Document Number: P0570R0
Date: 2017-02-06
Authors: Michael Wong
Project: Programming Language C++, SG14 Games Dev/Low Latency/Financial Trading/Banking/Simulation/Embedded
Reply to: Michael Wong <michael@codeplay.com>

SG14: Low Latency Meeting Minutes 2016/12/14-2017/02/01

Contents

Minutes for 2016/12/14 SG14 Conference Call 2
Minutes for 2017/01/11 SG14 Conference Call 9
Minutes for 2017/02/01 SG14 Conference Call 14

Minutes for 2016/12/14 SG14 Conference Call

Meeting minutes by Michael
With large numbers of participants, audio interference can be a problem. Please try to keep your phone muted whenever possible. If your phone does not have a mute button, the bridge will mute or un-mute your line if you dial *6.

I will take notes for the calls, though it will also be recorded.

Agenda:

1. Opening and introductions

1.1 Roll call of participants
Michael Wong, Billy Baker, David Hollman, ISabella Muerte, John Szwast, Brett Searles, John McFarlane, Marcelo Zimbres, Diane C.

1.2 Adopt agenda
Yes, Brett seconds

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

1.4 Review action items from previous meeting (5 min)
None

2. Main issues (125 min)
2.1 C++ Std Meeting Issaquah recaps
https://wongmichael.com/2016/12/09/the-view-from-nov-2016-c-standard-meeting-issaquah/

Discuss what to do about papers that are missed.
1. be there for the next meeting OR
2. send a note to the chair EWG, LEWG, LWG, CWG, SG1 OR
3. Assign a proxy who will be there

CPPCON 2017 SG14 will be in Resident Inn Marriott, about 5 minutes from Meydenbauer, and we will only 1 hour for lunch

2.2 SG14 for HPC meeting in Berkeley NL
Large interest in heterogeneous computing in C++, partly to support exascale computing requirement of accelerators, and there is interest in moving
away from traditional OpenMP/MPI/legacy languages. Each projects interested in its own runtime.

2.3 Meeting C++ SG14 discussions

large embedded interest in C++
limitations in embedded device (such as no virtual functions, exceptions) causes subsetting of C++, means we need to integrate with this community's needs before thy fragment

2.4 Review paper mailings and SG14 position/feedback on any papers from the mailing: 30 min

1 Additions: Intrusive smart pointer, Isabella Muerte
Bryce was to present, this did not happen due to LEWG triage, P0468, wants to present it at Kona,
David: people were aware of it and can help with it

2 inplace functions: Nicolas
not presented
Carl Cook wants work on it
3 Fixed point real numbers, P037 John McFarlane, P0106, Setwidth (P0381)
Had a special SG6 call last Wednesday
gave mini-presentation on this, compared them
will try to unify it for Kona,
P101 talks about fixed point + lower level
may write a position paper on rounding and overflow
4 Ring span, Guy Davidson
Michael Wong presenting on behalf of Guy Davidson
Ring span is a circular buffer as SG14 like contiguous structures
Changed from Ring buffer to span to refelct normal langauge as it is a view across non owned memory
Referred back from LEWG in Oulu
Nat: why is a requirement for single producer/ single consumer
Hans: what is in mind is the avoidance of contention because producer and consumer are at opposite ends
Michael: multi prod/cons adds overhead
Will: notes that concurrent ring buffers are in wide use between hardware and software
Detlef:How do I wait on space?
Will and David discussed memory order constraints .
Will: this is acquire release in most operations
Lawrence: Nothing else really makes sense
David: we cannot allow default SC if this is really performance based
Will and Lawrence: Comment that this is a sequential case with bolt ons and that is not working well
David does Olivier and Geoff discuss the intention of ContiguousIterator
Geoff: The point of ContiguousIterator is to allow type erasure
Nat: respond to Will's observatio. Notes that the author state that this is not a concurrent ring but rather is adapted from a concurrent queue
Michael seeks view from finance
Thomas confirms that this is a use case in HFT
Olivier concerned about single/single versus multi/multi which apparently involves a lot of razor blades and a close shave!
Detlef: I would like to see a more compelling use case that this is different to Lawrence's queue
Lawrence: Has a sequential interface that requires a test pattern to enable access. Needs and extension to the interface
Geoff: Why is there an iterator in this as it does not otherwise meet requirements?
Lawrence:This is for debugging purposes as it allows you to work our why it crashed. But in concurrent use this has issues
Neil: I don't think that that can be the motivation, it does not feel appropriate to SG14 type use cases if Lawrences motivation is correct
Nat: We may need

Thomas: This is a view type and thus has an underlying container.
Paul: Can we use a trait to support the iterator.
Hans: We need to performance justification.
Lawrence: history buffer and communication buffer are different things
Nat: Are there concurrent ring span use cases that cannot be addressed by the concurrent queue
Michael: get the authors to work with Lawrence
Nat: Use of an adaptor over another container, is mostly an avoidance of memory allocation, there is at least one impl of Lawrence's queue that can do this.
Can Lawrence's queue be adapted to work over an underlying memory?
Detlef: Push_back is different in serial versus concurrent, this is not acceptable in a single data structure.
Hans: We really need to understand why the two different use cases are represented in a single impl, as this complicates matters
Nat:Can we focus on the two cases of single/single multi/multi, not worry about the single/multi variants. Should direct the authors to come up
with a use case for concurrent communication that is not captured in Lawrence's queue. Maged: Notes that the single/multi variants can have special performance optimizations
Olivier: To counterNat, we need a non-concurrent sequential and a multi/multi/concurrent. Thus two interfaces, sequential and concurrent are the two that we need.
sequential is the single/single while multi/multi should be defined in terms of concurrent queue. Divorce the two interfaces as they are no compatible with one another
Our job is to provide the correct vocabulary for interfaces. Implementations are free to innovate but the user gets standardised API.
Advice to split this in to two papers. Thread unsafe goes straight to LEWG. SG1 is concerned with the concurrent and there are severe reservations around iterators
In order to allow the association between the underlying and the span, we need to be able to access the head and tail positions directly.
Hans: Java APIs would typically snapshot, is this something we can consider.
Michael summary
If concurrent interface is to remain we need a strong justification for divergence from Lawrence's interface.
A performance argument would probably need to be more precise about memory ordering.
Divorce the concurrent_ring_span and ring_span (ring_span can go to LEWG).

Concurrent ring span deferred to concurrent queue,

5 Hazard Pointers + RCU Maged + Paul
Maged Michael Hazard Pointers P0233R2
o Discussion about Hazard Pointers overview.
o David: Use of std::atomic in try_protect -- What if someone wanted to use atomic_view or some such? Would like to see use of pointer-type traits and decltype to enforce the API, but allowing additional generality.
o Mike: Use in std library data structures? Hans: Later.
o Discussion of differences between reference counting and hazard pointers.
o Lawrence: What are all the performancn tradeoffs, given all the possible data structures? Can we make a switchable API that can be dropped in anywhere? Can we have common code for data structures using these things?
Paul/David
Use template parameters.
o In response to JF questions: Lawrence: Want one API with multiple performance tradeoffs. Could have many concurrent data structures with all combinations laid out, which does not seem reasonable.

o JF: Do you want to hold up Hazard Pointers, RCU, etc all tied together in one TS? Lawrence: I want people to think about the eventual goal.
[Ensuing discussion during break covered ways that a single data structure could adapt to these approaches.]
o Potentially remove set() if needed.
o Olivier: Could we abstract out more hazard-pointer operations?
Olivier
Could we automate .clear()/.retire() for exception safety?
Exception safety discussion...
o Straw Poll: Produce wording for hazard pointers current interface?
SF F N A SA (By acclamation.)

RCU:
Difference between RCU and upgradable RW lock?
Paul: Usually nothing happens on reader lock acquisition. In about 3/4 of cases, rwlock could be replaced by RCU.
How are more (no action for reader lock) or less aggressive (update thread-local counter) implementations chosen?
Paul: Different objects provided.
How are RCU domains determined?
Paul: Currently often determined by compilation unit.
Geoff: Probably wouldn't fly for C++. Domains needed because you don't want to wait for slowest reader.
Nat & Paul: Allows readers requiring very small cycles.
Paul: tradeoffs between static functions and virtual function dispatch on rcu_domain.
rcu_head_delete better named enable_rcu_pointer_from_this.
rcu_head_ptr is then the non-intrusive version. This is what David has reimplemented in a more C++ way.
Interface. Convert rcu_domain virtual base class to concepts. User can re-introduce virtual class if needed.
David led the group through his implementation.
Suggestion: Rename "call()" to "retire()". Compatible with hazard pointers.
Nat: What is __nat? David: Help with the enable_if pattern in constructor, handing in a type that the user could not possibly provide. "nat" means "not a type".
Geoff Romer: Will check to see if a Google implementation can be presented.
Discussion of what lock-like class/concept rcu_read_lock() and rcu_read_unlock() should be mapped to. There is no shared_lockable concept, but it could be introduced. This would allow P0461R0's rcu_scoped_reader to go away.
Gor: Proposal for intrusive_pointer in play.
Handling of lambda capture in the simple case just does a new for the storage required for the lambda. A more complex approach would provide a small amount of memory for the common-case lambda captures, but would need allocation for sufficiently complex lambda captures.
Geoff: Use of rcu_head -- concerned about exposing this to the public API. Google's implemention uses a vector instead. Don't want to invalidate Google's implementation. Further discussion made it appear that there are some differences between userspace RCU's defer_rcu() and the Google approach.
Maged: What happens when vector fills?
Geoff: Quite possible that Google is using a non-throwing allocator, which would allow another vector to be allocated.
Pablo: Could hide the extra data structures in templates. So don't know why one would expose rcu_head.
David: Could have both low-level API that exposes rcu_head, and also high-level API that does not. Probably don't want to make rcu_head a concept. But maybe just leave rcu_head out.

Nat: Grace period guarantees that all pre-existing readers get done. Discussion of guaratees and what developers need to know to successfully use it.
Maged: Should non-intrusive implementations be preferred? David: Hazard pointers might have special considerations.
Nat: Unlike shared pointer, hazard pointer and RCU have different removal and reclamation times.
David: call() is almost always given a deleter. Note that you are not allowed to touch an RCU-protected object after you have passed it to call(). Small-block optimization.
However, value-capture lambdas might provide "interesting" lifetime issues. The value-capture lambda would likely need to be moved before invocation in that case.
Straw poll: Is David Hollman's implementation an advance in the right direction over P0461R0?
Yes by unanimous consent.
Hans: Can the C++ RCU API be decoupled from memory_order_consume? Olivier: Could parameterize the memory order. Could require the user to cover it. Could parameterize the equivalent to rcu_dereference() or atomic_load_explicit().
Straw poll: Any objections to this going forward?
No by unanimous consent.

HP moves to wording, RCU contineus work
6 Thread constructor attributes, Patrice
Continues work
7 Intrusive containers, Guy Davidson
Not presented by Hal, may be in common with intrusive ptrs

8 plf colony/stack, plflib.org Matt Bentley
Not presented
9 Likely/unlikely
Bryce Lelbach for Clay Trychta
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2016/p0479r0.html
This proposal on branch prediction hints comes from SG-14 list. This may be possible as a library feature.
Want to add two attributes, [likely] and [unlikely].

- Chandler: If anyone is unconvinced that we should standardize this, we should look at existing practice. This is incredibly wildly used.
- Ville: If people doubt that this is used they should take it up with Linus Torvalds as he can provide polite assistance.
- Bryce: This is in GCC, Clang, etc.
- Hubert: You can still use likely/unlikely in switches.
- Bryce: The way the builtin is used in GCC and Clang is that it takes and returns longs. There are places where you have to convert from a bool to a long just for this which is somewhat annoying.
- Matteus: Which types would you support? Any type, but only do optimization for long.
- Bryce: Paper suggests only adding these for if statements. Would like to see if we want to go with this approach or a library approach.
- Chandler: Want attributes for one case, but not the case in this paper. There's a use case for likely/unlikely but it's more about control flow than attributes. You should be able to put these in expressions for if blocks, else statements, case statements, etc. Compilers always try to make these control flow. Should stick these on the open curly.
- Matteus: Want these on while and switch.
- Daveed: Would also consider for the general statement. But when it's not on a control statement it feels more like hot and cold.
- Chandler: I'd say outside of control flow this attribute gets dropped.
- Hal: We end up with code falling into two groups: error handling that's not just unlikely, it's almost never. The other is branches. We'd like these to be different in the optimizer but don't have

that utility. There are a lot of things we'd like to get out of value expectation such as expecting a loop trip count to be x, or [4, 5, 7]. But we should have likely, unlikely, and almost never.

- David Stone: Is it valuable to be able to say this is probably 1, if not that, probably 2.
- ???: There may be a situation where likely and unlikely are distinct from optimization. You may want the unlikely to happen quickly.
- Hubert: Likely/unlikely is about branching, not effort to optimize.
- Chandler: As soon as you get into fine-grained things I'm concerned about putting it into source. Everyone who wants real fine-grained use a real profile. I see these as instructions to the optimizer: optimize for the case where this will happen.
- Hal: In theory, I agree, but in practice we end up with problems assigning probabilities.
- Matteus: We use this not to optimize what is likely.
- Daveed: Additionally, anything more precise will get stale very quickly.
- Chandler: Want to start with as strong a wording as we can to discourage the use.
- Hal: To push back on profiling, I don't think profiling is what you want in practice. Profiling gives you averages, but we don't have good value profiling technology. Intel has a way to tell loops a range of values. People use that a lot.
- Chandler: I understand the value of a value expectation framework.
- Ville: I'd like to poll whether we want to do this as attributes on statements.
- Tony: I want to be able to say with a large set of if-else that the final return x is unlikely.
- Matteus: I would like to bikeshed. Don't tie the names to probabilities.
- Hubert: If this is purely on the statement then it's hard to insert into the middle of an expression.
- Adam: As it stands now, people may overuse this.
- Chandler: Not as worried about conditional operators. Short-circuit chains come up all the time.

Straw polls: SF | F | N | A | SA

- Solve P0479 with attributes on statements? **0 | 19 | 6 | 0 | 0**

More work requested
10 Comparing virtual Functions , Scott Wardle (not present)
Not presented.

2.5 Other papers presented at Issaquah meeting: 25 min

SG14 at CPPCON 2017: presentation
NN C++ Std library interface for NN, interest in submitting common interface from Tensorflow, theanos, caffe, mxnet which are all C++ neural network libraries used by major companies: Google, amazon, facebook

Paper that address NB comments, that has not been reviewed, should be mailed in, even if there are post-meeting discussion (those can be added as a new section

2.6 Embedded group special meeting in Jan 11 SG14 meeting

2.7 Future F2F meetings:
1.. Embed.io : Feb 18, Bochum
2. CPP Kona: Feb 28

2.7 future C++ Standard meetings:

N4573 2017-02 Kona WG21 Meeting Information
2017-07-10-2017-07-15: University of Toronto/Canada
https://isocpp.org/files/papers/N4607.pdf
Nov 6-11 Sandia, Marriott downtown Abq
3. Any other business

- **Reflector**
  - **https://groups.google.com/a/isocpp.org/forum/?fromgroups=#!forum/sg14**
- **As well as look through papers marked "SG14" in recent standards committee paper mailings:**

http://open-std.org/jtc1/sc22/wg21/docs/papers/2015/
http://open-std.org/jtc1/sc22/wg21/docs/papers/2016/

- **Code and proposal Staging area**
  - **https://github.com/WG21-SG14/SG14**

4. Review

4.1 Review and approve resolutions and issues [e.g., changes to SG's working draft]

4.2 Review action items (5 min)

AI: michael to email Jeffrey Yaskin LEWG about intrusive ptr
5. Closing process

5.1 Establish next agenda
Embedded SIG SG14 meeting in Jan 11
5.2 Future meeting
Next call : Jan 11 2017
Feb 1: Note Feb 6 is the mailing deadline
Feb 11: Kona Std meeting
Close

Minutes for 2017/01/11 SG14 Conference Call

Meeting minutes by Michael

**With large numbers of
participants, audio interference can be a problem. Please try to keep
your phone muted whenever possible. If your phone does not have a mute
button, the bridge will mute or un-mute your line if you dial *6.**

I will take notes for the calls, though it will also be recorded.

Agenda:

1. Opening and introductions

1.1 Roll call of participants
Michael Wong, Billy Baker, John MacFarlane, John Szwasit, Marco Foco, Ronan Keryall, Odin Holmes, Lee Howes, Guy Davidson, Brett Serles, Mateusz, Paul Bendiction, Shay

1.2 Adopt agenda
Approve.

1.3 Approve minutes from previous meeting, and approve publishing previously approved minutes to ISOCPP.org
Approve.
1.4 Action items from previous meetings
N/A.

2. Main issues (125 min)

2.1 Embedded group special meeting in Jan SG14 meeting
Odin, C++ embedded usage conference
EmboC++ in Bochum Feb 18
metaprogramming, library design
Odin is based in Bochum
aim of the conference how to make C++ better for Embedded programming
1. exceptions is key
2. const expr is unfinished: assert is to throw an exceptions
3. std vector is too heavy weight, vector of ints zero initialize everything ; use [unique ptr] helps, but can still be slower then new and delete if you dont use no except, depends on the size of the

data you work on, 0-8 byte long, still have pitrs added to that
May be N3986
also add a GSL section for embedded section
May be SG13 HMI
Embedded namespace in the standard
interrupts is a good example, more event based run to completion style is ideal in embedded, because it is deterministic, memory management, free list, not the heap that could be fragmented
deque may be between 2 pool sizes
make vector grow by pool size instead of just doubling
also locking is diffeerent, if I am on a threading model, so I block so you cant
interrupt priority highest does not block, lowest interrupts higher one
atomic is hard as it defaults to a lock

apart from interrupt, these are the same concerns with video game developemnt, OpenGLES, not desktop openGL
worrying the Big O
being late is terrible, if you are not in the next frame
air bag esample, too late you are dead, must be deterministic

static assert/contract programming useful, make special function register safer, any change needs to be checked
wrote a whole DSK using expression template
, ccant corrtupt reserve bits, or memeory

regulation with C/C++
theoretical proven C, C++ may never reach that as proving C++ is much harder

unit tests needs a lot of improvement
swap out a trait template, to do unit tests
IOT automation, and C++ needs this regulation and C++ can help
Paul: to use those things, need to be very good at MPL, most C programmer are not,
show how little they need to know to get greater improvement

many use complexity as the ultimate argument why C++ cannot be used:
https://twitter.com/mmalex/status/818198653743591424

1. justify strong typing benefits
2. scientifically tested zero abstraction claim

Odin has a student that is testing these in a paper

Xilinx: fpga program easier C++ opencl , vhdl, verilog, not full fledged C++, no heap allocation
use a lot of metaprogramming to provide specific data types
improving our tools libraries, SG6 fixed points,

just rename your C to cpp files would be good

ecceptions and how volatile is imterpreted on micro controllers
rarely switches compiler in this domain
have layout tricks

volatile with intrrupts very tricky, may move it in
hwo to read a register and do nohting without uising volatile

thread safe ring buffer

Embedded C++ has not great reception, hated by some
because they removed:

- Multiple inheritance
- Virtual base classes
- Run-time type information (typeid)
- New style casts (static_cast, dynamic_cast, reinterpret_cast and const_cast)
- The mutable storage class specifier
- Namespaces
- Exceptions
- Templates

at the time, template was not implemented efficiently, now its ok

a living document that changes in embeeded using GSL as the basis

Plan of action
1. write a paper for Kona outlining our disucssion and approach
2. meet once every quarter to discuss GSL additions for embedded
3. write papers and talks to address specific issues for C++, CPP CON, or EMBO
4. more scientific research on type safety and zero abstraction verification
5. Tools support ISO IAR, greenhill, Kyle may be going to clang
6. also emphasis on safety critical - needs better assembler analysis,talked to ABSINT

MISRA C++ use template but uses an old standard

Shay: saids no problem with using MISRA C++ no problem with ASIL B: please contact Odin Holmes and Michael Wong

2.1.1 Previous papers:
N3986

	Adding Standard support to avoid padding within structures

N3990	Adding Standard Circular

	Shift operators for computer integers

N4049	0-overhead-principle violations in exception handling

N4234	0-overhead-principle violations in exception handling - part 2

N4226	Apply the [[noreturn]] attribute to main as a hint to eliminate global object destructor calls

Other current proposals that help embedded programming
fixed point
ring span, Paul Bendixen asked some signal processing some concerns, if you use more complex object , how is the destructor called if you overrun. Guy Davidson will reply
small vector clump,
set sso size for standard functions
inplace allocator construction
inplace string
may be try out in boost
building a library that supports embeded use will help advance the cause
it also may not be restricted to boost: like EASTL
discussion on whether Boost is the right place if it imports lots of library that use excveption, and TPL
atomic queue more embedded related super light wright

wish list:
event programming/run to completion
embeded programming

- show quoted text -

2.2 DIscussion about current ongoing proposals.

2.3 Future F2F meetings:

1.. Embed.io : Feb 18, Bochum
https://www.embo.io/
Possible SG14 meeting at the conference?

2.7 future C++ Standard meetings:

N4573 2017-02 Kona WG21 Meeting Information
2017-07-10-2017-07-15: University of Toronto/Canada
https://isocpp.org/files/papers/N4607.pdf

3. Any other business

- **Reflector**
  - **https://groups.google.com/a/isocpp.org/forum/?fromgroups=#!forum/sg14**
- **As well as look through papers marked "SG14" in recent standards committee paper mailings:**

http://open-std.org/jtc1/sc22/wg21/docs/papers/2015/
http://open-std.org/jtc1/sc22/wg21/docs/papers/2016/

- **Code and proposal Staging area**
  - **https://github.com/WG21-SG14/SG14**

4. Review

4.1 Review and approve resolutions and issues [e.g., changes to SG's working draft]

4.2 Review action items (5 min)

5. Closing process

5.1 Establish next agenda
Feb 1; review existing proposals for Feb 6 deadline.

5.2 Future meeting
Next call : Feb 1 2017, moved 1 week ahead of normal schedule due to Mailing deadline
Feb 6: Mailing deadline

Minutes for 2017/02/01 SG14 Conference Call

Minutes by John McFarlane

John
- show quoted text -
- show quoted text -
John McFarlane (Chair)
Vishal
Guy Davidson
Odin Holmes
Adam Yaxley
Isabella Muerte
Mateusz

1.2 Adopt agenda

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

2. Main issues (125 min)

2.1 Paper prep for Mailing Deadline
Logistics
Asking for a paper:jhs@edg.com
As a reminder, the mailing deadline for the pre-Kona mailing is 2017-02-06 at 14:00 UTC.

Also as a reminder, the paper guidelines and mailing procedures can be found here:

https://isocpp.org/std/standing-documents/sd-7-mailing-procedures-and-how-to-write-papers

I may or may not be reading email regularly during the weekend before the deadline, so I would recommend submitting paper number requests by Friday 2017-02-03.

Thanks,

John.

2.1.1 Papers and proposals
1 Additions: Intrusive smart pointer, Isabella Muerte
Bryce presenting
Yet to ask Bryce to present at Kona. Needs to discuss paper with Michael. Had revision versus GitHub version.
2 inplace functions: Nicolas
not presented
3 Fixed point real numbers, John McFarlane

New paper: P0554 - Composition of Arithmetic Types
4 Ring span, Guy Davidson
Concurrent ring span deferred to concurrent queue,
New version submitted to GitHub. Simple revision. No more concurrent ring-span.
In Meeting C++ last year, Michael mentioned SG1 said: don't make it concurrent.
But embedded still want concurrent. Suggestion: two layers.
Odin: popular container in embedded sphere.
Details of discussions regarding concurrent vs non-concurrent can be found in various minutes of previous meetings.
- show quoted text -
Discussion of an SG14 library. Should it be part of Boost? Should there be industry-specific libraries? Licensing issues.
Discussion of non-silent noexcept as compiler feature. And issue of forking compilers to prototype language features under the WG21-SG14 repo.

-
  -

4. Review

4.1 Review and approve resolutions and issues [e.g., changes to SG's working draft]
4.2 Review action items (5 min)

5. Closing process

5.1 Establish next agenda

5.2 Future meeting
Next call : March 8
Jan 11: EMbedded Special Interest Group meeting
Feb 1: Paper mailing deadline review
Feb 6: Mailing deadline
March 8: