Garbage Collection Without Paging

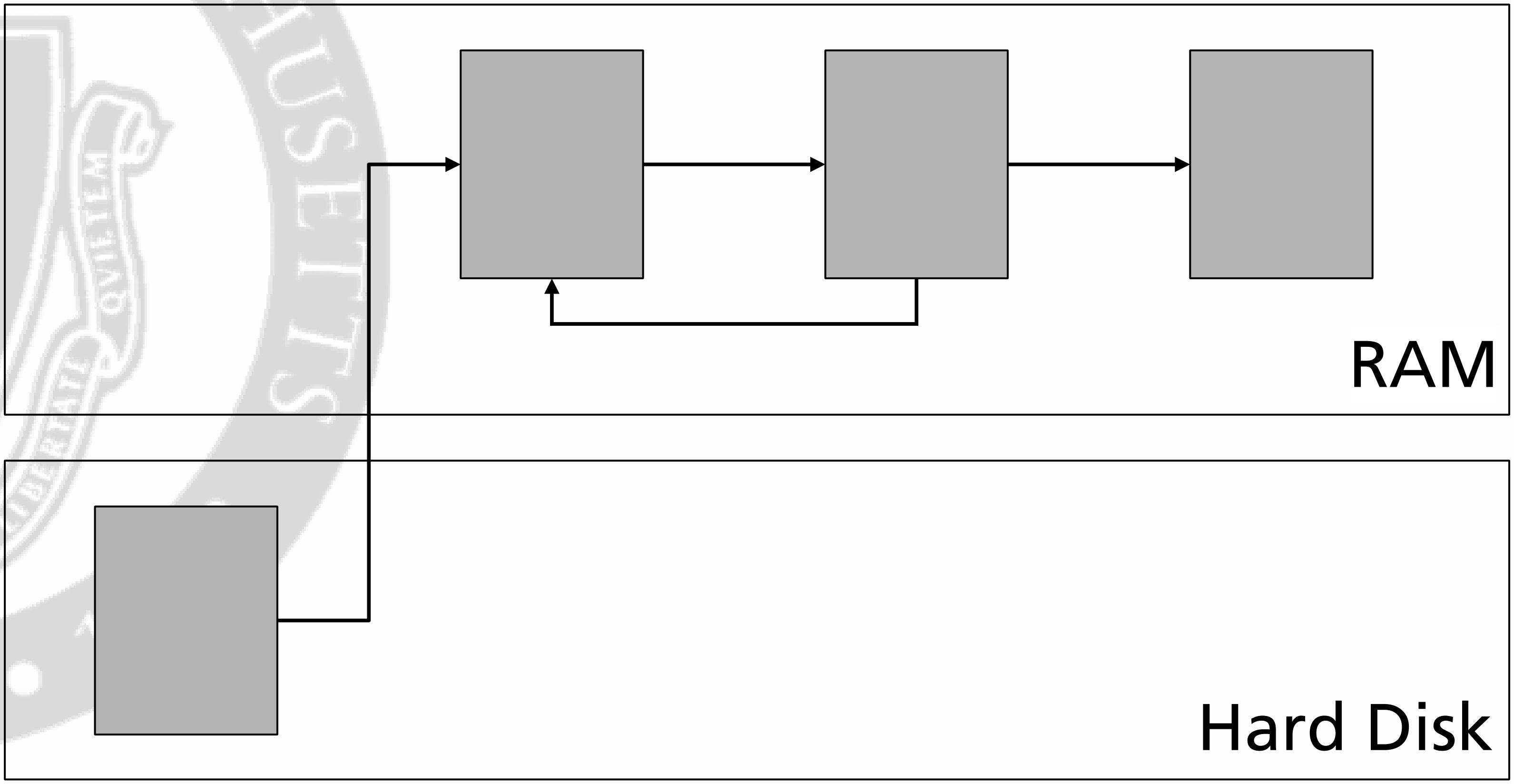
Matthew Hertz, Yi Feng,
Emery Berger

University of Massachusetts Amherst

Garbage Collection Performance

- Garbage collection now performs reasonably well
  - High throughput
  - Low pause times
  - *Given large heap and sufficient memory*
- But: what happens when there's not enough RAM?

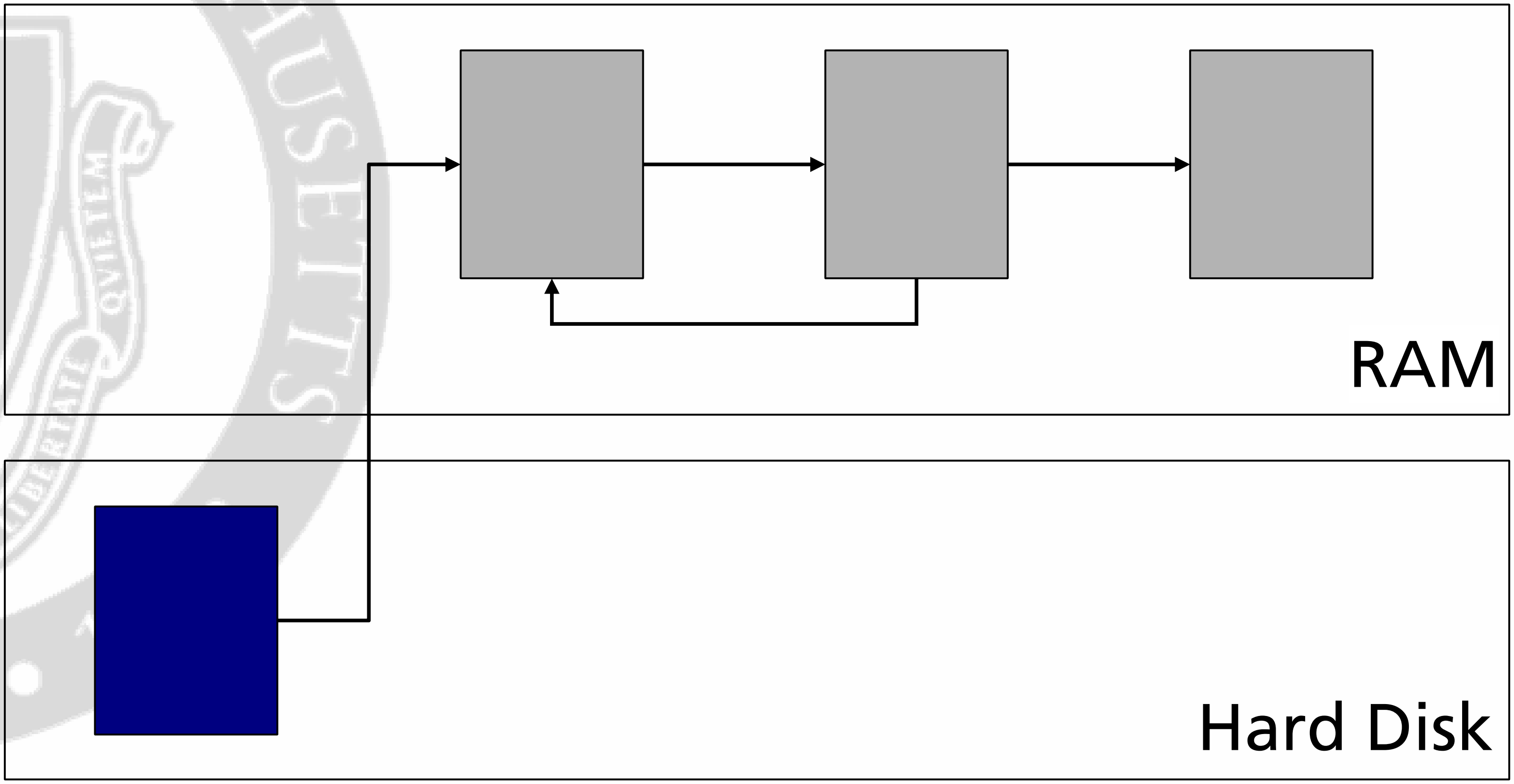
GC Performance While Paging



Heap: most pages in RAM, one on disk

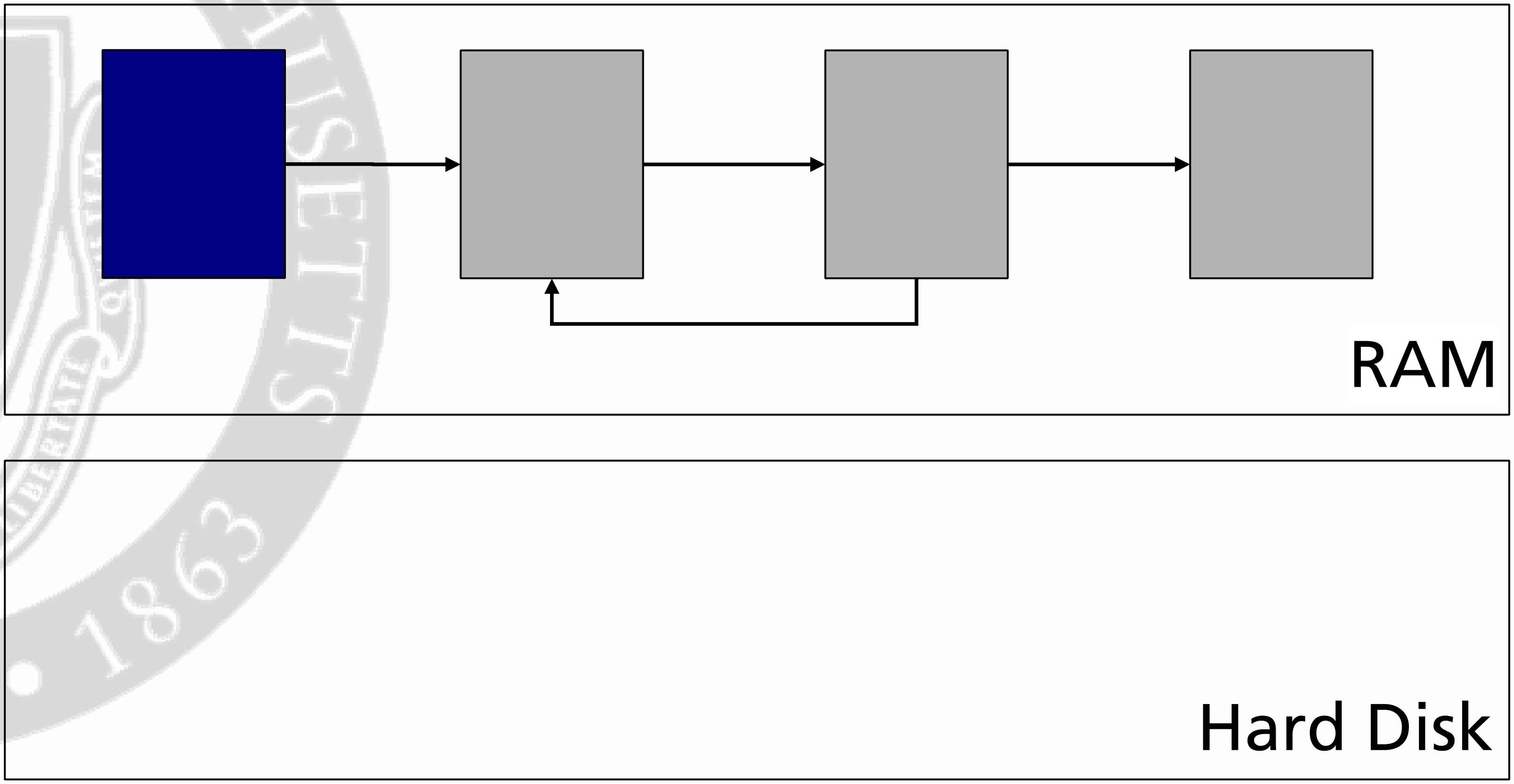
GC begins

GC Performance While Paging



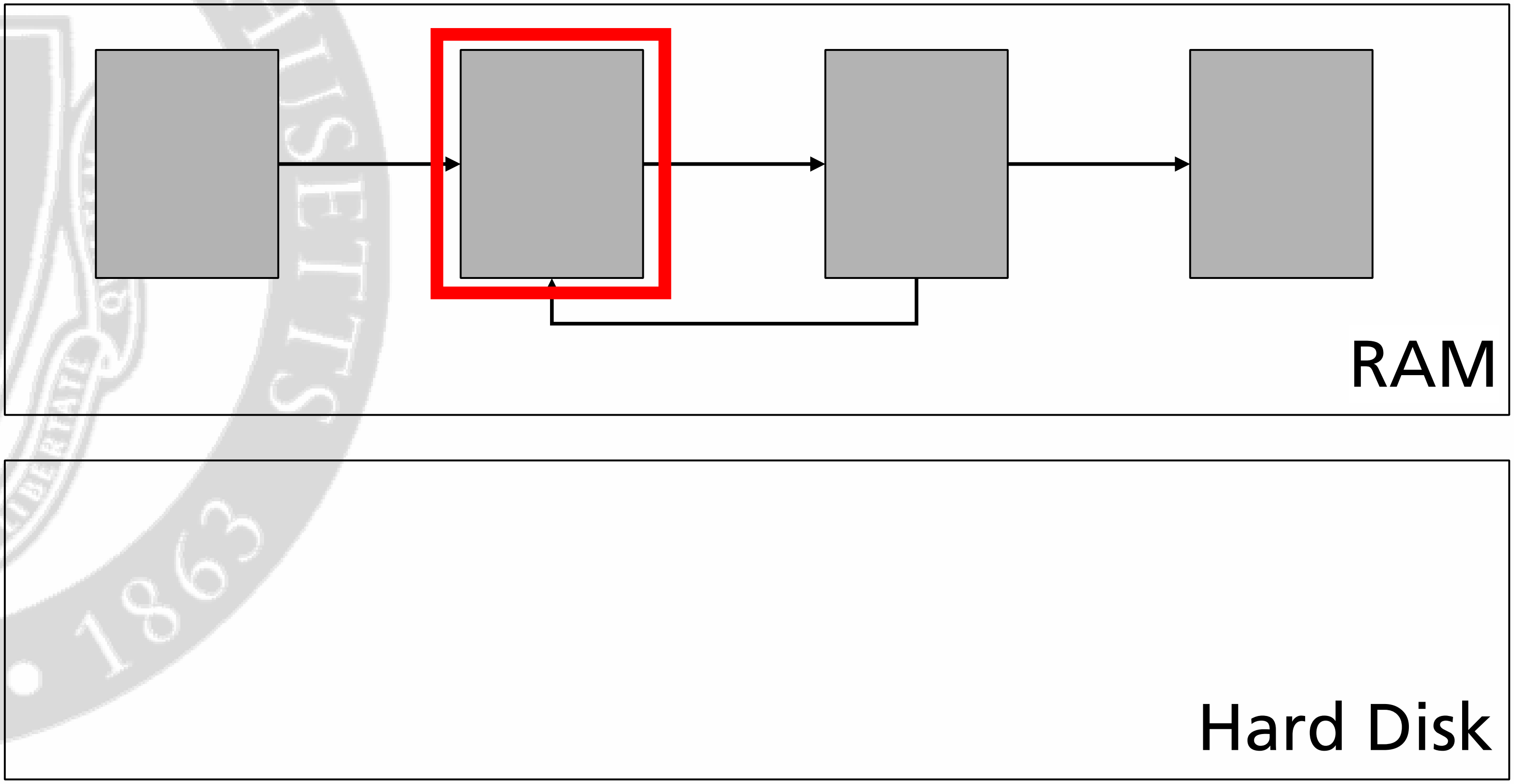
Collector touches an evicted page...

GC Performance While Paging



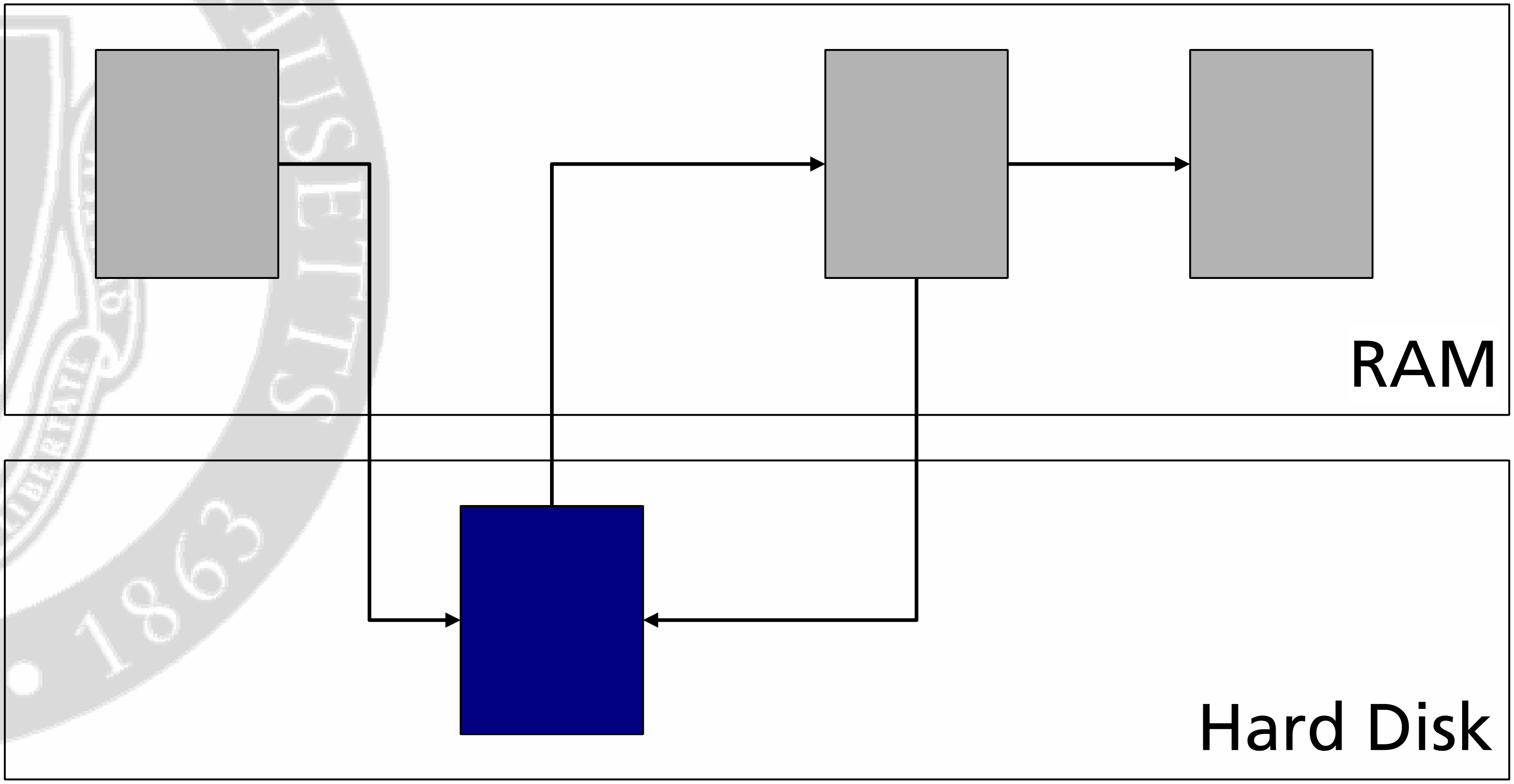
… bringing the page into memory …

GC Performance While Paging



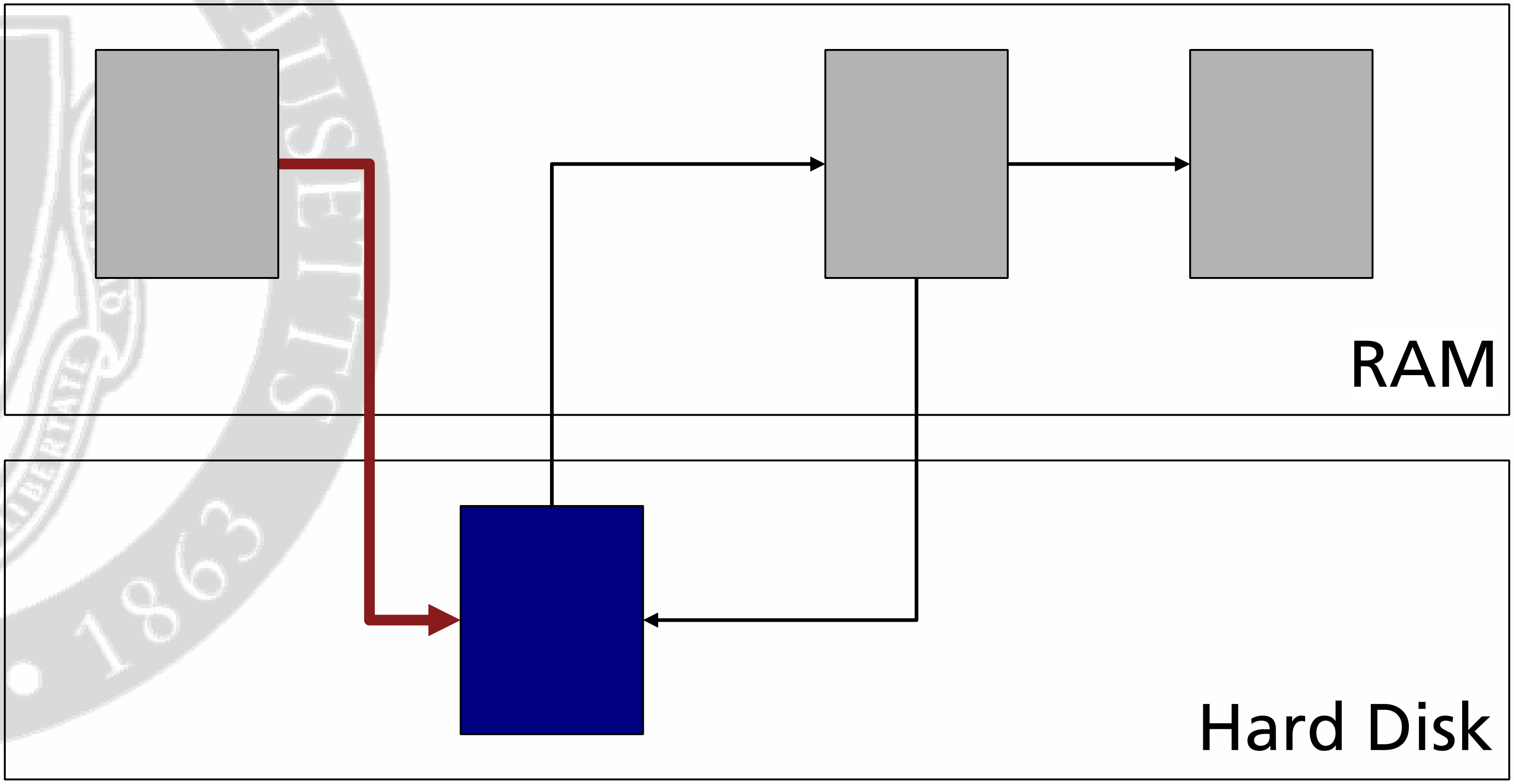
... but triggers another page eviction.

GC Performance While Paging



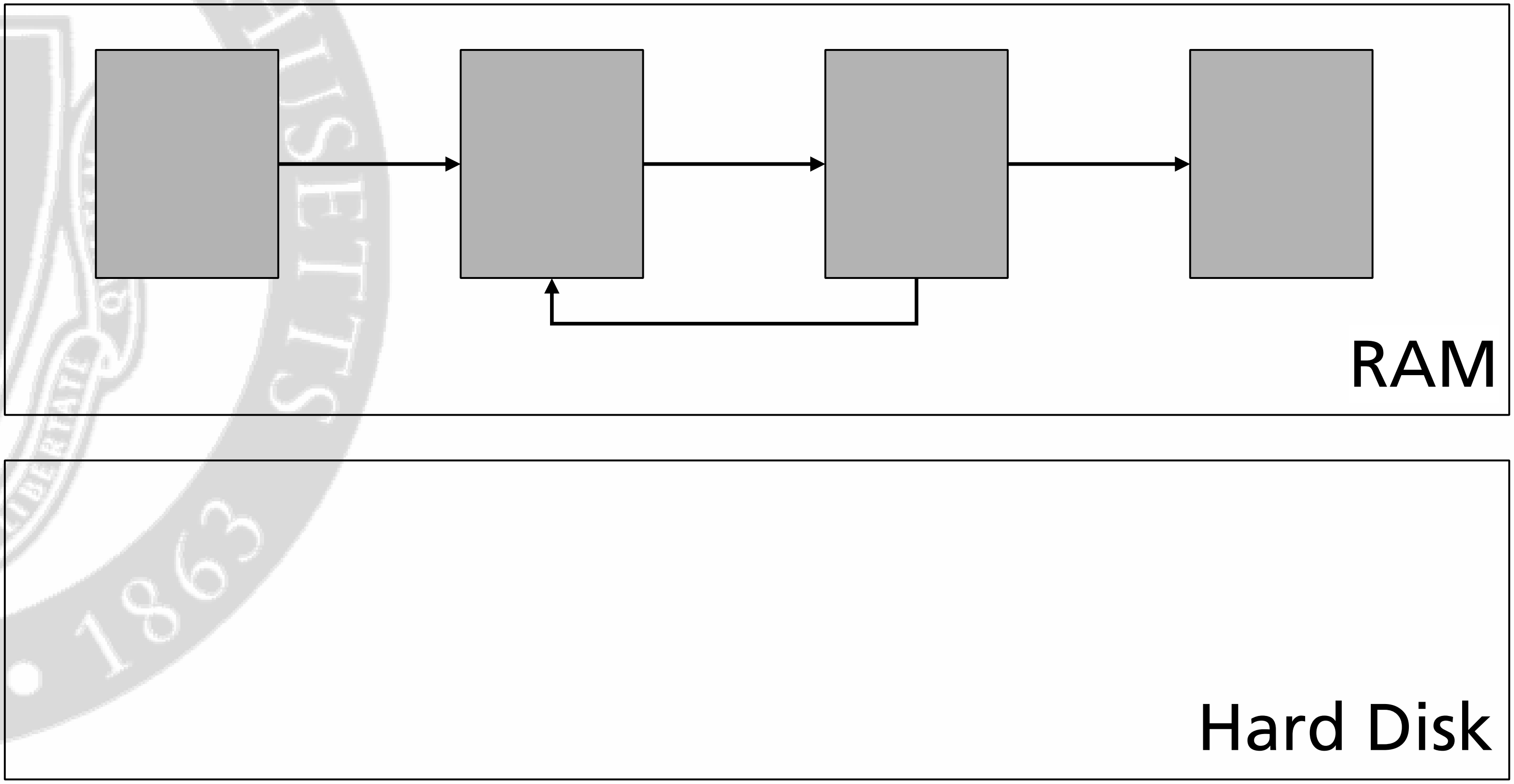
... but triggers another page eviction.

GC Performance While Paging



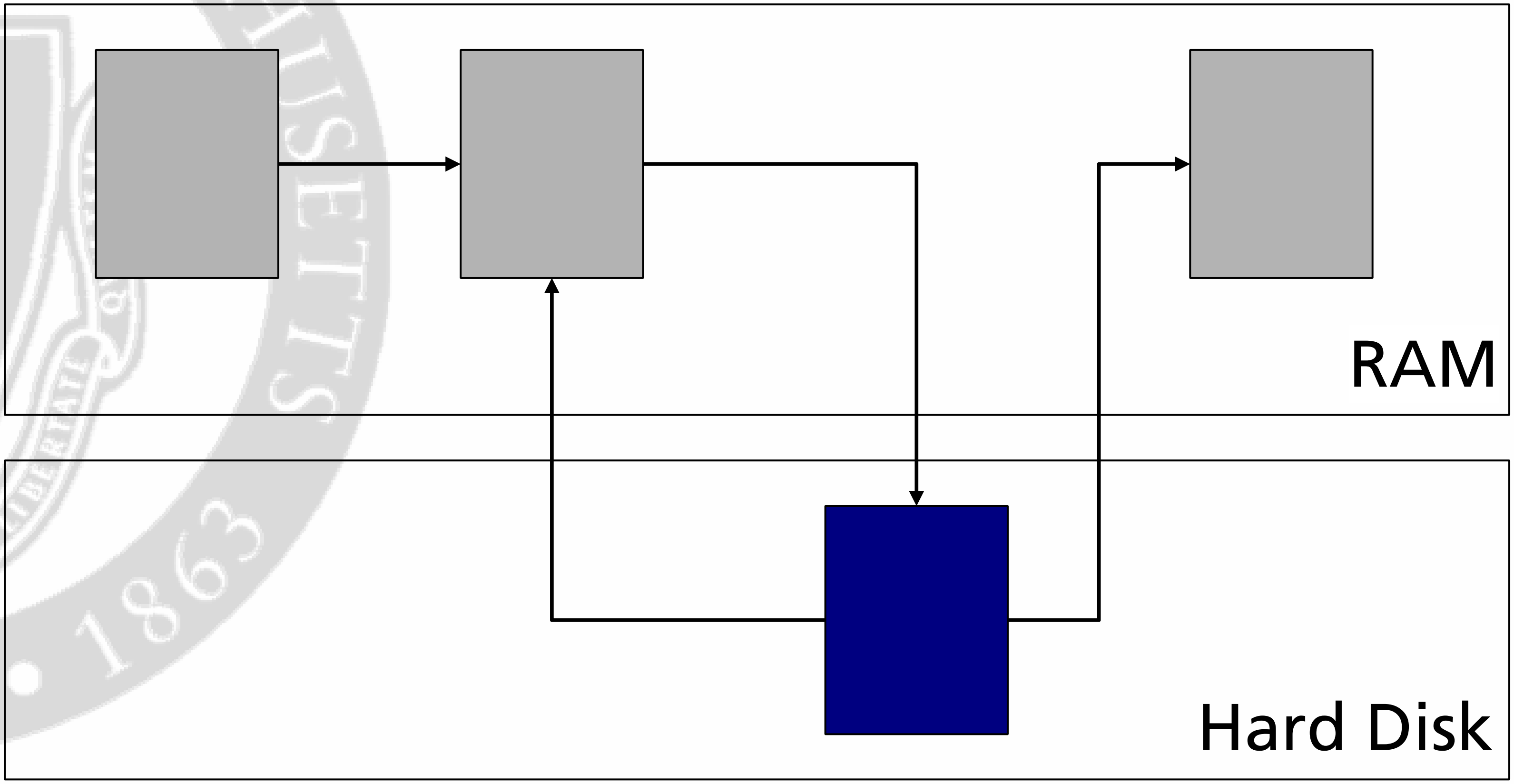
Collector touches newly-evicted page…

GC Performance While Paging



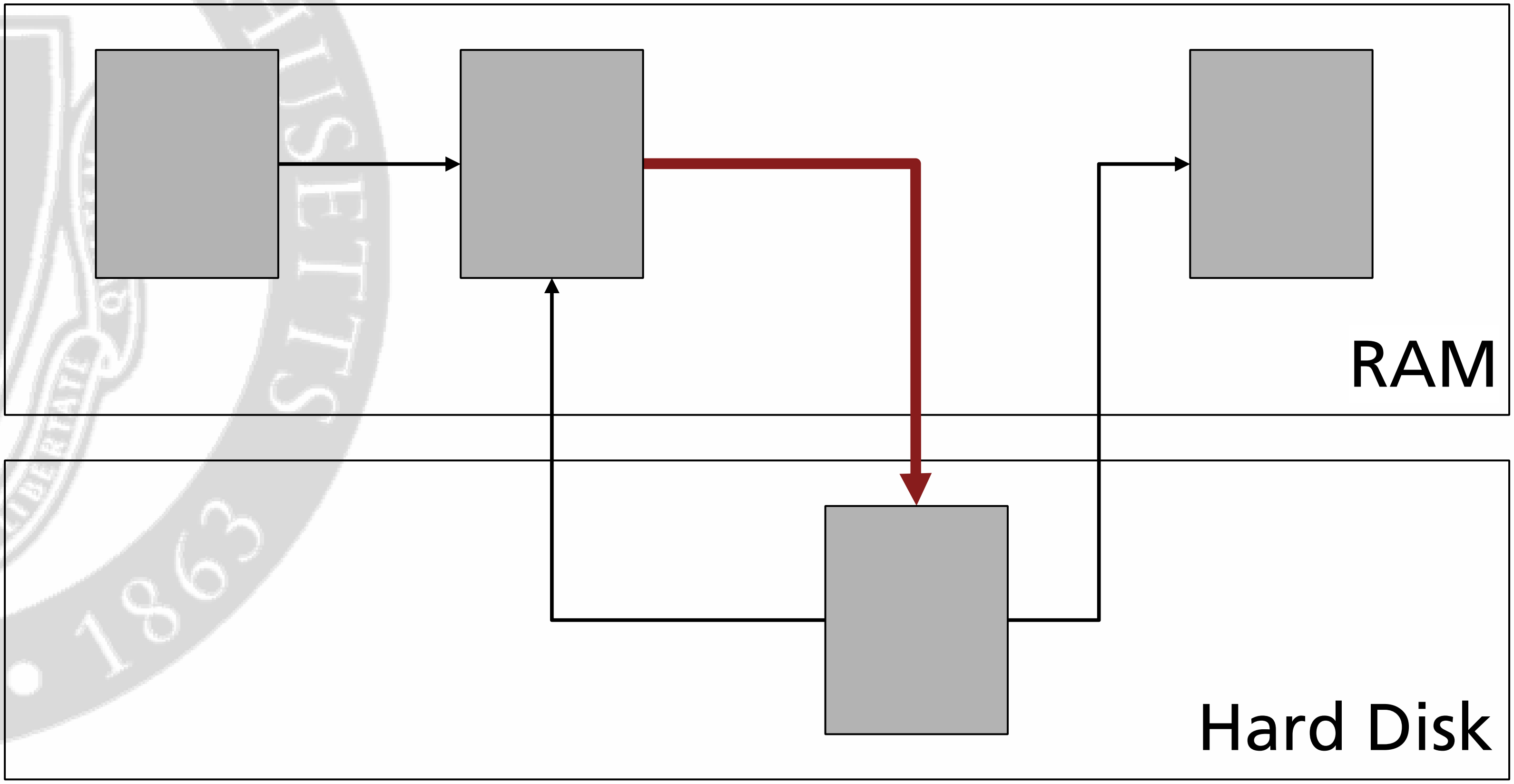
Collector touches newly-evicted page...

GC Performance While Paging



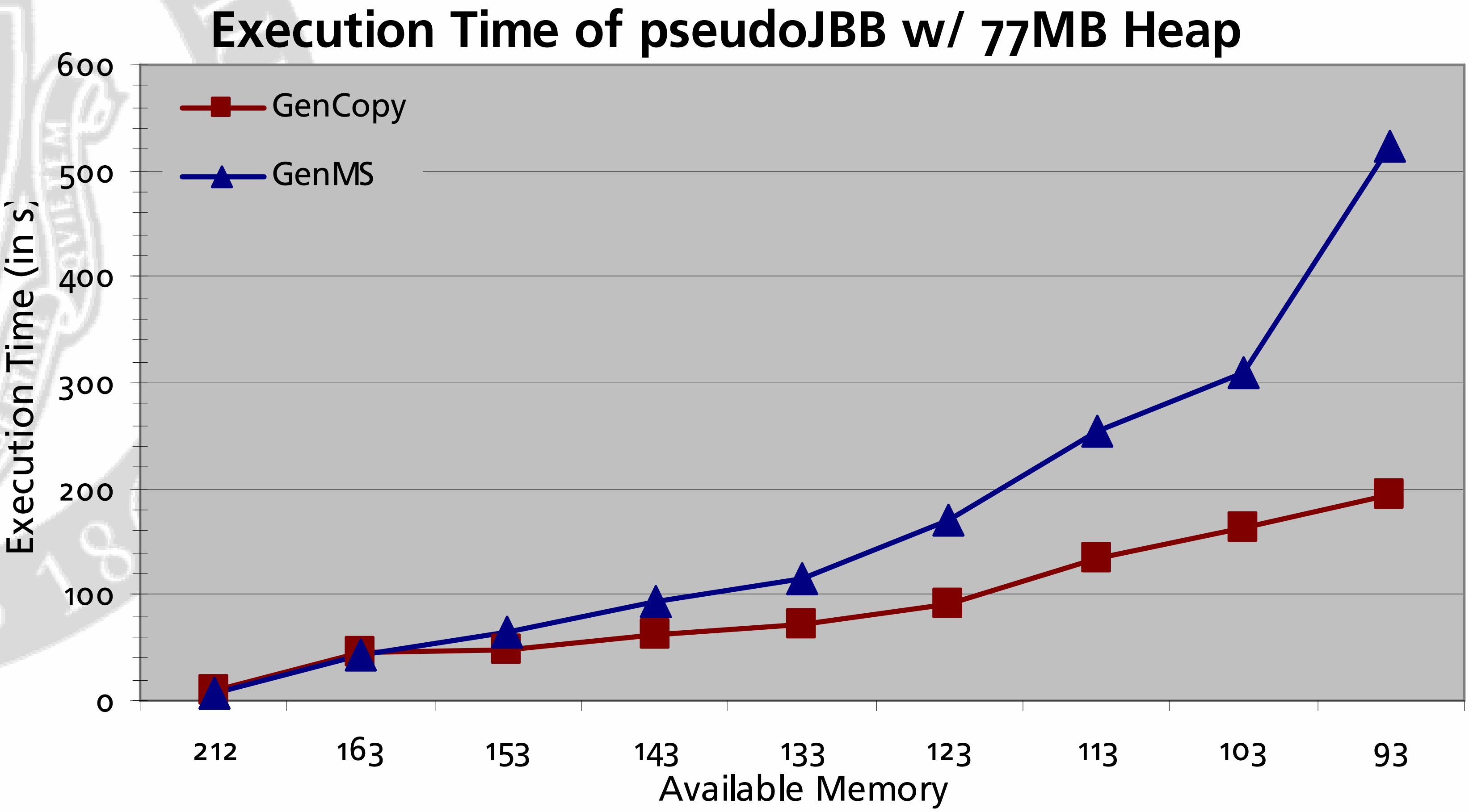
... leading to the eviction of yet another page…

GC Performance While Paging



… which eventually triggers more paging.

Program Throughput



Paging causes a 40 to 62-fold increase in runtime

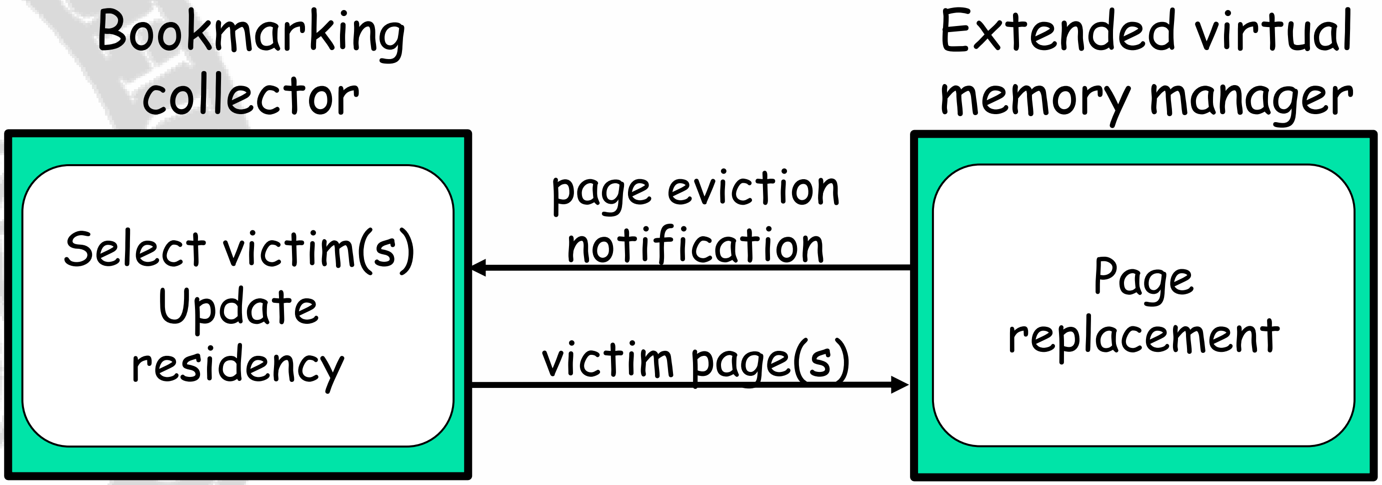
Outline

- Motivation
- GC without paging
  - Cooperative garbage collection
  - Bookmarking
- Results

The problem

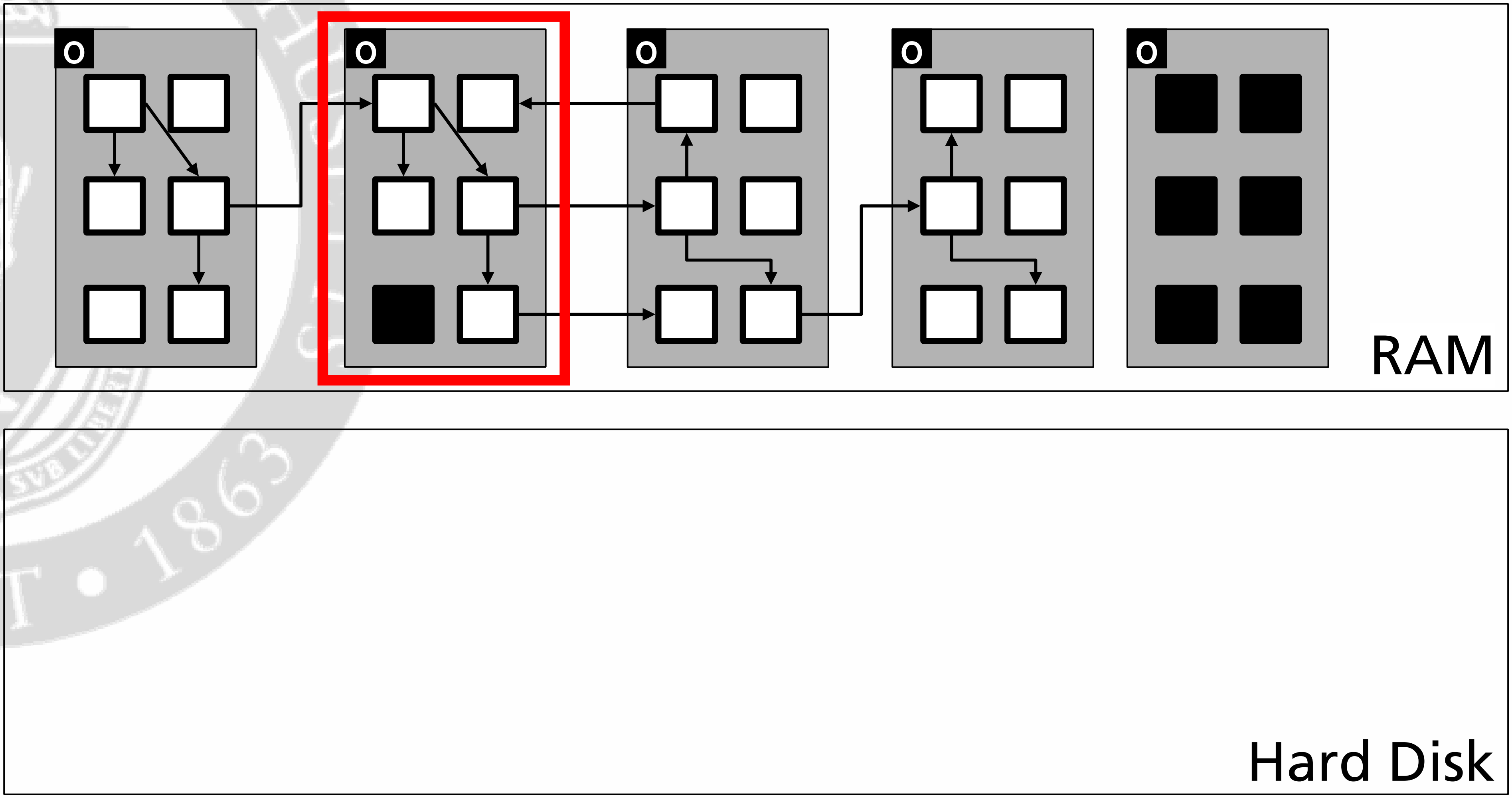
- Garbage collector: **VM-oblivious**
  - Examines all reachable objects
  - Lacks knowledge of page residency
    - Treats evicted and resident pages identically
- Virtual memory: **GC-oblivious**
  - GC & application have different access patterns
    - Likely to evict pages needed by GC

Cooperative Garbage Collection



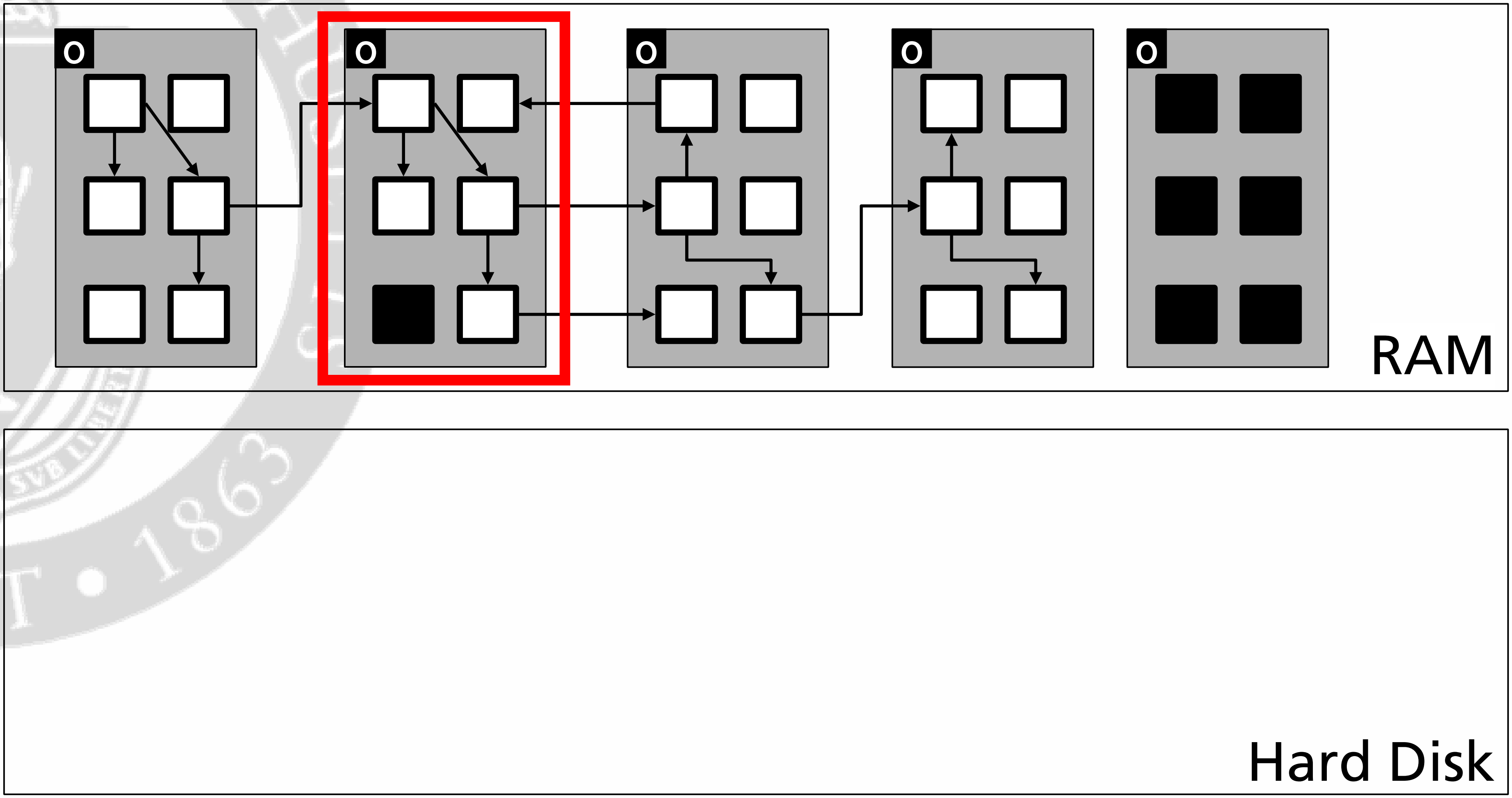
- In response to notifications, BC:
  - Adjusts heap size to fit in main memory
  - Prevents eviction of important pages
  - Avoids touching non-resident pages

Avoiding Page Evictions



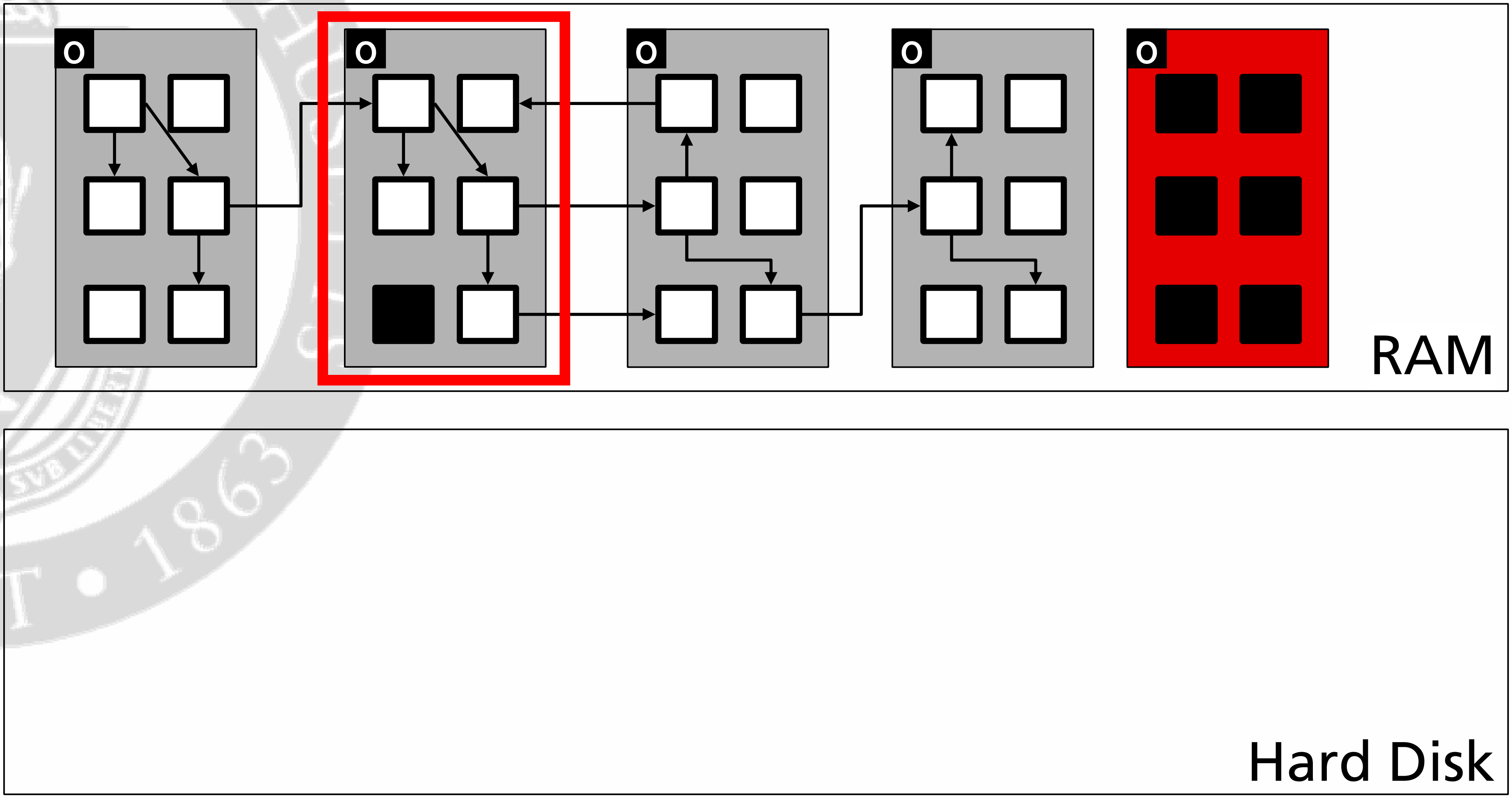
When notified, avoid a pending eviction…

Avoiding Page Evictions



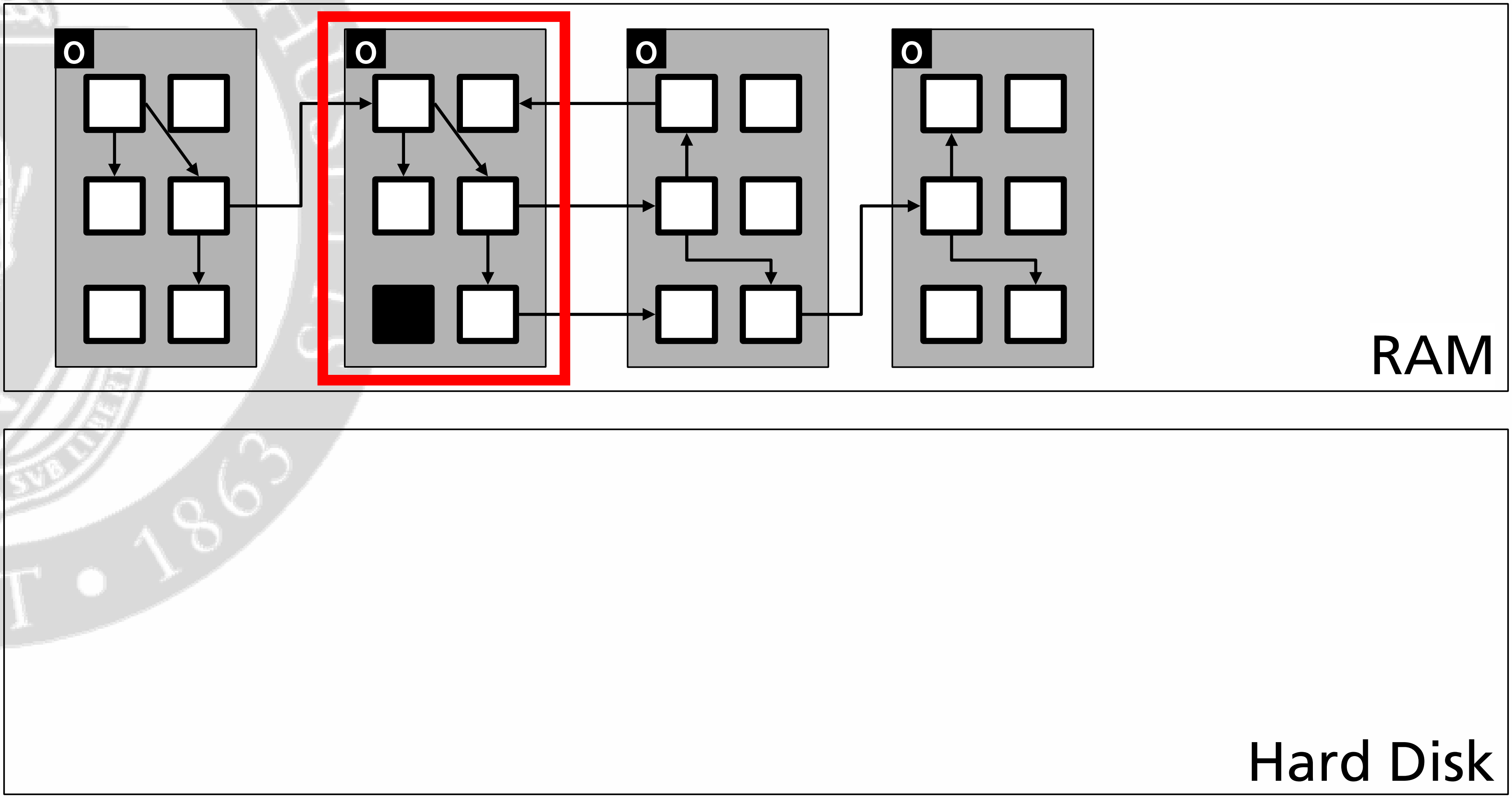
…find a page **BC knows** to be empty…

Avoiding Page Evictions



… and discard it…

Avoiding Page Evictions



… eliminating the need to evict a page.

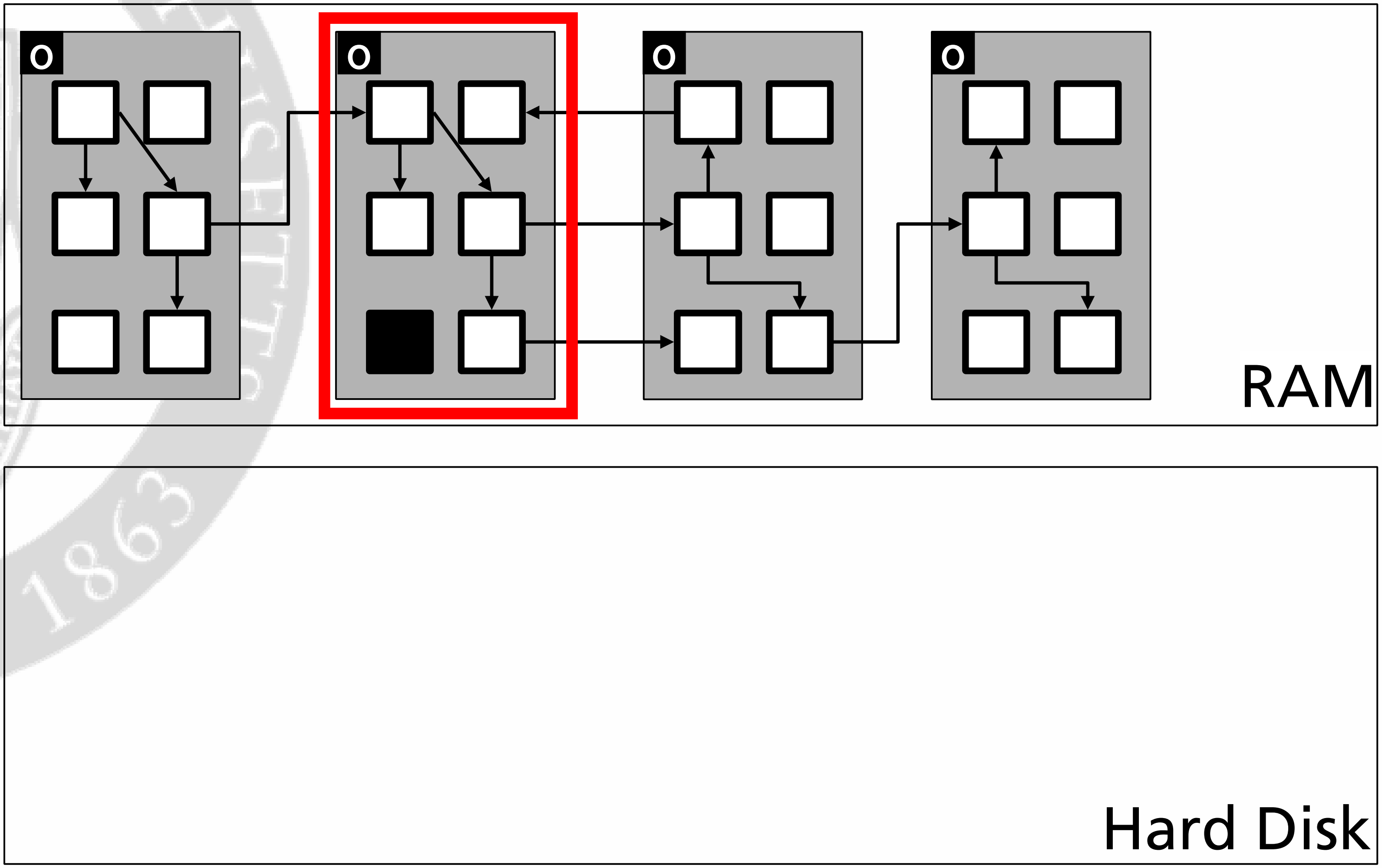
Limit of Heap Sizing

- Could collect, compact, compress, etc.
- Eventually:
  - Will run out of pages to discard
  - Going to have to evict non-empty pages
  - Result: Paging
- Can we avoid this?

Bookmarks

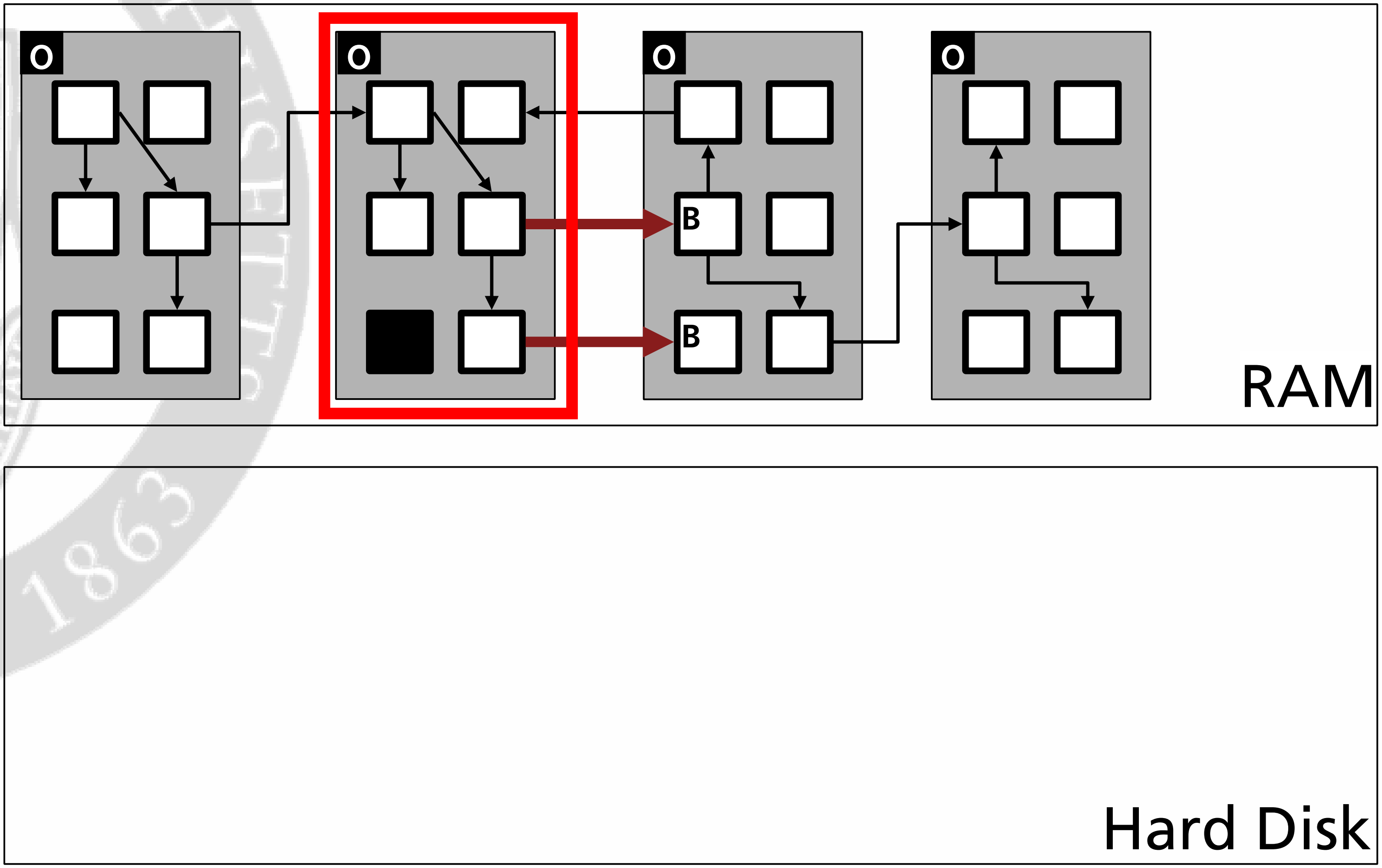
- We introduce *bookmarks:*
  - Summaries of connectivity info on evicted pages
    - References **from** objects on the page
  - These summaries enable GC w/o paging

Bookmarking



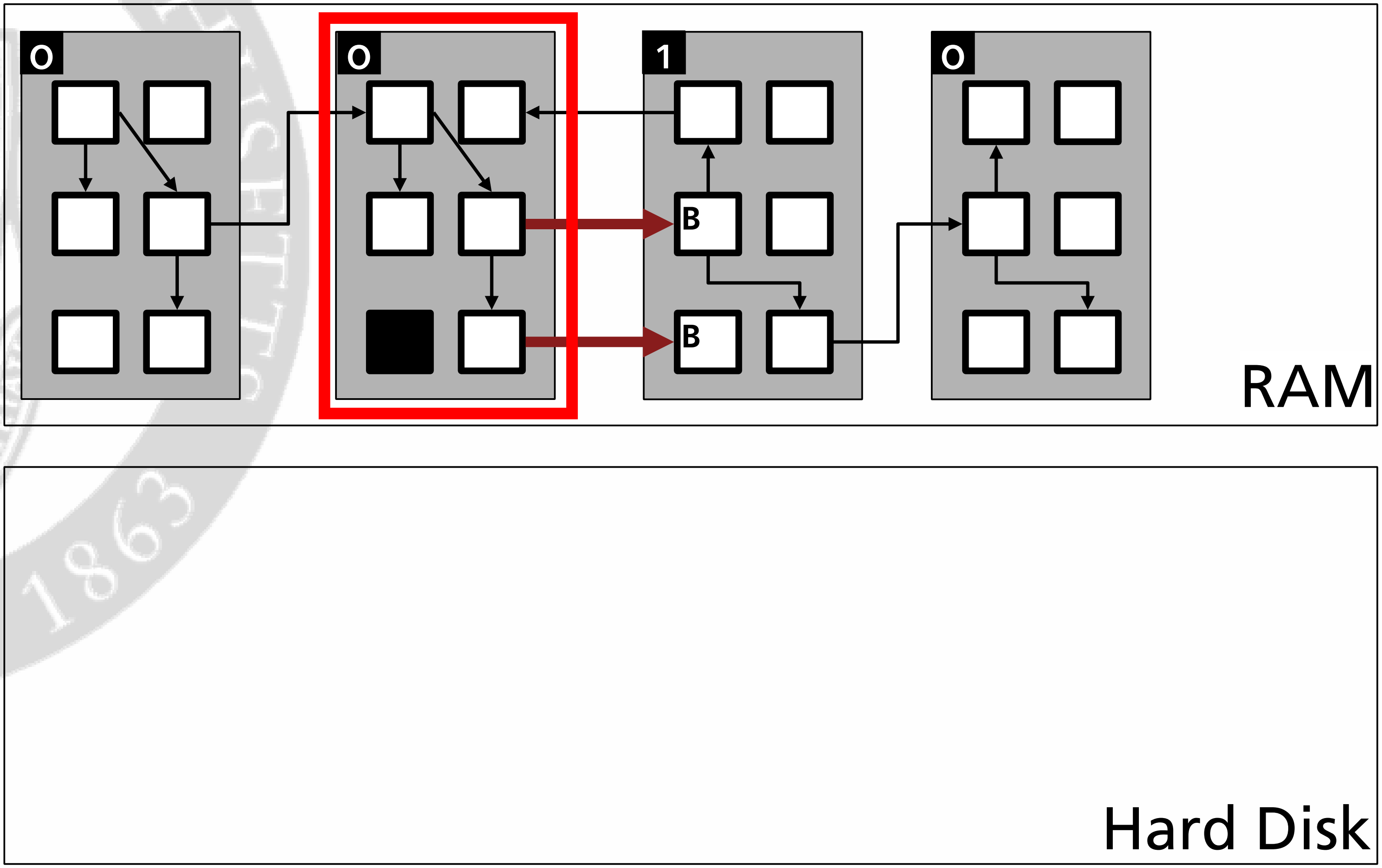
Process page before eviction…

Bookmarking



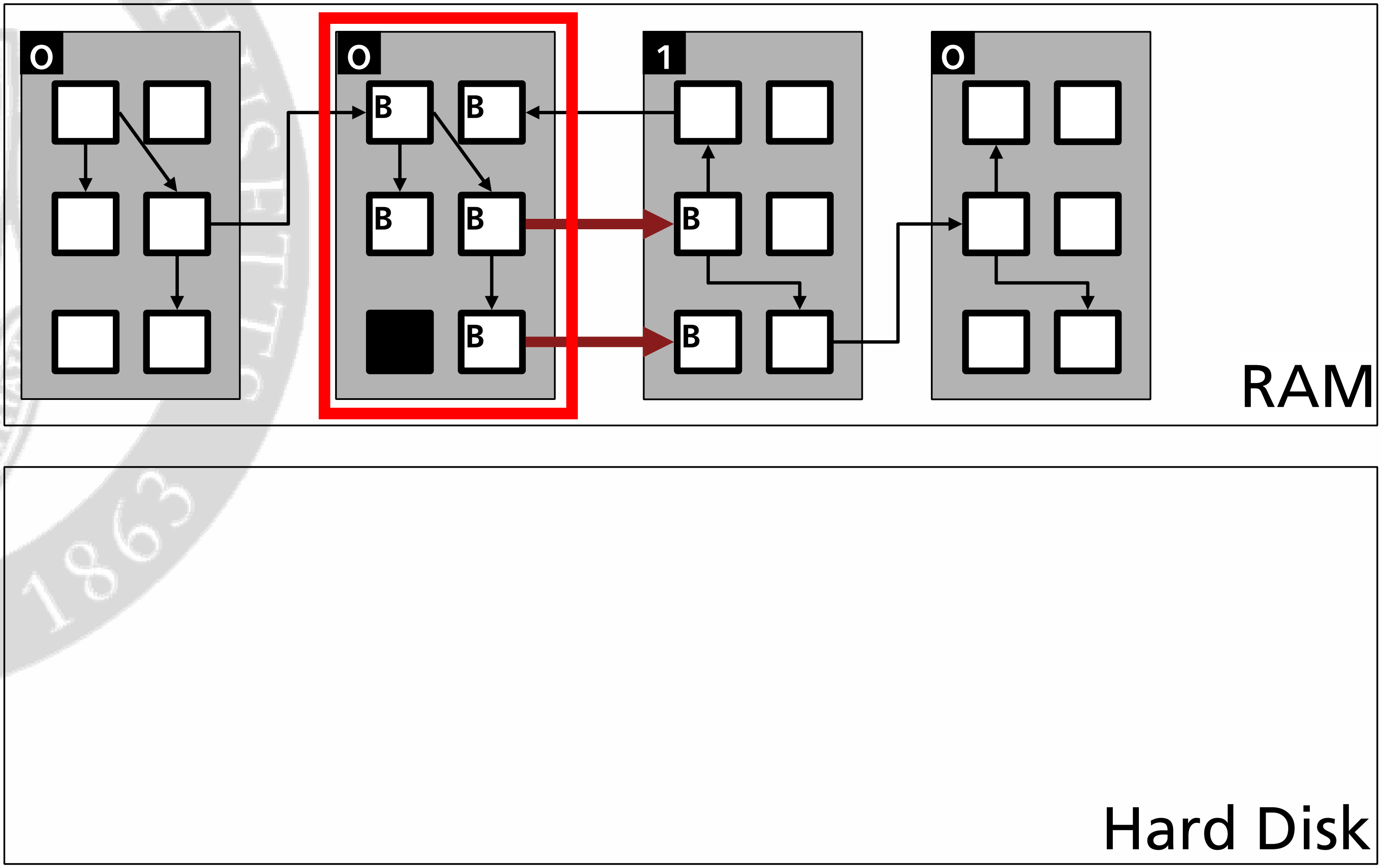
... by following pointers & *bookmark*-ing targets...

Bookmarking



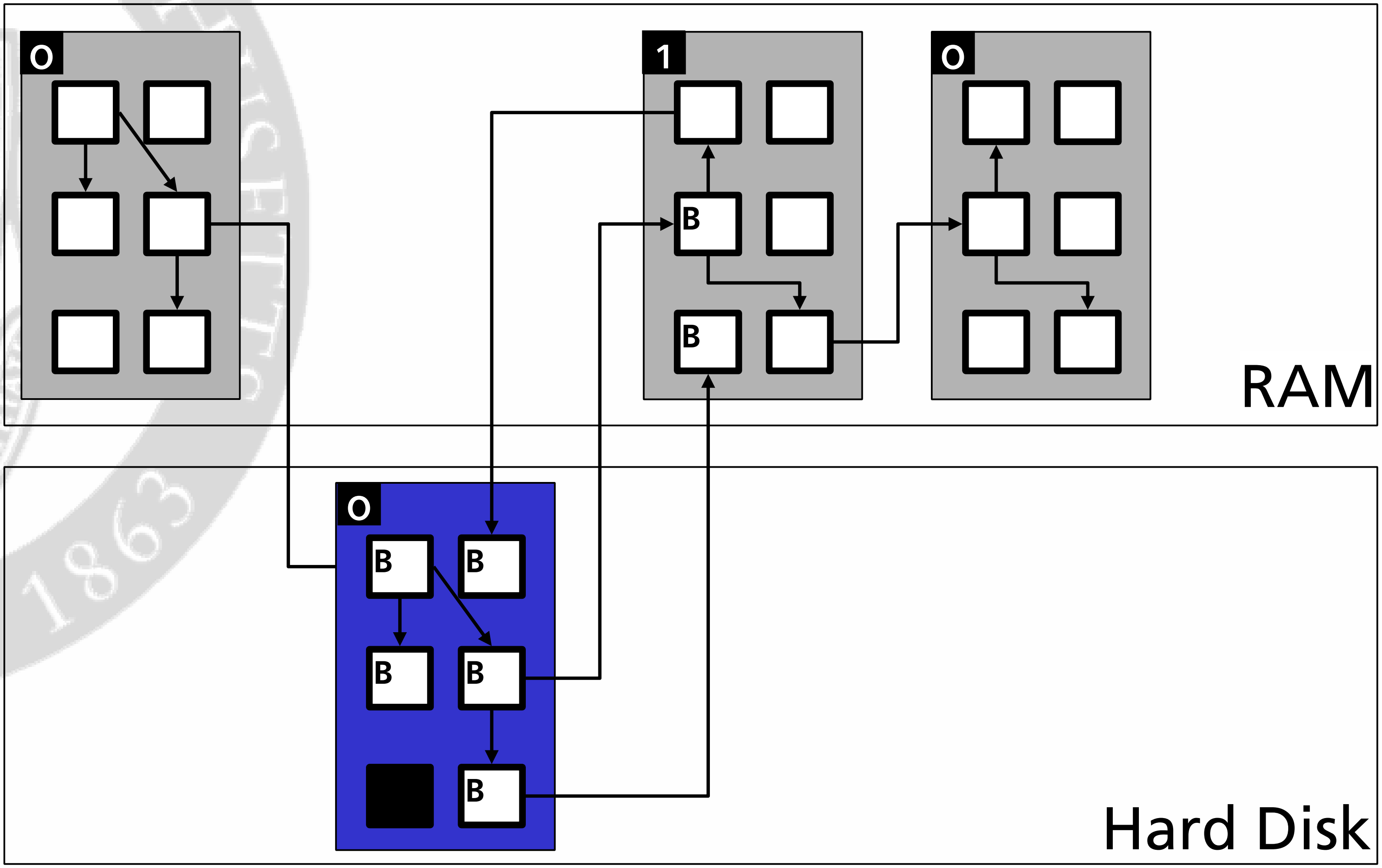
... increment the **referring page count**...

Bookmarking



... conservatively bookmark objects on the page...

Bookmarking

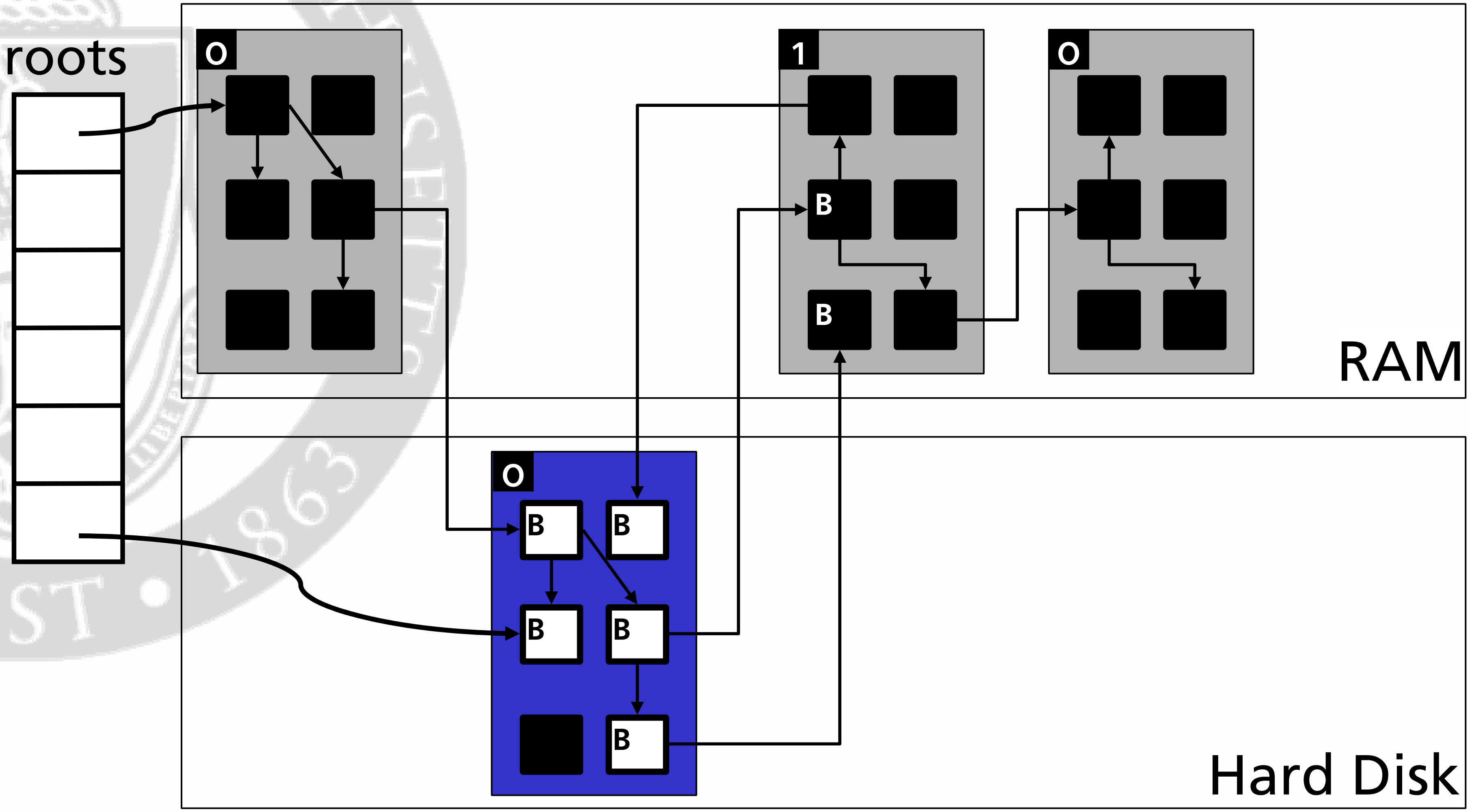


... then tell extended VM to evict the page.

Bookmarking Details

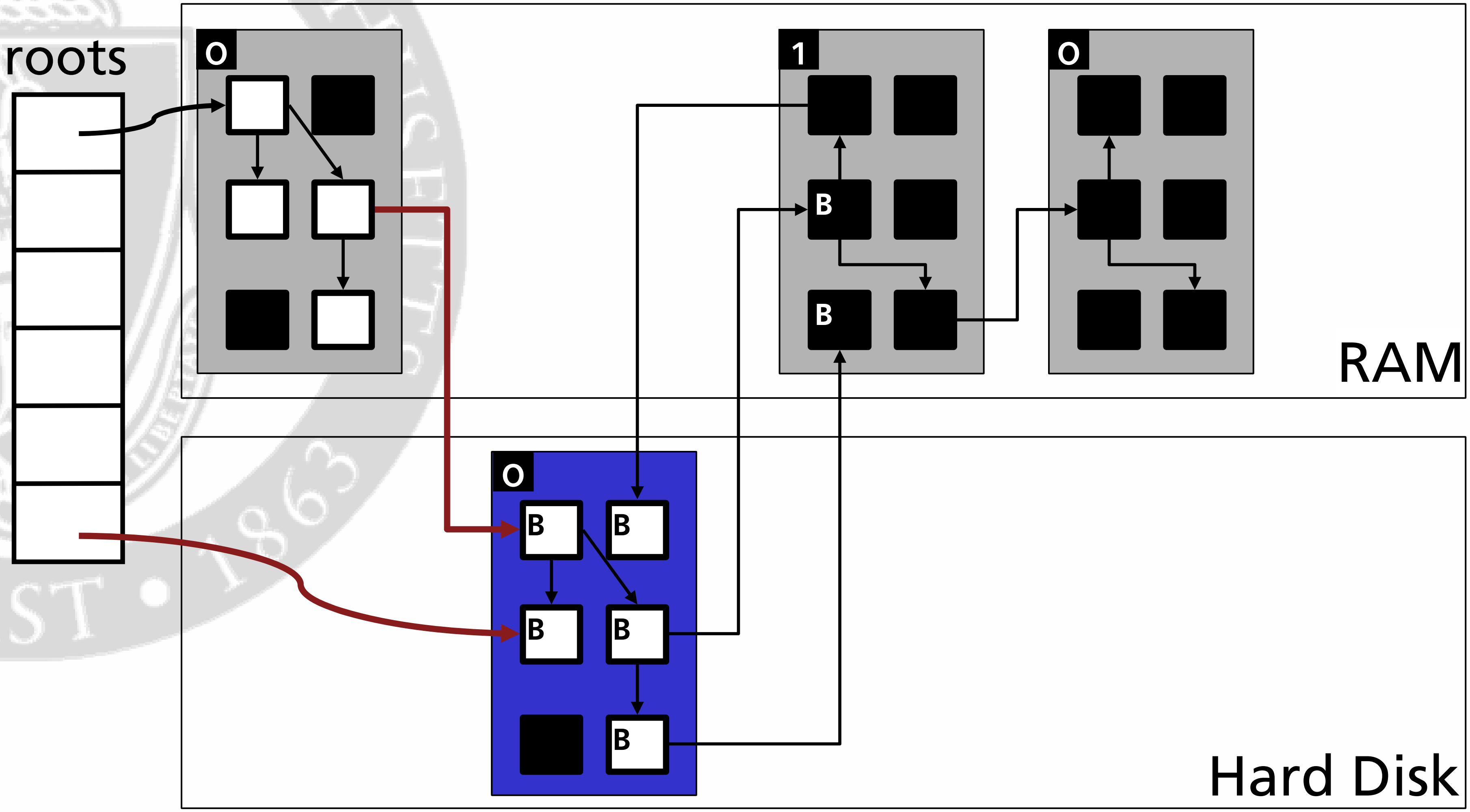
- Cheap summary of connectivity
  - One bit per object: free
  - One word per page: referring page count
  - Bookmarks cleared when count = zero
- Use bookmarks as **secondary roots** during garbage collection

Collection with Bookmarks



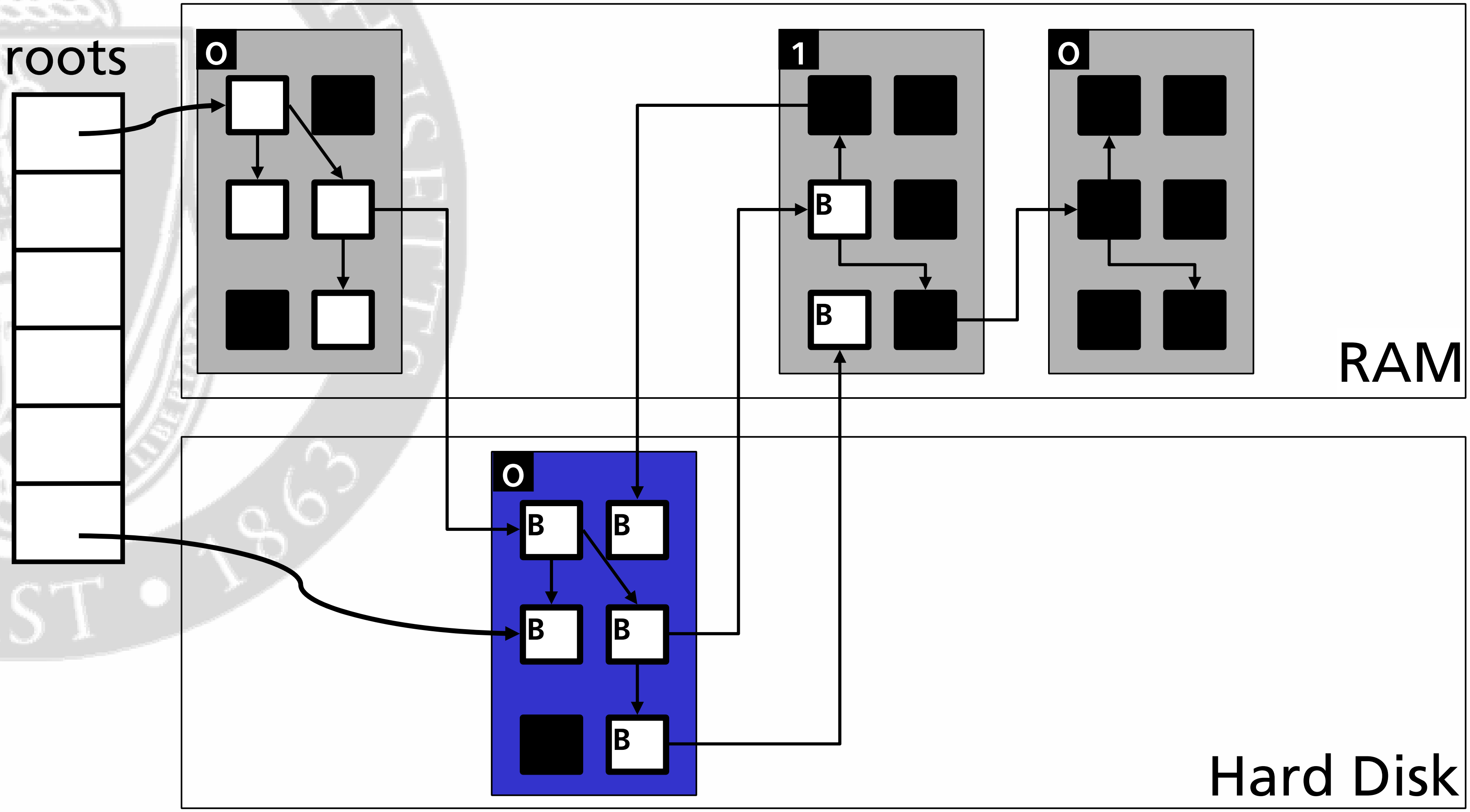
Process objects as usual, but…

Collection with Bookmarks



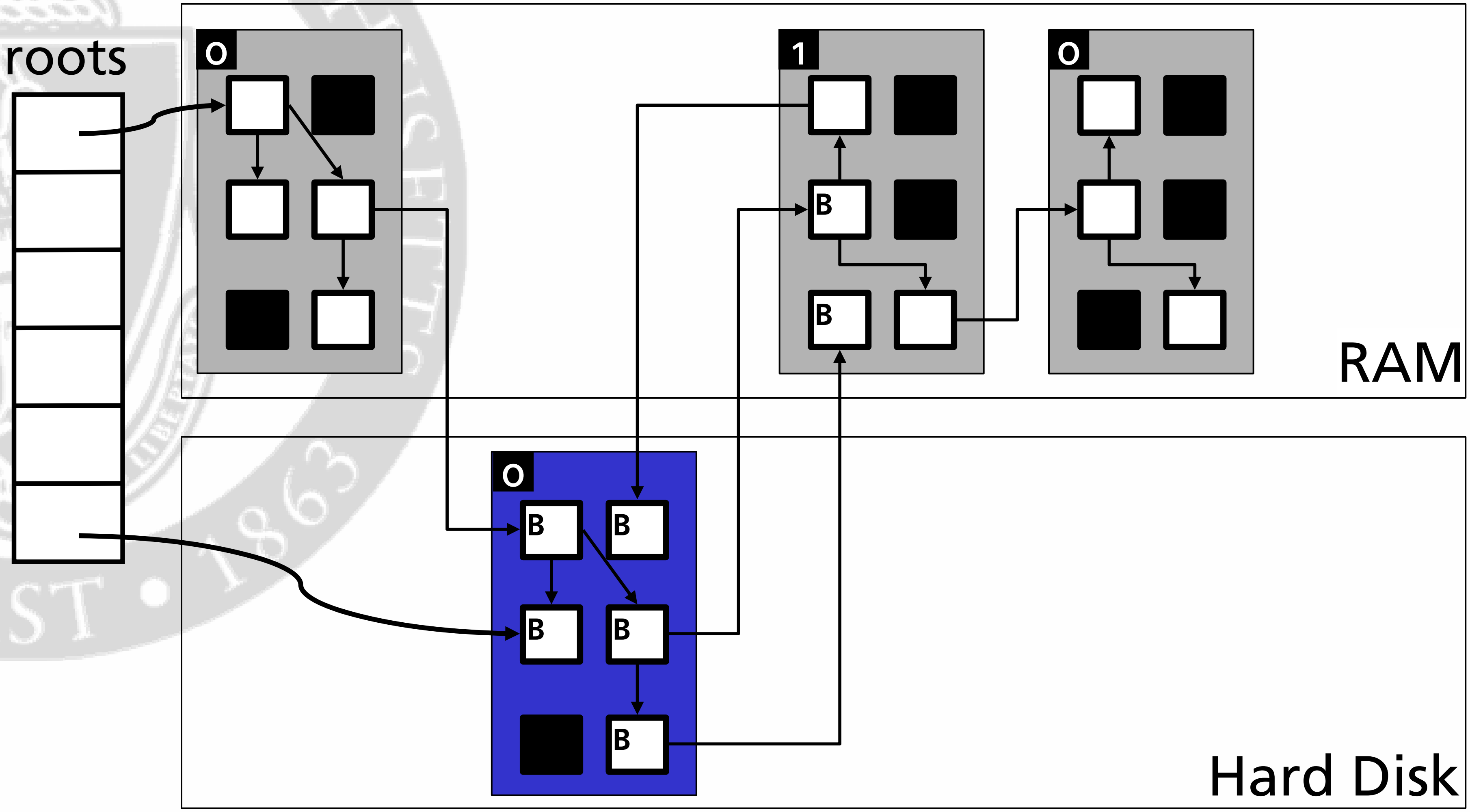
... ignore any references to evicted pages.

Collection with Bookmarks



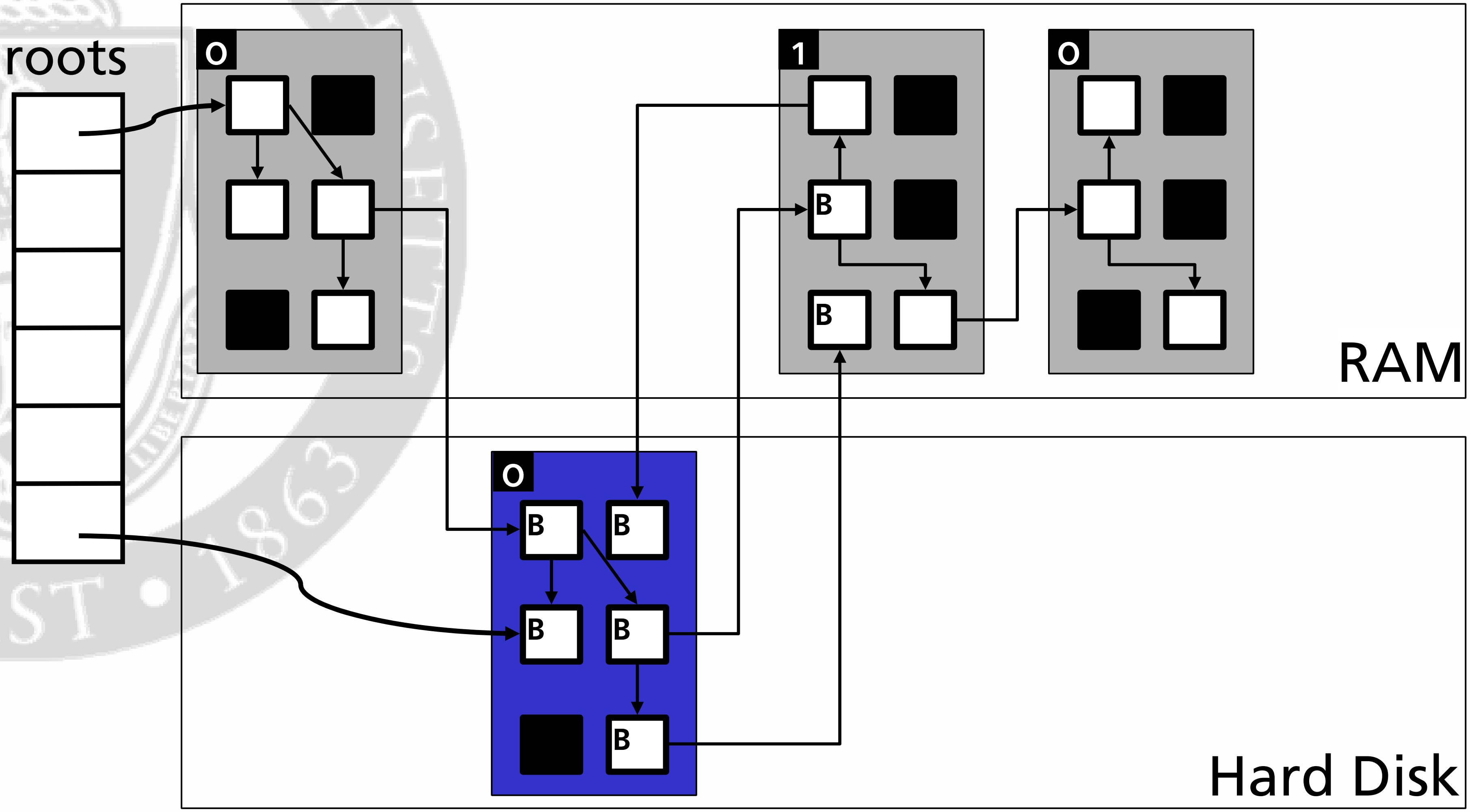
Use bookmarks to recreate evicted references…

Collection with Bookmarks



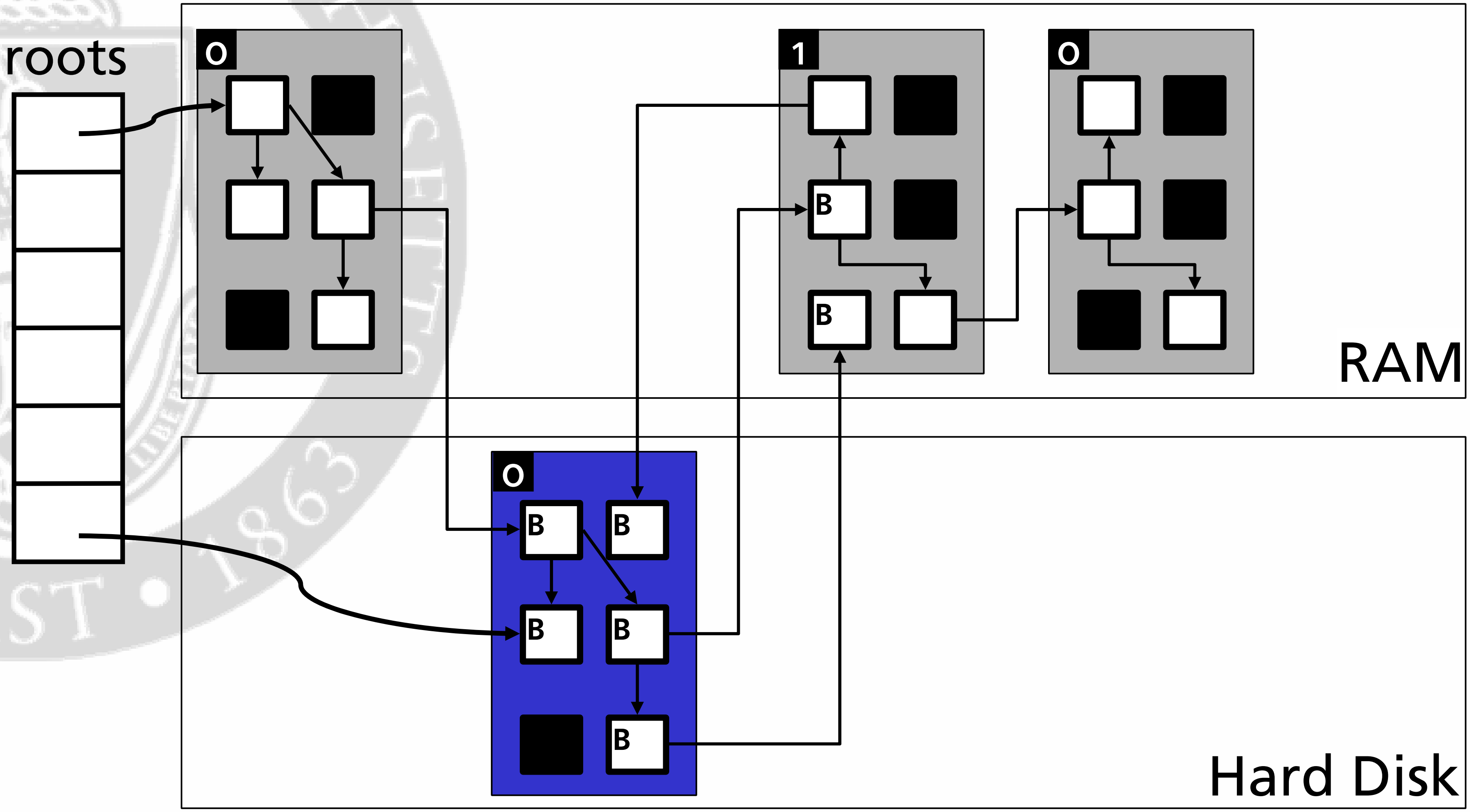
... and continue collection.

Collection with Bookmarks



Result: Garbage collection **without** paging!

Collection with Bookmarks

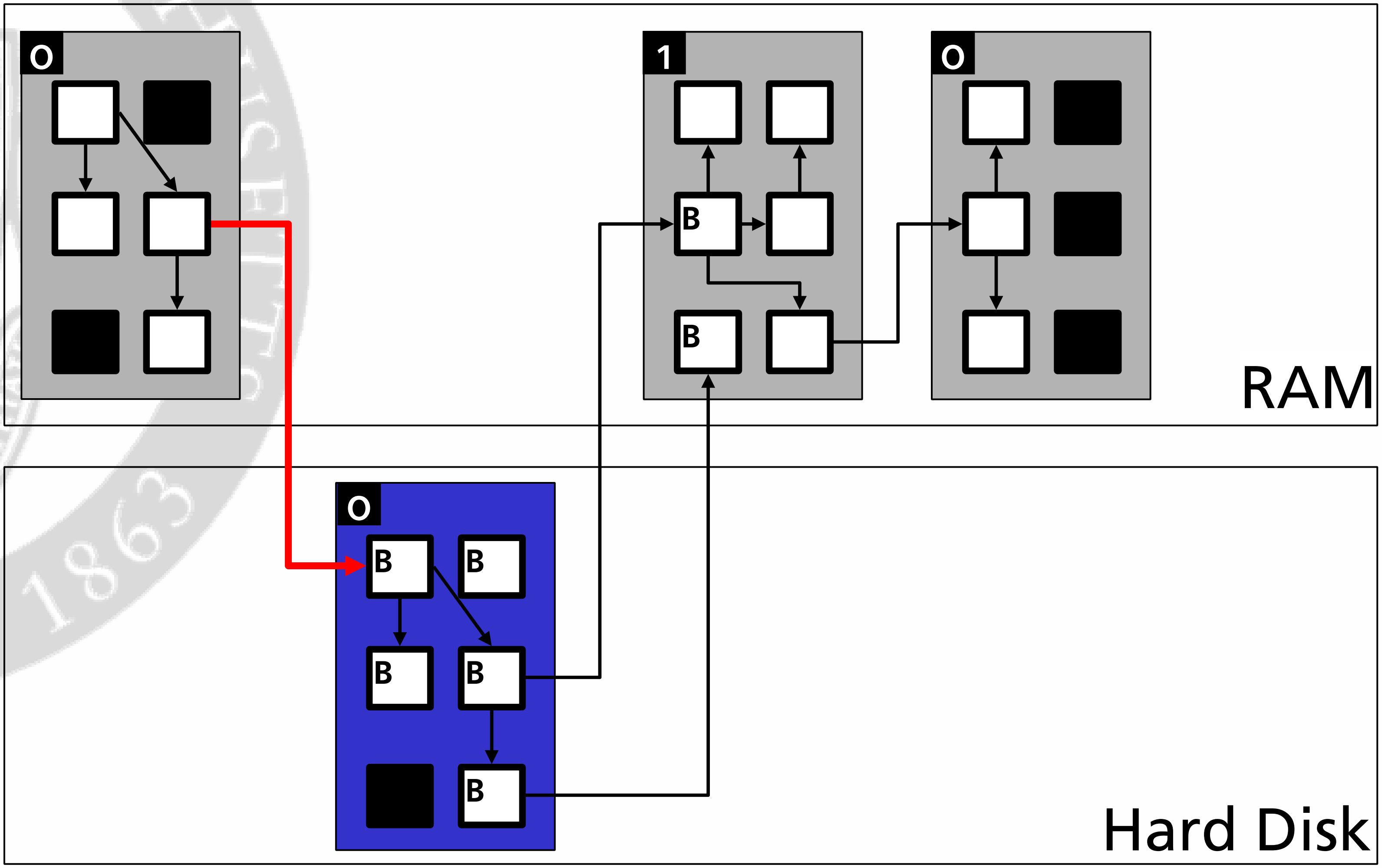


Note: can waste space on evicted pages.

Bookmarking Incompleteness

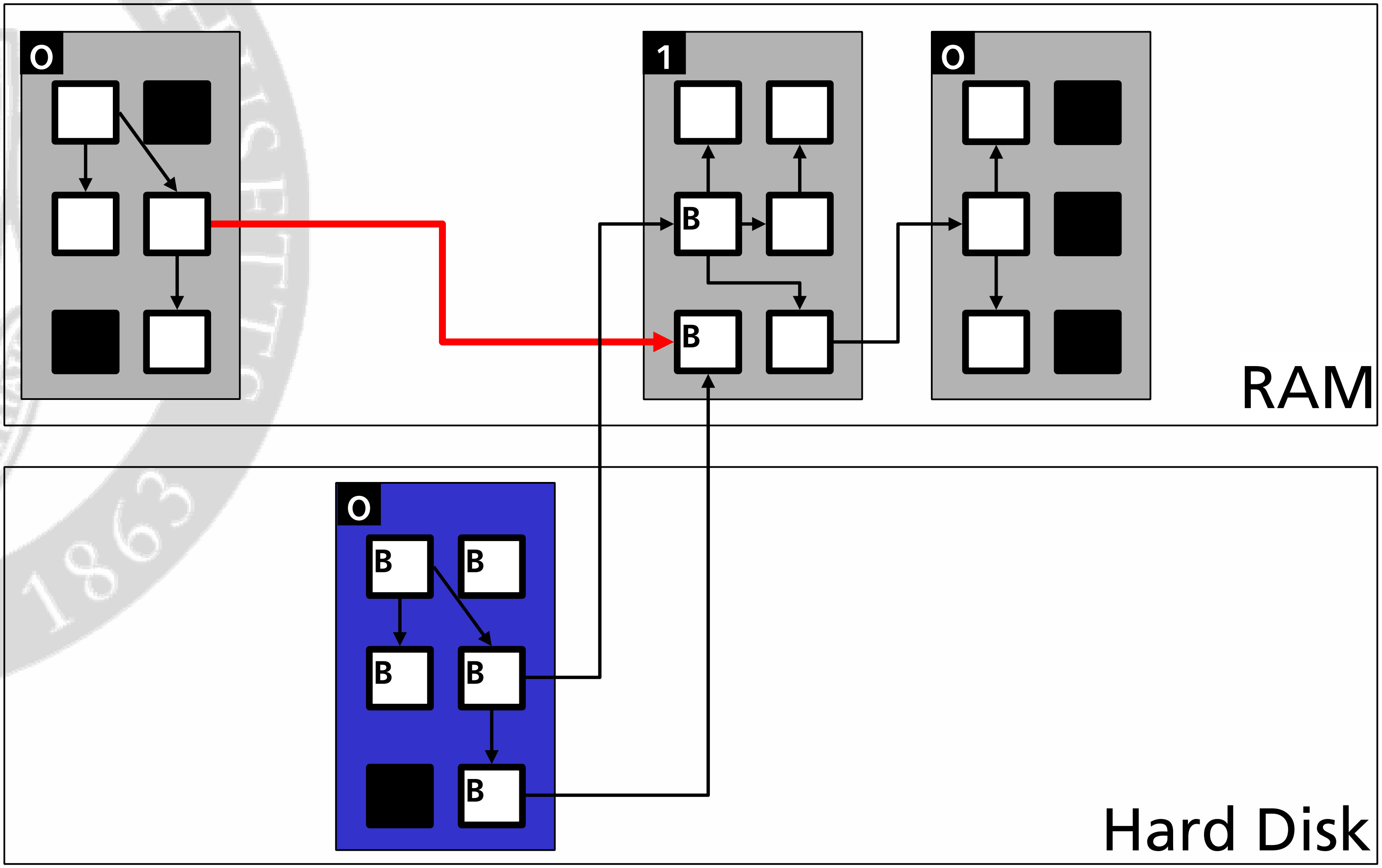
- Space waste not just on evicted pages
- Collection with bookmarks is necessarily *incomplete*
  - Not guaranteed to reclaim all memory

Bookmarking Incompleteness



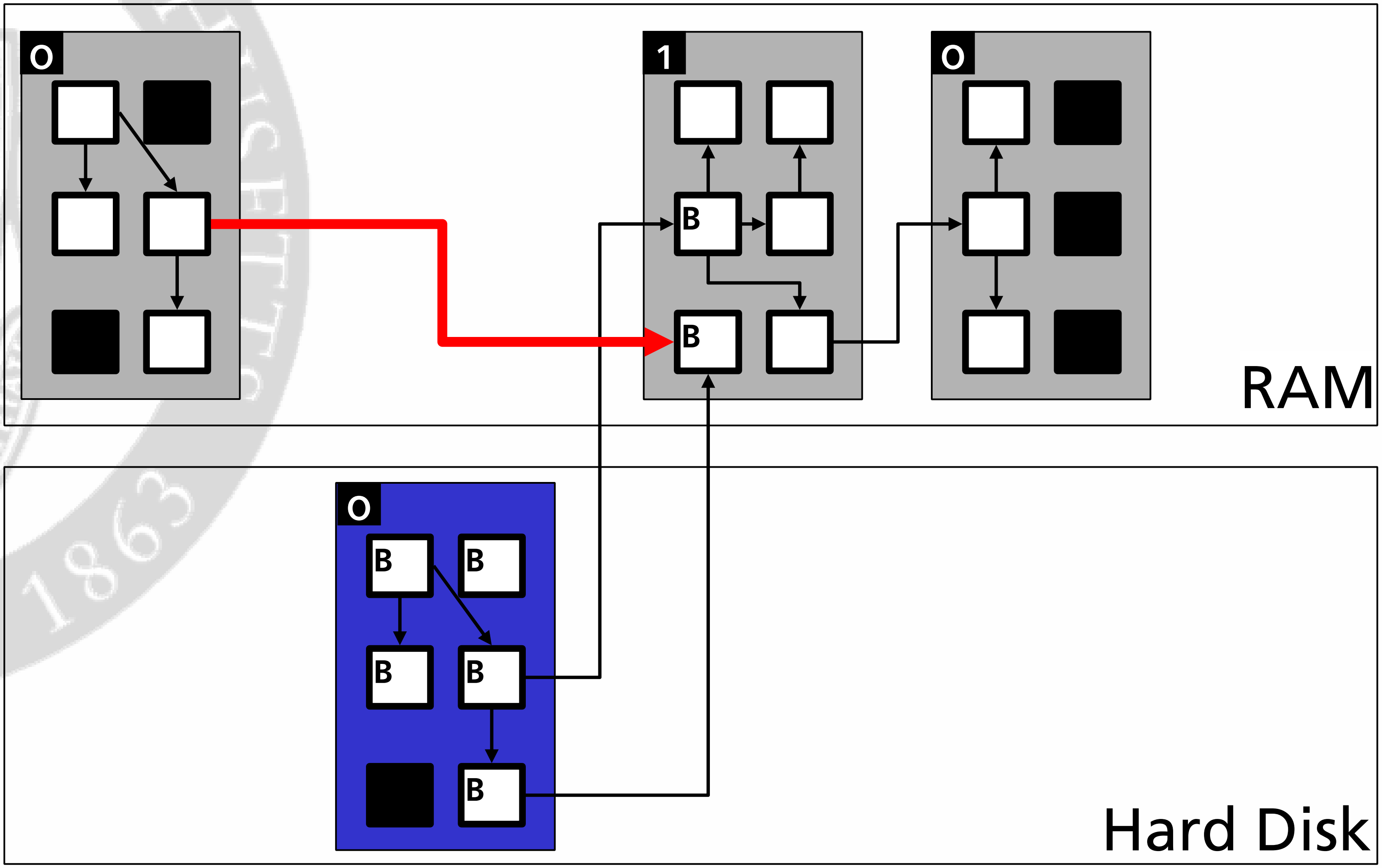
When a reference to an evicted object changes…

Bookmarking Incompleteness



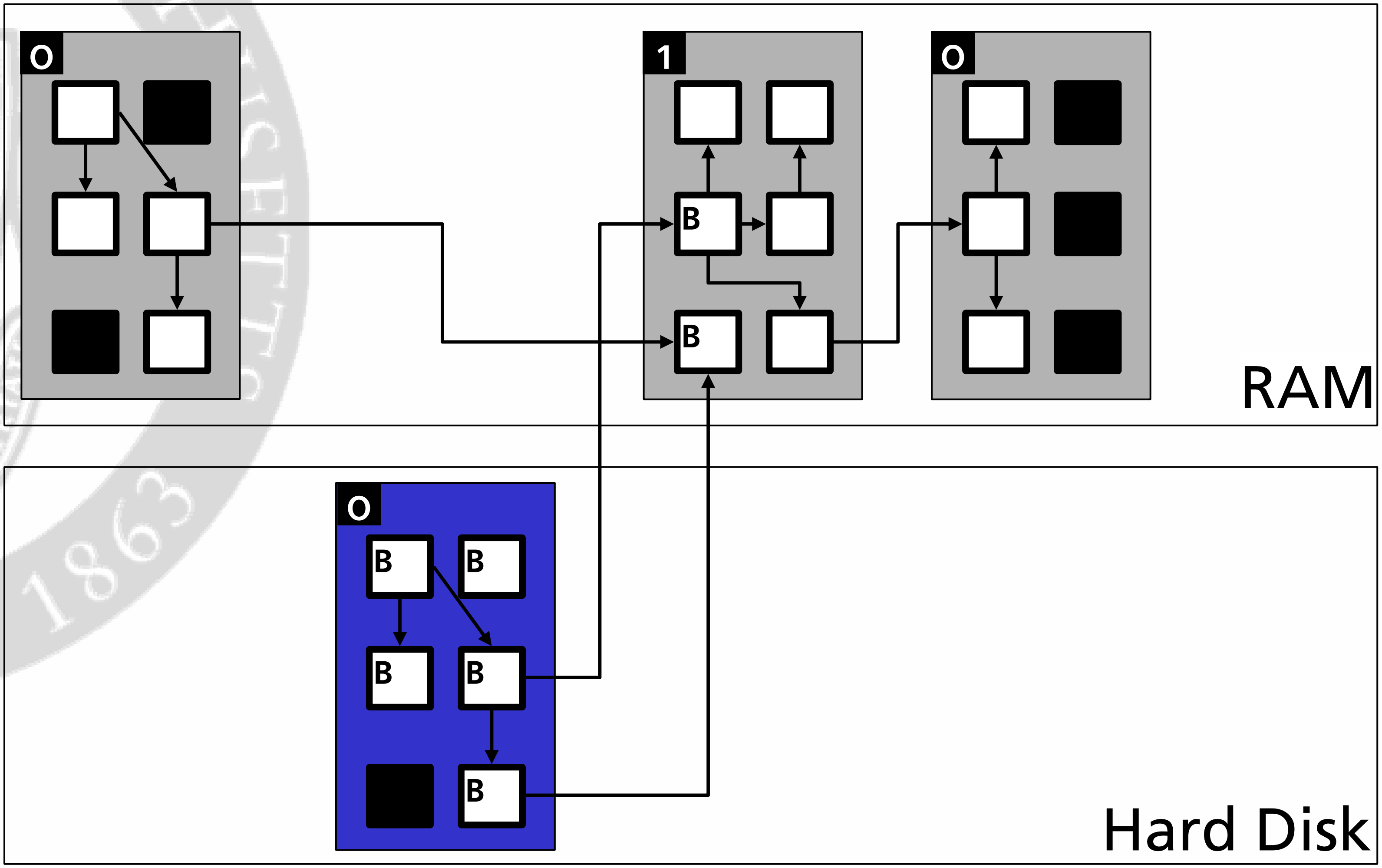
When a reference to an evicted object changes…

Bookmarking Incompleteness



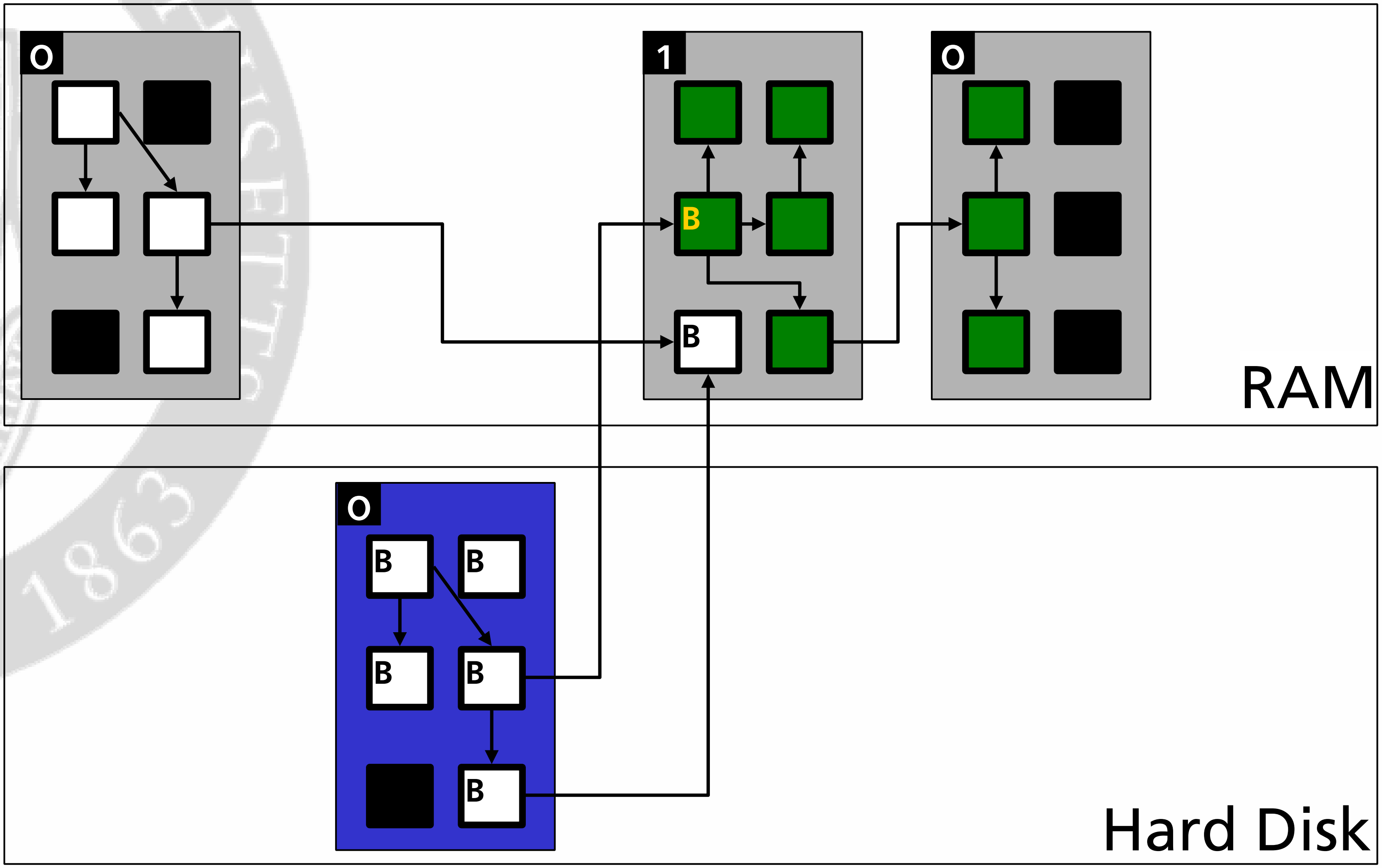
…it can make evicted objects **unreachable**.

Bookmarking Incompleteness



But bookmarks cannot be removed…

Bookmarking Incompleteness



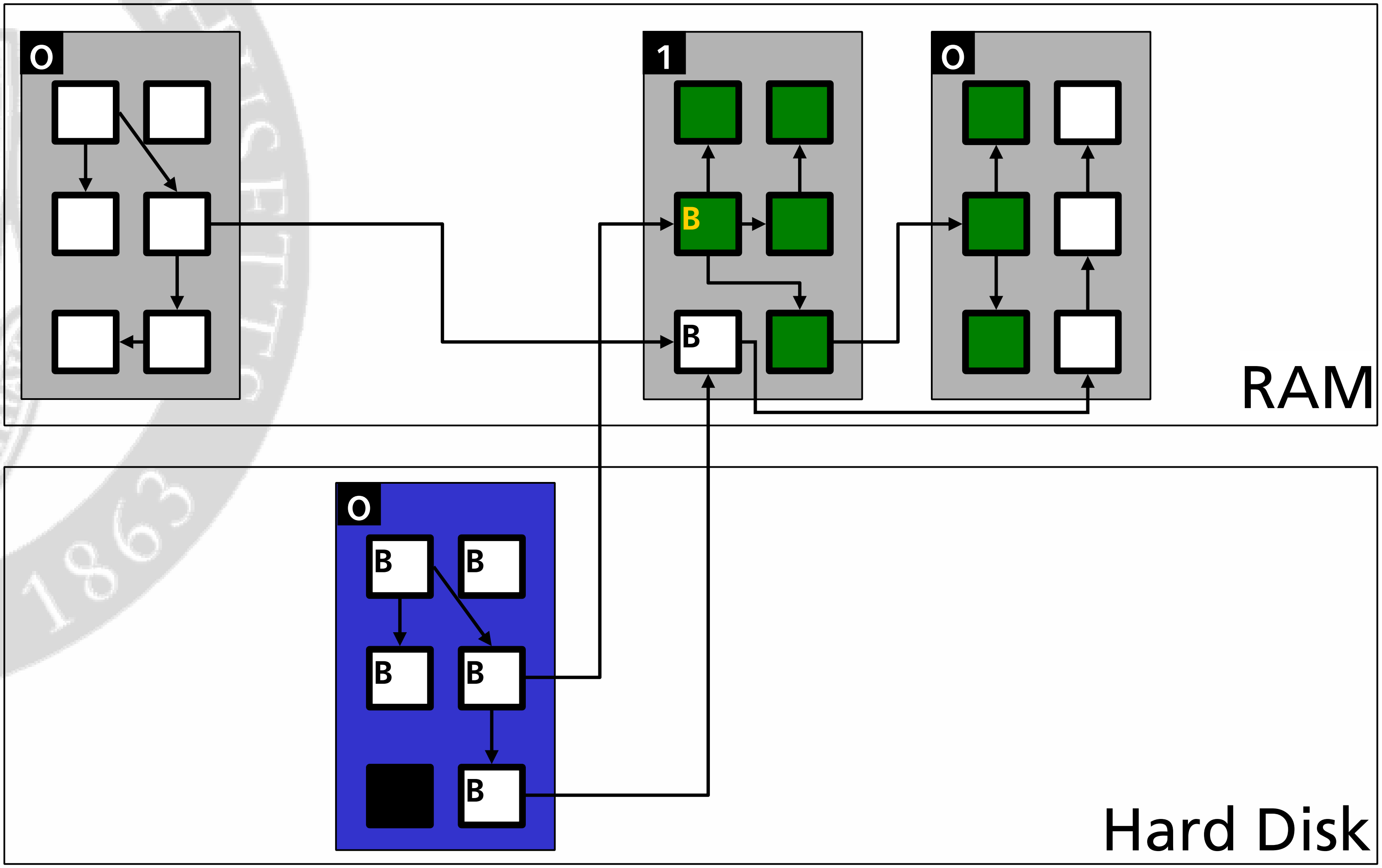
…retaining unreachable heap objects.

Bookmarking Completeness

- Worst-case: completeness requires *duplicating* evicted pages
  - See paper for more info

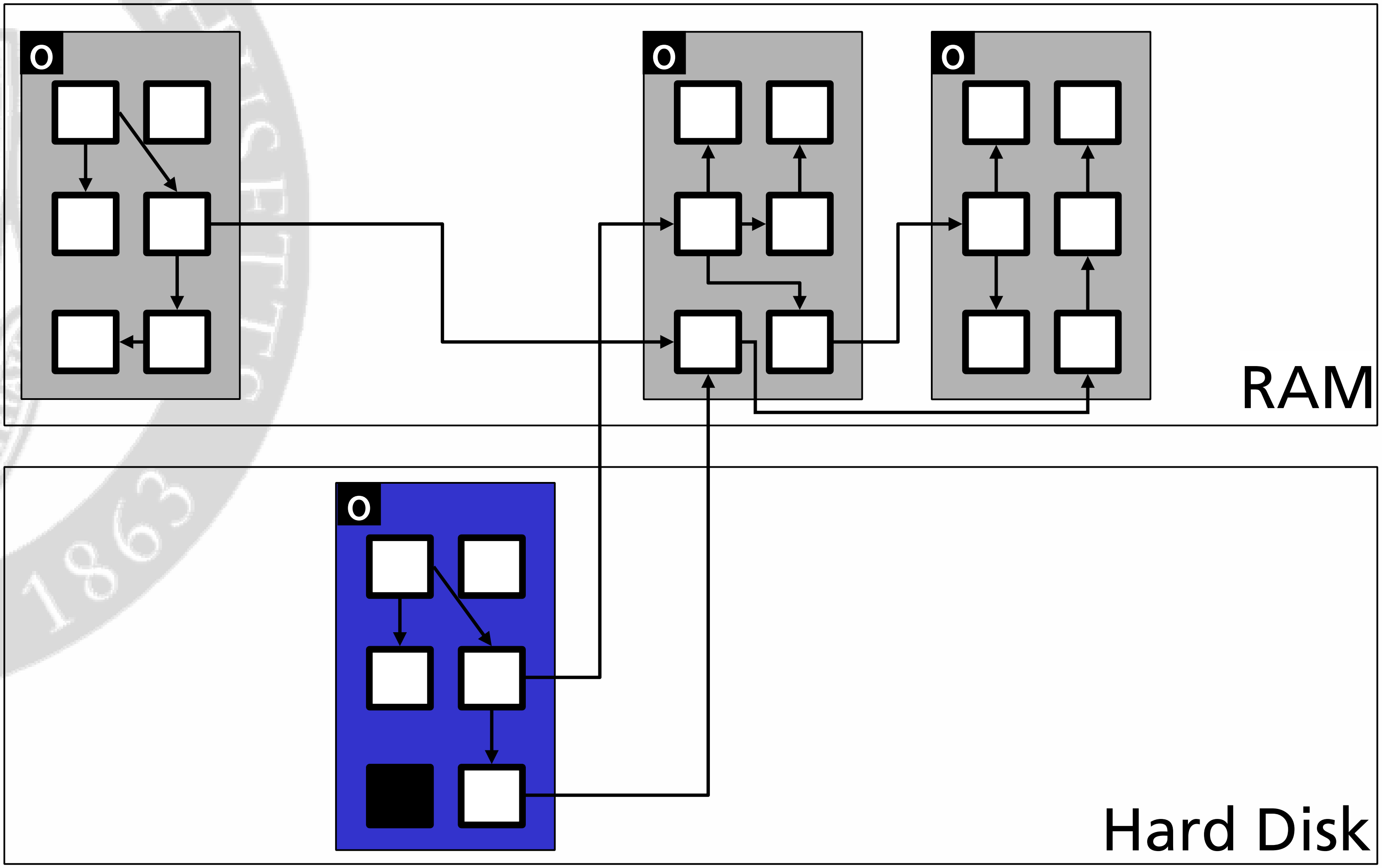
- How can we preserve completeness?

Bookmarking Completeness



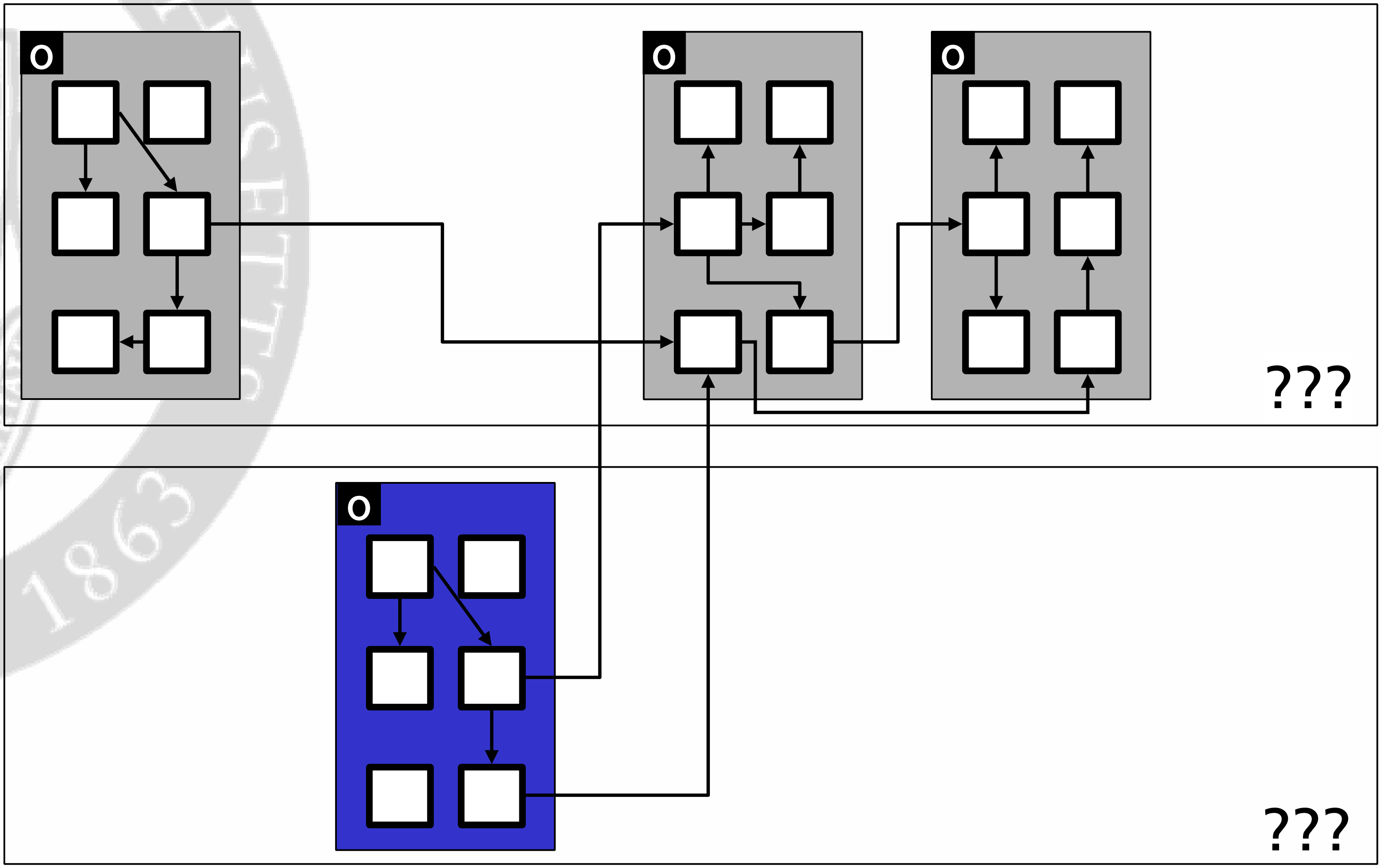
If the heap becomes full…

Bookmarking Completeness



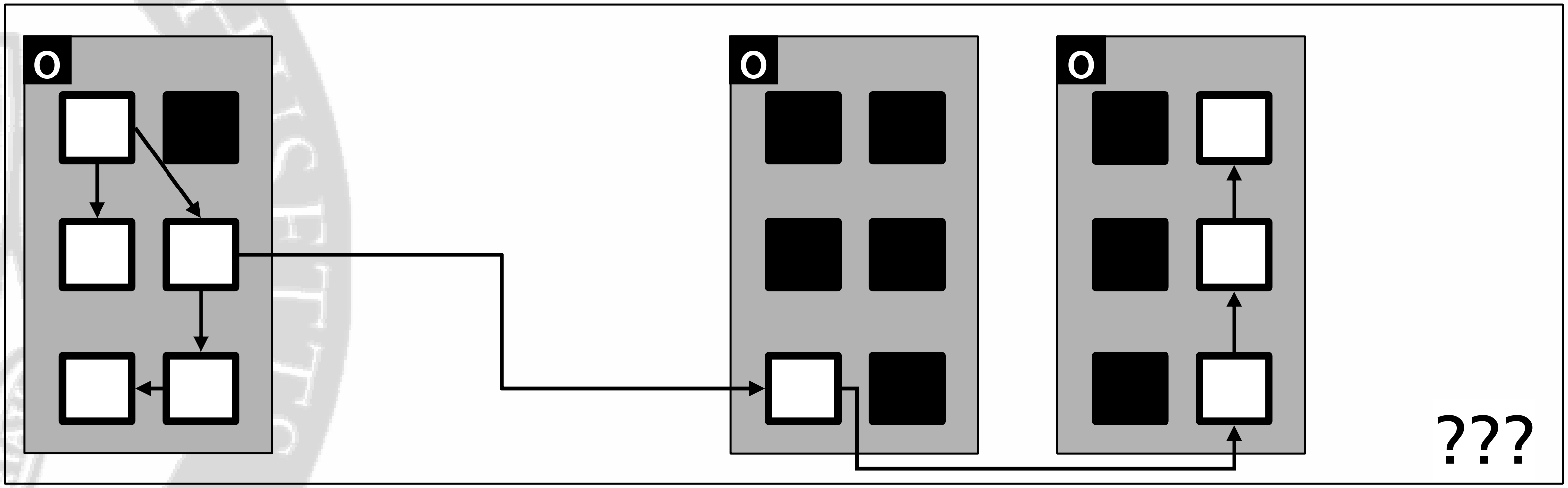
…BC removes all bookmarks…

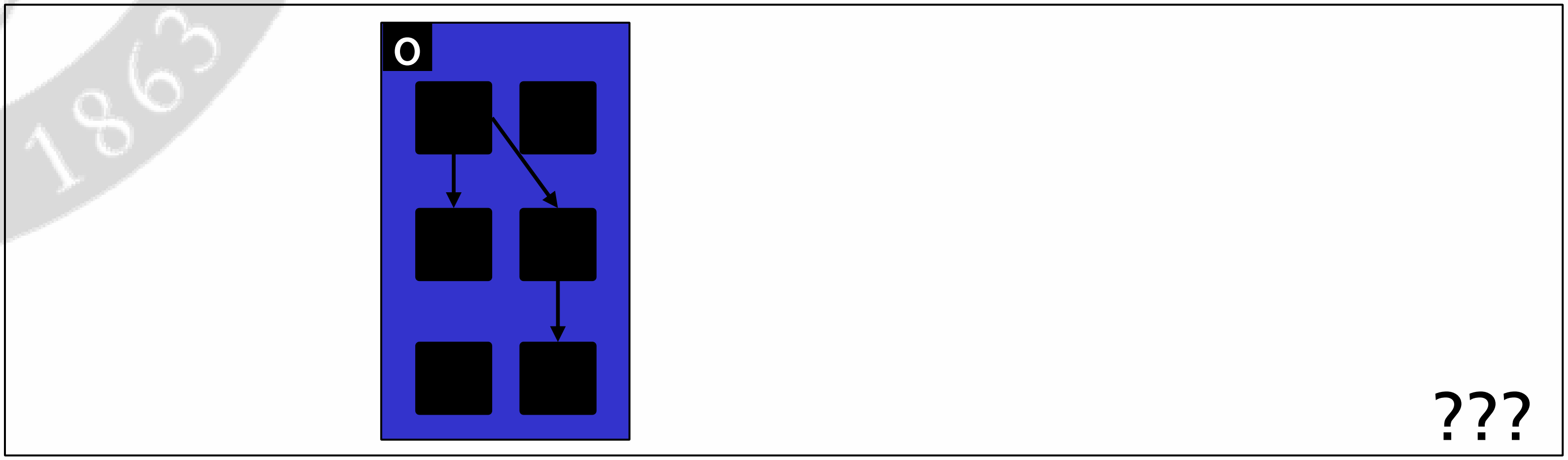
Bookmarking Completeness



…and performs a VM-oblivious collection…

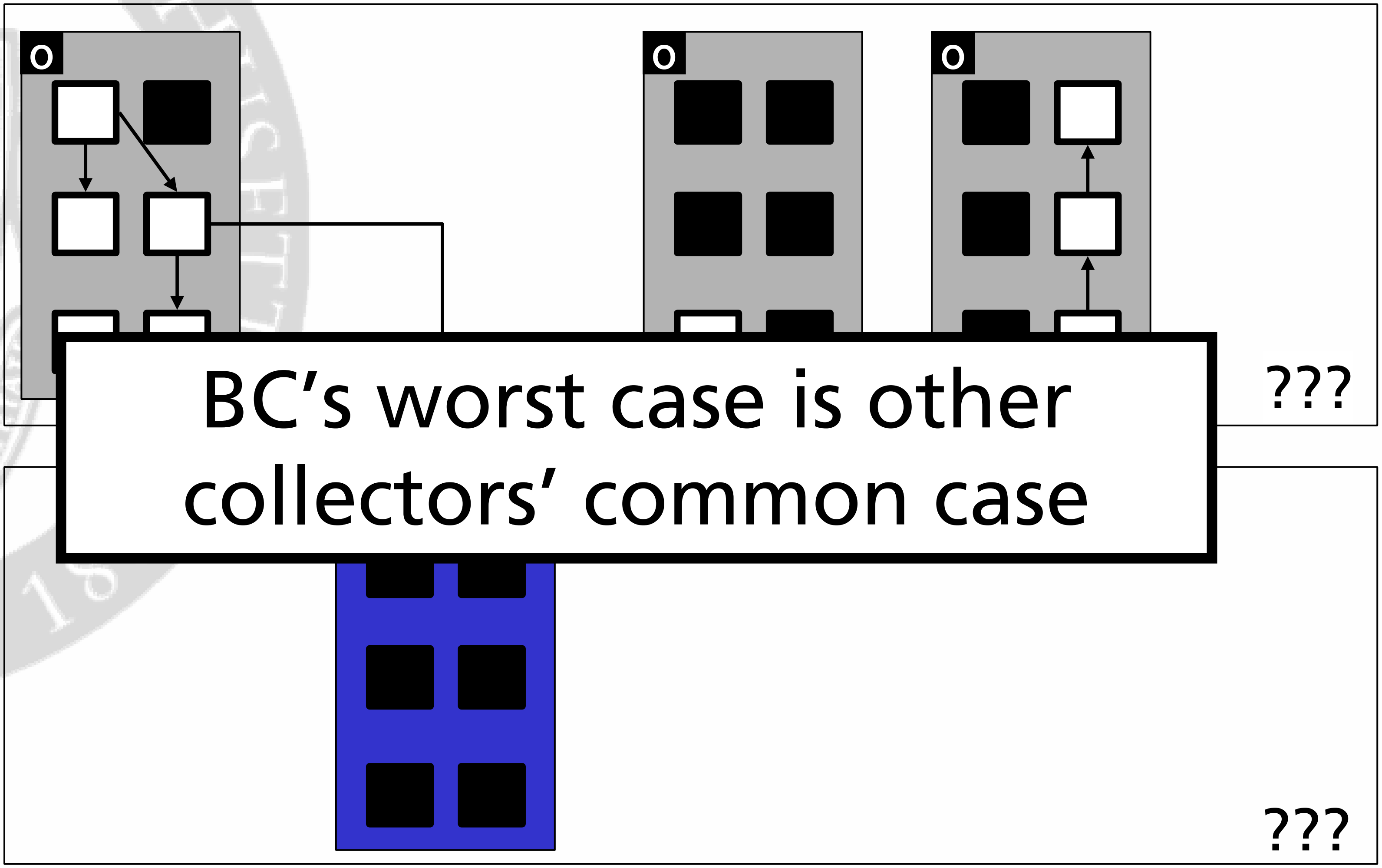
Bookmarking Completeness





…reclaiming all unreachable objects.

Bookmarking Completeness



BC's worst case is other collectors' common case

…reclaiming all unreachable objects.

BC Performance Optimizations

- Uses generational design similar to GenMS
  - Yields good performance when not paging
- Compacts heap to reduce pressure
  - Prevents single object from tying down a page

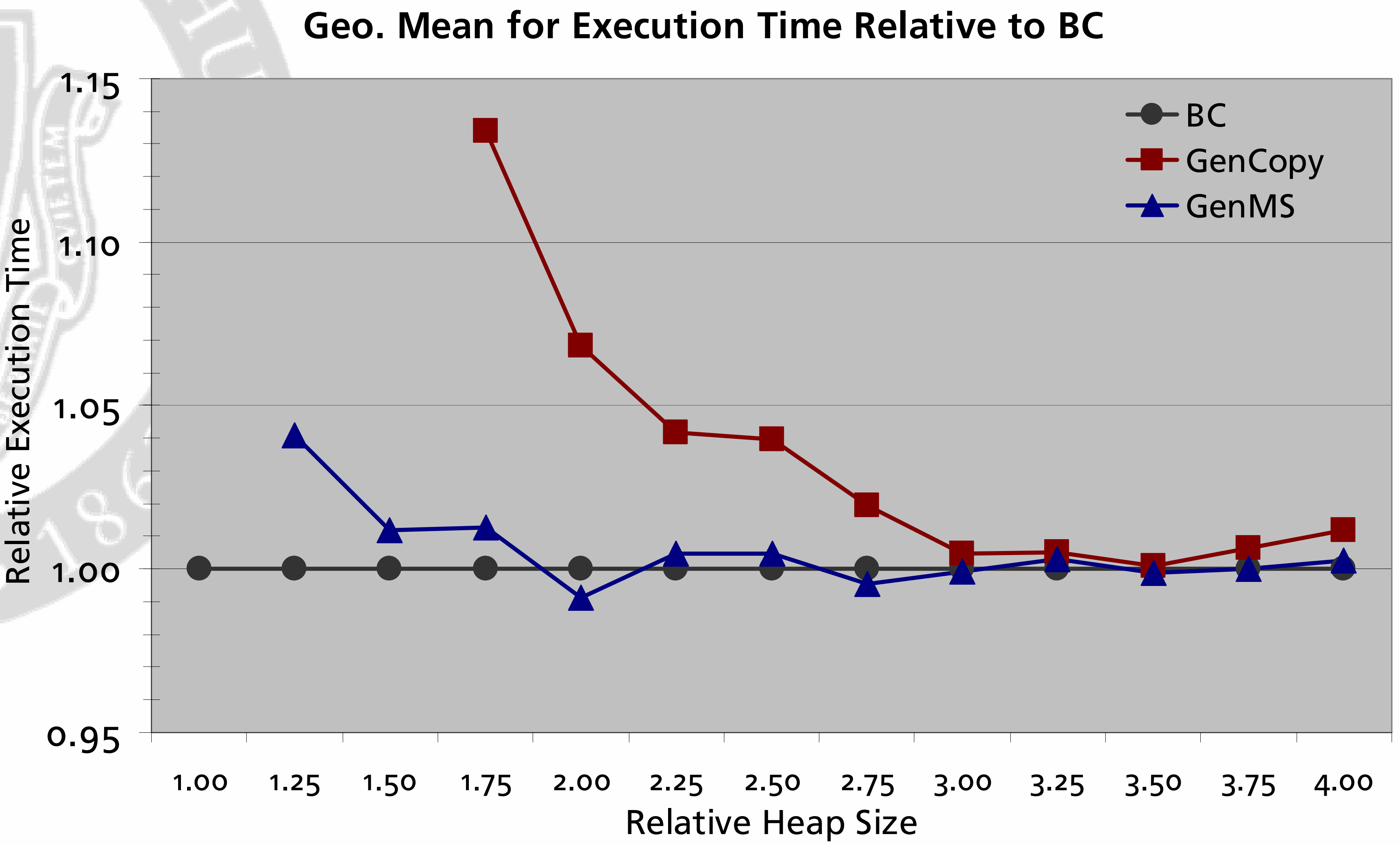
Outline

- Motivation
- GC without paging
  - Cooperative garbage collection
  - Bookmarking
- Results

Experimental Methodology

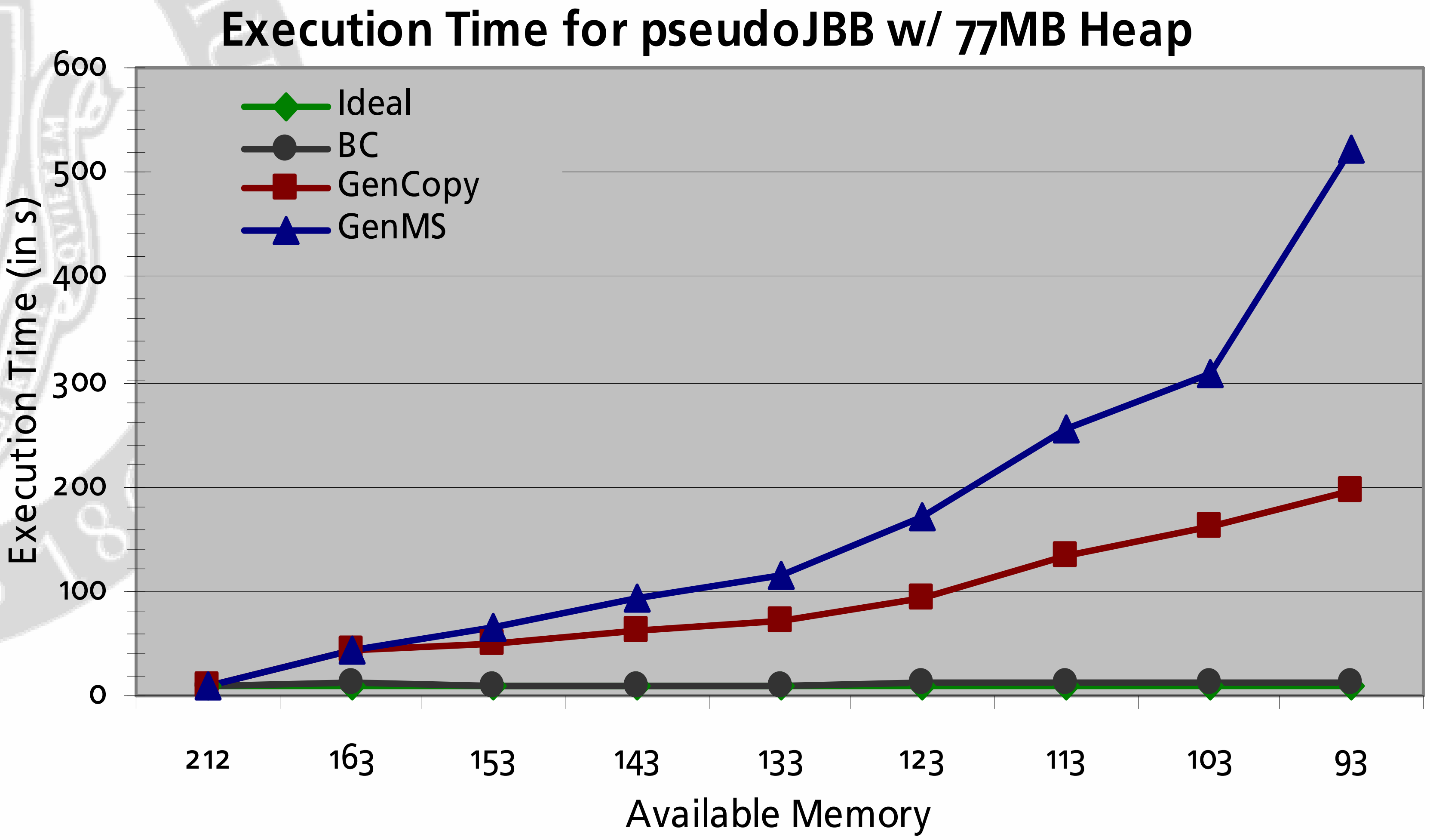
- Extended Linux kernel 2.4.20
  - Eviction notification
  - *vm_relinquish()*
  - Added only 600 LOC
- Jikes RVM 2.3.2 & MMTk
  - Compare BC to MarkSweep, SemiSpace, CopyMS, GenCopy, GenMS

Throughput w/o Memory Pressure



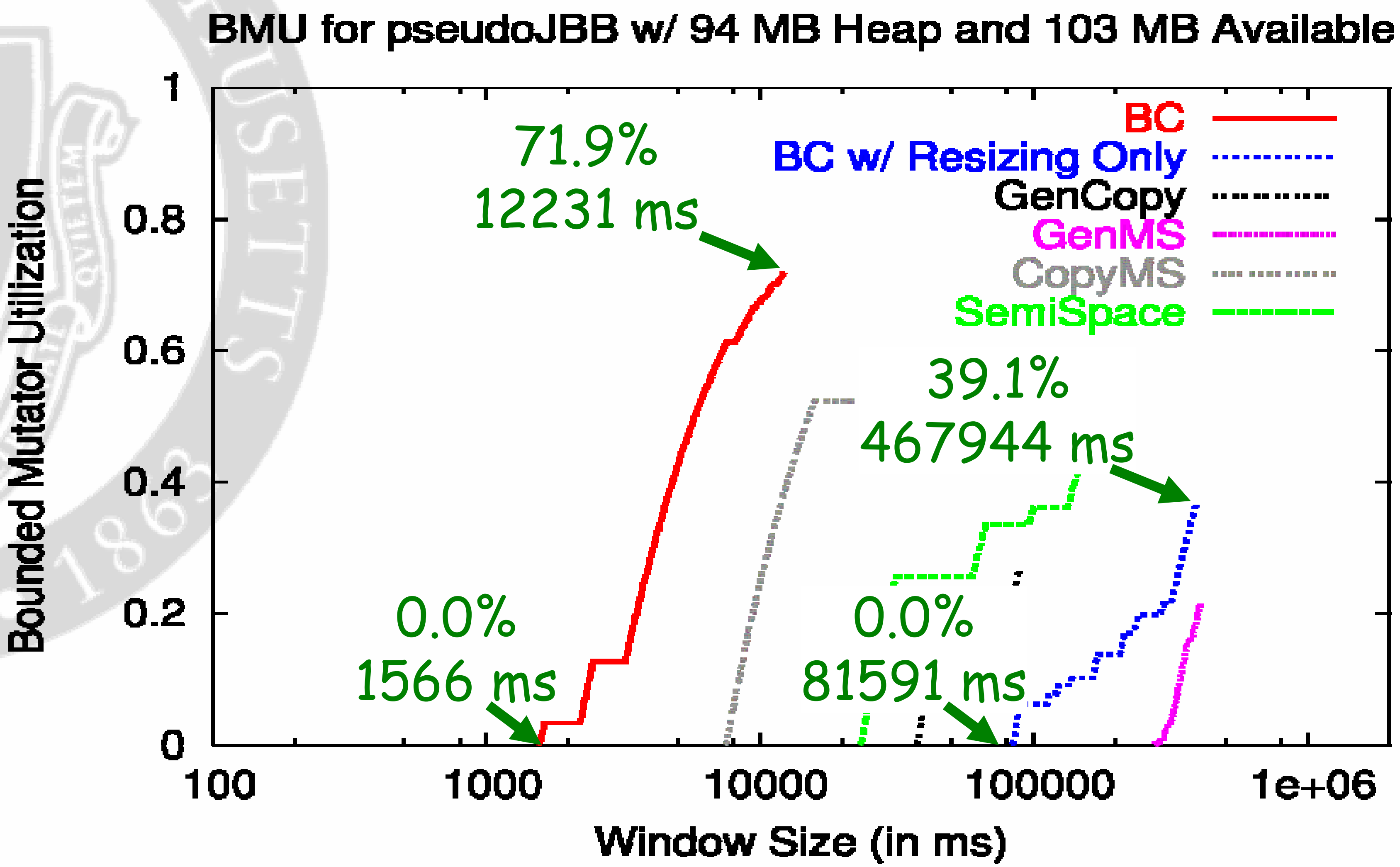
BC runtime comparable to GenMS

Throughput while Paging



BC throughput closely follows ideal curve

BMU Curve w/ Memory Pressure



Bookmarking crucial for good performance

Summary of Results

- **When not paging:**
  - BC as fast as GenMS
- **When paging:**
  - vs. GenMS (fastest when *not* paging)
    - 41x faster, avg pause up to 218x smaller
  - vs. CopyMS (next fastest when paging)
    - 5x faster, avg pause up to 45x smaller

Conclusion

- Bookmarking collector available at: http://www.cs.umass.edu/~hertz

Thank you

Other Pointer Summarizations

- Pointer Compression
  - Save space by compressing pointers
  - Tracing pointers becomes expensive
- Remembered Sets
  - Add evicted pointers to buffers
  - Requires space upon pages eviction
- Per Object Referring Page Counts
  - Increases BC overheads

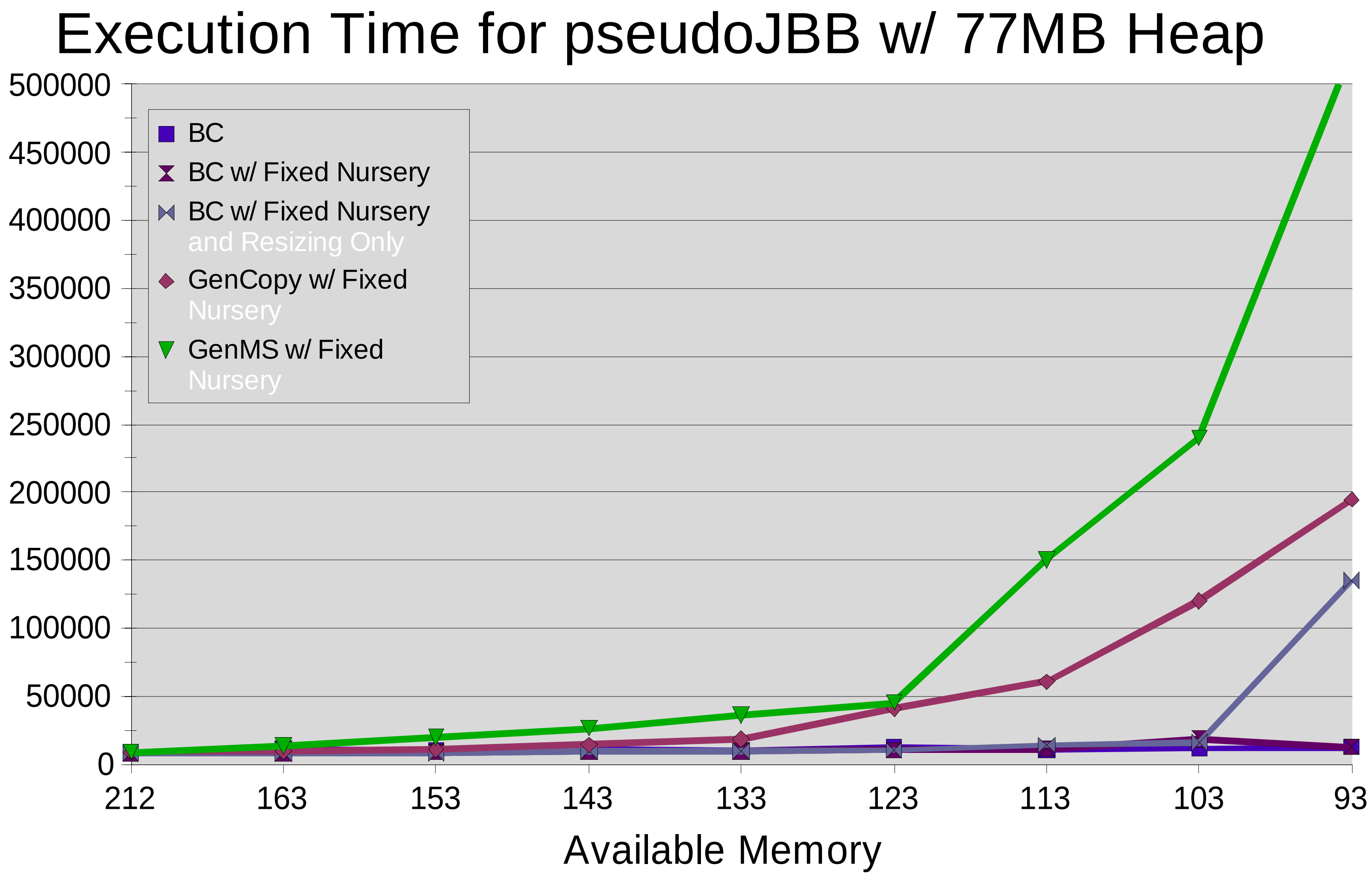
Related Work

- VM-Sensitive Garbage Collection
  - Linearizing LISP lists [Bobrow/Murphy]
    - Does not know or use which heap pages are memory resident
  - Independent heap regions [Bishop]
    - Cannot avoid page faults when region contains evicted pages
  - Ephemeral garbage collection [Moon]
    - Must access evicted pages when they contain pointers into generations being collected

Related Work

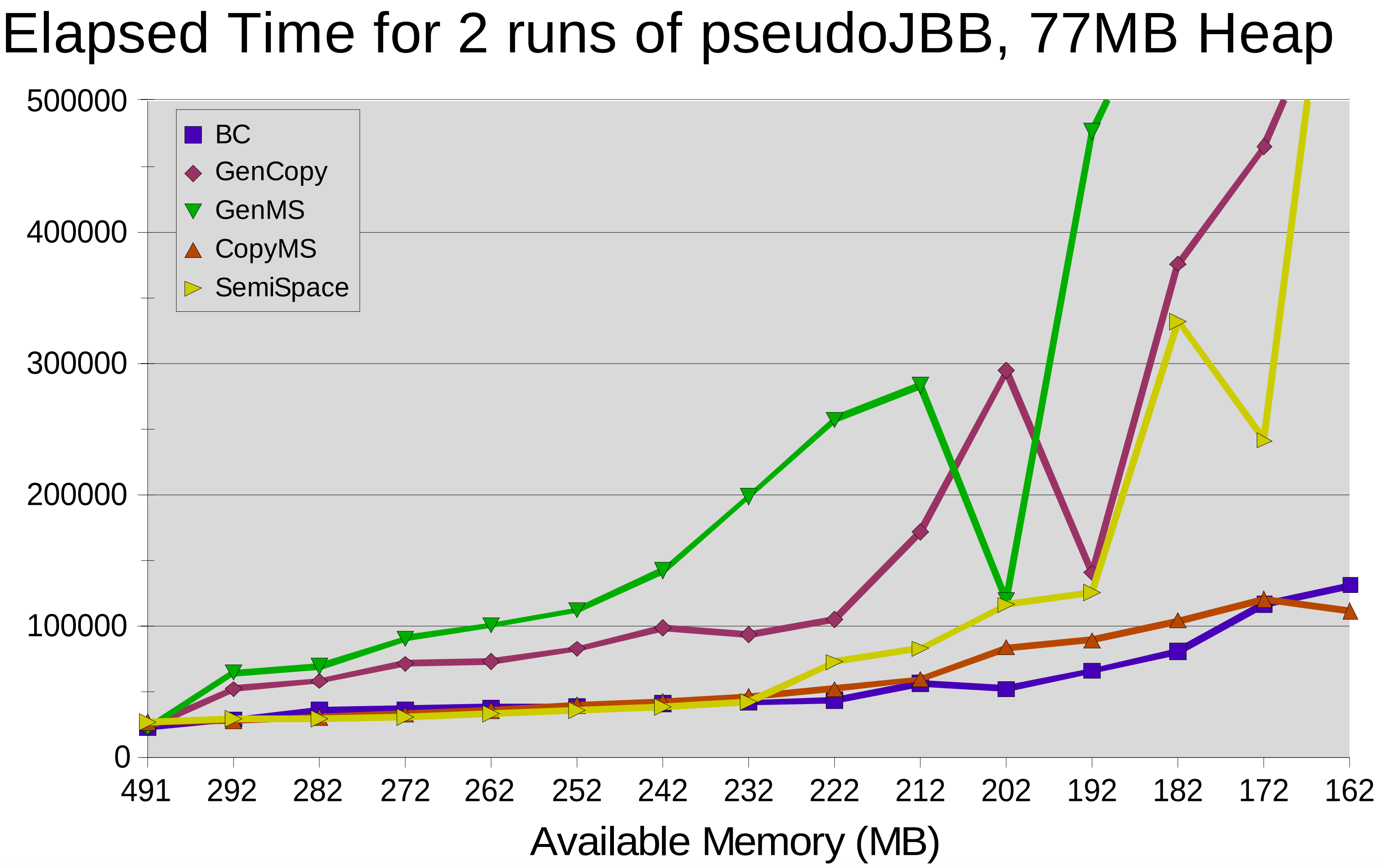
- VM-Cooperative Garbage Collection
  - Discarding empty pages [Cooper, et al.]
    - No mechanism for evicting non-empty pages
  - Shrinking heap size [Alonso/Appel]
    - Responds to memory pressure changes only after a collection
  - Automatic heap sizing [Yang, et al.]
    - Orthogonal approach that determines proper heap size

Throughput w/ Memory Pressure



- BC outperforms fixed nursery collectors

Multiprogramming Performance



- BC performs well in many environments

Experimental Methodology

- Benchmarks: SPECjvm98, ipsixql, jython, and pseudoJBB
- "Opt-and-reset" methodology
  - First iteration optimizes all code
  - Record results from second run