Reducing memory footprint of large unfamiliar codebase

Asked 16 years, 3 months ago Modified 15 years, 3 months ago

Viewed 1k times

4

Suppose you have a fairly large (~2.2 MLOC), fairly old (started more than 10 years ago) Windows desktop application in C/C++. About 10% of modules are external and don't have sources, only debug symbols.

How would you go about reducing application's memory footprint in half? At least, what would you do to find out where memory is consumed?

c++ winapi optimization visual-c++ memory-management

Share

Improve this question

Follow

edited Oct 17, 2008 at 15:53

asked Sep 23, 2008 at 18:28

Constantin

28.1k ● 10 ● 63 ● 79

7 Answers

Sorted by: Highest score (default)

7

Override malloc()/free() and new()/delete() with wrappers that keep track of how big the allocations are and (by recording the callstack and later resolving it against the symbol table) where they are made from. On shutdown, have your wrapper display any memory still allocated.

This should enable you both to work out where the largest allocations are and to catch any leaks.

Share Improve this answer Follow

answered Sep 23, 2008 at 18:54
moonshadow
88.9k ● 7 ● 86 ● 121

Can you elaborate on best way to override allocation functions application-wide, preferably with limited source code changes? Also, there's plenty of COM/BSTR processing; i believe their allocators can't be excluded from analysis. – Constantin Sep 23, 2008 at 19:39

3

this is description/skeleton of memory tracing application I used to reduce memory consumption of our game by 20%. It helped me to track many allocations done by external modules.

Share Improve this answer Follow

answered Sep 23, 2008 at 18:59
yrp
4,575 ● 2 ● 26 ● 10

2

It's not an easy task. Begin by chasing down any memory leaks you cand find (a good tool would be Rational Purify). Skim the source code and try to optimize data structures and/or algorithms.
Sorry if this may sound pessimistic, but cutting down memory usage by 50% doesn't sound realistic.

Share Improve this answer Follow

answered Sep 23, 2008 at 18:51

radu_c
147 ● 1 ● 4

1

There is a chance is you can find some significant inefficiencies very fast. First you should check what is the memory used for. A tool which I have found very handy for this is Memory Validator

Once you have this "memory usage map", you can check for Low Hanging Fruit. Are there any data structures consuming a lot of memory which could be represented in a more compact form? This is often possible, esp. when the data access is well encapsulated and when you have a spare CPU power you can dedicate to compressing / decompressing them on each access.

Share Improve this answer Follow

edited Oct 6, 2008 at 21:18

answered Sep 23, 2008 at 19:02

Suma
34.3k ● 18 ● 129 ● 199

It depends if he means "created 10 years ago, unchanged since", or "has been organically growing and accumulating cruft for the last 10 years." The second category of apps can definitely be memory hogs. – Brian Sep 23, 2008 at 19:08

It's been growing. And accumulating. The requirement to reduce resource consumption is passed down from above, as always. – Constantin Sep 23, 2008 at 19:31

0

I don't think your question is well posed.

The size of source code is not directly related to the memory footprint. Sure, the compiled code will occupy some memory but the application might will have memory requirements on it's own. Both static (the variables declared in the code) and dynamic (the object the application creates).

I would suggest you to profile program execution and study the code carefully.

Share Improve this answer Follow

answered Sep 23, 2008 at 18:38

Remo.D
16.5k ● 6 ● 49 ● 79

Size of source code hints at project's overall complexity. How much time do you think it will take to "carefully study" 400 KLOC? :) – Constantin Sep 23, 2008 at 19:01

it should be easy. 400k isn't what I'd call large :) as you only need to check heap allocations, find all the malloc and new calls and you're almost there. – gbjbaanb Sep 23, 2008 at 19:06

gbjbaanb, i wish i could show you *just* module dependency graph :) and regarding "find all the malloc and new", there's also at least SysAllocString (often buried inside CComBSTRs) – Constantin Sep 23, 2008 at 19:45

0

First places to start for me would be:

Does the application do a lot of preallocation memory to be used later? Does this memory often sit around unused, never handed out? Consider switching to newing/deleting (or better use a smart_ptr) as needed.

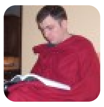
Does the code use a static array such as

```
Object arrayOfObjs[MAX_THAT_WILL_EVER_BE_USED];
```

and hand out objs in this array? If so, consider manually managing this memory.

Share Improve this answer Follow

answered Sep 23, 2008 at 19:00



Doug T.

65.5k ● 28 ● 141 ● 205

0

One of the tools for memory usage analysis is LeakDiag, available for free download from Microsoft. It apparently allows to hook all user-mode allocators down to VirtualAlloc and to dump process allocation snapshots to XML at any time. These snapshots then can be used to determine which call stacks allocate most memory and which call stacks are leaking. It lacks pretty frontend for snapshot analysis (unless you can get LDParser/LDGrapher via Microsoft Premier Support), but all the data is there.

One more thing to note is that you may have false leak positives from BSTR allocator due to caching, see "Hey, why am I leaking all my BSTR's?"

Share Improve this answer Follow

edited Sep 24, 2008 at 12:30

answered Sep 24, 2008 at 11:59

Constantin
28.1k ● 10 ● 63 ● 79