Metadata in the GRID

Birger Koblitz
GAG – Meeting August 6th, 2004

Overview

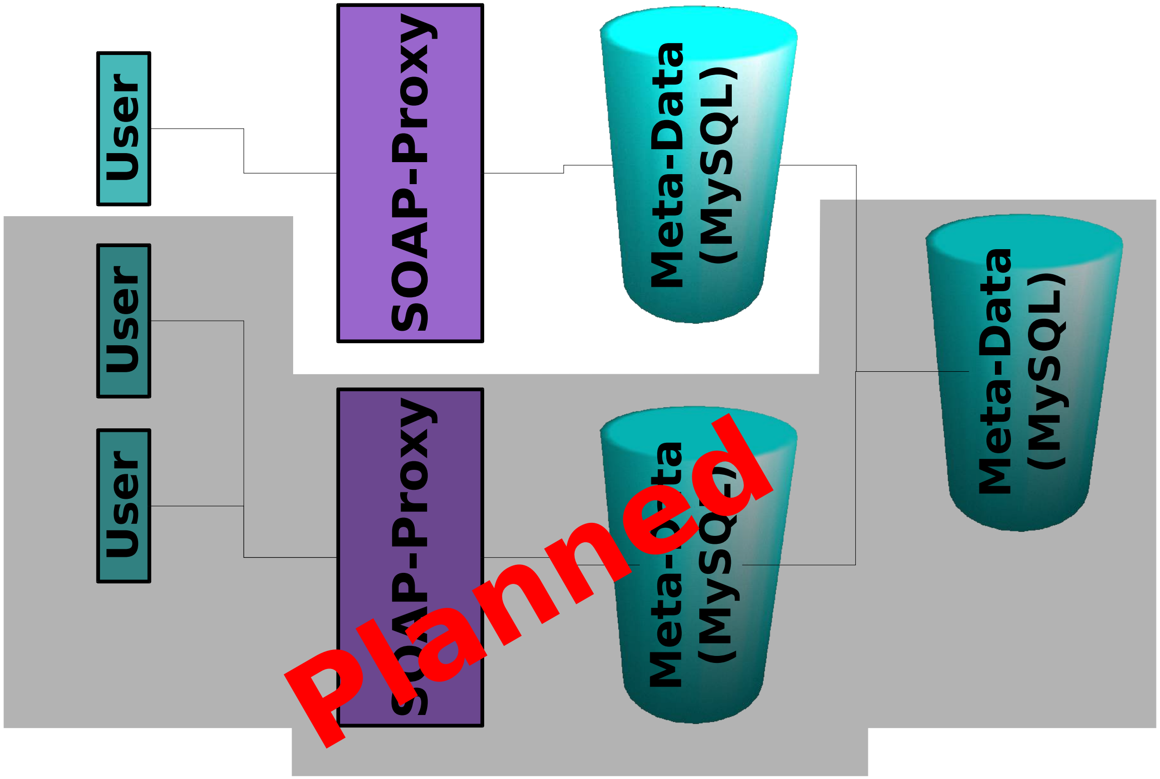
- Metadata solutions by experiments
  - AMI
  - RefDB
  - Alien/Glite
- A generic definition of metadata
- The POSIX metadata interface
- A prototype metadata catalogue
- Lessons learned

Atlas Metadata: AMI

Atlas Metadata-Catalogue, contains File-Metadata:

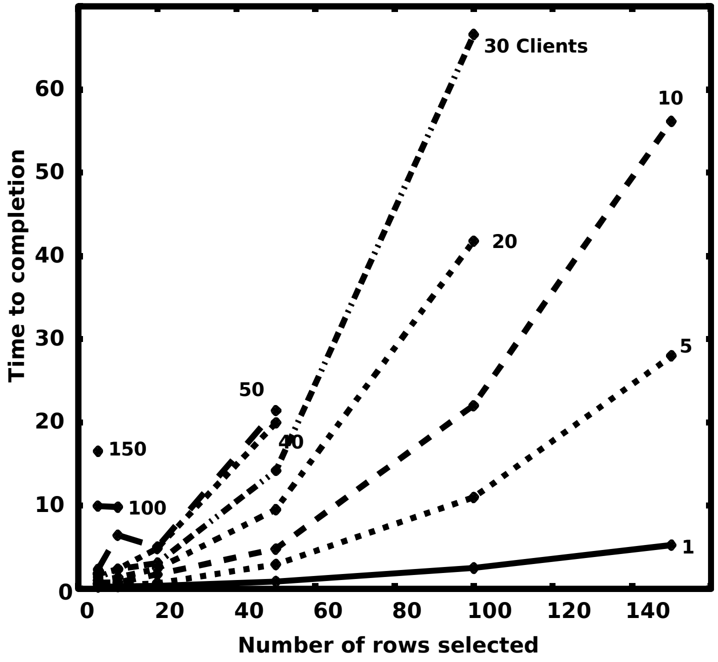
- Simulation/ Reconstruction-Version
- File-Content: Eventtypes

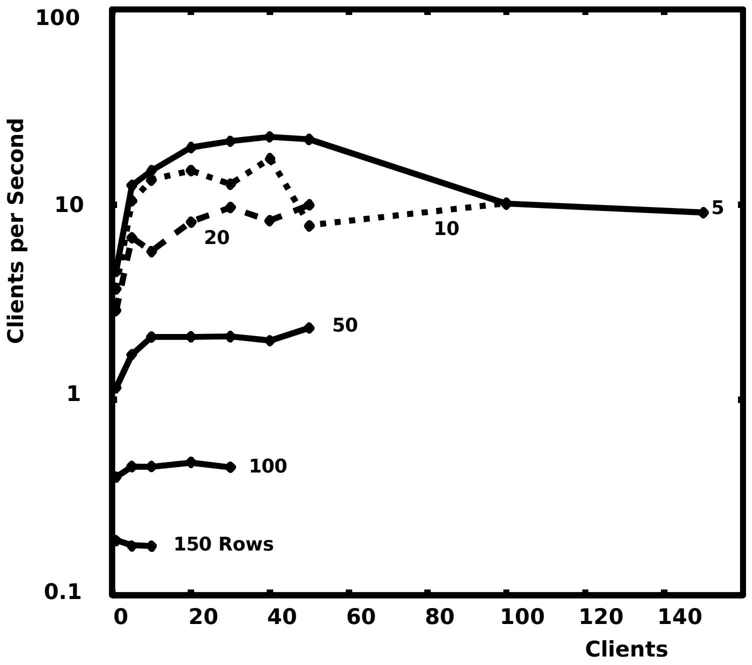
Not a File catalogue!



- SOAP-Proxy (in Java) frontend to hierachical databases (institute→collaboration)
- Proxy allows database schema evolution: Schema defined by database admins
- SOAP allows automatic code generation for client

AMI Tested





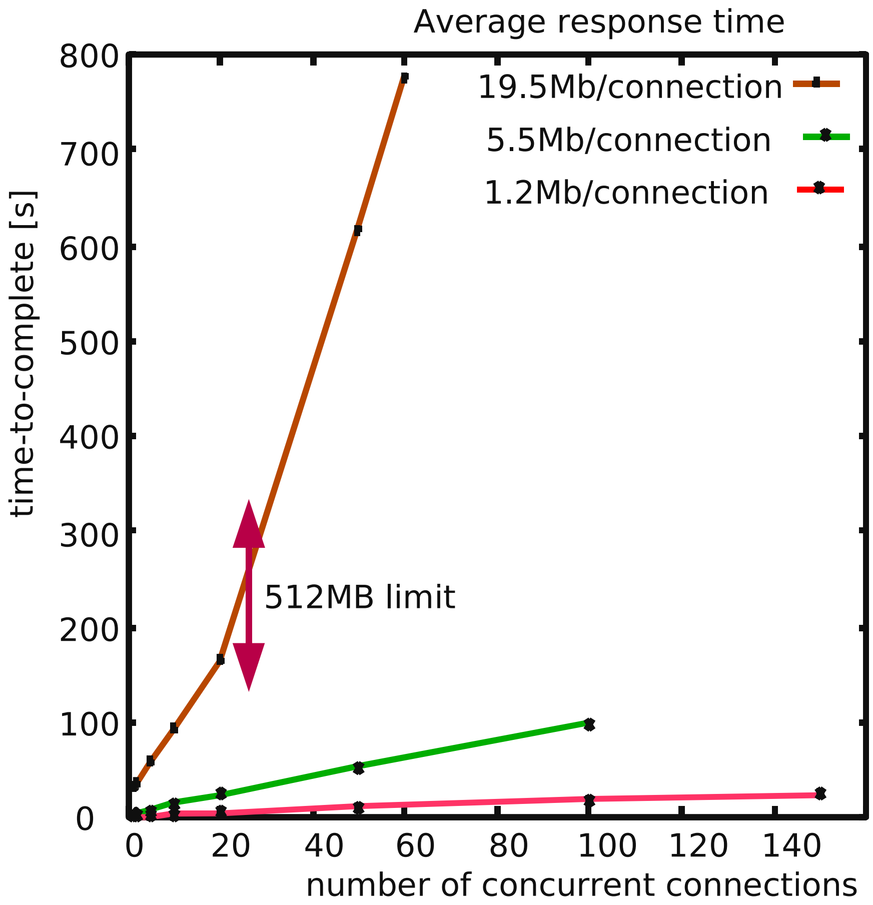
Many problems discovered:

- Large overhead (factor 10!) of traffic to DB due to schema independent tables
- SOAP blows up data transferred (factor 5-10)
- Server out of memory for large responses
- Proxy wants to be clever and reimplements DB funcionality, clashes with fact that WS are stateless

CMS: RefDB

RefDB is PHP front-end to MySQL DB:

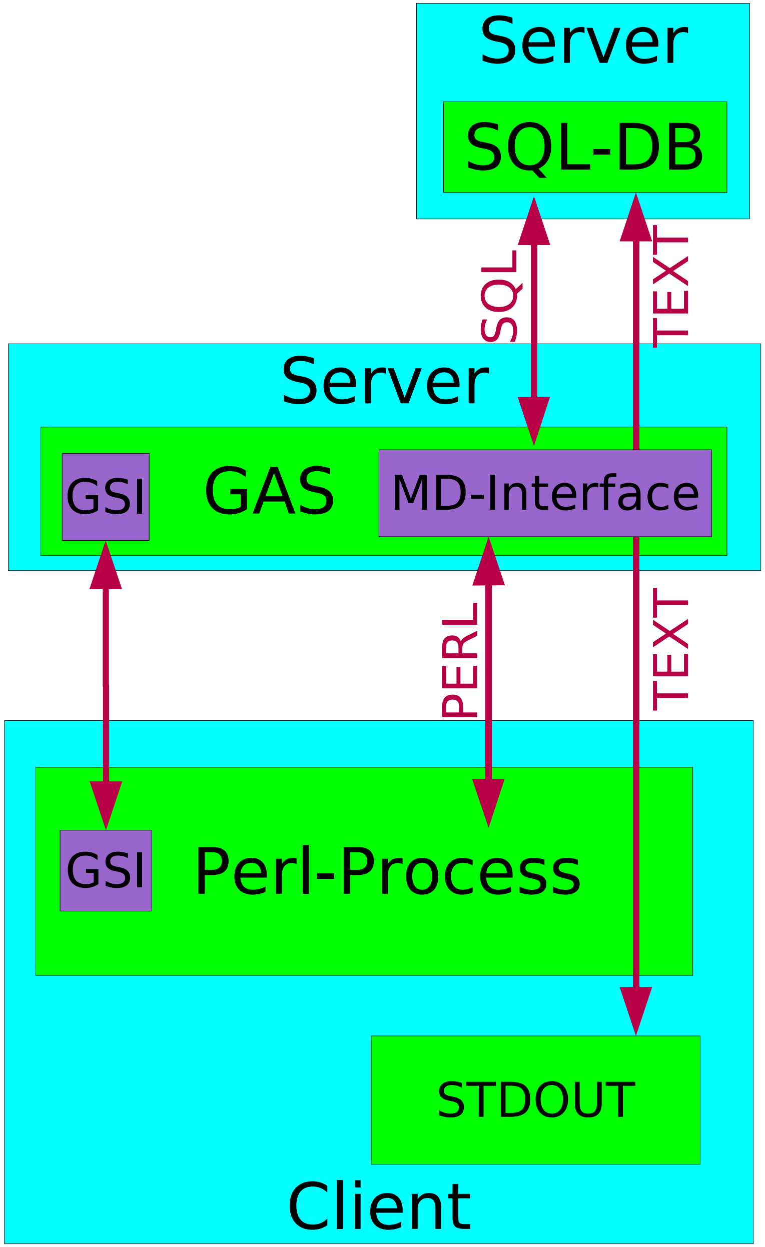
- Data retrieved via wget/http-get
- Performance limited by Data-Size not Processes



gLite (Alien)

gLite stores Metadata in additional Tables in File-Catalogue:

- User uploads SQL table description into File-Catalogue
- User associates directory with table
- User fills in table on a per file basis
- Access through glite-shell, perl scripts



Glite studied

Glite metadata catalogue simple and efficient:

- Searches through 100k entries in ½ minute
- No problem with memory due to streaming

Glite centred on production, several features missing:

- No schema evolution
- No API, not clear how to exchange data
- No copying of metadata, especially not into FS
- ARDA was first user, many bugs reported in Bug-Tracker
- Currently Denial Of Service possible because users share tables
- Currently no Data-Encryption

POSIX Metadata

POSIX defines extended attributes (Metadata) for files:

- Key-Value pairs associated with a file
  - Key: \0-terminated string
  - Value: Binary data of arbitrary length
- Copying a file copies metadata
- Metadata can be attached to directories (no inheritance)
- Metadata attached to inode (security)

Extended attributes are now widely used (NTFS, NFS, EXT2/3 SCL3, ReiserFS, JFS, XFS)
Uses with Namespaces for ACLs

Metadata searches not defined yet (No FS-Impl.):

- Windows Longhorn (2005)
- ReiserFS 5

GRID Metadata

Adjust definition in GRID context:

- Metadata attached to LFN
  - ➔ LFN is entry point to File-Catalogue, attached Metadata can be easily searched
- Files without LFN: GUID in special dirs
  - ➔ Otherwise problems with global searches
- Metadata for directories should provide default schemas/values for files
  - ➔ Easy schema copying
- Restrict values to ASCII strings
  - ➔ Backend is unknown: FileSystem/DB
- Need to define ways how to search for Metadata: Search restricted to (sub-)directories
  - ➔ Sallows hierachical databases, applicable for FS

Metadata API

POSIX defines the following commands:

-
  ```
  ssize_t getxattr(const char *path, const char *key,
                   void *value, size_t size);
  ```
  Returns value of key

-
  ```
  int setxattr(const char *path, const char *key,
               const void *value, size_t size, int flags)
  ```
  Sets key to value

-
  ```
  ssize_t listattr(const char *path, char *list, size_t size)
  ```
  Returns keys as \0-terminated strings

-
  ```
  int removeattr(const char path, const char *key)
  ```
  Removes a key

There could be a variant of the `setxattr` command be defined which takes a type as argument

In Addition the following command is required:

-
  ```
  size_t findfiles(const char *pattern, const char *query,
                   char *list, size_t size)
  ```
  Returns list of files matching pattern and query on keys

Metadata Protocol

The following protocol is proposed which clients talk to servers via sockets:

- Use plain text (ASCII)
- Query consists of one line of command
- Response returns 1 line of return status (OK/Error) and result line by line
- Result is in ASCII, user needs to encode/decode
- Commands are:

• `getattr file key`	Returns value of key
• `setattr file key value [type]`	Sets key to value
• `listattr file`	Returns keys line-by-line
• `removeattr file key`	Removes a key
• `find pattern query`	Returns list of files matching pattern and query on keys

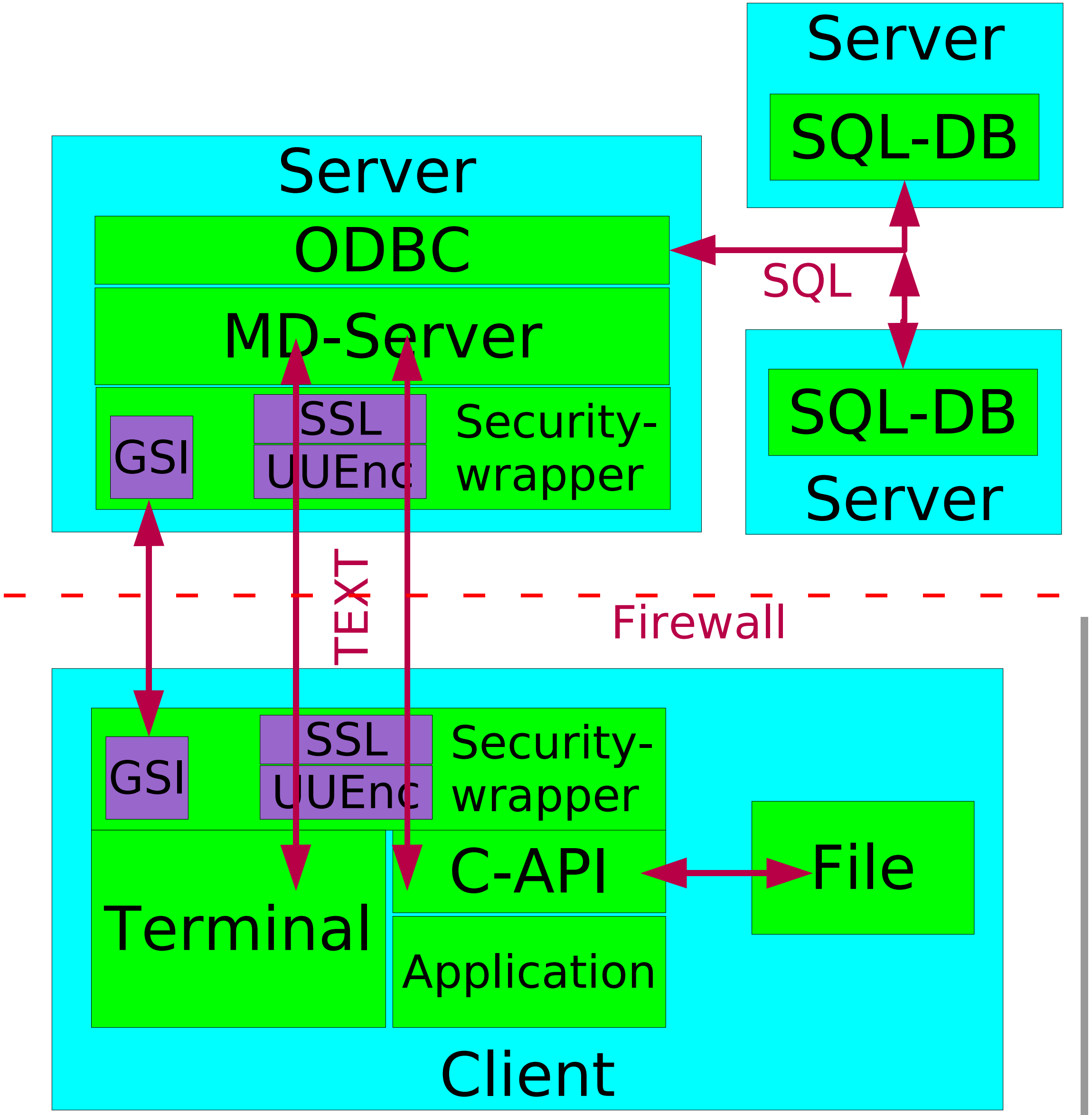
Focus is on Client: Needs to Encode/Decode data, do schema evolution

Prototype Implementation

Prototype for Reality Check:

- GSI for authentication
- Response streamed
  - Timeout problem better
  - Out of Memory solved
- Queries and commands UUEncoded and SSL encrypted (response optional)
- Backend: PostgreSQL
  - Distribution of Dbs
  - Transaction safe

C-API and distributed Databases not yet implemented



Example Session

```
koblitz@pcardabk:~/mi$ ./mdterm
Connected to DB
Query> getattr /home/koblitz/a gen
   >select table_name from masterindex where directory='/home/koblitz';<
   >select gen from dir1 where file='a' and gen is not null;<
0
lepto
Query> setattr /home/koblitz/a version 1.0
   >select table_name from masterindex where directory='/home/koblitz';<
   >select version from dir1 where version is not null limit 1;<
   >alter table dir1 add version varchar(256);<
   >insert into dir1 (file, version) values ('a' ,'1.0');<
   >update dir1 set version='1.0' where file='a';<
0
Query> getattr /home/koblitz/a version
   >select table_name from masterindex where directory='/home
   >select version from dir1 where file='a' and version is not null
0
1.0
Query> getattr /home/koblitz/b version
   >select table_name from masterindex where directory='/home/koblitz';<
   >select version from dir1 where file='b' and version is not null;<
2
Query> quit
```

```
metadata=# select * from dir1;
 file |   gen    | events
------+----------+---------
 a    | lepto    |     101
 b    | phytia   |     101
 c    | lepto    |   20001
 d    | lepto    |   30001
```

```
metadata=# select * from dir1;
 file |   gen    | events | version
------+----------+--------+---------
 b    | phytia   |    101 |
 c    | lepto    |  20001 |
 d    | lepto    |  30001 |
 a    | lepto    |    101 | 1.0
```

Future Plans

Will continue to investigate generic metadata server:

- No benchmarks with prototype yet
  (no GSI, memory leak)
- Will look into transaction safety for security
- Look into design of distributed Dbs
- See whether API and protocol is complete
- Look into how to tune backend:
  - Tune for insertion/update
  - Tune data types
  - Tune for search patterns

Lessons Learned

Learned a lot looking at existing implementations:

- Current implementations seem to be inventing the wheel over and over: Repeat each others mistakes
- Transfer Protocol:
    - SOAP is particularly bad for Metadata
    - SOAP blows up data by factors 5-10
    - SOAP is for single queries with small (structured responses)
    - Metadata queries require stateful connections with streamed data (of possibly unknown structure) as a response
- Schema evolution needs to be done by user (not admin!)
- FS Metadata concept looks very appealing

How many DBs?

There are File-Catalogues, Metadata-Catalogues, Replica-Catalogues, ...

Question: How many DBs do we need?

Answer: 1 + X

Files, Metadatas, and Replicass in 1 catalogue
+ X special purpose catalogues

- Severe performance problems with find if File- and Metadata seperated (CMS experiences)
- File-Permissions + GUIDs are Metadata
- Metadata needs info from File-Catalogue: Security!
- Reasons to separate Metadata- and File-Cat. seem:
  - Historical
  - People don't trust consistency of Metadata
  - Large special purpose metadata expected
    ➔Special purpose DB (pointer in Metadata)
- File- and Replica-Cat. easily seperated (different users)