Mark-Sweep GC and Mark-Compact GC

Richard Jones, Anthony Hoskins, Eliot Moss

Presented by Pavel Brodsky, 04/11/14

In the presentation we discussed two of the basic garbage collection algorithms: Mark-Sweep and Mark-Compact. Both algorithms attempt to locate all the *live* objects in the memory heap, and free up the memory of all the unmarked objects (the *garbage*). Because true *liveness* is undecidable, *pointer reachability* is used instead - an object is deemed *live* iff it can be reached by following a path from a set of known roots.

Mark-Sweep has two phases: first, we locate and *mark* all *live* objects in the heap (with a DFS from the roots). The marking is done by setting a *mark-bit* in the header of an object, or by using a special table - a *bitmap*. In the second phase, all the heap is traversed, and the memory occupied by any object that didn't have its *mark-bit* set is freed. Mark-Sweep is very fast and simple, but prone to severe *fragmentation* problems.

Mark-Compact also has two phases: first, we *mark* all the *live* objects, as before. Then we *relocate* (move) objects on the heap, with the intention of *compacting* the heap so that all the *live* objects would be located at the beginning of the heap, and all the memory that comes after those objects will be free for allocation. The most common and widely used relocation is technique is called *sliding* - all the live objects are slid to the beginning of the heap, "squeezing out" the garbage, while retaining their order (and thus, the spatial locality). Mark-Compact solves the *fragmentation* problem, usually at the cost of memory overhead, and longer running time.

My original contribution was the Snapshot algorithm. The idea behind it is the fact that the set of dead objects doesn't shrink, so the mutator threads don't have to be stopped while garbage collection is performed. Mutators are stopped for a short period, while a snapshot (a copy) of the relevant part of the heap is taken. That copy is then traversed to locate all the dead objects. The last step is to traverse the original heap (possibly concurrently with mutator threads) and free all the objects that were deemed dead in the copy (because those couldn't possibly come back to life in the original).

During the discussion after the presentation a couple of improvements to Mark-Sweep came up:

- Using a FIFO buffer for prefetching objects ahead of time.
- Using *edge-marking* to reduce the number of cache misses.