



EDG Project: Database Management Services

Leanne Guy

for the EDG Data Management Work Package

EDG::WP2

Leanne.Guy@cern.ch

<http://cern.ch/leanne>

Databases are used within the EDG Project to store information pertaining to:

- File replication
 - Replica catalogue, Replica location index
- Metadata
 - Application specific: physics collections, medical image headers
 - Grid middleware: Storage element metadata, Replication metadata
- Logging and bookkeeping
- Monitoring
- Network statistics
- Status of grid services
- Service lists

and much more

Logging and Bookkeeping service (WP1: Workload management)

- Grid service that provides information about jobs
 - pending, running, current status, where submitted, etc
- Events sorted in a SQL database
 - relational model allows fast complex querying

Network performance information (WP7: Networking)

- Measurement of network performance metrics
 - round trip time, packet loss, throughput
 - needed by other grid services such as query optimisation
- Information stored in an SQL database
 - summary statistics, information, reports generated by querying

Replica Catalogue (WP2: Data Management)

- Contains LFN ⇔ PFN mappings
 - associated metadata stored in an SQL database

Metadata

- Grid middleware services
 - Query optimisation service
 - Replica catalogue
 - Header information for medical images
 - nominative and anonymous, confidential and public
- Applications metadata
 - collections, logical file sets

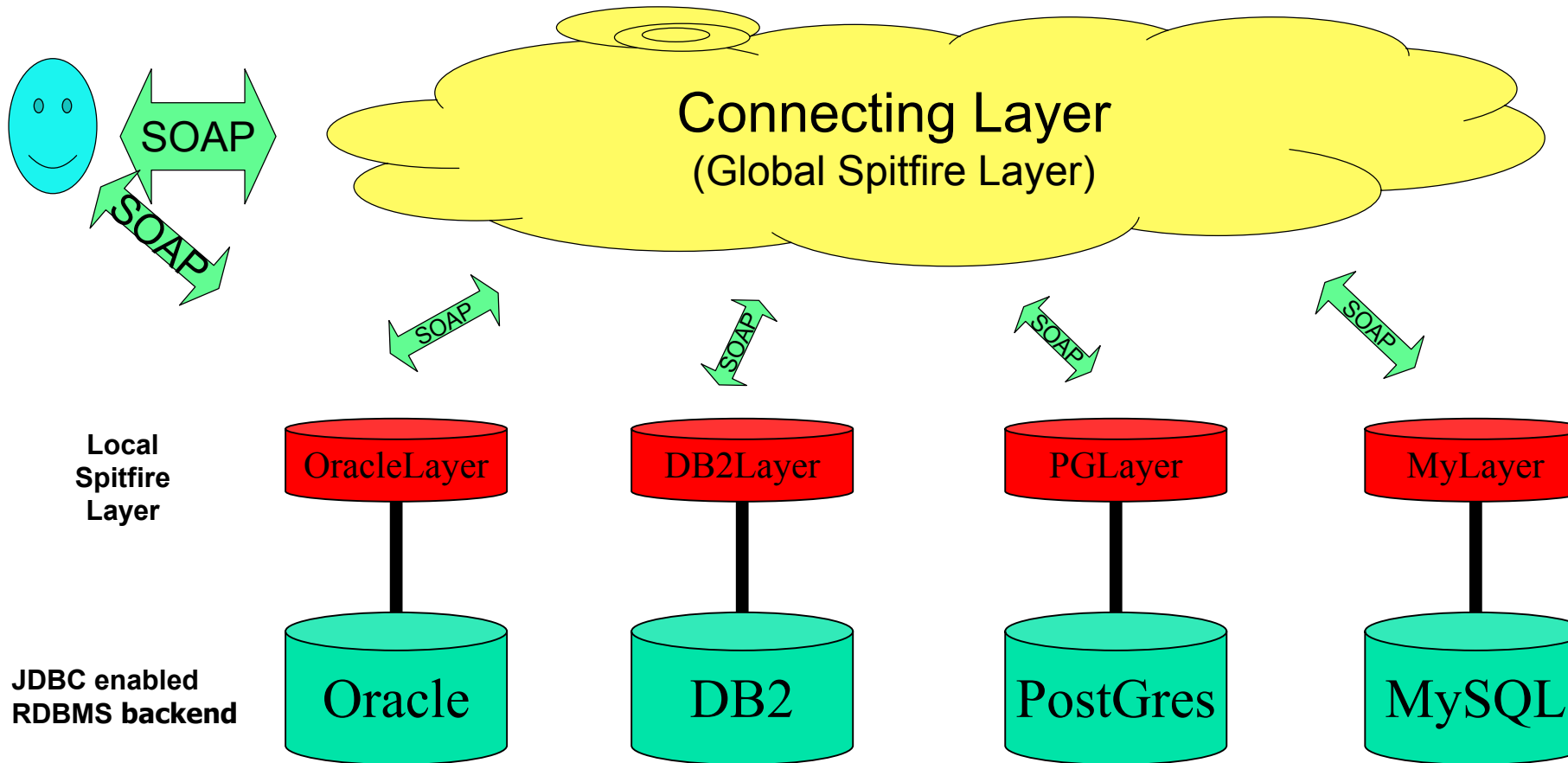
Spitfire: Grid Metadata Service provided by EDG::WP2

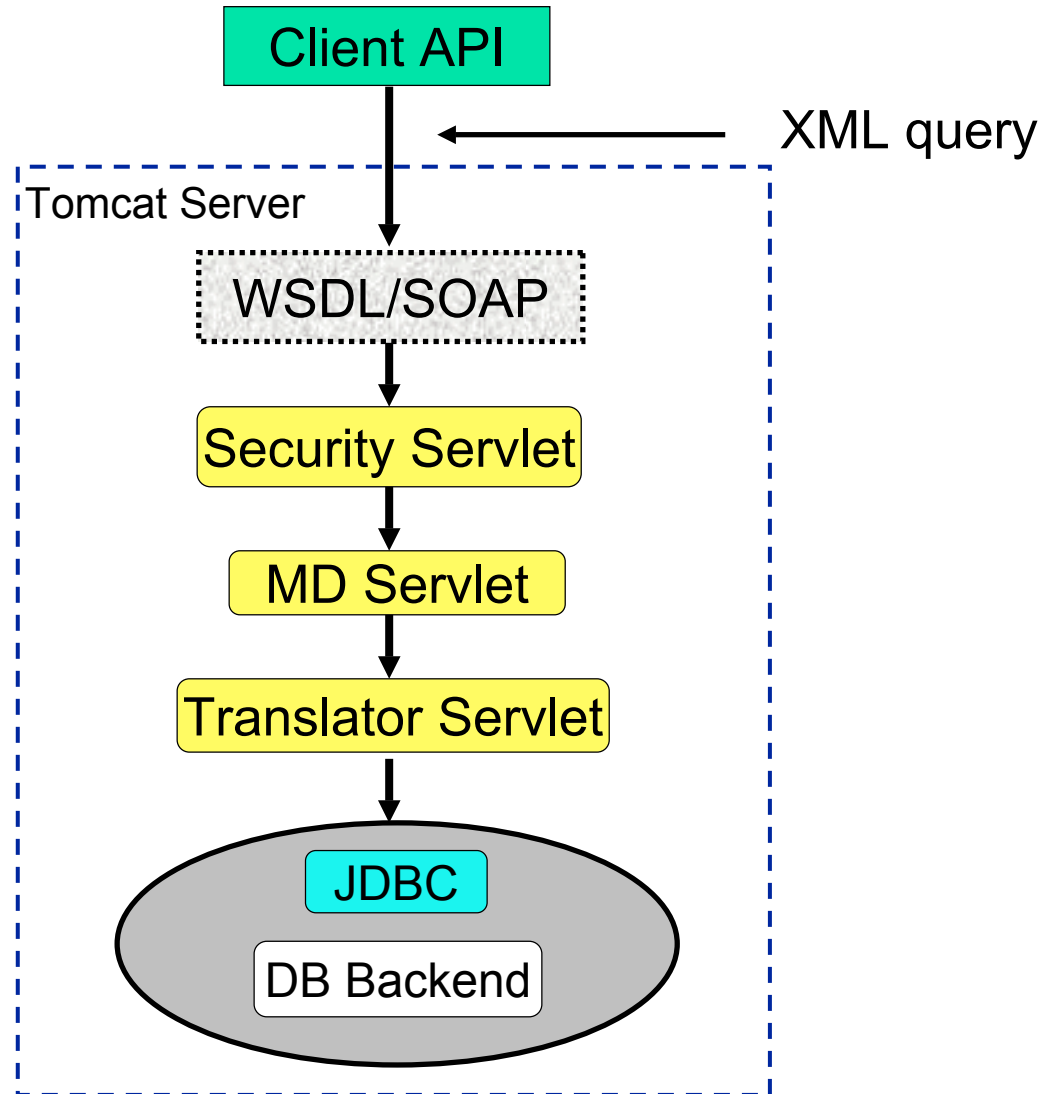
- provides access to various implementations of database backends via a grid enabled front end



Decouples the client from the RDBMS backend via a mediator

- Interoperability, ease-of-use, performance and plugability
- Atomic operations: consistency
- Core SQL functionality: update/insert/delete/query
- Web services
 - JDBC interface to RDBMS
 - SOAP interfaces
- Authorisation is role based
- jwget tool is provided for client side command line usage.





Client APIs to facilitate request issue and administration

- DB Administration API
 - Create / Delete Database
 - Create / Drop Table
- User Management API
 - Create / Delete / Update Role
 - Register / Unregister Users
- DB Information service API
 - Quotas, Memory, Disk space
 - User Info & roles, Schemata
- User Functionality API
 - Insert / Update / Delete / Select
 - Set Table Timestamp
 - Set Row Timestamp
 - Open Dedicated Connection
 - Close Connection

Higher level functionality

Not currently defined, will possibly include :

- Distributed Querying
 - Interactions with other metadata servers on the grid
 - Biomedical applications will need to query several metadata catalogues to obtain lists of specific medical images
 - Requires a definition of common schemata and indices.
- Replication / Caching mechanisms
 - Replication of a part or all of a database to another metadata service
- Expiration & Cleanup
 - Automatic removal of stale information based on stored timestamps
- Transactions

Medical metadata : WP10

- Use spitfire to store metadata for medical images
- Patient specific & image specific data
- Database will contain LFNs of collections of images
- Separation of confidential and non confidential data
- One SF metadata database instance per DICOM server
 - Queries on numerous distributed databases

Examples usage:

- Patient will query the SF database for all her/his medical images
- Physician will query SF database for all records on one patient
- Researcher queries SF database for all ultrasound images

Replica Catalogue

- Stores PFN <-> LFN mappings
- Metadata for files (PFNs) on the local storage element
 - file size, file status : durable, volatile, permanent

Replica Catalogue Metadata Catalogue: RepMeC

- Instance of Spitfire
- Stores attributes of LFNs independent of any associated PFNs
 - location of master and secondary copies for an LFN
 - VO specific data, e.g replication policy for LFNs
 - high level access control
- RepMec API
 - provides methods to define, store and extract metadata on replicas

Spitfire 1.2 : April release

- Integrated XSQL-Spitfire
- Installation decoupled from Tomcat & MySQL
- Tested to date with MySQL & PostgreSQL
- Java API , SOAP interfaces designed, not currently provided

Spitfire 1.3: end May 2002

- Alpha release of SOAP interface
- Schema and User Management
- Test-suite designed with Junit based tests implemented

Spitfire 1.4: end September (testbed 2.0)

- Version 1.0: Full integration of all components
- More running applications

In a distributed system such as a DataGrid:

- Want to maintain and query dynamic and timely information about Grid services, resources and virtual organisations?
- How should a database node maintain information populated from a large variety of unreliable, frequently changing, autonomous and heterogeneous remote data sources?
- In particular, how should it do so without sacrificing reliability, predictability and simplicity?
- How can powerful queries be expressed over time-sensitive dynamic information?

Use a Hyperlink Registry:

- A registry node has a database that holds a set of tuples
- A tuple may contain a piece of context
 - i.e a service description, file, picture, network load information, storage element information

- XML data model
 - allows for structured and semi-structured data, which is important for integration of heterogeneous content.
- Uses the XQuery language
 - allows for powerful searching
 - critical for non-trivial applications.
- Database state maintenance based on soft state
 - enables reliable, predictable and simple content integration from a large number of autonomous distributed content providers.
- Content link, content cache, hybrid pull/push communication model
 - allow for a wide range of dynamic content freshness policies,
 - driven by all three system components:
 - content provider, hyper registry and client.

- *Find all services that implement a replica catalog service interface and that CMS members are allowed to use, and that have an HTTP bindings for the replica catalog operation “XML getPFNs(String LFN)”.*

XQuery:

```
LET $repcat := "http://gridforum.org/interface/replicaCatalog-1.0"
FOR $tuple IN /tupleset/tuple[@type="service"]
WHERE SOME $op IN $tuple/content/service/interface[@type = $repcat]/operation
    SATISFIES ($op/name="XML getPFNs (String LFN)" AND
        $op/bindhttp/@verb="GET" AND
        contains($op/allow, "http://cms.cern.ch/everybody"))
RETURN $tuple
```

Detector and physics event simulation

- Evaluate reconstruction software, assist detector design
- Estimate eventual requirements for storage and CPU,

Characteristics

- Scheduled coordinated activity, sites assigned a specific task
- Lots of data output: $O(10^8)$ events output for a given dataset
- Goal: Objective maximise throughput

Simulation use case steps:

- Generate job script
 - event type, kinematical parameters, software versions
 - estimate of max/min output file size, job CPU time
- Identify CE and SE with sufficient space for output datasets
- Move job to CE, Execute job
- Move output files to final Storage; register output data with grid (optional)
- Store job metadata in an database
 - CE & SE node, provenance data, date run, code version/release, platform

Characteristics:

- Distributed access input data:
 - output of earlier Monte Carlo data production
- Output data size typically smaller than input data set

Event reconstruction use case steps:

- Generate job script
 - Input LFNs where known, software version
- Submit job on local host to run on a remote host
 - Grid services identify best CE for job, based on distribution of physical instances of input LFNs and other criteria, i.e software version
- Replica Manager identifies PFNs for input LFNs
 - Move PFNs to designated execution site
- Run reconstruction job
- User monitors job progress on remote host from local host
- Move output data to final storage site
 - Storage site may be different from execution and submission site

Characteristics:

- Many physicists accessing same data sets
- Entire job is as fast as the slowest event analysis
- Chaotic and random access patterns
 - Input datasets not always known a priori
- Goal: minimise latency

Physics analysis use case steps:

- User queries metadata to extract LFN list for an analysis
 - input data could be reconstructed MC or detector (“real”) data
- Resource broker identifies site for job execution
- Replica manager moves data to site for job execution
- Run analysis job
- Move output data to final SE
- Register output data with grid

Analysis of a set of melanoma medical images

- Images will contain nominative and anonymous or proprietary information
 - Confidential patient and physician details
 - **Cannot replicate in a grid environment !**

Medical analysis use case steps:

- Physician queries a metadata service (Spitfire) for interesting images
 - match analysis parameters: year, melanoma type, etc ...
- Submit job to analyse images
- Image files located only on hospital DICOM servers
 - need to be replicated to grid aware storage
- All nominative information must be stripped from data prior to replication
 - Header ⇔ image mappings stored in a local metadata database ... Spitfire
- Identify CE and run job
- Output data files may be registered with Grid

Example : subsequent to an analysis, a misdiagnosis is discovered

- A melanoma is in fact malignant and not benign

Researcher who performed the analysis must inform the examining physician

- All nominative information was stripped from image files
- Researcher updates a metadata database diagnosis field
 - Update reflects new diagnosis and triggers an action
 - Header information is found an examining physician is informed
- Researcher has no access to nominative information but has communicated the analysis results

WP2 Documents

1. <http://cern.ch/grid-data-management/publications.html>

Spitfire

1. <http://cern.ch/hep-proj-spitfire>

Service Index

1. [A Unified Peer-to-Peer Database Framework for XQueries over Dynamic Distributed Content and its Application for Scalable Service Discovery](#) Wolfgang Hoschek, PhD Thesis, Technical University of Vienna (submitted), 2002.
2. [A Data Model and Query Language for Service Discovery](#) Wolfgang Hoschek, Data Grid TechReport DataGrid-02-TED-0409, April 2002.