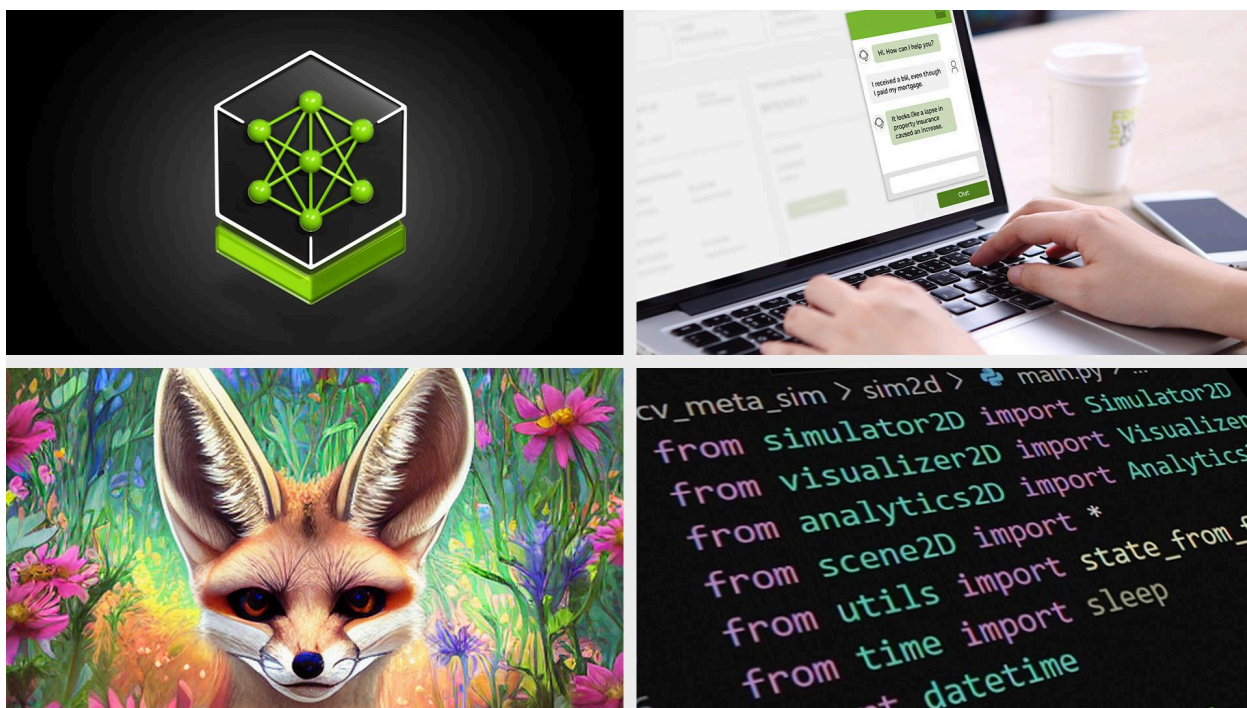




The Art of Balancing AI Inference Cost and Performance

E-book



This e-book is your essential roadmap to navigating the complexities of 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](#)

[Tackling the Cost Per Token Challenge](#)

[Customer Case Study: Wealthsimple](#)

[How Use Cases Impact Inference](#)

[Chatbot](#)

[Summarization](#)

[Question-Answering](#)

[AI Agents](#)

[Customer Case Study: Perplexity](#)

[Deployment Factors Impacting Inference Performance](#)

[Hardware Architecture Considerations](#)

[Inference Optimized GPU Architecture](#)

[Pairing GPUs with Power Efficient CPUs](#)

[Multiple GPUs Working Together as One During Inference](#)

[Model Size](#)

[Maximizing Utilization with Model Concurrency](#)

[Multiple Modalities in Models](#)

[Optimizing Performance with Advanced Batching Techniques](#)

[Dynamic Batching](#)

[Sequence Batching](#)

[Inflight Batching](#)

[Parallelizing Inference for Large Models: Key Methods and Tradeoffs](#)

[Data Parallelism \(DP\)](#)

[Tensor Parallelism \(TP\)](#)

[Pipeline Parallelism \(PP\)](#)

[Expert Parallelism \(EP\)](#)

[Combining Parallelism Methods for Optimal Performance](#)

[Unlocking AI Inference Performance and Cost Efficiency in the Cloud](#)

[Customer Case Study: Amdocs](#)

[Scaling AI Inference to Meet Fluctuating Demand](#)

[Flexibility with Kubernetes and the NVIDIA AI Inference Platform](#)

[Streamlining AI Pipelines with Model Ensembles](#)

[The Role of NVIDIA Triton Inference Server in Model Ensembles](#)

[Benefits of Model Ensembles with Triton](#)

[Practical Example: AI-Driven Generative Applications](#)

[The Power of Model Ensembles for AI Optimization](#)

[Case Study: Let's Enhance Turns Product Photos into Beautiful Marketing Assets](#)
[Advanced Inference Techniques to Boost Performance](#)
[Chunked Prefill: Maximizing GPU Utilization and Reducing Latency](#)
[Multiblock Attention: Boosting Throughput with Long Contexts](#)
[KV Cache Early Reuse: Cutting Down Time-to-First-Token](#)
[Disaggregated Serving: Independent Resourcing and Decoupled Scaling](#)
[Speculative Decoding: Accelerating Token Generation](#)
[Unlocking the Future of AI Inference](#)
[Getting Started with the NVIDIA AI Inference Platform](#)

What is Inference Performance and Why It Matters

[AI 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 workflows, 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 sufficient 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 financial 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 justifies 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 efficiency 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 first is the performance, or throughput, of the AI platform itself. The more requests a system can handle efficiently, 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 significantly, *cost per token* becomes an essential metric for assessing both system performance and overall cost efficiency.

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

What is Inference Performance and Why It Matters

dimensions often presents trade-offs. Increasing one can lead to the deterioration of another, so it's crucial for organizations to strike the right balance.

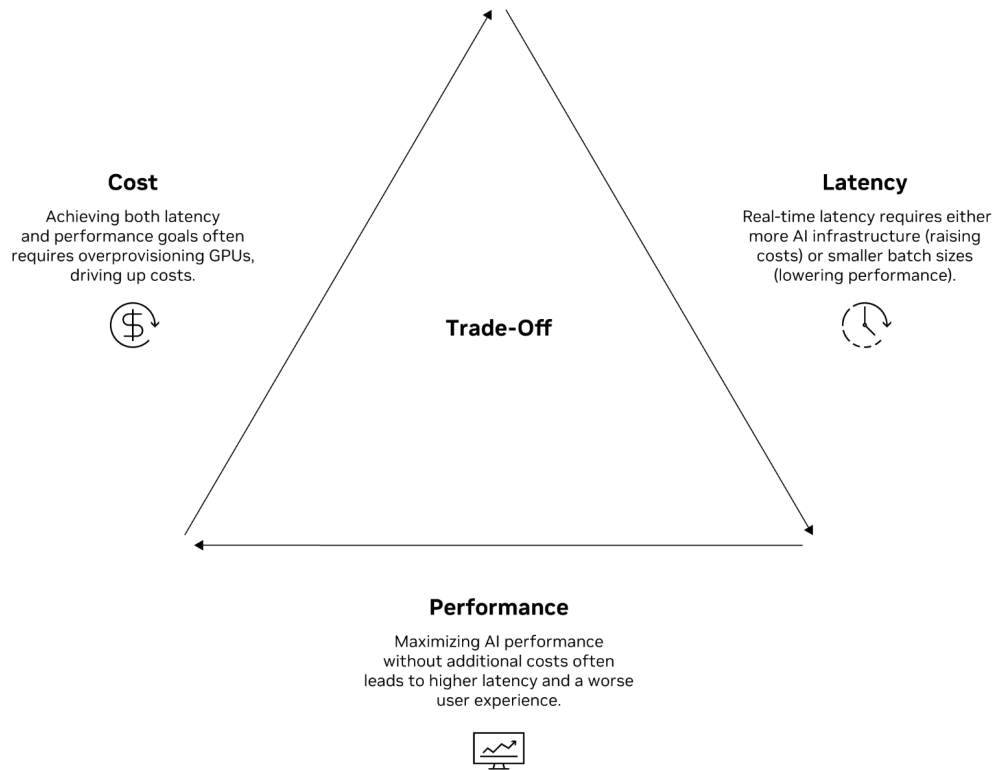


Figure 1. Optimizing Time to First Token and Time Between Tokens Often Presents Three Tradeoffs If Not Correctly Balanced

To account for these interdependencies, IT leaders are starting to measure *Goodput* — defined 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 efficiency and an exceptional user experience.

The second critical factor driving Inference costs is time to market. The AI landscape is evolving rapidly, and being first to market can provide a significant 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.

Customer Case Study: Wealthsimple



Figure 2. Wealthsimple Used NVIDIA AI Inference Platform to Reduce Model Delivery Time and Enhance Their Overall Customer Experience

Wealthsimple, a leading Canadian investment management company, sought to enhance its machine learning (ML) model deployment to improve client experiences. Managing over \$50 billion in assets under administration, the company faced delays in deploying new models due to an unstandardized AI inference platform. To address this, Wealthsimple turned to NVIDIA's AI inference platform, achieving remarkable results.

What is Inference Performance and Why It Matters

In the past 12 months, Wealthsimple deployed over 30 AI models, generating 145 million predictions. With NVIDIA's platform in place, the company reduced model delivery time from months to under 15 minutes, drastically improving speed and efficiency. The platform's reliability has been proven, with no IT tickets related to AI inferencing in the past year and a half.

NVIDIA's Triton Inference Server enhanced Wealthsimple's ability to provide ML-driven personalization, such as fraud detection and transaction optimization, ensuring faster and more accurate financial processes. The platform also improved uptime from 95% to 99.999%, eliminating delays for clients and streamlining electronic transfers.

Wealthsimple leverages NVIDIA accelerated computing in the cloud to run their models efficiently, with the support of NVIDIA AI Enterprise for security and stability. This transformation has allowed Wealthsimple to provide cutting-edge services while minimizing downtime, significantly advancing their ML-driven financial services.

Key Results:

- Reduced model deployment time from several months to 15 minutes
- Increased inference uptime from 95% to 99.999%
- Reduced inference serving latency by 20%
- Delivered 145 million predictions with zero IT tickets

How Use Cases Impact Inference

It can be tempting to base hardware infrastructure decisions on a single benchmark when evaluating the performance of different AI accelerators. However, this oversimplified approach often leads to rising system costs over time. Initially provisioned systems may fail to meet the performance demands of a specific use case, necessitating future investments in extra resources. This occurs because inference workloads from different 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 workflows. 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 differ in that question-answering bots are required to generate answers from a predefined 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 workflow 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 efficiency.

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 offers the necessary abstraction tools and blueprints to effectively orchestrate these interactions — is vital to ensure performant inference operations.

As demonstrated, different 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.

Customer Case Study: Perplexity

[Perplexity AI](#), an AI-powered search engine, handles over 435 million queries per month, each involving multiple AI inference requests. To meet growing demand while balancing cost and user experience, Perplexity's inference team adopted NVIDIA H100 Tensor Core GPUs, Triton Inference Server, and TensorRT-LLM. The team serves over 20 AI models simultaneously, including various Llama 3.1 models, and uses smaller classifier models to

match user requests with the appropriate model. Their deployment is managed in a Kubernetes cluster, ensuring service-level agreements (SLAs) are met even during fluctuating traffic.

The Perplexity team focuses on optimizing performance and cost through effective scheduling and batching strategies. They use Triton Inference Server to batch incoming requests and optimize GPU utilization, scaling the deployment based on demand. The team continuously evaluates and adjusts their scheduling algorithms to minimize inter-token latency, aiming to improve performance for latency-sensitive tasks. Additionally, A/B testing is conducted to refine configurations, ensuring that the models meet the required SLAs while maximizing GPU resource utilization.

By deploying models internally using NVIDIA GPUs, Perplexity achieves significant cost savings. Their deployment strategy, which includes [parallelizing models across multiple GPUs and using TensorRT-LLM](#), enables them to serve models at minimal costs while maintaining strict SLAs.

Deployment Factors Impacting Inference Performance

In the previous section, we examined the different performance metrics you can measure and optimize within an inference solution, and how these metrics are influenced by the specific use case. In this section, we will focus on how deployment factors and architectural considerations affect 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 significantly impacts performance. However relying solely on peak chip or instance metrics like rated FLOPS or memory specifications may not tell the full story, especially as delivered workload performance depends greatly on many factors, including the efficiency of the software stack. Similarly, the absolute price of a GPU, or its hourly cost in the cloud, isn't a meaningful metric. What truly matters is the performance delivered per watt per 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 and [AI factories](#) are power limited, making it critical to maximize data center or AI factory inference throughput within a given power budget. Delivered inference throughput for a given amount of energy use – in other words, [energy efficiency](#) – 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 significantly cutting down the time required for resource-intensive tasks. This shift has led to dramatic improvements, offering 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 offers 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.

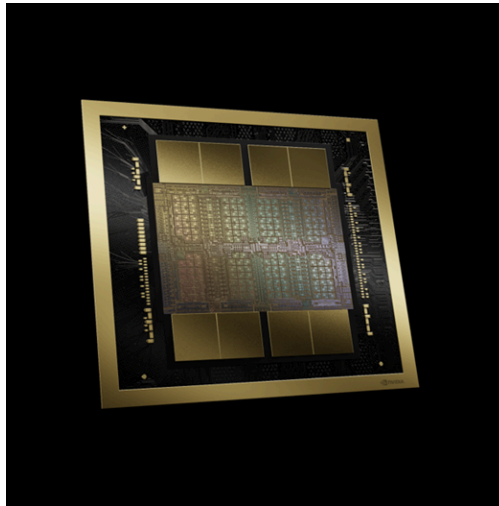


Figure 3. NVIDIA Blackwell GPU Built with 208 Billion Transistors to Deliver Unparalleled Inference Performance

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 offer 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 Energy-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.

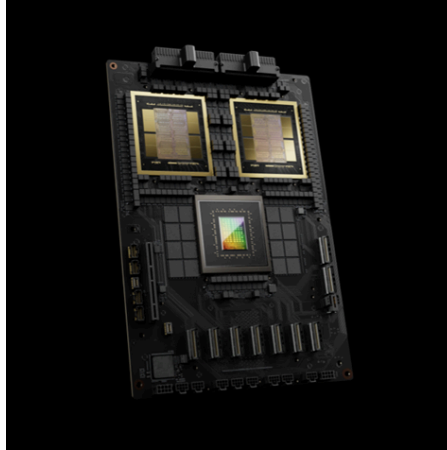


Figure 4. NVIDIA GB200 Superchip Includes Two Blackwell GPUs and One Grace CPU

The NVIDIA Grace CPU combines 72 high-performance and energy-efficient 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 first 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-fifth 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 fifth-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.

GPT-MoE-1.8T Real-Time Throughput

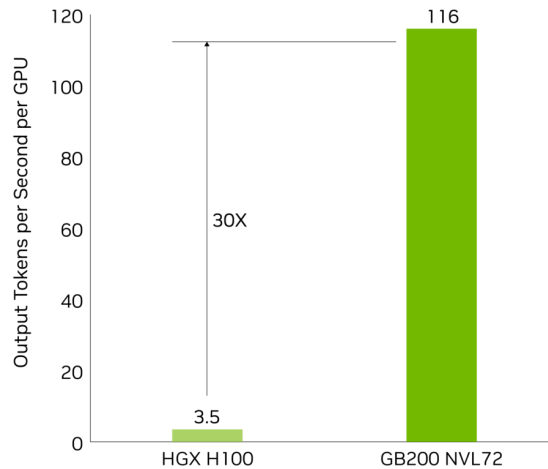


Figure 6. The GB200 Delivers 30x Real-Time Throughput Compared to the H100

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 offer are growing rapidly with each new generation. These next-generation GPUs are highly effective at accelerating massive-scale inferencing, such as serving Mixture of Experts reasoning 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 efficiently 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 offers 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.

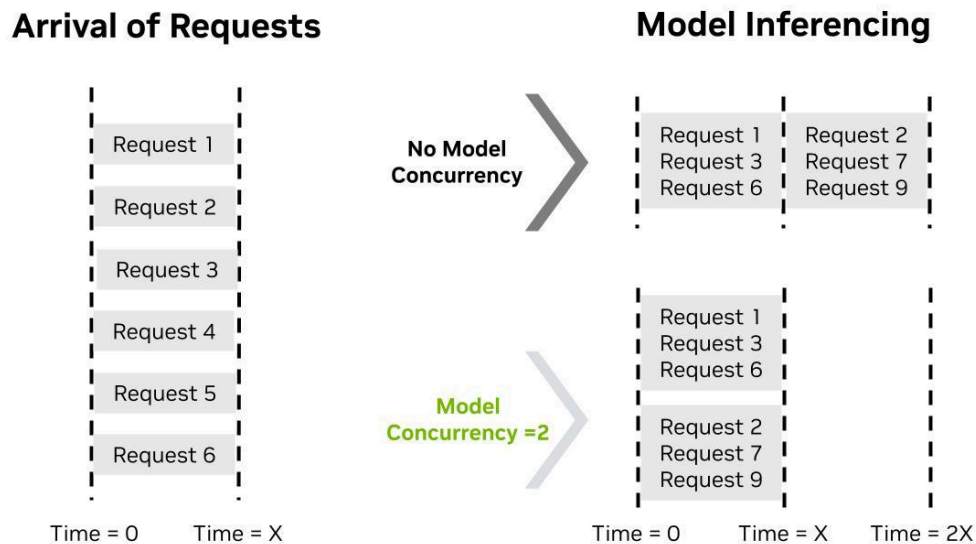


Figure 7. How Increasing Model Concurrency Impacts Throughput and Latency

NVIDIA Dynamo-Triton streamlines the process of deploying models with concurrency enabled. Developers can easily implement model concurrency by switching on a single flag in the model's configuration file. 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 configuration or manual scaling processes.

In scenarios where multiple independent models need to be served but cannot all fit concurrently on the same instance or node, NVIDIA Dynamo-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.

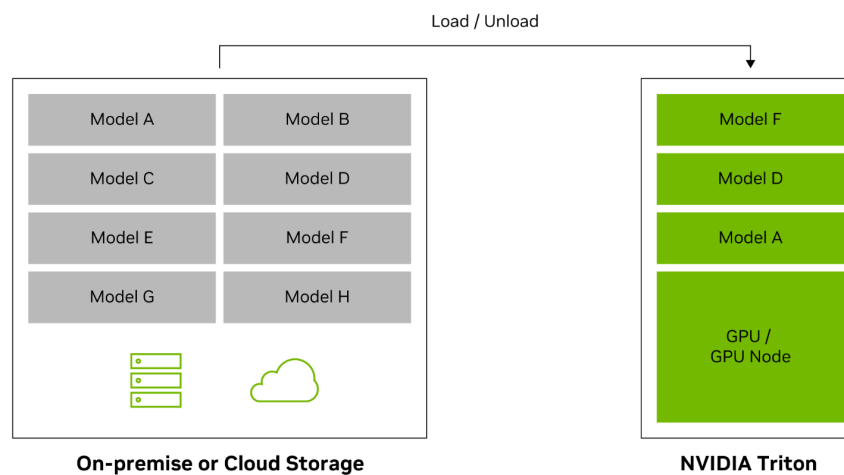


Figure 8. How NVIDIA Dynamo-Triton Can Load and Unload Models on a GPU, Increasing Service Availability and Performance

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 significant benefits:

- **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.

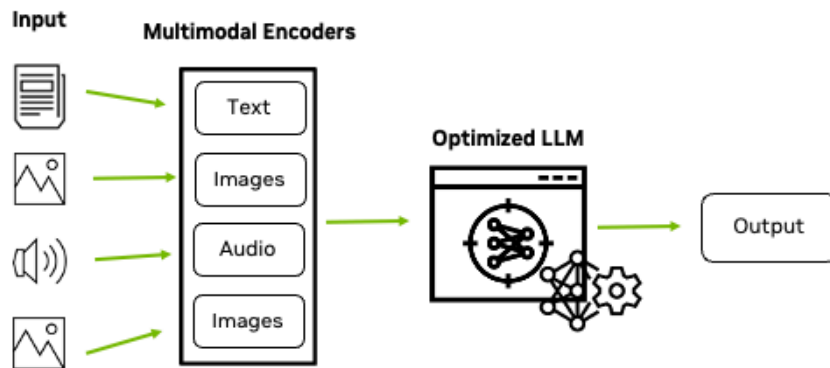


Figure 9. Multimodal LLM Fusing Information from Multiple Modalities and Optimized for High Throughput Output

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 Dynamo-Triton that supports several advanced features such as multi-image inference where a single request can contain multiple images, and effective 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 tradeoff 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 inefficiencies 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 Dynamo-Triton and NVIDIA TensorRT-LLM, can mitigate this tradeoff. 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 specific 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 configured. 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.

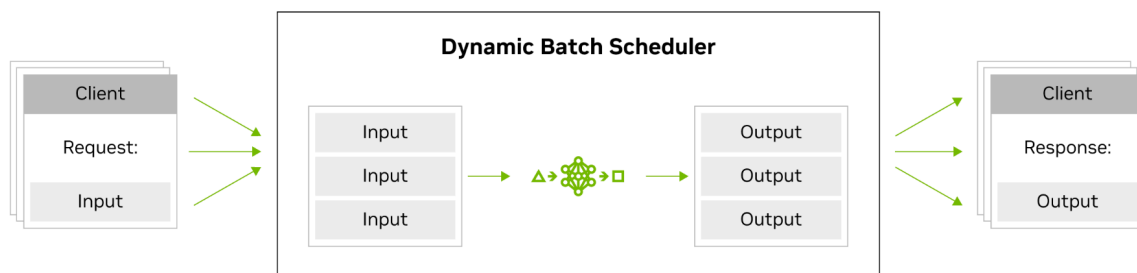


Figure 10. How Dynamic Batching Creates Batches of Multiple Requests

Sequence Batching

For use cases that require specific 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.

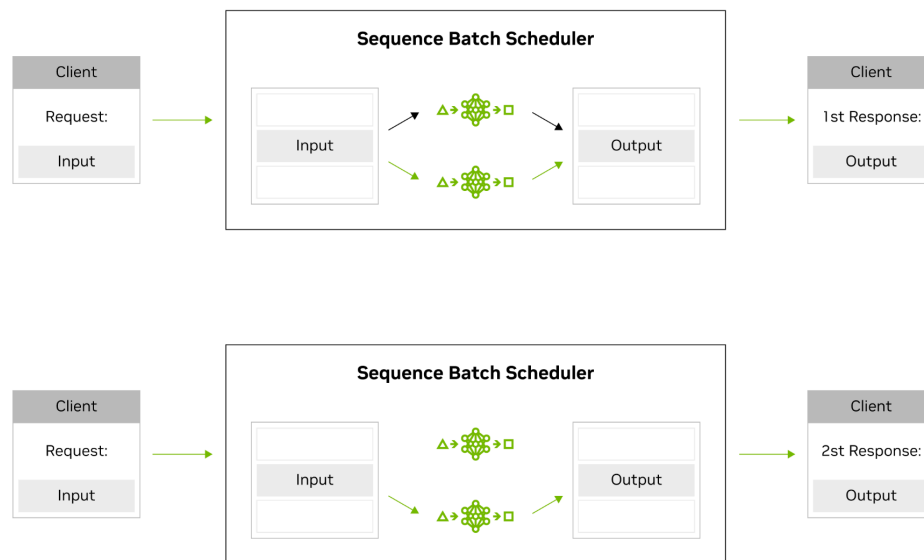


Figure 11. How Sequence Batching Creates Batches of Different Requests While Maintaining Request Relationships and Sequences

Inflight Batching

Inflight batching enhances traditional batch processing by enabling continuous request handling. With inflight 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 Inflight Batching Can Evict Requests within a Batch Once Completed and Replace them with New Ones

Efficient use of compute resources requires careful management of batch size, throughput, and latency. Advanced batching techniques offered by the NVIDIA AI inference platform, such as dynamic batching, sequence batching, and inflight batching, provide organizations with the flexibility 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 effectively.

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 efficiently 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 insufficient for the latest large-scale LLMs, as their model weights often cannot fit onto a single GPU. Consequently, DP is commonly used alongside other parallelism techniques.

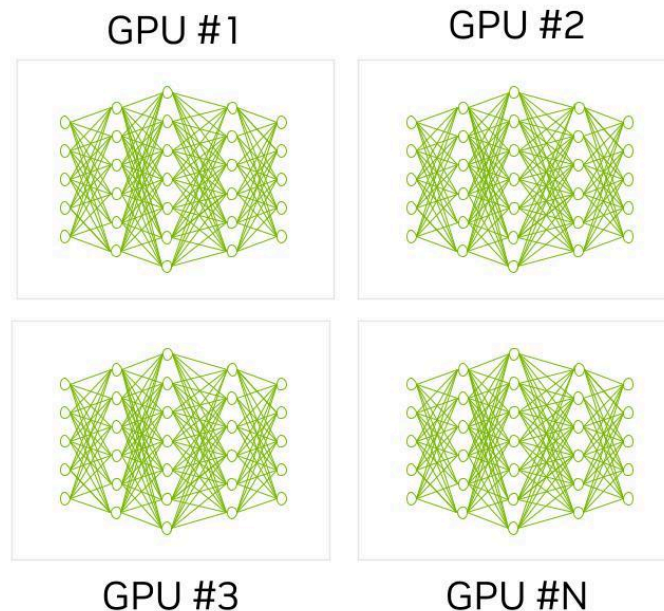


Figure 13. Applying Data Parallelism Across Multiple GPUs

Impact on Performance:

- **GPU Throughput:** Unaffected by DP alone as the model is duplicated.
- **User Interactivity:** No significant 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 different 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.

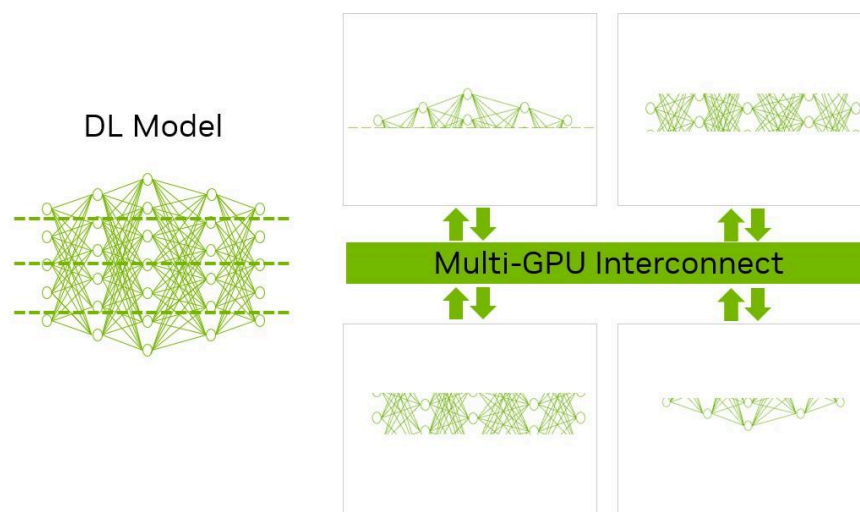


Figure 14. Applying Tensor Parallelism on a Deep Neural Network

Pipeline Parallelism (PP)

Pipeline parallelism divides the model layers, with each group of layers assigned to different GPUs. The model processes requests sequentially across GPUs, with each GPU performing computations on its assigned portion of the model. This method is beneficial for distributing large model weights that do not fit on a single GPU but has limitations in terms of efficiency.

Impact on Performance:

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

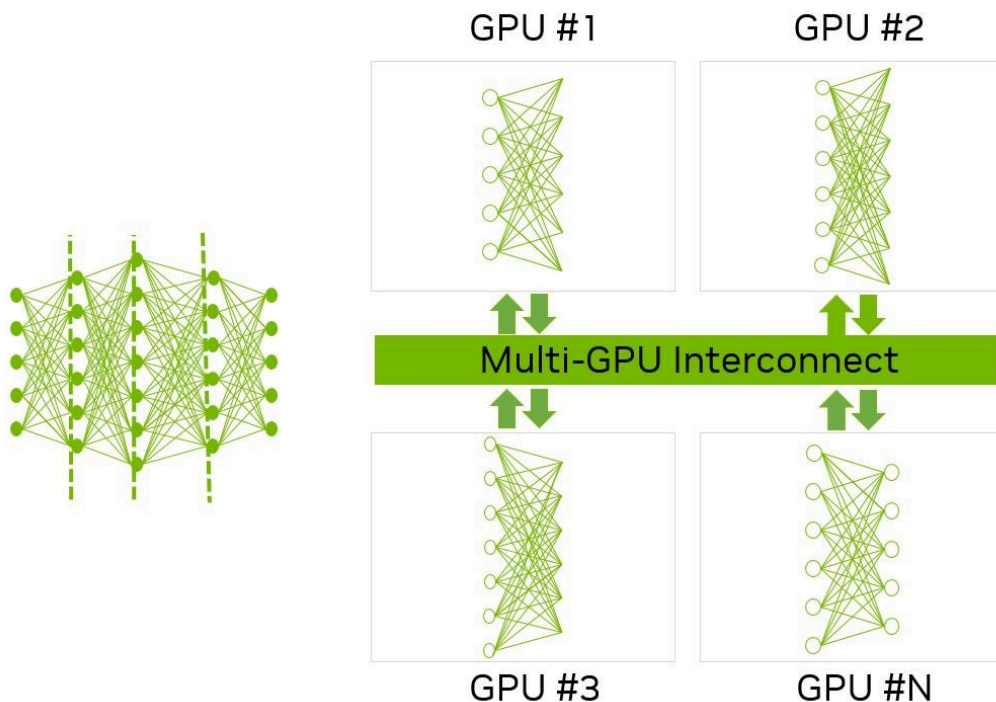


Figure 15. Using Pipeline Parallelism on a Deep Neural Network

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., specific experts), the system reduces computational overhead and optimizes processing. Requests are processed by individual experts before being recombined at the final output stage, which requires high-bandwidth GPU-to-GPU communication.

Impact on Performance:

- **GPU Throughput:** Expert parallelism can offer significant improvements in throughput, especially when combined with other techniques.
- **User Interactivity:** EP configurations can maintain or improve user interactivity without the steep tradeoffs seen in other methods.

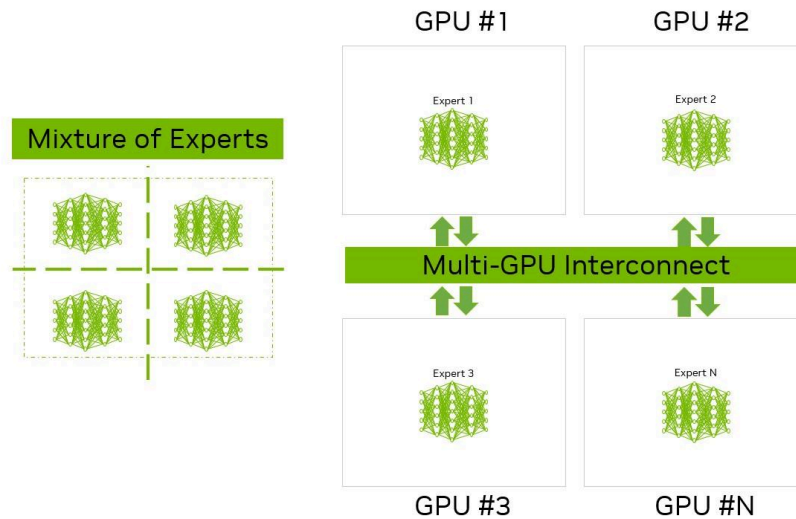


Figure 16. Using Expert Parallelism on a Deep Neural Network that Consists of Four Experts

Combining Parallelism Methods for Optimal Performance

The most effective way to optimize the performance of LLMs across multiple GPUs is by combining multiple parallelism techniques. By doing so, the tradeoff 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) offers a 2x improvement in user interactivity with negligible 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 tradeoffs and benefits 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 configurations 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 efficient AI model deployment.

Unlocking AI Inference Performance and Cost Efficiency in the Cloud

The cloud has long been a transformative force for enterprises, enabling scalable, efficient, and globally accessible IT infrastructures. As businesses transitioned workloads like storage, databases, and enterprise resource planning to the cloud for flexibility and cost savings, AI Inference is following this same path. According to a recent report from IDC, the fastest-growing segment for AI Inference in the enterprise will be on accelerated compute instances in the cloud, with adoption projected to outpace all other deployment models by a factor of three.

For IT leaders, this shift presents a significant opportunity to enhance performance without increasing costs. By combining cloud-based accelerated compute with optimized AI software, organizations can achieve greater speed, scalability, and cost-efficiency. To optimize AI Inference performance in the cloud, IT leaders must ensure their teams have the flexibility to choose from a broad range of different GPU types, matching the right GPU to each specific workload. This strategic choice is key to unlocking the full potential of AI Inference in the cloud by avoiding over- or underprovisioning of accelerated compute resources.

It's equally important to select an AI inference stack that integrates deeply with the cloud tools that IT teams are already trained and certified on. This integration minimizes training costs and allows teams to focus on inference performance optimization instead of dealing with complex software integration tasks. To further increase flexibility and avoid vendor lock-in, IT leaders should prioritize GPUs and Inference stacks that are available across all major cloud service providers (CSPs). This enables the deployment of GPUs closer to end users and gives AI inference teams the freedom to select the best cloud platform for their specific application needs.

One of the standout advantages of using NVIDIA's accelerated compute solutions in the cloud is the flexibility and choice it provides. Different generations of the NVIDIA GPUs power cloud compute instances across all major cloud service providers, offering a wide range of options to help enterprises and developers choose the accelerated compute solutions that best meet their needs.

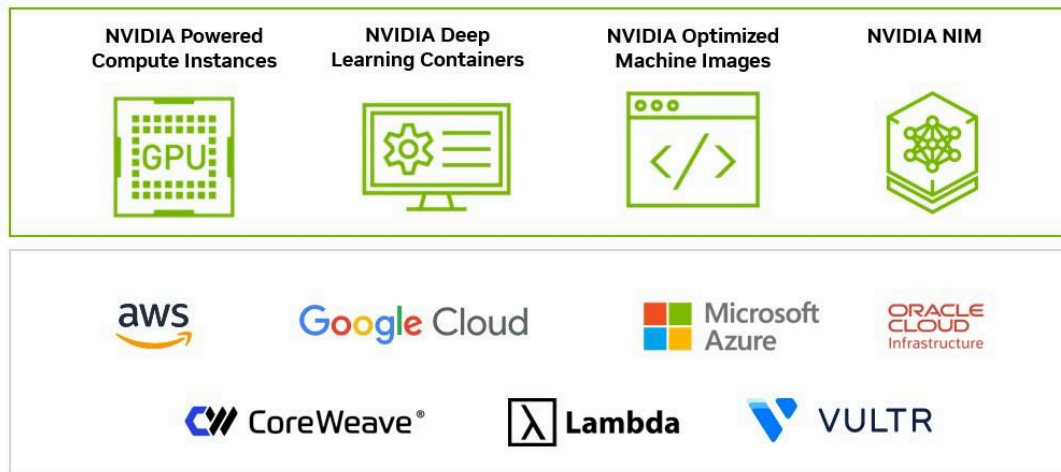


Figure 17. Broad Support of the NVIDIA AI Inference Platform in the Cloud

Additionally, NVIDIA's AI Inference stack is fully integrated with all major cloud MLOps tools, simplifying the deployment and scaling of AI workloads. For example, the NVIDIA Dynamo-Triton and NVIDIA NIM are integrated with all major cloud AI tools, allowing IT teams to deploy models with minimal effort. Whether using AWS SageMaker, Google Vertex AI, Azure ML, or OCI's Data Science Cloud, enterprises can launch the NVIDIA Dynamo-Triton and NVIDIA NIM with low-code or no-code options, saving time and IT resources.

For organizations looking for a more streamlined approach, NVIDIA also offers pre-configured Cloud Machine Images with all necessary drivers and dependencies preinstalled. This enables rapid deployment of AI workloads without manual configuration, allowing businesses to save both time and costs on setup.

By leveraging NVIDIA's accelerated compute and integrated AI Inference solutions, IT leaders can ensure that their AI Inference performance scales effectively in the cloud, with flexibility, cost-efficiency, and minimal overhead.

Customer Case Study: Amdocs

Amdocs, a leading provider of software and services for communications and media providers, developed amAlz—a telco-specific generative AI platform tailored for the telecommunications industry. To enhance performance and accelerate deployment,

Unlocking AI Inference Performance and Cost Efficiency in the Cloud

Amdocs used NVIDIA NIM inference microservices on NVIDIA DGX Cloud for seamless deployment of fine-tuned LLMs for real-time customer care and sales inquiries.

Amdocs curated a human-annotated dataset from anonymized call transcripts, customer interactions and bills used for parameter-efficient fine-tuning techniques, such as Low-Rank Adaptation (LoRA), of LLMs like Meta's Llama family of models and Mistral AI's Mixtral 8x7B. The team automated fine-tuned model deployment with NIM and achieved 80% reduced latency, enabling near real-time responses, as well as 30% accuracy improvements. Domain customization through fine-tuning also reduced token consumption by 40%, lowering infrastructure costs.

With NVIDIA NIM on DGX Cloud, Amdocs significantly improved LLM accuracy and response-times while lowering costs, empowering the amAlz platform to meet the demands of modern Telco applications.

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 fluctuate over time. These forecasts significantly influence 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 inefficient, especially when the peak demand is only temporary or irregular.

To address this, organizations need the flexibility to scale infrastructure up or down, responding to fluctuating 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 off-peak hours, Kubernetes offers a cost-effective 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 fluctuating 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 efficiently 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 offers substantial savings compared to over-provisioning hardware resources to handle peak workloads.

The NVIDIA AI inference platform, including NVIDIA Dynamo-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 NVIDIA Dynamo-Triton using Prometheus and sent to the Kubernetes HPA to inform scaling decisions is the *queue-to-compute ratio*. This ratio reflects the response time of inference requests. It's defined as the queue time divided by the compute time for an inference request.

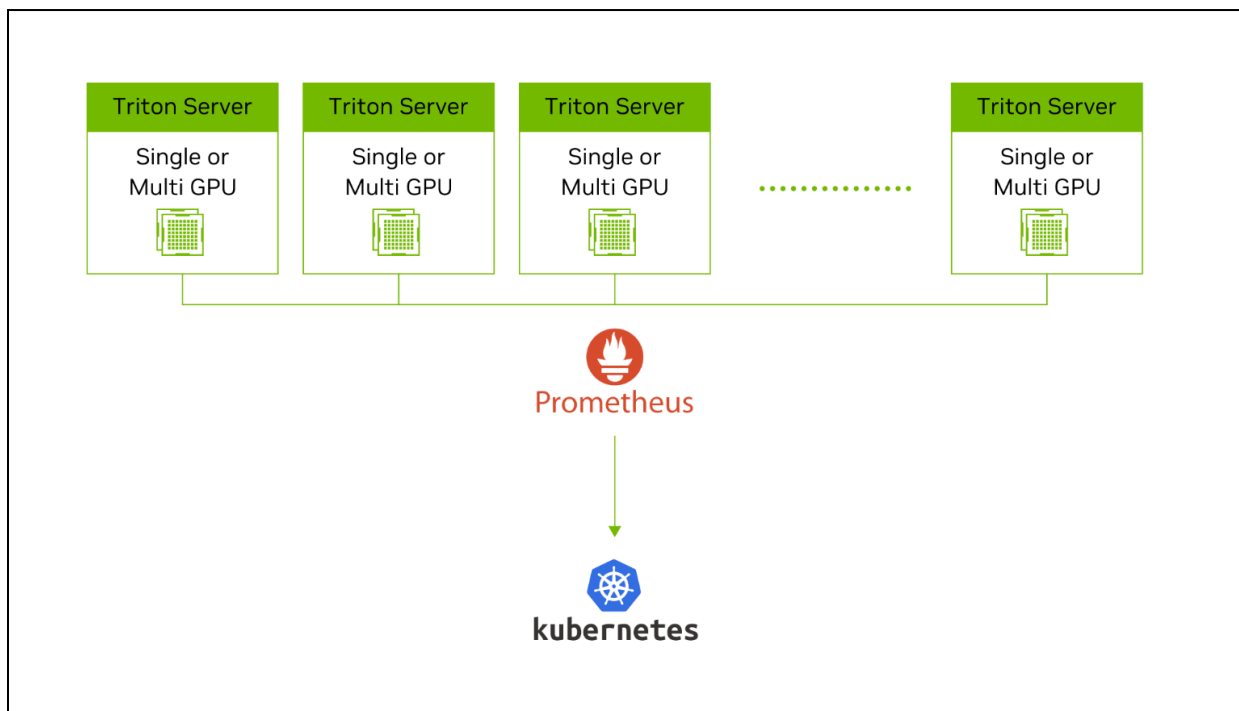


Figure 18. NVIDIA Triton Server Scaling with Kubernetes Architecture

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

Unlocking AI Inference Performance and Cost Efficiency in the Cloud

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 Dynamo-Triton and Kubernetes, visit [Scaling LLMs with NVIDIA Dynamo-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 workflows, especially in enterprise applications.

A Model Ensemble refers to the integration of multiple machine learning models, pre-processing, and post-processing steps into a unified pipeline. Instead of executing models individually, ensembles allow these components to work in tandem, streamlining the process and enhancing overall efficiency. Each model in the ensemble typically handles a different aspect of the task, whether it's processing data, generating predictions, or refining 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 workflow might require an initial zoom-in preprocessing step, followed by the generation of a scene using SDXL, and finishing 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 flow of data through the models while reducing latency and optimizing resource usage.

The Role of NVIDIA Dynamo-Triton in Model Ensembles

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

Dynamo-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, Dynamo-Triton supports running pre- and post-processing on CPUs, while the core AI model can run on GPUs, providing flexibility in balancing processing power. Additionally, Dynamo-Triton supports adding advanced features to the ensemble, such as conditional logic and loops, allowing developers to build more sophisticated and flexible AI workflows.

Benefits of Model Ensembles with Triton

The use of Model Ensembles comes with several key benefits, especially when integrated with Dynamo-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:** Dynamo-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 efficiency of their AI applications, reduce latency, and minimize resource usage.

The NVIDIA AI inference platform provides a powerful platform for managing these ensembles, offering flexibility in resource allocation and simplifying the integration of different 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 performance of their AI models.

Case Study: Let's Enhance Turns Product Photos into Beautiful Marketing Assets

A good example of a company harnessing the power of the NVIDIA AI inference platform to serve SDXL in production environments is [Let's Enhance](#). This pioneering AI startup has been using Triton Inference Server to deploy over 30 AI models on NVIDIA Tensor Core GPUs for over 3 years.

Recently, Let's Enhance celebrated the launch of their latest product, AI Photoshoot, which uses the SDXL model to transform plain product photos into beautiful visual assets for e-commerce websites and marketing campaigns.

With Triton Inference Server's robust support for various frameworks and backends, coupled with its dynamic batching feature set, Vlad Pranskevichus, Let's Enhance Founder and CTO, was able to seamlessly integrate the SDXL model into existing AI pipelines with minimal involvement from the ML Engineering teams, freeing up their time for research and development efforts.

Through a successful proof of concept, the AI image enhancement startup identified a 30% reduction in costs by migrating their SDXL models to NVIDIA L4 GPUs on Google Cloud G2 instances and has outlined a roadmap to complete the migration of several pipelines by mid-2024.

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 NVIDIA Dynamo and TensorRT-LLM, including Chunked Prefill, 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: prefill and decode. In the prefill 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 first token derived from the KV cache. However, traditional methods often struggle with balancing the heavy computational demand of the prefill phase and the lighter load of the decode phase.

Chunked Prefill is an optimization that divides the prefill 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 offers flexibility 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 configuration.

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.

Advanced Inference Techniques to Boost Performance

The Multiblock Attention technique within TensorRT-LLM addresses these challenges by efficiently distributing the decoding process across all Streaming Multiprocessors (SMs) on a GPU. By doing so, it significantly 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-first-token (TTFT), ensuring rapid responses even for complex queries.

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

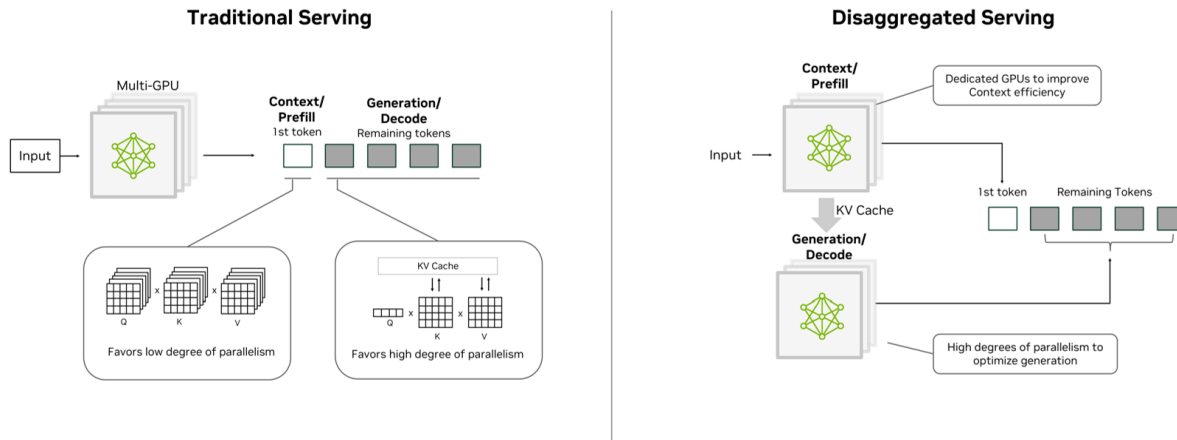
The generation of KV cache is a computationally intensive process. As such, KV cache reuse plays a critical role in speeding up token generation by avoiding redundant computations. However, traditional methods often require waiting for the entire KV cache to be generated before reuse can occur, leading to inefficiencies 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 significantly accelerates inference, especially in scenarios where system prompts or predefined 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 prefill and decode phases on the same GPU, leading to inefficient resource allocation and suboptimal throughput. [NVIDIA Dynamo](#) Disaggregated Serving decouples these phases, allowing AI inference teams to allocate resources independently based on the specific needs of each phase. This enables independent resourcing and decoupled scaling, meaning that more GPUs can be allocated to the prefill 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-specific parallelism — where compute-heavy tasks in the prefill phase can benefit from techniques like Pipeline Parallelism, while the decode phase can leverage Tensor and Expert Parallelism. NVIDIA Dynamo reduces inference costs 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 inefficient. 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 significant 3.5x increase in tokens per second, improving throughput and user experience without compromising output quality. This technique is particularly beneficial in low-latency, high-throughput environments, where maximizing the efficiency of each computational step is essential.

Context-Aware Routing: Avoiding KV Cache Recomputation

Reusing KV cache avoids the need to recompute it from scratch, reducing inference time and compute resources. This is particularly advantageous in use cases where the same request is frequently executed, such as system prompts, single-user multiturn chatbot interactions, and agentic workflows. NVIDIA Dynamo tracks KV cache across large fleets of GPUs in multinode and disaggregated deployments, and efficiently routes incoming requests, minimizing the need for their costly recomputation.

When new inference requests arrive, NVIDIA Dynamo computes an overlap score between incoming requests and the KV cache blocks already active across all the memory of all GPUs in the distributed cluster. By factoring in the overlap score and the distribution of workload across the GPU fleet, it intelligently routes requests to the most suitable workers, minimizing KV cache recomputation while ensuring a balanced load across the cluster.

Advanced Inference Techniques to Boost Performance

By reducing unnecessary re-computation of KV cache, NVIDIA Dynamo frees up GPU resources. This enables AI service providers to respond to more user requests, maximizing returns on their accelerated compute investments.

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 Efficiency in the Cloud](#), will help maximize system efficiency, performance, and minimize costs.

Getting Started with the NVIDIA AI Inference Platform

The NVIDIA AI inference platform offers flexible 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 offers prepackaged, optimized inference microservices for running the latest AI foundation models on NVIDIA accelerated infrastructure anywhere.

For maximum flexibility, configurability and extensibility to fit your unique AI inference needs, NVIDIA offers NVIDIA Triton Inference Server and NVIDIA TensorRT which provide the ability to customize and optimize your inference serving platform for your specific requirements.

Visit [NVIDIA AI Inference Solutions](#) to learn more and get started.