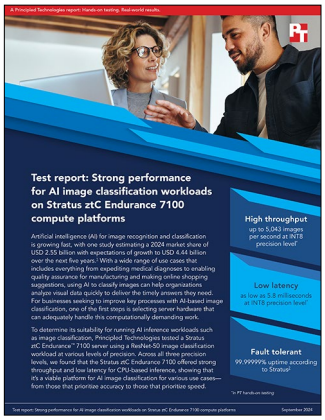

The science behind the report:

Test report: Strong performance for AI image classification workloads on Stratus ztC Endurance 7100 compute platforms

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 Test report : Strong performance for AI image classification workloads on Stratus ztC Endurance 7100 servers.

We concluded our hands-on testing on August 19, 2024. 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 August 16, 2024 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: Results of our throughput testing, in images per second.

Instances	Cores	Batch size	Precision	Run 1	Run 2	Run 3
48	1	116	fp32	699.277	700.169	699.665
48	1	80	bf16	3,100.77	3,099.33	3,097.01
48	1	116	int8	5,043.64	5,029.16	5,049.22

Table 2: Results of our latency testing, in milliseconds.

Instances	Batch size	Precision	Run 1	Run 2	Run 3
1	1	fp32	19.988	20.053	19.597
1	1	bf16	7.059	7.475	7.431
1	1	int8	6.164	5.625	5.804

System configuration information

Table 3: Detailed information on the systems we tested.

System configuration information	Stratus ztC Endurance™ 7100
BIOS name and version	Stratus 3.31.0.0
Non-default BIOS settings	Performance profile
Operating system name and version/build number	VMware® vSphere® 8.0 U2 2380479
Date of last OS updates/patches applied	8/16/2024
Power management policy	Performance
Processor	
Number of processors	2
Vendor and model	Intel® Xeon® Gold 5148Y
Core count (per processor)	24
Base frequency (GHz)	2.00
Stepping	1
Memory module(s)	
Total memory in system (GB)	1,024
Number of memory modules	16
Size (GB)	64
Type	DDR5
Speed (MHz)	4,800
Speed running in the server (MHz)	4,800
Local storage	
Number of drives	2 / 2
Drive vendor and model	Micron® 7450 MTFDKCB1T6TFS / Micron 7450 MTFDKCB3T2TFS
Drive size	1.6 TB / 3.2 TB
Drive information (type)	NVMe® / NVMe
Network adapter #1	
Vendor and model	Intel X550-T2
Number and type of ports	2x 10GbE
Driver version	VMware vSphere 8.0 U2 i40en
Network adapter #2	
Vendor and model	Intel X710-DA2
Number and type of ports	2x 10GbE
Driver version	VMware vSphere 8.0 U2 i40en

System configuration information	Stratus ztC Endurance™ 7100
Network adapter #3	
Vendor and model	Broadcom® Integrated 1GbE
Number and type of ports	2x 1GbE
Driver version	VMware vSphere 8.0 U2 igbn
Power supplies	
Vendor and model	Stratus Zen7100
Number of power supplies	2
Wattage of each (W)	2,400

How we tested

About our testing

Our testing used the following dual-socket solution:

- Stratus ztC Endurance 7100
- 2x Intel Xeon Gold 5148Y processors
- 1TB DDR5 memory

We used VMware vSphere 8.0 U2 as our hypervisor. We created a single VM running Ubuntu 22.04, with 48 vCPU and 900 GB virtual memory.

Installing VMware vSphere 8

1. Boot to the VMware vSphere 8 installation media.
2. To continue, press Enter.
3. To accept the license agreement, press F11.
4. Select the OS installation location.
5. Select a language, and create the root password.
6. To install, press F11.
7. Navigate to the management IP address.
8. Under Configure, set the Power Management policy to High performance.

Creating the base VM

1. Use a web browser to connect, and log into the vSphere instance.
2. Right-click the host, and click New VM.
3. Assign the VM the following properties:
   - 48 virtual CPU (24 cores on 2 sockets)
   - IOMMU enabled
   - 900 GB memory
4. Click Finish.

Installing the OS

1. Boot the VM to the Ubuntu Server 22.04 LTS installation media.
2. When prompted, select Install Ubuntu.
3. Select the desired language, and click Done.
4. Choose a keyboard layout, and click Done.
5. At the Network Connections screen, click Done.
6. At the Configure Proxy screen, click Done.
7. At the Configure Ubuntu Archive Mirror screen, click Done.
8. Select Use an entire disk, and click Done.
9. Click Continue.
10. Enter user account details, and click Done.
11. Enable OpenSSH Server install, and click Done.
12. At the installation summary screen, click Done.
13. When the installation finishes, unmount the installation media, and reboot the VM.

Configuring the OS

1. Boot the VM to the operating system, and log in with the configured user.
2. Update the system:

```
apt update -y
```

3. Upgrade the system:

```
apt upgrade -y
```

4. Install `tuned` and apply the `hpc-compute` profile:

```
apt install -y tuned
tuned-adm profile hpc-compute
```

Installing the benchmark

1. Install the prerequisite software:

```
sudo apt install -y install linux-image-generic-hwe-22.04 numactl google-perftools
```

2. Install Anaconda:

```
sudo apt install libgl1-mesa-glx libegl1-mesa libxrandr2 libxrandr2 libxss1 libxcursor1
libxcomposite1 libasound2 libxi6 libxtst6
curl -O https://repo.anaconda.com/archive/Anaconda3-2024.06-1-Linux-x86_64.sh
bash Anaconda3-2024.06-1-Linux-x86_64.sh
```

3. Create a new conda environment:

```
conda create --name intelpy38 python=3.8
conda activate intelpy38
pip install virtualenv
pip install --upgrade pip virtualenv -p ~/anaconda3/envs/intelpy38/bin/python ~/venv-tf
source ~/venv-tf/bin/activate
```

4. Install Intel Extension for TensorFlow:

```
pip install --upgrade intel-extension-for-tensorflow[cpu]
python -c "import intel_extension_for_tensorflow as itex; print(itex.__version__)"
```

5. Clone the IntelAI repository:

```
mkdir -p ~/github/intelai
cd ~/github/intelai
git clone https://github.com/IntelAI/models.git
```

6. Download and process the ImageNet ILSVRC2012 dataset using the instructions here: https://github.com/intel/ai-reference-models/tree/main/datasets/imagenet#imagenet-dataset-scripts.

Running the benchmark

1. Log into the VM.
2. Activate the conda environment:

```
conda activate intelpy38
source ~/venv-tf/bin/activate
```

3. Navigate to the run directory:

```
cd ~/github/intelai/models
```

4. Run the test script (found in the Scripts section below):

```
./start_all.sh
```

Scripts

Below are the scripts we used in testing:

start_all.sh

```
#!/usr/bin/bash
echo "Starting testing"
./run_resnet50.sh >out.txt 2>&1 & disown
# EOF
```

run_resnet50.sh

```
#!/usr/bin/bash
source ${HOME}/anaconda3/bin/activate intelpy38
source ${HOME}/venv-tf/bin/activate
DATE=$(date +"%Y%m%d%H%M")
export DATASET_DIR=${HOME}/imagenet/tf_records/
export CORES_PER_INSTANCE=1
export BATCH_SIZE="" # Default  ""
export OUTPUT_DIR=${HOME}/logs/${DATE}
export ITERATIONS=3
export TEST_HOME=./quickstart/image_recognition/tensorflow/resnet50v1_5/inference/cpu
mkdir -p ${OUTPUT_DIR}
echo "$(date) Starting resnet50 throughput testing"
export PROGRAM=inference_throughput_multi_instance.sh
./run_resnet50_int8.sh >${OUTPUT_DIR}/throughput.int8.txt
./run_resnet50_bfloat16.sh >${OUTPUT_DIR}/throughput.bfloat16.txt
./run_resnet50_fp32.sh >${OUTPUT_DIR}/throughput.fp32.txt
echo "$(date) Starting resnet50 latency testing"
export LATENCY_ITERATIONS=3
./run_resnet50_latency.sh >${OUTPUT_DIR}/latency.txt 2>&1
echo "$(date) Ended resnet50 testing"
# EOF
```

run_resnet50_int8.sh

```
#!/usr/bin/bash
export PRECISION=int8
export PRETRAINED_MODEL=~/models/resnet50v1_5_int8_pretrained_model.pb
for i in $(seq ${ITERATIONS})
do
    ${TEST_HOME}/${PROGRAM}
done
# EOF
```

run_resnet50_bfloat16.sh

```
#!/usr/bin/bash
export PRECISION=bfloat16
export PRETRAINED_MODEL=~/models/resnet50_v1_5_bfloat16.pb
for i in $(seq ${ITERATIONS})
do
    ${TEST_HOME}/${PROGRAM}
done
# EOF
```

run_resnet50_fp32.sh

```
#!/usr/bin/bash
export PRECISION=fp32
export PRETRAINED_MODEL=~/models/resnet50_v1.pb
for i in $(seq ${ITERATIONS})
do
    ${TEST_HOME}/${PROGRAM}
done
# EOF
```

run_resnet50_latency.sh

```
#!/usr/bin/bash
export PROGRAM=${HOME}/github/intelai/models/models/image_recognition/tensorflow/resnet50v1_5/inference/
cpu/eval_image_classifier_inference.py
export PYTHON=${HOME}/venv-tf/bin/python
echo "### INT8 PRECISION ###"
export PRECISION=int8
export PRETRAINED_MODEL=${HOME}/models/resnet50v1_5_int8_pretrained_model.pb
for i in $(seq ${LATENCY_ITERATIONS})
do
        LD_PRELOAD=/usr/lib/x86_64-linux-gnu/libtcmalloc.so.4 \
                OMP_NUM_THREADS=4 numactl --localalloc --physcpubind=0,1,2,3 \
                ${PYTHON} ${PROGRAM} \
                --input-graph=${PRETRAINED_MODEL} \
                --num-inter-threads=1 \
                --num-intra-threads=4 \
                --batch-size=1 \
                --warmup-steps=50 \
                --steps=1500 \
                --data-num-inter-threads=1 \
                --data-num-intra-threads=4 \
                --data-location=${DATASET_DIR}
done
```

```
echo "### FP32 PRECISION ###"
export PRECISION=fp32
export PRETRAINED_MODEL=~/models/resnet50_v1.pb
for i in $(seq ${LATENCY_ITERATIONS})
do
        OMP_NUM_THREADS=4 numactl --localalloc --physcpubind=0,1,2,3 \
                ${PYTHON} ${PROGRAM} \
                --input-graph=${PRETRAINED_MODEL} \
                --num-inter-threads=1 \
                --num-intra-threads=4 \
                --batch-size=1 \
                --warmup-steps=50 \
                --steps=1500 \
                --data-num-inter-threads=1 \
                --data-num-intra-threads=4 \
                --data-location=${DATASET_DIR}
done
echo "### BFLOAT16 PRECISION ###"
export PRECISION=bfloat16
export PRETRAINED_MODEL=~/models/resnet50_v1_5_bfloat16.pb

for i in $(seq ${LATENCY_ITERATIONS})
do
        OMP_NUM_THREADS=4 numactl --localalloc --physcpubind=0,1,2,3 \
                ${PYTHON} ${PROGRAM} \
                --input-graph=${PRETRAINED_MODEL} \
                --num-inter-threads=1 \
                --num-intra-threads=4 \
                --batch-size=1 \
                --warmup-steps=50 \
                --steps=1500 \
                --data-num-inter-threads=1 \
                --data-num-intra-threads=4 \
                --data-location=${DATASET_DIR}
done
# EOF
```

Read the report at https://facts.pt/KIrWE8h ▶

This project was commissioned by Penguin Solutions.

Principled Technologies®

Facts matter.®