

## **Distributed Provenance Data**

It has been stated that the provenance in the Grid can be distributed. I.e. the provenance records related to a sequence of actions may reside in multiple repositories. The following seem to be some of the reasons for that:

- 1 – The services involved in the sequence may allow the recording of the metadata only to designated repositories to control the user access to sensitive data.
- 2 - A certain sequence of activities may involve the collaboration of multiple users each of which may choose to record bits of metadata to their favourite repositories for convenience/security reasons.

## **Distributed Query Engine for Querying Provenance Data**

A Distributed Query Processing engine can help in querying the provenance data that is disseminated across multiple repositories. The following might be a scenario to use a DQP engine:

- 1 – A Virtual Organisation hosts a DQP service that supports queries over multiple provenance repositories. Dynamic addition of new repositories should also be supported. VO users should be granted access rights to added repositories based on their roles/privileges.
- 2 – The DQP engine may support a set of canned queries against well-known schemas of provenance data. It may also support generic queries via a query language (OQL/SQL).
- 3 – As the users submit queries, they may receive different results for the same query based on their roles.

## **Service-Based Management of Provenance Recording**

It would be useful if individual services (including the enactment engine) know what they are supposed to record (i.e. log) as part of their provenance tracking, and where to record it. I.e. a service should know the schema of the metadata it is supposed to record, during its execution. It should also know the repository (or the service which wraps the repository) for recording that information. In that way, service providers can control what information to record about their services, and the location of the information, which is important for authorisation, access control and non-repudiation issues. This, however, requires a mechanism to link/associate separate bits of related provenance records to each other. It would, therefore, be desirable to rely on globally unique identifiers for data elements (e.g. LSID), and also couple each interaction (call to a service) with certain information such as the userid, authority, an interaction contextid and possibly some user supplied annotation.

It also makes sense to record provenance information in this way, rather than generating and passing it to the client every time the user runs a workflow, from a performance point view, because there might be huge amount of data being generated

and the user, at times, may only be interested in a fraction of the generated information. So it is more reasonable to extract the needed information via a query to the distributed repositories that contain all the recorded information about a certain workflow.

## **Infrastructure Support**

There should be some infrastructure support for supporting this approach. For example it must be possible to ask a service to provide its schema for recording provenance information, as well as the address of the repository (or the service that wraps the repository) that holds the information. In OGSA terms MyGrid services would have dedicated SDEs for this information. It would also be desirable to have framework support for a standard provenance schema and container-managed recording of that information.