App note

Optimize Utilization & Reliability for High-Performance Distributed Database Apps

With CSD 3310 SSDs and Xinnor xiRAID

ScaleFlux • XINNOR

Table of Contents

1 About This Document

In this application note, we evaluate the feasibility of using Xinnor® xiRAID™ with ScaleFlux® CSD 3310 NVMe® SSDs in latency-sensitive distributed database applications. We use the Aerospike® certification tool (ACT) for SSDs to evaluate the latency response of an xiRAID level five array (distributed parity with single disk failure protection) using five CSD 3310 7.8TB SSDs.

2 Motivation

Xinnor's xiRAID is a flexible, high-performance software RAID solution ideally suited to the high IOPS and low latency provided by modern NVMe storage. When evaluating a RAID solution, evaluating the performance of the RAID array in the degraded state (one or more drives in the array is not functioning) and reconstructing state (one or more new drives added to the array) is just as critical as evaluating operation in the normal state. In the degraded state, fewer drives are available to serve IO requests. In the reconstructing state, intra-RAID IO adds traffic as data on good disks is used to reconstruct the data to the newly introduced disks. A key feature provided by xiRAID is the ability to control the rate of array reconstruction, enabling a predictable reconstruction workload sized to ameliorate contention with host IO. The ScaleFlux CSD 3310 NVMe SSDs feature transparent inline compression that also lowers the impact of intra-RAID reconstruction IO. By transparently compressing data written to the SSD, additional media bandwidth is available for reads. This capability allows the CSD 3310 to maintain a lower read latency profile in the presence of reconstruction IO. Combining these technologies enables a best-in-class software RAID solution capable of servicing high-performance workloads under all RAID operating states.

3 Test Setup

The test system consists of a dual-socket Xeon Gold 6342 CPU with 48 physical cores and 512GB of DRAM, installed with five 7.68TB ScaleFlux CSD 3310 PCIe® Gen 4 SSDs. The operating system is Ubuntu 22 with kernel version 5.15.0-70.

3.1 xiRAID Setup

Xinnor xiRAID is installed from the Xinnor official release repository using `sudo apt install xiraid-release`. The xicli utility manages all aspects of the RAID array. To create the RAID 5 array, use the following command:

```
$ sudo xicli raid create -n sfxtest -l 5 -d /dev/nvme0n1 /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1 /dev/nvme4n1
```

Next, we adjust the reconstruction priority down to 10%. This adjustment limits the maximum throughput consumed during RAID rebuild such that host IO latency impact is minimized:

```
$ sudo xicli raid modify -n sfxtest --recon-prio=10
```

To view a basic summary of the RAID array, use the `show` command:

```
$ xicli raid show
╔RAIDs═══════╤════════════════╤══════════════╤═══════════════╗
║ name   ║ static        ║ state     ║ devices            ║ info          ║
╟──────────┼──────────────┼──────────────┼───────────────╢
║ sfxtest ║ size: 28615 GiB ║ online      ║ 0 /dev/nvme0n1 online ║ memory_usage_mb   : - ║
║         ║ level: 5        ║ initialized ║ 1 /dev/nvme1n1 online ║                       ║
║         ║ strip_size: 16  ║             ║ 2 /dev/nvme2n1 online ║                       ║
║         ║ block_size: 4096 ║            ║ 3 /dev/nvme3n1 online ║                       ║
║         ║ sparepool: -    ║             ║ 4 /dev/nvme4n1 online ║                       ║
║         ║ active: True    ║             ║                       ║                       ║
║         ║ config: True    ║             ║                       ║                       ║
╚══════════╧══════════════╧══════════════╧═══════════════╝
```

To provide additional information in a similar format, including the reconstruction priority setting, use the -e option with `show`

3.2 Aerospike Certification Tool Setup

The Aerospike certification tool (ACT) is available on GitHub. This tool generates a workload at a specified transactions per second (TPS) target and monitors the tail latency of all transactions. If all transactions are complete while meeting all the latency goals, the test passes at that TPS level. The essential latency goal is that at most 5% of transactions exceed 1 millisecond.

In the Aerospike IO architecture, the number of threads depends on the number of CPU cores available and the number of drive partitions. Divide the RAID array into 16 partitions to increase parallelism:

```
$ lsblk
NAME          MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
xi_sfxtest     259:8    0  27.9T  0 disk
├─xi_sfxtest1  259:9    0   1.7T  0 part
├─xi_sfxtest2  259:10   0   1.7T  0 part
├─xi_sfxtest3  259:11   0   1.7T  0 part
├─xi_sfxtest4  259:12   0   1.7T  0 part
├─xi_sfxtest5  259:13   0   1.7T  0 part
├─xi_sfxtest6  259:14   0   1.7T  0 part
├─xi_sfxtest7  259:15   0   1.7T  0 part
├─xi_sfxtest8  259:16   0   1.7T  0 part
├─xi_sfxtest9  259:17   0   1.7T  0 part
```

```
├─xi_sfxtest10 259:18   0   1.7T  0 part
├─xi_sfxtest11 259:19   0   1.7T  0 part
├─xi_sfxtest12 259:20   0   1.7T  0 part
├─xi_sfxtest13 259:21   0   1.7T  0 part
├─xi_sfxtest14 259:22   0   1.7T  0 part
├─xi_sfxtest15 259:23   0   1.7T  0 part
└─xi_sfxtest16 259:24   0   1.7T  0 part
```

This evaluation sets the TPS level to 1.5 million with a 2:1 read-to-write ratio. Use the compress-pct parameter to configure compressibility. In this case, the configuration achieves approximately a 2:1 compression ratio on disk. The read object size is 1.5kB, but there are also large (128kB) reads and writes performed in parallel to simulate other database processes. The complete configuration file is as follows.

```
#
# ACT-storage config file.
#

# Mandatory device name(s) as comma-separated list:
device-names: /dev/xi_sfxtest1,/dev/xi_sfxtest2,/dev/xi_sfxtest3,/dev/xi_sfxtest4,/dev/xi_sfxtest5,\
/dev/xi_sfxtest6,/dev/xi_sfxtest7,/dev/xi_sfxtest8,/dev/xi_sfxtest9,/dev/xi_sfxtest10,\
/dev/xi_sfxtest11,/dev/xi_sfxtest12,/dev/xi_sfxtest13,/dev/xi_sfxtest14,/dev/xi_sfxtest15, \
/dev/xi_sfxtest16

# Mandatory non-zero test duration:
test-duration-sec: 172800

#--------------------------------------
# Transaction request rates.
#
# The standard "1x" load is 1000 writes and 2000 reads per second. To generate
# a standard "Nx" load, multiply these numbers by N. If testing with more than
# one device, also multiply by the number of devices. (The configured rates are
# spread across all devices in the test.)
#

read-reqs-per-sec: 1000000
write-reqs-per-sec: 500000

#--------------------------------------
# Items with default values.
#
# All remaining configuration items are shown below with default values. To try
# non-default values, just un-comment the relevant items and change the values.
```

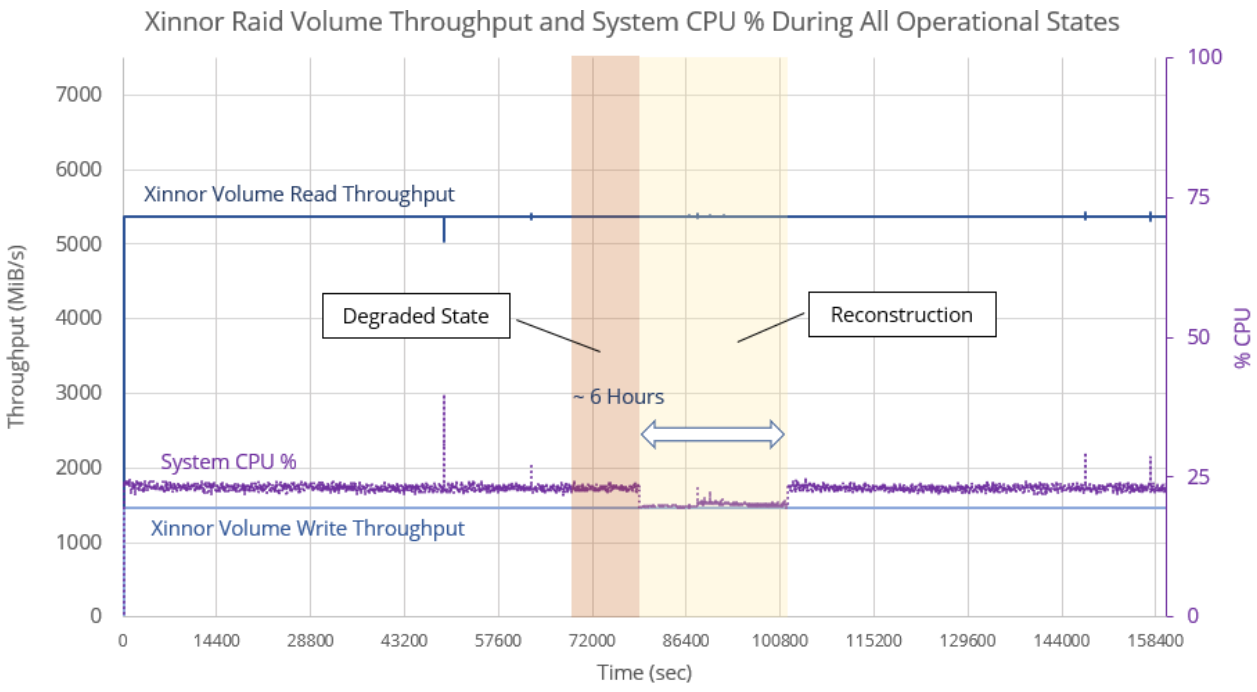
```
# See README.md for more information.
#

# service-threads: 40? # default is 5x detected number of CPUs
# report-interval-sec: 1
# microsecond-histograms: no
# record-bytes: 1536
# record-bytes-range-max: 0
# large-block-op-kbytes: 128
# replication-factor: 1
# update-pct: 0
# defrag-lwm-pct: 50
compress-pct: 40
# disable-odsync: no
# commit-to-device: no
# commit-min-bytes: 512? # default is detected minimum device IO size
# tomb-raider: no
# tomb-raider-sleep-usec: 0
# max-lag-sec: 10
# scheduler-mode: noop
```

4 Test Execution and Results

The ACT workload is applied continuously to measure RAID array performance under four different conditions:

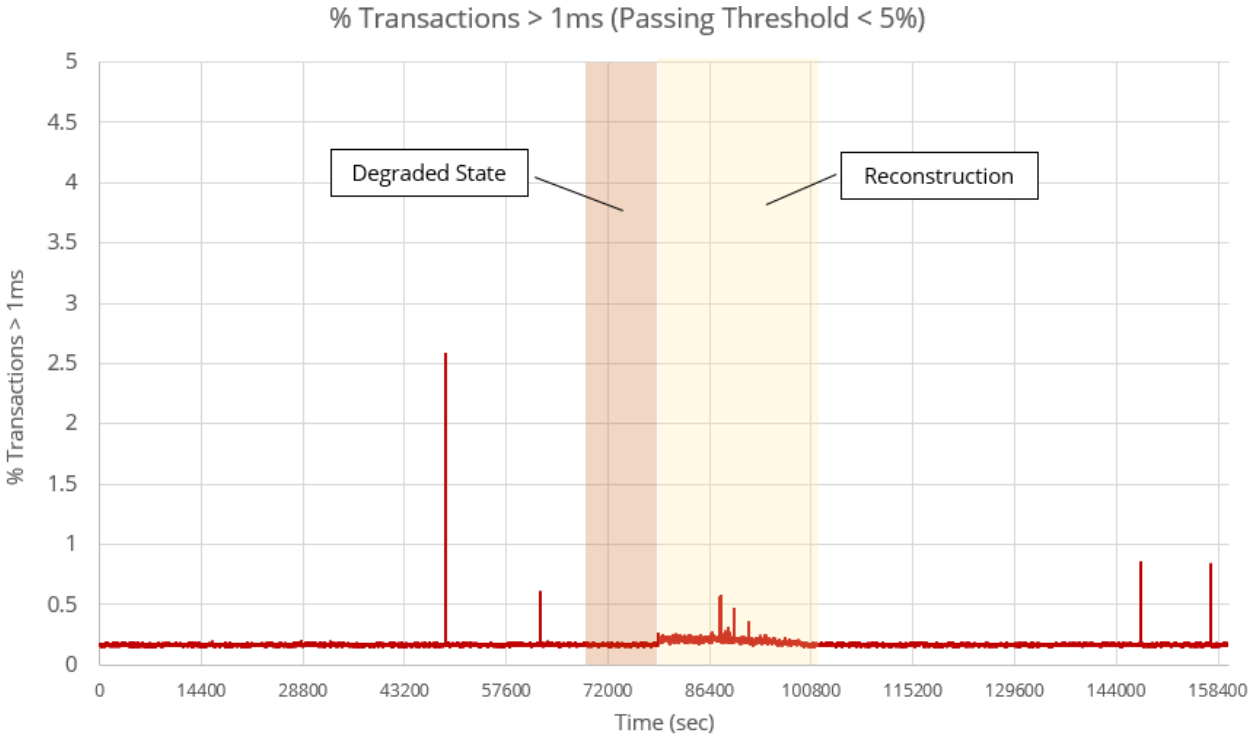
1. Normal operation, followed by a surprise removal of a drive from the array.
2. A surprise removal of the drive, followed by continued operation in the degraded state.
3. Adding an empty drive back into the array and operation in the reconstructing state.
4. Normal operation after full reconstruction.

Ideally, the host should not observe a difference in IO performance between these four states. In the following plot, we observe the read and write throughputs and total system (kernel) CPU utilization across all states:

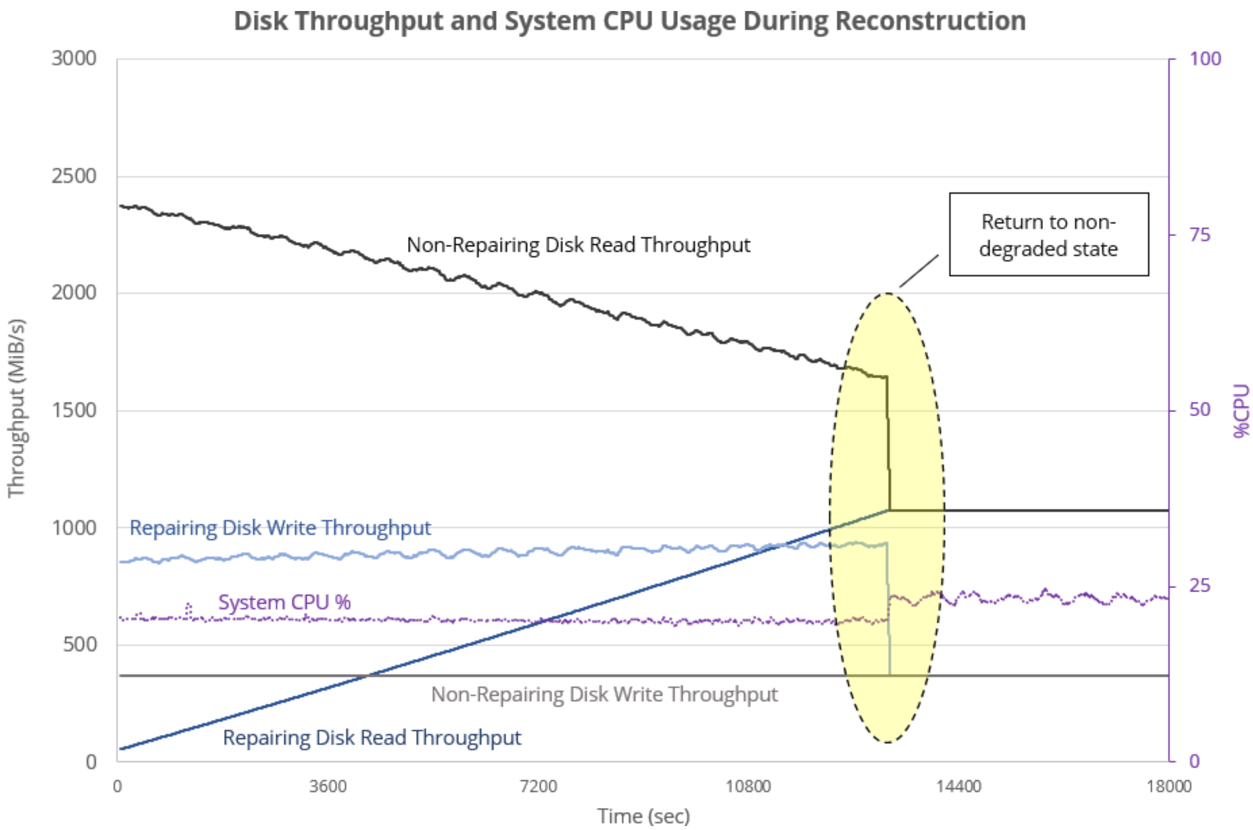


The total read and write throughput are constant (at 5.4GB/s and 1.7GB/s, respectively). There is a slight decline in CPU utilization while the array reconstructs.

Similarly, the tail latency remains remarkably low through all states:



Taking a closer look at the reconstruction phase, the read and write throughput of a normal drive and the drive under reconstruction is measured:



The non-repairing disk write throughput remains constant but serves a higher read throughput as it contributes to the data on the reconstructing disk. The volume of data read from the non-repairing disk decreases as reconstruction progresses as the repairing disk continues to serve data. At the end of the reconstruction phase, both disks contribute equally to the workload.

To monitor the reconstruction progress, use the `show` command (the output below is a partial view of the output.

```
$ xicli raid show
~╓──────────────────────────╥────────────────────────────────────────────────────────────────
─────────────────╥──────────────────────────────────────────╖
~ ║ state         ║ devices                    ║ info             ║
~╟──────────────────────────╫────────────────────────────────────────────────────────────────
─────────────────╫──────────────────────────────────────────╢
~ ║ online        ║ 0 /dev/nvme0n1 online               ║ recon_progress    : 80 ║
~ ║ degraded      ║ 1 /dev/nvme1n1 online               ║ memory_usage_mb   : -  ║
~ ║ reconstructing ║ 2 /dev/nvme2n1 online              ║                   ║
~ ║ initialized   ║ 3 /dev/nvme3n1 online               ║                   ║
~ ║               ║ 4 /dev/nvme4n1 online,reconstructing ║                  ║
~ ║               ║                                     ║                   ║
```

5 Conclusion

Maintaining consistent performance and low latency requires that the impact of the added IO traffic due to reconstruction does not affect host IO performance. Combining xiRAID's tunable reconstruction priority and minimizing the impact of reconstruction traffic through transparent compression combine to provide a consistently low impact on host IO performance.

With a high-performance and resilient RAID array suitable for low-latency applications, deploying a RAID solution can reduce the probability of node failure and avoid costly node-level rebuilds. Furthermore, the increased node data reliability may make it feasible to reduce the amount of node-level redundancy. For example, deploying double replication in place of triple replication. Such an implementation would reduce server count and rack space, lower network utilization, and decrease the quantity of any required node-level licenses.

About Xinnor

Xinnor is an Israeli-based software development company that specializes in creating innovative data storage solutions. Our main product is xiRAID, a patented software RAID technology that delivers exceptional performance. xiRAID is a product of a decade of math research, unique algorithms of data protection and in-depth knowledge of modern CPU operation. Although it works with all types of storage devices, xiRAID really shines when deployed together with NVMe® or NVMe-oF™ devices. xiRAID is the only software solution in the market capable of driving up to 97% of raw device performance in computationally heavy RAID configurations, while maintaining a very modest load on the host CPU and low memory footprint.

> *The fastest and most reliable software RAID*

XINNOR

+972 43 740 203

request@xinnor.io

www.xinnor.io

> *The Better SSD delivered to your door.*

ScaleFlux

About ScaleFlux

ScaleFlux helps customers harness data growth as a competitive advantage by building products that reduce complexity and accelerate the creation of value from data. In our first phase of rethinking the data pipeline for the modern data center, ScaleFlux has built a better SSD by embedding computational storage technology into flash drives. Now, customers can gain an edge, optimizing their data center infrastructure by deploying storage intelligence for workloads like databases, analytics, IoT, and 5G.

sales@scaleflux.com

www.scaleflux.com