

CoCoALib - Design #1703

Threadsafty, multithreading: optional fn arg (or separate fn)

21 Oct 2022 15:02 - John Abbott

Status:	Resolved	Start date:	21 Oct 2022
Priority:	Normal	Due date:	
Assignee:	John Abbott	% Done:	80%
Category:	Safety	Estimated time:	0.00 hour
Target version:	CoCoALib-0.99880	Spent time:	1.35 hour
Description			
Might it make sense to offer the caller the option of choosing between a "threadsafe" call and one which is not?			
Example: the new class SumBigRat has operator+= which updates the internal state. The current impl is not threadsafe. This is not a problem if the SumBigRat object is only ever used in a single thread. Should there be a myAdd_threadsafe(..) function which a user can call if the same object is used in several threads?			
I think there may be several other possible situations: e.g. random generation?			

History

#1 - 21 Oct 2022 15:05 - John Abbott

I have not (yet) checked what the penalty might be for using a threadsafe version in a multi-threaded situation but where the object is safely used only in a single thread (at any one time).

Anyway, even if I measure the overhead on my machine right now, that is not helpful for other platforms.

#2 - 21 Oct 2022 22:39 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

I have implemented threadsafe mem fns **myAdd** (equiv to **op+=**).

Now it is not so clear whether **myTotal** also needs a threadsafe version; I suppose so... which is tedious! What should it be called? Or should it have an optional parameter?

#3 - 25 Jan 2024 20:22 - John Abbott

Anna reported compilation warnings with clang:

std::mutex cannot be copied or moved, so attempting to have default copy/move ctors for SumBigInt and SumBigRat causes problems.

RESOLUTION: I have removed (commented out) the use of mutex inside SumBigInt and SumBigRat, so these classes are **no longer threadsafe**.

If we really want a threadsafe version, my guess is that the correct solution would be to wrap the entire class inside another SumBigInt_threadsafe which handles the mutex before calling the relevant member function -- I fear that almost every call needs to be guarded (to avoid reading partially updated data).

#4 - 25 Jan 2024 20:27 - John Abbott

- Status changed from In Progress to Resolved

- Assignee set to John Abbott

- % Done changed from 10 to 80

SUMMARY

Offering proper thread-safety is likely trickier than I had hoped, and probably decidedly more costly at run-time.

It seems most unlikely that making just a single member function thread-safe will make the whole class thread-safe because we cannot do a read operation while some data member is being asynchronously updated.

Thanks to the clang compiler for flagging the problem. Here is a possibly useful link to StackOverflow:
<https://stackoverflow.com/questions/30340029/copy-class-with-stdmutex>

#5 - 07 Mar 2024 20:04 - John Abbott

- Target version changed from CoCoALib-0.99850 to CoCoALib-0.99880