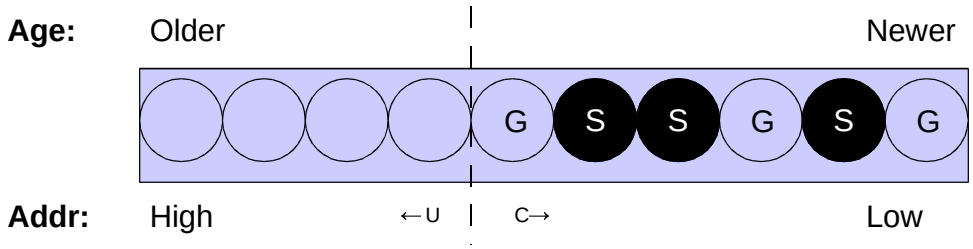
Age-Based Garbage Collection

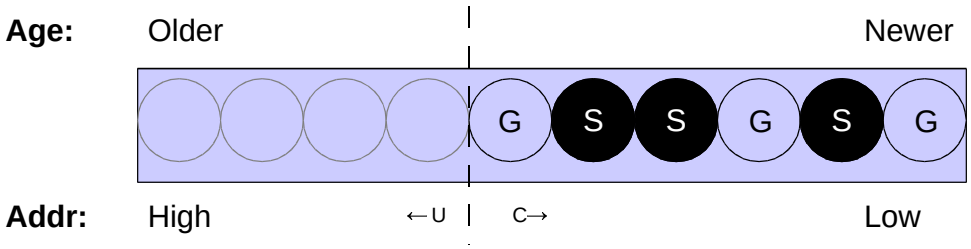
D. Stefanović
K. McKinley
J. E. Moss

Recap: Scavenging



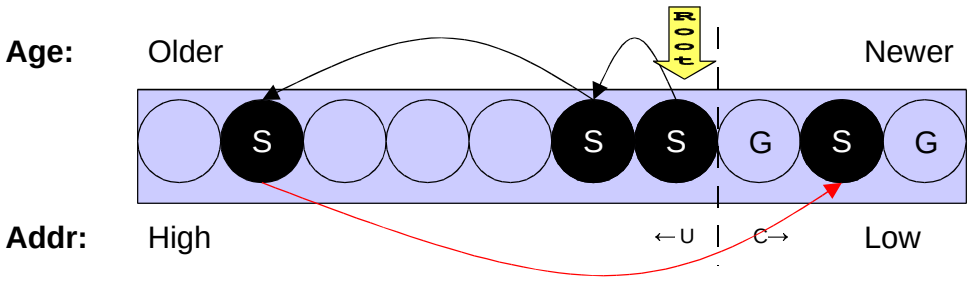
- **U** = {objects not examined} (assumed live)
- **C** = {objects examined}
  - **G** = {garbage}
  - **S** = {survivors} (copied/compacted)
- Work proportional to |**S**|

Minimizing |**S**|: Generational GC



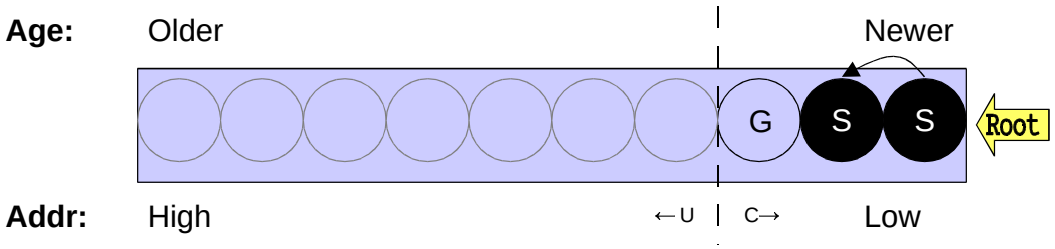
- Objects die young
- Let C = {young objects}
- Fewer survive ⇒ copy less

Generational Problem: Pointer Tracking



- Must track old-to-young pointers
- Total Cost = Copying + Pointer Tracking
- Tradeoff: Decreased copying ⇒ Increased tracking

Generational Problem: Promotion

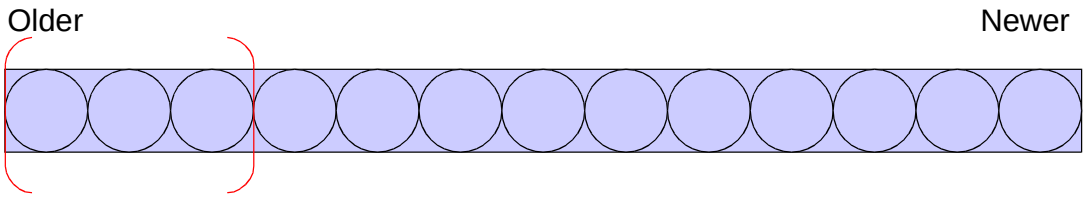


- Newly created objects promoted…
- … but die!
- Waste of time and space

Generational Solution: Older-First (OF)!

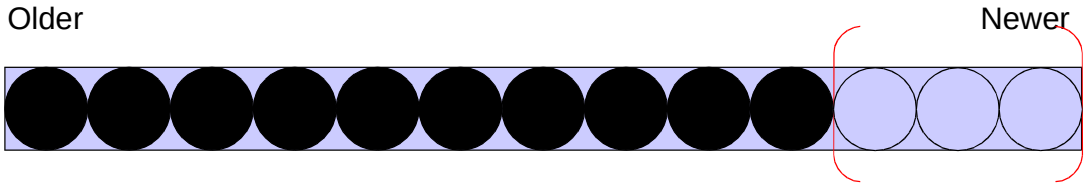
- “ Many objects wait until middle age to die…”
- Give objects time to die

Older-First Algorithm



- Sliding window determines **C**
  - Starts at at oldest end
- Survivors copied/compacted
- Window positioned to right of survivors

Older-First Algorithm: “Window Reset”

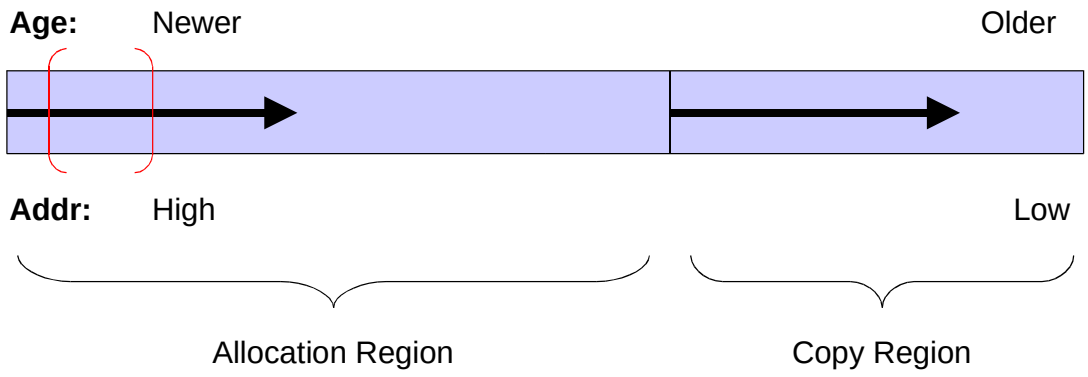


- Window hits allocation point
  - Collect/compact objects
  - Reset window to oldest objects

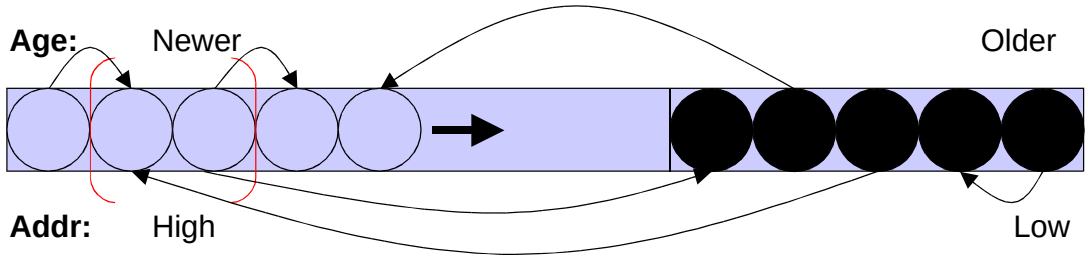
Copy Cost

- Does not copy youngest objects
- Catches middle-age objects just as they die
- Does not reconsider survivors
- “ Sweet Spot” ⇒ low mark/cons
- Revisits old (and large?) objects
- Fraction of other generational collectors

Write Barrier: Heap Organization



Write Barrier



- Remember $q \leftarrow p$
  - If q will be collected before p ($p < q$) and
  - If q and p in different blocks

Pointer-tracking Costs

- Time: can be made efficient
  - BUT: More boundaries than other collectors
- Space:
  - Measured 0.01*heap size more than 2G
  - Get good mark/cons in small heap
- Outweighed by savings in copy-cost
  - Up to 4x total cost reduction

Age-Based GC: Contributions

- Copying vs. Pointer-Tracking Cost
- Older-First (OF) Algorithm
- Write Barrier