demos:02

HPSW-Prof: A Provenance-Based Framework for Profiling High Performance Scientific Workflows

Maria Luiza Mondelli[1], Matheus Tonelli de Souza[2], Kary Ocaña[1], Ana Tereza R. Vasconcelos[1], Luiz M. R. Gadelha Jr[1]

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

[2]Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (Cefet/RJ)
Petrópolis – RJ – Brazil

{mluiza,matheust,karyann,atrv,lgadelha}@lncc.br

Abstract. *Scientific experiments usually demand high performance computing (HPC) and involve the execution of a flow of activities, so they can be modeled as scientific workflows. Scientific Workflow Management Systems (SWfMS) provide ways of defining and executing these experiments in HPC environments and they produce detailed information about the workflow composition and execution. However, analyzing this information is not always trivial. This paper presents a profiling framework called HPSW-Prof[1] that aims to provide the user with a set of features for the statistical treatment and manipulation of provenance information obtained from scientific experiments executed with SWfMS Swift. Through the HPSW-Prof, data analysis can become a transparent process since it also offers a visualization layer that supports users for better accessing and manipulating their results.*

1. Introduction

Scientific experiments in large scale usually involve complex and data-intensive simulations, demanding high-performance computing (HPC). These experiments comprise the execution of a flow of several programs or applications and can be modeled as scientific workflows. Scientific Workflow Management Systems (SWfMS) support the management of these experiments, providing ways to define, execute and analyze scientific workflows. Some of the SWfMSs can manage parallel and distributed execution in HPC environments such as clusters, grids or clouds and have the capability of gathering provenance data [Freire et al. 2008]. Provenance can be categorized in: *prospective* provenance, that describes how a computational process was planned and *retrospective* provenance, that defines what occurred during the execution. Provenance supports experiment cycle of life in its three phases: composition (conception, reuse), execution (distribution, monitoring), and analysis (query discovery, visualization), as detailed in [Mattoso et al. 2010]. SWfMS as Swift [Wilde et al. 2011] has these capabilities with focus on HPC. It uses a high-level parallel scripting language for describing the workflow and stores the provenance on a local relational database (SQLite or PostgreSQL).

The use of analysis tools for profiling programs and applications is a key point to better understand their computational behavior. Profilers

[1]Demo available at: https://youtu.be/ltPu7WvM4xk

[Lecomber and Wohlschlegel 2013] are widely used in parallel programming, providing measures and reports related to the performance of an application, such as frequency and duration of function calls. Profilers are used to evaluate and identify critical sections and bottlenecks of code that can be optmized. With regard to the SWfMS, this type of analysis has recently been exploited, since the retrospective provenance captured by these systems sometimes records computational information related to the workflow execution. However, profiling analysis based on provenance data is still a challenge. Depending on a particular SWfMS, scientists would need to instrument provenance through relational databases or log files which is not a trivial task. By accessing the data provenance information in a friendly way, it is possible to better understand the behavior (computational or domain specific) of experiments.

This work aims to present aspects of coupling profiling analysis to SWfMS. The profiling framework HPSW-Prof was designed to provide analyses related with the provenance data of an experiment, modeled and executed with Swift. This work is organized as follows: section 2 presents a background and the features of HPSW-Prof. Section 3 presents an evaluation using a scientific workflow for phylogenetic analysis. Section 4 discusses related works and section 5 concludes this work and describes future works.

2. HPSW-Prof

We present a background related to Swift in section 2.1, since the profiler was designed to access and query its provenance system. Section 2.2 describes the development process and structure of the profiling framework.

2.1. Swift parallel scripting language

Swift [Wilde et al. 2011] is a SWfMS that allows the specification, execution and analysis of scientific workflows through a high-level scripting language with a functional syntax. Swift is designed to compose application programs into parallel applications that can be executed in HPC environments. Users need to specify the activities, inputs and outputs and the data flow between activities. Swift will execute in parallel all the expressions in the script whose data dependencies are met.

Swift also supports the gathering and querying of provenance data [Gadelha et al. 2012] generated during the workflow execution. The provenance is stored in log files that records issues of parallelism, distribution, input/output files and other computational and statistical reports about the execution. Once the workflow is executed, it is possible to import all this information into a local relational database by using the *swiftlog* feature provided by Swift. The conceptual schema stores the provenance in its repository as presented in Figure 1. Its entities are described as follows: *script_run* describes the execution of a workflow; *app_exec* describes the execution of each activity that composes the workflow; *resource_usage* describes computational aspects related to the activities such as CPU, memory or I/O; *file* records the files manipulated during the workflow execution; *stage_in* and *staged_out* lists files consumed and produced by application executions. Additionally, key-value annotations can be added to these entities to describe other aspects of the workflow execution such as domain-specific information. Thus, information related to data derivations and computational statistics are naturally represented in the provenance system and the user can define extractors

in order to relate the domain data with the provenance stored by Swift. To improve usability, this local repository has a database schema that is simpler than the one found in MTCProv [Gadelha et al. 2012]. This schema includes derivation history of in-memory data collections that were manipulated during the Swift workflow executions.

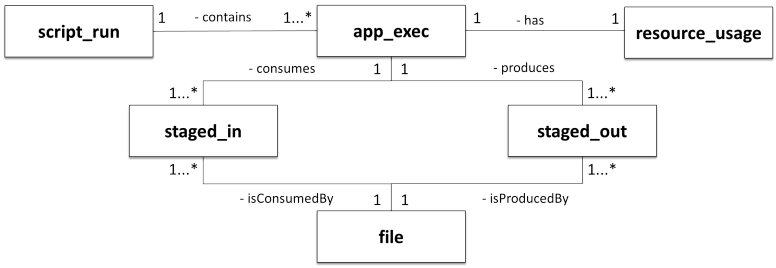


Figure 1. Conceptual schema of the Swift provenance's database

2.2. Profiling framework

With respect to the HPSW-Prof development process, the first step implements a Java web application aiming to allow the upload of local provenance databases into a single PostgreSQL repository. This first component of the HPSW-Prof framework was implemented using JSF (JavaServer Faces), a framework which allows the creation of web applications by using pre-built visual components. We also used an extension of JSF, a library that provides a set of graphic components called PrimeFaces[2]. The web application is connected with the PostgreSQL database, which contains the same schema presented in Figure 1.

The second component comprised the implementation of R functions to connect, manipulate and extract statistics for profiling analysis based on the provenance database. As a result, these functions generate a set of charts with aspects related to the performance of the workflow, i.e., the average execution time and I/O, CPU usage and memory statistics of each workflow activity. This information is important for identifying hot spots on the workflow that can be optimized to obtain better performance. Also, statisical regression analysis was implemented as a function to predict the workflow execution time based on the size of the input files. To visualize these charts, we integrated HPSW-Prof with the Shiny[3] framework, designed to turn R functions into interactive web applications. It allows users to have a more intuitive way to access and analyze their data.

Figure 2 presents the HPSW-Prof architecture. The user specifies and executes the workflow with Swift, which is responsible for managing the execution of its activities in HPC enviroments. After the execution, a set of log files are generated and the provenance must be imported into a local database by running the *swiftlog* feature. The user uploads this database into the repository and can access its analyses through the framework in a transparent and automatized interface. Without the framework, the user would have to run SQL queries to the database, collect and then plot the results. Furthermore, the main ideia of keeping a provenance repository is that many instances of local provenance databases can be uploaded into a single database. This allows, for example, research groups to be

[2]Available at `http://primefaces.org/` Last access: 2016/08/17
[3]Available at `http://shiny.rstudio.com/` Last access: 2016/08/17

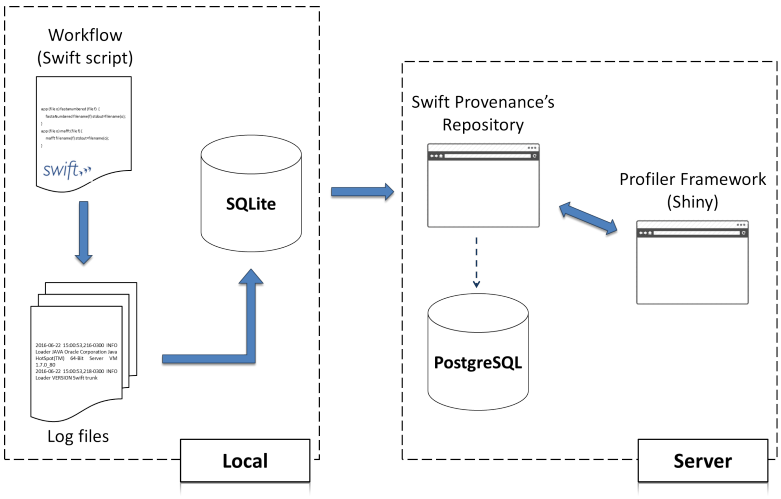


Figure 2. Conceptual view of the HPSW-Prof framework

able to follow up the experiments that are being executed by their researchers. However, the use of HPSW-Prof is not restricted only to a single repository and the framework can be connected to the local database.

3. Evaluation

In bioinformatics, phylogenetic experiments aim to process biological sequences, producing phylogenetic trees or providing statistical calculations to support biological inferences. Due the amount of biological data that need to be processed, these experiments demand HPC and are difficult to manage. The scientific workflow SciPhy [Ocana et al. 2011] was modeled for support phylogenetic analysis experiments. It consists of four main activities, as shown in Figure 3. SwiftPhylo[4] was modeled using Swift and chosen for the HPSW-Prof evaluation.

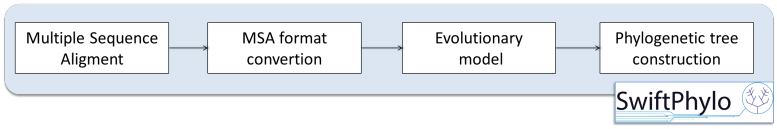
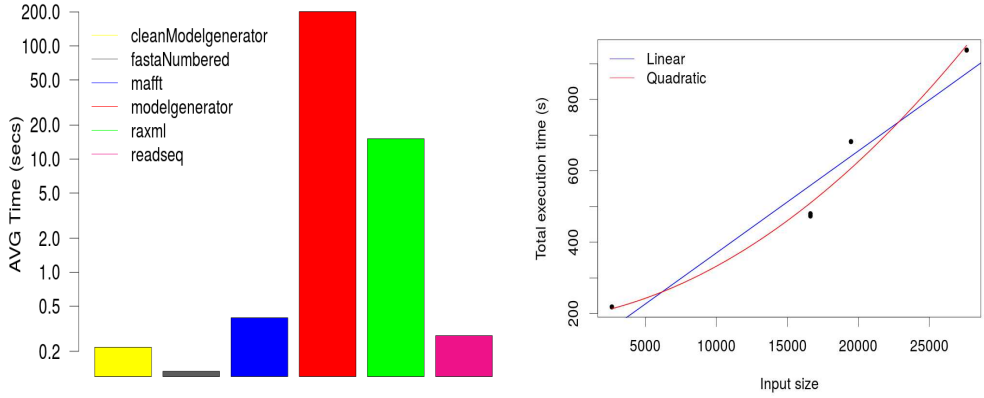


Figure 3. Conceptual view of the workflow SwiftPhylo

After SwiftPhylo execution, the provenance was imported into a SQLite database, through the *swiftlog* feature provided by Swift. The generated database was uploaded to a PostgreSQL repository, using HPSW-Prof. Once the upload is completed, its possible to access the provenance data in the HPSW-Prof web interface and visualize graphics and summaries related to the computational behavior of the workflow execution. The information that was previously available only in the form of relational database can now be queried analyzed through graphics, increasing its usability. Through the interface, the

[4]Available at `https://github.com/mmondelli/swift-phylo` Last access: 2016/08/17

user can choose a particular workflow execution to perform the analysis. Figure 4(a) shows an example of analysis by the average execution time of each activity is calculated. With this information it is possible to identify the activities that influence the total time of the workflow execution, thus being candidates for further optimization attempts. Figure 4(b) presents an analysis that uses statistical regression to predict the workflow execution time based on the size of the input files, obtained for previous executions. HPSW-Prof evaluates the quality of the results by comparing the test and training sets used to perform the statistical regression in a transparent way to the user. Consequently, the framework provides a broader understanding of the characteristics of the experiment.



(a) Avg execution time (apps) (b) Prediction analysis

Figure 4. Examples of analyses available through HPSW-Prof

4. Related Work

WfProf [Juve et al. 2013] is a profiler designed for the SWfMS Pegasus. It has two main components used to wrap the workflow jobs and to record data through a kernel-level tracing as these jobs are executed. Diverting from the kernel-level approach, ParaTrac [Dun et al. 2010] is a profiler for workflows based on user-level file system and process tracing techniques. It explores both data-process interactions and computational characteristics of files, processes and the entire application. In comparison to these approaches, HPSW-Prof takes advantage of the provenance captured with Swift that stores the computational data needed to perform profiling analysis. Additionally, HPSW-Prof provides prediction analysis based on the provenance. The profiling approach in [Souza et al. 2015] presents how querying the provenance database can help the monitoring performance process by using the SWfMS Chiron and the TAU tool for visualization of the analysis. However, this approach aims to detect anomalies in the execution of workflows at runtime.

5. Conclusion

This work presented a framework for profiling scientific workflows executions based on the provenance that is generated by Swift. Therefore, since the workflow is executed and the provenance is imported into a relational database, the framework is responsible for

querying and presenting graphically to the user a set of profiling analyses. We evaluated the profiler using a phylogenetic analysis workflow and presented statistical and performance analyses that are available through HPSW-Prof. As future work, we intend to provide further predictive analyses and analyses found in profiling tools used in parallel programming, such as function call traces. In addition, we aim to add more features to the framework for users adapt their analysis as needed, such as filtering by date of execution or by a specific user who performed the execution. We intend to update the current implementation of the framework for the use of REST services. Also, the source-code for the tools is available through SVN[5] and GitHub[6].

References

Dun, N., Taura, K., and Yonezawa, A. (2010). ParaTrac: a fine-grained profiler for data-intensive workflows. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing - HPDC '10*, page 37, NY, USA.

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. (2012). MTCProv: a practical provenance query framework for many-task scientific computing. *Distributed and Parallel Databases*, 30(5-6):351–370.

Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., and Vahi, K. (2013). Characterizing and profiling scientific workflows. *Future Generation Computer Systems*, 29(3):682–692.

Lecomber, D. and Wohlschlegel, P. (2013). Debugging at Scale with Allinea DDT. In *Tools for High Performance Computing 2012*, pages 3–12. Springer Berlin Heidelberg.

Mattoso, M., Werner, C., Travassos, G. H., Braganholo, V., Ogasawara, E., Oliveira, D., Cruz, S., Martinho, W., and Murta, L. (2010). Towards supporting the life cycle of large scale scientific experiments. *International Journal of Business Process Integration and Management*.

Ocana, K. A., de Oliveira, 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.

Souza, R., Silva, V., Neves, L., Oliveira, D. D., and Mattoso, M. (2015). Monitoramento de Desempenho usando Dados de Proveniência e de Domínio durante a Execução de Aplicações Científicas. In *XIV WPerformance*.

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.

[5]Available at `https://subversion.assembla.com/svn/provweb/` Last access: 2016/08/17

[6]Available at `https://github.com/mmondelli/swift-prof` Last access: 2016/08/17