Distributed & Heterogeneous Programming in C++ for HPC at SC17

Michael Wong (Codeplay), **Hal Finkel**

DHPCC++ 2018

The Panel

BoF @ SC17

- Ben Sanders (AMD, HCC, HiP, HSA)
- Carter Edwards (SNL, Kokkos, ISO C++)
- CJ Newburn (Nvidia, HiHat, Agency, CUDA)
- David Beckingsale (LLNL, RAJA)
- David Hollman (SNL, Dharma, ISO C++)
- Hartmut Kaiser (LSU, HPX, ISO C++)
- Michael Wong (Codeplay, ISOCPP Foundation, Khronos, ISO C/C++, SYCL)
- Ronan Keryell (Xilinx, Khronos, SYCL, ISO C++)
- Xinmin Tian (Intel/Altera, OpenMP C++)

The Structure

BoF @ SC17

- The BoF was one hour in length
- A 20-minute introduction by all of the panelists that described their work
- 30 minutes of discussion
- Five minutes to collate results
- A five-minute summary
- A pre-populated Google sheet with questions
- The audience was able to update the sheet with new questions which others could up-vote to push them to top of queue.
- The experts had ranked the pre-populated questions

The Questions

Topic Question:

What are the biggest challenges for heterogeneous and distributed computing in C++ (DHPC++) for HPC? And how do we approach the solution?

BoF @ SC17

Question	Votes	
2	15	What is the preferred way to expose massive parallelism: Directive based (OpenMP/ACC) (in C++, likely through attributes), thread/task based(TBB, Fibers, C++11 threads), explicit parallel API C++ AMP/HCC, CUDA, SYCL, Kokkos, C++17 ParallelSTL), stream (for DSP and FPGA over continuous stream of data), dispatch (Networking TS using runtime executor of function object on remote device via network)?
3	11	Is Performance Portability across heterogeneous devices really possible? Under what constraints?

BoF @ SC17

Question	Votes	
14	10	Do you see this ever replacing MPI or OpenMP/ACC?
1	9	What is the preferred Data movement if any: user directed with appropriate API (explicit e.g. most models) or runtime directed (implicit e.g. SYCL, C++AMP)?
5	8	Should C++ change its memory model from flat/single cache coherent address space (OpenMP/ACC, HSA, OPenCL 2.x, C/C++) to Multiple hierarchical memory model (SYCL, C++AMP, OpenCL 1.x) or some other form like non-coherent single address space (CUDA)?

BoF @ SC17

Question	Votes	
4	7	Is there a preferred way of compiling source code for a sub-architecture/node (Separate source: OpenCL C,/C++, GLSL), Single Source (CUDA, OpenMP/ACC, SYCL, C++ AMP), Embedded DSL expressions constructed from C++ operator overloading (RapidMind, Halide). Should we support different separate host and device compilers instead of monolithic tool chain?
20	7	Should we design heterogeneous C++ programming models to accommodate hardware that lacks support for existing C++ features (i.e. virtual functions, unified memory, C+11 atomics, hardware floating point) or simply restrict support to more capable devices?

BoF @ SC17

- And there were many more questions (~35 in total). Some overlapping with others.
- We only got to the top three questions...

BoF @ SC17

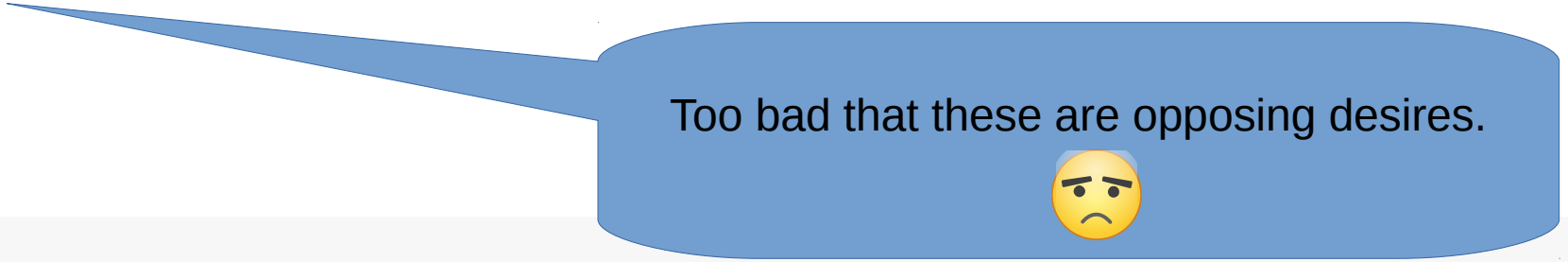
- Question 2: What is the preferred way to expose massive parallelism?
- Carter Edwards
  - We need all of the constructs, because patterns, policies, and data placement need to be first class citizens in the language, and in this way, they can be migrated into the compiler's intermediate representation, so they can be optimized.
  - His opinion is that OpenMP will not work well in the future, because it lacks this comprehensive scope.
- Michael Wong
  - Should be a strong and clear separation of concerns between interface and implementation, and conflating these will make it more difficult to program heterogeneous devices as we reach massive parallelism.
  - There is a big glob of existing legacy software and requirements for inter-language support, so while we may be able to use C++ attributes as a syntactic sugar for pragmas from OpenMP, the result will still be ugly.
- Hartmut Kaiser
  - Experience with HPX tells him that you can implement any kind of parallelism on top of a light-weight tasking system. This includes distributed, heterogeneous, asynchronous, and data parallelism.
  - A strong proponent of massively-scalable, lightweight tasking systems.

BoF @ SC17

- Question 2: What is the preferred way to expose massive parallelism? (cont.)
- David Hollman
  - We do care about that the programming model is lossless, meaning it does not omit info that in programmers' mind could be beneficial and that is not expressed in the programming model.
  - He felt that threads and mutexes, for example, are lossy programming models, and express mutual exclusion on data, or parallelism on data, without providing the explicit relationship between the parallelism construct and the data.
- CJ Newburn
  - We need to differentiate between what you could do with the language and what you could do in the language.
  - The proper level of abstraction means that we don't want to over-decompose if the underlying hardware does not support that much parallelism. As for how to enable decomposition, he believes that doing it in a hierarchical way to match the variety of hardware over the lifetime of the code is appropriate.
  - He wishes to ensure the design is application driven to make sure we address the real and immediate needs of the community.

BoF @ SC17

- Question 2: What is the preferred way to expose massive parallelism? (cont.)
- Ben Sanders
  - Parallelism should be expressed as part of the language similar to the way it will be in ISO C++, but noted that we cannot convince the whole world to use one programming language.
  - There will always be OpenMP, as it is still simple, still incremental, and we can't expect people to use iterators just to write a for loop.
- From the audience...
  - A member of the audience pointed out that we have already seen the failure of auto-vectorization due to its being not really repeatable, and added that massive parallelism will end with up with a similar problem.
  - David Beckinsale feels that we should try to focus on standardizing abstractions and not ways of executing things. He notes that if an abstraction is being implemented differently in different compilers, that is a problem. The abstractions we put in the standard are hopefully at the right level.



BoF @ SC17

- Question 3: Is performance portability across heterogeneous devices possible? (cont.)
- Carter Edwards - Kokkos has some proof points to support this assertion
- CJ Newburn - Feels it is still challenging, and the key is higher abstraction
- Michael Wong
    - There is an iron triangle of language design: performance, portability, and productivity. If you design a language and optimize for two of those, you will fall short on the third one.
    - He points out that the most productive parallel programming language is SQL (which is not a particularly high-performance language, but millions have learned to use it to do productive things, and it is quite portable) and that could be a far more important measure of a language's success then performance portability.
- Ronan Keryell feels that no matter what we think we can achieve, FPGA programming is still challenging. 

BoF @ SC17

- Question 14: Do you see this replacing OpenMP, MPI, or OpenACC?
- Hartmut Kaiser
  - Strongly feels that it will.
  - He feels OpenMP pragmas become more difficult to use especially as application characteristics can change dynamically, and these things need the runtime library to react.
  - He felt that you need to make the compiler aware of the runtime system, and code needs to convey information to runtime, particularly about how long it takes to execute each kernel. He feels that this kind of separation of concerns is key, and the more unpredictable code is, the less OpenMP or MPI will represent well the application's parallelism. In particular, he mentions the need for asynchrony.
  - His opinion is that OpenMP will not work well in the future, because it lacks this comprehensive scope.

BoF @ SC17

- Question 14: Do you see this replacing OpenMP, MPI, or OpenACC? (cont.)
- From the audience…
  - Graham Lopez from the audience questioned if runtime adaptivity needs C++ to use JIT in the future.
    - Most on the panel felt that the answer was no, although an adaptive runtime was likely necessary.
    - Carter Edwards adds that there is an algorithmic trade-off where you trade reproducible with floating point, but you lose performance and increase space, so those options need to be available.
    - Hal Finkel pointed out the difference between performance reproducibility and output reproducibility. However, when you use different algorithms based on the timing you want, this could give you different answers depending on how you set up the application, and thus the programmer can choose to tie the two together.

Final Questions for the Audience

BoF @ SC17

- How many of you are using task-based parallelism in your applications? Many.
- How many who are not using task-based parallelism now but will use it in next 2-3 years? About 15.
- How many do not expect to use task-based parallelism explicitly but believe that their underlying systems will use task-based parallelism? A few.
  - Ben Sanders pointed out anyone using a deep-learning framework should raise their hand.
- How many believe you have a performance-portable application right now? No one, maybe 1.
- How many believe you will be able to create performance-portable applications in the next few years? A few.
- How many believe you can do this but not in C++? A few.
- How many of you will not be able to create performance-portable applications? About half the room.
- How many are using MPI? Many.
- How many think you can replace MPI in next 2 years with some other C++-based library solution? A few.
- How many are using OpenMP/OpenACC in application now? A bunch.
- How many believe you can replace OpenMP/OpenACC with C++ abstractions in a few years? Many.

Acknowledgments

- The SC17 organizers and community
- ALCF, ANL, and DOE
- ALCF is supported by DOE/SC under contract DE-AC02-06CH11357