Optimizing & Scaling Machine Learning Workloads in VMware vSphere 8 with VMware Tanzu & NVIDIA AI Enterprise

Performance Study - September 22, 2023

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com

Table of Contents

Executive Summary

VMware vSphere with Tanzu Kubernetes Grid and NVIDIA AI Enterprise provides IT admins with multiple conveniences in deploying and managing ML/AI workloads at a large scale. In this paper, we showcase these benefits and provide guidance on optimizing the performance of ML/AI in vSphere with Tanzu. We list key takeaways at the end of this paper.

Introduction

Deploying machine learning/artificial intelligence (ML/AI) solutions in enterprises is challenging because it requires integrating many different hardware and software components. The increasing demand for delivering the latest advanced technologies in ML/AI makes ML infrastructure in enterprises increasingly complex. VMware vSphere seamlessly integrates Tanzu capabilities (including Tanzu Kubernetes Grid) with NVIDIA AI Enterprise, as shown in figure 1, and helps organizations save time and money on building ML infrastructure. In this paper, we provide best practices for ML/AL workload deployment in Tanzu with NVIDIA AI Enterprise and show you how to optimize and scale ML applications in vSphere.

When adopting cutting-edge machine learning and artificial intelligence (ML/AI) solutions, it can be challenging to get some enterprise IT systems to work together, and this introduces administrative complexity as more and different solutions must be integrated.

VMware and NVIDIA have joined forces to provide a fast, virtualized, and easy-to-integrate enterprise system for Kubernetes. The VMware vSphere platform facilitates the smooth incorporation of Tanzu functionalities, such as Tanzu Kubernetes Grid, in conjunction with NVIDIA AI Enterprise, as depicted in figure 1. This integration enables organizations to optimize resource utilization by reducing the effort and expenses associated with constructing machine learning infrastructure.

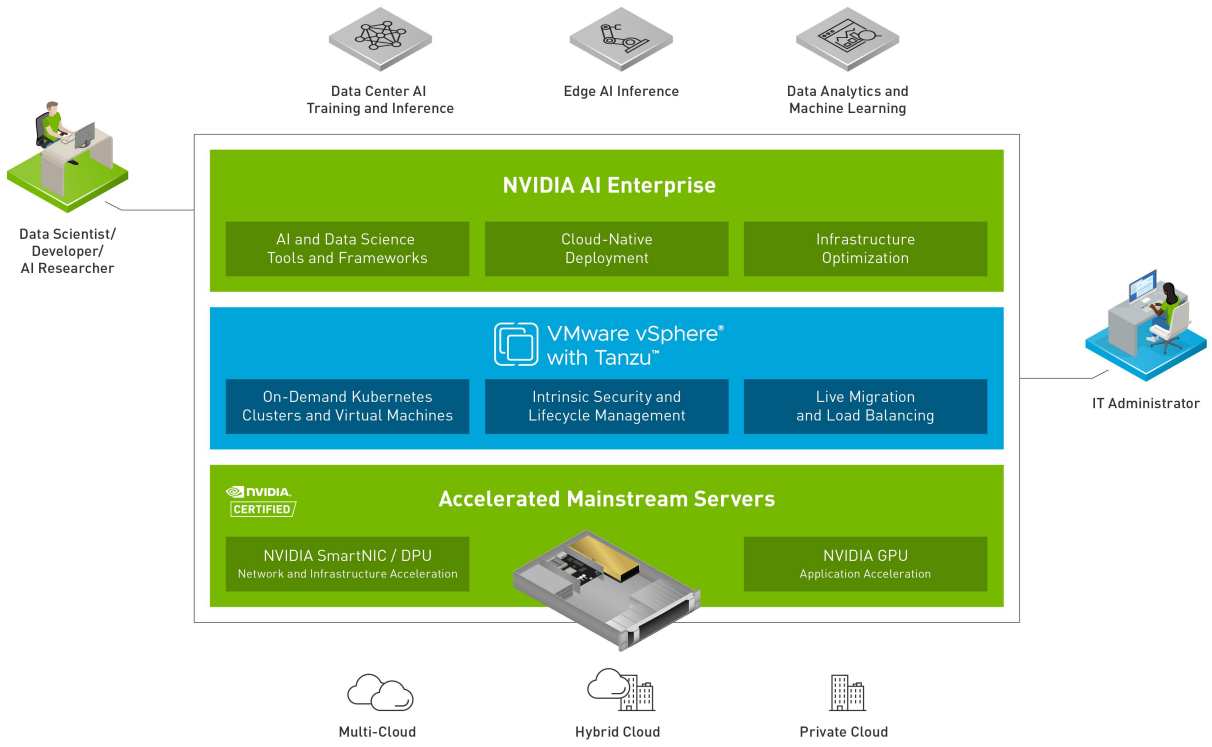

Figure 1. VMware vSphere with Tanzu & NVIDIA AI Enterprise

Deploying Machine Learning in vSphere with Tanzu & NVIDIA AI Enterprise

One of the important components provided in VMware Tanzu is the Tanzu Kubernetes Grid (TKG). TKG allows the orchestration and management of containerized workloads with Kubernetes without installing Kubernetes because it is integrated with Tanzu. For ML/AI workloads, Kubernetes can help automate the deployment, scaling, and management of ML workloads, making it easier for data scientists and developers to build, train, and deploy ML models. With TKG and NVIDIA AI Enterprise in vSphere, AI professionals and other users can deploy and run GPU-enabled workloads without needing to care much about the underlying cloud infrastructure. This simplicity can accelerate AI adoption.

In Tanzu, to deploy the GPU-enabled workloads with NVIDIA AI Enterprise, we need to install NVIDIA GPU Operator after deploying the TKG cluster. GPU Operator automates the management of all NVIDIA software components required to provision GPUs or vGPUs, as shown in figure 2. These components include the NVIDIA drivers (to enable CUDA), a Kubernetes device plugin for GPUs, the NVIDIA Container Toolkit, automatic node labeling using GPU Feature Discovery (GFD), Data Center GPU Manager (DCGM), and others. NVIDIA GPU Operator reduces the effort needed to enable NVIDIA AI Enterprise with Kubernetes.

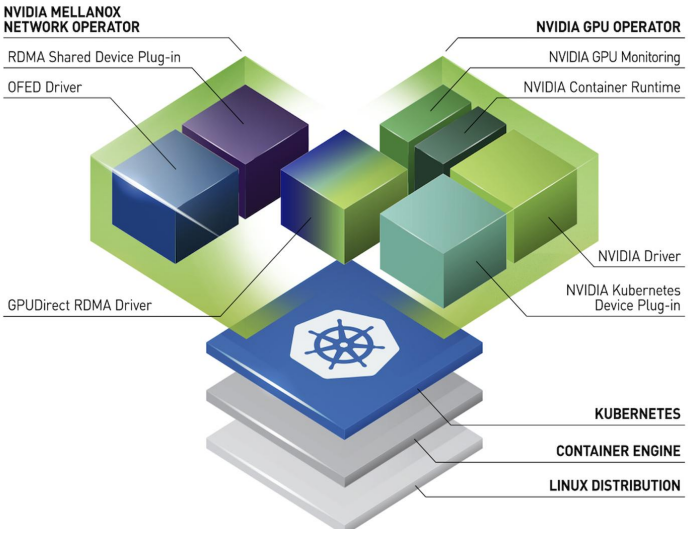

Figure 2. NVIDIA GPU Operator Architecture

Performance of ML/AI Workloads in vSphere with Tanzu & NVIDIA AI Enterprise

VMware vSphere 8 U1 and U2 with VMware Tanzu support the latest GPUs (like H100, A100, L40, etc.). It delivers good ML/AI workload performance while enabling the best-in-class Kubernetes environment for running and scaling these workloads. To demonstrate, we tested the performance of several popular ML/AI workloads—including MaskRCNN, BERT, and RNNT—in Tanzu with TKG and compared results with their performance when running in a non-Tanzu environment (which does not use TKG and GPU Operator)

Testbed Setup

Our testbed setup had the following hardware and software:

- A cluster of 3 Dell PowerEdge R7525 servers, each with dual AMD EPYC 7763 64-Core Processors, and 128 CPU cores with 512 GB of memory, 20TB vSAN storage, and Broadcom NetXtreme 25Gb Ethernet Adapters
- Each server had one NVIDIA A100 40GB Ampere architecture–based1 GPU.
- NVIDIA AI Enterprise software version 3.1, NVIDIA vGPU 15.2
- ESXi 8 U1 and ESXi 7 U3 with VSAN storage

[1] For more information about the NVIDIA A100 Tensor Core GPU architecture, see https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf

- Kubernetes node deployed as virtual machine (VM) running Ubuntu 20.04, 32GB RAM, 8 vCPUs

We ran the following ML/AI workloads:

- MaskRCNN: object detection model based on deep convolutional neural networks (CNN)
- RNNT: speech-to-text application
- BERT: language processing application

We used the NVIDIA vGPU best-effort scheduler for the vGPU profile. The best-effort scheduler shares GPU resources using a round-robin scheduling algorithm that can optimize the utilization of resources. The best-effort scheduling policy best utilizes the GPU during idle and not fully utilized times, allowing for optimized density and a good quality of service (QoS).

Zero Overhead of ML/AI Performance

The ML/AI workloads were tested with the two following scenarios: using a 40c vGPU profile with 40GB of GPU memory per vGPU.

1. **With Tanzu:** Workloads were deployed with Tanzu TKG and NVIDA GPU Operator.
2. **Without Tanzu:** Workloads were deployed in the traditional way of NVIDIA AI Enterprise without Kubernetes (Tanzu Kubernetes Cluster) or NVIDIA GPU Operator.

Figure 3. Inference throughput with and without Tanzu

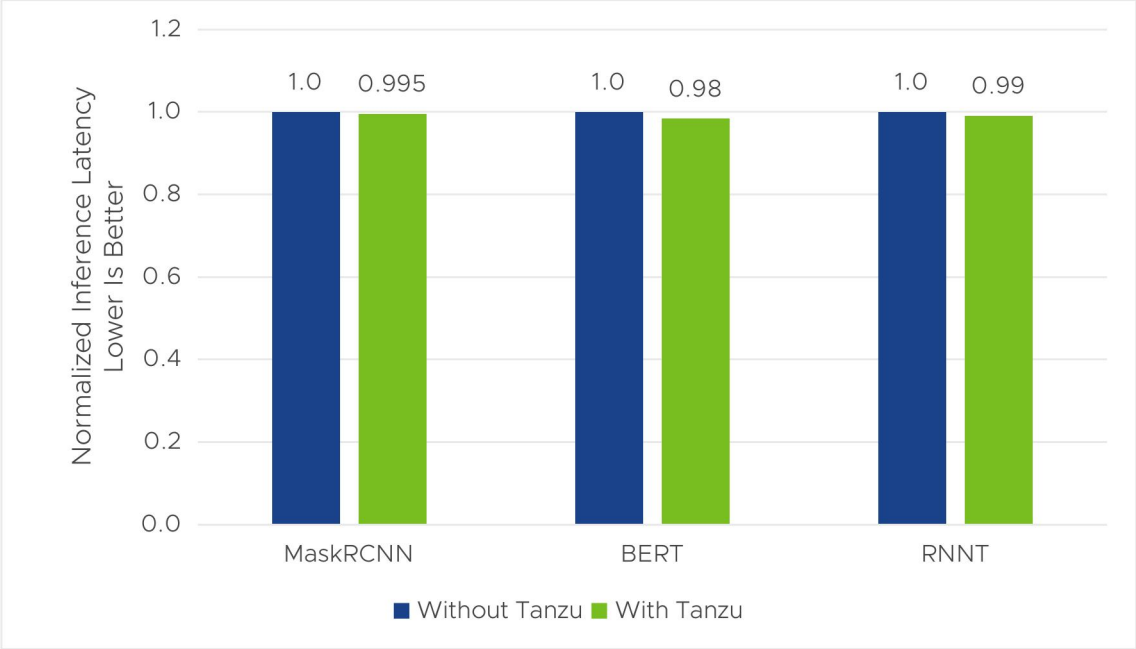

Figure 4. Inference latency with and without Tanzu TKG

The results in figures 3 and 4 show that the inference throughput and latency of MaskRCNN, BERT, and RNNT with Tanzu TKG show almost no difference or slightly better (1%-2%) when compared to the ones without Tanzu. The results are normalized to those without Tanzu TKG. In vSphere 8 U1 with Tanzu TKG, we used the default image of Ubuntu 20.04 provided by VMware. This reduced the effort of preparing VMs to run the workloads and provided optimized performance for the ML/AI applications.

Improvement of ML/AI Performance with vSphere 8 U1

VMware vSphere 8 U1 with NVIDIA AI Enterprise delivers better performance for ML/AI workloads when compared to vSphere 7 U3. Figures 5 and 6 show a 6%-7% performance improvement of vSphere 8 U1 over vSphere 7 U3 in training time and throughput of MaskRCNN.

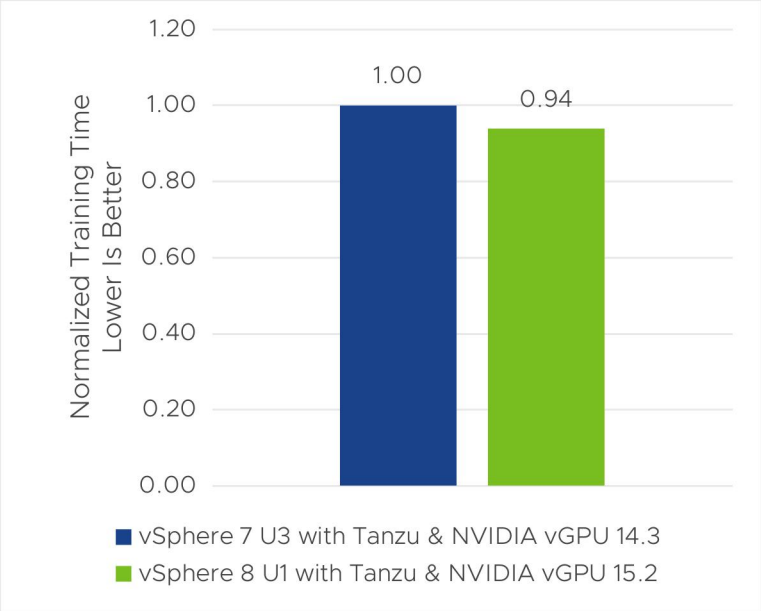

Figure 5. Training time of MaskRCNN in vSphere 8 U1 vs vSphere 7 U3 with Tanzu

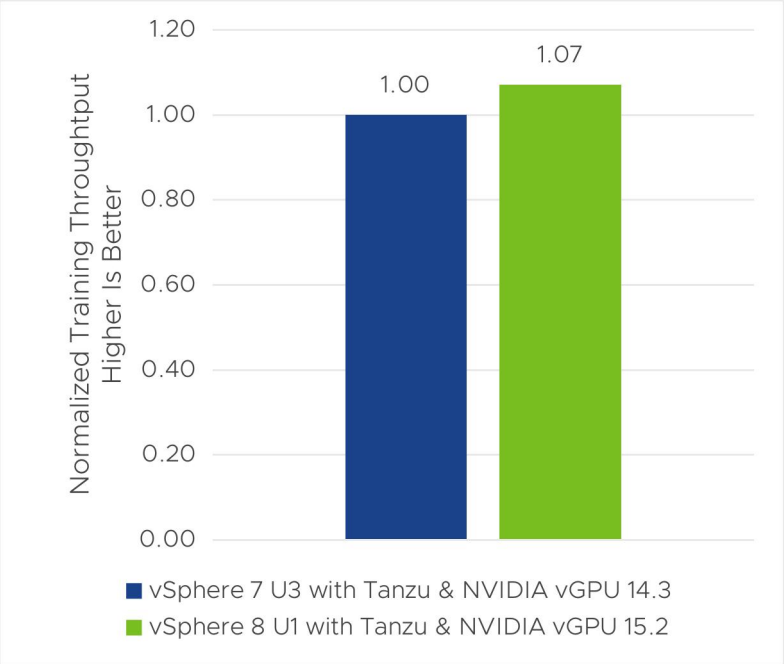

Figure 6. Training throughput of MaskRCNN in vSphere 8 U1 vs vSphere 7 U3 with Tanzu.

For the inference of MaskRCNN, we also see a 2% improvement in inference time and throughput, as shown in figures 7 and 8.

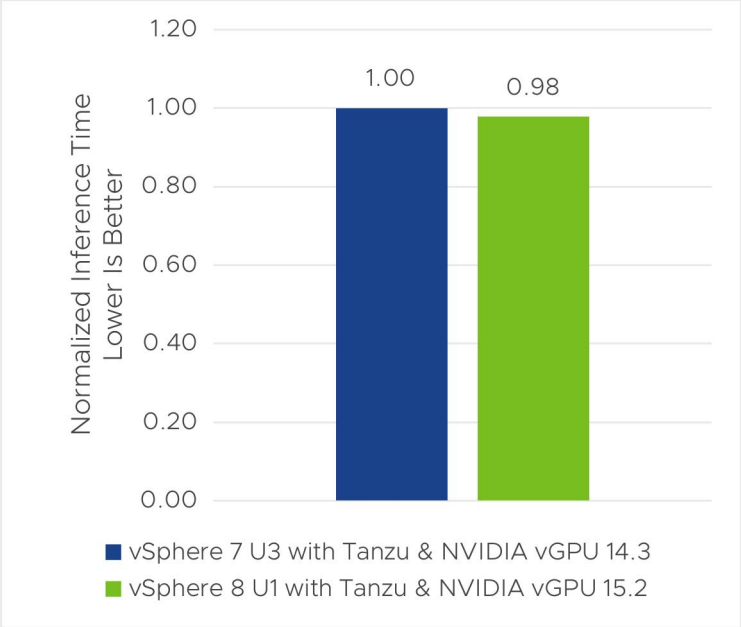

Figure 7. Inference time of MaskRCNN in vSphere 8 U1 vs vSphere 7 U3 with Tanzu

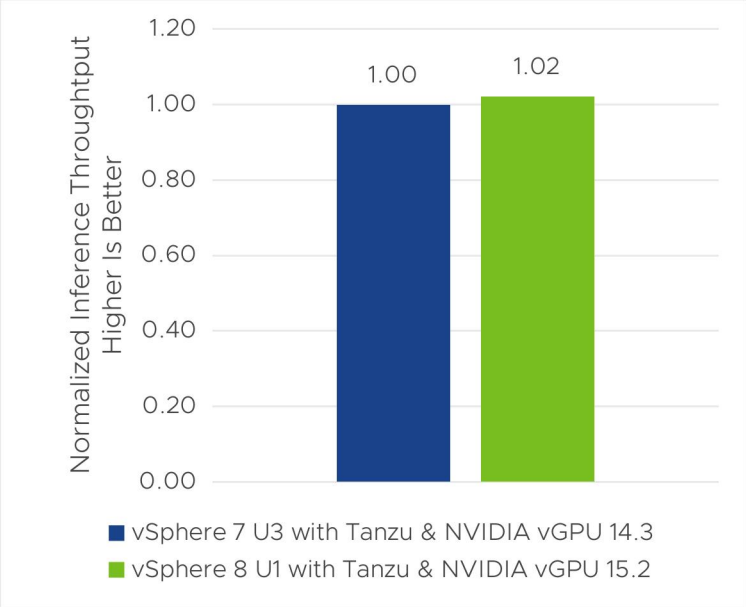

Figure 8. Inference throughput of MaskRCNN in vSphere 8 U1 vs vSphere 7 U3 with Tanzu

The performance gain comes from using the newer versions of both the vSphere and NVIDIA vGPU drivers. For the test cases in vSphere 8 U1, we used NVIDIA vGPU 15.2 with vGPU Operator 22.1.9, and for the test cases in vSphere 7 U3, we used NVIDIA vGPU 14.3.

Optimizing ML/AI Workload Performance

NVIDIA AI Enterprise offers two options to share GPUs among multiple VMs in VMware vSphere. You can use NVIDIA vGPU with a time-sliced scheduler or with multi-instance GPU (MIG) technology. Figure 9 depicts the difference between vGPU and MIG vGPU. With vGPU, CUDA cores of the GPU are shared among VMs using time slicing, while memory is equally partitioned among VMs. For MIG vGPU, currently, the GPU can be partitioned into as many as seven GPU instances, fully isolated at the hardware level with their own high-bandwidth memory, cache, and compute cores, and then statically partitioned among VMs as separate vGPUs. With TKG, a Kubernetes node is deployed as a VM, which can be assigned either a vGPU or MIG vGPU.

With these two options, which one should you choose for your ML/AI workload? In this section, we conducted experiments that involved Mask R-CNN training and inference in two scenarios (shown in table 1) that mimic the light and heavy loads of ML applications. The results of these experiments in figures 10–13 provide some guidelines for selecting a suitable option for your workload.

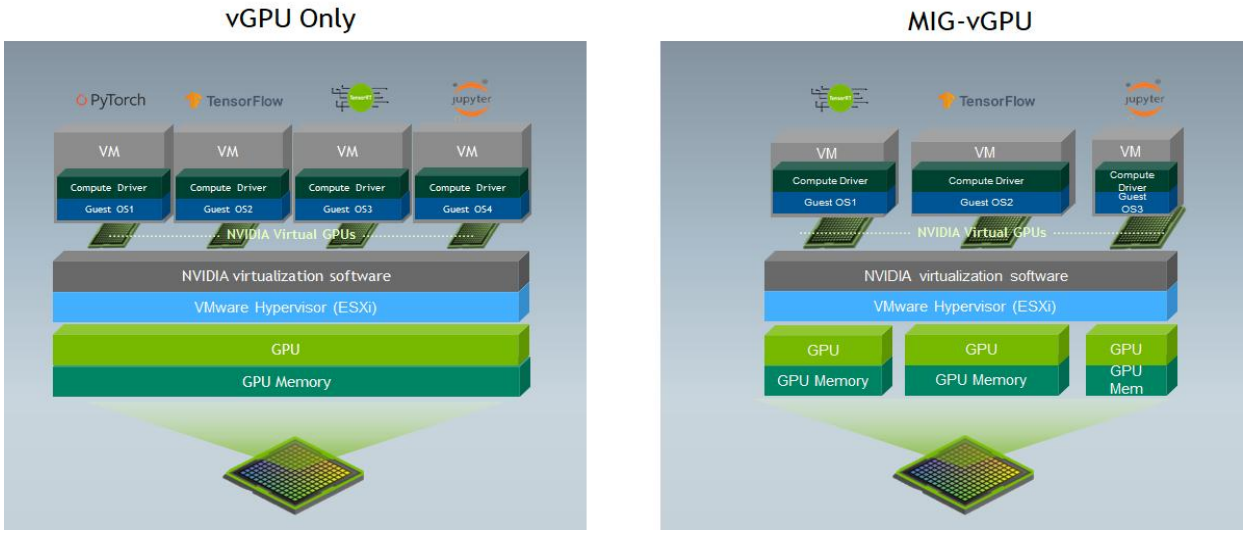

Figure 9. Multiple VMs (also Kubernetes node in TKG) sharing GPUs using vGPU or MIG vGPU

	Light Load	Heavy Load
Workload	Mask R-CNN	Mask R-CNN
Batch Size	2	8
Type of vGPU/MIG profile	5c or 1-5c	40c or 7-40c
Number of VMs (Kubernetes nodes)	4	1

Table 1. Test scenarios of sizing the ML workloads

In general, if your workload is light and does not fully utilize GPU resources, especially the compute resource, MIG vGPU will be a better option. Because compute cores are partitioned and independently executed among MIG vGPUs, MIG helps multiple VMs (Kubernetes nodes in a TKG cluster) that are running light workloads (like a small ML model or a model with a smaller batch size) to maximize the utilization of all available CUDA cores. vGPU uses time-sliced scheduling for GPU sharing and can have a large portion of CUDA cores underutilized when the workload is very light, and this makes it not work as well as MIG in GPU sharing for light workloads.

With heavy loads where you need one vGPU per GPU, vGPU gives better performance than MIG vGPU for the tested workload. When the workload has unknown characteristics, it is better to test both to select the right one. Please note that, when multiple full vGPUs are assigned to a VM for training a larger model, the time-slicing mode is required.

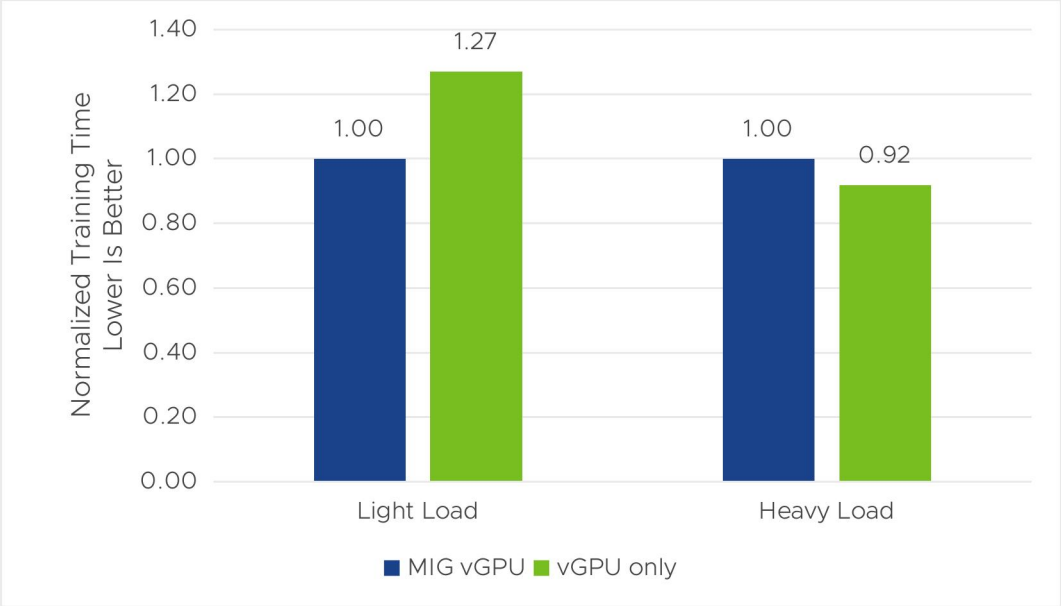

Figure 10. Training time of MIG vGPU vs. vGPU only in vSphere 8 U1 with Tanzu

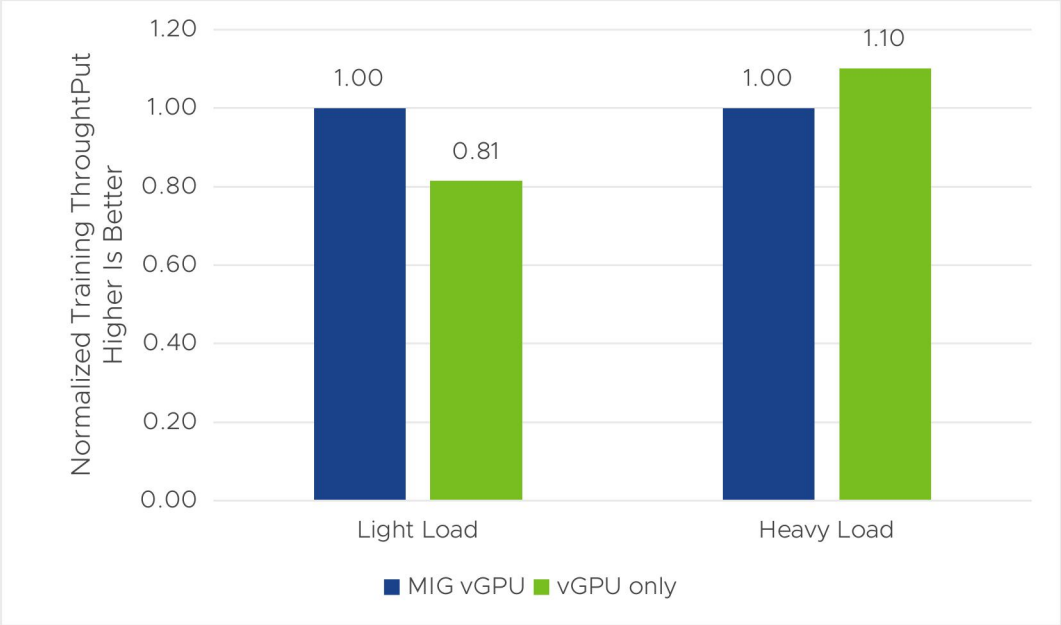

Figure 11. Training throughput of MIG vGPU vs. vGPU only in vSphere 8 U1 with Tanzu

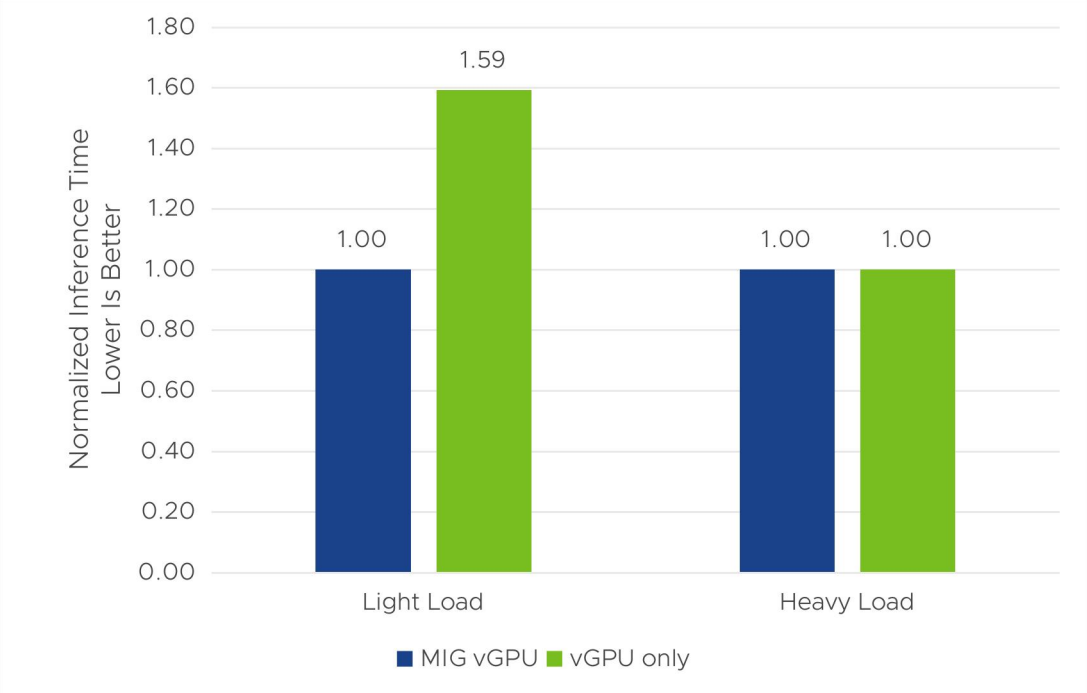

Figure 12. Inference time of MIG vGPU vs. vGPU only in vSphere 8 U1 with Tanzu

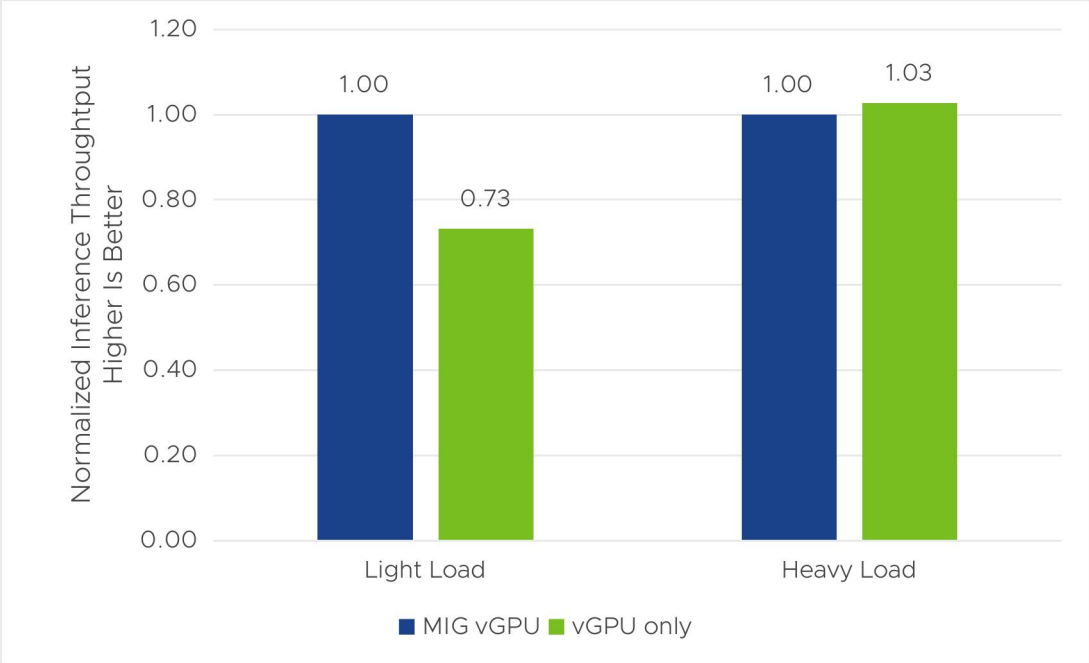

Figure 13. Inference throughput of MIG vGPU vs. vGPU only in vSphere 8 U1 with Tanzu

Scaling ML/AI Workloads Sharing a GPU

One of the key benefits of deploying ML/AI workloads with TKG is the fast and easy way of scaling ML/AI workloads by simply applying deployment configuration in a YAML file, similar to the way we usually do with Kubernetes. Virtual machines (or nodes in a Kubernetes cluster) are provisioned and scheduled based on the configurations we specify. Tanzu with TKG and NVIDIA AI Enterprise also provides good performance when scaling with more nodes running ML/AI workloads in a TKG cluster.

We demonstrate this in figures 14–17. In our experiments, we ran the training and inference Mask R-CNN workloads scaling from 1–7 VMs (or TKG nodes) concurrently, sharing a single A100 GPU using a MIG vGPU profile. Since we usually apply GPU sharing for a light load, we set batch size = 2 for Mask R-CNN so that the ML load was very light. We chose the lowest vGPU profile (A100-1-5c) when configuring the VM class in the workload management of the vSphere Client and used this VM class to deploy a TKG cluster with a number of nodes ranging from 1–7. The results (figures 14–17) show the linear scaling in throughput with multiple nodes sharing the same GPU, and the training and inference time have minimal impact with multiple VMs running concurrently. This comes from the use of MIG for the light-load case.

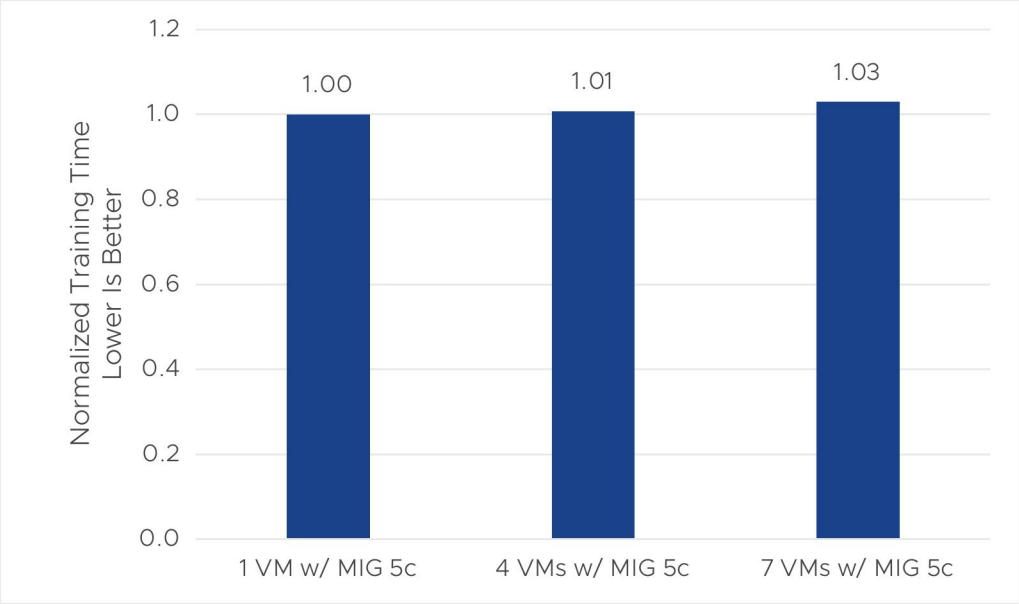

Figure 14. Training time when scaling in vSphere 8 U1 with Tanzu

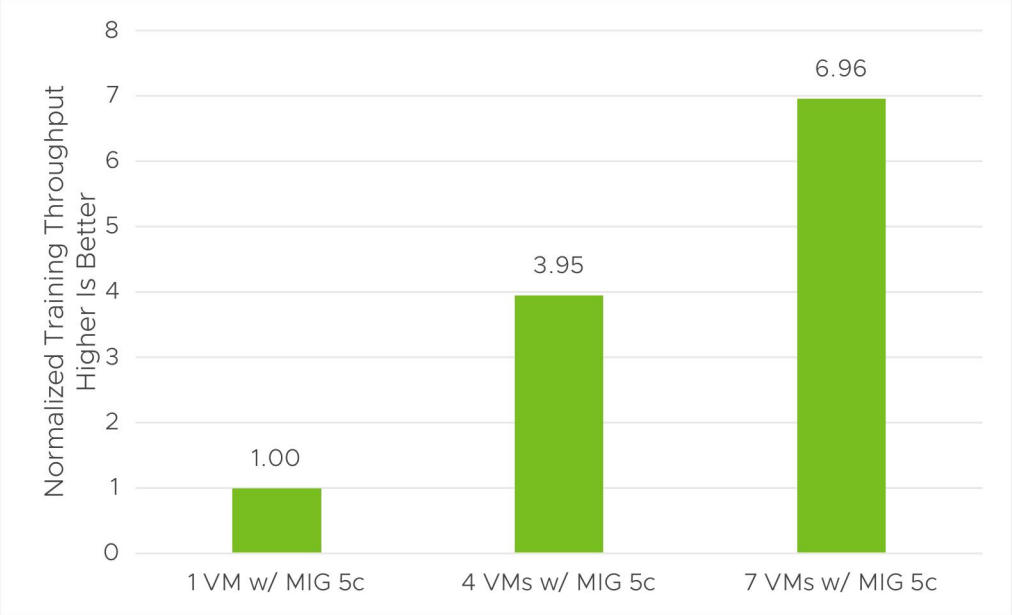

Figure 15. Training throughput when scaling in vSphere 8 U1 with Tanzu

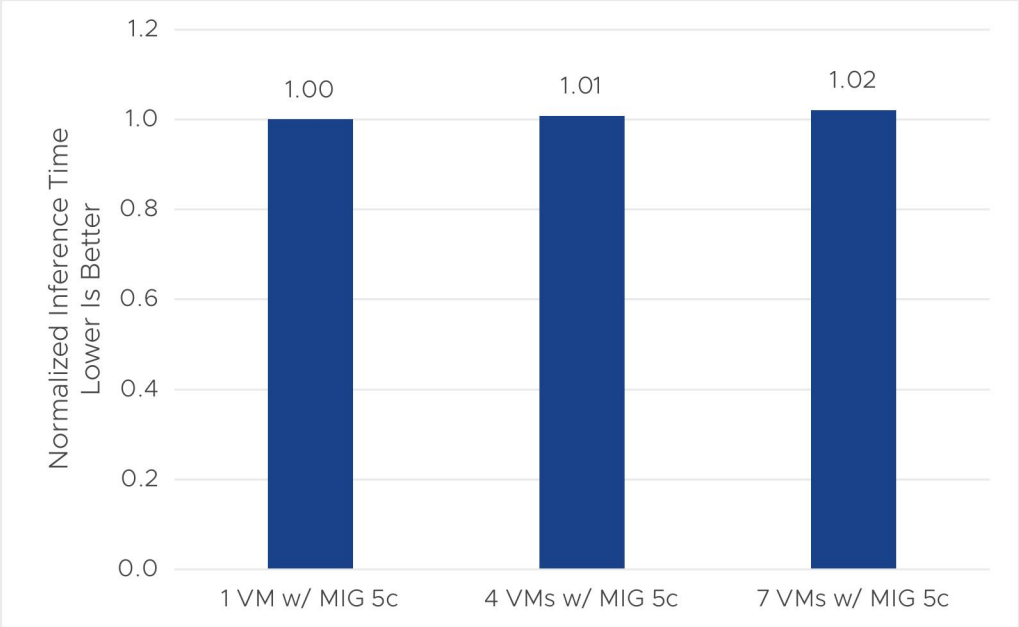

Figure 16. Inference time when scaling in vSphere 8 U1 with Tanzu

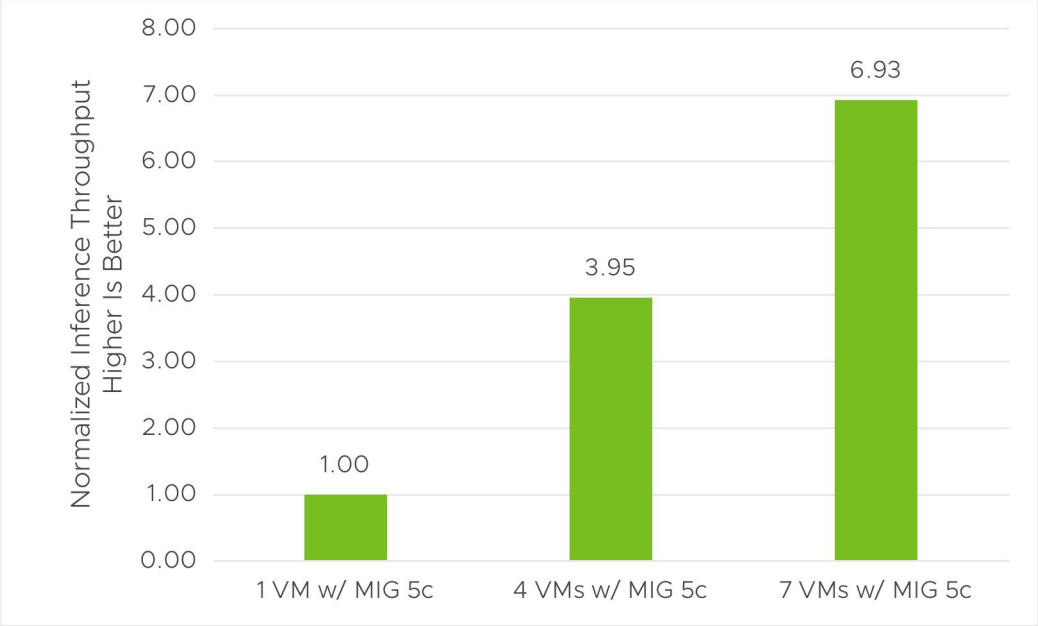

Figure 17. Inference throughput when scaling in vSphere 8 U1 with Tanzu

Takeaways

We explored ML/AI workloads in vSphere 8 U1 with TKG and NVIDIA AI Enterprise and provided best practices for optimizing and scaling these workloads. A few key takeaways from our performance study include:

- Compared to the traditional way of deploying ML/AI workloads with vGPUs, the deployment with TKG has no overhead or even gives better performance with an optimized configuration.
- ML/AI workloads in vSphere 8 U1 perform slightly better (2%–7%) than in vSphere 7 U3.
- When your workloads are lightweight (small models, small batch sizes, and small input data) and won't utilize all available CUDA cores, choosing MIG vGPU can give you a higher consolidation of VMs (TKG nodes) per GPU with better performance. This also means saving money for your ML/AI infrastructure.
- Scaling ML/AI workloads with TKG is fast, easy, and performs well. This is one of the key benefits of using TKG for ML/AI.

About the Authors

Lan Vu has worked at VMware for eight years, focusing on GPU virtualization with vSphere and machine learning. Lan is interested in developing solutions that bring high performance and low cost to customers' IT infrastructures so they can configure their systems to best utilize cloud resources. Lan holds a PhD in computer science from the University of Colorado Denver and has 24 issued and pending patents. She has been awarded VMware's Most Prolific Inventor.

Hari Sivaraman is a staff performance engineer at VMware. Hari has extensive experience in designing and running experiments to measure and predict the performance of computer systems, and he has built analytical and simulation models to estimate performance and answer "what-if" questions. He uses machine learning to solve problems in cloud management systems. Hari has written and collaborated on several papers and blog articles about AI/ML topics regarding vSphere performance. Hari has 25 patents and has been awarded VMware's Most Prolific Inventor.

Acknowledgments

We would like to thank Uday Kurkure, Justin Murray, Matthew McClure, Yu Wang, Karthik Ganesan, and Charlie Huang (NVIDIA) for providing feedback on our work, and Julie Brodeur for reviewing this paper. The authors acknowledge Juan Garcia-Rovetta and Tony Lin of VMware for their management support.