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

Classify images up to 6.17x more images per second

Generate ad recommendations up to 2.94x as fast

Speed up deep learning tasks with Amazon Web Services instances featuring 2nd Gen Intel Xeon Scalable processors

Newer M5n instances featuring Intel Xeon Platinum 8272CL processors performed more inference operations per second than M4 instances with older processors

Big data—the massive amounts of information that organizations collect—is useful only if the information is sorted and classified to deliver insights they can act on. By using deep learning networks for quick image classification and prediction, computers can offer insight into business patterns and offer suggestions to consumers in real time. Amazon Web Services (AWS) Elastic Compute Cloud (EC2) offers several cloud instances that can support deep learning models, including general-purpose M5n instances. These newer AWS M5n instances run on Intel® Xeon® Platinum 8272CL processors, which include a feature, Intel Deep Learning Boost, that Intel designed to improve machine learning workloads.

At Principled Technologies, we used two deep learning inference benchmarks from the Model Zoo for Intel Architecture—ResNet50, which classifies images, and Wide & Deep recommendation system, which generates advertisement recommendations—to compare the inference performance of older M4 instances to newer M5n instances at various instance sizes. We found that for both deep learning frameworks, the upgraded M5n instances offered significantly better inference performance, which shows that M5n instances featuring 2nd Generation Intel Xeon Scalable processors can help organizations make sense of their data faster.

Speed up deep learning tasks with Amazon Web Services instances featuring 2nd Gen Intel Xeon Scalable processors September 2021

The science behind the report:

Speed up deep learning tasks with Amazon Web Services instances featuring 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 Speed up deep learning tasks with Amazon Web Services instances featuring 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.

AWS instance	Average images per second processed
8 vCPUs	
m4.2xlarge	1
m5n.2xlarge	6.17
16 vCPUs	
m4.4xlarge	1
m5n.4xlarge	5.78
64 vCPUs	
m4.16xlarge	1
m5n.16xlarge	5.23

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

AWS instance	Average samples per second processed
8 vCPUs	
m4.2xlarge	1
m5n.2xlarge	2.86
16 vCPUs	
m4.4xlarge	1
m5n.4xlarge	2.94
64 vCPUs	
m4.16xlarge	1
m5n.16xlarge	2.67

System configuration information

Table 3: Detailed configuration information for the M4 instances under test.

System configuration information	M4 instance - 8 vCPU	M4 instance - 16 vCPU	M4 instance - 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	AWS EC2 – US East 1	AWS EC2 – US East 1	AWS EC2 – US East 1
Workload and version	ResNet50: ResNet50v1 Wide and Deep: Intel optimized models v2.3.0	ResNet50: ResNet50v1 Wide and Deep: Intel optimized models v2.3.0	ResNet50: ResNet50v1 Wide and Deep: Intel optimized models v2.3.0
WL specific parameters	ResNet50: 50 steps, 10 warmup steps, 1/128 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, 1/128 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, 1/128 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	m4.2xlarge	m4.4xlarge	m4.16xlarge
BIOS name and version	Xen 4.2.amazon 08/24/2006	Xen 4.2.amazon 08/24/2006	Xen 4.2.amazon 08/24/2006
Operating system name and version/build number	Ubuntu 20.04 5.4.0-1045-aws	Ubuntu 20.04 5.4.0-1045-aws	Ubuntu 20.04 5.4.0-1045-aws
Date of last OS updates/patches applied	ResNet50: 05/04/2021 Wide and Deep: 07/17/2021	ResNet50: 05/04/2021 Wide and Deep: 07/17/2021	ResNet50: 05/04/2021 Wide and Deep: 07/21/2021
Processor			
Number of processors	1	1	2
Vendor and model	Intel® Xeon® E5-2686 v4	Intel Xeon E5-2686 v4	Intel Xeon E5-2686 v4
Core count (per processor)	18	18	18
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)	32	64	256
NVMe™ memory present?	No	No	No
Total memory in GB (DDR+NVMe RAM)	32	64	256

System configuration information	M4 instance - 8 vCPU	M4 instance - 16 vCPU	M4 instance - 64 vCPU
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 (GP2)	Standard SSD (GP2)	Standard SSD (GP2)
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 (GP2)	Standard SSD (GP2)	Standard SSD (GP2)
Network adapter			
Vendor and model	Amazon Elastic Network Adapter	Amazon Elastic Network Adapter	Amazon Elastic Network Adapter
Number and type of ports	1x 10Gb	1x 10Gb	1x 25Gb

Table 4: Detailed configuration information for the M5n instances under test.

System configuration information	M5n instance - 8 vCPU	M5n instance - 16 vCPU	M5n instance - 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	AWS EC2 – US East 1	AWS EC2 – US East 1	AWS EC2 – US East 1
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, 1/128 batch size Wide and 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, 1/128 batch size Wide and 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, 1/ batch size Wide and 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	m5n.2xlarge	m5n.4xlarge	m5n.16xlarge
BIOS name and version	Amazon EC2 1.0 10/16/2017	Amazon EC2 1.0 10/16/2017	Amazon EC2 1.0 10/16/2017
Operating system name and version/build number	Ubuntu 20.04 5.4.0-1045-aws	Ubuntu 20.04 5.4.0-1045-aws	Ubuntu 20.04 5.4.0-1045-aws
Date of last OS updates/patches applied	ResNet50: 05/04/2021 Wide and Deep: 07/17/2021	ResNet50: 05/04/2021 Wide and Deep: 07/17/2021	ResNet50: 05/04/2021 Wide and Deep: 07/17/2021

System configuration information	M5n instance - 8 vCPU	M5n instance - 16 vCPU	M5n instance - 64 vCPU
Processor			
Number of processors	1	1	1
Vendor and model	Intel® Xeon® 8259CL	Intel Xeon 8259CL	Intel Xeon 8259CL
Core count (per processor)	24	24	24
Core frequency (GHz)	2.50	2.50	2.50
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)	32	64	256
NVMe memory present?	No	No	No
Total memory in GB (DDR+NVMe RAM)	32	64	256
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 (GP2	Standard SSD (GP2	Standard SSD (GP2
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 (GP2)	Standard SSD (GP2)	Standard SSD (GP2)
Network adapter			
Vendor and model	Amazon Elastic Network Adapter	Amazon Elastic Network Adapter	Amazon Elastic Network Adapter
Number and type of ports	1x 25Gb	1x 25Gb	1x 25Gb

How we tested

Testing overview

We compared the performance of AWS M5n instances with 2nd Gen Intel Xeon Scalable processors to M4 instances with Intel Xeon E5-2686 v4 processors using two inferencing workloads at varying instance sizes: 8, 16, and 64 vCPUs. We collected the throughput in images/sec using both INT8 (on M5n) and FP32 (on both instance types) for the TensorFlow ResNet50 workload and the samples per second for the Wide and Deep workload.

ResNet50 workload

Creating the Ubuntu Server 20.04 LTS baseline image

1. Log into AWS, and navigate to the AWS Management Console.
2. Click EC2.
3. Click Launch instance, and from the drop-down menu, click Launch instance. This opens the Launch Instance wizard.
4. In the search window, enter `Ubuntu` and press Enter.
5. On the Quick Start tab, click Select.
6. On the Choose Instance Type tab, select the desired instance type, and click Next: Configure Instance Details.
7. On the Configure Instance tab, set the following:
    a. Number of instances: 1
    b. Purchasing option: Leave unchecked
    c. Network: Default VPC.
    d. Subnet: Choose the availability zone you are working in.
    e. Auto-assign Public IP: Enable.
    f. Placement Group: Leave unchecked.
    g. Capacity Reservation: Open
    h. Domain join directory: No Directory
    i. IAM role: None
    j. Shutdown behavior: Stop
8. Click Next: Add Storage.
9. On the Add Storage tab, set the following:
    a. Size: 100 GiB
    b. Volume Type: General Purpose SSD (gp2)
    c. Delete on Termination: Checked
    d. Encryption: Not Encrypted
10. Click Next: Add Tags.
11. On the Add Tags tab, add any tags you wish to use. Click Next: Configure Security Group.
12. On the Configure Security Group tab, leave defaults, and click Review and Launch.
13. On the Review Tab, click Launch.
14. Choose the appropriate option for the key pair, and click Launch Instances.

Creating an AMI of your baseline instance

1. Log into AWS, and navigate to the AWS Management Console.
2. Click EC2.
3. Click Running instances.
4. Place a checkmark next to the instance you wish to create an image from.
5. Click the Action drop-down, and select Image→Create Image.
6. Enter the Image name, and click Create Image.
7. In the menu on the left side of the page, navigate to Images→AMIs to see your new image.

Creating an instance from the baseline AMI

1. Log into AWS, and navigate to the AWS Management Console.
2. Click Images→AMIs.
3. Check the box next to the image you created in the previous step, and click Launch.
4. On the Choose Instance Type tab, select your desired instance type, and click Next: Configure Instance Details.
5. On the Configure Instance tab, set the following:
    a. Number of instances: 1
    b. Purchasing option: Leave unchecked
    c. Network: Default VPC
    d. Subnet: Choose the availability zone you are working in
    e. Auto-assign Public IP: Enable
    f. Placement Group: Leave unchecked
    g. Capacity Reservation: Open
    h. Domain join directory: No Directory
    i. IAM role: None
    j. Shutdown behavior: Stop
6. Click Next: Add Storage.
7. On the Add Storage tab, set the following:
    a. Size: 100 GiB
    b. Volume Type: General Purpose SSD (gp2)
    c. Delete on Termination: Checked
    d. Encryption: Not Encrypted
8. Click Next: Add Tags.
9. On the Add Tags tab, add any tags you wish to use. Click Next: Configure Security Group.
10. On the Configure Security Group tab, leave defaults, and click Review and Launch.
11. On the Review Tab, click Launch.
12. Choose the appropriate option for the key pair, and click Launch Instance.

Building the imagenet-data test data volume

1. Log into AWS, and navigate to the AWS Management Console.
2. Click EC2.
3. Click Volumes.
4. Click Create Volume, and designate the following settings:
    a. Volume Type: General Purpose SSD (gp2)
    b. Size (GiB): 200
    c. Availability Zone: whatever Availability Zone your instances reside in
5. Click Create Volume, and click Done.
6. Rename the volume `imagenet-data`
7. Create an m5n.2xlarge instance using the steps described previously.
8. Attach the workload dataset volume to the instance:
9. Log into the AWS management console, and click EC2.
10. Under Elastic Block Store, click Volumes.
11. Right-click the imagenet-data volume, and click Attach.
12. Specify the instance you just created in the Instance field, and click Attach.
13. Start the instance, and log into it via SSH.
14. Create the filesystem (make sure to replace nvme1n1 with the device id of the imagenet-data volume as it appears in your instance's `/proc/partitions` file):
```
sudo mkfs -t ext4 /dev/nvme1n1
```
15. Mount the filesystem:
```
sudo mkdir /media/imagenet
sudo mount -t ext4 /dev/nvme1n1 /media/imagenet
sudo mkdir -p /media/imagenet/data
sudo chown ubuntu:ubuntu /media/imagenet/data
```

16. 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.
17. When the scripts have completed, stop the instance, and detach the imagenet-data volume so that it can be used on other instances under test:,
18. Log into the AWS management console, and click EC2.
19. Under Elastic Block Store, click Volumes.
20. Right-click the imagenet-data volume, and click Detach.

Preparing an instance for testing and running the ResNet50 workload

1. Create a new instance of the desired instance size from the baseline AMI (see above).
2. Attach the workload dataset volume to the instance.
3. Log into the AWS management console, and click EC2.
4. Under Elastic Block Store, click Volumes.
5. Right-click the imagenet-data volume you created previously, and click Attach.
6. Specify the instance you just created in the Instance field, and click Attach.
7. Start the instance, and SSH into it.
8. If testing an M4 instance, verify that the assigned CPU is Broadwell architecture by examining the output of `cat /proc/cpuinfo`. If not, terminate the instance and restart from step 1 until your instance has a Broadwell CPU.
9. Use the command `cat /proc/partitions` to find the name of the 200GB dataset volume you attached in step 2.
10. Mount the volume (replace `nvme1n1` with your volume name if different):
```
sudo mkdir /media/imagenet
sudo mount -t ext4 /dev/nvme1n1 /media/imagenet/
sudo mkdir -p /media/imagenet/data
sudo chown ubuntu:ubuntu /media/imagenet/data
```
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 models 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:
    a. For INT8:
    ```
    cd ~/project
    wget https://storage.googleapis.com/intel-optimized-tensorflow/models/v1_8/resnet50v1_5_int8_
    pretrained_model.pb
    ```
    b. 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:
    a. 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
    ```

   b. 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. Stop the instance.
19. Start the instance, and SSH into it.
20. If testing an M4 instance, verify that the assigned CPU is Broadwell architecture by examining the output of `cat /proc/cpuinfo`. If not, terminate the instance and restart from step 1 until your instance has a Broadwell CPU.
21. Use the command `cat /proc/partitions` to find the name of the 200GB dataset volume attached in step 2.
22. Mount the volume (replace `nvme1n1` with your volume name if different):
```
sudo mount -t ext4 /dev/nvme1n1 /media/imagenet/
```
23. Wait five minutes to ensure the instance startup has fully completed.
24. Navigate to the test run directory:
```
cd project/models/benchmarks/
```
25. 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
```
26. Launch the benchmark. Replace the `batch-size`, `steps`, and `warmup-steps` parameter values as indicated by the test plan:

   a. 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
```
   b. 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
```

27. When the run completes, record the resulting inference images/sec throughput reported by the script.
28. Reboot the VM:
```
sudo reboot
```
29. Repeat until three successful consecutive test runs for the desired inference type are completed.

Wide & Deep workload

Spawning the Ubuntu Server 20.04 LTS virtual machine

1. Log into AWS, and navigate to the AWS Management Console.
2. Click EC2.
3. To open the Launch Instance wizard, click Launch instance, and in the drop-down, click Launch instance.
4. In the search window, enter `Ubuntu` and press enter.
5. On the Quick Start tab, next to Ubuntu Server 20.04 LTS (HVM), click Select, and click SSD Volume Type.
6. On the Choose Instance Type tab, select your desired instance type, and click Next: Configure Instance Details.
7. On the Configure Instance tab, set the following:

   a. Number of instances: 1
   b. Purchasing option: Leave unchecked
   c. Network: Default VPC
   d. Subnet: Choose the availability zone you are working in.
   e. Auto-assign Public IP: Enable
   f. Placement Group: Leave unchecked
   g. Capacity Reservation: Open
   h. Domain join directory: No Directory
   i. IAM role: None
   j. Shutdown behavior: Stop

8. Click Next: Add Storage.
9. On the Add Storage tab, set the following:
    a. Size: 30 GiB
    b. Volume Type: General Purpose SSD (gp2)
    c. Delete on Termination: Checked
    d. Encryption: Not Encrypted
10. Click Add New Volume:
    a. Set Volume Type to EBS
    b. Set device to /dev/sdb
    c. Leave snapshot blank
    d. Set Size(GiB) to 10
    e. Set Volume Type to General Purpose SSD (gp2)
    f. Ensure Delete on Termination is checked
    g. Set Encryption to Not Encrypted
11. Click Next: Add Tags.
12. On the Add Tags tab, add any tags you wish to use. Click Next: Configure Security Group.
13. On the Configure Security Group tab, leave defaults, and click Review and Launch.
14. On the Review Tab, click Launch.
15. Choose the appropriate option for the key pair, and click Launch Instances.
16. For M4 instances, you may need to repeat this process until the spawned instance has your 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 and 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 then exit by pressing `CTRL+X`.
8. Set the file 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 name will be /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 the virtual machine for Wide & Deep testing

Installing Intel Xeon processor-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 and 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/oOUDy0F ▶