codeplay

THE HETEROGENEOUS SYSTEMS EXPERTS

The Birth and future of SG14 Low latency for HFT

Michael Wong

Codeplay Software, VP of Research and Development

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
Past OpenMP CEO: http://openmp.org/wp/about-openmp/

Chair of WG21 SG5 Transactional Memory: https://groups.google.com/a/isocpp.org/forum/?hl=en&fromgroups#!forum/tm
Chair of WG21 SG14 Games Dev/Low Latency/Financial Trading/Embedded:
https://groups.google.com/a/isocpp.org/forum/?fromgroups#!forum/sg14

Editor: C++ SG5 Transactional Memory Technical Specification

Editor: C++ SG1 Concurrency Technical Specification

http:://wongmichael.com/about

New York STAC 2016

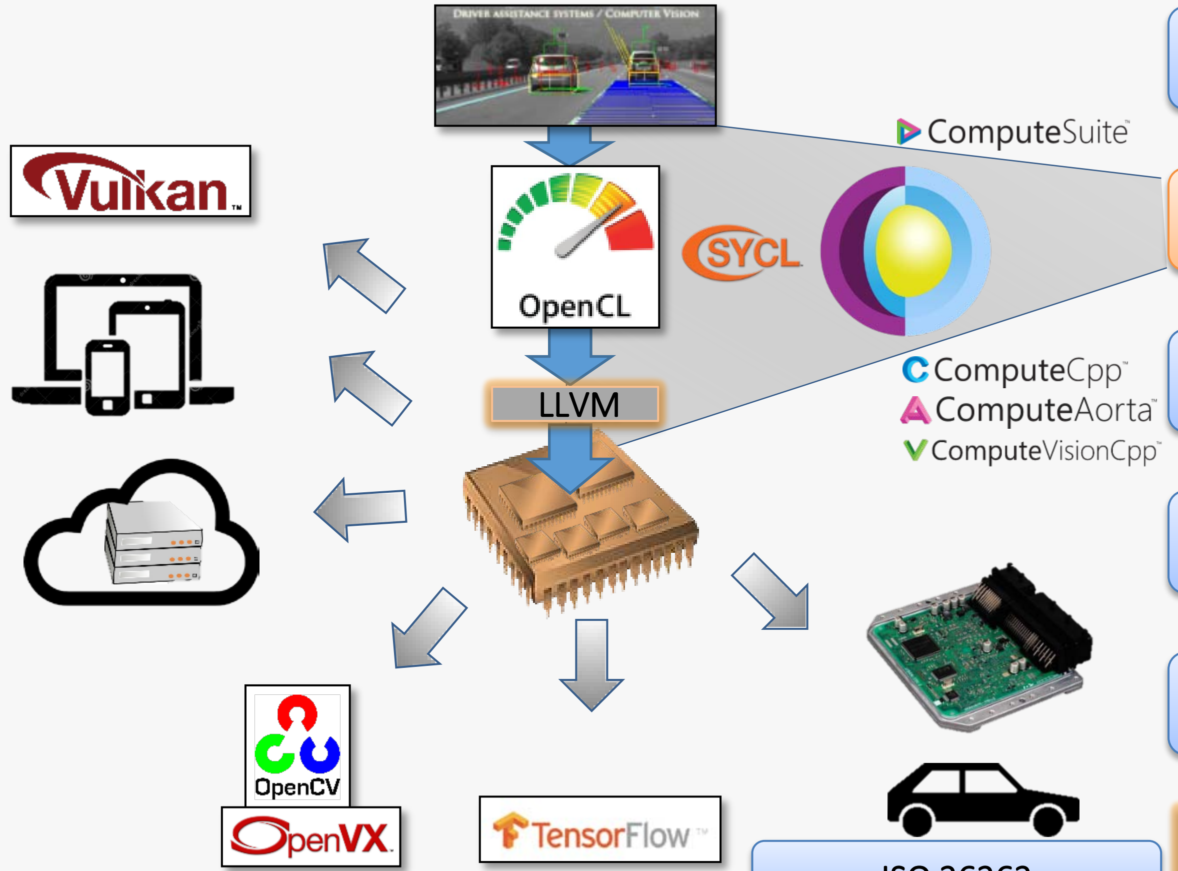
Acknowledgement and Disclaimer

- Numerous people internal and external to the original C++/Khronos/OpenCL/SYCL group, in industry and academia, have made contributions, influenced ideas, written part of this presentations, and offered feedbacks to form part of this talk.
- I even lifted this acknowledgement and disclaimer from some of them.
- But I claim all credit for errors, and stupid mistakes. **These are mine, all mine!**

Legal Disclaimer

- This work represents the view of the author and does not necessarily represent the view of Codeplay, IBM, Khronos, OpenMP, or ISOCPP.org.
- Other company, product, and service names may be trademarks or service marks of others.
- This is part of an ongoing keynote presentation for upcoming C/C++ Standard status modified as C and C++ changes. Please attribute credit accordingly.

Codeplay: world expert in Heterogeneous software platform for self-driving cars, AI/machine learning/neural networks, computer vision, data centres, graphics, mobile devices, with Open Standards



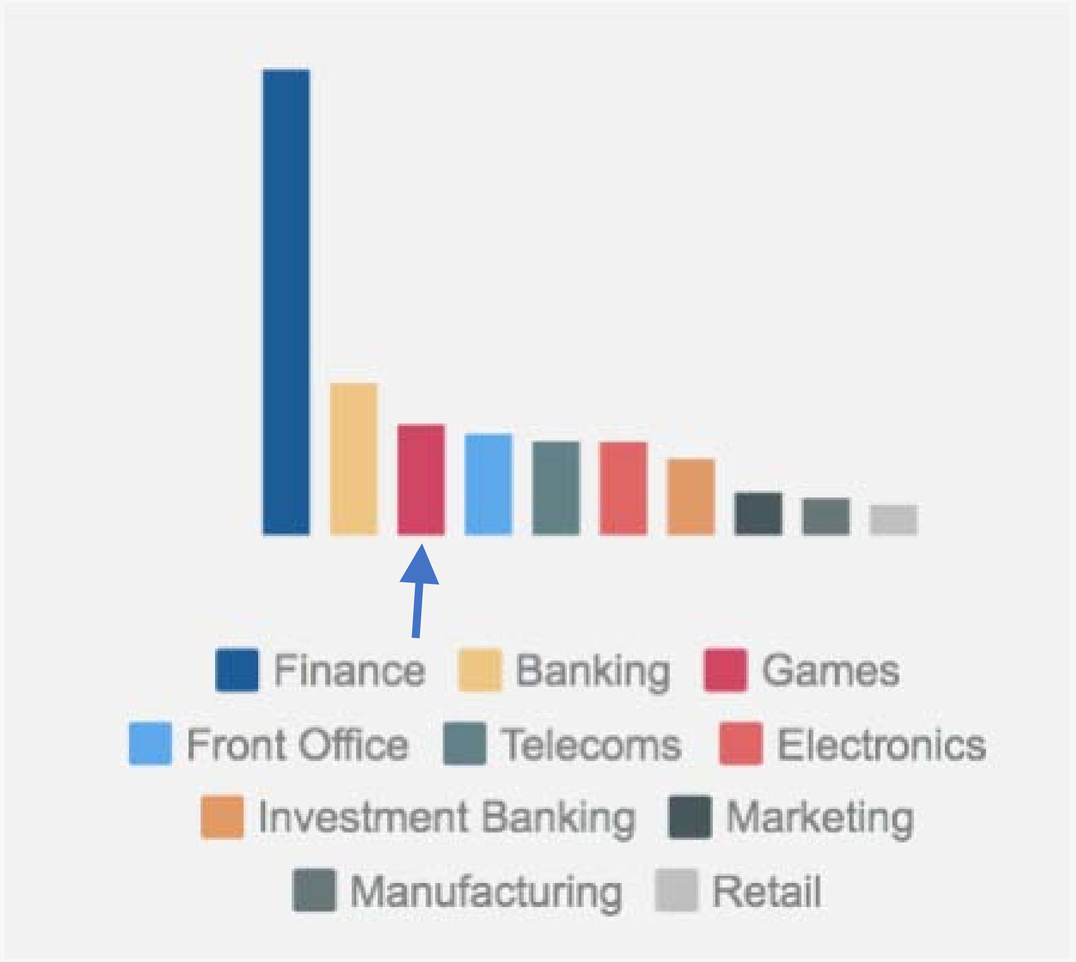
Major Corporation

- Algorithms
- Software/Middleware Tools & layers to Standards
- Processors IP CPU, GPU, DSP, VPU, ViSP
- Semiconductors
- Module Integration

ISO 26262

ISO C99/11 or C++ 11/14/17/20 compliant

Among the top users of C++!



http://blog.jetbrains.com/clion/2015/07/infographics-cpp-facts-before-clion/

The Breaking Wave: N4456

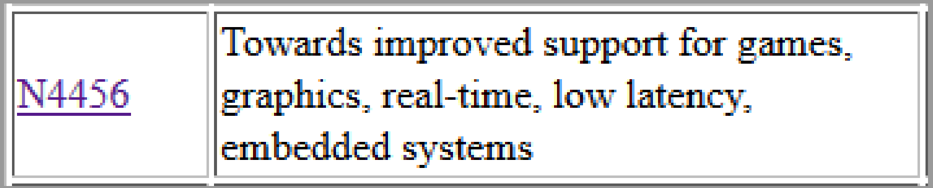
CppCon 2014

C++ committee panel leads to impromptu game developer meeting.

Google Group created.

Discussions have outstanding industry participation.

N4456 authored and published!



N4456	Towards improved support for games, graphics, real-time, low latency, embedded systems

Formation of SG14

SG14
Game Dev &
Low Latency

N4456 presented at Spring 2015 Standards Committee Meeting in Lenexa. Very well received!

Formation of Study Group 14:
Low Latency
Games/Financial/Trading/Simulation
+Embedded Devices
Chair: Michael Wong

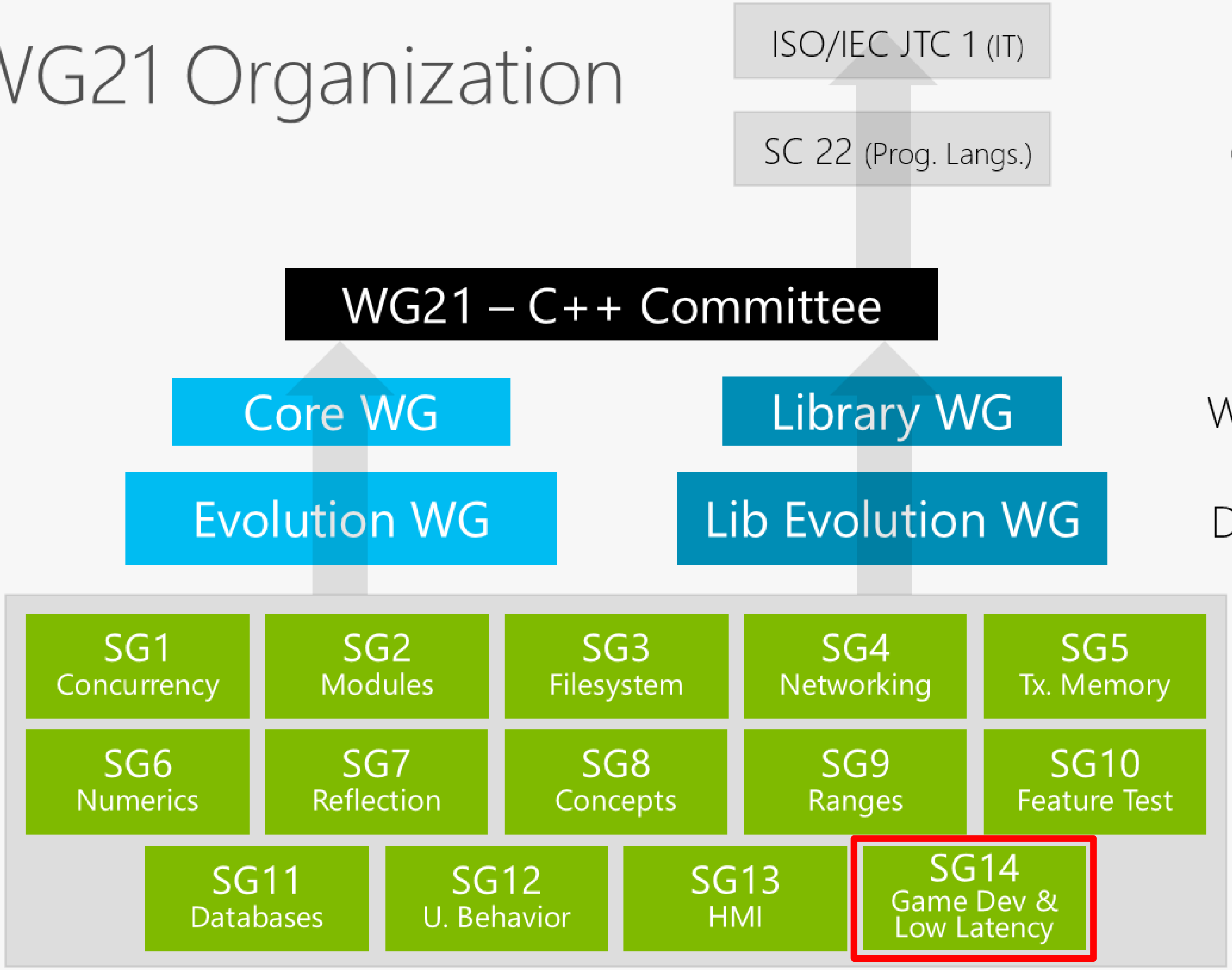
SG14 past meetings:

- CppCon 2015
- GDC 2016
- London STAC SG14 Neil Horlock
- Chicago STAC SG14 Tom Rodgers/Nevin Liber

SG14 future meetings:

- New York STAC: today
- Amsterdam HFT SG14 Optiver: June 27
- CPPCON 2016: Sept 21
- Meeting C++ Games Track: Nov 18/19

WG21 Organization



(F)DIS Approval

CD & PDTS Approval

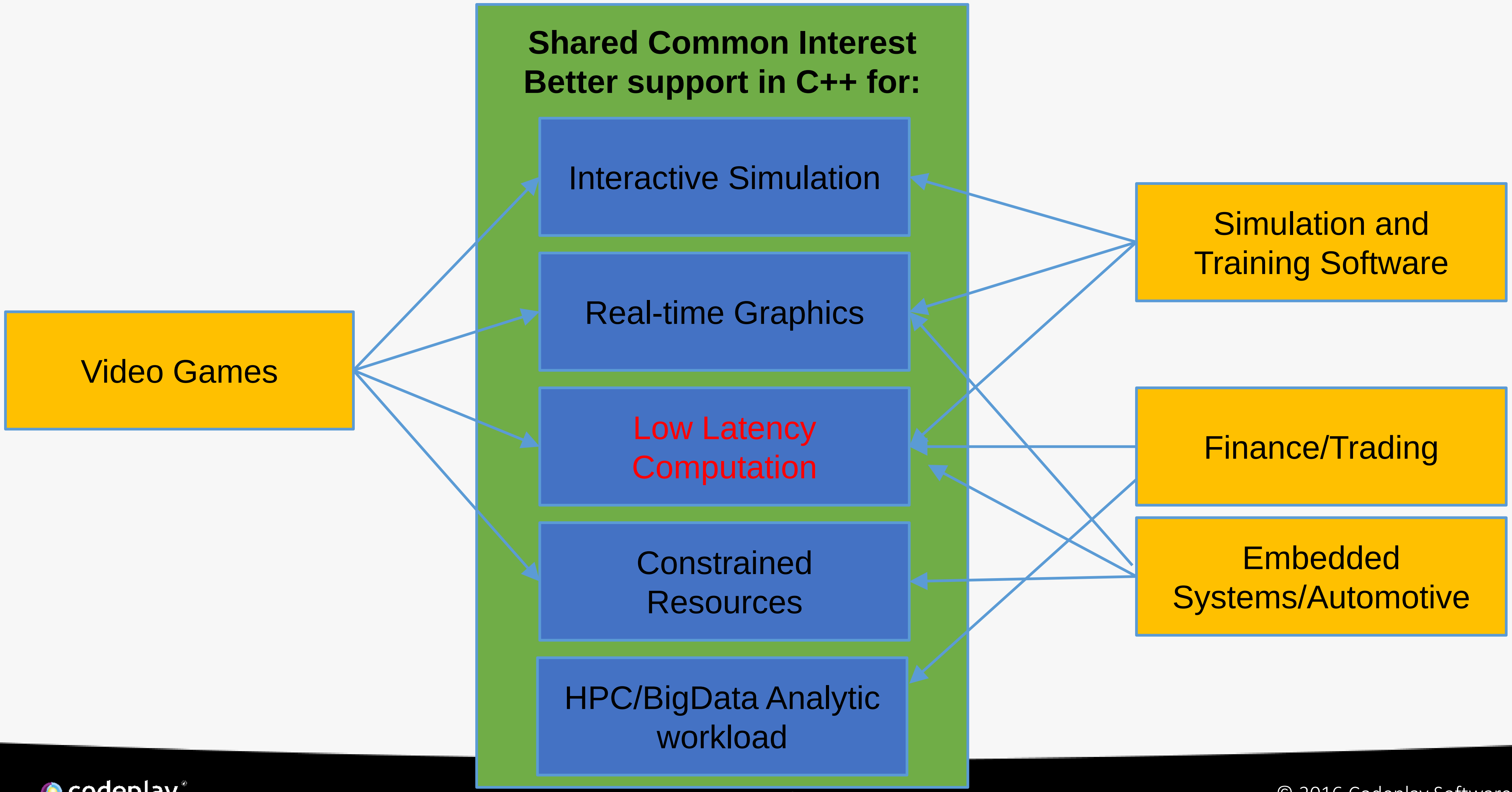
Internal Approval

Wording & Consistency

Design & Target (IS/TS)

Domain Specific Investigation & Development

Audience of SG14 Goals and Scopes



Memory Usage

- Fixed memory budgets
    - 100’s of MB to a couple GB
- Shared CPU/GPU memory
    - 100’s of MB in texture data, animations, framebuffers…
- No swap space and no temporary disk scratch space
- Upgrading hardware isn’t an option
    - Users may not even have the option (eg: game consoles)

Computation Time

- Cost of debug iterators in vendor libraries
  - Many game engines replace even std::vector
  - Each implementation has a different magic incantation to turn off unwanted “features”
- `dynamic_cast` versus home-grown reflection systems
- Not all O(log N) are made the same
  - `boost::flat_map` vs the node-based `std::map`

Inconsistent Allocation Patterns

- Container implementation differences
  - Does an empty container allocate?
  - Vector growth rate and initial capacity?
  - Small string and small object optimizations?
- What size functor will require `std::function` to allocate?
- Behavior is unpredictable when porting between platforms and C++ implementations

Implementation Details

- `std::async` uses a thread pool?
- Standard library features often re-implemented
  - EASTL (Electronic Arts' implementation)
  - STLport (based on SGI's implementation)
  - `folly::FBVector` (Facebook's custom `std::vector`)
  - llvm/ADT (LLVM's custom containers)

Traditionally Costly Features

- RTTI
  - Excessive data generated by eg. dynamic_cast
- Virtual functions
  - Less important these days, but still worth noting
- Poor inlining
  - C++ abstractions not always as free as we are taught to believe
- Exceptions
  - Restrict some optimizations for unwinding

Exceptions & RTTI

- Games often use `-fno-exceptions` and `-fno-rtti`
  - Some important platforms don't support exceptions reliably or at all
- Behavior of `try`/`throw`/`dynamic_cast` not defined when disabled
  - Usually results in a compile error making many libraries unusable without modification
- Not just a games thing or a niche concern
  - http://llvm.org/docs/CodingStandards.html#do-not-use-rtti-or-exceptions
  - https://google-styleguide.googlecode.com/svn/trunk/cppguide.html#Exceptions

Memory budgets

- Content creator and production focus
  - Artists, designers, distribution/publishing/QA
  - Let them answer questions of memory budget on their own (programmers' time is expensive and precious)
- Capture memory stats in the middle of a 3-hour test session without expensive or slow instrumentation
- Need finely-grained accounting and budgeting

Allocation Interfaces

- Standard allocator usage is rare in games
  - Interface is non-ideal
  - Built-in accounting support for distinct memory regions
- Custom allocators with an innate knowledge of alignment
  - Global `new` and `delete` on many platforms not aligned for SIMD
- Simpler interface for custom allocators, of which we have many
  - C++11 was a big improvement on this item, at least, though not perfect
  - Rebinding for node-based allocators is crazy
  - Allocator has no reason to know what it’s allocating
    - Even if the allocator has strict size or alignment limits

Performance

- Some hardware has ~~terrible~~ no branch prediction
- Cache locality increasingly critical
- Small inefficiencies permittable in desktop software unacceptable for us
- Performance matters even when debugging
- Memory usage and performance are tightly coupled
- Need algorithms and data structure designed for real hardware
    - Pure math is great and never changes, but hardware certainly does

Some missing algorithms

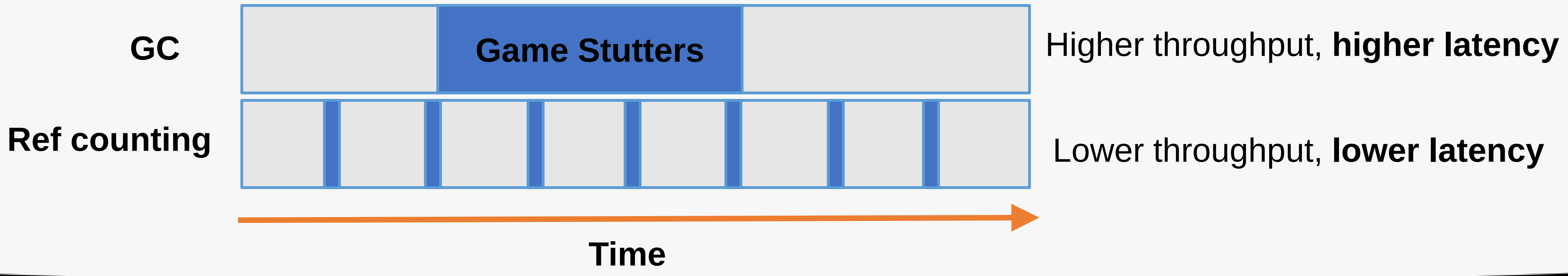
- Radix sort
  - Integer keys are king
  - Very efficient CPU comparison
  - Trumps `std::sort`
- Spatial and geometric algorithms
- Imprecise but faster alternatives for math algorithms

Some missing containers

- Intrusive linked list container
  - Fewer allocations and static initialization
  - No “self iterators”
- Cache-friendly hash table
- Contiguous containers
- Stack containers

Bounded worst case time, why we use C++ instead of C#, JAVA, or D

- Worst case time vs average case time
  - In general, steady 30fps > jittery 60fps
  - Especially important for VR (jitter = nausea)
- Note: garbage collection trade-off


GC
Game Stutters
Higher throughput, **higher latency**
Ref counting
Lower throughput, **lower latency**
Time

Long Compilation Times

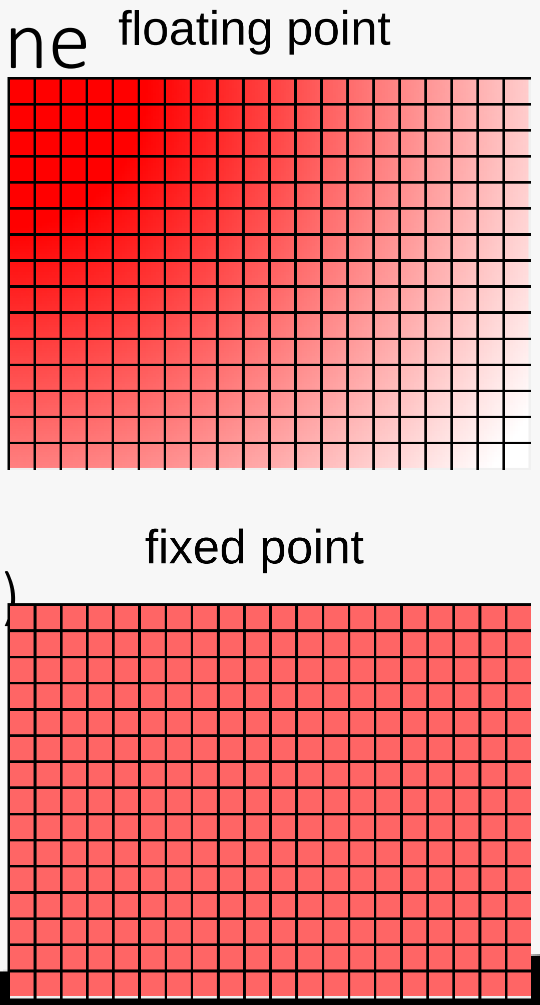
- Template/include bloat
  - `std::unique_ptr`/`std::array` vs C pointer/array
  - `<memory>` over 2 KLOC in VC14 (+ dependencies)
- "C with classes"-style code compiles *much* faster
- File I/O, complex grammar, template instantiation, optimizations
- Modules to the rescue?

https://xkcd.com/303/

Fixed-point Numbers

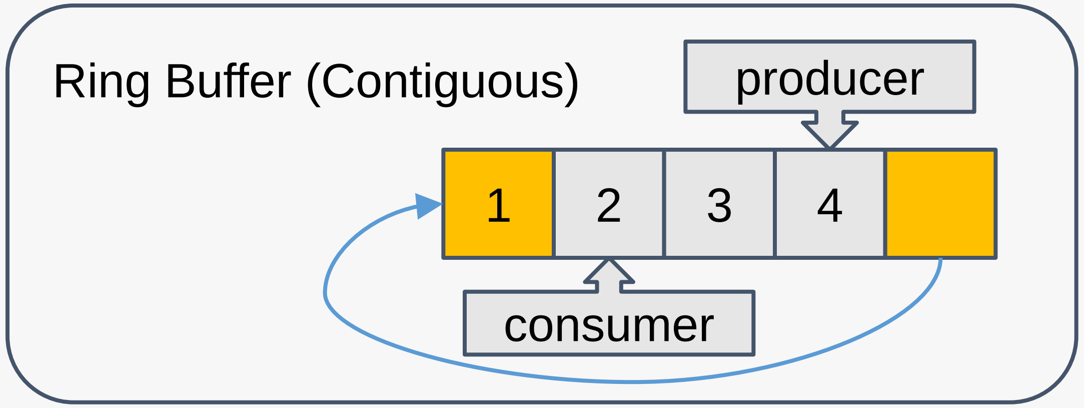
- Effort led by Lawrence Crowl and John McFarlane
  - Overlap with SG6 "Numerics"
  - P0037R0 Fixed point real numbers LEWG SG14/SG6 (McFarlane)Baker
  - N3352 "C++ Binary Fixed-Point Arithmetic" (Crowl)
- Example uses:
  - Platforms slow at floating point (eg: no FPU present)
  - Uniform precision (as opposed to float's varying precision)
- Proposed:
  - std::fixed_point<Repr,Exponent>
  - std::make_fixed<IntegerBits, FractionBits>

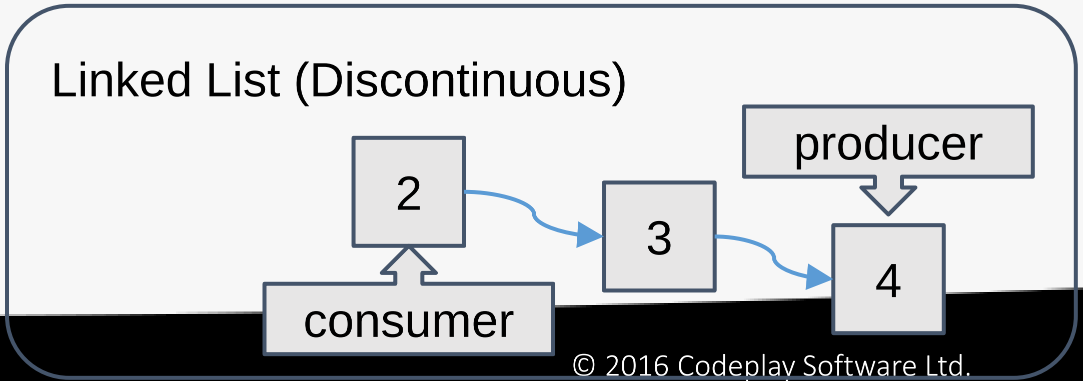
Precision of screen coordinates
(not actual precision, just example visualization)



Ring Buffer

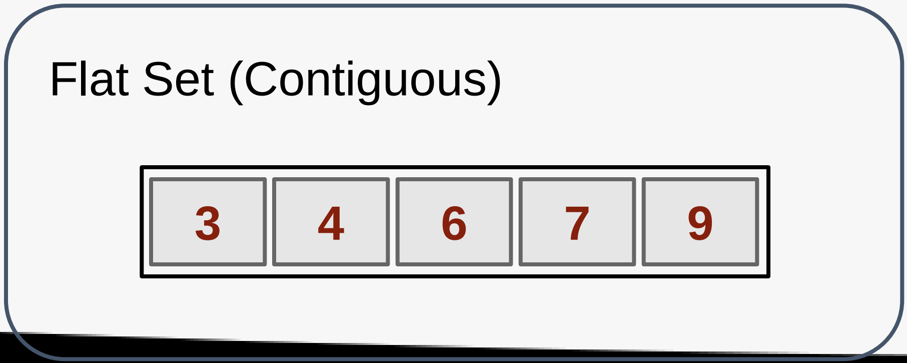
- Effort led by Guy Davidson
  - P0059R0 Add rings to the Standard Library Guy Davidson LEWG SG14: Michael
- Contiguous FIFO buffer
- Examples uses:
  - Feeding audio samples to a DAC
  - Queuing up network packets to be sent
  - Buffering frames of video
- Approved in SG14, LEWG,
  - proceeding to wording

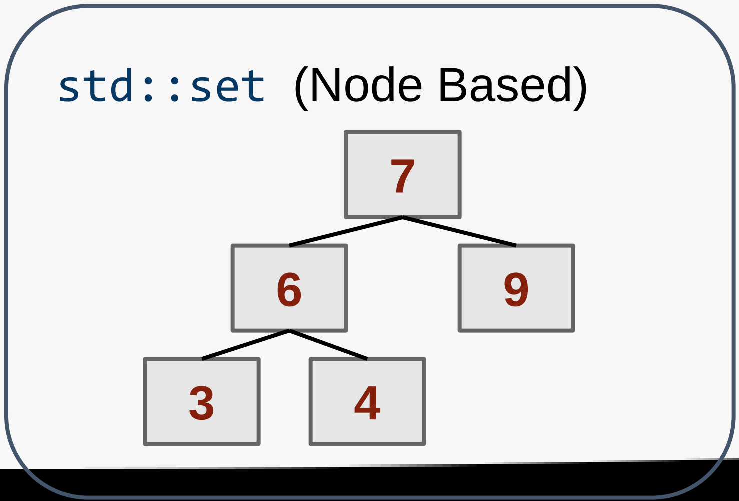




"Flat" Associative Containers

- Effort led by Sean Middleditch
  - P0038R0 Flat Containers Sean Middleditch LEWG SG14: Patrice Roy
- Cache-friendly associative containers
  - Binary search in sorted contiguous memory
  - Similar to `std::lower_bound` but with associative container interface
- Approved in SG14, LEWG





SG14 Financial/Trading major interest thrusts

- SG14 meeting after London STAC May 11
- SG14 meeting after Chicago STAC May 18

- Massively parallel dispatch to Heterogeneous devices
  - Accelerators
  - FPGA
- CPU/cache/memory affinity/HBM
- Composable Memory allocation
- Exception Handling Lite

SG14 HFT/Finance/Trading meeting future proposals

- Intrusive containers
- Interprocess Communication
- Array View
- Node-based Allocators
- String conversions
- hot set/hot set, likely/unlikely, frequency
- vector and matrix
- Executors
  - 3 ways: low-latency, parallel loops, server task dispatch
- Atomic views
- Coroutines
- SIMD/Vector support
- Ring that minimizes contention
- Non-allocating containers
- Small vectors that enable storing non-movable types (atomic.mutex)
- Networking
- openGL/Vulkan
- Read/write Contention attribute
- More precise time/date support
- Lock-free types/queues

codeplay

WORLD OF WARSHIPS
WARGAMING.NET

We're Hiring!

codeplay.com/careers/

codeplay®

THE HETEROGENEOUS SYSTEMS EXPERTS

@codeplaysoft

info@codeplay.co
m

codeplay.com