实验指导手册：物理机

Zhengdong Wang

实验环境

- CPU: AMD Ryzen 5 5600X
- RAM: 32GB DDR4 4000MHz
- SSD: Samsung PM981a NVMe M.2 1TB (For Host and VM)
- SSD: Samsung 980 PRO NVMe M.2 500GB (For SPDK, path=/dev/nvme0n1)
- OS: Ubuntu 20.04
- Kernel: Linux 5.11.0-25-generic
- QEMU: 5.2.0

配置实验环境

1. 在BIOS中启用硬件虚拟化(Intel VT-x/AMD SVM)和IOMMU
2. 安装qemu, nvme-cli, smartmontools
3. 配置NVMe盘

确认NVMe盘多队列功能已经启用, 如出现none和mq-deadline则为启用

Bash

```
$ cat /sys/block/nvme0n1/queue/scheduler
[none] mq-deadline
```

关闭写缓存 (Samsung only)

Bash

```
$ sudo nvme set-feature -f 6 -v 0 /dev/nvme0n1
set-feature:06 (Volatile Write Cache), value:00000000
```

将盘覆写两遍

```Bash
$ sudo dd if=/dev/zero of=/dev/nvme0n1 bs=512k oflag=nonblock
```

确保写入量>=2 * 容量，写入量可用smartctl查看

```Bash
$ sudo smartctl /dev/nvme0n1 -a
```

4. 安装fio

```Bash
$ git clone https://github.com/axboe/fio
$ cd fio && git checkout fio-3.27
$ make
$ sudo make install
```

5. 编译SPDK并运行单元测试

参考https://spdk.io/doc/getting_started.html

配置构建选项时注意加入fio和io_uring(需要先安装liburing-dev)支持

```Bash
$ ./configure --with-fio=<path to fio repo> --with-uring
```

6. 创建一个虚拟机并安装OS

测试项目

- 4K随机读取, 队列深度1, 线程数1 (4K-RR-QD1J1)
- 4K随机写入, 队列深度1, 线程数1 (4K-RW-QD1J1)
- 4K随机读取, 队列深度32, 线程数1 (4K-RR-QD32J1)
- 4K随机写入, 队列深度32, 线程数1 (4K-RW-QD32J1)
- 4K随机读写(7:3), 队列深度32, 线程数1 (4K-MIX-QD32J1)

参考配置文件见附录

在宿主机上对NVMe盘性能进行测试

使用内核NVMe驱动

使用默认配置文件运行fio即可

```Bash
$ sudo fio bench.conf
```

使用SPDK用户态NVMe驱动

1. 修改bench.conf, 将ioengine换为spdk_nvme, 保存为bench-spdk-nvme.conf (见附录)
2. 使用spdk中的scripts/setup.sh将NVMe盘从内核中解绑
3. 使用bench-spdk-nvme.conf运行fio

```Bash
$ sudo fio bench-spdk-nvme.conf
```

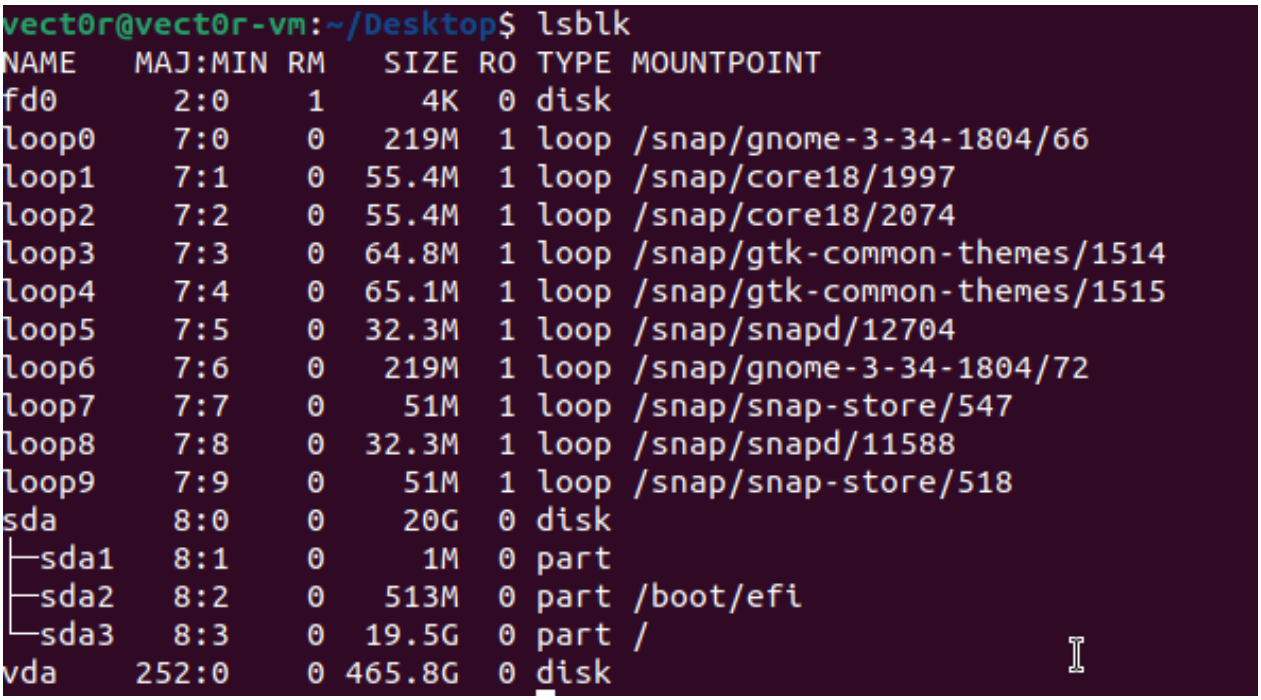
在虚拟机上对NVMe盘性能进行测试

将NVMe盘作为块设备挂载到虚拟机上

1. 用命令行启动QEMU时以virtio模式挂载指定块设备

```Bash
$ sudo qemu-system-x86_64 \
  --enable-kvm \
  -cpu host -smp 6 \
  -m 16G \
  -drive file=/var/lib/libvirt/images/ubuntu20.10.qcow2,if=none,id=disk \
  -device ide-hd,drive=disk,bootindex=0 \
  -drive file=/dev/nvme0n1,if=none,id=bdev \
  -device virtio-blk,drive=bdev,id=virtio0
```

2. 在虚拟机中可以找到挂载的块设备, 这里为vda



3. 将bench.conf导入虚拟机, filename修改为/dev/vda, 用fio运行即可.

将NVMe盘直通给虚拟机

1. 在宿主机中使用spdk中的scripts/setup.sh将NVMe盘从内核中解绑, 不需要设置大页内存

Bash

```
$ sudo HUGEMEM=0 scripts/setup.sh
```

2. 用命令行启动QEMU时以vfio-pci模式直通NVMe盘对应的PCI设备

Bash

```
$ sudo qemu-system-x86_64 \
  --enable-kvm \
  -cpu host -smp 6 \
  -m 16G \
  -drive
file=/var/lib/libvirt/images/ubuntu20.10.qcow2,if=none,id=disk \
  -device ide-hd,drive=disk,bootindex=0 \
  -device vfio-pci,host=0000:01:00.0
```

其中0000:01:00.0为NVMe盘的PCI地址, 可用spdk中的setup.sh查看

Bash

```
$ scripts/setup.sh status
```

3. 在虚拟机中可以找到直通的NVMe盘, 即这里的nvme0n1

```
vect0r@vect0r-vm:~$ lsblk
NAME    MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
fd0       2:0    1     4K  0 disk
loop0     7:0    0  55.4M  1 loop /snap/core18/2074
loop1     7:1    0  55.4M  1 loop /snap/core18/2128
loop2     7:2    0  64.8M  1 loop /snap/gtk-common-themes/1514
loop3     7:3    0    51M  1 loop /snap/snap-store/518
loop4     7:4    0  65.1M  1 loop /snap/gtk-common-themes/1515
loop5     7:5    0   219M  1 loop /snap/gnome-3-34-1804/66
loop6     7:6    0    51M  1 loop /snap/snap-store/547
loop7     7:7    0  32.3M  1 loop /snap/snapd/11588
loop8     7:8    0   219M  1 loop /snap/gnome-3-34-1804/72
loop9     7:9    0  32.3M  1 loop /snap/snapd/12704
sda       8:0    0    20G  0 disk
├─sda1    8:1    0     1M  0 part
├─sda2    8:2    0   513M  0 part /boot/efi
└─sda3    8:3    0  19.5G  0 part /
nvme0n1 259:0    0 465.8G  0 disk
```

4. 将bench.conf导入虚拟机, filename修改为/dev/nvme0n1, 用fio运行即可.

使用QEMU的NVMe用户态驱动

1. 在宿主机中使用spdk中的scripts/setup.sh将NVMe盘从内核中解绑, 不需要设置大页内存

Bash

```
$ sudo HUGEMEM=0 scripts/setup.sh
```

2. 用命令行启动QEMU

Bash

```
$ sudo qemu-system-x86_64 \
  --enable-kvm \
  -cpu host -smp 6 \
  -m 16G \
  -drive
file=/var/lib/libvirt/images/ubuntu20.10.qcow2,if=none,id=disk \
  -device ide-hd,drive=disk,bootindex=0 \
  -drive file=nvme://0000:01:00.0/1,if=none,id=drive0 \
  -device virtio-blk,drive=drive0,id=virtio0
```

其中0000:01:00.0/1为NVMe盘的PCI地址和namespace, 满足格式 <domain:bus:dev.func>/<namespace>, 前一部分可用spdk中的setup.sh查看(见上文), 后面的namespace默认为1

3. 在虚拟机中可以找到NVMe设备, 这里为vda

```
vect0r@vect0r-vm:~$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
fd0      2:0    1     4K  0 disk
loop0    7:0    0  55.4M  1 loop /snap/core18/1997
loop1    7:1    0  55.4M  1 loop /snap/core18/2074
loop2    7:2    0   219M  1 loop /snap/gnome-3-34-1804/66
loop3    7:3    0   219M  1 loop /snap/gnome-3-34-1804/72
loop4    7:4    0  64.8M  1 loop /snap/gtk-common-themes/1514
loop5    7:5    0  65.1M  1 loop /snap/gtk-common-themes/1515
loop6    7:6    0    51M  1 loop /snap/snap-store/518
loop7    7:7    0  32.3M  1 loop /snap/snapd/11588
loop8    7:8    0    51M  1 loop /snap/snap-store/547
loop9    7:9    0  32.3M  1 loop /snap/snapd/12704
sda      8:0    0    20G  0 disk
├─sda1   8:1    0     1M  0 part
├─sda2   8:2    0   513M  0 part /boot/efi
└─sda3   8:3    0  19.5G  0 part /
vda    252:0    0 465.8G  0 disk
vect0r@vect0r-vm:~$
```

4. 将bench.conf导入虚拟机, filename修改为/dev/vda, 用fio运行即可.

使用SPDK Vhost-BLK

1. 在宿主机中使用SPDK中的scripts/setup.sh将NVMe盘从内核中解绑, 同时设置大页内存总量(不宜设得太小), 本次实验设为16GB

Bash

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

可以使用scripts/setup.sh查看大页内存是否设置成功

```
~/spdk master ❯ sudo scripts/setup.sh status
Hugepages
node     hugesize     free /  total
node0   1048576kB        0 /      0
node0      2048kB     8192 /   8192

Type     BDF             Vendor Device NUMA    Driver           Device     Block devices
NVMe     0000:01:00.0    144d   a80a   0       vfio-pci         -          -
NVMe     0000:04:00.0    144d   a808   0       nvme             nvme1      nvme1n1
~/spdk master ❯
```

2. 在宿主机中启动vhost. 将3号和9号CPU分配给vhost用于IO轮询(可自行选择), 对应的cpumask为0x208. 分配内存数不能超过为SPDK设置的大页内存总数(这里为16G)

Bash

```
$ sudo build/bin/vhost -S /var/tmp -s 8192 -m 0x208
```

```
~/spdk master ❯ sudo build/bin/vhost -S /var/tmp -s 8192 -m 0x208
[2021-08-05 15:51:47.717716] Starting SPDK v21.10-pre git sha1 d1e67b8b1 / DPDK 21.05.0 initialization...
[2021-08-05 15:51:47.717742] [ DPDK EAL parameters: [2021-08-05 15:51:47.717748] vhost [2021-08-05 15:51:47.717752] --no-shconf [2021-08-05 15:51:47.7177
56] -c 0x208 [2021-08-05 15:51:47.717759] -m 8192 [2021-08-05 15:51:47.717763] --log-level=lib.eal:6 [2021-08-05 15:51:47.717766] --log-level=lib.cryptod
ev:5 [2021-08-05 15:51:47.717770] --log-level=user1:6 [2021-08-05 15:51:47.717773] --iova-mode=pa [2021-08-05 15:51:47.717777] --base-virtaddr=0x20000000
0000 [2021-08-05 15:51:47.717783] --match-allocations [2021-08-05 15:51:47.717790] --file-prefix=spdk_pid52719 [2021-08-05 15:51:47.717795] ]
EAL: No available 1048576 kB hugepages reported
TELEMETRY: No legacy callbacks, legacy socket not created
[2021-08-05 15:51:48.432088] app.c: 540:spdk_app_start: *NOTICE*: Total cores available: 2
[2021-08-05 15:51:48.469796] reactor.c: 931:reactor_run: *NOTICE*: Reactor started on core 3
[2021-08-05 15:51:48.469796] reactor.c: 931:reactor_run: *NOTICE*: Reactor started on core 9
[2021-08-05 15:51:48.469844] accel_engine.c: 988:spdk_accel_engine_initialize: *NOTICE*: Accel engine initialized to use software engine.
```

3. 创建NVMe块设备

Bash

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

4. 创建vhost-blk设备, 分配3号和9号CPU(必须包含在之前分配给vhost的CPU中)

Bash

```
$ sudo scripts/rpc.py vhost_create_blk_controller --cpumask 0x208
vhost.0 NVMe0n1
```

5. 用命令行启动QEMU

Bash

```
$ sudo taskset -c 0,1,2,6,7,8 qemu-system-x86_64 \
  --enable-kvm \
  -cpu host -smp 6 \
  -m 8G -object memory-backend-file,id=mem0,size=8G,mem-
path=/dev/hugepages,share=on -numa node,memdev=mem0 \
  -drive
file=/var/lib/libvirt/images/ubuntu20.10.qcow2,if=none,id=disk \
  -device ide-hd,drive=disk,bootindex=0 \
  -chardev socket,id=spdk_vhost_blk0,path=/var/tmp/vhost.0 \
  -device vhost-user-blk-pci,chardev=spdk_vhost_blk0,num-queues=2
```

这里使用taskset为虚拟机分配了0, 1, 2, 6, 7, 8号CPU(注意不能和分配给vhost的CPU重复), 分配了8G内存(分配给虚拟机的内存+分配给vhost的内存<=大页内存总数)

6. 在虚拟机中可以找到vhost-blk设备, 这里为vda

```
vect0r@vect0r-vm:~$ lsblk --output "NAME,SIZE,SUBSYSTEMS"
NAME     SIZE SUBSYSTEMS
fd0        4K block:platform
loop0   55.4M block
loop1   55.4M block
loop2    219M block
loop3    219M block
loop4   64.8M block
loop5   65.1M block
loop6     51M block
loop7   32.3M block
loop8     51M block
loop9   32.3M block
sda       20G block:scsi:pci
├─sda1     1M block:scsi:pci
├─sda2   513M block:scsi:pci
└─sda3  19.5G block:scsi:pci
vda    465.8G block:virtio:pci
vect0r@vect0r-vm:~$
```

7. 将bench.conf导入虚拟机, filename修改为/dev/vda, 用fio运行即可.

附录：fio配置文件 (仅供参考)

各参数的详细说明可参考https://fio.readthedocs.io/en/latest/fio_man.html

bench.conf

Plain Text

```
[global]
name=NVMe-benchmark
filename=/dev/nvme0n1      # NVMe设备路径
ioengine=io_uring          # 使用Linux内核(5.1+)提供的异步IO接口io_uring作为fio的IO引擎，也可使用libaio等
sqthread_poll=1            # 开启io_uring的轮询模式，增强性能
direct=1                   # 使用无缓冲IO
numjobs=1                  # 测试线程数
runtime=10                 # 每个测试项目的运行时间(单位：秒)，为防止SSD过热，此项不宜设置过大
thread
time_based
randrepeat=0
norandommap
refill_buffers
group_reporting

[4K-RR-QD1J1]
```

```
bs=4k
rw=randread
iodepth=1
stonewall

[4K-RW-QD1J1]
bs=4k
rw=randwrite
iodepth=1
stonewall

[4K-RR-QD32J1]
bs=4k
rw=randread
iodepth=32
stonewall

[4K-RW-QD32J1]
bs=4k
rw=randwrite
iodepth=32
stonewall

[4K-MIX-QD32J1]
bs=4k
rw=randrw
rwmixread=70
iodepth=32
stonewall
```

bench-spdk-nvme.conf

Plain Text

```
[global]
name=NVMe-benchmark-spdk-nvme
filename=trtype=PCIe traddr=0000.01.00.0 ns=1    # 将traddr设为NVMe盘的PCI地址，可用lspci -D查看(注意将地址中的:换为.)
ioengine=<path to spdk repo>/build/fio/spdk_nvme # spdk_nvme引擎地址
direct=1
numjobs=1
size=500G                                        # spdk_nvme要求必须设置size项，将该项设为硬盘容量即可
```

```
runtime=10
thread
time_based
randrepeat=0
norandommap
refill_buffers
group_reporting

[4K-RR-QD1J1]
bs=4k
rw=randread
iodepth=1
stonewall

[4K-RW-QD1J1]
bs=4k
rw=randwrite
iodepth=1
stonewall

[4K-RR-QD32J1]
bs=4k
rw=randread
iodepth=32
stonewall

[4K-RW-QD32J1]
bs=4k
rw=randwrite
iodepth=32
stonewall

[4K-MIX-QD32J1]
bs=4k
rw=randrw
rwmixread=70
iodepth=32
stonewall
```