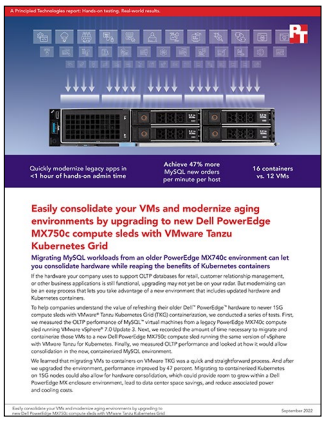

The science behind the report:

Easily consolidate your VMs and modernize aging environments by upgrading to new Dell PowerEdge MX750c compute sleds with VMware Tanzu Kubernetes Grid

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 Easily consolidate your VMs and modernize aging environments by upgrading to new Dell PowerEdge MX750c compute sleds with VMware Tanzu Kubernetes Grid.

We concluded our hands-on testing on July 25, 2022. 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 July 22, 2022 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 2: MySQL™ Tanzu modernization study performance. Note that each VM and container had 16 GB of memory with 14GB reserved.

	HammerDB transactions per minute (TPM)	HammerDB new orders per minute (NOPM)	CPU percentage utilization	Number of virtual users	Number of VMs or containers
Dell PowerEdge MX740c with traditional VMs	1,842,447	607,915	99.64%	12	12 VMs
Dell PowerEdge MX750c with Tanzu	2,717,241	897,230	99.24%	24	16 containers
Percentage win	47.48	47.59	N/A	N/A	33.33

System configuration information

Table 3: Detailed information on the systems we tested.

System configuration information	Dell PowerEdge MX740c	Dell PowerEdge MX750c
BIOS name and version	Dell 2.14.2	Dell 1.6.5
Non-default BIOS settings	Performance mode	Performance mode
Operating system name and version/ build number	VMware ESXi 7.0 U3 (Build 19898904)	VMware ESXi 7.0 U3 (Build 19898904)
Date of last OS updates/patches applied	06/22/2022	06/22/2022
Power management policy	Performance	Performance
Processor		
Number of processors	2	2
Vendor and model	Intel® Xeon® Gold 6230	Intel Xeon Gold 6330
Core count (per processor)	20	28
Core frequency (GHz)	2.1	2.0
Stepping	7	6
Memory module(s)		
Total memory in system (GB)	192	384
Number of memory modules	12	12
Vendor and model	Samsung® M393A2K43BB1-CTD	Hynix HMAA4GR7AJR8N-XN
Size (GB)	16	32
Type	DDR-4 Dual-Rank	DDR-4 Dual-Rank
Speed (MHz)	2,666	3,200
Speed running in the compute sleds (MHz)	2,666	3,200
Storage controller		
Vendor and model	Dell HBA330 MX	Dell HBA350i MX
Cache size (GB)	0 MB	0 MB
Firmware version	16.17.01.00	17.15.08.00
Driver version	17.00.12.00	19.00.03.00
Local storage (type A)		
Number of drives	1	1
Drive vendor and model	Micron® MTFDDAV240TDU	Micron MTFDDAV240TDU
Drive size (GB)	240	240
Drive information (speed, interface, type)	6Gbps SATA M.2 SSD	6Gbps SATA M.2 SSD

System configuration information	Dell PowerEdge MX740c	Dell PowerEdge MX750c
Local storage (type B)		
Number of drives	4	4
Drive vendor and model	Samsung PM1735	Samsung PM1735
Drive size (GB)	1.6	3.2
Drive type	NVMe SSD	NVMe SSD
Network adapter (type A)		
Vendor and model	Intel Ethernet 25G 2P XXV710 Mezz	Broadcom® 25G 4P BCM57504 Mezz
Number and type of ports	2x 25GbE	2x 25GbE
Driver version	20.5.13	21.80.15.60
Network adapter (type B)		
Vendor and model	N/A	QLogic QME2742 2x32Gb FC HBA
Number and type of ports	N/A	2x 32GbE
Driver version	N/A	15.35.08

Table 4: Detailed information for the enclosure.

Enclosure configuration information	Dell PowerEdge MX7000
I/O modules	
Vendor and model number	MX9116n Fabric Engine
Occupied bay(s)	1, 2
Cooling fans	
Total number of fan modules	9
Vendor and model number	Dell EMC
Power supplies	
Number of power supplies	6
Vendor and model number	Dell EMC
Wattage of each (W)	3,000

How we tested

We created a base Ubuntu virtual machine under test (VMUT) on a 14G compute sled as well as a client VM on an infrastructure compute sled. After configuring the VMUT with MySQL and the client with HammerDB, we built a 500-warehouse TPROC-C database on the VMUT and created a backup. We cloned the VMUT and client until we had a total of 12 each, then ran the TPROC-C workload against each VMUT.

Next, we created our Kubernetes (K8s) environment by installing Tanzu Kubernetes Grid Standalone via a utility VM in our vCenter. We deployed a K8s management cluster as well as a workload cluster with 16 nodes to a 15G compute sled. We deployed 16 MySQL pods and attached them to the workload nodes, scaled up the number of clients on our infrastructure compute sled to 16, and ran the TPROC-C tests again on the 15G compute sled.

Installing ESXi 7.0 U3 on the compute sleds under test (CSUT)

We followed the steps below to install ESXi 7.0 on a Dell PowerEdge MX740c and a Dell PowerEdge MX750c compute sled. We also installed ESXi 7.0 on an additional PowerEdge MX750c compute sled, which we used as our infrastructure compute sled to host our vCenter and TPROC-C client VMs.

Changing system profile to performance

1. Boot the Dell PowerEdge MX740c compute sled.
2. At the POST menu, select F2 for System Setup.
3. Select System BIOS.
4. Select System Profile Settings.
5. Click the drop-down menu for System Profile, and select Performance.
6. Click Back.
7. Click Finish.
8. Click Finish again, and reboot the compute sled.
9. Repeat steps 1 through 8 for the Dell PowerEdge MX750c compute sled.

Installing ESXi vSphere 7.0 U3

1. Attach the installation media.
2. Boot the compute sled.
3. At the POST menu, select F11 for Boot Manager.
4. Select One-shot UEFI Boot Menu.
5. Select the installation media.
6. At the VMware Installer screen, press Enter.
7. At the EULA screen, press F11 to Accept and Continue.
8. Under Storage Devices, select the appropriate virtual disk, and press Enter.
9. For the keyboard layout, select US, and press Enter.
10. Enter the root password twice, and press Enter.
11. To start the installation, press F11.
12. After the compute sled reboots, press F2, and enter root credentials.
13. Select Configure Management Network, and press Enter.
14. Select the appropriate network adapter, and click OK.
15. Select IPv4 settings, and enter the desired IP address, subnet mask, and gateway for the compute sled.
16. Select OK, and restart the management network.
17. Repeat steps 1 through 16 on the remaining compute sleds.

Deploying VMware vCenter 7.0 U3

1. On a Windows machine or VM, locate the VMware-VCSA installer image.
2. Mount the image, navigate to the vcsa-ui-installer folder, and double-click win32.
3. Double-click installer.exe
4. Click Install.
5. Click Next.
6. Accept the terms of the license agreement, and click Next.
7. Leave the default vCenter Server with an Embedded Platform Services Controller selected, and click Next.
8. Enter the FQDN or IP address of the host onto which you will deploy vCenter Server Appliance.

9. Provide the compute sled's credentials, and click Next.
10. At the Configure Network Settings page, configure the network settings for your environment, and click Next.
11. Review your settings, and click Finish.
12. When the deployment completes, click Next.
13. At the Introduction page, click Next.
14. At the Appliance configuration page, select the time synchronization mode and SSH access settings, and click Next. We used a local NTP server and enabled SSH.
15. Select Create a new SSO domain.
16. Provide a password, and confirm it.
17. Provide an SSO Domain name and SSO Site name, and click Next.
18. At the CEIP screen, click Next.
19. At the Ready to complete screen, click Finish.
20. When the installation completes, click Close.
21. Using the vSphere web client, log into the vCenter server using the credentials you previously provided.

Creating a cluster in VMware vCenter and adding hosts

Creating the CSUT cluster

1. Once you have logged into the vCenter, navigate to Hosts and Clusters.
2. Select the primary site management vCenter.
3. Right-click the vCenter object, and select New Datacenter…
4. Enter a name for the new datacenter, and click OK.
5. Right-click the new datacenter, and click New Cluster…
6. Name for the new cluster CSUT.
7. Click OK.
8. Repeat steps 1 through 7 to create an Infra cluster.

Adding the hosts

1. Once you have created the cluster, right-click it, and select Add Host.
2. Enter the FQDN or IP address of the first host, and click Next.
3. Enter the root credentials for the compute sled, and click Next.
4. To accept the compute sled's certificate, click Yes.
5. Review the compute sled details, and click Next.
6. Assign the desired license, and click Next.
7. Disable Lockdown mode, and click Next.
8. Click Finish.

Creating the 14G bare VMs

For the OLTP workload, we tested Ubuntu 20.04 LTS with MySQL v8.0.29. We created a base VM with Ubuntu, then altered the settings to the necessary vCPUs, memory, and drive sizes before installing MySQL and cloning out the VMs for the workload. This section has the steps to create the base VM, to install and configure Ubuntu, and to install and configure MySQL. See the workload section for the specific VM configurations and MySQL adjustments.

Creating the base MySQL and client VMs

1. Launch the VMware vCenter HTML5 client.
2. Right-click the MX740c host, and choose New Virtual Machine…
3. Choose Create a new virtual machine, and click Next.
4. Name the virtual machine, and click Next.
5. Choose the correct host, and click Next.
6. Choose the correct datastore, and click Next.
7. Choose ESXi 7.0 and later, and click Next.
8. Choose Linux for the Guest OS Family, and Ubuntu Linux (64-bit) for Guest OS Version. Click Next.
9. Set the vCPU count to 6, memory to 16GB, New Hard Disk capacity to 20GB, choose the proper network from the drop-down menu, and attach the CD drive. Add two more New Hard Disks. Set the database volume to 100GB, and the log volume to 20GB.
10. Double-check your settings on the summary page, and click Finish.

11. To create the base client VM, right-click the infrastructure host, and choose New Virtual Machine…
12. Repeat steps 3 through 10, adjusting the settings in step 9 to the following:
    - vCPU count: 4
    - Memory: 8GB
    - New Hard Disk: 20GBInstalling Ubuntu 20.04 LTS on the bare VMs
13. In vCenter, power on the newly created VM.
14. Launch the virtual console.
15. Select Install Ubuntu Server.
16. Select English.
17. Select Update to the new installer.
18. On the Keyboard configuration screen, click Done.
19. On the Network connections screen, make sure you have a DHCP address (or set a static address), and click Done.
20. On the Configure proxy screen, click Done.On the Configure Ubuntu archive mirror screen, click Done.
21. Select Use an entire disk, set the disk to your desired OS volume, and click Done.
22. Review the storage configuration, and click Done.
23. On the Confirm destructive action prompt, click Continue.
24. Enter your name, server name, username, and password, and click Done.
25. Check the box for Install OpenSSH server, and click Done.
26. On the Featured Server Snaps screen, leave everything unchecked, and click Done.
27. Once the installation is complete, press Enter to remove the installation media and reboot the compute sled.

Configuring Ubuntu 20.04 LTS and installing MySQL on the MySQL instance

1. Log into the MySQL instance via SSH.
2. Run the "mysql_host_prepare.sh" script:

```
sudo ./mysql_host_prepare.sh
```

3. Download the appropriate MySQL bundle for Ubuntu from https://dev.mysql.com/downloads/mysql/.
4. Extract the MySQL bundle:

```
tar -xf mysql-server_8.0.29-1ubuntu20.04_amd64.deb-bundle.tar
```

5. Install MySQL Community Server 8 and all dependencies:sudo dpkg -i mysql-community-server_8.0.29-1ubuntu20.04_amd64.deb
6. Stop the MySQL service:

```
sudo systemctl stop mysql
```

7. Copy the MySQL data directory to your data disk:

```
sudo cp -R -p /var/lib/mysql /mnt/mysqldata/
```

8. Copy the appropriate my.cnf config file over:

```
cp -p /etc/my.cnf{,.bak}
cp -f my.cnf /etc/my.cnf
```

9. Start the MySQL service:

```
sudo systemctl start mysql
```

10. Log into the MySQL instance as the root user:

```
mysql -u root -p
```

11. Create a new user named mysql with full permissions:

```
CREATE USER 'mysql'@'localhost' IDENTIFIED BY '[password]';
GRANT ALL PRIVILEGES ON *.* TO 'mysql'@'localhost'
->  WITH GRANT OPTION;
CREATE USER 'mysql'@'%' IDENTIFIED BY '[password]';
GRANT ALL PRIVILEGES ON *.* TO 'mysql'@'%'
->  WITH GRANT OPTION;
```

12. Enable mysql_native_password authentication on the mysql user:

```
ALTER USER 'mysql'@'localhost' IDENTIFIED WITH mysql_native_password BY '[password]';
ALTER USER 'mysql'@'%' IDENTIFIED WITH mysql_native_password BY '[password]';
```

13. Shut down the instance:sudo poweroff

Configuring Ubuntu and installing HammerDB 4.4 on the MySQL client instance

1. Log into the HammerDB instance via SSH.
2. Turn off SSH strict host key checking:echo 'StrictHostKeyChecking no' > .ssh/config

```
chmod 400 ~/.ssh/config
```

3. Install the required packages:sudo apt install -y wget vim tar zip unzip lz4 pigz nmon sysstat numactl ksh
4. Download the MySQL x86 RPM bundle from https://dev.mysql.com/downloads/mysql/.
5. Extract the MySQL bundle:

```
tar -xf mysql-server_8.0.29-1ubuntu20.04_amd64.deb-bundle.tar
```

6. Install the MySQL 8 Community Client and all of its dependencies:

```
sudo dpkg -i mysql-community-client_8.0.29-1ubuntu20.04_amd64.deb
```

7. Download HammerDB 4.4:sudo wget https://github.com/TPC-Council/HammerDB/releases/download/v4.4/HammerDB-4.4-Linux.tar.gz
8. Extract the HammerDB package:tar -xf HammerDB-4.4-Linux.tar.gz
9. Download and extract nmonchart tool:wget https://sourceforge.net/projects/nmon/files/nmonchart40.tar
   tar -xf nmonchart40.tar ./nmonchart
10. Copy all scripts and config files in the appendix section to the HammerDB MySQL client instance.
11. Shut down the instance:sudo poweroff

Create the database schema with HammerDB

1. Log into the MySQL client instance via SSH.
2. Navigate to the HammerDB directory:cd HammerDB-4.4
3. Start hammerdbcli:./hammerdbcli
4. Set the following variables:dbset db mysql
   diset connection mysql_host <IP_ADDRESS>
   diset tpcc mysql_user mysql
   diset tpcc mysql_pass <Password>
   diset tpcc mysql_count_ware <DB_SIZE>
   diset tpcc mysql_partition true
   diset tpcc mysql_num_vu 8
   diset tpcc mysql_storage_engine innodb
5. Build the schema:

```
buildschema
```

Backing up the database

1. Log into the MySQL instance.Shut down the database:systemctl stop mysql
2. Delete the log files:

```
cd /mnt/data/
rm -f data/ib_logfile*
```

3. Back up the database:

```
tar -cf- data/ | pigz -9 -c > mysql_tpcc_<DB_SIZE>warehouses_data.tar.gz
```

Cloning the 14G MySQL and client VMs

1. Once you have installed Ubuntu and MySQL, configured all the settings, and created the database, you can clone your VMs.
2. Open the vCenter console, and ensure that the VM is powered off.
3. Right-click the MySQL VM → Clone → Clone to Virtual Machine…
4. Name the VM, select your datacenter, and click Next.
5. Choose the host for your VM, and click Next.
6. Choose the datastore, and from the VM Storage Policy drop-down menu, select Keep existing VM storage policies. Click Next.
7. Click Next.
8. Click Finish.
9. Repeat steps 2 through 8 until you have a total of 12 MySQL VMs.
10. Ensure that the gold client VM is powered off.
11. Repeat steps 3 through 5.
12. Leave the default VM storage policy Keep existing VM storage policies, and click Next.
13. Click Next.
14. Click Finish.
15. Repeat steps 11 through 14 until you have 12 client VMs total.

Running the 14G tests

1. Start esxtop collection on 14G host.
2. Log into each of the HammerDB MySQL client instances via SSH.
3. Execute the hdb_tpcc_mysql_500wh.tcl script. Parameters and config options can be tuned by modifying the script and editing the variables at the start of the file.

```
./hammerdbcli auto hdb_tpcc_mysql_500wh.tcl
```

4. Stop esxtop and collect performance metrics and results. Repeat test two more times, keeping the median run.

Creating the utility VM

1. Launch the VMware vCenter HTML5 client.
2. Right-click the infrastructure host and choose New Virtual Machine…
3. Choose Create a new virtual machine, and click Next.
4. Name the virtual machine, and click Next.
5. Choose the correct host, and click Next.
6. Choose the correct datastore, and click Next.
7. Choose ESXi 7.0 and later, and click Next.
8. Choose Linux for the Guest OS Family, and Ubuntu Linux (64-bit) for Guest OS Version. Click Next.
9. Set the vCPU count to 4, memory to 8GB, New Hard Disk capacity to 50GB, choose the proper network from the drop-down menu, and attach the CD drive.
10. Double-check your settings on the summary page and click Finish.
11. Install Ubuntu 20.04 LTS according to the steps outlines in "Installing Ubuntu 20.04 LTS on the bare VMs".

Installing prerequisite software on the utility VM

We installed TKG Standalone following the installation steps in the VMware documentation: https://docs.vmware.com/en/VMware-Tanzu-Kubernetes-Grid/1.5/vmware-tanzu-kubernetes-grid-15/GUID-install-cli.html

1. Log onto the utility VM as the local user.
2. Install Docker on Ubuntu following the instructions at https://docs.docker.com/engine/install/ubuntu/:

```
sudo apt-get remove docker docker-engine docker.io containerd runc
sudo apt-get update
sudo apt-get install apt-transport-https ca-certificates curl gnupg lsb-release
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \
  sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] \
  https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

3. Add the local user to the Docker group:

```
sudo usermod -a -G docker ubuntu
```

4. Logout and log back in as the local user `ubuntu`
5. Start docker.

```
sudo service docker start
```

6. Create SSH keys:

```
ssh-keygen -t rsa -b 4096 -C ubuntu@ubu01
```

7. Download the packages for TKG CLI version 1.5.4 (`tanzu-cli-bundle-linux-amd64.tar`), TKG kubectl version1.22.9 (`kubectl-linux-v1.22.9+vmware.1.gz`), and one Ubuntu OS images (`ubuntu-2004-kube-v1.22.9+vmware.1-tkg.1-2182cbabee08edf480ee9bc5866d6933.ova`) from the VMware website: https://www.vmware.com/go/get-tkg.
8. Install the TKG CLI, TKG kubectl, and assorted tools on the VM:

```
tar -xf tanzu-cli-bundle-linux-amd64.tar
sudo install cli/core/v0.11.6/tanzu-core-linux_amd64 /usr/local/bin/tanzu
gunzip kubectl-linux-v1.22.9+vmware.1.gz
sudo install kubectl-linux-v1.22.9+vmware.1 /usr/local/bin/kubectl
gunzip cli/{imgpkg,kapp,kbld,ytt}-*.1.gz
for i in imgpkg kapp kbld ytt ; do
  sudo install cli/${i}-linux-amd64-*vmware.1 /usr/local/bin/${i}
done
```

9. Install the Tanzu cli plugins:

```
tanzu plugin install --local cli all
```

10. Create a VM folder with the vSphere client. Right-click the datacenter icon and select New Folder → New VM and template folder. In the pop-up window, enter the name of the folder.
11. Upload a base image for the nodes by r
12. ight-clicking on the datacenter and selecting Deploy an OVF Template.
13. Select Local file.

```
Click UPLOAD FILES and select the file
  ubuntu-2004-kube-v1.22.9+vmware.1-tkg.1-2182cbabee08edf480ee9bc5866d6933.ova
```

14. On the Select a name and folder screen, click the folder you created in step 10, and click NEXT.
15. Select the test's compute resource pool, and click NEXT.
16. On the Review details screen, click NEXT.
17. Accept the license agreement, and click NEXT.
18. Select the datastore, and click NEXT.
19. Select the cluster's network, and click NEXT.
20. On the Ready to complete screen, click FINISH.
21. When the upload completes, right-click the base-image VM and select Template → Convert to template, and click YES on the pop-up window.
22. Apply a workaround for the latest kernel:

```
sudo sysctl net/netfilter/nf_conntrack_max=131072
```

Deploying the management cluster via UI

1. Initialize the TKG environment for the local user on the VM:

```
tanzu management-cluster create --ui
```

2. Open a browser to `https://127.0.0.1:8080/`
3. In the VMware vSphere area, select DEPLOY.
4. On the vSphere 7.0.3 Environment Detected screen, click DEPLOY TKG MANAGEMENT CLUSTER.
5. Enter the following information:IP address for the vCenter ServerThe administrator account username and passwordClick CONNECT.
6. On the Verify SSL Thumbprint screen, click CONTINUE.
7. Enter or select the following information:
   - Datacenter for the cluster
   - The local account's SSH public key (`cat ~/.ssh/id_rsa.pub`)
8. Click NEXT.
9. In the Development box, do the following:
   a. Select the small INSTANCE TYPE.
   b. Enter " mgmt" for the MANAGEMENT CLUSTER NAME.
   c. Enter a local, non-DHCP IP address for the CONROL PLANE ENDPOINT; e.g., 10.211.42.5.
   d. Select the small WORKER NODE INSTACE TYPE.
10. Click NEXT.
11. To bypass the VMware NSX Advanced Load Balancer screen, click NEXT.
12. To bypass the Metadata screen, click NEXT.
13. Select the VM Folder, Datastore, and Resource pool, and click NEXT
14. Select the VM's NETWORK NAME, and click NEXT.
15. Disable Identify Management Settings, and click NEXT.
16. Select the OS IMAGE, and click NEXT.
17. To bypass the Register screen, click NEXT.
18. Unselect CEIPT, and click NEXT.
19. Click REVIEW CONFIGURATION.
20. Click DEPLOY MANGEMENT CLUSTER.
21. When the deployment of the management cluster has completed, switch back to the deployment VM's shell, and query the cluster:

```
tanzu management-cluster  get mgmt
kubectl get nodes
```

Creating the Kubernetes workload cluster

1. On the deployment VM., make a copy of the YAML cluster configuration file (which has a randomly generated name, such as 3nu4utlz5u.yaml), and name it app.yaml:

```
cp ~/.tanzu/tkg/clusterconfigs/3nu4utlz5u.yaml ~/.tanzu/tkg/clusterconfigs/app.yaml
```

2. Modify the new cluster file to change the cluster name, change their endpoint IP addresses, change the VM sizes (CPUS, memory, and local storage) for the worker nodes, and change the number of worker nodes. For example,

```
sed -i -e 's/^CLUSTER_NAME:.*/CLUSTER_NAME: app/' \
   -e 's/^VSPHERE_CONTROL_PLANE_ENDPOINT:.*/VSPHERE_CONTROL_PLANE_ENDPOINT: 10.211.42.6/' \
  ~/.tanzu/tkg/clusterconfigs/app.yaml
```

3. Create the Kubernetes cluster from its configuration file:

```
tanzu cluster create --file ~/.tanzu/tkg/clusterconfigs/app.yaml -v 6
```

4. Add the cluster-access credentials to the Kubernetes configuration, and set the Kubernetes context to that of the cluster:

```
tanzu cluster kubeconfig get app --admin
kubectl config use-context app-admin@app
```

Updating and configuring the workload cluster nodes

1. On the utility VM, obtain a list of the workload node IPs:

```
kubectl get nodes -o wide
```

2. SSH into each node:

```
ssh capv@[node-IP]
```

3. Run OS updates and upgrades:

```
sudo apt update
sudo apt upgrade -y
```

4. Create a path for the MySQL directory and open its permissions:

```
sudo mkdir -p /mnt/data
sudo chmod 777 /mnt/data
```

5. Repeat steps 2 through 4 for each node. We wrote a simple script to automate this process for the 16 nodes.

Migrating the MySQL backup from 14G bare VMs to 15G nodes

Note: By default, the utility VM is the only VM capable of SSHing to the workload cluster nodes. However, you can create additional VMs and give them permissions to SCP to and from the 15G nodes.

1. On the utility VM, create a directory for housing backups:

```
sudo mkdir /mnt/backup
sudo chmod 777 /mnt/backup
```

2. Copy the backup from the first 14G MySQL VM to the utility VM:

```
scp [user]@[14G_VM_IP]:/mnt/data/mysql_tpcc_[DB_SIZE]warehouses_data.tar.gz \ /mnt/backup/mysql_
tpcc_[DB_SIZE]warehouses_data.tar.gz
```

3. Copy the backup from the utility VM to the first 15G node:

```
scp /mnt/backup/mysql_tpcc_[DB_SIZE]warehouses_data.tar.gz \ capv@[15G_NODE_IP]:/mnt/data/mysql_
tpcc_[DB_SIZE]warehouses_data.tar.gz
```

4. Repeat step 3 for the remaining nodes. We wrote a simple script to automate this process for the 16 nodes.
5. Note: We used a single unique database for all VMs and nodes and therefore copied only one backup over to the utility VM. If you are using multiple unique databases, repeat step 2 for each.

Deploying the MySQL pods

The scripts we reference here are in scripts we used.

1. Copy over or create the following scripts and files on the utility VM:

```
storageclass.yaml; pv.yaml; pvc.yaml; dp.yaml; mtb.yaml; hdb_tpcc_mysql_500wh.tcl; my.cnf
```

2. From the utility VM, create a storage class based on the backing datastore for the 15G host:

```
kubectl apply -f storageclass.yaml
```

3. Install MetalLB to help Antrea assign external IPs from a pool:

```
kubectl apply -f \ https://raw.githubusercontent.com/metallb/metallb/v0.11.0/
manifests/namespace.yaml
kubectl apply -f \ https://raw.githubusercontent.com/metallb/metallb/v0.11.0/manifests/metallb.yaml
```

4. Configure MetalLB to use IP addresses from a pool:

```
kubectl apply -f mtb.yaml
```

5. Create the mysql namespace:

```
kubectl create ns mysql
```

6. Define a persistent volume:

```
kubectl apply -f pv.yaml
```

7. Define a persistent volume claim:

```
kubectl apply -f pvc.yaml
```

8. Create a configMap of the MySQL .cnf file:

```
kubectl -n mysql create configmap mysql-config --from-file=my.cnf
```

9. Create a K8s secret with the MySQL admin password:

```
kubectl create secret generic mysql-creds --from-literal=ROOT_PASSWORD=changeme -n mysql
```

10. Create the MySQL deployment:

```
kubectl apply -f dp.yaml
```

11. Create a service to give the MySQL deployment an external IP:

```
kubectl -n mysql expose deployment mysql --port=3306 --target-port=3306 \
--name mysql-service --type=LoadBalancer
```

12. Repeat steps 5 through 11 for the remaining nodes, updating the nodeSelector field in dp.yaml to manually choose which node each pod gets deployed to (you can remove this field if you wish to let the controller automatically place the pods).

Restoring backup from zip file

1. From the utility VM, SSH into the first workload cluster node:

```
ssh capv@[node-IP]
```

2. Remove the current MySQL data directory:

```
sudo rm -rf /mnt/data/mysql
```

3. Unzip the backup file:
4. pigz -d -c /mnt/data/mysql_tpcc_[DB_SIZE]warehouses_data.tar.gz | sudo tar -C /mnt/data -xf- ; sync
5. Repeat steps 1 through 3 on each node. We wrote a simple script to automate the process.
6. Delete all MySQL pods (the deployments you applied in the previous section will recreated them automatically):

```
kubectl -n mysql delete --all pods
```

Running the 15G tests

1. Start esxtop collection on the 15G host.
2. Log into each of the HammerDB MySQL client instances via SSH.
3. Execute the hdb_tpcc_mysql_500wh.tcl script. You can tune the parameters and config options by modifying the script and editing the variables at the start of the file.

```
./hammerdbcli auto hdb_tpcc_mysql_500wh.tcl
```

4. Stop esxtop and collect performance metrics and results. Repeat test two more times, recording the median run.

Scripts we used

mysql_host_prepare.sh

```
#!/bin/bash

sed -i -e '/vm.swappiness/d' -e '/fs.aio-max-nr/d' /etc/sysctl.conf
cat <<EOF >>/etc/sysctl.conf
vm.swappiness = 1
fs.aio-max-nr = 1048576
EOF
sysctl -p
#### Install tools ####
apt-get install -y wget vim tar zip unzip lz4 pigz nmon sysstat numactl ksh

#### Prepare storage ####
umount -v /mnt/data
mkdir -p /mnt/data

sed -i '/mysqldata/d' /etc/fstab
if [ -e /dev/nvme1n1 ]; then
  mkfs.xfs -f /dev/nvme1n1
  echo '/dev/nvme1n1  /mnt/data  xfs  defaults,nofail,x-systemd.device-timeout=5  0 2' >> /etc/fstab
else
  mkfs.xfs -f /dev/sdb
  echo '/dev/sdb  /mnt/data  xfs  defaults,nofail,x-systemd.device-timeout=5  0 2' >> /etc/fstab
fi

mount -v /mnt/data
restorecon -Rv /mnt/data
my.cnf
[mysqld]
default_authentication_plugin=mysql_native_password

# general
max_connections=4000
table_open_cache=8000
table_open_cache_instances=16
back_log=1500
default_password_lifetime=0
ssl=0
performance_schema=OFF
max_prepared_stmt_count=128000
skip_log_bin=1
character_set_server=latin1
collation_server=latin1_swedish_ci
transaction_isolation=REPEATABLE-READ

# files
innodb_file_per_table
innodb_log_file_size=1024M
innodb_log_files_in_group=8 #scale
innodb_open_files=4000

# buffers
innodb_buffer_pool_size=12800M #scale
innodb_buffer_pool_instances=16
innodb_log_buffer_size=64M

# tune
innodb_doublewrite=0
innodb_thread_concurrency=0
innodb_flush_log_at_trx_commit=0
innodb_max_dirty_pages_pct=90
innodb_max_dirty_pages_pct_lwm=10
join_buffer_size=32K
sort_buffer_size=32K
innodb_use_native_aio=1
innodb_stats_persistent=1
innodb_spin_wait_delay=6
innodb_max_purge_lag_delay=300000
```

```
innodb_max_purge_lag=0
innodb_flush_method=O_DIRECT_NO_FSYNC
innodb_checksum_algorithm=none
innodb_io_capacity=1000
innodb_io_capacity_max=2000
innodb_lru_scan_depth=9000
innodb_change_buffering=none
innodb_read_only=0
innodb_page_cleaners=4
innodb_undo_log_truncate=off

# perf special
innodb_adaptive_flushing=1
innodb_flush_neighbors=0
innodb_read_io_threads=16
innodb_write_io_threads=16
innodb_purge_threads=4
innodb_adaptive_hash_index=0

# monitoring
innodb_monitor_enable='%'
```

hdb_tpcc_mysql_500wh.tcl

```
#!/bin/tclsh
puts "SETTING CONFIGURATION"
global complete
proc wait_to_complete {} {
global complete
set complete [vucomplete]
if {!$complete} { after 5000 wait_to_complete } else { exit }
}
dbset db mysql
diset connection mysql_host [host_IP]
diset connection mysql_port 3306

diset tpcc mysql_user mysql
diset tpcc mysql_pass [password]
diset tpcc mysql_storage_engine innodb
diset tpcc mysql_partition true
diset tpcc mysql_driver timed

diset tpcc mysql_count_ware 500
diset tpcc mysql_num_vu 12
diset tpcc mysql_rampup 5
diset tpcc mysql_duration 10

vuset logtotemp 1
loadscript
vuset vu 12
vucreate
vurun
wait_to_complete
vwait forever
```

sc.yaml

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: nvme1
  annotations:
    storageclass.kubernetes.io/is-default-class: "false"
provisioner: csi.vsphere.vmware.com
parameters:
  datastoreurl: "ds:///vmfs/volumes/[volume path]/"
```

mtb.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: metallb-system
  name: config
data:
  config: |
    address-pools:
    - name: default
      protocol: layer2
      addresses:
      - 10.211.42.21-10.211.42.40
```

pv.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-persistent-storage
  namespace: mysql
  labels:
    type: local
spec:
  storageClassName: nvme1
  capacity:
    storage: 100Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data"
```

pvc.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-storage
  namespace: mysql
spec:
  storageClassName: nvme1
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
```

dp.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
  namespace: mysql
  labels:
    app: mysql
spec:
  replicas: 1
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: mysql
```

```
template:
  metadata:
    labels:
      app: mysql
  spec:
    nodeSelector:
            kubernetes.io/hostname: app-md-0-7954dc6896-vkxvs
    containers:
      - name: mysql
        image: mysql:8.0.29

        ports:
          - containerPort: 3306
        resources:
          requests:
            cpu: 300m
            memory: 400Mi
        volumeMounts:
          - name: mysql-storage
            mountPath: "/var/lib/mysql"
            subPath: "mysql"
          - name: mysql-conf
            mountPath: /etc/mysql/conf.d
        env:
          - name: MYSQL_ROOT_PASSWORD
            valueFrom:
              secretKeyRef:
                name: mysql-creds
                key: ROOT_PASSWORD
    volumes:
      - name: mysql-conf
        configMap:
          name: mysql-config
      - name: mysql-storage
        persistentVolumeClaim:
          claimName: mysql-storage
```

Read the report at https://facts.pt/a66WgAM ▶

This project was commissioned by Dell Technologies.

Principled Technologies®

Facts matter.®