A Virtual IOMMU with Cooperative DMA Buffer Tracking

Yu Zhang
Intel Corporation

yu.c.zhang@intel.com

Legal Disclaimer

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

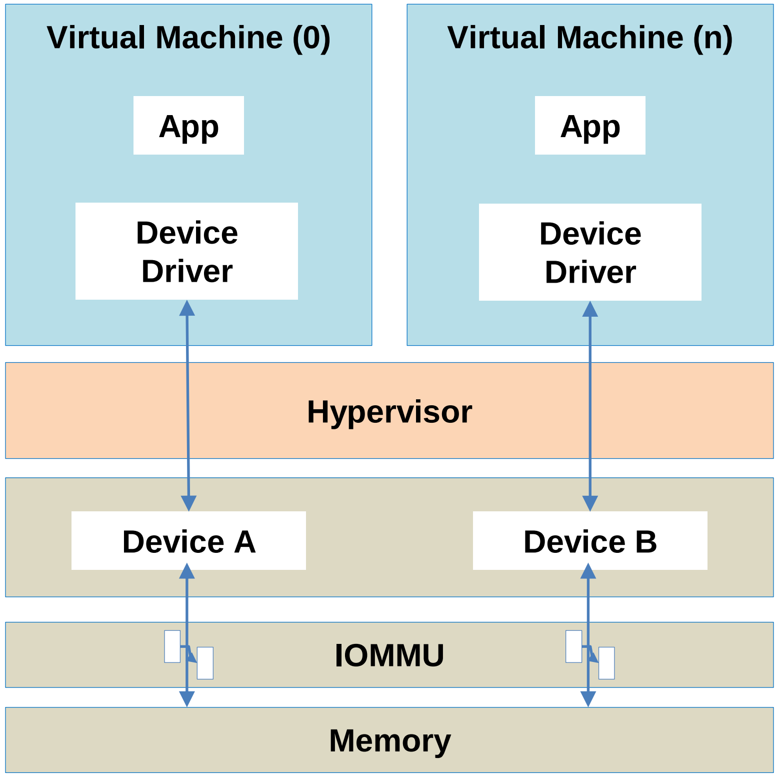
Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Agenda

- Background
  - Static pinning in direct I/O
  - The problem of static pinning and of vIOMMU
  - Motivation of DMA tracking
- Design of coIOMMU
  - A virtual IOMMU with cooperative DMA buffer tracking for direct I/O
- Upstream considerations

Direct I/O

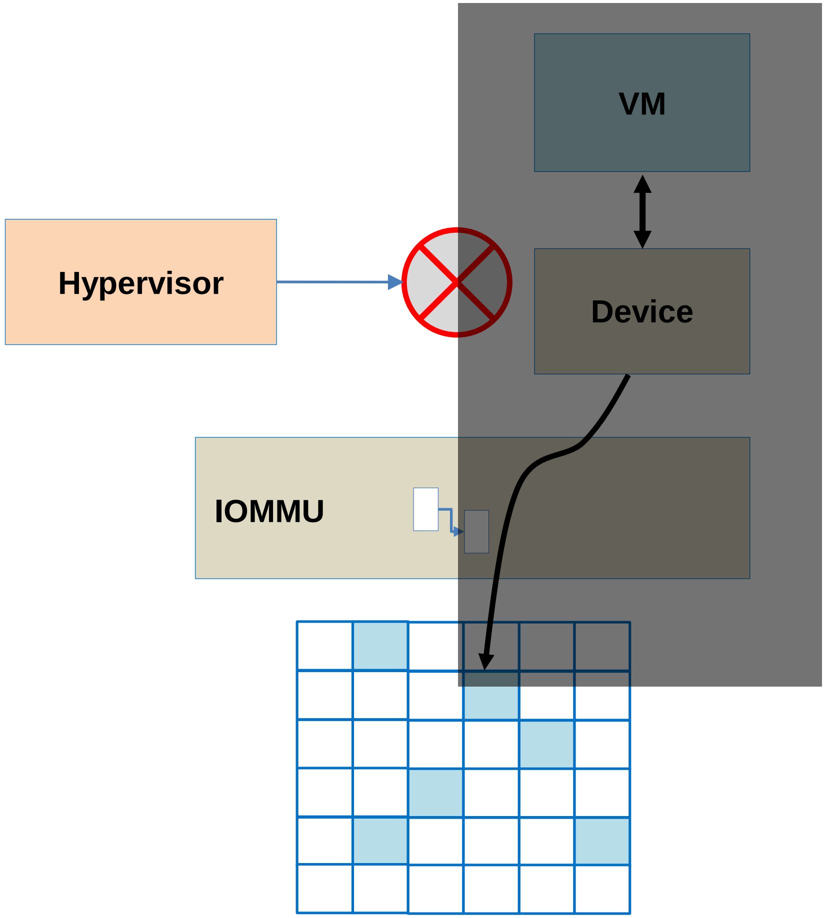
- The best performant I/O virtualization method, widely deployed in cloud and data centers.
- Guest directly interacts with I/O devices, eliminating the host intervention.
- Hardware IOMMU provides inter-guest protection with IOMMU page table.



Static Pinning in Direct I/O

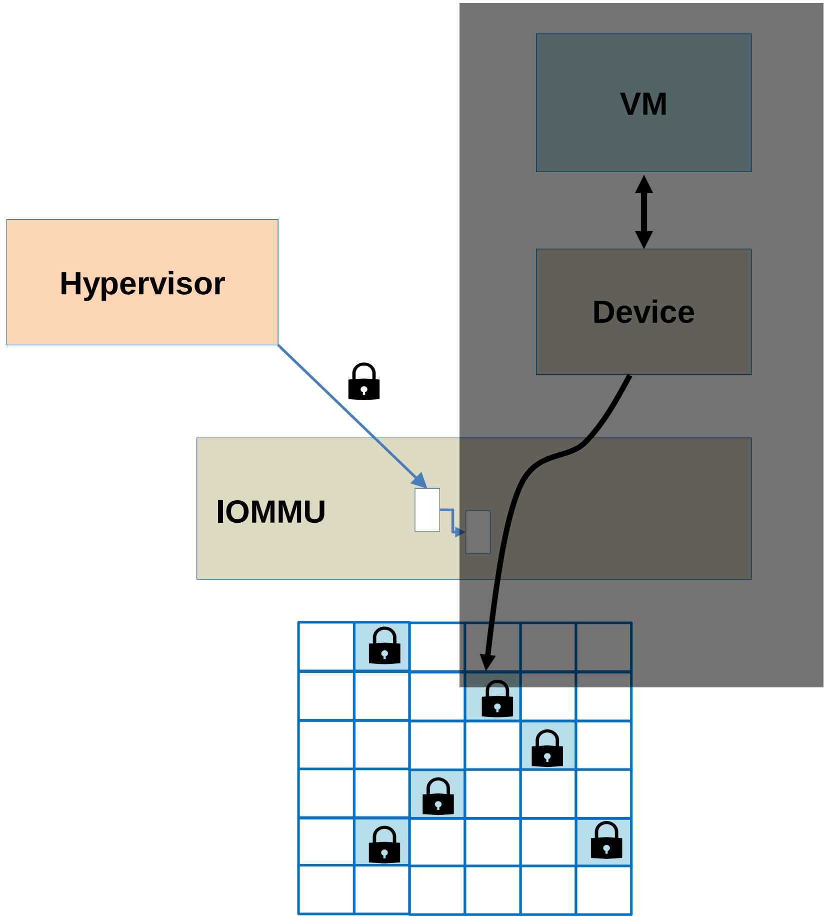
- Most devices do not support DMA page fault.
  - DMA buffers need be pinned first.

- Hypervisor has no visibility of guest DMA activities.



Static Pinning in Direct I/O

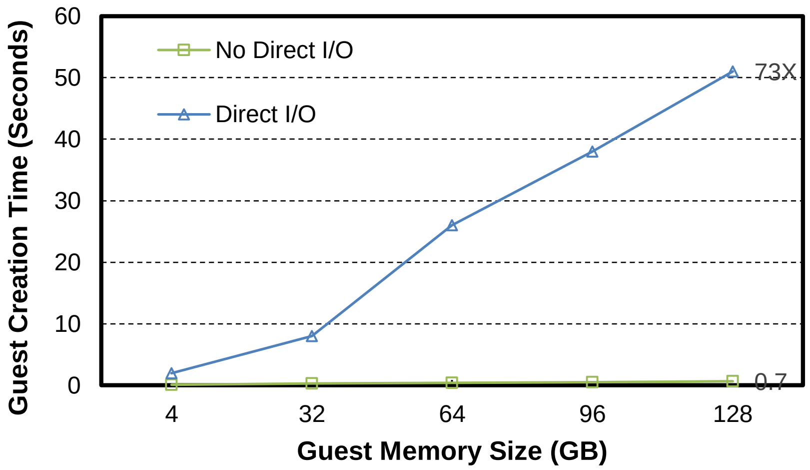
- Pre-allocate and pin the entire guest memory before guest DMA starts:
  - E.g. at VM creation time.



The Problem of Static Pinning

- Much increased VM creation time:
  - ➢ Up to 73x longer time observed for a VM with 128GB memory.

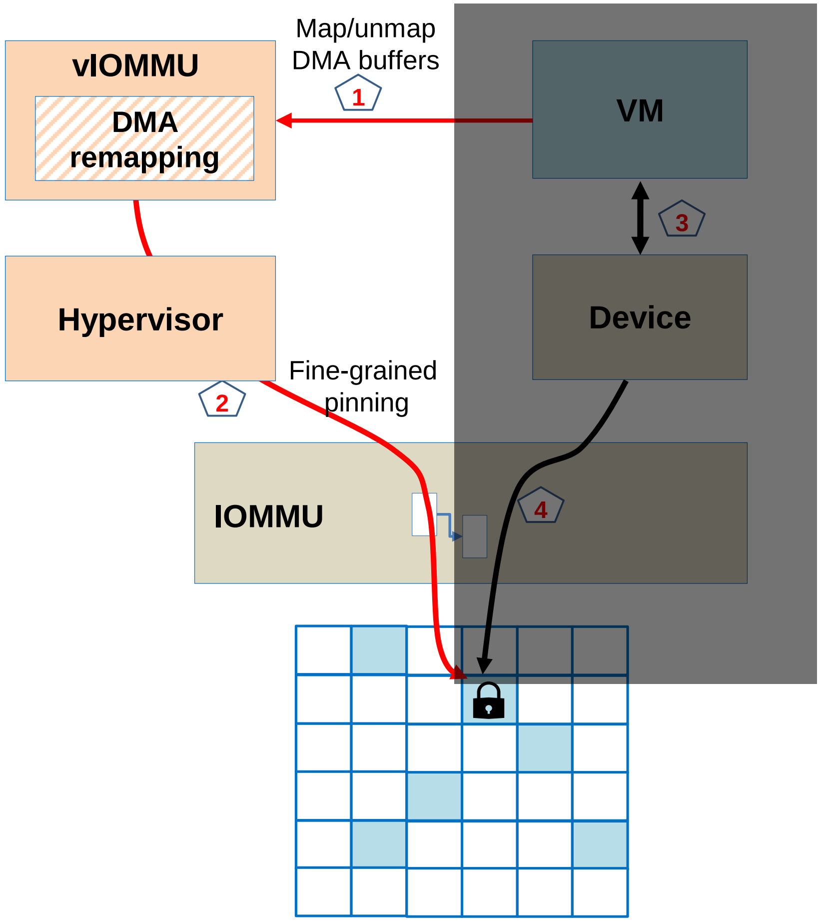
- Greatly reduced memory utilization:
  - ➢ Prevent many memory optimizations (overcommitment, late allocation, swap, etc.).



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 of vIOMMU

- 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.

- Virtual DMA remapping is not used by most guest Oses.

- Users may opt in when security requirement is over performance concern. E.g.,
  - when an untrusted device is plugged in.
  - when a device is assigned to userspace.

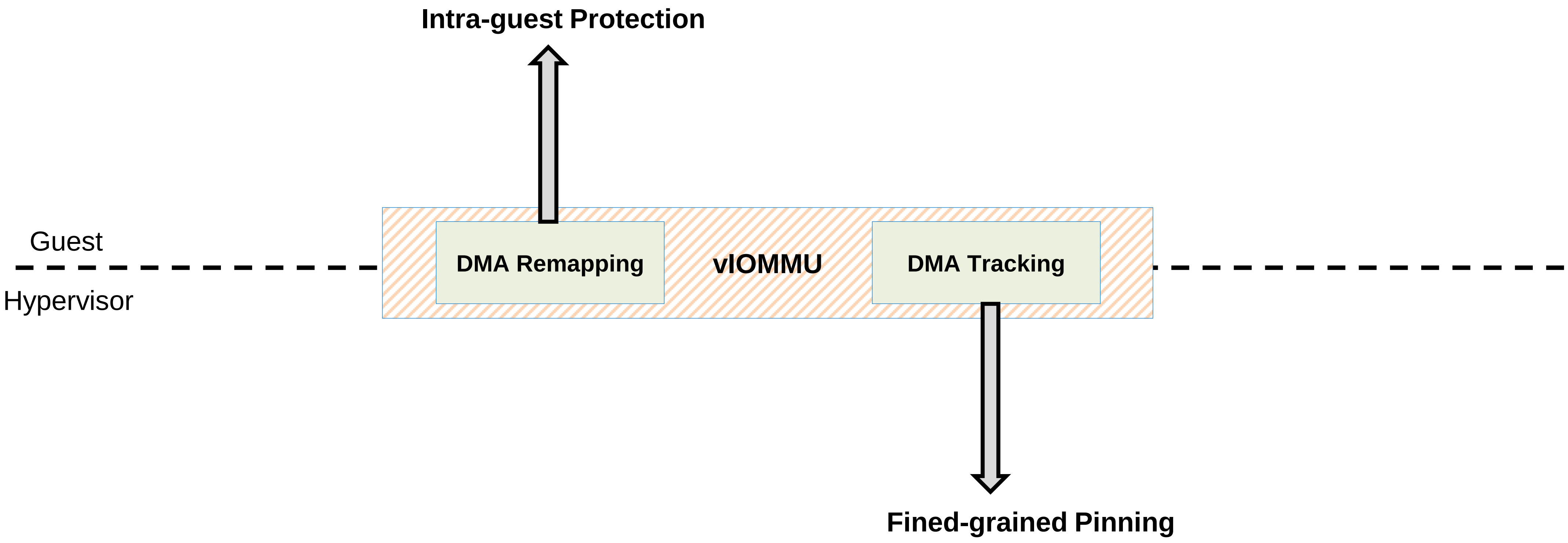
Motivation

- 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.

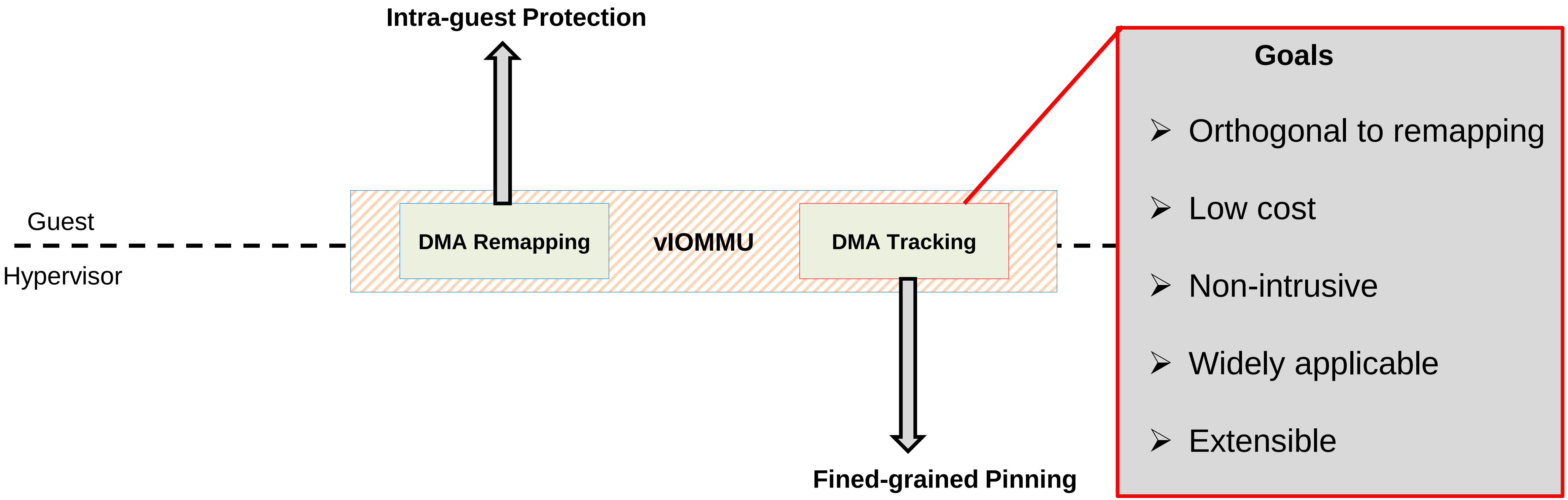
Motivation

- Decouple DMA tracking and DMA remapping in vIOMMU.



Motivation

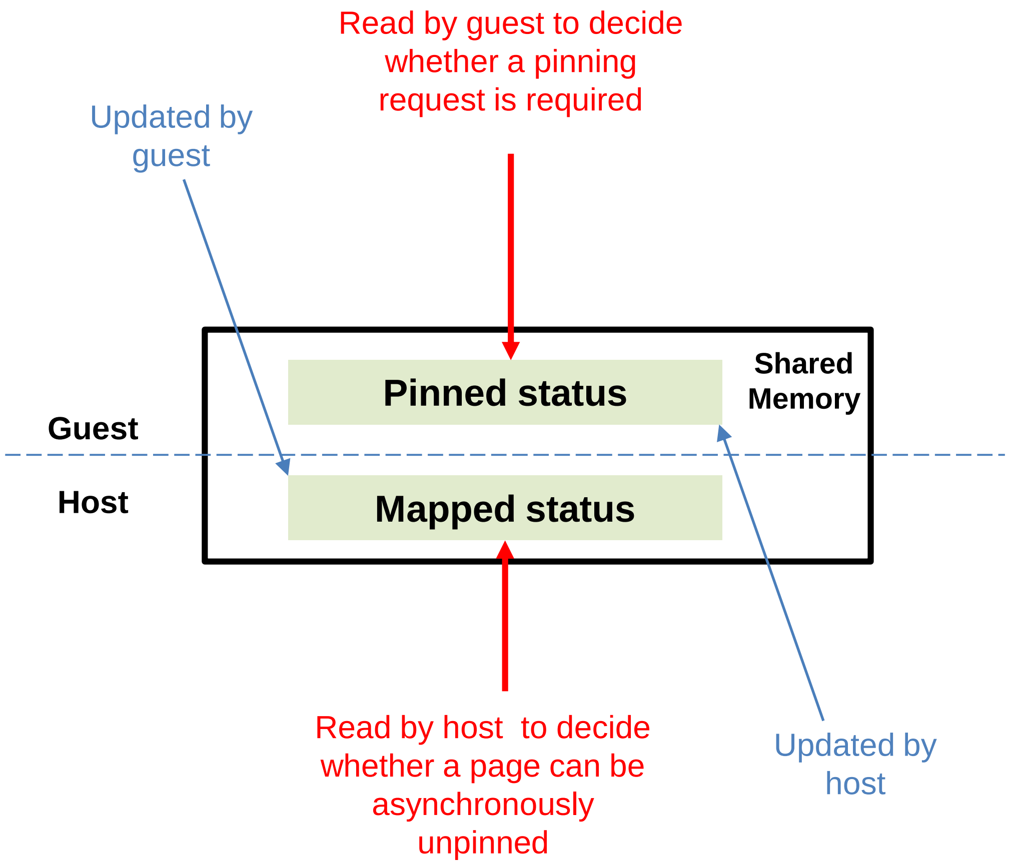
- Decouple DMA tracking and DMA remapping in vIOMMU.



Cooperative DMA Buffer Tracking

- Bi-directional shared DMA buffer information
  - To guest – whether a page is pinned by the host.
  - To host – whether a page is mapped by guest DMA API.

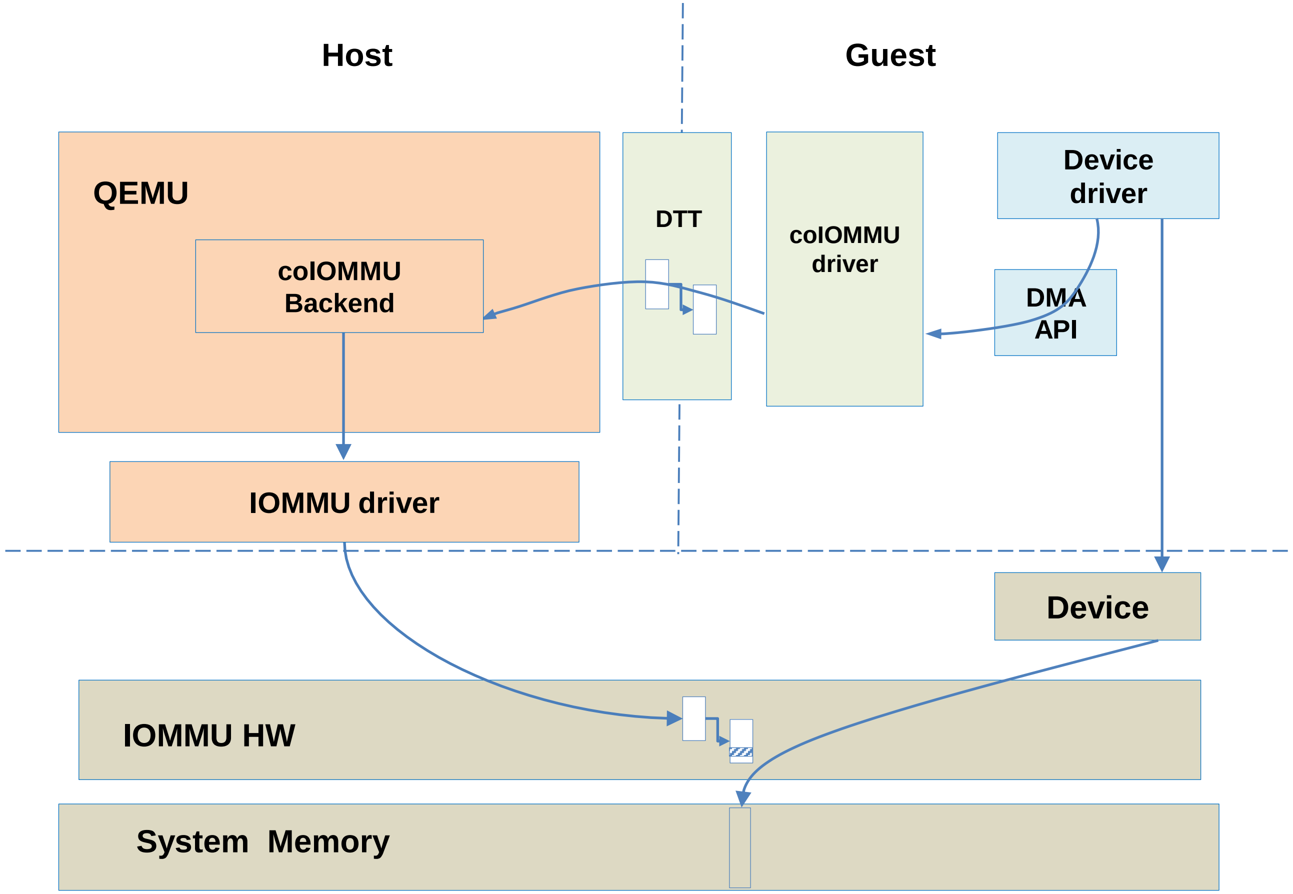
- A lightweight tracking interface for fine-grained pinning, when guest DMA remapping is disabled.
  - Minimize VM-exits when mapping DMA pages.
  - Eliminate VM-exits when unmapping DMA pages.
  - Enable flexible host memory management policies.



coIOMMU: A Virtual IOMMU with **Cooperative DMA Buffer Tracking** in Direct I/O

coIOMMU Architecture

- DMA Tracking Table (DTT)
  - ➢ Holds shared DMA buffer info, e.g. the “pinned” / “mapped” status of GFNs.
- coIOMMU driver
  - ➢ PV extension in guest IOMMU driver, which hooks to guest DMA API layer.
  - ➢ Updates mapped status and checks “pinned” status for each GFN in DTT.
  - ➢ Sends page pinning request to backend.
- coIOMMU backend
  - ➢ Handles page pinning requests and updates DTT.
  - ➢ Asynchronously unpin guest pages based on the “mapped” status in DTT.

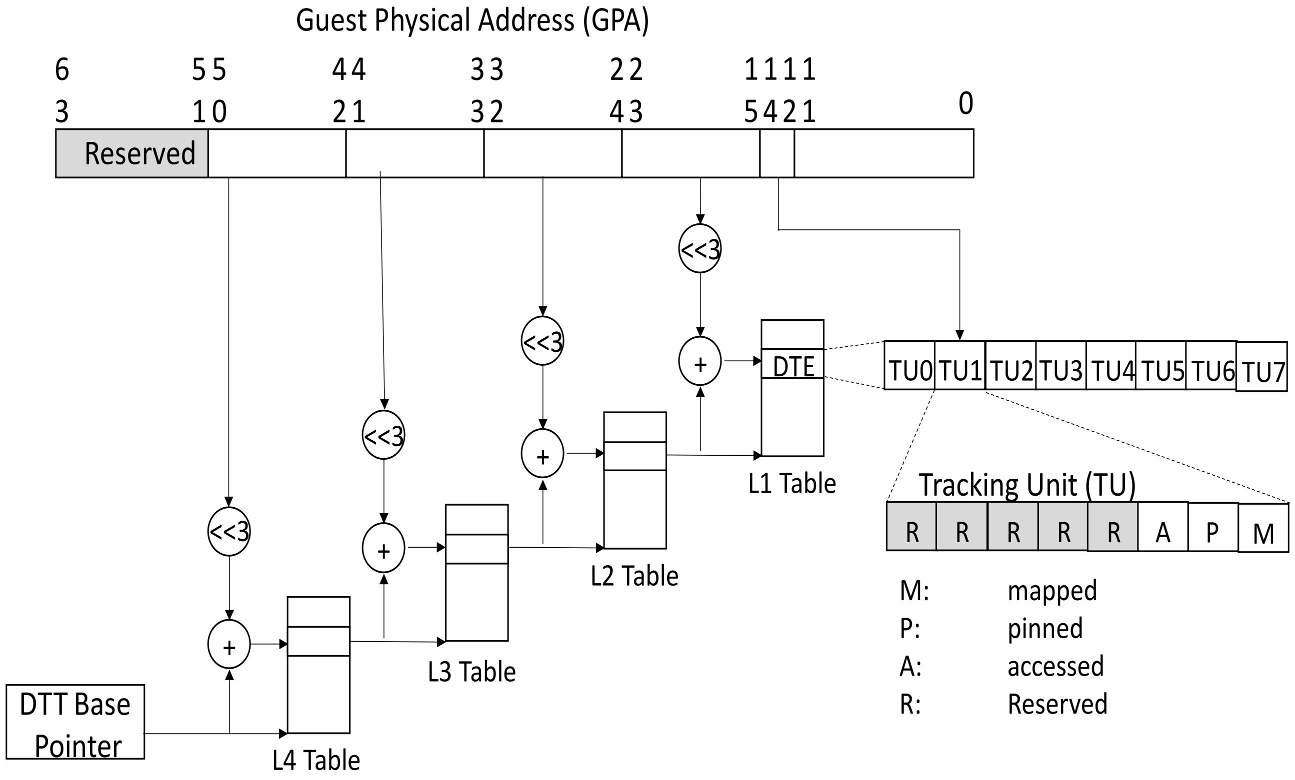


DMA Tracking Table (DTT)

- A multi-level paging structure
  - Shared between host & guest.
  - Indexed by GFNs.

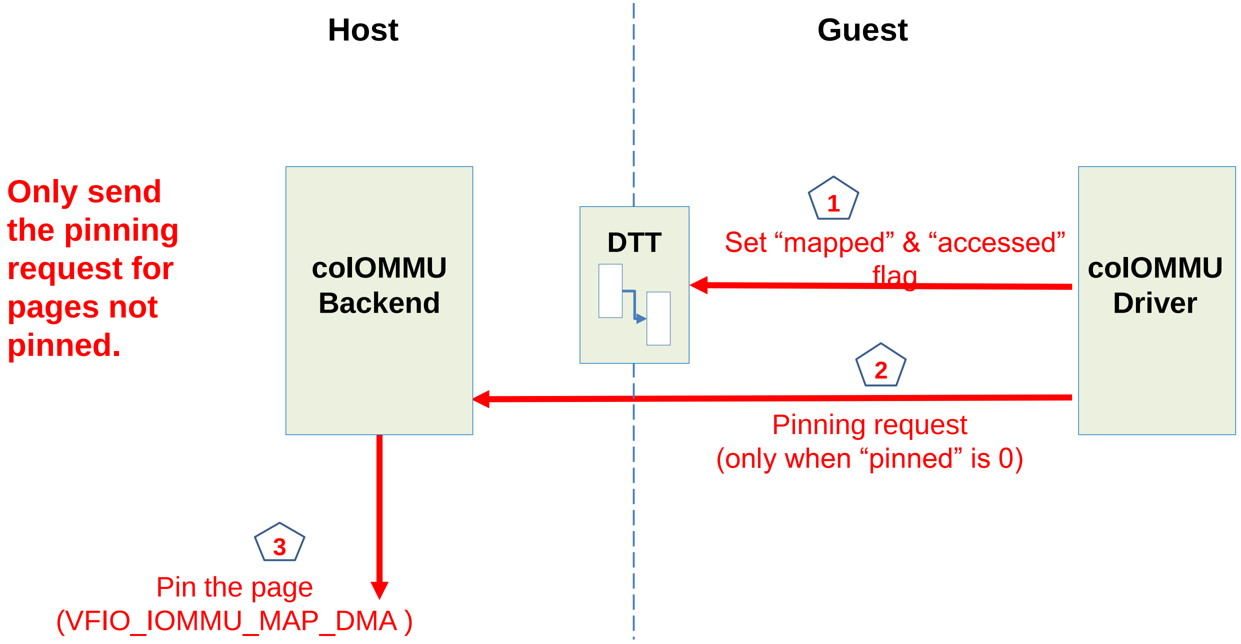
- TU - Tracking Unit for each GFN
  - ‘M’ (mapped) – set/cleared by guest.
  - ‘P’ (pinned) - set/cleared by host.
  - ‘A’ (accessed) – set by guest, cleared by host.

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



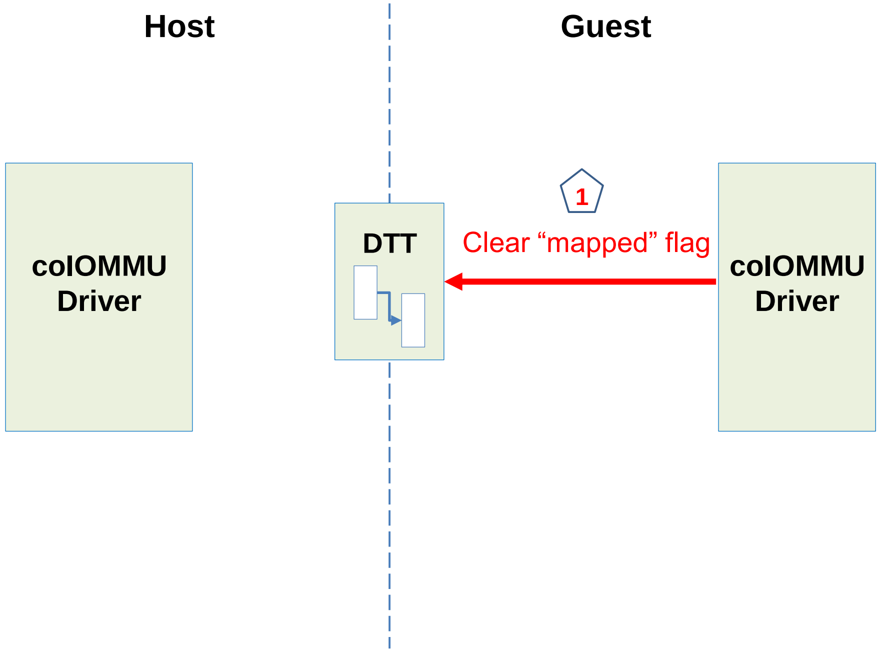
Precise pinning

Guest DMA map



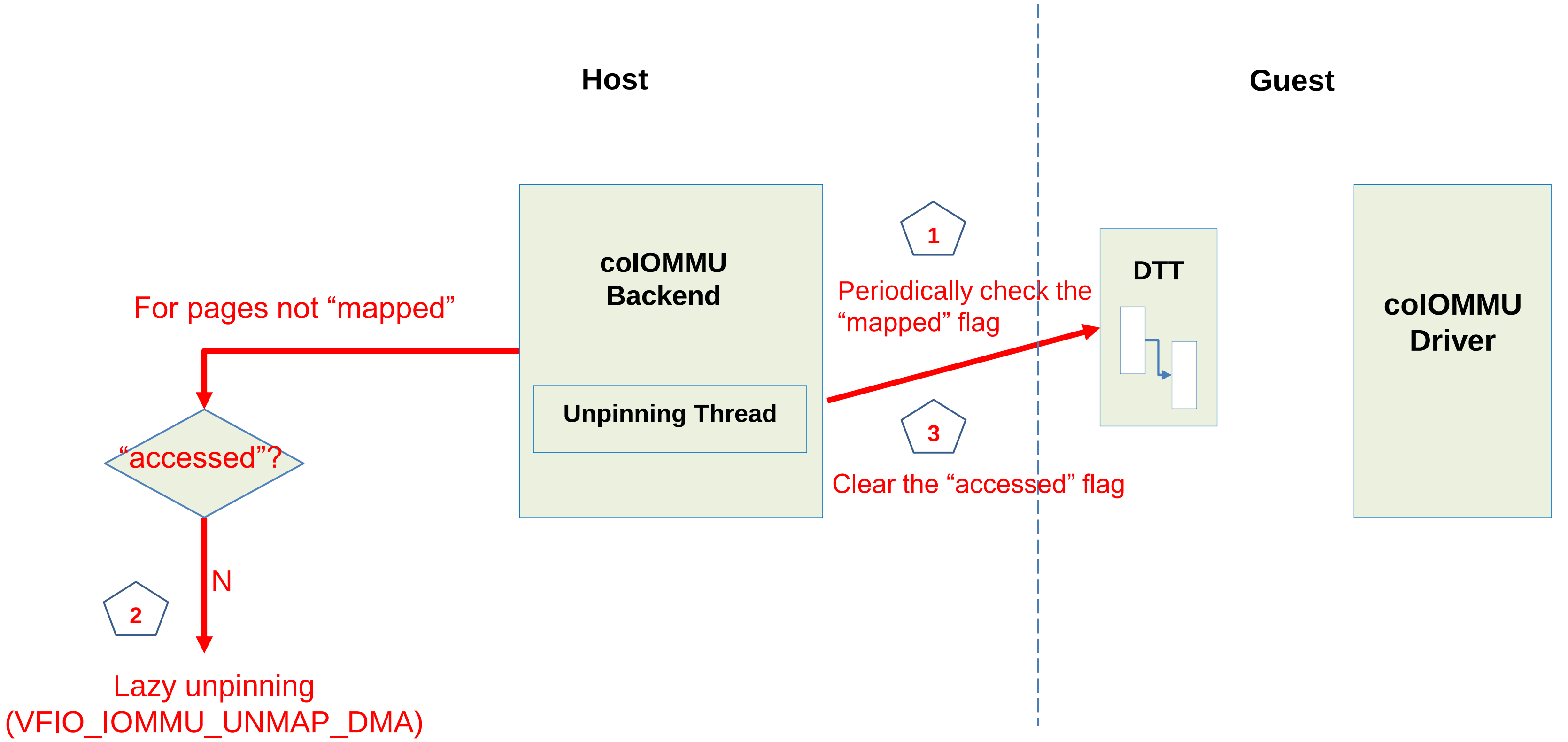
Up to 99.9992% VM exits can be avoided due to DMA buffer locality!

Guest DMA unmap



No VM exits at all!

Lazy Unpinning



1. Asynchronously unpin the pages that are no longer “mapped”.
2. LRU based unpinning based on the “mapped” and “accessed” status.
3. Unpinned pages are reclaimable.

DMA Tracking vs. DMA Remapping

- When DMA remapping is not used by guest (the majority case)
  - DMA tracking is an efficient solution to achieve fine-grained pinning.

- When DMA remapping is not always enabled
  - Guest may enable DMA remapping 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 case, DMA tracking helps provide a reliable way for fine-grained pinning.

- When DMA remapping is always enabled for all devices
  - Optional, no observable overhead.

Implementation

- POC done by extending virtual VT-d:
  - Guest Intel VT-d driver: ~900 LOC.
  - QEMU: ~700 LOC.

- coIOMMU concept can be applied in
  - Emulated IOMMUs.
  - Para-virtualized IOMMUs.

- Plan to upstream based on virtio-iommu.
  - New interfaces needed.
  - Other logics(guest DMA tracking, host unpinning etc.) can be reusable for different vIOMMU.

		New/Changed LOC
Guest	**Intel VT-d driver**	832 new
		47 changed
Host	**QEMU**	683 new

LOC of Previous POC

Upstream Considerations

- New interfaces in virtio-iommu
  - Feature negotiation - VIRTIO_IOMMU_F_PIN_PAGE (rely on VIRTIO_IOMMU_F_BYPASS).
  - Base address of a device bitmap - bit indexed by BDF, to indicate if the device is an assigned one.
  - Base address of DTT.
  - virtio-iommu request: virtio_iommu_req_pin.

- Lazy unpinning
  - A separate QEMU thread to perform the lazy unpinning periodically.
  - The unpinning interval can be manually configured, and an adaptive interval may be more desirable.
  - Current unpinning policy is LRU based. More policies can be examined.

Upstream Considerations

- Guest cooperation limitations
  - When creating a guest, the host has no idea if coIOMMU will be enabled by the guest later. Same issue exists in current vIOMMUs.
  - Guest BIOS may use direct I/O.
  - A selfish guest may choose to deliberately report fake DMA pages. A quota mechanism can be applied.

- Huge page mappings
  - DTT currently only tracks guest pages in 4KB granularity.
  - Backend can be optimized to conduct huge page pinning, however, will complex the lazy unpinning.
  - Most guest DMA workloads are not using huge page mapping(with exception of GPU workload).

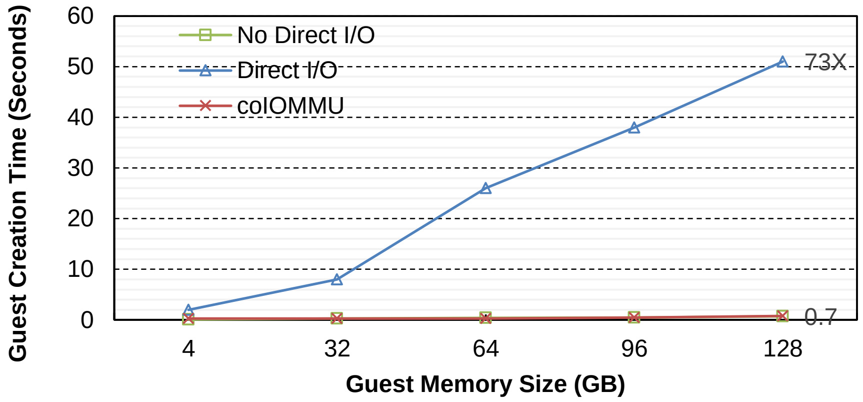
Upstream Considerations

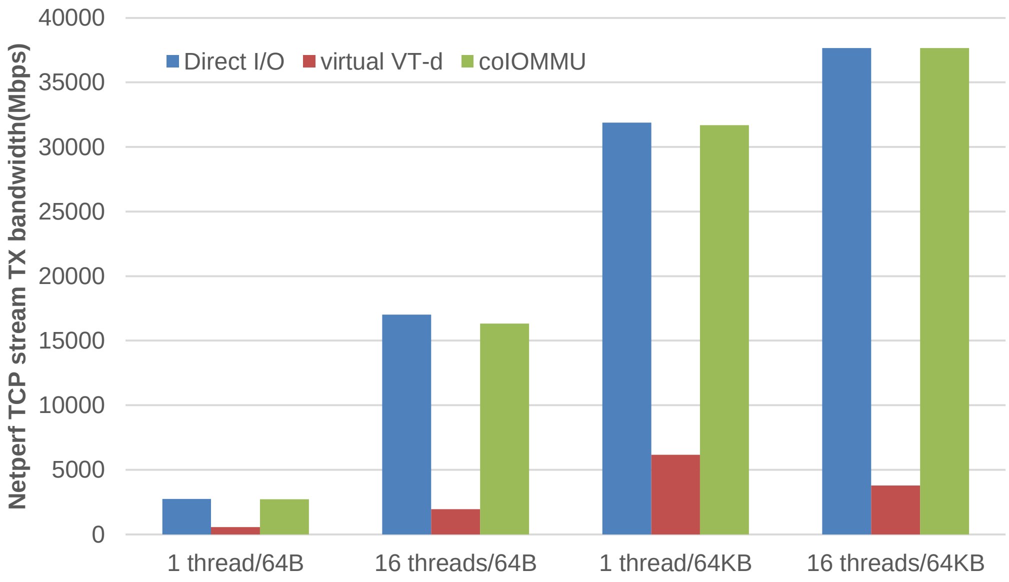
- Sub-page mappings
    - Multiple DMA buffers may co-locate in the same 4KB guest page (e.g. network packets).
    - Maintain a "map count" in each DTT entry.

- SVA capable devices
    - For SVA workloads, on-demand pinning is already implied by the support of IOMMU page fault.
    - However, typical SVA capable devices need to support mixed workloads(with SVA and non-SVA workloads), and global configuration data structures which are not faultable.

Performance Evaluation

- Measured on assigned devices e.g. 40Gbps NIC/NVMe SSD/Intel GPU.
- All benchmarks show near to 100% performance compared with direct I/O without vIOMMU.
- Much fewer pinned guest pages. E.g. peak pinned pages only ~1.3% of total guest memory(32GB).
- Much reduced VM creation time.
- Detailed environment and performance data at https://www.usenix.org/conference/atc20/presentation/tian.





Summary

- 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 different protection usages;
  - ➢ can be easily applied in various vIOMMU implementations.

- Call for suggestions! ☺

Q&A