# Metadata in the GRID

Birger Koblitz
ARDA Workshop, October 18th, 2004

Overview

- Trying to define metadata
- Our Experiences with metadata catalogues
  - Protocol problems
  - Schema handling
- A generic definition of metadata
- The POSIX metadata interface
- Testing a Prototype Metadata Catalogue
- Ideas for the Future

# What is Metadata?

1. Definition:

Metadata is information on contents of files.
(File Metadata)

Also other information found in DBs necessary to run jobs on the grid, share problems:

- Grid authentication
- Overcoming firewalls
- Talking efficiently to DBs
- Replication
- Distributed updates?

2. Definition:

Metadata is all kind of data needed by jobs to run on the grid
(apart from what is in the files).

# Hierarchy

Metadata needs a hierarchy to work well:

- Collect objects with shared attributes into collections

  ➔Allows queries on SQL tables

  (also other storage possible: XML-DB, DB-Files...)

  Analogy to file system (file metadata!):

  Collection ↔ Directory

  Object ↔ File

- Structure important for:
  - Structured searches
  - Schema handling
  - Distribution of databases

# Experience

ARDA tested several Metadata solutions from the experiments:

- CMS: RefDB
  (PHP in front of MySQL, giving back XML tables)
- Atlas: AMI
  (SOAP-Server in Java in front of MySQL)
- gLite (Alien Metadata)
  (Perl in front of MySQL parsing command, streaming back text)

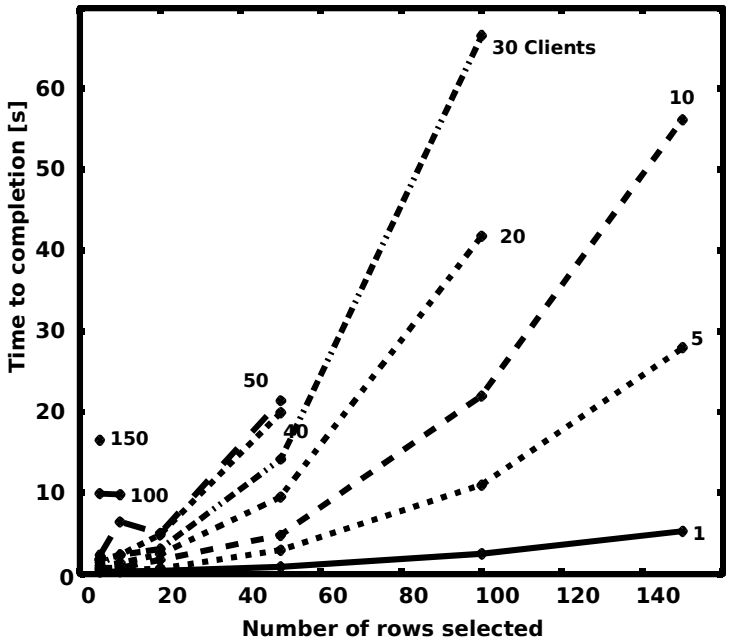
Learned a lot looking at existing implementations:

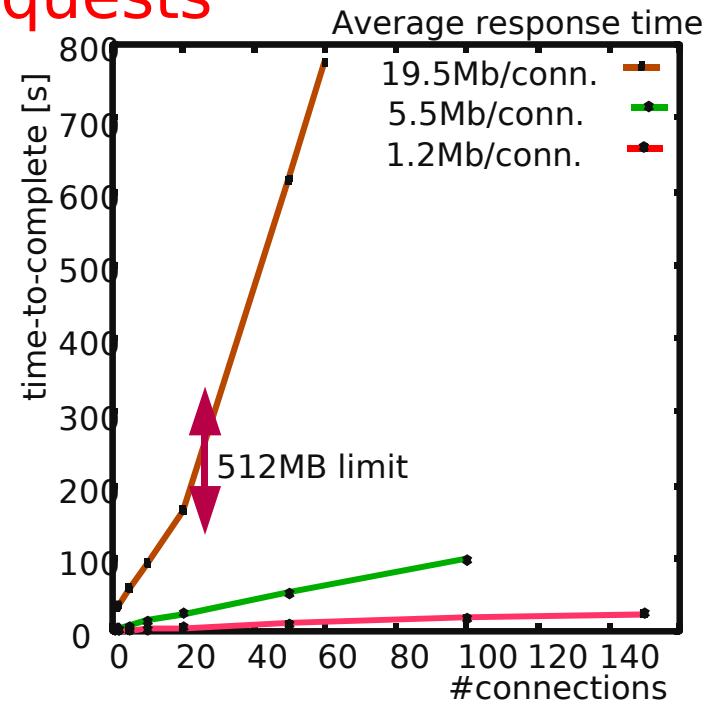
- Common pattern seen
- Implementations also share the same problems

# Protocol: SOAP

Both AMI & RefDB ship responses in single XML package

➔They can't handle large requests



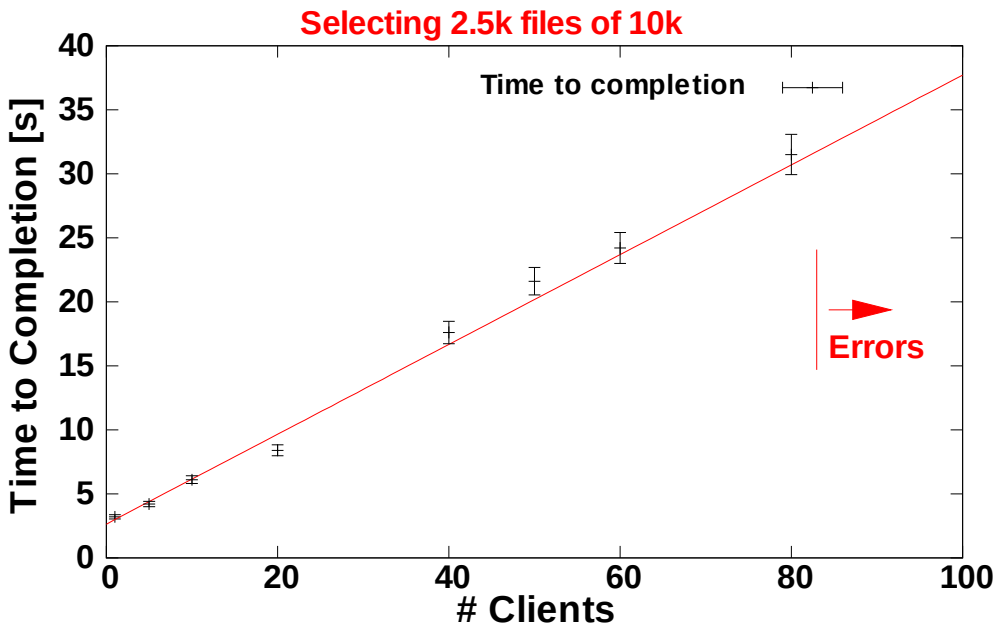

SOAP is particularly bad for Metadata:

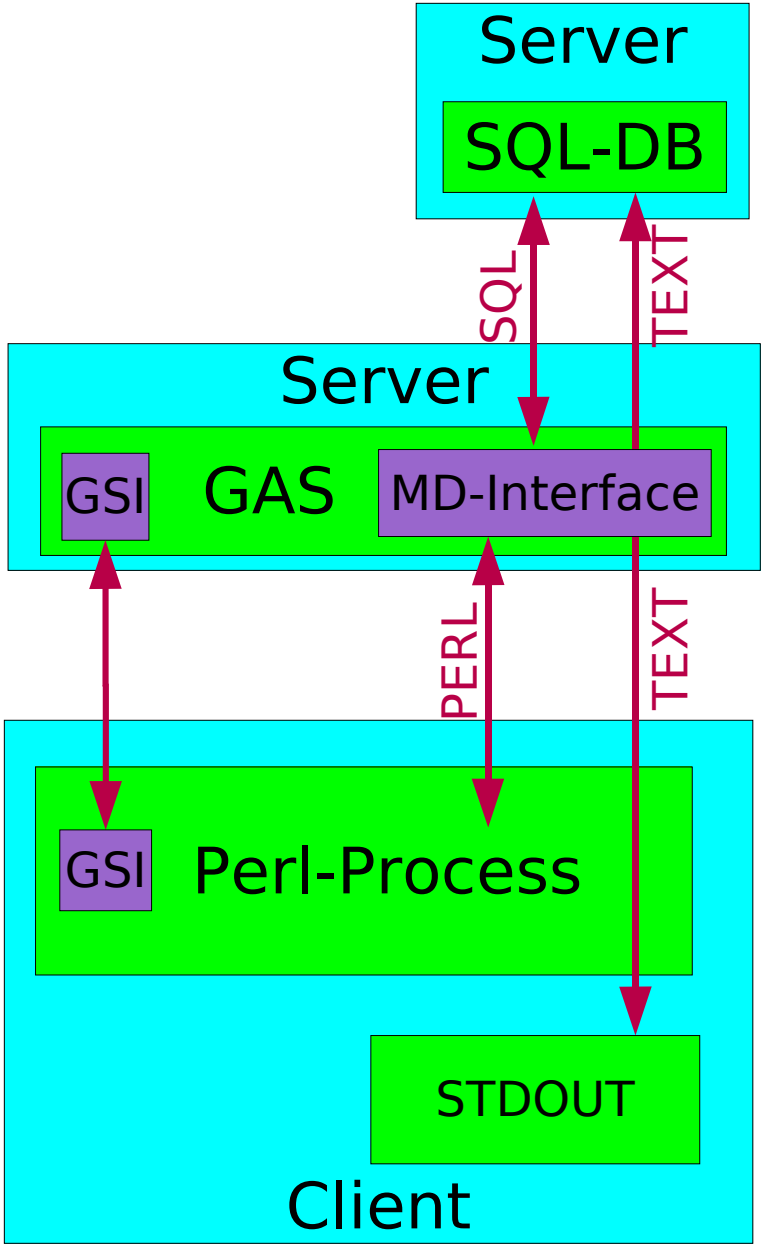
- SOAP blows up data by factors 5-10
- SOAP for single, small queries

Metadata queries do require stateful connections with Streamed Data / Iterators as a response

# Streamed Data

gLite(Alien) streams responses to the perl implemented shell



# Schema Handling

Schema evolution not really tackled by current Metadata Catalogues:

- Not really important for production...
- Admin can setup/copy new tables (work on backend)...

RefDB and Alien don't do schema evolution at all. AMI via admins adjusting tables.

For analysis, the following capabilities are mandatory:

- User must be able to discover schema
- User can setup/change schema (He can then do it's own schema management, or another application layer can take care)
- Offer solution for problems with storage types

# 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)
Used with Namespaces for ACLs

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

- Windows Longhorn (2005)
- ReiserFS 5

# Metadata on Linux

On ext3, XFS or ReiserFS, Linux supports extended attributes (file metadata)

```
koblitz@pcardabk:~/test$ touch a
koblitz@pcardabk:~/test$ attr --help
Usage: attr [-LRSq] -s attrname [-V attrvalue] pathname  # set value
       attr [-LRSq] -g attrname pathname                 # get value
       attr [-LRSq] -r attrname pathname                 # remove attr
      -s reads a value from stdin and -g writes a value to stdout
koblitz@pcardabk:~/test$ attr -s gen -V lepto a
Attribute "gen" set to a 5 byte value for a:
lepto
koblitz@pcardabk:~/test$ attr -s version -V 1.0 a
Attribute "version" set to a 3 byte value for a:
1.0
koblitz@pcardabk:~/test$ getfattr -d a
# file: a
user.gen="lepto"
user.version="1.0"
koblitz@pcardabk:~/test$ grep home /etc/fstab
/dev/hda5 /home ext3 defaults,acl,user_xattr,auto 0 0
```

Can we have a similar semantics on the Grid?

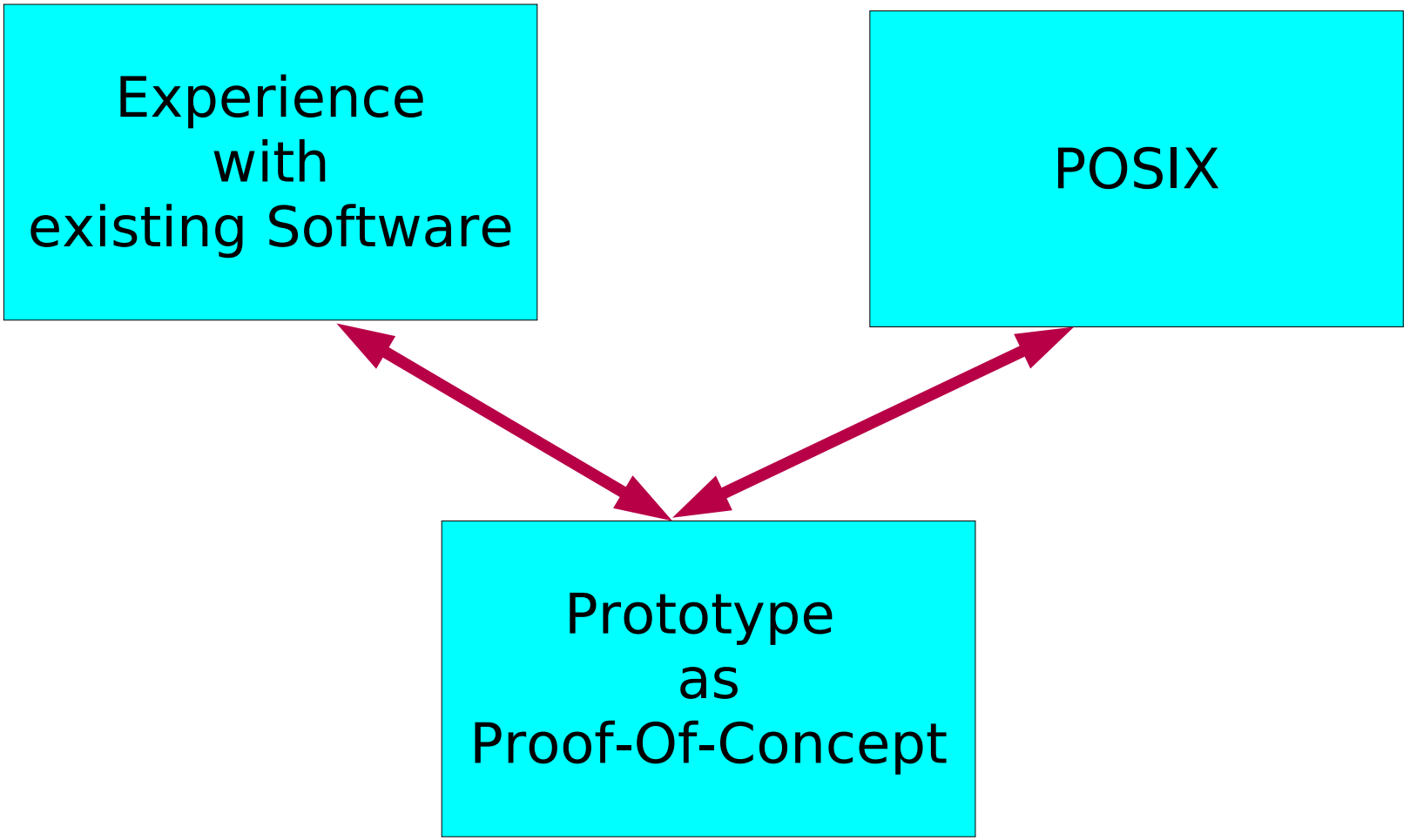
PS: API is POSIX, not the commands!

# GRID Metadata

A possible Grid approach:

- 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
  - ➔ Allows hierarchical databases, applicable for FS

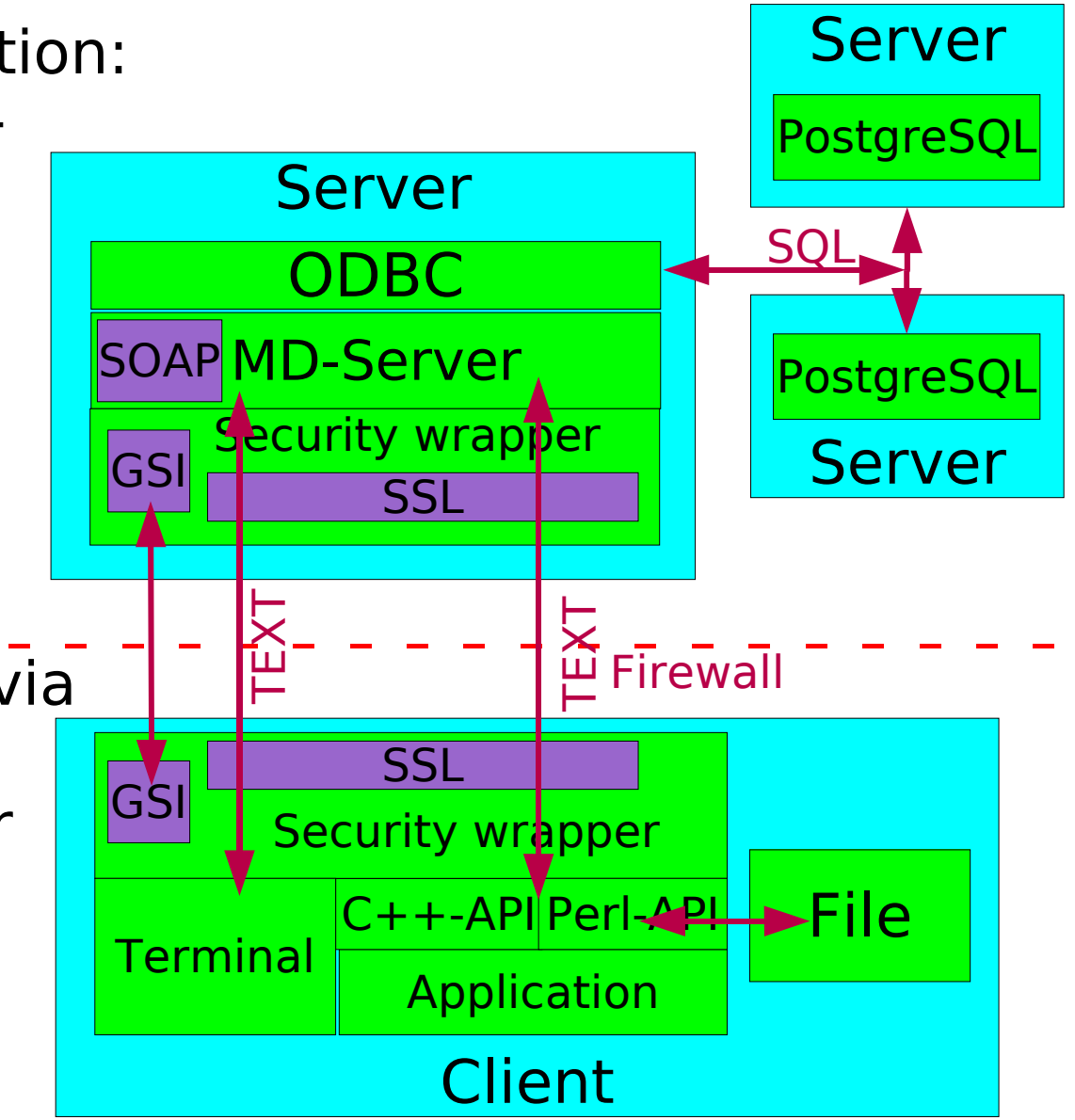
# Synthesis


Experience
with
existing Software
POSIX
Prototype
as
Proof-Of-Concept

# Prototype

Prototype Implementation:

- Multi-threaded C++ server in front of PostgreSQL
- Streams responses asynchronously
- Uses ODBC as RDBMS abstraction Layer: ODBC-types
- Access restrictions via ACLs
- Bison/flex parser for queries
  →Other backends
  →Query validation
  →Security



# 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 (and EOT at end)
- Result is in ASCII, user needs to encode/decode
- Commands are:

| | |
|---|---|
| `addattr dir key type` | Add new key to schema |
| `removeattr dir key` | Removes key from schema |
| `getattr file(s) key1 key2...` | Returns value of keys |
| `setattr file [key value]`$_n$ | Sets keys to values (bulk upd.) |
| `listattr file` | Returns keys&type line-by-line |
| `clearattr file(s) key` | Resets a key |
| `find pattern 'query '` | Returns list of files matching pattern and query on keys |

Client needs to Encode/Decode data, understand schemas (but can discover them)

With R. Rocha(gLite), V. Pose

# Metadata API

POSIX defines the following commands (implemented for compatibility):

- `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

For bulk operations C++ better suited:

- `int setAttr(const string &file, const list<string>&keys, const list<string> &attr)`
- `int getAttr(const string &pattern,const list<string> &keys, AttributeList &attributes)`

InAttributeList keeps connection to server, allows iteration on streamed data:

- `int AttributeList::getRow(string &file, vector <string>  &attributes)`

  Returns row of file and its attributes

# Example Session

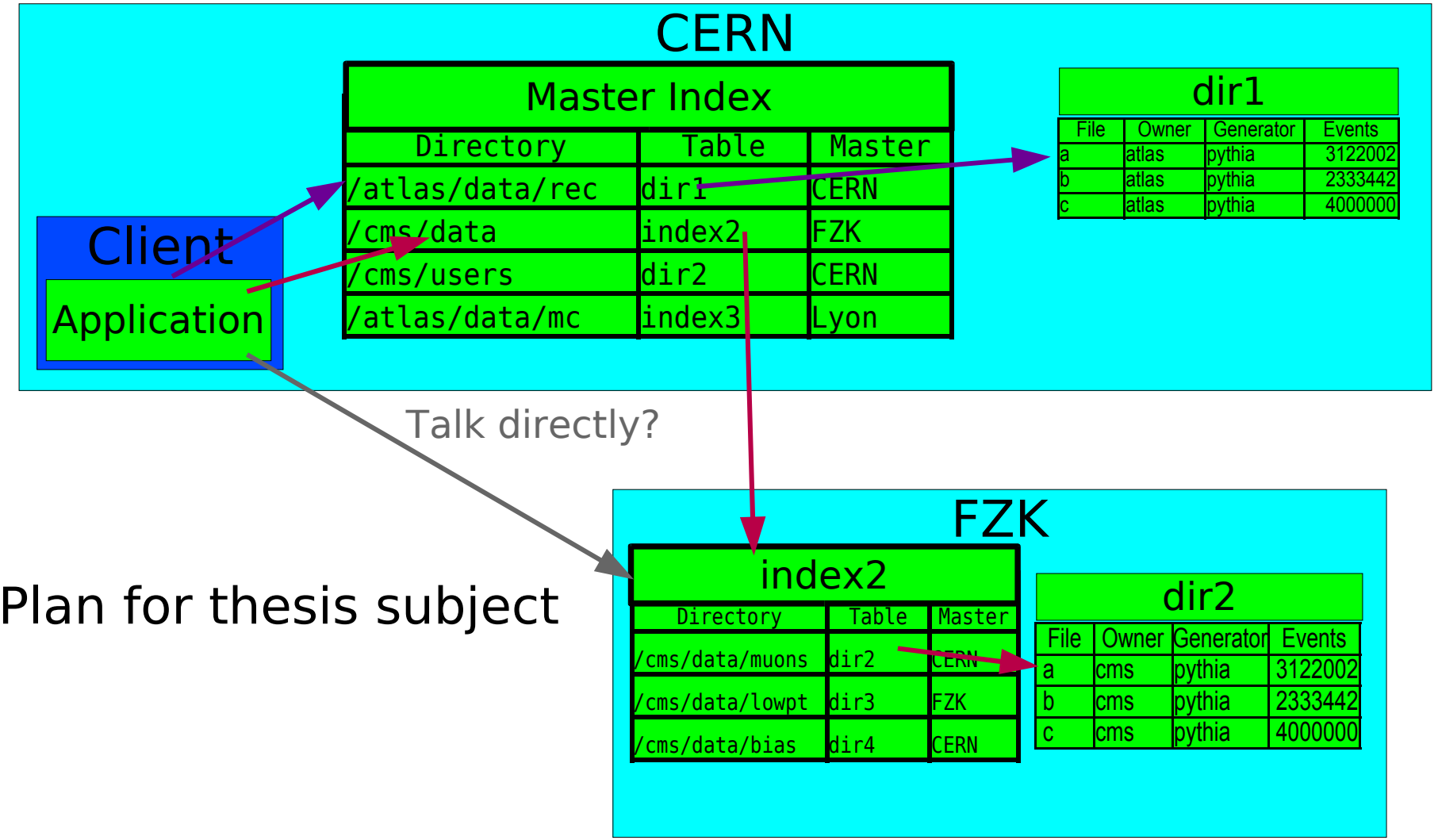
```
koblitz@pcardabk:~/mi$ telnet pcardabk 8822
Connected to DB
Query> getattr /home/koblitz/a gen
  >select table_name from masterindex where directory='/home/koblitz';<
  >select gen from dir1 where file='a' and gen is not null;<
0
lepto
Query> addattr /home/koblitz version int
  >select table_name from masterindex where directory='/home/koblitz';<
  >alter table dir1 add version int;
0
Query> setattr /home/koblitz/a version 1.0
  >select table_name from masterindex where directory='/home
  >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/koblitz';<
  >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
```

# Distributed Metadata Ideas

Use PostgreSQL per-table replication with different masters, make use of hierarchy:




Plan for thesis subject

# More Ideas

To be more generally useful:

- Create user indices with views:

```
create view dirs as select generator, file from dir1
    union
select generator, file from dir2;
CREATE INDEX gen_index ON dirs(generator);
```

- Or via inheritance:

```
CREATE TABLE “/atlas/data/2008” INHERITS “/atlas/data/2009”;
```

Copies 2008 schema, select on 2009-data gives also 2008 data.

(Both features available in PostgreSQL)

# Conclusions

- Many problems understood
- Seems possible to create metadata catalogue suitable for very different metadata
- But room for special database solutions exist
  (And could be pointed to from a central catalogue)
- Chose practical approach: Prototype
- Design and Implementation certainly challenging
  ➔Metadata experts from experiments need to work together, requirements must be named