Distributed Garbage Collection

Distributed Systems
Santa Clara University
Unreferenced Objects

- Root set
- Entities forming an unreachable cycle
- Reachable entity from the root set
- Unreachable entity from the root set
Unreferenced Objects

- Reference counting is necessary but not sufficient to identify unreachable objects
- Distributed systems:
  - More difficult if messages
    - can get lost
    - can be duplicated
Unreferenced Objects

Diagram:
- Process P
- Proxy p
- Object O

Steps:
1. +1
2. ACK
3. +1
4. ACK

Notes:
- ACK is lost
- Proxy p is now counted twice
- Skeleton (maintains reference counter)
Unreferenced Objects

A race condition and a possible solution
Unreferenced Objects

- Prevent race condition between incrementing and decrementing a reference counter
- **Weighted Reference Counting**
  - Only use decrement operations
Unreferenced Objects

(a) Initial assignment
(b) After creating a new reference
(c) When copying a new reference
Unreferenced Objects

• Weighted reference counting:
  • Two values: total weight and partial weight
  • When a process creates an object, both are set to $2^{**7}$
    • Half partial weight
    • Move partial weight to process
  • When a process sends a reference:
    • Half its weight
    • Send this weight to other process
  • When a reference is destroyed:
    • Send decrement message to object’s skeleton
    • Remove the weight from the total weight
  • Can remove object when total weight and partial weight are equal
Unreferenced Objects

- Weighted reference counting
  - When a process runs out of partial weights
  - Creates a new skeleton
Unreferenced Objects

- Weighted reference counting
  - Generation reference counting
  - Use generation numbers
Unreferenced Objects

- Skeleton needs to maintain a generation counter
  - $G[i]$ — number of copies for generation $i$
  - If proxy $p$ is removed, send delete message to proxy with generation number $k$ and the number $n$ of copies made
  - Skeleton decrements $G[k]$ and adds $n$ to $G[k+1]$
Reference Counting

• Reference Listing
  • Object contains a list of proxies with a reference to it
    • Adding and removing from a list are idempotent operations
  • Used in Java RMI
Unreachable Entities

• Naïve tracing
  • Mark and sweep collectors
    • Define a root set
    • Create a list of reachable objects by:
      • Expand each object by the objects it references
      • Maintain a frontier of objects reached, but not yet expanded
    • Sweep the space for objects that have not been marked
Unreachable Entities

• Need something that is scalable
  • Group processes into groups
  • Collect garbage within a group
  • Consider a larger group made up of recently cleaned groups
Unreachable Entities

• Skeletons can be marked
  • None
  • Hard — reachable from outside the group
  • Soft — reachable from proxies within the group

• Can only update from soft to hard
Unreachable Entities

- Proxies
  - Marked
    - Hard — reachable from an object in the root set
    - Soft — reachable from a skeleton that has been marked soft
  - Proxies marked soft can lie on a cycle that is not reachable from an object in the root set
Unreachable Entities

- Step 1: Mark skeletons
- Step 2: Propagate marks from skeletons to proxies
- Step 3: Propagate marks from proxies to skeletons
- Stabilize by repeating steps 2 and 3
- Now reclaim garbage
Unreachable Entities

• First step (initial marking):
  • Marking skeletons
    • Mark hard
      • Skeleton can be reached from a proxy outside of the group
    • Mark soft
      • Skeleton can only be reached from a proxy within the group
      • Should be all of them
  • Use reference count
    • For each proxy in group, decrement the reference counter of the associated skeleton
    • Mark those with reference count 0 soft, the other ones hard
Unreachable Entities

• Second step (propagation):
  • Each process runs *local* garbage collection
  • Get read of unneeded proxies
  • Mark needed proxies with the mark of the skeleton

• Algorithm:
  • Start with objects in root set and skeletons marked hard
  • Mark proxies and objects reachable with hard
  • Repeat for soft proxies (but do not change from hard to soft)
Unreachable Entities

- Third step (propagate marks from proxies to skeletons):
  - Propagate marks between processes:
    - Send message from a hard proxy to its skeleton
      - (as long as it is in the group)
    - Skeleton is now marked with hard
Unreachable Entities

• Fourth step (Update skeletons):
  • When a process P has a skeleton re-marked from soft to hard
    • Update the corresponding proxies
  • Repeat until the situation is stable
  • This is an example of group-commit:
    • All need to be ready
Unreachable Entities

• Fifth step:
  • Now can remove soft proxies, skeletons, and objects
(a) Initial marking of skeletons
(b) After local propagation
(c) Final marking
Unreachable Entities

• Organize group in a hierarchy
  • Low-level group collect garbage
  • Then higher-level group redoes the process

• Idea:
  • Most garbage collection can be done at the local level