Subject: Re: C++ FQA
Posted by mdelfede on Sun, 11 Nov 2007 10:25:19 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 11 November 2007 09:54mdelfede wrote on Sat, 10 November 2007 17:57luzr wrote on Sat, 10 November 2007 17:54cbpporter wrote on Sat, 10 November 2007 11:31
And I'm quite surprised to see people who don't like gc, but have nothing against reference counting, which is slower and almost impossible to use efficiently in a multi-threading application.

I mostly agree with this...

As with GC, refcount can be made thread safe, IMHO.

The things is that even thread unsafe reference counting is about as fast or slower than mark&sweep GC.

Well, here that depends on *what* do you mean with 'faster'.
Let's speak for example about linux kernels, the normal one and the low-latency one. Which is faster ? That depends on what do you mean! Low latency kernel responds faster than normal one, but in overall time it is slower. The normal kernel can have troubles with real-time apps, but is overall faster.
So, refcount DOES add overhead on allocation operations, so for the single op IS slower than GC. But, when GC comes in place, you do have a long latency. In overall application time, GC is of course faster than many refcount ops.... with the counterpart of some 'program stops' during GC.
In conclusion :
GC is faster in overall app time, and is faster on single memory operation, BUT it has the big problem of GC stop time
Refcount is slower on single memory ops, is slower globally BUT has no stop times and it's execution is smoother.
If you don't need real-time response, GC is better, otherwise can be very bad.
The big problem, as you say here later, is that is very difficult to use other memory allocations in conjunction with GC.

Quote:
And, w.r.t. thread safety, the another trouble is that you cannot safely use atomic operations only when the object is really shared between threads (when it is needed).

I didn't understand that one...

Quote:
Quote:
Even pick_ can be not thread safe, if it's bad coded, and must have some sort of synchronizing code to be thread safe.

Everything can be thread unsafe if you really try. Anyway, the simplest pick_ implementation is naturaly thread safe.

Let's say that pick_ is very easy to make thread safe stuff

Quote:
Quote:
I would not accept a language based mostly on GC, but I've got nothing against an optional gc among other management stuffs.

The problem is that this is not quite possible.

I agree with this one... mostly. That's why I don't like GC.

And, in conclusion, I agree with you that pick_ is the best when you don't need a true object copy, or even when you do need it but you're ready to think much on what your code will do.
Refcount lets you to 'turn brain off', you must not bother about what your code will do with your object, mostly. So, less error prone. GC is the best for 'lazy people', it does all the dirty job but it leads often to slow apps... because people start allocating hundreds of objects on the fly without thinking about optimization.

Ciao

Max