AMGA
Metadata Access on the GRID

Nuno Santos

NA4 Generic Applications Meeting
January 10th, 2006

Contents

- Background and Motivation for AMGA
- Interface, Architecture and Implementation
- Metadata Replication on AMGA
- Performance
- Deployment Examples

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

ARDA/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
  - Incorporates feedback from GridPP
- **Adopted as the official EGEE Metadata Interface**
  - Endorsed by PTF (Project Technical Forum of EGEE)

AMGA Implementation

- **ARDA developed an implementation of PTF 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 upcoming gLite 1.5
  - Also available as standalone component
- **Expanding user community**
  - HEP, Biomed, UNOSAT…
  - More on this later

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

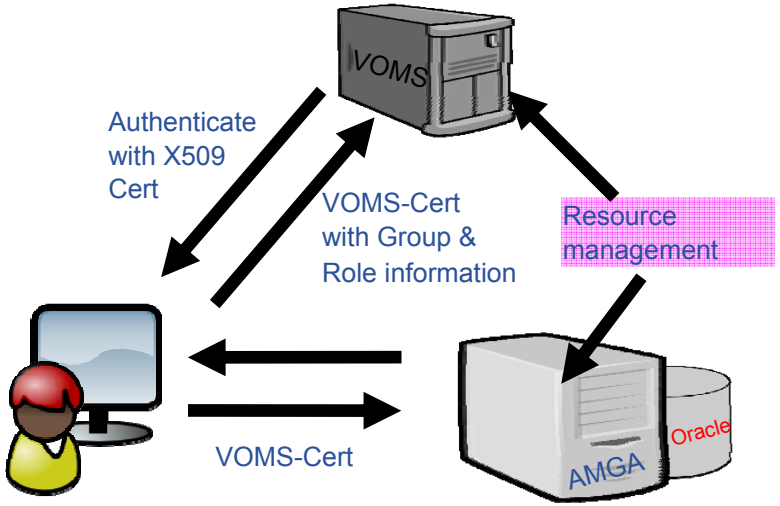
AMGA Features

- **Dynamic Schemas**
  - Schemas can be modified at runtime by client
    - Create, delete schemas
    - Add, remove attributes
- **Metadata organised as an hierarchy**
  - Schemas can contain sub-schemas
  - Analogy to file system:
    - Schema ⇔ Directory; Entry ⇔ File
- **Flexible Queries**
  - SQL-like query language
  - Joins between schemas
  - Example (adapted from Tony's presentation):

```
selectattr /DLibrary:FileName /DLAudio:Author /DLAudio:Album
          '/DLibrary:FILE=/DLAudio:FILE and like(/DLibrary:FileName, "%.mp3")'
```

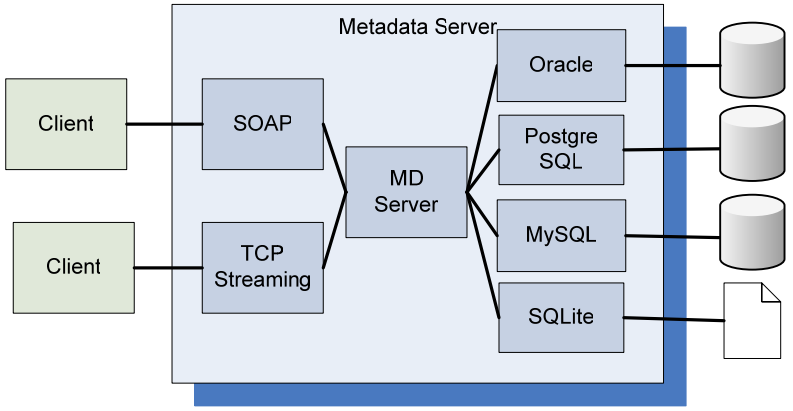
Security

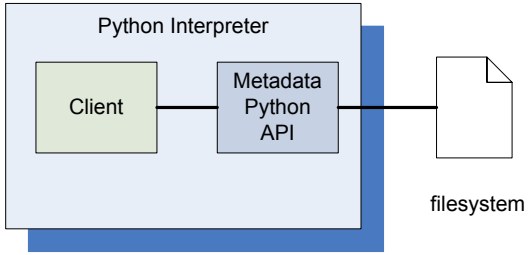
- **Unix style permissions**
- **ACLs – Per-collection or per-entry.**
- **Secure connections – SSL**
- **Client Authentication based on**
  - Username/password
  - General X509 certificates
  - Grid-proxy certificates
- **Access control via a Virtual Organization Management System (VOMS):**



AMGA Implementation

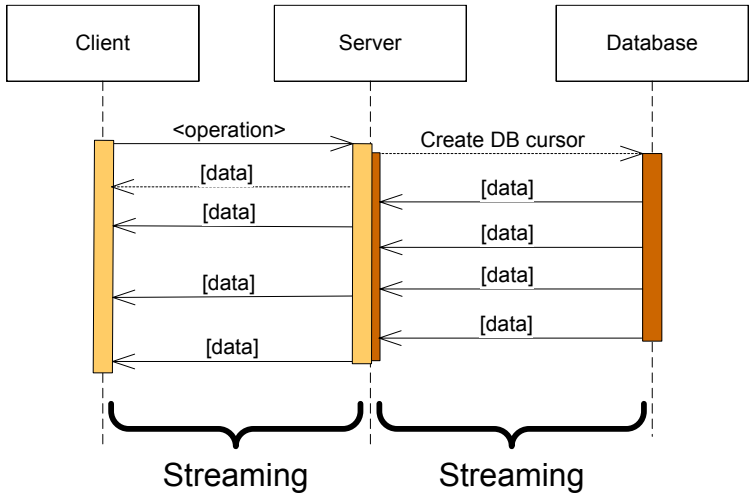
- **C++ multiprocess server**
  - Runs on any Linux flavour
- **Backends**
  - Oracle, MySQL, PostgreSQL, SQLite
- **Two frontends**
  - TCP Streaming
    - High performance
    - Client API for C++, Java, Python, Perl, Ruby
  - SOAP
    - Interoperability
- **Also implemented as standalone Python library**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





Architecture
TCP-Streaming frontend

- **Designed for scalability**
  - Asynchronous operation
    - Reading from DB and sending data to client
  - Response sent to client in chunks
    - No limit on the maximum response size
- **Example: TCP Streaming**
  - Text based protocol (like SMTP, POP3,…)
  - Response streamed to client



Client: `listattr entry`

Server:
```
0
entry
value1
value2
…
<EOT>
```

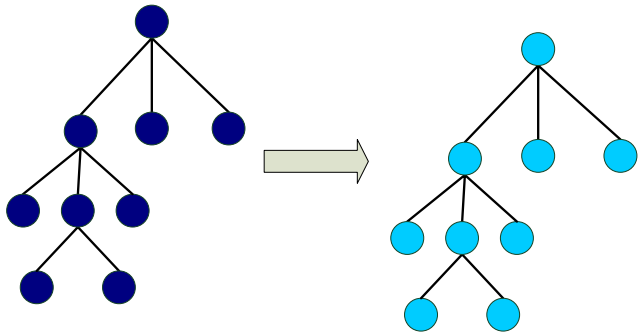
Metadata Replication

- **Currently working on replication/federation mechanisms for AMGA**
- **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

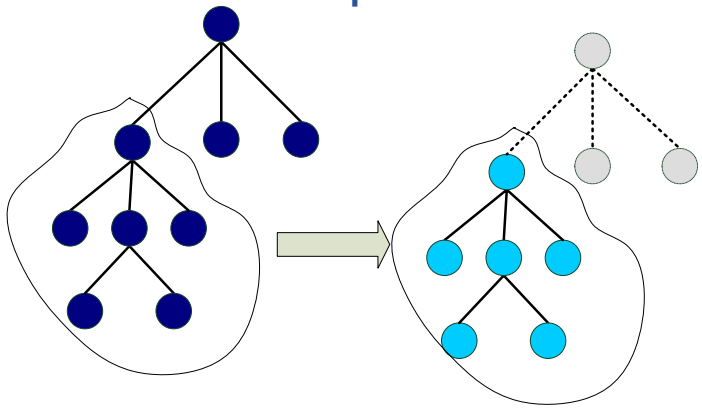
Metadata Replication

Some use cases

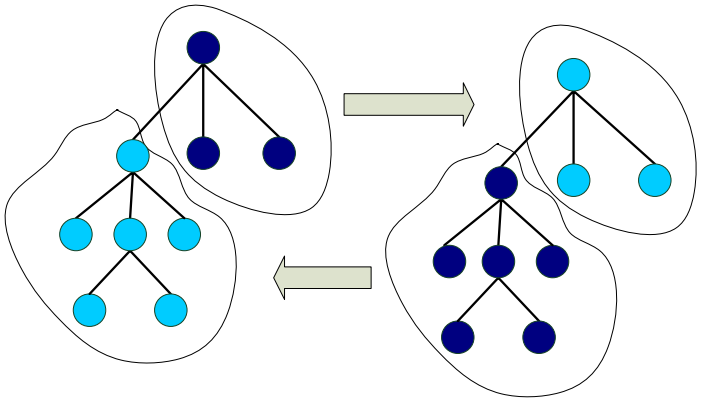
Full replication



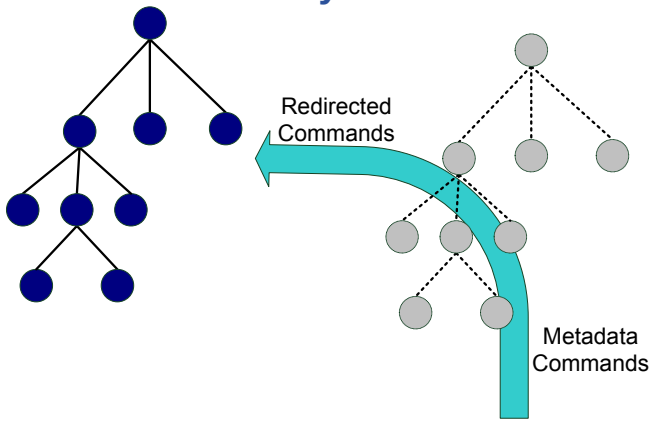
Partial replication



Federation



Proxy



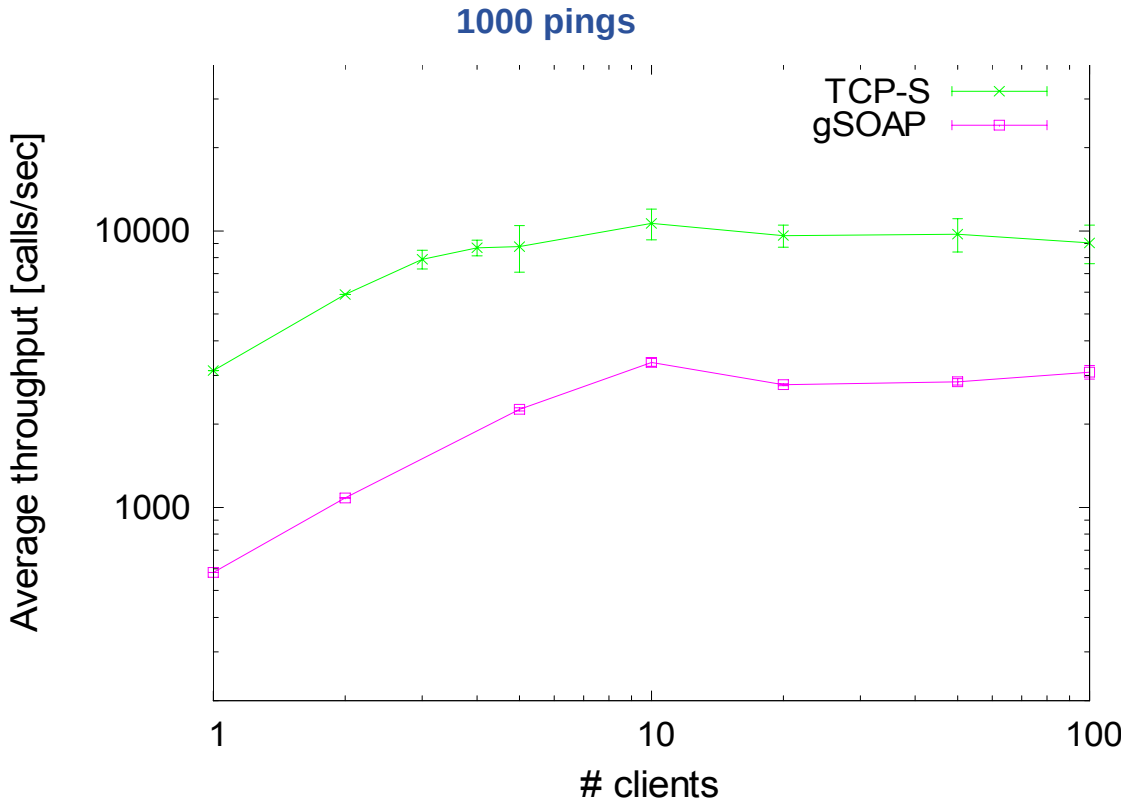
Metadata Replication

- **Current Status**
  - Implementation under way
  - Integrated on AMGA, no external software needed
  - Early prototype. Basic functionality working in a single slave configuration
    - Initial slave synchronization
    - Update propagation
- **Future Plans**
  - More development and testing needed
  - Working prototype expected during first quarter 2006.

Benchmark Study

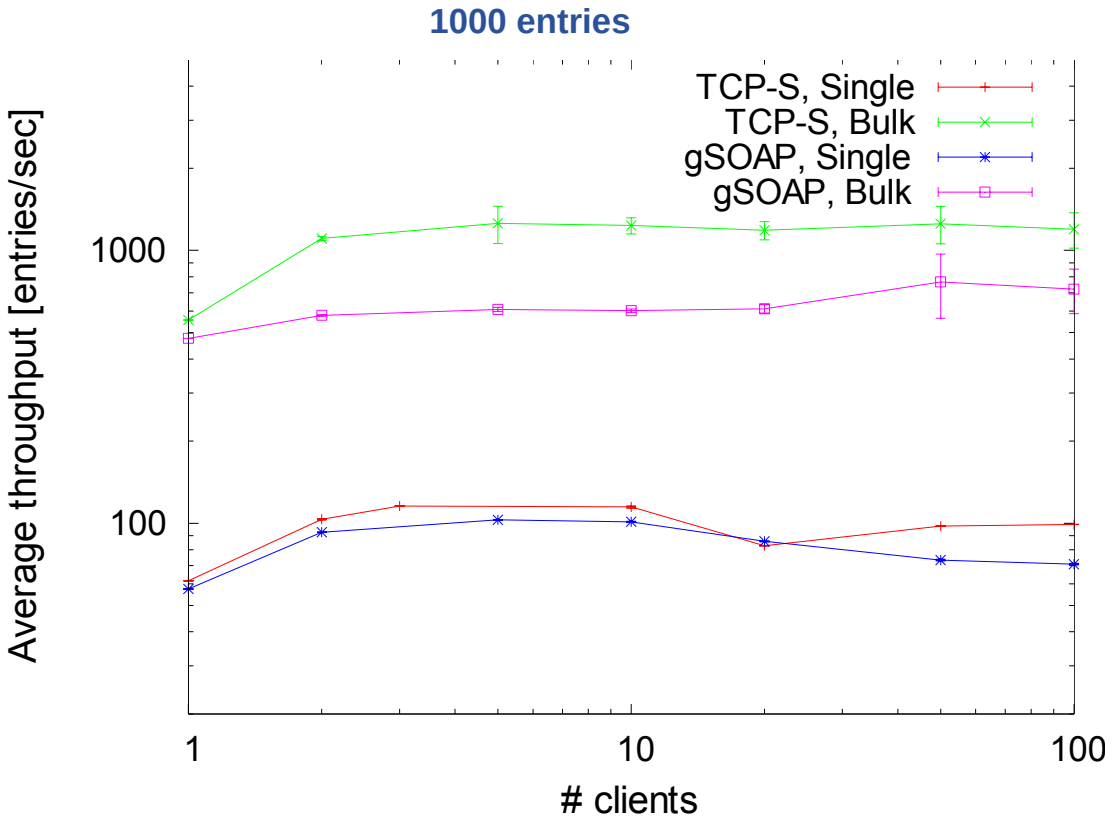
- **Objectives of Benchmark Study:**
  - Assess the scalability of AMGA
  - Compare the TCP-Streaming to the SOAP frontend
- **Protocols**
  - TCP-S – TCP Streaming, C++ client.
  - SOAP - gSoap client (C++)
- **Operations**
  - ping – A null RPC
  - get – Gets all attributes of an entry
  - get (bulk) – Gets all attributes of several entries in a single operation
- **Entries**
  - 60 attributes (ints, floats and strings), 700 bytes on average
- **Connections kept open between requests**
  - HTTP Keepalive for the SOAP frontend (gSoap feature)
- **Test Environment**
  - LAN, 100 MBits switched Ethernet
  - Single client machine
    - Simulates up to 100 concurrent clients, bottleneck was always on server

Scalability – Pings



- **Around 10.000 pings/s with TCP-S**
- **AMGA scales nicely up to 100 concurrent clients**
  - No throughput degradation

Scalability – get attributes



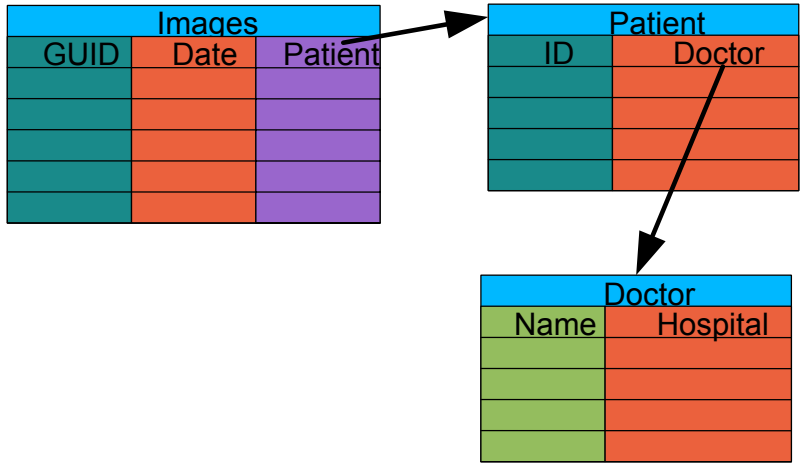
- **Up to 1.000 entries/sec (with bulk operations)**
- **Once again, server throughput does not degrade up 100 clients**

Early adopters of AMGA

- **LHCb-bookkeeping**
  - Migrated bookkeeping metadata to ARDA prototype
    - 20M entries, 15 GB
    - Large amount of static metadata
  - Feedback valuable in improving interface and fixing bugs
  - AMGA showing good scalability
- **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



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

Conclusion

- **AMGA – Metadata Service of gLite**
  - Part of gLite 1.5
  - Useful for simplified DB access
  - Integrated on the Grid environment (Security)
- **Replication/Federation under development**
- **Tests show good performance/scalability**
- **Already deployed by several Grid Applications**
  - LHCb, ATLAS, Biomed, …
  - DLibrary (next presentation)

- **AMGA Web Site**

  `http://project-arda-dev.web.cern.ch/project-arda-dev/metadata/`