CMPSCI 630 Systems **Fall 2014**

Lecture 3

Lecturer: Emery Berger *Scribes: Shaylyn Adams, Nick Braga*

3.1 Runtime Systems

Runtime system is generic term that refers to the "grab bag" of support features for the programming language. Examples include:

- Garbage collection
- Memory management (malloc, free, realloc, etc.)
- Libraries (C library, libc.so, libstdc++.so, etc.)
- Threads

3.2 Garbage Collection

The paper examined for today's class, "Garbage collection in an uncooperative environment" (Boehm et al., 1991), set out to make a stand-alone/external garbage collector that did not need to cooperate with the compiler/interpreter.

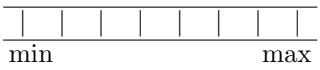
Garbage collectors can be broken down into two different categories; **precise** and **conservative**.

Precise This indicates a GC environment where all pointers are known. Real pointers ∩ discovered pointers = real pointers.

```
class foo{
int x; ------->NOT a pointer
int y; ------->NOT a pointer
char s; ------->NOT a pointer
String st; ------->YES a pointer to another object! (keeps track of this one)
}
```

Writing efficient, concurrent GC is difficult to do with stack maps which is where conservative GC comes in.

Conservative As expected, this does not keep track of pointers in the same way as precise GC. Here, we have to determine if something is a pointer. It is considered to be 'conservative' because memory will never be reclaimed too soon.



To discover the pointers, conservative GC will look through the allocated memory block, which has a minimum and maximum address range. If the object is not found within this range, then it cannot be a pointer. If it is within the range, a second check must occur as an attempt to catch any integers that look like an address. This check is for alignment, since every pointer has to be a multiple of 4 or 8, i.e. %4=0 or %8=0 (otherwise it is not a pointer). Thus, if the address is in range and it is also correctly aligned, the GC must treat it as a pointer.

However, it is still possible for a supposed pointer to actually be an integer that is in the range and divisible by 4 or 8. Generally, if a chain of pointers is followed and goes off into nowhere, it is safe to assume that it is not a pointer and it can be put on the blacklist in which no memory will ever be allocated from again. (JavaScript on Internet Explorer uses conservative garbage collection to this day.)

3.2.1 Mark-Sweep Garbage Collection

Mark-sweep garbage collection is the GC algorithm that recursively marks all reachable nodes black, and sweeps for white (unblackened) nodes that will be reclaimed (since anything unreachable needs to be reclaimed). After, it will color reachable nodes 'white' and go through the process again. Variables that indicate color correspondence are maintained.

Aside: The BiBop(= big bag of pointers) allocator achieves a lazy sweep process by keeping things in many big chunks with a bit map to indicate what is in use/allocated.

3.2.2 Stop the World Garbage Collection

The 'original' type of garbage collection is known as **Stop-the-world**, where the program would sit and stop, or *pause*, while garbage collection was performed. This pause will obviously take longer depending on the size of the heap. It is a linear process and there is no way around it. Hence, garbage collection received a bad reputation as being incredibly slow. There are other forms of garbage collection such as concurrent, parallel, and incremental, but regardless the garbage collector has to check all nodes/pointers or use a heuristic that keeps track of pointer sections. The type of garbage collector to strive for is one that is fast and happens rarely.

3.2.3 Garbage Collection Properties

There are 2 properties to be aware of when constructing a garbage collector and it is important to reach a balance between the two of them.

1) Soundness = no reachable memory is ever reclaimed. If you never ran a garbage collector you would achieve ultimate soundness.

2) Completeness = all unreachable is eventually reclaimed.

3.2.4 Problems with Garbage Collection

The biggest problem with conservative garbage collection and using mark-sweep is that it can result in a porous (fragmented) memory space, requiring:

1) More space (which in turn impacts the cache)
2) Underlying architecture optimizations

Locality the property that defines objects' proximity to one another in the memory space.

There are two types of locality:

1) Spacial locality: Similar items should be stored in the closest proximity feasible. For example, if we use byte 1, it is likely that we'll also use byte 2 in practice)
2) Temporal locality: items should be used in a close time range, before objects are then changed again

The GC cannot choose specific objects to keep in RAM, and therefore must sends 'chunks' of data to the cache. Fetching memory is expensive and can incur a penalty of around 100 cycles.
One technique to this end could be to move pointers (compaction). The problem with compaction is that it is possible to overwrite integers by doing so. Since there is no cooperation with the programming language, conservative garbage collection cannot move objects easily. Conservative GC will only reclaim pointers, but might accidentally overwrite in this way. It can find integers that look that addresses and then would end up trashing real memory.

Aside:

> "All problems in computer science can be solved by another level of indirection" -David Wheeler
> Corollary: "...except for the problem of too many layers of indirection." -Kevlin Henney

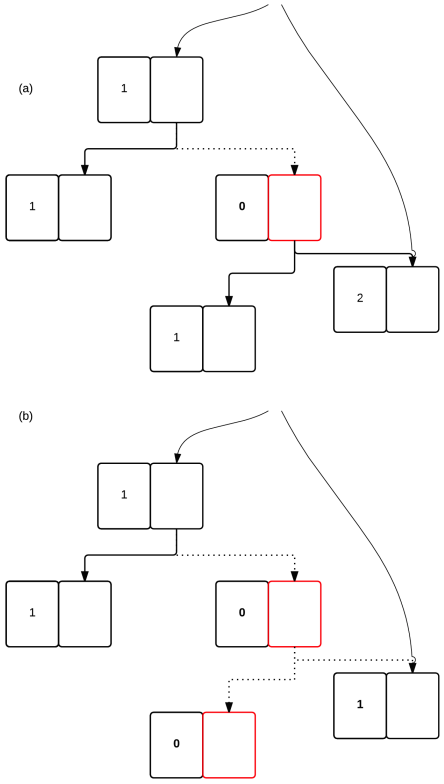
This problem of moving pointers can be solved with **handles**. Handles make it easier to reallocate memory; this is because only the 'handle table' points to actual objects. It is theoretically a pretty good solution but can be expensive. The number of operations is multiplied whenever an object is dereferenced, resulting in poor performance in practice. If you can allocate relatively small amounts of memory, GC overhead can be made small. This is not possible with handles. Handles tend to interact poorly with **prefetchers** since handles are unpredictable and the next move cannot be predicted at the architecture level even with smart prefectchers. (One example of a prefetcher is a stride prefetcher, which learns loops by observing the memory stream).

In conclusion, while conservative GC cannot overcome fragmentation, using handles has other performance overheads above conservative GC.

3.2.5 Garbage Collection Evolution Timeline

Everyone is always trying to 'reinvent the wheel'.

INTERPRETER: Perl, PHP, Ruby, Python, JavaScript, etc., all easy to write——->TERRIBLE GC: reference counting——->MARK-SWEEP: better GC——->JUST IN TIME COMPILER: Interpreter is too slow——->"FANCY GC": Typically doesn't work



a): In this diagram, each memory-allocated unit contains both a data bucket and a reference counting bucket. The garbage collector has determined an unreachable node (marked in red) and has reclaimed the memory space, also removing its pointer(s) and decrementing its reference count. (b): The GC then recursively moves through the reclaimed node's pointer from (a), and reclaims if unreachable, etc.)

Reference counting is a type of GC where all incoming pointers are kept track of. So, on every single pointer update the RC destination must be increased and the RC origin must be decreased. If RC count==0, recursively decrement references of all things after and then reclaim object. However, with circular objects like doubly linked lists, reference counting will not work.(Object(ionable)ive C uses explicit reference counting where the programmer must do it manually.)

Aside: Vaporware refers to software that is announced but never materializes.

3.2.6 Running out of Memory and Swapping Tangent

OOM-Killer, or out of memory killer, refers to the Linux nuance where when memory capacity is reached, the kernel will pick an unsuspecting victim to "kill" based on a heuristic. Ironically, Linux used to always effectively commit suicide by killing process 0 aka the kernel itself.
-Disk swapping via LRU frees up physical memory.
-Paging/thrashing refers to the process of constantly moving memory back and forth from disk when RAM is full.
-There is no swap space in iOS.
-Linux used to overcommit memory even when there was not enough swap room.