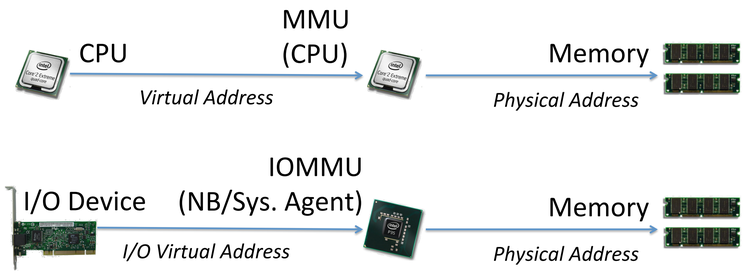
# vIOMMU: Efficient IOMMU Emulation*

Nadav Amit, Muli Ben-Yehuda, Dan Tsafrir, Assaf Schuster

* Full paper will appear in USENIX ATC'11 proceedings

## IOMMU

*I/O Memory Management Unit* (IOMMU) is similar to MMU, except it translates device accesses to memory instead of CPU accesses.




## Approach: IOMMU Emulation

Exposing an emulated IOMMU to guest allows:

**1. Guest can configure IOMMU**
Changes reflected in physical IOMMU

**2. Memory overcommit can be enabled**
Implicit notification of each allocated I/O buffer allows to pin only I/O buffers

**3. Inter-guest protection is kept intact**
I/O buffers are pinned to physical memory so it cannot be reallocated to other guests

However, trap & emulate (samecore) emulation is expensive. IOMMU (un)mappings are lengthy operations in both native-mode and emulation.

## Related Work

Protection strategies [Willmann08]
IOMMU paravirtualization [Yassour10]
Paravirtualization offloading [Kumar07]
Virtualization Polling Engine [Liu09]

## Direct Device Assignment

Direct device assignment is an I/O usage model in virtualization, which allows the guest to use the I/O device without the hypervisor intervention.

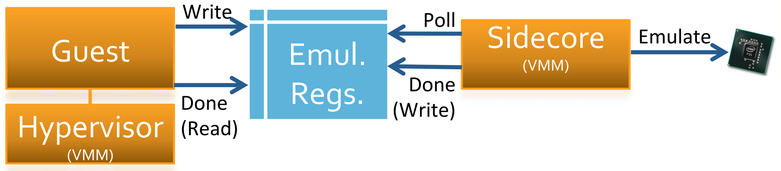
Direct device assignment requires IOMMU for protection of the other guests and the hypervisor.

This is the best performing I/O usage model, and does not require guest OS modifications.

## Efficient IOMMU Emulation

Our solutions:

1. Trap & emulate is expensive
   → Emulate in a *sidecore* (no traps):




2. IOMMU mapping layer optimizations:
   - Linux: defer IOTLB invalidations (*deferred*)
   - Reuse existing mappings to given PFN (*shared*)
   - Asynchronous IOTLB invalidations (*async*)
   - Defer unmappings and reuse (*optimistic teardown*)

## Results Summary (VT-d, KVM w/10GbE)

| Setting | Secure | Relaxed (Linux Default) | Opt. Teardown (Patched) |
|---|---|---|---|
| Bare-metal | 43% | 91% | 100% |
| Samecore | 10% | 11% | 82% |
| Sidecore | 30% | 49% | 100% |

## Contribution

Direct Device Assignment Disadvantages:

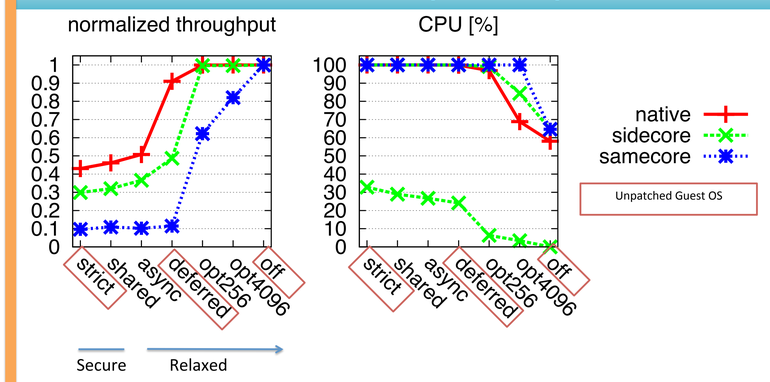
1. It requires pinning all of the guest's pages, thereby disallowing *memory overcommitment*.
2. It exposes the guest's memory to buggy drivers.

Our Main Contributions:

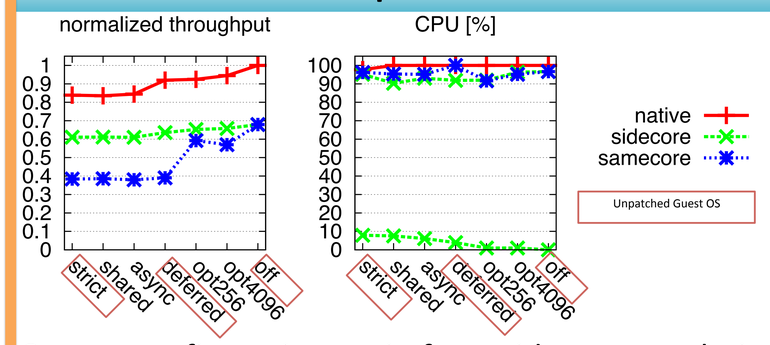
1. Overcoming the above challenges through IOMMU emulation.
2. Emulation made performant with 2 novel optimizations: sidecore and optimistic teardown

## Results (VT-d, KVM w/10GbE)

### TCP Stream (NetPerf)




### Apache




Secure configurations gain from sidecore emulation. Optimistic teardown improves performance with a modest security relaxation.