#ASLI ENGINEERING

Why Caching cannot improve Mark and Sweep?

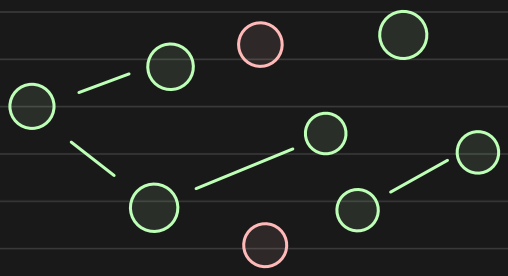
SWIPE

BY

ARPIT BHAYANI

Why caching cannot improve Mark and Sweep GC?

Mark and Sweep Garbage Collection



Mark: Start from root nodes and continue to mark the reachable objects
whatever's left is dead/garbage

Sweep: iterate to all objects in the heap and clean what's garbage

Garbage Collection needs to be extremely fast

We do not want the CPU cycles to be used to collect garbage.
We would want it to be constructively used in running user program

Given that Mark-and-Sweep is a simple DFS,
how can we improve its performance?

- smaller GC pauses
- more CPU cycles for user program

The first optimization technique we could think of is

Caching — can it work?

What is a cache?

Cache is anything that stores data, so that the future requests are served faster.

The data we "cache" can be

- result from the previous computation
- copy of the data from a slower storage

Where cache improves performance?

Cache can improve performance when the application exhibits

- temporal locality
- spatial locality

Temporal locality

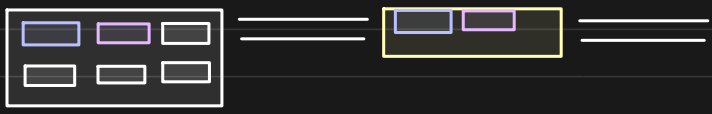
A recently accessed memory location is more likely to be accessed again.

Spatial locality

If a location is recently accessed, the locations adjecent to it will be accessed soon

Hardware / Cache Prefetching

To leverage temporal and spatial locality, hardware can pre-fetch some data likely to be accessed into the cache and thus boosting the overall performance of the system.



Data prefetching is done in two ways

1. intelligent hardware - Intel / AMD does this by analyzing the access
2. explicit `prefetch` instruction - your program can request the underlying hardware to cache

So, can we leverage caching to speedup the Garbage Collection process ?

The answer is **No**

Why garbage collector cannot leverage caching to improve performance?

To leverage caching, the use-case should exploit

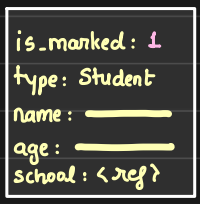
temporal locality or spatial locality

Temporal locality of a Mark-and-Sweep GC is poor

An object is accessed once and the "mark" bit in its header is read and set.

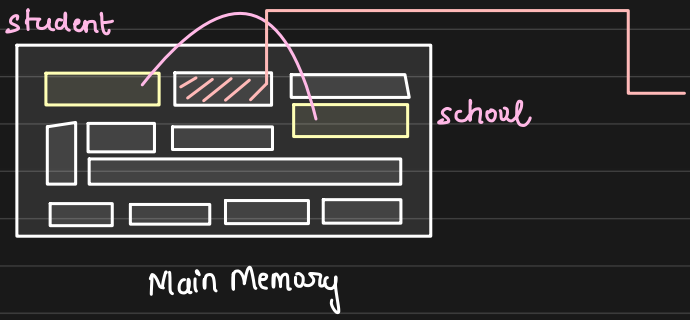
The object is never accessed again

So, caching won't help.



Spatial locality of a Mark-and-Sweep GC is poor

Mark-and-Sweep uses DFS to go through the "LIVE" objects which means you will not be accessing physically adjecent objects



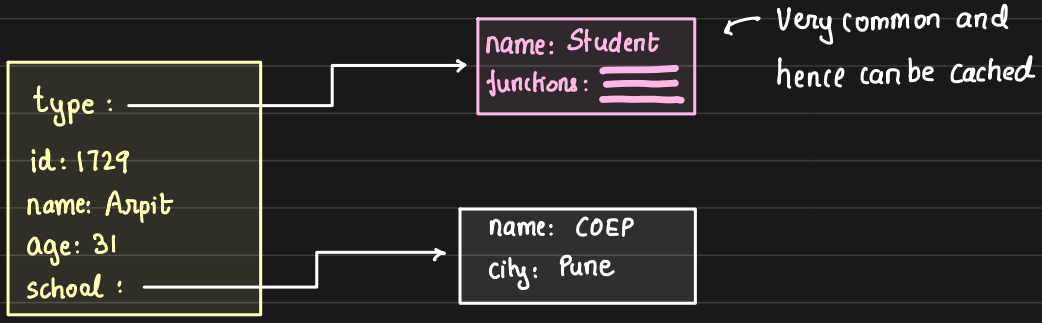
Minor speed up is possible though

We can cache the static low cardinality objects referenced

eg: a language supports multiple types

and for it to understand the type of an object

it would store info about it (which could be another object)



* This is a very crude example. In reality the type objects like Integer, String, Float, List, Set are cached.

Type objects tell the runtime engine what is the type of the object.