CMPSCI 691ST Systems	**Fall 2011**

Lecture 3

Lecturer: Emery Berger | *Scribe: Nicolas Scarrci*

3.1 Conservative Garbage Collection

The Boehm collector is the first example of conservative garbage collection. Conservative garbage collectors are garbage collectors that work without compiler cooperation. For this reason they must discover pointers on their own. Unfortunately pointers and integers look the same from the compiler's point of view causing some integers to be treated as pointers. Conservative garbage collection also requires that the running program stop and wait for the garbage collector to finish before it may proceed. If there are multiple threads all threads are required to stop.

3.2 Precise Garbage Collectors

Garbage collectors before the Boehm collector were called precise garbage collectors. These collectors used GC-maps which assumed complete knowledge of pointers.

Garbage Collection Terminology
Changing a pointer is called mutation. The real program is called the mutator.

Garbage Collection Concerns
Errors in garbage collection can cause old programs to fail. Conservative GC errs on the side of false positives. GC errors may take a while to surface.

3.3 Other Garbage Collection Techniques

Coordination through read and write barriers (uses indirection, see Fig 1.)
Mutator: Hey collector, I'm going to follow this pointer to an object and then write new stuff into it.
Collector: Oh sure, just write that stuff to this other location instead.

3.3.1 Semi space garbage collection

Divide all memory into two halves. (flat loss of half of memory)
Start by doing bump allocation in the first half.
When this half is full trigger garbage collection.
Copy live objects over to the other half of memory and leave a reference pointer in their place.
Move the allocation pointer to the first free bit of memory in the second half of memory.
Clear the first half of memory.
Resume allocation.
(This algorithm is called a Cheney scan.)

Semi space garbage collection is a stop the world garbage collector. This scheme still falls prey to the issue of misidentifying an int as a pointer. If the number of live objects is much larger than the number of dead objects then this process can slow down. In addition this process does not yield any benefit if all of the dead objects are at the end of the memory space.

3.3.2 Emery's idea.

Run two copies of a program.
When checking a suspected pointer check both copies of memory.
If the int/pointer in question is the same in both programs it is an int.
If the int/pointer has the same base but a different offset then it is a pointer.

3.3.3 Generational garbage collection

Generational garbage collection relies on the Generational Hypothesis: most objects die young. While this hypothesis holds true for most programs it is not universal. For this reason generational garbage collection is an example of common case optimization.

Initially allocate memory in a nursery.
When garbage collection is triggered check each object in the nursery for incoming references.
If an object is referenced from outside of the nursery copy it to a space outside of the nursery.
Clear all objects remaining in the nursery.
Resume memory allocation.

Variants:
Dynamic nursery size.
Multiple generations (nursery levels.)

Profiling: Track a program to see what sorts of objects it makes. Intentionally allocate memory chunks of these sizes.

Optimization: configuration parameters (or 'knobs'.) Bad because the number of options is very large. Bad because slight code changes may require new knob settings.

3.4 Tangents

Where is GC located?
Garbage collection is located above the OS. All memory allocation is actually done in the runtime environment of the language in use.

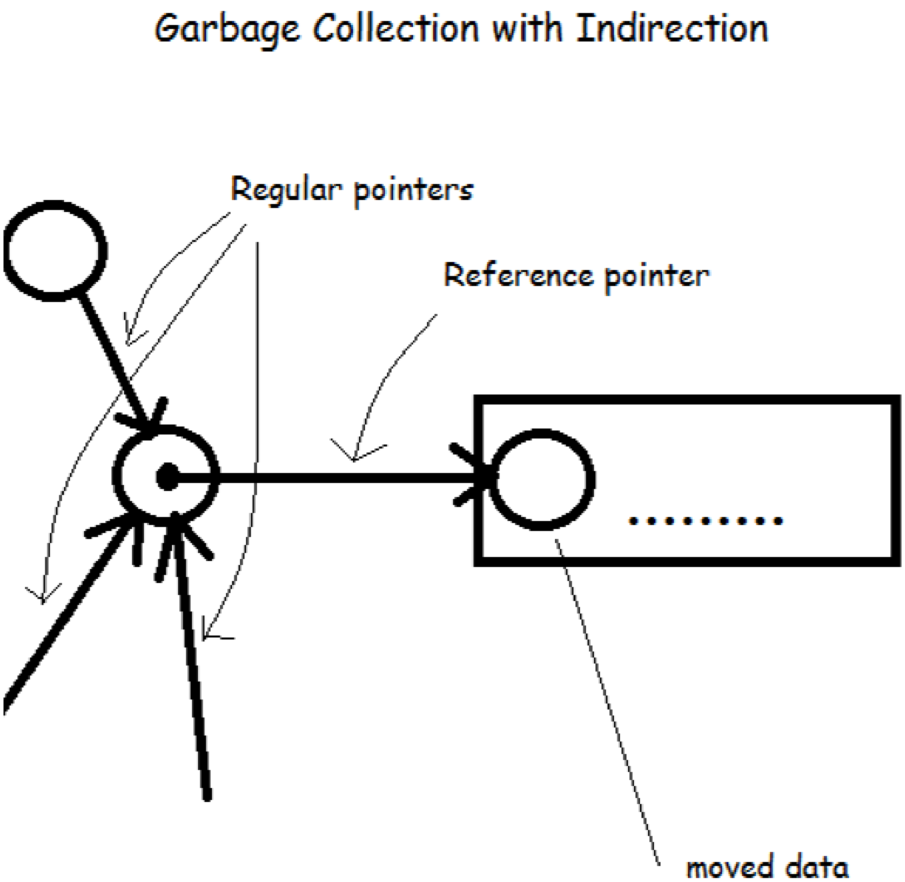
Since the initial paper Boehm has created incremental versions of his collector. The current implementation also blacklists integers that look like pointers but are later discovered to be integers.

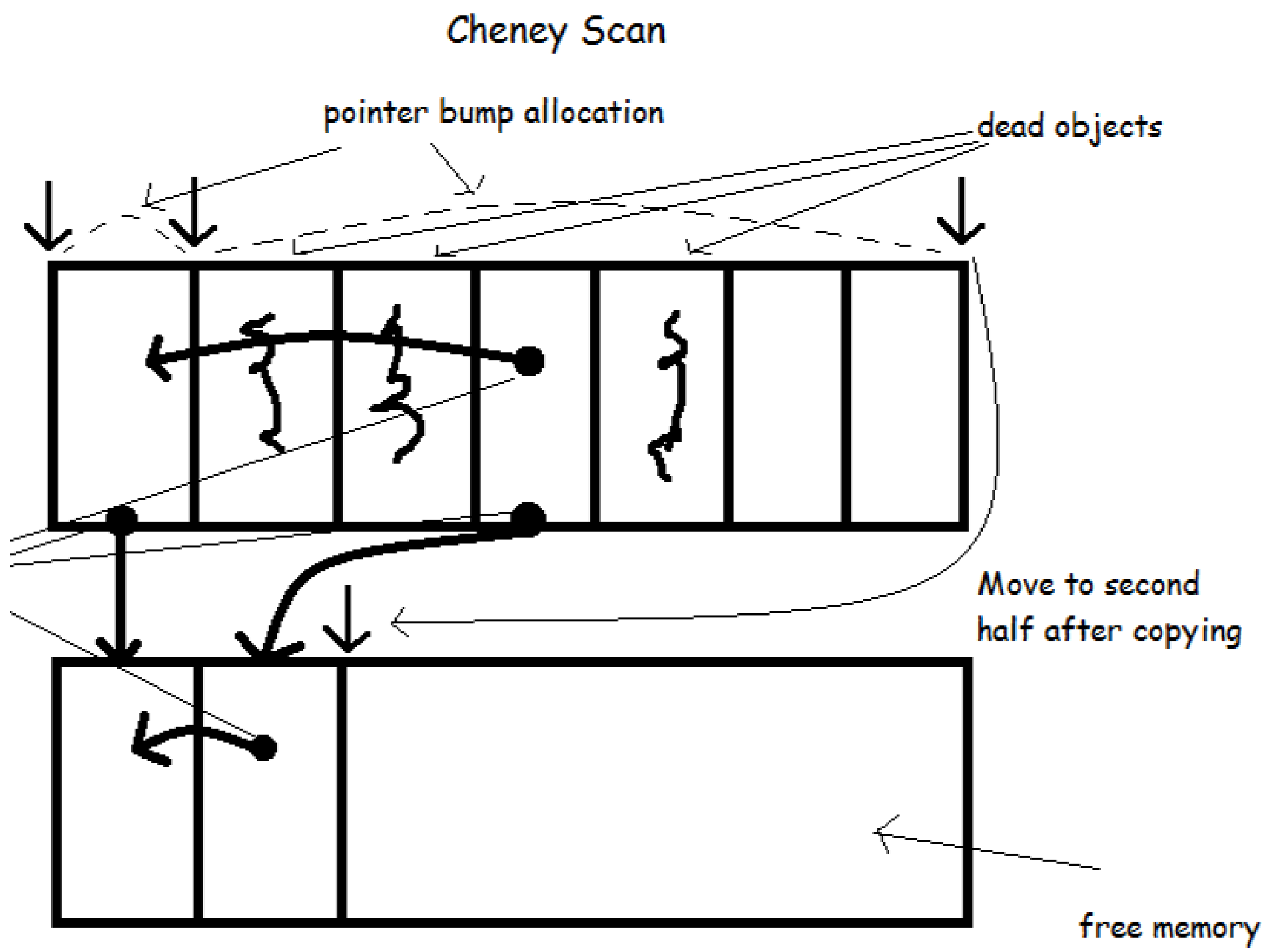
The shuttle program uses multiple sets of hardware. This is to prevent systematic hardware errors across multiple machines. A similar approach can be taken with software, this is called n-variant software. Software engineers argue that this is ineffective as a problem in the software specifications could cause a systematic error across multiple software systems. Another solution is to use checkpointing. Checkpointing is the process of periodically saving system state. If an error occurs the system can load the most recent checkpoint and retry. This is also called rollback or rety.

Cosmic rays may flip a bit. This is a soft error and may go undetected. One solution is to add a checksum bit or a parity bit. Another option is to use Error Correcting Codes (ECC). Eccs allow memory error correction. Unforutnately they are not widely used because they cause a slight performance decrease and memory with ECCs is more expensive. ECCs are in the domain of coding theory. We cannot send people to mars because cosmic radiation will kill them. If we instead send sophisticated robots to mars cosmic radiation will flip their bits.

The library problem in programming language design is the problem of having to reimplement popular libraries in the new language to make it more usable/approachable.

URLs are direct pointers, hence when a file gets moved we get a broken link. If we had a level of indirection (like that used in garbage collectors which move data) we could avoid this problem.


Garbage Collection with Indirection
Regular pointers
Reference pointer
....
moved data


Cheney Scan
pointer bump allocation
dead objects
Move to second
half after copying
free memory