Computational chemistry in 202x: challenges and opportunities

As we progress towards exa-scale computation, the number of research groups and chemistry/materials applications fully benefiting from HPC technology is shrinking, whether targeting laptop or supercomputer. Individual researchers and entire communities that have pressing needs for high-performance computation are struggling to field their applications, even at the tera- and peta-scales. For instance, how many of our applications can effectively use even just one modern computer with tens to hundreds of threads, 4-wide SIMD units, and perhaps with multiple GPGPUs.

Two disruptive changes [1] are derailing our progress towards realizing the full potential of HPC in urgent imperatives such as designing sustainable energy technologies and growing our economy. First, our ambitions for scientific computing are leading us to ask increasingly large and complex questions that causes the corresponding complexity of our software to exceed the capabilities of current programming systems. Advances in these fields are being inhibited by the lack of functionality on the largest computers and the time lag between innovation in small research groups and its realization in widely available, state-of-the-art codes. For instance, the huge equations of many-body physics and chemistry have transcended human ability to translate directly into software, and much modern chemistry and materials science are at the interface of disciplines forcing the composition of multiphysics applications with diverse numerical representations, solvers, data structures, and software frameworks. Second, while the high performance made possible by massive distributed-memory systems, multi-core processors with specialized vector instruction sets, and GPU architectures is a huge boon to the HPC community, achieving portable performance across different systems is virtually impossible today, and tomorrow brings new extreme-scale complexities such as resilience and power management. By 2020 it is entirely likely that we will have “clusters on a chip” in which groups of cores will not have cache-coherence between groups (e.g., as is the case now in GPGPUs).

Science software used to live for decades – but no more! Previously successful strategies for maintaining productivity and performance, such as frameworks and expert-written libraries, have been largely destroyed by the disruptive pace of change in architecture and programming models, which will continue and even accelerate for the foreseeable future. For instance, millions upon millions of lines of code within NWChem, GAMESS, QChem, Gaussian, Turbomol, Molpro, etc. must be rewritten by humans to take advantage of GPGPUs, Intel MIC, etc.. Some domains (e.g., fusion, climate, astrophysics) by necessity (e.g., funding or the singular nature of the science problem) have already reorganized into community approaches and codes that credibly use resources more effectively, and enable scientists to focus more on innovating in theory or applications rather than demanding broader computational expertise. There are potential downsides to this – e.g., recently in fusion there was a disagreement between results from two codes that used different numerical methods but since there was only one version of each code no-one was even sure if either code was correct. Some diversity is clearly necessary for what I call intellectual genetic health.

There are many more questions than answers

- What will we do when the next architectural revolution appears? Rewriting code for current computers is already overwhelmingly expensive and we have no general solution at hand.
- What should we be teaching students about “programming” if we don’t even know how to program current computers? Clearly very few students are able to shoulder the burden of becoming expert in both chemistry and modern HPC, and few research groups have the breadth of expertise to provide such preparation.

- What skills reflect the needs of HPC in 2020 rather than 2000 (or before)?
- Can we really aspire to greater productivity and portable performance? How to do this and what do we have to give up in return?
- What are our current success stories?
- What are our failures? Or where could we have been more successful?
- What resources (e.g., federal funding) could we be using more efficiently?
- What can computer scientists tell us about successful strategies to HPC code development and how to adapt them to our purposes?
- What are other disciplines doing that we are not?

The goal of this workshop, which focused primarily on the HPC aspects of our software, is not to concretely answer these questions (though if we have even partial answers, great!). Instead, our goal is to ask even more questions since within this S2I2 (scientific software innovation institute) conceptualization project we are seeking to

- understand our collective software requirements and challenges,
- understand and articulate how we can organize an interdisciplinary community to confront these challenges,
- start organizing and marshaling this community, and
- develop a compelling and competitive vision that will lead NSF to fund our community. There is every reason to believe these will be new monies and hence success will translate into a huge collective expansion of capability.

[1] *The Future of Computing Performance: Game Over or Next Level?*, Samuel H. Fuller and Lynette I. Millett, Editors; Committee on Sustaining Growth in Computing Performance; National Research Council, isbn=9780309159517, http://www.nap.edu/openbook.php?record_id=12980, 2011, The National Academies Press. This document is on the workshop website.