Challenges in managing implicit and abstract provenance data: experiences with ProvManager

Anderson Marinho, Marta Mattoso, Claudia Werner,
COPPE – Federal University of Rio de Janeiro, Brazil
Vanessa Braganholo and Leonardo Murta
Computing Institute – Fluminense Federal University, Brazil

Abstract

Running scientific workflows in distributed and heterogeneous environments has been motivating the definition of provenance gathering approaches that are loosely coupled to workflow management systems. We have developed a provenance management system named ProvManager to manage provenance data in distributed and heterogeneous environments independent of a specific Scientific Workflow Management System. The experience of using ProvManager in real workflow applications has shown many provenance management issues that are not addressed in current related work. We have faced challenges such as the necessity of dealing with implicit provenance data and the lack of higher provenance abstraction levels. This paper discusses and points to directions towards these challenges, contextualizing them according to our experience in developing ProvManager.

1. Introduction

Provenance provides historical information about data manipulated by a workflow [1]. This historical information expresses how data products were generated, showing their transformation processes from primary input and intermediary data. Managing this kind of information is extremely important since it provides scientists with a variety of data analysis applications. For instance, from provenance information it is possible to verify data quality of generated products, because one can look at the ancestral data to check reliability. Other examples are [2]: possibility to audit trails to verify what resources are being used; data derivation capability; experiment documentation; responsibility attribution; among other applications.

To improve the experiment analysis using provenance information, it has to be modeled, gathered, and stored for further queries. Provenance management is an open issue that is being addressed by several forums worldwide. One of the open problems relates to which provenance data should be gathered and how they can be collected. Provenance gathering becomes more complex when the experiment is executed at distributed and heterogeneous environments, such as clusters, P2P, grids, clouds, and on several different Scientific Workflow Management Systems (SWfMS).

One can foresee several scenarios of experiment execution in a distributed and heterogeneous environment. Often, a single experiment is executed by more than one workflow. The need for breaking a conceptual experiment in two or more separate workflow executions can occur due to several reasons. One important reason is the presence of long manual activities along the experiment. To avoid a long break point in the workflow, it is often broken into two separate workflow executions. Also, part of the experiment may need to be executed in a SWfMS that provides parallelism, while another part of the experiment may need to be executed in a system that supports results visualization. In this case, each SWfMS may manage provenance information in a decentralized and isolated way, meaning that each system considers provenance in a specific granularity, stores the information using a specific model, or even worse, some SWfMS may not provide any support for provenance management at all. In situations like that, experiments would benefit from a homogeneous management of provenance [1].

We developed a provenance management system named ProvManager [3] to address the problem of integrating provenance from different workflows that are a part of a single scientific experiment. For instance, suppose that an experiment workflow is broken into two workflows (workflows#1 and workflow#2) that execute in different machines. When workflow#1 and workflow#2 are executed, ProvManager can relate artifacts from these two executions as if they were one single workflow execution. Even though ProvManager has contributed to managing provenance in distributed and heterogeneous environments, it has also exposed more challenges that still need to be addressed. In fact, these challenges are generic provenance management issues, and not necessarily connected to distributed environment scenarios as the one previously exemplified. The two main challenges we address in this paper are: the necessity of registering implicit data manipulated by the workflow activities and the lack of higher abstraction levels to help scientists to better understand the experiment's provenance data. This paper details these challenges, which are considered as research opportunities to provide better homogeneous provenance management approaches. It is important to note that some queries of the fourth IPAW Provenance Challenge go in the same direction of the two challenges discussed in this paper. This provides additional insights of the relevance of these two challenges.

2. Background

The Open Provenance Model (OPM) [4] defines a generic representation for provenance data. It is an initiative towards a homogeneous model to provide interoperability between workflows from different SWfMS. However, this is only part of the solution. Even if all SWfMS are OPM compliant, there is still the need to gather compatible provenance data and to provide an integrated control of the distributed provenance data with query support. Additionally, OPM only supports retrospective provenance information.

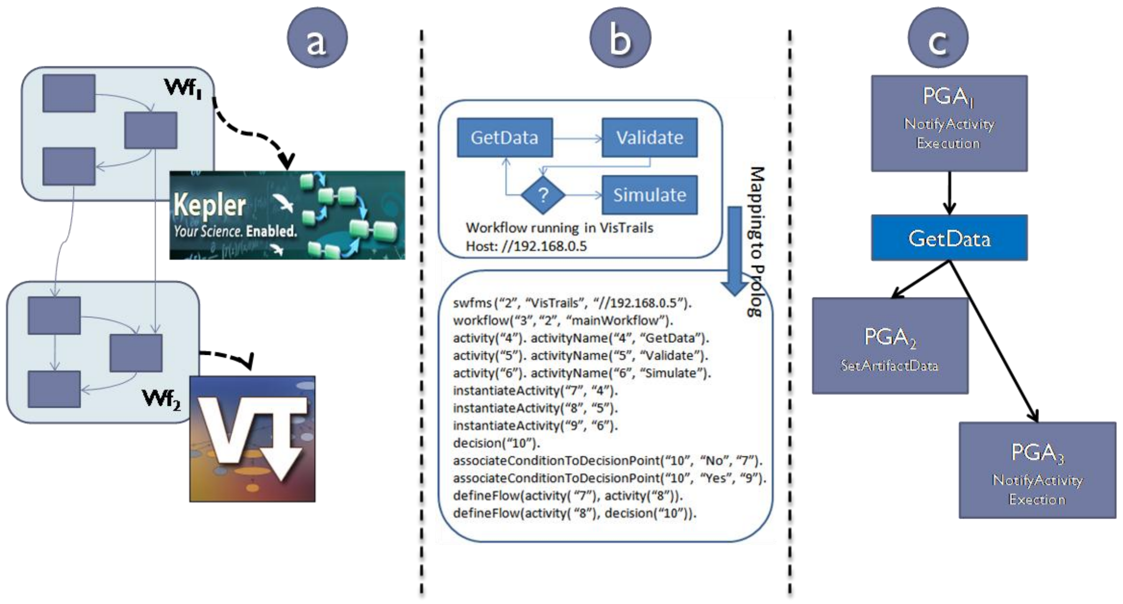
A solution to this heterogeneity and distribution problems is to transfer the responsibility of provenance management to a provenance system that does not depend on the SWfMS. This system would be responsible for allowing the modeling, capturing, storing, and querying of provenance data for the whole experiment. This idea is shared by several works [5-7], but the main difficulty of being SWfMS agnostic is that the SWfMS and the provenance management system need to communicate to exchange information. In order to make this communication possible, some initiatives [5], [8], [9] propose a series of manual activity adaptations over the workflow specification. However this solution introduces overhead to scientists.

In previous work [10], we have claimed that scientists should not have this computational burden. Furthermore, some workflow activities used by scientists are from third parties, which make their adaptation even more complex. In fact, in many cases these activities cannot be altered, at least not in a direct manner. For this reason, we have proposed ProvManager [3], an approach for dealing with integrated provenance management in distributed and heterogeneous environments. ProvManager gathers provenance data independently and allows scientists to focus on the essence of their experiments and make use of the best technologies to enact their workflows. ProvManager is able to transparently gather and store provenance data collected from different SWfMS, translating it to an integrated provenance model that represents the experiment as a whole. As a result, scientists are able to perform provenance queries over the experiment even if it is composed of multiple workflows and enacted on different SWfMS.

ProvManager gathers both prospective and retrospective provenance information. It uses the workflow specification created by the SWfMS as the source of information to extract prospective provenance. The retrospective provenance, based on OPM, is gathered by the workflow activities during the workflow execution. However, the workflow activities need to be adapted to support the provenance gathering mechanism.

ProvManager provides a mechanism to automatically adapt the workflow activities to support provenance gathering. Once a workflow activity is adapted, it is capable of collecting its own provenance information that is generated during the workflow execution. The adaptation is done indirectly by modifying the workflow specification and inserting new workflow activities to collect the provenance information. According to the workflow activity specification (e.g., output and input ports and how they are connected with other activities), Provenance Gathering Activities (PGA) are created to intercept data consumed or produced by each activity port. Additionally, information about the activity execution time is collected by other specific PGA. Finally, the workflow region containing the original workflow activity and the related PGA are "wrapped" into a composite activity in order to maintain the original workflow visual aspect.

Figure 1.a illustrates an experiment example being published in ProvManager. This experiment is segmented in two workflows: one workflow is instantiated in Kepler [11] that invokes parallel execution through Hadoop [12], while the other is instantiated in VisTrails [13], which focuses on visualization of the generated results. Figure 1.b shows the workflow in VisTrails with more details. The fragment is composed of three activities: GetData, Validate, and Simulate, running on a remote host with IP address 192.168.0.5.



```
swfms("2", "VisTrails", "//192.168.0.5").
workflow("3","2", "mainWorkflow").
activity("4"). activityName("4", "GetData").
activity("5"). activityName("5", "Validate").
activity("6"). activityName("6", "Simulate").
instantiateActivity("7", "4").
instantiateActivity("8", "5").
instantiateActivity("9", "6").
decision("10").
associateConditionToDecisionPoint("10", "No", "7").
associateConditionToDecisionPoint("10", "Yes", "9").
defineFlow(activity( "7"), activity("8")).
defineFlow(activity( "8"), decision("10")).
```

Figure 1. Conceptual and concrete activities

To capture provenance data from this experiment, the scientist has to publish it in ProvManager by uploading the workflow specifications (in the case of VisTrails, a .VT file). At this moment, ProvManager instruments the workflow by automatically adding PGA activities that are responsible for capturing and publishing retrospective provenance data in ProvManager during the workflow execution. During the instrumentation, ProvManager captures prospective provenance data from the workflow specification and publishes them in the repository. This repository is a Prolog database, so provenance data are mapped into Prolog predicates. Figure 1.b shows the .VT file mapped into Prolog predicates. Finally, at the end of the instrumentation, a new .VT file is returned to the scientist to be reloaded in VisTrails. This workflow specification is the one that should be executed. Currently, ProvManager can only instrument workflows executed in Kepler and VisTrails. However, the ProvManager architecture was conceived to easily accommodate additional SWfMS.

Figure 1.c illustrates some operational details on how the workflow activity *GetData* is adapted using PGA. Notice that some PGA is placed before the activity execution (PGA_1), and others are placed after it (PGA_2 and PGA_3). This decision depends on the type of provenance that needs to be gathered. For instance, the PGA agents that use the API operation *notifyActivityExecutionStart* have to be executed before the original activity in the sub-workflow. The opposite happens to PGA that uses the API operation *notifyActivityExecutionEnd*.

As discussed before, from the experience of constructing ProvManager, we have faced two main challenges: the necessity of dealing with implicit provenance data and the lack of higher provenance abstraction levels. In the next sections we detail these challenges.

3. Implicit provenance data

The strategy of indirectly adapting workflow activities by inserting PGA is interesting because it allows any workflow activity to support the provenance gathering mechanism. Additionally, it does not affect the workflow's basic structure. Nevertheless, when we tested ProvManager with some real experiments, we realized that this strategy had some limitations. Users claimed that important provenance information was not being collected by ProvManager.

By analyzing this problem, we noticed that ProvManager fails in gathering provenance data when these are not explicitly declared in the workflow specification. This is a problem because there are some types of experiments where the workflow activities are not completely specified in terms of consumed and produced data. In such cases, the activities input and output ports are not declared in the specification, but they do exist in the workflow execution. For example, workflow activities may generate files in a specific directory, and these files may serve as input data to other workflow activities. In many cases, these files (and the directory where they are generated) are not listed as output or input of those activities. In some other cases, they are only partially listed. For example, the directory where the workflow activity is going to read or create files may be specified in the workflow specification, but not the file names. Another common situation is when files do not have specific names, varying according to each workflow execution (e.g., file names contain the work-flow execution number id, execution date, etc.).

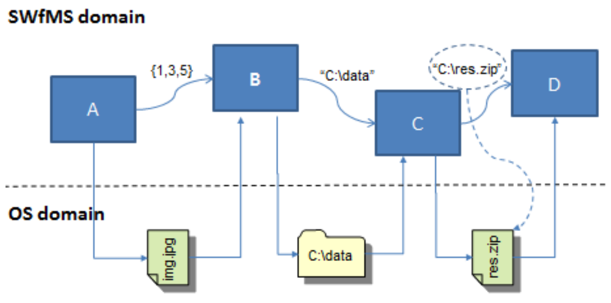


Figure 2. Conceptual and concrete activities

Figure 2 summarizes the scenarios that can occur when data are not explicitly declared in the workflow specification. When this happens, two different data flows occurs in parallel: the first one occurs in the SWfMS domain, representing the data that the workflow system can collect and the other one occurs in the operating system (OS) domain, representing the data that are not collected by the SWfMS. For example, in Figure 2 the workflow activity A transmits data to activity B by means of a shared specific file (img.jpg). The workflow specification, however, does not register this information. Instead, it is only aware that activity A sends an array of numbers to activity B ({1, 3, 5}). Similarly, activities B and C exchange information through files. However, in this case, the workflow specification defines the directory where the files are created (c:\data). This is better than the previous case but the SWfMS still does not know precisely what files are going to be created or used by these activities. Finally, workflow activities C and D illustrate the ideal scenario for provenance management, since the file they use is explicitly defined in the workflow specification (c:\res.zip). However, even in this last case some problems still arise. If the file name does not change at each workflow execution, all the data generated in previous executions will be lost or altered.

This is a typical scenario that happens frequently in the experiment design. However, provenance gathering approaches are currently not prepared to deal with it, including ProvManager. Exceptional cases are provenance gathering approaches that work at OS level [1]. For instance, [14] has defined, inside VisTrails, an infrastructure to collect provenance information that is created in specific directories in the OS. However, this has to be explicitly configured by the scientist by defining the directory or the file path where the information is going to be created. Currently, ProvManager totally depends on the information defined in the workflow specification to create the PGA to intercept all the data consumed and generated by the workflow activities. If there are implicit dataflow in the workflow specification, the provenance gathering mechanism of ProvManager will only register partial provenance information.

4. Lack of higher provenance abstraction levels

In a broad sense, a provenance gathering mechanism can work at three semantic levels [1]: operating system (OS), workflow, and activity.

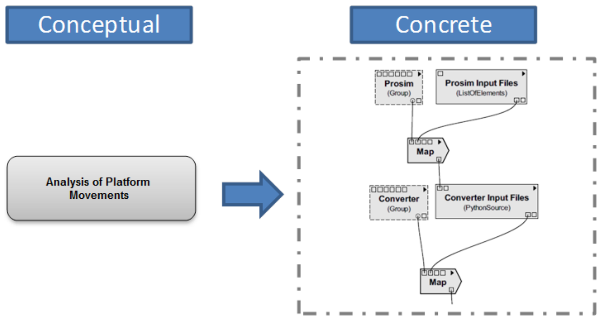
Mechanisms that work at the OS level gather provenance information by using OS tools (*e.g.,* file-system and system call tracer). They are SWfMS independent, but the collected provenance is usually at a fine grain. At the workflow level, a SWfMS is responsible for gathering all the provenance information. One of the advantages is the ease of implementation, but the gathering mechanism is bound to a specific SWfMS and it is difficult to use the same mechanism in other SWfMS. Finally, at the activity level, as previously discussed, each workflow activity is responsible for gathering its own provenance information. One of the advantages is the SWfMS independency, just like in the OS level. However, mechanisms working at this level demand extra effort from scientists to adapt workflow activities.

The key difference of provenance gathering mecha-nisms that work at OS level when compared to the other approaches is the provenance information granularity. At the workflow and activity level, the collected prove-nance information is partially or totally mapped to the information which scientists are used to dealing with (i.e., the information defined in the workflow specification). For example, from these mechanisms, it is possible to know the execution time of the workflow activity “calculate average” or the value generated by it in the output port “average”. This is not the case for mechanisms that gather provenance at OS level. The information collected by these mechanisms is fine grained, and as such, it is not represented in the same abstraction level that has been defined in the workflow specification.

For example, using the system call tracer to track the processes being executed in the OS, the gathering mechanism can register that a process running "matlab.exe" has been executed at a specific time. Furthermore, using the file system, the provenance gathering mechanism detects that a file (containing the average result) has been created by the same process. However, in a typical scenario, scientists may not know exactly which programs are used in the workflows and how they behave in the execution environment. In the aforementioned example, scientists would need to know that MatLab is the program used by the workflow activity "calculate average". The same happens to the file created by MatLab. Scientists would need to know that this file is the generated data from the output port "average" of the workflow activity "calculate average". Without additional info, the provenance data collected at this abstraction level are not really helpful to scientists in the experiment analysis.

Similarly, the same problem happens with higher-level provenance data abstractions. A workflow specification can be represented at least in two different abstraction levels: conceptual and concrete. The first one represents the experiment workflow in a high abstraction level, without concerning about aspects such as methodology, technology, and so forth. The concrete workflow is a specialization of the conceptual workflow which is instantiated in a specific SWfMS. At this level, the aforementioned aspects are defined resulting in a variation of the workflow structure, insertion of new workflow activities and adaptation of existing ones in order to comply with new constraints. Figure 3 presents a fragment of a deep-water oil exploitation workflow in a conceptual and concrete representation [15]. In the left hand side we have a conceptual activity named "Analysis of Platform Movements", which is related to a sequence of concrete activities shown in the right hand side of the Figure. These concrete activities clearly establish one possible way for a particular SWfMS to implement an analysis of platform movements.

Currently, most provenance systems are only concerned with managing provenance data at the abstraction level defined in concrete workflow specifications. However, in the analysis process of their experiments, many scientists need to analyze provenance in higher abstraction levels, and this requires the management of provenance at such level. For example, we can have a conceptual workflow with an activity named "characterize airplane". This activity is implemented in a concrete workflow by four activities: "identify distance", "identify speed", "identify model", and "identify direction". Each concrete workflow activity creates a specific piece of data about the airplane. However, the provenance analysis would benefit from information about the airplane as a whole, related to the conceptual workflow. Some works [16] manage conceptual provenance data but they do not deal with concrete provenance data and not even map these two provenance data abstractions.


Conceptual
Concrete
Analysis of Platform Movements
Prosim (Group)
Prosim Input Files (ListOfElements)
Map
Converter (Group)
Converter Input Files (PythonSource)
Map
Figure 3. Conceptual and concrete activities

5. Conclusions

In this paper, we have discussed two challenges in the scientific workflow provenance management research area. These challenges were evidenced while using ProvManager in the provenance data management of real scientific experiments. The first one regards to the provenance management systems inability of collecting provenance data that has not been explicitly defined in the workflow specification. The second challenge is related to the lack of higher level (abstract) provenance data to help scientists to have a macro experiment perception.

Some solutions can be envisioned to address these challenges. For example, an initial solution to manage implicit provenance data is to adopt a provenance gathering mechanism that works at the OS level. This mechanism would help the provenance system to monitor the creation of provenance data in specific directories. A first step towards addressing the second challenge could be to have the provenance system managing the so-called "conceptual provenance data" via a specific mapping to the existing "concrete provenance data".

Currently, the previously described issues and solutions are being investigated. Additionally, we continue to evolve ProvManager by managing provenance data from real experiments.

Acknowledgements

This work is partially sponsored by CNPq, CAPES and FAPERJ.

References

[1] J. Freire, D. Koop, E. Santos, e C. T. Silva, "Provenance for Computational Tasks: A Survey", *Computing in Science and Engineering, v.10*, n. 3, pp. 11-21, 2008.

[2] Y. L. Simmhan, B. Plale, e D. Gannon, *A Survey of Data Provenance Techniques*. Computer Science Department, Indiana University, 2005.

[3] A. Marinho et al., "Managing Provenance in Scientific Workflows with ProvManager", in *International Workshop on Challenges in e-Science - SBAC*, Petrópolis, RJ - Brazil, 2010, pp. 17-24.

[4] L. Moreau, J. Freire, J. Futrelle, R. McGrath, J. Myers, e P. Paulson, "The Open Provenance Model: An Overview", in *Provenance and Annotation of Data and Processes*, 2008, pp. 323-326.

[5] Y. L. Simmhan, B. Plale, e D. Gannon, "A Framework for Collecting Provenance in Data-Centric Scientific Workflows", in *ICWS*, 2006, pp. 427-436.

[6] P. Groth et al., "An Architecture for Provenance Systems", fev-2006. [Online]. Available: http://eprints.ecs.soton.ac.uk/13216/. [Accessed: 19-jul-2010].

[7] E. Ogasawara et al., "Exploring many task computing in scientific workflows", in *MTAGS 09*, Portland, Oregon, 2009, pp. 1-10.

[8] S. Munroe, S. Miles, L. Moreau, e J. Vázquez-Salceda, "PrIMe: a software engineering methodology for developing provenance-aware applications", in *International Workshop on Software Engineering and Middleware*, Portland, USA, 2006, pp. 39-46.

[9] C. Lin et al., "Service-Oriented Architecture for VIEW: A Visual Scientific Workflow Management System", in *Services*, 2008, pp. 335-342.

[10] A. Marinho et al., "Integrating Provenance Data from Distributed Workflow Systems with ProvManager", in *International Provenance and Annotation Workshop - IPAW*, Troy, NY, USA, 2010, pp. 0-3.

[11] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, e S. Mock, "Kepler: an extensible system for design and execution of scientific workflows", in *Scientific and Statistical Database Management*, Greece, 2004, pp. 423-424.

[12] J. Wang, D. Crawl, e I. Altintas, "Kepler + Hadoop: a general architecture facilitating data-intensive applications in scientific workflow systems", in *Proc. of the 4th Workshop on Workflows in Support of Large-Scale Science*, Portland, Oregon, 2009, pp. 1-8.

[13] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, e H. T. Vo, "VisTrails: visualization meets data management", in *Proc. SIGMOD*, Chicago, Illinois, USA, 2006, pp. 745-747.

[14] D. Koop, E. Santos, B. Bauer, M. Troyer, J. Freire, e C. T. Silva, "Bridging workflow and data provenance using strong links", *Scientific and statistical database management*, p. 397–415, 2010.

[15] W. Martinho et al., "A Conception Process for Abstract Workflows: An Example on Deep Water Oil Exploitation Domain", in *5th IEEE International Conference on e-Science*, Oxford, UK, 2009.

[16] L. Salayandia e P. da Silva, "On the Use of Semantic Abstract Workflows Rooted on Provenance Concepts", *Provenance and Annotation of Data and Processes*, p. 216–220, 2010.