Format

Name: comment/question

Discussion

Zarija: Wonder about recommendations for maintaining CPU systems. Looking toward past rather than the future? Better to ask for more funding to move to GPU architectures?

Kevin: Distinction between HTC and HPC problems (discussed in report); not all HEP problems can be made into HPC problems. If DOE leadership computing facilities want to provide an increasing portion of HEP computing, need to be appropriate for the kinds of computing we need to do. Otherwise, we can't use those machines at all, or efficiently.

Peter: Not all algorithms have the level of symmetric parallelism required by GPU instruction set models. Thread divergence can be compulsory without array type parallelism, and makes certain problems impossible in principle to parallelise efficiently.
There is also a long tail of CPU-based code, for which it may be possible but not most cost-effective to invest in GPU porting - provision of CPUs at a rightsize level seems to optimize the overall cost including software development. It may be simply cheaper to continue to use it on a small fraction of computing resource rather than rewriting it, which is costly. In future, can think about CPU cores w/ very high bandwidth memory. Discussion in report on these topics. Minimize pain for the programmer to use hierarchical memory.

Tobias/Andreas: We would like to stress the importance of CPU-only resources in perturbative precision calculations. Whether new algorithms can be developed to efficiently use GPUs is currently unclear as a matter of principle. While more funding for people that develop the tools/results for high precision predictions is desired, it can't replace an expansion of CPU resources in the future.

Liz: Can use of hierarchical memory be automated?

Peter: Yes - Nvidia has Unified virtual memory, AMD has this in HIP, Intel SVM in SYCL. CPU-GPU transfers triggered by page miss: not quite as fast as hand-written, but much easier to develop and maintain and would amortise cost if page size was bigger. Hierarchy will only grow with time. May even be a third layer of non-volatile memory. Vendors need to step up and DOE can encourage this. Larger of flexible software page sizes may optimise bandwidth.

Axel: I think we can generalize this by using/defining common algorithmic modules (MapReduce, Transform/ForAll, etc.), for example in the parallel C++ STL, but cannot hope for a silver bullet for implementing general purpose algorithms: approaches like PGAS w/o locality awareness do not scale. Thus, we should focus on APIs, making them modular, compatible and reusable and R&D modern abstractions for our modeling/computing needs.

Alex: From generator talk, using common data formats etc. But what if generator is taking input from something else? Common in neutrino usage.

Josh: Similar on LHC side, using LHAPDF, LHE files, etc. Discussing how this could work with fluxes. Neutrino community should decide on a format.

Alex: Neutrino community has had limited success at getting generator developers to agree. How did LHC do this?

Josh: Have some group that goes out and does it, then experiments adopt it and that forces others to use it. Les Houches meetings had someone show code to do this and then it grew.

Liz: There was a longstanding project called [] in the UK that funded a lot of this work, because it was critical to LHC physics. National support for generic programming work is severely lacking here/now. MCNet was just defunded. Still a lot of work to go.

Daniel: Question to generator community: what are your aspirations? Most development happens in EU universities or national labs. They have training, networking, opportunities for young people. Snowmass is the right place to discuss our (US) aspirations.

Josh: Talked about these topics, but were told by some people that this shouldn't be included in the white paper / report. Only two people funded in US and they don't have students because they work at Fermilab. Event generator community viewed as not really theory and not really experiment, so both sides disclaim responsibility.

Jeremy: Many talks reference large codebases that are developed and maintained over long periods of time, often decades. Is a more formal process for support of these collaborative software stacks needed?

Andreas: Yes, this is needed in generators for example. Similar story for lattice with Grid code, MILC code, etc.

Liz: Intrigued by new method to mitigate negative weight problem. It was a big issue in CMS during Run 2, many people went back to LO MC. Can this approach be applied to all modern generators, and is just a matter of personpower to implement? Or does it work better for some than others?

Josh: Some generators have more personpower than others. Some people get SciDAC grants to do this kind of thing, but are often "outside" generator author groups/collaborations. Then the R&D needs to go through the collaboration to be officially implemented. Stefan could say what Sherpa is doing.

Peter: Do you support the comments about engagement between HEP and ASCR, ECP, SciDac?

Josh: Has had good experience with SciDAC, but has heard of bad experiences. Problems that ASCR are interested in are not necessarily the same problems that HEP/generators want to solve. Problems may be transformed into something different and more complicated.

Peter: Perhaps a challenge of structuring the collaboration.

Kevin: Collaborations can be very productive, but need to ensure that they serve the actual needs in HEP.

Josh: Need to find people in ASCR who are interested in solving HEP problems.

Axel: Management criteria & style can be different, as can applications. ASCR in ECP puts a lot of emphasis on integration and solving complete problems, evaluating both domain and computer science by these criteria. Avoids having too many academic papers that never make it into code

Liz: Completely agree.

Jeremy: SciDAC and ECP are very different programs. Level of support and engagement from ASCR participants are different. ECP, wants to make sure to get as much science out of new machines as possible. SciDAC is about R&D. Saying what kind of collaborations with computer scientists could be included in recommendations.

Andreas: ECP is ending by FY23, might be extended a bit. Might be a child of the existing program in the future? (No?) Job of ECP is to develop code to make expensive machines run, that job will not be done when ECP ends. Different from developing new algorithms which is often ASCR's interest. SciDAC funding model has deliberations about what gets funded as collaboration between ASCR and HEP, can try to extend this.

Alex: A connection with Geant4 development: getting simulations to run on modern infrastructure doesn't necessarily draw in people who have been focusing on the physics side. A broader recommendation (beyond just generators) might be useful.

Jeremy: Someone mentioned that partnerships with industry would be helpful. What sort of partnerships?

Peter: Early engagement with future hardware is important. In community frontier meeting this morning, whenever DOE engages with industry, question of IP comes up very early. Having domain scientists get early access is invaluable. Need access to new exaflop-scale machines before they're built and open for science in order to develop the software, because it takes years. Can be simulated hardware or prototypes.

Axel: Has been very important to have access to vendor compilers & libraries as they evolve, even if quite remote from final hardware, 2 years in advance to catch bugs and quality issues in software. Vendor software has release cycles of 6months+ and we need to iterate multiple times for complex software stacks to even compile on brand-new toolchains. (Otherwise domain-scientists will spend much time on maintaining work-arounds.)

Liz: Don't know if this was talked about in papers, but another place DOE could help w/ pushing industry is in terms of standards in programming interfaces. We can try to get these pushed into C++ language. Not really in industry's financial interest, but it's our science interest. DOE can make that push.

Peter: parallel STL or open MP discussed.

Kevin: Nvidia actually pushing on getting parts of CUDA into C++ standard because they're spending a lot of time and money on support.

Peter: Problem of having multiple standards

Axel: Previous points: we should make these things part of our acceptance tests for machines (and already do in parts, which is a good motivator for vendors).
Multiple standards: not yet a problem, better to have multiple well-documented standards than everyone rolling their own (with consequently less documentation & testing).

Peter: Keep in mind that right now handwritten CUDA can be a lot faster than portable interfaces.

Ji: Need a bigger community agreement to talk about standards. Took a lot of time to get larger community engagement for MPI.

Krzysztof: Should work with industry, but we're all oversubscribed. Need to add more people, not just use the existing people. And need to retain people once they develop that knowledge.

Josh: Kokkos has been a successful DOE program with people on C++ standards committee. Should continue with this for developing parallel algorithms.

Fabio: I wonder whether clarifying who is the target of each recommendation would be useful.

Walter: why is additional factor of 10 in computing resources needed?

Kevin: for Lattice specifically, can try to clarify in report.

Jeremy: Many large collaborative software libraries are broadly used across frontiers and ongoing and uninterrupted support was cited as an issue. Is a mechanism needed to formally

support software with many stakeholders? Identify deliverables, review process, level of support, etc.