Basics of Garbage Collection

Merlin Laue

Universität zu Lübeck

9. November 2015

Table of Contents

1. Motivation
2. Problems
3. Algorithms
4. Exact vs. Conservative
5. Conclusion

Motivation

- Managing memory manually is time consuming
- Very precise work required to prevent memory leaks
- Solution $\rightarrow$ Automatic memory management

Basic Terms

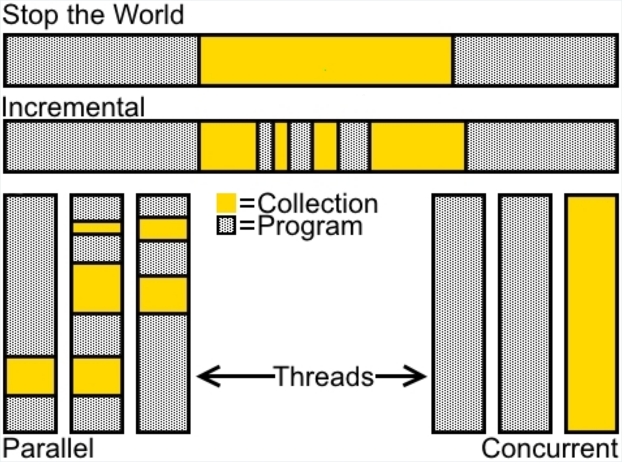
Definition (Mutator)

A mutator is part of a running program which executes application code. Its name is based on the fact that from the collector's point of view it mutates the graph of objects.

Definition (Free-list)

A free-list works by linking unallocated regions of memory together in a linked list. Newly freed memory is added to the list.

Timings


Stop the World
Incremental
=Collection
=Program
Threads
Parallel
Concurrent

Problems

1. Memory Usage
2. Fragmentation
3. Determinism
4. Time Effiency

Memory Usage

- Free-lists or tables
- Overhead consisting of state, location, or size of an object
- Copy spaces and Barriers
- Finalization
- Cyclic data structures

Fragmentation

- Method of collection is a big influence
- Performance can drop drasticially.
- Countering fragmentation is expensive if it is not inherently implemented

Determinism

- Finalization most important influence
- Destructor → Immediate freeing
- Finalization → Unforseeable
- Occupation of shared resources can be problematic

Time Efficiency

- Garbage collection show close affinity to the amount of allocated memory
- Number of collection cycles $<$ Mutator Locality with contiguous allocation

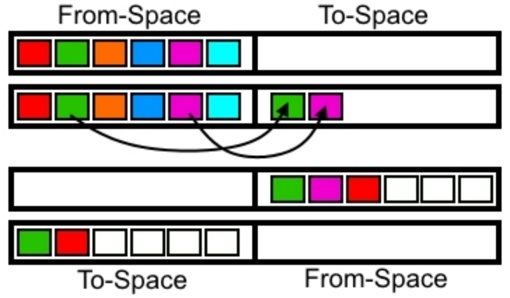
Algorithms

1. Semi Space
2. Reference Counting
3. Mark and Sweep
4. Mark and Compact
5. Generational

Semi Space

- Requires 2 spaces : To-Space and From-Space
- Collection time is proportional to the number of survivors inhabiting the full copy space
- Collects the entire heap everytime
- 'Stop the World' - collector

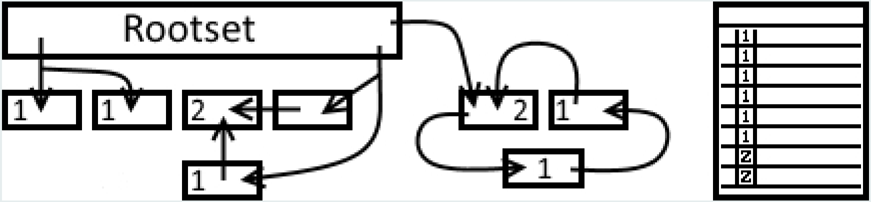
1. Swaps both spaces
2. Traces all root referents and copies uncopied objects to the complementary half
3. Leaves forwarding pointers in the old object
4. Adjusts references to point towards the new adress.



Reference Counting

- Uses free-lists to note the amount of references pointing towards any given object
- Requires write barriers to count
- The burden on the mutator is significantly increased
- Cyclic data structures are a big problem
- 'Stop the World' - collector

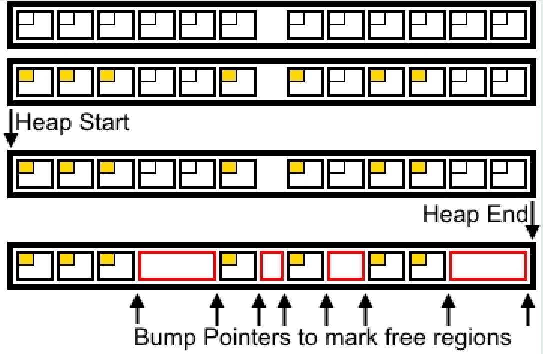
1. Upon first allocation an entry in the free-list is created
2. The write barrier records increases and decreases of references
3. Afterwards the amount of references is only buffered in the free-list
4. If the references of an object reach 0 then a bit in the free-list is set mand all all references are recursivly decreased



Mark and Sweep

- Uses a free-list with a tracing collection
- Uses a bitmap to mark objects
- 2 Phases of collection : Marking and Sweeping, hence its name
- Handles cycles due to whole heap scanning
- Severe fragmentation
- 'Stop the World' - collector

1. Stopping of the program and mutator activity
2. Marking of alive objects
3. Sweeping : Scanning the heap for unmarked objects and freeing these parts
4. Resuming mutator activity when reaching the end of the heap



Mark and Compact

- Essentially a Mark and Sweep algorithm
- Better allocation times due to compact memory
- Memory efficient if only the objects are observed
- Overhead for forwarding pointers extremely big
- 3 runs for each compacing phase

1. Marking of alive objects
2. Compute forwarding adresses
3. Update pointers for the referents
4. Relocation of objects

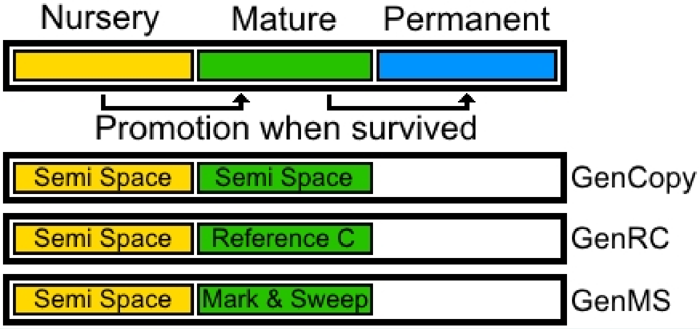
A new View : Generational

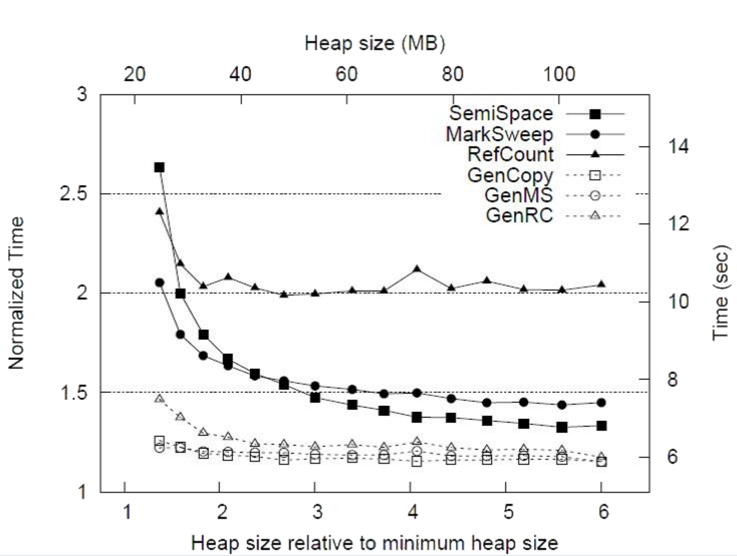
Theorem (Weak generational Hypothesis)

Newly created objects tend to die very young while old objects persist much longer.

- Partition the memory in 3 generations:
  1. Young (Nursery)
  2. mature
  3. Permanent
- Promote objects who survive collection to an older generation
- Use the different algorithms to collect each generation (Generational collectors are hybrids)

- Different size policies for the young generation:
  1. Flexible - Expands and retracts with collection and allocation
  2. Fixed - Fixed size which will never expand or retract
  3. Bound - Uses upper and lower boundry




Heap size (MB)
20
40
60
80
100
3
2.5
2
1.5
1
Normalized Time
SemiSpace
MarkSweep
RefCount
GenCopy
GenMS
GenRC
14
12
10
8
6
Time (sec)
1
2
3
4
5
6
Heap size relative to minimum heap size

Worlds collide - Exact vs. Conservative

Definition (Exact)

Exact algorithms are assisted by the compiler and language runtime. In case of strongly typed languages such as C, a shadowstack has to be used for exact collection making them uncooperative exact.

Definition (Conservative)

A conservative collection algorithm 'guesses' the references. It scans the memory (most notably the execution stack) for words looking like a reference and then makes the *conservative assumption* that it is a valid reference. These references are called ambigiuous references.

- Leads to 3 major restrictions and drawbacks:
  1. Pinning
  2. Filtering
  3. Excess retention
- Pinning incurs fragmentation
- Filtering adds additional workload
- Excess retention is space overhead
- Immix collectors are a recent way to show heavily optimized conservative algorithms

Conclusion

- Locality of the mutator is more important then the number of collections
- Compiler and language runtime limit the choice of garbage collectors
- Garbage collection does not necessarily need support.
- The Performance of garbage collection will steadily increase in the future.

Thank you for your attention

Feel free to ask questions