A Look at Non-Intrusive Conservative Garbage Collection

Gustavo Rodriguez-Rivera
Vincent Russo

Purdue University
{grr,russo@cs.purdue.edu}

Memory Management

Explicit

- Call *free()* and *delete()* explicitly
- Error prone

Automatic

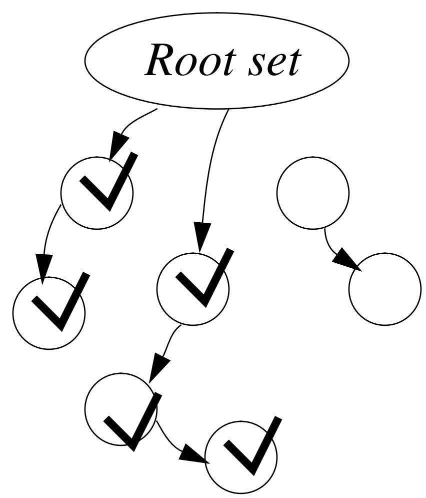
- Reference counting and Garbage Collection
- Safer
- Saves programmer's development time
  (30-40% in programs with complex
  data structures, Rovner 84)

Mark and Sweep Garbage Collection

- Recursively mark all objects reachable from root set

- Deallocate unmarked objects

- It needs to know pointer location (language cooperation)



Conservative Garbage Collection

- Used for languages that have partial information about the location of pointers

- If it looks like a pointer, it is a pointer
  1. *Value is aligned to word size,*
  2. *Points to some place inside heap boundaries, and*
  3. *Points to some previously allocated block*

- Some implementations are publically available:

  ftp:\\parcftp.xerox.com/pub/gc

- To use them applications need to be slightly modified and recompiled

  => Source code has to be available

Non Intrusive Garbage Collection

Goal:

- Use conservative garbage collection in any program, even when source code is not available

Advantages:

- Integrates garbage collection in commercial programs and libraries where source code is not available
- Convince people about the usefulness of garbage collection
- Non-intrusive memory debugging (better than Purify)

Disadvantages:

- Need a "*more conservative*" garbage collector

Our Garbage Collector Library

- Incremental, Generational, Conservative GC

- Uses run-time linker to replace malloc library with
  our GC library

```
csh> (setenv LD_PRELOAD libGC.so; program)
```

- Programs need to be dynamically linked
  (most of them are in modern OS)

Problems

- Finding the root set for any program is difficult:

  Different memory mappings for shared libraries, thread stacks, memory mapped files etc.

  Solution:

  root set = (all memory mappings != heap)

- Root set too large

  Solution: Only pages that have been dirty since the time the heap was initialized are considered part of the root set

  => Unused stack pages are not included in root set

  => Neither *Text* + some *data*

More Problems

- Too many false pointers:
  - Solution:
    - + Put heap high in memory
      - heap ~= 0xeb000000

- Slow pointer detection:
  - Solution:
    - + Use an independent memory map (mmap)
      - A continuous heap makes pointer detection faster

Applications: Garbage Collection

- Run garbage collection in programs without recompiling

- Used with matlab, netscape, frame maker, ghostview, Xserver...

- An independent thread runs the garbage collector

- Our allocator and garbage collector are comparable in speed with Boehm-Demers-Weiser garbage collector. Faster in multi-processor machines

Applications: Debugging Tool

- Not as intrusive as Purify

- **`heapinfo`**: Client program that interacts with GC library

- Allocation statistics

```
> heapinfo pid statistics
```

- Leak detector

```
> heapinfo program-pid reset
```

... let program run ...

```
> heapinfo program-pid leaks
```

```
lorenzo 30 % setenv GC_EXPLICIT_FREE_OFF;
lorenzo 31% setenv LD_PRELOAD
  "/u/u238/grr/SpringGC.August.28.95/build/Solaris/build/libGC.so /usr/lib/libthread.so"
lorenzo 32% matlab
lorenzo 79 % heapinfo 29851
```

Allocator Statistics By BlockSize

Unit Size (Bytes)	Opaque (Yes/No)	Total Allocs (#)	Recent Allocs (#)	Mem in Pages (Bytes)	Mem in Blocks (Bytes)	Mem in Objects (Bytes)	Fragmentation (%)
2048	No	6378	6378	3915776	3280896	1903088	51
1024	No	4930	4930	983040	734208	433240	55
512	No	10276	10276	1204224	1010688	760988	36
256	No	8394	8394	585728	389888	312068	46
128	No	8859	8859	397312	250368	219852	44
64	No	6926	6926	266240	217152	134896	49
32	No	9954	9954	151552	85536	71452	52
942080	No	3	3	1884160	942080	940852	50
315392	No	4	4	946176	630784	629040	33
241664	No	3	3	483328	241664	238328	50
65536	No	1	1	65536	0	0	100
53248	No	1	1	53248	53248	50288	5
40960	No	10	10	368640	368640	350352	4
20480	No	3	3	61440	61440	57544	6
16384	No	6	6	81920	16384	15440	81
12288	No	99	99	479232	184320	129636	72
8192	No	165	165	999424	212992	158608	84
4096	No	809	809	1048576	839680	659984	37
		56825	56825	14483456	9519968	7065656	51

```
               Garbage Collector Statistics
               ----------------------------

NumberOfGCs:                                         28        <---------
TotalMemoryAllocated                                 14483456  <---------
TotalMemoryCollectedInAllGCs:                        30024672
TotalMemoryCollectedInLastGC:                        3314944
TotalPossibleGarbageSinceLastGC:                     2582656
MaxPossibleGarbageBeforeNextGC:                      694832
TotalMemoryInMarkedAndAllocatedBlocksAfterLastGC:    5936352
MaxMemoryCollectedInLastGCs:                         3633504
MaxMemoryInMarkedAndAllocatedBlocksAfterAGC:         6948320
TotalRoots:                                          6844416
TotalDirtyRoots:                                     106496
TotalEverDirtyRoots:                                 290816
TotalBytesScanedInLastGC:                            2572904
MaxTotalBytesScanedInLastGCs:                        2671616
MaxPotentialFalsePointers:                           40        <---------
PotentialFalsePointersInLastGC:                      38
```

```
lorenzo 38 % setenv LD_PRELOAD
        "/u/u238/grr/SpringGC.August.28.95/build/Solaris/build/libGC.so /usr/lib/libthread.so"
lorenzo  39; Mosaic
lorenzo 76 % heapinfo 1281 -leaks

Total Leaks = 312 Bytes

0: GetCommand+52
1: ProcessMailcapEntry+1116
2: ProcessMailcapFile+172

     ( 32%) LeakedMemory=100     #LeakedMallocs=3      AvgMalloc=33

0: HTSACopy+116
1: HTParse+2972
2: mo_url_canonicalize+36
   "mo-www.c", line 657

     ( 23%) LeakedMemory=72      #LeakedMallocs=1      AvgMalloc=72

0: mo_setup_default_hotlist+128
   "hotlist.c", line 1116
1: mo_do_gui+2936
   "gui.c", line 2259
2: main+172
   "main.c", line 211

     ( 14%) LeakedMemory=44      #LeakedMallocs=1      AvgMalloc=44

0: ProcessMailcapEntry+1048
1: ProcessMailcapFile+172
2: HTLoadTypesConfigFile+16
```

(11%) LeakedMemory=36 #LeakedMallocs=3 AvgMalloc=12

0: XtMalloc+28
1: XmTextFieldGetString+172
"TextF.c", line 10329
2: TextFieldGetValue+160
"TextF.c", line 10118

(8%) LeakedMemory=28 #LeakedMallocs=1 AvgMalloc=28

0: strdup+12
1: mo_get_font_size_from_res+20
"gui.c", line 1483
2: mo_open_window_internal+768
"gui.c", line 1592

(5%) LeakedMemory=16 #LeakedMallocs=1 AvgMalloc=16

0: XtMalloc+28
1: XmeStringGetComponent+3496
"XmString.c", line 11033
2: _XmStringSingleSegment+248
"XmString.c", line 9594

(3%) LeakedMemory=12 #LeakedMallocs=1 AvgMalloc=12

0: TriggerMarkChanges+496
1: FormatChunk+100
2: FormatAll+1288

(1%) LeakedMemory=4 #LeakedMallocs=1 AvgMalloc=4

Applications: Debugging Tool (cont.)

- Boundary checks (Memory Smashing)
  - + Insert magic number before and after allocated block
  - + Check magic numbers when object is freed or garbage collected

- Initialization checks
  - + Blocks are initialized with special magic numbers
    0x10badbad: New blocks
    0x20badbad: Freed blocks
    0x30badbad: Garbage collected blocks

  - + Makes it easier to catch bugs related to uninitialized memory

Current Status

- Implemented as a shared library for Solaris 2.3
- Joint project with Sun Microsystems
- Used internally at Sun Microsystems as a debugging tool for the DOE project (Distributed Objects Everywhere)

Future Work

- Optimizations
- Port to other platforms (x86)
- Make it publicly available