Document Number: P0048R0
Date: 2015-09-25
Authors: Michael Wong
Project: Programming Language C++, SG14 Games Dev/Low Latency/Financial Trading/Banking
Reply to: Michael Wong <michaelw@ca.ibm.com>

SG14: Games Dev/Low Latency/Financial Trading/Banking Meeting Minutes 2015/08/12-2015/09/23

Contents

Minutes for 2015/08/12 SG14 Conference Call

Meeting minutes by Michael

1.1 Roll call of participants
Michael Wong, Sean Middleditch, Billy Baker, Guy Davidson, Scott Wardle, Johh McFarlane, Nicolas Guillame, brent friedman, Greg Bedwell

1.2 Adopt agenda
Yes with this modification
add 2.2 form small subgroups.
rolling queues.

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)
1.4.1. All: Consider attending SG14 meeting at CPPCON 2015 hosted by ISOCPP.org Wed. Sept 23rd, Meydenbauer Center, Redmond, WA

1.4.2. All: Consider attending Kona Meeting in Oct 19-24, 2015.
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4544.htm

1.4.3. All: consider attending SG14 meeting at GDC 2016, March 14-18 hosted by SONY

2. Main issues (50 min)

2.1 Review SG14 logistics, discuss CPPCON Games track, C++ Standard meetings

Guy Davidson, Scott Wardle, Billy Baker, Brent Friedman, Nicolas, Sean. Michael Wong, John McFarlane?

- Definitely games related:
  - C++ for cross-platform VR development
  - Testing Battle.net (before deploying to millions of players)

    - The current memory and C++ debugging tools used at Electronic Arts
    - our own SG14 talk
- Probably interesting to games developers:
    - Live lock-free or deadlock (practical Lock-free programming)
    - Reflection techniques in C++
    - (Maybe?) Cross-Platform Mobile App Development with Visual C++ 2015
    - How to make your data structures wait-free for reads
    - C++11/14/17 Atomics the Deep dive: the gory details, before the story consumes you! (by Michael)
    - C++ Atomics: The Sad Story of memory_order_consume: A Happy Ending at Last?
    - C++ in the Audio Industry
    - 3D Face Tracking and Reconstruction using Modern C++
    - Implementation of a component-based entity system in modern C++14
- Probably less interesting to games developers:
    - C++ Multi-dimensional Arrays for Computational Physics and Applied Mathematics
    - CopperSpice: A Pure C++ GUI Library

Monthly.
Next call Sept 9th, 2-3Et, 11-12Et

2.2 Assign proposals for writeup and form small subgroups

- flat_map:sean; Michael Wong
- fixed point: John Mcfarlane, Guy Davidson, Michael Wong.
- uninitialized algorithms: Brent Friedman,
- string stuff?:sprintf, streams, are all problematics, cant concatenate them, building strings

may not be relevant ?
2 fold- games have text, in multiple languages, use formatted strings, string copy is 70% of runtime
sso?
may be related to flatmap, file names, utf8 processing
2 types of strings,1. on screen 2. identifying asset
logging library, performance impact on these

Billy: start a reflector on Jens paper

- rolling queues: Guy Davidson, have github, a missing container, implementation dec/list consumes memory, may be able to get a paper together by Sept 9, circular buffer is too heavy and not thread safe, and not quite what we want
- intrusive containers?: Hal Finkel is interested could lead the group

EH costs: Patrice Roy; Greg Bretwell
Compare virtual function and see if a class has implementation or not: Scott Wardle, user

override an implementation, sort containers based on their virtual functions, in C sharp, unity is using this a bit, want to use it in C++: Scott Wardle:Brent Friedman

-
- other things?

3. Any other business

GDC submission Aug 27

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)
Billy Baker: Jens string paper
All: section 2.2 leaders lead discussions on reflectors and write possible papers for Sept 9.

5. Closing process

5.1 Establish next agenda
Review paper writeups before Sept 11 paper deadline.

5.2 Future meeting
Next call : Sept 9

Aug 12: First call
Sept 9: Before CPPCon call; Sept 11 Paper deadline for SG14 F2F CPPCon
Sept 23: SG14 F2F at CPPCon

Minutes for 2015/09/09 SG14 Conference Call

Meeting minutes by Michael

1.1 Roll call of participants
Michael Wong, Brad searl, John McFarlane, ville Voutilanen, Billy Baker, Nicolas, Sean Middledich, Mike McAughlin, Brent Friedman,

1.2 Adopt agenda
Approved

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

Approved.

1.4 Review action items from previous meeting (5 min)
1.4.1. All: Consider attending SG14 meeting at CPPCON 2015 hosted by ISOCPP.org Wed. Sept 23, Meydenbauer Center, Redmond, WA

1.4.2. All: Consider attending Kona Meeting in Oct 19-24, 2015.
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4544.htm

1.4.3. All: consider attending SG14 meeting at GDC 2016, March 14-18 hosted by SONY

2. Main issues (50 min)

2.1 Review SG14 logistics, discuss CPPCON Games track, C++ Standard meetings , SG14 mailing deadline (Sept 11), C++ Kona mailing deadline (Sept 25)
single unified SG14 library does not make sense. better to have smaller repositories,
flat_map means we have to maintain build system

linking to each person's github repository

2.2 Review proposals discussed so far and leaders. We won't have time to discuss in detail, but we can go over major questions and seek opinions and feedbacks, logistics, and how to prepare the paper for the SG14 deadline.

- flat_map:Sean; Michael Wong

mostly intent paper, exception safety/guarantees, alternate implementation, boost style, sorted array flat map, pairs vs splitting the keys and values,
collection of containers/adaptors
how is it different from existing map/set: stored in a single contiguous block memory/better performance

will request number

-
- fixed point: John Mcfarlane, Guy Davidson, Michael Wong.

will request number

- ptf colony, Michael to track down paper author to see if they want to submit that proposal

-
- uninitialized algorithms: Brent Friedman,

have implementation for initialized move, destroy, all the algos that we discussed, have papers for that; and for raw memory iterator; and remove algo that don't maintain stability guarantee

3 separate papers to be reviewed.

-
- string stuff?:None yet, but Jens Maurer already have a paper on it
- rolling queues: Guy Davidson,

check with Guy on whether he is ready to submit

-
- intrusive containers?: Hal Finkel is interested could lead the group
- EH costs: Patrice Roy; Greg Bretwell
- Compare virtual function and see if a class has implementation or not: Scott Wardle, Brent Friedman
- thread safe STL: Brett Searles

still working on the paper , looking at the locking mechanism; multi-threading and handling, hopefully at
request a number

3. Any other business

License:
ISO rules requires must be usable for standardization
MIT or Boost or UIUC

fast string formatting for logging error dialog; SG14 work on an alternative string formatting library like type-safe printf; Sean been experimenting with it.
Jens has similar for it.
michael will connect

genenral discussion on goals: adding direct support accelerators and financial low-latency as common intersection of interest.

experimental fundamental TS has source informaytion capture which is very useul

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 meeting: Sept 23rd at CPPCon
Next call: Oct 14 just before Kona meeting

Aug 12: First call
Sept 9: Before CPPCon call, review whats been discussed on the reflector
Sept 11: SG14 CPPCon mailing deadline
Sept 23: SG14 F2F at CPPCon
Sept 25: C++ Kona meeting Mailing Deadline
Oct 14: next telecon just before Kona.

Minutes for 2015/09/23 SG14 CPPCon Meeting

Special call out to the scribes
Billy Baker (Flight Safety)
John McFarlane (Zoox.com)
Guy Davidson (Creative Assembly)

Note that they did a hard job under less than ideal listening conditions, and if the notes are incorrect, or missing, please offer corrections to me, and I can update it on the wiki, which unless you are a committee you cannot see. So I am inlining the notes here.

We weren't able to get through all the items, but managed to review 9 papers and one major discussion. The remaining papers and items will be discussed in future telecons. Thank you.

http://wiki.edg.com/twiki/bin/view/Wg21kona2015/CPPCon2015SG14Meeting2015-09-23

CPPCon 2015 Agenda

1. Opening and introductions

- Attending SG14 at CPPCon and why we charged $25:
  - http://wongmichael.com/2015/09/02/why-are-we-charging-for-sg14-games-devlow-latency-meeting-at-cppcon-2015-and-how-you-can-get-in-for-free/
- Reflector
  - https://groups.google.com/a/isocpp.org/forum/?fromgroups=#!forum/sg14
- Code and proposal Staging area
  - https://github.com/WG21-SG14/SG14
- General Schedule (matching CPPCon break paid and provided by ISOCPP.org; Thank you to ISOCPP.org): Tentative
  - Start: 8:30
  - Review papers: 8:30-10
  - Morning break 10:00 – 10:30
  - Suspend for keynote: 10:30-12:00
  - Lunch 12:00 2:00
  - Discussion topics: 2:00-3:00
  - Afternoon (drinks only) 3:00- 3:15
  - Discussion topics: 3:15-4:15
  - Afternoon: 4:15 – 4:45
  - Discussion topics: 4:45-5:30
  - End: 5:30
- Room
  - Room 409 Joliot-Curie Meydenbauer Center

Opening remarks about how the wiki works

No read only access

VV: SG members usually have access, but are expected to behave

1.1 Roll call of participants

Protected list:
https://groups.google.com/a/isocpp.org/forum/?fromgroups=#!topic/sg14/dN4MF_vf85I

- MW Michael Wong
- VV Ville Voutilainen
- BB Billy Baker (scribe)
- LC Lawrence Crowl
- PM Paul McKenny
- PR Patrice Roy
- DS David Sankel
- JS Jeff Snyder
- BR Brent Friedman
- NG Nicolas Guillemot
- SM Sean Middleditch
- GD Guy Davidson
- JM John McFarlane
- SW Scott Wardel
- JFB J F Bastien

(lots of other individuals interested in gaming and low latentcy)

- Isabela
- Michael
- Alex
- Shane
- Andre
- Teddy spot trading
- Omar blizzard
- david disney
- Paul wargaming
- (missed name) Funkytron
- Vishal (end of row)
- Sunil sony
- brian monolith
- Charles google
- Victorio
- Vishal
- Koray sony
- Art Eagle trading

- David
- Marcus EA
- Bob verisign
- Walt AMD
- Arthur
- Michael google

1.2 Adopt agenda

SM: do we want to add an item about branches and memory prefetch control

MW: do we have some one to lead the discussions

no one volunteers

MW: need someone for intrusive containers

GD: what does leading a discussion mean

MW: describe the issue, advantages/disadvantages, then open for questions and answers

VV: may have to provide the answers

BF: knows enough to present unlikely/likely (conditional statements)

MW: anyone for intrusive containers

GD: will lead intrusive containers

MW: Shane for banking applications

Victorio: have a draft for bitset operations

SW: has a virtual functions comparison paper

MW: add bitset and virtual after thread stack at creation time

Isabela: function multi-versioning as a discussion topic

MW: no objections to agenda

MW: feedback from today needs to be incoporated into revisions for submission on Friday for Kona

JM: will the minutes have feedback

MW: yes, but they may serve to jog memory only and you may need to take your own

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

1.4 Review action items from previous meeting (5 min)

1.4.1. All: Consider attending SG14 meeting at CPPCON 2015 hosted by ISOCPP.org

Wed. Sept 23, Meydenbauer Center, Redmond, WA. Done We are here

1.4.2. All: Consider attending Kona Meeting in Oct 19-24, 2015.

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4544.htm

1.4.3. All: consider attending SG14 meeting at GDC 2016, March 14-18 hosted by SONY

2. Main issues (all day)

2.1 Review SG14 logistics, discuss CPPCON Games track, upcoming C++ Standard meetings , Github and reflector locations, C++ Kona mailing deadline (Sept 25)

GD: should someone else take over the github repository or assist

JM: there has been discussion about changing the existing repository to an organization

MW: not setting any policy, if you want your own repository fine, or you can use the existing SG14

GD: keep me posted if there is a problem

MW: go ahead and change to an organization structure so that others can approve

2.2 Review proposals.

- Fixed point real numbers: LEWG, EWG, SG14, SG6: D0037R0,
- C++ Binary Fixed Point arithmetic: Sg14 SG6: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2012/n3352.html

https://github.com/WG21-SG14/SG14/blob/master/Docs/Proposals/Fixed_Point_Library_Proposal.md

MW: would Lawrence like to present his paper

LC: will present

JM presents D0037

JM: Ongoing design question for arithmetic operations on fixed_point type.

Art: what is the rounding policy based on "Internals" slide?

JM: precision is lost at construction of fixed_point when converting from double to a fixed point value stored in a uint32_t

Charles: common for adding a "half", in this example 2^-30 would be added

JM: extra runtime operations?

Charles: common for hardware when converting from double to integer

crowd comment on subtracting if value is negative

JM: aiming in the future for template specialization for common integer types and exponents that are more hardware related

Charles: intermediate representation may be different during multiply/square, correct?

JM: yes, there is some doubling of storage size for multiply, it does cost a little more

JM: believes that GCC has some optimizations in this area

"Open issues" slide needs a change: numeric_types -> numeric_limits

Funkytron: will there always be rounding issue during initialization

JM: always

Funkytron: could there be an initialization with the bits that you want to address rounding, say pass in two 8 bit values

JM: sure, but there will be rounding somewhere

Arthur: to be clear, this is not decimal

JM: correct, this paper would be different than what banking would want

Lawrence presents N3352

A newer revision is in work

No presentation of details since SG6 has changed the details

David: is b_r a different type than b.r

LC: yes, there is work done during compilation of the template parameters to create a new return type

Charles: how do the types deal with a for-loop

LC: an accumulation, correct

Charles: yes

LC: when you accumulate, you get an assignment and thus a conversion

Arthur: that kind of means you have to pick an end representation

LC: if you do a difference, high bits will cancel out

LC: you would an explicit type rather than auto, the division would do the appropriate conversion

LC: the source and destination types are know and the rounding is

SW: you show a cast, is it always needed?

LC: this is creating a fixed point number

SW: I am assuming I could just add in integers if I didn't want to renormalize

LC: remember we are keeping all of the information, shifts usually only occur when doing a conversion except during division

LC: no shifts for multiplication unless there is different resolution

SW: common to use 16.16

LC: in this scenario, you would get a 32bit number

LC: if 16 bit coming in from a 12bit A/D convertor, then there are

PM: suppose you have a loop with auto usage, do you get overflow

LC: auto outside of loop, range and resolution is fixed

PM: range and resolution is fixed

LC: yes

JFB: what would be the default for the assignment?

LC: the default for rounding is near_odd, it is about the most expensive with the best results

LC: if you don't want to care about rounding, you should get the best one, if there is a performance issue, then you can change

JFB: could we have a named function that specifies the intermediate type?

LC: don't have that, not sure it is necessary

LC: it might be possible for compiler to figure out that you are doing 32bit add but throwing away the bits

LC: don't think the approach would be necessary

SW: thinking there would be cases where 128bit integers would be needed

LC: yes, once overflow then there would need to be a change

LC: the type is bound, there is another type in SG6 for unbounded type

SW: what about decimal types

LC: it has been through the committe once, there is no paper for decimal floating point

Vishal(end of row): can think of situations for decimal fixed point

LC: if care about decimal fixed point, then you care about the laws of rounding

Arthur: names were a little confusing, it would be nice if there were some notion of signed and unsigned

LC: believes that the SG14 presented paper should move to SG6

SW: originally didn't see how shifts would be avoided in N3352, but now sees what N3352 is doing

SW: the reason for fixed point in gaming is to get the exact results on two different CPU architectures

JFB: It is not obvsious that you get the same result on two differnt architectures from these papers

SW: also want to be fast

LC: don't imagine that there would be anything proposed different from that

MW: think we need some data

JM: can issue with code generation be addressed by showing the integer operations

JFB: sure the standard won't show assembly

JFB: discussion relating the performance of fixed_point to memory_order_consume

JFB: standardization of this would be pointless if you don't get the assembly that you want

MW: don't think we need a vote here

Vishal(end of row): would there be any benefit for this being a language feature

LC: the place where language support would be nice, would be low-level such as rounding right shift

LC: SG1 has approved an operation for such, then hardware vendors would have to make sure it is present

JM: less undefined behavior for shift operations would be nice

MW: you can always create a paper

Arthur: if you ran the alpha blend example through a compiler for both papers, then you can check the generated code

LC: but you get different results, don't think you are going to be able to put up two

DS: comment that users of this functionality really care about the generated assembly which isn't necessary the concern of the committee

Art: would using a rescursive call rather than a loop for a

Alex: gaming won't use these libraries if there is not a one-to-one mapping between what is coded and the assembly

Alex: if an add instruction is not generated for a fixed point add, then it won't be used

Alex: be aware of your target audience

LC: if the resolution is the same, then it should be one-to-one

JM: the same should hold for my paper

MW: (to LC) please send an updated paper to SG14

- Flat Map/Container design: LEWG, SG14: D0038R0, https://github.com/seanmiddleditch/CPlusPlus/blob/master/flat_containers.md

Sean M's proposal: Flat Container

SM: Does not propose additions to library; it's an intent paper intended for feedback. SM: Overview of FC. flat map is often mentioned. Trying to stick with FC instead. They are single - or mostly continuous - chunk of memory. # of allocations reduced. Array or vector is sorted by key and then binary search is used to search. Many existing examples work this way. Good for memory constrained devices. Also /can/ help with cache locality and therefore speed. SM: However, many ways to implement. Open question: Is this allocation strategy the best way? Jumping into middle of array can cause a cache miss. A more 'heat based' approach is to keep hot stuff close to the beginning of the array. Question: should we explore this approach.

Q: What size of allocator? Backed by vector. SM: Question: should there be an adaptor or a new type of container? Experience suggests container. Have tried adaptor. It worked. Adaptor might be a good approach. Similar performance but with more control.

Q: Semantics may change. If vector is underlying, iterators might be made invalid.

SM: Interface **will** invalidate iterators. But necessary to allow vector. Flat container is ordered. Important for some set functions.

Q: What if someone uses list? Would that adapter fail at compile time? Non-random access container as backing.

Patrice: Yes, can check this at compile time.

Q (Bob): Depending how you store elements, could be cache sensitive for lookups versus in-order traversals. Could there be a template argument policy for engine which caters to one or other based on user choice?

SM: Yes, that could happen.

Ville: Does boost give you this?

(Bob): Don't know.

Scott (EA): mass insertions then sorting afterward is quite common (delayed insert). EASTL may have a version.

SM: Our versions do this. Don't think boost offers it.

Q: Concern about adaptors. Allow direct access and invalidate invariants. It would help with delayed insert so there's a design decision there.

Ville: depends on the adaptor. Protected members prevent this.

Q: Delayed sort breaks invariants.

SM: No, can store size of previous (sorted) versus new (unsorted) entries.

Q: So the exception question WRT this?

SM: I may be making mountain out of mole hill. There are cases where exception can be thrown during move or copy where operation is unrecoverable. At this point rollback isn't possible. Cannot continue. Can still recover to clean up but container is in bad state. What does everyone think?

Zack: With sorted vector, in-order traversal order is nice. With deque branch is costly in iterator during traversal.

Matt K: Re. sorting policies, if intent is associative container with cache-friendly fast traversal, then maybe separate keys and values.

SM: That's in 'reference proxies' section. Specifically for map, instead of iterator returning the standard pair, instead return pair of references. This gets into proxy iterator problem? Scott Wardel has paper exploring this. I have tried separation but has impacts on some algorithms.

Q: Re. exception guarantees, basic guarantee means invariants are unbroken. Do we get that here?

SM: Yes, we can clear the container. Yes, it's undesirable though. When this data is vital, system is corrupt at larger scale. I'm not expert on this.

Q: So no valid program gets into invalid program state. Then, what does invariant mean in this situation? Solution? Other options such as program exit?

SM: Flat container can only deal with its own state. Best it can do. In case where more allocation not needed, just shifting around is sufficient for this to be a problem.

Q: So re. exception safety. In game code, often don't care about exceptions. Turned off entirely. So then this not a problem. So can we say noexcept is a requirement?

SM: Too many standard types no not have noexcept, e.g. std::string. So that not realistic. For games exceptions turned off entirely. So general guideline is make move `noexcept`.

Q: Basic guarantee, solution destroy objects up until the end and update bookkeeping.

SM: Yes or clearing. Either way we lose data.

Patrice: If you raise exceptions, you live and die with it.

Q: If you instantiate vector with T that doesn't throw on move, you get different performance patterns. In case where it might throw, there is the more complex code path. Guidance is always nothrow on move.

SM: Vector is different case to map or set because of assumption concerning no duplicate keys. Specifically, here when shifting elements we have the issue.

LC: If elements are going to throw, can guarantee container won't either. Otherwise, you have to empty container. Implementation will have two variants.

SM: Yes that or clear only remaining elements.

Scott (EA): When adding / delayed sort and grow occurs, you can mostly get it to work with strong guarantee. Correct?

SM: You can use swaps all the way down.

Isabella: Case where - with effort - can unwind the entire operation.

David: "Quality of Implementation" so long as we keep guarantee, this is a matter of degree. Lets not do too much work on this.

LC: Don't want to put out standard that gives implementor full reign. Say that "State is 'not usable'" when this happens. Otherwise, variation between platforms lulls user into false sense of security. Expectation important.

Scott (EA): Interesting: implementation deals with trouble of splitting hash from references and having two different lists. Is this going to be solved? If so should we have this yet?

SM: If we optimize for cache hits, this really matters. Eric Niebler will get far with standards. If proxy iterators are well supported we should give implementors the freedom of Proxy approach. Open question.

Q: When content cleared. Content lost. Keep track of lost data and allow user chance of recover.

SM: Thought about this. There is no precedence in standard. No idea what this would look like.

Zack: Re splitting keys & values: many say painful to iterator over. They'd rather just the sequence of values. Are we overloading map if we do this? A related name? Something with different name to map.

SM: We have begin end iterators that return proxy. Works for general maps. Possible to cater to both use cases, i.e. not necessary.

Q: 2 options split and not split. With separated key and value, insert and sort changes exception guarantee.

SM: Backing algorithm important. Affects iteration efficiency. If you want fast look-up, you want ordered container. Would like to research trade-offs and find out what demand is.

Q: Suggest: separate strategies and separate containers. These are not the same thing. Different interface, different goal therefore different container.

LC: Have application with need for little hash tables without the overhead. Have never needed to iterator over elements. So not an overhead I want to pay for. Maximum size fixed on creation is a possible design option. There is space of implementations we need to map out here.

SM: Yup. We need numbers and use cases.

Q: Look-up time when playing a game. That'll happen more.

SM: Depends on use case. Some cases I never call find.

Q: Have similar: flat multi-map. Allows for sorted region plus newly inserted, non-sorted. Begin/end invoke lazy sorting.

SM: Done similar thing. We retired the on-demand method because mutable state made MT a problem.

MW: Lots of design suggestions. Hard to vote with all this feedback. So, vote: should it be a container or an adaptor?

Vote for moving this paper forward... Strongly for: 2? For: 19 Neutral: 10 Against: 2 Strongly against: 0 (nobody dies)

Q: Explanation from people against?

1: Not found a case for this. If I have data to iterator over, use vector. If look-up by key I use map. This fills narrow intersection in my experience. 2: Like idea, but not generic enough. Want choose between lookup and iterate. Resizable storage should be compile-time option. Paper needs to be more generic.

Conclusion: Goes through to evolution.

Ville: Not an evolution working group matter.

Patrice will champion this.

SM: regrets cannot attend Kona.

- Extending raw_storage_iterator: LEWG, SG14: D0039R0, https://github.com/WG21-SG14/SG14/blob/master/Docs/Proposals/rawstorage.html

BF: Raw storage iterator already exists. Has problems. Want to fix them. Constructs objects at address of iterator. Problem #1: doesn't support move - only copy.

Ville: Library issue 2127 aims to fix this and committee will do this.

MW: Add your discussion to 2127.

BF: Factory function takes two template parameters. Intent to make easier.

BF: Doesn't actually support placement new itself. Odd case but we ought to support this.

Ville: Becomes in-place iterator?

BF: Yes.

Isabella: Implemented this this morning. Forwarding assignment found to be easier. Do this in combination of transform. Easier to use std::copy and std::transform. Should be assignable from type.

BF: Examples?

Isabella: Can't disclose use cases. Much easier to convert from raw storage. std::copy would be easier.

BF: Use that as emplacment. That should maybe be separate interface / type. Worth consideration.

MW: Feedback will ensue.

- Extending memory management tools: LEWG, SG14: D0040R0, https://github.com/WG21-SG14/SG14/blob/master/Docs/Proposals/rawstorage.html

BF: We make our own allocators. Lets make it easier. . Concerns raised: rename destroy to destruct? Stepanov set precedent there. value_construct and default_construct are also uncertain terminology. make_shared noinit is a different proposal that might share terminology with this. move_if_noexcept also a concern. Dinkumware has uninitialize_move so there is precedence here.

Isabella: Suggests there could be a transform iterator.

With no objections, paper goes forward.

BF: Also unlikely to go to Kona.

Vote on raw storage: no objections. Approved also.

- Unstable remove algorithms: LEWG, SG14: D0041R0, https://github.com/WG21-SG14/SG14/blob/master/Docs/Proposals/unstable_remove.html

BF: Existing algorithms are explicitly stable. This alternative can be much faster. Assembly examples provided (in paper) with different variations. This approach moves instead of swap. Unstable_erase might be another direction of many this could go in. Basic algorithm only covered here to save time.

Scott: You swap from end?

BR: Almost, we move from end.

Scott: EASTL has something related.

Jackie: URL for performance test links to repo.

BR: Yes, they're in the repo. Under SG14_tests directory.

MW: No objections. (Note: Unstable_remove and unstable_remove_if only are carried.)

MW: Clarifies Lawrence moves fixed_point paper. Assumed to be approved.

- rolling queues/ ring: LEWG. Sg14: D0059R0,https://github.com/WG21-SG14/SG14/blob/master/Docs/Proposals/RingProposal.pdf

GD: std::queue causes fragmentation problems. Fixed-size <= 1 allocation. Why 'ring'? circular_buffer already in boost, quite different. Rolling_queue looks strange. Cyclic_buffer and ring_buffer have "_buffer" and this is an adaptor. Looks like std::queue. Exceptions: push and emplace may fail because ring is full. You might not care and be happy to overwrite older entries. Open suggestion: aggressive_push and aggressive_emplace.

GD: vector allows decide ring size at run-time and array at compile-time. E.g. with vector, items only created when actually used in ring. std::array -> static_ring and std::vector -> dynamic_ring.

GD: In many places already use ring. When first suggested in discussion group, 6 implementations were submitted by other members.

GD: On to the paper. Synopsis not complete for dynamic_ring.

Scott: This is a wrapper right? Should array be a parameter? Why different classes? Why not array and vector be type parameter of a single class template?

GD: One reason instantiation an issue. Reason two: not knowing size of vector

LC: Another proposal submitted. Far more concurrency-related version. Paper: n3353

MW: Has to deal with concurrent erasure which is tough problem. Advises GD to look through other paper.

LC: Doesn't expect them to be unified.

GD: Purpose of this proposal is to avoid allocations.

LC: But there might be cross-fertilization and they should be distinct.

Q: Why not use perfect forwarding to pass array/vector to a unified class template c'tor? And what kind of iterators are they?

GD: FIFO iterators. No plan to run iterator over queue. (Ones in synopsis are boilerplate / internal.) Aims to make interface the same as std::queue.

BR: std::array can be expensive to move. So maybe not such a great idea.

Scott: Is this connected to discussion of fixed-sized containers?

GD: Both of these meant to be fixed sized. The names, dynamic_ and static_ are no concrete design decision currently.

Q: Buffer overrun: map has try_emplace(?) perhaps try_emplace here

GD:

Bob: fixed_ring instead of static_ring.

GD: Yes.

Bob: Lifetimes using c'tor / d'tor (?)

Q: Way to construct dynamic_ring from fixed_ring? E.g. when run out of space with static_ring?

GD: Not implemented. Good idea.

Q: Recommend don't bikeshed names. The library working group will do all of this. They know this stuff! You can actually leave things unnamed and they will just take care of it.

Authur: Retracting uninitialized storage suggestion. How do we do this with vector and array. How does construction happen.

GD: static_ size starts ready-constructed and dynamic_'s elements get constructed as they are pushed - just like the two rings' underlying storage types.

Authur: How to tell empty / capacity?

DG: Can look at iterators. No capacity.

Authur: so for static_ is emplace a misnomer because array elements are around all the time.

GD: Popping doesn't destroy. Was tried. Was a mess. The details of conversation with Jon Wakely will be added to paper.

SM: Ranged-based creation or assignment iterating through all the objects would make it easier to convert from static_ to dynamic_.

Charles: More concerns. Uninitialized buffer instead of buffer so c'tor / d'tor called - same as for dynamic_ suggested. By extension maybe this isn't an adaptor.

Q: If you have a container that manages this, you can still keep it as an adaptor.

GD: Maybe deal with this with template parameter on static_ring.

Ville: Non-assignable elements are the reason for need for emplace in interface.

Q: If I just have allocated memory, I feel I should be able to use that with this adaptor. Could you use array_view and make that the adaptor? Would keep ability to allocate everything at once.

Ville: Moving the container into the ring is another option. Then no double-allocation.

GD: Both have container move constructor.

Ville: So already possible to avoid double-allocator.

Q: Reiterates idea of passing a block of memory.

Scott: begin and end might be tricky? The situation when begin and end are the same.

GD: ring is FIFO. No intention to iterate through the buffer.

Q: Could ring be more like a ring?

GD: Not thought about it.

MW: Suggests Arthur work with GD on this.

*Ville: non owning view is very different trade-off. Passing by values then poses different set of problems. And when you return them by value - **very** different set of problem.*

Q: Could static_ring be composed of container and view?

MW: Now potentially looking at new proposal.

GD: Happy to develop this paper in response to feedback. Rather not turn this into a swiss army knife that gets turned down.

JM: Suggests static_ring and dynamic_ring don't deal with object lifetime within interface.

MW & GD discuss how to proceed given many suggestions.

MW: Revise before Friday but vote in the mean time about whether we 'think' we'll like.

Ville: Yes, can cast general 'up'/'down' vote that suggests whether if warrants continued development - as opposed to voting on passing it right now.

MW: We could even work on it today and vote late today.

'Up' / 'Down' vote shows support by show of hands. Guy to continue to work on it until Friday.

- *thread-safe STL: LEWG: SG14: D0064R0*

Brett: In MT environment STL, you get race conditions and other issues where multiple threads are involved. Proposal locks on STL containers by communicating with OS.

MK: There are different types of lock. What is meant by auto-locking.

Brett: Talking generally about locking.

Q: Maybe two ways to do this: use atomic when adding.

Brett: With atomics, how do other cores know about need to lock.

Q: ... or with push_back / pop_back use atomic use a counter. With expansion of vector use mutex and prevent reading / writing to communicate invalidation of existing memory. Or keep track of iterators and update all iterators to point to newly allocated memory.

Q: Does this introduce a function into the library that does this.

Brett: This is early stage. Yes, would cover thread management generally.

Q: So helping to make threading easier?

Brett: Yes.

Gor: Develop interface or special-access containers?

Brett: Interface.

MK: Should probably be divided into other proposals.

Gor: Suggests use cases you might have which might guide the design of the proposal.

Q: Suggestion: post in proposal mailing list.

Work to continue on this proposal.

- *Controlling Thread Stack Size at Creation Time: LEWG, SG14: http://h-deb.clg.qc.ca/WG21/SG14/thread_ctor_stack_size.pdf*

Patrice: Idea to be an alternative std::thread c'tor that takes another argument that is stack size. Other parameters can be fixed when we create threads. These are scoped out for brevity and they are quite specific.

Q: Available on every major platform?

Patrice: Yes.

Q: Could it be a suggested stack size.

Patrice: Minimum size or exact size? Audience might want exact size. Would probably be a lower bound.

Isabella: Many platforms wouldn't let you set stack size - esp. with virtual / abstract machine. Not clear that this could be done on all platforms.

LC: Saying that C++ has a stack traditionally opens up a can of worms. Suggests tabling.

Patrice: Clients ask for this.

Q: Could we avoid saying what a stack size actually is?

MK: Stack size is meaningless. Different platforms will make the same stack size go further.

Patrice: Understand need to add detail to this.

Ville: Easy solution between this discrepancy: Specify as implementation defined OR unspecified and then rely on QoI.

Patrice: Would be a start.

Q: Possible to pass in native handle?

Patrice: Already need to have created thread. Another option, a factory fixes nothing.

Q: Instantiated in thread code?

Patrice: No just additional argument for now.

Gor: Mechanism for additional attributes of which stack size one?

Patrice: Looking into something like this. Will end up very platform specific. This proposal is something shorter-term.

MK: Dumb suggestion: What if we could provide the memory for the thread.

Isabella: Boost context allows this.

Gor: Confirms boost context has this functionality.

Scott: What about thread name?

SM: Tricky WRT debugger

Ville: Passing a name / handle: brings back horror of creating threads with other APIs. No nice easy forwarding of arguments. Because thread created afterward. Horrid.

Q: Sticking required stack size in now, people will look back and ask why random stack size parameter is here.

Patrice: Summaries feedback he needs to take in.

Q: Why not call it suggested stack size?

Patrice: because we want a 'needed' amount. If we don't get it, we want to fail. E.g. in embedded system.

Ville: Standard cannot require that parameter is obeyed.

Patrice: Would count as not enough resource.

Q: E.g. vector::reserve is a bane because it is not explicit.

Patrice: Clients want exactly this amount.

no vote cast

- *bit set operations: LEWG, SG14: https://github.com/SuperV1234/cppcon2015/blob/master/sg14/bitsetSubsetProposal.md*

- *compare virtual: EWG, SG7, SG14: https://github.com/WG21-SG14/SG14/blob/master/Proposals/ComparingVirtualMethods/Docs/ComparingVirtualMethods.docx*

2.3 SG14 Goals and Scope and general discussion

2.3.1 RTTI and Exception costs (Patrice Roy, Alex Rosenberg, Gaby, Bjarne)

http://h-deb.clg.qc.ca/Sujets/Developpement/Exceptions-Costs.html

Patrice Roy Exceptions: good, bad, common question.

Wrote some test cases, checked them in several compilers, several scenarios

Code available on website

Exceptions behave badly when not handled, consuming time, even when they are not thrown.

Similar results on GCC, MSVC 2015, although GCC performs better when exceptions are not thrown.

Error codes win the race against exceptions.

Laurence Crowl believes it may be due to MS asynchronous exceptions

Questions about EHsc or EHa flag in MS, not answered.

Sean Parent: have tests been done on code that has no throws? PR will check.

Should do more checks on non-throwing code.

Stack unwinding: flat cost for exceptions, but return code checking for simple functions. However, with a deep sample (31 frames) exceptions still underperformed

Using a longer string MS compiler with exceptions was faster.

Using vector identical results to string with SSO

Binary sizes increase with exception handling, 1-3%

Graph summary available on PR's website.

Small difference was suggested to be down to the small size of the test program: the bulk of the code is runtime. It is difficult to get a true test of the impact without instrumenting a game.

Perhaps Quake or any open sourced game could be tried for recompilation with and without exception flags.

Laurence Crowl: is the objection quality or consistency? PR believes it's absence of use rather than anything else.

Thanks from MW.

Alex Rosenberg This is not a question, it's just off by default. Yes in tools, unit tests etc. but not in shipping games. Perhaps more detailed EH switches are appropriate. Decorating with noexcept is a heavy burden. How do we start the conversation?

Green Hill ex-employee: —noexceptions mode made linking tricky. You can't mix code from the two sides of exception using code and non-exception using code.

Sean Parent: this seems to be a library problem, not a language problem. Have we looked at seeing how to lighten the weight on the library side? STL has nearly no exceptions, other places are overly-generous.

AR: Library has increased its dependence on exceptions which is why it's unpopular in the games industry.

Sean Middleditch: Exceptions prevent use of other libraries besides std. Drives down participation.

David: So what is it you want?

AR: Start the conversation between the games industry and the committee. Address the places where the library depends on exceptions and spread from there.

VV: Make dependence optional? General consensus and agreement.

PR: Constructors throwing is a problem, relying on return codes is usual.

Anon: all compilers allow exception disabling, varying between vendors. Should the standard dictate what it means to turn off exceptions? SP: Jumping to a solution; identify the problem properly first. We should be looking at the reasons for the cost and are there other solutions? Are there things we can do to lighten that load? Additional compiler knowledge could fix things, e.g. knowing where to expect throws.

Otha(?) Code is slow because compilers are dumb. Compilers are improving. Standardising vendor practice, e.g. undefined behaviour rather than throwing.

SP: Compiler specification needs to be richer to remove the exception call chains.

Otha: That's a long way away, >10 years.

John Macfarlane: There is plenty of scope for standard library use with EH turned off. Allocators work. Boost builds without EH. Library vendors do seem to accommodate no EH.

SP: Vendors need to be sure that they know that throwing exceptions can mean terminate.

Vishal Oza: Is returning codes costly anyway? Do devs assume the golden path always anyway?

SP: clean execution will run faster.

Anon: His last game shipped with exceptions on purpose.

David Ludwig: Boost's date/time does not, as far as I've been able to tell, utilize error_code. Boost's "chrono" library does, though (along with filesystem, as cited in the meeting, and a few other sub-libraries).

Isabella Muerte: Filesystem has an error code approach which specifies how errors should be registered.

Vittorio Romeo: How about a nothrow naming system like with operator new?

IM: Calling terminate rather than throw could be an option.

David (not Ludwig): Formalise the idea of a no-exceptions mode which creates different meaning for the standard library.

VV: This doesn't need to go to plenary!

Otha: Macro solutions means nobody can ever write "throw" again

Zach: likes the error category pattern, but there needs to be a standard way: no exception is standard on embedded as well as in games.

Gor: null or kill when malloc fails? AR: kill, overridden new should have options.

IM: Containers need knowledge from the allocators that they need to call a recovery function in the event of an allocation failure.

PR: When would they be fast enough to be acceptable? AR: Complexity is a problem. Sunil: It will never be zero cost. Would like to use that cost elsewhere.

Solutions: Lighten the load from the library. What are the throw points and how can that be mitigated. (Guy) throw = UB, catches are magically gone. (Brent) thread local handler for cases

where exceptions might be thrown (IM, kills it) error category, system_error, as used by filesystem and networking (Guy) throw = quick_exit (Brent) end_exception definition (David Sankel) make noexcept(true) the default (Sean Middleditch) function decoration to assert non-throwingness, optimising accordingly (Sean Middleditch) standardising industry practice e.g. — noexception (Michael Spencer) tagging a class or namespace or module as noexcept (Sean Middleditch) error or exceptions (Gor Nishanov)

2.3.2 direct GPU/accelerator-based proposals (C++AMP/OpenMP accelerators/OpenACC/Vulcan) (Michael)

2.3.3 coroutine and games (Gor)

2.3.4 intrusive containers (Guy)

2.3.5 allocators (Guy)

2.3.6 help better support latency in financial/trading/banking applications (Shane)

2.3.7 likely/unlikelky (brent)

2.3.8 function multiversioning (Isabella)

3. Any other business

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 (past and future calls)

- *Next call : OCT 14 telecon*

- *Aug 12: First call*
- *Sept 9: Before CPPCon call, review whats been discussed on the reflector*
- *Sept 11: SG14 CPPCon mailing deadline*
- *Sept 23: SG14 F2F at CPPCon*
- *Sept 25: C++ Kona meeting Mailing Deadline*

- *Oct 14: SG14 monthly telecon*
- *Nov 11: SG14 monthly telecon*
- *Dec 9: SG14 monthly telecon*