codeplay®

THE HETEROGENEOUS SYSTEMS EXPERTS

C++17, is it great or just OK
All Features after Issaquah Nov 2017 C++ Standard Meeting

Michael Wong (Codeplay Software, VP of Research and Development), Andrew Richards, CEO

ISOCPP.org Director, VP http://isocpp.org/wiki/faq/wg21#michael-wong

Head of Delegation for C++ Standard for Canada

Vice Chair of Programming Languages for Standards Council of Canada

Chair of WG21 SG5 Transactional Memory
Chair of WG21 SG14 Games Dev/Low Latency/Financial Trading/Embedded

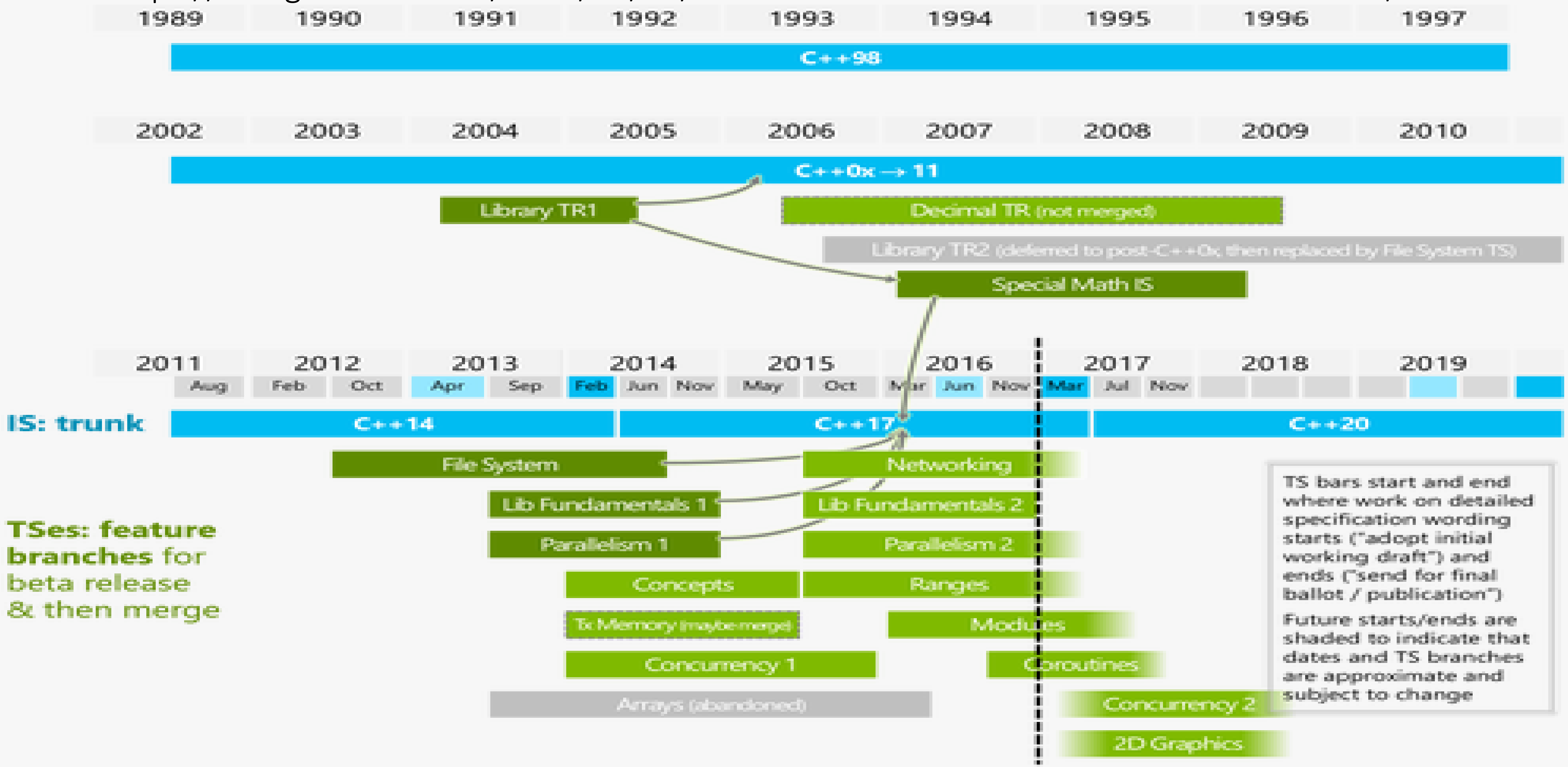
Editor: C++ SG5 Transactional Memory Technical Specification

Editor: C++ SG1 Concurrency Technical Specification

http:://wongmichael.com/about

C++ Std Timeline/status

https://wongmichael.com/2016/06/29/c17-all-final-features-from-oulu-in-a-few-slides/



Pre-C++11 projects

ISO number	Name	Status	What is it?	C++17?
ISO/IEC TR 18015:2006	Technical Report on C++ Performance	Published 2006 (ISO store) Draft: TR18015 (2006-02-15)	C++ Performance report	No
ISO/IEC TR 19768:2007	Technical Report on C++ Library Extensions	Published 2007-11-15 (ISO store) Draft: n1745 (2005-01-17) TR 29124 split off, the rest merged into C++11	Has 14 Boost libraries, 13 of which was added to C++11.	N/A (mostly already included into C++11)
ISO/IEC TR 29124:2010	Extensions to the C++ Library to support mathematical special functions	Published 2010-09-03 (ISO Store) Final draft: n3060 (2010-03-06). Under consideration to merge into C++17 by p0226 (2016-02-10)	Really, ORDINARY math today with a Boost and Dinkumware Implementation	YES
	Extensions for the programming language	Published 2011-10-25 (ISO Store) Draft: n2849 (2009-03-06)	Decimal Floating Point decimal32	

Status after Nov Issaquah C++ Meeting

ISO number	Name	Status	links	C++17?
ISO/IEC TS 18822:2015	C++ File System Technical Specification	Published 2015-06-18. (ISO store). Final draft: n4100 (2014-07-04)	Standardize Linux and Windows file system interface	YES
ISO/IEC TS 19570:2015	C++ Extensions for Parallelism	Published 2015-06-24. (ISO Store). Final draft: n4507 (2015-05-05)	Parallel STL algorithms.	YES but removed dynamic execution policy, exception_lists, changed some names
ISO/IEC TS 19841:2015	Transactional Memory TS	Published 2015-09-16, (ISO Store). Final draft: n4514 (2015-05-08)	Composable lock-free programming that scales	No. Already in GCC 6 release and waiting for subsequent usage experience.
ISO/IEC TS 19568:2015	C++ Extensions for Library Fundamentals	Published 2015-09-30, (ISO Store). Final draft: n4480 (2015-04-07)	optional, any, string_view and more	YES but moved Invocation Traits and Polymorphic allocators into LF TS2
ISO/IEC TS 19217:2015	C++ Extensions for Concepts	Published 2015-11-13. (ISO Store). Final draft: n4553 (2015-10-02)	Constrained templates	No. Already in GCC 6 release and and waiting for subsequent usage

codeplay

Status after Nov Issaquah C++ Meeting

ISO number	Name	Status	What is it?	C++17?
ISO/IEC TS 19571:2016	C++ Extensions for Concurrency	Published 2016-01-19. (ISO Store) Final draft: p0159r0 (2015-10-22)	improvements to future, latches and barriers, atomic smart pointers	No. Already in Visual Studio release and waiting for subsequent usage experience.
ISO/IEC DTS 19568:xxxx	C++ Extensions for Library Fundamentals, Version 2	DTS. Draft: n4564 (2015-11-05)	source code information capture and various utilities	No. Resolution of comments from national standards bodies in progress
ISO/IEC DTS 21425:xxxx	Ranges TS	In development, Draft n4569 (2016-02-15)	Range-based algorithms and views	No. Wording review of the spec in progress
ISO/IEC DTS 19216:xxxx	Networking TS	In development, Draft n4575 (2016-02-15)	Sockets library based on Boost.ASIO	No. Wording review of the spec in progress.
	Modules	In development, Draft p0142r0 (2016-02-15) and p0143r1 (2016-02-15)	A component system to supersede the textual header file inclusion model	No. Initial TS wording reflects Microsoft's design; changes proposed by Clang implementers expected.

Status after Nov Issaquah C++ Meeting

ISO number	Name	Status	What is it?	C++17?
	Numerics TS	Early development. Draft p0101 (2015-09-27)	Various numerical facilities	No. Under active development
ISO/IEC DTS 19571:xxxx	Concurrency TS 2	Early development	Exploring executors, synchronic types, lock-free, atomic views, concurrent data structures	No. Under active development
ISO/IEC DTS 19570:xxxx	Parallelism TS 2	Early development. Draft n4578 (2016-02-22)	Exploring task blocks, progress guarantees, SIMD.	No. Under active development
ISO/IEC DTS 19841:xxxx	Transactional Memory TS 2	Early development	Exploring on_commit, in_transaction.	No. Under active development.
	Graphics TS	Early development. Draft p0267r0 (2016-02-12)	2D drawing API	No. Wording review of the spec in progress
ISO/IEC DTS 19569:xxxx	Array Extensions TS	Under overhaul. Abandoned draft: n3820 (2013-10-10)	Stack arrays whose size is not known at compile time	No. Withdrawn; any future proposals will target a different vehicle

Status after Nov Issaquah C++ Meeting

ISO number	Name	Status	What is it?	C++17?
	Coroutine TS	Initial TS wording will reflect Microsoft’s await design; changes proposed by others expected.	Resumable functions	No. Under active development
	Reflection TS	Design direction for introspection chosen; likely to target a future TS	Code introspection and (later) reification mechanisms	No. Under active development
	Contracts TS	Unified proposal reviewed favourably.)	Preconditions, postconditions, etc.	No. Under active development
	Massive Parallelism TS	Early development	Massive parallelism dispatch	No. Under active development.
	Heterogeneous Device TS	Early development.	Support Hetereogeneous Devices	No. Under active development.
	C++17	On track for 2017	Filesystem TS, Parallelism TS, Library Fundamentals TS I, if constexpr, and various other enhancements are in. See slide 44-47 for details.	YES

Library Fundamental TS 2: being reviewed

- Source-code information capture (really a Reflection feature with a library interface)
- A generalized callable negator
- Uniform container erasure
- GCD and LCM functions (GCD/LCM moved into C++17)
- Delimited iterators
- observer_ptr, the world's dumbest smart pointer
- A const-propagating wrapper class
- make_array
- A metaprogramming utility dubbed the "C++ detection idiom"
- A replacement for std::rand()
- Logical type traits.

C++ 17 Language features already voted in

- static_assert(condition) without a message
- Allowing auto var{expr};
- Writing a template template parameter as template <…> typename Name
- Removing trigraphs
- ***Folding expressions***
- std::uncaught_exceptions()
- Attributes for namespaces and enumerators
- Shorthand syntax for nested namespace definitions
- u8 character literals
- Allowing full constant expressions in non-type template parameters
- Removing the register keyword, while keeping it reserved for future use
- Removing operator++ for bool
- Making exception specifications part of the type system.
- __has_include(),
- Choosing an official name for what are commonly called "non-static data member initializers" or NSDMIs. The official name is "default member initializers".
- A minor change to the semantics of inheriting constructors

- The [[fallthrough]] attribute,
- The [[nodiscard]] attribute,
- The [[maybe_unused]] attribute
- Extending aggregate initialization to allow initializing base subobjects.
- Lambdas in constexpr contexts
- Disallowing unary folds of some operators over an empty parameter pack
- Generalizing the range-based for loop
- *Lambda capture of *this by value*
- Relaxing the initialization rules for scoped enum types.
- Hexadecimal floating-point literals

C++17 Language features voted in Oulu Finland

`if constexpr` (formerly known as constexpr_if, and before that, static_if)

`Template parameter deduction for constructors`

`template <auto N>`

`Inline variables`

`Guaranteed copy elision`

`Guarantees on expression evaluation order`

`Dynamic memory allocation for over-aligned data`

`is_contiguous_layout` (really a library feature, but it needs compiler support)

`Removing exception specifications`

`Using attribute namespaces without repetition`

`Replacement of class objects containing reference members`

`Standard and non-standard attributes`

`Forward progress guarantees: Base definitions`

`Forward progress guarantees for the Parallelism TS`

- Introducing the term 'templated entity'
- Proposed wording for structured bindings
- Selection statements with initializer
- Explicit default constructors and copy-list-initialization
- Not in C++17 (yet!)
  - Default comparisons
    - For/against/neutral: 16/31/20
  - Operator dot
    - Not moved as CWG discovered a flaw

C++17 Library features already voted in

- Removing some legacy library components
- Contiguous iterators
- Safe conversions in unique ptr<T[]>
- Making std::reference wrapper trivially copyable
- Cleaning up noexcept in containers
- Improved insertion interface for unique-key maps
- void t alias template
- invoke function template
- Non-member size(), empty(), and data() functions
- Improvements to pair and tuple
- bool constant
- shared mutex
- Incomplete type support for standard containers
- Type traits variable templates.
- as_const()
- Removing deprecated iostreams aliases
- Making std::owner less more flexible
- Polishing <chrono>
- Variadic lock guard
- Logical type traits.

- Re-enabling shared from this
- *not fn*
- constexpr atomic::is always lock free
- Nothrow-swappable traits
- Fixing a design mistake in the searchers interface
- An algorithm to clamp a value between a pair of boundary values
- constexpr std::hardware {constructive,destructive} interference size
- A 3-argument overload of std::hypot
- Adding constexpr modifiers
- Giving std::string a non-const data() member function
- is callable, the missing INVOKE-related trait

C++17 Library features voted in Oulu Finland

- High-performance, locale-independent number <-> string conversions
- make_from_tuple() (like apply(), but for constructors)
- Letting folks define a default order<> without defining std::less<>
- Splicing between associative containers
- Relative paths
- C11 libraries
- shared_ptr::weak_type
- gcd() and lcm() from LF TS 2
- Deprecating std::iterator, redundant members of std::allocator, and is_literal
- Reserve a namespace for STL v2
- ***std::variant<>***
- Better Names for Parallel Execution Policies in C++17
- Temporarily discourage memory_order_consume
- A <random> Nomenclature Tweak

- Synopses for the C library
- Making Optional Greater Equal Again
- Making Variant Greater Equal
- Homogeneous interface for variant, any and optional
- Elementary string conversions
- Integrating std::string_view and std::string
- has_unique_object_representations
- Extending memory management tools
- Emplace Return Type
- Removing Allocator Support in std::function
- make_from_tuple: apply for construction
- Delete operator= for polymorphic_allocator
- Fixes for not_fn
- Adapting string_view by filesystem paths
- Hotel Parallelifornia: terminate() for Parallel Algorithms Exception Handling

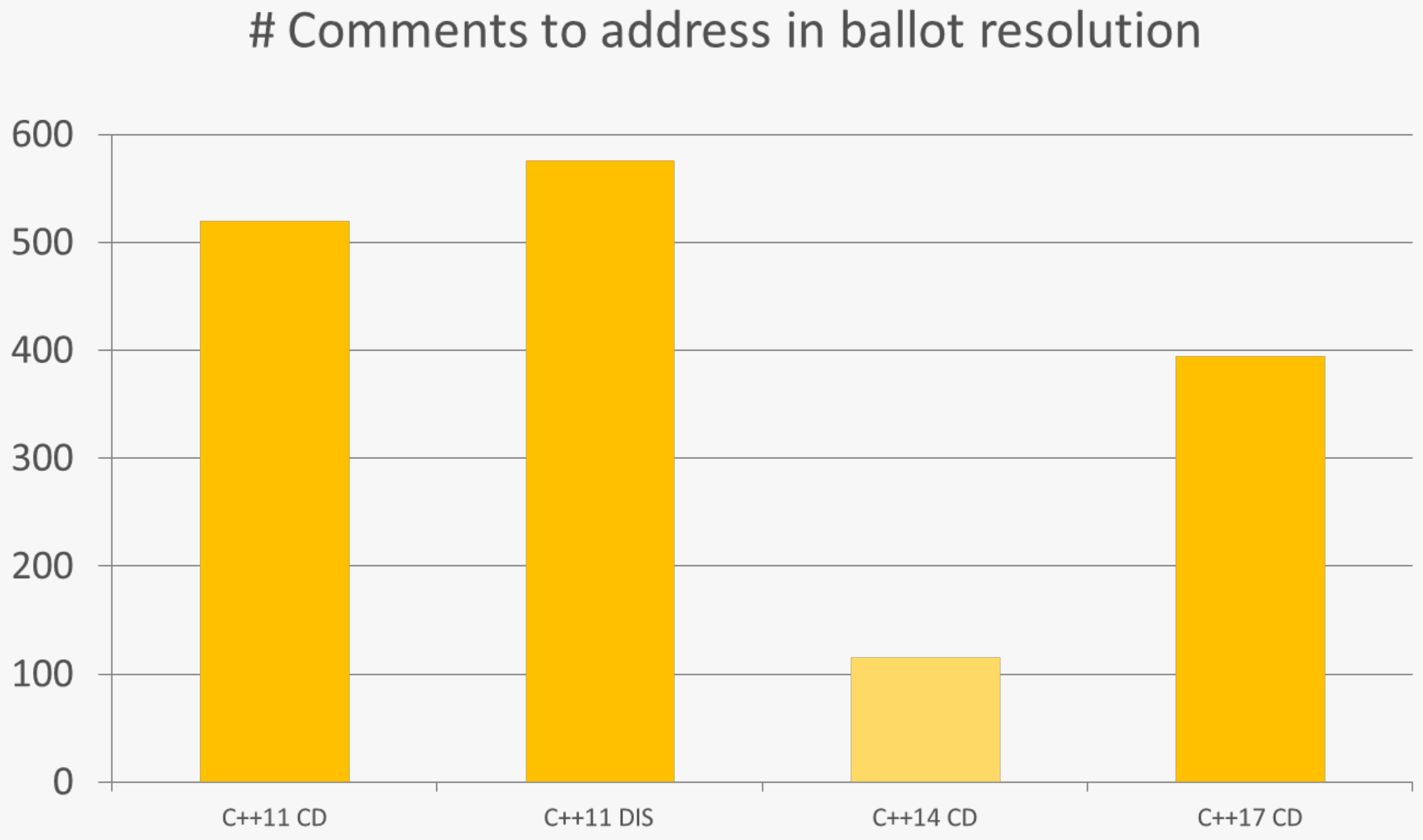
By the number of pages

- C++11 Std is
  - 1353 pages compared to 817 pages in C++03
- C++14 Std is
  - 1373 pages (N3937), n3972 (free)
- The new C++17 CD is
  - N4606: 1572 pages
- C99
  - 550 pages
- C11 is
  - 701 pages compared to 550 pages in C99

- OpenMP 3.1 is
  - 160 pages and growing
- OpenMP 4.0 is
  - 320 pages
- OpenMP 4.5 is
  - 359 pages
- OpenCL 2.0
  - 288 pages
- OpenCL 2.1
  - 300 pages
- OpenCL 2.2
  - 304 pages

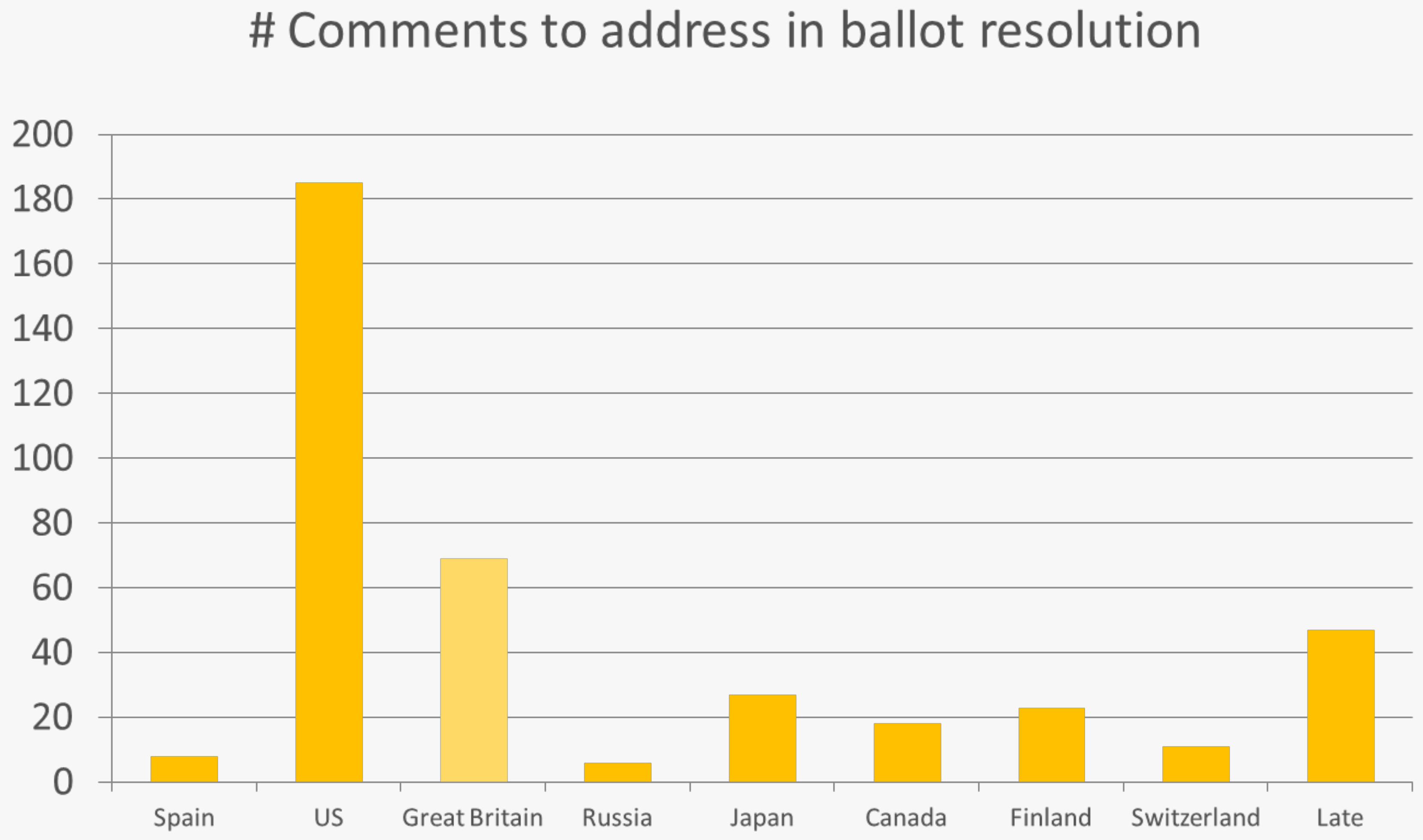
C++11/14/17: Stability

- Each round of international comment ballots generates bugs, tweaks, and requests



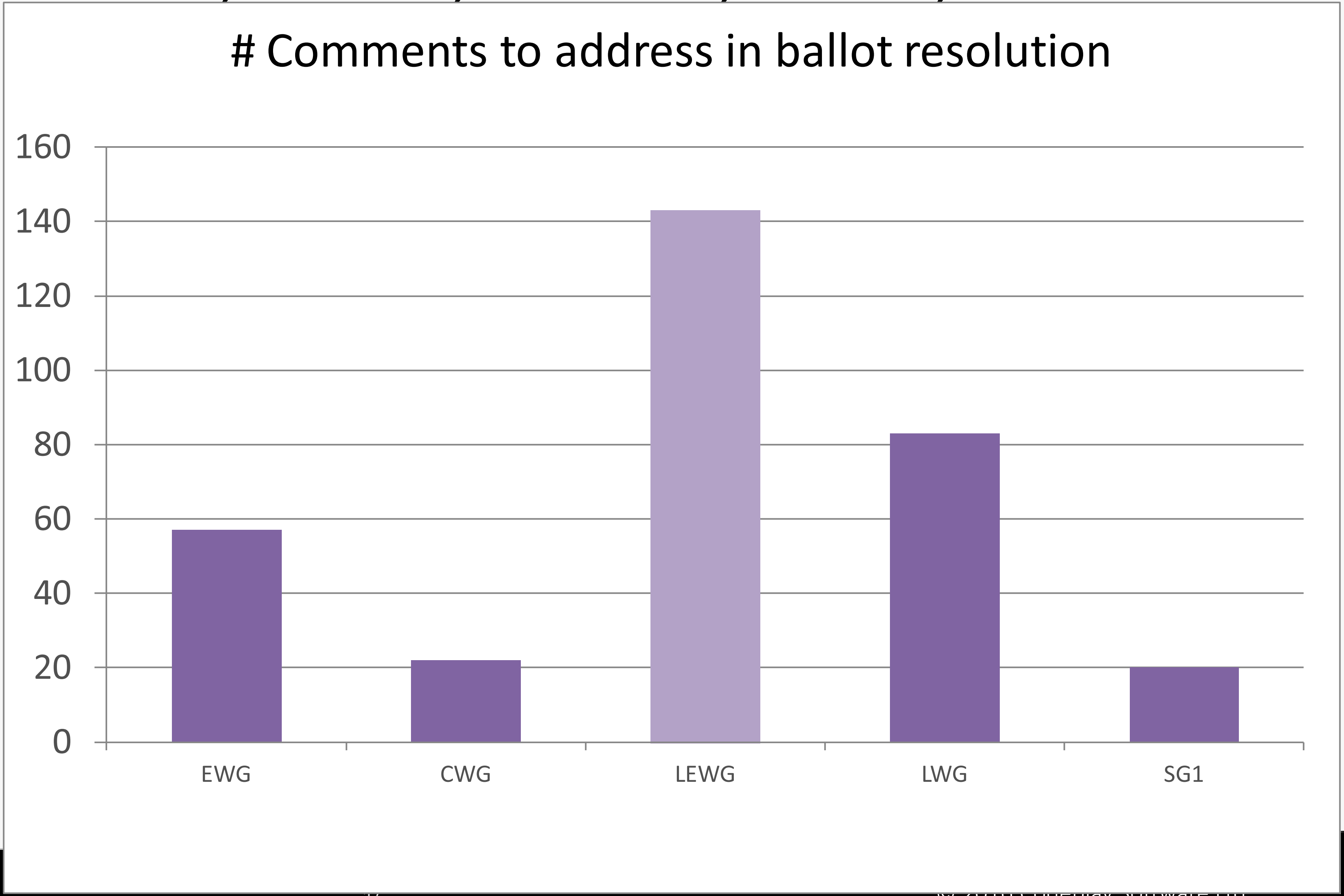
C++ 17: by Country

- Spain
- US
- Great Britain
- Russia
- Japan
- Canada
- Finland
- Switzerland
- Late



C++ 17: by EWG, CWG, LEWG, LWG, SG1

- Evolution
- Core
- Library Evolution
- Library
- Parallel/Concurrency



What did not change from Issaquah

No Concepts

No Unified Call Syntax

No Default Comparison

No operator dot

- Inline variable stays

Changes voted in Issaquah

Fixes to C+17

- Removing Deprecated Exception Specifications from C++17
- Added Elementary string conversions
- Std::byte was not added

Some New Features for C++20

- Pack expansions in *using-declarations*
- Lifting Restrictions on requires-Expressions
- Networking and Ranges TS feature freeze
- Modules and Coroutines almost ready
- Parallelism 2, Concurrency 2 and Reflection continue to incubate

Future C++ Standard schedules

- After Nov, Issaquah
  - Address additional returned comments in February Kona
  - Likely Issue DIS after Kona, Feb 2017, send it to National Body for final approval ballot; this is just an up/down vote, no comments
  - Most likely approved, then celebrate in July 2017 Toronto Meeting
  - Then send it to ISO Geneva for publication, likely by EOY 2017
- After C++17
  - Default is 3 yr cycle: C++20, 23
- C++20 prediction
  - Concepts, ranges, Concurrency TS1/TS2, Parallelism TS2, Executor TS1, Coroutine TS1, Networking TS1, Modules TS1, Transactional Memory TS1, Numerics TS1, Heterogeneous TS1

Improve support for large-scale dependable software

- ❌ Modules
  - to improve locality and improve compile time; n4465 and n4466
- ❌ Contracts
  - for improved specification; n4378 and n4415
- 🙂 A type-safe union
  - probably functional-programming style pattern matching; something based on my Urbana presentation, which relied on the Mach7 library: Yuriy Solodkyy, Gabriel Dos Reis and Bjarne Stroustrup: Open Pattern Matching for C++. ACM GPCE'13.

Provide support for higher-level concurrency models

- ❌ Basic networking
  - asio n4478
- ❌ A SIMD vector
  - to better utilize modern high-performance hardware; e.g., n4454 but I'd like a real vector rather than just a way of writing parallelizable loops
- ❌ Improved futures
  - e.g., n3857 and n3865
- ❌ Co-routines
  - finally, again for the first time since 1990; N4402, N4403, and n4398
- ❌ Transactional memory
  - n4302
- 🙂 Parallel algorithms (incl. parallel versions of some of the STL
- n4409

Simplify core language use and address major sources of errors

- Concepts (n3701 and n4361)
- concepts in the standard library
  - based on the work done in Origin, The Palo Alto TR, and Ranges n4263, n4128 and n4382
- default comparisons
  - to complete the support for fundamental operations; n4475 and n4476

May come back in limited form with National Body comment

- uniform call syntax
  - among other things: it helps concepts and STL style library use; n4474
- operator dot
  - to finally get proxies and smart references; n4477

May come back in limited form with National Body comment

- array_view and string_view
  - better range checking, DMR wanted those: "fat pointers"; n4480
- arrays on the stack
  - "stack_array" anyone? But we need to find a safe way of dealing with stack overflow; n4294
- optional
  - unless it is subsumed by pattern matching, and I think not in time for C++17, n4480

The Verdict on C++17? (from reddit)

- You blew it
- Not a Major release
- No risk, no gain
- Nobody implement TSs
- Tethering tower of Babel of TSs

- Did a nice job
- But not Minor either
- Safe and conservative wins
- TSs are implemented
- Followed the rules of a bus train model, how to get 110 people to work together

A Medium/OK Release

Codeplay

- HSA Foundation: Chair of software group, spec editor of runtime and debugging
- Khronos: chair & spec editor of SYCL. Contributors to OpenCL, Safety Critical, Vulkan
- ISO C++: Chair of Low Latency, Embedded WG; Editor of SG1 Concurrency TS
- EEMBC: members

- Members of EU research consortiums: PEPPHER, LPGPU, LPGPU2, CARP
- Sponsorship of PhDs and EngDs for heterogeneous programming: HSA, FPGAs, ray-tracing
- Collaborations with academics
- Members of HiPEAC

- HSA LLDB Debugger
- SPIR-V tools
- RenderScript debugger in AOSP
- LLDB for Qualcomm Hexagon
- TensorFlow for OpenCL
- C++ 17 Parallel STL for SYCL
- VisionCpp: C++ performance-portable programming model for vision

- Building an LLVM back-end
- Creating an SPMD Vectorizer for OpenCL with LLVM
- Challenges of Mixed-Width Vector Code Gen & Scheduling in LLVM
- C++ on Accelerators: Supporting Single-Source SYCL and HSA
- LLDB Tutorial: Adding debugger support for your target

- Based in Edinburgh, Scotland
- 57 staff, mostly engineering
- License and customize technologies for semiconductor companies
- ComputeAorta and ComputeCpp: implementations of OpenCL, Vulkan and SYCL
- 15+ years of experience in heterogeneous systems tools

2001 - 2003

VectorC for x86
Our VectorC technology was chosen and actively used for Computer Vision

First showing of VectorC{VU}

Delivered VectorC{VU} to the National Center for Supercomputing

VectorC{EE} released
An optimising C/C++ compiler for PlayStation®2 Emotion Engine (MIPS)

2005 - 2006

Ageia chooses Codeplay for PhysX
Codeplay is chosen by Ageia to provide a compiler for the PhysX processor.

Codeplay joins the Khronos Group

2007 - 2011

Sieve C++ Programming System released
Aimed at helping developers to parallelise C++ code, evaluated by numerous researchers

Offload released for Sony PlayStation®3

OffloadCL technology developed

Codeplay joins the PEPPHER project

2013

New R&D Division
Codeplay forms a new R&D division to develop innovative new standards and products

Becomes specification editor of the SYCL standard

2014

LLDB Machine Interface Driver released

Codeplay joins the CARP project

Codeplay shows technology to accelerate Renderscript on OpenCL using SPIR

2015

Chair of HSA System Runtime working group

Development of tools supporting the Vulkan API

2016

Open-Source HSA Debugger release

Releases partial OpenCL support (via SYCL) for Eigen Tensors to power TensorFlow

ComputeAorta 1.0 release

ComputeCpp Community Edition beta release
First public edition of Codeplay's SYCL technology

What our ComputeCpp users say about us

Benoit Steiner – Google TensorFlow engineer

"We at Google have been working closely with Luke and his Codeplay colleagues on this project for almost 12 months now. Codeplay's contribution to this effort has been tremendous, so we felt that we should let them take the lead when it comes down to communicating updates related to OpenCL. … we are planning to merge the work that has been done so far… we want to put together a comprehensive test infrastructure"

ONERA

"We work with royalty-free SYCL because it is hardware vendor agnostic, single-source C++ programming model without platform specific keywords. This will allow us to easily work with any heterogeneous processor solutions using OpenCL to develop our complex algorithms and ensure future compatibility"

Hartmut Kaiser -HPX

"My team and I are working with Codeplay's ComputeCpp for almost a year now and they have resolved every issue in a timely manner, while demonstrating that this technology can work with the most complex C++ template code. I am happy to say that the combination of Codeplay's SYCL implementation with our HPX runtime system has turned out to be a very capable basis for Building a Heterogeneous Computing Model for the C++ Standard using high-level abstractions."

WIGNER Research Centre for Physics

It was a great pleasure this week for us, that Codeplay released the ComputeCpp project for the wider audience. We've been waiting for this moment and keeping our colleagues and students in constant rally and excitement. We'd like to build on this opportunity to increase the awareness of this technology by providing sample codes and talks to potential users. We're going to give a lecture series on modern scientific programming providing field specific examples."

Further information

• OpenCL	https://www.khronos.org/opencl/
• OpenVX	https://www.khronos.org/openvx/
• HSA	http://www.hsafoundation.com/
• SYCL	http://sycl.tech
• OpenCV	http://opencv.org/
• Halide	http://halide-lang.org/
• VisionCpp	https://github.com/codeplaysoftware/visioncpp

SYCL™

ComputeCpp™

Community Edition

Available now for free!

Visit:

computecpp.codeplay.com

SYCL™

ComputeCpp™

- Open source SYCL projects:
  - ComputeCpp SDK - Collection of sample code and integration tools
  - SYCL ParallelSTL – SYCL based implementation of the parallel algorithms
  - VisionCpp – Compile-time embedded DSL for image processing
  - Eigen C++ Template Library – Compile-time library for machine learning

All of this and more at: http://sycl.tech codeplay®

THE HETEROGENEOUS SYSTEMS EXPERTS

Questions ?

@codeplaysoft

/codeplaysoft

codeplay.com