A Not-so-Simple Matter of Software

editorial by Jack Dongarra

When we try to assess how much progress we have made in computational modeling and simulation, recalling some history about the related approaches of experiment and theory can help keep things in perspective. For example, we can trace the systematic use of experiment back to Galileo in the early seventeenth century. Yet for all the incredible successes it enjoyed over its first three centuries, the experimental method arguably did not fully mature until the elements of good design and practice were finally analyzed and described in detail by R. A. Fisher and others in the first half of the twentieth century. In that light, it seems clear that while computational science has had many remarkable youthful successes, it is still at a very early stage in its growth.

Many of us today who want to hasten that growth believe that the most progressive steps in that direction require much more community focus on the vital core of computational science: *software and the mathematical models and algorithms it encodes.* Of course the widespread obsession with hardware is understandable. No one who helps administer the TOP500 Supercomputer Sites project, as I do, can claim to be immune to it. But when it comes to advancing the cause of computational modeling and simulation as a new part of the scientific method, there is no doubt that its complex software ecosystem must take center stage.

At the application level the science has to be captured in mathematical models, which in turn are expressed algorithmically and ultimately encoded as software. Accordingly, on typical projects the majority of the funding goes to support this translation, which over its course requires intimate collaboration among domain scientists, computer scientists, and applied mathematicians. This process also relies on a large infrastructure of mathematical libraries, protocols, and system software that has taken years to build up and that must be maintained, ported, and enhanced for many years to come if the value of the application codes that depend on it are to be preserved and extended. The software that encapsulates all this

time, energy, and thought routinely outlasts (usually by years, sometimes by decades) the hardware it was originally designed to run on, as well as the individuals who designed and developed it.

Thus the life of computational science revolves around a multifaceted software ecosystem. But today there is (and should be) a real concern that this ecosystem, including all of its complexities, is not ready for the major challenges that will soon confront the field. Domain scientists now want to create much larger, multi-dimensional applications in which a variety of previously independent models are coupled together, or even fully integrated. They hope to be able to run these applications on petascale systems with tens of thousands of processors, to extract all performance that these platforms can deliver, to recover automatically from the processor failures that regularly occur at this scale, and to do all this without sacrificing good programmability. This vision of computational science contains numerous unsolved and exciting problems for the software research community. Unfortunately, it also highlights aspects of the current software environment that are either immature, underfunded, or both, as Douglass Post and Lawrence Votta recently pointed out in *Physics Today.*

Advancing to the next stage of growth for computational simulation and modeling will require us to solve basic research problems in computer science and applied mathematics even as we create and promulgate a new paradigm for the development of scientific software. To make progress on both fronts simultaneously will require a level of sustained, interdisciplinary collaboration among the core research communities that, in the past, has only been achieved by forming and supporting research centers dedicated to such a common purpose. A stronger effort is needed by both government and the research community to embrace such a broader vision.

I believe that the time has come for the leaders of the computational science movement to focus their energies on creating such software research centers to carry out this indispensable part of the mission. The NCSA community has always been in the vanguard of efforts to catalyze and organize precisely these kinds of interdisciplinary research partnerships that we now require to transform the future of scientific software. I have every confidence that this community stands ready to step up again to this momentous new effort.

Jack Dongarra is university distinguished professor of computer science in the Computer Science Department at the University of Tennessee. He also holds the title of distinguished research staff in the Computer Science and Mathematics Division at Oak Ridge National Laboratory and is an adjunct professor in the

Computer Science Department at Rice University. He specializes in numerical algorithms in linear algebra, parallel computing, use of advanced computer architectures, programming methodology, and tools for parallel computers. He is executive editor of the Cyberinfrastructure Technology Watch, a publication of the NSF-funded CyberInfrastructure Parntership.

Access Online | posted 10-25-05