Document Number: P1909R0
Date: 2019-10-07
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 2019/08/14-2019/09/11

Contents

Minutes for 2019/08/14 SG14 Conference Call

1.1 Roll call of participants
>
Ben Saks, Ben Craig, Hubert Tong, Javier Cabezas, John McFarlane, Marco Foco, Matt Harrington, Matthew BUtler, Ronen Friedman, Staffan Tj. Michael Wong

> 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
>
> 2. Main issues (125 min)
>
> 2.1 General logistics
>
> Review Cologne results.
>
Flat map : P1727 in LEWG, deferred to LWG after C++20, some concerns about unordered maps, Zach Laine driving, may need range component maturaity

Deterministic exception: EWG presented by Herb:
1. continuing on exception violation
2. memory exhaustion

EWG polls had some split results, Bjarne, Daveed, Chandler were strongly against the out of memory aspects, and possibly implementation concerns

Affinity: high level affinity looks likely to move ahead
low level proposal topology discovery that is more lists like, instead tree like

Niall papers on elsewhere/unreacheable memory:
P1026 next step go to SG1 for more confirmation before a SG is started

Linear Algebra Sg14 F2F;
both proposals were passed to LEWG, they try for integration

Plan CPPCON SG14 meeting
>
Sept 18 Wednesday
Please get in touch about reserving a seat
Linear Algebra at CPPCON SG14
Please propose papers for next call Sept 11.as deadline.
Please submit draft to the reflector asap even before the Sept 11 meeting.
Patrice will be there.
SG14 members can enter for free but still need to register. others buy a ticket, but the cost can be commuted
34 paid and 9 free

Belfast C++ meetng:
2 events: committee starts on Nov 4, Nov 11 is ACCU Ireland follow on
can hold an SG14 meeting for games developers and low latency in the UK

2.2 Paper reviews
>
> 2.2.1 Improving Debug Builds Inline With User Expectation: John MacFarlane
>
> p1832.html <https://lists.isocpp.org/sg14/att-0190/p1832.html>
>
debug builds based on GCC -O0 -Og implementation
selecting which function to be inlined
generally disables inlining to enable stepping through
-Og enables the following optimization flags for function defined in system headers -isystem
affects numeric work
need something intermediate between debug and release build

needed as a queue to debug for embedded systems
other impacting effects as well, when debugging interactively, real pain to step over std:: begin and end for algorithm
tried to narrow the scope

yes have an issue: direction currently is compielr options and tweak compiler behaviour
much more suited as an attribute
like explicit inlining
eigen library has this problem, undebuggable
if you want to step into your own code, but not some dependent library code, so need fine grain control
good candidate for inlining not based on location of source code, but whether it a dependency
bad thing is that attribute need to be used in conjunction with macros, explicit inlining is a terible semantics

really need a hint for optimization, debuggability or some slider
macro to be used for my code that I want to debug

there is precedence for -isystem vs I, should be alternate interface that
is not based on location

on Msvc side there is no external I, they chose not to go that way due to
warnings, want to be able to give type conversion warnings in a template,
hard to get MSVC to take /Iexternal, but/Ob1 fo give good results
so I like the direction of turning more inlining
will find the blogpost

suspect msvc agree they have an issue. But in 2019 or 2017 one of the
change they introduced as a default is not step into, so they change the
step through experience to elide anything that was in angle bracket

modules dust is still setting, but they could help solve the problem, with
headers from one body of code is finding its place into a TU from another,
but still have to explictky say so that someone elses code is not to be
inlined

looks like there is some support external:I in Visual studio

like the pitch of the problem, the solution is general direction: to take
this beyond QOI
committee and if they want a more generic solution, then it will be more
load on compiler implementations
This problem is more chronic for numerics, where inlining and expression
templates with auto

No objections to moving onto SG15 tooling

Arthur also sent feedback.

> 2.2.2 Error Size Benchmarking by Ben Craig
>
> P1640R0: Error size benchmarking
>
>
> https://raw.githack.com/ben-craig/error_bench/master/error_size_benchmarking.html
> <
> https://urldefense.proofpoint.com/v2/url?u=https-3A__raw.githack.com_ben-2Dcraig_error-5Fbench_master_error-5Fsize-5Fbenchmarking.html&d=DwMGaQ&c=r2dcLCtU9q6n0vrtnDw9vg&r=bHyceIQQHQvbfTSwvF3b5ym3XCQIh0_iFRNJbNk-FCc&m=_OFSroXnnYHKfBQqw8TVSac0et4fEQ80IMeaj-lWcD4&s=LGjT-TVB94ptHzUmdPNh4LJr1eMpKuAcmL7pQSWzxxA&e=>

>
>
Didn't quite understand what the various scenarios mean.
Didn't manage to try the EDG portable exception mechanism.
Not sure only EH overhead is measured.
noexcept is grossly underused
- Because of Lakos rule
- Because library shies away from conditional noexcept
-> Maybe this is more of a library issue
It was a mistake to require termination from noexcept functions
that attempt to throw. Forces generation of EH tables.
Also, because something like vector::operator[] isn't noexcept
(due to Lakos rule), it's easy to get a situation where the
compiler cannot prove that no exception is thrown.
Analogy: Comparing the size of "int main() {}" to
"#include<iostream>
int main() { std::cout << "Hello\n"; }"
will see a huge growth in size.
We should compare the cost of the hot paths also.
Ben Craig concludes that C++ EH doesn't serve all domains well.

Ask for Patrice to restart his work
may be work with EDG, XLC, Sun compilers as well

2.2.3 PTF/Colony?
>
No one to present.
had a mini discussion on the reflector, to support SIMD.
Matt Bently working on implementation to make use the simd scatter gather instructions which only work on ARM for the core pieces on Colony. He wishes to know if this was worthwhile. Niall was mildly positive, staffan was mildly against. Matt will go off and experiement.

Is that to get better performance? Probably but it is not one of the detail that WG21 cares about. There was concern about intrusive vs nonintrusive interface.

>
> 2.2.4 Linear Algebra update from Aug 7
>

>
> Next call: Sept 4: 3-5 PM ET
>
> 2.2.5: Any serious study on cost of Exception vs cost of Error Codes

>
Done by Ben Craig's paper.

> 2.2.6 any other proposal for reviews?
>
> 2.3 Domain-specific discussions
>
> 2.3.1 Embedded domain discussions: Ben Craig, Wooter and Odin Holmes
> 2.3.3 Games Domain: John McFarlane, Guy Davidson and Paul Hampson
> 2.3.4 Finance Domain: Carl Cooke, Neal Horlock, Mateusz Pusz and Clay
> Trychta
>
> 2.3.5 Lnear Algenra: Bob Steagall, Mark Hoemman
>
> 2.4 Other Papers and proposals
>
> 2.5 Future F2F meetings:
>
> 2.6 future C++ Standard meetings:
> https://isocpp.org/std/meetings-and-participation/upcoming-meetings
>
>
> - *2019-11-04 to 09: Belfast, Northern Ireland;* Archer Yates
> -
> - 2020-02-10 to 15: Prague, Czech Republic
>
> - 2020-06-01 to 06: Bulgaria
> - 2020-11: (New York, tentative)
> - 2021-02-22 to 27: Kona, HI, USA
>
> 3. Any other business
> Reflector
> https://lists.isocpp.org/mailman/listinfo.cgi/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
>
> Sept 11
>
> 5.2 Future meeting
>
> Sept 11
>
> Oct 9: mailing deadline Monday Oct 7
>
> Nov 13: cancelled due to C++ Std meeting

Minutes for 2019/09/11 SG14 Conference Call

1.1 Roll call of participants
>
Andreas Fertig, Ben Craig, Billy Baker, Grafik Robot/rene Riviera, Matthew Butler, Hubert Tong, John McFarlane, Marco Foco, Ronen Friedman, Staffan Tj, Michael Wong, Neil Horlock, Mark Hoemmen

> 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
>
None.

> 2. Main issues (125 min)
>
> 2.1 General logistics
>
> Plan CPPCON SG14 meeting
>
Room booking is in Gaylord Hotel, about 30-40, regular seating
Rene Riviera.Steffan TJ, Matthew Butler. Michael Wong

2.2 Paper reviews
>
> 2.2.1 Discuss Possible paper agenda for CPPCON.
>
> 1. "Member Layout Control"
> <
>
>
https://raw.githack.com/grafikrobot/papers/master/wg21/member_layout/member_layout_D1605R0.html
> >
>
> C++ alliance Marsahl Clow, Vinnie

>
Allow optimizations that are difficult for embdded systems, rearranging
member of class for packing
consider interaction with alignas
should support alignas within this feature
talk about cache line and false sharing
talk about context sensitive keyword
solves a very specific issue, this could be not broad enogh for C++
would prefer all the controls in one place, unifying all the other
alignment controls padding ,layout, order, alignment
propose some form of unification, like pragma pack,
this depends on committee
suggest syntax be extendable, to alignment

2. Linear Algebra by Bob Steagall/Guy Davidson
>
> 3. BLAS by Mark Hoemmen
>
Working on implementation of the proposal with stub filled in
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1674r0.md
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1673r0.md

wrapping one of the BLAS function, takes mdspan as input and output, has
layout and accessors, not all are friendly to the BLAS
implementation has a selection process that picks out what is really needed
previous review has concerns about small matrixes which needs to pass by
value
mdarray proposal is a container version of mdspan may be useful for pass by
value
small linear algebra, but batches of them at a time

Games might want to target this as well
design is based on BLAS but that is based on larger matrixes
one optimization that is missing is when value is so small it could pass by
reference

need feedback on vector-vector, matrix-matrix operations, and these
vector-vector overlap standard algorithms,
some of them makes sense to keep, but others may not, e.g. dot-products,
norm,
copy-swap is one that overlaps standard algorithm
possible integration means implementing 1385 on top of this BLAS wrapper

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1385r2.html

Intention is not to compete with each other

Feedback request:
What about very small matrices and vectors?
Can take those as md-span or md-array by reference, so should we try to pass by value, if you have experience with very small matrices to weigh in.
Should we drop BLAS-1 and replace with standard algorithms? This is thin-BLAS

Can we have an engine in 1385 that would call the function in this proposal? Yes Bob and I have thought of that, then BLAS api would be there.
Though some people prefer an explicit interface, and additional capabilities like different data layouts for increased performance.

Reference vs value case, from games industry needs to understand needs
comparable performance between debug to release builds
debug builds have to inline more

one of the key requirement: debug builds are rare and slow, and no inlining in most libraries we use,
so we force inlining in both debug and release, need to reduce the number of calls if you can get the code to inline for small things
dereference and indirections kills debuggability

if matrices have compile dimensions, then we may be able to plug in our own vector code, with built-in intrinsics, pluggable only needed for one implementation, usually custom written code

without engine to plug in different backends, then you get what std lib provides, this will kill it for game developers
like batch interface? yes
how about as a library? probably
single data arrays that we transform is ok
this gives value to change BE and customize

Please consider Paper 1832 shows game industry expectation:
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1832r0.html

SG15 is meeting after CPPCON, Rene will be there and may proxy P1832

Any other papers?
>
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/p1832r0.html
Improving Debug Builds Inline With User Expectation before LA proxy by Rene

> 2.2.2 Error Size Benchmarking by Ben Craig Update.
>
> P1640R0: Error size benchmarking

>
>
> https://raw.githack.com/ben-craig/error_bench/master/error_size_benchmarking.html
> <
>
> https://urldefense.proofpoint.com/v2/url?u=https-3A__raw.githack.com_ben-2Dcraig_error-5Fbench_master_error-5Fsize-5Fbenchmarking.html&d=DwMGaQ&c=r2dcLCtU9q6n0vrtnDw9vg&r=bHyceIQQHQvbfTSwvF3b5ym3XCQIh0_iFRNJbNk-FCc&m=_OFSroXnnYHKfBQqw8TVSac0et4fEQ80IMeaj-lWcD4&s=LGjT-TVB94ptHzUmdPNh4LJr1eMpKuAcmL7pQSWzxxA&e=>
>
>
> 2 papers for Belfast: stripped down version of this same paper more focused
working on timing/speed paper, got data, will share it in the chat, need to
add words and graph
https://raw.githack.com/ben-craig/error_bench/err_gauss/results/ret_nd/happy_8.html
early data
showing exception/terminate to be less cost then other error returns
checking with implementers on the numbers
nothign surprising so far at macro level
micro level yes, returnign a struct that is trivial, is more expensive then
a non-trivial with a defined destructor, due to register filling
are you looking value in the struct, struct is a pointer to error domain
and a value, if domain is null then there is no error and i dont look
in some cases ,termnate is a recovery mechanism, when compiler running and
runs out of diskspace, then ok to terminate, could also be when u run out
of heap

2.2.3 PTF/Colony?
>
> 2.2.4 Linear Algebra update from Sept 4
>
> Next call: Oct 2: 3 PM ET
>
> 2.2.5: Any serious study on cost of Exception vs cost of Error Codes
>
Ben Craig is working on this.

> 2.2.6 any other proposal for reviews?
>
> 2.3 Domain-specific discussions
>
> 2.3.1 Embedded domain discussions: Ben Craig, Wooter and Odin Holmes ,
> John MacFarlane
> 2.3.3 Games Domain: Rene Riviera, Guy Davidson and Paul Hampson
> 2.3.4 Finance Domain: Carl Cooke, Neal Horlock, Mateusz Pusz and Clay

> Trychta
>

Peter Langford of STAC to think about the next London November event after Belfast meeting. Discussion of what C++20 may be offering for finance commnuiy.

SG1 concept of affinity may be different then affinity in low latency which requires placement and explciit control
HPC rely on more the way you launch a batch job on placement on MPI and OpenMP threads, and OpenMP deals wth first touch and thread teams
one virtual allocation that crosses numa domains
we dont use explicit control as much unless it is on GPUs

P484 thread constructor attribute
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/p0484r1.pdf

Should we create a scorecard to WG21 on each of our domains.

2.3.5 Linear Algenra: Bob Steagall, Mark Hoemman
>
> 2.4 Other Papers and proposals
>
> 2.5 Future F2F meetings:
>
Sept 18 CPPCON

> 2.6 future C++ Standard meetings:
> https://isocpp.org/std/meetings-and-participation/upcoming-meetings
>
> - *2019-11-04 to 09: Belfast, Northern Ireland;* Archer Yates
> -
> - 2020-02-10 to 15: Prague, Czech Republic
>
> - 2020-06-01 to 06: Bulgaria
> - 2020-11: (New York, tentative)
> - 2021-02-22 to 27: Kona, HI, USA
>
> 3. Any other business
> Reflector
> https://lists.isocpp.org/mailman/listinfo.cgi/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
>
>
> 5.2 Future meeting
>
> Oct 9
>
> Oct 9: mailing deadline Monday Oct 7
>
> Nov 13: cancelled due to C++ Std meeting