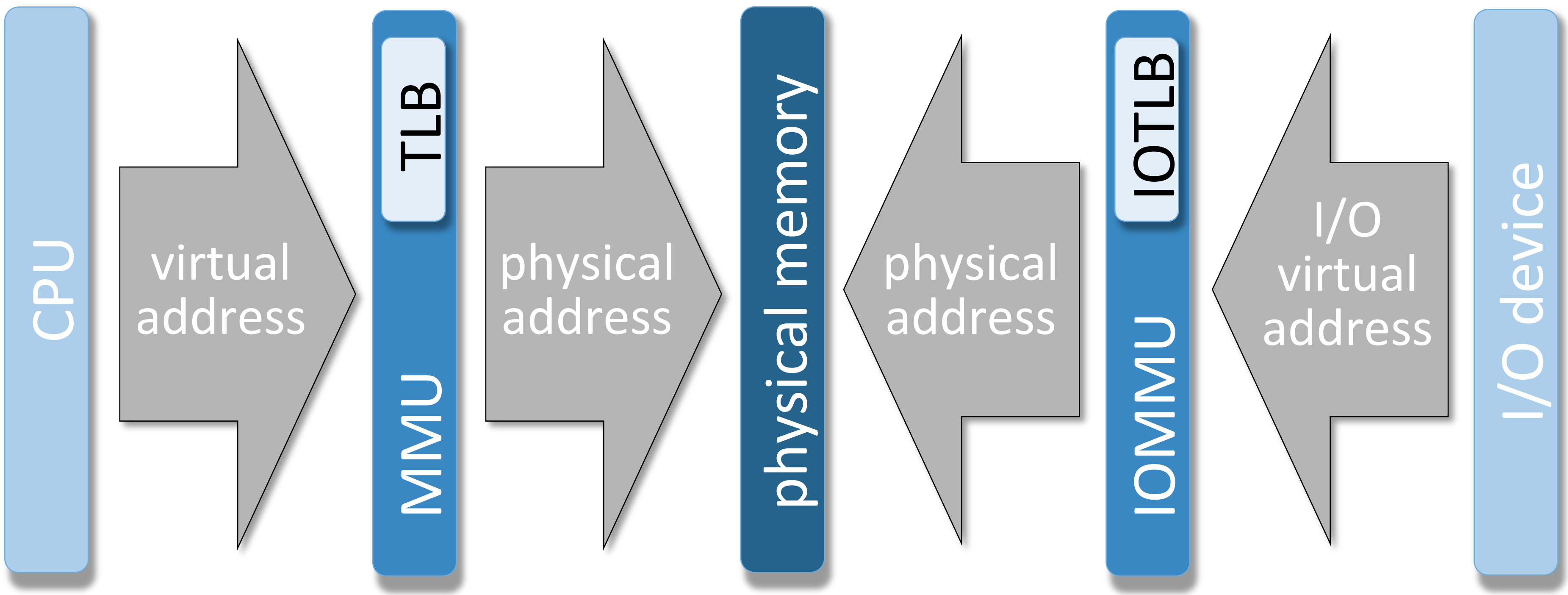
EiovaR

Efficient IOMMU Intra-Operating System Protection

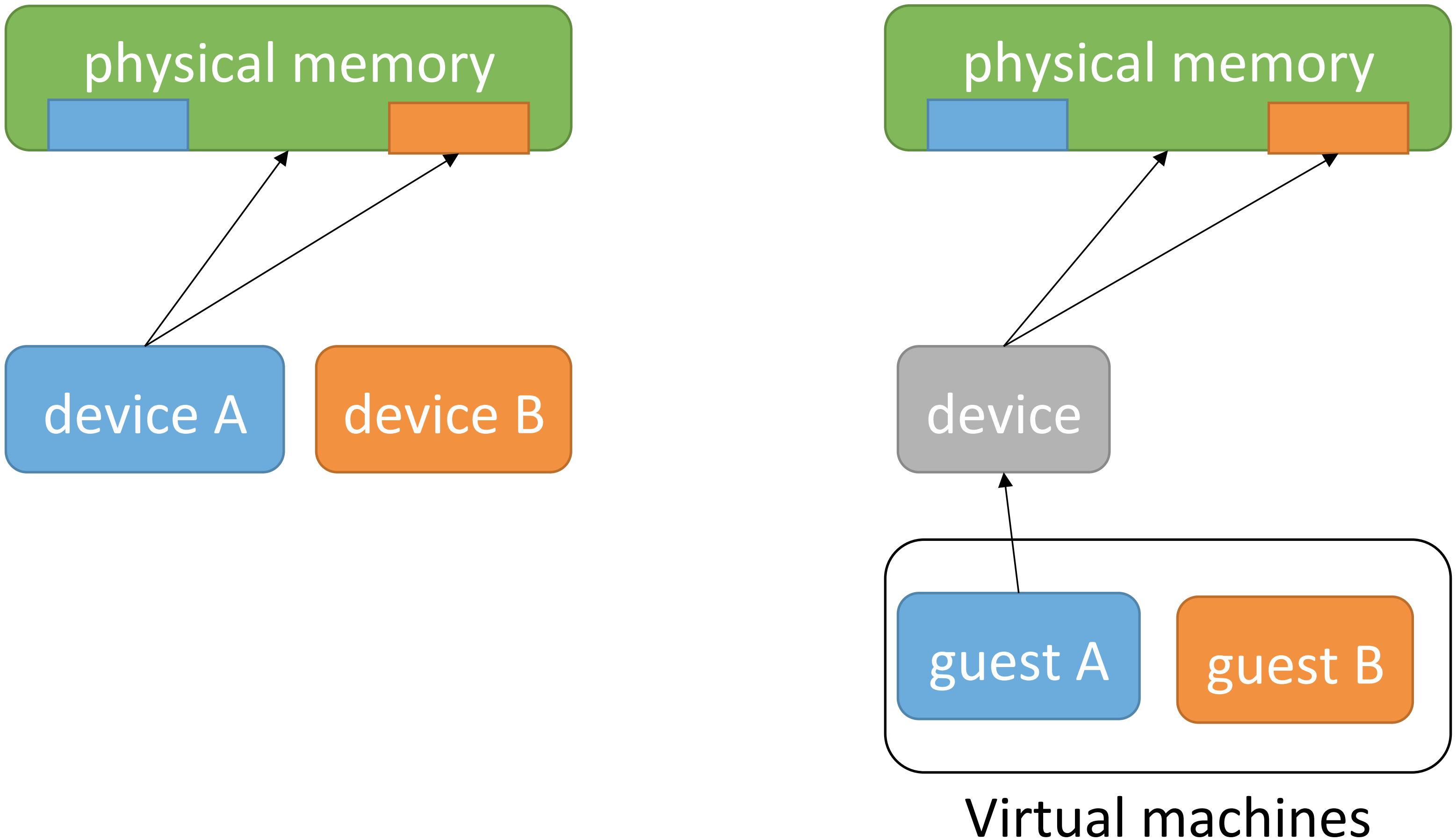
Moshe Malka, Nadav Amit, and Dan Tsafrir

Technion - Israel Institute of Technology

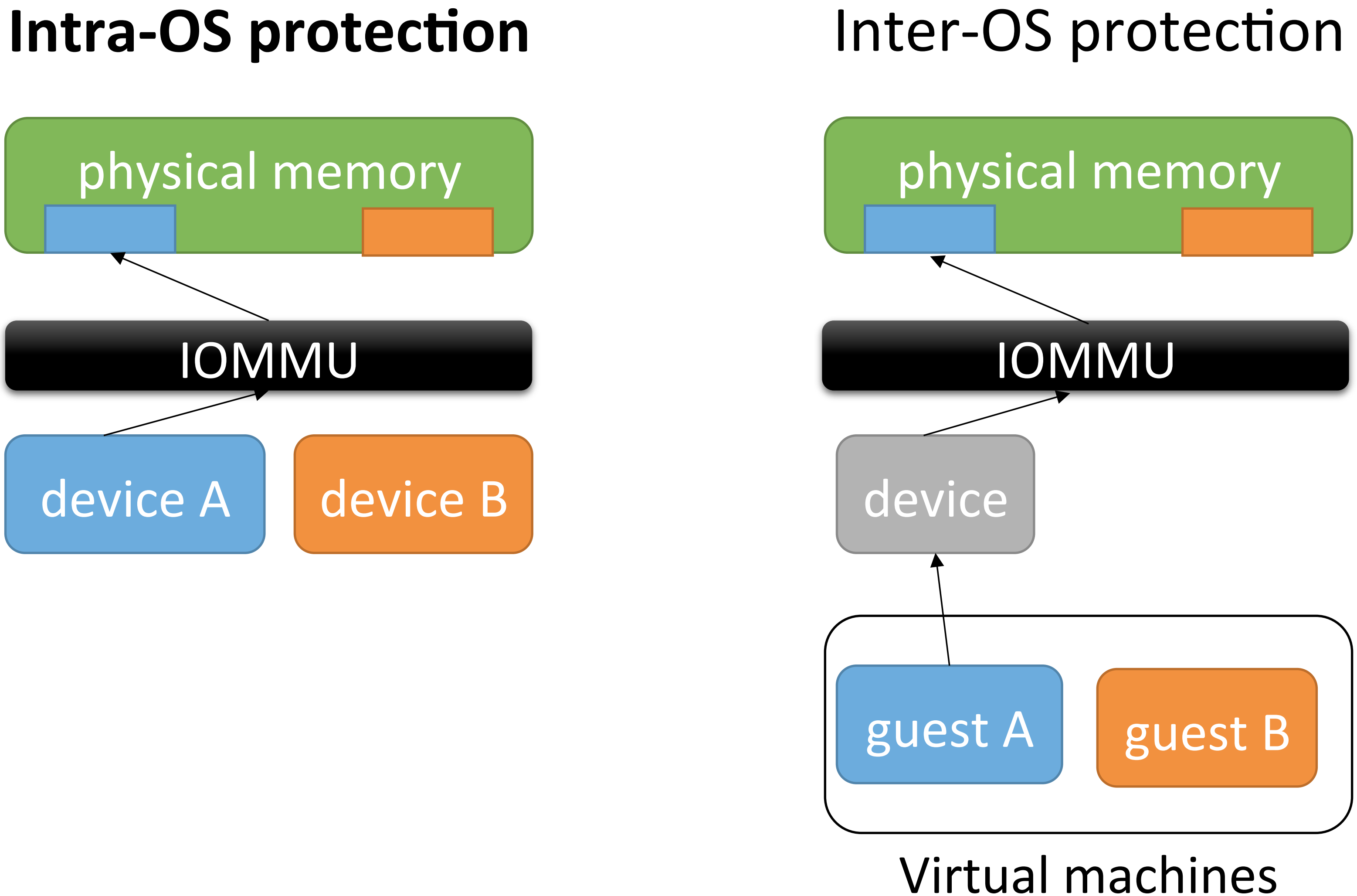
IOMMU = I/O memory management unit
IOVA = I/O virtual address



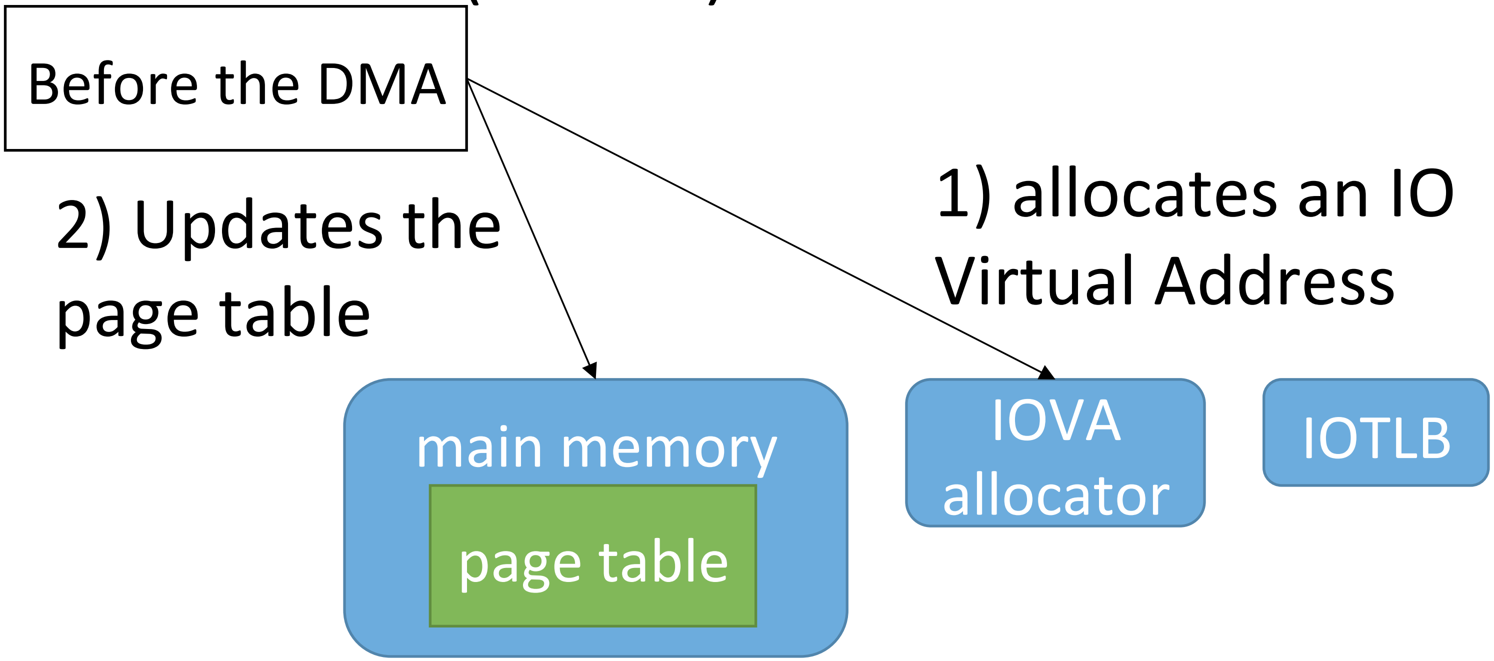
Direct memory access is dangerous



IOMMU provides memory protection

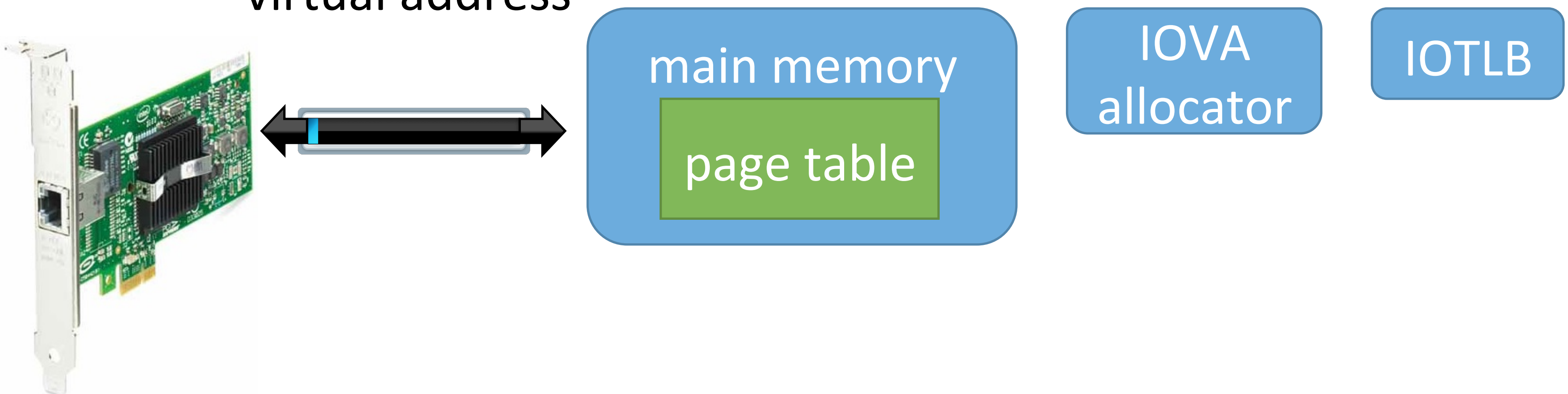


How Intra-OS protection works (strict)

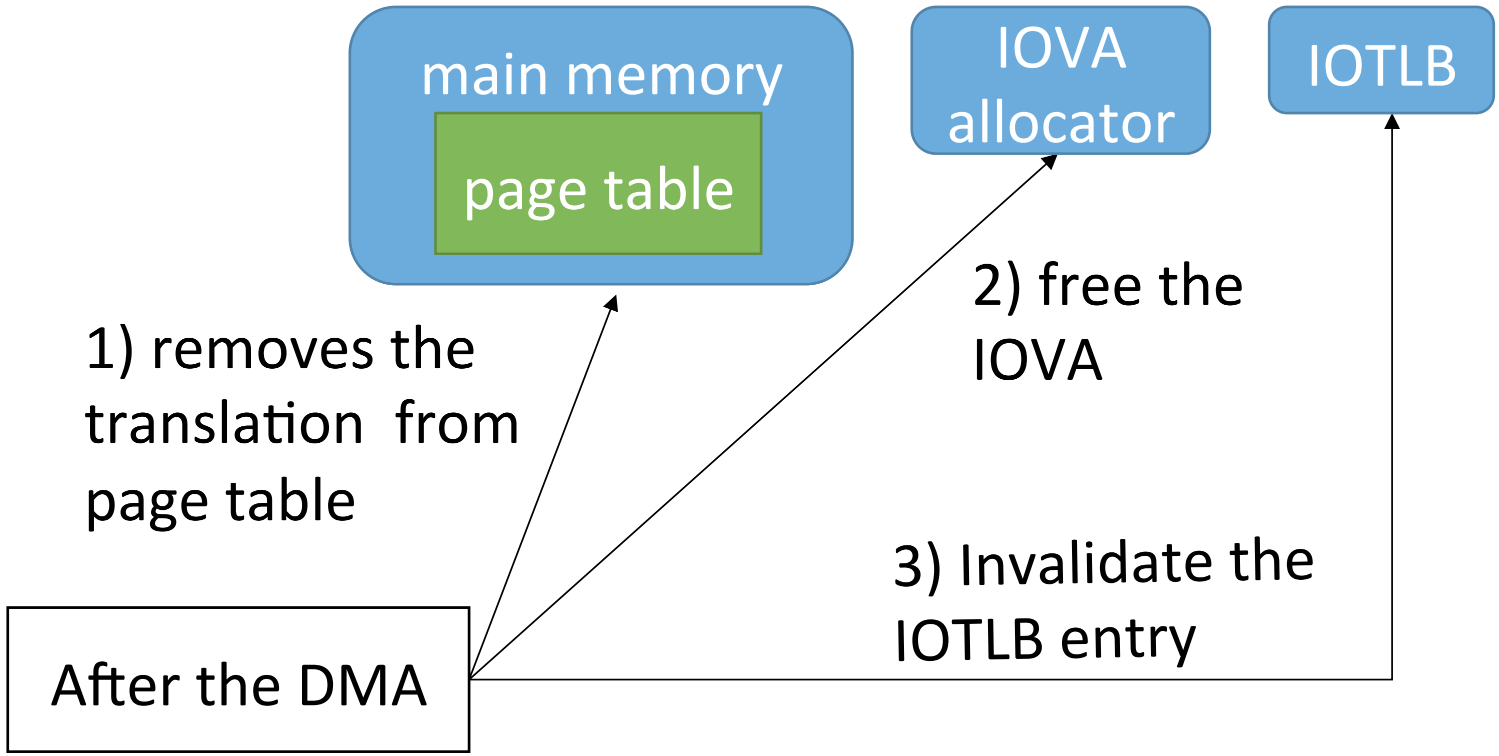


How Intra-OS protection works (strict)

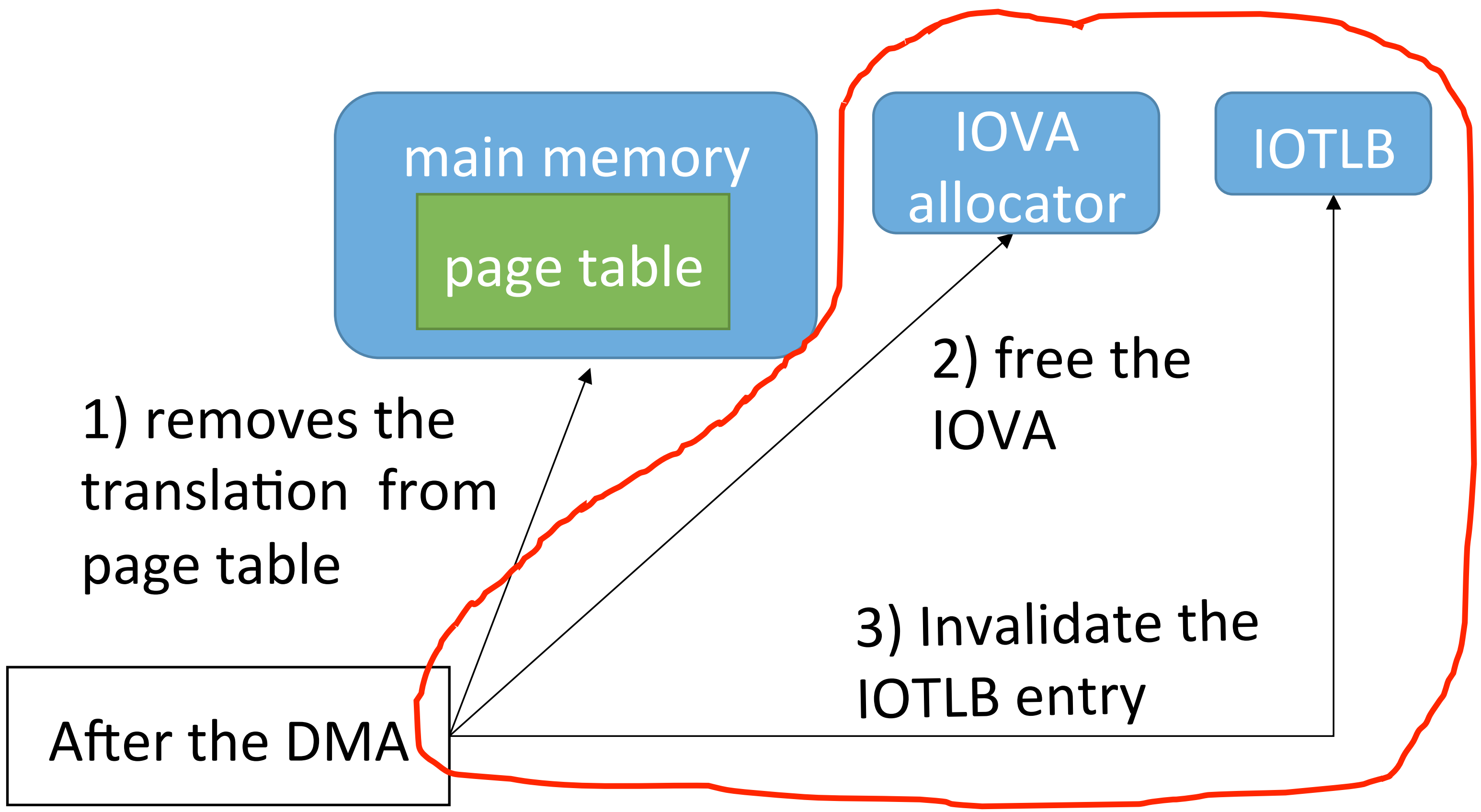
access the memory using virtual address



How Intra-OS protection works (strict)

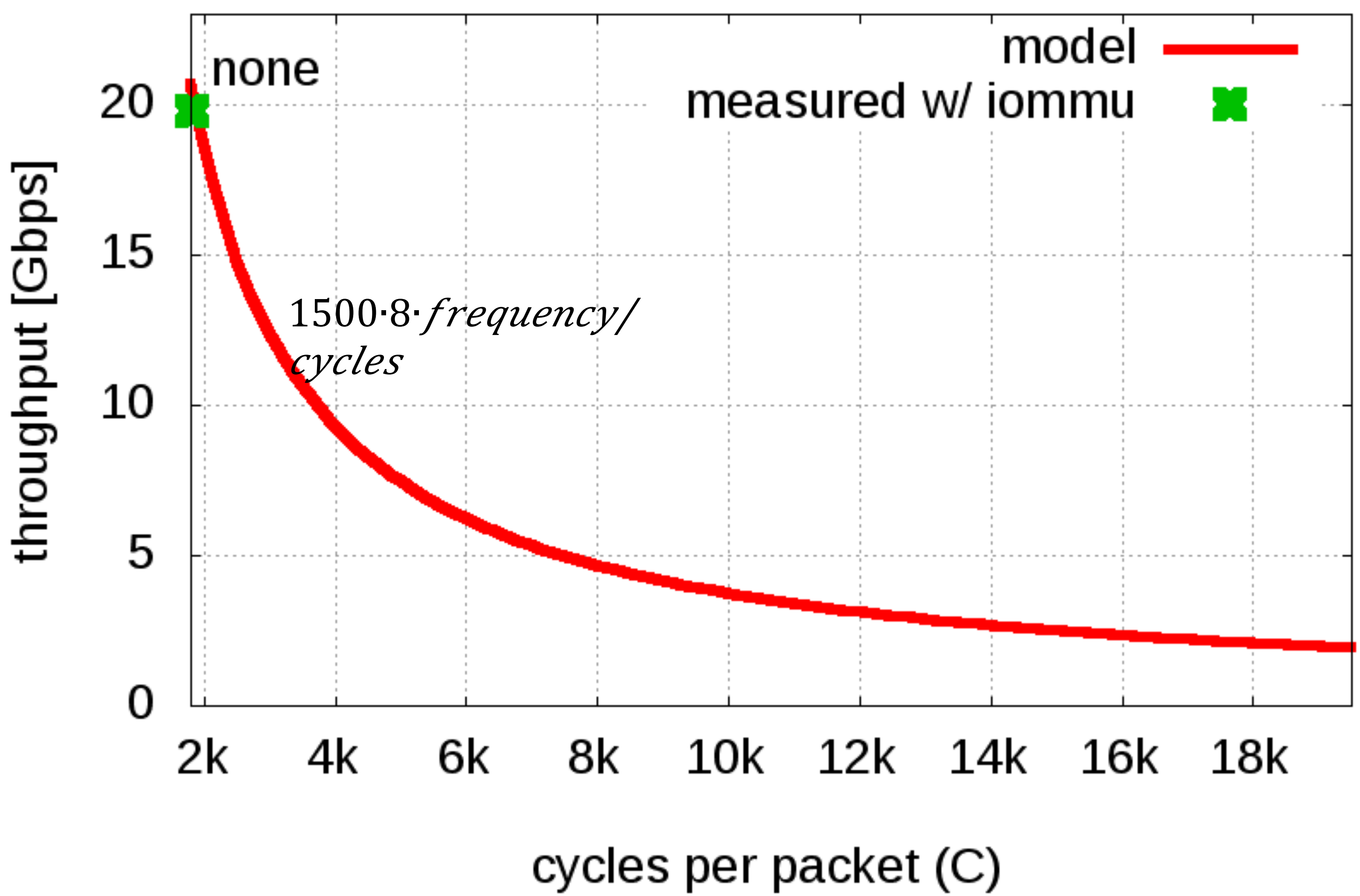


strict vs. defer
Protection tradeoff



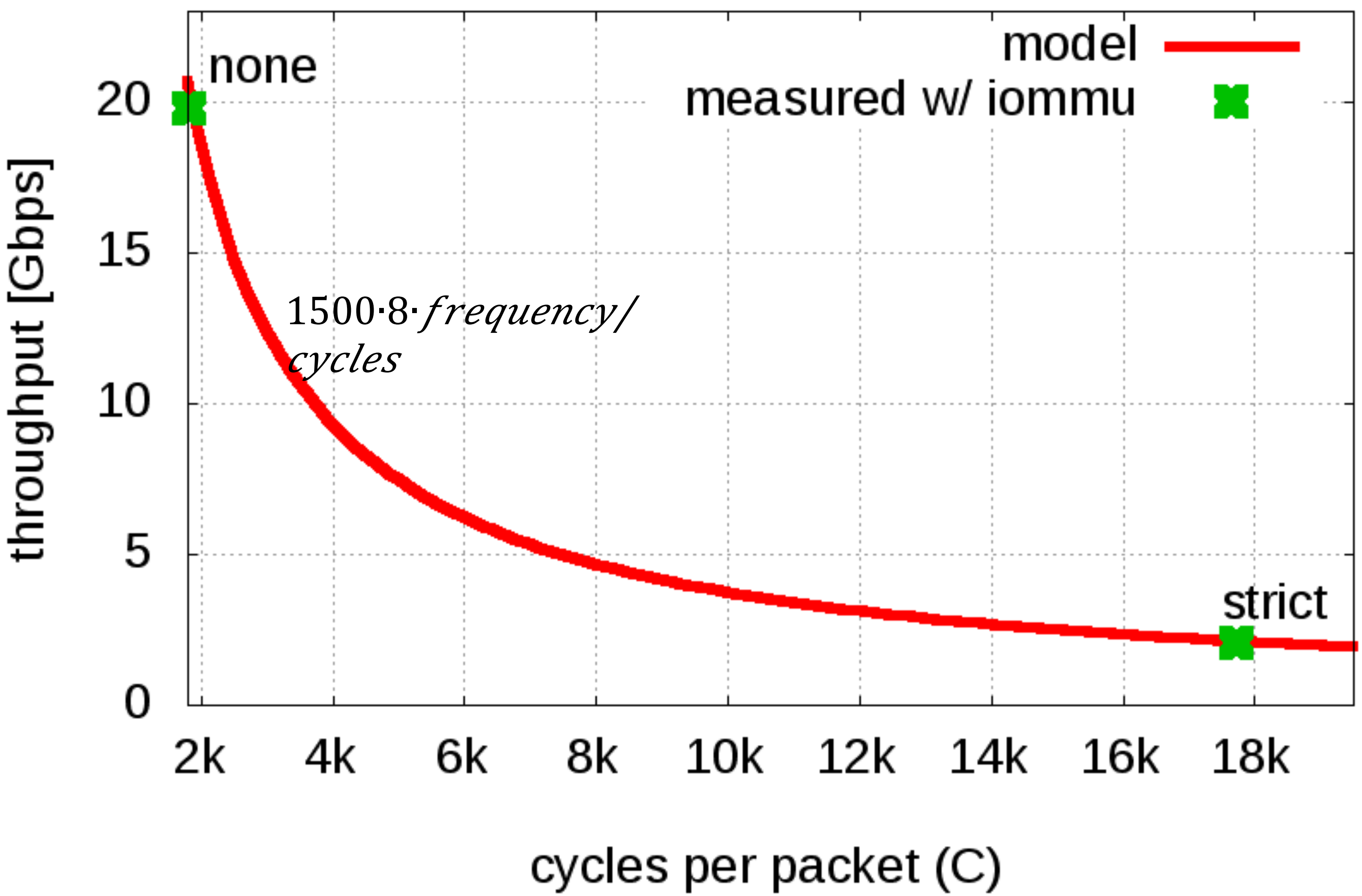
what's the performance cost?

NIC (mlx4) throughput without IOMMU



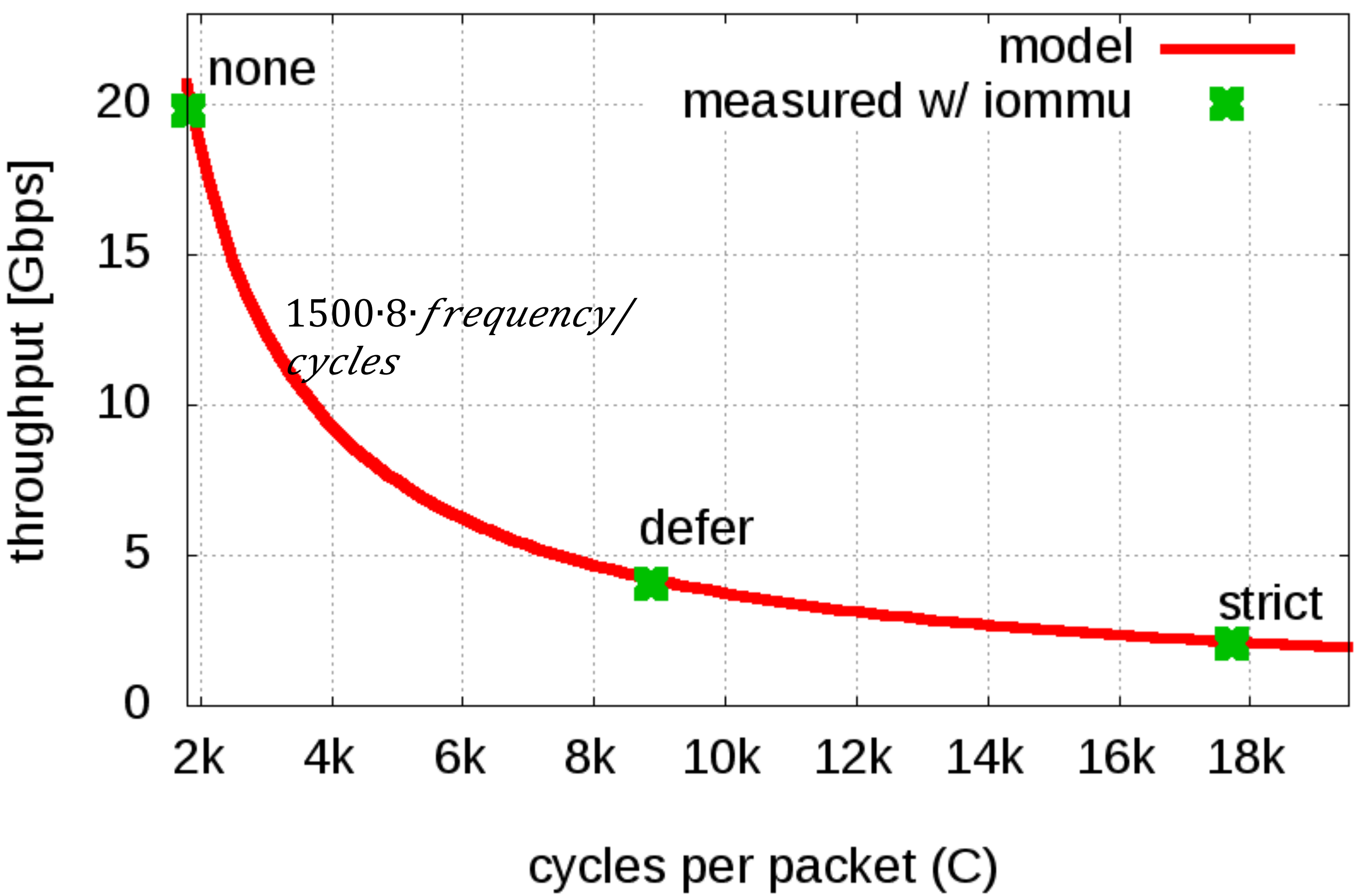
what's the performance cost?

NIC throughput with IOMMU (mlx4)



what’s the performance cost?

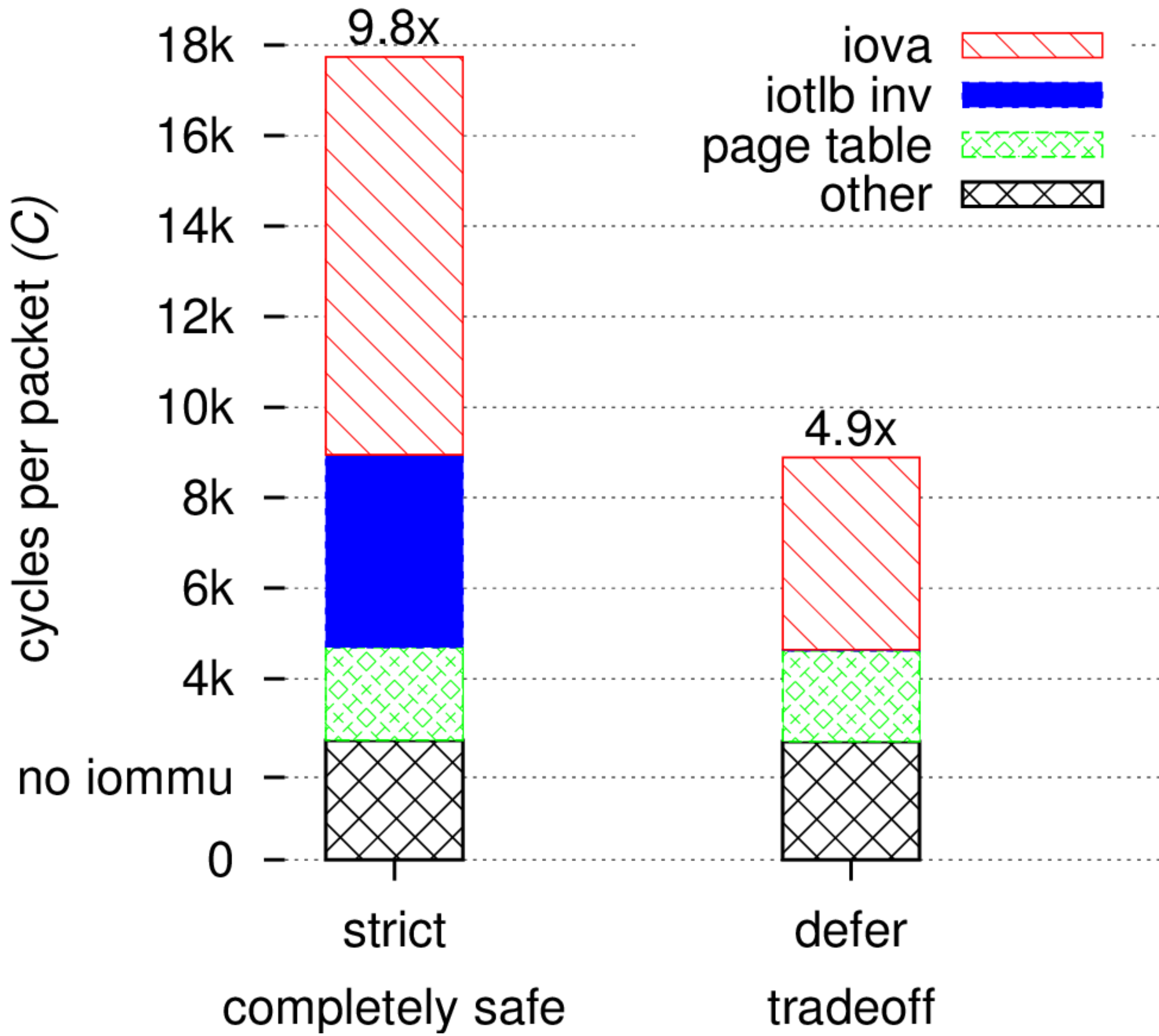
NIC throughput with IOMMU (mlx4)



Why is IOMMU so costly?

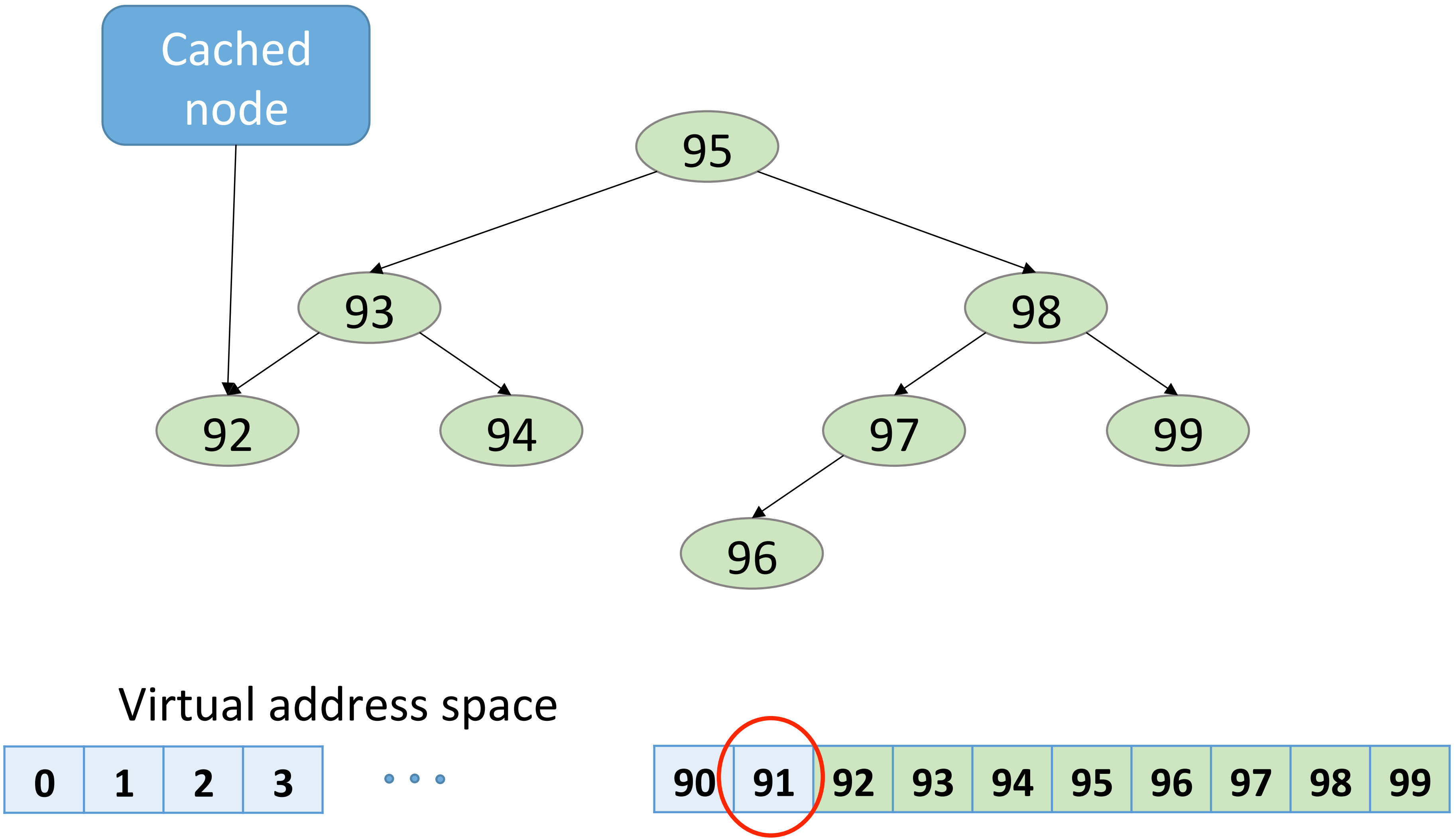
- **Common wisdom: "HW is to blame!"**
- **Correlation between the # of cycles it takes to process a packet and the throughput**
- **Our claim: SW is guiltier!**
  - OSes drive IOMMUs with suboptimal software
  - Could be much improved
  - All OSes we examined (not just Linux)

Overhead breakdown - same bench as before



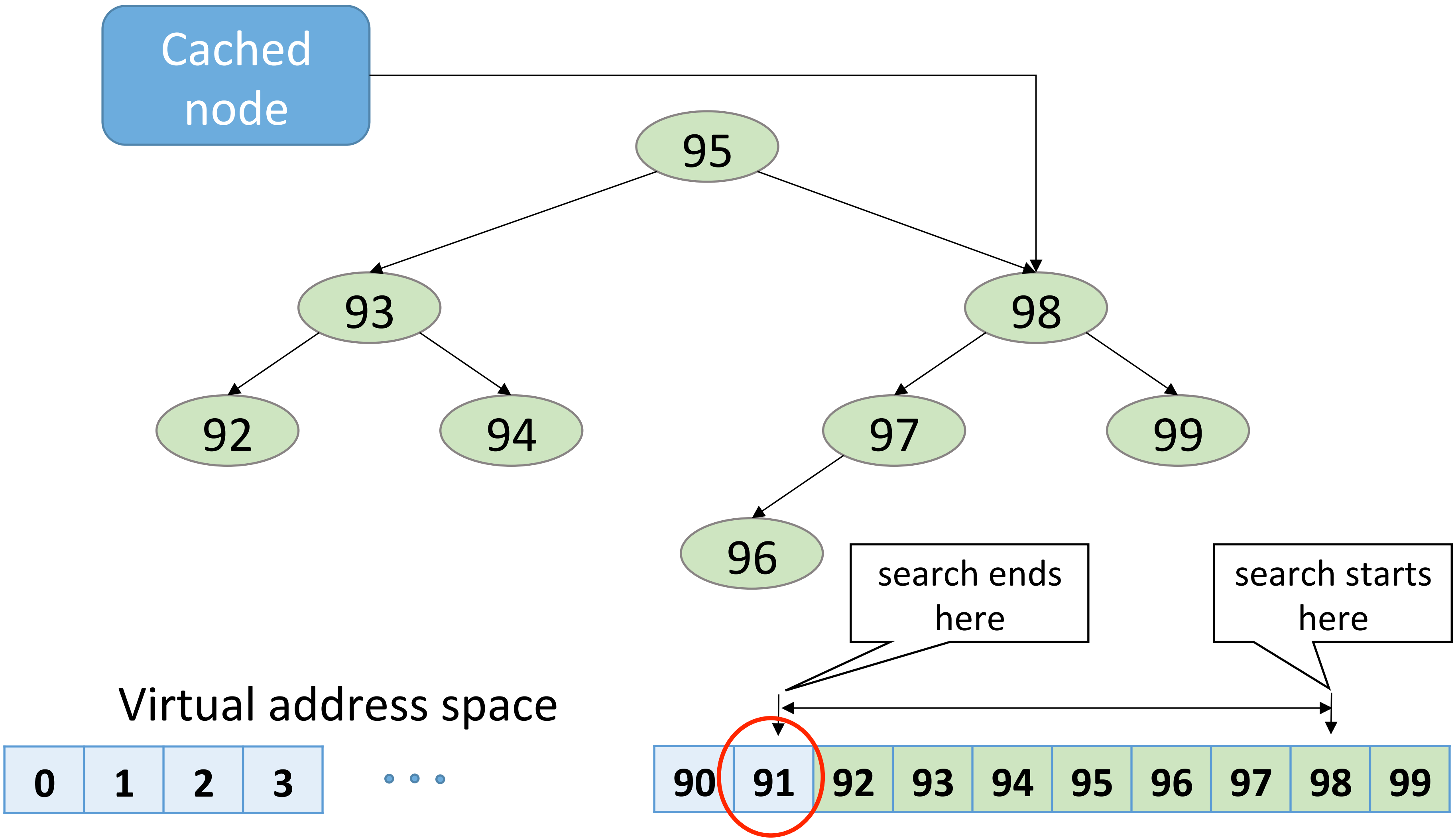
Why is IOVA (de)allocation so costly?

IOVA allocation

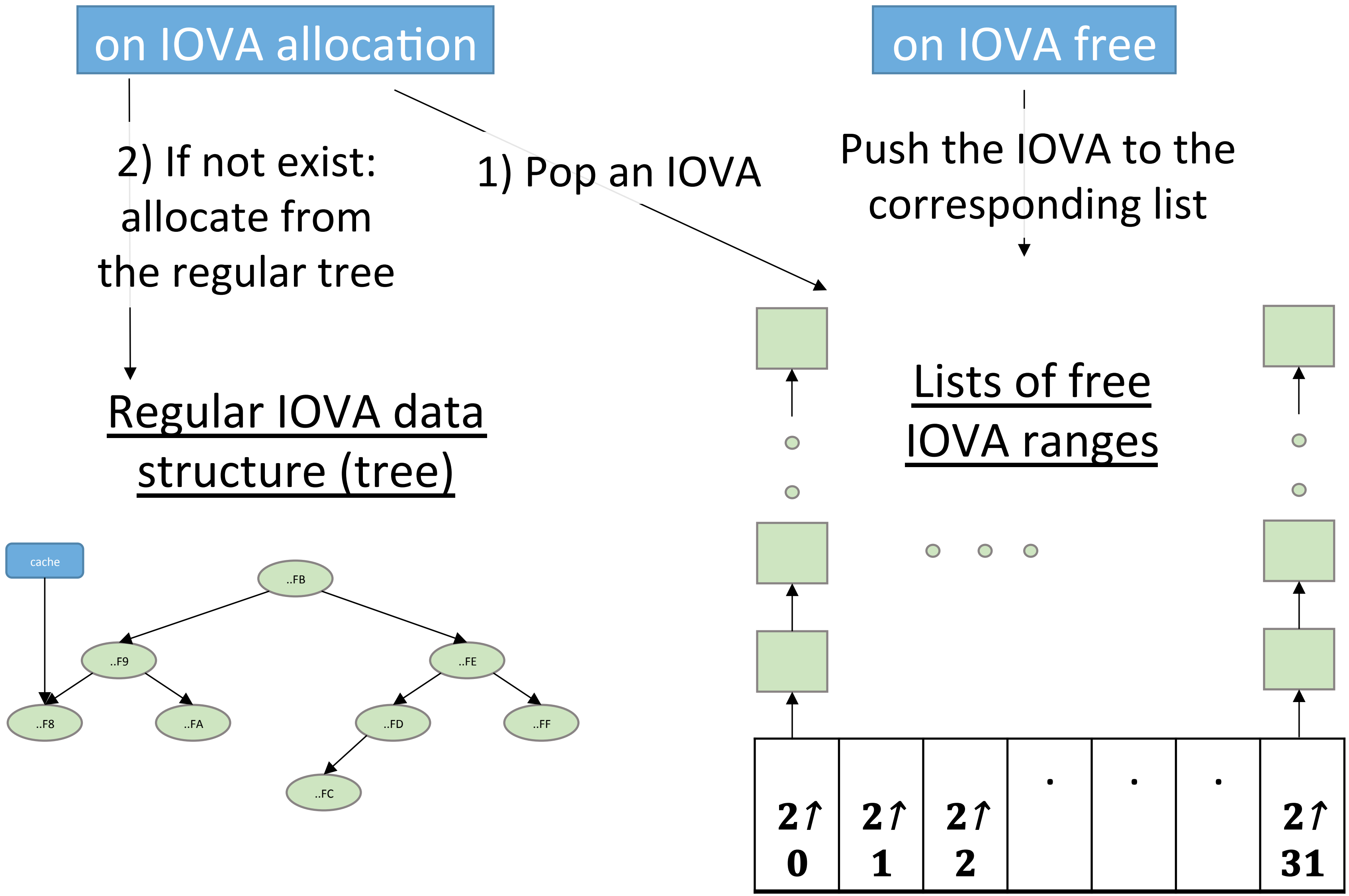


Why is IOVA (de)allocation so costly?

IOVA allocation after ..98 is **re**allocated

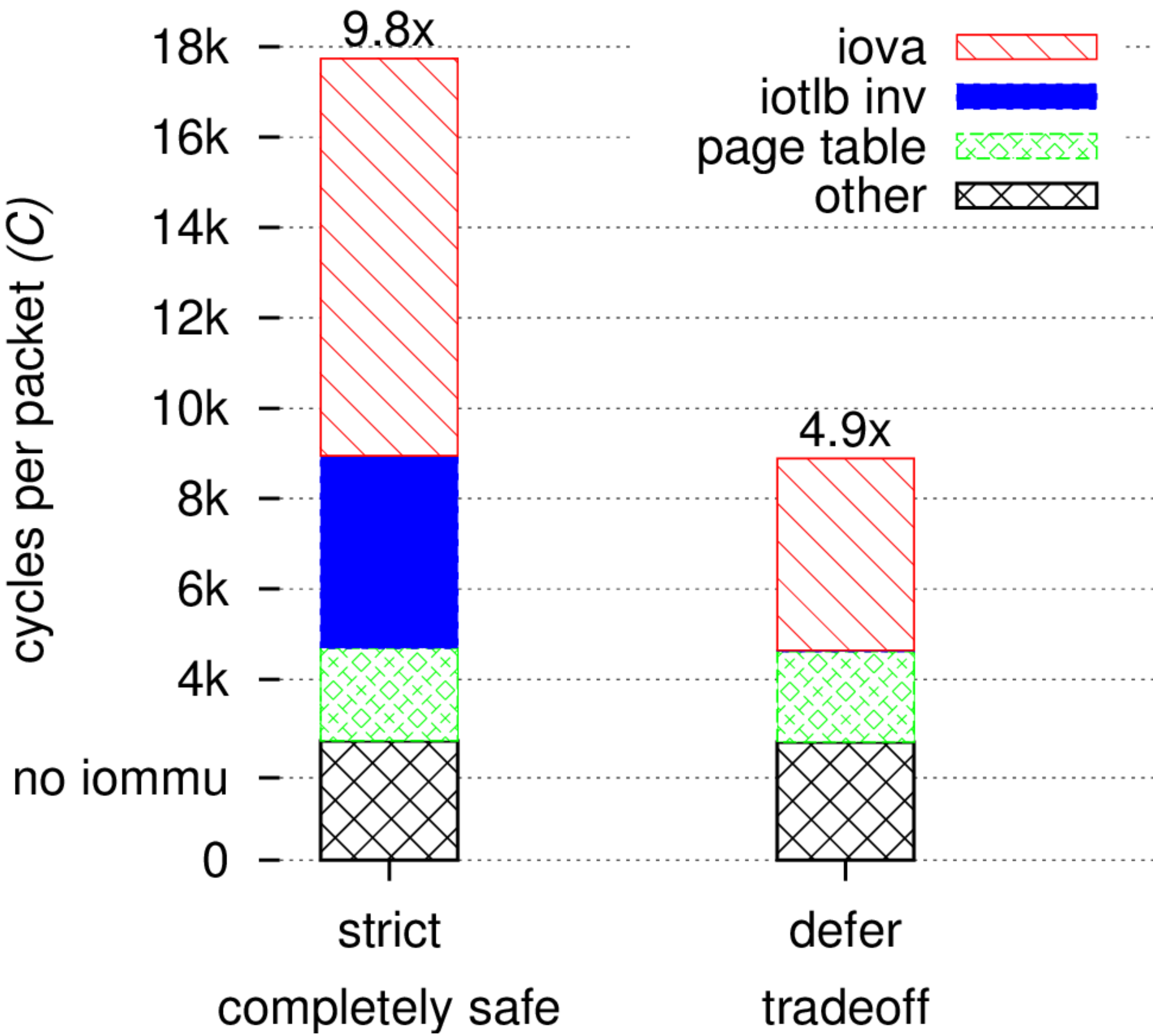


EiovaR: efficient IOVA allocator

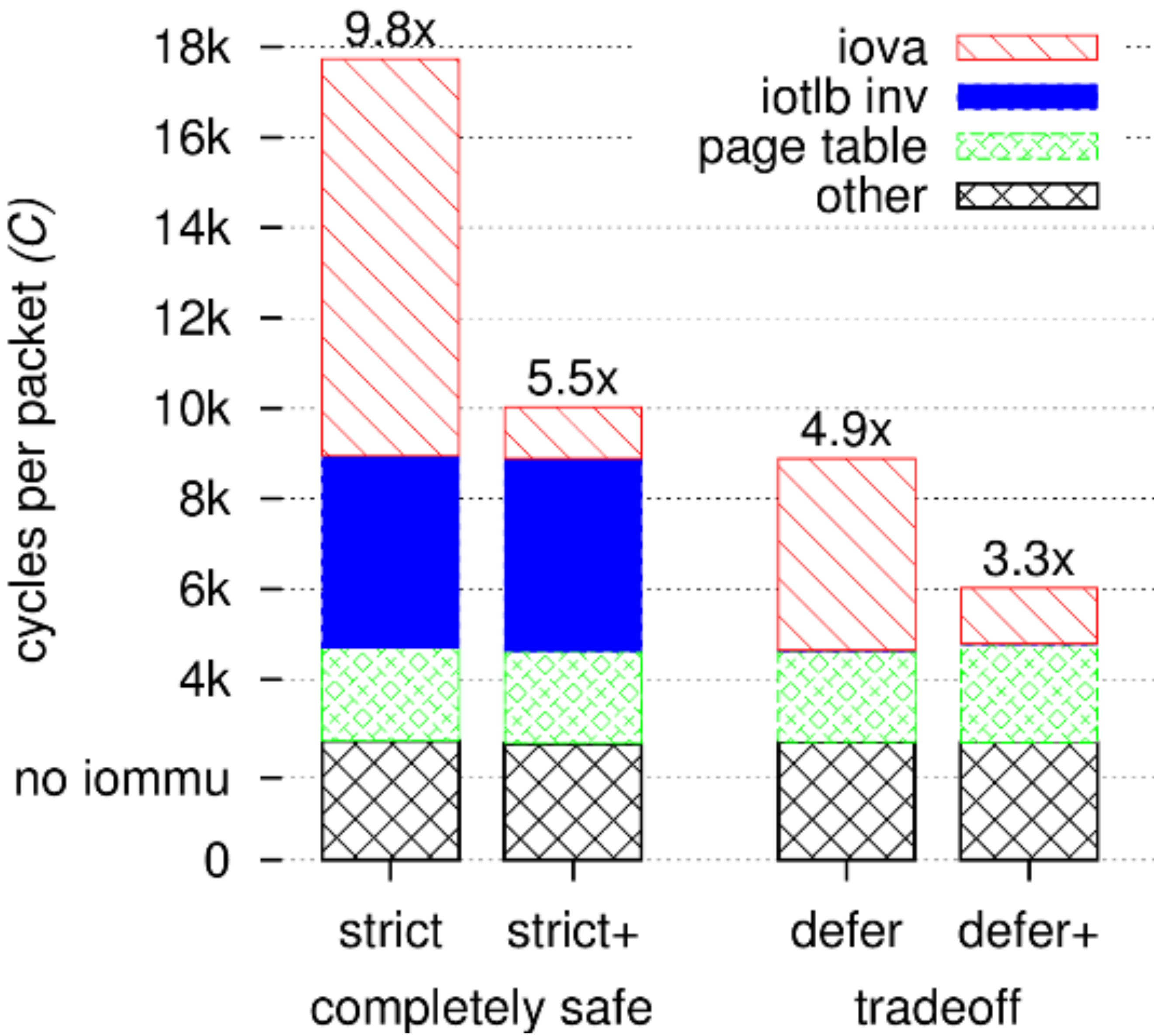


Results

How many cycles we reduced?

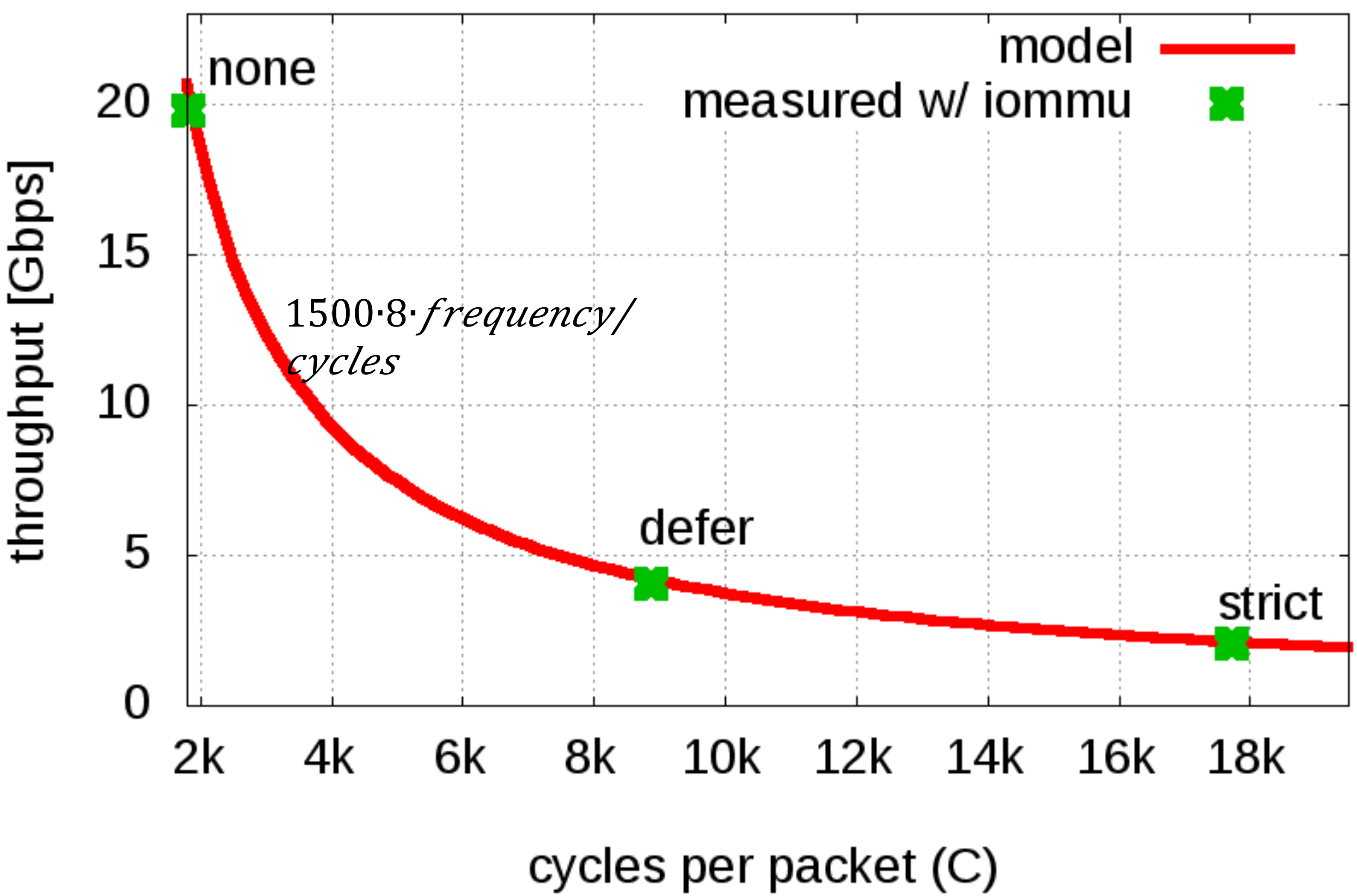


How many cycles we reduced?



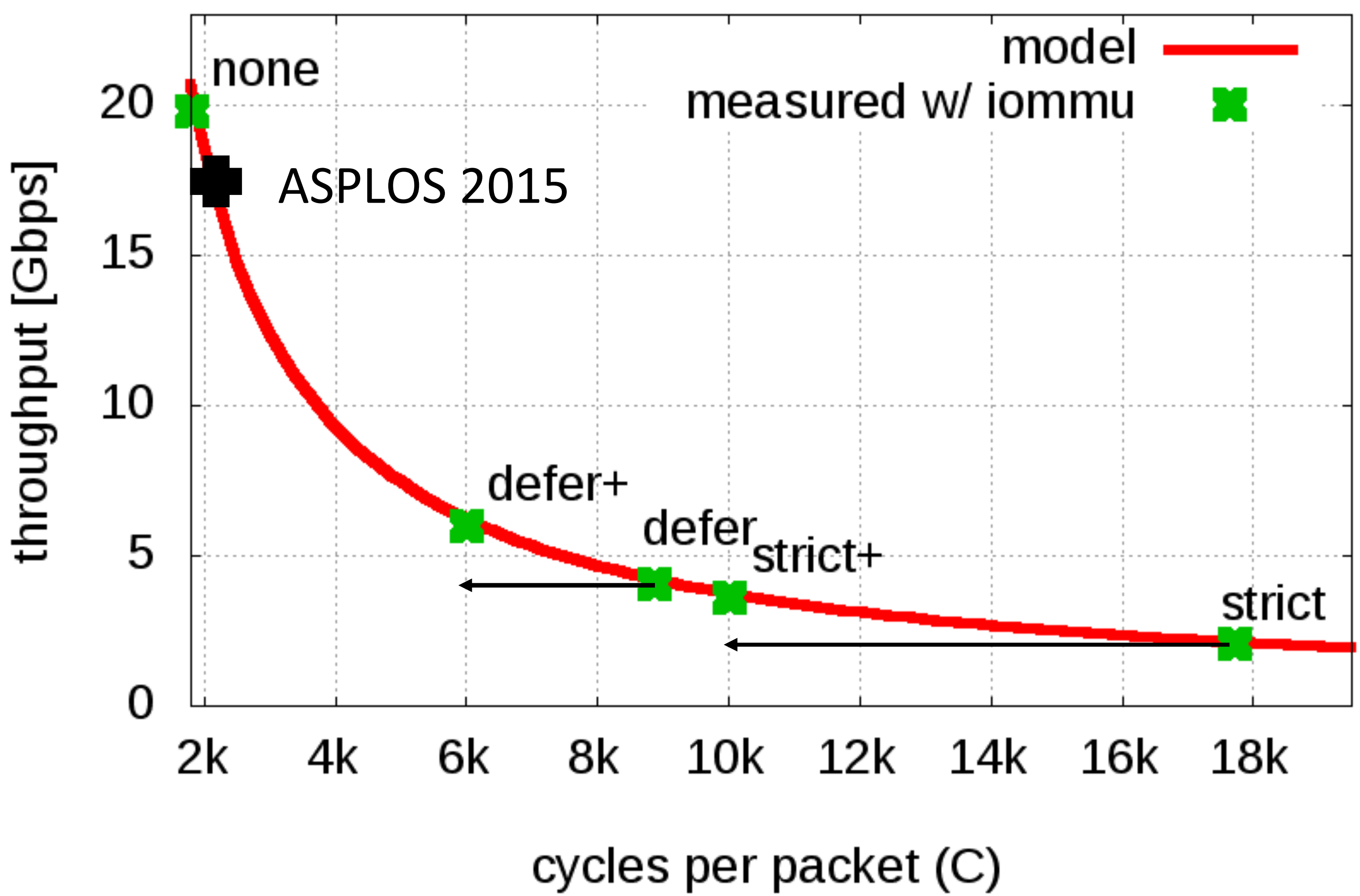
Performance improvement

NIC throughput with IOMMU (mlx4)



Performance improvement

NIC throughput with IOMMU (mlx4)



Micro- and macro-benchmarks (mlx)

protection	benchmark	baseline	EiovaR	diff
strict	Netperf Stream	1	2.37	+137%
	Netperf RR	1	1.27	+27%
	Apache 1MB	1	3.65	+265%
	Apache 1KB	1	2.35	+135%
	Memcached	1	4.58	+358%
defer	Netperf Stream	1	1.71	+71%
	Netperf RR	1	1.07	+7%
	Apache 1MB	1	1.21	+21%
	Apache 1KB	1	1.11	+11%
	Memcached	1	1.25	+25%

Generalizing

- So far, all results are related to Linux
- We found, however, that other OSes similarly drive the IOMMU with suboptimal SW
- In the paper, we show that FreeBSD is worse than Linux
    - (We even describe a patch that we already managed to push to FreeBSD that improves throughput by 76 %)
- We also found that Solaris and Darwin have similar deficiencies (not in the paper)

Conclusions

- Common wisdom is “IOMMU is slow”
  - Not true!
  - Suboptimal SW makes it much slower than it should be
- We find that characteristics of the workload experienced by the IOMMU allow for even our basic & minimal freelist optimization to deliver high performance (which BTW consists of about 250-lines patch)
- Although our analysis focuses on Linux, other OSes suffer from issues

Thank You!

Q & A