COMPSCI 590S Systems for Data Science **Fall 2016**

Lecture 5: February 5, 2018

Lecturer: Emery Berger *Scribe(s): Walter Brown*

5.1 Garbage Collection (GC)

5.1.1 Reference Counting (RC)

RC is often used, despite its issues with cycles. Recall that there can be a cycle of pointers in memory that cannot be reached but all have a count of 1. Graphical User Interfaces (GUIs) are commonly used to to demonstrate RC because of this. GUIs generally have a tree structure, which is ideal for RC.

In computer science, there are many other instances of special graphs that break things. For example, C++'s multiple inheritance. If some class inherits from two other classes and both parent classes have a method of the same name, which version of the method should the child class inherit?

RC can be implicit or explicit.

5.1.1.1 Explicit

Swift uses explicit RC.

5.1.1.2 Implicit

Objective C uses Automatic Reference Counting (ARC). Calls to increment and decrement the reference counters are done for you automatically.

5.1.2 Marking in Mark-Sweep

Recall Mark-Sweep, the process of doing a graph exploration to mark all values that are reachable and then sweeping away items that are not. During marking, you must decide if you want your graph exploration to interpret a given variable as a pointer to another object. There are different schemes on how to do this.

5.1.2.1 Precise GC

The program cooperates with GC and tells it which variables are pointers. On the programmer's side, they indicate that a variable is a pointer through variable typing.

To do Precise GC, you must remember a "map" of each class that indicates where the pointers will be. Every object needs an object header, which indicates what class it is so you know which map to use.

Can you steak out areas and dedicate ranges to different types during compilation to get around using object headers? Maybe now, but not back in the day. Too memory intensive!

However, you can have a large chunk of memory with many instances of one class inside that share an object header! When you want to make a new instance of object X, you check if any chunks are devoted to object X and have space to spare. If not, make a new chunk. Rounding can be used to find the object header of a given address. Furthermore, you can store this object header metadata far away if you are worried that it will be overwritten. This is called Big Bag Of Pages (BiBOP).

Note that it is possible for a buffer overflow to stomp out an object's header!

Managed languages are a special class of languages where this works well. They tend to have GC, make sure that objects don't change their type, and do not let you access arrays outside their bounds.

What if you have an unmanaged language like C?

5.1.2.2 Conservative GC

The Hans Boehm Collector was created so you could use GC without cooperation from the programmer. Rather than using a label to indicate that something is a pointer, it runs a "Duck Test" on variables. If the variable contains a value that would point to an address within a valid range if interpreted as an address, the collector treats it as a pointer.

Boehm was actually working on Russell (never heard of it? You're not alone!). However, he planned to have Russell programs get converted to C so he could leverage C's awesome compiler with all of its optimizations.

The Hans Boehm Collector still has the potential to break things in general C programs because you can do whatever you want with pointers.

1. XOR encoding: Say you have a doubly-linked list. Rather than storing two pointers in it, you can store the next pointers and have the previous pointer be the XOR of the node's address and the next pointer. If this is the case, you can calculate it rather than storing it! This would not be detected by the conservative collector.

2. Object Alignment: If you know your addresses will only use 6 out of 8 bits, you can jam 2 more arbitrary bits of data in there! This defeats the conservative collector.

3. You can write stuff to disk and temporarily remove it from memory. Conservative collection might then remove objects that only the stuff on the disk was pointing to. If you move the disk objects back to memory, they'll have pointers to collected objects.

5.1.3 Paging and GC

When a program needs too much memory, the machine will kick a "page" of memory to a region of disk called the swap partition. This runs on a Least Recently Used policy.

GC should be oblivious to paging. However, if too many pointers are followed during marking, you'll be moving a lot of pages in and out of memory, which is slow. In addition to this, the pages left in memory by GC won't be the ones the program previously wanted.

Try not to have programs with heaps larger than physical memory. :)

5.2 Security

People used to not be worried about security.

Consider the concept of a threaded free list. All of your freed unreclaimed objects can be connected by pointers, forming a list. However, this is dangerous if someone overwrites these pointers.

5.2.1 The Morris Worm

As an experiment, Robert Morris created the Morris Worm in 1988.

The Morris Worm took advantage of Unix vulnerabilities to spread itself to other machines on the network. Like Mark in Mark-Sweep, it wasn't interested in infecting computers that were already infected. However, it had a 1/7 chance of spreading to infected machines, as to prevent machines from inoculating themselves by pretending to be infected. The Morris Worm did not moderate its propagation rate, so it would consume all of the computing resources of infected machines.

People were popping out of their cubicals like mercats, asking if their coworkers' machines were down too.

Although nothing was written to disk, restarting the machine would just result in other machines on the network infecting it again. The Morris Worm was similar to a DDOS.

Computer felonies weren't a thing yet, so Robert was convicted of computer fraud and abuse. He paid damages, but did not serve jail time. Now he works at MIT.

Clifford Stoll wrote a book about this.

5.2.2 Internet Explorer

Microsoft's Internet Explorer still lets you launch any program on your computer. This has not been fixed yet.

5.3 IBM System / 360

The 360 featured many innovations:

1. Separation between Instruction Set Architecture and Microarchitecture
2. Hardware-based Virtual Machine (VM)
3. Caches

Note that IBM was responsible for FORTRAN too. They ruled the computer space!

5.3.1 VM

Every VM thinks it owns the entire machine.

Mendel Rosenblum and Carl Eschenbach made this feasible by doing privileged code in software. This would be really slow, but they would identify large regions of unpreveleged instructions that could all be executed at full speed. They made VMware.

"All problems in computer science can be solved by another level of indirection." -David Wheeler

5.3.2 Caches

Early in computing, memory was not a bottleneck because everything was comically slow. However, as instructions come from memory, slow memory accesses will slow down the entire machine. Dynamic Random Access Memory (DRAM) is still slow. The fact that CPUs have become so fast and memory hasn't is called the Memory Wall.

Conventional Instruction Set Computing (CISC) is a workaround for this. CISC has ridiculously powerful instructions that can do things like solving systems of equations. This way, you can do more with less instructions getting fetched.

Another solution is caching. Modern machines will generally have around 5 caches, labeled from L0-L4. L0 is close to the CPU, small, and fast. L0 is generally made of fast Static Random Access Memory (SRAM). L4 would be farther away, large, and slow. L4 is generally made of slow DRAM. Before requesting a variable on the bus, you can check if it's in a cache. This is a big time saver.

When multiple CPUs are using the same memory, you must be careful about cache coherency. This takes some extra bits on the cache line, but helps to invalidate out-of-date values.

MESI is a basic cache coherency system. Cache lines have a state machine with four states:

- Modified : You have modified this line. All other users must invalidate their copies.
- Exclusive : You're the only one with the cache line.
- Shared : Everyone might write and read this line.
- Invalidated : The copy of the cache line you have is out of date.

Eviction is the act of removing an item from a cache.

Caching schemes can be inclusive or exclusive.

5.3.2.1 Inclusive

Data in a cache will be stored in all caches below it.

Inclusive eviction is easy. There's a copy of the evicted data in the lower caches, so the data doesn't need to be copied during eviction.

This is the approach of Intel.

Inclusive cache policies were susceptible to the recent Spectre exploit.

5.3.2.2 Exclusive

Data might appear in only a single cache.

When data is evicted, you must find a new home for it.

This is the approach of AMD. They've gotten really good at eviction in exclusive caches.

5.3.2.3 NEXT TIME IN SYSTEMS 630

Cache misses...