ByteDance 字节跳动

KVM-devirt

Extending KVM to a Zero-overhead Partition Hypervisor

Liang Deng
System Technologies and Engineering (STE) team

KVM forum 2022

Motivation

ByteDance 字节跳动

Trends: the core number of a single server increases rapidly

Genoa： 384 cores

Icelake： 224 cores

Problems

Scalability: Linux kernel encounters many-core scalability bottlenecks, e.g., lock contentions in file system, network stack, scheduler, memory management and so on.

Fault isolation: More applications run on a single server. If one application crashes kernel, the whole sever crashes.

Kernel customization: a single kernel is hard to fulfill applications' custom requirements (e.g., kernel configuration, kernel boot parameters)

Partition the server using current KVM Virtualization

Separate guest kernels

Virtualization overhead is non-trivial

KVM Virtualization Overhead

ByteDance 字节跳动

1. VM exits

- Timer virtualization
- IPI virtualization
- virtio notifications
- HLT and MWAIT instructions
- CPUID instruction
- Host interrupt

2. Posted interrupt

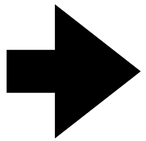
- Although VM exit can be eliminated
- Overhead is still non-trivial due to complex hardware path

3. Additional address translations

- Stage-2 address translations (EPT or NPT)
- DMA remap translation on IOMMU page tables

KVM-devirt

- Interrupt passthrough
- IPI passthrough
- Timer passthrough
- Memory de-virtualization
- DMA de-virtualization
- Virtio notification passthrough

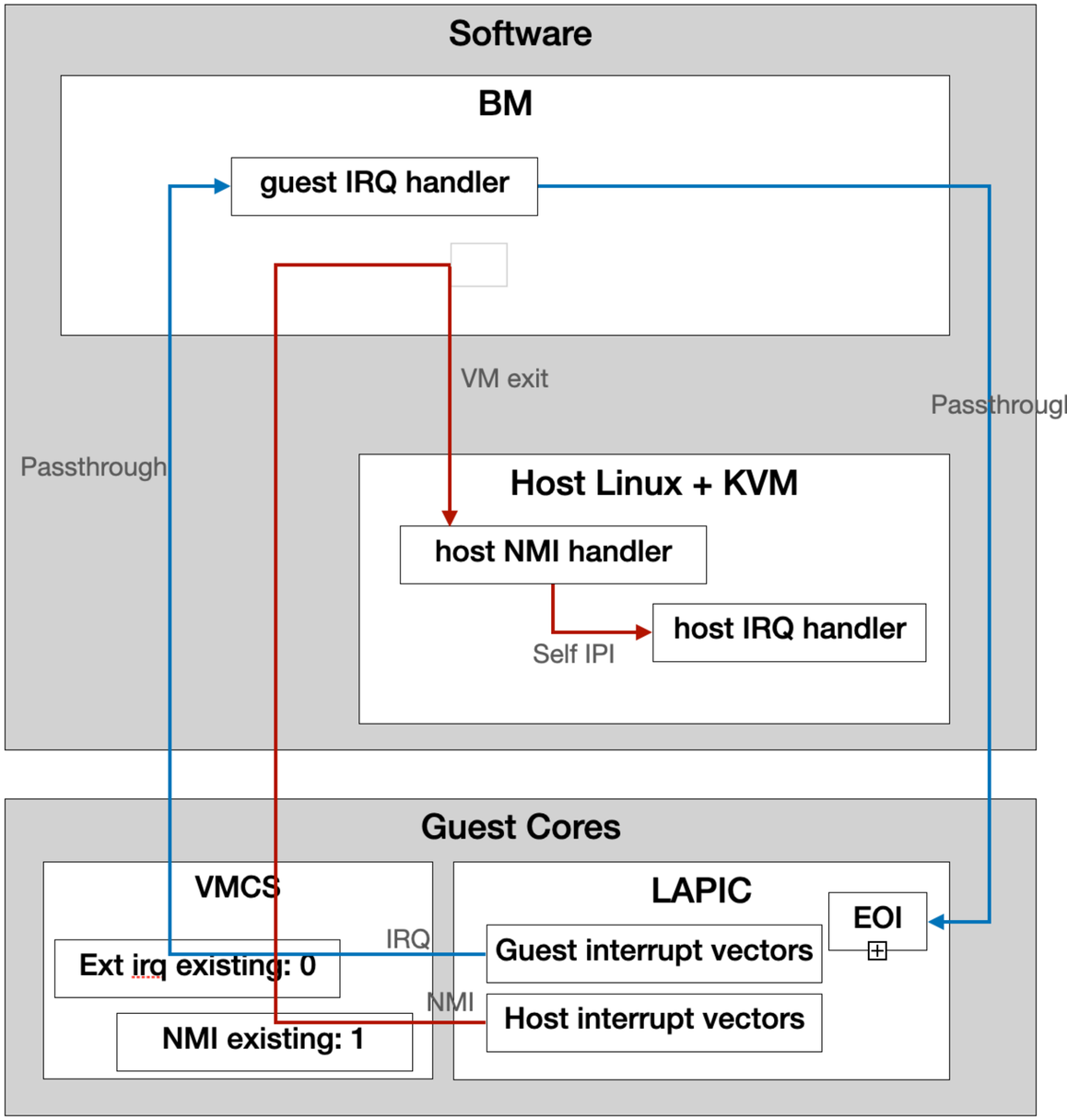


- Remove all VM exits after guest kernel initialization phase
- Remove all additional address translations
- Support both Intel and AMD platforms

Interrupt Passthrough

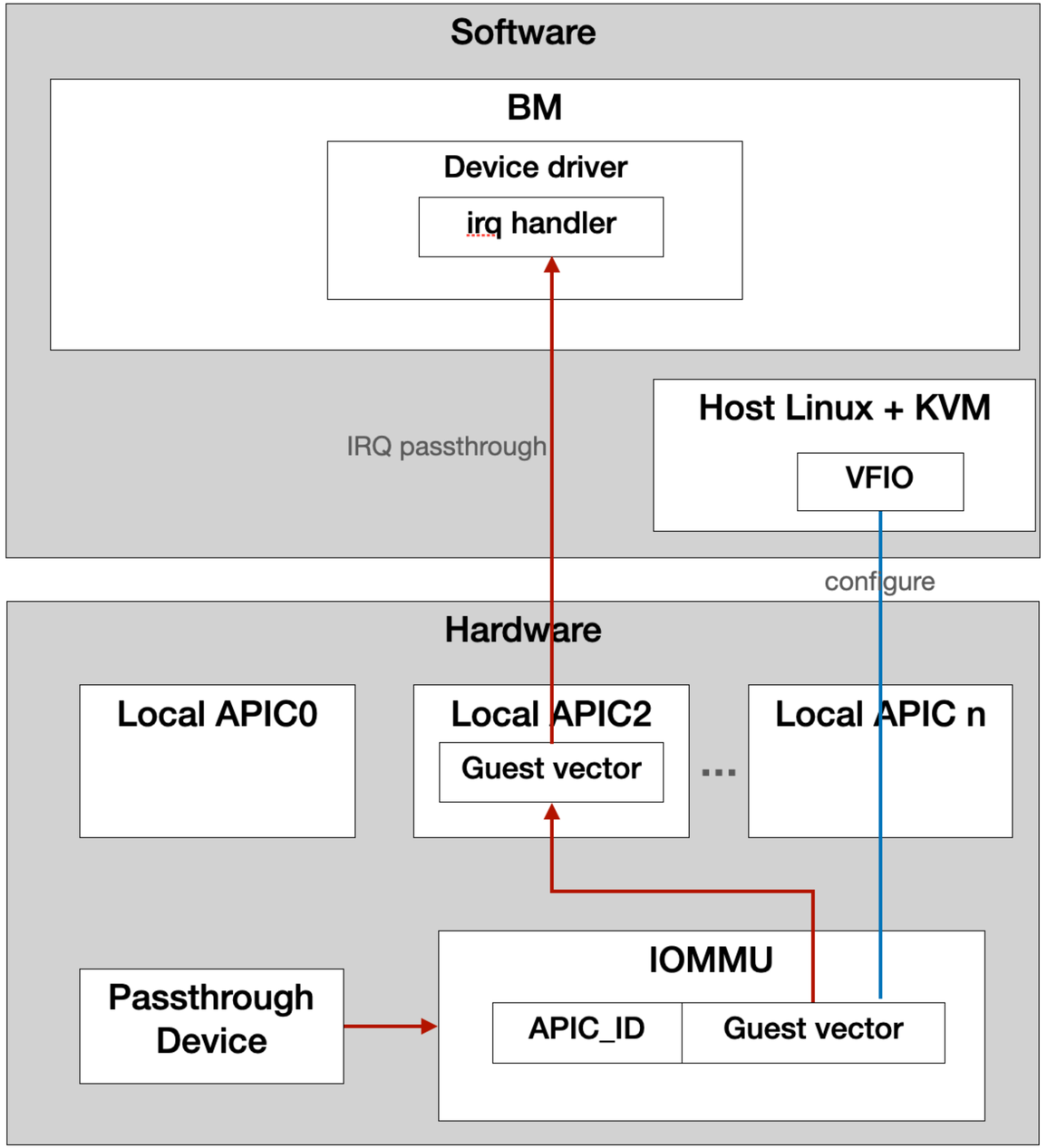
ByteDance 字节跳动

- Posted interrupt is not used due to its extra overhead in hardware path
- Pass local APIC registers (IRR, ISR, EOI) directly to the BM. Use separate host and guest interrupt vectors to avoid mixture.
- Guest interrupts arrived on guest cores are configured as IRQs which are directly delivered to non-root mode.
- Host interrupts arrived on guest cores are configured as NMIs which cause VM exits.
- Re-trigger a self-IPI (IRQ) in host NMI handler to solve IRQ-mask issue.
- Send self-IPIs at VM entry to inject virtual guest interrupts of emulated devices



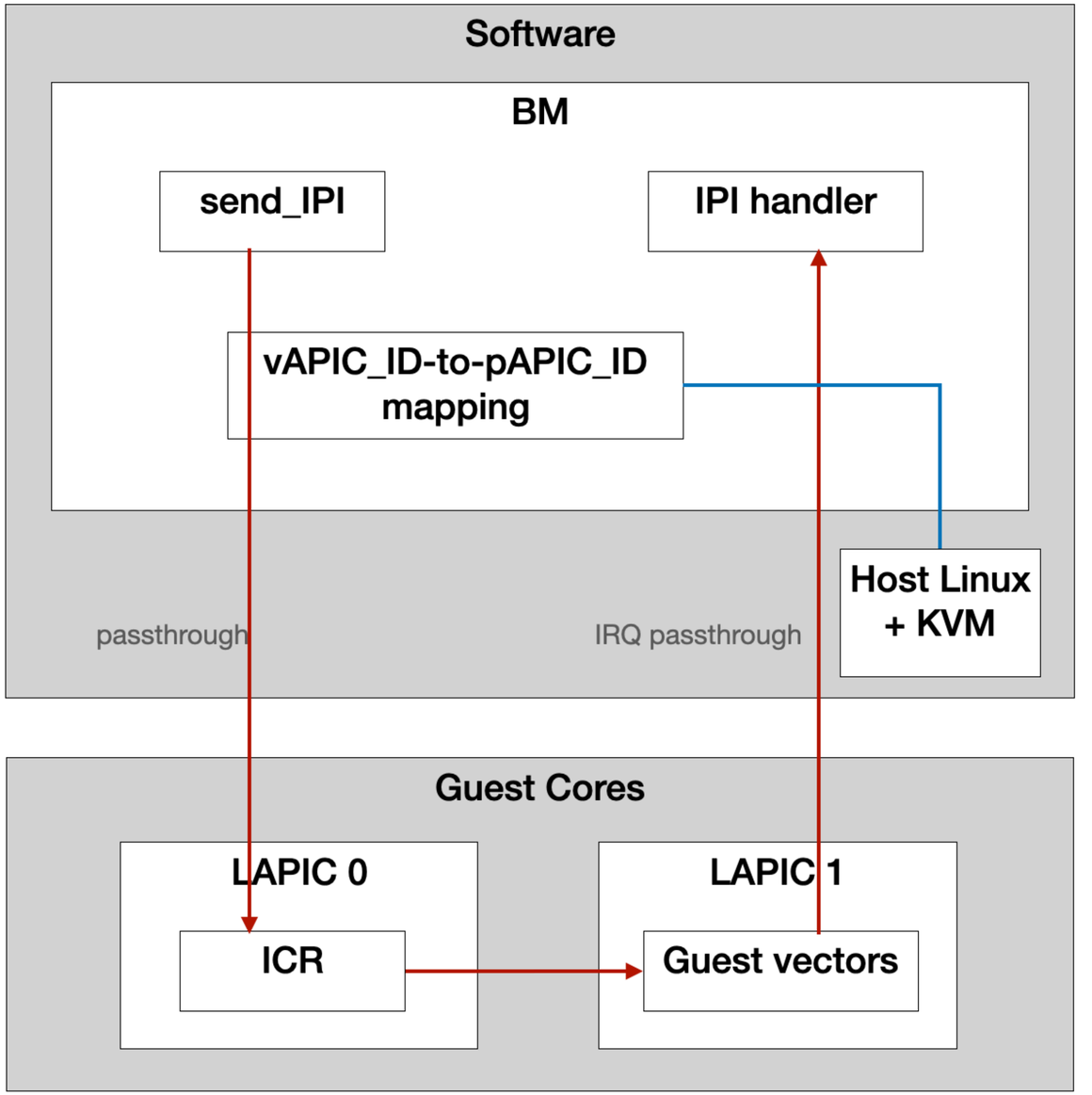
Interrupt Remap

- Interrupt posting capability is not used in IOMMU interrupt remapping.
- VFIO fills in the IRTE with guest vector of guest device interrupt and APIC_ID of the physical core where BM runs.
- When BM changes the virq-vcpu binding relations or guest vectors, VFIO updates the IRTE with the new value.



IPI Passthrough

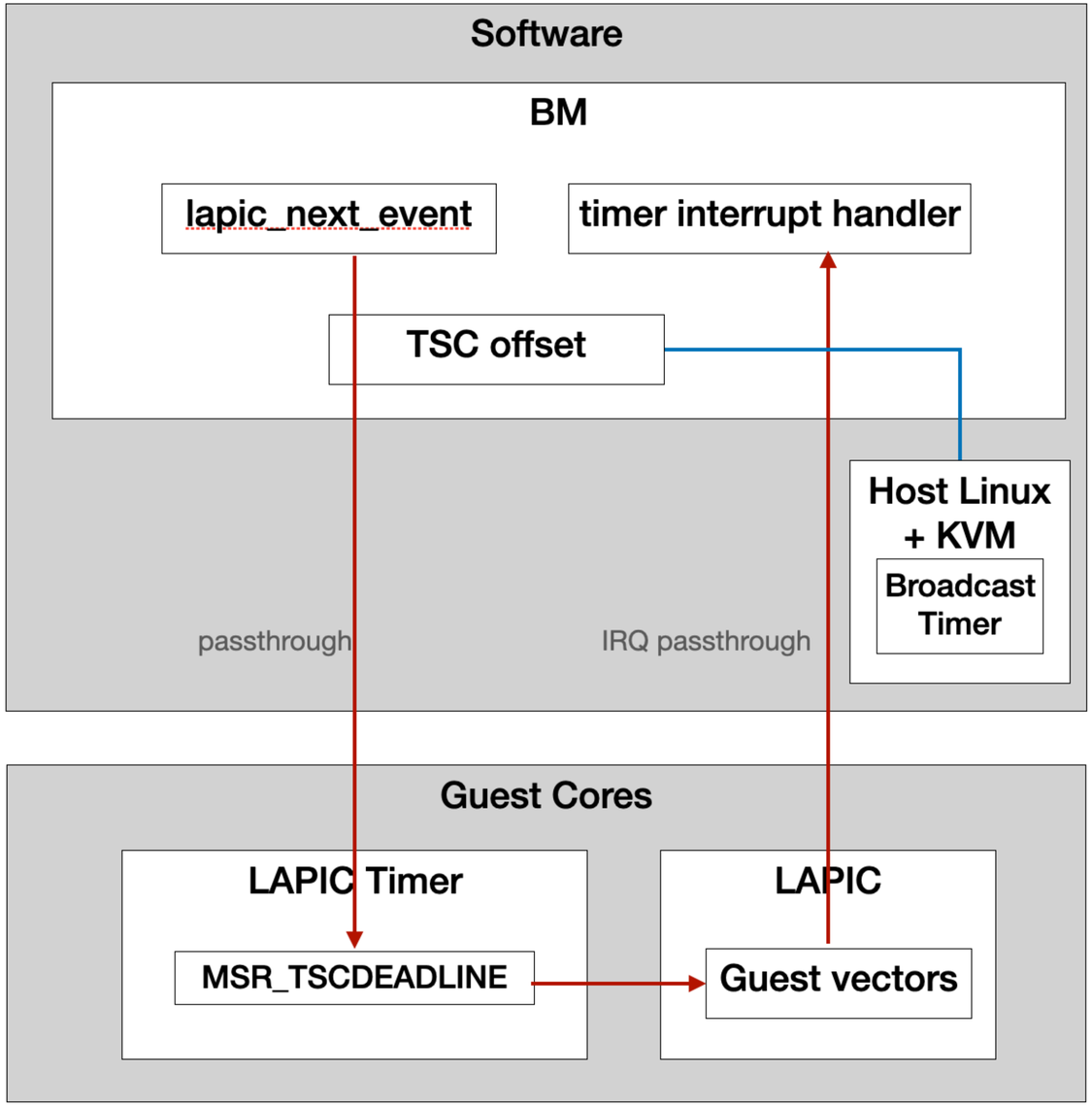
- KVM maps a vAPIC_ID-to-pAPIC_ID mapping into BM at BM startup and updates it whenever a VCPU thread is migrated to a new physical core.
- At the sending core, BM maps vAPIC_ID of target VCPU to pAPIC_ID, and directly accesses ICR to send IPI with guest vector.
- At the receiving core, IPI (IRQ) is directly delivered to BM without VM exits.



Timer Passthrough

ByteDance 字节跳动

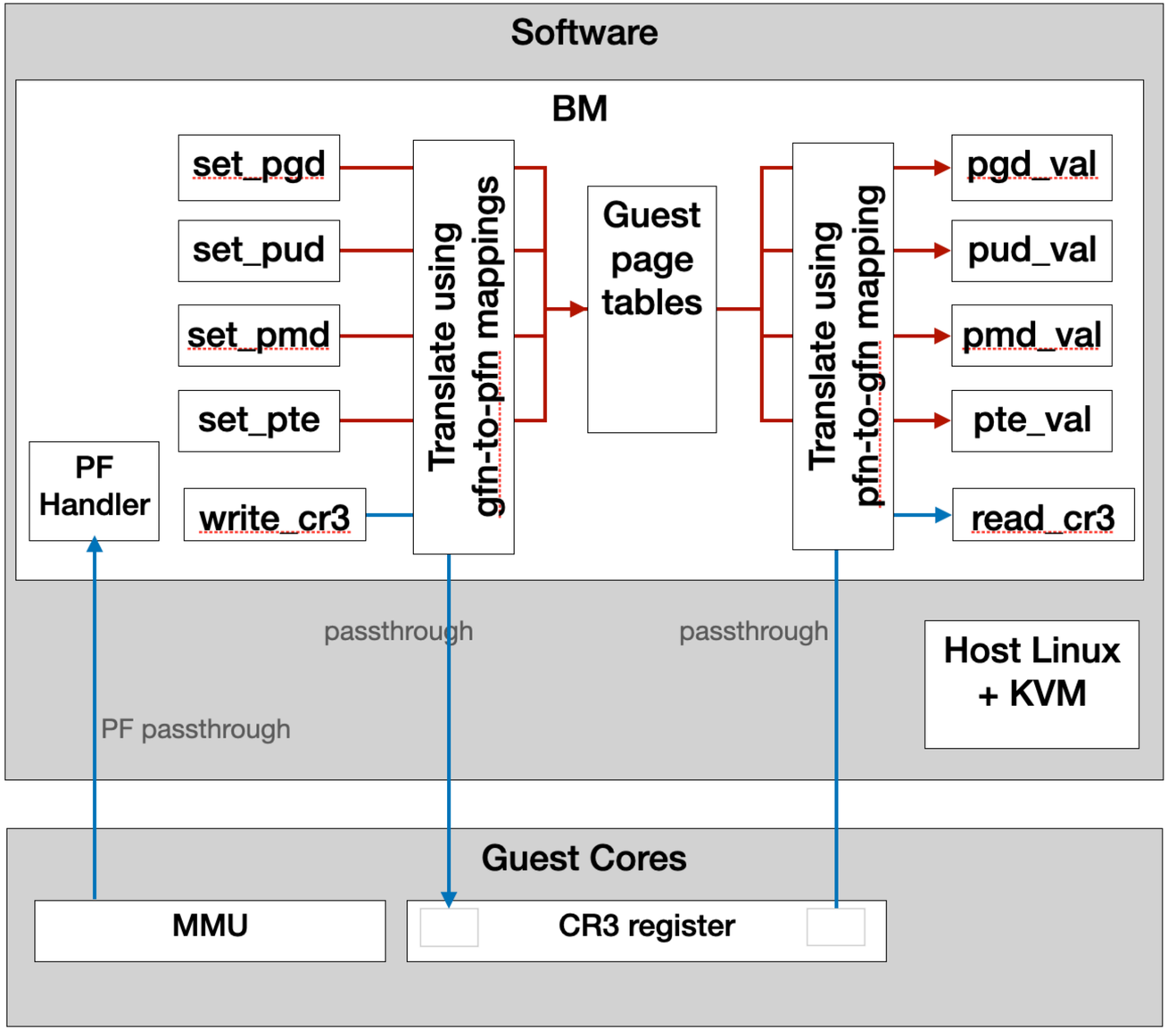
- BM uses physical Local APIC timer and Host Linux uses broadcast timer.
- KVM maps the TSC offset value into BM and updates it whenever modified.
- At lapic_next_event, BM subtract TSC offset value from guest TSC deadline and set it to MSR_TSCDEADLINE
- On LAPIC timer expiration, timer interrupt (IRQ) is directly delivered to BM without VM exits.



Memory De-virtualization

ByteDance 字节跳动

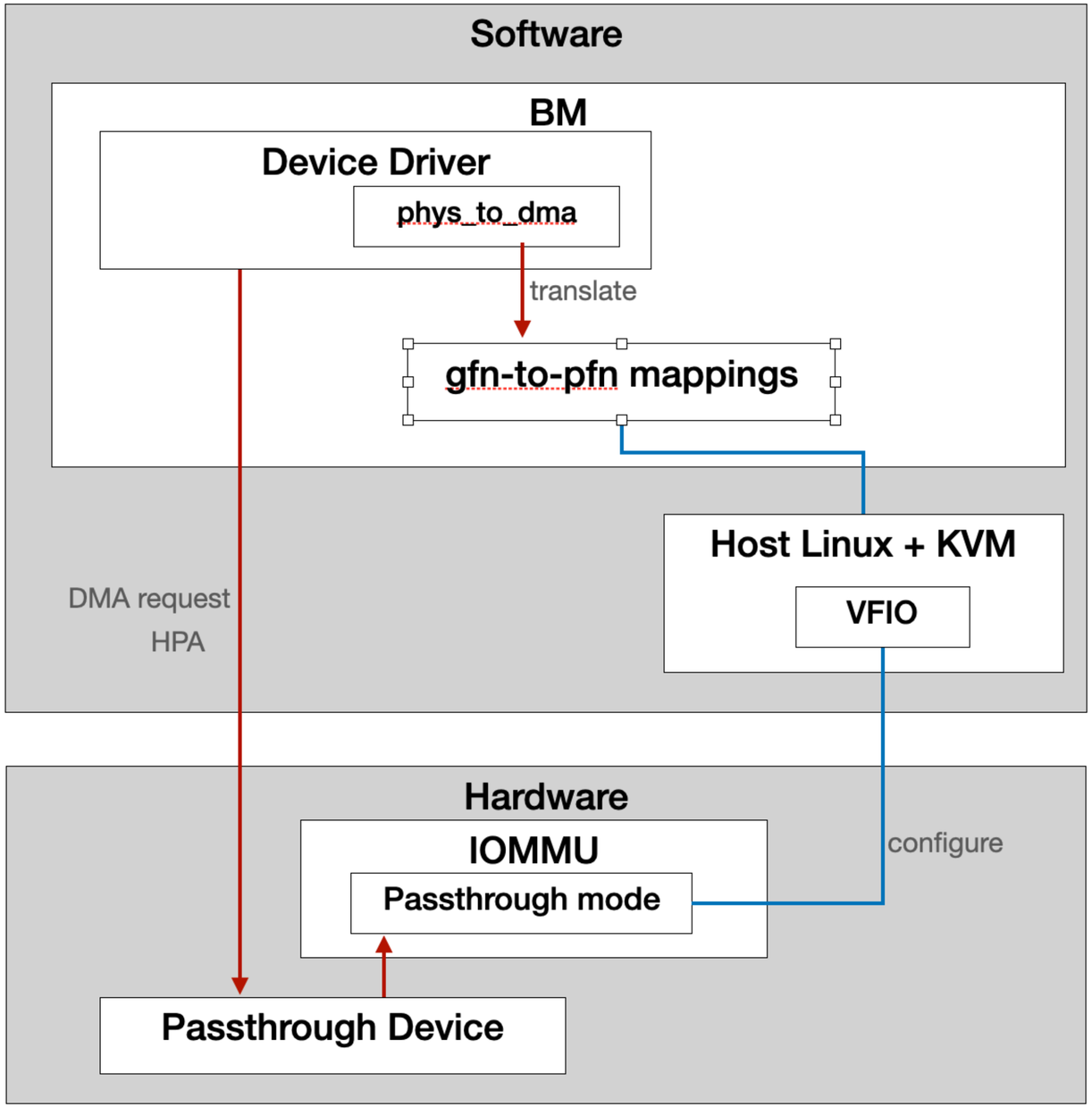
- BM only uses stage-1 page table, stage-2 (EPT or NPT) is disabled.
- At BM startup, KVM statically pins BM's guest memory, initializes both gfn-to-pfn and pfn-to-gfn tables and maps them into BM.
- When BM writes its own guest page tables (with set_pgd/set_pud/set_pmd/set_pte PV interfaces), it translates the gfn into pfn. Thus guest page tables directly use pfns.
- The hardware MMU directly uses BM's guest page tables
- When BM reads guest page tables (with pgd_val/ pud_val/pmd_val/pte_val PV interfaces), it translates the pfn into gfn.
- Use a hypercall in guest page fault handler to emulate a MMIO trap.



DMA De-virtualization

ByteDance 字节跳动

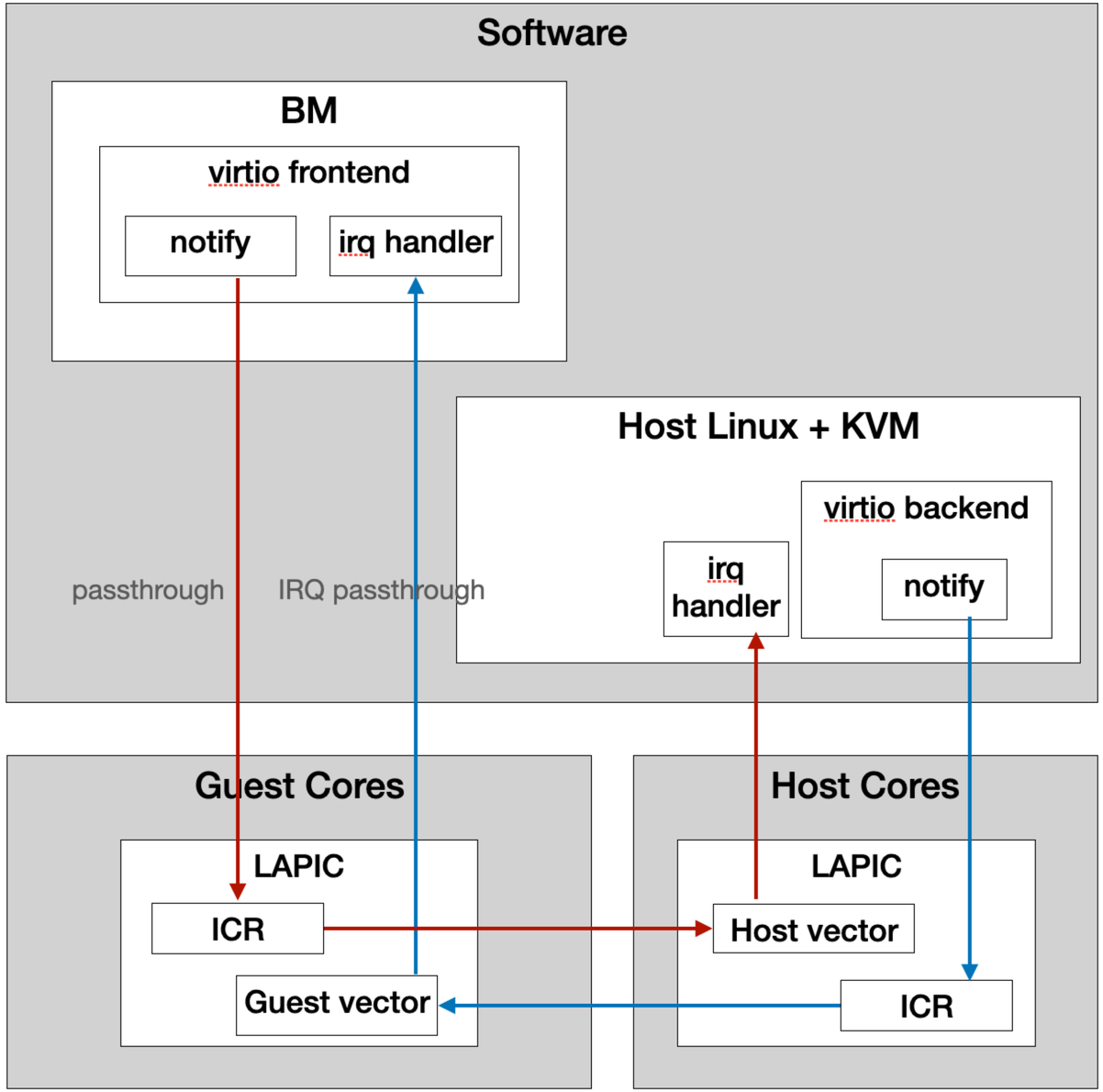
- At BM startup, KVM statically pins BM's guest memory and maps both gfn-to-pfn and pfn-to-gfn mappings into BM.
- When the passthrough device driver invokes dma_map to map dma buffer before issuing a dma request, it first relies on the gfn-to-pfn mappings to translate the gpa in the request to hpa. Thus DMA remap is not required.
- The VFIO in host configures the IOMMU as passthrough mode to disable DMA remap address translations.
- Additional modifications to the device driver is required to ensure that the dma_map is invoked as PAGE_SIZE granularity.



Virtio Notification Passthrough

ByteDance 字节跳动

- When virtio frontend in BM sends notification to backend, it directly accesses the ICR to send an IPI (with host vector) to the host core, without any VM exit.
- When backend sends notification to frontend in BM, it sends IPI (with guest vector) to the guest core. The IPI (configured as IRQ) is then directly delivered to BM based on the interrupt passthrough.



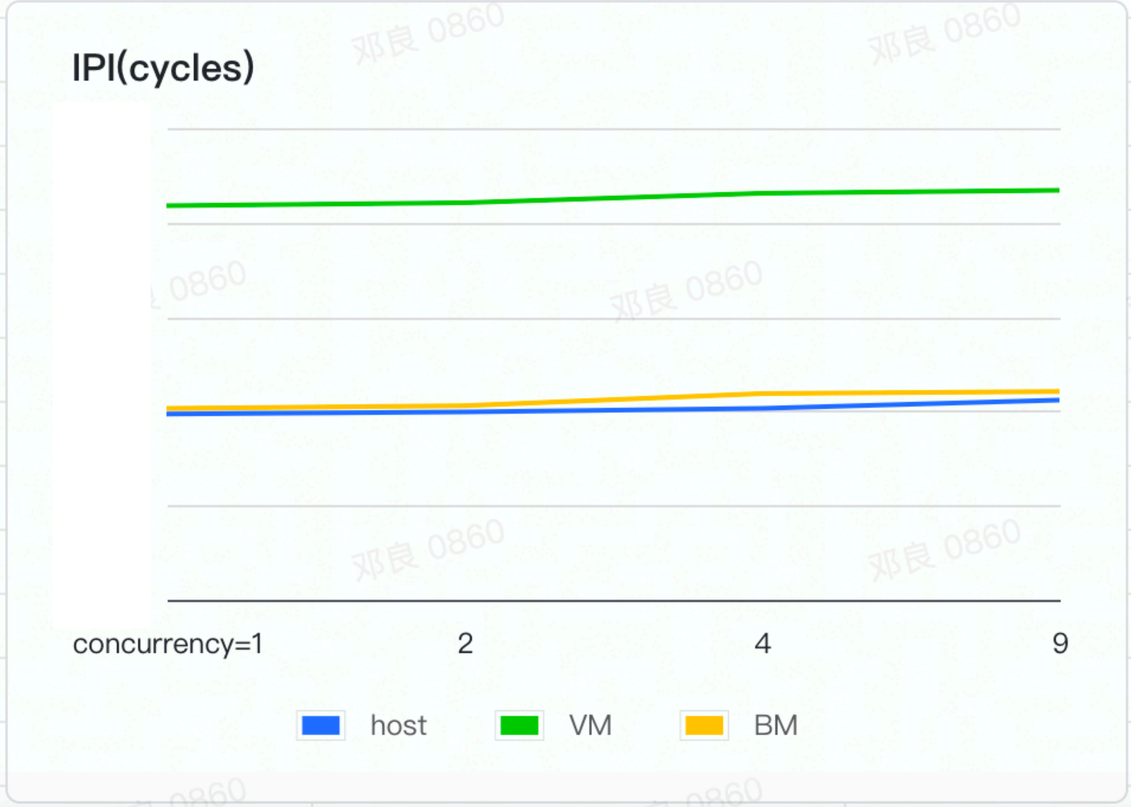
ByteDance字节跳动

Other Optimizations to Remove VM exits

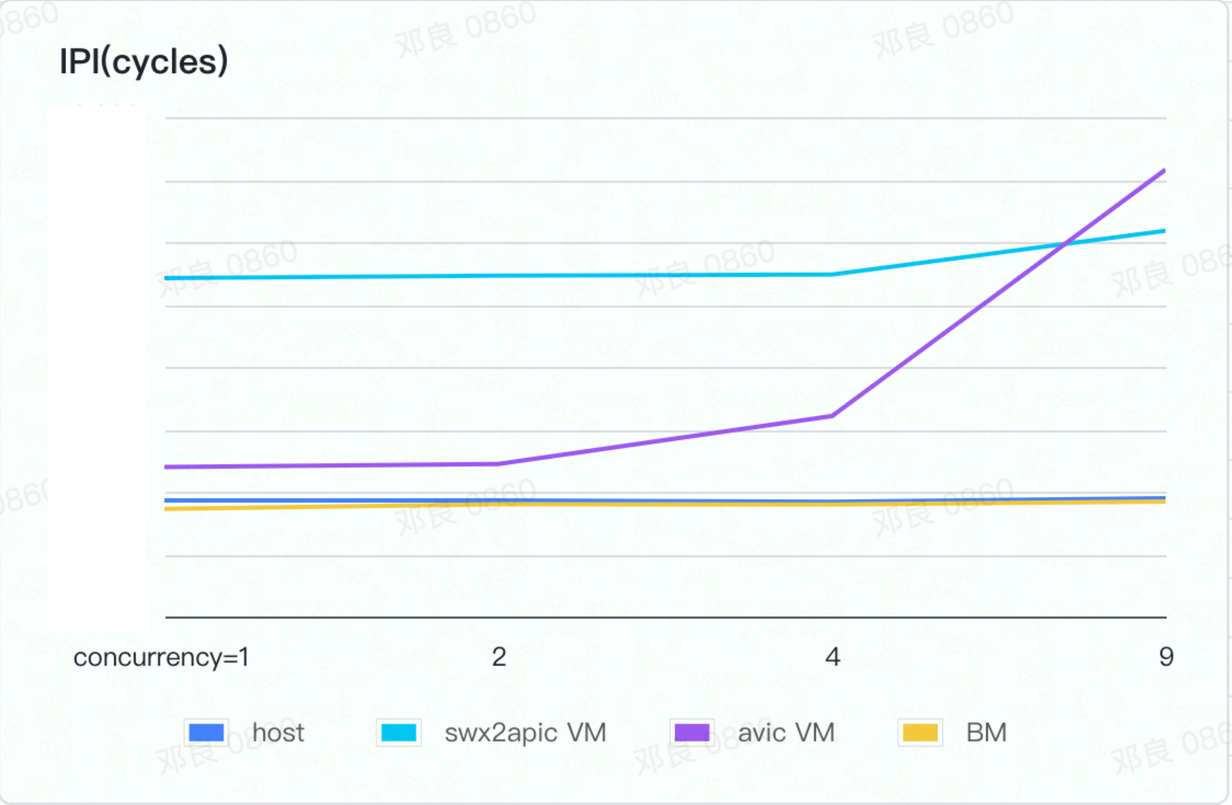
- CPU isolation and no-hz full
- Handle cpuid in BM with dynamic binary rewriting
- The handling of some host IPIs to guest cores are delayed to next VM exit.
- HLT and MWAIT instruction passthrough

ByteDance字节跳动

Micro-benchmark Result for IPI latency



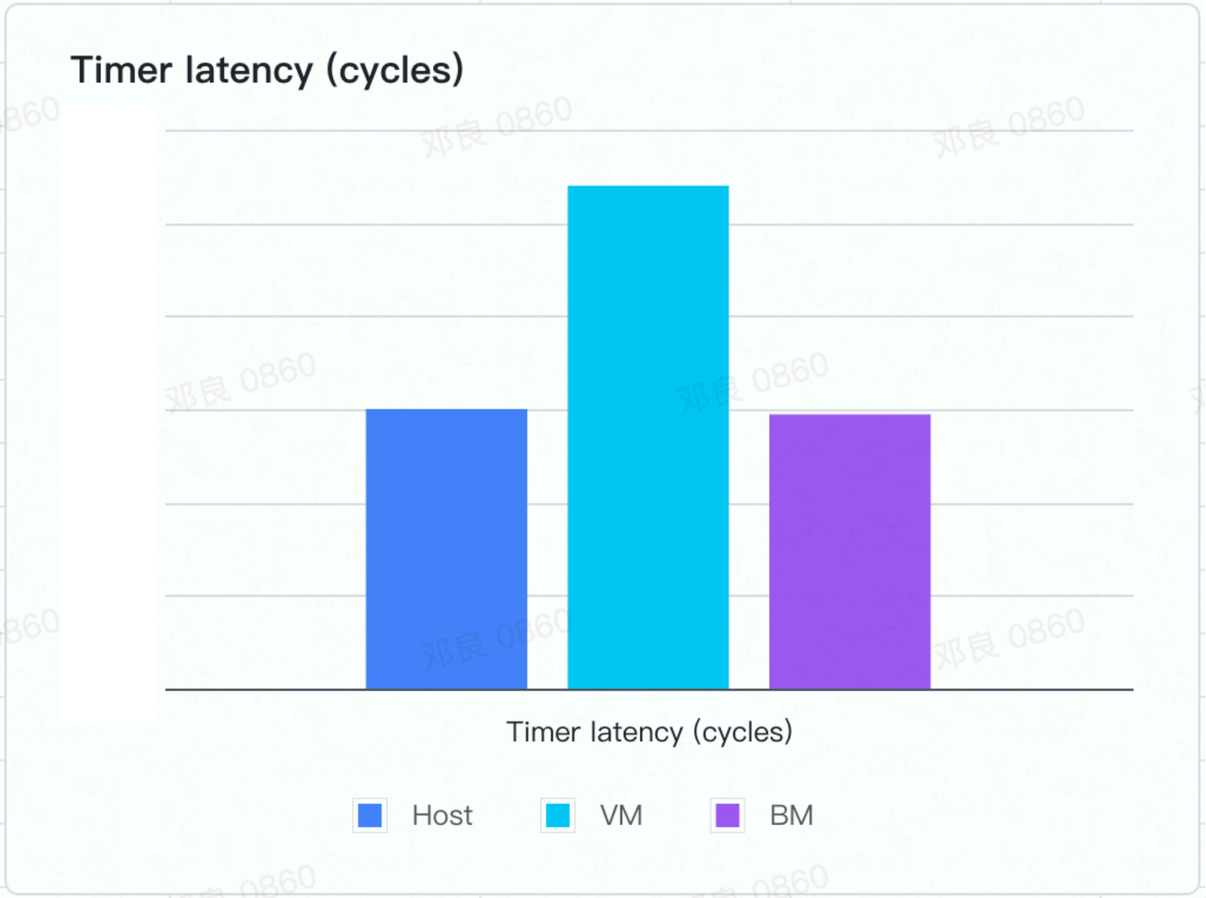
Intel result



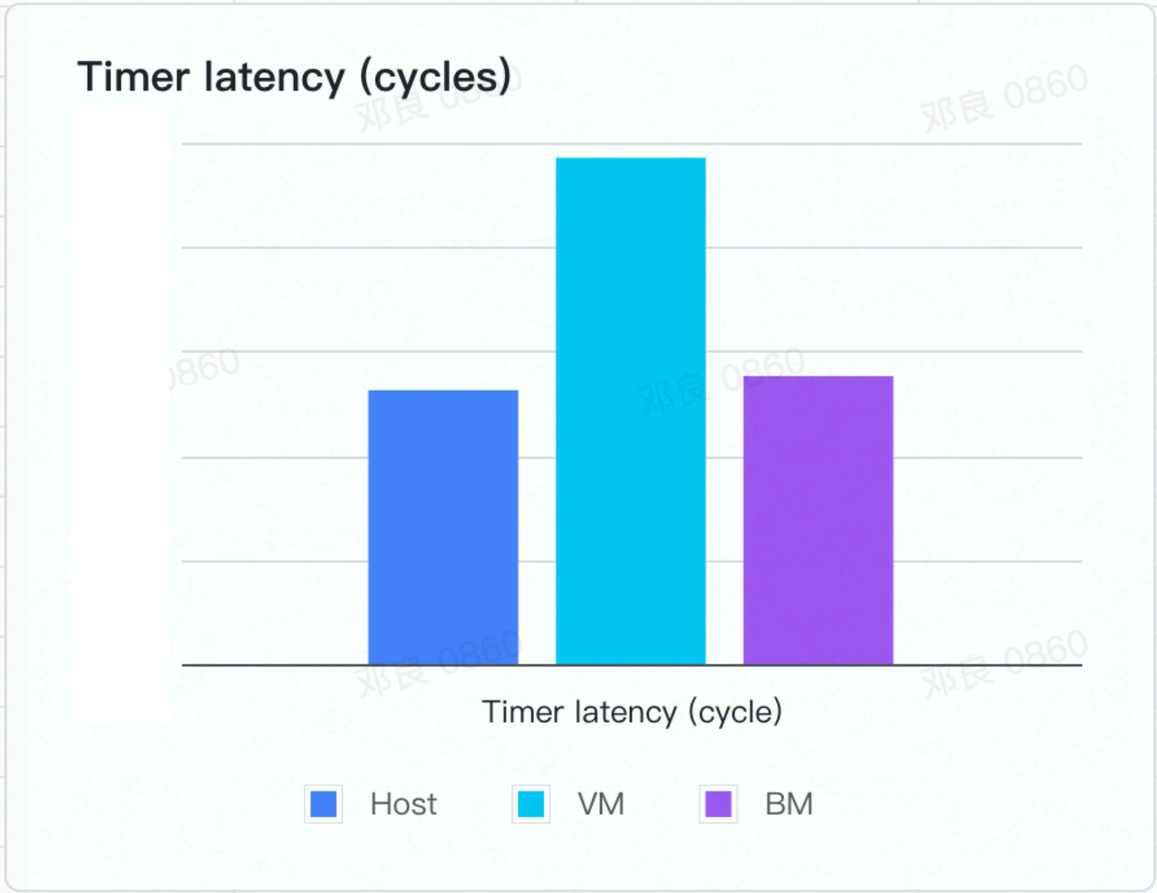
AMD result

ByteDance字节跳动

Micro-benchmark Result for Timer latency



Intel result



AMD result

ByteDance字节跳动

Micro-benchmark Result for Cache Line Prefetch

	Cache Line prefetch latency (lower is better)
Native Host	9.32
BM	9.35
VM with 1GB-size EPT	11.1
VM with 4KB-size EPT	14.3

ByteDance字节跳动

Real-world Application Result (BM vs VM)

Two Real-world Applications in ByteDance

- XX

BM vs VM Result

Optimizations	XX end-to-end latency improvement based on VM
Interrupt + IPI + Timer Passthrough	8%
Memory devirtualization	14%
DMA devirtualization	2%
ALL	20%-30%

ByteDance 字节跳动

Real-world Application Result (BM vs Native Host)

	XX End-to-end latency (normalized, lower is better)
Native Host	1
BM	1.01

One Partition Result

	XX End-to-end latency (normalized, lower is better)
Native Host	1
BM	0.91

Four Partitions Result

One Partition

- Native host: partition the server with only one runc container and run an XX in it.
- BM: partition the server with only one BM and run an XX in it.

Four Partitions

- Native host: partition the server with four runc containers and run an XX in each partition
- BM: partition the server with four BMs and run an XX in each partition

Status and Future Work

ByteDance 字节跳动

Status

- Support both Intel and AMD
- Support both QEMU and Cloud-hypervisor as VMM

Future work

- Kernel patches posted to upstream
- Live migration support
- virtio-balloon and memory hot plug support

ByteDance字节跳动

Thanks very much

Q & A