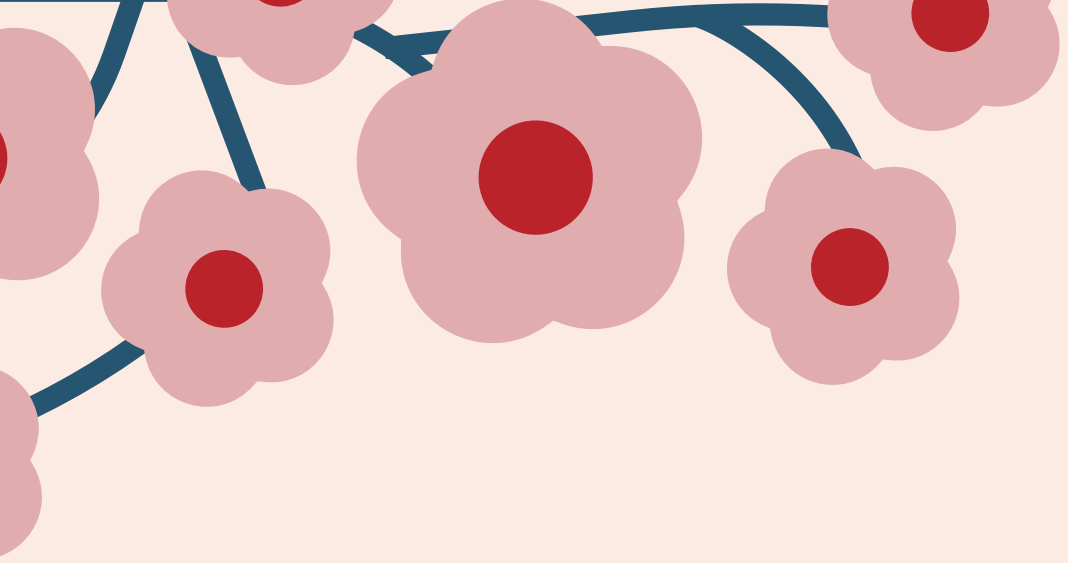
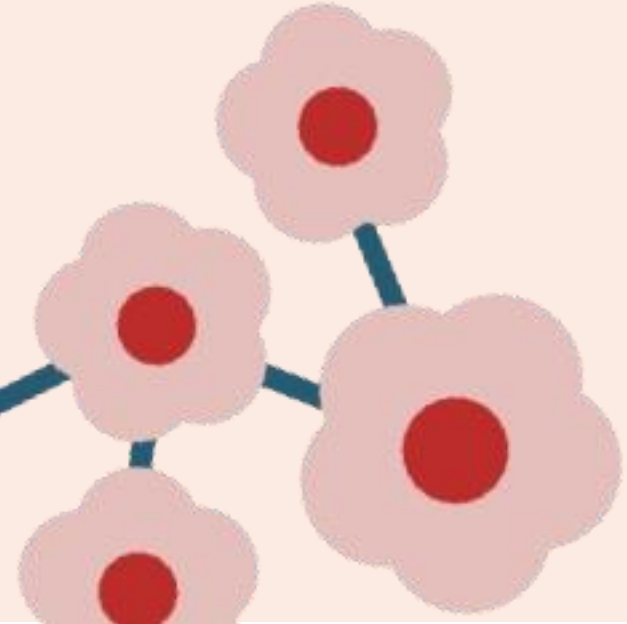
06 April 2026





MMU-based Revocation

Use-after-free/reallocate prevention for Virtual Memory based systems

Author

Tariq Kurd – Codasip

Alfredo Mazzinghi – University of Cambridge

Agenda

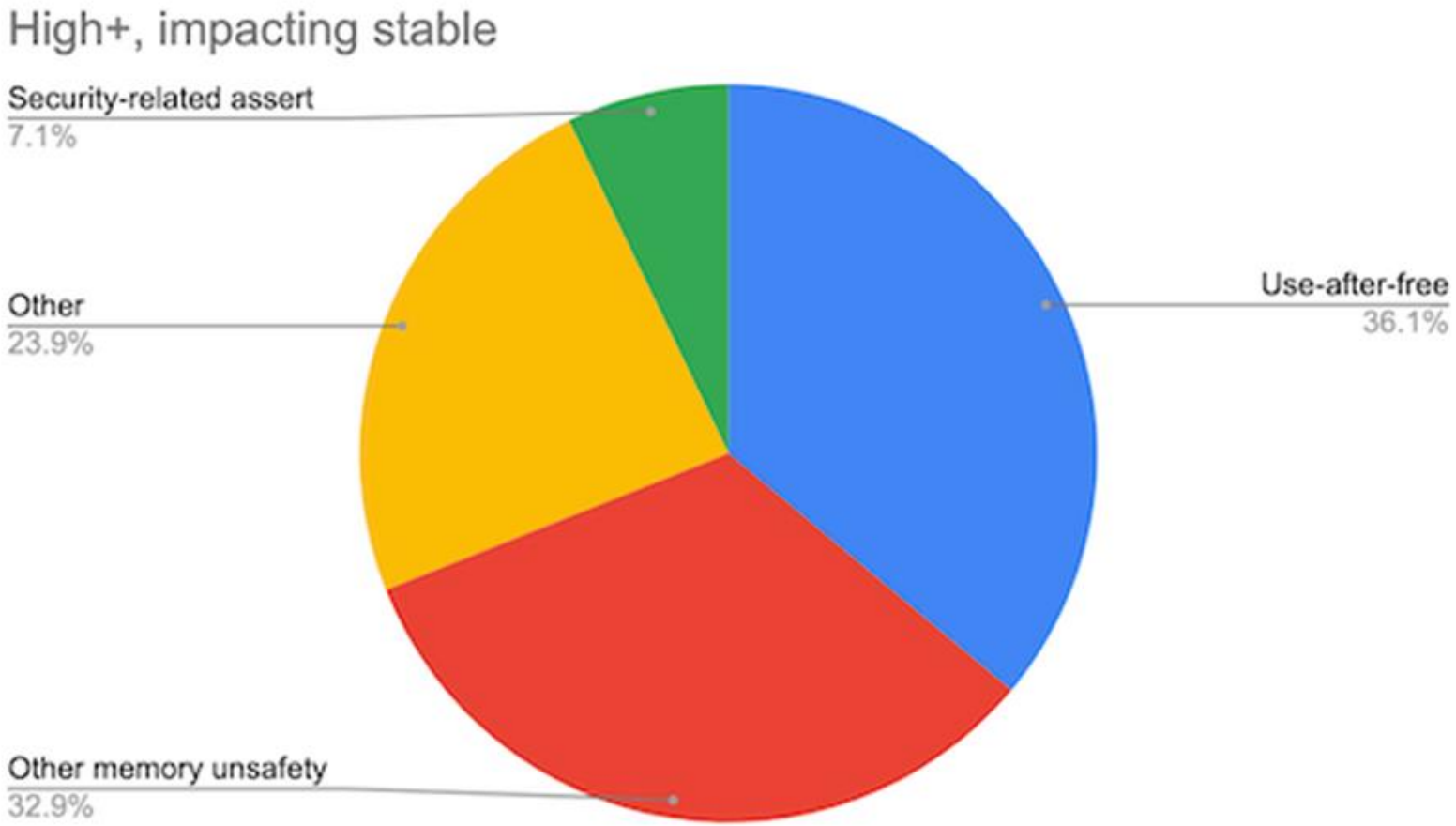
- What is revocation?
- Brief details of revocation algorithms
- Implementing the algorithms on an MMU-based CHERI system

What is revocation?

What is revocation (in the context of CHERI)?

- Revocation is the act of locating and invalidating **stale** or **dangling** pointers
  - Memory is dynamically allocated using *malloc()*
  - Some time later, the memory is freed via a call to *free()*
  - Pointers to the freed region often remain and are known as **stale** or **dangling**
    - Use of such a pointer at this point is use-after-free
  - Some time later, the memory is reallocated
    - Use of such a pointer at this point is use-after-reallocate
- CVEs list both types as use-after free.
- Here's an example of a recent one in Chromium:
  - https://www.cve.org/CVERecord?id=CVE-2026-2441

Chromium memory safety bugs
Use-after-free is very prevalent…..



https://www.chromium.org/Home/chromium-security/memory-safety/

What can we do about it?

And why does CHERI help?

- CHERI means that dangling pointers can be accurately identified and invalidated
  - It makes a huge difference being able to accurately identify valid capabilities
- This gives a big advantage over non-CHERI architectures where:
  - Heuristics are used to identify pointers
  - They can't be invalidated in such a simple way
  - Pointer values can be simply made-up to probe memory

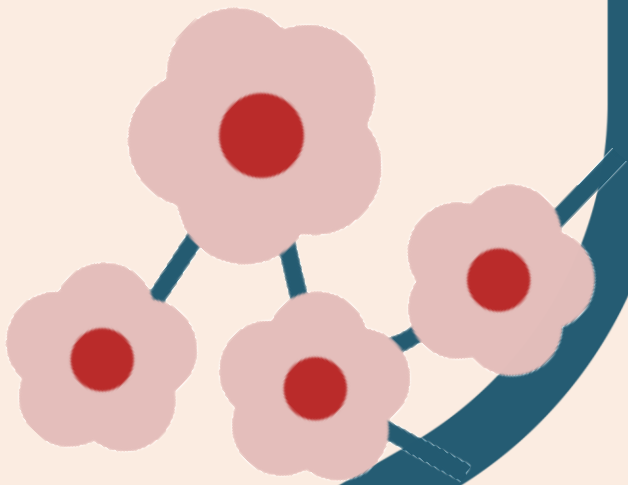
How do we locate and identify the pointers?

- **CHERIvoke** introduced the stop-and-sweep algorithm
  - After calling free() *stop-the-world* and sweep all virtual memory for dangling pointers. Effective but low performance.
- **Cornucopia** introduced the concept of background sweeping and sweeping epochs
- **Cornucopia Reloaded** further reduced the stop-the-world phase for improved parallel operation
  - The OS launches a background sweeper thread which is always running and searching for dangling pointers.
  - Freed memory is held in quarantine until all dangling pointers have been located.
- The Cornucopia Reloaded scheme is implemented on Morello and runs on CHERIBSD
  - Currently being ported to Linux on CHERI-RISC-V by Codasip

How do we achieve this using the MMU?

- We allocate four PTE bits to implement the states available on Morello to give us two mechanisms:
  - Load-barrier – the ability to trap on loading a capability from a VM page so the page can be scanned for dangling capabilities **before** the load completes
  - Capability dirty tracking – Track which VM pages have had capabilities stored to them. The mechanism is identical to normal A/D tracking, setting an additional YD bit on the store of a valid capability.

Revocation Overview



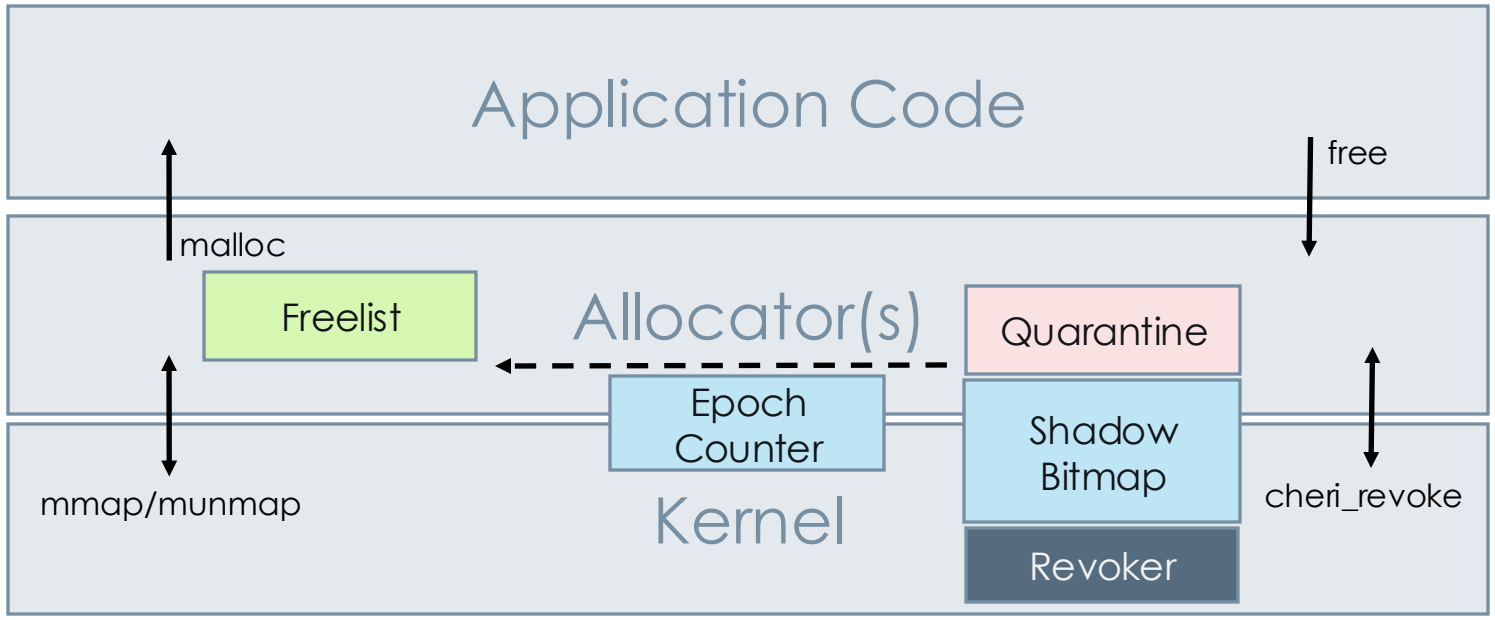
06 April 2026

CHERI

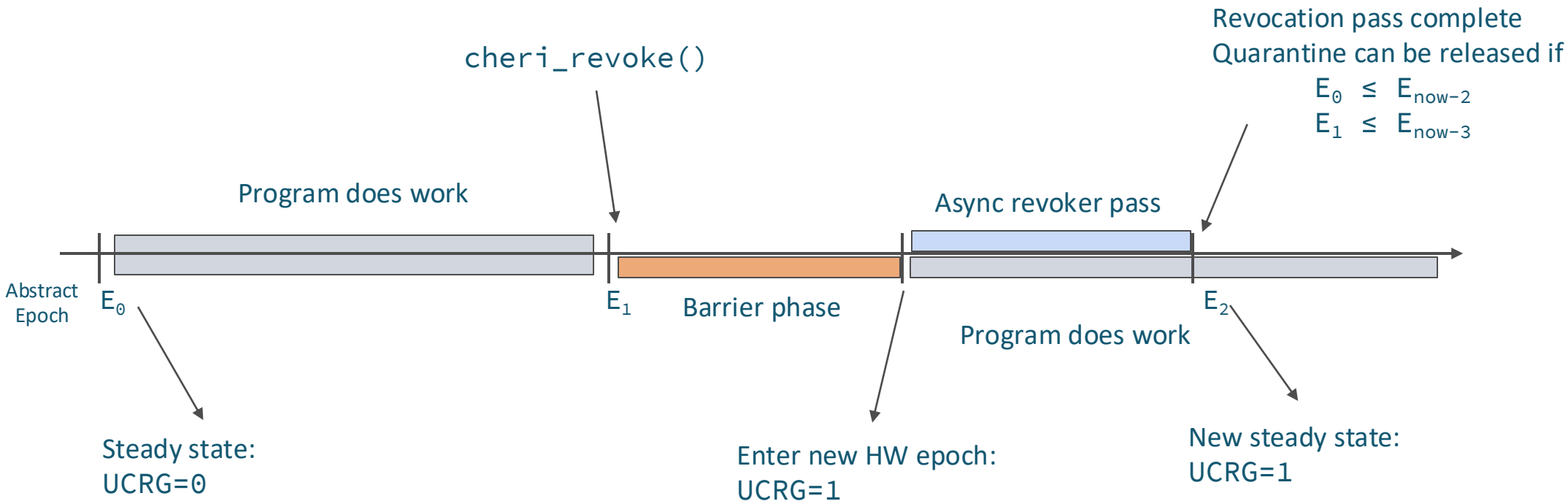
Revoker Operation Overview (1/4)

- Revocation is a shared responsibility
  - Revocation state is per-address space (per process).
  - A memory allocator must:
    - Mark quarantined memory via a shadow bitmap.
    - Trigger a revocation sweep.
    - Reclaim free memory from the quarantine after a sweep.
  - The kernel exposes a revocation API:
    - Provides a shadow bitmap for each mapped memory region.
    - Provides a mechanism to trigger a revocation sweep.
    - Maintains the revocation epoch counter.
    - Runs the asynchronous memory sweep and revoke capabilities.

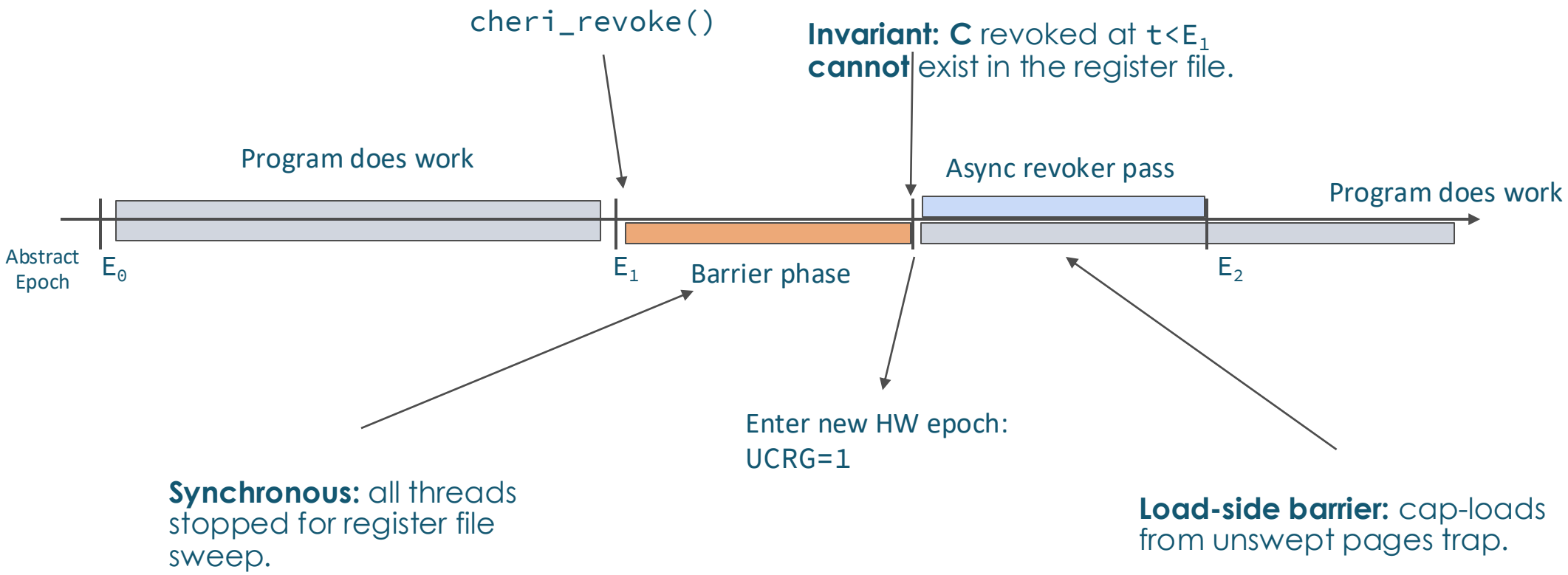
Revoker Operation Overview (2/4)



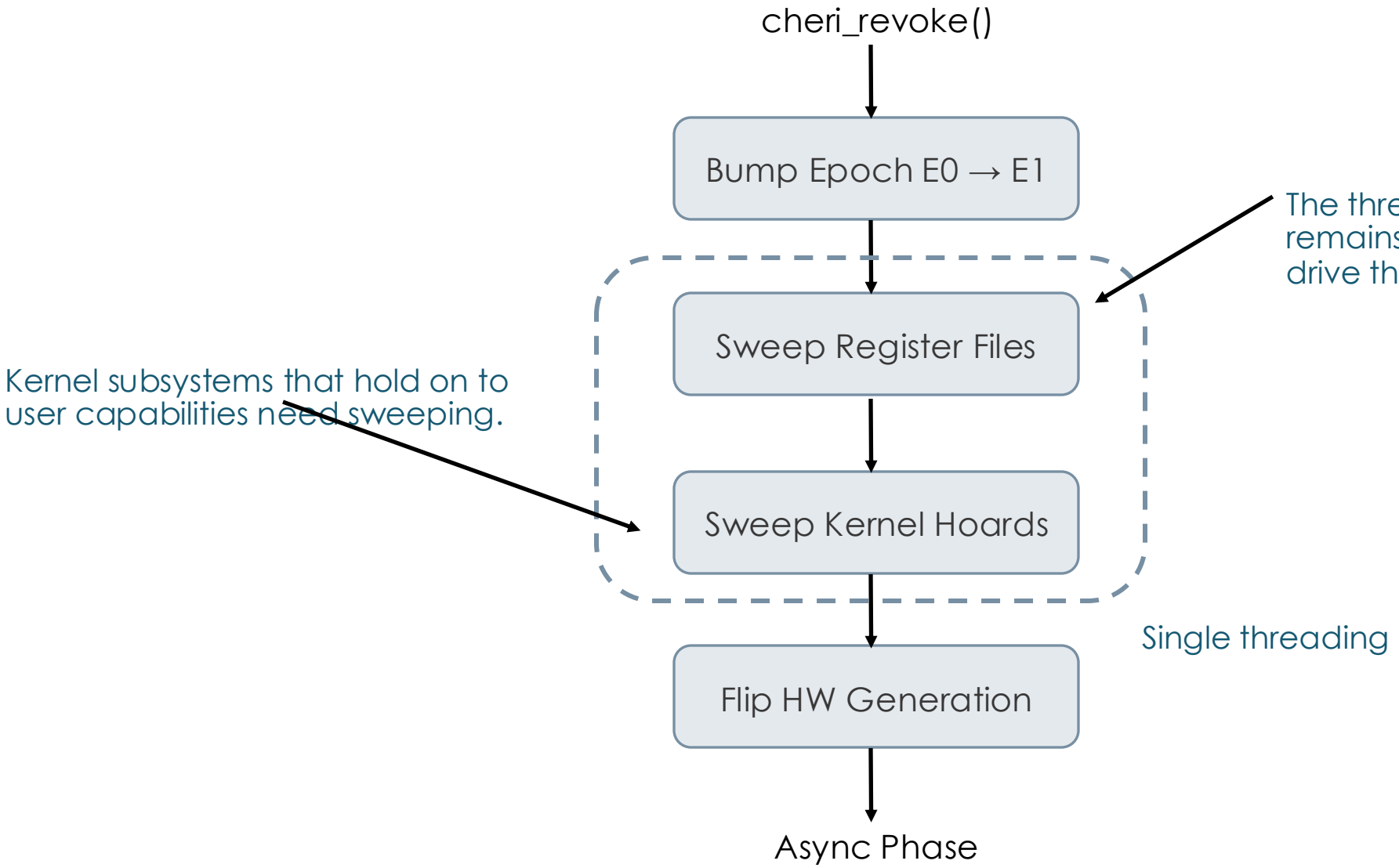
Revoker Operation Overview (3/4)



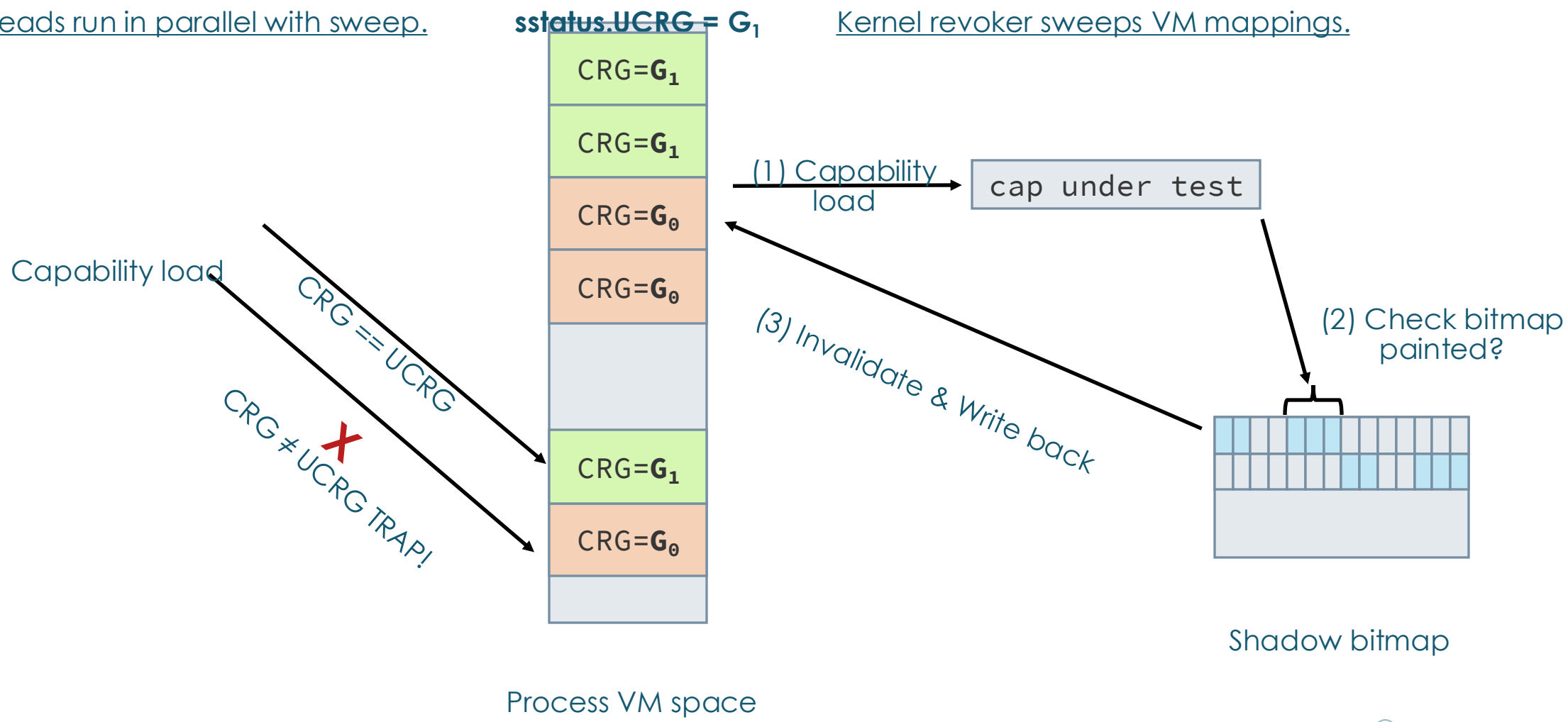
Revoker Operation Overview (4/4)



Barrier Phase



Asynchronous Sweep



Revoker Optimisations

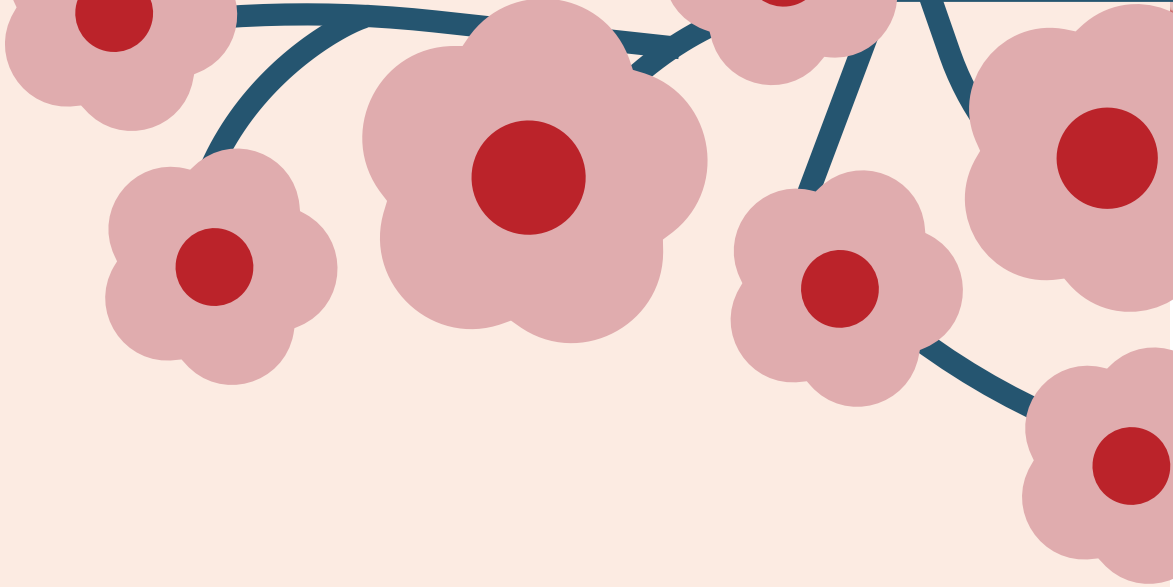
- Sweeping memory is costly!
  - DRAM traffic & cache effects.
- What if we could skip pages that we know do not have any capabilities?
  - Detect *cap-clean* (physical) pages
    - Simple for pages that **never** hold capabilities (e.g. mmap'ed files).
    - Need to track **capability dirtiness** for anonymous pages.
    - Additional PTE bits used to detect **ephemerally cap-clean** pages.

Sources of Complexity

- Aliased Mappings
  - Multiple virtual mappings for the same physical page
  - Don't scan twice
  - Need to arbitrate *cap-cleanliness* detection carefully
- Superpages and large pages (e.g. 64KiB granule)
  - Fault handler needs to balance latency, but splitting superpages is also costly.
- TLB management
  - Minimize TLB invalidation
  - Complex TLB management strategy to handle *cap-dirty* tracking

CHERI RISC-V Specification

- State of support:
  - Current QEMU and CheriBSD RV64Y work on older 0.9.3 spec.
  - Patches available to demonstrate RV64Y Svyrg latest extension.
  - Linux support in the works (Codasip).
- MMU-based revocation specified for RV64Y, not RV32Y.
  - Different approaches are possible (CherIoT model).



CHERI

THANK YOU

Contact tariq.kurd@codasip.com, am2419@cl.cam.ac.uk

Web www.cheri-alliance.org