Combining R with Scalable Libraries to Get the Best of Both for Big Data

George Ostrouchov [1ab], Drew Schmidt[b], Wei-Chen Chen[a], Pragneshkumar Patel[b]

[a]Computer Science and Mathematics Division, Oak Ridge National Laboratory
[b]National Institute for Computational Sciences, University of Tennessee

Abstract

The R programming language is known for its diversity and sophistication in data analysis, however its scalability to big data has been lacking. Our project ”Programming with Big Data in R” (pbdR) is adding scalability to its list of data virtues. We have developed several packages that provide a tight coupling of R with highly scalable libraries, enabling scalability to terabytes of data on tens of thousands of cores. We also added classes and methods to handle distributed data objects needed by the libraries so that the R language syntax is largely unchanged. Our philosophy is that the R developer is not asked to deal with the details of managing the data distribution and processor communication but the developer is asked to be aware of the data distribution and provided with high-level functions to manage it if needed. Many R functions are already instrumented to handle the distributed data classes. We encourage developers of compute intensive R packages to use pbdR methods for scalability to bigger data and to bigger computing platforms.

Keywords: pbdR, parallel computing, distributed computing, distributed data

1. Introduction

Practical research in parallel computing began in earnest when the first multicomputers were produced about thirty years ago. These machines and almost every supercomputer produced since that time are distributed memory computers. It is only the last ten years that begin the widespread use of multiple cores per processor and the use of co-processors, but still the “outer structure” is a distributed memory architecture. As a result, distributed computing is considered mature, while leading edge research now concentrates on utilizing shared memory resources and multiple layers of parallelism. A number of excellent scalable libraries are now available for solving various problems on distributed computers. Most notably, libraries for dense matrix operations and for solving dense linear algebra problems can now scale to hundreds of thousands of processors. Excellent libraries for sparse linear algebra are also available (Heroux et al., 2003; Balay et al., 1997).

It is only in the last ten years that parallel computing became a mainstream activity forced by stagnant clock rates of new processors and by the introduction of multiple cores per processor. In response to this, parallel computing activity in `R` picked up considerably and concentrated overwhelmingly on the use of the multiple cores in a manager-workers style, but not on distributed parallelism. Distributed parallelism has been available in R for a very long time (Li and Rossini, 2001; Yu, 2002), but its use has been mostly limited to a simple manager-workers paradigm.

In this paper, we describe the `pbdR` (Ostrouchov et al., 2012) series of `R` packages, which begin the work to integrate mature results from the parallel computing community into `R` and the work of bringing a high-level language for data analysis to supercomputers. To achieve this, we are careful to capture the mainstream style of parallel computing as we stand R on top of several scalable libraries so that performance does not suffer, and to abstract many low-level details away so that a high-level approach to distributed data analysis becomes available.

[1]Corresponding author: Computer Science and Mathematics Division, Oak Ridge National Laboratory, 1 Bethel Valley Road, Oak Ridge, Tennessee 37830, USA. E-mail: ostrouchovg@ornl.gov

The term "Big Data" is used to describe data sets where, loosely, data size is part of the problem that needs to be solved. This is certainly the case when the data do not fit in the memory of a single multicore processor and must be partitioned across multiple processors. Our pbdR methods and packages are the most compelling for such data because the collective memory of a distributed multiprocessor computer (a cluster) can be orders of magnitude larger than that of a single processor.

An important point is that MPI-based codes (MPI Forum, 2013), like those utilizing our pbdR packages, that are implemented for distributed memory processors will also work on shared memory architectures, such as multicore processors, with no modification. The opposite does not hold. Codes designed for shared memory processors can not utilize the full distributed machine without non-trivial modification. Distributed memory architectures are necessary for scaling to very high concurrency. This is evidenced by the absence of wholly shared memory architectures on the Top 500 list of supercomputers (Meuer, Strohmaier, Dongarra, and Simon, Meuer et al.). Of course, ideally a code can take advantage of more than one level of parallelism, as is the current practice on supercomputers.

In the remainder of the paper, we first describe some basic concepts and libraries in distributed memory parallel computing (Section 2) then Section 3 describes our packages. Finally, in Section 4, we discuss the implications of this work and a roadmap for the future of R analytics on parallel architectures.

2. Distributed Computing

Message passing between processors is necessary whenever the processors are working collectively on a data set that has been partitioned between them. Cluster computers are collections of processors, each with its own memory and possibly multiple cores, that are connected with an ultra fast communication network. This architecture model spans early parallel computers built in the 1980's through today's supercomputers. Today's supercomputers add more complexity by having large core counts and possibly co-processors within each processor, but the outer structure is still a distributed memory architecture. The accepted programming model for these architectures is single program multiple data (SPMD), meaning that an identical program is run on each processor. The program is executed asynchronously on each of the processors while communicating via MPI to coordinate the solution of a given problem. The logical number of the processor in the collection (known as processor "rank") is used by the program to decide what section of data to process and any other local differences. There is no separate master process as one would expect in an interactive session because the execution is in batch. The SPMD batch parallel programming model has a long history, even in statistical computing (Ostrouchov, 1987), but its use to date in R has been infrequent. It is a programming model that is more scalable than a map-reduce abstraction and that is more natural to use, particularly when solving more complex problems that would require map-reduce hierarchies or iterations.

Because of the many years of experience and research surrounding distributed memory architectures, a number of excellent scalable libraries are available. At the base of these libraries is the MPI standard (MPI Forum, 2013), which has several library implementations. The most popular of these are OpenMPI (Gabriel et al., 2004) and MPICH (Gropp et al., 1996). In addition, there are several vendor specific implementations.

Extensive research was conducted in the 1980's into distributed dense linear algebra algorithms, resulting in many such algorithms published by the end of that decade (Ortega et al., 1989). This set the stage for the development of scalable libraries for distributed dense linear algebra computations. The best known of these are ScaLAPACK (Blackford et al., 1997) and PBLAS, which are the distributed (near-)equivalents of LAPACK (Anderson et al., 1999) and the BLAS, respectively. A third library, BLACS, is an intermediary layer between these libraries and MPI. It provides a communication layer specialized for dense linear algebra computations. Figure 1 shows the dependence structure between the libraries. For distributed architectures, these libraries are still state-of-the-art today. While R uses LAPACK and the BLAS, before the development of pbdR it was not possible to use ScaLAPACK and PBLAS from a set of distributed R instances.

As scalable dense linear algebra research takes account of new architectures with several levels of parallelism, some faster approaches such as DPLASMA (Bosilca et al., 2011) are beginning to emerge but they still have a limited portfolio of available operations. As the ScaLAPACK API is so heavily used across many

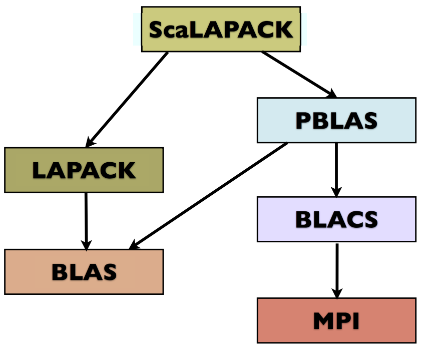

Figure 1: ScaLAPACK architecture

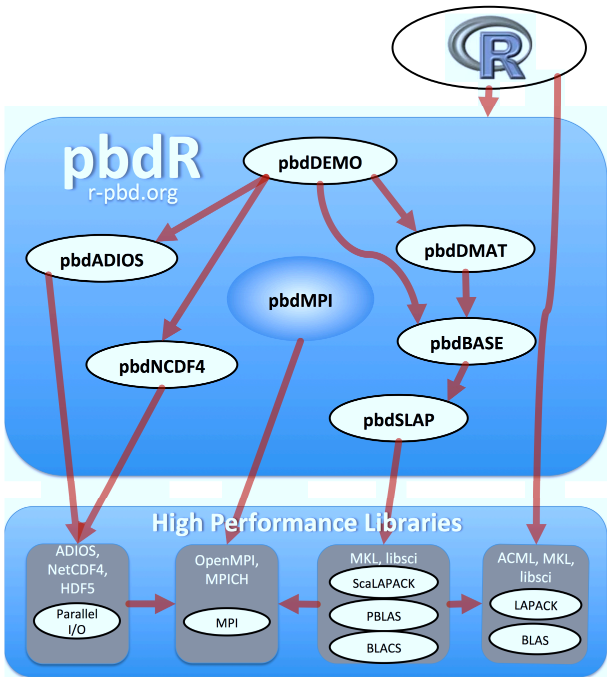

Figure 2: pbdR architecture

applications, it will remain a useful interface between application, such as data analysis with pbdR packages and state-of-the-art scalable dense linear algebra even after ScaLAPACK itself is retired.

3. Programming with Big Data in R: pbdR

Our philosophy is to let the experts in parallel computing and in scalable linear algebra deal with basic algorithms in dense or sparse matrix computation. Computational linear algebra is a complex field, and a novice is unlikely to outperform its experts, both in terms of performance and in computational accuracy. When mature libraries for these computations are available and are extensively used by various applications, it is safe to use their API to enable the same scalability in R. So it is natural to have a separating boundary (between R and dense matrix algebra) of a standard API for a library such as ScaLAPACK. We are experts in using these libraries for data analysis purposes, so that we build the high level interface to these libraries that is natural for data analysis, much in the same way that R does with serial BLAS and LAPACK. On the other hand, there are some instances when the dense linear algebra community does not anticipate all the needs for data analysis and we provide some additional functionality (possibly in collaboration with that community to ensure numerical stability).

We have built several packages to perform statistical computations in R in a tightly coupled manner with these libraries. This enables R to scale to very large data sets on thousands of processors. At the same time, we respect standard R syntax so that no (or minimal) changes to serial code are required. We use R's S4 methods (Chambers, 2006) whenever possible to keep identical syntax. We hide the distributed nature of the data, but we provide high level functions to manage the data layout and to examine it when needed. Best scalability is achieved when the user understands the data layout and manages it for optimality. However, reasonable scaling can be achieved in most cases by using our defaults. Figure 2 describes the dependence structure of our packages and the underlying scalable libraries. All of our packages are released and maintained on the Comprehensive R Archive Network (CRAN). Much more information as well as a link to our GitHub development versions is on our web site r-pbd.org. In the following, we give a brief description of each package in the `pbdR` ecosystem.

pbdMPI: At the core of `pbdR` is the **pbdMPI** package, which is a high-level interface to MPI for `R`. Unlike **Rmpi** (Yu, 2002), **pbdMPI** focuses on SPMD programming, improves `R` object communication speed, and simplifies function calls via S4 methods. Leveraging **pbdMPI**, MPI methods can be efficiently utilized without the programmer having to worry about the `R` object's underlying storage type. Data scientists can focus more on developing algorithms or analyzing large data instead of low-level programming details. Finally, **pbdMPI** also provides and exports information for compiling and linking with other MPI-using `R` packages, such as the `pbdR` packages, making the integration of high-performance `C` and `Fortran` codes with `R` even simpler. This tight integration means that distributed data is shared as a collective with other MPI-enabled software.

pbdSLAP, **pbdBASE**, and **pbdDMAT**: This set of packages, as far as the user is concerned, really just consist of **pbdDMAT**, the (dense, in-core) linear algebra and statistics layer. The other packages are split off on their own for development reasons. For instance, **pbdSLAP** is a distribution of ScaLAPACK. Ordinarily, ScaLAPACK can be very difficult to link; however, in bundling this distribution ourselves, the installation process becomes as simple as issuing an `install.packages()` command from `R`. Next, the **pbdBASE** package is the low-level interface to high performance `C` and `Fortran` codes which utilize ScaLAPACK. This is essentially a very shallow `R` layer atop ScaLAPACK and our ScaLAPACK-using extensions. Finally, **pbdDMAT** offers the high-level abstraction layer that makes all this high performance code human-usable. The "secret sauce" is a special S4 object for distributed matrices, together with numerous methods, including covariance, principal components, and linear model fitting.

pbdNCDF4 and **pbdADIOS**: Input and output (I/O) is a (often *the*) major bottleneck for analyzing and visualizing big data. The **pbdNCDF4** package is a parallel extension to the **ncdf4** package (Pierce, 2012), and is a tight coupling of the NetCDF library (NetCDF Group, 2008). In short, this package provides access to an I/O standard in the HPC community for handling very large datasets. In particular, **pbdNCDF4** focuses on parallel I/O, including methods for collectively reading and writing big datasets. With the help of parallel file systems such as Lustre, this package enables data access to and from disk in parallel. The package **pbdADIOS** is still under development. It enables use of the highly scalable ADIOS I/O library (Lofstead et al., 2008) on supercomputers.

pbdDEMO: Finally, the **pbdDEMO** package is a unifying package, tying all of the `pbdR` packages together through a set of over 30 demos and a 135 page (and growing), textbook-style (human-readable) vignette. This package is designed to take a researcher who is reasonably fluent in `R` and have him/her writing scalable codes using `pbdR` in no time. It is the best place in the `pbdR` "alphabet soup" package landscape for a beginner to start working with these tools.

4. Discussion

Our very early scalability results are reported in (Schmidt et al., 2012), where we demonstrate principal components computation on up to 12,000 cores. Since that time, we have scaled these and other computations to over 50,000 cores. We expect to report these details elsewhere. The scalability of `pbdR` is not unlike that of ScaLAPACK, the underlying HPC library.

We hope that through the ability of `R` to scale to supercomputers the statistics community becomes more active in supercomputing. In particular, we encourage developers of compute-intensive `R` packages to consider standing them up on the `pbdR` infrastructure for scalability. The diversity of `R` requires community involvement to get a large proportion of packages into scalable shape.

Many research funding agencies, such as the National Science Foundation in the United States and the PRACE program in the European Union, provide access to supercomputers through various national programs, sometimes nearly "just for the asking" (Ostrouchov, 2013). We encourage `R` developers to enquire about the availability of such programs in your country.

Supercomputing is still, for the most part, utilized by simulation science. As this endeavor is heading toward exascale computing, it will no longer be feasible to output all simulation data. More analysis of the data becomes necessary in tandem with the simulation. For this reason, it becomes necessary for data analysis software to scale on the same computational resources. As a result, some of our current effort is directed toward running in-situ with simulations.

Acknowledgement

This research was supported in part by the project "Visual Data Exploration and Analysis of Ultra-large Climate Data" and used the resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory both funded by U.S. DOE Office of Science under Contract No. DE-AC05-00OR22725. Additional support was provided by "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.

References

Anderson, E., Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen (1999). *LAPACK Users' Guide* (Third ed.). Philadelphia, PA: Society for Industrial and Applied Mathematics.

Balay, S., W. D. Gropp, L. C. McInnes, and B. F. Smith (1997). Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen (Eds.), *Modern Software Tools in Scientific Computing*, pp. 163–202. Birkhäuser Press.

Blackford, L. S., 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 (1997). *ScaLAPACK Users' Guide*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Bosilca, G., A. Bouteiller, A. Danalis, M. Faverge, A. Haidar, T. Hérault, J. Kurzak, J. Langou, P. Lemarinier, H. Ltaief, P. Luszczek, A. YarKhan, and J. Dongarra (2011). Flexible development of dense linear algebra algorithms on massively parallel architectures with dplasma. In *IPDPS Workshops*, pp. 1432–1441.

Chambers, J. (2006, Aug). How s4 methods work. Technical report, The R-Project for Statistical Computing.

Gabriel, E., 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 (2004, September). OpenMPI: Goals, concept, and design of a next generation MPI implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, Budapest, Hungary, pp. 97–104.

Gropp, W., E. Lusk, N. Doss, and A. Skjellum (1996, September). A high-performance, portable implementation of the mpi message passing interface standard. *Parallel Comput. 22*(6), 789–828.

Heroux, M., R. Bartlett, V. H. R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, and A. Williams (2003). An Overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories.

Li, N. and A. Rossini (2001). R interface to pvm (parallel virtual machine).

Lofstead, J. F., S. Klasky, K. Schwan, N. Podhorszki, and C. Jin (2008). Flexible IO and integration for scientific codes through the adaptable IO system (ADIOS). In *Proceedings of the 6th international workshop on Challenges of large applications in distributed environments*, CLADE '08, New York, NY, USA, pp. 15–24. ACM.

Meuer, H., E. Strohmaier, J. Dongarra, and H. Simon. TOP500 Supercomputer Site.

MPI Forum (2013). Message passing interface. https://www.mpi-forum.org.

NetCDF Group (2008). Network common data form. Software package.

Ortega, J. M., G. G. Voight, and C. H. Romine (1989). Bibliography on parallel and vector numerical algorithms.

Ostrouchov, G. (1987). Parallel computing on a hypercube: An overview of the architecture and some applications. In R. M. Heiberger (Ed.), *Proc. 19th Symp. on the Interface of Computer Science and Statistics*, Washington, D.C., pp. 27–32. American Statistical Association.

Ostrouchov, G. (2013, June). A call for participation in XSEDE computing: Statisticians needed for Big Data. *AMSTAT News* (432), 31–32.

Ostrouchov, G., W.-C. Chen, D. Schmidt, and P. Patel (2012). Programming with Big Data in R.URL http://r-pbd.org/.

Pierce, D. (2012). ncdf4: Interface to unidata netcdf (version 4 or earlier) format data files. R Package.

Schmidt, D., G. Ostrouchov, W.-C. Chen, and P. Patel (2012). Tight coupling of R and distributed linear algebra for high-level programming with Big Data. In *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, pp. 811–815. IEEE.

Yu, H. (2002). Rmpi: Parallel statistical computing in r. *R News 2*(2), 10–14.