CHERITech'26
CONFERENCE

12-13 Nov 2026
Munich, DE

Towards Native Temporal Memory Safety in CHERI

Yuecheng Wang
University of Cambridge

Located at semi | SEMICON EUROPA

Spatial bounds do not encode object lifetime

MOTIVATION

CHERI capability: address · bounds · permissions · integrity tag

Spatial safety

Where may this pointer access?

Object bounds	Outside

Out-of-bounds access is rejected.

Temporal safety

When may this pointer access?

Allocate ➡ **Free** ➡ **Use?**

Bounds can remain valid after free.

An in-bounds pointer can still be a dangling pointer.

Local checks make global revocation hard

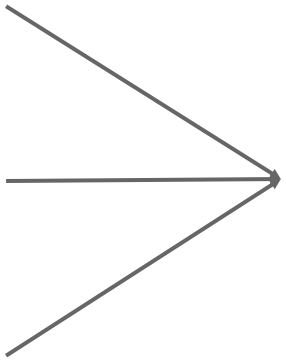
THE CAPABILITY-SYSTEM TRADE-OFF

Authority travels with pointers, so access checks stay local.

Thread A: capability copy

Thread B: capability copy

Heap: capability copy



Freed allocation
Copies still exist

> **Freeing one allocation cannot update every distributed capability.**

Make temporal validity part of CHERI

DESIGN GOAL

01 Capability enforcement

Reuse bounds, authority and hardware-enforced access checks.

02 Allocation-local state

Keep temporal metadata with the memory it protects.

03 Simple, local access

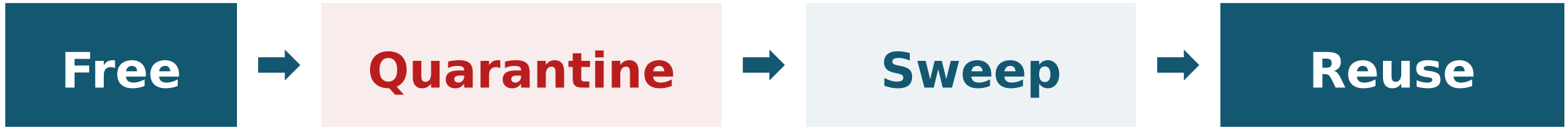
Preserve locality without dedicated per-line lifetime tags.

Evolve CHERI itself toward native spatial and temporal safety.

Quarantine makes reuse wait for revocation

EXISTING APPROACH

Freed memory waits until dangling capabilities have been revoked.



Shadow bitmap

Delayed reuse **Memory overhead** **Scanning work**

Can CHERI reject stale access while releasing memory sooner?

Lifetimes go beyond libc allocation

TWO FURTHER CHALLENGES

malloc: one large region

SQLite: smaller object lifetimes

Live	Freed	Live

Allocated ≠ initialized

Zeroing does not prove the program wrote the object.

Protection must understand hierarchy and initialization.

Capabilities can represent poisoned memory

On free, the allocator writes poison capabilities into the object.



Stale capability → Hardware rejects access

> **In-place poison replaces the freed-state shadow bitmap.**

PoisonCap · arXiv:2605.13210

One freed object, two different capabilities

The child allocator frees B. Both capabilities try to access B.

Application capability

Bounds: only object B



Stale access rejected

Parent allocator capability

Bounds: entire parent region



Management allowed

Poison bounds + capability bounds identify the accessing authority.

No separate freed-state bitmap for each allocation layer.

PoisonCap · arXiv:2605.13210

Poisoning also detects read-before-write

1 / POISONCAP

New allocation	→	Initializing write	→	Subsequent read
Poison remains		Clears poison		Allowed

Read while poison remains → detected by hardware

Prototype initialization tracking uses capability-word granularity.

> **Safe memory reuse still waits for revocation.**

PoisonCap · arXiv:2605.13210

The capability tells hardware where to find the ID

The allocator sets protected ID, IDMODE and IDLOC capability fields.

Capability		Locate the ID byte		Read and compare
Address · bounds **ID = 7** **IDMODE · IDLOC**	➡	Aligned region base + decoded IDLOC IDMODE: line or page	➡	Memory ID = 7 **7 = 7: allow** **After free: 7 ≠ 8**

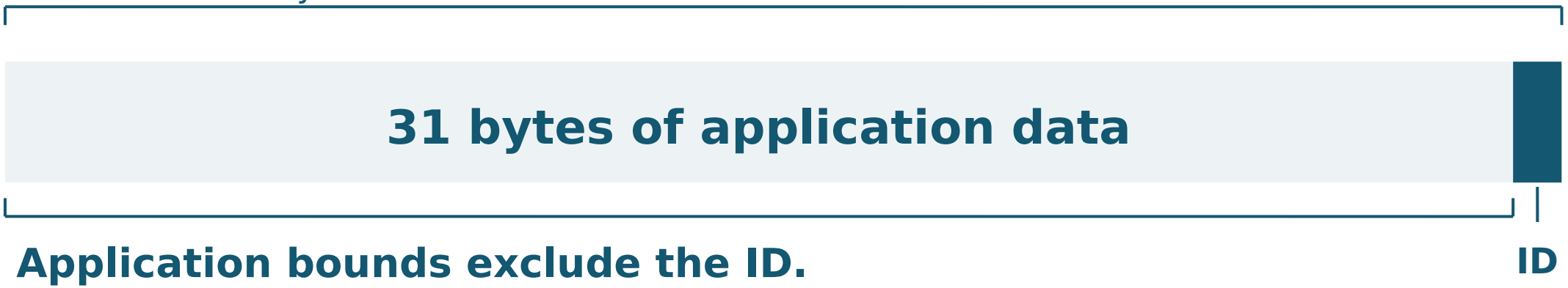
Change the stored ID on free; old capabilities stop matching.

CHERI-D · arXiv:2606.19055v2

Spatial bounds protect temporal metadata

An 8-bit lifetime ID needs only one spare byte.

Illustration: 31-byte object in a 32-byte allocator slot



Application bounds exclude the ID.

A single byte of padding or internal fragmentation is enough.

CHERI’s spatial guarantees secure allocation-local lifetime state.

CHERI-D · arXiv:2606.19055v2

One lifetime ID protects the whole allocation

Ordinary memory

No dedicated per-line lifetime-tag storage.

Metadata locality

Small objects share a cache line; larger ones use page-local IDs.

One ID per allocation

ID storage and updates do not grow with allocation size.

Access checks, ID buffering and coherence still need hardware support.

Spatial safety makes protected temporal metadata local.

CHERI-D · arXiv:2606.19055v2

Immediate reuse reduces revocation pressure

Revocation sweeps: one benchmark example

omnetpp · SPEC CPU2006 train · FPGA

2,792 → **10** → **2**

Cornucopia Reloaded

CHERI-D

Reincarnate

CHERI-D v2, §VI-F and Table IV; sweep counts.

Must an exhausted ID stop the underlying memory being reused?

CHERI-D · arXiv:2606.19055v2

Reincarnate · arXiv:2609.11590 · draft

Retire the ID, keep reusing the memory

The allocation slot and its ID location have separate lifetimes.

Same memory allocation slot	ID A retired	ID B active

Old capability still names A

Rejected; A waits for revocation

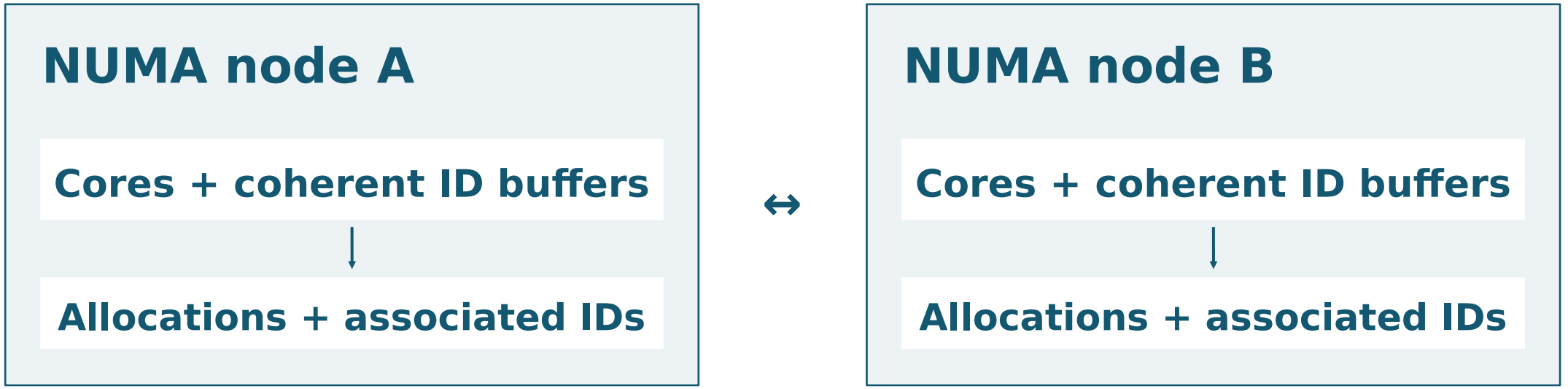
New capability names B

Allowed; memory stays reusable

> **Quarantine temporal authority while keeping memory reusable.**

CHERI-D Reincarnate · arXiv:2609.11590 · draft

Keep temporal state distributed and coherent



Metadata placement follows the allocation.

Coherence events invalidate stale cached IDs across cores.

Distributed authority, allocation-local state, coherent checks.

CHERI-D Reincarnate · arXiv:2609.11590 · draft

Towards native spatial and temporal safety

THREE WORKS, ONE PROTECTION PHILOSOPHY

PoisonCap	Capabilities express freed and uninitialized state.
CHERI-D	Spatial bounds secure allocation-local lifetime IDs.
Reincarnate	Quarantine exhausted IDs. Reuse the same memory with a fresh ID location.

One capability protection model for where and when.

Thank you · Questions