cvm-io

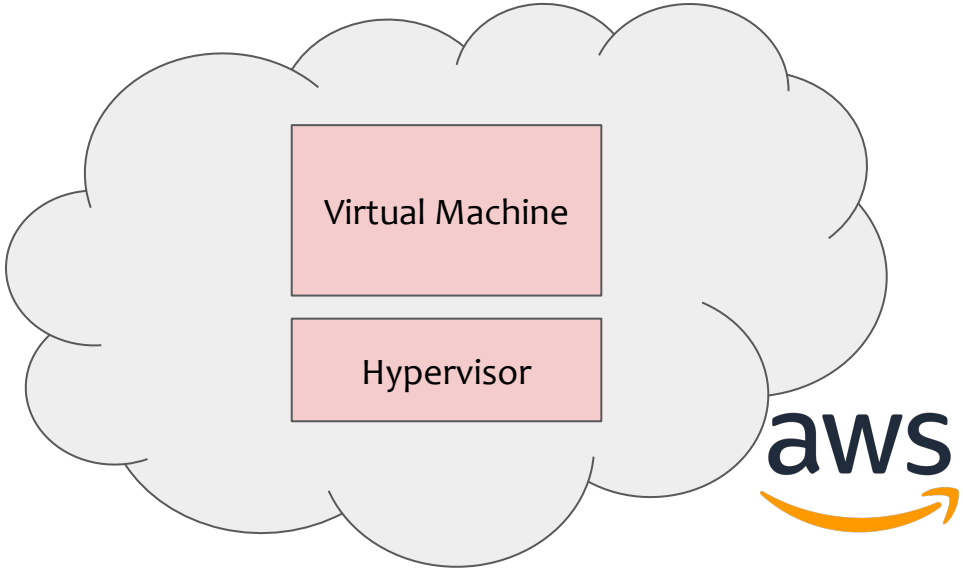
Improving Storage IO of Confidential Virtual Machines

Group Meeting

2024-01-15

Robert Schambach <scha@in.tum.de>

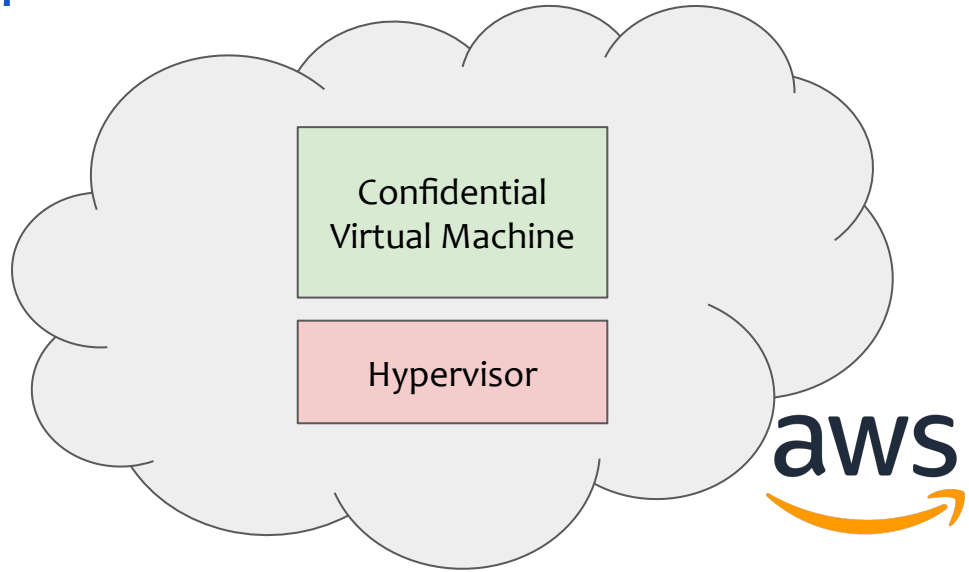
Security in the cloud



Untrusted cloud infrastructure

How do we protect VM code and data in untrusted cloud environments?

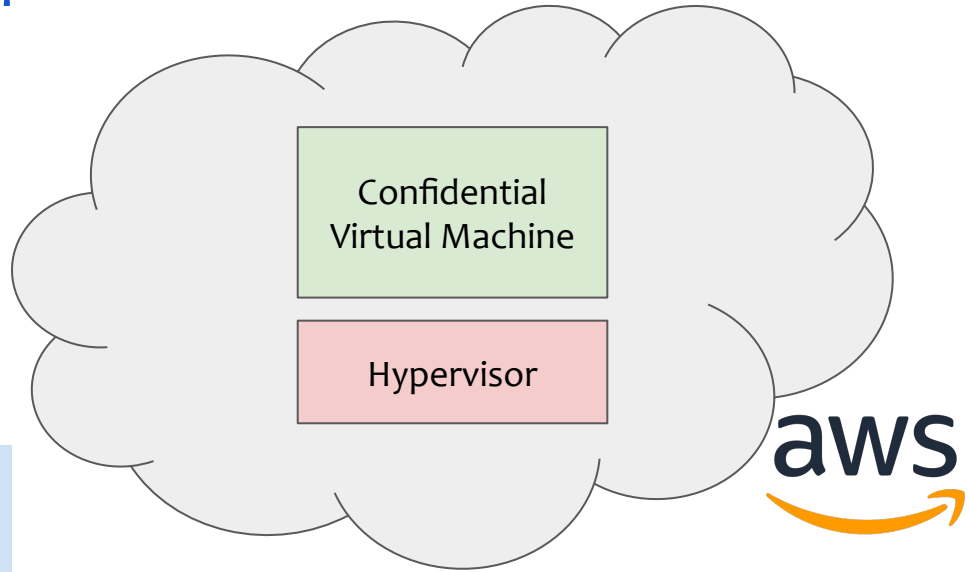
Confidential Computation



Untrusted cloud
infrastructure

Hardware-assisted Confidential Virtual Machines (CVMs)

Confidential Computation



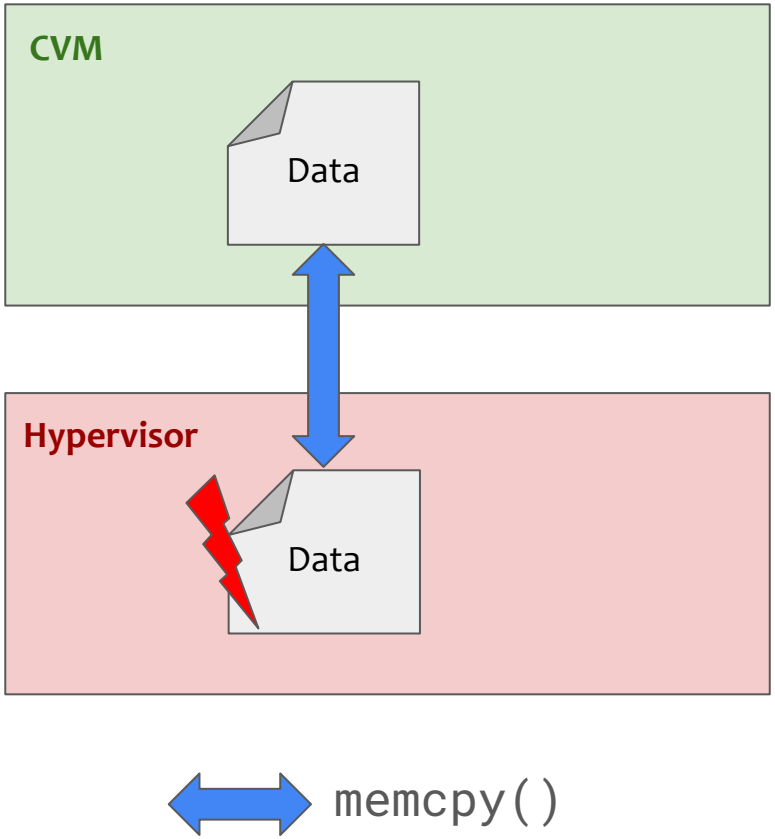
Untrusted cloud infrastructure

Major vendors support

- **Intel TDX**
- **AMD SEV-SNP**
- **ARM CCA**

Hardware-assisted Confidential Virtual Machines (CVMs)

Storage I/O in CVMs: no protection outside of CVM



CVM Guarantees

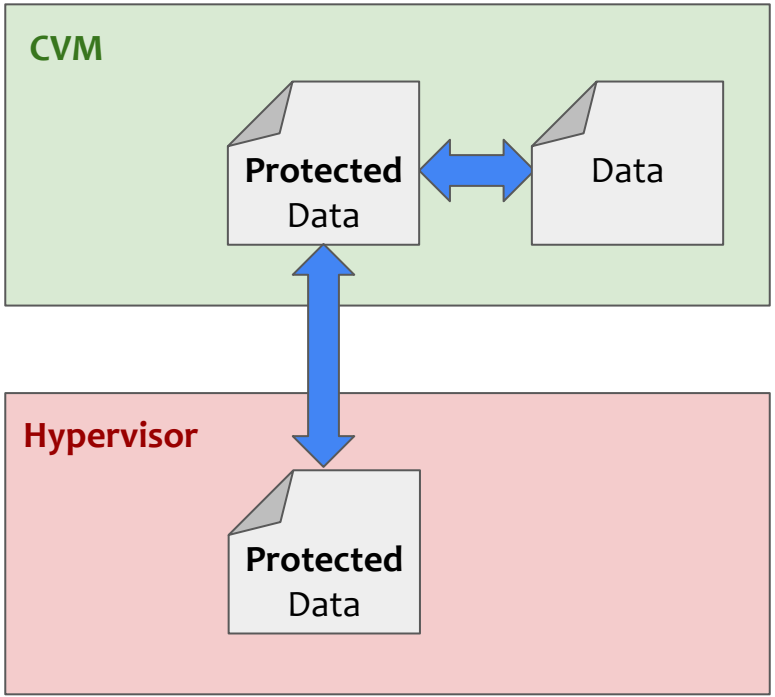
- **Confidentiality**
- **Integrity**
- **Freshness**
- **Authenticity**

Subject to:

- Unauthorized read
- Modifications
- Replay

…

Storage I/O in CVMs: no protection outside of CVM



memcpy()

CVM Guarantees

- **Confidentiality**
- **Integrity**
- **Freshness**
- **Authenticity**

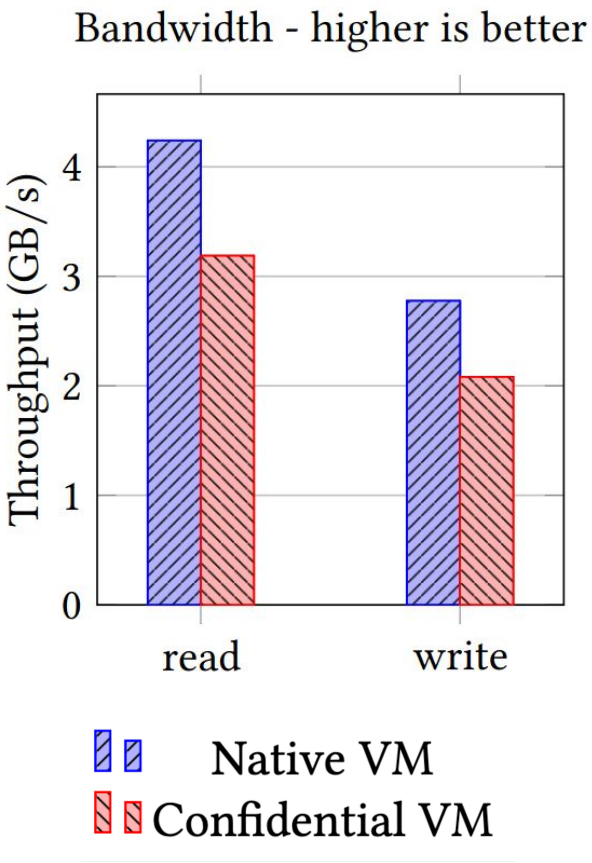
Subject to:

- Unauthorized read
- Modifications
- Replay

…

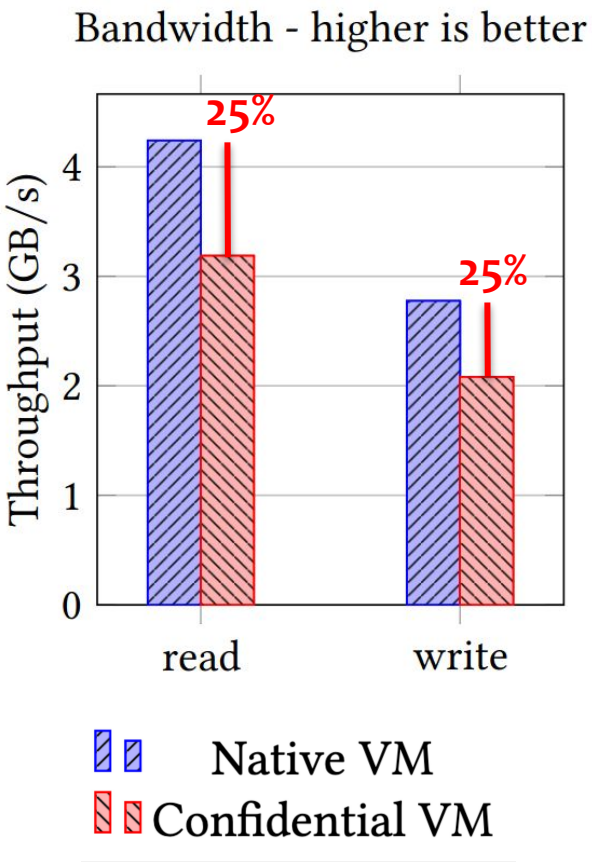
Storage I/O Performance without trusting hypervisor

- Synthetic benchmark (fio-v3.36) w/ **dm-crypt**
  - More detail in Eval
- Large performance difference!



Storage I/O Performance without trusting hypervisor

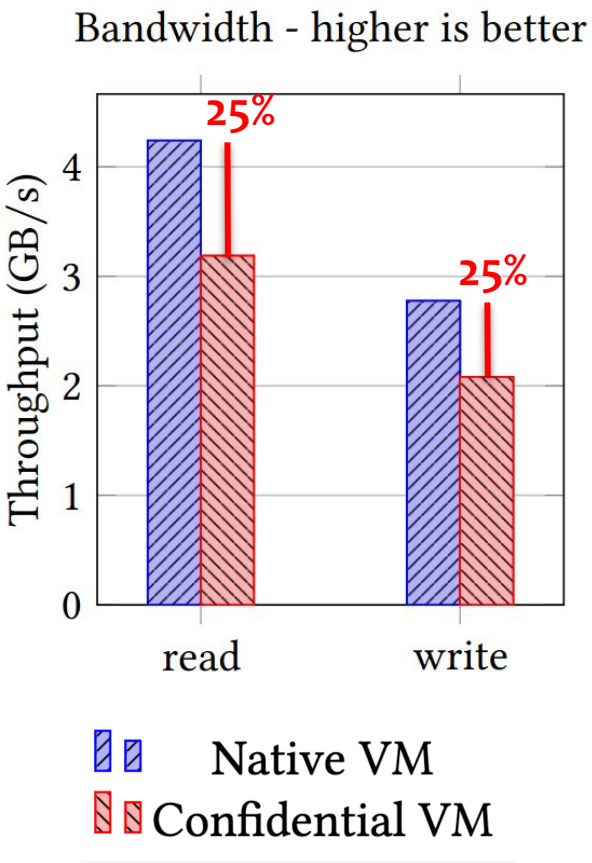
- Synthetic benchmark (fio-v3.36) w/ **dm-crypt**
  - More detail in Eval
- Large performance difference!
- Up-to **25%** overhead **(preliminary!)**



Storage I/O Performance without trusting hypervisor

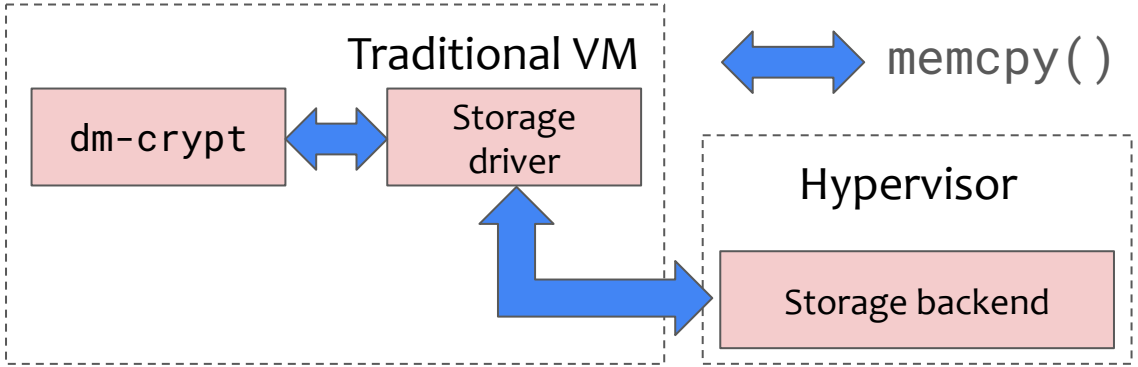
- Synthetic benchmark (fio-v3.36) w/ **dm-crypt**
  - More detail in Eval
- Large performance difference!
- Up-to **25%** overhead **(preliminary!)**

Why does CVM storage IO exhibit such high overheads?



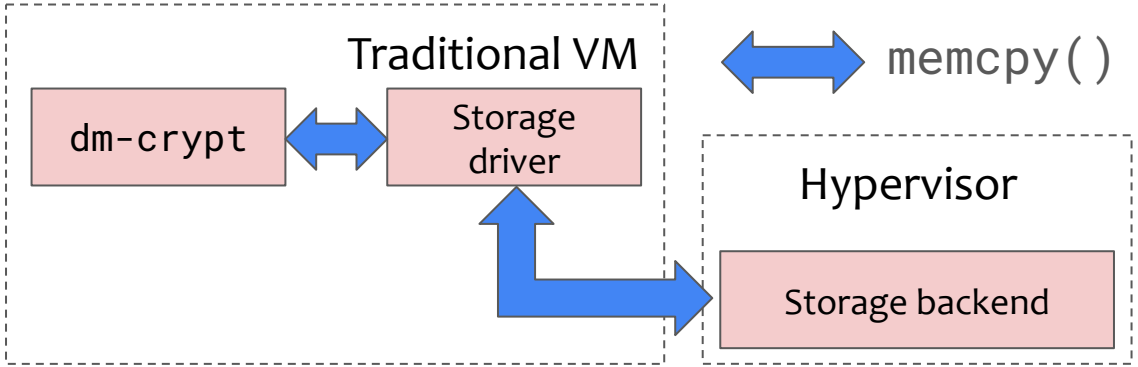
Paravirtual Storage I/O in CVMs: Data Transfer

- Primary virtualization choice for cloud providers
  - Typical use-case: high performance userspace I/O backend



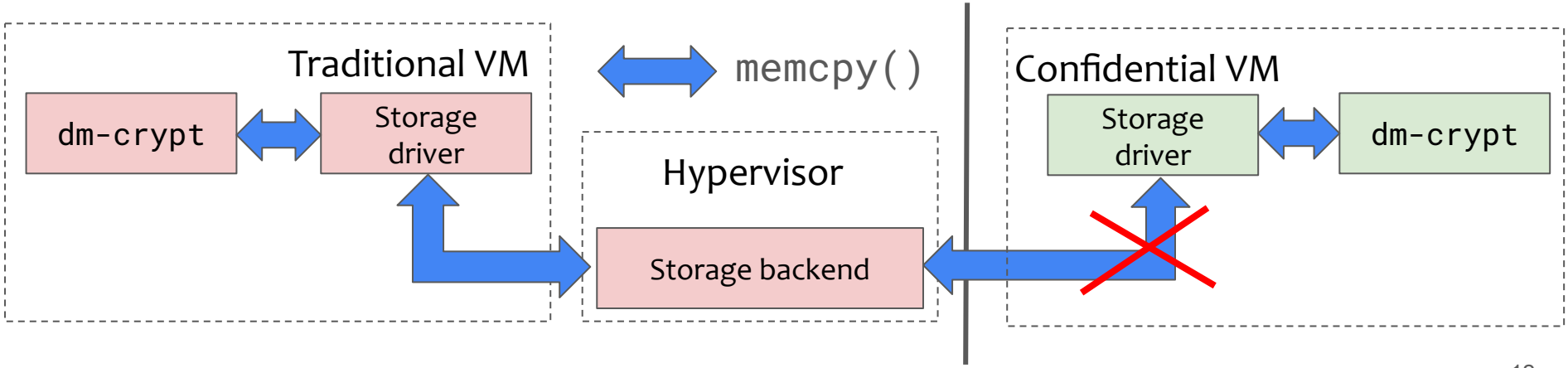
Paravirtual Storage I/O in CVMs: Data Transfer

- Primary virtualization choice for cloud providers
  - Typical use-case: high performance userspace I/O backend
- I/O data transfer
  - HV copies I/O data to/from VM memory



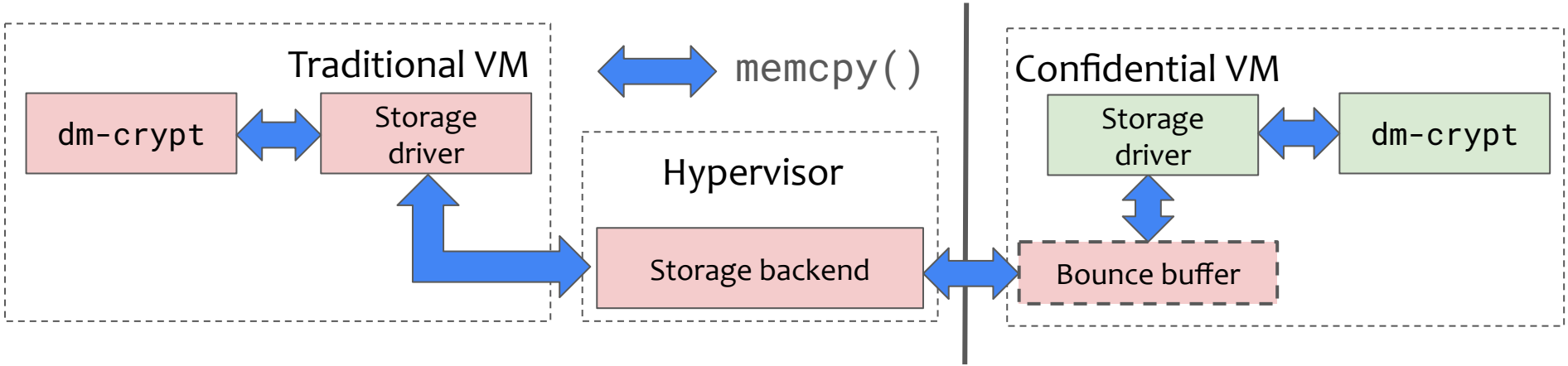
Paravirtual Storage I/O in CVMs: Data Transfer

- Primary virtualization choice for cloud providers
  - Typical use-case: high performance userspace I/O backend
- I/O data transfer
  - HV copies I/O data to/from VM memory
  - **Not applicable to CVMs!**



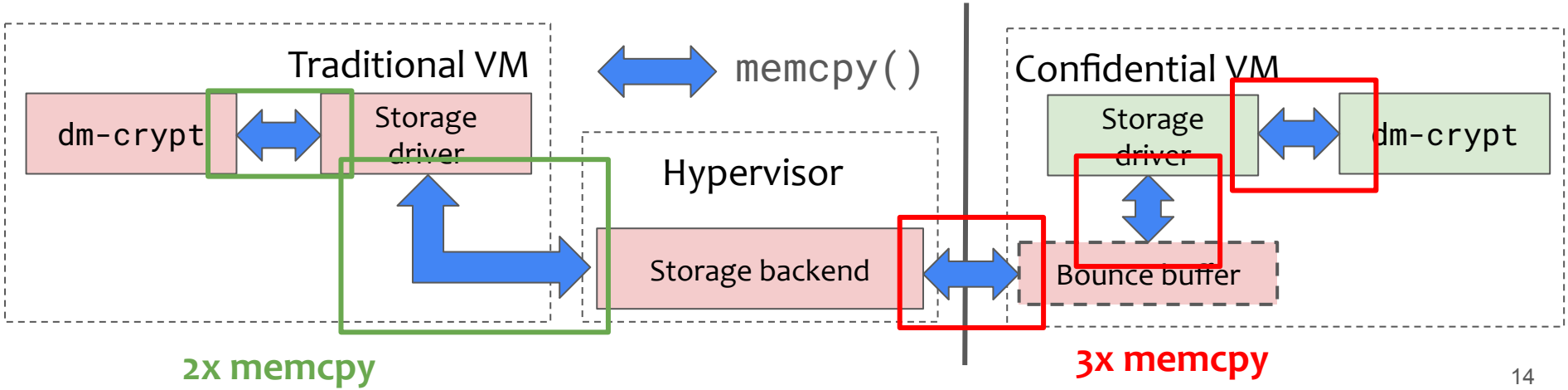
Paravirtual Storage I/O in CVMs: Data Transfer

- Primary virtualization choice for cloud providers
  - Typical use-case: high performance userspace I/O backend
- I/O data transfer
  - HV copies I/O data to/from VM memory
  - **Not applicable to CVMs!**
  - CVMs require bounce buffer as hypervisor cannot access private memory



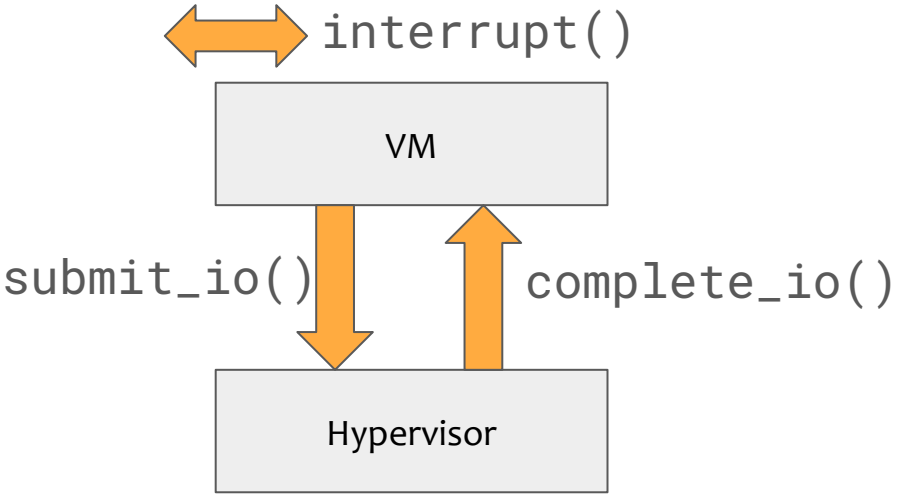
Paravirtual Storage I/O in CVMs: Data Transfer

- Primary virtualization choice for cloud providers
  - Typical use-case: high performance userspace I/O backend
- I/O data transfer
  - HV copies I/O data to/from VM memory
  - **Not applicable to CVMs!**
  - CVMs require bounce buffer as hypervisor cannot access private memory



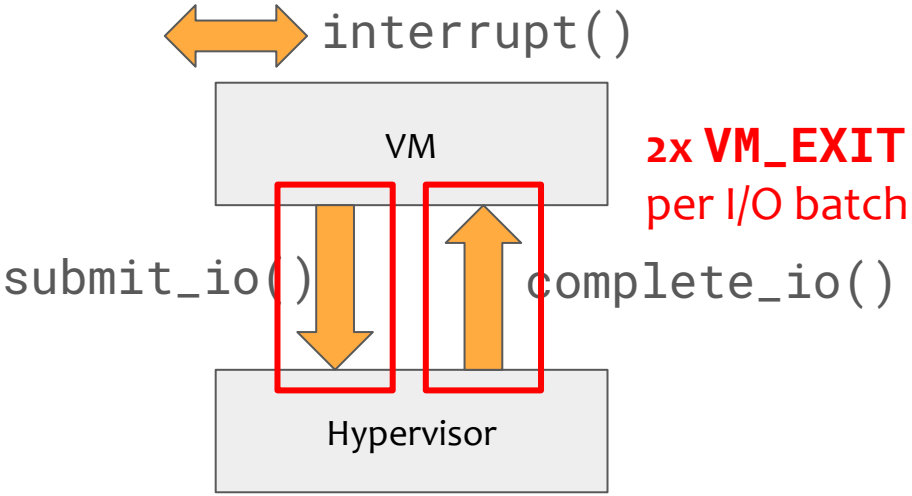
Paravirtual Storage I/O in CVMs: I/O Notifications

- I/O submission and completion: notifications via **interrupts**



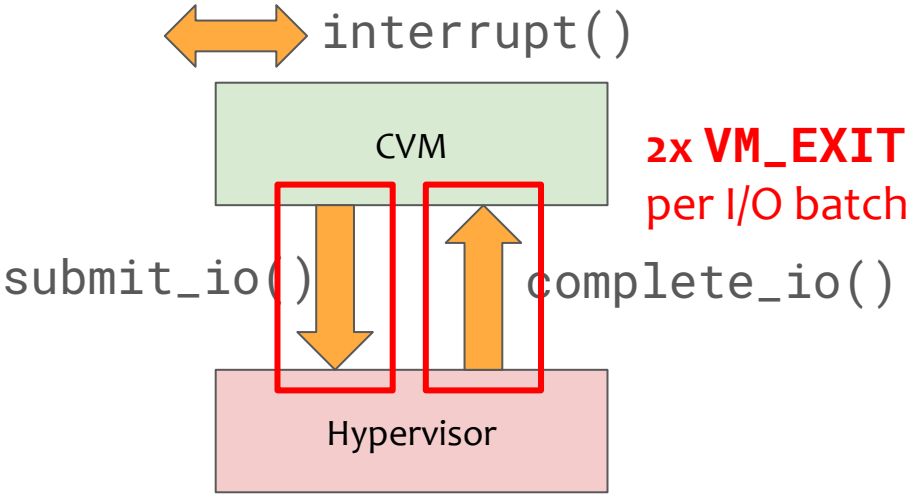
Paravirtual Storage I/O in CVMs: I/O Notifications

- I/O submission and completion: notifications via **interrupts**
- Interrupts cause **`VM_EXIT`**s



Paravirtual Storage I/O in CVMs

- I/O submission and completion: notifications via **interrupts**
- Interrupts cause **`VM_EXIT`**s



Confidential `VM_EXIT`s:
More expensive:

- Save CPU registers
- Scrub CPU registers
- Additional context switch

Research Question

How can we **design** the CVM storage IO stack to:

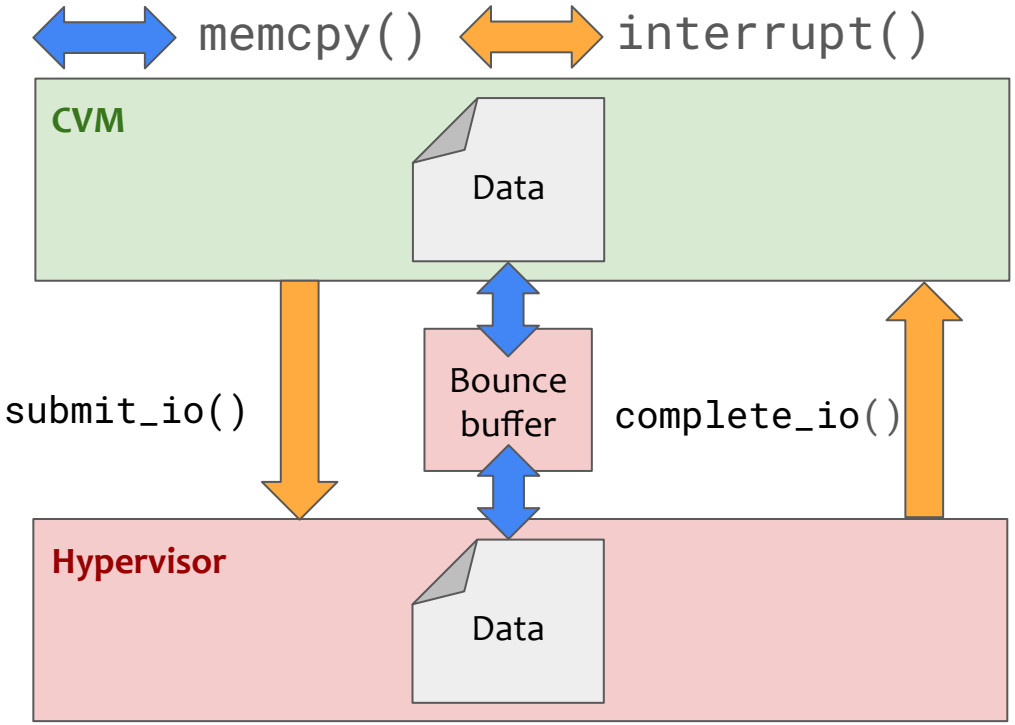
- **Remove performance loss** while
- **Upholding CVM guarantees to storage IO**

Stack protection goals:

- Confidentiality
- Integrity
- Freshness
- Authenticity

cvm-io: A storage IO stack for CVMs

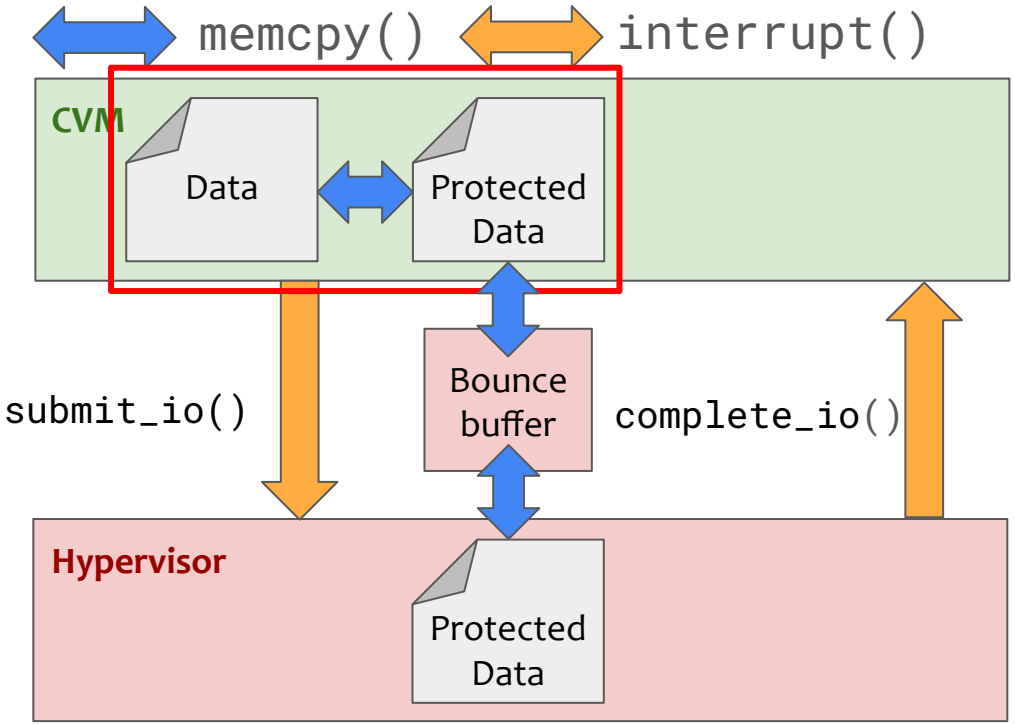
We redesign the CVM storage IO stack by adding…



cvm-io: A storage IO stack for CVMs

We redesign the CVM storage IO stack by adding…

- **Data Protection**

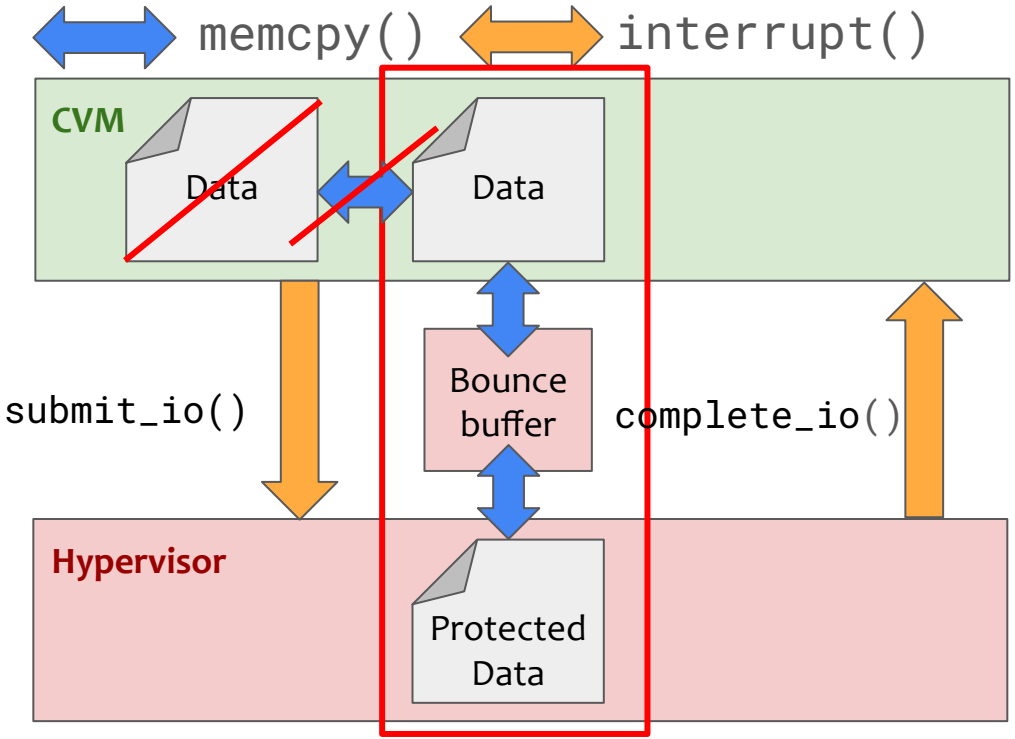


cvm-io: A storage IO stack for CVMs

We redesign the CVM storage IO stack by adding…

- Data Protection
- **Zero-copy**

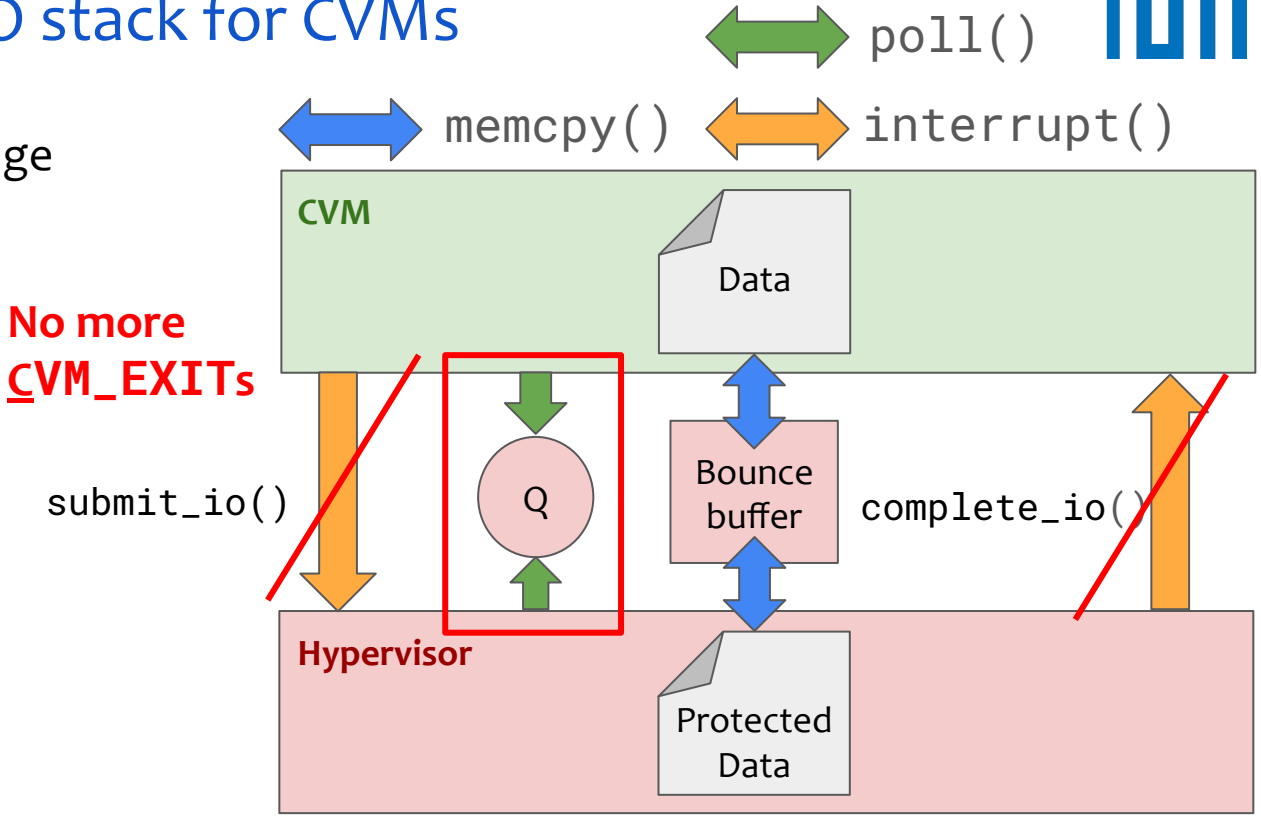
2x memcpy;
Not 3x



cvm-io: A storage IO stack for CVMs

We redesign the CVM storage IO stack by adding…

- Data Protection
- Zero-copy
- **Polling**



Design

1

Protection

2

Zero copy

3

Polling

1

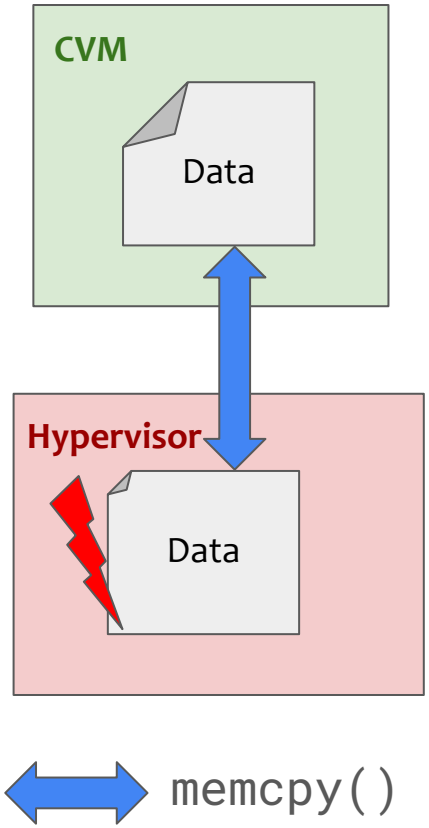
Protection

Extending CVM guarantees to storage IO

Extending CVM Protection Guarantees to Storage IO

Storage IO protection goals:

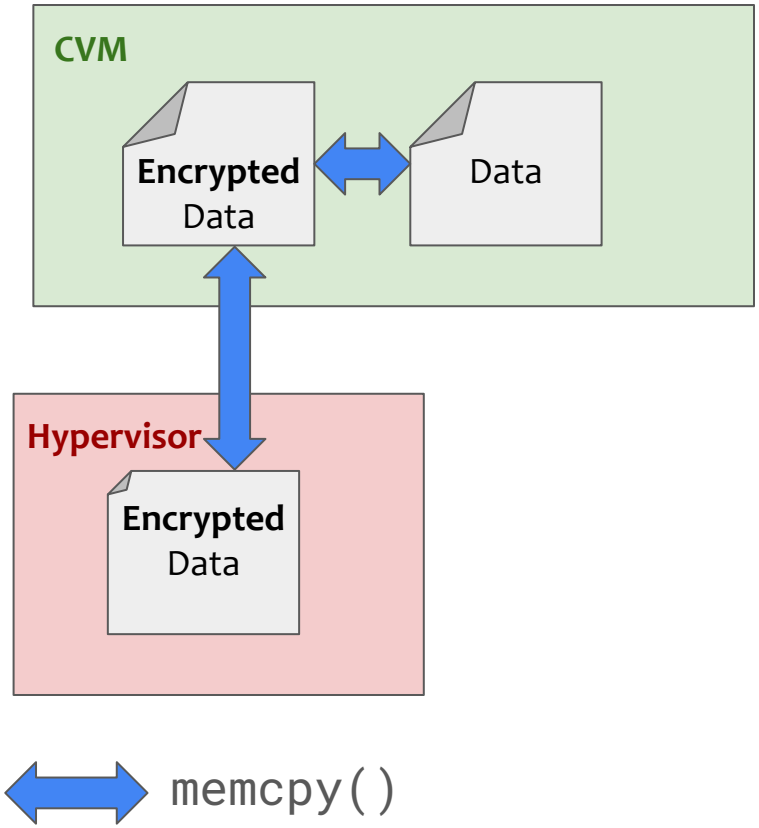
- Confidentiality
- Integrity
- Authenticity
- Freshness



Extending CVM Protection Guarantees to Storage IO

Storage IO protection goals:

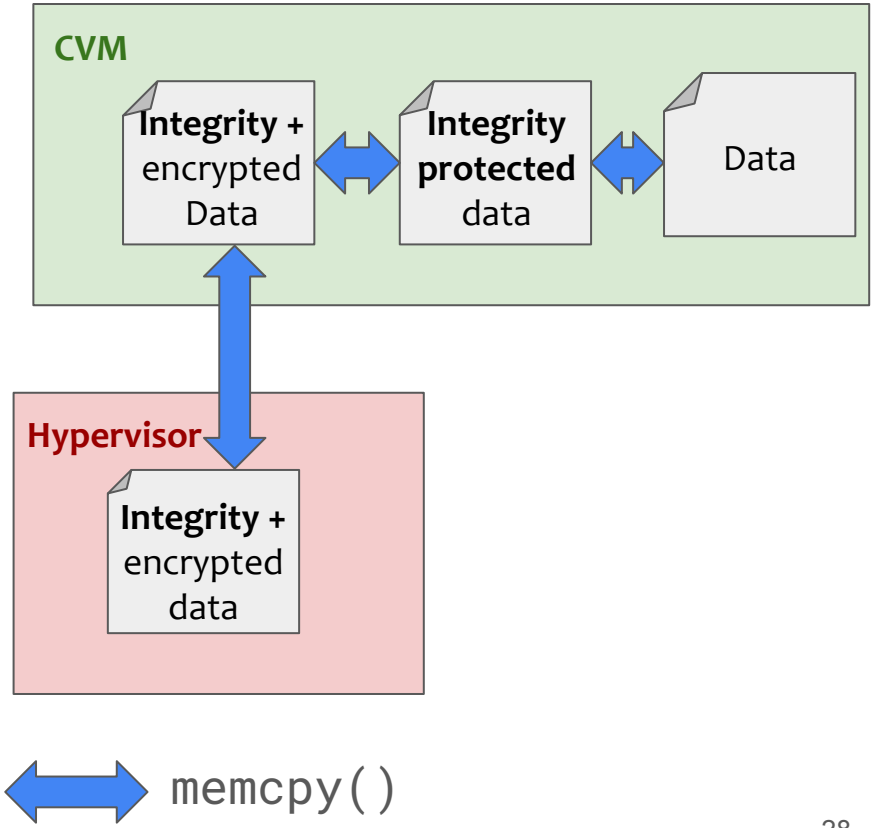
- Confidentiality ➡️ `dm-crypt`
- Integrity
- Authenticity ➡️ `dm-crypt`
- Freshness



Extending CVM Protection Guarantees to Storage IO

Storage IO protection goals:

- Confidentiality ➡ `dm-crypt`
- Integrity ➡ `dm-integrity`
- Authenticity ➡ `dm-crypt`
- Freshness ➡ **Future work**



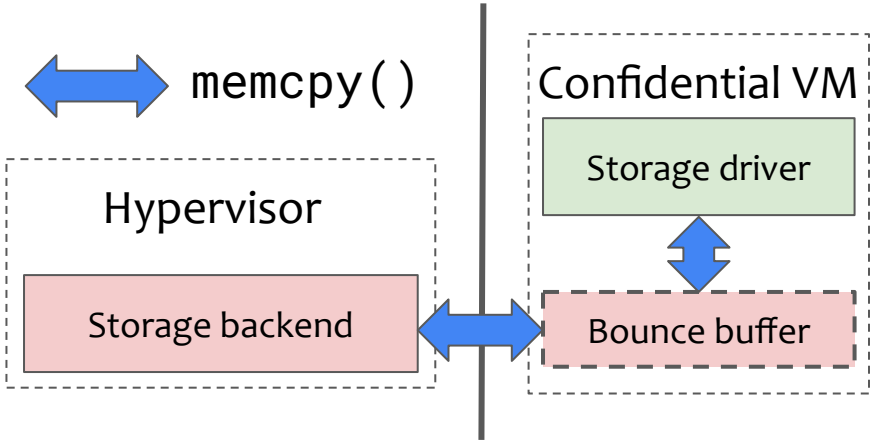
Only **sector level** integrity protection and **no replay** protection

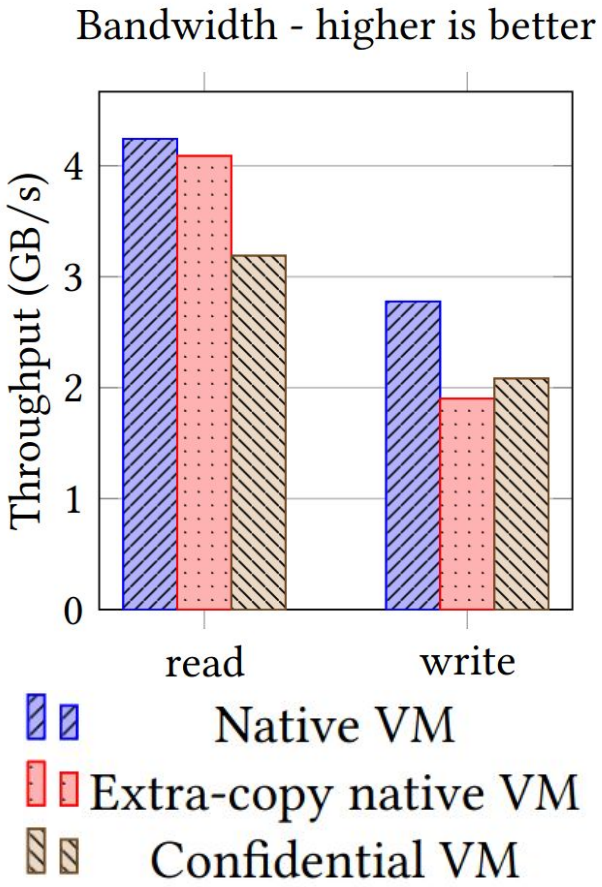
2

Zero copy:

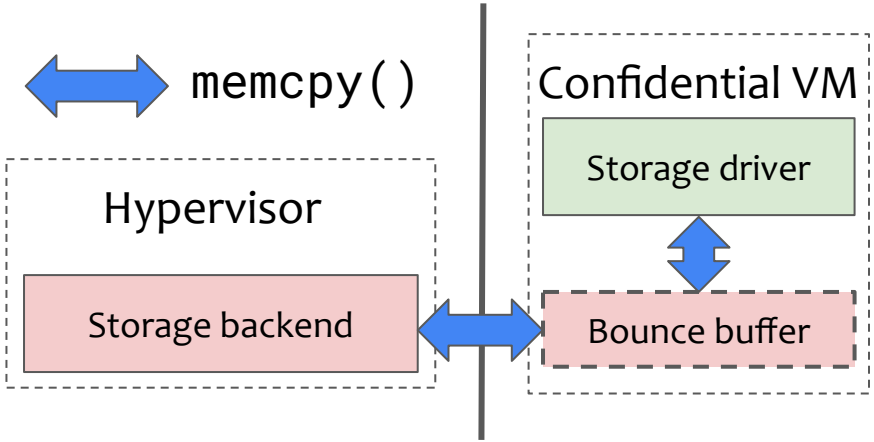
Avoiding extra copy

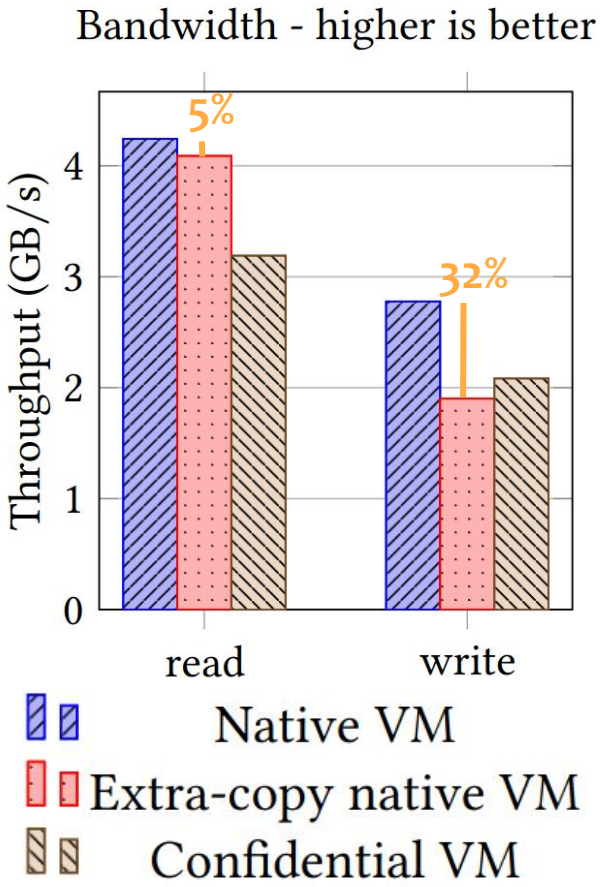
Extra copy overhead



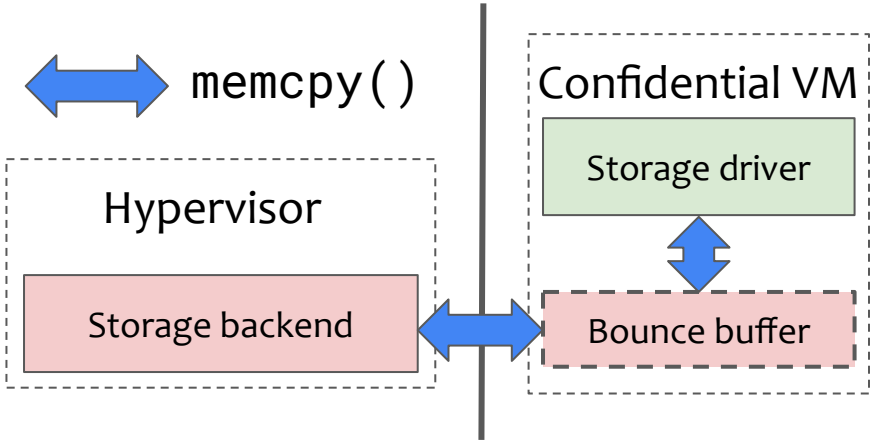


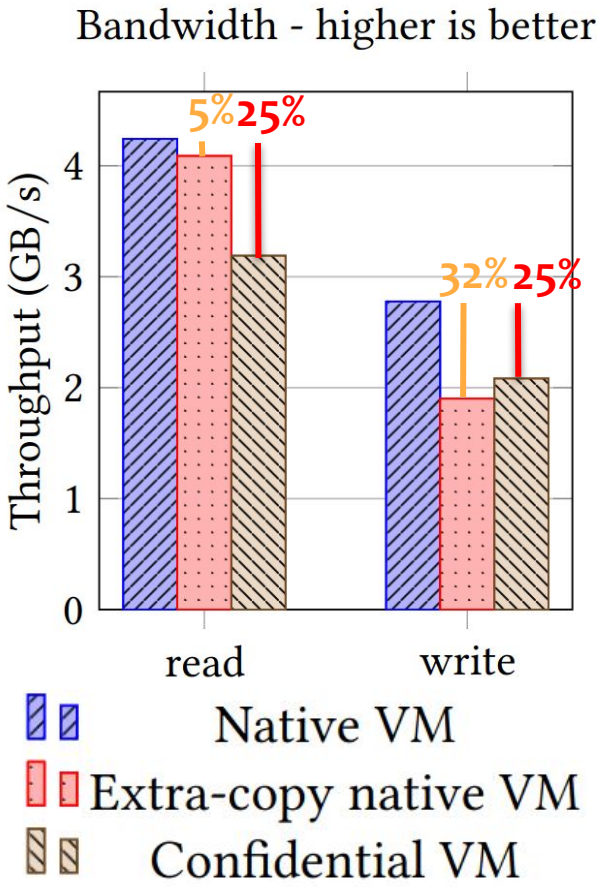
Extra copy overhead





Extra copy overhead

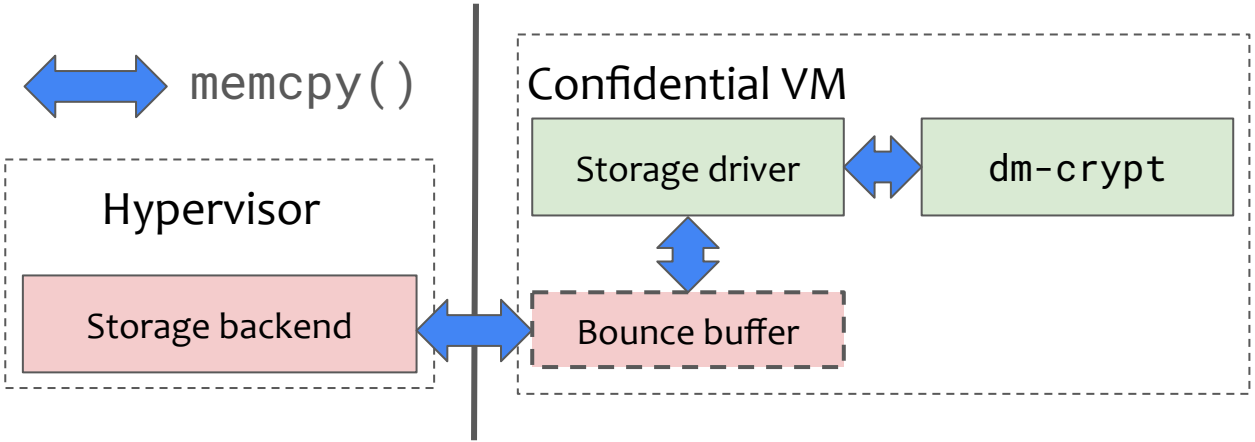




How can we avoid the extra copy of the bounce buffer?

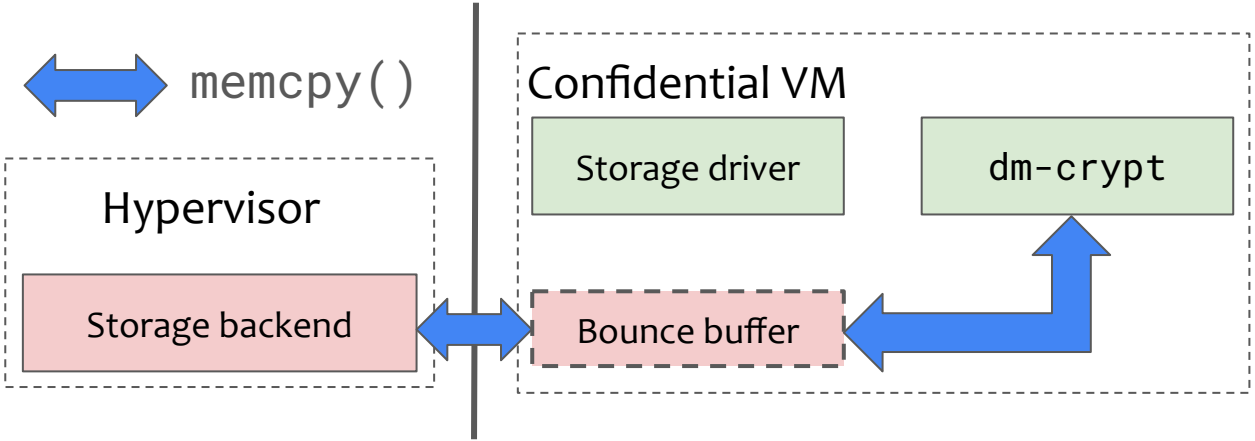
Inspecting CVM Storage Stack for zero-copy

- As IO is unprotected -> we add encryption
  - E.g. `dm-crypt`



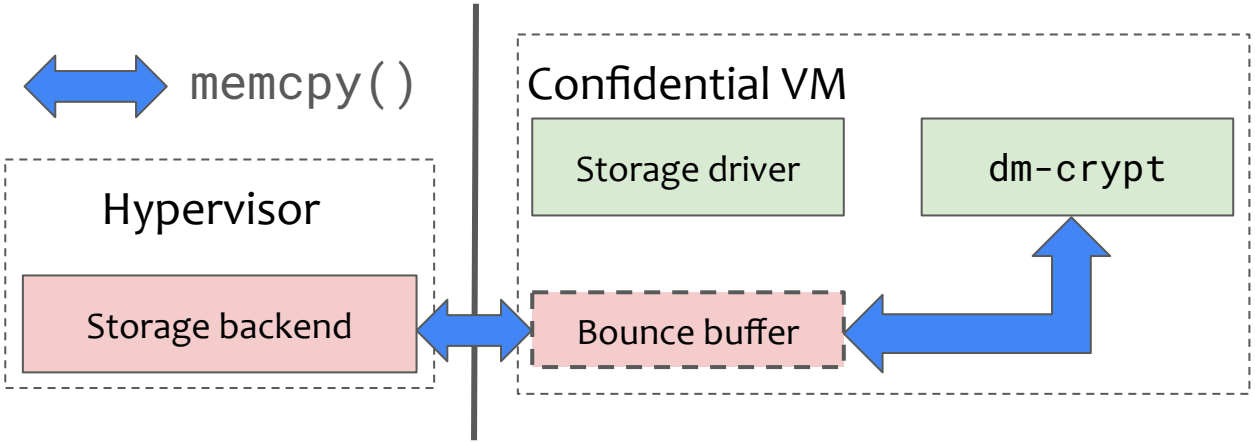
Inspecting CVM Storage Stack for zero-copy

- As IO is unprotected -> we add encryption
  - E.g. `dm-crypt`
- As IO is protected -> we can encrypt to shared memory



Inspecting CVM Storage Stack for zero-copy

- As IO is unprotected -> we add encryption
  - E.g. `dm-crypt`
- As IO is protected -> we can encrypt to shared memory
- Save extra copy!

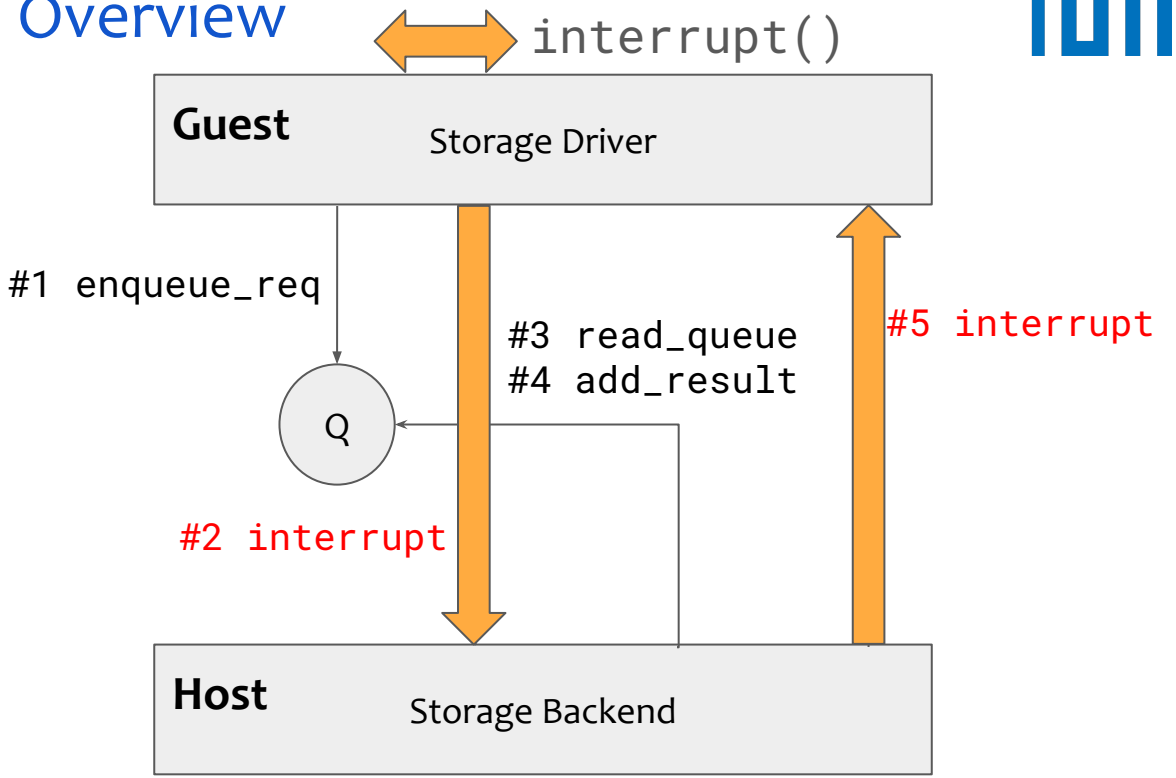


3

Polling:
Avoiding VM_EXITs

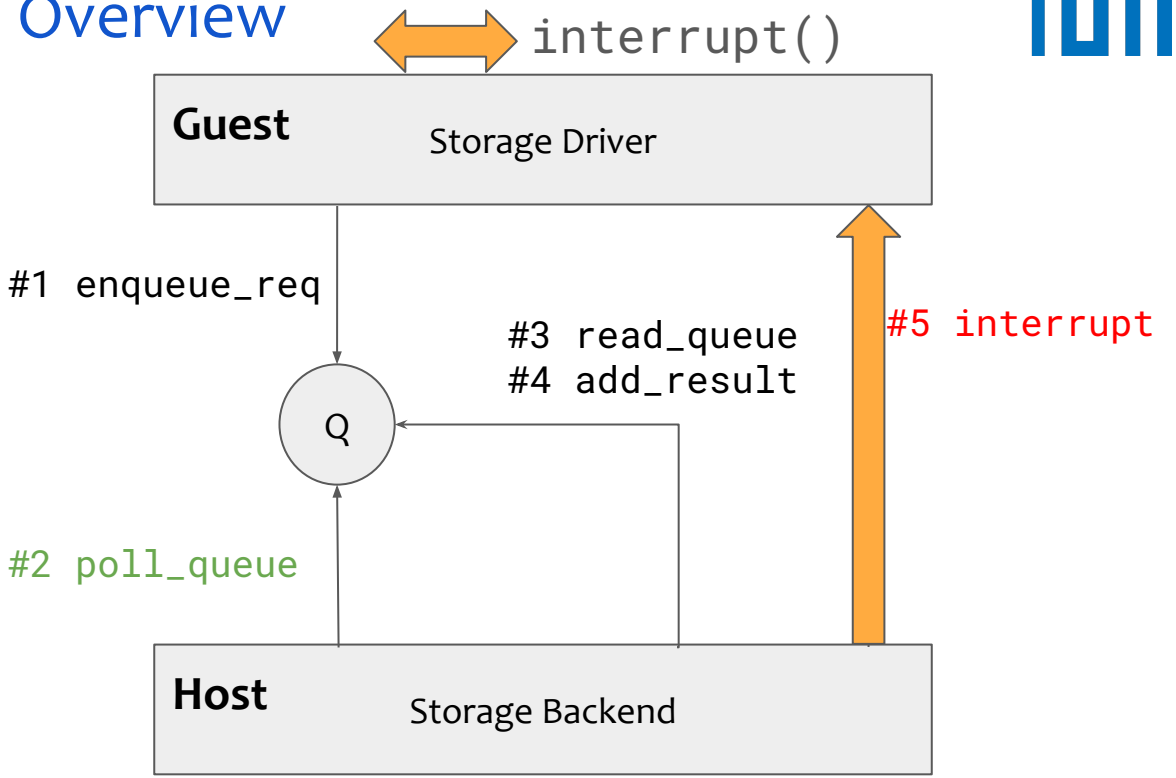
Paravirtual Storage IO Overview

- Default: interrupt-based



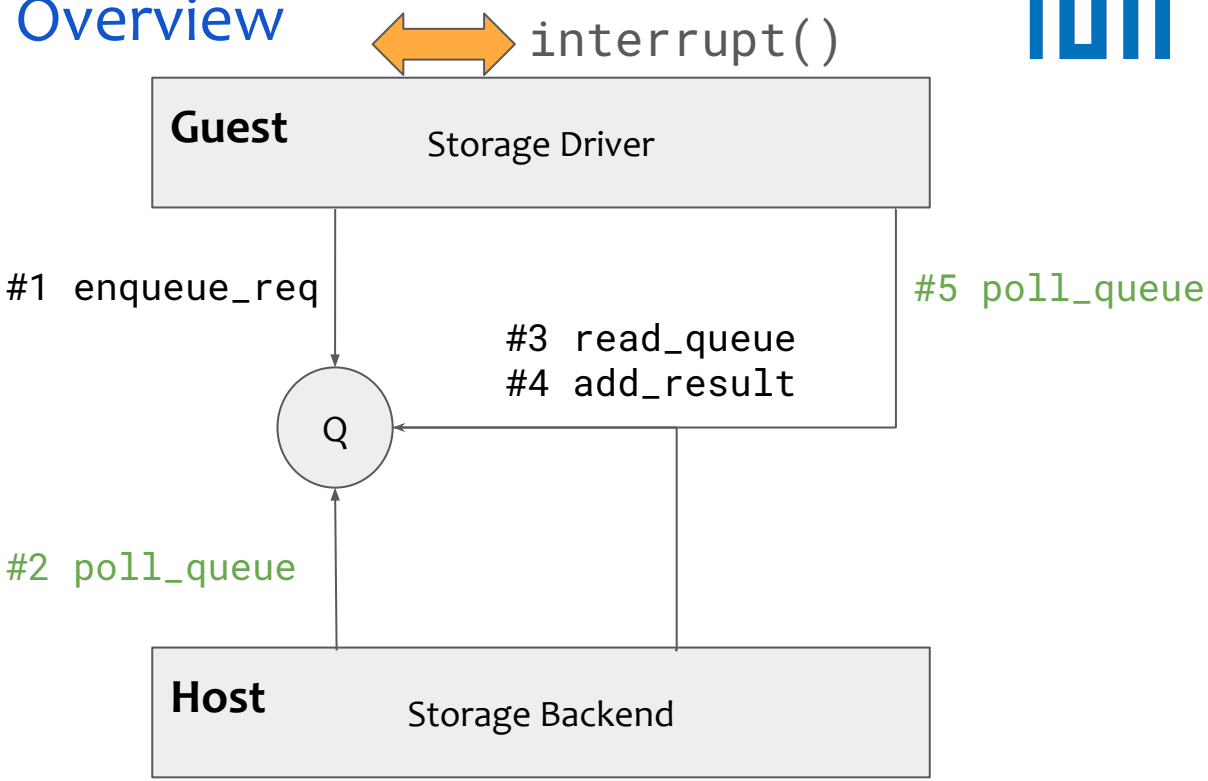
Paravirtual Storage IO Overview

- Default: interrupt-based
- Add host-side polling (submission polling)
  - Avoid guest-side interrupt



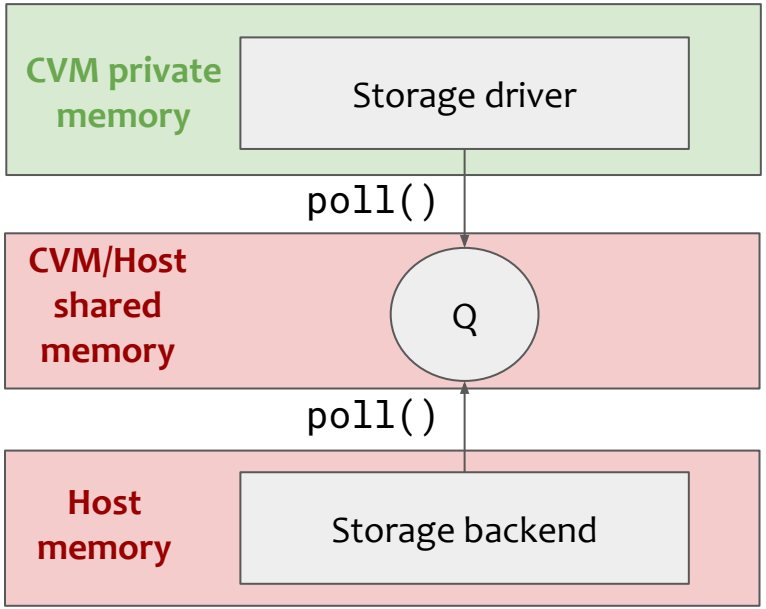
Paravirtual Storage IO Overview

- Default: interrupt-based
- Add host-side polling (submission polling)
  - Avoid guest-side interrupt
- Add guest-side polling (completion polling)
  - Avoid host-side interrupt



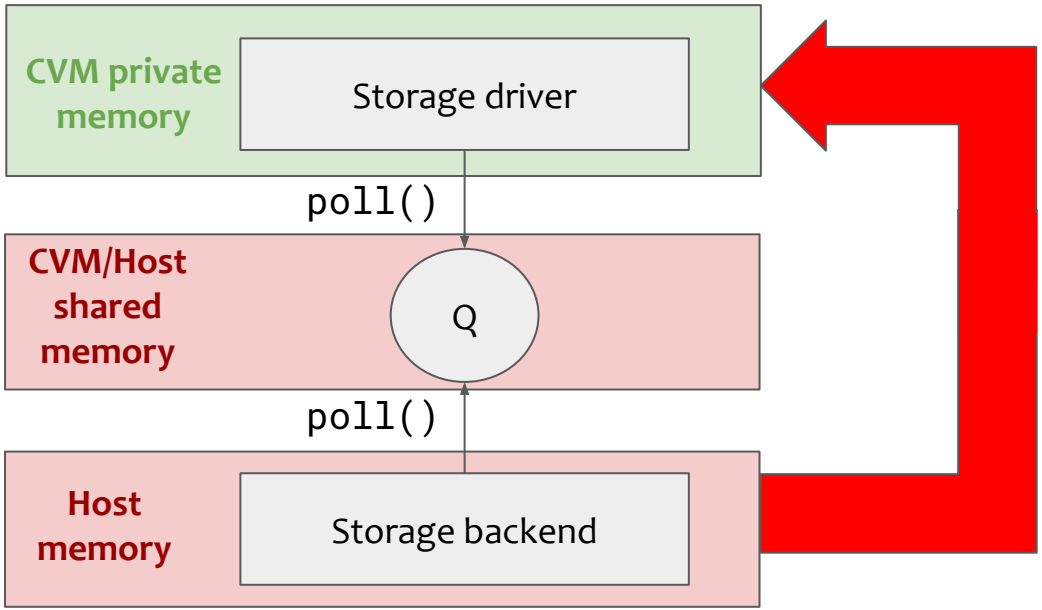
Issue: binds CPU resources

Paravirtual Storage IO in CVMs



No disadvantage to native VM storage IO

Paravirtual Storage IO in CVMs



Needs access to entire VM memory

Implementation: Future Work -> Not in Eval

Evaluation

(Preliminary) Evaluation: Setup and Benchmarks

Object of evaluation:

- cvm-io zero-copy
- w/ `dm-crypt`

Method:

- Synthetic benchmarks -> `fio-v3.36`

Setup:

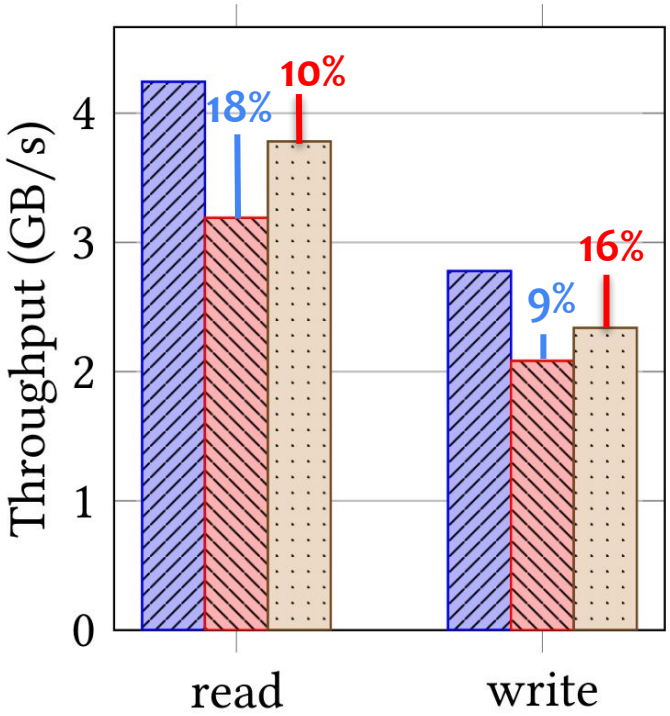
- Host: AMD SEV-SNP Linux
- Hypervisor: AMD Qemu
- Guest: cvm-io-patched Linux
- Storage frontend: **Interrupt-based** virtio-blk (no poll!)
- Storage backend: **Interrupt-based** Qemu virtio-blk device
- Target SSD: Samsung NVMe SSD PM173X 1.5TB

Baseline:

- Insecure: Native VM w/ dm-crypt
- Secure: CVM w/ `dm-crypt` **without zero-copy**

(Preliminary) cvm-io Evaluation

Bandwidth - higher is better



- **Extra-copy overhead eliminated**
  - Overhead corresponds to native VM extra-copy overhead

Native VM

Confidential VM

Zero-copy CVM

Eliminates extra-copy performance gap!

Summary

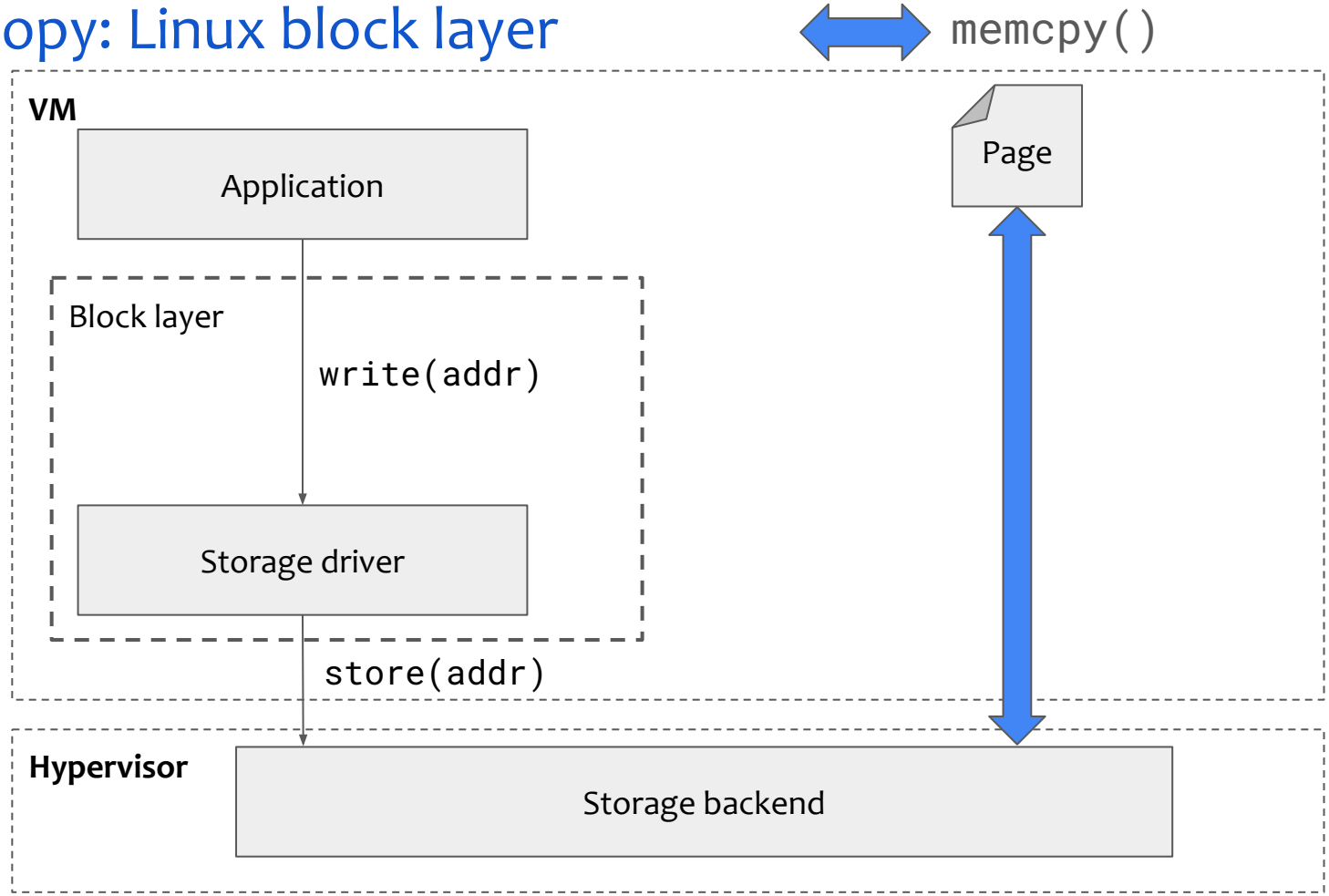
- Current CVMs are not designed for high-performance storage I/O
  - **Bounce buffer bottleneck:** Require extra copy for any I/O
  - **Costly `CVM_EXIT`s:** More expensive than normal VM_EXIT
  - **No CVM guarantees for IO:** No CIA+Replay protection out-of-the-box for storage IO
- cvm-io provides:
  - Near-native storage IO performance
  - Extension of CVM protection guarantees

cvm-io allows CVMs to serve as a secure drop-in VM for storage IO-heavy applications without large performance tax

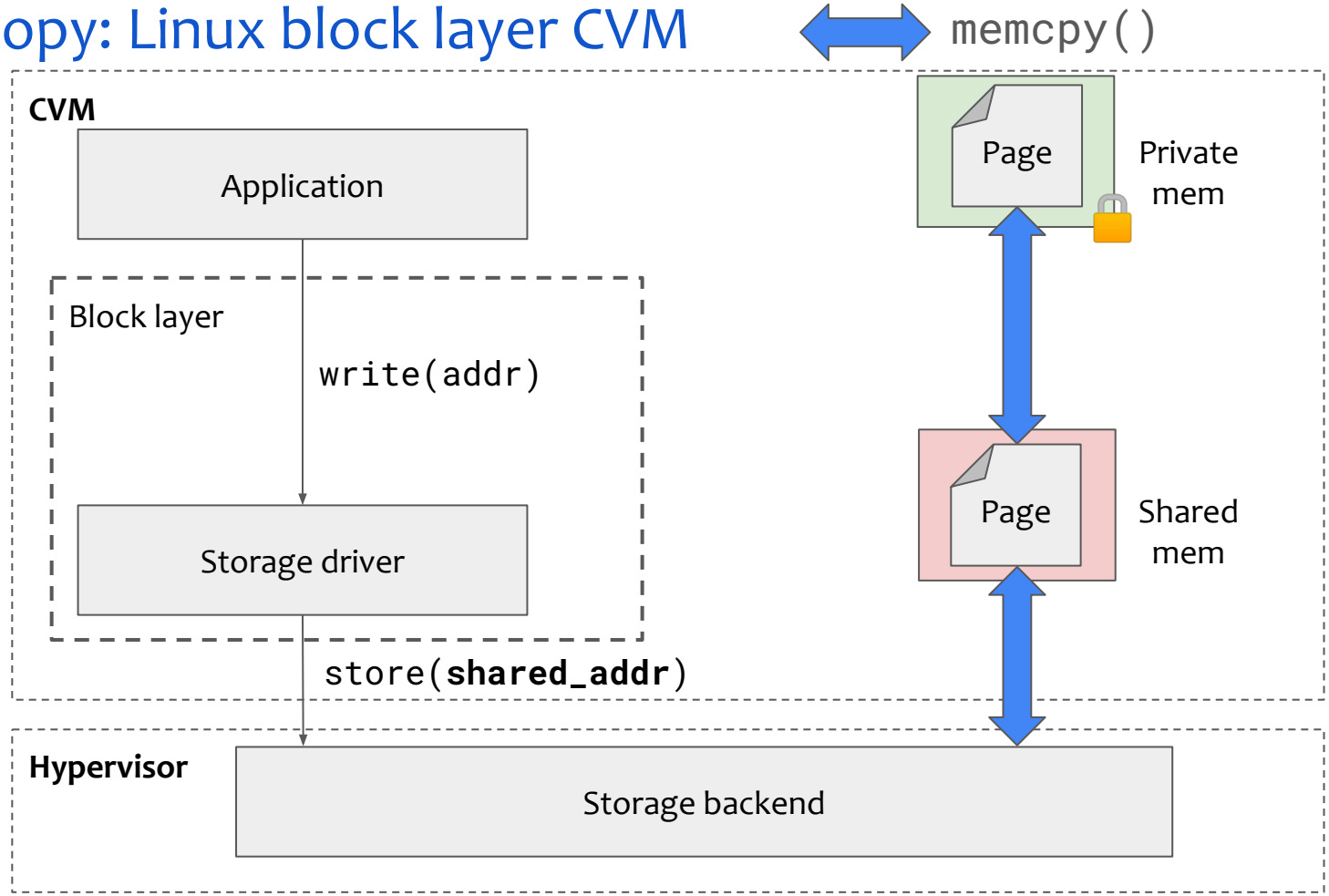
Backup

Implementation

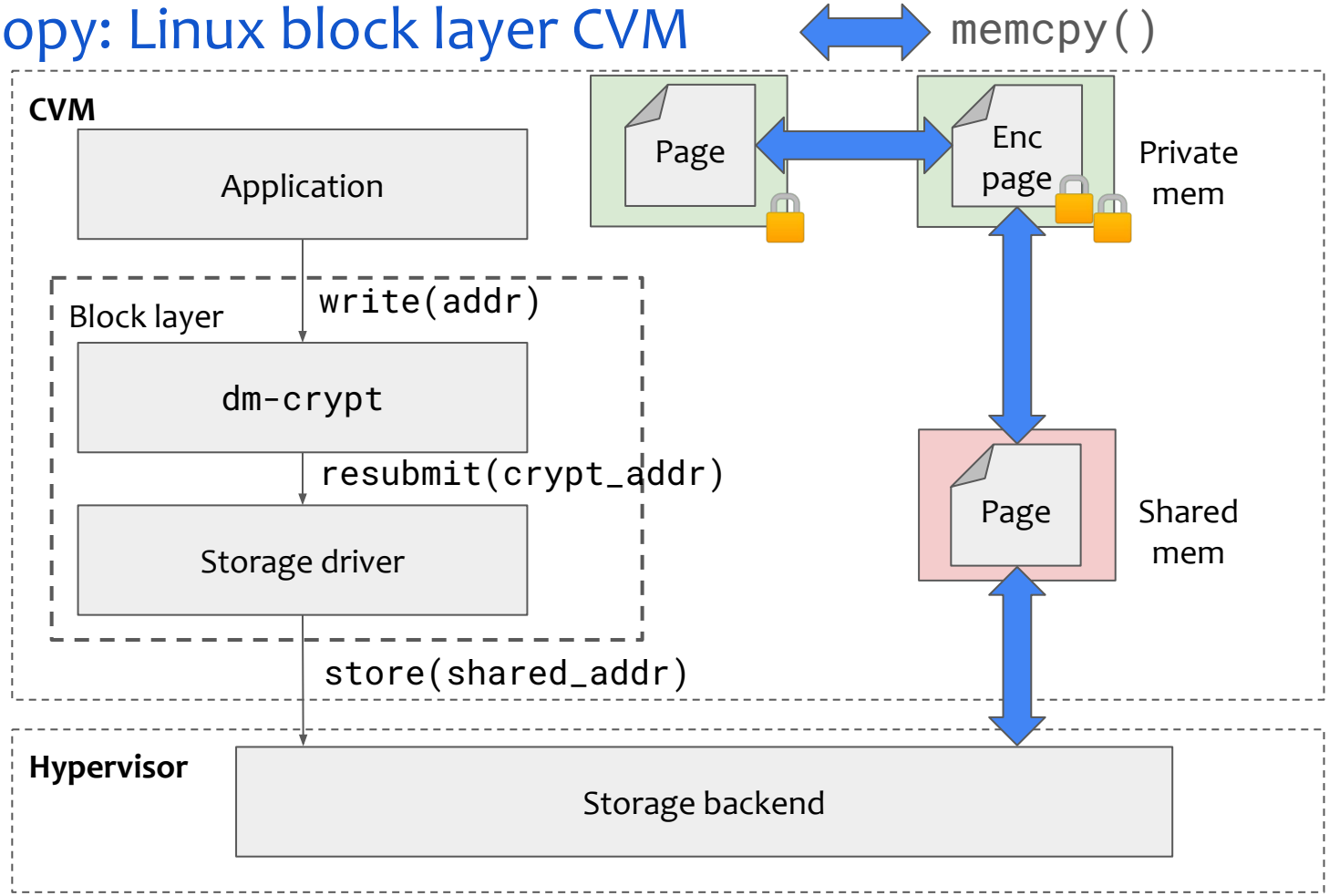
Zero-copy: Linux block layer



Zero-copy: Linux block layer CVM

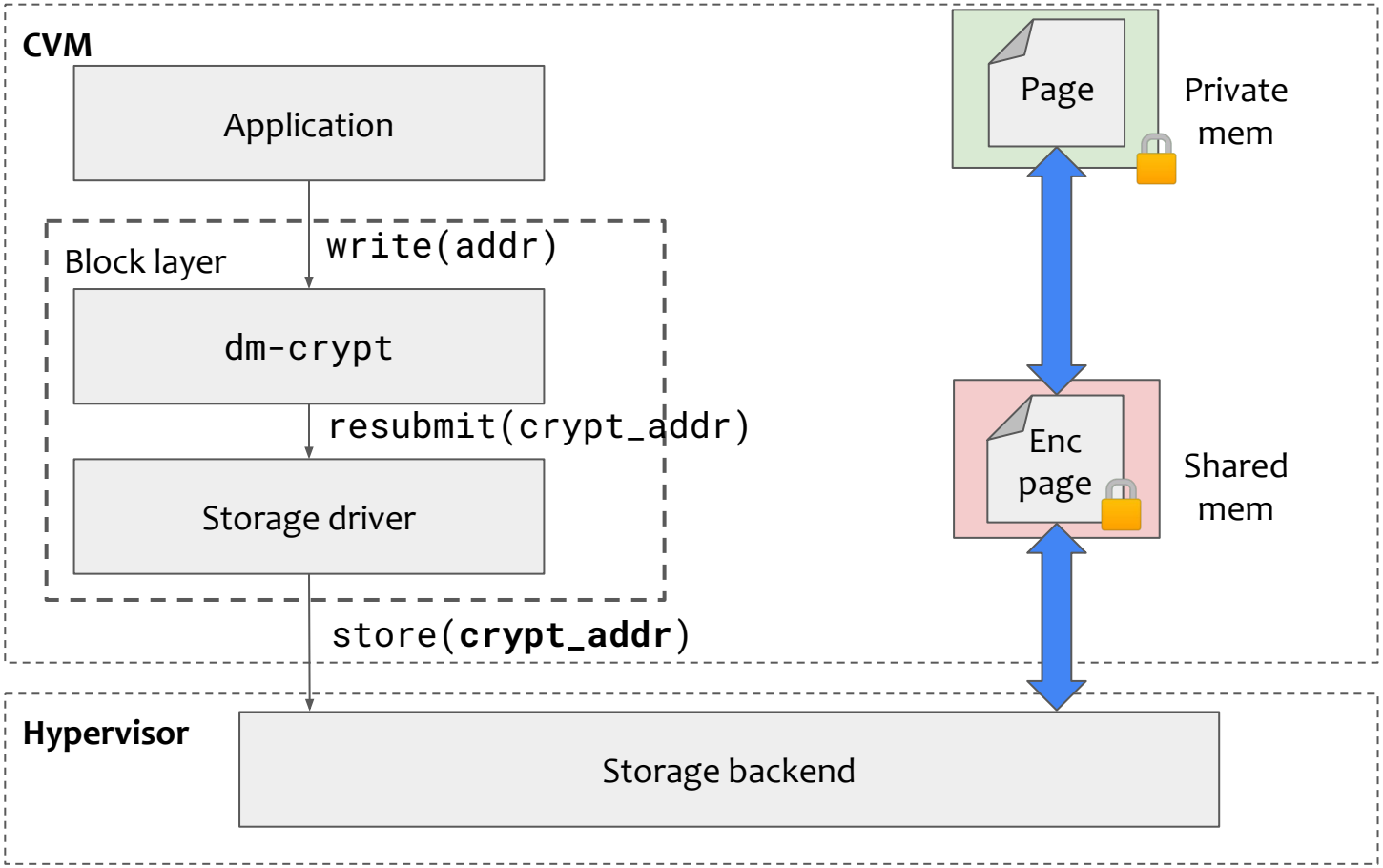


Zero-copy: Linux block layer CVM



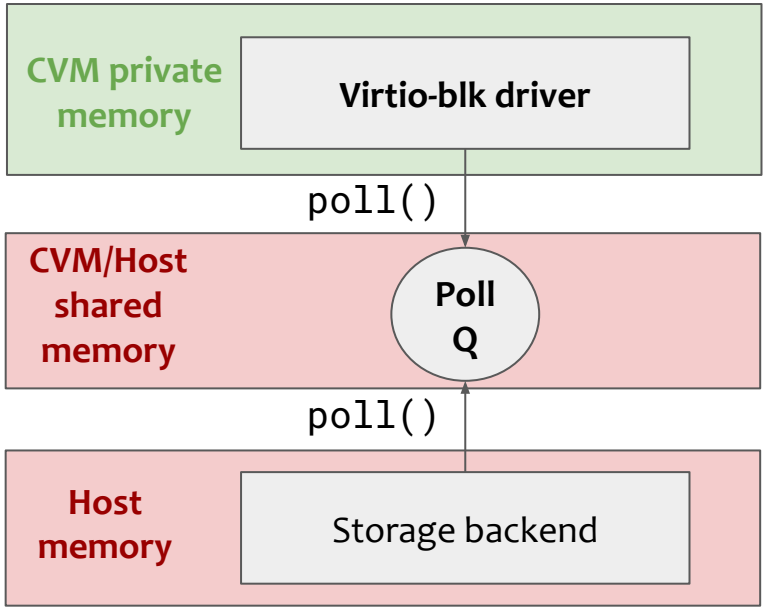
Zero-copy: Linux block layer cvm-io

memcpy()



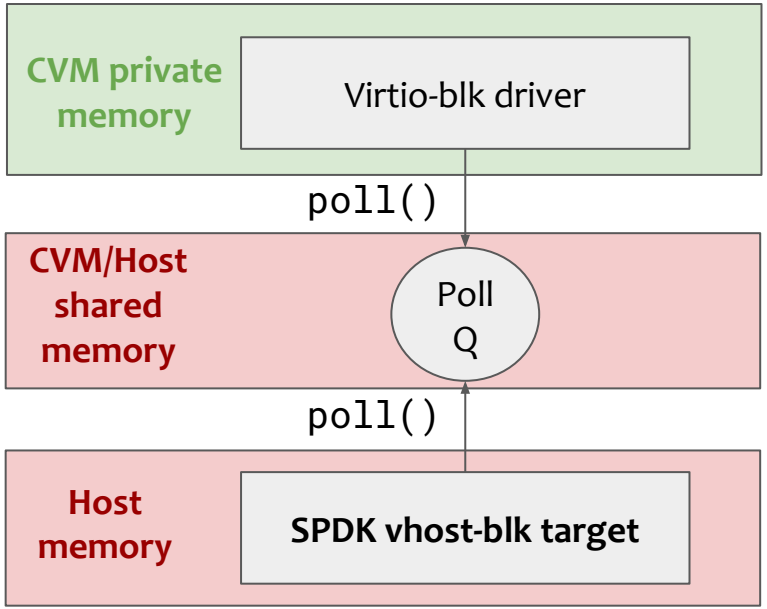
Polling

- Guest side: virtio-blk **poll-mode**
  - Designated poll-queues



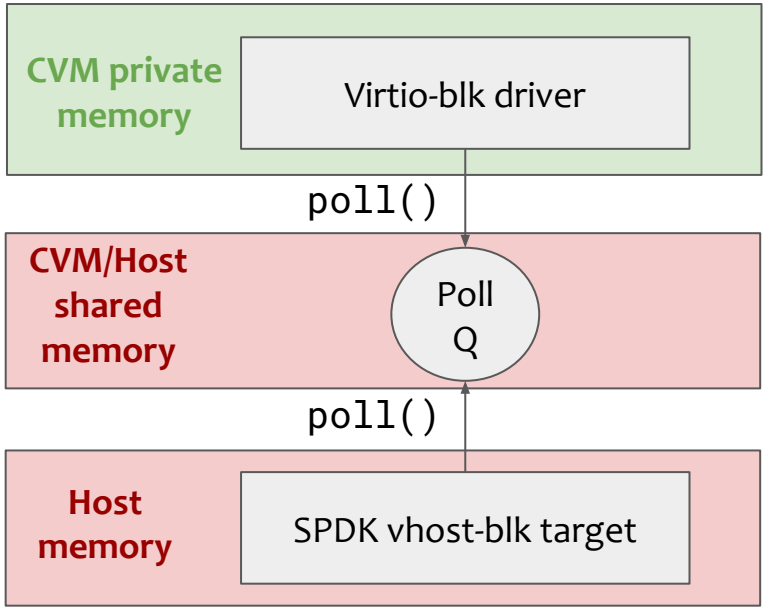
Polling

- Guest side: virtio-blk **poll-mode**
  - Designated poll-queues
- Host side: SPDK vhost-blk target
  - Userspace process w/ userspace SSD driver



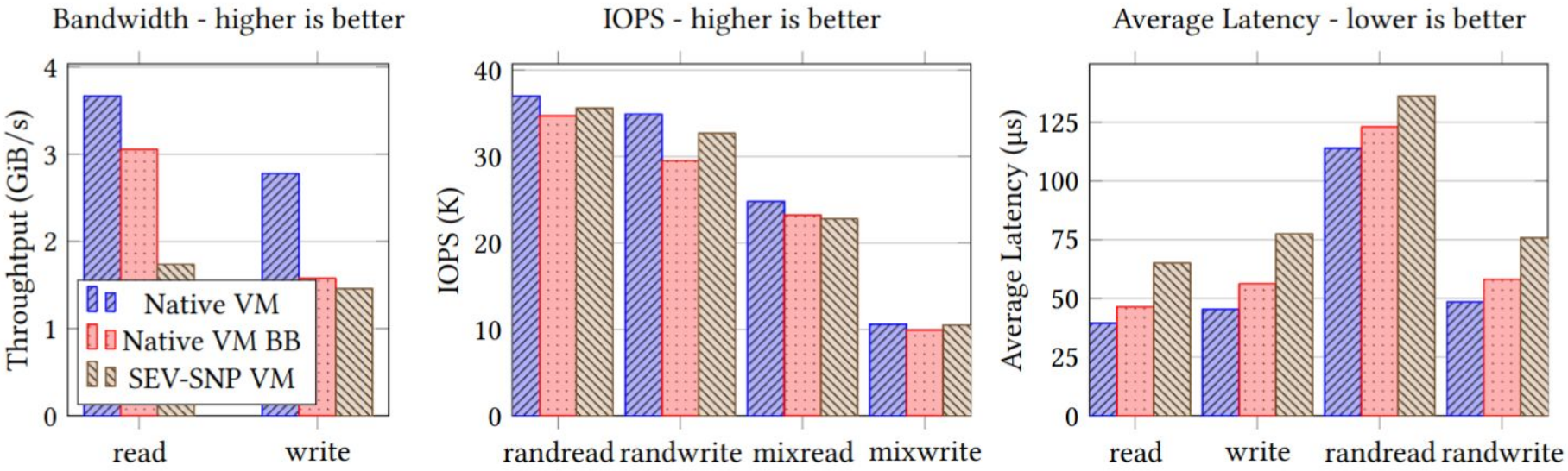
Polling

- Guest side: virtio-blk **poll-mode**
  - Designated poll-queues
- Host side: SPDK vhost-blk target
  - Userspace process w/ userspace SSD driver
- Transport protocol: virtio
  - Polling requires protocol negotiation
  - SPDK vhost-blk target doesn't support guest-side polling
    -> **future work**



Benchmarks

Benchmarks



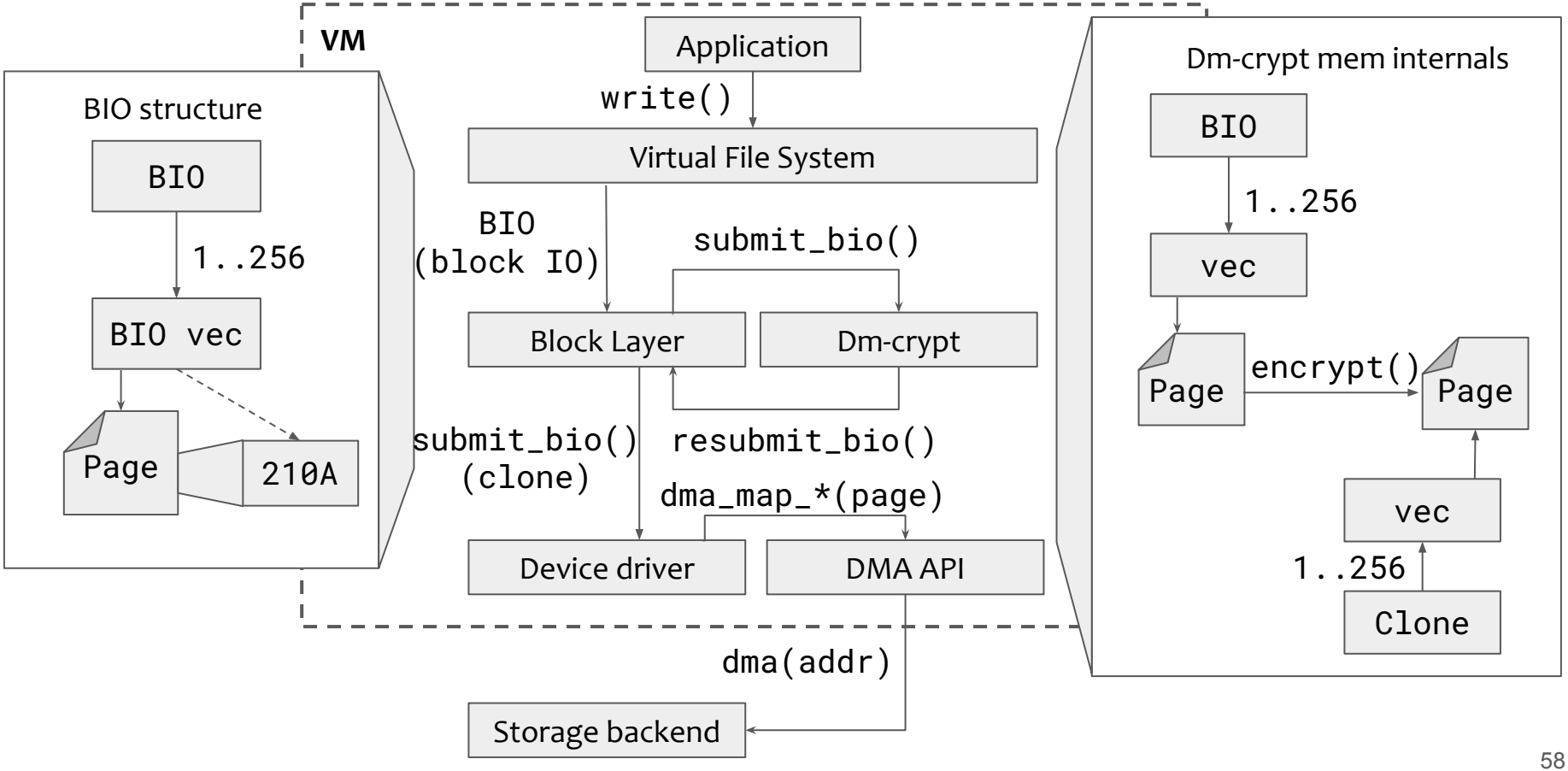
(a) Bandwith comparison. (b) IOPS comparison. (c) Average Latency comparison.

Figure 1: Native and confidential VM `virtio-blk` performance comparison.

In-depth CVM Linux Storage IO Stack

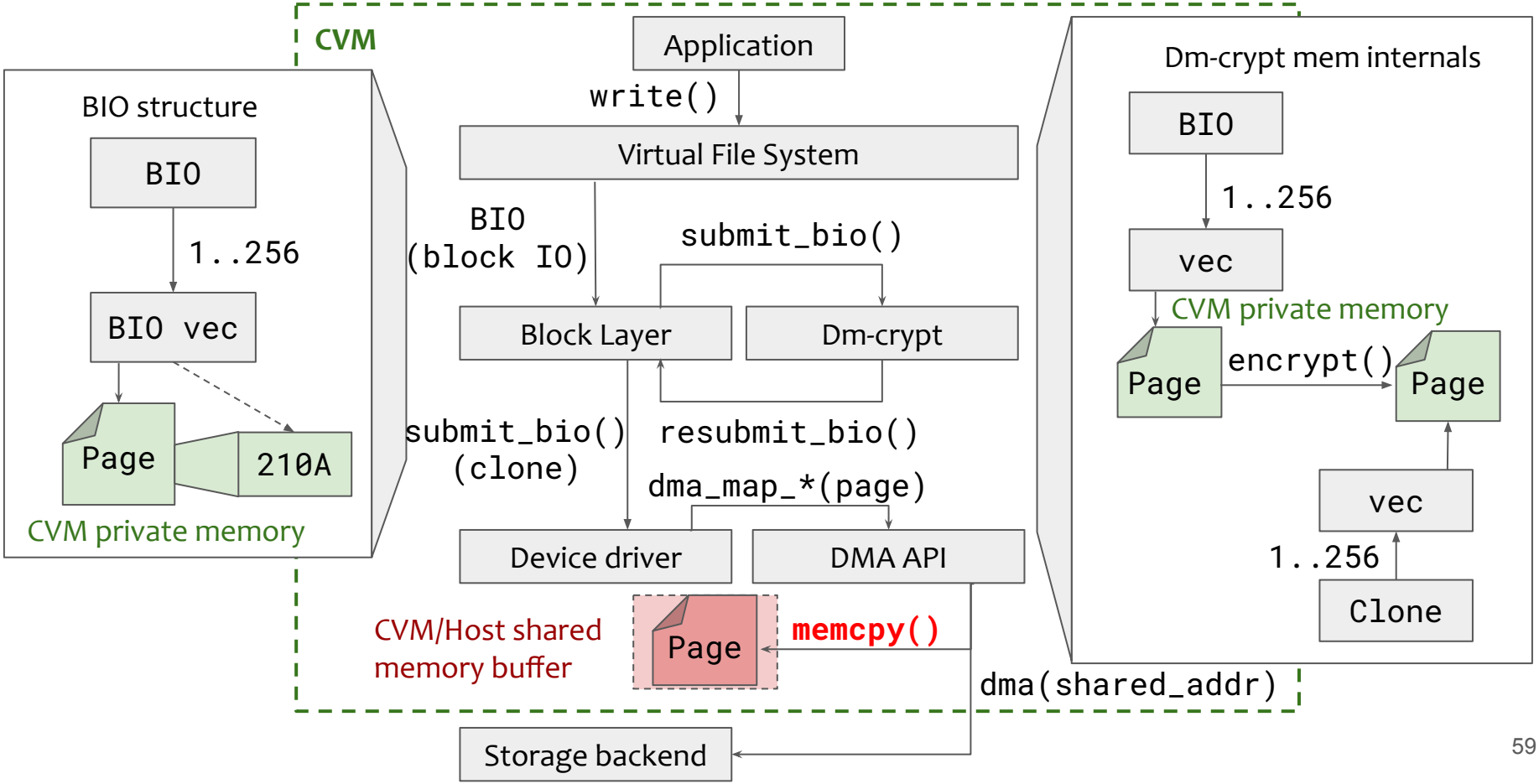
Native VM Storage Stack

Todo: animate



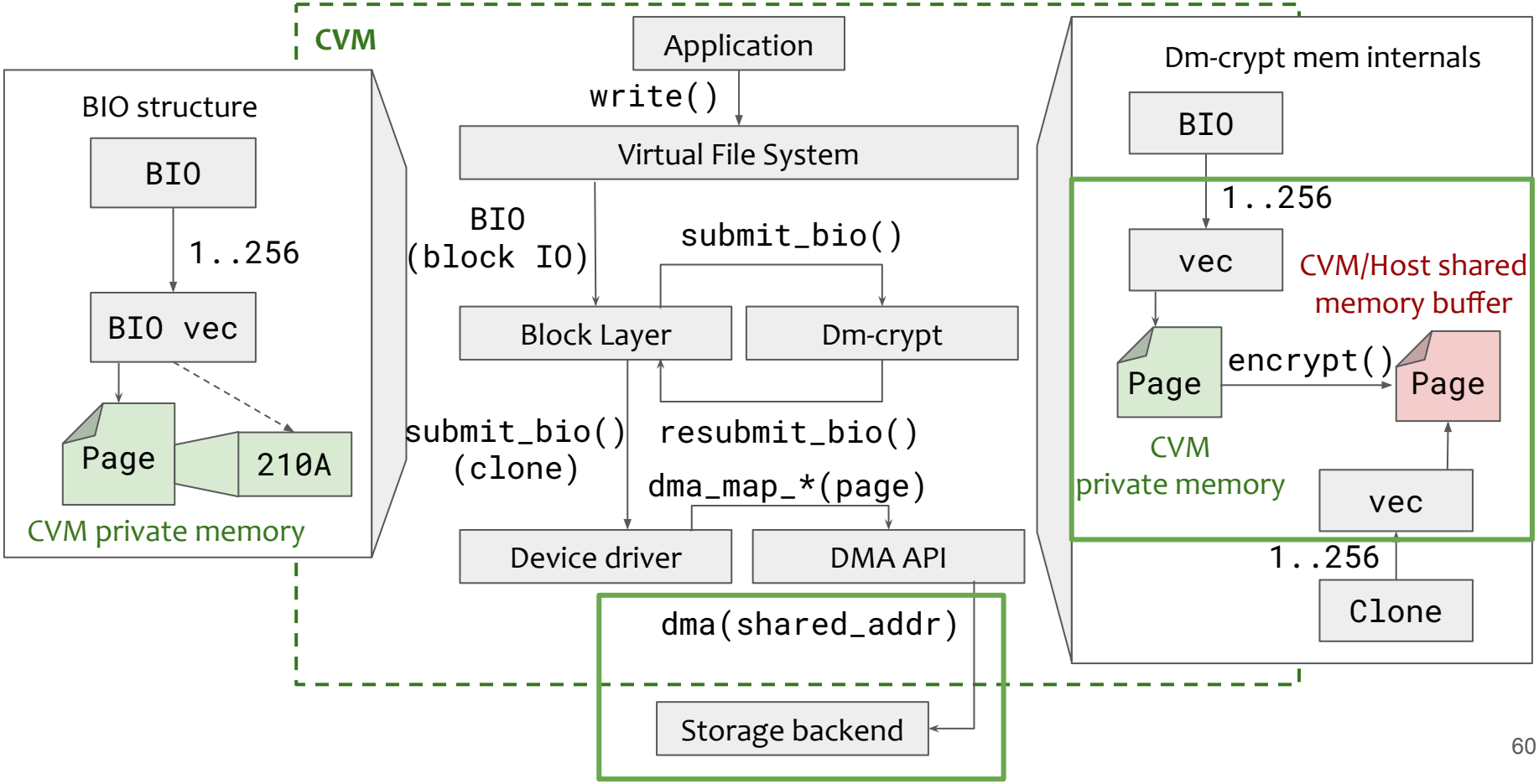
Native CVM Storage Stack

Todo: animate



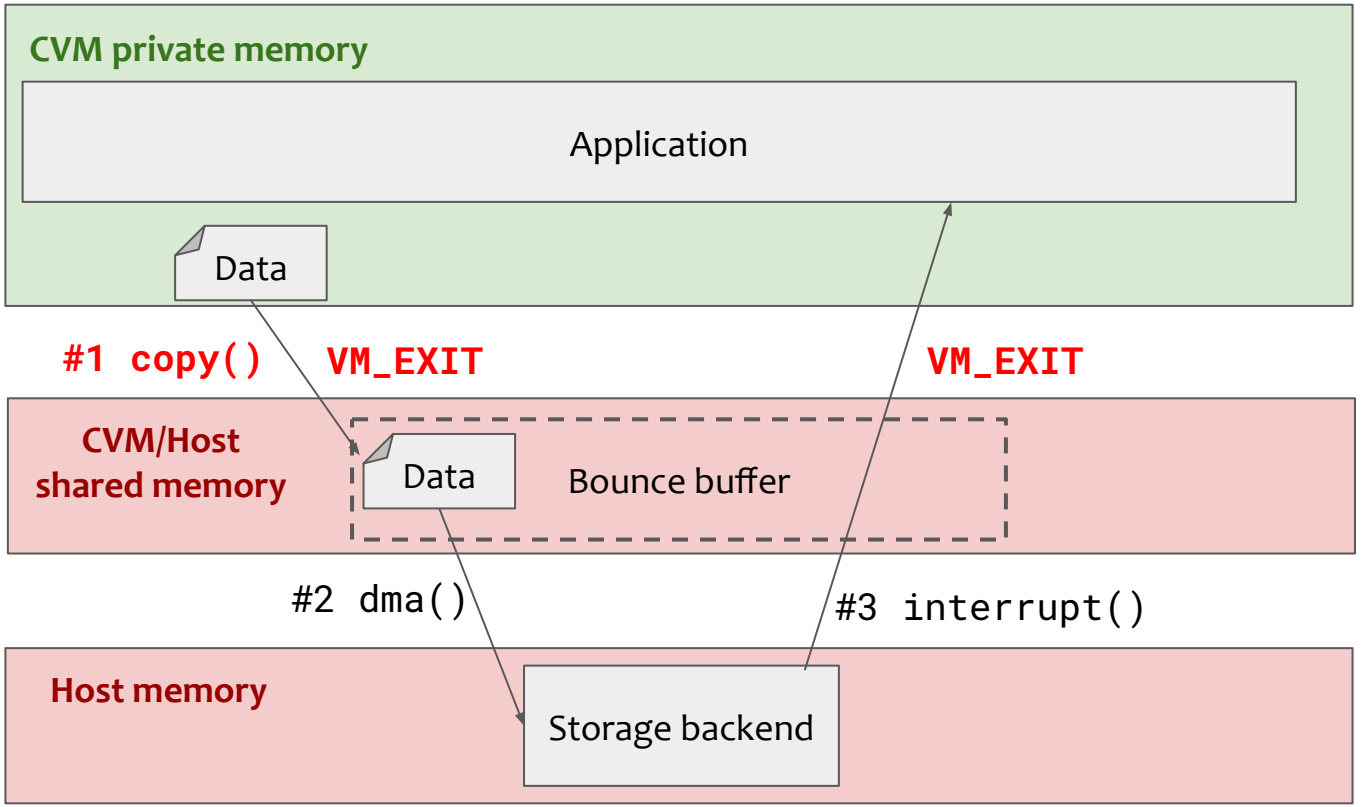
cvm-io Storage Stack

Todo: animate

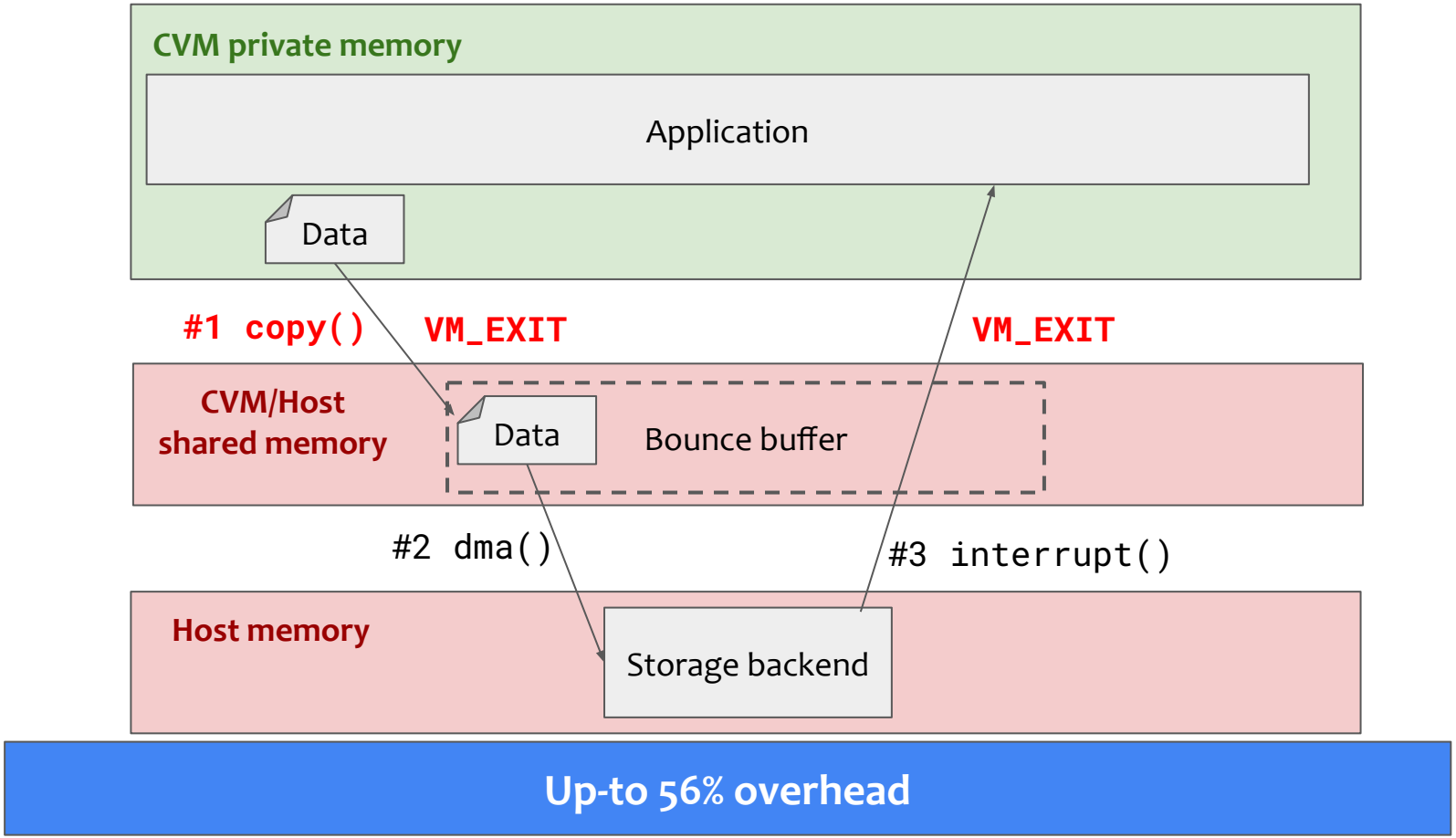


CVM-IO Flow Graphs

Storage I/O in CVMs: without trusting Host / Hypervisor



Storage I/O in CVMs: without trusting Host / Hypervisor



Storage Driver Polling Candidates

VM Polling Storage Drivers

Candidates:

- Virtio-blk: standard paravirtual VM driver
  - Interrupt-based; has polling mode
  - w/ vhost-user-target SPDK: kernel-bypass
- NVMe: same protocol as SSD
  - Polling-based; w/ vfio communicate directly w/ NVMe SSD
  - w/ SPDK: kernel-bypass
- Io_uring
  - Polling-based
  - Backend in kernel -> skip world-switch copy

VM Polling Storage Drivers

Candidates:

- Virtio-blk: standard paravirtual VM driver
  - Interrupt-based; has polling mode
  - w/ vhost-user-target SPDK: kernel-bypass
- NVMe: same protocol as SSD
  - Polling-based; w/ vfio communicate directly w/ NVMe SSD
  - w/ SPDK: kernel-bypass
- ~~Io_uring~~
  - ~~Polling-based~~
  - ~~Backend in kernel -> skip world-switch copy~~

Io_uring for VMs is WIP; currently, no VM isolation -> security issues!

VM Polling Storage Drivers

Candidates:

- Virtio-blk: standard paravirtual VM driver
  - Interrupt-based; has polling mode
  - w/ vhost-user-target SPDK: kernel-bypass
- ~~NVMe: same protocol as SSD~~
  - ~~Polling-based; w/ vfio communicate directly w/ NVMe SSD~~
  - ~~w/ SPDK: kernel-bypass~~
- ~~io_uring~~
  - ~~Polling-based~~
  - ~~Backend in kernel -> skip world-switch copy~~

1. **NVMe-vfio-pci: not in mainline Qemu -> not in CVM Qemu -> still WIP**
2. **Not recommended by Intel TDX hardening guide**

VM Polling Storage Drivers

Candidates:

- Virtio-blk: standard paravirtual VM driver
  - Interrupt-based; has polling mode
  - w/ vhost-user-target SPDK: kernel-bypass
- ~~NVMe: same protocol as SSD~~
  - ~~Polling-based; w/ vfio communicate directly w/ NVMe SSD~~
  - ~~w/ SPDK: kernel-bypass~~
- ~~io_uring~~
  - ~~Polling-based~~
  - ~~Backend in kernel -> skip world-switch copy~~

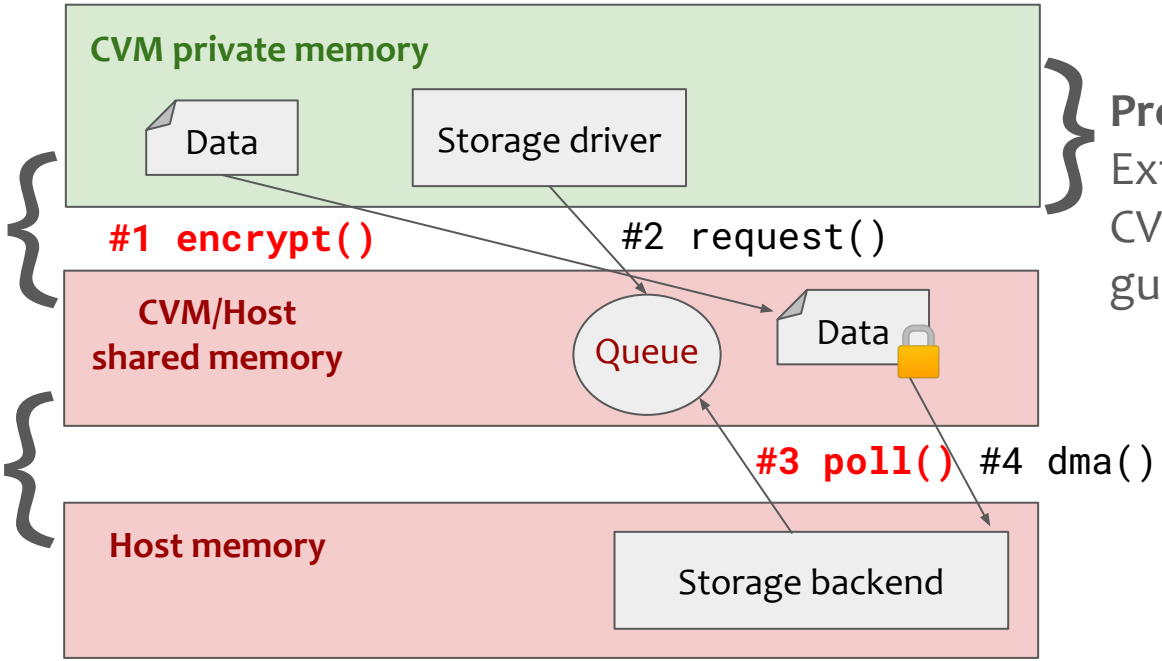
1. **Recommended by TDX hardening guide**
2. **Available in mainline Qemu**

Archive

cvm-io: A storage IO stack for CVMs

Zero-copy:
No extra copy into bounce buffer

Polling:
Avoid VM_EXITs by polling



Protection:
Extend CVM guarantees