Tighter development cycles between scientific applications and advanced computational infrastructure

Peter Kasson, University of Virginia
Disclosure: the author served as Visiting Faculty at Google from 2011-2013. The views expressed here are the author's own and neither those of the University of Virginia nor of Google.

Our primary experience is in large-scale biomolecular simulation, both performing simulations and using them in conjunction with experimental data to construct statistical models of physical processes. The projects we have run in recent years range from <10 large, long-running jobs each parallelized over 1000s of compute cores to ones that have had >100,000 multi-core tasks executing simultaneously. In addition to the architectural challenges of each of these systems, we believe a central challenge for advanced biomolecular computing is that many important problems occupy an intermediate space: from 100s to 10,000s of concurrent tasks, while benefitting from 100s of compute elements using current CPU cores. These problems can often involve task dependencies, so in addition to hardware architecture, critical needs include workflow or dataflow software that can manage both computational "book-keeping" as well as optimize parallelism and provide a framework for integration of simulations and large-scale data analysis, often involving statistical learning.

While much work has been done on many-task, moderate-parallelism infrastructure, these problems are not entirely well addressed by existing software or hardware solutions. These types of problems often require faster interconnects than traditional capacity systems yet do not benefit from the model of most current capability systems where the ideal task would scale over a large fraction of the machine and run for a relatively short period of time. Other architectures such as local network trees, where nodes have fast, low-latency local interconnects and good connections to storage but do not require the same interconnect to distant nodes, could provide substantial benefit. Such more complex architectures will require more advanced scheduling for efficient utilization, particularly at scale.

A number of recent systems developed outside of academia likely come closest to providing efficient many-task parallelism at scale. Large-scale task scheduling as well as workflow software by companies such as Google, Twitter, and Microsoft have made substantial process in practical large-scale solutions to these as well as data parallelization methods. Successful commercial implementations of this sort are driven by a relentless focus on efficiency for the workloads in typical use there. Advanced scientific computing, however, has several important differences that distinguish it from commercial large-scale computing architectures and tools. First, scientific workloads often differ substantially in their parallelism and resource utilization characteristics from the workloads in use at these companies, necessitating different design characteristics. Second and of equal importance, part of the goal of advanced scientific computing is to develop technologies on a longer timescale than are dictated by pure commercial efficiency for contemporary production workloads. Thus the optimum parallelism, scheduling, etc. for advanced scientific computing may deliberately involve lower utilization efficiency than commercial systems with the idea that the "payoff" will come

in the future. Nevertheless, there are important lessons to learn from commercial "cloud" and other systems, both in terms of infrastructure design and in terms of schedulers, file systems, and programming paradigms for working with many tasks operating at scale. The "more is different" adage is very relevant here, as scheduling and workflow systems have very different failure points and performance bottlenecks when dealing with 100,000 simultaneously running tasks than they do with 500.

Tighter coupling and faster development cycles between scientific needs and advanced computing solutions will provide a critical element both to efficiently serve those needs and to catalyze the next level of scientific and cyberinfrastructure advances. Current allocations processes are incentivized to find the scientific applications that best fit the chosen compute architecture (e.g. tasks that will scale to a certain number of compute elements, etc.). While it is important to select problems that truly involve advanced computing and that will use efficient computational methods, letting the scientific needs drive the computational resource engineering in more explicit and shorter cycles (ideally less than the lifetime of a given hardware resource) would help scientific problems and advanced computing capabilities co-evolve faster. As much as possible, we should let the needs of advanced computationally-intensive science drive the design of advanced computational infrastructure (both hardware and software) rather than allowing infrastructure design decisions to drive the selection of scientific applications. Computational efficiency is of course critical, but our collective focus should be on identifying key computationally intensive scientific problems and engineering the best solutions to solve them now and in the future.