Correctness-Preserving Derivation of Concurrent Garbage Collection Algorithms

Vechev, Yahav, and Bacon

Presented by

Suman Jana

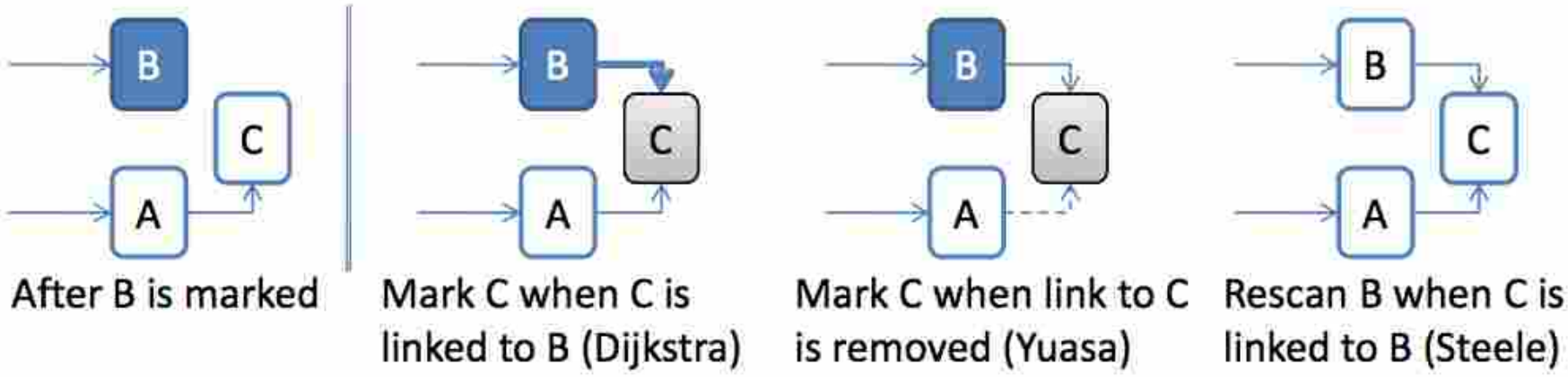
(Some slides courtesy Sam Harwell)

Big Picture

- Multiple known Concurrent GC algorithms.
  - Different composition of same 'bag of tricks'
  - Trade-offs not well understood
- Contributions
  - Parametric concurrent GC
  - Apex algorithm (simple/less concurrent)
  - Correctness-preserving transformations
- Limitations
  - Does not handle sweep phase
  - No correctness proof for Apex

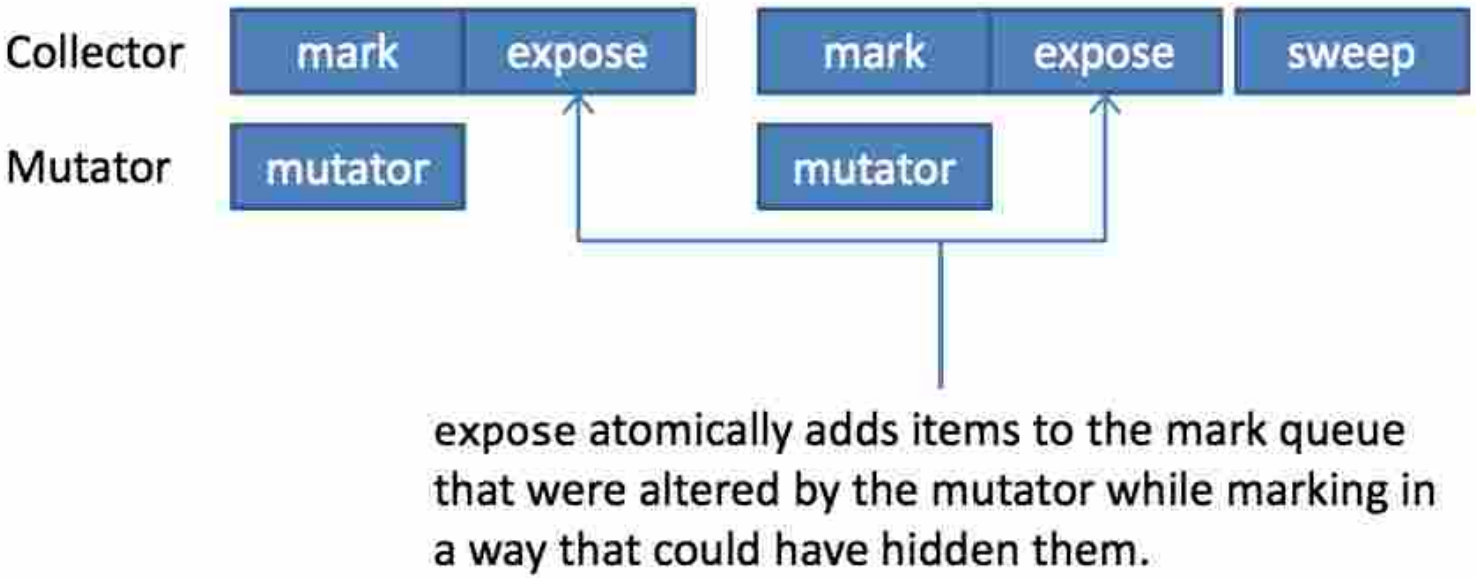
Hidden Objects

- Concurrent mutator can 'hide' accessible objects from GC



Parametric Concurrent GC

- Atomic 'expose' function is the single parameter
  - Different 'expose' functions result in different GCs



Background

- **Precision**: difference between estimated set of live objects and actual set of live objects
  - Low precision ⟶
    - Less collected garbage
    - Faster termination
    - Higher concurrency (less time spent in 'expose')
- Log: all tracing, mutation and allocation operations are stored in the log
- Input to 'expose' – log prefix P

Apex Algorithm

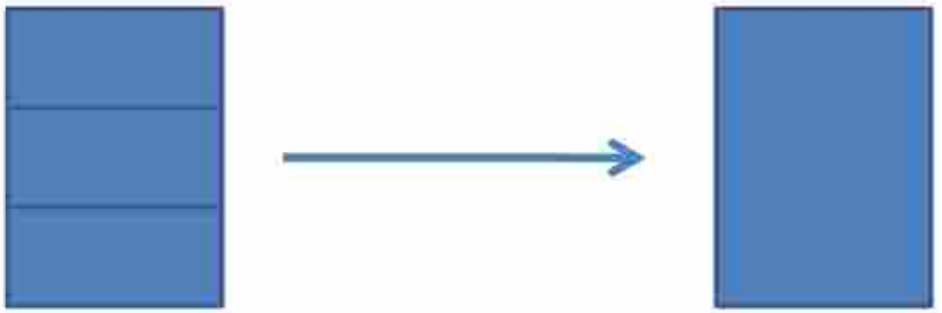
- An instance of the parallel concurrent GC
- Goals
  - As easy as possible to show correctness
  - Well-defined as a base for transformations
- Characteristics
  - Sacrifices concurrency for precision and clarity
  - Implements expose as rescanning, similar to Steele

Dimensions: variables in 'expose'

- **Wavefront**: how far has the collector progressed ?
- **Policy**: how are modified objects behind the wavefront treated ?
- **Threshold**: Maximum allowed cross-wavefront count?
- **Protection**: which objects must be traced to ensure all live objects are found?
- **Allocation**: how are newly allocated objects handled?

Wavefront: Dimension 1

- Granularity for tracking wavefront ?
  - Object Level (OL)
  - Field Level (FL)



Per-Field Wavefront
-Exact information
-More expensive
-More garbage collected

Per-Object Wavefront
-Approximate information
-Less expensive
-Less garbage collected

Policy: Dimension2

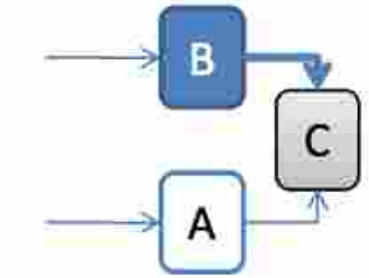
- Modifications to a field can be found by
  - Atomically scanning the heap (SR)
    - Less concurrency
  - Scanning the log (LR)
    - Write barrier overheads
- SR-{x} and LR U {x} is correctness preserving and precision reducing
  - Limited wavefront tracking precision
  - Limited threshold count

Threshold : Dimension3

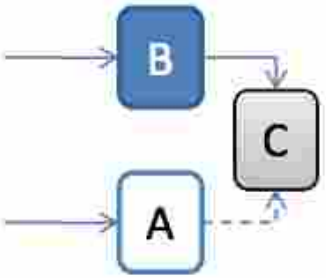
- Counts cross-wavefront(behind->ahead) pointers
  - Only for LR objects
- Add only objects with positive counts to pending mark queue
  - Coalesces multiple changes in the same field
- Threshold: Limiting counts
  - Needed for space restriction
  - No decrement when (obj.count==max)
- Decreasing threshold results in less precision but preserves correctness

Protection : Dimension4

- How to handle hidden objects ?
  - Track cross-wavefront (behind->ahead) pointer installations (IS) (incremental GC)
  - Track pointer deletions ahead of wavefront (DS) (Snapshot GC)



Installation-based protection



Deletion-based protection

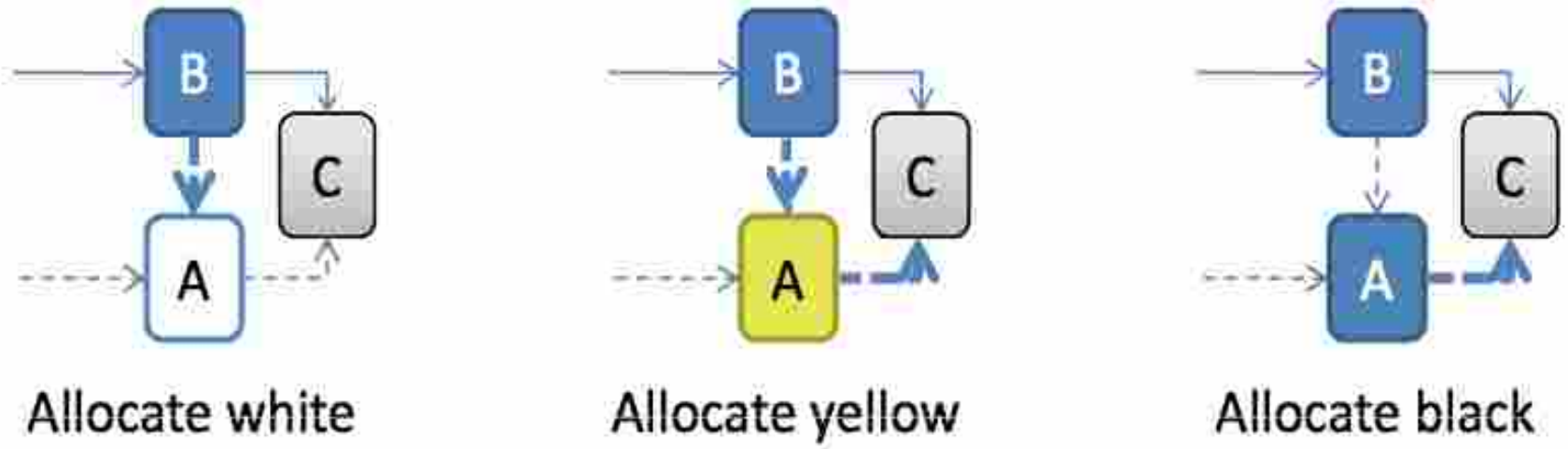
Transforming along Protection dimension

- Moving an object From IS to DS
  - Constraint: object must be transitively protected by a path of D-protected objects
  - Weak precision reducing and correctness preserving
    - If *GC* X is weakly less precise than *GC* Y: transitive closure of exposed objects by Y is a subset of transitive closure of exposed objects by X (for any log l)

Allocation : Dimension5

- How to handle objects allocated during 'mark'
  - Put ahead of the wavefront (Apex)
  - Put in special unmarked state (Yellow), treated as behind the wavefront for any IS pointers stored in the object
- Allocating an object yellow instead of white reduces precision but preserves correctness
- Allocating an object black further reduces precision
- Less precision $\rightarrow$ faster termination

Allocation : Dimension5 (cntd.)



DISCUSSION QUESTIONS

- Can we make the 'sweep' phase parametric as well ?
- Instead of atomic 'expose', can we make it concurrent to some extent ?
  - More complexity