Parallel R, Revisited

Norman Matloff

University of California, Davis
*Contact author: matloff@cs.ucdavis.edu

Keywords: parallel computation, multicore, GPU, embarrassingly parallel

As algorithms become ever more complex and data sets ever larger, *R* users will have more and more need for parallel computation. There has been much activity in this direction in recent years, notably the incorporation of **snow** and **multicore** into a package, **parallel**, included in *R*'s base distribution, and the various packages listed on the CRAN Task View on High-Performance and Parallel Computing. Excellent surveys of the subject are available, such as (Schmidberger, 2009). (Note: By the term *parallel R*, I am including interfaces from *R* to non-R parallel software.)

Yet two large problems loom: First, parallel *R* is still limited largely to "embarrassingly parallel" applications, i.e. those that are easy to parallelize and have little or no communication between processes. Many applications, arguably a rapidly increasing number, do not fit this paradigm.

Second, the platform issue, both in hardware and software support aspects, is a moving target. Though there is much interest in GPU programming, it is not clear, even to the vendors, what GPU architecture will prevail in the coming years. For instance, will CPUs and GPUs merge? Major changes in the dominant architecture will likely have significant impacts on the types of applications amenable to computation on GPUs. Meanwhile, even the support software is in flux, especially omnibus attempts to simultaneously cover both CPU and GPU programming. The main tool currently available of this sort, OpenCL, has been slow to win users, and the recent news that OpenACC, a generic GPU language, is to be incorporated into OpenMP, a highly popular multicore language, may have negative implications for OpenCL. These fast-moving and unpredictably-directed trends in hardware and software make it difficult for *R* to parallelize.

In this talk, I will first discuss the above issues, and then offer two approaches to partly deal with them. The first approach is algorithmic, applicable to general statistical estimators (functions of i.i.d. sample data). Here I replace what typically will be non-embarrassingly parallel computations by embarrassingly parallel asymptotic equivalents. (The asymptotic nature will be seen not to be an issue, since problems that are large enough to need parallel computing will be well past needed convergence levels.)

Second, I will introduce a new *R* package, **Rth**, which will be an *R* interface to Thrust. The latter is in turn an interface NVIDIA provides for CUDA GPU coding, but for which the user also can also take multicore CPU as the backend. Unlike CUDA, OpenCL, OpenMP and the like, Thrust operates at a high level, offering operations as sorting, reduction, prefix sum, search and so on, all usable either on GPUs or multicore CPUs (producing either CUDA or OpenMP code). **Rth** will thus provide *R* users will easy access to these operations in a cross-platform manner, much like Magma does for matrices. I will argue that this kind of hybrid approach (if not this particular implementation) may enabling the *R* community to "hedge their bets" in the face of the uncertain hardware situation.

References

M. Schmidberger *et al* (2009). State of the Art in Parallel Computing with R. *Journal of Statistical Software*, 1–27.