The science behind the report:

Get a clearer picture of potential cloud performance by looking beyond SPECrate 2017 Integer scores

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 **Get a clearer picture of potential cloud performance by looking beyond SPECrate 2017 Integer scores**.

Contents

SPECrate 2017 Integer (SPECint) testing

We concluded our hands-on testing on February 15, 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 February 15, 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 1: Results of our testing.

	D16ds_v5 with Intel® Xeon® Scalable processors	D16ads_v5 with AMD EPYC™ processors	Advantage
SPECint			
SPECrate 2017 Integer base rate median score (higher is better)	71.9	66.0	D16ds_v5 with Intel Xeon Scalable processors 8.3% higher

System configuration information

Table 2: Detailed information on the VMs we tested.

System configuration information	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors
Tested by	Principled Technologies	Principled Technologies
Test date	2/15/2022	2/11/2022
CSP / Region	Microsoft Azure / East US	Microsoft Azure – East US
Workload & version	SPECrate 2017 Integer base rate	SPECrate 2017 Integer base rate
Workload-specific parameters	None	None
Iterations and result choice	3 runs, median	3 runs, median
Server platform	Standard_D16ds_v5	Standard_D16ads_v5
BIOS name and version	American Megatrends Inc. 090008 (12/07/2018)	American Megatrends Inc. 090008 (12/07/2018)
Operating system name and version/ build number	Red Hat® Enterprise Linux® (RHEL) 8.5 4.18.0-348.12.2.el8_5.x86_64	RHEL 8.5 4.18.0-348.12.2.el8_5.x86_64
Date of last OS updates/patches applied	2/15/2022	2/11/2022
Processor		
Number of processors	1	1
Vendor and model	Intel Xeon Platinum 8370C CPU @ 2.80GHz	AMD EPYC 7763 64-Core Processor
Core count (per processor)	32	64
Core frequency (GHz)	2.8	2.45
Family, model, stepping	6, 106, 6	25, 1, 2
SMT	Yes	Yes
Turbo	Yes (3.5 GHz)	Yes (3.5 GHz)
Number of vCPU per VM	16	16

System configuration information	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors
Memory module(s)		
Total memory in system (GB)	64	64
NVMe™ memory present?	No	No
Total memory (DDR+NVMe RAM) (GB)	64	64
General HW		
Storage	Direct-attached / Instance	Direct-attached / Instance
Local storage		
OS		
Number of drives	1	1
Drive size (GB)	30	30
Drive information	StandardSSD_LRS	StandardSSD_LRS
Data drive		
Number of drives	1	1
Drive size (GB)	256	256
Drive information	StandardSSD_LRS	StandardSSD_LRS
Network adapter		
Vendor and model	Azure virtual network adapter version=5.13.0-1017-azure	Azure virtual network adapter version=5.13.0-1017-azure
Number and type of ports	1x 50Gb	1x 50Gb

How we tested

Creating the Red Hat Enterprise Linux 8.5 VM for SPEC CPU 2017

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 image Red Hat Enterprise Linux 8.5 (LVM) - Gen2.
    g. Select the instance size D16ds_v5 or D16ads_v5.
    h. Choose a username and key pair.
    i. Leave Public inbound ports set to Allow selected ports.
    j. 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.
    c. Click Create and attach a new disk.
    d. Click Change Size.
    e. Under Disk SKU, select Standard SSD (locally-redundant storage).
    f. Select 256 GiB, and click OK.
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.
9. Once the VM creation is finished, click Go to resource (or navigate to the virtual machine service, and click the new VM).

Updating Red Hat Enterprise Linux 8.5 and installing SPEC CPU 2017

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 yum update -y
sudo yum install -y libnsl numactl
sudo dnf install -y glibc.i686
```

3. Attach data disk to the VM:

```
echo ';' | sudo sfdisk /dev/sda
sudo mkfs.ext4 /dev/sda1
sudo mkdir /data
sudo mount /dev/sda1 /data
sudo chmod 777 /data
```

4. Copy cpu2017-1.1.8.iso to the VM.
5. Mount the ISO to the mnt folder:

```
sudo mount -t iso9660 -o ro,exec,loop /data/cpu2017-1.1.8.iso /mnt
```

6. Install SPEC CPU 2017 base:

```
cd /mnt
./install.sh -d /data/cpu2017 -u linux-x86_64
```

7. Copy the compiled binaries package to the VM.
8. Extract the package to the install directory:

```
xz -dck /data/Intel-cpu2017-1.1.8-ic2021.4-linux-binaries-20210924.tar.xz | tar -xf -
tar -xf /data/AMD-cpu2017-1.1.5-aocc3-lin-base-binaries-20210317_dt.tar
```

9. Source variables for testing:

```
cd /data/cpu2017
source shrc
```

10. Prepare the Intel Xeon Scalable processor-powered VMs:

```
sudo sh -c "sync; echo 3> /proc/sys/vm/drop_caches"
```

11. Prepare the AMD EPYC-processor-powered VMs:

```
sudo su
yum install -y clang
yum install -y elfutils-libelf-devel
yum install -y llvm
yum install -y cmake
yum install -y libquadmath
echo always > /sys/kernel/mm/transparent_hugepage/enabled
echo 1 > /proc/sys/vm/zone_reclaim_mode
echo 8 > /proc/sys/vm/dirty_ratio
echo 1 > /proc/sys/vm/swappiness
echo 0 > /proc/sys/kernel/randomize_va_space
echo always > /sys/kernel/mm/transparent_hugepage/defrag
sync; echo 3> /proc/sys/vm/drop_caches
exit
```

12. Execute run script:

```
Intel
./run_int_rate_ic2021.4-lin-core-avx512-rate-smt-on-20210924.sh
zip -r INTEL_RATE_INT_02082022.zip result


AMD
./run_intrate_aocc3.0-lin-znver3-rate-smt-on-20210317.sh
zip -r AMD_RATE_INT_02082022.zip result
```

OpenSSL testing

We concluded our hands-on testing on May 9, 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 May 2, 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 3: Results of our testing.

	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors	Advantage
SPECint			
SPECrate 2017 Integer base rate median score (higher is better)	71.9	66.0	D16ds_v5 with Intel Xeon Scalable processors 8.3% higher
OpenSSL			
sha256 median score (higher is better)	14,816,586,410.66	24,323,975,850.65	D16ads_v5 with AMD EPYC processors 64.1% more operations per second (OPS/s)
aes-256-gcm_enc median score (higher is better)	99,226,451,968.01	31,568,838,656.01	D16ds_v5 with Intel Xeon Scalable processors 214.3% more operations per second (OPS/s)
rsa2048_sign median score (higher is better)	62,560.56	13,596.40	D16ds_v5 with Intel Xeon Scalable processors 360.1% more operations per second (OPS/s)
rsa2048_verify median score (higher is better)	956,667.80	456,476.90	D16ds_v5 with Intel Xeon Scalable processors 109.5% more operations per second (OPS/s)
ecdsa256_sign median score (higher is better)	729,020.64	460,138.00	D16ds_v5 with Intel Xeon Scalable processors 58.4% more operations per second (OPS/s)
ecdsa256_verify median score (higher is better)	126,706.40	140,253.90	D16ads_v5 with AMD EPYC processors 10.6% more operations per second (OPS/s)
ecdh256_op median score (higher is better)	454,086.72	183,994.60	D16ds_v5 with Intel Xeon Scalable processors 146.7% more operations per second (OPS/s)

System configuration information

Table 4: Detailed information on the VMs we tested.

System configuration information	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors
Tested by	Principled Technologies	Principled Technologies
Test date	5/12/2022	5/12/2022
CSP / Region	Microsoft Azure / East US	Microsoft Azure / East US
Workload & version	OpenSSL speed 1.1.11	OpenSSL speed 1.1.11
Workload-specific parameters	async_jobs=8, QAT enabled	async_jobs=8, QAT enabled
Iterations and result choice	3 runs, median	3 runs, median
Server platform	Standard_D16ds_v5	Standard_D16ads_v5
BIOS name and version	American Megatrends Inc. 090008 (12/07/2018)	American Megatrends Inc. 090008 (12/07/2018)
Operating system name and version/ build number	Ubuntu 20.04 5.13.0-1017-azure	Ubuntu 20.04 5.13.0-1017-azure
Date of last OS updates/patches applied	5/2/2022	5/2/2022
Processor		
Number of processors	1	1
Vendor and model	Intel Xeon Platinum 8370C CPU @ 2.80GHz	AMD EPYC 7763 64-Core Processor
Core count (per processor)	32	64
Core frequency (GHz)	2.8	2.45
Family, model, stepping	6, 106, 6	25, 1, 2
SMT	Yes	Yes
Turbo	Yes (3.5 GHz)	Yes (3.5 GHz)
Number of vCPU per VM	16	16
Memory module(s)		
Total memory in system (GB)	64	64
NVMe memory present?	No	No
Total memory (DDR+NVMe RAM) (GB)	64	64
General HW		
Storage	Direct-attached / Instance	Direct-attached / Instance
Local storage		
OS		
Number of drives	1	1
Drive size (GB)	30	30
Drive information	StandardSSD_LRS	StandardSSD_LRS

System configuration information	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors
Network adapter		
Vendor and model	Azure virtual network adapter version=5.13.0-1017-azure	Azure virtual network adapter version=5.13.0-1017-azure
Number and type of ports	1x 50Gb	1x 50Gb

How we tested

Creating the test VM instances

1. Log into the Azure Portal, and navigate to the Virtual Machines.
2. Click Create, and click Virtual Machine.
3. Enter the subscription and resource group names, and give the virtual machine a name.
4. Choose the geographical region for the instance. We used East US.
5. Select the Ubuntu 20.04 image.
6. Select the Standard_D16ds_v5 or Standard_D16ads_v5 instance size, depending on the test target.
7. Select SSH public key, and generate a new key pair for the desired username.
8. Select Allow selected ports, and ensure SSH is enabled.
9. Click Review + create.
10. Click Create.
11. Click Save.

Installing OpenSSL with Intel QuickAssist Technology (QAT) and associated technologies

On each VM, we installed OpenSSL and the performance libraries by following these steps.

1. Log onto the VM as an ordinary user, such as ubuntu.
2. Install prerequisite software:

```
sudo apt update
sudo apt install build-essential checkinstall zlib1g-dev cmake nasm git
```

3. For convenience, define shell variables pointing to source code directories and to the installation location for OpenSSL, its libraries, its engines, and the performance libraries:

```
base=/home/ubuntu/openssl_test
export OPENSSL_SOURCE=$base/openssl_benchmark/openssl
export OPENSSL_LIB=$base/openssl_benchmark/openssl_install
export MB_LOCATION=$base/openssl_benchmark/mb_build
export IPPC_SOURCE=$base/openssl_benchmark/ipp-crypto
export IPSEC_SOURCE=$base/openssl_benchmark/intel-ipsec-mb
export QAT_ENGINE=$base/openssl_benchmark/QAT_Engine
export LD_LIBRARY_PATH=${OPENSSL_LIB}/lib:/lib64
```

4. Create the directory structure, and download source code from GitHub:

```
mkdir -p $base/openssl_benchmark
cd ~base/openssl_benchmark/
git clone https://github.com/openssl/openssl.git
git clone https://github.com/intel/ipp-crypto.git
git clone https://github.com/intel/intel-ipsec-mb.git
git clone https://github.com/intel/QAT_Engine.git
mkdir openssl_install
mkdir mb_build
```

5. Compile and install OpenSSL:

```
cd  $OPENSSL_SOURCE
git checkout OpenSSL_1_1_1l
./config --prefix=$OPENSSL_LIB -Wl,-rpath,$OPENSSL_LIB/lib
make update
make depend
make -j
make install -j
```

6. Compile and install OpenSSL associated performance libraries.
    a. Build Intel Integrated Performance Primitives Cryptography library:

    ```
    cd $IPPC_SOURCE/sources/ippcp/crypto_mb
    git checkout ipp-crypto_2021_5
    cmake . -B"../build" -DOPENSSL_ROOT_DIR=$OPENSSL_LIB -DCMAKE_INSTALL_PREFIX=$MB_LOCATION
    -DOPENSSL_LIBRARIES=$OPENSSL_LIB/lib
    cd ../build
    make -j
    make install -j
    ```

    b. Build Intel Multi-Buffer Crypto for IPsec Library:

    ```
    cd $IPSEC_SOURCE
    git checkout v1.1
    make -j
    make install PREFIX=$MB_LOCATION
    ```

    c. Build Intel QuickAssist Technology OpenSSL engine:

    ```
    cd $QAT_ENGINE
    git checkout v0.6.11
    ./autogen.sh
    ./configure --enable-qat_sw  --disable-qat_hw --with-qat_sw_install_dir=$MB_LOCATION --with-
    openssl_install_dir=$OPENSSL_LIB
    make -j
    make -j install
    ```

Running the tests

On each VM, we performed the OpenSSL testing using the script run.sh (Note: The test writes the results to a CSV file):

```
./run.sh
```

```
File: run.sh
#!/bin/bash
MODE=QAT
case $MODE in
DEF)
  echo "Using OS default libraries for openssl"
  unset OPENSSL_ENGINES
  unset LD_LIBRARY_PATH
  ENG=""
  ;;
QAT)
  echo "Using new and QAT libraries for openssl"
  base=/home/ubuntu/openssl_test
  OPENSSL_LIB=$base/openssl_benchmark/openssl_install
```

```
  export LD_LIBRARY_PATH=${OPENSSL_LIB}/lib:/lib64
  export OPENSSL_ENGINES=$OPENSSL_LIB/lib/engines-1.1
  ENG="-engine qatengine"
  unset base
  unset OPENSSL_LIB
  ;;
*)
  echo "Unknown mode = \"$MODE\". Please set."
  exit 2
  ;;
esac
OPENSSL="$base/openssl_install/bin/openssl"
HT="No"
CORELIST="1,3,5,7,9,11,13,15"
NUM_PROCS=8
test_type="hash sym rsa ecdsa ecdh ed"
ASYNC_JOBS="-async_jobs 8"
echo ""
echo "Using async_jobs option set to \"$ASYNC_JOBS\""
echo "OPENSSL_ENGINES = \"$OPENSSL_ENGINES\""
echo "Engine option set to \"$ENG\""
echo "LD_LIBRARY_PATH = \"$LD_LIBRARY_PATH\""
echo "Using OpenSSL = \"$OPENSSL\""

main() {
  clean_up
  run_bench $@
  clean_up
  echo "##############################"
  echo ""
}
hash_parse() {
  test_ran=$(grep -i -m1 "got.*\+f:" temp.txt|awk 'BEGIN{FS=":";}{printf("%s\n",$4)}')
  perf=$(tail -n 1 temp.txt|awk 'BEGIN{FS=":";}{printf("%f\n",$NF)}')
  echo "$perf" >>temp_perf.txt
  cpb=$(bc -l <<< "$freq/$perf")
  echo "$cpb" >>temp_data.txt
  echo "$test_ran" >>test_ran.txt
  paste -d, header.txt test_ran.txt temp_perf.txt temp_data.txt \
    >>openssl_qatsw_"$SUFFIX".csv
  cp temp.txt "temp_${test_ran}-${NUM_PROCS}_${SUFFIX}".txt
  rm -f temp.txt temp_data.txt test_ran.txt temp_perf.txt
}
sym_parse() {
  test_ran=$(grep -i -m1 "got.*\+f:" temp.txt|awk 'BEGIN{FS=":";}{printf("%s\n",$4)}')
  test_ran="${test_ran}_enc"
  perf=$(tail -n 1 temp.txt|awk 'BEGIN{FS=":";}{printf("%f\n",$NF)}')
  echo "$perf" >>temp_perf.txt
  cpb=$(bc -l <<< "$freq/$perf")
  echo "$cpb" >>temp_data.txt
  echo "$test_ran" >>test_ran.txt
  paste -d, header.txt test_ran.txt temp_perf.txt temp_data.txt \
    >>openssl_qatsw_"$SUFFIX".csv
  cp temp.txt "temp_${test_ran}-${NUM_PROCS}_${SUFFIX}".txt
  rm -f temp.txt temp_data.txt test_ran.txt temp_perf.txt
}
ecdh_parse() {
  test_ran=$(grep -i -m1 "got.*+f[0-9]:" temp.txt|awk 'BEGIN{FS=":";}{printf("%s\n",$4)}')
  test_ran="ecdh${test_ran}_op"
  perf=$(tail -n 1 temp.txt|awk 'BEGIN{FS=":";}{printf("%f\n",$4)}')
  echo "$perf" >>temp_perf.txt
  cpb=$(bc -l <<< "$freq/$perf")
  echo "$cpb" >>temp_data.txt
  echo "$test_ran" >>test_ran.txt
  paste -d, header.txt test_ran.txt temp_perf.txt temp_data.txt \
    >>openssl_qatsw_"$SUFFIX".csv
  cp temp.txt "temp_${test_ran}-${NUM_PROCS}_${SUFFIX}".txt
  rm -f temp.txt temp_data.txt test_ran.txt temp_perf.txt
}
rsa_parse() {
  test_ran=$(grep -i -m1 "got.*+f[0-9]:" temp.txt|awk 'BEGIN{FS=":";}{printf("%s\n",$4)}')
  stest_ran="rsa${test_ran}_sign"
  vtest_ran="rsa${test_ran}_verify"
  sperf=$(tail -n 1 temp.txt|awk 'BEGIN{FS=":";}{printf("%f\n",$4)}')
  vperf=$(tail -n 1 temp.txt|awk 'BEGIN{FS=":";}{printf("%f\n",$5)}')
  echo "$sperf" >>temp_sperf.txt
  echo "$vperf" >>temp_vperf.txt
  scpb=$(bc -l <<< "$freq/$sperf")
  vcpb=$(bc -l <<< "$freq/$vperf")
  echo "$scpb" >>stemp_data.txt
  echo "$vcpb" >>vtemp_data.txt
  echo "$stest_ran" >>stest_ran.txt
  echo "$vtest_ran" >>vtest_ran.txt
  paste -d, header.txt stest_ran.txt temp_sperf.txt stemp_data.txt \
    >>openssl_qatsw_"$SUFFIX".csv
  paste -d, header.txt vtest_ran.txt temp_vperf.txt vtemp_data.txt \
    >>openssl_qatsw_"$SUFFIX".csv
  cp temp.txt "temp_rsa${test_ran}-${NUM_PROCS}_${SUFFIX}".txt
  rm -f temp.txt stemp_data.txt vtemp_data.txt stest_ran.txt \
    vtest_ran.txt temp_sperf.txt temp_vperf.txt
}
ecdsa_parse() {
  test_ran=$(grep -i -m1 "got.*+f[0-9]:" temp.txt|awk 'BEGIN{FS=":";}{printf("%s\n",$4)}')
  stest_ran="ecdsa${test_ran}_sign"
  vtest_ran="ecdsa${test_ran}_verify"
  sperf=$(tail -n 1 temp.txt|awk 'BEGIN{FS=":";}{printf("%f\n",$4)}')
  vperf=$(tail -n 1 temp.txt|awk 'BEGIN{FS=":";}{printf("%f\n",$5)}')
  echo "$sperf" >>temp_sperf.txt
  echo "$vperf" >>temp_vperf.txt
  scpb=$(bc -l <<< "$freq/$sperf")
  vcpb=$(bc -l <<< "$freq/$vperf")
  echo "$scpb" >>stemp_data.txt
  echo "$vcpb" >>vtemp_data.txt
  echo "$stest_ran" >>stest_ran.txt
  echo "$vtest_ran" >>vtest_ran.txt
  paste -d, header.txt stest_ran.txt temp_sperf.txt stemp_data.txt \
    >>openssl_qatsw_"$SUFFIX".csv
  paste -d, header.txt vtest_ran.txt temp_vperf.txt vtemp_data.txt \
    >>openssl_qatsw_"$SUFFIX".csv
  cp temp.txt "temp_ecdsa${test_ran}-${NUM_PROCS}_${SUFFIX}".txt
  rm -f temp.txt stemp_data.txt vtemp_data.txt stest_ran.txt \
    vtest_ran.txt temp_sperf.txt temp_vperf.txt
}
ed_parse() {
  test_ran=$(grep -i -m1 "got.*+f[0-9]:" temp.txt|awk 'BEGIN{FS=":";}{printf("%s\n", $5)}')
  stest_ran="${test_ran}_sign"
  vtest_ran="${test_ran}_verify"
  sperf=$(grep -i "got.*+f[0-9]:" temp.txt|awk 'BEGIN{FS=":";}{printf("%f\n",$((NF-1)))}' \
    |awk '{s+=$1}END{print s}')
  vperf=$(grep -i "got.*+f[0-9]:" temp.txt|awk 'BEGIN{FS=":";}{printf("%f\n",$NF)}' \
    |awk '{s+=$1}END{print s}')
  echo "$sperf" >>temp_sperf.txt
  echo "$vperf" >>temp_vperf.txt
  scpb=$(bc -l <<< "$freq/$sperf")
  vcpb=$(bc -l <<< "$freq/$vperf")
  echo "$scpb" >>stemp_data.txt
  echo "$vcpb" >>vtemp_data.txt
  echo "$stest_ran" >>stest_ran.txt
  echo "$vtest_ran" >>vtest_ran.txt
  paste -d, header.txt stest_ran.txt temp_sperf.txt stemp_data.txt \
    >>openssl_qatsw_"$SUFFIX".csv
  paste -d, header.txt vtest_ran.txt temp_vperf.txt vtemp_data.txt \
    >>openssl_qatsw_"$SUFFIX".csv
  cp temp.txt "temp_ed${test_ran}-${NUM_PROCS}_${SUFFIX}".txt
  rm -f temp.txt stemp_data.txt vtemp_data.txt stest_ran.txt \
    vtest_ran.txt temp_sperf.txt temp_vperf.txt
}

run_speed() {
  if [ -f temp.txt ]; then
    rm -f temp.txt
  fi
  for test in $test_type; do
    echo "Test set = $test"
    case $test in
    hash)
      for hash in sha256; do
        taskset -c $CORELIST $OPENSSL speed $ENG $ASYNC_JOBS -multi $NUM_PROCS \
          -mr $hash 2>&1 |tee >>temp.txt
        hash_parse
      done
      ;;
    sym)
      for sym in aes-256-gcm; do
        taskset -c $CORELIST $OPENSSL speed $ENG $ASYNC_JOBS -multi $NUM_PROCS \
          -mr -evp $sym 2>&1 |tee >>temp.txt
        sym_parse
      done
      ;;
    rsa)
      for rsa in rsa2048; do
        taskset -c $CORELIST $OPENSSL speed $ENG $ASYNC_JOBS -multi $NUM_PROCS \
          -mr $rsa 2>&1 |tee >>temp.txt
        rsa_parse
      done
      ;;
    ecdsa)
      for ecdsa in ecdsap256; do
        taskset -c $CORELIST $OPENSSL speed $ENG $ASYNC_JOBS -multi $NUM_PROCS \
          -mr $ecdsa 2>&1 |tee >>temp.txt
        ecdsa_parse
      done
      ;;
    ecdh)
      for ecdh in ecdhp256; do
        taskset -c $CORELIST $OPENSSL speed $ENG $ASYNC_JOBS -multi $NUM_PROCS \
          -mr $ecdh 2>&1 |tee >>temp.txt
        ecdh_parse
      done
      ;;
    ed)
      for ed in ed25519; do
        taskset -c $CORELIST $OPENSSL speed $ENG $ASYNC_JOBS -multi $NUM_PROCS \
          -mr $ed 2>&1 |tee >>temp.txt
        ed_parse
      done
      ;;
    *)
      echo "Error: $test is unmatched. Continuing..."
      ;;
    esac
  done
}

clean_up() {
  rm -f header.txt
}

run_bench() {
  export SUFFIX=${MODE}_$(date +"%Y%m%d_%H%M")
  freq=$(lscpu |grep -i "cpu\smhz" |awk '{printf("%d", $3)}')
  freq=$(bc -l <<< "$freq*1000*1000")

  echo "Running ($test_type) in mode $MODE with $NUM_PROCS procs on cpus $CORELIST"
  echo "File suffix is \"$SUFFIX\""
  clean_up
  get_system_info
  run_speed $@
}

get_system_info() {
 {
  descriptor=$RANDOM$RANDOM
  echo "$descriptor" |awk '{printf $1 "," }'
  gcc --version |grep gcc |awk '{printf $NF ","}' |sed 's/\./_/g'
  ld -v |grep GNU |awk '{printf $7 ","}' |sed 's/\./_/g'
  as --version |grep Binutils |awk '{printf $7 ","}' |sed 's/\./_/g'
  awk '{printf("%s %s ",$1,$2)}' /etc/issue|sed 's/\([1-Z]\)\s\([1-Z]\)/\1_\2/g' \
    |sed 's/\./_/g' |awk '{printf $1 ","}'
  uname -r |sed 's/-/_/g' |awk '{printf $1 }' |sed 's/\./_/g'
  echo "," |awk '{printf $1}'>>header.txt
  $OPENSSL version |sed 's/\s/_/g' |awk '{printf $1","}'
  printf "%s%\n" $HT
 }>>header.txt
}

main $@
```

LAMMPS testing

We concluded our hands-on testing on March 10, 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 March 30, 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 5: Results of our testing.

	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors	Advantage
SPECint			
SPECrate 2017 Integer base rate median score (higher is better)	71.9	66.0	D16ds_v5 with Intel Xeon Scalable processors 8.3% higher
LAMMPS			
LAMMPS median score (higher is better)	52.9	32.6	D16ds_v5 with Intel Xeon Scalable processors 59.0% more timesteps per seconds

System configuration information

Table 6: Detailed information on the VMs we tested.

System configuration information	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors
Tested by	Principled Technologies	Principled Technologies
Test date	3/30/2022	3/30/2022
CSP / Region	Microsoft Azure / East US	Microsoft Azure / East US
Workload & version	LAMMPS Sep 29 2021	LAMMPS Sep 29 2021
Iterations and result choice	3 runs, median	3 runs, median
Server platform	Standard_D16ds_v5	Standard_D16ads_v5
BIOS name and version	American Megatrends Inc. 090008 (12/07/2018)	American Megatrends Inc. 090008 (12/07/2018)
Operating system name and version/ build number	Ubuntu 20.04.4 LTS	Ubuntu 20.04.4 LTS
Date of last OS updates/patches applied	3/30/2022	3/30/2022

System configuration information	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors
Processor		
Number of processors	1	1
Vendor and model	Intel Xeon Platinum 8370C CPU @ 2.80GHz	AMD EPYC 7763 64-Core Processor
Core count (per processor)	32	64
Core frequency (GHz)	2.8	2.45
Family, model, stepping	6, 106, 6	25, 1, 2
SMT	Yes	Yes
Turbo	Yes (3.5 GHz)	Yes (3.5 GHz)
Number of vCPU per VM	16	16
Memory module(s)		
Total memory in system (GB)	64	64
NVMe memory present?	No	No
Total memory (DDR+NVMe RAM) (GB)	64	64
General HW		
Storage	Direct-attached / Instance	Direct-attached / Instance
Local storage		
OS		
Number of drives	1	1
Drive size (GB)	30	30
Drive information	Standard HDD LRS	Standard HDD LRS
Data drive		
Number of drives	1	1
Drive size (GB)	30	30
Drive information	Standard HDD LRS	Standard HDD LRS
Network adapter		
Vendor and model	Azure virtual network adapter version=5.13.0-1017-azure	Azure virtual network adapter version=5.13.0-1017-azure
Number and type of ports	1x 50Gb	1x 50Gb

How we tested

Creating the Ubuntu Server 20.04 VM for LAMMPS

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 image Ubuntu Server 20.04 LTS - Gen2
    g. Select the instance size D16ds_v5 or D16ads_v5.
    h. Choose a username and key pair.
    i. Leave Public inbound ports set to Allow selected ports.
    j. 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.
9. Once the VM creation is finished, click Go to resource (or navigate to the virtual machine service, and click the new VM).

Building LAMMPS on the D16ds_v5 VM

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. Download and install Intel OneAPI Base Kit:

```
wget https://registrationcenter-download.intel.com/akdlm/irc_nas/18487/l_
BaseKit_p_2022.1.2.146_offline.sh
sudo sh ./l_BaseKit_p_2022.1.2.146_offline.sh
```

4. Download and install Intel OneAPI HPC Kit:

```
wget https://registrationcenter-download.intel.com/akdlm/irc_nas/18479/l_
HPCKit_p_2022.1.2.117_offline.sh
sudo sh ./l_HPCKit_p_2022.1.2.117_offline.sh
```

5. Install git, Cmake, pkg-config and GNU development tools:

```
sudo apt -y install git cmake make pkg-config build-essential
```

6. Source environment variables for the Intel OneAPI kit you just installed:

```
source /opt/intel/oneapi/setvars.sh
```

7. Download LAMMPS source code:

```
git clone https://github.com/lammps/lammps.git
```

8. Build LAMMPS using Intel oneAPI compilers and LAMMPS/INTEL package:

```
cd lammps
mkdir build
cd build
cmake ../cmake -D CMAKE_CXX_COMPILER=icpx -D CMAKE_C_COMPILER=icx -D PKG_INTEL=yes -D
PKG_MANYBODY=yes
cmake --build .
```

9. Download the 3D Lennard-Jones atomic fluid simulation settings:

```
wget -O in.lj https://raw.githubusercontent.com/lammps/lammps/develop/src/INTEL/TEST/in.intel.lj
```

10. Run the benchmark with one MPI task per core:

```
mpirun -np 8 ./lmp -in ~/in.lj -sf intel -v N on
```

Building LAMMPS on the D16ads_v5 VM

1. In the Connect tab for the VM, log into the test VM using the SSH command.
2. Install the latest updates on the VM:

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

3. Install git, Cmake, make, and GNU development tools:

```
sudo apt -y install git cmake make g++ build-essential
```

4. Download the LAMMPS source code:

```
git clone https://github.com/lammps/lammps.git
```

5. Download AOCC compiler aocc-compiler-3.2.0.tar from **https://developer.amd.com/amd-aocc/**.
6. Install the AOCC compiler:

```
tar -xvf aocc-compiler-3.2.0.tar
cd aocc-compiler-3.2.0/
bash install.sh
source ./setenv_AOCC.sh
```

7. Building LAMMPS with AOCC compiler:

```
spack -d install -v -j 16 lammps@20220217 build_type=Release %aocc@3.2.0 target=zen3 ~kim
+asphere +class2 +kspace +manybody +molecule +mpiio +opt +replica +rigid +granular +user-omp
+openmp +kokkos ^amdfftw@3.1 ^openmpi@4.1.1 fabrics=auto
```

8. Load the LAMMPS build with AOCC 3.2.0 module into the environment:

```
spack load lammps@20220217 %aocc@3.2.0
```

9. Download the 3D Lennard-Jones atomic fluid simulation settings:

```
wget -O in.lj https://raw.githubusercontent.com/lammps/lammps/develop/src/INTEL/TEST/in.intel.lj
```

10. Run the benchmark with one MPI task per core:

```
export LMP_MPI=/<path_to_lammps_installed_folder>/bin/lmp
mpirun -np 8  $LMP_MPI -v N on -in ~/in.lj -k on -sf kk -v N on
```

SPECrate 2017 Floating Point (SPECfp) testing

We concluded our hands-on testing on February 10, 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 February 10, 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 7: Results of our testing.

	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors	Advantage
SPECint			
SPECrate 2017 Integer base rate median score (higher is better)	71.9	66.0	D16ds_v5 with Intel Xeon Scalable processors 8.3% higher
SPECfp			
SPECrate 2017 Floating Point median score (higher is better)	100.0	63.3	D16ds_v5 with Intel Xeon Scalable processors 58.4% faster fp_base rate

System configuration information

Table 8: Detailed information on the VMs we tested.

System configuration information	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors
Tested by	Principled Technologies	Principled Technologies
Test date	2/10/2022	2/10/2022
CSP / Region	Microsoft Azure / East US	Microsoft Azure / East US
Workload & version	SPEC CPU2017 base Floating-Point rate	SPEC CPU2017 base Floating-Point rate
Workload-specific parameters	None	None
Iterations and result choice	3 runs, median	3 runs, median
Server platform	Standard_D16ds_v5	Standard_D16ads_v5
BIOS name and version	American Megatrends Inc. 090008 (12/07/2018)	American Megatrends Inc. 090008 (12/07/2018)
Operating system name and version/ build number	Red Hat Enterprise Linux 8.5 4.18.0-348.12.2.el8_5.x86_64	Red Hat Enterprise Linux 8.5 4.18.0-348.12.2.el8_5.x86_64
Date of last OS updates/patches applied	2/10/2022	2/10/2022

System configuration information	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors
Processor		
Number of processors	1	1
Vendor and model	Intel Xeon Platinum 8370C CPU @ 2.80GHz	AMD EPYC 7763 64-Core Processor
Core count (per processor)	32	64
Core frequency (GHz)	2.8	2.45
Family, model, stepping	6, 106, 6	25, 1, 2
SMT	Yes	Yes
Turbo	Yes (3.5 GHz)	Yes (3.5 GHz)
Number of vCPU per VM	16	16
Memory module(s)		
Total memory in system (GB)	64	64
NVMe memory present?	No	No
Total memory (DDR+NVMe RAM) (GB)	64	64
General HW		
Storage	Direct-attached / Instance	Direct-attached / Instance
Local storage		
OS		
Number of drives	1	1
Drive size (GB)	30	30
Drive information	StandardSSD_LRS	StandardSSD_LRS
Data drive		
Number of drives	1	1
Drive size (GB)	256	256
Drive information	StandardSSD_LRS	StandardSSD_LRS
Network adapter		
Vendor and model	Azure virtual network adapter version=5.13.0-1017-azure	Azure virtual network adapter version=5.13.0-1017-azure
Number and type of ports	1x 50Gb	1x 50Gb

How we tested

Creating the Red Hat Enterprise Linux 8.5 VM for SPEC CPU 2017

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 image Red Hat Enterprise Linux 8.5 (LVM) - Gen2.
    g. Select the instance size D16ds_v5 or D16ads_v5.
    h. Choose a username and key pair.
    i. Leave Public inbound ports set to Allow selected ports.
    j. 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.
    c. Click Create and attach a new disk.
    d. Click Change Size.
    e. Under Disk SKU, select Standard SSD (locally-redundant storage).
    f. Select 256 GiB, and click OK.
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.
9. Once the VM creation is finished, click Go to resource (or navigate to the virtual machine service and click the new VM).

Updating Red Hat Enterprise Linux 8.5 and Installing SPEC CPU 2017

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 yum update -y
sudo yum install -y libnsl numactl
sudo dnf install -y glibc.i686
```

3. Attach data disk to the VM:

```
echo ';' | sudo sfdisk /dev/sda
sudo mkfs.ext4 /dev/sda1
sudo mkdir /data
sudo mount /dev/sda1 /data
sudo chmod 777 /data
```

4. Copy cpu2017-1.1.8.iso to the VM.
5. Mount the ISO to the mnt folder:

```
sudo mount -t iso9660 -o ro,exec,loop /data/cpu2017-1.1.8.iso /mnt
```

6. Install SPEC CPU 2017 base:

```
cd /mnt
./install.sh -d /data/cpu2017 -u linux-x86_64
```

7. Copy compiled binaries package to the VM.
8. Extract the package to the install directory.

```
xz -dck /data/Intel-cpu2017-1.1.8-ic2021.4-linux-binaries-20210924.tar.xz | tar -xf -
tar -xf /data/AMD-cpu2017-1.1.5-aocc3-lin-base-binaries-20210317_dt.tar
```

9. Source variables for testing.

```
cd /data/cpu2017
source shrc
```

10. Prepare the Intel Xeon Scalable processor-powered VMs:

```
sudo sh -c "sync; echo 3> /proc/sys/vm/drop_caches"
```

11. Prepare the AMD EPYC processor-powered VMs:

```
sudo su
yum install -y clang
yum install -y elfutils-libelf-devel
yum install -y llvm
yum instally cmake
yum install -y libquadmath
echo always > /sys/kernel/mm/transparent_hugepage/enabled
echo 1 > /proc/sys/vm/zone_reclaim_mode
echo 8 > /proc/sys/vm/dirty_ratio
echo 1 > /proc/sys/vm/swappiness
echo 0 > /proc/sys/kernel/randomize_va_space
echo always > /sys/kernel/mm/transparent_hugepage/defrag
sync; echo 3> /proc/sys/vm/drop_caches
exit
```

12. Execute run script:

```
Intel
./run_fp_rate_ic2021.4-lin-core-avx512-rate-smt-on-20210924.sh
zip -r INTEL_RATE_FP_02082022.zip result

AMD
./run_fprate_aocc3.0-lin-znver3-rate-smt-on-20210317.sh
zip -r AMD_RATE_FP_02082022.zip result
```

ResNet 50 testing

We concluded our hands-on testing on March 29, 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 March 28, 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 9: Results of our testing.

	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors	Advantage
SPECint			
SPECrate 2017 Integer base rate median score (higher is better)	71.9	66.0	D16ds_v5 with Intel Xeon Scalable processors 8.3% higher
ResNet50			
INT8 - BS1 median score (higher is better)	154.415	136.473	D16ds_v5 with Intel Xeon Scalable processors 13.0% more samples per second
INT8 - BS128 median score (higher is better)	265.947	172.548	D16ds_v5 with Intel Xeon Scalable processors 54.0% more samples per second

System configuration information

Table 10: Detailed information on the VMs we tested.

System configuration information	D16ds_v5 with Intel Xeon Scalable processors			D16ads_v5 with AMD EPYC processors		
Tested by	Principled Technologies			Principled Technologies		
Test date		BS=1	BS=128		BS=1	BS=128
	INT8	3/29/2022	3/28/2022	INT8	3/29/2022	3/28/2022
CSP / Region	Microsoft Azure / East US			Microsoft Azure / East US		
Workload & version	Model Zoo for Intel Architectures v2.6.2: resnet50v1_5			Model Zoo for Intel Architectures v2.6.2: resnet50v1_5		
Workload-specific parameters	*See Tables 14 and 15*			*See Tables 14 and 15*		
		BS=1	BS=128		BS=1	BS=128
	INT8	C	C	INT8	C	C
Iterations and result choice	3 runs, median			3 runs, median		
Server platform	Standard_D16ds_v5			Standard_D16ads_v5		
BIOS name and version	American Megatrends Inc. 090008 (12/07/2018)			American Megatrends Inc. 090008 (12/07/2018)		
Operating system name and version/ build number	Ubuntu 20.04 5.13.0-1017-azure			Ubuntu 20.04 5.13.0-1017-azure		
Date of last OS updates/patches applied		BS=1	BS=128		BS=1	BS=128
	INT8	3/29/2022	3/28/2022	INT8	3/29/2022	3/28/2022
Number of processors	1			1		
Vendor and model	Intel Xeon Platinum 8370C CPU @ 2.80GHz			AMD EPYC 7763 64-Core Processor		
Core count (per processor)	32			64		
Core frequency (GHz)	2.8			2.45		
Family, model, stepping	6, 106, 6			25, 1, 2		
SMT	Yes			Yes		
Turbo	Yes (3.5 GHz)			Yes (3.5 GHz)		
Number of vCPU per VM	16			16		
Memory module(s)						
Total memory in system (GB)	64			64		
NVMe memory present?	No			No		
Total memory (DDR+NVMe RAM) (GB)	64			64		
General HW						
Storage	Direct-attached / Instance			Direct-attached / Instance		

System configuration information	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors
Local storage		
OS		
Number of drives	1	1
Drive size (GB)	30	30
Drive information	StandardSSD_LRS	StandardSSD_LRS
Data drive		
Number of drives	1	1
Drive size (GB)	200	200
Drive information	StandardSSD_LRS	StandardSSD_LRS
Network adapter		
Vendor and model	Azure virtual network adapter version=5.13.0-1017-azure	Azure virtual network adapter version=5.13.0-1017-azure
Number and type of ports	1x 50Gb	1x 50Gb

How we tested

We started with Azure VMs with Intel Xeon Scalable or AMD EPYC processors and an Ubuntu operating system, installed Docker and Python, and built Docker containers with Model Zoo for Intel Architectures and prerequisites, such as TensorFlow, tcmalloc, and more. We ran the benchmark in containers with settings we determined from a tuning process that explored the impact of various tunable parameters. For each configuration/host, we performed the test three times and selected the run with the median throughput (frames/s) as the characteristic run. We compared the median runs to those with similar geometry in either processor vendor (Intel vs. AMD).

For the VM with Intel Xeon Scalable processors, we used the unaltered Model Zoo code. For the VM with AMD EPYC processors, we altered the Docker image build process to incorporate ZenDNN and use an AMD-optimized analog of the Intel-optimized default stack in Model Zoo for Intel Architectures. We accomplished this with a set of wrapper scripts to provide a common interface for both AMD and Intel testing. This code is available upon request by emailing **info@principledtechnologies.com**.

Software stacks and versions

For these tests, we aimed to provide software stacks with identical framework and library versions where possible. However, we were unable to match versions exactly due to constraints of software compatibility. Table 11 defines the software versions we used for the primary stack components for each test configuration. Note: These refer to the containers in which the benchmark runs, and not the host operating system of the Azure images.

Table 11: Software versions for primary stack components.

CPU vendor	Intel		AMD	
Model	Resnet50 v1.5		Resnet50 v1.5	
Precision	FP32	INT8	FP32	INT8
Operating system	Ubuntu 20.04	Ubuntu 20.04	Ubuntu 20.04	Ubuntu 20.04
Model Zoo for Intel Architetures	2.6.2	2.6.2	2.6.2*	2.6.2*
TensorFlow	2.6.2	2.6.2	2.7.0	2.5.0
Python	3.8.10	3.8.10	3.8.10	3.8.10
ZenDNN	N/A		3.2	3.2

*We made modifications to support AMD processors and describe these changes in the appropriate sections below.

Creating the test instance

1. Log into the Azure Portal, and navigate to the Virtual Machines.
2. Click Create, and click Virtual Machine.
3. Enter the subscription and resource group names, and give the virtual machine a name.
4. Choose the geographical region for the instance. We used East US.
5. Select the Ubuntu Server 20.04 LTS – Gen 1 image.
6. Select the Standard_D16ds_v5 or Standard_D16ads_v5 instance size, depending on the test target.
7. Select SSH public key, and generate a new key pair for the desired username.
8. Select Allow selected ports, and ensure SSH is enabled.
9. Click Next: Disks.
10. Set OS disk type to Standard SSD
11. Click Create and attach a new disk.
12. Click Change Size.
13. Select 200 GiB size and Standard SSD tier.
14. Click OK twice.
15. Click Review + create.
16. Click Create.

Installing the prerequisite software packages

1. SSH into the instance:

```
ssh -i INSTANCE_SSH_KEY_FILE ubuntu@INSTANCE_IP
```

2. Become root:

```
sudo su
```

3. Update APT cache:

```
apt-get update -y
```

4. Install utility packages:

```
apt-get install -y git wget htop curl tree jq ntp ntpdate openssh-server rsync
```

5. Configure GIT:

```
git config --global user.email "YOUR_EMAIL_ADDRESS"
git config --global user.name "YOUR_NAME"
```

6. Install Python3:

```
apt-get install -y build-essential python3 python3-setuptools python3-dev python3-wheel \
python3-pip python3-venv
```

7. Upgrade Python PIP:

```
pip install --upgrade pip
```

8. Install Docker prerequisites:

```
apt-get install -y aptitude expect ca-certificates gnupg lsb-release
```

9. Add Docker's official GPG key:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o \
/usr/share/keyrings/docker-archive-keyring.gpg
```

10. Add the stable Docker repository to APT:

```
echo  "deb [arch=$(dpkg --print-architecture) \
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
```

11. Update APT:

```
cacheapt-get update -y
```

12. Install Docker Engine:

```
apt-get install docker-ce docker-ce-cli containerd.io
```

13. Verify Docker has installed correctly:

```
docker run -it --rm hello-world
```

14. Install Python packages useful for working with Docker:

```
pip install docker jsondiff docker-compose enum34
```

15. Install NMON:

```
apt-get install -y nmon
```

Preparing the data volume

1. SSH into the instance:

```
ssh -i INSTANCE_SSH_KEY_FILE ubuntu@INSTANCE_IP
```

2. Become root:

```
sudo su
```

3. Create the persistent volume mount point:

```
mkdir /persistent
```

4. Create a partition on the persistent volume:

```
parted /dev/nvme1n1
mktable gpt
mkpart none ext2 2048kb 100%
print
quit
```

5. Format the persistent volume with EXT4:

```
mkfs -t ext4 /dev/nvme1n1p1
```

6. Mount the persistent volume:

```
mount -t ext4 /dev/nvme1n1p1 /persistent
```

7. Add the persistent volume to /etc/fstab:

```
echo -e "/dev/nvme1n1p1\t/persistent\text4\tdefaults 0 0" >> /etc/fstab
```

8. Create folder structure on persistent volume:

```
mkdir -p /persistent/datasets/resnetmkdir -p /persistent/models/resnet50v1_5
```

9. Follow external instructions to prepare the dataset: **https://github.com/IntelAI/models/blob/master/datasets/imagenet/README.md**
10. Once complete, copy the dataset to persistent volume, and delete the directory you created while getting the dataset:

```
mv ${IMAGENET_DIR}/* \ /persistent/models/resnet/
rm -rf ${IMAGENET_DIR}
```

11. Download the pretrained model for precision FP32:

```
cd /persistent/models/resnet50v1_5
wget https://zenodo.org/record/2535873/files/resnet50_v1.pb
```

12. Download the pretrained model for precision INT8:

```
cd /persistent/models/resnet50v1_5
wget
https://storage.googleapis.com/intel-optimized-tensorflow/models/v1_8/resnet50v1_5_int8_
pretrained_model.pb
```

Gathering the base image resources (for VMs with AMD EPYC processors only)

For steps 4 and 7, please contact Principled Technologies at **info@principledtechnologies.com** for the script and Docker file. The ZenDNN installer script installs ZenDNN 3.2 and creates an environment file used when running the benchmark, and the ZenDNN 3.2 TensorFlow 2.7.0 Dockerfile defines the docker image with software stack including TensorFlow 2.7.0 and integrated ZenDNN 3.2.

1. On your local machine, open a terminal.
2. Create a directory structure:

```
mkdir -p /tmp/amd-base-image/build/tensorflow-v2.7.0/binaries
mkdir -p /tmp/amd-base-image/build-common/binaries
```

3. Download AMD AOCL 3.0 for Linux:
    a. In a web browser, navigate to **https://developer.amd.com/amd-aocl/archives/**.
    b. Scroll to the section AOCL 3.0 compiled with AOCC 3.0.
    c. Click aocl-linux-aocc-3.0-6_1_amd64.deb, and download it.
    d. Save the file as /tmp/amd-base-image/build-common/binaries/aocl-linux-aocc-3.0-6_1_amd64.deb.
4. Create the ZenDNN installer script:
    a. Contact **info@principledtechnologies.com** to request a copy of the ZenDNN installer script.
    b. Create a new file with the nano editor:

    ```
    nano /tmp/amd-base-image/build-common/binaries/install-zendnn.sh
    ```

    c. Copy and paste the content from the script you received in step 4a.
    d. To save the file, press CTRL+O.
    e. To exit nano, press CTRL+X.
5. Download AMD AOCC compiler 3.1:
    a. In a web browser, navigate to **https://developer.amd.com/aocc-archive-section/**.
    b. Scroll down to the section AOCC v3.1.
    c. Click aocc-compiler-3.1.0_1_amd64.deb, and download it.
    d. Save the file as /tmp/amd-base-image/build/tensorflow-v2.7.0/binaries/aocc-compiler-3.1.0_1_amd64.deb.
6. Download AMD ZenDNN 3.2:
    a. In a web browser,navigate to **https://developer.amd.com/zendnn/zendnn-archives/**.
    b. Scroll down to the section ZenDNN 3.2.
    c. Click TF_v2.7_ZenDNN_v3.2_Python_v3.8.zip, and download it.
    d. Save the file as /tmp/amd-base-image/build/tensorflow-v2.7.0/binaries/TF_v2.7_ZenDNN_v3.2_Python_v3.8.zip.

7. Create the TensorFlow v2.7.0 Docker file.
    a. Contact **info@principledtechnologies.com** to request a copy of the ZenDNN installer script.
    b. Create a new file with the nano editor:

    ```
    nano /tmp/amd-base-image/build/tensorflow-v2.7.0/Dockerfile
    ```

    c. Copy and paste the content from the script you received in step 7a.
    d. To save the file, press CTRL+O.
    e. To exit nano, press CTRL+X.

Preparing the AMD TensorFlow 2.7.0 base image (for VMs with AMD EPYC processors only)

1. Copy the build files to the Azure VM:

```
cd /tmp/amd-base-image/build/
rsync -rav -e "ssh -i INSTANCE_SSH_KEY_FILE" \
./tensorflow-v2.7.0/ ubuntu@INSTANCE_IP:~/amd-tensorflow-v2.7.0/
rsync -rav -e "ssh -i INSTANCE_SSH_KEY_FILE" \
../build-common/ ubuntu@INSTANCE_IP:~/amd-tensorflow-v2.7.0/
```

2. SSH into the instance:

```
ssh -i INSTANCE_SSH_KEY_FILE ubuntu@INSTANCE_IP
```

3. Become root:

```
sudo su
```

4. Change to the build directory:

```
cd /home/ubuntu/amd-tensorflow-v2.7.0/
```

5. Build and tag the Docker image:

```
docker build -f Dockerfile -t amd/tensorflow:v2.7.0
```

Installing Model Zoo for Intel Architectures

1. SSH into the instance:

```
ssh -i INSTANCE_SSH_KEY_FILE ubuntu@INSTANCE_IP
```

2. Become root:

```
sudo su
```

3. Create the install directory:

```
mkdir -p /opt/imz-tools/intel-model-zoo
```

4. Change to the install directory's parent directory:

```
cd /opt/imz-tools
```

5. Clone the Model Zoo git repository:

```
git clone https://github.com/IntelAI/models ./intel-model-zoo
```

6. Check out version 2.6.2:

```
git checkout v2.6.2
```

7. Create a base branch:

```
git checkout -b v2.6.2-base
```

Applying a patch file for AMD support (for VMs with AMD EPYC processors only)

For step 1, please contact Principled Technologies at **info@principledtechnologies.com** for the AMD patch. The patch creates AMD-suitable Docker image slice definitions and provides a small patch to the benchmark CPU binding logic. Note: The primary change accomplished with alternate slice definitions is the replacement of the base image; aside from some naming-related differences and the small patch to CPU binding logic, the software stack above the base image is identical.

1. Contact **info@principledtechnologies.com** to request a copy of the AMD/ZenDNN patch for Model Zoo for Intel Architectures.
2. Save the patch file as ~/amd.patch.
3. Copy the AMD patch files from your local machine to the instance:

```
rsync -e "ssh -I INSTANCE_SSH_KEY_FILE" -av ~/amd.patch ubuntu@INSTANCE_IP:~/amd.patch
```

4. SSH into the instance:

```
ssh -i INSTANCE_SSH_KEY_FILE ubuntu@INSTANCE_IP
```

5. Become root:

```
sudo su
```

6. Change to the install directory:

```
cd /opt/imz-tools/intel-model-zoo
```

7. Check out the base branch:

```
git checkout v2.6.2-base
```

8. Create a new branch to hold patched version of code:

```
git checkout -b v2.6.2-amd
```

9. Apply patch file to the newly created branch:

```
git apply /home/ubuntu/amd.patch
```

10. Commit changes to the branch:

```
git add . && git commit -m "Applied AMD patch files."
```

11. Remove the patch file:

```
rm -rf /home/ubuntu/amd.patch
```

Creating the model builder tools Docker image

1. SSH into the instance:

```
ssh -i INSTANCE_SSH_KEY_FILE ubuntu@INSTANCE_IP
```

2. Become root:

```
sudo su
```

3. Change to the tools directory:

```
cd /opt/imz-tools/intel-model-zoo/tools
```

4. List the models (this will trigger tools image to be built):

```
bash ./scripts/model-builder -f tensorflow images
```

Creating the benchmark image

1. SSH into the instance:

```
ssh -i INSTANCE_SSH_KEY_FILE ubuntu@INSTANCE_IP
```

2. Become root:

```
sudo su
```

3. Change to the tools directory:

```
cd /opt/imz-tools/intel-model-zoo/tools
```

4. Invoke the image builder using the MODEL_NAME from Table 12:

```
bash ./scripts/model-builder -f tensorflow make MODEL_NAME
```

Table 12: Model names for the image builder.

CPU vendor	Precision	MODEL_NAME
AMD	FP32	resnet50v1-5-fp32-inference
AMD	INT8	resnet50v1-5-int832-inference
Intel	FP32	resnet50v1-5-fp32-inference
Intel	INT8	resnet50v1-5-int832-inference

Starting NMON performance data collection

1. SSH into the instance:

```
ssh -i INSTANCE_SSH_KEY_FILE ubuntu@INSTANCE_IP
```

2. Become root:

```
sudo su
```

3. Start NMON:

```
nmon -F /tmp/output.nmon -s 1 -c 36000 -J -t -p > /tmp/nmon.pid
```

Invoking the benchmark

1. SSH into the instance:

```
ssh -i INSTANCE_SSH_KEY_FILE ubuntu@INSTANCE_IP
```

2. Become root:

```
sudo su
```

Note: Substitutions not listed in the table are tunable parameters, and vary by cpu/model/precision/batch-size. See **Tuning profiles** *for more information.*

Table 13: List of substitutions.

CPU vendor	Precision	MODEL_NAME MODEL_MOUNT DS_MOUNT DOCKER_IMAGE IN_GRAPH DATA_LOCATION EXTRA_COMMAND
Intel	FP32	resnet50v1_5 /persistent/models/resnet50v1_5 /persistent/datasets/resnet intel-model-zoo:latest-image-recognition-resnet50-fp32-inference /models/resnet50_v1.pb /datasets <no extra command>

CPU vendor	Precision	MODEL_NAME MODEL_MOUNT DS_MOUNT DOCKER_IMAGE IN_GRAPH DATA_LOCATION EXTRA_COMMAND
Intel	INT8	resnet50v1_5 /persistent/models/resnet50v1_5 /persistent/datasets/resnet intel-model-zoo:latest-image-recognition-resnet50-int8-inference /models/resnet50v1_5_int8_pretrained_model.pb /datasets <no extra command>
AMD	FP32	resnet50v1_5 /persistent/models/resnet50v1_5 /persistent/datasets/resnet amd-model-zoo:latest-amd-image-recognition-resnet50v1-5-fp32-inference /models/resnet50v1_5_int8_pretrained_model.pb /datasets . /.ZenDNNrc && \
AMD	INT8	resnet50v1_5 /persistent/models/resnet50v1_5 /persistent/datasets/resnet amd-model-zoo:latest-amd-image-recognition-resnet50v1-5-int8-inference /models/resnet50v1_5_int8_pretrained_model.pb /datasets . /.ZenDNNrc && \

3. Using the substitutions from Table 13, run the benchmark in a Docker container:

```
docker run \
--rm \
--privileged \
-v /opt/imz-tools/results:/output \
-v MODEL_MOUNT:/models \
-v DS_MOUNT:/datasets \
-v /opt/imz-tools/intel-model-zoo:/intel-model-zoo \
-w /intel-model-zoo \
DOCKER_IMAGE \
/bin/bash -c '\
  export DEBIAN_FRONTEND=noninteractive && \
  cd /intel-model-zoo && \
  apt-get update -y && \
  apt-get install -y numactl util-linux && \
  apt-get install -y google-perftools && \
  python3 -m pip install pandas ptutils pyyaml &&\
  export LD_PRELOAD="/usr/lib/x86_64-linux-gnu/libtcmalloc.so.4.3.0${LD_PRELOAD:+:${LD_
PRELOAD}}" && \
  export KMP_AFFINITY="verbose,granularity=fine,compact" && \
  EXTRA_COMMAND
  python3 /intel-model-zoo/benchmarks/launch_benchmark.py \
    --output-dir=/output/NAME_OF_TEST_RUN \
    --model-name MODEL_NAME\
    --precision PRECISION \
    --mode inference \
    --framework tensorflow \
    --benchmark-only \
    --batch-size BATCH_SIZE \
    --in-graph IN_GRAPH \
    --data-location DATA_LOCATION \
    --numa-cores-per-instance NUMA_CORES_PER_INSTANCE \
    --num-cores NUM_CORES \
    --num-intra-threads NUM_INTRA_THREADS \
    --num-inter-threads NUM_INTER_THREADS \
    --disable-tcmalloc=False \
    --num_omp_threads= NUM_OMP_THREADS \
    --warmup-steps=WARMUP_STEPS \
    --steps=STEPS \
'
```

Collecting NMON performance data

1. SSH into the instance:

```
ssh -i INSTANCE_SSH_KEY_FILE ubuntu@INSTANCE_IP
```

2. Become root:

```
sudo su
```

3. Stop NMON:

```
kill -s USR2 'cat /tmp/nmon.pid'
```

4. Change output file ownership:

```
chown ubuntu:ubuntu /tmp/output.nmon
```

5. Open a terminal on your local machine.
6. Copy the output file to your local machine:

```
rsync -e "ssh -I INSTANCE_SSH_KEY_FILE" -av ubuntu@INSTANCE_IP:/tmp/output.nmon ./output.nmon
```

Tuning profiles

Tests for each unique combination of CPU, precision, and batch-size used tuning parameters we define in Table 14.

Table 14: Tuning profiles for testing.

Profile	A	C	E	F	G	H	I	J
NUM_INSTANCES	1	1	1	1	1	1	1	2
NUM_INTRA_THREADS	4	8	2	4	4	4	4	4
NUM_INTER_THREADS	2	2	2	1	4	2	2	2
NUM_OMP_THREADS	4	4	4	4	4	4	4	4
NUMA_CORES_PER_INSTANCE	4	4	4	4	4	2	8	8
NUM_CORES	4	4	4	4	4	2	8	8

Table 15 identifies the tuning profile used for each benchmark run. We selected profiles by running the benchmarks with each profile and determining which profile provided the maximum throughput.

Table 15: Tuning profile for each benchmark run.

Instance type	vCPUs	Precision	Batch size	Profile selected
Standard_D16ds_v5	16	INT8	1	C
Standard_D16ds_v5	16	INT8	128	C
Standard_D16ads_v5	16	INT8	1	E
Standard_D16ads_v5	16	INT8	128	F

MySQL testing

We concluded our hands-on testing on February 23, 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 February 23, 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 16: Results of our testing.

	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors	Advantage
SPECint			
SPECrate 2017 Integer base rate median score (higher is better)	71.9	66.0	D16ds_v5 with Intel Xeon Scalable processors 8.3% higher
MySQL™			
MySQL median score (higher is better)	766,170	580,800	D16ds_v5 with Intel Xeon Scalable processors 31.9% more transactions per minute

System configuration information

Table 17: Detailed information on the VMs we tested.

System configuration information	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors
Tested by	Principled Technologies	Principled Technologies
Test date	2/23/2022	2/23/2022
CSP / Region	Microsoft Azure / East US	Microsoft Azure / East US
Workload & version	HammerDB 4.4	HammerDB 4.4
Workload-specific parameters	500 warehouses, 32 virtual users	500 warehouses, 32 virtual users
Iterations and result choice	3 runs, median	3 runs, median
Server platform	Standard_D16ds_v5	Standard_D16ads_v5
BIOS name and version	American Megatrends Inc. 090008 (12/07/2018)	American Megatrends Inc. 090008 (12/07/2018)
Operating system name and version/ build number	Red Hat Enterprise Linux 8.5 4.18.0-348.12.2.el8_5.x86_64	Red Hat Enterprise Linux 8.5 4.18.0-348.12.2.el8_5.x86_64
Date of last OS updates/patches applied	2/23/2022	2/23/2022

System configuration information	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors
Processor		
Number of processors	1	1
Vendor and model	Intel Xeon Platinum 8370C CPU @ 2.80GHz	AMD EPYC 7763 64-Core Processor
Core count (per processor)	32	64
Core frequency (GHz)	2.8	2.45
Family, model, stepping	6, 106, 6	25, 1, 2
SMT	Yes	Yes
Turbo	Yes (3.5 GHz)	Yes (3.5 GHz)
Number of vCPU per VM	16	16
Memory module(s)		
Total memory in system (GB)	64	64
NVMe memory present?	No	No
Total memory (DDR+NVMe RAM) (GB)	64	64
General HW		
Storage	Direct-attached / Instance	Direct-attached / Instance
Local storage		
OS		
Number of drives	1	1
Drive size (GB)	30	30
Drive information	StandardSSD_LRS	StandardSSD_LRS
Data drive		
Number of drives	1	1
Drive size (GB)	128	128
Drive information	Premium SSD P50	Premium SSD P50
Network adapter		
Vendor and model	Azure virtual network adapter version=5.13.0-1017-azure	Azure virtual network adapter version=5.13.0-1017-azure
Number and type of ports	1x 50Gb	1x 50Gb

How we tested

Creating the test and client instance

This section contains the steps we took to create the test instance hosting the MySQL database and the client instance for remotely running the HammerDB benchmark client software.

Creating the test instance

1. Log into the Azure Portal, and navigate to the Virtual Machines.
2. Click Create, and click Virtual Machine.
3. Enter the subscription and resource group names, and give the virtual machine a name.
4. Choose the geographical region for the instance. We used East US.
5. Select the RHEL 8.5 image.
6. Select the Standard_D16ds_v5 or Standard_D16ads_v5 instance size, depending on the test target.
7. Select SSH public key, and generate a new key pair for the desired username.
8. Select Allow selected ports, and ensure SSH is enabled.
9. Click Next: Disks.
10. Click Create and attach a new disk.
11. Click Change Size.
12. Select 128 GiB size and P50 performance tier.
13. Click OK twice.
14. Click Review + create.
15. Click Create.

Creating the HammerDB 4.4 client instance

1. Log into the Azure Portal, and navigate to the Virtual Machines.
2. Click Create, and click Virtual Machine.
3. Enter the subscription and resource group names, and give the virtual machine a name.
4. Choose the geographical region for the instance. We used East US.
5. Select the RHEL 8.5 image.
6. Select the Standard_D8ds_v5 instance size.
7. Select SSH public key and generate a new key pair for the desired username.
8. Select Allow selected ports and ensure SSH is enabled.
9. Click Review + create.
10. Click Create.

Configuring Red Hat Enterprise Linux 8 and install MySQL on test instance

1. Log into the MySQL instance via SSH.
2. Run the host preparation script from the Scripts section:

```
sudo /mysql_host_prepare.sh
```

3. Download the appropriate MySQL bundle for either x86 or ARM architecture on RHEL from https://dev.mysql.com/downloads/mysql/.
4. Extract the MySQL bundle:

```
tar -xf mysql-8.0.28-1.el8.x86_64.rpm-bundle.tar
```

5. Install MySQL Community Server 8 and all dependencies:

```
sudo yum --disablerepo=* localinstall mysql-community-server-8.0.28-1.el8.x86_64.rpm
```

6. Stop the MySQL service:

```
sudo service mysqld stop
```

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

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

8. Replace the MySQL CNF file with my-500.cnf from the Scripts section:

```
cp -p /etc/my.cnf{,.bak}
cp -f my-500.cnf /etc/my.cnf
```

9. Start the MySQL service:

```
sudo service mysqld start
```

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;
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]’;
```

12. Shut down the instance:

```
sudo poweroff
```

Configuring Red Hat Enterprise Linux 8 and installing HammerDB 4.4 on client instance

1. Log into the HammerDB instance via SSH.
2. Disable SELINUX:

```
sudo sed -I 's/SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config
sudo setenforce 0
```

3. Turn off SSH strict host key checking:

```
echo 'StrictHostKeyChecking no' > .ssh/config
chmod 400 ~/.ssh/config
```

4. Install required packages:

```
sudo dnf install -y epel-release
sudo dnf install -y wget vim tar zip unzip lz4 pigz nmon sysstat numactl ksh
```

5. Download the MySQL x86 RPM bundle from **https://dev.mysql.com/downloads/mysql/**.
6. Extract the MySQL bundle:

```
tar -xf mysql-8.0.28-1.el8.x86_64.rpm-bundle.tar
```

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

```
sudo yum --disablerepo=* localinstall mysql-community-client-8.0.26-1.el8.x86_64.rpm
```

8. Download HammerDB 4.4:

```
sudo wget https://github.com/TPC-Council/HammerDB/releases/download/v4.4/HammerDB-
4.4-Linux.tar.gz
```

9. Extract the HammerDB package:

```
tar -xf HammerDB-4.4-Linux.tar.gz
```

10. Download and extract the nmonchart tool:

```
wget https://sourceforge.net/projects/nmon/files/nmonchart40.tar
tar -xf nmonchart40.tar ./nmonchart
```

11. Shut down the instance:

```
sudo poweroff
```

Creating 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 500
diset tpcc mysql_partition true
diset tpcc mysql_num_vu 32
diset tpcc mysql_storage_engine innodb
```

5. Build the schema:

```
buildschema
```

Backing up the database

1. Log into the MySQL instance.
2. Shut down the database:

```
systemctl stop mysqld
```

3. Delete the log files:

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

4. Back up the database:

```
tar -cf- data/ | pigz -9 -c > mysql_tpcc500warehouses_data.tar.gz
```

Running the tests

In this section, we list the steps to run the HammerDB TPROC-C test on the VMs under test.

1. Log into the HammerDB MySQL client instance via SSH.
2. Navigate to the HammerDB directory:

```
cd HammerDB-4.4
```

3. Run the test using the TCL file from the Scripts section:

```
./hammerdbcli auto 500-test.tcl
```

4. Once the test is finished, stop the mysqld service, untar the backup file to reset the database, and restart the mysqld service.
5. Reboot the MySQL instance and client instance.

Scripts

In this section, we show the scripts we used in the steps above.

mysql_host_prepare.sh

```
#!/bin/bash
setenforce 0
sed -i 's/SELINUX=.*/SELINUX=Permissive/' /etc/selinux/config
systemctl disable --now firewalld
#### System tuning ####
tuned-adm profile virtual-guest

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 ####
yum -y install wget vim tar zip unzip lz4 pigz nmon sysstat numactl ksh
#### Prepare storage ####
umount -v /mnt/mysqldata
mkdir -p /mnt/mysqldata
sed -i '/mysqldata/d' /etc/fstab
if [ -e /dev/sdc ]; then
  mkfs.xfs -f /dev/sdc
  echo '/dev/sdc  /mnt/mysqldata  xfs  defaults,nofail,x-systemd.device-timeout=5  0 2' >> /etc/fstab
else
  echo 'NO /dev/sdc!'
fi
mount -v /mnt/mysqldata
restorecon -Rv /mnt/mysqldata
```

my-500.cnf

```
[mysqld]
datadir=/mnt/mysqldata/mysql
default_authentication_plugin=mysql_native_password
socket=/mnt/mysqldata/mysql/mysqld.sock
log-error=/var/log/mysql/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
port=3306
bind_address=0.0.0.0
# 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=16 #scale
innodb_open_files=4000
# buffers
innodb_buffer_pool_size=48000M #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=2000
innodb_io_capacity_max=4000
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='%'
[client]
socket=/mnt/mysqldata/mysql/mysqld.sock
```

500-test.tcl

```
dbset db mysql
diset connection mysql_host <IP_ADDRESS>
diset tpcc mysql_user mysql
diset tpcc mysql_pass <Password>
diset tpcc mysql_count_ware 500
diset tpcc mysql_partition true
diset tpcc mysql_num_vu 32
diset tpcc mysql_storage_engine innodb
diset tpcc rampup 2
diset tpcc duration 5
vucreate
vurun
```

Login Enterprise testing

We concluded our hands-on testing on February 15, 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 February 8, 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 18: Results of our testing.

	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors	Advantage
SPECint			
SPECrate 2017 Integer base rate median score (higher is better)	71.9	66.0	D16ds_v5 with Intel Xeon Scalable processors 8.3% higher
Login Enterprise			
Login Enterprise relative median score (higher is better)	1.14	1	D16ds_v5 with Intel Xeon Scalable processors 14.0% faster at 36 users

System configuration information

Table 19: Detailed information on the VMs we tested.

System configuration information	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors
Tested by	Principled Technologies	Principled Technologies
Test date	2/15/2022	2/15/2022
CSP / Region	Microsoft Azure / East US	Microsoft Azure / East US
Workload & version	Login Enterprise 4.8.4	Login Enterprise 4.8.4
Workload-specific parameters	36 VDI users	36 VDI users
Iterations and result choice	3 runs, median	3 runs, median
Server platform	Standard_D16ads_v5	Standard_D16ads_v5
BIOS name and version	American Megatrends Inc. 090008 (12/07/2018)	American Megatrends Inc. 090008 (12/07/2018)
Operating system name and version/ build number	Windows 10 Enterprise 21H1	Windows 10 Enterprise 21H1

System configuration information	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors
Date of last OS updates/patches applied	2/01/2022	2/01/2022
Processor		
Number of processors	1	1
Vendor and model	Intel Xeon Platinum 8370C CPU @ 2.80GHz	AMD EPYC 7763 64-Core Processor
Core count (per processor)	32	64
Core frequency (GHz)	2.8	2.45
Family, model, stepping	6, 106, 6	25, 1, 2
SMT	Yes	Yes
Turbo	Yes (3.5 GHz)	Yes (3.5 GHz)
Number of vCPU per VM	16	16
Memory module(s)		
Total memory in system (GB)	64	64
NVMe memory present?	No	No
Total memory (DDR+NVMe RAM) (GB)	64	64
General HW		
Storage: NW or Direct Att / Instance	Direct-attached / Instance	Direct-attached / Instance
Local storage		
OS		
Number of drives	1	1
Drive size (GB)	32	32
Drive information	Premium SSD	Premium SSD
Data drive		
Number of drives	1	1
Drive size (GB)	600	600
Network adapter		
Vendor and model	Azure virtual network adapter version=5.13.0-1017-azure	Azure virtual network adapter version=5.13.0-1017-azure
Number and type of ports	1x 50Gb	1x 50Gb

How we tested

The virtual desk infrastructure (VDI) we hosted on Azure included one domain controller VM, four Login Enterprise launcher VMs, one Login Enterprise virtual appliance, one jump box VM, and two instances under test (one Standard_D16ds_v5 instance featuring 3rd Gen Intel Xeon Scalable processors and one Standard_D16ads_v5 instance featuring AMD EPYC processors). The domain controller services were synchronized with Azure Active Directory (AD) to provide authentication and facilitate the deployment and access to Azure Virtual Desktop services. We also used FSLogix profiles containers stored on Azure file shares to host the user profile data necessary to provide VDI user data persistency during testing. We created test pools using the same Windows 10 Enterprise multi-user Azure image template, which was optimized for VDI and had Microsoft Office 365 apps installed with all the latest patches and updates available at the time of testing. We performed all testing with a custom Login Enterprise Knowledge worker workload using Microsoft Word, PowerPoint, Excel, Outlook, Edge, and Teams. For more information on how to configure FSLogix for Azure Virtual Desktop services, visit: **https://docs.microsoft.com/en-us/azure/virtual-desktop/fslogix-containers-azure-files**.

Deploying and configuring the cloud environment

Deploying Azure gold image

1. Sign into the Azure control panel.
2. Create an Azure Active Directory for authentication.
3. Click +Create a Resource, and select Azure Active Directory.
4. Click Create.
5. Enter the Organization name and Initial domain name.
6. Create a gold image with Windows 10 Enterprise multi-session.
7. Click +Create a resource, search for Microsoft Windows 10, and select the Microsoft-issued resource.
8. For plan, select Windows 10 Enterprise multi-session, version 21H2.
9. Select the subscription and resource group to use for the environment.
10. Enter a name for the gold VM image.
11. Select the region and size for the VM.
12. Enter credentials for the Administrator account, and adjust inbound port rules if necessary.
13. Click to confirm the Licensing validation.
14. Click Next: Disks >.
15. Select OS disk type: Premium SSD.
16. Click Next: Networking >.
17. Select the Virtual network for the VM.
18. Click Review + create.

Configuring the Windows gold image VM

Installing the Microsoft Edge driver for the gold image VM

1. Find the Microsoft Edge driver matching your current version of Edge at **https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/**.
2. Connect to the Login Enterprise appliance via SSH.
3. Run the following command:

```
chmod 707 /loginvsi/content/selenium
```

4. Create a folder named EdgeChromium[Major Version #] inside /loginvsi/content/selenium.
5. Extract msedgedriver.exe from the downloaded zip file, and copy it to the created folder.

Installing Office 365 on the gold image VM

1. Install the Microsoft Office Professional Plus 2019 Volume using the offline installer, and install a pre-generated XML configuration file to install Word, Excel, PowerPoint, Outlook and Teams only.
2. Open Windows Update, and click Advanced options.
3. Enable Receive updates for other Microsoft products, and click Back.
4. Click Check for updates, and reboot if necessary.

Creating a session host and AVD test pool

1. Sign into the Azure control panel.
2. Click Home.
3. Click Host Pools, and click Create.
4. Select subscription and resource group.
5. For location, select East US.
6. For host pool type, select Pooled.
7. For load balancing, select Breadth-first.
8. For Max session limit, enter `120`.
9. Click Next: Virtual Machine.
10. Enter a name for session hosts.
11. Select no infrastructure redundancy required.
12. Under Image type, select Gallery.
13. Under Image, select See all images, and browse to gold image.
14. For Virtual machine size, select Standard_D16ds_v5 or Standard_D16ads_v5 (16 vCPU, 64 GiB memory).
15. For number of VMs, enter `1`.
16. For OS disk type, select Premium SSD.
17. Select the appropriate virtual network.
18. For public inbound ports, click No.
19. Enter Active Directory credentials.
20. Enter Virtual Machine Administrator account credentials
21. Click Next: Workspace.
22. To register the desktop app group, select Yes.
23. Select the appropriate workspace.
24. Click Create.

Configuring Login Enterprise

Configuring Login Component for Login Enterprise

1. In the Login Enterprise appliance web interface, click Accounts.
2. Scroll to the bottom of the page, and click to download the Login Executable.
3. Extract LoginPI.Login.exe from the zip, and copy it to the sysvol share on the domain controller.
4. Edit the properties on the user group for the test users, and type the following into the login script field:

```
LoginPI.Login.exe <URL of the LoginVSI appliance>
```

5. In the Virtual User Accounts pane, click the +, and click Bulk Accounts.
6. Enter the base username, password, domain, number of digits for the user sequence, and number of accounts to be created. Click Save.
7. In the Windows Active Directory Users and Computers control panel, create a security group called VDI Users.
8. Create an OU in Active Directory named VDI Users, and create accounts to match those in Login Enterprise.
9. Open the VDI Users security group, and add the newly created users to the group.

Creating application test profiles

1. In the Login Enterprise appliance web interface, click Applications.
2. Click Add new application.
3. Enter the Application name, type, and target. Click Save.
4. In the application list, click the pencil to edit the newly created application profile.
5. Click Upload Script, and upload the script for each application. Click Save.
6. Complete steps 1 through 5 for each application under test.

Configuring test parameters

1. In the Login Enterprise appliance web interface, click Manage Tests.
2. Click Create a new Load Test.
3. Enter a name for the test, and select Custom.
4. Enter the RDS URL and Resource to be used.

5. In the Accounts field, select All users.
6. In the Launchers field, select All Launchers.
7. Click Save.
8. Enter the number of users you wish to test.
9. Set the test duration to 60 minutes.
10. In the Actions pane, click Add action(s), and click the Application(s) option.
11. Select all the applications you wish to test. Click Save.

Running a test

1. Click the Play icon next to the test you wish to run.
2. Validate the test parameters, and click Confirm.

WordPress testing

We concluded our hands-on testing on March 14, 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 March 14, 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 20: Results of our testing.

	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors	Advantage
SPECint			
SPECrate 2017 Integer base rate median score (higher is better)	71.9	66.0	D16ds_v5 with Intel Xeon Scalable processors 8.3% higher
WordPress			
WordPress median score (higher is better)	805.6	725.3	D16ds_v5 with Intel Xeon Scalable processors 11.0% more requests per second

System configuration information

Table 21: Detailed information on the VMs we tested.

System configuration information	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors
Tested by	Principled Technologies	Principled Technologies
Test date	3/14/2022	3/14/2022
CSP / Region	Microsoft Azure / East US	Microsoft Azure / East US
Workload & version	oss-performance v2019.02.13.00	oss-performance v2019.02.13.00
Workload-specific parameters	200 threads, remote siege	200 threads, remote siege
Iterations and result choice	3 runs, median	3 runs, median
Server platform	Standard_D16ds_v5	Standard_D16ads_v5
BIOS name and version	American Megatrends Inc. 090008 (12/07/2018)	American Megatrends Inc. 090008 (12/07/2018)
Operating system name and version/ build number	Bitnami Wordpress for Intel Gen 2 (Debian) 5.9.3-debian-10-r47	Ubuntu 20.04 5.13.0-1017-azure
Date of last OS updates/patches applied	3/14/2022	3/14/2022

System configuration information	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors
Processor		
Number of processors	1	1
Vendor and model	Intel Xeon Platinum 8370C CPU @ 2.80GHz	AMD EPYC 7763 64-Core Processor
Core count (per processor)	32	64
Core frequency (GHz)	2.8	2.45
Family, model, stepping	6, 106, 6	25, 1, 2
SMT	Yes	Yes
Turbo	Yes (3.5 GHz)	Yes (3.5 GHz)
Number of vCPU per VM	16	16
Memory module(s)		
Total memory in system (GB)	64	64
NVMe memory present?	No	No
Total memory (DDR+NVMe RAM) (GB)	64	64
General HW		
Storage	Direct-attached / Instance	Direct-attached / Instance
Local storage		
OS		
Number of drives	1	1
Drive size (GB)	30	30
Drive information	StandardSSD_LRS	StandardSSD_LRS
Network adapter		
Vendor and model	Azure virtual network adapter version=5.13.0-1017-azure	Azure virtual network adapter version=5.13.0-1017-azure
Number and type of ports	1x 50Gb	1x 50Gb

How we tested

Testing overview

We created our Intel Xeon Scalable processor-powered WordPress VMs using the image from the Azure image library called "WordPress for Intel packaged by Bitnami - Gen2." This image includes NGINX, PHP, MariaDB, and Wordpress pre-installed along with the latest Intel QuickAssist Technology enhancements for increased SSL performance on the latest Intel v5 instances. We created our AMD EPYC processor-powered WordPress VMs on Ubuntu 20.04, as there is no Bitnami image optimized specifically for AMD. We used the same sized database and workload for both VMs. The database is small and easily fits in RAM, so we didn't need to worry about fluctuations in disk performance, though we used SSD storage for all instances just to ensure there was no delay loading the database into memory.

Creating the client VM using Ubuntu Server 20.04 LTS

This section contains the steps we took to create our client instance.

Creating the client VM

1. Log into the Azure Portal, and navigate to the Virtual Machines service.
2. Click Add to open the Add VM wizard.
3. On the Basics tab, set the following:
    a. Choose your Subscription from the drop-down menu.
    b. Choose your Resource group from the drop-down menu.
    c. Name the Virtual Machine.
    d. Choose your Region from the drop-down menu.
    e. Leave the Availability options set to No infrastructure redundancy required.
    f. Click Browse all public and private images
    g. In the Search field, enter Ubuntu Server 20.04 LTS
    h. Select Ubuntu Server 20.04 LTS - Gen2 from the list of results.
    i. Leave Azure Spot instance set to No.
    j. Select the instance size you wish to use. We used D8ds_v4.
    k. Leave the Authentication type set to SSH public key.
    l. Either choose a new Username or leave the default.
    m. Choose Generate new key pair for the SSH public key source.
    n. Enter a name for the Key pair name.
    o. Leave Public inbound ports set to Allow selected ports.
    p. For Select inbound ports, choose SSH (22).
4. On the Disks tab, set the following:
    a. For the OS disk type, choose Standard SSD from the drop-down menu.
    b. Leave the default Encryption type.
5. On the Networking tab, set the following:
    a. Choose your Virtual network from the drop-down menu.
    b. Choose Create new to create a new Public IP.
    c. Leave the rest of the settings at defaults.
6. On the Management tab, set the following:
    a. Choose your Diagnostics storage account from the drop-down menu.
    b. Leave the rest set to defaults.
7. On the Advanced tab, leave all defaults.
8. On the Tags tab, add any tags you wish to use.
9. On the Review + create tab, review your settings, and click Create.

Configuring Ubuntu Server 20.04 LTS

1. Log in as the azureuser user using the SSH key generated during Azure instance creation. For example:

```
ssh -i wordpress_key.pem azureuser@<INSTANCE_PUBLIC_IP_ADDRESS>
```

2. Install the latest update packages, and reboot the VM:

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

3. Set the time zone on the VM:

```
sudo timedatectl set-timezone America/New_York
```

4. Install additional tools:

```
sudo apt-get install -y nmon ksh numactl virt-what jq spectre-meltdown-checker siege git
wget sysstat lshw
```

Downloading benchmark client files and prerequisites

1. Download latest siege client URL file:

```
wget https://raw.githubusercontent.com/intel/Updates-for-OSS-Performance/main/targets/wordpress/
WordpressTarget_v5.urls
```

2. Install nmonchart as a prerequisite to running the benchmark scripts:

```
wget https://raw.githubusercontent.com/aguther/nmonchart/master/nmonchart
chmod +x nmonchart
```

Creating the VM under test

In this section we list the steps required to create a VM using the Bitnami or Ubuntu image. For our testing, we used the East US Region and Availability Zone 1.

Creating the VM

1. Log into the Azure Portal, and navigate to the Virtual Machines service.
2. Click Add to open the Add VM wizard.
3. On the Basics tab, set the following:
    a. Choose your Subscription from the drop-down menu.
    b. Choose your Resource group from the drop-down menu.
    c. Name the Virtual Machine.
    d. Choose your Region from the drop-down menu.
    e. Leave the Availability options set to No infrastructure redundancy required.
    f. Click Browse all public and private images.
    g. In the Search field, enter `WordPress for Intel`.
    h. For the Intel Xeon Scalable processor-powered VM, select WordPress for Intel packaged by Bitnami - Gen2 from the list of results. For the AMD EPYC processor-powered VM, select Ubuntu 20.04.
    i. Leave Azure Spot instance set to No.
    j. Select D16ds_v5 or D16ads_v5 depending on the target instance type.
    k. Leave the Authentication type set to SSH public key.
    l. Either choose a new Username or leave the default.

   m. Choose Generate new key pair for the SSH public key source.
   n. Enter a name for the Key pair name.
   o. Leave Public inbound ports set to Allow selected ports.
   p. For Select inbound ports, choose SSH (22).

4. On the Disks tab, set the following:
   a. For the OS disk type, choose Standard SSD from the drop-down menu.
   b. Leave the default Encryption type.
5. On the Networking tab, set the following:
   a. Choose your Virtual network from the drop-down menu.
   b. Choose Create new to create a new Public IP.
   c. Leave the rest of the settings at defaults.
6. On the Management tab, set the following:
   a. Choose your Diagnostics storage account from the drop-down menu.
   b. Leave the rest set to defaults.
7. On the Advanced tab, leave all defaults.
8. On the Tags tab, add any tags you wish to use.
9. On the Review + create tab, review your settings, and click Create.

Configuring the WordPress VMs

1. Log in as the azureuser user using the SSH key generated during Azure instance creation. For example:

```
ssh -i wordpress_key.pem azureuser@<INSTANCE_PUBLIC_IP_ADDRESS>
```

2. Install the latest update packages, and reboot the VM:

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

3. Set the time zone on the VM:

```
sudo timedatectl set-timezone America/New_York
```

4. Install additional tools:

```
sudo apt-get install -y nmon ksh numactl virt-what jq spectre-meltdown-checker siege git wget
sysstat curl lsb-release lshw
```

Loading the WordPress Database

1. Stop all WordPress services, and restart MariaDB:

```
sudo /opt/bitnami/ctlscript.sh stop
sudo /opt/bitnami/ctlscript.sh start mariadb
```

2. Download and extract the WordPress database backup for the benchmark:

```
wget https://github.com/intel/Updates-for-OSS-Performance/raw/main/targets/wordpress/
dbdump_v5.sql.gz
gunzip dbdump_v5.sql
```

3. Locate the generated MariaDB password from the image:

```
sudo cat /home/bitnami/bitnami_credentials
```

4. Load the WordPress becnhmark database backup into MariaDB using the password obtained in the previous step:

```
cat dbdump_v5.sql | mysql -u root --password=<MARIADB_PASSWORD> bitnami_wordpress
```

5. Restart all WordPress services:

```
sudo /opt/bitnami/ctlscript.sh restart
```

Generating and exchanging SSH keys

1. On the WordPress client VM, run the following command to create a new SSH key pair:

```
ssh-keygen
```

2. Press Enter four times to save the key pair to the default location with no password.
3. Once the key pair has been generated, run the following command to obtain your new public SSH key:

```
cat .ssh/id_rsa.pub
```

4. Copy the output key to your clipboard.
5. Log into the VM under test.
6. Open the authorized_key file with the following command, and copy the public SSH key from the VM under test into the file:

```
sudo vim .ssh/authorized_keys
```

7. Repeat for the WordPress client VM, copying the public SSH key to the authorized_keys file.

Running the tests

In this section, we list the steps to run the oss-performance benchmark on the VMs under test. A script starts the benchmark from the client VM and automates the entire process.

1. Run the benchmark script, substituting the host name of the VM under test:

```
./run_test.sh <HOSTNAME_OF_VM_UNDER_TEST>
```

2. Results are automatically saved in the results subdirectory of the client VM.

Scripts

In this section, we show the scripts we used in our testing.

run_test.sh

```
#!/bin/bash
RUNTIME=300
STEP=5
COUNT=$((RUNTIME/STEP))
TIMESTAMP=$(date '+%Y%m%d_%H%M%S')
CLIENT_THREADS=200
TEST_HOST=${1}
REMOTE_SIEGE=${2:-$(hostname -s)}
echo -n "siege host: "
ssh ${TEST_HOST} "ssh ${REMOTE_SIEGE} 'echo ${REMOTE_SIEGE}'"
sleep 1
scp -p 50-server.cnf ${TEST_HOST}:
ssh ${TEST_HOST} "sudo mv -f 50-server.cnf /etc/mysql/mariadb.conf.d/50-server.cnf ; sync ; sudo
systemctl restart mysqld"
```

```
RESULTS_DIR=results/${REMOTE_SIEGE}_${TEST_HOST}_${TIMESTAMP}
mkdir -p ${RESULTS_DIR}
RESULTS_FILE=${TEST_HOST}_${TIMESTAMP}
killall -w nmon
sleep 1
if [ "${REMOTE_SIEGE}" == "$(hostname -s)" ]; then
  nmon -F ${RESULTS_DIR}/wordpress-client_${TIMESTAMP}.nmon -s${STEP} -J -t -C nginx:mysqld:php:siege
:nmon:systemd-resolve
fi

ssh ${TEST_HOST} "cd oss-performance ; php perf.php \
  --wordpress \
  --php=/usr/sbin/php-fpm7.4 \
  --remote-siege=testuser@${REMOTE_SIEGE} \
  --i-am-not-benchmarking \
  --client-threads=${CLIENT_THREADS} \
  --benchmark-time=${RUNTIME} \
  --exec-after-warmup='nmon -F /tmp/${TEST_HOST}.nmon -s${STEP} -c${COUNT} -J -t -C nginx:mysqld:php
:siege:nmon:systemd-resolve' \
  --trace 2>&1" | tee ${RESULTS_DIR}/${RESULTS_FILE}.txt
killall -w nmon
scp ${TEST_HOST}:/tmp/${TEST_HOST}.nmon ${RESULTS_DIR}/${RESULTS_FILE}.nmon
ssh ${TEST_HOST} "pkill nmon ; sleep 1 ; rm -f /tmp/${TEST_HOST}.nmon"
for nmonfile in 'find ${RESULTS_DIR}/*.nmon';
do
  ./nmonchart $nmonfile
done
cp -pf ${0} ${RESULTS_DIR}/
```

50-server.cnf

```
[server]
[mysqld]
user                    = mysql
pid-file               = /run/mysqld/mysqld.pid
socket                  = /run/mysqld/mysqld.sock
#port                   = 3306
basedir                 = /usr
datadir                 = /var/lib/mysql
tmpdir                  = /tmp
lc-messages-dir         = /usr/share/mysql
bind-address            = 127.0.0.1
max_connections         = 2000
query_cache_size        = 0
query_cache_type        = 0

log_error = /var/log/mysql/error.log
expire_logs_days        = 10
character-set-server  = utf8mb4
collation-server      = utf8mb4_general_ci
[embedded]
[mariadb]
[mariadb-10.3]
```

HiBench testing

We concluded our hands-on testing on March 8, 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 March 8, 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 22: Results of our testing.

	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors	Advantage
SPECint			
SPECrate 2017 Integer base rate median score (higher is better)	71.9	66.0	D16ds_v5 with Intel Xeon Scalable processors 8.3% higher
HiBench k-means			
HiBench k-means median score (higher is better)	207.3	198.5	D16ds_v5 with Intel Xeon Scalable processors 4.43% greater throughput (MB/s)

System configuration information

Table 23: Detailed information on the VMs we tested.

System configuration information	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors
Tested by	Principled Technologies	Principled Technologies
Test date	3/8/2022	3/8/2022
CSP / Region	Microsoft Azure / East US	Microsoft Azure / East US
Workload & version	HiBench K-means 7.1.1	HiBench K-means 7.1.1
Workload-specific parameters	spark.executor.memory=52g	spark.executor.memory=52g
Iterations and result choice	3 runs, median	3 runs, median
Server platform	Standard_D16ds_v5	Standard_D16ads_v5
BIOS name and version	American Megatrends Inc. 090008 (12/07/2018)	American Megatrends Inc. 090008 (12/07/2018)
Operating system name and version/ build number	RHEL 8.5 4.18.0-348.12.2.el8_5.x86_64	RHEL 8.5 4.18.0-348.12.2.el8_5.x86_64
Date of last OS updates/patches applied	3/8/2022	3/8/2022

System configuration information	D16ds_v5 with Intel Xeon Scalable processors	D16ads_v5 with AMD EPYC processors
Processor		
Number of processors	1	1
Vendor and model	Intel Xeon Platinum 8370C CPU @ 2.80GHz	AMD EPYC 7763 64-Core Processor
Core count (per processor)	32	64
Core frequency (GHz)	2.8	2.45
Family, model, stepping	6, 106, 6	25, 1, 2
SMT	Yes	Yes
Turbo	Yes (3.5 GHz)	Yes (3.5 GHz)
Number of vCPU per VM	16	16
Memory module(s)		
Total memory in system (GB)	64	64
NVMe memory present?	No	No
Total memory (DDR+NVMe RAM)	64	64
General HW		
Storage	Direct-attached / Instance	Direct-attached / Instance
Local storage		
OS		
Number of drives	1	1
Drive size (GB)	30	30
Drive information	StandardSSD_LRS	StandardSSD_LRS
Data drive		
Number of drives	1	1
Drive size (GB)	500	500
Drive information	Ultra_SSD (20k IOPS/900 MBps)	Ultra_SSD (20k IOPS/900 MBps)
Network adapter		
Vendor and model	Azure virtual network adapter version=5.13.0-1017-azure	Azure virtual network adapter version=5.13.0-1017-azure
Number and type of ports	1x 50Gb	1x 50Gb

How we tested

Testing overview

For this project, we tested Azure VMs featuring Intel Xeon Scalable processors vs. Azure VMs featuring AMD EPYC processors. We ran the k-means test from the HiBench suite to show a performance increase in terms of total throughput. Our results reflect the performance customers might expect to see when using each VM series.

Using our methodology to aid your own deployments

While the methodology below describes in great detail how we accomplished our testing, it is not a deployment guide. However, because we include many basic installation steps for operating systems and testing tools, reading our methodology may help with your own installation.

Creating the namenode and datanode VMs

Creating the test instance

1. Log into the Azure Portal, and navigate to the Virtual Machines.
2. Click Create, and click Virtual Machine.
3. Enter the subscription and resource group names, and give the virtual machine a name.
4. Choose the geographical region for the VM. We used East US.
5. Select the RHEL 8.5 image.
6. Select the Standard_D16ds_v5 or Standard_D16ads_v5 instance size, depending on the test target.
7. Under Availability Options, select Availability Zone.
8. Select Zone 1.
9. Select SSH public key, and generate a new key pair for the desired username.
10. Select Allow selected ports and ensure SSH is enabled.
11. Click Review + create.
12. Click Create.
13. For each datanode, when the VM is ready, go to the VM page, and select Disks from the left-hand side.
14. Click Additional Settings.
15. Under Enable Ultra disk compatibility, select Yes.
16. Click Save.
17. Click Create and attach a new disk.
18. Select 512 GB and UltraSSD, with 20,000 IOPS and 900 MBps.
19. Click Save.

Configuring RHEL 8 and installing Apache Hadoop and Spark

1. Via SSH, log into the namenode instance.
2. Log into your VM.
3. Change the root password:

```
sudo passwd root
```

4. Switch to the root user:

```
su -
```

5. Modify SSH to allow a pre-shared key login:

```
mkdir -p /root/.ssh
chmod 700 /root/.ssh
cd /root/.ssh
ssh-keygen -t rsa -q
cp id_rsa.pub authorized_keys
echo "StrictHostKeyChecking=no" > config
```

6. Set the hostname:

```
hostnamectl set-hostname [HOSTNAME]
```

7. To add your hostname to your IP address, modify your hosts file.
8. Turn off and disable your firewall:

```
systemctl stop firewalld
systemctl disable firewalld
```

9. Edit your SELinux to disable its enforcing:

```
setenforce 0
vi /etc/selinux/config (modify "enforcing" to "disabled" in the file)
```

10. Update your OS:

```
yum upgrade -y
```

11. Install the prerequisites via YUM:

```
yum install -y mdadm vim tar wget java-1.8.0-openjdk maven git blas64 lapack64 python2 bc
```

12. Download Apache® Hadoop® and Apache Spark®:

```
wget http://www.gtlib.gatech.edu/pub/apache/spark/spark-3.2.1/spark-3.2.1-bin-hadoop3.3.tgz wget
http://www.gtlib.gatech.edu/pub/apache/hadoop/common/hadoop-3.3.1/hadoop-3.3.1.tar.gz
```

13. Modify your bash profile, and add the following lines:

```
JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.265.b01-0.el8_2.x86_64/jre PATH=$PATH:$HOME/bin:/
opt/yarn/hadoop-3.3.1/bin
```

14. Reboot your system.
15. Add in the Hadoop users:

```
groupadd hadoop
useradd -g hadoop yarn
useradd -g hadoop hdfs
useradd -g hadoop mapred
```

16. Extract the Hadoop and Spark compressed files:

```
cd /opt/yarn
tar xvzf /root/hadoop-3.3.1.tar.gz
tar -xvzf ~/spark-3.2.1-bin-hadoop3.3.tgz
```

17. Move into the Hadoop directory, and make a yarn directory:

```
cd hadoop-3.3.1/
mkdir logs
chmod g+w logs
chown yarn:hadoop . -R
```

18. Navigate into the Hadoop configuration directory:

```
cd etc/hadoop/
```

19. Modify the Hadoop configuration files with the following settings:

core-site.xml

```
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://[MANAGER IP ADDRESS]:9000</value>
</property>
<property>
<name>hadoop.http.staticuser.user</name>
<value>hdfs</value>
</property>
</configuration>
```

hdfs-site.xml

```
<configuration>
<property>
<name>dfs.replication</name>
<value>3</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:/var/data/hadoop/hdfs/nn</value>
</property>
<property>
<name>fs.checkpoint.dir</name>
<value>file:/var/data/hadoop/hdfs/snn</value>
</property>
<property>
<name>fs.checkpoint.edits.dir</name>
<value>file:/var/data/hadoop/hdfs/snn</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file:/var/data/hadoop/hdfs/dn</value>
</property>
</configuration>
```

mapred-site.xml

```
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
<property>
<name>yarn.app.mapreduce.am.env</name>
<value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
</property>
<property>
<name>mapreduce.map.env</name>
<value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
</property>
<property>
<name>mapreduce.reduce.env</name>
<value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
</property>
</configuration>
```

yarn-site.xml

```
<configuration>
<property>
<name>yarn.resourcemanager.hostname</name>
<value>[MANAGER HOSTNAME HERE]</value>
</property>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>
```

20. Uncomment the JAVA_HOME line in hadoop-env.sh, and add the following information:

```
JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.265.b01-0.el8_2.x86_64/jre
```

21. Power off the instance:

```
poweroff
```

Configuring and starting the cluster

1. To set the hostname on the namenode and each of the datanode instances, edit /etc/hostname.
2. Add the FQDN, hostname, and IP address of each VM to the /etc/hosts file on the manager and datanode instances.
3. Verify that you can SSH into each instance without a password.
4. On the datanode instances, create an XFS file system on the data and temp drive:

```
mkfs.xfs /dev/<DATA DRIVE> && mkfs.xfs /dev/<TEMP DRIVE>
```

5. On the datanode instances, remove the Hadoop data subdirectories:

```
rm -rf /var/data/hadoop/*
```

6. On the datanode instances, mount the data drive:

```
mount /dev/<DATA DRIVE> /var/data/hadoop
```

7. On the datanode instances, create the Hadoop subdirectories for the namenode, secondary namenode, and datanode data, and set the permissions:

```
mkdir -p /var/data/hadoop/hdfs/nn
mkdir -p /var/data/hadoop/hdfs/snn
mkdir -p /var/data/hadoop/hdfs/dn
chown hdfs:hadoop /var/data/hadoop/hdfs/ -R
```

8. On each of the instances, format the hdfs file system, and create the temp location:

```
hdfs namenode -format
mkdir -p /var/data/hadoop/hdfs/tmp
```

9. On the datanode instances, mount the temp drive:

```
mount /dev/<TEMP DRIVE> /var/data/hadoop/hdfs/tmp
```

10. Append the following line to /opt/yarn/spark-3.2.1-bin-hadoop3.3/conf/spark-env.sh on each instance:

```
export SPARK_LOCAL_DIRS=/var/data/hadoop/hdfs/tmp
```

11. Start the Hadoop services and Spark on the namenode:

```
/opt/yarn/hadoop-3.3.1/bin/hdfs --daemon start namenode
/opt/yarn/hadoop-3.3.1/bin/hdfs --daemon start secondarynamenode
/opt/yarn/hadoop-3.3.1/bin/yarn --daemon start resourcemanager
/opt/yarn/hadoop-3.3.1/bin/yarn --daemon start nodemanager
/opt/yarn/spark-3.2.1-bin-hadoop3.3/sbin/start-manager.sh
```

12. Start the Hadoop services and Spark on each of the datanodes:

```
/opt/yarn/hadoop-3.3.1/bin/hdfs --daemon start datanode
/opt/yarn/spark-3.2.1-bin-hadoop3.3/sbin/start-worker.sh spark://[NAMENODE IP ADDRESS]:7077
```

Installing and configuring HiBench

Perform the following steps on the namenode to install and configure HiBench.

1. Create the directories you will use for HiBench:

```
hdfs dfs -mkdir -p /user/root
hdfs dfs -mkdir /HiBench
hdfs dfs -chown -R root:hadoop /HiBench
hdfs dfs -chown root /user/root
```

2. Navigate to your home directory, and download HiBench:

```
cd ~
git clone https://github.com/intel-hadoop/HiBench.git
```

3. Install HiBench for Apache Spark 3.2:

```
cd HiBench/
mvn -Dspark.version=3.2.1 -Dscala=2.12 clean package | tee hibench_build.log cd conf/
```

4. Modify the HiBench configuration files with the following information:

hadoop.conf

```
# Hadoop home
hibench.hadoop.home     /opt/yarn/hadoop-3.3.1
# The path of hadoop executable
hibench.hadoop.executable       ${hibench.hadoop.home}/bin/hadoop
# Hadoop configraution directory
hibench.hadoop.configure.dir ${hibench.hadoop.home}/etc/hadoop
# The root HDFS path to store HiBench data
hibench.hdfs.master     hdfs://[MANAGER IP ADDRESS]:9000
# Hadoop release provider. Supported value: apache, cdh5, hdp
hibench.hadoop.release   apache
```

spark.conf

```
# Spark home
hibench.spark.home       /opt/yarn/spark-3.2.1-bin-hadoop3.3/
# Spark manager
# standalone mode: spark://xxx:7077 #   YARN mode: yarn-client
hibench.spark.master spark://[MANAGER IP ADDRESS]:7077
```

Running the tests

In this section, we list the steps to run the k-means benchmark on the VMs under test. The benchmark is started from the namenode using a script that automates the entire process.

1. Log into the namenode via SSH.
2. Create a results directory:

```
mkdir ~/results
```

3. Navigate to the directory with your scripts:

```
cd ~/scripts
```

4. Run the benchmark script.

```
./run_test.sh
```

Test scripts

In this section, we show the scripts we used in the steps above.

run_test.sh

```
#!/bin/bash results_dir=~/results
mkdir -p ${results_dir}/${vcpu_count}
mkdir -p ${results_dir}/${vcpu_count}/kmeans
timestamp=$(date '+%Y%m%d_%H%M%S') kmeans_results=${results_dir}/${vcpu_count}/kmeans
#Start Apache Hadoop & Spark
~/scripts/start-spark.sh sleep 120
#Delete input and output directories from prior testing
/opt/yarn/hadoop-3.3.1/bin/hadoop --config /opt/yarn/hadoop-3.3.1/etc/hadoop fs -rm -r -skipTrash
hdfs://spark1.gyasi.local:9000/HiBench/Kmeans/Input
/opt/yarn/hadoop-3.3.1/bin/hadoop --config /opt/yarn/hadoop-3.3.1/etc/hadoop fs -rm -r -skipTrash
hdfs://spark1.gyasi.local:9000/HiBench/Kmeans/Output
sleep 3
#Clear the memory cache on each datanode
~/scripts/reset-testbed.sh sleep 30

#Change memory used by Spark
sed -i '/spark.executor.memory/ c\spark.executor.memory 52g' ~/Hibench/conf/spark.conf sed -i '/
spark.driver.memory/ c\spark.driver.memory 52g' ~/Hibench/conf/spark.conf
#Change dataset size to bigdata
sed -i '3s/gigantic/bigdata/' ~/Hibench/conf/hibench.conf
#Run Kmeans test and copy results to Kmeans subdirectory within the results directory for i in {1..3}
do
~/scripts/kmeans-test.sh mkdir ${kmeans_results}/run$i
for j in {1..5} do
cp -fv /tmp/spark$j.nmon ${kmeans_results}/run$i/${platform}_kmeans_node${j}_
run${i}_${timestamp}.nmon
done
cp -fv ~/Hibench/report/hibench.report ${kmeans_results}/run$i/${platform}_kmeans_hibench_
```

```
run${i}. report
~/scripts/reset-testbed.sh sleep 30
done
#Remove input and output directories generated for Kmeans data
/opt/yarn/hadoop-3.3.1/bin/hadoop --config /opt/yarn/hadoop-3.3.1/etc/hadoop fs -rm -r -skipTrash
hdfs://spark1.gyasi.local:9000/HiBench/Kmeans/Input
/opt/yarn/hadoop-3.3.1/bin/hadoop --config /opt/yarn/hadoop-3.3.1/etc/hadoop fs -rm -r -skipTrash
hdfs://spark1.gyasi.local:9000/HiBench/Kmeans/Output
sleep 3
#Stop Spark & Hadoop
~/scripts/stop-spark.sh
#Poweroff datanodes for i in {2..5}
do
ssh spark$i poweroff done
```

start_spark.sh (namenode)

```
/opt/yarn/hadoop-3.3.1/bin/hdfs --daemon start namenode
/opt/yarn/hadoop-3.3.1/bin/hdfs --daemon start secondarynamenode
/opt/yarn/hadoop-3.3.1/bin/yarn --daemon start resourcemanager
/opt/yarn/hadoop-3.3.1/bin/yarn --daemon start nodemanager
/opt/yarn/spark-3.2.1-bin-hadoop3.3/sbin/start-manager.sh sleep 60
ssh spark2 '~/start-spark.sh' ssh spark3 '~/start-spark.sh' ssh spark4 '~/start-spark.sh' ssh spark5
'~/start-spark.sh'
```

start_spark.sh (datanodes)

```
/opt/yarn/hadoop-3.3.1/bin/hdfs --daemon start datanode
/opt/yarn/spark-3.2.1-bin-hadoop3.3/sbin/start-worker.sh spark://[MANAGER IP ADDRESS]:7077
```

stop_spark.sh (namenode)

```
ssh spark2 '~/stop-spark.sh' ssh spark3 '~/stop-spark.sh' ssh spark4 '~/stop-spark.sh' ssh spark5
'~/stop-spark.sh'
/opt/yarn/hadoop-3.3.1/bin/hdfs --daemon stop namenode
/opt/yarn/hadoop-3.3.1/bin/hdfs --daemon stop secondarynamenode
/opt/yarn/hadoop-3.3.1/bin/yarn --daemon stop resourcemanager
/opt/yarn/hadoop-3.3.1/bin/yarn --daemon stop nodemanager
/opt/yarn/spark-3.2.1-bin-hadoop3.3/sbin/stop-manager.sh
```

stop_spark.sh (datanodes)

```
/opt/yarn/hadoop-3.3.1/bin/hdfs --daemon stop datanode
/opt/yarn/spark-3.2.1-bin-hadoop3.3/sbin/stop-worker.sh
```

reset_testbed.sh

```
#!/bin/sh
sync; echo 3 > /proc/sys/vm/drop_caches
ssh -t spark2 'sync; echo 3 > /proc/sys/vm/drop_caches' ssh -t spark3 'sync; echo 3 > /proc/sys/vm/
drop_caches' ssh -t spark4 'sync; echo 3 > /proc/sys/vm/drop_caches' ssh -t spark5 'sync; echo 3 > /
proc/sys/vm/drop_caches'
```

kmeans-test.sh

```
#!/bin/sh
echo "Preparing Kmeans test" echo " "
echo " " sleep 5
~/Hibench/bin/workloads/ml/kmeans/prepare/prepare.sh sleep 60
nmon -F /tmp/spark1.nmon -s1 -c3600 -J -t for i in {2..5}
do
ssh spark$i nmon -F /tmp/spark$i.nmon -s1 -c3600 -J -t done
echo "Running Kmeans test" echo " "
echo " " sleep 5
time ~/Hibench/bin/workloads/ml/kmeans/spark/run.sh sleep 5
pkill nmon
for i in {2..5} do
ssh spark$i pkill nmon done
for i in {2..5} do
scp spark$i:/tmp/spark$i.nmon /tmp/ done
```

Read the report at https://facts.pt/odi9nGQ ▶

This project was commissioned by Intel.