Parallel Computing with Iterators

David Smith[1,*]

1. REvolution Computing, Inc.
* Contact author: david@revolution-computing.com

Keywords: iterator, parallel computing, distributed computing, ParallelR

While the R language provides means for programmer to iterate through the elements of a vector or list, it currently has no support for creating iterator objects as defined in languages like Java or Python. Using an iterator as a "cursor" to a list of values that can be defined dynamically solves a number of problems in R programming. In this talk, we'll begin by introducing a new function for creating iterator objects in R, describe its operators and methods, and provide practical examples of its use.

Although iterators are useful in their own right, one of the most useful applications we have found is in parallel and distributed programming. One of the biggest obstacles to the practical implementation of parallel programming is that for many existing systems a "new way" of thinking about R programming is required. We introduce instead a new function foreach() which we have found to be a natural and elegant construct for programming loops in R which can then easily be run in parallel. foreach() combines the simplicity of a for() loop to repeat arbitrary segments of code, with the power of the lapply() function to iterate over an object (with iterators). foreach() works efficiently to process loops sequentially, but with the addition of the ParallelR library it is trivial to make those same loops run in parallel. This has the result of dramatically reducing the both the time to process loops in R on multicore workstations or clusters, and the time it takes to program and debug them in the first place.