GMA and the relational model – progress report, protocols, and where we might have strayed...

Abdeslem Djaoui, RAL

Steve Fisher, RAL <s.m.fisher@rl.ac.uk>

James Magowan, IBM & RAL

Gavin McCance, Glasgow

Manfred Oevers, IBM & RAL

(people with code in CVS)
European DataGrid

- EU DataGrid project with 21 partners from a number of countries around Europe is part-funded by the EU.

- It brings together:
  - particle physics,
  - earth observation
  - bio-informatics
  - computer science

- For information and monitoring system, seeing if we can do “better” than MDS/LDAP
Grid Monitoring Architecture (gma)

- We use it not only for monitoring but also as the basis of an information system

Producer -> register -> Registry
Consumer -> lookup

We have chosen a relational implementation
Relational Approach

• Not a general distributed RDBMS system, but a way to use the relational model in a distributed environment where ACID properties are not generally important.

• Producers announce what they have by an SQL "CREATE TABLE" statement and publish it with an SQL "INSERT"

• Consumers collect information by an SQL "SELECT" statement.

• It may use a number of RDBMS for archival, for the schema and for the registry
Registration of producers

• Tables are effectively partitioned over several sites – we assume that each partition is characterised by fixed values of one or more columns (e.g. “site=CERN” or “site=RAL”)

• Register the name of the table with the names of any attributes which are fixed and the values of those attributes.

• No explicit registration is carried out by the normal user.
Schema

• When a producer registers itself as a producer of a certain table, if the table is not known it is added to the schema. If a table is no longer used its definition can be removed.
  – Handles schema evolution.

• The schema information must be universally known within a VO.

• Schema will need to be replicated for scalability and reliability, but it must done in such a way as to allow producers around the world to add new tables.
Implementation - servlets

- Code is concentrated in servlets
- With one exception, all communication with servlets (even from another servlet) is via an interface object.
  - This interface object can be coded in multiple programming languages
  - Hides the communication bit (so it can be changed more easily)
Schematic
(no RDBMS)

Sensor → Producer → Producer Servlet → Query

Invocation

Consumer → Consumer Servlet → Schema

Information

Registry → Registry Servlet

Schema → Schema Servlet
Implementation - XML

• We use normal http parameters so that URLs such as:
  \[\text{http://localhost:8080/ProducerServlet/getOne?id=6&select=SELECT} \ldots\]
  will work – i.e. no XML encoding of the input.

• All responses as XML
  – Exceptions are trapped by the servlet and error message sent back as XML
```xml
<?xml version = '1.0' standalone = 'yes'?>
<ROWSET>
  <ROWDESC>
    <COL>animal</COL>
    <COL>legs</COL>
  </ROWDESC>
  <ROW>
    <COL>aardvark</COL>
    <COL>4</COL>
  </ROW>
  <ROW>
    <COL>tarantula</COL>
    <COL>8</COL>
  </ROW>
</ROWSET>
```
Multipart response

- We use multipart response to stream information
  - content type defines separator
  - keeps connection open.
Comparison with proposed “Simple XML protocol”

- XML between consumer and producer is described by a single DTD/Schema
  - This is more compact though less “XML like”
- XML is only used for the response
  - Otherwise use http parameters – hence would need more if it were separated from http
- Relies upon http(s)
Comparison with “normal” GMA

• true to the GMA spirit as we understand it

• but:

  – Producer’s don’t find consumers (e.g. archives)
    • It is not really consistent with the GMA model for the producer to solicit client
      – In our case a person instantiating a producer could also instantiate an archive or tell an archive to consume from the producer
    – The user is not aware of the registry
  – Sensors not explicit – a sensor/application instantiates a producer to publish its information
Schematic (no RDBMS)

- Sensor
- Producer
- Producer Servlet
- Query
- Registry
- Registry Servlet
- Schema
- Schema Servlet
- Consumer
- Consumer Servlet

Invocation
Information
Archiver (with RDBMS)

- Archiver
- RDBMS (JDBC)
- Producer
- Producer Servlet
- Consumer
- Consumer Servlet
- Registry

Invocation
Information
Query Splitter

- Consumer
- Consumer Servlet
- Query Splitter
- Query Splitter Servlet
- Registry
- Consumers

Invocation
Information
Technology choices

- All in Java
- Tomcat servlet engine
- MySQL (and postgresql) via JDBC
- HTML queries – return XML
- Multipart response
- DOM parser to read XML
- JavaCC to build SQL parser
- JUnit as test framework
- Log4j for internal logging
Problems

• How to maintain schema
  – Fairly static
  – Replicated
  – Reliable and consistent
  – Might require that dynamic names are prefaced by the originating DNS

• How to maintain registry of producers
  – Dynamic
  – Replicated
  – Maybe inconsistent
  – Maybe adopt/adapt an idea of EU DataGrid WP2

• How to do the query splitting.
  – Probably mostly in the textbooks
  – Hope we can find suitable open source code out there
Future work

- Tidy up what we have
- Archiver
- Access to data from GRIS
- C++ API for Producer and Consumer
- Package into RPM
- First release end of month (31st July 2001)

We would welcome early (tolerant and friendly) adopters