QEMU Backend Block Device Driver In Practice

CLK 2017 | Oct 21, 2017

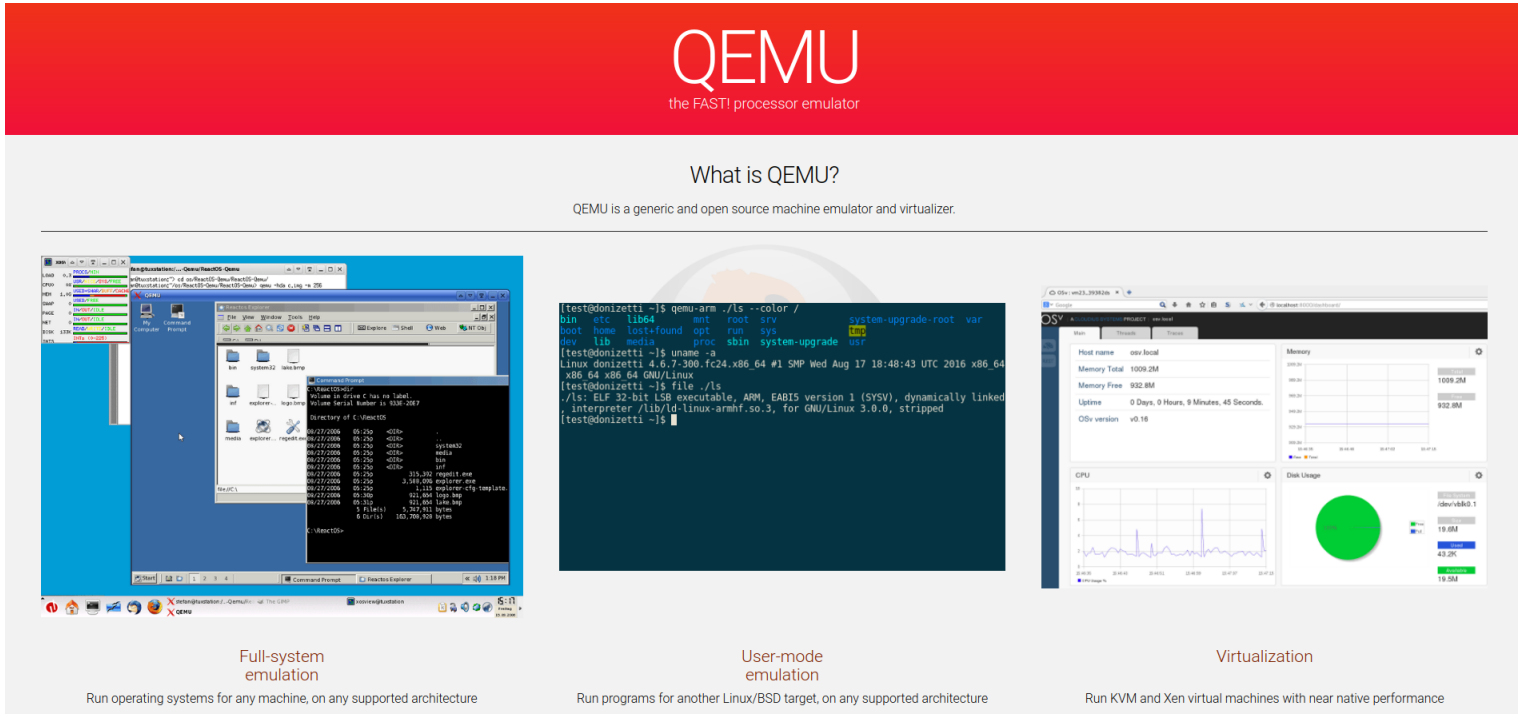
neilsun # yunify.com

Background

- QingCloud NeonSan(ServerSAN)
- RDMA
  - IOPS: 100k+
  - Delay: 90us
- TCP

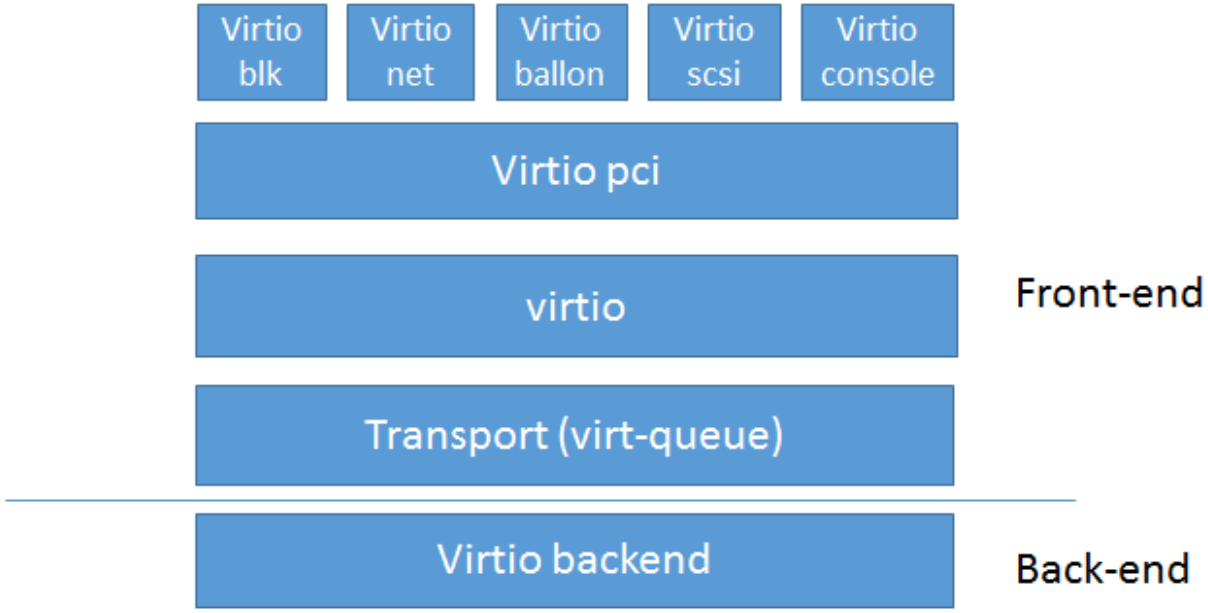
What is QEMU

- QEMU is a generic and open source machine emulator and virtualizer.
- https://www.qemu.org

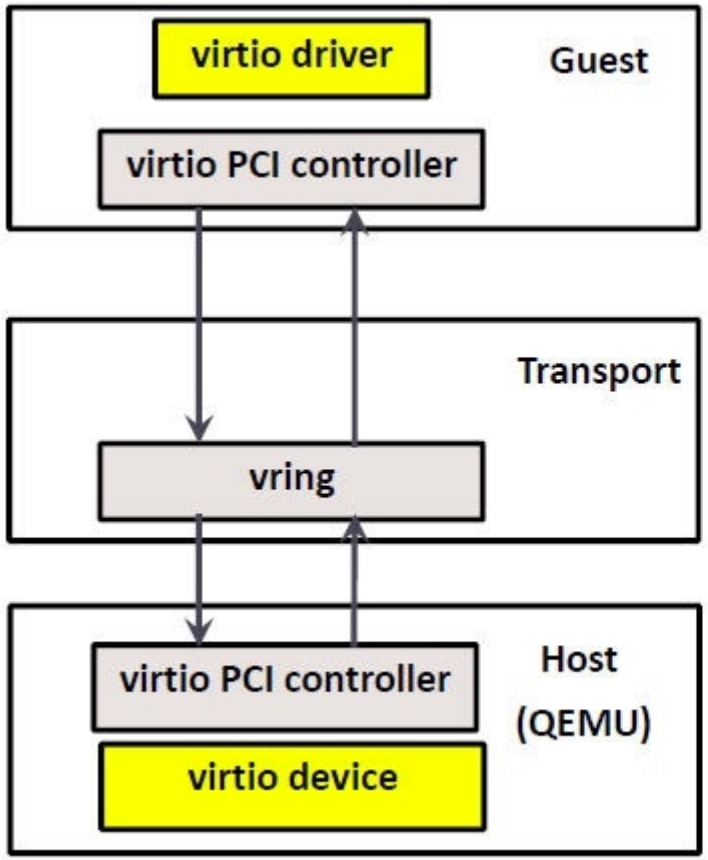


What is VirtIO

- Paravirtualized drivers for kvm/Linux.
- https://www.linux-kvm.org/page/Virtio



How VirtIO Works



Front-end

A kernel module in guest OS.
Accepts I/O requests from user process.
Transfer I/O requests to back-end.

Back-end

A device in QEMU.
Accepts I/O requests from front-end.
Perform I/O operation via physical device.

- **Virtqueues (per device)**
- **Vring (per virtqueue)**
- **Queue requests**

Backing Image Format

- raw
  - Highest possible performance
  - Almost no features (like snapshots etc.)
- qcow2
  - Sparse images
  - Snapshots (internal and external)
  - Encryption
  - Compression
- vmdk,vhd,vdi
  - Best to convert to raw or qcow2 for running VMs

Backing Storage Type

- file
  - Local file system
  - -driver file=vol.img
- block
  - Whole disk or partition
  - Logical volume
  - External implementation of iscsi, NBD
  - -drive file=/dev/sda2
  - -drive file=/dev/nbd0
- network
  - Device connected through network protocol.
  - -drive file=nbd:192.168.1.2:1234
  - -drive file=gluster+tcp://192.168.1.2/vol/vol.img

QEMU Utils

- qemu-img
- qemu-img create -f qcow2 -o size=100G vol.qcow2
- qemu-img info vol.qcow2
- qemu-img convert -f qcow2 vol.qcow2 -O raw vol.raw [-p]
- qemu-img resize +50G vol.qcow2
- qemu-img snapshot -c ss1 vol.qcow2
- qemu-img create -b vol.qcow2 -f qcow2 vol_in.qcow2

QEMU Utils

- qemu-nbd
- modprobe nbd nbds_max=100 max_part=8
- qemu-nbd -c /dev/nbd0 vol.img [-f qcow2] [-o offset]
- qemu-nbd -d /dev/nbd0

QEMU Utils

- qemu-io
- qemu-io -c help # ‘help read’
- qemu-io -c “open vol.qcow2” -f qcow2
- qemu-io -c “write -**pP** 0xa 0 4096” vol.raw
- qemu-io -c "read -**pP** 0xa 0 4096" -f raw vol.raw # “ -v” for dump
- qemu-io -c "aio_write -P 0xa 0 512" -f raw vol.raw

~~p: use blk_pwrite to write the file~~
P: use different pattern to fill file

~~p: use blk_pread to write the file~~
P: use a pattern to verify read data

QEMU Utils

- qemu-system-*

```
qemu-system-x86_64 --enable-kvm --name ubuntu -nographic \
-smp 2 -m 2048 -boot strict=on \
-drive format=raw,file=/path/to/vol.raw,if=none,id=drive-virtio-disk0 \
-device virtio-blk-pci,scsi=off,bus=pci.0,addr=0x4,drive=drive-virtio-disk0,id=virtio-disk0,bootindex=1 \
-monitor tcp:0.0.0.0:5555,server,nowait \
-vnc :20
```

- https://qemu.weilnetz.de/doc/qemu-doc.html

Backend Block Device Structures

- BlockDriver
- BlockDriverState

```
struct BlockDriverState {
    /* Protected by big QEMU lock or read-only after opening.  No special
     * locking needed during I/O...
     */
    int open_flags; /* flags used to open the file, re-used for re-open */
    bool read_only; /* if true, the media is read only */
    bool encrypted; /* if true, the media is encrypted */
    bool sg;        /* if true, the device is a /dev/sg* */
    bool probed;    /* if true, format was probed rather than specified */
    bool force_share; /* if true, always allow all shared permissions */
    bool implicit;  /* if true, this filter node was automatically inserted */

    BlockDriver *drv; /* NULL means no media */
    void *opaque;

    AioContext *aio_context; /* event loop used for fd handlers, timers, etc */
    /* long-running tasks intended to always use the same AioContext as this
     * BDS may register themselves in this list to be notified of changes
     * regarding this BDS's context */
    QLIST_HEAD(, BdrvAioNotifier) aio_notifiers;
    bool walking_aio_notifiers; /* to make removal during iteration safe */

    char filename[PATH_MAX];
    char backing_file[PATH_MAX]; /* if non zero, the image is a diff of
                                    this file image */
    char backing_format[16]; /* if non-zero and backing_file exists */

    QDict *full_open_options;
    char exact_filename[PATH_MAX];

    BdrvChild *backing;
    BdrvChild *file;

    /* I/O Limits */
    BlockLimits bl;

    /* Flags honored during pwrite (so far: BDRV_REQ_FUA) */
```

```
struct BlockDriver {
    const char *format_name;
    int instance_size;

    /* set to true if the BlockDriver is a block filter */
    bool is_filter;
    /* for snapshots block filter like Quorum can implement the
     * following recursive callback.
     * It's purpose is to recurse on the filter children while calling
     * bdrv_recurse_is_first_non_filter on them.
     * For a sample implementation look in the future Quorum block filter.
     */
    bool (*bdrv_recurse_is_first_non_filter)(BlockDriverState *bs,
                                             BlockDriverState *candidate);

    int (*bdrv_probe)(const uint8_t *buf, int buf_size, const char *filename);
    int (*bdrv_probe_device)(const char *filename);

    /* Any driver implementing this callback is expected to be able to handle
     * NULL file names in its .bdrv_open() implementation */
    void (*bdrv_parse_filename)(const char *filename, QDict *options, Error **errp);
    /* Drivers not implementing bdrv_parse_filename nor bdrv_open should have
     * this field set to true, except ones that are defined only by their
     * child's bs.
     * An example of the last type will be the quorum block driver.
     */
    bool bdrv_needs_filename;

    /* Set if a driver can support backing files */
    bool supports_backing;

    /* For handling image reopen for split or non-split files */
    int (*bdrv_reopen_prepare)(BDRVReopenState *reopen_state,
                               BlockReopenQueue *queue, Error **errp);
    void (*bdrv_reopen_commit)(BDRVReopenState *reopen_state);
    void (*bdrv_reopen_abort)(BDRVReopenState *reopen_state);
    void (*bdrv_join_options)(QDict *options, QDict *old_options);

    int (*bdrv_open)(BlockDriverState *bs, QDict *options, int flags,
                     Error **errp);
    int (*bdrv_file_open)(BlockDriverState *bs, QDict *options, int flags,
                          Error **errp);
    void (*bdrv_close)(BlockDriverState *bs);
    int (*bdrv_create)(const char *filename, QemuOpts *opts, Error **errp);
    int (*bdrv_make_empty)(BlockDriverState *bs);

    void (*bdrv_refresh_filename)(BlockDriverState *bs, QDict *options);

    /* aio */
    BlockAIOCB *(*bdrv_aio_readv)(BlockDriverState *bs,
        int64_t sector_num, QEMUIOVector *qiov, int nb_sectors,
        BlockCompletionFunc *cb, void *opaque);
```

Basic information

- .format_name
- .protocol_name
- .instance_size
- .bdrv_needs_filename
- .bdrv_parse_filename

```
static BlockDriver bdrv_qbd = {
    .format_name            = "qbd",
    .protocol_name          = "qbd",
    .instance_size          = sizeof(BDRVQBDState),
    .bdrv_needs_filename    = true,
    .bdrv_parse_filename    = qemu_qbd_parse_filename,

    .create_opts            = &qemu_qbd_create_opts,
    .bdrv_create            = qemu_qbd_create,

    .bdrv_file_open         = qemu_qbd_open,
    .bdrv_close             = qemu_qbd_close,
    .bdrv_getlength         = qemu_qbd_getlength,

    .bdrv_aio_readv         = qemu_qbd_aio_readv,
    .bdrv_aio_writev        = qemu_qbd_aio_writev,
    .bdrv_probe_blocksizes  = qemu_qbd_probe_blocksizes,
};
```

Block Size

- Physical Block Size
- Logical Block Size
- .bdrv_probe_blocksizes

```
static BlockDriver bdrv_qbd = {
    .format_name            = "qbd",
    .protocol_name          = "qbd",
    .instance_size          = sizeof(BDRVQBDState),
    .bdrv_needs_filename    = true,
    .bdrv_parse_filename    = qemu_qbd_parse_filename,

    .create_opts            = &qemu_qbd_create_opts,
    .bdrv_create            = qemu_qbd_create,

    .bdrv_file_open         = qemu_qbd_open,
    .bdrv_close             = qemu_qbd_close,
    .bdrv_getlength         = qemu_qbd_getlength,

    .bdrv_aio_readv         = qemu_qbd_aio_readv,
    .bdrv_aio_writev        = qemu_qbd_aio_writev,
    .bdrv_probe_blocksizes  = qemu_qbd_probe_blocksizes,
};
```

QEMU

```
-drive file=vol.qcow2,format=qcow2,if=none,id=drive-virtio-disk1 \
-device virtio-blk-pci,bus=pci.0,drive=drive-virtio-disk1,id=virtio-disk1,logical_block_size=512,physical_block_size=4096
```

Libvirt

```
<disk type='file' device='disk'>
  <driver name='qemu' type='qcow2'/>
  <source file='/path/to/vol.qcow2' />
  <target dev='vdb' bus='virtio' />
  <blockio logical_block_size='512' physical_block_size='4096'/>
</disk>
```

Basic IO Functions

- .create_opts
- .bdrv_create
- .bdrv_get_length
- .bdrv_file_open
- .bdrv_close
- .bdrv_aio_readv
- .bdrv_aio_writev

```
static BlockDriver bdrv_qbd = {
    .format_name            = "qbd",
    .protocol_name          = "qbd",
    .instance_size          = sizeof(BDRVQBDState),
    .bdrv_needs_filename    = true,
    .bdrv_parse_filename    = qemu_qbd_parse_filename,

    .create_opts            = &qemu_qbd_create_opts,
    .bdrv_create            = qemu_qbd_create,

    .bdrv_file_open         = qemu_qbd_open,
    .bdrv_close             = qemu_qbd_close,
    .bdrv_getlength         = qemu_qbd_getlength,

    .bdrv_aio_readv         = qemu_qbd_aio_readv,
    .bdrv_aio_writev        = qemu_qbd_aio_writev,
    .bdrv_probe_blocksizes  = qemu_qbd_probe_blocksizes,
};
```

How to Verify(on Host)

- QEMU IO Tests(tests/qemu-iotests)
- ./check
- ./check 001-100

How to Verify(on Host)

- **blkverify** is used to test that a block driver is operating correctly. It will catch data corruption inside QEMU the first time bad data is read and reporting the disk sector that is corrupted.
- qemu-system-x86_64 -cdrom debian.iso -drive file=blkverify:raw.img:test.qcow2
- docs/devel/blockverify.txt

How to Verify(on Host)

- The **blkdebug** block driver is a rule-based error injection engine. It can be used to exercise error code paths in block drivers including ENOSPC (out of space) and EIO.
- cat blkdebug.conf

  [inject-error]

  event = "read_aio"

  errno = "28"

  The core events are:
  - read_aio - guest data read
  - write_aio - guest data write
  - flush_to_os - write out unwritten block driver state (e.g. cached metadata)
  - flush_to_disk - flush the host block device's disk cache

  qapi/block-core.json:BlkdebugEvent for the full list of events.
- qemu-system-x86_64 -drive if=none,cache=none,file=blkdebug:blkdebug.conf:test.img,id=drive0 \

  -device virtio-blk-pci,drive=drive0,id=virtio-blk-pci0
- docs/devel/blkdebug.txt

How to Verify(on Guest VM)

- ltp fs test
- http://linux-test-project.github.io/
- ./runltp -f [io|dio|fs] -d /path/to/storage -t 24h -o result.log

How to Verify(on Guest VM)

- fio
- fio --name=fio-4k --filename=/dev/vdb --ioengine=libaio \
  --runtime=1h --direct=1 --bs=4k --iodepth=16 --numjobs=1 \
  --rw=randwrite --verify=md5
- https://github.com/axboe/fio

```
[global]
ioengine=libaio
filename=/dev/vdb
runtime=1h
direct=1
bs=4k
iodepth=16

[fio-4k]
numjobs=1
rw=write
```

Performance Test(on Guest VM)

- fio

```
root@os10:~/qbd# fio --name=fio-4k --filename=/dev/vdb --ioengine=libaio --runtime=60 --direct=1 --bs=4k --iodepth=16 --numjobs
=1 --rw=randwrite
fio-4k: (g=0): rw=randwrite, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio, iodepth=16
fio-2.19
Starting 1 process
Jobs: 1 (f=1): [w(1)][100.0%][r=0KiB/s,w=388MiB/s][r=0,w=99.5k IOPS][eta 00m:00s]
fio-4k: (groupid=0, jobs=1): err= 0: pid=1497: Wed Sep 27 12:11:15 2017
  write: IOPS=111k, BW=432MiB/s (453MB/s)(25.4GiB/60001msec)
    slat (usec): min=1, max=955, avg= 3.10, stdev= 1.92
    clat (usec): min=49, max=7938, avg=140.58, stdev=38.97
     lat (usec): min=68, max=7941, avg=143.68, stdev=39.16
    clat percentiles (usec):
     |  1.00th=[   95],  5.00th=[  106], 10.00th=[  112], 20.00th=[  120],
     | 30.00th=[  126], 40.00th=[  131], 50.00th=[  137], 60.00th=[  143],
     | 70.00th=[  149], 80.00th=[  159], 90.00th=[  173], 95.00th=[  189],
     | 99.00th=[  223], 99.50th=[  239], 99.90th=[  302], 99.95th=[  338],
     | 99.99th=[ 1048]
    lat (usec) : 50=0.01%, 100=2.01%, 250=97.65%, 500=0.32%, 750=0.01%
    lat (usec) : 1000=0.01%
    lat (msec) : 2=0.01%, 4=0.01%, 10=0.01%
  cpu          : usr=15.55%, sys=46.40%, ctx=1345786, majf=0, minf=10
  IO depths    : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=100.0%, 32=0.0%, >=64=0.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.1%, 32=0.0%, 64=0.0%, >=64=0.0%
     issued rwt: total=0,6635628,0, short=0,0,0, dropped=0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=16

Run status group 0 (all jobs):
  WRITE: bw=432MiB/s (453MB/s), 432MiB/s-432MiB/s (453MB/s-453MB/s), io=25.4GiB (27.2GB), run=60001-60001msec

Disk stats (read/write):
  vdb: ios=91/6623273, merge=0/0, ticks=16/841860, in_queue=840824, util=100.00%
root@os10:~/qbd#
```

Performance Check(on Guest VM)

- iostat

```
root@os10:~/qbd# iostat -xm 1
Linux [redacted]                    _x86_64_        (8 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          0.30    0.00    0.90    0.34    0.00   98.47

Device:         rrqm/s   wrqm/s     r/s     w/s    rMB/s    wMB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
vda               0.00     0.63    4.27    0.47     0.12     0.03    67.47     0.00    0.29    0.14    1.56   0.12   0.06
vdb               0.00     0.00    0.65 17682.92     0.00    69.07     8.00     2.39    0.14    0.20    0.14   0.01  17.40

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          1.53    0.00    4.84    0.00    0.00   93.63

Device:         rrqm/s   wrqm/s     r/s     w/s    rMB/s    wMB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
vda               0.00     0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00    0.00    0.00   0.00   0.00
vdb               0.00     0.00    0.00 101640.00     0.00   397.03     8.00    13.03    0.13    0.00    0.13   0.01  99.60
```

- dstat

```
root@os10:~/qbd# dstat -rd --disk-util --disk-tps -D vdb 1 5
---io/vdb-- --dsk/vdb-- vdb- --dsk/vdb--
 read  writ| read  writ|util|reads writs
0.19  28.0k| 783B  109M|28.1|   0    28k
   0   111k|   0   435M| 100|   0   111k
   0   115k|   0   448M| 100|   0   114k
   0   116k|   0   451M| 100|   0   115k
   0   108k|   0   420M| 100|   0   107k
   0   113k|   0   440M| 100|   0   112k
root@os10:~/qbd#
```

Performance Check(on Guest VM)

- blktrace
- mount -t debugfs debugfs /sys/kernel/debug
- blktrace -d /dev/vdb -o - | blkparse -i - # or ‘btrace /dev/vdb’ for live
- blktrace -d /dev/vdb [-o vdb -w 10]
- blkparse -i vdb -d vdb.blktrace.bin -O
- btt -i vdb.blktrace.bin [-o vdb]
- https://www.cse.unsw.edu.au/~aaronc/iosched/doc/blktrace.html

Performance Check(on Gues

- blkparse

```
253,16   1        1     0.000000000  3180  Q   R 2147483520 + 8 [systemd-udevd]
253,16   1        2     0.000001260  3180  G   R 2147483520 + 8 [systemd-udevd]
253,16   1        3     0.000002336  3180  P   N [systemd-udevd]
253,16   1        4     0.000003088  3180  U   N [systemd-udevd] 1
253,16   1        5     0.000003260  3180  I  RS 2147483520 + 8 [systemd-udevd]
253,16   1        6     0.000003636  3180  D  RS 2147483520 + 8 [systemd-udevd]
253,16   1        7     0.000504532     0  C  RS 2147483520 + 8 [0]
253,16   1        8     0.000516748  3180  Q   R 2147483632 + 8 [systemd-udevd]
253,16   1        9     0.000517269  3180  G   R 2147483632 + 8 [systemd-udevd]
```

D c s T t p a d S n C

- btt

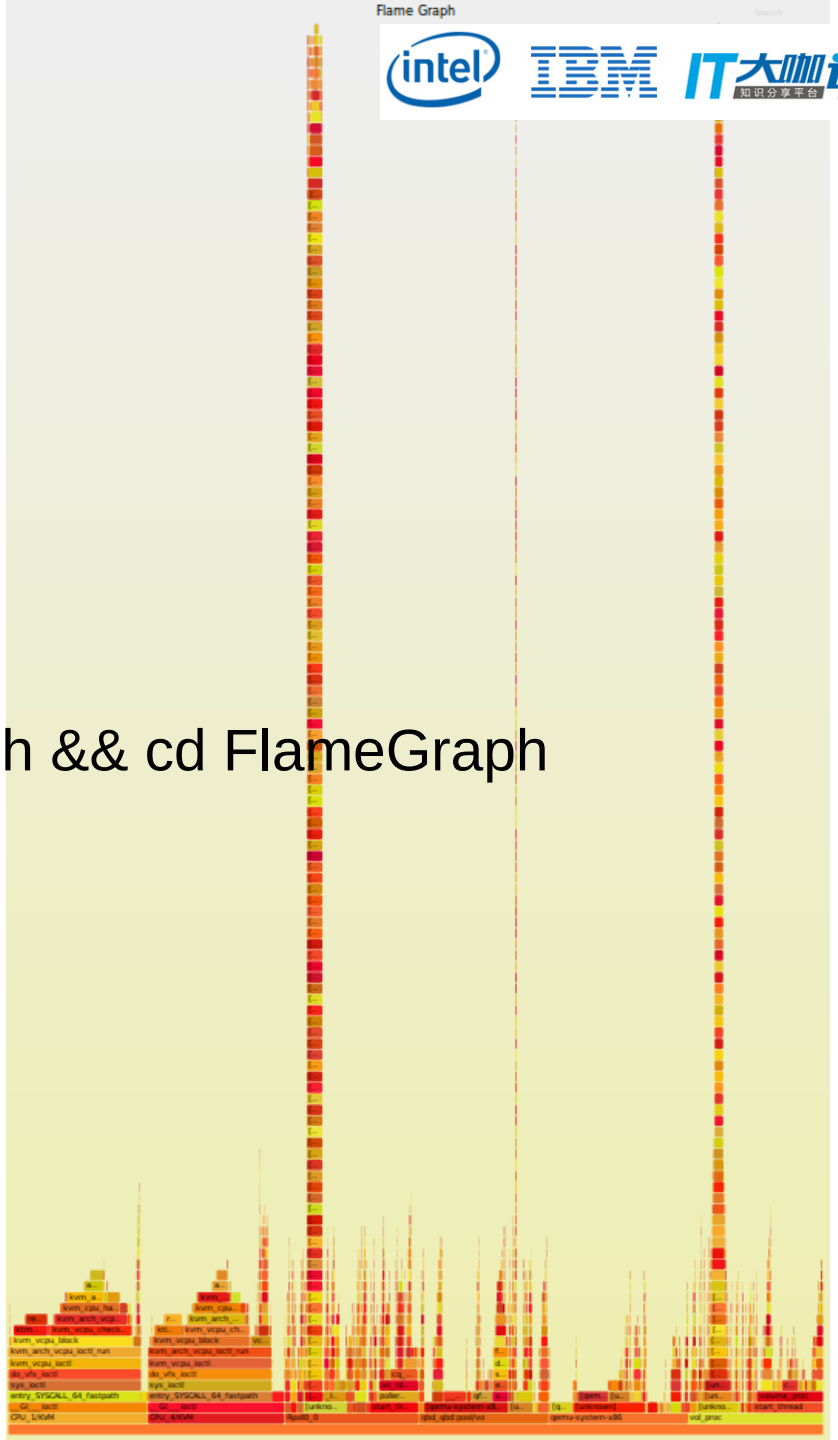
```
==================== All Devices ====================

            ALL           MIN           AVG           MAX           N

Q2Qdm             0.000002931   0.000010542   0.103742442      782231
Q2Cdm             0.000064612   0.000133993   0.008394993      782216

Q2G               0.000000102   0.000000244   0.000038303      782232
G2I               0.000000443   0.000000784   0.000072435      782232
I2D               0.000000098   0.000000200   0.000038383      782232
D2C               0.000063607   0.000132765   0.008394041      782216
```

a2a

D	major and minor numbers of the event's device
c	CPU id
s	Sequence numbers
T	Time stamp (seconds)
t	Time stamp (nanoseconds)
p	Process ID
a	Action A IO was remapped to a different device B IO bounced C IO completion D IO issued to driver F IO front merged with request on queue G Get request I IO inserted onto request queue M IO back merged with request on queue P Plug request Q IO handled by request queue code S Sleep request T Unplug due to timeout U Unplug request X Split
d	RWBS field: R read W write B barrier operations S synchronous operations
S	Sector number
n	Number of blocks
C	Command

Performance(on Host)

- Perf & FrameGraph
- perf record -g -p 50214 -- sleep 10
- perf script > out.perf
- git clone https://github.com/brendangregg/FlameGraph && cd FlameGraph
- ./stackcollapse-perf.pl ../out.perf > out.folder
- ./flamegraph.pl ../out.folder > perf.svg



QEMU Misc

- QMP(QEMU Machine Protocol) is a JSON-based protocol which allows applications to control a QEMU instance.
- HMP(Human Monitor Interface) is the simple interactive monitor on QEMU, designed primarily for debugging and simple human use.
- -qmp tcp:0.0.0.0:4444,server,nowait
- -monitor tcp:0.0.0.0:5555,server,nowait
- nc ip port
- docs/devel/writing-qmp-commands.txt

Libvirt Support

- configure.ac
- + m4/virt-storage-qbd.m4
- src/qemu/qemu_command.c (qemuBuildNetworkDriveStr)
- ...

```
<disk type='network' device='disk'>
  <driver name='qemu' type='raw'/>
  <source protocol='qbd' name='pool/vol_100T' />
  <target dev='vdb' bus='virtio'/>
</disk>
```

Q & A

Thank you.

Join us:
Kernel Team Job & Intern:
kernel_group # yunify.com