TECHNION
Israel Institute
of Technology

Utilizing the IOMMU Scalably

USENIX Annual Technical Conference 2015

Omer Peleg, Adam Morrison, Benjamin Serebrin[*] and Dan Tsafrir

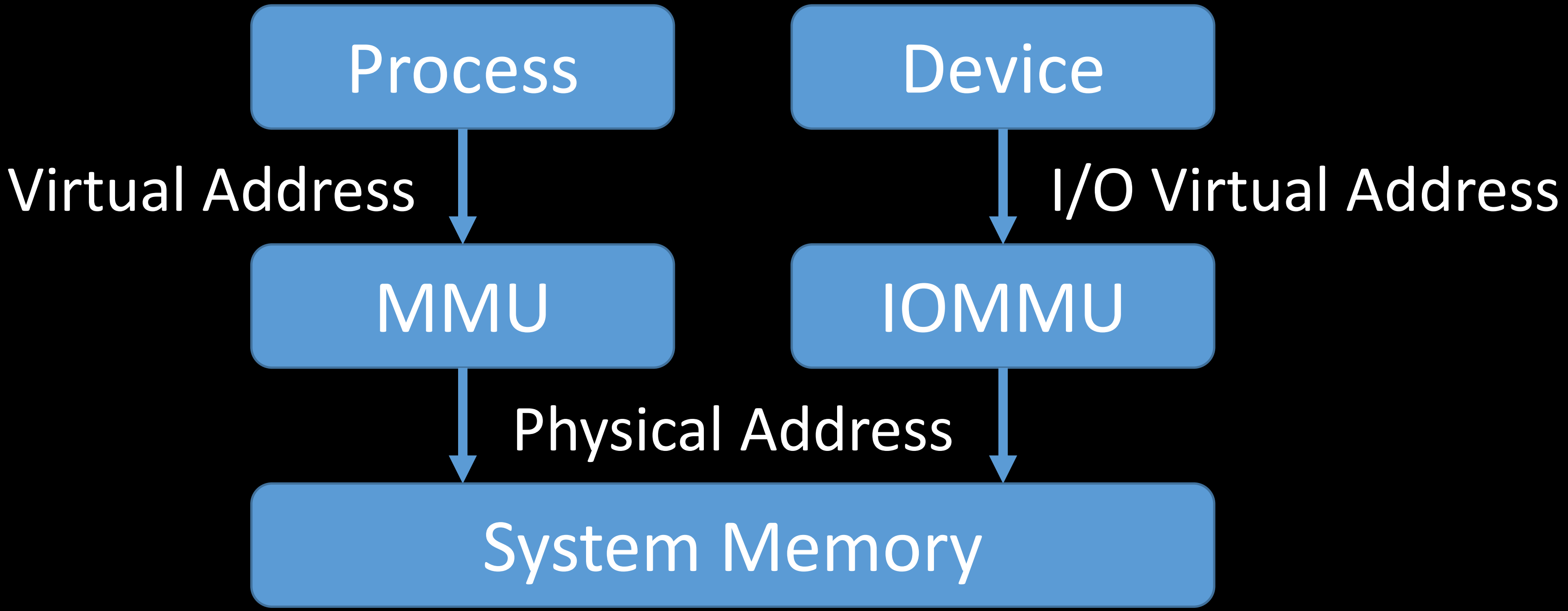
[*] Google

In This Talk

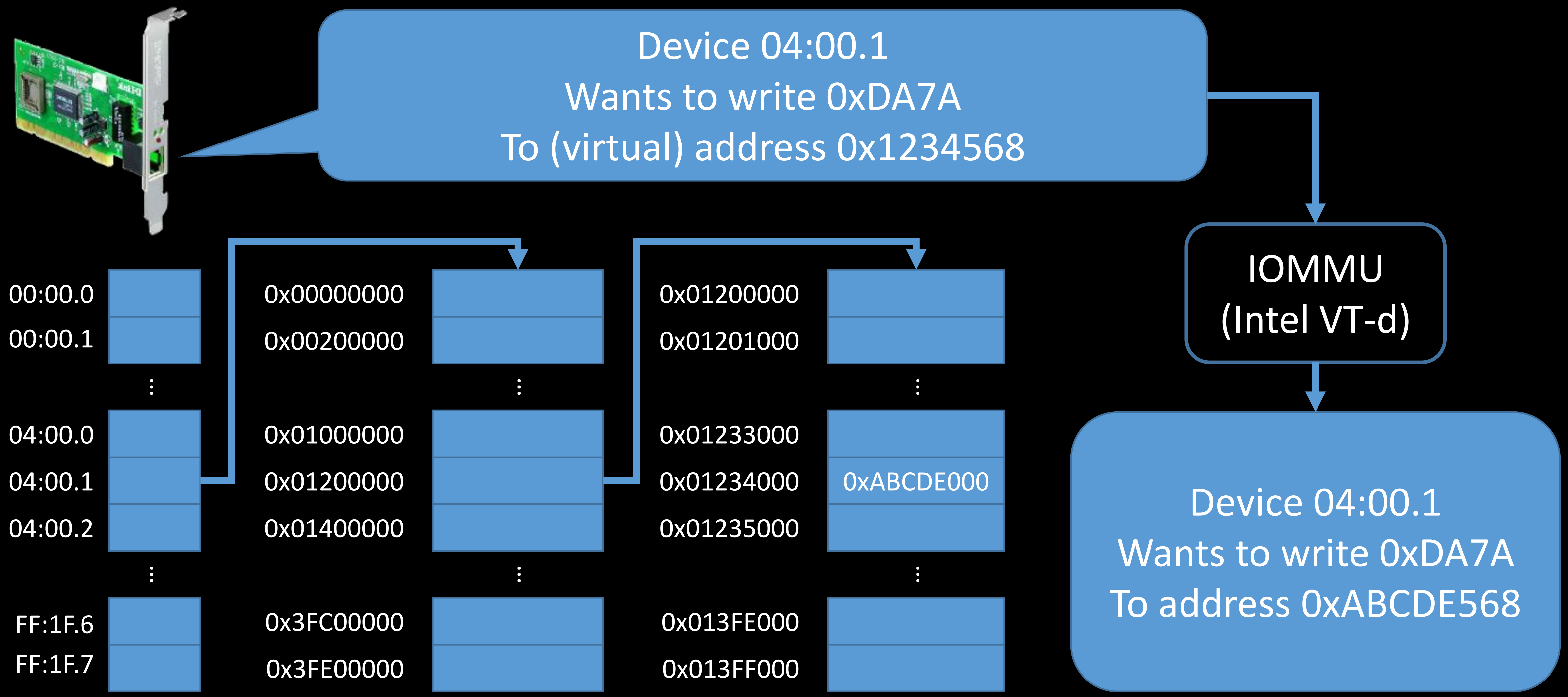
- IOMMU overview
- Main challenges to OSes
- Current solutions – they don’t scale
- Exploring scalable solutions

What is an IOMMU?

- Similar to MMU
- Translates DMA accesses



How Does it Work?



What is the IOMMU for?

- Protecting the system from untrusted elements
  - MMU protects memory from processes
  - IOMMU protects memory from devices

What is the IOMMU for?

- Protecting the system from untrusted elements

NEWS

Researcher creates proof-of-concept malware that infects BIOS, network cards

New Rakshasa hardware backdoor is persistent and hard to detect, researcher says

– July 29th 2012 (Computerworld)

Someone (probably the NSA) has been hiding viruses in hard drive firmware

By Russell Brandom on February 16, 2015 05:03 pm Email @russellbrandom

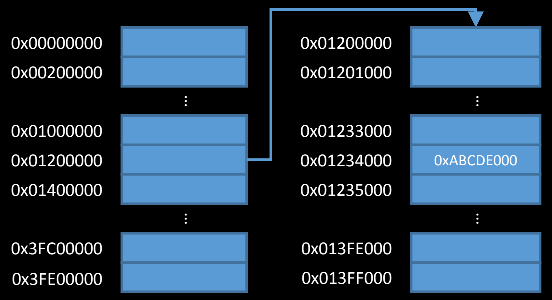
– February 16th 2015 (The Verge)

Where MMU and IOMMU differ

MMU (process)

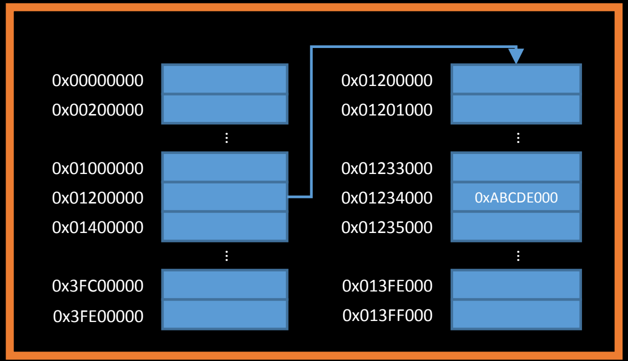
malloc/free

mmap/munmap



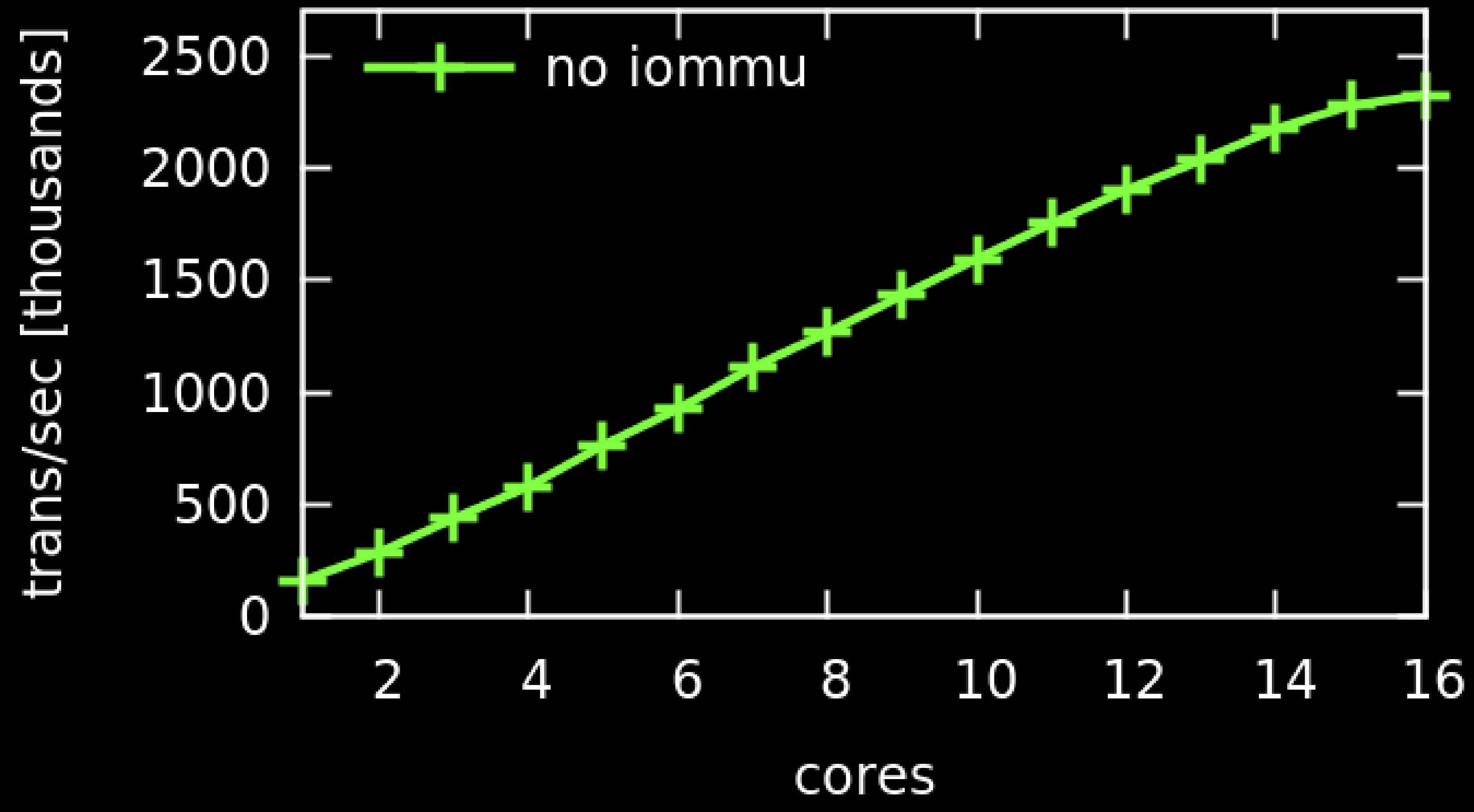
IOMMU (device driver)

dma_map/unmap



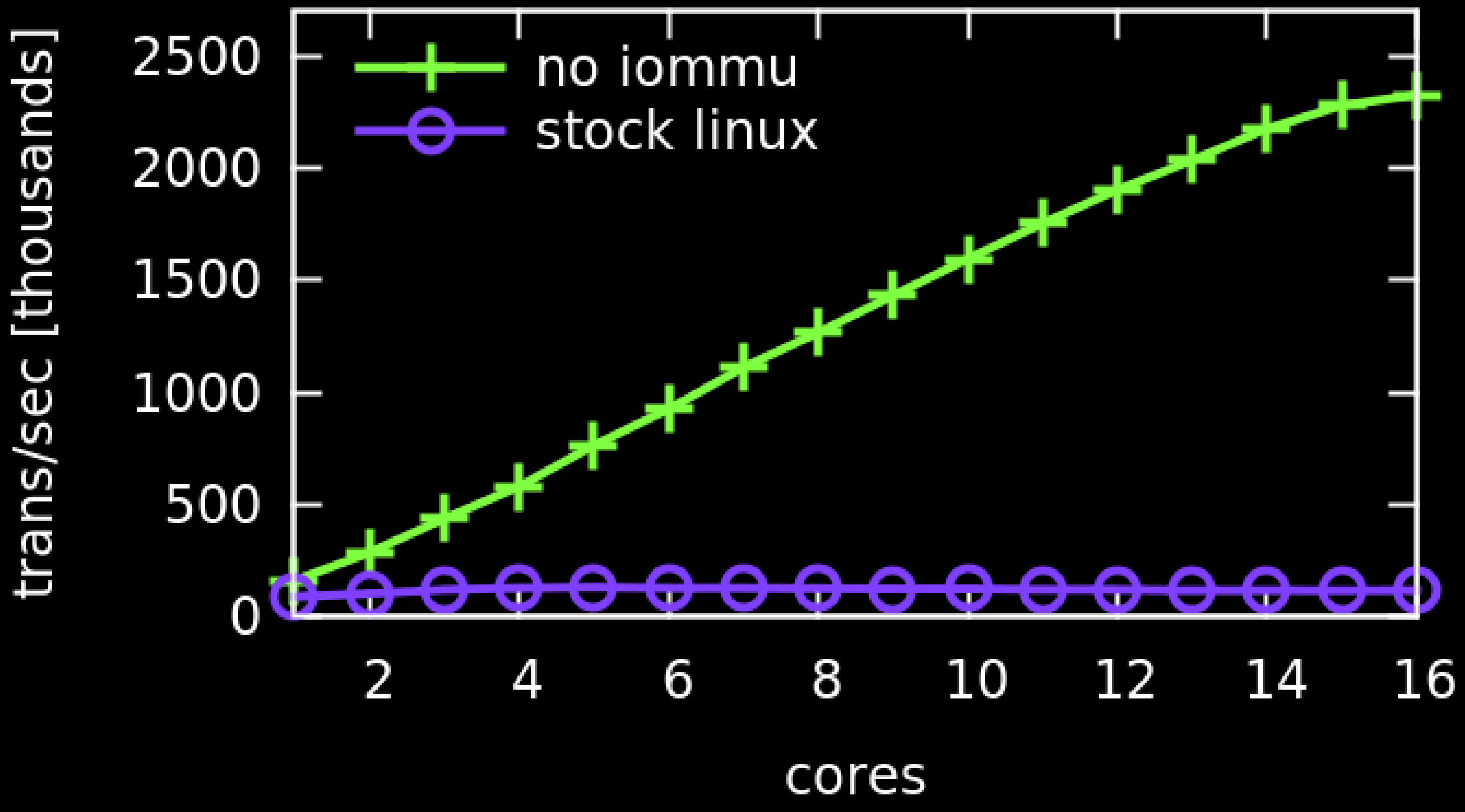
IOMMU Limits Performance?

Aggregate throughput – 270 Netperf TCP Request/Response



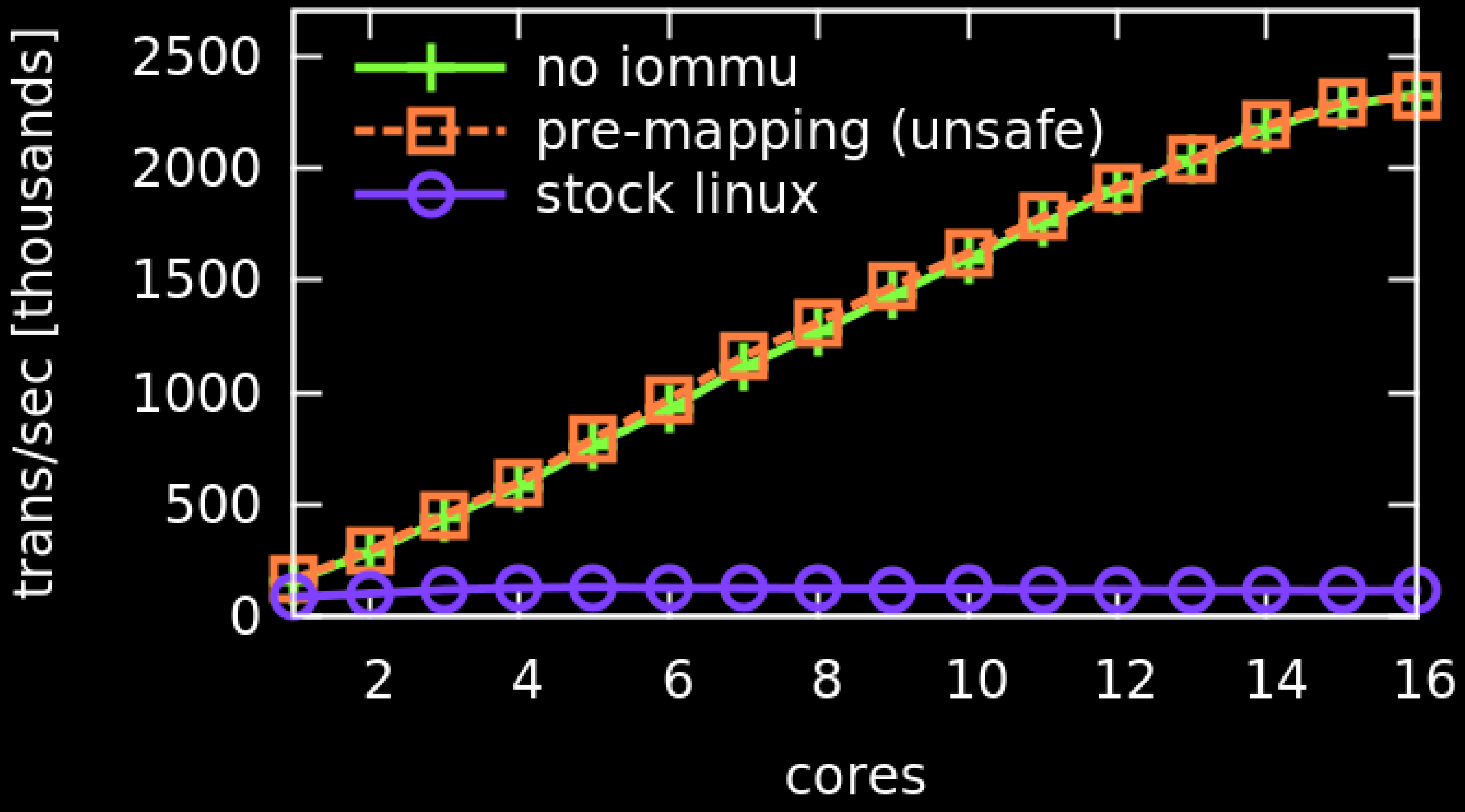
IOMMU Limits Performance?

Turning IOMMU on in Linux is prohibitive



IOMMU Limits Performance?

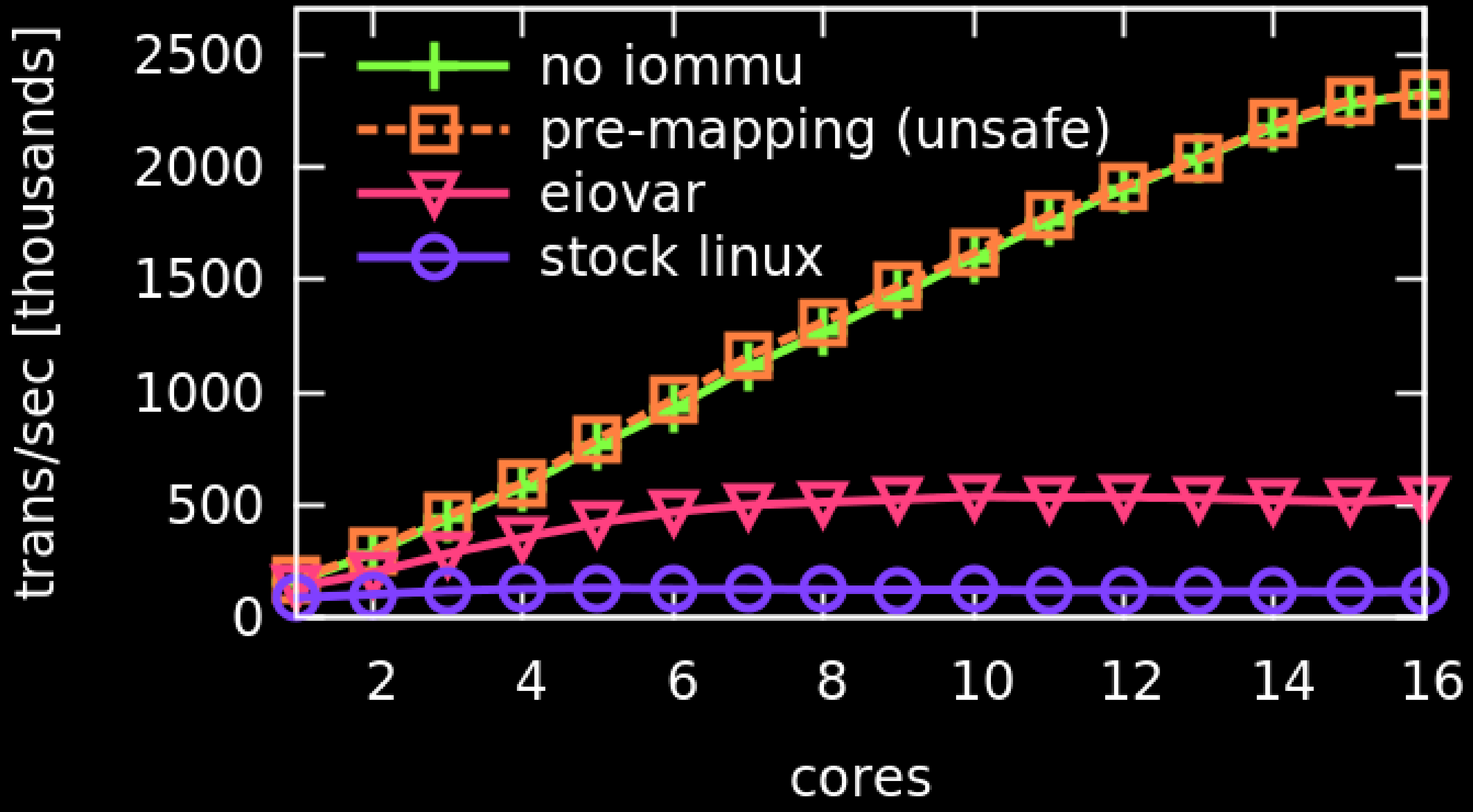
Meltdown is not due to hardware, though



IOMMU – State of the Art

- EiovaR – Efficient IOVA allocatoR
- Malka et al., FAST ‘15
- Baseline for our talk
- Optimized IOMMU single core performance

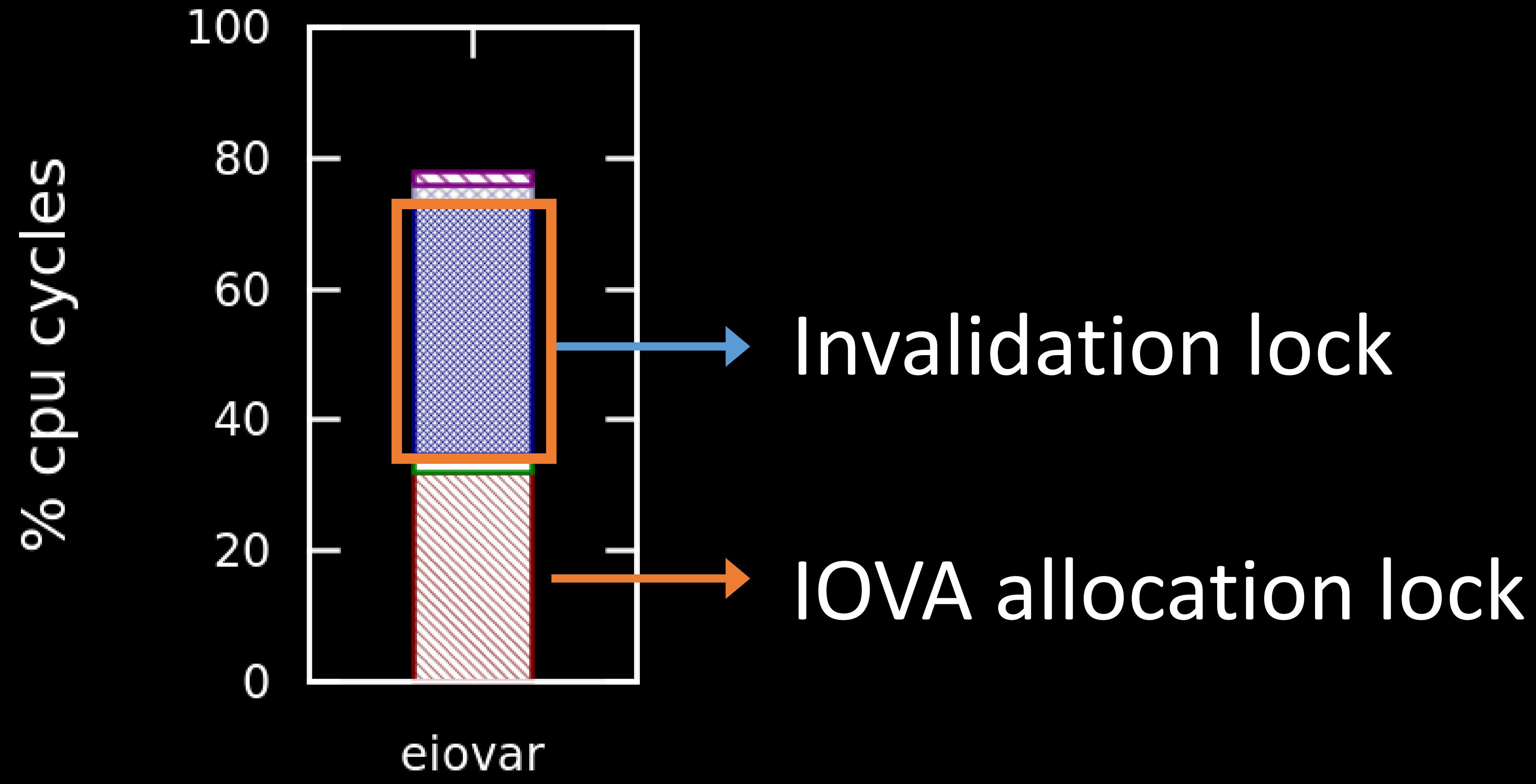
IOMMU – State of the Art



Our Contribution

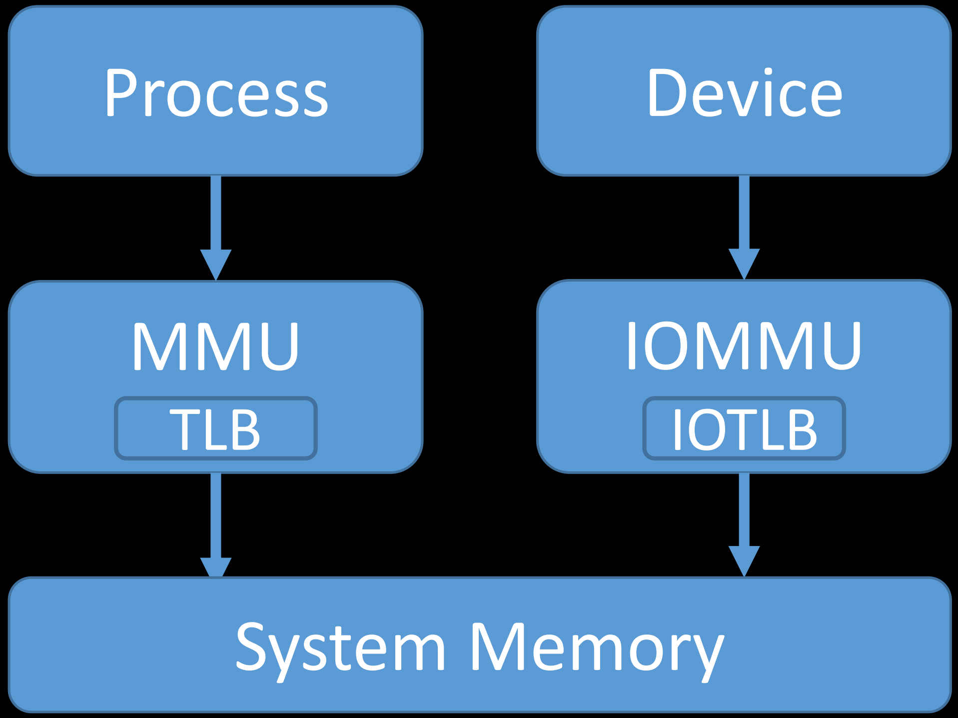
- Identify scalability bottlenecks
  - Linux, FreeBSD, OpenSolaris, Mac OS X
  - All have:
    - Globally locked IOVA allocation
    - Globally locked Invalidations
- Design and compare scalable solutions

EiovaR – Scalability (@16 Cores)



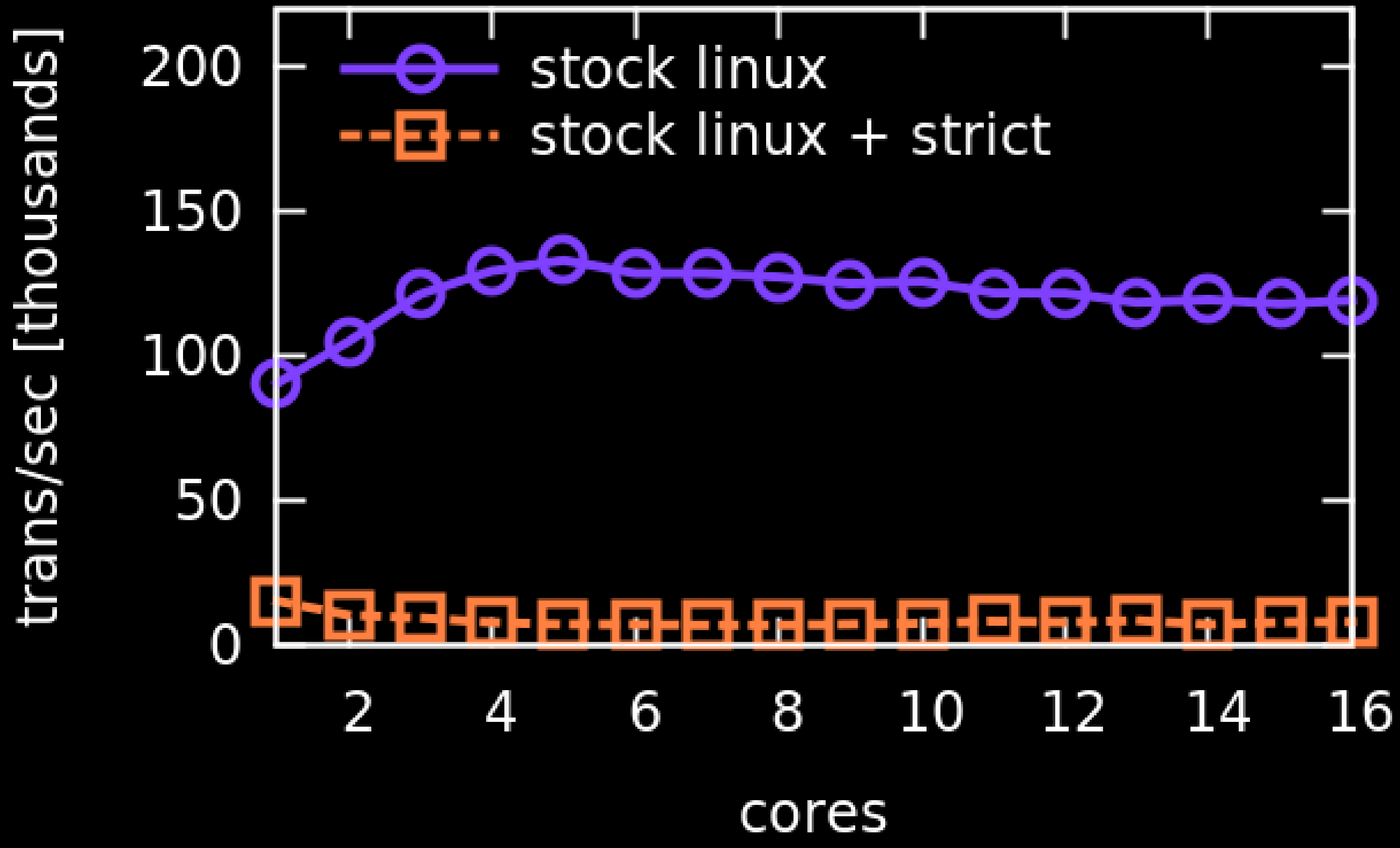
Invalidation Complicates Things

- IOMMU caches translations
- Invalidations needed
  - Before address reuse
  - For security



- Strict (invalidation on unmap) – too costly
  - Contention on invalidation interface

Linux – Strict Invalidation Cost



Linux – Deferred Invalidation

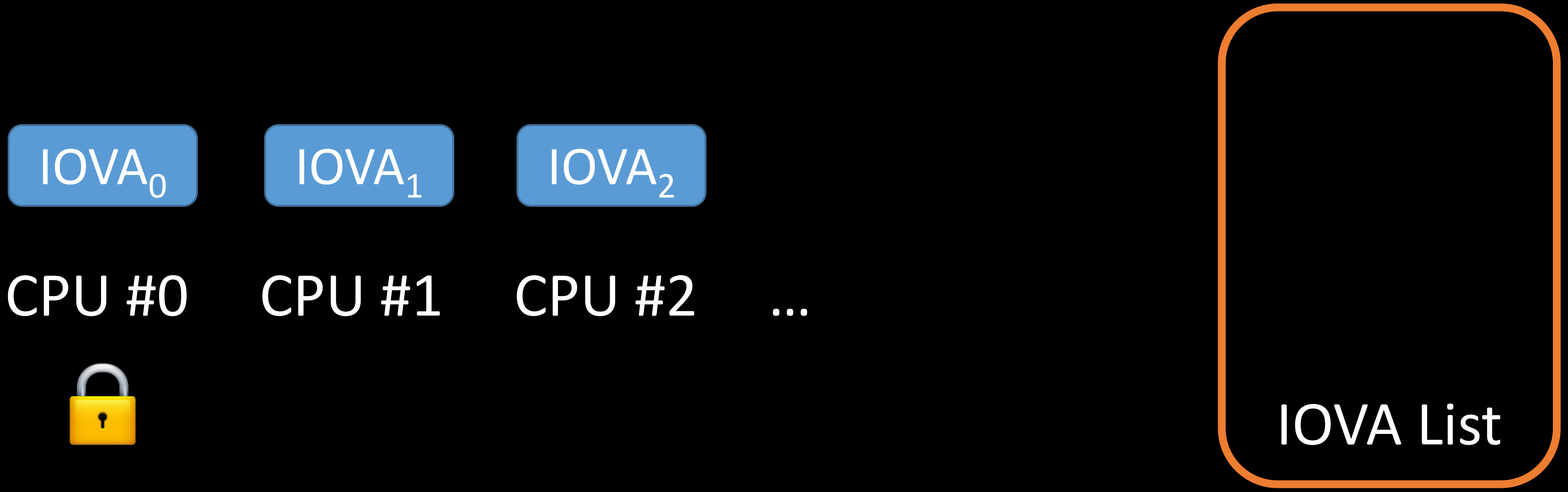
- Linux's default policy
- Batch (up to 250) invalidations
  - Invalidate IOTLB globally
  - Free batched IOVAs only after invalidation
- Creates a vulnerability window
  - Not a correctness problem, though

Deferred Invalidation - The Problem

- Linux saves IOVAs it will free upon invalidation
- In a globally locked data structure

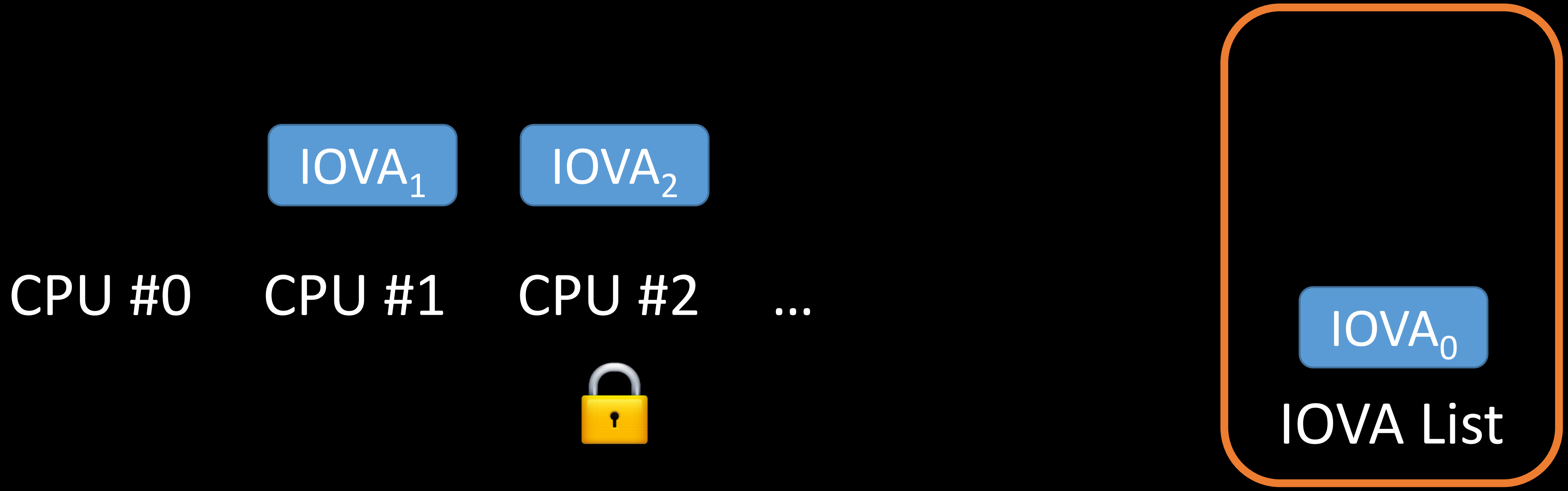
Deferred Invalidation - The Problem

- Linux saves IOVAs it will free upon invalidation
- In a globally locked data structure



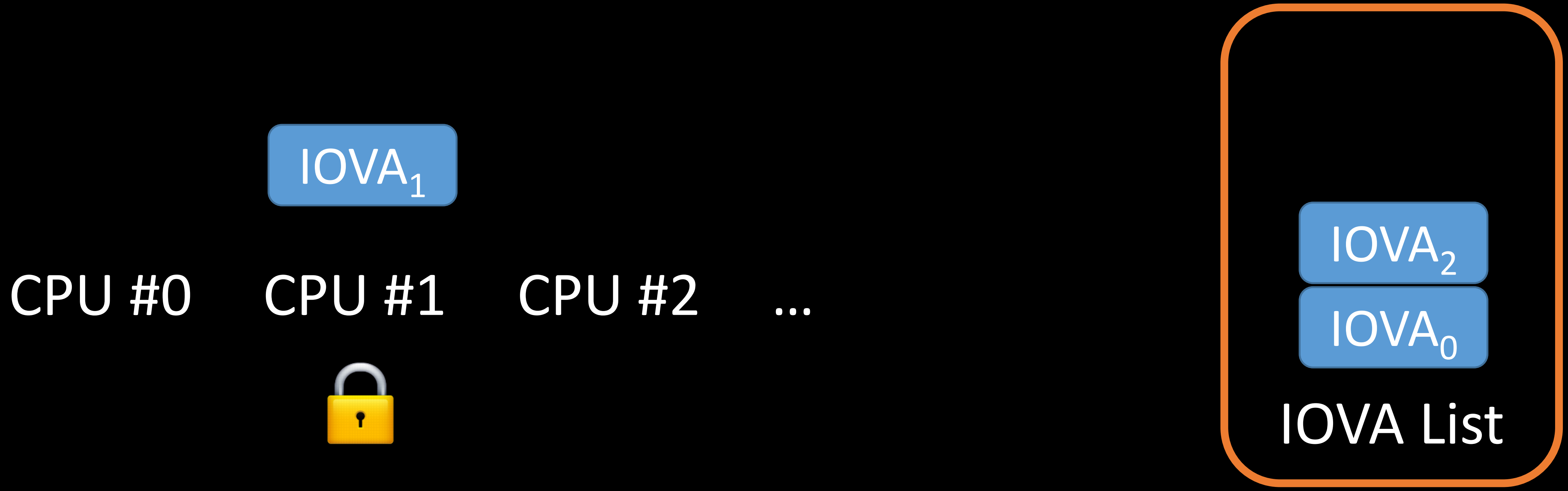
Deferred Invalidation - The Problem

- Linux saves IOVAs it will free upon invalidation
- In a globally locked data structure



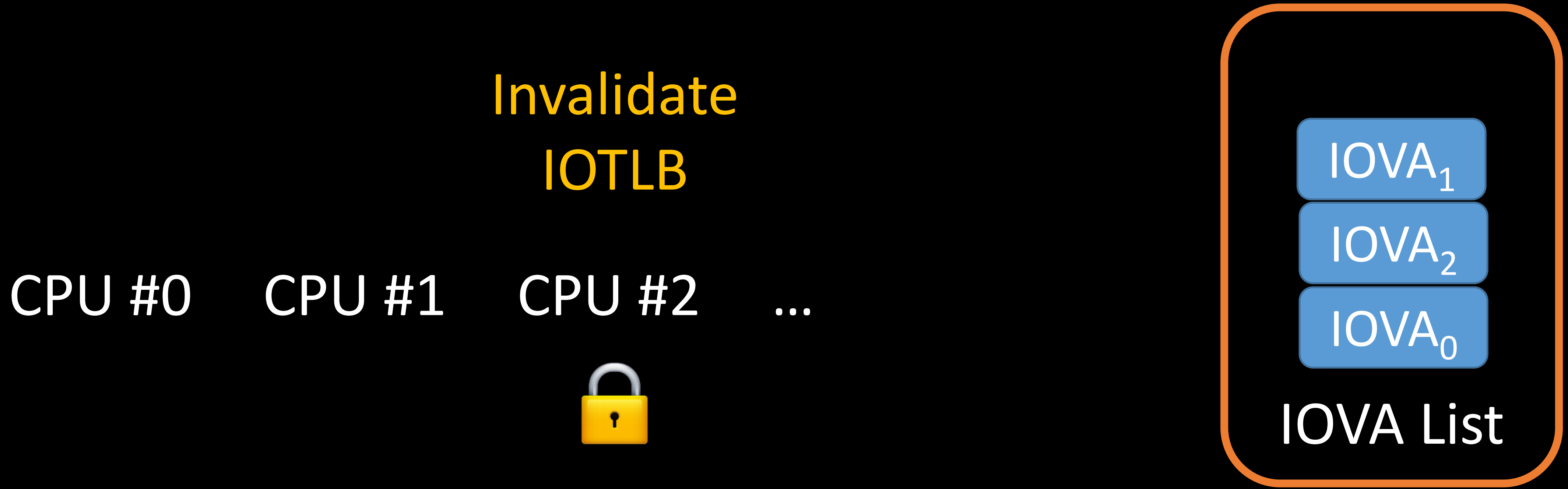
Deferred Invalidation - The Problem

- Linux saves IOVAs it will free upon invalidation
- In a globally locked data structure



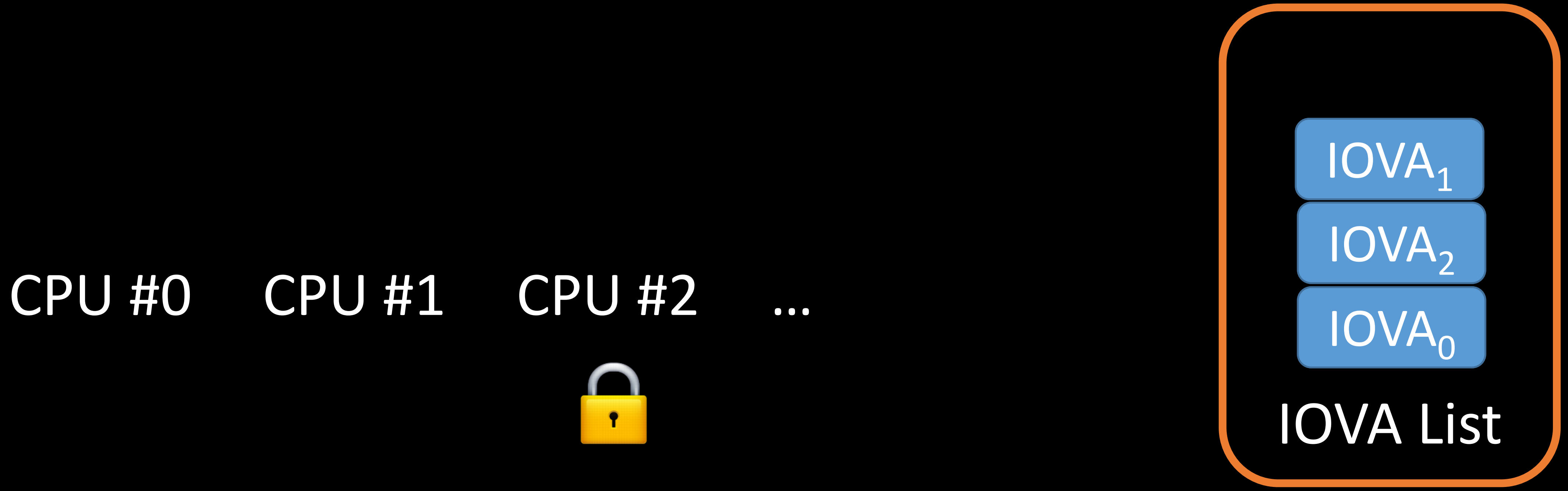
Deferred Invalidation - The Problem

- Linux saves IOVAs it will free upon invalidation
- In a globally locked data structure



Deferred Invalidation - The Problem

- Linux saves IOVAs it will free upon invalidation
- In a globally locked data structure



Deferred Invalidation - The Problem

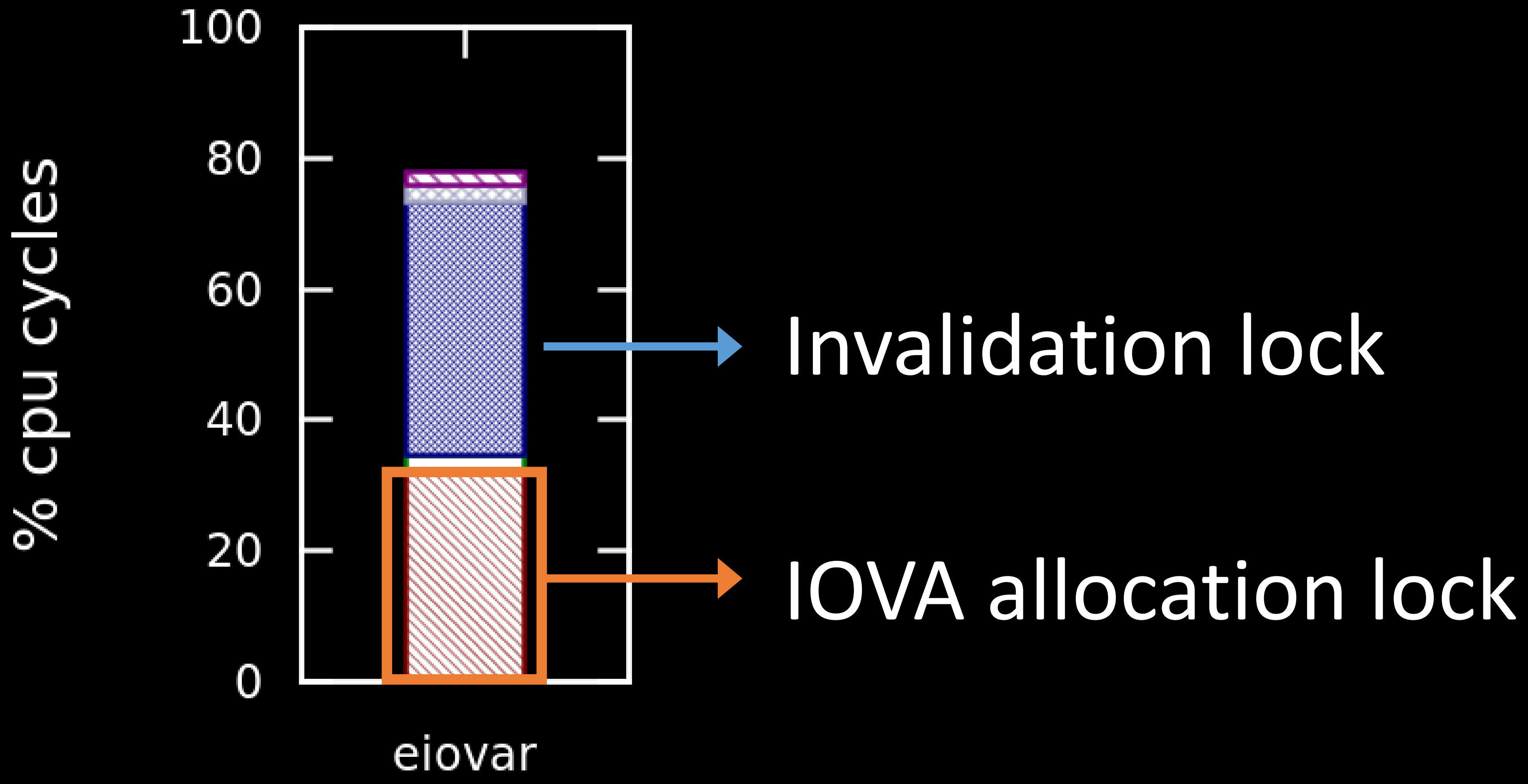
- Linux saves IOVAs it will free upon invalidation
- In a globally locked data structure
    - Contention

Solving Deferred Invalidation

- But prompt freeing of IOVAs is not significant!
- Use per-core deferred invalidation

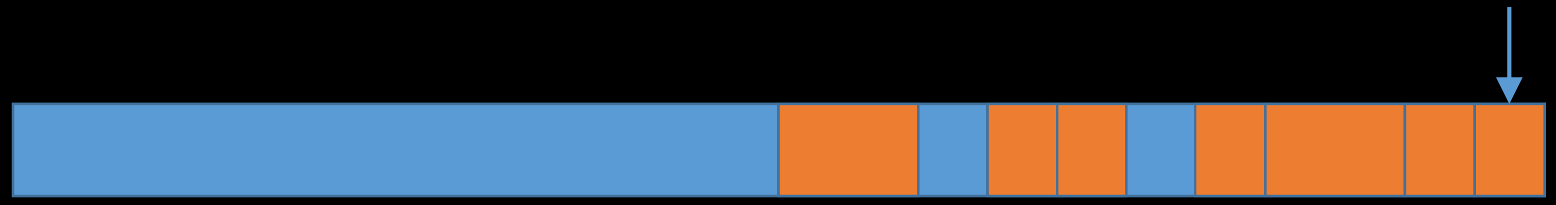
- Access to hardware still 250:1 vs strict
- Correctness: maintained

EiovaR – Scalability (@16 Cores)



Linux – IOVA Allocation

- Globally locked
- Finds first fit from top of virtual space
  - EiovaR does that in constant time



- ***Packs allocations in a bounded area***

Linux – Page Table Management

- Page table lock = BAD!
- Linux manages tables in parallel with no lock
- The price – page tables are never freed
- Good thing IOVA range is bounded

Solving IOVA Assignment

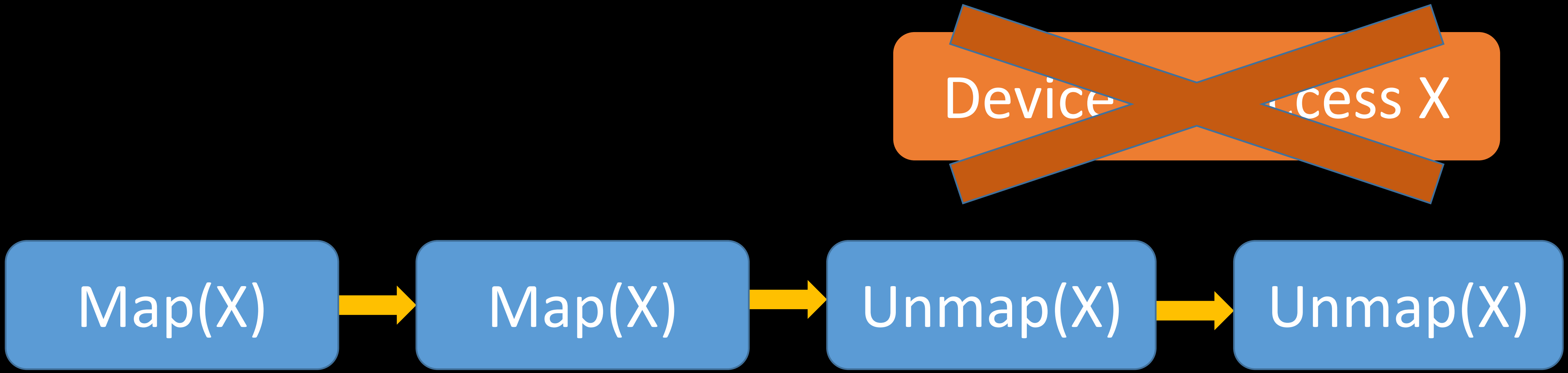
- IOVA assignment doesn’t scale
- We explore three different solutions

Solving IOVA Assignment #1 – Dynamic 1:1

- Do we even need an allocator?
  - Page being mapped already has an address
- Use physical address as virtual

Solving IOVA Assignment #1 – Dynamic 1:1

- Use physical address as virtual
- Reference count

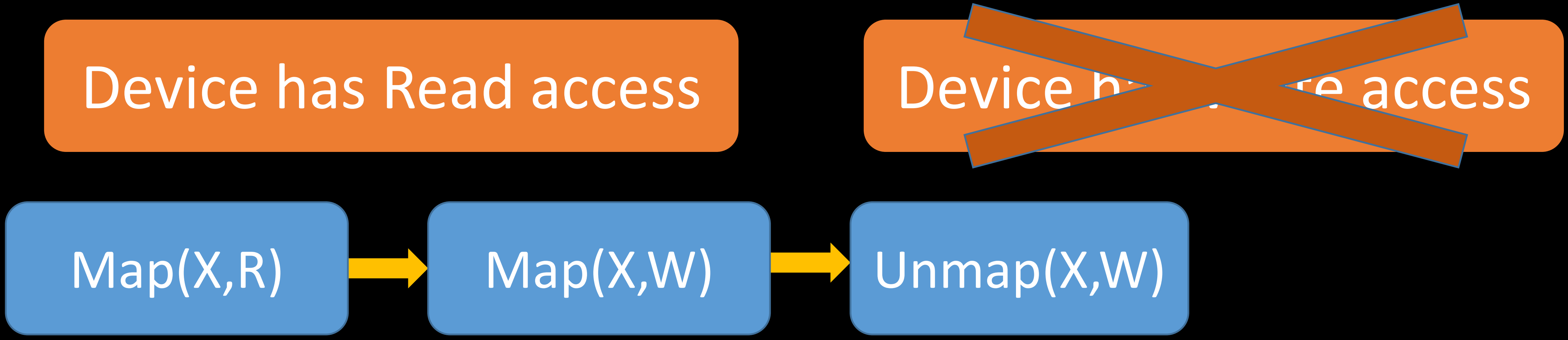


Solving IOVA Assignment #1 – Dynamic 1:1

- Use physical address as virtual
- Reference count
  - Use spare bits in page table entry

Solving IOVA Assignment #1 – Dynamic 1:1

- Use physical address as virtual
- Reference count
- Keep permissions accurate



Solving IOVA Assignment #1 – Dynamic 1:1

- Use physical address as virtual
- Reference count
- Keep permissions accurate
  - Separate virtual space by access rights

What is allocating an IOVA?

- Allocate range of virtual page numbers
- Allocating a unique range of integers

- Regular memory allocators allocate a range of bytes
  - Which have a range of unique addresses
  - Use the address range as an unique integer range
  - Disregard the memory

Solving IOVA Assignment #2 – IOVA-kmalloc

- Use existing, optimized, general purpose allocator
- For a *k* page range: *kmalloc(k)*
  - Use address as virtual page number
  - Completely disregard the actual memory

Solving IOVA Assignment #3 – Magazines

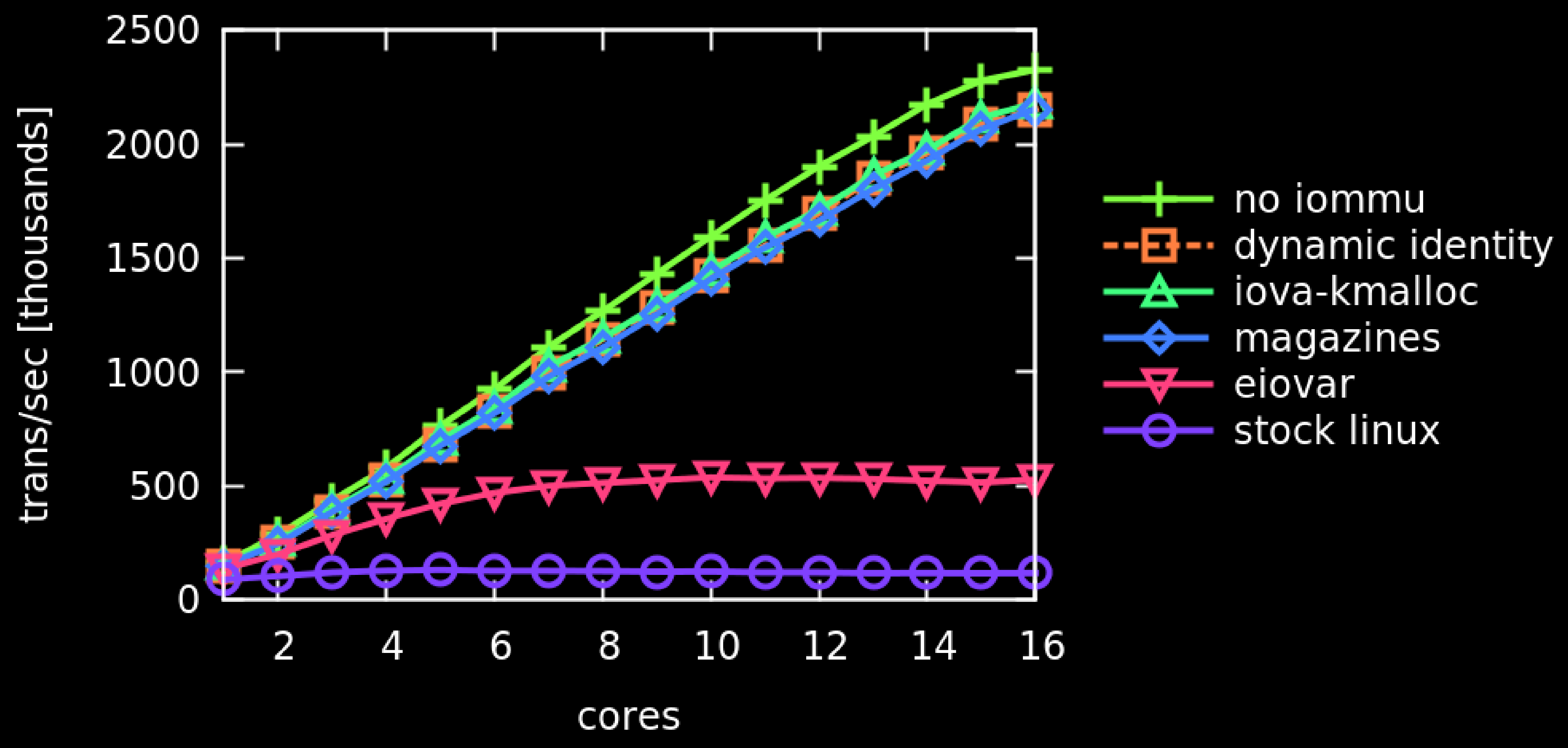
- Build on top of the Linux allocator
- Save freed IOVAs for reallocation
  - Use local caches to avoid contention
- Magazines (Bonwick 01)
- Still packs allocations

Evaluation

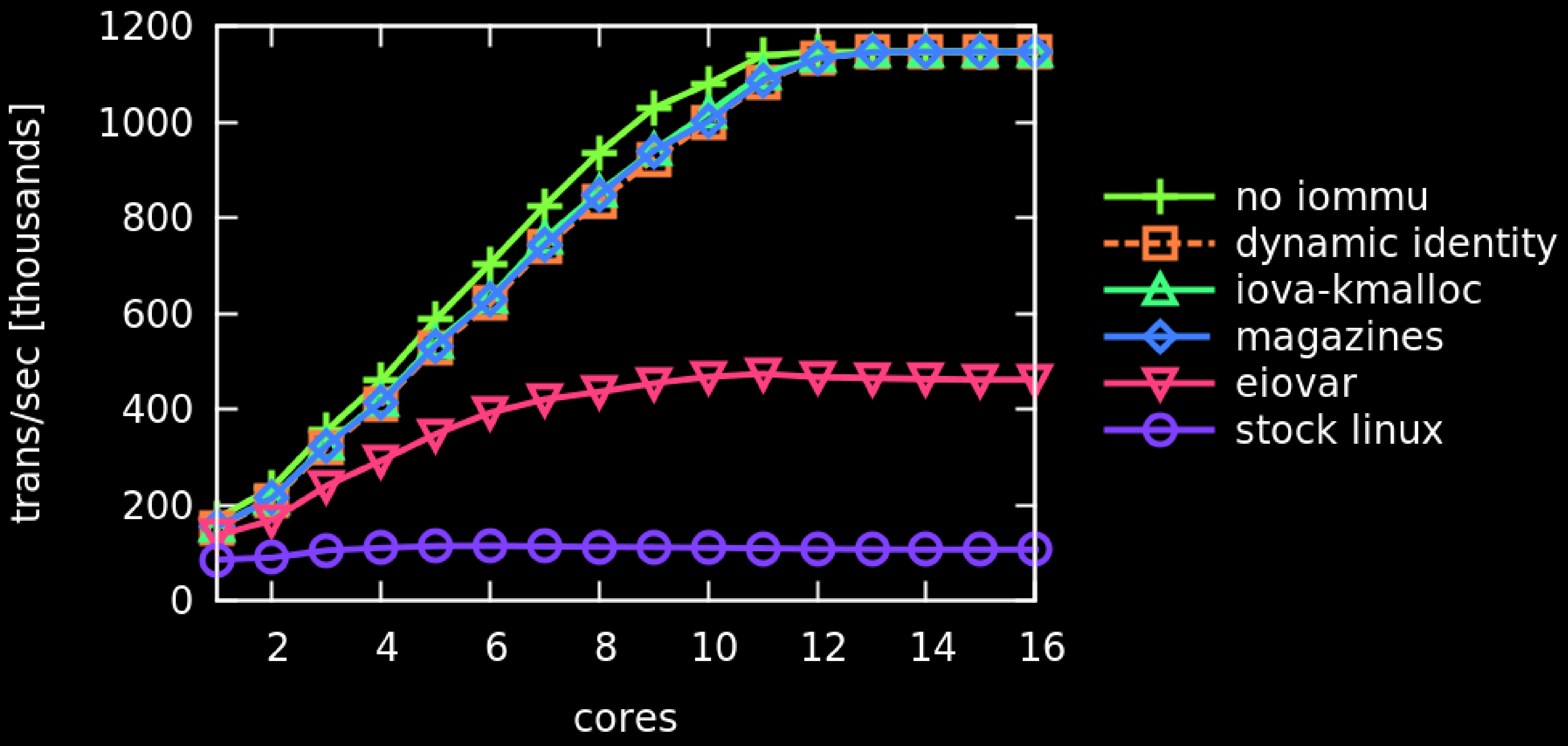
Our Setup

- 2x Dell PowerEdge R430, each
  - 16 Haswell E5 cores @2.4GHz
  - 10 Gigabit Ethernet NIC
- Server
  - Modified Linux 3.17.2
- Client
  - IOMMU turned off
  - Stock Linux 3.13.0-45 (Ubuntu)

High Throughput TCP Request-Response

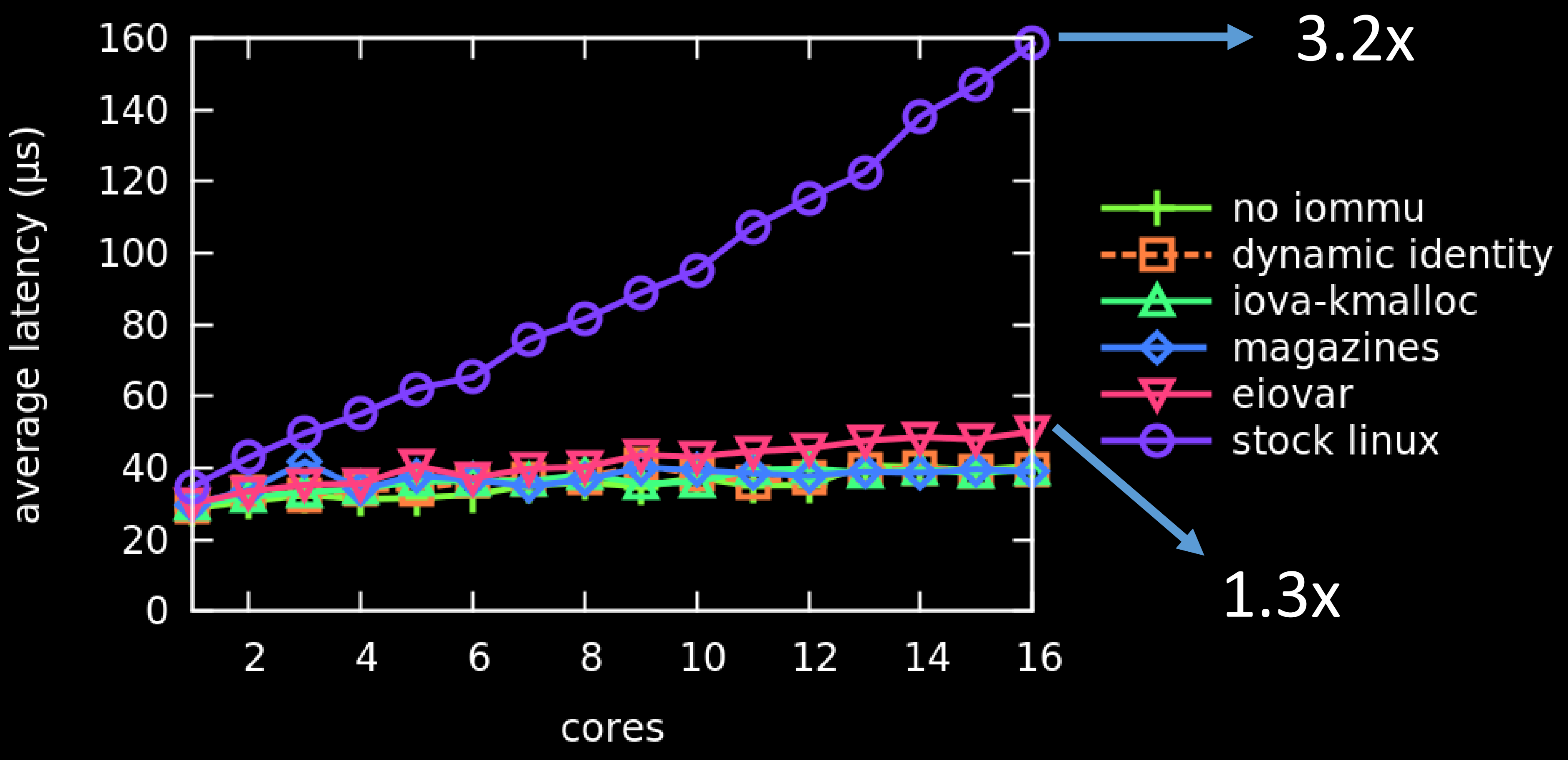


Memcached

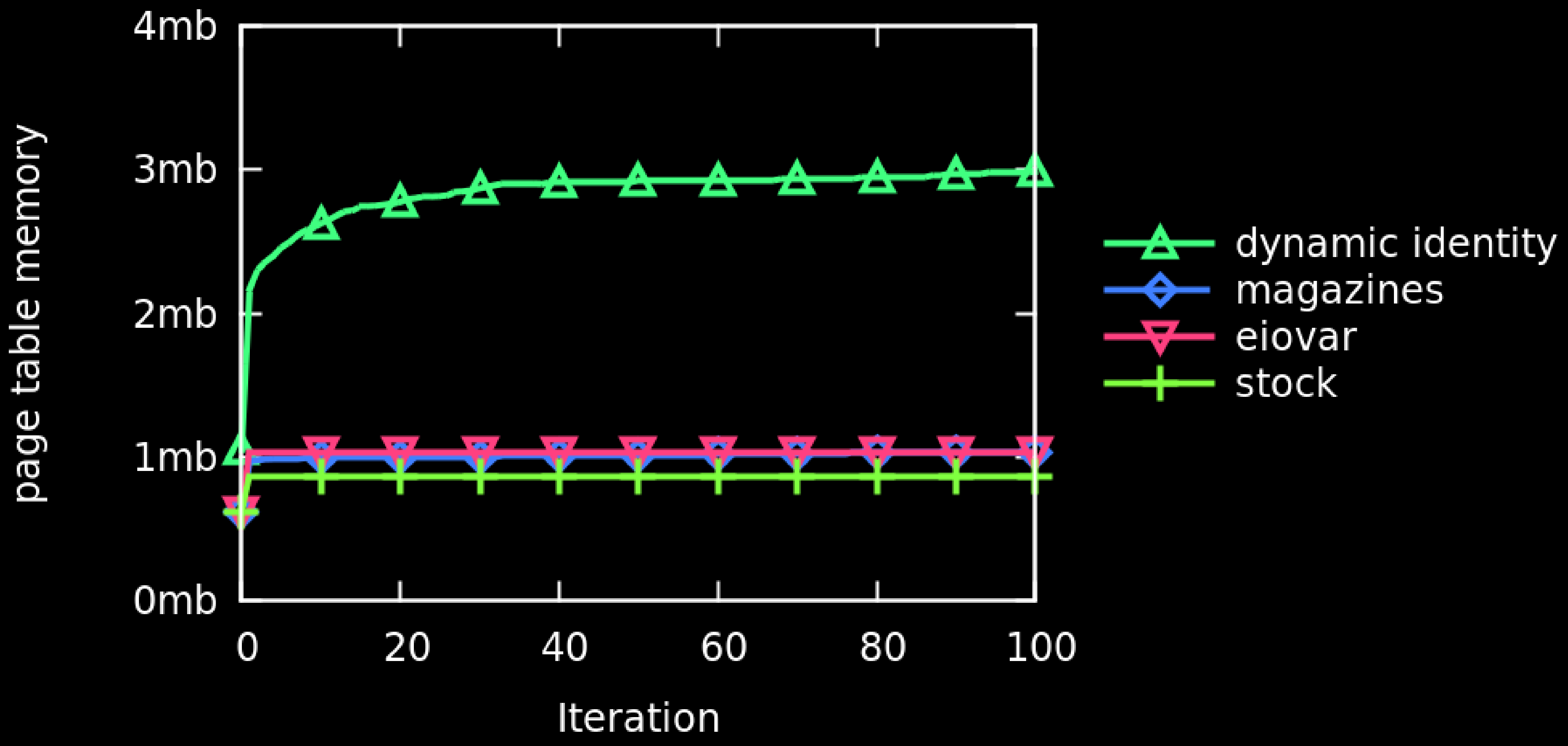


1200
1000
800
600
400
200
0
trans/sec [thousands]
2
4
6
8
10
12
14
16
cores
no iommu
dynamic identity
iova-kmalloc
magazines
eiovar
stock linux

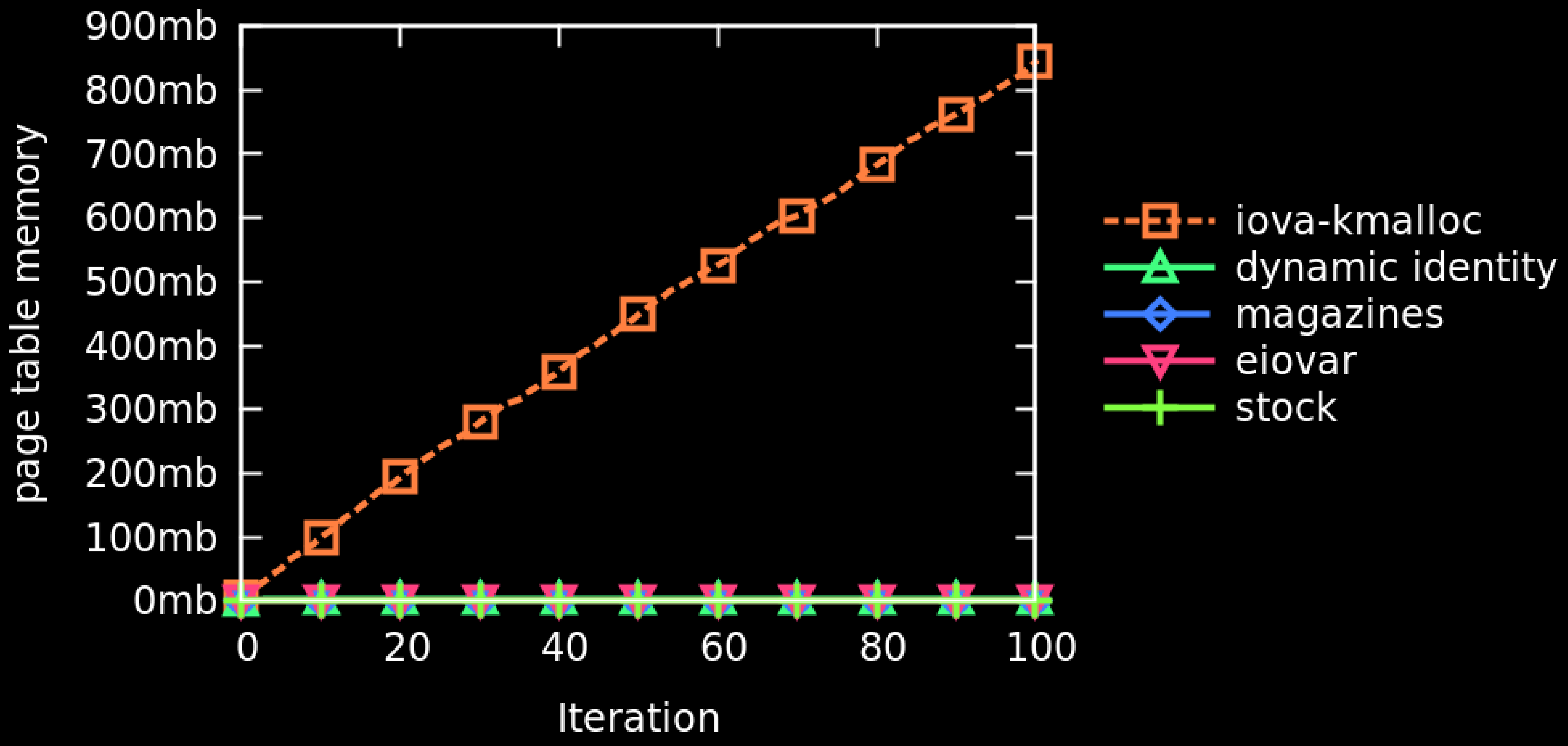
Latency - Multiple Dedicated Cores



Page Tables

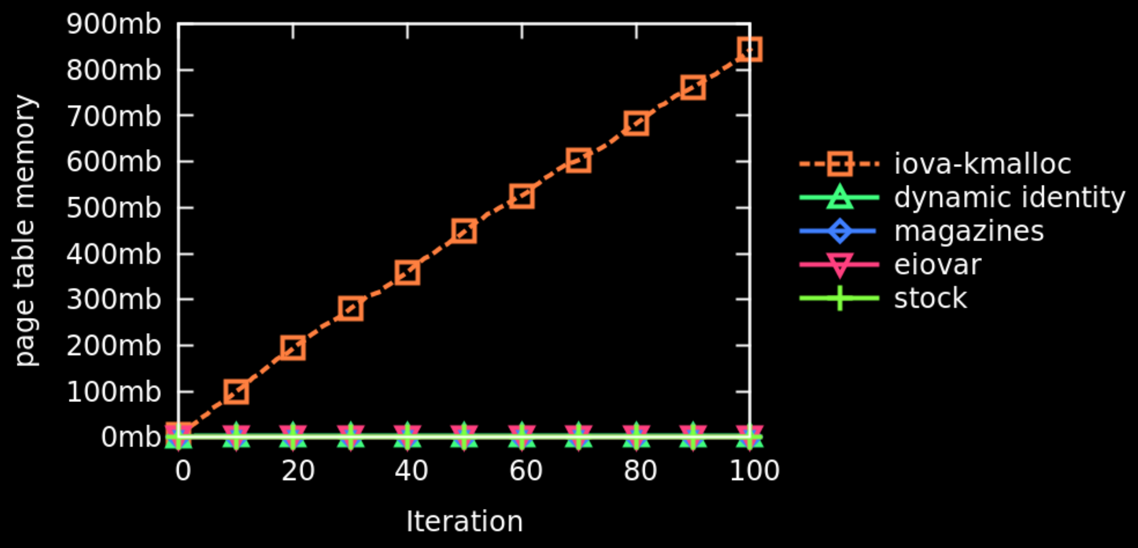

4mb
3mb
2mb
1mb
0mb
page table memory
0
20
40
60
80
100
Iteration
dynamic identity
magazines
eiovar
stock

Page Tables (with iova-kmalloc)



Page Tables

- Linux never frees page tables
- Need IOVA allocator that accounts for that
  - Can take notes from general purpose allocators



Design Space - Summary

	Time to Implement	Control of Page Tables?	Scale?
Dynamic 1:1	Weeks	No*	✓
IOVA-kmalloc	Hours	No	✓
Magazines	Days	Yes	✓

Conclusions

- MMU and IOMMU are different
- First IOMMU management schemes to scale
- Future work
  - Strict invalidation
  - Better I/O page table management
  - Subpage protection
- Questions?