Generative AI in the Enterprise - Inferencing

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

March 2024

H19686.2

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

Copyright

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.

In 2023, 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 delivered 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. This security is 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 one of 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 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.

This design guide can be read along with the associated white paper Generative AI in the Enterprise. 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 used in the architecture. We also recommend reading the related design guide Generative AI in the Enterprise – Model Customization, which focuses on Inferencing and deployment of pretrained models and the technical white paper Generative AI in the Enterprise – Training, which focuses on training new models from scratch.

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
March 2024	3	Added support for PowerEdge R760xa servers with NVIDIA L40S GPUs. Removed NVIDIA FasterTransformer and replaced by TensorRT-LLM Updated models used for validation.

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, final content, documents, and reports. Inferencing can be used to generate concise summaries of longer texts, making information more accessible and digestible. It can help create 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 that 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 in multiple ways to assist or automate various tasks. It can perform code generation or code autocompletion, based on high-level descriptions or prompts and on context to make writing code faster and more efficient. Inferencing 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 for 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, that 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.

  Named Entity Recognition (NER) is a natural language processing (NLP) technique used to identify and categorize specific entities in 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 identify 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 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 pretrained 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 starting point makes the process of customizing and fine-tuning for specific tasks 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 that 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, a smaller model that is trained to be specific to a particular outcome might be more accurate. Models with higher parameters also require more compute resources, especially GPU resources. Therefore, a balance must 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 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 validated Llama 2 and Mistral as the foundation models for inferencing with this infrastructure design with Triton Inference Server. For more information about the models that we validated, see Table 9 in Validation Results.

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.

Mistral

Mistral 7B, a model boasting 7.3B parameters, exhibits exceptional performance when compared to models of similar magnitude. Employing Grouped-query attention (GQA) enhances its inference speed, while Sliding Window Attention (SWA) efficiently manages longer sequences at a reduced computational expense. Released under the Apache 2.0 license, Mistral 7B is freely available for unrestricted use and is effortlessly adaptable for fine-tuning across various tasks.

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 design are available as part of NVIDIA AI Enterprise:

- Triton Inference Server with TensorRT-LLM
- 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 **Error! Reference source not found.Error! Reference source not found.**.

Triton Inference Server

NVIDIA Triton Inference Server (also known as Triton) is inference serving software that standardizes AI model deployment and execution. It 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 TensorRT-LLM, 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 and 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 design. It is the software that hosts generative AI models. Triton Inference Server provides an ideal software for deploying generative AI models.

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 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

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 with several GPU options tailored for generative AI purposes:

- PowerEdge R760xa server, capable of supporting up to four NVIDIA H100 GPUs or four NVIDIA L40S 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 L40S PCIe GPU
Supported latest PowerEdge servers (and maximum number of GPUs)	PowerEdge XE9680 (8) PowerEdge XE8640 (4)	PowerEdge R760xa (4)	PowerEdge R760xa (4)
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
Multi-instance GPU support	Up to 7 MIGs	Up to 7 MIGs	None
Max thermal design power (TDP)	700 W	350 W	350 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	Discriminative/Predictive AI Inference Gen AI Inferencing

GPU 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.

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:

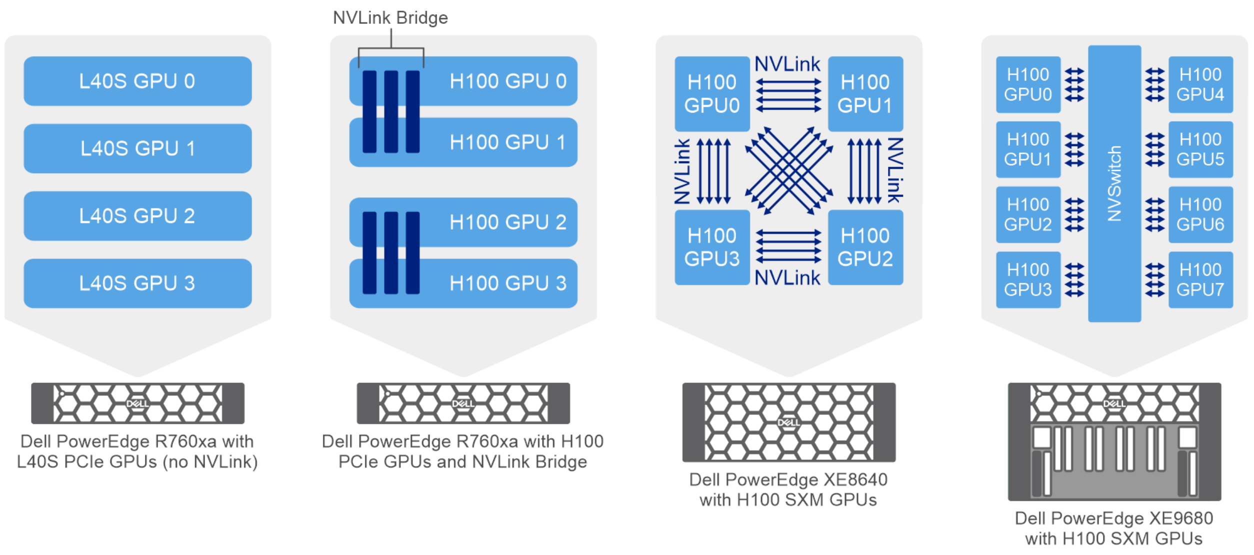

Figure 1. NVIDIA GPU connectivity in PowerEdge servers

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.

PowerEdge R760xa server with NVIDIA L40S do not support NVLink. The communication between GPUS is through the PCIe bus.

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.

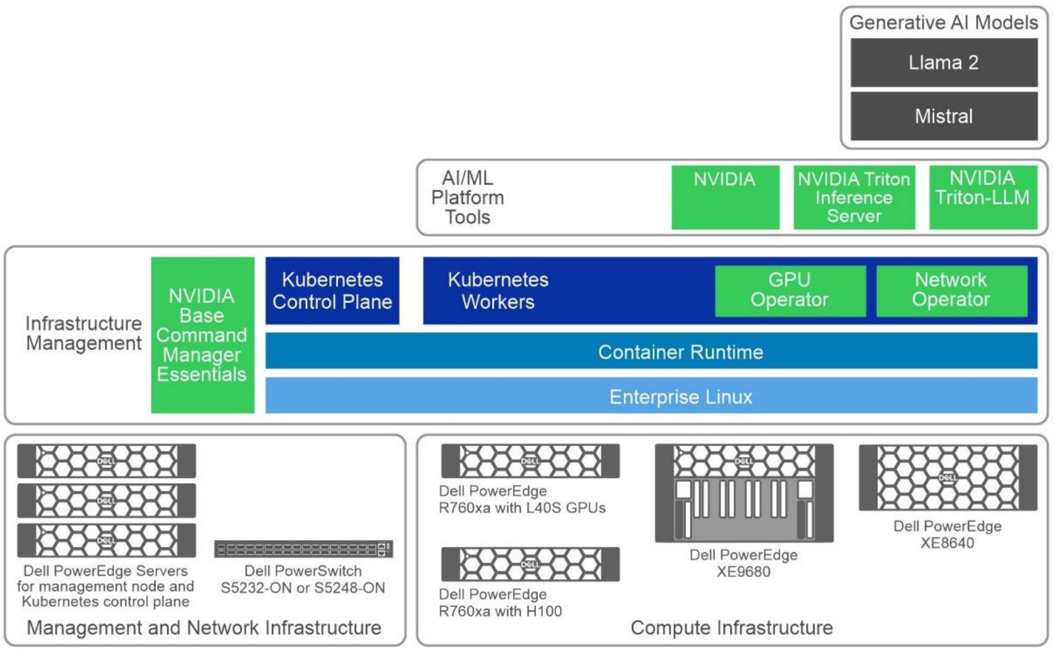
Generative AI Models
Llama 2
Mistral
AI/ML Platform Tools
NVIDIA
NVIDIA Triton Inference Server
NVIDIA Triton-LLM
Infrastructure Management
NVIDIA Base Command Manager Essentials
Kubernetes Control Plane
Kubernetes Workers
GPU Operator
Network Operator
Container Runtime
Enterprise Linux
Dell PowerEdge Servers for management node and Kubernetes control plane
Dell PowerSwitch S5232-ON or S5248-ON
Management and Network Infrastructure
Dell PowerEdge R760xa with L40S GPUs
Dell PowerEdge R760xa with H100
Dell PowerEdge XE9680
Dell PowerEdge XE8640
Compute Infrastructure

Figure 2. High-level solution 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 with GPU options were validated 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.
- PowerEdge R760xa servers supporting up to four NVIDIA L40S PCIe GPUs

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 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 practice 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 and TensorRT-LLM

The heart of the AI inference system is the NVIDIA Triton Inference Server with TensorRT-LLM, described earlier, that 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. TensorRT-LLM is the engine for LLM inference.

AI models

TensorRT-LLM supports the latest LLM models including Llama 2 and Mistral. T Chapter 5 lists the specific models that we validated in this design.

MLOps

Organizations seeking comprehensive model life 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 that 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 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 Optimize Machine Learning through MLOPs with Dell Technologies and cnvrg.io design guide.

With all the architectural elements described in this section, 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 3. 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 6438M 2.2G, 32C/64T
Memory	8 x 16 GB DDR5 4800 MT/s RDIMM
Operating system	BOSS-N1 controller card + with 2 M.2 960GB (RAID 1)
RAID controller	PERC H755 with rear load Brackets
Storage	4 x 3.84 TB SSD SAS RI 24Gbps 512e 2.5in Hot-Plug, AG Drive 1DWPD
PXE network	Broadcom 5720 Dual Port 1 GbE Optional LOM
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 that are 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 **Error! Reference source not found.**.

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 4. PowerEdge R760xa GPU worker node

Component	Details
Server model	PowerEdge R760xa
CPU	2 x Platinum 8468 2.1G, 48C/96T
Memory	16 x 32 GB DDR5 4800 MT/s RDIMM
Operating system	BOSS-N1 controller card + with 2 M.2 960GB (RAID 1)
Storage	2 x 3.84 TB Data Center NVMe Read Intensive AG Drive U2 Gen4
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 L40S, 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 5. 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
Operating system	BOSS-N1 controller card + with 2 M.2 960GB (RAID 1)
Storage	2 x 3.84 TB Data Center NVMe Read Intensive AG Drive U2 Gen4
PXE Network	Broadcom 5720 Dual Port 1 GbE Optional LOM
K8S/Storage Network	1 x NVIDIA ConnectX-6 Dual Port 100 GbE QSFP56 Adapter, OCP 3.01 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 6. PowerEdge XE9680 GPU worker node

Component	Details
Server model	PowerEdge XE9680
CPU	2 x Platinum 8468 2.1G, 48C/96T
Memory	16 x 64 GB RDIMM, 4800 MT/s Dual Rank
Operating system	BOSS-N1 controller card + with 2 M.2 960GB (RAID 1)
Storage	2 x 3.84 TB Data Center NVMe Read Intensive AG Drive U2 Gen4
PXE Network	Broadcom 5720 Dual Port 1 GbE Optional LOM
K8S/Storage Network	2 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 PowerEdge XE9680 GPU worker node configuration exceeds that of the PowerEdge XE8640 configuration. This increase is attributed to the presence of twice as many GPUs that 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 allowance for various data processing activities, machine learning operations, monitoring, and logging tasks. Our goal is to guarantee that the servers provide ample CPU and memory resources for these functions, preventing any potential disruptions to the critical inferencing operations run on the GPUs.

Secured Component Verification

Dell Technologies Secured Component Verification (SCV) is a step in the Dell production process that provides assurance of product integrity from the time an order is fulfilled at the Dell factory to end-user delivery. When a client or server product is built, a manifest of installed components is generated, cryptographically signed by a Dell Certificate Authority, and stored securely in the system. When the product is received, customers have a designated SCV validation application, allowing them to verify and validate that no unauthorized system modifications have been made to the components. For more information, see Dell Technologies Secured Component Verification.

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.

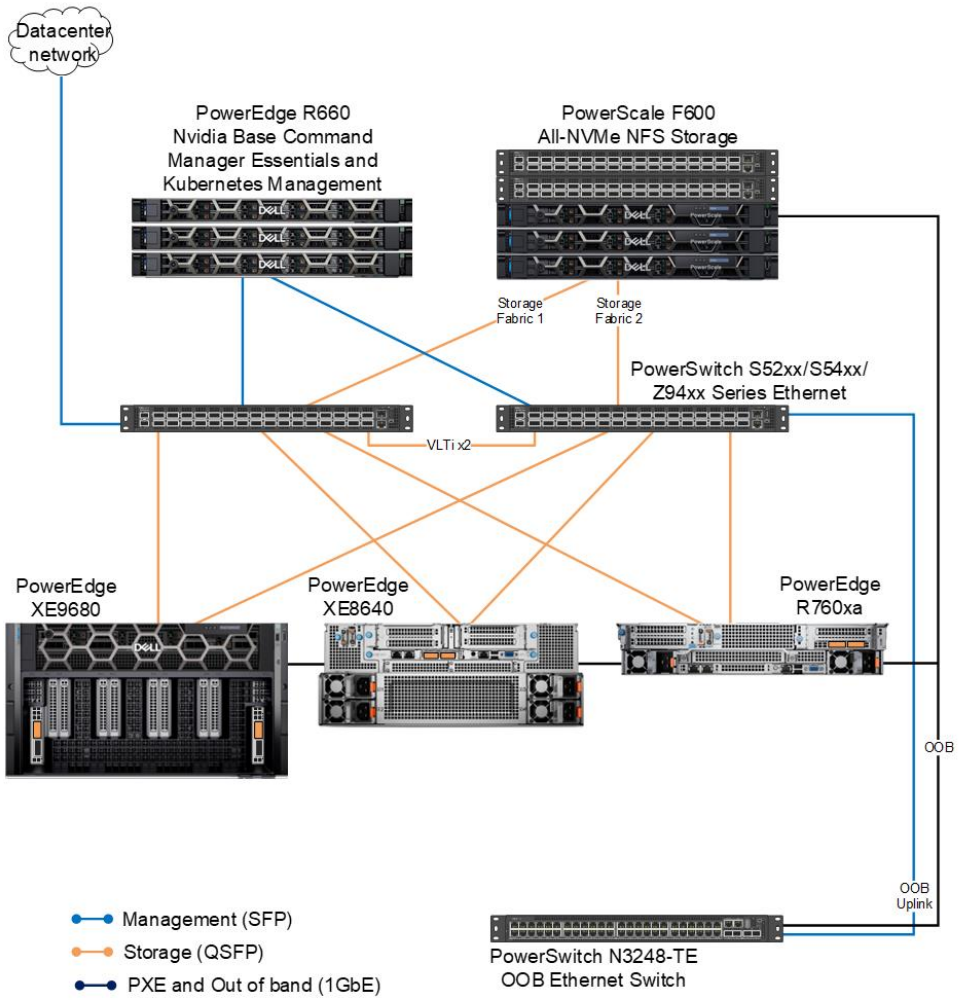

Figure 3. Network architecture

This 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.

Rack design

The following figure shows an example rack design for this design.

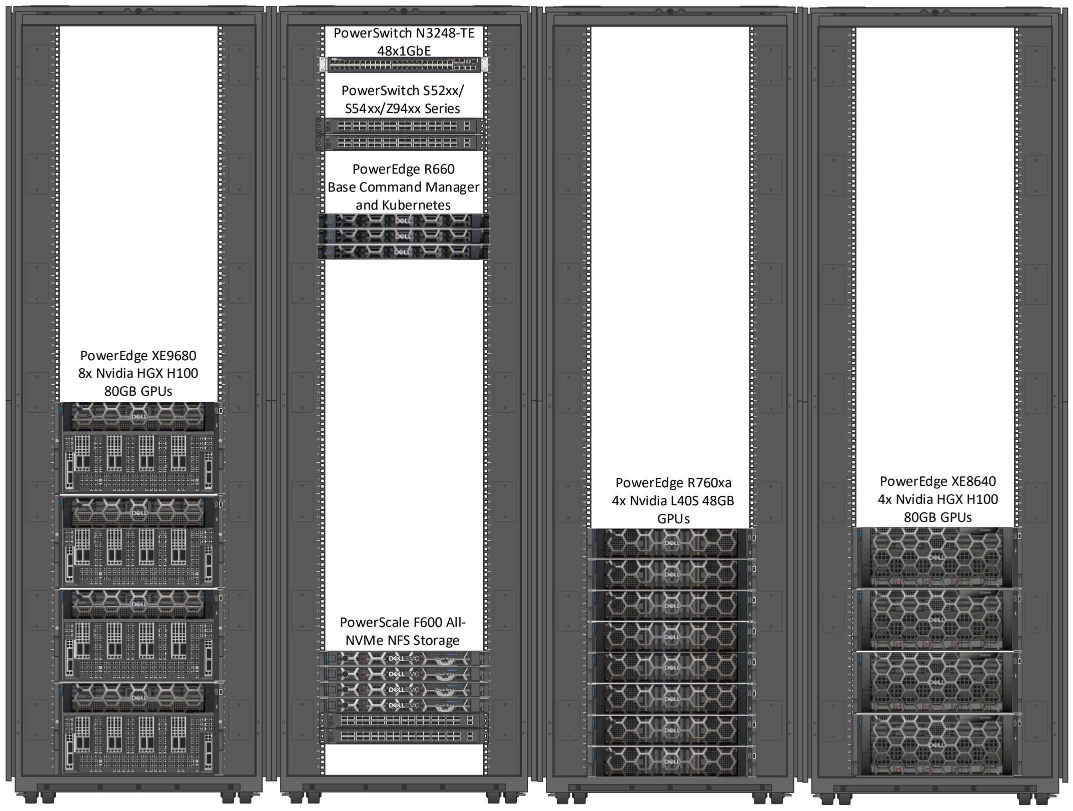
PowerSwitch N3248-TE 48x1GbE
PowerSwitch S52xx/ S54xx/Z94xx Series
PowerEdge R660 Base Command Manager and Kubernetes
PowerEdge XE9680 8x Nvidia HGX H100 80GB GPUs
PowerScale F600 All-NVMe NFS Storage
PowerEdge R760xa 4x Nvidia L40S 48GB GPUs
PowerEdge XE8640 4x Nvidia HGX H100 80GB GPUs

Figure 4. Example rack configuration for Validated Design for Model Customization

This rack was created by using the Dell Enterprise Infrastructure Planning Tool (with the illustrations of the switches enhanced). Filler panels are not shown here. You can use the tool to determine your solution and determine weight, power requirements, airflow, and other details.

This example shows four PowerEdge XE9680 servers in a single rack as well as four PowerEdge XE8640 servers and four PowerEdge R760xa servers in a separate single rack. The four PowerEdge XE9680 servers require four 17kW Power Distribution Units (PDUs). However, you must carefully evaluate your own power and cooling requirements and preference for rack layout, power distribution, airflow management, and cabling design.

If significant growth is anticipated in the size of the deployment, consider separate racks for compute, storage, and management nodes to allow sufficient capacity for that growth.

The following table provides example APC rack and PDU recommendations for the Americas region. Other rack and PDU vendors and options may be used. We recommend that you consult your Dell or APC representative to understand your unique data center requirements to provide an accurate PDU recommendation.

Table 7. Example rack and PDU recommendations for the PowerEdge XE9680 server

Servers per cabinet	Rack U height	APC rack model	PDU quantity	APC PDU model
2	42	AR3300	2	APDU10452SW
4	42	AR3350	4	APDU10452SW
2	48	AR3307	2	APDU10450SW
4	48	AR3357	4	APDU10450SW

To understand the critical aspects of deploying a PowerEdge XE9680 server, see the PowerEdge XE9680 Rack Integration technical white paper.

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	4 x PowerEdge R760xa	2 x PowerEdge XE9680	2 x PowerEdge XE8640
GPUs per server	4 x NVIDIA L40S PCIe GPUs	4 x NVIDIA H100 PCIe GPUs	8 x NVIDIA H100 SXM GPUs	4 x NVIDIA H100 SXM GPUs
Network adapter	1 x NVIDIA ConnectX-6 DX Dual Port 100 GbE	1 x NVIDIA ConnectX-6 DX Dual Port 100 GbE	2 x NVIDIA ConnectX-6 DX Dual Port 100 GbE	1 x NVIDIA ConnectX-6 DX Dual Port 100 GbE, OCP NIC 3.0
Network switch	2 x PowerSwitch S5232F-ON	2 x PowerSwitch S5232F-ON	2 x PowerSwitch S5232F-ON	2 x PowerSwitch S5232F-ON

Table 9. Software components and versions

Component	Details
Operating system	Ubuntu 22.04.1 LTS
Cluster management	NVIDIA Base Command Manager Essentials 10.23.12
Kubernetes	Upstream Kubernetes - Version v1.27.6
GPU operator	NVIDIA GPU operator v22.9.2
Inference server	NVIDIA Triton Inference Server v23.04
Inference Engine	NVIDIA TensorRT-LLM 0.7.1 and 0.8.0

We deployed and validated the models in the following scenarios:

- Deployment using Triton Inference Server with TensorRT-LLM
- Deploying using TensorRT-LLM container
- Model validation using a Text Classification

Models used for validation

TensorRT-LLM supports a wide variety of models as listed here. In this design, we validated the following generative AI models:

- Llama 2 7B, 13B, and 70B
- Mistral
- Falcon 180B

Deploying with Triton Inference Server

We used the following steps to validate the models in Triton Inference Server:

1. Clone the GitHub repository and configure the libraries:

```
$ git clone https://github.com/triton-inference-
server/tensorrtllm_backend.git

Cloning into 'tensorrtllm_backend'...
remote: Enumerating objects: 870, done.
remote: Counting objects: 100% (348/348), done.
remote: Compressing objects: 100% (165/165), done.
remote: Total 870 (delta 229), reused 242 (delta 170), pack-reused
522
Receiving objects: 100% (870/870), 387.70 KiB | 973.00 KiB/s,
done.
Resolving deltas: 100% (439/439), done.
$ cd tensorrtllm_backend/

user@node002:/aipsf600/TensonRT-LLM/v0.8.0/tensorrtllm_backend$
git submodule update --init --recursive
user@node002:/aipsf600/TensonRT-LLM/v0.8.0/tensorrtllm_backend$
git lfs install
Updated git hooks.
Git LFS initialized.
user@node002:/aipsf600/TensonRT-LLM/v0.8.0/tensorrtllm_backend$
git lfs pull
user@node002:/aipsf600/TensonRT-LLM/v0.8.0/tensorrtllm_backend$ cp
../TensorRT-LLM/examples/llama/out/*
all_models/inflight_batcher_llm/tensorrt_llm/1/
```

2. Copy the template models to `llama_ifb` and modify the configuration files from the repository skeleton with the following commands:

```
user@node002:/aipsf600/TensonRT-LLM/v0.8.0/tensorrtllm_backend$
export HF_LLAMA_MODEL=meta-llama/Llama-2-70b-chat-hf

user@node002:/aipsf600/TensonRT-LLM/v0.8.0/tensorrtllm_backend$ cp
all_models/inflight_batcher_llm/ llama_ifb -r

user@node002:/aipsf600/TensonRT-LLM/v0.8.0/tensorrtllm_backend$
python3 tools/fill_template.py -i
llama_ifb/preprocessing/config.pbtxt
tokenizer_dir:${HF_LLAMA_MODEL},tokenizer_type:llama,triton_max_ba
tch_size:64,preprocessing_instance_count:1

user@node002:/aipsf600/TensonRT-LLM/v0.8.0/tensorrtllm_backend$
python3 tools/fill_template.py -i
llama_ifb/postprocessing/config.pbtxt
tokenizer_dir:${HF_LLAMA_MODEL},tokenizer_type:llama,triton_max_ba
tch_size:64,postprocessing_instance_count:1

user@node002:/aipsf600/TensonRT-LLM/v0.8.0/tensorrtllm_backend$
python3 tools/fill_template.py -i
llama_ifb/tensorrt_llm_bls/config.pbtxt
triton_max_batch_size:64,decoupled_mode:False,bls_instance_count:1
,accumulate_tokens:False

user@node002:/aipsf600/TensonRT-LLM/v0.8.0/tensorrtllm_backend$
python3 tools/fill_template.py -i llama_ifb/ensemble/config.pbtxt
triton_max_batch_size:64

user@node002:/aipsf600/TensonRT-LLM/v0.8.0/tensorrtllm_backend$
python3 tools/fill_template.py -i
llama_ifb/tensorrt_llm/config.pbtxt
triton_max_batch_size:64,decoupled_mode:False,max_beam_width:1,eng
ine_dir:/llama_ifb/tensorrt_llm/1/,max_tokens_in_paged_kv_cache:25
60,max_attention_window_size:2560,kv_cache_free_gpu_mem_fraction:0
.5,exclude_input_in_output:True,enable_kv_cache_reuse:False,batchi
ng_strategy:inflight_batching,max_queue_delay_microseconds:600
```

3. Build a container based on the Triton trt-llm backend from GitHub. Using Docker, create the local image for triton with the most recent `trt-llm` backend.

```
user@node002:/aipsf600/TensonRT-LLM/v0.8.0/tensorrtllm_backend$
DOCKER_BUILDKIT=1 docker build -t triton_trt_llm -f
dockerfile/Dockerfile.trt_llm_backend .
```

4. Verify that the following images are created on the local repository:

```
user@node002:/aipsf600/TensonRT-LLM/v0.8.0/tensorrtllm_backend$
docker images
REPOSITORY  IMAGE ID       CREATED        SIZE
triton_trt_llm latest 03f416455199   2 hours ago    53.1GB
```

5. Run the Docker container and launch the Triton server. Specify the “world size,” which is the number of GPUs the model was built for, and point to the model repository.

```
user@node002:/aipsf600/TensonRT-
LLM/v0.8.0/tensorrtllm_backend$ docker run --rm -it --net
host --shm-size=2g --ulimit memlock=-1 --ulimit
stack=67108864 --gpus all -v $(pwd)/llama_ifb:/llama_ifb -v
$(pwd)/scripts:/opt/scripts triton_trt_llm:latest bash
```

6. Set up the Hugging Face login and install additional dependencies:

```
root@node002:/app# huggingface-cli login --token <token>
```

7. Run the Triton Server:

```
root@node002:/app# python
/opt/scripts/launch_triton_server.py --model_repo /llama_ifb/
--world_size 4
```

8. Test inference by running the following command:

```
root@node002:/app# curl -X POST
localhost:8000/v2/models/ensemble/generate -d '{
"text_input": " <s>[INST] <<SYS>> You are a helpful
assistant  <</SYS>> What is the capital of Texas?[/INST]",
"parameters": {
"max_tokens": 100,
"bad_words":[""],
"stop_words":[""],
"temperature":0.2,
"top_p":0.7
}
}'
```

9. Note that the response is similar to the following:

```
{"context_logits":0.0,"cum_log_probs":0.0,"generation_logits"
:0.0,"model_name":"ensemble","model_version":"1","output_log_
probs":[0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0],"sequence_end":false,"sequence_id":0,"seq
uence_start":false,"text_output":" Sure, I'd be happy to
help! The capital of Texas is Austin."}
```

To run the procedure from another server, replace `localhost` with the IP address or name of the host.

Deploying the TensorRT container

TensorRT-LLM can be installed in a stand-alone container without Triton Inference Server. It can also be installed with other inference server like Kserve. To deploy the container, follow the instruction described here.

Model validation using text classification

We validated the Llama 2 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 Llama 2 70B models. Other Llama 2 models scores were similar. An explanation of the performance metrics can be found here.

Table 10. Score summary

Model	Label	Precision	Recall	F1
Llama 2 70B	Polite	93.46	92.86	93.16
	Impolite	92.52	93.15	92.83

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. Also, 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. We found that the HuggingFace LLM Leaderboard is a good starting point for comparing model accuracy.

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 understand the performance of a large language model (LLM), you can measure several metrics, each of which is relevant for assessing different aspects of the model's performance. Each metric contributes to understanding LLM performance as follows:

- **First token latency—**This metric, also known as time to first token (TTFT), measures the time it takes for the model to generate the first token of a response after receiving an input prompt. It reflects the initial processing time of the model and can be important for real-time applications where low latency is crucial.
- **Tokens per second** This metric measures the throughput of the model, indicating how many tokens (words or characters) the model can generate per second on average. It provides insight into the overall speed of the model and its ability to process input data efficiently.
- **Overall response latency** Unlike first token latency, this metric measures the total time it takes for the model to generate a complete response to a given input prompt. It includes the time for processing the entire input sequence and generating the output sequence. It is crucial for assessing the end-to-end latency experienced by users interacting with the model.
- **Number of concurrent users** This metric measures the model's ability to handle multiple simultaneous requests or users. It helps determine the scalability and resource requirements of deploying the model in production environments with varying levels of user concurrency.

The following factors impact the performance of the LLM model inference:

- **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. The following table below provides GPU memory consumption for various models. The models were built with the following parameters:

- Tensor parallelism = 1
- Pipeline parallelism = 1
- Maximum input length: 2048
- Maximum output length: 2048
- Maximum batch size: 64

The memory consu =128, and output le

Table 11. Optimal Tensor and Pipeline parallelism

Model	GPU memory consumed
LLama 2 7B - FP8	23.54 GB
LLama 2 7B - AWQ	16.54 GB
LLama 2 13B - FP8	33.52 GB
LLama 2 13B - AWQ	23.67 GB
LLama 2 70B - FP8	70.06 GB
LLama 2 70B - AWQ	66.28 GB
Mistral - FP8	25.64 GB
Falcon 180B - FP8	52.29 GB

GPU type, compute, and memory use— he choice of GPU type, compute capability, and available memory are crucial factors in optimizing inference performance. Different GPU architectures (NVIDIA H100 GPUs compared to NVIDIA L40S GPUs) offer varying levels of compute power and memory bandwidth, which can affect the speed and efficiency of LLM inference.

Model build parameters—

- Parameters such as maximum input length, maximum output length, and maximum batch size determine the resources consumed by the model during inference. These parameters influence the memory and computational requirements of the model and must be carefully tuned to optimize inference performance while ensuring that the model can handle input of varying lengths and complexities.
- **Model parallelism** Model parallelism is the technique of dividing the computation of a neural network across multiple GPUs to improve throughput and reduce inference latency. By partitioning the model into smaller segments and processing them in parallel, it is possible to exploit the computational capabilities of multiple GPUs and accelerate inference speed. However, implementing model parallelism requires careful consideration of communication overhead and load balancing to ensure efficient use of resources.
- **Model quantization** Model quantization is a technique to reduce the precision of numerical values in the model parameters, typically from 32-bit floating-point numbers to lower precision formats such as 16-bit floating-point or fixed-point integers. Quantization can significantly reduce memory bandwidth requirements

and improve inference speed by allowing more efficient storage and computation of model parameters. However, quantization might also impact the accuracy and quality of model predictions. It is essential to balance the trade-offs between inference speed and model accuracy when applying quantization techniques.

Performance test results

We assessed the performance of LLMs measuring first token latency in two distinct scenarios. The first scenario pertains to chatbot interactions, characterized by short input lengths. The second scenario pertains to Retrieval-Augmented Generation (RAG) tasks, which typically entail longer input lengths. Additionally, we gauged tokens per second, reflecting offline task scenarios.

Note: These results were measured with TensorRT-LLM version 0.7.1. Newer versions of TensorRT-LLM could yield better performance results.

We used benchmarking scripts that are packaged with NVIDIA to measure the model performance in the described scenarios. These benchmarking scripts were run on TensorRT-LLM without incorporating Triton Inference Server.

The model's inference performance fluctuates depending on model parallelism. To determine the best parallelism, we selected Llama 2 13B and compared its performance by adjusting tensor and pipeline parallelism. Later, we identified the optimal TP and PP values for each GPU configuration based on the results.

Table 12. Optimal Tensor and Pipeline parallelism

NVIDIA GPU	Optimal parallelism
1 x L40S	TP=1, PP=1
2 x L40S	TP=2, PP=1
4 x L40S	TP=4, PP=1
2 x H100 SXM	TP=2, PP=1
4 x H100 SXM	TP=4, PP=1
8 x H100 SXM	TP=8, PP=1

We used the following model build parameters:

- Maximum input length: 2048
- Maximum output length: 2048
- Maximum batch size: 64

Latency tests for online inference

The goal of this benchmarking is to measure the first token latency to represent online tasks. The following tables show the results that we observed for a batch size of 1 and output length of 1 (to measure first token latency).

Table 13. First token latency for an input length of 128 (milliseconds)

Model/GPU	1 x L40S	2 x L40S	4 x L40S	1 x H100 SXM	2 x H100 SXM	4 x H100 SXM	8 x H100 SXM
LLama 2 7B - FP8	18.03	18.23	15.52	8.05	8.95	6.17	7.49
LLama 2 13B - FP8	29.56	29.77	23.49	12.16	12.61	10.24	9.40
LLama 2 70B - FP8	Not supported	Not supported	76.34	Not supported	42.05	25.09	24.53
LLama 2 7B - AWQ	10.38	10.98	11.37	9.27	9.36	Not recommended	Not recomme
LLama 2 13B - AWQ	18.33	19.33	19.56	15.21	15.86	10.88	Not recomme
LLama 2 70B - AWQ	Not supported	Not supported	60.87	58.01	58.87	30.61	Not recomme
Mistral - FP8	19.26	19.34	15.89	8.48	8.28	6.38	7.15
Falcon 180B - FP8	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported	28.15

Table 14. First token latency for an input length of 2048 (milliseconds)

Model/GPU	1 x L40S	2 x L40S	4 x L40S	1 x H100 SXM	2 x H100 SXM	4 x H100 SXM	8 x H100 SXM
LLama 2 7B - FP8	68.45	72.09	119.57	29.87	29.96	19.33	18.64
LLama 2 13B - FP8	132.55	138.92	193.84	54.89	55.32	30.9	26.20
LLama 2 70B - FP8	Not supported	Not supported	674.99	Not supported	224.21	106.33	83.44
LLama 2 7B - AWQ	116.59	125.49	134.94	80.24	81.28	Not recommended	Not recommended
LLama 2 13B - AWQ	228.81	244.12	221.86	151.15	152.19	57.52	Not recommended
LLama 2 70B - AWQ	Error	Error	825.29	755.56	731.28	252.76	Not recommended
Mistral - FP8	73.26	76.40	120.25	31.22	31.66	19.60	20.89
Falcon 180B - FP8	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported	128.84

Throughput tests for offline inference

The goal of this benchmarking is to measure the tokens per second to represent offline tasks. The following table shows the results that we observed for a batch size of 64 and an input and output length of 128.

Table 15. Throughput for LLMs (tokens per second)

Model/GPU	L40S x 1	L40S x 2	L40S x 4	XE8640 x 1	XE x 2	XE x 4	XE9680 x 8
LLama 2 7B - FP8	3355.63	4018.46	4690.64	10033.68	9917.65	14458.99	13542.70
LLama 2 13B - FP8	1901.83	2346.48	2851.63	6055.56	6594.96	9874.79	9587.35
LLama 2 70B - FP8	Not supported	Not supported	861.97	Not supported	2115.99	3189.37	3794.54
LLama 2 7B - AWQ	3046.51	3630.44	5392.88	5804.99	6649.89	Not recommended	Not recommended
LLama 2 13B - AWQ	1876.85	2258.64	2692.63	3512.79	4289.96	6117.19	Not recommended
LLama 2 70B - AWQ	Not supported	749.73	960.82	900.50	1280.65	1811.31	Not recommended
Mistral - FP8	3767.75	4268.13	4835.76	9963.33	10194.46	14208.13	13540.23
Falcon 180B - FP8	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported	2861.58

Sizing guidelines

End users typically have requirements regarding first token latency, input and output length, batch size, overall latency, and throughput. As mentioned earlier, model performance is influenced by various factors, including GPU type, model build parameters, model parallelism, and quantization. Determining the appropriate infrastructure size relies on aligning user requirements with the factors affecting model performance.

To illustrate the process of sizing, we consider two examples. In the first example, we consider a small deployment that requires smaller models in production such as the Llama 2 13 and Mistral models. These models are built with a maximum batch size of 64 (hence, they can serve a maximum of 64 concurrent requests). Two instances of each model can serve a maximum of 128 concurrent users. First token latency and tokens per second can be inferred from Table 13 and Table 15. Note that Table 13 provides first token latency for a batch size of 1. Increasing the batch size impacts first token latency and overall latency. Using the data from the previous section, we can host 2 x Llama 2 13B – AWQ and 2 x Mistral - FP8 in a single PowerEdge R760xa server with 4 x NVIDIA L40S GPUs.

In the second example, we consider a larger deployment that requires models of a mixed parameter count such as Llama 2 70B, 13B, and 7B, all requiring FP8. Using the data from the previous section, we can host all these models in a single PowerEdge XE9680 server with an NVIDIA H100 GPU.

The following table lists the server requirements for the example scenarios that we described:

Table 16. Sizing examples

Example scenario 1 with PowerEdge R760xa with 4 x NVIDIA L40S GPUs	Example scenario 2 with PowerEdge XE9680
2 x Llama 2 13B – AWQ 2 x Mistral - FP8	1 x Llama 2 70B - FP8 1 x Llama 2 13B - FP8 2 x Llama 2 7B - FP8

The performance and sizing guidelines do not consider NVIDIA’s time slicing or MIG capabilities that can partition the GPU and potentially improve the number of models hosted on a particular server. As we characterize the performance of inference with these advanced capabilities, we will update this section.

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.

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 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 generative AI operations with Managed Services for Generative AI, with Dell managing your NVIDIA solutions stack so that you can focus on your generative AI 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 NVIDIA software components used for inferencing, including NVDIA AI Enterprise, Triton Inference Server, the NeMo framework for generative AI models, TensorRT-LLM, and the 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 just one 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 Technologies Info Hub for AI
- White Paper – Generative AI in the Enterprise
- Design Guide – Generative AI in the Enterprise – Model Customization
- Technical White Paper – •Design Guide – Generative AI in the Enterprise - Training
- Design Guide – Optimize Machine Learning Through MLOps with Dell Technologies and cnvrg.io
- Dell Professional Services for Generative AI

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