
# NVMe Emulation Performance Optimization

**Jinhao Fan** *<fanjinhao21s@ict.ac.cn>*
Chinese Academy of Sciences

# About me

- GSoC contributor for “NVMe Emulation Performance Optimization”
- Mentors: Klaus Jensen (Samsung), Keith Busch (Meta)
  - With help from Stefan Hajnoczi and others
- Second-year graduate student at Institute of Computing Technology, Chinese Academy of Science (ICT, CAS)
  - Doing research on storage systems
- Eager to learn how real systems works, and how they are built

# QEMU NVMe – Current Status

- Used by developers and researcher to experiment with new features
- Performance surprisingly low
  - Maximum IOPS: **30K**
  - Unable to emulate super fast NVMe devices nowadays
- **Our goal: Make QEMU NVMe's performance comparable to virtio-blk**

| QD | 1 | 4 | 16 | 64 |
|---|---|---|---|---|
| nvme | 31 | 30 | 30 | 30 |
| virtio-blk | 59 | 185 | 260 | 256 |

**Unit:** KIOPS
**Test setup**: FIO 4KB random reads
**Host**: 96-core Xeon Gold 6248R @ 3GHZ, 256GB DRAM, Ubuntu 22.04 with Linux 5.15.0-46

# What did we do?

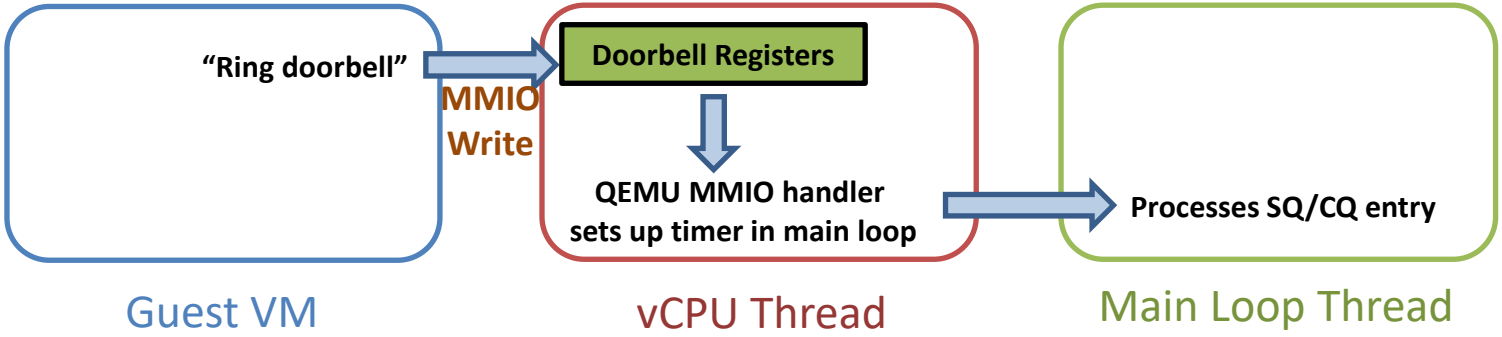
- Reduced MMIO's with **shadow doorbell buffer**
- Lightweight MMIO with **ioeventfd**
- Dedicated emulation thread with **iothread**
- Minimal latency with **polling**
- Thread-safe **eventfd-based** interrupts

# NVMe Primer

- NVMe uses circular lock-free queues for submissions and completions
  - **tail** incremented when producing to the queue
  - **head** incremented when consuming from the queue
- Host informs the device about new entries in the queue by “**ringing the doorbell**”
  - A ”doorbell” is the common name for a **write-only memory-mapped I/O register**

# NVMe Primer

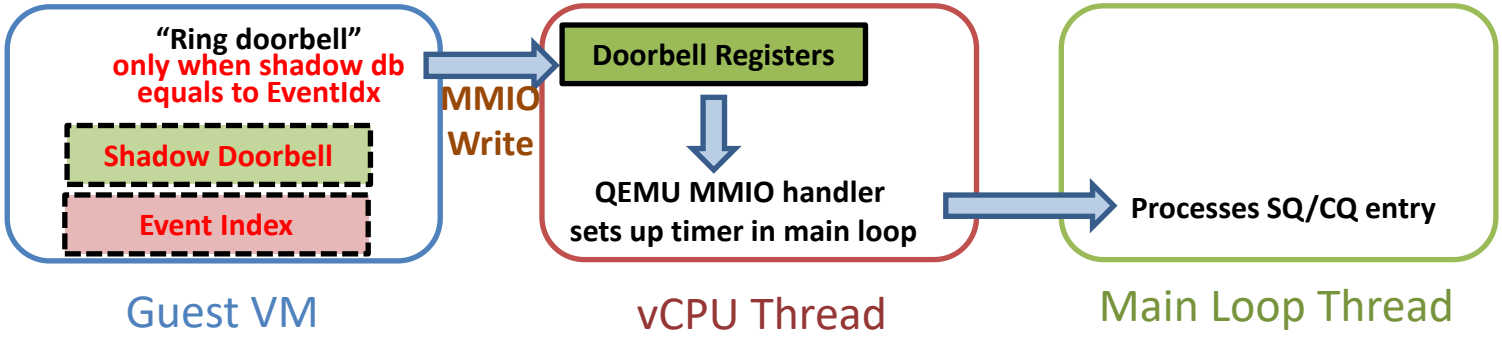
- An NVMe command involves two db writes
  - One for SQ tail doorbell, one for CQ head doorbell
- Applications tend to ring db frequently for latency
- Quite a lot of overhead due to “trap-emulate”



# Shadow Doorbell Buffer

- A para-virtualization feature similar to VIRTIO_F_EVENT_IDX, introduced in NVMe 1.3
- The host registers two buffers that mirror the doorbell registers as **perceived** by the host and controller respectively
  - "**doorbell buffer**", updated by the host
  - "**event index buffer**", updated by the controller
- No MMIO when writing these buffers

# Performance After Shadow DB



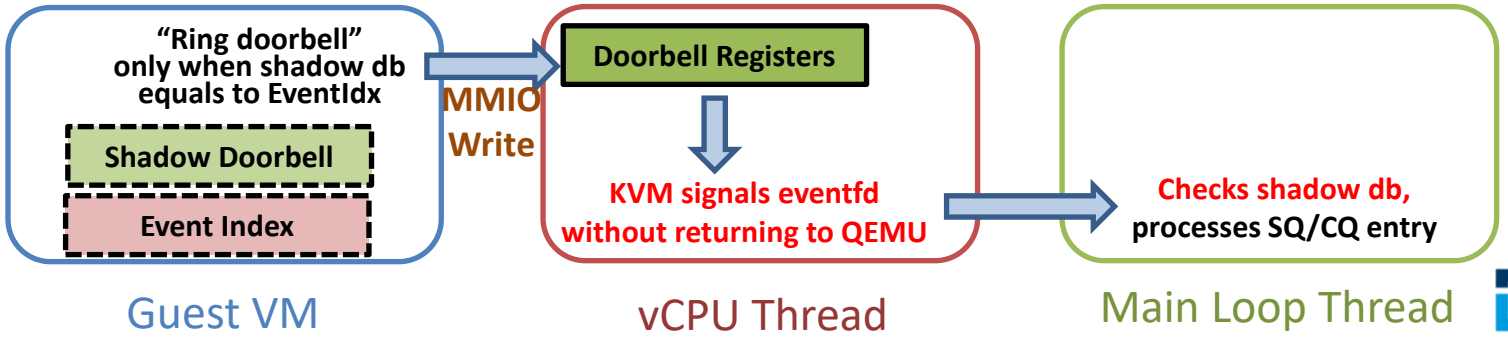

| QD | 1 | 4 | 16 | 64 |
|---|---|---|---|---|
| nvme | 31 | 30 | 30 | 30 |
| +shadow db | 35 | 121 | 176 | 153 |
| virtio-blk | 59 | 185 | 260 | 256 |

# ioeventfd

- Even with shadow doorbell buffers **MMIO is required**
  - Whenever the device is idle (i.e. event index is equal to shadow doorbell value)
- The **ioeventfd** mechanism allows light-weight VMEXITs
  - VMEXIT handler writes to an eventfd and resume running guest code, without going back to QEMU

# ioeventfd

- ioeventfd only tells that an MMIO region is written, but not **what** exact value is written
- **Our solution**: register ioeventfd on **doorbell registers**, and check **shadow doorbell** in the event handler



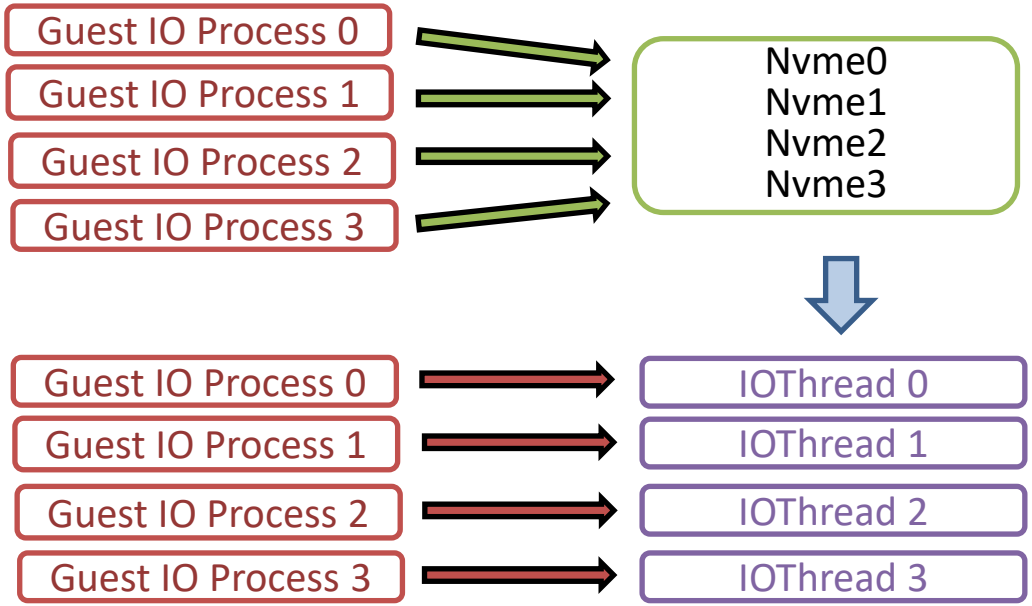
# ioeventfd

- Performance looks good, already close to virtio-blk
- But that is not enough
  - IO emulation in a dedicated thread
  - Polling for low-latency devices

| QD | 1 | 4 | 16 | 64 |
|---|---|---|---|---|
| nvme | 31 | 30 | 30 | 30 |
| +shadow db | 35 | 121 | 176 | 153 |
| +ioeventfd | 41 | 133 | 258 | 313 |
| virtio-blk | 59 | 185 | 260 | 256 |

# IOThread

- Bottleneck of current architecture: the main loop thread




Single Main loop thread:
Overwhelmed :(

Multiple *IOThreads*:
One thread per device! :)

# Thread-safe IRQ Delivery

- QEMU’s default interrupt injection emulation is **not** thread safe.
- This was not a problem when all devices were emulated in the main loop thread
- But challenges arise when emulating in a separate iothread

# Thread-safe IRQ Delivery

- Two **eventfd**-based ways to get around thread safety problems in QEMU interrupt emulation
- When **irqfd is available** (interrupt is MSI-X and KVM supports irqfd)
  - Register a *virtual irq* in KVM and let KVM assert the interrupt when the irqfd is signalled, bypassing QEMU
- When **irqfd is unavailable**
  - Register an event notifier to always (de)assert interrupts in main loop thread

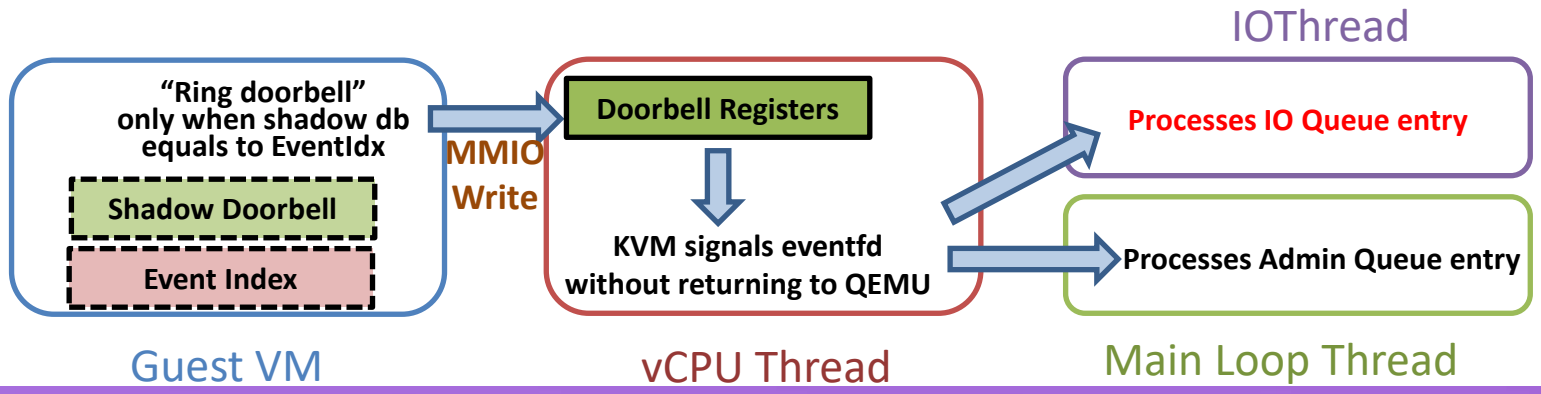
# Performance After irqfd

- irqfd is not necessarily faster than KVM ioctl interrupt injection although it bypasses QEMU

| QD | 1 | 4 | 16 | 64 |
|---|---|---|---|---|
| nvme | 31 | 30 | 30 | 30 |
| +shadow db | 35 | 121 | 176 | 153 |
| +ioeventfd | 41 | 133 | 258 | 313 |
| +irqfd | 41 | 136 | 242 | 338 |
| virtio-blk | 59 | 185 | 260 | 256 |

# IOThread

- NVMe Admin Queue
  - Often involve memory region transactions
  - Emulated in **main loop thread** with BQL held
- NVMe IO Queues
  - Emulated in **IOThread** to get predictable performance



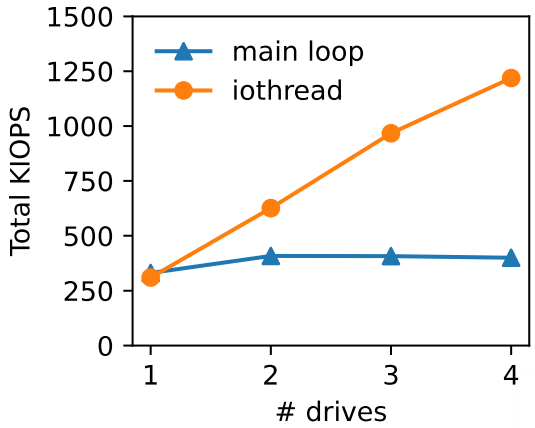
# IOThread

- With *AioContext*, changing emulation thread is easy!
  - event_notifier_set_handler -> aio_set_event_notifier
  - timer_new -> aio_timer_new
  - qemu_bh_new -> aio_bh_new
- Remember to hook up the correct *AioContext* !

# Performance After IOThread

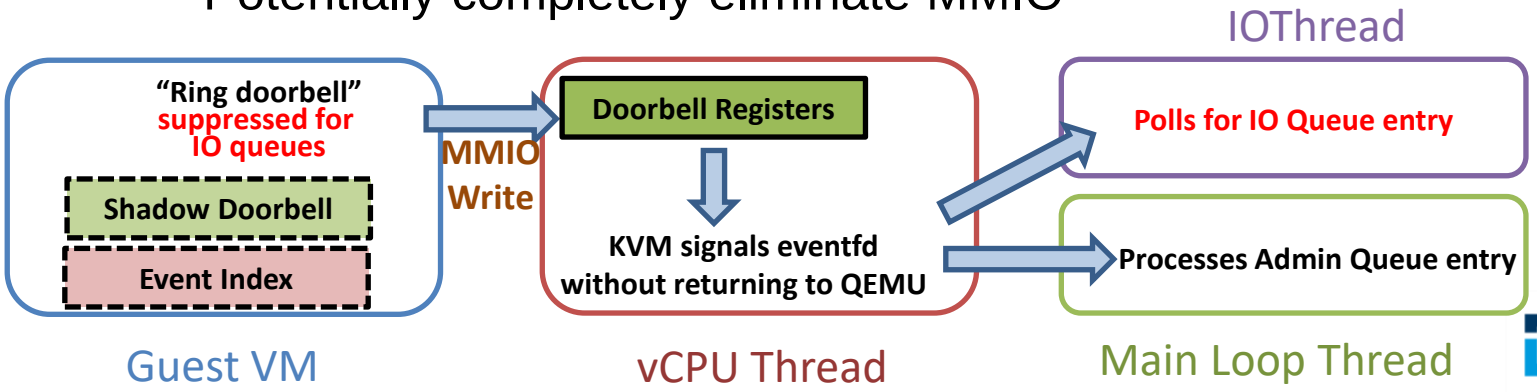
- Slight improvement at low QD because IOThread has a lightweight event loop
- IOPS grow linearly with number of devices

| QD | 1 | 4 | 16 | 64 |
|---|---|---|---|---|
| nvme | 31 | 30 | 30 | 30 |
| +shadow db | 35 | 121 | 176 | 153 |
| +ioeventfd | 41 | 133 | 258 | 313 |
| +irqfd | 41 | 136 | 242 | 338 |
| +iothread | 53 | 155 | 245 | 309 |
| virtio-blk | 59 | 185 | 260 | 256 |



# Polling

- "Poll for submission" in order to
  - Start command processing as soon as SQE becomes available
    - No extra latency from MMIO and ioeventfd processing
  - (In theory) No need to ring doorbells anymore
    - Potentially completely eliminate MMIO



# In The End

- ~700 LOC change
- Performance on par with virtio-blk under both polling and non-polling setup

| QD | 1 | 4 | 16 | 64 |
|---|---|---|---|---|
| nvme | 53 | 155 | 245 | 309 |
| virtio-blk | 59 | 185 | 260 | 256 |
| nvme+polling | 123 | 165 | 189 | 191 |
| virtio-blk+polling | 88 | 212 | 210 | 213 |

Still investigating why polling has worse IOPS at high QD

# Lessons Learned

- The source *(hw/virtio*)* is basically **the** documentation for this stuff
  - You have to **know** that you need an eventfd-based interrupt mechanism for thread safety
  - You have to **know** that you should hook up the MSI-X vector notifiers for irqfd-based interrupts to work correctly

# Lessons Learned

- Want to add iothread to your device?
  - **1) Are my mmio handlers safe?**
    - Make sure you schedule work on the right thread
  - **2) Are my interrupt handlers thread safe?**
    - Use an eventfd notifier to schedule the handler on a specific thread

# A Wild NVMe Spec Violation Appeared

- Specification requires doorbell buffers be used on **all** queues, including the **Admin Queue**
  - **But**... No existing drivers (Linux, SPDK) or devices (SPDK's vfio-user) uses it on the Admin Queue
  - Can **not** be fixed in drivers

**Figure 164: Doorbell Buffer Config – Shadow Doorbell and EventIdx**

| Start (Offset in Buffer)[1, 2] | End (Offset in Buffer)[1, 2] | Description [2] |
|---|---|---|
| 00h | 03h | Submission Queue 0 Tail Doorbell or EventIdx (Admin) |
| 00h + (1 * (4 << CAP.DSTRD)) | 03h + (1 * (4 << CAP.DSTRD)) | Completion Queue 0 Head Doorbell or EventIdx (Admin) |
| 00h + (2 * (4 << CAP.DSTRD)) | 03h + (2 * (4 << CAP.DSTRD)) | Submission Queue 1 Tail Doorbell or EventIdx |
| 00h + (3 * (4 << CAP.DSTRD)) | 03h + (3 * (4 << CAP.DSTRD)) | Completion Queue 1 Head Doorbell or EventIdx |
| … | … | … |

# The “fix” *(Keith Busch)*

- Overwrite the shadow doorbell buffer value with the doorbell register value in the doorbell mmio handler
  - Safe (in vmexit/trap context)
- Works for both compliant and non-compliant host drivers
- Drivers will probably continue to be non-compliant in this regard

# Future Work

- Making hw/nvme a viable virtio-blk alternative for cloud deployments
  - Needs live migration support
  - Split off a version of the controller without all the faked features (Simple Copy, Zoned Namespace emulation, etc.)
  - Security audit

# Future Work

- Additional iothread optimizations
  - An iothread **per namespace**?
    - Submission queues are not exclusive to namespaces, still need a thread for those
  - An iothread **per submission queue**?

# Patches

- Shadow doorbell
  - hw/nvme: Add shadow doorbell buffer support
- Ioeventfd
  - hw/nvme: Use ioeventfd to handle doorbell updates
- Irqfd
  - hw/nvme: support irq (de)assertion with eventfd
  - hw/nvme: use KVM irqfd when available
- IOThread
  - hw/nvme: add iothread support
- Polling
  - hw/nvme: add polling support

# KVM FORUM