NVME RAID cache optimisations on (V)RAM

- Extending conventional RAM with NVME based RAM
- Using a highly optimized and cache-buffered NVME raid set

In this experiment the following system was used

- AMD Threadripper 1920x with 32 GB (8*4GB) RAM memory
- 256GB SSD system drive
- 7*256GB NVME PCIE3 x4 SSD
- Ubuntu 23.04 OS and a 5.19 kernel

In this test we used the following file systems

- ZFS
- EXT4

In the test we used the following packages

- MDADM 4.2
- ZPOOL 4.1.7
- BCACHE 1.08
- NVidia-Docker 2.11
- ZSWAP

For monitoring system processes we used

- Glances
- NVTop

In the test we used the Qrackmin docker image that we started with the following settings

- docker run -m=4G --memory-swap 1T --device=/dev/dri:/dev/dri --mount type=bind,source=/var/log/qrack,target=/var/log/qrack -d twobombs/qrackmin:pocl

As benchmark we use a Qrack benchmark script running inside the docker container

- ./benchmarks --optimal-cpu --max-qubits=40 --benchmark-depth=400 --single test_stabilizer_t_cc_nn --samples=1
- Separation is set at a reasonable 0.2 to allow for a gradual building of swap reqs
- 40 qubits @FP64 generates a state vector of 8TB, exhausting memory
- 7*256GB raid set on both SWAPFS and ZFS

To set a baseline we run the Qrack benchmark script without additional buffering or cache optimisations

No RAID - 1TB SATA SSD

DISK I/O	R/s	W/s	CPU% >0.2
nvme0n1	0	0	5.6
nvme1n1	0	0	0.0
nvme2n1	0	0	0.0
nvme3n1	0	0	0.0
nvme4n1	0	0	0.0
nvme5n1	0	0	0.0
nvme6n1	0	0	0.0
sda	0	0	0.0
sda1	0	0	0.0
sda2	0	0	0.0
sda3	0	0	0.0
sdb	0	42.7M	0.0
zd0	0	0	0.0

Typical throughput of one SSD SATA 6Gbit
0MB read / 42MB write @50% IO wait and 25 load
No detectable Qrack activity (< 0.1%)

MDADM + SWAP

DISK I/O	R/s	W/s	CPU% >1098
md0	475M	481M	0.0
nvme0n1	67.9M	68.3M	1.0
nvme1n1	67.9M	68.4M	0.0
nvme2n1	67.8M	69.1M	11.0
nvme3n1	67.7M	68.6M	0.5
nvme4n1	67.8M	68.7M	3.2
nvme5n1	67.8M	68.8M	4.4
nvme6n1	67.9M	68.9M	0.2

Typical throughput per NVME stick
67MB read / 68MB write @20% IO wait and 24 load
creating room for 1100% Qrack

ZFS + SWAP

			CPU%
DISK I/O	R/s	W/s	>61.3
nvme0n1	2.07M	3.35M	5.1
nvme0n1p1	2.07M	3.35M	1.7
nvme0n1p9	0	0	1.9
nvme1n1	2.26M	3.34M	1.9
nvme1n1p1	2.26M	3.34M	1.9
nvme1n1p9	0	0	1.9
nvme2n1	2.12M	3.33M	1.9
nvme2n1p1	2.12M	3.33M	1.9
nvme2n1p9	0	0	1.7
nvme3n1	2.21M	3.40M	1.9
nvme3n1p1	2.21M	3.40M	1.9
nvme3n1p9	0	0	1.9
nvme4n1	2.13M	3.16M	1.9
nvme4n1p1	2.13M	3.16M	1.7
nvme4n1p9	0	0	1.9
nvme5n1	2.01M	3.32M	1.9
nvme5n1p1	2.01M	3.32M	1.7
nvme5n1p9	0	0	1.7
nvme6n1	2.08M	3.41M	1.9
nvme6n1p1	2.08M	3.41M	1.7
nvme6n1p9	0	0	1.9

Typical throughput per NVME stick
~2.3MB read / 3.3MB write @70% IO wait and 56 load
creating room for ~60% Qrack

Baseline review

- Regular SSD swap can’t offset Qrack
- SSD swap memory usage is so low it can’t even be measured

- The latency costs for running Qrack in swap on ZFS vs MDADM are more then 18x
- For now, the use of MDADM in raid0 for swap is highly preferred.

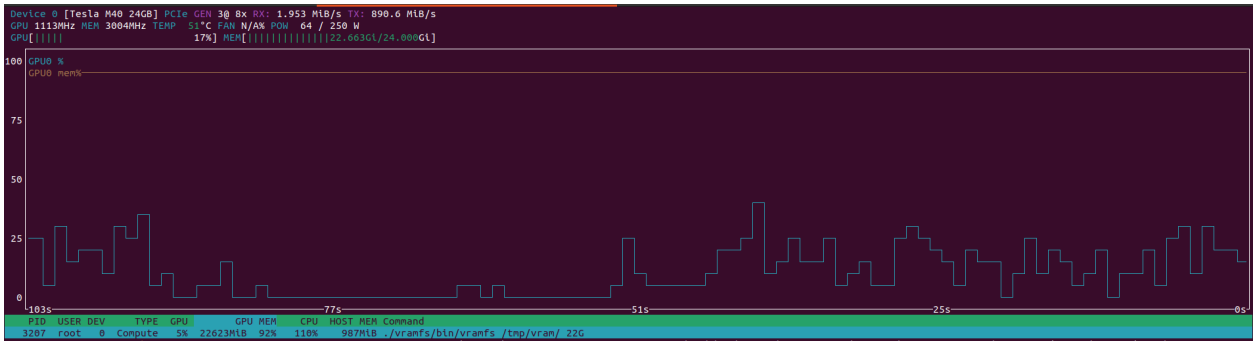
Step 2: MDADM - raid 0 set with VRAM from a Tesla M40

Now we are going to create a buffered MDADM raid set and do cache optimisations for the virtual memory paging requests by using the bcache kernel module and vRAM of a GPU

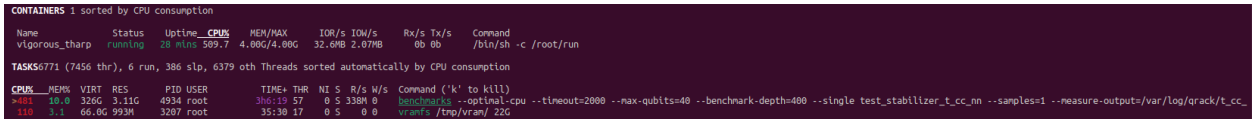
Because the Qrack container is limited by 4GB maximum RAM other parts of the System are, even at high CPU load, fully available and responsive.

The vRAM serves as a buffer for the read/write calls from the container to consolidate the chunks of random virtual memory IO into more sequential blocks, and caching requests, thus streamlining IO.

For this we create a RAM disk and connect the video memory as a buffer to the NVME sticks



We again create a 40 qubits TCC_NN run with 400 layers to make sure we exhaust all memory



As the both the vram buffer and the NVME raid set fills up we see a typical 20/70% efficiency

```
root@thr34d:~# cat /sys/fs/bcache/b978dc54-f5d7-4b10-bd3f-a8e802134a09//cache0/priority_stats
Unused:          6%
Clean:           71%
Dirty:           22%
Metadata:        0%
Average:         85
```

The Tesla M40 has more throughput than the memory demands, also optimizing disk pressure

Conclusion

We can use vRAM from a GPU as a buffer for swap to help streamline and increase performance of swap IO in a high performance environment. A COW FS such as ZFS, interesting for compression and optimisations in day-to-day use, is not a good filesystem for these sorts of simulations. Latency is, as mentioned, 18x higher on ZFS than on MDADM.

Any and all questions or remarks can be made by emailing me directly at ablaauw@gmx.net , file a github request