Managing Provenance in iRODS

Andrea Weise[1], Adil Hasan[2], Mark Hedges[3], and Jens Jensen[4]

[1] Centre for Advanced Computing and Emerging Technologies (ACET), University of Reading, UK
a.weise@reading.ac.uk
[2] English Department, Liverpool University, UK
adilhasan2@googlemail.com
[3] Centre for e-Research, King's College London, UK
mark.hedges@kcl.ac.uk
[4] Science and Technology Facilities Council, Rutherford Appleton Laboratory, UK
jens.jensen@stfc.ac.uk

Abstract. Nowadays provenance is an important issue. Provenance data does not only give a history of events, it also provides enough information to allow the opportunity to verify the authenticity of the data, as well as, determine the quality of the data. The data grid management system, iRODS, comes with metadata which can be used as provenance data. Currently, iRODS's metadata is not sufficient for tracking and reconstructing procedures applied to data. In this paper, we describe the provenance needs of iRODS and we survey briefly current provenance and provenance enabled workflow systems. We describe an architecture that can be used to manage provenance in iRODS (and other systems) in a fault-tolerant way.

1 Introduction

1.1 Background

The work presented in this paper grew out of investigations in the "Architecture for a Shibboleth-Protected iRODS System" ASPiS project. The aim of this project is to add support for provenance capture to the Rule Oriented Data System (iRODS), to develop Shibboleth single sign-on access to iRODS, and to deploy this for groups of users who are currently using the Storage Resource Broker (SRB). iRODS is developed by the Data Intensive Cyber Environments (DICE) group (developers of the SRB), and collaborators [1].

This paper presents two results, first an evaluation of two existing general purpose provenance systems with respect to use cases provided by our current users of SRB. Secondly, based on this evaluation we describe work done in the ASPiS project to develop a generic provenance system for a distributed environment, in particular for the data grid management system iRODS.

1.2 iRODS

The Rule Oriented Data System (iRODS) [1], is an implementation of a "data grid". It is often seen as the successor to the Storage Resource Broker (SRB).

G. Allen et al. (Eds.): ICCS 2009, Part II, LNCS 5545, pp. 667–676, 2009.

Both systems are able to provide uniform access to heterogeneous storage devices across the network and, as a result, make the storage infrastructure appear transparent to the end user [2]. The significant difference between the SRB and iRODS lies in the way the data can be managed within the system itself. iRODS, recently released as version 2.0, comes with a rule engine. With the introduction of rules, iRODS is able to adapt to many scenarios and the user is able the manage their own data in almost any way. Additionally, the implementation of services to manage or process the data, such as data conversion, can be easily achieved by the end user. The rule engine enforces rules and therefore, acts as interpreter of the rules [3]. Rules are composed of the actual event, conditions, action sets and recovery sets [4], and applied through microservices. Microservices are functions written in C [1], which can be provided by anyone to organise and structure the data. Through rules and microservices different applications can be accessed by iRODS, e.g. communication to a web service or data conversion, and workflows can be defined. The rule engine is invoked from certain procedures within the iRODS code, at points which correspond to certain actions being performed; for example, just before (or just after) reading (or depositing) a file. The rule engine will first analyse the request to determine whether there is a rule defined in the rule base that corresponds to the action. If there is, the rule engine will execute the rule as defined (e.g. converting a deposited file to a different file format). If the rule executes successfully, the main processing continues from just after the point at which the engine was invoked. To manage the data, iRODS keeps "standard" metadata such as information about the file itself (e.g. file size, last accessed, owner, etc.) as well as user-defined metadata. The user-defined metadata can be connected with individual files which have been submitted to iRODS. The metadata are stored in the "Meta Data Base" (iCAT).

1.3 Related Work

In this project, provenance is seen as the history of the operations applied to a digital object. Other words for provenance can be history, pedigree, parentage, genealogy, filiation, or lineage[5]. The term "provenance" in scientific research can be defined in a number of ways in different contexts, [6], [7]. Based on the variety of different provenance systems, it can be said that provenance capturing systems play more and more of a dominant role in research. The Taverna [8] software from the myGrid project was primarily developed to manage user-defined workflows, and has a bioinformatic background. Some tools, such as VisTrails [6] and Taverna, aim to support the researcher by providing a platform that can combine different components, e.g different web services, and make them accessible through a single interface. Systems like REDUX [9] and the Virtual Data System (VDS) [10] use proprietary approaches to provide new ways of capturing and querying provenance data. Each of these systems were designed for a particular purpose and cannot easily be applied to any other existing system. Generic systems are provided by the Provenance Aware Service-Oriented Architecture (PASOA) and the Karma framework which we evaluate in this paper.

Another generic information system is Relational Grid Monitoring Architecture (R-GMA) from the "Enabling Grids for E-science in Europe" (EGEE) project [11]. Building on a producer/ consumer model, this system carries generic logging and accounting information in grids, and could be adapted to manage provenance data in a distributed and heterogeneous environment.

2 Requirements

Our use cases posed the following requirements:

1. Manage data throughout its lifecycle: from inception to final publication. It is the norm that data is written once and must not then be modified; when data is analysed, new files are created. Some data is not useful forever, or cannot be kept forever for other reasons, and may need to cleaned up.
2. Capture and record information about the data analysis: which files were analysed, how they were processed, where the result is stored.
3. Enforce proper ownership of data throughout its lifetime: for some of our users, the owner is the Virtual Organisation (VO). For others the owner is the *research proposal* which was submitted by the principal investigator – data created in the project is associated with the proposal throughout and the investigator can add or remove people. Ownership is used by investigators to define access control policies.
4. Ensure data access is auditable.
5. Ensure the infrastructure is *robust* and *scalable*.

We focus on the provenance related requirements, integrating them with iRODS-based data storage with callouts to systems managing provenance data, thus fulfilling most of the requirements.

All of our users have requirements for data provenance. Some of this data is (or can be) maintained natively by iRODS, such as file length, basic checksums, time created, etc. Checksums and file length are especially valuable provenance data because those information can be used to identify a particular file version, detect changes and verify integrity. For the remaining some can be automatically generated by custom built microservices, others must be provided by the user. We primarily focused on the former, and chose to keep the data in an external provenance system instead of storing it in the iCAT, because iRODS' built-in user metadata system is not built for provenance. Finally, the iRODS schema may change with new releases.

3 Evaluation of Karma and PASOA

In this section, we provide brief description of the two generic provenance systems we surveyed and evaluated.

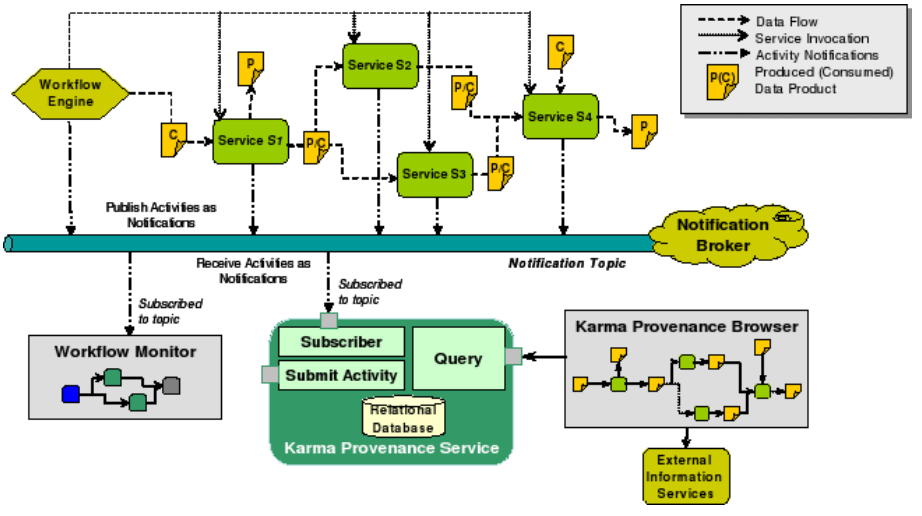

Fig. 1. Karma Provenance Framework

3.1 Karma

Karma aims to develop a provenance framework that will "record uniform and usable provenance metadata independent of the workflow or service framework used" [12]. The framework is written in Java and is therefore platform independent. Karma is used in an environment with workflows, services, service clients, and data products. The output of one service can serve as the input to the next [12]. Provenance data ("activities") within Karma are formatted XML messages which are submitted via a web services interface (synchronously). The Karma framework also provides an asynchronous publish-subscribe notification protocol based on [13]. Workflow engines and participating services publish their activities (events) to the notification broker. The subscriber and the Karma provenance service, respectively, subscribe to certain channels of interest and will get the information only for those channels they subscribed to. The publisher and subscriber are thus decoupled from each other and the subscriber does not necessarily know the actual source of the information [13]. Fig. 1 taken from [12] shows the interaction of the Karma2 framework with a workflow engine.

3.2 PASOA

An outcome of the "EU Provenance Project" funded by the European Commission's Sixth Framework Programme was the open provenance architecture [14]. The "Provenance Aware Service-oriented Architecture" – PASOA – project is based on that architecture.

PASOA aims to provide an independent and standardised protocol for capturing, recording, and accessing provenance which should be applicable to any

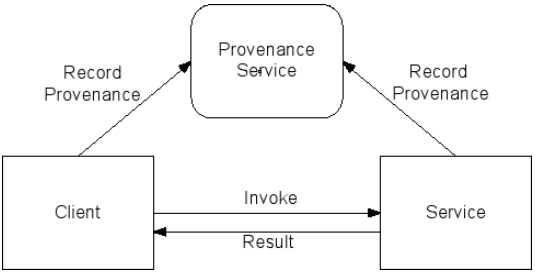

Fig. 2. The interaction between a client service and provenance service

system. The requirements in the project are trust, preservation, security, scalability, generality, and customisability [15]. These fit well with our requirements.

PASOA relies on a third party provenance service. One advantage of outsourcing the provenance service, according to PASOA, is that the workflow system itself does not have to deal with the provenance handling. Fig. 2 taken from [15] displays the basic approach of PASOA.

Each participant of a collaboration sends information to the provenance service. The provenance service itself is a web service with a local storage mechanism, e.g. a database. PASOA implements its own proprietary communication protocol. Future plans for PASOA are to provide eventually a protocol, which can serve as a standard for provenance capturing. Currently, the protocol consists of four phases: negotiation, invocation, provenance recording, and termination. The negotiation phase is used to arrange which of the provenance services is used, e.g. recording or querying. The invocation phase will invoke the actual service which was determined during the previous phase. Within the provenance recording phase, the provenance data is submitted to PASOA, whose success will be confirmed by sending a finished message in the termination phase[15].

4 Architecture

4.1 Using iRODS to Manage Provenance

Extending previous work in [6], we distinguish between capturing, recording and storing, processing, displaying, and querying provenance metadata. Capturing needs to recognise the data that is coming in. For recording and storage, we develop a way for microservices to call out to a provenance store (web service). We are not currently addressing display of provenance metadata within the ASPiS project. Until such features can be implemented, there are external applications that let users view and query the metadata held in the provenance store.

iRODS does not by default capture changes made to data in rule-based workflows. We can not rely on the user entering it – the user may be absent or may not know the workflow in detail. The metadata kept by iRODS itself is not sufficient: it does not capture the workflow.

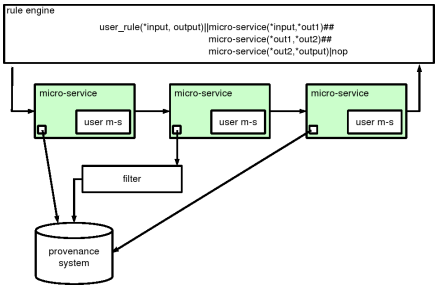

Fig. 3. Microservice Wrapper

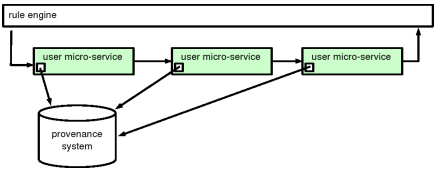

Fig. 4. User Microservice Chain

We looked at two ways of capturing metadata about the workflow within iRODS. Figure 3 shows a wrapper microservice which captures all information (but may in principle filter it before recording it). Figure 4 shows user microservices modified to record provenance metadata. The former has the advantage of requiring no changes to the user microservices; all metadata is recorded. Another advantage is that it can be used to optimise the workflow because it can keep track of profiling data for the microservices. The disadvantage is that, in general, it does not know about the data and is not able to capture specific user defined information. In this project we chose to focus on the latter case, where user microservices are modified to record provenance data. This enables us to record precisely the data that is needed, which may include the following[1]:

- User defined provenance data.
- Data about the user: identity, authorisation such as roles and group memberships, home institution, potentially other Shibboleth attributes (subject to data protection and other rules)
- File provenance data: filename, length, checksum, dates written and modified, owner, and various types of integrity management metadata: checksums, number of copies held in the underlying storage, where they are physically stored, their access latency. Which other iRODS services have this data?
- Access to data: who accessed it and did what to it, why they were granted permission?
- Content provenance data: what is contained in the file, what is the format of the file, version of file (a version that makes sense to the user and one that makes sense to iRODS).
- Which microservices, and which version of the services, have processed the data.
- Which rules and which version of rules, have processed the data.
- We discussed keeping track of versions and configurations of iRODS itself. Changing the configuration or upgrading iRODS to a new version will not change the data, but might introduce changes to the iCAT. Moreover, our use cases may require users using microservices in iRODS to manage their

[1] Some of this is relevant to provenance and some of it is data storage "housekeeping".

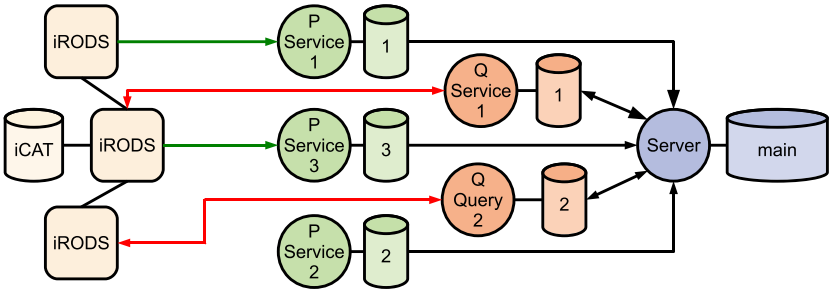

Fig. 5. iRODS Provenance Framework Overview

analysis workflow, and an upgrade to the system could change the way these are run.

Since iRODS is able to contact existing applications such as external workflow systems there are issues with how to extract provenance data from an external system, how to relate it to the iRODS provenance information, and how that information is stored (format). However, this is currently out of the scope in this project.

4.2 Proposed Provenance System

PASOA and Karma rely on a web service to be provenance aware application independent. In a distributed environment with an unknown amount of nodes sending requests, a single web service can become a single point of failure. For both systems it can be said, if either the web service or the database behind the system fails, the entire provenance capturing framework will fail. Further, the single web service can also become a performance bottleneck.

To resolve the problem of having a single point of failure, the querying and recording services will have several access points. Fig. 5 shows the outline for the proposed distributed provenance system which will be used in connection with iRODS, whereas iRODS will be able to access the web services through microservices as described in chapter 4.1.

We distinguish between web services for querying (in the following referred to as "Q-Services") and those for recording provenance metadata ("P-Services"). Each service has its own local storage mechanism (e.g. database) which serves as storage cache and replicates part of the main database. This will reduce the traffic to the core data storage device as well as increase the response time for the Q-Services. Each service caches its storage content independently. There will be a constant synchronisation between the main database and each P-Service cache to ensure consistency with a configurable synchronisation interval. Consistency can be forced by using triggers. In that case, the content will be forwarded immediately to the main storage device. The Q-Services will try to answer queries from the data contained in the attached database. If the service can not answer the

query, it will forward the request successively to n other services where n is configurable. This forwarding will reduce the traffic to the core database and will in general reduce the response time for the requesting client. If a query service gets a forwarded query, it will respond to the sender with either a positive or negative acknowledgement. If the requested data is available (positive acknowledgement), the service will respond directly to the client by returning the query results. If a sending Q-Service receives a negative acknowledgement, the next service in the queue is contacted. If all attempts fail to find another suitable query service, the original contacted Q-Service will request the data from the core database. It can be assumed that the possibility for the client to contact the same service with a similar query is higher than for the client to contact a different service due to the way web services are discovered. Introducing a cache control system can reduce the time of finding another suitable query service and can control the number of requests for each service. But such a system is currently out of scope for this project. The need to have a cache control system e.g. proposed by Katoaka et. al [16] can be analysed after a successful implementation and resultant tests.

If a query is submitted before the data has been migrated to the central database, the Q-Service will fail to find the data as it cannot query P-Service directly. The use of triggers in P-Services and the core database service can reduce the delay for data which needs to be highly available. Although the main database should be backed up, the contents can be partly recovered by resynchronising with the P- or Q-Services databases.

When implemented in an iRODS-based data grid, each node will discover its nearest P-Service using the peer-to-peer (P2P) based approach of a "balanced distributed web service" look-up system described in [17]. Furthermore, each iRODS node will cache the address of its last accessed P or Q service.

5 Discussion

5.1 Technical Issues

Karma and PASOA provide web services for recording and querying metadata. We chose to use Tomcat to run the web services. The current Tomcat 6.x release, published under the Apache Software License, is the result of a nine year development and the acceptance in the user community is very high. Therefore, we consider this software product mature. In addition, our tests on PASOA showed that Tomcat itself was very stable.

Since microservices are written in C, and provenance services in Java, they are best linked using web services. To implement web services in C, we generate code with gSOAP, a cross-platform open source kit which is able to generate platform independent C/C++ source code based on a given WSDL file. We discovered that gSOAP was sensitive to the WSDL input, and a namespace compatibility problem had to be debugged. Moreover, some WSDL files generated code which could not be linked. Once these problems were fixed, however, we managed to get a stable and reliable interface.

5.2 Future Work

Within the ASPiS project we have looked at recording, storing, and querying provenance metadata. We have not looked in depth at displaying metadata, but there are tools that can be used to query data captured by PASOA [18]. In future work, we will look at closer integration with such tools, as well as workflow and other computational metadata, particularly for the National Grid Service and the portals used by the users of this service.

We will also need to look at the mechanism for registering microservices and maintaining the versions and integrity of them. Currently, we rely on the developers to maintain the version information, i.e., update it when the microservices is modified, but this does not protect against malicious modifications (or accidental ones, or developers who forget to update the version number).

6 Conclusion

In this paper we have enhanced iRODS by incorporating provenance functionality. As part of our research we surveyed existing provenance systems. The survey resulted in the fact that most provenance system or provenance enabled work flow systems are designed for a certain puprose and are therefore system dependent. Only the frameworks of PASOA and Karma offer system independency. iRODS rules and microservices were chosen to connect the data management system with such independent provenance frameworks. This way, there will be no interference with the iRODS core system and the emerging system can be seamlessly integrated in any distributed environment.

PASOA and Karma work with web services, which are the current state of the art of web applications. However, both provenance systems have a single point of failure which makes them difficult to use in a distributed environment. The proposed system eliminates this complication by dividing the provenance services into two major categories, submitting and querying services, which will enhance stability. By increasing the number of access points for those services and storage mechanisms, the new system becomes more fault tolerant and therefore, highly available. The access nodes will automatically scale by applying a P2P based algorithm which will provide an efficient and fault tolerant web service discovery mechanism.

Acknowledgement

This work, which is part of the ASPiS project, was funded by the UK Joint Information Systems Committee (JISC) as part of its e-Infrastructure programme, with additional support from UK Science and Technology Facilities Council (STFC). The authors would like to express their gratitude to the SRB and iRODS staff at SDSC and the University of North Carolina. The first author would like to express her gratitude to Prof. V. Alexandrov, ACET, Reading University.

References

1. About irods, https://www.irods.org/index.php/Introduction_to_iRODS
2. Rajasekar, A., Wan, M., Moore, R., Schroeder, W., Kremenek, G., Jagatheesan, A., Cowart, C., Zhu, B., Chen, S.Y., Olschanowsky, R.: Storage resource broker - managing distributed data in a grid. Technical report, San Diego Supercomputer Center (SDSC), University of California
3. Rule engine, https://www.irods.org/index.php/Rule_Engine
4. Rules, https://www.irods.org/index.php/Rules
5. Simmhan, Y.L., Plale, B., Gannon, D.: A survey of data provenance techniques. Technical report (2005)
6. Freire, J., Koop, D., Santos, E., Silva, C.T.: Provenance for computational tasks: A survey. Computing in Science & Engineering 10(3), 11–21 (2008)
7. Simmhan, Y.L., Plale, B., Gannon, D.: A framework for collecting provenance in data-centric scientific workflows. In: IEEE International Conference on Web Services, pp. 427–436 (2006)
8. Taverna 1.7.1 manual, http://www.mygrid.org.uk/usermanual1.7/
9. Barga, R.S., Digiampietri, L.A.: Automatic capture and efficient storage of e-science experiment provenance. Concurrency and Computation: Practice and Experience 20(5), 419–429 (2008)
10. Foster, I., Vöckler, J.S., Wilde, M., Zhao, Y.: Chimera: A virtual data system for representing, querying, and automating data derivation. In: SSDBM 2002: Proceedings of the 14th International Conference on Scientific and Statistical Database Management, Washington, DC, USA, pp. 37–46. IEEE Computer Society, Los Alamitos (2002)
11. R-gma: Relational grid monitoring architecture, http://www.r-gma.org/
12. Simmhan, Y.L., Plale, B., Gannon, D.: Karma2: Provenance management for data-driven workflows. Int. J. Web Service Res. 5(2), 1–22 (2008)
13. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.M.: The many faces of publish/subscribe. ACM Computing Surveys 35, 114–131 (2003)
14. Moreau, L., Ibbotson, J.: The EU Provenance Project: Enabling and Supporting Provenance in Grids for Complex Problems (Final Report). Technical report, The EU Provenance Consortium (2006)
15. Groth, P., Luck, M., Moreau, L.: Formalising a protocol for recording provenance in grids. In: The UK OST e-Science second All Hands Meeting 2004, AHM 2004 (2004)
16. Kataoka, M., Toumura, K., Okita, H., Yamamoto, J., Suzuki, T.: Distributed cache system for large-scale networks. In: International Multi-Conference on Computing in the Global Information Technology, 2006. ICCGI 2006, p. 40 (August 2006)
17. Sioutas, S., Sakkopoulos, E., Drossos, L., Sirmakessis, S.: Balanced distributed web service lookup system. J. Netw. Comput. Appl. 31(2), 149–162 (2008)
18. The provenance architecture client side library, http://www.gridprovenance.org/software/CSLPage.html