D-Grid Initiative

A Design Proposal for Interactive Data Analysis on the Grid

Anar Manafov, GSI

HEPCG Workshop, Nov 30 2006

last modified date : 28-11-2006; version: 1.03

Presentation “Road map”

- The ALICE model of distributed data analysis
- Gap Analysis
- The model and its implementation
- The design concept and development plan

The ALICE model of distributed data analysis

- **ROOT** is a de facto standard framework in HEP community and provides three key points:
  - the possibility to do interactive analysis work in familiar C++ style syntax;
  - data visualization;
  - an object-oriented I/O system.

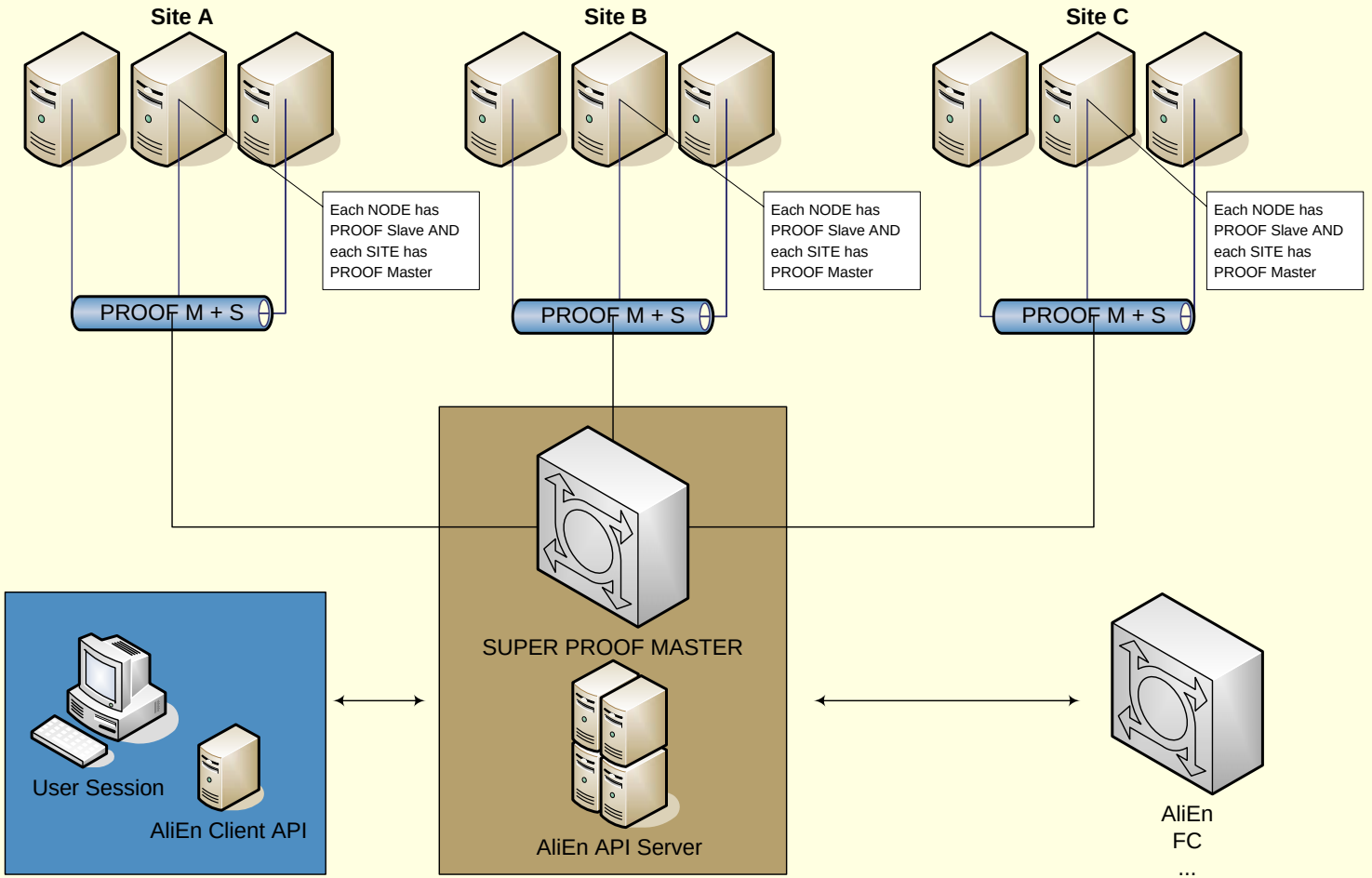
*It is successfully used within frameworks of different experiments as an "**all in one**" solution.*

- **PROOF** extends a workstation based concept of ROOT to a parallel processing on a cluster, where:
  - user procedures are kept identical during an analysis session;
  - tasks are distributed automatically in the background.

*The motto of PROOF is: "**Bring the KB to the PB, not the PB to the KB!**"*

- **AliEn** (**ALI**CE production distributed **en**vironment) as a Grid analysis platform provides two key elements:
  - a global file system, files are indexed and tagged in a virtual file catalogue and everywhere globally accessible;
  - a global queue system, global job scheduling according to resource requirements.

The ALICE model of distributed data analysis



Gap Analysis

- **High level gaps**
  - There is no general and Grid enabled data analysis schema;
  - There is no standardized API of the Grid;
  - So far it is not possible to use gLite to distribute PROOF servers without ALICE/AliEn.
- **Low level gaps**

Mainly there is missing or erroneous functionality of the ROOT Grid interface. It has only been tested using AliEn, our task is to improve it using other MWs (gLite, GLOBUS4, UNICORE).

- **Third-party gaps**

As a third-party gap we would like to call the one that was discovered and analyzed by our effort, but unfortunately can't be fixed or redesigned by ourselves (gLite MW issues for example).

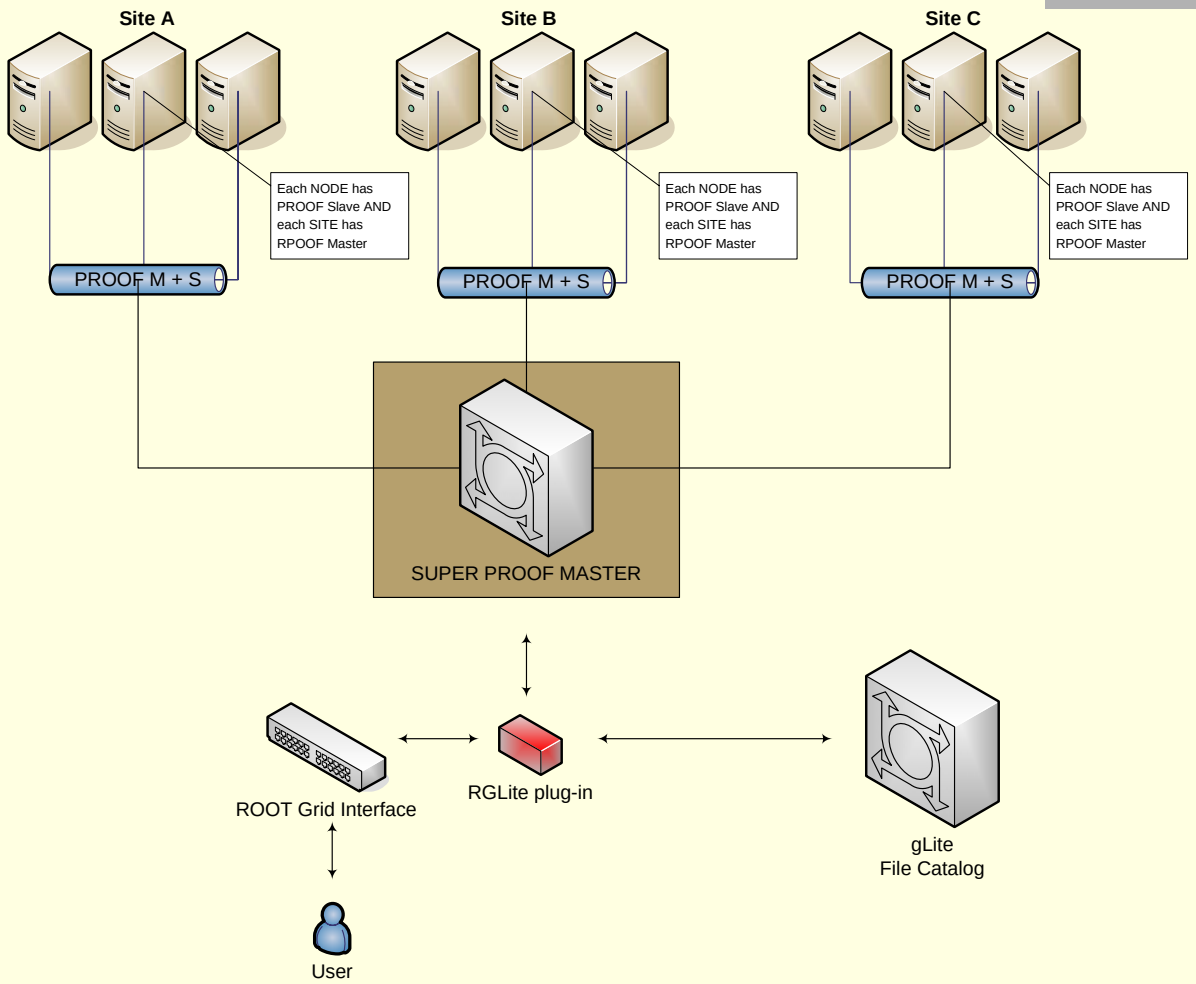
For detailed information we would like to refer to:
GSI's "Stage One Report on WP 3: Distributed Data Analysis under usage of Grid Resources –Gap Analysis**",**
http://wiki.gsi.de/cgi-bin/view/Grid/D-GridProjectReport

The model and its implementation

Components of our Model are:

- gLite testbed (gLite R1.4, R1.5 and R3.0 were used),
- SRM compatible storage (currently dCache),
- ROOT frame work,
- glite-api-wrapper library (GAW),
- RGLite - a ROOT plug-in for gLite.

The model and its implementation



gLite MW

We base our project on gLite Grid Middleware, since it is widely used and one of the biggest operational Grids today. gLite is a strategic Grid middleware for HEP.

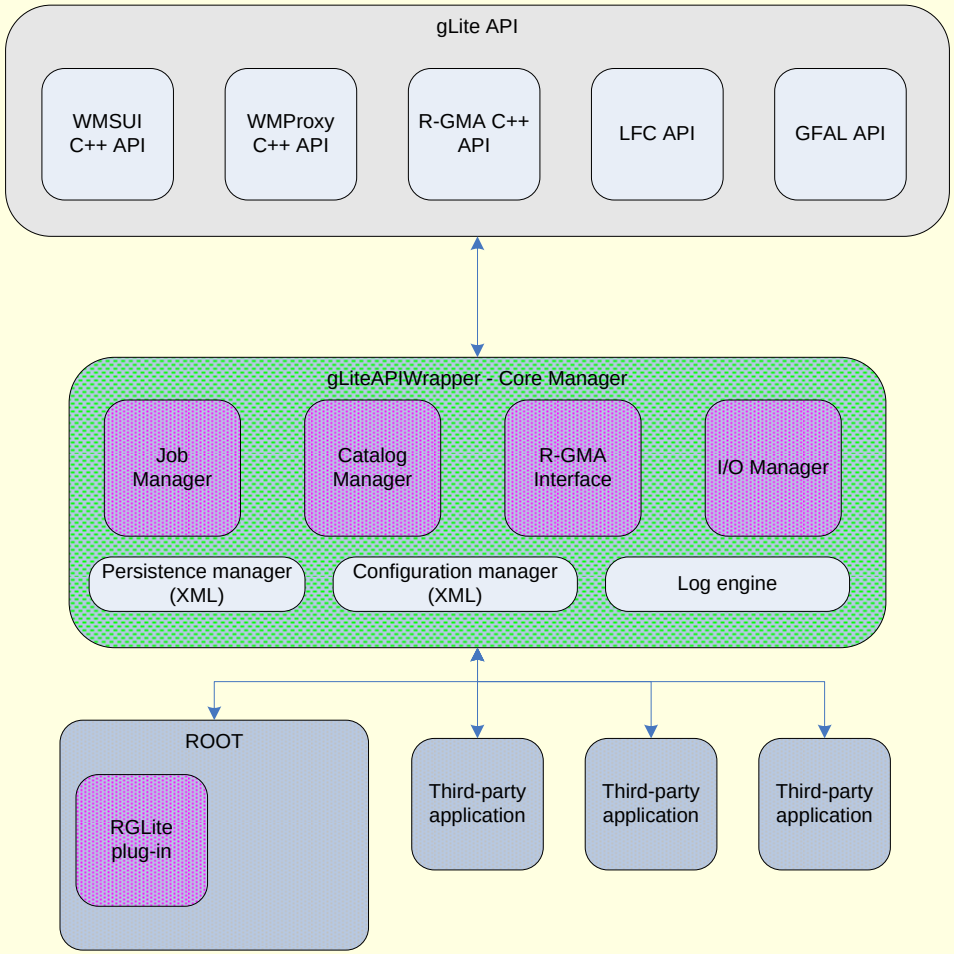
The current implementation of our project is based on:

- gLite WMSUI – for job submission, status querying and output retrieving;
- LFC (Local LCG File Catalog) API – for file catalog operations;
- GFAL (Grid File Access Library) API – for file operations on the gLite Grid.

But we also used the following components of gLite:

- WMProxy – for job submission;
- FiReMan – for file catalog operations;
- gLite I/O server — for file operations on the gLite Grid.

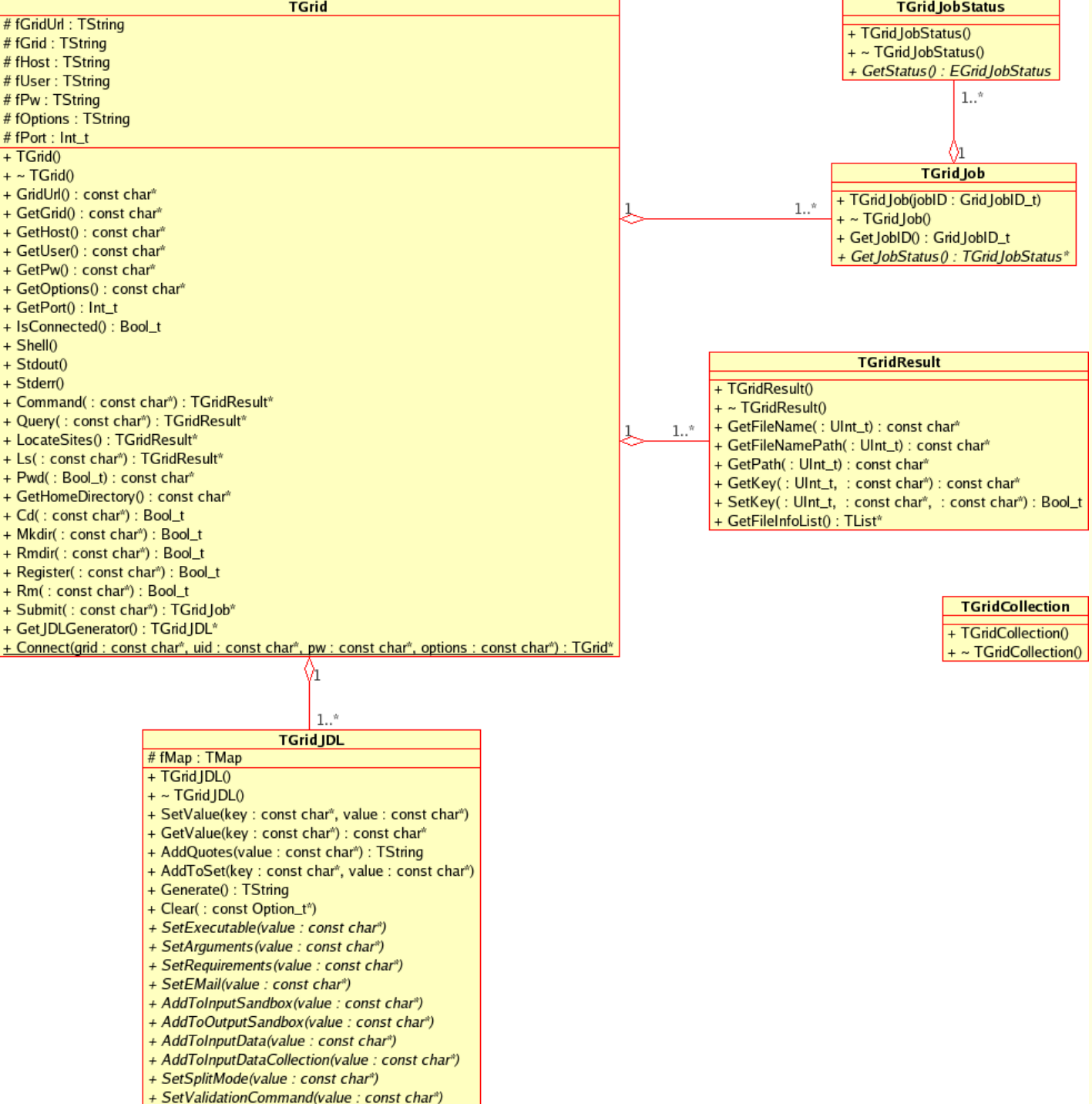
GAW – glite-api-wrapper library



- GAW Core Manager,
- GAW Job Manager,
- GAW Catalog Manager,
- GAW Persistence Manager,
- GAW Configuration Manager,
- GAW Log Engine.

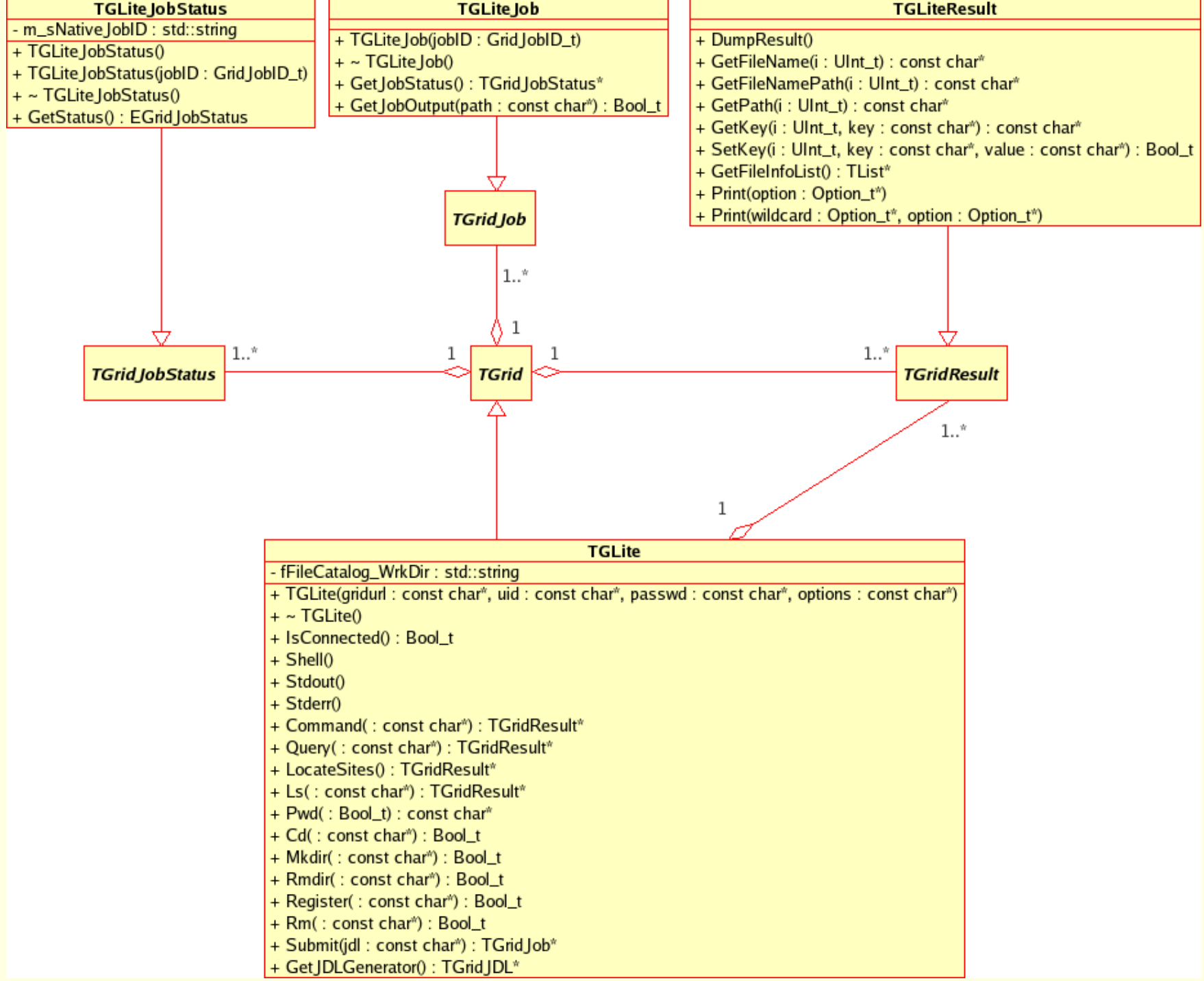
RGLite

- Workload Management System (WMS) operations:
  - job submission,
  - job status querying,
  - job output retrieving.
- File Catalog operations:
  - to set and query current working catalog directory,
  - to list the content of the selected catalog directory (list files, directories and their stats),
  - to create a file in a catalog namespace,
  - to add replica(s) to a given file in the catalog namespace,
  - to remove replica(s) from a given file in the catalog namespace,
  - to remove a file or directory from a given file in the catalog namespace.
- An executive logging.
- Support of an external configuration.
- File I/O operations could be performed with the help of *TGFALFile* - a ROOT class wrapper of GFAL.



ROOT

UML class diagram of the ROOT Grid Interface


TGLiteJobStatus
- m_sNativeJobID : std::string
+ TGLiteJobStatus()
+ TGLiteJobStatus(jobID : GridJobID_t)
+ ~ TGLiteJobStatus()
+ GetStatus() : EGridJobStatus
TGLiteJob
+ TGLiteJob(jobID : GridJobID_t)
+ ~ TGLiteJob()
+ GetJobStatus() : TGridJobStatus*
+ GetJobOutput(path : const char*) : Bool_t
TGLiteResult
+ DumpResult()
+ GetFileName(i : UInt_t) : const char*
+ GetFileNamePath(i : UInt_t) : const char*
+ GetPath(i : UInt_t) : const char*
+ GetKey(i : UInt_t, key : const char*) : const char*
+ SetKey(i : UInt_t, key : const char*, value : const char*) : Bool_t
+ GetFileInfoList() : TList*
+ Print(option : Option_t*)
+ Print(wildcard : Option_t*, option : Option_t*)
TGridJob
1..*
1
TGridJobStatus
1..*
1
TGrid
1
1..*
TGridResult
1..*
1
TGLite
- fFileCatalog_WrkDir : std::string
+ TGLite(gridurl : const char*, uid : const char*, passwd : const char*, options : const char*)
+ ~ TGLite()
+ IsConnected() : Bool_t
+ Shell()
+ Stdout()
+ Stderr()
+ Command(: const char*) : TGridResult*
+ Query(: const char*) : TGridResult*
+ LocateSites() : TGridResult*
+ Ls(: const char*) : TGridResult*
+ Pwd(: Bool_t) : const char*
+ Cd(: const char*) : Bool_t
+ Mkdir(: const char*) : Bool_t
+ Rmdir(: const char*) : Bool_t
+ Register(: const char*) : Bool_t
+ Rm(: const char*) : Bool_t
+ Submit(jdl : const char*) : TGridJob*
+ GetJDLGenerator() : TGridJDL*

RGLite Samples

```
// Initializing RGLite plug-in
TGrid::Connect("glite");
// Submitting a Job to gLite Grid
TGridJob *job = gGrid->Submit("JDLs/proofd.jdl");
// querying a Status of the Job
TGridJobStatus *status = job->GetJobStatus();
status->GetStatus();
// Getting a Job's output back to the user
TGLiteJob* job_glite(job);
job_glite->GetJobOutput("/home/anar/");
```

Job submission, Status querying, Output retrieving.

```
// Initializing RGLite plug-in
TGrid::Connect("glite");
// Changing current File Catalog directory to "dteam"
gGrid->Cd("dteam");
// Querying a list of files of the current FC directory
TGridResult* result = gGrid->Ls();
// Printing the list out
Int_t i=0;
while (result->GetFileName(i))\
> printf("File %s\n",result->GetFileName(i++));
```

Changing file catalog directory, querying lists of files.

RGLite Samples

```
// Initializing RGLite plug-in
TGrid::Connect("glite");
// Changing current File Catalog directory to "dteam"
gGrid->Cd("dteam");
// Querying a list of files of the current FC directory
TGridResult* result = gGrid->Ls();
// Printing the list out, including full file information
result->Print("all");
```

List full file information of a catalog folder.

```
using namespace std;
// Initializing RGLite plug-in
TGrid::Connect("glite");
// Changing current File Catalog directory to "dteam"
gGrid->Cd("dteam");
// Printing a current working directory
cout << "Working Directory is" << gGrid->Pwd() << endl;
// Creating a new File Catalog Folder
Bool_t b = gGrid->Mkdir("root_test2");
```

Retrieving a name of a current file catalog folder,
Creating a new file catalog folder.

RGLite Samples

```
[anar@depc218 ~]$ root -b
  ******************************************
  *                                        *
  *       W E L C O M E  to  R O O T       *
  *                                        *
  *   Version   5.11/06       1 June 2006  *
  *                                        *
  *  You are welcome to visit our Web site *
  *          http://root.cern.ch           *
  *                                        *
  ******************************************

Compiled on 30 July 2006 for linux with thread support.

CINT/ROOT C/C++ Interpreter version 5.16.12, May 16, 2006
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0] TGrid::Connect("glite");
Info in <TGLite::TGLite>: gLite API Wrapper engine has been successfully initialized.
root [1] TGridJob *job = gGrid->Submit("jdl/test.jdl");
Info in <TGLite::Submit>: Job successfully submitted
Info in <TGLite::Submit>: NativeJobID https://grid25.gsi.de:9000/ouz0AOWsjR7LYFPoeXqoJw; ROOT JobID 1;
root [2] TGridJobStatus *status = job->GetJobStatus();
root [3] status->GetStatus()
Info in <TGLiteJobStatus::GetStatus>: Native JobID = https://grid25.gsi.de:9000/ouz0AOWsjR7LYFPoeXqoJw
Info in <TGLiteJobStatus::GetStatus>: Job status is [3]; gLite status code is "Ready"; gLite status string is "matching resources fo
und"
Info in <TGLiteJobStatus::GetStatus>: Job status is kWAITING
(const enum TGridJobStatus::EGridJobStatus)1
root [4] status->GetStatus()
Info in <TGLiteJobStatus::GetStatus>: Native JobID = https://grid25.gsi.de:9000/ouz0AOWsjR7LYFPoeXqoJw
Info in <TGLiteJobStatus::GetStatus>: Job status is [6]; gLite status code is "Done"; gLite status string is "execution finished, ou
tput is available"
Info in <TGLiteJobStatus::GetStatus>: Job status is kDONE
(const enum TGridJobStatus::EGridJobStatus)5
root [5] TGLiteJob* job_glite(job);
root [6] job_glite->GetJobOutput("/home/anar/");
root [7]
```

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Design concept and development plan

- Local PROOF cluster
- “Marriage” of PROOF and gLite
- Complete GAW and RGlite plug-in implementation
- Deployment
- Testing by user
- Implementation for other Grid MW
- Investigate other Grid APIs
  - *GAT*
  - *SAGA*