coIOMMU: A Virtual IOMMU with Cooperative DMA Buffer Tracking for Efficient Memory Management in Direct I/O

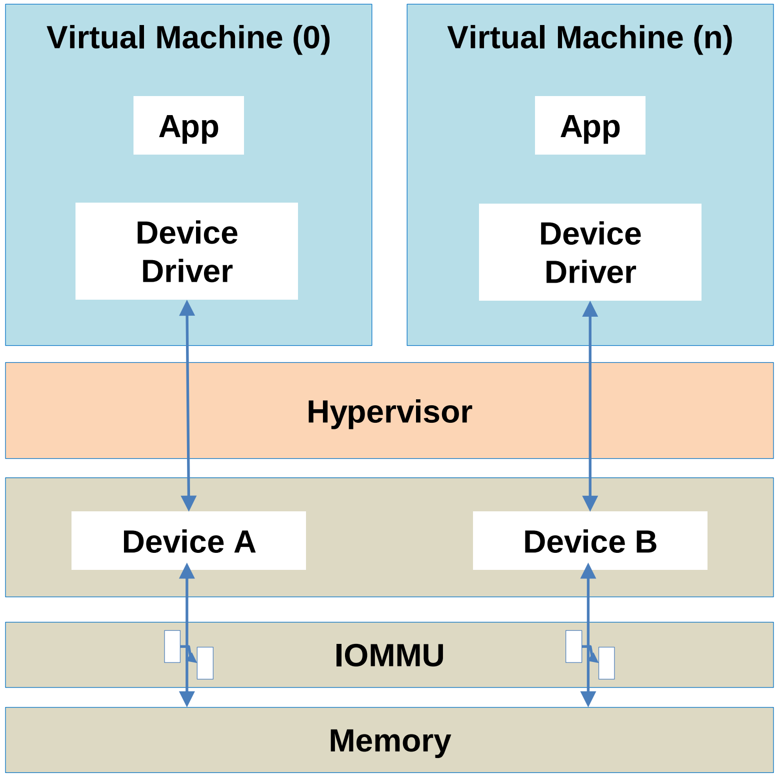
Kun Tian, Yu Zhang (presenter), Luwei Kang, Yan Zhao, Yaozu Dong

Intel Corporation

Legal Disclaimer

Direct I/O

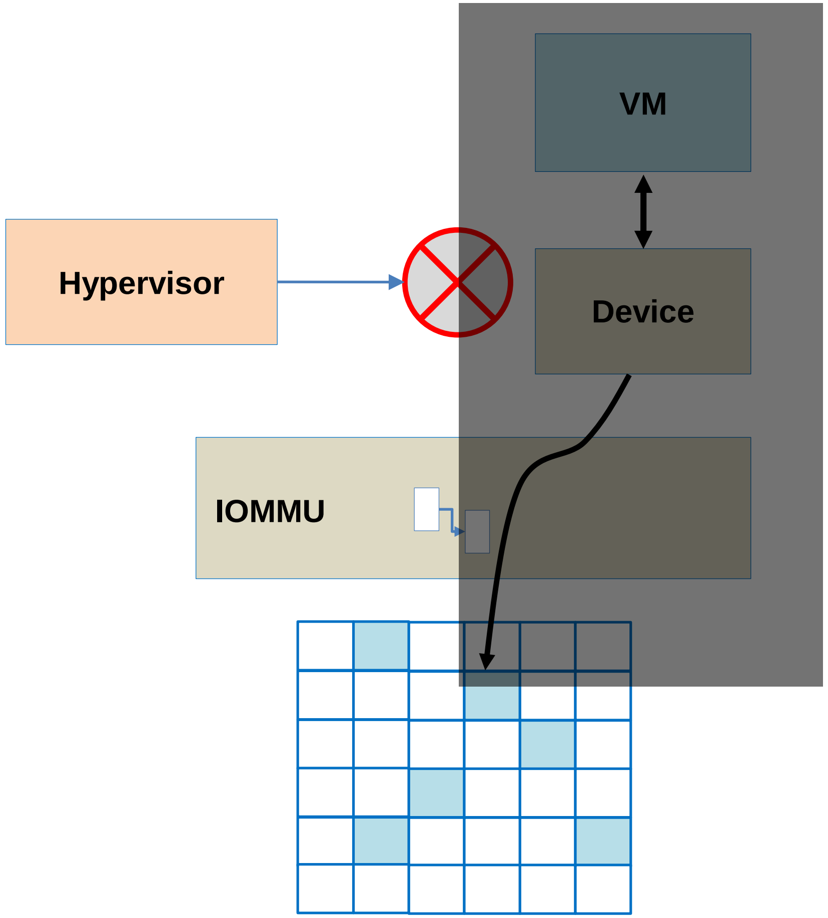
- The best performant I/O virtualization method, widely deployed in cloud and data centers.
- Guest directly interacts with I/O devices, eliminating the host intervention.
- Hardware IOMMU provides inter-guest protection with IOMMU page table (IOPT).



Static Pinning in Direct I/O

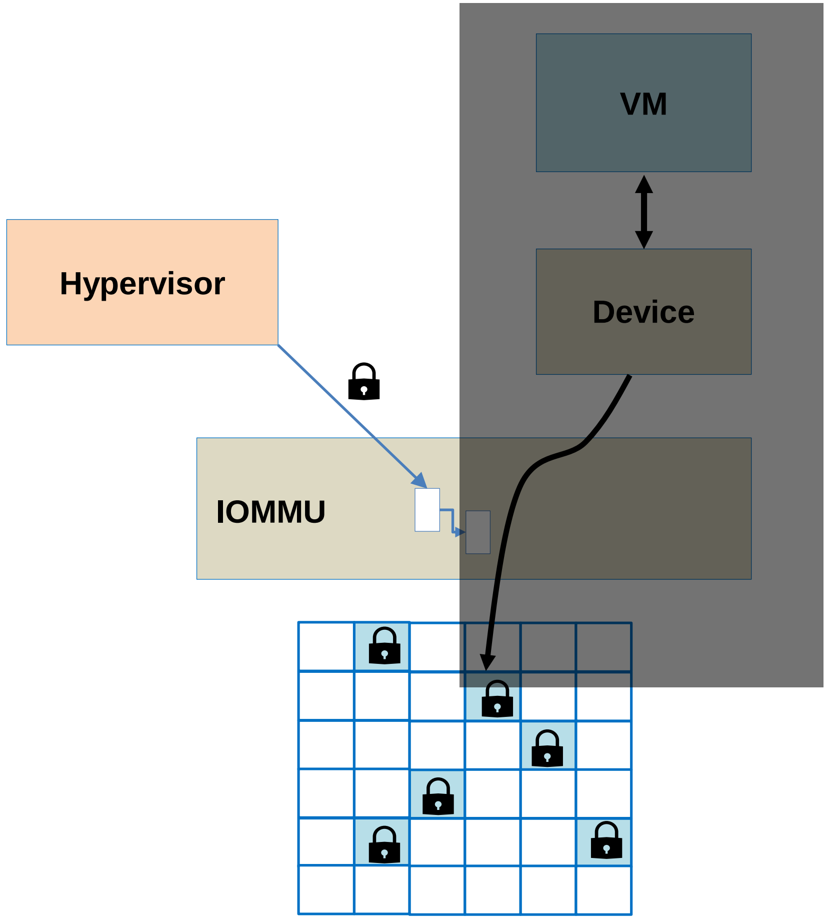
- Most devices do not support DMA page fault.
  - ➢ DMA buffers need be pinned in the IOMMU.

- Hypervisor has no visibility of guest DMA activities.



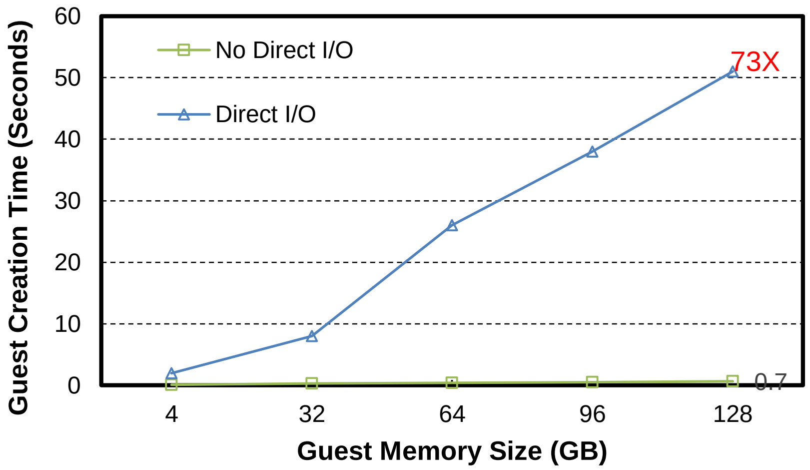
Static Pinning in Direct I/O

- Pre-allocate and pin the entire guest memory before guest DMA starts.
  - ➢ E.g. at VM creation time.



The Problem of Static Pinning

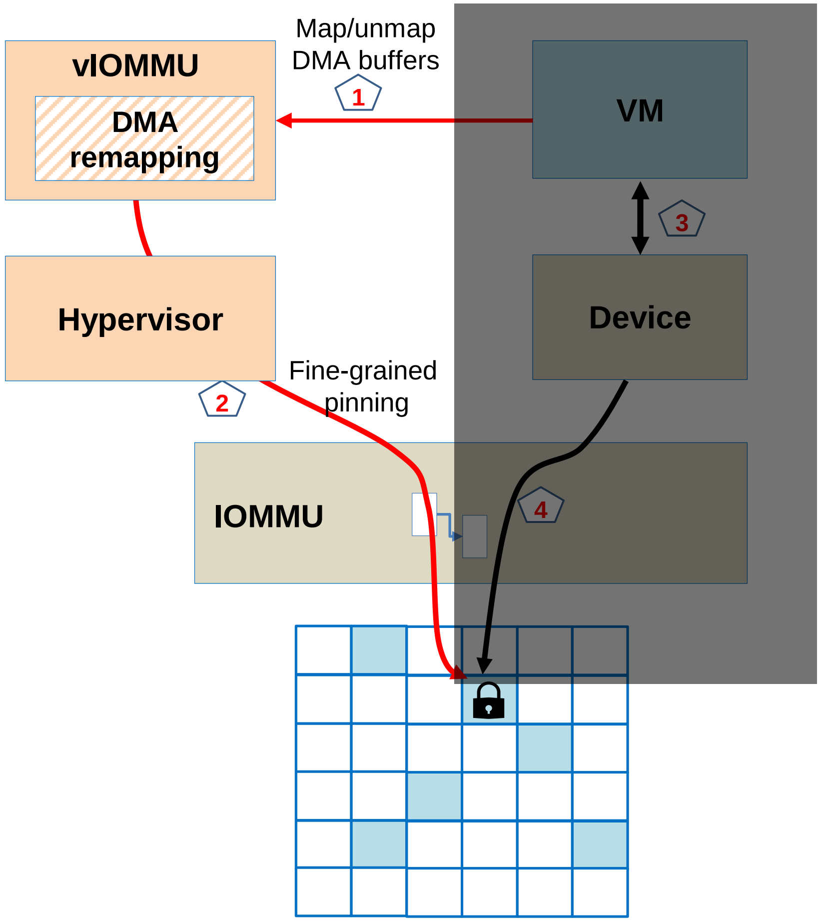
- Much increased VM creation time
  - Up to 73x longer time observed for a VM with 128GB memory.
- Greatly reduced memory utilization
  - Prevent many memory optimizations (overcommitment, late allocation, swap, etc.).



VM creation time increases with guest memory size in static pinning.

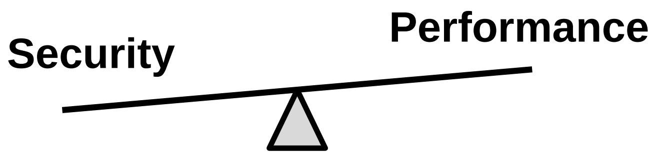
Virtual IOMMU (vIOMMU)

- Primary purpose: intra-guest protection
  - E.g. protection with virtual DMA remapping against bogus guest drivers.
- Side-effect: fine-grained pinning
  - Guest uses vIOMMU to map/unmap DMA buffers.
  - vIOMMU requests hypervisor to pin/unpin guest DMA buffers.
- A vIOMMU could be emulated or para-virtualized.



The Problem

- Emulation cost of established vIOMMUs could be significant!
  - E.g. 96.6% performance downgrade in memcached through 40Gbps NIC.
  - SLA violation if forcing all tenants to turn on vIOMMU.



- Aggressive optimizations may compromise security!
  - E.g. side-core emulation [8], map cache [52], etc.

The Reality

- Virtual DMA remapping is disabled in established vIOMMUs by most guest OSes.
  - Users may opt in when security requirement is over performance concern.
  - E.g. Linux uses ‘passthrough’ by default, leaving ‘strict’/’lazy’ for user opt-in.

- The guest security requirement varies. E.g.
  - when an untrusted device is plugged in;
  - when a device is assigned to untrusted userspace.

Established vIOMMUs are not suitable as a reliable way for fine-grained pinning!

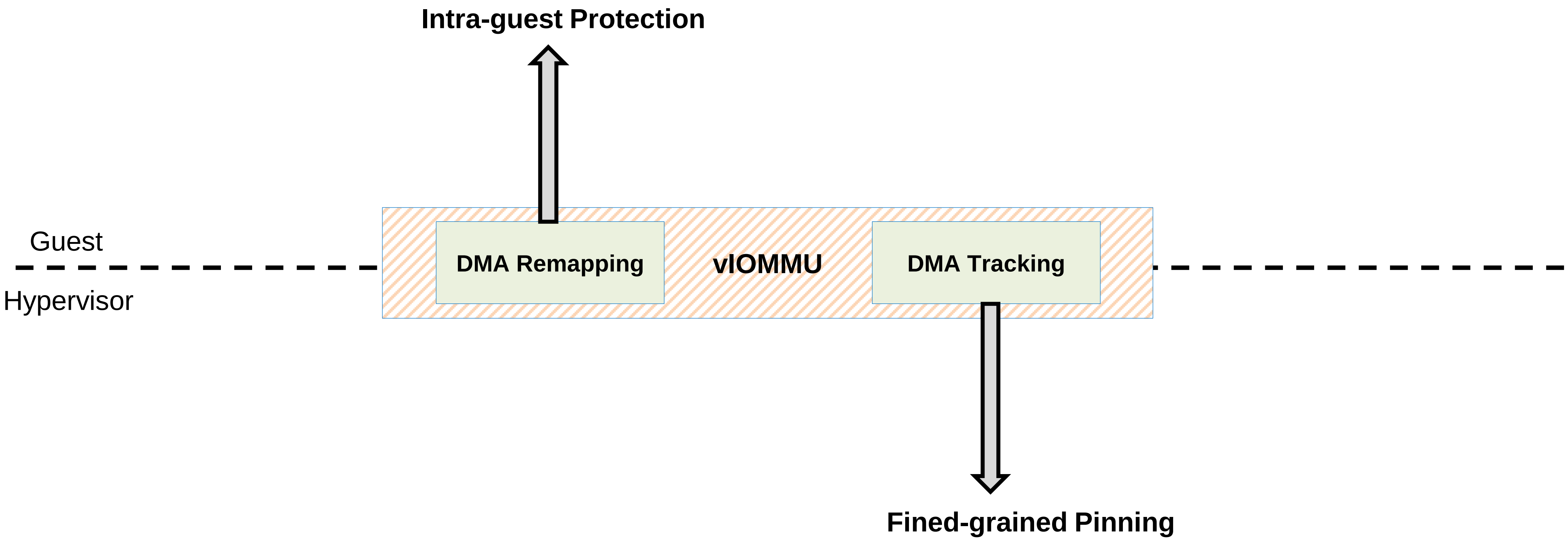
Motivation

- vIOMMU provides an architectural way for learning guest DMA buffers.

- However, mixing the requirements of protection and pinning, through the same costly DMA remapping interface, is needlessly constraining.
  - Protection is an OPTIONAL guest-side requirement.
  - Fine-grained pinning is a GENERAL host-side requirement.

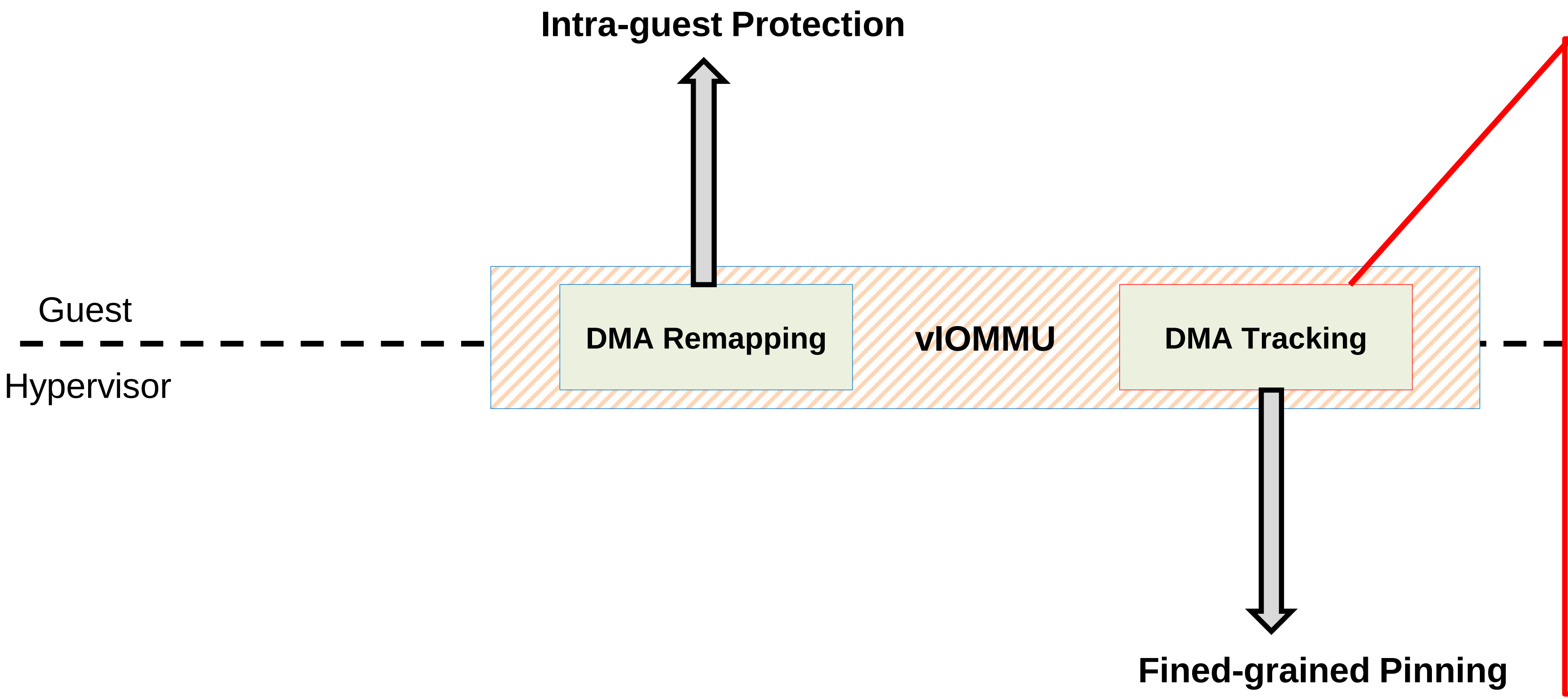
Motivation

- Decouple DMA tracking and DMA remapping in vIOMMU.



Motivation

- Decouple DMA tracking and DMA remapping in vIOMMU.

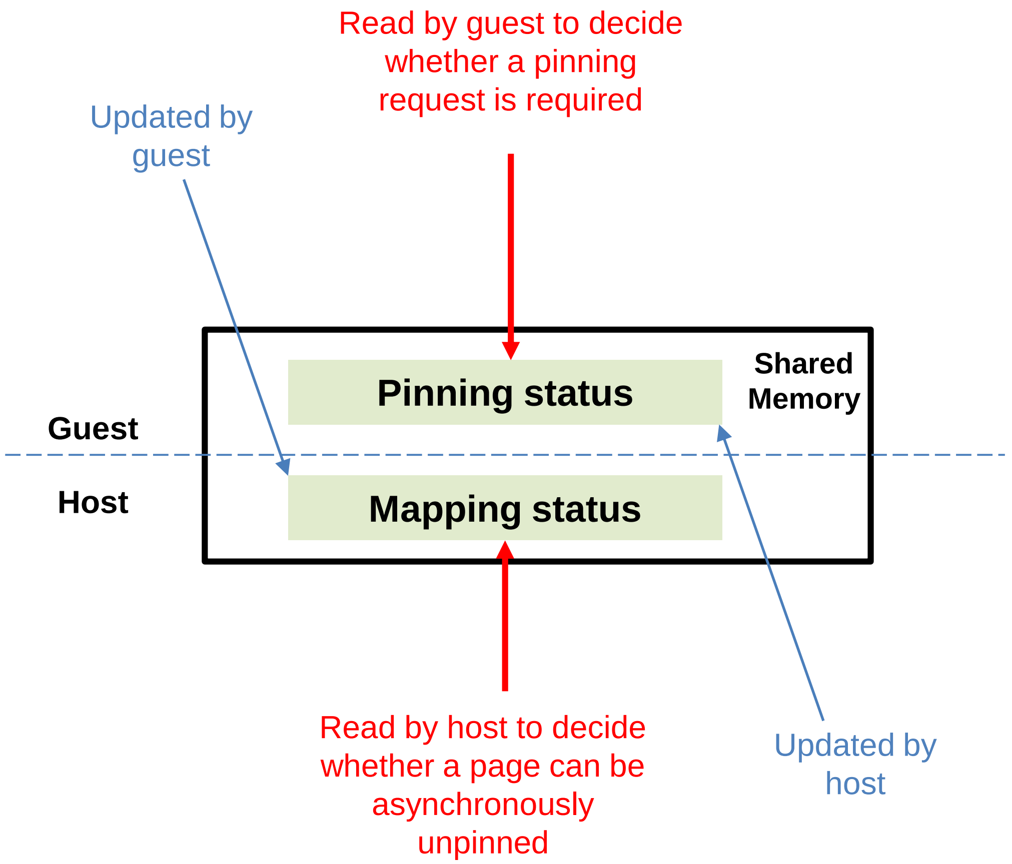


Goals

- Orthogonal to remapping
- Low cost
- Non-intrusive
- Widely applicable
- Extensible

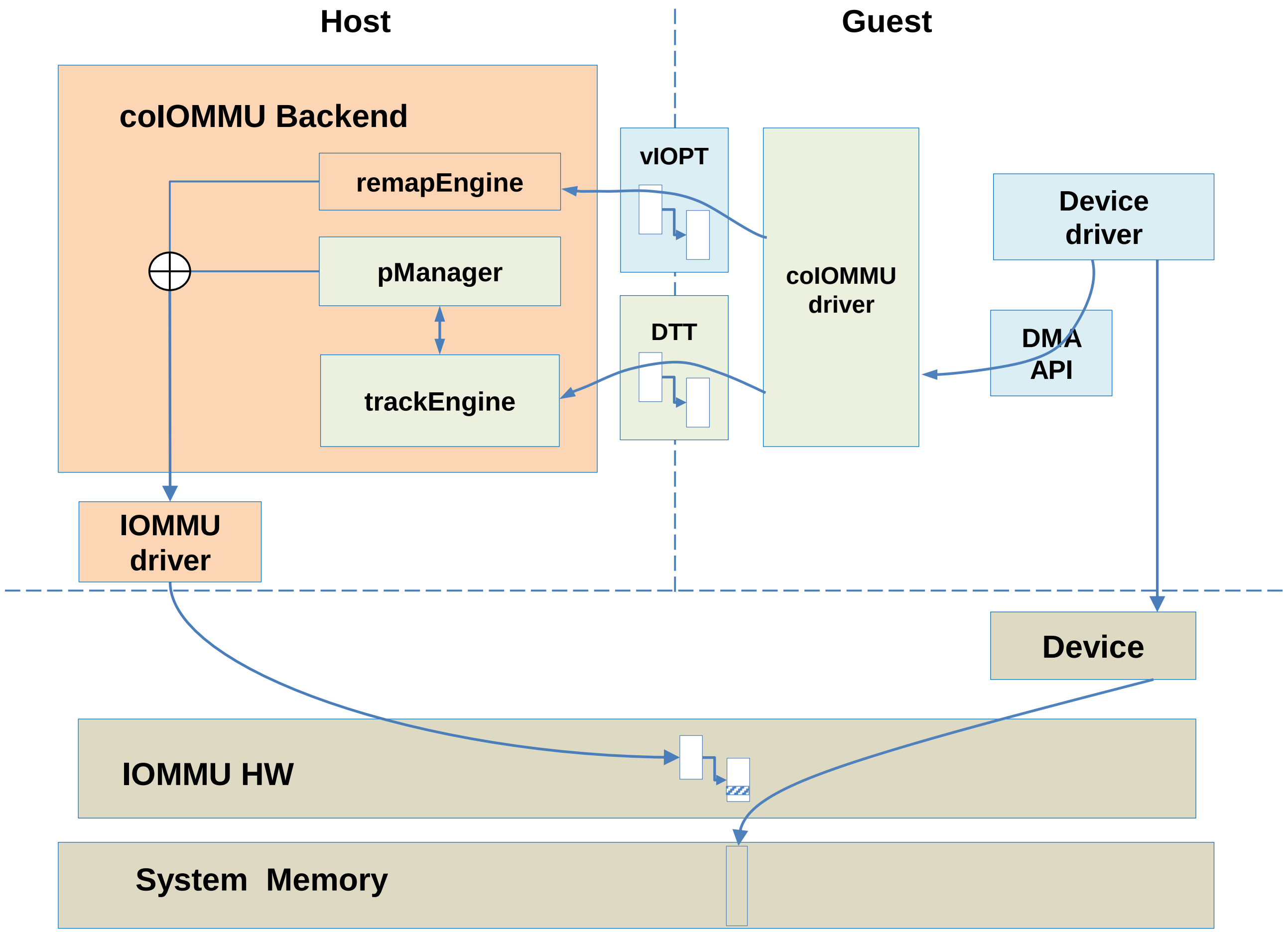
Cooperative DMA Buffer Tracking

- Bi-directional shared DMA buffer information
  - To guest – whether a page is pinned in the IOMMU.
  - To host – whether a page is mapped for DMA.
- A lightweight tracking interface for fine-grained pinning
  - Minimize VM-exits when mapping DMA pages
  - Eliminate VM-exits when unmapping DMA pages
  - Enable flexible host memory management policies



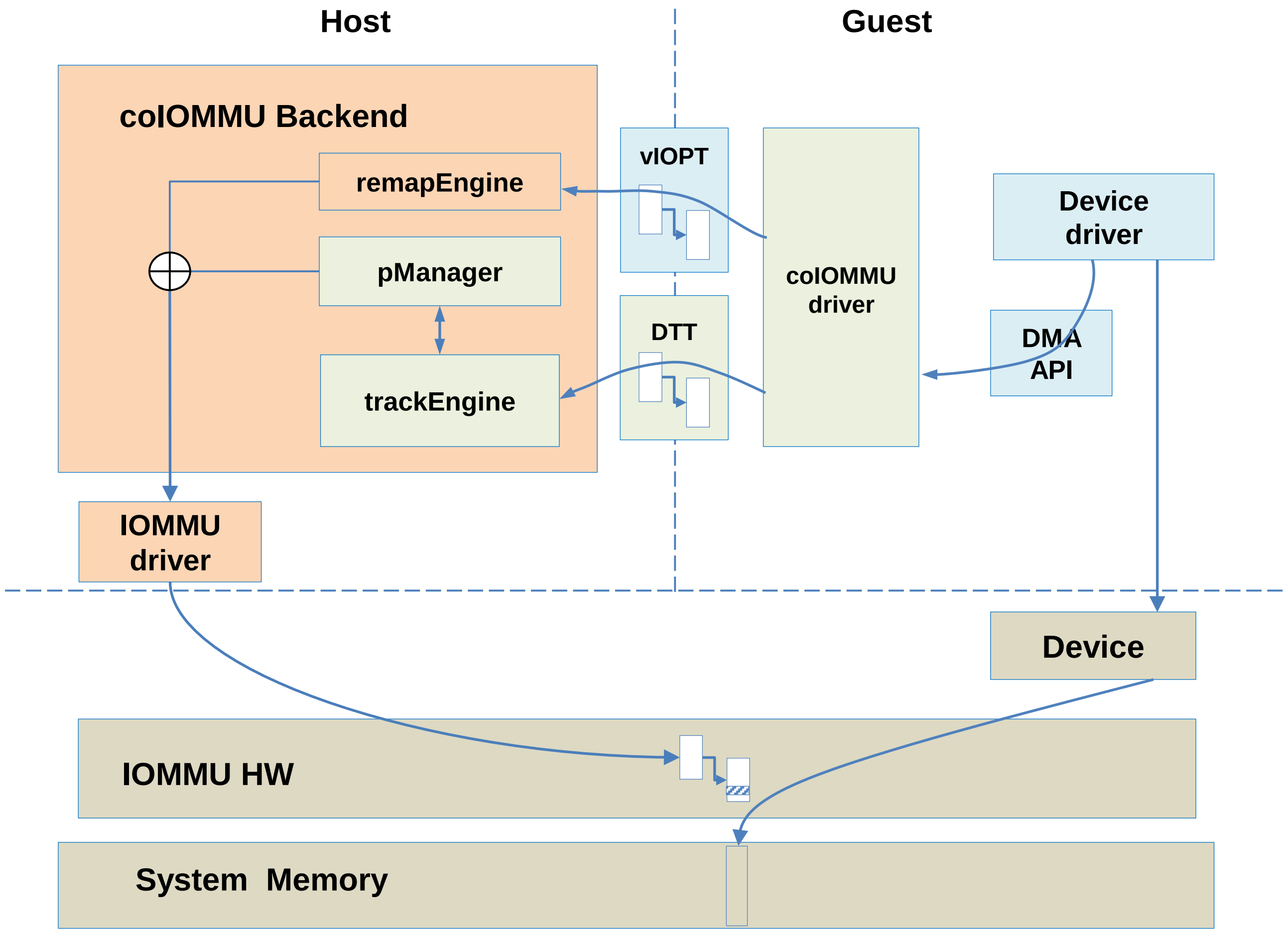
coIOMMU Architecture

- DMA Tracking Table (DTT)
  - Hold shared DMA buffer info.
- coIOMMU driver
  - Hook to guest DMA API layer.
- coIOMMU backend
  - DMA remapping engine (remapEngine)
  - DMA tracking engine (trackEngine)
  - Page pinning manager (pManager)



coIOMMU Architecture

- remapEngine
  - Same as established DMA remapping interface.
- trackEngine
  - Holds base address of the DTT.
  - Emulates a doorbell register for notifying the host.
- pManager
  - Implements fine-grained pinning policy.
  - Invisible to guest.

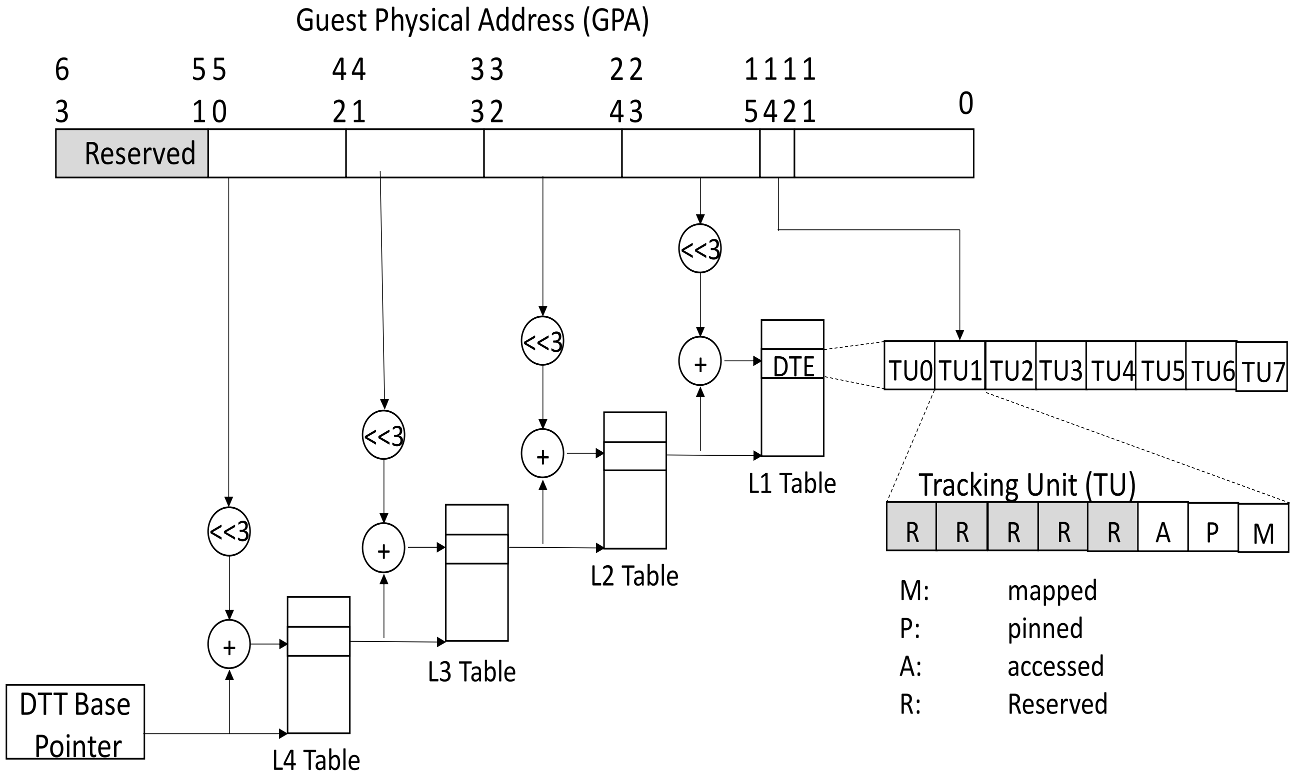


DMA Tracking Table (DTT)

- A multi-level paging structure
  - Shared between host & guest.
  - Indexed by guest page frame numbers (GFNs).

- TU - Tracking Unit for each guest page frame number(GFN)
  - ‘M’ (mapped) – set/cleared by guest.
  - ‘P’ (pinned) - set/cleared by host.
  - ‘A’ (accessed) – set by guest, cleared by host.

- Extensible through 5 reserved bits
  - E.g. add a ‘D (dirty)’ bit to assist dirty page tracking in live migration.



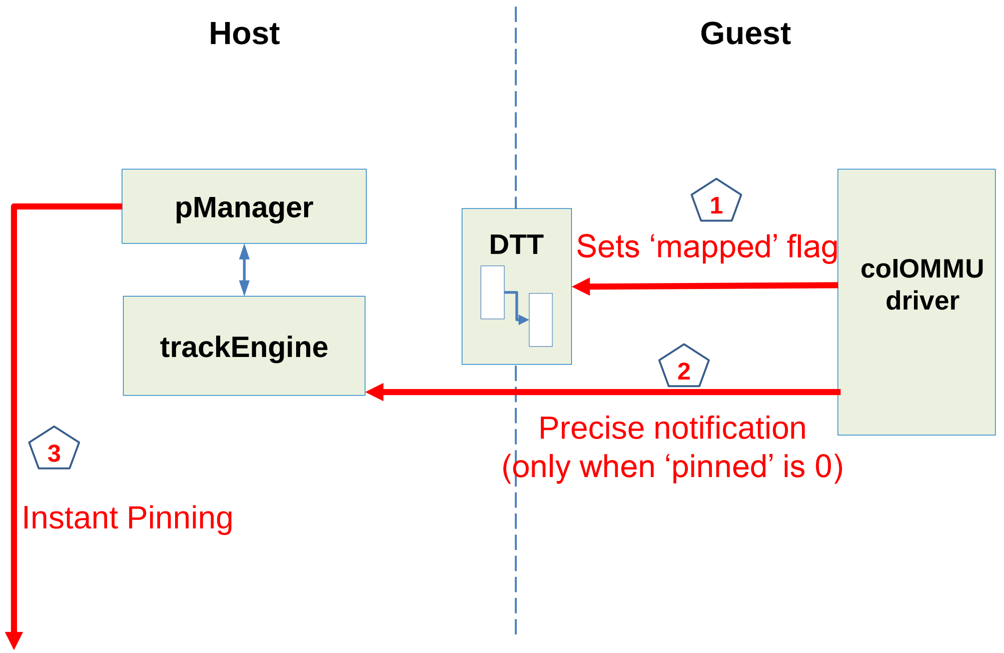
Fine-grained Pinning

- Smart pinning
  - Instant pinning – pinning must be instantly done before any mapped page is used for DMA.
  - Precise notification – only notify the hypervisor for pages not pinned.
  - Speculative pinning – pMananger speculatively pins frequently used pages.

- Lazy unpinning
  - Asynchronously done by pManager.
  - Only tries to unpin the pages that are no longer mapped.
  - Unpinned pages are reclaimable.

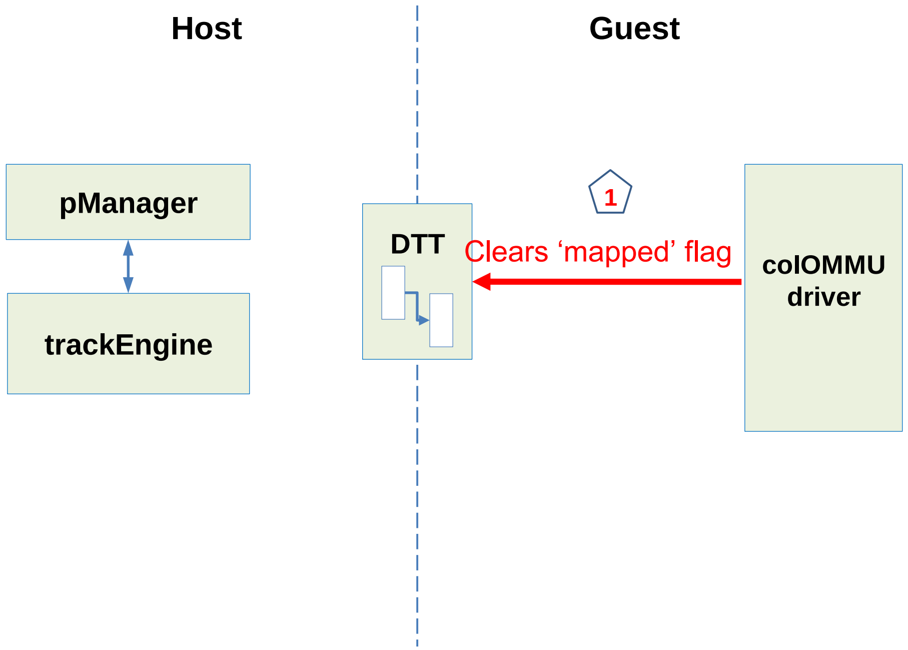
Guest Mapping Operations

Guest DMA map



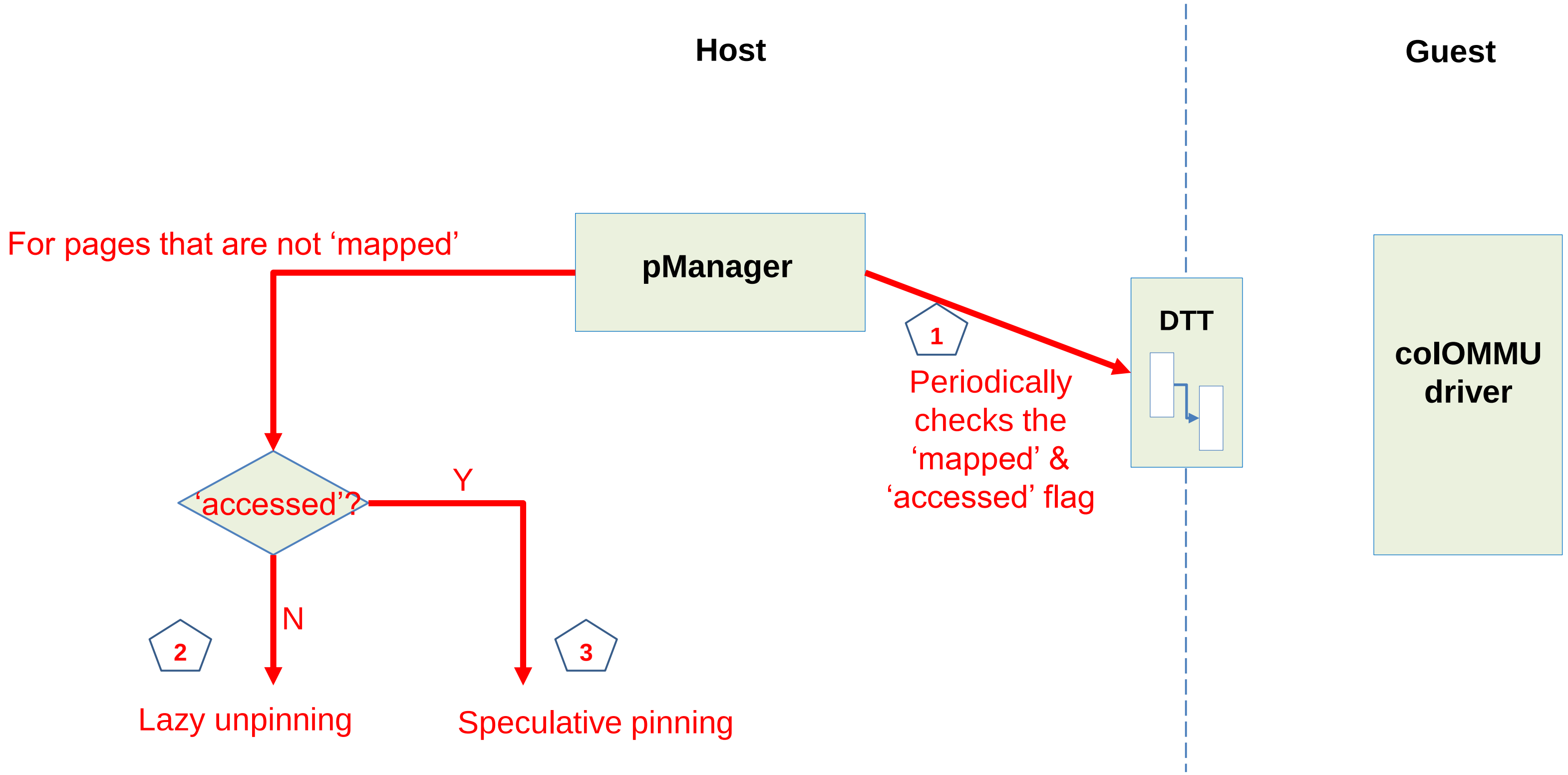
Up to 99.9992% notifications can be avoided due to DMA buffer locality!

Guest DMA unmap



No notification is required

Lazy Unpinning & Speculative Pinning



Host asynchronously manages pinning & unpinning in a separate thread.

DMA Tracking vs. DMA Remapping

- When DMA remapping is disabled by guest (the majority case).
  - DMA tracking is an efficient solution to achieve fine-grained pinning.
- When DMA remapping is conditionally enabled.
  - E.g. only for selective devices (e.g. untrusted), or only in specific period (e.g. when the device is assigned to userspace).
  - However, hypervisor requires full visibility of guest DMA activities for the entire VM life-cycle.
  - In such cases, DMA tracking helps provide a reliable way for fine-grained pinning.
- When DMA remapping is always enabled (for all devices at all times).
  - DMA tracking provides a consistent tracking interface as other two categories, with negligible cost.

Implementation

- Based on KVM/QEMU.
- Extend existing virtual Intel VT-d.
  - Reused the remapping logic in vIOMMU as remapEngine.
  - Developed pManager and trackEngine from scratch.
  - Extended guest intel-iommu driver to support DMA tracking.
- Applicable to all kinds of direct I/O usages.
  - No ad-hoc changes in hardware or device drivers.
- Applicable to other OSes.
  - As long as a generic DMA API layer is afforded.
- Applicable to other vIOMMUs.
  - New tracking interface is vendor-agnostic and self-contained.

	New/Changed LOC		
Guest	**Intel VT-d driver**		832 new
			47 changed
Host	**QEMU**	trackEngine	131 new
		pManager	552 new

- Less than 700 LOC in QEMU.
- Less than 1000 LOC in guest.

Implementation

- Huge page mappings
  - The DTT tracks guest pages in 4KB granularity.
  - pManager is optimized to conduct 2MB page pinning by merging continuous guest pages.

- Sub-page mappings
  - Multiple DMA buffers may co-locate in the same 4KB guest page (e.g. network packets).
  - Guest coIOMMU driver tracks the mapping count of each mapped page.

Implementation

- Kernel Bypassing
  - Kernel bypass APIs require userspace to pre-register a trunk of memory.
  - Pre-registered memory is mapped through kernel driver, thus still trackable in coIOMMU.

- Concurrency
  - coIOMMU must properly handle concurrent pinning/unpinning requests between multiple vCPU threads and the unpinning thread.

Evaluation

- Evaluation targets
  - Performance overhead imposed by coIOMMU.
  - Memory footprint in various direct I/O usages.
  - The desired performance and security under different intra-guest protection policies.

- Evaluated modes – coIOMMU vs. virtual VT-d
  - Passthrough mode: no DMA remapping
    - PT-N (coIOMMU) vs. PT-O (virtual VT-d)
  - Strict mode: full protection with DMA remapping
    - ST-N (coIOMMU) vs. ST-O (virtual VT-d)
  - Lazy mode: relaxed protection with DMA remapping
    - LA-N (coIOMMU) vs. LA-O (virtual VT-d)

mode	abbr.	DMA remapping	DMA buffer tracking	pinning model	protection
passthrough (virtual VT-d)	*PT-O*	unused	n/a	static	no
passthrough (coIOMMU)	*PT-N*	unused	used	fine-grained	no
strict (virtual VT-d)	*ST-O*	used	n/a	fine-grained	full
strict (coIOMMU)	*ST-N*	used	used	fine-grained	full
lazy (virtual VT-d)	*LA-O*	used	n/a	fine-grained	relaxed
lazy (coIOMMU)	*LA-N*	used	used	fine-grained	relaxed

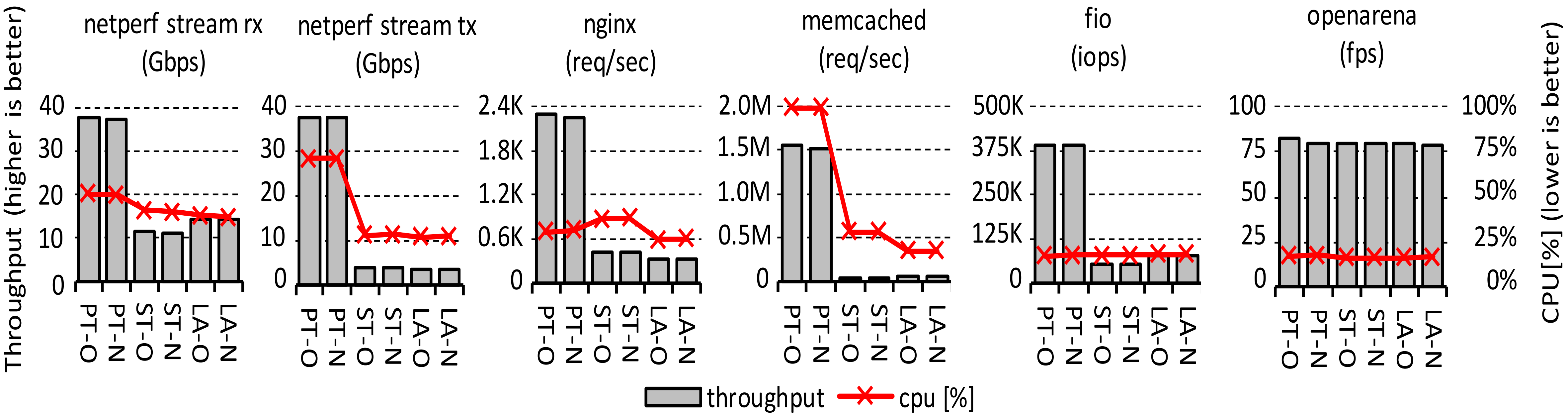
Evaluation

- Three direct I/O usages: NIC/NVMe/GPU.

- Benchmarks
  - Netperf: Aggregated throughput reported, for 16 concurrent Netperf instances running stream RX & TX tests.
  - Nginx: Requests/second reported, for 16 concurrent requests to Nginx server installed in guest.
  - Memcached: Requests/second reported, for 16*8 concurrent requests to Memcached installed in guest.
  - FIO: IO requests/second reported, for 16 concurrent fio threads, each performing asynchronous direct random reads to NVMe.
  - Open Arena: Frame-per-second (fps) reported as benchmark

VM configuration		
Direct I/O device	vCPU number	RAM size
Intel XL710 40Gbps NIC	16	32GB
Intel 760P series 1TB NVMe SSDs	16	32GB
Intel® Iris® Plus graphics 650 GPU	4	4GB

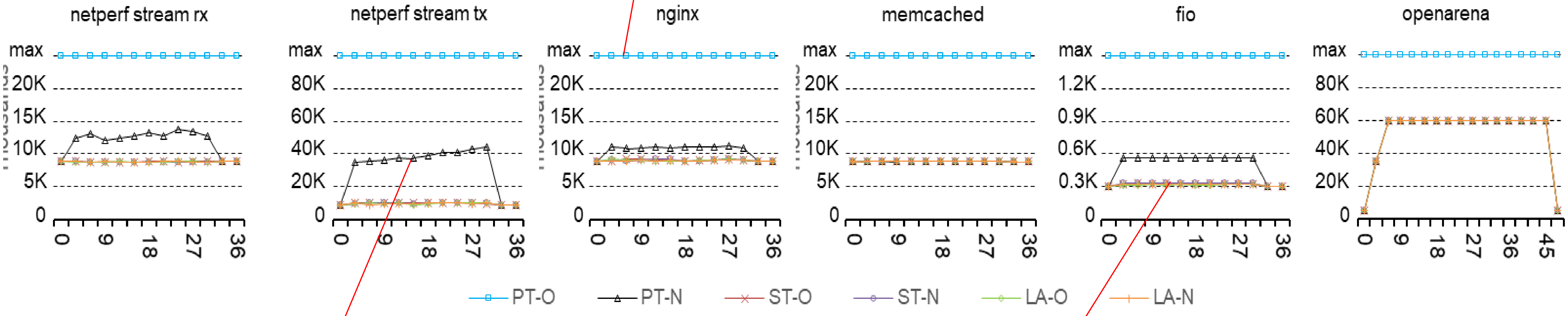
Performance



No observable performance impact with DMA Tracking!
(even in mixed netperf/fio scenario – data not shown here)

Memory Footprint

Entire 32GB guest memory is statically pinned w/o DMA tracking



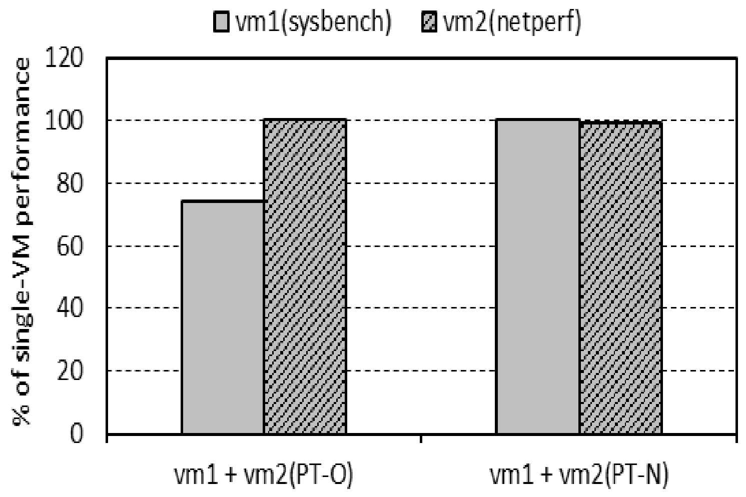
The number of pinned pages sampled in 3 second interval, taken from the beginning of the benchmarks to 6 seconds after their completion. 'max' indicates the total pages of guest memory.

Fine-grained pinning with DMA Tracking, with only 0.5% of guest memory pinned

All four DMA remapping modes pin the minimal number of pages.

Memory Overcommitment

- Test setup
  - Host: 64GB RAM size.
  - VM1: 32GB RAM, running sysbench(no assigned device).
  - VM2: 48GB RAM, assigned with Intel XL710 40Gbps NIC, running Netperf.
  - Performance compared with running each benchmark alone.

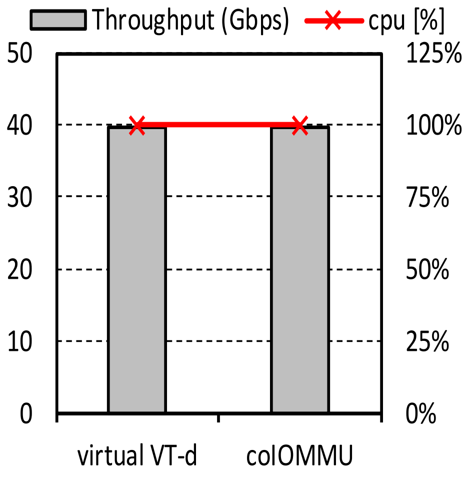


The impact of memory overcommitment:
static pinning (*PT-O*) vs. fine-grained pinning (*PT-N*)

- Results
  - PT-O: Sysbench suffers 25+% performance drop, frequent page swaps.
  - PT-N: No performance drop, with 49GB free memory.

Guest User Space Driver

- Run DPDK with coIOMMU and with the virtual VT-d respectively.



Spent Cycles

	virtual VT-d	coIOMMU	ratio
Create VM	7554ms	407ms	18x
Assign NIC	183ms	2ms	91x
Deassign NIC	815ms	2ms	407x

Pinned Pages

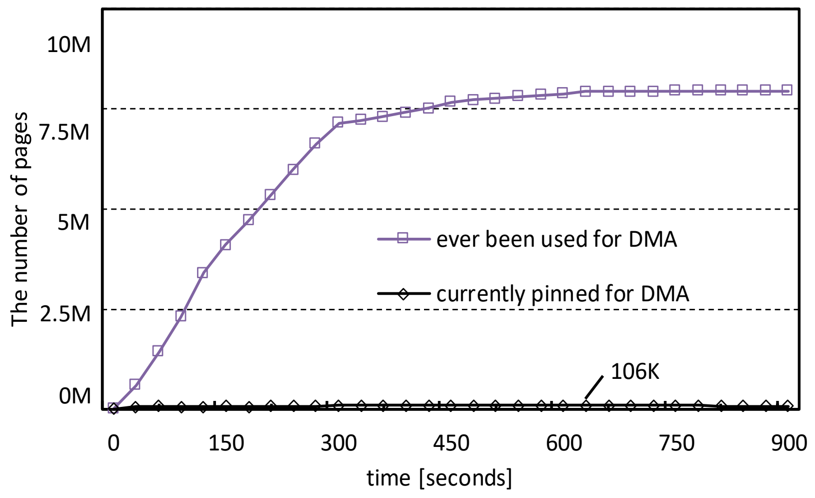
	virtual VT-d	coIOMMU	ratio
Before DPDK	8388608	548	15307x
In DPDK	186368	186368	1x
After DPDK	8838608	548	15307x

- No need to allocate and pin the entire guest memory in coIOMMU.
- No need to unpin the entire guest memory in coIOMMU.
- Likewise, static-pinning is avoided in coIOMMU when assigning NIC back to kernel driver.
- coIOMMU always adapts to the actual DMA buffer requirement, while virtual VT-d fails to do so when DMA remapping is off.

DMA Temporal Locality

- Test setup
  - 16 Netperf TX instances ran for 15 minutes.
  - ‘dd’ the virtual disk to /dev/zero, to contend the page allocation with the networking stack.

- Conclusion
  - DMA temporal locality stays good, even in stressed scenario.



DMA temporal locality when running Netperf with ‘dd’

Future Work

- Co-work with DMA page faults
  - Help reduce the number of DMA faults by proactively pre-pin hot pages.
  - Mitigate non-faultable data paths if a device (e.g. many GPUs) only partially supports DMA page faults.

- Guest cooperation
  - A selfish guest may choose to not cooperate, e.g., by deliberately report fake DMA pages.
  - A quota mechanism can be applied, based on the service level agreement.

- Support two-level IOMMU address translation.
  - Hardware optimization to reduce virtual IOTLB invalidations.

Conclusions

- Established vIOMMUs cannot reliably eliminate static pinning in direct I/O.

- coIOMMU offers a reliable approach to achieve fine-grained pinning, with a cooperative DMA buffer tracking method.

- coIOMMU
  - ➢ dramatically improves the efficiency of memory management in wide direct I/O usages with negligible cost;
  - ➢ meanwhile sustains the desired security as required in specific protection usage;
  - ➢ can be easily applied in various vIOMMU implementations.

Thanks!

kevin.tian@intel.com
yu.c.zhang@intel.com
luwei.kang@intel.com
yan.y.zhao@intel.com
eddie.dong@intel.com