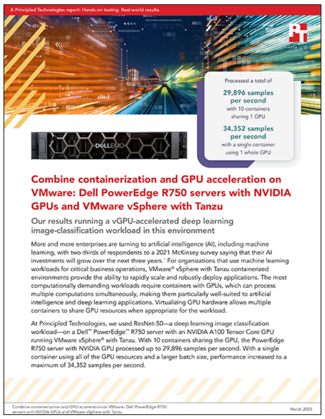

A Principled Technologies report: Hands-on testing. Real-world results.

Processed a total of

29,896 samples per second

with 10 containers sharing 1 GPU

34,352 samples per second

with a single container using 1 whole GPU

Combine containerization and GPU acceleration on VMware: Dell PowerEdge R750 servers with NVIDIA GPUs and VMware vSphere with Tanzu

Our results running a vGPU-accelerated deep learning image-classification workload in this environment

More and more enterprises are turning to artificial intelligence (AI), including machine learning, with two-thirds of respondents to a 2021 McKinsey survey saying that their AI investments will grow over the next three years.[1] For organizations that use machine learning workloads for critical business operations, VMware® vSphere with Tanzu containerized environments provide the ability to rapidly scale and robustly deploy applications. The most computationally demanding workloads require containers with GPUs, which can process multiple computations simultaneously, making them particularly well-suited to artificial intelligence and deep learning applications. Virtualizing GPU hardware allows multiple containers to share GPU resources when appropriate for the workload.

At Principled Technologies, we used ResNet-50—a deep learning image classification workload—on a Dell™ PowerEdge™ R750 server with an NVIDIA A100 Tensor Core GPU running VMware vSphere® with Tanzu. With 10 containers sharing the GPU, the PowerEdge R750 server with NVIDIA GPU processed up to 29,896 samples per second. With a single container using all of the GPU resources and a larger batch size, performance increased to a maximum of 34,352 samples per second.

Combine containerization and GPU acceleration on VMware: Dell PowerEdge R750 servers with NVIDIA GPUs and VMware vSphere with Tanzu

March 2023

The science behind the report:

Combine containerization and GPU acceleration on VMware: Dell PowerEdge R750 servers with NVIDIA GPUs and VMware vSphere with Tanzu

This document describes what we tested, how we tested, and what we found. To learn how these facts translate into real-world benefits, read the report Combine containerization and GPU acceleration on VMware: Dell PowerEdge R750 servers with NVIDIA GPUs and VMware vSphere with Tanzu.

We concluded our hands-on testing on February 14, 2023. During testing, we determined the appropriate hardware and software configurations and applied updates as they became available. The results in this report reflect configurations that we finalized on February 13, 2023 or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

Our results

To learn more about how we have calculated the wins in this report, go to http://facts.pt/calculating-and-highlighting-wins. Unless we state otherwise, we have followed the rules and principles we outline in that document.

Table 1: Configuration details and results

vGPU type	R750_GRID-A100-40C	R750_GRID-A100-4C
Tuning name	40c_x1_x1-production	4c_x1_x10-production
vGPU RAM (MiB)	40	4
Tanzu node count	1	10
Batch size (# images/batch)	2,048	128
Number of copy streams	6	5
Number of inference streams	2	2
Target QPS (queries/sec)	34,000	3,180
Per-node throughput (samples/s)	34,352	2,989
Aggregate throughput (samples/s)	34,352	29,896
Percentage CPU utilization	3.5	20.6
Memory usage (MiB)	173,345	174,154
System power usage (Watts)	743.2	755.7
Percentage GPU utilization	98.6	98.7
GPU memory usage (MiB)	40,320	39,681
GPU power usage (W)	239.7	245.0
GPU temperature (°C)	63.5	63.2

System configuration information

Table 2: Detailed information on the systems we tested.

Server configuration information	PowerEdge R750 (Server under test)	PowerEdge R7525 (Infrastructure)	PowerEdge R7525 (Infrastructure)
BIOS name and version	Dell 1.7.5	Dell 2.6.6	Dell 2.6.6
Non-default BIOS settings	System Profile=Performance, SR-IOV Global Enable=True	N/A	N/A
Operating system name and version/ build number	VMware ESXi 7.0.3	VMware ESXi 7.0.3	VMware ESXi 7.0.3
Date of last OS updates/patches applied	2023-01-03	2023-01-03	2023-01-03
Power management policy	Performance	Performance per Watt	Performance per Watt
Processor			
Number of processors	2	2	2
Vendor and model	Intel® Xeon® Gold 6330	AMD EPYC™ 7763	AMD EPYC 7763
Core count (per processor)	28	64	64
Core frequency (GHz)	2.0	2.45	2.45
Family/Model/Stepping	6/106/6	25/1/1	25/1/1
HT/SMT Enabled	Yes	Yes	Yes
Memory module(s)			
Total memory in system (GB)	1,024	1,024	1,024
Memory type 1			
Number of modules	16	16	16
Vendor and model	Samsung® M393A8G40AB2-CWE	Hynix® HMA84GR7DJR4N-XN	Hynix HMA84GR7DJR4N-XN
Size (GB)	64	32	32
Type	DDR4 3200	DDR4 3200	DDR4 3200
Max speed (MHz)	3,200	3,200	3,200
Speed (MHz)	2,933	2,933	2,933
Memory type 2			
Number of modules	N/A	16	16
Vendor and model	N/A	Micron® 36ASF4G72PZ-3G2E7	Hynix HMA84GR7CJR4N-XN
Size (GB)	N/A	32	32
Type	N/A	DDR4 3200	DDR4 3200
Speed (MHz)	N/A	3,200	3,200
Speed (MHz)	N/A	2,933	2,933

Server configuration information	PowerEdge R750 (Server under test)	PowerEdge R7525 (Infrastructure)	PowerEdge R7525 (Infrastructure)
Storage controller 1			
Vendor and model	Dell Boss-S2	Dell Boss-S2	Dell Boss-S2
Cache size (GB)	0	0	0
Firmware version	2.5.13.4008	2.5.13.4008	2.5.13.4008
Storage controller 2			
Vendor and model	Dell PERC H755N	Dell HBA355i	Dell HBA355i
Cache size (MB)	8,192	0	0
Firmware version	7.718.02.00	17.15.08.00	17.15.08.00
Local storage 1			
Number of drives	2	2	2
Drive vendor and model	Micron MTFDDAV480TDS	Micron MTFDDAV480TDS	Micron MTFDDAV480TDS
Drive size (GB)	480	480	480
Drive information (speed, interface, type)	6Gbps, SATA, M.2 SSD	6Gbps, SATA, M.2 SSD	6Gbps, SATA, M.2 SSD
Local storage 2			
Number of drives	4	3	4
Drive vendor and model	Samsung MZ-WLJ1T60	Samsung MZ-WLJ1T60	Samsung MZ-WLL1T6A
Drive size (GB)	1600	1600	1600
Drive information (speed, interface, type)	PCIex4, U.2 NVMe	PCIex4, U.2 NVMe	PCIex4, U.2 NVMe
Local storage 3			
Number of drives	N/A	1	N/A
Drive vendor and model	N/A	Samsung MZ-WLL1T6A	N/A
Drive size (GB)	N/A	1600	N/A
Drive information (speed, interface, type)	N/A	PCIex4, U.2 NVMe	N/A
Network adapter 1			
Vendor and model	Broadcom® NetXtreme BCM5720 Gigabit Ethernet	Broadcom NetXtreme BCM5720 Gigabit Ethernet	Broadcom NetXtreme BCM5720 Gigabit Ethernet
Number and type of ports	1x1Gbps	1x1Gbps	1x1Gbps
Network adapter 2			
Vendor and model	Intel Ethernet Controller E810-XXV for SFP	Mellanox® ConnectX-5 EN 25GbE Dual-port SFP28 Adapter	Mellanox ConnectX-5 EN 25GbE Dual-port SFP28 Adapter
Number and type of ports	2x25Gbps	2x25Gbps	2x25Gbps
Fibre Channel 1			
Vendor and model	Marvell® QLogic QLE2742	Marvell QLogic QLE2742	Marvell QLogic QLE2742
Number and type of ports	2x32Gbps	2x32Gbps	2x32Gbps
Interface	PCIe 3.0 x8	PCIe 3.0 x8	PCIe 3.0 x8

Server configuration information	PowerEdge R750 (Server under test)	PowerEdge R7525 (Infrastructure)	PowerEdge R7525 (Infrastructure)
Fibre Channel 1			
Vendor and model	Marvell QLogic QLE2772	N/A	N/A
Number and type of ports	2x32Gbps	N/A	N/A
Interface	PCIe 4.0 x8	N/A	N/A
Cooling fans			
Vendor and model	Foxconn® PIA060K12Q	Foxconn PIE060M12M	Foxconn PIE060M12M
Number of cooling fans	6	6	6
Power supplies			
Vendor and model	Dell 01CW9GA03	Dell 0M63JNA00	Dell 0M63JNA00
Number of power supplies	2	2	2
Wattage (W)	1,400	2,400	2,400

For storage for the system under test and support infrastructure, we used a commercially available all-flash storage solution connected via Fibre Channel. For this test we used a 10 TB volume.

Table 3: Detailed configuration information for the storage solution.

Storage configuration information	Storage solution
Number of storage controllers	2
Number of storage shelves	1
Number of drives per shelf	48
Drive size (TB)	1.92
Drive information	SAS SSD
NVRAM count	0
NVRAM size (GB)	N/A

Table 4: Detailed configuration information for the network switches we used.

Network switch configuration information	Dell S5248F-ON
Firmware revision	10.4.3.6.244 (2019-08-19T17:26:44-0700)
Number and type of ports	48x25GbE,4x100GbE,2x200GbE
Number and type of ports used in test	3x25GbE
Non-default settings used	None

Table 5: Detailed configuration information for the network switches we used.

Network switch configuration information	Brocade 6520
Firmware revision	v7.4.0a
Number and type of ports	96x32Gbps
Number and type of ports used in test	6x32Gbps
Non-default settings used	None

How we tested

We installed and configured the most recent version of VMware vSphere 7.0 Update 3 on one Dell PowerEdge R750 server and two Dell PowerEdge R7525 servers. We installed an NVIDIA A100 GPU in the PowerEdge R750 server. We installed the OS on internal SATA drives in all three servers. We configured and created a 10TB volume on an all-flash storage solution. We mapped the 10TB volume to all servers and it served as a shared datastore for Tanzu deployment.

We used a Dell X1052 switch for VM network, vMotion, and Management Network for Tanzu. We used a Dell S5248F-ON switch for Workload Network for Tanzu. We isolated the Workload Network behind a NAT gateway, and it utilized private addresses for all connectivity. We configured the Workload Network port group on a Distributed vSwitch, Tanzu Kubernetes deployment requires.

Installing vSphere 7.0u3 on the Dell PowerEdge R750 and R7525

1. Download the Dell Custom Image for ESXi 7.0 U3 from the following link: https://my.vmware.com/group/vmware/evalcenter?p=vsphere-eval-7#tab_download
2. Open a new browser tab, and connect to the IP address of the Dell PowerEdge server iDRAC.
3. Log in with the iDRAC credentials. We used root/calvin.
4. In the lower left corner of the screen, click Launch Virtual Console.
5. In the console menu bar, click Connect Virtual Media.
6. Under Map CD/DVD, click Browse… and select the image you downloaded in step 1. Click Open.
7. Click Map Device. Click Close.
8. On the console menu bar, click Boot, and select Virtual CD/DVD/ISO. Click Yes to confirm.
9. On the console menu bar, click Power, and select Power On System. Click Yes to confirm. The system boots to the mounted image and the Loading ESXi installer screen appears.
10. When prompted, press Enter to continue.
11. Press F11 to accept the EULA, and click Continue.
12. Select the storage device to target for installation. We selected the internal SD card. Press Enter to continue.
13. Press Enter to confirm the storage target.
14. Select the keyboard layout, and press Enter.
15. Provide a root password, and confirm the password. Press Enter to continue.
16. Press F11 to install.
17. Upon completion, press Enter to reboot the server.

Installing vCenter Server Appliance 7.0u3

1. Download the VMware vCenter 7.0u3 from the VMware support portal at https://my.vmware.com.
2. Mount the image on your local system and browse to the vcsa-ui-installer folder. If the installer doesn't automatically begin, expand the folder for your OS and launch it.
3. When the vCenter Server Installer wizard opens, click install.
4. To begin installation of the new vCenter server appliance, click Next.
5. Check the box to accept the license agreement and click Next.
6. Enter the IP address of one of your newly deployed Dell PowerEdge R7525 servers with ESXi 7.0u3. Provide the root password, and click Next.
7. To accept the SHA1 thumbprint of the server's certificate, click Yes.
8. Accept the VM name, and provide and confirm the root password for the VCSA. Click Next.
9. Set the size for the environment you're planning to deploy. We selected medium. Click Next.
10. Select the datastore on which to install. Accept the datastore defaults, and click Next.
11. Enter the FQDN, IP address information, and DNS servers you want to use for the vCenter server appliance. Click Next.
12. To begin deployment, click Finish.
13. When Stage 1 has completed, click Close. Click Yes to confirm.
14. Open a browser window, and connect to https://[vcenter.FQDN:5480/.
15. Click Set up on the Getting Started - vCenter Server page.
16. Enter the root password, and click Log in.
17. Click Next.
18. Enable SSH access, and click Next.
19. To confirm the changes, click OK.
20. For the single sign-on domain name, enter `vsphere.local`
21. Enter a password for the administrator account, confirm it, and click Next.
22. Click Next.
23. Click Finish.

Installing the vGPU host driver on the Dell PowerEdge R750 server

1. Open a new browser tab, and connect to the IP address of the Dell PowerEdge R750 server iDRAC.
2. Log in with the iDRAC credentials.
3. On the top menu bar, click Configuration, and click BIOS Settings from the drop-down menu.
4. Click and expand Integrated Devices.
5. Locate the SR-IOV Global feature, click the drop-down menu at the right of it, and enable it.
6. To apply the new setting, click Apply.
7. Open a new browser tab, and connect to the IP address of the vCenter.
8. Log into vCenter as an administrator user.
9. Click Hosts and Clusters.
10. Click Dell PowerEdge R750 host.
11. Navigate to Configuration → Hardware → Graphics.
12. Under the Host Graphics tab, set Default graphics type to Shared Direct and Shared passthrough GPU assignment policy to Spread VMs across GPUs (best performance).
13. To apply the changes, click OK.
14. From the vCenter console, right-click the Dell PowerEdge R750 host, and click Maintenance Mode to enter maintenance mode.
15. Download NVIDIA AIE ESXi Driver VIB file from https://docs.nvidia.com/grid/latest/grid-software-quick-start-guide/index.html#redeeming-pak-and-downloading-grid-software
16. SSH into Dell PowerEdge R750 server with Administrator privilege.
17. Install the NVIDIA vGPU driver by running to following command:

```
esxcli software vib install -v Absolute_Path_of_Directory_of_the_VIB_File/NVIDIA-AIE*.vib
```

18. Right-click the Dell PowerEdge R750 host, and click Maintenance Mode to exit maintenance mode.
19. Verify that the NVIDIA kernel driver can successfully communicate with the physical GPUs in the system by running the `nvidia-smi` command without any options.

Installing instrumentation script on the Dell PowerEdge R750

1. Contact support@principledtechnologies.com to get a copy of monitor-service.sh script.
2. Copy monitor-service.sh to a datastore accessible to the GPU-equipped ESXi host.
3. Set the execution permissions on the script:

```
chmod +x /THE/PATH/TO/monitor-service.sh
```

4. Test that the script works:

```
cd $(dirname /THE/PATH/TO/monitor-service.sh)
./monitor-service.sh start; sleep 5
./monitor-service.sh status; sleep 5
./monitor-service.sh stop; sleep 5
echo "---stopped---"
./monitor-service.sh status
```

5. You should see something similar to the following:

```
The nvidia-smi process is active with PID XXXXXXXX
The esxtop process is active with PID XXXXXXXX
---stopped---
The nvidia-smi process is not active
The esxtop process is not active
where XXXXXXXX are os-determined process id numbers.
```

Creating a cluster in VMware vSphere 7.0u3

1. Open a browser, and enter the address of the vCenter server you deployed (https://[vcenter.FQDN]/ui).
2. Select the vCenter server in the left panel, right-click, and select New Datacenter.
3. Provide a name for the new data center, and click OK.
4. Select the data center you just created, right-click, and select New Cluster.
5. Give a name to the cluster, and enable vSphere DRS. Click OK.
6. In the cluster configuration panel, under Add hosts, click Add.
7. Check the box for Use the same credentials for all hosts. Enter the IP Address and root credentials for the first host, and the IP addresses of all remaining hosts. Click Next.
8. Check the box beside Hostname/IP Address to select all hosts. Click Ok.
9. Click Next.
10. Click Finish.
11. Click the cluster, navigate to Configuration → Services → vSphere DRS, and click EDIT.
12. Select Fully Automated, and click OK.
13. Click the cluster, navigate to Configuration → Services → vSphere Availability, and click EDIT.
14. Switch the toggle to the right for vSphere HA, and click OK.

Creating a Distributed vSwitch and Port Group

1. From vSphere client, click Home → Networking.
2. Select your Datastore and, in the Actions pulldown menu on the right panel, select Distributed vSwitch → New Distributed vSwitch.
3. Give your vSwitch a name or accept the default. Click Next.
4. Select 7.0.0 - ESXi 7.0 and later as the version, and click Next.
5. Select the number of uplinks per ESXi host you'll give to the vSwitch. We selected 1. Click Next.
6. Click Finish.
7. Right-click the new DvSwitch, and select Add and Manage Hosts.
8. Leave Add hosts selected, and click Next.
9. Click the + sign to add new hosts.
10. Check the box beside Host to select all the hosts in your target cluster. Click OK, and click Next.
11. Select the NIC you want to use for this DvSwitch and click Assign Uplink.
12. Accept the defaults at the top of the panel, but check the box beside "Apply this uplink assignment to the rest of the hosts." Click OK, and click Next.
13. Do not assign vmkernel adapters at this time. Click Next.
14. Do not migrate any VM networking at this time. Click Next.
15. Click Finish.
16. Right-click the DvSwitch, and select Distributed Port Group → New Distributed Port Group.
17. Give it the name Workload Network, and click Next.
18. Click Next.
19. Click Finish.

Creating a DevOps user

1. From vSphere client, click Home → Administration.
2. In the left panel, click Users and Groups.
3. In the right panel, click Users, select the vsphere.local domain, and click Add.
4. Provide a username and password, and click Add.
5. For simplicity, we added the DevOps user to a group with Administrator privileges. Click Groups, and select the Administrators group. Click Edit.
6. Under Add Members, search for the DevOps user you just created, and add them to the administrators group. Click Save.

Creating the HAproxy content library

1. Click the following link to download the vSphere compatible HAproxy ovf (v0.1.8): https://github.com/haproxytech/vmware-haproxy
2. From vSphere client, in the left menu pane, click Content Libraries.
3. In the Content Libraries panel on the right, click Create.
4. Name the content library HAproxy-cl, and click Next.
5. Accept the default, and click Next.
6. Choose the storage location for the content library, and click Next.
7. Review, and click Finish.
8. Click the newly created HAproxy-cl content library.
9. In the upper portion of the right panel for HAproxy-cl, click the actions pull-down menu, and select Import Item.
10. Change the selection to local file, and click Upload files.
11. Browse to the location of the ovf file you downloaded in step 1, and click Open.
12. Click Import.

Creating TKG content library

1. From the vSphere client, in the left menu pane, click Content Libraries.
2. In the Content Libraries panel on the right, click Create.
3. Name the content library TKG-cl, and click Next.
4. Select Subscribed content library, and use https://wp-content.vmware.com/v2/latest/lib.json for the subscription URL. Click Next.
5. To verify, click Yes.
6. Choose the storage location for the content library, and click Next.
7. Review, and click Finish.

Creating the storage tag

1. From the vSphere client, select Menu → Storage.
2. From the left pane, select the shared storage you created for Tanzu on the all-flash storage solution.
3. Under the Summary tab, locate the Tags panel, and click Assign.
4. Click Add Tag.
5. Name the tag Tanzu. Click Create New Category.
6. Give the category name Tanzu Storage. Clear all object types except Datastore, and click Create.
7. Use the Category pull-down menu to select Tanzu Storage, and click Create.
8. Check the box beside the newly created tag, and click Assign.

Creating the VM storage policy

1. From the vSphere client, click Menu → Policies and Profiles.
2. On the left panel, click VM Storage Policies.
3. Click Create.
4. Create a new VM Storage policy named `tkg-clusters` and click Next.
5. Check the box for Enable tag-based placement rules, and click Next.
6. Use the Tag Category pull-down menu to select the Tanzu Storage policy you created. Click Browse Tags.
7. Click the Tanzu checkbox, and click OK.
8. Click Next.
9. Review the compatible storage to make sure your storage target is marked as compatible, and click Next.
10. Click Finish.

Deploying HAProxy

1. From the vSphere client, click Menu → Content Libraries.
2. Click the HAproxy-cl library.
3. In the left panel, click OVF & OVA Templates, and right-click the HAproxy template that appears in the panel below. Select New VM from This Template…
4. Provide a simple name (we used HAproxy), and select the data center and/or folder to which you want to deploy. Click Next.
5. Select the cluster or compute resource where you want to deploy the HAproxy VM, and click Next.
6. Review details, and click Next.
7. Check the box for I accept all license agreements, and click Next.
8. Accept the default configuration, and click Next.
9. Select the target storage for the VM, and click Next.
10. Select VM Network for the Management network, and choose a network for the Workload network. Choose the same network for the Frontend network, and click Next.
11. Customize the template using the following:
    - Appliance Configuration section
        - For the root password, we used Password1!
        - Check the box for Permit Root Login.
        - Leave the TLS CA blank.
    - Network Configuration section
        - We left the default haproxy.local.
        - For local DNS server, we used 10.41.0.10.
        - For management IP, we used 10.222.201.200/16.
        - For management gateway, we used 10.222.0.1. (NOTE: The description asks for the workload network gateway address. You should enter the management gateway address instead.)
        - For Workload IP, we used 192.168.1.2/24.
        - For Workload gateway, we used 192.168.1.1.
    - Load Balancing section
        - For load balancer IP ranges, we used 192.168.1.240/29.
        - Accept the default management port.
        - For HAProxy User ID, we used admin.
        - We used Password1! For the HAProxy password.
12. Click Next.
13. Review the summary, and click Finish. The deployment takes a few minutes to completely deploy and configure.
14. Power on the HAProxy VM.

Configuring Workload Management

1. From the vSphere client, click Menu → Workload Management.
2. Click Get Started.
3. Review the messages and warnings regarding supported configurations, and click Next.
4. Select the cluster on which you want to enable workload management, and click Next.
5. Choose the capacity for the control plane VMs (we chose Small), and click Next.
6. Choose the storage policy to be used for the control plane nodes (we chose tkg-clusters), and click Next.
7. Configure the Load Balancer section as follows:
    - Name: haproxy
    - Type: HA proxy
    - Data plane API Addresses: 10.222.201.200:5556
    - User name: admin
    - Password: Password1!
    - IP Address Ranges for Virtual Servers: 192.168.1.240-192.168.1.247
    - Server Certificate Authority:
        - Open an SSH session to the HAProxy management address, and connect using root and Password1!
        - Type the following: `cat /etc/haproxy/ca.crt`
        - Copy the entire output (including the first and last lines), and paste the contents into the Server Certificate Authority box.
8. Close the SSH session.
9. Click Next.
10. Configure Workload Management as follows:
    - Network: VM Network
    - Starting IP Address: 10.222.201.201
    - Subnet Mask: 255.255.0.0
    - Gateway: 10.222.0.1
    - DNS Server: 10.41.0.10
    - NTP Server: 10.40.0.5
11. Click Next.
12. Configure Workload Network as follows:
    - Leave the default for Services addresses.
    - DNS Servers: 10.41.0.10
    - Under Workload Network, click Add.
    - Accept default for network-1.
    - Port Group: Workload Network.
    - Gateway: 192.168.1.1.
    - Subnet: 255.255.255.0
    - IP Address Ranges: 192.168.1.65-192.168.1.126
13. Click Save.
14. For TKG Configuration, do the following:
15. Beside Add Content Library, click Add.
16. Select the TKG-cl library, and click OK.
17. Click Next.
18. Click Finish. The workload management cluster deploys and configures. You may see apparent errors during configuration; these will resolve upon successful completion.

Configuring Kubernetes namespace for service deployment

1. In Workload Management, click Namespaces.
2. Click Create Namespace.
3. Select the target cluster, and provide a name. We used tanzu-ns. Click Create.
4. Click the Permissions tab, and click Add.
5. Choose vSphere.local for the identity source. Search for the DevOps user you created. Select the "can edit" role, and click OK.
6. Click the Storage tab.
7. In the Storage Policies section, click Edit.
8. Select the tkg-clusters policy, and click Ok. The environment is ready for connection and deployment of containers.

Uploading Ubuntu ISO to DataStore

1. Download the Ubuntu Server (64 bit) ISO from https://ubuntu.com. We used Ubuntu 22.04.
2. Log into vCenter, and from the Menu drop-down, click Storage.
3. Select datastore1, and click Files.
4. Click Upload Files, and upload the Ubuntu ISO image.

Provisioning the Tanzu CLI, Builder, and Registry VMs

1. Right-click the cluster, and click New Virtual Machine.
2. Click Next.
3. Enter a name for the VM, and click Next.
4. Click Next.
5. Select DataStore, and click Next.
6. Click Next.
7. From the Guest OS Family drop-down menu, select Linux. From the guest OS version drop-down menu, select Ubuntu Linux (64 bit). Click Next.
8. Assign the VM 16 vCPUs, 16 GB of memory, and a 256GB hard disk.
9. Set the memory reservation to 100%.
10. From the New CD/DVD Drive drop-down menu, select Datastore ISO File. Select the Ubuntu ISO you uploaded to the datastore previously. Ensure Connect At Power On is checked, and click Next.
11. Click Finish.
12. Connect via the remote console, and complete the Ubuntu installation process.

Installing Ubuntu 22.04 (Registry, Tanzu CLI, and Builder VMs)

1. In vCenter, locate the VM in the inventory.
2. Power on the VM, and click Launch Remote Console.
3. Click Install Ubuntu.
4. Click Continue.
5. Select Minimal installation, and click Continue.
6. Click Install now.
7. Click Continue twice.
8. Enter your desired full name, computer name, username, and password, and click Continue.
9. Click Restart Now.
10. Press Enter.
11. Enter your password, and click Sign In.
12. Install OS and software updates when prompted by the Software Updater.
13. Click Restart Later, and power off the VM.

Adding the Workload network to Tanzu CLI VM

1. Right-click the VM in vCenter, and click Edit Settings.
2. Click Add New Device, and select Network Adapter.
3. From the New Network drop-down menu, select Browse.
4. Click Workload Network, and click OK.
5. Click OK.
6. Power on the VM, and click Launch Remote Console.
7. In the remote console, log in with your username and password.
8. In a terminal, edit the network config:

```
sudo nano /etc/netplan/00-installer-config.yaml
```

9. Determine interface names and MAC addresses:

```
sudo ip addr show
```

10. Make the file look like the following (replace the last part of IP addresses XXX and mac addresses XX:XX:XX:XX:XX:XX as appropriate):

```
# This is the network config written by 'subiquity'
network:
 ethernets:
   ens192:
      addresses:
      - 10.222.222.XXX/16
      dhcp4: false
      link-local: []
      match:
        macaddress: XX:XX:XX:XX:XX:XX
      nameservers:
        addresses:
        - 10.41.0.10
        search:
        - principledtech.com
      routes:
      - to: default
        via: 10.222.0.1
      set-name: ens192
    ens224:
      addresses:
      - 192.168.1.XXX/24
      dhcp4: false
      link-local: []
      match:
        macaddress: XX:XX:XX:XX:XX:XX
      set-name: ens224
  version: 2
```

11. Save the file by pressing `CTRL+O`.
12. Exit by pressing `CTRL+X`.
13. Apply the new networking settings:

```
sudo netplan apply
```

Preparing the Tanzu CLI VM Tanzu environment

Installing Tanzu CLI tools

1. SSH into the Tanzu CLI VM.
2. Become root:

```
sudo su
```

3. Create a directory for the installers:

```
mkdir -p /opt/tanzu-install
```

4. Change directories to installer directory:

```
cd /opt/tanzu-install
```

5. In a web browser, go to https://customerconnect.vmware.com/en/downloads/details?downloadGroup=TKG-160&productId=988&rPId=86183&download=true&fileId=62abc392fe09cf6bdd81c32a47ae3170&uuId=c3fa1a42-aaee-4048-a800-4b4fd15b4037.
6. Beside VMware Tanzu CLI for Linux, click Download Now.
7. Use SCP or similar tool to move the file to the Tanzu CLI VM. Save the file as /opt/tanzu-install/tanzu-v1.6-cli-bundle-linux-amd64.tar.
8. Extract Tanzu CLI installer tarball:

```
tar -xvf ./tanzu-v1.6-cli-bundle-linux-amd64.tar
```

9. Install the Tanzu binary and create a symlink:

```
install cli/core/v1.6/tanzu-core-linux_amd64 /usr/local/bin/tanzu-v1.6
ln -s /usr/local/bin/tanzu-v1.6 /usr/local/bin/tanzu
```

10. Initialize Tanzu:

```
tanzu init
```

11. Check that you have installed Tanzu:

```
tanzu version
```

12. Install Tanzu plugins:

```
tanzu plugin clean
tanzu plugin sync
tanzu plugin list
```

13. Change to the installer packaged archives directory:

```
cd /opt/tanzu-install/cli/cli
```

14. Install ytt:

```
gunzip ytt-linux-amd64-v0.41.1+vmware.1.gz
install ytt-linux-amd64-v0.41.1+vmware.1 /usr/local/bin/ytt-tanzu-v1.6
ln -s /usr/local/bin/ytt-tanzu-v1.6 /usr/local/bin/ytt
```

15. Install kapp:

```
gunzip kapp-linux-amd64-v0.49.0+vmware.1.gz
install kapp-linux-amd64-v0.49.0+vmware.1 /usr/local/bin/kapp-tanzu-v1.6
ln -s /usr/local/bin/kapp-tanzu-v1.6 /usr/local/bin/kapp
```

16. Install imgpkg:

```
gunzip imgpkg-linux-amd64-v0.29.0+vmware.1.gz
install imgpkg-linux-amd64-v0.29.0+vmware.1 /usr/local/bin/imgpkg-tanzu-v1.6
ln -s /usr/local/bin/imgpkg-tanzu-v1.6 /usr/local/bin/imgpkg
```

17. Install kbld:

```
gunzip kbld-linux-amd64-v0.34.0+vmware.1.gz
install kbld-linux-amd64-v0.34.0+vmware.1 /usr/local/bin/kbld-tanzu-v1.6
ln -s /usr/local/bin/kbld-tanzu-v1.6 /usr/local/bin/kbld
```

Installing vSphere with Tanzu kubectl

1. SSH into the Tanzu CLI VM.
2. Become root:

```
sudo su
```

3. Change directories to installer directory:

```
cd /opt/tanzu-install
```

4. Go to https://customerconnect.vmware.com/en/downloads/details?downloadGroup=TKG-160&productId=988&rPId=86183&download=true&fileId=62abc392fe09cf6bdd81c32a47ae3170&uuId=c3fa1a42-aaee-4048-a800-4b4fd15b4037.
5. Beside kubectl cli v1.23.8 for Linux, click Download Now.
6. Use SCP or similar tool to move the file to the Tanzu CLI VM. Save the file as /opt/tanzu-install/opt/tanzu-install/tanzu-v1.6-kubectl.gz
7. Extract Tanzu kubectl installer tarball:

```
gunzip -krv ./tanzu-v1.6-kubectl.gz
```

8. Install the extracted binary:

```
install ./tanzu-v1.6-kubectl /usr/local/bin/kubectl-tanzu-v1.6
```

9. Create a symlink to kubectl:

```
ln -s /usr/local/bin/kubectl-tanzu-v1.6 /usr/local/bin/kubectl
```

Installing Kubernetes buildkit (optional)

1. SSH into the Tanzu CLI VM.
2. Become root:

```
sudo su
```

3. Change directories to installer directory:

```
cd /opt/tanzu-install
```

4. Download the buildkit deb package:

```
wget https://github.com/vmware-tanzu/buildkit-cli-for-kubectl/releases/download/v0.1.6/kubectl-
buildkit_0.1.6_amd64.deb
```

5. Install the package:

```
dpkg -i kubectl-buildkit_0.1.6_amd64.deb
```

Adding the vGPU to the builder VM

1. In vCenter, locate the builder VM in the Inventory panel.
2. Power off the VM.
3. Under VM Hardware, click Edit Settings…
4. Click Add new device.
5. Click PCI Device.
6. Click NVIDIA GRID vGPU.
7. For the NVIDIA GRID vGPU profile, select the desired profile. We used grid_a100-4c and grid_a100-40c.
8. Power on the VM.

Installing the NVIDIA client driver

1. Log into the VM via SSH.
2. Copy the client driver file to the VM:

```
scp ./NVIDIA-Linux-x86_64-510.47.03-grid.run \
  USER@HOSTNAME:~/NVIDIA-Linux-x86_64-510.47.03-grid.run
```

3. Run the NVIDIA installer:

```
cd ~
sudo ./NVIDIA-Linux-x86_64-510.47.03-grid.run
```

4. Follow the prompts, accepting the defaults.
5. Test that the driver is installed correctly:

```
nvidia-smi
```

6. You should see something similar to the following:

```
Wed Feb  8 21:05:26 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 510.47.03    Driver Version: 510.47.03    CUDA Version: 11.6     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  GRID A100-4C        On   | 00000000:02:00.0 Off |                  N/A |
| N/A   N/A    P0    N/A /  N/A |      0MiB /  4096MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Installing NVIDIA License Server

This section contains the steps we took to install the DLS license server virtual appliance and create a license server on the NVIDIA Licensing Portal.

Installing the NVIDIA Licensing Server DLS virtual appliance

1. Log into the NVIDIA Enterprise Application Hub.
2. Click NVIDIA LICENSING PORTAL.
3. Navigate to Software Downloads.
4. Click the Non-Driver downloads tab.
5. Download the NLS License Server (DLS) 1.1 for VMware vSphere.
6. When the download finishes, import the OVA template onto a VMware vSphere installation suitable for hosting the licensing server virtual appliance.
7. When the VM is created, open a browser to https://<dls-vm-ip-address>.
8. Click NEW INSTALLATION.
9. Provide the credentials for the DLS administrator.
10. Click REGISTER.
11. Copy the local reset secret to a safe location.
12. Click CONTINUE TO LOGIN.
13. Log in with the DLS administrator credentials.
14. Click SERVICE INSTANCE.
15. Click CREATE STANDALONE.

Creating the NVIDIA Licensing Server on the NVIDIA Licensing Portal

1. Log into the NVIDIA Enterprise Application Hub.
2. Expand LICENSE SERVER, and click CREATE SERVER.
3. Give the server a name, and click Next.
4. On the Select Features page, select the licenses you want to apply.
5. Click CREATE SERVER.

Registering the virtual appliance with the NVIDIA Licensing Portal

1. Log into the virtual appliance at https://<dls-vm-ip-address>.
2. Click SERVICE INSTANCES.
3. Click DOWNLOAD DLS INSTANCE TOKEN.
4. In the NVIDIA Licensing Portal, click Actions, and click Upload on-premises (DLS) instance token.
5. Click UPLOAD DLS INSTANCE TOKEN.
6. Click SELECT INSTANCE TOKEN.
7. Navigate to the token file you downloaded in Step 3 and upload the file.
8. Click Register.

Binding the License Server to a Service Instance

1. In the NVIDIA Licensing Portal, expand LICENSE SERVERS, and click LIST SERVERS.
2. Click Actions, and click Bind.
3. Select the service instance, and click BIND.

Installing a License Server on the DLS Instance

1. In the NVIDIA Licensing Portal, expand LICENSE SERVERS, and click LIST SERVERS.
2. Click Actions, and click Download.
3. Click Download to obtain the .bin file.
4. Log into the virtual appliance at https://<dls-vm-ip-address>.
5. Click UPLOAD SERVER.
6. Click SELECT LICENSE SERVER FILE.
7. Upload the .bin file.
8. Click INSTALL.

Generating a client token

1. Log into the virtual appliance at https://<dls-vm-ip-address>.
2. Click SERVICE INSTANCE.
3. Click Actions, and click Generate client configuration token.
4. Select the DLS instance, and click Download Client Configuration Token.
5. Copy the downloaded client configuration token to each server that requires the license.

Installing NVIDIA License on vGPU VM

1. Log into the VM via SSH.
2. Become root:

```
sudo su
```

3. Edit the NVIDIA grid service config file:

```
nano /etc/nvidia/gridd.conf
```

4. Make the file look like the following:

```
ServerAddress=LICENSE_SERVER_IP
ServerPort=7070
```

5. Save the file:

```
[CTRL] + O
```

6. Exit the editor:

```
[CTRL] + X
```

7. Restart NVIDIA grid service:

```
systemctl restart nvidia-gridd
```

8. Check license status:

```
nvidia-smi -q
```

Installing the Kubectl Plugin for vSphere on the Tanzu CLI VM

1. Open a browser on the VM, and navigate to the IP of one of the three SupervisorControlPlane VMs. In our environment, the first control plane VM is at 10.218.201.201.
2. Click Advanced, and click Accept the Risk and Continue to bypass the certificate warning.
3. Click Download CLI Plugin Linux.
4. Select Save File, and click OK.
5. Open the Files app, and navigate to the Downloads folder.
6. Right-click vsphere-plugin.zip, and select Extract Here.
7. Open a terminal and navigate to the vsphere-plugin binary within the extracted folder:

```
cd Downloads/vsphere-plugin/bin
```

8. Make the vsphere-plugin binary executable, and add it to PATH:

```
sudo mv kubectl-vsphere /usr/local/bin/
sudo mv kubectl /usr/local/bin/
```

Installing Python on Registry and Builder VMs

1. Open a bash prompt or SSH session.
2. Become root:

```
sudo su
```

3. Update APT cache:

```
apt-get update -y
```

4. Install required packages:

```
apt-get install -y build-essential python3 python3-setuptools python3-dev python3-wheel python3-
pip python3-venv
```

5. Upgrade Python pip:

```
pip install --upgrade pip
```

Installing Docker on Registry and Builder VMs

1. Open a bash prompt or SSH session.
2. Update APT cache:

```
sudo apt-get update -y
```

3. Install handy system utilities:

```
sudo apt-get install -y aptitude build-essential git ntp ntpdate \
  openssh-server htop wget curl expect
```

4. Install Docker keyring:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

5. Verify docker works as root user:

```
sudo docker run --rm hello-world
```

6. Ensure docker-py python package is not installed:

```
sudo pip uninstall docker-py
```

7. Install python utilities for Docker:

```
sudo pip install docker jsondiff docker-compose enum34
```

8. Create the Docker group:

```
sudo groupadd docker
```

9. Add user to Docker group (substitute USER with your login):

```
sudo usermod -a -g docker ${USER}
```

10. Verify docker works as non-root user:

```
docker run --rm hello-world
```

Installing NVIDIA Docker on Builder VM

1. Open a bash prompt or SSH session.
2. Become root:

```
sudo su
```

3. Add NVIDIA GPG key and repository sources:

```
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | \
sudo gpg --dearmor -o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg
curl -s -L https://nvidia.github.io/libnvidia-container/$distribution/libnvidia-container.list | \
sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg]
https://#g' | \
tee /etc/apt/sources.list.d/nvidia-container-toolkit.list
```

4. Update APT cache:

```
apt-get update -y
```

5. Install NVIDIA Docker:

```
apt-get install -y nvidia-container-toolkit
```

6. Configure Docker daemon:

```
nvidia-ctk runtime configure --runtime=docker
```

7. Restart Docker:

```
systemctl restart docker
```

8. Check that NVIDIA Docker is working:

```
docker run --rm --runtime=nvidia --gpus all nvidia/cuda:11.6.2-base-ubuntu20.04 nvidia-smi
```

9. You should see something similar to this:

```
Wed Feb  8 21:05:26 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 510.47.03    Driver Version: 510.47.03    CUDA Version: 11.6     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  GRID A100-4C        On   | 00000000:02:00.0 Off |                  N/A |
| N/A   N/A    P0    N/A /  N/A |      0MiB /  4096MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Configuring SystemD service template on Registry VM

1. Connect to the registry VM via SSH through VMware vCenter remote desktop.
2. Become root:

```
sudo su
```

3. Create the following file at `/ets/systemd/system/docker-compose@.service`:

```
###############################################################
# This is a generic systemd script to treat a docker-compose.yml
# as a service. It is derived from
#   https://github.com/docker/compose/issues/4266#issuecomment-302813256
# This is compatible with systemd on ubuntu, and may need to be
# modified for other systems.
# To use this, create a directory named
#   /opt/docker-services/YOUR_SERVICE_NAME/
# Add a docker-compose.yml file and any necessary supporting
# files.
# Then, start the service as follows:
#  systemctl daemon_reload
#  systemctl start docker-compose@YOUR_SERVICE_NAME
#  systemctl enable docker-compose@YOUR_SERVICE_NAME
###############################################################
[Unit]
Description=%i service with docker compose
Requires=docker.service
After=docker.service
[Service]
Restart=always
WorkingDirectory=/opt/docker-services/%i
# Remove old containers, images and volumes
ExecStartPre=/usr/local/bin/docker-compose -p %i down -v
ExecStartPre=/usr/local/bin/docker-compose -p %i rm -fv
ExecStartPre=-/bin/bash -c 'for x in $(docker volume ls -qf "name=%i_"); do docker volume rm
$x; done; true'
ExecStartPre=-/bin/bash -c 'for x in $(docker network ls -qf "name=%i_"); do docker network rm
$x; done; true'
ExecStartPre=-/bin/bash -c 'for x in $(docker ps -aqf "name=%i_*"); do docker rm $x; done; true'
# Compose up
ExecStart=/usr/local/bin/docker-compose -p %i up
# Compose down, remove containers and volumes
ExecStop=/usr/local/bin/docker-compose down -v
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=docker-compose@%i
[Install]
WantedBy=multi-user.target
```

4. Reload systemd services:

```
systemctl daemon-reload
```

Installing Docker Registry Service on Registry VM

Creating service directory

1. Connect to the registry VM via SSH through VMware vCenter remote desktop.
2. Become root:

```
sudo su
```

3. Create the directory:

```
mkdir /opt/docker-services/registry/
chmod 755 /opt/docker-services/registry/
```

Creating self-signed SSL certificates

1. Connect to the registry VM via SSH through VMware vCenter remote desktop.
2. Become root:

```
sudo su
```

3. Create certificates directories:

```
mkdir /opt/docker-services/registry/ssl
mkdir /opt/docker-services/registry/ssl/ca
chmod 700 /opt/docker-services/registry/ssl
chmod 700 /opt/docker-services/registry/ssl/ca
```

4. Change to SSL directory:

```
cd /opt/docker-services/registry/ssl
```

5. Generate CA private key:

```
openssl genrsa -out ./ca/private-key.pem 2048
```

6. Generate CA public key:

```
openssl rsa -in ./ca/private-key.pem -pubout > ./ca/public-key.pem
```

7. Create CA Certificate Signing Request (CSR) and certificate:

```
openssl req -new -x509 -days 356 -key ./ca/private-key.pem -out ./ca/ca.crt
```

8. When prompted, set the following CSR fields:

```
Country Name (2 letter code) [XX]:US
State or Province Name (full name) []:North Carolina
Locality Name (eg, city) [Default City]:Durham
Organization Name (eg, company) [Default Company Ltd]:Principled Technologies
Organizational Unit Name (eg, section) []:Testing
Common Name (eg, your name or your server's hostname) []:registry-ca.local
Email Address []:nobody@principledtechnologies.com
```

9. Generate private key:

```
openssl genrsa -out ./private-key.pem 2048
```

10. Generate public key:

```
openssl rsa -in ./private-key.pem -pubout > ./public-key.pem
```

11. Create Certificate Signing Request (CSR) and self-signed certificate:

```
openssl req -new -x509 -days 356 -key ./private-key.pem -out ./ca/ca.crt
```

12. When prompted, set the following CSR fields:

```
Country Name (2 letter code) [XX]:US
State or Province Name (full name) []:North Carolina
Locality Name (eg, city) [Default City]:Durham
Organization Name (eg, company) [Default Company Ltd]:Principled Technologies
Organizational Unit Name (eg, section) []:Testing
Common Name (eg, your name or your server's hostname) []:registry.local
Email Address []:nobody@principledtechnologies.com
```

13. Create the configuration file.
14. Connect to the registry VM via SSH through VMware vCenter remote desktop.
15. Become root:

```
sudo su
```

16. Create the file `/opt/docker-services/registry/config.yml` with the following content:

```
---
version: 0.1
log:
  fields:
    service: registry
storage:
  cache:
    blobdescriptor: inmemory
  filesystem:
    rootdirectory: /var/lib/registry
http:
  addr: :443
  headers:
    X-Content-Type-Options: [nosniff]
  tls:
    certificate: /ssl/server.crt
    key:         /ssl/private-key.pem
health:
  storagedriver:
    enabled: true
    interval: 10s
    threshold: 3
```

Creating the docker-compose file

1. Connect to the registry VM via SSH through VMware vCenter remote desktop.
2. Become root:

```
sudo su
```

3. Create the file `/opt/docker-services/registry/docker-compose.yml` with the following content:

```
---
version:        "3.7"
services:
  registry:
    image:      registry:latest
    hostname:   registry
    ports:
      -         443:443
    volumes:
      -         "/opt/docker-services/registry/config.yml:/etc/docker/registry/config.yml:ro"
      -         "/opt/docker-services/registry/content:/var/lib/registry"
      -         "/opt/docker-services/registry/ssl:/ssl:ro"
```

Starting and enabling registry service

1. Connect to the registry VM via SSH through VMware vCenter remote desktop.
2. Become root:

```
sudo su
```

3. Reload systemd services:

```
systemctl daemon-reload
```

4. Enable the registry service:

```
systemctl enable docker-compose@registry.service
```

5. Start the registry service:

```
systemctl start docker-compose@registry.service
```

Building MLPerf Image on builder VM

Downloading source code

1. Connect to the builder VM via SSH through VMware vCenter remote desktop.
2. Become root:

```
sudo su
```

3. Install prerequisite system packages:

```
apt-get install -y wget curl htop git jq sshpass rsync build-essential cmake
```

4. Create source code directory:

```
mkdir /data/mlperf-inference-v2.1
chmod 777 /data/mlperf-inference-v2.1
```

5. Clone source repository:

```
git clone https://github.com/mlcommons/inference_results_v2.1.git /data/mlperf-inference-v2.1
```

Downloading dataset

1. Connect to the builder VM via SSH through VMware vCenter remote desktop.
2. Become root:

```
sudo su
```

3. Follow the instructions on https://image-net.org/download.php to download ILSVRC2012, or download an ILSVRC2012-compatible dataset and save it on the builder VM in the folder `/opt/pt/datasets/ilsvrc2012`.

Modifying source code

1. Connect to the builder VM via SSH through VMware vCenter remote desktop.
2. Become root:

```
sudo su
```

3. Change into the mlperf directory:

```
cd /data/mlperf-inference-v2.1/Closed/Dell
```

4. Replace the content of `configs/resnet50/Offline/custom.py`:

```
#!/bin/false
from . import *
@ConfigRegistry.register(HarnessType.LWIS, AccuracyTarget.k_99, PowerSetting.MaxP)
class R750_GRID_A100_40Cx1(OfflineGPUBaseConfig):
    system = KnownSystem.R750_GRID_A100_40Cx1
    gpu_batch_size = 2048
    gpu_copy_streams = 6
    gpu_inference_streams = 2
    offline_expected_qps = 34000
    run_infer_on_copy_streams = True
```

5. Change `40C` to match the desired vGPU slice type (`4C`, `5C`, … `40C`).
6. Change batch size, copy streams, inference streams, and expected qps, to optimize mlperf resnet50 inference throughput in offline mode with the selected vGPU slice type.
7. Edit code/common/systems/system_list.py as follows:
   a. Below the line that reads:

```
# A100_PCIe_40GB and 80GB based systems:
```

Add the following line:

```
add_systems('R750_GRID_A100_40Cx{}', 'R750_GRID-A100-40Cx{}', MatchAllowList([KnownCPU.x86_64_
Generic.value]), KnownGPU.GRID_A100_40C.value, [1], Memory(1, ByteSuffix.GiB))
```

   b. Change occurrences of `40C` to match the desired vGPU slice type (`4C`, `5C`, … `40C`).

8. Edit code/common/systems/known_hardware.py as follows:
   a. Below the line that reads:

```
class KnownGPU(MatchableEnum):
```

Add the following line (including indentation on the first line):

```
    GRID_A100_40C =  GPU( ‘GRID A100-40C’, AcceleratorType.Discrete, match_float_approximate(
Memory(40, ByteSuffix.GiB) ), None, “0x20F110DE”, 80)
```

   b. Change occurrences of `40C` to match the desired vGPU slice type (`4C`, `5C`, … `40C`).
   c. Remove dangling references to known systems:

```
cat code/common/systems/system_list.py | \
grep -vE '^ +KnownSystem\.R750xa_A100_PCIE_80GBx4,$' \
grep -vE '^ +KnownSystem\.XE8545_A100_SXM_80GBx4,$'  \
> code/common/systems/system_list.py.tmp
mv code/common/systems/system_list.py{.tmp,}
```

Prebuilding the mlperf image

1. Connect to the builder VM via SSH through VMware vCenter remote desktop.
2. Become root:

```
sudo su
```

3. Change into the mlperf directory:

```
cd /data/mlperf-inference-v2.1/Closed/Dell
```

4. Run the prebuild phase of the makefile:

```
export DOCKER_DETACH=1 && make prebuild
```

Building mlperf and downloading the models

1. Connect to the builder VM via SSH through VMware vCenter remote desktop.
2. Become root:

```
sudo su
```

3. Launch the build in a Docker container:

```
docker run -it -w /work \
  -v /etc/timezone:/etc/timezone:ro \
  -v /etc/localtime:/etc/localtime:ro \
  -v /data/mlperf-inference-v2.1/Closed/Dell/scratch:/scratch:rw \
  -v /opt/pt/datasets/ilsvrc2012:/scratch/preprocessed_data/imagenet:rw \
  -v /model:/scratch/models:ro \
  -v /data/mlperf-inference-v2.1/Closed/Dell:/work \
  -v /data/mlperf-inference-v2.1/Closed/Dell/closed/NVIDIA/data_maps/imagenet/cal_map.txt: /work/
data_maps/imagenet/cal_map.txt:ro
  -v /data/mlperf-inference-v2.1/Closed/Dell/closed/NVIDIA/data_maps/imagenet/val_map.txt: /work/
data_maps/imagenet/val_map.txt:ro
  --cap-add SYS_ADMIN \
  --security-opt apparmor=unconfined \
  --security-opt seccomp=unconfined \
  --name mlperf-inference-user \
  -h mlperf-inference-userv2.1 \
  --add-host mlperf-inference-userv2.1:127.0.0.1 \
  --net host \
  --device /dev/fuse \
  -e MLPERF_SCRATCH_PATH=/scratch \
  -e NVIDIA_MIG_CONFIG_DEVICES=/scratch \
mlperf-inference:user \
bash -c 'export DEBIAN_FRONTEND=noninteractive && apt-get install -y tree jq htop git && make
-j download_model BENCHMARKS=resnet50 && make -j build && make -j generate_engines RUN_ARGS="--
benchmarks=resnet50 --scenarios=Offline,Server --config_ver=default"'
```

4. List Docker containers and take note of the build container’s ID:

```
docker container ls --all
```

5. Commit the container, replacing CONTAINER_ID with the build container’s ID:

```
docker commit CONTAINER_ID mlperf:pre-assembly
```

Creating self-contained mlperf image

1. Connect to the builder VM via SSH through VMware vCenter remote desktop.
2. Become root:

```
sudo su
```

3. Create Docker build directory and subdirectories:

```
mkdir /opt/mlperf-docker-image
mkdir /opt/mlperf-docker-image/dataset
mkdir /opt/mlperf-docker-image/model
mkdir /opt/mlperf-docker-image/code
```

4. Stage mlperf files in image build directory:

```
rsync -rav /opt/pt/datasets/ilsvrc2012/ /opt/mlperf-docker-image/dataset/
rsync -rav /model/ /opt/mlperf-docker-image/model/
rsync -rav /data/mlperf-inference-v2.1/closed/Dell/ /opt/mlperf-docker-image/code/
cp /data/mlperf-inference-v2.1/closed/NVIDIA/data_maps/imagenet/cal_map.txt /opt/mlperf-docker-
image/code/data_maps/imagenet/cal_map.txt
cp /data/mlperf-inference-v2.1/closed/NVIDIA/data_maps/imagenet/val_map.txt /opt/mlperf-docker-
image/code/data_maps/imagenet/val_map.txt
```

5. Create dockerfile at `/opt/mlperf-docker-image/Dockerfile` with the following content:

```
FROM mlperf:pre-assembly
RUN mkdir -p /scratch/preprocessed_data && mkdir -p /scratch/models && mkdir -p /work
ENV PREPROCESSED_DATA_DIR=/scratch/preprocessed_data
ENV MLPERF_SCRATCH_PATH=/scratch
ENV NVIDIA_MIG_CONFIG_DEVICES=all
ADD --chown=root:root dataset/ /scratch/preprocessed_data/
ADD --chown=root:root model/ /scratch/models/
ADD --chown=root:root code/ /work/
WORKDIR /work
```

6. Change directories:

```
cd /opt/mlperf-docker-image
```

7. Invoke Docker build:

```
docker build -h mlperf-inference-userv2.1 --net host -t mlperf:gpu-grid_a100_40cx1-latest -t
registry.local/pt/mlperf:gpu-grid_a100_4cx1-latest
```

8. Change `40C` to match the desired vGPU slice type (`4C`, `5C`, … `40C`).
9. Push the image to the registry:

```
docker push registry.local/pt/mlperf:gpu-grid_a100_40cx1-latest
```

10. Change `40C` to match the desired vGPU slice type (`4C`, `5C`, … `40C`).

Powering off the Builder VM

1. Log into vCenter.
2. In the inventory view, locate the builder VM.
3. Right-click the VM.
4. Click Power.
5. Click Power Off.

Creating vGPU-enabled MachineClass for VMware Tanzu

1. Log into vCenter.
2. In the namespaces panel, click the tkg-gpu-ns.
3. In the VM service panel, click MANAGE VM CLASSES.
4. In the popup window, click MANAGE VM CLASSES.
5. Under Available VM Classes, click CREATE VM CLASS.
6. Set the name to `a100-40c`
7. Change `40c` to match the desired vGPU slice type (`4c`, `5c`, … `40c`).
8. Select the number of vCPUs, Memory, and Resource Reservation.
9. Click the PCI Devices checkbox next to Add Advanced Configuration.
10. Click Next.
11. From the ADD PCI Device drop-down menu, select NVIDIA vGPU.
12. From the Model drop-down menu, select NVIDIA A100-PCIE-40GB.
13. Set GPU Sharing to Time Sharing.
14. Set GPU Mode to Compute.
15. Set GPU memory to match the desired vGPU slice (i.e., 4, 5, … 40 for 4C, 5C, … 40C).
16. Set the number of vGPUs to 1 (when permitted to do so).
17. Click Next.
18. Click Finish.

Adding support for external registry to VMware vSphere with Tanzu

1. From the Tanzu CLI VM, log into the Supervisor Cluster:

```
kubectl vsphere login --insecure-skip-tls-verify --server=https://10.218.201.201--vsphere-username
administrator@vsphere.local
```

2. Copy the CA certificate from the registry VM to the Tanzu CLI VM:

```
rsync -av ubuntu@registry.local/etc/docker-services/registry/ssl/ca/ca.crt ./ca.crt
```

3. Convert the CA certificate to base64:

```
cat ca.crt | base64
```

4. Create a service configuration yaml file service-config.yaml, replacing XXXXXXXXXXXX with the base64 encoded CA certificate:

```
apiVersion:                     run.tanzu.vmware.com/v1alpha2
kind:                           TkgServiceConfiguration
metadata:
  name:                         tkg-service-configuration
spec:
  defaultCNI:                   antrea
  trust:
    additionalTrustedCAs:
      - name:                   Registry CA
        data:                   XXXXXXXXXXXX
```

5. Apply the configuration file:

```
kubectl apply -f service-config.yaml
```

Creating the Workload Cluster

1. From the Tanzu CLI VM, log into the Supervisor Cluster:

```
kubectl vsphere login --insecure-skip-tls-verify --server=https://10.218.201.201--vsphere-username
administrator@vsphere.local
```

2. Create a cluster.yaml file as follows:

- Notes:
  - Replace NUM_WORKERS with the number of vGPU slices possible, so 1 for 40c, and 10 for 4c
  - Change `40c` in the worker vmClass to match the desired vGPU slice type (`4c`, `5c`, … `40c`)

```
---
apiVersion:           run.tanzu.vmware.com/v1alpha2
kind:                 TanzuKubernetesCluster
metadata:
  name:               "tkg-cluster-gpu-ubuntu"
  namespace:          "tkg-gpu-ns"
spec:
  topology:
    controlPlane:
      replicas:       1
      vmClass:        "best-effort-xlarge"
      storageClass:   "tkg-clusters"
      tkr:
        reference:
          name:       "v1.21.6---vmware.1-tkg.1"
    nodePools:
    - name:           workers
      replicas:       1
      vmClass:        "a100-40c"
      storageClass:   "tkg-clusters"
      volumes:
        - name:       containerd
          mountPath:  /var/lib/containerd
          capacity:
            storage:  300Gi
      tkr:
        reference:
          name:       "v1.21.6---vmware.1-tkg.1"
  settings:
    network:
      cni:
        name:         antrea
      services:
        cidrBlocks:   [ "10.96.1.0/24"  ]
      pods:
        cidrBlocks:   [ "172.16.0.0/16" ]
```

3. Create the cluster:

```
kubectl apply -f cluster.yaml
```

4. Create the cluster login script /cluster-login.sh:

```
#!/bin/bash
export KUBECTL_VSPHERE_PASSWORD='Password1!'
kubectl vsphere login --insecure-skip-tls-verify  \
--server=10.222.201.201 \
--vsphere-username "Administrator@vsphere.local" \
--tanzu-kubernetes-cluster-name "tkg-cluster-gpu-ubuntu" \
--tanzu-kubernetes-cluster-namespace "tkg-gpu-ns" && \
kubectl config use-context "tkg-gpu-ns" &&\
kubectl config get-contexts
```

5. Make cluster login script executable:

```
chmod +x /cluster-login.sh
```

6. Log into the new cluster:

```
/cluster-login.sh
```

7. Set node labels:

```
kubectl config use-context tkg-gpu-ns
n=0
for name in $(kubectl get nodes | grep -v control-plane | tail -n +2 | awk '{print $1}'); do
  kubectl label nodes ${name} --overwrite vgpu.type=40c
  kubectl label nodes ${name} --overwrite vgpu.index=$n
  n=$(( n + 1 ))
done
```

Prepopulating the MLPerf Image in worker node cache

1. Log into the new cluster:

```
/cluster-login.sh
```

2. Create and start image pre-population jobs:
3. Note: Change 40c to match the desired vGPU slice type (4c, 5c, … 40c):

```
kubectl config use-context tkg-gpu-ns
n=0
for name in $(kubectl get nodes | grep -v control-plane | tail -n +2 | awk '{print $1}'); do
   echo "{ apiVersion: batch/v1, kind: job, metadata: { name: populate-job-$n, namespace:
default }, spec: { template: { spec: { containers: [ { name: populate-ctr-$n, image: registry.
local/pt/mlperf:gpu-grid_a100_40cx1-latest, command: [ \"/bin/bash\", \"-c\" ], args: [ \"date
-u\" ], securityContext: { capabilities: { add: [ SYSLOG, SYS_ADMIN, DAC_READ_SEARCH ] } } } ],
restartPolicy: Never, nodeSelector: { vgpu.index: $n } } }, backoffLimit: 0, podFailurePolicy: {
rules: [ { action: Ignore, onExitCodes: { operator: NotIn, values: [0]} } ] } } }" > job-$n.yaml
  kubectl apply -f job-$n.yaml
  n=$(( n + 1 ))
done
```

4. Wait for all jobs to complete, periodically monitoring status:

```
kubectl get jobs -o yaml
```

5. Delete Jobs.
6. Change n to the number of slices/containers/nodes/VMs (i.e., 1 for 40c and 10 for 4c):

```
for n in $(seq 0 $((N-1)) ); do
  kubectl delete jobs/populate-job-$n
done
```

Installing entrypoint script on the Tanzu CLI VM

1. Contact support@principledtechnologies.com to get a copy of `tanzu-mlperf-entrypoint.sh` script.
2. Copy `tanzu-mlperf-entrypoint.sh` to `/root/entrypoint.sh` on the Tanzu CLI VM.
3. Create the configmap:

```
kubectl create configmap mlperf-entrypoint-scripts-configmap \
  --from-file=entrypoint.sh=/root/entrypoint.sh
```

Running the benchmark

Creating MLPerf job YAML file(s)

1. SSH into the Tanzu CLI VM.
2. Log into the new cluster:

```
/cluster-login.sh
```

3. Create MLPerf job yaml file(s):

- Note: Change 40c to match the desired vGPU slice type (4c, 5c, … 40c).

```
kubectl config use-context tkg-gpu-ns
n=0
for name in $(kubectl get nodes | grep -v control-plane | tail -n +2 | awk '{print $1}'); do
   echo "{ apiVersion: batch/v1, kind: job, metadata: { name: mlperf-job-$n, namespace: default
}, spec: { template: { spec: { volumes: [ { name: mlperf-entrypoint-scripts-volume, configMap: {
name: mlperf-entrypoint-scripts-configmap, defaultMode: 0777 } } ], containers: [ { name: mlperf-
ctr-$n, image: registry.local/pt/mlperf:gpu-grid_a100_40cx1-latest, command: [ \"/bin/bash\", \"-
c\"], args: [ \"/entrypoint-scripts/entrypoint.sh $n\"], volumeMounts: [ {name: mlperf-entrypoint-
scripts-volume, mountPath: /entrypoint-scripts} ], securityContext: { capabilities: { add: [ SYSLOG,
SYS_ADMIN, DAC_READ_SEARCH ] } } } ], restartPolicy: Never, nodeSelector: { vgpu.index: \"$n\" } }
}, backoffLimit: 0, podFailurePolicy: { rules: [ { action: Ignore, onExitCodes: { operator: NotIn,
values: [0] } } ] } } }" > mlperf-job-$n.yaml
  n=$(( n + 1 ))
done
```

Starting performance monitoring

1. SSH into the ESXi host.
2. Change to the directory where you installed the monitor-service.sh script:

```
cd /THE/PATH/TO/monitor-service.sh
```

3. Start monitoring:

```
./monitor-service.sh start
```

Running MLPerf job(s)

1. SSH into the Tanzu CLI VM.
2. Log into the new cluster:

```
/cluster-login.sh
```

3. Start the jobs.
4. Change n to the number of slices/containers/nodes/VMs (i.e., 1 for `40c` and 10 for `4c`):

```
For n in $(seq 0 $((N-1)) ); do
  kubectl apply -f mlperf-job-$n.yaml
done
```

5. Wait for the jobs to complete, periodically monitoring status:

```
kubectl get jobs -o yaml
```

6. Collect log files:

```
for name in $(kubectl get pods | grep mlperf- | awk '{print $1}' ); do
  kubectl logs pods/$name > $name.log
done
```

7. Delete the jobs:

```
for name in $(kubectl get jobs | grep mlperf- | awk '{print $1}' ); do
  kubectl delete jobs/$name
done
```

Stopping/collecting performance monitoring

1. SSH into the ESXi host.
2. Change to the directory where you installed the `monitor-service.sh` script:

```
cd /THE/PATH/TO/monitor-service.sh
```

3. Stop monitoring:

```
./monitor-service.sh stop
```

4. Copy files to your local machine:

```
nvsmi.stdout nvsmi.stderr esxtop.stdout esxtop.stderr
```

Read the report at https://facts.pt/Hi5jvB2 ▶

This project was commissioned by Dell Technologies.

Principled Technologies®

Facts matter.®

DISCLAIMER OF WARRANTIES; LIMITATION OF LIABILITY:
Principled Technologies, Inc. has made reasonable efforts to ensure the accuracy and validity of its testing, however, Principled Technologies, Inc. specifically disclaims any warranty, expressed or implied, relating to the test results and analysis, their accuracy, completeness or quality, including any implied warranty of fitness for any particular purpose. All persons or entities relying on the results of any testing do so at their own risk, and agree that Principled Technologies, Inc., its employees and its subcontractors shall have no liability whatsoever from any claim of loss or damage on account of any alleged error or defect in any testing procedure or result.

In no event shall Principled Technologies, Inc. be liable for indirect, special, incidental, or consequential damages in connection with its testing, even if advised of the possibility of such damages. In no event shall Principled Technologies, Inc.'s liability, including for direct damages, exceed the amounts paid in connection with Principled Technologies, Inc.'s testing. Customer's sole and exclusive remedies are as set forth herein.