MICRO 2026

Decoupled-DSM

A Scalable Secure Trusted Execution Environment for Multi-Host CXL Disaggregated Shared Memory

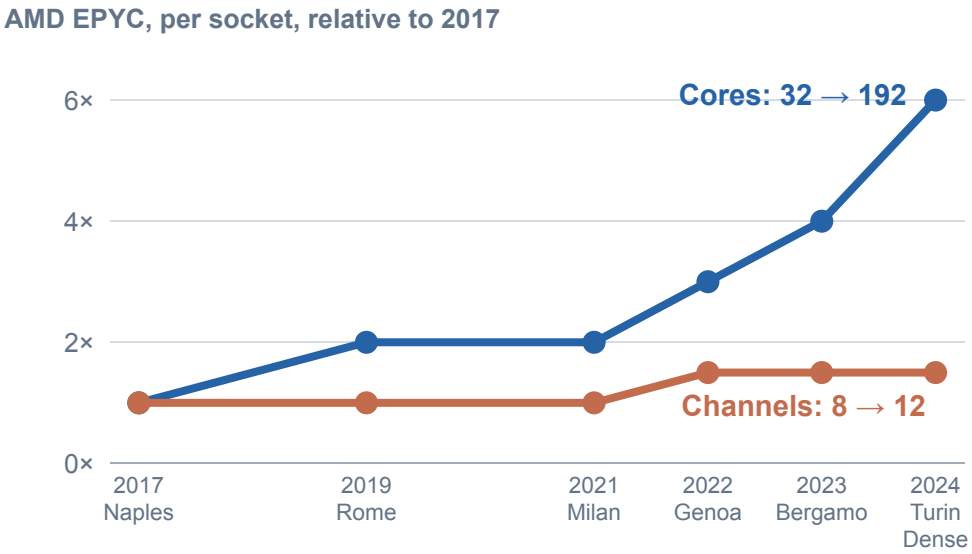
Yuxin Yang[‡], Yuanchao Xu[‡]

[‡]University of California, Santa Cruz



Memory capacity is the bottleneck

Cores keep multiplying, but the memory a server can attach does not

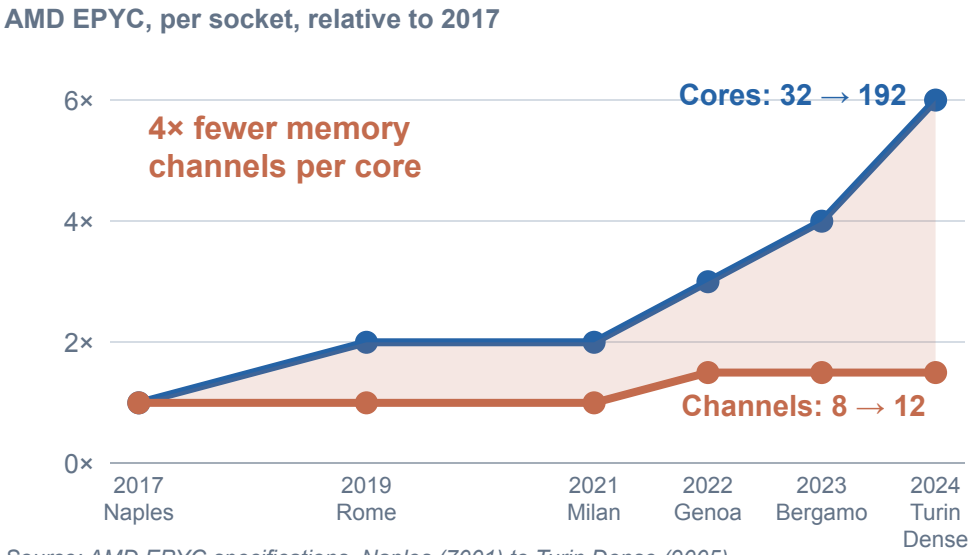


6× more cores per socket (2017 → 2024)

1.5× more memory channels (DIMM slots)

Memory capacity is the bottleneck

Cores keep multiplying, but the memory a server can attach does not



Source: AMD EPYC specifications, Naples (7001) to Turin Dense (9005)

6× more cores per socket (2017 → 2024)

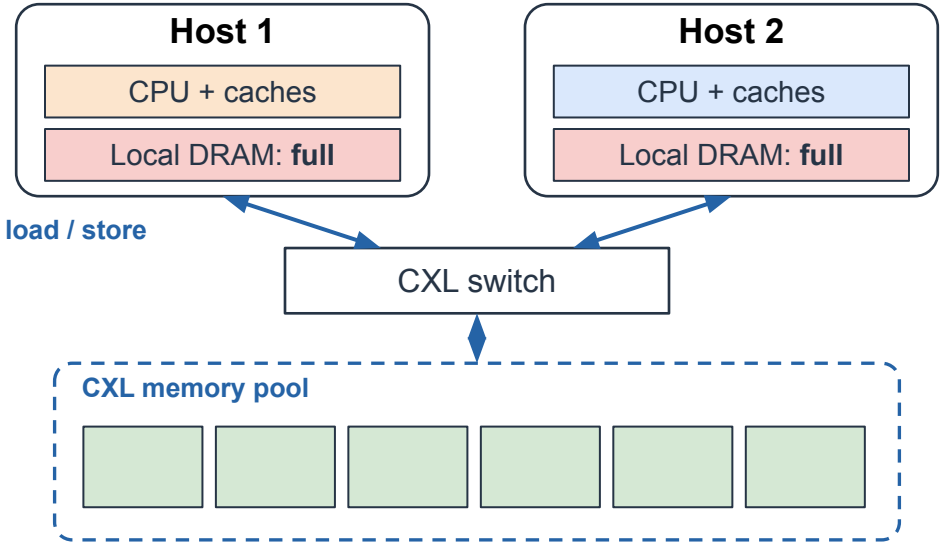
1.5× more memory channels (DIMM slots)

DRAM can be ~50% of server cost and up to 25% gets stranded
Pond, ASPLOS '23

Servers need more memory capacity than local DIMMs give

Memory is moving to shared CXL pools

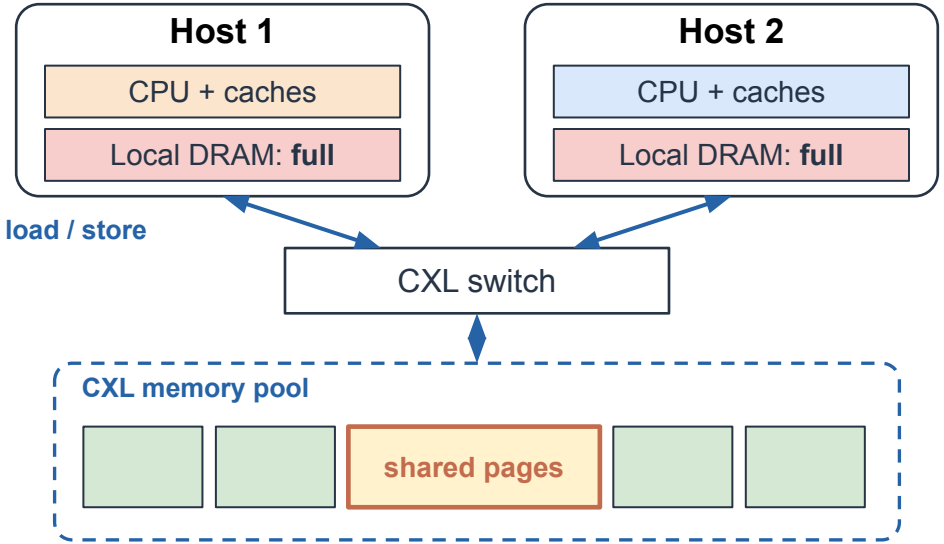
Trend in cloud data centers: CXL memory pools shared by many hosts



Expand memory over CXL with load/store

Memory is moving to shared CXL pools

Trend in cloud data centers: CXL memory pools shared by many hosts

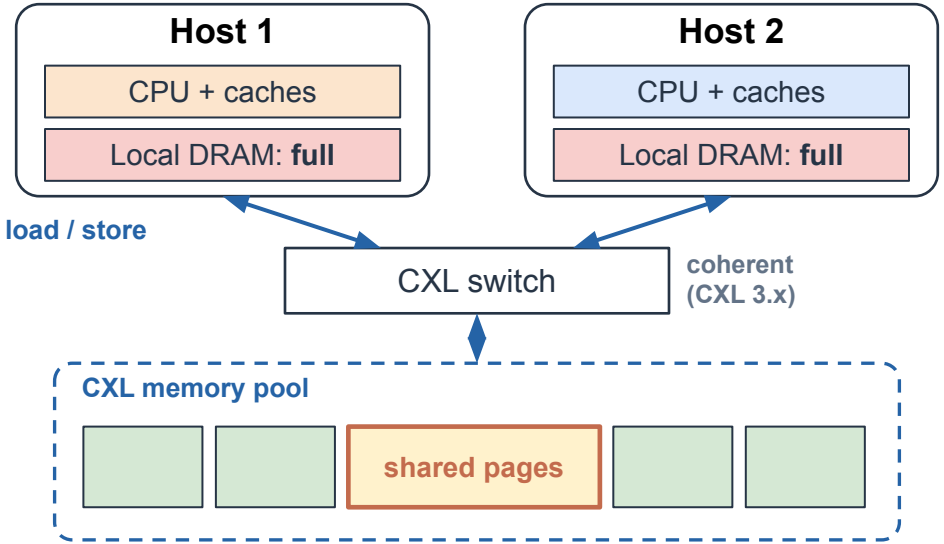


Expand memory over CXL with load/store

Share the same pages in place, no copies

Memory is moving to shared CXL pools

Trend in cloud data centers: CXL memory pools shared by many hosts



Expand memory over CXL with load/store

Share the same pages in place, no copies

Coherent caches in hardware

Cloud memory is moving from per-host DRAM to shared pools

Public clouds need hardware isolation

Recent attacks in the news

THE HACKER NEWS · MAR 4, 2025 — VM escape

VMware Security Flaws Exploited in the Wild—Broadcom Releases Urgent Patches

THE REGISTER · DEC 10, 2024 — Memory aliasing

AMD secure VM tech undone by DRAM meddling

THE HACKER NEWS · OCT 14, 2025 — RMP corruption

RMPocalypse: Single 8-Byte Write Shatters AMD's SEV-SNP Confidential Computing

Public clouds need hardware isolation

Recent attacks in the news

THE HACKER NEWS · MAR 4, 2025 — VM escape

VMware Security Flaws Exploited in the Wild—Broadcom Releases Urgent Patches

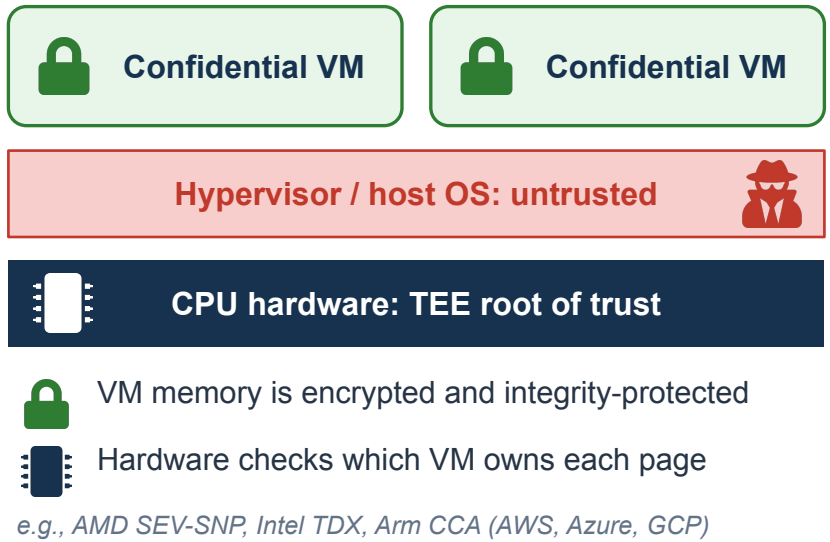
THE REGISTER · DEC 10, 2024 — Memory aliasing

AMD secure VM tech undone by DRAM meddling

THE HACKER NEWS · OCT 14, 2025 — RMP corruption

RMPocalypse: Single 8-Byte Write Shatters AMD's SEV-SNP Confidential Computing

TEEs: hardware-isolated VMs



VM memory is encrypted and integrity-protected

Hardware checks which VM owns each page

e.g., AMD SEV-SNP, Intel TDX, Arm CCA (AWS, Azure, GCP)

Public clouds need hardware isolation

Recent attacks in the news

THE HACKER NEWS · MAR 4, 2025 — VM escape

VMware Security Flaws Exploited in the Wild—Broadcom Releases Urgent Patches

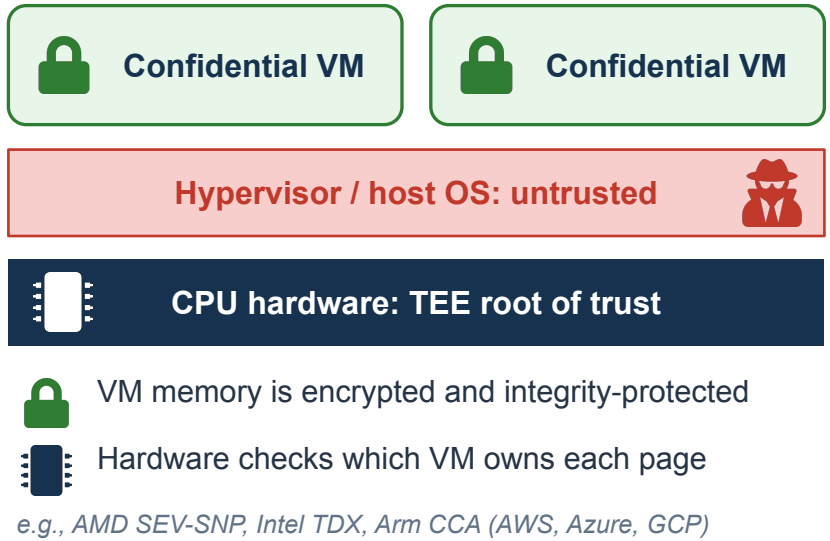
THE REGISTER · DEC 10, 2024 — Memory aliasing

AMD secure VM tech undone by DRAM meddling

THE HACKER NEWS · OCT 14, 2025 — RMP corruption

RMPocalypse: Single 8-Byte Write Shatters AMD's SEV-SNP Confidential Computing

TEEs: hardware-isolated VMs



VM memory is encrypted and integrity-protected

Hardware checks which VM owns each page

e.g., AMD SEV-SNP, Intel TDX, Arm CCA (AWS, Azure, GCP)

Can existing TEEs still protect memory shared across hosts?

Outline

1	**Background**	TEE memory isolation, using AMD SEV-SNP
2	**Motivating example**	The cross-host isolation gap
3	**Key idea**	Separate permission management from memory accesses
4	**Decoupled-DSM**	System calls, page faults, and memory accesses
5	**Evaluation**	SPEC, scaling, stress tests, and Redis

AMD SEV-SNP checks page ownership

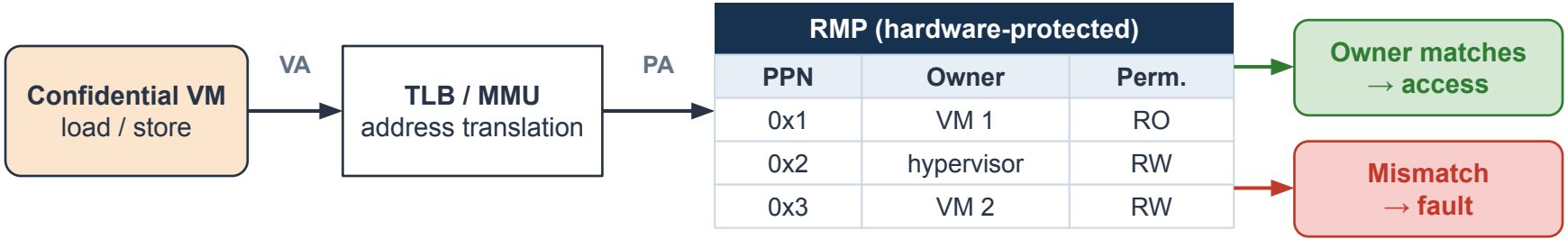
Background: hardware checks each translation against the RMP



RMP (hardware-protected)		
PPN	**Owner**	**Perm.**
0x1	VM 1	RO
0x2	hypervisor	RW
0x3	VM 2	RW

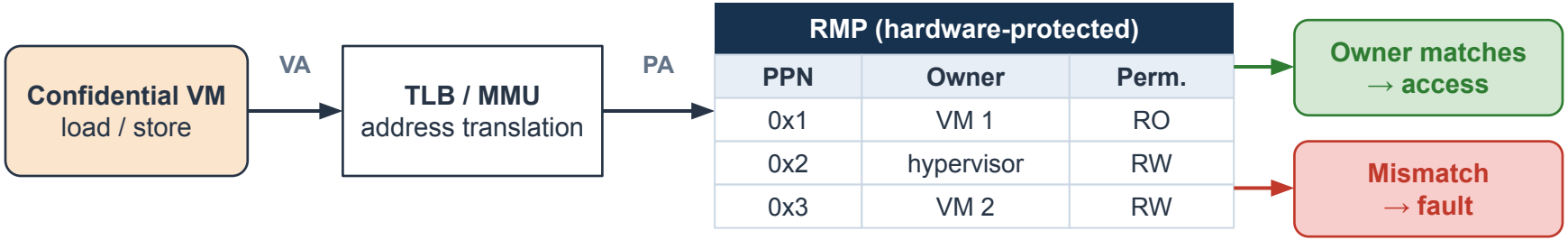
AMD SEV-SNP checks page ownership

Background: hardware checks each translation against the RMP



AMD SEV-SNP checks page ownership

Background: hardware checks each translation against the RMP

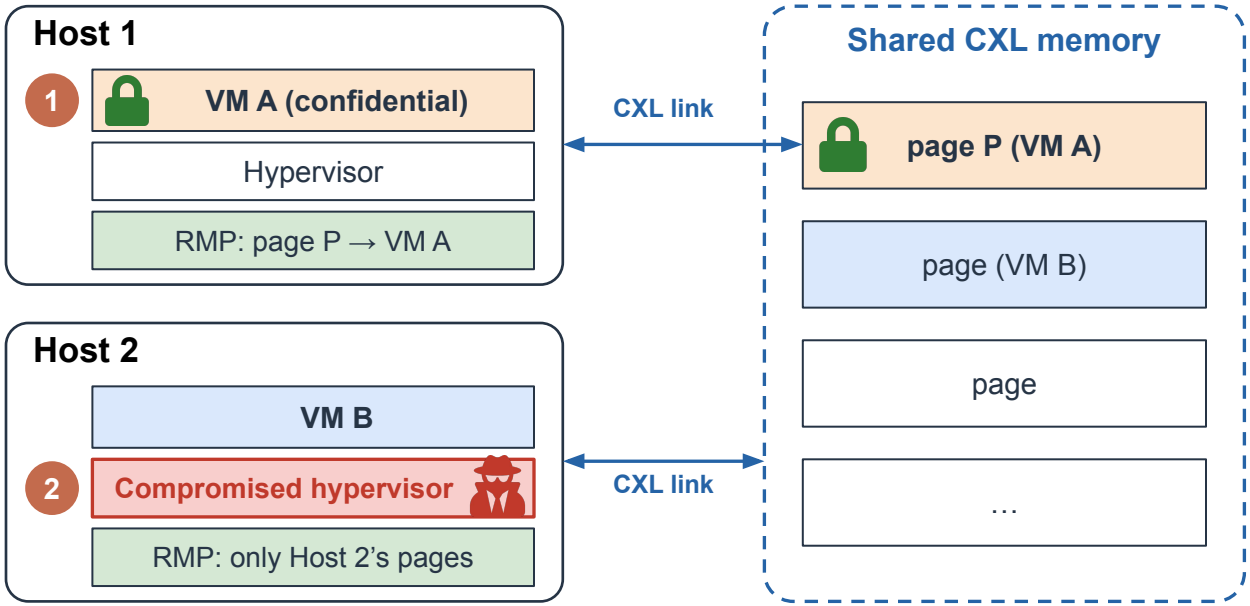


The hypervisor **can** edit page tables, but **cannot** write the RMP

Intel TDX (PAMT) and Arm CCA (GPT) keep similar per-page metadata

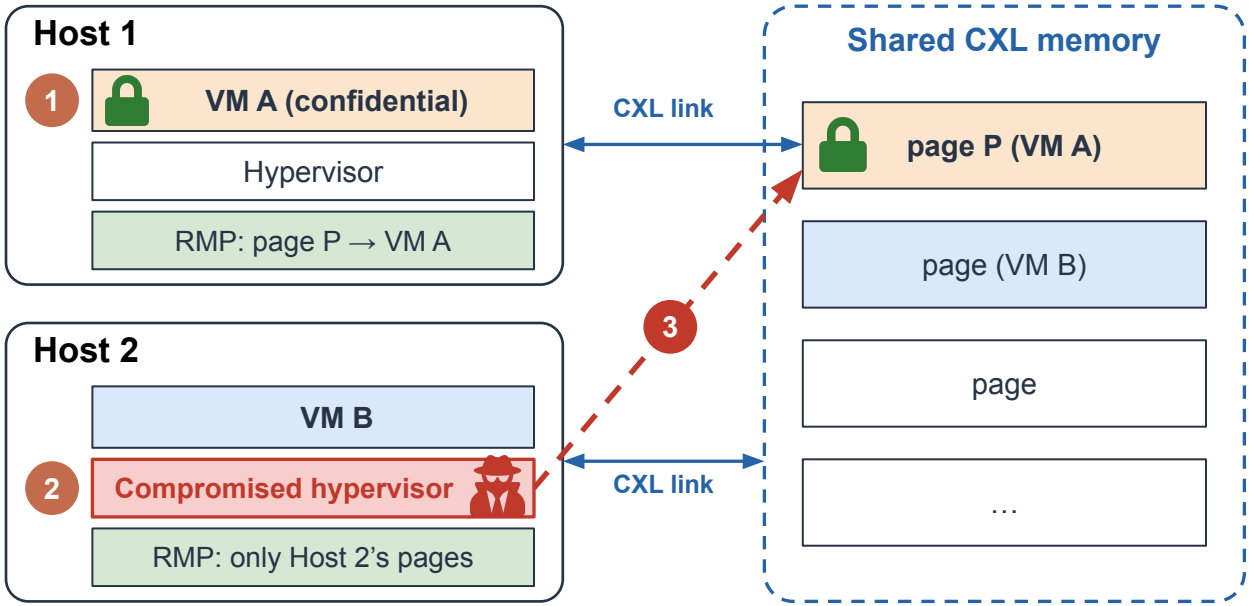
Each host's RMP tracks only pages owned on that host

Motivation: forging a mapping



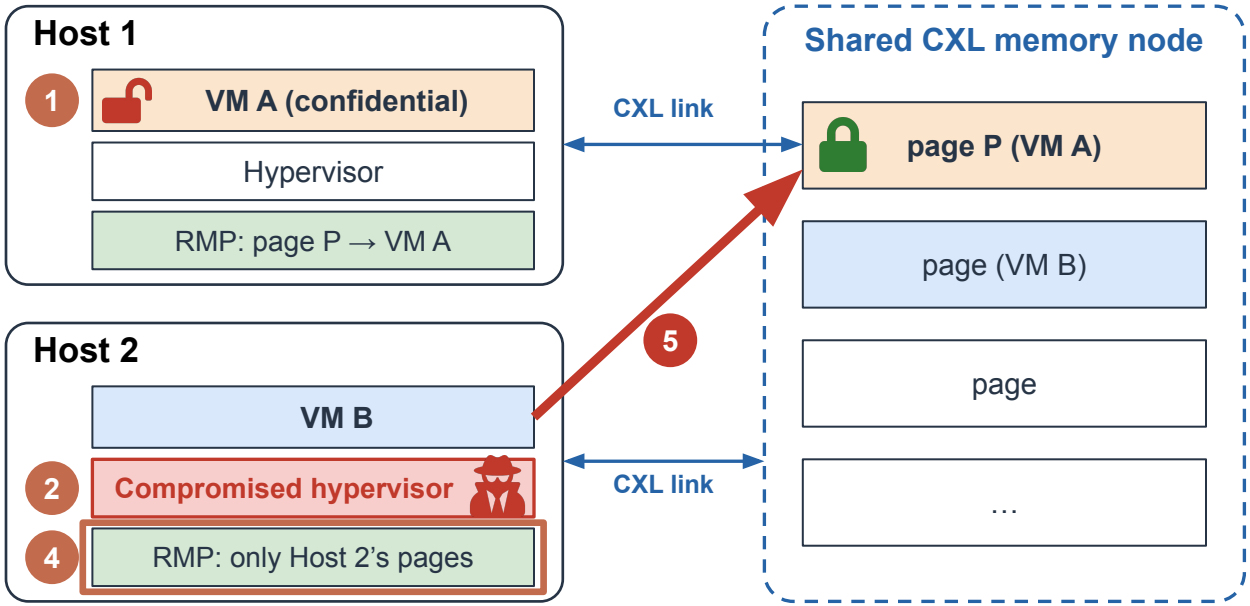
1. VM A on Host 1 owns page P in shared CXL memory
2. Host 2's hypervisor is compromised (standard TEE threat model)

Motivation: forging a mapping



1. VM A on Host 1 owns page P in shared CXL memory
2. Host 2's hypervisor is compromised (standard TEE threat model)
3. It writes a forged PTE that maps VM B to page P

Motivation: no host catches it



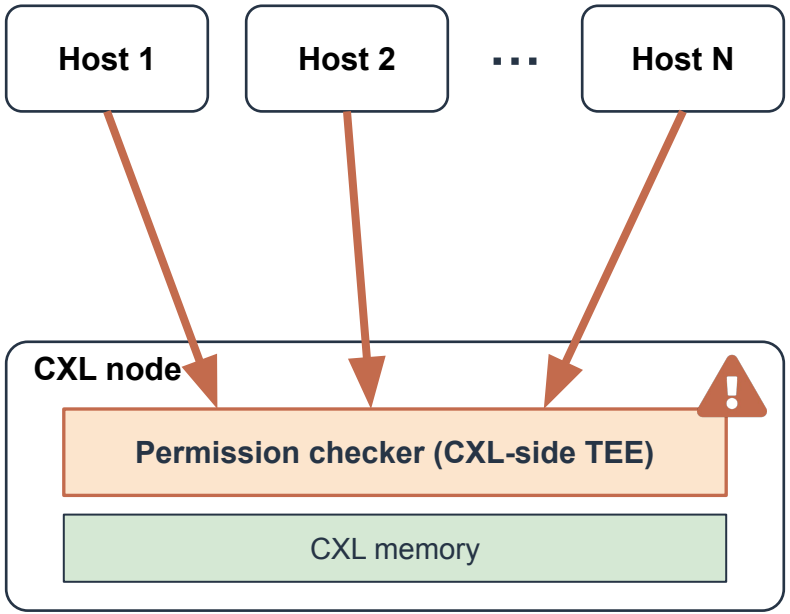
4 Host 2's RMP has no record of page P's owner, so it cannot reject it

5 VM B reads and writes VM A's page: no fault, no alarm

i CXL IDE does not help: it protects the link, not who may access a page

TEEs check ownership per host; nobody checks across hosts

Centralized checks become a bottleneck



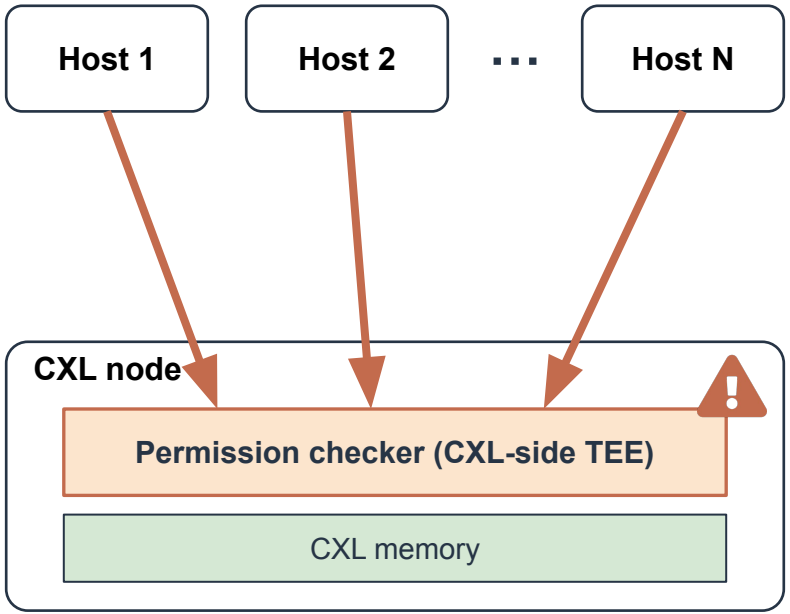
Centralized-DSM (strawman)

One CXL-side TEE holds all cross-host permissions

and checks every access from every host

More hosts → one shared checking path

Centralized checks become a bottleneck



Centralized-DSM

One CXL-side TEE holds all cross-host permissions

and checks every access from every host

More hosts → one shared checking path

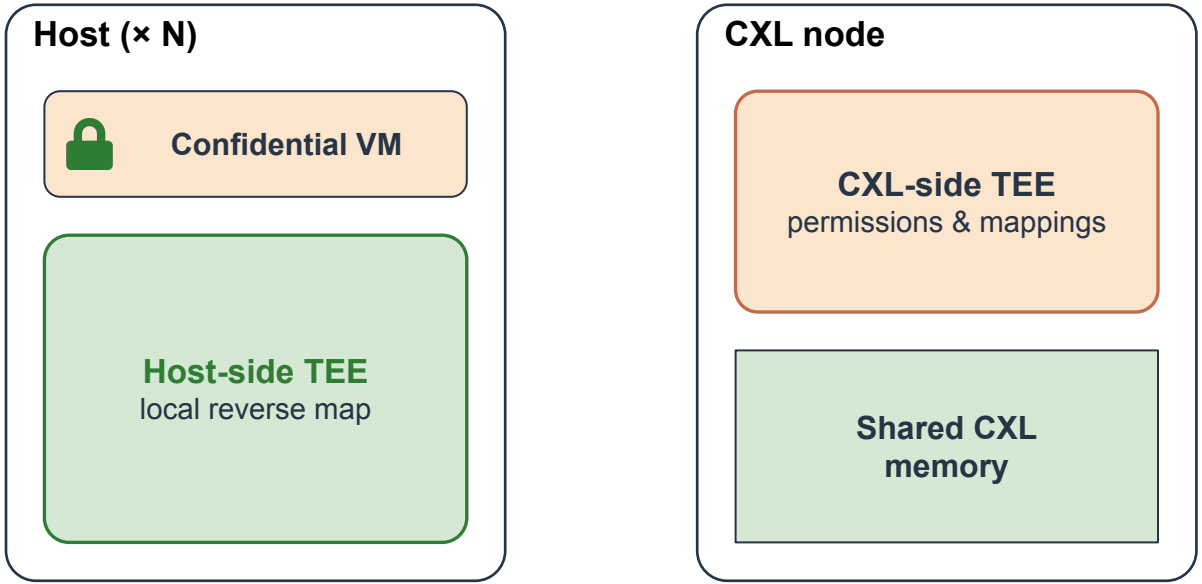
Key observation

System calls change permissions; **page faults** establish authorized mappings. Both are rare.

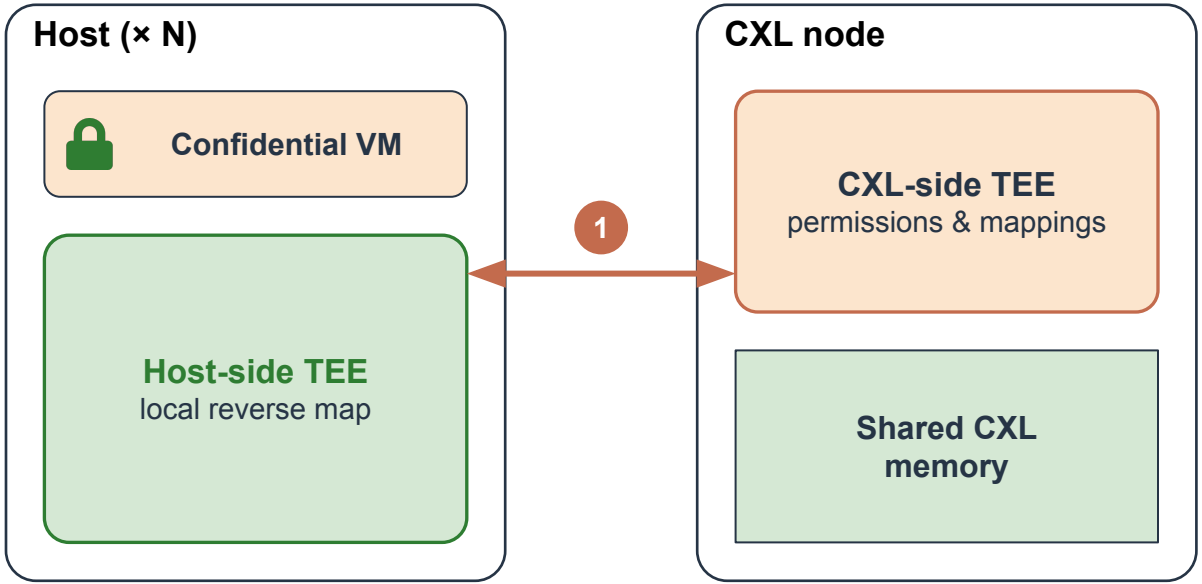
Memory accesses: every load/store; must stay fast

Key idea: global permissions and mappings, local access checks

Decoupled-DSM: global policy, local checks

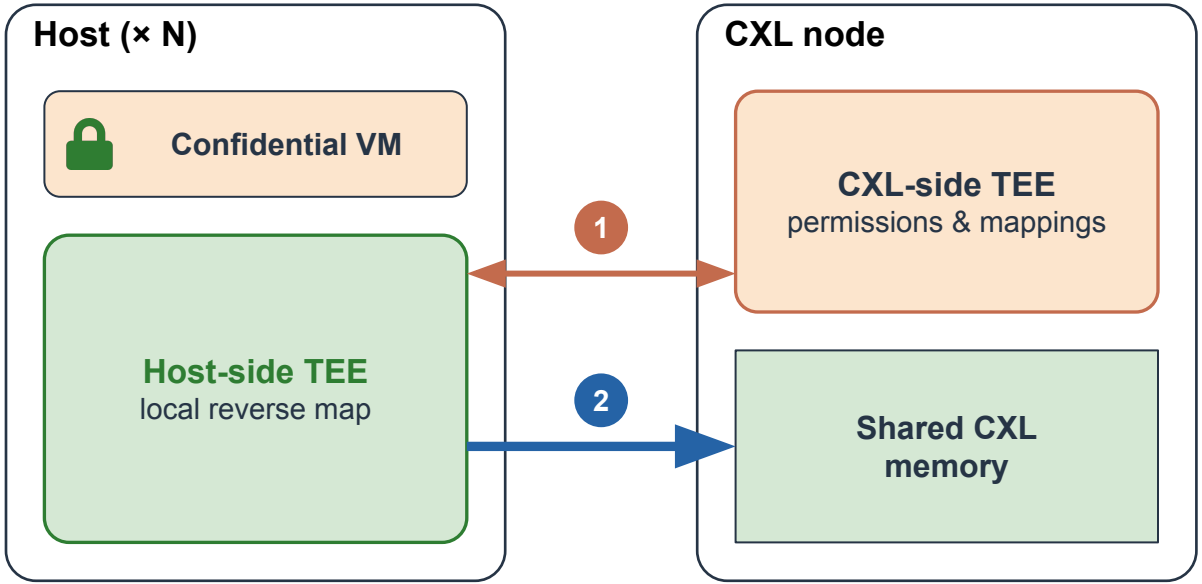


Decoupled-DSM: global policy, local checks



1 **System calls** update permissions, **page faults** set up mappings; both rare, via the CXL-side TEE

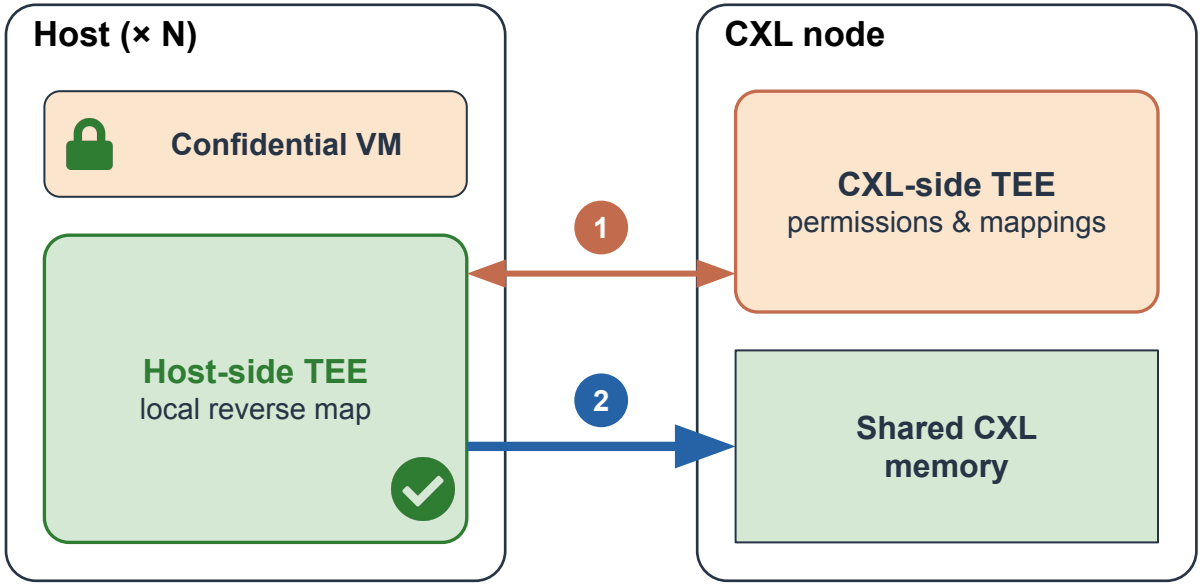
Decoupled-DSM: global policy, local checks



1. **System calls** update permissions, **page faults** set up mappings; both rare, via the CXL-side TEE

2. **Memory accesses** (frequent): checked locally, then sent directly

Decoupled-DSM: global policy, local checks



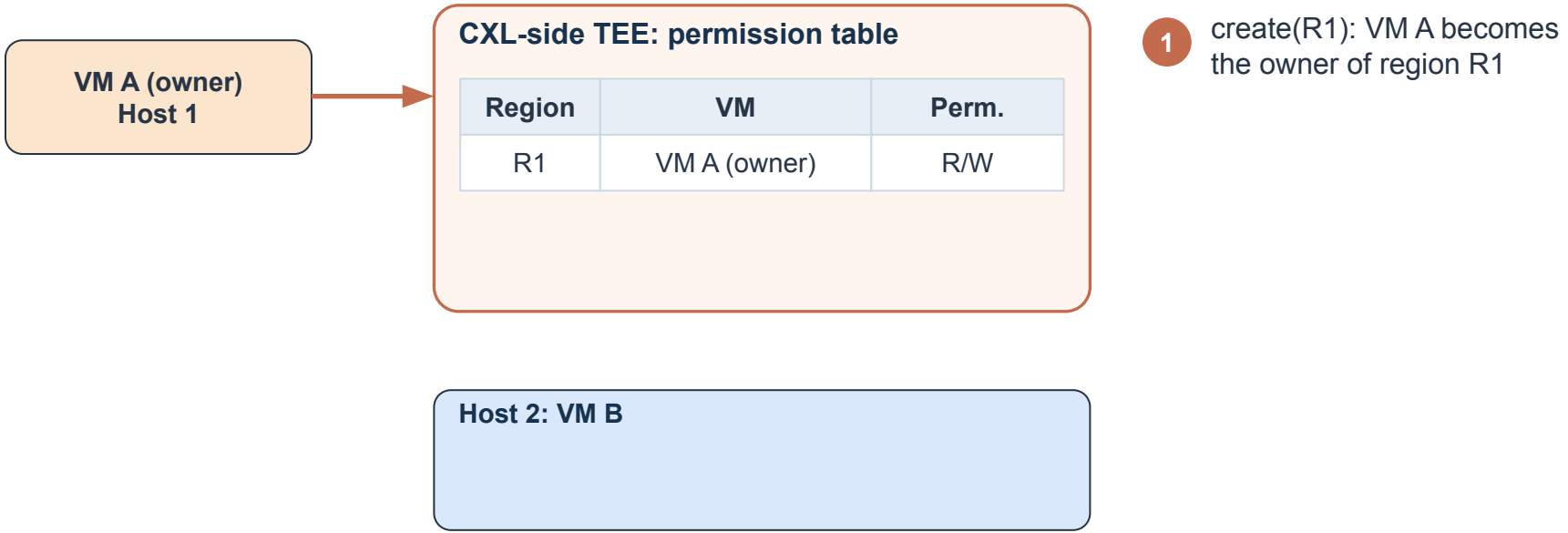
1. **System calls** update permissions, **page faults** set up mappings; both rare, via the CXL-side TEE

2. **Memory accesses** (frequent): checked locally, then sent directly

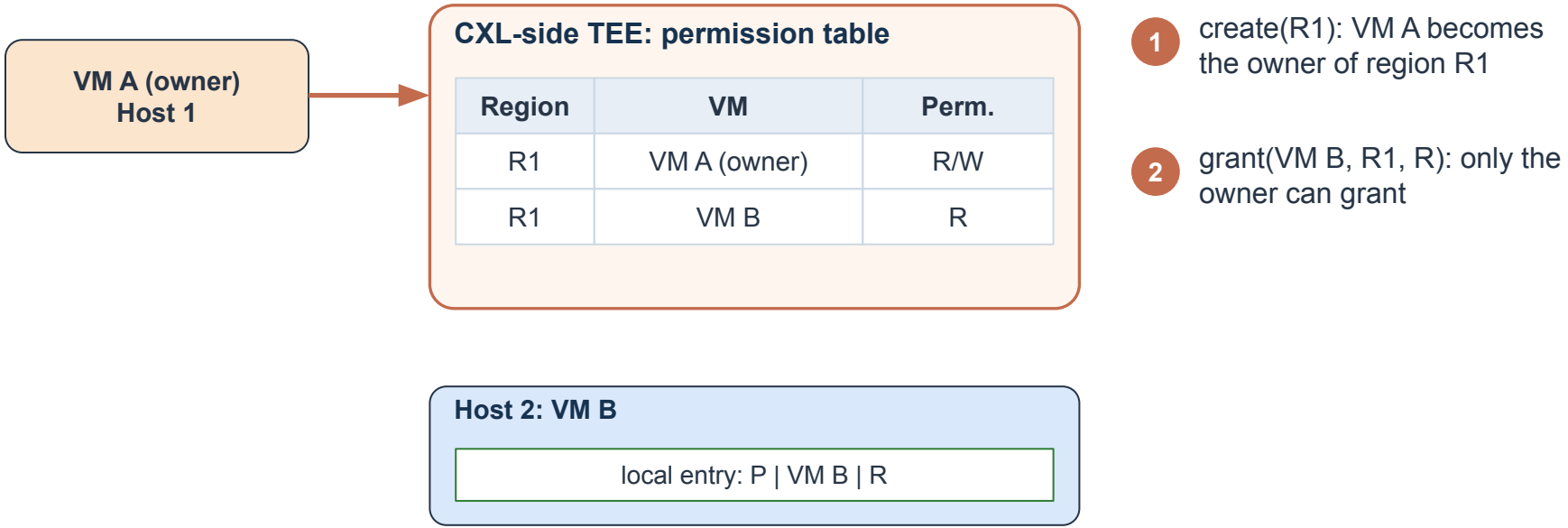
✓ The local check is the RMP-style check CPU TEEs already do

Infrequent operations go global; frequent accesses stay local

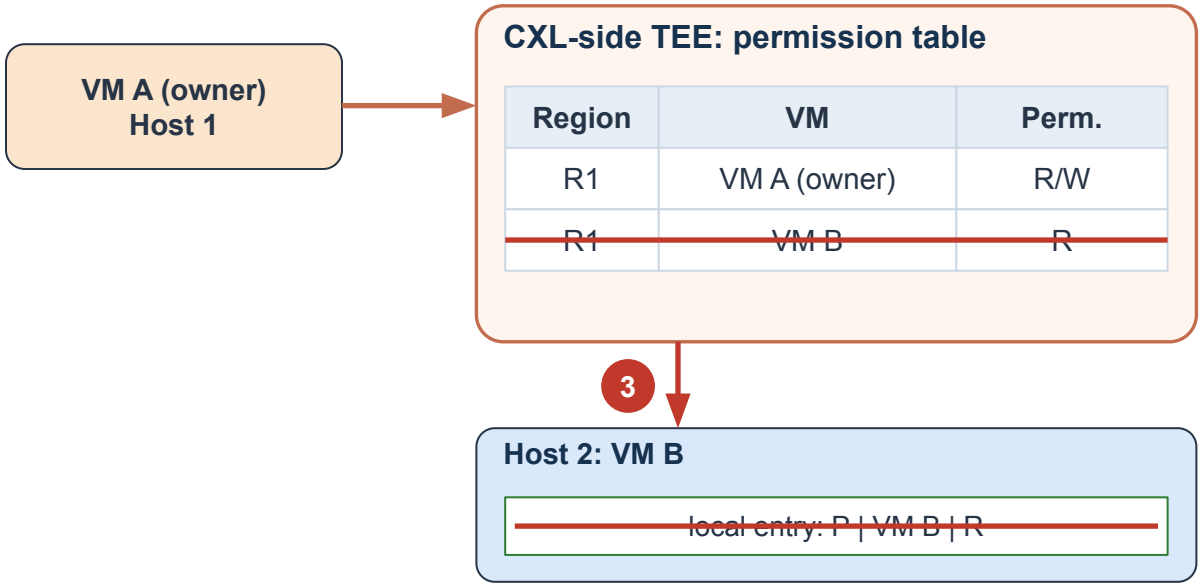
System calls update permissions



System calls update permissions

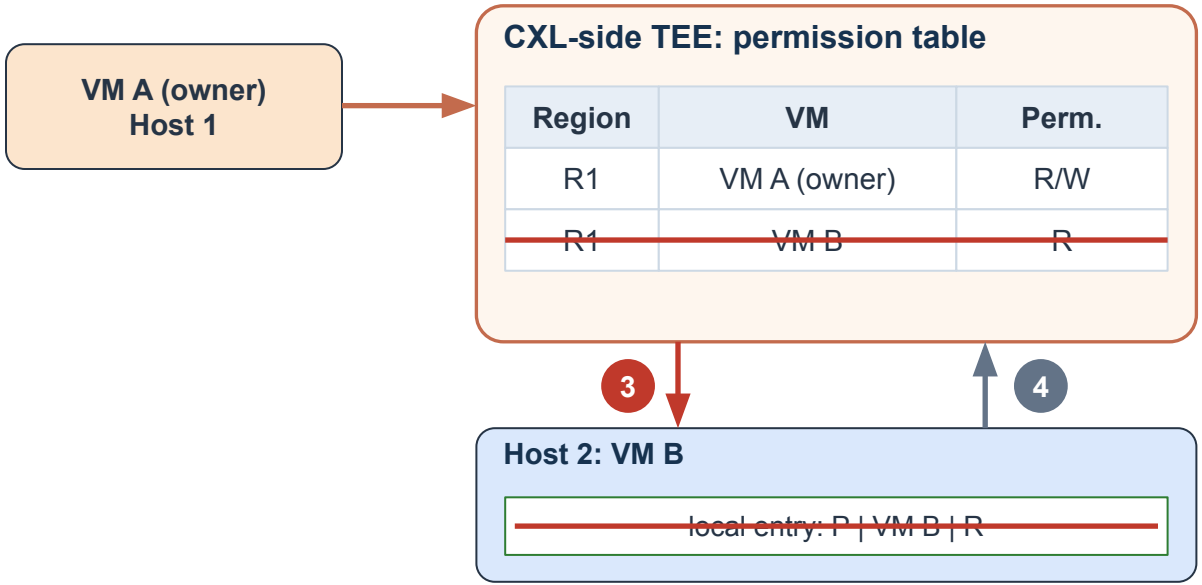


System calls update permissions



1. create(R1): VM A becomes the owner of region R1
2. grant(VM B, R1, R): only the owner can grant
3. revoke(VM B, R1): Host 2 drops its local entry

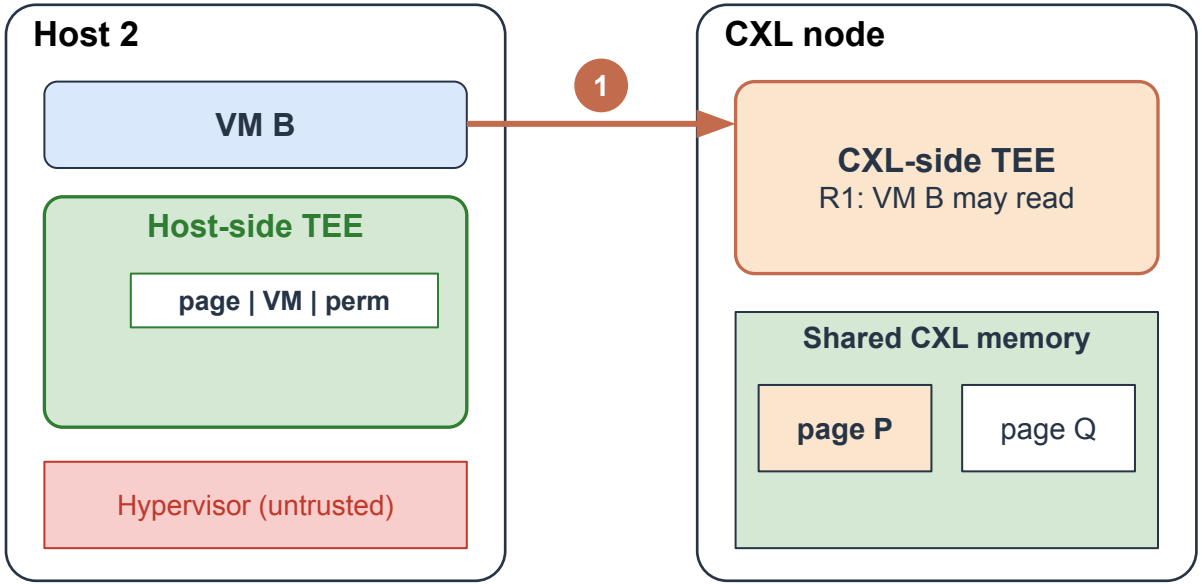
System calls update permissions



1. create(R1): VM A becomes the owner of region R1
2. grant(VM B, R1, R): only the owner can grant
3. revoke(VM B, R1): Host 2 drops its local entry
4. Host 2 acks; only then does the revoke complete

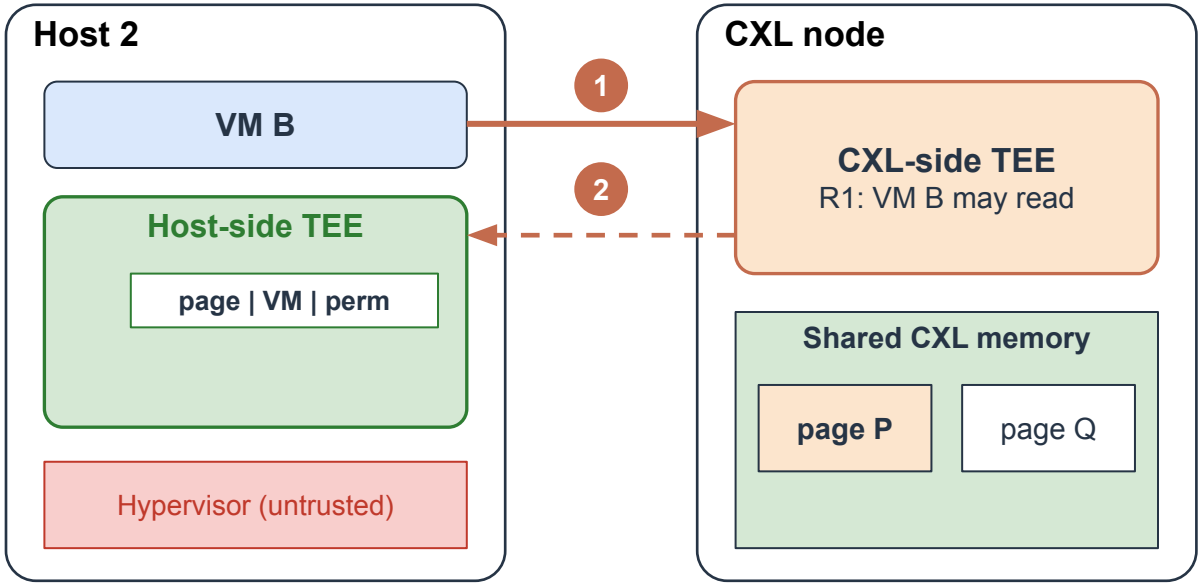
Permission changes are infrequent, so a global check here is cheap

Page faults establish authorized mappings



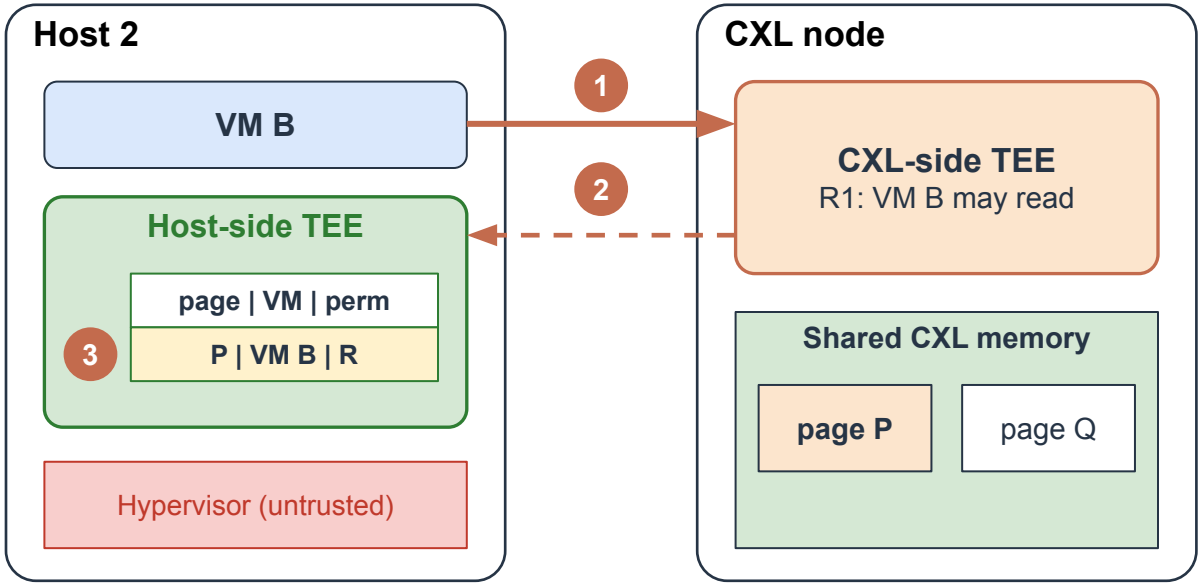
1 Page fault: request (R1, VM B, offset)

Page faults establish authorized mappings



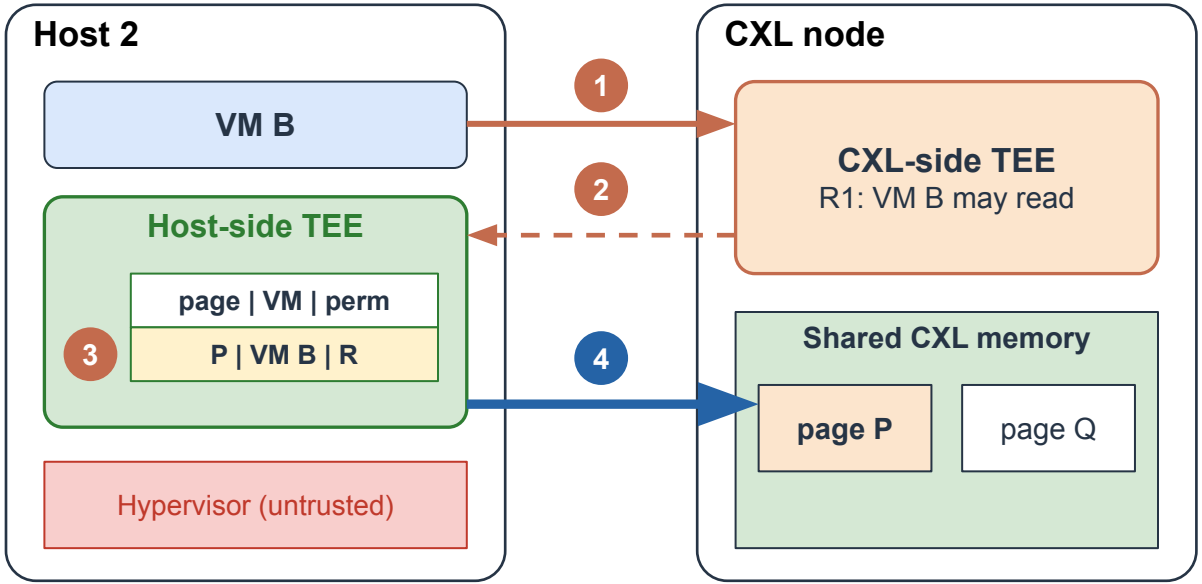
1. Page fault: request (R1, VM B, offset)
2. CXL-side TEE checks R1 → mapping to page P

Page faults establish authorized mappings



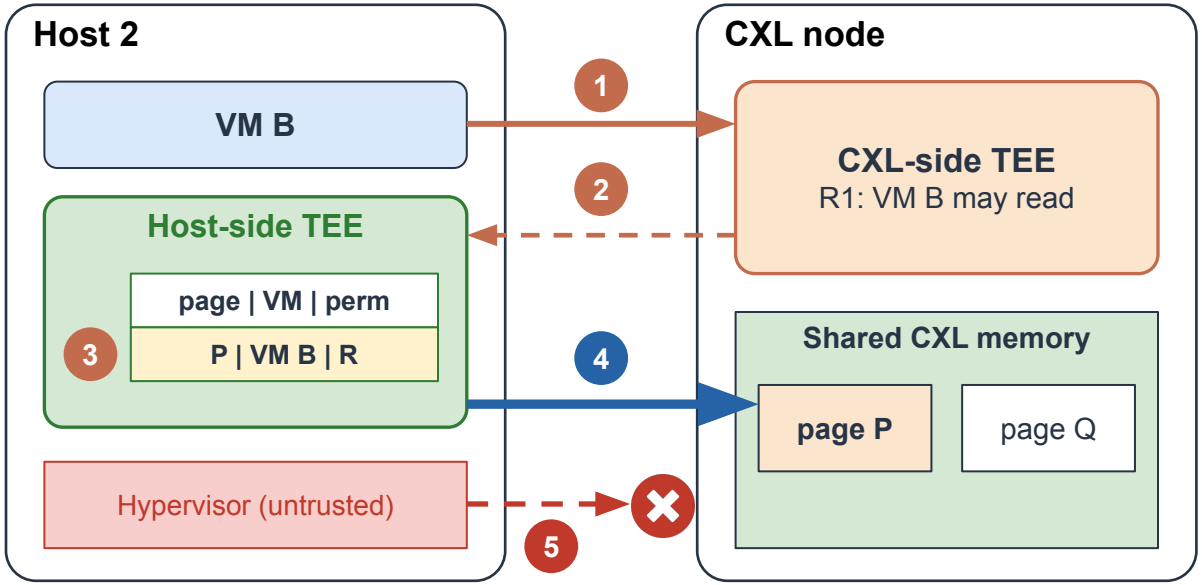
1. Page fault: request (R1, VM B, offset)
2. CXL-side TEE checks R1 → mapping to page P
3. Host-side TEE installs PTE + local entry

Page faults establish authorized mappings



1. Page fault: request (R1, VM B, offset)
2. CXL-side TEE checks R1 → mapping to page P
3. Host-side TEE installs PTE + local entry
4. Later accesses: local check, straight to memory

Page faults establish authorized mappings



1. Page fault: request (R1, VM B, offset)
2. CXL-side TEE checks R1 → mapping to page P
3. Host-side TEE installs PTE + local entry
4. Later accesses: local check, straight to memory
5. Forged PTE to page Q: no local entry → fault

A hypervisor can forge a PTE, but not a trusted local entry

More details in the paper

Region permission API

create, grant, revoke, mmap, munmap, destroy; owner-controlled sharing

Safe revocation

Hosts invalidate stale local state before the CXL side acknowledges

Shared-data encryption

Per-VM CXL IDE on the link; per-region keys inside the CXL device

Local metadata cache

4 KB reverse-map cache per host; misses fill from a trusted table

Security analysis

Forged mappings, VM identity spoofing, stale permissions

Hardware cost

108 KB on-chip storage and about 0.48 W at 8 hosts (45 nm model)

Plus sensitivity studies: CXL latency and cache size

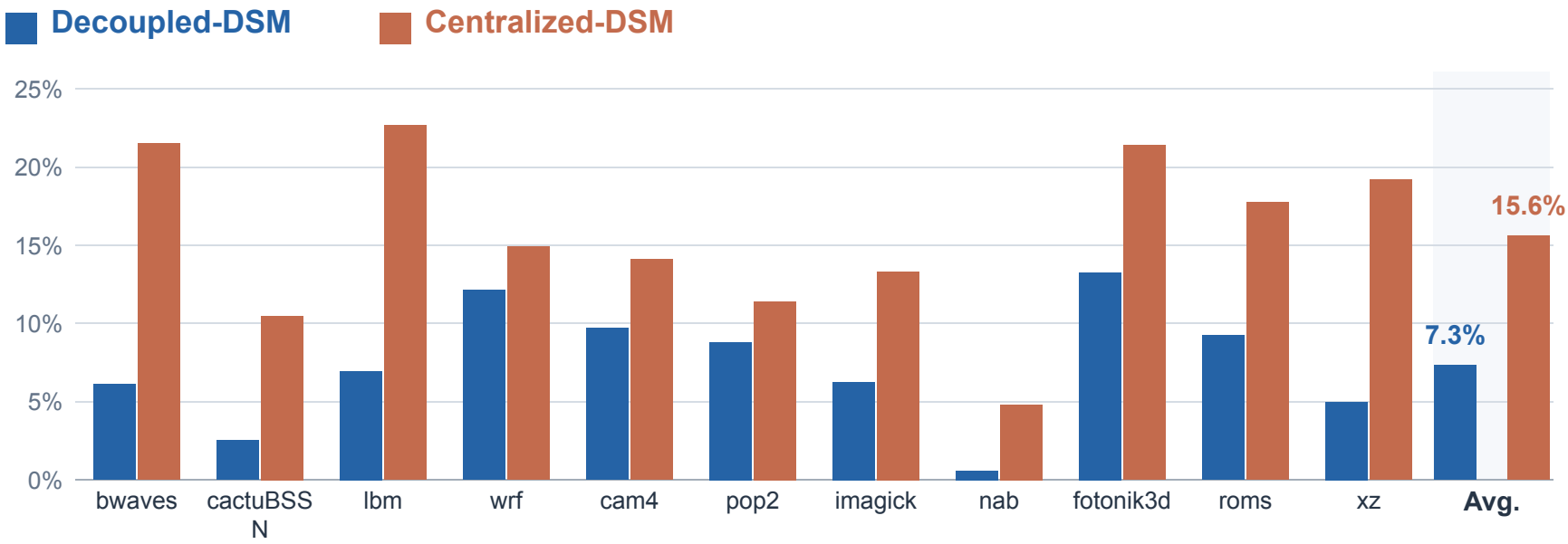
Evaluation setup

Dimension	Configuration
Simulation	ChampSim, 1–8 hosts, 2 cores per host
Memory	CXL-resident data, DDR5-6400, 70 ns CXL round trip
SPEC CPU2017	11 workloads, 1B instructions/core after 200M warm-up
Additional workloads	Redis, B-tree and hashmap sharing microbenchmarks
Security metadata	4 KB local cache/host, equal aggregate centralized cache

Performance normalized to insecure CXL-DSM

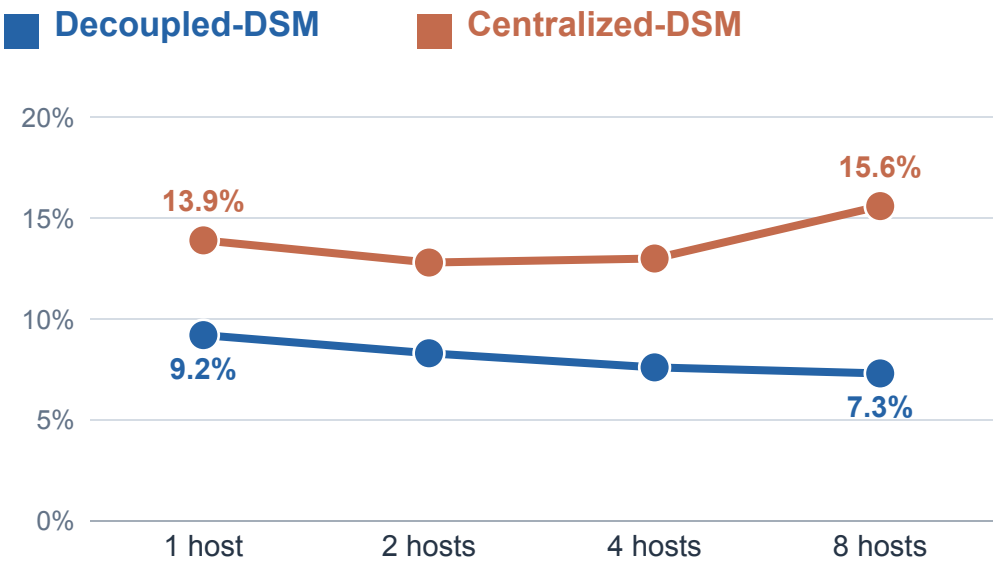
7.3% average overhead at 8 hosts

SPEC CPU2017, 8 hosts: execution-time overhead over insecure CXL-DSM



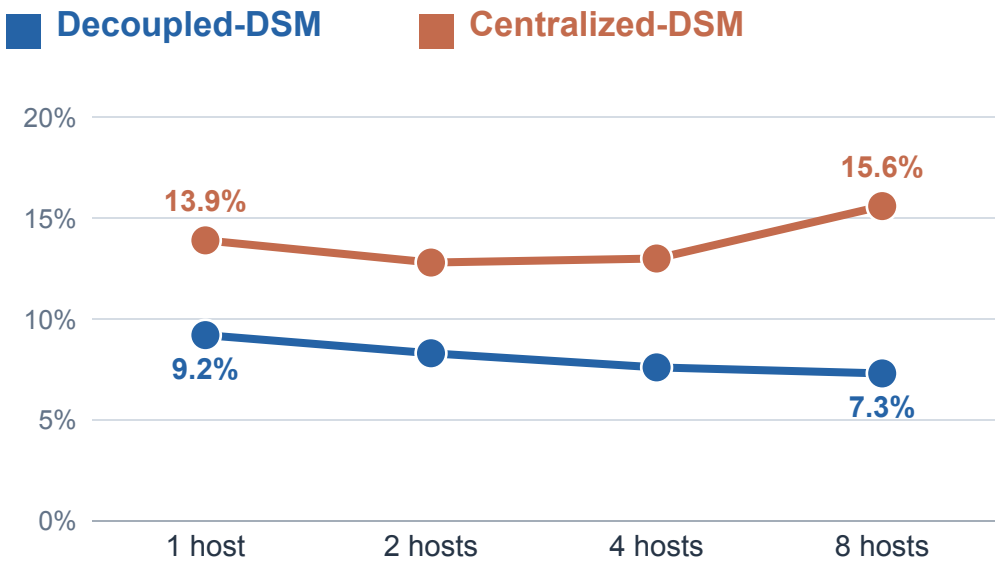
Scaling from 1 to 8 hosts

SPEC CPU2017: average execution-time overhead over insecure CXL-DSM



Scaling from 1 to 8 hosts

SPEC CPU2017: average execution-time overhead over insecure CXL-DSM



8.1%

average overhead across 1–8 hosts

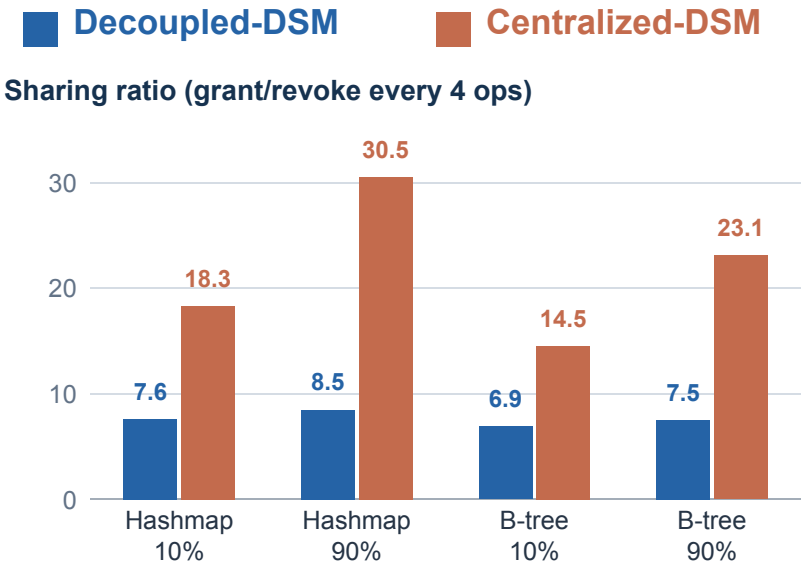
41.2%

lower security overhead than Centralized-DSM (average)

More hosts: our overhead stays low; centralized checks get worse

Robust to sharing and permission churn

8 hosts, hashmap and B-tree microbenchmarks: overhead (%) over insecure CXL-DSM

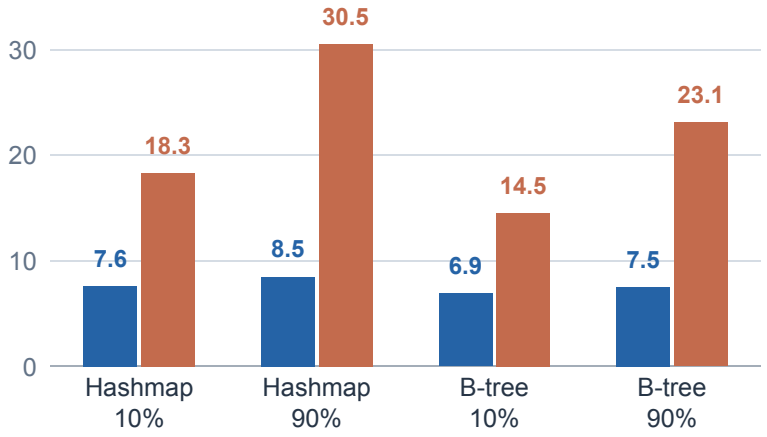


Robust to sharing and permission churn

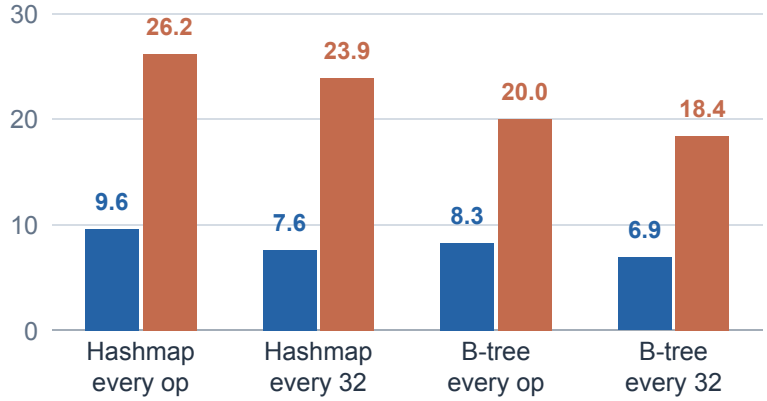
8 hosts, hashmap and B-tree microbenchmarks: overhead (%) over insecure CXL-DSM

Decoupled-DSM **Centralized-DSM**

Sharing ratio (grant/revoke every 4 ops)



Grant/revoke frequency (50% sharing)

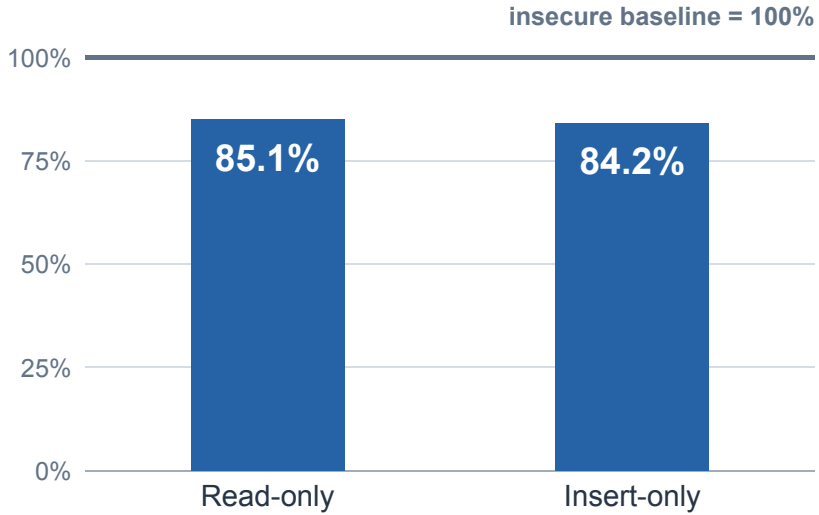


Under 10% even with grant/revoke on every operation

Redis: secure in-place communication

Client and server exchange requests through shared CXL memory (1M transactions)

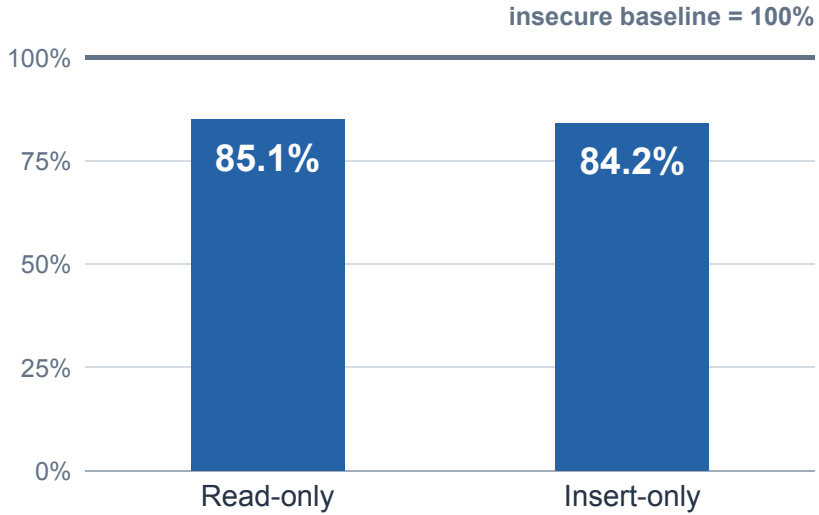
Decoupled-DSM throughput, normalized to insecure CXL-DSM



Redis: secure in-place communication

Client and server exchange requests through shared CXL memory (1M transactions)

Decoupled-DSM throughput, normalized to insecure CXL-DSM



+19.7%

throughput vs. Centralized-DSM

+281%

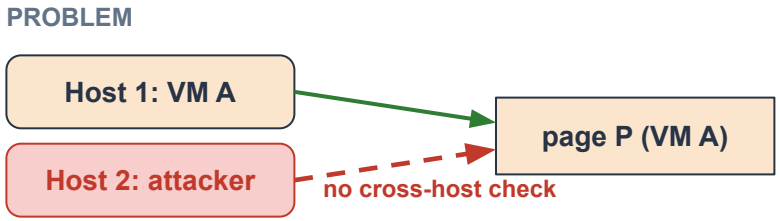
throughput vs. encrypt-and-copy sharing (existing TEEs)

A real application keeps ~85% of insecure throughput

Conclusion

Problem Per-host TEEs cannot protect CXL memory shared across hosts

Key idea Separate infrequent permission updates and mapping setup from frequent accesses

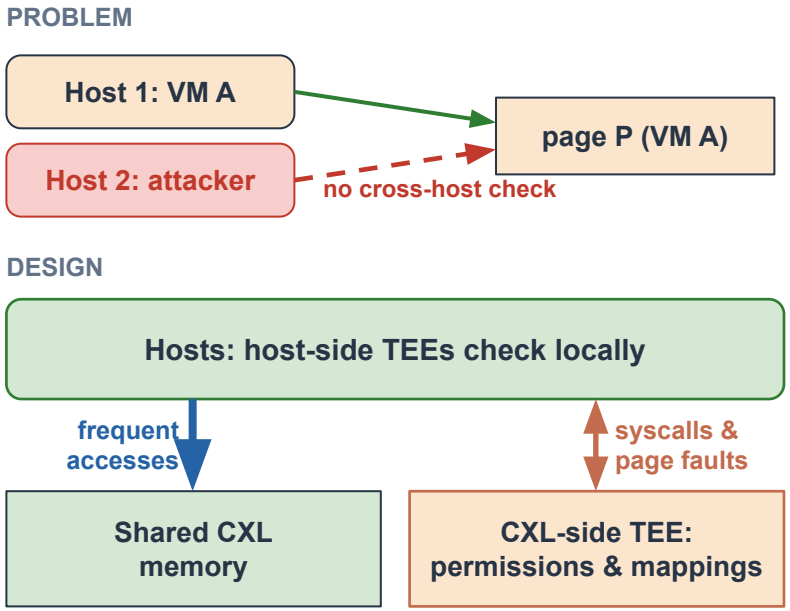


Conclusion

Problem Per-host TEEs cannot protect CXL memory shared across hosts

Key idea Separate infrequent permission updates and mapping setup from frequent accesses

Design CXL-side TEE: permissions & mappings; host-side TEEs: local access checks



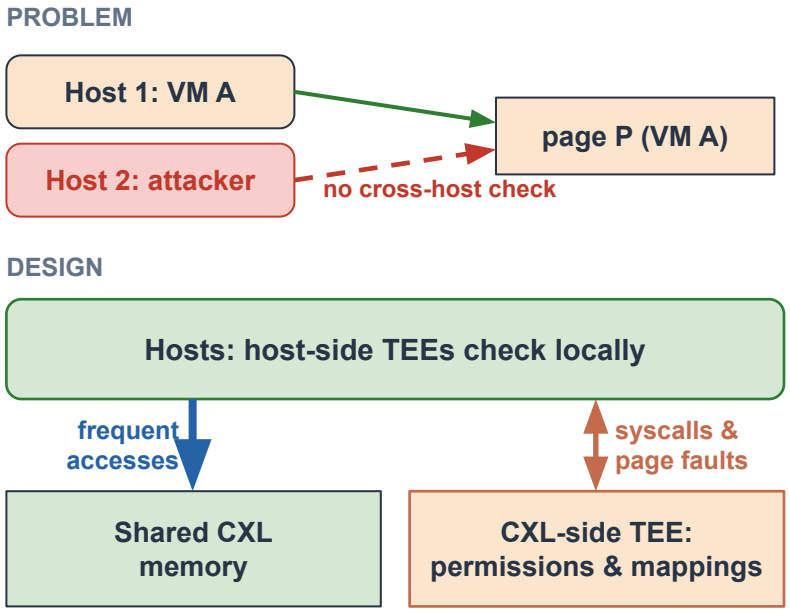
Conclusion

Problem Per-host TEEs cannot protect CXL memory shared across hosts

Key idea Separate infrequent permission updates and mapping setup from frequent accesses

Design CXL-side TEE: permissions & mappings; host-side TEEs: local access checks

Result 7.3%–9.2% overhead; 41.2% lower than centralized checking



Thank you! Questions?

yyang460@ucsc.edu