pR: Enabling Automatic Parallelization of Data-Parallel Tasks and Interfacing Parallel Computing Libraries in R with Application to Fusion Reaction Simulations†

Neil Shah[1, 2], Guruprasad Kora[2], Paul Breimyer[1, 2], Yekaterina Shpanskaya[1, 2], Nagiza F. Samatova[1, 2, *]

[1]North Carolina State University; [2]Oak Ridge National Laboratory; *Contact author: samatovan@ornl.gov

The ever-growing size and complexity of modern scientific data sets constantly challenge the capabilities of existing statistical computing methods. High-Performance Statistical Parallel Computing is a promising strategy to address these challenges, especially with the advent of powerful multi-core computing architectures. However, parallel statistical computing techniques introduce many implementation complexities, resulting in a need for more efficient and streamlined processes. In response to this need, we introduce **pR**, a lightweight, easy-to-use middleware for the statistically-rich *R* engine that further enables parallelization in statistical computing. **pR** branches into two main approaches that have strong potential to simplify and improve parallel-computing capabilities: 1) interfacing existing third-party parallel codes with the flexible scripting statistical environment of *R* and 2) supporting the automatic parallelization of data-parallel tasks in hybrid multi-node and multi-core environments.

R's inherent extensibility and flexibility make it an ideal platform for statistical computing. However, *R* has limited native support for parallel computing. To enable parallelization, researchers have previously produced add-on packages such as **Rmpi** and **rpvm** to provide a low-level base for writing parallel codes. Other packages, such as **snow** use these packages as a foundation for handling embarrassingly-parallel statistical computations. However, these packages burden the end-user with the responsibilities of implementing such parallel codes. Additionally, they can also result in slower execution times as a result of the interpreted nature of *R*. By bridging the *R* environment with existing parallel-computing libraries, **pR** allows developers to leverage existing parallel codes written in compiled languages without modifying the package itself. Additionally, **pR** shifts the responsibility of handling parallel programming details away from end-users [1].

Although bridging parallel computing libraries to *R* is a promising approach, an ideal system would automatically execute researchers' serial codes in parallel. However, the Holy Grail of statistical computing is elusive. A simpler, yet powerful approach involves the automatic execution of a single task on multiple sets of data in parallel, thereby avoiding inter-process dependency issues. *R* supports a family of *apply* methods that serially execute a given function to each element in a collection, making it an ideal candidate for automatic parallelization. For example, the *lapply* method in *R* accepts a list and executes a function against each element in that list. Previous parallel implementations of *lapply* have utilized approaches involving assigning the elements of the list to different processes, computing, and gathering results. Current projects implementing the parallel *lapply* function include **snow** and **multicore**, for multi-node and multi-core environments, respectively. However, neither provides support for hybrid multi-node, multi-core environments. To assuage this issue, we extend **pR** to support automatic parallelization of statistical computations in such settings. **pR** implicitly migrates the R environment to each node and distributes data equally among the nodes. The work per node is further divided among cores and, thereby striving for a 'best of both worlds' approach. Using *R*'s *lapply* method, we demonstrate **pR**'s benefits, particularly in improved overall performance and transparent parallelization [2].

In application, **pR** can serve to expedite the end-to-end pipelines of knowledge-discovery workflows and process extreme-scale data in realistic time using hybrid multi-node, multi-core architectures. One tested application of **pR** is the discovery and analysis of turbulent-fronts in simulation data produced by the XGC particle-in-cell gyrokinetic fusion simulation code. In practice, the discovery of fronts in fusion simulations could provide insight to engineering a solution for viable fusion-energy production.

References

1) P. Breimyer, W. Hendrix, G. Kora, N.F. Samatova, 2009. pR: Lightweight, Easy-to-Use Middleware to Plugin Parallel Analytical Computing with R. In *The International Conference on Information and Knowledge Engineering (IKE)*.
2) P. Breimyer, G. Kora, W. Hendrix, N. Shah, N.F. Samatova, 2009. pR: Automatic Parallelization of Data-Parallel Statistical Computing Codes for *R* in Hybrid Multi-node and Multi-core Environments. In *International Association for Development of the Information Society Applied Computing Conference* (*IADIS ACC*).

† We would like to acknowledge Prof. CS Chang for suggesting the application problem. This work was funded by the Scientific Data Management Center under the Department of Energy's SciDAC program. Oak Ridge National Laboratory is managed by UT-Battelle for the LLC U.S. D.O.E. under contract no. DEAC05-00OR22725.