



CoCo VM's Soft Lockups During High-Bandwidth Storage Reads

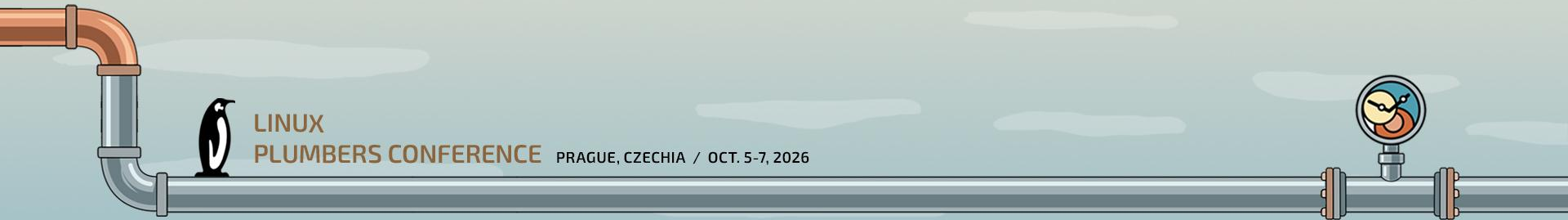
LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

By: Ryan Afranji (afranji@google.com)

Discussion Overview

1. SWIOTLB Background
2. Guest Soft Lockups
3. Mitigation
4. Proposed Solution



1) SWIOTLB Background



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



SWIOTLB Background



01

1. Key Access

Current CoCo architecture
limits encryption key access.

CPU can access key but IO
devices cannot.



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



SWIOTLB Background



01

1. Key Access

Current CoCo architecture limits encryption key access.

CPU can access key but IO devices cannot.



02

2. Bounce Buffer

Confidential VM's IO devices still use traditional DMA.

Only can access unencrypted SWIOTLB bounce buffer.



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



SWIOTLB Background



01

1. Key Access

Current CoCo architecture limits encryption key access.

CPU can access key but IO devices cannot.



02

2. Bounce Buffer

Confidential VM's IO devices still use traditional DMA.

Only can access unencrypted SWIOTLB bounce buffer.



03

3. Memory Copy

Data transferred between the unencrypted SWIOTLB buffer and encrypted memory must be CPU-copied.



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



SWIOTLB Background



03

3. Memory Copy

Data transferred between the unencrypted SWIOTLB buffer and encrypted memory must be CPU-copied.



IRQ Context Bottleneck

Expensive CPU memory copy happens within IRQ context (for storage IO reads)

CODE CONTEXT

```
static irqreturn_t nvme_irq(int irq, void *data)
[...]
    nvme_pci_complete_batch(&iob);
```



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



SWIOTLB Background

NVMe Interrupt Handler

Broken up into distinct parts:

CODE CONTEXT

```
static irqreturn_t nvme_irq(int irq, void *data)
[...]  
    if (nvme_poll_cq(nvmeq, &iob)) {  
        [...]  
        nvme_pci_complete_batch(&iob);
```



1. Gather IO Batch

Gathers all DMA completed entries from completion queue (cq) into an IO batch (iob).



2. SWIOTLB Memory Copy

Completes IO batches' final steps (including SWIOTLB bounce).



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



2) Guest Soft Lockups



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Guest Soft Lockups

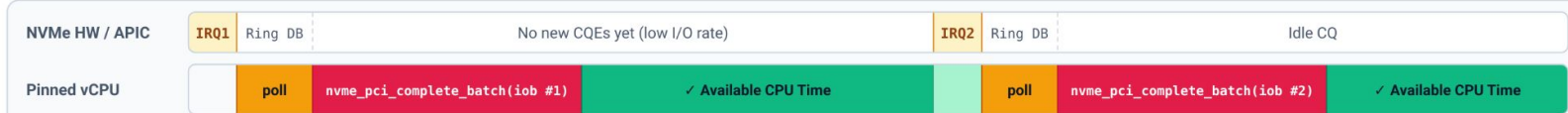
NVMe Hard IRQ & SWIOTLB Bounce Timeline (Pinned vCPU)

Comparison of CPU execution contexts and MSI-X interrupt re-arming across workloads

 `nvme_poll_cq()` + ring CQ doorbell  `nvme_pci_complete_batch()` (SWIOTLB memcpys)  Available CPU Time (free to schedule tasks)  Latched MSI-X IRQ waiting on `iret`

1) Light NVMe Read I/O Workload – Short SWIOTLB Bounces → Available CPU Time for Other Tasks

CPU free to schedule tasks (watchdog fed)



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Guest Soft Lockups

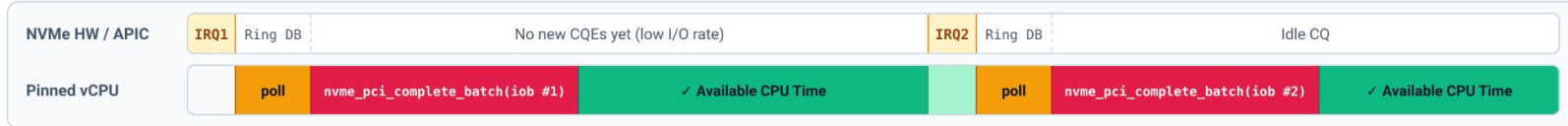
NVMe Hard IRQ & SWIOTLB Bounce Timeline (Pinned vCPU)

Comparison of CPU execution contexts and MSI-X interrupt re-arming across workloads

 `nvme_poll_cq()` + ring CQ doorbell  `nvme_pci_complete_batch()` (SWIOTLB memcpys)  Available CPU Time (free to schedule tasks)  Latched MSI-X IRQ waiting on `iret`

1) Light NVMe Read I/O Workload – Short SWIOTLB Bounces → Available CPU Time for Other Tasks

CPU free to schedule tasks (watchdog fed)



2) Heavy NVMe Read I/O Workload – IRQ Latches During SWIOTLB Bounces → 0% Available CPU Time (Softlockup)

100% HardIRQ → BUG: soft lockup



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



3) Mitigation



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Pinned IRQ

01



1. Queue Vectoring

Each storage I/O queue is assigned a dedicated interrupt vector.

1 Queue = 1 Vector

02



2. CPU Pinning

This interrupt vector gets pinned to a single/dedicated CPU to avoid lock contention.

1 Vector → 1 Dedicated CPU

03



3. Resource Impact

E.g. A system with 4 NVMe I/O queues will have dedicated 4 CPUs for SWIOTLB bounces.

4 Queues = 4 CPUs Consumed



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Mitigation

1 Storage IO Queue Count
4 → **16** queues

2 Per-Queue Entries
512 → **128** entries

KEY IMPACT

4x CPU Distribution

The same total entries are now evenly distributed across 4x CPUs (equating to 16 total CPUs).

✓ **Eliminated soft lockups!**



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Mitigation

1 Storage IO Queue Count
4 → 16 queues

2 Per-Queue Entries
512 → 128 entries

NOT A SOLUTION

Eliminated soft lockups... but is only a mitigation, not a permanent solution!

1. Expensive SWIOTLB bounces **still occur within the storage interface's IRQ** context.
2. Limited number of HW storage queues.
3. **Will soft lockup eventually** once storage speeds increase even more.



4) Proposed Solution



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Proposed Solution

ORIGINAL

```
static irqreturn_t nvme_irq(int irq, void *data)
[...]  
    nvme_pci_complete_batch(&iob);
```

IRQ Execution

SWIOTLB bounce buffers are processed synchronously inside the NVME interrupt handler.



PROPOSED

```
static irqreturn_t nvme_irq(int irq, void *data)
[...]  
    if (force_bounce_swiotlb_wq)  
        nvme_pci_complete_batch_deferred(iob);  
    else  
        nvme_pci_complete_batch(&iob);
```

Deferred Workqueue Processing

- 1) Offloads heavy bounce copying out of hard IRQ context **eliminating soft lockups!**
- 2) Workqueue is unbound allowing bounce to be deferred to **any available CPU**.

Patch Link:

<https://lore.kernel.org/all/20261006143329.2415252-1-afranji@google.com/>



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Performance Results & Key Questions

Benchmark (40 runs)

4 vCPUs (2 Queues)

Metric	p5	p50
IOPS	+7.59%	+4.05%
Bandwidth	+1.53%	+1.18%
Latency (clat p50)	+10.41%	+3.88%
Latency (clat p90)	-1.74%	-5.51%
Latency (clat p99.9)	-16.44%	-9.86%

176 vCPUs (4 Queues)

Metric	p5	p50
IOPS	+90.32%	+93.32%
Bandwidth	+9.57%	+10.68%
Latency (clat p50)	+9.58%	+8.70%
Latency (clat p90)	+9.91%	+7.51%
Latency (clat p99.9)	+8.00%	-4.58%

* Green = performance improvement | Red = regression

Remaining Questions

- 1 Are these acceptable performance regressions?
- 2 Should this feature be conditionally enabled?
(e.g. module parameter)
- 3 Is there a better approach?



Q & A



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

