Fast Conservative Garbage Collection

Rifat Shahriyar, Stephen M. Blackburn, Kathryn S. McKinley

Presentation by Alexander Miller

Concurrency and Memory Management Seminar
Univ.-Prof. Dr. Christoph Kirsch

Department of Computer Sciences
University of Salzburg

June 30, 2015

Garbage collection?

Garbage collection (GC) is the automatic recycling of dynamically allocated memory.

Preventing common memory related bugs

- Memory leaks
- Double frees
- Dangling pointers

Two approaches

1. *Tracing*: Operating on live objects
2. *Reference counting (RC)*: Operating on dead objects

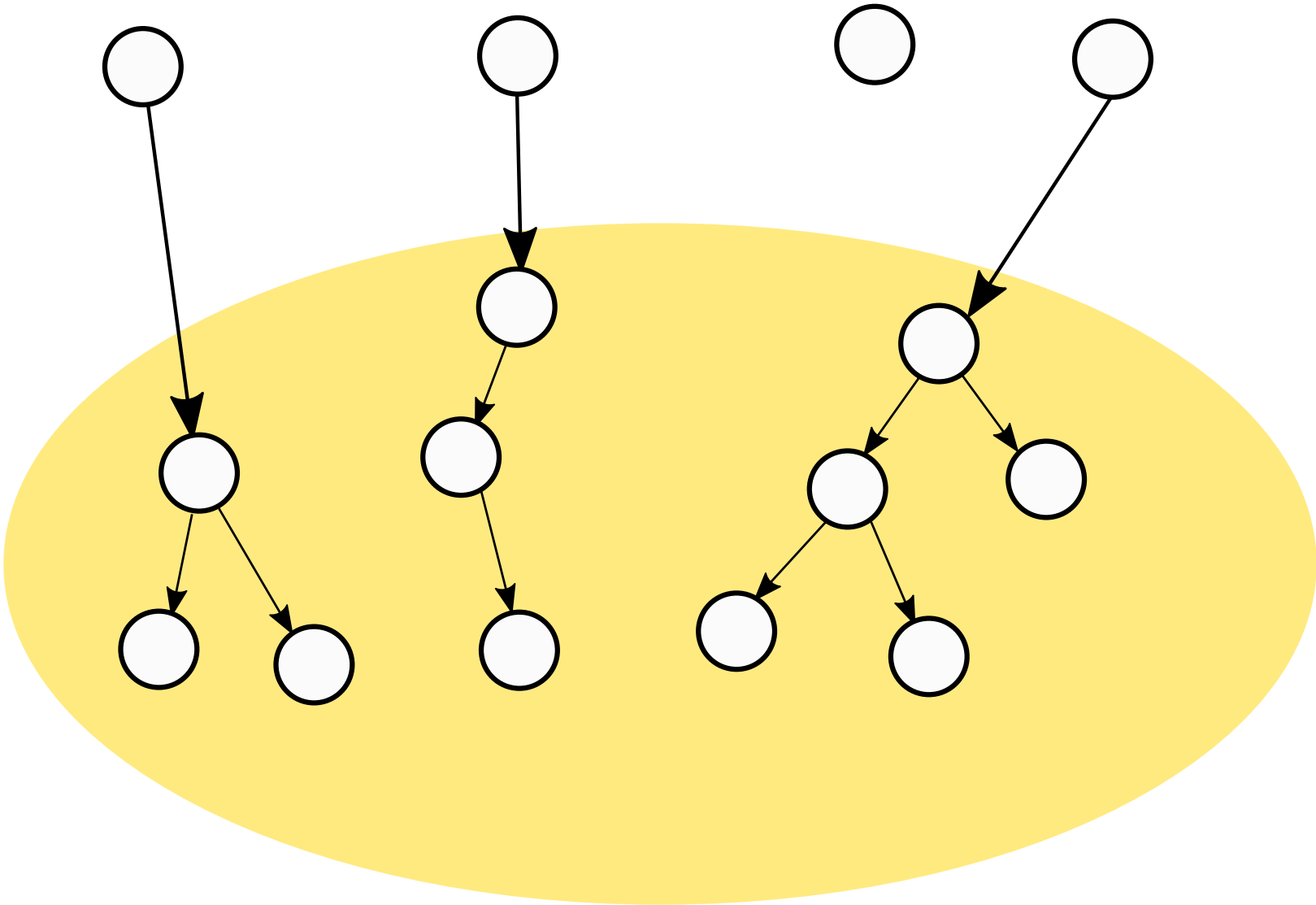
Exact vs. conservative collector

An exact collector identifies all references precisely.

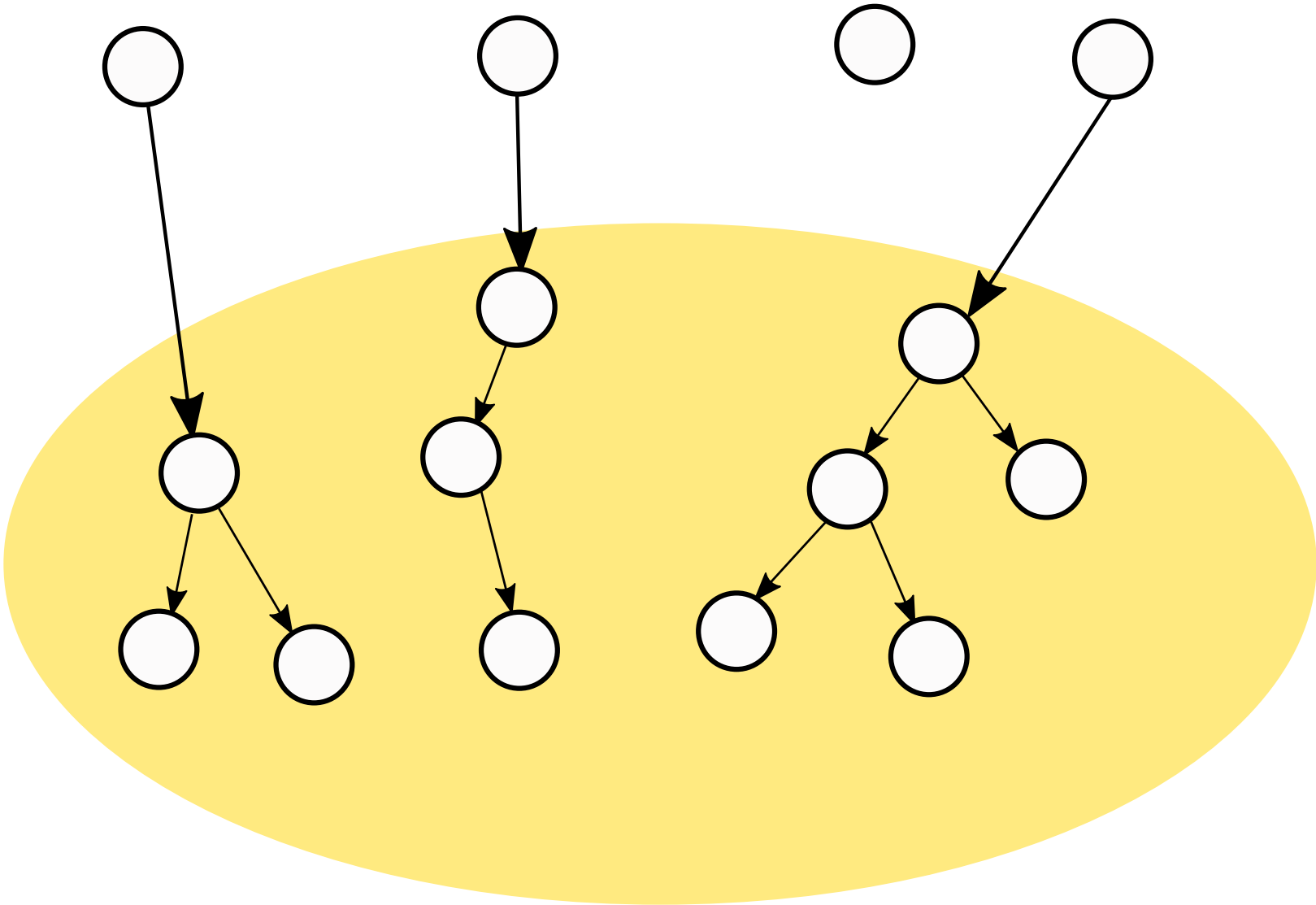
A conservative collector has to reason about *ambiguous references*.

1. As ambiguous references may be pointers, the collector must conservatively retain referents.
2. As ambiguous references may be values, the collector must not change them.

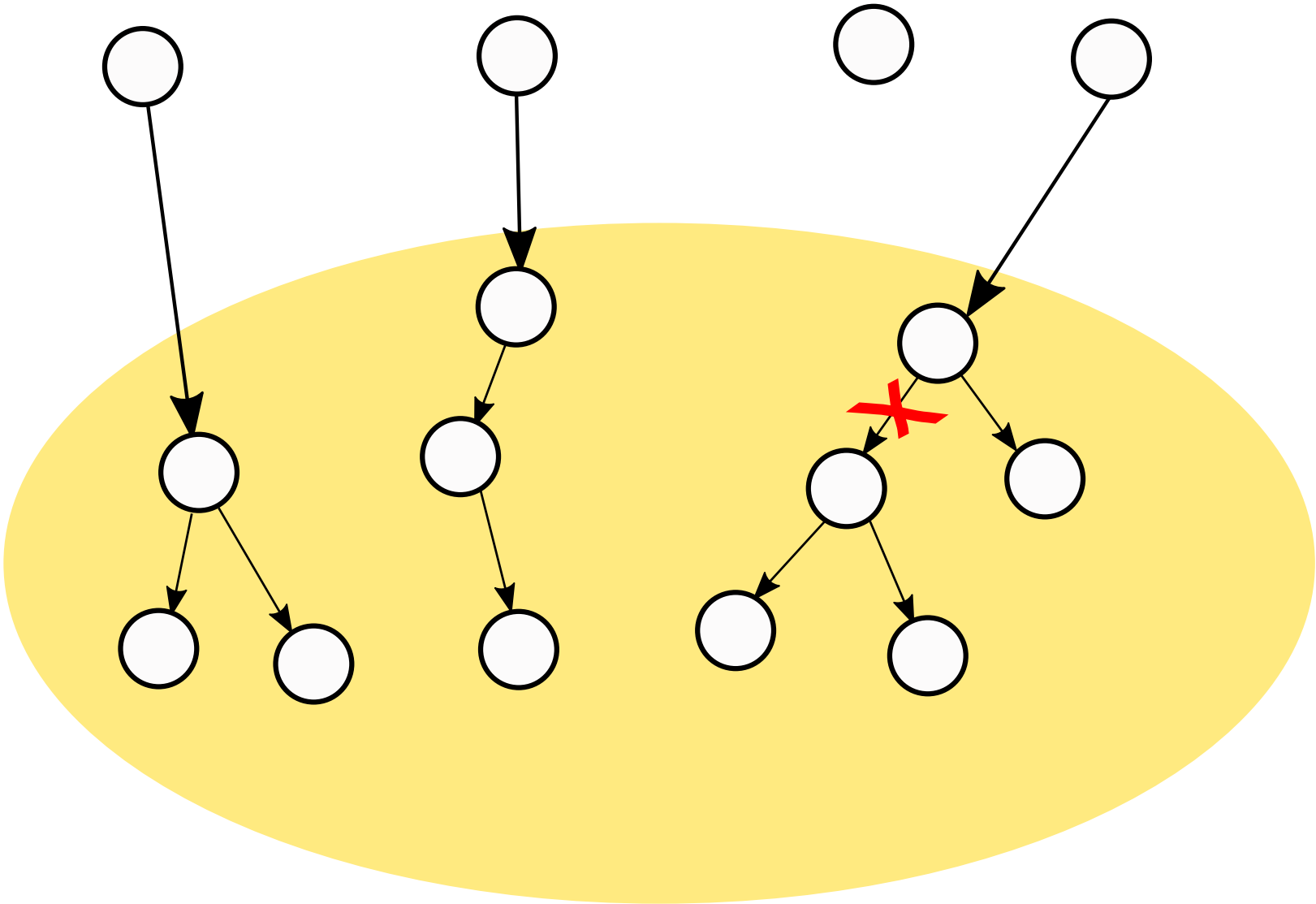
Example



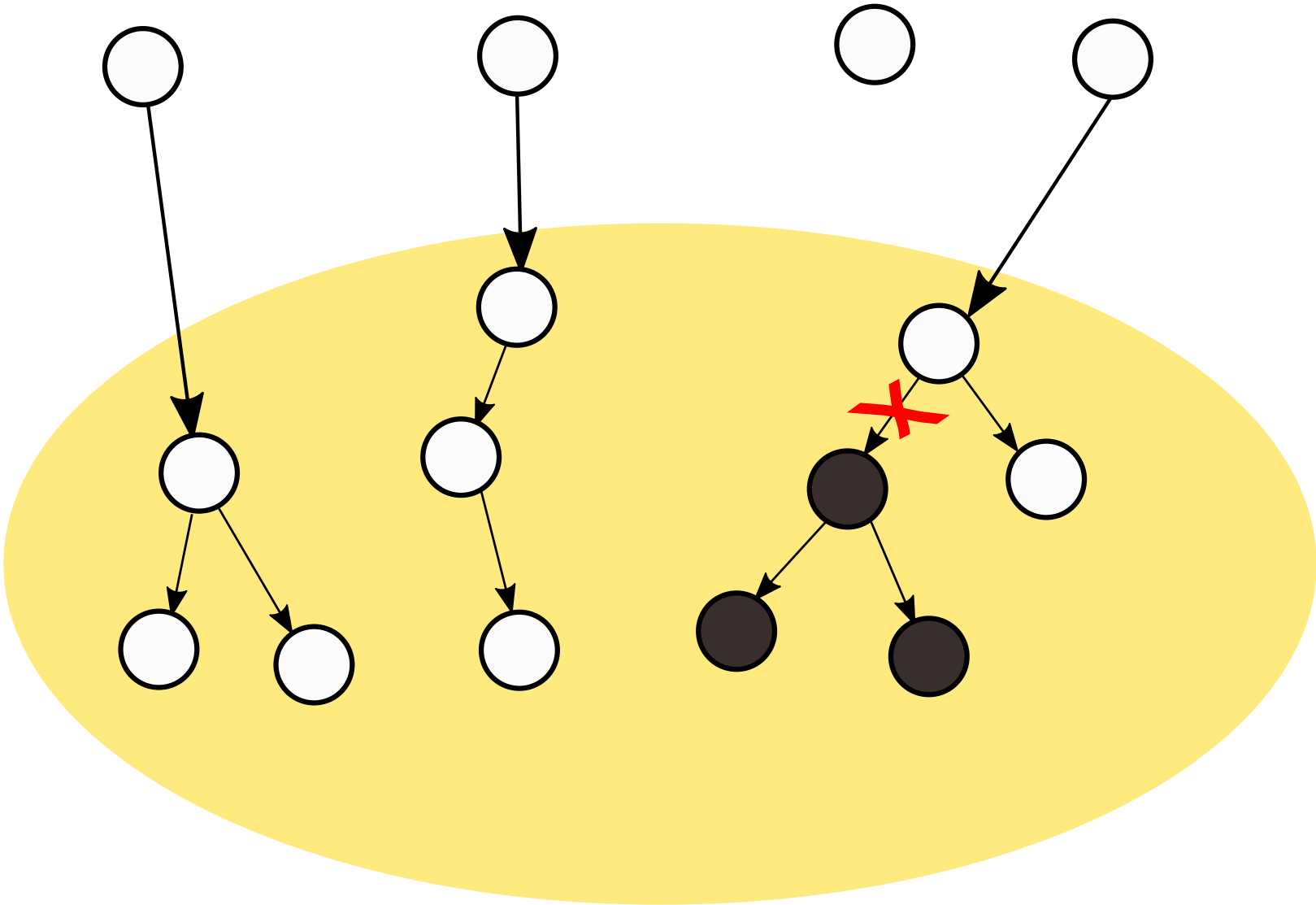
Exact collection



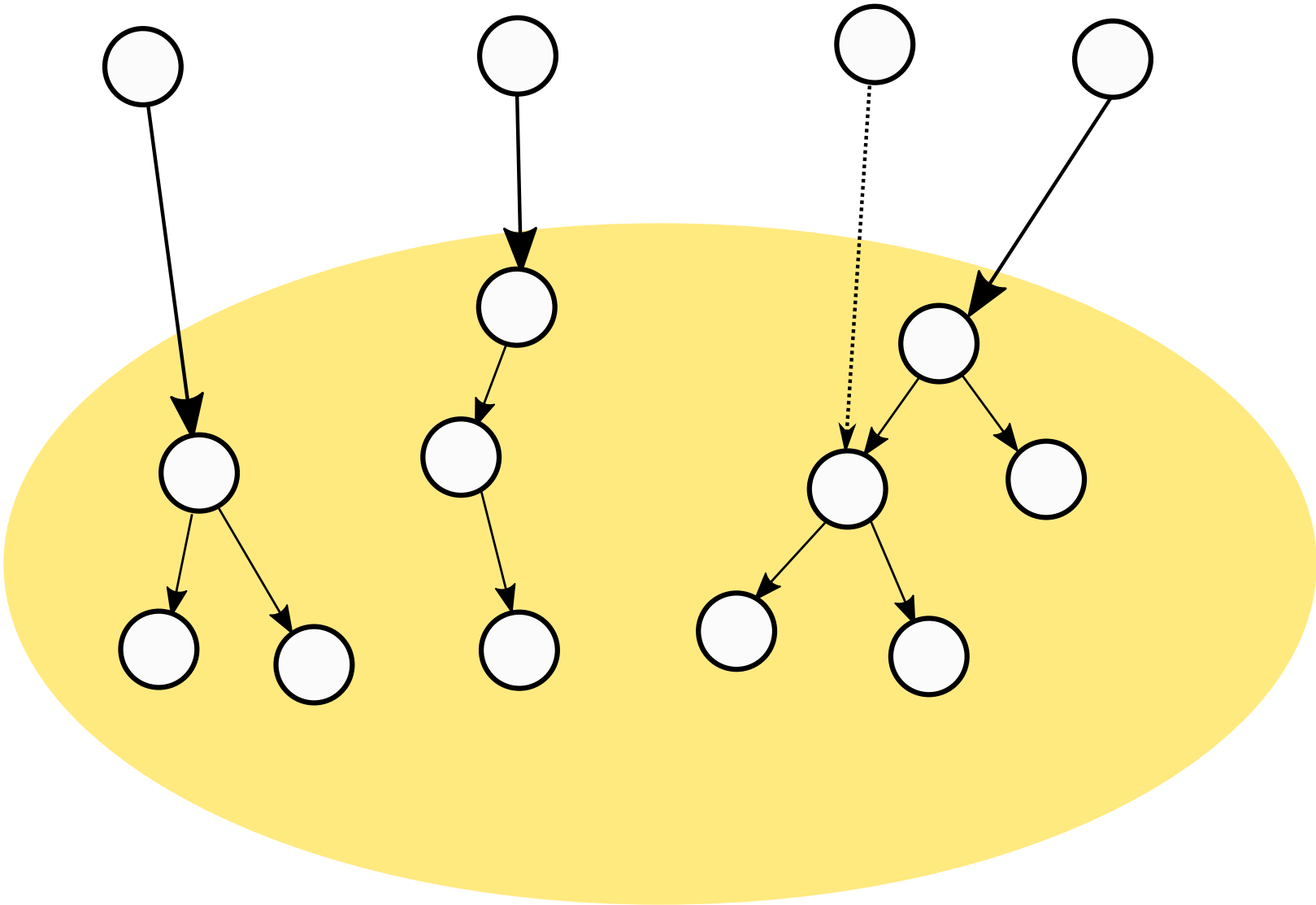
Exact collection



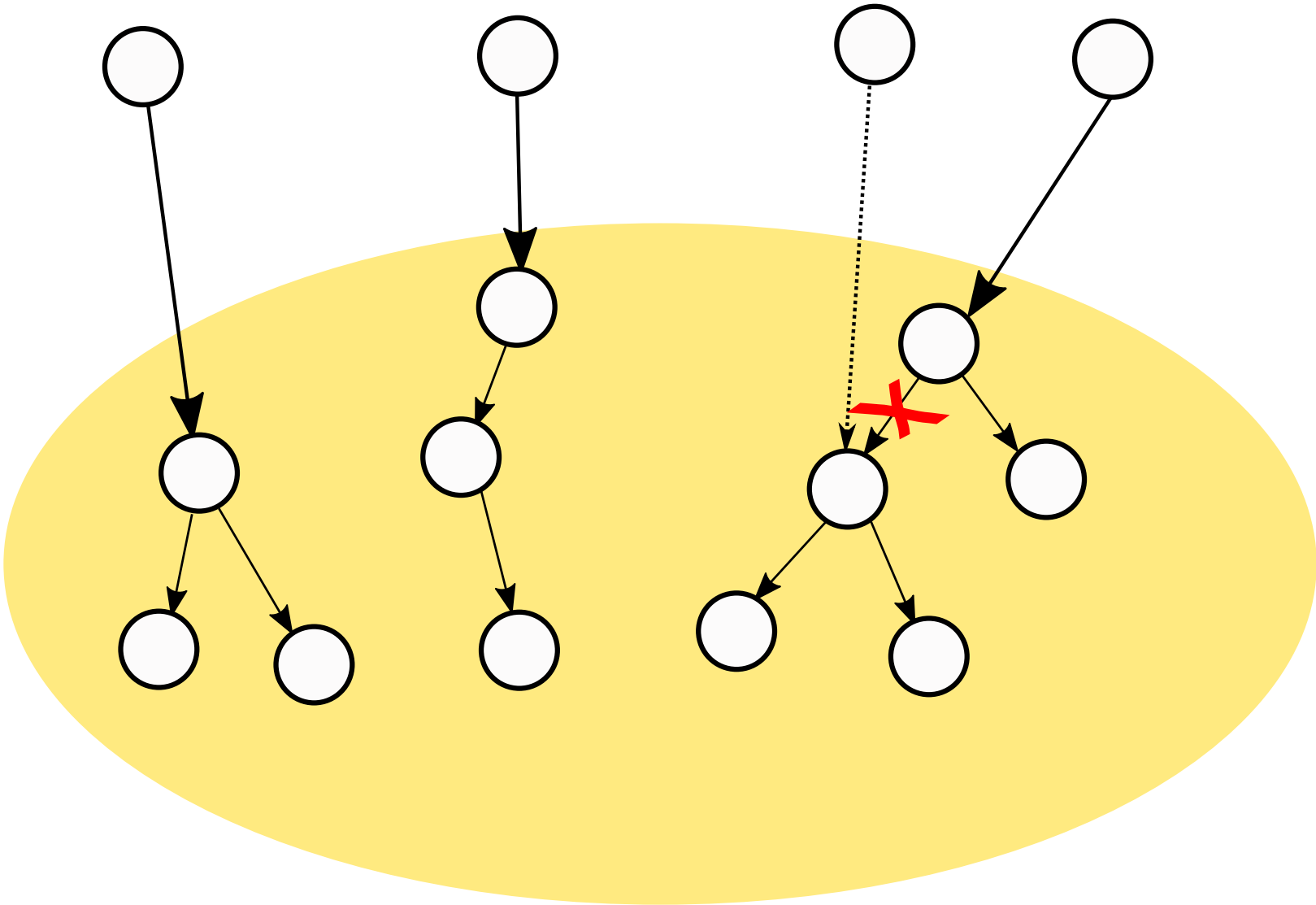
Exact collection



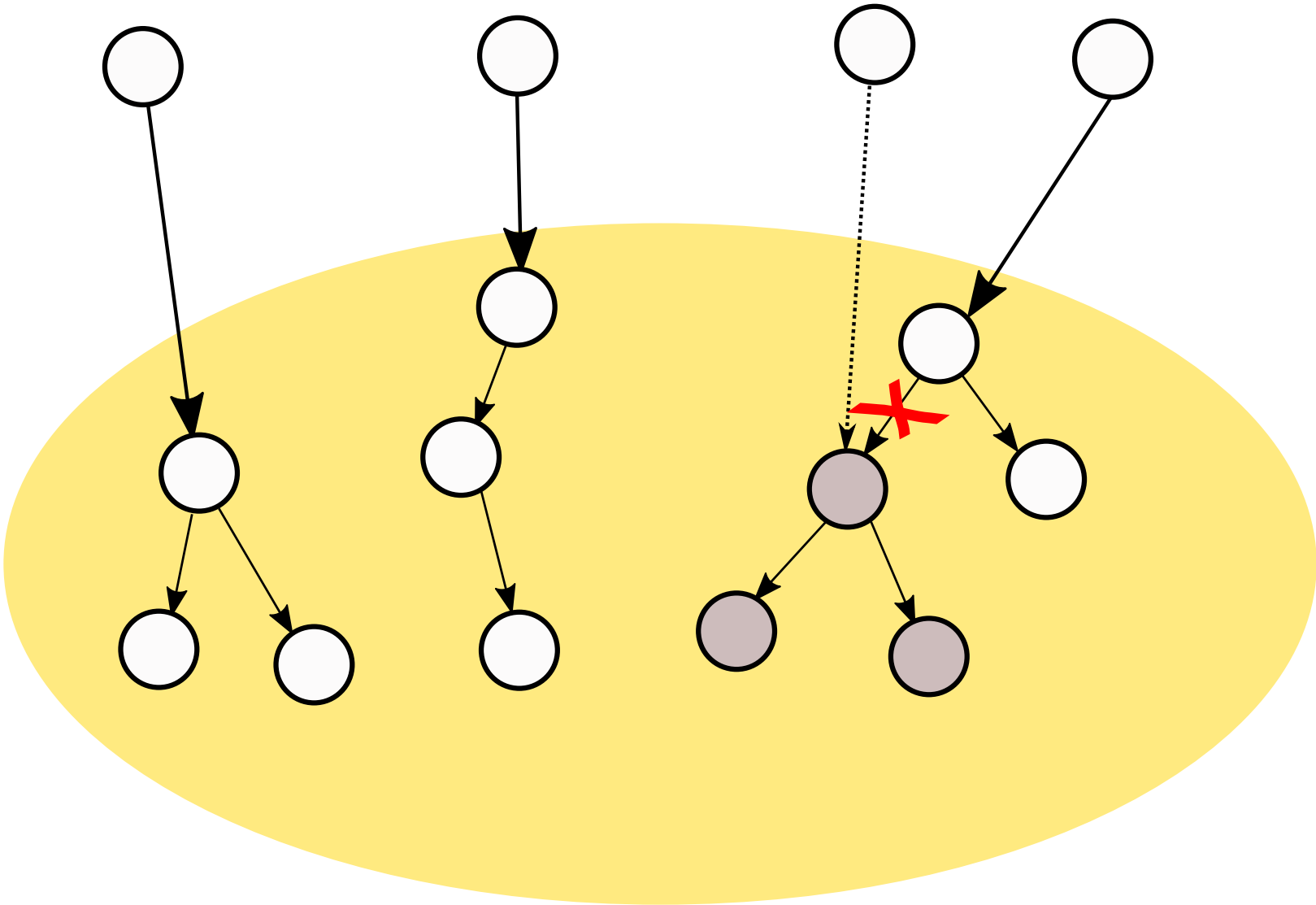
Conservative collection



Conservative collection



Conservative collection



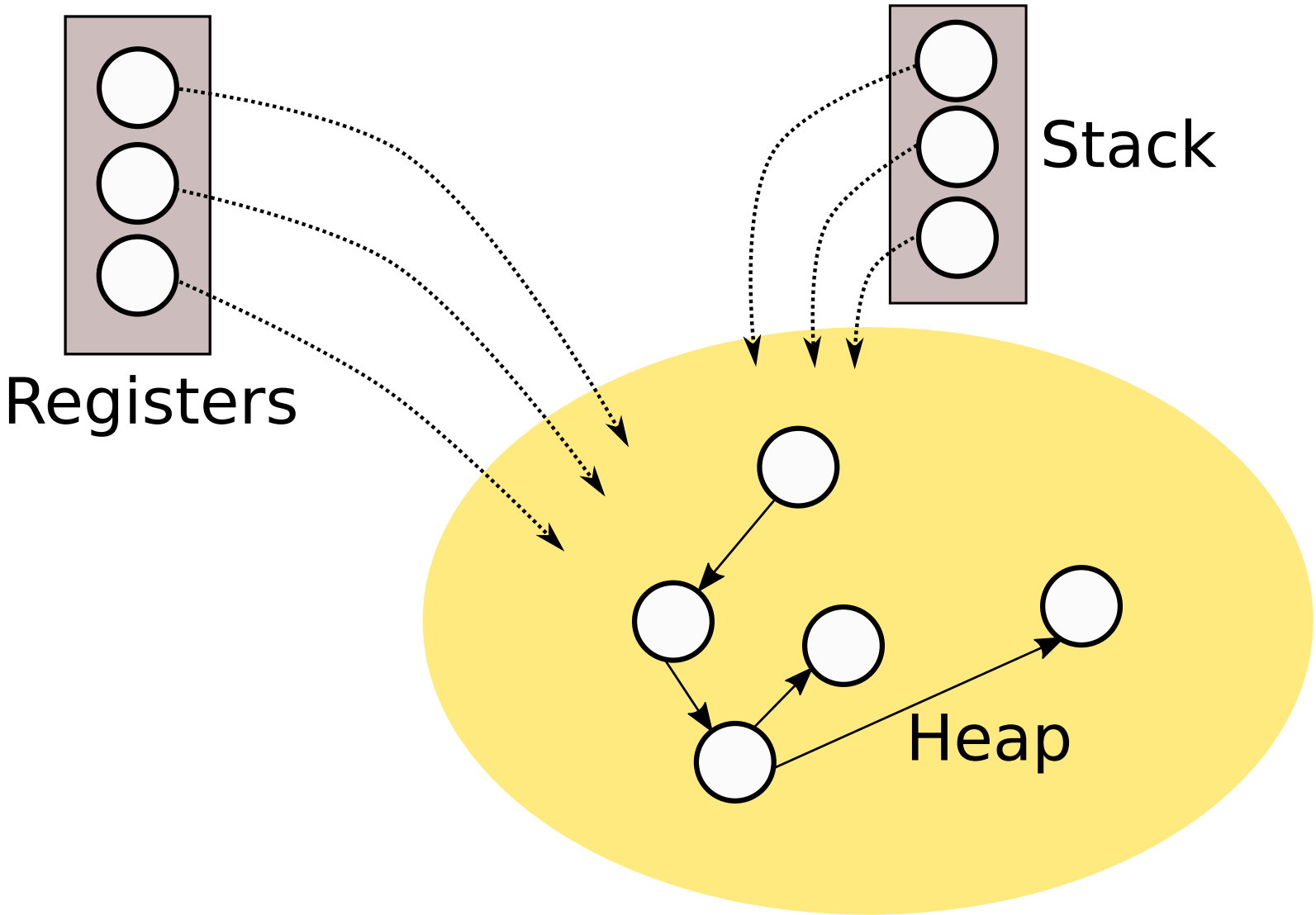
Why conservative GC?

Conservative collectors trade implementation complexity for performance.
Also, it makes garbage collection independent of the compiler.

Applications of conservative GC

- Unsafe languages (C, C++)
- Managed languages
  - Microsoft's Chakra JavaScript VM
  - Apple's WebKit JavaScript VM
  - Objective-C, Swift, PHP, ...

Assumptions about references



Methodology

- 18 benchmarks from DaCapo, SPECjvm98, and pjbb2005
- Jikes RVM 3.1.3+hg r10761 and MMTk
- Ubuntu 12.04.3 LTS Server with a x86_64 Linux 3.8.0-29 kernel
- 3.4 GHz Core i7 with 4 cores

Ambiguous pointers

	Average
Unique exact roots	98
All exact roots	2.21x
All unfiltered conservative roots	8.9x
All conservative roots	4.7x
Unique conservative roots	1.6x

Excess retention

	Average
Excess retention	44 KB
Excess retention / live	0.02%

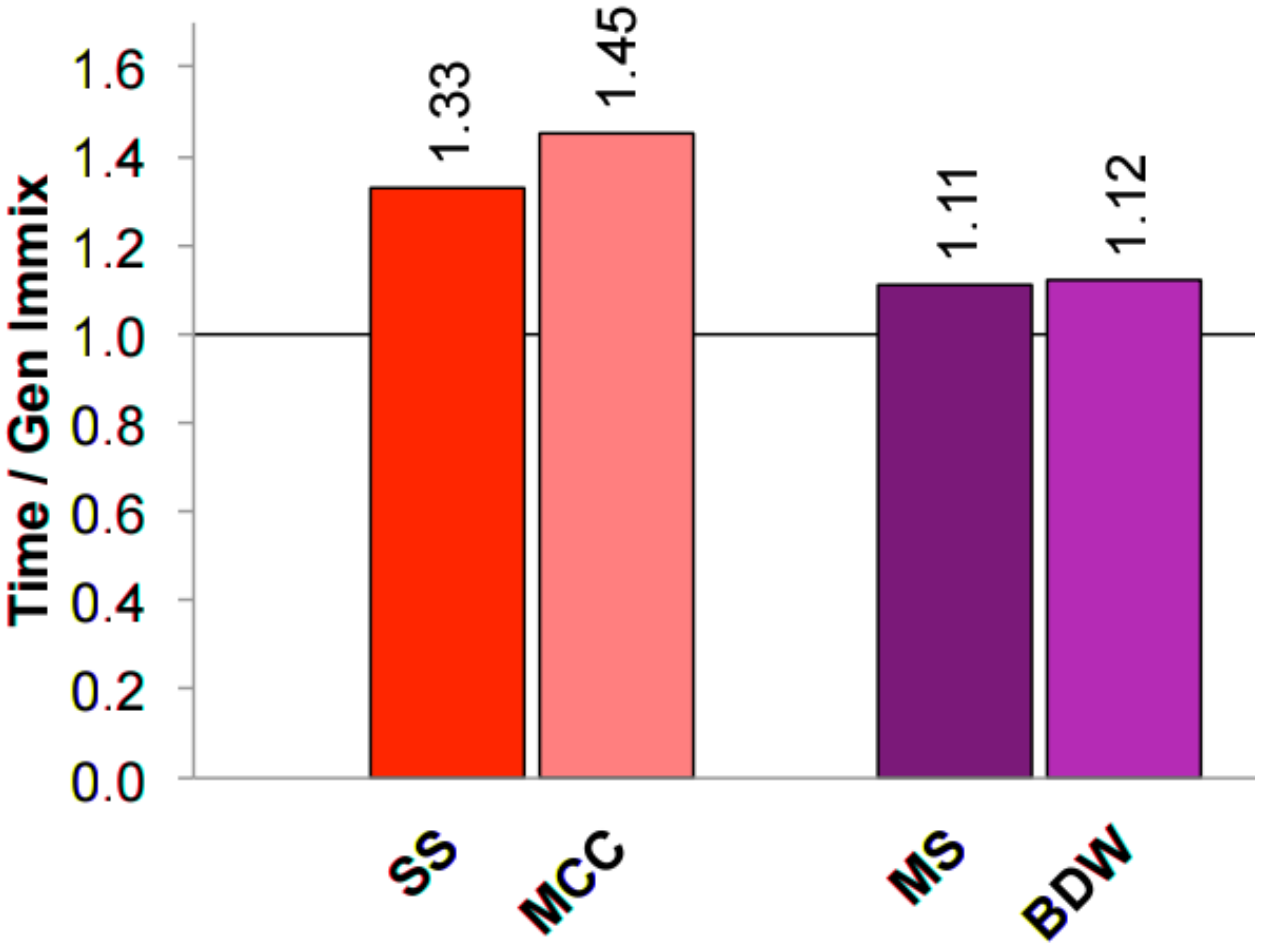
Comparison of existing GCs

Comparing exact collectors and their conservative counterparts

- Exact semi-space (SS)
- Conservative mostly-copying collector (MCC)
- Exact mark-sweep (MS)
- Conservative Boehm, Demers, Weiser style collector (BDW)

compared to the Jikes production collector Gen Immix

Comparison of existing GCs - Results



Towards a fast conservative GC

Extending exact RC Immix[1] for conservative collection.

[1] R. Shahriyar et al. Taking Off the Gloves with Reference Counting Immix. In OOPSLA 2013.

RC Immix

Combines two concepts

- Deferred reference counting
- Immix heap structure

Deferred RC

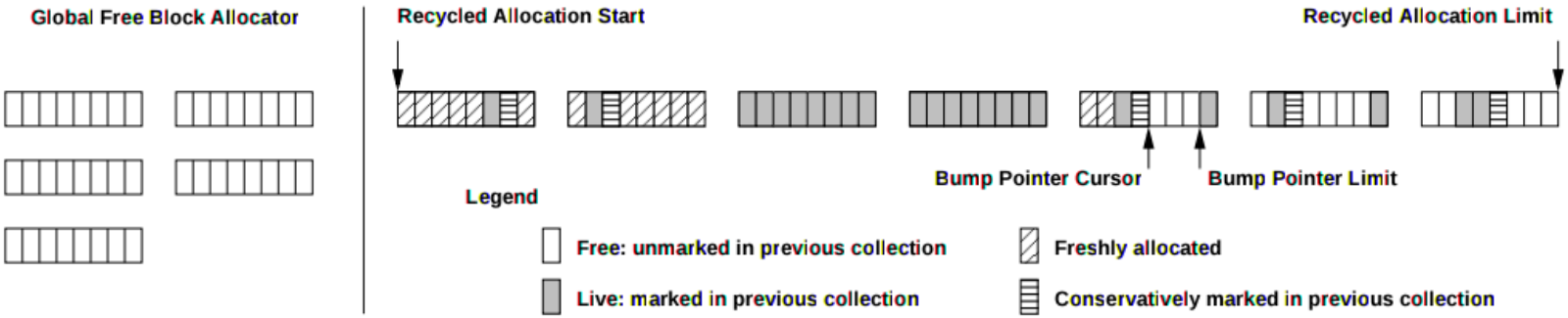
- References are count periodically
- Eliminates increments and decrements of new objects
- Object recycling is no longer immediate
- Roots have to be enumerated

Immix

- Immix[2] is a copying mark-region collector
- Memory is divided into *blocks* of 32 KB
- Blocks are divided into *lines* of 256 B
- Objects may span lines
- *Opportunistic copying*

[2] S. M. Blackburn et al. Immix: A Mark-Region Garbage Collector with Space Efficiency, Fast Collection, and Mutator Performance. In PLDI 2008.

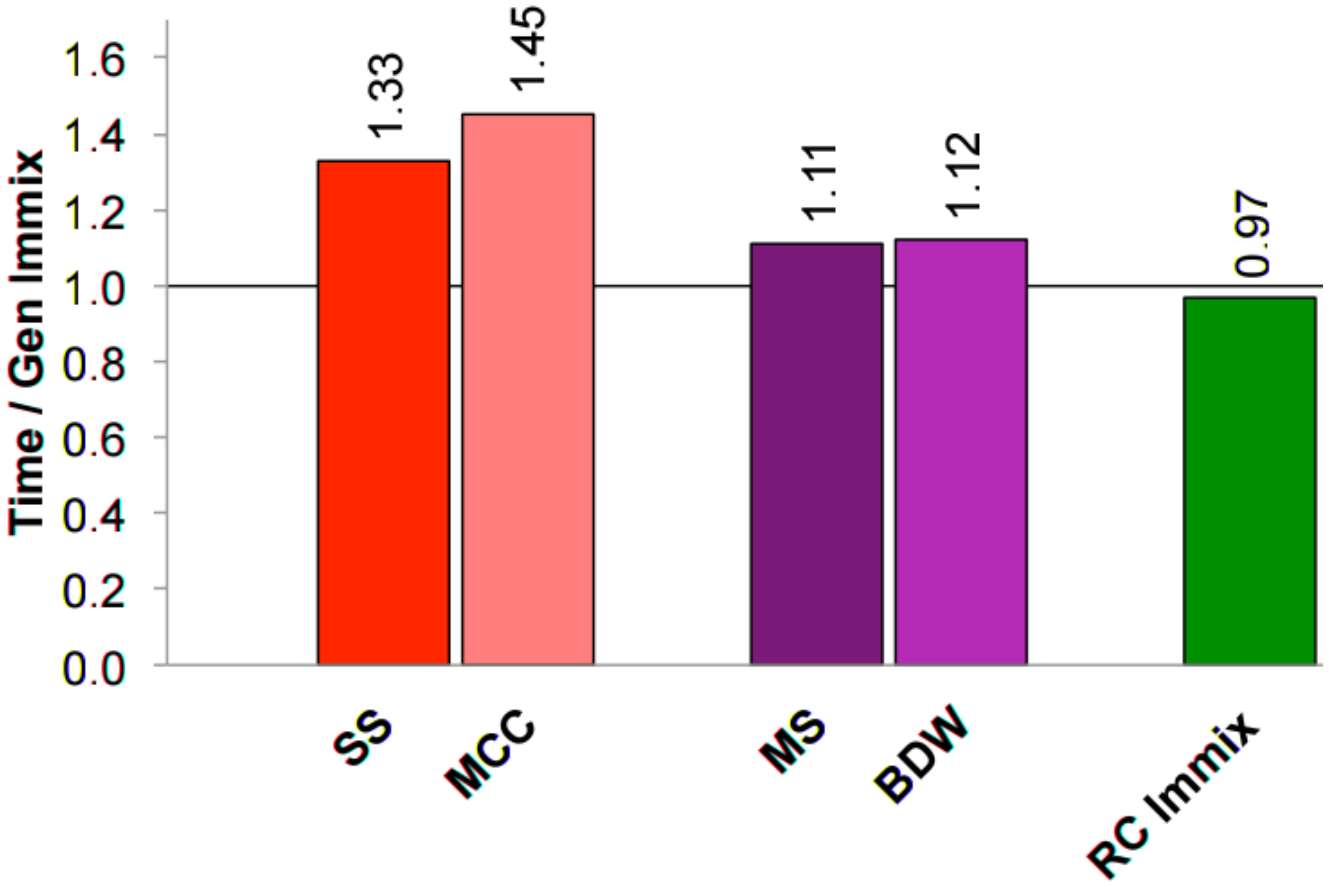
Immix heap


Global Free Block Allocator
Recycled Allocation Start
Recycled Allocation Limit
Bump Pointer Cursor
Bump Pointer Limit
Legend
Free: unmarked in previous collection
Freshly allocated
Live: marked in previous collection
Conservatively marked in previous collection

RC + Immix = RC Immix

- Reference counting counts live objects on a line
- Tracing GC as backup cycle detection
- Copies young objects during reference counting
- Copies old objects during tracing

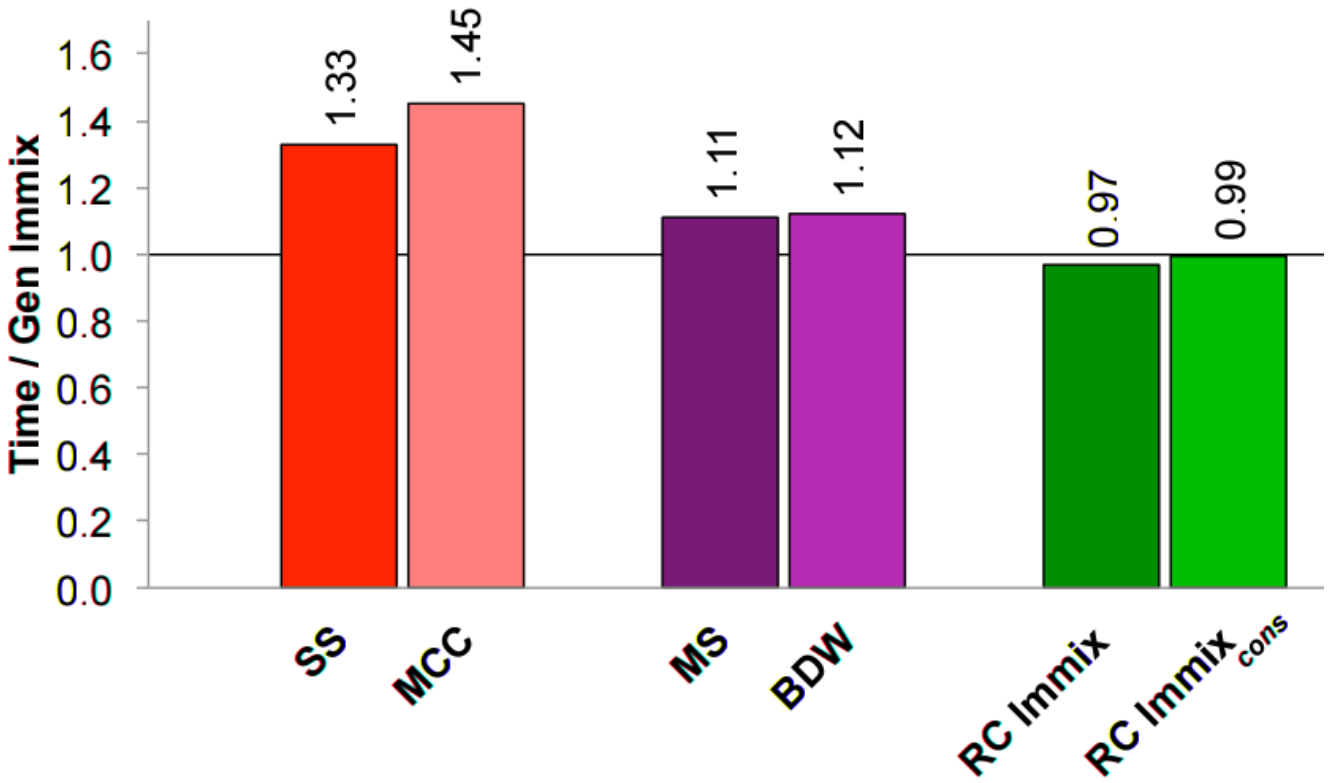
Comparison with Gen Immix



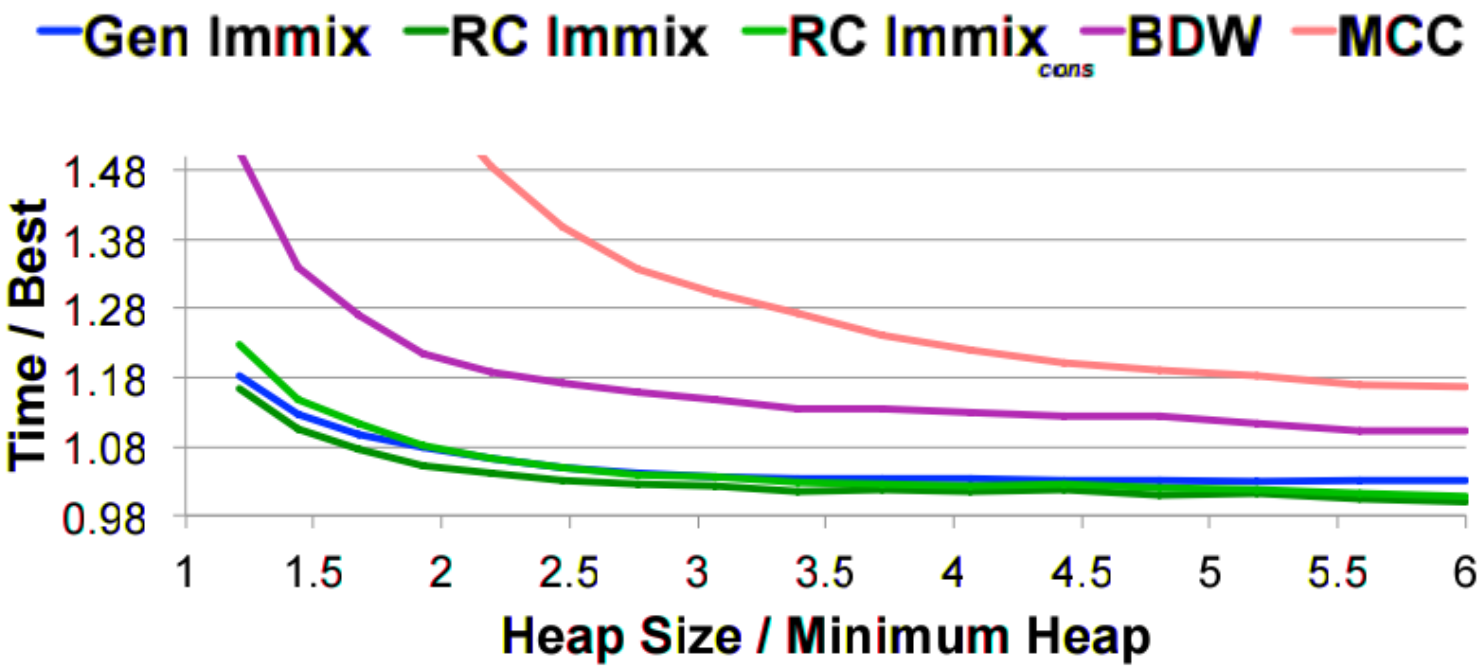
RC Immix$_{cons}$

- Object map to filter valid pointers
- Referents of valid pointer are marked as live and pinned

RC Immix$_{cons}$ performance

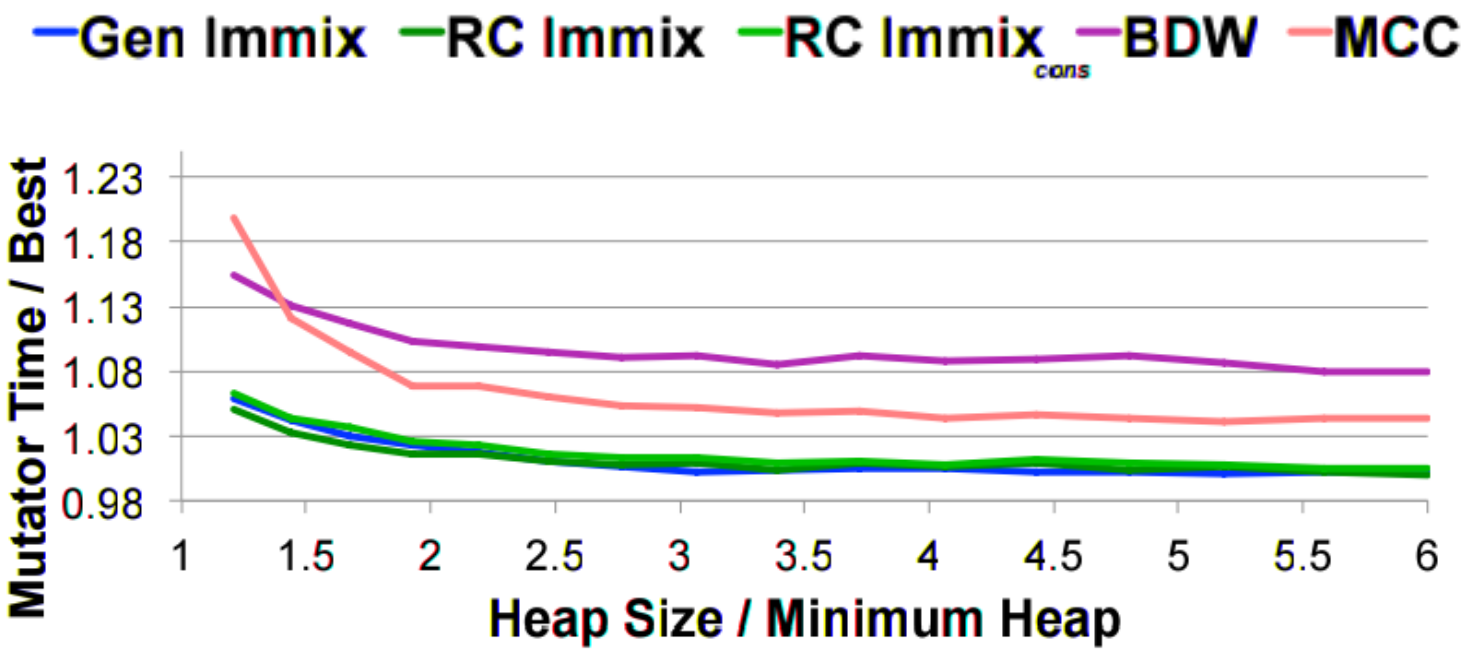


Performance over heap size



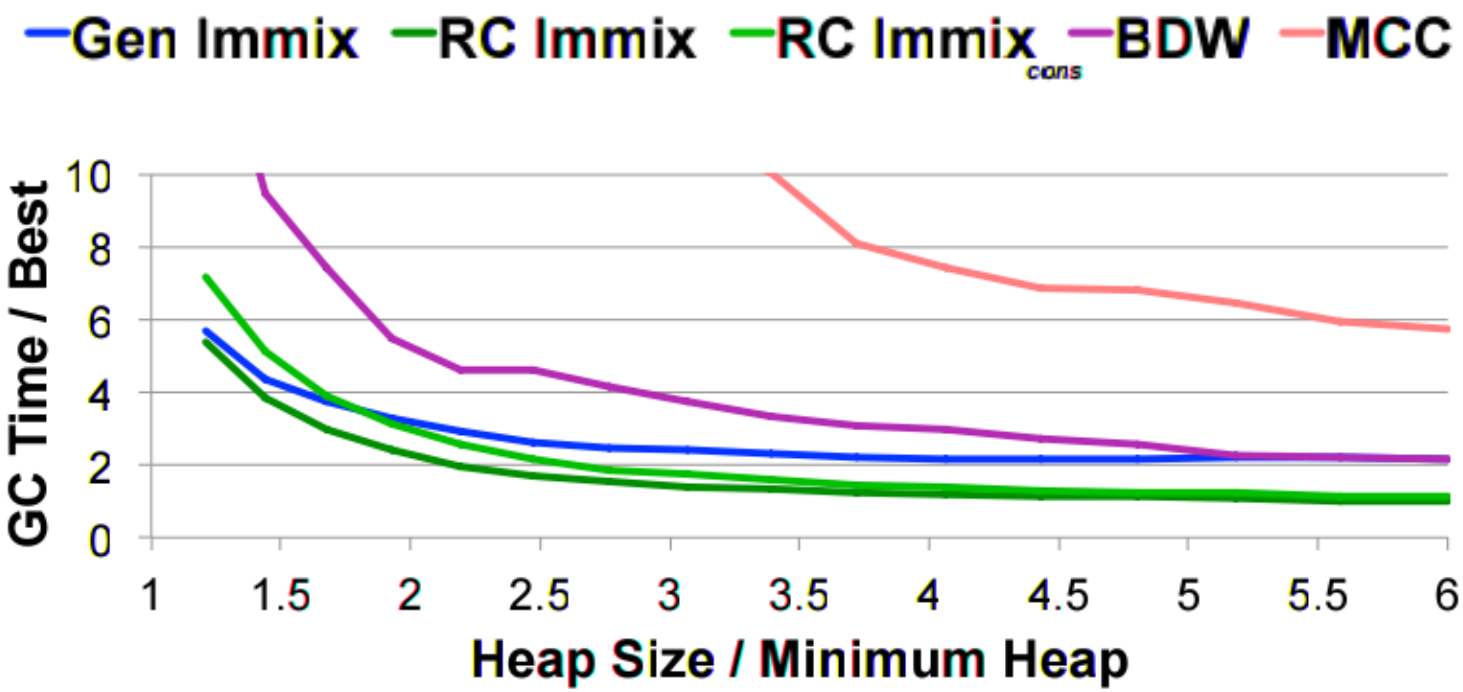
(a) Total time

Performance over heap size



(b) Mutator time

Performance over heap size



(c) GC time

Conclusion

- Comparison of exact and conservative collectors
- Implementation of a high performance conservative collector
- Performance evaluation