Generative AI in the Enterprise - Inferencing

A Scalable and Modular Production Infrastructure with NVIDIA for Artificial Intelligence Large Language Model Inferencing

October 2023

H19686.1

Design Guide

Abstract

This design guide describes the architecture and design of the Dell Validated Design for Generative AI Inferencing with NVIDIA, a collaboration between Dell Technologies and NVIDIA to enable high performance, scalable, and modular full-stack generative AI Inferencing solutions for large language models in the enterprise.

Dell Generative AI Solutions

Dell
Validated Design

DELL Technologies

Contents

Chapter 1 Introduction

Overview

Generative AI, the branch of artificial intelligence (AI) that is designed to generate new data, images, code, or other types of content that humans do not explicitly program, is rapidly becoming pervasive across nearly all facets of business and technology.

Inferencing, the process of using a trained AI model to generate predictions that make decisions or produce outputs based on input data, plays a crucial role in generative AI as it enables the practical application and real-time generation of content or responses. It enables near instantaneous content creation and interactive experiences, and when properly designed and managed, does so with resource efficiency, scalability, and contextual adaptation. It allows generative AI models to support applications ranging from chatbots and virtual assistants to context-aware natural language generation and dynamic decision-making systems.

Earlier this year, Dell Technologies and NVIDIA introduced a groundbreaking project for generative AI, with a joint initiative to bring generative AI to the world’s enterprise data centers. This project delivers a set of validated designs for full-stack integrated hardware and software solutions that enable enterprises to create and run custom AI large language models (LLMs) on-premises using unique data that is relevant to their own organization.

An LLM is an advanced type of AI model that has been trained on an extensive dataset, typically using deep learning techniques, which is capable of understanding, processing, and generating natural language text. However, AI built on public or generic models is not well suited for an enterprise to use in their business. Enterprise use cases require domain-specific knowledge to train, customize, and operate their LLMs.

In addition, operating LLMs for inferencing on-premises is inherently more secure for the enterprise, due to stronger data control, reduced exposure to external networks, better control over compliance adherence, improved protection from third-party vulnerabilities, and the ability to implement customized security measures tailored to their infrastructure and environment.

Dell Technologies and NVIDIA have designed a scalable, modular, and high-performance architecture that enables enterprises everywhere to create a range of generative AI solutions that apply specifically to their businesses, reinvent their industries, and give them competitive advantage.

This design for inferencing is the first in a series of validated designs for generative AI that focus on all facets of the generative AI life cycle, including inferencing, model customization, and model training. While these designs are focused on generative AI use cases, the architecture is more broadly applicable to more general AI use cases as well.

Document purpose

This guide describes the Dell Validated Design for Generative AI Inferencing with NVIDIA.

It describes the validated design and reference architecture for a modular and scalable platform for generative AI in the enterprise. The guide focuses specifically on inferencing, which is the process of serving a trained model to generate predictions, make decisions, or produce outputs based on input data for production outcomes. Subsequent guides will address validated designs for model customization and training.

This design guide can be read with the associated Generative AI in the Enterprise white paper. The white paper provides an overview of generative AI, including its underlying principles, benefits, architectures, and techniques; the various types of generative AI models and how they are used in real-world applications; the challenges and limitations of generative AI; and descriptions of the various Dell and NVIDIA hardware and software components to be used in the series of validated designs to be released.

Audience

This design guide is intended for experts interested in the implementation of solutions and infrastructure for generative AI, including professionals and stakeholders involved in the development, deployment, and management of generative AI systems.

Key roles include AI architects and IT infrastructure architects and designers**.** Other audience members might include system administrators and IT operations personnel, AI engineers and developers, and data scientists and AI researchers. Some knowledge of generative AI principles and terminology is assumed, including familiarity with the associated white paper.

Revision history

Table 1. Revision History

Date	Version	Change summary
July 2023	1	Initial release
October 2023	2	Added validation and configuration data for Dell PowerEdge XE8640 and XE9680 servers Added support for NVIDIA Base Command Manager Essentials and NVIDIA AI Enterprise 4.0

Chapter 2 AI Model Inferencing

This chapter describes inferencing in generative AI, and how inferencing fits into the general workflow of generative AI development and operations. It also presents numerous use cases for both language-based and nonlanguage examples of inferencing.

What is inferencing?

Inferencing in generative AI is the process of using a pretrained model to generate predictions, make decisions, or produce outputs based on specific input data and contexts. It applies the learned knowledge and patterns acquired during the model's training phase to respond with new and unique content. It represents a crucial step in leveraging the capabilities of generative models for a wide range of applications.

At the core of generative AI is a model that has been trained on vast amounts of data to understand and generate human-like text, images, or other forms of content. During the training phase, the model learns to recognize patterns, relationships, and structures in the data. For instance, a language model like Generative Pre-trained Transformer (GPT) learns the statistical properties of language, enabling it to generate coherent and contextually relevant text.

During inferencing, the trained model processes input data through its computational algorithms or neural network architecture to produce an output or prediction. The model applies its learned parameters, weights, or rules to transform the input data into meaningful information or actions.

Inferencing is the culminating and operational stage in the life cycle of an AI system. After training a model on relevant data to learn patterns and correlations, inferencing allows the model to generalize its knowledge and make predictions or generate responses that are accurate and appropriate to the specific context of the business.

For example, in a natural language processing task like sentiment analysis, the model is trained on a labeled dataset with text samples and corresponding sentiment labels (positive, negative, or neutral). During inferencing, the trained model takes new, unlabeled text data as input and predicts the sentiment associated with it.

Inferencing can occur in various contexts and applications, such as image recognition, speech recognition, machine translation, recommendation systems, and chatbots. It enables AI systems to provide meaningful outputs, help with decision-making, automate processes, or interact with users based on the learned knowledge and patterns captured by the model during training. Generative AI inferencing is what allows AI systems to produce coherent and contextually relevant responses or content in real time.

Inferencing use cases

Inferencing using LLMs for natural language generation in generative AI has numerous practical use cases across various domains. The Generative AI in the Enterprise white paper discusses multiple use cases for generative AI across various industries. Some particular examples of use cases based specifically on inferencing include the following:

- **Intelligent documentation creation and processing** – Inferencing is used extensively or content generation including natural language and text generation tasks such as document writing, dialogue generation, summarization, or new content creation. It can be used to create drafts, outlines, or final content or documents and reports. It can be used to generate concise summaries of longer texts, making information more accessible and digestible. It can help in creating technical documentation, providing detailed explanations and instructions. It can assist in language translation, ensuring that documentation is accessible to a wider audience.

  With models trained to an organization's unique data, inferencing can be used effectively for documentation and content creation, so businesses can streamline their processes, improve productivity, and ensure that their content is relevant and of high quality.

- **Code generation, assistance, and documentation** – Software development can use inferencing applications is multiple ways to assist or automate various tasks. It can perform code generation or code autocompletion, based on high-level descriptions or prompts and based on context, making the writing of code faster and more efficient. It can also assist with refactoring or syntax error detection, generating test cases, and debugging. It can generate documentation or inline comments for code, improving its readability and maintainability. By using inferencing in coding tasks, developers can accelerate their workflow, reduce manual effort, and potentially discover more efficient or optimized solutions.

- **Marketing and sales** – Inferencing is used in marketing and sales to automate and enhance communication with customers. It allows for personalized responses, product recommendations, and content creation. For instance, in chatbots, it interprets customer queries and generates relevant responses, improving customer support. Additionally, it assists in creating targeted marketing content, such as personalized email messages or social media posts, which helps create market demand and can lead to higher customer engagement and conversion rates.

- **Sentiment analysis** – Inferencing can analyze customer sentiment and emotional cues from their messages or interactions. This analysis allows organizations to monitor customer satisfaction levels, identify potential issues, and take proactive measures to address concerns.

  Related, Named Entity Recognition (NER) is a natural language processing (NLP) technique used to identify and categorize specific entities within text, to extract and classify them to provide more context and meaning to the text. It works with sentiment analysis, where understanding context is crucial for accurate analysis and responses.

- **Customer service** – Customer service and support activities use conversational agents, chatbots, and virtual assistants extensively by generating natural language responses based on user queries or instructions. In addition, there are multiple other applications of inferencing in customer service and troubleshooting environments, including applications such as:

- **Self-service knowledge bases** – Generative AI can automatically generate and update knowledge base articles, FAQs, and troubleshooting guides. When customers encounter issues, they can search the knowledge base to find relevant self-help resources that provide step-by-step instructions or solutions.
- **Contextual responses, problem solving, and proactive troubleshooting** – Generative AI can analyze customer queries or problem descriptions and generate contextually relevant responses or troubleshooting suggestions. By understanding the context, the AI system can offer tailored recommendations, guiding customers through the troubleshooting process.
- **Interactive diagnostics** – Generative AI can simulate interactive diagnostic conversations to identify potential issues and guide customers towards resolution. Through a series of questions and responses, the system can narrow down the problem, offer suggestions, or provide next steps for troubleshooting.
- **Intelligent routing and escalation** – Generative AI models can intelligently route customer inquiries or troubleshoot specific issues based on their complexity or severity. They can determine when a query must be escalated to human support agents, ensuring efficient use of resources and timely resolution.

While the validation work that we performed in this design is centered primarily on language and text applications such as those applications described above, there are also other non-LLM use cases of generative AI inferencing that include:

- **Image synthesis**—Generative AI models can generate initial realistic images or modify existing images by applying various transformations, such as style transfer, image inpainting, or super-resolution.
- **Music composition**—Generative AI can create music compositions, harmonies, melodies, or even entire music tracks in various genres or styles based on the learned patterns from training data.
- **Video generation**—Generative AI models can synthesize new video content or modify existing videos, enabling applications such as video completion, deepfake creation, or video enhancement.
- **Virtual worlds and environments**—Generative AI can generate virtual worlds, landscapes, or architectural designs for use in video games, virtual reality (VR), or simulations.
- **Virtual characters and avatars**—Generative AI can create virtual characters, avatars, or digital personas that exhibit specific traits, behaviors, or personalities.

These examples show how inferencing in generative AI is applied across various domains. The versatility of generative AI allows for creative and innovative applications in content generation, creative arts, virtual environments, personalization, customer service, and more. The use cases continue to expand as generative AI technologies advance.

Challenges with inferencing

Inferencing, the process of using a trained model to generate predictions or responses, can come with several challenges. Common challenges associated with inferencing in AI include:

- **Computational resources**—Inferencing can be computationally intensive, especially for large and complex models. Even though inferencing can be less demanding than model training or fine-tuning, generating predictions or responses in real time might require significant processing power, memory, and efficient use of hardware resources.
- **Latency and responsiveness**—Achieving low-latency and highly responsive inferencing is crucial in many real-time applications. Balancing the computational demands of the model with the need for fast responses can be challenging, particularly when dealing with high volumes of concurrent user requests.
- **Model size and efficiency**—LLMs, such as GPT-3, can have millions or even billions of parameters. Deploying and running such models efficiently, particularly on resource-constrained devices or in edge computing scenarios, can be a challenge due to memory and storage requirements.
- **Deployment scalability**—Scaling up the deployment of a model handles increasing user demand. Ensuring that the system can handle concurrent inferencing requests and dynamically allocate resources to meet the workload can be complex, requiring careful architecture design and optimization.
- **Model optimization and compression**—Optimizing and compressing models for inferencing is necessary to reduce memory and computational requirements, enabling efficient deployment on various devices or platforms. Balancing the trade-off between model size, inference speed, and accuracy is a nontrivial task.
- **Explainability and interpretability**—Understanding and explaining the reasoning behind the model's predictions or responses is crucial in many applications, particularly in domains where accountability, transparency, and ethical considerations are of paramount importance. Ensuring the interpretability of the model's decisions during inferencing can be a challenge, especially for complex models like deep neural networks.
- **Quality control and error handling**—Detecting and handling errors or inaccuracies during inferencing is important to maintain the quality and reliability of the system. Implementing effective error handling, monitoring, and quality control mechanisms to identify and rectify issues is essential.

These challenges highlight the need for careful consideration and optimization in various aspects of inferencing, ranging from computational efficiency and scalability to model optimization, interpretability, and quality control. Addressing these challenges effectively contributes to the development of robust and reliable AI systems for inferencing.

Dell Technologies and NVIDIA help solve these challenges by collaborating to deliver a validated and integrated hardware and software solution, built on Dell high-performance best-in-class infrastructure, and using the award-winning software stack and the industry-leading accelerator technology and AI enterprise software stack of NVIDIA.

Inferencing concepts

There are several key architectural concepts that are related to generative AI inferencing and relevant to this design, including LLM characteristics and examples.

Large language models

LLMs are advanced natural language processing models that use deep learning techniques to understand and generate human language. LLMs can include a range of architectures and approaches, such as recurrent neural networks (RNNs), transformers, or even rule-based systems. Generative Pre-trained Transformer (GPT) is a popular and influential example of an LLM that is based on transformer architecture, which is a deep neural network architecture designed to handle sequential data efficiently. Transformers use self-attention mechanisms to process input sequences and learn contextual relationships between words, enabling them to generate coherent and contextually relevant language.

Foundation models

A foundation or pre-trained model is a machine learning model that has been trained on a large dataset for a specific task before it is fine-tuned or adapted for a more specialized task. These models are typically trained on vast amounts of general data to learn basic features, patterns, and context within the data.

Foundation models are crucial because they provide a starting point that already understands a broad range of concepts and language patterns. This makes the process of customizing and fine-tuning for specific tasks much more effective and efficient. While this design is focused on inferencing using existing foundation models, a subsequent design will address model customization, including fine tuning and other methods.

Parameters

Parameters in LLMs refer to the learnable components or weights of the neural network that make up the model. These parameters determine how the model processes input data and makes predictions or generates output. Typically, GPTs have millions (M) to billions (B) of parameters. These parameters are learned during the training process, in which the model is exposed to vast amounts of data and adjusts its parameters to generate language. Assuming the model architecture and training data are comparable, generally the higher the parameters in the model, the greater the accuracy and capability of the models. Although sometimes a smaller model that is trained to be specific to a particular outcome may be more accurate. Models with higher parameters also require more compute resources, especially GPU resources. So there is a balance to be considered when choosing a model.

Accuracy

Accuracy of LLMs is typically measured based on their performance on specific natural language processing (NLP) tasks. The evaluation metrics used depend on the nature of the task. Some commonly used tools to evaluate LLMs include ARC, HellaSwag, and Stanford Question Answering Dataset (SQuAD). HuggingFace maintains a leader board for the open LLM models.

Publicly available NVIDIA NeMo, BLOOM, and Llama models are foundation models. While these models offer a strong starting point with general capabilities, they are typically customized for specific use. This design guide does not address model customization.

Tokenization

Tokenization in generative AI refers to the process of breaking down a piece of text into smaller units, called tokens. These tokens can be words, subwords, or even characters, depending on the granularity chosen for the tokenization process.

In NLP tasks, tokenization is a critical step because it transforms continuous text into discrete units that machine learning models can process. By segmenting text into tokens, the model gains a structured representation of the input on which it can then analyze, understand, and generate responses. The choice of tokenization strategy (word-level, subword-level, character-level) and the specific tokenizer used can significantly impact the model's performance on various tasks.

Examples of LLMs

We have validated several foundation models for inferencing with this infrastructure design, including most of the NeMo GPT models (ranging from 345 M to 20 B parameters) with Triton Inference Server. We also ran inferencing on BLOOM, Llama, and Stable Diffusion models on standard Python or PyTorch containers that are available from the NVIDIA NGC Catalog. For more information about the models we explicitly validated, see Table 9 in Validation Results.

NVIDIA NeMo GPT models

GPT-style models are decoder-only transformer models. NVIDIA NeMo GPT models are trained using the NeMo framework. Several NeMo GPT models with varying parameter sizes exist, including 345 M, 1.3 B, 2 B, 5 B, and 20 B parameters. These models were trained on the Pile-dataset, an 825 GiB English text corpus specifically curated by Eleuther.AI for training LLMs. The 2 B parameter model, for instance, was trained on 1.1 T tokens encompassing 53 languages and code.

BLOOM

BigScience Large Open-science Open-access Multilingual Language Model (BLOOM) is an autoregressive LLM developed collaboratively by over 1000 researchers. It is based on a decoder-only transformer architecture. The model was trained on the ROOTS corpus, which consists of sources from 46 natural and 13 programming languages, totaling 1.61 TB of text. BLOOM offers multiple models with different parameter sizes, including 560 M, 1 B, 3 B, 7 B, and 176 B.

Llama 2

Llama 2, jointly developed by Meta and Microsoft, is freely available for research and commercial use. It offers a collection of pretrained models for generative text and fine-tuned models optimized for chat use cases. The Llama 2 models are trained on an extensive 2 T tokens dataset, featuring double the context length of Llama 1. Moreover, Llama 2-chat models have been further enriched through over 1 million new human annotations. These models are built on an optimized transformer architecture and come in various parameter sizes, including 7 B, 13 B, and 70 B.

Stable Diffusion

Stable Diffusion is a latent, text-to-image diffusion model. Latent diffusion models (LDMs) operate by repeatedly reducing noise in a latent representation space and then converting that representation into a complete image. Although Stable Diffusion is not an LLM, it was validated to illustrate the capability of this architecture to carry our inferencing for other generative AI modalities, such as image generation.

Chapter 3 Solution Components

In this chapter, we describe the primary software components used for inferencing, including NVDIA AI Enterprise components such as Triton Inference Server, the NeMo framework for generative AI models, and the NVIDIA Base Command Manager Essentials.

We also explain the Dell PowerEdge servers and NVIDIA GPUs used in the design, including GPU configurations and GPU connectivity and networking methods.

Inferencing using NVIDIA AI Enterprise

NVIDIA AI Enterprise provides enterprise support for various software frameworks, toolkits, workflows, and models that support inferencing. See the NVIDIA AI Enterprise documentation for more information about all components available with NVIDIA AI Enterprise. The following components incorporated in this validated design are available as part of NVIDIA AI Enterprise:

- Triton Inference Server with FasterTransformer and Model Analyzer
- NVIDIA NeMo models
- NVIDIA Base Command Manager Essentials

The following sections describe the key software components and how they are used in this design. For more information about how these key components work together, see Figure 4 in Software architecture.

Triton Inference Server

NVIDIA Triton Inference Server (also known as Triton) is inference serving software that standardizes AI model deployment and execution and delivers fast and scalable AI in production. Enterprise support for Triton is available through NVIDIA AI Enterprise. It is also available as an open-source software.

Triton streamlines and standardizes AI inference by enabling teams to deploy, run, and scale trained machine learning or deep learning models from any framework on any GPU- or CPU-based infrastructure. It provides AI researchers and data scientists the freedom to choose the appropriate framework for their projects without impacting production deployment. It also helps developers deliver high-performance inference across cloud, on-premises, edge, and embedded devices.

Benefits

The benefits of Triton for AI inferencing include the following:

- **Support for multiple frameworks**—Triton supports all major training and inference frameworks, such as TensorFlow, NVIDIA TensorRT, NVIDIA FasterTransformer, PyTorch, Python, ONNX, RAPIDS cuML, XGBoost, scikit-learn RandomForest, OpenVINO, custom C++, and more.

- **High-performance AI inference**—Triton supports all NVIDIA GPU-, x86-, Arm CPU-, and AWS Inferentia-based inferencing. It offers dynamic batching, concurrent execution, optimal model configuration, model ensemble, and streaming audio/video inputs to maximize throughput and utilization.
- **Designed for DevOps and MLOps**—Triton integrates with Kubernetes for orchestration and scaling, exports Prometheus metrics for monitoring, supports live model updates, and can be used in all major public cloud AI and Kubernetes platforms. It is also integrated into many MLOps software solutions.
- **Support for model ensembles**—Because most modern inference requires multiple models with preprocessing and postprocessing to be run for a single query, Triton supports model ensembles and pipelines. Triton can run the parts of the ensemble on CPUs or GPUs and allows multiple frameworks inside the ensemble.
- **Enterprise-grade security and API stability**—NVIDIA AI Enterprise includes NVIDIA Triton for production inference, accelerating enterprises to the leading edge of AI with enterprise support, security, and API stability while mitigating the potential risks of open-source software.

Triton Inference Server is at the core of this validated design. It is the software that hosts generative AI models. Triton Inference Server, along with its integration with Model Analyzer, Fast Transformer, and the NeMo framework provides an ideal software for deploying generative AI models.

Faster Transformer

In NLP, the encoder and decoder are crucial components, and the transformer layer has gained popularity as an architecture for both. FasterTransformer from NVIDIA offers a highly optimized transformer layer for both the encoder and decoder, specifically designed for efficient inference.

When running on NVIDIA GPUs, FasterTransformer automatically uses the computational power of Tensor Cores, especially when the data and weights are represented in FP16 precision, enabling faster computations.

FasterTransformer is built using CUDA, cuBLAS, cuBLASLt, and C++. It provides convenient APIs for popular deep learning frameworks like TensorFlow, PyTorch, and Triton backend. These APIs allow users to seamlessly integrate FasterTransformer into their existing workflows using these frameworks.

In this validated design, we use FasterTransformer for inferencing NVIDIA NeMo models.

NVIDIA recently unveiled TensorRT-LLM as a new backend for hosting LLMs. It encompasses the TensorRT deep learning compiler, featuring optimized kernels, preprocessing and postprocessing workflows, and specialized multi-GPU and multinode communication components, all of which contribute to remarkable performance gains on NVIDIA GPUs. TensorRT-LLM empowers developers to explore novel LLMs, providing both peak performance and rapid customization options. For additional insights and in-depth information, see NVIDIA's Technical Blog. At the time of writing, TensorRT-LLM is in an early access phase, and this white paper will be updated when it becomes generally available.

Triton Model Analyzer

The Triton Inference Server offers a powerful solution for deploying AI models. However, each deployment presents its unique challenges, such as meeting latency targets, working with limited hardware resources, and accommodating various model requirements. To address these complexities, the Model Analyzer provides insight for planning and decision making.

The Model Analyzer enables users to send requests to their models while monitoring GPU memory and compute use. This tool provides a mechanism for a deep understanding of an AI model's GPU memory requirements under different batch sizes and instance configurations. Using this information, users can make informed decisions about efficiently combining multiple models on a single GPU, ensuring optimal memory usage without exceeding capacity.

The Model Analyzer is a command-line interface (CLI) that significantly enhances comprehension of Triton Inference Server model compute and memory demands. It provides this awareness by conducting customizable configuration "sweeps" and generating comprehensive reports summarizing performance metrics.

With the Model Analyzer, you can:

- Conduct tailored configuration sweeps to identify the most suitable setup for your specific workload and hardware.
- Obtain detailed reports, metrics, and graphs, summarizing findings related to latency, throughput, GPU resource use, power draw, and more. This information facilitates performance comparison across different configurations.
- Fine-tune model deployments to meet Quality of Service requirements, such as specific latency requirements, GPU memory use, and minimum throughput.

By using the insights provided by the Model Analyzer, you can make better-informed decisions and optimize their AI model deployments for peak performance and efficiency.

In this validated design, we use the Model Analyzer to generate load and monitor the performance of LLM model inference. We used Prometheus and Grafana to gather and visualize the performance metrics. For the results of our validation, which uses Model Analyzer, see Chapter 5. For our sizing guidelines based on the reports of Model Analyzer, see Chapter 6.

NVIDIA NeMo models

NVIDIA supports various generative AI models available that can be used for inferencing or on which to perform model customization of transfer learning.

In this validated design, we used open-source NeMo GPT models to demonstrate inference of LLM models in our solutions. Also, we used the NeMo toolkit available in the NGC Catalog to deploy those models. Our validation efforts were performed with that NVIDIA NeMo Docker container.

NVIDIA Base Command Manager Essentials

NVIDIA Base Command Manager Essentials facilitates seamless operationalization of AI development at scale by providing features like operating system provisioning, firmware upgrades, network and storage configuration, multi-GPU and multinode job scheduling, and system monitoring. It maximizes the use and performance of the underlying hardware architecture.

In this validated design, we use NVIDIA Base Command Manager Essentials for:

- Bare metal provisioning, including deploying the operating system and drivers, and configurating local storage in PowerEdge compute nodes
- Network configuration, including configuring networks for PXE boot, internal node access, POD networking, and storage networking
- Kubernetes deployment, including configuring control plane node and worker nodes, access control and provision of Kubernetes management toolkits and frameworks like Prometheus
- NVIDIA software deployment, including deploying NVIDIA GPU operator and Fabric Manager
- Cluster monitoring and management, including health monitoring, fault tolerance, resource utilization monitoring, software and package management, security and access control, and scalinginfohub

Dell PowerEdge Servers and NVIDIA GPUs

Dell Technologies provides a diverse selection of acceleration-optimized servers with an extensive portfolio of accelerators featuring NVIDIA GPUs. In this design, we showcase three Dell PowerEdge servers tailored for generative AI purposes:

- PowerEdge R760xa server, capable of supporting up to four NVIDIA H100 GPUs or four NVIDIA L40 GPUs
- PowerEdge XE8640 server, supporting four NVIDIA H100 GPUs
- PowerEdge XE9680 server, supporting eight NVIDIA H100 GPUs

In this section, we describe the configuration and connectivity options for NVIDIA GPUs, and how these server-GPU combinations can be applied to various LLM use cases.

NVIDIA GPUs configurations

This design for inferencing supports several options for NVIDIA GPU acceleration components. The following table provides a summary of the GPUs used in this design:

Table 2. NVIDIA GPUs – Technical specifications and use cases

	NVIDIA H100 SXM GPU	NVIDIA H100 PCIe GPU	NVIDIA L40 PCIe GPU
Supported latest PowerEdge servers (and maximum number of GPUs)	PowerEdge XE9680 (8) PowerEdge XE8640 (4)	PowerEdge R760xa (4) PowerEdge R760 (2)	PowerEdge R760xa (4) PowerEdge R760 (2)
GPU memory	80 GB	80 GB	48 GB
Form factor	SXM	PCIe (dual width, dual slot)	PCIe (dual width, dual slot)
GPU interconnect	900 GB/s PCIe	600 GB/s NVLink Bridge supported in PowerEdge R760xa 128 GB/s PCIe Gen5	None

	NVIDIA H100 SXM GPU	NVIDIA H100 PCIe GPU	NVIDIA L40 PCIe GPU
Multi-instance GPU support	Up to 7 MIGs	Up to 7 MIGs	None
Decoders	7 NVDEC 7 JPEG	7 NVDEC 7 JPEG	3 NVDEC 3 NVENC
Max thermal design power (TDP)	700 W	350 W	300 W
NVIDIA AI Enterprise	Add-on	Included with H100 PCIe	Add-on
Most applicable use cases	Generative AI training Large scale distributed training	Discriminative/Predictive AI Training and Inference Generative AI Inference	Small scale AI Visual computing Discriminative/Predictive AI Inference

GPU memory and connectivity

NVIDIA GPUs support various options to connect two or more GPUs, offering various bandwidths. GPU connectivity is often required for certain multi-GPU applications, especially when higher performance and lower latency are crucial. LLMs often do not fit in the memory of a single GPU and are typically deployed spanning multiple GPUs. Therefore, these GPUs require high-speed connectivity between them.

GPU connectivity

NVIDIA NVLink is a high-speed interconnect technology developed by NVIDIA for connecting multiple NVIDIA GPUs to work in parallel. It allows for direct communication between the GPUs with high bandwidth and low latency, enabling them to share data and work collaboratively on compute-intensive tasks.

The following figure illustrates the NVIDIA GPU connectivity options for the PowerEdge servers used in this design:

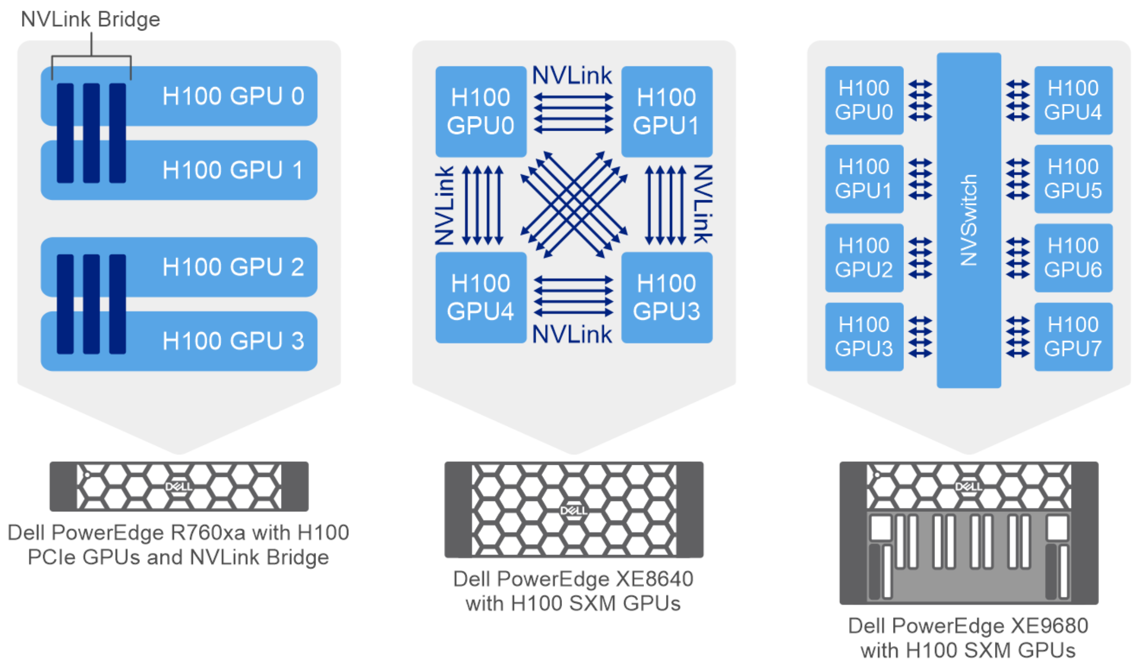


Figure 1. NVIDIA GPU connectivity in PowerEdge servers

PowerEdge servers support several different NVLink options:

1. **PowerEdge R760xa server with NVIDIA H100 GPUs and NVLink Bridge**—NVIDIA NVLink is a high-speed point-to-point (P2P) peer transfer connection. An NVLink bridge is a physical component that facilitates the connection between NVLink-capable GPUs. It acts as an interconnect between the GPUs, allowing them to exchange data at extremely high speeds.

   The PowerEdge R760xa server supports four NVIDIA H100 GPUs; NVLink bridge can connect each pair of GPUs. The NVIDIA H100 GPU supports an NVLink bridge connection with a single adjacent NVIDIA H100 GPU. Each of the three attached bridges spans two PCIe slots for a total maximum NVLink Bridge bandwidth of 600 Gbytes per second.

2. **PowerEdge XE8640 server with NVIDIA H100 SXM GPUs and NVLink**—The PowerEdge XE8640 server incorporates four H100 GPUs with NVIDIA SXM5 technology. NVIDIA SXM is a high-bandwidth socket solution for connecting NVIDIA Compute Accelerators to a system.

   The NVIDIA SXM form factor enables multiple GPUs to be tightly interconnected in a server, providing high-bandwidth and low-latency communication between the GPUs. NVIDIA's NVLink technology, which allows for faster data transfers compared to traditional PCIe connections, facilitates this direct GPU-to-GPU communication. The NVLink technology provides a bandwidth of 900 GB/s between any two GPUs.

3. **PowerEdge XE9680 server with NVIDIA H100 GPUs and NVSwitch**—The PowerEdge XE9680 server incorporates eight NVIDIA H100 GPUs with NVIDIA SXM5 technology. The server includes NVIDIA NVSwitch technology, which is a high-performance, fully connected, and scalable switch technology. It is designed to enable ultrafast communication between multiple NVIDIA GPUs. NVIDIA

NVSwitch facilitates high-bandwidth and low-latency data transfers, making it ideal for large-scale AI and high-performance computing (HPC) applications. The NVSwitch technology provides a bandwidth of 900 GB/s between any two GPUs.

GPU memory

During inference, an AI model's parameters are stored in GPU memory. LLMs might require multiple GPU memory units to accommodate their entire neural network structure. In such cases, it is necessary to interconnect the GPUs using NVLink technology to effectively support the model's operations and ensure seamless communication between the GPUs. Therefore, the size of the LLM model required by an enterprise determines both the number of GPUs that are required and the nature of their interconnection. As a result, the choice of PowerEdge server model for the inference infrastructure is contingent on the dimensions of the LLM models.

For models with memory footprints under 160 GB, we recommend using the PowerEdge R760xa server for inference. Most models fall within this category. Notable examples include Llama 2 7B and 13B.

If your model's memory footprint falls between 160GB and 320GB, we recommend the PowerEdge XE8640. The XE8640 features four fully connected GPUs with NVLInk, enabling models to leverage this capability for enhanced performance. While there are no widely recognized community models in this range, customers have the option to deploy their own custom models.

For models with memory footprints exceeding 320 GB, we recommend the PowerEdge XE9860. Models such as Llama 2 70 B and BLOOM 175 B are examples that fall into this category. The following table summarizes examples of LLM models that can be deployed in the PowerEdge servers.

Table 3. Example models supported in PowerEdge servers

Model characteristics	**PowerEdge R760xa with NVIDIA H100 PCIe using NVLink Bridge**	**PowerEdge XE8640 with NVIDIA H100 SXM**	**PowerEdge XE9680 with NVIDIA H100 SXM**
Total memory available	320 GB	320 GB	640 GB
Maximum memory footprint of a model that can run	160 GB	320 GB	640 GB
Example open-source LLMs	NeMo GPT 345M, 1.3B, 2B, 5B and 20B Llama 2 7B and 13B	NeMo GPT 345M, 1.3 B, 2B, 5B and 20B Llama 2 7 B and 13 B	NeMo GPT 345M, 1.3B, 2B, 5B and 20B Llama 2 7B and 13B Llama 2 70 B (available for 8 GPUs) and BLOOM 175B

A single LLM model can support a specific number of concurrent users. Beyond this threshold, the model's latency becomes prohibitively high for practical use. To accommodate a substantial volume of concurrent user requests, enterprises might find it necessary to deploy multiple instances of the same model. This consideration drives the requirement for a specific number of servers to host the LLM model in alignment with the specific use case. For more information, see Sizing guidelines.

For more information about generative AI models that were validated as part of this design, see Table 10.

Note: The preceding table does not consider where a model can span multiple nodes (multiple servers) and if the nodes are interconnected with a high-speed network like InfiniBand.

Inferencing using NVIDIA AI Enterprise

NVIDIA AI Enterprise provides enterprise support for various software frameworks, toolkits, workflows, and models that support inferencing. See the NVIDIA AI Enterprise documentation for more information about all components available with NVIDIA AI Enterprise. The following components incorporated in this validated design are available as part of NVIDIA AI Enterprise:

- Triton Inference Server with FasterTransformer and Model Analyzer
- NVIDIA NeMo models
- NVIDIA Base Command Manager Essentials

The following sections describe the key software components and how they are used in this design. For more information about how these key components work together, see Figure 4 in Software architecture.

Triton Inference Server

NVIDIA Triton Inference Server (also known as Triton) is inference serving software that standardizes AI model deployment and execution and delivers fast and scalable AI in production. Enterprise support for Triton is available through NVIDIA AI Enterprise. It is also available as an open-source software.

Triton streamlines and standardizes AI inference by enabling teams to deploy, run, and scale trained machine learning or deep learning models from any framework on any GPU- or CPU-based infrastructure. It provides AI researchers and data scientists the freedom to choose the appropriate framework for their projects without impacting production deployment. It also helps developers deliver high-performance inference across cloud, on-premises, edge, and embedded devices.

Benefits

The benefits of Triton for AI inferencing include the following:

- **Support for multiple frameworks**—Triton supports all major training and inference frameworks, such as TensorFlow, NVIDIA TensorRT, NVIDIA FasterTransformer, PyTorch, Python, ONNX, RAPIDS cuML, XGBoost, scikit-learn RandomForest, OpenVINO, custom C++, and more.
- **High-performance AI inference**—Triton supports all NVIDIA GPU-, x86-, Arm CPU-, and AWS Inferentia-based inferencing. It offers dynamic batching, concurrent execution, optimal model configuration, model ensemble, and streaming audio/video inputs to maximize throughput and utilization.
- **Designed for DevOps and MLOps**—Triton integrates with Kubernetes for orchestration and scaling, exports Prometheus metrics for monitoring, supports live

model updates, and can be used in all major public cloud AI and Kubernetes platforms. It is also integrated into many MLOps software solutions.

- **Support for model ensembles**—Because most modern inference requires multiple models with preprocessing and postprocessing to be run for a single query, Triton supports model ensembles and pipelines. Triton can run the parts of the ensemble on CPUs or GPUs and allows multiple frameworks inside the ensemble.
- **Enterprise-grade security and API stability**—NVIDIA AI Enterprise includes NVIDIA Triton for production inference, accelerating enterprises to the leading edge of AI with enterprise support, security, and API stability while mitigating the potential risks of open-source software.

Triton Inference Server is at the core of this validated design. It is the software that hosts generative AI models. Triton Inference Server, along with its integration with Model Analyzer, Fast Transformer, and the NeMo framework provides an ideal software for deploying generative AI models.

Faster Transformer

In NLP, the encoder and decoder are crucial components, and the transformer layer has gained popularity as an architecture for both. FasterTransformer from NVIDIA offers a highly optimized transformer layer for both the encoder and decoder, specifically designed for efficient inference.

When running on NVIDIA GPUs, FasterTransformer automatically uses the computational power of Tensor Cores, especially when the data and weights are represented in FP16 precision, enabling faster computations.

FasterTransformer is built using CUDA, cuBLAS, cuBLASLt, and C++. It provides convenient APIs for popular deep learning frameworks like TensorFlow, PyTorch, and Triton backend. These APIs allow users to seamlessly integrate FasterTransformer into their existing workflows using these frameworks.

In this validated design, we use FasterTransformer for inferencing NVIDIA NeMo models.

NVIDIA recently unveiled TensorRT-LLM as a new backend for hosting LLMs. It encompasses the TensorRT deep learning compiler, featuring optimized kernels, preprocessing and postprocessing workflows, and specialized multi-GPU and multinode communication components, all of which contribute to remarkable performance gains on NVIDIA GPUs. TensorRT-LLM empowers developers to explore novel LLMs, providing both peak performance and rapid customization options. For additional insights and in-depth information, see NVIDIA's Technical Blog. At the time of writing, TensorRT-LLM is in an early access phase, and this white paper will be updated when it becomes generally available.

Triton Model Analyzer

The Triton Inference Server offers a powerful solution for deploying AI models. However, each deployment presents its unique challenges, such as meeting latency targets, working with limited hardware resources, and accommodating various model requirements. To address these complexities, the Model Analyzer provides insight for planning and decision making.

The Model Analyzer enables users to send requests to their models while monitoring GPU memory and compute use. This tool provides a mechanism for a deep understanding of an AI model's GPU memory requirements under different batch sizes and instance configurations. Using this information, users can make informed decisions about efficiently combining multiple models on a single GPU, ensuring optimal memory usage without exceeding capacity.

The Model Analyzer is a command-line interface (CLI) that significantly enhances comprehension of Triton Inference Server model compute and memory demands. It provides this awareness by conducting customizable configuration "sweeps" and generating comprehensive reports summarizing performance metrics.

With the Model Analyzer, you can:

- Conduct tailored configuration sweeps to identify the most suitable setup for your specific workload and hardware.
- Obtain detailed reports, metrics, and graphs, summarizing findings related to latency, throughput, GPU resource use, power draw, and more. This information facilitates performance comparison across different configurations.
- Fine-tune model deployments to meet Quality of Service requirements, such as specific latency requirements, GPU memory use, and minimum throughput.

By using the insights provided by the Model Analyzer, you can make better-informed decisions and optimize their AI model deployments for peak performance and efficiency.

In this validated design, we use the Model Analyzer to generate load and monitor the performance of LLM model inference. We used Prometheus and Grafana to gather and visualize the performance metrics. For the results of our validation, which uses Model Analyzer, see Chapter 5. For our sizing guidelines based on the reports of Model Analyzer, see Chapter 6.

NVIDIA NeMo models

NVIDIA supports various generative AI models available that can be used for inferencing or on which to perform model customization of transfer learning.

In this validated design, we used open-source NeMo GPT models to demonstrate inference of LLM models in our solutions. Also, we used the NeMo toolkit available in the NGC Catalog to deploy those models. Our validation efforts were performed with that NVIDIA NeMo Docker container.

NVIDIA Base Command Manager Essentials

NVIDIA Base Command Manager Essentials facilitates seamless operationalization of AI development at scale by providing features like operating system provisioning, firmware upgrades, network and storage configuration, multi-GPU and multinode job scheduling, and system monitoring. It maximizes the use and performance of the underlying hardware architecture.

In this validated design, we use NVIDIA Base Command Manager Essentials for:

- Bare metal provisioning, including deploying the operating system and drivers, and configurating local storage in PowerEdge compute nodes
- Network configuration, including configuring networks for PXE boot, internal node access, POD networking, and storage networking

- Kubernetes deployment, including configuring control plane node and worker nodes, access control and provision of Kubernetes management toolkits and frameworks like Prometheus
- NVIDIA software deployment, including deploying NVIDIA GPU operator and Fabric Manager
- Cluster monitoring and management, including health monitoring, fault tolerance, resource utilization monitoring, software and package management, security and access control, and scaling

Chapter 4 Solution Architecture

Architecture overview

The Dell Validated Design for Generative AI Inferencing is a reference architecture designed to address the challenges of deploying LLMs in production environments. LLMs have shown tremendous potential in natural language processing tasks but require specialized infrastructure for efficient deployment and inferencing.

This reference architecture serves as a blueprint, offering organizations guidelines and best practices to design and implement scalable, efficient, and reliable AI inference systems specifically tailored for generative AI models. While its primary focus is generative AI inferencing, the architecture can be adapted for discriminative or predictive AI models, as explained further in this section.

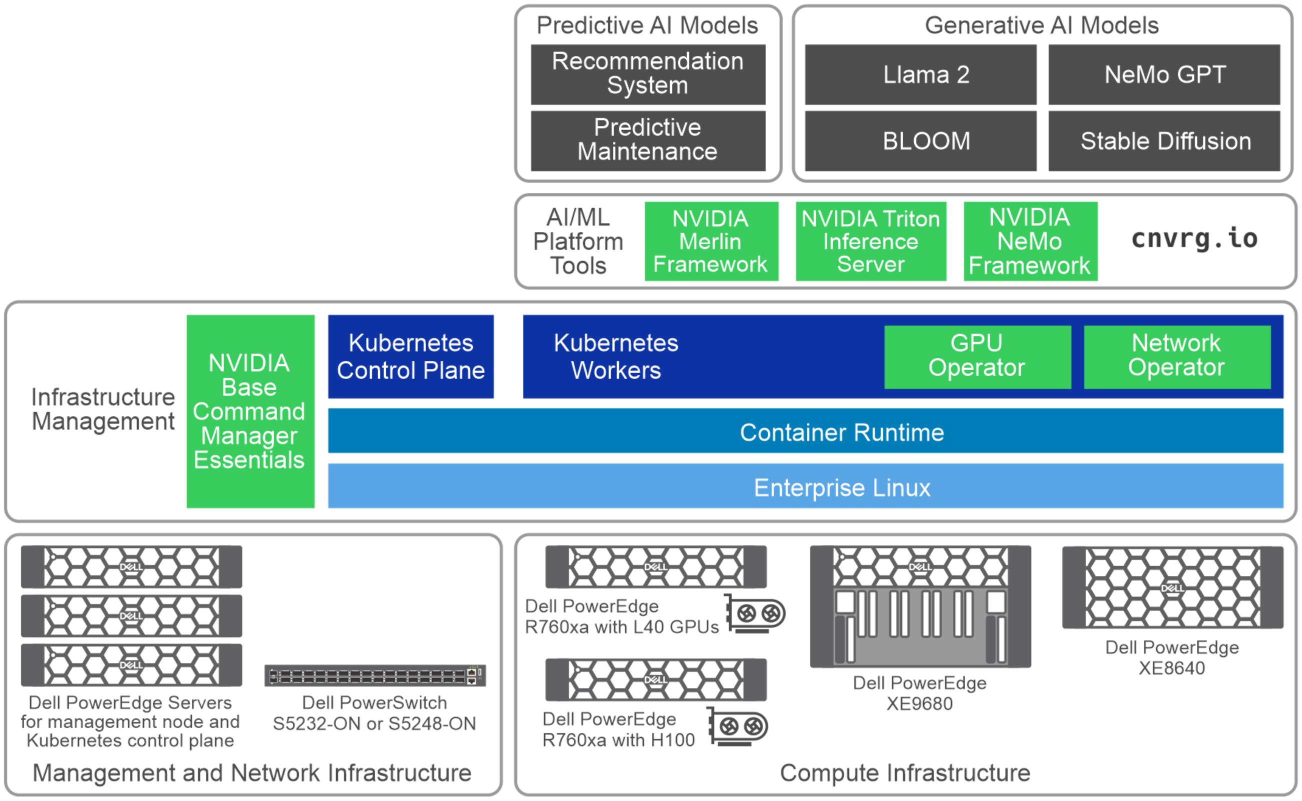


Figure 2. Reference architecture

The following sections describe the key components of the reference architecture.

Compute infrastructure

The compute infrastructure is a critical component of the design, responsible for the efficient execution of AI models. Dell Technologies offers a range of acceleration-optimized servers, equipped with NVIDIA GPUs, to handle the intense compute demands of LLMs. The following server models are available as compute resources for deploying LLM models in production:

- PowerEdge XE9680 server equipped with eight NVIDIA H100 SXM GPUs with NVSwitch
- PowerEdge XE8640 server equipped with four NVIDIA H100 SXM GPUs with NVLink
- PowerEdge R760xa servers supporting up to four NVIDIA H100 PCIe GPUs with NVLink Bridge.

Additionally, the PowerEdge R760xa server configured with NVIDIA L40 GPUs can be used for inferencing predictive AI, such as recommendation systems.

Network infrastructure

Organizations can choose between 25 Gb or 100 Gb networking infrastructure based on their specific requirements. For LLM inferencing tasks using text data, we recommend using existing network infrastructure with 25 Gb Ethernet, which adequately meets text data's bandwidth demands. To future-proof the infrastructure, a 100 Gb Ethernet setup can be used. PowerSwitch S5232F-ON or PowerSwitch S5248F-ON can be used as the network switch. PowerSwitch S5232F-ON supports both 25 Gb and 100 Gb Ethernet, while PowerSwitch S5248F-On is a 25 Gb Ethernet switch. ConnectX-6 Network adapters are used for network connectivity. They are available in both 25 Gb and 100 Gb options.

The scope of the reference architecture only includes AI models in production that can fit in a single PowerEdge server. It does not include models that span multiple nodes and require high-speed interconnect.

Management infrastructure

The management infrastructure ensures the seamless deployment and orchestration of the AI inference system. NVIDIA Base Command Manager Essentials performs bare metal provisioning, cluster deployment, and ongoing management tasks. Deployed on a PowerEdge R660 server that serves as a head node, NVIDIA Base Command Manager Essentials simplifies the administration of the entire cluster.

To enable efficient container orchestration, a Kubernetes cluster is deployed in the compute infrastructure using NVIDIA Base Command Manager Essentials. To ensure high availability and fault tolerance, we recommend that you to install the Kubernetes control plane on three PowerEdge R660 servers. The management node can serve as one of the control plane nodes.

Storage infrastructure

Local storage that is available in PowerEdge servers is used for operating system and container storage. Kubernetes, deployed by NVIDIA Base Command Manager Essentials, deploys the Rancher local path Storage Class that makes local storage available for provisioning pods.

The need for external storage for AI model inference depends on the specific requirements and characteristics of the AI model and the deployment environment. In many cases, external storage is not strictly required for AI model inference, as the models reside in GPU memory. In this validated design, external storage in not explicitly required as part of the architecture.

However, PowerScale storage can be used as a repository for models, model versioning and management, model ensembles, and for storage and archival of inference data, including capture and retention of prompts and outputs when performing inferencing operations. This can be useful for marketing and sales or customer service applications where further analysis of customer interactions may be desirable.

The flexible, robust, and secure storage capabilities of PowerScale offer the scale and speed necessary for operationalizing AI models, providing a foundational component for AI workflow. Its capacity to handle the vast data requirements of AI, combined with its reliability and high performance, cements the crucial role that external storage plays in successfully bringing AI models from conception to application.

Triton Inference Server

The heart of the AI inference system is the NVIDIA Triton Inference Server, which was described in detail earlier and which handles the AI models and processes inference requests. Triton is a powerful inference server software that efficiently serves AI models with low latency and high throughput. Its integration with the compute infrastructure, GPU accelerators, and networking ensures smooth and optimized inferencing operations.

AI tools and frameworks

To enable the deployment of generative AI models in production, the NeMo framework, combined with Triton Inference Server, offers powerful AI tools and optimizations. Specifically, the NeMo framework, combined with FasterTransformer, unlocks state-of-the-art accuracy, low latency, and high throughput inference performance across both single-GPU and multi-GPU configurations. This combination empowers organizations to unleash the full potential of their generative AI models.

The reference architecture can also be used for predictive or discriminative AI. For example, recommendation systems can be built on this reference architecture. Recommendation systems are AI models that analyze user preferences, behaviors, and historical data to suggest relevant and personalized items or content. NVIDIA Merlin is a framework that accelerates and streamlines the development and deployment of large-scale deep learning-based recommendation systems. It aims to optimize the entire recommendation pipeline, from data preprocessing and feature engineering to model training and deployment, with a focus on performance and scalability.

AI models

The reference architecture benefits from the extensive collection of pretrained models provided by the NeMo framework. These models address various categories, including Automatic Speech Recognition (ASR), NLP, and Text-to-Speech (TTS). Additionally, GPT models are readily available for download from locations such as the Hugging Face model repository, offering a vast selection of generative AI capabilities. The specific models that we validated in this design are listed in Chapter 5.

MLOps

Organizations seeking comprehensive model life cycle management can optionally deploy Machine Learning Operations (MLOps) platforms and toolsets, like cnvrg.io, Kubeflow, MLflow, and others.

MLOps integrates machine learning with software engineering for efficient deployment and management of models in real-world applications. In generative AI, MLOps can automate model deployment, ensure continuous integration, monitor performance, optimize resources, handle errors, and ensure security and compliance. It can also manage model versions, detect data drift, and provide model explainability. These practices ensure generative models operate reliably and efficiently in production, which is critical for interactive tasks like content generation and customer service chatbots.

We have validated the MLOps platform from cnvrg.io as part of this validated design. cnvrg.io delivers a full stack MLOps platform that helps simplify continuous training and deployment of AI and ML models. With cnvrg.io, organizations can automate end-to-end ML pipelines at scale and make it easy to place training or inferencing workloads on CPUs and GPUs, based on cost and performance trade-offs. For more information, see the design guide Optimize Machine Learning through MLOPs with Dell Technologies and cnvrg.io.

With all the architectural elements described in this section for the Dell Validated Design for Generative AI Inferencing, organizations can confidently implement high-performance, efficient, and reliable AI inference systems. The architecture's modularity and scalability offer flexibility, making it well suited for various AI workflows, while its primary focus on generative AI inferencing maximizes the potential of advanced LLMs.

Physical architecture

Selecting the appropriate server and network configuration for generative AI inference is crucial to ensure adequate resources are allocated for both management and inference tasks. This section provides example configurations for both management and compute workloads and network architecture.

Management server configuration

PowerEdge R660 head and control plane node

The following table provides the recommended minimum configuration for the management head node and Kubernetes control plan node.

Table 4. PowerEdge R660 head node and Kubernetes control plane configuration

Component	Head node and control plane nodes
Server model	3 x PowerEdge R660
CPU	1 X Intel Xeon Gold 6426Y 2.5 G, 16C/32T
Memory	8 x 16 GB DDR5 4800 MT/s RDIMM
RAID controller	PERC H755 with rear load Brackets
Storage	4 x 960 GB SSD vSAS Read Intensive 12 Gbps 512e 2.5in Hot-Plug, AG Drive SED, 1DWPD
PXE network	Broadcom 5720 Dual Port 1 GbE Optional LOM

Component	Head node and control plane nodes
PXE/K8S network	NVIDIA ConnectX-6 Lx Dual Port 10/25GbE SFP28, OCP NIC 3.0
K8S/Storage network	1 x NVIDIA ConnectX-6 Lx Dual Port 10/25GbE SFP28 Adapter, PCIe (optional)

Consider the following recommendations:

- For the number of nodes, we recommend three PowerEdge servers for management. NVIDIA Base Command Manager Essentials is installed on one of the servers. If the customer requires high availability, NVIDIA Base Command Manager Essentials can be installed on two nodes as an active-passive cluster. Kubernetes control plane is installed on all three nodes. One node acts as both the NVIDIA Base Command Manager head node and the Kubernetes control plane node. We do not recommend a single-node Kubernetes control plane.
- We recommend the same hardware configuration for all three head nodes for ease of configuration and maintenance.
- Because both the head node and control plane node do not require heavy computing, a single-processor server is sufficient.
- For the head node, we recommend opting for a storage-rich configuration, as this configuration will facilitate convenient storage of images and other essential tools. We recommend four SSD drives. Customers can choose more drives or upgrade to NVMe for better performance.

GPU worker node configuration

Dell Technologies provides a selection of three GPU-optimized servers suitable for configuration as worker nodes for Generative AI inference: the PowerEdge R760xa, PowerEdge XE9680, and PowerEdge XE8640 servers. Customers have the flexibility to choose one of these PowerEdge servers based on the specific model size that they require. Larger models, characterized by a greater parameter size, require servers equipped with a higher GPU count and enhanced connectivity. For specific examples of LLM models that can be deployed on each server model, see Table 3.

The GPU-optimized servers act as worker nodes in a Kubernetes cluster. The number of servers depends on the number of models and the number of concurrent requests served by those models. We have validated an eight-GPU worker node cluster. The minimum number of worker nodes in the cluster is one.

PowerEdge R760xa GPU worker node

The following table shows a recommended configuration for a PowerEdge R760xa GPU worker node.

Table 5. PowerEdge R760xa GPU worker node

Component	Details
Server model	PowerEdge R760xa
CPU	2 x Intel Xeon Gold 6438M 2.2G, 32C/64T
Memory	16 x 32 GB DDR5 4800 MT/s RDIMM

Component	Details
Storage	2 x 1.92 TB Enterprise NVMe Read Intensive AG Drive U.2 Gen4 with carrier
PXE Network	Broadcom 5720 Dual Port 1 GbE Optional LOM
K8S/Storage Network	• 1 x NVIDIA ConnectX-6 Lx Dual Port 10/25 GbE SFP28, No Crypto, OCP NIC 3.0 • 1 x NVIDIA ConnectX-6 DX Dual Port 100 GbE QSFP56 Network Adapter (Optional)
GPU	Either: • 2 x or 4 x NVIDIA L40, 48 GB PCIe GPU • 2 x or 4 x NVIDIA Hopper H100, 80 GB, PCIe GPU with NVLink Bridge

PowerEdge XE8640 GPU worker node

The following table shows a recommended configuration for a PowerEdge XE8640 GPU worker node.

Table 6. PowerEdge XE8640 GPU worker node

Component	Details
Server model	PowerEdge XE8640
CPU	2 x Intel Xeon Platinum 8468 2.1G, 48 C/96 T, 16 GT/s
Memory	16 x 32 GB RDIMM, 4800MT/s Dual Rank
Storage	2 x 1.92 TB Enterprise NVMe Read Intensive AG Drive U.2 Gen4 with carrier
PXE Network	Broadcom 5720 Dual Port 1 GbE Optional LOM
K8S/Storage Network	• 1 x NVIDIA ConnectX-6 Lx Dual Port 10/25 GbE SFP28, No Crypto, OCP NIC 3.0 • 1 x NVIDIA ConnectX-6 DX Dual Port 100 GbE QSFP56 Network Adapter (Optional)
GPU	4 x NVIDIA H100 SXM

PowerEdge XE9680 GPU worker node

The following table shows a recommended configuration for a PowerEdge XE9680 GPU worker node.

Table 7. PowerEdge XE9680 GPU worker node

Component	Details
Server model	PowerEdge XE9680
CPU	2 x Intel Xeon Platinum 8470 2G, 52C/104T
Memory	16 x 64 GB RDIMM, 4800 MT/s Dual Rank
Storage	2 x 1.92 TB Enterprise NVMe Read Intensive AG Drive U.2 Gen4 with carrier
PXE Network	Broadcom 5720 Dual Port 1 GbE Optional LOM
K8S/Storage Network	• 1 x NVIDIA ConnectX-6 Lx Dual Port 10/25 GbE SFP28, No Crypto, OCP NIC 3.0 • 1 x NVIDIA ConnectX-6 DX Dual Port 100 GbE QSFP56 Network Adapter (Optional)
GPU	8 x NVIDIA H100 SXM

The CPU memory allocation in the XE9680 GPU worker node configuration exceeds that of the XE8640 configuration. This increase is attributed to the presence of twice as many GPUs, which implies a heightened demand for overall inferencing capacity and, therefore, greater CPU memory requirements.

While inferencing tasks primarily rely on GPUs and do not significantly tax the CPU and memory, it is advisable to equip the system with high-performance CPUs and larger memory capacities. This provisioning ensures sufficient headroom for various data processing activities, machine learning operations, monitoring, and logging tasks. Our goal is to guarantee that the servers boast ample CPU and memory resources for these functions, preventing any potential disruptions to the critical inferencing operations carried out on the GPUs.

Networking design

The following figure shows the network architecture. It shows the network connectivity for compute servers. The figure also shows three PowerEdge head nodes, which incorporate NVIDIA Base Command Manager Essentials and Kubernetes control plane nodes.

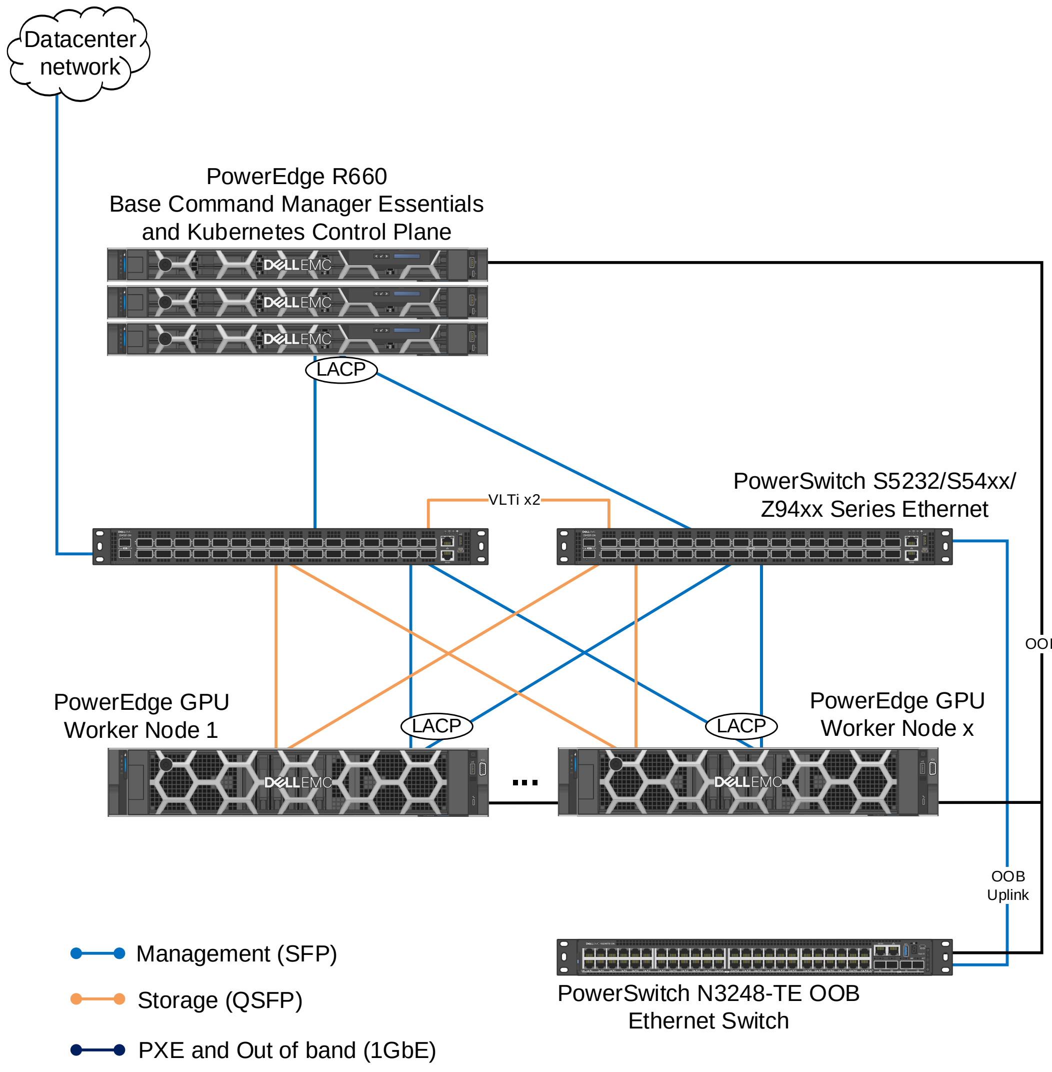


Figure 3. Network architecture

This validated design requires the following networks to manage the cluster and facilitate communication and coordination between different components and nodes within the cluster:

- **Management network**—This network is used for communication between the management server and the cluster nodes. It allows the management server to send commands, configurations, and updates to the nodes. It also enables the nodes to report status, resource usage, and other information back to the management server. This network also serves as the Kubernetes network for internode communication in the cluster. It allows the nodes, Kubernetes pods, and services to exchange data, synchronize tasks, and collaborate efficiently during cluster operations.

- **External/data center network**—The external network connects the cluster to the Internet, allowing the cluster nodes to communicate with external systems, services, and the Internet. This network is essential for accessing external resources, downloading software updates, and interacting with users or applications outside the cluster.
- **Storage network**—In some configurations, a dedicated storage network might be used to facilitate data transfer between the cluster nodes and storage devices. This network helps to optimize data access and reduce latency for storage operations.
- **OOB and PXE network**—The out-of-band (OOB) network is a separate and dedicated network infrastructure used for managing and monitoring servers. It is a 1Gb Ethernet network that connects to the Integrated Dell Remote Access Controller (iDRAC) of the PowerEdge servers in the cluster. This network is also used for PXE to automate the provisioning and deployment of operating systems.

Software architecture

In our validated design, we use NVIDIA Base Command Manager Essentials for bare metal provisioning, in which the operating system (through PXE), drivers, and local storage configuration are deployed in PowerEdge compute nodes. NVIDIA Base Command Manager Essentials deploys Kubernetes, configuring control plane and worker nodes, access control, and provisions essential Kubernetes management toolkits and frameworks such as Prometheus. NVIDIA Base Command Manager Essentials also handles network configuration, including cluster networking, POD networking, and storage networking. Finally, NVIDIA Base Command Manager Essentials is used to deploy NVIDIA software, including the NVIDIA GPU operator and Fabric Manager, essential components for optimizing GPU performance and management.

When the infrastructure is ready to deploy AI models in production, the following workflow explains the software architecture of an AI model in production.

Inference workflow for performance optimization

AI models are typically optimized for performance before deployment to production. Optimized models offer faster inference speed, improved resource efficiency, and reduced latency, resulting in cost savings and better scalability. They can handle increased workloads, require fewer computational resources, and provide a better user experience with quick responses.

The following figure shows an example LLM model optimization and deployment using the software components described in the reference architecture:

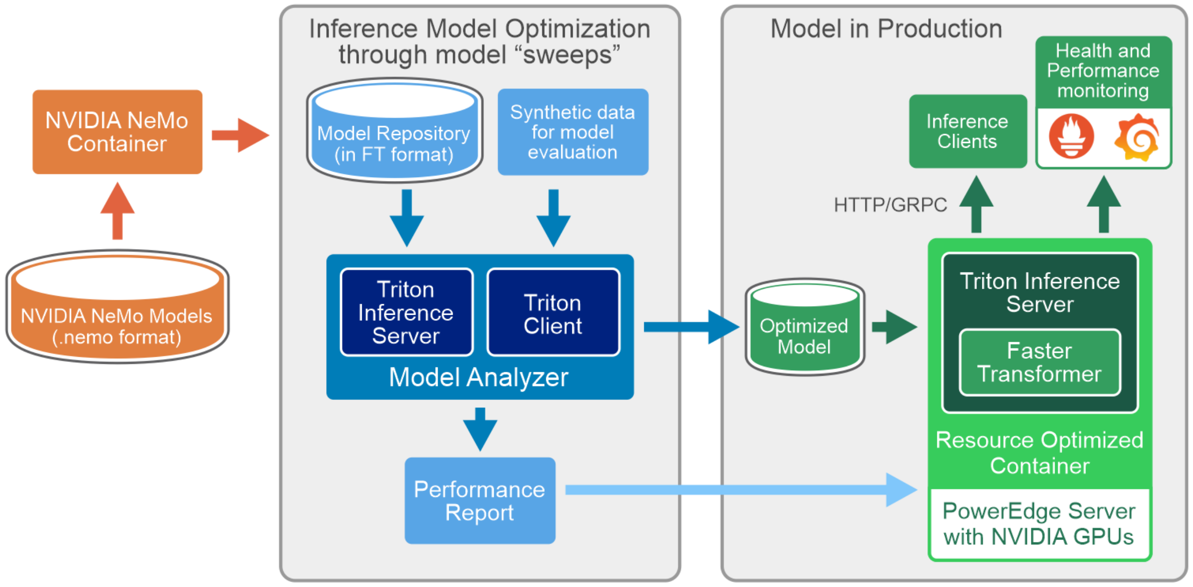

NVIDIA NeMo Container
NVIDIA NeMo Models (.nemo format)
Inference Model Optimization through model "sweeps"
Model Repository (in FT format)
Synthetic data for model evaluation
Triton Inference Server
Triton Client
Model Analyzer
Performance Report
Model in Production
Optimized Model
HTTP/GRPC
Inference Clients
Health and Performance monitoring
Triton Inference Server
Faster Transformer
Resource Optimized Container
PowerEdge Server with NVIDIA GPUs

Figure 4. Workflow to optimize NeMo LLM models for inferencing using NVIDIA toolkits

NVIDIA offers several pretrained models in NeMo format. To optimize the model's throughput and latency, it can be converted to the FasterTransformer format, which includes performance modifications to the encoder and decoder layers in the transformer architecture. FasterTransformer enables the model to serve inference requests with three times quicker latencies or more compared to their non- FasterTransformer counterparts. The NeMo framework training container includes the FasterTransformer framework and scripts for converting a `.nemo` file to the FasterTransformer format.

Note: NeMo models available in Hugging Face are foundation models. While these models offer a strong starting point with general capabilities, they are typically customized for specific use. This design guide does not address or show model customization, though that will be covered in a subsequent guide.

When the model is converted, it can be optimized using the Model Analyzer tool. Model Analyzer helps gain insights into the compute and memory requirements of Triton Inference Server models by analyzing various configuration settings and generating performance reports. These reports summarize metrics like latency, throughput, GPU resource utilization, power draw, and more, enabling easy comparison of performance across different setups and identifying the optimal configuration for the inference container.

The final optimized model is ready for production deployment on a PowerEdge server equipped with NVIDIA GPUs using Triton Inference Server. It can be accessed through an API endpoint or using HTTPS/GPRC protocols. Triton Inference Server also offers health and performance metrics of the model in production, which can be consumed and visualized through Prometheus and Grafana. These monitoring tools provide valuable insights into the model's performance and overall system health during deployment.

The same workflow is applicable for predictive or discriminative AI. The starting point for a recommendation system use case is a model developed using the Merlin framework (as opposed to a NeMo model).

Chapter 5 Validation Results

The Dell Validated Design for Generative AI Inferencing aims to simplify and accelerate the deployment of complex infrastructure for generative AI by providing validated and proven architectures. It helps customers by reducing the guesswork and potential risks associated with designing and implementing initial custom solutions.

We validated multiple models on our reference architecture to ensure the Dell and NVIDIA hardware and software are optimized and integrated to deliver reliable and high-performance solutions. We validated both NeMo and open-source models.

Generative AI model validation

System configurations

The following tables list the system configurations and software stack used for generative AI validation:

Table 8. System configuration

Component	Details		
Compute server for inferencing	4 x PowerEdge R760xa	2 x PowerEdge XE9680	2 x PowerEdge XE8640
GPUs per server	4 x NVIDIA H100 PCIe GPUs	8 x NVIDIA H100 SXM GPUs	4 x NVIDIA H100 SXM GPUs
Network adapter	ConnectX6 25 Gb Ethernet	ConnectX6 25 Gb Ethernet	ConnectX6 25 Gb Ethernet
Network switch	2 x PowerSwitch S5248F-ON	2 x PowerSwitch S5248F-ON	2 x PowerSwitch S5248F-ON

Table 9. Software components and versions

Component	Details
Operating system	Ubuntu 22.04.1 LTS
Cluster management	NVIDIA Base Command Manager Essentials 9.2
Kubernetes	Upstream Kubernetes - Version v1.24.9
GPU operator	NVIDIA GPU operator v22.9.2
Inference server	NVIDIA Triton Inference Server v23.04
AI framework	NVIDIA NeMo Container v23.04

Models used for validation

GPT models are advanced language models known for their impressive text generation and natural language processing capabilities. They use the Transformer architecture, which allows them to understand complex language patterns and relationships. These models are pretrained on vast amounts of text data and can be fine-tuned for specific tasks, enabling them to perform well in various NLP applications.

In this validated design, we validated the generative AI models shown in table 10 below. We validated a significant number of the NeMo GPT and Stable Diffusion models with Triton Inference Server on all three server models. We ran inference on BLOOM 7B, Llama 2 7B, and Llama 213B on standard Python or PyTorch containers available from NVIDIA NGC on all three server models.

The Llama 2 70B model in a standard configuration for inference uses model parallelism, where groups of model layers are spread across multiple GPUs. This model uses a model-parallel (MP) parameter of 8, which requires eight GPUs. Therefore, we deployed this model in its standard configuration on a single PowerEdge XE9680 server that offers eight H100 GPUs connected by NVSwitch.

Table 10. Generative AI models and validation container

Model	Description	Base Container used for validation
NeMo GPT 20B	Transformer-based language model with 20 B total trainable parameter count	Triton - nvcr.io/ea-bignlp/bignlp-inference:22.08-py3 NeMo - nvcr.io/nvidia/nemo:23.04
NeMo GPT-2B-001	Transformer-based language model with 2 B total trainable parameter count	Triton - nvcr.io/ea-bignlp/bignlp-inference:22.08-py3 NeMo - nvcr.io/nvidia/nemo:23.04
NeMo GPT 1.3B	Transformer-based language model with 1.3 B total trainable parameter count	Triton - nvcr.io/ea-bignlp/bignlp-inference:22.08-py3 NeMo - nvcr.io/nvidia/nemo:23.04
NeMo GPT 345M	Transformer-based language model with 345 M total trainable parameter count	Triton - nvcr.io/ea-bignlp/bignlp-inference:22.08-py3 NeMo - nvcr.io/nvidia/nemo:23.04
NeMo GPT 5B	Transformer-based language model with 5 B total trainable parameter count	Triton - nvcr.io/ea-bignlp/bignlp-inference:22.08-py3 NeMo - nvcr.io/nvidia/nemo:23.04
BLOOM 7B	Big Science Large Open-science Open-access Multilingual Language Model with 7B parameter count	nvcr.io/nvidia/cuda:12.1.0-devel-ubi8
Llama 2 7B	Optimized transformer model with 7B parameter count	nvcr.io/nvidia/pytorch:23.06-py3
Llama 2 13B	Optimized transformer model with 13B parameter count	nvcr.io/nvidia/pytorch:23.06-py3
Llama 2 70B	Optimized transformer model with 70B parameter count	nvcr.io/nvidia/pytorch:23.06-py3

Model	Description	Base Container used for validation
Stable Diffusion 2.0	Text to image generation model	Triton - nvcr.io/ea-bignlp/bignlp-inference:22.08-py3 python:latest

Validation scenarios

We deployed and validated the preceding models in the following scenarios:

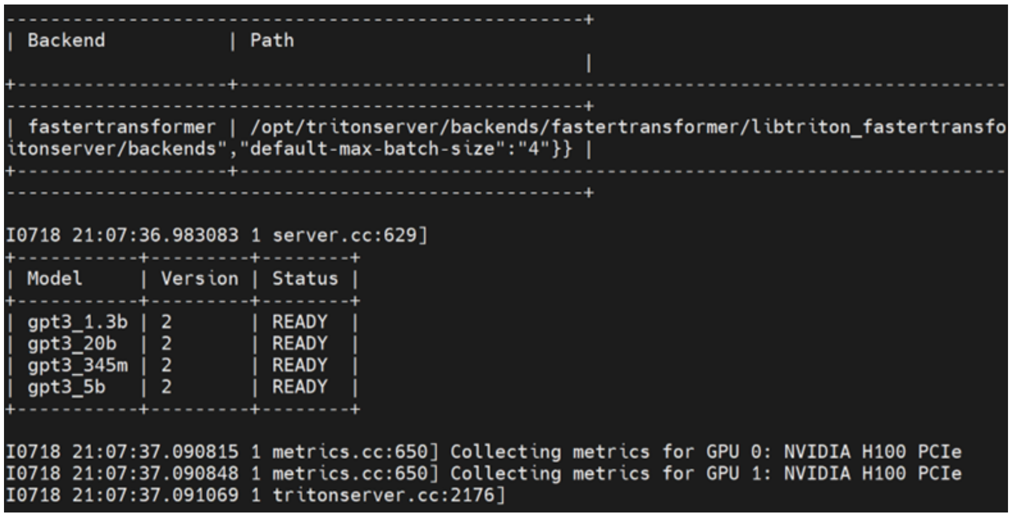
- NeMo and Stable Diffusion model deployment using Triton Inference Server
- BLOOM and Llama deployment using the Hugging Face framework and on standard Python or PyTorch containers available from NVIDIA NGC.
- Model validation using a Text Classification and AI21 RTE

Not all the models are validated in all the different scenarios due to limitations noted in the following sections.

NeMo model deployment using Triton Inference Server

To optimize the NeMo model's throughput and latency, it can be converted to the FasterTransformer (FT) format, which includes performance modifications to the encoder and decoder layers in the transformer architecture. This conversion is done by launching the NeMo Docker container with the command specified in this blog post. We deployed all the NeMo models listed in Table 6 with Triton Inference Server, except for the NeMo GPT-2B-00 1 model, which we deployed using the inference server described in the model card.

The following figure shows the models in production using Triton Inference Server:



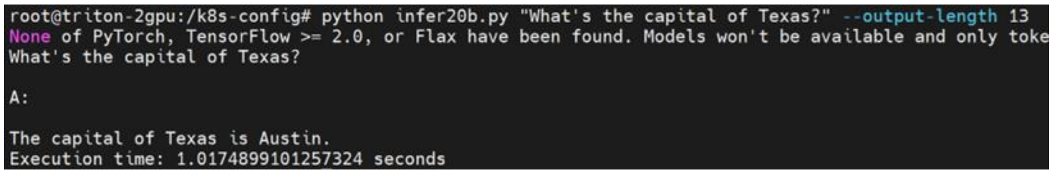
```
----------------------------------------------------+
| Backend            | Path
                                                    |
+-------------------+-------------------------------------------------------------------
----------------------------------------------------+
| fastertransformer | /opt/tritonserver/backends/fastertransformer/libtriton_fastertransfo
itonserver/backends","default-max-batch-size":"4"}} |
+-------------------+-------------------------------------------------------------------
----------------------------------------------------+

I0718 21:07:36.983083 1 server.cc:629]
+-----------+---------+--------+
| Model     | Version | Status |
+-----------+---------+--------+
| gpt3_1.3b | 2       | READY  |
| gpt3_20b  | 2       | READY  |
| gpt3_345m | 2       | READY  |
| gpt3_5b   | 2       | READY  |
+-----------+---------+--------+

I0718 21:07:37.090815 1 metrics.cc:650] Collecting metrics for GPU 0: NVIDIA H100 PCIe
I0718 21:07:37.090848 1 metrics.cc:650] Collecting metrics for GPU 1: NVIDIA H100 PCIe
I0718 21:07:37.091069 1 tritonserver.cc:2176]
```

Figure 5. Models in production using Triton Inference Server

We validated the model by asking simple questions, as shown in the following figure:



```
root@triton-2gpu:/k8s-config# python infer20b.py "What's the capital of Texas?" --output-length 13
None of PyTorch, TensorFlow >= 2.0, or Flax have been found. Models won't be available and only toke
What's the capital of Texas?

A:

The capital of Texas is Austin.
Execution time: 1.0174899101257324 seconds
```

Figure 6. Prompt and response with a NeMo GPT model

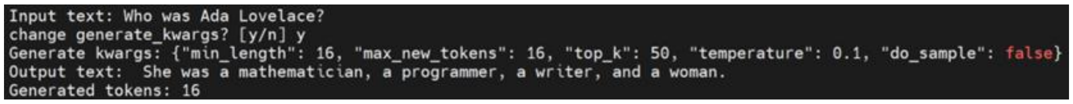
BLOOM model deployment

The BLOOM model has been proposed with its various versions through the BigScience Workshop. BigScience is inspired by other open science initiatives in which researchers have pooled their time and resources to collectively achieve a higher impact. The BLOOM architecture is similar to GPT3 (autoregressive model for next token prediction) but has been trained on 46 different languages and 13 programming languages. Several smaller versions of the models have been trained on the same dataset.

We deployed the BLOOM 7B model on a PowerEdge R760xa server using instructions available in the GitHub page. We used the following latest libraries so that the model can be deployed on NVIDIA H100 GPUs:

- torch 2.0.0+cu118
- transformers 4.27.4
- deepspeed 0.8.3
- accelerate 0.18.0
- gunicorn 20.1.0
- fastapi 0.89.1
- uvicorn 0.19.0
- jinja2 3.1.2
- pydantic 1.10.2
- huggingface_hub 0.12.1
- grpcio-tools 1.50

The following figure shows the model inference:



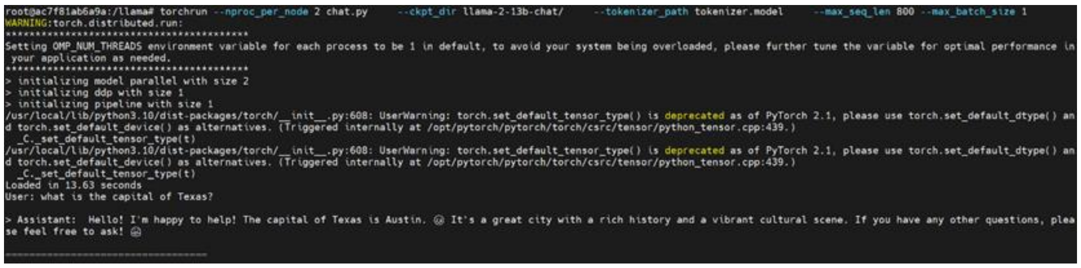
```
Input text: Who was Ada Lovelace?
change generate_kwargs? [y/n] y
Generate kwargs: {"min_length": 16, "max_new_tokens": 16, "top_k": 50, "temperature": 0.1, "do_sample": false}
Output text:  She was a mathematician, a programmer, a writer, and a woman.
Generated tokens: 16
```

Figure 7. Model inference using BLOOM

Llama 2B and 13B model deployment

Llama 2 is a collection of pretrained and fine-tuned generative text models ranging in scale from 7 billion to 70 billion parameters. Llama 2 is available from Hugging Face and for accessing the repository it is required to accept Meta license agreement. We deployed the model on a Pytorch:23.06 container from NVIDIA NGC after installing the packages required by Llama and listed in its `requirements.txt` file.

The following figure is an example of running inference of the Llama model:



```
root@ac7f81ab6a9a:/llama# torchrun --nproc_per_node 2 chat.py     --ckpt_dir llama-2-13b-chat/     --tokenizer_path tokenizer.model     --max_seq_len 800 --max_batch_size 1
WARNING:torch.distributed.run:
*****************************************
Setting OMP_NUM_THREADS environment variable for each process to be 1 in default, to avoid your system being overloaded, please further tune the variable for optimal performance in
 your application as needed.
*****************************************
> initializing model parallel with size 2
> initializing ddp with size 1
> initializing pipeline with size 1
/usr/local/lib/python3.10/dist-packages/torch/__init__.py:608: UserWarning: torch.set_default_tensor_type() is deprecated as of PyTorch 2.1, please use torch.set_default_dtype() an
d torch.set_default_device() as alternatives. (Triggered internally at /opt/pytorch/pytorch/torch/csrc/tensor/python_tensor.cpp:439.)
  _C._set_default_tensor_type(t)
/usr/local/lib/python3.10/dist-packages/torch/__init__.py:608: UserWarning: torch.set_default_tensor_type() is deprecated as of PyTorch 2.1, please use torch.set_default_dtype() an
d torch.set_default_device() as alternatives. (Triggered internally at /opt/pytorch/pytorch/torch/csrc/tensor/python_tensor.cpp:439.)
  _C._set_default_tensor_type(t)
Loaded in 13.63 seconds
User: what is the capital of Texas?

> Assistant:  Hello! I'm happy to help! The capital of Texas is Austin. 😊 It's a great city with a rich history and a vibrant cultural scene. If you have any other questions, plea
se feel free to ask! 😊

==================================
```

Figure 8. Model inference for Llama

Stable Diffusion

Stable Diffusion is a deep learning model that performs text-to-image tasks with a focus on generating detailed images based on text descriptions. Additionally, this versatile model can be extended to handle various tasks, including inpainting, outpainting, and image-to-image translations guided by textual prompts.

We followed the instructions in GitHub to download and run the model. We used the following Docker file.

```
FROM python:latest
RUN rm -rf /usr/local/cuda/lib64/stubs
COPY requirements.txt /
RUN pip install -r requirements.txt \
  --extra-index-url https://download.pytorch.org/whl/cu118

RUN useradd -m huggingface
USER huggingface
WORKDIR /home/huggingface
ENV USE_TORCH=1
RUN mkdir -p /home/huggingface/.cache/huggingface \
  && mkdir -p /home/huggingface/input \
  && mkdir -p /home/huggingface/output
COPY docker-entrypoint.py /usr/local/bin
COPY token.txt /home/huggingface
ENTRYPOINT [ "docker-entrypoint.py" ]
```

We used the following requirements.txt file:

```
diffusers[torch]==0.17.1
onnxruntime==1.15.1
safetensors==0.3.1
torch==2.0.1+cu118
transformers==4.30.1
xformers==0.0.20
```

The following figure is an image generated using the prompt `Astronaut riding a horse on Mars`:

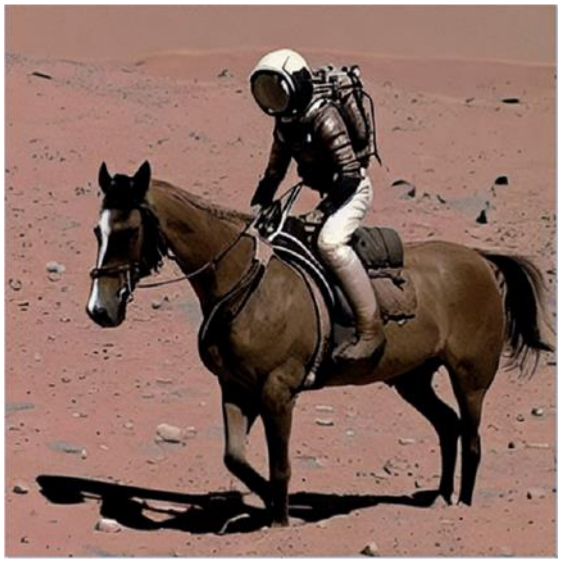


Figure 9. Image generated by Stable Diffusion for prompt "Astronaut riding a horse on Mars"

Model validation using AI21 RTE

We used Language Model Evaluate Test Suite from AI21 Labs, specifically the RTE suite, to validate NeMo model inference. The goal is to reproduce the results made available in the NeMo Model Card. RTE includes datasets that prompt for a True or False response.

The following text is an example of a prompt and the response:

Prompt: Tropical Storm Irene on August 11, 2005 at 16:15 UTC. Tropical Storm Irene will increase in strength over the next several days, possibly developing into a hurricane that will hit the east coast of the United States, said the National Hurricane Center of Miami, Florida in a report today. Irene was located approximately 975 kilometers south-southeast of Bermuda at 16:00 UTC today. Forecasters say that the storm is now moving in a west-northwest direction with top sustained winds of 40 miles per hour. A storm called Irene is going to approach the east coast of the US. True or False?

Response: True

The following table shows the scores that we measured. Comparing this score to other models provides a mechanism to compare the accuracy of the models.

Table 11. RTE score

Model	RTE score
NeMo GPT 20B	0.527076
NeMo GPT-2B-001	0.519856

Model validation using text classification

We validated the NeMo model for binary text classification using the Stanford Politeness Corpus dataset, which has text phrases labeled as polite or impolite. We evaluate models using a fixed test dataset containing 700 phrases. The dataset requests sentiment analysis from the LLM models.

The following text is an example of a prompt and the response:

Prompt: What do you mean? How am I supposed to vindicate myself of this ridiculous accusation? Question: polite or impolite?

Response: Impolite

The following table summarizes the score we obtained for the NeMo models. An explanation of the performance metrics can be found here.

Table 12. Score summary

Model	Label	Precision	Recall	F1
NeMo GPT-345M	Polite	95.36	93.51	94.43
	Impolite	93.29	95.21	94.24
NeMo GPT-2B-001	Polite	94.67	92.21	93.42
	Impolite	92	94.52	93.24
NeMo GPT 20B	Polite	94.08	92.86	93.46
	Impolite	92.57	93.84	93.2

Measuring the accuracy of LLMs is vital for assessing their performance, comparing different models, and tracking progress in the field of NLP. It will guide users in determining which foundation model to use for further customization and fine-tuning. Additionally, accurate evaluation metrics validate research findings, aid in model optimization, and address potential biases or ethical concerns. Ultimately, accurate assessment ensures informed and responsible deployment of LLMs in diverse real-world applications.

Every day, newer and better-performing LLM models are being released. Furthermore, the evaluation metrics for LLMs have evolved over time to tackle the unique challenges posed by these intricate models and to assess their performance more effectively across a wide spectrum of natural language processing tasks. Our validation primarily focused on publicly available NeMo models. In the future, we will update this section to incorporate recently released models using metrics and tools that are widely recognized and accepted in the field. We will also update the section to demonstrate deployment of other open-source available models from HuggingFace using Triton Inference Server.

Chapter 6 Performance Characterization and Sizing

Overview

Sizing the infrastructure for LLM inference is essential due to the computational demands, high memory requirements, and unique characteristics of these models. Properly sizing the infrastructure ensures efficient handling of LLMs' massive parameter count during inference, avoiding out-of-memory errors and achieving low latency. It enables resource efficiency, scalability, and cost optimization by accommodating varying workloads and selecting optimal configurations. Overall, proper infrastructure sizing for LLM inference ensures optimal performance, scalability, and user experience while managing operational costs effectively and removing any resource bottlenecks.

To characterize various GPT models and gain insights into their inference performance, our goal is to provide sizing guidelines. Consider several critical factors when sizing an infrastructure for LLM model inference:

1. **Model size in parameters**—The number of parameters in the LLM directly impacts its memory footprint and computational requirements. Larger models with more parameters might require more powerful hardware for efficient inference.
2. **Latency**—Low latency is crucial for real-time applications and user interactions. Properly sizing the infrastructure helps meet latency targets and ensures fast response times.
3. **Number of concurrent requests**—Understanding the expected number of concurrent inference requests is essential for determining the required computational resources and ensuring smooth and responsive inference under varying workloads.
4. **GPU compute and memory use**—For GPU-accelerated inference, considering both compute and memory use are critical to optimizing resource allocation and ensuring efficient GPU use.

Performance test results

We conducted performance characterization of the following NeMo models: 345M, 1.3B, 5B, and 20B on a PowerEdge R760xa server with four H100 GPUs, with NVLink Bridge being used to connect each GPU to one other GPU to create two GPU pairs. We used the Model Analyzer tool, which allowed us to run model sweeps using synthetic datasets. During these sweeps, Model Analyzer performed inference against each NeMo model while varying the number of concurrent requests, enabling us to measure response latencies and gather essential GPU metrics. All NeMo models were run on a single GPU, and Model Analyzer generated comprehensive reports for each of these sweeps.

Model GPU memory consumption

Because GPU memory is a primary bottleneck in the efficient inference of LLM models, it was crucial that we analyzed the model memory consumption. Understanding the GPU memory consumed by various LLM models is pivotal for determining the appropriate resource allocation in production environments. By accurately assessing GPU memory use, we can ensure that the infrastructure is properly sized, avoiding out-of-memory errors and optimizing the overall performance of LLM inference.

Table 13. Model parameter and GPU memory consumption

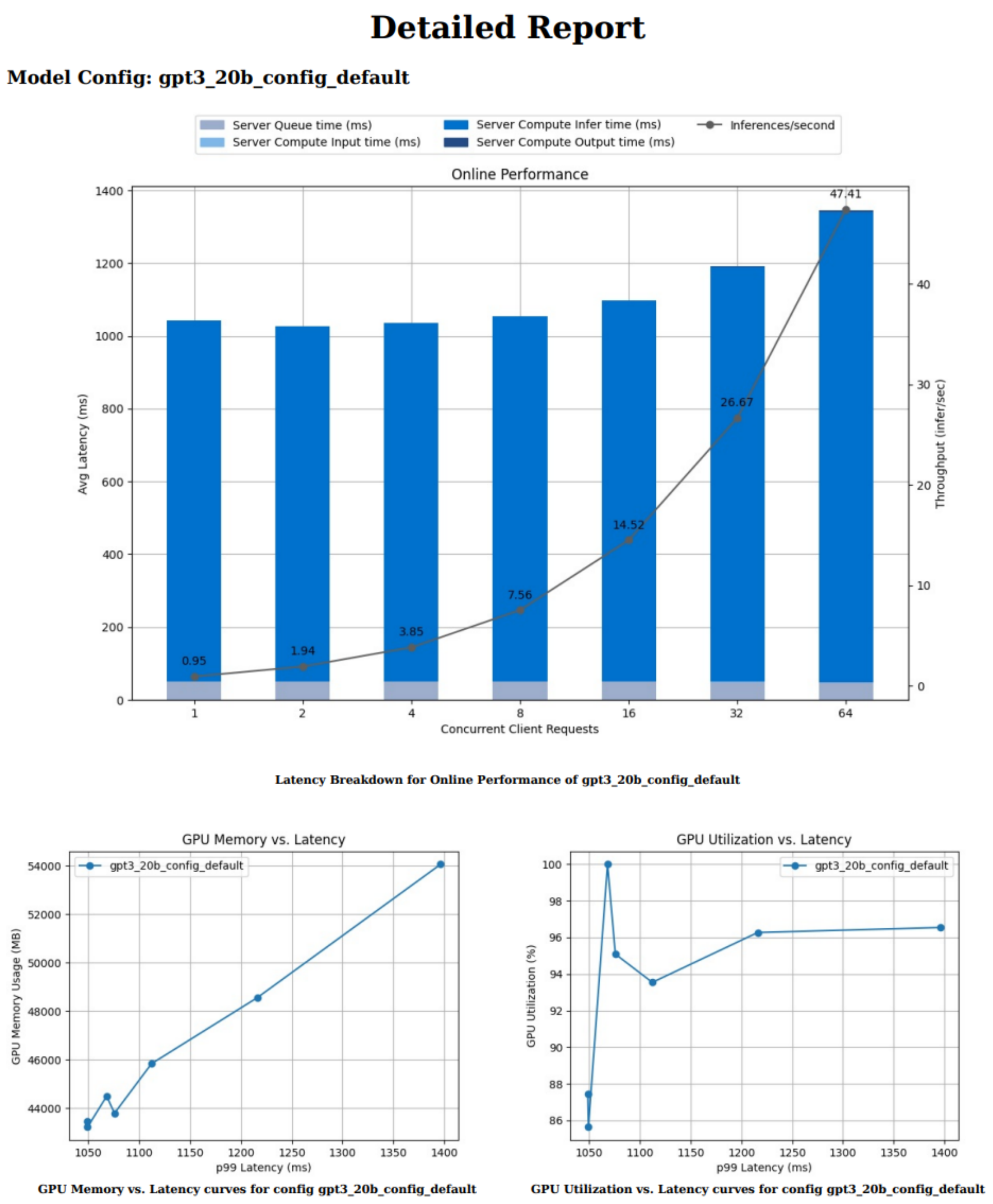
Model	GPU memory consumed (for one concurrent request)	Precision
NeMo GPT 345M	3.1 GB	BF16
NeMo GPT 1.3B	4.1 GB	BF16
NeMo GPT 5B	12.4 GB	BF16
NeMo GPT 20B	42.4 GB	BF16
Bloom 7B	16.6 GB	BF16
Stable Diffusion 2.0	7.9 GB	FP16
Llama-2-7B	14.2 GB	FP16
Llama-2-13B	29.5 GB	FP16
Llama-2-70B	154.7 GB	FP16

As a rule, we can estimate that an LLM model consumes approximately 2.1GB of memory for every 1B parameter (at BF16 precision). However, there are exceptions to this rule, such as smaller models like NeMo 345M. Additionally, non-LLM models like Stable Diffusion have their own memory requirements and might not follow the same pattern. Note that memory consumed by a model might increase slightly as the number of concurrent requests handled by the model increases.

FP16 (half precision) and BF16 (bfloat16) are two different floating-point formats used in deep learning and high-performance computing to represent numerical values with reduced precision. Both formats offer benefits in terms of memory usage, computation speed, and energy efficiency. BF16 is typically preferred in deep learning training and inference on specialized hardware like tensor processing units.

Resource use

Using the Model Analyzer report, we can visualize how the latency of the requests changes with the concurrent client requests. The following figure shows a graph produced by Model Analyzer for the 20B model:



Detailed Report

Model Config: gpt3_20b_config_default

Latency Breakdown for Online Performance of gpt3_20b_config_default

GPU Memory vs. Latency curves for config gpt3_20b_config_default

GPU Utilization vs. Latency curves for config gpt3_20b_config_default

Figure 10. Model Analyzer report for 20B model

For every model, we identified the optimal number of concurrent requests that minimally impact latency while effectively using available resources. For example, using the preceding graph, we concluded that a single instance 20B model can support a concurrency of 16. This information allows us to achieve the right balance between computational efficiency and responsiveness for each of the NeMo models.

The following table summarizes the resource use, latency, and concurrent request metrics for each model for their optimal number of concurrent requests:

Table 14. Resource use and model metrics for the optimal number of concurrent requests

Metric	NeMo GPT 345m	NeMo GPT 1.3B	NeMo GPT 5B	NeMo GPT 20B
Request concurrency	4	4	4	16
p95 Latency (ms)	295.85	406.69	1135.52	1112.39
Client Response Wait (ms)	275.34	396.45	1101.13	1099.72
Server Queue (ms)	206.29	297.04	824.52	49.58
Throughput (inference/sec)	14.5	10.07	3.61	14.51
Max server memory Usage (GB)	4.68	4.35	4.68	4.59
Max GPU Memory Usage (GB)	3.1	4.12	12.41	44.76
Avg GPU Utilization (%)	98	99	99	93.5

Sizing guidelines

To illustrate the process of sizing the compute infrastructure, let's examine a specific scenario. In this scenario, our goal is to deploy a NeMo GPT 5B model for 32 concurrent users and a NeMo GPT 20B model for 64 concurrent users. We will guide you through the steps to determine the infrastructure requirements for this example.

- **Identify the Number of Model Instances:** First, we need to ascertain the number of model instances required to meet these user concurrency targets. Based on the model analyzer reports, we've determined that the optimal concurrency levels for these models are four for the 5B model and 16 for the 20B model. We've chosen these values because exceeding these levels significantly impacts latency.

- **Calculate Model Instances:** Next, we calculate the number of instances needed for each model to fulfill the concurrent user requirements. Through straightforward calculations, we find that we require eight instances of the 5B model and four instances of the 20B model. It's important to note that latency requirements also play a role in determining the required number of model instances. For this example, we assume that the observed latency meets the criteria.

- **Determine GPU Requirements:** After establishing the number of instances, we calculate the total number of GPUs necessary to deploy these models. As outlined in Table 14, GPU utilization hovers around 100 percent for the considered concurrency levels. Consequently, each model instance requires one GPU for hosting. Therefore, to accommodate eight instances of the 5B model and four instances of the 20B model, we need a total of 12 GPUs.

- **Server Selection:** To support this scenario, we require three PowerEdge R760xa servers to that is configured with 4 GPUs each to meet the demands of our deployment.

By following these steps, you can effectively size the compute infrastructure to cater to your specific user concurrency and model requirements.

The following table summarizes the server requirements for the example scenario that we described. We have added one additional scenario for PowerEdge XE9680:

Table 15. Sizing examples

Requirements and sizing	Example scenario with a PowerEdge R760xa or PowerEdge XE8640 server	Example scenario with a PowerEdge XE9680 server
Models and concurrent users	• NeMo GPT 5B model for 32 concurrent users • NeMo GPT 20B model for 64 concurrent users	• NeMo GPT 1.3B model for 32 concurrent users • NeMo GPT 5B model for 64 concurrent users • NeMo GPT 20B model for 128 concurrent users
Total models needed to support required concurrency	• 8 x NeMo GPT 5B models • 4 x NeMo GPT 20B models	• 8 x NeMo GPT 1.3B models • 16 x NeMo GPT 5B models • 8 x NeMo GPT 20B models
Total PowerEdge R760xa required	Three PowerEdge R760xa or PowerEdge XE8640 servers, each with 4 x H100 GPUs	Four PowerEdge XE9680 servers, each with 8 x H100 GPUs

These calculations are based on the FasterTransformer backend. NVIDIA recently announced early access for TensorRT-LLM, which includes several performance optimizations. Our sizing guidelines will be updated when TensorRT-LLM is generally available.

Chapter 7 Inferencing Operations

Model deployment

NVIDIA Triton Inference Server is an open-source high-performance deep learning inference server developed by NVIDIA. It is designed to serve AI models in production environments, enabling scalable and efficient deployment of machine learning models for real-time inferencing. Triton supports various deep learning frameworks such as TensorFlow, PyTorch, ONNX, and more. It enables you to serve multiple models simultaneously, making it flexible for deploying different AI applications. Triton is built for scalability, allowing you to deploy and serve models on multiple GPUs and across multiple nodes in a distributed manner. This scalability makes it suitable for handling large-scale inference workloads.

Life cycle

NVIDIA Triton Inference Server operates by serving models from one or more specified model repositories during server startup. Using this repository enables users to manage different model versions effectively, similar to version control systems, providing the life cycle for AI models in production. Each model version can be maintained as a distinct entity in the repository, facilitating retrieval of the wanted version when necessary. Moreover, some users employ external configuration management tools like Ansible to streamline the management of model versions. Such tools offer automation capabilities, making it easier to switch between different model versions seamlessly in Triton.

Backup and restore

AI models can be backed up to ensure their preservation and availability if there is data loss, system failures, or accidental changes. NVIDIA Triton Inference Server does not natively support backup and restore functionalities for model data and inference state. Triton Inference Server primarily focuses on efficiently serving AI models for inference and does not include integrated mechanisms for data backup and restoration. Models are typically stored in a repository in Persistent Volumes offered by Kubernetes. Customers can take advantage of backup solutions for Kubernetes to backup and restore AI models and inference data.

Secure model serving

Securing AI models in production is crucial to protect sensitive data, maintain system integrity, and prevent unauthorized access. Important security considerations for AI models in production include:

- **Secure model serving**—When deploying the AI model, employ secure containerization and sandboxing techniques to isolate the model from the underlying infrastructure and prevent potential attacks. Docker and Kubernetes provide these mechanisms.

- **Authentication and authorization**—Implement strong authentication mechanisms to verify the identity of users and applications accessing the AI model. Use access control lists (ACLs) or role-based access control (RBAC) to enforce appropriate permissions for different user roles. Kubernetes deployed through NVIDIA Base Command Manager Essentials supports both ACL and RBAC to provide fine-grained authorization and control over GPU resources and AI models in the cluster. These mechanisms help enforce security policies, and ensure that only authorized users and processes have access to perform inference operations on Kubernetes resources.
- **Secure APIs**—Triton Inference Server exposes the AI models through APIs. These APIs can be secured using authentication and authorization mechanisms. Use tokens or API keys to validate requests and prevent unauthorized access.
- **Secure Communications**—Triton Inference Server supports secure communication protocols, such as HTTPS, to protect data transmission between clients and the AI model under inference.

Model monitoring

Triton Inference Server provides model metrics that offer valuable insights into the performance and behavior of the deployed models during inference. These metrics help monitor and optimize the inference process in production environments. Some of the key model metrics provided by Triton Inference Server include:

- **Latency**—Triton reports the average time it takes to process a single inference request (in milliseconds).
- **Throughput**—Triton measures the number of inference requests processed per second.
- **GPU use**—For models running on GPUs, Triton provides metrics on GPU use, indicating how much of the GPU's processing power is being used.
- **Memory use**—Triton reports the GPU memory used by each model instance, ensuring that the GPU memory is efficiently used and avoiding memory-related issues.
- **Inference count**—Triton monitors the number of inferences made by each model instance.

These metrics can be accessed and monitored through Triton's integrated monitoring endpoints, Prometheus, or other monitoring and visualization tools like Grafana. By analyzing these model metrics, developers and system administrators can gain a comprehensive understanding of the model's performance, resource use, and overall health, enabling them to optimize the deployment and ensure the model operates efficiently in production.

Chapter 8 Dell Professional Services for Generative AI

Dell Professional Services for Generative AI harness the power of this rapidly evolving technology in a meaningful and secure way to drive the outcomes that your business expects. By partnering with Dell Technologies, your business can confidently advance generative AI initiatives, knowing you can rely on us every step of the way, with services for strategy, implementation, adoption, and scaling generative AI solutions across your organization, including the Dell Validated Design for Generative AI Inferencing with NVIDIA.

Advisory Services for Generative AI

Create a strategy and road map to achieve your generative AI vision:

- Dell experts assess your current environment, including generative AI business drivers, goals, challenges, and constraints.
- Prioritizing your business use cases, our professionals define your ideal future state, design a new generative AI solution architecture leveraging this design and identify the skills your IT organization needs to succeed.
- We then create and validate a generative AI road map for your business to realize the value of generative AI, define your qualitative business rationale, develop recommendations and next steps, and present the road map to your executives.

Implementation Services for Generative AI

Establish your generative AI inferencing platform using the Dell Validated Design for Generative AI Inferencing with NVIDIA:

- Dell Technologies holds a workshop with your project stakeholders, reviewing the approach required to implement platform architecture using Dell Validated Design for Generative AI with NVIDIA
- We then deploy the necessary tools and frameworks to establish an operational generative AI platform, guide by this validated design.
- Before handing over operations to your team, we conduct generative AI platform knowledge transfer focused on deployed solutions to enable your team’s generative AI success.
- Prepare your data for Generative AI inferencing and model customization with Data preparation consulting services.

Adoption Services for Generative AI

Achieve your unique inferencing use cases using a pretrained model with the deployed generative AI platform using the Dell Validated Design for Generative AI with NVIDIA:

- Through multiple workshops, Dell professionals align with project stakeholders to review your use cases and determine the best model to meet your needs.
- With your unique use cases in mind, Dell generative AI experts deploy and configure the pretrained model for your business.
- We then conduct knowledge transfer sessions covering software stack use, architecture, and best practices for adoption of your new inferencing model.

Scale Services for Generative AI

Optimize processes and advance a generative AI mindset throughout your organization:

- Address key IT skills gaps with Education Services for Generative AI; we work directly with your team and ensure that you are up to speed.
- Simplify your GenAI operations with Managed Services for Generative AI, with Dell managing your NVIDIA solutions stack so that you can focus on your GenAI model development and use cases.
- Dell resident experts provide the expertise that is needed to drive your generative AI initiative and keep your generative AI infrastructure using the Dell Validated Design for Generative AI with NVIDIA running at its peak.

Chapter 9 Conclusion

Summary

The Dell Validated Design for Generative AI Inferencing with NVIDIA has been developed to address the needs of enterprises that need to develop and run custom AI LLMs using domain-specific data that is relevant to their own organization.

Dell Technologies and NVIDIA have designed a scalable, modular, and high-performance architecture that enables enterprises to more quickly design and deploy an inferencing solution that has been validated and performance-tested to accelerate the time to value and to reduce the risk and uncertainty by using a proven design.

This guide provides design guidance and a fully validated reference architecture for generative AI inferencing. Topics that were discussed include:

- The definition of inferencing and how it fits into the AI model development life cycle
- Use cases for inferencing, including some of the challenges in implementation
- Explanation of LLM characteristics and examples
- Details about the Dell PowerEdge servers and NVIDIA GPUs used in the design, including GPU configurations, GPU connectivity, and networking methods
- Descriptions of the primary software components used for inferencing, including NVDIA AI Enterprise, Triton Inference Server, the NeMo framework for generative AI models, and the NVIDIA Base Command Manager Essentials
- A detailed description of the reference architecture for generative AI inferencing, including both the physical hardware and the software architecture
- A presentation of the validation results, including the numerous models used for validation and the multiple validation scenarios
- A listing of the performance test results and how they influence the infrastructure sizing recommendations
- Guidelines for inferencing operations when the solution is deployed and operational
- Descriptions of the Dell professional consulting services that have been designed specifically for this validated design for generative AI, including advisory, implementation, adoption, and scaling services

While this design focuses on AI inferencing of pretrained models, it is the first in a series of validated designs for generative AI that focus on all facets of the generative AI life cycle, including inferencing, model customization, and model training. While these designs focus on generative AI use cases, the architecture is more broadly applicable to more general AI use cases as well.

With this project, Dell Technologies and NVIDIA enable organizations to deliver full-stack generative AI solutions built on the best of Dell infrastructure and software, combined with the latest NVIDIA accelerators, AI software, and AI expertise. This combination of components enables enterprises to use purpose-built generative AI on-premises to solve their business challenges. Together, we are leading the way in driving the next wave of innovation in the enterprise AI landscape.

We value your feedback

Dell Technologies and the authors of this document welcome your feedback on the solution and the solution documentation. Contact the Dell Technologies Solutions team by email.

Chapter 10 References

The following links provide additional information about the solution design and components in this guide.

Dell Technologies documentation

The following Dell Technologies documentation provides additional and relevant information.

- Dell Generative AI Solutions (Dell.com/ai)
- Dell Info Hub for AI
- White Paper – Generative AI in the Enterprise
- Design Guide – Optimize Machine Learning Through MLOps with Dell Technologies and cnvrg.io

NVIDIA documentation

The following NVIDIA documentation provides additional and relevant information:

- What is NeMo?
- NVIDIA AI Enterprise documentation

Other documentation

The following documentation provides additional and relevant information:

- Stanford Politeness Corpus dataset