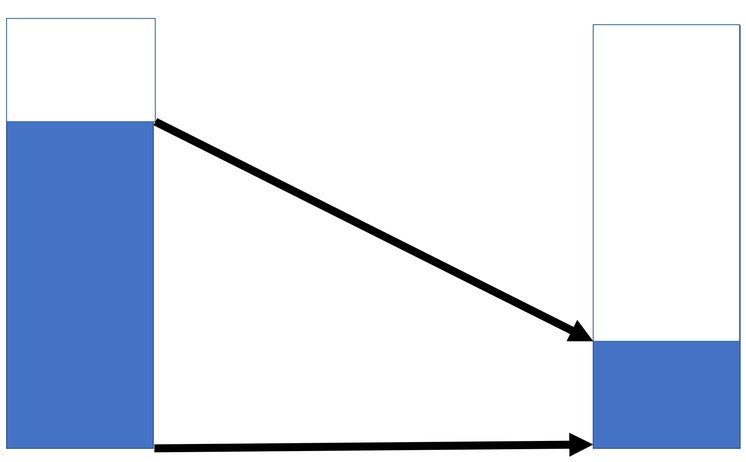
Reordering Objects During Garbage Collection

Samuel Kelley, Kenneth B. Kent, Gerhard Dueck
University of New Brunswick, IBM Canada
Faculty of Computer Science
Samuel.Kelley@unb.ca

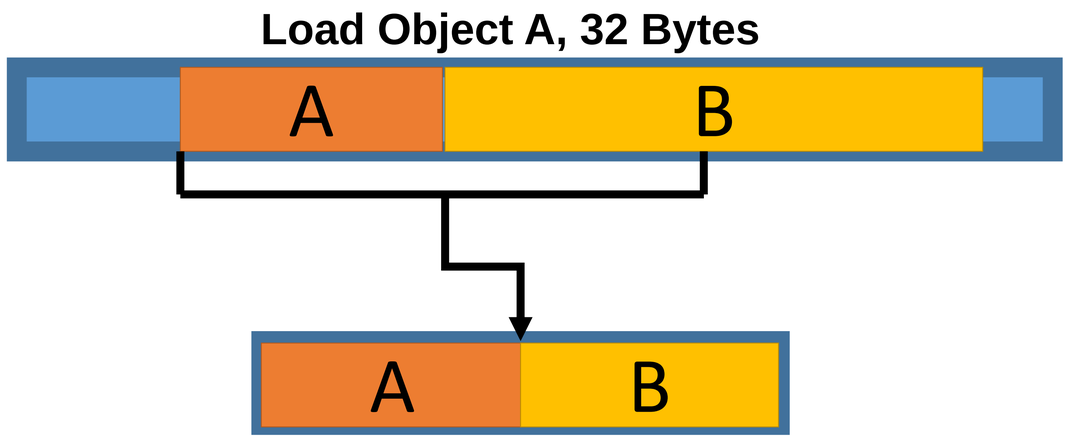
Copying Garbage Collection

One of the basic garbage collection techniques is when all the live objects in a section of memory (Fromspace) are copied to another, fresh section of memory (Tospace).



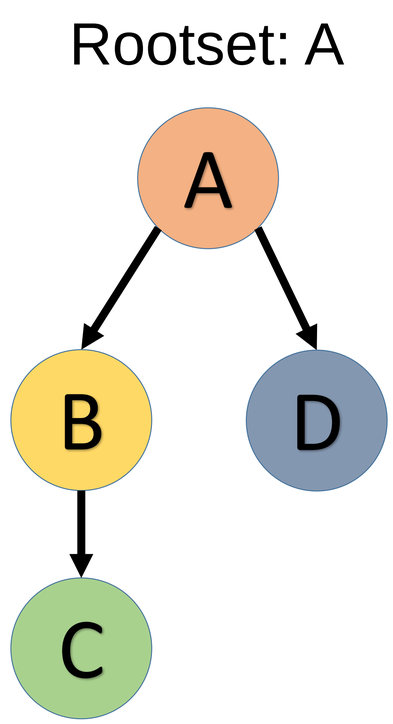
Cache Hardware

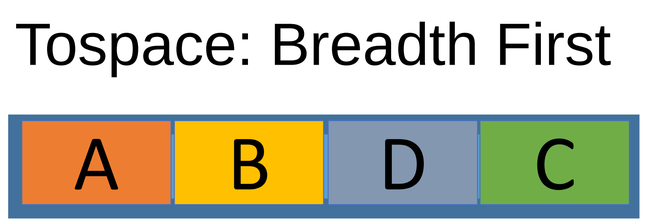
During program operation, any required data that is not found in the cache is loaded from main memory to the cache before use. This usually has the side effect of loading adjacent data as well.



Copy Collection Order

Live objects can be treated as a tree, with objects known to be alive at the roots (rootset). By traversing the tree in different ways, the order of objects in the Tospace changes.





Optimizing Order

Combining these ideas together, our research is to choose a copy collection order that maximizes the amount of beneficial cache side effects during loads. For object B in the above tree, is it more likely that objects B and D will be used together often because they are children of A, or that objects B and C will be used together often because they are parent and child?

Our current approach is to shift the problem to an object's class. If we can run a program and gather data about how popular the children of a class are in general, we can use that as a heuristic to change how objects of that class are treated during garbage collection.

We are implementing this approach in our garbage collection simulator, to avoid implementation challenges in the real JVM. The results will show whether an implementation in the JVM is likely to show an improvement in execution time.