1-client-intel-nvme-tcp-sw

System description

MangoBoost

August 21, 2024

Contents

1 Hardware

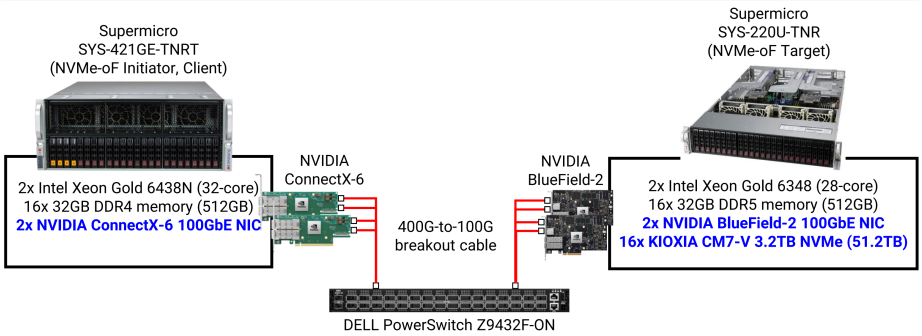

Figure 1: System under test.

Figure 1 shows the system under test. One client and one storage server are connected to the same switch. The client and server has 400G aggregate bandwidth with four 100G ports.

1.1 Client

The client is a commercial Supermicro GPU SuperServer SYS-421GE-TNRT.

1.2 Server

The server is a commercial Supermicro Ultra SuperServer SYS-220U-TNR.

1.3 Switch

The switch is a commercial DELL PowerSwitch Z9432F-ON.

2 Software

2.1 Client

The OS is Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-101-generic x86_64). The client uses the stock Linux `nvme` and `mlx5` drivers to establish NVMe/TCP connections.

2.2 Server

The OS is Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-101-generic x86_64). We use Storage Performance Development Kit v23.9 (SPDK) to provide NVMe/TCP target to the client.

3 Settings

3.1 Client

The client connects 16 NVMe devices in the storage server through NVMe/TCP. Each device is seen as a local NVMe device at /dev/nvme* on the client.

```
sudo nvme connect-all -a 100.100.1.71 -t tcp -s 4420
sudo nvme connect-all -a 100.100.2.71 -t tcp -s 4420
sudo nvme connect-all -a 100.100.3.71 -t tcp -s 4420
sudo nvme connect-all -a 100.100.4.71 -t tcp -s 4420
```

Then we configured RAID0 to fully exploit the aggregate performance of NVMe SSDs.

```
sudo pvcreate /dev/nvme*n1
sudo vgcreate lvm_raid /dev/nvme*n1
sudo lvcreate --extents 100%FREE --name lvm_raid lvm_raid -i16 -I64k

sudo mkfs.ext4 /dev/lvm_raid/lvm_raid
sudo mount /dev/lvm_raid/lvm_raid /mnt
```

3.1.1 Drop pagecache in the background

When pagecache is full, OS needs to reclaim pagecache pages to read the data from the storage. The training workload is not cache friendly since the dataset size is at least five times bigger than the host memory size. In this case, the page reclamation overhead can significantly slow down the read performance. To avoid page reclamation overhead in the critical path of read I/O, we ran a background process to drop pagecache periodically. This optimization can significantly improve the number of supported accelerators.

```
#!/usr/bin/env bash

PID=$$
PROC_LIMIT=15

rm /tmp/*drop_cache.pid 2&>/dev/null

drop () {
  echo $1 | sudo tee /proc/sys/vm/drop_caches > /dev/null &
  CHILD_PID=$!
  touch /tmp/$CHILD_PID.drop_cache.pid
  echo "$(date) START $CHILD_PID ($1, $(ls /tmp/*drop_cache.pid | wc -l) processes)"
  wait
  rm /tmp/$CHILD_PID.drop_cache.pid
  echo "$(date) FIN $CHILD_PID ($(ls /tmp/*drop_cache.pid 2>/dev/null | wc -l) processes)"
}

while true; do
  REMAINED=$(ls /tmp/*drop_cache.pid 2>/dev/null | wc -l)
  if [[ $REMAINED -le $PROC_LIMIT ]];
  then
    drop 1 &
    drop 2 &
  fi
  sleep 7
done
```

3.2 Server

The server exposes NVMe devices using SPDK software. Run four SPDK targets, each target exposes 4 NVMe devices on each of four NIC ports.

```
#!/usr/bin/env bash

RPC=/home/sunghwan.kim/repo/mango-workload/test_management/external/rpc/rpc.py

screen -dm sudo pkill -9 -f nvmf_tgt
sleep 1

set -x

echo 128 | sudo tee /proc/sys/vm/nr_hugepages

screen -dm sudo nvmf_tgt -r /var/tmp/spdk0.sock --cpumask '[0,2,3,5,6,8,9,11]' --wait-for-rpc
screen -dm sudo nvmf_tgt -r /var/tmp/spdk1.sock --cpumask '[12,14,15,17,18,20,21,23]' --wait-for-rpc
screen -dm sudo nvmf_tgt -r /var/tmp/spdk2.sock --cpumask '[24,26,27,29,30,32,33,35]' --wait-for-rpc
screen -dm sudo nvmf_tgt -r /var/tmp/spdk3.sock --cpumask '[36,38,39,41,42,44,45,47]' --wait-for-rpc

sleep 5

DEV="1e0f:0013"
init_tgt () {
  TGTID=$1
  SOCK=/var/tmp/spdk$TGTID.sock
  sudo $RPC -s $SOCK sock_set_default_impl --impl posix
  sudo $RPC -s $SOCK iobuf_set_options \
    --small-pool-count 32767 --large-pool-count 16383 --large_bufsize 1050000
  sudo $RPC -s $SOCK framework_start_init
  sudo $RPC -s $SOCK nvmf_create_transport \
    --trtype TCP --max-queue-depth 128 --max-io-qpairs-per-ctrlr 127 \
    --in-capsule-data-size 4096 --max-io-size 1048576 --io-unit-size 1048576 \
    --max-aq-depth 128 --num-shared-buffers 8192 --buf-cache-size 32 \
    --dif-insert-or-strip --sock-priority 0 --abort-timeout-sec 1

  sudo $RPC -s $SOCK nvmf_create_subsystem --allow-any-host \
    --serial-number SPDK$TGTID nqn.2019-07.io.spdk:cnode$TGTID \
    --max-namespaces 16
  IP=$(ip a | grep 100.100 | tr "/" "␣" | awk '{print␣$2}' | sort | tail -n +$((TGTID + 1)) | head -n 1)
  sudo $RPC -s $SOCK nvmf_subsystem_add_listener \
    --trtype TCP --adrfam IPv4 --traddr $IP \
    --trsvcid 4420 nqn.2019-07.io.spdk:cnode$TGTID

  NVME_ID=0
  for PCI in $(lspci -d $DEV -D | awk '{print␣$1}' | tail -n +$((1 + TGTID * 4)) | head -n 4)
  do
    sudo $RPC -s $SOCK bdev_nvme_attach_controller \
        --name Nvme$NVME_ID --trtype PCIe --traddr $PCI
    sudo $RPC -s $SOCK nvmf_subsystem_add_ns \
        --nsid $((NVME_ID + 1)) nqn.2019-07.io.spdk:cnode$TGTID Nvme${NVME_ID}n1
    NVME_ID=$((NVME_ID + 1))
  done
}

init_tgt 0
init_tgt 1
init_tgt 2
init_tgt 3
```