Tight Coupling of R and Distributed Linear Algebra for High-Level Programming with Big Data

Drew Schmidt*, George Ostrouchov†*, Wei-Chen Chen†, and Pragneshkumar Patel*
*Remote Analysis and Visualization Center, University of Tennessee, Knoxville, TN
†Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN

***Abstract*—We present a new distributed programming extension of the R programming language. By tightly coupling R to the well-known ScaLAPACK and MPI libraries, we are able to achieve highly scalable implementations of common statistical methods, allowing the user to analyze bigger datasets with R than ever before. Early benchmarks show great optimism for the project and its future.**

***Index Terms*—Big data, Large scale analytics, Distributed computing, R, ScaLAPACK, MPI**

I. Introduction

R [1] is a domain specific language for data analysis, which is, according to some measures, the most popular such platform [2]. The language syntax has roots in the 1970's at Bell Labs, and was developed specifically as a language for statistical data analysis. Over the decades it went through several implementations and additions, called S, New S, S3, and S4 [3]. Later, the language was licensed and became S+. The most recent implementation of the language is R. The language is a high level mix of functional and object oriented paradigms that are particularly convenient for the mathematics of data and for fast prototyping of analytics. With some effort, one can use native (or nearly native) C code on R objects, and so the implementation of R is a mix of interpreted and compiled components. As such, its speed can vary greatly in different applications.

In the age of big data, R's place in the hierarchy of languages has been called into question. Indeed, R's ability to utilize HPC resources is fairly limited and focused primarily on multicore desktops through a number of packages and some multithreaded libraries. In particular, R's ability to read in, analyze, and visualize big data is virtually non-existent. We report on the "Programming with Big Data in R" project [4] (**pbd**) that enables high-level distributed data parallelism in R, so that it can easily utilize large HPC platforms and begin to use multiple levels of parallelism while retaining its original convenience for mathematics of data. We achieve this through, in part, providing a simplified interface to MPI as well as a tight coupling with distributed dense linear algebra software (PBLAS and ScaLAPACK [5], and their MPI library BLACS [6]). The routines in these libraries are engaged through R's classes and methods, so that the R language syntax is unchanged. While the user is not asked to deal with the details of managing the data distribution and MPI communication, the user is asked to be aware of the data distribution and provided with high-level functions to manage it if needed.

Currently, the **pbd** ecosystem consists of **pbdMPI** [7], **pbdSLAP** [8], **pbdBASE** [9], and **pbdDMAT** [10], which are, respectively, custom MPI bindings for R, a distribution of the ScaLAPACK, PBLAS, and BLACS libraries, a set of distributed data classes and low-level methods for manipulating these distributed objects, and finally, a set of computational methods for the distributed matrix data type. Figure 1 demonstrates the current package landscape.

All **pbd** packages install and run on a single machine as well as on shared memory and distributed clusters. Additionally, the packages fully support major platforms including Linux, Mac OS, and MS Windows via either OpenMPI [11] or MPICH2 [12]. For Linux and Mac OS, The source of packages are available on the "Comprehensive R Archive Network" (CRAN) at http://cran.r-project.org for Linux and Mac OS, while binary packages for MS Windows and development versions are available at the **pbd** website http://r-pbd.org/.

II. Architecture

pbdMPI servers as the bottom layer for the **pbd** project to handle data communication. The design of **pbdMPI** is focused on the Single Program Multiple Data (SPMD) programming paradigm, with MPI communication. The **pbdMPI** package is mainly an interface to MPI but one that is greatly simplified compared to the C API for MPI. Much of this simplification is possible by using

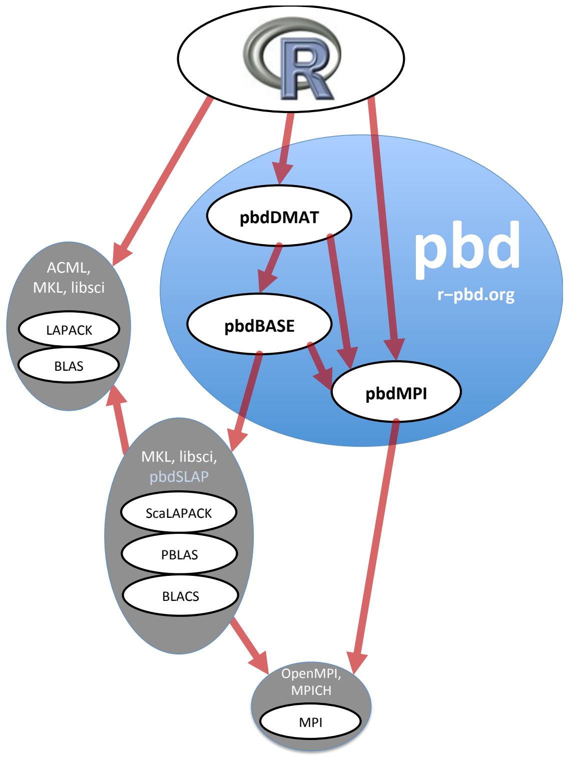

Fig. 1. Dependence graph between the **pbd** packages, linear algebra libraries, and MPI

R's primary method of Object Oriented Programming (OOP), namely R's S4 methods.

Through **pbdMPI**, a user is able to use R on large distributed machines communicating over MPI without needing to be experts at MPI programming. For example, collective calls can be done simply as:

```
x <- bcast(x)
x <- allgather(x)
x <- allreduce(x, op = "sum")
x <- reduce(x, op = "sum")
x <- scatter(x)
```

where `x` can be any R object as long as calls are valid, and neither data type nor buffer need to be specified by the user. However, advanced programmers do not need to worry about losing performance via the simplified API.

The **pbdSLAP** package contains a subset of double precision subroutines from the open source ScaLAPACK library available from netlib at http://www.netlib.org/scalapack/. However, by adding a simple flag when installing the package, the user is able to use more highly tuned ScaLAPACK implementations, such as MKL [13] or LibSci [14].

Additionally, the package provides linking information for other **pbd** packages which need to make use of ScaLAPACK functionality, such as **pbdBASE** and **pbdDMAT**. The package provides a library for further extensions. Thus, if a developer desires to build an R package using some ScaLAPACK functionality, then the developer can simply build his or her package on top of **pbdSLAP**. In doing so, the developer need not worry about many of the more complicated compiling and linking issues involved with this process; indeed, we settled those issues for them. Finally, the package provides some basic functionality for testing the specialized MPI communicator used by ScaLAPACK.

The remaining two packages, **pbdBASE** and **pbdDMAT**, are the infrastructure that supports distributed matrix computation. The former, **pbdBASE**, includes, among other things, a set of efficient wrappers for the ScaLAPACK, PBLAS, and BLACS subroutines provided by **pbdSLAP**. These wrappers offer the R interpreter access to the low level MPI and Fortran routines in such a way that these functions can interact natively on R objects (so that extra data copies are not created). This is done in three phases. From the bottom up, a simple Fortran wrapper is written to act as a generic routine for calling the desired foreign library subroutine. Next, a C wrapper is written as a go-between for this Fortran subroutine and the R interpreter (a C program), allowing the programmer to natively operate on R data structures. Finally, a simple R wrapper is written which prepares the additionally necessary information for the foreign function call and then calls the C function. Lastly, **pbdDMAT**, contains most of the methods for the distributed matrix class involving computation, including linear algebra. These allow the user to achieve very high level access to the low level MPI and Fortran routines which interact natively on R objects.

III. Distributed Matrix Computing

Currently, there is one distributed data type, `ddmatrix`, which is a class for storing real-valued *d*istributed *d*ense *matrix* data. This class and its methods have been developed with existing R users in mind, so that new objects and operations behave as closely as possible to existing ones. Each processor instantiates a class object with a common name among all processors, but with (potentially) different local members. The class object contains the following "slots", or members:

- the local submatrix
- the global matrix dimension
- the dimension of the local submatrix
- blocking factor for the block-cyclic distribution
- the special MPI communicator information

At present, all ddmatrix methods assume a block-cyclic data distribution, which ScaLAPACK requires [5]. Of note, ScaLAPACK performs best on a block-cyclic data distribution across a (non-degenerate) 2-dimensional grid of processors. However, this is an incredibly cumbersome data structure, which is, at best, highly inconvenient for the common data restructuring tasks R users are accustomed to, such as dropping rows and columns of a matrix on the fly. As such, the **pbdBASE** package includes data redistribution methods for moving between different data distributions. For example, the user can effortlessly redistribute data between a general $P \times Q$ grid of processors, a $1 \times PQ$ grid, and a $PQ \times 1$ grid. The latter two are convenient for reading and manipulating data, while the former is most beneficial for analysis using ScaLAPACK linear algebra routines. It is worth noting that data redistribution across processors can be costly and is only performed implicitly when it is deemed absolutely necessary.

For the sake of demonstration, we will consider a seemingly contrived example. Suppose that we have a distributed data matrix x, and suppose for some reason that we first wish to apply a log transformation to the absolute value of the data, and then find the Cholesky factorization of the matrix:

$$\left(x^T x\right)^{-1}$$

Implementing this is simple enough:

```
x <- log(abs(x))
xtx <- t(x) %*% x
xtx.inv <- solve(xtx)
ans <- chol(xtx.inv)
```

It is worth mentioning that this is identical to the syntax one would use in R on a non-distributed matrix. This computation is actually related to computing the square root of a covariance matrix inverse, which can be used to decorrelate data.

IV. Performance Results

We performed a simple test to compare the performance of our newer **pbdMPI** to the original package of MPI bindings for R, **Rmpi** [15]. The test consisted of calling the allgather collective function on a $10,000 \times 10,000$ distributed matrix with different numbers of processors. We used the allgather function in **pbdMPI** and the corresponding mpi.allgather.Robj function in **Rmpi** to aggregate the matrix on all processors. The data were randomly generated from the standard normal distribution. Table I shows the results for this test.

TABLE I
Runtimes (seconds) for **Rmpi** and **pbdMPI**.

Cores	**Rmpi**	**pbdMPI**	Speedup
32	24.6	6.7	3.67
64	25.2	7.1	3.55
128	22.3	7.2	3.10
256	22.4	7.1	3.15

We also performed some preliminary benchmarks for packages **pbdBASE** and **pbdDMAT**, and they are equally optimistic. We report on several runs of principal components analysis (PCA) [16] using different numbers of processors on a randomly generated $10,000 \times 10,000$ matrix with random normal data. In brief, a principal components analysis is a very common technique in statistics which is, at its core, a dressed up singular value decomposition (SVD). Internally, the code also chains together a few minor computations and linear algebra operations, which amount to projecting the data matrix onto its right singular vectors. At no time is a costly and cumbersome re-distribution of data necessary, so that the communication required is only a small amount of overhead in addition to that required by ScaLAPACK.

This benchmark used R's native code for the serial case and a combination of **pbdDMAT**, **pbdBASE**, **pbdSLAP**, and **pbdMPI** for the several-core tests. Table II shows the results of this series of tests.

TABLE II
Full PCA computation on a $10,000 \times 10,000$ matrix.

Cores	Seconds	Speedup	Speedup64
1	14,908	1	
64	297	50	1.00
128	177	84	1.68
256	113	131	2.63
512	83	179	3.58

With some pride, we note that from the user's perspective, there are no complicated parallel programming techniques to master, nor must the user learn the, certainly by comparison, cumbersome Fortran language, nor the ScaLAPACK library and its highly specialized MPI communicator. Indeed, the user need only issue the command:

```
Z_pca <- prcomp(Z, scale = TRUE)
```

on a distributed matrix, which is a carbon copy of the serial R syntax. This gives the scientist access to powerful tools for scalable programming with big data, using a syntax with which he or she is already familiar, bridging the gap between idea and implementation.

Beyond merely doing computations faster, our work allows us to break a computing barrier inherent to R at the time of writing. Currently, the largest square matrix which is possible to hold in memory in a single R process is $46,340 \times 46,340$. However, by distributing our data, we are able to smash this cap, making R into an analytics powerhouse for exploring and analyzing big data. Table III shows a second benchmark using our **pbd** tools. Again we perform a PCA benchmark, but here we do so on a $50,000 \times 50,000$ matrix with many more cores. In reality, this problem is a bit small for the number of cores we use here, explaining the less than ideal scaling. However, this is well beyond the capabilities of R, in both the size of the data set we analyze, as well as the number of cores utilized.

TABLE III
FULL PCA COMPUTATION ON A SQUARE MATRIX.

	$50,000 \times 50,000$		$100,000 \times 100,000$	
Cores	Seconds	Speedup	Seconds	Speedup
1,024	3,842	1.00	11,730	1.00
2,048	2,816	1.36	7,508	1.56
4,096	2,178	1.76	5,174	2.27
8,192	1,976	1.94	4,223	2.78
12,000	1,875	2.05	3,874	3.02

V. Future Work

There is much work planned for future development, not just within existing packages, but also in adding more functionality to the **pbd** ecosystem through the creation of additional packages. Currently, we anticipate introducing more distributed data types and associated methods, interfaces to parallel readers, a set of benchmarking tools, as well as a client/server interface. Longer term plans include analytics to help manage the data distribution and multiple levels of parallelism.

R has interfaces to many other software tools, as is evident from the Omega Project for Statistical Computing [17] repository. It is also often embedded in other software as an analytics engine. However, these generally do not take advantage of HPC resources. The package **pbdMPI** can accept an MPI communicator from other software. This provides R with the capability of managing data that is already distributed with minimal or no data movement. Our first effort to this end is a preliminary work embedding **pbd**-enabled R in VisIt [18]. This benefits both R and VisIt. Namely, R gains a powerful visualization engine, with the added benefit of additionally gaining access to parallel readers for many data formats, including ADIOS, NetCDF, and HDF. Additionally, VisIt gains a diverse and powerful analytics and graphics engine. This is an early work, and so performance results are not yet available.

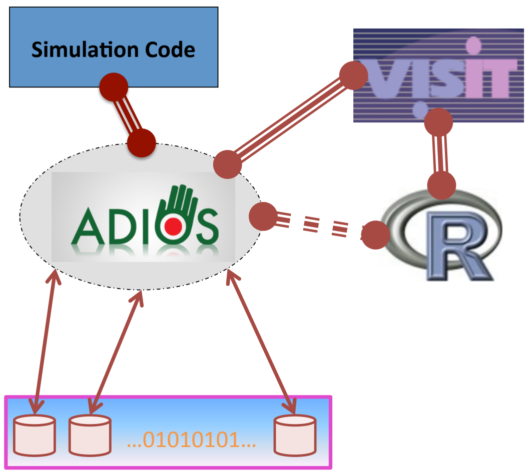

Fig. 2. Planned tight integration between ADIOS, VisIt, and R for potential in situ analysis and visualization.

Acknowledgment

D. Schmidt, G. Ostrouchov, and P. Patel were supported in part by the project "NICS Remote Data Analysis and Visualization Center" funded by the Office of Cyberinfrastructure of the U.S. National Science Foundation under Award No. ARRA-NSF-OCI-0906324 for NICS-RDAV center. G. Ostrouchov and W.-C. Chen were supported in part by the project "Visual Data Exploration and Analysis of Ultra-large Climate Data" funded by U.S. DOE Office of Science under Contract No. DE-AC05-00OR22725.

This work used resources of National Institute for Computational Sciences at the University of Tennessee, Knoxville, which is supported by the Office of Cyberinfrastructure of the U.S. National Science Foundation under Award No. ARRA-NSF-OCI-0906324 for NICS-RDAV center. The computations were performed on Kraken and Nautilus at the National Institute for Computational Sciences.

This work also used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. This work used resources of the Newton HPC Program at the University of Tennessee, Knoxville.

References

[1] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2012, ISBN 3-900051-07-0. [Online]. Available: http://www.r-project.org/

[2] R. Muenchen, "The popularity of data analysis software," 2012. [Online]. Available: http://r4stats.com/articles/popularity/

[3] J. Chambers, *Software for Data Analysis: Programming with R*, ser. Statistics and Computing. Springer, 2008.

[4] G. Ostrouchov, W.-C. Chen, D. Schmidt, and P. Patel, "Programming with big data in R," 2012. [Online]. Available: http://r-pbd.org/

[5] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley, *ScaLAPACK Users' Guide*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1997. [Online]. Available: http://netlib.org/scalapack/slug/scalapack_slug.html/

[6] J. Dongarra and R. C. Whaley, "A user's guide to the blacs," University of Tennessee, Tech. Rep., mar 1995, uT-CS-95-281.

[7] W.-C. Chen, G. Ostrouchov, D. Schmidt, P. Patel, and H. Yu, "pbdMPI: Programming with big data – interface to MPI," 2012, R Package. [Online]. Available: http://cran.r-project.org/package=pbdMPI

[8] W.-C. Chen, D. Schmidt, G. Ostrouchov, and P. Patel, "pbdSLAP: Programming with big data – scalable linear algebra packages," 2012, R Package. [Online]. Available: http://r-pbd.org/

[9] D. Schmidt, W.-C. Chen, G. Ostrouchov, and P. Patel, "pbdBASE: Programming with big data – BASE," 2012, R Package. [Online]. Available: http://r-pbd.org/

[10] ——, "pbdDMAT: Programming with big data – distributed matrix algebra computation," 2012, R Package. [Online]. Available: http://r-pbd.org/

[11] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall, "Open MPI: Goals, concept, and design of a next generation MPI implementation," in *Proceedings, 11th European PVM/MPI Users' Group Meeting*, Budapest, Hungary, September 2004, pp. 97–104.

[12] W. Gropp, "Mpich2: A new start for mpi implementations," in *Proceedings of the 9th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*. London, UK, UK: Springer-Verlag, 2002, pp. 7–. [Online]. Available: http://dl.acm.org/citation.cfm?id=648139.749473

[13] Intel Corporation, "Intel Math Kernel Library (Intel MKL)," 2010. [Online]. Available: http://software.intel.com/en-us/intel-mkl

[14] Cray, "Cray xt-libsci," 2011. [Online]. Available: http://www.nersc.gov/users/software/programming-libraries/math-libraries/libsci/

[15] H. Yu, "Rmpi: Interface (wrapper) to mpi (message-passing interface))," 2010, R Package (v:0.5-9). [Online]. Available: http://cran.r-project.org/package=Rmpi

[16] A. Rencher, *Methods of Multivariate Analysis*, ser. Wiley series in probability and mathematical statistics. Probability and mathematical statistics. John Wiley & Sons, 2002. [Online]. Available: http://books.google.bg/books?id=SpvBd7IUCxkC

[17] D. Bates, J. Chambers, D. Cook, P. Dalgaard, R. Gentleman, K. Hornik, R. Ihaka, F. Leisch, T. Lumley, M. Mächler, G. Masarotto, P. Murrell, B. Narasimhan, B. Ripley, G. Sawitzki, D. Lang, L. Tierney, and B. Venables. The omega project for statistical computing. [Online]. Available: http://www.omegahat.org/

[18] H. Childs, E. Brugger, K. Bonnell, J. Meredith, M. Miller, B. Whitlock, and N. Max, "A contract based system for large data visualization," in *Visualization, 2005. VIS 05. IEEE*, oct. 2005, pp. 191 – 198.