Intel® Virtual RAID on CPU (Intel® VROC) Integrated Caching (Intel VROC IC) with MySQL

Performance Evaluation Guide

November 2020

Document Number: 344690-001US

Revision History

Revision	Description	Revision Date
001	• Initial Release	November 2020

Contents

1 Introduction

1.1 Purpose

This is an Intel® Virtual Raid on CPU (Intel® VROC) Integrated Caching configuration guide to help users set up Intel VROC RAID arrays and caching layers to achieve high performance results with the MySQL application. There are two scenarios that this guide will address:

1. Users looking to reproduce the performance results achieved and documented by Intel.
2. Users looking to implement this storage architecture on a production system.

To reproduce the performance result in this guide, follow the steps exactly.

1.2 Platform Support

Intel® Wolf Pass server board with Intel® Xeon® Scalable Platform family and Intel® C620 series chipset.

https://ark.intel.com/content/www/us/en/ark/products/89015/intel-server-board-s2600wft.html

§

2 Evaluation Setup

This document is intended as a guide for using MySQL workload in systems containing Intel® Xeon® Scalable Processors with Red Hat Enterprise Linux 7.8, and the latest Intel® Virtual RAID on CPU (Intel® VROC) Integrated Caching (Intel® VROC IC) package. It includes testing of the Intel® SSD DC P4610 and Intel® Optane™ SSD DC P4800X. Specific steps are for testing MySQL OLTP (on-line transaction processing) workload with NVMe SSD RAID arrays on Intel® Volume Management Device (Intel® VMD) - enabled PCIe lane and Intel® VROC Integrated Caching (Intel® VROC IC).

Further details for each command are provided in subsequent sections of this document.

2.1 Hardware Configuration

For maximum use of Intel® VROC with availability to RAID 0, RAID 5, RAID 1, and RAID 10, an Intel VROC hardware key ("Premium" or "Intel-SSD-only") must be installed on the motherboard.

The following info is based on an Intel® Wolf Pass server board S2600WFT.

1. **TLC Performance**

   System configuration: Intel S2600WFT Platform, 2x Intel® Xeon® Gold 6240M CPU 18cores@2.60GHz, DRAM 192 GB, BIOS Version: SE5C620.86B.02.01.0010.010620200716
   OS: RedHat Enterprise Linux v7.8, 3.10.0-1127.el7.x86_64, mdadm - v4.1 - 2018-10-01, THP disabled
   BIOS setting: PackageC-State(C6), HardwareP-States(Native Mode), CPU Power and Performance Policy (Balanced Performance)
   Intel VROC (SATA RAID) 4-Disk RAID5, chunk size=8k, group_thread_cnt=8
   Intel VROC Storage: 4x Intel® DC SSD P4510 2 TB with Intel VROC RAID5 chunk size=8k, group_thread_cnt=8, Intel VROC Integrated Caching with 2x 375 GB Intel® Optane™ SSD DC P4800X Series (Model: SSDPE21K375GA) with Intel VROC RAID1, Write Only Mode, 4k Cache Line Size
   RAID HBA Storage: 4x Intel® DC SSD P4610 1.6 TB with Intel RAID Adapter RSP3TD160F RAID5
   MySQL Benchmark: Sysbench 1.1.0-bd4b418, oltp_read_write, MySQL 8.0.21, 120 GB Database, 1hr test, 64 threads
   Performance results are based on testing as of 8/12/2020 and may not reflect all publicly available security updates

While these steps are based on the Intel® Wolf Pass server board, they may be replicated on other platforms with the directions followed as closely as possible. For more information, see Section 1.2, Platform Support.

Note: For best results, Intel recommends populating all DIMM channels with applicable RAM (same manufacturer, same model, same size, same speed, etc).

2.2 BIOS Settings

The following BIOS settings are based on an Intel® Wolf Pass server board S2600WFT and are recommended for maximum performance. Please refer to the instructions that have been supplied by the user's platform BIOS vendor, as those instructions may differ from the set below.

Enter the BIOS at boot up by pressing <F2>. Load the defaults by pressing <F9>. After loading the defaults in the BIOS, navigate to the following:

Verify... Advanced → Processor Configuration → Intel® Hyper-Threading Tech → Enabled
Change... Advanced → Power & Performance → CPU Power and Performance Policy → Performance
Change... Advanced → Power & Performance → Workload Configuration → I/O Sensitive
Verify... Advanced → Power & Performance → Uncore Power Management → Uncore Frequency Scaling → Enabled
Verify... Advanced → Power & Performance → Uncore Power Management → Performance P-limit → Enabled
Verify... Advanced → Power & Performance → CPU P State Control → Enhanced Intel SpeedStep® Tech → Enabled
Verify... Advanced → Power & Performance → CPU P State Control → Intel Configurable TDP → Disabled
Verify... Advanced → Power & Performance → CPU P State Control → Intel® Turbo Boost Technology → Enabled
Verify... Advanced → Power & Performance → CPU P State Control → Energy Efficient Turbo → Enabled
Verify... Advanced → Power & Performance → Hardware P States → Hardware P-States → Native Mode
Verify... Advanced → Power & Performance → Hardware P States → HardwarePM Interrupt → Disabled
Verify... Advanced → Power & Performance → Hardware P States → EPP Enable → Enabled
Verify... Advanced → Power & Performance → Hardware P States → APS rocketing → Disabled
Verify... Advanced → Power & Performance → Hardware P States → Scalability → Disabled
Verify... Advanced → Power & Performance → Hardware P States → PPO-Budget → Disabled
Change... Advanced → System Acoustic and Performance Configuration → Set Fan Profile → Performance

With Intel VROC, Intel VMD will also need to be enabled on your new platform. The information below is based on a 4-port retimer (x16) installed on Riser 2, PCIe slot 1; your setup may be different. Please enable VMD and their respective ports at the applicable Riser/PCIe Slot in your configuration.

Note: *8-port switches (x8) have only two VMD ports.*

Change... Advanced → PCI Configuration → PCIe Slot Bifurcation Setting → Riser_Slot_2 Bifurcation → x4x4x4x4
Change... Advanced → PCI Configuration → Volume Management Device → Riser2, Slot1 Volume Management → Enabled
Change... Advanced → PCI Configuration → Volume Management Device → VMD Port 2A → Enabled
Change... Advanced → PCI Configuration → Volume Management Device → VMD Port 2B → Enabled
Change... Advanced → PCI Configuration → Volume Management Device → VMD Port 2C → Enabled
Change... Advanced → PCI Configuration → Volume Management Device → VMD Port 2D → Enabled

2.3 Intel® Virtual RAID On CPU (from the BIOS)

The following information is based on an Intel® Wolf Pass server board S2600WFT. For other platforms, accessing Intel Virtual RAID On CPU in the BIOS may differ from instructions below.

After adjusting the BIOS settings as described in Section 2.2 above, reboot the system, and enter BIOS again by pressing <F2>.

Navigate to Advanced → PCI Configuration → UEFI Option ROM Control → Intel® Virtual RAID on CPU

From here, you can see the Intel VROC hardware key installed, Intel® VROC Pre-OS version number, and any existing Intel VROC RAID volumes. Select "All Intel VMD Controllers" to view the applicable NVMe SSDs.

2.4 Red Hat Enterprise Linux Installation/Dependencies

All Red Hat Enterprise Linux installations must be done in UEFI mode.

After entering the "INSTALLATION SUMMARY" screen, the "Minimal Install" option under "SOFTWARE SELECTION" is sufficient for this performance testing.

Once installation is complete, set up a repository containing all the necessary rpm packages, and install the following dependencies by running the following command:

```
#yum install gcc libaio-devel zlib-devel unzip sysstat libreport-filesystem numactl
    redhat-lsb-core sg3_utils nvme-cli iotop hdparm wget htop –y
```

Note: Please refer to the Redhat website for more details on configuring the repository: *https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/deployment_guide/sec-configuring_yum_and_yum_repositories*

Note: User must have root privilege to install necessary rpm packages.

Note: The environment tested in this guide is configured with RHEL installed in a separate single SATA drive.

2.5 Installing Intel® VROC Integrated Caching

Intel VROC packages, the kernel module, ledmon and mdadm userspace packages are included inbox in the RHEL 7.8. Follow the steps below to install the cache acceleration software in RHEL 7.8 for Intel VROC IC.

1. Copy "open-cas-linux-20.03.1.0292-master.x86_64.rpm" and "open-cas-linux-modules_k3.10.0_1127.el7-20.03.1.0292-master.x86_64" files to system.
2. Once RHEL 7.8 has been installed, reboot the system. Navigate to the directory where rpm packages are stored and install packages using the following rpm commands:

```
#sudo rpm -i open-cas-linux-modules_k3.10.0_1127.el7-20.03.1.0292-master.x86_64
#sudo modprobe cas_cache
#sudo rpm -i open-cas-linux-20.03.1.0292-master.x86_64.rpm
```

3. Use **casadm** command to confirm that installation was successful:
   ```
   #sudo casadm -V
   ```

Note: *sudo is not needed if the user has root privilege.*

The followiing output indicates all of the kernel modules, and CLI utility is installed with the correct version.

Name	Version
CAS Cache Kernel Module	20.03.01.00000723
CAS Disk Kernel Module	20.03.01.00000723
CAS CLI Utility	20.03.01.00000723

2.6 Downloading and Installing Intel® Memory and Storage Tool

Intel® Memory and Storage Tool (Intel® MAS) is used to update firmware on Intel SSDs, and is used to perform a low-level format on Intel NVMe SSDs.

1. Download Intel® MAS at:
   https://downloadcenter.intel.com/download/29821?v=t

   ```
   #unzip Intel®_MAS_CLI_Tool_1.2_Linux.zip
   #rpm -i intelmas-1.2.79-0.x86_64.rpm
   ```

2. Verify Intel® MAS installation by running the intelmas command:

   ```
   #intelmas version
   ```

3. To list Intel SSD devices in Intel® MAS:

   ```
   #intelmas show -intelssd
   ```

2.7 Setting System to Performance Mode

Set up the system in performance mode with the systemd service.

1. First create and update the rc.local script into /etc/rc.d then restart the systemd rc-local service.

   ```
   #vi /etc/rc.d/rc.local
   ```

2. The content of the rc.local:

   ```
   #!/bin/bash
   cpupower frequency-set -g performance
   ```

3. Run/restart the systemd service using systemctl:

   ```
   #chmod 755 /etc/rc.d/rc.local
   #systemctl enable rc-local.service
   #systemctl start rc-local.service
   #systemctl status rc-local.service
   ```

2.8 Test Drive Configuration and Preparation

2.8.1 Formatting the NVMe Intel SSDs

Drives with previous RAID metadata, or that were previously used for other applications, could have diminished performance. Clearing the metadata and erasing or performing a low-level format can stabilize your drives, providing more consistent results.

1. To remove the RAID metadata:

```
#mdadm –-zero-superblock /dev/nvme*n1
```

2. To show Intel SSD in Intel MAS indexes:

```
#intelmas show -intelssd
```

3. To perform a low-level format:

```
#intelmas start -force -nvmeformat -intelssd <intelmas index number>
```

Note: Repeat these steps for all SSDs to be used in new array.

2.8.2 Setup RAID Volume and Cache Device

1. Create the Core 4Disk RAID5 Volume:

```
#mdadm -C /dev/md/imsm0 /dev/nvme[0-3]n1  -n4 -e imsm  (example to create container
    with 4 disk nvme[0-3]n1)

#mdadm -C /dev/md1  /dev/md/imsm0 -l5 -n4 -c8  (example to create 4Disk RAID5 with
    chunk size 8k)
```

2. Create the cache 2Disk RAID1 volume:

```
#mdadm -C /dev/md/imsm1  /dev/nvme[4-5]n1 -n2 -e imsm  (example to create container
    with 2 disk nvme[4-5]n1)

#mdadm -C /dev/md2 /dev/md/imsm1 -l1 -n2  (example to create 2Disk RAID1 for
    caching volume)
```

Note: The re-sync will be triggered after RAID volume is created. It may take some time to complete re-sync depending on the RAID volume capacity.

3. Change the group_thread_cnt for the performance.

   After creating a RAID5 volume, there is a setting that has demonstrated slightly improved RAID5 performance. This setting is not found with RAID 0, RAID 1, or RAID 10.
   Apply the following setting to a RAID 5 volume:

```
#echo 8 > /sys/block/md1/md/group_thread_cnt  (assume md1 is the RAID5 volume)
```

Note: A group thread count of 0 is the default and limited testing has been conducted with other group thread counts. A group thread count of 4 was determined to be the most efficient, boosting performance ~10%. More testing is planned to determine optimal performance and latency impacts. Larger group thread counts appear to impact read performance.

4. Change the sync_speed_max for initialization speed.

   After creation of RAID 5, RAID 1, or RAID 10 volumes, initialization will start automatically.
   To speed up initialization or rebuild times, the following setting can be applied.

```
#echo 1000000 > /sys/block/md1/md/sync_speed_max   (assume md1 is the RAID5 volume)
```

Note: The default setting for “sync_speed_max” is 200,000.

2.9 Configure the Run Caching

2.9.1 Configure Caching

1. Start caching on either write-only or write-back cache policy.

   a. Start Caching with write-only cache policy:

   ```
   #casadm -S -d /dev/md2 -c wo -x 4   (md2 is the cache optane 2 disks RAID1 volume)
   #casadm -A -i 1 -d /dev/md1  (md1 is the core 4 disk RAID5 volume)
   ```

   b. Start caching with write-back policy:

   ```
   #casadm -S -d /dev/md2 -c wb -x 4   (md2 is the cache optane 2 disk RAID1 volume)
   #casadm -A -i 1 -d /dev/md1   (md1 is the core 4 disk RAID5 volume)
   ```

2. Load the IO classification file based on your cache policy

   Create the IO classification file for write-only (a) or write-back (b).

   a. Create the ioclass.csv as follow content in /etc/opencas/ioclass_mysql_wo.csv

   ```
   IO class id,    IO class name,    Eviction priority,Allocation
   0,  unclassified,  255,    0
   1,  request_size:ne:16384&done    ,0, 1
   ```

   b. Create the ioclass.csv as follow content in /etc/opencas/ioclass_mysql_wb.csv

   ```
   IO class id,    IO class name,    Eviction priority,Allocation
   0,  unclassified,  22, 1
   1,  metadata&done, 1,  1
   2,  request_size:ne:16384&done,  0,  1
   ```

3. Load the IO classification file for write-only (a) or write-back (b).

   a. Load the write-only caching ioclass file if you are enabling write-only caching:

   ```
   #casadm -C -C -i 1 -f /etc/opencas/ioclass_mysql_wo.csv
   ```

   b. Load the Write-back caching ioclass file if you are enabling write-back caching

   ```
   #casadm -C -C -i 1 -f /etc/opencas/ioclass_mysql_wb.csv
   ```

Note: The MySQL testing was performed using the 4K cacheline size with 4DR5 on 8K chunk size. Performance may vary with different numbers of RAID member drives, different cacheline size, and chunk size configurations.

2.9.2 Create Filesystem and Mount the Storage Node

1. Create the filesystem with xfs on the cache device, and mount:

   ```
   #mkfs.xfs -K -f /dev/cas1-1

   #mount /dev/cas1-1 /var/lib/mysql/
   #chown mysql:mysql /var/lib/mysql/
   ```

2. Update the /etc/fstab with UUID to mount the drive in the system reboot to list all the UUID in the sysfs:

   ```
   #lsblk -f
   ```

3. Edit the /etc/fstab as in the following example:

   ```
   UUID=c346151a-dedc-41c4-b362-97033c69def6 /var/lib/mysql  xfs   defaults      1 2
   ```

§

3 Preparing MySQL

3.1 Download and Install sysbench

MySQL performance evaluation uses sysbench with the OLTP benchmark. Follow these steps to download and install sysbench.

1. Download sysbench repo lists:

```
#curl –s https://packagecloud.io/install/repositories/akopytov/sysbench/script.rpm.sh
   | bash
```

2. Verify the sysbench repository was added to the repo lists:

```
#yum repolist
```

The output should be similar to the following, which shows sysbench repository in repo lists.

```
repo id                          repo name                                   Status
akopytov_sysbench/x86_64         akopytov_sysbench                           30
akopytov_sysbench-source         akopytov_sysbench-source                    0
base/7/x86_64                    CentOS-7 - Base                             9,911
epel/x86_64                      Extra Packages for Enterprise Linux 7 - x86_64   12,646
extras/7/x86_64                  CentOS-7 - Extras                           370
mysql-connectors-community/x86_64   MySQL Connectors Community               63
mysql-tools-community/x86_64     MySQL Tools Community                       69
mysql80-community/x86_64         MySQL 8.0 Community Server                  33
updates/7/x86_64                 CentOS-7 - Updates                          1,054
```

3. Install sysbench:

```
#yum -y install sysbench
```

4. After installation, verify the installation using the sysbench command and list the version:

```
#sysbench –-version
```

The sysbench installed version show v1.0.20 (it may be updated based on repository).

```
Sysbench 1.0.20
```

3.2 Download and Install MySQL 8.0

1. Install MySQL 8.0 rpm package from yum repository in MySQL web site. It will add the repository in your local repo list.

```
#rpm -ihv https://dev.mysql.com/get/mysql80-community-release-el7-3.noarch.rpm
```

2. Verify the local repo list by running the yum command. Your local repo list should display all MySQL Connectors Community, MySQL 8.0 Community Server, and MySQL Tools Community.

```
#yum repolist
```

The output should be similar to the following, showing MySQL repository in local repo list.

```
Repo id                          repo name                          status
mysql-connectos-community/x86_64 MySQL Connectors Community         43
mysql-tools-community/x86_64     MySQL Tools Community              74
mysql80-community/x86_64         MySQL 8.0 Community Server         33
```

3. Install MySQL using the yum command:

```
#yum install mysql-community-server
```

```
================================================================================================
 Package                        Arch        Version              Repository              Size
================================================================================================
Installing:
 mysql-community-libs           x86_64      8.0.21-1.el7         mysql80-community      4.5 M
     replacing  mariadb-libs.x86_64 1:5.5.65-1.el7
 mysql-community-libs-compat    x86_64      8.0.21-1.el7         mysql80-community      1.2 M
     replacing  mariadb-libs.x86_64 1:5.5.65-1.el7
 mysql-community-server         x86_64      8.0.21-1.el7         mysql80-community      499 M
Installing for dependencies:
 mysql-community-client         x86_64      8.0.21-1.el7         mysql80-community       48 M
 mysql-community-common         x86_64      8.0.21-1.el7         mysql80-community      617 k

Transaction Summary
================================================================================================
Install  3 Packages (+2 Dependent packages)
```

4. After installation, use the mysql command to verify MySQL installation and list the version:

```
#mysql --version
```

The installed MySQL version shows v8.0.31 (it may be updated based on repository):

```
mysql  Ver 8.0.21 for Linux on x86_64 (MySQL Community Server - GPL)
```

3.3 Starting MySQL Service

3.3.1 Pre-requisites:

1. Remove all files in /var/lib/mysql
2. Check and disable SELinux (only for testing). Issue the following command to check if SELinux is disabled in the system:

```
#cat /etc/sysconfig/selinux
```

The following output shows SELinux is disabled:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of three two values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

If SELinux is not disabled, take the following steps to disable it.

1. Edit the sysconfig file and add “SELINUX=disabled” in the file:

```
#vi /etc/sysconfig/config
```

2. Edit grub configure file, in the GRUB_CMDLINE_LINUX parameter add “selinux=0”

```
#vi /etc/default/grub
```

3. Rebuild the grub configure by running grub2 command:

```
#grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

4. Reboot the system and once again check if SELinux is disabled.

3.3.2 Configuration

Edit the MySQL configuration file.

1. Edit the MySQL configuration with following content:

   ```
   #vi /etc/my.cnf
   ```

2. Add the following configuration content in the file:

```
# This group are read by MySQL server.
# Use it for options that only the server (but not clients) should see
#
# For advice on how to change settings please see
# http://dev.mysql.com/doc/refman/en/server-configuration-defaults.html

# Settings user and group are ignored when systemd is used.
# If you need to run mysqld under a different user or group,
# customize your systemd unit file for mysqld according to the
# instructions in http://fedoraproject.org/wiki/Systemd

[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
log-error=/var/log/mysqld.log
pid-file=/run/mysqld/mysqld.pid

# general
ssl=0
server_id=5400
performance_schema=on
max_connections=4000
back_log=1500
table_open_cache=8000
table_open_cache_instances=16
max_prepared_stmt_count=512000
default_password_lifetime=0
character_set_server=latin1
collation_server=latin1_swedish_ci
skip-character-set-client-handshake
transaction_isolation=REPEATABLE-READ
lc_messages=en_US
default_authentication_plugin=mysql_native_password

# files
innodb_file_per_table
innodb_log_file_size=1024M
innodb_log_files_in_group=32
innodb_open_files=4000

# buffers
#innodb_buffer_pool_size=128000M
#innodb_buffer_pool_size=20000M
innodb_buffer_pool_size=32000M
innodb_buffer_pool_instances=16
innodb_log_buffer_size=64M

# tune
innodb_thread_concurrency=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
innodb_io_capacity=10000
innodb_io_capacity_max=40000
innodb_lru_scan_depth=9000
innodb_change_buffering=none
innodb_read_only=0
innodb_page_cleaners=16
innodb_undo_log_truncate=off

# data protection
innodb_flush_log_at_trx_commit=1
innodb_doublewrite=1
innodb_checksum_algorithm=crc32

# binlog-ON
# ---------------------
log_bin=binlog
sync_binlog=1
binlog_format=ROW
binlog_row_image=minimal

# binlog-OFF
# ---------------------
#skip_log_bin=1

# 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='%'
```

3.3.3 Initialize and Run MySQL

1. Remove all of the content from the MySQL directory:

   ```
   #rm -rf /var/lib/mysql/*
   ```

2. Create mysql-files directory and change ownership:

   ```
   #mkdir /var/lib/mysql-files
      #chown mysql:mysql /var/lib/mysql-files
   ```

3. Run MySQL daemon and service:

   ```
   #mysqld --initialize-insecure --user=mysql
   ```

   ```
   #systemctl start mysqld
   ```

4. Verify that mysqld service is running successfully:

   ```
   #systemctl status mysqld
   ```

3.4 Preparing the Database (OLTP Usage)

3.4.1 Create the Database with Sysbench

1. Create the MySQL database using the mysql command and exit:

   ```
   #mysql
   ```

2. Inside the MySQL command console, run the following command to create the database:

   ```
   mysql > create database test_database;
   ```

3. Run the sysbench command to prepare the database.
   Following is an example to create different database tables and table_size with 123 GB and 756 GB.
   Prepare small database ~123 GB using sysench command.

   ```
   #sysbench oltp_read_write --mysql-socket=/var/lib/mysql/mysql.sock --mysql-user=root -
      -mysql-db=test_database --threads=16 --tables=8 --table_size=50000000 prepare
   ```

Note: It may take some time to complete database preparation.

Note: If user wants to create larger database ~756 GB for testing. Change the tables property from 8 to 32 and table_size from 50,000,000 to 100,000,000.

3.4.2 Verify the Database Creation

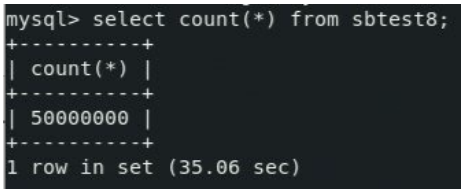
The database creation takes hours. Once it is done, check the newly created database using the show databases command inside the MySQL command console.

1. Log in to the MySQL command console, and check the records; use database and show tables:

   ```
   mysql> show databases;
   mysql> use test_database;
   mysql> show tables;
   ```

2. Use the following command to return the table_size and verify that you have all records:

   ```
   mysql> select count(*) from sbtest8;
   ```



§

4 Running the Benchmark

4.1 Benchmark with point_select and oltp_read_write

The primary benchmarks are point_select and oltp_read_write. The my.cnf offers a lot of variations and the vendors provide benchmarking capable my.cnf, as shown below.

- Oracle benchmark is documented here:
  http://dimitrik.free.fr/blog/posts/mysql-performance-80-and-sysbench-oltp_rw-updatenokey.html
- Percona benchmark is documented here:
  https://github.com/Percona-Lab-results/201805-sysbench-tpcc-binlog-fs/blob/master/cnf/my.cnf
- Important safe (production worthy) parameters are documented here:
  https://www.percona.com/blog/2018/07/13/on-mysql-and-intel-optane-performance/
- Be aware of randomization types which tremendously affect the benchmark and IO usage.
  Pseudo-Random Numbers Generator options:

  --rand-type=STRING random numbers distribution {uniform,gaussian,special,pareto} [special]

  --rand-spec-iter=N number of iterations used for numbers generation [12]

  --rand-spec-pct=N percentage of values to be treated as 'special' (for special distribution) [1]

  --rand-spec-res=N percentage of 'special' values to use (for special distribution) [75]

  --rand-seed=N seed for random number generator. When 0, the current time is used as a RNG seed. [0]

  --rand-pareto-h=N parameter h for pareto distribution [0.2]

- Example mysql.sh script with 64 and 128 thread run oltp_read_write test for 60 mins:

```
#!/bin/bash
log=log-nowarmup-pointselect-`date +%y%m%d-%H%M%S`

for t in  64 128
do
sysbench oltp_read_write --mysql-socket=/var/lib/mysql/mysql.sock --mysql-user=root --
   mysql-db=test_database --time=3600 --threads=$t --report-interval=1 --percentile=99
   --tables=8 --table_size=50000000 --rand-type=uniform run >> $log 2>&1
done
```

- Run the script:

```
#chmod +x mysql.sh
#./mysql.sh
```

Note: If you are running an official test, Intel recommends warming up DRAM first with short test of point_select

Note: Numbers of tables and table_size must be in proportion to the size of the OLTP (on-line transaction processing) da*tabase.*

§

5 Test Results

Intel VROC IC for MySQL with Med-Capacity Performance (NVMe TLC SSD)

MySQL			
Metric	Legacy HBA	Intel VROC IC (375 GB)	%
TPS	2,606 tps	5,104 ops	↑96%
Avg. Latency	24 us	12 us	↓50%
Storage Lifetime Ops.	533B	480B	↓10%

The achieved results, presented above, are based on the documented steps, and the configuration detail provided in Section 2.1, Hardware Configuration.

Results may vary based on, among other things, different hardware and software configuration.

§