Supporting parallel R code in clinical trials: a grid-based approach

Dennis Wegener
Fraunhofer IAIS
Schloss Birlinghoven
53754 St. Augustin, Germany
dennis.wegener@iais.fraunhofer.de

Thierry Sengstag
Swiss Institute of Bioinformatics
Bâtiment Génopode
CH-1015 Lausanne, Switzerland
thierry.sengstag@isb-sib.ch

Stelios Sfakianakis
Biomedical Informatics Laboratory
Institute of Computer Science
FORTH, Greece
ssfak@ics.forth.gr

Stefan Rüping
Fraunhofer IAIS
Schloss Birlinghoven
53754 St. Augustin, Germany
stefan.rueping@iais.fraunhofer.de

Abstract

In this paper, we describe an extension to the ACGT GridR environment which allows the parallelization of loops in R scripts in view of their distributed execution on a computational grid. The ACGT GridR service is extended by a component that uses a set of pre-processor-like directives to organize and distribute calculations. The use of parallelization directives as special R comments provides users with the potential to accelerate lengthy calculations with changes to preexisting code.

The GridR service and its extension are developed as components of the ACGT platform, one aim of which is to facilitate the data mining of clinical trials involving large datasets. In ACGT, GridR scripts are executed in the framework of a specifically developed workflow environment, which is also briefly outlined in the present article.

1. Introduction

With the accelerating development of high-throughput technologies in the domain of biomedical research and of their use in the context of clinical trials, hospitals and clinical research centers are facing new needs in terms of data storage and analysis.

For instance the microarray analysis of a tumor biopsy of a single patient provides 10'000s to 100'000s of gene-expression values summarizing up-to millions of microarray features. New technologies based on imaging, genome sequencing and proteomics are pushing even further the needs for data processing in the clinical research.

These new sources of data about patients and diseases are used in conjunction with classical clinical information, such as age, gender, status of various biochemical markers, pathological classification of tissues, etc.

The analysis of such complex data sets require appropriate data exploitation approaches, integrating the know-how acquired in many independent fields into a powerful environment that physicians can easily and safely use, for the benefit of the patients. One of the goals of the European project ACGT (Advancing Clinico-Genomics Trials on Cancer) is to address this issue [1].

Several initiatives with a similar goal have been started worldwide, among which NCI's caBIG (Cancer Biomedical Informatics Grid) [2] in the USA and CancerGrid in the UK [3]. ACGT differs from these projects in that in addition to the purely technological aspects of the project, a strong emphasis is put on the compliance of the IT infrastructure to ethical and legal guidelines, which should increase its potential for acceptation by the clinical community.

From the data processing viewpoint, the ACGT project aims at providing an IT infrastructure supporting the management of clinical trials (e.g. patient follow-up), as well as the data-mining involved in the translational research that often occurs in parallel. It is in relation to the latter aspect of the project that the need for a high-performance environment supporting the R language [4] and the vast collection of already existing biostatistics algorithms was recognized.

In a previous contribution [5] we have introduced GridR, showing how R can be used on a grid. The present paper describes how this approach can be further developed by introducing grid-based parallelism in the solution of highly demanding computational problems.

As our parallel version of GridR is meant to address concrete data mining issues occurring in clinical trials, we also introduce the ACGT data architecture in which it is embedded.

2. ACGT

The project ACGT aims at addressing the needs of the biomedical community by providing a secured, integrated data management and data mining environment in support of large multi-centric clinical trials.

From the technological point of view, ACGT offers a modular environment in which new data processing and data mining services can be integrated as plug-ins as they become available. ACGT also provides a framework for semantic integration of data sources (e.g., clinical databases) and data mining tools, through the use of a specifically developed ontology and of a semantic mediator.

In the version to be released to the public in early 2009, the various elements of the data mining environment can be integrated into complex analysis pipelines through the ACGT workflow editor and enactor, itself embedded in a user-friendly portal.

In terms of the technology infrastructure the ACGT platform is based on three state-of-the-art middleware technologies: The *Grid*, which takes care of the user management, the management of Virtual Organizations (VOs), the security infrastructure, the data management, and the efficient utilization of the available computing power, the *Service Oriented Architecture* and its infrastructure, which prescribes the needed Web Service interfaces for the annotation, invocation, and composition of the ACGT components as services, the *Semantic Web*, which provides the "glue" in various places such as for the semantic annotation of data and services and the linking to shared ontologies.

In this context a set of ACGT services have been developed which can be roughly classified as follows:

- Data access services. These services are responsible for the retrieval of data shared in the context of a clinical trial. This category includes the Data Wrappers which are adapters for existing clinical and imaging databases exposing database contents to other ACGT components, Microarray services that provide access to BASE repositories [6], and finally Mediator Services that offer uniform access to distributed and heterogeneous clinical and biomedical databases.
- Services for the Semantics-aware use of the platform. In this category, the Ontology Services provide a conceptualization of the domain, by the mean of the Master Ontology for Cancer, for constructing complex queries for the mediator services. Furthermore Metadata Repositories and associated services ensure the persistence and proper management of the description of the services available to the users.
- Service Enactment, which includes the basic grid mechanisms used for the submission and execution of jobs in processing nodes, and the higher-level Workflow Enactment Services that support the management and execution of complex biomedical workflows.
- Data Analysis and Knowledge Discovery Services, which are a number of data mining and knowledge discovery tools and services that fulfill the data-analysis requirements of ACGT, with GridR as one of the most prominent tools.

ACGT aims at reusing existing open-source tools as much as possible; R and the associated project Bioconductor are thus natural candidates for integration in the ACGT environment. It is in this context that GridR was developed.

In ACGT GridR plays a dual role: on one hand it can be used interactively, giving the users access to the whole ACGT environment, on the other hand it is deployed as a data-analysis service exposing a Web Service interface for the execution of scripts incorporated in scientific workflows. Its design and internals are further described in the following section.

3. GridR

3.1. GridR in ACGT

GridR [5] consists of an R package and a web service which allow using the statistical software R [2] in a grid environment.

In the ACGT platform, the GridR service is implemented as a GSI-secured grid service based on the Globus Toolkit 4 libraries [7] and on the Gridge Toolkit [8], mainly using the DMS (data management system) and GRMS (grid resource management system) components of the grid substrate. The interface of the service supports the execution of user-defined scripts as well as the execution of scripts that had been pre-registered in a repository. In the latter case, scripts can be referred to by using their unique repository id.

3.2. Input and output

The interface to the DMS is based on files; this implies that all input and output data have to be passed to and from GridR by physical files. For this purpose, the GridR service attaches a header to each script which makes the contents of input files accessible in the R session on the execution machine as elements of a predefined R list. The interface for the output is a list of file or directory names that the user can use to export data from the session.

3.3. GridR and parallel processing

The motivation for the parallelization of R code is that a large set of advanced biostatistics tasks are computationally very intensive but have a structure which is trivially parallelizable (e.g. Monte Carlo or resampling algorithms). This means that there are elements of calculations in R scripts that can be run independently of each other. In order to avoid making workflow management too complex on the ACGT platform, it was decided to hide the parallelization from the workflow environment and to deal with it directly at the level of the GridR environment. This is also justified by the fact that developers of R scripts are ultimately best placed to know which parts of the script can benefit from parallelization.

Hiding parallelism from the workflow environment also ensures that all GridR tasks are executed in a consistent environment, namely with the same scheduler and data management.

The developer of a parallel GridR script is offered a "directive"-like mechanism for the annotation of the parts of the script that can run in parallel. Currently only a simple parallelization of for-loops is supported, requiring the exact number of iterations to be known before execution. This number, which represents the degree of parallelism, is used to spawn a corresponding number of GridR jobs, each executing a single iteration of the loop.

Internally, the GridR preprocessor parses the script, extracts the annotation information, and guides the submission of the specified number of run-time jobs by splitting the original code in an adequate number of smaller scripts, each containing the code of the body of the loop.

Hence, the R script is no longer computed as a whole, but as elements of a pipeline in which the parallel sections are preceded and followed by sequential start- and end-sections dealing with the DMS interfaces. Internally the interfaces between the script sections are again file-based, each section receiving a section header and/or a section footer for interfacing with the other parts of the script. In greater detail, the interfaces between the R-script sections are:

- for non-parallel sections
  - Header - loads the full workspace saved by the previous non-parallel section as well as the output object stored by the directly previous parallel sections
  - Footer - saves the full workspace for the following non-parallel section as well as the objects needed as input by the directly following parallel sections.
- for parallel sections
  - Header - loads the objects stored by the previous non-parallel section. In addition the index variable specifying the iteration number is passed, letting the user to take iteration-specific actions.
  - Footer - saves the output objects for the following non-parallel section.

As the creation of the job descriptions for the GRMS has to be done before the actual R sessions are started, and as the parallelization has to be transparent at the workflow level, the information required for the parallel execution of individual iterations is prepared by parsing the script in a preprocessing phase. This information is made of the specification for the inputs (including functions if they are user defined) and outputs of the parallel section, and of the index variable. In addition the degree of parallelization, i.e. the number of parallel tasks, and two pointers marking where the parallel section of the code starts and ends are also determined during the preprocessing phase. In order to avoid having a different code version for standalone and GridR parallel execution of the script, the directives needed to parse the R code and make a parallelized version of it are passed to GridR as R comments.

The GridR service thus translates the parallelization information into a job description for the GRMS containing detailed information about which tasks depend one others, which files have to be staged in and out, etc. This mechanism is illustrated with the following example:

```
#GRIDR-PARALLEL-START; index=i;
    degree=3; input=result1;
    skipNextLines=1
for (i in 1:3) {
     result2[i] = add(result1,i)
}
#GRIDR-PARALLEL-END; output=result2;
    skipPrevLines=1
```

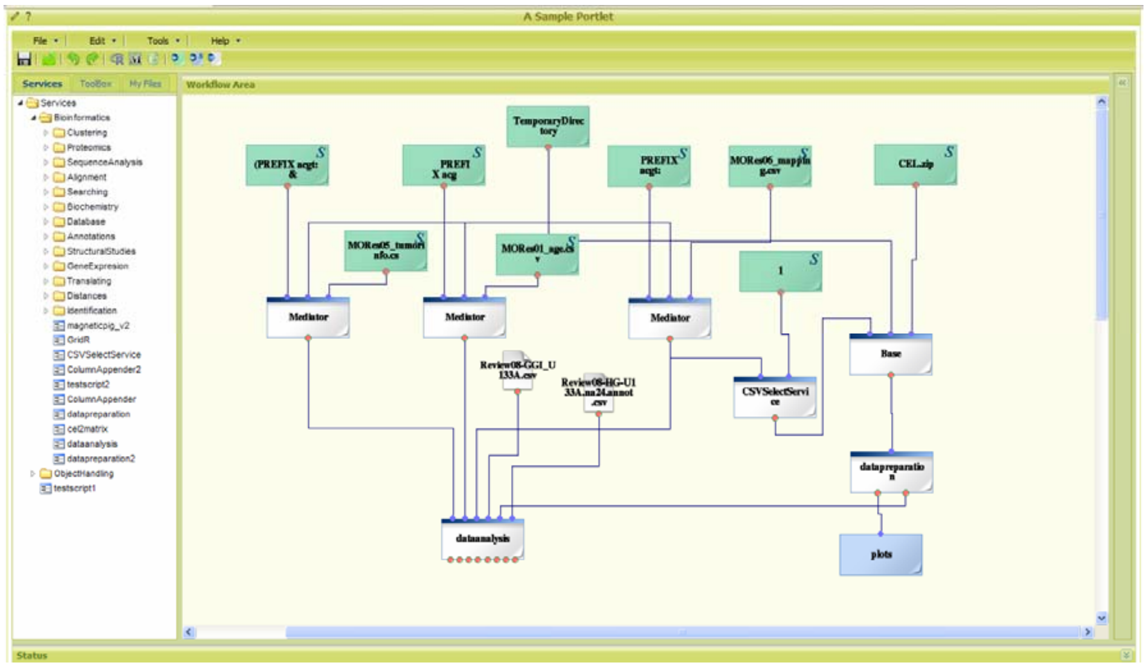

Fig. 1: A complex genomics data analysis workflow, represented in side the ACGT workflow editor. The tree explorer on the left of the picture allows the user to select services registered in the metadata repository to add them to the workflow. See text for the description of the workflow.

For data-transfer performance reasons, all non-parallel parts of the script are executed on the same machine, which means that only the subsets of R objects required to perform the parallel sections have to actually be transferred to compute nodes. The list of objects to transfer is given as parameters to the "input" field of the "#GRIDR-PARALLEL-START" directive (see example). A similar mechanism is used to retrieve data after execution.

The technique of parallelization described here is very simple, which allowed a rapid integration of the method into the ACGT environment. Thus the ACGT environment readily provides a way to distribute parallel tasks over the whole grid environment with only minimal editing of pre-existing code, the edition being limited to the addition of a few comment-like parallelization directives.

4. Scenario

A natural application of this mechanism in clinico-genomics context follows from the fact that many biomedical quantities follow ill-defined statistical distributions making the need to use empirical determination methods for summaries and confidence intervals a necessity. Resampling techniques play thus a central role in the mining of biomedical data, though this usually implies computationally intensive calculations. Fortunately, given their structure, such methodologies can easily benefit from the use of parallelization.

In the scenario retained for the illustration of parallel execution of R code, GridR is used to identify a genomic signature associated to the response of breast-cancer patients to a specific treatment. The signature is then used as a predictor for patient response to the treatment. (A gene signature is a group of genes, to which a numerical summary score is associated, usually a weighted average of the gene expression of the genes belonging to the signature.)

This scenario follows the line of the clinical trial on breast cancer "TOP", which is currently run in the context of the ACGT project.

4.1. Simulated data set

The data set used in the present scenario is a simulation of the actual TOP trial, with 198 "simulated patients", which was constructed using only public or simulated data.

Each simulated patient was associated to an Affymetrix gene-expression microarray (HG-U133A) from a preexisting breast-cancer dataset available on the Gene Expression Omnibus [9] (GEO series GSE7390). Each microarray measures the expression of 22283 genes (or transcripts). A simulated pathological response was computed for each simulated patient based on the gene expression pattern, using a known group of genes. Many of the known genes should be retrieved after reconstruction of the signature, thus providing a validation criterion.

4.2. Actual scenario

The scenario is to construct the signature discriminating between patients having good and poor response based on the gene expression, then, using the signature score to predict patient response. The assessment of the accuracy of the classifier is made using a cross-validation loop. Essentially, the workflow implementing this scenario contains of the following components:

- Microarray data retrieval from a database
- Normalization of microarray data (GridR script)
- Retrieval of patient pathological response from clinical database.
- Determination of pathological response signature and assessment of classification performance with a cross-validation loop (parallel processing with GridR)

Data exchange between these components occurs through DMS-stored files. Fig. 1 illustrates the implementation of this workflow as constructed with the ACGT workflow editor. In the figure the green blocks represent input fields, which are related to the preparation of queries to clinical and microarray databases. White blocks in the middle of the workflow are responsible for actually enacting the queries ("Mediator"), while the two bottommost white blocks are the GridR elements, with the one on the right ("datapreparation") implementing the data normalization step and the one one the left ("dataanalysis") implementing the gene signature discovery and the associated cross-validation loop.

The classifier performance assessment uses a 10-fold cross-validation loop distributed to the compute nodes by the GridR parallel service. In each of these loops a signature is constructed using the genes discriminating best between responsive and non-responsive patients. Significant genes are identified via a logistic regression of pathological response (coded as 0 and 1) against gene expression. Tens of thousands of logistic regressions are thus required in each iteration.

This scenario is a proof-of-concept for the support for parallel processing of R code with GridR used in biomedical context. The same scenario in a real clinical application would require much heavier calculations, as, for example, the threshold for gene selection would be determined iteratively and a bootstrapping loop would enclose the cross-validation loop to mitigate sampling effects in the assessment of classifier performance. This further justifies the need for supporting parallel execution in the context of GridR.

Benchmarking the method described in this paper still remains to be done. Clearly as in all parallelization problems the largest expected benefit will be obtained by minimizing the time spend in data transfer with regards to the time spent in calculations. We do not expect a great benefit for the present exercise. However the benefit for larger scale, more realistic, biomedical applications should be significant, despite the use of a file based approach to information exchange.

5. Related Work

The approach we have followed in the present work is an example of data parallelism: the tasks submitted for concurrent execution on the grid are identical but they apply to different slices of the data. In particular we are parallelizing loops so that all the iterations are executed in parallel in different grid jobs. The parallelization of loops has been used extensively in various programming languages and toolkits, e.g., in High Performance Fortran [10], Fortress [11], and in NESL [12] or Data Parallel Haskell [13] using parallel arrays and list comprehensions.

The marking of R code sections to be run in parallel in our work is similar to the approach in OpenMP [14]. A current limitation in our implementation is that the exact number of parallel jobs has to be known in advance. Such limitations seem not to exist in the "parallel-R" (pR) approach [15]. In pR an "on the fly" parallelization of R code is taking place and the parallel tasks are executed through the means of MPI in a cluster of machines.

A number of other tools providing support for concurrent computations exist in R, e.g., rpvm [16], rmpi [17] and snow (Simple Network Of Workstations) [18]. Rpvm and rmpi provide wrappers to the parallel programming packages parallel virtual machine (PVM) [19] and message-passing interface (MPI) [20] which can only be used in homogeneous environments and require explicit orchestration of message passing in the parallel execution of R scripts. The snow package provides a higher level of abstraction that is independent of the communication technology.

However, in contrast to GridR, all these approaches lack a seamless integration with grid technology, especially considering the security requirements which are essential when dealing with real clinical data.

With respect to the ACGT Workflow Environment, workflows are enacted by a BPEL [21]] compliant workflow engine. BPEL version 2.0 also supports parallelism to a certain degree: it features a specific language construct ("flow") to support the execution of the contained activities in parallel, and, similarly, a parallel version of the "ForEach" loop supports parallelism in the context of iterations. Therefore GridR parallelism could be achieved at the level of the workflow layer, although this would require either some custom handling of parallel GridR scripts or having the user to manually separate the single script into multiple sections that are then executed as autonomous activities in the workflow. The applicability of such mechanisms to the ACGT platform remains to be investigated further.

6. Future Work and Conclusion

A current limitation of our solution is that the degree of parallelization has to be known in advance, because the jobs for parallel processing are not submitted at runtime of the R script itself. Alternative approaches to set up parallel GridR jobs are under consideration, for instance by making use of the GridR client within the script [22]. The latter allows submitting sub-tasks from within the running R session, thus avoiding the constraint of knowing the number of iteration before execution. However, other practical constraints appear on the infrastructure side (e.g., firewall configuration), and the script code has to be modified.

The approach presented here was pursued as it met the security constraints of present ACGT environment (GT4 machines and Condor pools). Moreover, a request from the clinical user community was to be also able to test the R code in standalone fashion on a local machine without modification of the code, thus allowing easy moves from development (standalone) to production (parallel) versions of analysis scripts.

The parallel version of GridR presented in this contribution addresses the practical needs of the ACGT biomedical community. However, we believe that, despite its current limitations, it can be helpful to a broader R audience, as it brings the power and security features of grid infrastructure to R developers, at an extremely minimal cost in terms of script adaptation.

7. Acknowledgments

The authors gratefully acknowledge the financial support of the European Commission for the Project ACGT, FP6/2004/IST-026996.

8. References

[1] ACGT (EU): http://eu-acgt.org/

[2] Cancer Biomedical Informatics Grid, caBIG (USA): https://cabig.nci.nih.gov/

[3] CancerGrid (UK): http://www.cancergrid.org/

[4] R Development Core Team (2005), “R: A Language and Environment for Statistical Computing”, R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0 (http://www.r-project.org/)

[5] D. Wegener, T. Sengstag, S. Sfakianakis, S. Rüping, A. Assi, “GridR: An R-based grid-enabled tool for data analysis in ACGT clinico-genomic trials”. In: Proceedings of the 3rd IEEE International Conference on e-Science and Grid Computing (eScience 2007), Bangalore, India, 2007, pp. 228-235.

[6] BASE: http://base.thep.lu.se/

[7] I. Foster, “Globus Toolkit Version 4: Software for Service-Oriented Systems”, IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2006

[8] J. Pukacki, M. Kosiedowski, R. Mikołajczak, M. Adamski, P. Grabowski, M. Jankowski, M. Kupczyk, C. Mazurek, N. Meyer, J. Nabrzyski, T. Piontek, M. Russell, M. Stroiński, M. Wolski “Programming Grid Applications with Gridge”, Computational Methods in Science and Technology vol. 12, Poznan 2006.

[9] GEO: http://www.ncbi.nlm.nih.gov/projects/geo/

[10] High Performance Fortran Forum. *High Performance Fortran Language Specification*, May 1993.

[11] G. Steele, “Parallel Programming and Parallel Abstractions in Fortress”, Functional and Logic Programming, 8th International Symposium (FLOPS 2006), Proceedings, LNCS, Vol. 3945(1), 2006

[12] G.E. Blelloch,, Programming parallel algorithms, *Communications of the ACM*, Vol. 39(3), pp 85--97, 1996, ACM New York, NY, USA

[13] M.M.T. Chakravarty, G. Keller, R. Lechtchinsky, W. Pfannenstiel, “Nepal -- Nested Data-Parallelism in Haskell”, In R. Sakellariou, J. Keane, J.R. Gurd, and L. Freeman, editors, Euro-Par 2001: Parallel Processing, 7th International Euro-Par Conference, Springer-Verlag, LNCS 2150, pp524-534, 2001.

[14] R. Chandra, R. Menon, L. Dagum, D. Kohr, D. Maydan, J. McDonald, „Parallel Programming in OpenMP“. Morgan Kaufmann, 2000.

[15] X. Ma, J. Li, N.F. Samatova, “Automatic Parallelization of Scripting Languages: Toward Transparent Desktop Parallel Computing”, *IEEE International Parallel and Distributed Processing Symposium, 2007..*

[16] rpvm: R interface to PVM: http://cran.r-project.org/src/contrib/Descriptions/rpvm.html

[17] H. Yu. Rmpi package for R. http://www.stats.uwo.ca/faculty/yu/Rmpi/

[18] A. Rossini, L. Tierney, and N. Li. “Simple parallel statistical computing”. UW Biostatistics working paper series, 2003

[19] PVM: http://www.csm.ornl.gov/pvm/pvm_home.html

[20] MPI Forum: http://www.mpi-forum.org

[21] A. Alves et al., “Web Services Business Process Execution Language Version 2.0”, OASIS Standard 11 April 2007, available from http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[22] D. Wegener, D. Hecker, C. Körner, M. May, M. Mock, “Parallelization of R-programs with GridR in a GPS-trajectory mining application“, submitted to UKD08@ECML/PKDD08