GGF International Summer School on Grid Computing 2005 (ISSGC05)

Day 6 Exercise – Using OGSA-DAI for data access

1 Introduction to the Exercise

The OGSA-DAI middleware supports Access and Integration of data held in relational and XML databases, and in collections of files. The technology includes a variety of ways of delivering result data, including third-party delivery. The framework is also extensible, so you can write activities which can be run in an OGSA-DAI container to provide specialised data extraction close to the stored data. You will explore a basic sample of the available facilities. The version that you use will be running on the GT4 platform.

2 Learning Goals

This exercise will reinforce and build on the outcomes of stage 3. It will add the following skills.

1 The ability to access both relational and XML databases via a grid or web service.

2 The ability to read, combine and write data from databases accessed in this way.

3 The integration of database access into a data search application.

3 Stages in the Exercise

The exercise is presented as a series of stages so that you can incrementally learn about OGSA-DAI and more about the use of web and grid services. You will complete parts of a standard OGSA-DAI tutorial to help you prepare for the exercise. You will learn most if you work individually until the end of step 4.2 then work as a team.

By now you should be automatically noting how long understanding and using a technology takes and how fast that technology accomplishes tasks. We will not remind you further.

Step 4.0 Access to databases using OGSA-DAI

Complete the OGSA-DAI tutorial at least up to section "Processing Results". You will learn how to query relational and XML databases on the Grid and how to process the results.

Step 4.1 Access a relational table

Write a client application that uses OGSA-DAI and its client library to obtain and display the contents of table `cuboids`. The table contains a set of x, y coordinates each of which is near or within a cuboid on a surface. You could think of this as the results of earlier research that have obtained some approximate and partial results which you are going to improve. Those earlier researchers believe their results are within ±3 of the corresponding cuboid's centre in each case. Organise your search in the next step assuming they are correct.

Step 4.2 Search guided by database data

Write a client application that reads the same data as used in step 4.1 and finds a cuboid corresponding to each row in table `cuboid`s on the surface returned by the service `cuboids`. Obtain values that have an estimated accuracy of 1% for the location (min_x, min_y) and dimensions of each cuboid. Generate and store a table of this information in an RDB.

You may add an improvement. Cubiods cannot intersect. Therefore, if there is a cuboid recorded in the database that apparently intersects, you could raise a warning rather than storing the new value. If they are coincident within experimental error, then ignore the re-discovery. You could do this as you record results or as a post-processing phase.

Each search can use a modification the code that you developed in exercise 3.5. You could submit a number of jobs to perform these searches concurrently, provided you ran an initial job to set up the table to hold the results. The scale of this task does not require that strategy.

Step 4.3 Search with data integration

In many cases a researcher makes a breakthrough by combining existing data from more than one source, using a relationship that is derived from a hypothesis behind their research. This step will emulate such a combination.

Use the table `mountains` to obtain approximate locations of mountains from earlier observations. These may be less accurate than in previous examples. It refers to cones rising from a peniplane (geographic term for geomorphology that is nearly flat – we have approximated it as flat) that are represented by the surface-generating service `mountains`. They all have bases of radius 1 but vary in height. Write a client application to find all of them and determine their height.

The document `mountainNames` contains XML giving names for these mountains and their context (continent or country) in the order of their previously estimated heights (highest first, i.e. with id=1). Present the mountain locations, their heights, names and context.