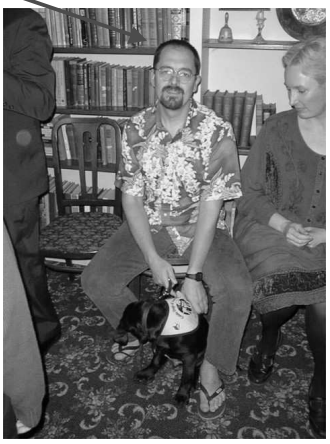
Using Key Object Opportunism to Collect Old Objects

Barry Hayes

Presented by Mike Bond

Barry Hayes

- Stanford grad student in 1991
- PhD thesis: “Key Objects in Garbage Collection” (1993)
- Co-author of 1997 OOPSLA paper on GC
- Worked at Placeware at least through 1999 – published some computational geometry stuff



The basic idea

- The strong generational hypothesis isn't useful for old objects
- Objects tend to die in groups

- So let's use *key objects* as representatives for groups of objects
- How do we pick good key objects?

Outline

- The generational hypotheses
- Insights into behavior of object death rates
- The algorithm
- Finding key objects
- Conclusion
- Discussion

The weak generational hypothesis

- "Newly-created objects have a significantly lower survival rate than objects that are older"

  … assuming that the newly-created objects have been given enough time to die

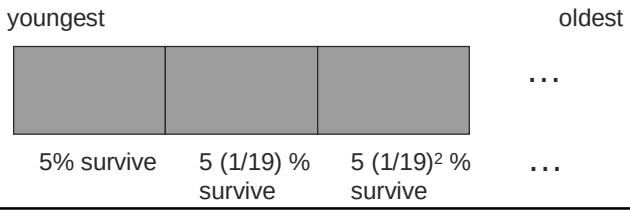
- Divide memory into nursery and old area
- Collect nursery more often

The strong generational hypothesis

- "The relatively younger objects have a lower survival rate than the relatively older objects"
- This means death rate decreases over time
- Implies that many generations is good idea

Suppose otherwise

- What if no relation between age and death rate?
- Suppose constant death rate of 95% per nursery collection



Strong generational hypothesis, continued

- Hayes and others do not observe this exponential decay in survival rate
- Data suggests roughly linear relationship between object age and half-life
- This means that for old objects, when age increases, death rate goes down, but not by a whole lot

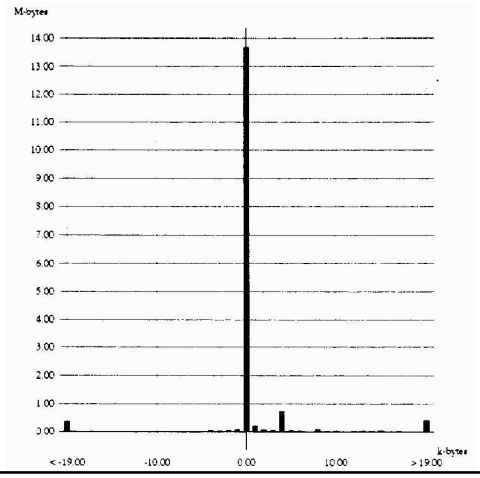
Discussion: Strong generational hypothesis

- Does Hayes agree with the generational hypothesis?
- Is the strong generational hypothesis correct?
- Generational collectors typically only have two or three generations

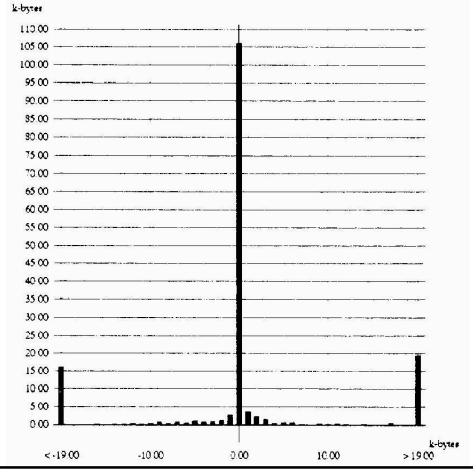
Alternative indicators

- For new objects *that have been given time to die*, death rate decreases a lot as object age increases
- For old objects, death rate does not decrease much as object age increases
- A death indicator other than age might work well for old objects

Difference in age between successively deallocated objects



Same, but for oldest 1% of objects



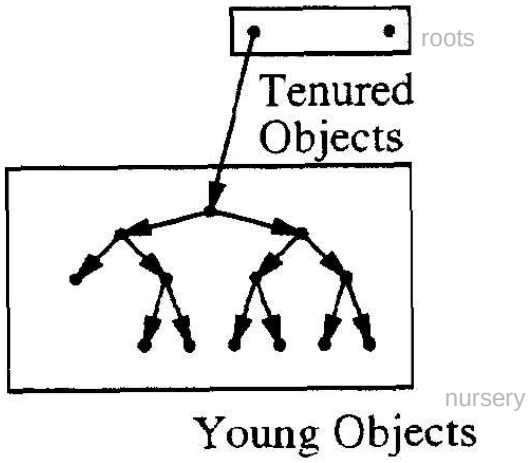
Discussion: Motivation

- First, paper says that for old objects, death rate doesn't drop enough as objects age
- Then it changes direction and suggests that objects die with their siblings
- Does the first part motivate the second?

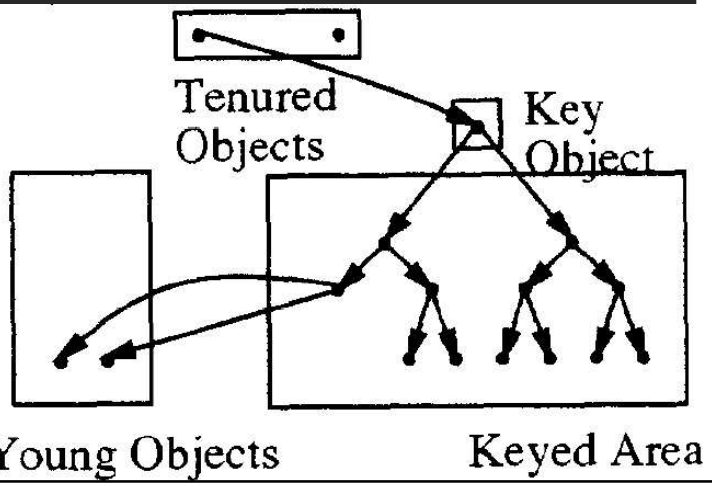
Key object opportunism

- Objects that are allocated at about the same time tend to die at about the same time
- Pick a representative object, called the *key*, and check its liveness more often than for the other objects in the cluster

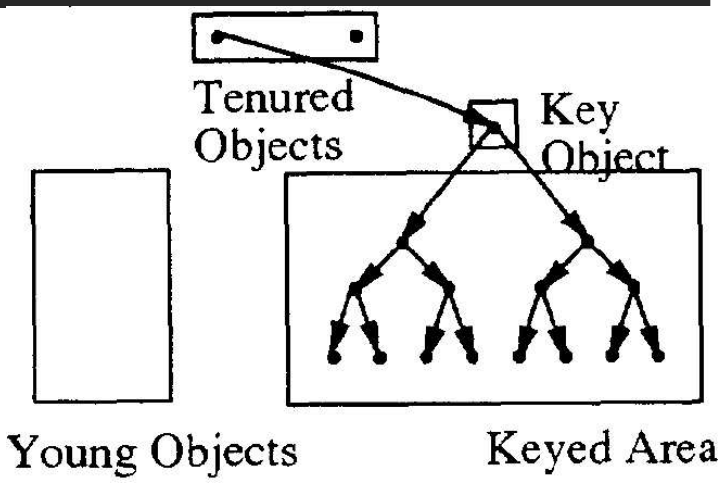
Cluster of new objects in nursery



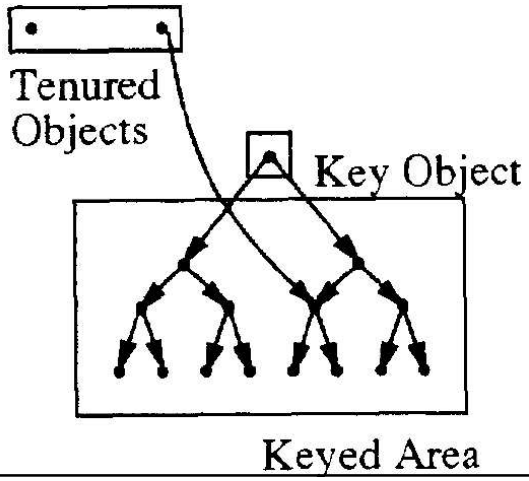
Collection occurs; key object identified



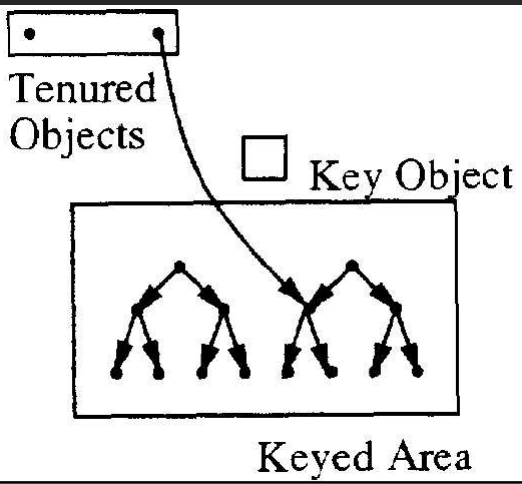
Recently-created objects copied to keyed area



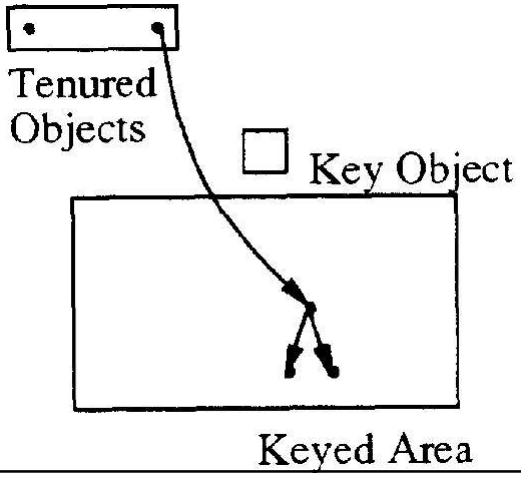
Key object becomes unreachable



First try to collect key object



Key object was dead, so collect keyed area



Finding key objects

- Nice idea in theory… but how do we choose key objects?
  - Random selection
  - Key discovery
  - Stack-based
  - Serendipity
  - Hints

Random selection

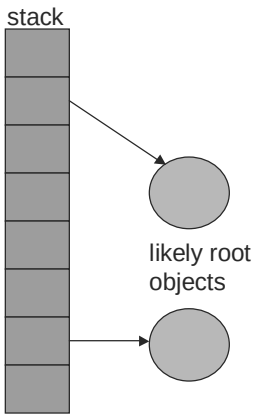
- After a collection, randomly select fraction of survivors to be keys
- Many pointers from keyed area to keys
  - Keys look live even when cluster dead
- Discussion: What would random selection do to locality?

Key discovery

- Collector observes generational behavior
- Tries to pick likely roots when the survivors are promoted out of the generational space

Stack-based key objects

- At collection, objects pointed at by stack frame are likely keys
- Intuition comes from functional languages
- Discussion: Does it apply to Java?



Serendipity

- A good key might not be part of the cluster
  - Might be in an associated structure
- Good key might not be found by the key-finding heuristic
  - If so, application can allocate a dummy object that will get picked to be a key
  - Similar to hint but without extra semantics

Hints

- Implicit: Collector derives likely keys based on program behavior
- Explicit: User specifies key objects in source
  - `implements PragmaKey`

Conclusion/Contributions

- Data analysis that suggests that the strong generational hypothesis is not very helpful for old objects
- General algorithm for key object opportunism
- Several vague heuristics for finding keys

Discussion: Object clusters

- Paper suggests that objects created at the same time tend to be connected to each other
- Is this always true?
- Does it matter?

Discussion: Encapsulation

- In object-oriented languages, container objects might make good keys
  - Example: container class for a binary tree
- Compiler could mark likely container objects
  - Might be hard to pick statically
- Your thoughts?