Hyper-converged infrastructure based on Proxmox VE accelerated with Bcache

Performance analysis on a three-node Ceph cluster with three-OSDs' pools

IORUX

PREFACE

IORUX is an IT and engineering company founded in 2017 and based in Huelva, in the south of Spain. We develop our activities mostly employing Free Software alternatives. But the Free Software or Open Source philosophy cannot be applied to the hardware, sadly. This is why we decided to contribute by designing datacenter-class and vendor-lock free servers with the Proxmox software in mind.

The goal of this paper is to demonstrate how powerful Free Software can be alongside an assembled hardware that is specifically built focused on the requirements of this software.

TEST BED CONFIGURATION

The tests described in this report were performed on hardware components that were specifically chosen for the Ceph workload. The evaluation took place in May 2020 in environmental conditions meant to subject the hardware to worst-case scenarios so that results will always be better when the hardware is placed in closer to true-world scenarios in which optimal datacenter conditions are observed.

Hardware specifications

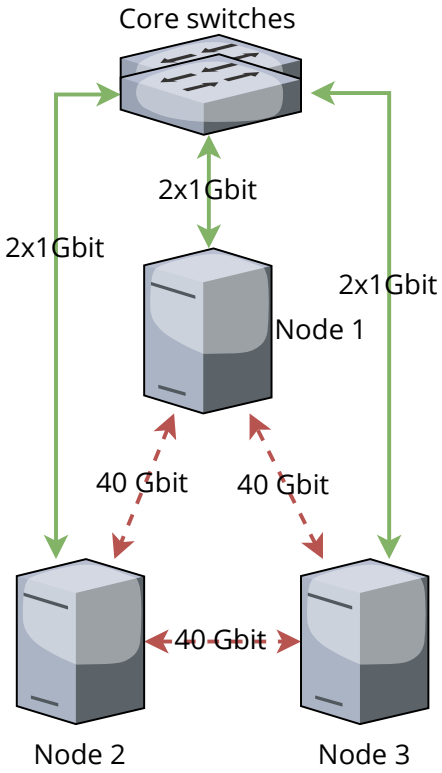
A minimum of three identical nodes were used for the benchmarking with this specifications:

CPU	AMD Ryzen 5 3600 3,6 GHz 6/12 2666 MHz
Mainboard	ASRock Rack X470D4U
Chassis	IORUX Ryzen ATX Tower 2020
Dual 40 Gbit NIC	Intel XL710-QDA2
Memory	2 x 32 GB DDR4 2666 MHz non-ECC

Table 1: System characteristics.

Network

This cluster has been deployed using a 40 Gbit ring network over the three nodes bridged with the 1 Gbit accessing network. Linux bridges and RSTP handled the network.



Ilustration 1: Network schema.

A inter-node performance test using iperf3 is shown:

```
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  4.63 GBytes  39.8 Gbits/sec    0   14.2 MBytes
[  5]   1.00-2.00   sec  4.61 GBytes  39.6 Gbits/sec    0   14.2 MBytes
[  5]   2.00-3.00   sec  4.61 GBytes  39.6 Gbits/sec    0   14.2 MBytes
[  5]   3.00-4.00   sec  4.61 GBytes  39.6 Gbits/sec    0   14.2 MBytes
[  5]   4.00-5.00   sec  4.61 GBytes  39.6 Gbits/sec    0   14.2 MBytes
[  5]   5.00-6.00   sec  4.61 GBytes  39.6 Gbits/sec    0   14.2 MBytes
[  5]   6.00-7.00   sec  4.61 GBytes  39.6 Gbits/sec    0   14.2 MBytes
[  5]   7.00-8.00   sec  4.61 GBytes  39.6 Gbits/sec    0   14.2 MBytes
[  5]   8.00-9.00   sec  4.61 GBytes  39.6 Gbits/sec    0   14.2 MBytes
[  5]   9.00-10.00  sec  4.61 GBytes  39.6 Gbits/sec    0   14.2 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  46.1 GBytes  39.6 Gbits/sec    0             sender
[  5]   0.00-10.00  sec  46.1 GBytes  39.6 Gbits/sec                  receiver
```

Software version

At the moment of the benchmarking (May 2020), Proxmox VE was on its version 6.2-1, pve kernel 5.4.41-1, Ceph version 14.2.9 Nautilus and bcache-tools 1.0.8-3.

Storage for OSDs

All storage attached to the Ceph cluster is datacenter and enterprise class. It features power-loss protection systems, high performance and high endurance characteristics.

Results from a 4k fio (Flexible I/O test utility) test are shown in the following table:

Disk	Bandwidth (KB/s)	IOps	Latency (ms)
Samsung PM983 960GB NVME M.2	193536	48300	0.02
Samsung PM883 240GB SATA III	91750	22900	0.04
Toshiba MG05ACA800E 8TB SATA III	479	119	8.35

Table 2: fio test results.

The storage schema consists of three PM983 (one OSD per node) for a NVME pool; and three PM883 (cache) alongside three MG05ACA800E (backing) -that means, a pair belongs to each node as a single OSD- as a HDD pool.

The command used for the testing is as follows (**warning, it will destroy containing data on the disk**):

```
fio --ioengine=libaio -filename=/dev/xxxx --direct=1 --sync=1 --rw=write --bs=4K
--numjobs=1 --iodepth=1 --runtime=60 --time_based --group_reporting --name=fio --
output-format=terse,json,normal --output=fio.log --bandwidth-log
```

More information about OSD caching using Bcache can be found in the *Annex I. Bcache setup and tuning* of this document.

CEPH PERFORMANCE BENCHMARK

Performance data were modelled using the rados bench test utility from Ceph. The next commands where executed along its pool modifier:

4M	Write	rados bench 60 write -b 4M -t 16 --no-cleanup -p *pool*
	Seq. read	rados bench 60 seq -t 16 -p *pool*
	Rand. read	rados bench 60 rand -t 16 -p *pool*
4K	Write	rados bench 60 write -b 4K -t 16 --no-cleanup -p *pool*
	Seq. read	rados bench 60 seq -t 16 -p *pool*
	Rand. read	rados bench 60 rand -t 16 -p *pool*

Table 3: Ceph pool benchmark commands.

NVME performance

NVME pool	rados 4M		rados 4K	
	Bandwidth (MB/s)	IO/s	Bandwidth (MB/s)	IO/s
Writes	1127.29	281	74.2538	19008
Sequential reads	2306.51	576	225.545	57739
Random reads	2370.44	592	281.929	72173

Table 4: NVME pool benchmark results.

HDD performance (Bcache disabled)

HDD pool (no cache)	rados 4M		rados 4K	
	Bandwidth (MB/s)	IO/s	Bandwidth (MB/s)	IO/s
Writes	187.696	46	0.917933	234
Sequential reads	410.382	102	188.835	48341
Random reads	448.003	112	311.984	79867

Table 5: HDD (no cache) pool benchmark results.

HDD+SSD Bcache performance

HDD pool (SSD cache)	rados 4M		rados 4K	
	Bandwidth (MB/s)	IO/s	Bandwidth (MB/s)	IO/s
Writes	310.593	77	35.0302	8967
Sequential reads	1382.68	345	215.365	55133
Random reads	1471.8	367	303.95	77811

Table 6: HDD (with cache) pool benchmark results.

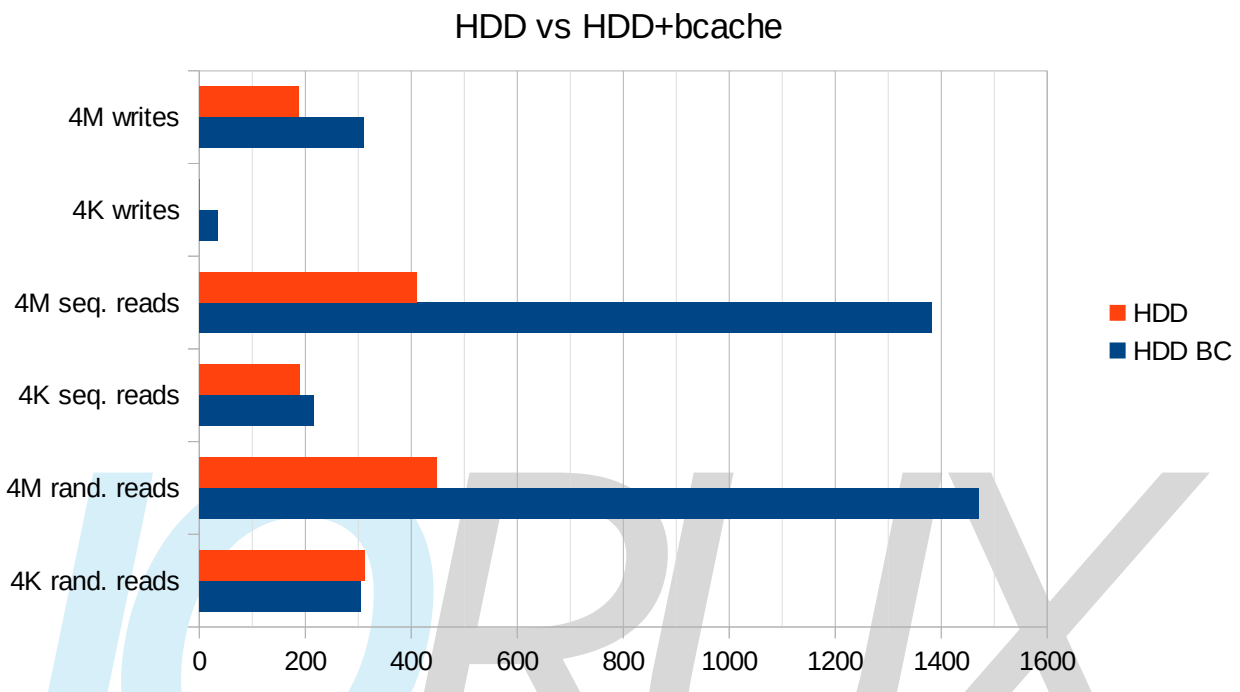
Conclusions

Results show a good performance on the NVME pool even with a three-node and 3 OSDs configuration.

The same could be said for the HDD cached pool as it could be used as storage for VDI VMs, pre-production testing machines, file servers, etc. This configuration allows a high storage density and a reasonably good performance at a reasonable price point.

None of the above could be applied when 3 mechanical disks are attached to the cluster without any kind of cache. Its 4K write performance is horribly bad. That means that if an application needs to perform some writes and seeks at the same time, it will not run correctly. This configuration should be avoided in general and in virtualisation specifically.

A comparison chart is shown:



On the other hand, network has not been a bottleneck on any of the tests. A four node cluster could run safety in this configuration, even if sharing some bandwidth with cluster tasks and VM traffic. Of course, with every OSD added, more bandwidth for Ceph is needed. So, this topology needs to be used carefully.

Other kind of cache techniques had been considered and tested before choosing Bcache. At the first time, a Ceph SSD tiering was deemed as an option, but it was discarded because of its low improvement in performance. Lvmcache was evaluated too as a dm-cache hot-spot cache device (dm-writecache is not supported for the LVM version distributed by Proxmox), with similar performance as the Ceph SSD tiering.

ANNEX I. BCACHE SETUP AND TUNING

Bcache module is available as module in the PVE kernel. In the same way, the package bcache-tools can be installed from the Debian repositories. Once installed, the cached and backing disks must be initialized. It can be done by executing the following command:

```
make-bcache -B /dev/sdX -C /dev/sdY
```

Where sdX is the backing device (the slow spinning disk) and sdY is the cache disk (fast SSD). This command also configures the system for attaching the cache disk to the backing disk at booting time automatically.

Some parameters should be tuned for getting the cache working correctly:

- *sequential_cutoff*: this parameter bypasses writing big sequential files to the cache and sends it to the backing device. With large cache devices it may be unfavourable and should be set to 0. But if the cache gets steadily filled, it may be raised. Default is 4M.
- *cache_mode*: it defines the cache method. For writes improvement, it has to be set to "writeback". The cache can be got around by setting it to "none".
- *congested_read_threshold_us* and *congested_write_threshold_us*: these values order bcache to distribute evenly the work between the caching device and the backing (final storage) device if its latency is getting higher than the specified values. We found 0 as the better value for both.
- *writeback_percent*: bcache tries to maintain the value (percentage) specified of the cache dirty by throttling background writeback and using a PD controller to adjust the rate. We set this value to 80 (percent).

This parameters can be set at boot time by creating the file /etc/tmpfiles.d/bcache.conf and pasting this lines:

```
w /sys/block/bcacheX/bcache/sequential_cutoff             - - - - 0
w /sys/block/bcacheX/bcache/cache_mode                    - - - - writeback
w /sys/fs/bcache/bacheID/congested_read_threshold_us      - - - - 0
w /sys/fs/bcache/bcacheID/congested_write_threshold_us    - - - - 0
w /sys/block/bcacheX/bcache/writeback_percent             - - - - 80
```

BcacheX and bcacheID must be replaced with the correct one for the system.

Proxmox will not be able to create an OSD on top of a bcache device. It must have been done manually.

First, the volume will be provisioned as a BlueStore device:

```
ceph-volume lvm prepare --bluestore -data /dev/bcacheX
ceph-volume lvm activate ID FSID
```

The ID and FSID can be obtained with the following command if needed:

```
ceph-volume lvm list
```

Then, the OSD has to be added to the cluster:

```
ceph auth del osd.{osd-num} # It may be necessary on some cases.
ceph auth add osd.{osd-num} osd 'allow *' mon 'allow rwx' -i
/var/lib/ceph/osd/ceph-{osd-num}/keyring
ceph osd crush add osd.{osd-num} {some.weight} host={host-osd-belongs-to}
```

A class should be assigned to the OSD also. Something like this should suffice:

```
ceph osd crush set-device-class hdd osd.{osd-num}
```

Finally, the OSD is ready to be brought into a pool.