Subject: Re: C++ FQA
Posted by mdelfede on Tue, 13 Nov 2007 17:24:18 GMT
View Forum Message <> Reply to Message

exolon wrote on Tue, 13 November 2007 17:30
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.

When I say "I don't want latency", I mean "I don't want a big, unpredictable latency".
As I said, GC *can* globally be more performant, and *can* be in the very short time very performant too. What i can't is assure the performance *all* the time.
If I do a benchmark with a lot of allocations/deallocations, GC may behave wonderful. But then, if you stress benchmark beyond a threshold, you'll have your app stopping for maybe 2-3 seconds collecting stuffs. If you go forth, global app time *can* be shorter than with manual allocation, but in the middle you had the infamous stop.

Besides real-time tasks (on wich that's not acceptable), there are also many apps on which you couldn't accept such a behaviour, like multimedia, but even in a GUI a 3 second stop makes a bad feeling.

Quote:
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.

As I said before, one must know what "killing program performance" means. if you accept that your app can lag for seconds, GC is the best for you.

Quote:
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.

I'm surely not a purist, as I tried so many languages in the past. I'd prefere a more modern C++, but with no GC, but that's of course my opinion. I like very much C# syntax, for example, but you can't call him a fast or multipurpose language. When Java came out, it seemed to all people that it'd be the language of future.... but now I don't see many people thet like it. Nor it seems to me a quick lang, or a very error-free one. D language seems interesting, but I didn't test it.
Of one thing I'm quite sure : I don't want a C++ with memory management mixed between malloc() and GC.

BTW, in the faq you showed, the focus seems to be on "you program is surely buggy, it shourely have leaks, so GC is wonderful for you". I receive about 100 spam mails in this idiom each day
Quote:
.....

* Most allocated objects are dynamically referenced by a very small number of pointers. The most important small number is ONE.

for that, the best is pick_ behaviour, of course

Quote:
* Most allocated objects have short lifetimes.

And ? does it mean that GC is better or not ?

Max
