Subject: Re: C++ FQA
Posted by Zardos on Tue, 13 Nov 2007 00:56:25 GMT
View Forum Message <> Reply to Message

luzr 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...

Mirek

If we talk about a naive reference counting implementation I agree without hestiation.

About the prejudice: "GC is fast" / "Reference counting is slow":

I have read 3 papers approximately one year ago which showed reference counting can be as fast as garbage collection. In addition there are some benefits. Unfortunately I don't have the links anymore. But you may try goolgle with something like "cycles reference counting" and read the /newer/ papers!

But some basic ideas I still remember:

1.) Aggressive removal of AddRef/RemRef by a static flow analysis of the programm: The idea is to only inc. or dec. reference counters if neccessary. Usually only if a (smart)pointer manipulation is done inside a struct/class/... It is not neccessary to do inc./dec. operation for (smart)pointers on the stack or as function arguments. Probably special care has to be taken for threads. Even for (smart)pointers inside structs many inc./dec. operation can be elimanated by a (not so complicated) static analysis of the programm flow.

2.) Avoiding "Atmoic operations": The simple but effective idea is this. Instead of doing AtmoicInc/AtomicDec operations directly on the refence counters all Increments and Decrements are logged in a buffer with a fixed length. Each thread has it's own buffers! So, no locks are required for the ordinary Inc/Dec. If the buffer is full the buffer is given to a "memory management" thread. This requires a Lock operation, but if we assume a buffer has place for 10000 "Inc/Dec" ops. the synchronization cost boils down to 1/10000 which is negligible.
The memory management thraed finally can perform the inc/dec operations without the expensive "Atmoic" vesions just ordinary ++/--
Benefit: The memory management thread can be tuned to only do a fixed number of delete operations per second. This avoids "cascading deletes" where one destructer call triggers a chain of delete operations. => Even with manual memory management there can be stalls!

=> So we have a concurent - nearly log free - reference counting garbage collection.

3.) Cycles: I have forgotten how they get removed. But it was not to complicated to understand. Yes it costs time to detect the cycles... The cost is reduced by point 1.). In addition the cylcle detection can be performed by the memory management - concurrently - without any additional

locks. And again this memory management can be tuned to do only N. operations per second. To really spread the memory mangament cost over time (Of course, this means for some time periods more memory is allocated than necessary).

I think it's a really interesting topic. I always had the feeling Refernce Counting can be supperior over garbage collection... May be the furture brings a revival of the counters....

EDIT: I think 2.) was implemented differently: Instead of having buffer with a fixed length: The memory management thraed performed a "stop the world" (pausing all threads) periodically and fetched the buffers from all running threads. While stop the world sounds like stalling - you have to remember only some pointers to buffers have to be transfered to the menory management thread... After the transfer the world starts rotating again.
