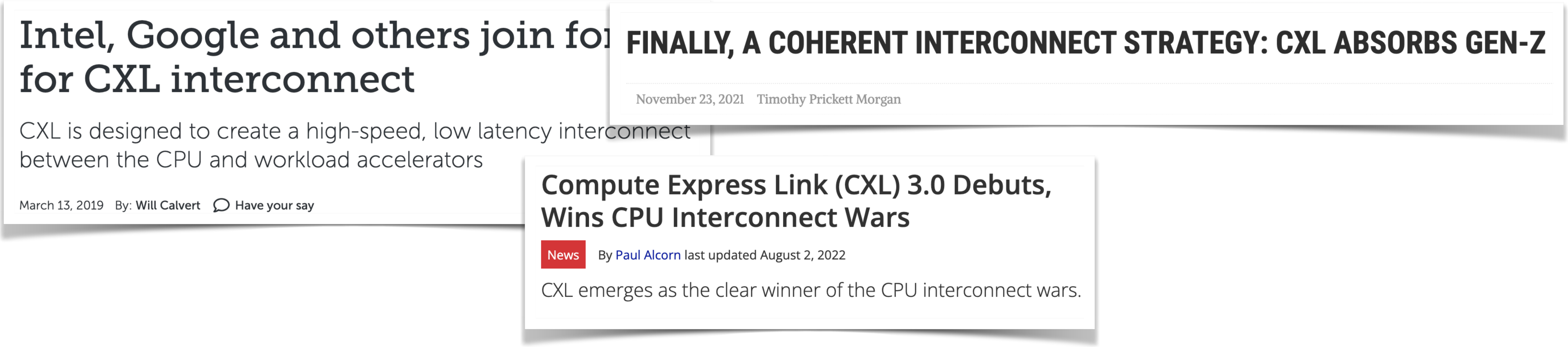
BROWN

Rethinking Secure NVMs in the Age of CXL

Samuel Thomas

CXL is here!



Intel, Google and others join for
for CXL interconnect

CXL is designed to create a high-speed, low latency interconnect between the CPU and workload accelerators

March 13, 2019 By: Will Calvert Have your say

FINALLY, A COHERENT INTERCONNECT STRATEGY: CXL ABSORBS GEN-Z

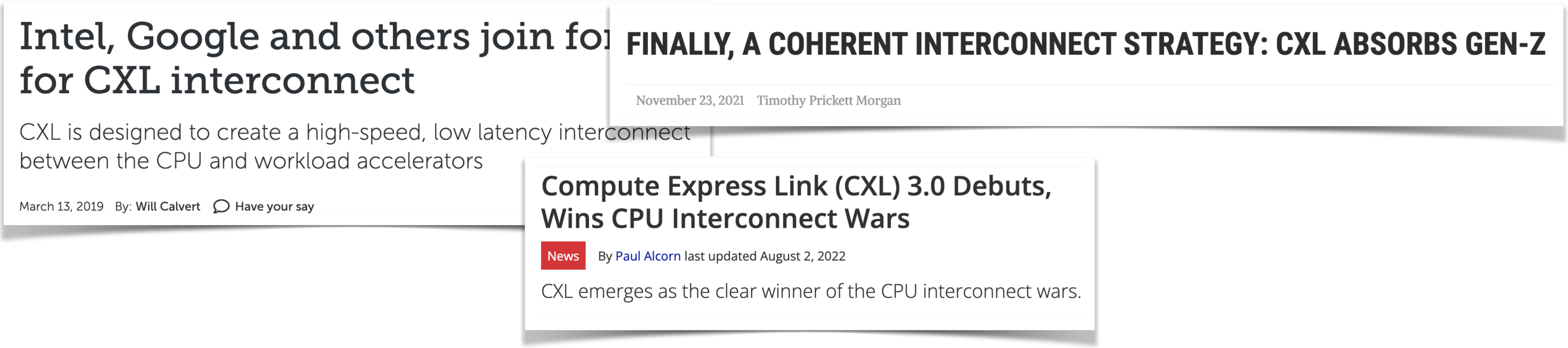
November 23, 2021 Timothy Prickett Morgan

Compute Express Link (CXL) 3.0 Debuts, Wins CPU Interconnect Wars

News By Paul Alcorn last updated August 2, 2022

CXL emerges as the clear winner of the CPU interconnect wars.

CXL is here!



Intel, Google and others join for
for CXL interconnect

CXL is designed to create a high-speed, low latency interconnect between the CPU and workload accelerators

March 13, 2019 By: Will Calvert Have your say

FINALLY, A COHERENT INTERCONNECT STRATEGY: CXL ABSORBS GEN-Z

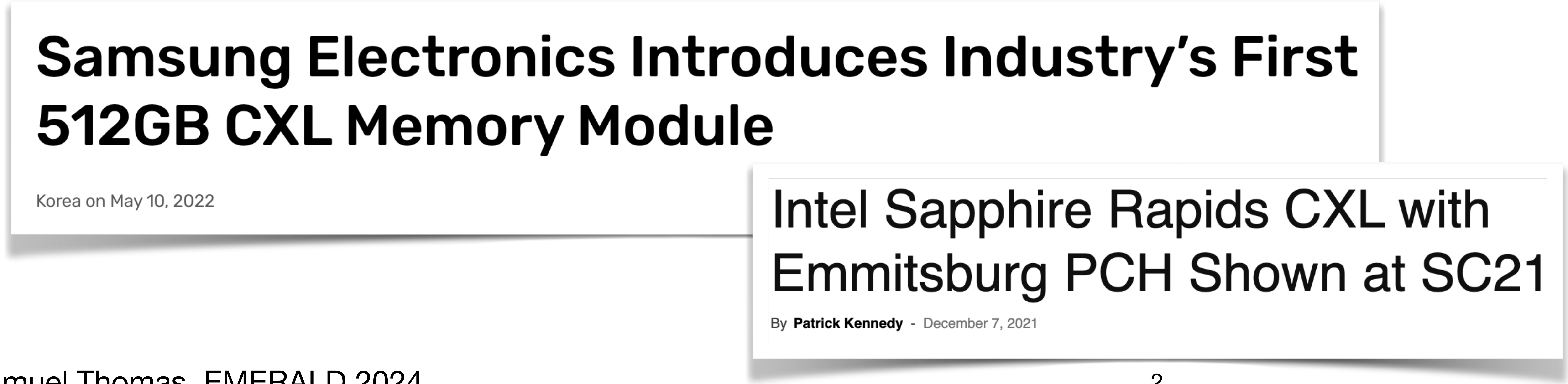
November 23, 2021 Timothy Prickett Morgan

Compute Express Link (CXL) 3.0 Debuts, Wins CPU Interconnect Wars

News By Paul Alcorn last updated August 2, 2022

CXL emerges as the clear winner of the CPU interconnect wars.

CXL is here!

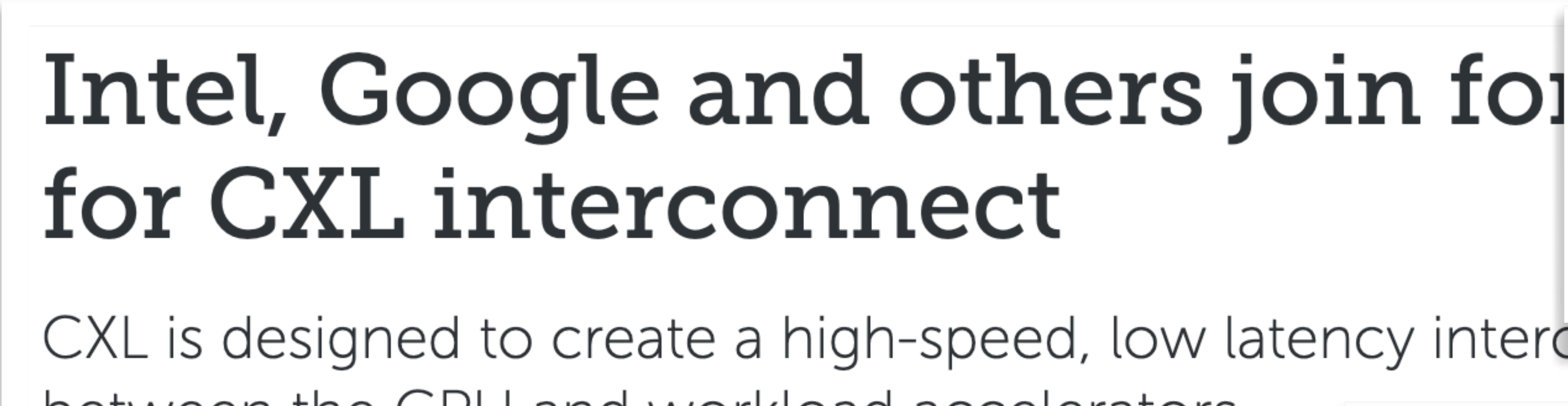


Samsung Electronics Introduces Industry's First 512GB CXL Memory Module

Korea on May 10, 2022

Intel Sapphire Rapids CXL with Emmitsburg PCH Shown at SC21

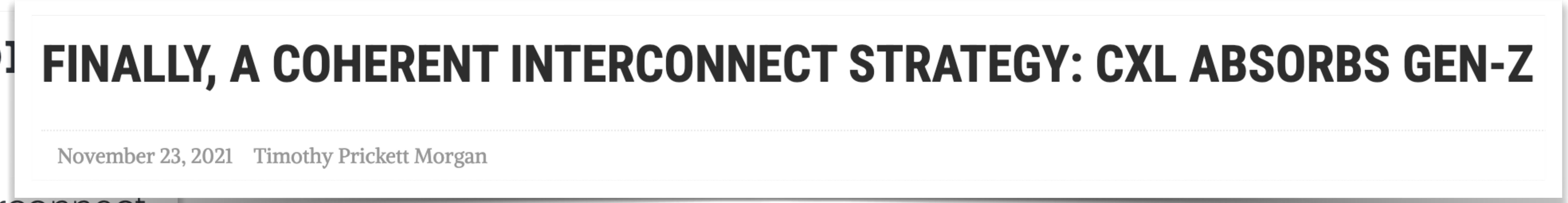
By Patrick Kennedy - December 7, 2021



Intel, Google and others join for CXL interconnect

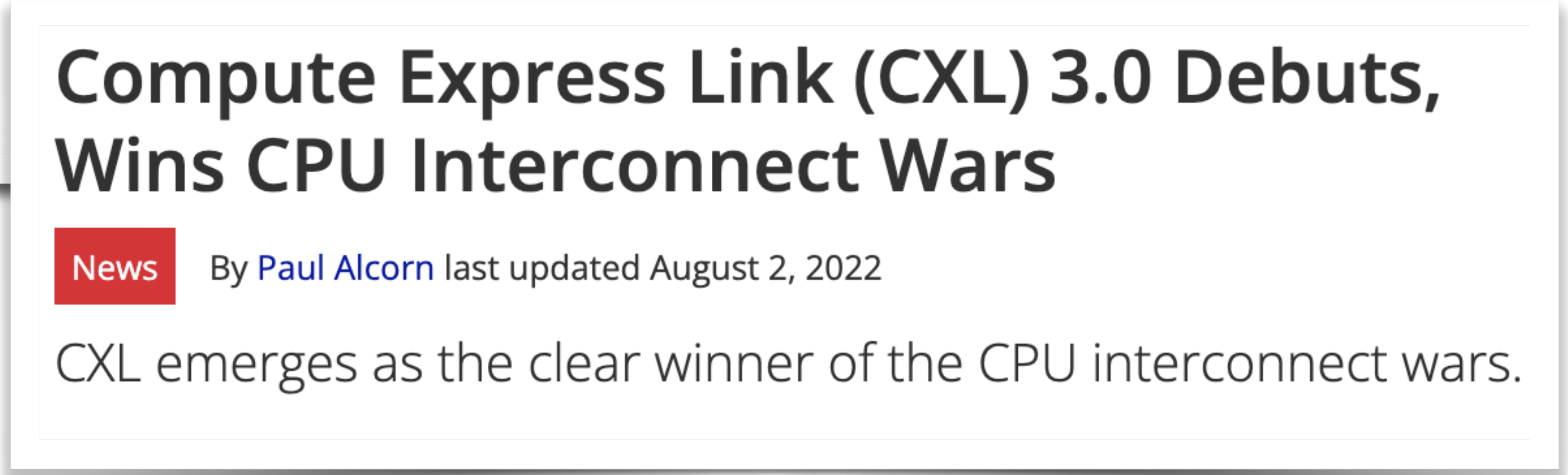
CXL is designed to create a high-speed, low latency interconnect between the CPU and workload accelerators

March 13, 2019 By: Will Calvert Have your say



FINALLY, A COHERENT INTERCONNECT STRATEGY: CXL ABSORBS GEN-Z

November 23, 2021 Timothy Prickett Morgan

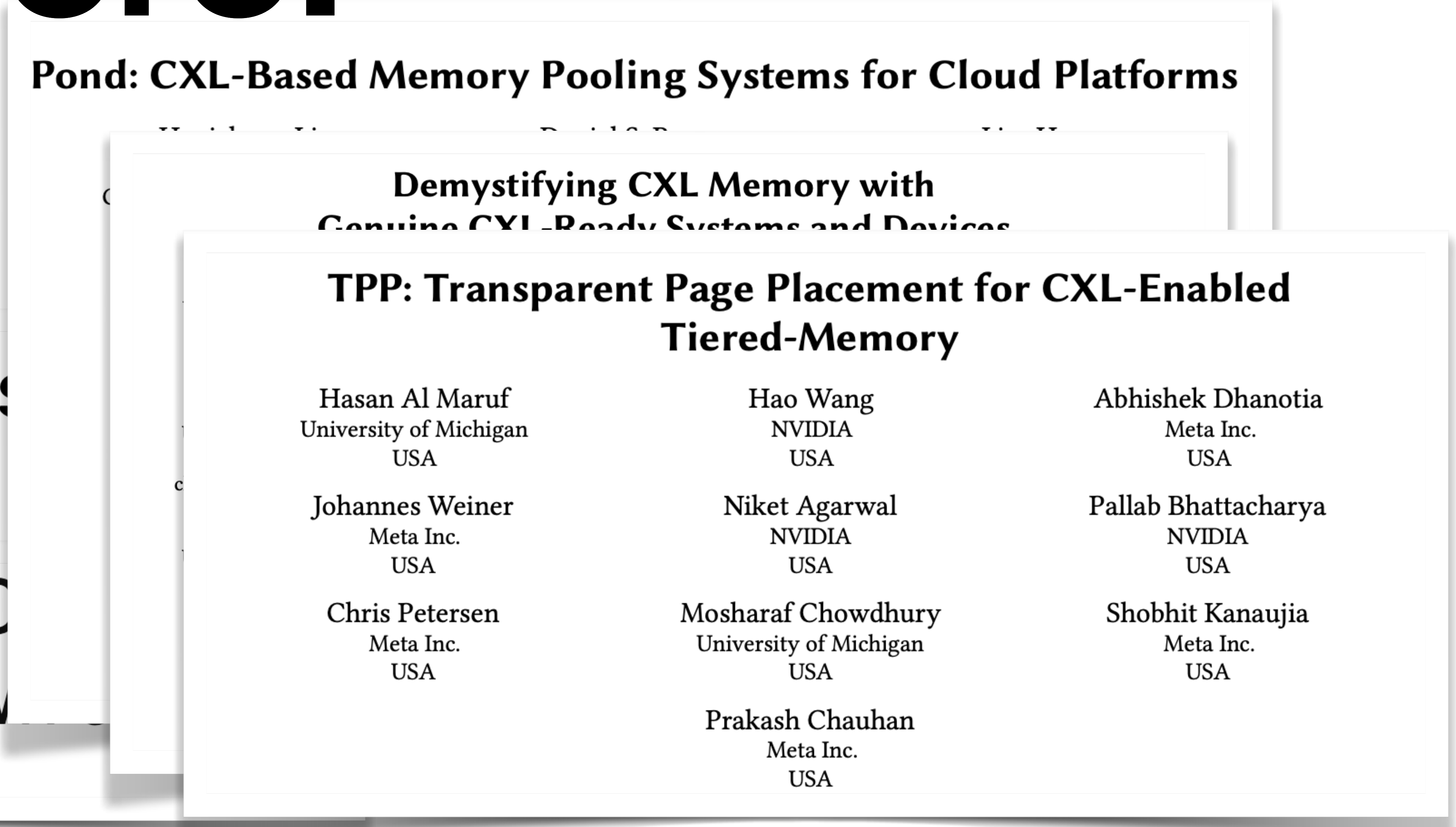


Compute Express Link (CXL) 3.0 Debuts, Wins CPU Interconnect Wars

News By Paul Alcorn last updated August 2, 2022

CXL emerges as the clear winner of the CPU interconnect wars.

CXL is here!



Pond: CXL-Based Memory Pooling Systems for Cloud Platforms

Demystifying CXL Memory with Genuine CXL-Ready Systems and Devices

TPP: Transparent Page Placement for CXL-Enabled Tiered-Memory

Hasan Al Maruf
University of Michigan
USA

Hao Wang
NVIDIA
USA

Abhishek Dhanotia
Meta Inc.
USA

Johannes Weiner
Meta Inc.
USA

Niket Agarwal
NVIDIA
USA

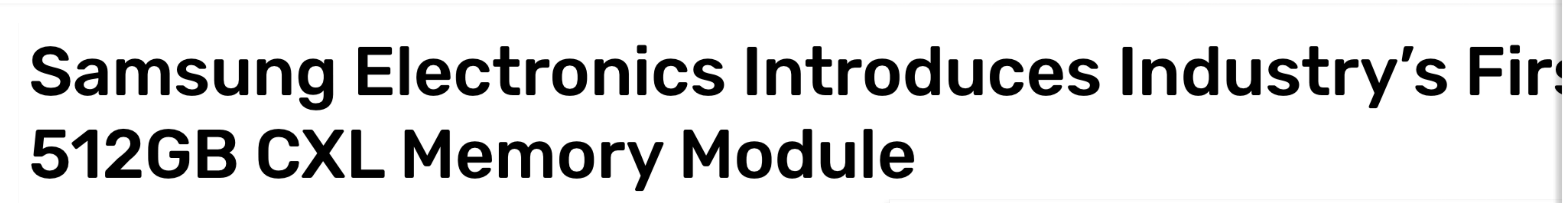
Pallab Bhattacharya
NVIDIA
USA

Chris Petersen
Meta Inc.
USA

Mosharaf Chowdhury
University of Michigan
USA

Shobhit Kanaujia
Meta Inc.
USA

Prakash Chauhan
Meta Inc.
USA

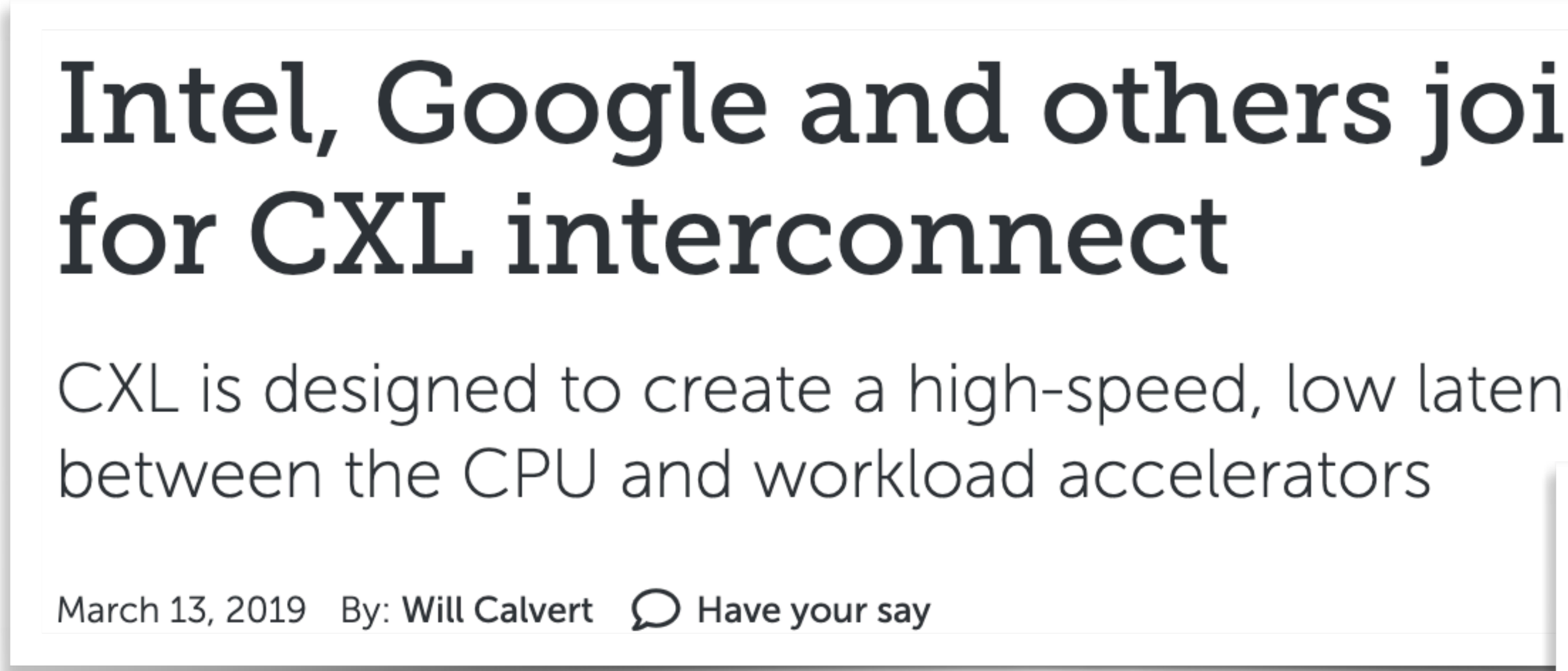


Samsung Electronics Introduces Industry's Firs 512GB CXL Memory Module

Korea on May 10, 2022

Intel Sapphire Rapids C Emmitsburg PCH Show

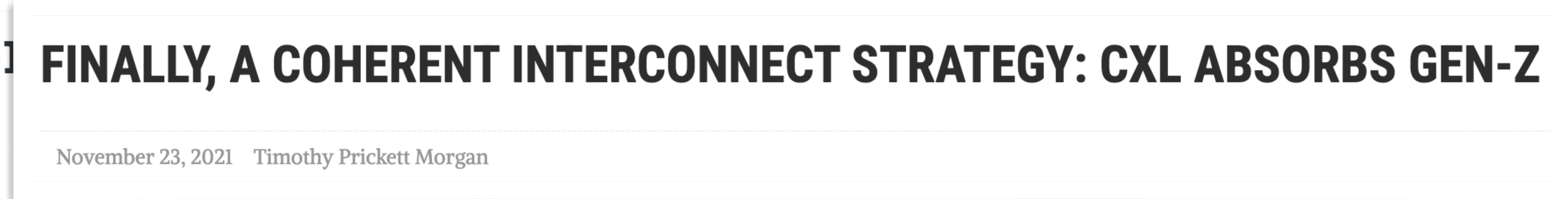
By Patrick Kennedy - December 7, 2021



Intel, Google and others join for CXL interconnect

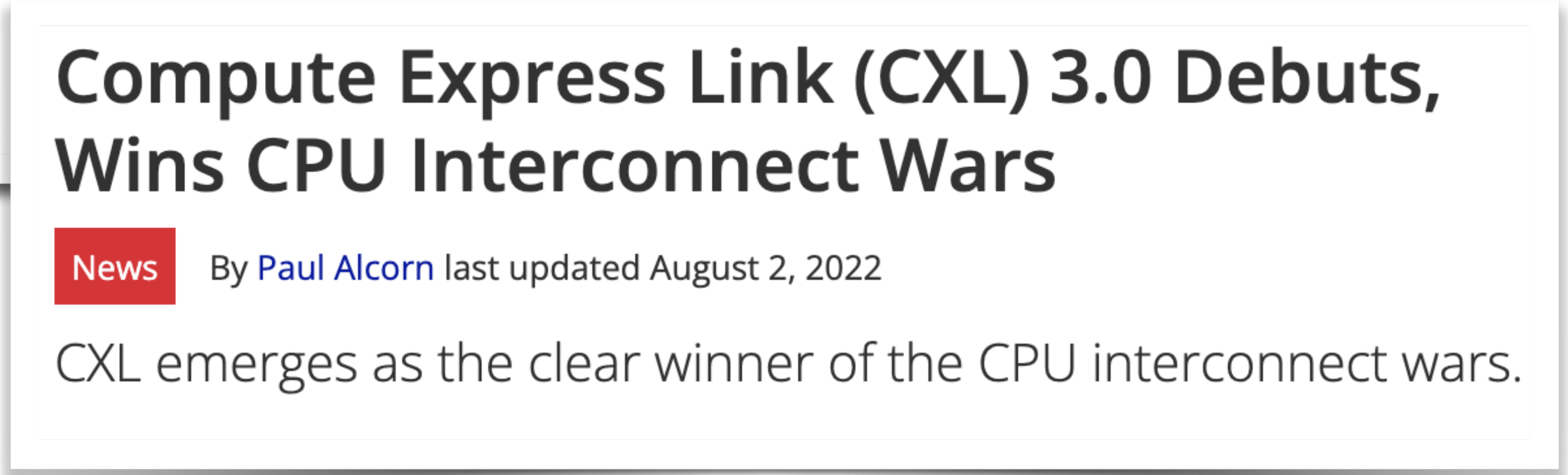
CXL is designed to create a high-speed, low latency interconnect between the CPU and workload accelerators

March 13, 2019 By: Will Calvert Have your say



FINALLY, A COHERENT INTERCONNECT STRATEGY: CXL ABSORBS GEN-Z

November 23, 2021 Timothy Prickett Morgan



Compute Express Link (CXL) 3.0 Debuts, Wins CPU Interconnect Wars

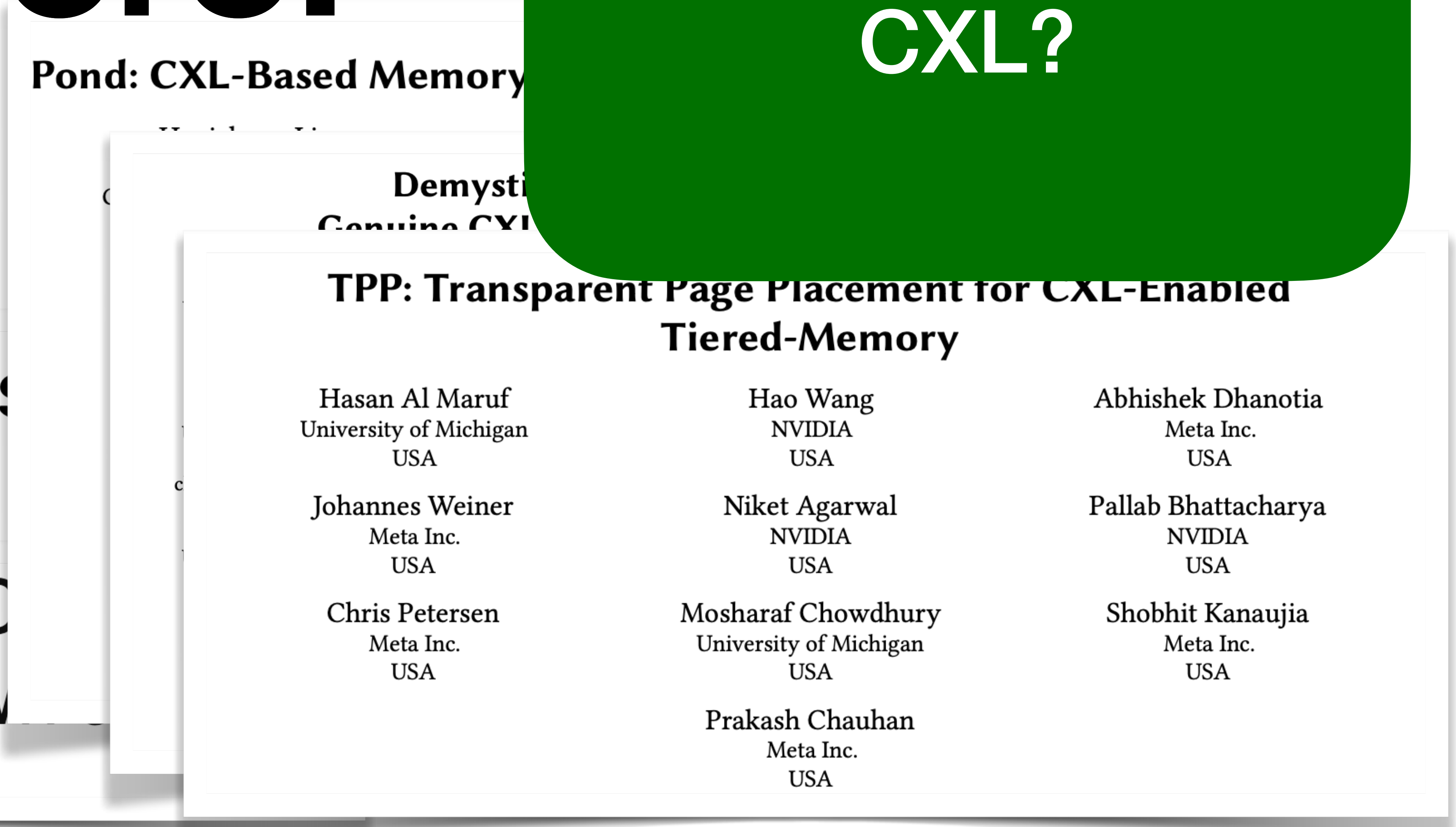
News By Paul Alcorn last updated August 2, 2022

CXL emerges as the clear winner of the CPU interconnect wars.

How should we think about the CXL architecture?

CXL is here!

How should we design systems for CXL?



Pond: CXL-Based Memory

Demysti

TPP: Transparent Page Placement for CXL-Enabled Tiered-Memory

Hasan Al Maruf
University of Michigan
USA

Hao Wang
NVIDIA
USA

Abhishek Dhanotia
Meta Inc.
USA

Johannes Weiner
Meta Inc.
USA

Niket Agarwal
NVIDIA
USA

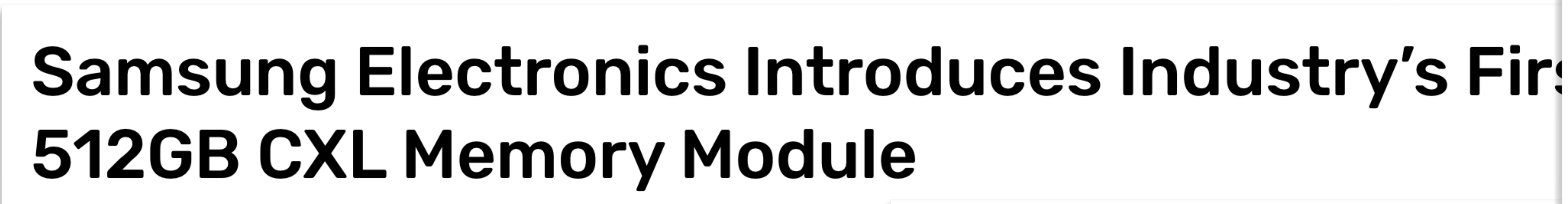
Pallab Bhattacharya
NVIDIA
USA

Chris Petersen
Meta Inc.
USA

Mosharaf Chowdhury
University of Michigan
USA

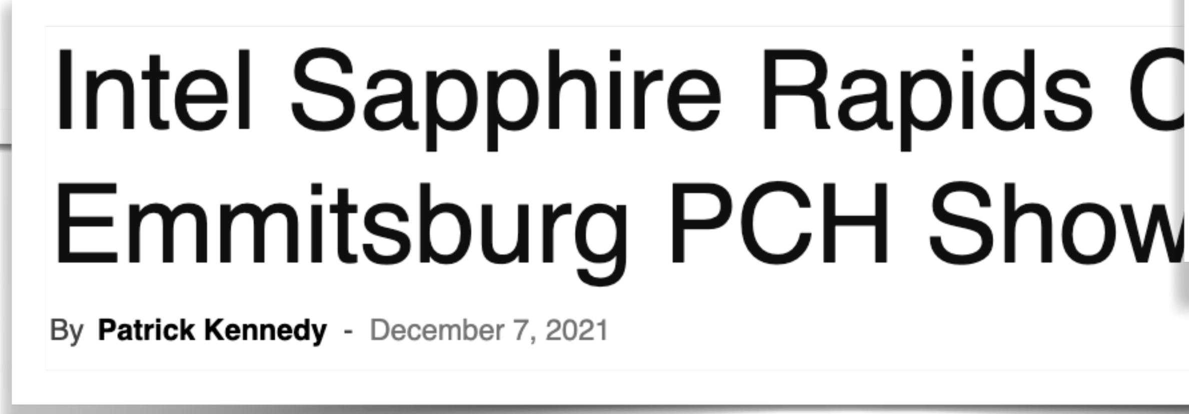
Shobhit Kanaujia
Meta Inc.
USA

Prakash Chauhan
Meta Inc.
USA



Samsung Electronics Introduces Industry's Firs 512GB CXL Memory Module

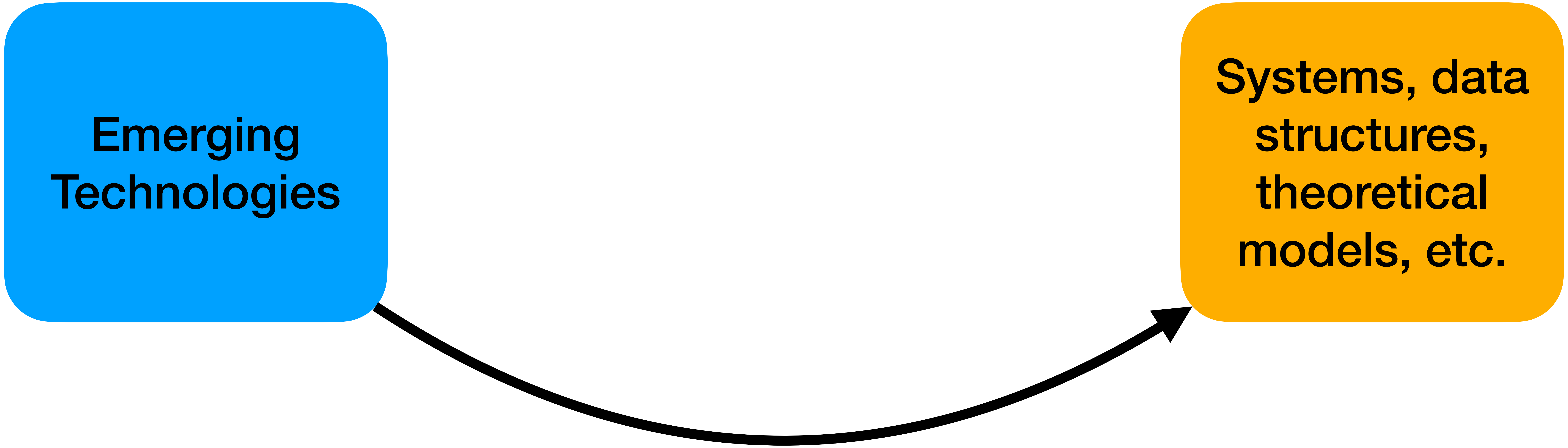
Korea on May 10, 2022



Intel Sapphire Rapids C Emmitsburg PCH Show

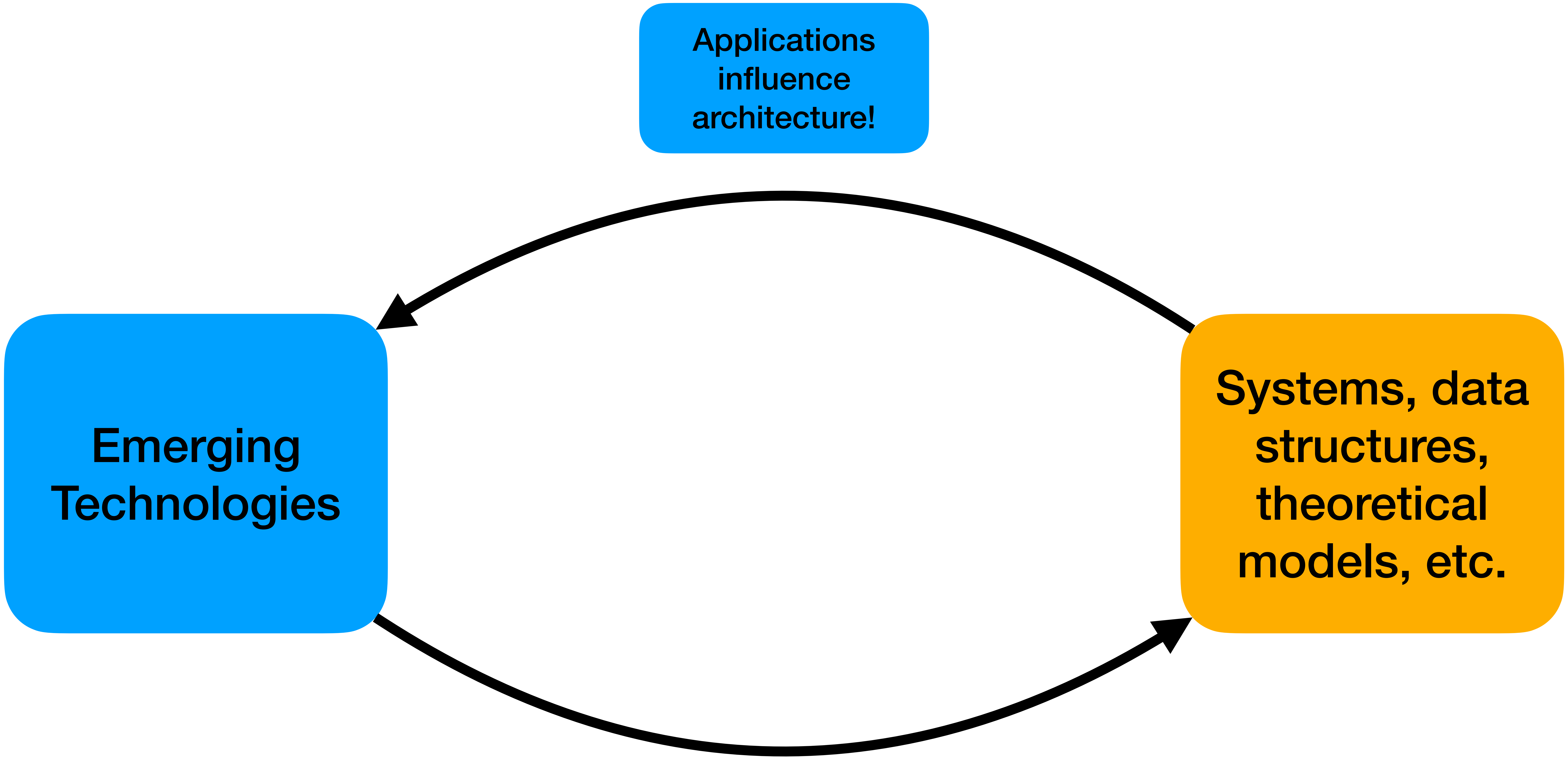
By Patrick Kennedy - December 7, 2021

Emerging Technologies



Emerging Technologies

Systems, data structures, theoretical models, etc.

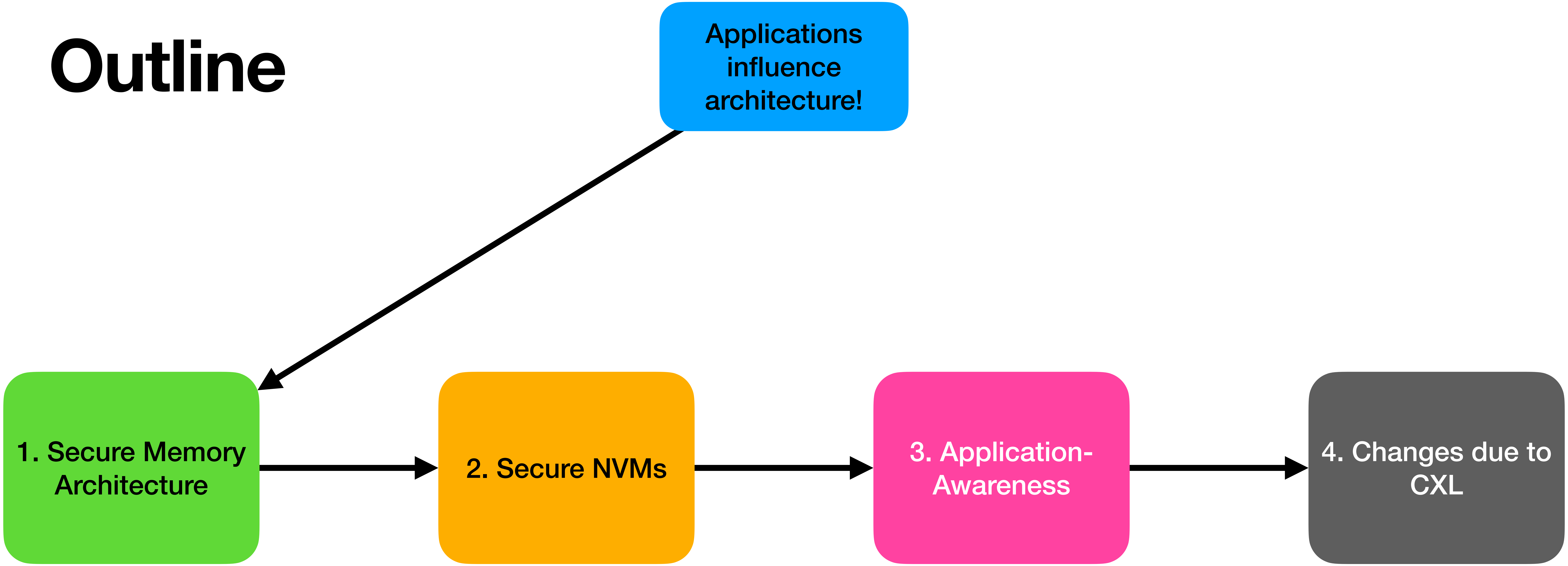


Applications influence architecture!

Emerging Technologies

Systems, data structures, theoretical models, etc.

Outline



Applications influence architecture!

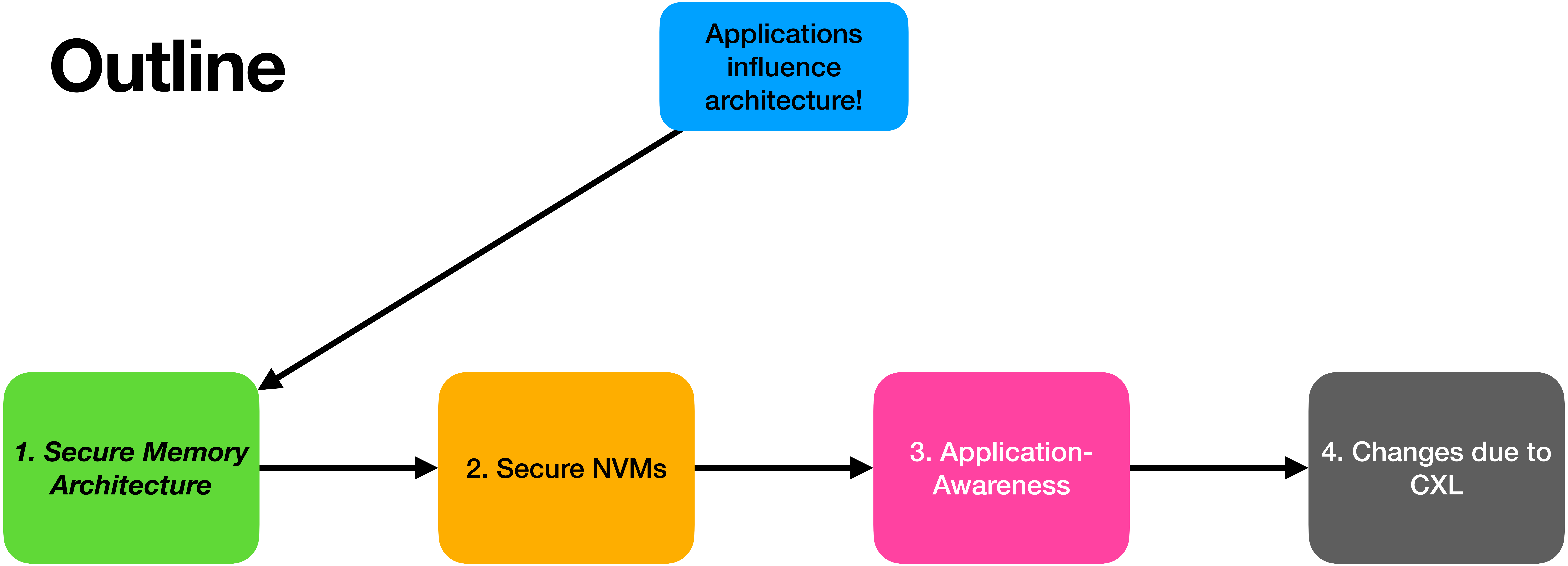
1. Secure Memory Architecture

2. Secure NVMs

3. Application-Awareness

4. Changes due to CXL

Outline



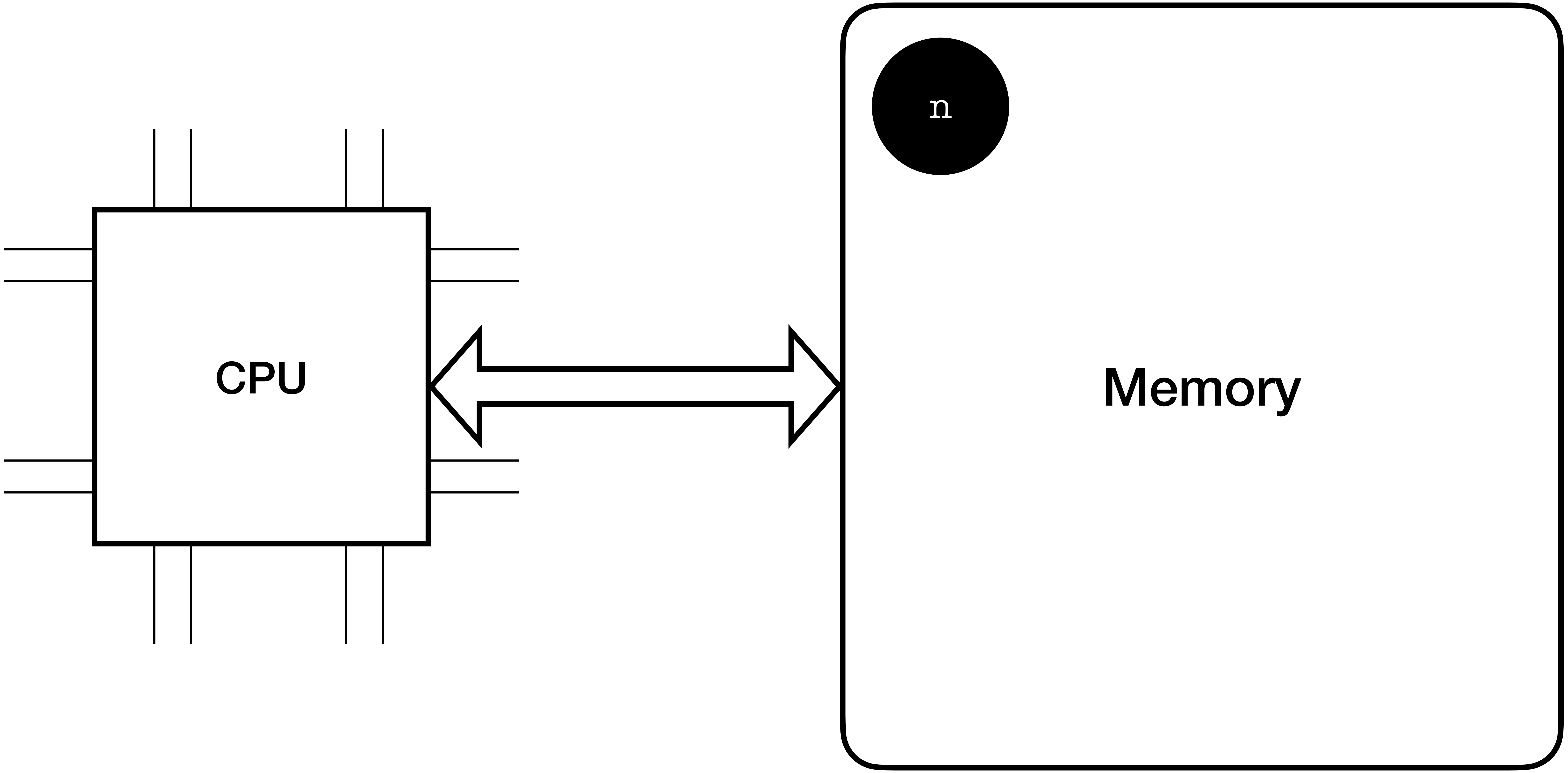
Applications influence architecture!

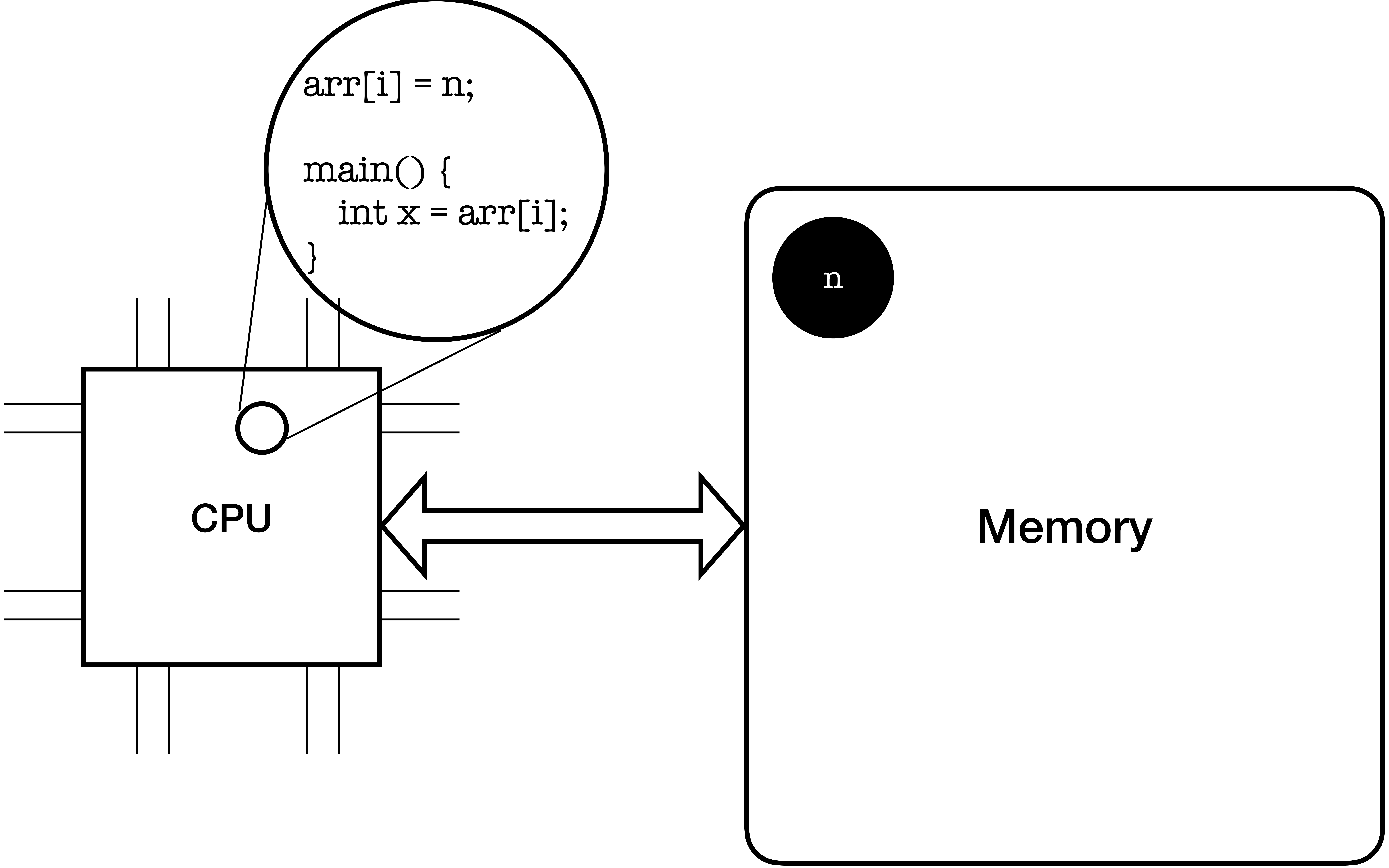
1. Secure Memory Architecture

2. Secure NVMs

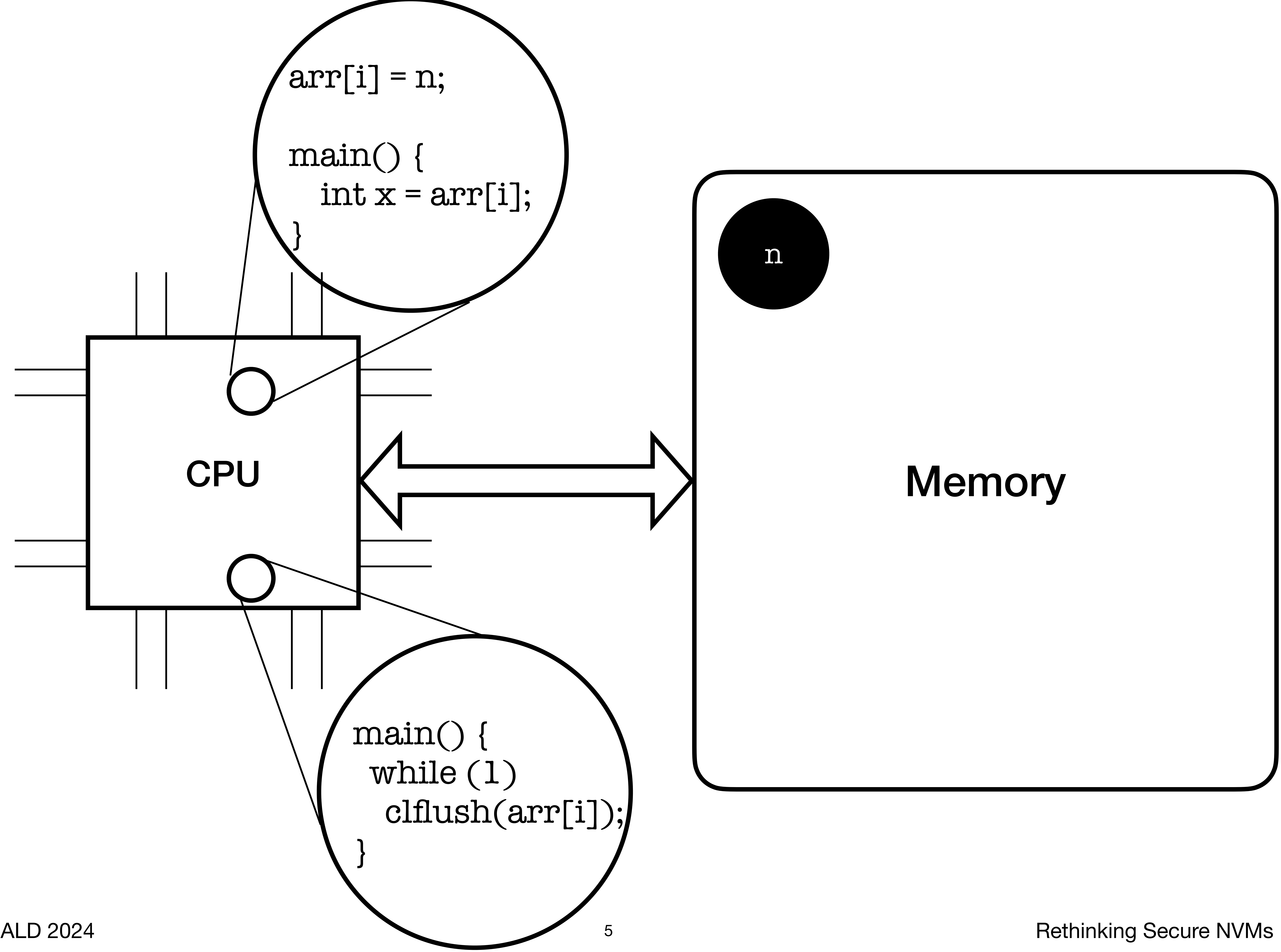
3. Application-Awareness

4. Changes due to CXL


n
CPU
Memory

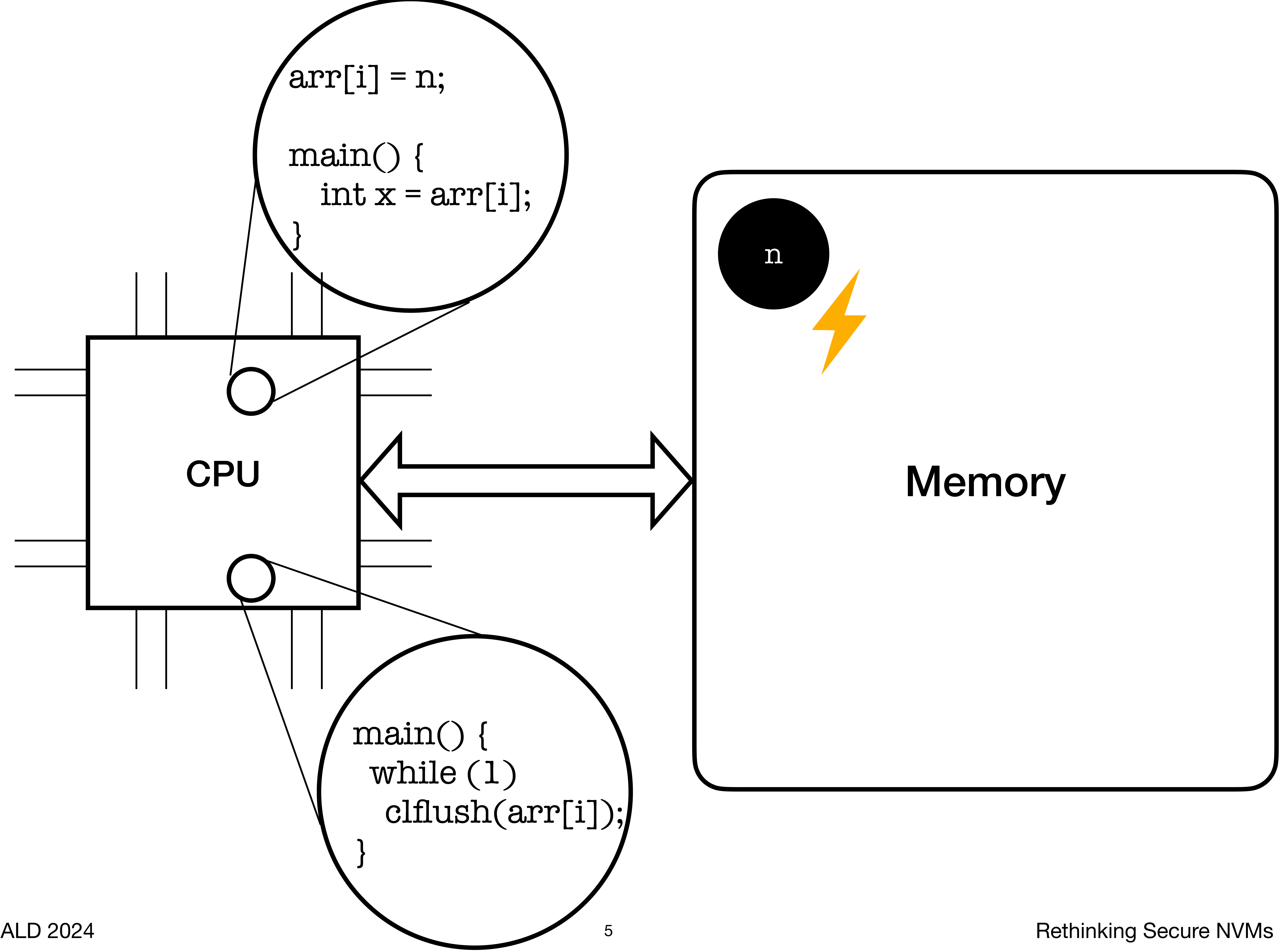


```
arr[i] = n;

main() {
   int x = arr[i];
}
```



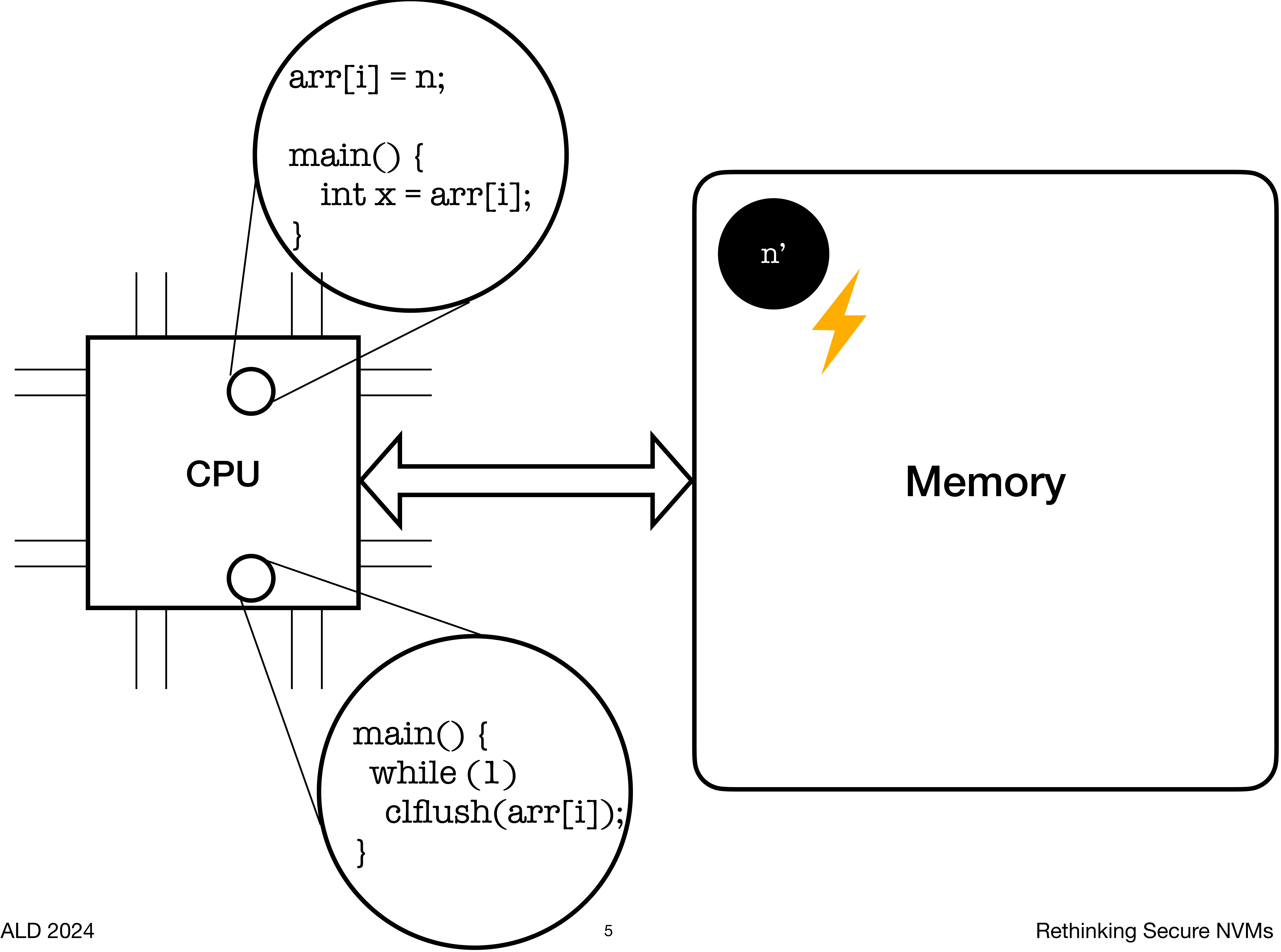
```
arr[i] = n;

main() {
   int x = arr[i];
}
```

```
main() {
 while (1)
   clflush(arr[i]);
}
```



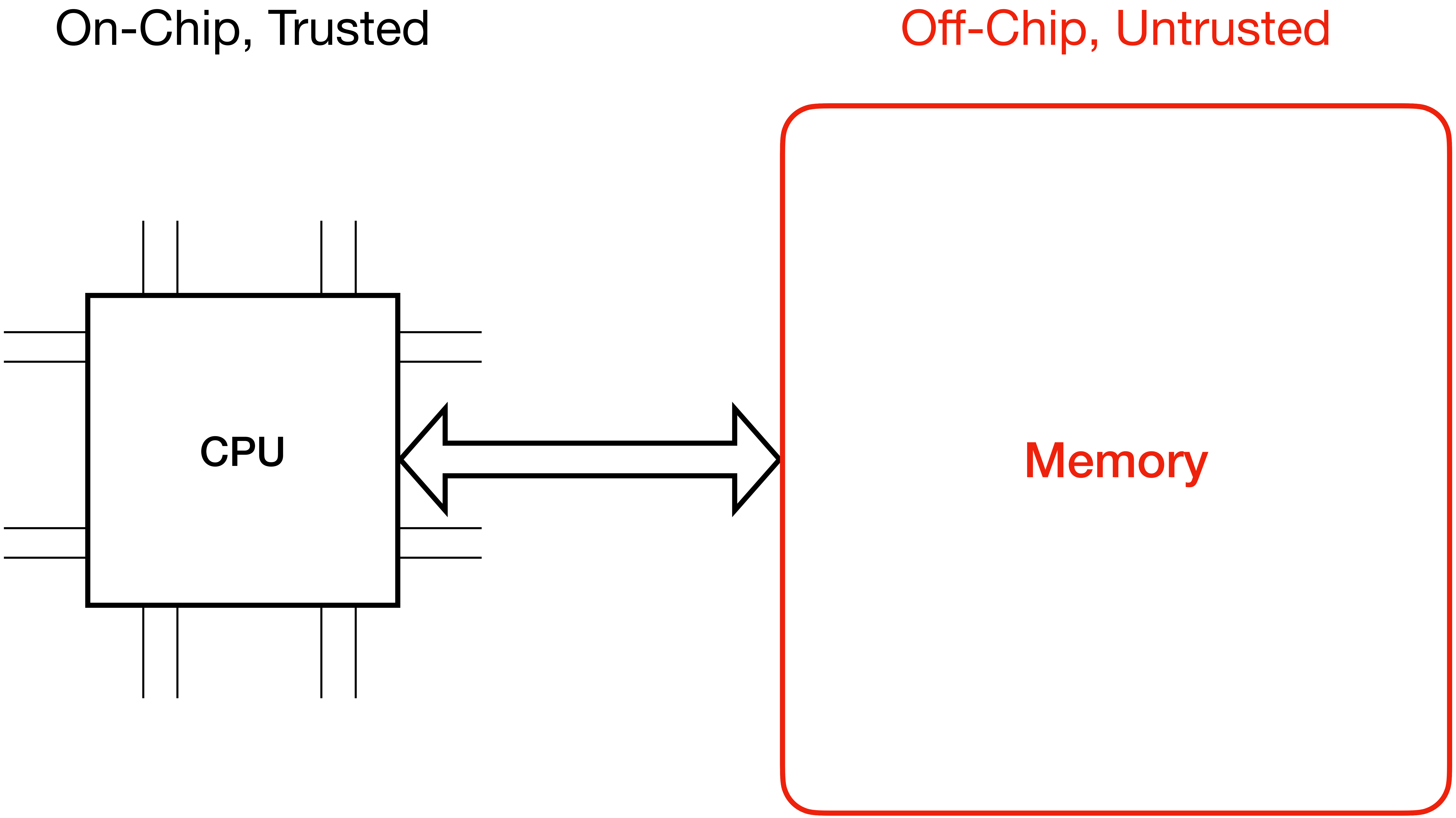
```
arr[i] = n;

main() {
   int x = arr[i];
}
```

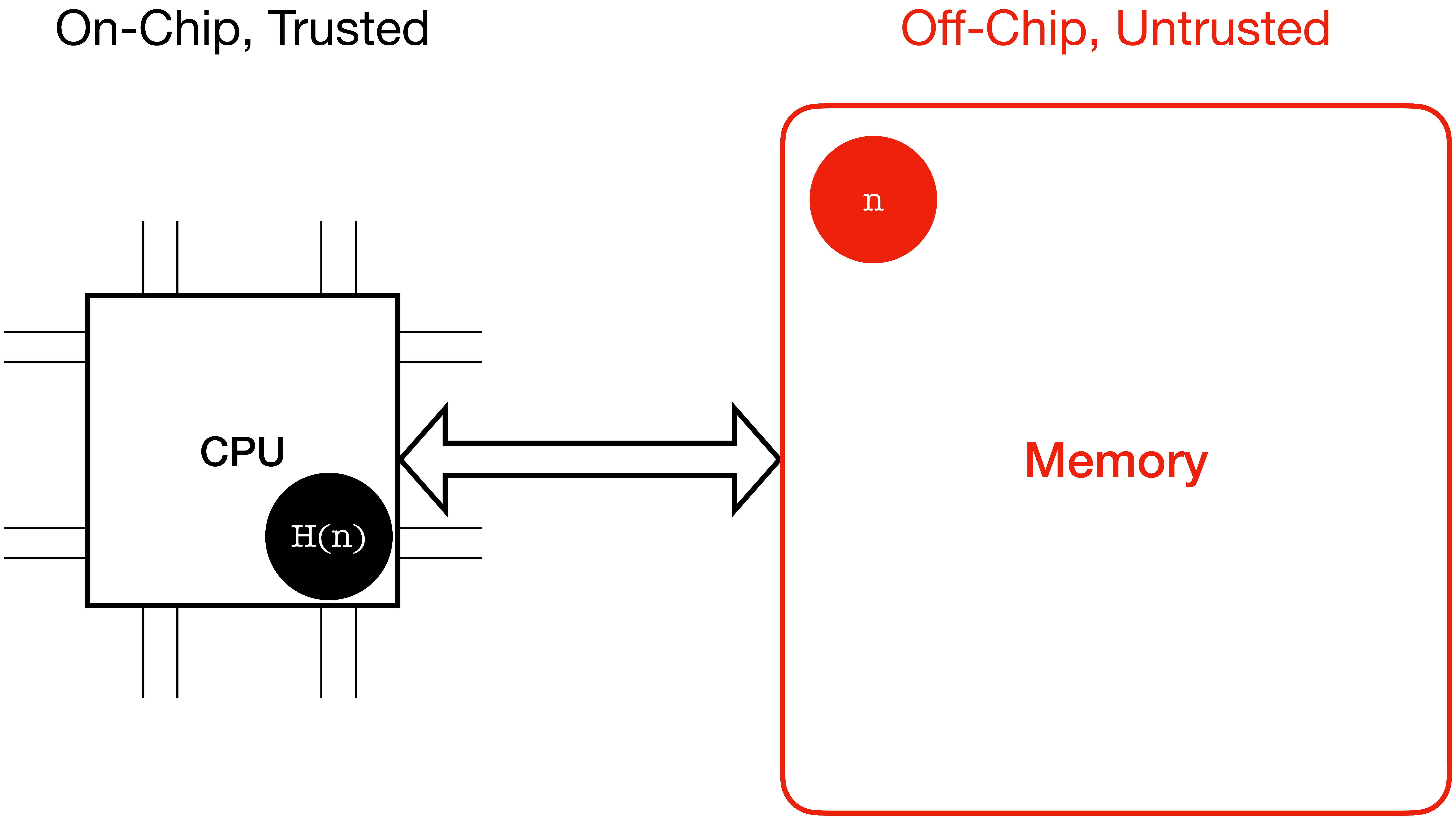
```
main() {
 while (1)
   clflush(arr[i]);
}
```



```
arr[i] = n;

main() {
   int x = arr[i];
}
```

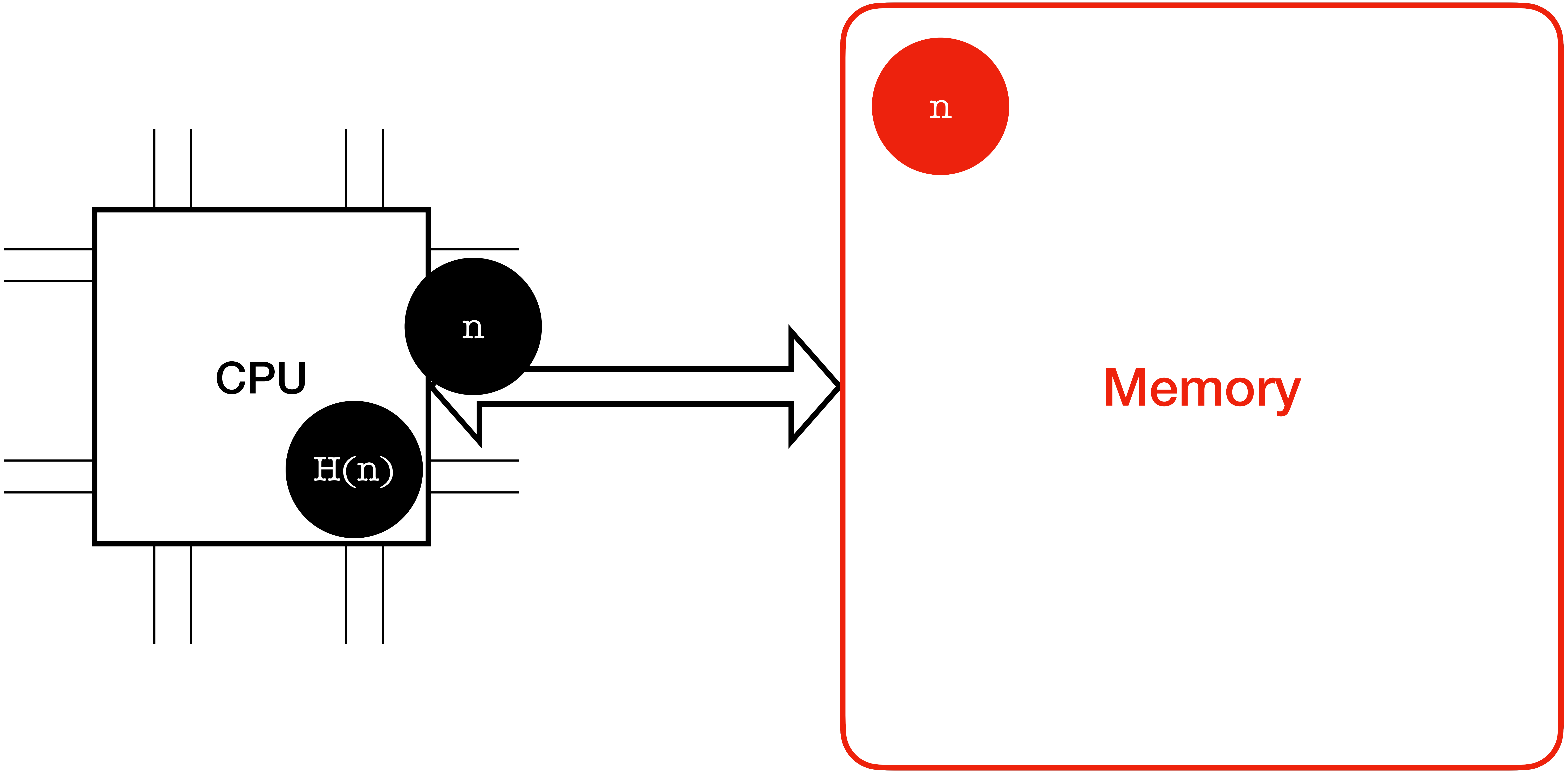
```
main() {
  while (1)
    clflush(arr[i]);
}
```

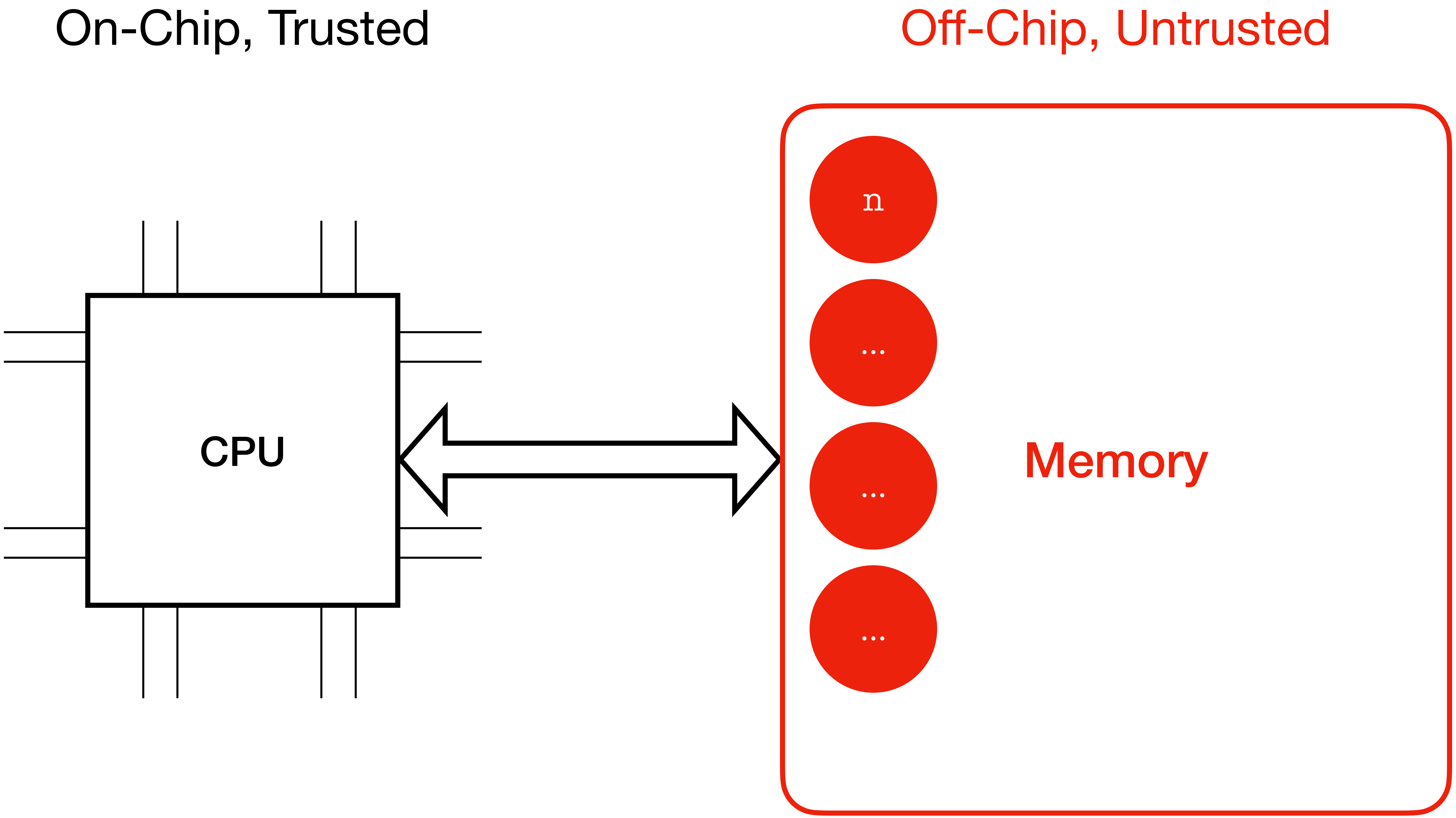

On-Chip, Trusted
Off-Chip, Untrusted
CPU
Memory


On-Chip, Trusted
Off-Chip, Untrusted
CPU
H(n)
n
Memory

On-Chip, Trusted

Off-Chip, Untrusted





On-Chip, Trusted

Off-Chip, Untrusted

CPU

Memory

n

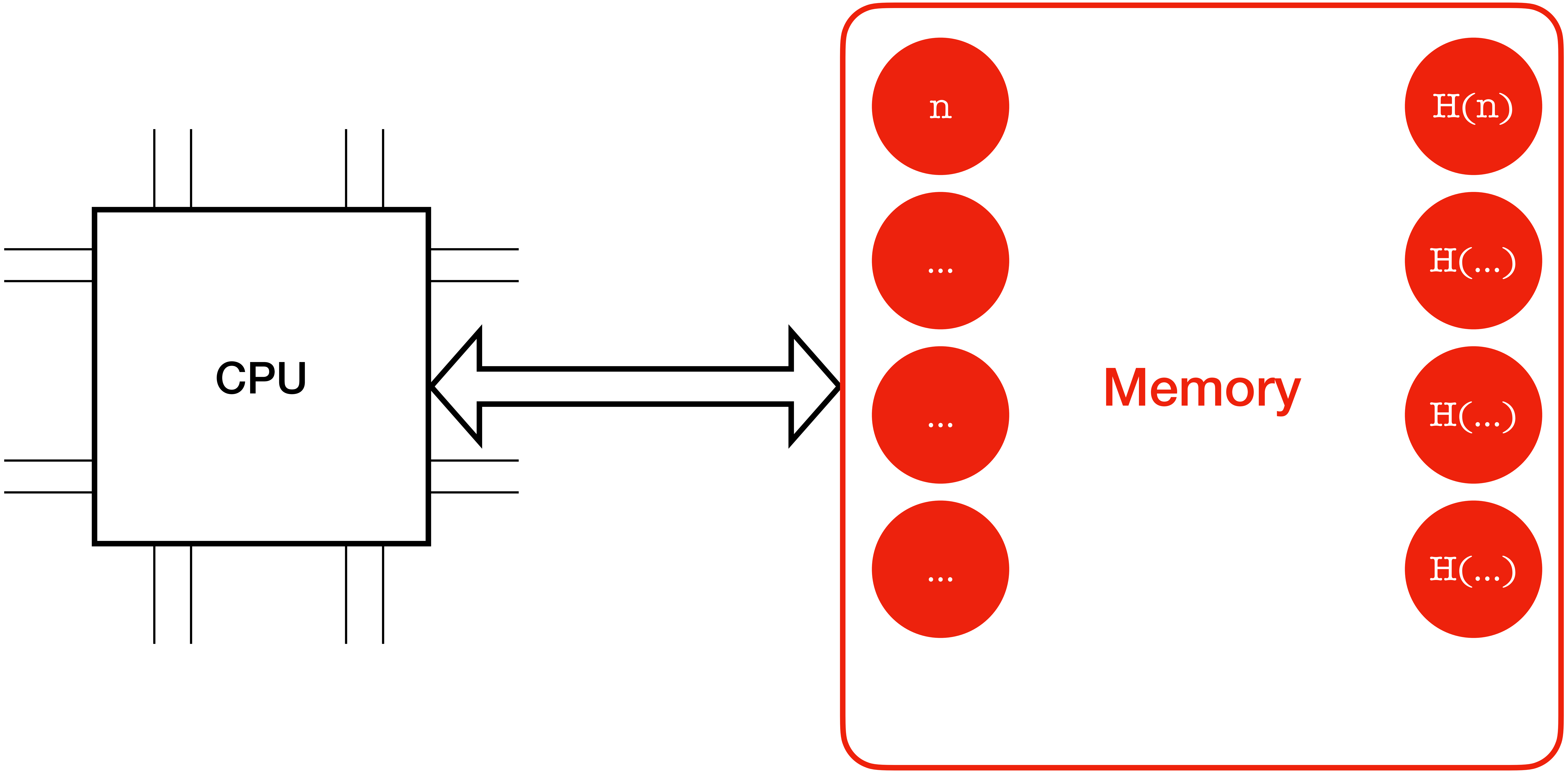
...

...

...

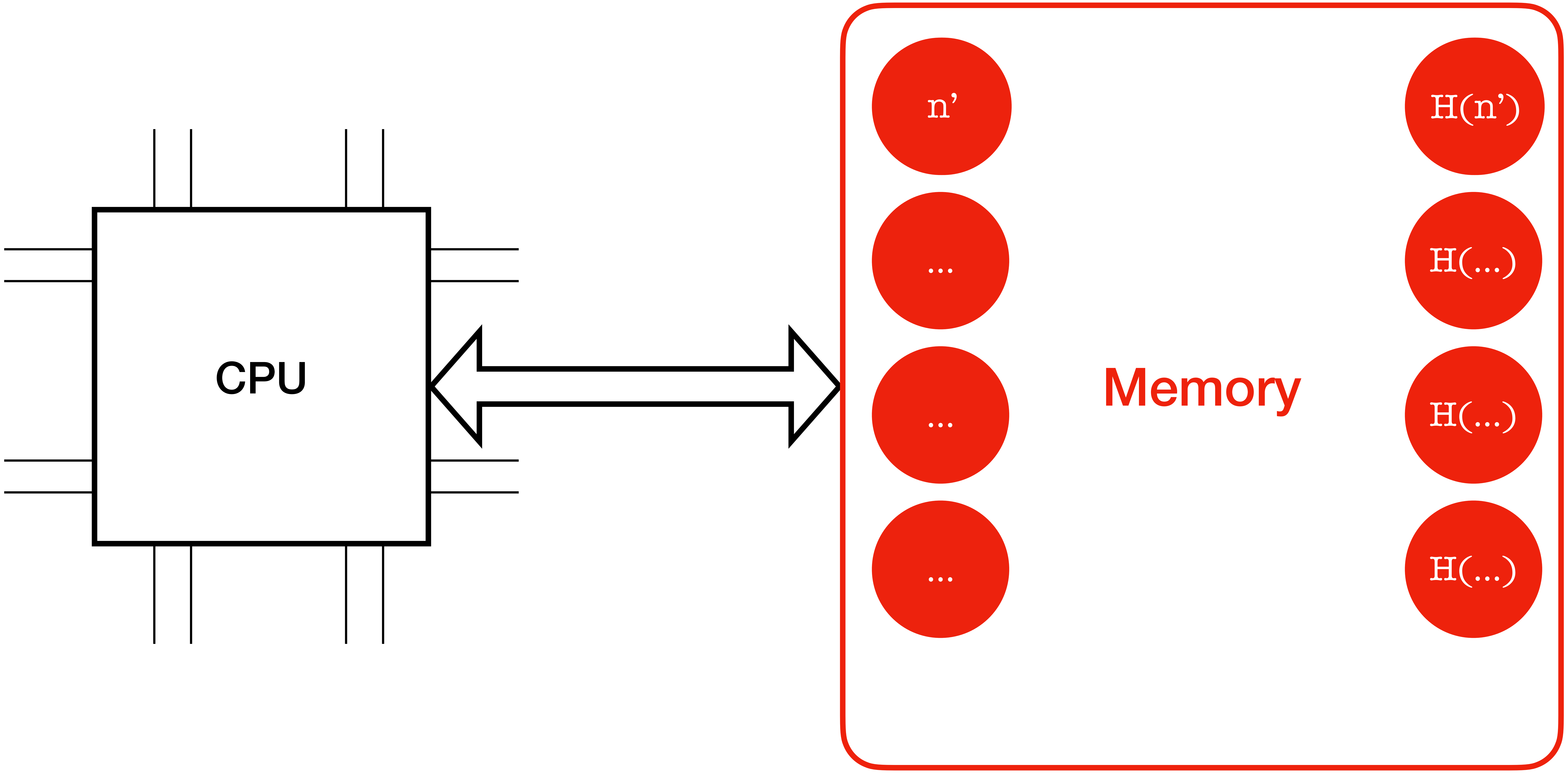
On-Chip, Trusted

Off-Chip, Untrusted

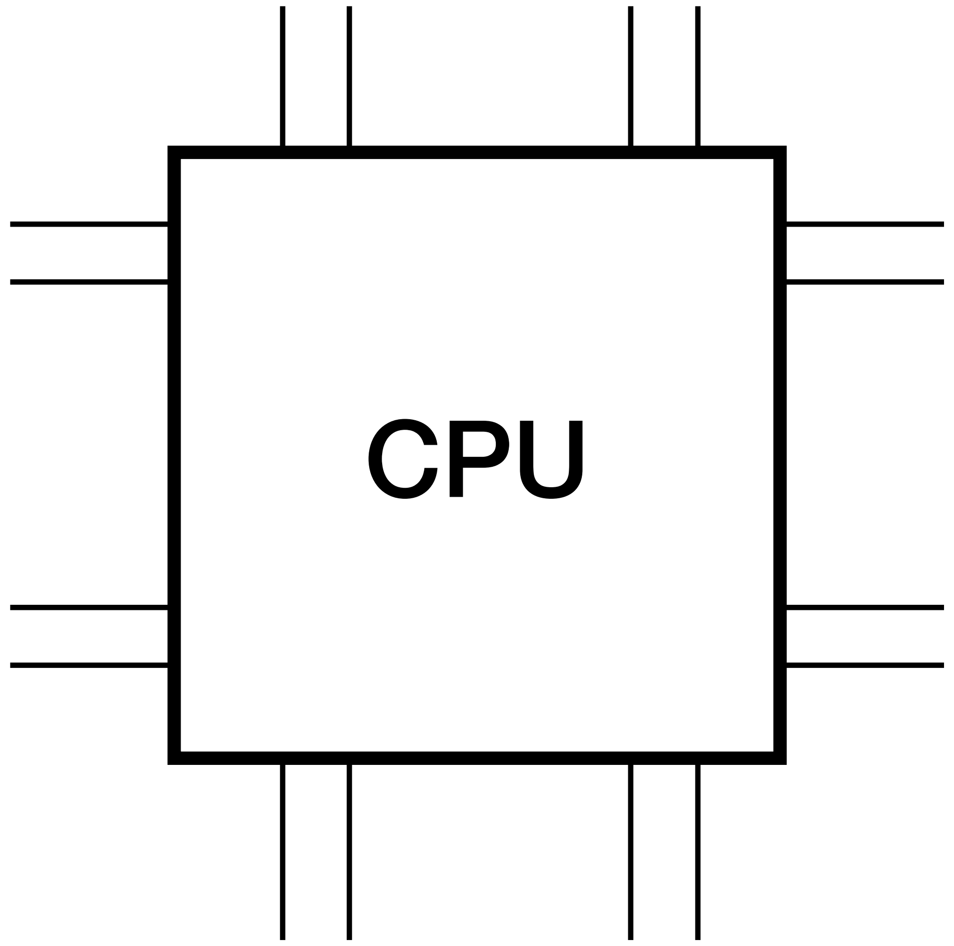


On-Chip, Trusted

Off-Chip, Untrusted



On-Chip, Trusted



Off-Chip, Untrusted

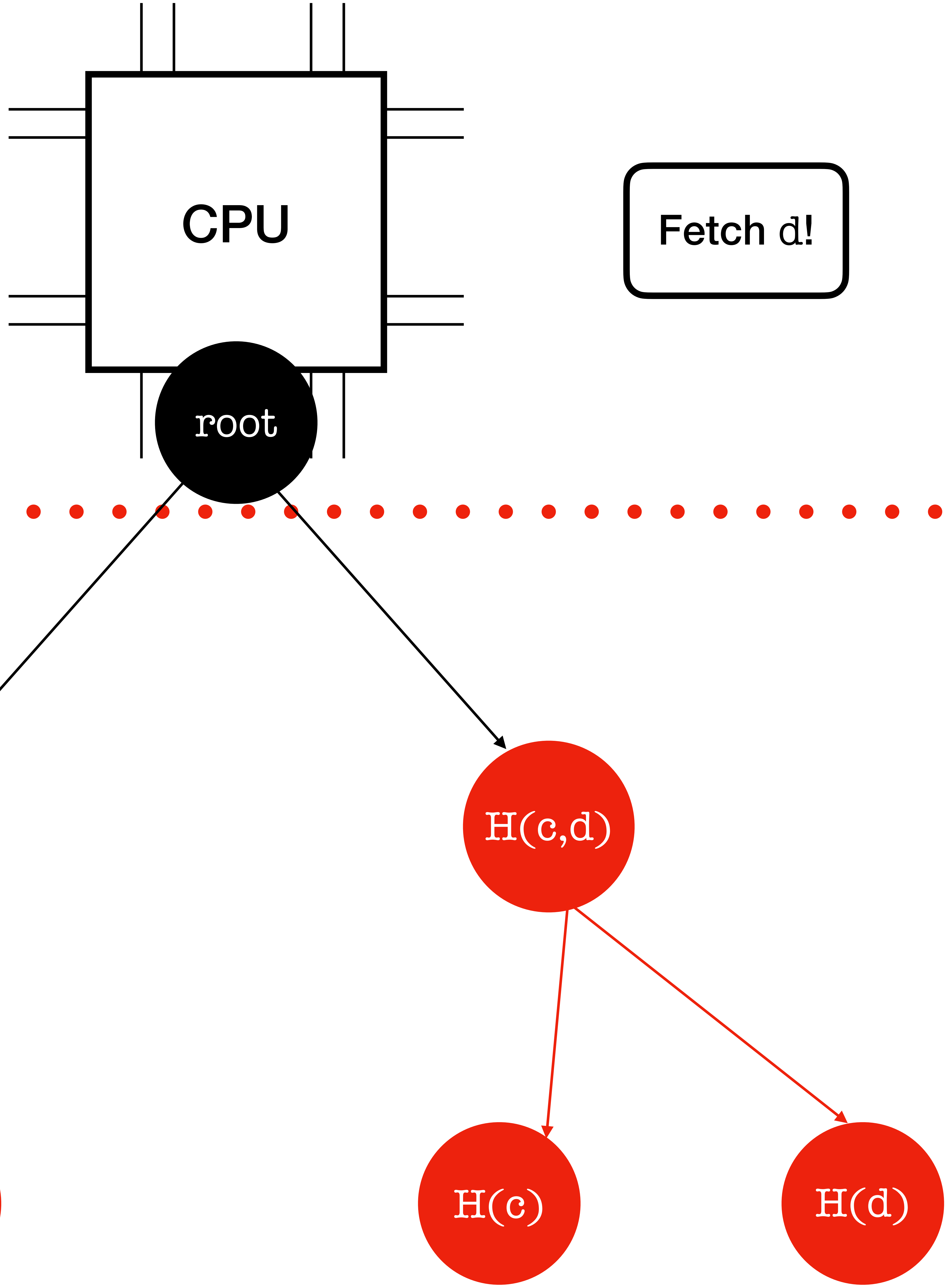
On-Chip, Trusted

Off-Chip, Untrusted

CPU
root
H(a,b)
H(c,d)
H(a)
H(b)
H(c)
H(d)

On-Chip, Trusted

Off-Chip, Untrusted



On-Chip, Trusted
CPU
Fetch d!
root
Off-Chip, Untrusted
H(a,b)
H(c,d)
H(c,d)
H(a)
H(b)
H(c)
H(d)
H(d)

On-Chip, Trusted

CPU

Metadata Cache

root

Off-Chip, Untrusted

H(a,b)

H(c,d)

H(a)

H(b)

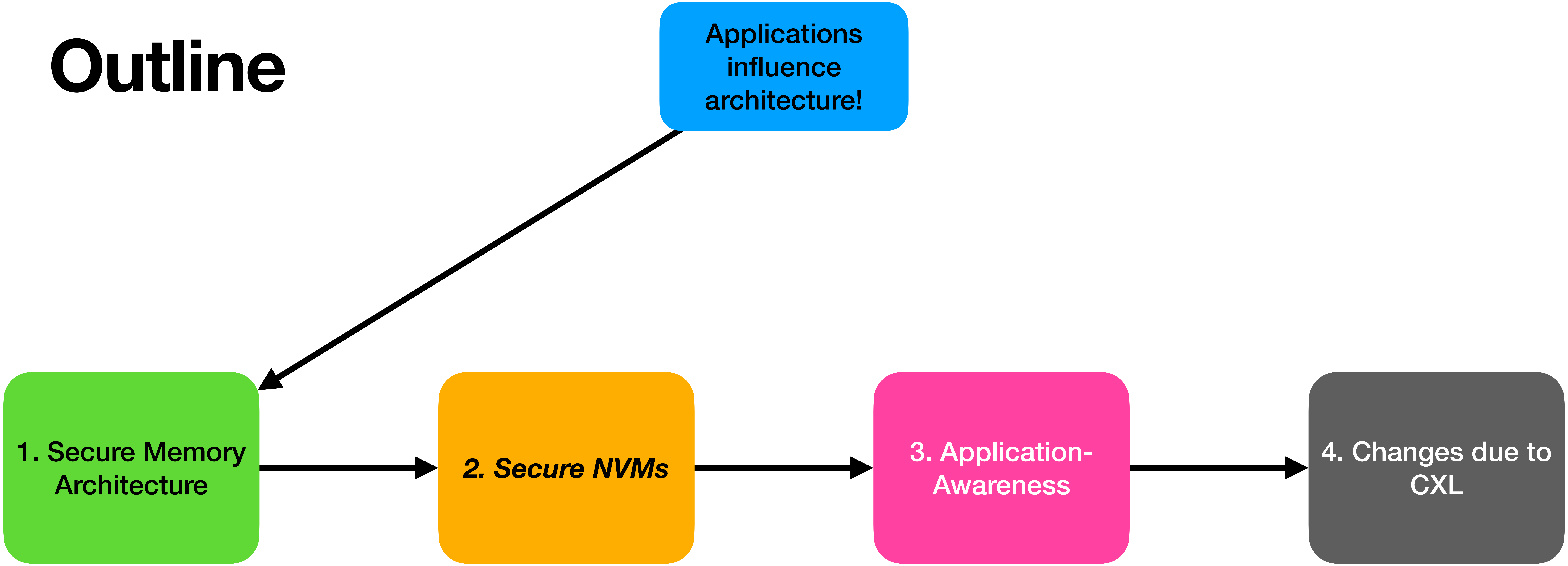
H(c)

H(d)

On-Chip, Trusted
CPU
Metadata Cache
H(c,d)
root
Off-Chip, Untrusted
H(a,b)
H(c,d)
H(a)
H(b)
H(c)
H(d)

On-Chip, Trusted
CPU
Metadata Cache
H(c,d)
root
Off-Chip, Untrusted
H(a,b)
H(c,d)
H(a)
H(b)
H(c)
H(d)

Outline



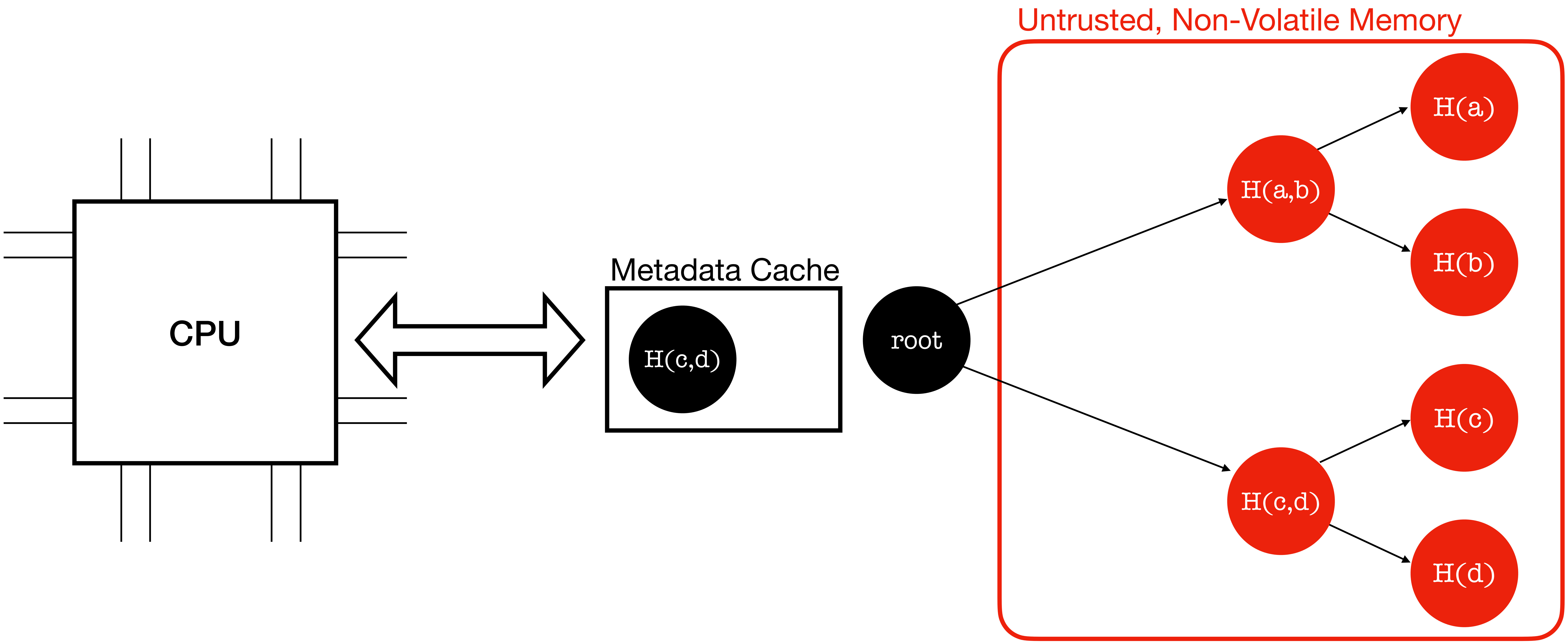
Applications influence architecture!

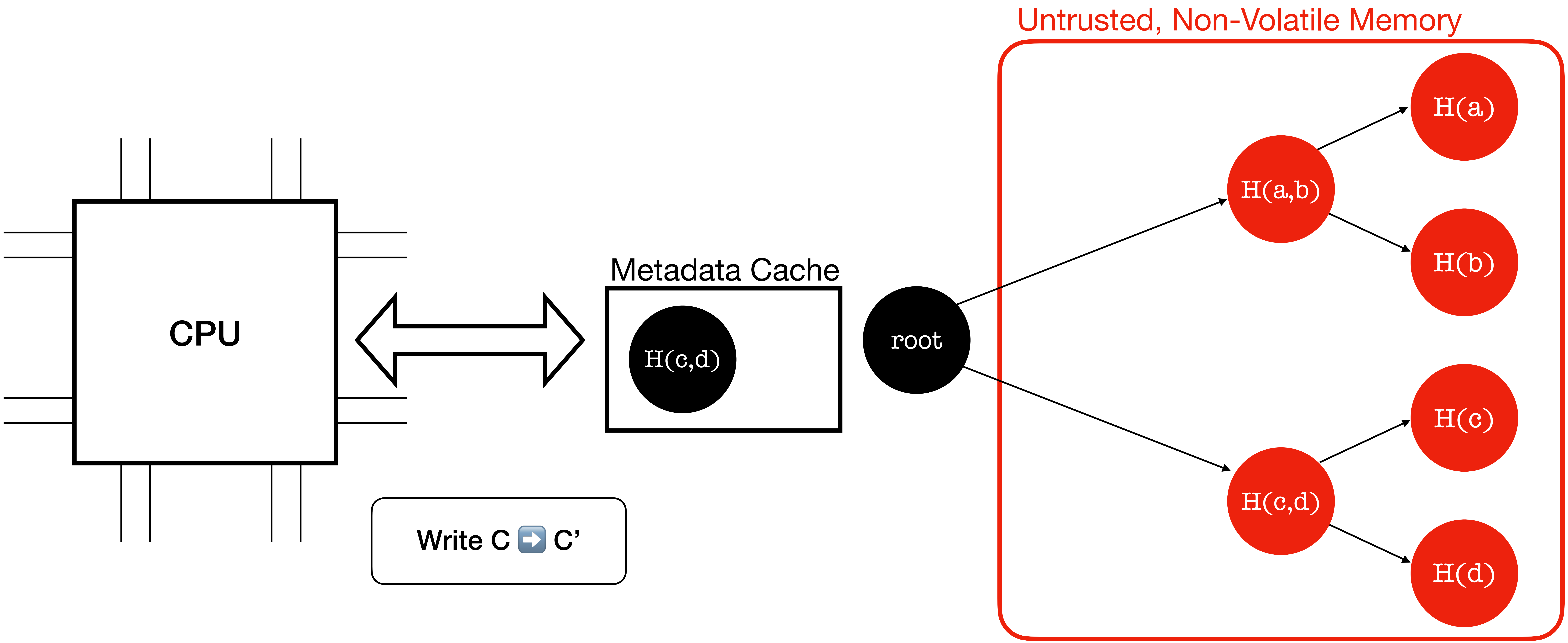
1. Secure Memory Architecture

2. Secure NVMs

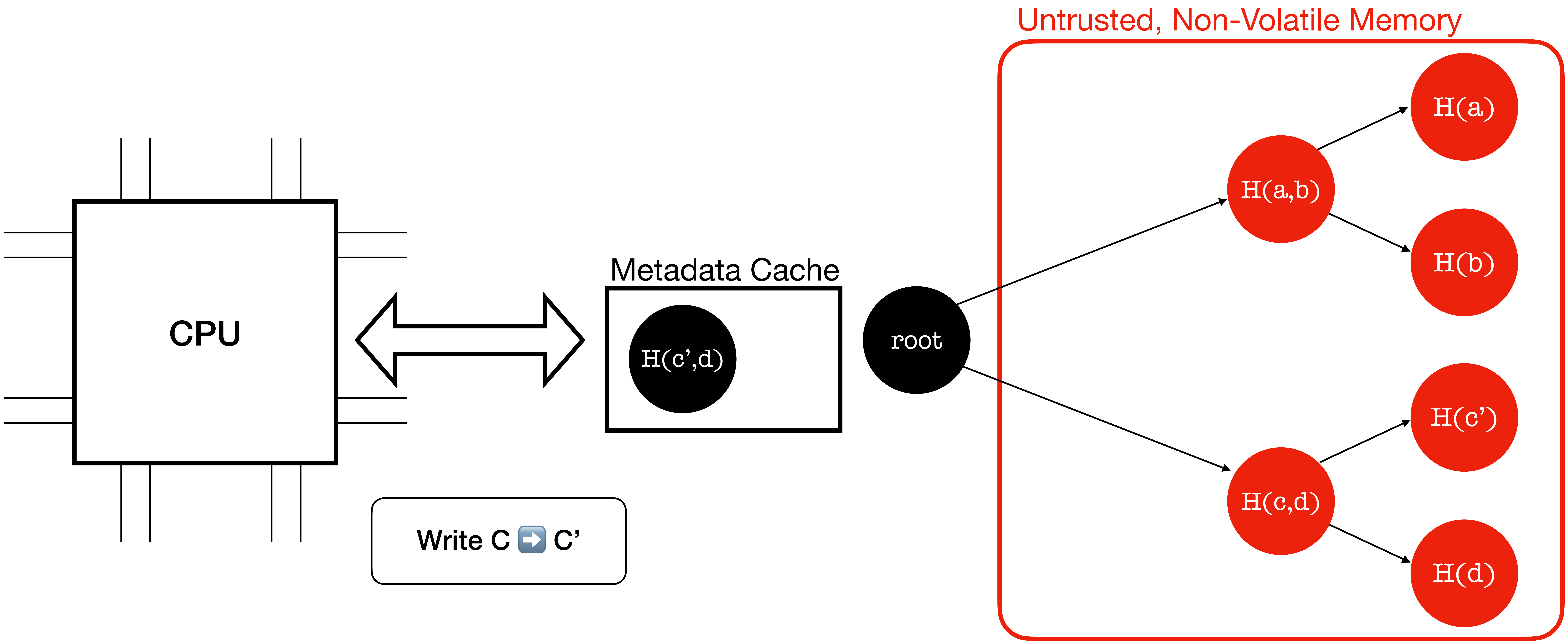
3. Application-Awareness

4. Changes due to CXL

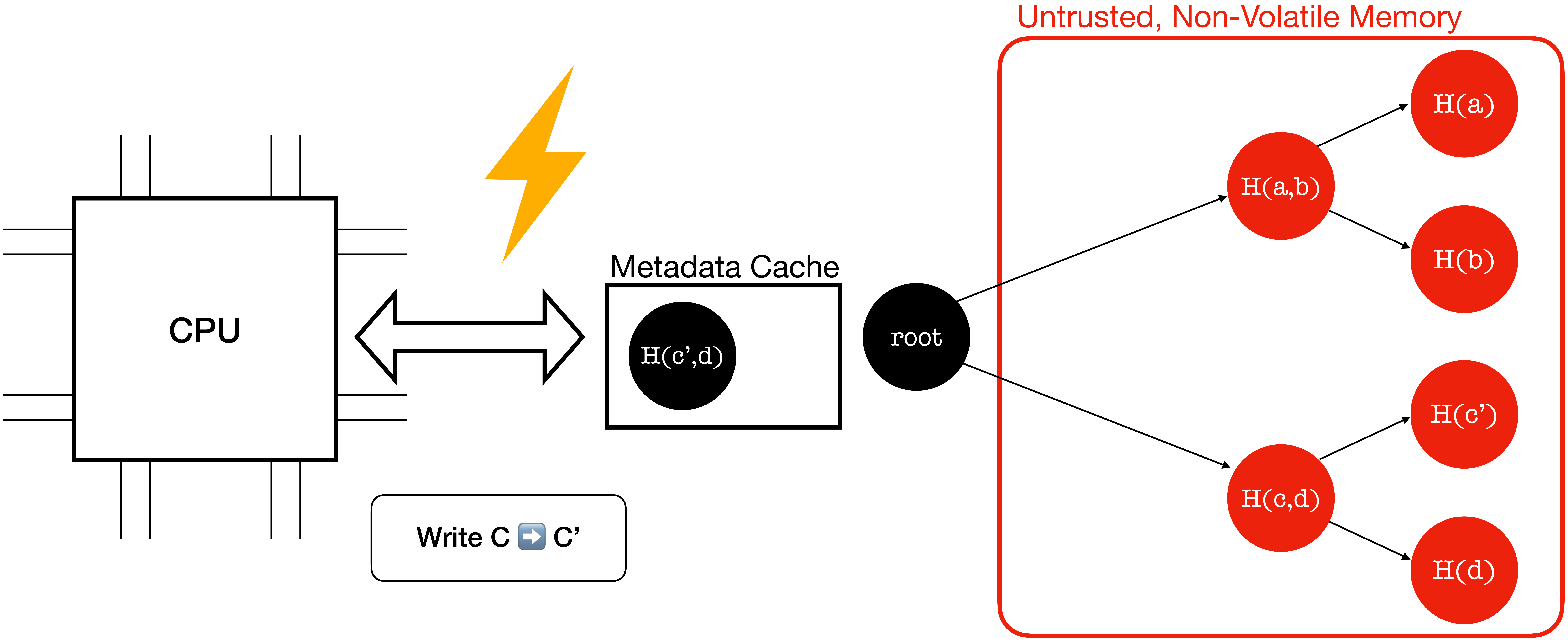

Untrusted, Non-Volatile Memory
CPU
Metadata Cache
H(c,d)
root
H(a,b)
H(c,d)
H(a)
H(b)
H(c)
H(d)


Untrusted, Non-Volatile Memory
CPU
Metadata Cache
H(c,d)
root
H(a,b)
H(a)
H(b)
H(c,d)
H(c)
H(d)

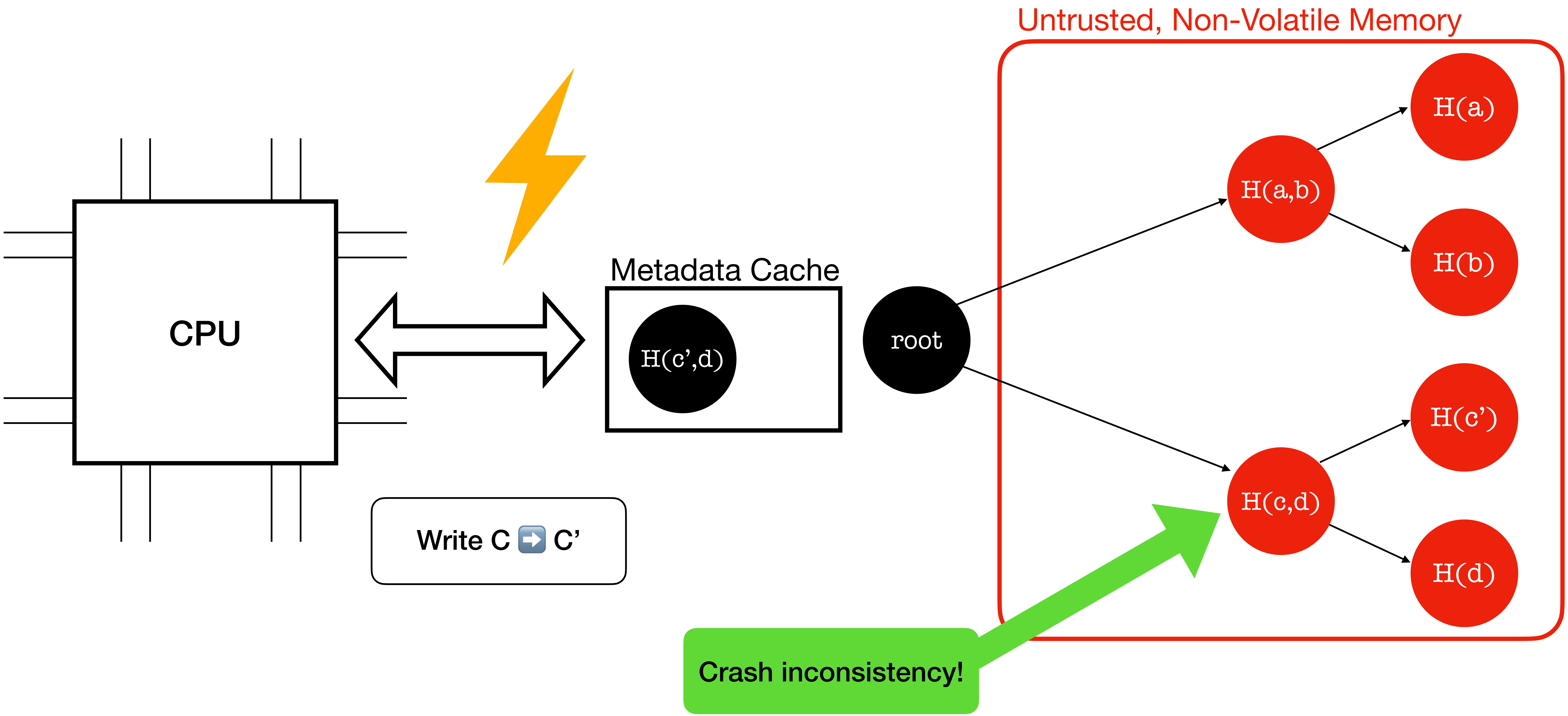
Write C ➡️ C’


Untrusted, Non-Volatile Memory
CPU
Metadata Cache
H(c’,d)
root
H(a,b)
H(a)
H(b)
H(c,d)
H(c’)
H(d)

Write C ➡️ C’


Untrusted, Non-Volatile Memory
CPU
Metadata Cache
H(c',d)
root
H(a,b)
H(a)
H(b)
H(c,d)
H(c')
H(d)

Write C ➡️ C'



Untrusted, Non-Volatile Memory

CPU

Metadata Cache

H(c',d)

root

H(a,b)

H(a)

H(b)

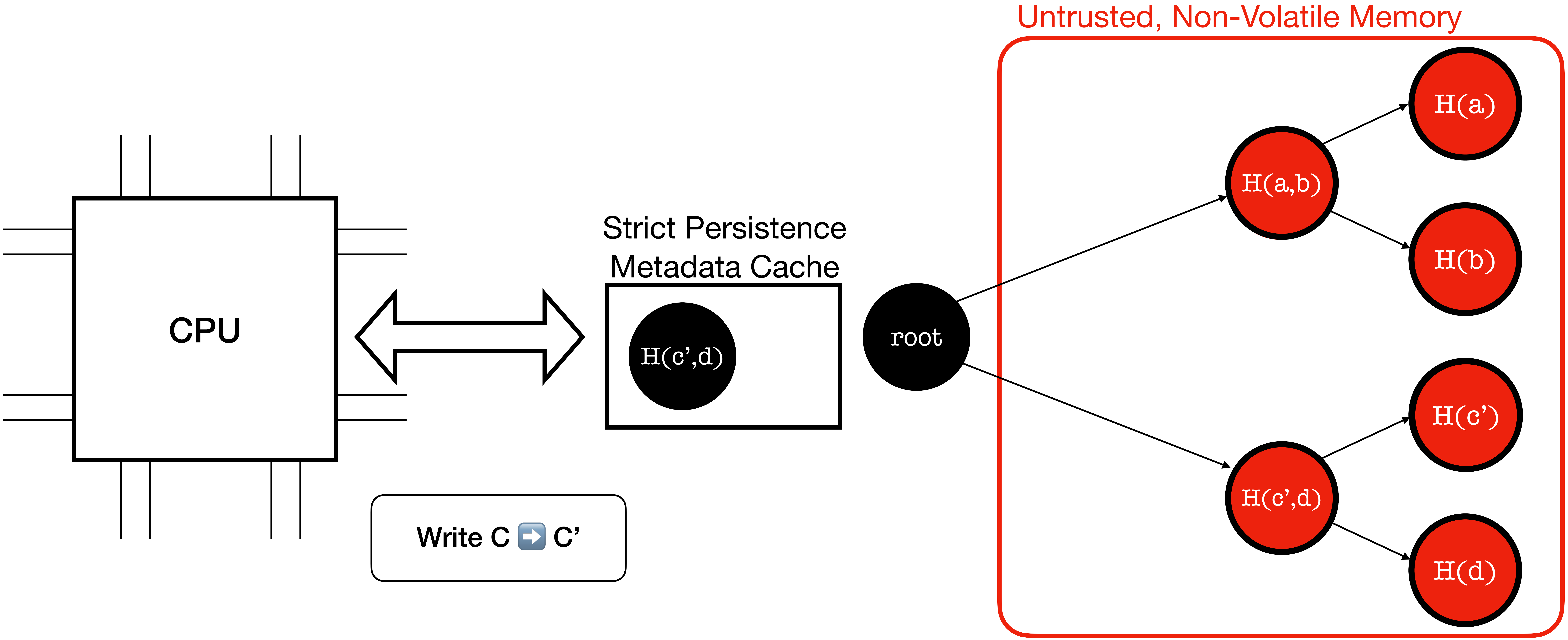
H(c,d)

H(c')

H(d)

Write C ➡️ C'

Crash inconsistency!

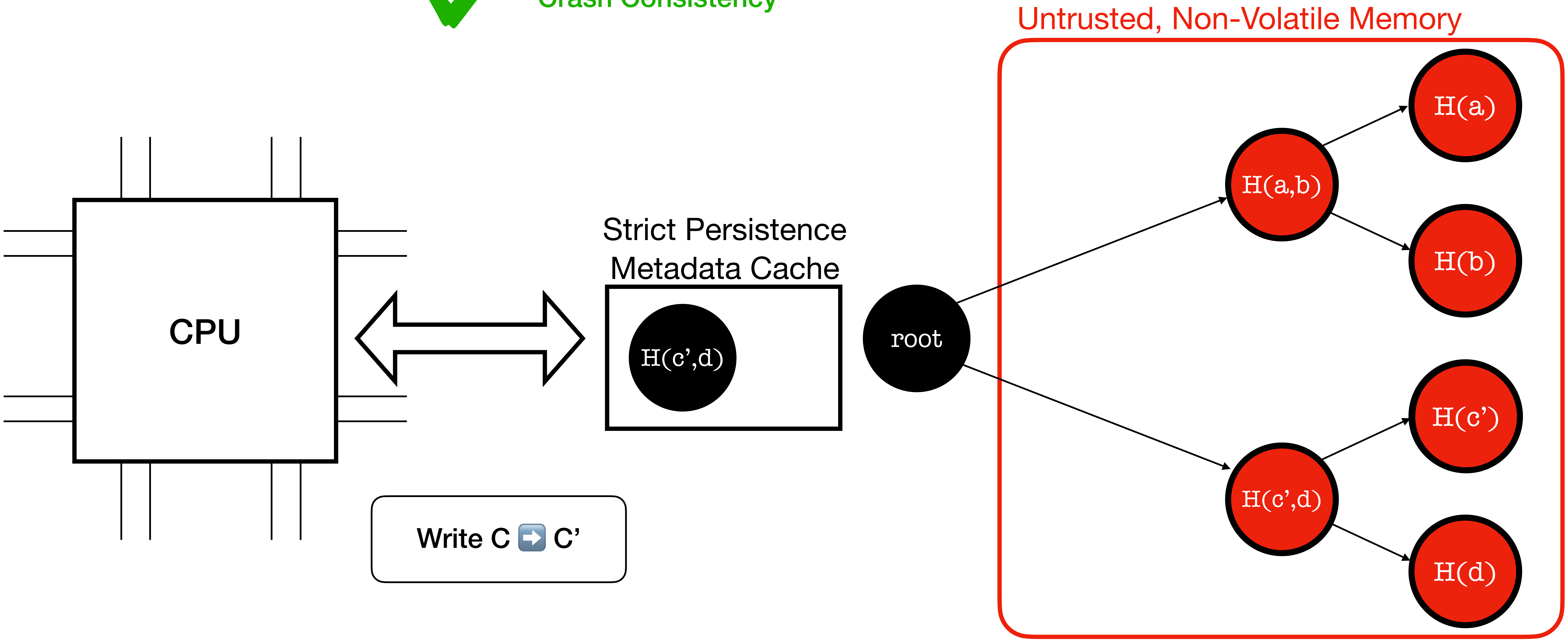


Untrusted, Non-Volatile Memory

Strict Persistence Metadata Cache

Write C ➡️ C'

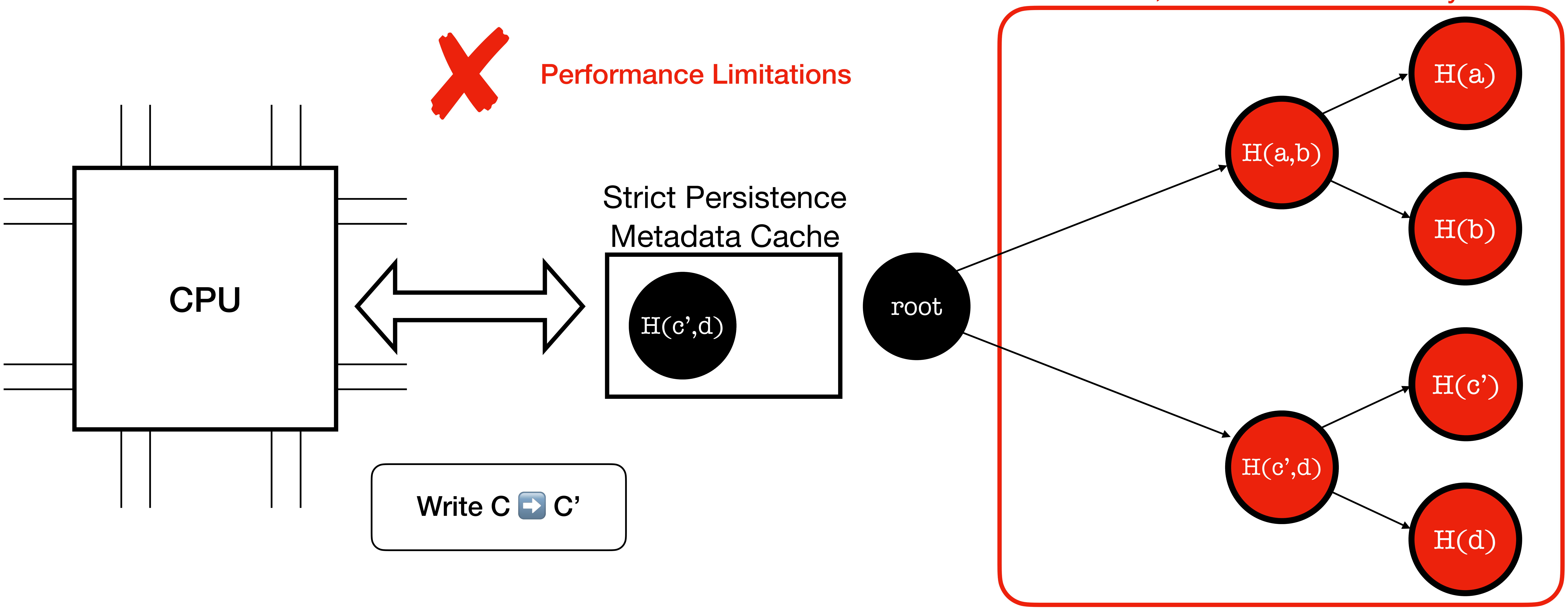
✓ Crash Consistency


Untrusted, Non-Volatile Memory
CPU
Strict Persistence Metadata Cache
H(c',d)
root
H(a,b)
H(a)
H(b)
H(c',d)
H(c')
H(d)

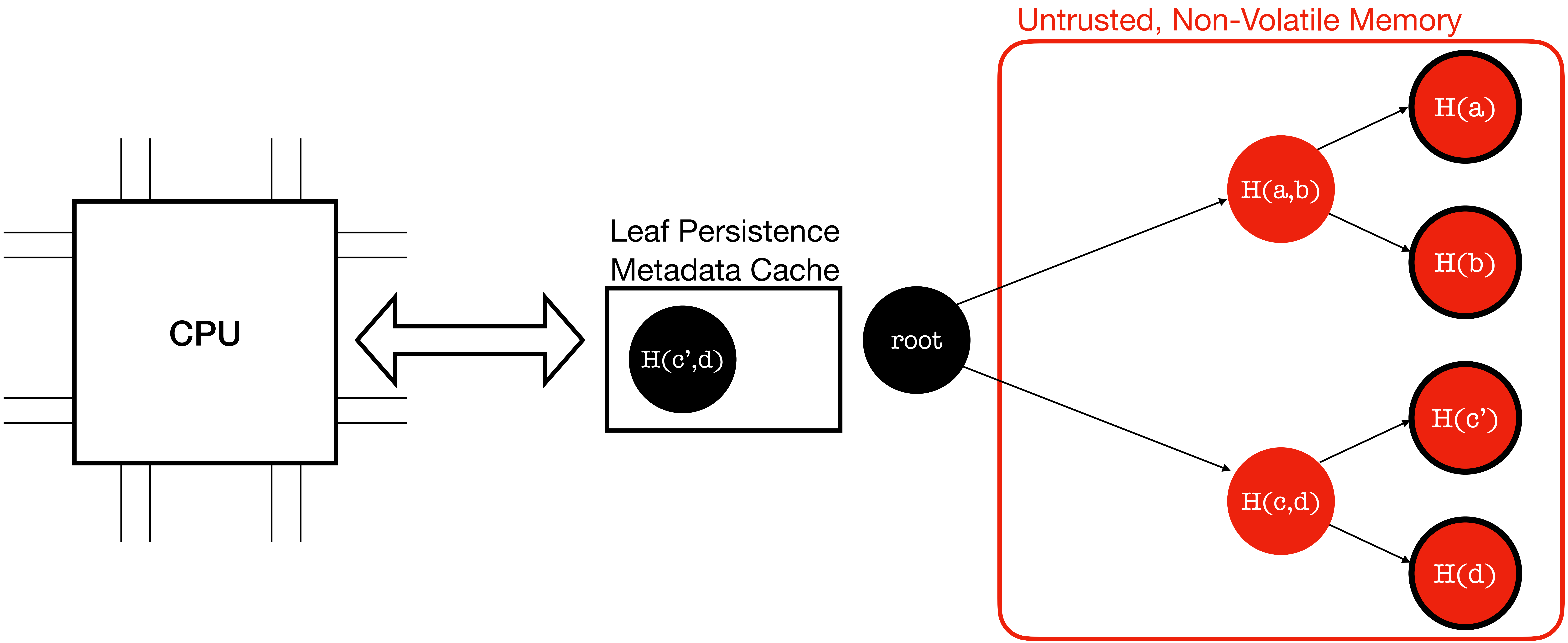
Write C ➡️ C'

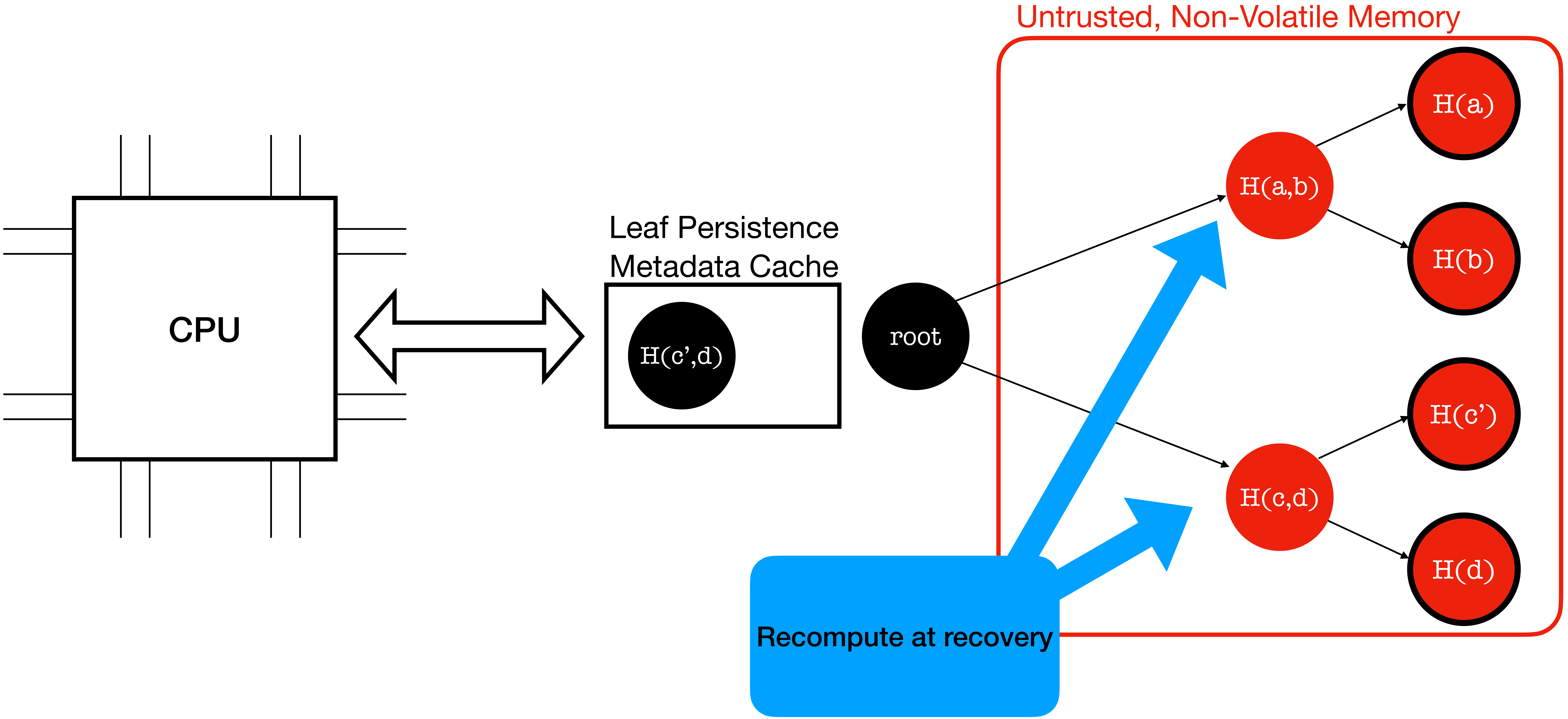
✓ Crash Consistency

✗ Performance Limitations


CPU
Strict Persistence Metadata Cache
H(c’,d)
root
Untrusted, Non-Volatile Memory
H(a,b)
H(a)
H(b)
H(c’,d)
H(c’)
H(d)

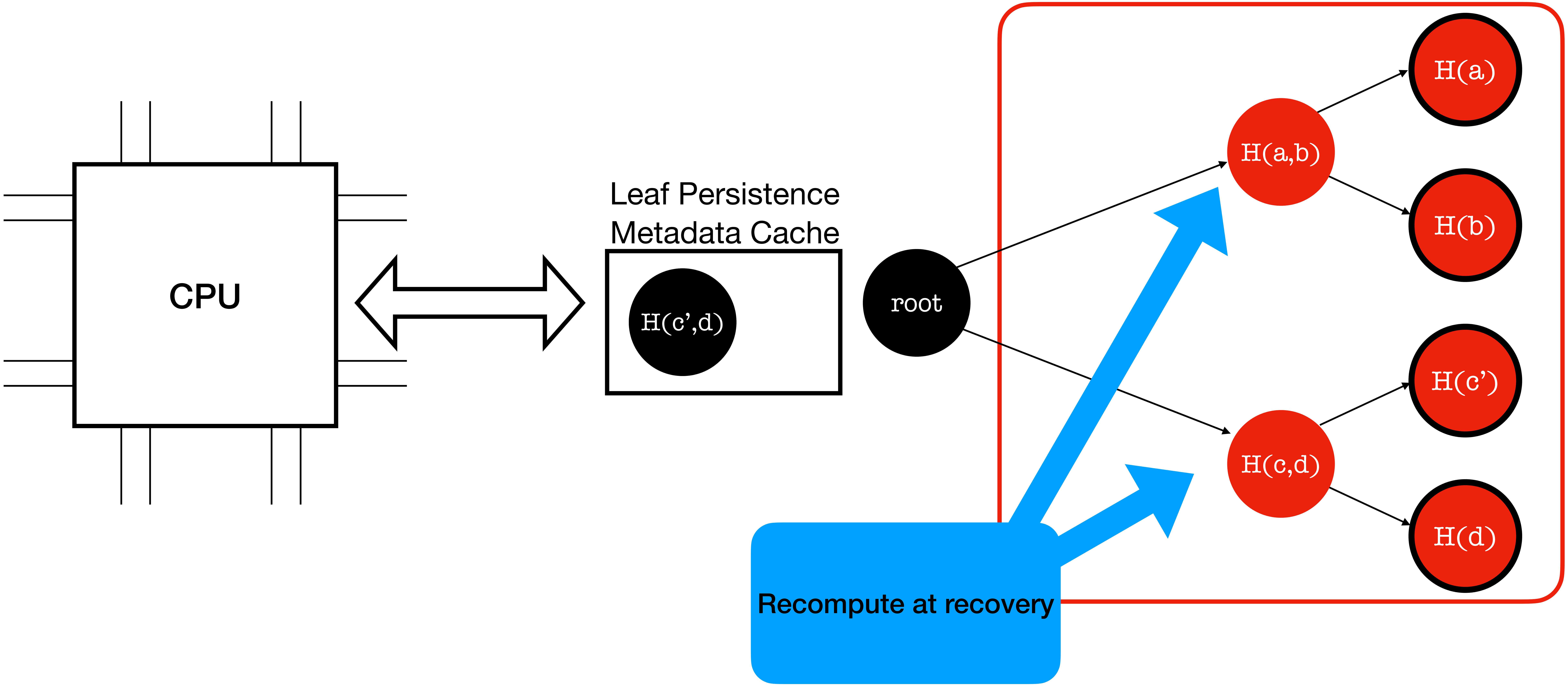
Write C ➡️ C’

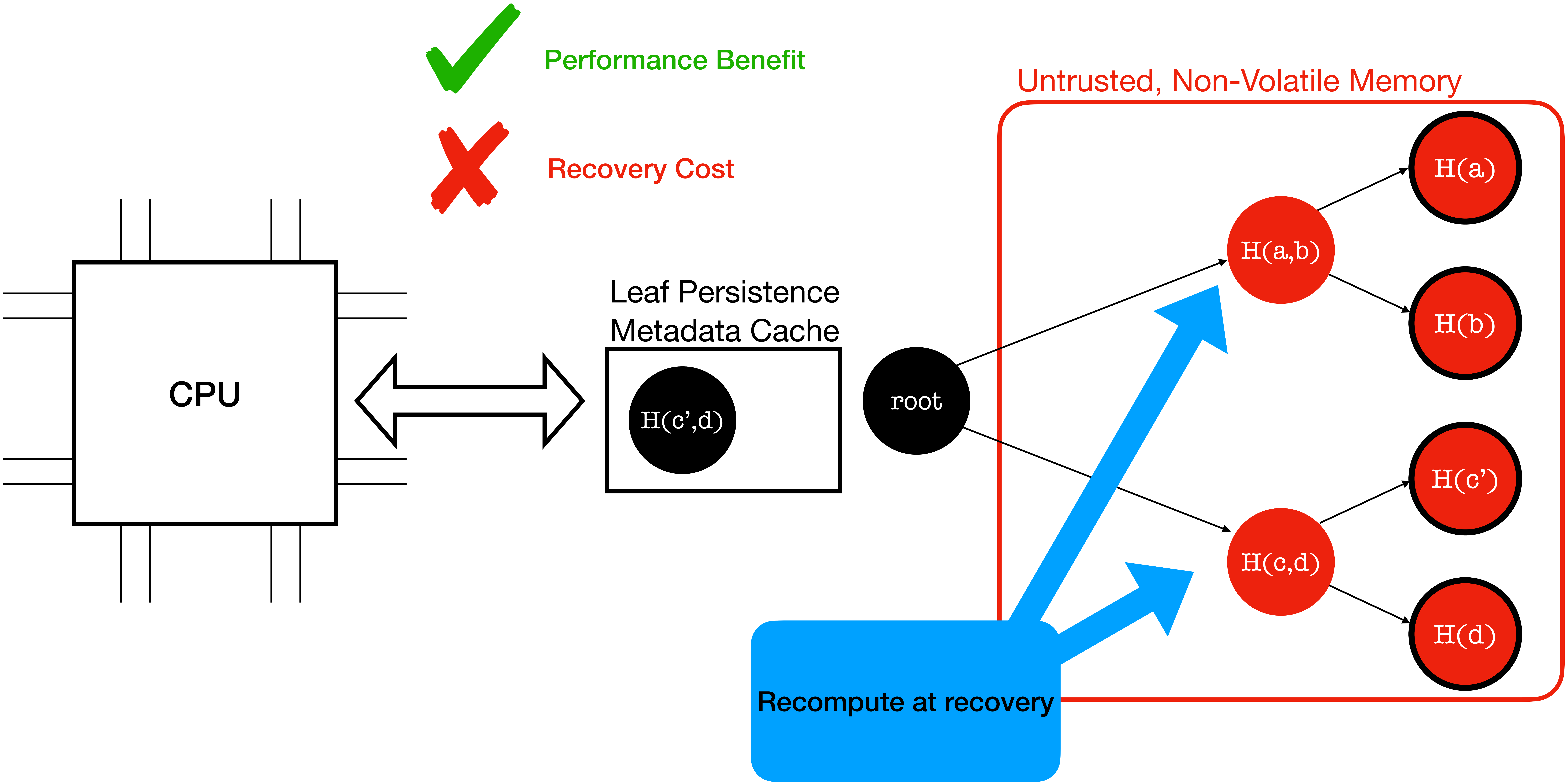

Untrusted, Non-Volatile Memory
CPU
Leaf Persistence Metadata Cache
H(c',d)
root
H(a,b)
H(c,d)
H(a)
H(b)
H(c')
H(d)



Recompute at recovery

✓ Performance Benefit





✓ Performance Benefit

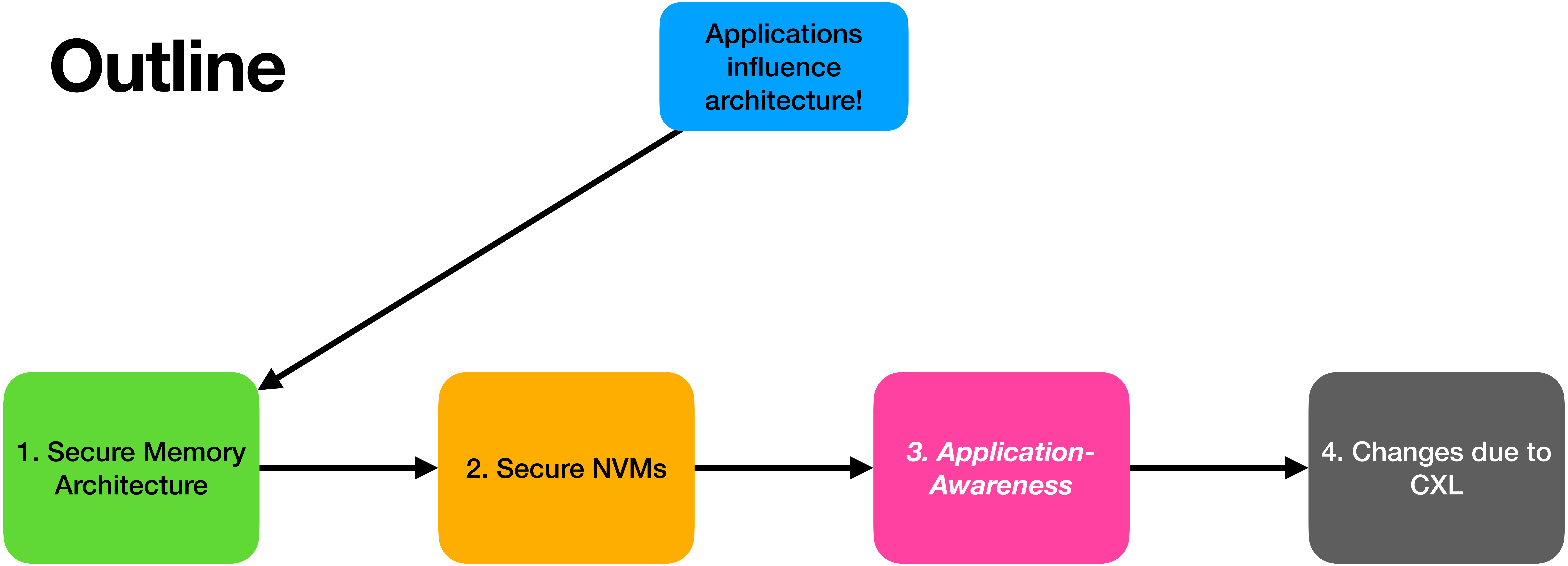
✗ Recovery Cost

Untrusted, Non-Volatile Memory

Leaf Persistence Metadata Cache

Recompute at recovery

Outline



Applications influence architecture!

1. Secure Memory Architecture

2. Secure NVMs

3. Application-Awareness

4. Changes due to CXL

BROWN

Samuel Thomas

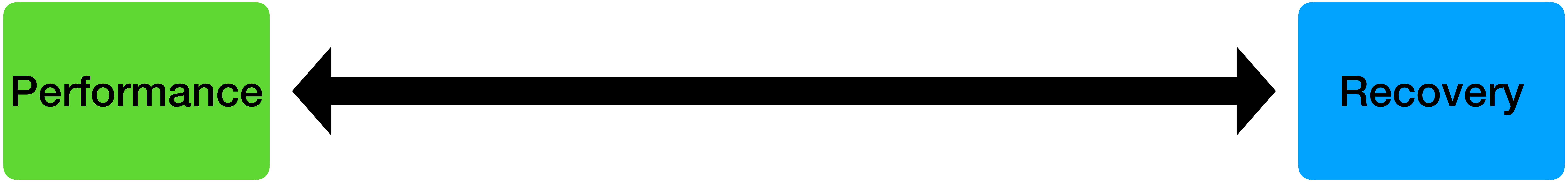
University of Colorado Boulder

Kidus Workneh, Joseph Izraelevitz, Tamara Lehman

Jac McCarty

R. Iris Bahar

QR code to the paper!

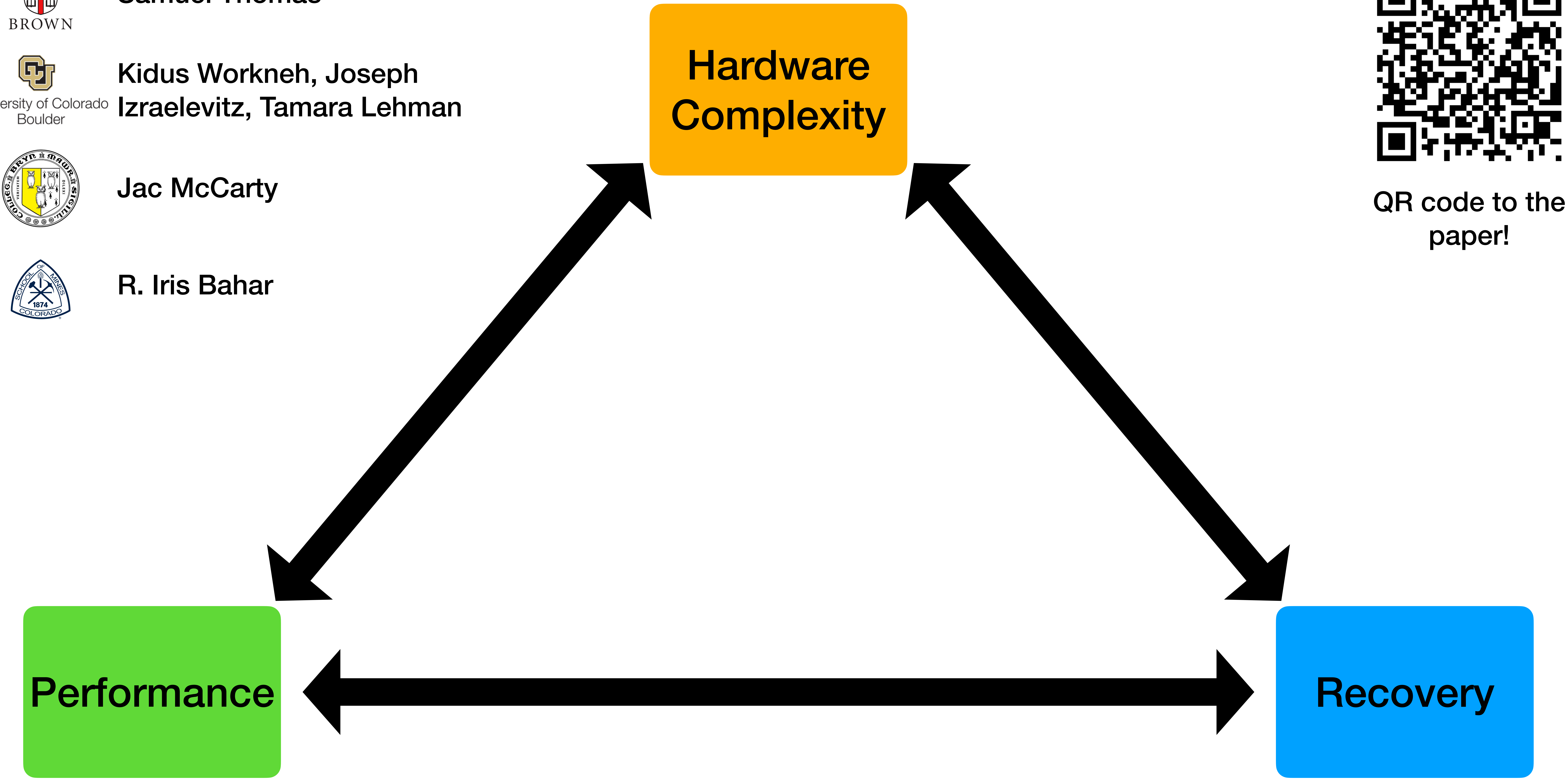


BROWN
Samuel Thomas

University of Colorado Boulder
Kidus Workneh, Joseph Izraelevitz, Tamara Lehman

Jac McCarty

R. Iris Bahar



Hardware Complexity

Performance

Recovery

QR code to the paper!

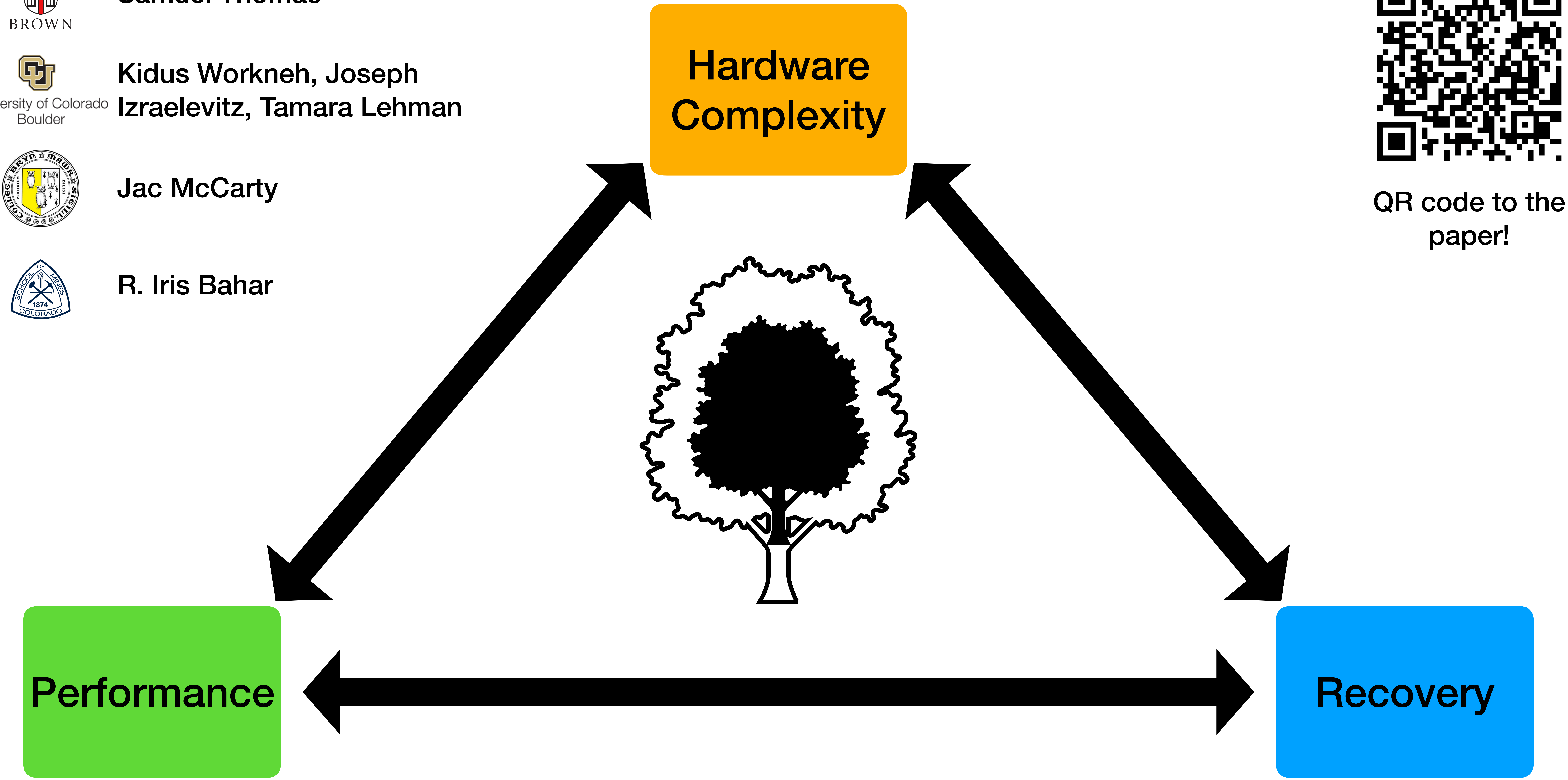
BROWN
Samuel Thomas

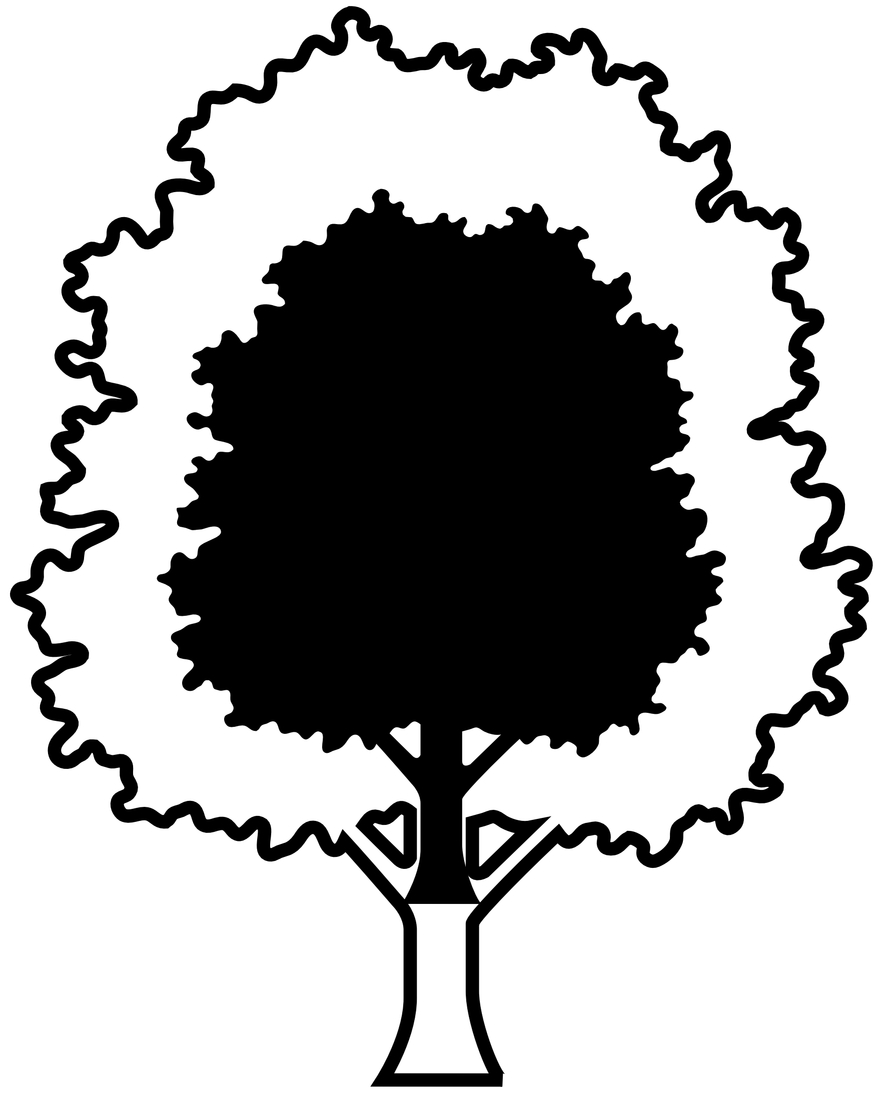
University of Colorado Boulder
Kidus Workneh, Joseph Izraelevitz, Tamara Lehman

Jac McCarty

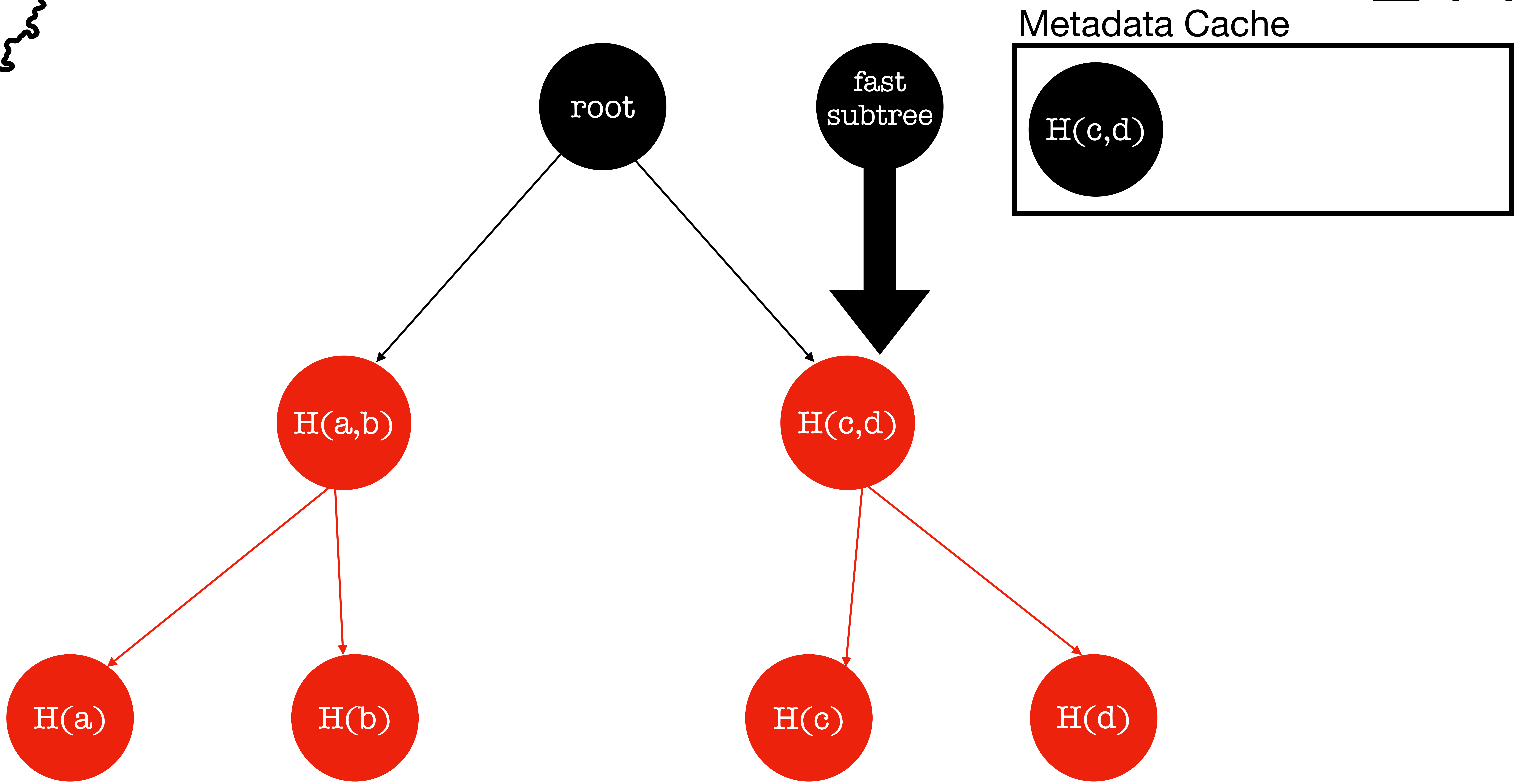
R. Iris Bahar

QR code to the paper!

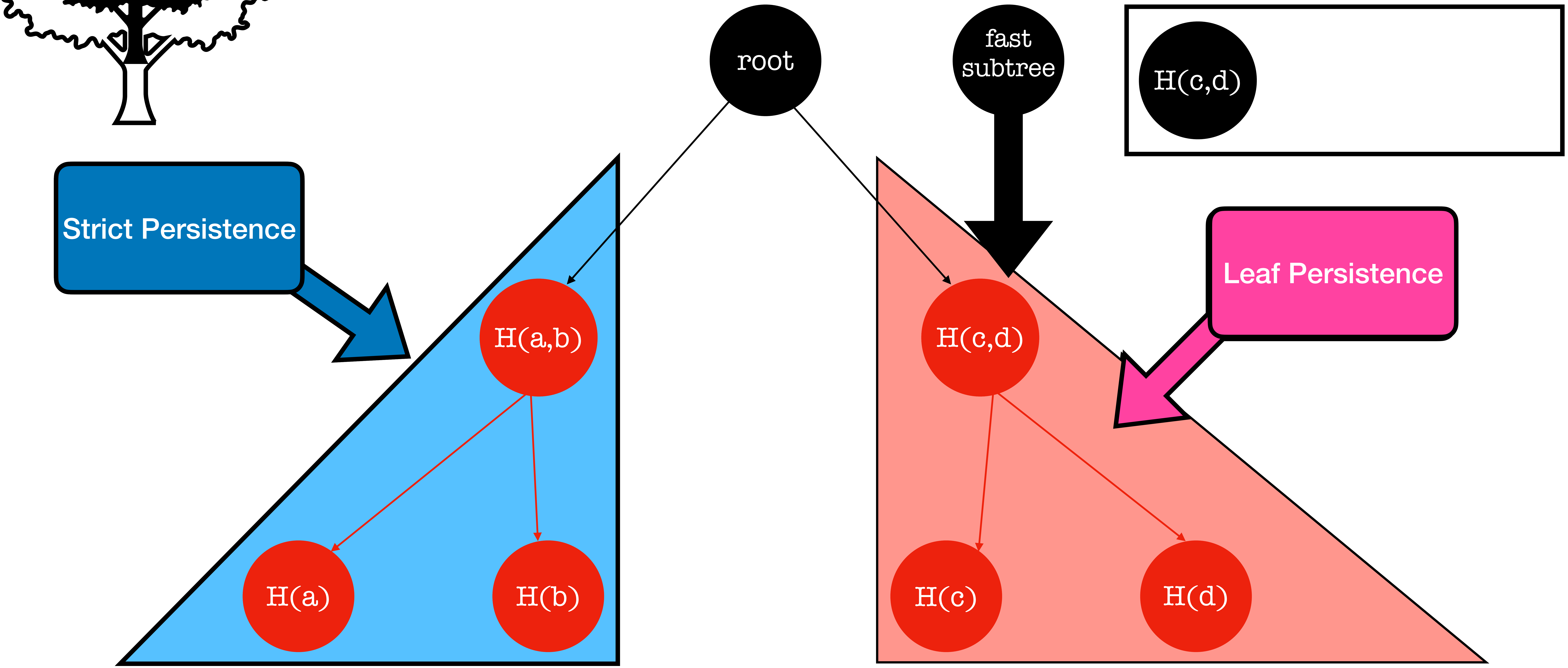


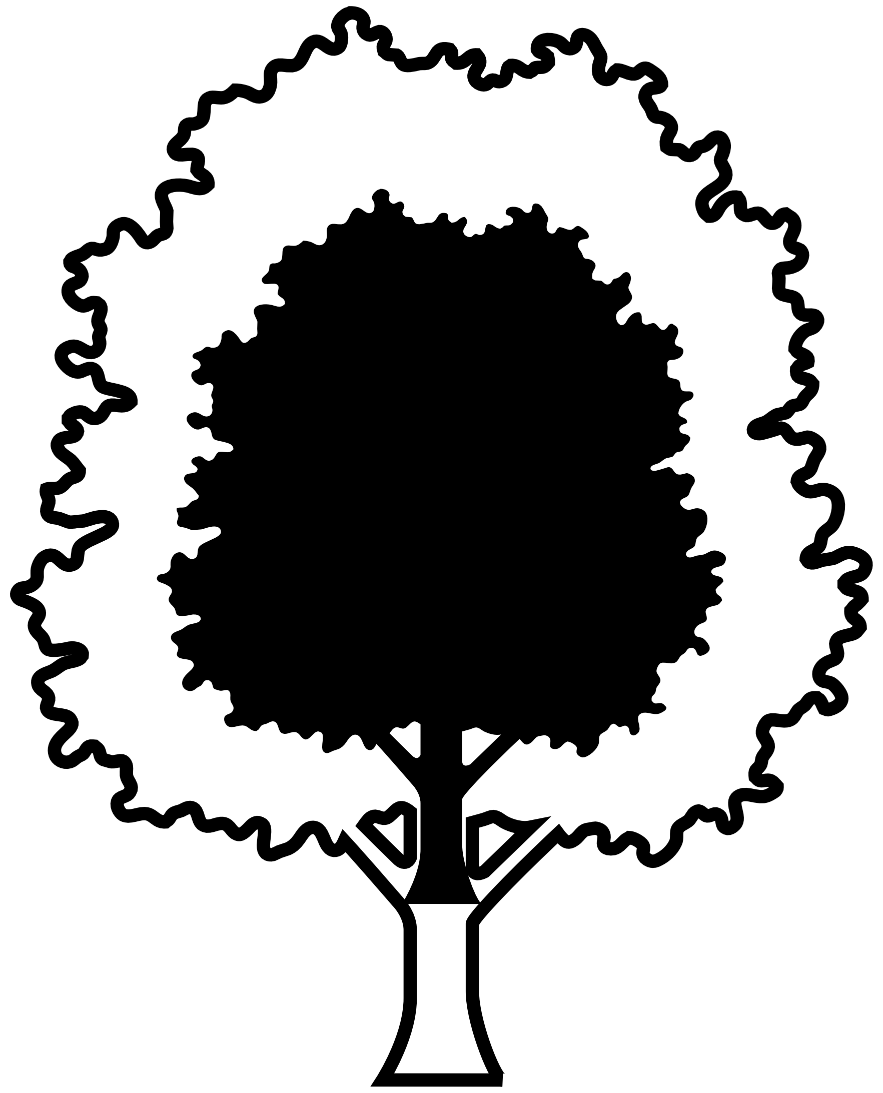


A Midsummer Night’s Tree

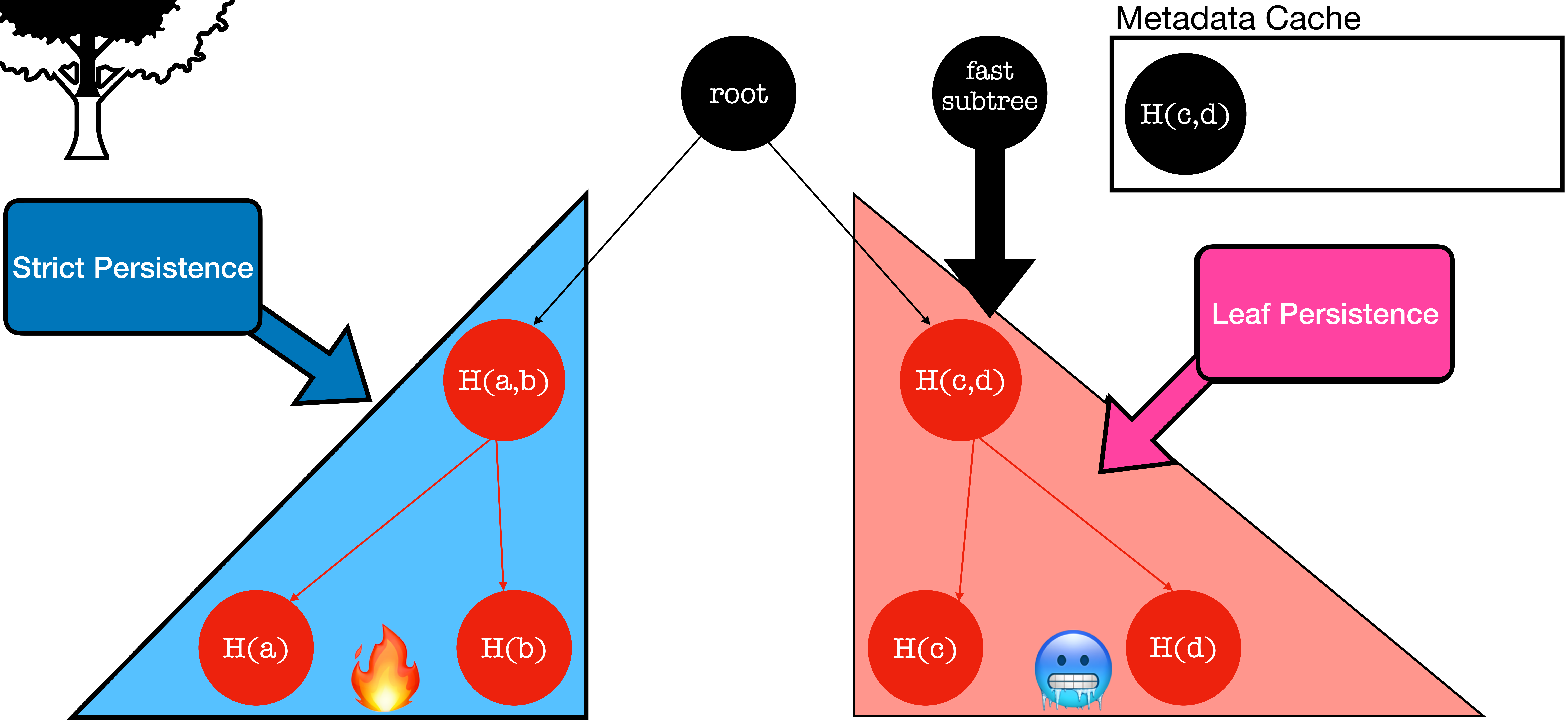


A Midsummer Night’s Tree

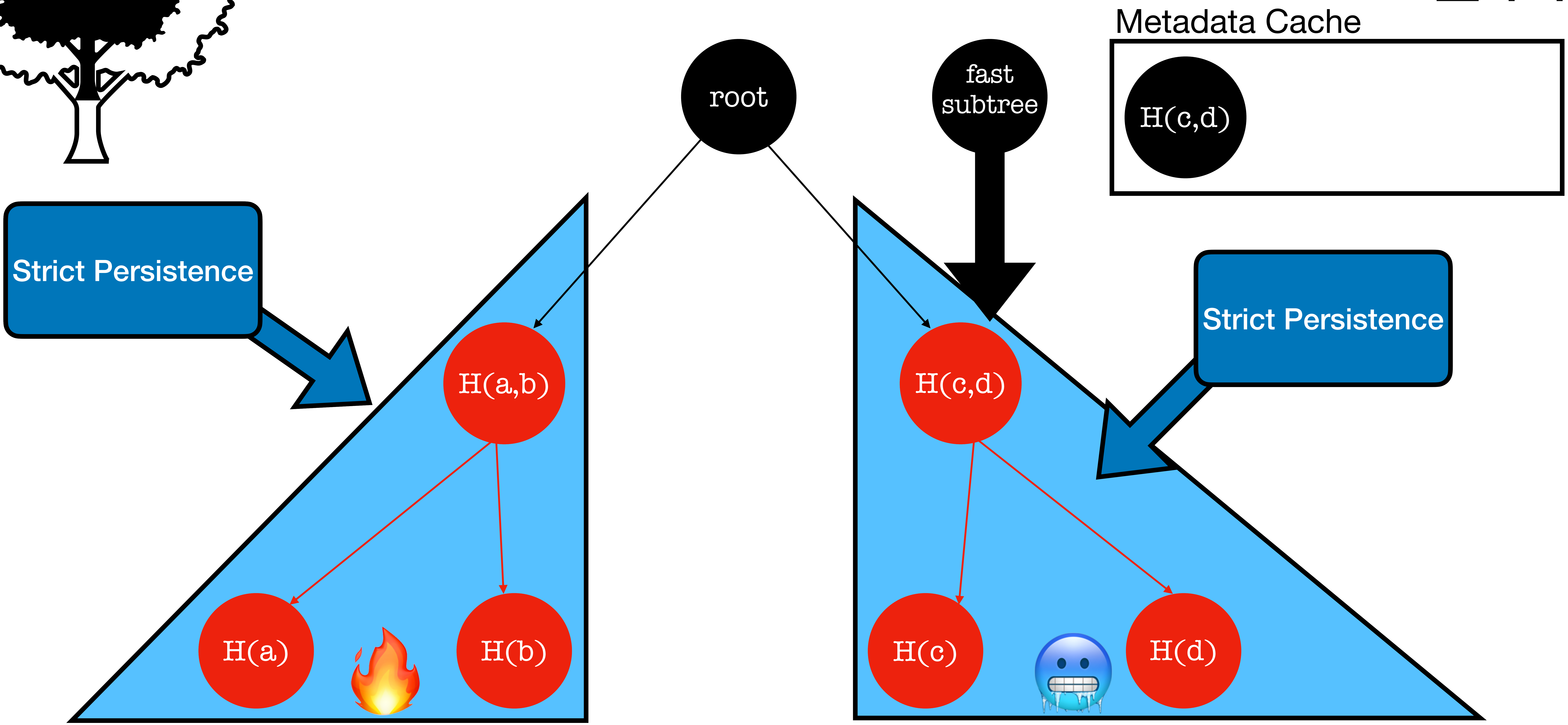




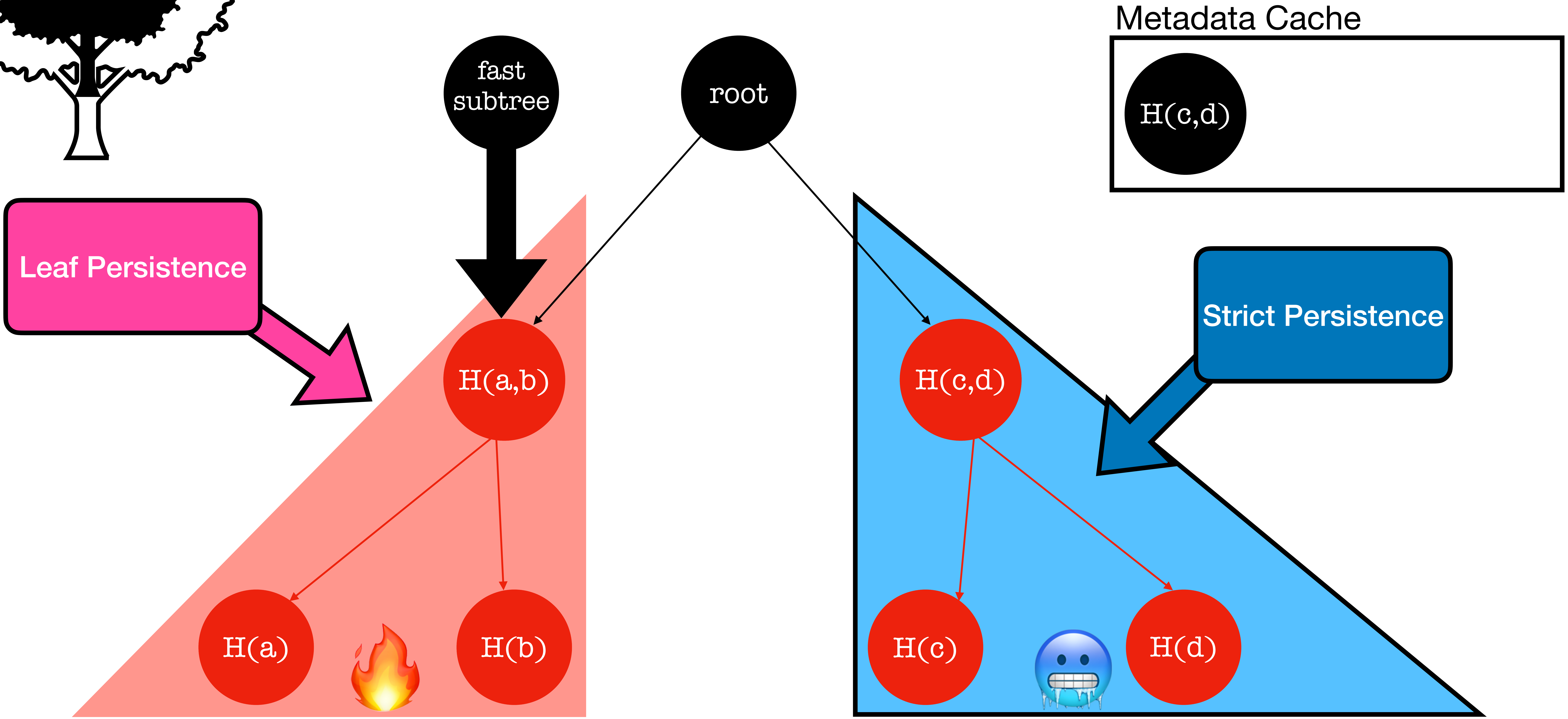
A Midsummer Night’s Tree

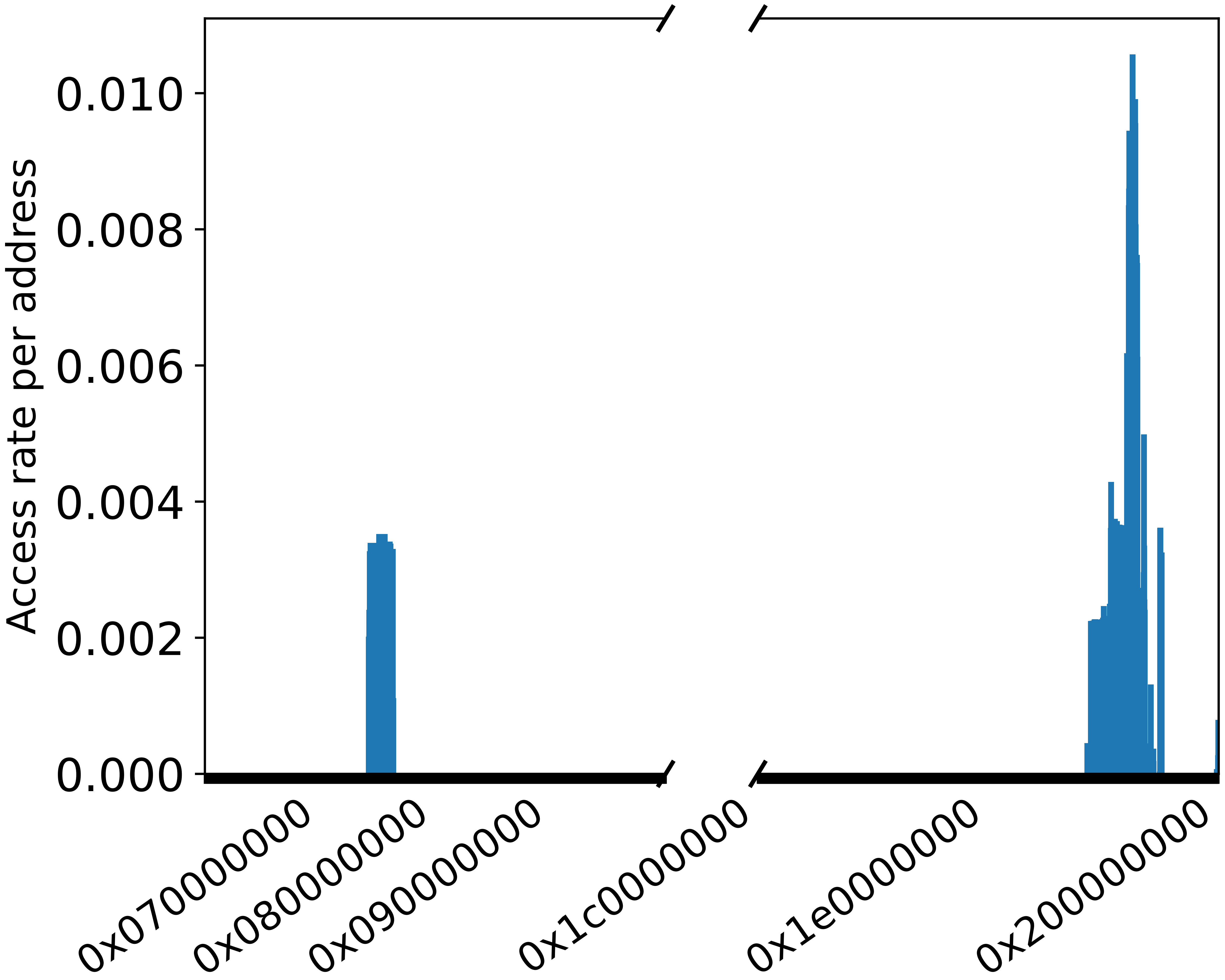


A Midsummer Night’s Tree

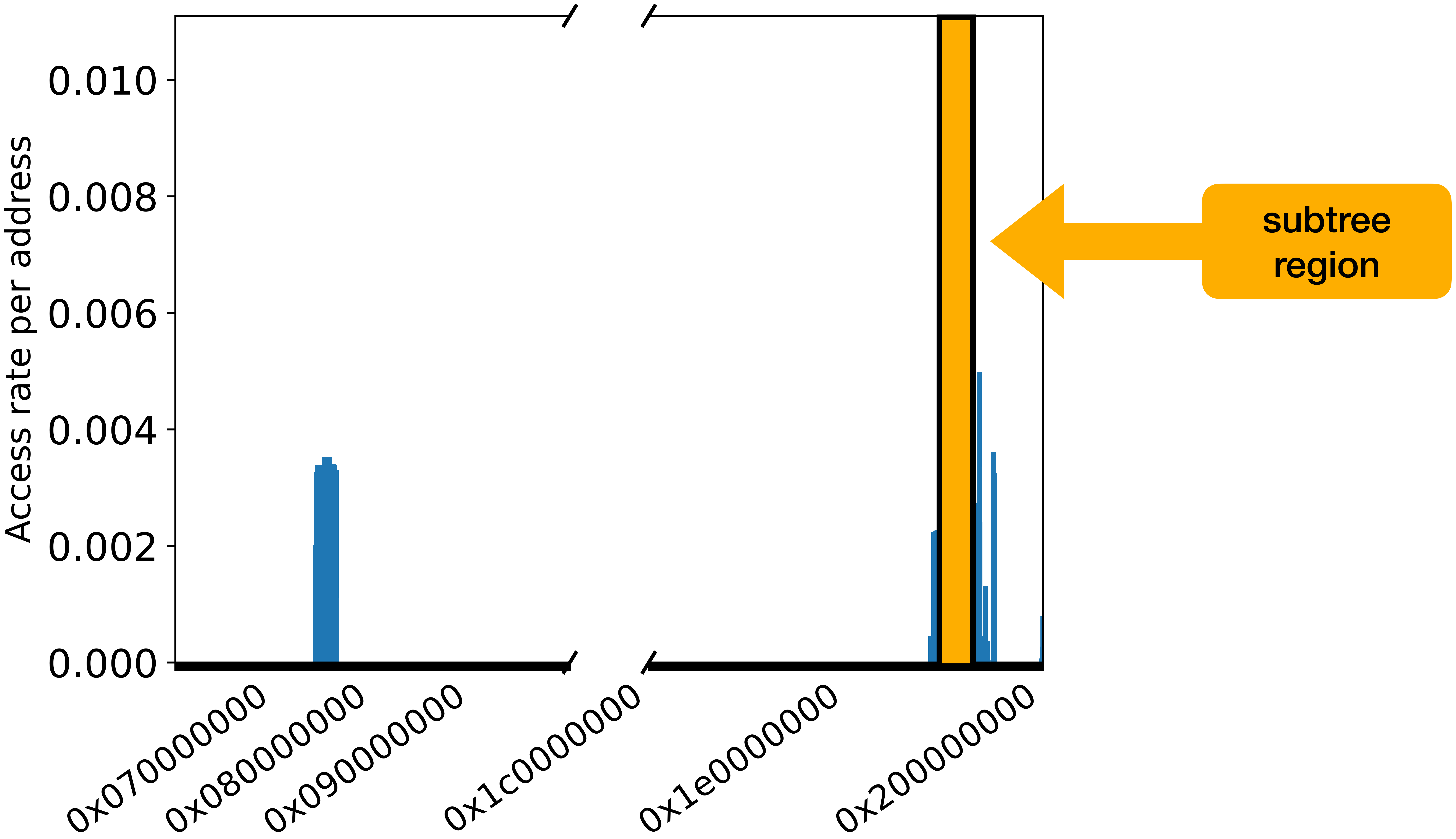


A Midsummer Night’s Tree

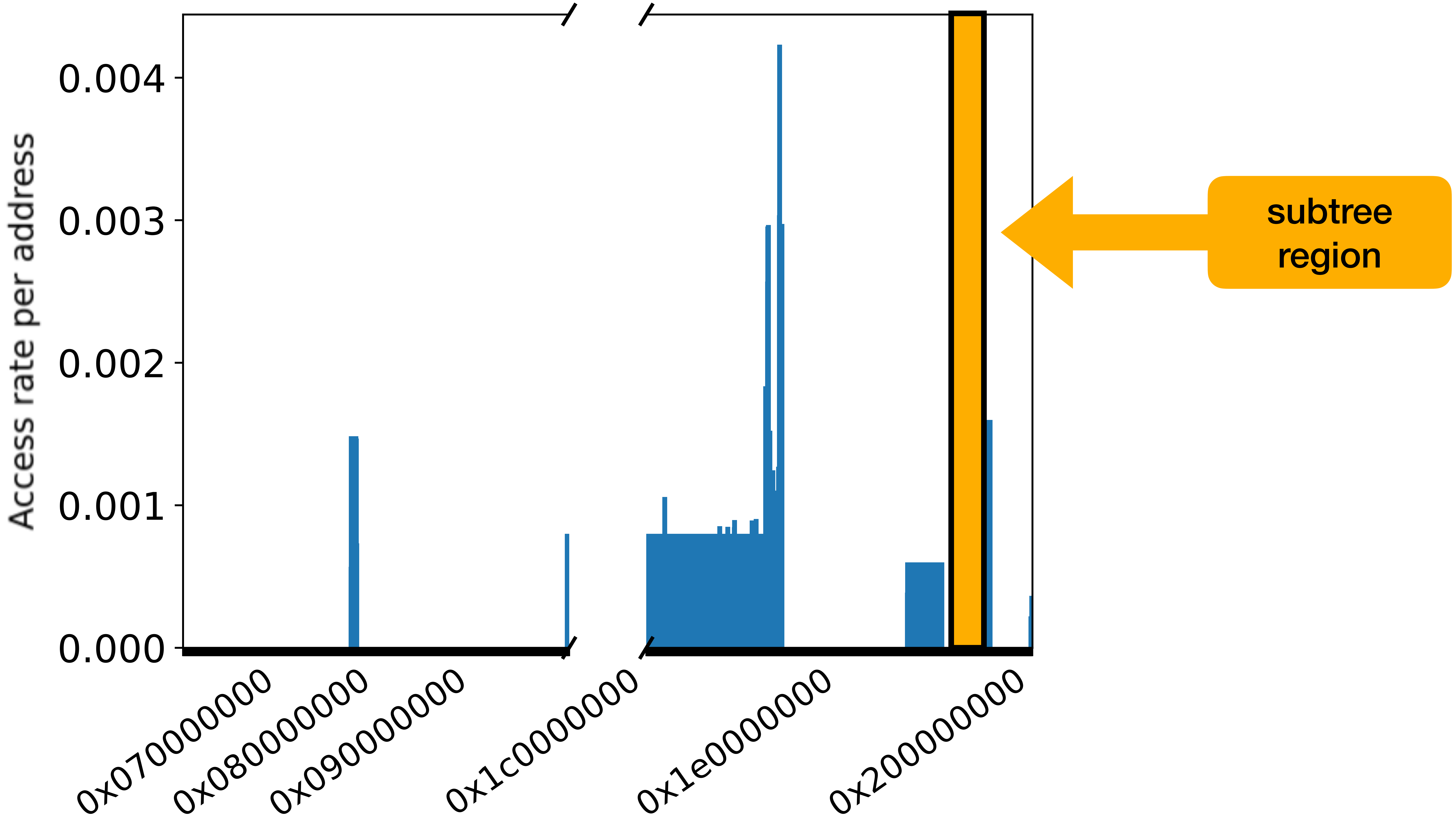



Access rate per address
0.010
0.008
0.006
0.004
0.002
0.000
0x070000000
0x080000000
0x090000000
0x1c0000000
0x1e0000000
0x200000000

perlbench


Access rate per address
0.010
0.008
0.006
0.004
0.002
0.000
0x070000000
0x080000000
0x090000000
0x1c0000000
0x1e0000000
0x200000000
subtree region

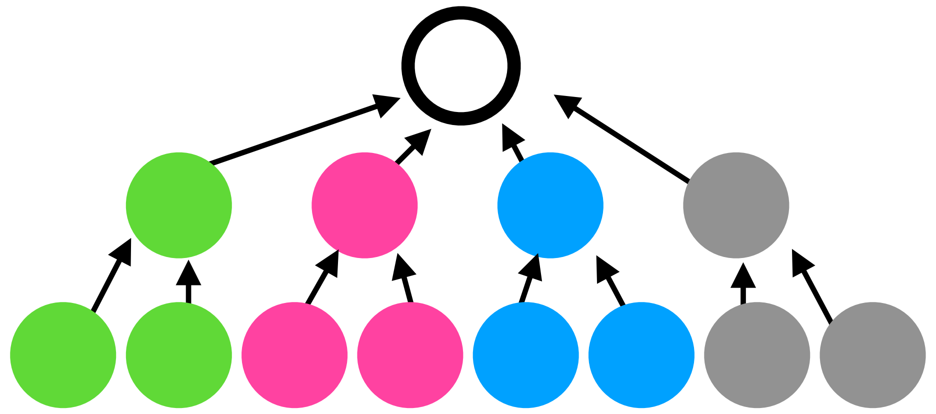
perlbench

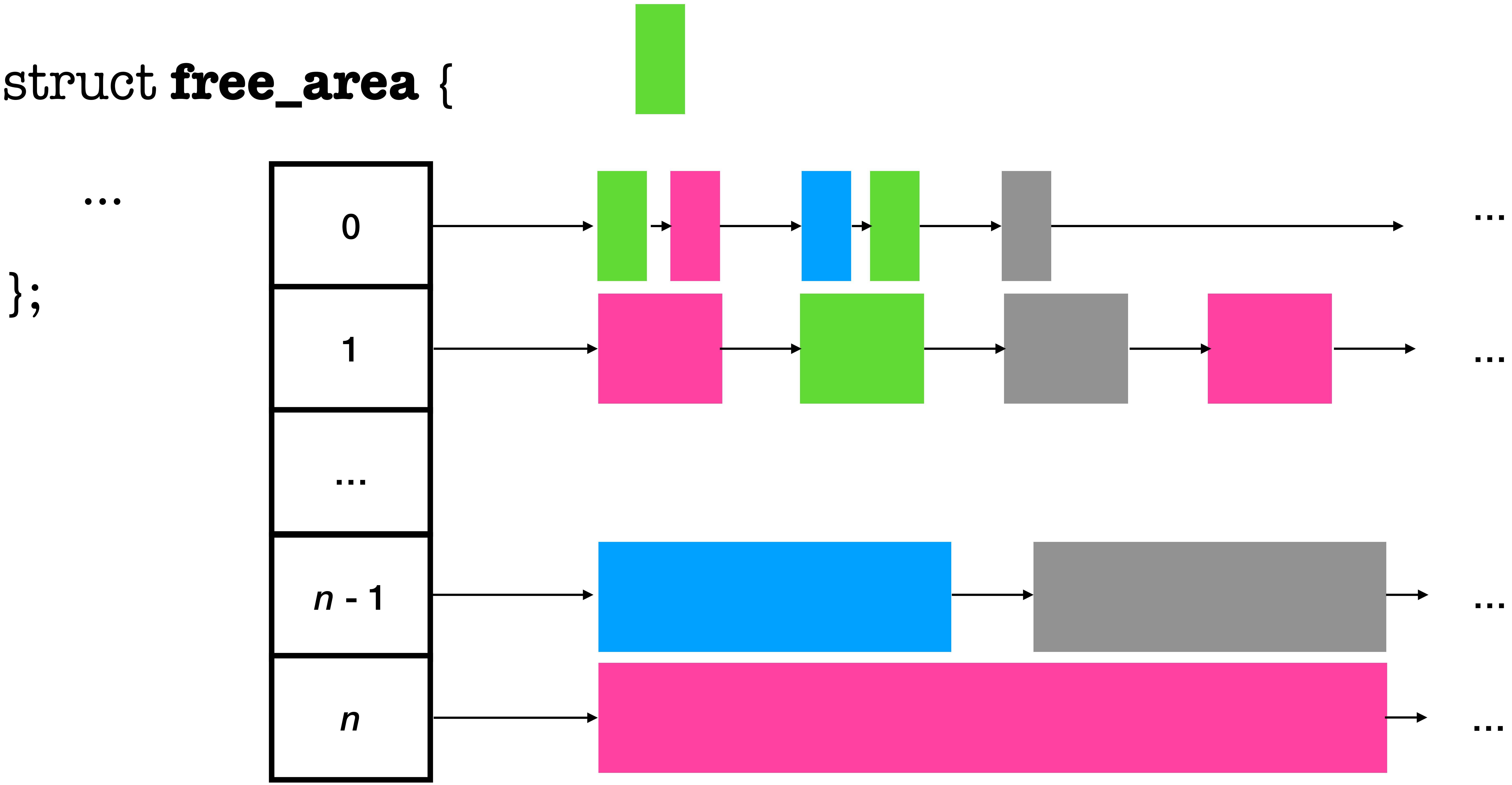


subtree region

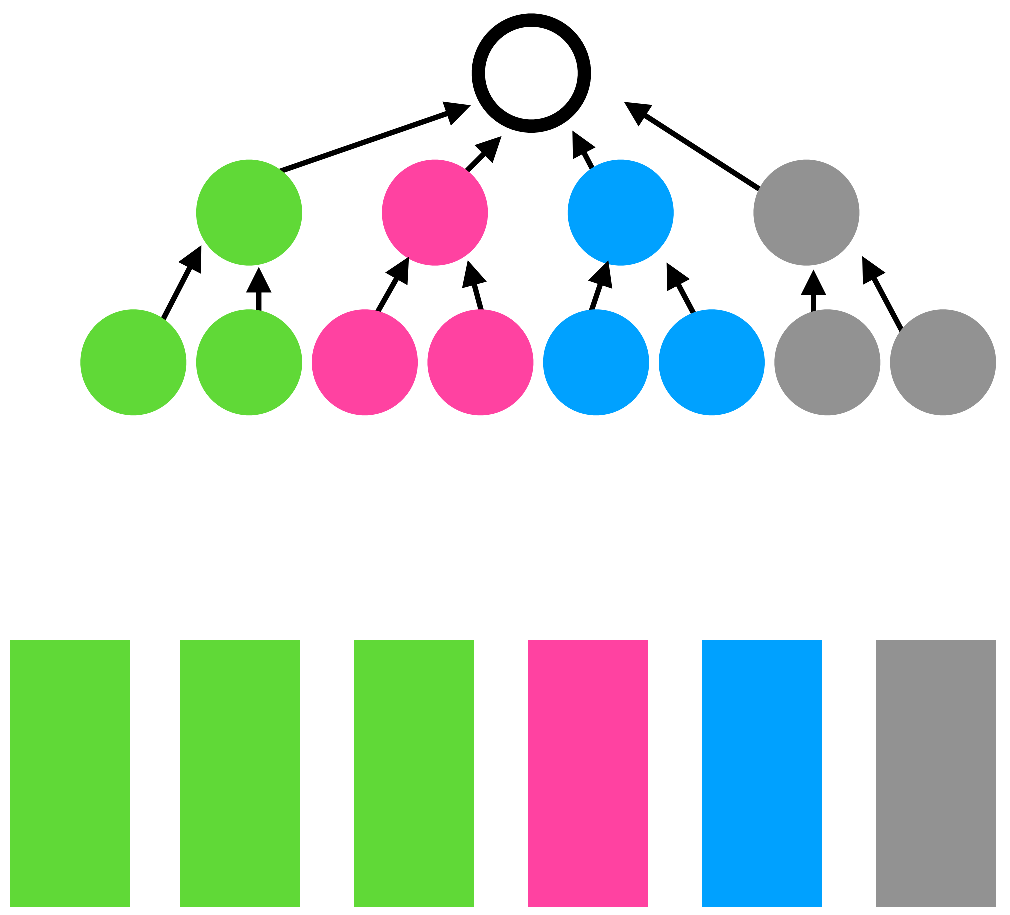
perlbench and lbm

Biased Physical Page Allocation

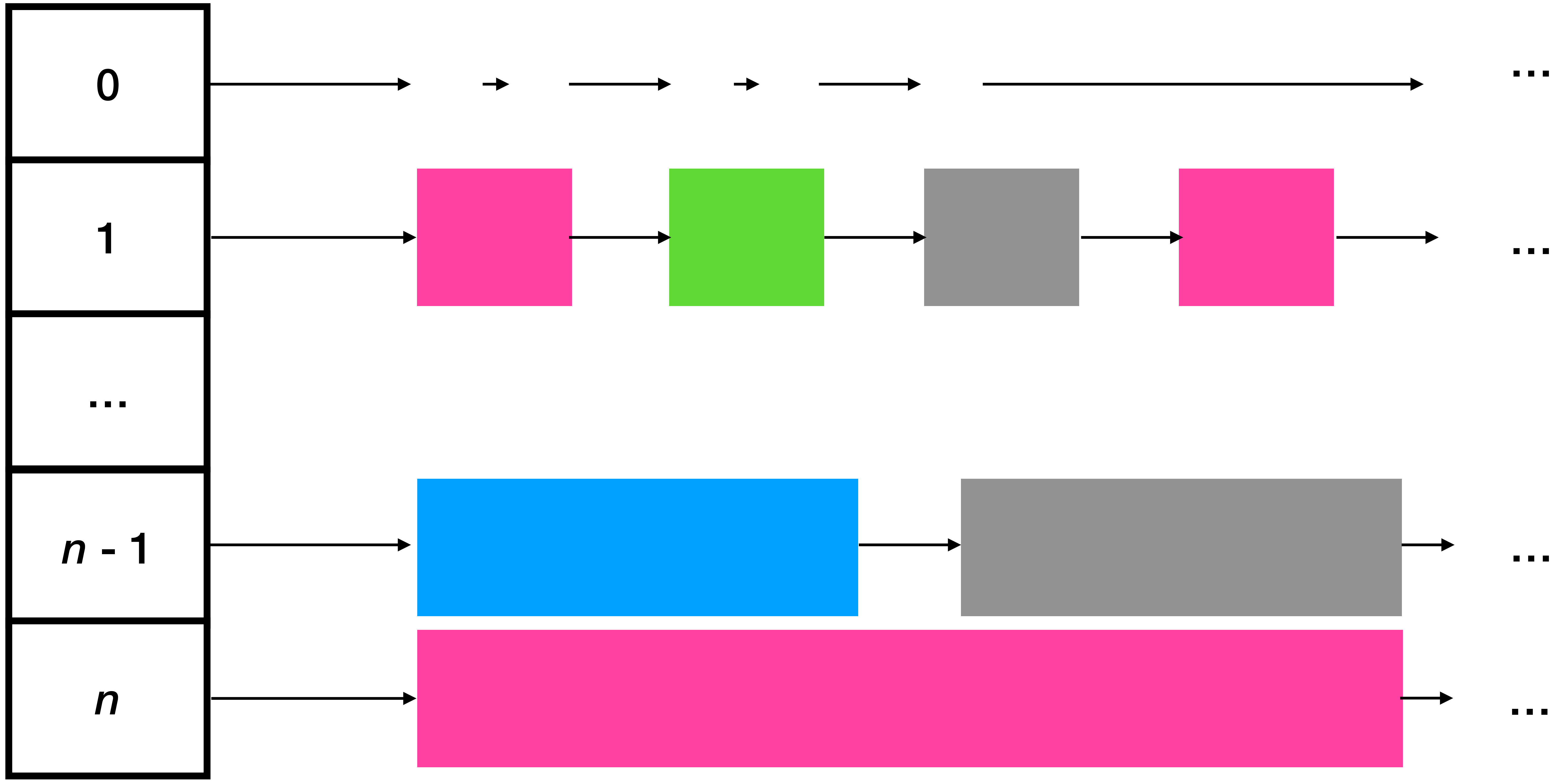




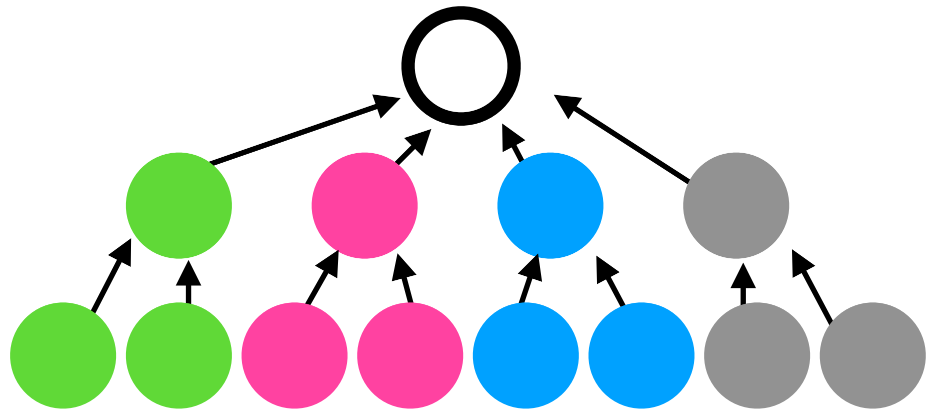
Biased Physical Page Allocation



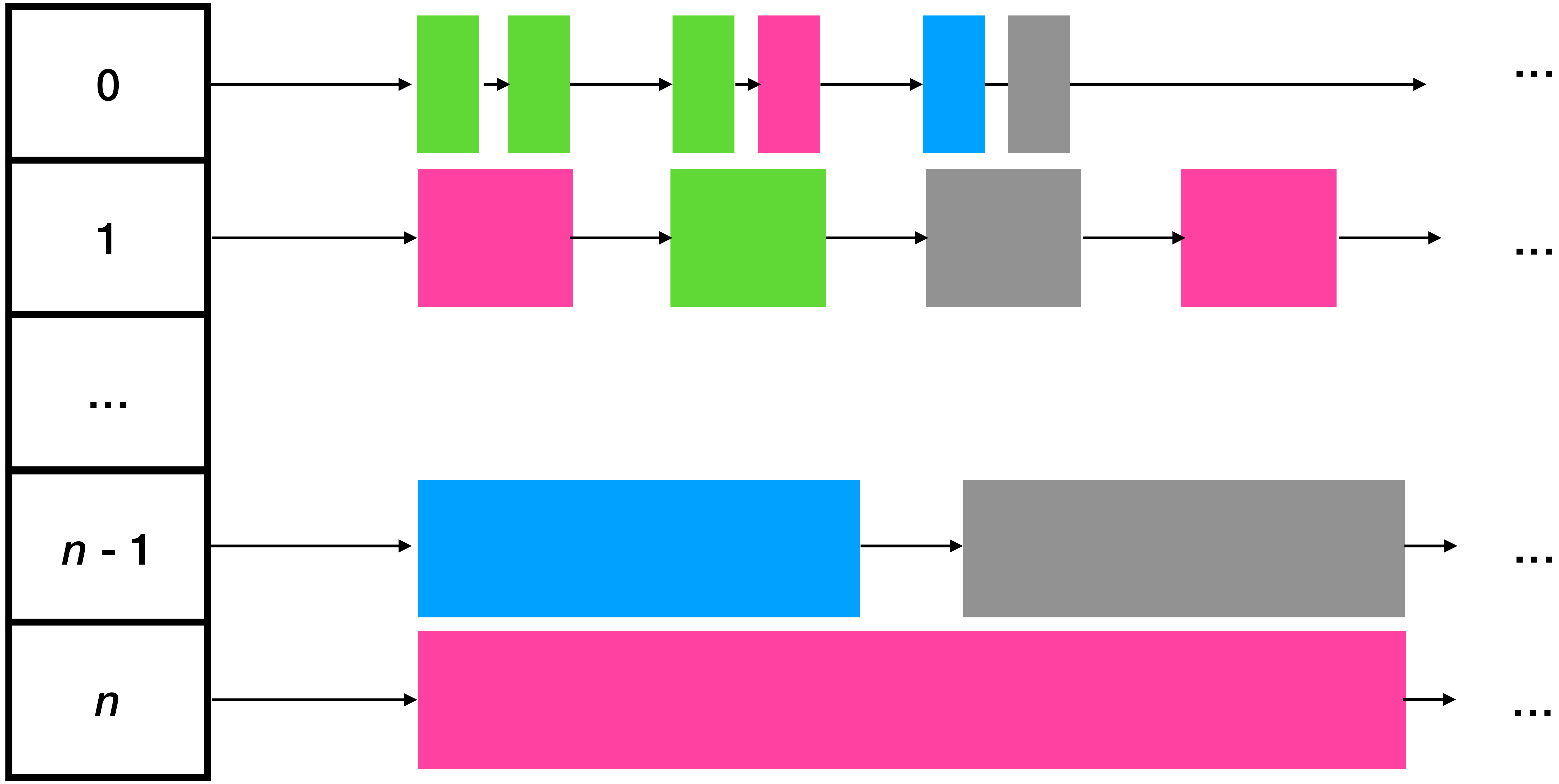
```
struct free_area {
    ...
};
```

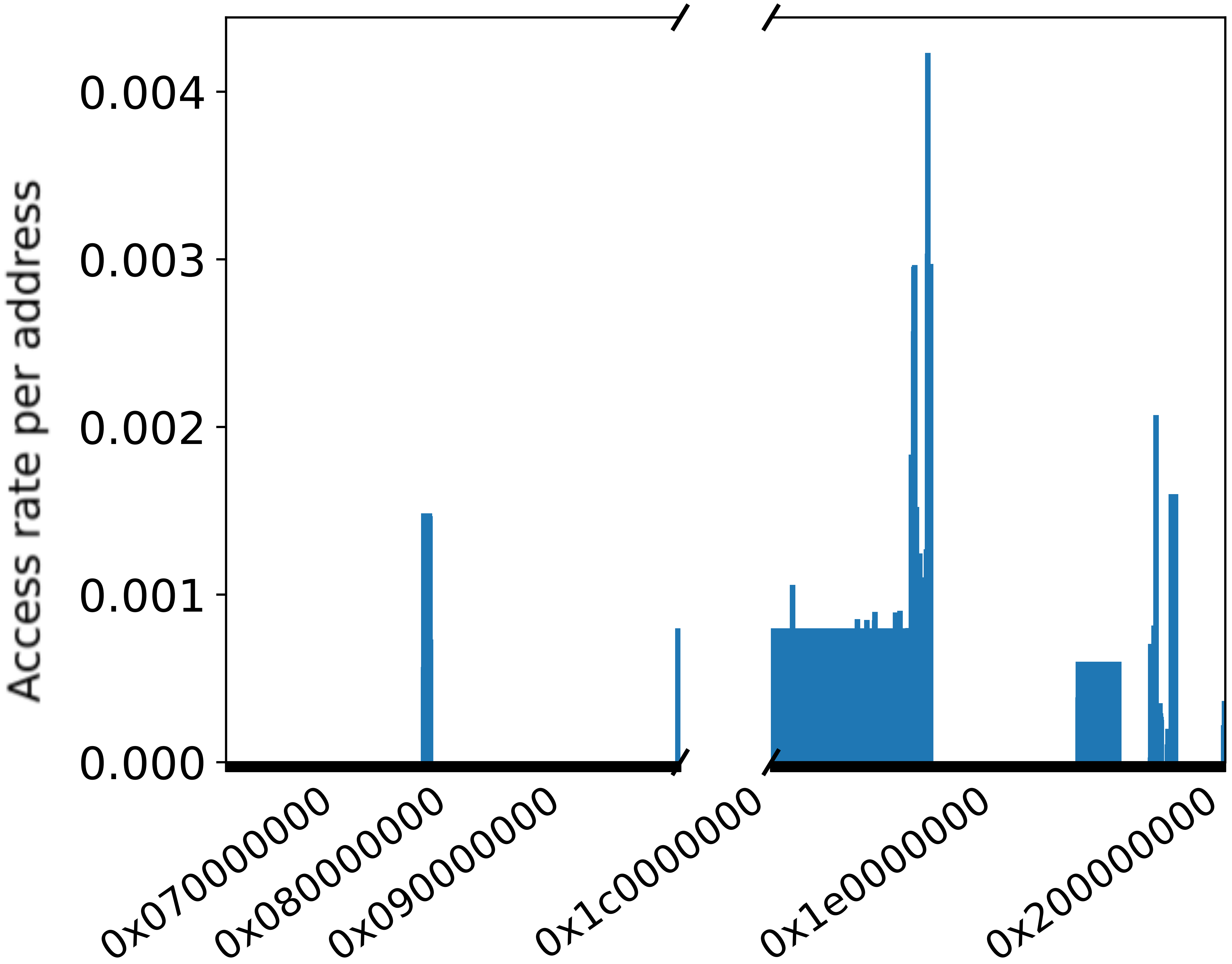


Biased Physical Page Allocation

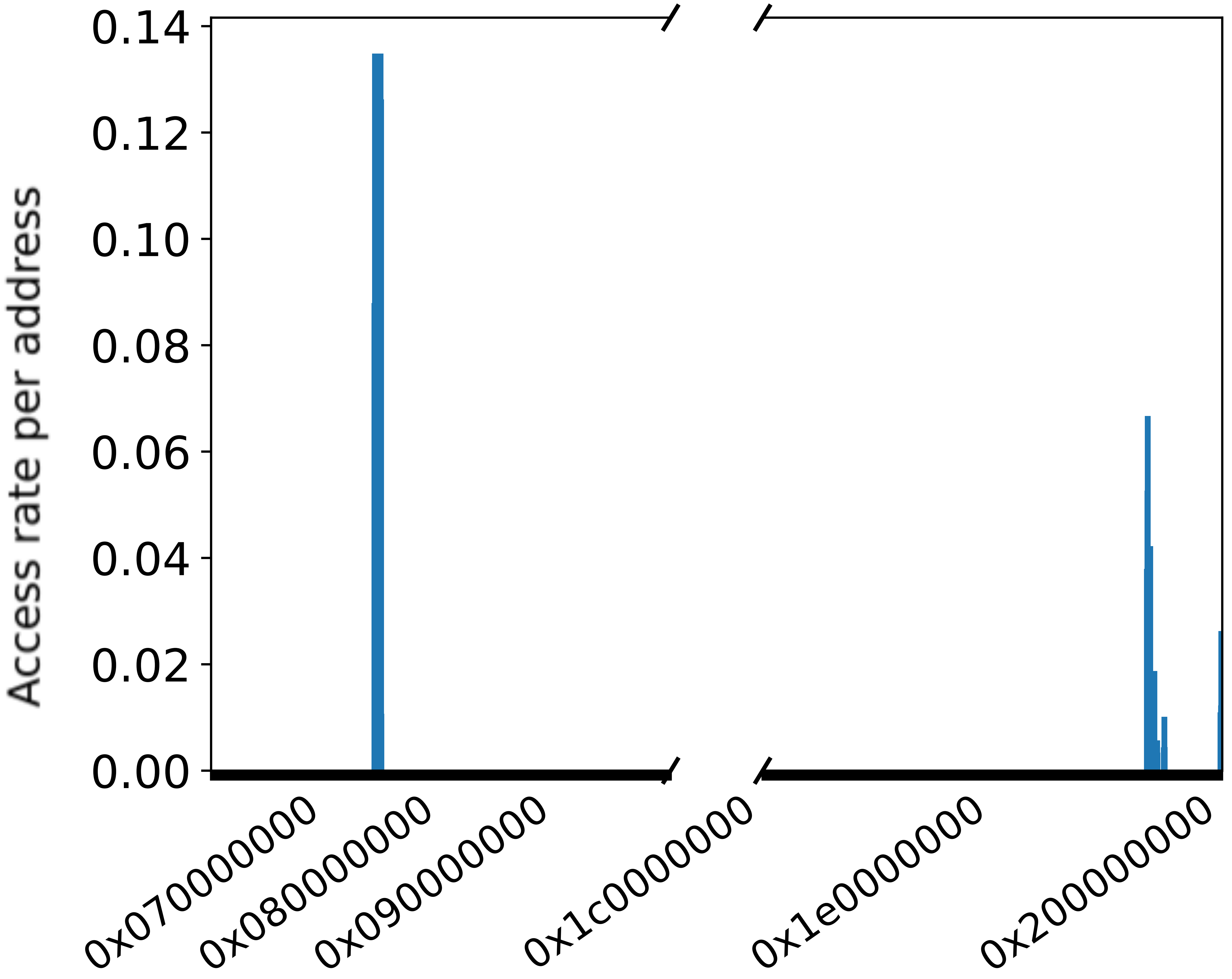


```
struct free_area {

    ...

};
```




Access rate per address
0.000
0.001
0.002
0.003
0.004
0x070000000
0x080000000
0x090000000
0x1c0000000
0x1e0000000
0x200000000

unbiased

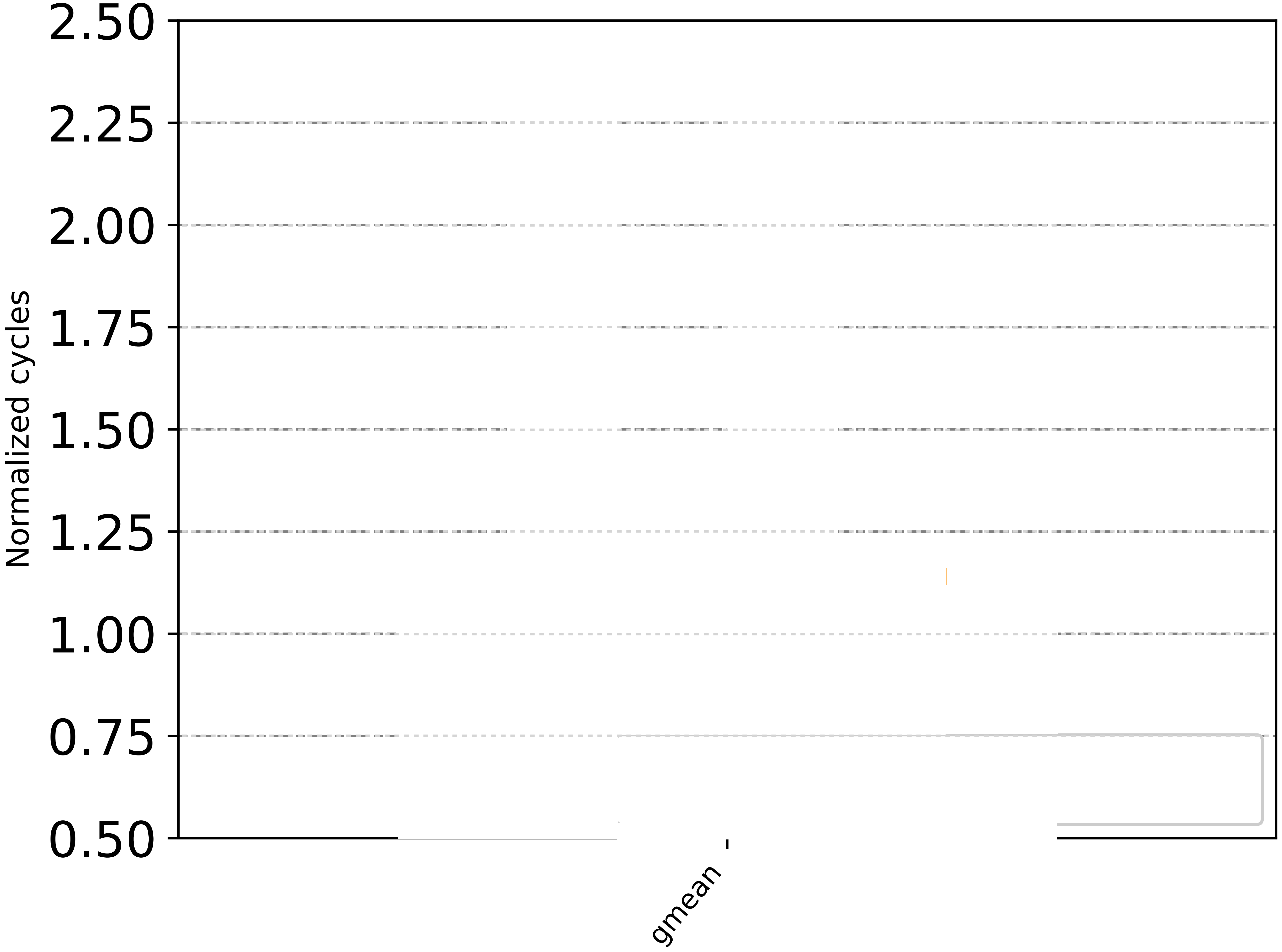

Access rate per address
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0x070000000
0x080000000
0x090000000
0x1c0000000
0x1e0000000
0x200000000

biased

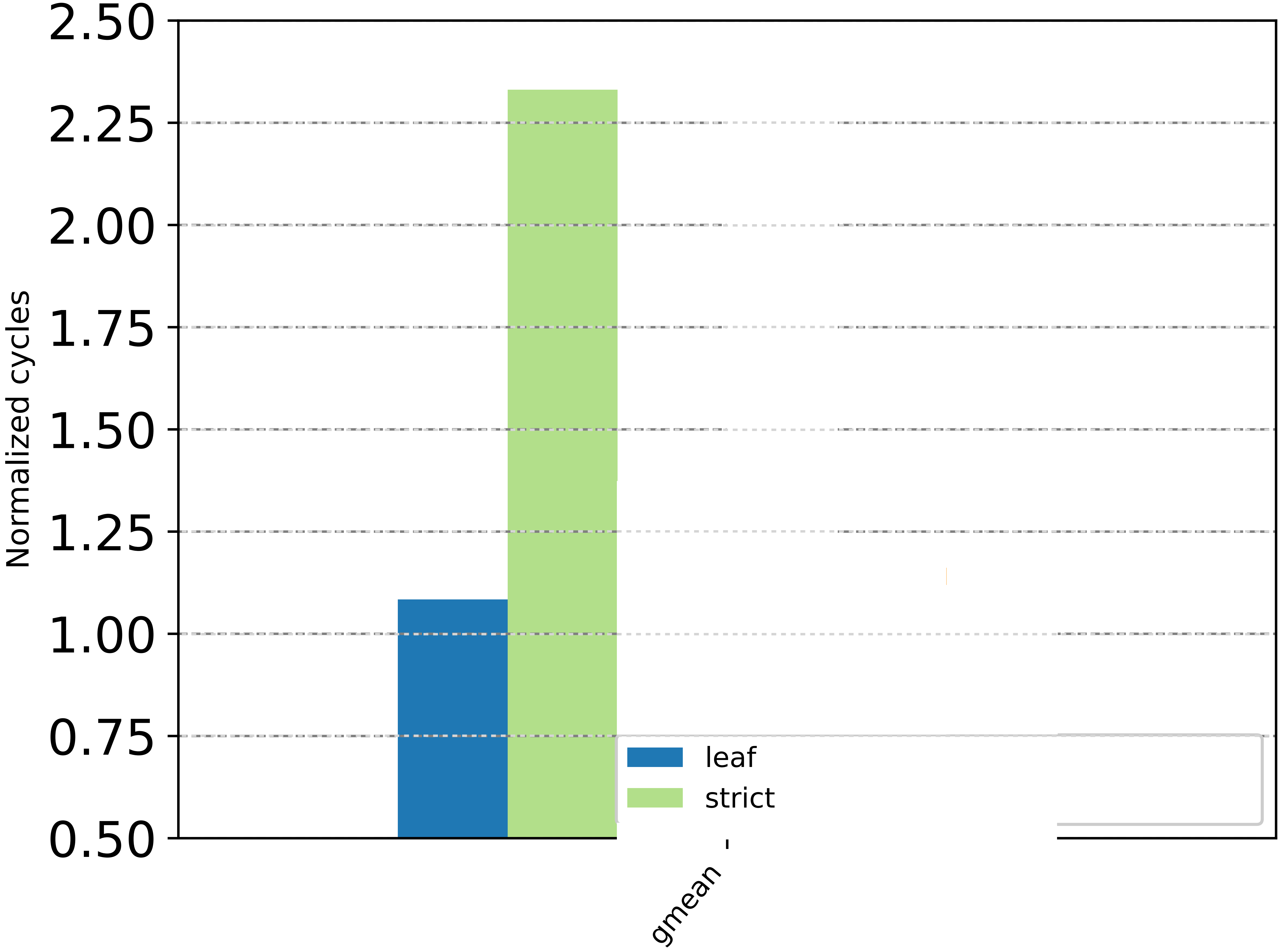
Hardware Overhead

	Volatile Overhead	Non-Volatile Overhead
AMNT	96 bytes	64 bytes
Anubis, ISCA19	37 kB	64 bytes
BMF, MICRO21	768 bytes	4 kB

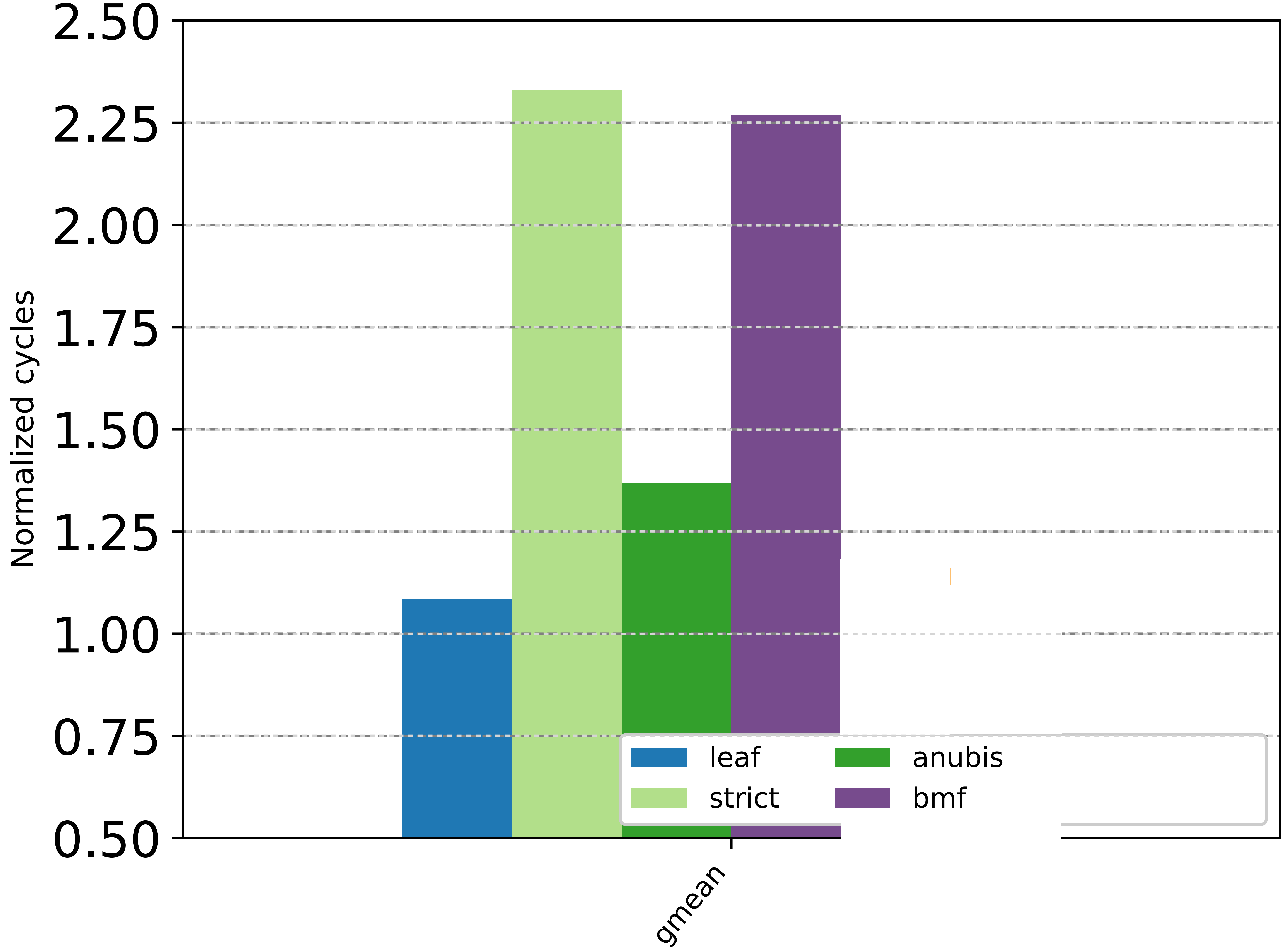
Performance Overhead



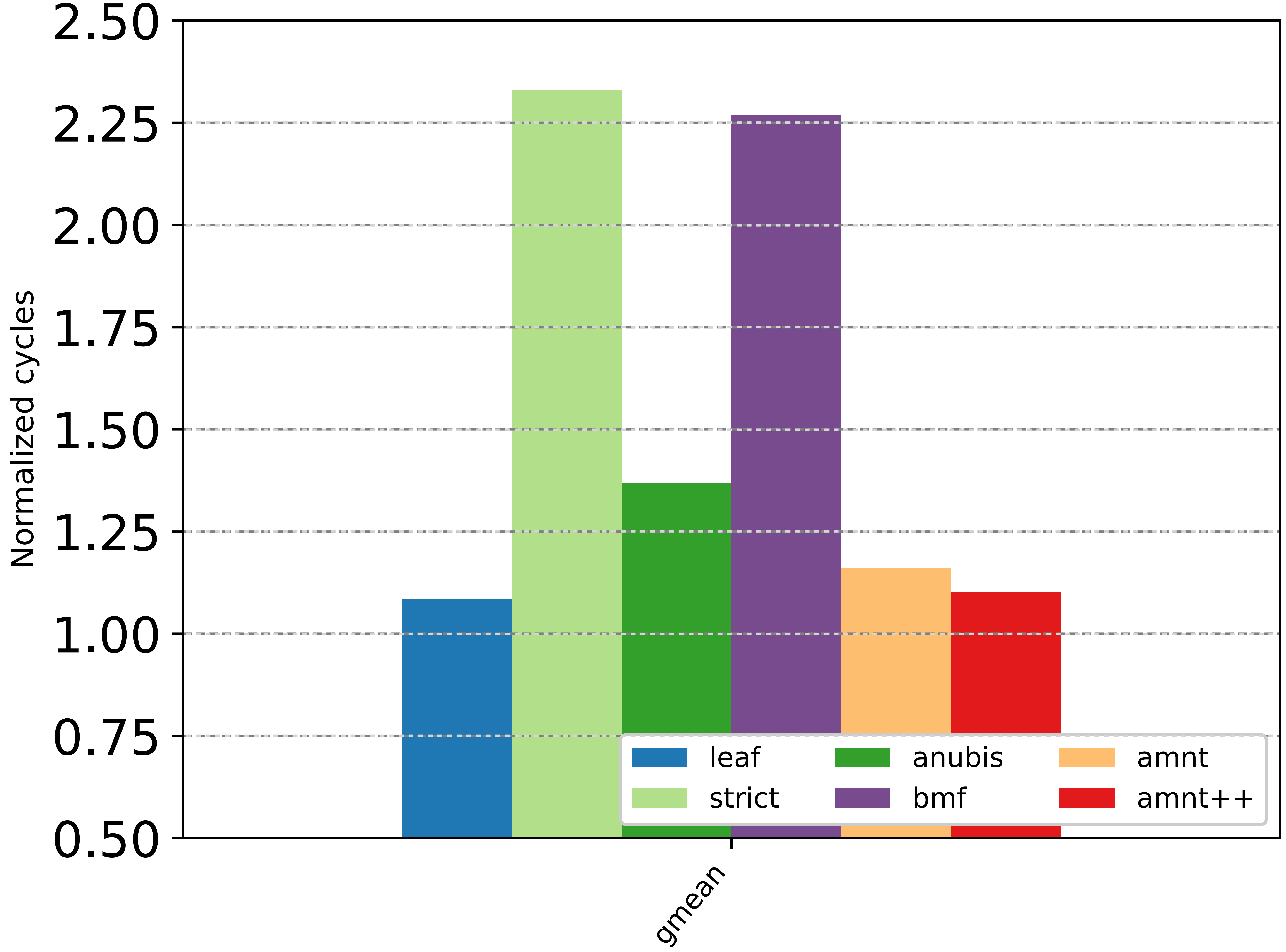
Performance Overhead



Performance Overhead

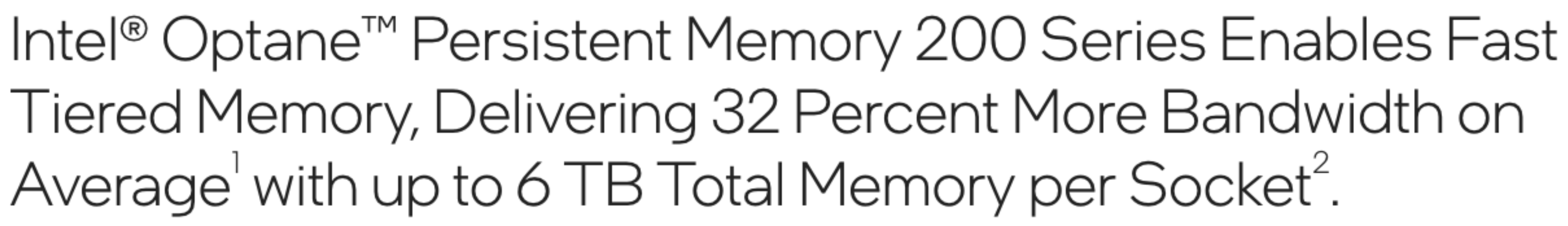


Performance Overhead

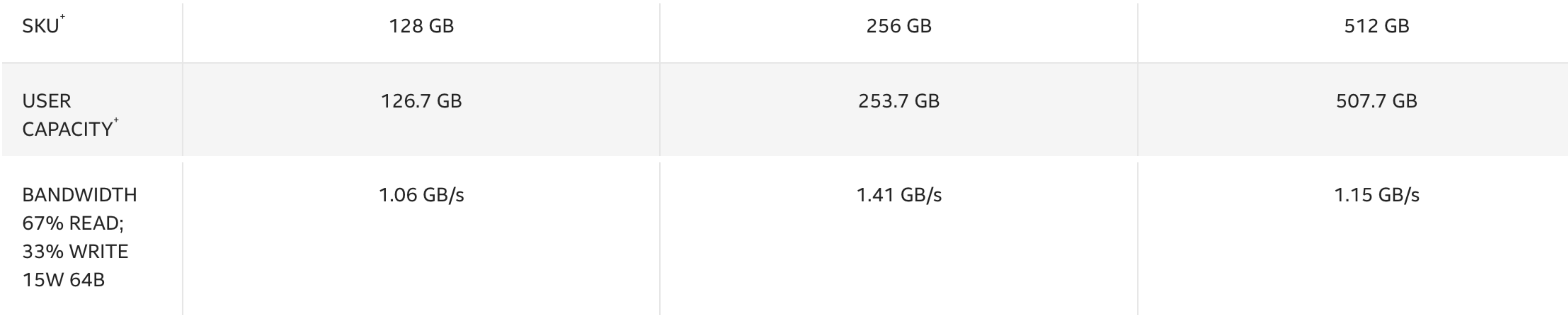


Recovery

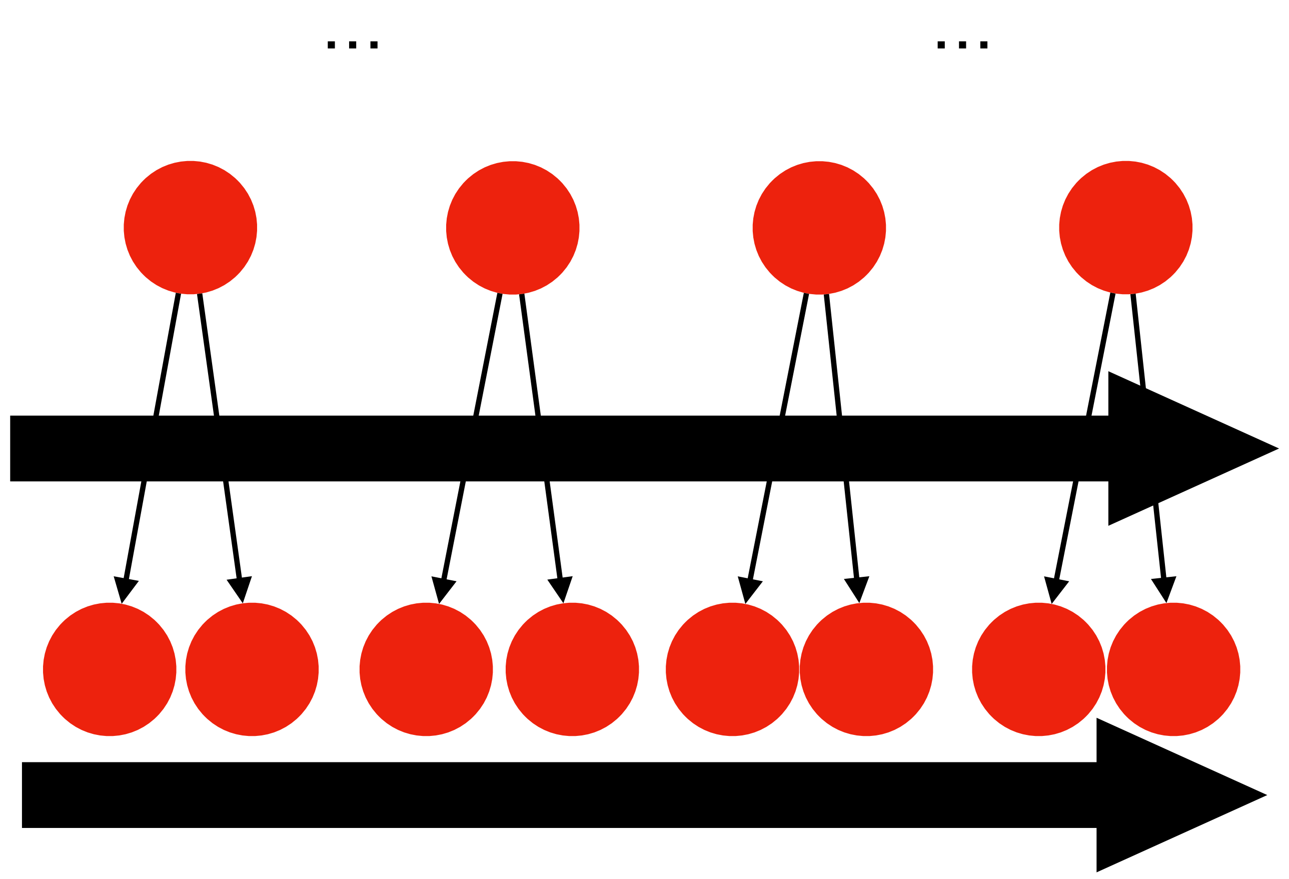
https://www.intel.com/content/www/us/en/products/docs/memory-storage/optane-persistent-memory/optane-persistent-memory-200-series-brief.html



Intel® Optane™ Persistent Memory 200 Series Enables Fast Tiered Memory, Delivering 32 Percent More Bandwidth on Average[1] with up to 6 TB Total Memory per Socket[2].

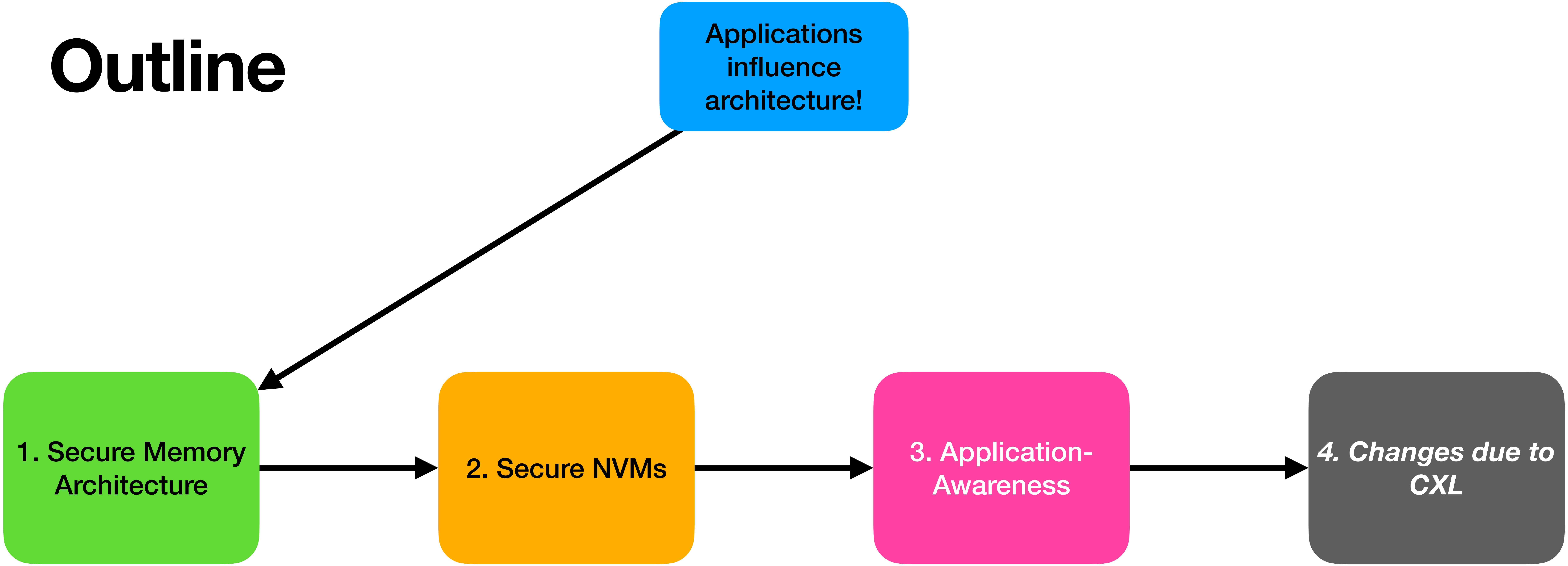


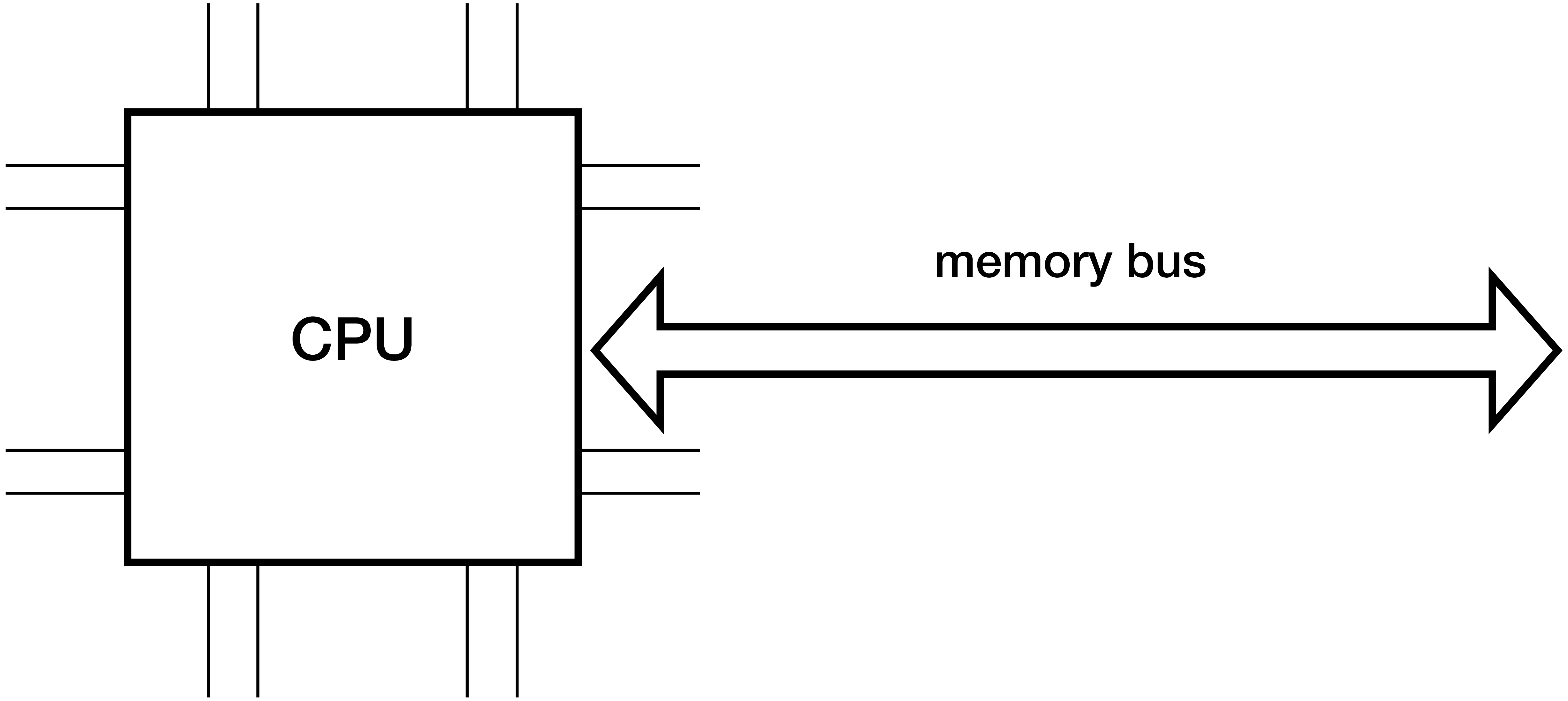
SKU*	128 GB	256 GB	512 GB
USER CAPACITY*	126.7 GB	253.7 GB	507.7 GB
BANDWIDTH 67% READ; 33% WRITE 15W 64B	1.06 GB/s	1.41 GB/s	1.15 GB/s



	128GB	256GB	512GB	...	128TB
64B words to fetch	38.3M	76.7M	153.4M	...	39.3B
time to recover (leaf)	1.89 sec	2.84 sec	6.9 sec	...	30 min
time to recover (AMNT)	0.03 sec	0.04 sec	0.11 sec	...	32 sec

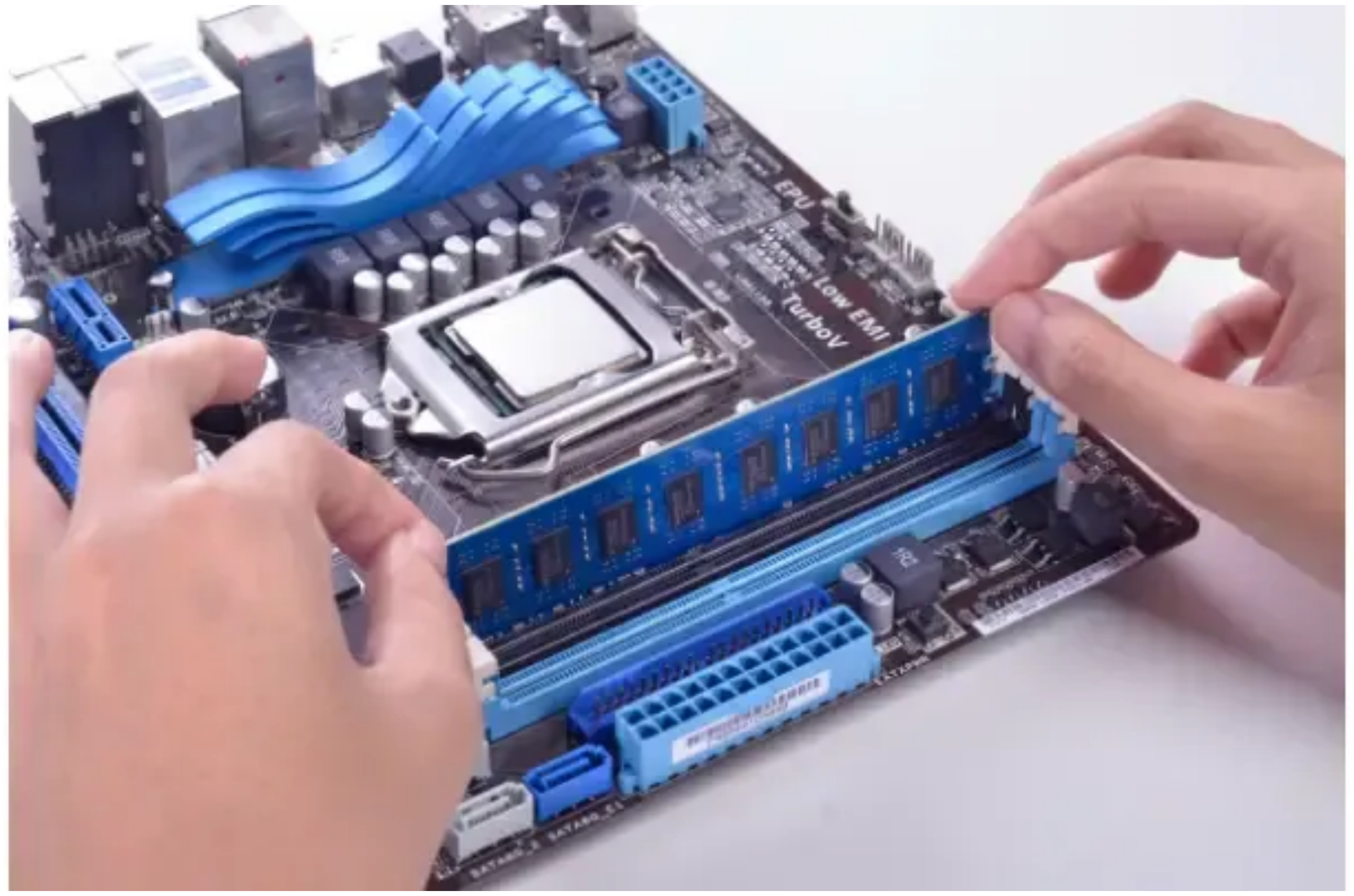
Outline

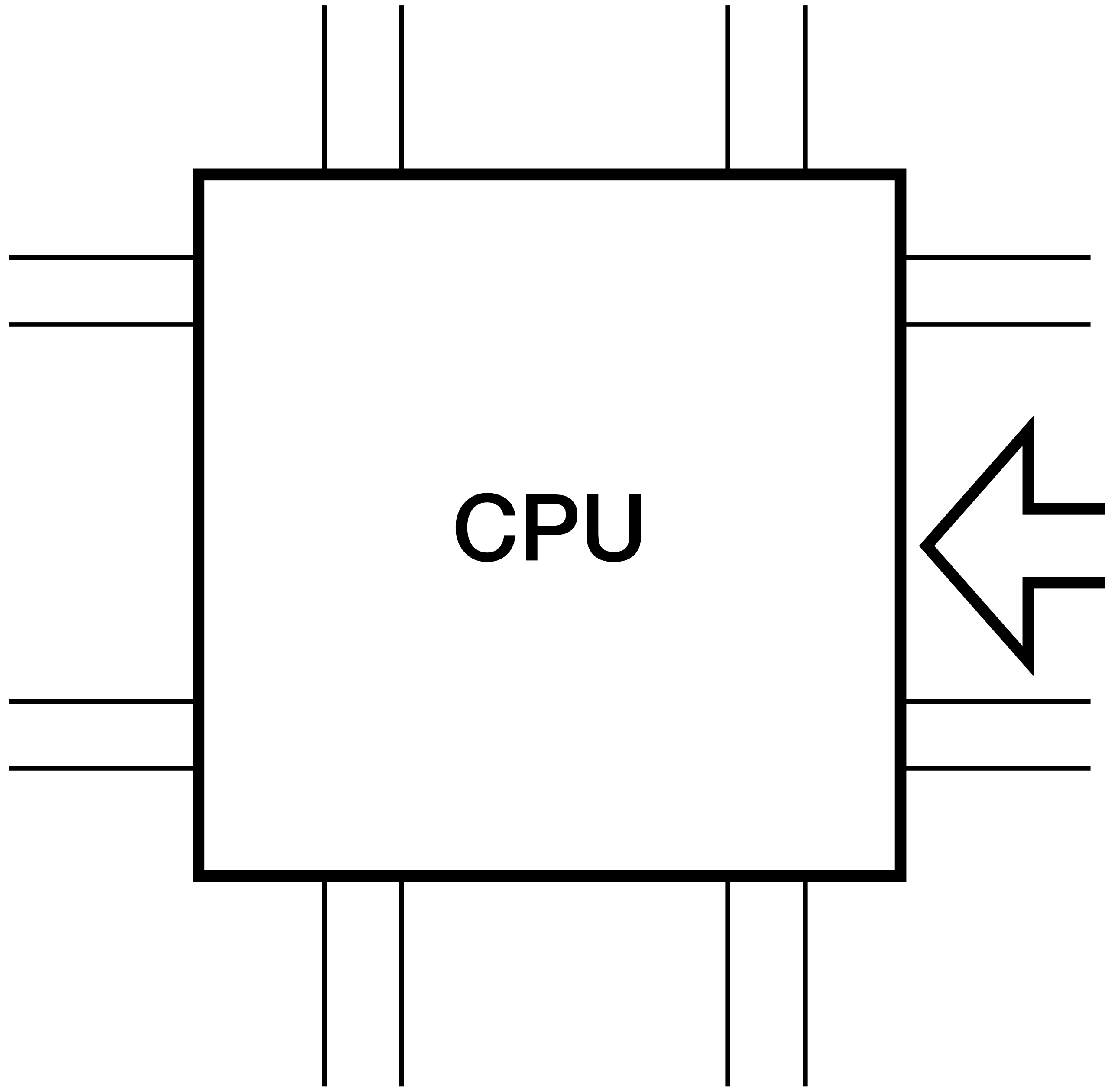

Applications influence architecture!
1. Secure Memory Architecture
2. Secure NVMs
3. Application-Awareness
4. Changes due to CXL



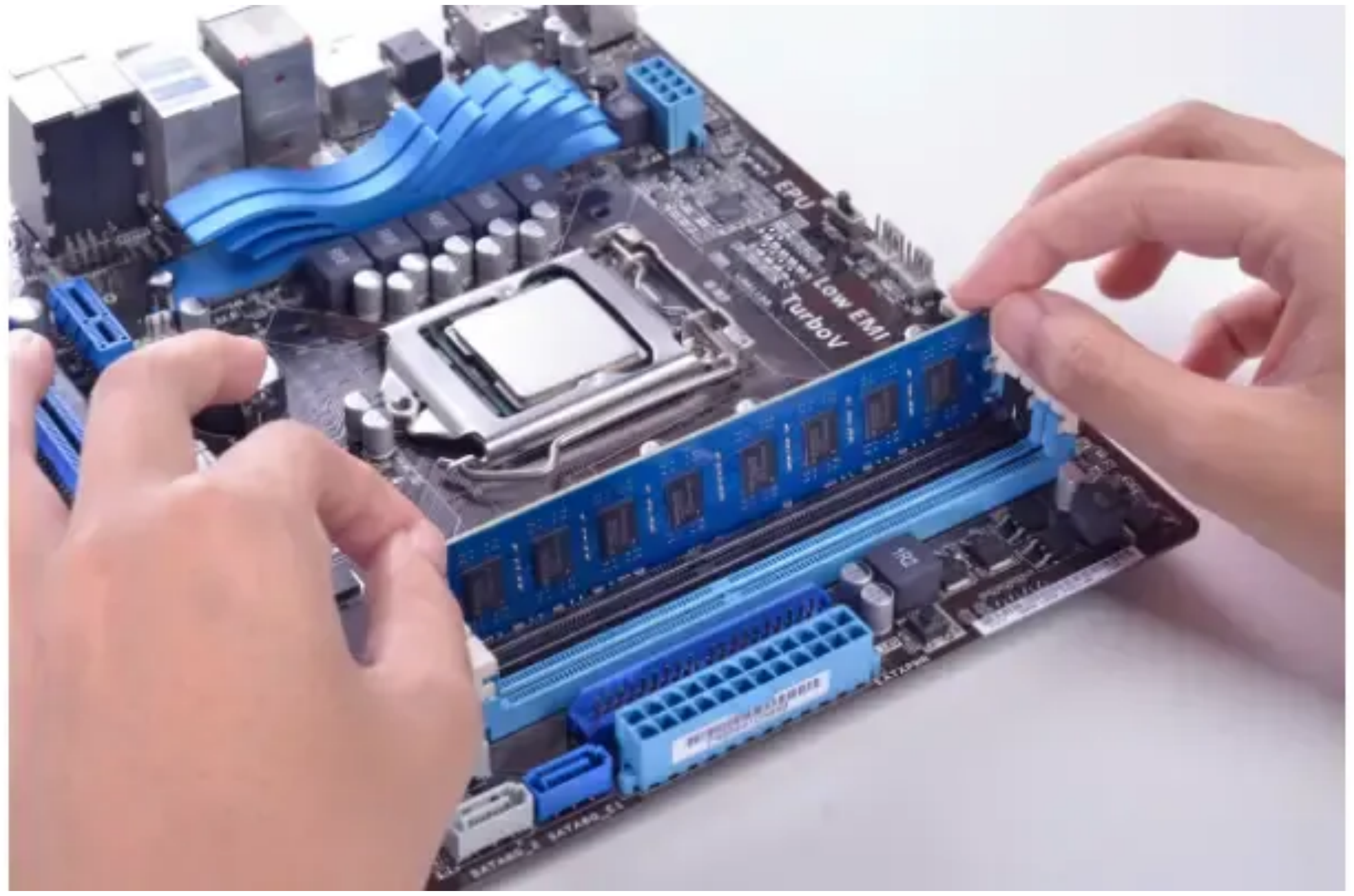
CPU

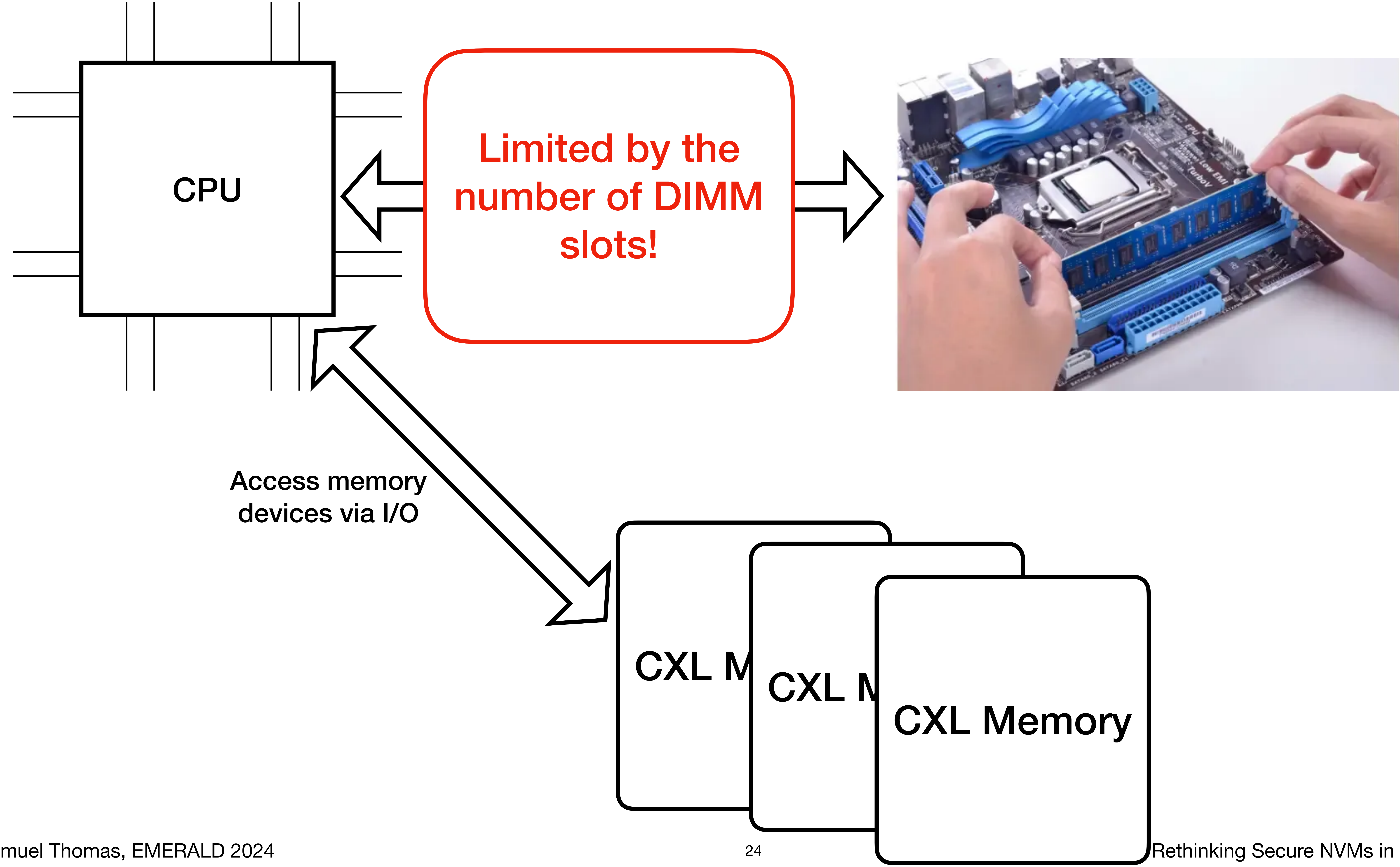
memory bus





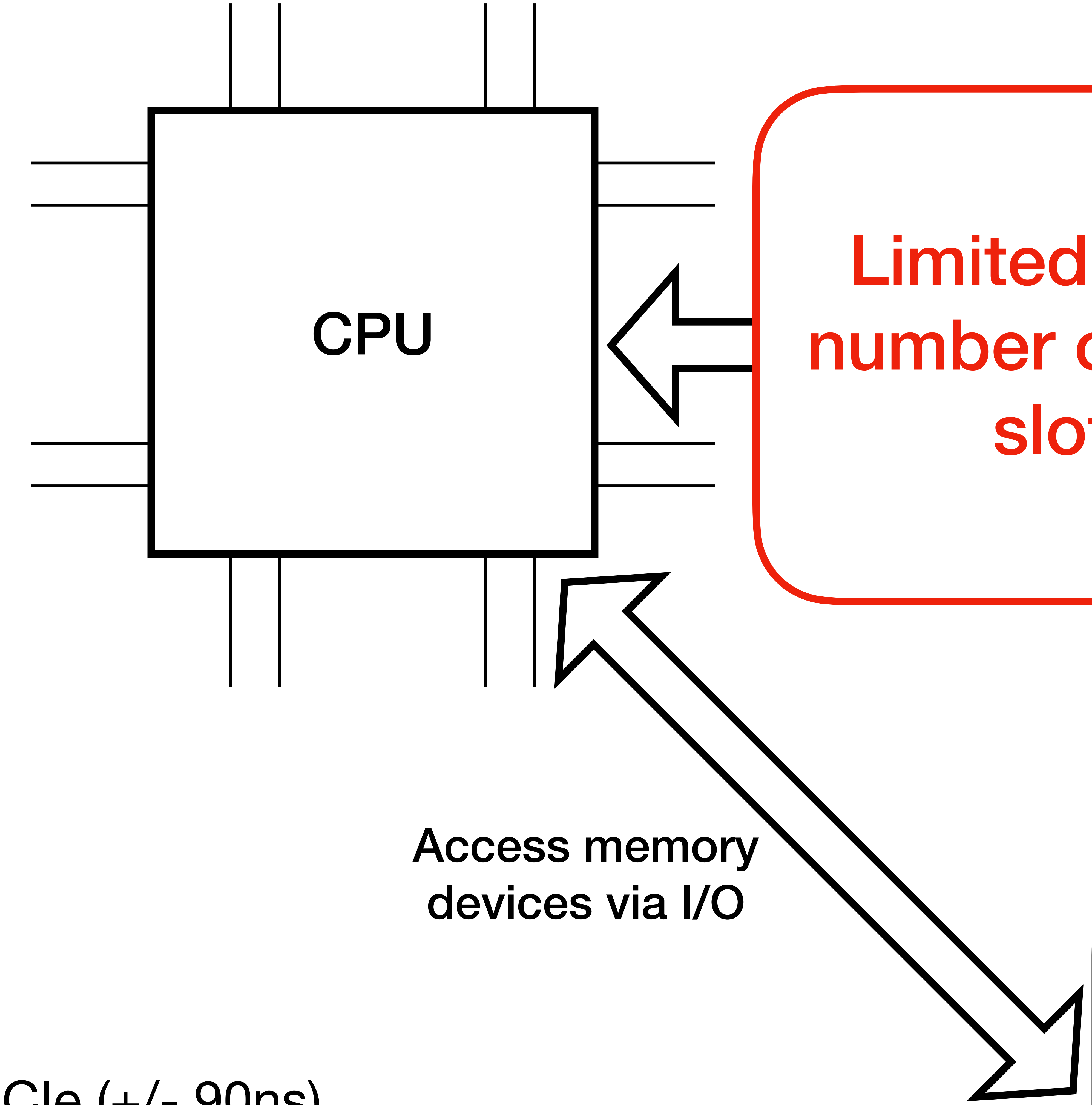
Limited by the number of DIMM slots!



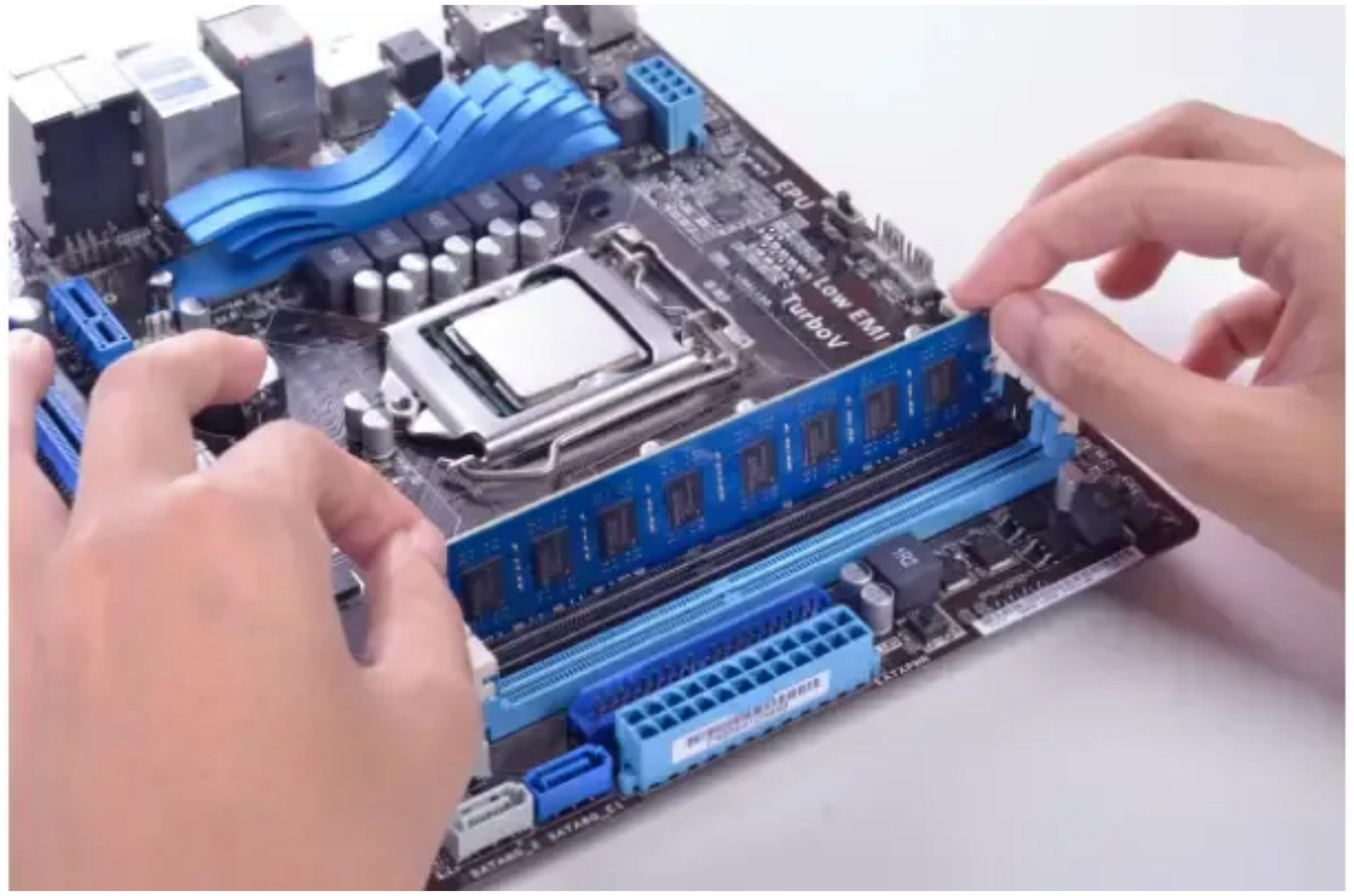


Limited by the number of DIMM slots!

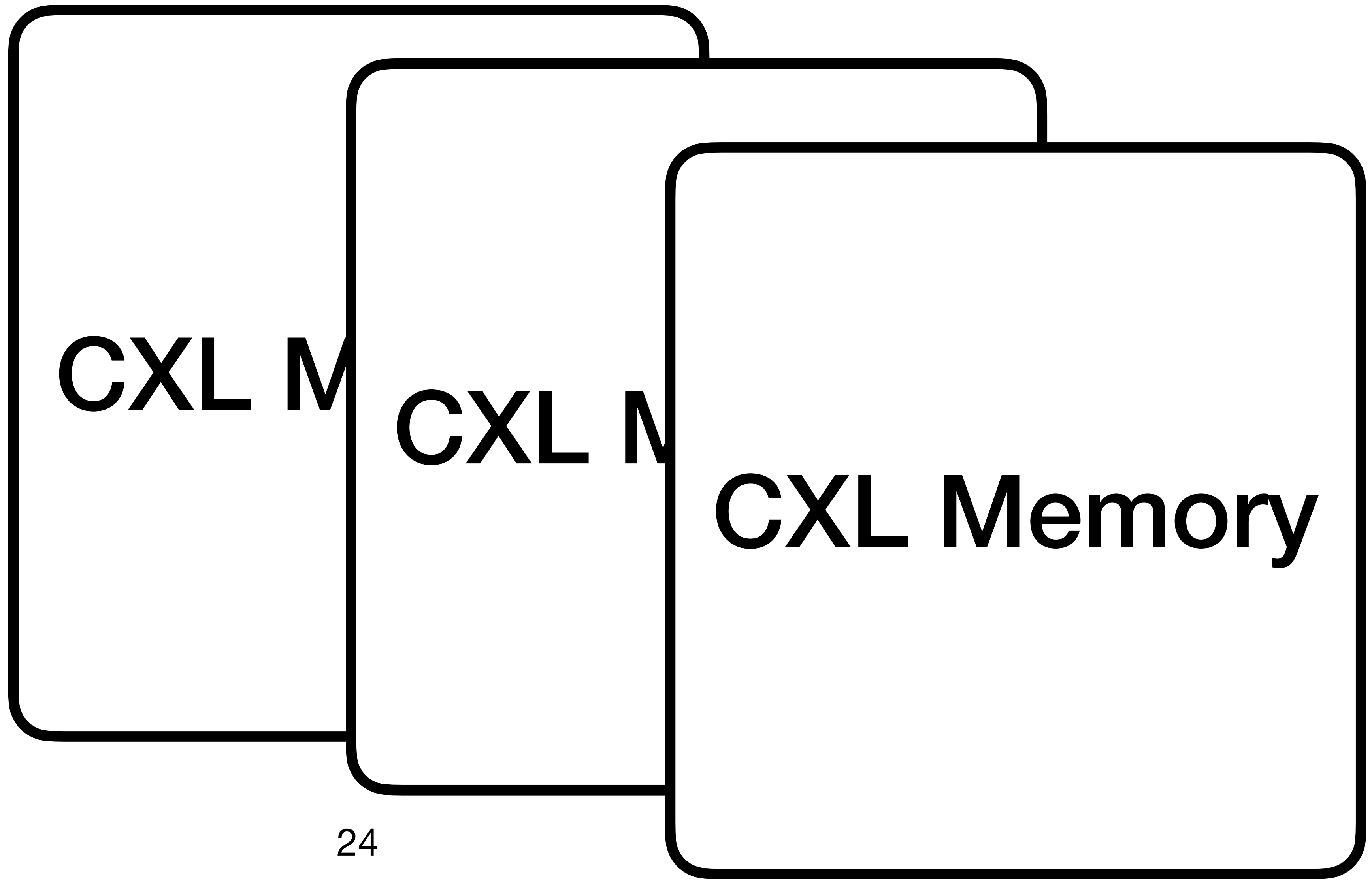
Access memory devices via I/O



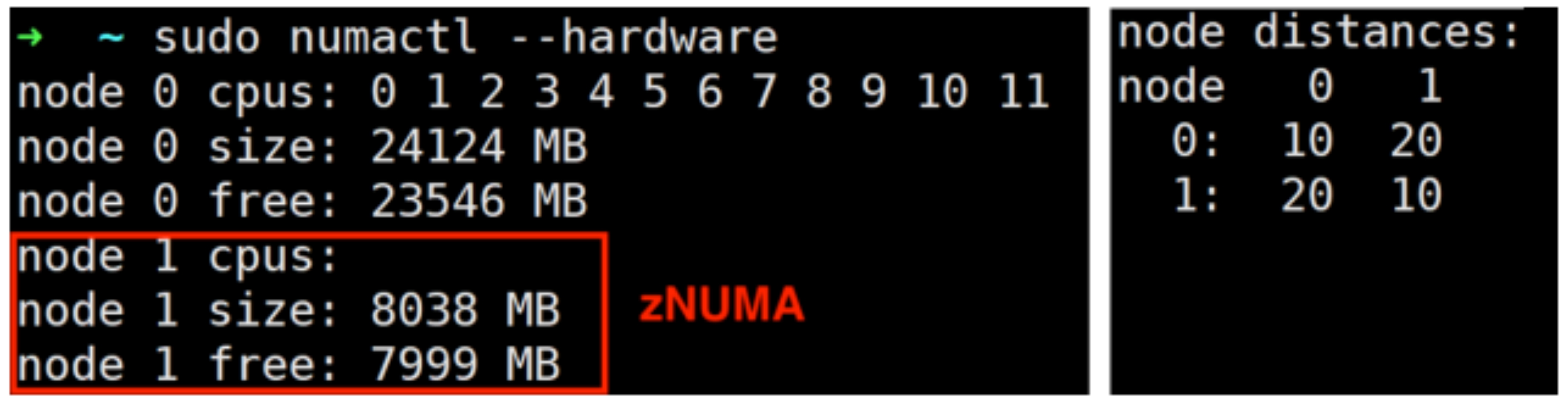
Limited by the number of DIMM slots!



Access memory devices via I/O



PCIe (+/- 90ns)

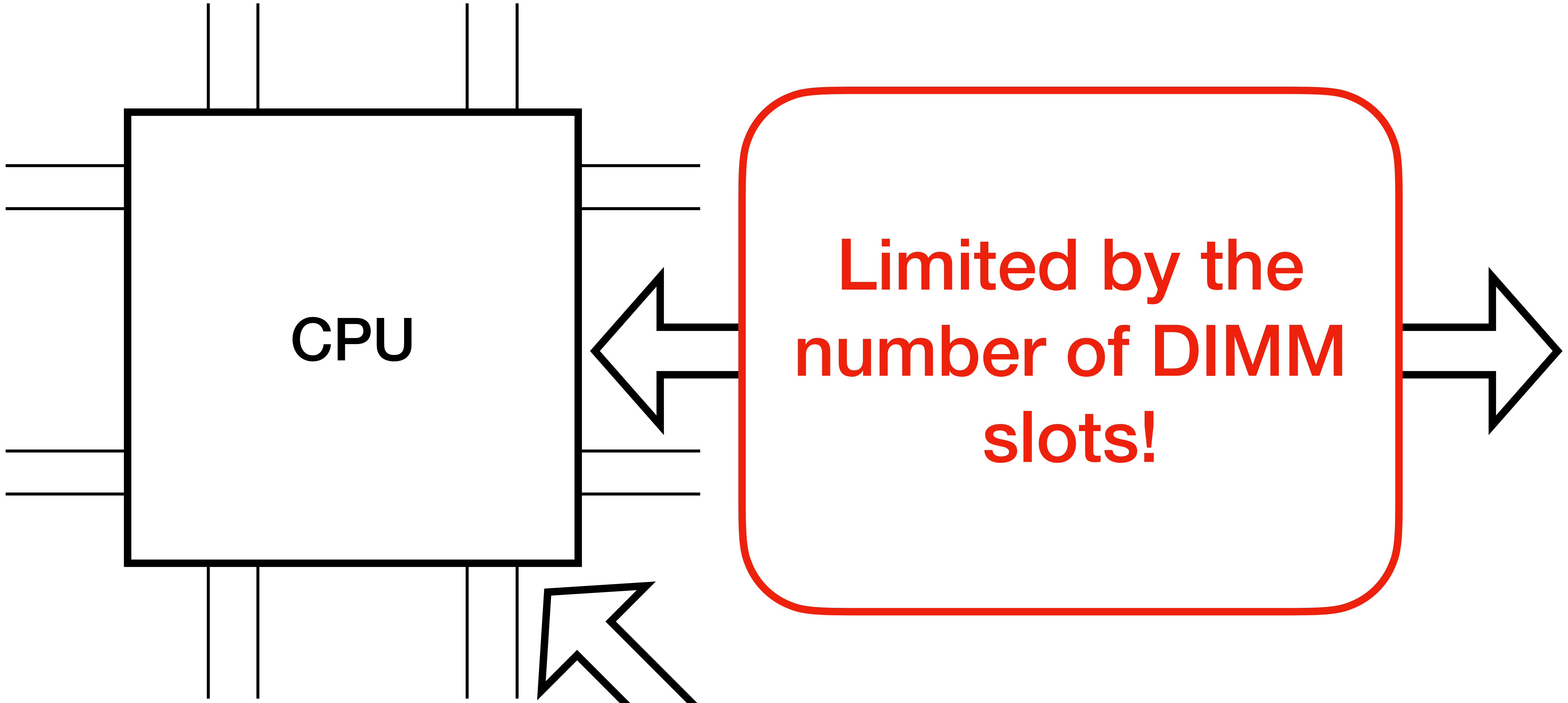


```
➜  ~ sudo numactl --hardware
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11
node 0 size: 24124 MB
node 0 free: 23546 MB
node 1 cpus:
node 1 size: 8038 MB
node 1 free: 7999 MB
```

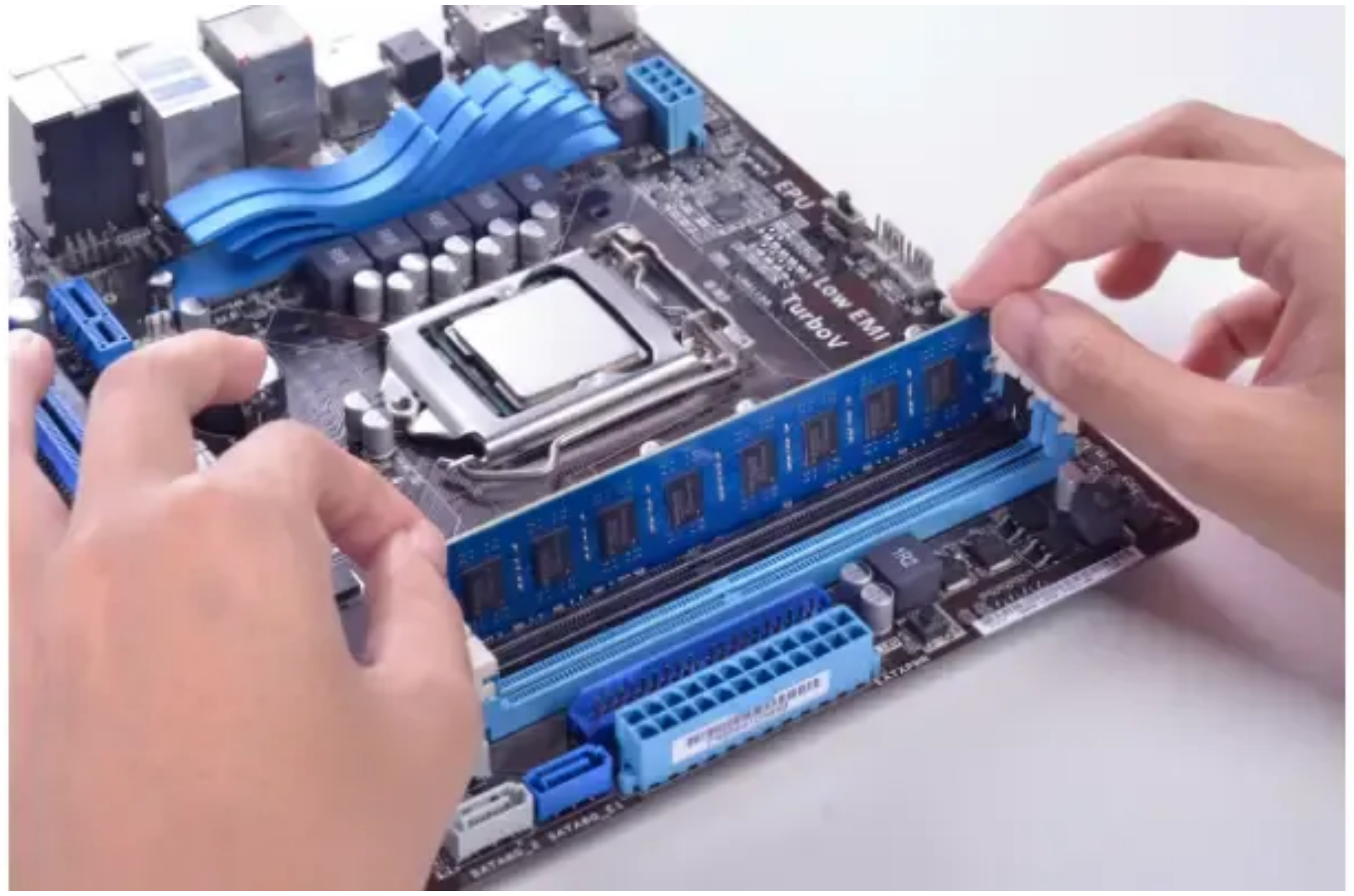
zNUMA

```
node distances:
node   0   1
  0:  10  20
  1:  20  10
```

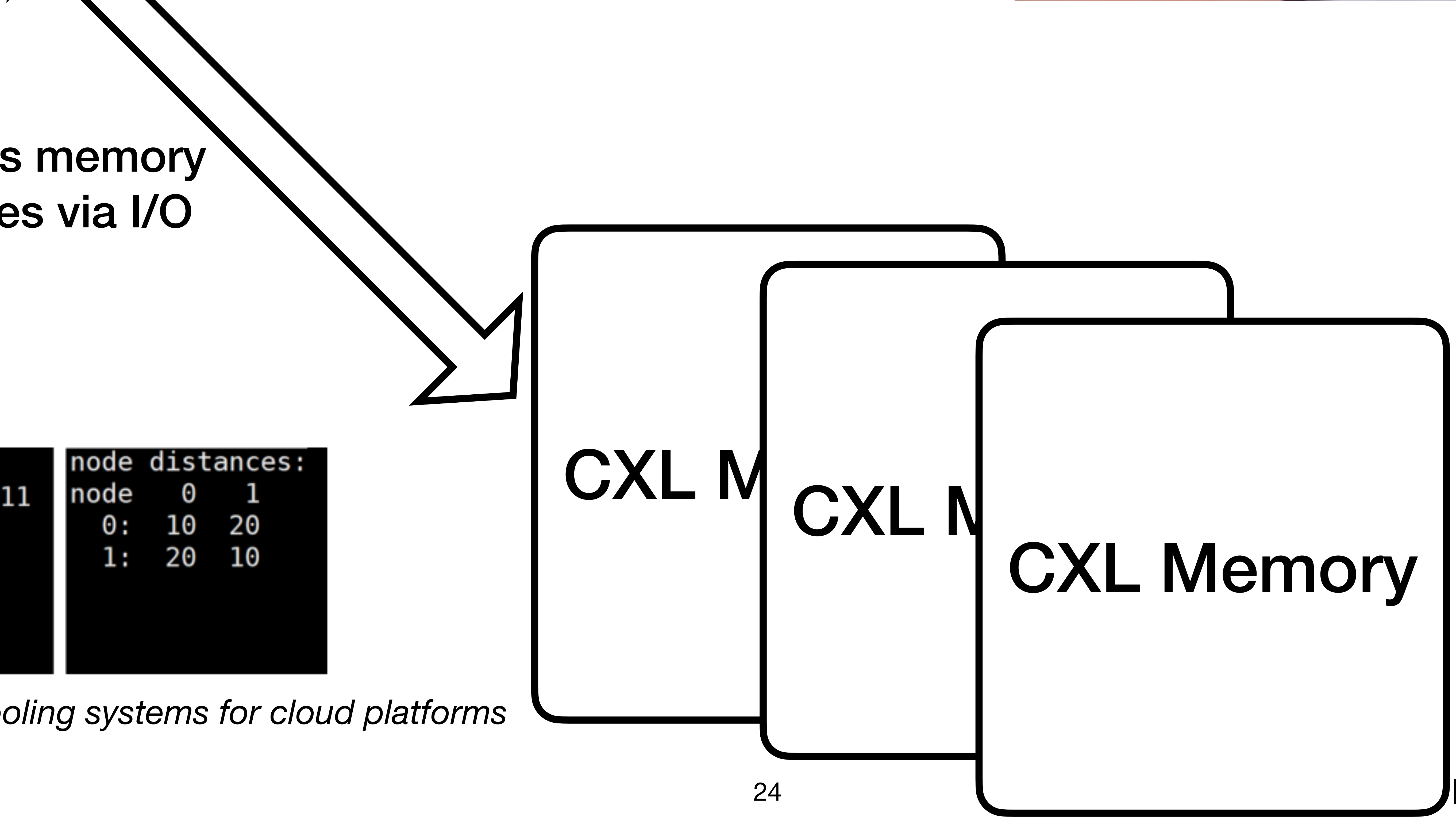
Li, et al. *Pond: CXL-based memory pooling systems for cloud platforms*



Limited by the number of DIMM slots!

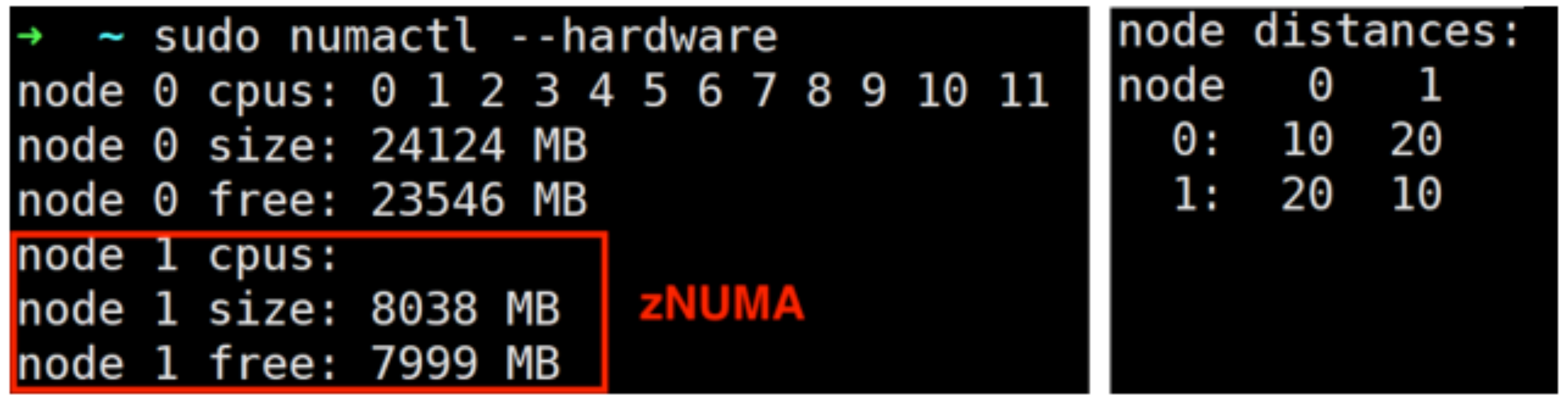


Access memory devices via I/O



- ✓ Scalable capacity
- ✓ Increased potential bandwidth
- ✗ No good programming model (yet)!

PCIe (+/- 90ns)



```
➜  ~ sudo numactl --hardware
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11
node 0 size: 24124 MB
node 0 free: 23546 MB
node 1 cpus:
node 1 size: 8038 MB
node 1 free: 7999 MB
```

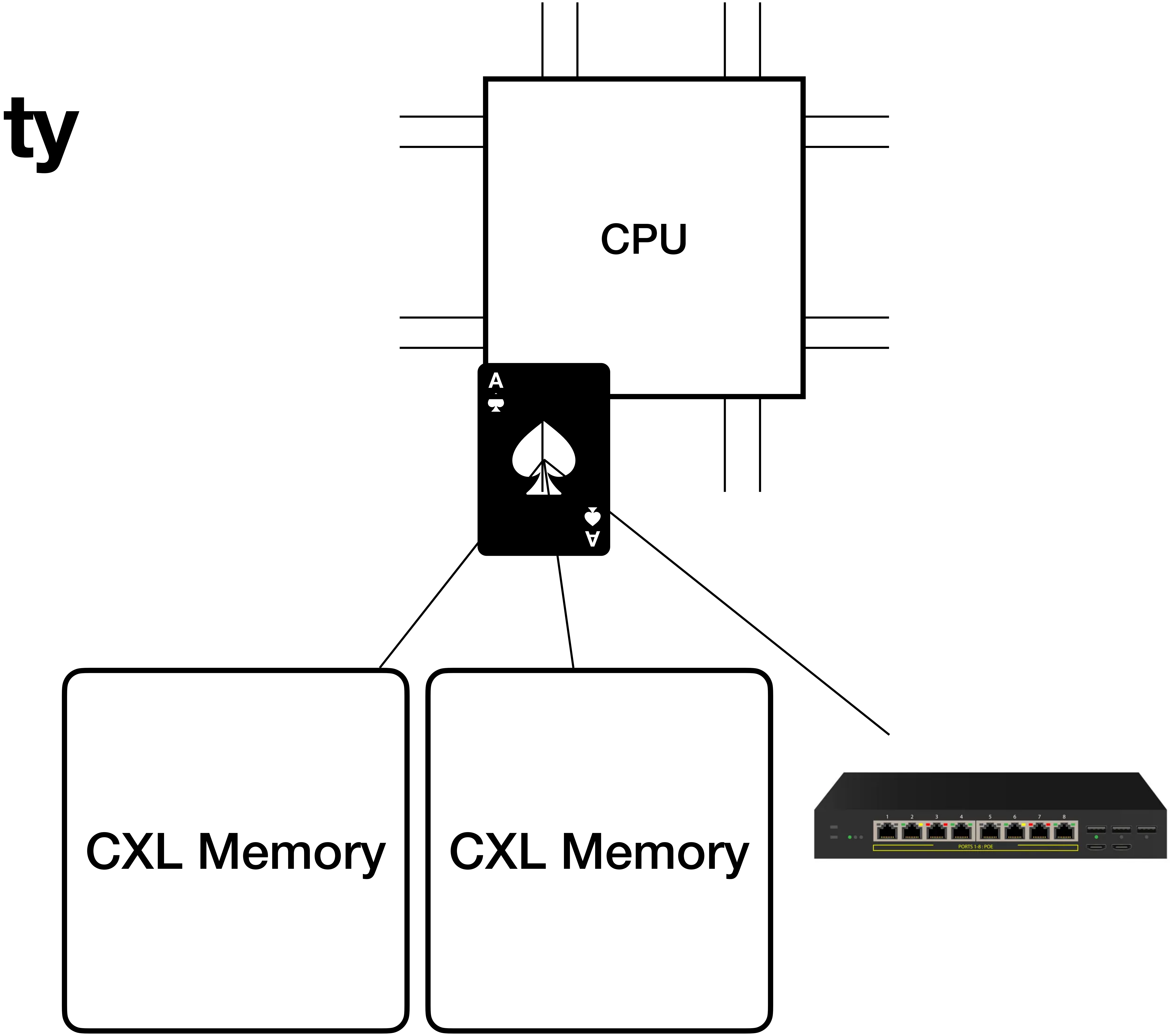
zNUMA

```
node distances:
node   0   1
  0:  10  20
  1:  20  10
```

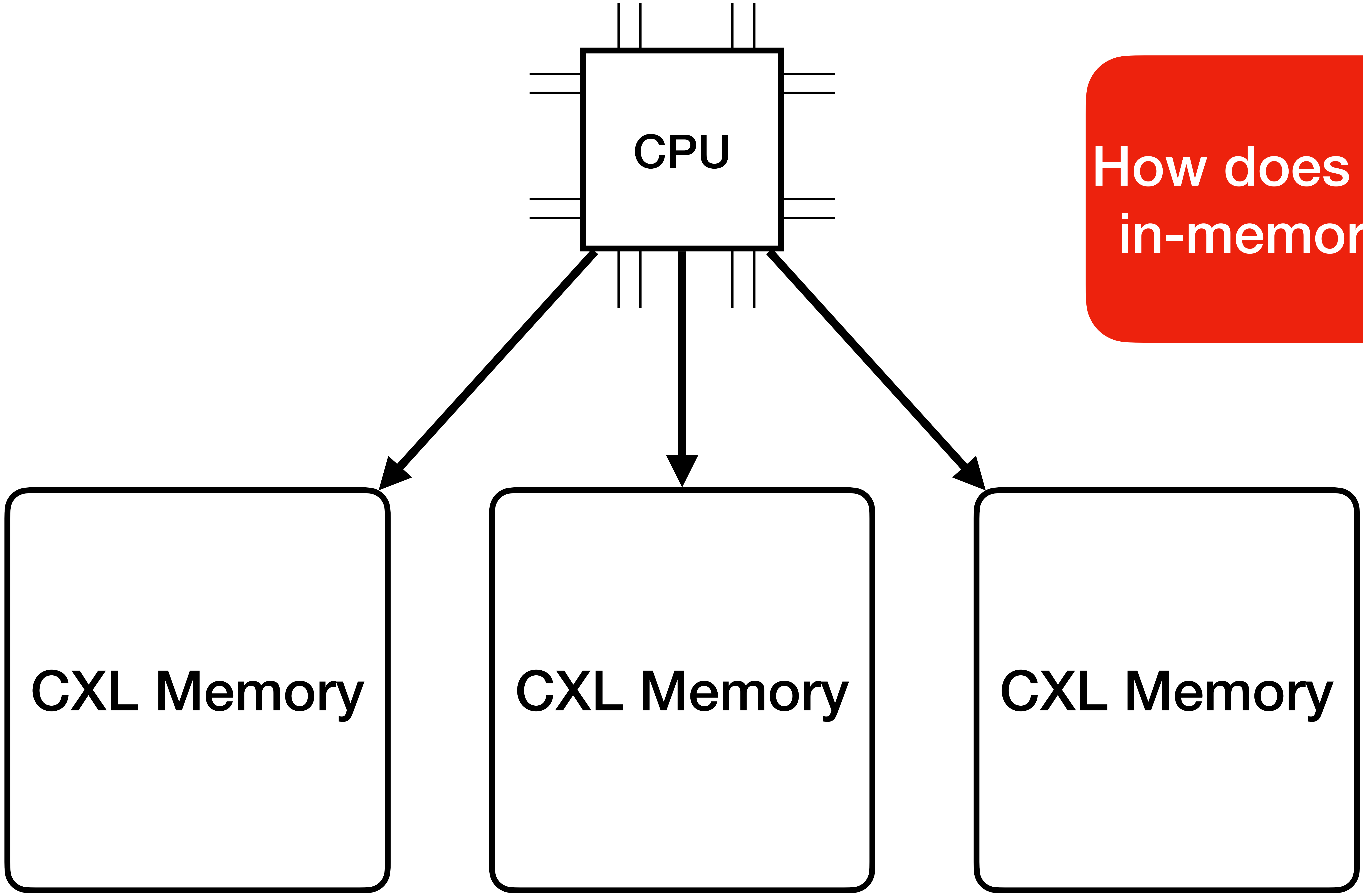
Li, et al. *Pond: CXL-based memory pooling systems for cloud platforms*

✓Scalable Capacity

- Cheaper than buying new devices
- Doesn't require inter-device coordination
- Extra hardware for computer architects to mess around with :-)



✓ Increased Potential Bandwidth



How does this impact in-memory locality?

✗ No good programming model (yet)!

Applications need to tolerate partial failure!

Is this shared memory consistency or a distributed system?

In-memory fault tolerance?

How to recover this device?

0x00 - 0xN
CXL Memory
0xN - 0x2N
CXL Memory
0x2N - 0x3N
CXL Memory

BROWN

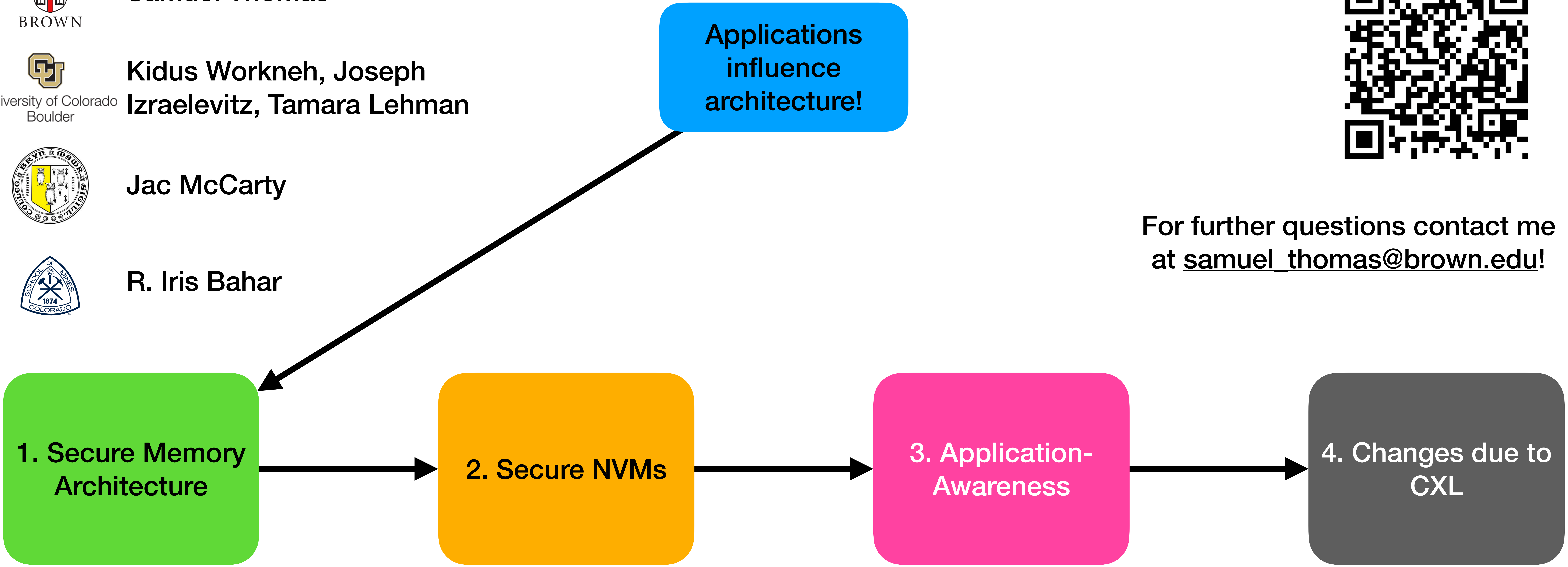
Samuel Thomas

University of Colorado Boulder

Kidus Workneh, Joseph Izraelevitz, Tamara Lehman

Jac McCarty

R. Iris Bahar



Applications influence architecture!

1. Secure Memory Architecture

2. Secure NVMs

3. Application-Awareness

4. Changes due to CXL

Data in memory is protected by a Merkle Tree

Merkle tree nodes require explicit persistence from metadata caches

Our work demonstrates how architects can leverage application behavior to influence architecture

CXL poses new requirements on applications, which will influence architecture

For further questions contact me at samuel_thomas@brown.edu!