OGSA-DAI: Open Grid Services Architecture Data Access and Integration

OGSA-DAI demonstrates the University of Edinburgh's ability to combine research on data access and integration using grid and web service technology and in-house software engineering expertise to create outputs which benefit international research.

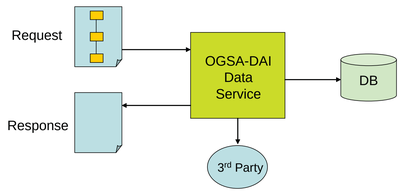


Figure 1: The basic OGSA-DAI data service model

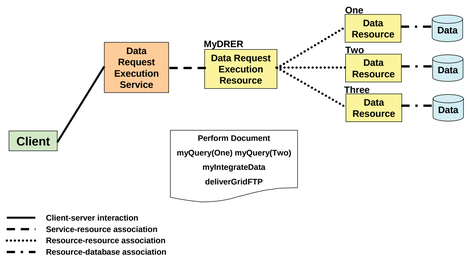


Figure 3: The OGSA-DAI Resource Model

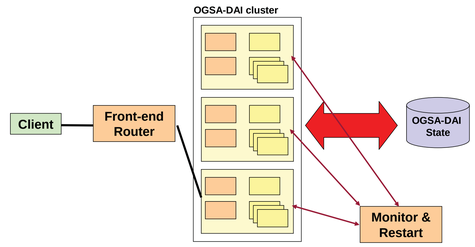
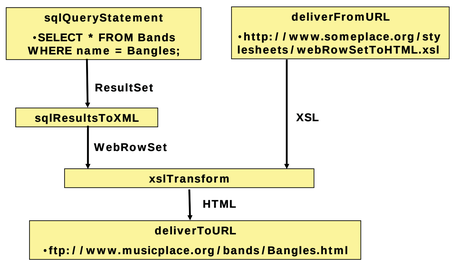


Figure 5: Clustering services to provide scalability

Figure 2: A pictorial representation of a simple perform document showing a query about the Bangles being performed which is then transformed into HTML, based on a stylesheet provided via an URL, and uploaded to a website.



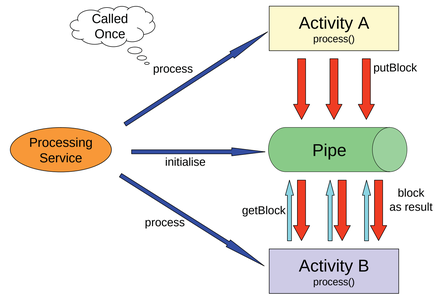


Figure 4: The Activity Framework, showing a Pipe being utilised to transfer data between activities

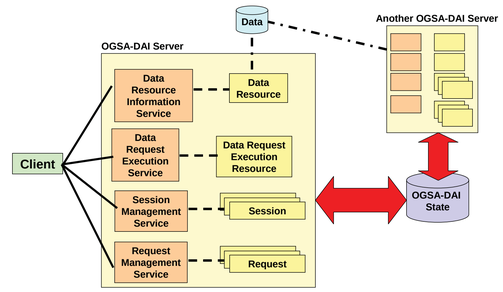


Figure 6: Increasing service robustness through persistence

Introduction

OGSA-DAI [1,2] is an extensible framework for data access and integration providing consistent interfaces to heterogeneous data resources in a grid/web environment with an emphasis on flexibility, functionality, performance and scalability. It allows interaction with these resources and provides additional processing functionality which can efficiently handle the data exposed. The University of Edinburgh has worked closely with database vendors (IBM, Oracle) and other research groups (University of Manchester, University of Newcastle) to lead a project which has both defined standards through the Open Grid Forum, and produced an implementation which is being used to support further research and e-Science.

Seen at a high level (Figure 1), an OGSA-DAI data service accepts a request (called a perform document) specified in XML and produces a response according to the instructions in the request. A perform document contains activities (the base unit of work in OGSA-DAI) that specify the required functionality. Activities can be chained together, thus creating a pipeline where data is streamed between activities and iteratively processed in small, manageable blocks. Once the activities have completed, a response document is sent containing details of the completion status. Results can be delivered in the response, or via a separate delivery method. OGSA-DAI provides a powerful, data-centric workflow for e-Science, which in turn can be nested in higher-level workflow languages such as BPEL [3]. This architecture [4] was developed from the outputs of the UK Database Task Force [5] and, in particular, the perform document model was evolved following discussions and a scenario pattern evaluation with IBM, which pointed to the need for a more powerful interface than the simple Remote Procedure Call (RPC) style utilised by many web services.

A perform document (see Figure 2 for pictorial representation) will typically contain instructions to: access a data resource; perform transformations on the data acquired; and deliver the results (either through the response document or via OGSA-DAI's delivery activities to an alternative destination). This highlights three of the four sets of operations which are useful for data services, the fourth being management operations.

OGSA-DAI is based on the Web Services Resource Framework (WS-RF) model [6]. An OGSA-DAI execution service (Figure 3) contains a Data Request Execution Resource that is associated to many Data Resources which each expose an underlying data source. General purpose integration activities exploit this model, and are implemented as parameterised activities with java methods that provide functionality like sorting, joining, and selecting relational structures. Activities may combine common patterns of several operations to improve efficiency e.g.: TupleGenericSort, TupleGenericSelectProject, TupleGenericJoinSelectProject, TupleGenericProductSelectProject.

Efficient intra-service data transfer

The Activity Framework (Figure 4) is a core component of OGSA-DAI, responsible for enacting activities and streaming the data between them. When two activities are connected, one will usually produce some data blocks and pass them to a pipe object. When the other activity wants to consume a data block, it will ask for one from the pipe. The pipes between activities act as buffers for the data blocks that have to be exchanged. Activities and pipes are initialised once. Activities execute in their own threads, producing and sending blocks to the pipe for buffering and subsequent consumption and retrieval by other activities. This approach maximises utilisation between consumer and producer.

Because of the way that the activity framework has been designed, it is easy to supplement the generic activities provided with custom activities based on the application or research domain. These activities can utilise the entire framework, reducing the barrier to extending OGSA-DAI functionality. Developers can also write their own Data Resources to expose any kind of resource that they may have available, allowing data virtualisation.

An Authorization Model for Data Services

Most security research related to Grid computing has concentrated on protecting job submission systems. Data services such as OGSA-DAI require a general framework that can support a variety of technologies and allows finer grained access control [7,8]. In particular, as well as providing policy execution points (PEPs) at the resource and activity level, the security infrastructure should allow the authorisation model to pass restrictions (obligations) on how a request is executed to activities. One example may be that a client is allowed to execute an SQL query so long as it does not return more than 1000 rows. There is a design trade off between fine and coarse grained approaches. With a fine grained approach an activity must specify all the details regarding how the resource can be accessed, including what activities can run on it. This can place a large burden on how the system administrator configures the activity. A more coarse grained approach may associate permission groupings with resources.

Scalability and Robustness

The OGSA-DAI architecture provides the basis for future provision of clustering and load balancing (Figure 5), including a monitoring service which could provide information such as load and utilisation. This provides better scalability since the number of concurrent requests that can be handled increases with the number of OGSA-DAI services in the cluster. Moreover, these services can be monitored, and in case of node failures requests could be seamlessly migrated to other OGSA-DAI services.

To allow configuration and state to persist between container shutdowns and restarts there is a mechanism for storing this data in databases and allowing the reconfiguration of new services from that persistent information. Persistence could also be used for caching to reduce overheads of OGSA-DAI to database communications.

Research using OGSA-DAI

The OGSA-DAI project started in 2002 and now has over 2500 registered users. In this time, several other research projects have built on top of OGSA-DAI as a base platform, including:

- OGSA-DQP (Distributed Query Processing) [9] provides a query planning and evaluation framework which allows complex queries to be split into sub-queries which are executed in parallel across many OGSA-DAI services.
- GridMiner [10] has developed tools to support data mining of medical infosets using OGSA-DAI, including a BPEL enactment engine and a data mediation component.
- DatabaseGrid [11] extends the uniform view of heterogeneous database resources in grid environments to WebDBs and RDF stores.

Across the world, a number of large collaborative projects have made OGSA-DAI a part of their software environment. Some examples include:

- OntoGrid, the provision of semantics in the Grid in a principled manner.
- caBIG, a network to connect the entire USA cancer community;
- NAREGI, a large-scale computing environment for widely-distributed advanced research and education;
- SIMDAT, federation of problem solving environments for industry;
- UNIDART, a uniform data request interface for the access to meteorological data and products.

References

[1] M. Antonioletti et al., The Design and Implementation of Grid Database Services in OGSA-DAI. Concurrency and Computation: Practice and Experience, Volume 17, Issue 2-4, p 357-376, 2005.

[2] K. Karasavvas et al., An Introduction to OGSA-DAI Services. LNCS Volume 3458, p1-12, 2005.

[3] T. Andrews et al., Business Process Execution Language for Web Services. (BPEL4WS). Version 1.1. May 2003.

[4] M. Atkinson et al., A New Architecture for OGSA-DAI in UK e-Science All Hands Meeting, 2005.

[5] M.P. Atkinson et al., Grid Database Access and Integration: Requirements and Functionalities. Global Grid Forum Informational Document (GFD.13), 2003.

[6] S. Graham et al., Web Services Resource 1.2 (WS-Resource). OASIS, January 2006.

[7] D. Power et al., A Secure Wrapper for OGSA-DAI. LNCS Volume 3470, p 485-494, 2005.

[8] A.L. Pereira et al., Role-Based Access Control for Grid Database Services. First DIALOGUE Workshop: Applications-Driven Issues in Data Grids, Columbus, Ohio, 2005.

[9] N. Alpdemir et al., OGSA-DQP: A grid service for distributed querying on the grid. LNCS Volume 2992, p 858-861, 2004.

[10] P. Brezany et al., GridMiner: A Fundamental Infrastructure for Building Intelligent Grid Systems. Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), p 150-156, 2005.

[11] S. Mirza et al., OGSA-WebDB: An OGSA-Based System for Bringing Web Databases into the Grid, Journal of Digital Information Management, Vol.2. No.2., 2004.

This work has been supported by the UK eScience Core Programme, EPSRC and the DTI.

National e-Science Centre

www.ogsadai.org.uk

epcc