Subject: Re: C++ FQA
Posted by exolon on Tue, 13 Nov 2007 16:30:03 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Mon, 12 November 2007 22:33Well, also being absolutely not an expert of GC algorithms, I always think that to allocate a lot of memory just because there is enough of it it's not a great practice.
First thing, you must look to what does OS in respect of free memory.... If your OS tells you that you've got 1 GB free ram, even when it's already swapping out another GB on disk, what this behaviour does is slowing down your system.

IMHO a truly efficient GC should be hardware implemented, or at least have a strong hardware support. Doing it software you'll face everytimes with some sort of problem, as latency, memory inefficiency or both.
And it should also collect freed memory asap.
Well, these last two statements don't seem logical together. If you don't want latency, why collect freed memory ASAP, unless you're on a very constrained (hard real-time embedded board...?) platform always operating close to the limit of available memory or starting to thrash the swap? This implies doing GC runs _all_ the time.

mdelfede wrote on Mon, 12 November 2007 22:33No doubt that manual memory allocation is more efficient than GC, even if it can be a bot slower on the short time.
And a good framework can help to keep things simple.

Id rather extend C++ (or make some more modern language, without GC) to include some helpful features, than switch to less efficient languages.
I think you should really look at some proper comparisons of real efficiency impacts of using GC, rather than automatically assuming that it kills your program's performance.

Also, you shouldn't just assume for sure that manual memory allocation must be more efficient than GC.
This FAQ is worth a read, with an open mind, rather than being a hardened C++ "oldskool all the (hard and error-prone) way" purist.

http://www.iecc.com/gclist/GC-faq.htmlFolk myths
* GC is necessarily slower than manual memory management.
* GC will necessarily make my program pause.
* Manual memory management won't cause pauses.
* GC is incompatible with C and C++.
Folk truths
* Most allocated objects are dynamically referenced by a very small number of pointers. The most important small number is ONE.
* Most allocated objects have short lifetimes.
* Allocation patterns (size distributions, lifetime distributions) are bursty, not uniform.
* VM behavior matters.
* Cache behavior matters.
* "Optimal" strategies can fail miserably.