atomic<shared_ptr<T>>

- `std::atomic<std::shared_ptr<T>>` probably won't be lock free :(

Tweet

Timur Doumler
@timur_audio

Hey C++ standard library implementers!

When C++20 comes out, and you start shipping `std::atomic<std::shared_ptr>`, will you ship a lock-free implementation?

@StephanTLavavej? And the others - who seem to be not on Twitter :(

5:27 PM · Mar 11, 2019 · Twitter Web Client

Stephan T. Lavavej @StephanTLavavej · Mar 11

Replying to @timur_audio

It is unclear to me how that can be implemented - double-wide atomics can copy the pointers, but how do you increment the refcount? BTW, @MalwareMinigun, @CoderCasey, @Eric01, @mclow are on Twitter (Jonathan Wakely was too, not sure if he still is).

6 6

JF Bastien @jfbastien · Mar 11

Replying to @timur_audio and @StephanTLavavej

Last I looked you needed a lock-free allocator to implement atomic<shared_ptr<T>> in case your inline buffer of class inheritance was too small. That's because the T can travel you class hierarchy. @a_williams has an implementation, probably knows better.

2 2

JF Bastien @jfbastien · Mar 11

I guess @LouisDionne will have fun 😐

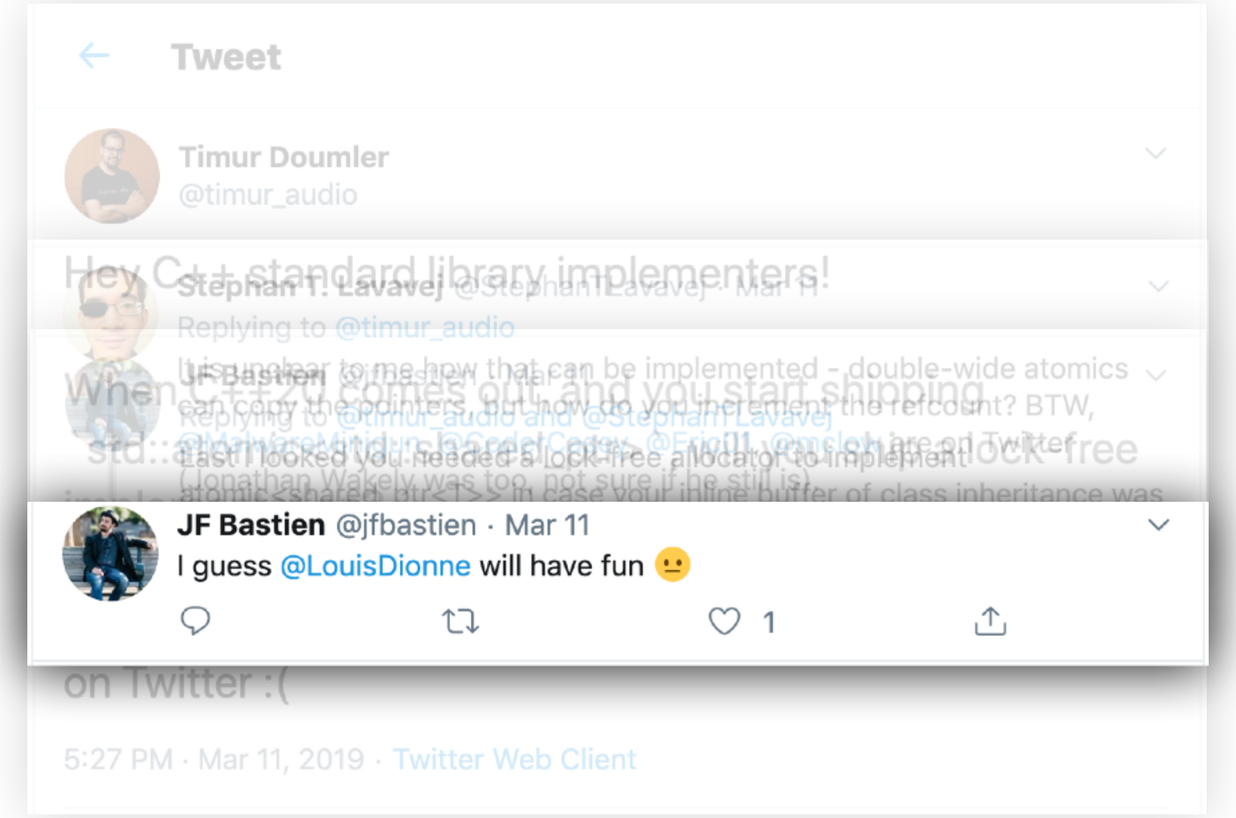
1


ADC

23

atomic<shared_ptr<T>>

- `std::atomic<std::shared_ptr<T>>` probably won't be lock free :(



Making Copies

- Copies of objects can involve system calls and synchronisation