Promotion policies, Generation Org and Age

How do we organize generations or record age?

Goals of generational collection

- Aims of generational GC:
  - Reduce cost of dealing with long lived objects
  - Reduce garbage collection pause time
    - Interactive program test
    - Depends on amount of data that survives a collection
    - Depends on size of generation
      - small → more frequent collection
      - Large → less frequent collection
  - Achievable by segregating objects by age

What policies to use for promotion?

- Multiple generations
- Promotion threshold
- Adaptive tenuring

Dilema for fixed promotion policies

- Consider small youngest generation
  - Shortens interval between scavenges
  - Shortens pause length
- Consider larger generations
  - Reduces promotion rates
  - Gives objects longer to die
  - Scavenges less often → copying overhead is reduced
    - But pause length is increased
- So how does fixed promotion policies handle this dilema?

Adaptive tenuring

- Tuning generational collection is complex and time consuming
- What if program has varying allocation rates?
  - Fixed policies does not have a way to adjust tenure rate and prevent collector from thrashing
- Adaptive tenuring:
  - Promotion policy that allows promotion criteria to vary

How Adaptive tenuring works

- Invoke collector when volume of data allocated since last collection exceeds an **allocation threshold**
- Dynamically vary size of semi-spaces if necessary
- Threshold-based policy are more stable than fixed-size generation policy

Two flavors of adaptive tenuring

- Only tenure when it is necessary
- Only tenure as many objects as necessary
- Note:
  - Objects' age given in bytes allocated
    - More memory allocated since object creation → older object
    - Less memory allocated since object creation → younger object
  - Pause time given as bytes copied

Tenure only when necessary

- # of objects that survive a scavenge is used to predict pause time of next scavenge
  - Definition of pause time
  - Time measured in bytes
  - If few objects survive a scavenge (less than threshold)
    - Probably not worth promoting them
    - GC pause less than max acceptable pause
    - Consider write-barrier cost

Tenure # of objects as necessary

- If survivor size suggests maximum pause time (in bytes) would be exceeded at next scavenge
  - Set age threshold to value to allow excess data to be promoted
  - Survivors scanned to produce table recording volume of object of each age
  - Table then scanned (descending order) to look for promotion threshold for next minor collection

Pioneers of adaptive tenuring

- Ungar and Jackson → feedback mediation
  - Varies tenure rate depending on volume of survivors
- Barett and Zorn → threatening boundary and remembered set
  - Boundary between 2 generations is allowed to move in either direction
  - Set of addresses of objects in old generation that point objects in younger generations

Generation organization

- One semi-space per generation
  - Simplest promotion policy:
    - Advance all live objects at once
    - No need to record object ages
    - Use older generation as to_space OR recycle youngest gen.
    - Requires multiple generations to filter tenured garbage
    - Promotion rate is high

Generation organization

- Creation space
  - Divide generation into **creation space** and **aging space**
  - Allocate objects in creation space
  - Aging space stores survivors from creation space
  - # of survivors of each scavenge expected to be low, both spaces can be small
  - For good performance
    - Hold creation space in memory
    - Do not swap it out

Age recording

- Not necessary for *en masse* promotion schemes
  - All survivors are promoted
- Methods requiring object's age must
  - Record object's age in its header
    - Cost? Manipulated? Copied?
  - Segregate objects of different ages within a generation
    - Shaw uses buckets
    - New bucket → aging bucket → next generation
    - Buckets != generation

What about large objects?

- Use large object space
  - Use as in copy collector
  - Save pause time