

Intel® Solid State Drive DC P4510 Series using Linux*
Intel® Virtual RAID on CPU (Intel® VROC) with FIO*

Performance Evaluation Guide

June 2019



Ordering Information

Contact your local Intel sales representative for ordering information.

Revision History

Revision Number	Description	Revision Date
001	<ul style="list-style-type: none"> Initial release 	February 2018
002	<ul style="list-style-type: none"> Updated and reworked document to align with current best known methods (BKMs) and processes for testing performance. 	January 2019
003	<ul style="list-style-type: none"> Added link to FIO Script Reference Code 	May 2019
004	<ul style="list-style-type: none"> Fixed hyperlinks Moved FIO Download & Installation details 	June 2019

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Performance results are based on testing as of the date set forth in the Configurations and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.

System configuration, October 11, 2018: Intel® Server Board S2600WFQ family, Intel® Xeon® 8170 Series Processors, BIOS Release 7/09/2018, BIOS Version: SE5C620.86B.00.01.0014.070920180847, 26 cores@ 2.1GHz, RAM 192GB

OS: RedHat® Linux 7.4, kernel- 3.10.0-693.33.1.el7.x86_64, mdadm - v4.0 - 2018-01-26 Intel build: RSTe_5.4_WW4.5, Intel® VROC Pre-OS version 5.3.0.1039, 4x Intel® SSD DC P4510 Series 2TB drive firmware: VDV10131, Retimer

Results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.

All documented performance test results are obtained in compliance with JESD218 Standards; refer to individual sub-sections within this document for specific methodologies. See www.jedec.org for detailed definitions of JESD218 Standards.

Intel does not control or audit the design or implementation of third party benchmark data or Web sites referenced in this document. Intel encourages all of its customers to visit the referenced Web sites or others where similar performance benchmark data are reported and confirm whether the referenced benchmark data are accurate and reflect performance of systems available for purchase.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

For copies of this document, documents that are referenced within, or other Intel literature please contact you Intel representative.

Intel, the Intel logo, Xeon, and VROC are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

© Intel Corporation



Contents

1	Setup	4
1.1	Hardware Configuration.....	4
1.2	BIOS Settings	5
1.3	Intel® Virtual RAID On CPU (from the BIOS).....	5
1.4	Red Hat Enterprise Linux Installation/Dependencies	6
1.4.1	Downloading and Installing FIO Benchmarking Tool.....	6
1.5	Downloading and Installing Intel® Virtual RAID on CPU (VROC).....	6
1.6	Downloading and Installing Intel® SSD Data Center Tool (ISDCT).....	7
2	Preparing the RAID (VROC)	8
2.1	Formatting the Intel® NVMe SSDs	8
2.2	Creating the RAID Array	8
2.3	Creating XFS Filesystem (Not Required for Benchmark Testing)	9
3	Running a Performance Test.....	10
3.1	Conditioning the Drive or Volume Before Testing	10
3.2	Notes About FIO and Some Parameters	11
3.3	Checking CPU Usage	11
4	Sample FIO Script for Testing Intel® SSD DC P4510 Series.....	12
5	Analyzing the Results.....	15
6	Test Results using Intel® VROC and Intel® SSD DC P4510 (2TB capacity)	17



1 Setup

This is a quick-start guide to using the FIO* benchmarking tool for systems containing Intel® Xeon® Scalable Processors with Red Hat Enterprise* Linux 7.4 and the latest Intel® Virtual RAID On CPU (Intel® VROC) package. This document includes testing of the Intel® Solid State Drive DC P4510 Series. Specific steps are for testing NVMe* SSD RAID arrays on Intel® Volume Management Device (Intel® VMD) - enabled PCIe* lanes.

Note: This document references FIO Script code for performance evaluations; this script is also available in a format that you can copy and paste here: [FIO Script Reference Code](#).

1.1 Hardware Configuration

For maximum use of Intel® VROC with availability to RAID 0, RAID 5, RAID 1, and RAID 10, a 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.

- Verify Riser 2 card, PCIe slot 1 indicates “x16”. Install 4-port retimer (x16) on Riser 2, PCIe slot 1 (Note: As of BIOS version SE5C620.86B.00.01.0014.070920180847, a retimer is no longer supported on Riser 1).
- Connect Oculink (straight connectors) to Oculink (straight connectors) cables from retimer to desired drive slot connections on backplane.
- For six disk configurations, in addition to the retimer, connect Oculink to Oculink cables from the onboard Oculink connections on second CPU on the motherboard to desired drive slot Oculink connections on backplane.
- Install NVMe SSDs into applicable drive slots.

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



1.2 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 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 → Power & Performance → CPU C State Control → Package C-State → C6 (non Retention) state
 Verify... Advanced → Power & Performance → CPU C State Control → C1E → Disabled
 Verify... Advanced → Power & Performance → CPU C State Control → Processor C6 → Enabled
 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 setup below is based on a 4-port retimer (x16) installed on Riser 2, PCIe slot 1. Your setup may be different than below. 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

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

The following info 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 below.

After setting the BIOS settings from the “BIOS settings” section 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 will be able to see the VROC hardware key installed, Intel® VROC Pre-OS version number, and any existing VROC RAID volumes. Select “All Intel VMD Controllers” to view the applicable NVMe SSDs.



1.4 Red Hat Enterprise Linux Installation/Dependencies

All Red Hat Enterprise Linux installations shall be in UEFI mode.

Some platforms require “nomodeset” appended to the end of the “linuxefi” command in the grub configuration menu like below.

```
setparams 'Install Red Hat Enterprise Linux 7.4'

linuxefi /images/pxeboot/vmlinuz inst.stage2=hd:LABEL=RHEL-7.4\x20Serv\
er.x86_64 quiet nomodeset_
initrdefi /images/pxeboot/initrd.img
```

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

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

```
yum install gcc libaio-devel zlib-devel unzip sysstat libreport-filessystem numactl
redhat-lsb-core sg3_utils nvme-cli
```

1.4.1 Downloading and Installing FIO Benchmarking Tool

Download the FIO master zip from GITHUB at <https://github.com/axboe/fio>

You will also need libaio-devel rpm in order to install the necessary .h files to build the libaio ioengine. Also rpms gcc and zlib-devel are needed for proper FIO operation. These rpms should be on the installation source for your Linux distribution.

```
unzip fio-master.zip
cd fio-master
./configure
make
make install
# verify fio installed by running the below command
fio --version
```

1.5 Downloading and Installing Intel® Virtual RAID on CPU (VROC)

The Intel® Virtual RAID on CPU (VROC) package for Intel® S2600WFT platforms using Linux* can be downloaded at <https://downloadcenter.intel.com/download/28158/?product=89015>

```
unzip RSTe_VROC_RHEL7.4_5.4_2018.3.30.zip
cd RSTe_VROC_RHEL7.4_5.4_2018.3.30
mkdir -p /mnt/iso
mount -o loop rste-5.4_PV_rhel7.4.iso /mnt/iso
cd /mnt/iso/rpms
rpm -Uvh --force *.x86_64.rpm
# verify VROC installed by running the below command
mdadm --version
cd ~
# unmount the iso by running the below command
umount /mnt/iso
```

If you are running performance tests on a different platform (not a S2600WFT), please download the Intel® VROC software from the platform manufacturer's download center. Intel® VROC release packages are custom to platform hardware and the correct version must be used for reliable results.



1.6 Downloading and Installing Intel® SSD Data Center Tool (ISDCT)

Intel® SSD Data Center Tool (ISDCT) is used to update firmware on Intel® SSDs and is used to perform a low level format on Intel® NVMe SSDs. ISDCT can be downloaded at <https://downloadcenter.intel.com/download/28035?v=t>

```
unzip Intel_SSD_Data_Center_Tool_3.0.14_Linux.zip
yum install isdct-3.0.14-1.x86_64.rpm
# verify ISDCT installation by running the below command
isdct version
# to list devices ISDCT recognizes, run the following command
isdct show -intelssd
```

§



2 Preparing the RAID (VROC)

2.1 Formatting the Intel® NVMe SSDs

Drives with previous RAID metadata or used for other applications could impact performance. Clearing the metadata and erasing or performing a low level format can stabilize your drives providing more consistent results.

To remove the metadata:

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

To show ISDCT indexes:

```
isdct show -intelssd
```

To low level format:

```
isdct start -force -nvmeformat -intelssd <isdct index number>
```

Repeat only the low level instruction for all drives to be used in new array.

2.2 Creating the RAID Array

Intel VROC uses mdadm to create and manage RAID volumes when VMD is enabled in the BIOS. The available RAID volumes are RAID 0, RAID 5, RAID 1, and RAID 10.

First start by creating a VROC metadata container:

```
mdadm -C /dev/md/ims0 /dev/<NVMe devices to use> -n <num of disks used> -e imsm
```

Example:

```
mdadm -C /dev/md/ims0 /dev/nvme[0-3]n1 -n 4 -e imsm
```

Then create the RAID (Note: leaving out “-c <chunk size>” will set the chunk size to default.):

```
mdadm -C /dev/md/<name of volume> /dev/md/ims0 -n <num of disk used> -l <level of RAID> -c <chunk size>
```

Example:

```
mdadm -C /dev/md/4DR0 /dev/md/ims0 -n 4 -l 0 -c 64
```

or

```
mdadm -C /dev/md/4DR0 /dev/md/ims0 -n 4 -l 0
```

Default chunk sizes:

```
RAID 0: 128k  
RAID 5: 128k  
RAID 1: n/a  
RAID 10: 128k
```

Note: If no other RAID volumes existed before creating the RAID volume, the block devices will be /dev/md127 for the container and /dev/md126 for the RAID volume.

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 4 > /sys/block/md126/md/group_thread_cnt
```




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

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.

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

```
echo 5000000 > /sys/block/md126/md/sync_speed_max
```

2.3 Creating XFS Filesystem (Not Required for Benchmark Testing)

Most benchmark testing is performed on raw devices, however if you would like to see results from a filesystem installed, this section will explain it. Drives perform better when data is aligned. 4k alignment is ideal for most SSDs. The following steps will demonstrate how to create partitions so they are aligned (minimal and optimal) and the steps will show how to create the XFS filesystem. The following example applies to a VROC RAID volume.

```
parted -a optimal -s /dev/md126 mklabel gpt mkpart primary 0% 100%
```

Checking the alignment:

```
parted /dev/md126 align-check optimal 1
parted /dev/md126 align-check minimal 1
```

Creating the XFS filesystem:

```
mkfs.xfs /dev/md126p1 -Kf
```

Mount filesystem:

```
mkdir -p /mnt/md126
mount /dev/md126p1 /mnt/md126
```

Note: If you plan on running FIO on any XFS filesystem, you must fill the drive with sequential writes first before running random workloads. Running random workloads first on an empty XFS system will generate many errors.

Note: There are a few parameters that need to be adjusted in the FIO script before running tests on the XFS filesystem. First the “filename=” parameter will need to reflect the location of the filesystem mount plus the FIO filename you plan to use (e.g., “filename=/mnt/md126p1/iofile”). If the filesystem does not currently have the FIO file (e.g., “iofile”) in the directory, then the “time_based” parameter must be eliminated in the first FIO workload which is the sequential pre-conditioning run as described in the [“Conditioning the drive or volume before testing”](#) section of this document. Also, the “size=” parameter cannot be set using a percentage like “100%” and it cannot be the full size of the volume. An actual size must be specified and it must be at least a few Gigabytes smaller than the full size. For example, if the RAID volume is 16TB in size, the parameter can be “size=15892G”. After the FIO file is created in the directory, all tests should use the “time_based” parameter and the “size=” parameter can go back to “size=100%”.

3 Running a Performance Test

Stopping the IRQBALANCE service before testing is recommended. This can be stopped by typing “systemctl stop irqbalance”. Do not disable IRQBALANCE because it may cause CPU soft lockup errors and kernel panics. If IRQBALANCE was disabled, please re-enable and reboot the system.

Benchmarking a drive, array, or workload should be reproducible and relevant. A short burst test, (e.g., 60 seconds) does not give the storage media enough time to reach steady state, and therefore the results can be unreliable. Also, multiple iterations should be run to verify the reproducibility of results. All testing should be run for a minimum of 10 minutes (preferably 10 minutes of ramping followed by 10 minutes of run time). The test sequence should be repeated at least 2 more times to get a statistical set of data to take averages and standard deviation. If your results vary greater than +5%, you may have to check your test parameters, devices under test, and overall system. Additionally, each OS has its own set of performance counters to capture IOPS, bandwidth, and CPU usage (overall). More data is better when attempting to predict real world performance by using synthetic benchmarks.

When testing 128KB sequential READS/WRITEs, with four devices in a RAID 0 or RAID 5, it is recommended to have an IO depth of 128, but the device may not peak until it reaches an IO depth of 256. For 4K random, the `numjobs` (threads or workers) and `iodepth` should be increased in order to fully saturate the device. If using four devices in a RAID 0 or RAID 5, the `iodepth` 256 may be needed to obtain peak value, depending on the SSD. 16 or more `numjobs` or threads may be needed to identify the right combination of IOPS and Latency.

3.1 Conditioning the Drive or Volume Before Testing

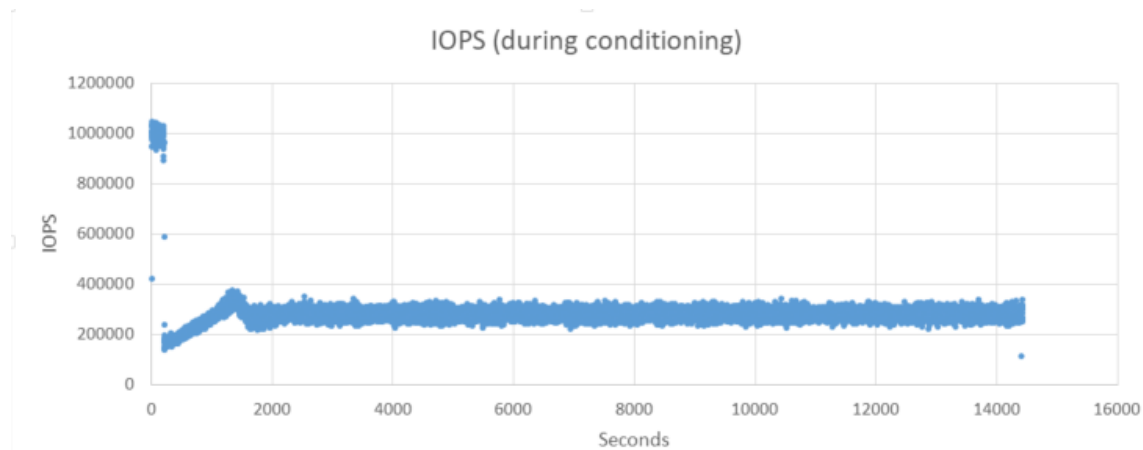
Conditioning is important prior to running the performance testing. The devices should start testing from a steady state to allow accurate performance measurements. Conditioning involves writing packets to the entire drive multiple times. We suggest separating the sequential workload tests and random workload tests to get more repeatable results. **It is crucial to condition the drives/volume for at least 4 hours (for larger volumes, this time may need to be increased) with writes of each applicable block size (e.g., 128k or 4k) before running the sequential tests and random tests. This means if you change block sizes during the testing procedure, we recommend preconditioning again to the new block size.** Getting steady state or stable performance is critical to avoid latency spikes or other performance bottlenecks.

Note: Conditioning is not required on Intel® Optane™ SSDs.

The [sample FIO script](#), as presented in Section 4, is configured for pre-conditioning.

The sample chart below represents IOPS during conditioning. The IOPS start really high, then drop down quite a bit, then eventually stabilize reaching steady state. This is why conditioning is important before starting the actual performance testing. The data here is just an example. Results will vary depending on setup.

Figure 1: Pre-Test Conditioning





3.2 Notes About FIO and Some Parameters

Adding CPU affinity may be helpful. For example, if a 4-port retimer is installed on Riser 2, PCIe slot 1 which is on the second CPU (CPU Node 1), and each processor contains 24 cores, then the command to kick off FIO would be “numactl --cpunodebind=1 fio...”. This command would also contain FIO parameters of “--cpus_allowed_policy=split --cpus_allowed=24-47”.

FIO output explained:

<https://tobert.github.io/post/2014-04-17-fio-output-explained.html>

FIO manual page:

<https://linux.die.net/man/1/fio>

... or type “man fio > man_fio.txt” in your system under test and copy the file to laptop/PC.

How FIO works:

<https://github.com/axboe/fio/blob/master/HOWTO>

Table 1: FIO Parameters

FIO Parameter	Notes
randrepeat=0	The parameter “randrepeat” defaults to 1 (enabled) and should be disabled to get more accurate performance. Results have shown that it could provide incorrect performance values if enabled (default).
unified_rw_reporting=1	FIO normally reports the write results in one column and read results in another column. With this parameter enabled, it combines the results into a “mixed” column therefore making it easier to process in Microsoft Excel. Remove this if separate reads and writes are needed.
random_generator=tausworthe64	For large capacity SSDs like 4TB or 8TB, the default value of “tausworthe” for parameter “random_generator” could present errors. Forcing it to use “tausworthe64” will eliminate these errors.
iodepth_batch=<iodepth> iodepth_batch_complete_min=<iodepth>	FIO may limit the iodepth to 16 even if iodepth is set to a larger number. Adding these parameters insures all iodepths are utilized which increases IOPS and bandwidth.
minimal	This parameter outputs the results to semicolon-delimited values which can then be changed to comma-separated values by typing ... “sed -i -e 's/;/,/g' <fio output file>”

3.3 Checking CPU Usage

Tools like `iostat` and `dstat` can be used to collect IOPS to compare with the results that FIO outputs. These tools can also be used to collect overall system CPU usage. When using `iostat` or `dstat`, observe the CPU idle percentages. The outcome is an accurate value of CPU usage not active due to all of the supporting processes for RAID, drives, and other tools that may be used for background activities. These idle percentages can be subtracted from 100 to get actual CPU usage.

To install `iostat`, run “yum install sysstat”. To install `dstat`, run “yum install dstat”.

The sample FIO script included in this document is configured to use `iostat`.



4 Sample FIO Script for Testing Intel® SSD DC P4510 Series

The following bash script will need to be saved as a .sh file. To be able to run the script, "chmod 755 <script file>.sh" or "chmod 777 <script file>.sh" may need to be applied to make it executable by the administrative user. You may have to run this script as root.

The script below can also be found [here](#). You can download the linked .txt file for a version of the script that can be copied and pasted with proper formatting.

Note: Many variables in this script can be changed per your needs.

```
#!/bin/bash

# Adjust any of the variables below
numOfHoursToPrepDrive=4
# Variables fioRuntime and fioRamptime are in seconds
fioRuntime=600
fioRamptime=600
numOfIterations=3
# If unsure of what CPU the NVMe SSDs are on, set "affinitized=false"
affinitized=true
# If NVMe SSDs are installed on 1st CPU then set "numaCpu=0"
# If NVMe SSDs are installed on 2nd CPU then set "numaCpu=1"
numaCpu=1
# Adjust variable "fioDevice" to applicable device like /dev/nvme0n1 for pass-through
# or /dev/md126 for VROC RAID volume
fioDevice=/dev/md126
if [[ "${fioDevice:0:7}" == "/dev/md" ]]; then
    # Variable "raidVolume" can be changed to whatever the RAID level is
    # (e.g., "RAID0" or "4DR0" or "RAID5" or "4DR5" or etc)
    raidVolume="RAID"
else
    raidVolume="Passthru"
fi

# Start of pre-defined functions
RunFioTest ()
{
    iostatRuntime=$(( tempRuntime + tempRamptime - 2 ))
    tempFioOutputStr="${configName}_${blockSize}_${SeqOrRandom}_${writePercent}write_IODepth${
ioDEPTH}_Workers${workers}_${iteration}"
    iostatFile="./${resultsFolder}/${tempFioOutputStr}_iostat.txt"
    fioParams="--time_based --ioengine=libaio --direct=1 --buffered=0 --norandommap"
    fioParams="${fioParams} --refill_buffers --stonewall"
    fioParams="${fioParams} --disable_slrat=1 --disable_lat=1 --disable_bw=1"
    fioParams="${fioParams} --randrepeat=0 --thread --unified_rw_reporting=1"
    fioParams="${fioParams} --group_reporting --do_verify=0"
    if [[ $affinitized == true ]]; then
        fioParams="${fioParams} --cpus_allowed_policy=split --cpus_allowed=${affinity}"
    fi
    if [[ "${SeqOrRandom}" == "Random" ]]; then
        fioParams="${fioParams} --percentage_random=100 --random_generator=tausworthe64"
    fi
    fioParams="${fioParams} --runtime=${tempRuntime} --ramp_time=${tempRamptime}"
    fioParams="${fioParams} --name=${tempFioOutputStr}"
    fioParams="${fioParams} --size=100%"
    fioParams="${fioParams} --filename=${fioDevice}"
    if [ $writePercent -eq 100 ] && [[ "${SeqOrRandom}" == "Seq" ]]; then
        fioParams="${fioParams} --rw=write"
    elif [ $writePercent -eq 0 ] && [[ "${SeqOrRandom}" == "Seq" ]]; then
        fioParams="${fioParams} --rw=read"
    elif [ $writePercent -eq 100 ] && [[ "${SeqOrRandom}" == "Random" ]]; then
        fioParams="${fioParams} --rw=randwrite --rwmixwrite=100"
    elif [ $writePercent -eq 0 ] && [[ "${SeqOrRandom}" == "Random" ]]; then
        fioParams="${fioParams} --rw=randread --rwmixread=100"
    else
        fioParams="${fioParams} --rw=randrw --rwmixwrite=${writePercent}"
    fi
}
```



```

fi
fioParams="${fioParams} --bs=${blockSize}"
fioParams="${fioParams} --iodepth=${iodepth} --iodepth_batch=${iodepth}"
fioParams="${fioParams} --iodepth_batch_complete_min=${iodepth}"
fioParams="${fioParams} --numjobs=${workers}"
fioParams="${fioParams} --minimal --eta=never"
fioParams="${fioParams} --output=./${resultsFolder}/${tempFioOutputStr}.csv"
echo "$(date +%x %T) __FioTest_${tempFioOutputStr}"
iostat -tmyx -p ${fioDevice} 1 ${iostatRuntime} > ${iostatFile} &
$numactlCmd fio ${fioParams} &
wait $(pidof fio)
# The below commands change the results from semicolon delimited to comma delimited
sed -i -e 's/;/,/g' ./${resultsFolder}/${tempFioOutputStr}.csv
sed -i -e 's/_/,/g' ./${resultsFolder}/${tempFioOutputStr}.csv
}
# End of pre-defined functions

#####

# Stopping irqbalance service
systemctl stop irqbalance

# Setting CPU affinity command and applicable CPUs if variable "affinitized=true"
numactlCmd=""
if [[ $affinitized == true ]]; then
    numactlCmd="numactl --cpunodebind=$numaCpu"
    numOfNumaNodes=$(( $( lscpu | grep "NUMA node(s):" | rev | awk '{print $1}' | rev ) ))
    numaNodesStrings=( $( lscpu | grep "NUMA node" | grep "CPU" | awk '{print $2}' ) )
    if [ $numaCpu -ge $numOfNumaNodes ] && [ $numOfNumaNodes -ge 1 ]; then
        numaCpu=$(( numOfNumaNodes - 1 ))
    fi
    if [ $numaCpu -lt $numOfNumaNodes ]; then
        nodeCpus=$( lscpu | grep "NUMA ${numaNodesStrings[$numaCpu]}" | awk '{print $4}' )
        nodeCpus=( ${nodeCpus//,/ } )
        affinity=${nodeCpus[0]}
    fi
fi

configName="VROC-${raidVolume}"
timestamp=$(date +%Y_%m%d_%H%M)
resultsFolder=${timestamp}_${configName}
mkdir -p ./${resultsFolder}

# Start of sequential workload testing section
SeqOrRandom=Seq
for blockSize in 128k; do
    workers=1
    # Start of pre-conditioning section
    iteration=Prep; writePercent=100; iodepth=128
    tempRuntime=$(( numOfHoursToPrepDrive * 60 * 60 )); tempRamptime=30
    RunFioTest
    # End of pre-conditioning section
    for (( iteration=1; iteration<=$numOfIterations; iteration++ )); do
        for writePercent in 100 0; do
            # Modify the below iodepths with whatever workloads are needed
            for iodepth in 128; do
                tempRuntime=${fioRuntime}; tempRamptime=${fioRamptime}
                RunFioTest
            done
        done
    done
done
# End of sequential workload testing section

# Start of random workload testing section
SeqOrRandom=Random
for blockSize in 4k; do
    # Start of pre-conditioning section
    iteration=Prep; writePercent=100; workers=8; iodepth=32
    tempRuntime=$(( numOfHoursToPrepDrive * 60 * 60 )); tempRamptime=30

```



```
RunFioTest
# End of pre-conditioning section
for (( iteration=1; iteration<=$numOfIterations; iteration++ )); do
  for writePercent in 100 30 0; do
    for workers in 16 8; do
      # Modify the below iodepths with whatever workloads are needed
      if [ $workers -eq 16 ]; then
        iodepths=(256 128 64 32 16 8)
      elif [ $workers -eq 8 ]; then
        iodepths=(256 128 64 32 16 8)
      elif [ $workers -eq 4 ]; then
        iodepths=(256 128 64)
      else
        iodepths=(256 128 64 32 16 8)
      fi
      for iodepth in "${iodepths[@]}; do
        tempRuntime=${fioRuntime}; tempRamptime=${fioRamptime}
        RunFioTest
      done
    done
  done
done
# End of random workload testing section

# Combining all results into one file
cat ./${resultsFolder}/*.csv > ./${resultsFolder}/AllFioResults.csv

# Resuming irqbalance service
systemctl start irqbalance
```

§



5 Analyzing the Results

With the recommendations from this document, performing at least three iterations of each preferred workload should produce repeatable results. Once the results are viewed from Microsoft Excel and all rows are sorted to the applicable workloads, columns can be added for determining averages and standard deviations. To determine the standard deviation percentage for a group of cells in Excel, take the standard deviation of that group of cells and divide it by the average of the same group of cells and convert to a percentage format. This standard deviation percentage should indicate within 5%. If it shows a high variance across multiple workloads, you may need to check drive firmware or health of each drive in your RAID volume. If the standard deviation percentage is high in the beginning of the testing (after the conditioning runs), then conditioning of the drive/volume may need to be extended.

After looking at the overall performance results, if they appear to be low, it could be that the device is not being fully saturated. Increase the `numjobs` (threads) to see if this increases the performance but keep in mind that this could impact latency. Looking at the `dmesg` logs may help determining any bottlenecks as well.

If the sample FIO script from this document was used, below is a useful CSV header to identify all of the columns from the "AllFioResults.csv" file that was created at the end of the script. This can be copied and pasted into the first line of that file.

Note: Since the "unified_rw_reporting=1" FIO command was used, all results, whether they were reads or writes, are combined into all read columns (now labeled as "combined") in this header.

```
terse_version_3,fio_version,jobname,blocksize,SeqOrRandom,%write,iodepth,threads,iteration,grou
pid,error,combined_kb,combined_bandwidth,combined_iops,combined_runtime_ms,combined_slat_min,co
mbined_slat_max,combined_slat_mean,combined_slat_dev,combined_clat_min,combined_clat_max,combin
ed_clat_mean,combined_clat_dev,combined_clat_pct01,combined_clat_pct02,combined_clat_pct03,comb
ined_clat_pct04,combined_clat_pct05,combined_clat_pct06,combined_clat_pct07,combined_clat_pct08
,combined_clat_pct09,combined_clat_pct10,combined_clat_pct11,combined_clat_pct12,combined_clat_
pct13,combined_clat_pct14,combined_clat_pct15,combined_clat_pct16,combined_clat_pct17,combined_
clat_pct18,combined_clat_pct19,combined_clat_pct20,combined_tlat_min,combined_lat_max,combined_
lat_mean,combined_lat_dev,combined_bw_min,combined_bw_max,combined_bw_agg_pct,combined_bw_mean,
combined_bw_dev,write_kb,write_bandwidth,write_iops,write_runtime_ms,write_slat_min,write_slat_
max,write_slat_mean,write_slat_dev,write_clat_min,write_clat_max,write_clat_mean,write_clat_dev
,write_clat_pct01,write_clat_pct02,write_clat_pct03,write_clat_pct04,write_clat_pct05,write_cla
t_pct06,write_clat_pct07,write_clat_pct08,write_clat_pct09,write_clat_pct10,write_clat_pct11,wr
ite_clat_pct12,write_clat_pct13,write_clat_pct14,write_clat_pct15,write_clat_pct16,write_clat_p
ct17,write_clat_pct18,write_clat_pct19,write_clat_pct20,write_tlat_min,write_lat_max,write_lat_
mean,write_lat_dev,write_bw_min,write_bw_max,write_bw_agg_pct,write_bw_mean,write_bw_dev,cpu_us
er,cpu_sys,cpu_csw,cpu_mjf,cpu_minf,iodepth_1,iodepth_2,iodepth_4,iodepth_8,iodepth_16,iodepth_
32,iodepth_64,lat_2us,lat_4us,lat_10us,lat_20us,lat_50us,lat_100us,lat_250us,lat_500us,lat_750u
s,lat_1000us,lat_2ms,lat_4ms,lat_10ms,lat_20ms,lat_50ms,lat_100ms,lat_250ms,lat_500ms,lat_750ms
,lat_1000ms,lat_2000ms,lat_over_2000ms,disk_name,disk_read_iops,disk_write_iops,disk_read_merge
s,disk_write_merges,disk_read_ticks,write_ticks,disk_queue_time,disk_util,disk_name,disk_read_i
ops,disk_write_iops,disk_read_merges,disk_write_merges,disk_read_ticks,write_ticks,disk_queue_t
ime,disk_util,disk_name,disk_read_iops,disk_write_iops,disk_read_merges,disk_write_merges,disk_
read_ticks,write_ticks,disk_queue_time,disk_util,disk_name,disk_read_iops,disk_write_iops,disk_
read_merges,disk_write_merges,disk_read_ticks,write_ticks,disk_queue_time,disk_util,disk_name,d
isk_read_iops,disk_write_iops,disk_read_merges,disk_write_merges,disk_read_ticks,write_ticks,di
sk_queue_time,disk_util,disk_name,disk_read_iops,disk_write_iops,disk_read_merges,disk_write_me
rges,disk_read_ticks,write_ticks,disk_queue_time,disk_util
```

Note: Below is the information provided by typing "man fio" in your Linux system in regards to the header above.

```
READ status (COMBINED status if using "unified_rw_reporting=1"):
  Total IO (KiB), bandwidth (KiB/sec), IOPS, runtime (msec)
  Submission latency: min, max, mean, stdev (usec)
  Completion latency: min, max, mean, stdev (usec)
  Completion latency percentiles: 20 fields
  Total latency: min, max, mean, stdev (usec)
  Bandwidth (KiB/s): min, max, aggregate percentage of total, mean, stdev
```



WRITE status (these fields will show as 0 if using "unified_rw_reporting=1"):

- Total IO (KiB), bandwidth (KiB/sec), IOPS, runtime (msec)
- Submission latency: min, max, mean, stdev (usec)
- Completion latency: min, max, mean, stdev (usec)
- Completion latency percentiles: 20 fields
- Total latency: min, max, mean, stdev (usec)
- Bandwidth (KiB/s): min, max, aggregate percentage of total, mean, stdev (usec)

CPU usage:

- user, system, context switches, major faults, minor faults

I/O depths:

- <=1, 2, 4, 8, 16, 32, >=64

I/O latencies microseconds:

- <=2, 4, 10, 20, 50, 100, 250, 500, 750, 1000

I/O latencies milliseconds:

- <=2, 4, 10, 20, 50, 100, 250, 500, 750, 1000, 2000, >=2000

Disk utilization:

- disk name, read ios, write ios, read merges, write merges, read ticks, write ticks, time spent in queue, disk utilization percentage



6 Test Results using Intel® VROC and Intel® SSD DC P4510 (2TB capacity)

Results below include IOPS, Bandwidth, and overall CPU usage.

Note: Performance results are based on testing as of October 11, 2018 and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure.

Table 2: Test Results

Workload	VROC 5.4 4DR0	VROC 5.4 4DR5	VROC 5.4 2DR1	VROC 5.4 4DR10
	IOPS	IOPS	IOPS	IOPS
4k_Random_100%-Write_24-Threads_256-IOdepth	327,876	110,070	83,948	167,402
4k_Random_30%-Write_24-Threads_256-IOdepth	719,850	328,118	217,769	431,107
4k_Random_0%-Write_24-Threads_256-IOdepth	2,496,905	1,938,936	1,291,400	2,417,984
Workload	VROC 5.4 4DR0	VROC 5.4 4DR5	VROC 5.4 2DR1	VROC 5.4 4DR10
	Bandwidth (MB/s)	Bandwidth (MB/s)	Bandwidth (MB/s)	Bandwidth (MB/s)
4k_Random_100%-Write_24-Threads_256-IOdepth	1,343	451	344	686
4k_Random_30%-Write_24-Threads_256-IOdepth	2,949	1,344	892	1,766
4k_Random_0%-Write_24-Threads_256-IOdepth	10,227	7,942	5,290	9,904
Workload	VROC 5.4 4DR0	VROC 5.4 4DR5	VROC 5.4 2DR1	VROC 5.4 4DR10
	% CPU Utilization	% CPU Utilization	% CPU Utilization	% CPU Utilization
4k_Random_100%-Write_24-Threads_256-IOdepth	1.12%	2.34%	0.61%	1.27%
4k_Random_30%-Write_24-Threads_256-IOdepth	2.17%	3.26%	1.00%	2.53%
4k_Random_0%-Write_24-Threads_256-IOdepth	9.01%	9.38%	6.41%	16.75%