Automated Trace File Generator for Virtual Machine Technique Evaluation

Konstantin Nasartschuk, Marcel Dombrowski, Kenneth B. Kent
University of New Brunswick
Faculty of Computer Science
kons.na@unb.ca, marcel.dombrowski@unb.ca, ken@unb.ca

Outline

- A trace file generation which mimics the behavior of an application in a virtual machine was developed.
- Based on specified probabilistic parameters, trace file of arbitrary structure and size can be created for experimentation purposes.
- An n-generational garbage collection policy was implemented in order to test and evaluate the benefits and trade offs of multiple garbage collection generations.
- The evaluation of the policy was performed using the trace file generator.

Motivation

- Virtual machine testing relies on existing applications. The size and structure of a benchmark is application specific.
- A general evaluation of a garbage collection technique based on probabilities provides the benefit on testing different application structures without the need of an actual application.
- Trace files mimic application behaviour in handling memory on the heap such as object creation and reference change.

Trace File Generator

- A random walk based algorithm which generates allocations and reference changes, creates an object graph aiming to mimic the objects of an actual application.

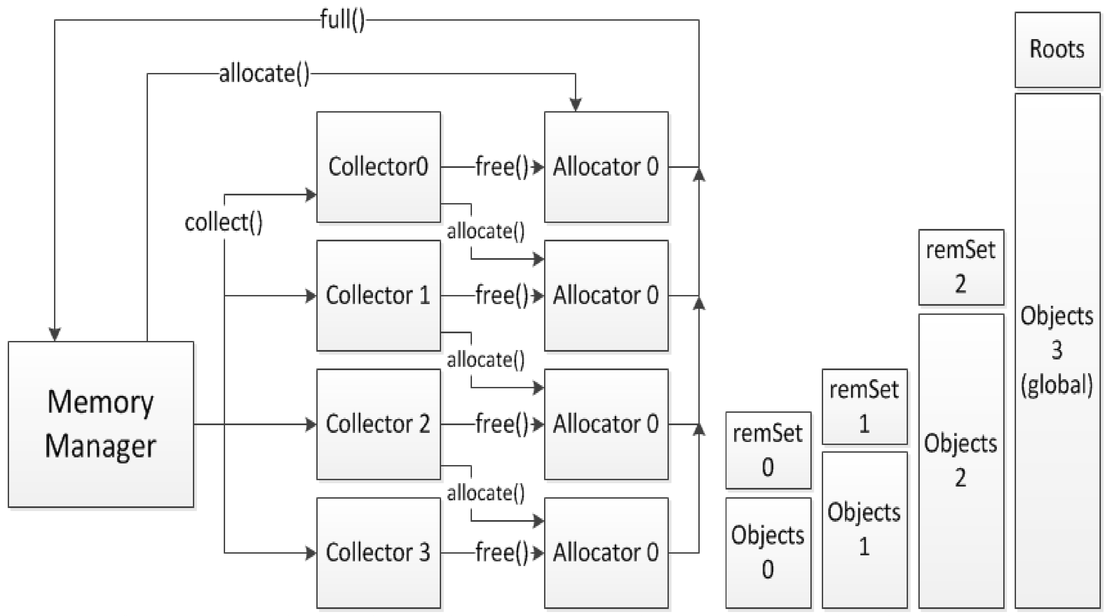

Figure 1: Object oriented structure of the n-Gen-GC implementation

- Parameters which are taken into account during generation of the trace file include: min/max object size, reference/allocation operation, min/max reference count, root set size, thread number.
- Trace files implement a specified structure, which can be used as virtual machine input for testing purposes.
- Specific application behavior can be mimiced using extracted probabilistic characteristics of the application.

n-Generational GC

- Implemented from scratch in C++.
- Implements the generational garbage collection policy.
- Allows developers to change the generation count, collection policy for different generations, generation sizes, promotion age and others.
- Currently available collection policies are Mark Sweep, Mark Compact and Copying Collection.
- Object oriented implementation allows the implementation of additional collection policies for experimentation without the overhead of an actual virtual machine.

Results*

- The trace file proides a rough estimate of application behavior as parameters used for evaluation were not taken from an actual application.
- Trade offs identified for n-generational garbage collection include that higher promotion age, higher generation count and lower heap size ration between generations (old/young) reduce time spend for each garbage collection, but increase the GC count.

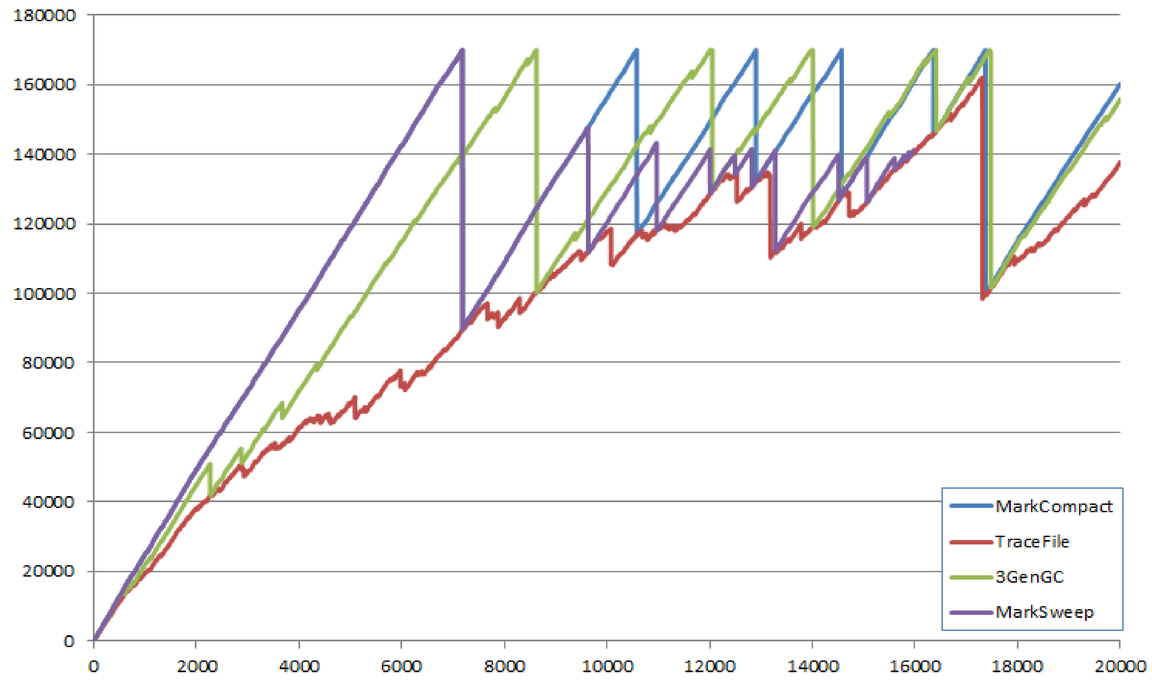

Figure 2: Allocated space comparison between traditional policies and 3-gen-GC

** Results unpublished at time of poster submission*