Experiences of Designing and Implementing Grid Data Services in the OGSA-DAI project

Neil Chue Hong / Mike Jackson
EPCC
N.ChueHong@epcc.ed.ac.uk
michaelj@epcc.ed.ac.uk
with: Mario Antonioletti, Ally Hume, Amy Krause, Jeremy Nowell, Charaka Palansuriya, Tom Sugden, Martin Westhead
Outline

- About the project
  - Aims

- About the software
  - Design
  - Functionality
  - Service and lifetime model
  - Security
  - Performance
Contributing to the global grid computing community

£3 million, 18 months, started February 2002
Funded by the Grid Core Programme

Designing and Building Grid Services workshop - 8 October 2003 - www.ogsadai.org.uk
Running a Grid project

- **Distributed teams, distributed development**
  - need to be able to share information, ideas and “chatter”

- **Use technology**
  - CVS, Eclipse for development
  - Bugzilla for issue tracking
  - Twiki for information sharing
  - IRC for discussion
  - Email, telcons, face to faces for brainstorming and agreement

- **Use process: change control, close knit teams, buddying**
OGSA-DAI Approach

- Reuse existing technologies and standards
  - OGSA, GT3, Query langs, Java, transport
- Build portTypes and services which will enable:
  - controlled exposure of heterogenous data resources on an OGSI-compliant grid
  - access to these resource via common interfaces using existing underlying query mechanisms
  - (ultimately) data integration across distributed data resources
1a. Request to Registry for sources of data about “x”

1b. Registry responds with Factory handle

2a. Request to Factory for access to database

2b. Factory creates GridDataService to manage access

2c. Factory returns handle of GDS to client

3a. Client queries GDS with SQL, XPath, XQuery etc

3b. GDS interacts with database

3c. Results of query returned to client as XML

OR

3d. Results of query delivered to to 3rd party consumer e.g. via GridFTP

Database (Xindice MySQL Oracle DB2)

Factory GDSF

Analyst

Registry DAISGR

Consumer

Grid Data Service GDS

SOAP/HTTP

service creation

API Interactions

Designing and Building Grid Services workshop - 8 October 2003 - www.ogsadai.org.uk 37
Service Data Elements

- Used to provide information about
  - Service
    - Status reports on query completion
    - Capabilities
  - Data resource
    - Structure
    - Supported query languages
    - Type
  - Data
    - Structure
    - Classification
Service and Lifetime

- A GDS is roughly analogous to a connection
  - A GDS may be shared (could represent the result of a query)
  - but what is the behaviour when retrieving partial result sets?
- A GDSF is normally persistent
  - maps to a DB management system
- But a GDSF/GDS pair may also be an abstraction to an underlying system
Components of the GDS

GDS

Engine

Activity Handler

Activity

Activity Handler

Activity

Database
Security in OGSA-DAI

- **Message Level, Secure Conversation**
  - SimpleFileRoleMapper similar to Globus gridmap file
    - Does not scale!!!
  - Expect application developers to produce own Rolemappers for particular domains
  - Could “call out” to Community Authorisation Services
Profiling OGSA-DAI

- Where are the bottlenecks?
- Can these be addressed by:
  - Straightforward re-coding
  - Fundamental re-design
- Common Suspects:
  - Strings and StringBuffers
  - DOM trees
  - Threading
- Unknown black holes
Profiling OGSA-DAI

- **EJ-Profiler and Borland Optimizeit**
  - Quickly identify potential performance hot-spots

- **Log4J**
  - Logging to the milli-second
  - Facilitate data collection over repeated runs
Querying for N Rows

GridDataService::perform

Time (ms) vs. Rows

0 1000 2000 3000 4000 5000 6000 7000 8000 9000 10000

100 700 1300 1900 2500 3100 3700 4300 4900 5500 6100
Querying for N Rows

![Graph showing Engine Invocation time vs. rows]

- Time (ms) on the y-axis
- Rows on the x-axis
- Graph shows a trend indicating increasing time with increasing rows.
Querying for N Rows

GridDataService::Perform

- Engine Invocation

= 10 ms
Querying for N Rows

![Graph showing time vs. number of rows for different operations: Schema Validation, ConX / Stmt, Run Query, No More Rows, and CleanUp / Return. The graph uses a logarithmic scale for rows to better visualize the data at higher numbers.](image-url)
Querying for N Rows

![Chart showing the relationship between processing time and number of rows.](image-url)
Processing the WebRowSet

- **Human-readable format:**
  - Disabling this option only yields a small constant performance improvement

- **Extensive use of StringBuffer-String conversion**

- **Threading and synchronization:**
  - SimpleHandler: SQLActivity->CallThroughPipe
    - XMLRowSetInputStream: java.sql.ResultSet -> Row -> SynchronizedPipe
    - SQLActivity <- XMLRowSetInputStream
  - RunAheadHandler: DeliverToResponse<-CallThroughPipe

- **DOM tree traversal:**
  - Each string containing one row is appended to an increasingly large DOM tree
Future directions

- Get the basic functionality right first
- Make sure it’s efficient, robust and scalable
- Develop client APIs to make it easier for developers
- Extend towards higher level integration services
- Talk to users, projects to re-evaluate usage scenarios and requirements
- Let us know what you want!
Further information

- The OGSA-DAI Project Site:
  - http://www.ogsadai.org.uk
- The DAIS-WG site:
  - http://cs.man.ac.uk/grid-db
- OGSA-DAI Users Mailing list
  - users@ogsadai.org.uk
  - General discussion on grid data access and integration
- Formal support for OGSA-DAI releases
  - http://www.ogsadai.org.uk/support + support@ogsadai.org.uk
- OGSA-DAI training courses
The End

Questions?