On Distributed R Computations over BOINC

Alexander Rumyantsev[3,4] Anna Eparskaya[3] Enrico Blanzieri[1] Valter Cavecchia[2]

[1]DISI, Department of Information Engineering and Computer Science
University of Trento, Via Sommarive 9, 38123 Povo (TN), Italy
[2]CNR-IMEM
Via alla Cascata 56/C, 38123 Trento, Italy
[3]Petrozavodsk State University
33 Lenina Pr., Petrozavodsk, 185910, Russia
[4]Institute of Appiled Mathematical Research, Karelian Research Centre of RAS
11 Pushkinskaya Str., Petrozavodsk, 185910, Russia
ar0@krc.karelia.ru

Abstract

We discuss the technologies for executing R language applications in high-performance and distributed computing environments. The concept of running R applications in BOINC infrastructure is presented. Possible applications that might benefit from the distributed computing environment available in R are discussed.

1 Introduction

High-performance and cloud computing is keeping the dominant positions in supplying the enterprise and scientists with computing power. However, relatively high capital expenses (for a high-performance computing system), or high operational expenses (in case of a cloud-based service/infrastructure) prevents usage of these types of resources by small research groups. Volunteer Computing (a particular case of the Desktop Grid paradigm) allows to overcome these difficulties, equipping the scientists with a relatively large amount of computing power with little to no price. However, this approach is mainly applicable to some special classes of tasks, in particular, the so-called embarrassingly parallel applications (e.g. parameter sweep experiments). In this class, the task requires a large amount of computing power, but it can be separated into small subtasks, that may be independently completed in a parallel/distributed system. Afterwards the solution of the whole task is assembled from the results of the subtasks.

One of the most actively developed environments for the simulation experiments in various fields of applications (including bioinformatics, medical statistics, machine learning etc.) is the R language [R C17]. There are several methods to make the R application run over a parallel or distributed computing system. However, there is still no package that allows to use Volunteer Computing resources in a flexible and straightforward way. For this reason, we present the concept of RBOINC package, that will allow to run R scripts over BOINC grid [And04], currently the most sophisticated software to organize Volunteer Computing.

The structure of the paper is as follows. In section 2 we discuss the methods of computations acceleration provided by R extension packages. In section 3 we present the concept of RBOINC package. In section 4 we discuss a possible application case for the presented package.

In: E. Ivashko, A. Rumyantsev (eds.): Proceedings of the Third International Conference BOINC:FAST 2017, Petrozavodsk, Russia, August 28 - September 01, 2017, published at http://ceur-ws.org

2 High-Performance and Distributed Computing in R Language

In this section we summarize the methods of parallel and distributed execution of an R script/command. We refer the reader to an extensive review of R parallel and distributed computing `https://cran.r-project.org/web/views/HighPerformanceComputing.html`, whereas we provide an alternative grouping of packages, based on the level of abstraction.

First we note, that code vectorization is one of the basic concepts of R language, and many basic functions accept vector arguments. As well as providing computations speedup, it also makes a specific language usage pattern based on the `apply`-type functions (applying a function to a vector/array of arguments) as an alternative to `for` loops. At that, one of the expected ways of organizing *implicit* parallelism in R is done by a background parallelization of the `apply` function. This programming style is more natural for applied scientists, who prefer to focus on their application rather than optimize the code. Alternatively, we expect that a parallel programming professional, who is using R as one of the steps in the analysis, would use (at various abstraction level) wrappers of parallel programming instruments, such as MPI, OpenMP, CUDA, OpenCL etc., which we will refer below as *explicit* parallelism. There are also some distinct promising concepts of parallel code execution, such as the futures concept (packages `future` and `future.batchjobs`) for running independent jobs on parallel and distributed systems, or `ddR` package family for distributed data analysis, however, we will focus below on the first two approaches.

The explicit parallelism model requires low-level programming and expertise in the high-performance and/or distributed computing field, but provides a more computationally effective code. Traditionally, writing an explicitly parallel application in R required the following steps:

1. choosing and configuring a parallel/distributed backend (including some configuration work outside the R environment);
2. writing the code with parallel execution in mind, including the data distribution and result aggregation, self identification of the process in Single Instruction Multiple Data model, data transfer by means of MPI, Sockets etc.;
3. caring about load balancing, interaction with schedulers and cluster management software (if any), as well as fault tolerance.

This workflow has recently been added to the core functionality of R language, by means of the `parallel` package (which is distributed as an essential package with the R core), inheriting the shared-memory multicore computing features (formerly provided by `multicore` package) and the multiserver computing features on a Beowulf-type cluster, known as Simple Network of Workstations, with socket-based communications (the functionality provided by a deprecated `snow` package). When the application requires intensive data transfer between computing nodes, the `Rmpi` package is used as a wrapper for MPI API. Finally, the `parallel` package provides the `mcapply` command allowing to perform parallel execution of `apply`-type fuction.

A higher-level abstraction of the explicit parallelism model is provided with a combination of packages `foreach` and the `do*` package family (including `doParallel`, `doMC`, `doMPI`, `doSNOW` etc.) built over the aforementioned `parallel` package. This combination is based on the concept of iterators (recalling the `for` loops with asynchronous execution of loop body without an explicit loop counter), and allows to organize not only data parallel processing, but also working with streaming data. The `do*` packages allow to organize the backend, including working with high-performance clusters, whereas the `foreach` package with a special `%dopar%` pragma allows to parallelize the loop body execution. Note, that the code style provided by this package combination resembles more the C++ programming, rather than R scripting. In this regard, even more performance-oriented and fine-grained parallel programming can be done with special package wrappers for parallel programming tools outside R environment, including `RcppParallel` and `RcppBlaze`. The package `snowfall` allows to implement the master–worker parallel execution scheme over an MPI cluster, however, with an unavoidable configuration work, including LAM/MPI configuration. A similar functionality is provided by an `Rhpc` package (with a programming style resembling writing an MPI equipped parallel application on C++/Fortran). The `multidplyr` package, acting as a backend for `dplyr` package and based on the functionality of `parallel`, is more focused on the applied analysis, however, still requiring some explicit work such as data spreading and cluster initialization.

High-performance computing on GP-GPU and co-processors is widely used for specific type of parallel applications. Following this trend, a number of packages is available for R language, which either are interfaces for the specific language (e.g. `OpenCL`, `RCUDA`, `RViennaCL`), or implement specific algorithms in various research

fields, such as data mining (gputools), statistics (cudaBayesreg), and bioinformatics (permGPU). There are also some supplementary packages dealing with data structures on the GPU and data transfer between the GPU/co-processor and the host machine (gmatrix, gpuR).

Distributed computing with distributed data structures is a natural way of working with big datasets. At that, several R packages implement the explicit distributed computing methods, some of them relying on the widely used backends, such as Hadoop, e.g. datadr, fileplyr, DSL, startR, or a web application oriented package distcomp. We note, however, that the aforementioned packages for distributed computing do not directly fit the Volunteer Computing model, but instead, are capable of working in an infrastructure and environment controlled by the code writer (e.g. require remote access to machine memory etc.). Among the several attempts to create a package for Volunteer Computing backend (including the archive GridR package), to the best of our knowledge, there is no actively maintained package.

We conclude this section with discussing several packages for implicit parallelization. These packages mostly provide an apply-type function for performing parallel execution of batch jobs over an existing parallel infrastructure operated by a scheduler (e.g. LSF, SGE, SLURM), or distributed system (e.g. docker Swarm). The package rslurm allows a seamless operation with SLURM-based high-performance cluster, performing transparently the task distribution and results gathering in an asynchronous way. The packages clustermq and batchtools, instead, allow to interact in a common way with a large variety of high-performance schedulers, including SLURM, TORQUE, docker Swarm etc. While clustermq allows a more fine tuning of the system, by manual initialization of the master-worker scheme, the batchtools allow a more transparent and user-friendly way to setup parallel execution and wait for the result reduction.

To summarize the section, we note, that the focus of package developers in the field of parallel and distributed computing has shifted from low-level explicit parallelism to more high-level implicit methods. The recently introduced packages allow to interact with a given infrastructure (including schedulers etc.), rather than to create your own. And currently there are no actively maintained packages for Volunteer Computing infrastructure backends in R language.

3 RBOINC Concept

In this section we discuss the RBOINC extension package concept, that allows to utilize the computational power of idle time of volunteer computers. We note, that BOINC software [And04] is a sophisticated tool for organizing both the Volunteer Computing projects, as well as desktop grids of enterprise level [Iva15].

Following the conclusions in Section 2, we define the following general requirements for the RBOINC package:

- seamless integration of a BOINC project as a desktop grid backend,
- transparent split and merge operations of the parameters array,
- apply-type function for asynchronous, fault-tolerant and reliable obtaining of the result of computations.

We note, that RSLURM package is the package with the most similar functionality and architecture. Inspired by that, we recommend an umbrella BOINC project for running R code to be used as a BOINC backend for the package. We assume, that the essential settings of the project (including the level of redundancy for a workunit, the deadline for computing etc.) will be set up at the workunit level, with a possibility to pass the specific settings to the BOINC server by means of the package functionality.

We assume, that the apply-type function will allow to perform a seamless parameter space split, workunit generation and computations initialization. Similarly to RSLURM package, we assume, that the results of computations will be obtained asynchronously, with a possibility to obtain intermediate (incomplete) results, as well as to monitor the current progress level. The architecture of the concept is presented on Fig. 1.

We note some technical difficulties that the package might have to deal with. First, the R software should be installed on the volunteer's host. In this regard, we will consider the virtual machines creation, or, alternatively, the portable R software setup. Fault tolerance issues should be taken into account, however, BOINC platform has internal methods to deal with faults, deadline violations and malicious activity. Note also, that a BOINC application should use the checkpoints mechanism to suspend the activity when the volunteer's host activity increases over the given threshold. The checkpointing mechanism should essentially be implemented. We also assume to utilize the built-in multicore features provided by the parallel package.

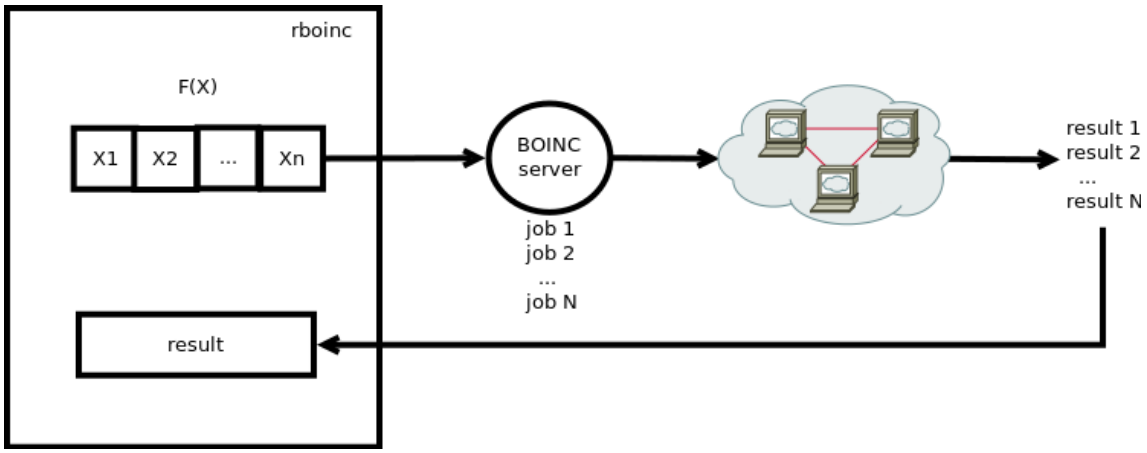

Figure 1: Architecture concept of the RBOINC package.

4 Gene Network Expansion

In this section we consider the task of gene network expansion has addressed by the gene@home project and how the project would have benefited and could benefit in the future from the availability of the `RBOINC` package.

The biological knowledge about the regulation of genes is in general incomplete. Without the details of the whole set of events that connects the transcription abundance of two genes, a gene regulatory network represents by means of connections (facilitatory or inhibitory) the causal relationships between genes transcription level. Their correct description allows either predicting the behaviour or manipulating the system. Given an incomplete gene regulatory network the task of Gene Network Expansion (GNE in the following) is to provide a list of other genes (possibly equipped with their connections with the genes of the input network) whose order should reflect, for each gene, the confidence of being actually part of the correct gene regulatory network. The task emerges from the observation that, in life science practice the biologist has, or experimentally gain, notion of what could be some of the genes that are relevant for the process under study. Expanding this initial knowledge with other genes can suggest relevant genes and pathways for further scrutiny and testing, leading to an increase of the biological knowledge.

In order to address the GNE task a Trento-based group of researchers (including two of the authors) devised, with the help of several collaborators, the gene@home project (`http://gene.disi.unitn.it`) running on the BOINC platform. In particular the platform hosted the implementation of an algorithm called Network Expansion by Stratified Subsetting and Ranking Aggregation (NES^{2}RA) [AMC$^+$16] that is based on the stratified subsetting of the variables that are fed to the PC-algorithm [SG91] (PC in the following). PC is named after the first names of the authors who proposed it, and it infers causal relationships between variables by means of Conditional Independence (CI) tests and its application permits to discover direct causal relationships between correlated variables using observational data. PC has been applied to a wide range of domains including Yeast gene expression data [M$^+$10] and QSAR/QSPR analyses [SGLP16] So far the gene@home project has systematically applied NES^{2}RA to *Escherichia coli* and *Vitis vinifera* data.

The gene@home developers have done many activities to setup the application. At the very beginning they tested the PC algorithm whose 'skeleton' part is used in the iterated version of the PC (called PC-IM) aimed to GNE and distributed via BOINC, see Fig. 2. NES^{2}RA can be considered a parameter sweep of the PC-IM and it is computationally demanding, for example, it requires ∼260K PC-algorithm runs for expanding a network of *Vitis vinifera* (∼28K genes). The tests of the PC algorithm were done using the implementation available in the R pcalg package (Methods for Graphical Models and Causal Inference) that proved to be impractical for the use planned by the researchers. The computational power required for processing several loosely-coupled parallel subtasks coming from NES^{2}RA, led them to rewrote the 'skeleton' procedure in C++ and to use BOINC with a very fast application written with the BOINC API. They also improved the code with the removal of recursion. The benchmarks showed that the speedup they achieved was ∼240x, also reducing memory requirements (see [ASM$^+$15]). They also planned to 'push back' their code into R, that would have given to the R users community a much more efficient version of the original algorithm, however the activity is currently postponed to the near future.

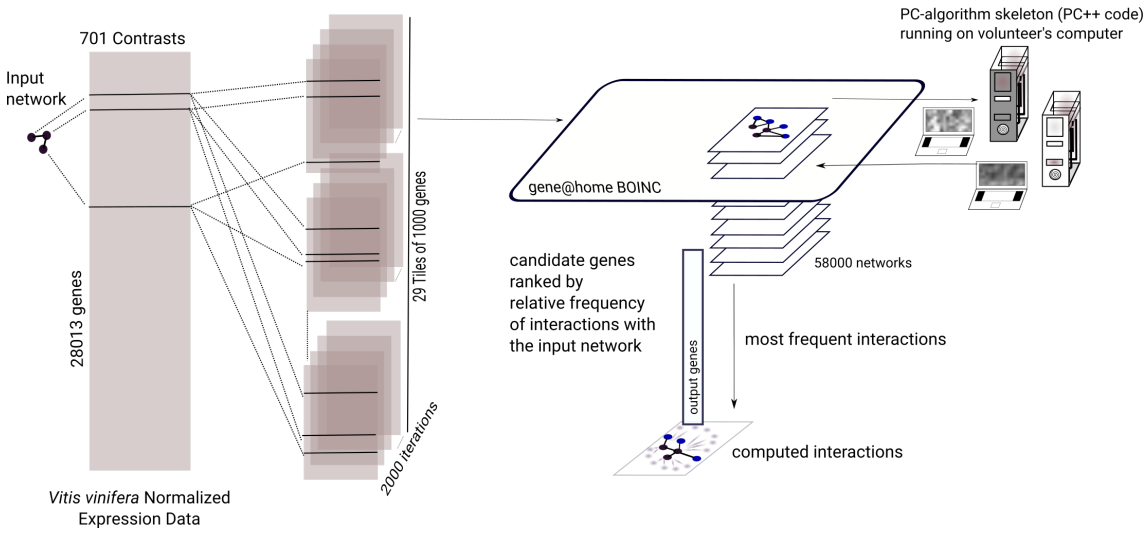

Figure 2: Design of the PC-IM procedure on the BOINC platform as performed by the gene@home project. The data is randomly divided in "tiles" that includes also the genes of the input networks. The PC-algorithm skeleton procedure runs on the BOINC clients, the results are post processed obtaining a genes list that expand the network.

Was the `RBOINC` package available to the gene@home team at the very beginning, the effort to set-up and evaluate the approach would have been considerably smaller. Moreover RBOINC would have given a convenient way to test the competitors, for example the network inference methods in R packages, using BOINC without the need of developing specific BOINC applications. The need to rewrite the PC-algorithm would have been impossible to avoid but the new version would be now integrated in R for general use. Finally, the general availability of NES^{2}RA for other users would be now rather straightforward within the R environment.

When the `RBOINC` package will be ready we plan to use it for improving several aspects of the gene@home project. The main benefit is that our approach to GNE will be available in an integrated analysis environment. This circumstance will permit us to use BOINC for tasks, like ranking aggregation in NES^{2}RA or postprocessing or even data subsets generation, that now run just on servers and, in some cases, have proved to be the bottleneck of the application. As mentioned above the issue of the availability would be solved and the general user would also have the possibility to integrate the GNE results with comparisons and tests in a familiar and flexible environment.

5 Conclusions and Future Work

The paper presented the concept of `RBOINC` package whose goal is to allow for the deployment of computations over desktop grids running under the BOINC platform. This functionality should be seamlessly available within the R environment. We reviewed the related R packages and sketched the main requirements of the new package. Moreover we considered the potential impact over the project gene@home devoted to gene network expansion. Gene@home is a BOINC project that implements a specific data analysis procedure, more in general the `RBOINC` package will permit the run of several different potentially-demanding analysis on the distributed volunteer grid. Future work is required to actually develop and implement the package. Nowadays with the growing quantity of data produced, the `RBOINC` package can play a key role not only for the solution of the GNE task but also to tackle several other problems that requires intense computation on extensive resources.

Acknowledgements

The work of AR is partially supported by RFBR, projects 15-07-02341, 15-07-02354, 15-29-07974, 16-07-00622, and by President RF's grant No.MK-1641.2017.1.

References

[AMC+16] Francesco Asnicar, Luca Masera, Emanuela Coller, Caterina Gallo, Nadir Sella, Thomas Tolio, Paolo Morettin, Luca Erculiani, Francesca Galante, Stanislau Semeniuta, Giulia Malacarne, Kristof Engelen, Andrea Argentini, Valter Cavecchia, Claudio Moser, and Enrico Blanzieri. NES^{2}RA: Network expansion by stratified variable subsetting and ranking aggregation. *The International Journal of High Performance Computing Applications*, 0(0):1094342016662508, 2016.

[And04] David P. Anderson. Boinc: A system for public-resource computing and storage. In *5th IEEE/ACM International Workshop on Grid Computing*, pages 4–10, 2004.

[ASM+15] Francesco Asnicar, Nadir Sella, Luca Masera, Paolo Morettin, Thomas Tolio, Stanislau Semeniuta, Claudio Moser, Enrico Blanzieri, and Valter Cavecchia. TN-Grid and gene@ home project: Volunteer computing for bioinformatics. In *Second international conference BOINC-based high performance computing: Fundamental research and development (BOINC: FAST 2015)*, number 1502, pages 1–15, 2015.

[Iva15] Evgeny Ivashko. Enterprise desktop grids. In *Proceedings of the Second International Conference BOINC-based High Performance Computing: Fundamental Research and Development (BOINC:FAST 2015)*, pages 16–21. CEUR Workshop Proceedings, Vol-1502, 2015.

[M+10] M. H. Maathuis et al. Predicting causal effects in large-scale systems from observational data. *Nat. Methods*, 7(4):247–248, Apr 2010.

[R C17] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017.

[SG91] P. Spirtes and C. Glymour. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9:62–72, 1991.

[SGLP16] Natalia Sizochenko, Agnieszka Gajewicz, Jerzy Leszczynski, and Tomasz Puzyn. Causation or only correlation? application of causal inference graphs for evaluating causality in nano-QSAR models. *Nanoscale*, 8(13):7203–7208, 2016.