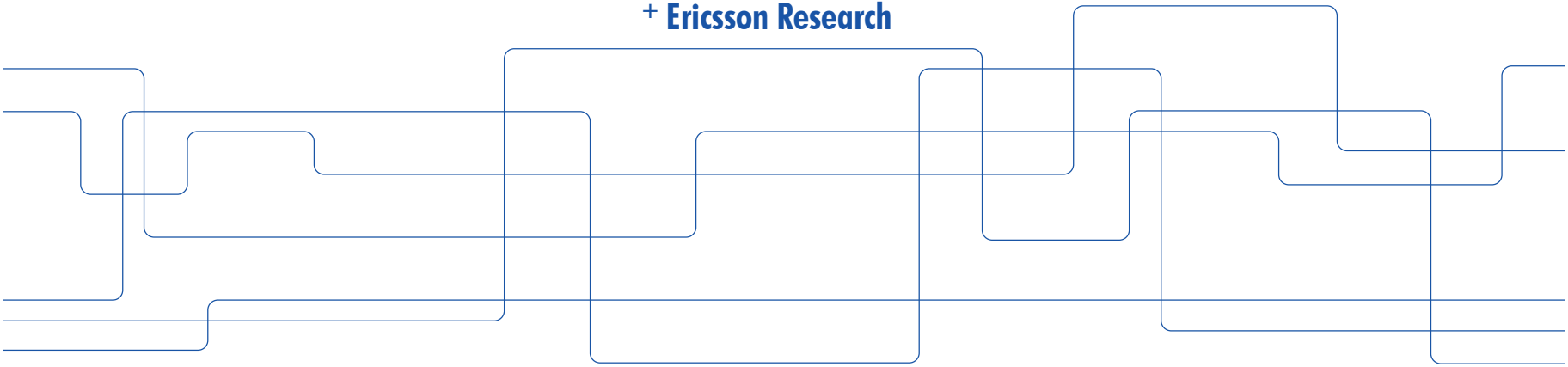
Reexamining Direct Cache Access to Optimize I/O Intensive Applications for Multi- hundred- gigabit Networks

Alireza Farshin[*], Amir Roozbeh[*+], Gerald Q. Maguire Jr.[*], Dejan Kostić

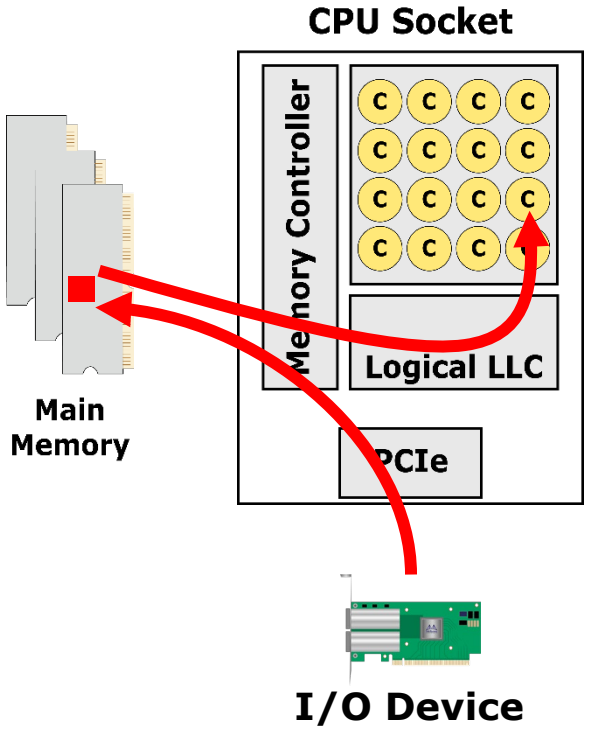
[*] KTH Royal Institute of Technology, School of Electrical Engineering and Computer Science (EECS)

[+] Ericsson Research



Traditional I/O

1. I/O device DMAs* packets to main memory
2. CPU later fetches them to cache

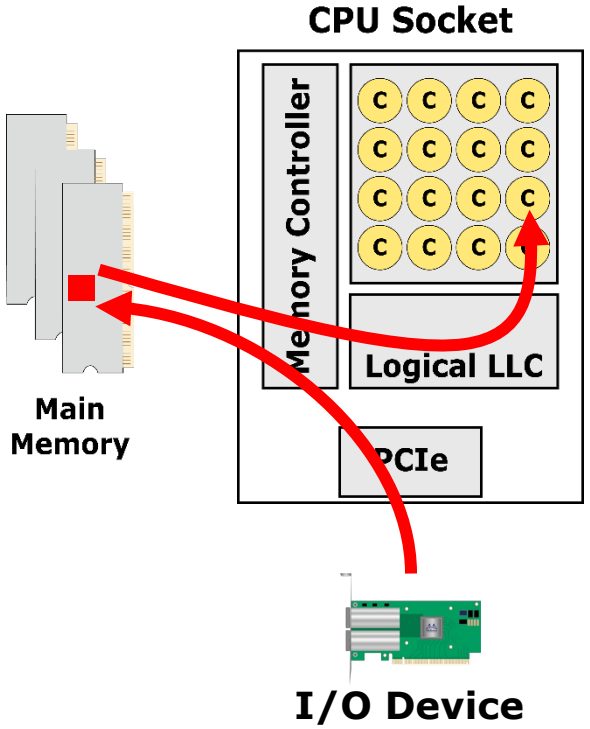


* Direct Memory Access (DMA)

Traditional I/O

1. I/O device DMAs* packets to main memory
2. CPU later fetches them to cache

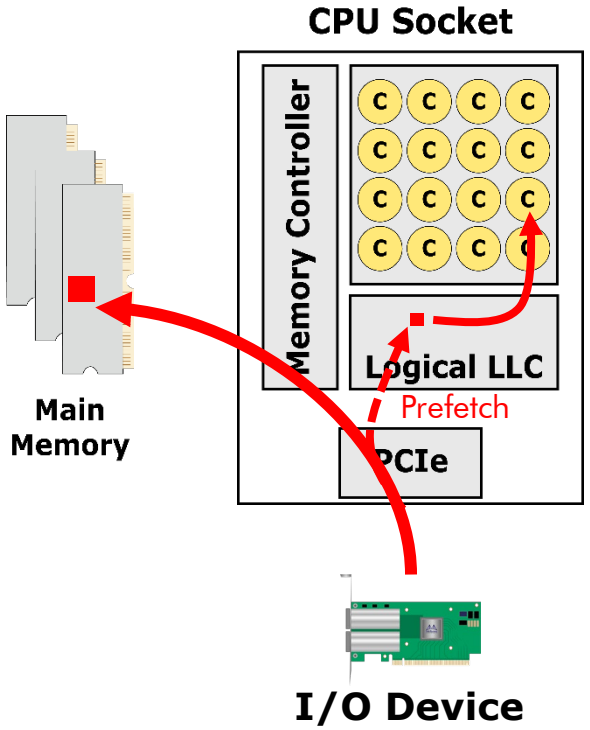
Inefficient:
- Large number of accesses to main memory
- High access latency (>60ns)
- Unnecessary memory bandwidth usage



* Direct Memory Access (DMA)

Direct Cache Access (DCA)

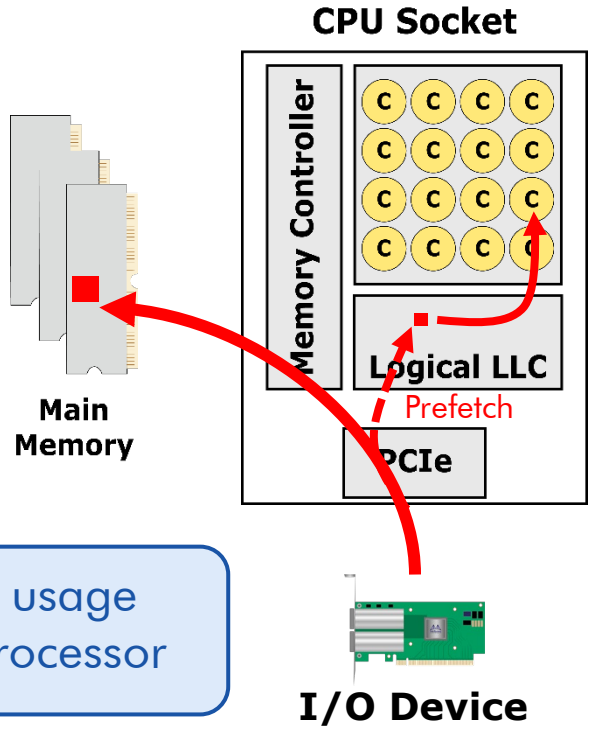
1. I/O device DMAs packets to main memory
2. DCA exploits TPH* to prefetch a portion of packets into cache
3. CPU later fetches them from cache



* PCIe Transaction protocol Processing Hint (TPH)

Direct Cache Access (DCA)

1. I/O device DMAs packets to main memory
2. DCA exploits TPH* to prefetch a portion of packets into cache
3. CPU later fetches them from cache

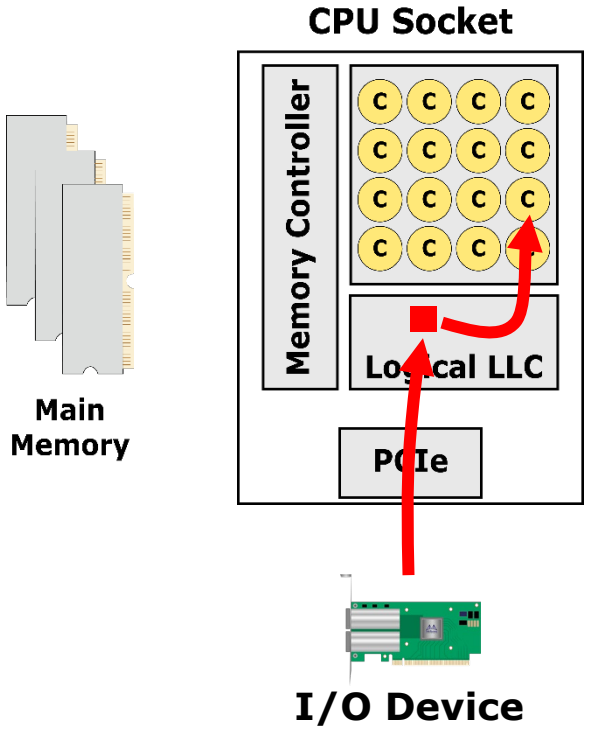


- Still inefficient in terms of memory bandwidth usage
- Requires OS intervention and support from processor

* PCIe Transaction protocol Processing Hint (TPH)

Intel Data Direct I/O (DDIO)

- DDIO in Xeon processors since Xeon E5
- DMA packets or descriptors directly to/from Last Level Cache (LLC)



Trends

More in-network computing + offloading capabilities

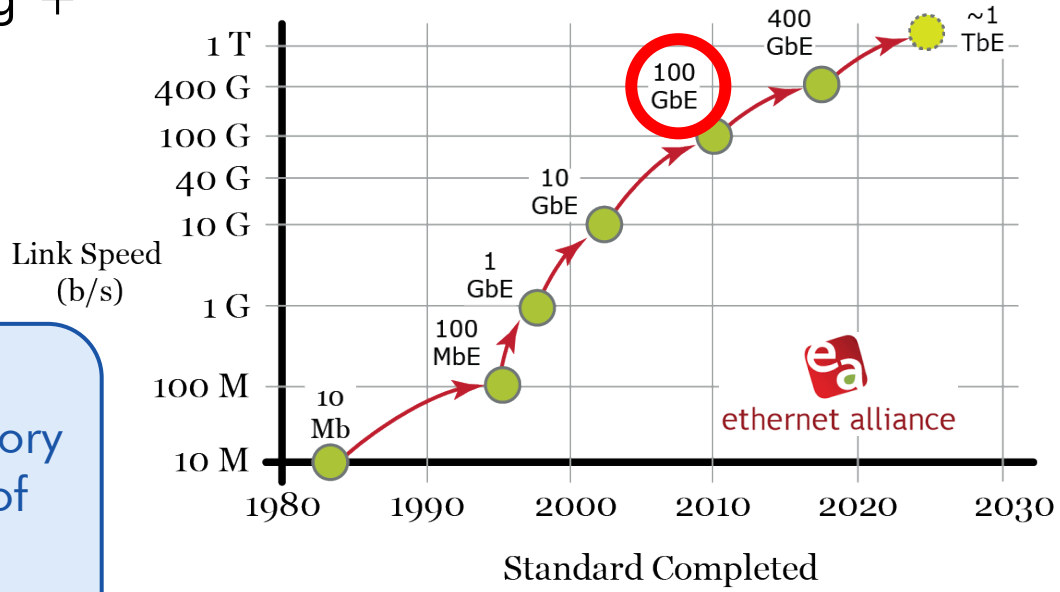
Push costly calculations into the network
and perform **stateful** functions at the processor,
which makes applications more I/O intensive.

Pressure from these trends

Every 6.72 ns a new (64-B+20-B*) packet arrives at 100 Gbps

More in-network computing + offloading capabilities

Faster link speeds



Multi-hundred-gigabit networks cannot tolerate memory access and interarrival time of packets continues to **shrink**

* 7B preamble + 1B start-of-frame delimiter +12B inter-frame gap = 20B

DCA matters because

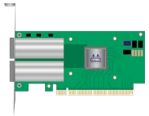
Without DCA we are unable to process I/O at line rate, thus *increasing* packet loss or latency when utilizing multi-hundred-gigabit networks.

Forwarding Packets at 100 Gbps

Packet Generator

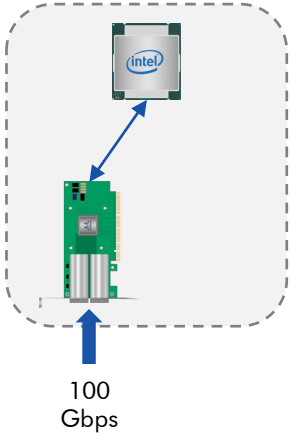
Device under Test Forwarding Packets

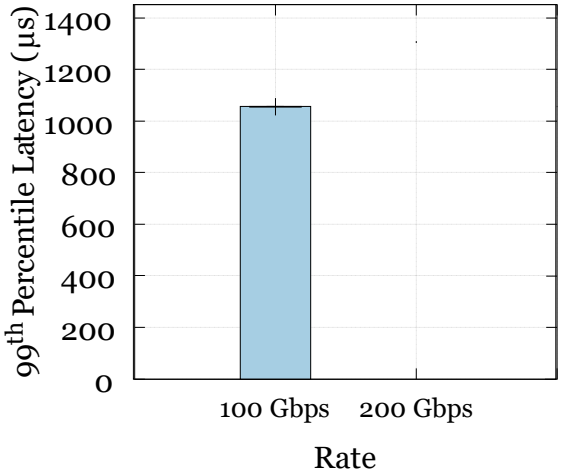
Intel Xeon Gold 6140



Mellanox ConnectX-5

Each NIC is placed in a PCIe 3.0 16x slot*



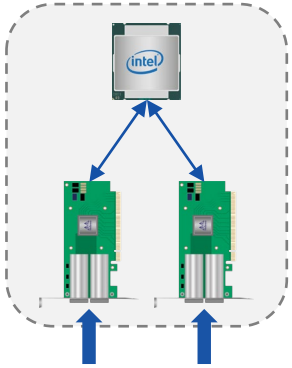


* A PCIe 3.0 16x slot is capable of providing ~125 Gbps effective full-duplex bandwidth.

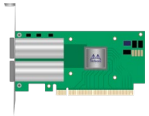
What happens at 200 Gbps?

Packet Generator

Device under Test Forwarding Packets



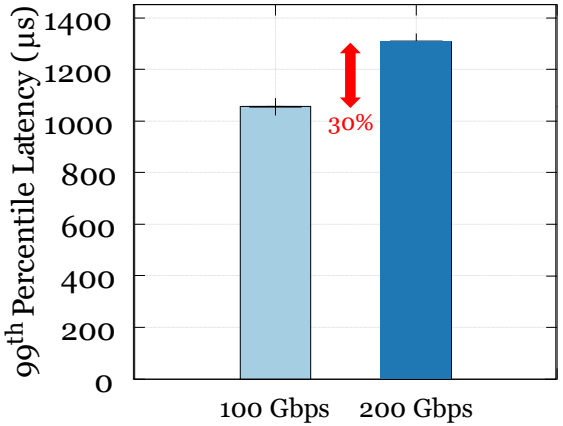
Intel Xeon Gold 6140



Mellanox ConnectX-5

Each NIC is placed in a PCIe 3.0 16x slot*

When forwarding at 200 Gbps, 30% higher latency for the NIC forwarding at 100 Gbps



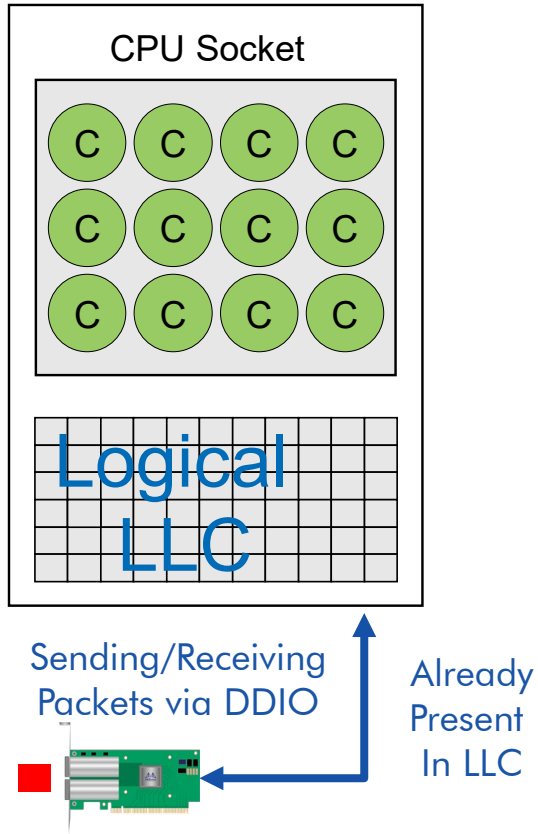
Latency of the first NIC, when forwarding at indicated aggregate rate

* A PCIe 3.0 16x slot is capable of providing ~125 Gbps effective full-duplex bandwidth.

How does DDIO work?

Writing packets/descriptors:

DDIO overwrites a cache line **if** it is already present in ***any*** LLC ways (≡ write update or hit)

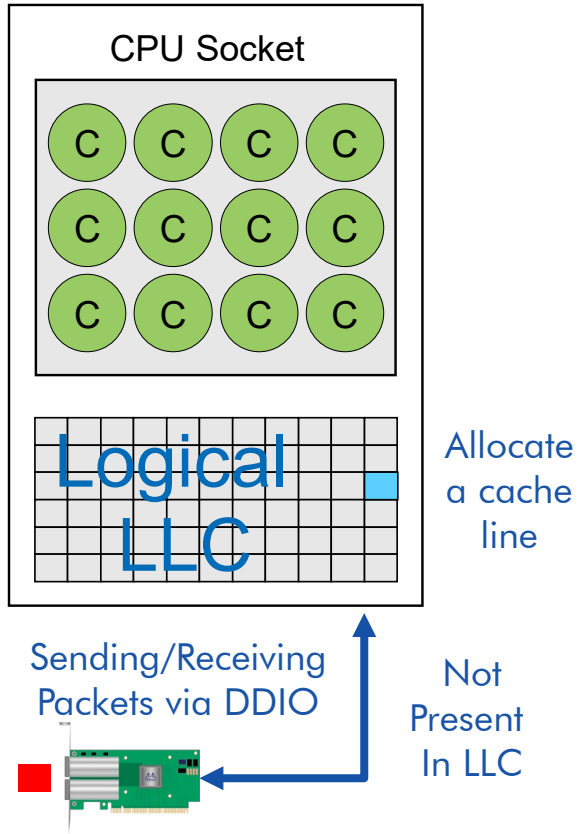


How does DDIO work?

Writing packets/descriptors:

DDIO overwrites a cache line **if** it is already present in ***any*** LLC ways (≡ write update or hit)

Otherwise, DDIO allocates a cache line in a limited portion of LLC (≡ write allocate or miss)



How does DDIO work?

Writing packets/descriptors:

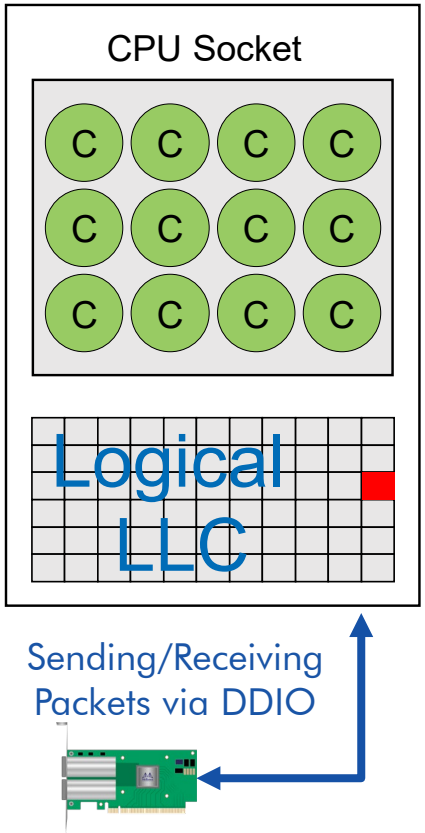
DDIO overwrites a cache line **if** it is already present in ***any*** LLC ways (≡ write update or hit)

Otherwise, DDIO allocates a cache line in a limited portion of LLC (≡ write allocate or miss)

Reading packets/descriptors:

NIC reads a cache line if it is already present in *any* LLC ways (≡ read hit)

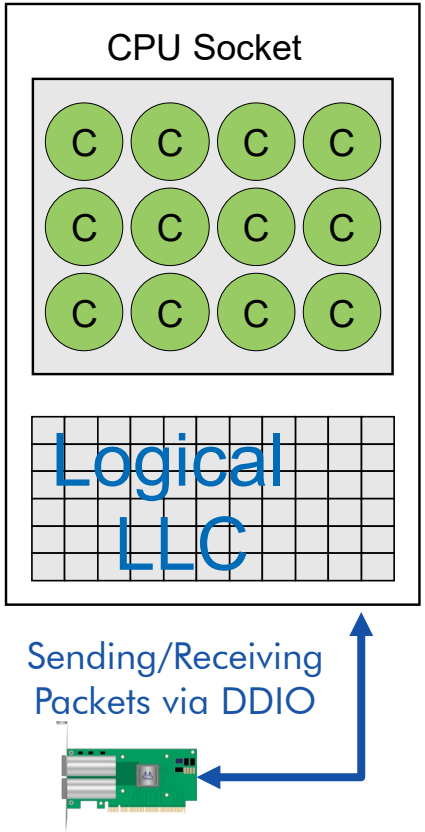
Otherwise, NIC reads it from main memory (≡ read miss)



How does DDIO work?

Designed a set of micro-benchmarks to learn about DDIO:

- Which ways are used for allocation?
- How does DDIO interact with other applications?
- Does DMA via a remote CPU socket pollute LLC?

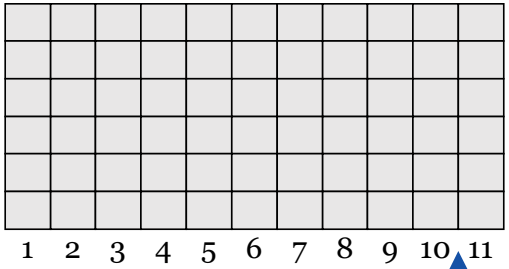


LLC ways used by DDIO

I/O Application

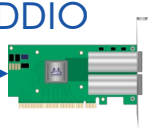
C0

Logical LLC



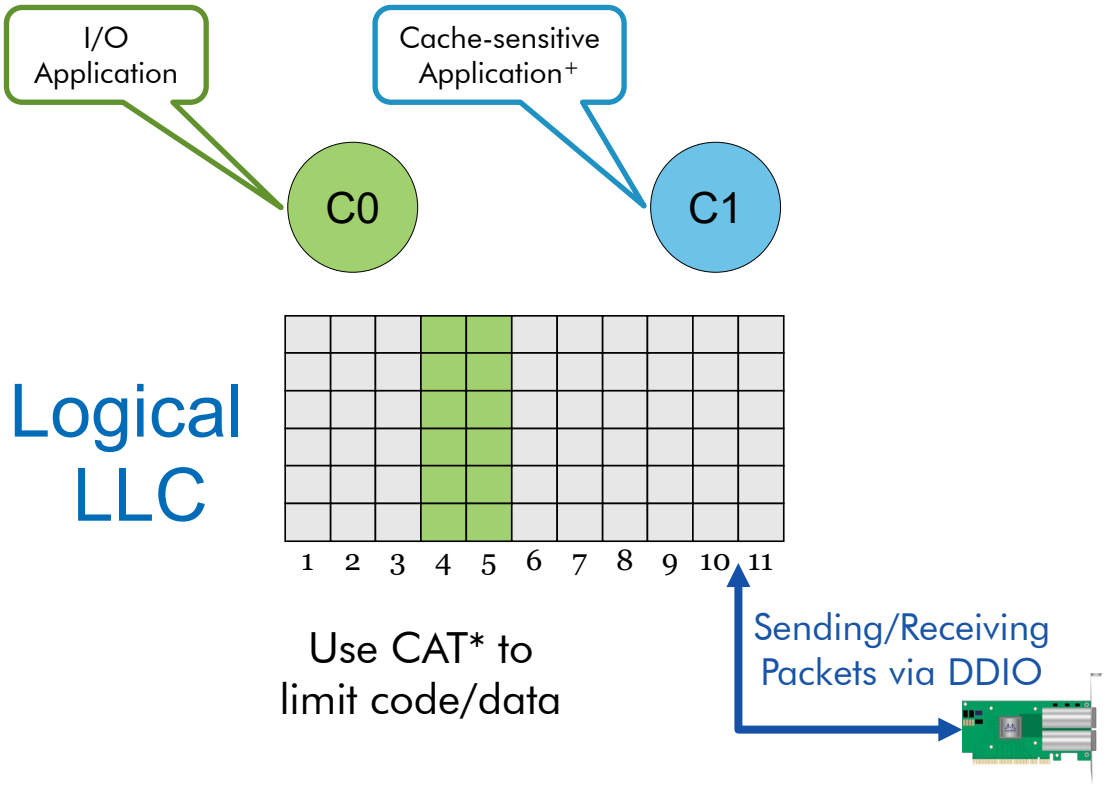
Use CAT* to limit code/data

Sending/Receiving Packets via DDIO



* Cache Allocation Technology

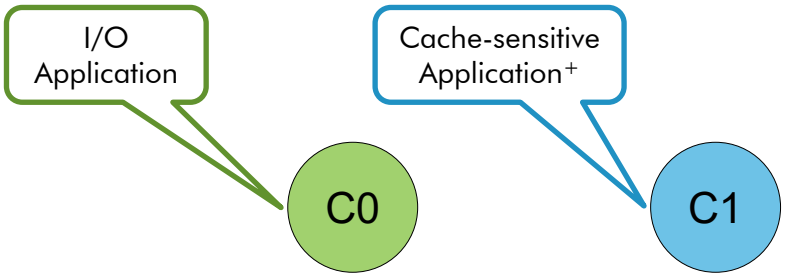
LLC ways used by DDIO



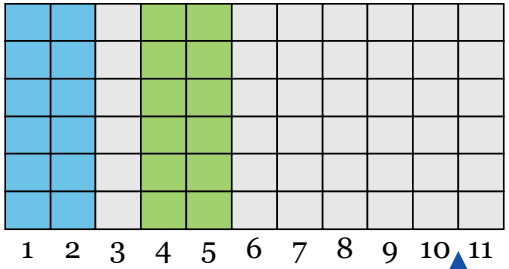
* Cache Allocation Technology

+ water_nsquared from Splash-3 benchmark

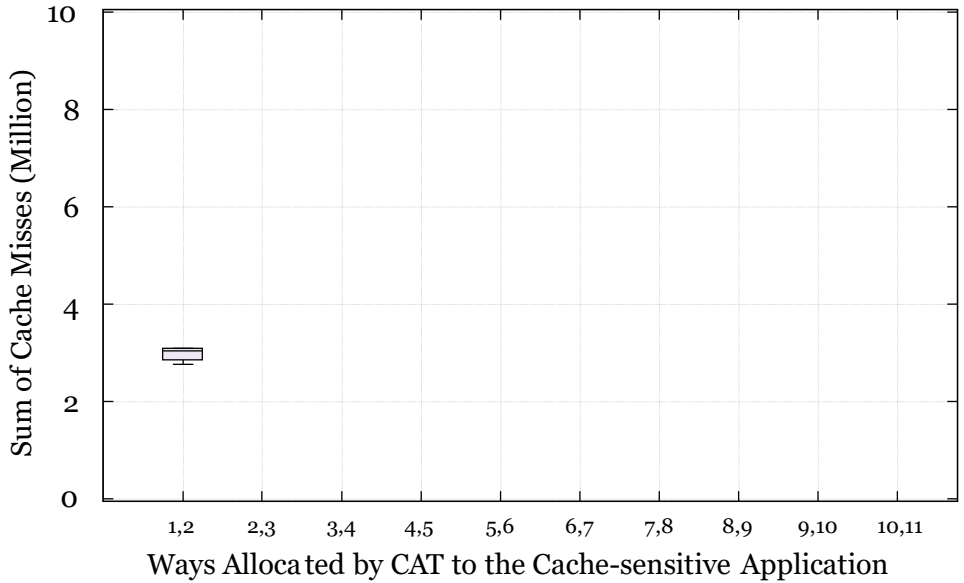
LLC ways used by DDIO



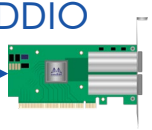
Logical LLC



Use CAT* to limit code/data



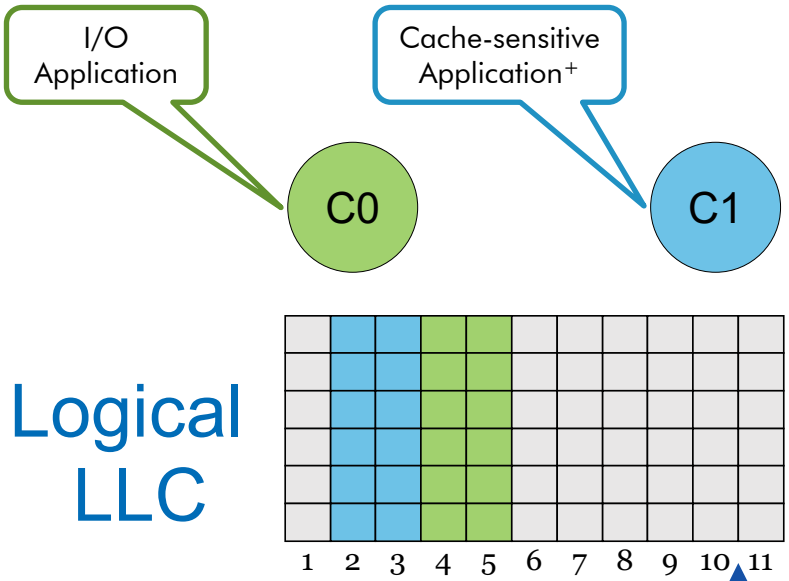
Sending/Receiving Packets via DDIO



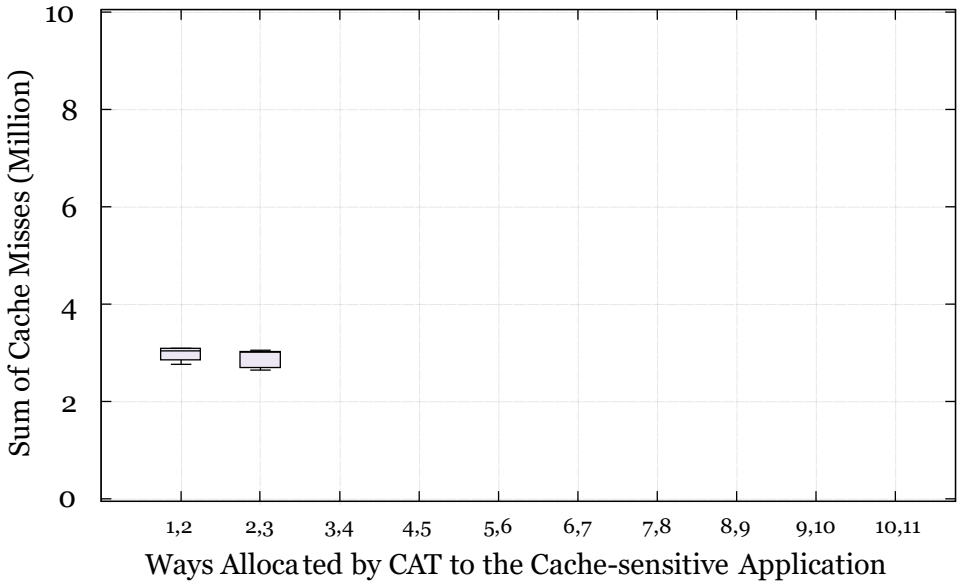
* Cache Allocation Technology

+ water_nsquared from Splash-3 benchmark

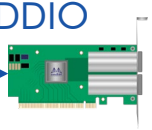
LLC ways used by DDIO



Use CAT* to limit code/data



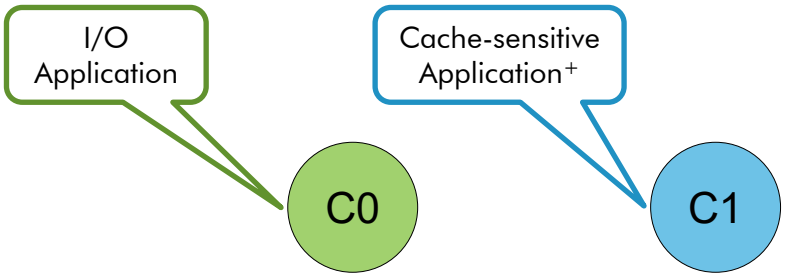
Sending/Receiving Packets via DDIO



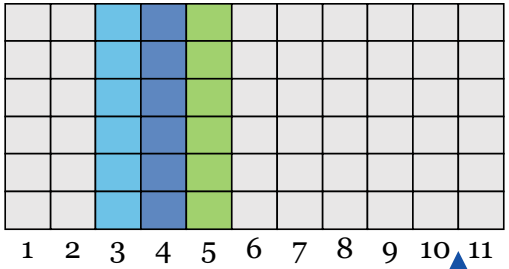
* Cache Allocation Technology

+ water_nsquared from Splash-3 benchmark

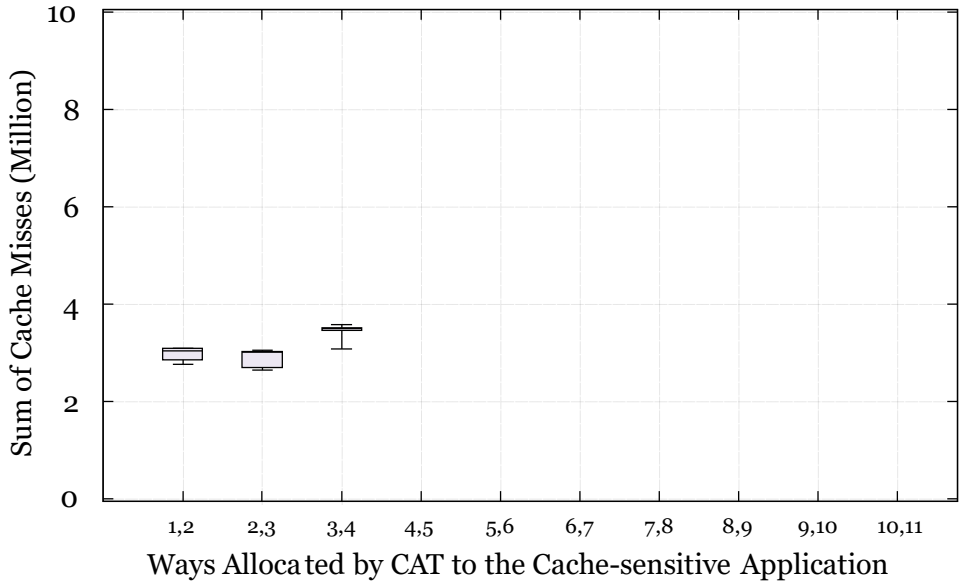
LLC ways used by DDIO



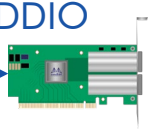
Logical LLC



Use CAT* to limit code/data



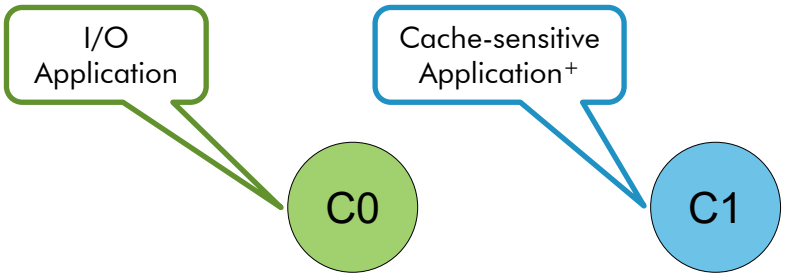
Sending/Receiving Packets via DDIO



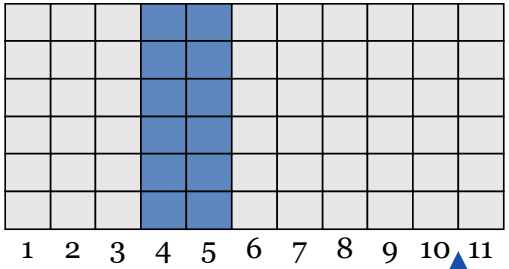
* Cache Allocation Technology

+ water_nsquared from Splash-3 benchmark

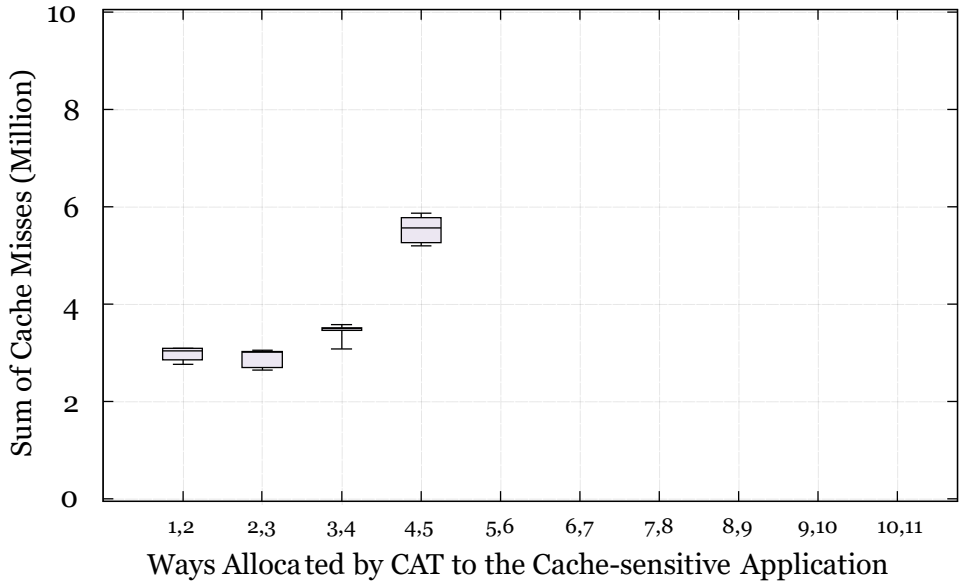
LLC ways used by DDIO



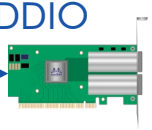
Logical LLC



Use CAT* to limit code/data



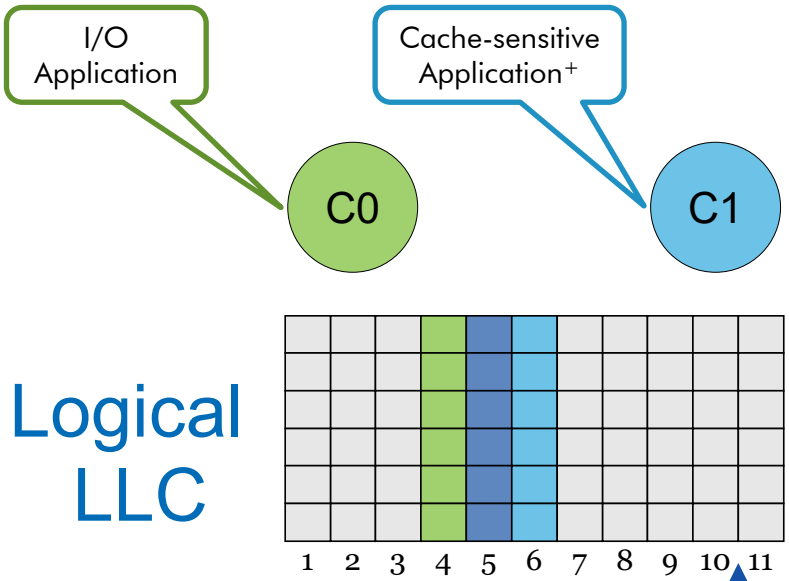
Sending/Receiving Packets via DDIO



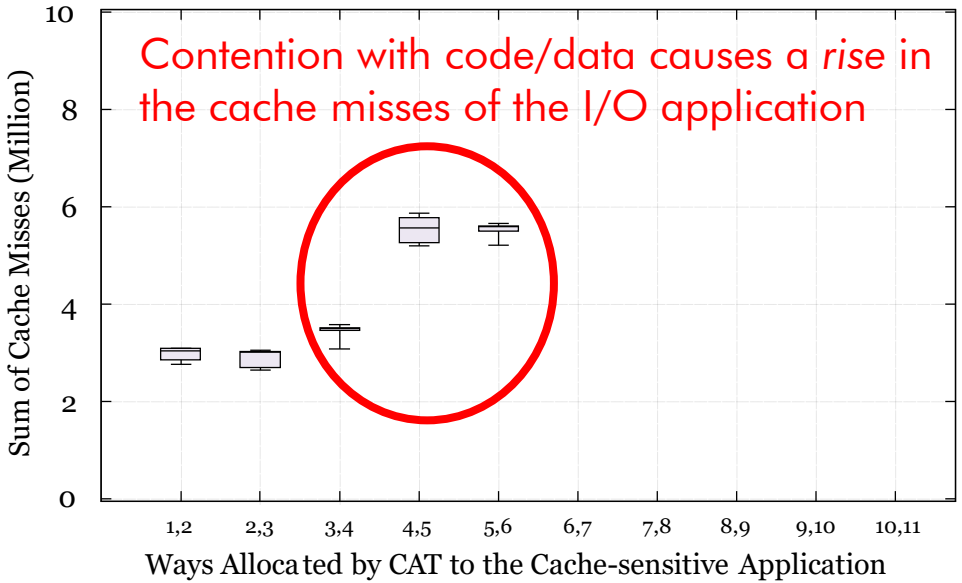
* Cache Allocation Technology

+ water_nsquared from Splash-3 benchmark

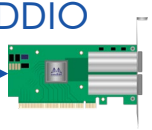
LLC ways used by DDIO



Use CAT* to limit code/data



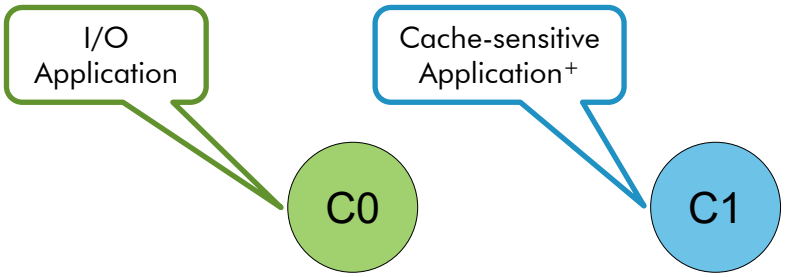
Sending/Receiving Packets via DDIO



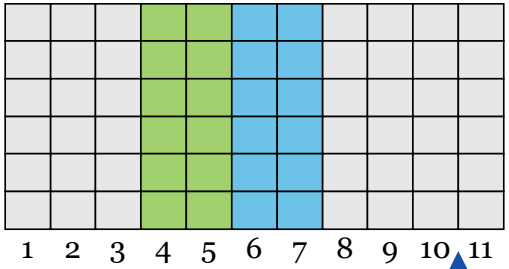
* Cache Allocation Technology

+ water_nsquared from Splash-3 benchmark

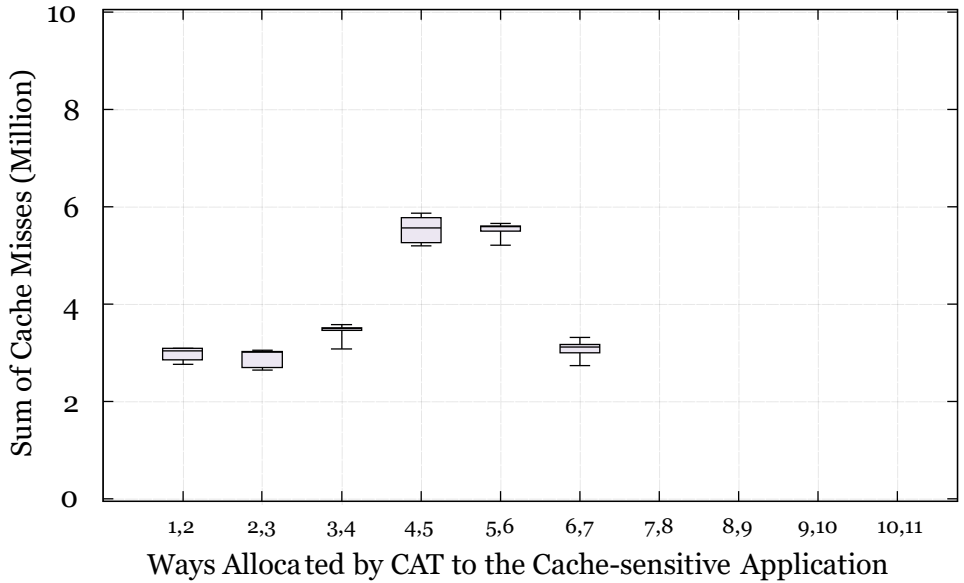
LLC ways used by DDIO



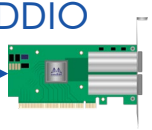
Logical LLC



Use CAT* to limit code/data



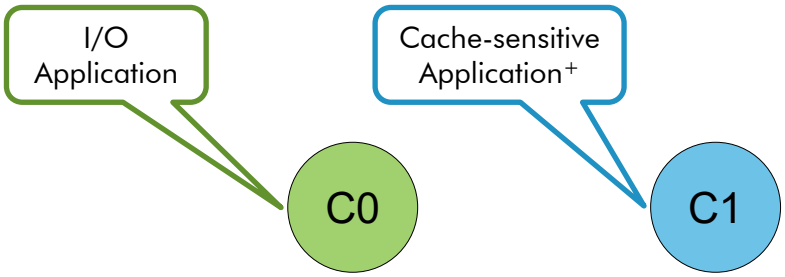
Sending/Receiving Packets via DDIO



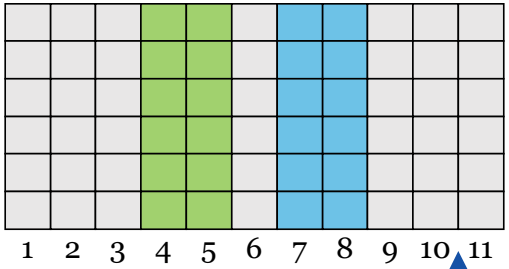
* Cache Allocation Technology

+ water_nsquared from Splash-3 benchmark

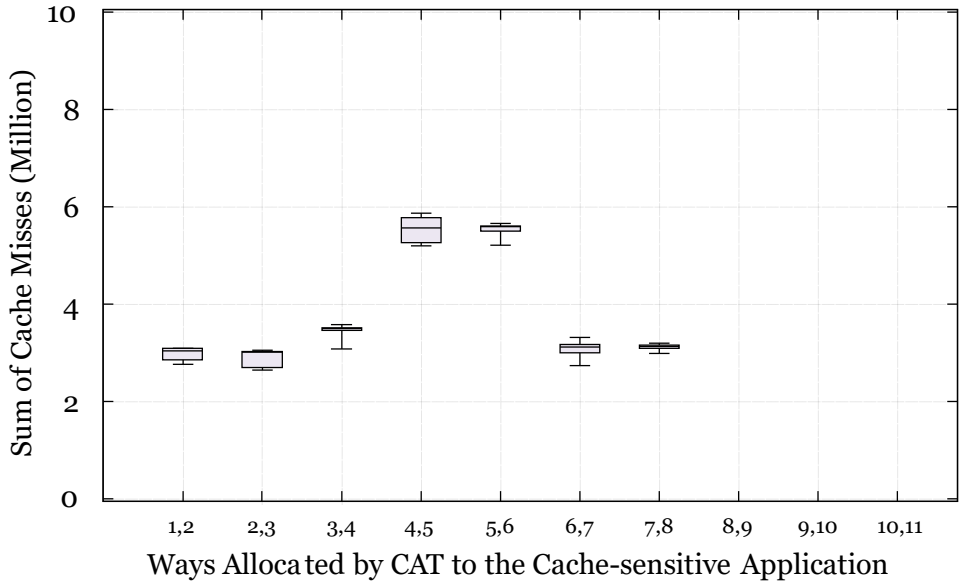
LLC ways used by DDIO



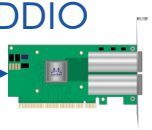
Logical LLC



Use CAT* to limit code/data



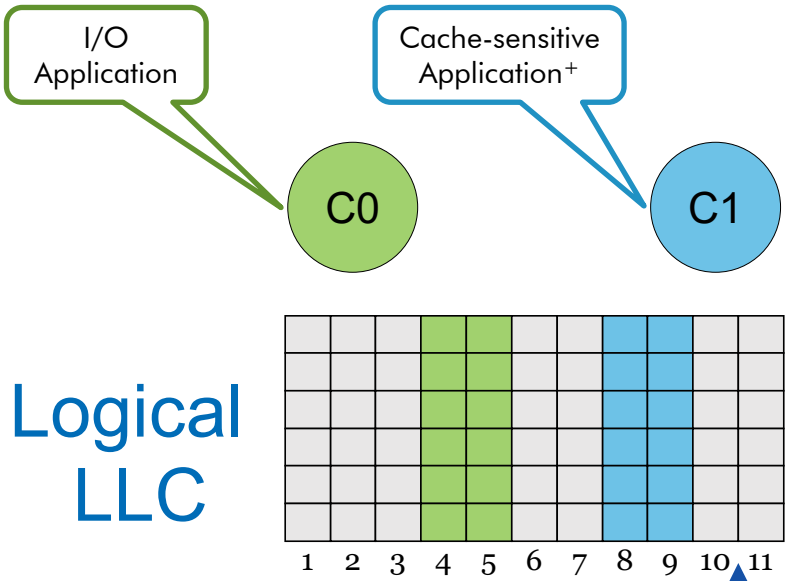
Sending/Receiving Packets via DDIO



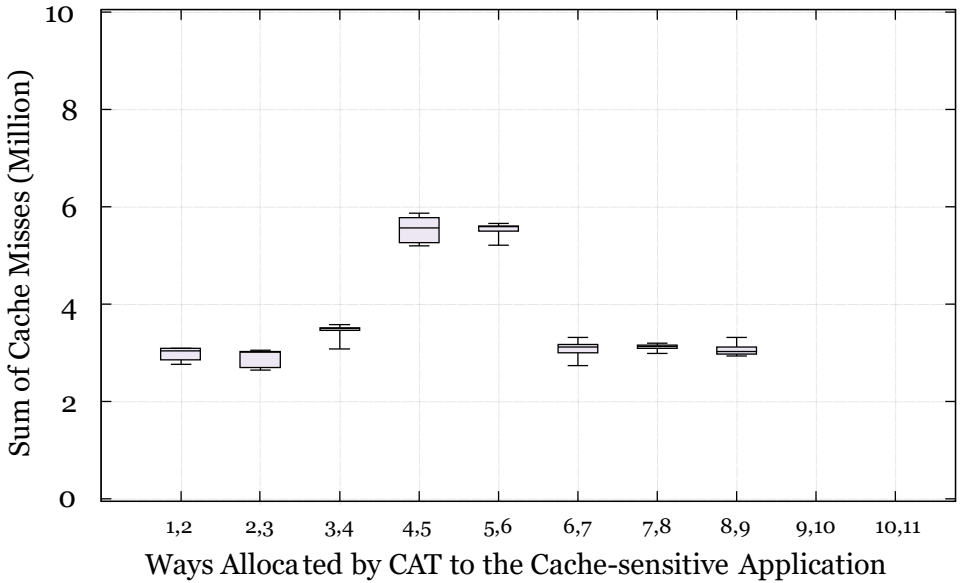
* Cache Allocation Technology

+ water_nsquared from Splash-3 benchmark

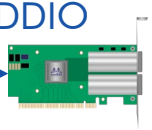
LLC ways used by DDIO



Use CAT* to limit code/data



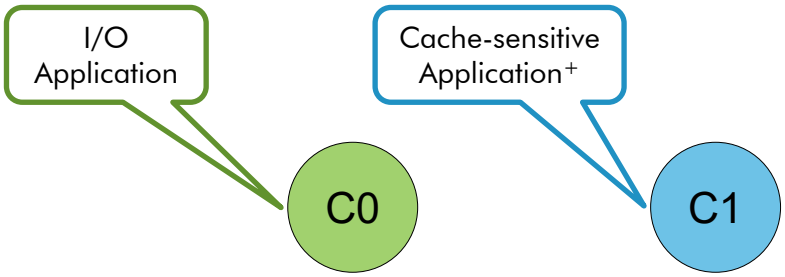
Sending/Receiving Packets via DDIO



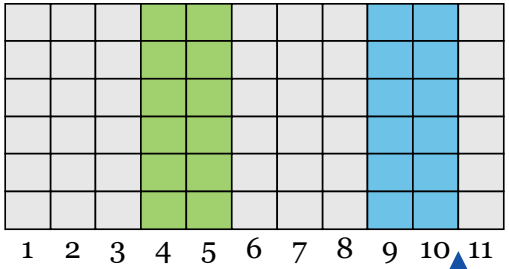
* Cache Allocation Technology

+ water_nsquared from Splash-3 benchmark

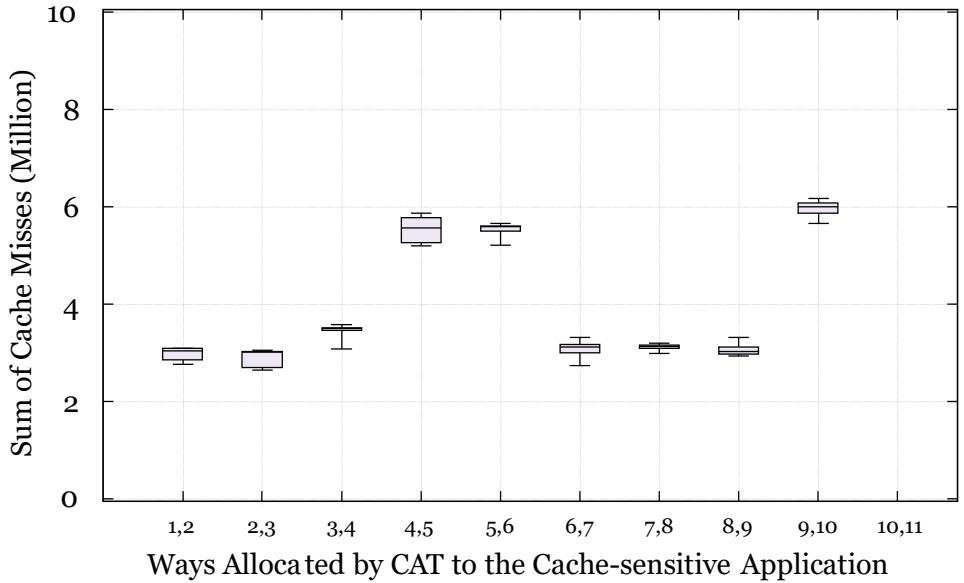
LLC ways used by DDIO



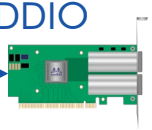
Logical LLC



Use CAT* to limit code/data



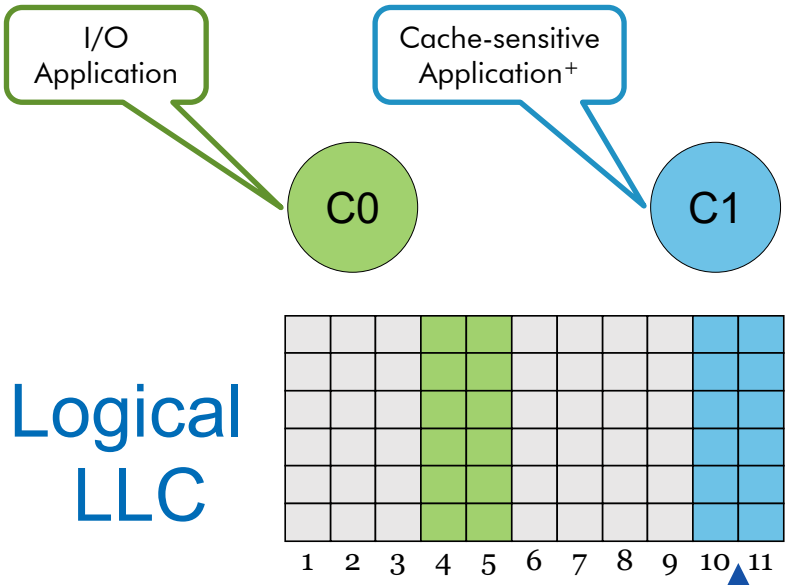
Sending/Receiving Packets via DDIO



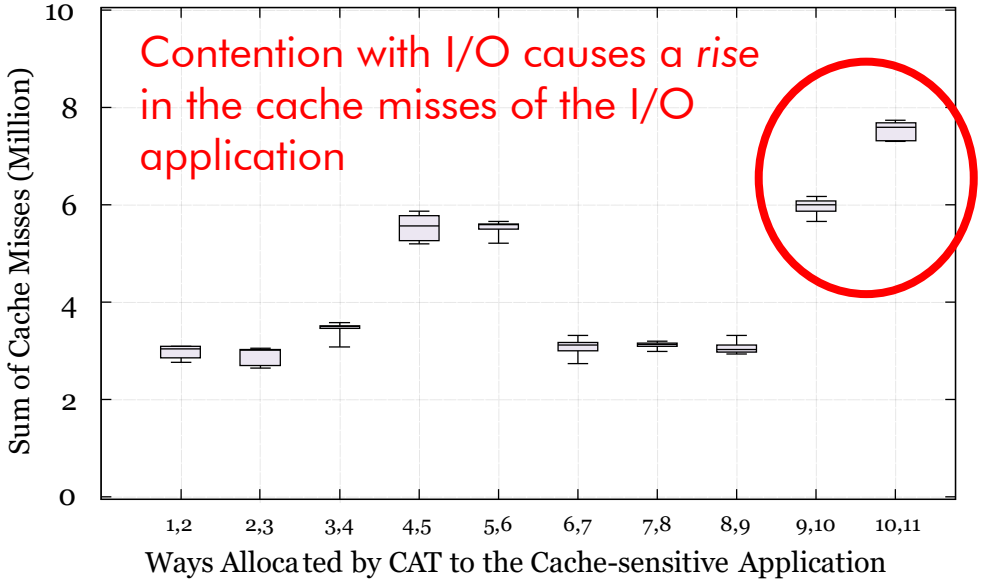
* Cache Allocation Technology

+ water_nsquared from Splash-3 benchmark

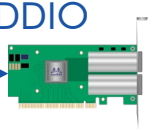
LLC ways used by DDIO



Use CAT* to limit code/data



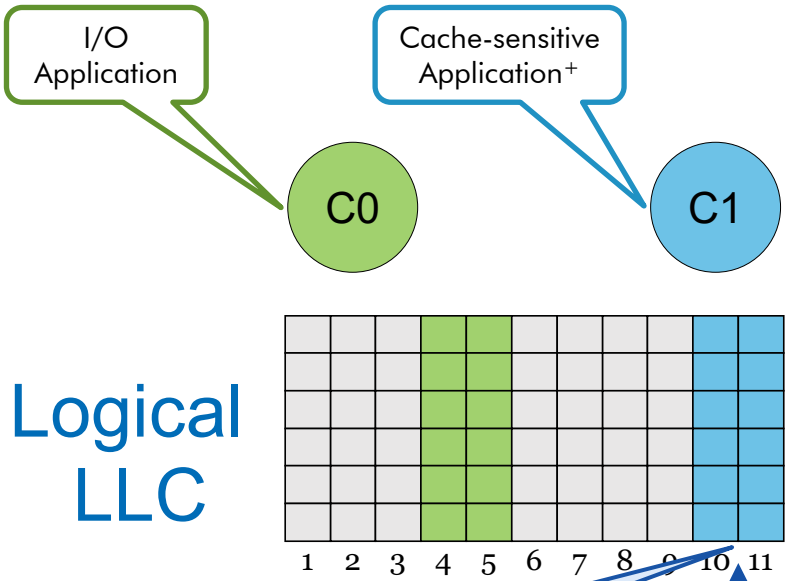
Sending/Receiving Packets via DDIO

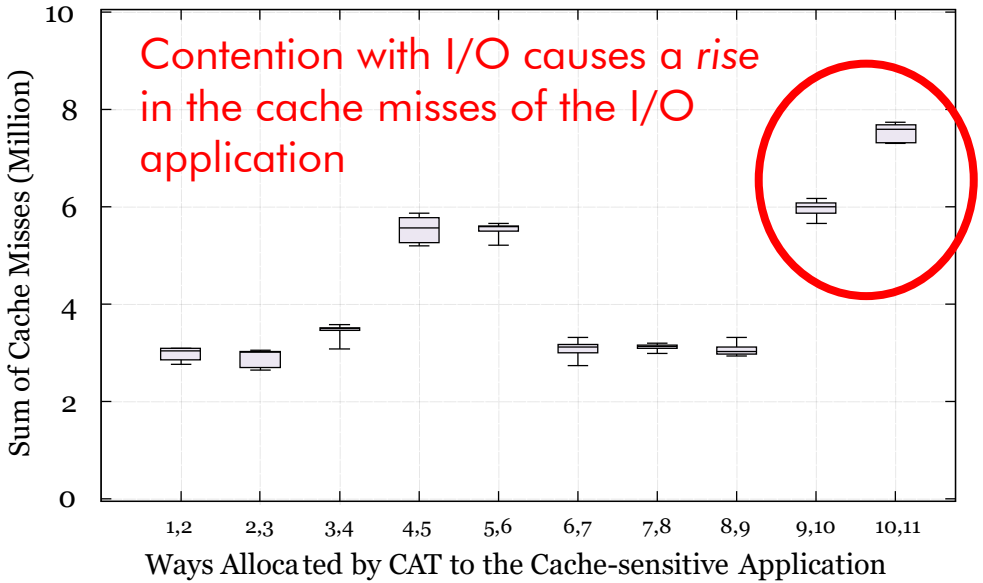


* Cache Allocation Technology

+ water_nsquared from Splash-3 benchmark

LLC ways used by DDIO





See our paper

* Cache Allocation Technology

+ water_nsquared from Splash-3 benchmark

How does DDIO perform?

DDIO *cannot* provide expected benefits!

- ResQ* [NSDI'18]
- Intel reports

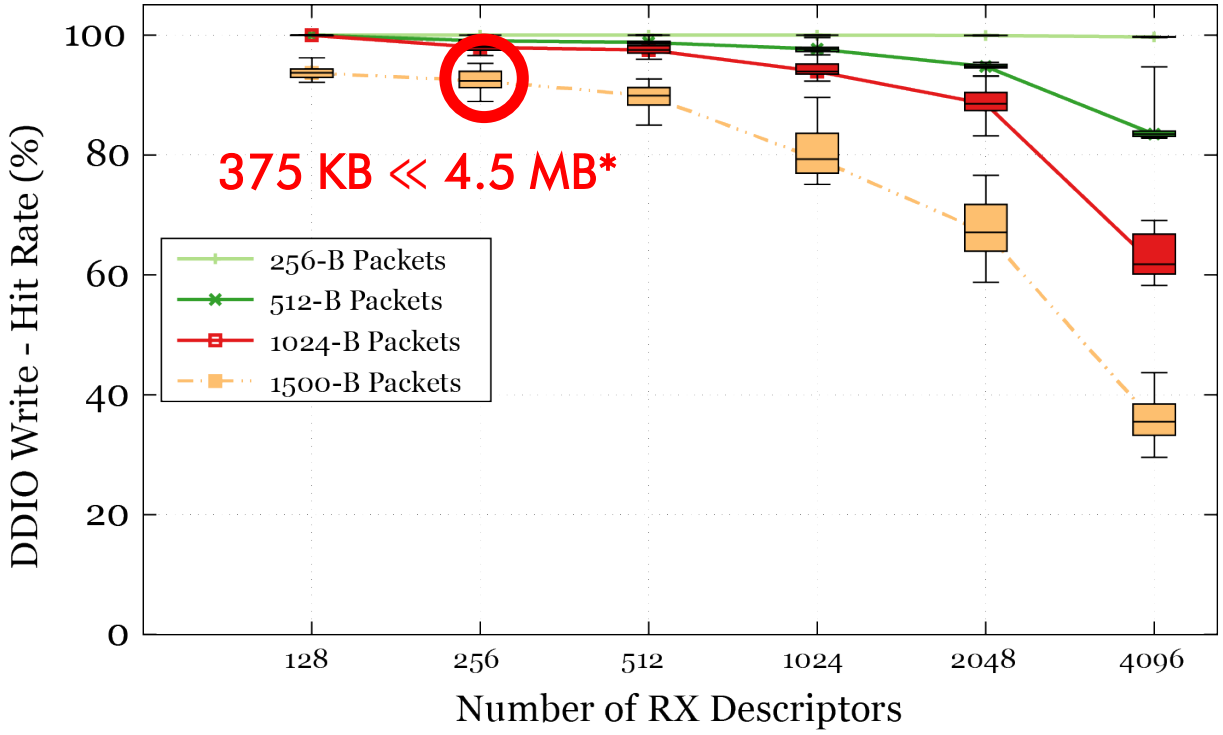
Write-allocate DDIO could evict *not-yet-processed* and *already-processed* packets from LLC

Packet should be read from main memory rather than LLC

Reduce the number of RX descriptors so that the buffer fit in the limited DDIO portion.

* ResQ: Enabling SLOs in Network Function Virtualization

Reducing #Descriptors is Not Sufficient! (1/2)

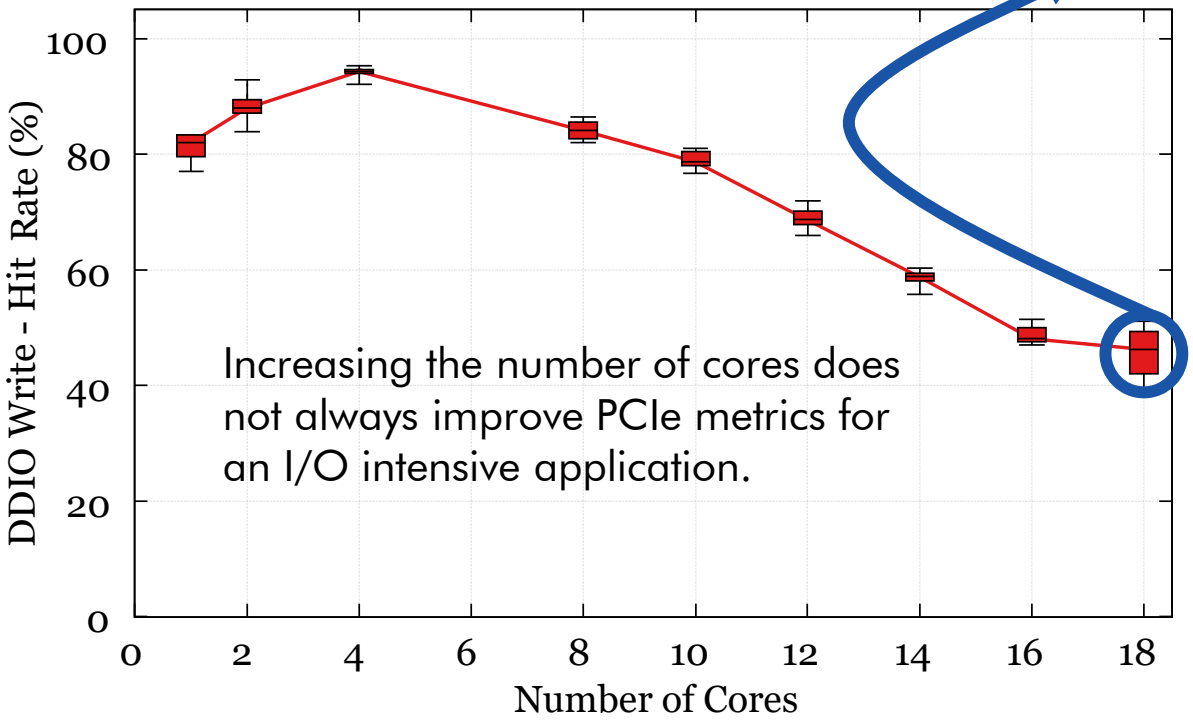


Increasing the number of RX descriptors and packet size adversely affects the performance of DDIO

DDIO cannot use the whole reserved capacity in LLC

* DDIO uses 2 ways out of 11 ways, i.e., 24.75 MB x 2 / 11 = 4.5 MB

Reducing # Descriptors is Not Sufficient! (2/2)



1500-B Packets x 256 x 18
≈ **6.59 MB** >> 4.5 MB

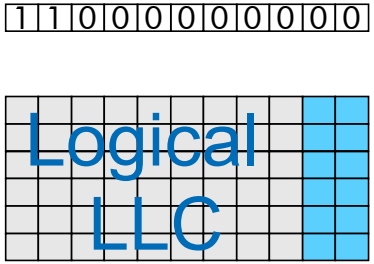
Forwarding 1500-B Packets at 100 Gbps with 256 per-core RX descriptors

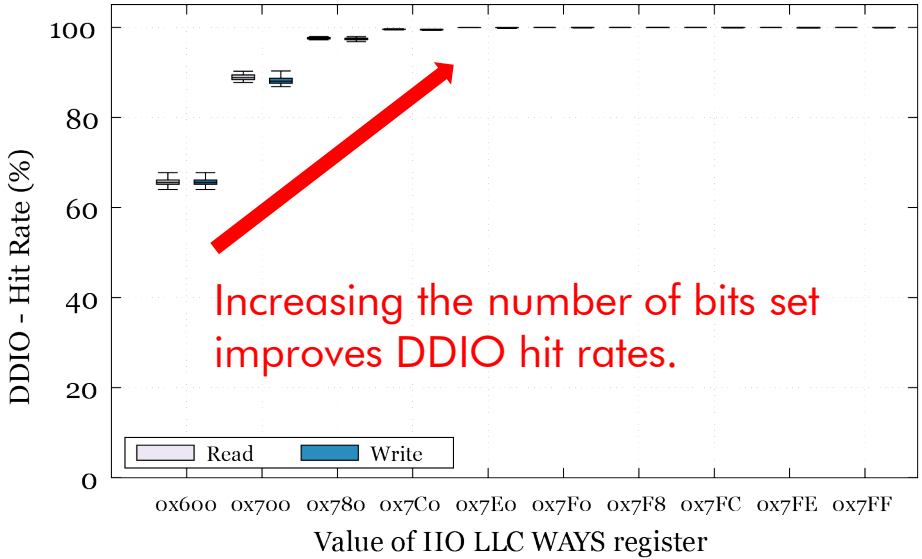
DDIO should be able to perform well with high number of RX descriptors!

IIO LLC WAYS Register

Tuning a little-discussed register can improve the performance of DDIO

Default value is 0x600

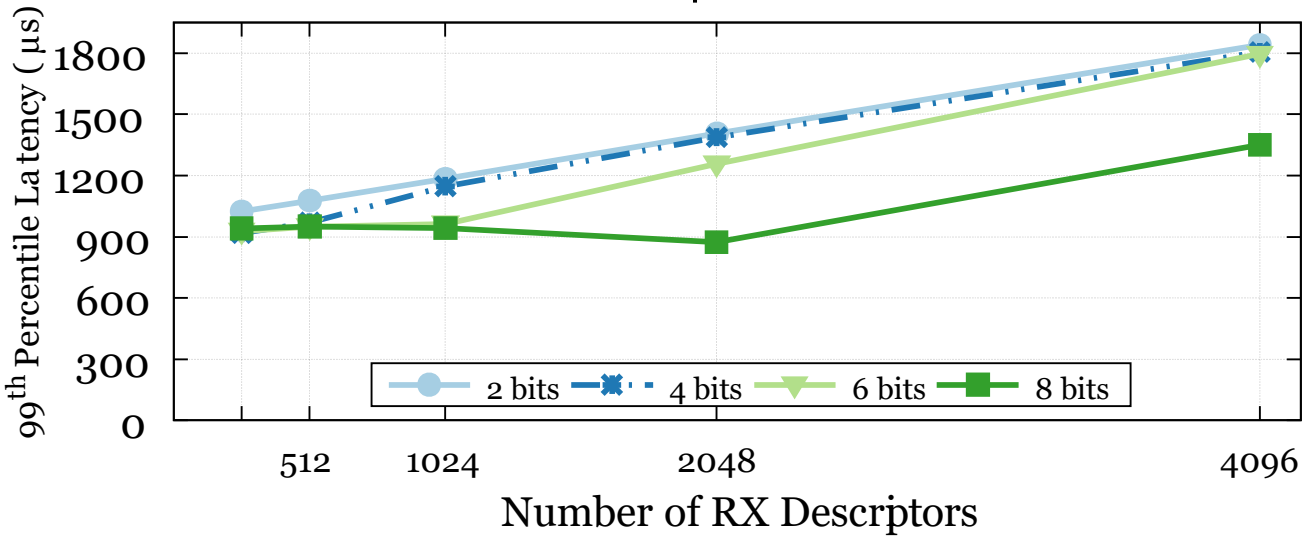




Impact of Tuning DDIO

DDIO's effect on hit rates can affect application-level performance based on an application's characteristics

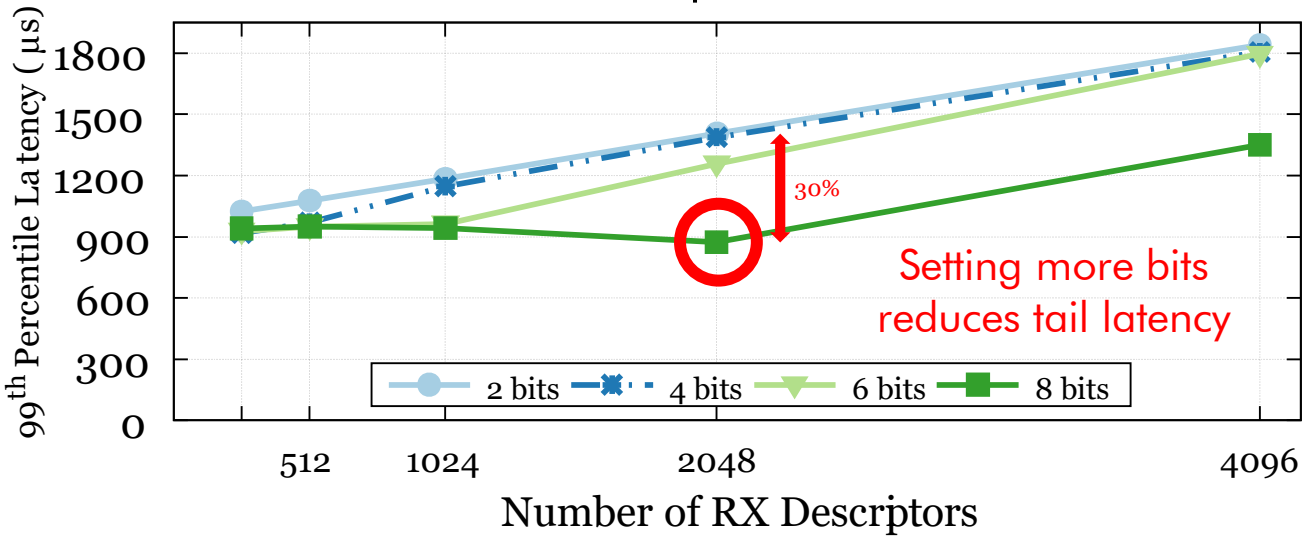
For example, an I/O intensive application: 2 cores forwarding 1500-B Packets at 100 Gbps



Impact of Tuning DDIO

DDIO's effect on hit rates can affect application-level performance based on an application's characteristics

For example, an I/O intensive application: 2 cores forwarding 1500-B Packets at 100 Gbps



Is Tuning DDIO Enough?

Tuning is **not** a perfect solution! Due to:

- Cache is used for code/data,
- Smaller per-core cache quota, and
- Coarse-grained partitions.

Next generation DCA should provide:

Fine-grained placement: Similar to CacheDirector* [EuroSys'19]
I/O isolation: Extend CAT+ and CDP++ to include I/O
Selective DCA/DMA: only transfer relevant parts of the packet to LLC

+ Cache Allocation Technology ++ Code/Data Prioritization

* Make the Most out of Last Level Cache in Intel Processors

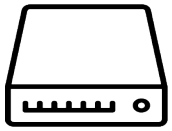
What about Current Systems?

DMA should **not** be directed to the cache if this would cause I/O evictions!

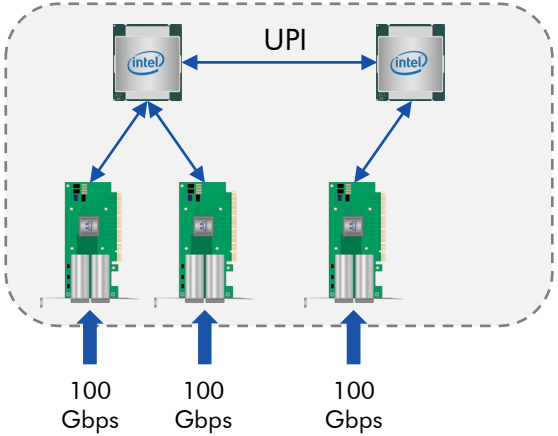
Bypassing cache is beneficial in multi-tenant/application environment, where some performance isolation is desired.

- Disabling DDIO for a specific PCIe port
- Exploiting a remote socket

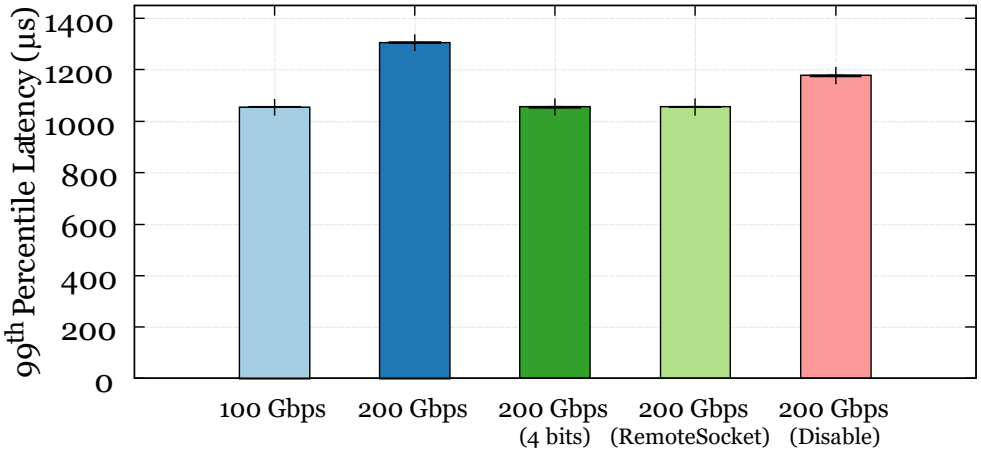
Using Our Knowledge for 200 Gbps



Device under Test
Forwarding Packets



Tuning DDIO improves packet processing at 200 Gbps



Latency of the first NIC versus aggregate Rate

Better cache management is necessary for multi-hundred-gigabit-per-second networks

Other Insights

See our paper for more results about:

- How does receiving rate affect the DDIO performance?
- How does processing time affect the DDIO performance?
- Is DDIO always beneficial?
- Scaling up and DDIO.

We study the performance of DDIO in different scenarios

Our Key Findings (1/2)

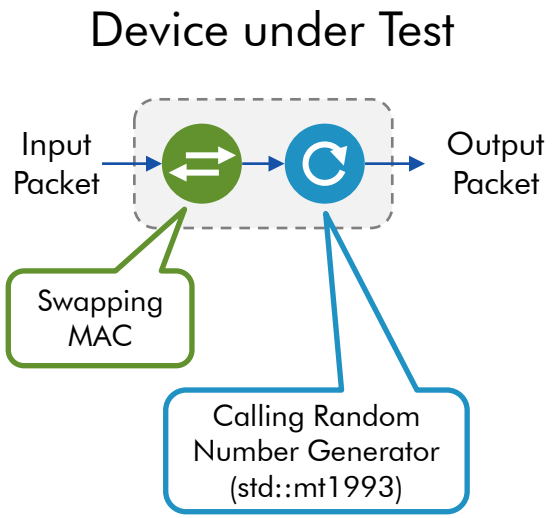
- If an application is I/O bound, adding excessive cores could degrade its performance.
- If an application is I/O bound, tuning a little-discussed register called IIO LLC WAYS could improve performance and lead to the same improvements as adding more cores.
- If an application starts to become CPU bound, adding more cores could improve its throughput, but it is important to balance load among cores to maximize DDIO's benefits.
- Getting close to ~100 Gbps can cause DDIO to become a bottleneck. Therefore, it is essential to know when to bypass the cache to realize performance isolation.

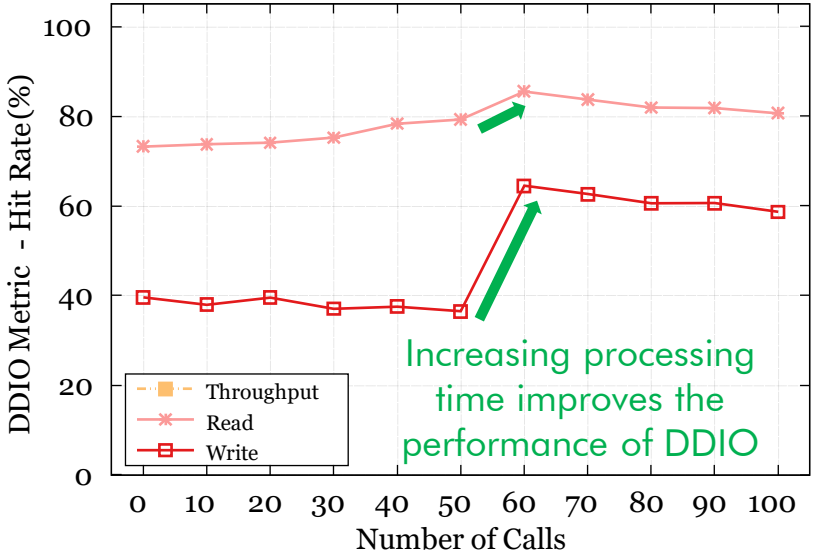
Our Key Findings (2/2)

- If an application is truly CPU/memory bound, tuning DDIO is less efficient.

We now explain the impact of processing time
on the performance DDIO, which resulted in this finding.

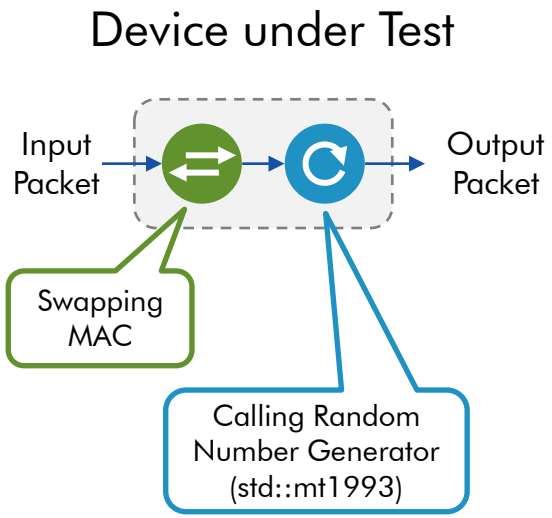
Impact of Processing Time

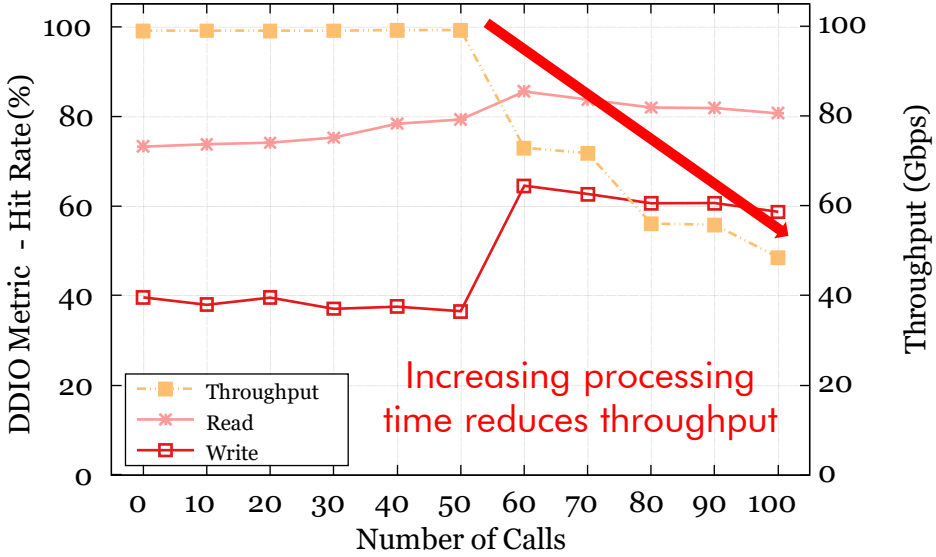




Increasing processing time improves DDIO performance

Impact of Processing Time





DDIO performance **matters most** when an application is **I/O bound**, rather than CPU/memory bound.

Conclusion

- DCA/DDIO should be tuned for I/O intensive applications.
- DCA/DDIO needs to be rearchitected for multi-hundred-gigabit networks.
- Benchmark your testbed with our source code.

https://github.com/aliireza/ddio-bench

This work is supported by ERC, SSF, and WASP.

Thanks for listening

Do not hesitate to contact us if you have any questions.

farshin@kth.se and amirrsk@kth.se