Document Number: P0938R0
Date: 2018-02-12
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 2017/12/13-2018/01/10

Contents

Minutes for 2017/12/13 SG14 Conference Call

Minutes by Patrice

Notes, SG14 meeting, Dec. 13th 2017 14:09 EST

John McFarlane chairing

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

Roll call :

Guy Davidson
Ben Craig
Brett Searles
Jan Wilmans
Ben Saks
Mateusz Pusz
Andreas Fertig
Charley
Kévin Alexandre-Boissonneault
Piotr Balcer
Ronan Keryell
Sergey Vinogradov
Tomasz Kapela
Andy Rudoff
Aleksey Kukanov
Paul M. Bendixen
Patrice Roy (no way to speak, but taking notes)

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

Main Issues

McFarlane: in ABQ, some SG14 members were present.

McFarlane: [[likely]]/[[unlikely]] went to EWG, then CWG

McFarlane: it was given wording by Davis Herring and Adrew Pardoe, first, then that wording got worked on in CWG

McFarlane: Chandler Carruth seems to think this could help code generation in many ways

McFarlane: the colony paper did not make much progress, but sparked a discussion. Nico Josuttis has agreed to help champion this paper and provide guidance

Davidson: there is a teleconf meeting tomorrow on that topic. I will participate, among others

McFarlane: the relative absence of SG14 members in WG21 meetings makes presenting SG14 papers a challenge

McFarlane: Matt Calabrese has written a competing paper for one of mine. We have worked on a merger. There was no Numerics meeting in ABQ

McFarlane: wide integers has made it out of Numeric. Some bits-related stuff also seems to be making its way

McFarlane: the Freestanding paper from Ben Craig has sparked a discussion

Craig: a report on how to use error_condition/error_code was not well-received by Beman Dawes. He had issues with the educational side of things, and requested more polish

McFarlane: the constexpr new paper seems to allow baking complex types into firmware, opening up avenues for statically scoping things that traditionally required dynamic allocation. It's exciting for embedded developers

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

Paper Reviews

-=-=-=-=-=-=-

Freestanding

McFarlane: let's start with Ben Craig's Freestanding paper

Craig: I presented R0 in ABQ, got some feedback

Craig: the general goal of this paper is to expose the minimum amount of the C++ library with all the language features

Craig: I'm trying to establish the maximal subset that does not require system calls, space overhead or heap allocation

Craig: I want portable memcpy(), sort()-ing in the kernel, and so on

Craig: I suspect the definition I'm providing might help code destined for GPUs

Searles: I work with GPUs, two different compilers, and we have problem with STL support / compatibility

Craig: this is not trying to solve cross-vendor ABI compatibility. I don't want to introduce incompatibility either

Craig: I plan to avoir exceptions, TLS, heap, floating point, RTTI, and what depends on that. Floating point, in most kernels, requires jumping through extra hoops. Transitioning from kernel mode to user mode can corrupt user mode if one's not careful enough

Craig: I support some conversion functions, those that don't require errno as this requires TLS

Craig: things that depend on string or other kinds of heap allocation are out

(Craig gives a list of "ins" and "outs". See the paper for details)

Craig: there would be things in C++ freestanding that are not in C freestanding

Craig: I could not find a good reason to exclude the wide-character stuff, so it's there

Craig: <random> is one of the higher risk elements of the paper, as the non-float-dependent utilities might be implemented in terms of float-dependent

Roy: can be keep things such as bitset, but without the throwing functions / with these functions but non-throwing error reporting mechanisms?

Craig: the initial version did, but there was a poll in ABQ on selecting some functions but not all in a class and it was no (not a strong no; McFarlane suggests that there might be avenues to change this; Craig says he would prefer going for less at first and getting the paper in)

Craig: I tried to remove <typeinfo>, but typeid would not work

Wilmans: you talk about "the list of functions that could be marked constexpr, and conditionally noexcept", but do not address those items, why it that? it seems indeed very similar and logical to discuss

Craig: ABQ was my first meeting. I did not know how difficult it is to mark memcpy() as constexpr. There would need to be a kind of intrinsic memcpy() to make it work

Craig: the conditionally noexcept might follow the Lakos Rule (wide / narrow contracts and noexcept)

Roy: unique_ptr does not need to rely on heap allocation. Why exclude it?

Craig: because of the default deleter; it introduces a dependency on the heap. I might put it back, but I'm hoping for scope_exit and unique_resource fit that niche in a no-heap context

McFarlane: if something is constexpr, does it go in its favor?

Craig: it's in favor, but it doesn't guarantee inclusion. I do try to reach the maximal subset that can be included without causing problems

McFarlane: excluding <array> is annoying

Craig: it has a throwing member function

Roy: at() should die :)

Craig: I like at()

Wilmans: I like at() too

Craig: I have an initial prototype in the works. I want to iron out issues with <random>, and pull of some (amateur) wording for JAX

Craig: one question asked in ABQ is are there enough people that care about freestanding. In ABQ, there were not many such people present

Keryell: I work with FPGAs. It seems useful to me

Craig: embedded systems developers should speak up if it doesn't work for them

Wilmans: I did not know about freestanding, but for us, we make firmware for embedded boards, is sound really great

McFarlane: you're planning on splitting the paper in "things going out/things going in" to make the non-controversial parts work better?

Craig: yes. I know there has been an embedded C++ proposal in the past but could not find it. Anyway, it tried to ban templates, and I would not do that

McFarlane: make sure you keep a revision history to help people get on track faster

Craig: the html version of the paper has it. The R1 version I'm working on is not complete yet

McFarlane: do you want to get polls on some things?

Craig: I think I had most of my feedback in ABQ

Searles: we're trying to set-up something with universities to show them where C++ can help in embedded development. They ask why not use plain C? Does this paper help?

Craig: this paper shows how and where C++ can help in embedded development

Craig: I keep qsort() and std::sort() as the second one is faster, but if you have many types the first one could lead to smaller code

-=-=-=-=-=-=-

Heterogeneous Lookup for Unordered Containers

Pusz: it's like what we have for std::map / std::set, but we're paying conversion costs when passing a key that requires conversion to the key typeid

Pusz: when inner type is_transparent is aliased to void, user confirms [s]he knows what [s]he's doing

Pusz: it helps performance; we benchmarked it and found the numbers pleasant

Craig: Can the paper include a class similar to your example of string_hash?

Pusz: not for now. I also don't want std::hash to be transparent

Craig: the const char* right now (in std::hash) hashes the address

Pusz: the "diamond operator" from C++14 inspired the is_transparent technique

Craig: suppose I have an unordered_map<short,T> and I pass it key 3 (an int), if I don't provide a transparent hasher, will this affect behavior?

Pusz: as long as you don't provide a transparent hash, it won't affect you. It's strictly opt-in through emable_if. I don't want to standardize transparent hashers for now

Craig: I like the proposal; the workarounds I had for those problems were a lot uglier than this

Wilmans: I like it too

Pusz: I would like to fix the interface of unordered_map in the standard at some point

Pusz: I'm thinking of another paper for lookups based on precomputed hash values. This would extend the interface of associative containers further

Pusz: I plan to present these papers in JAX

McFarlane: this might help people way beyond SG14

McFarlane: did someone want to talk about the Google hash maps?

Pusz: I used a hopscotch map, it was really fast. We used it in production

McFarlane: I think the Google one uses that too

Craig: the "int hash map" in LLVM seems good too

Pusz: this is my first paper. Please tell me how I can make it better :)

-=-=-=-=-=-=-

Enabling Persistent Memory

Balcer explains what Persistent Memory is (memory that acts like storage, and keeps its contents when power's out)

Balcer: it tends to accelerate networking speed a lot

Balcer: the caches remain (mostly) volatile

Balcer: ensuring consistency of data is a problem with such memory (e.g.: memory leaks become persistent)

Balcer: if an algorithm goes through several steps and gets interrupted, it influences what has to be done when coming back. It's similar to multithreaded programming

Balcer: we cannot use regular malloc() with this

Balcer: it's exposed to applications with straight mmap()/msync()

Balcer: we implemented library libpmemobj to manage such memory

Balcer: the library was written in C and is difficult to use correctly. We have implemented C++ Bindings for it

Balcer: one problem we have is that our allocator works well with LLVM libc++, not with GNU libstc++. The main issue is that we want to use standard containers, but allocator<T>::pointer is assumed to be T*

McFarlane (quoting Craig): Arthur O'Dwyer is working on something about Fancy Pointers. There seems to be a correlation with the problems you are facing

Pusz: we talked about it in ABQ

Balcer: interaction with transactional memory brings visibility atomic; we want failure atomic too

Roy: There will be a form of "remote_ptr" with executors at some point, I'm pretty sure, so there might be correlations there too

Keryell: It reminds me also https://github.com/keryell/ronan/raw/gh-pages/Talks/2016/2016-03-13-PPoPP-SYCL-triSYCL/2016-03-13-PPoPP-SYCL-triSYCL-expose.pdf plus the proposals on accessors and managed pointers, in a slightly different context

Boissonneault: wasn't there a proposal for C++20 to move from Fancy Pointers to raw pointers?

Balcer: not sure

Boissonneault: I'll find it

Balcer: send it to me, we will be grateful for any support

Balcer: there is a risk of Undefined Behavior with virtual functions stored in persistent memory and invoking functions on non-constructed objects as objects can exceed the lifetime of their process

Balcer: the fact that standard container layout can differ between implementations and versions implies we might need a versioning mechanism to avoid (silent, really not good) corruption

Boissonneault: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/p0653r2.html is what I had in mind

Craig: 32 bits vs 64 bits layout lead to similar issues

Craig: https://capnproto.org/ performs memcpy()-based serialization, and can inspire versioning solutions

Balcer: is it based on precompiling?

Craig: it uses an IDL of sorts. I think it can be alleviated with static reflection

Wilmans: I'm putting all the links on nullptr.nl/sg14

McFarlane: the heterogeneous computing people face similar issues

Balcer: instead of hardware transaction memory, I suggest we use software transactional memory which already does tracking

Balcer: our work is Open Source, and the presentation is public. Look for pmem.io

McFarlane: this is interesting stuff, thank you

Keryell: we have constexpr new coming, and now persistent RAM :)

Balcer: we can imagine games running without having to update databases

McFarlane: it's like heterogeneous but without the processing required to move things around

Keryell: in TOR, SG1, we discussed similar issues

Wilmans: is this Linux-specific or is this portable?

Balcer: it's not Linux-specific. Windows has similar support, and our library is multiplatform (Windows, Linux, FreeBSD)

Wilmans: does it require LLVM on Windows?

Balcer: we have Visual Studio projects, so it compiles on Windows

Balcer: for the slides, see https://docs.google.com/presentation/d/e/2PACX-1vRH35IHLm4rj9woMZj19c15RhW69-z-hQzdbufLItiVltRfUCDJHojqRFL1NeTkfxP1COZPh_RSXF1X/pub

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

Domain-Specific Discussions

-=-=-=-=-=-=-

Games:

Davidson: we had a BSI meeting yesterday, and we discussed over-aligned memory. This paper looks like what we need to standardize alignment requirements in that area

Davidson: I think it's only going to be a matter to getting the wording right

McFarlane: this will be useful for many applications

Davidson: we also discussed the identity metafunction

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

McFarlane: we've had some pretty good presentations today

McFarlane: anything else we need to talk about today? (nope)

McFarlane: future Face-to-Face meetings? One in Germany (McFarlane might go)...

Wilmans: emBO++ will take place from 9th to the 11th of March 2018 in Bochum, ruhr-valley, Germany

Bendixen: emBO++ will have Face-to-Face on Sunday from what I can tell

McFarlane: does anyone plan on going to JAX?

Roy: I will

Pusz: me too (a few others too, I missed them)

Fertig: I have a proposal idea but it send that my mic is not working :-(

-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

McFarlane: thanks for attending

(we close at 15:53 EST)

Minutes for 2018/01/10 SG14 Conference Call

Meeting minutes by Michael

1.1 Roll call of participants

Andreas Fertig Ronan Keryell, Brett, Michael Wong, Kevin Boissonneault, Billy Baker, Ben Craig, Allan Deutsch, Mateusz Pusz, Jeff Hammond, John McFarlane, Wouter Van ooijen, Ben Saks, Shay Morag, Arthur O'Dwyer, Charley Bay, Paul Bendixen, Nevin Liber
John McFarlane chairing as Michael had audio problems

1.2 Adopt agenda

Yes

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

1.4 Action items from previous meetings

2. Main issues (125 min)

2.1 General logistics

Review last call discussions

None

2.2 Paper reviews

2.2.1 Compile time only constexpr by Andreas Fertig

https://github.com/andreasfertig/isocpppapers/blob/master/compile-time-only-constexpr.pdf

aim for embedded systems.
constexpr functions are great for embedded systems, but some usecases require 100% sure that they are executed at compile time
use an attribute to mark constexpr fn that it should really be evaluated at compile time or fail

ensure functions will never be executed at on device or increase binary size
Peter Sommerlad suggested constexpr parameter functions
Also Daveed had constexpr parameter with type traits or compiler intrinsics
idea is important, not sure about use of constexpr
try overload on constexpr and delete the non-const expr
another Daveed paper allows vector at compile time, and use std::allocator to do that. love to use it at compile time but not runtime
P0595

attribute on function, what does it mean, want to produce a hard compiler erro that is static assert, overload resolution is sfinae

no way to detect currently o detect whether function is executed in a constexpr context, right compiler may not know until it looks at the functionbody
can't detect but can force it to happen at compile time with enumerant value , or template parameter, or initializer for constexpr variable
these are at point of use of function, not at the declaration
that is valuable to have that guarantee
look at 2nd para of Daveed's paper is very valuable and tackles the problem heads on, though I don't like the answer
has many complicated conditions and one bit to give you the answer, need a way to expose more of those bits
is there a way to separate the constexpr fn to compile and runtime

John M has a paper with Louis Dionne on constant presented in ABQ, taking integral constant making it general purpose, making a template parameter,
P0870

use cases need to be clearer, likes the proposal
variadic square root solver was not optimal compared to modern fpu

both Daveed and this paper need more examples

There is an interest to this paper, Daveed shows a way for constexpr operator to provide 2 different implementations within one function.
Still aim for the case where it should be at compile time.
Use case needs more work
If it was available now, I would use it immediately

no compiler reference implementation yet, tried to do it with clang but could not complete
may be internal EDG
Daveed seems to imply compiler after first pass fills in whether this is compile time, it tries to fill in body but if it throws, then it may abandon that path

alternative may be operator or type trait as you can statc_assert or sfinae on them, but still not right

Timur Doumler
Pointer to over-aligned memory

https://drive.google.com/open?id=107tefQn_qiymX4VaP89TXB8qW-U5GPZS

presented by Arthur O'dwyer

array of 4 floats will be aligned to some alignment, but cannot communicate alignment effectively in C++

could put it in a struct with data member with align_as, still need type punning to convert to that alignmnent

every compiler gives you a way explict build_in to control this, gcc, clang has this, and also offers attribute

this just standardize this to C++11 attribute

also a statement-like intrinsic with input, and compiler assumes that input is aligned

The attribute is the right

only allowed on fn declaratins and parameters, also can say return value is aligned

does not say if MSVC has the attribute

ARM compiler has a way of aligning with registers, 128 and 256 are the only 2 ways

if passing something is 128 bit aligned and compiler assume 32, its fine

the opposite will get undefined behaviour

if it ignores attribute, it will not be aligned at all

would liek a contract that says this is going to be aligned otherwise fail horribly, instead of running and get performance degredation

can already to that with a runtime assert;

can we use contract for this? but is contract ready as there was some reflector drama

how does it affect the ABI?

existing solutions require macros and compiler intrinsics

Done

Allen Deutsch mention he has the slot map paper revision 2 ready P0661

presented at CPPCON

only changes is the format, markdown now

ready for wording and name bikeshedding

involvement with colony from Matthew? Yes

similar to unoreded map which we hate with reserved slots

needed due to small reserve capacity with high churn, generation counter for those will grow a lot

wording does not mean design is over

any sample code that uses this?

this works like a map

store elements and contiguous elements at front, lookup is fixed cost, 2 ptr chases

LEWG looked at this in Toronto in small groups

slot_map gives you the key, instead of you providing it with the value

I can give some sample use cases, when crossing user kernel boundary or process boundary, we have to use unique id and use std map, which is not performant

worsk if key is only used on one side,

the other side just hold on to the number, then serialzie across the boundary

yes slot map is perfect for this

aiming for JAX

no other comments

2.2.2 any other proposal for reviews?

Timur Doumler
Pointer to over-aligned memory

https://drive.google.com/open?id=107tefQn_qiymX4VaP89TXB8qW-U5GPZS

tower120

High performance thread-safe container unordered iteration + access technique.

Experimental implementation:
https://github.com/tower120/SyncedChunkedArray
Technically, container is linked list of fixed size Chunks.
Do compact/merge chunks whenever possible, hence elements stored continuously, and container does not degrade in speed like plf::colony after erase. Have trackable_iterator for element access. O(1) erase/emplace.

2.3 Domain-specific discussions

2.3.1 Embedded domain discussions

Embo++ in early March

2.3.3 Games Domain

Tim Sweeney CTO of Unreal engine, actively looking at Garbage collection in C++
GDC in SF
will we have one at GDC?
maybe

2.3.4 Finance Domain

nothing to add

2.4 Other Papers and proposals

Will present numerics next time
reminder to learn how to get on slack channel

2.5 Future F2F meetings:

Embo++ conference hosting SG14 meeting on Sunday, can get free pass if SG14 member

2.6 future C++ Standard meetings:

Mailing deadline is Feb12

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4677.pdf

2018-03 JAX WG21 meeting information

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)

Please make recording available.

Minutes are in ISO mailings

5. Closing process

5.1 Establish next agenda
Feb 14

5.2 Future meeting

Jan 10 this call
Feb 14 Note Feb 12 10 am ET is mailing deadline for JAX
Mar 14 (JAX meeting)