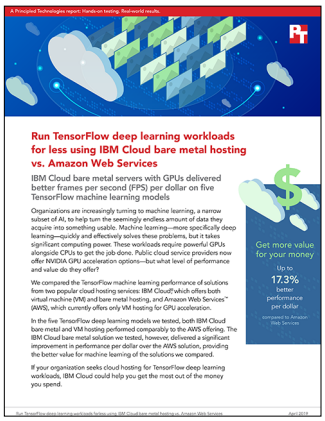


The science behind the report:

# Run TensorFlow deep learning workloads for less using IBM Cloud bare metal hosting vs. Amazon Web Services

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 Run TensorFlow deep learning workloads for less using IBM Cloud bare metal hosting vs. Amazon Web Services.

We concluded our hands-on testing on March 7, 2019. 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 March 7, 2019 or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

## Our results

The table below presents our findings in detail.

| | IBM bare metal | IBM VM | AWS | IBM bare metal | IBM VM | AWS |
|---|---|---|---|---|---|---|
| | Machine learning frames per second | | | Frames per second per dollar | | |
| **resnet50** | 814.3 | 807.0 | 826.2 | 0.4311 | 0.3596 | 0.3672 |
| **inception3** | 484.9 | 483.2 | 502.4 | 0.2567 | 0.2153 | 0.2233 |
| **vgg16** | 498.2 | 499.2 | 519.5 | 0.2637 | 0.2225 | 0.2309 |
| **alexnet** | 8,477.9 | 8,467.5 | 8,948.7 | 4.4881 | 3.7733 | 3.9773 |
| **googlenet** | 1,702.0 | 1,703.1 | 1,793.0 | 0.9010 | 0.7589 | 0.7969 |

# System configuration information

The table below presents detailed information on the systems we tested.

| Server configuration information | IBM bare metal | IBM VM | AWS VM |
|---|---|---|---|
| Tested by | Principled Technologies | Principled Technologies | Principled Technologies |
| Test date | 3/7/2019 | 3/7/2019 | 3/7/2019 |
| Workload and version | Tensorflow CNN_TF_v1.12 | Tensorflow CNN_TF_v1.12 | Tensorflow CNN_TF_v1.12 |
| Server platform | Intel® Xeon® E5-2690 v4/28 Cores/2.60 GHz/Up to 1536 GB/Up to 12 Drives/GPU | AC2.8x60 | p3.2xlarge |
| BIOS name and version | American Megatrends Inc. 3.1 | American Megatrends Inc. 3.1 | Xen 4.2.amazon |
| Operating system name and version/build number | Ubuntu 18.04.2 LTS - 4.15.0-46-generic | Ubuntu 18.04.2 LTS - 4.15.0-46-generic | Ubuntu 18.04.1 LTS - 4.15.0-1033-aws |
| Date of last OS updates/patches applied | 3/7/2019 | 3/7/2019 | 3/7/2019 |
| Processor | | | |
| Number of processors | 2 | 1 | 1 |
| Vendor and model | Intel Xeon E5-2690 v4 | Intel Xeon E5-2690 v4 | Intel Xeon E5-2686 v4 |
| Thread count (per processor) | 56 | 8 | 8 |
| Core frequency (GHz) | 2.6 | 2.6 | 2.3 |
| Stepping | 1 | 1 | 1 |
| Hyper-threading | Yes | No | No |
| Turbo | No | No | Yes |
| Memory module(s) | | | |
| Total memory in system (GB) | 64 | 60 | 61 |
| Number of memory modules | 8 | 4 | 4 |
| Vendor and model | Micron 18ASF1G72PZ-2G1A2 | Not listed | Not listed |
| Size (GB) | 8 | 16 | 16 |
| Type | DDR4 | Not listed | Not listed |
| Speed (MHz) | 2,133 | Not listed | Not listed |
| Speed running in the server (MHz) | 2,133 | Not listed | Not listed |
| NVMe memory present? | No | No | No |
| Total memory (DDR+NVMe RAM) | 64 | 60 | 61 |
| Local storage (OS) | | | |
| Number of drives | 1 | 1 | 1 |
| Drive vendor and model | AVAGO MR9361-8i | Not listed | gp2 |
| Drive size | 1TB | 100GB | 100GB |
| Drive information (speed, interface, type) | HDD | SAN | SSD |

| Server configuration information | IBM bare metal | IBM VM | AWS VM |
|---|---|---|---|
| Network adapter | | | |
| Vendor and model | SuperMicro AOC-2UR6-i4XT | IBM Virtual Adapter | Elastic Network Adapter (ENA) |
| Number and type of ports | 1x 100Mb | 1x 100Mb | 1x 100Mb |

# How we tested

## Provisioning the IBM bare metal cloud server

1. Navigate to https://console.bluemix.net, and log in using your credentials.
2. Navigate to https://www.ibm.com/cloud/bare-metal-servers, and click Customize your server.
3. Select the following options:
    a. Bare metal server
        i. Quantity: 1
        ii. Billing: Monthly
        iii. Hostname: baremetal01
        iv. Location: Dal10
        v. All Servers / GPU: Intel Xeon E5-2690 v4/28 Cores/2.60 GHz/Up to 1536 GB/Up to 12 Drives/GPU
    b. Configuration
        i. GPU or PCIe Card: NVIDIA Tesla V100 Graphics Card
        ii. Quantity: 1
        iii. RAM: 64GB
        iv. SSH Keys: None
        v. Image: Ubuntu 18.04 Minimal LTS (64-bit) – HVM
    c. Storage disks
        i. Type: Individual
        ii. Disks: 1
        iii. Hot Spares: 0
        iv. Disk Media: SATA
        v. Disk Size: 1TB
    d. Network interface
        i. Uplink Port Speeds: 100Mbps Public & Private Network Uplinks
        ii. Private VLAN: Auto Assigned
4. Agree to the terms, and click Create.

## Provisioning the IBM Cloud VM

1. Navigate to https://console.bluemix.net, and log in using your credentials.
2. Select Catalog.
3. Select Virtual Server.
4. Select Public Virtual Server, and click Continue.
5. Select the following options:
    a. Public Instance
        i. Quantity: 1
        ii. Billing: Monthly
        iii. Hostname : virtualserver01
        iv. Placement Group: None
        v. Location: Dal10
        vi. All Profiles / GPU: AC2.8x60
        vii. SSH Keys: None
        viii. Image: Ubuntu 18.04 Minimal LTS (64-bit) – HVM
    b. Attached Storage Disks
        i. Disk: Boot Disk
        ii. Type: SAN
        iii. Size: 100 GB (SAN)
    c. Network Interface
        i. Uplink Port Speeds: 100Mbps Public & Private Network Uplinks
        ii. Private Security Group: Allow All
        iii. Public Security Group: Allow All
6. Agree to the terms, and click Create.

## Provisioning the Amazon AWS EC2 VM

1. Navigate to the AWS management console at https://console.aws.amazon.com/, and log in using your credentials.
2. Select EC2.
3. Under Create Instance, select Launch Instance.
4. Select Ubuntu Server 18.04 LTS (HVM), and click Next.
5. Under GPU Instances, select p3.2xlarge.
6. Click Next: Configure Instance Details.
7. Enable auto-assigned public IP.
8. Click Next: Add Storage, and select the following:
    a. Volume Type: Root
    b. Device: /dev/sda1
    c. Snapshot: default
    d. Size (GiB): 100
    e. Volume Type: General Purpose SSD (gp2)
9. Click Next: Add Tags.
10. Click Next: Configure Security Group, and select the following:
    a. Type: All Traffic
    b. Protocol: All
    c. Port Range: 0 - 65535
    d. Source: Anywhere
11. Click Review and Launch.
12. Click Launch.

## Updating and configuring the operating system

After launching the VMs and bare metal systems, we performed the following steps on the three instances.

1. Log into the operating system as the root user.
2. Update the operating system, and reboot:

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

3. Install the NVIDIA driver:

```
sudo apt-get install -y apt-transport-https curl
cat <<EOF | sudo tee /etc/apt/sources.list.d/cuda.list > /dev/null
deb https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64 /
EOF
curl -s \
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/7fa2af80.pub \
 | sudo apt-key add -

cat <<EOF | sudo tee /etc/apt/preferences.d/cuda > /dev/null
Package: *
Pin: origin developer.download.nvidia.com
Pin-Priority: 600
EOF
sudo apt-get update && sudo apt-get install -y --no-install-recommends cuda-drivers
sudo reboot
```

4. Verify that the NVIDIA drivers installed correctly:

```
nvidia-smi
```

## Installing the NVIDIA Docker container

1. Install Docker:

```
sudo apt-get install -y ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs)
stable"
```

2. Install NVIDIA Docker:

```
curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | \
  sudo apt-key add -
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | \
  sudo tee /etc/apt/sources.list.d/nvidia-docker.list
sudo apt-get update
sudo apt-get install -y nvidia-docker2
sudo usermod -aG docker $USER
sudo reboot
```

3. Test NVIDIA Docker and NVIDIA GPU Cloud (NGC):

```
sudo docker run --runtime=nvidia --rm nvcr.io/nvidia/cuda nvidia-smi
```

## Installing and running the benchmark

We pulled the benchmark tf_cnn_benchmark from the TensorFlow Github repository. We adjusted the run length and sample size to ensure steady-state GPU temperature. To make these changes, run the following commands inside the NVIDIA Docker container on each server:

1. Pull tf_cnn_benchmark:

```
cd /tensorflow
git clone https://github.com/tensorflow/benchmarks.git -b cnn_tf_v1.12_compatible --single-branch
```

2. Initialize TensorFlow benchmarks:

```
nvidia-docker exec ${CONTAINER_NAME} ${PYTHON} -u /tensorflow/benchmarks/scripts/tf_cnn_benchmarks/
tf_cnn_benchmarks.py --model=resnet50 --data_name=imagenet
nvidia-docker exec ${CONTAINER_NAME} ${PYTHON} -u /tensorflow/benchmarks/scripts/tf_cnn_benchmarks/
tf_cnn_benchmarks.py --model=resnet56 --data_name=cifar10
nvidia-docker exec ${CONTAINER_NAME} ${PYTHON} -u /tensorflow/benchmarks/scripts/tf_cnn_benchmarks/
tf_cnn_benchmarks.py --model=ssd300 --data_name=coco
```

3. Run the benchmarks:

   **All tests:**

```
nvidia-docker exec ${CONTAINER_NAME} ${AFFINITY_CMD} ${PYTHON} -u /tensorflow/benchmarks/scripts/tf_
cnn_benchmarks/tf_cnn_benchmarks.py ${TS_PARAMS}
```

   **Switching between FP16 and FP32:**

```
FP16:
FP32: --use_fp16=False
```

## Server test parameters

We ran a variety of models, including resnet50, inception3, vgg16, alexnet, and googlenet. For each of these models, we varied batch_size, variable_update, and all_reduce_spec. We ran the varied settings against a fixed num_batches count for each model that varied from model to model. We added the sample syntax of the test execution below. Where we noted with <variable>, the requisite parameter for that configuration needs to be passed.

```
--num_gpus=1 --data_format=NCHW --model=<model> --data_name=imagenet --data_dir= --use_fp16=True
--batch_size=<variable> --num_batches=<variable> --optimizer=sgd --gpu_thread_mode=gpu_private
--distortions=False --variable_update=<variable> --local_parameter_device=gpu --all_reduce_
spec=<variable> --gradient_repacking=0
```

Example:

```
--num_gpus=1 --data_format=NCHW --model=resnet50 --data_name=imagenet --data_dir= --use_fp16=False
--batch_size=64 --num_batches=500 --optimizer=sgd --gpu_thread_mode=gpu_private --distortions=False
--variable_update=replicated --local_parameter_device=gpu --all_reduce_spec=nccl --gradient_
repacking=0
```

**Read the report at http://facts.pt/yoal9hz** ▶

This project was commissioned by IBM.

Principled Technologies®

Facts matter.®