Optimize AI Deployments with VMware Cloud Foundation® to Maximize Performance, Efficiency, and Flexibility

White Paper

vmware® by Broadcom

VCF-OptimizeAICloud-WP100
September 17, 2026

Table of Contents

Chapter 1: Introduction

VMware Cloud Foundation® (VCF) provides the operational elasticity and resource efficiency required for modern, scalable AI environments and helps transform an AI hardware system into an enterprise-grade private, flexible, AI-ready infrastructure. VCF 9 simplifies the deployment of these components through automation, breaks down the barriers of managing complex AI data centers, and addresses the hardware limits to efficiently managing AI private cloud.

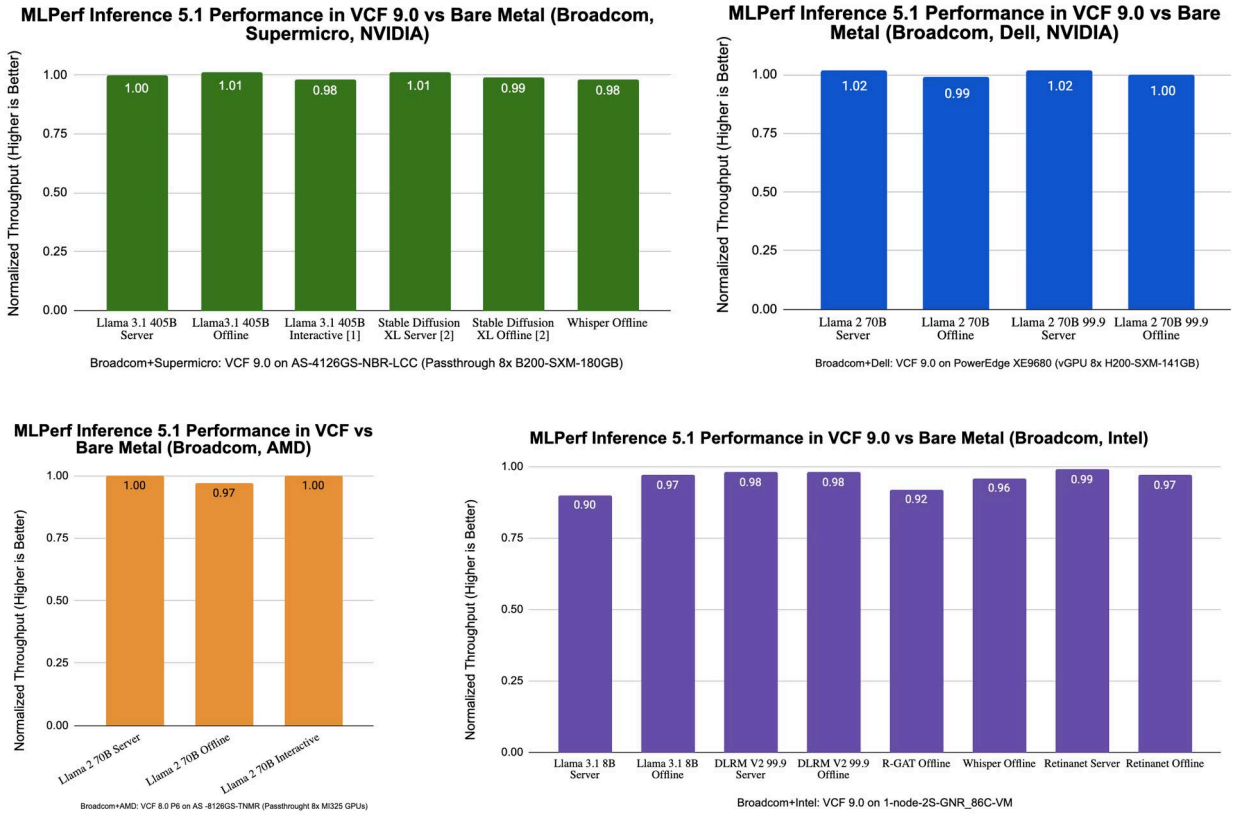
With the sharp increase in costs for data center equipment over the last year, enterprise CIOs are scrambling to maximize every component in their data center. VCF 9 can help enterprises maximize utilization of their expensive infrastructure. This paper explores how VCF 9 transcends virtualization to maximize return on investment (ROI) for enterprise AI private cloud management. This white paper provides a roadmap for building an efficient, secure, and highly elastic AI infrastructure that scales with the needs of the modern enterprise.

Chapter 2: Benefits of VCF for AI Private Cloud Management

VCF 9 empowers enterprises to build a highly secure and high-performance AI private cloud. VCF helps to optimize resource utilization for AI workloads and lower total cost of ownership (TCO) while simplifying the management of AI private clouds. Some benefits of VCF include the following:

- **Optimal AI platform:** VCF 9 is the premier platform for accelerating AI/ML, offering near-bare-metal or superior performance, particularly in AI inference. Customers can use either DirectPath I/O (passthrough) or NVIDIA vGPU technologies to allocate GPU resources for AI applications. Both passthrough and vGPU deliver enterprise-grade performance for AI workloads and enable flexible deployment options with VCF. Moreover, vGPU allows multiple virtual machines (VMs) to share physical GPUs for higher GPU utilization while maintaining workload isolation and security. The previous performance study[1], as shown in Figure 1, demonstrates that VCF delivered industry-leading results in MLPerf Inference 5.1 benchmarks across various GPUs and CPUs.

Figure 1: Performance Comparison of Virtualized vs. Bare-Metal AI/ML Workloads in VCF on Systems with Accelerators (NVIDIA and AMD GPUs) and Intel CPUs



1. Kurkure, U., Vu, L. (2025, December 15). *The New Paradigm: MLPerf Inference 5.1 Confirms VCF is the Future of AI/ML Performance*. blogs.vmware.com/cloud-foundation/2025/12/15/mlperf-5-1-confirms-vcf-future-of-ai-ml-performance

- **Unified infrastructure management:** Enterprises don't need to manage separate platforms for VMs and containers. VCF 9 provides a single, unified infrastructure platform that supports both VMs and containers through the VMware vSphere® Kubernetes Service (VKS)[2]. This unified platform provides the operational elasticity required to scale large language models (LLMs) and diverse AI/ML workloads, seamlessly bridging the gap between specialized AI hardware and enterprise-grade management.
- **Unified workload management:** By enabling mixed workloads, both GPU-intensive and CPU-intensive workloads can concurrently execute on the same server. This helps maximize resource sharing and improve infrastructure utilization, leading to lower TCO for enterprises.
- **Private cloud management benefits:** VCF brings data center management benefits to the AI world; capabilities like vMotion and live patching allow for live migration and flexible maintenance with minimal downtime. This ensures high availability and rapid recovery from hardware failures, both critical for maintaining the momentum of continuous model training or real-time inference.

2. Prasad, K. (2026, April 30). *One Platform. No Compromises.* broadcom.com/company/news/articles/cloud/vmware-cloud-foundation-unified-platform-vm-kubernetes

Chapter 3: Maximize Performance, Efficiency, and Flexibility in Multi-tenant Environments

In a multi-tenant environment, each tenant can be a department, a project, or an external customer, and all tenants share the same hardware. VCF provides the essential isolation, resource optimization, and self-service capabilities needed to make AI more efficient and operationally secure. Highlights of this chapter include the following:

- VCF enables many flexible deployments of AI cloud architectures to meet enterprise management needs while maximizing the usage of cloud resources.
- For strict isolation and security, each tenant's workloads can be deployed in a separate VM. Alternatively, they can share the same VM but be isolated in containers. GPUs can be allocated to VMs using either passthrough GPU, vGPU, or MIG vGPU, and any of these methods can be used whether tenants share a VM or are in separate VMs.
- As demonstrated, these two different deployment architectures delivered highly comparable throughput and latency in our experiments with LLM inference.

3.1 Optimize the Deployment of AI Workloads in Multi-tenant Environments

VCF supports many flexible deployment architectures for AI systems that can help maximize GPU utilization and maintain the highest performance while still ensuring the isolation, security, and high availability of AI private clouds. For high-demand AI workloads that fully consume a server's compute and memory resources, IT administrators can use VCF to provision dedicated physical servers, mirroring a bare-metal deployment while still benefiting from many of VCF's features.

For many real-world use cases, the computing resources of enterprise private clouds are underutilized much of the time because their workloads are lightweight (for example, low inference request rates or small batch size). Provisioning dedicated hardware for these workloads creates significant inefficiencies and inflates operational costs. In these cases, consolidating multiple workloads to share the same hardware is critical for the sustainable and cost-effective operation of AI private clouds.

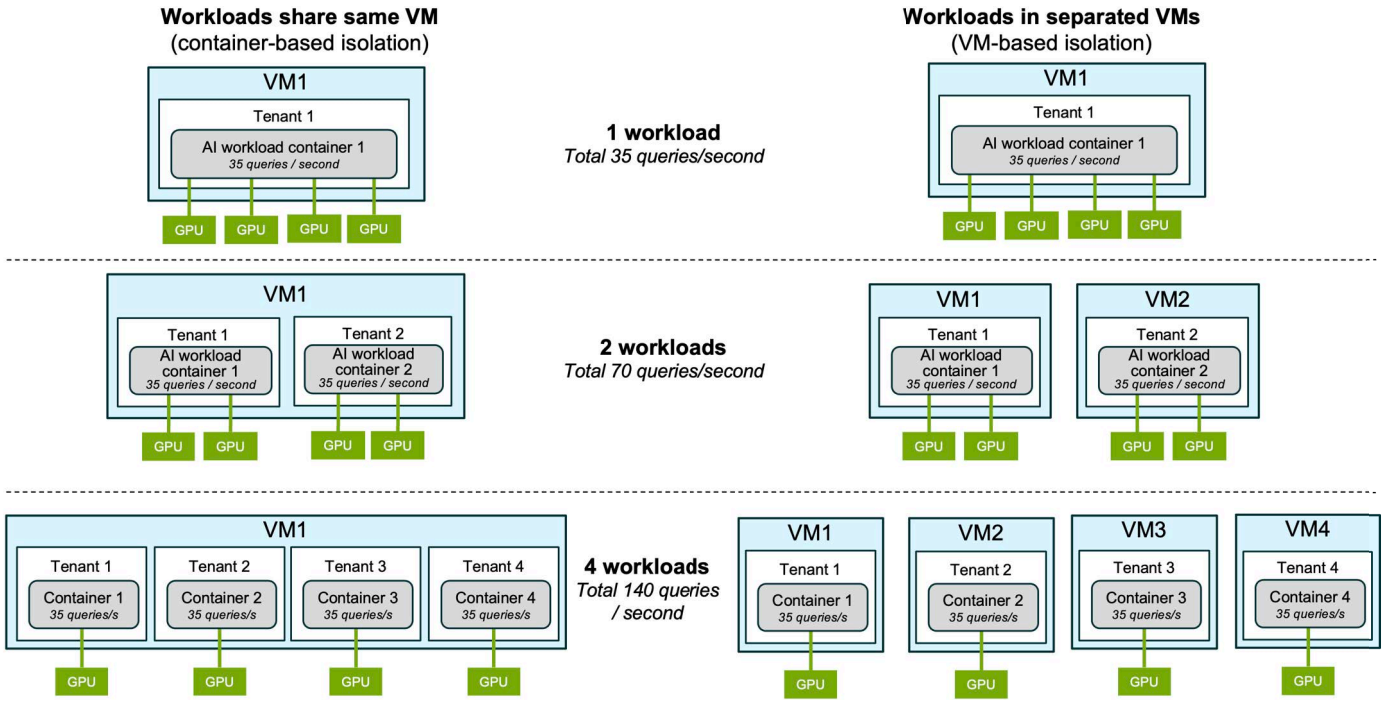
In VCF multi-tenant environments, IT administrators can choose to deploy or manage their tenants manually or through VCF Automation (refer to Tenancy Deployment Models). Depending on your enterprise's security and isolation requirements, AI workloads can be managed in the following ways:

- Tenant workloads in separated VMs (VM-based isolation):
  - Security and use case: Best for strict security and isolation requirements.
  - Configuration: Workloads for each tenant are deployed in separate VMs. GPUs are allocated to VMs using direct GPU passthrough or NVIDIA vGPU. With vGPU, a VM can be allocated full or partial (fractional) GPUs, enabling GPU resource sharing across tenant VMs. The right column in Figure 2 shows an example of this isolation. This configuration is fully supported by both VMware Cloud Foundation® Automation and manual management.
- Tenant workloads in the same VM (container-based isolation):
  - Security and use case: Best for environments with lower security requirements where workload consolidation and simplified AI cloud management are prioritized.
  - Configuration: Workloads for different tenants are consolidated into the same VMs, which can be assigned GPUs via vGPU or direct passthrough. The left column in Figure 2 shows an example of this isolation. Colocating workloads of multiple tenants within the same VM is supported only when managing tenancy manually; VCF Automation requires each tenant to reside in a dedicated VM.

3.2 Performance Comparison of the Two Deployments

For illustration, an experiment was conducted in which a tenant's workload is initially allocated to all four GPUs in the system, as shown in the first row in Figure 2 (1 workload). Assuming that this workload heavily underutilized GPUs, like the test with a single workload shown in the GPU SM utilization chart in Figure 3, more workloads can be placed on the server to increase the GPU utilization, as shown in the second row (2 workloads) and third row (4 workloads) of Figure 2. The left column shows the test cases where workloads of all tenants are in the same VM; the right column shows workloads of each tenant in a separate VM. GPUs are assigned to VMs in both cases using either passthrough GPU or vGPU.

Figure 2: Test Cases of Workloads Sharing the Same VM vs. Workloads in Separated VMs



To simulate a tenant's AI workload MLPerf Inference was run with the Llama3.1 8B workload at 35 inference queries per second. Each of these inference workloads runs inside a container. Because this workload underutilized GPUs with its lightweight inference load, up to four similar workloads could be hosted in this same server, as shown in Figure 2. When all four workloads run concurrently, the total load on the servers will be 140 queries per second. These experiments used a Dell PowerEdge XE7745 with dual AMD EPYC 9555 64-core processors, 768-GB memory, and 4x NVIDIA RTX 6000 Blackwell 96 GB GPUs, as shown in Table 1. The experiments were conducted with both passthrough GPU (Figure 3) and vGPU (Figure 4) to showcase that VCF supports both for multi-tenant deployments.

Table 1: Hardware and Software for Sizing VM with Llama3.1 8B in VCF 9.1

Parameter	Configuration
Server	Dell PowerEdge XE7745, 2x AMD EPYC 9555 64-core processors, 256 total logical cores, 768-GB memory
GPUs	4x NVIDIA RTX 6000 Blackwell 96-GB GPUs
ESX	9.1.0
NVIDIA Driver	580.126.08
Workload	MLPerf Inference 5.1, Llama3.1 8B (FP4 format) using TensorRTLLM. MLPerf Inference was used to simulate AI workloads; the results were not officially submitted to MLCommon.
VM Configuration	32 vCPUs, 64-GB memory, 1-TB disk, OS: Ubuntu 24.04

NOTE: This experiment used FP4 for Llama3.1 8B supported by RTX 6000 GPUs. For GPUs that do not support FP4, the throughput results can be lower. To get GPU SM utilization information, use the command `nvidia-smi dmon --gpm-metrics 2`; run in ESX if using vGPU or in the VM if using passthrough GPUs.

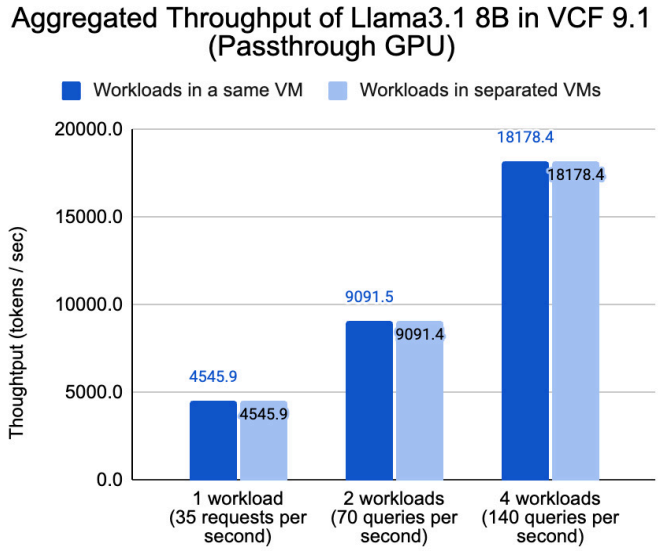
Some of the key results of our experiments include the following:

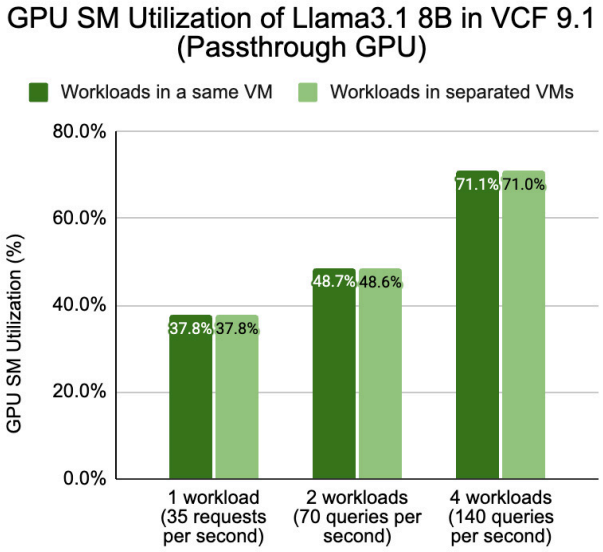
- By increasing the number of workloads with lightweight load on the same server from 1 to 4, the inference throughput of the system increased almost four times, from ~4500 to ~18,000 tokens per second, as shown in Figure 3 and Figure 4. The load is defined by the number of concurrent requests to the inference engine.
- The GPU streaming multiprocessor utilization increased to 71% with four tenants instead of being underutilized at 36% to 38% when only one tenant was deployed.
- There was not much difference in the inference throughput and latency whether loads were distributed among VMs (VM-based isolation) or aggregated into a single VM (container-based isolation).
- Comparing passthrough GPU and vGPU deployments, the throughput and GPU utilization were almost the same, while time to first token (TTFT) and time to output token (TTOT) was 2% to 7% higher with vGPU than with the passthrough GPU.
- Because of the trade-off between throughput and inference latency, consolidating more inference load per server to improve inference throughput also increases the inference latencies as shown in the orange and purple charts in Figure 3 and Figure 4 that present the TTFT and TTOT results of our experiments. When adding more GPU loads, IT administrators should ensure that the application latency of each workload still meets the enterprise requirement (SLA), as discussed in a previous blog[3].

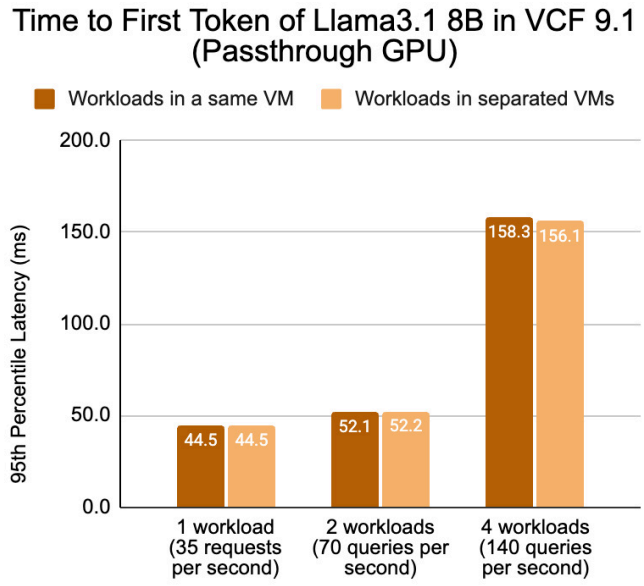
3. Corro, E., Fu, Y., (2026, April 30). *How Many Users Can Your LLM Server Really Handle?* blogs.vmware.com/cloud-foundation/2026/04/30/how-many-users-can-your-llm-server-really-handle/

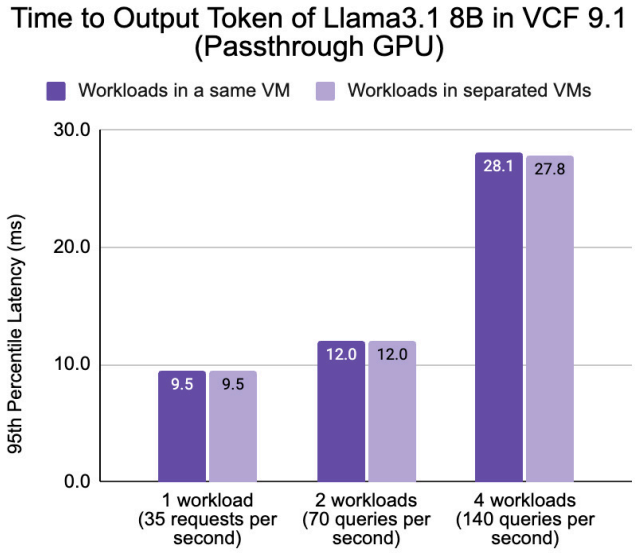
3.2.1 Performance Comparison when Deployed with Passthrough GPU

Figure 3: Performance of Tenants in Separate VMs vs. Tenants Sharing the Same VM with Passthrough GPU



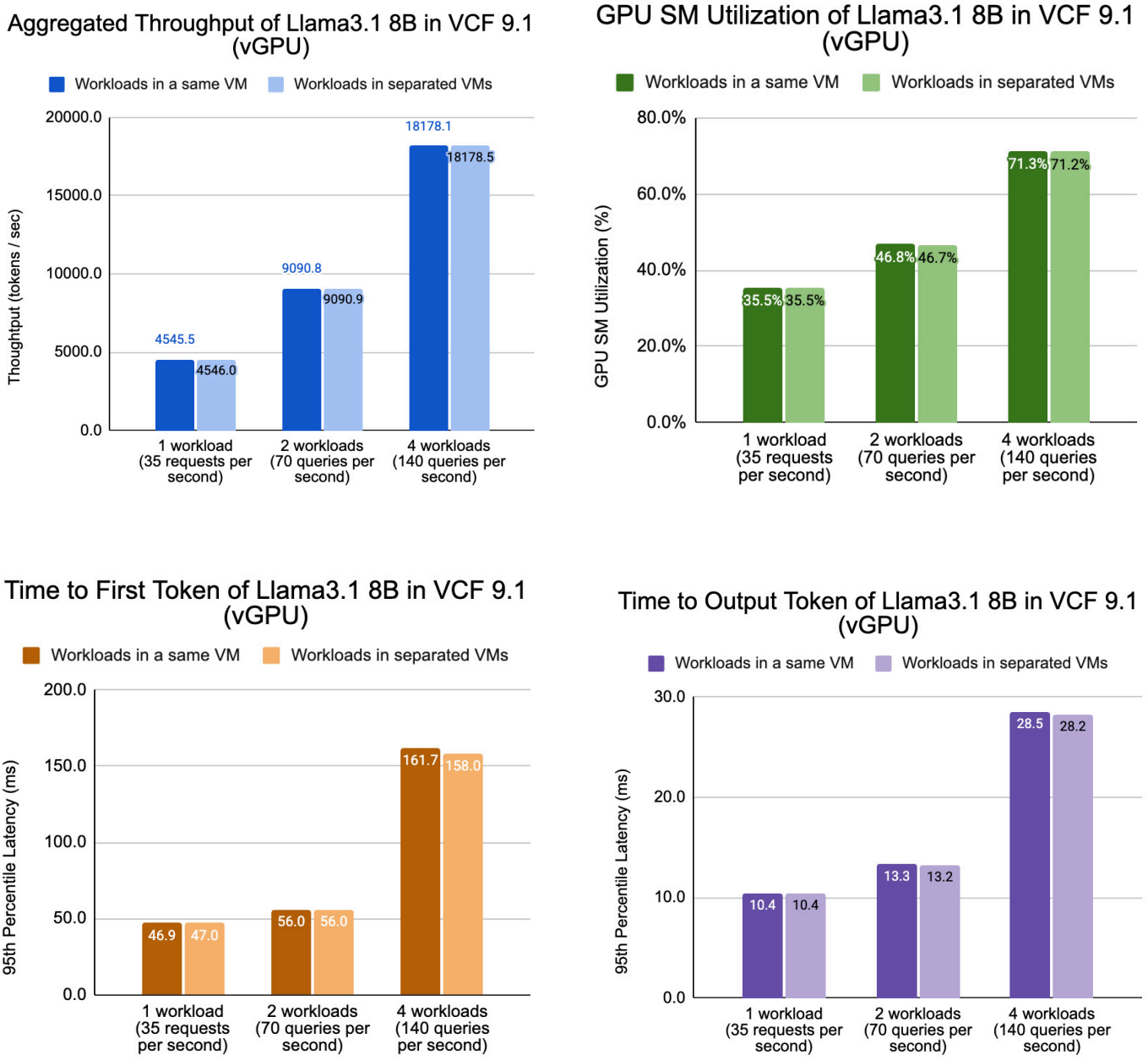






3.2.2 Performance Comparison when Deployed with vGPU

Figure 4: Performance of Tenants in Separate VMs vs. Tenants Sharing the Same VM with vGPU



Chapter 4: Maximize GPU Utilization for Best Total Cost of Ownership

Low GPU utilization may cost enterprises millions to billions of dollars annually in wasted compute resources. Because of the large investment in AI enterprise cloud, it is important for organizations to optimize the usage of their AI cloud infrastructure. This chapter presents various methods of maximizing GPU utilization to drive the efficiency of private AI clouds in VCF. Highlights of this chapter include the following:

- In VCF, DirectPath I/O (passthrough) and NVIDIA vGPU technologies are used to allocate GPU resources to VMs and enable flexible ways of deploying AI workloads to best fit the diverse needs of enterprises that bare-metal deployments normally don't support.
- Enterprises frequently underutilize their GPU resources. GPU utilization can be significantly improved by properly allocating the GPU resources for each workload, consolidating more GPU workloads per server, and consolidating more load per VM.
- vGPU and MIG vGPU can increase GPU utilization by allowing many VMs to share the same GPUs.
- Use cases and experiments show how to choose between vGPU and MIG vGPU.

VCF supports GPU technologies that ensure the best AI workload performance, including NVIDIA NVLink, NVIDIA NVLink Switch, InfiniBand/RDMA over Converged Ethernet (RoCE), and NVIDIA GPUDirect Storage (GDS). In the simplest deployments, IT administrators can have a single VM that owns the entire hardware resources of a server, similar to a bare-metal deployment, but with the added management, reliability, and flexibility benefits of VCF. Virtualization technologies supported by VCF offer many other flexible options for AI cloud deployment that help to meet the diverse needs of enterprises and optimize the usage of hardware, thus minimizing TCO.

This chapter discusses a variety of options for assigning GPU resources to VMs in order to optimize the performance of AI applications in VCF and maximize GPU utilization for highest return on investment (ROI).

4.1 Methods of Allocating GPUs to Virtual Machines in VCF

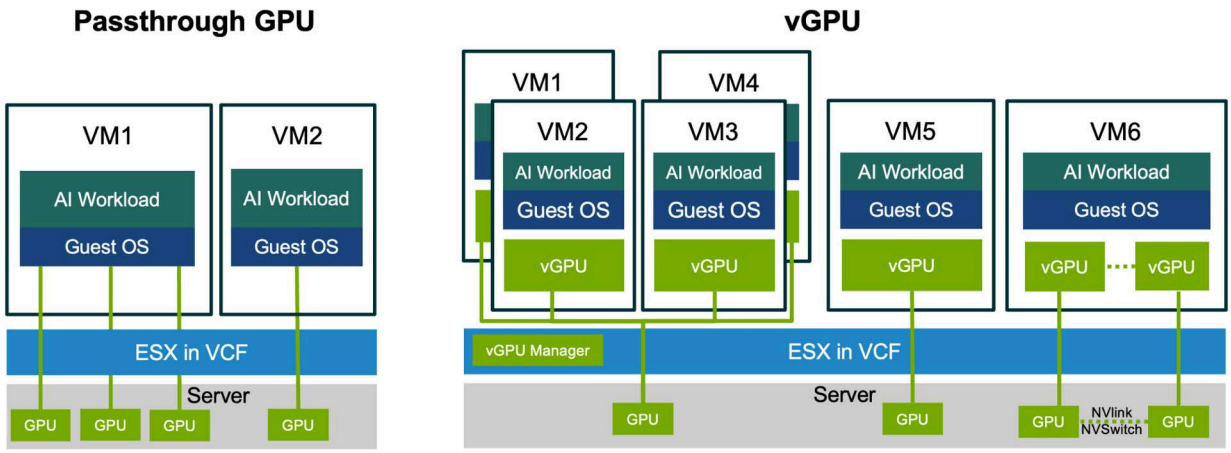
In VCF, DirectPath I/O (passthrough) and NVIDIA vGPU technologies can be used to allocate GPU resources to VMs running AI applications. Table 2 and Figure 5 present some key differences between these two methods.

Table 2: Comparing Passthrough GPU vs. vGPU

Parameter	Passthrough GPU (DirectPath I/O)	Virtual GPU (vGPU)
Architecture	Each GPU is dedicated to a single VM.	GPUs can be dedicated to VMs or shared among VMs.
Performance	Close to bare metal[a]	Close to bare metal[a]
Apps in the Same VM Can Share GPUs of the VM	Yes	Yes
Apps in Different VMs Can Share a GPU	No	Yes when using partial vGPU profiles
Consolidation Ratio of VMs with GPU	1 VM per 1 or many GPUs	■ 1 VM per 1 or many GPUs, or ■ Multiple VMs per 1 GPU: – Improve GPU utilization – Lower TCO
Device Group with NVIDIA NVLink	No device groups, so, only one VM with all GPUs has NVIDIA NVLink support.	Multiple VMs can have a device group of GPUs with NVIDIA NVLink support.
Live Migration	No	Yes, when workloads do not use Unified Virtual Memory (UVM).
Live Patching and Snapshotting	No	Yes
Workload Flexibility	VMs can have different OS and software stacks. A single server can have both AI and graphics workloads.	VMs can have different OS and software stacks. A single server can have both AI and graphics workloads.

a. Kurkure, U., Vu, L. (2025, December 15). *The New Paradigm: MLPerf Inference 5.1 Confirms VCF is the Future of AI/ML Performance.* blogs.vmware.com/cloud-foundation/2025/12/15/mlperf-5-1-confirms-vcf-future-of-ai-ml-performance

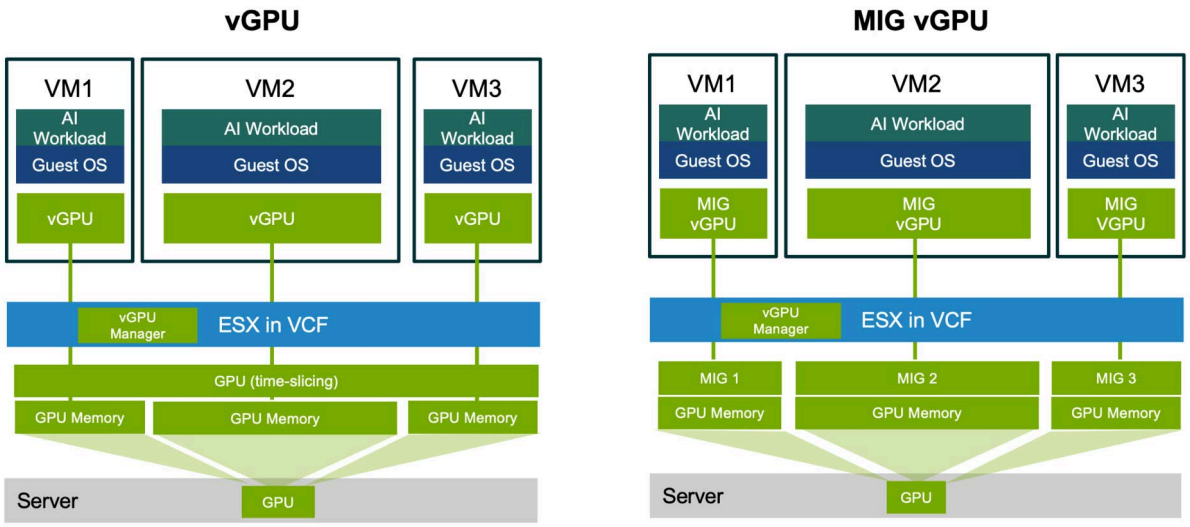
Figure 5: Methods of Allocating GPUs to Virtual Machines in VCF 9.1



With NVIDIA vGPU, a single physical GPU can be fully allocated to a VM or be partitioned among multiple VMs into virtual instances, where each receives a dedicated portion of GPU memory. There are two ways of sharing a GPU: vGPU only and MIG vGPU, both of which are shown in Figure 6. Chapter 4 discusses how to choose between vGPU and MIG vGPU.

- Time-sliced vGPU (referred to as just vGPU in this paper): All CUDA cores of the GPU are shared among VMs using time slicing.
- MIG vGPU (NVIDIA multi-instance GPU (MIG) with vGPU): Each GPU can be partitioned into multiple GPU instances—fully isolated at the hardware level, with their own high-bandwidth memory, cache, and CUDA cores—and then statically partitioned among VMs as separate vGPUs. Refer to the NVIDIA website for more information about MIG vGPU.

Figure 6: vGPU and MIG vGPU



4.2 Maximize GPU Utilization: Consolidate GPU Workloads and Share GPUs to Reduce Cost in VCF

GPUs in enterprise and data center environments are frequently underutilized, with average utilization often hovering at or below 50%[4]. VCF provides a variety of flexible ways to deploy enterprise workloads that help meet many real-world use cases and more fully utilize cloud resources. When IT administrators find that their AI workloads heavily underutilize GPU resources, they have multiple options in VCF to maximize GPU utilization of the system, reducing the TCO and achieving the best ROI on expensive AI Infrastructure. The following experiments are an illustration.

Table 3: Hardware and Software Configurations Used in these Experiments

Parameter	Configuration
Server	Dell PowerEdge XE7745, 2x AMD EPYC 9555 64-core processor, 256 total logical cores, 768-GB memory
GPU	4x NVIDIA RTX 6000 Blackwell 96-GB GPUs
ESX	9.1.0
NVIDIA Driver	580.126.08
Workload	MLPerf Inference 5.1, Llama3.1 8B (FP4 format) using TensorRTLLM. MLPerf Inference was used to simulate AI workloads; the results were not officially submitted to MLCommon.
VM Configuration	32 vCPUs, 64-GB memory, 1-TB disk, OS: Ubuntu 24.04

4. Gao, Y.et al. (2024, April 12). *An Empirical Study on Low GPU Utilization of Deep Learning Jobs*. doi.org/10.1145/3597503.3639232.

NOTE: This experiment used FP4 for Llama3.1 8B supported by RTX 6000 GPUs. For GPUs that do not support FP4, the throughput results can be lower. To get GPU SM utilization information, use the command `nvidia-smi dmon --gpm-metrics 2`; run in ESX if using vGPU or in the VM if using passthrough GPUs.

For example, suppose an enterprise has four workloads, each running on its own bare-metal server, and all underutilizing GPU resources. This was modeled in VCF as four ESX servers, as shown in Config 1 in Figure 7. To simulate a workload that underutilizes the GPU, the Llama3.1 8B (FP4) inference was run in VMs in the system detailed in Table 3, using Config 1 at only 18 queries per second per VM. In this case, four servers with four GPUs each had about 34% to 38% GPU SM utilization.

To increase GPU utilization, the four workloads can be consolidated into fewer servers. However, in some cases the enterprise prefers to keep these workloads isolated on separate machines for security and management reasons rather than colocating them on the same machine. This causes severe resource waste when deployed in bare metal.

Figure 7: Test Configurations on Servers with Four RTX 6000 Pro GPUs

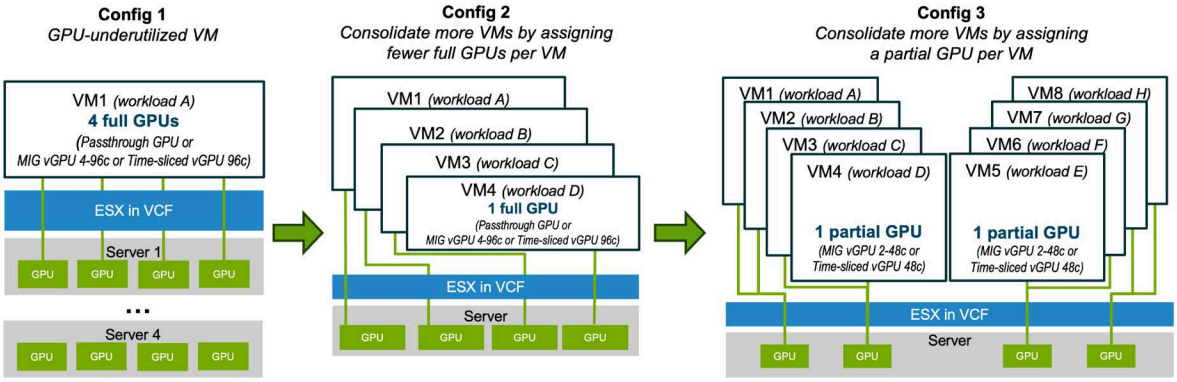
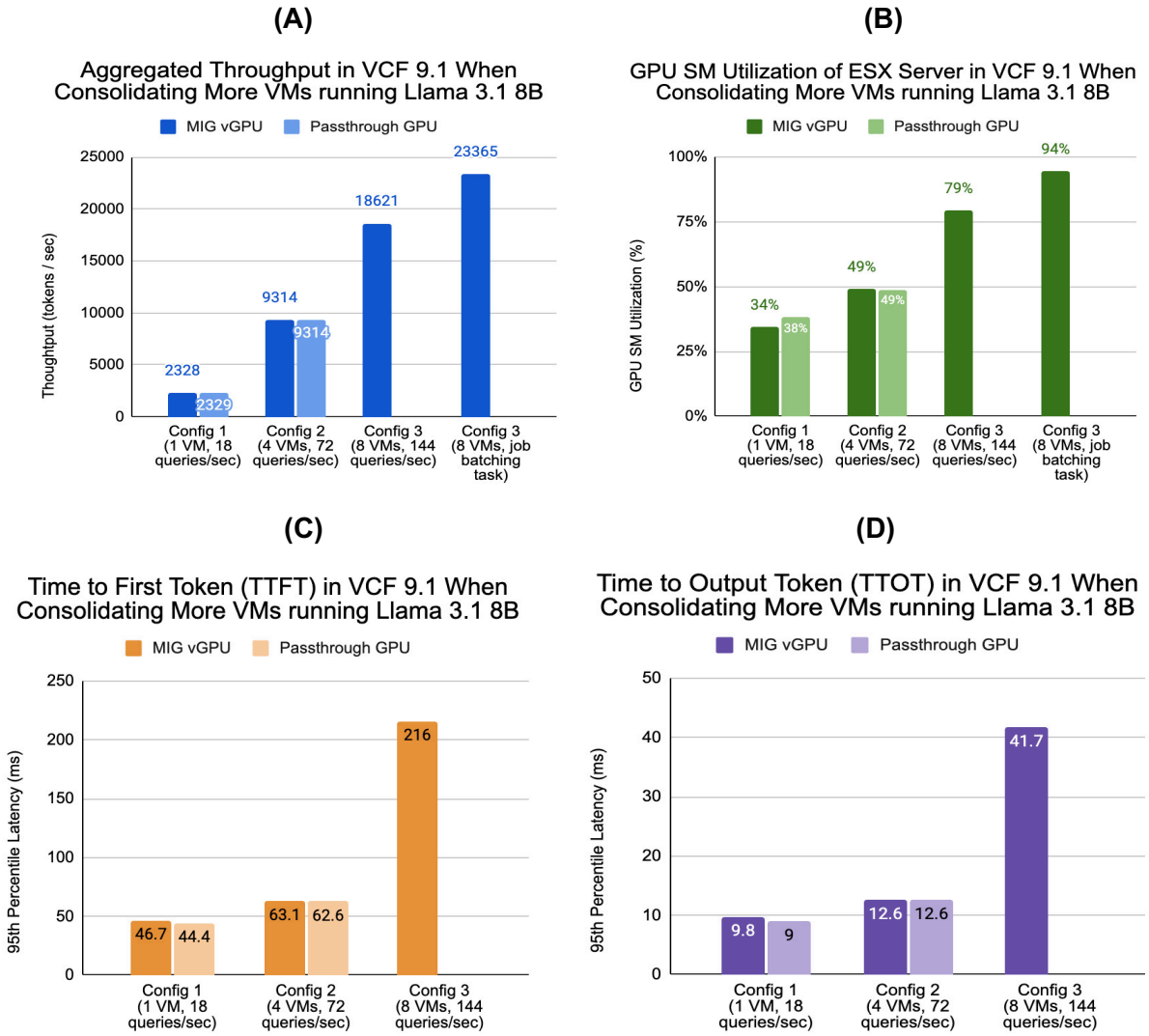


Figure 8: Inference Throughput, Latency, and GPU SM Utilization when Consolidating More VMs per Server. GPU SM Utilization Is the Percentage of Time that SMs are Actively Occupied by CUDA Kernels.



In VCF, IT administrators can use the following techniques to maximize GPU utilization:

- **Assign fewer full GPUs per VM and more AI workload VMs per server.** For example, the deployment can be changed from Config 1 to Config 2 in Figure 7 so that each workload in a VM uses one GPU and all VMs can share a server. The unused GPU servers can be used for other GPU-based workloads. GPU streaming multiprocessor (SM) utilization for the same Llama3.1 8B workloads increased to 49%, as shown in the Config 2 results in Figure 8-B. The system inference throughput quadrupled from 2328 tokens per second to 9314 tokens per second, as shown in the Config 2 results in Figure 8-A. Loading and running some workloads that use large AI models might require multiple GPUs, so reducing the number of GPUs per VM also needs to meet this constraint. The inference loads presented in the charts were the combined loads of all VMs of each test configuration in which each VM did 18 queries per second.
- **Assign a partial GPU per VM so multiple VMs share a single GPU.** For some AI models (like Llama3.1 8B in our experiments) that require less than 50% of a GPU's available memory, multiple VMs running these workloads can share physical GPUs using either time-sliced vGPU or MIG vGPU. This helps to increase the number of VMs with access to a GPU in a server and further reduce cost. This was illustrated by assigning the VMs a MIG vGPU 2-48c profile (or a half of GPU compute and memory) and adding 8 VMs with 8 workloads running Llama3.1 8B into the server, as shown in Config 3 in Figure 7. This configuration is not available for passthrough GPU, so our test results of 8 VMs are for MIG

vGPU only. Figure 8-B shows that doubling the number of VMs helps increase GPU SM utilization from 49% to 79%, as well as increasing the inference throughput from 18,621 to 23,365 tokens per second. Because of the trade off between throughput and inference latency, consolidating more VMs to improve inference throughput also increases the inference latencies, such as the TTFT and TTOT, as shown in the Config 3 results in Figure 8-C and Figure 8-D. When adding more VMs or more GPU loads, IT administrators should ensure that the application latency of each workload still meets the enterprise requirement (SLA), as discussed in a prior blog[5].

- **Increase AI workloads per VM.** If workloads are not required to be isolated in a separated VM, GPU utilization can also be increased by adding more workloads to each VM. This was demonstrated in Chapter 3, where workloads of multiple tenants were hosted in a single VM. Another way is that consolidating workloads that have loose or no latency requirements can bring better GPU utilization, as the inference requests can be batched into fewer queries to enhance the system throughput. To demonstrate, Llama3.1 8B inference was run using the offline test of MLperf Inference in Config 3 to simulate job batching of AI tasks without latency requirements. The results, in Figure 8-A and Figure 8-B, show the best throughput (23,365 tokens per second) and GPU SM utilization (94%) in our experiment.

Deployments using vGPU, as in our experiments, can also use time-sliced vGPU instead of MIG vGPU. Our performance studies[6] showed that when multiple workloads using partial (fractional) GPU concurrently share a single GPU, as in Config 3 of Figure 7, MIG vGPU usually gives better performance; that is why MIG vGPU was chosen for this test. Table 4 shows the inference throughput in Config 3 running MLPerf Inference with Llama3.1 8B in Offline Mode (job batching). The results show that MIG-vGPU gave better throughput and GPU SM utilization compared to using time-sliced vGPU.

Table 4: MIG-vGPU vs. vGPU for Eight Concurrent VMs (Config 3 in Figure 7)

Test Cases	Throughput (tokens per second)	GPU SM Utilization
8 VMs sharing 4 GPUs Each VM has a partial GPU (time-slice vGPU 48c)	18,914	84.60%
8 VMs sharing 4 GPUs Each VM has a partial GPU (MIG-vGPU-2-48c)	23,365	94.50%

4.3 Maximize GPU Utilization: Use vGPU to Share GPUs between Day and Night Distinguished Workloads

In VCF, when VMs are allocated full GPUs using either vGPU or MIG-vGPU, workloads inside these VMs access the full compute and memory capacity of the GPUs, similar to bare metal. For partial GPU use cases, vGPU and MIG-vGPU differ in how they partition the GPU and their performance for AI workloads differs as well. These use cases can guide the selection of either vGPU or MIG vGPU, and also demonstrate how they helped maximize the GPU utilization.

As shown in our previous performance study[6], as well as the experiments presented in Table 4, when a partial GPU is used for GPU sharing, choosing MIG vGPU can give better performance than vGPU when all workloads use GPUs at the same time. However, with vGPU, when two VMs share a single GPU and one of the VMs is idle, vGPU with the best-effort scheduling algorithm (the default) will allow the other VM to use all GPU time and CUDA cores. This capability benefits some enterprise use cases where using time-sliced vGPU can bring better performance than MIG vGPU.

5. Corro, E., Fu, Y., (2026, April 30). *How Many Users Can Your LLM Server Really Handle?* blogs.vmware.com/cloud-foundation/2026/04/30/how-many-users-can-your-llm-server-really-handle/
6. Vu, L., Sivaraman, H., Kurkure, U. (2022, June 3) *VMware vSphere 7 with NVIDIA AI Enterprise Time-sliced vGPU vs MIG vGPU: Choosing the Right vGPU Profile for Your Workload.* blogs.vmware.com/cloud-foundation/2022/06/03/vsphere7-vgpu-vs-mig-perf/

Examples include running real-time processing AI jobs during the day and offline processing jobs (job batching) at night, or a multinational corporation with two distinct R&D teams: Team A (North America Team) and Team B (Asia-Pacific Team). Each team runs different AI applications. Because these teams operate in nearly opposite time zones, each team's peak compute demand happens while the other team is offline. Let's say on average team A uses the system 48% of the time while team B uses the system 46% of the time (Figure 9). Instead of having two separate GPU clusters that each sit idle for 12 hours a day, the company deploys a single, shared GPU cluster and helps increase system utilization, as shown in Figure 9.

Figure 9: Achieve Peak AI Infrastructure Efficiency across Two Global Regions

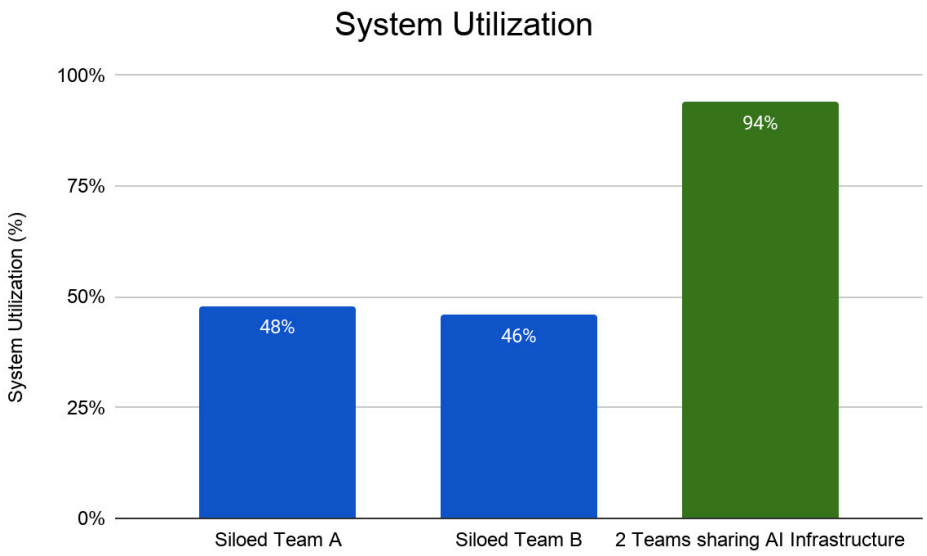
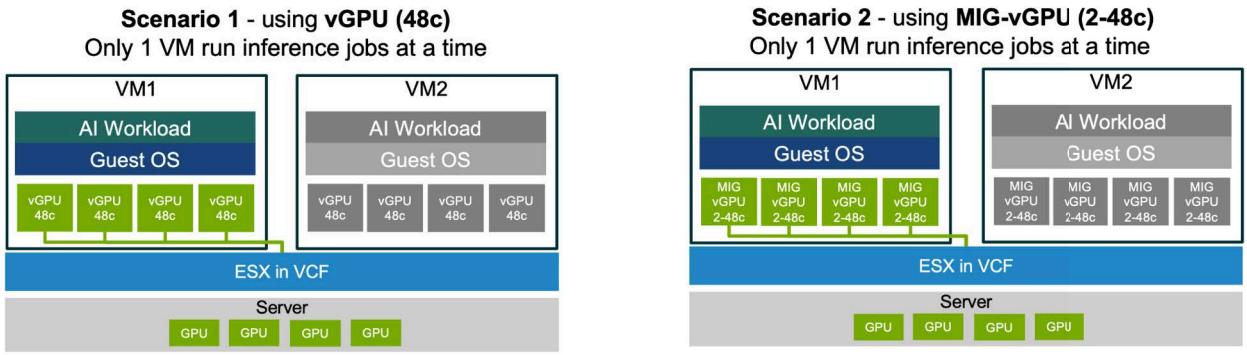


Figure 10: Test Scenarios of vGPU vs. MIG-vGPU when Only One VM Runs at a Time



To simulate these scenarios, an experiment was run with two VMs, each with four partial GPUs (either vGPU-48c or MIG-vGPU-2-48C) as shown in Figure 10 (gray VMs are powered on, but not actively running inference workloads). Each VM runs Llama3.1 8B at a load of 35 requests per second in Interactive Mode of the MLperf Inference benchmark. Only one VM at a time ran the benchmark. The results in Table 5 show that Scenario 1 (vGPU) gave better inference latency than Scenario 2 (MIG-vGPU) because when one VM runs the benchmark and the other does not, vGPU allows the running VM access to all GPU time and CUDA cores while MIG-vGPU (2-48c profile) allows the running VM to use only half the GPU time and CUDA cores. This experiment shows that if workloads use partial GPUs and their loads spike at different times, or if each VM heavily underutilizes GPUs, vGPU helps increase GPU utilization and gives better latency, both TTFT and TTOT.

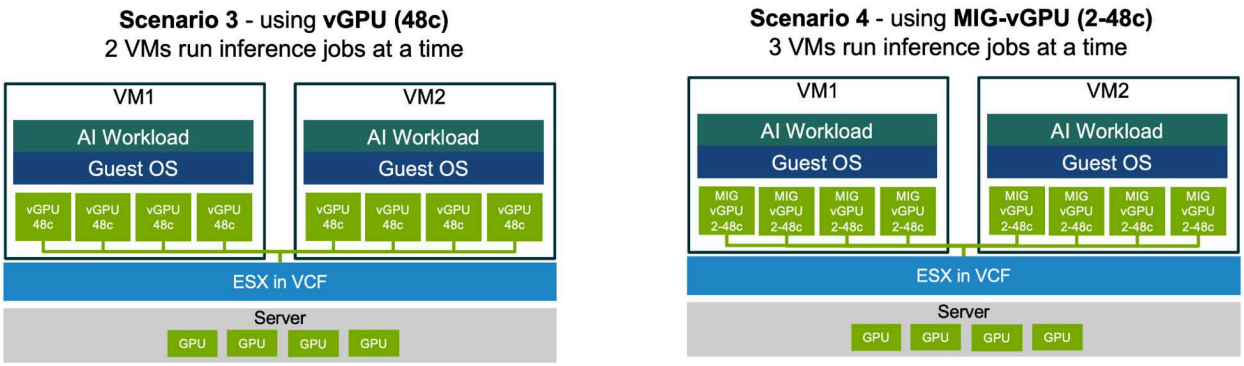
Table 5: Throughput and Latency of vGPU vs. MIG-vGPU when Only One VM Runs at a Time

Scenario	Deployment Configuration	Throughput (tokens/sec)	TTFT (ms) 95 Percentile	TTOT (ms) 95 Percentile
1	2 VMs, each has 4 time-sliced vGPUs (48c profile) Only one VM runs inference (35 queries/sec)	4546	47	10
2	2 VMs, each has 4 MIG vGPUs (2-48c) Only one VM runs inference (35 queries/sec)	4544	82	18

4.3.1 Share GPUs for Workloads that Run 24/7 Workloads

When multiple VMs share GPUs and all VMs frequently use those GPUs (24/7), MIG-vGPU provides better inference and more predictable latency. To illustrate this, tests were performed in three scenarios with the deployment configurations shown in Figure 11 and the results shown in Table 6. Each VM ran Llama3.1 8B at a load of 35 queries per second (70 queries per second for two VMs) in Interactive Mode of MLPerf Inference.

Figure 11: Test Scenarios when Two VMs Run Inference Jobs Concurrently



MIG-vGPU (2-48c profile) in Scenario 4 gives better latency (TTFT and TTOT) compared to vGPU in Scenario 3. This is because MIG-vGPU allocates to each VM a fixed partition of the GPU (both compute and memory) and the VM can utilize them the entire time. In the case of vGPU, GPU time is allocated to the VM using a time-slicing mechanism that requires context switching among VMs and creates overhead. Hence, MIG-vGPU gives better latency when all VMs utilize the GPU concurrently.

Table 6: Throughput and Latency of vGPU vs. MIG-vGPU when Two VMs Run Inference Jobs Concurrently

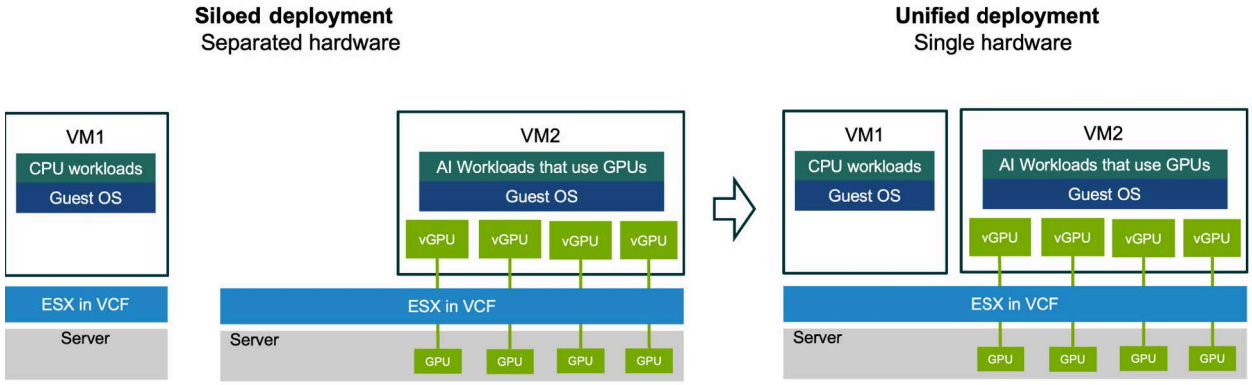
Scenario	Deployment Configuration	Throughput (tokens/sec)	TTFT (ms) 95 Percentile	TTOT (ms) 95 Percentile
3	2 VMs, each has 4 vGPUs (48c profile) Both run concurrently with total 70 queries/sec	9086	116	31
4	2 VMs, each has 4 MIG vGPUs (2-48c) Both run concurrently with total 70 queries/sec	9086	82	18

Chapter 5: Maximize CPU and Memory Utilization for Best TCO

Many GPU-accelerated AI inference workloads don't require a server's full CPU and memory capacity to perform optimally. In many cases, CPU and system memory utilization are very low. With VCF, IT administrators can optimize resource allocation by reducing the vCPU and memory assigned to these AI VMs, freeing up valuable capacity to consolidate more CPU-intensive workload VMs on the same host for best TCO. This chapter demonstrates that CPU-intensive and GPU-heavy AI VMs can coexist on the same ESX host without degrading the performance of either workload.

VCF streamlines data center management through workload consolidation, increasing density on existing hardware and reducing the costs of cloud infrastructure. According to the performance study using various MLPerf Inference workloads, most GPU-accelerated AI inference workloads have low CPU utilization. Even when only 25% to 50% of available CPU cores were allocated to the VM, the inference performance was about the same as bare metal[7]. This creates a significant opportunity to leverage the remaining CPU and memory resources to run CPU-intensive workloads, such as agentic AI and databases. Disparate workloads, even those running on different operating systems, such as Windows and Linux, can run in separate VMs on the same host and ensure full isolation, enhanced security, and simplified management. This helps save the cost of having two separate sets of hardware for different types of workloads, as illustrated in Figure 12.

Figure 12: Consolidate Workloads for Cost Savings



In VCF, CPU-intensive and GPU-intensive workload VMs can coexist on the same platform, simplifying cloud management and optimizing the usage of cloud resources, including AI private cloud. For example, suppose an enterprise has some new CPU-intensive workloads. Instead of buying new hardware for these workloads, IT administrators can complete the following steps:

1. Monitor the servers running GPU-accelerated AI workloads to identify the VMs that heavily underutilize CPUs and memory.
2. Reduce the vCPU and memory resources allocated to these VMs.
3. Allocate the newly available vCPUs and memory to the new CPU-intensive workload VMs.

This is demonstrated in Chapter 5.1 by analyzing the performance of GPU-based AI inference workloads running in VMs with various CPU and memory configurations. Chapter 5.2 showcases that GPU-intensive AI workloads can colocate with CPU workloads while still delivering excellent performance for both.

7. Kurkure, U., Vu, L. (2025, December 15). *The New Paradigm: MLPerf Inference 5.1 Confirms VCF is the Future of AI/ML Performance*. blogs.vmware.com/cloud-foundation/2025/12/15/mlperf-5-1-confirms-vcf-future-of-ai-ml-performance

5.1 Impact of CPU and Memory on AI Inference Performance

To illustrate that many GPU-accelerated AI inference workloads require only minimal CPU and memory resources to perform at maximum performance, an experiment was completed where the number of CPU cores and amount of system memory allocated to a VM running LLM inference with Llama3.1 8B and Llama3 70B were varied and measured for throughput and latency performance.

Table 7: Hardware and Software Configurations for Sizing VMs with Llama3.1 8B in VCF 9.1

Parameter	Configuration
Server	Dell PowerEdge XE7745, 2x AMD EPYC 9555 64-core processor, 256 total logical cores, 768-GB memory
GPU	4x NVIDIA RTX 6000 Blackwell 96-GB GPUs
ESX	9.1.0
NVIDIA Driver	580.126.08
Workload	MLPerf Inference 5.1, Llama3.1 8B (FP4 format) using TensorRTLLM Interactive Test with 140 queries per second
VM Configuration	Guest OS: Ubuntu 24.04, 4x vGPUs (96c profile), 1-TB disk

NOTE: This experiment used FP4 for Llama3.1 8B supported by RTX 6000 GPUs. For GPUs that do not support FP4, the throughput results can be lower.

The first experiment ran Llama3.1 8B of MLPerf Inference 5.1 (Interactive mode) in VCF 9.1 on the system described in Table 7. These tests include two scenarios: memory sizing tests and vCPU sizing tests. In the memory sizing tests, the VM was configured with 64 vCPUs and the VM memory varied from 32 GB to 512 GB. In the vCPU sizing tests, the VM was configured with 128 GB of memory and the number of vCPUs varied from 16 to 128. The results of both scenarios in Figure 13 and Figure 14 show that changing the number of vCPUs while fixing the memory size, and changing the memory size while fixing the number of vCPUs, had little or no impact on AI workloads that use GPUs for acceleration.

Figure 13: Throughput of Llama3.1 8B for Various vCPUs and Memory Sizes in VCF 9.1

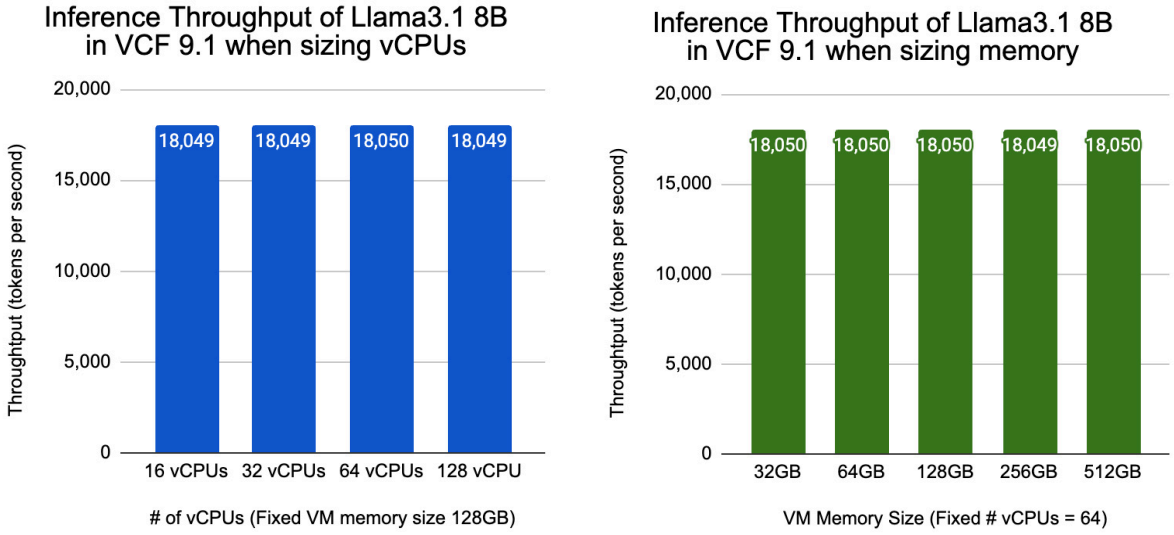
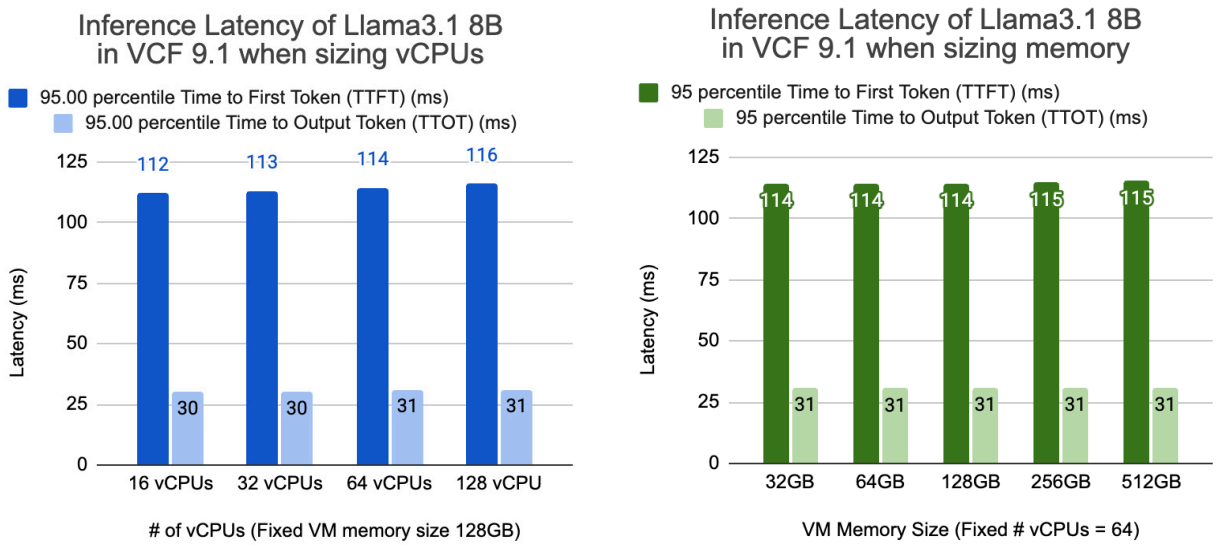


Figure 14: Latency of Llama3.1 8B for Different vCPUs and Memory Sizes in VCF 9.1



As long as the VM is allocated enough vCPUs and memory to perform the inference task, the inference throughput and latency remain the same. Moreover, another test was completed with a very small VM with 16 vCPUs (~7% of available vCPUs) and 32-GB memory (~4% of available memory). Table 8 shows that the results were almost identical to the tests in Figure 13 and Figure 14. These tests show that we don't need to allocate more vCPUs and memory than a workload uses. These unused resources can then be used to consolidate more CPU-based workloads/VMs onto the same server, as demonstrated later in Chapter 5.2.

Table 8: Inference Performance of Llama3.1 8B in a Very Small VM

VM Configuration	Throughput (tokens per second)	TTFT (ms) 95 Percentile	TTOT (ms) 95 Percentile
Guest OS: Ubuntu 24.04, 16 vCPU, 32-GB memory, 4x vGPUs (96c profile), 1-TB disk	18,049	111.7	30.2

To confirm similar impacts with different workloads, another experiment was completed in a VM running LLM inference on a Llama3 70B model compiled using NVIDIA TRT-LLM on the system described in Table 9. The model was compiled to use tensor parallelism of 4, so that it would use all four H200 GPUs on the server. The virtual memory size allocated to the VM was varied over the values of 32 GB, 128 GB, 256 GB, 512 GB, and 1024 GB. For each virtual memory size, many different values of input sequence length (ISL) and batch size were tried.

Table 9: Hardware and Software Configurations for Sizing VM with Llama3 70B in VCF 9.1

Parameter	Configuration
Server	PowerEdge XE774, AMD EPYC 9555 64-core processor, 256 total logical cores, 1.5-TB memory
GPU	4x NVIDIA H200 GPUs with NVLink
ESX	9.1.0
NVIDIA Driver	580.105
Workload	The developed benchmark with Llama3 70B (FP16 format) compiled with tensor parallelism of 4
VM Configuration	Guest OS: Ubuntu 24.04, 96 vCPUs, 4x vGPUs, 96c profile

The results for three combinations of ISL and batch size are shown in Figure 15. These results demonstrate that virtual memory size has a negligible impact on the throughput of LLM inference.

The results show that 32 GB of virtual memory is sufficient to support LLM inference using a Llama3 70B model. Allocating up to 128 GB can provide a marginal performance gain, but any memory resources provisioned beyond this threshold yielded no measurable impact on throughput or latency. These surplus resources are better utilized by being allocated to other concurrent workloads to maximize server density and ROI.

Figure 15: Normalized Throughput vs. Memory Size for a VM with 4x NVIDIA H200 GPUs

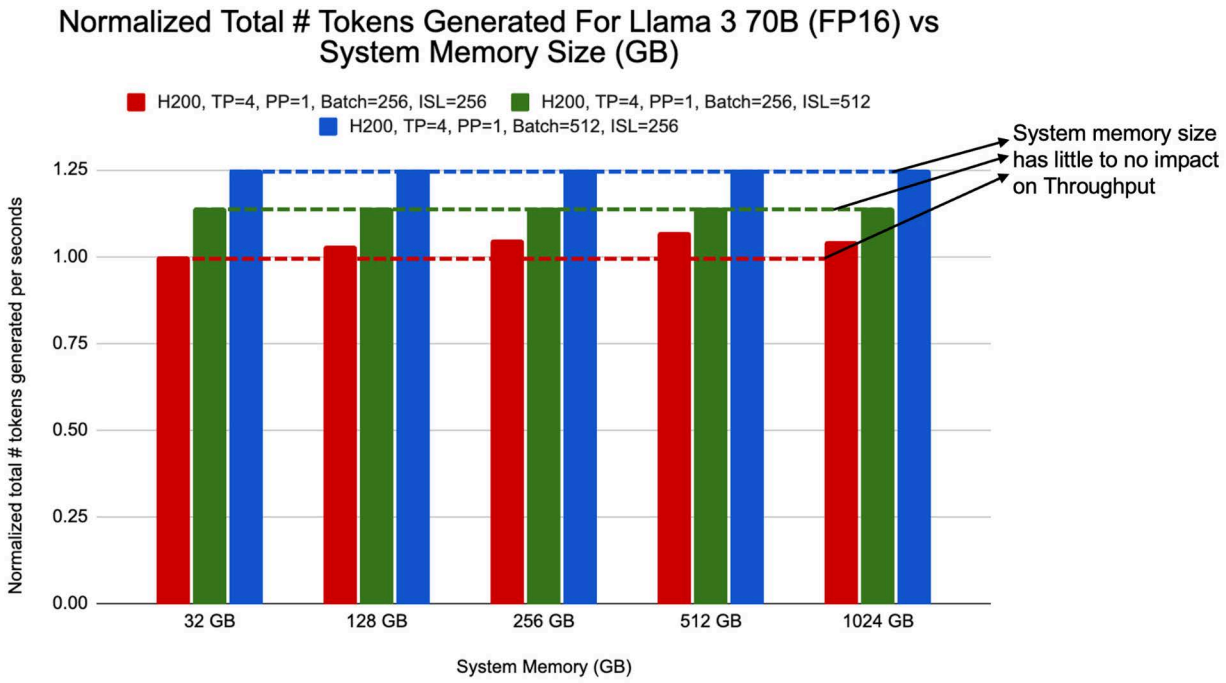


Figure 15 shows the throughput comparison for three combinations of ISL and batch-size:

- ISL = 256, batch size = 256
- ISL = 512, batch size = 256
- ISL = 256, batch size = 512

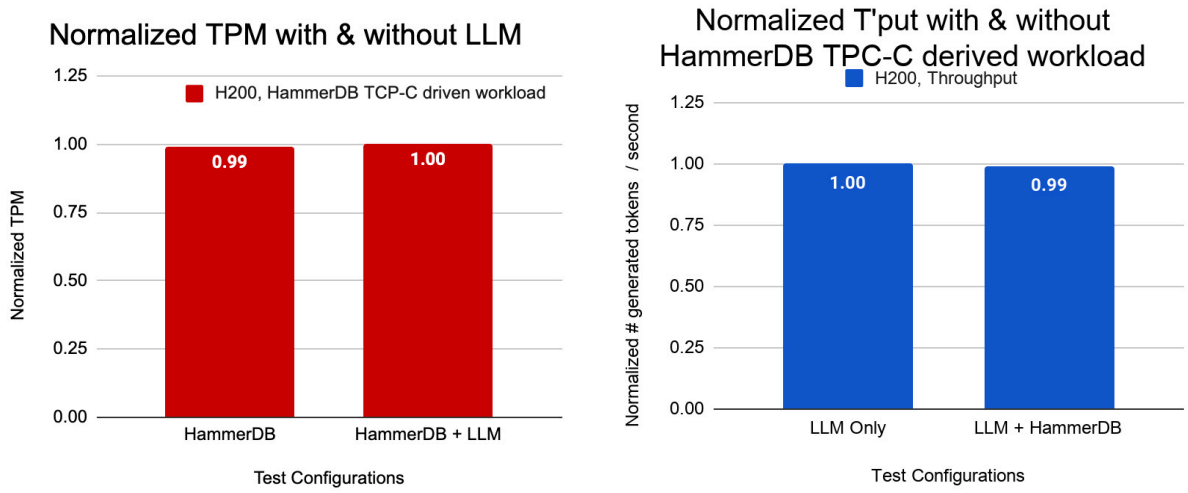
The throughput values are normalized with respect to the throughput at a virtual memory size of 32 GB, ISL = 256, and batch size = 256.

5.2 No More Silos: Consolidate CPU and GPU Workloads to Maximize Resource Utilization

Many enterprises prefer colocating both AI (including autonomous, agentic workflows) and non-AI workloads on the same infrastructure for better resource utilization and simpler private cloud management. Agentic AI, often involving RAG and tool use, acts like a specialized microservice requiring efficient GPU-CPU coordination, as CPU resources are heavily utilized for orchestration, RAG, and state management, while GPUs handle reasoning.

VCF optimizes this mix by allowing flexible sizing of VM CPU and memory resources based on agent orchestration intensity. VCF's ability to colocate GPU-accelerated and CPU-intensive applications on the same servers offers superior flexibility compared to bare-metal or separate systems, reducing deployment and operational costs for next-generation agentic systems. In this chapter, we demonstrate that colocating AI and non-AI workloads does not have any measurable performance impact on either workload.

Figure 16: Results Running a Database Workload Concurrently with LLM Inference on the Same Server



Both HammerBD TPC-C derived workload and Llama3 70B inference co-exist on a single server with no measurable impact on each other.

In Figure 16, for LLM inference, the input sequence length (ISL), output sequence length (OSL), and batch size were chosen so that it would run at the highest token generation rate with tensor parallelism of 4 on 4X H200 GPUs. The HammerDB workload runs at 1.8 million TPM, which represents a heavy enterprise-level load. The database server VM runs at 75% utilization on 12 vCPUs. The LLM-inference VM uses 1 vCPU. The results show that neither workload experiences a measurable performance impact.

The left panel in Figure 16 compares the normalized TPM for the HammerDB TPC-C derived workload running by itself to the normalized TPM, while running concurrently with LLM inference (using the Llama3 70B model compiled using NVIDIA TRT-LLM with tensor parallelism of 4). We can clearly see that the database workload shows no measurable impact due to sharing the server with LLM inference.

The right panel in Figure 16 compares the normalized throughput for LLM inference alone (using the same model as above) with the throughput for LLM inference while sharing the server with the database application. The data demonstrates that LLM inference suffers no measurable performance impact due to sharing the server with the database workload.

These results show that enterprises can run multiple workloads on the same server with no performance impact, obtaining much higher resource utilization than if the workloads ran on separate servers. The CPU-based workloads can be traditional non-AI workloads, like the databases in this experiment, or they can be CPU-based AI workloads, including agentic AI.

Because many enterprises run database workloads that are accessed by numerous agents, the HammerDB TPC-C derived workload was selected as the benchmarking tool. For the LLM-inference workload, Llama3 70B was chosen, compiled with NVIDIA TRT-LLM with tensor parallelism of 4. The configuration is shown in Table 10.

Table 10: Hardware and Software Configuration for Workload Consolidation in VCF 9.1

Parameter	Configuration
Server	PowerEdge XE774, AMD EPYC 9555 64-core processor, 256 total logical cores, 1.5-TB memory
GPU	4x NVIDIA H200 GPUs with NVLink
ESX	9.1.0
NVIDIA Driver	580.105
Workload	HammerDB TPC-C derived + Llama3 70B inference
VM Configuration	DB-Server OS: Oracle Linux 8, 12 vCPUs, 48-GB DRAM
	DB-Client OS: Oracle Linux 8, 12 vCPUs, 48-GB DRAM
	LLM OS: Ubuntu 24.04, 96 vCPUs, 4x vGPUs, 96c profile

Chapter 6: Explore Time Sharing to Optimize Inference Throughput

This chapter explores how NVIDIA vGPU on VCF optimizes GPU utilization using cycle stealing. Unlike traditional partitioning, this approach allows virtualized workloads to outperform bare-metal configurations by intelligently sharing compute resources.

6.1 Key Mechanisms of Time-Shared vGPU vs. MIG

The paper addresses two primary ways of sharing GPU resources:

- **Time-sliced vGPU (referred to as vGPU in this paper):** Statically partitions HBM (memory) but time shares the compute cores. This allows a workload to access maximum compute power during its turn, while keeping memory isolated.
- **MIG vGPU (NVIDIA multi-instance GPU (MIG) with vGPU):** Statically partitions both memory and compute cores, creating rigid boundaries that prevent one workload from using another's idle cycles.

6.2 The Layout B Strategy: Maximize Throughput

The core innovation, shown in Figure 18, lies in spreading a workload's vGPU profiles across more physical GPUs than strictly necessary for memory requirements.

Table 11: Two Layout Strategies

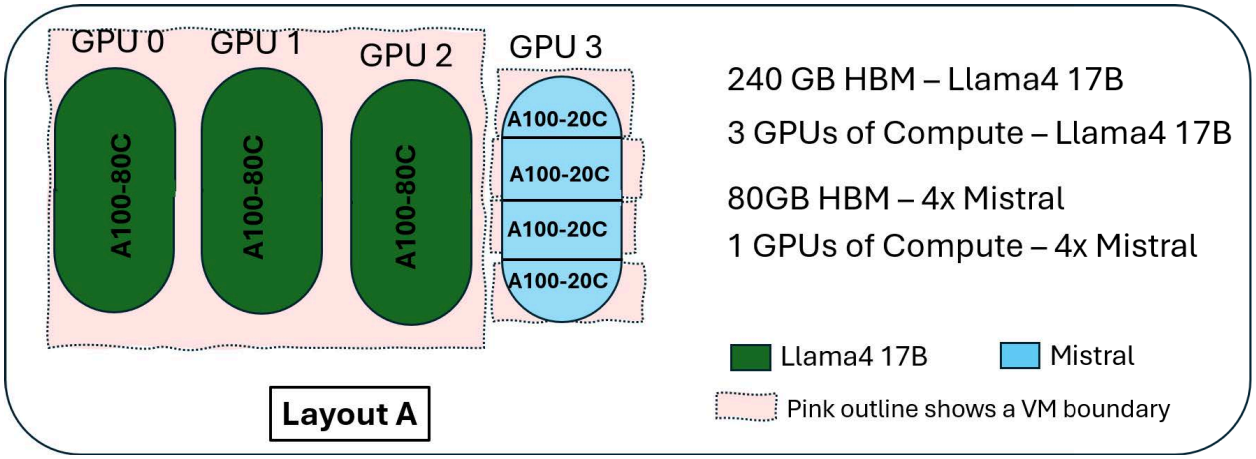
Strategy	Resource Allocation	Result
Layout A (Traditional)	Uses 3 vGPUs on 3 physical GPUs.	Baseline performance based on 3-GPU compute.
Layout B (Optimal)	Spreads the same memory over 4 physical GPUs.	Accesses compute cores from all 4 GPUs, increasing throughput.

This strategy is particularly effective for embarrassingly parallel workloads like:

- LLM inference: Llama 3/4, Mistral
- Financial analytics: High-performance computing (HPC) tasks, like computing the Greeks.

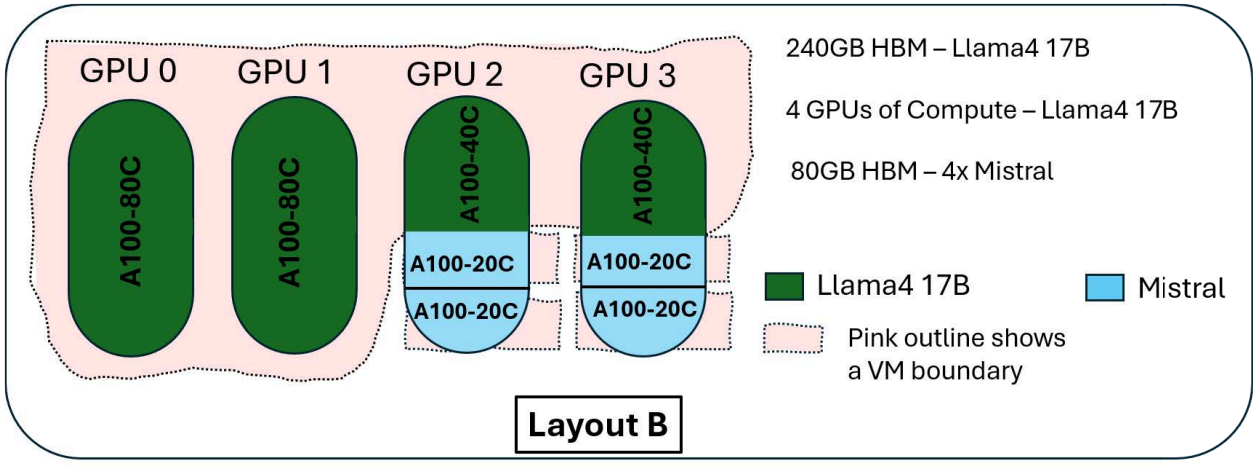
6.3 The Time-Sharing Advantage

Figure 17: Layout A



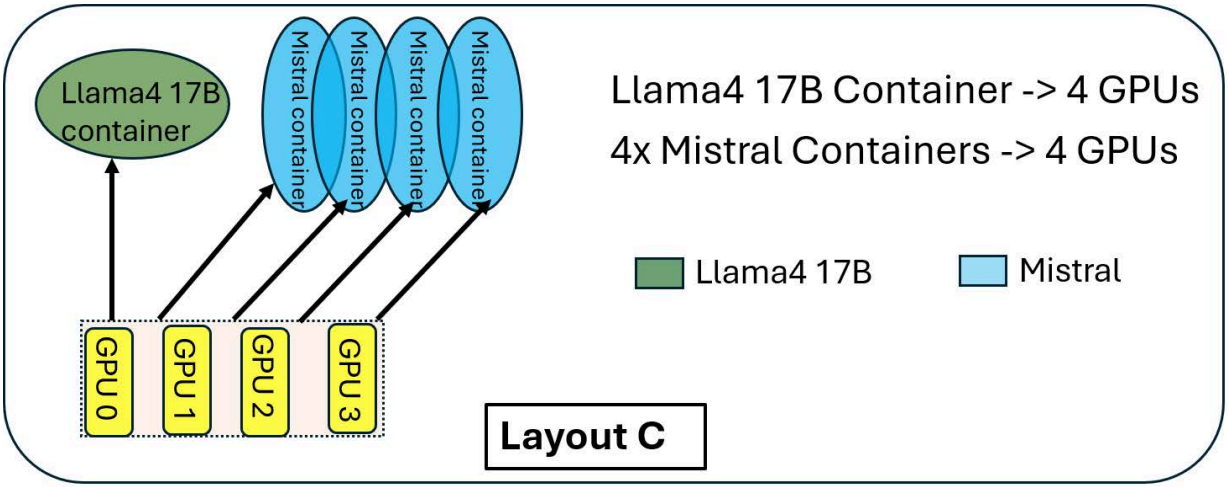
Layout A, as shown in Figure 17, is a conventional architectural arrangement of virtualized workloads deployed on a server equipped with four physical GPUs. In this specific scenario, the environment hosts five distinct virtual machines: a single VM executing the Llama4 17B model and four VMs supporting Mistral 7B. This configuration accounts for an aggregate GPU memory footprint of 240 GB, where the Llama4 17B workload is provisioned with three dedicated physical GPUs, leaving the fourth GPU to be shared among the Mistral instances.

Figure 18: Layout B



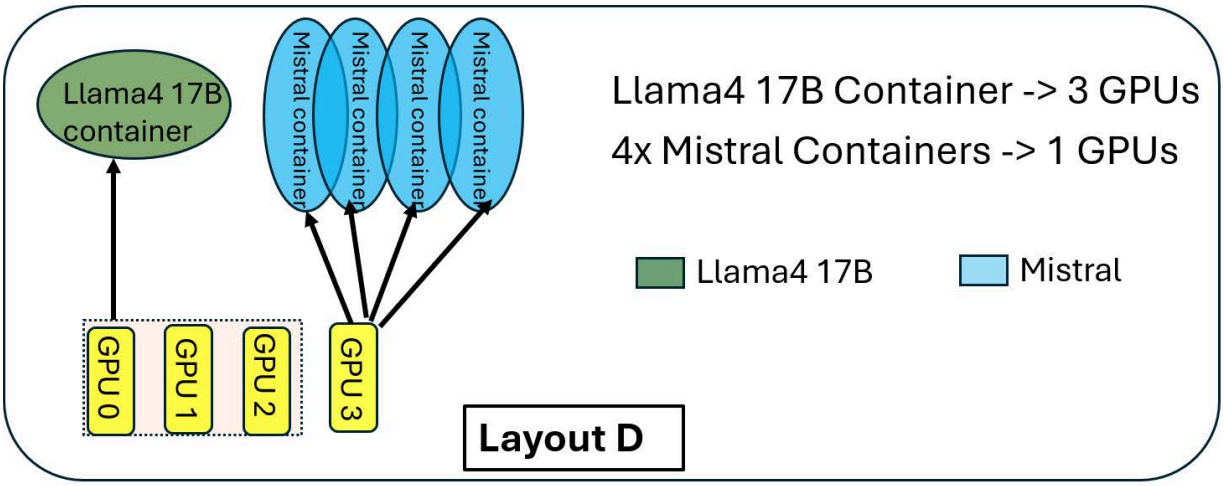
Layout B, as shown in Figure 18, is an innovative architectural strategy designed to leverage the benefits of computational time sharing. By distributing the Llama4 17B vGPU profiles across all four physical GPUs and spreading the Mistral workload over two, this configuration facilitates compute cycle stealing. This mechanism enables a workload to harness idle processing cycles from concurrent tasks on the same GPU, driving higher aggregate throughput compared to traditional setups. Because NVIDIA vGPU implements a round-robin time-slicing algorithm, every virtual machine is assured a baseline resource allotment, preventing starvation while allowing elastic access to surplus compute capacity when other workloads are inactive.

Figure 19: Layout C



Layout C, as shown in Figure 19, is a configuration where a Llama4 17B instance and four Mistral containers are deployed on a host with 4x NVIDIA A100-80 GPUs. Within this architectural arrangement, every container is granted access to the full cluster of four GPUs, facilitating a shared resource environment across all active workloads.

Figure 20: Layout D



Layout D, as shown in Figure 20, is a configuration where a Llama4 17B instance is deployed in a container that is assigned 3x NVIDIA A100-80 GPUs and four Mistral containers share one NVIDIA A100-80 GPU. By placing vGPU profiles so they span all available hardware, multiple workloads can leverage idle compute cycles from one another.

- **How It Works:** When one model (e.g., Llama 4) isn't fully utilizing a GPU's compute cores, another model (e.g., Mistral) can seamlessly step in and use those cycles.
- **The Result:** This approach effectively turns unused overhead into a performance gain, allowing virtualized environments to exceed the efficiency of traditional physical servers. In a fully-loaded scenario, the vGPU configuration achieved 30% higher aggregate throughput than a bare-metal setup on the exact same hardware.
- **The Caveat:** This works only if workloads aren't already 100% optimized to use every available cycle; if there are no idle cycles to steal, the advantage disappears.

6.4 Experiment One: Unoptimized Workloads (HuggingFace)

To demonstrate the throughput advantage of time-sharing, four experiments were run using Llama4 17B and Mistral 7B models running concurrently. The hardware consisted of a PowerEdge XE8545 server with 2x AMD EPYC processors and 4x NVIDIA A100-80 GPUs. The detailed configuration is shown in Table 12.

For each experiment, five configurations were tested:

1. Multi VM (Layout B): Exploits time sharing via vGPU
2. Bare metal (Layout C): Containers share the GPUs on standard Ubuntu
3. Single VM (Layout C): Containers share the GPUs inside a single VM
4. Bare metal (Layout D): Containers share the GPUs on standard Ubuntu
5. Single VM (Layout D): Containers share the GPUs inside a single VM

The detailed configuration is shown in Table 13.

Table 12: Experiment One Configurations: NVIDIA vGPU on VCF with Bare Metal Using Llama4 17B and Mistral Models from HuggingFace

Parameter	System Configuration
Server	PowerEdge XE8545, AMD EPYC 7543 32-core processor, 2 Socket, 4x NVIDIA A100-80 GPUs
ESX	9.0.0
NVIDIA Driver on VCF	580.126
NVIDIA Driver on Bare Metal	580.105

Table 13: Workload Configurations Used for Experiment One

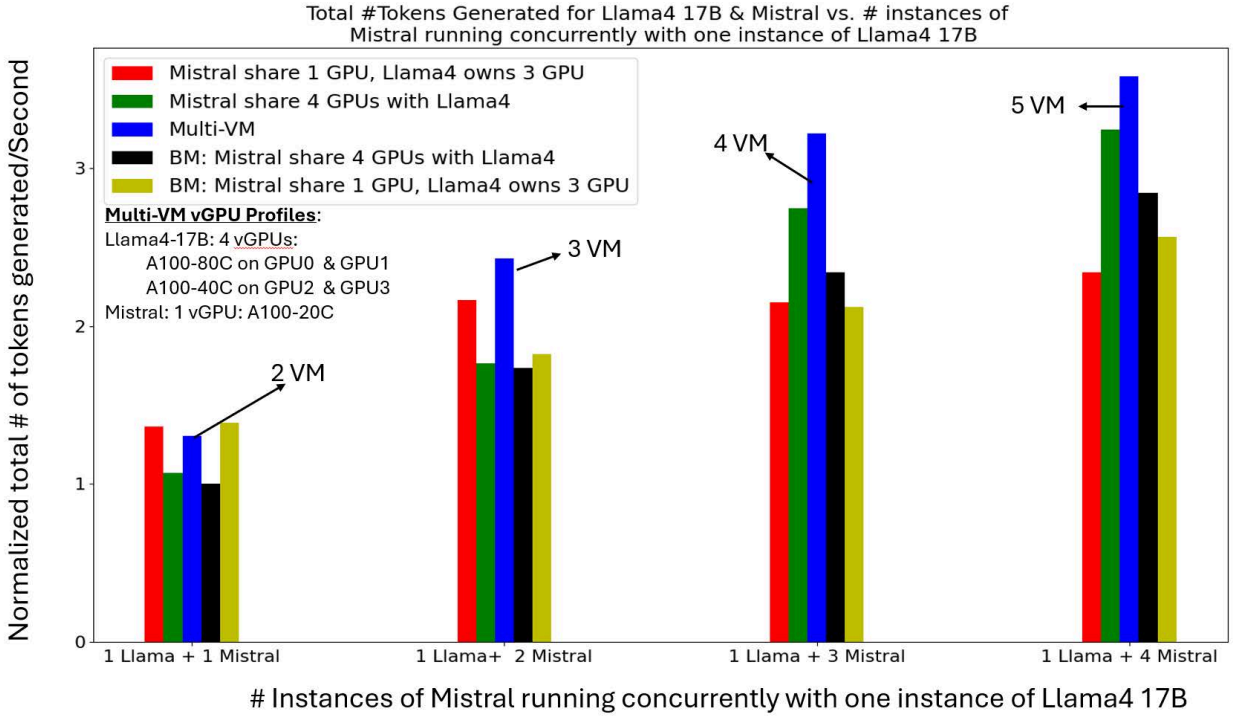
Bar in Figure 21	VCF or Bare Metal	VMs	vGPUs/ Llama	vGPUs/ Mistral	GPUs/ Llama	GPUs/ Mistral	GPU Sharing Mechanism
Workload 1: 1 Llama4 + 1 Mistral							
Red	VCF	1	3	1	—	—	GPU assignment at container level
Green		1	4	4	—	—	All containers access all GPUs
Blue		2	4	1	—	—	GPU sharing by vGPU
Black	Bare metal Ubuntu 22.04	0	—	—	4	4	All containers access all GPUs
Yellow		0	—	—	3	1	GPU assignment at container level
Workload 2: 1 Llama4 + 2 Mistral							
Red	VCF	1	3	1	—	—	GPU assignment at container level
Green		1	4	4	—	—	All containers access all GPUs
Blue		3	4	1	—	—	GPU sharing by vGPU
Black	Bare metal Ubuntu 22.04	0	—	—	4	4	All containers access all GPUs
Yellow		0	—	—	3	1	GPU assignment at container level
Workload 3: 1 Llama4 + 3 Mistral							
Red	VCF	1	3	1	—	—	GPU assignment at container level
Green		1	4	4	—	—	All containers access all GPUs
Blue		4	4	1	—	—	GPU sharing by vGPU
Black	Bare metal Ubuntu 22.04	0	—	—	4	4	All containers access all GPUs
Yellow		0	—	—	3	1	GPU assignment at container level

Table 13: Workload Configurations Used for Experiment One (Continued)

Bar in Figure 21	VCF or Bare Metal	VMs	vGPUs/ Llama	vGPUs/ Mistral	GPUs/ Llama	GPUs/ Mistral	GPU Sharing Mechanism
Workload 4: 1 Llama4 + 4 Mistral							
Red	VCF	1	3	1	—	—	GPU assignment at container level
Green		1	4	4	—	—	All containers access all GPUs
Blue		5	4	1	—	—	GPU sharing by vGPU
Black	Bare metal Ubuntu 22.04	0	—	—	4	4	All containers access all GPUs
Yellow		0	—	—	3	1	GPU assignment at container level

Results are shown in Figure 21. The multi-VM configuration (Layout B) utilizing NVIDIA vGPU showed the highest aggregate throughput. By optimizing the use of compute cycles, this time-shared configuration achieved about a 30% gain in aggregate throughput over the bare-metal configuration.

Figure 21: Normalized Throughput for Workloads Run Five Different Ways to Configure GPU Resources on the Server



In Figure 21, the Llama4 17B and Mistral models were downloaded from the HuggingFace repository and used as is, with no changes of any kind.

6.5 Experiment Two: Optimized Workloads (NVIDIA TRT-LLM)

In a second set of experiments, a Llama3 70B model was compiled with NVIDIA TRT-LLM. The server has four A100-80 GPUs, so several configurations of tensor parallelism and pipeline parallelism are possible. The experiment included different configurations for tensor and pipeline parallelism with NVIDIA vGPU on VCF and on bare metal, as shown in Table 14. The results from these experiments are shown in Figure 22. Clearly, the configuration that runs two instances of Llama3 70B using pipeline parallelism of two with two separate VMs shows the highest throughput, beating the best bare-metal throughput by about 10%.

Four configurations were tested, utilizing the exact same hardware resources:

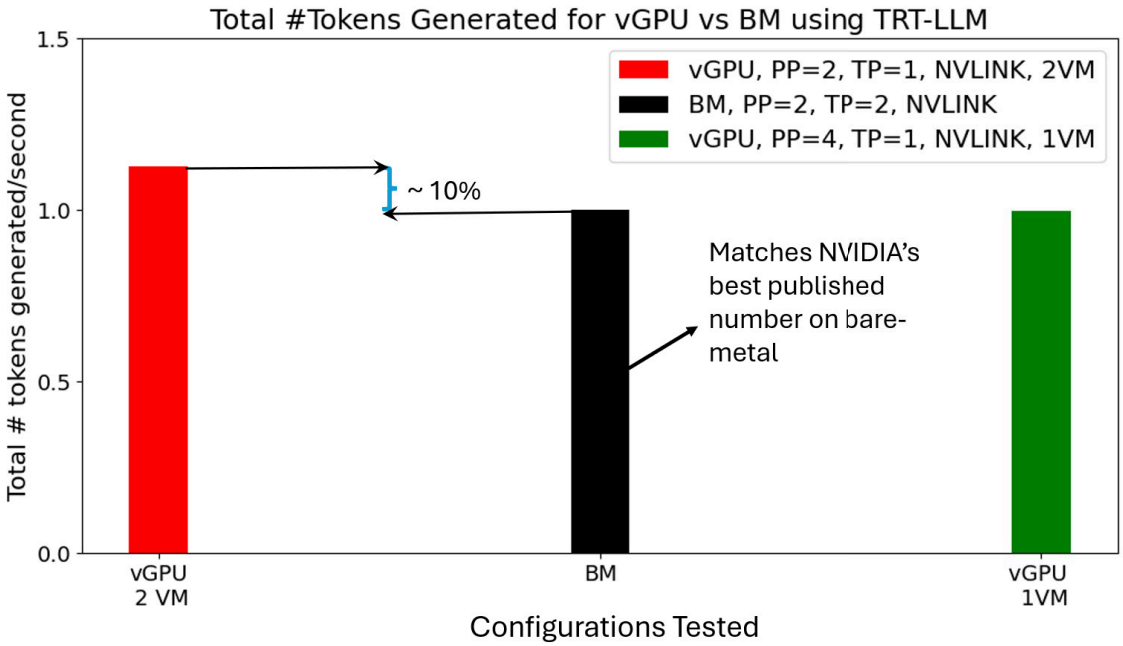
1. Multi VM: Two VMs, each running an instance of Llama3 70B with pipeline parallelism of 2; each instance spans two physical GPUs.
2. Bare metal (BM): One instance with tensor parallelism of 2 and pipeline parallelism of 2, spreading the model over all four GPUs.
3. Bare metal (BM): One instance with pipeline parallelism of 4, spreading the model over all four GPUs.
4. Single VM: One instance with pipeline parallelism of 4, spreading the model over all four GPUs.

Table 14: Experiment Two Configurations: Llama3 70B Model (FP16 Format) Compiled with NVIDIA TRT-LLM

Bar in Figure 22	VCF or Bare Metal	VMs	vGPUs	GPUs	Instances of Model on Server	Tensor Parallelism	Pipeline Parallelism	NVLink
Red	VCF	2	2X A100-80C	—	2	1	2	Yes
Green		1	4X A100-80C	—	1	1	4	Yes
Black	Bare metal Ubuntu 22.04	—	—	4X A100-80C	1	2	2	Yes

Results are shown in Figure 22.The configuration running two instances of Llama3 70B across two separate VMs showed the highest throughput, beating the best bare-metal throughput by about 10%. When NVIDIA TRT-LLM was used to compile the model, the resultant version was more optimized than the original version from HuggingFace. This implies that the workload had fewer unused computation cycles to steal, and therefore the benefit of cycle stealing that NVIDIA vGPU on VCF achieves was lower than if the model from HuggingFace were used without compilation.

Figure 22: Normalized Throughput Using Different Configurations to run Llama3 70B (FP16 Format) with NVIDIA TRT-LLM on a Server with 4x A100-80 GPUs



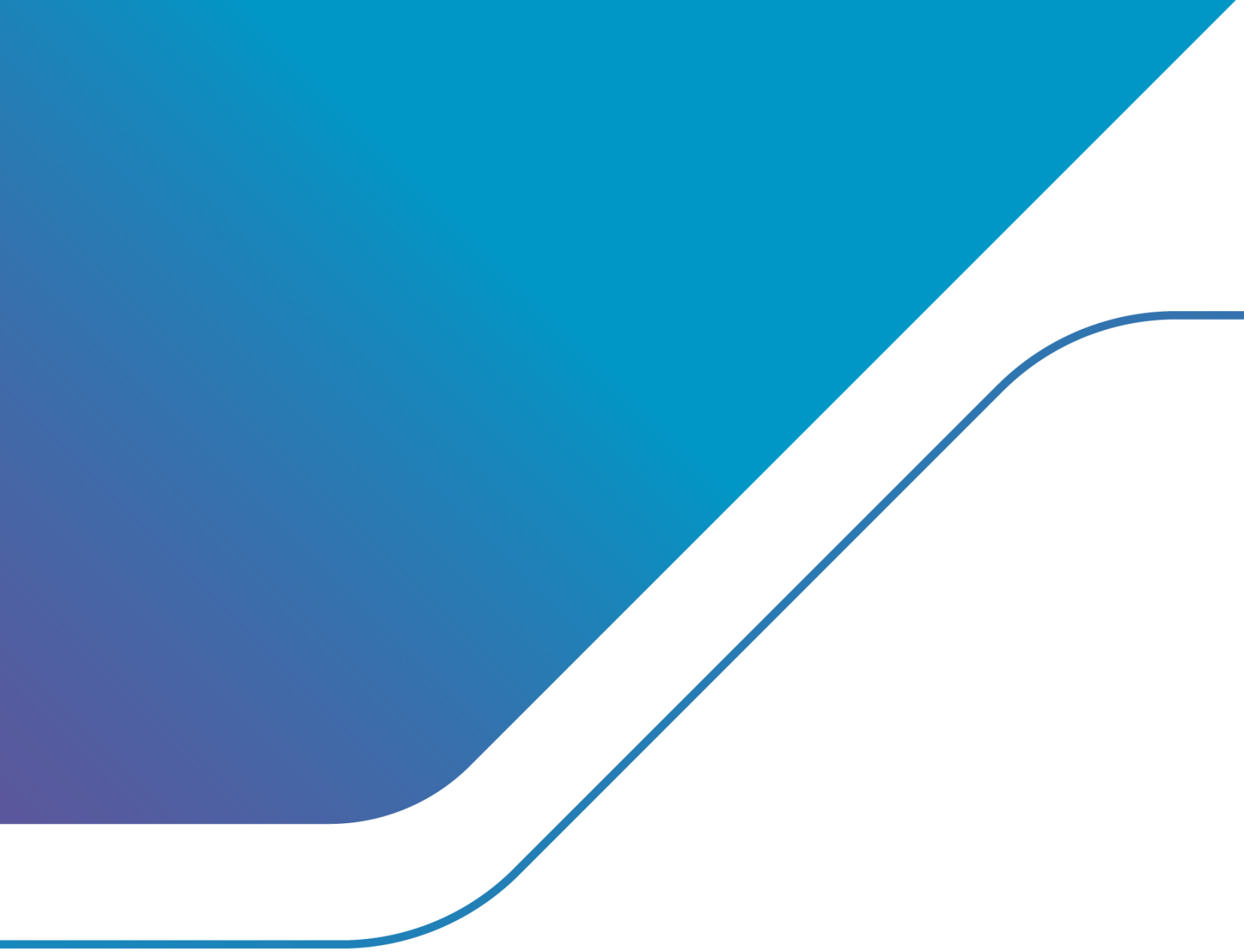
As shown in Figure 22, the configuration with two VMs on the server with each VM using pipeline-parallelism of 2 over two GPUs had the highest throughput, exceeding the best throughput on bare metal by about 10%.

Chapter 7: Conclusion

VCF transcends virtualization to maximize ROI and reduce TCO for enterprise AI private cloud management by allowing enterprises to consolidate more workloads/VMs per server while maintaining the strictest isolation, security, and flexibility of workload deployment. This paper demonstrates various use cases where GPU-intensive and CPU-intensive workloads can be consolidated into fewer servers to increase the efficiency of AI infrastructure by maximizing the utilization of CPU, GPU, and system memory. Use cases show how to choose between vGPU and MIG-vGPU for optimal performance of AI applications.

In addition, VCF is a highly flexible, efficient, and secure platform for managing private AI within multi-tenant enterprise environments. By giving IT administrators the ability to choose between VM-based isolation for strict security requirements and container-based isolation within a shared VM for simplified management, VCF addresses diverse operational and infrastructure needs. The experimental findings clearly demonstrate that consolidating multiple lightweight, underutilized workloads onto shared hardware addresses the common issue of resource underutilization while delivering virtually no performance penalty whether workloads were distributed across separate VMs or aggregated into a single VM. Ultimately, VCF empowers enterprises to optimize their AI cloud architectures, ensuring high availability, robust security, and maximum GPU efficiency. This helps enterprises optimize the cost of their AI infrastructure.

Please reach out to the VMware Performance Engineering team at vcf.ai.performance.pdl@broadcom.com with any questions regarding the performance of AI workloads.



vmware®
by Broadcom