Int J Parallel Prog (2009) 37:1–2
DOI 10.1007/s10766-008-0090-5

EDITORIAL

Guest Editor Introduction: Special Issue on High Performance Computing for High Productivity Environments

Nicholas Carriero

Published online: 22 January 2009

Rapid prototyping and application development environments are often the tool of choice for many developers and users. These environments and languages include MATLAB®, perl, python, R and Ruby. They are popular, in large part, because they make computing accessible to "accidental programmers" (scientists, statisticians, economists and others not specifically trained as programmers) and because they deliver on their promise of rapid prototyping to more seasoned programmers. Users of these systems often face a dilemma as their problems grow: live within the performance and resource constraints of these systems, or leave these comfortable environments behind and take up tools that are traditionally associated with high(er) performance computing (C/C++/FORTRAN, MPI, OpenMP). This special issue presents six projects aimed at resolving this dilemma in favor of remaining in the environment of choice, but augmenting it so that larger problems can be handled with relative ease (although, perhaps, not with absolutely optimal performance). The projects presented cover a good sampling of popular environments: MATLAB, python, R and Ruby, with a touch of Perl on the side. The enhancement strategies span a range of approaches: easy to use interfaces for accessing high performance libraries that were written the "old fashioned" way, parallel versions of apply/map functions, data parallel frameworks, cleanly wrapped RPC systems, and virtual shared memory. Some approaches are specific to one environment; others are applicable to many or most. Taken together, they are indicative of the interest in and the importance of creative responses to the challenge of making HPC accessible to a broad range of users.

Sharma and Martin discuss parallel constructs for MATLAB, one of the most popular commercial systems in this area. They address two related challenges: (i) develop parallel functionality that fits well with the MATLAB computational model;

N. Carriero (✉)
Department of Computer Science, Yale University, New Haven, CT, USA
e-mail: nicholas.carriero@yale.edu

(ii) provide an implementation that can be efficiently deployed and supported across a large base of commercial users.

Seki's work on dRuby and Rinda explores a kind of radical transparency of distributed access in which much is staked on the system's default behavior "doing the right thing". When it does, the result is, as Seki quotes a user, a "stupid easy" system.

Drummond and colleagues' work addresses large scale scientific computing with python. Their emphasis is on streamlining access to well established and efficient high-performance libraries written in compiled languages. The interfaces they have developed reduce coding effort and the need for detailed understanding of the mechanics of the underlying libraries while preserving a high level of performance.

Tierney, Rossini and Li describe *snow*, a package for R—the pre-eminent open source statistical computing environment. snow is at its core a parallel apply mechanism, but one that is augmented with facilities to simplify its application to existing R code and that is relatively agnostic with respect to the underlying communication mechanism. It is an example of a clean, focused solution that meets the needs of a diverse group of users (academic and industrial).

Hudak and colleagues' contribution offers an "academic", open source perspective on HPC enhancements to MATLAB. Their work is distinguished by a concern for efficient execution in the context of the typical supercomputer center. It is designed to simplify the transition from the desktop to the center, to work well with queuing facilities and to make good use of high performance interconnects like InfiniBand and Myrinet.

Finally, Bjornson and colleagues describe NetWorkSpace, a simple explicit coordination model based on a form of shared variables. NWS, by design, offers a straightforward development path from sequential to distributed execution, as well as encourages the construction of heterogeneous ensembles, enabling coordination across many kinds of systems, including all of those discussed in this special issue. NWS is distributed with a facility (implemented on top of NWS) similar to snow that supports the broadly useful parallel apply idiom. An open-source server implementation includes a simple web interface for monitoring NWS activity.