Back to Basics: Achieving High Performance Garbage Collectors by Improving Reference Counting

Rifat Shahriyar
Australian National University
rifat.shahriyar@anu.edu.au

Reference counting is one of the fundamental approaches to garbage collection. It is important but overlooked and not studied extensively compared to the tracing collectors. In fact we find out that the state of the art reference counting is totally outperformed by the simplest tracing collector mark-sweep (28% slower in total-time and 68% slower in collection-time). That is why, despite having some interesting characteristics reference counting is not widely used compared to the tracing collectors. It has been observed that some garbage collectors are in fact some form of hybrid of tracing and reference counting. So reference counting has definite impact on other garbage collectors. We try to understand reference counting comprehensively and find out its major design points - a) storing the reference count, b) processing increment and decrement of the reference count, and c) collecting cyclic objects. We study its intrinsic properties on real benchmarks and also perform a detail quantitative analysis.

Reference counting needs space to store the reference count for each object. Usually one extra word is used in the object header to store the count. But in our analysis we find out that most of the objects (95% or more) reference count is between 1 to 10. Typical object header contains few unused bits. So we can store the count in those unused bits instead of using the extra word. In that case some objects count cannot be stored in the limited bits and separate strategies will be required to handle those objects. We proposed different strategies for limited bit reference counting and thoroughly evaluated them. The best strategy can improve total-time by 4% and collection-time by 12% on average compare to the state of the art reference counting over the 19 benchmarks we used from DaCapo and SPEC suites.

The processing of increment and decrement of the reference count are performed using two separate queues, *mod-buffer* (for increments) and *dec-buffer* (for decrements). The state of the art reference counting is a deferred reference counting and uses coalescing write barrier where a series of increments and decrements are coalesced into one. A newly allocated object is initialized with a count of one, an entry to the *dec-buffer* and eagerly pushed to the *mod-buffer*. In our analysis we found out that majority of the increments and decrements are performed because of the newly allocated objects that are eagerly pushed to the *mod-buffer*. So we proposed a strategy where the new allocated objects are initially overlooked and pushed into the *mod-buffer* lazily and no increment and decrement are performed for those newly allocated objects those did not survive a single collection. So we are able to reduce huge amount of unnecessary processing for those objects whose increment and decrement will eventually cancel each other out and the object become garbage. This idea is guided by the result of the analysis we performed. This novel strategy can improve total-time by 21% and collection-time by 63% on average compare to state of the art reference counting.

By combining two above-mentioned ideas we can achieve improvement of total-time by 23% and collection-time by 67% over the state of the art reference counting. If we compare with mark sweep then it is only 7% slower in total-time and 5% slower in collection-time. We are able to speed up reference counting whose performance is now competitive to the tracing collectors. We believe by making reference counting competitive to tracing, performance of a large class of garbage collectors can be improved.