

# AMGA - Metadata Catalogue Service

**Dr. Marco Fargetta**  
*Marco.Fargetta@ct.infn.it*  
**INFN, Italy**

***ACGRID-II school***



# Outline

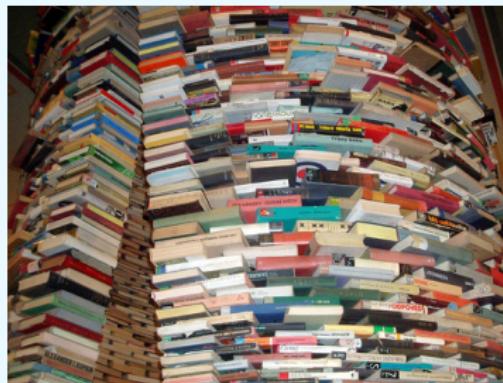
- 1 Introduction
- 2 AMGA
- 3 Implementation
- 4 Interface and Commands
- 5 AMGA in Grid applications
- 6 Conclusion



# Introduction

# Why Grid Needs MetaData

- Grids allow to save millions of files spread over several storage sites

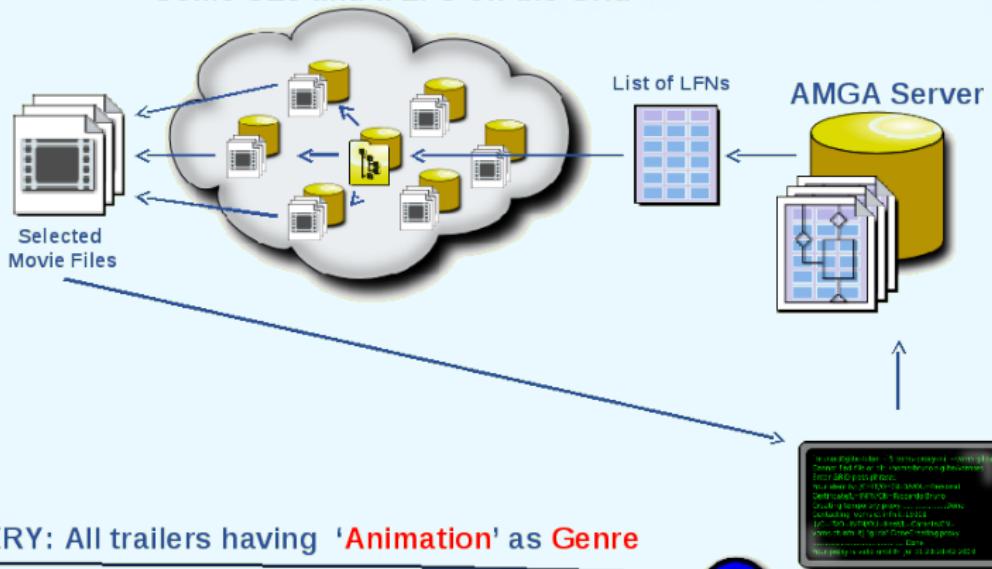


# Why Grid Needs MetaData

- Grids allow to save millions of files spread over several storage sites
- Users and applications need an efficient mechanism
  - to describe files
  - to locate files based on their contents
- This is achieved by
  - associating descriptive attributes to files
    - **Metadata is data about data**
  - answering user queries against the associated information

# General MetaData Scenario

Some SEs and a LFC on the Grid



QUERY: All trailers having 'Animation' as Genre

## Example: Movie Trailers

- Movie trailers are saved in Grid
- We want to add metadata to describe movie content
- Possible MetaData could be:
  - Title: varchar
  - Runtime: int
  - Cast: varchar
  - LFN: varchar
- A metadata catalogue will be the repository of the movies metadata and allow to find movies satisfying users queries

# Example: Movie Trailer

Schema

Attributes

<b>Entry names</b>	<b>Title</b>	<b>Ru</b>	<b>Cast</b>	<b>LFN</b>
8c3315c1-811f-4823-a778-60a203439689	My Best Friend's	80	Julia Roberts	lfn:/grid/gilda/movies/mybfwed.avi
51a1857a-fd21-4b2c-aa74-4c53ee64846a	Spider-man 2	120	Kirsten Dunst	lfn:/grid/gilda/movies/spiderman2.avi
401e6df4-c1be-4822-958c-ce3eb5c54fc8	The God Father	113	Al pacino	lfn:/grid/gilda/movies/godfather.avi

Entry

Collection

# AMGA

# AMGA Metadata Catalogue

- AMGA: Arda Metadata Grid Application
- It is the Metadata Service for gLite middleware
  - available for other environments
  - Since *data* in gLite means files, AMGA was originally designed to manage metadata for Grid files!
- Provide a complete but easy to use interface for all user
- Designed with scalability in mind in order to deal with large number of entries
- Support for Grid security mechanisms
- Flexible with support to dynamic schemas in order to serve several application domains

# AMGA Concepts

- Entries
  - Representation of real world entities which we are attaching metadata to describing the
- Attributes
  - 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
- Metadata - List of Attributes (including their values) associated with entries

# AMGA Analogies

- AMGA vs RDBMS:
  - Schema  $\Leftrightarrow$  Table schema
  - Collection  $\Leftrightarrow$  Data table
  - Attribute  $\Leftrightarrow$  Schema columns
  - Entry  $\Leftrightarrow$  Table record
- AMGA vs File System
  - Collection  $\Leftrightarrow$  Directory
  - Entry  $\Leftrightarrow$  File

# Features

- Dynamic Schemas
  - Schemas can be modified at runtime by client
    - Create, delete schemas
    - Add, remove attributes
- AMGA collections are hierarchical organised
  - Collections can contain sub-collections
  - Sub-collections can inherit/extend parent collection schema
- Flexible Queries
  - Support for SQL-like query language
- Support for Views, Constraints, Indexes

# AMGA Security

- Unix style permissions: users and groups
- ACLs: Per-collection or per-entry (table row)
  - Secure client/server connections - SSL
- Client Authentication based on
  - Username/password
  - General X509 certificates (DN based)
  - Grid-proxy certificates (DN based)
- VOMS support:
  - VO attribute maps to defined AMGA user
  - VOMS Role maps to defined AMGA user
  - VOMS Group maps to defined AMGA group

# MetaData Replication

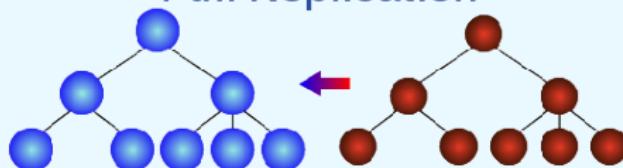
- AMGA provides replication/federation mechanisms for:
  - Scalability
    - Support of thousand concurrent users
  - Geographical distribution
    - Hide network latency
  - Reliability
    - No single point of failure
  - Disconnected computing
    - Off-line access (laptops)

# Replication Architecture

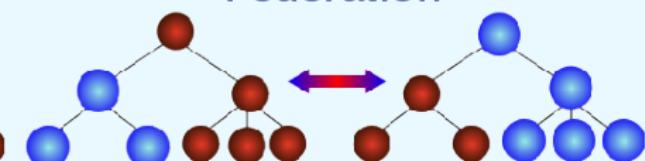
- Asynchronous replication
- Master-slave: writes only allowed on the master
- Application level replication
  - Replicate Metadata commands
- Partial replication
  - supports replication of sub-trees
- DB Independent replication

# Replication Models

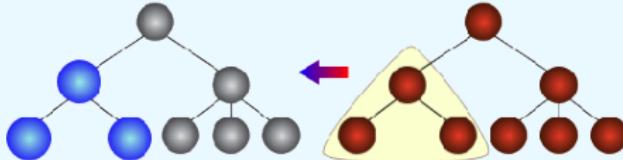
Full Replication



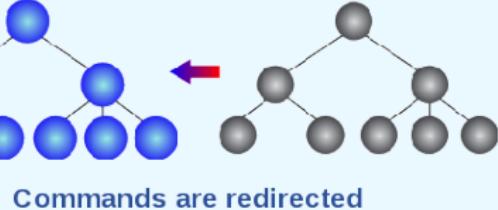
Federation



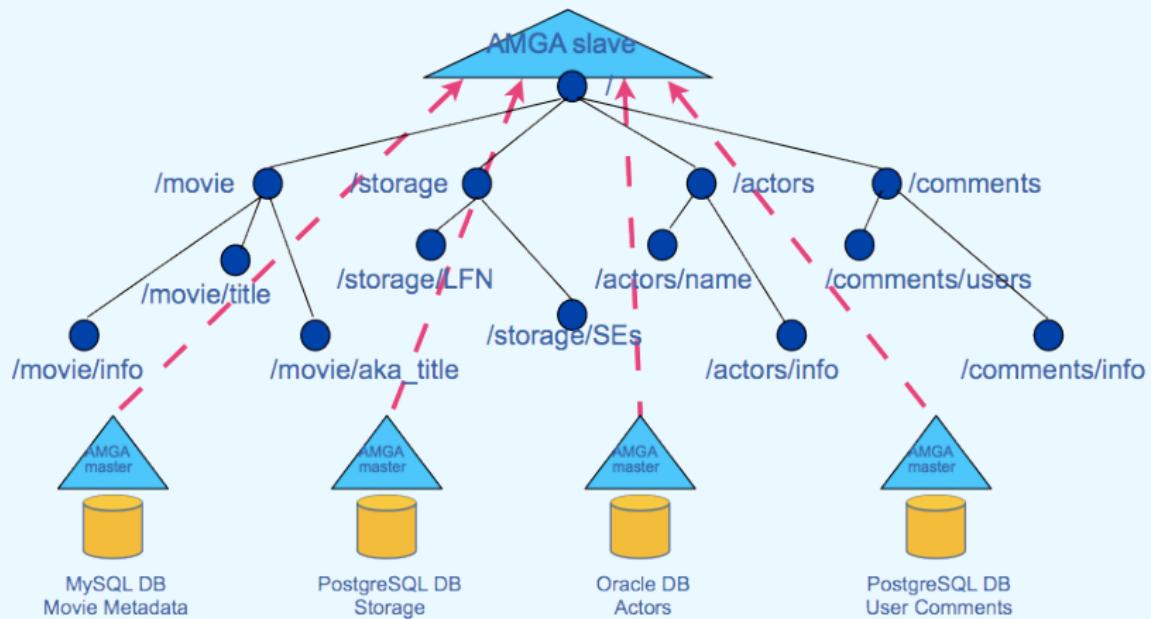
Partial Replication



Proxy



# DB Access and Replication



# Existing DB access

- Since AMGA 1.2.10, a new import feature allows to access existing DB table
- Once imported into AMGA the tables from one or more DBs you want to access through AMGA, you can exploit many of the features brought to you by AMGA for your existing tables
- Advantages:
  - your db tables can be accessed by grid users/applications, using grid authentication (VOMS proxies)/authorisation with ACLs
  - exploiting AMGA federation features you can access several databases together from the Grid

# Native SQL syntax Support

- Goal
  - To implement native SQL query processing functionality in AMGA
- Reason
  - A lot of requests from user communities
    - take advantage of their SQL expertise
    - ease the work needed to port existing SQL DB application to the Grid with AMGA
  - Complement the exiting AMGA metadata query language
- SQL-92 Entry Level direct data statements
  -

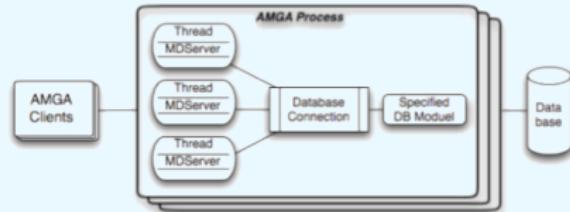
# Implementation

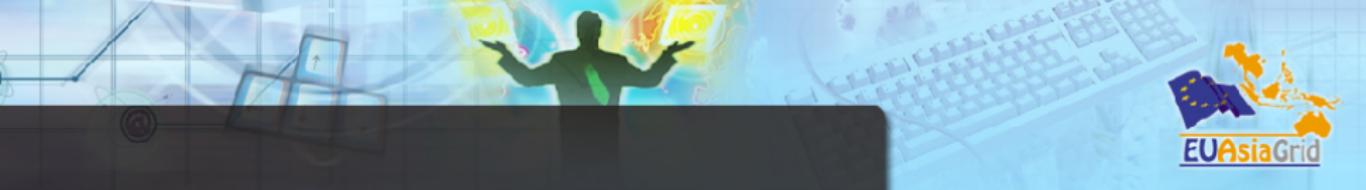
# Multi-Threading Server

- Classic AMGA server was implemented as a multi-process daemon
  - each process with its own DB connection and take care of one connected client
  - number of listening processes configurable in `amgad.config`
- A DB connection per client is required with related scalability problems
  - db connections are very expensive resources
- From version 1.9 a new multi-threaded AMGA server is available
  - each process contains multiple threads sharing a db connection

# Implementation

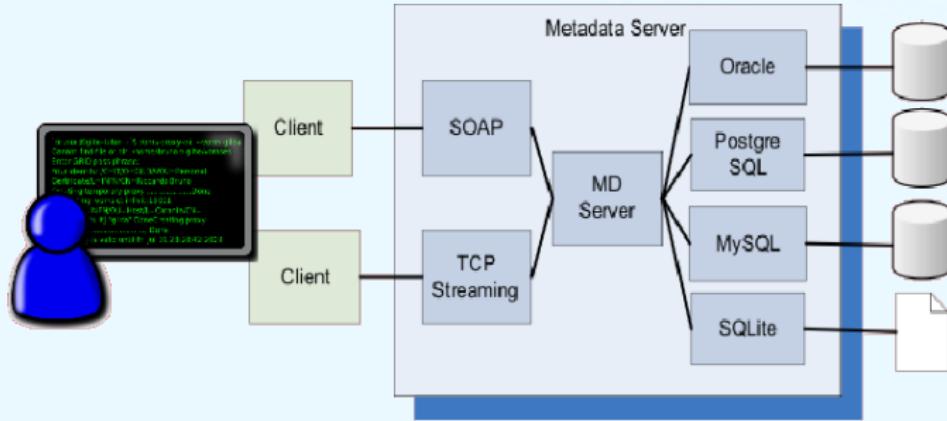
- Thread pool
  - Pre-forked threads for each server
  - configurable number in the amgad.config
- DB Connection sharing
  - all threads belonging to the same process share the same DB connection
- Architecture
  - using Pthread library
  - each thread has its own MDServer instance





# Interface and Commands

# Interacting with AMGA



- Users may access AMGA by two frontends
  - Streaming front end (TCP) ⇒ `amgad` daemon
    - CLI interactive session: `mdclient mdjavaclient`
    - CLI single command: `mdcli`
    - APIs (C++, Java, Python, Perl, PHP)
  - SOAP frontend (WSDL) ⇒ `mdsoapserver` daemon

# AMGA Datatypes

AMGA	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(unsupported)
text	text	text	long	text	string
numeric(p,s)	numeric(p,s)	numeric(p,s)	numeric(p,s)	numeric(p,s)	float

- The above datatypes allows metadata to be easily moved to all supported backends
- If you do not care about DB portability, you can use, any datatypes supported by the back-end
  - Are Excluded Oracle MySQL and PostgreSQL binary types (BLOBS)

# mdcli/mdclient Commands

- Both commands allow to connect AMGA server
  - `mdcli` executes a single command
  - `mdclient` open an interactive session
- The file `mdclient.config` contains the connection parameters
  - A template available in the directory:  
`/opt/glite/etc/mdclient.config`
  - Standard location is the home directory
    - The file may start with a dot

# mdclient.config example

## mdclient.config

```
Host = grid-test-33.trigrid.it
Port = 8822
Login = NULL
PermissionMask = rwx
GroupMask = r-x
Home = /
UseSSL = try
AuthenticateWithCertificate = 1
UseGridProxy = 1
TrustedCertDir = /etc/grid-security/certificates
CertFile=/home/marco/.globus/usercert.pem
KeyFile=/home/marco/.globus/userkey.pem
```

# Basic AMGA commands

- Create a collection
  - `createdir <path>/<collection_name>`
- Associate a schema to the collection
  - `addattr <path>/<collection_name> <attr_name> <attr_type> [<attr_name> <attr_type>]`
- List Attributes
  - `listattr <path>/<collection_name>`
- Remove Attributes
  - `removeattr <path>/<collection_name> <attr_name>`
- Rename Attributes
  - `renameattr <path>/<collection_name> <attr_name>`



# Basic AMGA commands



- Add entries and attribute values
  - addentry <path>/<entry\_name> <attr\_name><attr\_value> [<attr\_name> <attr\_value>]
- Set an attribute value
  - setattr <path>/<entry\_name> <attr\_name><attr\_value> [<attr\_name> <attr\_value>]
- List entries
  - listentries <path>/<collection\_name>
- Get an attribute value
  - getattr <path>/<entry\_name> <attr\_name>
- Get help for the commands
  - help [topic|command]

# Advanced MetaData query

find pattern 'query'

```
Query> find *.avi 'Duration > 10'  
>> madagascar.avi  
>> moulinrouge.avi
```

selectattr <attrib> ... 'query'

```
Query> selectattr trailers:Title 'like(trailers:Cast , ''%Kidman%'')'  
>> moulin rouge!
```

# SQL Support

- Recognised SQL statements
  - SELECT, INSERT, UPDATE, DELETE
- INSERT statement automatically generates a unique ID as entry name

## SQL examples

```
Query> SELECT Title FROM trailers WHERE trailers.Duration > 10
>> trailers.Title
>> madagascar
>> moulin rouge!
>> madagascar
Query> SELECT trailers.Title FROM trailers, trailers/italian WHERE tr
>> trailers.Title
>> madagascar
>> madagascar
Query>
```

# ACLs

- AMGA allow users to define ACLs for collection and entry
  - Commands: acl\_show, acl\_add and acl\_remove

## acl\_show

```
Query> acl_show trailers
>> gilda rwx
>> gilda:users rwx
>> system:anyuser rx
```



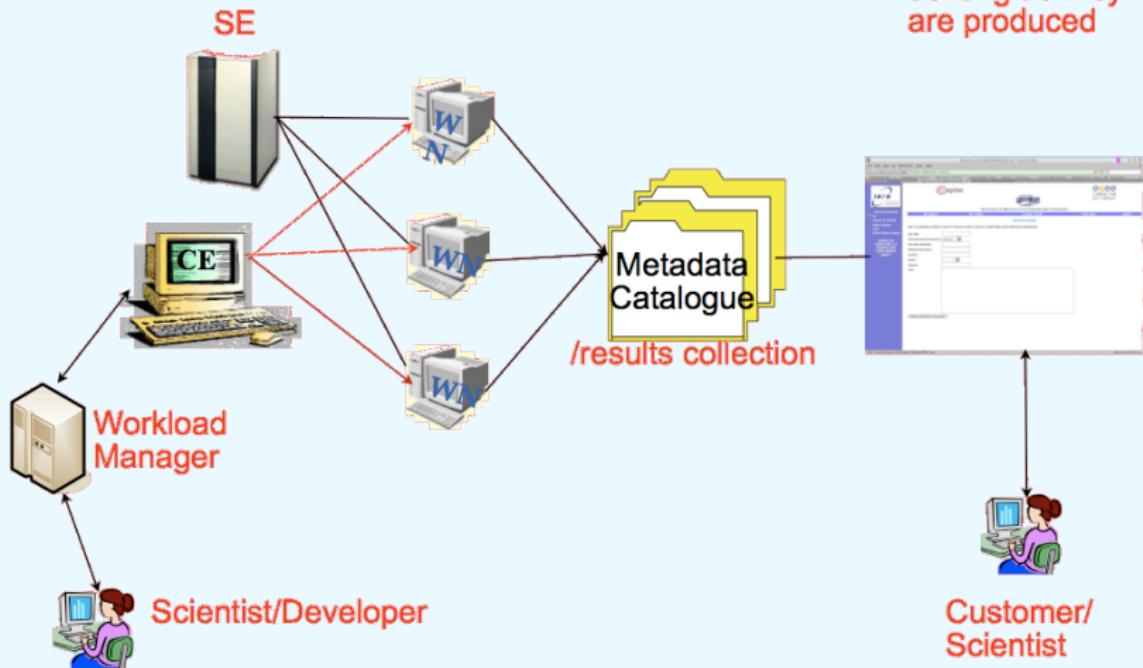
# AMGA in Grid applications

# Jobs with AMGA

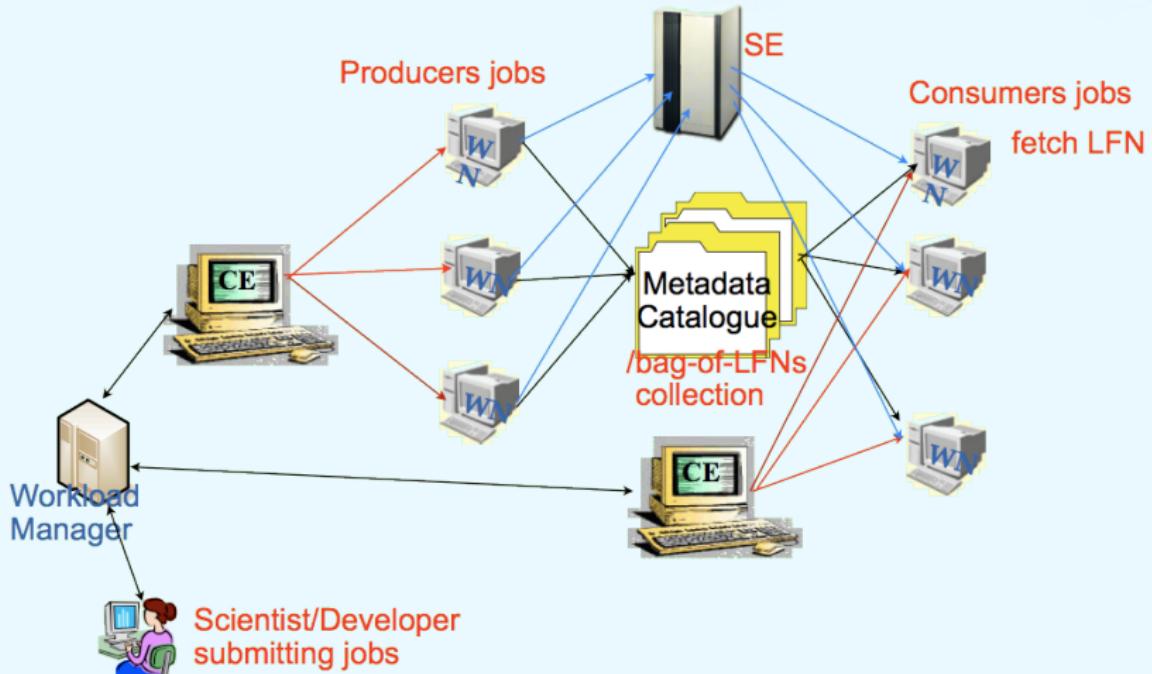
- Since AMGA supports Grid Proxies, jobs may access to any AMGA server
  - An `mdclient.config` should be available to the job
- Usually, a Job Script uses `mdcli` client applications to get/set metadata

# Monitoring of Running Application

showing results  
as long as they  
are produced



# Exchange data among peers



# Conclusion

# Conclusion

- AMGA - Metadata Service of gLite
  - Part of gLite 3.1
  - Can be used with other middleware platforms
  - Useful to realize simple Relational Schemas or add metadata information to Grid Files
- Features:
  - Replication/Federation
  - Importing existing databases
  - SQL support
  - Security (SSH, X509,  
G.Proxyies,VOMS,users/groups,ACLs)
  - Access by APIs / client Applications / SOAP

# References

- AMGA Web Site

<http://cern.ch/amga>

- AMGA Tutorials

<https://grid.ct.infn.it/twiki/bin/view/GILDA/UserTutorials>

[http://www.euasiagrid.org/wiki/index.php/The\\_AMGA\\_Metadata\\_Service](http://www.euasiagrid.org/wiki/index.php/The_AMGA_Metadata_Service)

# Questions!!!

