io_uring in QEMU: high-performance disk IO for Linux

FOSDEM 2020

Julia Suvorova, Red Hat
Software Engineer
What we’ll discuss today

- io_uring API
- QEMU structure
- Features of io_uring and how they helped QEMU
- Benchmarks
- What left to do
I/O path in VM

VM

userspace

virtio-blk

kernel

vHW

QEMU

userspace

driver

kernel

Host

HW
Existing solutions

Async I/O
- Linux AIO (aio=native)
- Thread pool (aio=threads)

Other
- NVME passthrough (vfio)
- SPDK
I/O path in VM

----------------This part we can improve----------------
io_uring

Yet another kernel ring buffer

- New interface for truly asynchronous communication with kernel: latest versions support network and some other syscalls
- Part of linux 5.1
Main features

- Unlike Linux AIO, separate queues for submission and completion (sqes and cqes)
- Sqes and cqes are shared between userspace and kernel
- Async flush

Submission: QEMU -> kernel -> hw
Completion: QEMU <- kernel <- hw
io_uring interface

Interface

Three new system calls:

```c
io_uring_setup(u32 entries, struct io_uring_params *p)
```
- Can choose different regimes

```c
io_uring_enter(unsigned int fd,
    unsigned int to_submit,
    unsigned int min_complete,
    unsigned int flags,
    sigset_t *sig)
```
- Submit submissions and fetches completions within one syscall
  (Not in Linux AIO!)

```c
io_uring_register(unsigned int fd, unsigned int opcode,
                  void *arg, unsigned int nr_args);
```
- Register fd ahead. No need to do fget() and fput() on each submission and completion respectively
- Register buffers (struct iovec) ahead. Saves get_user_pages() and put_pages()
How fast is it?

Benchmarks on bare metal

Test with fio 3.14:
- aio=libaio
- operation=randread

NVMe SSD Intel
Optane 320G
CPU Intel Xeon Silver
2.20GHz
Integration into QEMU

What’s done:

- Outreachy project idea
- Implemented by Aarushi Mehta
- Basic functionality is merged upstream (will be in QEMU 5.0)

Known issues:

- Problems with file locking in fd registration
- IOPOLL is not implemented
Integration into QEMU

Reuse Linux AIO approach

Qemu event loop is based on AIO context (future improvement: can be switched to io_uring)

Add aio context -> use epoll for completion check

Now we submit requests with io_uring_enter() and check completions on irq

Liburing usage:

Easier to use, less mistakes
Integration into QEMU

How to launch

-drive file=test.img,format=raw,cache=none,aio=io_uring

Works with both IO_DIRECT and cache workload
How fast has it got without extra features?

Test with fio 3.14:
- aio=libaio
- operation=randread

NVMe SSD Intel Optane 320G
CPU Intel Xeon Silver 2.20GHz
Fd registration

Register set of fd on which I/O is operated with io_uring_register()

Saves atomic fget() on submission path

Saves atomic fput() on completion path
Features and benchmarks

Does this help much?

Not really by itself

Test with fio 3.14:
- aio=libaio
- operation=randread

NVMe SSD Intel Optane 320G
CPU Intel Xeon Silver 2.20GHz
Submission polling

Run a kernel thread to wait for submissions, need to wake up with syscall

io_uring_setup() with flag SQ_POLL

Needs fd registration for effective usage

Now we submit requests without syscall and get completions on irq - path without syscalls
Features and benchmarks

Completion polling

Poll completions with busy waiting on `io_uring_enter()`

`io_uring_setup()` with

CPU consuming, but no context switching

In combination with SQ POLL - the fastest way on heavy workloads
Features and benchmarks

Performance

Not implemented yet
Future improvements

Merge SQ_POLL and fd registration
File buffers registration and IO_POLL
Switch to io_uring as default aio (if supported)

Ideas:
Switch main loop to io_uring
Thank you

Questions?

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat