The AMGA Metadata Catalog

Introduction and hands-on exercises

Nuno Santos

CERN

Health e-Child Tutorial

CERN(Geneve), October 10th, 2006

A R D A

Information Society and Media

Contents

- **Background and Motivation for AMGA**
- **Interface, Architecture and Implementation**
- **Metadata Replication on AMGA**
- **Deployment Examples**
- **Hands-on Exercises**

Metadata on the GRID

- **Metadata is data about data**
- **On the Grid: information about files**
  - Describe files
  - Locate files based on their contents
- **But also simplified DB access on the Grid**
  - Many Grid applications need structured data
  - Many applications require only simple schemas
    - Can be modelled as metadata
  - Main advantage: better integration with the Grid environment
    - Metadata Service is a Grid component
    - **Grid security**
    - Hide DB heterogeneity

gLite Metadata Interface

- **2004 - ARDA evaluated existing Metadata Services from HEP experiments**
  - AMI (ATLAS), RefDB (CMS), Alien Metadata Catalogue (ALICE)
  - Similar goals, similar concepts
  - Each designed for a particular application domain
    - Reuse outside intended domain difficult
  - Several technical limitations: large answers, scalability, speed, lack of flexibility
- **ARDA proposed an interface for Metadata access on the GRID**
  - Based on requirements of LHC experiments
  - But generic - not bound to a particular application domain
  - Designed jointly with the gLite/EGEE team
- **Adopted as the official EGEE Metadata Interface**

AMGA Implementation

- **ARDA developed an implementation of the EGEE interface**
  - AMGA – ARDA Metadata Grid Application
- **Began as prototype to evaluate the Metadata Interface**
  - Evaluated by community since the beginning:
    - LHCb and Ganga were early testers (more on this later)
  - Matured quickly thanks to users feedback
- **Now part of gLite middleware**
  - Official Metadata Service for EGEE
  - First release with gLite 1.5
  - Planned for inclusion on gLite 3.1 (not present on gLite 3.0)
  - Also available as standalone component
- **Expanding user community**
  - HEP, Biomed, UNOSAT…

Metadata Concepts

- **Some Concepts**
  - Metadata - List of attributes associated with entries
  - Attribute – key/value pair with type information
    - Type – The type (int, float, string,…)
    - Name/Key – The name of the attribute
    - Value - Value of an entry's attribute
  - Schema – A set of attributes
  - Collection – A set of entries associated with a schema
  - Think of schemas as tables, attributes as columns, entries as rows

Example from gLibrary

- **gLibrary is a use case developed by GILDA.**
  - Attempt to create a Multimedia Management System on the Grid
    - Images, Movies, Audio Files, Office Documents
- **Two collections presented below:**
  - /gLibrary
  - /glAudio

Collection	**/gLibrary**			
Entry Names	**Attributes**			
	FileName	**PathName**	**Type**	**Submitter**
4ffaffc8-26e7-4826-b460-3d5bf08081a4	DedicatoAte.mp3	/grid/gilda/calanducci	Audio	Tony Calanducci
00454dca-a269-4b93-8a45-c4012af05600	ardizzonelarocca_is_231005.ppt.gpg	/grid/gilda/calanducci/EGEE	EGEEDOC	Tony Calanducci

Collection	**/gLAudio**					
Entry names	**Attributes**					
	SongTitle	**Duration**	**Album**	**Genre**	**Singer**	**Format**
4ffaffc8-26e7-4826-b460-3d5bf08081a4	Dedicato A Te	00:03:27	Dedicato A Te	Pop	Le Vibrazioni	MP3

Enabling Grids for E-sciencE

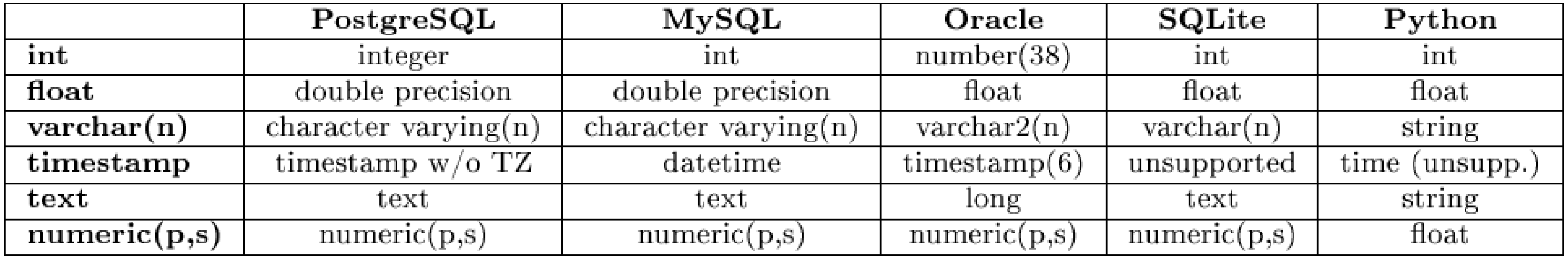
AMGA Features

- **Dynamic Schemas**
  - Schemas can be modified at runtime by client
    - Create, delete schemas
    - Add, remove attributes
- **Metadata organised as an hierarchy**
  - Collections can contain sub-collections
  - Analogy to file system:
    - Collection ⬄ Directory; Entry ⬄ File
- **Flexible Queries**
  - SQL-like query language
  - Joins between schemas
  - Example

```
selectattr /gLibrary:FileName /gLAudio:Author /gLAudio:Album
          '/gLibrary:FILE=/gLAudio:FILE and like(/gLibrary:FileName, “%.mp3")‘
```

AMGA Datatypes

- **Database systems from different vendors support different datatypes.**
  – Obstacle to portability
- **AMGA defines six standard datatypes –**
  – mapped transparently to the most appropriate type of the DB backend in use

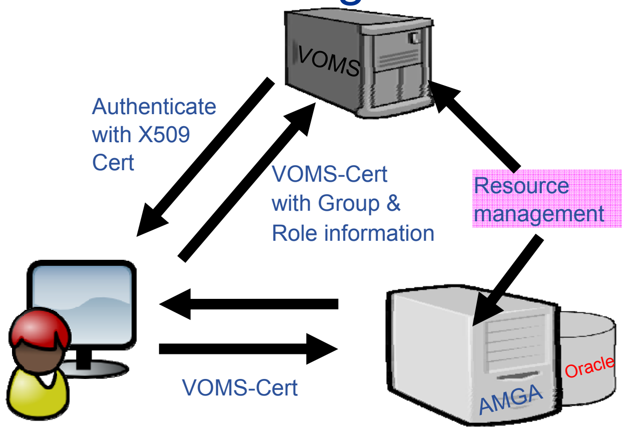


	PostgreSQL	MySQL	Oracle	SQLite	Python
int	integer	int	number(38)	int	int
float	double precision	double precision	float	float	float
varchar(n)	character varying(n)	character varying(n)	varchar2(n)	varchar(n)	string
timestamp	timestamp w/o TZ	datetime	timestamp(6)	unsupported	time (unsupp.)
text	text	text	long	text	string
numeric(p,s)	numeric(p,s)	numeric(p,s)	numeric(p,s)	numeric(p,s)	float

- **Using the above datatypes you are sure that your metadata can be easily moved to all supported back-ends**
- **If you do not care about DB portability, you can use, in principle, as entry attribute type ALL the datatypes supported by the back-end**
  – PostgreSQL Network Address type or Geometric ones

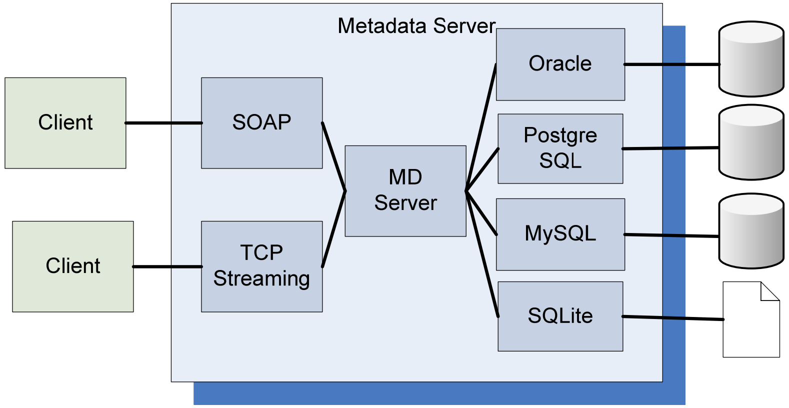
Security

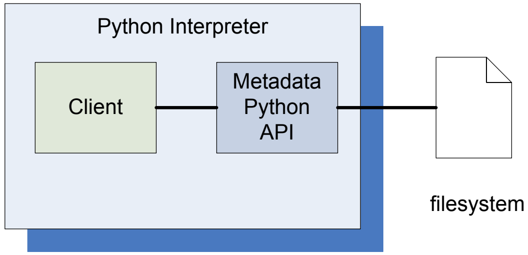
- **Secure connections – SSL**
- **Authentication based on**
  - Username/password
  - General X509 certificates
  - Grid-proxy certificates
- **Authorisation:**
  - Users/groups
  - Unix style permissions
  - ACLs – Per-collection or per-entry
  - Access control via a Virtual Organization Management System (VOMS):


VOMS
Authenticate with X509 Cert
VOMS-Cert with Group & Role information
Resource management
VOMS-Cert
AMGA
Oracle

AMGA Implementation

- **C++ Server**
  - Runs on any Linux flavour
- **Backends**
  - Oracle, MySQL, PostgreSQL, SQLite
- **Two frontends**
  - TCP Streaming
    - High performance
    - Client API for C++, Java, Python, Perl, Ruby
  - SOAP
    - Interoperability
- **Also implemented as standalone Python library**
  - Data stored on filesystem



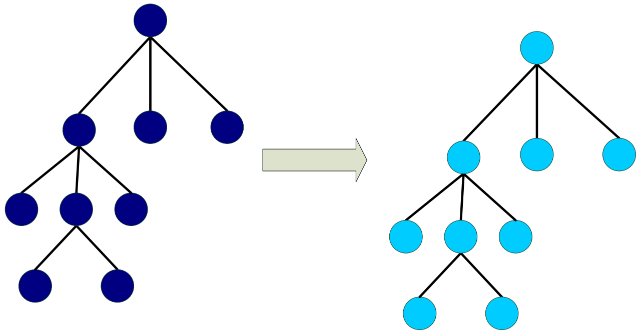


Metadata Replication

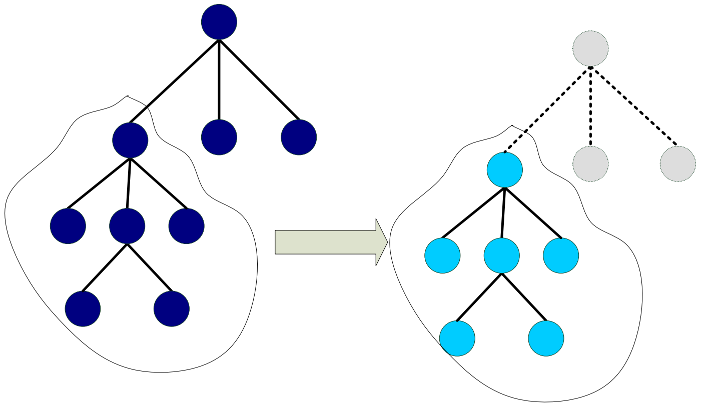
- **Motivation**
  - Scalability – Support hundreds/thousands of concurrent users
  - Geographical distribution – Hide network latency
  - Reliability – No single point of failure
  - DB Independent replication – Heterogeneous DB systems
  - Disconnected computing – Off-line access (laptops)
- **Architecture**
  - Asynchronous replication
  - Master-slave – Writes only allowed on the master
  - Replication at the application level
    - Replicate Metadata commands, not SQL → DB independence
  - Partial replication – supports replication of only sub-trees of the metadata hierarchy

Main use cases

Full replication



Partial replication

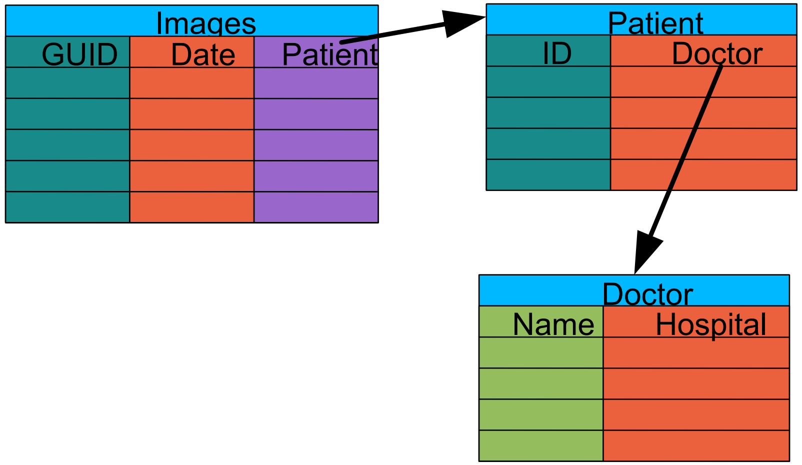


Early adopters of AMGA

- **LHCb-bookkeeping**
  - Migrated bookkeeping metadata to ARDA prototype
    - 20M entries, 15 GB
    - Large amount of static metadata
- **Ganga**
  - Job management system
    - Developed jointly by Atlas and LHCb
  - Uses AMGA for storing information about job status
    - Small amount of highly dynamic metadata

Biomed

- **Medical Data Manager – MDM**
  - Store and access medical images and associated metadata on the Grid
  - Built on top of gLite 1.5 data management system
  - Demonstrated at last EGEE conference (October 05, Pisa)
- **Strong security requirements**
  - Patient data is sensitive
  - Data must be encrypted
  - Metadata access must be restricted to authorized users
- **AMGA used as metadata server**
  - Demonstrates authentication and encrypted access
  - Used as a simplified DB



Images

GUID	Date	Patient

Patient

ID	Doctor

Doctor

Name	Hospital

- **More details at**
  - https://uimon.cern.ch/twiki/bin/view/EGEE/DMEncryptedStorage

Conclusion

- **AMGA – Metadata Service of gLite**
  - Part of gLite (but still not certified in gLite 3.0. it will be done with 3.1 release)
  - Useful for simplified DB access
  - Integrated on the Grid environment (Security)
- **Replication/Federation features**
- **Tests show good performance/scalability**
- **Already deployed by several Grid Applications**
  - LHCb, ATLAS, Biomed, …
  - GILDA applications – gLibrary

- **AMGA Web Site**

  `http://cern.ch/amga`

Any questions?

End of theory

“Hands on” to follow…

Accessing AMGA

- **We will use the TCP Streaming Front-end**
- **Programming APIs:**
  - C++ API (md_cli.h, MD_Client.h)
  - Java Client API and command line mdjavaclient.sh & mdjavacli.sh (also under Windows !!)
  - Python Client API
- **Interactive access**
  - mdcli – executes a metadata command and exits. Useful for scripts.
  - mdclient – interactive shell
- **We will use the mdclient interactive shell**

Access the Metadata Catalog

- **Copy a template of config file for the MDC:**

```
$ cp $GLITE_LOCATION/etc/mdclient.config
    $HOME/.mdclient.config
```

- **Start up the Metadata Catalog Client with**

```
$ mdclient
```

- **Once logged in, you can list the available commands, typing help.**

```
Connected to amga.ct.infn.it:8822
ARDA Metadata Server 1.2.0
Query> help
>> >help [topic]<
>> >Displays help on a command or a topic.<
>> >Valid topics are: help metadata metadata-optional directory entry group acl index
    schema sequence user view ticket commands<
```

- Commands are grouped by topic. You can get the list of valid commands for each topic, typing help [topic]
- **Example**: help entry

MDC directory related commands

- **Browse the contents of a directory**
  - **dir [path]**

  Returns the name of all subdirectories and files in the given *path* or in the current directory if not specified
- **Print the current working directory**
  - **pwd**
- **Change the current working directory**
  - **cd directory**

  Example: cd /gilda/rio
- **Directory creation**
  - ***createdir /parentdir/dir***

  Creates the directory *dir* if it does not yet exist but *parentdir* already does

  Example: createdir /gilda/rio/tcaland
- **Directory removal**
  - **rmdir path**

  Removes the directory given by path

MDC: Handling attributes

- Schema population
  - addattr dir attr type

  Adds a new attribute to the schema of a directory. Type is the name of an SQL datatype which will translated (if necessary) into a data type understood by the back end DB.

  Examples of valid datatypes are `int, float, varchar(n), timestamp, text, numeric(p,s)`

  Examples:

  ```
  addattr /gilda/merida/tcaland MovieTitle varchar(100)
  addattr /gilda/merida/tcaland Runtime int
  addattr /gilda/merida/tcaland PlotOutline text
  ```

- Attribute listing
  - listattr path

  Returns a list of all attributes of the given file/direcory

- Attribute Removal
  - removeattr dir attribute

  Removes an attribute from a directory if it is not used by any entry in the directory

MDC: managing entries

- **Entry creation**
  - **addentry entry (attribute value)+**

  Add a new entry and initializes some attributes

  Example: addentry /gilda/rio/tcaland/madagascar.mov MovieTitle Madagascar

- **Setting attribute values**
  - **setattr entry (attribute value)+**

  Sets one or more attributes of an entry to given values

  Example: setattr /gilda/rio/tcaland/madagascar.mov Runtime 86

- **Getting attribute values**
  - **getattr pattern (attribute)+**

  Returns the entries and all the attributes for every file matching pattern

  Example: getattr /gilda/rio/tcaland/*.mov Title

- **Entry deletion**
  - **rm pattern**

  Removes all entries matching pattern

  Example: rm /gilda/rio/m*.mov

Metadata Queries

- **find pattern ‘query_condition’**

Returns all entries matching pattern for which query_condition is true

Examples:

```
find /gilda/riotcaland/ ‘Runtime > 80’
find /gilda/rio/tcaland/ ‘like(MovieTitle, “Mad%”)’
find /gilda/rio/tcaland ‘like(MovieTitle, “Mad%”) AND Runtime > 80’
```

- **selectattr attr... condition**

Returns the values of given attributes for all files matching condition

Example:

```
cd /gilda/rio/tcaland
selectattr .:MovieTitle .:Runtime 'Runtime > 80'
>> >Madagascar<
>> >86<
```

Exercise:

- Log into the Metadata Catalog
- Create a directory with your surname into the /gilda/merida directory
- Add some attributes (Description (varchar(100), Value int, Comment text) to the directory just created
- Add some entries using as entry name the LFNs you uploaded and registered into the File Catalog during the DMS hands-on session
- Fill the attribute fields for the inserted entries
- Look for the entry with Value > 50