Incremental Tracing Collectors

- “Stop-and-copy” collectors are prone to unpredictable delays.
  - These delays arise because the mutator is halted for the full duration of a collection cycle, so that it experiences the collection as a single atomic action.
  - This may not be acceptable in many real-time scenarios.

Incremental Tracing Collectors

- "Stop-and-copy" collectors are prone to unpredictable delays.
    - These delays arise because the mutator is halted for the full duration of a collection cycle, so that it experiences the collection as a single atomic action.
    - This may not be acceptable in many real-time scenarios.
- This leads us to incremental collector designs.
    - Small units of collector activity must be interleaved with small units of mutator activity.
    - Reference counting collectors are easy to make incremental.
    - *What about tracing collectors (either mark-sweep or copying)?*

Incremental Tracing Collectors

- "Stop-and-copy" collectors are prone to unpredictable delays.
  - These delays arise because the mutator is halted for the full duration of a collection cycle, so that it experiences the collection as a single atomic action.
  - This may not be acceptable in many real-time scenarios.
- This leads us to incremental collector designs.
  - Small units of collector activity must be interleaved with small units of mutator activity.
  - Reference counting collectors are easy to make incremental.
  - *What about tracing collectors (either mark-sweep or copying)?*
- Fundamental problem
  - Since tracing happens as a sequence of units interleaved with units of mutator activity, the graph of reachable objects is not fixed.
  - The collector needs to have some way of tracking these changes and preventing (or recovering from) any adverse consequences.

Towards The Tricolor Marking Abstraction

- Consider a *non-incremental* tracing collector.
  - Conceptually, we can think of the tracing process as traversing the object graph and coloring its nodes.
  - A node that must be retained is conceptually colored black.
  - A node subject to reclamation is conceptually colored white.
  - When there are no reachable nodes left to blacken, tracing is complete.

Towards The Tricolor Marking Abstraction

- Consider a *non-incremental* tracing collector.
  - Conceptually, we can think of the tracing process as traversing the object graph and coloring its nodes.
  - A node that must be retained is conceptually colored black.
  - A node subject to reclamation is conceptually colored white.
  - When there are no reachable nodes left to blacken, tracing is complete.
- Mark-sweep collector
  - Objects whose mark bit is set are black; others are white.
- Copying collector
  - Unreached objects in from-space are white.
  - Objects moved to to-space are black.

Towards The Tricolor Marking Abstraction

- Consider a *non-incremental* tracing collector.
  - Conceptually, we can think of the tracing process as traversing the object graph and coloring its nodes.
  - A node that must be retained is conceptually colored black.
  - A node subject to reclamation is conceptually colored white.
  - When there are no reachable nodes left to blacken, tracing is complete.
- Mark-sweep collector
  - Objects whose mark bit is set are black; others are white.
- Copying collector
  - Unreached objects in from-space are white.
  - Objects moved to to-space are black.
- Unfortunately, this “bicolor” abstraction won’t work as-is for incremental collectors.

The Tricolor Marking Abstraction

- To handle incremental tracing, we need to introduce a third color, grey.
  - A node that has been reached by the traversal, *but whose descendants may not have been*, is conceptually colored grey.
  - When an object is reached by the traversal, it is initially colored grey.
  - After it is fully scanned and pointers to its descendants have been traversed, it is blackened (and its descendants are greyed).

The Tricolor Marking Abstraction

- To handle incremental tracing, we need to introduce a third color, grey.
  - A node that has been reached by the traversal, *but whose descendants may not have been*, is conceptually colored grey.
  - When an object is reached by the traversal, it is initially colored grey.
  - After it is fully scanned and pointers to its descendants have been traversed, it is blackened (and its descendants are greyed).
- Mark-sweep collector
  - Grey objects correspond to those in the stack or queue used to control the marking traversal.

The Tricolor Marking Abstraction

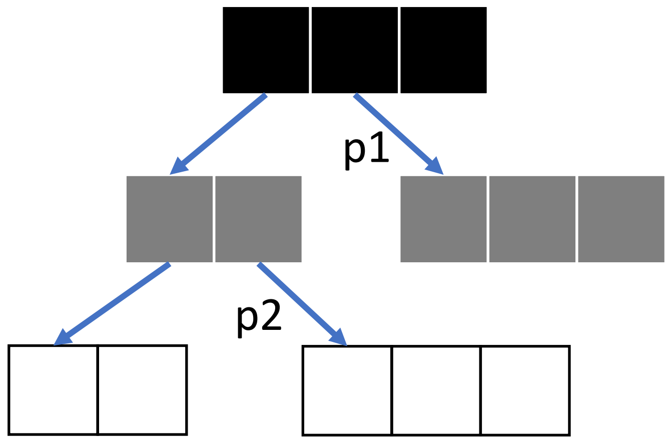
- To handle incremental tracing, we need to introduce a third color, grey.
  - A node that has been reached by the traversal, *but whose descendants may not have been*, is conceptually colored grey.
  - When an object is reached by the traversal, it is initially colored grey.
  - After it is fully scanned and pointers to its descendants have been traversed, it is blackened (and its descendants are greyed).
- Mark-sweep collector
  - Grey objects correspond to those in the stack or queue used to control the marking traversal.
- Copying collector
  - Grey objects are the ones between the scan and free pointers in to-space.
  - Objects that have been passed by the scan pointer are black.

Implications of Tricolor Marking

- Traversal of the object graph proceeds in a wavefront of grey objects that separates black objects from white ones.
- This fringe of grey objects must be well-defined and identifiable.
- The mutator must preserve the invariant that no black object hold a pointer directly to a white object.
    - Any attempt to create such a pointer needs to be coordinated with the collector to update its bookkeeping.

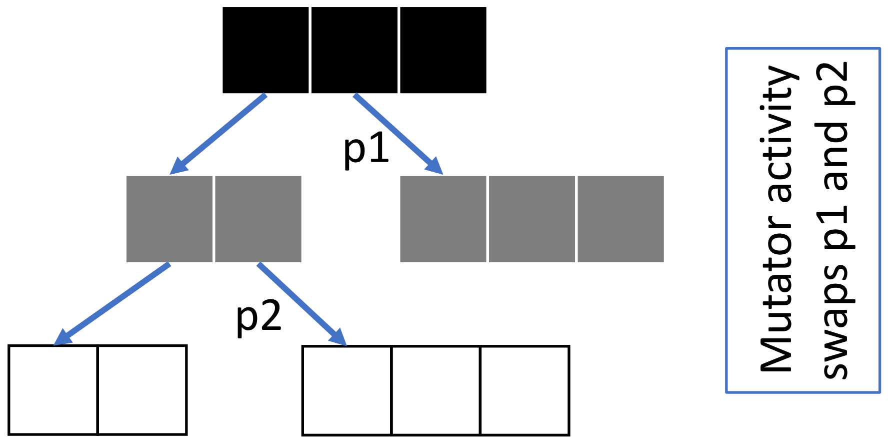
Implications of Tricolor Marking

- Traversal of the object graph proceeds in a wavefront of grey objects that separates black objects from white ones.
- This fringe of grey objects must be well-defined and identifiable.
- The mutator must preserve the invariant that no black object hold a pointer directly to a white object.
  - Any attempt to create such a pointer needs to be coordinated with the collector to update its bookkeeping.



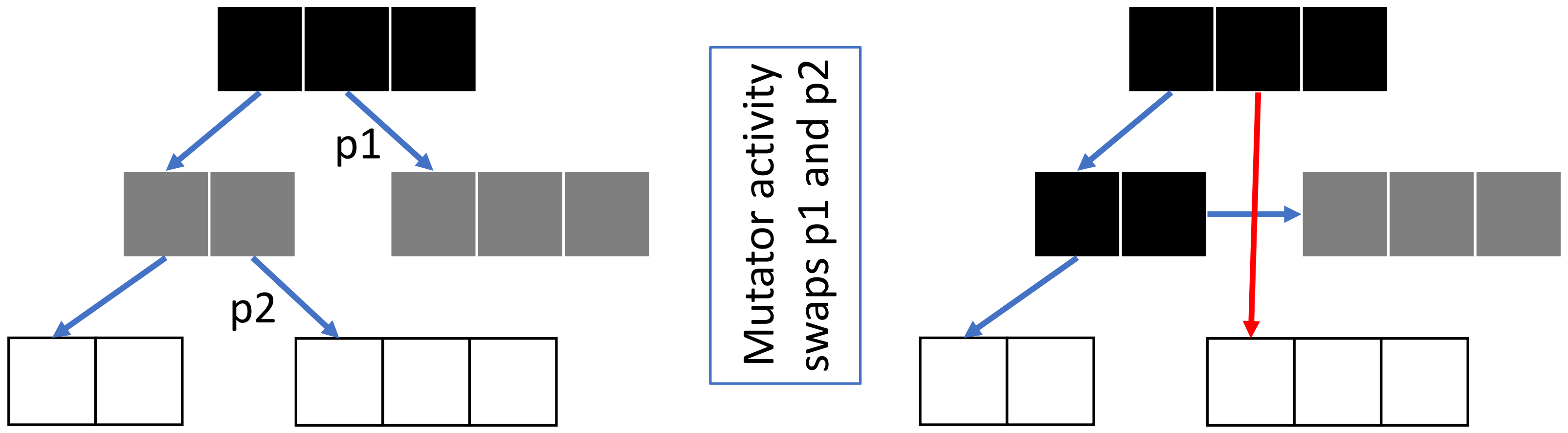
Implications of Tricolor Marking

- Traversal of the object graph proceeds in a wavefront of grey objects that separates black objects from white ones.
- This fringe of grey objects must be well-defined and identifiable.
- The mutator must preserve the invariant that no black object hold a pointer directly to a white object.
    - Any attempt to create such a pointer needs to be coordinated with the collector to update its bookkeeping.



Implications of Tricolor Marking

- Traversal of the object graph proceeds in a wavefront of grey objects that separates black objects from white ones.
- This fringe of grey objects must be well-defined and identifiable.
- The mutator must preserve the invariant that no black object hold a pointer directly to a white object.
    - Any attempt to create such a pointer needs to be coordinated with the collector to update its bookkeeping.



Collector-Mutator Coordination

- The problem scenario arises when:
  1. The mutator writes a pointer to a white object into a field of a black object; AND
  2. The mutator destroys the original pointer to the white object before the collector has traversed it.

Collector-Mutator Coordination

- The problem scenario arises when:
  1. The mutator writes a pointer to a white object into a field of a black object; AND
  2. The mutator destroys the original pointer to the white object before the collector has traversed it.
- Two basic approaches: read barrier and write barrier.

Collector-Mutator Coordination

- The problem scenario arises when:
  1. The mutator writes a pointer to a white object into a field of a black object; AND
  2. The mutator destroys the original pointer to the white object before the collector has traversed it.
- Two basic approaches: read barrier and write barrier.
- Read barrier
  - Detect when the mutator attempts to access a pointer to a white object, and immediately color it grey.

Collector-Mutator Coordination

- The problem scenario arises when:
  1. The mutator writes a pointer to a white object into a field of a black object; AND
  2. The mutator destroys the original pointer to the white object before the collector has traversed it.
- Two basic approaches: read barrier and write barrier.
- Read barrier
  - Detect when the mutator attempts to access a pointer to a white object, and immediately color it grey.
- Write barrier
  - When the mutator attempts to write a pointer into an object that fits a problem scenario, the write is trapped or recorded.
    - Snapshot-at-beginning: Prevent condition #2 from happening by first saving a copy of the old pointer for the collector to use.
    - Incremental update: Records pointers stored into a black object (thereby reverting it to grey), or immediately greying the referent.