ISO/C++17 and Beyond: Parallelism and Concurrency (breakout session)

DOE COE Performance Portability

August 22-24, 2017
Denver, CO

SAND2017-8950 PE
Agenda

- DOE Lab participation in ISO/C++ Standard Committee
  - Why we are involved – HPC performance portability!
  - Your lab’s point-of-contact
  - How the ”sausage is made”

- Overview of HPC relevant C++11 & 17 features

- Overview of potential (in-the-works) C++20 features
  - What in-the-works features are most important to you?
  - What high-priority features not in-the-works?
ISO/C++ Committee : Your Lab’s Point-of-Contact

- Each lab is an independent corporate member of committee
  - One primary representative exercises formal voting rights
    - Annual dues, listed in ISO directory, participate regularly
  - Alternate representative voting rights in the absence of primary
    - Also listed in ISO directory
  - Anyone from member org can participate in meetings

- Your lab’s primary representative (some alternates?)
  - SNL – Carter Edwards
  - LANL – Stuart Herring
  - LLNL – Jim Reus
  - ANL – Hal Finkel
  - LBNL – Bryce Lelbach
  - ORNL – Graham Lopez

- FYI: SNL hosting next ISO/C++ Committee Working Meeting
  - Nov 6-11 @ Albuquerque
ISO/C++ Committee: How Sausage is Slowly Made

Proposal Paper

**Domain Expert** Study Groups
Concurrency and Parallelism Study Group (SG1)

Library Evolution Working Group (LEWG)

Language Evolution Working Group (EWG)

Library Working Group (LWG)

Core Working Group (CWG)

**Subject** Technical Specifications (TS)
Concurrency TS

FULL COMMITTEE OFFICIAL VOTING

Draft Standard
ISO/C++ Committee : Roles & Responsibilities

- Proposal Papers
  - *Anyone* may submit a tracked proposals for
  - Additions or modifications to draft standard or technical specifications

- Domain Expert Study Groups
  - Apply domain specific (beyond language/library) expertise; e.g.,
  - Concurrency & Parallelism (SG1), File System (SG3), Networking (SG4), ...

- Library (LEWG) and Language (EWG) Evolution
  - Prioritization among always-too-many proposal papers...
  - Broad enough impact to be worth supporting?
  - Well-specified scope, semantics, interactions, and domain-expert review?

- Library (LWG) and Language (CWG)
  - Well-specified *standardese* (wording for standard)?
  - “Quality Assurance” – my personal view
  - *Typically* the only groups to bring “straw poll” motions to the full committee
    - Bad form and drama ensues attempting to bypass quality assurance
HPC-Relevant C++11 & 17 Features

- Parallel algorithms (C++17)
  - Allow functors to be executed in parallel on unspecified resources
  - ... important incremental progress, C++20 improvements in the works

- Atomic Operations and Memory Model (C++11, improved C++17)
  - Efficient inter-thread communication / synchronization
  - Scalable concurrently modified data structures

- Threads, Mutexes, and Conditions Variables
  - C++11 pulled pthreads into the standard w/ name changes & reduced scope

- Futures, Promises, and Async 😞
  - ... infamous, whispered about C++11 “Kona compromise”
  - ... avoid using these for now; at least be very cautious
Productivity-Relevant C++11 & 17 Features

- **Lambda Expressions – inline functors**
  - C++11: Introduced, dramatic improvement to productivity
  - C++17: Language flaw fixed for [*this]
  - NVIDIA CUDA 8: Offload lambda expressions to GPU
  - Essential for ease-of-use in Kokkos, RAJA, ...

- **Template meta-programming improvements**
  - C++11: Variadic template arguments
  - C++11: `<type_traits>`
  - C++17: constexpr conditional statements 😊😊😊😊
HPC-Relevant Potential C++20 Features

- Atomic Operations Enhancements
  - Floating point fetch_add and atomic operations on non-atomic types
  - Building blocks enabling scalable parallel scatter-add algorithms

- Latches and Barriers
  - Atomic-like thread synchronization mechanisms

- Executors and Execution Context
  - Executor – specify how concurrent/parallel work is dispatched
  - Context – specify where concurrent/parallel work is dispatched
  - Fix futures and async

- Wavefront extensions to parallel algorithms
  - "Staggered" or "pipelined" parallel execution of loops
HPC-Relevant Potential C++20 Features

- **Coroutines (TS)**
  - Functions designed to be called iteratively/concurrently
  - Well-defined suspension/resumption

- **SIMD Types – Portable Vector Intrinsics**
  - Guarantee arithmetic operations map to intrinsics for vector hardware
  - Address vector width, intra-lane operations, conditional control flow, ...

- **Multidimensional Arrays (finally!) with Polymorphic Layout**
  - Motivated by Kokkos multidimensional arrays
  - Array type includes row major, column major, ... layout specification

- **Modules (TS) – improve compilation performance**

- **Concepts “light” – improve template meta-programming**
  - Many, many years in the making, with some reputations on the line...
  - ... question has been raised if, given C++17 features, this is still useful 😐
Open Dialogue / Deep Dive as Requested

- C++11 spec, C++17 draft, proposal papers
  - cppreference.com for C++11, C++14, C++17, and TS toward C++20
  - also have some in-hand

- Priorities for potential features mentioned?
  - May need to be championed by you/your DOE Lab reps

- Important needed features not mentioned?
  - May have failed to mention
  - May need to be added and championed by you/your DOE Lab reps

- Deeper look at existing / proposed features?
  - Sufficient number present want to deep-dive? do it now
  - Otherwise an off-line small group activity