Isolation Levels for Data Sharing in Large-Scale Scientific Workflows

Miguel Liroz-Gistau, Hinde Lilia Bouziane, Esther Pacitti

Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier
LIRMM

Research Report RR-11018

Isolation levels for data sharing in large-scale scientific workflows

Miguel Liroz-Gistau
Miguel.Liroz_Gistau@inria.fr

Hinde Lilia Bouziane
Hinde.Bouziane@lirmm.fr

Esther Pacitti
Esther.Pacitti@lirmm.fr

May 2011

Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier (LIRMM)
UMR 5506 - CC 477
161 rue Ada, 34392 Montpellier, Cedex 5

Isolation levels for data sharing in large-scale scientific workflows

Miguel Liroz-Gistau Hinde Lilia Bouziane Esther Pacitti

Abstract

Scientists can benefit from Grid and Cloud infrastructures to face the increasing need to share scientific data and execute data-intensive workflows at a large scale. However, these workflows are creating more and more challenging problems in the automation of data management during execution. Existing workflow management systems focus on how data is stored, transfered and on data provenance. However they lack in managing isolation during the execution of tasks of the same or different workflows that read/update shared data. In this scope, we propose three isolation levels taking into account data provenance and multiversioning. In the best of our knowledge this is the first proposal in such context.

Keywords: Scientific workflows, Isolation levels, Data sharing, Provenance

Résumé

Les infrastructures, comme le nuage ou la grille, sont de plus en plus sollicitées par les scientifiques pour répondre aux besoins croissants de stockage et de partage de données à grande échelle par les applications d'analyse et de simulation numérique. Dans ce cadre, des recherches intensives sont menées pour étudier des solutions de gestion automatique des données dans ces applications. Les systèmes de workflows, permettant de concevoir et d'exécuter ce type d'application, offrent un support pour le stockage, le transfert et la provenance. Toutefois, dans les systèmes existants, nous constatons l'absence de traitement de l'isolation des tâches au sein d'un ou de plusieurs workflows, qui pendant l'exécution partagent des données en lecture et écriture. Nous proposons trois niveaux d'isolations dédiés aux workflows scientifiques. Ces niveaux prennent en compte la provenance et le multi-versionning pour le stockage et le contrôle de l'évolution des données partagées. A notre connaissance, il n'existe pas encore de solution traitant la même problématique.

Mots-clefs : Workflows scientifiques, Niveaux d´isolation, Partage de données, Provenance

1 Introduction

Scientists increasingly tend to share scientific data and computational services at a large scale [3, 4]. Shared infrastructures like the Cloud or the Grid are more and more requested to publish large amounts of data, reuse data published by other scientists and execute large scale workflows. As an attempt to simplify the usage of these complex infrastructures, many scientific workflow management systems (WFMS) [9] have been proposed. Through such systems, a scientist should be able to simply query data to use, deploy workflows on computational resources for efficient runs, analyze intermediate results or change parameters on the fly, capture necessary meta-data allowing the results to be understood and reproduced, etc. This ideal vision requires a system to be able to automatically manage the execution of workflows, including scheduling, data movement and storage, data sharing, data provenance, fault tolerance, security, interactions with the users, etc. However, to reach this ideal vision, significant challenges still have to be addressed.

For data-centric workflows, intrinsic to scientific applications, data-oriented approaches for execution management are primordial. Current systems focus on how data are transfered, stored, replicated and shared [7, 10]. However, they lack in managing isolation among tasks of the same or different workflows for data sharing.

Isolation management must take into account the past execution history of each involved workflow. Notice that the evolution of shared data provided by different workflow updates requires also the ability to efficiently store and manage different *versions* of the same *logical* data to avoid data redundancy. This situation may correspond to an experiment processing on a given data after a cleaning step, for instance in sequencing applications.

The history of the workflow execution with respect to the tasks dependencies and consumed and produced data, is normally represented by a provenance graph. Therefore, each task or service invocation reading or writing shared data, requires the choice of the appropriate data version taking into account the provenance data. Otherwise workflows may produce unpredictable data. To handle coherent data sharing, we propose specific isolation levels. The originality of our research lies in the usage of data provenance [5] to enforce our proposed isolation levels.

Several efforts have been done in the development of data management services for grids [1] and clouds [10, 4] in the context of scientific workflows. Such services already offer support for storing large volumes of data, sharing at a large scale and for replication management and retrieval. However, current systems have a limited support for multiversion management in provenance and they often focus on read-only data access. For the target isolation purpose, we need to propose new solutions to overcome these limitations. In addition in the best of our knowledge none of the existing solutions addresses isolation as we propose.

Considering the fact that the storage in scientific workflows is often done by means of files, a file versioning system seems to be adequate to manage data versions. The version management (naming, selection for usage) is a low level concern that should be transparent for the users. Shared data may comprise experiment inputs, final results as well as intermediate results, avoiding their recomputation. Finally, the proposed isolation levels aims at enabling the automatic choice of appropriate file versions required by one or multiple concurrent workflow runs.

The rest of this paper is organized as follow: Section 2 describes the workflow model we assume in our work. This model is used in Section 3 to define three isolation levels and illustrate their semantic through examples. Finally, Section 4 discusses our research directions.

2 Workflow model

A workflow can be represented through a directed graph of tasks and dependencies constraining the order of their invocations. There are two classical formalisms for designing workflows: data flows and control flows [9]. In data flows, the dependencies represent the stream of data that takes place between the outputs of one task and the inputs of another. Provided that data flows are specially suited for scientific workflows, which make use of data-intensive applications, we use this formalism in the rest of the paper.

A workflow has a life cycle, from its design to its execution, by which it is represented by three levels of abstraction [9]. The cycle starts with the composition of a workflow template, which simply represents the tasks that have to be executed and the flow of data between them. Then, this workflow has to be parametrized with its input data to be transformed into a *workflow instance*. Finally, the instance is refined to introduce management processes, e.g., data-transfer, and deployed on the target resources. The result is called an *executable workflow*. As mentioned in Section 1, data is stored in files. In a workflow instance (user view), referenced data is associated to logical files. An executable workflow, on the other hand, references file versions storing the specific used data.

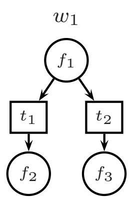
Workflow instance. A workflow instance is defined as a DAG, where nodes are both tasks and files and the connections between them represent data dependencies. Formally, a *workflow instance* is a directed acyclic bipartite graph $w = (T \cup F, D)$ where T is the set of the workflow tasks, F the set of input and output files of the tasks and D the dependencies between files and tasks. In this way, given a file $f \in F$ and a task $t \in T$, $f \to t \in D$ indicates that f is assigned as one the t's inputs and $t \to f \in D$ denotes that f is the file generated for one of t's outputs.

Executable Workflow. As mentioned above, data is associated to files and as the they are updated, new versions are generated. The implication is that a task effectively uses/produces a specific file version. When multiple versions of the same file are stored in the system, a choice needs to be made. This choice is reflected in the executable workflow.

Let $w = (T \cup F, D)$ be a workflow instance. An associated *executable workflow* is defined as the directed acyclic bipartite graph $\textit{exec}(w) = (T \cup FV, D')$, where FV corresponds to the file versions used by the execution and D' is the set of dependencies between file versions and tasks. The assignation must satisfy that **1)** for each file $f \in F$, there exists a file version $f^v \in FV$ and that **2)** for each dependency $f \to t$ or $t \to f$ in D there is a version $f^v \in FV$ such that $f^v \to t \in D'$ or $t \to f^v \in D'$ respectively, i.e., files are replaced by file versions in the dependencies

3 Isolation management

The coexistence of multiple versions of the same data in a shared workflow execution environment may lead to unexpected behaviors. For instance, if a workflow w accesses a file concurrently updated by other workflows, a non controlled access may allow the usage of different versions in a w's run. Such a possibility may produce non coherent results. In order to avoid such a situation, we propose three isolation levels that define the semantics by which data is accessed and modified when executing scientific workflows. These levels are explained through motivating examples.



Storage	Actions *by* w_1	Actions *external*
$f_1^{v_1}$		
$f_1^{v_1}, f_2^{v_1}$	$f_1^{v_1} = \textit{read}(f_1)$ $\textit{exec}(w_1.t_1)$ $\textit{update}(f_2) = f_2^{v_1}$	
$f_1^{v_1}, f_2^{v_1}, f_1^{v_2}$		$\textit{update}(f_1) = f_1^{v_2}$
$f_1^{v_1}, f_2^{v_1}, f_1^{v_2}, f_3^{v_1}$	$f_1^{v_2} = \textit{read}(f_1)$ $\textit{exec}(w_1.t_2)$ $\textit{update}(f_3) = f_3^{v_1}$	

Figure 1. Violation of the 1W-isolation level. On the left, the executed workflow. On the right, the files versions used and produced along the workflow execution (actions).

3.1 One workflow instance isolation

The first level applies to individual workflows instances. For a given instance, the same version of each file should be used consistently throughout its execution. Figure 1 illustrates a scenario which motivates the use of this level. As it can be seen, f_1 is updated between the execution of t_1 and t_2, after what different versions may be used. A reasonable requirement could be that the outputs f_2 and f_3 be produced from the same version of f_1. Nevertheless, without special measures, that condition does not always hold. To avoid similar scenarios, we propose the following rule:

1W-isolation: Let $w = (T \cup F, D)$ be a workflow instance, an execution *exec*$(w) = (T \cup FV, D')$ respects the 1W-isolation level iff for each $f \in F$, the same version $f^v \in FV$ is always used.

3.2 Provenance isolation

Scientific workflows are usually part of series of experiments. Therefore, more advanced semantics may be required in order to preserve the consistency among related executions. We address this issue by imposing some requirements in the creation of related files. To illustrate the problem, let consider the scenario presented in Figure 2. The example includes two workflow instances w_1 and w_2 sharing a same input f_1. w_2 also uses f_2, which is generated by w_1. That situation may correspond to an experiment processing on a given data after a cleaning step, for instance in sequencing applications. The problem arises when a user or another workflow instance updates f_1 between the execution of w_1 and w_2, and w_2 uses the new version $f_1^{v_2}$. In this case, w_2 uses results produced by two different versions of the same data f_1. Although that situation may be admissible in some cases, it can lead to inconsistent results.

To detect situations like the present example, information about the provenance relations between file versions is needed. We capture it within the *file dependency graph*, denoted by $dg(H)$, which consists of a DAG containing all the file version dependencies produced by the execution of a set of workflows $H = \{w_1, ..., w_n\}$. Formally, $f_1^{v_i} \to f_2^{v_j} \in dg(H)$ iff there exists a workflow instance $w \in H$ whose execution *exec*$(w) = (T \cup FV, D')$ contains a task $t \in T$ satisfying $(f_1^{v_i} \to t), (t \to f_2^{v_j}) \in D'$. Now, we can describe the isolation level as follows:

PR-isolation: Let H be a history of workflow instances. $w \in H$ respects the PR-isolation level if on its execution *exec*$(w) = (T \cup FV, D)$, for each $f^v \in FV$, only one version of each file appears on its predecessors in the file dependency graph $dg(H)$.

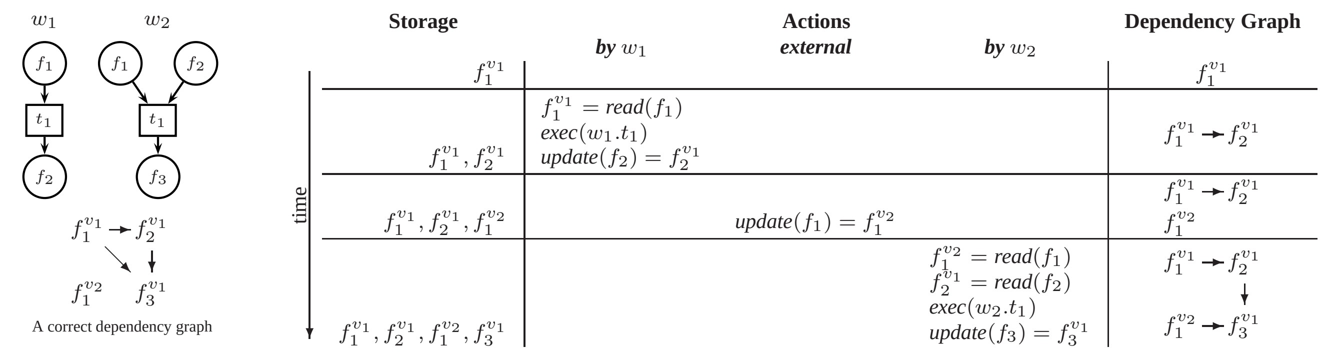

w_1
w_2
f_1
t_1
f_2
f_1
f_2
t_1
f_3
$f_1^{v_1} \rightarrow f_2^{v_1}$
$f_1^{v_2}$
$f_3^{v_1}$
A correct dependency graph
Storage
Actions
Dependency Graph
by w_1
external
by w_2
time
$f_1^{v_1}$
$f_1^{v_1}, f_2^{v_1}$
$f_1^{v_1}, f_2^{v_1}, f_1^{v_2}$
$f_1^{v_1}, f_2^{v_1}, f_1^{v_2}, f_3^{v_1}$
$f_1^{v_1} = read(f_1)$
$exec(w_1.t_1)$
$update(f_2) = f_2^{v_1}$
$update(f_1) = f_1^{v_2}$
$f_1^{v_2} = read(f_1)$
$f_2^{v_1} = read(f_2)$
$exec(w_2.t_1)$
$update(f_3) = f_3^{v_1}$
$f_1^{v_1} \rightarrow f_2^{v_1}$
$f_1^{v_2}$
$f_1^{v_2} \rightarrow f_3^{v_1}$

Figure 2. Violation of the PR-isolation level. On the left, the executed workflow and a correct file dependency graph. On the right, the files versions used and produced along the workflow execution and a corresponding file dependency graph.

3.3 Group isolation

There may be other situations where the previous levels are not enough to ensure consistent executions. To illustrate this, Figure 3 represents a parameter sweep performed by instantiating several times a workflow template with two inputs and one output. The input f_c is a common constant data, while f_{var_i} represents a different parameter used by instance $w_i, i = 1, ..., n$. This is a study of how a parameter (f_{var_i}) variation affects the result for a given value (f_c). The illustrated inconsistency situation appears if f_c is updated during the parameter sweep and before a set of workflow instances are executed ($w_{k+1}, ..., w_n$). Provided that the constant parameter f_c does not actually remain constant during the execution, the usage of the new version $f_c^{v_2}$ produces a non coherent result.

To avoid such a problem, the idea is to enable dependent workflows instances to be join together to satisfy a common consistency condition. For that, we introduce the concept of *workflow group*, allowing a user to define a set of such workflows instances. Finally, we can describe the Group-isolation level:

G-isolation: Let w_i be a workflow belonging to a group $g = \{w_1, ..., w_i, ..., w_n\}$. w_i respects the G-isolation level if the same version of each input/output file is used in the execution of $w_1, ..., w_i, ..., w_n$.

3.4 Discussion

The proposed isolation levels specify the way a WFMS is expected to select files versions to be used by a workflow execution and to ensure the production of consistent data. Even if the three levels are suitable in many situations, their relaxation may be desirable. In fact, their enforcement may require the usage of old file versions, which is not suited if the user is aiming at exploiting fresh data. To overcome this issue, one possibility is to allow the user to specify a limit on the data freshness or to ignore some isolation levels. Then, if the available file versions do not satisfy the user constraint, some of them should be recomputed. Both indications about isolation relaxation and data freshness could be included in metadata, often associated to a workflow instance to specify some execution constraints.

4 Challenges and future work

In this paper we introduced the concept of isolation levels for scientific workflows infrastructures such as clouds and grids, to manage consistently data sharing, taking into account data provenance. In the best

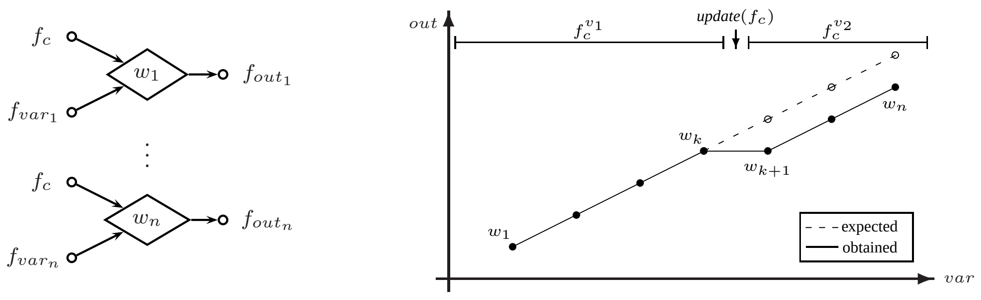


Figure 3. Violation of the G-isolation level on a parameter sweep. On the left the executed instances. On the right, the obtained and expected output.

of our knowledge this is the first proposal in such context. In our research, we plan to investigate several aspects, mainly three: provenance tied with version management, efficient execution management and the binding with a storage system. This section gives more details about our directions for each aspect.

First, the isolation levels rely on dependency graphs, which may be well represented using a data provenance model. Several provenance management systems [2] already offer the possibility of automatically capturing and storing provenance for later querying. The challenge is to be able to reuse such systems, introducing new functionalities for isolation management as we propose. This involves research on provenance data models, storing strategies, querying expressiveness and performance.

Second, one of the fundamental aspects on execution management (scheduling in particular) is to be able to efficiently select the files to be used and to move the computation near to them. The isolation levels would notably affect this aspect by imposing conditions on the file selection. As a consequence, we need to propose new scheduling algorithms that incorporate these new requirements.

Last, several systems have been suggested for the management of files in the context of workflow execution both in the Grid [1] and in the Cloud [10]. In the same way, numerous versioning file systems have been presented [6, 8]. Nevertheless, to the best of our knowledge, there is no integral solution. Hence, our challenge is twofold: first, we should study the optimization of the file versions' storage by taking into account the strategies offered by those systems, e.g. compression and pruning for versioning, data placement or replication; and second, we have to combine a versioning system with a file management system. At a long term, the result have to be integrated with a WFMS for data management.

References

[1] A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunszt, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger, and B. Tierney. Giggle: a framework for constructing scalable replica location services. In *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, SC' 02, pages 1–17, Baltimore, Maryland, 2002. IEEE Computer Society Press.

[2] S. B. Davidson and J. Freire. Provenance and scientific workflows: challenges and opportunities. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 1345–1350, Vancouver, Canada, 2008. ACM.

[3] D. De Roure, C. Goble, and R. Stevens. The design and realisation of the myExperiment Virtual Research Environment for social sharing of workflows. *Future Generation Computer Systems*, 25(5), May 2009.

[4] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. P. Berman, and P. Maechling. Data sharing options for scientific workflows on amazon ec2. In *Proceedings of the 2010 ACM/IEEE International Conference*

for High Performance Computing, Networking, Storage and Analysis, SC'10, pages 1–9, Washington, DC, USA, 2010. IEEE Computer Society.

[5] L. Moreau, J. Freire, J. Futrelle, R. E. McGrath, J. Myers, and P. Paulson. The open provenance model. TR 14979, ECS, Univ. of Southampton, 2007.

[6] B. Nicolae, G. Antoniu, L. Bougé, D. Moise, and A. Carpen-Amarie. BlobSeer: Next generation data management for large scale infrastructures. *Journal of Parallel and Distributed Computing*, Aug. 2010.

[7] O. Sonmez, N. Yigitbasi, S. Abrishami, A. Iosup, and D. Epema. Performance analysis of dynamic workflow scheduling in multicluster grids. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC'10, pages 49–60, New York, NY, USA, 2010. ACM.

[8] C. A. N. Soules, G. R. Goodson, J. D. Strunk, and G. R. Ganger. Metadata Efficiency in Versioning File Systems. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, FAST '03, 2003.

[9] I. Taylor, E. Deelman, D. Gannon, and M. Shields, editors. *Workflows for e-Science : Scientific Workflows for Grids*. Springer, New York, 2007.

[10] D. Yuan, Y. Yang, X. Liu, G. Zhang, and J. Chen. A data dependency based strategy for intermediate data storage in scientific cloud workflow systems. *Concurrency and Computation: Practice and Experience*, 2010.