

Document Number: P0230R0  
Date: 2016-02-12  
Authors: Michael Wong  
Project: Programming Language C++, SG14 Games Dev/Low Latency/Financial  
Trading/Banking  
Reply to: Michael Wong <fraggamuffin@gmail.com>

## **SG14: Games Dev/Low Latency/Financial Trading/Banking Meeting Minutes 2015/10/14- 2015/02/10**

### **Contents**

Minutes for 2015/10/14 SG14 Conference Call .....	2
Minutes for 2015/11/11 SG14 Conference Call .....	7
Minutes for 2015/12/09 SG14 Conference Call .....	10
Minutes for 2016/01/13 SG14 Conference Call .....	15
Minutes for 2016/01/20 SG14 Conference Call .....	24
Minutes for 2016/02/03 SG14 Conference Call .....	26
Minutes for 2016/02/10 SG14 Conference Call .....	35

## Minutes for 2015/10/14 SG14 Conference Call

Meeting minutes by Michael

### 1.1 Roll call of participants

Guy Davidson, John McFarlane, Brett Serles, vishal oza, sunil sricastava, Brent Friendman, Michael B. McLaughlin, Niolas G, Sean M, Billy Baker, Michael Wong, Ville,

### 1.2 Adopt agenda

Approve.

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

Approve.

### 1.4 Review action items from previous meeting (5 min)

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

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

1.4.2. 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 report, C++ Standard meetings , C++ Kona mailing deadline (Sept 25)

Please see the slides for the report here which was presented in the Friday session by Sean and Nicolas which also outlines the problem domain, our motivation as well as the status of the discussion.

The Birth of SG14

[https://docs.google.com/presentation/d/19iM3VIa24bStFHQwgniSO1J4vQjHR8curjOt6j3Liz8/e/dit#slide=id.g66ea100d2\\_0\\_58](https://docs.google.com/presentation/d/19iM3VIa24bStFHQwgniSO1J4vQjHR8curjOt6j3Liz8/e/dit#slide=id.g66ea100d2_0_58)

Please ask John Spicer for document number.

And send John Spicer the actual document.

[jhs@edg.com](mailto:jhs@edg.com)

2.2 Kona Papers and their presenters. 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.

We have 8 papers upstreamed from the meeting for Kona

[P0037R0](#) Fixed point real numbers John McFarlane LEWG SG14/SG6: Lawrence Crowl

[P0038R0](#) Flat Containers Sean Middleditch LEWG SG14: Patrice Roy

[P0039R0](#) Extending raw\_storage\_iterator Brent Friedman LEWG SG14: Billy Baker

[P0040R0](#) Extending memory management tools Brent Friedman LEWG SG14: Billy Baker

[P0041R0](#) Unstable remove algorithms Brent Friedman LEWG SG14: Billy Baker

[P0048R0](#) Games Dev/Low Latency/Financial Trading/Banking Meeting Minutes 2015/08/12-2015/09/23 Michael Wong SG14: none

[P0059R0](#) Add rings to the Standard Library Guy Davidson LEWG SG14: Michael

has an update D0059R1

[P0130R0](#) Comparing virtual functions Scott Wardle, Roberto Parolin EWG SG14: Michael

motivation? goal to instruct instruction cache usage, vector of polymorphic objects such that you can call the same vfuncs over and over again  
branch predictor cannot help.

[P0125R0](#) std::bitset inclusion test methods: Michael/Walter

2.3 Exception handling brainstorm

We had a major brain storm on why games programmers tends to turn off exception handling, and a brain storm session to see what kind of options and ideas to address their concerns.

We intend to bring to discussion in addition to Exception/RTTI costs, topics regarding the following, some of which crosses into other groups such as SG1 Concurrency, SG6 Numerics and SG7 Reflections.

Wednesday evening session on SG14

Sean:

1. make noexcept the default but still allow exceptions to be enabled  
allows games an easy way to not mark up every single function every class  
its just a compiler optimization  
hoops to jump through

2. function decoration to assert non-throwing-ness  
unlike noexcept, but still allows exception to be passed but have no unwinding information  
exceptions are rare and will crash anyway

3. more in language alternative, mark a namespace or modules as noexcept, to get developers to keep exception off, and still allow to use STL, bringing more developers to Standard C++

Vishal: allow an opt in way of generating exception info only when needed

Sean: still need compat with C and older C++

Vishal: do it on the linker stage or LTO

, undefined behaviour

delayed to link time compiler would have to generate multiple versions, longer compile time, and larger space

People are sending me snippets of their brainstorm. Please continue to post to this list. Thanks.

## 2.4 Other papers from the mailing of interest to SG14

There are a few papers in the current mailing that is of interest to SG14.

GPU Accelerator support

[P0069R0](#) A C++ Compiler for Heterogeneous Computing

We actually intend to study the design of C++AMP, OpenMP Accelerator/OpenACC, OpenCL, OpenGL and Vulkan, Diectx12 for a GPU accelerator design to support gamers. We plan to review this paper under a co-located SG14 session which can run with SG1, but we are interested in taking this work further.

Michael Wong, rasha

Allocators

[P0089R0](#) Quantifying Memory-Allocation Strategies

Guy interested

Get Andrey

Coroutines and games

a lot of compiler magic, can't predict very reasonably how it behaves due to multiple platforms support

[P0054R0](#) Coroutines: reports from the fields

[P0055R0](#) On Interactions Between Coroutines and Networking Library

[P0070R0](#) Coroutines: Return Before Await

[P0071R0](#) Coroutines: Keyword alternatives

[P0073R0](#) On unifying the coroutines and resumable functions proposals

[P0099R0](#) A low-level API for stackful context switching

Intrusive containers

likely/unlikely

function multiversioning

low latency for financial/trading

matrix operations

SIMD vectors

[P0076R0](#) Vector and Wavefront Policies

2 competing problem space:

1. getting high performance SIMD: ISPC
2. standard communication between library with simple geometric types:

Reflection

Numerics:

[P0106R0](#) C++ Binary Fixed-Point Arithmetic

String:

<http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/p0067r0.html>

Sean

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

Next meeting: Kona Std meeting (no telecon)

Next call : Nov 11 coroutines

Nov 25: SIMD

- *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/Gor on coroutines*

Nov 25: SG14 special telecon/Pablo and Xinmin on SIMD

- *Dec 9: SG14 monthly telecon/ Andrey on Allocators*

## Minutes for 2015/11/11 SG14 Conference Call

Meeting minutes by Michael

### 1.1 Roll call of participants

Please add your name here  
Michael Wong

### 1.2 Adopt agenda

Yes

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

Yes

### 1.4 Review action items from previous meeting (5 min)

1.4.1. All: Consider attending SG14 F2F meeting hosted by SONY at GDC 2016

Tentative: Tuesday March 15 and/or Wednesday March 16

<https://groups.google.com/a/isocpp.org/forum/?fromgroups#!topic/sg14/Euo2So0kYew>

1.4.2. All: Consider attending Jacksonville Fl, C++ Std meeting Feb 29-Mar 5.

## 2. Main issues (50 min)

2.1 Review SG14 logistics, for upcoming telecons and meetings: 5 min

New papers for the next C++ Standard meeting in Jacksonville should be named PxxxxRyy advancing the revision to the next number.

We use DxxxxRyy only for papers that we plan to discuss in a special SG14 meeting, which will be changed to a P paper when published officially for WG21 Meeting mailings.

WG21 Meeting Mailings deadlines are:

Feb 12 for the pre-meeting mailing

Current logistics is to focus on revised papers of the papers we presented in Kona for the Jacksonville meeting, reserving any new papers for the SG14 GDC meeting. Of course, new papers can still be accepted for Jacksonville if someone is heavily motivated to work on it in the short time frame we have (which is only about 3-4 telecon calls).

the papers presented at GDC will be presented in June at the Oulu Finland meeting in June.

We are still waiting on SONY to confirm details of the SG14 GDC meeting. No update yet.

## 2.2. Coroutine and Games (Gor): 30 min

Gor presented with notes he emailed just before the call. Thank you Gor.

[https://11950069482448417429.googlegroups.com/attach/44f49f1fe5dde/D0XXX\\_CoroutinesAndGames.html?part=0.1&view=1&vt=ANaJVrH5dxD1W4lGefIqQEFOY6\\_Afct\\_SdXqXw9ZsrMaBWCOTNOpBTUWTh5Ir1vWfGnqKdiQdpKpg\\_IGjWmP2T4Yxq\\_fQv44QVc2K3QfdxTuA-DDpbbRvFM](https://11950069482448417429.googlegroups.com/attach/44f49f1fe5dde/D0XXX_CoroutinesAndGames.html?part=0.1&view=1&vt=ANaJVrH5dxD1W4lGefIqQEFOY6_Afct_SdXqXw9ZsrMaBWCOTNOpBTUWTh5Ir1vWfGnqKdiQdpKpg_IGjWmP2T4Yxq_fQv44QVc2K3QfdxTuA-DDpbbRvFM)

Questions and issues?

## 2.2 Review proposals results from Kona: 15 min

<https://groups.google.com/a/isocpp.org/forum/?fromgroups#!topic/sg14/-Dw-jYbhlAc>

We won't have time to discuss in detail, but we can go over major questions and seek opinions and feedbacks

[P0037R0](#) Fixed point real numbers John McFarlane LEWG SG14/SG6: Lawrence Crowl  
John has an implementation and this helps. Lawrence will try to work on one for Jacksonville so that a meaningful comparison can be made, otherwise John's approach would be favored.

[P0038R0](#) Flat Containers Sean Middleditch LEWG SG14: Patrice Roy  
Proceeding well with some edits. Please prepare an updated paper.

[P0039R0](#) Extending raw\_storage\_iterator Brent Friedman LEWG SG14: Billy Baker

[P0040R0](#) Extending memory management tools Brent Friedman LEWG SG14: Billy Baker

[P0041R0](#) Unstable remove algorithms Brent Friedman LEWG SG14: Billy Baker

[P0059R0](#) Add rings to the Standard Library Guy Davidson LEWG SG14: Michael  
Arthur O'Dwyer is interested in a merging of his proposal and this one. Please work with him.

[P0130R0](#) Comparing virtual functions Scott Wardle, Roberto Parolin EWG SG14: Michael

[P0125R0](#) std::bitset inclusion test methods: Michael/Walter

## 2.3 Ongoing topics placeholder (which we won't likely get to but will schedule)

Exceptions/RTTI

SIMD vector and Matrixes

Allocators

GPU/Accelerator design

Given the GPU presentation in Kona from AMD's HCC, I would like to add this now to the schedule for discussion.

Array View and Bounds checking

## 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

Next call : Dec 9

(I decided to not hold extra meetings in Nov as many people are busy with delivery)

Nov 11: coroutines and games

Dec 9: review committee decisions

Jan 13: more paper discussions, review SIMD?

Feb 10: prep for Jacksonville WG21 meeting

Feb 12: pre-meeting mailing deadline

Feb 29-March 5: Jacksonville, Fl; WG21 meeting

March 4: Pre-GDC SG14 mailing deadline

Mar 15/16: GDC SG14 meeting

## Minutes for 2015/12/09 SG14 Conference Call

Minutes by Michael Wong

Michael Wong, Guy Davidson, Sean Middleditch, Hartmut Kaiser, Brent Styles, Scott Wardle, Jared Hoberrock, Mike Garland, Al Grant, Billy Baker, Rosha O, Nicolas G, Matt, John mcgluaghin,

1.2 Adopt agenda

Yes

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

yes

1.4 Review action items from previous meeting (5 min)

1.4.1. All: Consider attending SG14 F2F meeting hosted by SONY at GDC 2016

Tentative: Tuesday March 15 and/or Wednesday March 16

<https://groups.google.com/a/isocpp.org/forum/?fromgroups#!topic/sg14/Euo2So0kYew>

1.4.2. All: Consider attending Jacksonville FL, C++ Std meeting Feb 29-Mar 5.

2. Main issues (50 min)

2.1 Review SG14 logistics, for upcoming telecons and meetings: 5 min

To make C++17, proposals must be approved in the next 2 meetings.

2.2. Accelerator design using Nvidia's Agency (Jared Hoberrock) : 40 min

<https://github.com/jaredhoberrock/agency/wiki/Quick-Start-Guide>

Jared and Mike G presenting slides

Slide 2: bulk\_invoke

Bulk synchronous parallelism: just fork join, valient 1990 paper

[https://en.wikipedia.org/wiki/Bulk\\_synchronous\\_parallel](https://en.wikipedia.org/wiki/Bulk_synchronous_parallel)

std::invoke like

par

invocations are not in order with each other, in parallel,

2nd arg is just parallel agent a handle to an execution agent

take self parameter and ask it for its index, use to identify and do arithmetic  
after we call bulk invoke it is a synchronous call,  
n is the number of agent inside par

slide 3:

still experimental at this stage

manipulating execution agents

abstract mapping to underlying platforms, whether threads, GPUs, simd units,

no guar of performance; agency does not prevent programmer write low level  
architecture specific code

I usually write to prevent pathological cases

slide 6: structured execution

matches C++ Std

work placement: where execution happens

slide 7

Vision for parallelism

borrowed executor idea from standardization

ex agents are basic units of work

futures for dependencies

slide 8: .then, when any, when\_all, tools concurrency

executors: tools for low level work creation

agency is a bridge high level algorithms with low level work creation of executors

slide 9:

bulk\_invoke sync with caller (caller must wait until all ex agents finish)

sometimes want async

bulk\_async: a massive form of std::async

today, you need to nest a for loop inside a call to std::async

or call std::async inside each array element

inefficient to create a new thread for each array element, this is better

this returns a future, so dont have to wait for the saxpy to finish

slide 11:

further generalize bulk\_async to bulk\_then

can line up a continuation and allow it to execute in bulk

this saxpy is not allowed to exec until the future is satisfied

Hartmut Q? just a guar that the code in the lambda can run concurrently; currently only  
used for fork-join parallelism only

allowed to run concurrently with itself

now it means the lambda can run concurrently with any other code in addition to itself

A: par is saying in slide 9/10/11 is that when u create this group of agents, they run with  
no order with any agent in the group

arbitrary exec in any order in any thread, but parallelism TS is only for fork-join

Hartmut: now bulk\_async returns a future, it implicitly allows the lambda execution with  
any other code

HPX has solved issue:

Mike G: this entails a semantic change from parallelism TS: around any call in the TS,

you can inject other calls, this world has agents in the loop to run along with other things;  
this relaxes it but does not change semantics of par

slide 12:

Jared: why bother with exec agents?

now allow us to encode ordering semantics of lambda and get niceties like overloading

can compose agent local storage; stash states instead of TLS

type system allows us to specialize

Q ?: pipeline parallelism allowed? any interagent communication?

Yes in a few slides

slide 13: define sequential agent

category: seq, par?

index: multi dimension

group\_size: shape

slide 14: parallel agent

slide 15: clusters, nodes, sockets, cores, lanes hierarchy emerge naturally

so our code is structured to match that hierarchy

expose that hierarchy to match mapping to hardware

slide 16: so this by nesting par(2, seq(2))

results in the indices which become tuples from 0.,0 to 1,1

slide 17:

nested policy to create a nested hierarchy for matrix transpose

slide 18:

how agency exposes concurrency

concurrent\_agent adds wait function (barrier)

slide 19: now can optimize matrix transpose

common to use onchip memory to accelerate program

added new argument share(tile)

instructs bulk\_invoke to create a copy of arg per group of agent

take the input and copy into this shared tile buffer, wait for the copy to complete with the barrier fn,

then do the transpose, and then copy it back out to the scratch pad

.wait synchronizes

slide 20: reduction

reduce done in a recursive way

traverse hierarchy of machine with nested execution

slide 21:

when writing this reduction, can ignore the number of core, simd width, only think about hierarchical information

slide 22:

executors is lowest level api for agency, like allocators for exec agent

agency model for locality: when u tell it to create some work, know how to allocate memory, where to exec

a customization point, bulk\_invoke manipulates them for users

slide 23: first exposes exec category; advertize its semantics

just exec a fn in a for loop

slide 24: place work on an executor: cpu\_executor

bulk\_invoke with .on will send it to cpu executor  
compose work placement with algorithm  
Slide 25: for gpus  
has a CUDA lambda  
now we can send it to gpu\_executor or multigpu executor  
slide 26: can get fancy  
nest exec in one in the other and create a hierarchy on the fly  
flattened takes a nested executor and reindex to be flat  
exec\_array: multiple GPU and make an array of them; without making individual calls to  
each gpu  
slide 27:  
take all this compose to make a parallel executor  
nesting can cause the ... to insert a for loop  
a simple flat executor whose agents exec in parallel  
slide 28:  
more ideas  
vectorize  
customization point to introduce new platforms  
slide 29: summary  
control structures create lots of parallelism on mass  
exec policies parameterize the control structures  
exe agents: param lambdas  
exe: doles out work; interact underlying platform

Q: to use agency to parallelize work, pop command from the queue to go towards gpu or  
cpu

A: dont think so; that queue of work can be stashed inside an executor, gpu executor that  
submits item to the queue, a user defined queue can compose with agency by writing a  
special kind of executor and mix it in

Q: I am thinking a task queue to a cloud based system or gpu and cpu  
abstracting the actual work item so it can be put in the actual device

Q: memory model is a major issue for sharedness or discrete

A: agency can have a preferred allocator

Q: how to split code between gpu and cpu

A: a compiler that takes a function that can generate code that can go to cpu or gpu thread  
decision can be made at compiler time or link time discussion

Q: allocator with interkernel communication like nvidia kepler, also AMD GPUs FIFO  
pipes in opencl/fpga

A: 2 ways that 2 kernels can communicate: 1. one consume the result of another using  
continuation,

2. if 2 kernels exec concurrently, then they communicate memory

Thank you to Jared and Michael !

- show quoted text -

Dec 9: review Nvidia's Agency with Jared Hoberrock

Jan 20: more paper discussions, review HCC  
Feb 10: prep for Jacksonville WG21 meeting  
Feb 12: pre-meeting mailing deadline  
Feb 29-March 5: Jacksonville, FL; WG21 meeting  
March 4: Pre-GDC SG14 mailing deadline  
Mar 15/16: GDC SG14 meeting

## Minutes for 2016/01/13 SG14 Conference Call

Minutes by Michael Wong

### 1.1 Roll call of participants

about 21-22 people; some of them are:

Michael Wong, Pablo Halpern, Jen Maurer, Brett Searle, Sean Middleditch, Jared Hoberock, Al Grant, Robert Geva, Harmut Kaiser, Ben Sander, JF Bastien and Hans, Michael Garland, Brett feidman, Sunil from Sony, Scott Wardle, Paul Mckenny, Olivier Giroux

### 1.2 Adopt agenda

Add item for JF's overview

### 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 F2F meeting hosted by SONY at GDC 2016  
Tentative: Tuesday March 15 and/or Wednesday March 16

<https://groups.google.com/a/isocpp.org/forum/?fromgroups#!topic/sg14/Euo2So0kYeW>

1.4.2. All: Consider attending Jacksonville Fl, C++ Std meeting Feb 29-Mar 5.

<http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4555.pdf>

## 2. Main issues (125 min)

### 2.1 Review SG14/SG1 logistics, for upcoming telecons and meetings: 5 min

JF. Bastien gave an overview on current status of user-aided vector programming:

<https://docs.google.com/document/d/1pzp-vEjDgI7UoHfhy10MU2tcIX3Omz8oVRYLg9LIKEY/>

This is a snapshot for today

for compilers, internal of compilers are quite ugly

want implementation and standardize practiced

Will send out paper in the next few days.

2.2. Intel Vector SIMD proposal : Pablo Halpern 90 min.

<http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/p0076r0.pdf>

Suggest we also include discussion on Implicit vs Explicit Wavefront issues  
Slides sent before meeting. Thank you Pablo.

Pablo Presenting

This is the high level proposal, a separate proposal on type  
acquire mutex in multiple lanes, can get race condition  
so synchronization is limited

Slide 4: current TS missing VEC

5: missing indexed based loop, can generate it but that obfuscates the code, and opaque to the compiler as well

as close as possible to something that looks like a control construct

6. in the if statement, it prematurely returns from lambda, this is like a continue, skips the rest of this iteration, and go on to the next iteration; body of the loop is the continuation

7. now add induction vars; vars that move in sync with the main vars

jv (arg to lambda starts out with 0 or j) advanced by 2, iterations are happening in parallel, so could have reused jv again, this is the linear induction

at the end, the for loop writes back to the induction

Robert G: can also have geometric induction using  $x*=4$ ; proposal only have linear ones for built in ones

8: reductions are critical too, want the sum is closely related and same as sum outside the loop but inside the loop is a partial sum, some value which is not useful yet, but you keep adding to it with local partial sums; min and max possible

9. can have user defined reduction types, using classes/types and starting value for the initializer; the identity

; requires associativity, but not commutativity

not commutative will maintain order and can do string and list

must have associativity, but may or many not need commutative

10: moving on to the bigger picture on the execution policies

unseq is a single threaded version of par vec

its mathematically parallel; no interdependency between the iterations at all

it differs from parallel in that it has restrictions from different iterations of the loops

as long as there is no mutex or atomics, then unseq can be more efficient than par, especially in small loops

can avoid some kinds of races,

the most interesting is VEC, vector execution policy

gets you classic vector loop execution

applies only to for loop and for loop strided

ordering between different iterations of the same loop

if it is a sort, you don't know the order of callback function, so can't take advantage of any vector ordering

having some kind of meaning on dependency is silly

for each i the logical one



Robert: transform is another logical one

Par\_vec is the most restrictive on programmer, avoid races and synchronization

(Hans: does not prohibit synchronization inside functions, same for unseq and vec side;

If you are spinning on same lock, you are ok

If different lanes each acquire different mutexes, you are ok

Par\_vec implementation will treat as sequential code if it sees synchronization

Now you may need to specify which library functions is allowed,

AI: will add action item for future SG1 on above)

Par side allows to synchronization, but cannot make any assumption on order of evaluation or how many threads

Unseq-vec side allows assumption on order of evaluation on iterations, but not allowed to make synchronizations between iterations

11.unseq works with anything

Vec supports different architecture

12. Long machines have a pipeline and arith unit that gets reused, at the same time as maincode moves onto next instructions

Before finished  $a(0)$ , can start on  $a(10)$ , and when  $A(0)$  is finished, it moves onto  $B(0)$  on long pipeline vector machines

Some of the length can be arbitrary

Modern uses wide registers that hold more than one val at a time, and they can do multiple step at once in predecided chunks

Can get 2x more floats and doubles in these registers

Difference between this and the vector data proposal, loops hide this and treats each partial register as a separate lane

If  $a(0)$ ,  $a(2)$  grabs the same mutex at once, likely deadlock, hence avoid synchronization is important

13. can do software pipelining

Can begin issuing  $b(0)$  until  $a(0)$  is done; can begin issuing  $a(2)$  while  $a(0)$  is starting

Can handle  $a$  again because instruction is freed up

Can before the next iteration before end of current iteration

14. what is in common, and what we can capitalize

Wavefront

Which is a sequence of instructions and guarantees

A somewhat predictable wavefront, no earlier application can fall behind a later application

You enabled forward dependencies for vectorizable code, and auto-vectorizable code

P0076r0 rules is complex, and being simplified

15. upper left to lower right

$B(0)$  and  $a(10)$  can be simultaneous

Such that there is a wavefront that propagates through the instructions

16. similarly

May or may not be instruction ordering

But still a wavefront that can propagate

Slanted because there is still an implicit order for some architectures

Are writes to memory going to be ordered? Not clear? AVX and SSE do write in order, so always get the last write

If they  $A(0)$  and  $a(1)$  writes to the same memory location, then  $a(1)$  wins because of the slant

17. software pipelining also has this slanted wavefront

Moving from corner to corner

Al Grant asked:

? Is it expected there will be one with ordering and one without?

Unseq is without

And vec has ordering

?within vec mode?

Yes SG1 needs to decide on this

If loading from unrelated thread?

No that will be undefined and is a race

Even an atomic load?an atomic racing with it

2 different lanes reading from the same location

Sequencing across thread, leave as unspecified

Bottom case, no way to get reasonable results

Has w before r dependency, so wavefront propagation is not good enough, so must be done sequentially

19. can turn off vectorization for regions of loop

No concurrency no vectorization at all

A way to do compress, cannot do it concurrently without a race

The entire vec\_off construct is treated executing left to right

20. a policy that derives from vector execution policy, that compiler can understand

Can pass to for loop

One way to add tuning parameters

Some can standardize: safelength (This is actually a semantic constraint)

Mechanism is standardized, the specific tuning parameters is portable

21. future has the histogram pattern of incrementing value inside an array with race

Or compress expand pattern, cant just index off control variables

22. because we don't really know how to specify it for others, but may be in time

23. summary

Main controversial points:

1. Whether wavefront should be implicit or explicit

Slide 18: whether the relation between the first and second line should be implicit

Whether  $v[i]$  when written when  $v[i-1]$  has been seen

Do we need some annotation to ensure that

Offline discussions in Silicon Valley, is that in history this stuff works without annotation, compiler can figure them out

1. So implicit Wavefronts; explicit in that you have created vec policy

2. Order of writes: similar to histogram problem, slide 21, if instead of ++, it is = some expression

And if  $b[i]$  is not unique, should it be defined behavior such that latest in sequential ordering should win; seem to be a corner case, Intel: last one wins; nvidia checked and say their hardware does not do that

Olivier: discussion off email list mostly talked about implicit vs explicit; there was wording for an explicit version, have an idea how to specify it, in the end; decided that continue forward implicit wavefront

Explicit wavefront wording opens door for another method, but hasn't been done yet

ON the racy store issue, still checking with nvidia hardware

Want us to put out spec that fits in well with Parallelism TS and Torvalds forward progress and apparent scalar thread of control and scalar lock of C++, also library functions restrictions

Brent: games care about data alignment; vector data type proposal probably has it

Anyway to deal with that issue separately

Clark's paper? Ability to specify that the data is aligned in a type is already in C++11

Can you do that with arbitrary pointer

Can create a local or static duration object of any alignment you want, but not heap allocated data

P0035r0 is Clark/Pablo paper that address that

No way to standardize a way to specify vector alignment, you just have to know

Robert: how to allocate memory such that its aligned?

It's the allocating the memory that is specified

Don't know a way to say to compiler that this is already aligned, should say that as well

Robert: also laying a classes as a formal structure of arrays vs array of structures

Also requires some communicating of the alignment stuff

SIMD uses SOA: can gain more performance

Sean M: How much template logic goes into these libraries; what kind of impact on compiler time

Pablo: will try to answer

To get vectorization, can not just do it with compiler independent code, needs pragma or intrinsics to identify the code

That can be done at a high level.

Compiler can recognize the for loop, as a vector loop, and do vector inside

Or use OpenMP pragmas and have template magic to translate them

Ok should not be any slower

Sean: not convinced; no pressure on compiler vendors will have no compile time effect

JF: clang wants to fix both compile and execution time aspects

Robert: modern compiler on OpenMP side, and the more they challenge the compiler to do more; from code generation perspective, we are proposing template;

2.2 Review proposals results from Kona: 20 min

<https://groups.google.com/a/isocpp.org/forum/?fromgroups#!topic/sg14/-Dw-jYbhlAc>

We won't have time to discuss in detail, but we can go over major questions and seek opinions and feedbacks

[P0037R0](#) Fixed point real numbers John McFarlane LEWG SG14/SG6: Lawrence

Crowl

[P0038R0](#) Flat Containers Sean Middleditch LEWG SG14: Patrice Roy

getting benchmark work  
initial draft to add container

[P0039R0](#) Extending raw\_storage\_iterator Brent Friedman LEWG SG14: Billy Baker  
dropped

[P0040R0](#) Extending memory management tools Brent Friedman LEWG SG14: Billy Baker

updated; most interested in; going to LWG

[P0041R0](#) Unstable remove algorithms Brent Friedman LEWG SG14: Billy Baker  
rejected due to lack of motivation

rewritten with better motivation

[P0059R0](#) Add rings to the Standard Library Guy Davidson LEWG SG14: Michael

[P0130R0](#) Comparing virtual functions Scott Wardle, Roberto Parolin EWG SG14:  
Michael

Scott Wardle: can go to game dev; finish first paper on virtual function comparisons

[P0125R0](#) std::bitset inclusion test methods: Michael/Walter

2.3 Ongoing topics placeholder (which we won't likely get to but will schedule)

Exceptions/RTTI

SIMD vector and Matrixes

Allocators

GPU/Accelerator design

Array View and Bounds checking

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

AMD's Heterogeneous Compute Compiler

+HSA research into latency effect for discrete and integrated GPU

## 5.2 Future meeting

Next call : Jan 20: 2 hours

Dec 9: Nvidia's Agency With Jared and Mike Garland - DONE

**Jan 13: 1st vector/SIMD call - 2 hours:** Archie/Pablo- DONE

Jan 20: AMD's HCC compiler with Ben Sander & HSA research

**Jan 27: 2nd vector/SIMD call- 2 hours:**Matthias proposal? Other proposal?

Feb 3: LSU's HPX runtime with Hartmut Kaiser

**Feb 10: possibly 3rd vector/simd call:** Other proposal?

Feb 17: Khronos' OpenCL C++ SYCL with Andrew Richards

## Minutes for 2016/01/20 SG14 Conference Call

Minutes by Michael

- hide quoted text -

2.3 HSA's research: 45 minutes

2.4 Review proposals results from Kona: 20 min

<https://groups.google.com/a/isocpp.org/forum/?fromgroups#!topic/sg14/-Dw-jYbhlAc>

We won't have time to discuss in detail, but we can go over major questions and seek opinions and feedbacks

[P0037R0](#) Fixed point real numbers John McFarlane LEWG SG14/SG6: Lawrence Crowl

[P0038R0](#) Flat Containers Sean Middleditch LEWG SG14: Patrice Roy

[P0039R0](#) Extending raw\_storage\_iterator Brent Friedman LEWG SG14: Billy Baker

[P0040R0](#) Extending memory management tools Brent Friedman LEWG SG14: Billy Baker

[P0041R0](#) Unstable remove algorithms Brent Friedman LEWG SG14: Billy Baker

[P0059R0](#) Add rings to the Standard Library Guy Davidson LEWG SG14: Michael

[P0130R0](#) Comparing virtual functions Scott Wardle, Roberto Parolin EWG SG14: Michael

[P0125R0](#) std::bitset inclusion test methods: Michael/Walter

2.3 Ongoing topics placeholder (which we won't likely get to but will schedule)

Exceptions/RTTI

SIMD vector and Matrixes

Allocators

GPU/Accelerator design

Array View and Bounds checking

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

**2nd vector/SIMD**

5.2 Future meeting

Next call : Jan 27

Dec 9: Nvidia's Agency With Jared and Mike Garland - DONE

**Jan 13: 1st vector/SIMD call - 2 hours:** Pablo - Done

Jan 20: AMD's HCC compiler with Ben Sander - 2 hours

**Jan 27: 2nd vector/SIMD call- 2 hours:**Matthias proposal? Other proposal?

Feb 3: LSU's HPX runtime with Hartmut Kaiser

**Feb 10: possibly 3rd vector/simd call:** Other proposal?

Feb 17: Khronos's OpenCL SYCL with Andrew Richards

## Minutes for 2016/02/03 SG14 Conference Call

Minutes by Michael

### 1.1 Roll call of participants

Brett Searles, Arch Robison, Lee Howles, Jarred Hoberock, Andrew Richard, Michael Garland, Michael Wong, Matt Newport, Scott Wardell, Hartmut Kaiser, Al Grant, Ben Sandres, Tony Tye, Maria , John McFarlane

### 1.2 Adopt agenda

Switch order.

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 F2F meeting hosted by SONY at GDC 2016  
Tentative: Tuesday March 15 and/or Wednesday March 16

<https://groups.google.com/a/isocpp.org/forum/?fromgroups#!topic/sg14/Euo2So0kYew>

1.4.2. All: Consider attending Jacksonville Fl, C++ Std meeting Feb 29-Mar 5.

<http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4555.pdf>

## 2. Main issues (125 min)

2.1 Review SG14/SG1 logistics, for upcoming telecons and meetings: 5 min

SG14 meeting moved to San Mateo.

Time of Wednesday is not idea.

### 2.2. LSU's HPX : 45 minutes

2. general purpose runtime system

works on single node, but all is for distributed

clusters as well in HPC, but also in cloud

HPX create a light weight user threading system, like TBB, PPL

don't want to specifically know what data is where  
load balancing comes in as an issue  
use 128 bit virtual address called AGAS  
can move things from A to B without updating all references to that object  
overhead to create thread is 800ns  
an active messaging system  
explicit support accelerators  
3. can outperform MPI and OpenMP  
no strings attached licensing  
4.  
took all C++11/14 and more and ported to HPX  
why reimplement? so everything works distributed as well  
also full parallelism and concurrency TS  
6. need integration of all these proposals in current state  
some commonality, but some things are missing  
need parallel ranges  
vectorization  
add distributed and high performance computing  
goal: make parallelism and concurrency features independent of external solutions including  
MPI, OpenMP  
7. 4 things to reason about  
1. thread safety limitations, this can run concurrently or that can run sequentially  
2. what goes first: sequence; needs a result to go ahead  
3. where to run, affinity,  
4. additional parameters like grain size, short items, then don't schedule a thread, select algo  
(kind of sort)  
8. + 9.  
Conceptified is required  
10. existing exec policies extends beyond just parallel algo  
11. need to run algo asynchronously  
with task execution policies  
par(task), seq(task)  
advantage with parallel algo is mitigate the join or the reduce, join has implicit barrier  
turn it into async algo, can do other things, can mitigate the effect  
HPC aims to threads of runtime in the hundreds of microseconds, plenty of other work to be  
performed while join is happening  
12. executors  
N4466 use exec agents to perform the work  
limits to parallel algo, but what to use it in broader context  
how do you run work, where, what sequence  
13. bulk async, single async, through executor traits, easy to implement, uniform to execute them  
async version returns a future  
we should reuse this in C++,  
14  
executor has fire and forget semantics  
and timed executor trait, at a time or duration

15. have NUMA awareness executors

16. added executor parameter, how many iterations in each run  
OpenMP static dynamic, guided,  
use `parameter_traits`

17. has to expose one function called `get_chunk_size`  
how many processing units we have and divide by the number of iterations

18. use `par.on` for executor context and extend with `par.with` for param  
can sequentially compose them to rebind both

19. will combine hundreds for each spawn thread underneath with chunk size,  
this is sufficient to make openMP parallel for loop unneeded in C++

21. future object allows to synchronize with another rthread and also data dependency explicit  
enable autparallelization at runtime without user make execution graph explicit  
will always be generated by compiler for you

22. adds sequential composition `.then`  
one future comes back from `async` and whenever `f1` is ready, then lambda is invoked, `f1` is  
passed through to lambda, `f.get` will give 123  
`f.get` will not suspend; lambda is invoked after future is ready  
on millions of threads, don't want large number to be suspended  
stack has stack space associated and will eat virtual address space  
composition is very important, better than calling `get`, then calling the function

23. parallel composition in concurrency ts

24. hpx did this with dataflow  
can extract value from the future only after you know future is ready  
dataflow guarantees only after future is ready  
this changes the whole game

25. also added optional first arg, so can pass executor to these functions, where to invoke and  
how to invoke  
can mix single futures and vector of futures  
and `when_some`, `wait_any`, `wait_all`, `wait_some`

27. sequential algo adds policy as first arg

28. shows rebind executor of `par.on()` or `par.with()` to control how many iterations per thread

29. how useful `async` algos are  
list of elements  
in a mail client, marked up that you want to move to a folder  
use a partitioning

31. `gather` can take multiple `Iter`, `marked`, and `make_pair`  
1st with inverted predicate  
2nd with straight predicate  
give insertion range

32. `gather_async` takes same set of arguments  
returns a future to a pair of iterators  
invoke `stable_partition` with task exec policy, gives future back, `f1` (returns immediately),  
`f2` (scheduled afterwards) and both can run in  
parallel using dataflow  
`unwrap` allows passing future along without blocking, helps get rid of future

async algo which returns an implicit async execution graph, which is built out of futures,  
a future represents a data dependency  
writing straight code, but does not execute immediately  
dataflow is awkward?

33. but with await and resumable functions will now makes this semantically equivalent to 32,  
but simpler

improves system utilization massively  
code is not impeded by the join, as long as I have other work to do

34. have task block proposal shows canonical fork join traversing a tree  
key function is `define_task_block` invokes a lambda, can run additional tasks  
all these tasks will be synchronized before `define_task_block` returns

so essentially a fork join block C++ way

36. adds 2 optional things

exec policy to make aysnc

add args to enable fine grain control of executors

37 shows these additions

38. future plans

GPU

distributed data structure

partition vector and each partition vector can be on different nodes, gpus and enable working on  
it with parallel algo

work has to follow data

39.

90% taken from existing

added 5% missing functionality to enable more things

avoid coming up 100 ways to do paralellism

a single versatile way for all different ways

uniform versatile generic

OpenMP difficult to use in generic way.

## 2.3 Khronos SYCL (tentative depending if he has time) : 45 minutes

Andrew Chair of SYCL group

focused on HW level, especially graphics, and the new Vulkan, SYCL is part of that

slide 3: how does SYCL compares to other models

what are the key decisions

also an overview

OpenCL 1.2 feature set

4: modern C++ separate s the what from the how

what do users want to do

details of SYCL: needs to enable users to do this, expose the performance capability of OpenCL  
devices

5: enable a C++ ecosystem for OpenCL, enables other STDs to work and C++ Lib to work with  
SYCL

must run on wide range of GPU devices/but OpenCL can run on CPU and GPU, FPGA, DSP

can build C++ template libraries

royal-free open Std

Michael: what would be the intellectual burden (IP) if we adapt any part of this for C++

may need to independently work it out

any vendor can implement SYCL

expose capability of HW

6: many ways to map C++ to parallel processors

follow established approaches

enabler for other to build C++ parallel STL on top SYCL

make sure all of the OpenCL HW capability are supported in SYCL

added no more then needed so as not to burden

8: SPIR is intermediate binary format allows to put other languages on top of OpenCL

SYCL has a C++ Compiler that outputs SPIR and a runtime for OpenCL

the same C++ template library on top of SYCL, could also be compile for CPU, Custom Processor,

9: align with C++ Std; future proof for future OpenCL device capabilities

other then a few little things, did not add any language extensions to C++; no restrict; no cv qualified for classes

means all your code can compile for host

provide full OpenCL feature set in C++, any thing you can do with OpenCL can now be done through C++

Hartmut: does model of SYCL assume the device is connected to the host, is it in the same node?

Not an assumption, not a requirement of SPEC; in HPC context it is assumed it is single node;

one company can offload code to another node in another network

Hartmut: HPX can access arbitrary devices in a cluster;

HPC context is really aimed at single node

multi node programming would be as a C++ library

10: looked at other language styles and where SYCL fits in

an embedded DSL:

problem with compile-time compilation approach, hard to express control flow, mapped calls that adds to expression calls, hard with expressibility, as you would have to add your own types

can get a library with this approach and can compile /port to SYCL

another approach is C++ Kernel language: host and kernel separate source

very explicit what is running where

independent host and device compilers

can generate code at runtime

very popular in graphics

hard to move code around, and define where the interface is

third is single source

easy to compose things

can have offline compilation, can type check

can conflict between host and device compiler, can have any device in OpenCL that is just plugged in, SYCL has Shared source as a solution

11: directive-based OpenMP/OpenACC

is popular in HPC, can incrementally parallelize serial code

hard to template things

code is out of order

debug is weird

2 is thread parallelism in C++11 thread

does not map well to highly parallel architecture like GPU

3 is explicit parallelism

composable,

requires user to know the parallelism

12

if its CPU, HSA, cache coherent single-address space;

can pass pointers around, low latency

bandwidth limited; costs power

SYCL needs wider range of devices

2 is non-coherent single address space

can still have pointers to access data, may need to pass ownership to device

race free software needs at least this level to control ownership

3 is multi-address space

pointers may have different address space

wide device support

just install a driver

13. 14.

need all the optimization options

how data is moved around, how they are accessed

must handle parallelism of openCL ND ranges (1,2,3 dimensional ranges, broken up into work groups, workgroup into work items, ...

multi-device from multi-vendor

enable OpenCL C code to work with SYCL

15.

to access data, must encapsulate data in a buffer

created 4 float 1D buffers

construct the buffers from the arrays

synthesize buffers from arrays

create a queue, critical for OpenCL, runtime will queue and find the device, fallback to cpu if none

submit work to the queue using lambda which is passed handler, will call command group handler

lambda is called immediately but only enqueue work

but kernel will run within the queue

Michael: is the lambda called by?

by the host

called by submitting side of the queue,

a parallel for that will be executed on the device

may enqueue work to move data to the device

have separated access from storage of data

a buffer cannot be accessed directly (no operators)

create accessors 3 read 1 write

for lifetime of accessors, it will manage define the lifetime

a few optional things like constant memory  
parallel for is executed on device with range  
end of submit scope will submit all this in atomic form  
the next end scope will wait for all operations to complete using RAI

16. access within that is to build up task graph  
work out dataflow  
and dependency

Tony: destructor of the get access guarantees Happens before that will get everything back  
up to a point, if do it on the host then yes it will instantly wait and release on destruction  
if inside a queue submit, then it would not do it immediately, then it will go into task block, will  
not block on construction of accessor on a task block, just enqueue

handler from concurrency group in C++

Lee: SYCL is not compiled so host compiler can see all dependencies  
so all dependencies are done at runtime, so why we need to use scope  
handler has a temp var that gives a lifetime

handler is a node in the task graph

Maria: all these submits ... can follow dependencies without state, all is explicit, need a read or  
write

all these submits are one kind of transactions

17.

accesses with different pointer sizes like FPGA

accesses allow you to asynchronously move data around and different memory systems  
define how you can use pointers

accessors can be raw pointers

18.

hierarchy of parallelism

limit on how many work items in a work group, size of work groups, data a work group can  
store, how many separate items of work

allow people to program this hierarchy in a C++ way

19.

single threaded buffer and queue submit

can have a parallel\_for\_workgroup, workgroup scope only execute once per workgroup

parallel\_for\_workitem, only 1 workitem in a workgroup is run

easy to have parallel\_for in parallel\_for

easy to have parallel reduce and pass in actual reduction functions

Michael: is reduction associative or commutative

assume people will write their own parallel reduce, operator may need to be commutative or  
associative

20.

can also be shared source

wants to support lots of devices with different compilers, diff options, diff algo for diff devices  
shared source means pass it through multiple compilers

device compiler extracts kernel from same file

allows FE to optimize for specific devices; give better performance and portability

21.

header file can include predefined compiled device file



accessors is task graph, implemented as opencl memory objects, can be pointers  
host can have different view then device and can see constants differently  
parallel for device compiler extracts content of that parallel for,  
no eh, no rtti, all functions inside must also conform  
the lambda has no name; on host cannot access the compiled kernel with the host code; which  
kernel

SYCL requires a templated name for lambda

Michael: what data can you capture

OpenCL 1.2 requires pass by value, no ref, classes must be standard layout  
data is captured on host by handler and read on the device by body of lambda,  
same code being compiled, if data captured on host then it (host runtime) will be passed as an  
argument

if an accessor, then go through OpenCL access system

Shared from slide 12 is not used

23. can write parallel embedded DSL, a key benefit of SYCL

working with tensorflow

24. 2.0 has shared virtual memory so can share pointers

1.2 SPIR is optional. 2.0 SPIR-v is required

enable separate source for kernel and host

General discussions

reducing latency

embeded , HPC, consumer devices marry

embrace latency

hide latency

avoid starvation, contention points

do useful networking in the background

increase 90%

expose parallelism

support memory Hierarchy, affinity,

reduce contention

can we do this with futures?

move enabled futures, or handles

physical location of data matters

when is data allocated in a future?

data enters future by applications  
continuation function allocates the data  
return with unique ptr or move semantics, or put it in global namespace and use references and do synchronization yourself  
SYCL preallocate on host, but dynamically allocated on device  
can we return a future parameterized by the set of accesses with the set of handle in prior future  
future should not block  
diamond flowcontrol, requires await stuff  
basically what SYCL does, dont have access to the data until you get it back from the buffers, all asynchronous  
await does not suspend or block

- show quoted text -  
- show quoted text -  
Next call : Feb 10

Dec 9: Nvidia's Agency With Jared and Mike Garland - DONE

**Jan 13: 1st vector/SIMD call - 2 hours:** Pablo - Done

Jan 20: AMD's HCC compiler with Ben Sander - 2 hours

**Jan 27: 2nd vector/SIMD call- 2 hours:** Matthias proposal? Other proposal?  
Cancelled

Feb 3: LSU's HPX runtime with Hartmut Kaiser/Andrew Richards - 2 hours

**Feb 10: 3rd vector/simd call:** Matthias Kretz/Joel Falcou, Mathias Gunard

Feb 12: C++ Std meeting mailing deadline (no call)

Feb 17: Uniform install/Packaging proposal

## Minutes for 2016/02/10 SG14 Conference Call

Minutes by Michael

### 1.1 Roll call of participants

Michael Wong, Xinmin Tian, Mattias Kretz, Billy Baker, Marco POCO, Visal, Arch Robison, Robert Geva, Ewo from Codeplay, Detlef Vollman, Mattias Gaunard, Michael Garland, JF Bastien, Hans Boehm, Al Grant, Jens Maurer, Lees Howe Facebook, Jared

### 1.2 Adopt agenda

Yes

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 F2F meeting hosted by ... at GDC 2016

Tentative: Monday March 14 or Wednesday March 16

finalized to be March 14, Feb 29 mailing deadline

<http://wongmichael.com/2016/02/09/c-standard-sg14-meeting-games-devlow-latencyfinancial-at-gdc-2016/>

<http://wongmichael.com/2016/02/09/c-standard-sg14-meeting-games-devlow-latencyfinancials-at-cppcon-2016/>

sg14 cppcon 2016 is Sept 21

<https://groups.google.com/a/isocpp.org/forum/?fromgroups#!topic/sg14/9CcqpRTe-yw>

Possible plan and venue changes.

1.4.2. All: Consider attending Jacksonville FL, C++ Std meeting Feb 29-Mar 5.

<http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4555.pdf>

### 2. Main issues (125 min)

## 2.1 Review SG14/SG1 logistics, for upcoming telecons and meetings: 5 min

### 2.0.1. Meta discussion

don't make decisions at these calls  
but expose issues

We are reversing order of presentation as it makes more sense

## 2.2. Matthias Gunard vector Proposal: 45 minutes

P203r0

comments on Kretz's paper

based on boost simd

agree on most details

concern how it may impact implementation

in Vc depends on target architecture, may not be able to do portable code

want some ability to get vectors of arbitrary size

extend on Vc to do that

a number of operations which are not provided for simplicity

3 operators missing are

shuffle to reorder elements inside vector

provide the index of the element want to reorder

sensitive to size and need to know in advance; could use metafunction

another is things that manipulate vector of different types,

want to convert float to int and same number integers

finally architecture with different sizes of simd registers

x86 avx 128 and 256, need way to convert between the 2

arm also has that; not all operations are available in both types

full multiply and add is an example

how much freedom will be in size of vectors

simd vector of 2 floats, can I do it as vector of 4 floats

syntax Vc requires expression template, explore extending language instead

e.g. condition operator needs this translates to inline if

extends ternary operator to overload it

Kretz: do not use expression template in Vc, conditional can be done without it

want different syntax anyway, SG1 has a different syntax, will need to go through different syntax.

SG1 has not tied proposals to language changes

not entirely sure it can be done without expression templates,

take offline, should be doable

argument made that u can write code that is agnostic of the size

bit idealistic

number of application where you need a fixed sized

Kretz agree

just don't agree how we want to have the low level types

JF: what is meant by portability can you expand

I mean source portability

others mean pre-compiled code, not quite binary compatibility

please clarify in the paper that it is source

what is shuffle issue? are u working with chandler

Kretz: want to keep it out unless we have a good solution in Lenexa, with deadline approaching fast, will not have solution

good chance to have Mattias Guanard take the lead there if he has a solution

still an open issue

need fixed size simd vector with shuffle

unwritten simd code which uses ???

no question we need shuffle

question how to shift something and add shuffle later, or now

sure no issue with it later on

does not change binary compatibility

main question is how to add it independently

slides: x is by reference, what if I modified

it is written back to container, this is what lambda does

for each element it is read and write

want to hide away loads and stores

does this interface require to read memory?

no

OK

Gunard: Want to do aliasing?

Kretz: Not very good idea. use the non-const subscript to do aliasing

Good for an Sg1 discussion

it is important and can be done efficiently in all compilers

Gunard: is everything set on the ABI requirements?

not happy with it

problematic target of x86 with avx vector on 512 and 256 differences

different TUs will have different types

simd vector type call from different translation unit,

will need native vector width; prefer to see default if not specified is the largest vector width

will get abi if with different compiler flag

JF: can be shipped without that support

if default not changed later on

Pablo: this is QOI? right

cant change the default from one C++ to another

default is dependent on implementation, then QOI cannot change the default

people often use simd as extensions

will link code built for different targets

C++ may need to support this

Pablo: if width of simd type is encoded in the type

not for derived type

generate multi version

Geva; what do you mean no link error

point\_v and float\_v will be independent

but get undefined behavior if you call the code

ABi is about performance, so u need understanding: a foot gun

Geva: language is not specifying abi

cross lane operations:

extraction and insertion

cross lane operations

vectorABI

Michael: are we that far apart

abi problem can make a pitch for solution for it

should talk to Torvald to make abi stable

Geva: have abi that call different translation unit when working on OpenMP simd

### 2.3 Matthias **Kretz**/Joel Falcou vector proposal: 45 minutes

2. not collaborated with Joel, he is working with Gunard on boost.simd

3. have packed registers in CPU

4. have natural size, how many value can fit into registers, and how many value can process in parallel

if we build something with a data parallel storage xpress in allo

5. can be an arbitrary number of element in x,

w is width of the vector

multiply however many elements makes sense for the target system on the right

auto vec is nice, but not good solution

information gets lost,

2 other ways to express parallelism: control structures to execute in parallel

other way is threads; its an abstraction for independent control flow; will need explicit synchronization

hw level needs

?how is this different from valarray

it is runtime sized so cannot optimize at compile time

will have to copy data, access data on heap; compiler cannot optimize it to full extent

cannot get to 400% that intrinsic could get

float\_v is statically sized, still need a loop

slide not obvious that we have loop involved.

6. stored by compiler at compile time

mask is also dependent on T

type can compile to different system for portability

7. express data parallelism, not register specific

even though came from motivation from what to abstract ugly intrinsic away

not programming for specific HW anymore

8. do all this in parallel as much as HW allows

9. need to know how many u process; so how do you do your loop?

loads and stores are the main issue

? why does it need to be a compile time constant

W(t) want compiler to optimize for target HW, compiler need to know

if compiling for several width of vectors, compile multiple times for different arch

10. float\_v will have multiple answers

$x > 0.f$  will not return a boolean

11. do condition assignment instead of branching

12. can get vector type to get speed up on various problems

13. every op implies lock step

each operation internally can run with any other model, most ops does not have inter lane dependency

chance for compiler to take it out

14. transformation to make it work on GPU with M Gunard

15. have memory access patterns, both AoS and SoA are bad choices

access to x,y,z would fall apart in memory and not be very efficient

Alternative is on slide 16

16. for array of vectorized structures

compiler will transparently make this happen

enable more performance portability

17. most of input output has to be scalar objects

enables broadcast, load store, subscripting

if u have to do this for every structure, very hard

want it automatically

write scalar structure, tell compiler to vectorize, and get new types, and get new functionalities

works great with tuples to get reflection

anything that works wit tuple can with this simdized structure

18. this shows how we do that

do nearest neighbor search

(4,5) works in parallel, then do reduction, what index value in the set

parallelize nicely for calculation of distance square

19. scalar is an exhaustive search

vectorized gives speedup

auto vectorization has a lot of trouble to figure this out

transformation cannot be done by compiler because data size is fixed to language

20. templated to type you are working with for float and float\_v

syntax for conditional assignment

21. use parallel algo from SG1

use a generic lambda so it can be scalar or vector type

to get rid of loads and stores that you would have to do manually

parallel loop needed at language level, will have parallel execution of different steps in the loop

have issues with exception, i/o in lambda body,

but no problem with simd in lambda body

22. loop vectorization need countable loop

when 1 is found, wants to break out

this is an uncountable loop

23. go over all the pixels in picture, determine if it is in the set or not  
 take multiple pixels in x direction, and calculate in parallel  
 inner loop that does the serious calculations, number of iteration is different for different pixels,  
 capture it with a mask  
 before going to y loop  
 some will drop out earlier  
 will be full speedup of mandelbrot  
 searching in multi dimensions with k-d tree  
 want interface that is scalar,  
 modified k-d tree on a single point whether to left or right  
 type T is the point with z,y,z, internally will use that as the node  
 from the outside not clear anything simd is going on  
 once with reflection in C++, can be nicer and see through it  
 25: graph of kd performance  
 26. 6 dim  
 27. 9 dim more speedup  
 28. what we want to have in our vectors  
 options to parallelize;  
 decide for one of the region, that we want simd  
 horizontal from benoit, author of eigen  
 do I take the same data member from several objects? horizontal  
 take data member in one object and process those similarly: vertical  
 think of markets  
 29. distance square function, iterate over the complete container, and look for entry in container  
 with distance square  
 scalar is exhaustive search  
 30. where is parallelism  
 show horizontal vs vertical  
 31. same function, par sq a-ref, do 3 of those in parallel and sum over result  
 or we take multiple points and calculate whole thing in parallel  
 32 shows graphical representation for vector of 4 entries  
 3/4th of the vector in use.  
 33. create a point v, then store multiple distances  
 sets that are interleaved and look for nearest neighbor  
 use remaining 4 or 8 and get the best one  
 34  
 no reduction in for loop  
 only reduction in line 12 and 13  
 often smaller objects should do horizontal; if 16 entries does not make sense to just use 3 of  
 those  
 few but large objects linpack should use vertical vectorization  
 35. vertical less invasive to code can hide it in function  
 horizontal depends on good data structure

Michael: should submit paper as is  
 can add new paper describing set up for solutions.



- hide quoted text -

2.4 Review proposals results from Kona: 20 min

<https://groups.google.com/a/isocpp.org/forum/?fromgroups#!topic/sg14/-Dw-jYbhlAc>

We won't have time to discuss in detail, but we can go over major questions and seek opinions and feedbacks

[P0037R0](#) Fixed point real numbers John McFarlane LEWG SG14/SG6: Lawrence Crowl

[P0038R0](#) Flat Containers Sean Middleditch LEWG SG14: Patrice Roy

[P0039R0](#) Extending raw\_storage\_iterator Brent Friedman LEWG SG14: Billy Baker

[P0040R0](#) Extending memory management tools Brent Friedman LEWG SG14: Billy Baker

[P0041R0](#) Unstable remove algorithms Brent Friedman LEWG SG14: Billy Baker

[P0059R0](#) Add rings to the Standard Library Guy Davidson LEWG SG14: Michael

[P0130R0](#) Comparing virtual functions Scott Wardle, Roberto Parolin EWG SG14: Michael

[P0125R0](#) std::bitset inclusion test methods: Michael/Walter

2.3 Ongoing topics placeholder (which we won't likely get to but will schedule)

Exceptions/RTTI

SIMD vector and Matrixes

Allocators

GPU/Accelerator design

Array View and Bounds checking

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

**C++ Install packaging proposal from Activision/Blizzard**

5.2 Future meeting

Next call : Feb 17

Dec 9: Nvidia's Agency With Jared and Mike Garland - DONE

**Jan 13: 1st vector/SIMD call - 2 hours:** Pablo - Done

Jan 20: AMD's HCC compiler with Ben Sander - 2 hours

**Jan 27: 2nd vector/SIMD call- 2 hours:**Matthias proposal? Other proposal?

Cancelled

Feb 3: LSU's HPX runtime with Hartmut Kaiser/Andrew Richards - 2 hours- Done

**Feb 10: possibly 3rd vector/simd call:** Matthias Kretz/Joel Falcou, Matthias Gunard

Feb 12: C++ Std meeting mailing deadline (no call)

Feb 17: Uniform install/Packaging proposal

Feb 24: low level bit manipulation proposal