实验指导手册： NVMe over Fabrics实验初探

Zhengdong Wang

本次实验主要参考了https://spdk.io/doc/nvmf.html

实验环境

- 1号机作为NVMe-oF Target, 配置同 使用推荐环境进行实验 (By WZD)
- 2号机作为NVMe-oF Host, 配置如下:
  - OS: Manjaro 21.1.0
  - Kernel: Linux 5.10.59-1-MANJARO
- 1号机与2号机可以通过网络互相访问 (有条件的话可以直接用网线连接两台机器)

  本次实验中1号机IP为192.168.1.244, 2号机IP为192.168.1.13

在1号机上配置NVMe-oF Target

由于没有RDMA网卡, 本次实验中NVMe-oF使用TCP传输。SPDK默认配置已经包含了对TCP传输的支持。

1. 解绑NVMe盘并分配大页内存。由于本次实验要创建内存块设备, 所以设置了较多的大页内存。

Bash

```
$ sudo HUGEMEM=16384 scripts/setup.sh
```

2. 启动SPDK NVMe-oF Target, 分配0, 1号CPU, 对应的cpumask为0x3

Bash

```
$ sudo build/bin/nvmf_tgt -m 0x3
```

3. 初始化TCP传输

```bash
$ sudo scripts/rpc.py nvmf_create_transport -t TCP -u 16384 -m 8 -c 8192
```

4. 创建块设备

   SPDK NVMe-oF Target支持多种块设备, 本次实验以NVMe块设备, 内存块设备和uring块设备为例。

   创建NVMe块设备

```bash
$ sudo scripts/rpc.py bdev_nvme_attach_controller -b NVMe0 -t PCIe -a 0000:01:00.0
```

   创建内存块设备, 大小为8G, 块大小512K

```bash
$ sudo scripts/rpc.py bdev_malloc_create -b Malloc0 8192 512
```

   创建uring块设备。

```bash
$ sudo scripts/rpc.py bdev_uring_create <path to disk file> Uring0 512
```

5. 创建NVMe-oF subsystem

```bash
$ sudo scripts/rpc.py nvmf_create_subsystem nqn.2016-06.io.spdk:cnode1 -a -s SPDK00000000000001 -d SPDK_Controller1
```

其中nqn.2016-06.io.spdk:cnode1为NVMe Qualified Names (NQN), 用于NVMe-oF Host识别相应的subsystem

6. 将块设备挂载到subsystem上

Bash

```
$ sudo scripts/rpc.py nvmf_subsystem_add_ns nqn.2016-06.io.spdk:cnode1 NVMe0n1
$ sudo scripts/rpc.py nvmf_subsystem_add_ns nqn.2016-06.io.spdk:cnode1 Malloc0
$ sudo scripts/rpc.py nvmf_subsystem_add_ns nqn.2016-06.io.spdk:cnode1 Uring0
```

7. 设置subsystem的监听地址和端口, 192.168.1.244为1号机的IP地址

Bash

```
$ sudo scripts/rpc.py nvmf_subsystem_add_listener nqn.2016-06.io.spdk:cnode1 -t tcp -a 192.168.1.244 -s 12345
```

8. 以上工作都完成后检查subsystem状态

Bash

```
$ sudo scripts/rpc.py nvmf_get_subsystems
```

```
~/spdk master ❯ sudo scripts/rpc.py nvmf_get_subsystems
[
  {
    "nqn": "nqn.2016-06.io.spdk:cnode1",
    "subtype": "NVMe",
    "listen_addresses": [
      {
        "transport": "TCP",
        "trtype": "TCP",
        "adrfam": "IPv4",
        "traddr": "192.168.1.244",
        "trsvcid": "12345"
      }
    ],
    "allow_any_host": true,
    "hosts": [],
    "serial_number": "SPDK00000000000001",
    "model_number": "SPDK_Controller1",
    "max_namespaces": 32,
    "min_cntlid": 1,
    "max_cntlid": 65519,
    "namespaces": [
      {
        "nsid": 1,
        "bdev_name": "NVMe0n1",
        "name": "NVMe0n1",
        "nguid": "929989EC1594495D8E8A4177CAE181BF",
        "uuid": "929989ec-1594-495d-8e8a-4177cae181bf"
      },
      {
        "nsid": 2,
        "bdev_name": "Malloc0",
        "name": "Malloc0",
        "nguid": "391306AE45B449DCBE4E12F0DE53D325",
        "uuid": "391306ae-45b4-49dc-be4e-12f0de53d325"
      },
      {
        "nsid": 3,
        "bdev_name": "Uring0",
        "name": "Uring0",
        "nguid": "2D2F736B80AD4351AB9BB92FEA9AB70C",
        "uuid": "2d2f736b-80ad-4351-ab9b-b92fea9ab70c"
      }
    ]
  }
]
```

在2号机上配置NVMe-oF Host

Linux内核和SPDK都提供了NVMe-oF Host的支持, 可以分别尝试两种方案并用fio测试性能

使用内核提供的NVMe-oF Host

1. 加载nvme-tcp模块

Bash

```
$ sudo modprobe nvme-tcp
```

2. 使用nvme-cli工具配置NVMe-oF Host

   发现指定地址的NVMe-oF Target

Bash

```
$ sudo nvme discover -t tcp -a 192.168.1.244 -s 12345
```

```
shizuku@shizuku-manjaro ~/spdk master ❯ sudo nvme discover -t tcp -a 192.168.1.244 -s 12345

Discovery Log Number of Records 1, Generation counter 10
=====Discovery Log Entry 0======
trtype:  tcp
adrfam:  ipv4
subtype: nvme subsystem
treq:    not required
portid:  0
trsvcid: 12345
subnqn:  nqn.2016-06.io.spdk:cnode1
traddr:  192.168.1.244
sectype: none
```

   连接target

Bash

```
$ sudo nvme connect -t tcp -n "nqn.2016-06.io.spdk:cnode1" -a
192.168.1.244 -s 12345
```

3. 连接到target之后可以查看到NVMe-oF设备, 即图中的nvme1

```
shizuku@shizuku-manjaro ~/spdk master ❯ lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda           8:0    0 238.5G  0 disk
├─sda1        8:1    0    64M  0 part /boot/efi
├─sda2        8:2    0    16M  0 part
├─sda3        8:3    0   512M  0 part
├─sda4        8:4    0 237.3G  0 part
└─sda5        8:5    0   621M  0 part
nvme0n1     259:0    0 476.9G  0 disk
├─nvme0n1p1 259:1    0    16M  0 part
├─nvme0n1p2 259:2    0   184G  0 part
├─nvme0n1p3 259:3    0   500M  0 part /boot
└─nvme0n1p4 259:4    0 292.5G  0 part /var/lib/docker/btrfs
                                      /home
                                      /
nvme1n1     259:8    0 465.8G  0 disk
nvme1n2     259:11   0     8G  0 disk
nvme1n3     259:13   0     8G  0 disk
```

nvme1有3个namespace, 分别对应我们创建的NVMe、内存、uring块设备

4. 把NVMe-oF设备当作一般的NVMe盘使用即可, 如挂载到虚拟机等等

   由于使用了TCP传输, NVMe-oF的性能受NIC性能、网络带宽、路由器性能等因素影响很大, 本实验中对3种块设备的写入速率基本相同, 均为18 MB/s 左右

```
shizuku@shizuku-manjaro ~/spdk master 3m 54s ❯ sudo dd if=/dev/zero of=/dev/nvme1n1 bs=512k count=1000
 oflag=nonblock
1000+0 records in
1000+0 records out
524288000 bytes (524 MB, 500 MiB) copied, 29.239 s, 17.9 MB/s
```

   使用完毕后用nvme-cli断开连接

Bash

```
$ sudo nvme disconnect -n "nqn.2016-06.io.spdk:cnode1"
```

使用SPDK提供的NVMe-oF Host

SPDK可以通过NVMe-oF创建NVMe块设备, 以 使用推荐环境进行实验 (By WZD) 中启动Vhost-BLK的过程为例

在创建NVMe块设备时, 可以指定NVMe-oF Target的IP地址、端口和NQN

Bash

```
$ sudo scripts/rpc.py bdev_nvme_attach_controller -b NVMe0 -t tcp -a
192.168.1.244 -f IPv4 -s 12345 -n nqn.2016-06.io.spdk:cnode1
```

```
shizuku@shizuku-manjaro ~/spdk master ❯ sudo scripts/rpc.py bdev_nvme_attach_controller -b NVMe0 -t
 tcp -a 192.168.1.244 -f IPv4 -s 12345 -n nqn.2016-06.io.spdk:cnode1
[sudo] password for shizuku:
NVMe0n1 NVMe0n2 NVMe0n3
```

这样就可以分别用NVMe0n1, NVMe0n2, NVMe0n3创建三个vhost-blk设备挂载到虚拟机上

其他可能的实验环境(未测试)

进行NVMe-oF实验的基本条件是两台机器+网络, 两台机器可以是物理机也可以是虚拟机, 只要能通过网络连接即可, 如果仅为验证实验步骤的话只有一台机器也可(本机同时作为target和host)