Document Number: P0793R0
Date: 2017-10-16
Authors: Michael Wong
Project: Programming Language C++, SG5 Transactional Memory
Reply to: Michael Wong <michael@codeplay.com>

SG5: Transactional Memory (TM) Meeting Minutes 2017/06/19-2017/10/09

Contents

Minutes for 2017/06/19 SG5 Conference Call

Transactional Memory Teleconference minutes by Jens Maurer

2017-06-19 19:12 UTC

1.1 participants
Victor Luchangco
Michael Spear
Michael Scott
Jens Maurer (minutes)

1.2 adopt agenda
talk about 2.2 first
unanimous

1.3 approve minutes
unanimous

1.4 action items from previous meeting
none

1.5 call schedule
Victor will be away during the first three weeks of July.
Michael Spear will lead the July 3 meeting.
Discuss then whether to have the meeting on July 17, depending on attendance.
Michael Scott will be absent on July 31.

2.2 Interaction with executors
Jens: Gives rough overview of executors. Expresses doubt on hiding TM
inside an executor.
Michael Spear: Synchronized blocks can be expected to use TM features,
but coarse level of control; conflicts strategy cannot be customized.
Idea: Lambda-based interface to synchronized blocks:
  std::sync_block(lambda, config_object)
Jens: This is at a similar level of ignorance of cross-translation
unit issues as SIMD-parallel algorithms in the STL. Both will
performance-fail spectacularly if a function in an opaque, foreign
translation unit is called often.
Jens: Ask Robert Geva (Intel) for the state of cross-translation unit
concerns for SIMD-parallel algorithms.
Jens: Gauging WG21 (as a whole) support for moving TM into the C++ IS
without actually asking the WG21 plenary is guesswork at best, so

detailing a plan B seems a bit premature. Opposition of GPU folks has been noted, though (they'd rather spend their silicon on TFlops than TM).
Jens: What's our response to the GPU folks? Make TM optional in the C++ IS?
Michael Spear: For HTM, trying twice and then falling back to global synchronization doesn't really work (cf. gcc), so configurability of conflict resolution is probably useful.
Victor: We still lack demonstrable use of TM for going into the IS.

2.1 Paper for advancing TM into C++20
Not yet ready, maybe someone wants to champion this.

4.3 Action items
Action item for everybody: Form an opinion on TM in C++20, and address the concerns of the GPU folks.
Action item for Victor: Send telecon credentials to Michael Spear.

5.1 Next agenda
Same as this agenda.

5.2 Next meeting
July 3, 2017

20:01 UTC

Minutes for 2017/08/14 SG5 Conference Call

Attendees: Mike Spear, Michael Scott, Jens Maurer, Hans Boehm, Victor Luchangco (scribe)

Secretary rota: Michael W, Hans, Maged, Michael Scott, Michael Spear, Jens, Victor

Amended agenda to include report about C++ standards meeting.

2. Main issues:

2.0: Report on C++ standards meeting

Jens:

Concepts added concepts to C++ working paper (again).

Modules: Microsoft proposal out for ballot for Technical Specification (TS). (Google has a more ambitious proposal, thinking about whether they can live with the Microsoft proposal for now, possibly extending it incrementally).

Coroutines also going out as TS (stackless version: coroutines as continuations, rather than saving the stacks).

Ranges TS also out for publications (concepts-based).

Networking TS (Boost IO) going out for publication.

Clarification about “wavefront sequencing guarantees” for vectorization.

Michael Scott: What is the long-term steady-state for C++? Will it continue to grow forever, or is there some idea that it will stabilize.

Jens: Some people want to keep adding “shiny stuff” to the language, which is necessary to keep C++ ecosystem “fresh”, even if this means incompatible changes (as long as there is a clear migration path). Others are concerned about legacy code, and don’t want novice changes. We definitely aren’t stabilizing yet. There are definitely things we need to add/improve (e.g., networking) and other things for which there are strong advocates (e.g., contracts), but in any case, the official stance is that C++ will keep evolving as there is need.

Hans: On Coherence TS: From Herb: Should concurrency TS move into C++20? There is some push to incorporate some parts (latches, atomic shared ptr), or at least discuss some features that might be incorporated (e.g., futures).
There’s an ongoing discussion of the C++ memory model. Hans is in charge of this, though much of the push is from work by Victor Vafaides’ group: current memory model isn’t implemented by many implementations, and will be difficult to implement efficiently on some

architectures. Also, some features are too weak (semantics of memory fences), or seem badly specified (e.g., release sequence).

Jens: The feeling is that there are “bugs” that need to be fixed, but the overall approach (i.e., specifying memory model in terms of partial orders) will not change.
We will need to reformulate what we have in TM TS when dust settles.

2.1: Consider putting in a paper for advancing TM into C++20

> Herb has pushed out an email consulting National Body Heads on their opinion on various TSes advancing into C++20, including:
>
> Ranges, Networking, Modules, Coroutines, Concepts, Concurrency.
>
> This is because they either have or is expecting papers in the Toronto meeting proposing to advance it into C++20.
>
> We should start to consider putting a paper in the next meeting (there is still time after Toronto), proposing what we wish to add to C++20 (part of TM TS, all of TM TS, a modified form of TM TS, or nothing) by talking about what we wish to aim for.

Mike Spear: Could I suggest putting forward “TM-lite”?

Mike Spear: It’s frustrating to work on this for a long time, but no progress in standardization. Can we include just the least controversial parts in, and then we can get feedback: too much, just right, or here are more things that we need?

Jens: synchronized blocks or atomic blocks?

Mike: don’t know I want either, in current names. Ease of coarse-grained locks with scalability of TM.

Michael Scott: one possibility: atomic blocks only, with executor style API.

Mike: To play devil’s advocate: that requires us to define transaction-safe.

Michael: Yes, but that’s what springs to (my) mind from your motivation.

Mike: yes to executor-style, but we just give the guarantees of reentrant single global lock.

Jens: executor-style is a way to sidestep the hard questions related to translation between different execution domains (like vectorization does)

Mike: What does standard say about “warnings”?

Jens: standard doesn't specify this. Hard part is not giving warning. The hard part is allowing programming to annotate program to avoid the warning.

Michael: For TM: if we want warnings that ensure that transactions don't do unsafe things is to add transaction-safety into the type system.

Mike: the runtime can participate in determining whether transaction is running something unsafe.

Jens: if it can only be determined at runtime, then you won't get static checking (i.e., we need to get warning-free compilation).

Mike: we can have library-based functions that assert (by the programmer) that some code is safe. That is, have one function that you can pass a lambda that the programmer claims is safe, or another that the programmer is okay with it causing serialization if it isn't safe.

Jens: I think serializing is akin to failure.

Hans: that can be useful where I expect there not to be much contention, so I'm okay with this serializing.

Hans: weird to cast this into an executor framework: it doesn't look like it to me. (P0290-R2 may be better syntax)

Sorry, I forgot to provide the link to this: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/p0290r2.html

Mike: I think Michael and I are using "executor" in a common sense, not necessarily the specific one in C++ (but we should be careful not to give the impression that we mean the specific one in C++).

Hans: I'm pushing back because we have a problem with executors growing to include the whole world. But I agree with a library approach.

Mike: Here's a way we can get lock elision without having to actually change the type system in the immediate future.

Jens: I'm feeling that we should sidestep the difficult issues, and say that for the semantics point of view, it's a global mutex, but with the idea that it should be elided using TM.

Mike: Chandler had some interest in not defining this as a lock, to avoid some synchronization. Perhaps we could have the function take an extra parameter that doesn't do anything yet, but could provide an upgrade path.

Hans: I'm not sure that we want to just fall back to defining it as a lock: it's more complicated.

Jens: from a core-language perspective: I'm uneasy with function call not having normal function call semantics. (If we go this way, we definitely need it looked at by the evolution group.)

Hans: not sure that this fundamentally break function semantics
<we should discuss this in more detail when we have something to look at>

Mike Spear offers to start writing a draft for "TM-lite" proposal; Michael Scott and Victor offer to help.

Jens: want concrete examples (i.e., with specific syntax).

AI (Mike Spear, Michael Scott, Victor): Write above-mentioned draft
AI (everyone): Read draft when it is written

Meeting adjourned. (Next meeting on Aug 28.)

Minutes for 2017/10/09 SG5 Conference Call

Michael W, Hans, Maged,Michael Scott,Michael Spear,Jens, Victor,

Notes by Michael W

Agenda:

1. Opening and introductions

1.1 Roll call of participants

Michael W, Jens, Michael Scott, Victor, Mike Spear

1.2 Adopt agenda

Yes

1.3 Approve minutes from previous meeting, and approve publishing previously approved minutes to ISOCPP.org

Yes

1.4 Review action items from previous meeting (5 min)

1.5 Call schedules

Aug 14 DONE

Aug 28

Sep 11

Sep 25 Michael away

Oct 9 Mailing deadline Oct 16

Oct 23

Nov 6 C++ Meeting Albuquerque

2. Main issues (50 min)

2.1: Should we change the times of this call

Time is still OK,
changing now could change semester schedules
changing frequency will cause even more meetings to missed. So not desirable.

2.2 Future of TM

By pass

2.3: Interaction with Executors

https://groups.google.com/a/isocpp.org/forum/#!topic/tm/jG9XPJetNkc

The last discussion has us considering an alternative lambda form.

See Paper emailed out on Lambda proposal

Synchronized lock is lock elision
what if we make it a call to a library function passed to a lambda
no new keyword
uses a simple implementation
use global reentrant lock that executes your code, and release the lock
Michael W interested in seeing if this will fit the executor interface, then passing a callable as an input to executor
appears as a library interface
but can use compiler magic, which makes it be better then Justin's Gottslieb library-only interface
4 level of implementations questions here are being addressed:

1. reentrant global lock, advantage implement semantics as specified of this tm synchronized function; disadvantage, disappoints user as it does not obtain the concurrency that the name of the function suggests

2. implementation essentially replaces global lock with lock elision in HW

3. instrument the compiler to recognize the TM synchronize with a lambda pattern ; instrument (clone) for stm; serialize if across TU

4. get compiler to instrument/clone even if passing a functor that was obtained from somewhere else across TU.

transaction_callable may still be needed (does not change type system)

need a clone of the body,

discussion on many things in C++ that is inappropriate on GPU, including branches, exceptions, not just TM

Mike Spear's group has implemented it in llvm plugin up to level 4 under 10000 lines of code

in gcc, we have to do a lot more work

do we need transaction_callable to preserve semantics? No just for optimization
even if passed to something that is not labelled as TM, will still work
So is there agreement to add transaction_callable as well

Is this a replacement of TM TS? No, it could be completely different.

Some within this group already does not like that there is no atomic block support
Could be taken further to atomic blocks, could take more arguments to enable this

We seem to need a few more rounds of discussion on this so this is not aiming for
- show quoted text -
1. Michael to import doc to google and enable people to work
2. Michael to add the 4 levels of implementations
3. Michael to ask whether TM callable to be supported in paper