TeraGrid's GRAM Auditing & Accounting, & its Integration with the LEAD Science Gateway

Stuart Martin[1,2], Peter Lane[1,2], Ian Foster[1,2,3], and Marcus Christie[4]

Abstract— Science Gateways have been proposed as a means of lowering the barrier to scientists and their applications using TeraGrid resources. A Science Gateway provides an application- or domain-specific interface that a scientist can easily understand; under the covers, its implementation uses Web Services interfaces (including those provided by the Globus Toolkit) to access computers, storage, and other TeraGrid resources. A single gateway may have 1000s of users who may collectively generate many thousands of file transfers and job submissions. Thus, efficiency and scalability are paramount. The GRAM service provides secure scalable efficient access to remote computing resources, but some auditing enhancements are needed to allow gateway users to be multiplexed over a common/service credential. We describe how auditing information is produced by Globus Toolkit services and integrated with local accounting systems for use by TeraGrid's resource providers. We also describe how the LEAD Gateway has integrated GRAM auditing capabilities to provide auditing and accounting information for workflows of jobs submitted by users of the LEAD Portal.

Index Terms— gateways, security, auditing, accounting, Globus, Grid, Job Execution Service, GRAM

1 INTRODUCTION

We define a Web Services interface and associated mechanisms to provide access to audit and accounting information associated with Grid services. The current focus for TeraGrid is accounting of compute resources consumed by users, so the Globus Toolkit's GRAM job submission and management service [4] was enhanced to produce an auditing record for each job. However, we believe this approach can apply to any grid service and associated accounting database. For example, users are typically not charged for data transfer and data storage services in production grids, but this system provides the structure to make it possible. Some parts of this system are TeraGrid specific, but it can be (1) leveraged by any Grid Service Providers to provide GRAM clients new options for secure access to service audit information in scalable and efficient ways, and (2) integrated with a Grid's accounting system to provide Web Service access via OGSA-DAI to usage information.

We use the LEAD gateway [8] to demonstrate a successful integration with this new GRAM audit and accounting interface for TeraGrid's compute resources.

2 GRAM BACKGROUND

The Globus Toolkit provides both Web Services and "pre-Web Services" interfaces for securely and reliably submitting, monitoring, and controlling jobs on remote resources. The implementations of both sets of interfaces are known as "GRAM," for Grid Resource Allocation and Management; the term "WS GRAM" (or "GRAM4" [5]) refers only to the Web Services implementation.

Jobs are computational tasks that may perform input/output operations that affect the state of the computational resource and its associated file systems. In practice, such jobs may require the coordinated staging of data into the resource prior to job execution and out of the resource following execution. Some users may want to access output data as the job is running. Monitoring consists of querying and/or subscribing for status information such as job state changes. Control operations allow the user to terminate the job via both soft-state lifetime management and explicit terminations [6].

Grid computing resources are typically operated under the control of a local resource manager (e.g. PBS, LSF, SGE, Condor, etc) that implements allocation and prioritization policies while optimizing the execution of all submitted jobs for efficiency and performance. GRAM is not a local resource manager, but rather a protocol engine for communicating with a range of different local resource managers using a standard message format.

3 AUDITING AND ACCOUNTING USE CASES

We use three use cases to motivate the design of the facilities described in this document.

UC1: Group Access. A grid resource provider allows a remote service (e.g., a gateway or portal) to submit jobs on behalf of multiple users. The grid resource provider only obtains information about the identity of the remote submitting service and thus does not know the identity of the users for which the grid jobs are submitted. This group access is allowed under the condition that the remote service store audit information so that, if and when needed, the grid resource provider can request and obtain information to track a specific job back to an individual user. This approach is consistent with TeraGrid's plans for Gateway access using a "community" account.

UC2: Query Job Accounting. A client that submits a job needs to be able to obtain, after the job has completed, information about the resources consumed by that job. In

[1] Computation Institute, University of Chicago & Argonne National Laboratory, USA
[2] Math & Computer Science Division, Argonne National Laboratory, Argonne IL, USA
[3] Department of Computer Science, University of Chicago, IL, USA
[4] Indiana University, USA

portal and gateway environments where many users submit many jobs against a single allocation, the per-job accounting information is needed soon after the job completes so that client-side accounting can be updated. Accounting information is sensitive and thus should only be released to authorized parties.

UC3: Auditing. In a distributed multi-site environment, it can be necessary to investigate various forms of suspected intrusion and abuse. In such cases, we may need to access an audit trail of the actions performed by a service. When accessing this audit trail, it will frequently be important to be able to relate specific actions to the user requests that caused them to be performed. Audit information is sensitive and thus should only be released to authorized parties.

4 Requirements

We extract from the use cases the following requirements for an audit and accounting system:

- **Grid Job Identifier (GJID)**: We require an identifier that a client can use to refer to a job when requesting audit and accounting records (UC1, UC3), and that a GRAM service can use to refer to a job when requesting submitter identity information from a client (UC2). This identifier should be globally unique in time and space.
- **Client Interface**: We require an interface that allows a client to access audit and accounting records associated with the jobs executed by a GRAM service (UC1, UC3). This interface should allow the client to retrieve information about a particular job or set of jobs (subject to authorization).
- **Creation of Audit and Accounting Information**: We require mechanisms for recording “audit records” for important events that occur when a grid job is processed and “accounting records” for the resources consumed by a job
- **Access to Audit and Accounting Information**. We require mechanisms for retrieving this information from other sources, such as accounting databases maintained by local resource managers.
- **Scalability**: We need interfaces and implementations capable of dealing with large numbers of audit records. We do not have firm data yet on the exact numbers, but O(100,000) records seems likely.
- **Authentication and Authorization**: We require secure authentication and authorization mechanisms to ensure that only appropriately authenticated and authorized clients can access audit and accounting information.

We do not list as an immediate requirement support for remote monitoring and management of the auditing and accounting service, although such capabilities may be required in the future.

5 Interface Design

We present a design based on the following concepts and mechanisms:

- A *GJID construction algorithm* is used to generate unique GJIDs from the endpoint references (EPRs) constructed by GRAM4 for each job.
- An OGSA-DAI-based [3] *audit and accounting interface* provide access to audit and accounting information via standard Web Services mechanisms.
- Standard GT4 *authentication and authorization mechanisms* can be used to control access to audit and accounting information.

5.1 Grid Job Identifier (GJID)

We require a GJID that can be shared between client and service and used to uniquely identify a grid job. Ideally, this identifier will be something that is available in the normal interaction (protocol) with the service, so that no additional communication is required to obtain it.

In GRAM4, we can use the EPR returned by a job submission request for this purpose. An EPR is an XML document that contains an *address* field plus zero or more *reference properties*. We convert this document into a GJID as follows:

1) Obtain the address part of EPR as a string
2) Obtain the first reference property (assuming it is the resource key)
3) Canonicalize the reference property element
4) Generate a digest of the element from 3
5) Base64-encode this digest
6) Construct a string as follows: address from 1 + ? + encoded digest from 5

For example, given the following example EPR returned by a GRAM4 service to a client:

```
<ns1:managedJobEndpoint xmlns:ns1=
    "http://www.globus.org/namespaces/2004/10/gram/job">
  <ns2:Address xmlns:ns2=
    "http://schemas.xmlsoap.org/ws/2004/03/addressing">
https://127.0.0.1:8443/wsrf/services/ManagedExecutableJobService
  </ns2:Address>
  <ns3:ReferenceProperties xmlns:ns3=
       "http://schemas.xmlsoap.org/ws/2004/03/addressing">
    <ns1:ResourceID cca8169a-c65f-11da-a61c-000d61215ff0
    </ns1:ResourceID>
  </ns3:ReferenceProperties>
  <ns4:ReferenceParameters
     xmlns:ns4=
     "http://schemas.xmlsoap.org/ws/2004/03/addressing"/>
</ns1:managedJobEndpoint>
```

we obtain the following GJID:

```
https://127.0.0.1:8443/wsrf/services/ManagedExecutableJobService?
QQDzjbFVYImtVg8
```

5.2 Client Interface

We define a flexible query interface to the audit and accounting information. To simplify remote client access, we also define operations to invoke specific queries. For example, motivated by UC2 (query job accounting), we define an operation that takes as input a GJID and returns the “charged” amount from the TeraGrid’s central accounting database.

We could have implemented these interfaces within the GRAM service instead of in a separate audit service. However, the interface would then presumably provide access only to audit and accounting records associated with that specific GRAM service. The use of a separate audit service means that a single service can (but need not) provide access to records from multiple GRAM services.

Ultimately, we may also wish to associate a management interface with the audit service, to enable remote management of distributed components.

5.3 Authentication and Authorization

We want to apply standard GT4 authentication and authorization to operations invoked via the auditing and accounting interface. Thus:

- Any client issuing a request must present a valid X.509 certificate.
- Policy enforcement can then be applied based on a range of policies.

Initially, we implement a simple policy that allows access to a record if and only if the Grid UID of the requestor matches the Grid UID in the audit record for the GJID. This policy is sufficient for UC1 and UC2, but not UC3.

To support this policy, the Grid UID (i.e., X509 subject name) of the client that originally submitted the job must be included in the audit record.

6 Implementation

The implementation of our audit and accounting interface must ensure that required audit and accounting records are generated and/or obtained from existing sources. For brevity, we refer to any service that implements this interface as an "audit service." TeraGrid's audit service for compute resources consists of GRAM, OGSA-DAI and TeraGrid's central database (TGCDB). To understand their relationship, we present an architectural diagram (Figure 1) with a numbered sequence of events with descriptions.

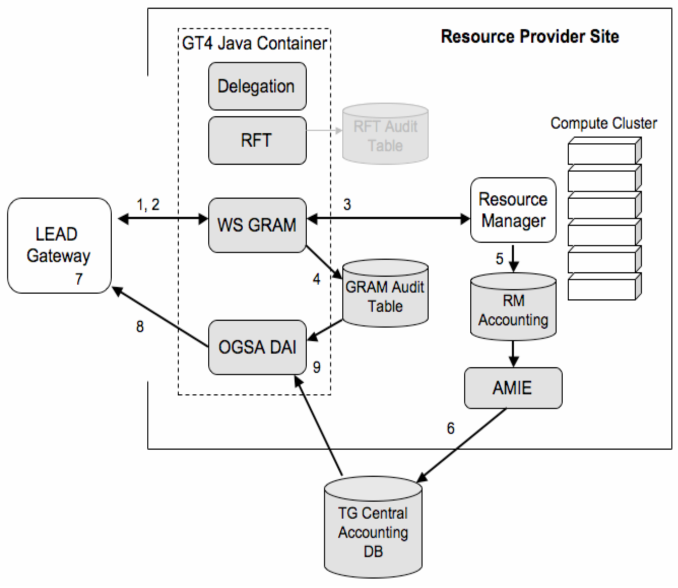


Fig. 1: TG GRAM Auditing and Accounting Architecture

1) Gateway submits job and gets an EPR on the reply
2) Gateway controls and monitors job with EPR
3) GRAM submits and monitors job in RM
4) GRAM inserts audit record at end of job
5) RM writes job accounting record
6) AMIE uploads RM accounting records to TGCDB. The RM accounting record is converted to TG accounting units.
7) Gateway locally converts EPR to GJID
8) Gateway calls OGSA-DAI getChargeForJob with GJID nd gets the job usage on the reply
9) OGSA-DAI processes remote join between GRAM audit and TGCDB

Audit and accounting records may be generated and stored by different entities in different contexts. In this case, we assume that audit records are generated by the GRAM service itself and accounting records by the local resource manager (LRM) to which the GRAM service submit jobs. Thus, we can expect that audit records are stored in a database (of some sort) indexed by GJID, while accounting records are maintained by the LRM indexed by a local job identifier (JID). For TeraGrid, the GRAM audit records are maintained in a database co-located with each GRAM service, while the LRM accounting records from all grid resource providers are maintained in TGCDB.

To connect these two sets of records, GRAM records a job's local JID in each audit record that it generates. It is then straightforward for the audit service to respond to requests such as those described in Section 5.2 above by first selecting matching record(s) from the audit table and then using the local JID(s) to retrieve relevant accounting record(s) from the accounting table. In reality, this join to TGCDB requires more than just the local JID, but those details are configurable in the audit service, in order to keep the interface simple for the Gateways.

The UK eScience OGSA-DAI software (distributed as part of the Globus Toolkit) can create a single virtual database from two or more remote databases. We use OGSA-DAI to create a virtual database from the GRAM audit database and TGCDB, in order to implement a "getChargeForJob" operation. We also use OGSA-DAI to provide a Web Services SQL query interface to the audit and accounting databases.

Other per-job information such as job performance data can be stored using the GJID or local JID as an index, and then made available in the same virtual database.

Additional details about the TeraGrid's audit-enabled GRAM services implementation is available online [1].

6.1 GRAM Service Auditing Database Schema

The postgresql database schema for the GRAM service auditing table is as follows.

```
create table gram_audit_table (
    "job_grid_id" varchar(256) primary key,
    "local_job_id" varchar(512) not null,
    "submission_job_id" varchar(512) not null,
    "subject_name" varchar(256) not null,
    "username" varchar(16) not null,
    "creation_time" timestamp not null,
    "queued_time" timestamp not null,
    "stage_in_gid" varchar(256),
    "stage_out_grid_id" varchar(256),
    "clean_up_grid_id" varchar(256),
    "globus_toolkit_version" varchar(16) not null,
    "resource_manager_type" varchar(16) not null,
    "job_description" text not null,
    "success_flag" boolean not null
);
```

Both GRAM2 and GRAM4 have been modified to insert records into a database table using this schema.

6.2 GRAM Service GJID creation

GRAM4 returns an EPR that is used to control the job. The EPR is an XML document and thus cannot be used effectively as a primary key for a database table. It needs to be converted from an EPR to an acceptable GJID format. We wrote a utility class EPRUtil.java to provide the method to perform the conversion. This method will be included in the upcoming GT 4.0.4 release. It is used by both the GRAM service before storing the audit record and the GRAM client (Gateway) before doing a getChargeForJob() invocation.

GRAM2 returns a “job contact” string that is used to control the job. The job contact is by default an acceptable GJID format, so we use it as the GJID directly. The GRAM2 client and service do not need to convert it in any way.

7 Example OGSA-DAI Queries

Charge: A client can use the following Java code to retrieve the charge for a GRAM job.

```
TeraGridGetChargeForJob chargeQuery =
   new TeraGridGetChargeForJob(gridJobId);
service.perform(chargeQuery);
double charge = chargeQuery.getCharge();
System.out.println("The charge for this job is: " + charge);
```

Audit: A client can retrieve desired fields from an audit record by using the standard OGSA-DAI client interface to execute a “perform document.” In the following, the local_job_id and the queued_time are returned for the given GJID for the given Grid UID.

```
<?xml version="1.0" encoding="UTF-8"?>
<perform xmlns=
      "http://ogsadai.org.uk/namespaces/2005/10/types">
  <documentation>
    This example performs a simple select statement to retrieve
    one row from the test database. The results are delivered
    within the response document.
  </documentation>--
  <sqlQueryStatement name="statement">
    <expression>
      select local_job_id,queued_time
      from gram_audit_table
      where job_grid_id='https://tg-
grid1.uc.teragrid.org:9554/wsrf/services/ManagedExecutableJobServ
ice?Tb1eLvO6mVl/Of9KGw9nSOmgGmU='
      AND subject_name=
'/DC=org/DC=doegrids/OU=People/CN=Peter G Lane 364243'
    </expression>
    <resultStream name="statementOutputRS"/>
  </sqlQueryStatement>
  <sqlResultsToXML name="statementRSToXML">
    <resultSet from="statementOutputRS"/>
    <webRowSet name="statementOutput"/>
  </sqlResultsToXML>
</perform>
```

Accounting: Using the standard OGSA-DAI client interface, a remote query can be executed using a “perform document” to retrieve desired fields from an accounting record. Here, we focus on the <expression> element, since the rest of the auditing perform document example can be reused. The charge is returned for a specific TeraGrid job. Note the additional resource_name and time constraints that are required to ensure that a unique (and correct) job is identified. These details are hidden from a client in the getChargeForJob operation.

```
<expression>
  select charge from jobs
    where local_jobid='287254.tg-master.uc.teragrid.org' and
      resource_name='dtf.anl.teragrid' and '2006-06-22 15:44:10'
      between submit_time - INTERVAL '24 hours' and
      submit_time + INTERVAL '24 hours'
</expression>
```

8 GRAM Auditing in the LEAD Gateway

The Linked Environments for Atmospheric Discovery (LEAD) project [8] is a large, five-year Information Technology Research project funded by the National Science Foundation in October 2003. The project has two primary objectives. The first is “to lower the entry barrier for using, and increase the sophistication of problems that can be addressed by, complex end-to-end weather analysis and forecasting/simulation tools.” The second objective involves “improving our understanding of and ability to detect, analyze and predict mesoscale atmospheric phenomena by interacting with weather in a dynamically adaptive manner.”

The LEAD Gateway is fronted by the LEAD Portal, a web container that principally addresses the first objective by providing easy to use web based user interfaces (implemented as JSR-168 Java portlets) to powerful forecasting and analysis tools. The LEAD Portal provides interfaces to the user for creating, configuring, launching and monitoring workflows of application web services that run on the LEAD testbed. These workflows are defined using the WS-BPEL specification and are executed by the GPEL (Grid Process Execution Language) workflow engine. GPEL invokes application Web Services—Web Services wrappers for meteorological application codes. These application Web Services (implemented using the Generic Service Toolkit [12]) are the processes that run the application codes on TeraGrid, using GRAM to execute the codes and GridFTP [2] to manage input and output data files (these services have access to the LEAD community account’s grid proxy credential). The application web services communicate with the workflow and other services via a publish/subscribe notification system.

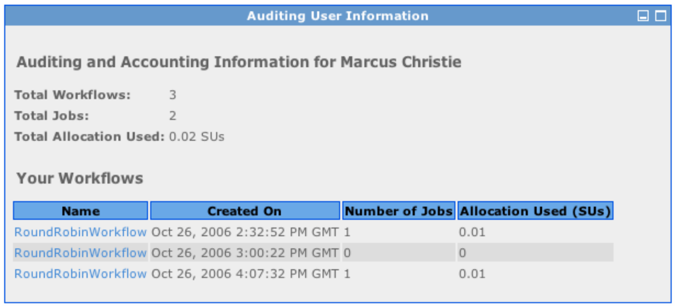
To integrate the GRAM auditing capability with the LEAD Gateway, we introduce a new Audit service in the LEAD architecture. This service collects auditing and accounting records related to jobs run on behalf of users and aggregates this information, by users and by workflows.

When a user submits a workflow to execute in the LEAD Portal, a call is made to the LEAD Audit service with the ID of the user who submitted the workflow, the ID of the workflow, the subscription topic on which notifications about this workflow will be published, etc. The LEAD Audit service stores this information in a database and then it subscribes to notifications related to this workflow.

When application services are invoked by the workflow engine, they invoke the application via a GRAM job submission and then send a notification containing, amongst other things, the grid job id of that job. The LEAD Audit service receives these notifications and stores their information in its database.

In summary, the LEAD Audit service is plugged into the LEAD architecture by listening for events on the notification bus. Neither the workflow engine nor the application services are aware of the LEAD Audit service and thus there is no dependency between them. Finally, at periodic intervals, the LEAD Audit service scans the jobs in its database and contacts the GRAM auditing and accounting OGSA-DAI services to update each job's local job id and the amount of the allocation used. Using database queries it is able to provide the total amount of allocation used per user and per workflow.

We also introduce an Audit portlet into the LEAD Portal. This portlet provides a simple interface to the information stored in the LEAD Audit service database. A user can see how much of the community allocation they have consumed, as well as a break down of allocation used per workflow. The user can further drill down to see how much allocation was used by each job within a workflow.



Auditing and Accounting Information for Marcus Christie

Total Workflows: 3
Total Jobs: 2
Total Allocation Used: 0.02 SUs

Your Workflows

Name	Created On	Number of Jobs	Allocation Used (SUs)
RoundRobinWorkflow	Oct 26, 2006 2:32:52 PM GMT	1	0.01
RoundRobinWorkflow	Oct 26, 2006 3:00:22 PM GMT	0	0
RoundRobinWorkflow	Oct 26, 2006 4:07:32 PM GMT	1	0.01

Fig. 2: LEAD Audit portlet shows aggregate and per workflow usage

ACKNOWLEDGMENTS

Many people providd input to the design and implementation of the GRAM auditing capabilities. In particular, we thank Ally Hume, Rachana Ananthakrishnan, John Paul Navarro, Ravi Madduri, and Martin Feller. This work was supported in part by the National Science Foundation under contract OCI-0534113 and by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Dept. of Energy, under Contract DE-AC02-06CH11357.

REFERENCES

1. TeraGrid GRAM Audit Service, www.teragridforum.org/mediawiki/index.php?title=GRAM4_Audit, 2007.
2. Allcock, B., Bresnahan, J., Kettimuthu, R., Link, M., Dumitrescu, C., Raicu, I. and Foster, I., The Globus Striped GridFTP Framework and Server. in *SC'2005*, (2005).
3. Antonioletti, M., Atkinson, M., Baxter, R., Borley, A., Chue Hong, N., Dantressangle, P., Hume, A., Jackson, M., Krause, A., Laws, S., Parsons, P., Paton, N., Schopf, J., Sugden, T., Watson, P. and Vyvyan, D., OGSA-DAI Status and Benchmarks. in *UK e-Science All Hands Meeting*, (Nottingham, England, 2005).
4. Czajkowski, K., Foster, I., Karonis, N., Kesselman, C., Martin, S., Smith, W. and Tuecke, S., A Resource Management Architecture for Metacomputing Systems. in *4th Workshop on Job Scheduling Strategies for Parallel Processing*, (1998), Springer-Verlag, 62-82.
5. Feller, M., Foster, I. and Martin, S. *GT4 GRAM: A Functionality and Performance Study*. Globus Alliance, 2007.
6. Foster, I., Czajkowski, K., Ferguson, D., Frey, J., Graham, S., Maguire, T., Snelling, D. and Tuecke, S. Modeling and Managing State in Distributed Systems: The Role of OGSI and WSRF. *Proceedings of the IEEE*, *93* (3). 604-612. 2005.
7. Kandaswamy, G., Fang, L., Huang, Y., Shirasuna, S., Marru, S. and Gannon, D. Building Web Services for Scientific Grid Applications. *IBM Journal of Research and Development*, *50* (2/3). 249-260. 2006.
8. Plale, B., Gannon, D., Brotzge, J., Droegemeier, K., Kurose, J., McLaughlin, D., Wilhelmson, R., Graves, S., Ramamurthy, M., Clark, R.D., Yalda, S., Reed, D.A., Joseph, E. and Chandrasekar, V. CASA and LEAD: Adaptive Cyberinfrastructure for Real-Time Multiscale Weather Forecasting. *IEEE Computer* (Nov). 56-64. 2006.
9. Droegemeier, K.K. and Co-Authors, 2007: A new paradigm for mesoscale meteorology: Grid and web services-oriented research and education in LEAD. Preprints, *23rd Int. Conf. on Interactive Information Processing Systems for Meteorology*, 14-18 January, San Antonio, TX, Amer. Meteor. Soc.
10. Marcus Christie and Suresh Marru, "The LEAD Portal: a TeraGrid gateway and application service architecture", Concurrency and Computation: Practice and Experience, John Wiley & Sons, 2007, To Appear.
11. Aleksander Slominski, "On using BPEL extensibility to implement OGSI and WSRF grid workflows", Concurrency and Computation: Practice and Experience, John Wiley & Sons, Volume 18, Issue 10, pp. 1229-1241, 2006.
12. Generic Service Toolkit, online, http://www.extreme.indiana.edu/gfac/, Feb 14, 2007.