Beyond the Report:

A Comparative Study of Software and Hardware RAID Performance and Benefits

This methodology report provides the system configuration details and step-by-step procedures for the Prowess Consulting benchmark testing on the following Dell™ platforms:

1. One Dell™ PowerEdge™ R760 server with the Red Hat® Enterprise Linux® operating system (OS) and built-in software RAID
2. One PowerEdge R760 server with the Windows Server® 2022 OS and Dell™ PowerEdge RAID Controller (PERC) S160 software RAID
3. One PowerEdge R760 server with the Red Hat Enterprise Linux OS and Dell PERC H965i hardware RAID
4. One PowerEdge R760 server with the Windows Server 2022 OS and Dell PERC H965i hardware RAID

For the full analysis, read the **technical research report**.

Testing was concluded on October 23, 2024.

Server Configurations

	Server 1	Server 2	Server 3	Server 4
System	Dell™ PowerEdge™ R760	Dell™ PowerEdge™ R760	Dell™ PowerEdge™ R760	Dell™ PowerEdge™ R760
CPU	2 x Intel® Xeon® Platinum 8460Y+ processor (model 143, stepping 8)	2 x Intel® Xeon® Platinum 8460Y+ processor (model 143, stepping 6)	2 x Intel® Xeon® Platinum 8460Y+ processor	2 x Intel® Xeon® Platinum 8452Y processor
Total cores/threads per CPU	80/160	80/160	80/160	72/144
CPU frequency	2.0 GHz	2.0 GHz	2.0 GHz	2.0 GHz
Storage controller 01	Marvell Technology Group Ltd. Dell™ Boot-Optimized Server Storage (BOSS)-N1 monolithic	Marvell Technology Group Ltd. Dell™ Boot-Optimized Server Storage (BOSS)-N1 monolithic	Marvell Technology Group Ltd. Dell™ Boot-Optimized Server Storage (BOSS)-N1 monolithic	Marvell Technology Group Ltd. Dell™ Boot-Optimized Server Storage (BOSS)-N1 monolithic
Disk	Dell™ Enterprise NVM Express® (NVMe®) CM6 MU 1.6 TB firmware 2.2.0	Dell™ Enterprise NVM Express® (NVMe®) CM6 MU 1.6 TB firmware 2.2.0	Dell™ NVM Express® (NVMe®) PE8010 RI M.2 960 GB (SK hynix®)	Dell™ Enterprise NVM Express® (NVMe®) ISE 7400 RI M.2 480 GB (Micron Technology Inc)
Number of disks	2	2	2	2
Storage controller 02	Multi-device RAID	Dell PERC S160 software RAID	Broadcom®/LSI® Dell™ PERC H965i Front (firmware: 8.4.0.0.18-29)	Broadcom®/LSI® Dell™ PERC H965i Front (firmware: 8.4.0.0.18-29)

Disk	1.6 TB Dell™ Enterprise NVMe® CM6 MU (KIOXIA Corporation) (firmware: 2.2.0)	1.6 TB Dell™ Enterprise NVMe® CM6 MU (KIOXIA Corporation) (firmware: 2.2.0)	1.6 TB Dell™ Enterprise NVMe® CM6 MU (KIOXIA Corporation) (firmware: 2.2.0)	1.6 TB Dell™ Enterprise NVMe® CM6 MU (KIOXIA Corporation) (firmware: 2.2.1)
Number of disks	8	8	8	8
Memory	8 x 32 GB (256 GB) DDR5, 4,800 megatransfers per second (MT/s)	8 x 32 GB (256 GB) DDR5, 4,800 MT/s	16 x 16 GB (256 GB) SK hynix® HMCG78MEBRA174N, 4,800 MT/s single-rank	16 x 16 GB (256 GB) SK hynix® HMCG78MEBRA174N, 4,800 MT/s single-rank
Network	1 x 10/25/40/50/100/200 Gb Broadcom® NetXtreme® E-Series BCM57504 (rev 12)	N/A (in Prowess Consulting's lab; no network card needed)	2 x 100 Gb Broadcom® NetXtreme® E-Series P2100D BCM57508 QSFP PCIe®	N/A (in Prowess Consulting's lab; no network card needed)
OS	Red Hat® Enterprise Linux® version 9.3	Windows Server® 2022 Datacenter Evaluation Desktop Experience version 21H2	Red Hat® Enterprise Linux® version 9.2	Windows Server® 2022 Datacenter Evaluation Desktop Experience version 21H2
BIOS version	1.5.6	1.5.6	1.5.6	2.1.5

Testing Summary

To understand the current benefits of software and hardware RAID, Prowess Consulting conducted a research study on four similarly configured Dell PowerEdge R760 servers using each RAID type. Through this study, sponsored by Dell Technologies, we aimed to provide a deeper understanding of the performance and resource utilization of both RAID types, hardware and software, and the scenarios in which one type might be better than the other.

Our engineers ran tests comparing performance for each server using eight total configurations encompassing three RAID types:

- RAID 0 with 1, 2, 4, and 8 drives
- RAID 10 with 4 and 8 drives
- RAID 5 with 4 and 6 drives

For our study, we used the following benchmarking tools:

- **Flexible I/O Tester (fio)**, a free and open-source tool for benchmarking and testing the performance of storage systems. Fio can be used to generate a wide range of input/output (I/O) workloads to simulate various real-world scenarios.
- **Iometer**, an I/O subsystem-measurement and characterization tool for single and clustered systems. Iometer was originally developed by Intel and is now maintained by an international group of individuals.
- **Performance Monitor (PerfMon)**, a system-monitoring tool for Windows® that monitors computer activities such as CPU usage and memory usage and that also measures I/O operations per second (IOPS). PerfMon performs asynchronous I/O and allows the configuration of disk parameters such as maximum disk size, starting disk sector, and number of outstanding I/O operations.
- **I/O statistics (iostat)**, which collects and shows OS storage I/O statistics. Iostat is often used to identify performance issues with storage devices, including local disks, or remote disks accessed over file systems such as Network File System (NFS).

Test Parameters

Our test parameters for Iometer/fio on Windows Server 2022 and fio on Red Hat Enterprise Linux 9 consisted of the following:

- The full raw array size was used under test.
- I/O sizes: 4 KiB, 8 KiB, 64 KiB, and 128 KiB
- Queue depth levels: 1, 4, 16, 32, 64, and 128
- Sequential tests:
  - One worker
  - 100% sequential reads and writes
- Random tests:
  - 8, 16, and 32 workers
  - 100% random reads and writes
  - Online transaction processing (OLTP) profile of 2:1 read:write
    - 70:30 read:write

- Pin the workload to the appropriate nonuniform memory access (NUMA) node.
- Run the tests while the RAID is optimal.
- Run the tests while the RAID is degraded.
- Run the tests while the RAID is under rebuild.

Red Hat® Enterprise Linux® 9.2

This section outlines the steps needed to set up the Red Hat Enterprise Linux environment and to configure the software RAID and conduct the testing.

Server Setup

These steps cover the initial instance setup for Red Hat Enterprise Linux using software RAID, multi-device admin (mdadm), and the Dell PERC H965i Front.

1. Install Red Hat Enterprise Linux 9.2 via the Integrated Dell™ Remote Access Controller (iDRAC) remote console, accepting the default values.
2. Once installation completes, reboot and boot into Red Hat Enterprise Linux 9.2.
3. At the resulting command prompt, run **sudo su** to switch to the root user.
4. To update the system and install monitoring tools, run the following commands:

```
dnf -y update
dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
dnf install -y nmon sysstat dstat
```

5. Run **vim /root/fio_runner.sh** to open the file for editing.
6. Paste in the contents of the of the **fio_runner.sh appendix**.
7. Press **:wq <enter>** to save and exit the file.
8. Run **chmod +x fio_runner.sh** to grant the execute permission.
9. Run **vim /root/rebuild_runner.sh** to open the file for editing.
10. Paste in the contents of the of the **rebuild_runner appendix**.
11. Press **:wq <enter>** to save and exit the file.
12. Run the following command to grant the execute permission.

```
chmod +x /root/rebuild_runner.sh
```

13. For each test parameter defined, create a fio job file as prescribed in the fio readme documentation, **1. fio - Flexible I/O tester rev. 3.36 — fio 3.36 documentation**.
14. Run **vim /root/fio_runner_rebuilding.sh** to open the file for editing.
15. Paste in the contents of the of the **fio_runner_rebuilding appendix**.
16. Press **:wq <enter>** to save and exit the file.
17. Run the following command to grant the execute permission:

```
chmod +x /root/fio_runner_rebuilding.sh
```

18. Run the following command to open raid_kick.sh for editing:

```
vim /root/raid_kick.sh
```

19. Paste in the contents of the of the **raid_kick appendix**.
20. Press **:wq <enter>** to save and exit the file.
21. Run the following command to grant the execute permission:

```
chmod +x /root/raid_kick.sh
```

22. Run the following command to open rebuild_logger.sh for editing:

```
vim /root/rebuild_logger.sh
```

23. Paste in the contents of the of the **rebuild_logger appendix**.
24. Press **:wq <enter>** to save and exit the file.
25. Run the following command to grant the execute permission:

```
chmod +x /root/rebuild_logger.sh
```

Software RAID Preparation and Configuration

This section covers the steps for creating RAID volumes on Red Hat Enterprise Linux using software RAID and mdadm.

Initial Configuration

1. To identify which physical drive slots are associated with which CPU nodes, look at **iDRAC** > **system** > **PCIe devices** > **CPU column associated with a given slot ID**. This information will be used to ensure the drives and the workload are associated with the same NUMA nodes.
2. Also confirm the disk serial numbers associated with each slot ID.
3. Run the following command to open disk_id.sh for editing:

```
vim /root/disk_id.sh
```

4. Paste in the contents of the **disk_id appendix** into the file.
5. Update the disk IDs to match the information checked in step 2.
6. Press **:wq <enter>** to save and exit the file.
7. Run the following command to grant the execute permission:

```
chmod +x /root/disk_id.sh
```

8. Run the following command to open drive_prep_script.sh:

```
vim /root/drive_prep_script.sh
```

9. Paste in the contents of the of the **drive_prep_script appendix**.
10. Press **:wq <enter>** to save and exit the file.
11. Run the following command to grant the execute permission:

```
chmod +x /root/drive_prep_script.sh
```

12. Run the following command to identify which NVM Express® (NVMe®) label is associated with which slot ID and thus which NUMA node. This information will be referenced when setting up the RAID volume.

```
/root/disk_id.sh
```

13. Run the following command to apply settings to each disk that might later be part of the RAID array:

```
for each in `/root/disk_id.sh | cut -d ' ' -f3`; do /root/drive_prep_script.sh $each; done
```

14. Confirm the type of RAID (0, 5, or 10) and number of disks (2, 4, 6, or 8) that you will be testing on, and then follow only the appropriate section from the following.

Create a One-Disk Volume

1. Referencing step 12 of the initial configuration section's output, select one disk that will be tested; for example, **nvme1n1**.
2. Run the following command to apply the drive settings to the volume:

```
/root/drive_prep_script.sh /dev/nvme1n1
```

3. Run the following command to create a file system on the new volume, **nvme1n1**:

```
mkfs.xfs /dev/nvme1n1
```

Create a Two-Disk RAID 0 Volume

1. Referencing step 12 of the initial configuration section's output, select two disks that will be part of the RAID array; for example, **nvme1n1** and **nvme2n1**.
2. Select the user's chosen name for the new RAID volume; for example, **md0**.
3. Run the following command to create the RAID volume:

```
mdadm -C /dev/md0 --level=0 /dev/nvme1n1 /dev/nvme2n1 --raid-devices=2
```

4. Run the following command to apply the drive settings to the volume:

```
/root/drive_prep_script.sh /dev/md0
```

5. Run the following command to create a file system on the new volume, named **md0**:

```
mkfs.xfs /dev/md0
```

Create a Four-Disk RAID 0 Volume

1. Referencing step 12 of the initial configuration section's output, select four disks that will be part of the RAID array; for example, **nvme1n1**, **nvme2n1**, **nvme3n1**, and **nvme4n1**.
2. Select the user's chosen name for the new RAID volume; for example, **md0**.
3. Run the following command to create the RAID volume:

```
mdadm -C /dev/md0 --level=0 /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1 /dev/nvme4n1 --raid-devices=4
```

4. Run the following command to apply the drive settings to the volume:
```
/root/drive_prep_script.sh /dev/md0
```
5. Run the following command to create a file system on the new volume, named **md0**:
```
mkfs.xfs /dev/md0
```

Create an Eight-Disk RAID 0 Volume

1. Referencing step 12 of the initial configuration section's output, select eight disks that will be part of the RAID array; for example, **nvme1n1**, **nvme2n1**, **nvme3n1**, **nvme4n1**, **nvme5n1**, **nvme6n1**, **nvme7n1**, and **nvme8n1**.
2. Select the user's chosen name for the new RAID volume; for example, **md0**.
3. Run the following command to create the RAID volume:
```
mdadm -C /dev/md0 --level=0 /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1 /dev/nvme4n1 /dev/nvme5n1 /dev/nvme6n1
/dev/nvme7n1 /dev/nvme8n1 --raid-devices=8
```
4. Run the following command to apply the drive settings to the volume:
```
/root/drive_prep_script.sh /dev/md0
```
5. Run the following command to create a file system on the new volume, named **md0**:
```
mkfs.xfs /dev/md0
```

Create a Four-Disk RAID 10 Volume

1. Referencing step 12 of the initial configuration section's output, select four disks that will be part of the RAID array; for example, **nvme1n1**, **nvme2n1**, **nvme3n1**, and **nvme4n1**.
2. Select the user's chosen names for the two component RAID 1 volumes; for example, **md11** and **md12**.
3. Run the following command to create the first RAID 1 volume:
```
mdadm -C /dev/md11 --level=1 /dev/nvme1n1 /dev/nvme2n1 --raid-devices=2
```
4. Run the following command to create the second RAID 1 volume:
```
mdadm -C /dev/md12 --level=1 /dev/nvme3n1 /dev/nvme4n1 --raid-devices=2
```
5. Select a name for the RAID 10 volume; for example, **md10**.
6. Run the following command to create the RAID 0 of the two RAID 1 volumes, thus creating a RAID 10 volume:
```
mdadm -C /dev/md10 --level=0 /dev/md11 /dev/md12 --raid-devices=2
```
7. Run the following command to apply the disk settings to the volume:
```
for each in md11 md12 md10; do /root/drive_prep_script.sh $each;done
```
8. Run the following command to create a file system on the new volume, named **md0**:
```
mkfs.xfs /dev/md10
```

Create an Eight-Disk RAID 10 Volume

1. Referencing step 12 of the initial configuration section's output, select eight disks that will be part of the RAID array; for example, **nvme1n1**, **nvme2n1**, **nvme3n1**, **nvme4n1**, **nvme5n1**, **nvme6n1**, **nvme7n1**, and **nvme8n1**.
2. Select the user's chosen names for the two component RAID 1 volumes; for example, **md11** and **md12**.
3. Run following command to create the first RAID 1 volume:
```
mdadm -C /dev/md11 --level=1 /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1 /dev/nvme4n1 --raid-devices=4
```
4. Run the following command to create the second RAID 1 volume:
```
mdadm -C /dev/md12 --level=1 /dev/nvme5n1 /dev/nvme6n1 /dev/nvme7n1 /dev/nvme8n1 --raid-devices=4
```
5. Select a name for the RAID 10 volume; for example, **md10**.
6. Run the following command to create the RAID 0 of the two RAID 1 volumes, thus creating a RAID 10 volume:
```
mdadm -C /dev/md10 --level=0 /dev/md11 /dev/md12 --raid-devices=2
```
7. Run the following command to apply the disk settings to the volume:
```
for each in md11 md12 md10; do /root/drive_prep_script.sh $each;done
```
8. Run the following command to create a file system on the new volume, named **md0**:
```
mkfs.xfs /dev/md10
```

Create a Four-Disk RAID 5 Volume

1. Referencing step 12 of the initial configuration section's output, select four disks that will be part of the RAID array; for example, **nvme1n1**, **nvme2n1**, **nvme3n1**, and **nvme4n1**.
2. Select a name for the new RAID volume; for example, **md5**.
3. Run the following command to create the RAID 5 volume:
```
mdadm --create /dev/md5 --level=5 /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1 /dev/nvme4n1--raid-devices=4
```

4. Run the following command to apply settings to the volume:
```
/root/drive_prep_script.sh /dev/md5
```
5. Run the following command to apply a file system to the volume:
```
mkfs.xfs /dev/md5
```

Create a Six-Disk RAID 5 Volume

1. Referencing step 12 of the initial configuration section's output, select six disks that will be part of the RAID array; for example, **nvme1n1**, **nvme2n1**, **nvme3n1**, **nvme4n1**, **nvme5n1**, and **nvme6n1**.
2. Select a name for the new RAID volume; for example, **md5**.
3. Run the following command to create the RAID 5 volume:
```
mdadm --create /dev/md5 --level=5 /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1 /dev/nvme4n1 /dev/nvme5n1 /dev/
nvme6n1--raid-devices=6
```
4. Run the following command to apply settings to the volume:
5. /root/drive_prep_script.sh /dev/md5
6. Run the following command to apply a file system to the volume:
```
mkfs.xfs /dev/md5
```

Hardware RAID Preparation and Configuration

This section covers the steps for creating RAID volumes on Red Hat Enterprise Linux and Windows Server 2022 using iDRAC storage virtual disk creation tools. Between each RAID configuration test, delete the RAID volume group in iDRAC.

Create a One-Disk RAID 0 Volume

1. Log in to iDRAC.
2. Select **Storage** > **Overview**.
3. Select the first RAID controller.
4. From the **Actions** drop-down menu, select **Create Virtual Disk**.
   a. On the **Set up virtual disk** page, select or enter the following:
      i. **Name: Data**
      ii. **Controller: PERC H965i Front (Embedded)**
      iii. **Layout: RAID-0**
      iv. **Media Type: SSD**
      v. **Physical Disk Selection: New Group**
      vi. **Security: Disabled**
      vii. **Stripe Element Size: 64 KB**
      viii. **Read Policy: No Read Ahead**
      ix. **Write Policy: Force Write Back**
      x. **Disk Cache Policy: Default**
   b. Select the **Physical Disk** page, select one disk, and then click **Next**.
   c. On the **Virtual Disk Settings** page, click **Next**.
   d. On the **Confirmation** page, click **Add to Pending**.
   e. Click **Apply Now**.
   f. On the **Information** page, click **OK**.

Create a Two-Disk RAID 0 Volume

1. Log in to iDRAC.
2. Select **Storage** > **Overview**.
3. Select the first RAID controller.
4. From the **Actions** drop-down menu, select **Create Virtual Disk**.
   a. On the **Set up virtual disk** page, select or enter the following:
      i. **Name: Data**
      ii. **Controller: PERC H965i Front (Embedded)**
      iii. **Layout: RAID-0**
      iv. **Media Type: SSD**
      v. **Physical Disk Selection: New Group**
      vi. **Security: Disabled**

vii. **Stripe Element Size: 64 KB**
viii. **Read Policy: No Read Ahead**
ix. **Write Policy: Force Write Back**
x. **Disk Cache Policy: Default**

b. Select the **Physical Disk** page, select two disks, and then click **Next**.
c. On the **Virtual Disk Settings** page, click **Next**.
d. On the **Confirmation** page, click **Add to Pending**.
e. Click **Apply Now**.
f. On the **Information** page, click **OK**.

Create a Four-Disk RAID 0 Volume

1. Log in to iDRAC.
2. Select **Storage** > **Overview**.
3. Select the first RAID controller.
4. From the **Actions** drop-down menu, select **Create Virtual Disk**.
   a. On the **Set up virtual disk** page, select or enter the following:
      i. **Name: Data**
      ii. **Controller: PERC H965i Front (Embedded)**
      iii. **Layout: RAID-0**
      iv. **Media Type: SSD**
      v. **Physical Disk Selection: New Group**
      vi. **Security: Disabled**
      vii. **Stripe Element Size: 64 KB**
      viii. **Read Policy: No Read Ahead**
      ix. **Write Policy: Force Write Back**
      x. **Disk Cache Policy: Default**
   b. Select the **Physical Disk** page, select four disks, and then click **Next**.
   c. On the **Virtual Disk Settings** page, click **Next**.
   d. On the **Confirmation** page, click **Add to Pending**.
   e. Click **Apply Now**.
   f. On the **Information** page, click **OK**.

Create an Eight-Disk RAID 0 Volume

1. Log in to iDRAC.
2. Select **Storage** > **Overview**.
3. Select the first RAID controller.
4. From the **Actions** drop-down menu, select **Create Virtual Disk**.
   a. On the **Set up virtual disk** page, select or enter the following:
      i. **Name: Data**
      ii. **Controller: PERC H965i Front (Embedded)**
      iii. **Layout: RAID-0**
      iv. **Media Type: SSD**
      v. **Physical Disk Selection: New Group**
      vi. **Security: Disabled**
      vii. **Stripe Element Size: 64 KB**
      viii. **Read Policy: No Read Ahead**
      ix. **Write Policy: Force Write Back**
      x. **Disk Cache Policy: Default**
   b. Select the **Physical Disk** page, select eight disks, and then click **Next**.
   c. On the **Virtual Disk Settings** page, click **Next**.
   d. On the **Confirmation** page, click **Add to Pending**.
   e. Click **Apply Now**.
   f. On the **Information** page, click **OK**.

Create a Four-Disk RAID 10 Volume

1. Log in to iDRAC.
2. Select **Storage** > **Overview**.
3. Select the first RAID controller.
4. From the **Actions** drop-down menu, select **Create Virtual Disk**.
   a. On the **Set up virtual disk** page, select or enter the following:
      i. **Name: Data**
      ii. **Controller: PERC H965i Front (Embedded)**
      iii. **Layout: RAID-10**
      iv. **Media Type: SSD**
      v. **Physical Disk Selection: New Group**
      vi. **Security: Disabled**
      vii. **Stripe Element Size: 64 KB**
      viii. **Read Policy: No Read Ahead**
      ix. **Write Policy: Force Write Back**
      x. **Disk Cache Policy: Default**
   b. Select the **Physical Disk** page, select four disks, and then click **Next**.
   c. On the **Virtual Disk Settings** page, click **Next**.
   d. On the **Confirmation** page, click **Add to Pending**.
   e. Click **Apply Now**.
   f. On the **Information** page, click **OK**.

Create an Eight-Disk RAID 10 Volume

1. Log in to iDRAC.
2. Select **Storage** > **Overview**.
3. Select the first RAID controller.
4. From the **Actions** drop-down menu, select **Create Virtual Disk**.
   a. On the **Set up virtual disk** page, select or enter the following:
      i. **Name: Data**
      ii. **Controller: PERC H965i Front (Embedded)**
      iii. **Layout: RAID-10**
      iv. **Media Type: SSD**
      v. **Physical Disk Selection: New Group**
      vi. **Security: Disabled**
      vii. **Stripe Element Size: 64 KB**
      viii. **Read Policy: No Read Ahead**
      ix. **Write Policy: Force Write Back**
      x. **Disk Cache Policy: Default**
   b. Select the **Physical Disk** page, select eight disks, and then click **Next**.
   c. On the **Virtual Disk Settings** page, click **Next**.
   d. On the **Confirmation** page, click **Add to Pending**.
   e. Click **Apply Now**.
   f. On the **Information** page, click **OK**.

Create a Four-Disk RAID 5 Volume

1. Log in to iDRAC.
2. Select **Storage** > **Overview**.
3. Select the first RAID controller.
4. From the **Actions** drop-down menu, select **Create Virtual Disk**.
   a. On the **Set up virtual disk** page, select or enter the following:
      i. **Name: Data**
      ii. **Controller: PERC H965i Front (Embedded)**
      iii. **Layout: RAID-5**
      iv. **Media Type: SSD**
      v. **Physical Disk Selection: New Group**

   vi. **Security: Disabled**
   vii. **Stripe Element Size: 64 KB**
   viii. **Read Policy: No Read Ahead**
   ix. **Write Policy: Force Write Back**
   x. **Disk Cache Policy: Default**
   b. Select the **Physical Disk** page, select four disks, and then click **Next**.
   c. On the **Virtual Disk Settings** page, click **Next**.
   d. On the **Confirmation** page, click **Add to Pending**.
   e. Click **Apply Now**.
   f. On the **Information** page, click **OK**.

Create a Six-Disk RAID 5 Volume

1. Log in to iDRAC.
2. Select **Storage** > **Overview**.
3. Select the first RAID controller.
4. From the **Actions** drop-down menu, select **Create Virtual Disk**.
   a. On the **Set up virtual disk** page, select or enter the following:
      i. **Name: Data**
      ii. **Controller: PERC H965i Front (Embedded)**
      iii. **Layout: RAID-5**
      iv. **Media Type: SSD**
      v. **Physical Disk Selection: New Group**
      vi. **Security: Disabled**
      vii. **Stripe Element Size: 64 KB**
      viii. **Read Policy: No Read Ahead**
      ix. **Write Policy: Force Write Back**
      x. **Disk Cache Policy: Default**
   b. Select the **Physical Disk** page, select six disks, and then click **Next**.
   c. On the **Virtual Disk Settings** page, click **Next**.
   d. On the **Confirmation** page, click **Add to Pending**.
   e. Click **Apply Now**.
   f. On the **Information** page, click **OK**.

Fio Testing

This section outlines the steps taken to run a fio workload. These steps apply to all of the previously listed software and hardware RAID configurations.

1. Referencing the information from step 1 of the **Initial Configuration** section, note if your drives are associated with NUMA node 0, 1, or split across both. This will correspond to a value for **$NUMA_NODE** of "0," "1," or "0,1," respectively.
2. Take note of the volume name chosen for the volume in step 2 of the selected RAID creation section, henceforth **$VOLUME_ID**; for example, **md0**.
3. Determine the full path to the fio configuration files for the test from step 12 of the **Server Setup**; for example, **/root/fio_configs/Seq_Read**, or **/root/fio_configs/Random_Mix**, henceforth **$CONFIG_PATH**.
4. Choose a name for the folder that will house the results from the test run, henceforth **$OUTPUT_PATH**.

Sequential Access Tests Optimal

1. Run the following command to precondition the volume:

```
fio --name=write_twice --filename=/dev/$VOLUME_ID --size=100% --rw=write --ioengine=libaio --iodepth=64
--direct=1 --blocksize=4k --numjobs=4 --loops=2
```

2. Run the following command to start monitoring and test the workload:

```
/root/fio_runner.sh -s -t $CONFIG_PATH -v /dev/$VOLUME_ID -n $NUMA_NODE -o ./$OUTPUT_PATH
```

Random Access Tests Optimal

1. Run the following command to precondition the volume:

```
fio --name=write_twice --filename=/dev/$VOLUME_ID --size=100% --rw=randwrite --ioengine=libaio --iodepth=64
--direct=1 --blocksize=4k --numjobs=4 --loops=2
```

2. Run the following command to start monitoring and test the workload:

```
/root/fio_runner.sh -r -t $CONFIG_PATH -v /dev/$VOLUME_ID -n $NUMA_NODE -o ./$OUTPUT_PATH
```

Sequential Access Tests Degraded, Software RAID

1. Run the following command to identify the NVMe disk ID of the disk to be removed from the RAID array to trigger the degraded state, henceforth **$DISK_TO_REMOVE**:

```
cat /proc/mdstat
```

2. Run the following commands to degrade the RAID array:

```
mdadm --manage /dev/$VOLUME_ID --fail /dev/$DISK_TO_REMOVE
mdadm --manage /dev/$VOLUME_ID --remove /$DISK_TO_REMOVE
mdadm --zero-superblock /dev/$DISK_TO_REMOVE
mdadm --detail /dev/$VOLUME_ID
```

3. Run the following command to start monitoring and test the workload:

```
/root/fio_runner.sh -s -t $CONFIG_PATH -v /dev/$VOLUME_ID -n $NUMA_NODE -o ./$OUTPUT_PATH
```

Random Access Tests Degraded, Software RAID

1. Run the following command to identify the NVMe disk ID of the disk to be removed from the RAID array to trigger the degraded state, henceforth **$DISK_TO_REMOVE**:

```
cat /proc/mdstat
```

2. Run the following to degrade the RAID array:

```
mdadm --manage /dev/$VOLUME_ID --fail /dev/$DISK_TO_REMOVE
mdadm --manage /dev/$VOLUME_ID --remove /$DISK_TO_REMOVE
mdadm --zero-superblock /dev/$DISK_TO_REMOVE
mdadm --detail /dev/$VOLUME_ID
```

3. Run the following command to start monitoring and test the workload:

```
/root/fio_runner.sh -r -t $CONFIG_PATH -v /dev/$VOLUME_ID -n $NUMA_NODE -o ./$OUTPUT_PATH
```

Sequential Access Tests Performance Under Rebuild, Software RAID

1. Run the following command to verify what volume raid_kick.sh is referencing:

```
cat /root/raid_kick.sh
```

2. Run the following command to update the raid_kick.sh script and update the drive path to reference the md5 or md10 volume:

```
vim /root/raid_kick.sh
```

1. Press **:wq <enter>** to save and exit the file.
2. Run the following command to trigger a workload run that will continue running until the RAID array is rebuilt to an optimal state:

```
/root/rebuild_runner.sh -s -t $CONFIG_PATH -v /dev/$VOLUME_ID -n $NUMA_NODE -o ./$OUTPUT_PATH
```

Random Access Tests Performance Under Rebuild, Software RAID

1. Run the following command to verify what volume raid_kick.sh is referencing:

```
cat /root/raid_kick.sh
```

2. Run the following command to update the raid_kick.sh script and update the drive path to reference the md5 or md10 volume:

```
vim /root/raid_kick.sh
```

3. Press **:wq <enter>** to save and exit the file.
4. Run the following command to trigger a workload run that will continue running until the RAID array is rebuilt to an optimal state:

```
/root/rebuild_runner.sh -r -t $CONFIG_PATH -v /dev/$VOLUME_ID -n $NUMA_NODE -o ./$OUTPUT_PATH
```

Sequential Access Tests Degraded, Hardware RAID

1. During the RAID rebuild, perform the following tasks:
   a. Set the disk as offline.
   b. For each RAID configuration, RAID 5 and RAID 10, update the controller and disk identifiers in the PERCCLI2 command:

```
/opt/MegaRAID/perccli/perccli2 /c1/e304/s7 set offline
```

2. Run the following command to start monitoring and test the workload:

```
/root/fio_runner.sh -s -t $CONFIG_PATH -v /dev/$VOLUME_ID -n $NUMA_NODE -o ./$OUTPUT_PATH
```

Random Access Tests Degraded, Hardware RAID

1. During the RAID rebuild, perform the following tasks:
   a. Set the disk as offline.
   b. For each RAID configuration, RAID 5 and RAID 10, update the controller and disk identifiers in the PERCCLI2 command:

```
/opt/MegaRAID/perccli/perccli2 /c1/e304/s7 set offline
```

2. Run the following command to start monitoring and test the workload:

```
/root/fio_runner.sh -r -t $CONFIG_PATH -v /dev/$VOLUME_ID -n $NUMA_NODE -o ./$OUTPUT_PATH
```

Sequential Access Tests Rebuilding, Hardware RAID

1. During the RAID rebuild, perform the following tasks:
   a. Set the disk as offline.
   b. Set the disk as online and rebuilding.
   c. Verify the status of the rebuild.
   d. For each RAID configuration, RAID 5 and RAID 10, update the controller and disk identifiers in the PERCCLI2 command:

```
/opt/MegaRAID/perccli/perccli2 /c1/e304/s7 set offline
/opt/MegaRAID/perccli/perccli2 /c1/e304/s7 start rebuild
/opt/MegaRAID/perccli/perccli2 /c1/e304/s7 show rebuild
```

2. Run the following command to trigger a workload run that will keep the RAID array in a rebuilding state:

```
/root/fio_runner_rebuilding.sh -s -t $CONFIG_PATH -v /dev/$VOLUME_ID -n $NUMA_NODE -o ./$OUTPUT_PATH
```

Random Access Tests Rebuilding, Hardware RAID

1. During the RAID rebuild, perform the following tasks:
   a. Set the disk as offline.
   b. Set the disk as online and rebuilding.
   c. Verify the status of the rebuild.
   d. For each RAID configuration, RAID 5 and RAID 10, update the controller and disk identifiers in the PERCCLI2 command:

```
/opt/MegaRAID/perccli/perccli2 /c1/e304/s7 set offline
/opt/MegaRAID/perccli/perccli2 /c1/e304/s7 start rebuild
/opt/MegaRAID/perccli/perccli2 /c1/e304/s7 show rebuild
```

2. Run the following command to trigger a workload run that will keep the RAID array in a rebuilding state:

```
/root/fio_runner_rebuilding.sh -r -t $CONFIG_PATH -v /dev/$VOLUME_ID -n $NUMA_NODE -o ./$OUTPUT_PATH
```

HammerDB Testing

This section outlines the steps taken to install and configure Microsoft SQL Server 2022 on the SUT, utilizing a RAID 5 volume for the data and a RAID 0 volume for the logs, and utilizing HammerDB to generate a TPC-C–like workload. The HammerDB workload was generated from a jump box via the GUI.

Server under test preparation

1. Run the following command to switch to the root user:

```
sudo su
```

2. Referencing the **Hardware RAID Preparation and Configuration section**, create either a 4- or a 6-disk RAID 5 volume to be used as the data volume.
3. Referencing the same section, create a 2-disk RAID 1 volume to be used as the log volume.
4. Run the following command to create the folders to which the volumes will be mounted:

```
mkdir /ms_sql_logs; mkdir /ms_sql_data
```

5. Run the following command to edit the fstab:

```
vim /etc/fstab
```

6. Add the following contents:

```
/dev/md0                /ms_sql_logs            xfs     defaults        0 0
/dev/md5                /ms_sql_data            xfs    rw,attr2,noatime        0 0
```

7. Press :wq <enter> to save the changes and exit the file.
8. Run the following command to mount the volumes:

```
mount -a
```

9. Run the following commands to set system tunning:

```
systemctl enable tuned
dnf install tuned-profiles-mssql
tuned-adm profile mssql
systemctl stop firewalld
systemctl disable firewalld
```

10. Run the following command to modify SELINUX configuration:

```
vim /etc/selinux/
```

11. Modify the below setting in the file:

```
SELINUX=disabled
```

12. Press **:wq <enter>** to save the changes and exit the file.
13. Run the following command to open the HammerDB automation file for editing:

```
vim /root/HammerRunner.sh
```

14. Paste in the contents of the of the **HammerRunner appendix**.
15. Press **:wq <enter>** to save the changes and exit.
16. Run the following command to grant the execute permission:

```
chmod +x /root/HammerRunner.sh
```

17. To install Microsoft SQL Server run the following commands:

```
curl -o /etc/yum.repos.d/mssql-server.repo https://packages.microsoft.com/config/rhel/9/mssql-server-2022.repo
yum install -y mssql-server
/opt/mssql/bin/mssql-conf setup
```

18. Use the following parameters when prompted:
    a. **Version: 1**
    b. **License Agreement Terms: Yes**
    c. **SQL Account Password: $SQL_PASSWORD** (NOTE: Will be used for future connection to the SQL Server)
19. Run the following commands to install additional tools:

```
curl https://packages.microsoft.com/config/rhel/9/prod.repo | sudo tee /etc/yum.repos.d/mssql-release.repo
yum install -y mssql-tools18 unixODBC-devel
```

20. Use the following parameters when prompted:
    a. **License Agreement Terms: Yes**
21. Run the following commands to add the Microsoft SQL Server tools to the user's path:

```
echo 'export PATH="$PATH:/opt/mssql-tools18/bin"' >> ~/.bash_profile
echo 'export PATH="$PATH:/opt/mssql-tools18/bin"' >> ~/.bashrc
```

22. Run the following command to update the permissions on the data and log folders:

```
chown mssql:mssql /ms_sql_*
```

23. Run the following commands to modify the Microsoft SQL Server configuration settings:

```
/opt/mssql/bin/mssql-conf set filelocation.defaultdatadir /ms_sql_data/
/opt/mssql/bin/mssql-conf set filelocation.defaultlogdir /ms_sql_logs/
/opt/mssql/bin/mssql-conf traceflag 3979 on
/opt/mssql/bin/mssql-conf set control.writethrough 1
/opt/mssql/bin/mssql-conf set control.alternatewritethrough 0
```

24. Run the following command to restart the Microsoft SQL service and apply the changes

```
systemctl restart mssql-server.service
```

25. Run the following command to create a script that will optimize disk settings:

```
vim /root/mssql_disk_opt.sh
```

26. Add the below to the file:

```
echo none > /sys/block/$1/queue/scheduler
echo 0 > /sys/block/$1/queue/add_random
echo 2 > /sys/block/$1/queue/nomerges
echo 2048 > /sys/block/$1/queue/nr_requests
echo 0 > /sys/block/$1/queue/rotational
echo 2 > /sys/block/$1/queue/rq_affinity
echo 1024 > /sys/block/$1/queue/max_sectors_kb
echo 1024 > /sys/block/$1/device/queue_depth
echo 1 > /sys/block/$1/queue/iostats
```

27. Press **:wq <enter>** to save and exit the file.
28. Run the following command to grant the execute permission:

```
chmod +x /root/mssql_disk_opt.sh
```

29. Run the following command to apply optimizations; invalid arguments warnings may be ignored:

```
./mssql_disk_opt.sh md5; ./mssql_disk_opt.sh md0
```

30. Run the following command to set memory settings for Microsoft SQL Server:

```
vim /etc/security/limits.d/99-mssql-server.com
```

31. Add the below to the file:

```
mssql hard nofile 32727
mssql soft nofile 16000
```

32. Press **:wq <enter>** to save and exit the file.
33. Run the following command to set additional settings for Microsoft SQL Server:

```
vim /etc/sysctl.d/99-mssqls.conf
```

34. Add the below to the file:

```
kernel.sched_min_granularity_ns = 15000000
kernel.sched_wakeup_granularity_ns = 2000000
vm.dirty_ratio = 80
vm.dirty_background_ratio = 3
vm.swappiness = 1
```

35. Press **:wq <enter>** to save and exit the file.
36. Run the following command to apply the changes:

```
sysctl -p /etc/sysctl.d/99-mssqls.conf
```

37. Run the following command in lieu of pinning the workload to a specific NUMA node:

```
sysctl -w kernel.numa_balancing=0
```

38. Run the below to install hammerdb:

```
wget https://github.com/TPC-Council/HammerDB/releases/download/v4.10/HammerDB-4.10-Linux.tar.gz
tar -xvzf HammerDB-4.10-Linux.tar.gz
```

39. Run the following command to enter the HammerDB directory:

```
cd /HammerDb-4.10
```

40. Run the following command to open the HammerDB builder file for editing:

```
vim ./builder.tcl
```

41. Paste in the contents of the of the **builder.tcl appendix**.
42. Press **:wq <enter>** to save the changes and exit.
43. Run the following command to open the Microsoft SQL Server command line interface (CLI) interface:

```
sqlcmd -s localhost -U sa -P $SQL_PASSWORD -C
```

44. Run the following commands to modify the temporary database files:

```
ALTER DATABASE tempdb MODIFY FILE (NAME = tempdev, FILENAME = '/ms_sql_logs/tempdb01.mdf', SIZE = 1024,
FILEGROWTH = 8192MB)
go
ALTER DATABASE tempdb MODIFY FILE (NAME = tempdev2, FILENAME = '/ms_sql_logs/tempdb02.ndf', SIZE = 1024,
FILEGROWTH = 8192MB)
go
ALTER DATABASE tempdb MODIFY FILE (NAME = tempdev3, FILENAME = '/ms_sql_logs/tempdb03.ndf', SIZE = 1024,
FILEGROWTH = 8192MB)
go
ALTER DATABASE tempdb MODIFY FILE (NAME = tempdev4, FILENAME = '/ms_sql_logs/tempdb04.ndf', SIZE = 1024,
FILEGROWTH = 8192MB)
go
ALTER DATABASE tempdb MODIFY FILE (NAME = tempdev5, FILENAME = '/ms_sql_logs/tempdb05.ndf', SIZE = 1024,
FILEGROWTH = 8192MB)
go
ALTER DATABASE tempdb MODIFY FILE (NAME = tempdev6, FILENAME = '/ms_sql_logs/tempdb06.ndf', SIZE = 1024,
FILEGROWTH = 8192MB)
go
ALTER DATABASE tempdb MODIFY FILE (NAME = tempdev7, FILENAME = '/ms_sql_logs/tempdb07.ndf', SIZE = 1024,
```

```
FILEGROWTH = 8192MB)
go
ALTER DATABASE tempdb MODIFY FILE (NAME = tempdev8, FILENAME = '/ms_sql_logs/tempdb08.ndf', SIZE =
1024, FILEGROWTH = 8192MB)
go
```

45. Run the following command to exit Microsoft SQL Server CLI:

```
quit
```

46. Run the following command to restart and apply changes to the MS SQL service:

```
systemctl restart mssql-server.service
```

47. Run the following command to create the 2500 warehouse DB:

```
export TMP=/ms_sql_logs/ ; time ./hammerdbcli auto /root/hammerScripts/builder.tcl
```

48. Run the following command to modify the database restore options:

```
sqlcmd -S localhost -U SA -P $SQL_PASSWORD -C -Q "ALTER DATABASE TPCC_2500 SET RECOVERY SIMPLE"
```

49. Run the following command to create a backup of the database:

```
sqlcmd -S localhost -U SA  -C -Q "BACKUP DATABASE [TPCC_2500] TO DISK = N'/ms_sql_logs/cleanBackup2500.
bak' WITH NOFORMAT, NOINIT, NAME = 'TPCC_2500', SKIP, NOREWIND, NOUNLOAD, STATS = 10"
```

50. Run the following command and note the IP address, henceforth $MS_SQL_IP:

```
ifconfig
```

51. From the jump box, open the HammerDB GUI.
52. Select **SQL Server** > **TPROC-C** > **Driver Script** > **Options** and set the below options:
    a. **SQL Server: $MS_SQL_IP**
    b. **Authentication: SQL Server**
    c. **Password: $SQL_PASSWORD**
    d. **TPROC-C Driver Script: Timed Driver Script**
    e. **Total Transactions per User: 10000000**
    f. **Minutes of rampup: 7**
    g. **Minutes for Test Duration: 20**
    h. **Use all warehouses: yes**
53. Click **OK** to close the Driver options window.
54. Select **SQL Server** > **TPROC-C** > **Virtual Users** > **Options** and set the below options:
    a. **Log Output to Temp: true**
    b. **Use Unique Log Name: true**
    c. **Log Timestamps: True**
55. Click **OK** to close the window.

HammerDB test Optimal

On the instance under test, ensure that the Degrade RAID and Rebuild RAID sections of hammerRunner.sh are commented out.

2. Choose a name for the results folder, henceforth $RunID.
3. Run the following command to start the database restore:

```
./hammerRunner.sh ./$RunID
```

4. When prompted after the restore has completed press **<enter>** to start the monitoring.
5. On the jump box, open the HammerDB GUI.
6. Select **SQL Server** > **TPROC-C** > **Virtual Users** > **Options** and set **Virtual Users** to **50**.
    a. Click **OK** to close the window.
    b. Click the **Play** button to start the run.
    c. Upon completion, press **Stop**.
7. Select **SQL Server** > **TPROC-C** > **Virtual Users** > **Options** and set **Virtual Users** to **100**.
    a. Click **OK** to close the window.
    b. Click the **Play** button to start the run.
    c. Upon completion, press **Stop**.
8. Select **SQL Server** > **TPROC-C** > **Virtual Users** > **Options** and set **Virtual Users** to **500**.
    a. Click **OK** to close the window.
    b. Click the **Play** button to start the run.
    c. Upon completion, press **Stop**.
9. Select **SQL Server** > **TPROC-C** > **Virtual Users** > **Options** and set **Virtual Users** to **800**.
    a. Click **OK** to close the window.

   b. Click the **Play** button to start the run.
   c. Upon completion, press **Stop**.
10. On the SUT, run the following command to stop data collection:

```
killall nmon; killall iostat; echo "]}]}}" >> $RunID/iostat.out
```

11. Copy the HammerDB results file from the jump box to the $RunID folder.

HammerDB test Degraded

On the instance under test, ensure that the Degrade RAID is **not** commented out and that the Rebuild RAID section of hammerRunner.sh **is** commented out.

1. Choose a name for the results folder, henceforth $RunID.
2. Run the following command to start the database restore:

```
./hammerRunner.sh ./$RunID
```

3. When prompted after the restore has completed, press **<enter>** to start the monitoring.
4. On the jump box, open the HammerDB GUI.
5. Select **SQL Server** > **TPROC-C** > **Virtual Users** > **Options** and set **Virtual Users** to **50**.
   a. Click **OK** to close the window.
   b. Click the **Play** button to start the run.
   c. Upon completion, press **Stop**.
6. Select **SQL Server** > **TPROC-C** > **Virtual Users** > **Options** and set **Virtual Users** to **100**.
   a. Click **OK** to close the window.
   b. Click the **Play** button to start the run.
   c. Upon completion, press **Stop**.
7. Select **SQL Server** > **TPROC-C** > **Virtual Users** > **Options** and set **Virtual Users** to **500**.
   a. Click **OK** to close the window.
   b. Click the **Play** button to start the run.
   c. Upon completion, press **Stop**.
8. Select **SQL Server** > **TPROC-C** > **Virtual Users** > **Options** and set **Virtual Users** to **800**.
   a. Click **OK** to close the window.
   b. Click the **Play** button to start the run.
   c. Upon completion, press **Stop**.
9. On the SUT, run the following command to stop the metrics collection:

```
killall nmon; killall iostat; echo "]}]}}" >> $RunID/iostat.out
```

10. Copy the HammerDB results file from the jump box to the $RunID folder.

HammerDB test Rebuilding

On the instance under test, ensure that the Degrade RAID and Rebuild RAID sections of hammerRunner.sh are not commented out.

1. Choose a name for the results folder, henceforth $RunID.
2. Run the following command to start the database restore:

```
./hammerRunner.sh ./$RunID
```

3. When prompted after the restore has completed, press <enter> to start the monitoring.
4. On the jump box, open the HammerDB GUI.
5. Select **SQL Server** > **TPROC-C** > **Virtual Users** > **Options** and set **Virtual Users** to **50**.
   a. Click **OK** to close the window.
   b. Click the **Play** button to start the run.
   c. Upon completion, press **Stop**.
6. Select **SQL Server** > **TPROC-C** > **Virtual Users** > **Options** and set **Virtual Users** to **100**.
   a. Click **OK** to close the window.
   b. Click the **Play** button to start the run.
   c. Upon completion, press **Stop**.
7. Select **SQL Server** > **TPROC-C** > **Virtual Users** > **Options** and set **Virtual Users** to **500**.
   a. Click **OK** to close the window.
   b. Click the **Play** button to start the run.
   c. Upon completion, press **Stop**.
8. Select **SQL Server** > **TPROC-C** > **Virtual Users** > **Options** and set **Virtual Users** to **800**.
   a. Click **OK** to close the window.

   b. Click the **Play** button to start the run.
   c. Upon completion, press **Stop**.
9. On the SUT, run the following command to stop the metrics collection:

```
killall nmon; killall iostat; echo "]}]}}" >> $RunID/iostat.out
```

10. Copy the HammerDB results file from the jump box to the $RunID folder.

Windows Server® 2022

Use the following instructions to install Windows Server 2022 on the software RAID– and hardware RAID–based servers.

1. Open the iDRAC terminal.
2. Click **virtual media**.
3. Select the Windows Server 2022 ISO to **Map CD/DVD**.
4. Click **Map Device**.
5. Reboot the server under test (SUT).
6. When prompted, press any key to boot to CD/DVD.
7. Leave **English** selected as the **Language to install** and **Time and currency format**, as well as **US** for the keyboard or input method.
8. Click **Next**.
9. Click **Install Now**.
10. Select **Windows Server 2022 Datacenter Evaluation Desktop Experience**.
11. Click **Next**.
12. Select the checkbox to accept the license agreement.
13. Click **Next**.
14. Select **Custom: Install Microsoft Server Operating system only**.
15. Select the correct volume to install the operating system on (the Dell™ BOSS card, which is a 900 GB volume).
16. When prompted for Windows to create needed partitions, click **OK**.
17. Click **Next**.
18. When prompted, enter and confirm the **Administrator password**.
19. Log in to the fresh Windows installation.
20. From the Server Manager window's local server view, beside **Remote desktop**, click the **disabled** link.
21. Select **Allow remote connections to this computer**.
22. When prompted, click **OK** to add the firewall exception.
23. Click **Apply**.
24. Click **OK**.
25. Mount Dell Technologies updates as an ISO.
26. Install Dell Technologies updates.
27. Install drivers.
28. Open Windows update settings:
    a. Check for updates.
    b. Install now.
    c. Restart now.

Software RAID

1. Download Dell™ OpenManage™ Server Administrator: Dell OpenManage Server Administrator Managed Node for Windows, v11.0.1.0.
2. Double-click to extract the files.
3. Navigate to the extracted files, and then, in the **File Explorer** folder, double-click the setup executable.
4. After the installer loads the prerequisite information, click **Install Server Administrator**.
5. Once installation is completed, click **Finish**.
6. Download and install the Dell OpenManage Server Administrator patch: Dell OpenManage Server Administrator Managed Node for Windows, v11.0.1.1.
7. From the desktop, launch **OpenManage Server Administrator**.
8. Log in with local administrator credentials.
9. On the left pane, select **Storage** > **PERC S160(Embedded)** > **Virtual Disks**.
10. Click **Go to the Create A Virtual Disk Wizard**.
11. Select **Advanced Wizard**.

12. In the **RAID Level** dropdown, select the applicable RAID level.
13. In the **Bus Protocol** dropdown, select **PCIe**.
14. In the list of available disks on the connector pane, select the desired drive quantity.
15. Click **Continue**.
16. Enter a name, and then confirm the virtual disk capacity.
17. Click **Finish**.
18. From the **Start** menu, search for and launch **Disk Management**. **Note**: Utilize the following steps on the hardware RAID server to online and create a new volume.
19. Right-click the newly created virtual disk, and then select **Initialize Disk**.
20. Select **GPT**.
21. Click **OK**.
22. Right-click the unallocated partition, and then select **Create New Simple Volume**.
23. Click **Next**.
24. Confirm the volume size, and then click **Next**.
25. Assign the drive letter to the volume, and then click **Next**.
26. Select **Format this volume with the following settings**, and then set the following:
    a. **File System: NTFS**
    b. **Allocation Size: 4096**
    c. **Volume Label:** *<drive name>*
27. Click **Next**, and then click **Finish**.
28. Repeat steps 1–27 to create the additional RAID configurations for testing. Delete the existing RAID array prior to creating the new one.

Windows Server 2022 Iometer Testing

The Prowess Consulting engineers utilized Iometer to test performance on both the hardware and software RAID configurations of Windows Server 2022.

1. Download and extract **Iometer**.
2. For each of the **test parameters**, create an Iometer configuration file. Follow steps in the "**Iometer Users' Guide**" to learn how to create an Iometer configuration file.
3. Create a **PowerShell® script** to run all the tests as prescribed by the type of test: random read, random write, random mix, sequential read, and sequential write.
4. For hardware RAID rebuild testing, create a PowerShell script that will put the RAID into a continuous rebuild state. Download and install **perccli2** for this testing.
5. For hardware RAID degraded testing, run the following command to set a drive offline, and then run the tests as run during the optimal testing:

```
perccli2 /c0/e<enclosure identifier>/s<slot identifier>
```

6. For hardware RAID degraded testing or to set the drive as online, run the following command:

```
perccli2 /c0/e<enclosure identifier>/s<slot identifier> start rebuild
```

7. For software RAID degraded and rebuild testing, the hard drive is physically removed and inserted into the SUT, and the Iometer testing is conducted during those conditions.

HammerDB Testing

The following steps were utilized to install Microsoft SQL Server 2022 and run a HammerDB workload.

1. Download Microsoft SQL Server 2022.
2. Download the installation media in ISO format.
3. Double-click SQLServer2022-x64-ENU.ISO to mount the ISO.
    a. Select **New SQL Server standalone installation or add features to an existing installation**.
    b. On the **Select the edition of SQL Server 2022 you want to install** page, in the **Specify a free edition** drop-down box, leave the default settings and click **Next**.
    c. On the **To install SQL Server 2022, you must accept the Microsoft Software License Terms** page, click **I accept the license terms**, and click **Next**.
    d. On the **Microsoft Update** page, click **Use Microsoft Update to check for updates**, and click **Next**.
    e. On the **Azure Extension for SQL Server** page, unselect **Azure Extension for SQL Server**, and click **Next**.
    f. On the **Feature Selection** page, select the following features:
    g. Database Engine Services

h. Click **Next**.
i. On the **Instance Configuration** page, leave the default instance, and click **Next**.
j. On the **Server Configuration** page, and click **Next**.
k. On the **Database Engine Configuration** page, under **Authentication Mode**, select **Mixed Mode**. Enter and confirm the system administrator account password.
l. Under the **Specify SQL Server administrators** box, click **Add Current User**.
m. From the **Database Engine Configuration** page, select **Data Directories**, and use the following parameters:
   i. **User database directory: <RAID 5 storage>\data**
   ii. **User database log direcotry: <RAID 1 Storage>\logs**
   iii. **Backup Directory: <RAID 5 storage>\backup**
n. From the **Database Engine Configuration** page, select **TempDB**, and use the following parameters:
   i. **Data directories: <RAID1 storage>\tempdb**
   ii. **Log directory: <RAID1 storage>\tempdb**
o. From the **Database Engine Configuration** page, select **MaxDOP**, and use the following parameters:
   i. **Maximum degree of parallelism: Half the amount of logical processors**
p. From the **Database Engine Configuration** page, select **Memory**, and use the following parameters:
   i. **Select Recommended Radio button**
   ii. **Select Click here to accept the recommended memory configurations for the sql server database engine**
q. Click **Next**.
r. On the **Ready to Install** page, click Install.

4. Install ODBC Driver, Download ODBC Driver for SQL Server - ODBC Driver for SQL Server | Microsoft Learn.
5. Install SSMS, Download SQL Server Management Studio (SSMS) - SQL Server Management Studio (SSMS) | Microsoft Learn.
6. From the start menu, search for and launch **Configure ODBC Data source Administrator**.
   a. From **User DSN**, click **Add**.
   b. From the **Create New Data Source** page, select **ODBC Driver 18 for SQL Server** and click **Finish**.
   c. On the **Create a New Data Source to SQL Server** page, use the following parameters:
      i. **Name: Localhost**
      ii. **Server: Localhost**
   d. Click **Next**.
   e. When asked **How should SQL Server verify the authenticity of the login ID**, select **With SQL Server Authentication using a login ID and password entered by the user** and use the following parameters:
      i. **Login ID: sa**
      ii. **Password: <SA Password>**
   f. Click **Next**.
   g. On the **Database Configuration** page, click **Next**.
   h. On the **Connection details** page, select **Trust Server Certificate** and click **Finish**.
   i. Click **Test Data Source**.
   j. Click **OK** to close the ODBC Data source administrator tool.
7. Download Hammer DB version 4.10 for Windows Zip.
8. Extract HammerDB to the desktop.
9. From the HammerDB folder, double-click and launch hammerdb.bat.
10. From HammerDB, double-click **SQL server** and select **TPROC-C**.
11. Expand **Schema**, double-click **Options**, and use the following parameters:
    c. Trust Server Certificate: selected
    d. Authentication: SQL Server
    e. SQL Server User ID: sa
    f. SQL Server User Password: <SA password>
    g. Number of Warehouses: 2500
    h. Virtual Servers to build schema: 100
12. Click **OK**.
13. Double-click **Build** and wait…
14. After the database is created, from the start menu, search for and launch SQL server management studio.
    a. Connect to the local server.

b. Right-click **TPCC database** and select **Properties**.
c. Select the **Options** tab; from Recovery Model, select **Simple** and click **OK**.
d. Right-click **TPCC database**, select **Tasks** > **Backup**.
e. From Back UP Database - tpcc, click **OK**.

15. From HammerDB, expand **Driver Script**.
16. Double-click **Options** and use the following parameters:
    a. Minutes of rampup time: 7
    b. Minutes for test duration: 20
    c. Use All Warehouses: selected
17. Click **OK**.
18. Expand **Autopilot**, double-click **Options**.
    a. From **Autopilot** options, select **Autopilot Enabled**, and update with the following parameters:
        i. Minutes per test in virtual sequence: 30
        ii. Active Virtual User Sequence: 50 100 500 800
        iii. Show virtual User output: selected
        iv. Log virtual user output to temp: selected
        v. Use unique log name: selected
        vi. Log timestamps: selected
19. Click **OK**.
20. Double-click **Autopilot** to start testing and add log volume to data collection.
21. Start PERFMON data collection.

Appendix

These appendices contain the base contents for the scripts referenced in this document. Any adjustments required to be made to the script to account for local variables will be stated in the summary section of the relevant appendix.

Appendix 1: fio-runner

This script is used to start the monitoring processes and to trigger the fio workloads for the optimal and degraded RAID cases.

Software RAID

```
#!/bin/bash
# Define the optstring
optstring="rspt:o:v:n:f:"
# Define the variables to store the option values
random=false
sequential=false
precondition=false
# Parse the command-line arguments
while getopts "$optstring" opt; do
  case "$opt" in
    r) random=true
    testType=random ;;
    s) sequential=true
    testType=sequential;;
    p) precondition=true ;;
    t) test_path="$OPTARG" ;;
    o) requested_output_path="$OPTARG" ;;
    v) volume="$OPTARG" ;;
    n) numa_node="$OPTARG" ;;
  esac
done
# Sanity check arguments
if (("$random" == true && "$sequential" == true ) || ("$random" == false && "$sequential" == false ));
then
    echo "-s OR -r is required, specify one and only one of the two options"
```

```
    exit 1
elif  [[ -z "$test_path" || -z "$requested_output_path" || -z "$volume" || -z "$numa_node" ]]; then
  echo "Error: missing flags!  All below are required
    -t path to test files
    -o directory to create and save output into
    -v volume to be tested
    -n allowed numa nodes"
  exit 1
fi
ts=$(date +%s)
output_path=${requested_output_path}.${ts}
# Prepare files and output path
mkdir -p $output_path
cp $test_path/$testType* $output_path/
escaped_volume=$(sed 's/\//\\\//g' <<< "$volume")
short_volume=`echo $volume | cut -d\/ -f3`
find $output_path -type f -exec sed -i "s/\/dev\/DISK_ID/$escaped_volume/g" {} \;
find $output_path -type f -exec sed -i "s/NUMA_NODE/$numa_node/g" {} \;
sleep 2
# Start the monitors
nmon -F $output_path/nmon.out -s2 -c100000 -t &
# Check if mapped volume or device and start dstat accordingly
if [[ $short_volume == md* ]]; then
    dstat -tam -C total --md -M total,$short_volume --output $output_path/dstat.out &
    d_pid=$!
elif [[ $short_volume == sd* || $variable == nv* ]]; then
    dstat -tam -C total -dD total,$short_volume --output $output_path/dstat.out &
    d_pid=$!
fi
iostat -k -t -o JSON -cdx $volume 2 > $output_path/iostat.out &
sleep 2
# Run the tests
for fio_job in `ls -v $output_path/*.fio`;do
    sleep 5
    fio $fio_job --output=$fio_job.log --output-format=json+
    sleep 10
done
# Cleanup monitors
sleep 180
killall nmon
kill $d_pid
killall iostat
echo "]}]}}" >> $output_path/iostat.out
```

Hardware RAID

```
#!/bin/bash
# Define the optstring
optstring="rspt:o:v:n:f:"
# Define the variables to store the option values
random=false
sequential=false
precondition=false
raid_to_rebuild="/dev/md12"
# Parse the command-line arguments
while getopts "$optstring" opt; do
  case "$opt" in
```

```
    r) random=true
    testType=random ;;
    s) sequential=true
    testType=sequential;;
    p) precondition=true ;;
    t) test_path="$OPTARG" ;;
    o) requested_output_path="$OPTARG" ;;
    v) volume="$OPTARG" ;;
    n) numa_node="$OPTARG" ;;
    f) rebuild="$OPTARG" ;; #### if set to a raid id (ie /dev/md12) it will re-add the drive to the raid
and begin a rebuild and logging
  esac
done
# Sanity check arguments
if (("$random" == true && "$sequential" == true ) || ("$random" == false && "$sequential" == false ));
then
    echo "-s OR -r is required, specify one and only one of the two options"
    exit 1
elif  [[ -z "$test_path" || -z "$requested_output_path" || -z "$volume" || -z "$numa_node" ]]; then
  echo "Error: missing flags!  All below are required
    -t path to test files
    -o directory to create and save output into
    -v volume to be tested
    -n allowed numa nodes"
  exit 1
fi

ts=$(date +%s)
output_path=${requested_output_path}.${ts}
# Fix dstat's csv output, included as a comment in case dstat starts complaining about bad imports with
the csv handler, as with the type/types error. Sanity check the line numbers before using.
#wd=`which dstat`
#cp $wd{,.fixed}
#sudo sed -i '547s/.*/            if isinstance( self.val[ name], (tuple, list)):/' $wd.fixed
#sudo sed -i '552s/.*/            elif isinstance( self.val[ name], str):/' $wd.fixed

# Prepare files and output path
mkdir -p $output_path
cp $test_path/$testType* $output_path/
escaped_volume=$(sed 's/\//\\\//g' <<< "$volume")
short_volume=`echo $volume | cut -d\/ -f3`
find $output_path -type f -exec sed -i "s/\/dev\/DISK_ID/$escaped_volume/g" {} \;
find $output_path -type f -exec sed -i "s/NUMA_NODE/$numa_node/g" {} \;
sleep 2
# Start the monitors

if [ -n "$rebuild" ];then
    echo "Rebuilding $rebuild"
    /root/raid_rebuilder.sh $output_path/rebuild.log &
fi

nmon -F $output_path/nmon.out -s2 -c100000 -t &
# Check if mapped volume or device and start dstat accordingly
if [[ $short_volume == md* ]]; then
    dstat -tam -C total --md -M total,$short_volume --output $output_path/dstat.out &
    d_pid=$!
```

```
elif [[ $short_volume == sd* || $variable == nv* ]]; then
    dstat -tam -C total -dD total,$short_volume --output $output_path/dstat.out &
    d_pid=$!
fi
iostat -k -t -o JSON -cdx $volume 2 > $output_path/iostat.out &
sleep 2
# Run the tests
for fio_job in `ls -v $output_path/*.fio`;do
    ##### Only uncomment to keep RAID in a rebuilding state
#   cat /proc/mdstat >> $output_path/rebuilding_state.log
#   /root/raid_kick.sh
#   date >> $output_path/rebuilding_state.log
#   cat /proc/mdstat >> $output_path/rebuilding_state.log
#   echo "Starting run $fio_job" >> $output_path/rebuilding_state.log
    #### End rebuilding section
    sleep 5
    fio $fio_job --output=$fio_job.log --output-format=json+
    sleep 10

done
# Cleanup monitors
killall nmon
kill $d_pid
killall iostat
echo "]}]}}" >> $output_path/iostat.out

Appendix 2: fio_runner_rebuilding
This script is used to trigger the monitoring and fio workloads while ensuring that the RAID array is in a
freshly rebuilding state prior to each test.

Software RAID
#!/bin/bash
# Define the optstring
optstring="rspt:o:v:n:f:"
# Define the variables to store the option values
random=false
sequential=false
precondition=false
# Parse the command-line arguments
while getopts "$optstring" opt; do
  case "$opt" in
    r) random=true
    testType=random ;;
    s) sequential=true
    testType=sequential;;
    p) precondition=true ;;
    t) test_path="$OPTARG" ;;
    o) requested_output_path="$OPTARG" ;;
    v) volume="$OPTARG" ;;
    n) numa_node="$OPTARG" ;;
  esac
done
# Sanity check arguments
if (("$random" == true && "$sequential" == true ) || ("$random" == false && "$sequential" == false ));
then
    echo "-s OR -r is required, specify one and only one of the two options"
```

```
    exit 1
elif  [[ -z "$test_path" || -z "$requested_output_path" || -z "$volume" || -z "$numa_node" ]]; then
  echo "Error: missing flags!  All below are required
    -t path to test files
    -o directory to create and save output into
    -v volume to be tested
    -n allowed numa nodes"
  exit 1
fi
ts=$(date +%s)
output_path=${requested_output_path}.${ts}
# Prepare files and output path
mkdir -p $output_path
cp $test_path/$testType* $output_path/
escaped_volume=$(sed 's/\//\\\//g' <<< "$volume")
short_volume=`echo $volume | cut -d\/ -f3`
find $output_path -type f -exec sed -i "s/\/dev\/DISK_ID/$escaped_volume/g" {} \;
find $output_path -type f -exec sed -i "s/NUMA_NODE/$numa_node/g" {} \;
sleep 2
# Start the monitors
nmon -F $output_path/nmon.out -s2 -c100000 -t &
# Check if mapped volume or device and start dstat accordingly
if [[ $short_volume == md* ]]; then
    dstat -tam -C total --md -M total,$short_volume --output $output_path/dstat.out &
    d_pid=$!
elif [[ $short_volume == sd* || $variable == nv* ]]; then
    dstat -tam -C total -dD total,$short_volume --output $output_path/dstat.out &
    d_pid=$!
fi
iostat -k -t -o JSON -cdx $volume 2 > $output_path/iostat.out &
sleep 2
# Run the tests
for fio_job in `ls -v $output_path/*.fio`;do
    cat /proc/mdstat >> $output_path/rebuilding_state.log
    /root/raid_kick.sh
    date >> $output_path/rebuilding_state.log
    cat /proc/mdstat >> $output_path/rebuilding_state.log
    echo "Starting run $fio_job" >> $output_path/rebuilding_state.log
    sleep 5
    fio $fio_job --output=$fio_job.log --output-format=json+
    sleep 10
done
# Cleanup monitors
killall nmon
kill $d_pid
killall iostat
echo "]}]}}" >> $output_path/iostat.out
```

Hardware RAID

```
#!/bin/bash
# Define the optstring
optstring="rspt:o:v:n:f:"
# Define the variables to store the option values
random=false
sequential=false
precondition=false
```

```
raid_to_rebuild="/dev/sda"
# Parse the command-line arguments
while getopts "$optstring" opt; do
  case "$opt" in
    r) random=true
    testType=random ;;
    s) sequential=true
    testType=sequential;;
    p) precondition=true ;;
    t) test_path="$OPTARG" ;;
    o) requested_output_path="$OPTARG" ;;
    v) volume="$OPTARG" ;;
    n) numa_node="$OPTARG" ;;
    f) rebuild="$OPTARG" ;; #### if set to a raid id (ie /dev/md12) it will re-add the drive to the raid
and begin a rebuild and logging
  esac
done
# Sanity check arguments
if (("$random" == true && "$sequential" == true ) || ("$random" == false && "$sequential" == false ));
then
    echo "-s OR -r is required, specify one and only one of the two options"
    exit 1
elif  [[ -z "$test_path" || -z "$requested_output_path" || -z "$volume" || -z "$numa_node" ]]; then
  echo "Error: missing flags!  All below are required
    -t path to test files
    -o directory to create and save output into
    -v volume to be tested
    -n allowed numa nodes"
  exit 1
fi

ts=$(date +%s)
output_path=${requested_output_path}.${ts}
# Fix dstat's csv output, included as a comment in case dstat starts complaining about bad imports with
the csv handler, that is the type/types error. Sanity check the line numbers before using.
#wd=`which dstat`
#cp $wd{,.fixed}
#sudo sed -i '547s/.*/            if isinstance( self.val[ name], (tuple, list)):/' $wd.fixed
#sudo sed -i '552s/.*/            elif isinstance( self.val[ name], str):/' $wd.fixed

# Prepare files and output path
mkdir -p $output_path
cp $test_path/$testType* $output_path/
escaped_volume=$(sed 's/\//\\\//g' <<< "$volume")
short_volume=`echo $volume | cut -d\/ -f3`
find $output_path -type f -exec sed -i "s/\/dev\/DISK_ID/$escaped_volume/g" {} \;
find $output_path -type f -exec sed -i "s/NUMA_NODE/$numa_node/g" {} \;
sleep 2
# Start the monitors

if [ -n "$rebuild" ];then
    echo "Rebuilding $rebuild"
    /root/Scripts/raid_rebuilder.sh $output_path/rebuild.log &
fi

nmon -F $output_path/nmon.out -s2 -c100000 -t &
```

```
# Check if mapped volume or device and start dstat accordingly
if [[ $short_volume == md* ]]; then
    dstat -tam -C total --md -M total,$short_volume --output $output_path/dstat.out &
    d_pid=$!
elif [[ $short_volume == sd* || $variable == nv* ]]; then
    dstat -tam -C total -dD total,$short_volume --output $output_path/dstat.out &
    d_pid=$!
fi
iostat -k -t -o JSON -cdx $volume 2 > $output_path/iostat.out &
sleep 2
# Run the tests
for fio_job in `ls -v $output_path/*.fio`;do
    ##### Only uncomment to keep RAID in a rebuilding state
#   /opt/MegaRAID/perccli2/perccli2 /c0/e314/s8 show rebuild >> $output_path/rebuilding_state.log
#   /root/Scripts/raid_kick.sh
#   sleep 60
#   date >> $output_path/rebuilding_state.log
#   /opt/MegaRAID/perccli2/perccli2 /c0/e314/s8 show rebuild >> $output_path/rebuilding_state.log
#   echo "Starting run $fio_job" >> $output_path/rebuilding_state.log
    #### End rebuilding section
    sleep 5
    fio $fio_job --output=$fio_job.log --output-format=json+
    sleep 10

done
# Cleanup monitors
killall nmon
kill $d_pid
killall iostat
echo "]}]}}" >> $output_path/iostat.out
```

Appendix 3: rebuild_runner

This script is used to degrade and monitor the state of the rebuild, and to start monitors and a fio workload against a RAID 5 (md5) volume.

Software RAID

```
#!/bin/bash
### Quirky usage, place the fio files to be tested into a specific directory, probably just one file, and
then use that directory as the -t value.
# Define the optstring
optstring="rspt:o:v:n:"
# Define the variables to store the option values
random=false
sequential=false
precondition=false
# Parse the command-line arguments
while getopts "$optstring" opt; do
  case "$opt" in
    r) random=true
    testType=random ;;
    s) sequential=true
    testType=sequential;;
    p) precondition=true ;;
    t) test_path="$OPTARG" ;;
    o) requested_output_path="$OPTARG" ;;
    v) volume="$OPTARG" ;;
    n) numa_node="$OPTARG" ;;
```

```
  esac
done
# Sanity check arguments
if (("$random" == true && "$sequential" == true ) || ("$random" == false && "$sequential" == false ));
then
    echo "-s OR -r is required, specify one and only one of the two options"
    exit 1
elif  [[ -z "$test_path" || -z "$requested_output_path" || -z "$volume" || -z "$numa_node" ]]; then
  echo "Error: missing flags!  All below are required
    -t path to test files
    -o directory to create and save output into
    -v volume to be tested
    -n allowed numa nodes"
  exit 1
fi
ts=$(date +%s)
output_path=${requested_output_path}.${ts}
# Prepare files and output path
mkdir -p $output_path
cp $test_path/$testType* $output_path/
escaped_volume=$(sed 's/\//\\\//g' <<< "$volume")
short_volume=`echo $volume | cut -d\/ -f3`
find $output_path -type f -exec sed -i "s/\/dev\/DISK_ID/$escaped_volume/g" {} \;
find $output_path -type f -exec sed -i "s/NUMA_NODE/$numa_node/g" {} \;
sleep 2
# Start the monitors
nmon -F $output_path/nmon.out -s2 -c100000 -t &
# Check if mapped volume or device and start dstat accordingly
if [[ $short_volume == md* ]]; then
    dstat -tam -C total --md -M total,$short_volume --output $output_path/dstat.out &
    d_pid=$!
elif [[ $short_volume == sd* || $variable == nv* ]]; then
    dstat -tam -C total -dD total,$short_volume --output $output_path/dstat.out &
    d_pid=$!
fi
iostat -k -t -o JSON -cdx $volume 2 > $output_path/iostat.out &
sleep 2
# Start RAID rebuild monitoring every two seconds
/root/rebuild_logger.sh $output_path/rebuilding_mdstat.log &
rl_pid=$!
# Degraid raid and start rebuild
/root/raid_kick.sh
rid=1
sleep 10
#fio test loop
while true; do
        rebuild_status=$(mdadm --detail /dev/md11 | grep "State :" | xargs)
    if [[ $rebuild_status == *degraded* ]]; then
            for fio_job in `ls -v $output_path/*.fio`;do
                    fio $fio_job --output=${fio_job}_pass_${rid}.log --output-format=json+
                    rid=$((rid+1))
            done
    else
            break
    fi
done
```

```
# Cleanup monitors
killall nmon
kill $d_pid
kill $rl_pid
killall iostat
echo "]}]}}" >> $output_path/iostat.out
```

Hardware RAID

```
#!/bin/bash
# Define the optstring
optstring="rspt:o:v:n:"
# Define the variables to store the option values
random=false
sequential=false
precondition=false
# Parse the command-line arguments
while getopts "$optstring" opt; do
  case "$opt" in
    r) random=true
    testType=random ;;
    s) sequential=true
    testType=sequential;;
    p) precondition=true ;;
    t) test_path="$OPTARG" ;;
    o) requested_output_path="$OPTARG" ;;
    v) volume="$OPTARG" ;;
    n) numa_node="$OPTARG" ;;
  esac
done
# Sanity check arguments
if (("$random" == true && "$sequential" == true ) || ("$random" == false && "$sequential" == false ));
then
    echo "-s OR -r is required, specify one and only one of the two options"
    exit 1
elif  [[ -z "$test_path" || -z "$requested_output_path" || -z "$volume" || -z "$numa_node" ]]; then
  echo "Error: missing flags!  All below are required
    -t path to test files
    -o directory to create and save output into
    -v volume to be tested
    -n allowed numa nodes"
  exit 1
fi

ts=$(date +%s)
output_path=${requested_output_path}.${ts}

# Prepare files and output path
mkdir -p $output_path
cp $test_path/$testType* $output_path/
escaped_volume=$(sed 's/\//\\\//g' <<< "$volume")
short_volume=`echo $volume | cut -d\/ -f3`
find $output_path -type f -exec sed -i "s/\/dev\/DISK_ID/$escaped_volume/g" {} \;
find $output_path -type f -exec sed -i "s/NUMA_NODE/$numa_node/g" {} \;
sleep 2
# start the monitors
```

```
if [ -n "$rebuild" ];then
    echo "Rebuilding $rebuild"
    /root/Scripts/raid_rebuilder.sh $output_path/rebuild.log &
fi

nmon -F $output_path/nmon.out -s2 -c100000 -t &
# Check if mapped volume or device and start dstat accordingly
if [[ $short_volume == md* ]]; then
    dstat -tam -C total --md -M total,$short_volume --output $output_path/dstat.out &
    d_pid=$!
elif [[ $short_volume == sd* || $variable == nv* ]]; then
    dstat -tam -C total -dD total,$short_volume --output $output_path/dstat.out &
    d_pid=$!
fi
iostat -k -t -o JSON -cdx $volume 2 > $output_path/iostat.out &
sleep 2

# Start RAID rebuild monitoring every two seconds
/root/Scripts/rebuild_logger.sh $output_path/rebuilding_mdstat.log &
rl_pid=$!
# Degraid raid and start rebuild
/opt/MegaRAID/perccli2/perccli2 /c0/e314/s8 set offline
sleep 90
/opt/MegaRAID/perccli2/perccli2 /c0/e314/s8 start rebuild
sleep 1
rid=1
sleep 10
#fio test loop
while true; do
        rebuild_status=$(/opt/MegaRAID/perccli2/perccli2 /c0/e314/s8 show rebuild | grep "Status" -A3 |
grep 314)
    if [[ $rebuild_status == *"In Progress"* ]]; then
            for fio_job in `ls -v $output_path/*.fio`;do
                    fio $fio_job --output=${fio_job}_pass_${rid}.log --output-format=json+
                    rid=$((rid+1))
            done
    else
            break
    fi
done

# Cleanup monitors
killall nmon
kill $d_pid
kill $rl_pid
killall iostat
echo "]}]}}" >> $output_path/iostat.out
```

Appendix 4: raid_kick

This script is used to degrade the same disk across all tests, regardless of NVMe ID. When used, $RAID_ID should be replaced with the $VOLUME_ID of the current RAID array being tested.

Software RAID

```
# Because the NVMe ID will change, this will degrade the same physical disk/PCIe slot across tests.
diskToDegrade=`/root/disk_id.sh  | grep “Slot 1” | cut -d’ ‘ -f3`
mdadm --manage /dev/$RAID_ID --fail /dev/$diskToDegrade
mdadm --manage /dev/$RAID_ID --remove /dev/$diskToDegrade
mdadm --zero-superblock /dev/$diskToDegrade
mdadm --detail /dev/$RAID_ID
mdadm --manage /dev/$RAID_ID -a /dev/$diskToDegrade
```

Hardware RAID

```
# Because the NVMe ID will change, this will degrade the same physical disk/PCIe slot across tests.
#!/bin/bash
/opt/MegaRAID/perccli2/perccli2 /c0/e314/s8 set offline
sleep 60
/opt/MegaRAID/perccli2/perccli2 /c0/e314/s8 start rebuild
```

Appendix 5: disk_Id

This script is used to obtain an NVMe ID based on a drive serial number to ensure consistency between reboots where an NVMe ID might change. Replace **$DISK-ID** with the serial numbers found in step 2 of the **Initial configuration section** for the disk associated with each slot. This script is used for software RAID only.

```
# Usage: Confirm disk serial number and CPU association in iDRAC. Based on the serial number shown in the
storage/physical disks section and the NUMA associations from system PCIe devices, update the grep value
below for a given disk.
echo “Slot 0: `ls -l /dev/disk/by-id | grep $DISK-ID | cut -d/ -f3 | sort | uniq`”
echo “Slot 1: `ls -l /dev/disk/by-id | grep $DISK-ID | cut -d/ -f3 | sort | uniq`”
echo “Slot 2: `ls -l /dev/disk/by-id | grep $DISK-ID | cut -d/ -f3 | sort | uniq`”
echo “Slot 3: `ls -l /dev/disk/by-id | grep $DISK-ID | cut -d/ -f3 | sort | uniq`”
echo “Slot 4: `ls -l /dev/disk/by-id | grep $DISK-ID | cut -d/ -f3 | sort | uniq`”
echo “Slot 5: `ls -l /dev/disk/by-id | grep $DISK-ID | cut -d/ -f3 | sort | uniq`”
echo “Slot 6: `ls -l /dev/disk/by-id | grep $DISK-ID | cut -d/ -f3 | sort | uniq`”
echo “Slot 7: `ls -l /dev/disk/by-id | grep $DISK-ID | cut -d/ -f3 | sort | uniq`”
```

Appendix 6: rebuild_logger

This script is used to monitor the rebuild progress and log that output to a file.

Software RAID

```
#!/bin/bash
outputFile=$1
sleep 5
while true; do
        rebuild_status=$(mdadm --detail /dev/md5| grep “State :” | xargs)
        cat /proc/mdstat >> $outputFile
        echo “`date` ^^^^” >> $outputFile
        if [[ $rebuild_status == “State : clean” ]]; then break;fi
        sleep 2
done
```

Hardware RAID

```
#!/bin/bash
outputFile=$1
echo $(/opt/MegaRAID/perccli2/perccli2 /c0/e314/s8 show rebuild)
sleep 5
while true; do
        rebuild_status=$(/opt/MegaRAID/perccli2/perccli2 /c0/e314/s8 show rebuild | grep “Status” -A3 |
grep “In Progress”)
        echo $rebuild_status >> $outputFile
```

```
        echo "`date` ^^^^" >> $outputFile
        if [[ $rebuild_status == "Not in progress" ]]; then break;fi
        sleep 2
done
```

Appendix 7: drive_prep_script

This script is used to prepare the drive for testing. This script is used for both software and hardware RAID.

```
each=$1
echo none > /sys/block/$each/queue/scheduler
echo 0 > /sys/block/$each/queue/add_random
echo 2 > /sys/block/$each/queue/nomerges
echo 1023 > /sys/block/$each/queue/nr_requests
echo 0 > /sys/block/$each/queue/rotational
echo 2 > /sys/block/$each/queue/rq_affinity
echo 1024 > /sys/block/$each/queue/max_sectors_kb
echo 1 > /sys/block/$each/queue/iostats

echo "$each updated.  Current values are now:
/sys/block/$each/queue/scheduler: `cat /sys/block/$each/queue/scheduler`
/sys/block/$each/queue/add_random: `cat /sys/block/$each/queue/add_random`
/sys/block/$each/queue/nomerges: `cat /sys/block/$each/queue/nomerges`
/sys/block/$each/queue/nr_requests: `cat /sys/block/$each/queue/nr_requests`
/sys/block/$each/queue/rotational: `cat /sys/block/$each/queue/rotational`
/sys/block/$each/queue/rq_affinity: `cat /sys/block/$each/queue/rq_affinity`
/sys/block/$each/queue/max_sectors_kb: `cat /sys/block/$each/queue/max_sectors_kb`
/sys/block/$each/queue/iostats: `cat /sys/block/$each/queue/iostats`"
```

Appendix 8: Iometer PowerShell Script

The following PowerShell script was utilized to run the Iometer workloads.

```
<#  Iometer-runner.ps1 -runID <run number of the test> -results_dir <where to store test results> -test_
config_dir <Location of Iometer Configuration Files>
#>
param(

    $runID,
    $results_dir,
    $test_config_dir

)
# Define the iometer executable file
$iometerFile = "<location of Iometer>\IOmeter.exe"
#################### Monitor, run, unmonitor ####################
$resultsDir = "$results_dir\$runID\"
# Make output directory
New-Item -ItemType Directory -Path "$resultsdir"
# Start the logging
logman.exe start <Perfmon Data Collector Set>
# Run Iometer with the config file and the result file
foreach ($file in Get-ChildItem -Path $test_config_dir -Filter *.icf) {
    # Assign $filename to the name of the file
    $filename = $file.Name
    # Assign $outputfile to the name of the file, but replacing the .icf with a .csv
    $outputfile = $file.Name -replace ".icf", ".csv"
    $runner = Start-Process -FilePath $iometerFile -ArgumentList "/c $test_config_dir$filename /r
${resultsdir}IometerOutput${filename}_$runId.csv" -Wait
```

```
}
# Stop perfmon collecting
logman.exe stop <Perfmon Data Collector Set>
```

Appendix 9: Iometer Hardware RAID Rebuild PowerShell Script

The following PowerShell script was utilized to run the Iometer workloads during hardware RAID rebuild.

```
<#  Iometer-runner.ps1 -runID <run number of the test> -results_dir <where to store test results> -test_
config_dir <Location of Iometer Configuration Files>
#>
# Define a run number; this, along with others, should really be a command-line option
param(
    $runID,
    $results_dir,
    $test_config_dir
)
# Define the iometer executable file
$iometerFile = "<location of Iometer>\IOmeter.exe"
$resultsDir = "$results_dir\$runID\"
#################### Monitor, run, unmonitor ####################
# Make output directory
New-Item -ItemType Directory -Path "$resultsDir"
# Start the Logging
logman.exe start <Perfmon Data Collector Set>
## Break the RAID ##
function set-Offline{
    $setOffline = perccli2 /c0/e314/s8 set offline
    Start-Sleep -Seconds 30
}
function start-Rebuild{
    $startRebuild = perccli2 /c0/e314/s8 start rebuild
    Start-Sleep 5
}
function job-Log{
    $rebuildJobLog = "$resultsDir\rebuild.txt"
    $rebuildStatus = start-job -Name "raidRebuild" -FilePath C:\Users\Administrator\Desktop\Scripts\raid-
status.ps1 -ArgumentList @($rebuildJobLog)
}
function get-rebuild{
    $getRebuild =perccli2 /c0/e314/s8 show rebuild
    return $getRebuild | Select-String -CaseSensitive "Minutes"
}
set-Offline
start-Rebuild
job-Log
get-rebuild
Start-Sleep 5
# Run the iometer with the config file and result file
foreach ($file in Get-ChildItem -Path $test_config_dir -Filter *.icf) {
    # Assign $filename to the name of the file
    $filename = $file.Name
    # Check and break raid if needed
    if(get-rebuild){
        # Assign $outputfile to the name of the file, but replacing the .icf with a .csv
        $outputfile = $file.Name -replace ".icf", ".csv"
        $runner = Start-Process -FilePath $iometerFile -ArgumentList "/c $test_config_dir$filename /r
${resultsDir}IometerOutput${filename}_$runId.csv" -Wait
```

```
        Start-Sleep -Seconds 90
    }
    else{
        set-Offline
        start-Rebuild
        $outputfile = $file.Name -replace ".icf", ".csv"
        $runner = Start-Process -FilePath $iometerFile -ArgumentList "/c $test_config_dir$filename /r
${resultsDir}IometerOutput${filename}_$runId.csv" -Wait
        Start-Sleep -Seconds 90
    }
}
#stop perfmon collecting
logman.exe stop <Perfmon Data Collector Set>
#Stop raid state monitoring
Stop-Job raidRebuild
```

Appendix 10: HammerRunner

This script is used to trigger a restore of the Microsoft SQL database from the backup and start the monitoring of the system metrics.

Replace **$SQL_PASSWORD** with the password set in Step 18c of the **HammerDB instance** preparation section.

Software RAID

```
#!/bin/bash
#node_numa=
rid=$1
disk_ids="md0 md5"
ts=$(date +%s)
results_dir=${rid}.${ts}
#### Prepare DB
#touch /ms_sql_logs/sqlbackup_${ts}.txt
echo " Restoring DB to original state"
sqlcmd -S localhost -U SA -P $SQL_PASSWORD -C -Q "RESTORE DATABASE [TPCC_2500] FROM DISK = N'/ms_sql_
logs/cleanBackup2500.bak' WITH FILE = 1, NOUNLOAD, REPLACE, RECOVERY, STATS = 1" | tee -a /ms_sql_logs/
sqlbackup_${ts}.txt
sleep 15
sqlcmd -S localhost -U SA -P $SQL_PASSWORD -C -Q "ALTER DATABASE TPCC_2500 SET RECOVERY SIMPLE"
echo "restore complete, sleeping 930"
sleep 930
echo "Restore of database complete Press enter to degrade raid and start monitoring"
read -r
echo "starting test run $rid"
#Start monitoring
mkdir $results_dir
nmon -F $results_dir/nmon.out -s2 -c100000 -t &
iostat -k -t -o JSON -cdx $disk_ids 2 > $results_dir/iostat.out &
### Degrade RAID
### uncomment to degrade the raid
#diskToDegrade=`/root/disk_id.sh  | grep "Slot 1" | cut -d' ' -f3`
#mdadm  --manage /dev/md5 --fail  /dev/$diskToDegrade
#mdadm  --manage /dev/md5 --remove /dev/$diskToDegrade
#mdadm --zero-superblock /dev/$diskToDegrade
#mdadm --detail /dev/md5
####
### Rebulid RAID
```

```
### Uncomment to rebuild
#sleep 30
#echo $(mdadm –a /dev/md5 /dev/$diskToDegrade)
```

Hardware RAID

```
#!/bin/bash
rid=$1
disk_ids=”sdb sda”
ts=$(date +%s)
results_dir=${rid}.${ts}

#### Prepare DB
echo “ Restoring DB to original state”
sqlcmd –S localhost –U SA –P TestPassword1 –C –Q “RESTORE DATABASE tpcc FROM DISK = N’/mssql/data/
cleanBackup2500.bak’ WITH FILE = 1, NOUNLOAD, REPLACE, RECOVERY, STATS = 1” | tee –a /mssql/log/
sqlbackup_${ts}.txt
sleep 15
sqlcmd –S localhost –U SA –P TestPassword1 –C –Q “ALTER DATABASE tpcc SET RECOVERY SIMPLE”
echo “restore complete, sleeping 600”
sleep 300
#echo “Restore of database complete Press enter to degrade raid and start monitoring”
#read –r
echo “starting test run $rid”
#Start monitoring
mkdir $results_dir
nmon –F $results_dir/nmon.out –s2 –c100000 –t &
iostat –k –t –o JSON –cdx $disk_ids 2 > $results_dir/iostat.out &
### break and start rebuild of raid
/root/230210-Scripts/raid_rebuilder.sh $results_dir/raid_rebuilder.out
```

Appendix 11: Builder.tcl

This is the configuration used with HammerDB to build out the 2500 warehouse database.

Replace **$SQL_PASSWORD** with the password set in Step 18c of the **HammerDB instance** preparation section.

```
dbset db mssqls
dbset bm TPROC-C
diset connection mssqls_linux_server localhost
diset connection mssqls_port 1433
diset connection mssqls_uid sa
diset connection mssqls_pass $SQL_PASSWORD
diset tpcc mssqls_dbase TPCC_2500
diset tpcc mssqls_count_ware 2500
diset tpcc mssqls_num_vu 1
vuset vu 1
vuset showoutput 1
buildschema
vudestroy
```