Results of HEP CG WP 3, GSI: Interactive Data Analysis with PROOF

P.Malzacher@gsi.de

Anna Kreshuk, Peter Malzacher, Anar Manafov, Victor Penso, Carsten Preuss, Kilian Schwarz, Mykhaylo Zynovyev

16. June 2008

HEP CG Workshop

16.-17. June 2008

Dresden

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Interactive Data Analysis with PROOF

PROOF Overview

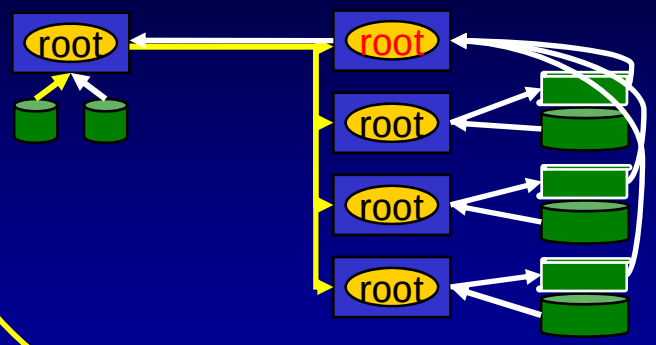
Integration of the GSI AF into the general purpose batch farm (for Grid and local batch)

Extending PROOF

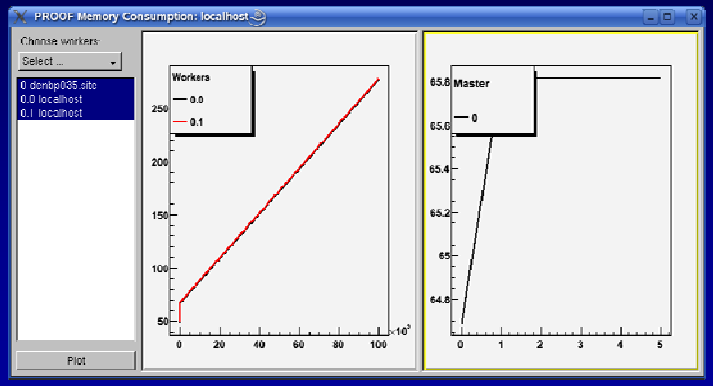
PROOF on the Grid

GSI results of HEP CG WP 3

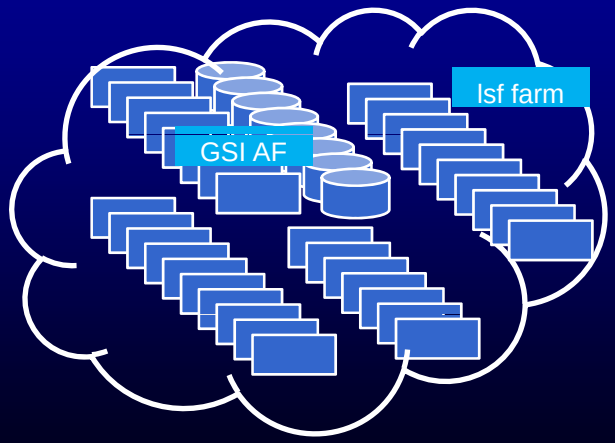
PROOF Overview



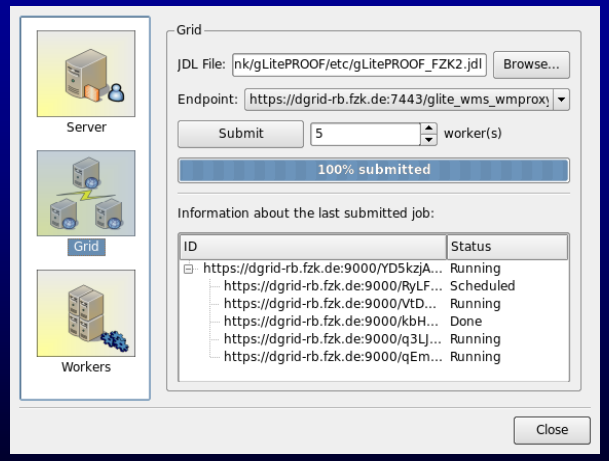
Extending PROOF



Integration of PROOF in the farm at GSI



PROOF on the Grid



PROOF: Parallel ROOT Facility

Interactive parallel analysis on a local cluster

- Parallel processing of (local) data (trivial parallelism)
- Fast Feedback
- Output handling with direct visualization
- **Not** a batch system, no Grid

The usage of PROOF is transparent

- The same code can be run locally and in a PROOF system (certain rules have to be followed)

~ 1997 : First Prototype

- Fons Rademakers

2000…: Further developed by MIT Phobos group

- Maarten Ballintijn, ...

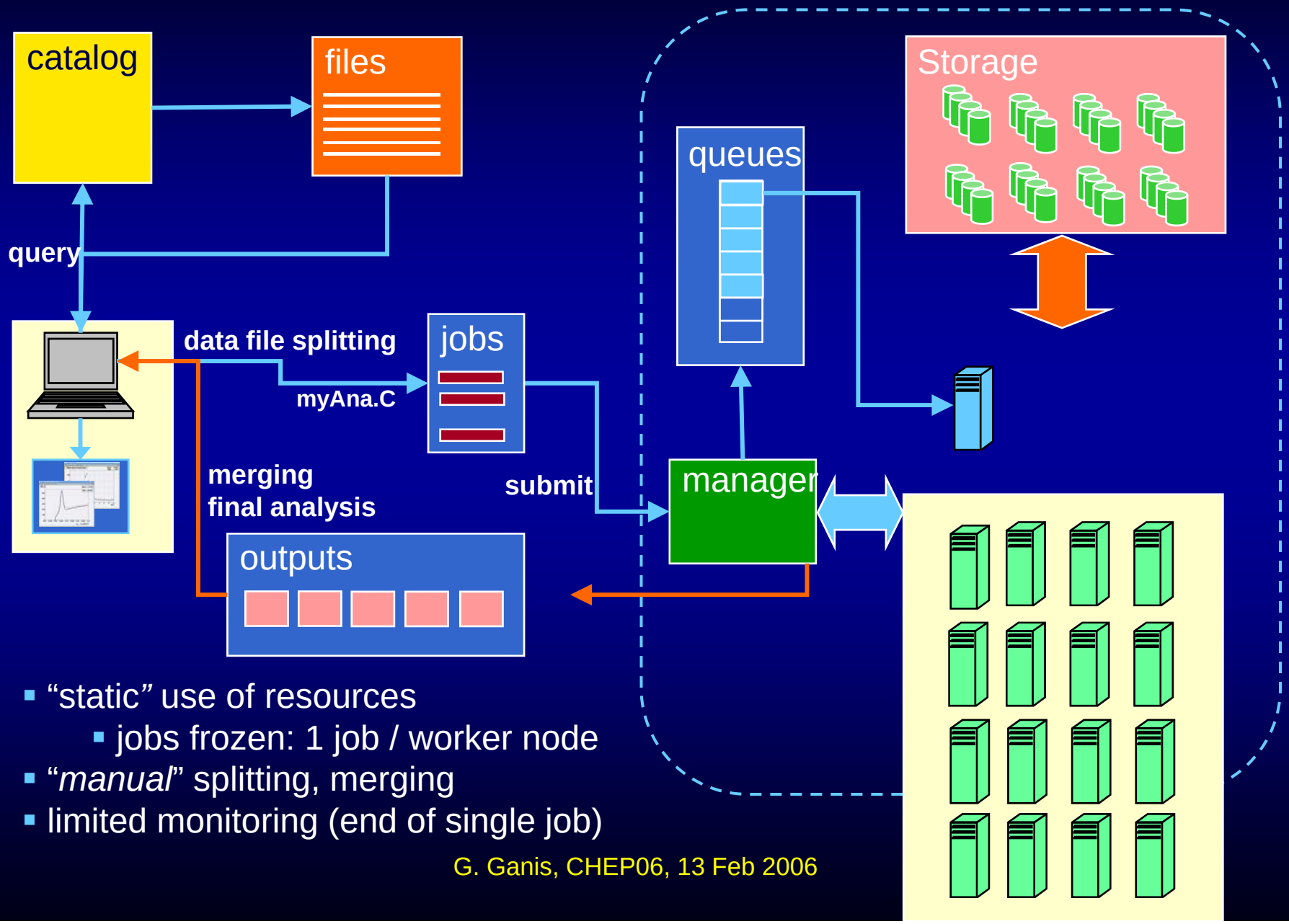
2005…: Alice sees PROOF as strategic tool

2007...: Gerri Ganis, ..., CERN AF

http://root.cern.ch/root/PROOF2007/

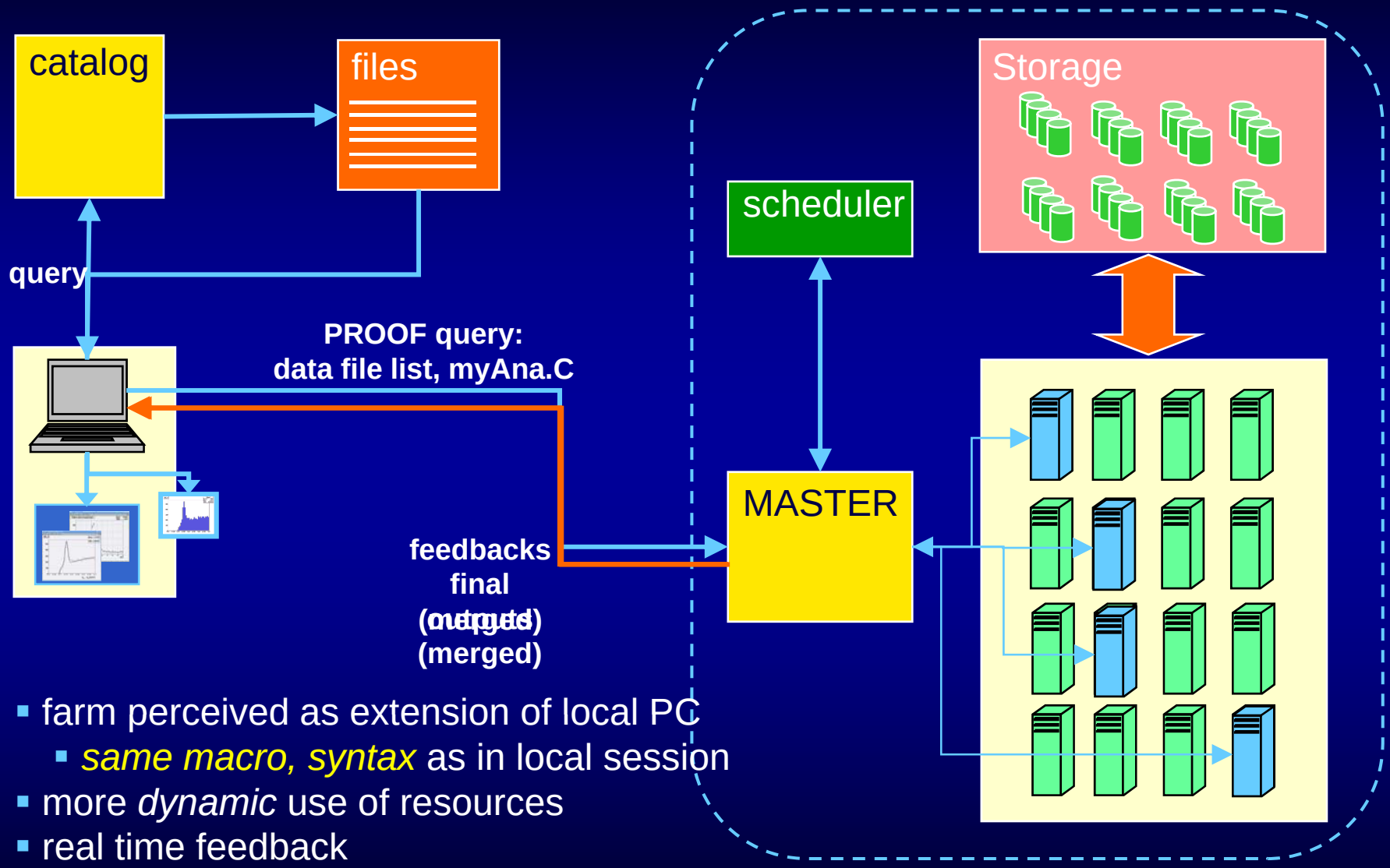
- ~ 60 participants, most from Alice, individuals from other exp.

Job Split approach



- "static" use of resources
  - jobs frozen: 1 job / worker node
- "*manual*" splitting, merging
- limited monitoring (end of single job)

The PROOF approach



- farm perceived as extension of local PC
  - *same macro, syntax* as in local session
- more *dynamic* use of resources
- real time feedback
- automated *splitting* and merging

Classes TTree /TChain

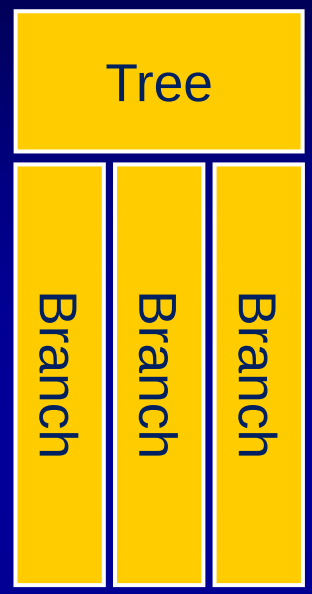
A tree is a container for data storage

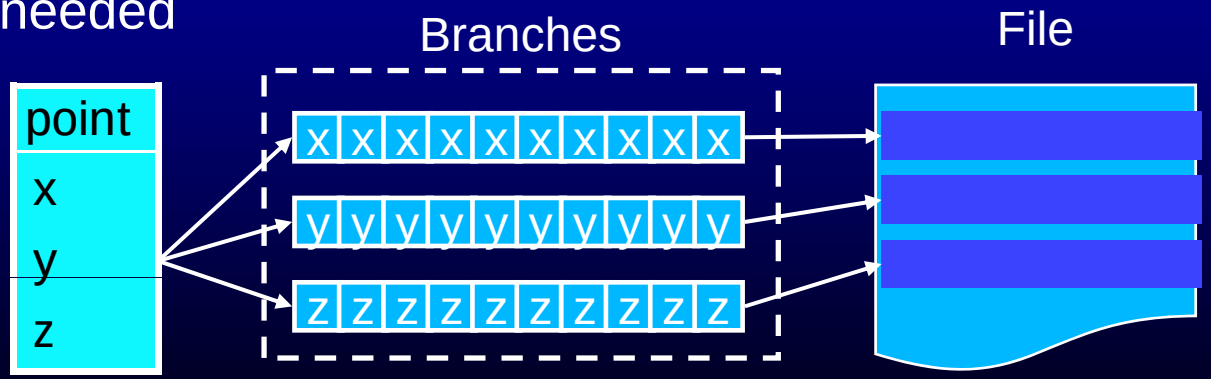
It consists of several *branches*

- These can be in one or several files
- Branches are stored contiguously (split mode)
- When reading a tree, certain branches can be switched off → speed up of analysis when not all data is needed

Compressed

A chain is a list of trees (in several files)



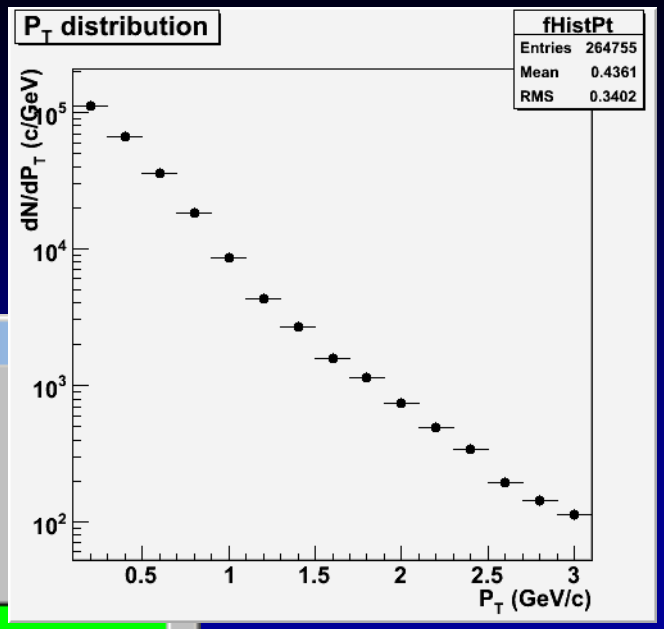


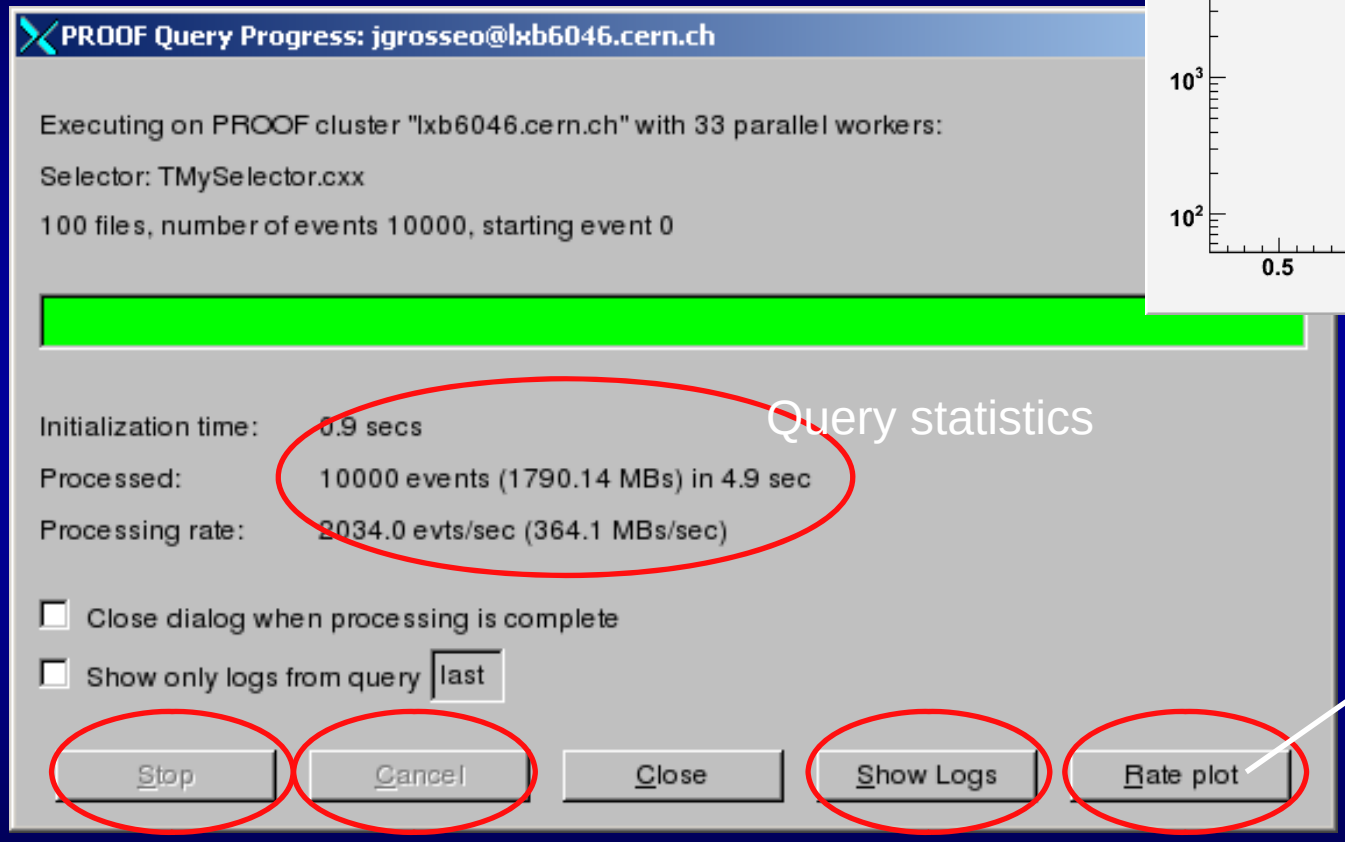
TSelector

Classes derived from TSelector can run locally and in PROOF

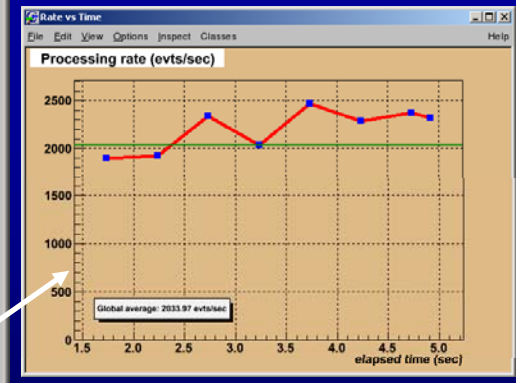
- Begin() — once on your client
- SlaveBegin() — once on each Slave
- Init(TTree* tree) — for each tree
- Process(Long64_t entry) — for each event
- SlaveTerminate()
- Terminate()

Progress dialog





Query statistics



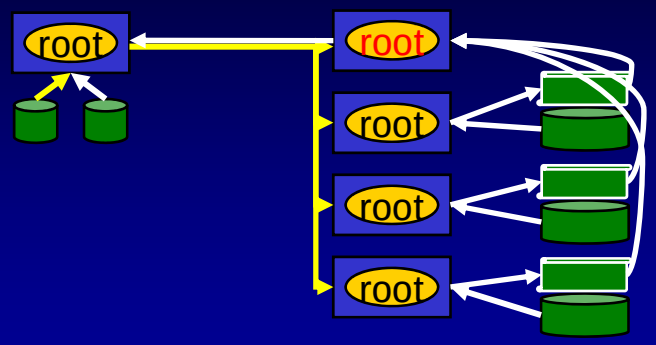
Abort query and view results up to now

Abort query and discard results

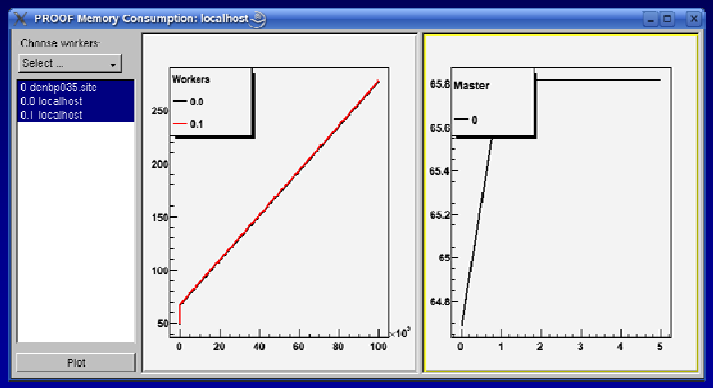
Show log files

Show processing rate

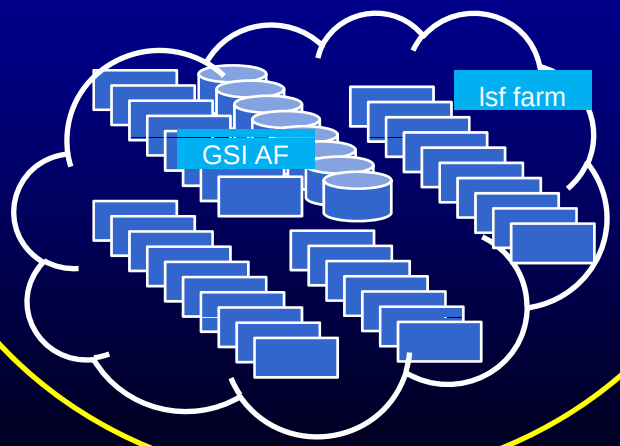
PROOF Overview



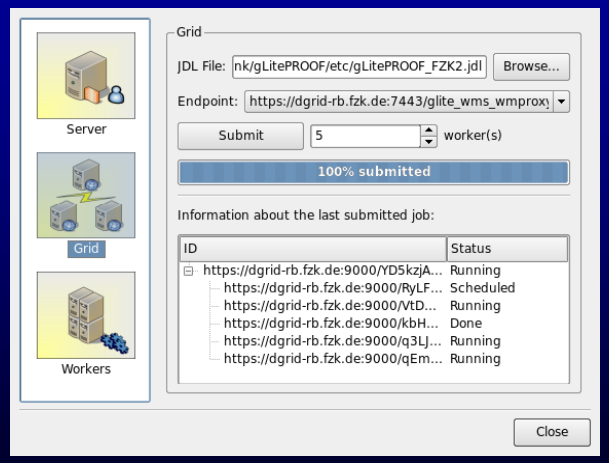
Extending PROOF



Integration of PROOF in the farm at GSI



PROOF on the Grid



Plans for the Alice Tier 2&3 at GSI: Size

Year	2007	2008	2009	2010	2011
ramp-up	0.4	1.0	1.3	1.7	2.2
CPU (kSI2k)	400	1000	1300	1700	2200
Disk (TB)	120	300	390	510	660
WAN (Mb/s)	100	1000	1000	1000	...

Germany, GSI, Darmstadt	Pledged	Planned to be pledged			
	2006	2007	2008	2009	2010
CPU (kSI2K)	100	260	660	860	1100
Disk (Tbytes)	30	80	200	260	340
Nominal WAN (Mbits/sec)	100	100	1000	1000	1000

2/3 of that capacity is for the tier 2 (fixed via WLCG MoU)

1/3 for the tier 3

To support ALICE and to learn for FAIR computing.

GSI Setup: ~40% = ALICE Tier2/3 usable via batch, grid and PROOF

~1400 cores

- batch system lsf
- debian sarge, etch32 & etch64

including

- 80 2*4core 2.67GHz Xeon with
- 4*500 GB internal disk
- ~15 used as PROOF cluster
- = GSIAF

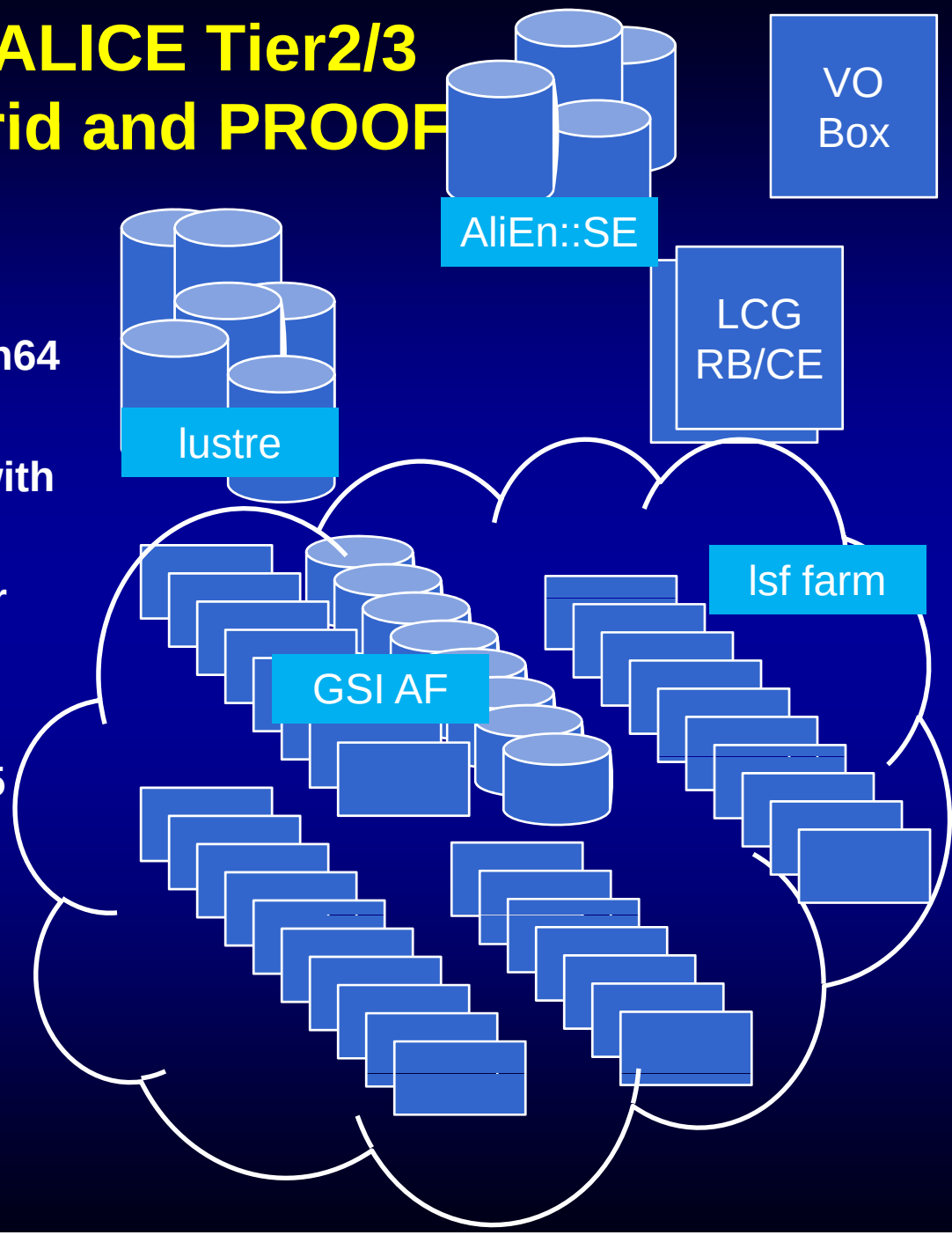
~ 500 TB in file server

- 3U 15*500GB SATA, RAID 5
- ~ 50 AliEn storage element
- ~ 450 lustre as cluster file system

data import via AliEn SE

movement to lustre or PROOF

- via staging scripts



Result: Experience Report
Set-up of an integrated batch/grid/proof farm

Layout of the queues

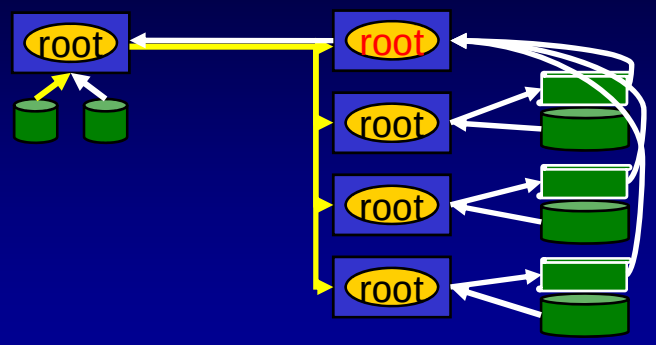
I/O: Mix of lustre and SE via xroot

GSI AF with local I/O

Suspend batch/grid jobs when proof is active

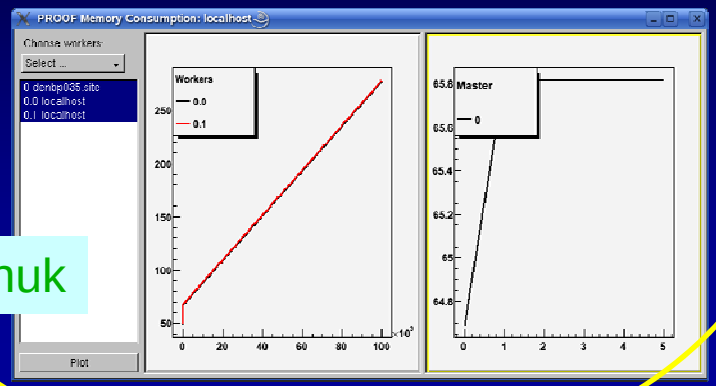
Staging to GSI AF vs usage from SE

PROOF Overview

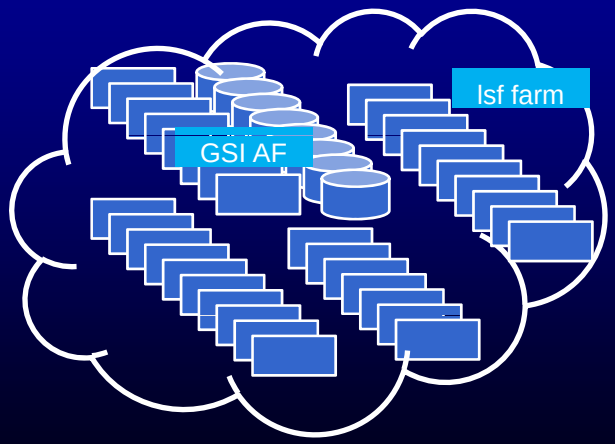


Extending PROOF

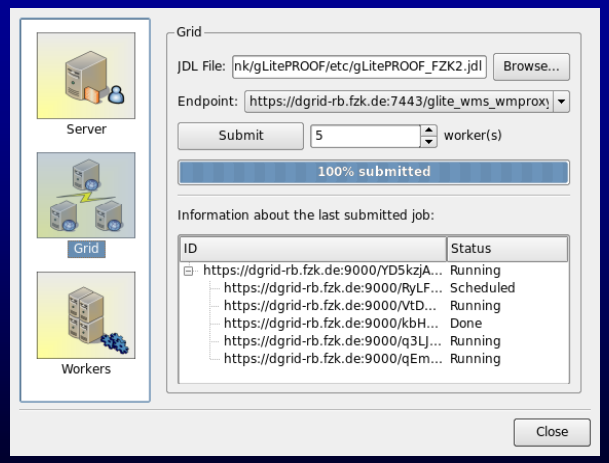
A. Kreshuk



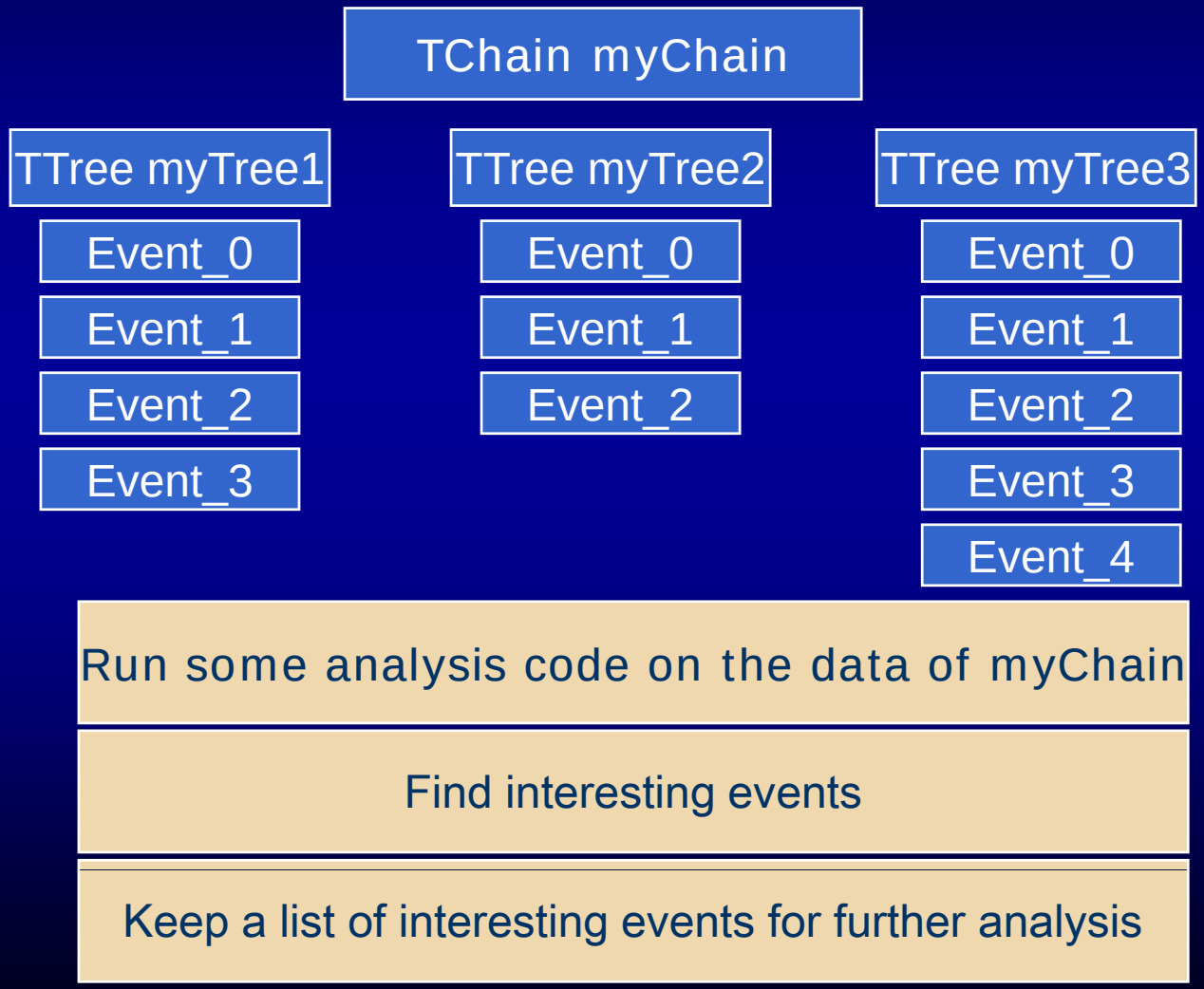
Integration of PROOF in the farm at GSI



PROOF on the Grid



TEntryList: store lists of events

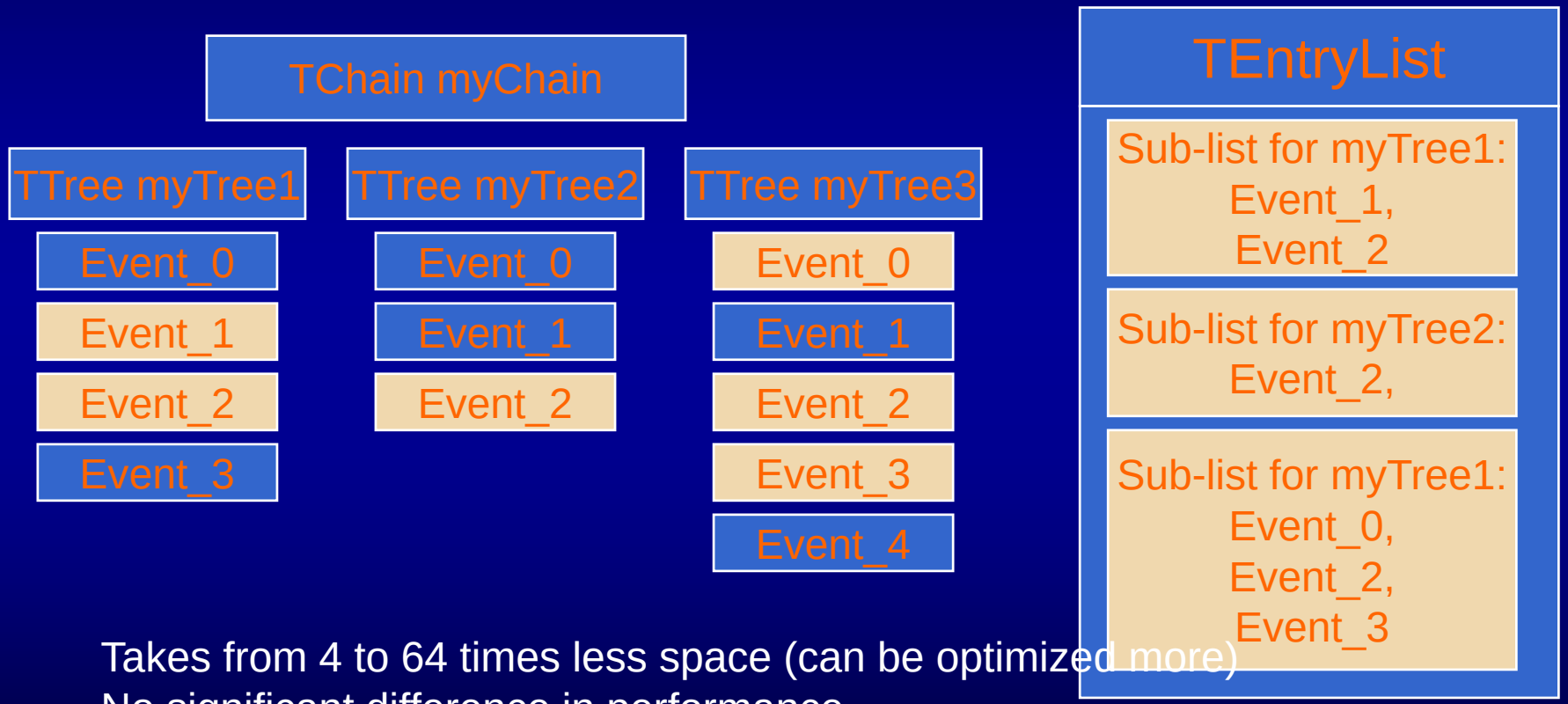


A "naive" implementation:TEventList

Basically, an array of longlong numbers

Because of global indexing in the TEventList, it's not very suitable for parallel processing – it's hard to extract sub-parts that belong to separate trees and process them independently

TEntryList - new



Takes from 4 to 64 times less space (can be optimized more)
No significant difference in performance
Can be only partially loaded in memory
Sub-lists can be extracted and used to compose lists for other chains
Sub-lists can be extracted and processed independently

Improving Debugging:
Memory consumption monitoring

An easy way to access logfiles after a session crashed.

Workers monitor their memory usage and save info in the log file.

New button in the dialog box to display the evolution of memory usage per node in real time.

Client get warned of high usage:

- The session may be eventually killed

PROOF Processing Logs: lxgrid2.gsi.de

Choose workers:

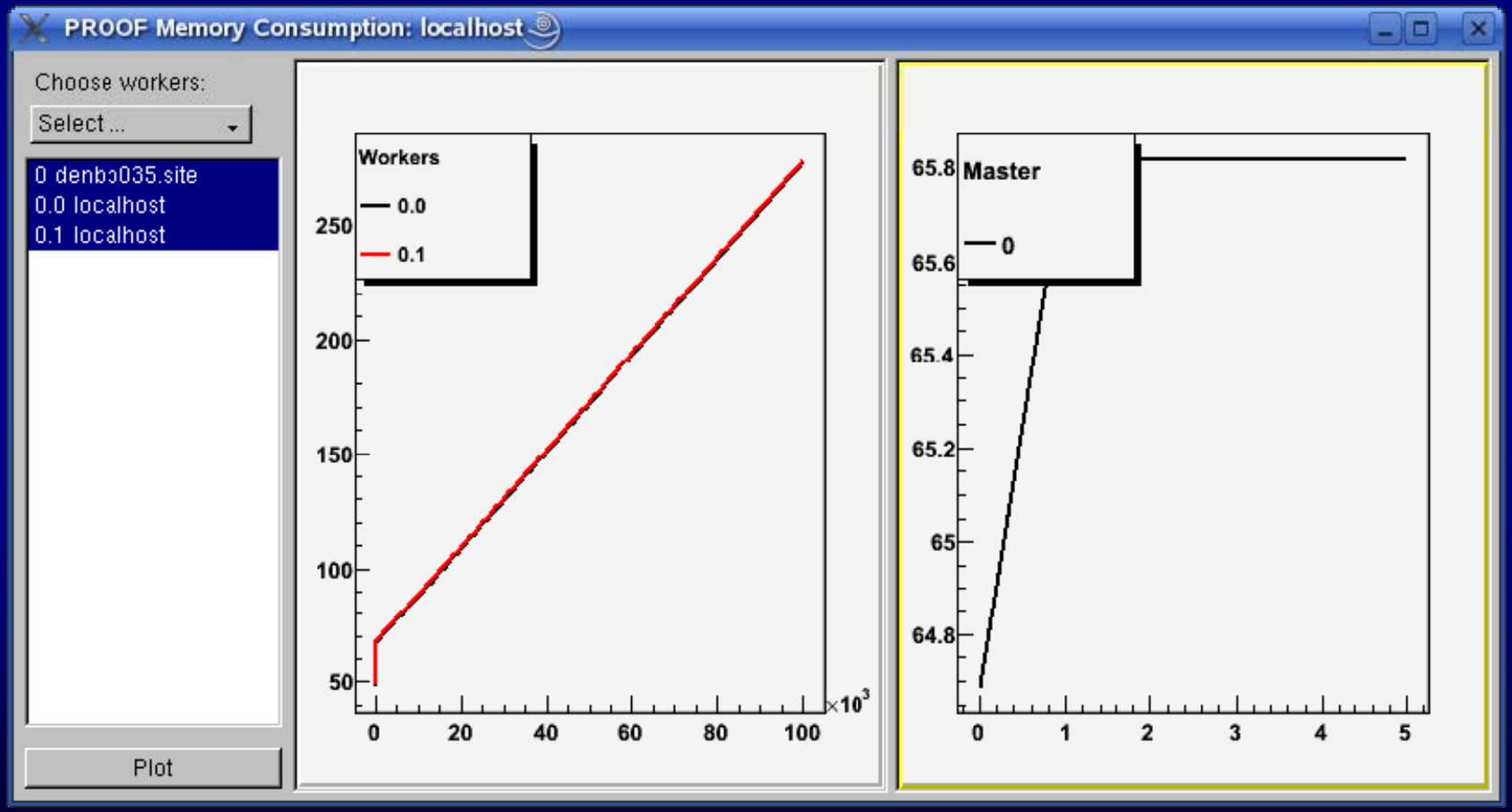
Select ...

0 lxgrid2.gsi.de
0.0 lxb284.gsi.de
0.1 lxb285.gsi.de
0.2 lxb286.gsi.de
0.3 lxb289.gsi.de
0.4 lxb291.gsi.de
0.5 lxb293.gsi.de
0.6 lxb294.gsi.de
0.7 lxb295.gsi.de
0.8 lxb296.gsi.de
0.9 lxb297.gsi.de

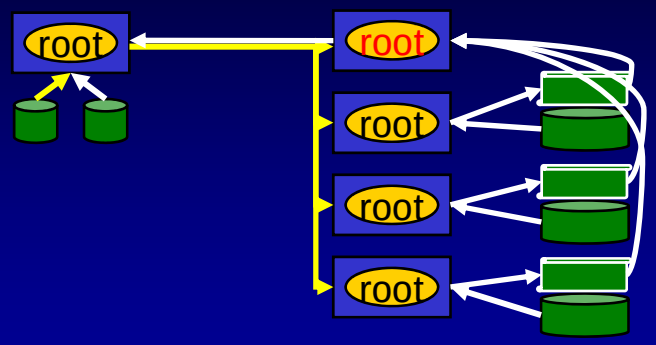
```
// --------- Start of element log -----------------

// Ordinal: 0.1 (role: worker)

// Path: akreshuk@lxb285.gsi.de:1093//data.local2/proof/akreshuk/session-lxgrid2-1213178939-13939/worker-0.1-lxb285-1213

// # of retrieved lines: 100
(displaying lines: 91 -> 100)

// ------------------------------------------------

Info in <EventTree_Proc_leak::Process|Svc> on worker-0.1: VirtualMemory used 268132 event 83000

Info in <EventTree_Proc_leak::Process|Svc> on worker-0.1: VirtualMemory used 269748 event 84000

Info in <EventTree_Proc_leak::Process|Svc> on worker-0.1: VirtualMemory used 271356 event 85000

Info in <EventTree_Proc_leak::Process|Svc> on worker-0.1: VirtualMemory used 272920 event 86000
```

Display

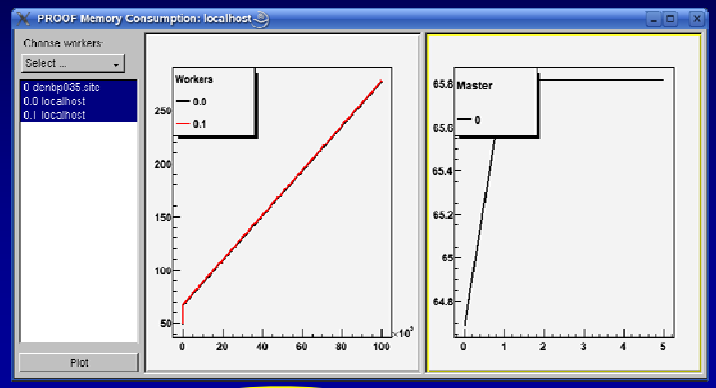
Lines: all From -10 to 0 Grep for: Grep Save to a file: <session-tag>.log Save Close


PROOF Memory Consumption: localhost
Choose workers:
Select ...
0 denbo035.site
0.0 localhost
0.1 localhost
Plot
Workers
0.0
0.1
250
200
150
100
50
0
20
40
60
80
100
×10³
Master
0
65.8
65.6
65.4
65.2
65
64.8
0
1
2
3
4
5

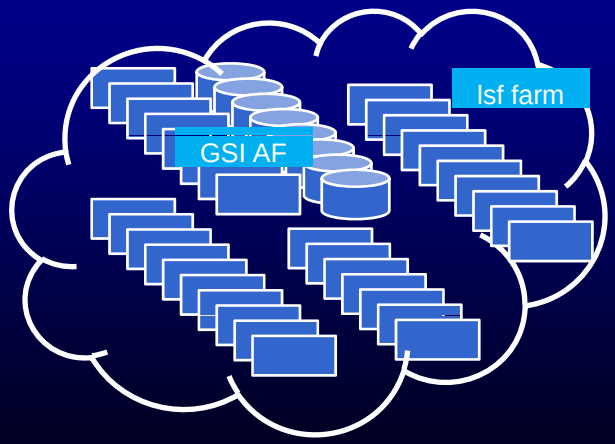
PROOF Overview



Extending PROOF

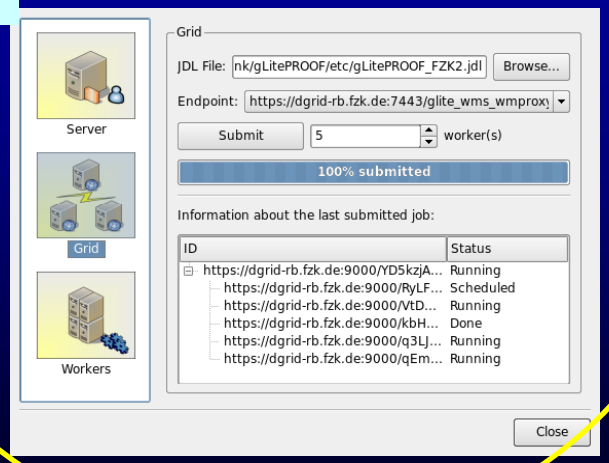


Integration of PROOF in the farm at GSI



PROOF on the Grid

A. Manafov



How to create a PROOF Cluster

Add connecting to the cluster

- TProof::Open("lxb6046")

A PROOF Cluster is a set of demons waiting to start PROOF processes (master, or worker)

It can be setup

1. statically by the system administrator

e.g. CERNAF, GSIAF,...

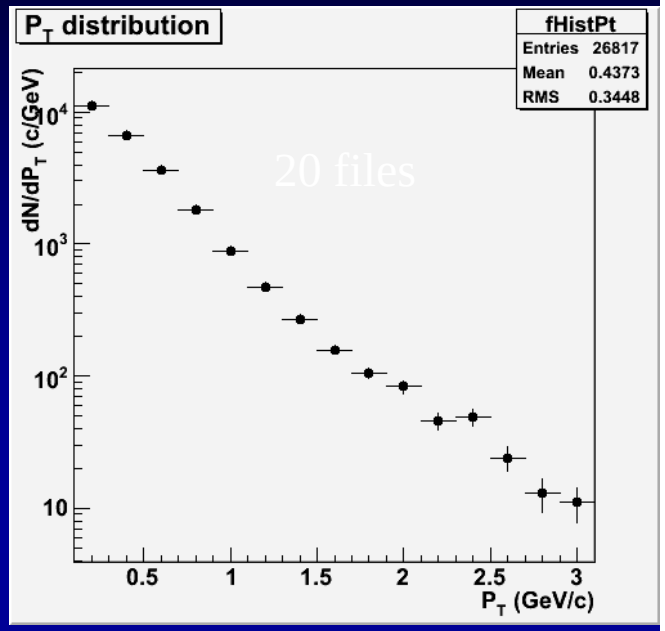
2. by the user

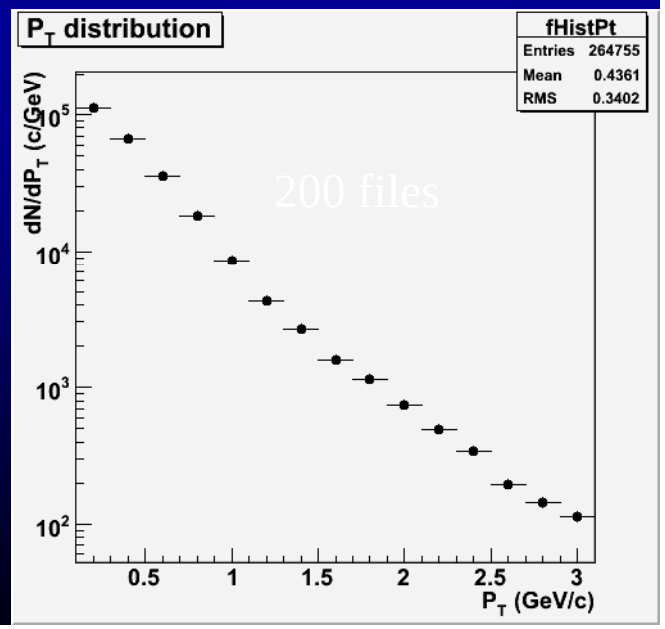
on machines where he can login

multiple processes on a multicore laptop

at GSI we have scripts for our batch system

3. via gLitePROOF on the GRID





gLitePROOF : a gLite PROOF package

A number of utilities and configuration files to implement a PROOF distributed data analysis on the gLite Grid.

Built on top of RGlite:

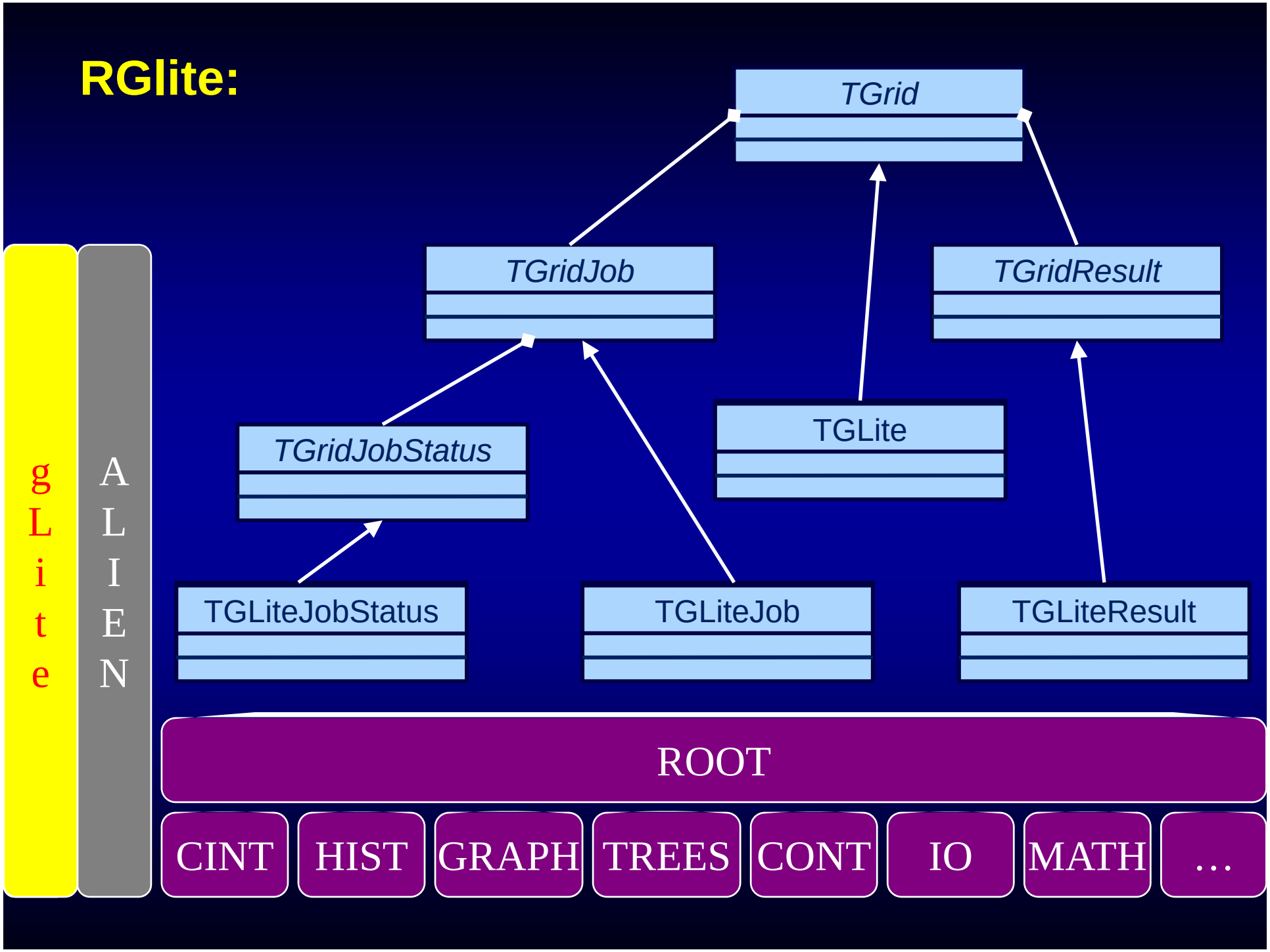
- TGridXXX interface are implemented in RGLite for gLite MW.
- ROOT team accepted our suggestions to TGridXXX interface.

gLitePROOF package

- It setups “on-the-fly” a PROOF cluster on gLite Grid.
- It works with mixed type of gLite worker nodes (x86_64, i686...)
- It supports reconnection.

http://www-linux.gsi.de/~manafov/D-Grid/docz/

RGlite:


TGrid
TGridJob
TGridResult
TGridJobStatus
TGLite
TGLiteJobStatus
TGLiteJob
TGLiteResult
gLite
ALIEN
ROOT
CINT
HIST
GRAPH
TREES
CONT
IO
MATH
…

RGLite example

```
// 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
job->GetOutputSandbox("/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))
   cout << "File " << result->GetFileName(i++));
```

- Changing file catalog directory,
- querying lists of files.

ROOT Version 5.19/02 Release Notes

ROOT version 5.19/02 has been released March 15, 2008. In case you are upgrading from version 5.14, please read the releases notes of version 5.16 and version 5.18 in addition to these notes.

Binaries for all supported platforms are available at:

```
http://root.cern.ch/root/Version519.html
```

Versions for AFS have also been updated. See the list of supported platforms:

```
http://root.cern.ch/Welcome.html
```

For more information, see:

```
http://root.cern.ch
```

RGLITE: A ROOT GRID interface

RGLite plug-in - a ROOT plug-in module, which implements the ROOT Grid interface and offers to ROOT users possibilities to perform a number of operations using gLite middleware from within ROOT. Supported features:

- Workload Management System operations:
  - job submission – normal, DAG and parametric jobs (gLite WMProxy API),
  - smart look-up algorithm for WMP-Endpoints,
  - job status querying (gLite LB API),
  - job output retrieving (Globus GridFTP).
- File Catalog operations (gLite/LCG LFC API):
  - smart session manager,
  - set/query the current working catalog directory,
  - list files, directories and their stats,
  - add/remove files in a catalog namespace,
  - add/remove directories,
  - add/remove replicas from a given file.
- An executive logging.
- Support of an external XML configuration file with according XML schema.

gLitePROOF components:

PROOFAgent – a lightweight, standalone C++ application. Acts as a multifunctional proxy client/server and helps to use proof/xrootd on the Grid worker nodes behind a firewall.

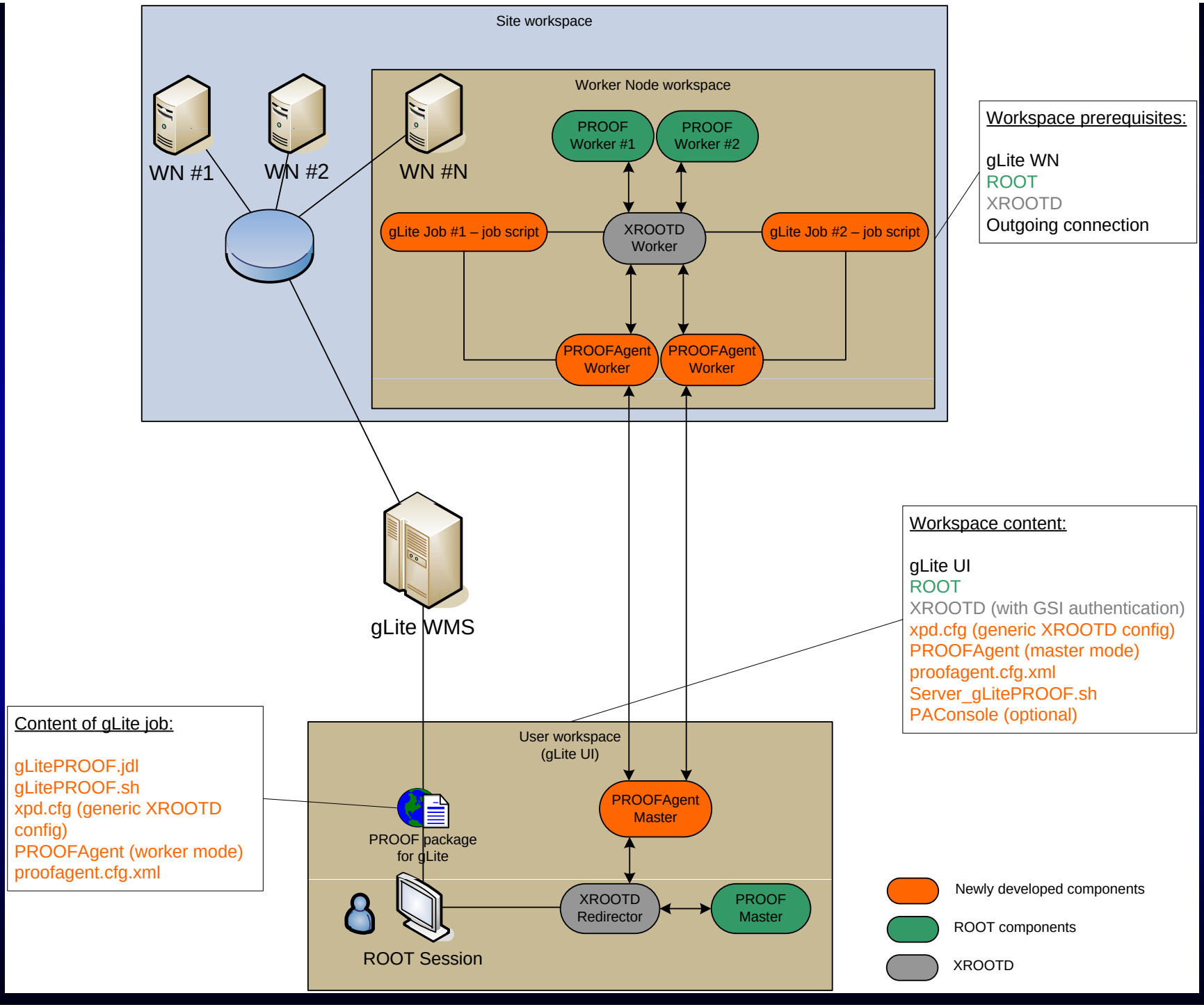
PAConsole – a standalone C++ application, provides a GUI and aims to simplify the usage of PROOFAgent and gLitePROOF configuration files. PAConsole uses GAW to perform gLite job submissions. Users can control jobs directly using ROOT and RGLite plug-in instead of using PAConsole.

xpd.cfg – a generic XROOTD configuration file (configures redirector and remote Grid workers)

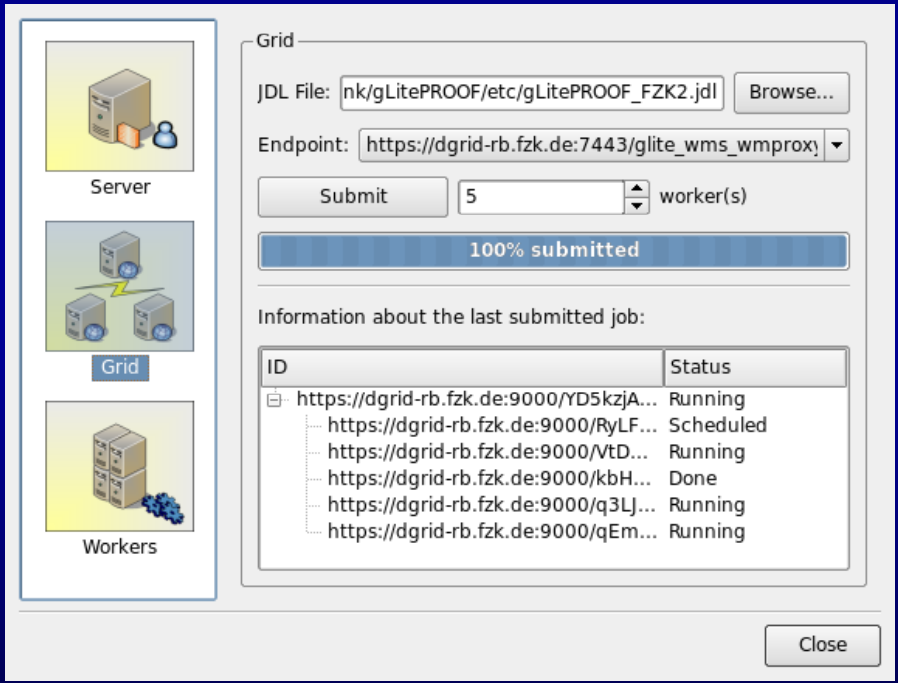
Server_gLitePROOF.sh – a server side script. Helps to start/stop services of gLitePROOF. Could be used via command line or PAConsole GUI.

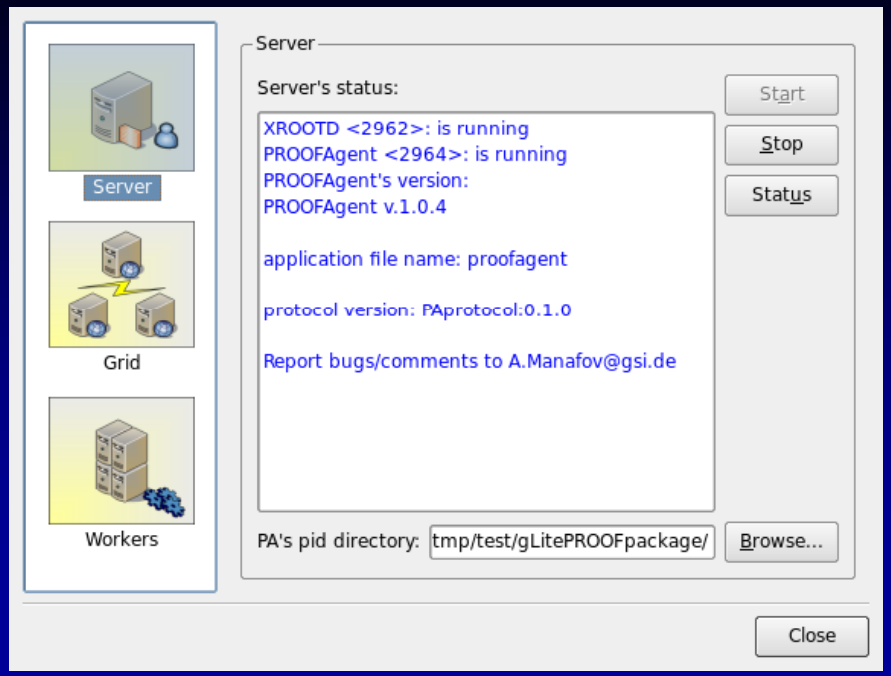
gLitePROOF.jdl – a JDL file, describes a generic, parametric Grid job, which is submitted to gLite and aims to execute gLitePROOF workers on Grid worker nodes.

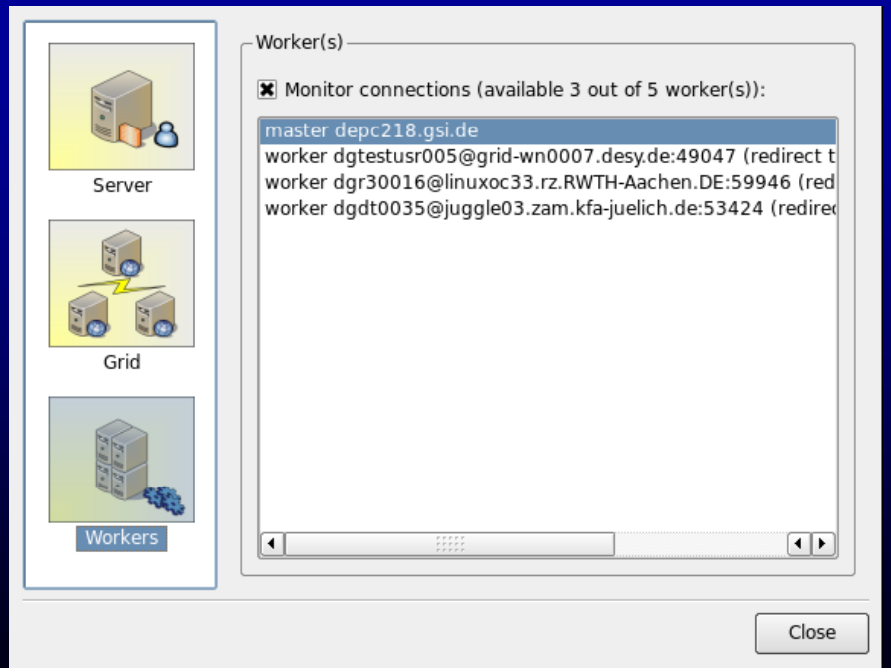
gLitePROOF.sh – a job script. Executed by LRMS on remote workers. Script makes environment recon, uploads necessary packages and starts gLitePROOF services.


Site workspace
Worker Node workspace
WN #1
WN #2
WN #N
PROOF Worker #1
PROOF Worker #2
gLite Job #1 – job script
XROOTD Worker
gLite Job #2 – job script
PROOFAgent Worker
PROOFAgent Worker
Workspace prerequisites:
gLite WN
ROOT
XROOTD
Outgoing connection
gLite WMS
Workspace content:
gLite UI
ROOT
XROOTD (with GSI authentication)
xpd.cfg (generic XROOTD config)
PROOFAgent (master mode)
proofagent.cfg.xml
Server_gLitePROOF.sh
PAConsole (optional)
Content of gLite job:
gLitePROOF.jdl
gLitePROOF.sh
xpd.cfg (generic XROOTD config)
PROOFAgent (worker mode)
proofagent.cfg.xml
User workspace (gLite UI)
PROOF package for gLite
PROOFAgent Master
XROOTD Redirector
PROOF Master
ROOT Session
Newly developed components
ROOT components
XROOTD

PAConsole: a GUI to setup a PROOF Cluster on demand

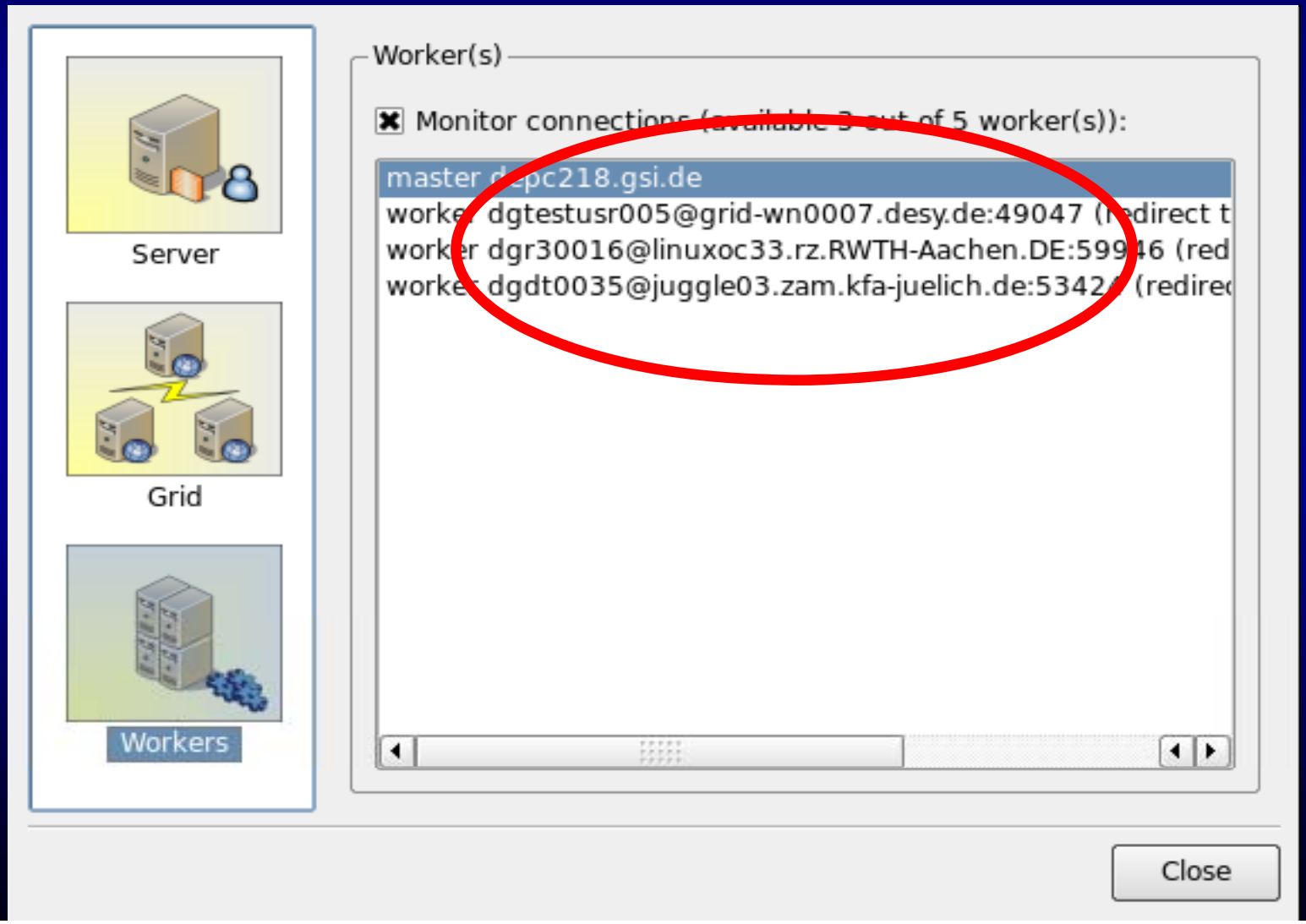






newest feature: supports reconnection

Workers on different sites



https://subversion.gsi.de/trac/dgrid/

RGLite plug-in & gLitePROOF

Search

Login | Settings | Help/Guide | About Trac

Wiki | Timeline | Roadmap | Browse Source | View Tickets | Search | Doc.Portal | Build

Start Page | Index by Title | Index by Date | Last Change

Welcome to RGLite and gLitePROOF

Projects

- glite-api-wrapper - a library, which wraps some parts of gLite API and adds automation and helpers to simplify access to the API.
- RGlite plug-in - a ROOT plug-in module, which implements the ROOT Grid interface and offers to ROOT users a possibility to use gLite middleware from within ROOT.
- PROOFAgent - a multi-functional client/server application. It helps to use proofd on the Grid's worker nods which are behind a firewall, also PROOFAgent has number of additional useful functionality which helps to process PROOF interactive analysis on the Grid.
- PAConsole - a GUI application, helps to manage the PROOFAgent daemon and gLite_PROOF package.
- gLitePROOF - implementation of the PROOF distributed data analysis on the gLite MW.

Documentation

- Source code documentation you can find by pressing "Source Documentation" button in the main menu bar above or by the following link.
- Developers Area

Support

Interactive Data Analysis with PROOF: Summary

ALICE sees PROOF as strategic tool

Integration of the GSI AF into the general purpose batch farm (for Grid and local batch)

To be done: Experience Report

Extending PROOF:

all results go into the ROOT distr.

PROOF on the Grid:

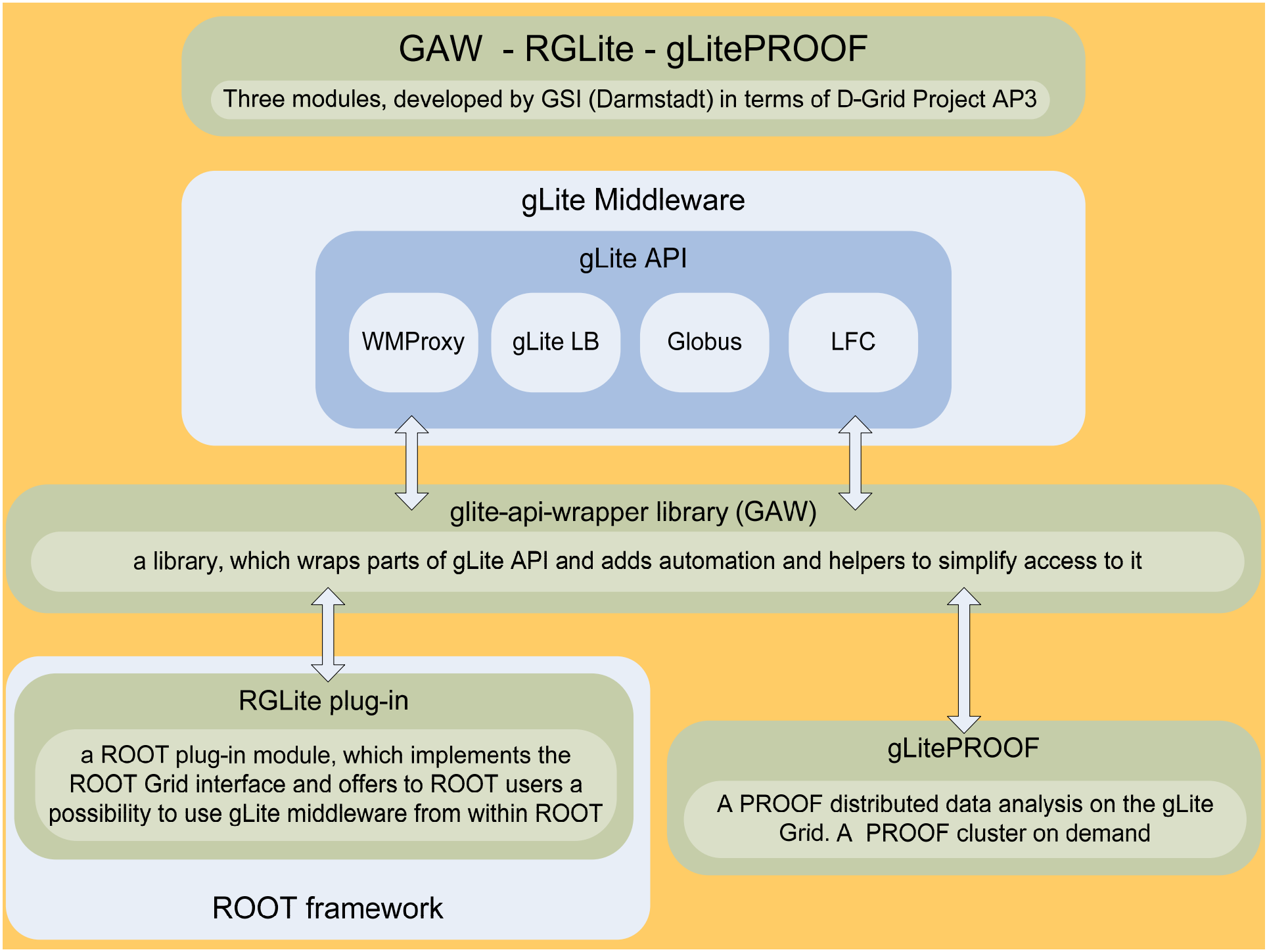
RGlite is in the ROOT distr.

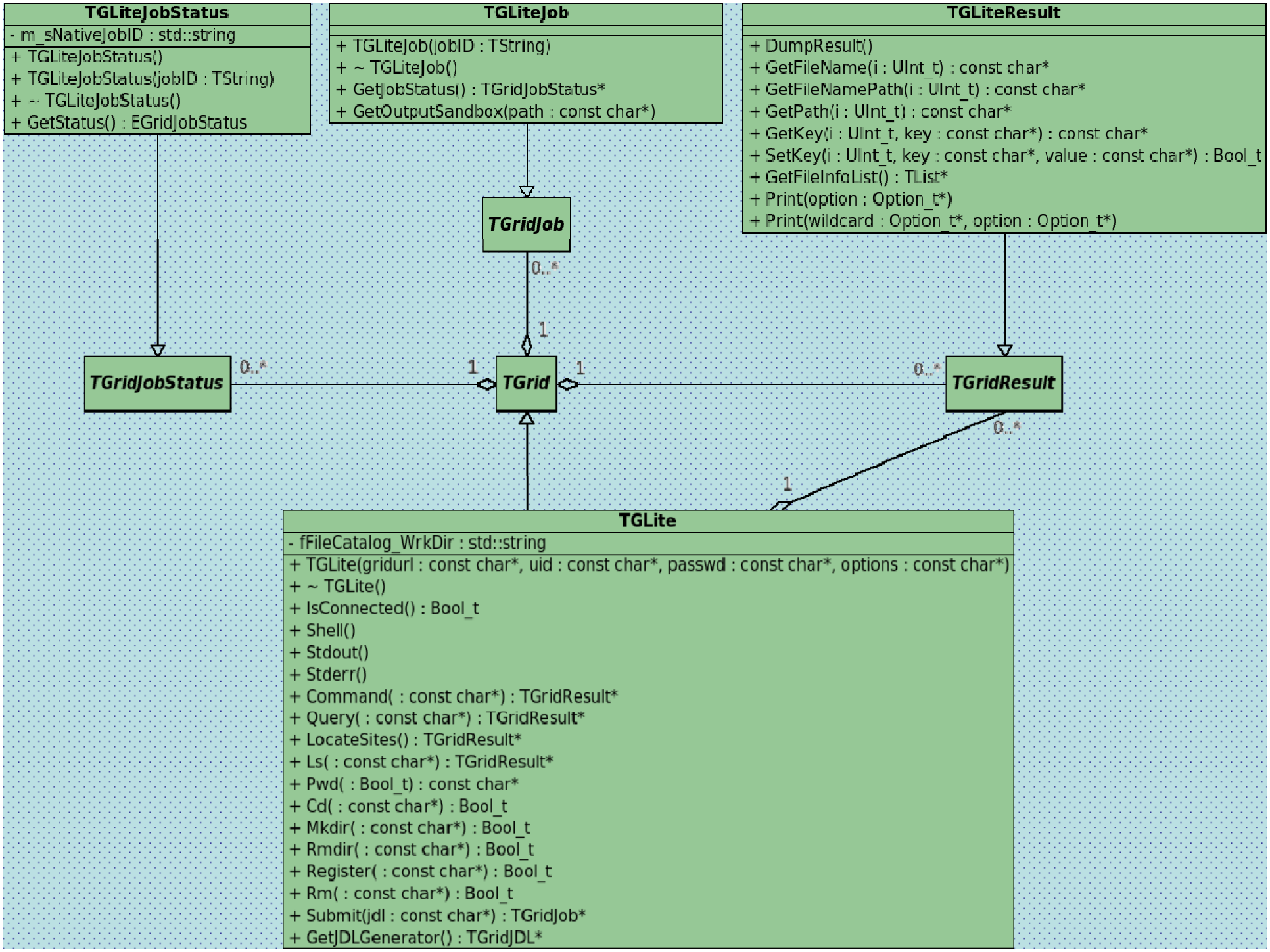
for the other packages see the project wiki:

http://wiki.gsi.de/Grid/RGLiteAndGAW

RGLite plug-in and gLitePROOF package

Anar Manafov, GSI,
HEPCG Workshop, Jun 2008


GAW - RGLite - gLitePROOF
Three modules, developed by GSI (Darmstadt) in terms of D-Grid Project AP3
gLite Middleware
gLite API
WMProxy
gLite LB
Globus
LFC
glite-api-wrapper library (GAW)
a library, which wraps parts of gLite API and adds automation and helpers to simplify access to it
RGLite plug-in
a ROOT plug-in module, which implements the ROOT Grid interface and offers to ROOT users a possibility to use gLite middleware from within ROOT
ROOT framework
gLitePROOF
A PROOF distributed data analysis on the gLite Grid. A PROOF cluster on demand


TGLiteJobStatus
- m_sNativeJobID : std::string
+ TGLiteJobStatus()
+ TGLiteJobStatus(jobID : TString)
+ ~ TGLiteJobStatus()
+ GetStatus() : EGridJobStatus
TGLiteJob
+ TGLiteJob(jobID : TString)
+ ~ TGLiteJob()
+ GetJobStatus() : TGridJobStatus*
+ GetOutputSandbox(path : const char*)
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
0..*
1
TGridJobStatus
0..*
1
TGrid
1
0..*
TGridResult
0..*
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 example

```
// 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
job->GetOutputSandbox("/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.

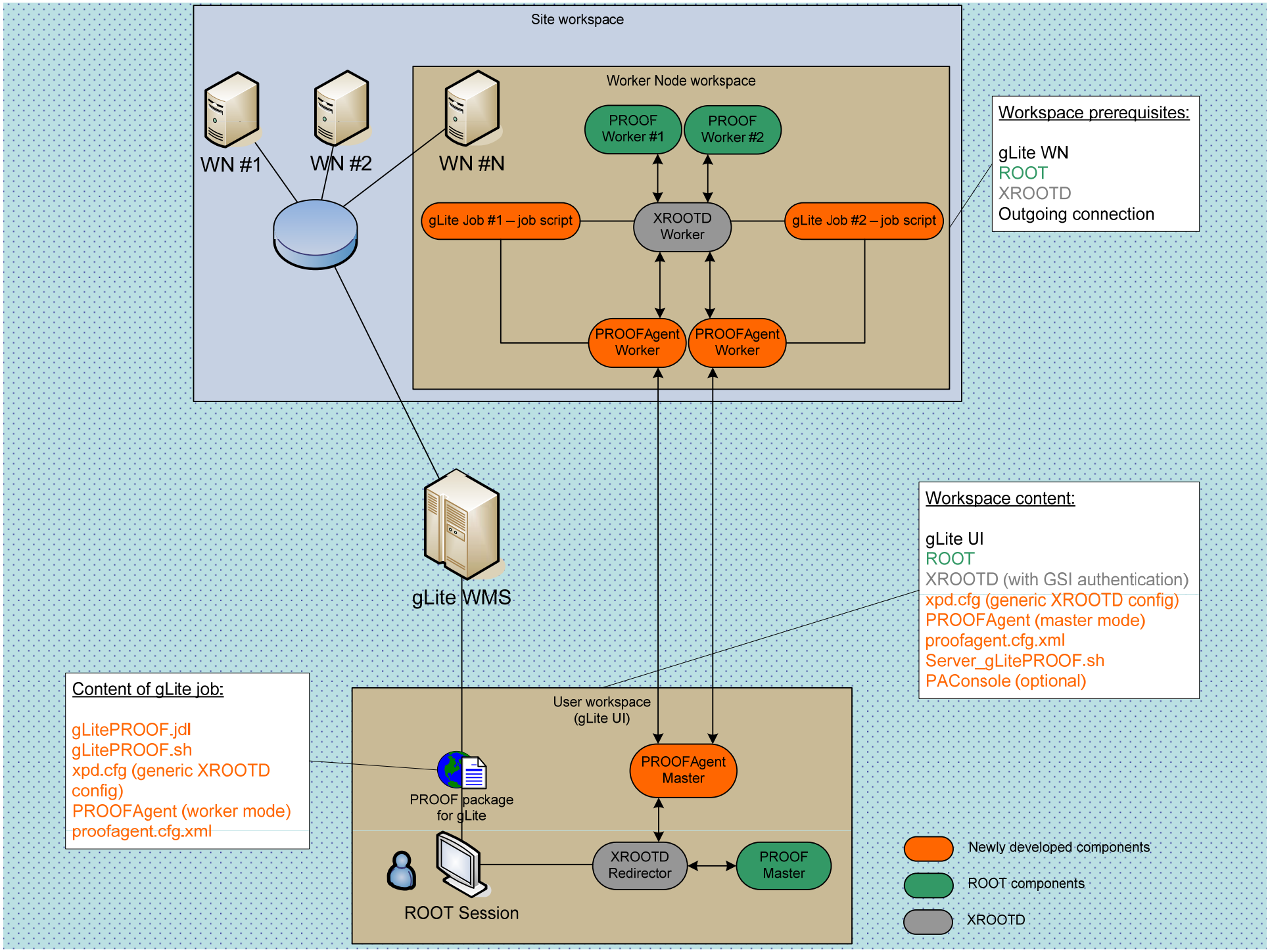
RGLite features

- NEW An official part of the ROOT distribution since ROOT v5.19.
- Workload Management System operations:
  - job submission – normal, DAG and parametric jobs (gLite WMProxy API),
  - smart look-up algorithm for WMP-Endpoints,
  - job status querying (gLite LB API),
  - job output retrieving (Globus GridFTP).
- File Catalog operations (gLite/LCG LFC API):
  - smart session manager,
  - set/query the current working catalog directory,
  - list files, directories and their stats,
  - add/remove files in a catalog namespace,
  - add/remove directories,
  - add/remove replicas from a given file.
- An executive logging.
- Support of an external xml configuration file.

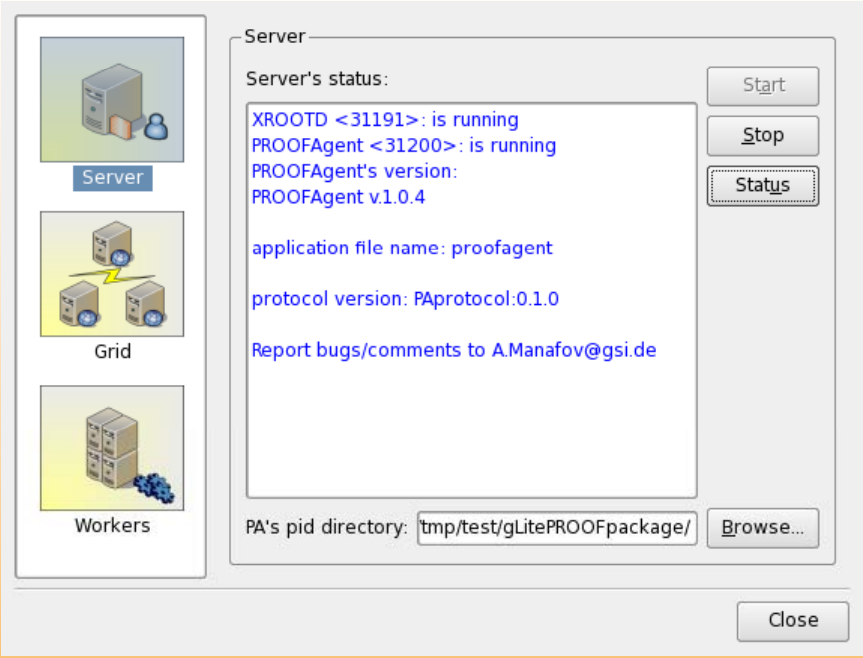
gLitePROOF

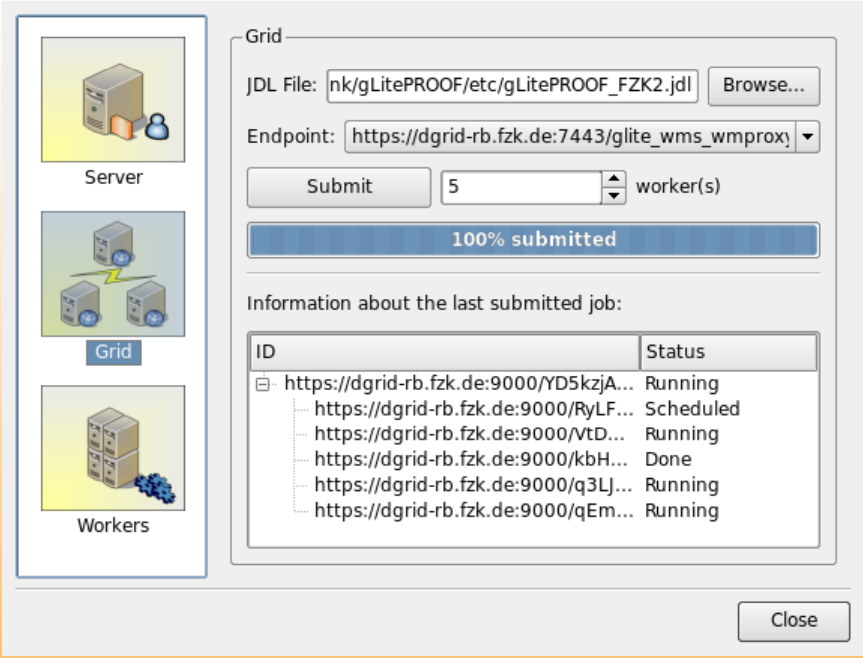
gLitePROOF – a gLite PROOF package.

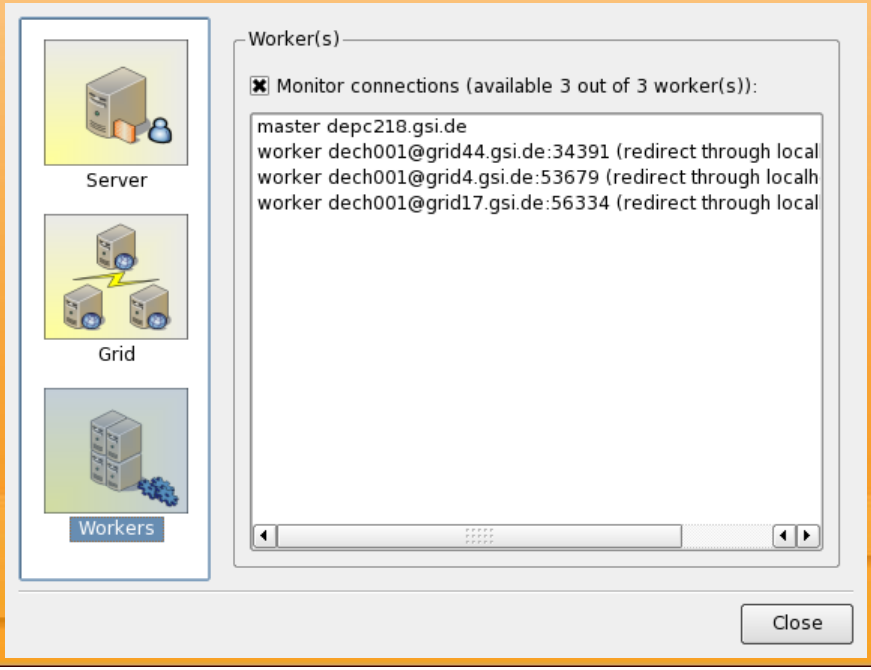
It is a number of utilities and configuration files, developed at GSI in terms of the D-Grid project and aims to implement a PROOF distributed data analysis on the gLite Grid.

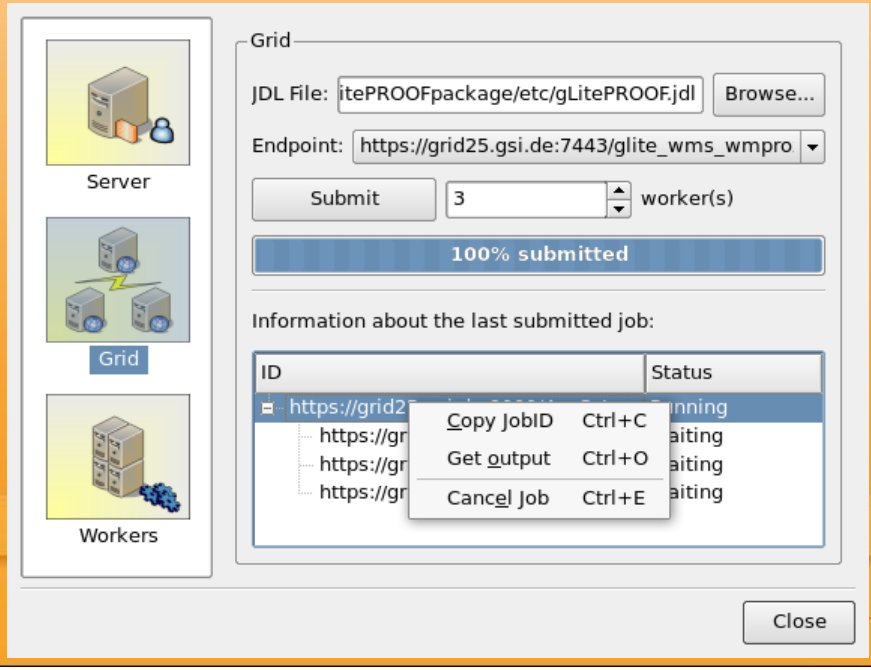

Site workspace
Worker Node workspace
WN #1
WN #2
WN #N
PROOF Worker #1
PROOF Worker #2
gLite Job #1 – job script
XROOTD Worker
gLite Job #2 – job script
PROOFAgent Worker
PROOFAgent Worker
Workspace prerequisites:
gLite WN
ROOT
XROOTD
Outgoing connection
gLite WMS
Workspace content:
gLite UI
ROOT
XROOTD (with GSI authentication)
xpd.cfg (generic XROOTD config)
PROOFAgent (master mode)
proofagent.cfg.xml
Server_gLitePROOF.sh
PAConsole (optional)
Content of gLite job:
gLitePROOF.jdl
gLitePROOF.sh
xpd.cfg (generic XROOTD config)
PROOFAgent (worker mode)
proofagent.cfg.xml
User workspace (gLite UI)
PROOF package for gLite
PROOFAgent Master
XROOTD Redirector
PROOF Master
ROOT Session
Newly developed components
ROOT components
XROOTD

PAConsole - a GUI of gLitePROOF









gLitePROOF summary

- Requirements
  - Client-Side: gLite UI 3.1, ROOT 5.18+,
  - Remote-Side: WMP Endpoint, open out. traffic (Globus ports on WNs).
- Easy in use
  - NEW one-click-installation,
  - NEW user's manual (http://www-linux.gsi.de/~manafov/D-Grid/docz/)
  - user friendly GUI,
  - works out of the box.
- Easy to extend
- Combines resources of the Grid and advantages of PROOF
  - transparency (local ROOT analysis <-> PROOF <-> gLitePROOF),
  - scalability (The basic architecture should not put any implicit limitations on the number of workers).
- Works on heterogeneous machines (gLite WNs)
- Uses xrootd and Grid methods to access data

Summary

- RGLite
  - TGridXXX interface are implemented in RGLite for gLite MW.
  - ROOT team accepted our suggestions to TGridXXX interface.
  - uses WMP job submission.
  - compatible with gLite UI 3.1.
  - NEW a part of the ROOT distributive (since ROOT v5.19).
- gLitePROOF package
  - NEW New stable release.
  - NEW Two officially registered users from ATLAS (CPPM and LAL, France).
  - setups “on-the-fly” a PROOF cluster on gLite Grid.
  - works with mixed type of gLite worker nods (x86_64, i686...).
  - supports reconnections.
  - provides GUI.
- Use of Agile methods of software development:
  - Continues integration (automated builds on SLC3, SL4, and F8, nightly builds).
  - Unit tests.
  - Projects metrics.
  - Task tickets and sprints…

Trac: https://subversion.gsi.de/trac/dgrid

Wiki: http://wiki.gsi.de/cgi-bin/view/Grid/RGLiteAndGAW

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung