Journal of Physics: Conference Series **119** (2008) 072022 doi:10.1088/1742-6596/119/7/072022

RGLite, an interface between ROOT and gLite – PROOF on the Grid

Peter Malzacher, Anar Manafov, Kilian Schwarz

Gesellschaft für Schwerionenforschung mbH, Planckstr. 1, D-64291 Darmstadt, Germany

E-mail: K.Schwarz@gsi.de, A.Manafov@gsi.de

Abstract. Using the gLitePROOF package it is possible to perform PROOF-based distributed data analysis on the gLite Grid. The LHC experiments managed to run globally distributed Monte Carlo productions on the Grid, now the development of tools for data analysis is in the foreground. To grant access interfaces must be provided. The ROOT/PROOF framework is used as a starting point. Using abstract ROOT classes (TGrid, ...) interfaces can be implemented, via which Grid access from ROOT can be accomplished. A concrete implementation exists for the ALICE Grid environment AliEn. Within the D-Grid project an interface to the common Grid middleware of all LHC experiments, gLite, has been created. Therefore it is possible to query Grid File Catalogues from ROOT for the location of the data to be analysed. Grid jobs can be submitted into a gLite based Grid. The status of the jobs can be asked for, and their results can be obtained.

1. Introduction

1.1. Motivation

Due to the fast growing amount of data, the complex and CPU intensive computations, and the participation of scientific groups on all continents, the data analysis of present and future experiments in the field of particle and nuclear physics requires the development of a distributed computing infrastructure based on resources all over the globe.

With the start of the Large Hadron Collider (LHC) [1] at the European Centre for Particle Physics (CERN) [2] the demands on distributed computing technology will reach new levels. The worldwide LHC computing Grid (WLCG [3]) has been developed to provide the computing infrastructure for the four LHC experiments ALICE, ATLAS, CMS, and LHCb. WLCG consists of several logical layers, the so called tiers, and each tier has been assigned a specific number of computing tasks. Tier-0 is located at CERN and responsible for data taking, data storage and distribution, as well as for first data processing. Further tasks are simulation of experiment data as well as reconstruction of particle tracks, and preselection of interesting physics channels (first level analysis, centrally steered). These will be computed at the large tier-1 centres of which one will be built in each of the larger CERN member states or in clearly defined regions. At the smaller tier-2 centres, which are often connected to a specific tier-1 centre, more simulation will take place, but also data analysis done by individual physicists.

Simulation of experiment data, data storage at CERN, data distribution to the tier-1 centres, as well as data reconstruction have been extensively exercised by all four LHC experiments in so called data challenges. The current state of the art is, that, centrally managed, production

quality distributed computing is possible by using tens of thousands of CPUs distributed over the whole globe.

As mentioned above, the largest challenge in the high energy physics community will be the analysis of the produced data in a distributed environment by using Grid technology. A working analysis environment is absolutely necessary to be able to profit from the high investment in accelerator and detectors. The demand on the resource management is significantly more complex for distributed analysis. Instead of one or two production managers in the case of distributed Monte Carlo simulation, now hundreds or thousands of physicists per experiment may send jobs to the Grid. To facilitate this also for users without Grid experience, tailored user interfaces have to be created.

Especially for members of the ALICE [4] experiment ROOT (section 1.2) provides a starting point for most common tasks. Since the ALICE framework AliRoot [5] is based directly on ROOT, an ALICE user can do data simulation, reconstruction, and analysis all by using the same framework - all from within the ROOT system. Additionally a working interface between ROOT and the ALICE Grid Environment AliEn [6] exists so that Grid access from within ROOT is also possible.

AliEn is a lightweight, but fully functional, Grid framework which has been developed by the ALICE collaboration and is built around Open Source components using a combination of Web Services and distributed agents. The majority of the implementation has been done in Perl. Next to the usual site services such as the Computing Element (CE), a Storage Element (SE), file transfer service, and monitoring, the main components of AliEn are a well performing central file catalogue, which contains the mapping of logical to physical file names as well as corresponding metadata, a package manager for automatic application software installation, various user interfaces, an application API, and a central task queue. Here also the match making between the requirements of submitted jobs and the capabilities of AliEn Grid sites takes place. One of the key points of AliEn is the pull architecture compared to the push architecture as it is used in EGEE. In AliEn the site CEs advertise themselves at the central task queue as soon as they have free capacities available. This way it is not necessary to have a central Resource Broker which has to have the overview over the Grid at any given time and place. AliEn has been put into production in 2001 and up to now more than 7 million ALICE-jobs have been running under AliEn control worldwide.

The combination of AliEn and ROOT directly enables analysis of large and globally distributed data sets by querying the file catalogue and having the analysis batch jobs distributed via Grid methods to the sites where the data are stored. But for a fast prototyping of new analysis code this technique is not feasible since the response time is too slow. Therefore ALICE is setting up prompt and interactive analysis facilities based on PROOF [7] which enable parallel and interactive analysis of large data sets situated at a single site. Working examples are the CERN analysis facility (CAF) and the PROOF analysis facility at GSI [8] (GSIAF). A possiblity to enable fast and interactive analysis of global data sets would be to combine the advantages of Grid and PROOF. That this is technically possible has been already demonstrated by the ALICE experiment in two SuperComputing conferences by using PROOF and AliEn.

Currently all LHC experiments try to achieve these or comparable features by implementing their own analysis environment. But especially with respect to the Grid, synergy should be possible by making use of the common Grid project of all four experiments, WLCG. To be able to realise a powerful analysis environment in an integrated platform an essential building block would be to provide general Grid access directly from ROOT to gLite [9], the Grid Middleware which has been agreed to be used by all LHC experiments.

This has been realised within the D-Grid-Project [10]. D-Grid was started in March 2004 and

follows similar programs in the UK and the USA. The main purpose of the project is to build a sustainable Grid infrastructure in Germany. D-Grid supports a comprehensive integration project and several community Grid projects, e.g. the HEP community Grid (HEPCG, [11]), which supports primarily the LHC experiments but also future high energy and nuclear physics experiments like FAIR [12] at GSI. As a sub project of the HEPCG an interface between ROOT and gLite, RGLite [13] has been developed. RGLite makes use of the abstract ROOT Grid interface classes TGridXXX (chap. 2.1) and follows the already existing implementation for AliEn. By using the gLitePROOF package [14], which has also been developed in this context, additionally an interactive PROOF analysis of a large data set distributed over several sites can be done within a plain gLite environment.

1.2. The ROOT System

The ROOT framework [15] is an analysis software package, which is widely used within the HEP environment. The ROOT system provides a set of object oriented frameworks with all the functionality needed to handle and analyse large amounts of data in a very efficient way. Having the data defined as a set of objects, specialised storage methods are used to get direct access to the separate attributes of the selected objects, without having to touch the bulk of the data. Included are histograming methods in various dimensions, curve fitting, function evaluation, minimisation, graphics and visualisation classes to allow the easy setup of an analysis system that can query and process the data interactively or in batch mode. Although the emphasis of ROOT is on the data analysis domain also features and tools for event generation, detector simulation, event reconstruction, and data acquisition are provided. ROOT is an open system that can be dynamically extended by linking external libraries. Thanks to the built in CINT [16] C++ interpreter the language for both macros and compiled code is C++.

1.2.1. PROOF The Parallel ROOT Facility, PROOF, is an extension of the ROOT System, which enables physicists to analyse and understand much larger data sets on a shorter time scale by analysing many ROOT files in parallel on local or remote compute clusters. PROOF is designed to work on objects in ROOT data stores and makes use of the inherent parallelism in event data. In principle there should be as little difference as possible between a local ROOT based analysis session and a remote parallel PROOF session.

PROOF consists of a 3-tier architecture, the ROOT client session, the PROOF master server and the PROOF workers. The user connects from his ROOT session to a master server on a remote cluster, and the master server in turn creates worker processes on all nodes in the cluster. Queries are processed in parallel by all PROOF workers. Using the pull scheduling paradigm the PROOF workers ask the master for work packages, which allows the master to distribute customised packages for each worker. The main tunable parameter is the packet size. When all events are processed the workers send the partial results to the master which combines all of them and sends the information back to the client. A typical PROOF session has been visualised in fig. 1. Performance measurements show a very good and efficient scalability of PROOF (fig. 2).

A central role in the PROOF system plays the TSelector framework. To use the framework a user derives a class from TSelector and implements the member functions that are part of the protocol. The framework also specifies how input objects are made available to the selector object and how output objects are returned to the user or client session. By using specific data set classes it is foreseen that also Grid catalogues can be queried to receive a list of data to be analysed and logical files names can be used as well as physical file names for file specification.

In complex analysis environments the analysis scripts very likely depend on external libraries. To be able to run these scripts successfully on PROOF it is required that these libraries are available on each worker node. The PROOF package manager has been designed to distribute

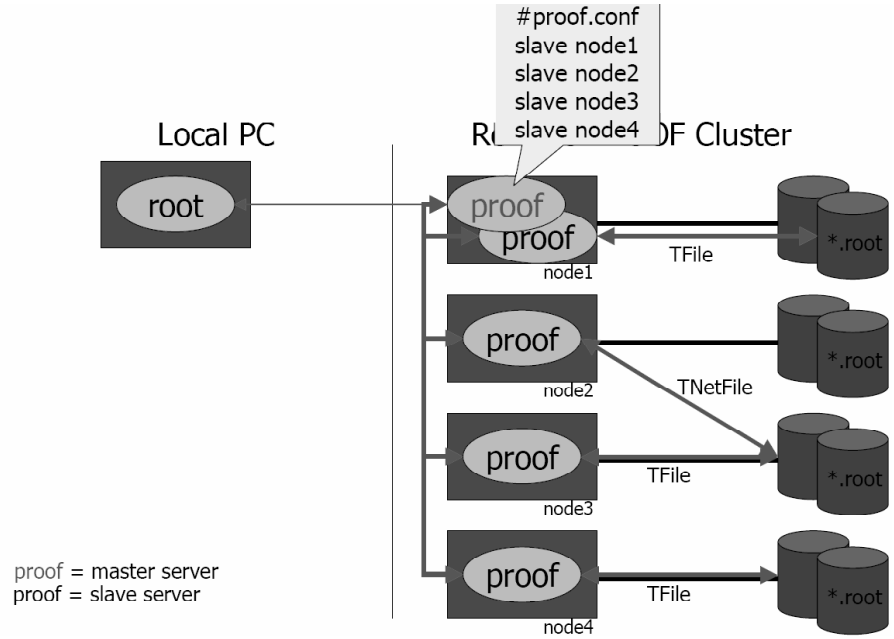

Figure 1. An example PROOF session. The user connects to PROOF and executes the analysis script on the cluster. After processing the data PROOF returns the output back to the user.

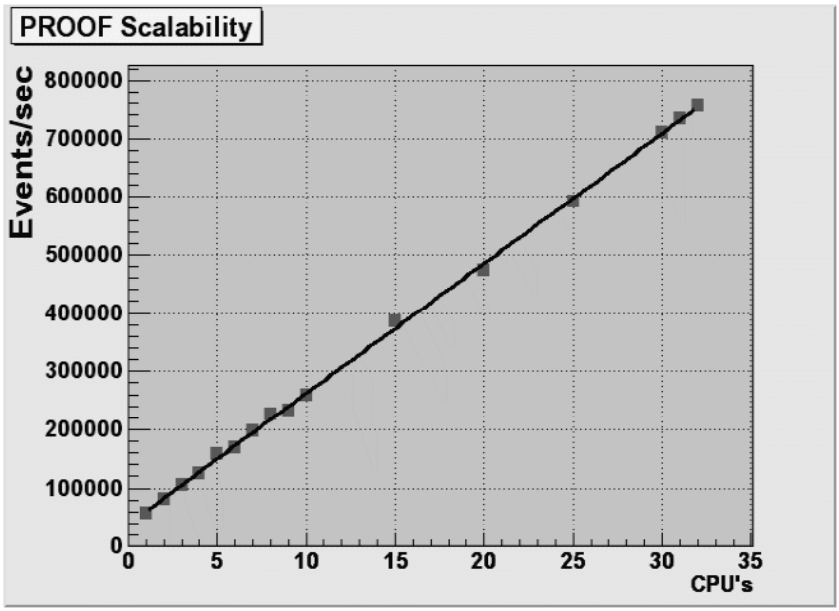

Figure 2. The figure shows the scaleability of a PROOF cluster. If enough I/O speed is provided PROOF scales almost linearly.

and install these libraries on a PROOF cluster. Packages are compressed tar files which can contain the library sources or binaries.

2. Technical Overview

2.1. RGLite

ROOT offers a set of abstract base classes, which provide an interface between ROOT and the Grid. These classes are TGrid, TGridJDL, TGridJob, TGridJobStatus, TGridResult, and TGridCollection. TGrid itself defines an interface to common Grid services. To open a connection to a Grid from inside ROOT, e.g., the static method TGrid::Connect() can be used. Depending on the desired Grid flavor an appropriate plug-in library is loaded to provide the real interface. TGridJDL is used to generate JDL files and for job submission to the Grid. TGridCollection manages file collections on the Grid, and TGridResult contains the result of a Grid query, for example the content of a directory in a Grid file catalogue (fig. 4). The functionality of the remaining classes should be self explanatory.

A concrete implementation, TAliEnxxx [17], exists already for the ALICE Grid environment AliEn, and is part of the ROOT framework. An interface to the more general Grid middleware gLite has been created. The ROOT plugin, RGLite, provides the whole range of functionality, which is defined by the ROOT Grid interface. This offers ROOT users the possibility to use gLite directly from within their ROOT macros and executables. The interface includes querying the file catalogue, submitting jobs, inquiring the status, and receiving the output of jobs (fig. 3). Since the standard file operation mechanism in gLite is GFAL [18] and the GFAL support is already included as part of the standard ROOT installation it has not been necessary to include this functionality in RGLite.

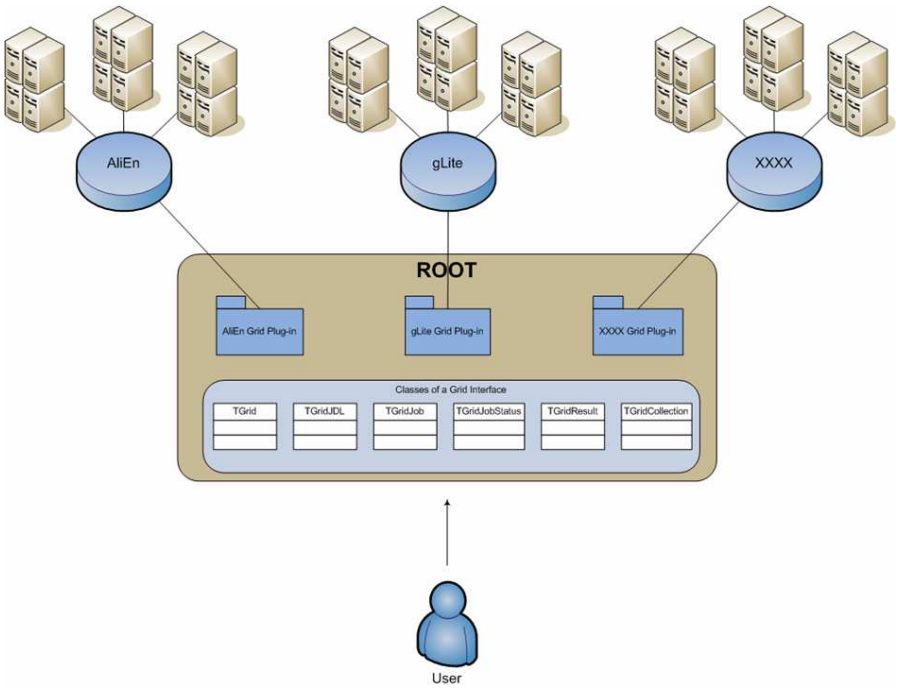

Figure 3. The abstract base classes providing the ROOT Grid interface. Concrete implementations exist for the ALICE Grid environment AliEn and for gLite.

An example usage of the RGLite interface from a ROOT shell can be seen in fig. 4. In the given example a user connects to gLite, changes to the chosen directory of the used Grid file catalogue and lists the content of that directory.

The concrete implementation of RGLite uses the gLite WMProxy C++ API [19] The

```
// Initializing RGLite plug-in
TGrid::Connect( "glite" );
// Changing current File Catalog directory to "dech"
gGrid->Cd( "dech" );
// 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" );
```

Figure 4. Example usage of the RGLite interface from within ROOT. The user connects to a gLite flavored Grid, changes to the chosen directory of the Grid File catalogue used, and lists the content of that directory.

WMProxy is a simple service providing access to the Workload Management System (WMS) functionality of gLite through a Web Services based interface. It accepts job submission requests described with the job description language (JDL) and other job management and control requests such as job cancellation, job output retrieval etc. Compared to the older WM UIAPI it provides additional functionality such as bulk job submission and support for shared and compressed sandboxes. The WMProxy client API supplies the client applications with a set of corresponding interfaces and includes a Brokerinfo Access API, via which information from the Resource Broker can be retrieved. Also a simple API for the management of job identifiers is provided. RGLite has been intensively tested with recent versions of ROOT.

2.2. GAW Library

The current implementation of RGLite does not use the API of gLite directly. Instead a glite-api-wrapper (GAW) library which contains all necessary information has been created and used. The schema is displayed in fig. 5.

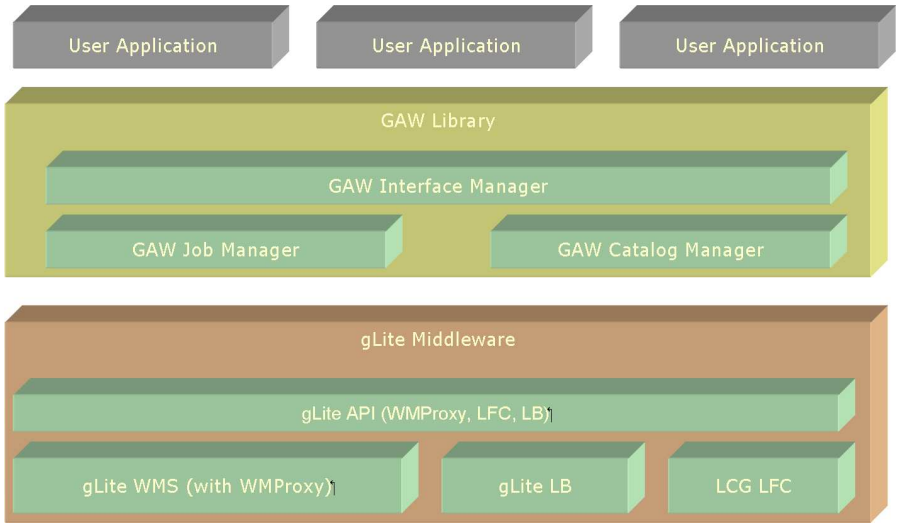

Figure 5. glite-api-wrapper (GAW) - a C++ library which wraps some part of the gLite API, adds automation and helpers. It has been designed for the needs of the RGLite project.

The GAW library is a C++ library which wraps the parts of the gLite API which is of interest for the RGLite project. However, GAW is not intended to be a general gLite API wrapper. Next

to the wrapping functionality the GAW library additionally provides automation and helpers in order to simplify access to the required functionality of gLite. This includes detailed logging, and support of an external xml configuration file. In fig. 6 a short example usage of the GAW library is displayed. The listing shows how initialisation, job submission, as well as retrieval of job status and output can be achieved in only a few lines of code. For the same operations, in comparison, one would need several hundred lines of code when using the native gLite API directly.

```
…
// GAW Init
CGLiteAPIWrapper::Instance().Init();
// Job submission
CJobManager &mng( CGLiteAPIWrapper::Instance().GetJobManager() );
mng.DelegationCredential();
const string jdl( "test.jdl" );
string jobID;
mng.JobSubmit(jdl, &jobID);
…
// Checking job's status
const glite::lb::JobStatus::Code status = mng.JobStatus(jobID);
…
// Job's output
mng.JobOutput(jobID, output_dir, &joboutput_path);
…
```

Figure 6. A short example usage of the glite API wrapper library (GAW). In a few lines of code initialisation, job submission, status and output retrieval is accomplished.

GAW requires the preinstallation of gLiteUI (gLite User Interface), the BOOST C++ libraries [20], and the XML parser Xerces C++ [21].

2.3. PROOF on the Grid

In order to perform a PROOF analysis on the Grid, the gLitePROOF package has been developed to support interactive analysis of large data sets distributed over several sites. A multi site data analysis by combining PROOF and the Grid has been demonstrated several times by representatives of the ALICE experiment using the ALICE Grid implementation AliEn. Using the gLitePROOF package a comparable functionality is given now with gLite. GLitePROOF consists of a number of utilities and configuration files which have been developed at GSI in terms of the D-Grid project. The main components are PROOFAgent and PAConsole.

PROOFAgent is a lightweight, standalone C++ application. It acts as a multifunctional proxy client and server and helps in using proof and xrootd processes behind firewalls of remote sites. Also PROOFAgent provides a number of additional functionalities which help to start, process and control an interactive PROOF analysis. The default configuration file is included in the package.

PAConsole (fig. 7), also a standalone C++ application, provides a graphical user interface to simplify the usage of PROOFAgent and gLitePROOF configuration files. PAConsole uses the GAW library to provide gLite job submission functionality. A user can therefore control PROOF jobs either from within ROOT by using the RGLite plug in or via the GUI provided by PAConsole.

Further to the main components described above, the gLitePROOF package provides a server side script for stopping and starting the gLitePROOF services, a generic XRootd configuration file for configuring the XRootd redirector and the remote Grid workers, as well as a JDL file which describes a Grid job aiming to execute gLitePROOF workers on clusters connected to the Grid.

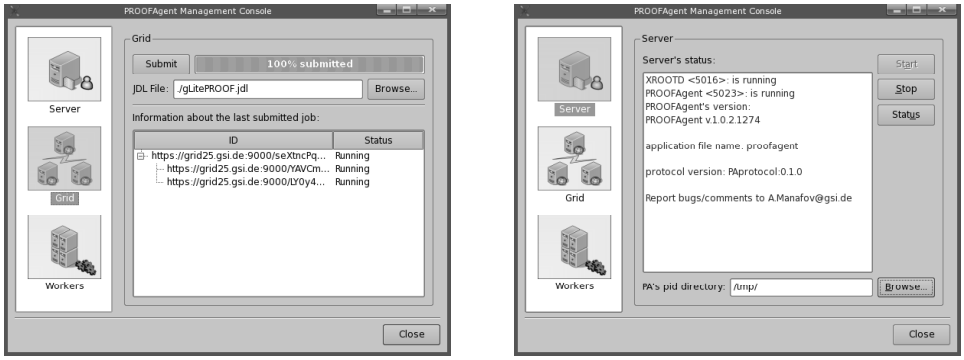

Figure 7. PAConsole provides a GUI for an easy management of an interactive PROOF analysis on a gLite flavored Grid.

3. Use Case

The main use case of the gLitePROOF package is to set up a distributed PROOF cluster. The cluster can be distributed over several sites. The only precursor is that all sites have to be interconnected via the gLite Grid middleware. A schematic picture of this use case can be seen in fig. 8.

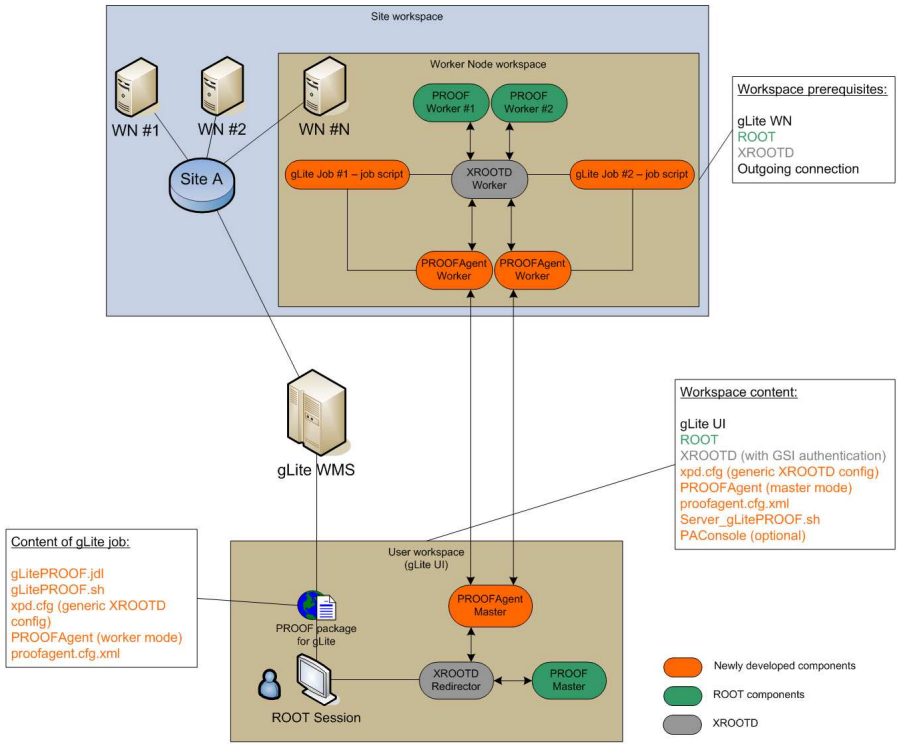

Figure 8. Setting up a distributed PROOF cluster by using the gLitePROOF package. The PROOFAgent clients are submitted as gLite Grid jobs, connect to the PROOFAgent master, and xrootd and proofd processes are started according to an automatically generated configuration file. The user can then start the PROOF analysis.

The first thing a user has to do is to start the server side processes on a central machine.

These are the PROOFAgent server and the XRootd redirector. The PROOFAgent server then connects to the PROOF port of the XRootd redirector and waits for PROOFAgent clients to connect. These processes can conveniently be started by using PAConsole.

By using the RGLite interface the user starts querying a Grid file catalogue for the data to be analysed (see also fig. 4).

Then the user submits the predefined gLitePROOF parametric job to the Grid, either by using RGLite from within ROOT or via the GUI of PAConsole. By specifying the desired parameters in the JDL file the jobs should arrive at a site where the data to be analysed are stored. As soon as a job arrives at a remote worker node it automatically configures the environment and starts all needed client services including an XRootd data server and a PROOFAgent client. The PROOFAgent client on the remote cluster, possibly behind a firewall, communicates with the PROOFAgent server, exchanges environment data, and initiates a network tunnel. After having accepted a client connection, the PROOFAgent server adds the new node to the PROOF configuration file and the client is ready to be used as a PROOF worker. In case the PAConsole is used as session management tool, each new connection is immediately reflected in the GUI.

When the instantiated PROOF workers of all submitted gLitePROOF jobs are connected or when the user is satisfied with the number of connected worker processes, the PROOF analysis can be processed as if on a local batch farm. The user then starts a ROOT session, e.g. on the private laptop, connects to the PROOF master, registers the data, and runs the analysis script (compare section 1.2.1). In this scenario the PROOF master can run also as one of the Grid jobs on a remote cluster. Since PROOFAgent can manage disconnects, the user can also disconnect from ROOT, restart the ROOT session and reconnect to the same PROOF cluster without having to resubmit the gLitePROOF jobs.

4. Summary and Outview

Within the HEP Community Grid of the D-Grid project an interface between ROOT and gLite, RGLite, has been developed. By using RGLite, direct access to gLite from within ROOT is possible. The gLitePROOF package has been developed to use either RGLite or the graphical user interface and management software PAConsole. gLitePROOF enables the setup of distributed PROOF clusters and therefore data analysis of data sets which are distributed over several sites. The sites have to be interconnected via gLite. The key component of gLitePROOF is PROOFAgent, which is a multifunctional proxy client and server and helps starting PROOF and xrootd processes behind firewalls of remote sites.

The functionality of RGLite and gLitePROOF has been successfully demonstrated during various occasions. Live demonstrations have been shown during the GridKa School 2007 [22] and on two HEPCG/D-Grid meetings [23]

The next steps are to start distributing RGLite together with new ROOT releases. Based on the experiences gained during the development of RGLite, a further concrete implementation of the ROOT Grid interface TGridxxx for another popular Grid middleware, the Globus Toolkit version 4 [24], is planned. Globus is one of the Grid flavors officially supported by D-Grid and therefore a large number of communities in D-Grid, including future projects like GSI FAIR, and also communities outside of high energy physics would profit significantly from a ROOT Globus interface.

The nightly releases of all software packages as described above, RGLite, GAW, PROOFAgent, and PAConsole, can be downloaded from the project web page [25] where detailed code documentation and and various project metrics can also be found. Current stable versions are PAConsole 1.0.1, which requires GAW v2.0.x and Qt v4.x [26]. PROOFAgent comes in the stable version 1.0.2 and requires BOOST v1.32 (boost_thread, boost_program_options), and XERCES-C v2.5. gLitePROOF 2.0.1 requires PROOFAgent v1.0.x and PAConsole v1.0.x.

5. Acknowledgments

This work is partly funded by EGEE [27] (European Union under contract number INFSO-RI-031688) and D-Grid (BMBF, Förderkennzeichen 01AK802G).

6. References

[1] LHC project http://lhc.web.cern.ch/lhc/
[2] CERN public web page http://public.web.cern.ch/Public/Welcome.html
[3] J. Shiers, Summary of WLCG Collaboration Workshop 1-2 September 2007, CHEP 2007, Victoria, Canada (2007).
for more information see http://lcg.web.cern.ch/LCG/
[4] Physics Performance Report, Volume II, ALICE Collaboration *et al* **2006** J. Phys. G: Nucl. Part. Phys. **32** 1295-2040
[5] F. Carminati, The ALICE Offline Environment, CHEP 2007, Victoria, Canada (2007).
for more information see http://aliceinfo.cern.ch/Offline/AliRoot/Manual.html
[6] P. Buncic, A.J. Peters, P. Saiz, J.F. Grosse-Oetringhaus, The architecture of the AliEn system, CHEP 2004, Interlaken, Switzerland (2004) 440.
for more information see http://alien.cern.ch
[7] F. Rademakers, B. Bellenot, B. Ballintijn, G. Ganis, Latest Developments in the PROOF System, CHEP 2007, Victoria, Canada (2007).
for more information see http://root.cern.ch/twiki/bin/view/ROOT/PROOF
[8] Homepage of GSI http://www.gsi.de
[9] gLite project http://glite.web.cern.ch/glite/
[10] D-Grid project http://www.d-grid.de/
[11] D-Grid HEP Community Grid http://www.d-grid.de/index.php?id=44&L=1
[12] FAIR project http://www.gsi.de/fair/index_e.html
[13] K. Schwarz, A. Manafov, P. Malzacher, RGLite, eine Schnittstelle zwischen ROOT und gLite, DPG (Deutsche Physikalische Gesellschaft) Frühjahrstagung (Teilchenphysik), Heidelberg (2007)
[14] unpublished
[15] ROOT - An Object Oriented Data Analysis Framework, Proceedings AIHENP'96 Workshop, Lausanne, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86.
for more information see http://root.cern.ch
[16] ROOT: An object oriented data analysis framework, Rene Brun & Fons Rademakers, Linux Journal 998 July Issue 51, Metro Link Inc
for more information see http://directory.fsf.org/CINT.html
[17] for more information see presentation "AliEn API in Root" on Alice off-line week 03-07 October 2005 http://indico.cern.ch/getFile.py/access?contribId=s0t5&sessionId=s0&resId=0&materialId=0&confId=a045061
[18] GFAL http://grid-deployment.web.cern.ch/grid-deployment/gis/GFAL/gfal.3.html
[19] gLite WMProxyAPI http://egee-jra1-wm.mi.infn.it/egee-jra1-wm/glite-wmproxy-api-index.shtml.
[20] BOOST project http://www.boost.org/
[21] XERCES-C http://xerces.apache.org/xerces-c/
[22] GridKa School 2007 http://gks07.fzk.de/
[23] HEPCG events https://indico.desy.de/conferenceTimeTable.py?confId=134
and https://indico.desy.de/conferenceOtherViews.py?view=standardconfId=327
[24] D. Fraser, Globus BOF, CHEP 2007, Victoria, Canada (2007).
for more information see http://www.globus.org
[25] Project Wiki page http://wiki.gsi.de/cgi-bin/view/Grid/RGLiteAndGAW
or http://www-linux.gsi.de/ manafov/D-Grid/Release/
[26] QT http://trolltech.com/products/qt
[27] EGEE project http://www.eu-egee.org/