Integrating Scientific Workflows with Scientific Gateways: A Bioinformatics Experiment in the Brazilian National High-Performance Computing Network

Maria Luiza Mondelli[1], Marcelo Monteiro Galheigo[1], Vívian Medeiros[1], Bruno F. Bastos[1], Antônio Tadeu Azevedo Gomes[1], Marta Mattoso[2], Ana Tereza R. Vasconcelos[1], Luiz M. R. Gadelha Jr.[1]

[1]Laboratório Nacional de Computação Científica (LNCC)
Petrópolis – RJ – Brazil

[2]Programa de Engenharia de Sistemas e Computação (PESC/COPPE)
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brazil

{mluiza,galheigo,vivian,bfbastos,atagomes,atrv,lgadelha}@lncc.br

marta@cos.ufrj.br

Abstract. *Bioinformatics experiments are rapidly and constantly evolving due improvements in sequencing technologies. These experiments usually demand high performance computation and produce huge quantities of data. They also require different programs to be executed in a certain order, allowing the experiments to be modeled as workflows. However, users do not always have the infrastructure needed to perform these experiments. Our contribution is the integration of scientific workflow management systems and grid-enabled scientific gateways, providing the user with a transparent way to run these workflows in geographically distributed computing resources. The availability of the workflow through the gateway allows for a better usability of these experiments.*

1. Introduction

Biology is generating data sets that are increasingly large due, for instance, to high-throughput sequencing technologies. Various computational tools can be composed to perform complex analyses of these sequences in areas such as metagenomics, transcriptomics and comparative genomics. To better manage these analyses, scientists can use scientific workflow management systems [Liu et al. 2015] (SWfMS), which allow them for easily composing and executing many-task computations through the automation of the steps commonly involved: scheduling activity execution based on data dependencies; parallel execution of independent activities; synchronization of workflow execution; managing data transfer and activity execution in distributed computing environments. In addition to these steps, it is also interesting to gather provenance information [Freire et al. 2008] describing the history of composition and execution of computational processes, such as scientific workflows. This allows, for instance, determining how data sets were derived from other data sets and computational activities. They allow for the precise description of how a computational process was planned, which is called *prospective provenance*, and what occurred during execution, which is called *retrospective provenance*. Applications of provenance information include reproducibility and validation of computational

scientific experiments, scientific workflow reuse, data quality evaluation and attribution of scientific results.

Parallel and distributed computing is frequently used for scaling up the execution of computationally and data-intensive scientific workflows. These computations often involve the collaboration of a large number of geographically distributed individuals and heterogeneous resources. Grid computing [Foster 2001] was proposed to allow for managing users and computational resources in these distributed computing environments that often include multiple institutions. Finally, scientific gateways [Wilkins-Diehr et al. 2008] consolidate this functionality to provide grid-enabled scientific applications through web interfaces. In this work, we describe an experiment of implementing a scientific gateway to provide pre-existing Bioinformatics workflows in the Brazilian National High-Performance Computing Network (SINAPAD). It allows users to execute these scientific workflows using computational resources geographically distributed through an easy-to-use web interface.

This work is organized as follows: in section 2, we describe how SciCumulus and Swift were integrated to the grid middleware [Lima et al. 2005] and scientific gateway toolkit [Gomes et al. 2015] of SINAPAD. In section 3, we evaluate this integration by experimenting with two pre-exisiting scientific workflows, SwiftGecko [Mondelli et al. 2015] for comparative genomics, and SciPhy [Ocana et al. 2011] for phylogenetics. In the same section we present a performance evaluation of SwiftGecko. In section 4, we compare the integration to related efforts in this area. Finally, in section 5, we describe future work and make some concluding remarks.

2. Integrating Scientific Workflows with Grid Middleware

SINAPAD uses the CSGrid [Lima et al. 2005] middleware to support the usage and management of distributed computational resources. It offers features to integrate resources and applications as well as to manage data and users with a very rich and flexible authentication and authorization scheme. Through the system, users have access to a workspace in which they can find and execute all available applications as well as add new applications. On top of this infrastructure, CSGrid offers two key entry points for users: a Java desktop client and a service bus based on CORBA (called OpenBus) through which job (OpenDreams) and file (HDataService) management services are provided as secure APIs. To facilitate the access to these CSGrid services through the Web, the mc2 toolset [Gomes et al. 2015] has been developed for supporting the implementation of web-based science gateways and Webservice/REST-based APIs.

For the integration of scientific workflows and CSGrid, two parallel SWfMSs were used: Swift [Wilde et al. 2011] and SciCumulus [de Oliveira et al. 2010]. Swift supports the specification, execution and analysis of scientific workflows through a high-level scripting language with many characteristics commonly found in functional programming. Swift has the capability to evaluate and execute in parallel the expressions whose data dependencies are met. It also has support for gathering and querying for core provenance information [Gadelha et al. 2012], related to datasets and computational activities managed during the workflow execution, and additional information that scientists are usually interested in querying, such as domain-specific annotations. The provenance is stored and can be accessed through a relational database (SQLite3 or PostgreSQL).

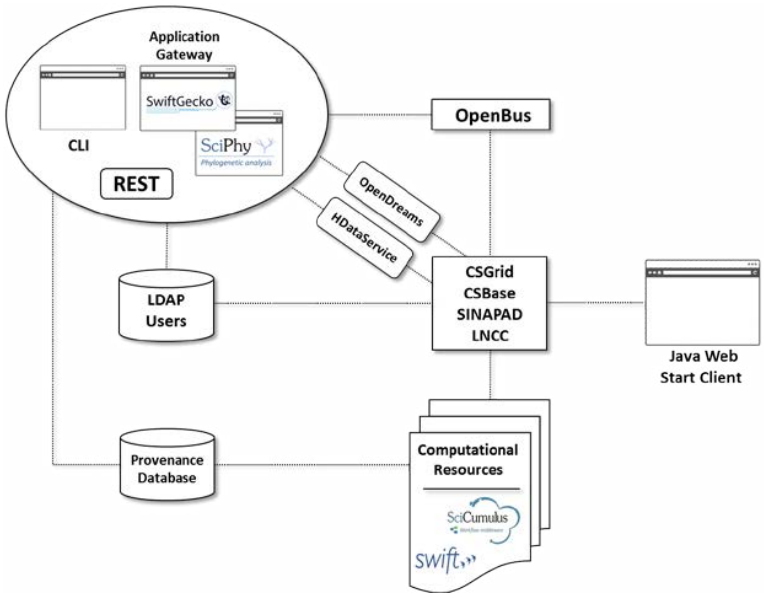

Figure 1. Conceptual schema of the integration between scientific workflows and CSGrid middleware

SciCumulus also allows scientific workflow modeling, execution and analysis. It was first developed as a cloud middleware to explore parameter sweep and data fragmentation parallelism in scientific workflow activities. Currently, SciCumulus supports the parallel execution in both clusters and cloud enviroments. The activities and the data flow between them are defined by the user through a file in XML format. SciCumulus uses an algebraic approach, focused on data, to the execution of workflows. This approach makes the provenance, which is stored using a relational database (PostgreSQL), a key point both for the execution and for the analysis of the workflow.

In order to deploy workflows in SINAPAD, as depicted in Figure 1, we first installed and configured SciCumulus, Swift and the applications used by the workflows on the computational resources managed by CSGrid. Through the CSGrid platform, we created two *algorithms* for executing these two workflows. "Algorithm" is a term used in CSGrid for describing a non-interactive application that can be dispatched directly by CSGrid clients with a clearly-defined set of input and output parameters. A CSGrid algorithm comprises an XML-based configuration file that defines these parameters and one or more submission scripts that deal with the specifics of the involved applications. For instance, in the case of the Swift we created scripts to configure the environment to allow the SWfMS to be executed from a submitting node in the HPC clusters managed by CSGrid. For Scicumulus, because of execution peculiarities and common allocation policy settings in HPC clusters available in SINAPAD, it was necessary to use a network proxy to allow SciCumulus access to an external provenance database.

3. Evaluation

For the evaluation, we will present two workflows, SciPhy [Ocana et al. 2011] and SwiftGecko [Mondelli et al. 2015], that were deployed in SINAPAD.

In bioinformatics, phylogeny is an area that process biological sequences with the

objective of obtaining phylogenetic trees and other statistical calculations that support inferences about the evolutionary life and relationships of organisms of interest. Designing, executing and analyzing phylogenetic (or phylogenomic) experiments is not an easy task since they are complex and execute time/CPU-intensive applications. Then, they require technologies such as SWfMSs and HPC environments. To support these scenarios, the scientific workflow SciPhy for phylogenetic analyses was modeled using SciCumulus. SciPhy consume amino acid or nucleotide datasets and generate phylogenetic trees. It presents the following four main activities (Figure 2):

1. Multiple Sequence Alignment (MSA): To construct individual MSA. This activity receives files in the multi-FASTA format as input and it produces an MSA as output.
2. MSA Conversion: converts the output from the activity 1 to PHYLIP format.
3. Search for the best evolutionary model: It tests the PHYLIP files to find the best evolutionary model.
4. Construction of phylogenetic trees: it receives the outputs from activities 2 and 3 and constructs phylogenetic trees.

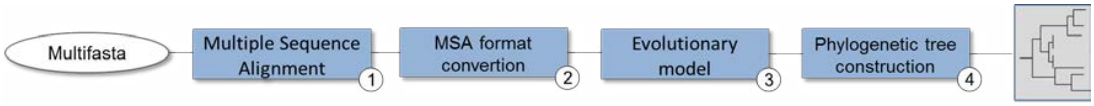

Figure 2. Conceptual view of the workflow SciPhy

Comparative genomics is an example of another area of bioinformatics in which experiments also aim to infer relationships between organisms. They are considered large-scale bioinformatics workflows as they require HPC to process the huge amount of data, known as biological big data. Addressing these data-intensive problems provides measures to perform, for instance, several analysis with the ultimate objective to better understand the biology of the organisms. The GECKO [Torreno and Trelles 2015] application is an example of workflow for genome comparisons designed to identify collections of high-scoring segment pairs (HSPs). To take benefit from distribution and paralelism, this application was implemented in Swift as the SwiftGecko[1] workflow, which is composed by three main computational modules, as presented in Figure 3 and described as follows:

1. Dictionary Calculation: Correspond to the creation of dictionaries for each sequence, composed by the activities 1-4 in blue boxes.
2. HSP Detection: Detection of the seed points or hits used to identify the HSP locations. It is composed by the activities 5-9 in green boxes.
3. Post-processing Module: to attach complementary tasks such as for performing statistical calculations or functional annotations, composed by the activity 10 in orange box.

In Figure 4 we show the user web interface of the SwiftGecko algorithm, obtained as a result of the integration. In this interface, the user must enter the required parameters to execute the SwiftGecko workflow without the need to install any program related to the workflow or the SWfMS that will manage its execution. The CSGrid middleware along

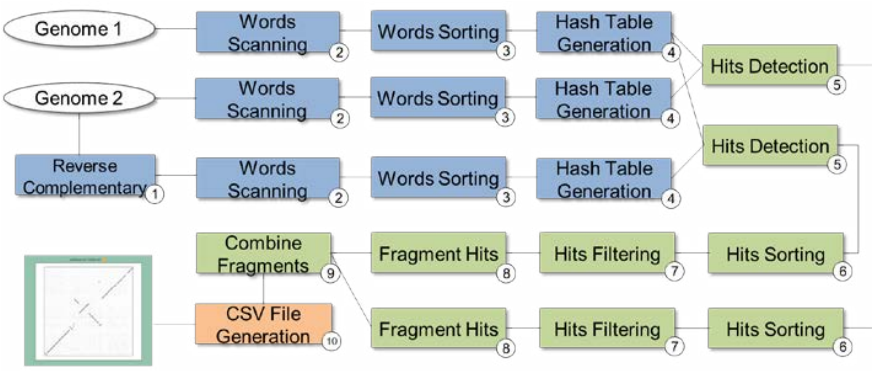

Figure 3. Conceptual view of the workflow SwiftGecko

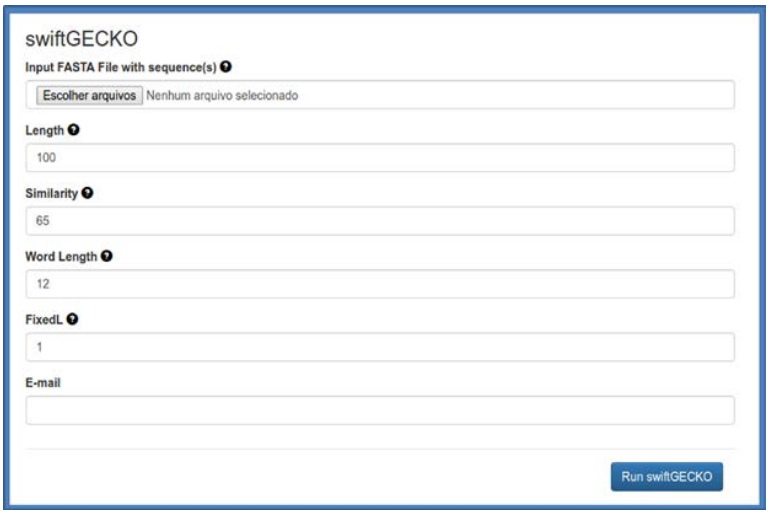

Figure 4. Example of interface for workflow execution

with the execution environment of each SWfMS are responsible for the distribution of jobs through the SINAPAD network.

SwiftGecko was used as a case example in order to evaluate the performance of the integration. The workflow was executed with five bacterial complete genomes as input, resulting in 135 task executions. The executions were performed in a 72-node cluster with 16GB of RAM and 8 computing cores per node. In this evaluation, presented in Figure 5, we measure the performance (total execution time) of SwiftGecko for serial and parallel executions directly in the cluster and parallel through CSGrid, also using the same cluster. Results present performance improvements of up to 47% in the parallel execution time when compared to the sequential execution, which drops from around 8 minutes (using a single core) to around 4 minutes using 8 cores. The execution through CSGrid presents improvements of approximately 23% when compared to the sequential execution, dropping to 6 minutes. It is worth mentioning that the SwiftGecko applications use an out-of-core strategy and are I/O intensive. Therefore, scalability could be improved by using higher throughput storage systems or more efficient data management

[1] Available at `https://github.com/mmondelli/swift-gecko`

strategies. Regarding the execution of the workflow through the portal, in addition to the submission and execution times, the file transfer time between CSGrid and the cluster is also taken into account. For this reason, we can understand the difference between the parallel executions in the cluster and through the portal.

Figure 5. Evaluation of the execution times of SwiftGecko workflow

4. Related Work

There are some approaches in the literature that present mechanisms to execute large-scale scientific workflows. Globus Genomics [Madduri et al. 2014] presents a system for rapid analysis of large quantities of next-generaton sequencing (NGS) genomic data. It integrates Galaxy WfMS for the specification of a data analysis workflow, and Globus tools for data movement and access-control management. However, the work is focused only in the cloud infrastructure, using Amazon services to storage data and to execute the data analysis pipeline.

Askalon [Nadeem et al. 2007] proposes an application development and computing environment to provide an invisible Grid to the application developers. It uses the Unified Modelling Language (UML) diagrams to represent the workflow graphically. It also provides an XML-based representation, to translate the UML diagram and pass it to the runtime system to be scheduled and executed. Although the application involves the concepts of workflows and Grids, instead of integrating existent systems, they built a single system from scratch.

5. Conclusion

Providing scientific workflows through an application management system that operates on a computational Grid environment as CSGrid, allows users to have access to more resources to execute their experiments. This is an important aspect for users who do not have the computing infrastructure needed to perform their executions. In this work, we presented an approach to integrate SWfMSs with grid-enabled scientific gateways, providing a way to run through an interface workflows that previously needed to be run via command line in the cluster. Thus, since the workflow is available through CSGrid, the execution process becomes a transparent task for the user. We evaluate a bioinformatics

workflow for genome comparisons, which have demonstrated satisfactory performance when executed through this infrastructure. As future work, we intend to support the provenance that is already captured by SWfMSs [de Oliveira et al. 2010] [Gadelha et al. 2011] to allow the analysis of information that is generated through the workflow execution.

In addition, we aim at integrating the SWfMS functionality with grid middleware in a native way, through a prototype of a SWfMS with support to CSGrid developed in SINAPAD called OSC Workflow Manager (GWO). This prototype uses a Open Scientific Connectors (OSC) language to describe workflows [Medeiros and Gomes 2012] in which the main feature consists in the interaction of tasks mediated by connectors. Tasks and connectors interfaces are called, respectively, from ports and roles. A workflow described in OSC is composed by linking tasks ports and connectors roles. Tasks, connectors, ports and roles can be associated with properties that allow its parameterization.

OSC has basic types, used to associate abstract workflows concepts with its concrete implementation. A task can be, for instance, an executable program or an encapsulated workflow and a connector may represent a file transfer or a character pipe. The language also allows types of non-functional attributes to be combined with basic types, such as task parallelism, fault tolerance and provenance tracking [Medeiros and Gomes 2013]. This combination is defined by the workflow developer, that may decide with great flexibility in which parts the treatment is performed.

The OSC language was developed based on the architectural description language known as Acme [Garlan et al. 2010]. Therefore, workflows described in OSC may be graphically modeled by using AcmeStudio tool. The executable tasks of the workflow defined during its modeling correspond to the CSGrid algorithms. After modeling, the workflow can be stored in the GWO workflow repository and then executed, so that the workflow model is transparent to the end user. Thus, to execute the workflow through the GWO, the user must define the input and output parameters required for its execution. The GWO is in charge of managing the execution through CSGrid, including parallel execution of independent tasks. It also takes responsibility for the processing of non-functional attributes included in the model by the workflow developer. In this way, the various features that are available externally to SINAPAD infrastructure will be available natively. This will allow for a better integration and control of the workflows within the plataform.

References

de Oliveira, D., Ogasawara, E., Baião, F., and Mattoso, M. (2010). SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 378–385. IEEE.

Foster, I. (2001). The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222.

Freire, J., Koop, D., Santos, E., and Silva, C. (2008). Provenance for Computational Tasks: A Survey. *Computing in Science & Engineering*, 10(3):11–21.

Gadelha, L. M. R., Wilde, M., Mattoso, M., and Foster, I. (2011). Exploring provenance in high performance scientific computing. In *Proc. of the 1st Annual Workshop on High*

Performance Computing meets Databases - HPCDB '11, pages 17–20. ACM Press.

Gadelha, L. M. R., Wilde, M., Mattoso, M., and Foster, I. (2012). MTCProv: a practical provenance query framework for many-task scientific computing. *Distributed and Parallel Databases*, 30(5-6):351–370.

Garlan, D., Monroe, R., and Wile, D. (2010). Acme. In *CASCON First Decade High Impact Papers on - CASCON '10*, pages 159–173. ACM Press.

Gomes, A. T. A., Bastos, B. F., Medeiros, V., and Moreira, V. M. (2015). Experiences of the Brazilian national high-performance computing network on the rapid prototyping of science gateways. *Concurrency and Computation: Practice and Experience*, 27(2):271–289.

Lima, M. J. d., Melcop, T., Cerqueira, R., Cassino, C., Silvestre, B., Nery, M., and Ururahy, C. (2005). CSGrid: um sistema para integração de aplicações em grades computacionais. In *Salão de Ferramentas do XXIII SBRC. Anais do XXIII SBRC*, pages 1207–1214.

Liu, J., Pacitti, E., Valduriez, P., and Mattoso, M. (2015). A Survey of Data-Intensive Scientific Workflow Management. *Journal of Grid Computing*, 13(4):457–493.

Madduri, R. K., Sulakhe, D., Lacinski, L., Liu, B., Rodriguez, A., Chard, K., Dave, U. J., and Foster, I. T. (2014). Experiences building Globus Genomics: a next-generation sequencing analysis service using Galaxy, Globus, and Amazon Web Services. *Concurrency and Computation: Practice and Experience*, 26(13):2266–2279.

Medeiros, V. and Gomes, A. T. A. (2012). Towards Fully Configurable Support to Non-Functional Attributes in Scientific Workflows. In *IEEE eScience Early Results and Works-in-Progress Poster Papers*, pages 2–3.

Medeiros, V. and Gomes, A. T. A. (2013). Expressando Atributos Não-Funcionais em Workflows Científicos. In *Proc. of VII Brazilian e-Science Workshop*.

Mondelli, M. L., Torreño, O., Ocaña, K. A. C. S., Mattoso, M., Wilde, M., Vasconcellos, A. T., Trelles, O., and Gadelha, L. M. R. (2015). SwiftGECKO: a provenance-enabled parallel comparative genomics workflow. In *Proceedings X-Meeting 2015*, page 268.

Nadeem, F., Nerieri, F., Podlipnig, S., Qin, J., Siddiqui, M., Truong, H.-l., and Villazon, A. (2007). ASKALON : A Development and Grid Workflows. In *Workflows for e-Science*, pages 450–471. Springer.

Ocana, K. A., Oliveira, D. d., Ogasawara, E., Davila, A. M., Lima, A. A., and Mattoso, M. (2011). SciPhy: A Cloud-Based Workflow for Phylogenetic Analysis of Drug Targets in Protozoan Genomes. In *Advances in Bioinformatics and Computational Biology - 6th Brazilian Symposium on Bioinformatics, BSB 2011. Proceedings*, pages 66–70.

Torreno, O. and Trelles, O. (2015). Breaking the computational barriers of pairwise genome comparison. *BMC bioinformatics*, 16(1):250.

Wilde, M., Hategan, M., Wozniak, J. M., Clifford, B., Katz, D. S., and Foster, I. (2011). Swift: A language for distributed parallel scripting. *Parallel Computing*, 37(9):633–652.

Wilkins-Diehr, N., Gannon, D., Klimeck, G., Oster, S., and Pamidighantam, S. (2008). TeraGrid Science Gateways and Their Impact on Science. *Computer*, 41(11):32–41.