Discussion questions for *Parallel Generational-Copying Garbage Collection with a Block-Structured Heap*

N. Ramsey and D. Nunez, for COMP 250RTS

October 23, 2017

We will focus on sections 1 to 4. I recommend also taking a brief look at sections 5.5, 5.6, and 5.8 (lock contention, fragmentation, and other topics). We will also compare GHC's parallel collector with Andrew Appel's simple generational collector.

Basics

(1) *Allocation*.

  (a) When the mutator needs to allocate an object, what does it do?
  (b) How does the allocation cost compare with Appel's allocation cost?

(2) *Pending Block Set*. Section 3.3 describes the Pending Block Set. What invariant describes the blocks in the Pending Block Set?

(3) *Generations and steps*.

  (a) What is the role of generations?
  (b) What is the role of steps?
  (c) Why have both?
  (d) Can Appel's collector be described using generations and steps? If so, explain how many generations Appel's collector has, and how many steps are in each generation. If not, explain why not.

Scavenging

(4) *State transitions*. A thread's state can change in response to one of two events: an object is scanned, or an object is evacuated into an Allocation Block.

  (a) Convert Figure 3 to a state diagram. Label each state with invariants.
  (b) Label each edge in the diagram. Write an event, plus any necessary side conditions.
  (c) How does Appel's collector traverse this state diagram? What states does it visit, and what edges does it use?
  (d) Suppose you adapt GHC's collector to run sequentially, on a single processor. How would the adapted collector traverse this state diagram?

(5) *Correctness*.

  (a) When all GC threads have finished scavenging, what objects have been evacuated?
  (b) How would you prove it?

Bonus questions

(6) *Block descriptors*. Section 2.3 describes the placement of block descriptors. An early, important collector by Hans Boehm and Mark Weiser placed each block descriptor at the beginning of the block. What reasons can you think of why this might be a bad idea? Does GHC's design address these reasons?

(7) *Extended load-balancing*. (Note: This question seems really interesting, but it's too much for a 75-minute class.)

  Section 3.6 describes a relatively sophisticated algorithm for balancing load among GC threads.

  (a) Using pseudocode, diagrams, or really anything but the informal English used in the paper, model the algorithm.

  (b) With respect to your model, how would you characterize both *correctness* conditions and desired *performance* (including fragmentation)?