Automatic Memory Management

marcel.hlopko@fit.cvut.cz

Manual Deallocation

need for bookkeeping
can hinder design and extensibility
gc can be faster, usually comparative
all large applications end up
implementing some form of
automatic memory management
themselves

History of GC

list and prolog
smalltalk
self
java & .NET

Phases

finding live / garbage objects
reclaiming garbage

Root Set

globals
local variables in stack frames
(registers)

Reachability

Any object reachable from root set is live
Everything else is garbage

Finding Live Objects

Reference Counting
Tracing

Reference Counting

refCount stored in object header

refCount == 0 -> object reclaimed

Reference Counting

- pros
  - incremental nature
  - easy to make real-time
  - degrades well with full heap
- cons
  - cycles
  - overhead
  - fragmentation

Mark-Sweep

tracing algorithm
marking live objects
sweeping all unmarked

Mark-Sweep

- pros
    - handles cycles
- cons
    - fragmentation
    - bigger heap - longer run

Mark-Compact

solves fragmentation and ref. locality

pros

- simple allocation

cons

- still multiple passes over heap

Copying GC

semispaces

forwarding pointer

pros

- only one run over heap

cons

- still has to stop the world

Non-Copying Implicit Collection

- two sets (used, free) implemented as doubly-linked lists
- pros
    - does not move objects in memory
- cons
    - still has to stop the world

Incrementa Collectors

GC runs in parallel with application (mutator)

relaxed consistency

Tricolor Marking

black - will be retained
white - will be collected
gray - will be expanded

Assigning White Pointer to Black Object

collector has to be notified

- read barrier
- write barrier

Read Barrier

detect accesses to white objects and
color them gray immediately
usually too expensive

Write Barrier

detect writes into black objects

snapshot at beginning

- copy-on-write

incremental update

- marking black to gray again

Baker's Read Barrier GC

atomic flip
any fromspace object used by mutator has to be copied to tospace first
(special hw support)

Generational GC

80-98% short lived objects

many survivors will survive a lot :)

Remembered Sets

pointers from old to new added to root set

Questions And Discussion