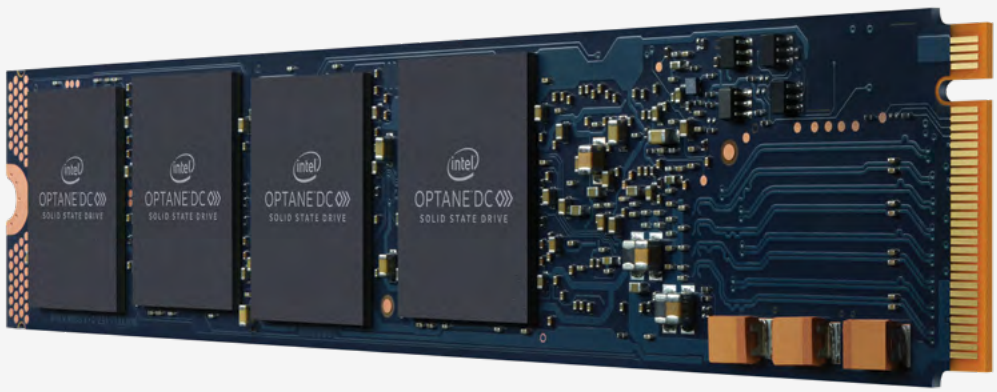
sysclay®

White Paper

ZFS acceleration with Intel® Optane™ SSDs



intel®

Created January 2020
Reviewed October 2020

Introduction

Intel® Optane™ - is a brand name for devices based on the 3D XPoint technology developed jointly by Intel® and Micron®. It utilizes new non-volatile memory based on changing materials bulk resistance – this is the only thing we know because neither Intel® nor Micron® explained this technology in a more detailed way.

There are two main advantages of drives using this technology:

- speed (few times faster than flash-based drives)
- endurance reaching 30-60 DWPD (usually 1-3 for standard flash drives, sometimes 5 for tougher ones)

OpenZFS

ZFS is a quite popular file system designed in the early 2000s by Sun engineers, with a surprisingly wide range of features. Its design concept to combine filesystem and logical volume manager is really one of a kind, allowing to manage volumes in a very flexible way. It supports different RAID levels (striping and mirroring), snapshots, compression, deduplication, integrity checking, automatic repairs and it is scalable to yobibytes (up to 2128 bytes). That makes it a perfect solution for storage servers: thanks to hierarchical checksumming stored data is secured against drive physical errors or any kind of logical ones like OS misprocessing and even possible data loss due to corruption, which can happen over time.

ZFS may be a great solution itself, it may have similar performance to RAID controllers, but it is still limited by hardware. When you use standard mechanical drives or even SSDs, you are still limited by their performance.

Our goal is to check how much we will be able to speed up HDD-based ZFS storage using Intel® Optane™ drives. We will also compare results to SSD-based ZFS storage. We will not be focusing on exact numbers like the maximum number of IOPS. Instead, we rather want to verify how much more performance we can get out of ZFS with Intel® Optane™ help.

Testing procedure

ZFS write cache

First, before we can tell more about the tests itself, we need to explain what **ZIL** and **SLOG** are. **ZIL** is ZFS Intent Log – a place where ZFS inserts data before writing it to a final destination. In a basic configuration, ZIL is located on the same devices, which we use to store data. This means data is written twice. The second acronym (**SLOG**) is Second ZFS Intent Log - a separate device acting as a write cache.

So, to get most of **SLOG**, we need to use high-performance drive, like, as you may suspect, Intel® Optane™. That's the place where this drive improves ZFS performance at most.

Test steps and tools

We have performed tests on three different drives configurations:

1. First, we have tested ZFS performance on with no read and write cache.
2. Second, we have configured a flash-based enterprise SSD drive (Intel® S4510) as a **SLOG** device and performed the test again.
3. And at last, we configured Intel® Optane™ drives – and tested once again.

As a test tool, we use **fio** version 3.1, the default one for Ubuntu 18.04, with a test configuration shown below. We were running the tests for at least 5 minutes each.

We were focusing on IOPS during tests because it's the main limitation – bandwidth usually it's just block size multiplied by IOPS, which I will prove later.

```
[global]
sync=1
numjobs=1
iodepth=32
group_reporting
bs=4k
time_based=1
filesize=4G

[hdd-write]
filename=/mnt/hdd-pool/test
rw=write

[ssd-write]
filename=/mnt/ssd-pool/test
rw=write

[hdd-randwrite]
filename=/mnt/hdd-pool/test
rw=randwrite

[ssd-randwrite]
filename=/mnt/ssd-pool/test
rw=randwrite
```

Test environment

Hardware configuration

As a test platform, we will use a 2U server equipped with a Supermicro X11 motherboard, with a total of 12 drives: 4 x 4TB mechanical drives, 4 x SSDs, 2 x Intel® Optane™ NVMe drives, and 2 x 120GB consumer-grade SSDs just for OS. In the server, we installed 2 x Intel® Xeon® Gold 5218 and 256GB RAM to avoid any hardware bottlenecks. Ubuntu 18.04 was installed on the test server and packages were updated.

MOTHERBOARD	Supermicro X11DDW-NT, BIOS
CPU	2 x Intel® Xeon® Gold 5218
RAM	256GB @ 2666 MHz
HDDS `hdd-pool`	4 x HGST Ultrastar 7K6000 4TB 3.5'' 7200 RPM (HUS726040AL)
SSDS `ssd-pool`	4 x Intel S4510 3.84TB 2.5'' (SSDSC2KB03)
HARDWARE RAID	None
OS DRIVES	2 x Intel 535 120GB 2.5'' (SSDSC2BW12)
OPERATING SYSTEM	Ubuntu 18.04, kernel 4.15.0-74

We created two ZFS pools: `hdd-pool` from 4 x HDD drives and `ssd-pool` from SSDs. Devices were configured resembling RAID10 as striped mirrored `vdevs`.

```
$ zpool status

  pool: hdd-pool
 state: ONLINE
  scan: none requested
config:

    NAME          STATE        READ                WRITE                CKSUM
    hdd-pool      ONLINE          0                    0                    0
      mirror-0    ONLINE          0                    0                    0
        sdh       ONLINE          0                    0                    0
        sdi       ONLINE          0                    0                    0
      mirror-1    ONLINE          0                    0                    0
        sdj       ONLINE          0                    0                    0
        sdk       ONLINE          0                    0                    0
errors: No known data errors

  pool: ssd-pool
 state: ONLINE
  scan: none requested
config:

    NAME          STATE        READ                WRITE                CKSUM
    ssd-pool      ONLINE          0                    0                    0
      mirror-0    ONLINE          0                    0                    0
        sdd       ONLINE          0                    0                    0
        sde       ONLINE          0                    0                    0
      mirror-1    ONLINE          0                    0                    0
        sdf       ONLINE          0                    0                    0
        sdg       ONLINE          0                    0    0Pre-configuration
```

Turn off cache

When `fio` is run with out of the box configuration you will experience strange results:

```
$ fio --section=hdd-write --runtime=15 ~/tests.fio

(...)

  write: IOPS=33.0k, BW=133MiB/s (139MB/s)(1992MiB/15004msec)
```

Mechanical drives, even during sequential writes and even in RAID10 configuration, should not reach over 30 000 IOPS. It's just not physically possible for those drives to achieve such speeds. Their performance should be around 100-200 IOPS.

This is happening because data is saved asynchronously and is just stored in RAM before written to disk. Speeds we observe have little to do with drives itself. The reason is standard ZFS configuration and `sync` property set to `standard`.

```
$ zfs get sync hdd-pool

NAME      PROPERTY  VALUE     SOURCE
hdd-pool  sync      standard  default
```

Let's check in `zfs` manual:

```
$ man zfs

(...)

sync=standard|always|disabled
Controls the behavior of synchronous requests (e.g. fsync, O_DSYNC).

standard is the POSIX specified behavior of ensuring all synchronous requests are
written to stable storage and all devices are flushed to ensure data is not cached
by device controllers (this is the default).

always causes every file system transaction to be written and flushed before its
system call returns. This has a large performance penalty.

disabled disables synchronous requests. File system transactions are only
committed to stable storage periodically. This option will give the highest
performance. However, it is very dangerous as ZFS would be ignoring the
synchronous transaction demands of applications such as databases or NFS. Adminis-
trators should only use this option when the risks are understood.
```

So we need to change `sync` property to `always` to ensure what we observe is ZFS using drives, not RAM.

```
$ zfs set sync=always hdd-pool

$ zfs get sync hdd-pool

NAME      PROPERTY  VALUE     SOURCE

hdd-pool  sync      always    local
```

Now we test again:

```
$ fio --section=hdd-write --runtime=15 ~/tests.fio

(...)

  write: IOPS=140, BW=563KiB/s (576kB/s)(8448KiB/15007msec)
```

And now, results are much closer to reality.
Also, we will force `fio` to use synchronous writes. Please note `fio` config file:

```
$ cat tests.fio | grep sync

sync=1
```

By the way, please look closer at the bandwidth results. As was mentioned before, we focus on measuring IOPS, as bandwidth is usually just a block size multiplied by number of IOPS:

```
33.0k IOPS * 4kB = 132MB/s
(BW=133MiB/s on first test)

140 IOPS * 4kB = 560KB/s
(BW=563KiB/s on second one)
```

Turn off cache again

ZFS adjustable replacement cache (or ARC) is another unique feature of this filesystem. It stores the data first in RAM and then on an additional designated device, usually a high-performance SSD drive (L2ARC). As you may suspect, Intel® Optane™ again is the perfect choice to be used as L2ARC. This cache is one of the reasons why ZFS likes systems with a LOT of RAM).

To avoid cache interference during the tests, we need to disable ARC (and as a result, L2ARC too) because results will be disrupted again:

```
$ zfs get primarycache hdd-pool

NAME      PROPERTY      VALUE  SOURCE
hdd-pool  primarycache  all    local

$ fio --section=hdd-read --runtime=15 ~/tests.fio

(...)

   read: IOPS=42.9k, BW=168MiB/s (176MB/s)(2515MiB/15012msec)
```

After disabling **ARC** and **L2ARC**, results make more sense:

```
$ zfs set primarycache=none hdd-pool

$ zfs set secondarycache=none hdd-pool

$ zfs get all hdd-pool | grep cache

hdd-pool  primarycache    none   local
hdd-pool  secondarycache  none   local

$ fio --section=hdd-read --runtime=15 ~/tests.fio

(...)

   read: IOPS=2600, BW=10.2MiB/s (10.7MB/s)(152MiB/15001msec)
```

Turn off compression and deduplication

Same reasons as before, no need to explain.

```
$ zfs get dedup hdd-pool

NAME      PROPERTY  VALUE    SOURCE
hdd-pool  dedup       off      local

$ zfs get compression hdd-pool

NAME      PROPERTY     VALUE   SOURCE
hdd-pool  compression  off     local
```

Configuration of SLOG

On drives to be used for SLOG, we created a partition of 48GB and attached it to the ZFS pool:

```
$ zpool add hdd-pool log nvme1n1p2
$ zpool status hdd-pool
  pool: hdd-pool
 state: ONLINE
  scan: none requested
config:

    NAME          STATE        READ                 WRITE                CKSUM
     hdd-pool     ONLINE          0                    0                    0
      mirror-0    ONLINE          0                    0                    0
         sdg      ONLINE          0                    0                    0
         sdh      ONLINE          0                    0                    0
      mirror-1    ONLINE          0                    0                    0
         sdi      ONLINE          0                    0                    0
         sdj      ONLINE          0                    0                    0
    logs
     nvme1n1p2    ONLINE          0                    0                    0
errors: No known data errorsTest results
```

You can find the results of tests with different drive configurations below. Please note again that the goal of this test was not the performance itself, but verification of how much we can benefit from using Intel® Optane™. As you can see, using Intel® Optane™ allows us to get maximum performance of HDD drives in ZFS, comparable or sometimes better than pure SSD configuration. We also calculated speed gain compared to HDD.

	SEQUENTIAL WRITES		RANDOM WRITES	
DRIVES CONFIGURATION	IOPS	SPEED GAIN	IOPS	SPEED GAIN
4 x HDD 10TB RAID10	110	---	89.52	---
4 x HDD 10TB RAID10 + Intel® Optane™ P4801X SLOG	3922	36 x	528.36	6 x
4 x SSD 3.84TB RAID10 Intel® S4510	3175	29 x	1327.12	15 x
Intel® Optane™ P4801X 200G standalone (ZFS pool)	3424	31 x	2259.43	25 x

Cost per IOPS analysis

We have calculated the approximate cost per TB per IOPS using estimated prices, i.e., from sellers like newegg.com.

```
Price per IOPS = Price per GB / IOPS test result
```

		PRICE PER IOPS	
DRIVES CONFIGURATION	PRICE PER TB	SEQUENTIAL WRITE	RANDOM WRITE
4 x HDD 10TB RAID10	$76	$0.69	$0.84
4 x HDD 10TB RAID10 + Intel® Optane™ P4801X SLOG	$83	$0.03	$0.19
4 x SSD 3.84TB RAID10 Intel® S4510	$453	$0.14	$0.34
Intel® Optane™ P4801X 200GB standalone (ZFS pool)	$8 540	$2.49	$3.78

Conclusion

As you can see, we can draw two main conclusions. Using Intel® Optane™ as a SLOG drive for HDD allows us to achieve similar speeds during sequential writes as on pure SSD configurations. Random writes will be 2.5 times slower compared to pure SSD, but still over 6 times higher than using HDDs without SLOG. So when we need great performance, SSDs are unbeatable, but their price per IOPS is about 2-3 times higher than on HDD with Intel® Optane™ as SLOG configuration.