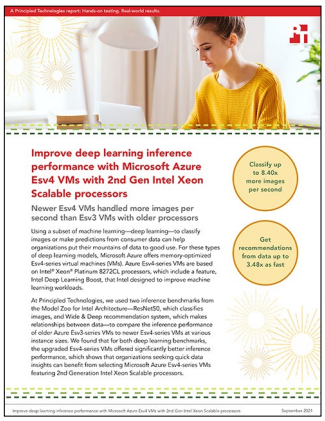

The science behind the report:

Improve deep learning inference performance with Microsoft Azure Esv4 VMs with 2nd Gen Intel Xeon Scalable processors

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 Improve deep learning inference performance with Microsoft Azure Esv4 VMs with 2nd Gen Intel Xeon Scalable processors.

We concluded our hands-on testing on August 12, 2021. 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 17, 2021 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: Relative results of our Resnet50 testing.

Microsoft Azure VM	Average images per second processed
8 vCPUs	
E8s_v3	1
E8s_v4	8.4
16 vCPUs	
E16s_v3	1
E16s_v4	6.67
64 vCPUs	
E64s_v3	1
E64s_v4	5.96

Table 2: Relative results of our Wide & Deep testing.

Microsoft Azure VM	Average samples per second processed
8 vCPUs	
E8s_v3	1
E8s_v4	3.48
16 vCPUs	
E16s_v3	1
E16s_v4	3.23
64 vCPUs	
E64s_v3	1
E64s_v4	2.99

System configuration information

Table 3: Detailed information about the ESv3 VMs we tested.

System configuration information	ESv3 VM - 8v CPU	ESv3 VM - 16 vCPU	ESv3 VM - 64 vCPU
Tested by	Principled Technologies	Principled Technologies	Principled Technologies
Test date	ResNet50: 05/04/2021 Wide and Deep: 08/12/2021	ResNet50: 05/04/2021 Wide and Deep: 08/12/2021	ResNet50: 05/04/2021 Wide and Deep: 08/12/2021
CSP and region	Microsoft Azure / East US	Microsoft Azure / East US	Microsoft Azure / East US
Workload and version	ResNet50: ResNet50v1 Wide & Deep: Intel optimized models v2.3.0	ResNet50: ResNet50v1 Wide & Deep: Intel optimized models v2.3.0	ResNet50: ResNet50v1 Wide & Deep: Intel optimized models v2.3.0
WL specific parameters	ResNet50: 50 steps, 10 warmup steps, 100 batch size Wide & Deep: 512 batch size, 46,875 batches, OMP_NUM_THREADS=4, NUM_INTRA_THREADS=4, PRECISION=INT8 or FP32	ResNet50: 50 steps, 10 warmup steps, 100 batch size Wide & Deep: 512 batch size, 46,875 batches, OMP_NUM_THREADS=8, NUM_INTRA_THREADS=8, PRECISION=INT8 or FP32	ResNet50: 500 steps, 100 warmup steps, 100 batch size Wide & Deep: 512 batch size, 46,875 batches, OMP_NUM_THREADS=16, NUM_INTRA_THREADS=16, PRECISION=INT8 or FP32
Iterations and result choice	3 runs, median	3 runs, median	3 runs, median
Server platform	E8s_v3	E16s_v3	E64s_v3
BIOS name and version	Hyper-V UEFI Release v4.1	Hyper-V UEFI Release v4.1	Hyper-V UEFI Release v4.1
Operating system name and version/build number	Ubuntu 20.04 5.8.0-1036-azure	Ubuntu 20.04 5.8.0-1036-azure	Ubuntu 20.04 5.8.0-1036-azure
Date of last OS updates/patches applied	ResNet50: 05/04/2021 Wide & Deep: 07/17/2021	ResNet50: 05/04/2021 Wide & Deep: 07/17/2021	ResNet50: 05/04/2021 Wide & Deep: 07/17/2021
Processor			
Number of processors	1	1	2
Vendor and model	Intel® Xeon® E5-2673 v4	Intel Xeon E5-2673 v4	Intel Xeon E5-2673 v4
Core count (per processor)	20	20	20
Core frequency (GHz)	2.30	2.30	2.30
Stepping	1	1	1
Hyper-threading	Yes	Yes	Yes
Turbo	Yes	Yes	Yes
Number of vCPUs per VM	8	16	64
Memory module(s)			
Total memory in system (GB)	64	128	512
NVMe memory present?	No	No	No
Total memory in GB (DDR+NVMe RAM)	64	128	512
General hardware			
Storage: NW or Direct Att / Instance	Direct Att	Direct Att	Direct Att

System configuration information	ESv3 VM - 8v CPU	ESv3 VM - 16 vCPU	ESv3 VM - 64 vCPU
Local storage			
OS			
Number of drives	1	1	1
Drive size (GB)	ResNet50: 100 Wide & Deep: 30	ResNet50: 100 Wide & Deep: 30	ResNet50: 100 Wide & Deep: 30
Drive information	Standard SSD	Standard SSD	Standard SSD
Data drive			
Number of drives	1	1	1
Drive size (GB)	ResNet50: 200 Wide & Deep: 10	ResNet50: 200 Wide & Deep: 10	ResNet50: 200 Wide & Deep: 10
Drive information	Standard SSD	Standard SSD	Standard SSD
Network adapter			
Vendor and model	Hyper-V Virtual Adapter	Hyper-V Virtual Adapter	Hyper-V Virtual Adapter
Number and type of ports	1x 40Gb	1x 40Gb	1x 50Gb

Table 4: Detailed information about the ESv4 VMs we tested.

System configuration information	ESv4 VM - 8 vCPU	ESv4 VM - 16 vCPU	ESv4 VM - 64 vCPU
Tested by	Principled Technologies	Principled Technologies	Principled Technologies
Test date	ResNet50: 05/04/2021 Wide & Deep: 08/12/2021	ResNet50: 05/04/2021 Wide & Deep: 08/12/2021	ResNet50: 05/04/2021 Wide & Deep: 08/12/2021
CSP and region	Microsoft Azure / East US	Microsoft Azure / East US	Microsoft Azure / East US
Workload and version	ResNet50: ResNet50v1 Wide & Deep: Intel optimized models v2.3.0	ResNet50: ResNet50v1 Wide & Deep: Intel optimized models v2.3.0	ResNet50: ResNet50v1 Wide & Deep: Intel optimized models v2.3.0
WL specific parameters	ResNet50: 50 steps, 10 warmup steps, 100 batch size Wide & Deep: 512 batch size, 46,875 batches, OMP_NUM_THREADS=4, NUM_INTRA_THREADS=4, PRECISION=INT8 or FP32	ResNet50: 50 steps, 10 warmup steps, 100 batch size Wide & Deep: 512 batch size, 46,875 batches, OMP_NUM_THREADS=8, NUM_INTRA_THREADS=8, PRECISION=INT8 or FP32	ResNet50: 500 steps, 100 warmup steps, 100 batch size Wide& Deep: 512 batch size, 46,875 batches, OMP_NUM_THREADS=16, NUM_INTRA_THREADS=16, PRECISION=INT8 or FP32
Iterations and result choice	3 runs, median	3 runs, median	3 runs, median
Server platform	E8s_v4	E16s_v4	E64s_v4
BIOS name and version	Hyper-V UEFI Release v4.1	Hyper-V UEFI Release v4.1	Hyper-V UEFI Release v4.1
Operating system name and version/build number	Ubuntu 20.04 5.8.0-1036-azure	Ubuntu 20.04 5.8.0-1036-azure	Ubuntu 20.04 5.8.0-1036-azure
Date of last OS updates/patches applied	ResNet50: 05/04/2021 Wide & Deep: 07/17/2021	ResNet50: 05/04/2021 Wide & Deep: 07/17/2021	ResNet50: 05/04/2021 Wide & Deep: 07/17/2021

System configuration information	ESv4 VM - 8 vCPU	ESv4 VM - 16 vCPU	ESv4 VM - 64 vCPU
Processor			
Number of processors	1	1	2
Vendor and model	Intel® Xeon® Platinum 8272CL	Intel Xeon Platinum 8272CL	Intel Xeon Platinum 8272CL
Core count (per processor)	26	26	26
Core frequency (GHz)	2.60	2.60	2.60
Stepping	7	7	7
Hyper-threading	Yes	Yes	Yes
Turbo	Yes	Yes	Yes
Number of vCPUs per VM	8	16	64
Memory module(s)			
Total memory in system (GB)	64	128	512
NVMe memory present?	No	No	No
Total memory in GB (DDR+NVMe RAM)	64	128	512
General hardware			
Storage: NW or Direct Att / Instance	Direct Att	Direct Att	Direct Att
Local storage			
OS			
Number of drives	1	1	1
Drive size (GB)	ResNet50: 100 Wide & Deep: 30	ResNet50: 100 Wide & Deep: 30	ResNet50: 100 Wide & Deep: 30
Drive information	Standard SSD	Standard SSD	Standard SSD
Data drive			
Number of drives	1	1	1
Drive size (GB)	ResNet50: 200 Wide & Deep: 10	ResNet50: 200 Wide & Deep: 10	ResNet50: 200 Wide & Deep: 10
Drive information	Standard SSD	Standard SSD	Standard SSD
Network adapter			
Vendor and model	Vendor and model Hyper-V Virtual Adapter	Vendor and model Hyper-V Virtual Adapter	Vendor and model Hyper-V Virtual Adapter
Number and type of ports	1x 50Gb	1x 50Gb	1x 50Gb

How we tested

Testing overview

We tested newer Azure Esv4 VMs featuring 2nd Gen Intel Xeon Scalable processors vs. Azure Esv3 VMs with older processors. We ran the ResNet50 and Wide & Deep inferences at INT8 and FP32 precision to show throughput (images per second, samples per second) performance, comparing newer instance series vs. the older.

ResNet50 workload

Creating the Ubuntu Server 20.04 baseline image

Creating the baseline image VM

1. Log into the Azure Portal, and navigate to the Virtual Machines service.
2. To open the Add VM wizard, click Add.
3. On the Basics tab, set the following:
    a. From the drop-down menu, choose your Subscription.
    b. From the drop-down menu, choose your Resource group.
    c. Name the Virtual Machine.
    d. From the drop-down menu, choose your Region.
    e. Leave the Availability options set to No infrastructure redundancy required.
    f. Select the instance size you wish to use.
    g. Choose a new Username and Password for the Administrator account.
    h. Leave Public inbound ports set to Allow selected ports.
    i. For Select inbound ports, choose SSH (22).
4. On the Disks tab, set the following:
    a. From the drop-down menu, choose Standard HDD for the OS disk type.
    b. Leave the default Encryption type.
5. On the Networking tab, leave all defaults.
6. On the Advanced tab, leave all defaults.
7. On the Tags tab, add any tags you wish to use.
8. On the Review + create tab, review your settings, and click Create.

Creating a snapshot of your baseline VM

1. In the Azure portal, navigate to the Snapshots service.
2. To open the Snapshot wizard, click Create.
3. On the Basics tab, set the following:
    a. From the drop-down menu, choose your Subscription.
    b. From the drop-down menu, choose your Resource group.
    c. Enter a name for your snapshot.
    d. From the drop-down menu, choose your Region.
    e. For the Snapshot type, select Full - make a complete read-only copy of the selected disk.
    f. Choose the OS disk from your baseline VM.
    g. For the Storage type, choose Standard HDD.
4. On the Encryption tab, leave all defaults.
5. On the Tags tab, add any tags you wish to use.
6. On the Review + create tab, review your settings, and click Create.

Creating your image with the baseline snapshot

To create an image, you must first have a Shared Image Gallery. The steps below will walk you through creating the gallery and image. Once you have created your gallery, you will not need to do so again to add new images.

1. In the Azure portal, navigate to the Shared image galleries service.
2. To open the Add gallery wizard, click Add.
3. On the Basics tab, set the following:
    a. From the drop-down menu, choose your Subscription.
    b. From the drop-down menu, choose your Resource.
    c. Name your gallery.
    d. From the drop-down menu, choose your Region.
    e. Enter a Description.
4. On the Tags tab, add any tags you wish to use.
5. On the Review + create tab, review your settings, and click Create.
6. Click on your new image gallery, and click Add new image definition to open the wizard.
7. On the Tags tab, add any tags you wish to use.
8. On the Review + create tab, review your settings, and click Create.
9. Click on the image definition you've created, and click Add version to open the wizard.
10. On the Basics tab, set the following:
    a. Enter a version number such as 1.0.0.
    b. From the drop-down menu, choose the OS disk snapshot of the baseline VM you created.
    c. Leave the rest as defaults.
11. On the Encryption tab, leave defaults.
12. On the Tags tab, add any tags you wish to use.
13. On the Review + create tab, review your settings, and click Create.

Creating the VMs from the specialized image

1. Open the Azure Portal and navigate to the Share image galleries service.
2. Click on the Shared image gallery you created.
3. Navigate to the image version you created (we used 1.0.0 above), and click Create VM.
4. On the Basics tab, set the following:
    a. From the drop-down menu, choose your Subscription.
    b. From the drop-down menu, choose your Resource group.
    c. Enter a Virtual machine name.
    d. Choose Availability Zone, and set the Zone you desire.
    e. Select the instance size you want.
    f. Leave the rest as defaults.
5. On the Disks tab, set the following:
    a. Change the OS disk type to Standard HDD.
6. Click Create, and attach a new disk.
7. In the Create a new disk wizard, click Change size and pick the size of Premium SSD that matches your instance type.
8. Leave the rest as defaults, and click OK.
9. On the Tags tab, assign any tags you wish to use.
10. On the Review + create tab, review your settings, and click Create.
11. Once the VM creation is finished, click Go to resource (or navigate to the virtual machine service and click on the new VM).

Installing the ResNet50 workload

Configuring the test data

1. Log into the test VM using the SSH command in the Connect tab for the VM.
2. Install the latest updates on the VM:

```
sudo apt update
sudo apt-get upgrade -y
sudo reboot
```

3. Prepare the 200GB disk (replace /dev/sdb with the correct location).

```
sudo mkfs -t ext4 /dev/sdb
sudo mkdir /media/imagenet
sudo mount -t extr /dev/sdb / media/imagenet
sudo mkdir -p /media/imagenet/data
sudo chown ubuntu:ubuntu /media/imagenet/data
```

4. Download the ImageNet training and validation images, and set up a Python3 environment to preprocess and update the data set by following the instructions and using the scripts from the IntelAI ImageNet GitHub repository: https://github.com/IntelAI/models/blob/master/datasets/imagenet/README.md
   Note that you will only need to copy the "train_XXXXX_of_01024" and "validate_XXXXX_of_00128" files into /media/imagenet/data for the purposes of this testing.
5. After the images have downloaded, move the 200GB disk from VM to VM as needed to save having to re-download the images.

Preparing an instance for testing

1. Create a new Ubuntu 20.04 instance of the desired instance size as described above.
2. Attach the workload dataset volume to the instance.
3. Log into the Azure Portal, and click Virtual Machines.
4. Click the virtual machine.
5. Click Disks.
6. Select the 200GB disk, and click Attach.
7. Click Save.
8. Start the instance, and SSH into it.
9. If testing a EXXs_v3 instance, verify that the assigned CPU is Broadwell architecture by examining the output of `cat /proc/cpuinfo`. If not, stop the instance and wait for the Stopped (Deallocated) status. Restart the instance and check for a new CPU.
10. Mount the volume using the steps in the previous section.
11. Install prerequisite packages:

```
sudo apt-get update -y
sudo apt-get install -y build-essential cmake git pkg-config python3 python3-venv python3-pip curl
google-perftools wget rysnc htop numactl nmon
```

12. Install Docker:

```
sudo apt-get remove docker docker-egine docker.io containerd runc
sudo apt-get update
sudo apt-get install apt-transport-https ca-certifications 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
```

13. Clone the Intel optimized model repository:

```
cd ~
mkdir project
cd project
git clone https://github.com/IntelAI/models.git
```

14. Download the appropriate pretrained model for the precision under test:
    For INT8:

```
cd ~/project
wget https://storage.googleapis.com/intel-optimized-tensorflow/models/v1_8/resnet50v1_5_int8_
pretrained_model.pb
```

    For FP32:

```
cd ~/project
wget https://zenodo.org/record/2535873/files/resnet50_v1.pb
```

15. Create dockerfile:
```
cd ~
mkdir image
cd image
nano Dockerfile
```
16. Paste the following into the dockerfile based on the precision under test:
For INT8:
```
FROM intel/intel-optimized-tensorflow:2.4.0
RUN apt-get update -y
RUN apt-get install -y wget nano google-perftools
RUN ln -s /usr/lib/x86_64-linux-gnu/libtcmalloc.so.4.5.3 /usr/lib/libtcmalloc.so
```

For FP32:
```
FROM intel/intel-optimized-tensorflow:2.3.0
RUN apt-get update -y
RUN apt-get install -y wget nano google-perftools
RUN ln -s /usr/lib/x86_64-linux-gnu/libtcmalloc.so.4.5.3 /usr/lib/libtcmalloc.so
```
17. Build the docker image:
```
cd ~
sudo docker build ./image/ -t PT-resnet50v1.5
```
18. Power off the VM using the OS shutdown command.

Running the ResNet50 workload

1. Start the instance and SSH into it. Verify that the CPU is Broadwell. If not, use the steps in the previous section to reallocate the VM.
2. Mount the 200GB volume using the steps in the previous section.
3. Wait five minutes to ensure the instance startup has fully completed.
4. Navigate to the test run directory:
```
cd project/models/benchmarks/
```
5. Start nmon to collect test instance performance counters. Replace `XX` with the number of seconds to collect counters:
```
nmon -F output.nmon -s 1 -c XX -J -t
```
6. Launch the benchmark. Replace the `batch-size`, `steps`, and `warmup-steps` parameter values as indicated by the test plan:
For INT8:
```
sudo python3 ./launch_benchmark.py --data-location /media/imagenet/data --in-graph /home/ubuntu/
project/resnet50v1_5_int8_pretrained_model.pb --model-name resnet50v1_5 --framework tensorflow
--precision int8 --mode inference --batch-size=XXX --warmup-steps=XXX --steps=XXX --docker-image PT-
resnet50v1.5:latest
```

For FP32:
```
sudo python3 ./launch_benchmark.py --data-location /media/imagenet/data --in-graph /home/ubuntu/
project/resnet50_v1.pb --model-name resnet50v1_5 --framework tensorflow --precision fp32 --mode
inference --batch-size=XXX --warmup-steps=XXX --steps=XXX --docker-image PT-resnet50v1.5:latest
```
7. When the run completes, record the resulting inference images/sec throughput reported by the script.
8. Reboot the VM:
```
sudo reboot
```
9. Repeat steps 2 through 8 until three successful consecutive test runs for your desired inference type are completed.

Wide & Deep workload

Spawning the Ubuntu Server 20.04 LTS virtual machine

1. Log into the Azure console, and click Virtual Machines.
2. Click Add, and click Virtual Machine.
3. For the Image, select Ubuntu Server 20.04 LTS – Gen1.
4. Select your desired VM size (E8s_v4/E16s_v4/E64s_v4/E8s_v3/E16s_v3/E64s_v3).
5. Set the username to `ubuntu`
6. Select your desired SSH key. We used an existing key stored in Azure.
7. Click the Select inbound ports drop-down menu, and enable all ports.
8. Click Next: Disks.
9. For OS Disk Type, select Standard SSD (locally redundant storage)
10. Leave other options as defaults.
11. Click Create and attach a new disk.
12. Click Change size.

13. For Disk SKU, select Standard SSD (locally redundant storage).
14. Set Custom disk size (GiB) to 10.
15. Click OK.
16. Click OK.
17. Click Review + create.
18. Click Create.
19. For v3 instances, you may need to repeat this process until the spawned instance has the desired processor SKUs.

Installing system packages

1. Connect to the virtual machine via SSH.
2. Become root:
```
sudo su
```
3. Update the APT package manager and install useful system packages:
```
apt-get update -y
apt-get upgrade -y
apt-get install -y curl wget htop tree jq nmon python3-pip python3-venv
```

Installing the Wide & Deep utility script

1. Contact Principled Technologies at info@principledtechnologies.com to request a copy of the utility script: wide_and_deep.sh.
2. Connect to the virtual machine via SSH
3. Become root:
```
sudo su
```
4. Create a directory to hold the script:
```
mkdir -p /opt/wide_and_deep
```
5. Edit the utility script.
```
nano /opt/wide_and_deep/wide_and_deep.sh
```
6. Copy and paste in the content from the script provided by Principled Technologies.
7. Save the file by pressing CTRL+O, and press CTRL+X to exit.
8. Set the file's and folder permissions.
```
chmod 755 /opt/wide_and_deep
chmod 755 /opt/wide_and_deep/wide_and_deep.sh
```

Formatting the persistent volume

Determining the device identifier of the persistent and boot volumes

1. Connect to the virtual machine via SSH.
2. Become root:
```
sudo su
```
3. List the partitions:
```
cat /proc/partitions
```
4. Take note of the 30 GiB device and primary partition. This is the boot device and boot partition.
5. Take note of the 10 GiB Device. This is the persistent device.
6. For the persistent device, the primary partition will not exist yet, but its name will depend on the device name. For /dev/sdXXX or /dev/xvdXXX devices, the corresponding partition name will be /dev/sdXXX1 or /dev/xvdXXX1, respectively. For device names /dev/nvmeXXXn1, the corresponding partition will be named /dev/nvmeXXXn1p1.

Formatting and mounting the persistent volume

1. Connect to the virtual machine via SSH.
2. Become root:
```
sudo su
```
3. Invoke the utility script to format the persistent volume:
Note: Be sure to replace PERSISTENT_DEVICE_ID and PERSISTENT_PARTITION_ID with the values determined for the persistent volume.

```
/opt/wide_and_deep/wide_and_deep.sh \
  --noninteractive \
  -d 'PERSISTENT_DEVICE_ID' \
  -P 'PERSISTENT_PARTITION_ID' \
  -m '/persistent' \
  -D benchmark \
  init_persistent
```

Preparing virtual machines for Wide & Deep testing

Installing Intel-optimized model repository and associated prerequisites

1. Connect to the virtual machine via SSH.
2. Become root:
   ```
   sudo su
   ```
3. Invoke the utility script to format the persistent volume:
   ```
   /opt/wide_and_deep/wide_and_deep.sh install
   ```

Downloading and installing pre-trained FP32 and INT8 Wide & Deep models

1. Connect to the virtual machine via SSH.
2. Become root:
   ```
   sudo su
   ```
3. Invoke the utility script to download the pre-trained models:
   Note: Be sure to replace **PERSISTENT_DEVICE_ID** and **PERSISTENT_PARTITION_ID** with the values determined for the persistent volume.

   ```
   /opt/wide_and_deep/wide_and_deep.sh \
     --noninteractive \
     -d 'PERSISTENT_DEVICE_ID' \
     -P 'PERSISTENT_PARTITION_ID' \
     -m '/persistent' \
     -D benchmark \
     get_models
   ```

Downloading and installing Wide & Deep datasets

1. Connect to the virtual machine via SSH.
2. Become root:
   ```
   sudo su
   ```
3. Invoke the utility script to download and preprocess the datasets:
   Note: be sure to replace **PERSISTENT_DEVICE_ID** and **PERSISTENT_PARTITION_ID** with the values determined for the persistent volume.

   ```
   /opt/wide_and_deep/wide_and_deep.sh \
   --noninteractive \
   -d 'PERSISTENT_DEVICE_ID' \
   -P 'PERSISTENT_PARTITION_ID' \
   -m '/persistent' \
   -D benchmark \
   get_datasets
   ```

Running a Wide & Deep test on the virtual machine

1. Connect to the virtual machine via SSH.
2. Become root:
   ```
   sudo su
   ```
3. Start nmon to capture system statistics:
   ```
   cd /opt/wide_and_deep
   NMON_PID='nmon -F output.nmon -s 1 -c 7200 -J -t -p'
   ```
4. Invoke the utility script to download and preprocess the datasets:

   Note: Be sure to replace `PERSISTENT_DEVICE_ID` and `PERSISTENT_PARTITION_ID` with the values determined for the persistent volume. Replace `PRECISION` with the desired precision (either int8 or fp32). Replace `BATCH_SIZE` with the desired batch size (we used 512 for all tests). Replace `NUM_INTRA_THREADS` and `NUM_OMP_THREADS` with the desired number of threads of each type. For our testing, we used 4 for both counts. Replace `INSTANCE_COUNT` with the desired number of concurrent instances. For our testing, we used the number of vcpus / 4.

   ```
   /opt/wide_and_deep/wide_and_deep.sh \
     --noninteractive \
     -I INSTANCE_COUNT \
     -d 'PERSISTENT_DEVICE_ID' \
     -P 'PERSISTENT_PARTITION_ID' \
     -m '/persistent' \
     -D benchmark \
     -p PRECISION \
     -b BATCH_SIZE \
     -k 'tight' \
     -i NUM_INTRA_THREADS \
     -o NUM_OMP_THREADS \
     run | tee test.log
   ```
5. Stop nmon:
   ```
   kill -s USR2 ${NMON_PID}
   ```
6. Copy the test.log and NMON.output file in the /opt/wide_and_deep directory to your local machine.
7. Repeat steps 3 through 6 for three or more iterations and report the run with the median throughput (inferenced samples per second).

Read the report at http://facts.pt/YX3rsPQ ▶

This project was commissioned by Intel.