Real-Time Garbage Collection

Based on "The Garbage Collection Handbook" – Chapter 19

By Richard Jones, Antony Hosking and Eliot Moss

Presented by Boris Dogadov
20.1.15

Summary

The paper begins with describing real-time applications. Special requirements and programming paradigms are introduced.

Real-Time GC properties

"Minimum mutator utilization time" (MTU) parameter is introduced. MTU measures the mutator execution time as the percentage of overall execution time. Blelloch and Cheng define this property, and notice that it is a generalization of the previous wide-spread term "pause time". MTU is the most used parameter in the modern real-time garbage collectors.

Approaches used in implementation of Real-Time GC

Work-Based: Every mutator performs K steps of collector's work, as a tax for mutator work. Blelloch and Cheng (2001) present their GC based on this technique. They are first to derive strict time and space bounds for the worst case. A drawback of this algorithm is that the worst case is very far from the average case, which causes overprovisioning.

Slack-Based: Collector high-priority task is presented, along with the real-time tasks (which have the highest priority), and the low-priority tasks. Mutators are not taxed with collector's works, and are free from almost any overhead. Collector task is scheduled as any other real-time task, according to its priority. Henriksson introduces implementation of this idea, which results with much usable analysis of time and space bounds.

Time-Based: Collector and mutator are scheduled for well-known quanta of times. The quanta granularity and the scheduling are derived from desired MTU, minimum pause time, and the memory constrains. A well-known real-time GC based on this technique is Metronome, which is successfully implemented and used by variety of applications.

Tax-and-Spend: A combination of all the former approaches. A global 'Tax Bank' is introduced. Collector's work is contributed to the tax-bank, and mutator's works draws credit from it. Whenever there is not enough credit available, the mutator are obligated to perform collector's work (work-based), and vise-versa. The underlying GC is incremental, concurrent and parallel. There is a working variation of previously mentioned Metronome, based on this technique, which yields better performance than the original Metronome.

Original Idea

1. I've decided to investigate a bit the practical aspect of real-time garbage collection. Personally I 'grew up' in a paradigm which automatically relates the terms like 'real-time, performance, embedded' to C++ or GC-Disabling techniques in managed languages. I have found a variety of embedded, and considered to be "real-time" projects that make use of Java micro-edition and .Net micro framework.
2. I've presented a recent research regarding Garbage Collections in GPU. This is very experimental and uninvestigated topic. Two major works were presented in class:
    a. Dr. Ronald Veldema and Prof. Michael Philippsen present a garbage collection for CUDA. They introduce a small wrapper around CUDA. Allocator and collector are written in CUDA. This is first work that presents GC for GPU.
    b. Martin Maas, Philip Reames, Jeffrey Morlan, Krste Asanovic, Anthony D. Joseph, John Kubiatowicz present an experimental GC, that offloads part of the GC work to GPU. As opposed to the previous work, this is regular Java GC, which partly utilizes the GPU for its computation. The underlying GC is mark-and-sweep. The main idea is to maintain a reference graph (kind of a heap snapshot), and to offload the 'mark' work into the GPU.
        i. https://github.com/preames/gpu-garbage-collection a variation of **Jikes RVM** with GPU offload.

Discussion

- The need and usability of real-time garbage collectors in general.
- A discussion about the difficulty of fine-tuning the different Real-time garbage collectors. The need of providing strict memory assumptions ahead.
- The need of GC for languages (libraries) like CUDA was questioned, and feasibility of further work in the area of GC GPU-offloadinga was discussed.