Microarchitectural Minefields:
4K-Aliasing Covert Channel and Multi-Tenant Detection in IaaS Public Clouds

DEAN SULLIVAN, ORLANDO ARIAS*, TRAVIS MEADE*, YIER JIN

UNIVERSITY OF FLORIDA, *UNIVERSITY OF CENTRAL FLORIDA

TL;DR

4K-Aliasing timing channel:

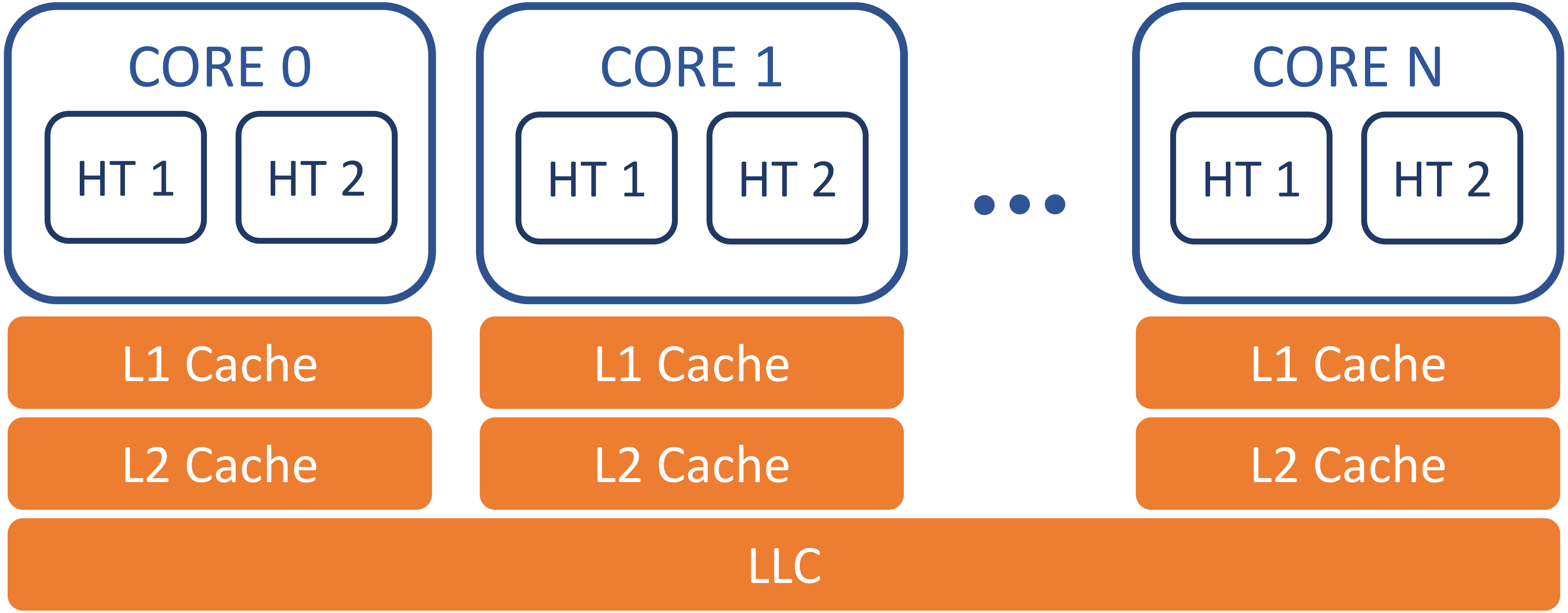
- Speculatively executed younger writes falsely aliasing with older loads
- Side effect of memory ordering in the memory order buffer
- Measurable across address spaces
  - Processes
  - Virtual machines

On public IaaS clouds:

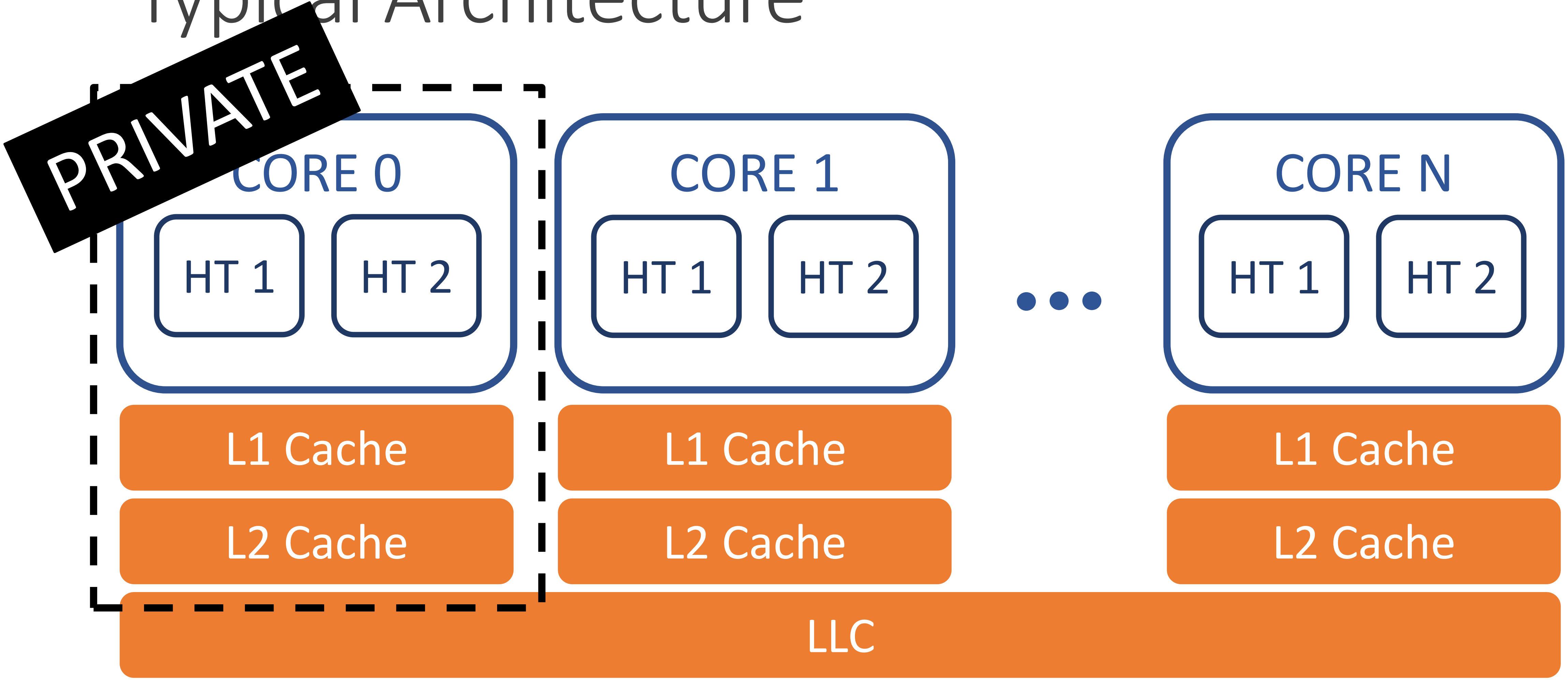
- Fast and robust covert channel
- Practical multi-tenant detection

Timing Channel Background

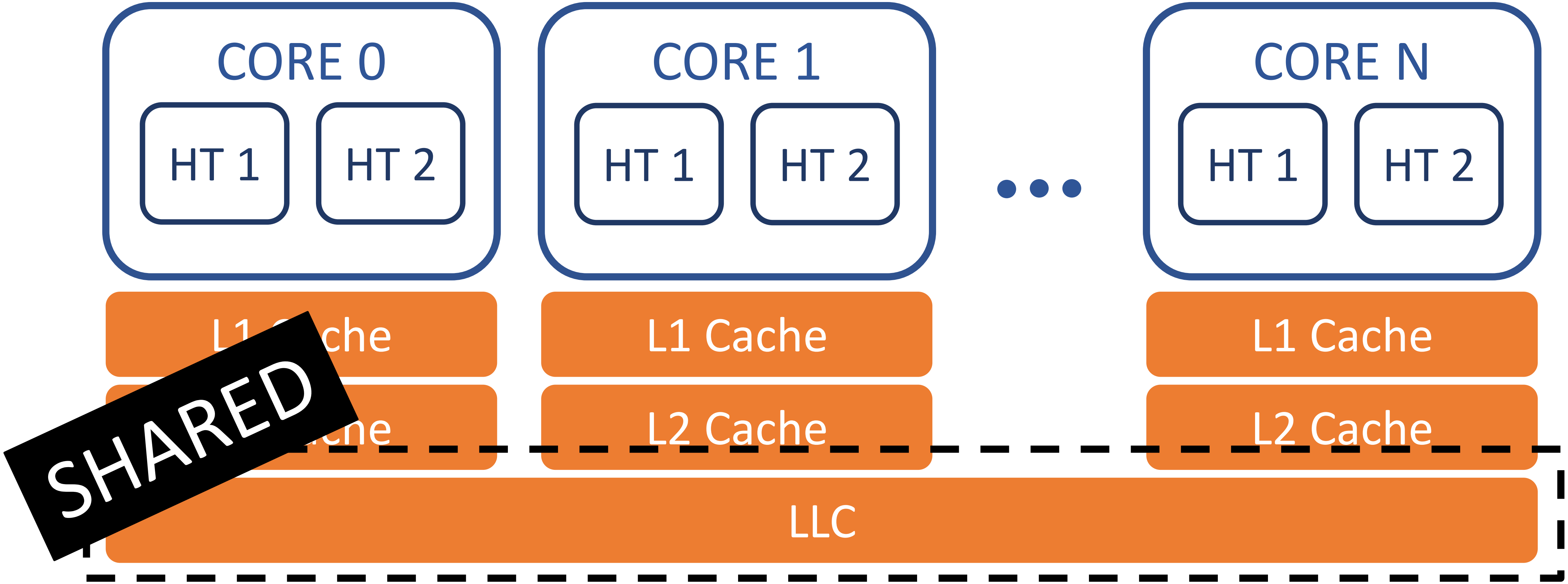
Typical Architecture



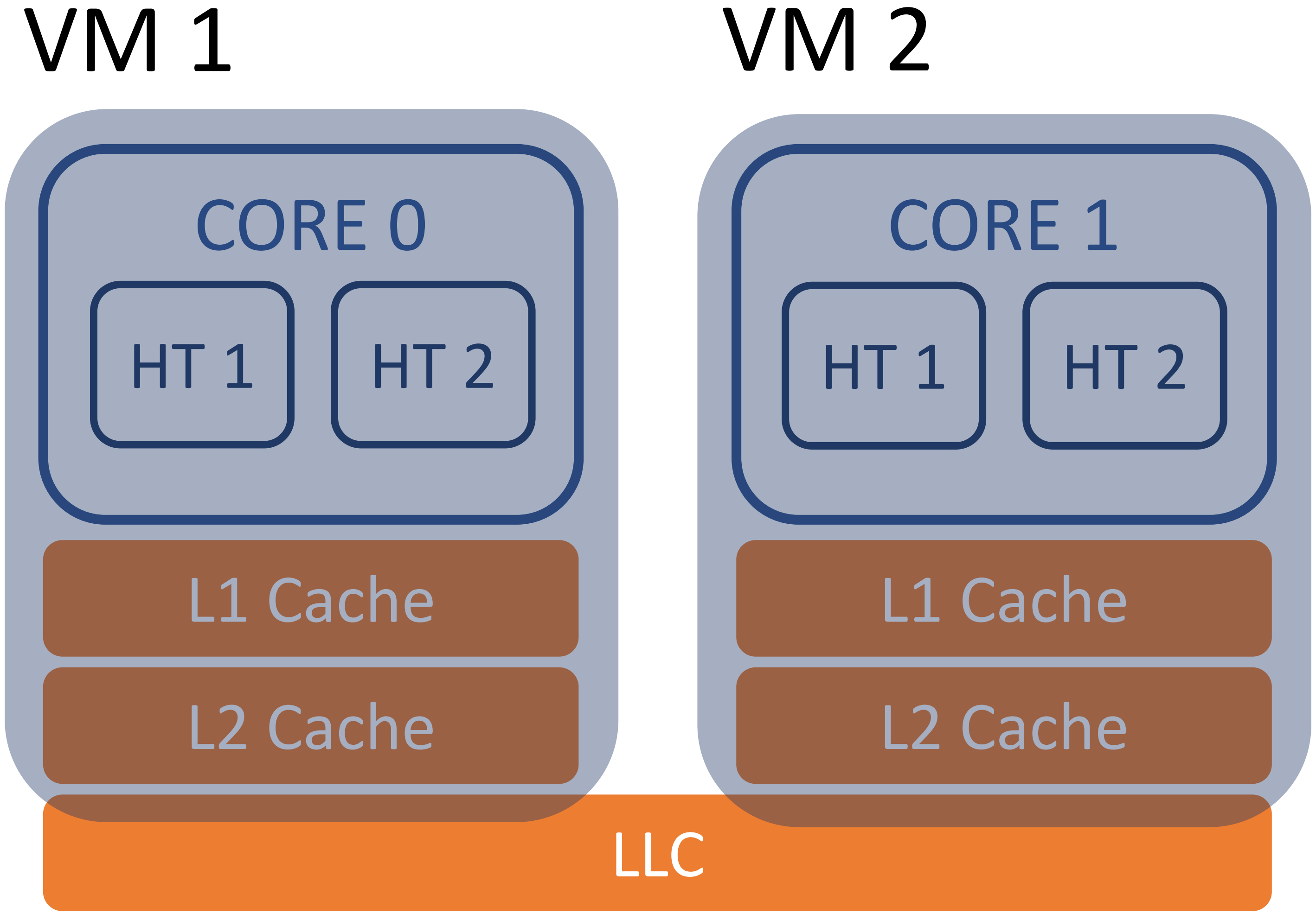
Typical Architecture



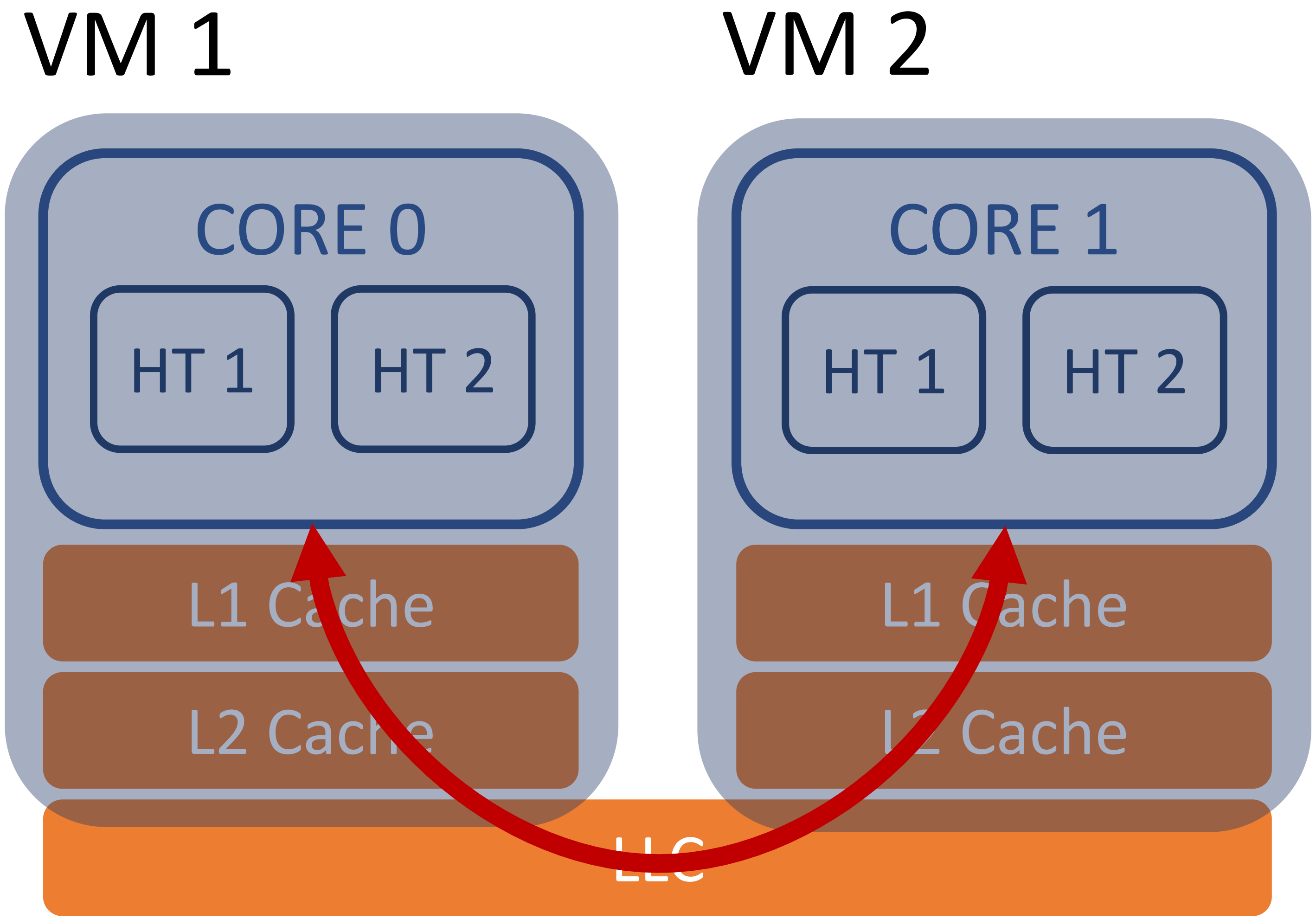
Typical Architecture



Covert Channel in the Cloud



Covert Channel in the Cloud



Covert Channel Related Works

A lot of great work has made these covert channels

- Fast
- Robust
- Practical

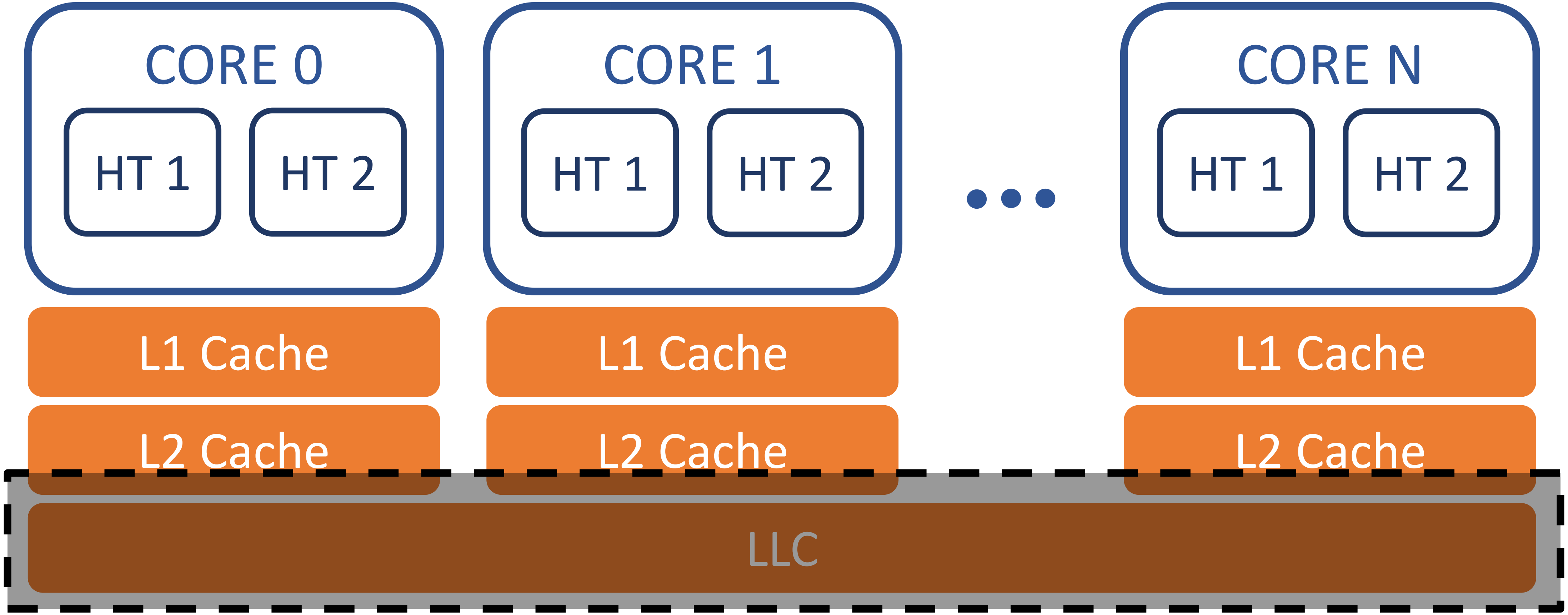
Limitations of Prior Covert Channels

Speed bounded by time to access shared resource

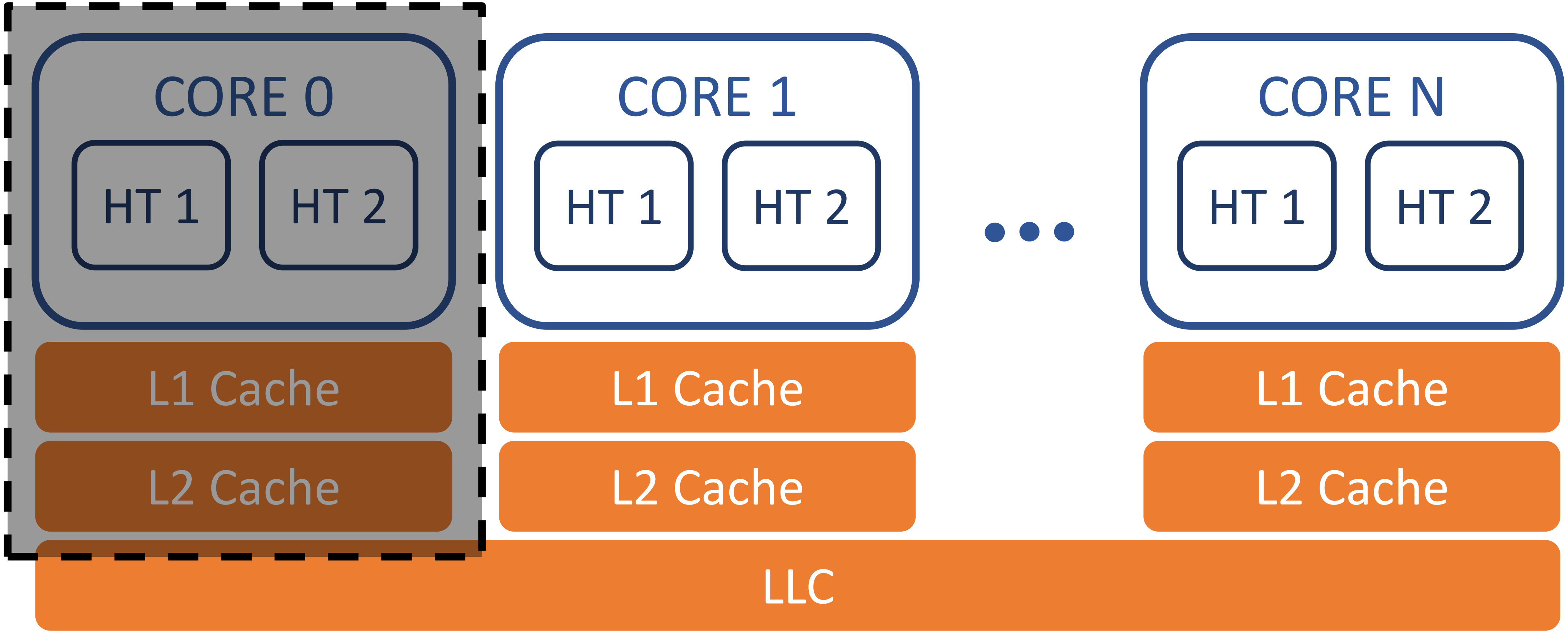
Susceptible to detection

Can we do as good, or better, with a core-private resource?

Not this…


CORE 0
HT 1
HT 2
CORE 1
HT 1
HT 2
…
CORE N
HT 1
HT 2
L1 Cache
L1 Cache
L1 Cache
L2 Cache
L2 Cache
L2 Cache
LLC

...this

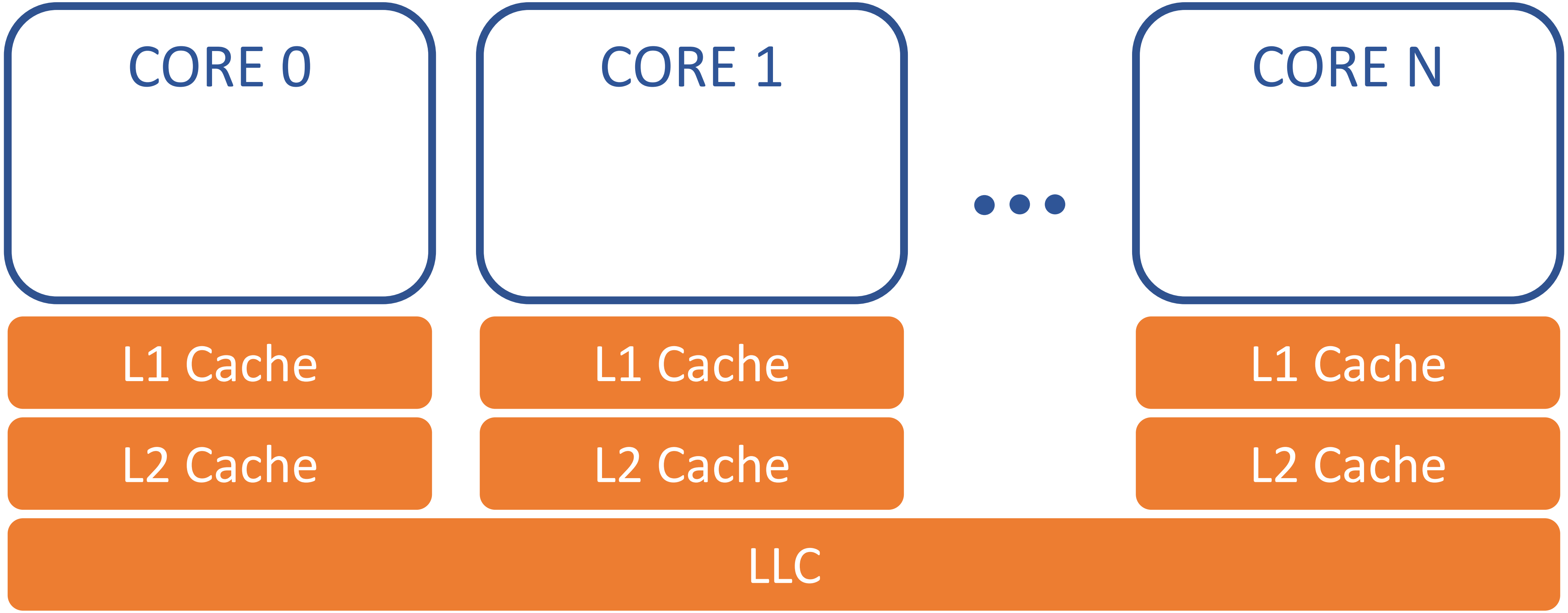

CORE 0
HT 1
HT 2
L1 Cache
L2 Cache
CORE 1
HT 1
HT 2
L1 Cache
L2 Cache
• • •
CORE N
HT 1
HT 2
L1 Cache
L2 Cache
LLC

Why?

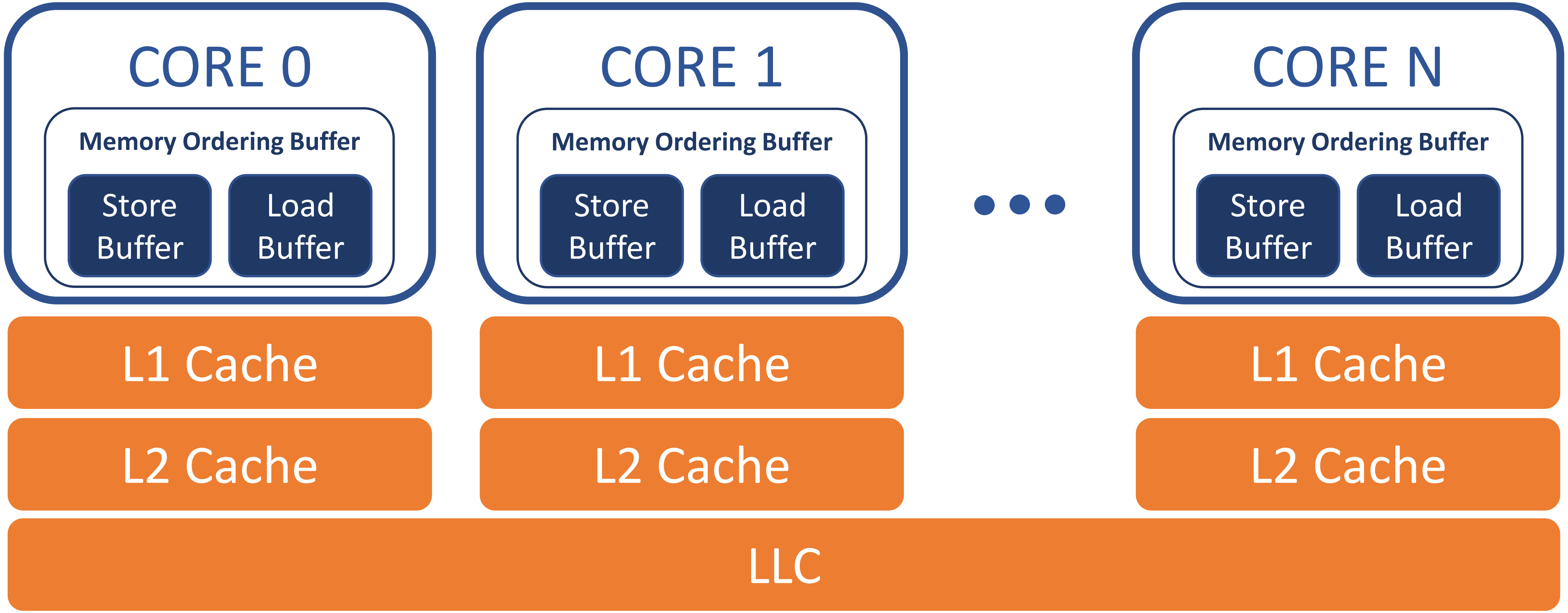
Faster? → Send more!

Core private? → Avoid detection!

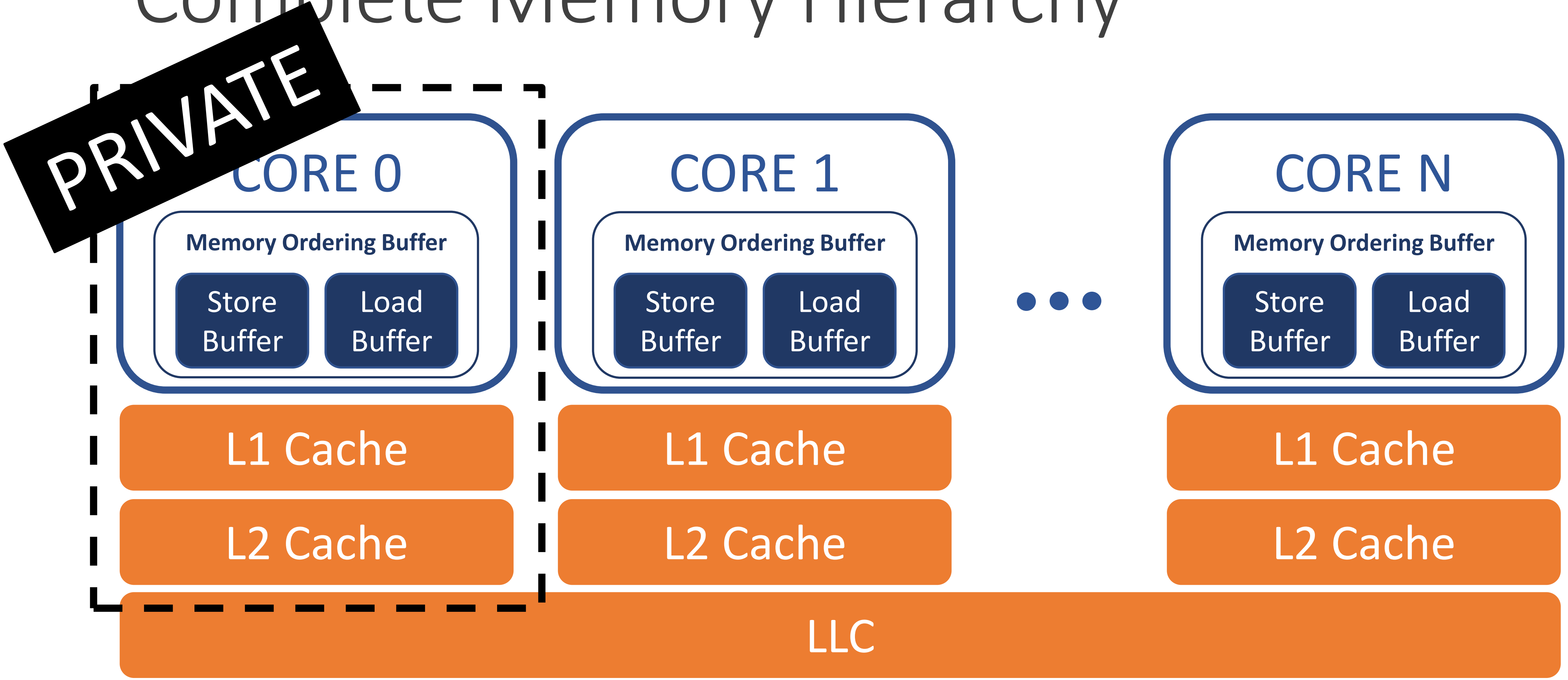
Partial Memory Hierarchy



Complete Memory Hierarchy



Complete Memory Hierarchy



Memory Ordering Buffer

Handles in-flight memory loads and stores that execute:

- Out-of-order
- Speculatively

Enforce memory ordering rules:

- Retire loads and stores with correct values
- For example:
  - Loads can be reordered with older stores to different locations

Implements methods for dynamically extracting ILP

- Memory disambiguation prediction
- Store-to-load forwarding

4K-Aliasing?

Intel assumes dependency between 4 KB separated memory loads and stores

4K-Aliasing?

Intel assumes dependency between 4 KB separated memory loads and stores

Avoids potential write-after-read hazard

4K-Aliasing?

Intel assumes dependency between 4 KB separated memory loads and stores

Avoids potential write-after-read hazard

- When a later write passes an earlier read

```
mov rax, [rbx]    \\read
mov [rbx], rcx    \\write
```

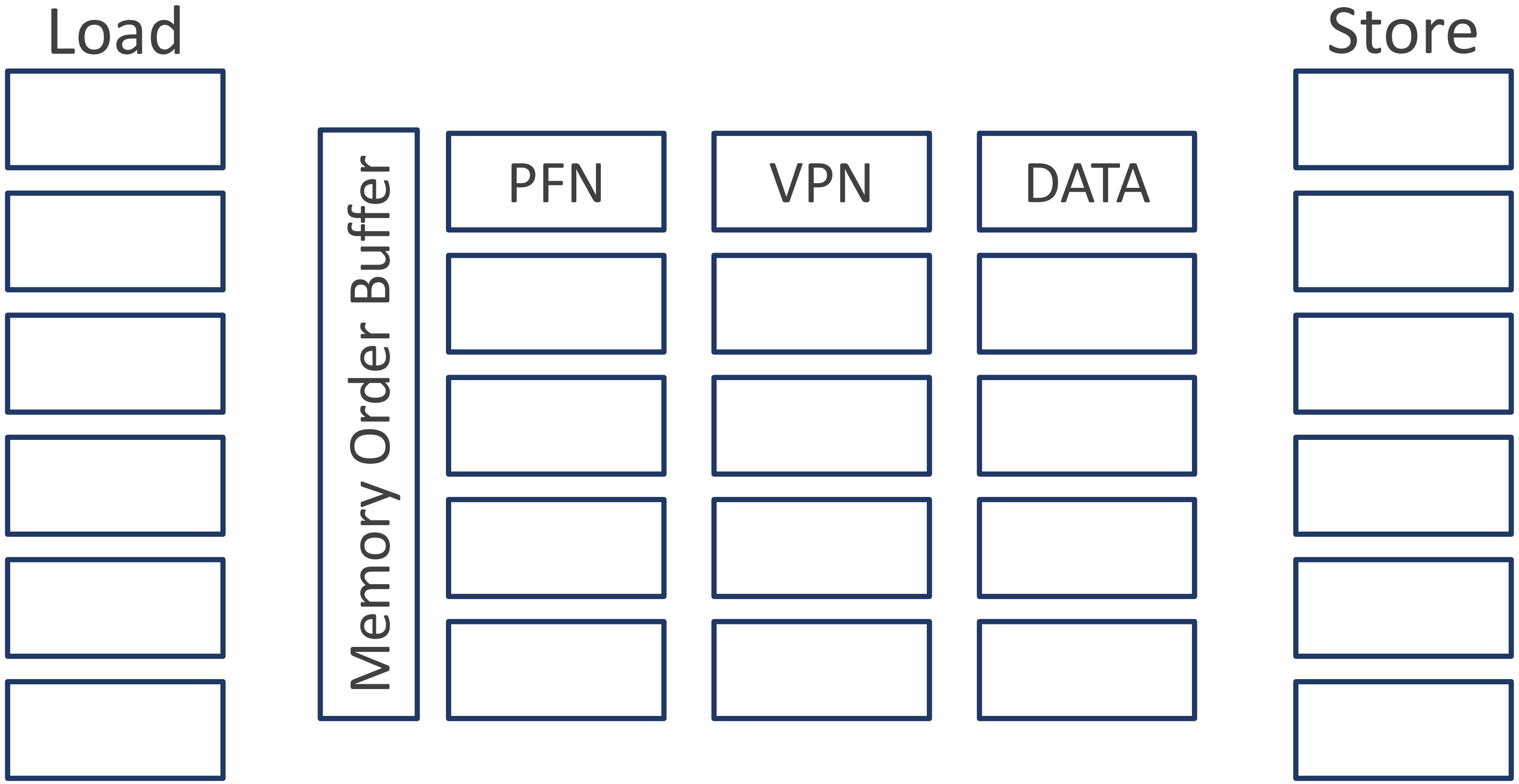
- The earlier read **must NOT** load the result written by the later store

Loads and stores separated by 4 KB will falsely alias

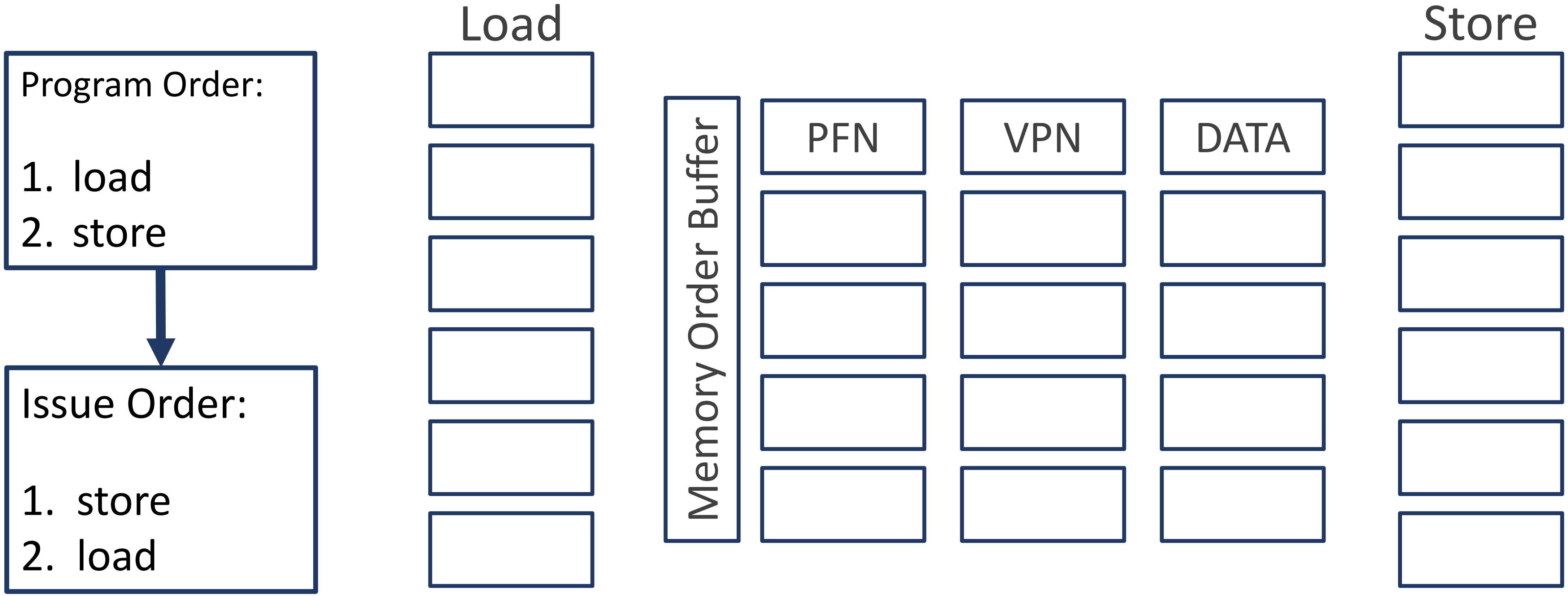
4K-Aliasing

Program Order:

1. load
2. store



4K-Aliasing



Program Order:

1. load
2. store

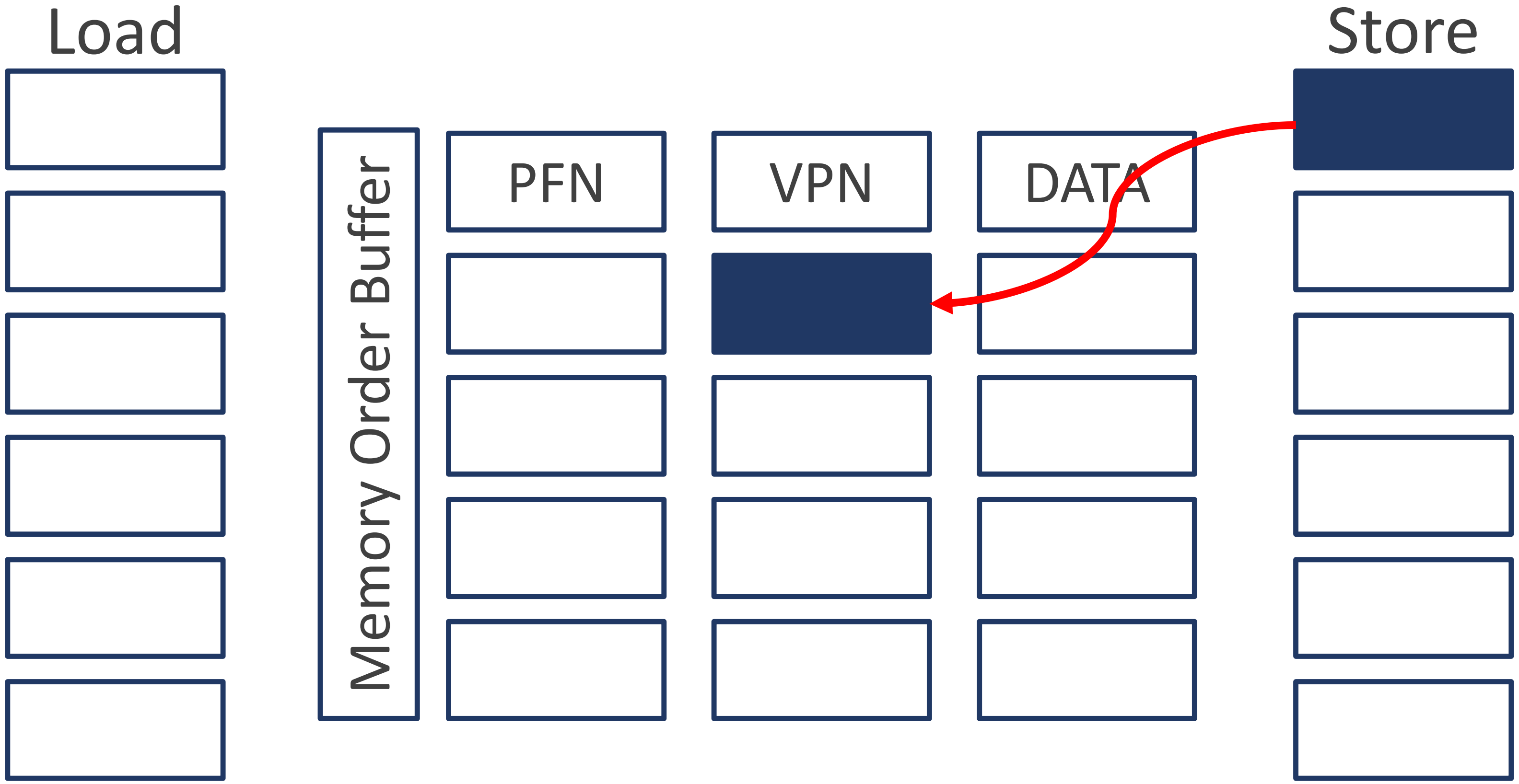
Issue Order:

1. store
2. load

4K-Aliasing

Issue Order:

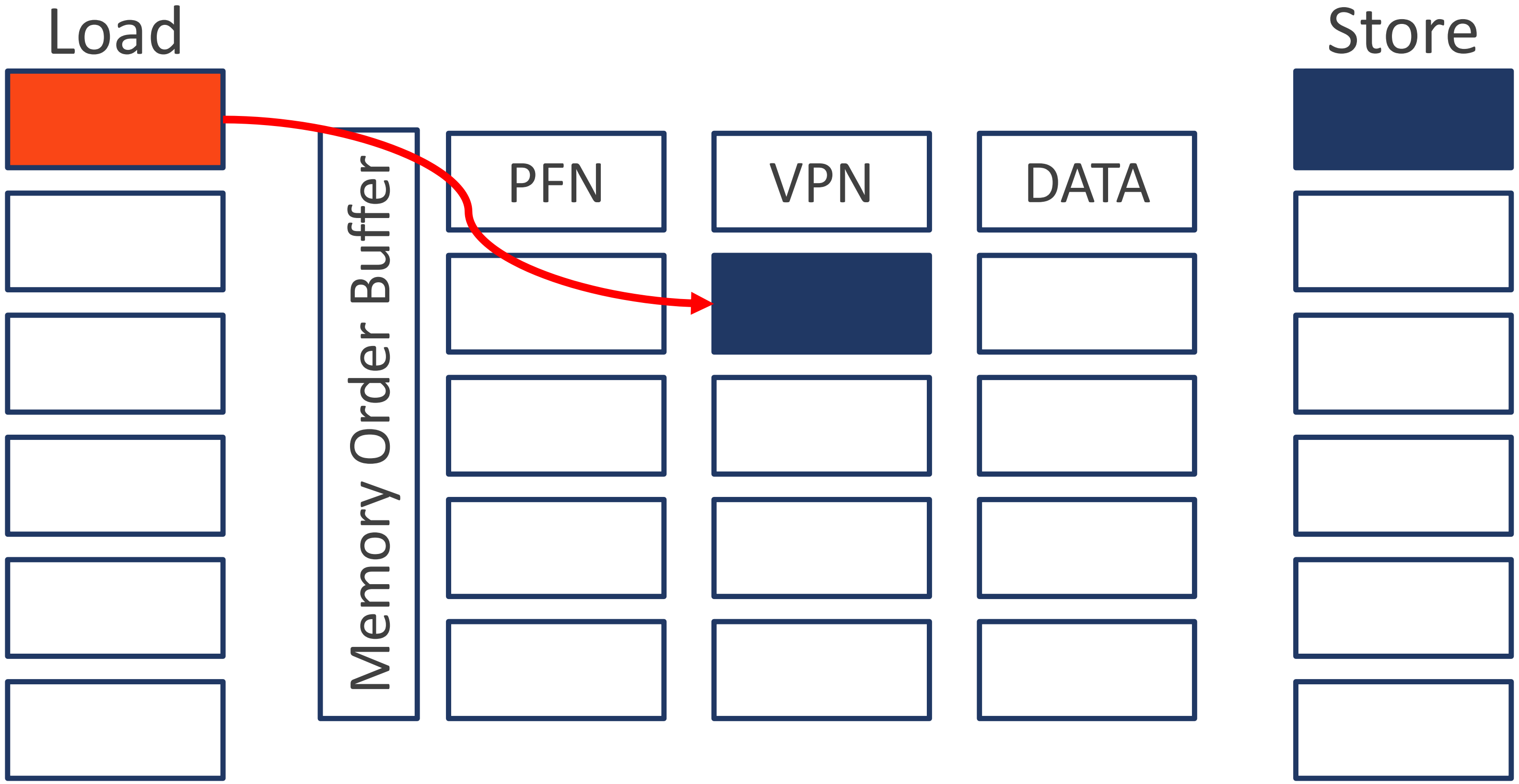
1. store
2. load



4K-Aliasing

Issue Order:

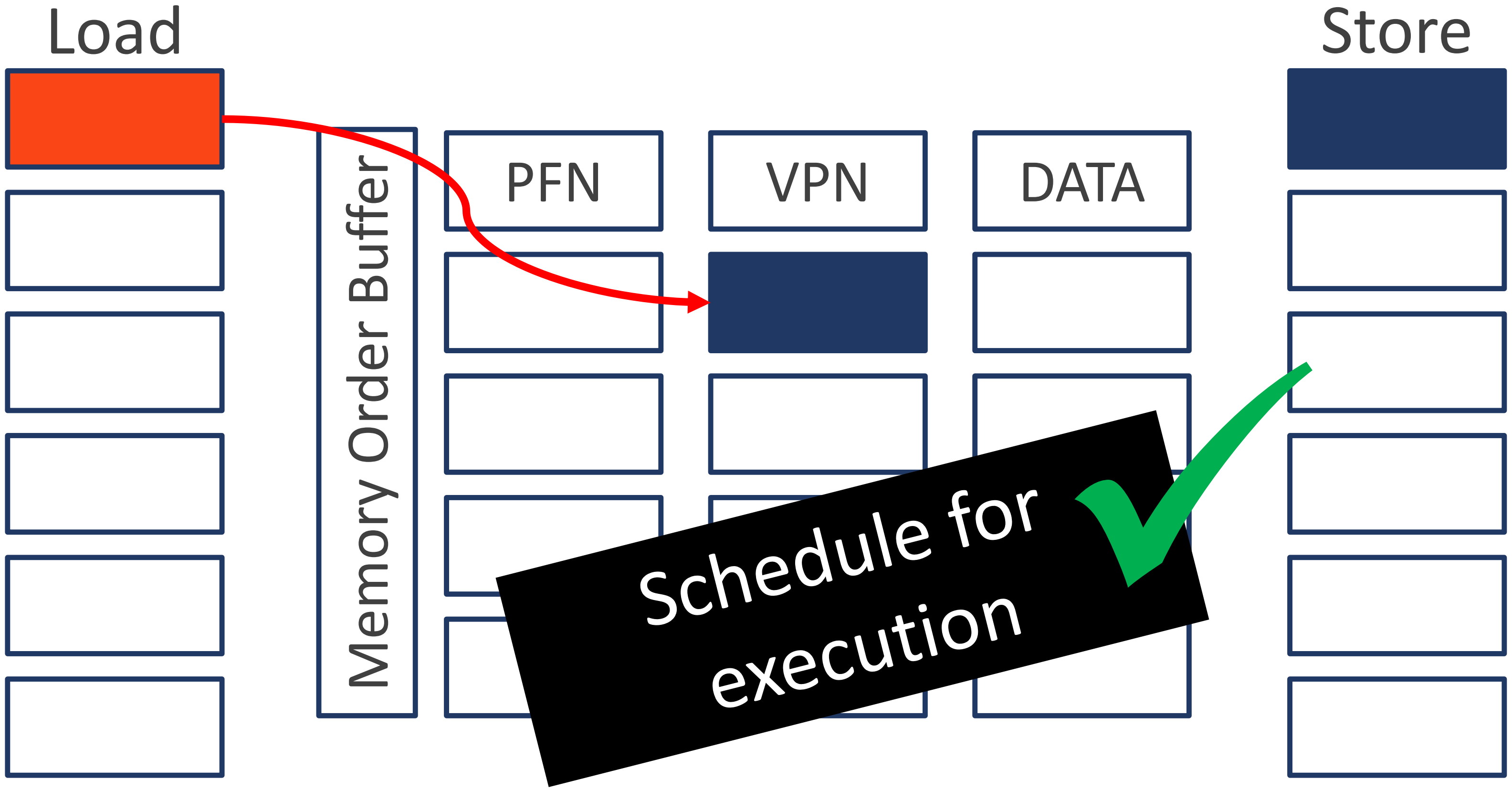
1. store
2. load



4K-Aliasing

Issue Order:

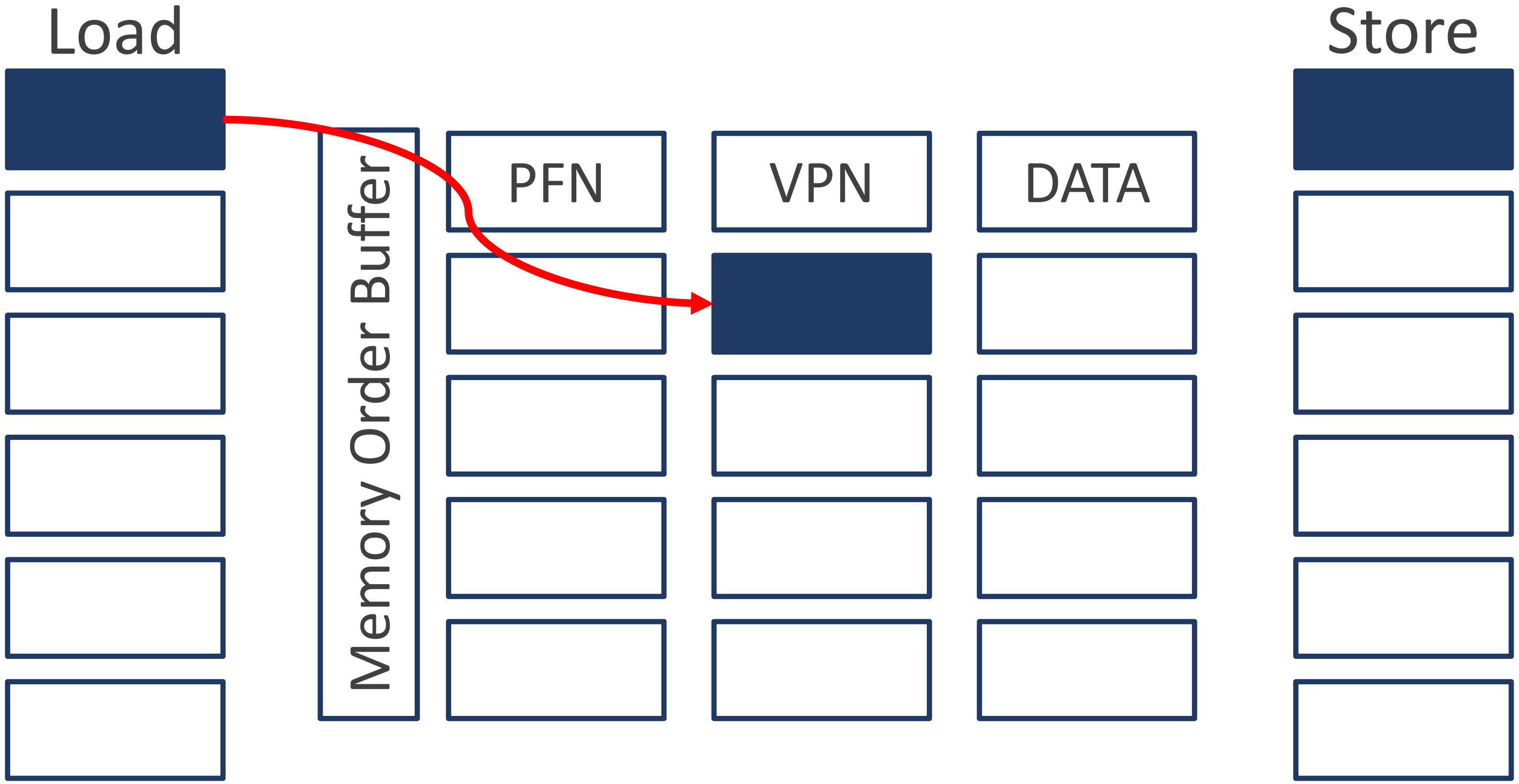
1. store
2. load



4K-Aliasing

Issue Order:

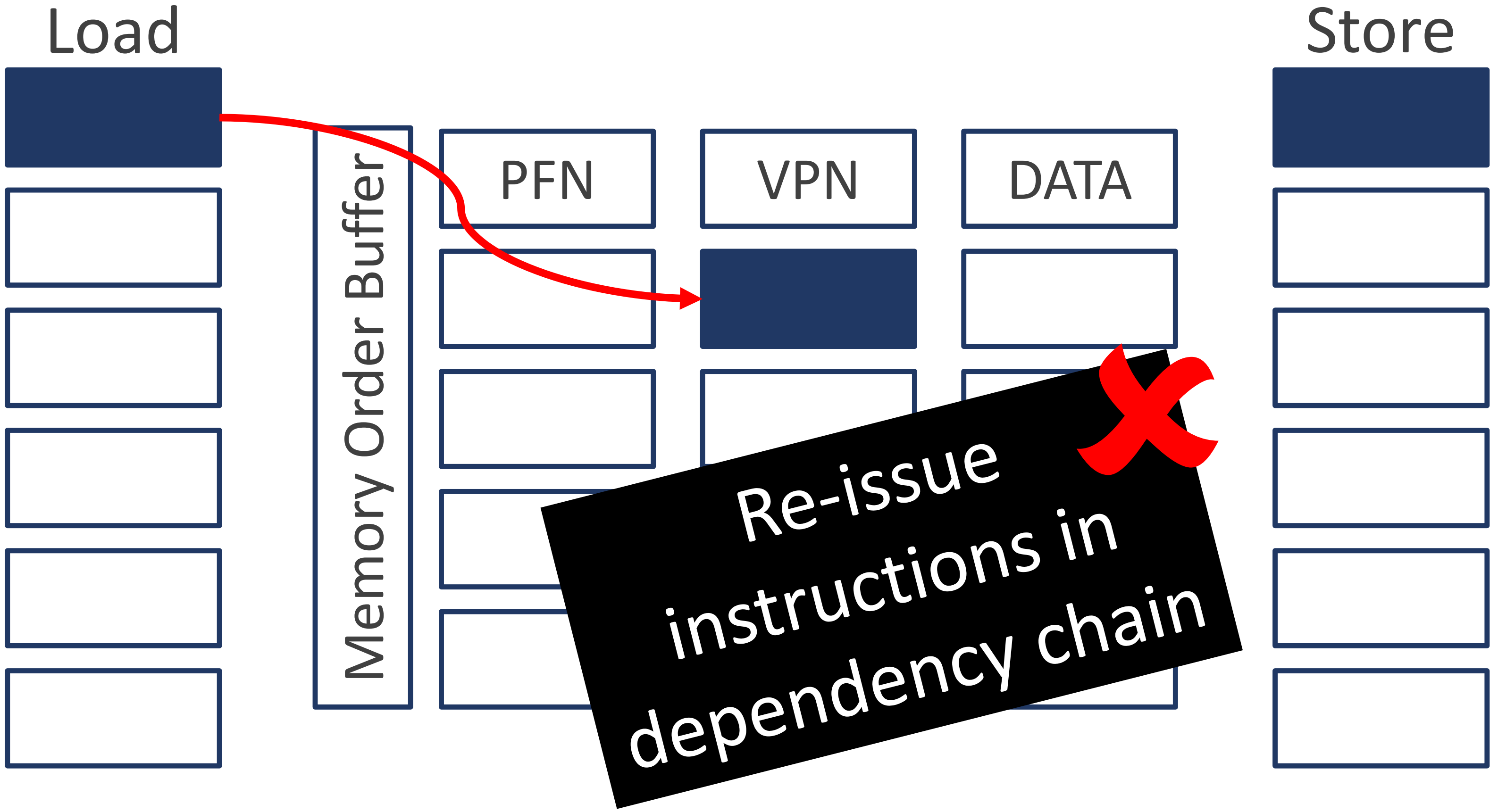
1. store
2. load



4K-Aliasing

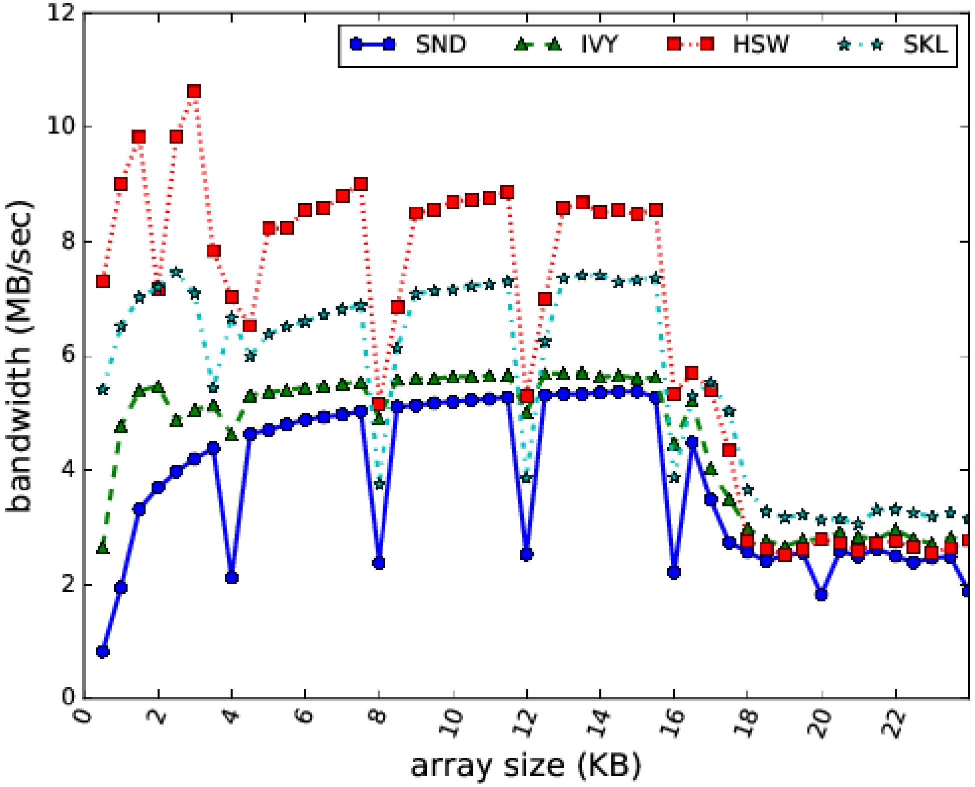
Issue Order:

1. store
2. load



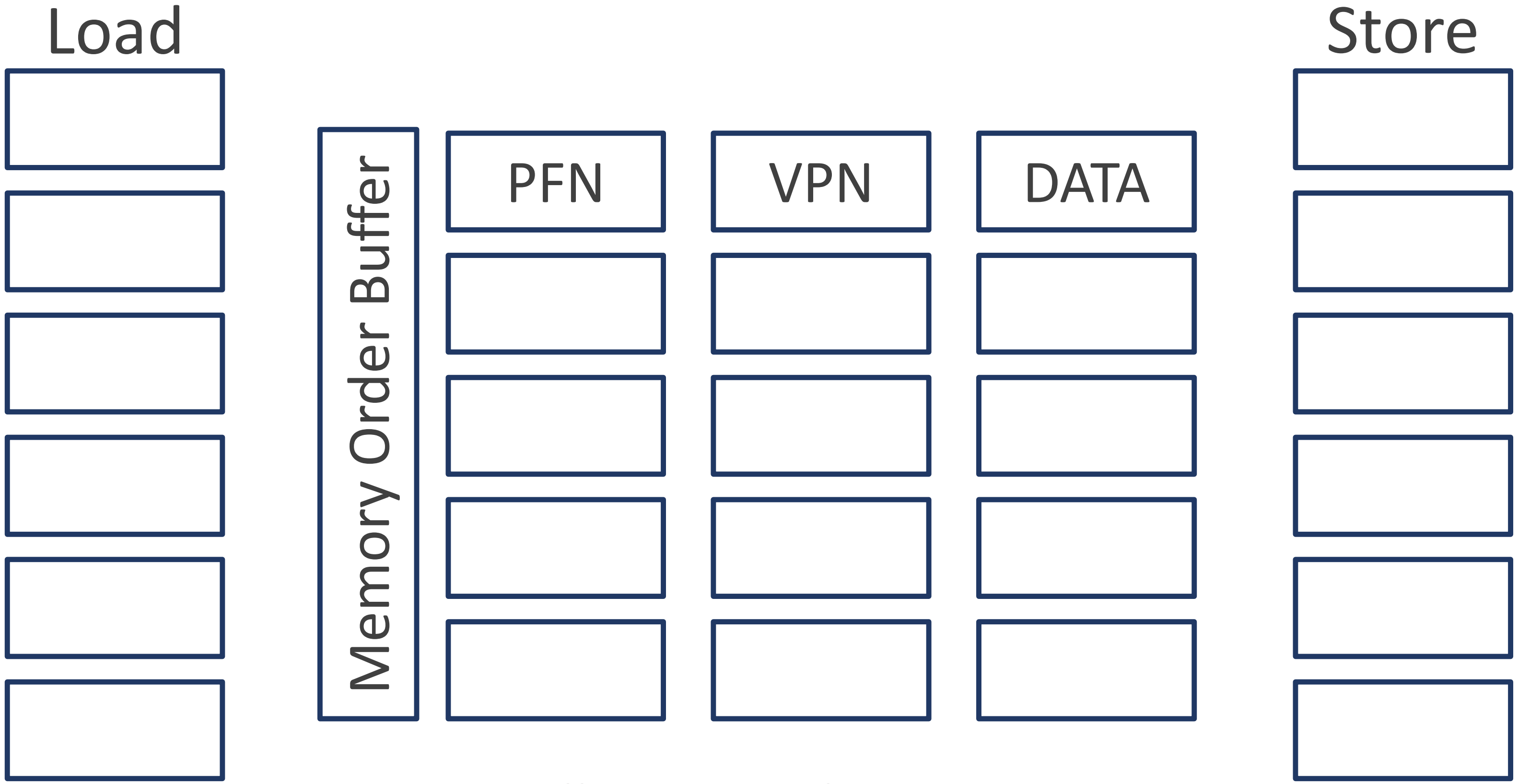
Re-issue instructions in dependency chain

False 4K-Aliasing



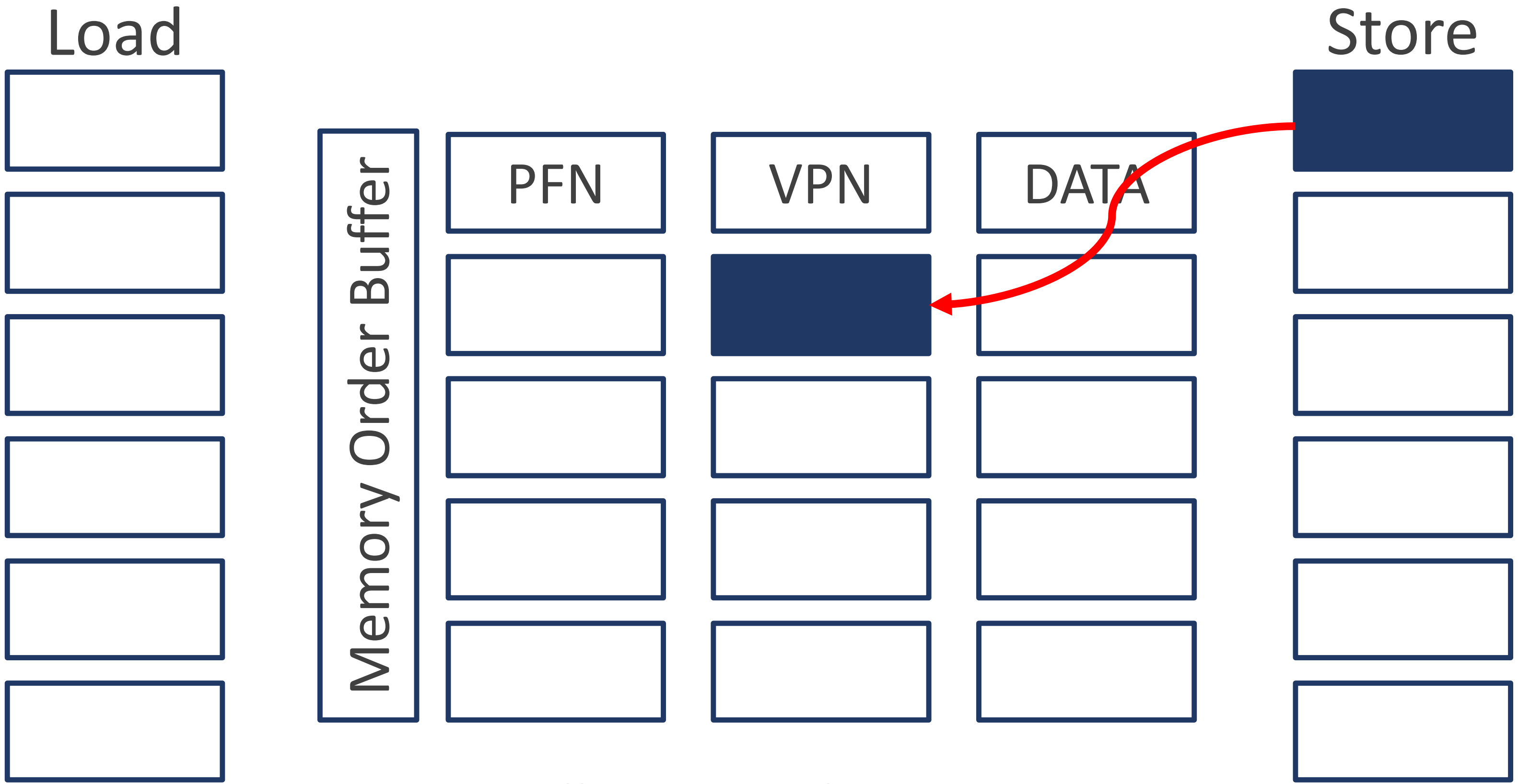
Performance of memory copy routine falls off when source and destination buffer are separated by *n* * 4 KB

4K-Aliasing Timing Channel



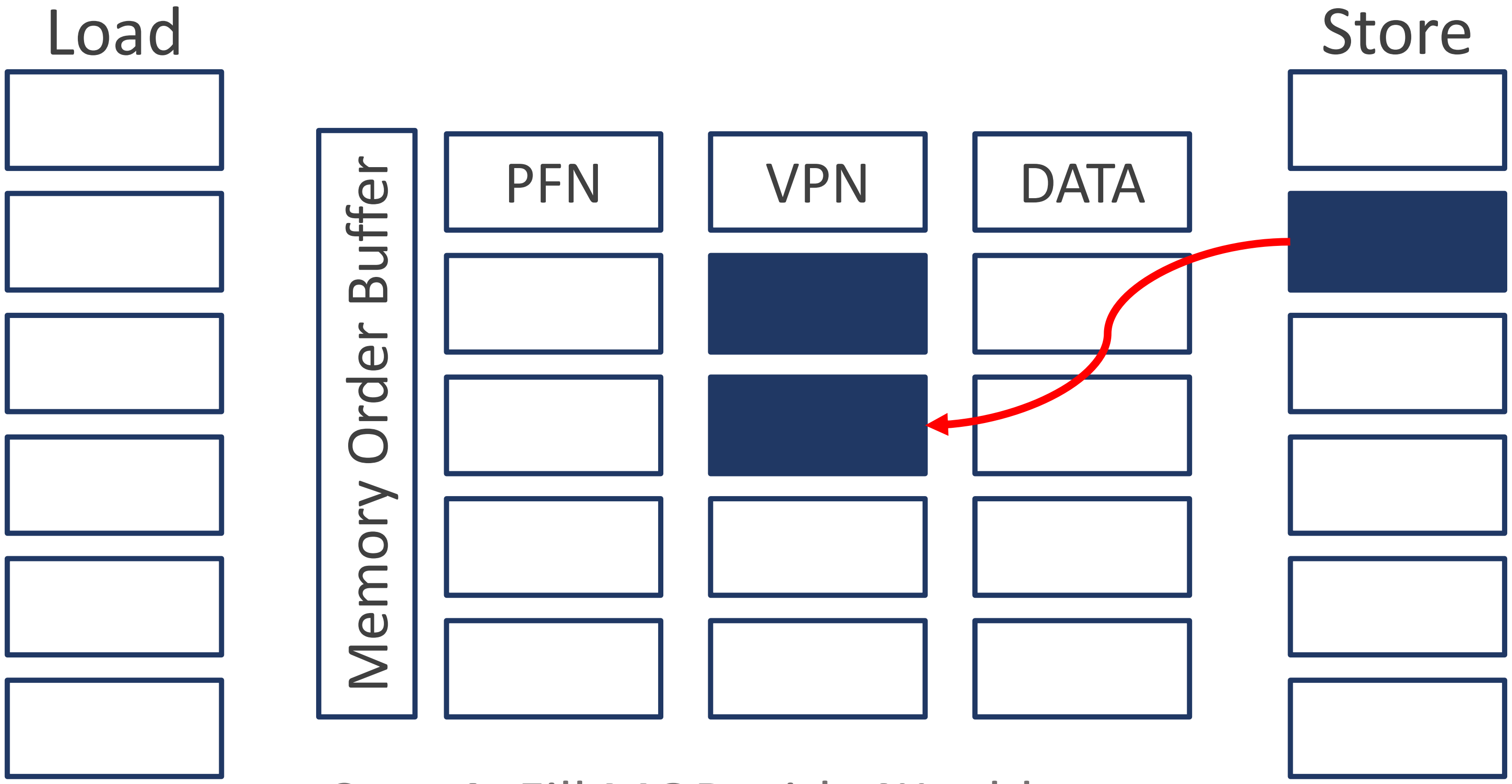
Step 1: Fill MOB with 4K addresses

4K-Aliasing Timing Channel



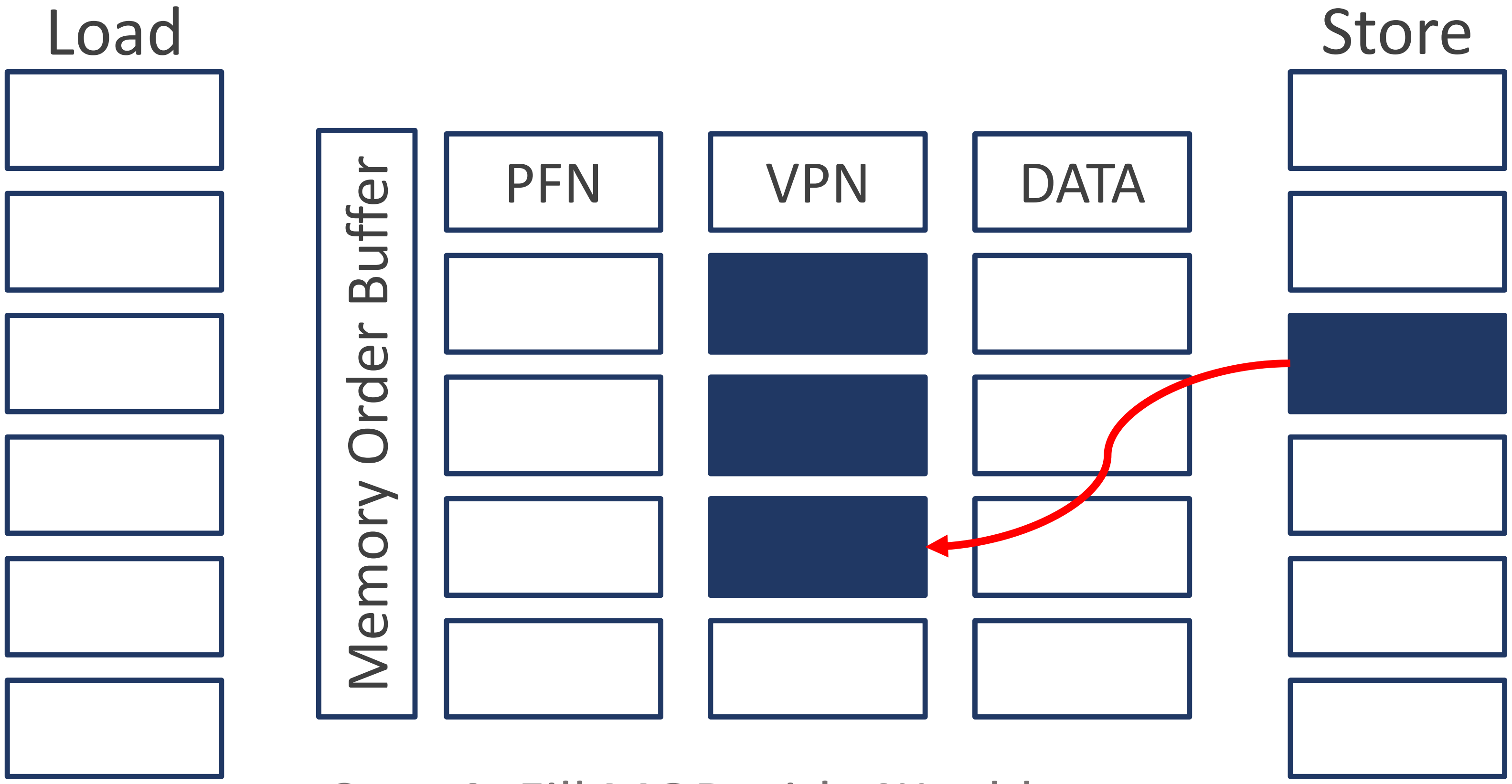
Step 1: Fill MOB with 4K addresses

4K-Aliasing Timing Channel



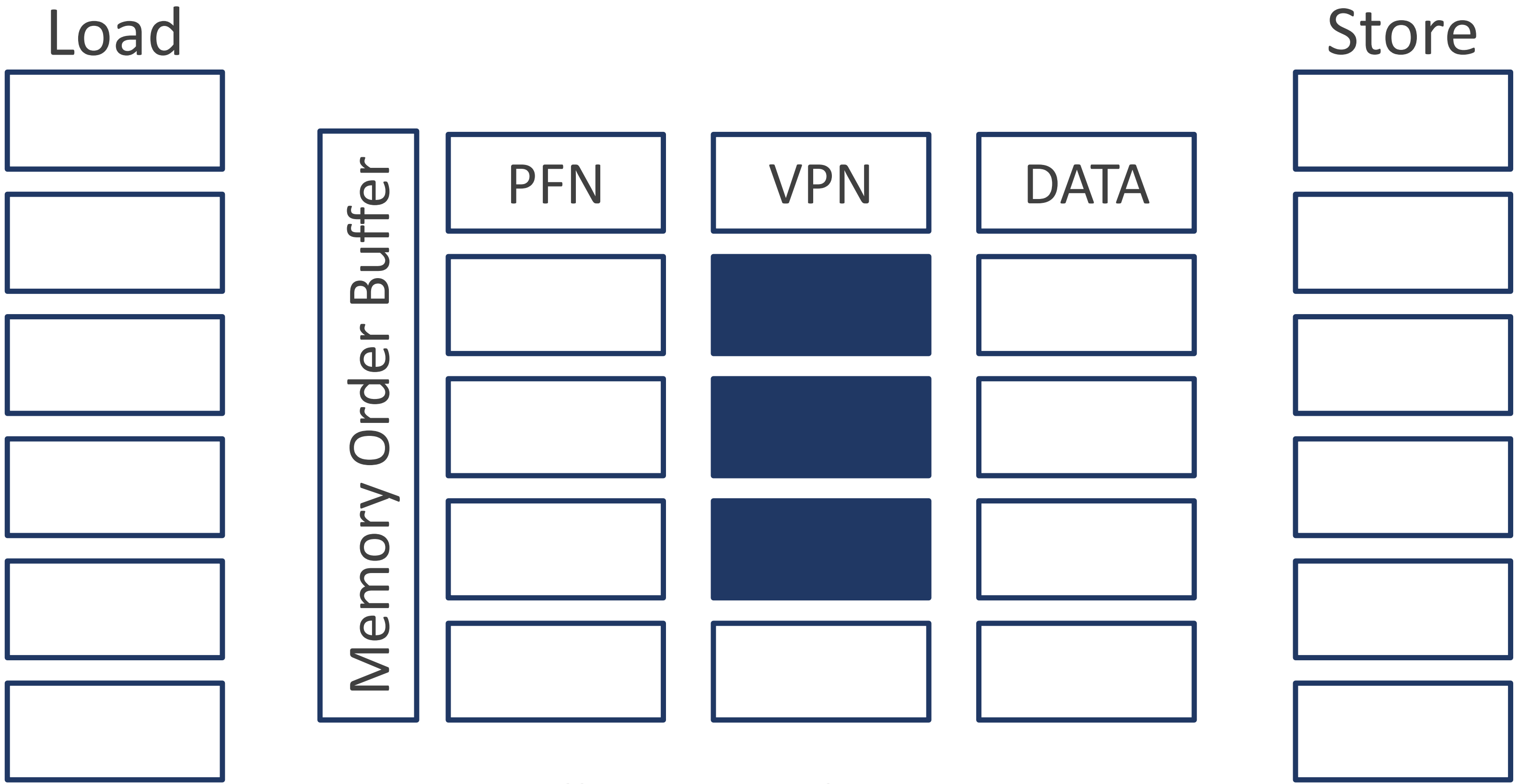
Step 1: Fill MOB with 4K addresses

4K-Aliasing Timing Channel



Step 1: Fill MOB with 4K addresses

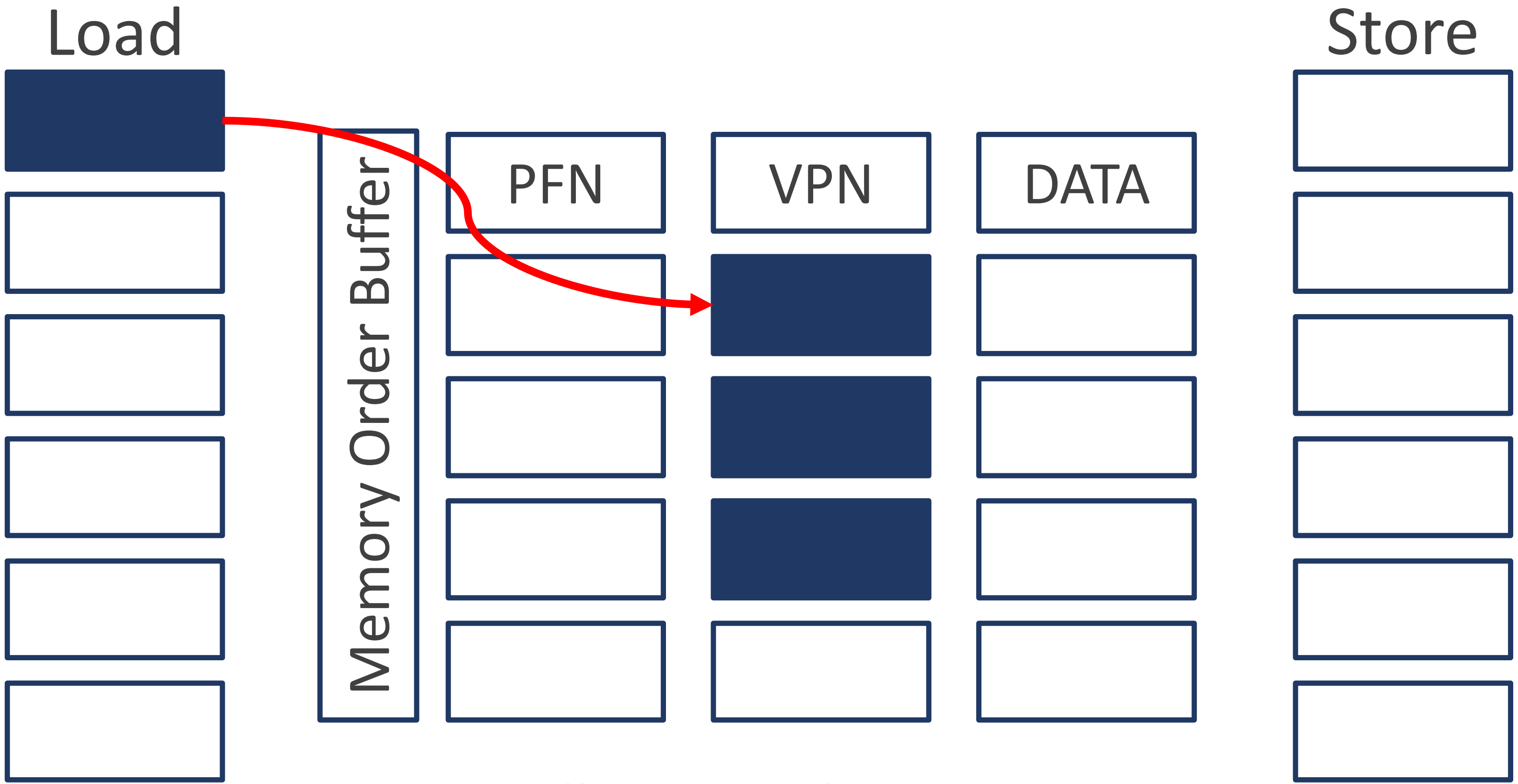
4K-Aliasing Timing Channel



Step 1: Fill MOB with 4K addresses

Step 2: Load from 4K-aligned address

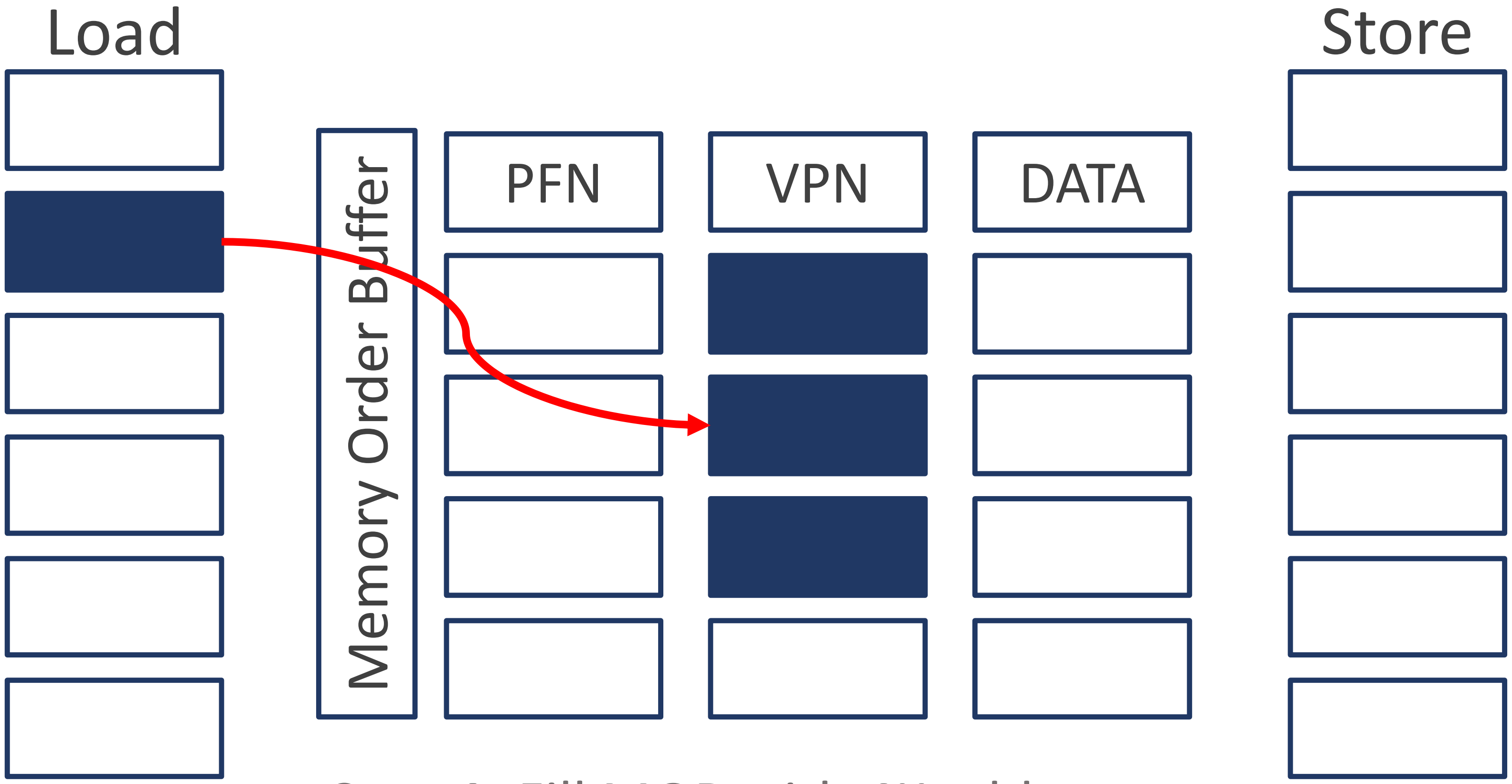
4K-Aliasing Timing Channel



Step 1: Fill MOB with 4K addresses

Step 2: Load from 4K-aligned address

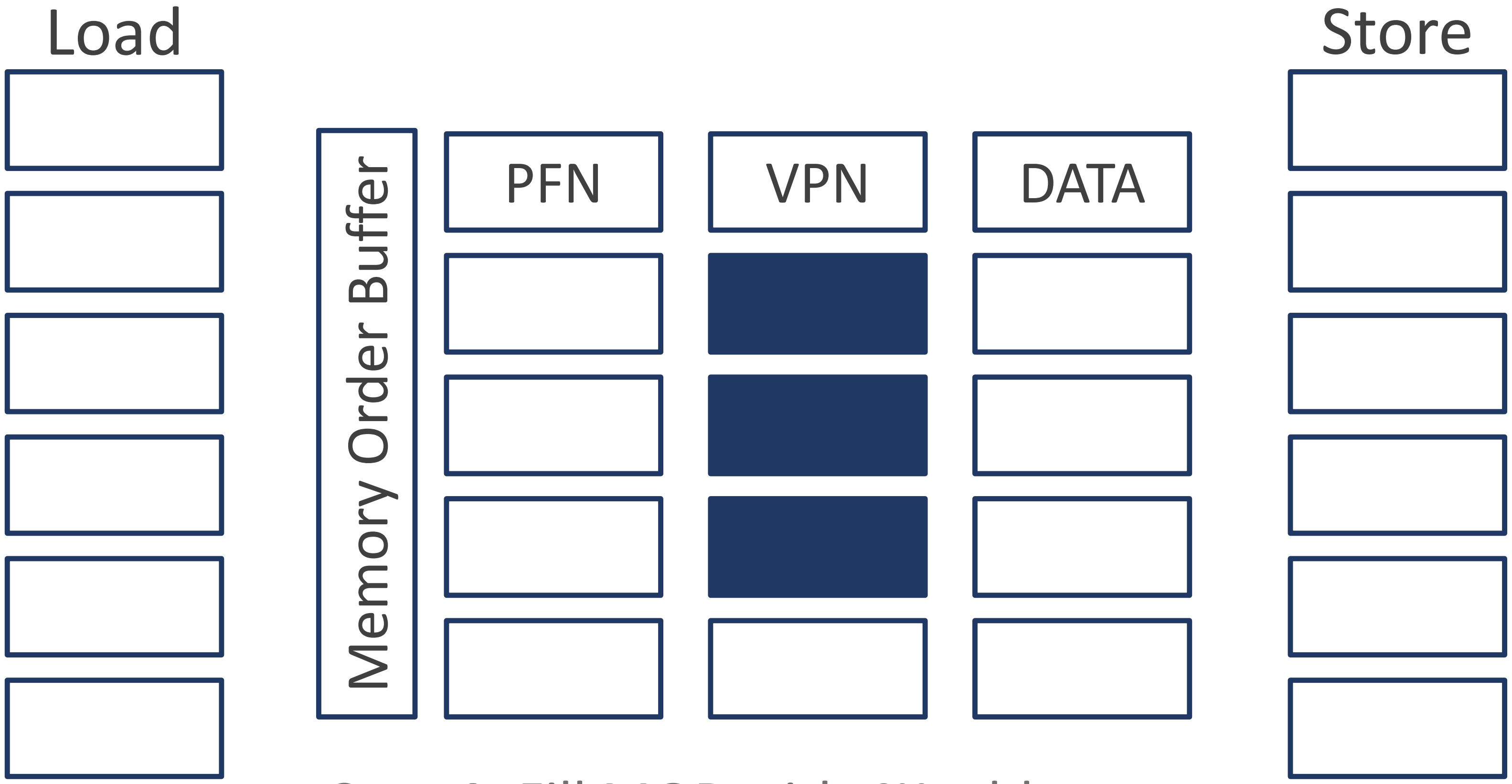
4K-Aliasing Timing Channel



Step 1: Fill MOB with 4K addresses

Step 2: Load from 4K-aligned address

4K-Aliasing Timing Channel



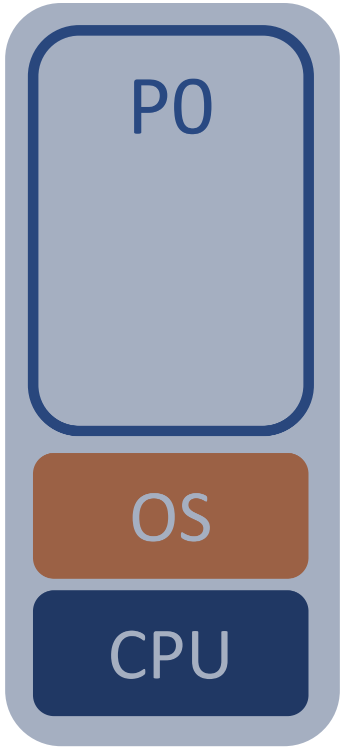
Step 1: Fill MOB with 4K addresses

Step 2: Load from 4K-aligned address

Step 3: Latency of 4K-aliasing load slow

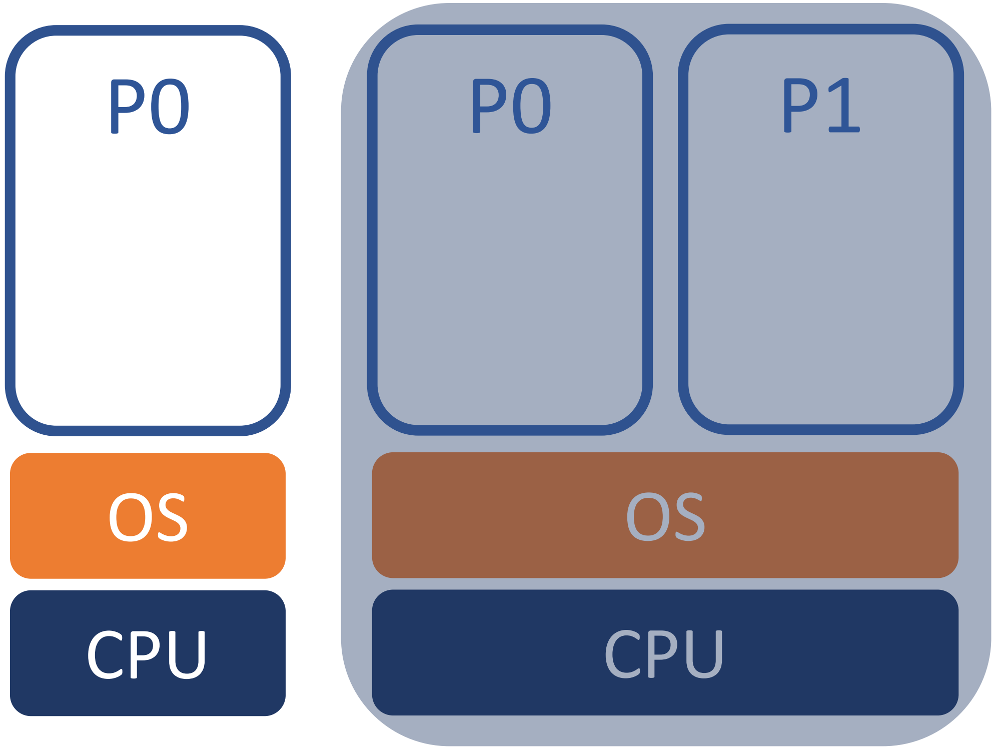
4K-Aliasing Analysis Setting

Single Process



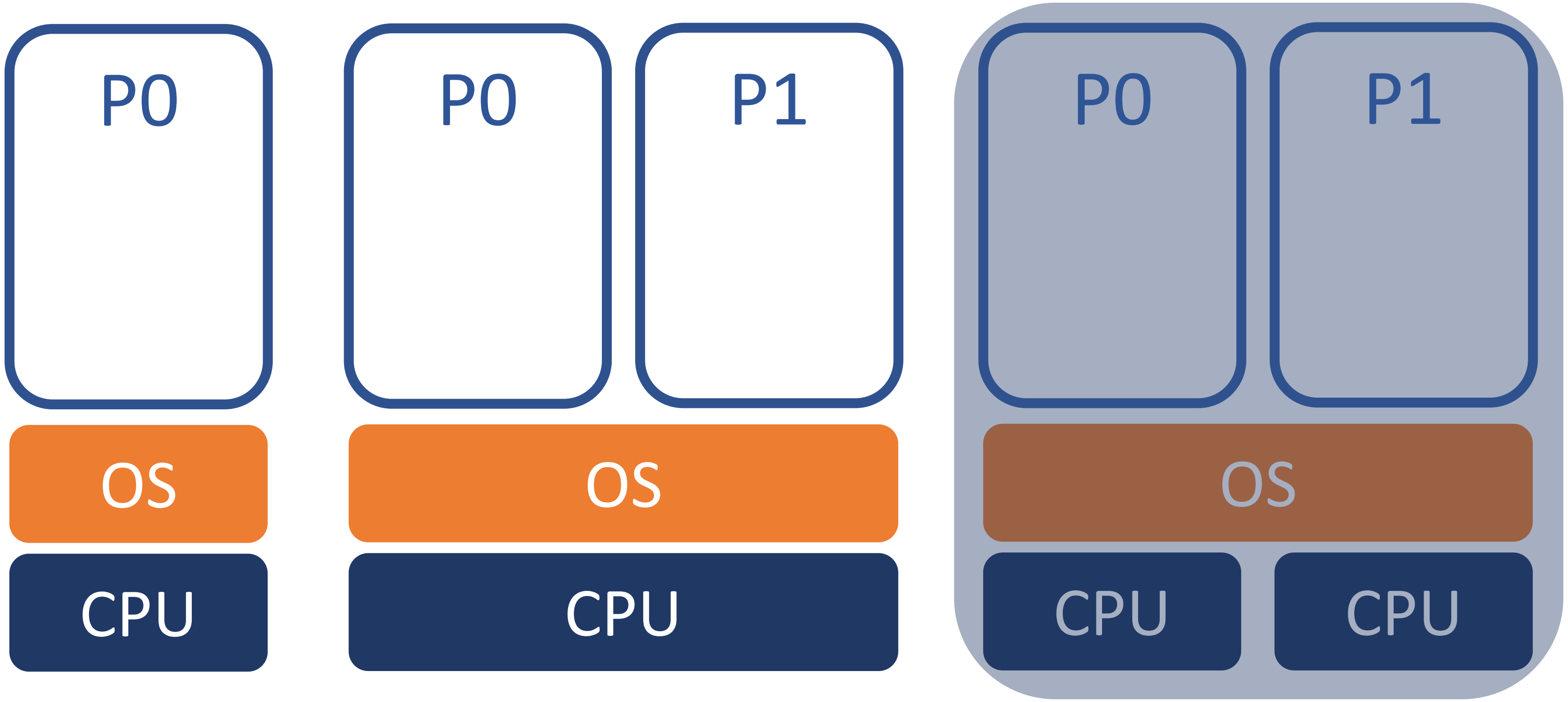
4K-Aliasing Analysis Setting

Across Processes



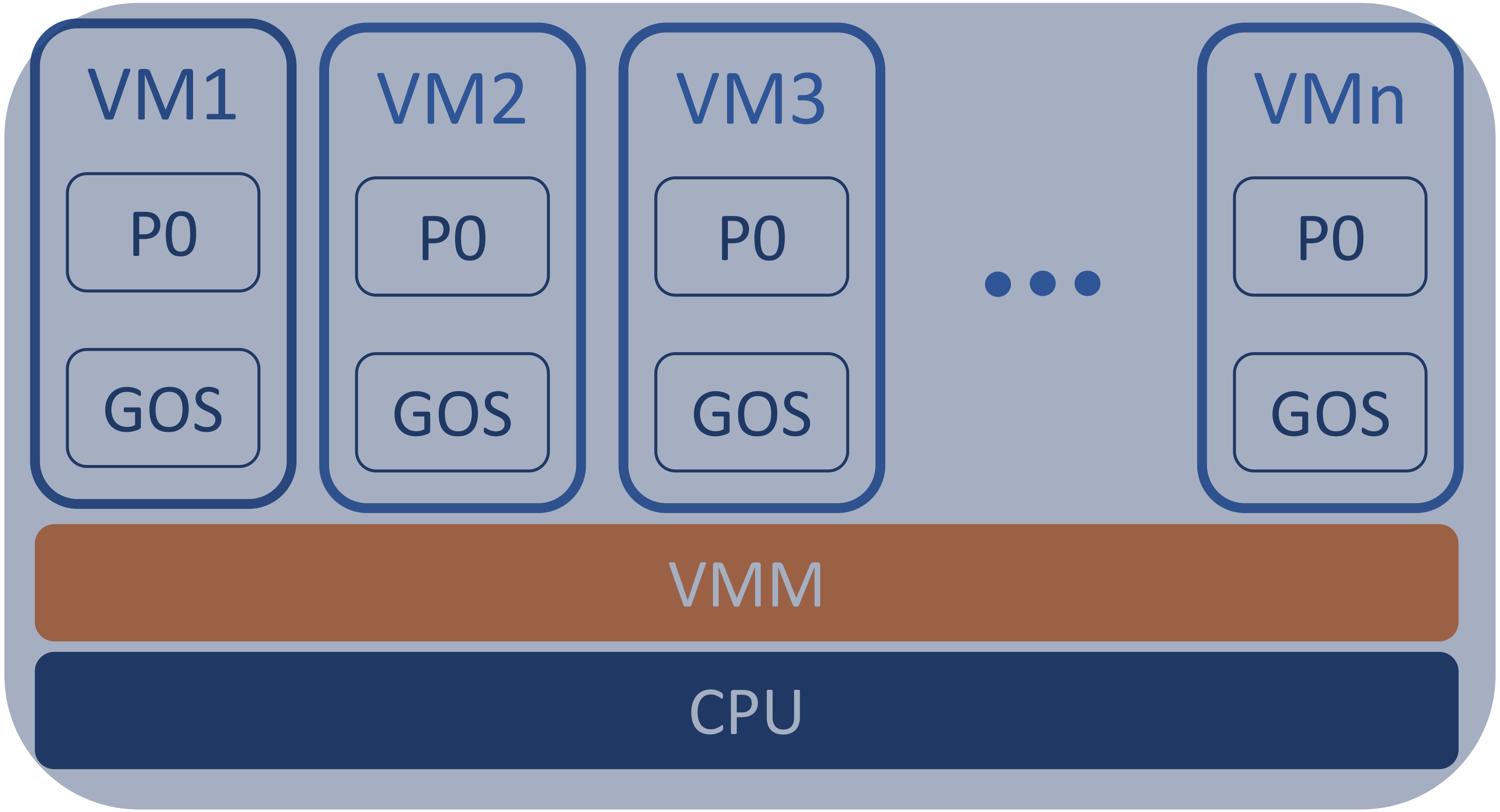
4K-Aliasing Analysis Setting

Across Cores

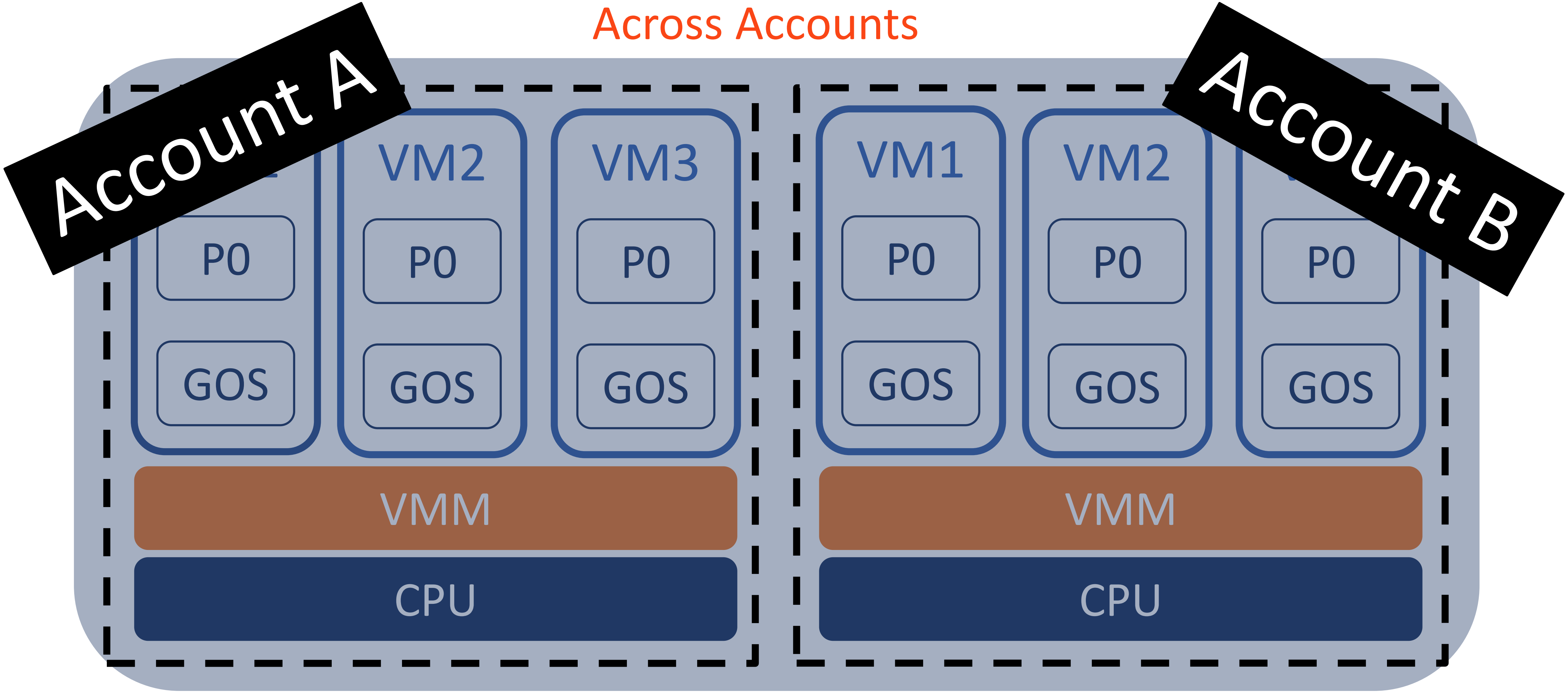


4K-Aliasing Analysis Setting

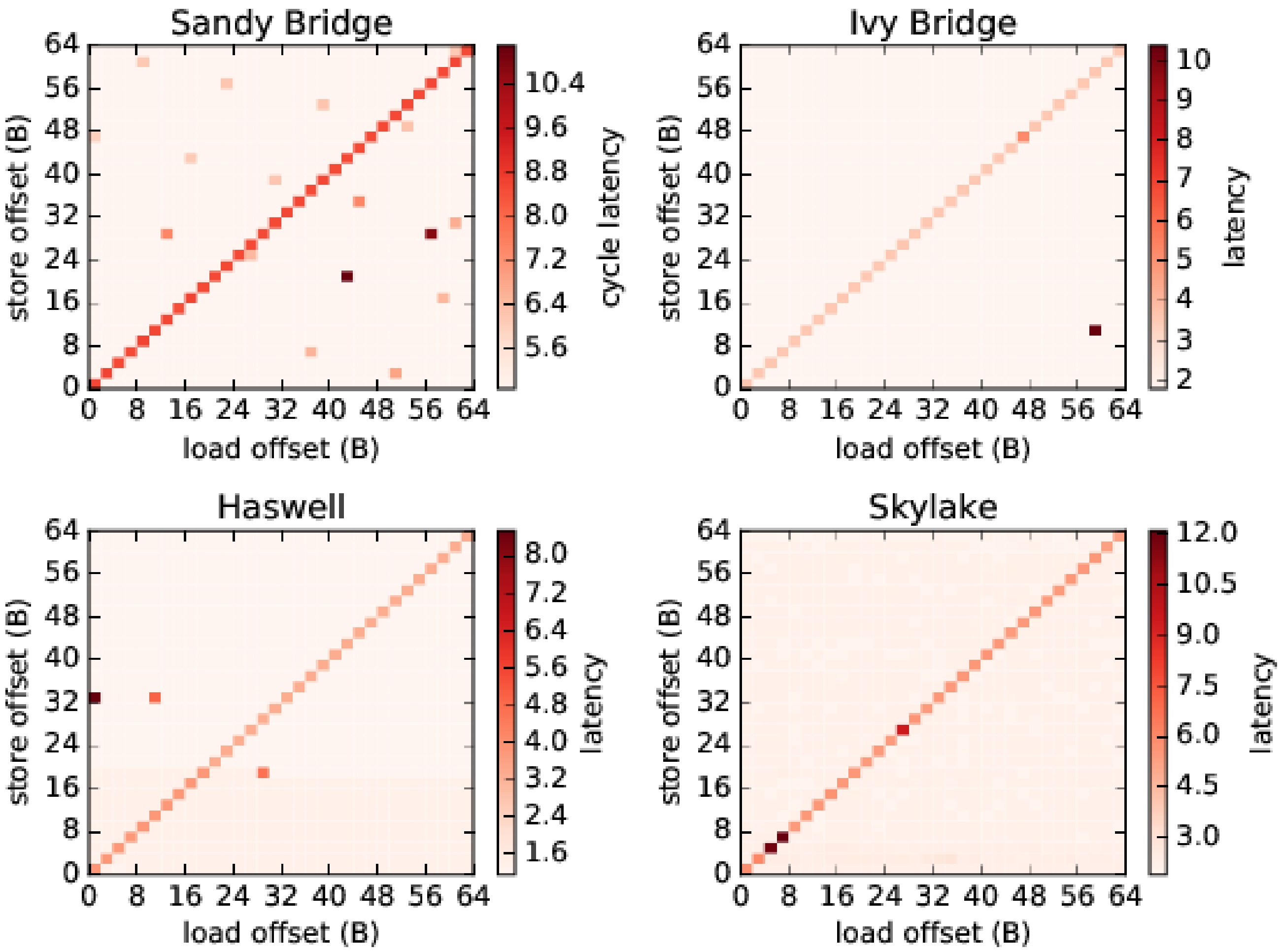
Across VMs



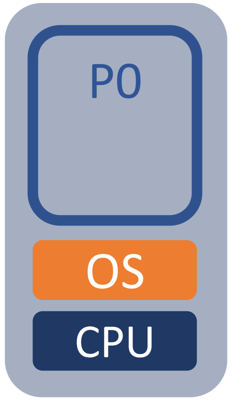
4K-Aliasing Analysis Setting



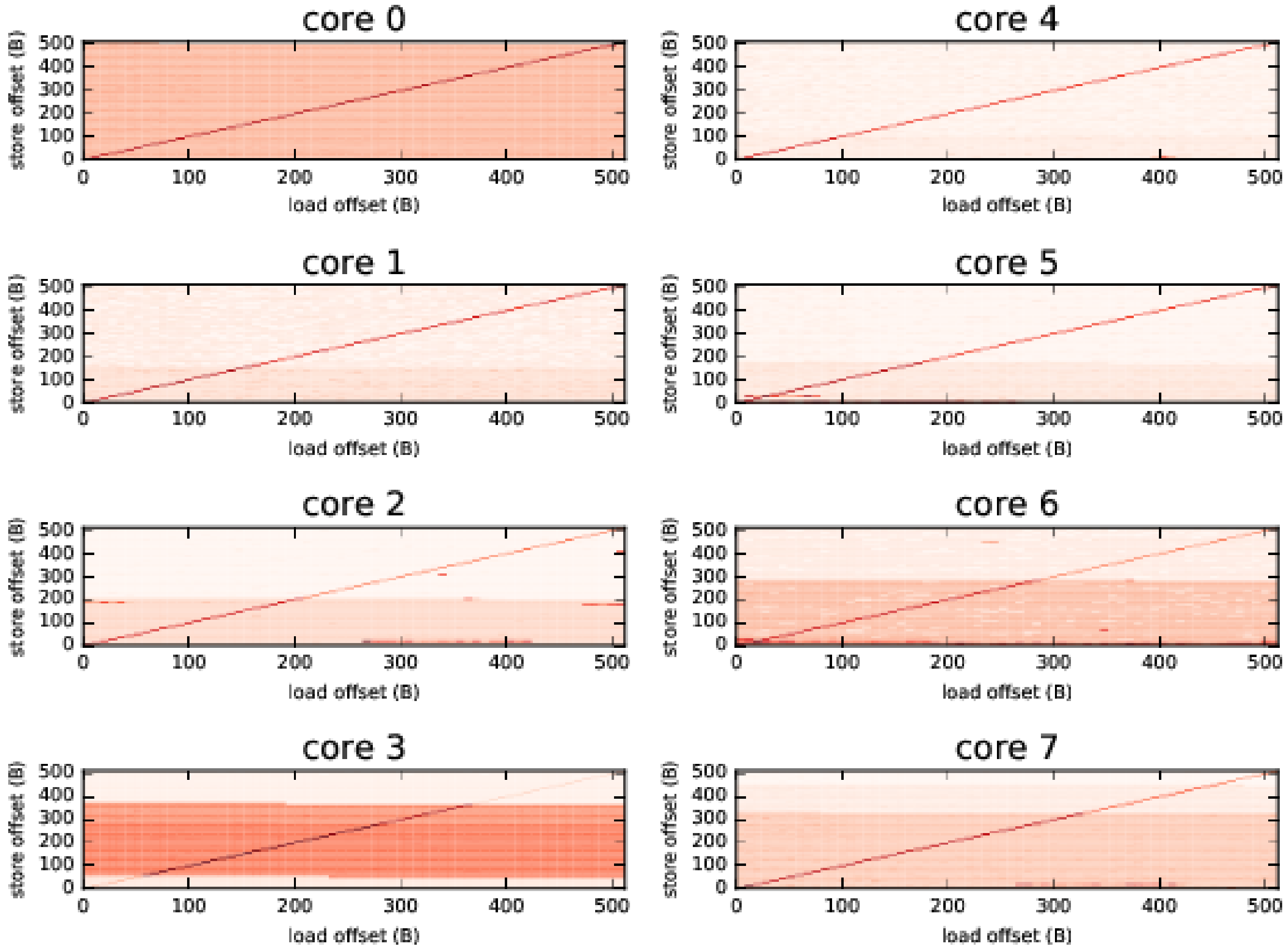
4K Latency Across Micro-Arch Families



- Average ~4.3 cyc.
- Limited noise



4K Latency Across Processes

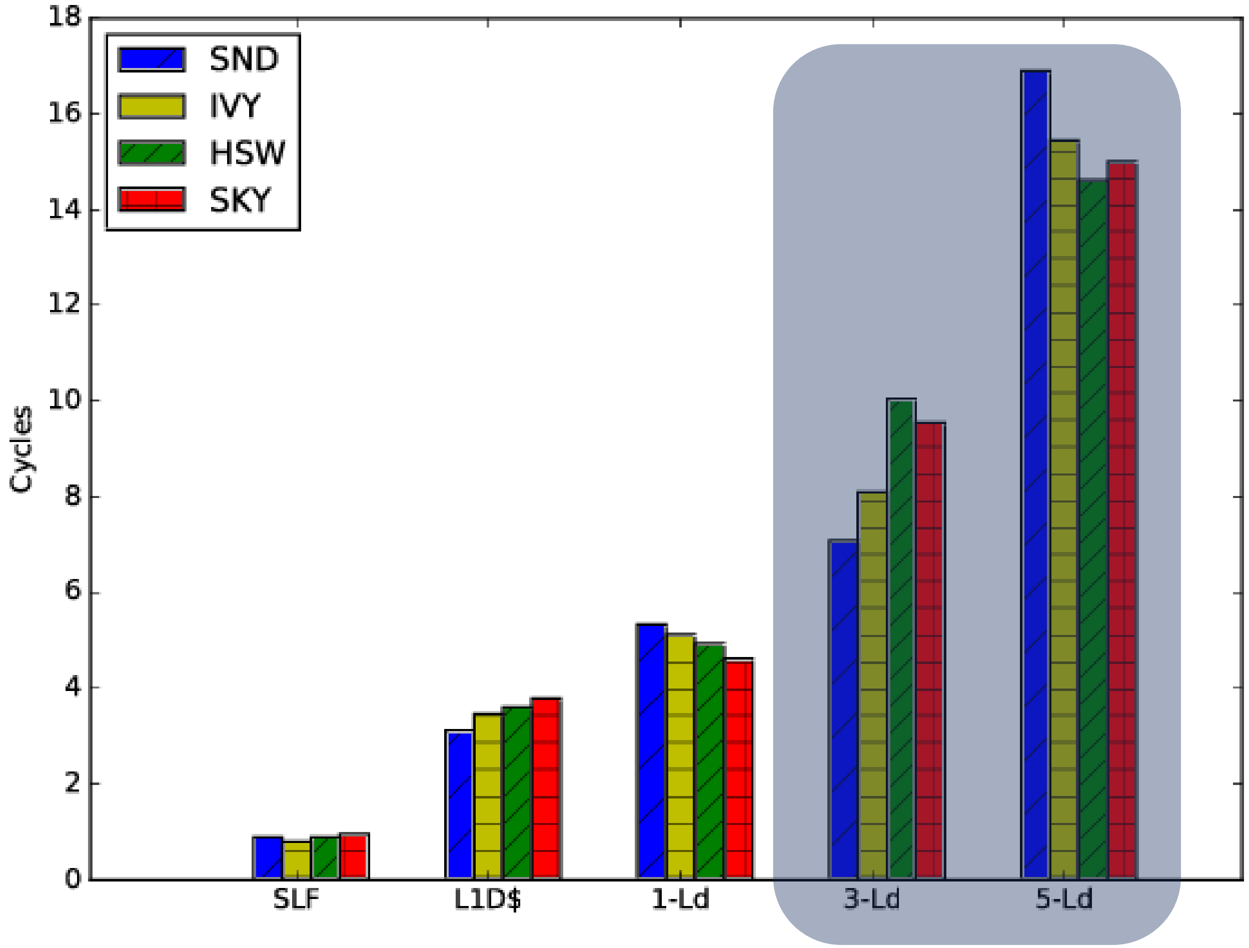


- Considerable background noise

- Similar cycle latency to single process

Can we eliminate background noise?

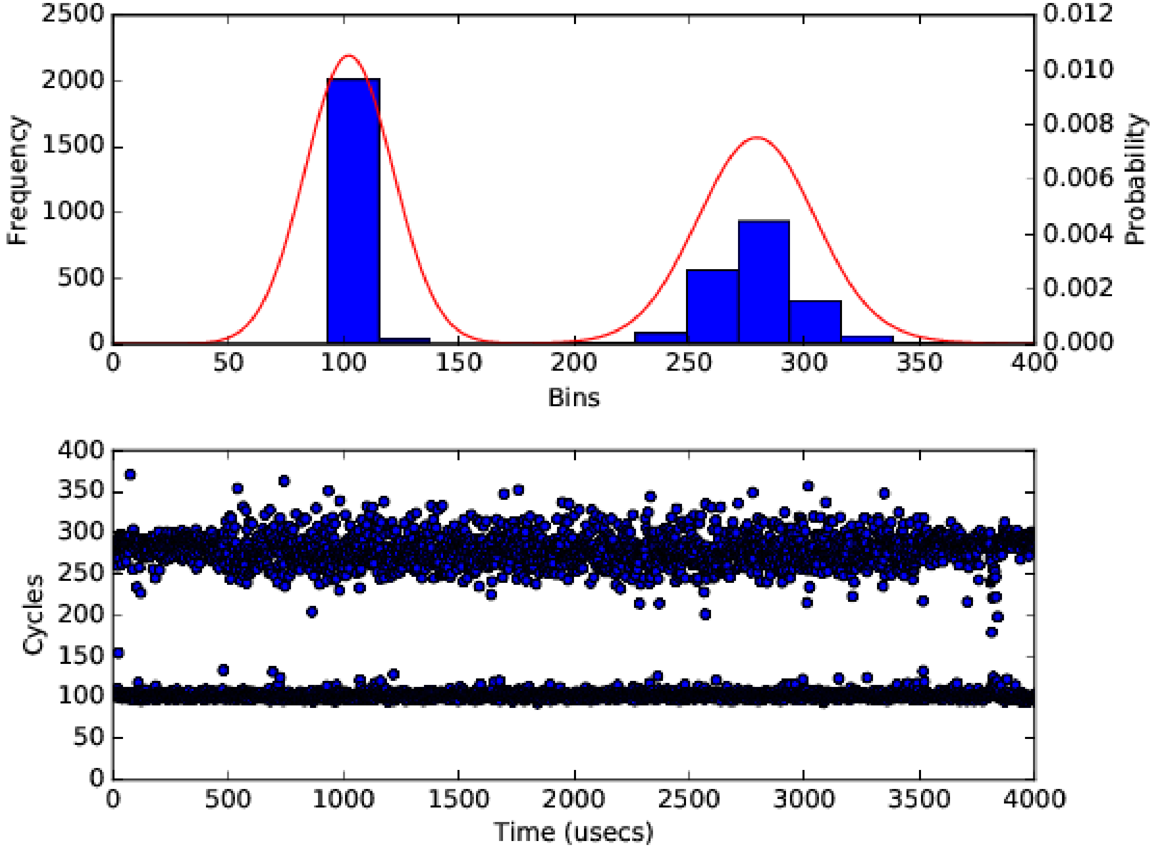
Improving 4K-Aliasing Latency



- Linear correlation between no. of aliasing loads and cycle latency

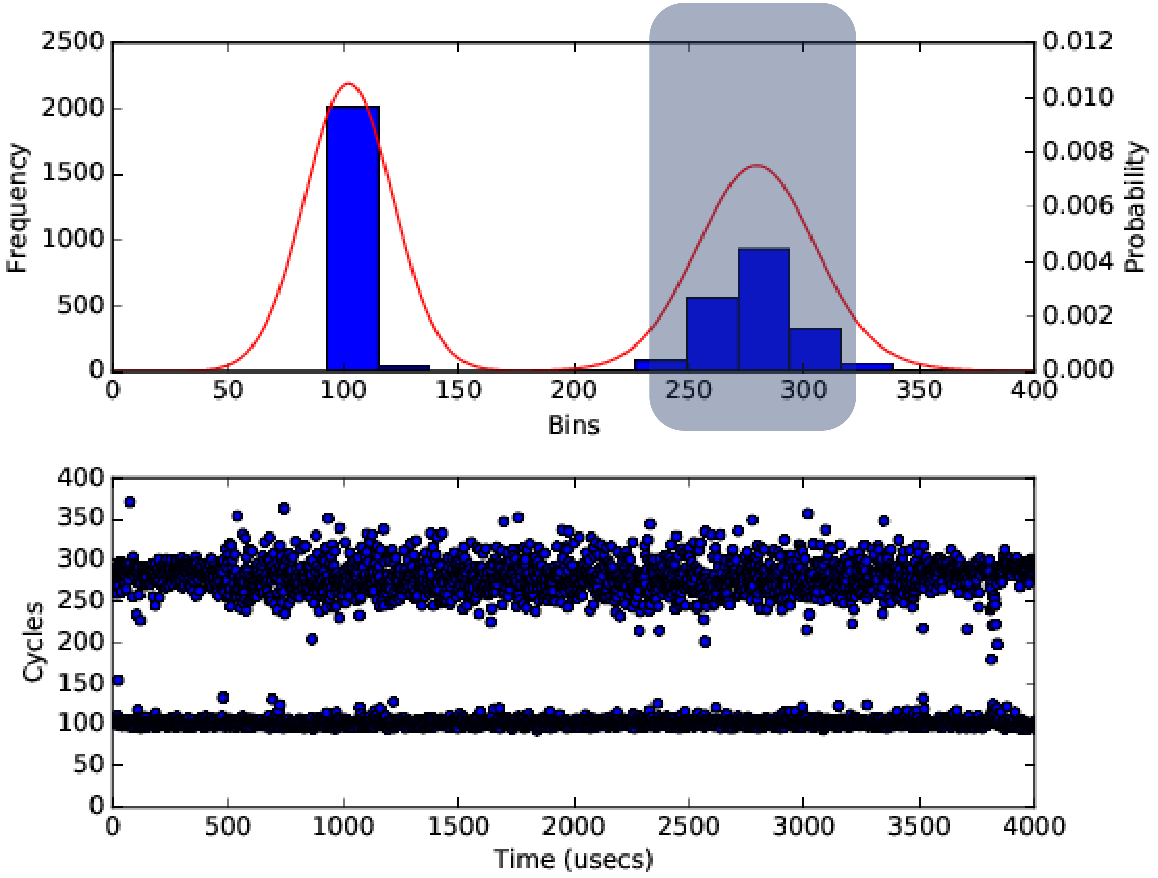
- We can improve measured latency by adding more loads within measurement window

Improving 4K-Aliasing Threshold



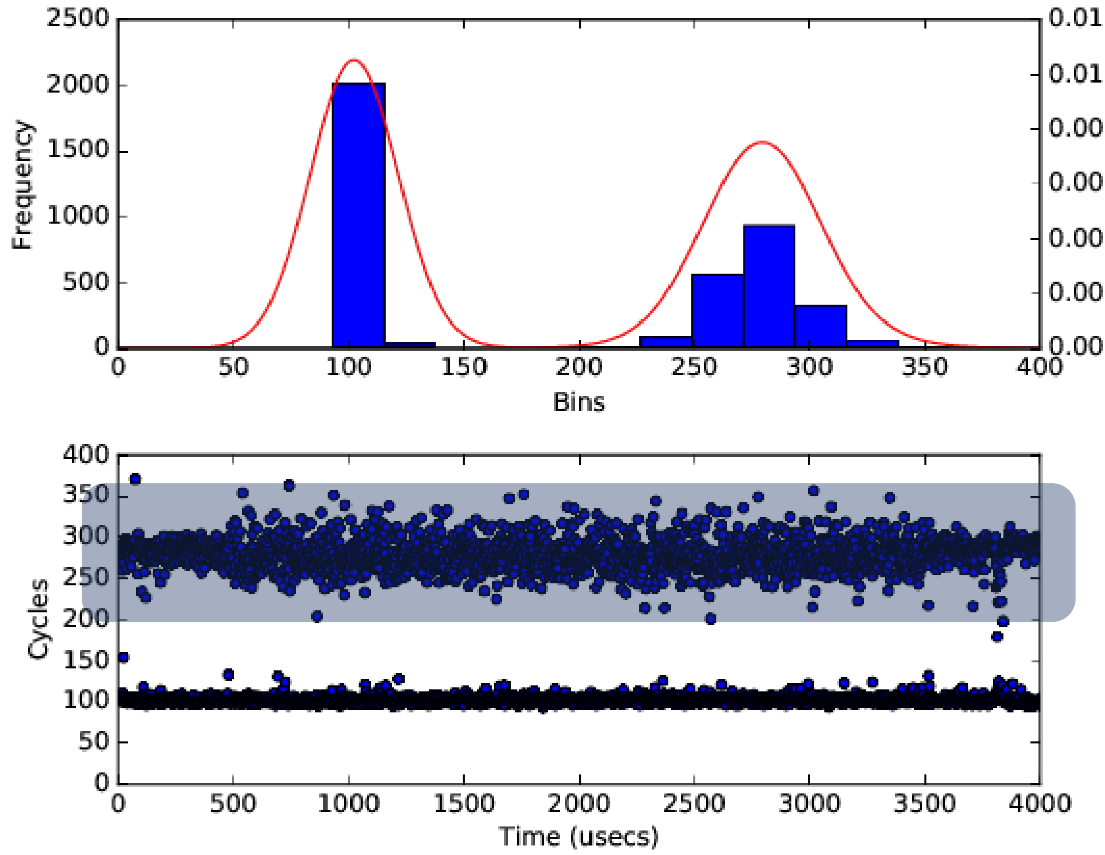
- Issue 4K aligned load half the time

Improving 4K-Aliasing Threshold



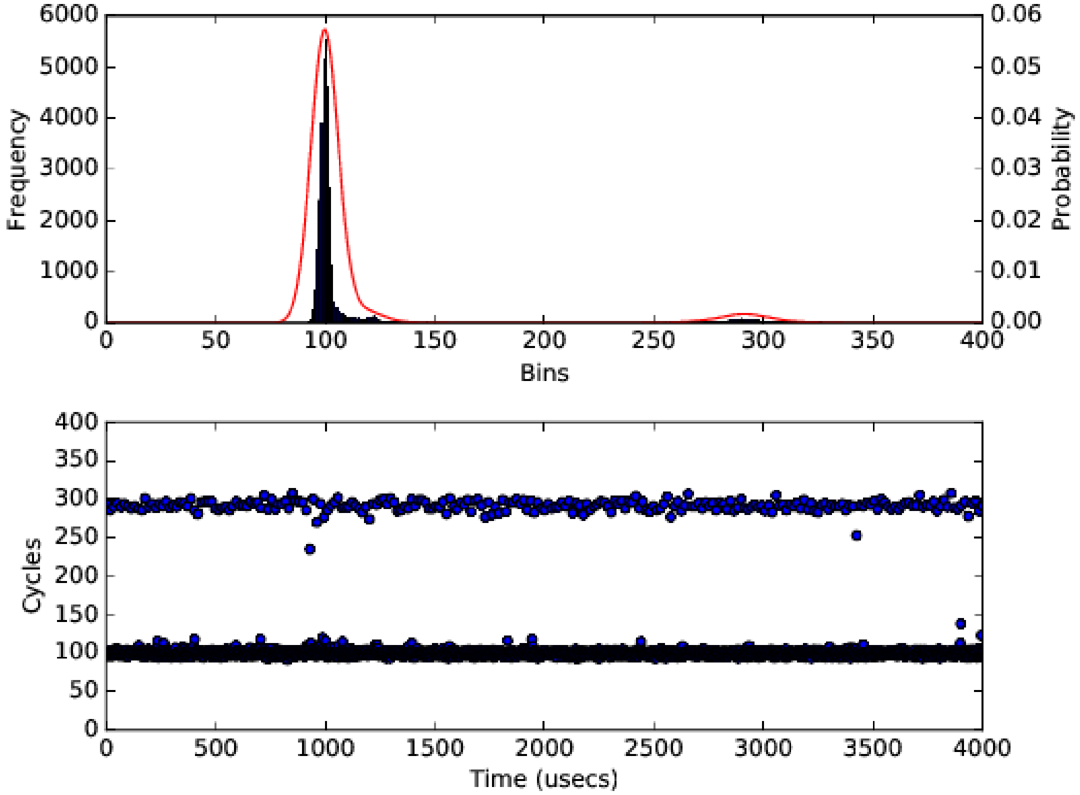
- Issue 4K aligned load half the time
- Clear 4K signal

Improving 4K-Aliasing Threshold



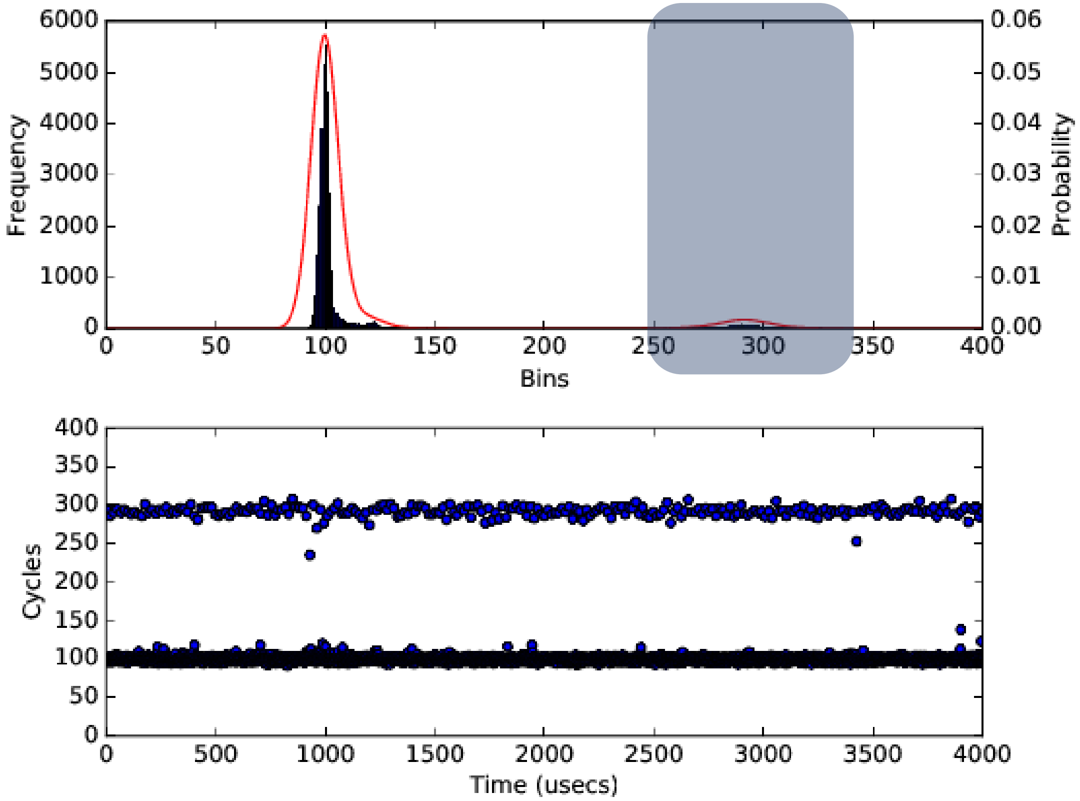
- Issue 4K aligned load half the time
- Clear 4K signal
- Noisy threshold

4K-Aliasing Modulation: Separated by 256 B



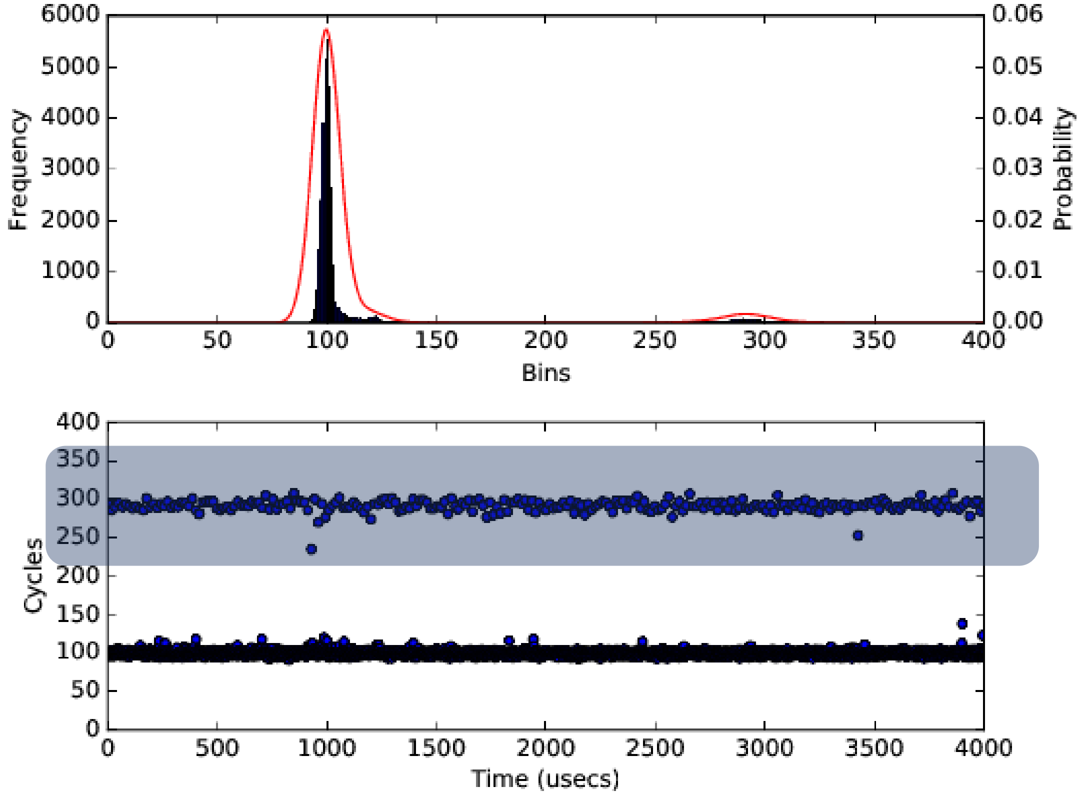
- Issue 4K aligned load every 16th time

4K-Aliasing Modulation: Separated by 256 B



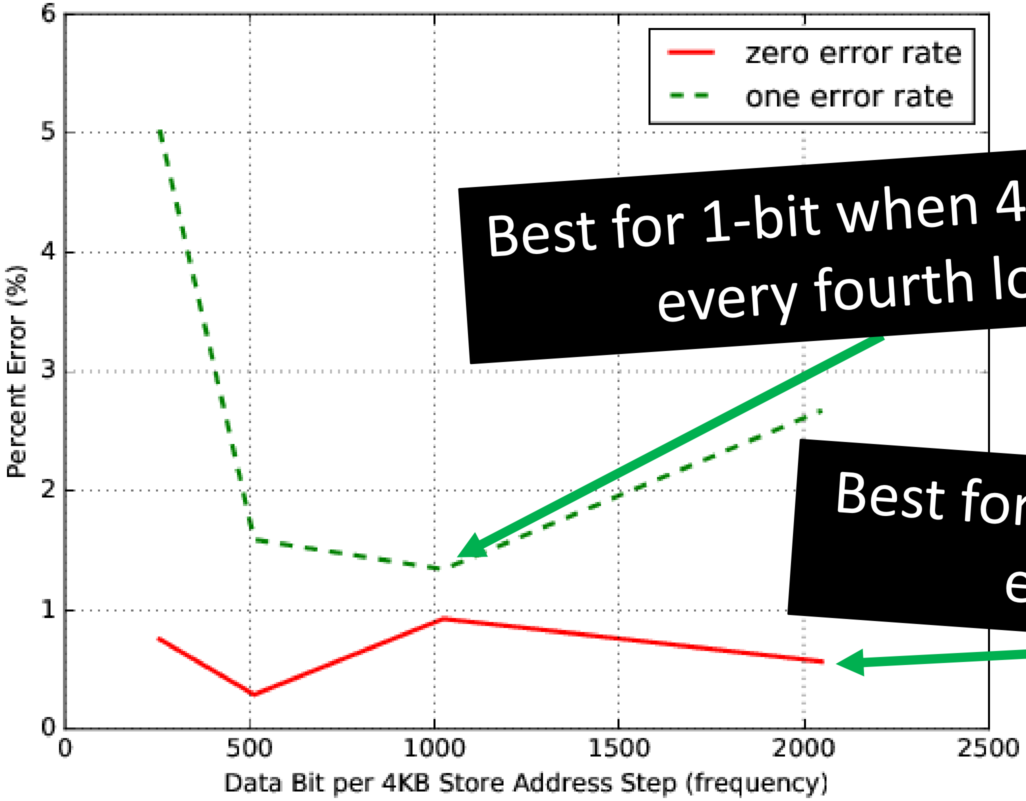
- Issue 4K aligned load every 16th time
- 4K signal?

4K-Aliasing Modulation: Separated by 256 B



- Issue 4K aligned load every 16th time
- 4K signal?
- Better threshold

4K Load vs. Error Rate



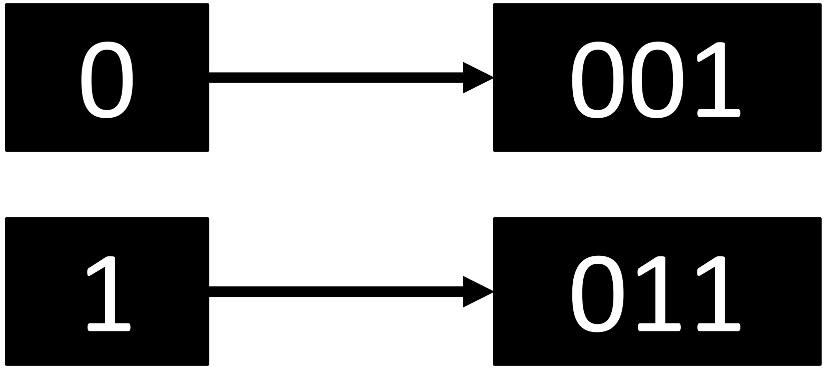
Best for 1-bit when 4K aligned every fourth load

Best for 0-bit when 4K aligned every other load

Communication Protocol?

Detecting Sender

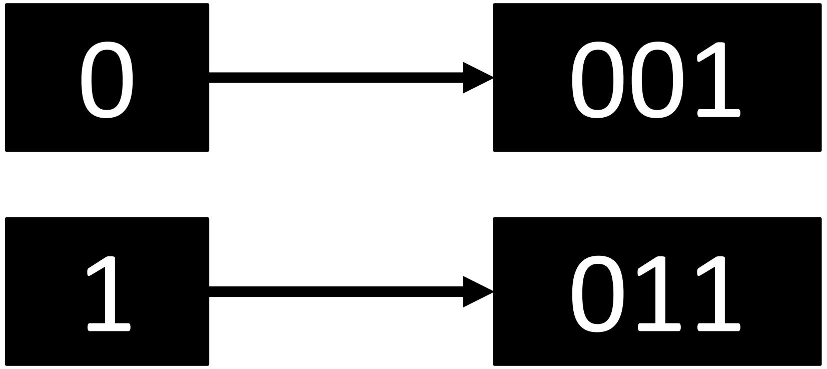
1-wire communication


0 → 001

1 → 011

Detecting Sender

1-wire communication



Automatically detect sender

Detecting Receiver

Use store-to-load forwarding loop

Competition for hyperthreading resources degrades performance

Message Recovery

Initialization and completion messages

- Our channel is fast, so we can deal with repeated tries

Break the message up into packets

- Limits impact of retransmission

In-House Channel Capacity

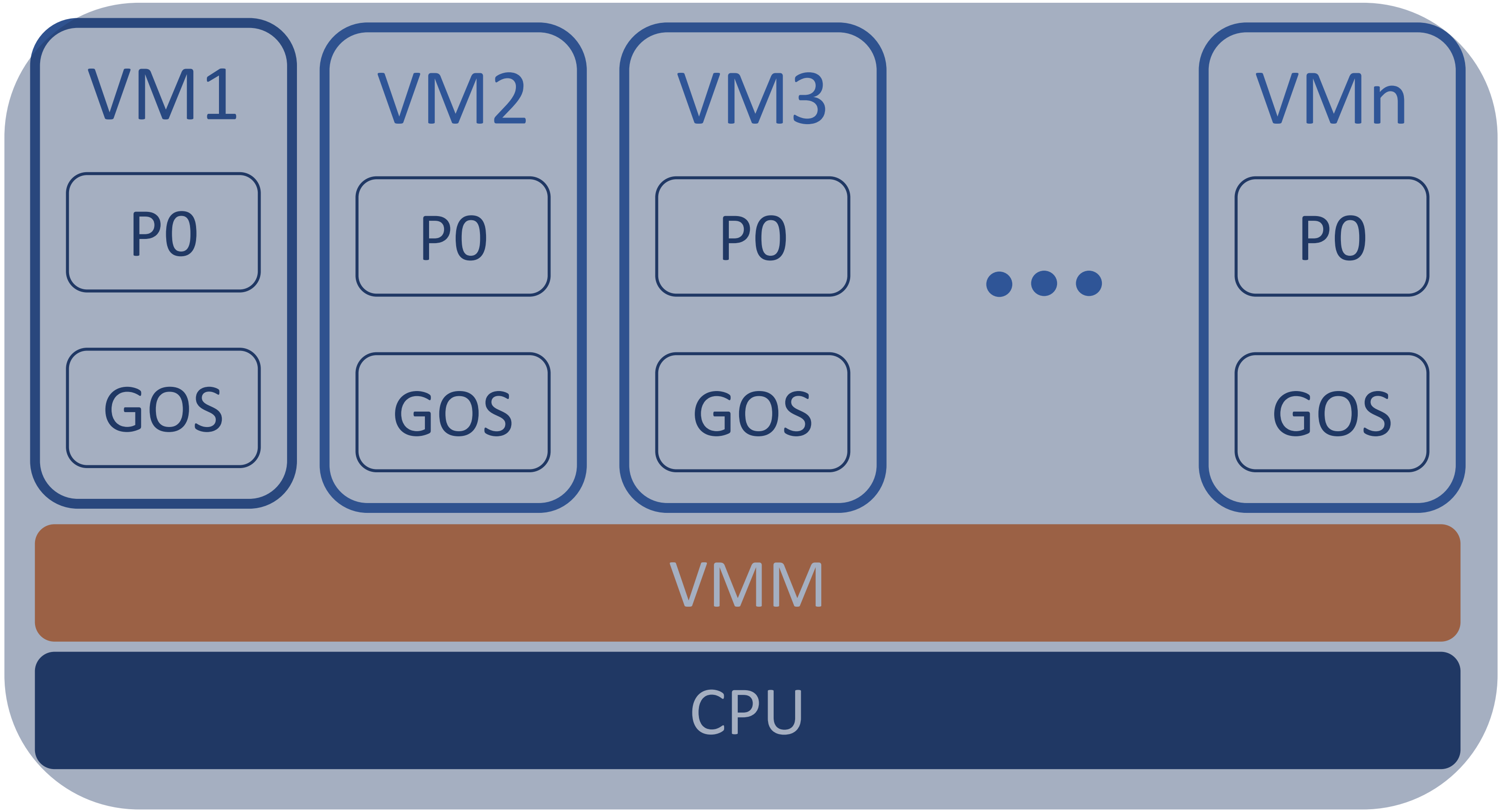
	256 B	512 B	1024 B	2048 B
ε_0	0.0075	0.0029 ✗	0.0093	0.0057
ε_1	0.0502	0.0159	0.0134	0.0267
Bits per Ch.	0.824	0.927	0.918	0.886
Ch. Cap (Mbps)	1.62	1.83	1.81	1.75

In-House Channel Capacity

	256 B	512 B	1024 B	2048 B
ε_0	0.0075	0.0029	0.0093	0.0057
ε_1	0.0502	0.0159	0.0134 ✓	0.0267
Bits per Ch.	0.824	0.927	0.918	0.886
Ch. Cap (Mbps)	1.62	1.83	1.81	1.75

In-House Channel Capacity

	256 B	512 B	1024 B	2048 B
ε_0	0.0075	0.0029	0.0093	0.0057
ε_1	0.0502	0.0159	0.0134	0.0267
Bits per Ch.	0.824	0.927	0.918	0.886
Ch. Cap (Mbps)	1.62	1.83	1.81	1.75


VM1
PO
GOS
VM2
PO
GOS
VM3
PO
GOS
• • •
VMn
PO
GOS
VMM
CPU

Requires HW Hyperthreading?!?

Requires HW Hyperthreading?!?

EC2 does it

GCE does it

Azure does it

Requires HW Hyperthreading?!?

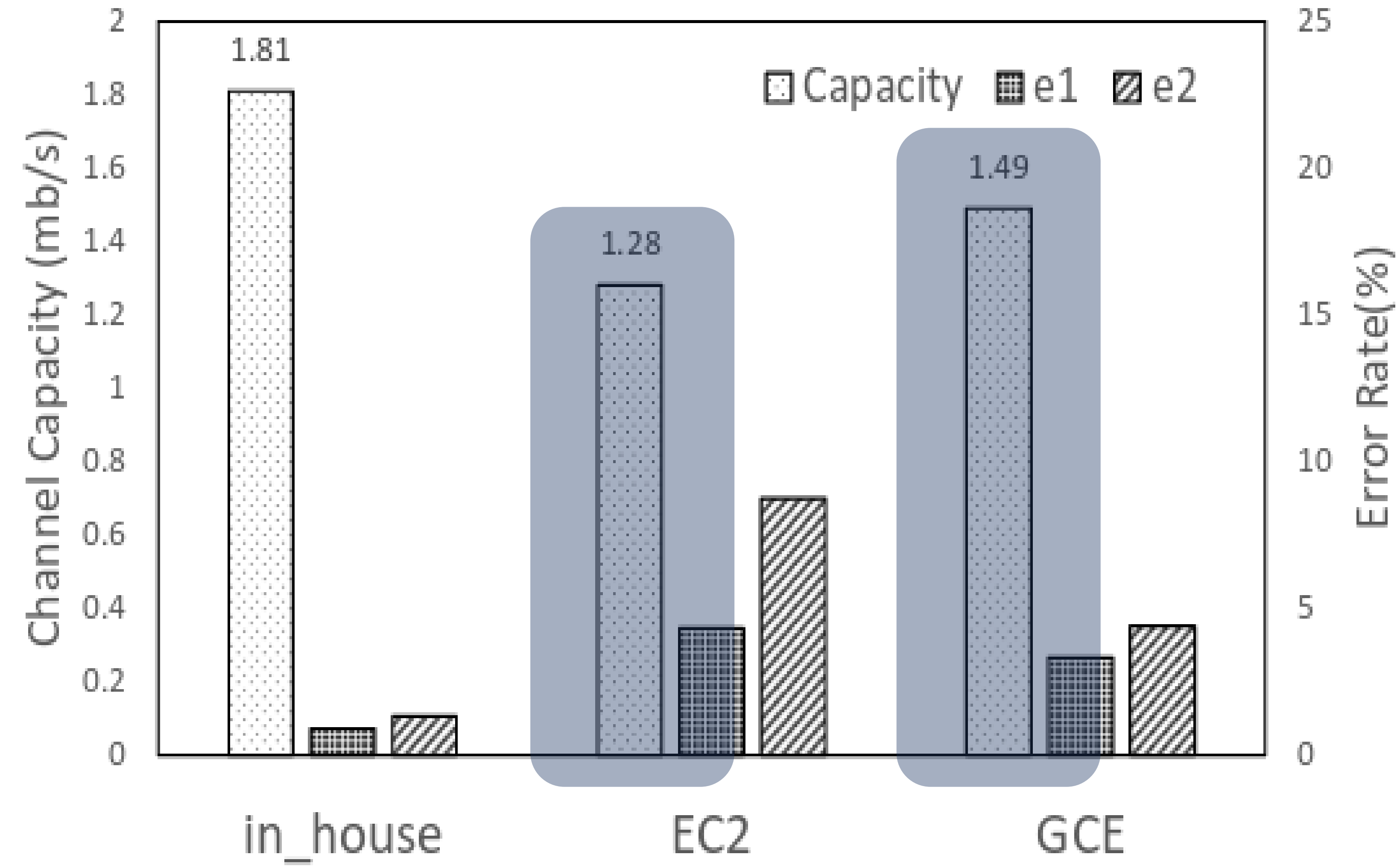
EC2 does it

GCE does it

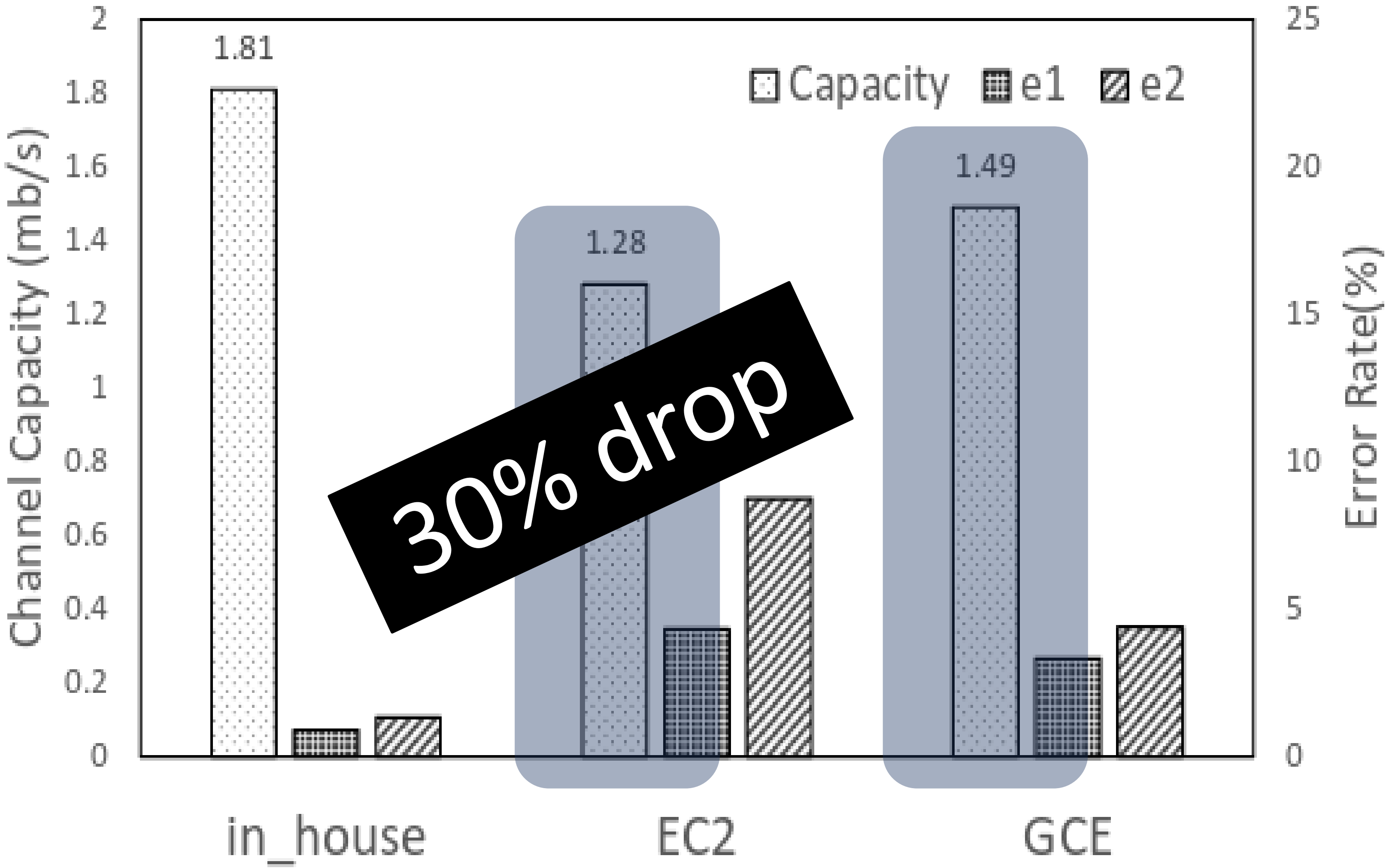
Azure does it

Lowers the total cost of ownership

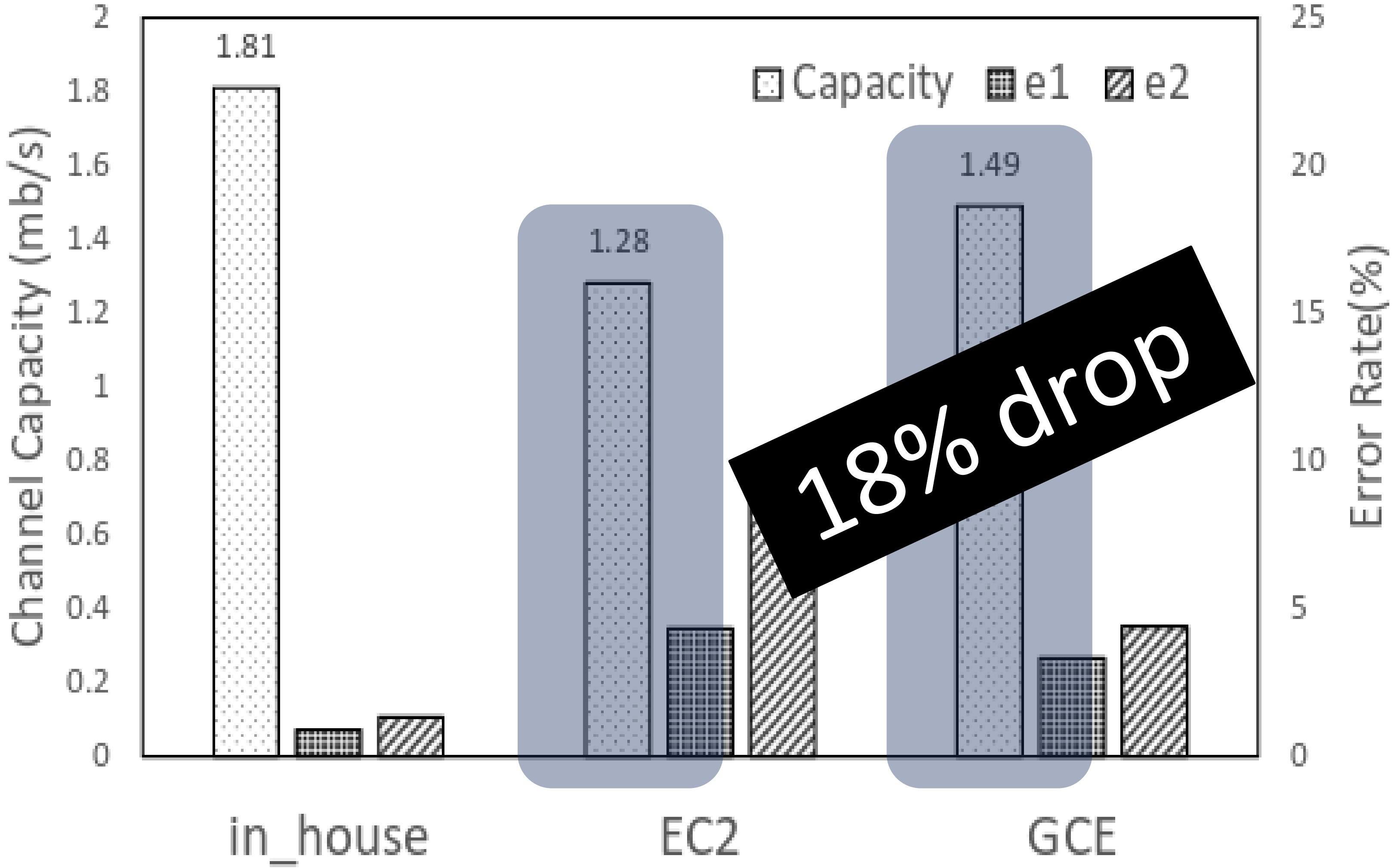
Amazon EC2 and Google CE



Amazon EC2 and Google CE

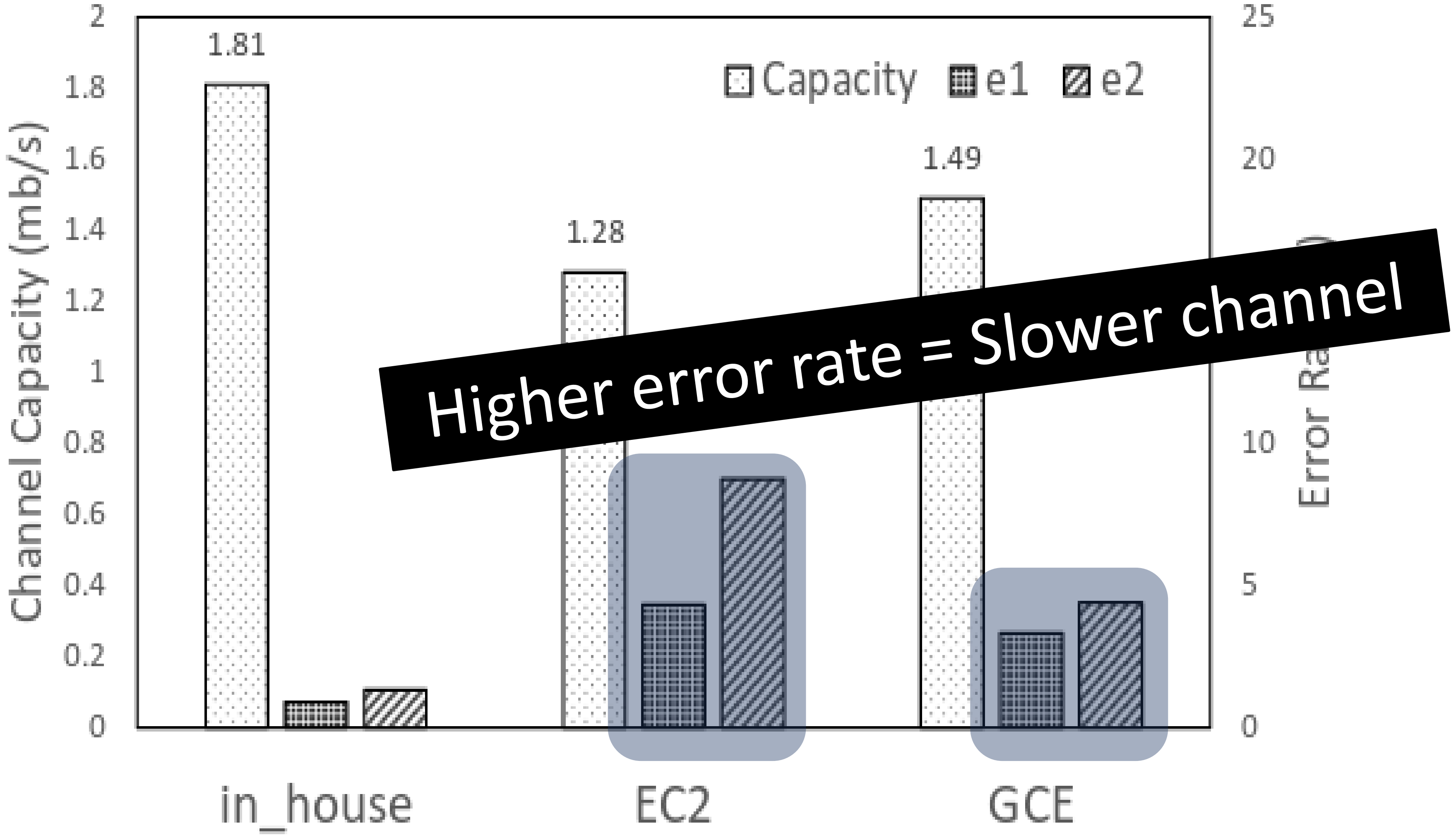


Amazon EC2 and Google CE

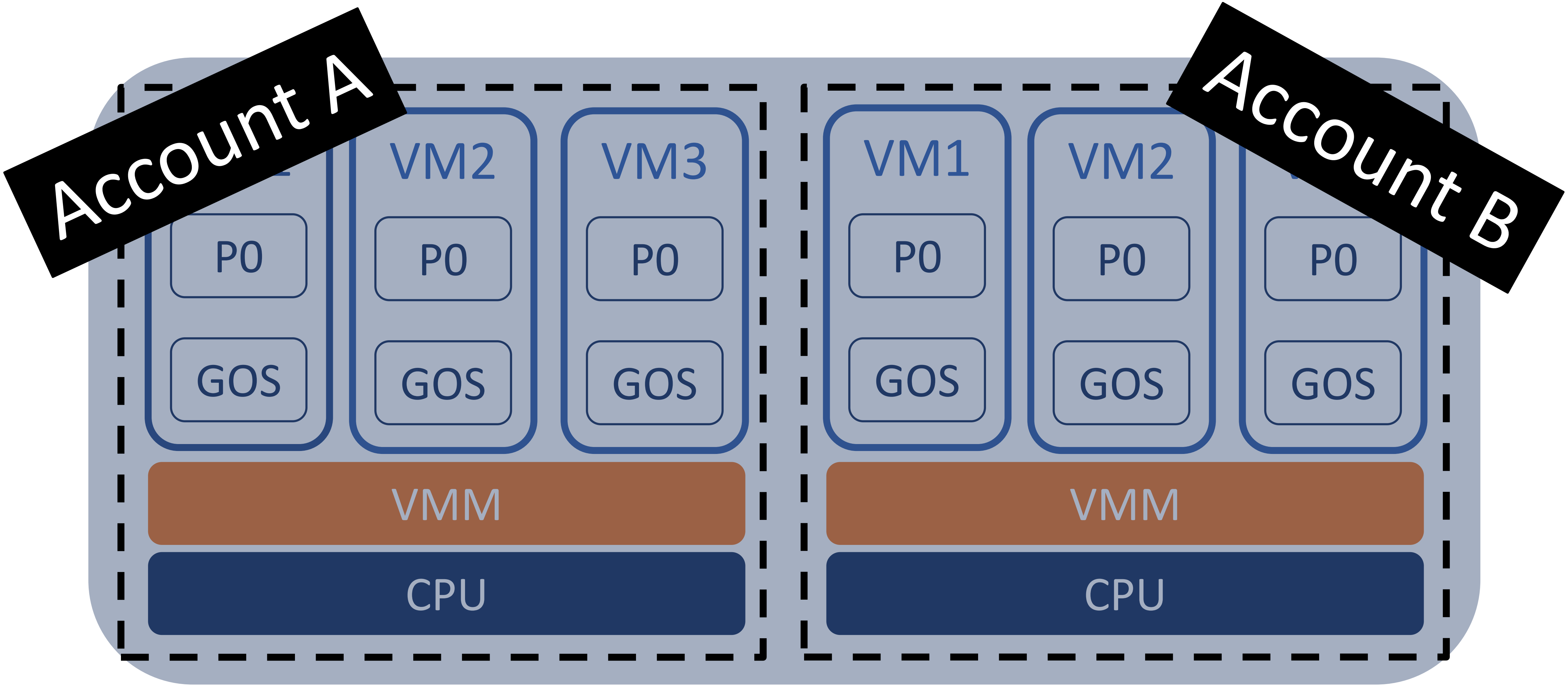


18% drop

Amazon EC2 and Google CE



Higher error rate = Slower channel


Account A
VM2
VM3
P0
GOS
P0
GOS
P0
GOS
VMM
CPU
Account B
VM1
VM2
P0
GOS
P0
GOS
P0
GOS
VMM
CPU

Challenges

Separating 4K-aliasing from background noise

- Establish baseline without cooperating VM
- Iteratively scale-up VM instances transmitting 4K signal
- Repeat the measurement 5 times

Challenges

Launch strategy

- Launch pairwise sender and receiver VMs
- Utilize prior [1] colocation placement strategies
- Scale up to 20 pairwise sender/receiver VMs

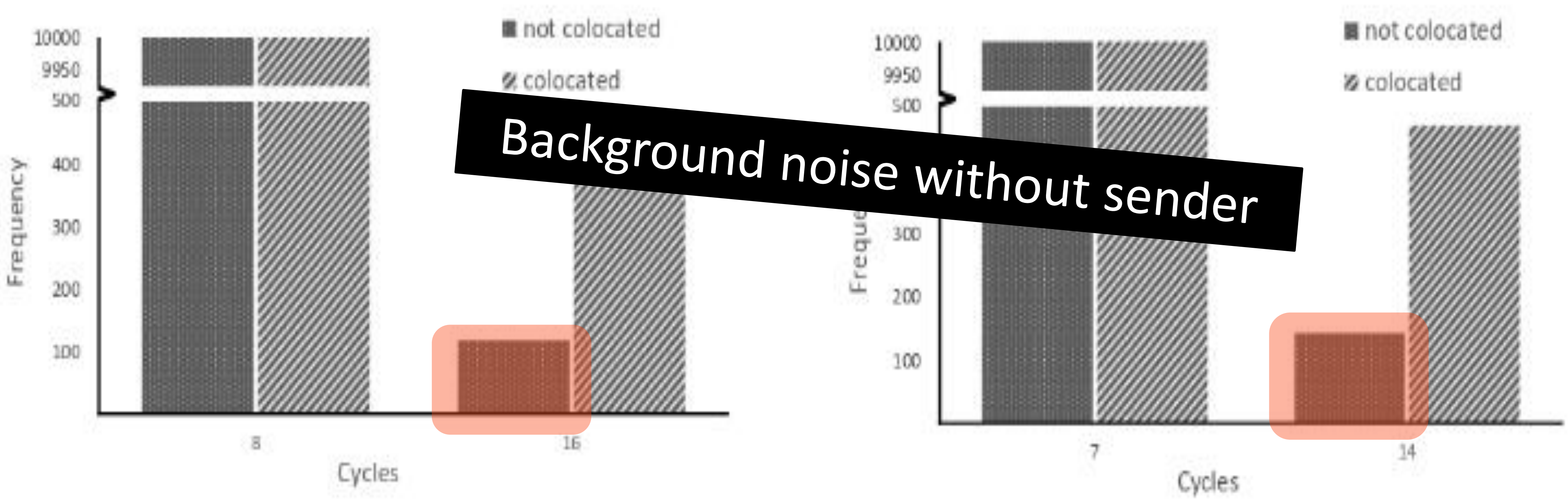
[1] V. Varadarajan, et al. **A placement vulnerability study in multi-tenant public clouds**. Usenix Security Symposium, 2015.

Challenges

Efficient test setup

- Sender continuously transmits/Receiver polls for 4K-aliasing for 10 s
- Decrease measurement time by launching all senders at once
- Sequentially launch receiver VMs every hour

Colocation Results

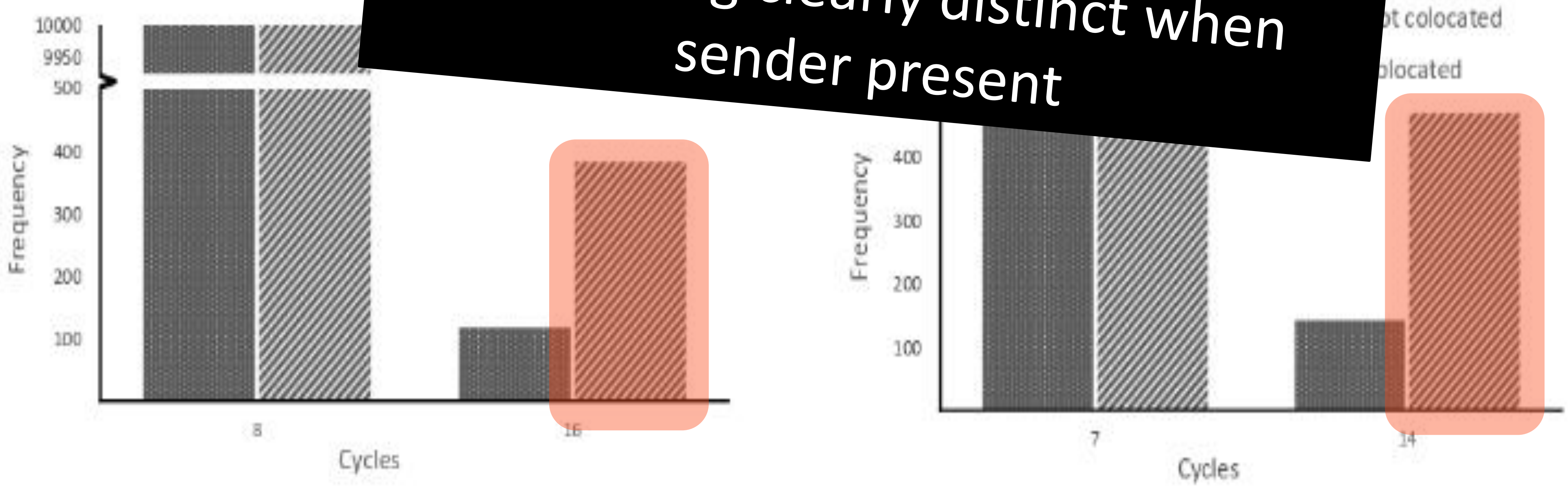


Background noise without sender

(a) EC2

(b) GCE

Colocation Results



4K-aliasing clearly distinct when sender present

(a) EC2

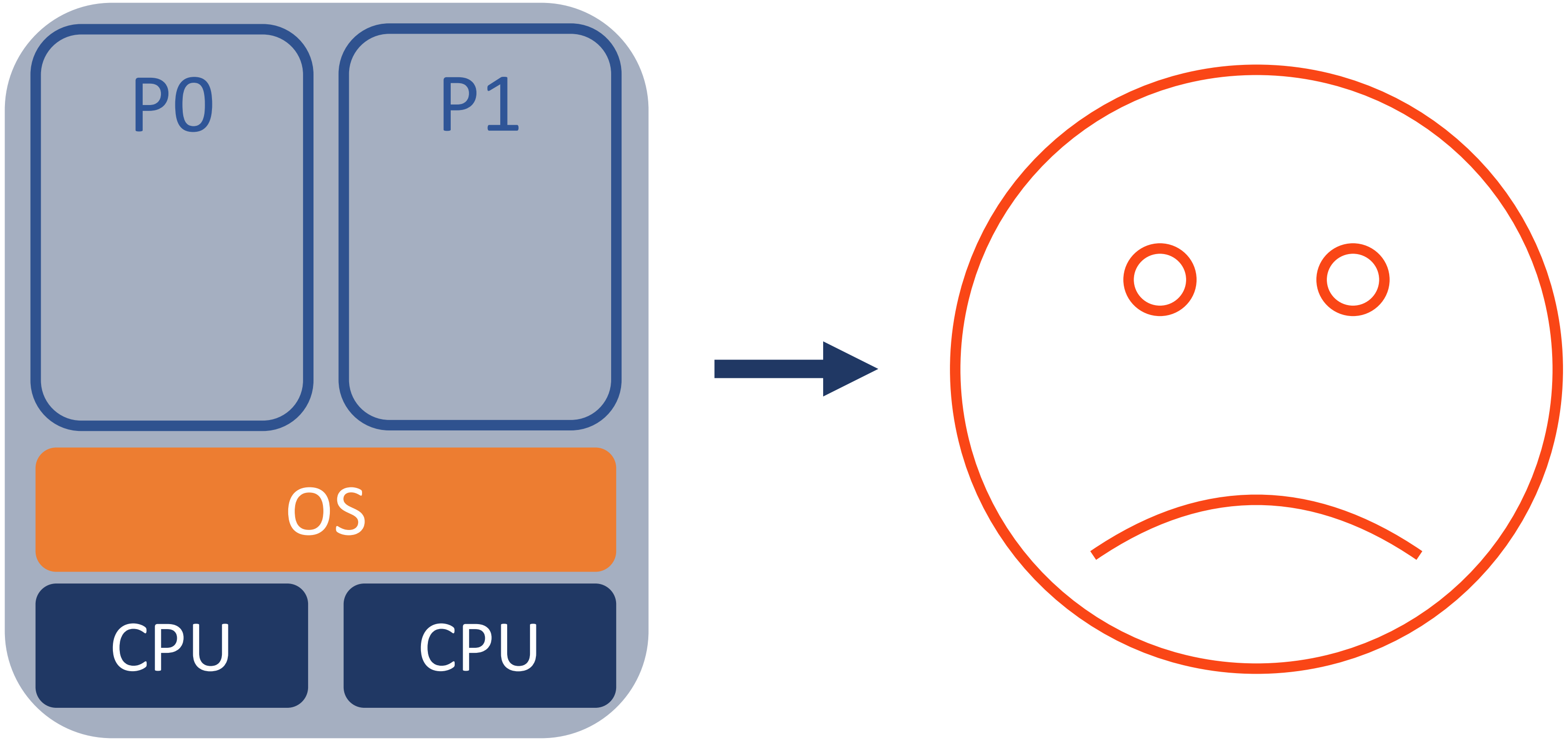
(b) GCE

Colocation Results

As good as cross-core multi-tenant detection techniques

- WRT launch strategy
- No. of instance pairs to detect multi-tenancy

What about Cross-Core?



M.F. Chowdury and D.M. Carmean. Maintaining processor ordering by checking load addresses of unretired load instructions against snooping store address. Feb 3 2004, US Patent 6,687,809

Conclusion

Out-of-Order execution and speculative execution are new attack vectors

4K-aliasing (ab)uses speculation on memory instructions and the microarchitecture used to maintain memory consistency

We demonstrate 4K-aliasing on public IaaS clouds

- Fast and robust covert channel
- Practical multi-tenant detection

UF | UNIVERSITY of FLORIDA

Questions?