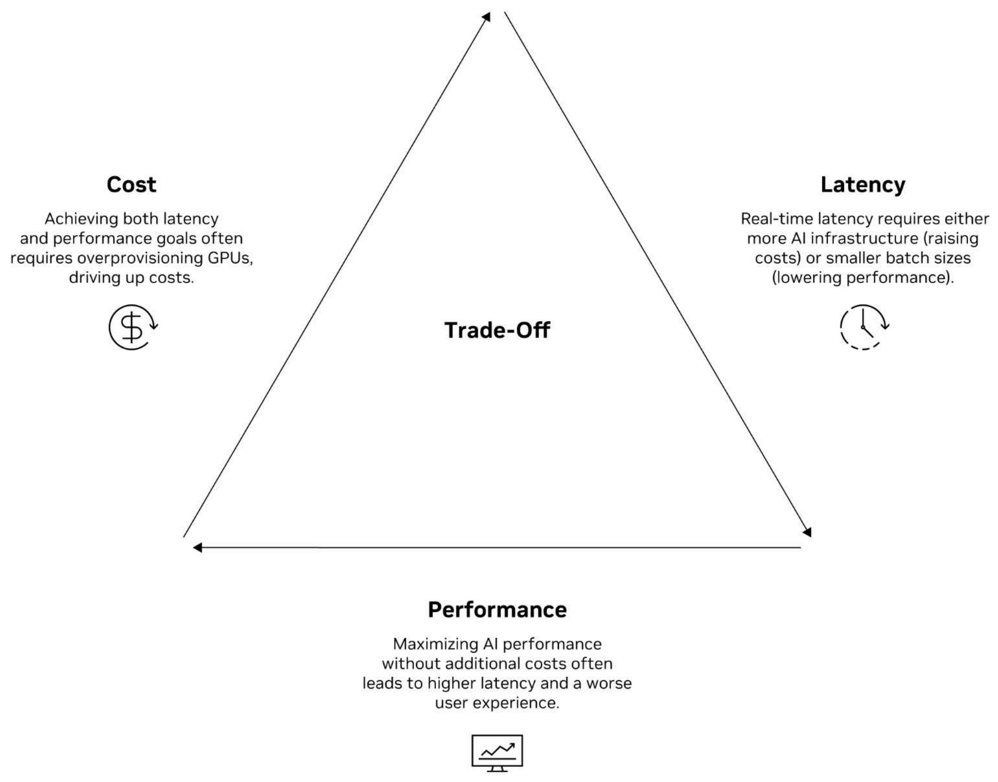

Cost
Achieving both latency and performance goals often requires overprovisioning GPUs, driving up costs.
Latency
Real-time latency requires either more AI infrastructure (raising costs) or smaller batch sizes (lowering performance).
Trade-Off
Performance
Maximizing AI performance without additional costs often leads to higher latency and a worse user experience.

Figure 1. Optimizing Time to First Token and Time Between Tokens Often Presents Three Tradeo s If Not Correctly Balanced

To account for these interdependencies, IT leaders are starting to measure *Goodput* — de ned as the throughput achieved by a system while maintaining target levels of TTFT and TPOT. This metric allows organizations to evaluate performance in a more holistic manner, ensuring that throughput, latency, and cost are aligned to support both operational e ciency and an exceptional user experience.

The second critical factor driving Inference costs is time to market. The AI landscape is evolving rapidly, and being rst to market can provide a signi cant competitive edge. However, an inference software stack that is unreliable, poorly tested, or lacks comprehensive documentation and strong community support can lead to costly delays, integration challenges, and steep learning curves for internal development teams.

These combined obstacles hinder an organization's ability to capitalize on emerging opportunities. While quantifying the exact impact of these issues on inference costs can be challenging, the effect is tangible, and IT leaders should incorporate it into their overall inference cost management strategy.

As you continue reading, we will dive deeper into how you can address these factors to optimize AI Inference performance, reduce costs, and maximize the long-term value of your AI investments.

How Use Cases Impact Inference

It can be tempting to base hardware infrastructure decisions on a single benchmark when evaluating the performance of di erent AI accelerators. However, this oversimpli ed approach often leads to rising system costs over time. Initially provisioned systems may fail to meet the performance demands of a speci c use case, necessitating future investments in extra resources. This occurs because inference workloads from di erent use cases have varying demands for accelerated compute resources and software stack optimizations — factors that are crucial for achieving the desired total cost of ownership (TCO) and ensuring optimal user interactivity.

Chatbot

Consider, for instance, LLM-powered chatbots commonly used in e-commerce and customer service work ows. These chatbots are responsible for providing quick responses to short user inquiries and concerns, making responsiveness imperative to prevent user churn. In this context, a fast TTFT – a measure of how quickly the chatbot begins outputting a response – is essential for delivering a positive user experience, while a moderate TPOT that matches or slightly surpasses human reading speed is considered acceptable. Typical production implementations of chatbots serve very large user bases and have large batch sizes, grouping multiple user requests together and sending them to the AI system for inference in a single request.

Summarization

On the other end of the spectrum are document summarization use cases, which are gaining traction across various industries, such as healthcare for summarizing medical research papers, news and media for summarizing key events and developments, and web conferencing providers for generating action items from video meetings. In these scenarios, the input sequences are lengthy, ranging from a few pages to hundreds, with the focus on quickly generating the full summary rather than instant responsiveness. As a result, the system must be optimized for fast TPOT, far exceeding human reading speed, as users are more tolerant of longer TTFT — particularly at long input sequence lengths. To achieve fast TPOT, the batch sizes in these cases tend to be smaller, requiring advanced software stack optimizations to maximize throughput at low batch sizes.

Question-Answering

Question-answering bots share many similarities with chatbots, particularly in terms of fast responsiveness and handling short user queries. However, they di er in that question-answering bots are required to generate answers from a prede ned knowledge base or dataset. While the user query itself may be brief, the actual inference request sent to the LLM is much longer, as it includes relevant data in the form of embeddings pulled from the knowledge base. This work ow is commonly referred to as Retrieval Augmented Generation or RAG. In this scenario, it's essential to choose an AI system and accelerator that performs optimally with long input sequence lengths and short to moderate output sequence lengths to achieve the best TCO and user experience. Additionally, since the knowledge base data is typically stored outside the GPU memory, in a much larger CPU memory, deploying systems with fast interconnects between the CPU and GPU — surpassing traditional PCIe interconnect speeds — can further optimize the user experience and reduce TCO.

AI Agents

AI chatbots currently leverage generative AI to provide responses based on single interactions, utilizing natural language processing to reply to user queries. The next evolution in AI is agentic AI, which goes beyond simple responses by employing advanced reasoning and iterative planning to solve complex, multi-step problems. This type of AI ingests large amounts of data from various sources, allowing it to independently analyze challenges, develop strategies, and execute tasks. By continuously learning and improving through a feedback loop, agentic AI systems enhance decision-making and operational e ciency.

Agentic AI necessitates function calling and the coordination of multiple models that result in the generation of additional tokens during inference. For leaders leveraging AI agents within their organization, selecting the right inference platform stack — one that o ers the necessary abstraction tools and blueprints to e ectively orchestrate these interactions — is vital to ensure performant inference operations.

As demonstrated, di erent use cases have distinct requirements for Input Sequence Lengths, Batch Sizes, Time to First Token, Time Per Output token and CPU to GPU interconnect. Focusing on benchmarking that closely mirrors your target production use case when selecting the right instance is crucial for maintaining consistent user service level agreements and preventing infrastructure costs from gradually increasing due to under-provisioned hardware.

Deployment Factors Impacting Inference Performance

In the previous section, we examined the di erent performance metrics you can measure and optimize within an inference solution, and how these metrics are in uenced by the speci c use case. In this section, we will focus on how deployment factors and architectural considerations a ect performance, such as:

1. Hardware Architecture Considerations
2. Model Size
3. Maximizing Utilization with Model Concurrency
4. Multiple Modalities in Models
5. Optimizing Performance with Advanced Batching Techniques

The aim is not to provide detailed guidance on deploying or architecting your particular solution, but to equip you with the knowledge needed to design it for optimal performance.

Hardware Architecture Considerations

Selecting the right GPU is a crucial factor that signi cantly impacts performance. However relying solely on peak chip or instance metrics like rated FLOPS or memory speci cations may not tell the full story, especially as delivered workload performance depends greatly on many factors, including the e ciency of the software stack. Similarly, the absolute price of a GPU isn't a meaningful metric. What truly matters is the performance delivered per unit of power or dollar spent. This means that one GPU with a higher per-hour cost than another may provide a lower overall cost per token than one with a lower per-hour cost.

Data centers are increasingly power limited, making it critical to maximize data center inference throughput within a given power budget. Delivered inference throughput for a given amount of energy use – in other words, energy e ciency – is critical to maximizing data center inference throughput and ultimately revenue potential.

Inference Optimized GPU Architecture

Since their debut in data centers in 2012, NVIDIA GPUs have transformed the industry by enabling parallel processing and signi cantly cutting down the time required for resource-intensive tasks. This shift has led to dramatic improvements, o ering up to 30x more performance per watt and 60x more performance per dollar compared to traditional CPU-based systems.

NVIDIA continues to push the boundaries of innovation, delivering breakthrough performance with each generation. The latest NVIDIA Blackwell architecture o ers 30x faster inference speeds for trillion-parameter models compared to the previous generation. This is made possible by groundbreaking advancements, such as the integration of a second-generation transformer engine and faster, wider NVLINK interconnects, resulting in orders of magnitude greater performance than its predecessor.

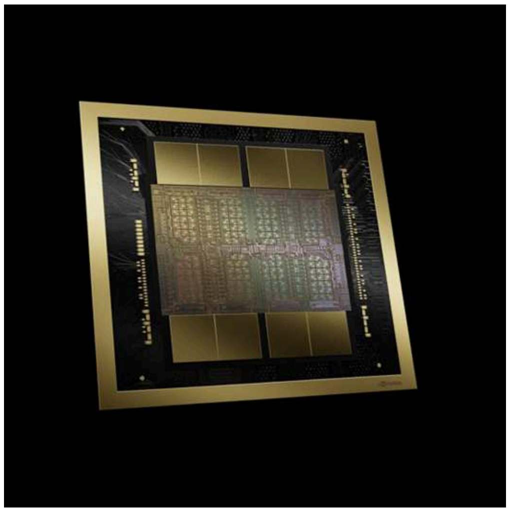


Figure 3. NVIDIA Blackwell GPU Built with 208 Billion Transistors to Deliver Unparalleled Inference Performance

Blackwell, built with 208 billion transistors, over 2.5x the transistors of its predecessor, is the largest GPU ever made. It introduces the second-generation Transformer Engine, combining custom Blackwell Tensor Cores with TensorRT-LLM to accelerate inference for LLMs and Mixture of Experts (MoE) models. Blackwell Tensor Cores o er new precisions and microscaling formats for improved accuracy and throughput. The Transformer Engine enhances performance with micro-tensor scaling and enables FP4 AI, doubling performance, HBM bandwidth, and model size per GPU, compared to FP8.

Pairing GPUs with Power-Efficient CPUs

The NVIDIA GB200 Grace Blackwell Superchip connects two high-performance NVIDIA Blackwell Tensor Core GPUs and an NVIDIA Grace CPU using the NVIDIA® NVLink®-C2C interconnect that delivers 900 gigabytes per second (GB/s) of bidirectional bandwidth to the two GPUs.

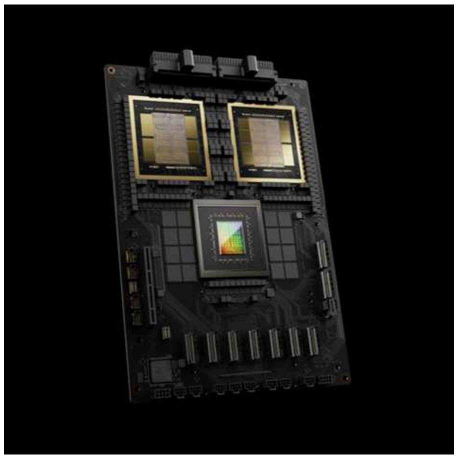


Figure 4. NVIDIA GB200 Superchip Includes Two Blackwell GPUs and One Grace CPU

The NVIDIA Grace CPU combines 72 high-performance and energy-e cient Arm Neoverse V2 cores, connected with the NVIDIA Scalable Coherency Fabric (SCF). The NVIDIA SCF is a high-bandwidth, on-chip fabric that provides a total of 3.2 TB/s of bisection bandwidth — double that of traditional CPUs.

Grace is the rst data center CPU to use high-speed LPDDR5X memory with server-class reliability through mechanisms like error-correcting code (ECC). Grace delivers up to 500 GB/s of memory bandwidth while consuming just one- fth the energy of traditional DDR memory at similar cost.

These numerous innovations mean that NVIDIA Grace delivers outstanding performance, memory bandwidth, and data-movement capabilities with breakthrough performance per watt.

Multiple GPUs Working Together as One During Inference

While selecting the right GPU is critical for AI deployments, exascale computing and trillion-parameter AI models require seamless GPU-to-GPU communication allowing multiple GPUs to work in tandem as one single massive GPU during Inference.

The NVIDIA GB200 NVL72 connects 36 GB200 Superchips (36 Grace CPUs and 72 Blackwell GPUs) in a rack-scale design. The GB200 NVL72 is a liquid cooled, rack-scale 72-GPU NVLink domain, that can act as a single massive GPU.

NVIDIA GB200 NVL72 leverages NVIDIA's fth-generation NVLink which doubles performance over the previous generation to 100 GB/sec per link, enabling faster data transfer and optimized scaling for large AI models. It also features the NVIDIA NVLink

Switch, which enables 130TB/s GPU bandwidth in a 72-GPU NVLink domain (NVL72) for model parallelism. When combined, NVLink and the NVLink Switch support multi-server clusters with 1.8 TB/s interconnect, scaling GPU communications and computing.

Figure 5. NVIDIA GB200 NVL72

GB200 NVL72 delivers a 30X inference speedup compared to prior generation with 25X lower TCO and 25X less energy with the same number of GPUs for massive models such as a GPT-MoE- 1.8T.

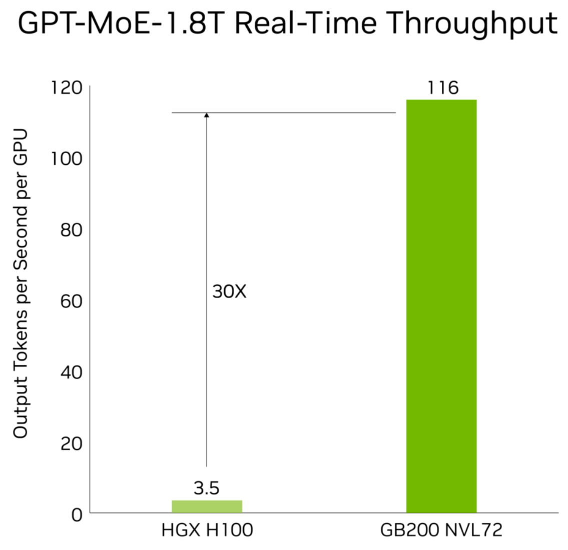


Figure 6. The GB200 Delivers 30x Real-Time Throughput Compared to the H100

Model Size

When selecting an LLM for workload deployment, an important consideration is the number of parameters in the model, or the model size. Over time, the number of parameters of frontier LLMs has continued to increase, yielding improvements in model capabilities and response quality.

To serve a broad set of use cases, deployment scenarios, and budgets, model developers will often provide variants of their models in a range of sizes. For example, the Llama 3.1 family of open models is available in sizes of 8B, 70B, and 405B parameters. Generally, within a model family, larger models will provide more accurate responses. It is important to keep in mind, however, that larger models also require more computational resources to generate tokens than smaller ones do.

This means that model size selection depends on both the intended use case as well as available compute resources. For use cases that demand the highest response accuracy for complex tasks, larger models may be the preferred choice. However, for scenarios where output token generation speed is critical or available compute resources are limited, using a smaller model may be preferred.

Maximizing Utilization with Model Concurrency

As data center GPUs continue to evolve, the compute and memory resources they o er are growing rapidly with each new generation. These next-generation GPUs are highly e ective at accelerating massive-scale inferencing, such as serving trillion-parameter models. However, this power comes with a caveat: when serving smaller models, the large compute and memory resources often remain underutilized, leading to higher Total Cost of Ownership (TCO).

In these scenarios, the GPU's resources are not being e   ciently used, as large portions of the compute capacity and memory are left idle. This results in unnecessary infrastructure costs, creating an imbalance between deployed resources and actual workload requirements.

Model concurrency o ers an elegant solution to this challenge. By enabling multiple models, or multiple instances of the same model, to run simultaneously on a single GPU or across GPUs in a node, system throughput increases and latency decreases—all without requiring incremental investments in infrastructure. This approach allows organizations to maximize the use of their existing GPU resources, ensuring that compute is fully utilized and that infrastructure costs are optimized.

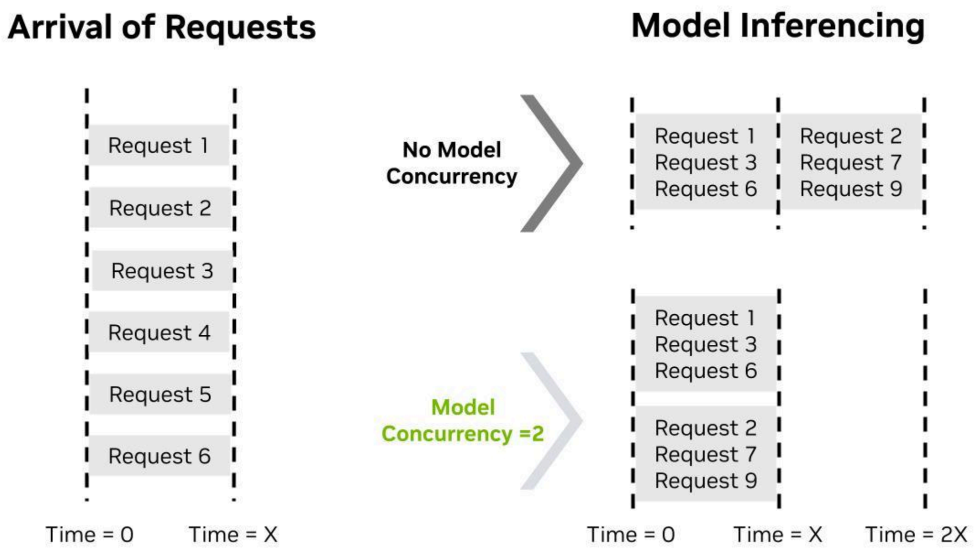


Figure 7. How Increasing Model Concurrency Impacts Throughput and Latency

NVIDIA Triton Inference Server streamlines the process of deploying models with concurrency enabled. Developers can easily implement model concurrency by switching on a single ag in the model's con guration le. From there, they can specify how many concurrent model instances to deploy and which target GPUs to use. This simplicity empowers developers to leverage the full power of their GPU infrastructure without complex con guration or manual scaling processes.

In scenarios where multiple independent models need to be served but cannot all t concurrently on the same instance or node, NVIDIA Triton can function as a multiplexer. It dynamically loads and unloads models based on incoming user requests, ensuring that the required models are available at the right time. This allows organizations to maintain high service availability and performance without the need for additional infrastructure investments.

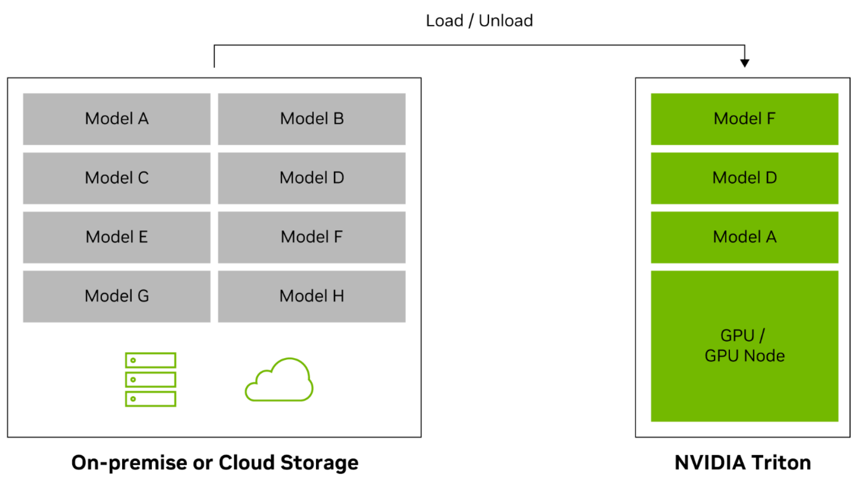


Figure 8. How NVIDIA Triton Can Load and Unload Models on a GPU, Increasing Service Availability and Performance

In addition to improving throughput and reducing cost, model concurrency also addresses situations where user demand for a service is not predictable in advance. IT teams can begin with a single instance of a model and scale up gradually by deploying multiple concurrent instances of the model as demand increases. The ability to adjust the number of concurrent models makes it possible to meet changing demands while minimizing the cost of unused compute resources.

Model concurrency is a powerful inference performance tool, particularly when deploying small to medium-sized models on Data Center GPUs. By utilizing this approach, organizations can experience signi cant bene ts:

- **Increased Throughput:** Running multiple models or model instances in parallel enhances overall system throughput, allowing it to handle more inference requests simultaneously.
- **Lower Cost per Token:** By making full use of existing resources, model concurrency reduces the cost per token served, ensuring that infrastructure investments are maximized.
- **Reduced Latency:** Concurrently running models or model instances reduces wait times, enhancing the responsiveness of the service and improving the user experience.

Multiple Modalities in Models

AI models capable of processing and integrating information from multiple data modalities simultaneously such as text, images, audio and video are called multimodal models. Unlike the traditional unimodal AI models that work with a single input modality, multimodal models have complex inference serving requirements that must enable processing inputs and generating outputs across various modalities.

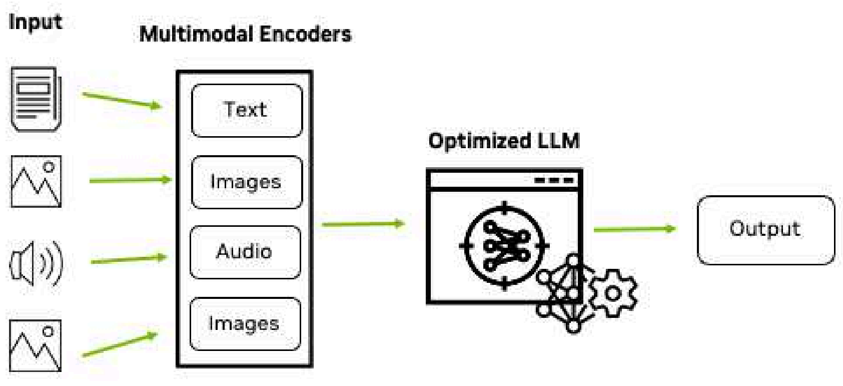


Figure 9. Multimodal LLM Fusing Information from Multiple Modalities and Optimized for High Throughput Output

NVIDIA AI inference platform provides optimized support for multimodal models with highly performant input encoders for audio and image using the NVIDIA TensorRT library and text decoder optimized using the TensorRT-LLM library. These models can be deployed using NVIDIA's Triton Inference Server that supports several advanced features such as multi-image inference where a single request can contain multiple images, and e ective KV cache reuse where images could be shared across multiple text input tokens to help reduce latency and improve performance proportional to the length of the KV cache reuse.

Optimizing Performance with Advanced Batching Techniques

Data Center AI accelerators, such as NVIDIA GPUs, utilize multiple cores to process requests in parallel. To maximize the potential of these cores, requests are often grouped together into batches and sent for inference. However, a critical tradeo exists between batch size, throughput, and latency. Adjusting any two of these factors can negatively impact the third, creating challenges for businesses seeking optimal performance.

For instance, a smaller batch size can reduce latency but leads to underutilized GPU resources, resulting in ine ciencies and higher operational costs. On the other hand, larger batches maximize the use of GPU resources but come at the cost of increased latency, which can harm the user experience.

IT teams often perform A/B testing to determine the ideal balance for their use case, but this can be a complex and time-consuming process. Leveraging an advanced inference software platform, such as NVIDIA Triton Inference Server and NVIDIA TensorRT-LLM, can mitigate this tradeo . NVIDIA AI inference platform deploys sophisticated batching techniques to help organizations strike a better balance between throughput and latency, improving overall system performance.

Dynamic Batching

Dynamic batch creates batches based on speci c criteria such as maximum or preferred batch size and maximum waiting time. If the batch can be formed at the preferred size, it will be processed at that size; otherwise, the system will form a batch of the largest possible size that meets max batch size con gured. By allowing requests to remain in the queue for a short time, dynamic batching can wait till additional requests arrive to create a larger batch size enhancing throughput and resource utilization, ultimately lowering cost per token.

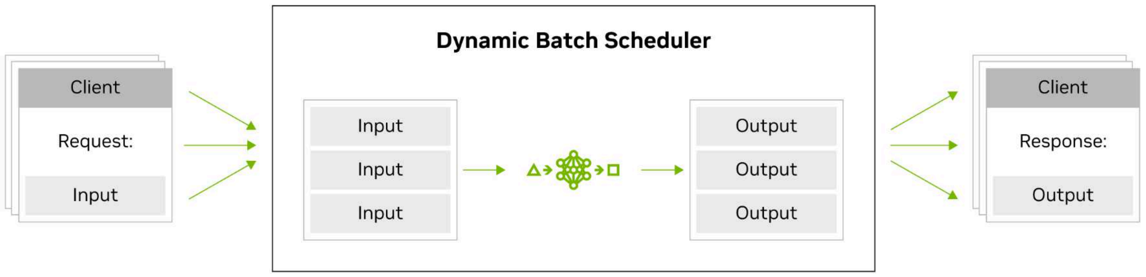


Figure 10. How Dynamic Batching Creates Batches of Multiple Requests

Sequence Batching

For use cases that require speci c request sequencing, such as video streaming, sequence batching ensures that related requests (e.g., frames in a video) are processed in a meaningful order. When requests are correlated, such as those in a video stream where the model needs to maintain state between frames, sequence batching ensures that requests are processed sequentially on the same instance. This guarantees the correct output while optimizing performance.

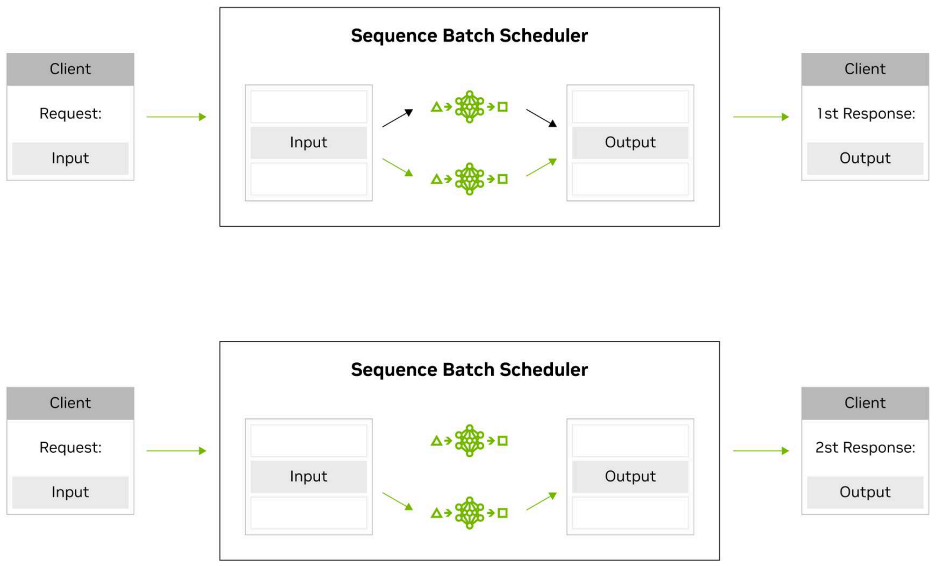


Figure 11. How Sequence Batching Creates Batches of Di erent Requests While Maintaining Request Relationships and Sequences

Inflight Batching

In ight batching enhances traditional batch processing by enabling continuous request handling. With in ight batching, the TensorRT-LLM runtime (an NVIDIA SDK that optimizes LLMs for inference) processes requests as soon as they are ready, without waiting for the entire batch to complete. This allows for faster processing and higher throughput by immediately initiating new requests while others are still being processed.

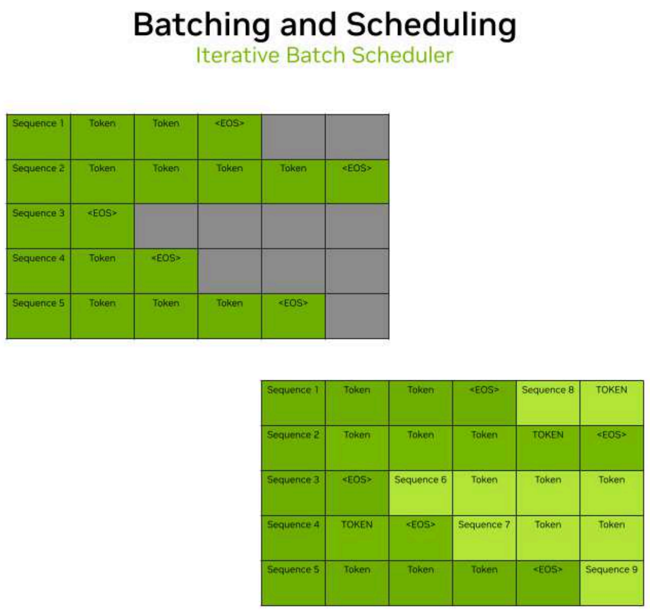

Batching and Scheduling
Iterative Batch Scheduler
Sequence 1 Token Token <EOS>
Sequence 2 Token Token Token Token <EOS>
Sequence 3 <EOS>
Sequence 4 Token <EOS>
Sequence 5 Token Token Token <EOS>
Sequence 1 Token Token <EOS> Sequence 8 TOKEN
Sequence 2 Token Token Token TOKEN <EOS>
Sequence 3 <EOS> Sequence 6 Token Token Token
Sequence 4 TOKEN <EOS> Sequence 7 Token Token
Sequence 5 Token Token Token <EOS> Sequence 9

Figure 12. How In ight Batching Can Evict Requests within a Batch Once Completed and Replace them with New Ones

E cient use of compute resources requires careful management of batch size, throughput, and latency. Advanced batching techniques o ered by the NVIDIA AI inference platform, such as dynamic batching, sequence batching, and in ight batching, provide organizations with the exibility to optimize performance, reduce costs, and improve the user experience. These technologies help strike a balance between throughput and latency, enabling businesses to scale AI-driven solutions more e ectively.

Parallelizing Inference for Large Models: Key Methods and Tradeoffs

As the size and complexity of large language models (LLMs) increase, ensuring that these models can be deployed e ciently across multiple GPUs becomes crucial. When models exceed the memory capacity of a single GPU, parallelization techniques are employed to split the workload and optimize performance. For executives and IT leaders, understanding the primary parallelism methods and their impact on throughput and user interactivity is key to making informed infrastructure decisions. Below are the primary methods for parallelizing inference in large models and examples from leading case studies.

Data Parallelism (DP)

Data parallelism involves duplicating the model across multiple GPUs or GPU clusters. Each GPU independently processes groups of user requests, ensuring that no communication is required between these request groups. This method scales linearly with the number of GPUs used, meaning the number of requests served increases directly with the GPU resources allocated. However, it is important to note that data parallelism alone is usually insu cient for the latest large-scale LLMs, as their model weights often cannot t onto a single GPU. Consequently, DP is commonly used alongside other parallelism techniques.

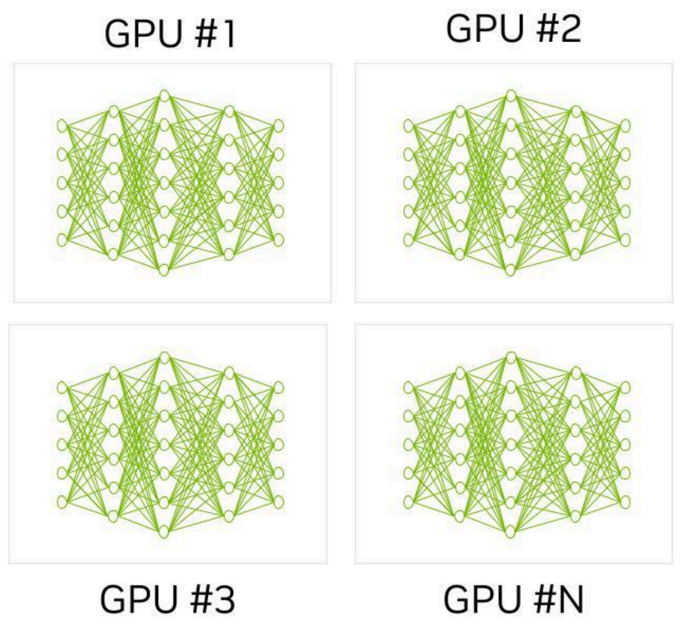


Figure 13. Applying Data Parallelism Across Multiple GPUs

Impact on Performance:

- **GPU Throughput:** Una ected by DP alone as the model is duplicated.
- **User Interactivity:** No signi cant impact as requests are processed independently.

Tensor Parallelism (TP)

In tensor parallelism, the model parameters are split across multiple GPUs, with user requests shared across GPUs or GPU clusters. The results of the computations performed on di erent GPUs are combined over the GPU-to-GPU network. This method can improve user interactivity, especially for transformer-based models like Llama 405B and GPT4 1.8T MoE (1.8 Trillion Parameters Mixture of Expert Model), by ensuring that each request gets processed with more GPU resources, thus speeding up processing time.

Impact on Performance:

- **GPU Throughput:** Scaling TP to large GPU counts, without a fast interconnect like NVIDIA NVLINK, can reduce throughput due to increased communication overhead.
- **User Interactivity:** Enhanced as user requests are processed faster with more GPU resources allocated to each request.

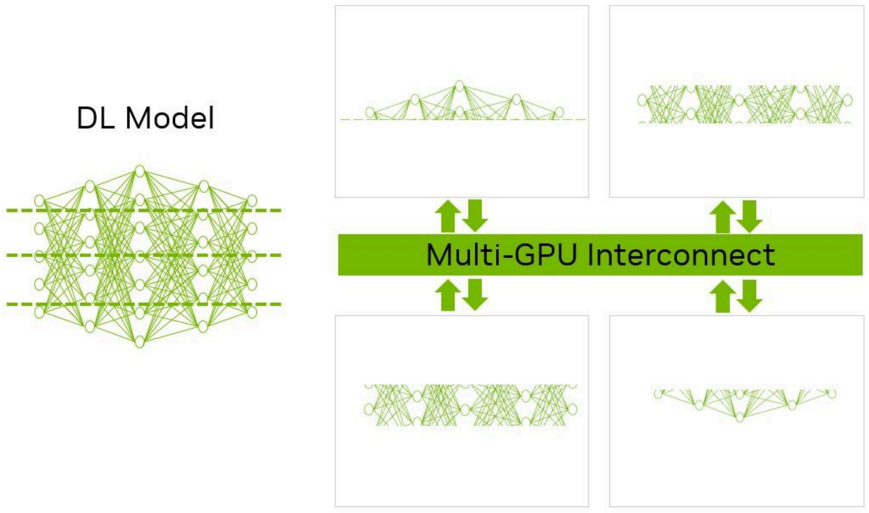


Figure 14. Applying Tensor Parallelism on a Deep Neural Network

Pipeline Parallelism (PP)

Pipeline parallelism divides the model layers, with each group of layers assigned to di erent GPUs. The model processes requests sequentially across GPUs, with each GPU performing computations on its assigned portion of the model. This method is bene cial for distributing large model weights that do not t on a single GPU but has limitations in terms of e ciency.

Impact on Performance:

- **GPU Throughput:** May result in lower e ciency as model weights are distributed
- **User Interactivity:** Does not enable the signi cant optimization of user interactivity as processing must proceed sequentially across GPUs.

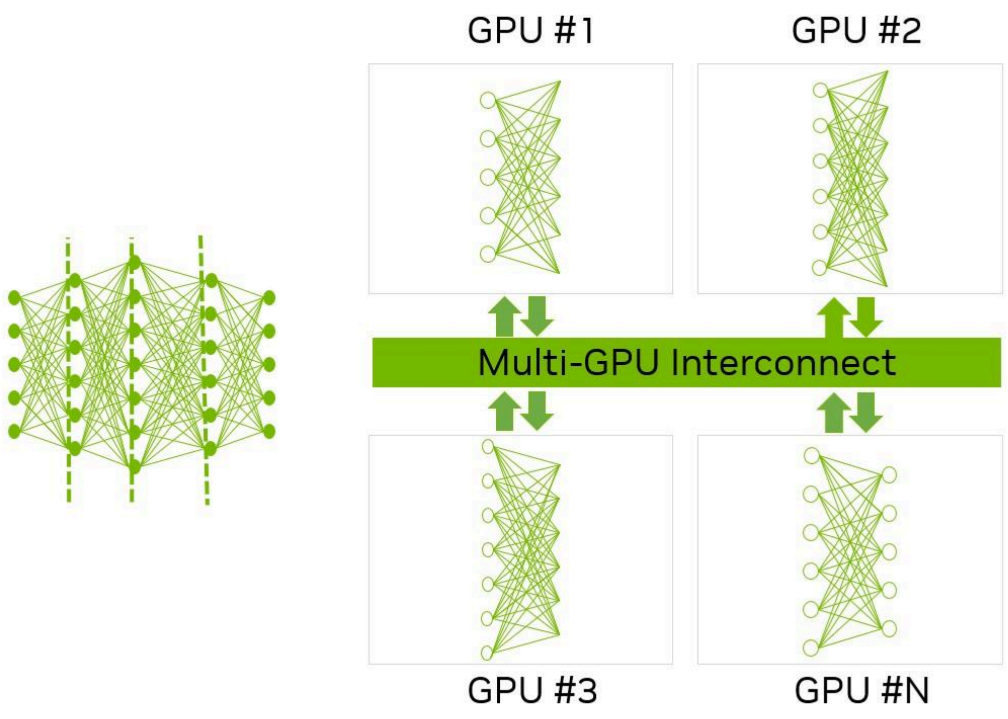


Figure 15. Using Pipeline Parallelism on a Deep Neural Network

Expert Parallelism (EP)

Expert parallelism involves routing user requests to specialized "experts" within the model. By limiting each request to a smaller set of model parameters (i.e., speci c experts), the system reduces computational overhead and optimizes processing. Requests are processed by individual experts before being recombined at the nal output stage, which requires high-bandwidth GPU-to-GPU communication.

Impact on Performance:

- **GPU Throughput:** Expert parallelism can o er signi cant improvements in **throughput, especially when combined with other techniques.**
- **User Interactivity:** EP con gurations can maintain or improve user interactivity without the steep tradeo s seen in other methods.

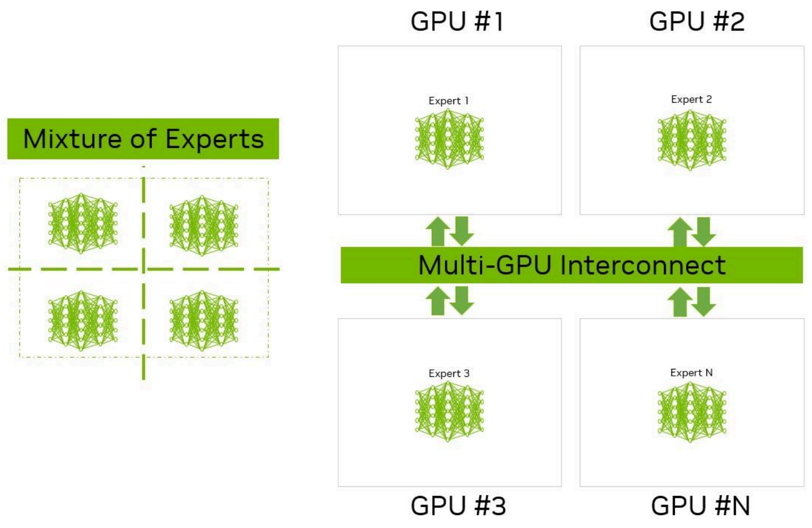


Figure 16. Using Expert Parallelism on a Deep Neural Network that Consists of Four Experts

Combining Parallelism Methods for Optimal Performance

The most e ective way to optimize the performance of LLMs across multiple GPUs is by combining multiple parallelism techniques. By doing so, the tradeo between throughput and user interactivity can be minimized, ensuring both high performance and responsive user experiences.

For instance, when serving the GPT 1.8T MoE model with 16 experts, using 64 GPUs, each with 192 GB of memory, we observe that Expert + Pipeline Parallelism (EP16PP4) o ers a 2x improvement in user interactivity with negligible reduction reduction in GPU throughput compared to expert-only parallelism and Tensor + Expert + Pipeline Parallelism (TP4EP4PP4) delivers 3x more GPU throughput compared to tensor-only parallelism, while maintaining user interactivity.

For CIOs and IT leaders overseeing the deployment of LLMs, it is essential to understand the tradeo s and bene ts of various parallelization strategies. Combining parallelism methods like data, tensor, pipeline, and expert parallelism allows organizations to optimize

GPU resource utilization and improve both throughput and user interactivity. In practice, well-planned con gurations can deliver optimal results, balancing high performance with responsive user experiences across multiple GPUs. As demand for larger and more complex models increases, these parallelization techniques will play a critical role in ensuring scalable and e cient AI model deployment.

Scaling AI Inference to Meet Fluctuating Demand

As IT leaders roll out AI applications, one of the most challenging decisions they face is forecasting user demand and understanding how that demand will uctuate over time. These forecasts signi cantly in uence infrastructure decisions, particularly in relation to provisioned resources like GPUs, which directly impact both cost and performance. Balancing these elements is key to ensuring that AI systems can scale dynamically while controlling overhead.

AI applications, particularly those leveraging optimized LLMs, require substantial computational power for real-time inference. A common approach to scaling these applications involves provisioning GPUs based on projected peak demand. However, this can be ine cient, especially when the peak demand is only temporary or irregular.

To address this, organizations need the exibility to scale infrastructure up or down, responding to uctuating demands. Kubernetes provides an ideal solution, enabling IT teams to dynamically scale the deployment of LLMs. Whether scaling from a single GPU to multiple GPUs or reducing resources during o -peak hours, Kubernetes o ers a cost-e ective and performance-optimized approach to infrastructure management.

Flexibility with Kubernetes and the NVIDIA AI Inference Platform

Enterprises, particularly those in industries like e-commerce or customer service, are constantly faced with uctuating volumes of inference requests. Whether it's handling high volumes of customer inquiries during a sale or managing fewer requests during late-night hours, the ability to scale e ciently is paramount. Scaling with Kubernetes allows organizations to dynamically adjust the number of GPUs needed, ensuring that hardware resources align with real-time demand. This approach o ers substantial savings compared to over-provisioning hardware resources to handle peak workloads.

The NVIDIA AI inference platform, including NVIDIA Triton as well as NVIDIA NIM, supports Kubernetes to facilitate the scaling process. As inference requests increase, Triton metrics are scraped by Prometheus and sent to Kubernetes Horizontal Pod Autoscaler (HPA), which adds more pods to the deployment, each with one or more GPUS. One example of a custom metric that can be scrapped from Triton using Prometheus and sent to the Kubernetes HPA to inform scaling decisions is the *queue-to-compute ratio*. This ratio re ects the response time of inference requests. It's de ned as the queue time divided by the compute time for an inference request.

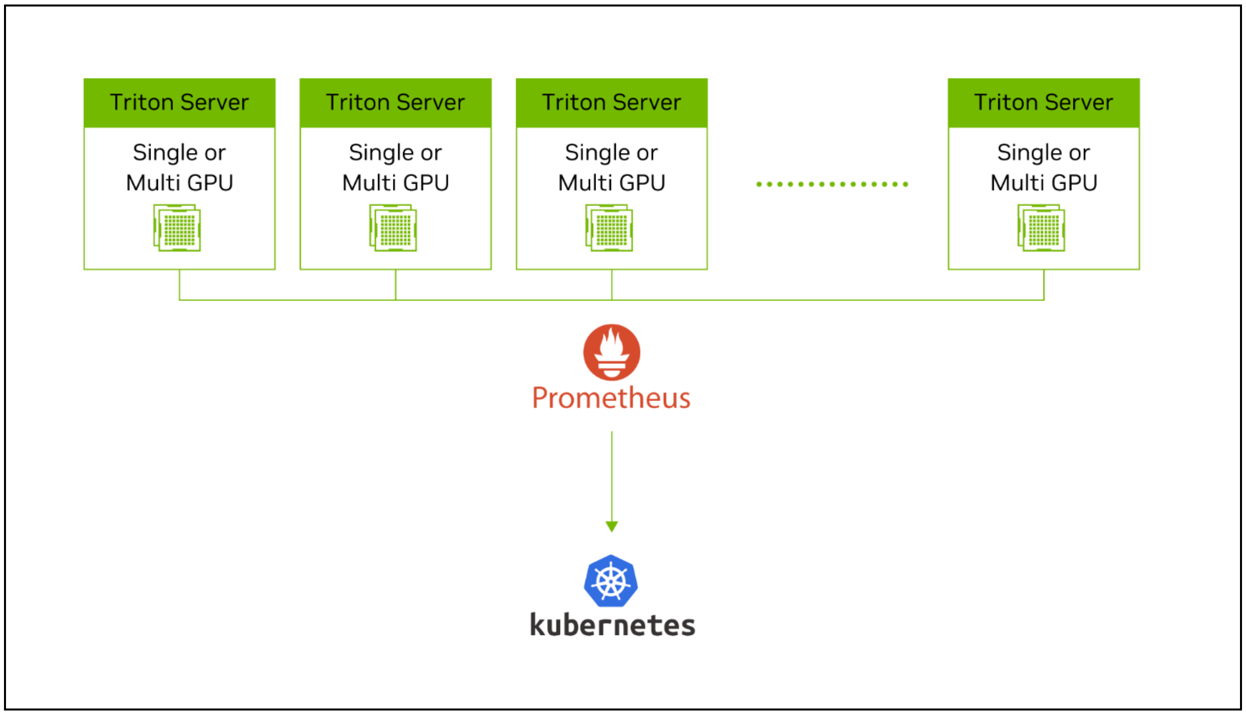


Figure 18. NVIDIA Triton Server Scaling with Kubernetes Architecture

Key Benefits of Scalable AI Deployments:

1. **Cost Efficiency:** Scaling based on demand allows enterprises to optimize their infrastructure costs, ensuring that they're only provisioning the necessary hardware to handle real-time requests. This dynamic scalability reduces the need for over-provisioning, which can be expensive.
2. **Performance Optimization:** By balancing GPU resources dynamically, Kubernetes and NVIDIA Triton ensure that performance remains high, even during periods of heavy load. This allows AI applications to meet strict latency and accuracy requirements, which are essential in customer-facing environments.
3. **Flexibility Across Industries:** From handling the surge in traffic during online shopping events to managing fluctuating request volumes in call centers, scalable AI deployment offers flexibility that can accommodate the needs of a wide range of industries.

For further details on optimizing AI deployments with NVIDIA Triton and Kubernetes, visit Scaling LLMs with NVIDIA Triton and NVIDIA TensorRT-LLM Using Kubernetes. For further details on optimizing AI deployments with NVIDIA NIM and Kubernetes, visit Autoscaling of NVIDIA NIM on Kubernetes and Managing AI Inference Pipelines on Kubernetes with NVIDIA NIM Operator.

Streamlining AI Pipelines with Model Ensembles

AI and ML systems are rarely deployed as standalone models. Instead, they are often part of a broader, more complex pipeline that integrates various models, pre-processing steps, and post-processing tasks. This approach, known as Model Ensembles, has become a cornerstone of modern AI work ows, especially in enterprise applications.

A Model Ensemble refers to the integration of multiple machine learning models, pre-processing, and post-processing steps into a uni ed pipeline. Instead of executing models individually, ensembles allow these components to work in tandem, streamlining the process and enhancing overall e ciency. Each model in the ensemble typically handles a di erent aspect of the task, whether it's processing data, generating predictions, or re ning outputs.

In practical scenarios, such as in enterprise applications, an AI pipeline may include multiple computer vision models, image editors, and other specialized models, all working together to produce an end result. For instance, when creating a marketing campaign using SDXL (a generative AI model), the work ow might require an initial zoom-in preprocessing step, followed by the generation of a scene using SDXL, and nishing with a post-processing step, such as upscaling the image for high-resolution display.

By stitching together these individual steps into a cohesive pipeline, businesses can ensure the smooth ow of data through the models while reducing latency and optimizing resource usage.

The Role of NVIDIA Triton Inference Server in Model Ensembles

Building and managing these complex model pipelines can be challenging. However, NVIDIA AI inference platform tools like the Triton Inference Server o er powerful solutions for automating the process providing advanced capabilities for orchestrating model ensembles.

Triton's Model Ensembles feature eliminates the need for writing manual code to manage each step, reducing complexity and minimizing the risk of errors. Additionally, Triton supports running pre- and post-processing on CPUs, while the core AI model can run on GPUs, providing exibility in balancing processing power. Additionally, Triton supports adding advanced features to the ensemble, such as conditional logic and loops, allowing developers to build more sophisticated and exible AI work ows.

Benefits of Model Ensembles with Triton

The use of Model Ensembles comes with several key benefits, especially when integrated with Triton:

1. **Reduced Latency:** By combining pre-processing, inference, and post-processing into a single pipeline, enterprises can significantly reduce the time taken to move data between models. This reduction in latency is crucial for real-time AI applications, such as generative AI and computer vision.
2. **Improved Resource Utilization:** Model Ensembles allow teams to optimize the deployment of resources. For instance, by running lighter pre- and post-processing tasks on CPUs and reserving GPU resources for more intensive model tasks, businesses can ensure cost-effective scaling without sacrificing performance.
3. **Lower Network Overhead:** Traditionally, each model in a machine learning pipeline would need to communicate with others over the network, transferring intermediate data between steps. With Model Ensembles, however, these intermediate data transfers are minimized, reducing the number of network calls and optimizing the use of bandwidth.
4. **No-Code Integration:** Triton's Model Ensembles feature offers a low-code or no-code approach to connecting different AI models. This makes it easier for IT leaders and inference teams to integrate pre- and post-processing workflows (often built with Python) into a seamless pipeline. This streamlined integration allows for a single inference request to trigger the entire process, further improving efficiency.

Practical Example: AI-Driven Generative Applications

To better understand how Model Ensembles work, let's consider an example involving generative AI. Imagine a text-to-image application where an input text is converted into a synthesized image. The pipeline for such an application typically consists of two main components: a LLM for encoding the input text, and a diffusion model for generating the image.

Before the input text is fed to the LLM, some pre-processing is needed. This could involve cleaning the text, tokenizing it, or formatting it in a way that's compatible with the LLM. Similarly, the output image might require post-processing, such as resizing or adding effects, before it can be used in the final application.

Using Model Ensembles in this case would allow the text-to-image process to be fully automated and optimized. Triton could seamlessly connect each step, from text preprocessing to image generation, and even handle the post-processing, all within one unified pipeline.

The Power of Model Ensembles for AI Optimization

By integrating multiple models and pre- and post-processing steps into a single, optimized pipeline, IT leaders can enhance the e ciency of their AI applications, reduce latency, and minimize resource usage.

The NVIDIA AI inference platform provides a powerful platform for managing these ensembles, o ering exibility in resource allocation and simplifying the integration of di erent components. With its low-code approach and automated optimization, it enables teams to build and scale AI systems with greater ease, making it an essential tool for enterprises looking to increase the performance of their AI models.

Advanced Inference Techniques to Boost Performance

In the previous chapter, we covered foundational strategies for boosting inference performance. In this chapter, we dive into more advanced techniques tailored for IT leaders, particularly those for whom AI inference is integral to their organization's revenue generation strategy, or those who have already optimized their systems and are now seeking to unlock the next level of performance from their AI investments. Our goal is to provide an overview of cutting-edge optimization techniques featured in the NVIDIA AI inference platform, while also guiding leaders on how to direct their teams for further learning. We will explore advanced strategies from Triton and TensorRT-LLM, including Chunked Pre ll, Multiblock Attention, KV Cache Early Reuse, Disaggregated Serving, and Speculative Decoding — each designed to push the boundaries of AI inference in production environments.

Chunked Prefill: Maximizing GPU Utilization and Reducing Latency

LLM inference typically involves two key phases: pre ll and decode. In the pre ll phase, the system computes the contextual understanding of the user's input (KV cache), which is computationally intensive. The decode phase follows, generating tokens sequentially, with the rst token derived from the KV cache. However, traditional methods often struggle with balancing the heavy computational demand of the pre ll phase and the lighter load of the decode phase.

Chunked Pre ll is an optimization that divides the pre ll phase into smaller chunks, improving parallelization with the decode phase and reducing bottlenecks. This approach helps in handling longer contexts and higher concurrency levels while maximizing GPU memory and compute resources. It also o ers exibility by decoupling memory usage from input sequence length, making it easier to process large requests without straining memory capacity. With dynamic chunk sizing, TensorRT-LLM intelligently adjusts chunk sizes based on GPU utilization, simplifying deployment and eliminating the need for manual con guration.

Multiblock Attention: Boosting Throughput with Long Contexts

As AI models evolve, the size of context windows — allowing for better cognitive understanding — has grown exponentially. Llama 2 started with 4K tokens, and the recent Llama 3.1 expanded this to an impressive 128K tokens. Handling these long sequences in real-time inference scenarios presents unique challenges, particularly with GPU resource allocation.

The Multiblock Attention technique within TensorRT-LLM addresses these challenges by e ciently distributing the decoding process across all Streaming Multiprocessors (SMs) on a GPU. By doing so, it signi cantly boosts throughput and reduces bottlenecks associated with large KV cache sizes. This method ensures that even with small batch sizes, common in real-time applications, GPUs are fully utilized, leading to up to 3.5x more tokens per second on NVIDIA's HGX H200 platform. This performance boost doesn't come at the cost of time-to- rst-token (TTFT), ensuring rapid responses even for complex queries.

KV Cache Early Reuse: Cutting Down Time-to-First-Token

The generation of KV cache is a computationally intensive process. As such, KV cache reuse plays a critical role in speeding up token generation by avoiding redundant computations. However, traditional methods often require waiting for the entire KV cache to be generated before reuse can occur, leading to ine ciencies in high-demand environments.

KV Cache Early Reuse optimizes this process by allowing portions of the cache to be reused as they are being generated, rather than waiting for full completion. This technique signi cantly accelerates inference, especially in scenarios where system prompts or prede ned instructions are required. For instance, in enterprise chatbots, where user requests often share the same system prompt, this feature can accelerate TTFT by up to 5x during periods of high demand, providing a faster and more responsive user experience.

Disaggregated Serving: Independent Resourcing and Decoupled Scaling

Traditional inference setups often co-locate the pre ll and decode phases on the same GPU, leading to ine cient resource allocation and suboptimal throughput. NVIDIA Triton Disaggregated Serving (DistServe) strategy decouples these phases, allowing AI inference teams to allocate resources independently based on the speci c needs of each phase. This enables independent resourcing and decoupled scaling, meaning that more GPUs can be allocated to the pre ll phase to optimize TTFT, while additional GPUs can be dedicated to the decode phase to improve Time Between Tokens (TBT).

This separation also introduces phase-speci c parallelism — where compute-heavy tasks in the pre ll phase can bene t from techniques like Pipeline Parallelism, while the memory-intensive decode phase can leverage Tensor Parallelism. Triton DistServe reduces inference costs by up to 50% while maintaining Service Level Objectives (SLOs) for throughput, TTFT, and TBT.

Speculative Decoding: Accelerating Token Generation

The process of generating tokens in an autoregressive manner (one at a time) can be slow and ine cient. Speculative Decoding optimizes this by generating multiple potential token sequences in parallel, reducing the time required for token generation. TensorRT-LLM integrates various speculative decoding methods, such as Draft Target and Eagle Decoding, which allow the system to predict multiple tokens and select the most appropriate one.

For models like Llama 3.3 70B, speculative decoding leads to a signi cant 3.5x increase in tokens per second, improving throughput and user experience without compromising output quality. This technique is particularly bene cial in low-latency, high-throughput environments, where maximizing the e ciency of each computational step is essential.

Unlocking the Future of AI Inference

The NVIDIA AI inference platform is built to support organizations at any stage of their AI inference journey. For those still in the experimentation phase or focused on faster time to market, the strategies outlined in the chapter, Deployment Factors Impacting Inference Deployment, provide an excellent starting point. However, for organizations where AI inference represents a major cost driver impacting gross margins, the advanced performance and cost-saving techniques covered in the chapter, Unlocking AI Inference Performance and Cost E ciency in the Cloud, will help maximize system e ciency, performance, and minimize costs.

Getting Started with the NVIDIA AI Inference Platform

The NVIDIA AI inference platform o ers exible deployment options for inference to meet a broad range of business and IT requirements.

Enterprises seeking the fastest time to value can leverage NVIDIA NIM, which o ers prepackaged, optimized inference microservices for running the latest AI foundation models on NVIDIA accelerated infrastructure anywhere.

For maximum fl xibility, configu ability and extensibility to t your unique AI inference needs, NVIDIA o ers NVIDIA Triton Inference Server and NVIDIA TensorRT which provide the ability to customize and optimize your inference serving platform for your specifi requirements.

Visit NVIDIA AI Inference Solutions to learn more and get started.

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA, the NVIDIA logo, NVIDIA Grace GPU, CUDA, NVLink, NVIDIA GPU Cloud, and NSight are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

VESA DisplayPort

DisplayPort and DisplayPort Compliance Logo, DisplayPort Compliance Logo for Dual-mode Sources, and DisplayPort Compliance Logo for Active Cables are trademarks owned by the Video Electronics Standards Association in the United States and other countries.

HDMI

HDMI, the HDMI logo, and High-Definition Multimedia Interface are trademarks or registered trademarks of HDMI Licensing LLC.

Arm

Arm, AMBA, and Arm Powered are registered trademarks of Arm Limited. Cortex, MPCore, and Mali are trademarks of Arm Limited. All other brands or product names are the property of their respective holders. "Arm" is used to represent Arm Holdings plc; its operating company Arm Limited; and the regional subsidiaries Arm Inc.; Arm KK; Arm Korea Limited.; Arm Taiwan Limited; Arm France SAS; Arm Consulting (Shanghai) Co. Ltd.; Arm Germany GmbH; Arm Embedded Technologies Pvt. Ltd.; Arm Norway, AS, and Arm Sweden AB.

OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

Copyright

© 2025 NVIDIA Corporation. All rights reserved.

To learn more about NVIDIA AI, visit: **www.pny.com/ai**

Contact: **gopny@pny.com**