Journal of Physics: Conference Series **396** (2012) 032044 doi:10.1088/1742-6596/396/3/032044

PEAC - A set of tools to quickly enable PROOF on a cluster

G. Ganis
CERN, PH-SFT, CH-1211 Geneva 23, Switzerland
E-mail: gerardo.ganis@cern.ch

M. Vala
Institute of Experimental Physics, Slovak Academy of Science, Košice, Slovakia
E-mail: martin.vala@cern.ch

Abstract. With advent of the analysis phase of LHCdata-processing, interest in PROOF technology has considerably increased. While setting up a simple PROOF cluster for basic usage is reasonably straightforward, exploiting the several new functionalities added in recent times may be complicated. PEAC, standing for PROOF Enabled Analysis Cluster, is a set of tools aiming to facilitate the setup and management of a PROOF cluster. PEAC is based on the experience made by setting up PROOF for the ALICE analysis facilities. It allows to easily build and configure ROOT and the additional software needed on the cluster, and may serve as distributor of binaries via XROOTD. PEAC uses PROOF-ON-DEMAND (POD) for resource management (start, stop or daemons). Finally, PEAC sets-up and configures dataset management (using the AFDSMGRD daemon), as well as cluster monitoring (machine status and PROOF query summaries) using MONALISA. In this respect, a MONALISA page has been dedicated to PEAC users, so that a cluster managed by PEAC can be automatically monitored. In this paper we present and describe the status and main components of PEAC and show details about its usage.

1. Introduction

With the advent of the analysis phase of LHC data-processing, several PROOF-enabled [1] facilities have seen the light, with varying configurations. The ALICE experiment has developed an ALICE Analysis Facility [2] which features GRID-aware dataset staging and monitoring. Groups of the other experiments have developed their own facilities following the local needs and resources. These different experiences allowed to identify the main components of a PROOF-enabled analysis facility for LHC data analysis: resource management (daemon control, software distribution and installation), local storage management, dataset meta-information management and monitoring. While setting up a simple PROOF cluster for basic usage is reasonably straightforward, exploiting the all functionalities and optimally configuring the several components can be complicated. PEAC, standing for PROOF Enabled Analysis Cluster, is a set of tools aiming to facilitate the setup and management of a PROOF cluster. PEAC is based on the experience made by setting up PROOF for the ALICE analysis facilities. PEAC allows to easily build and configure ROOT and the additional software needed on the cluster. It sets up XROOTD [3] systems for binary distribution and to manage the local storage available

on the nodes. It uses PROOF-ON-DEMAND (POD) [4] for resource management (start, stop or daemons) and enables dataset management using the AFDSMGRD daemon [5]). Finally, PEAC enables cluster monitoring (machine status and PROOF query summaries) using MONALISA [6], posting the monitoring data to a dedicated page, so that any cluster managed by PEAC can be automatically monitored.

This paper is organized as follows. In Section 2 we describe the typical components of a PROOF enabled facility. In Section 3 we present the basic PEAC concepts. In Section 4 we list the third-party software which will be installed and used by PEAC. In Section 5 we describe the main use-cases addressed by PEAC. In Section 6 we describe the usage of the `peac` script. In Section 7 we show the main steps to get a PROOF-enabled facility managed by PEAC and in Section 8 the possibilities of advanced configurations. In Section 9 we describe how to access to the monitoring information. Finally, in Section 10 we summarize and outline the future directions of development.

2. Components of a PROOF-enabled facility

The tools described in this paper address the case of a multicore machine or farms of multicore machines. Typical figures for today's commodity hardware are 8 or 16 cores, possibly hyper-threaded, with 2 GB/core RAM, 1 Gb/s network connection and a few TB of local disk space. The machines may share a file system, possibly read-only, like AFS or CVMFS. The service components required to effectively run PROOF for analysis are:

- ROOT(one or more versions) and XROOTD, used to run PROOF;
- additional software required by the specific activity, e.g. software to read the data or analysis tools;
- software to manage the local disk pool;
- software to manage file meta-information, typically at dataset level;
- some level of monitoring (when more than one machine).

The installation and configuration of these components is typically done by the administrator of the cluster of the user itself. The difficulty of the installation/configuration step depends on the component, but some advanced skills are typically required.

3. Basic PEAC concepts

The idea behind PEAC is to simplify the setup of a PROOF-enabled facility with full functionality. There are basically two basic ideas who drove the development. The first is that well-designed command line tools are the most effective ones to manage services. The examples we have in mind are `svn` or `git` commands, or, to get closer to PROOF, the `pod` command [4]. The second is that starting from a good set of default configurations is typically a recipe to success. Bad defaults with hidden or non flexible configurations are problematic because it is typically difficult to explain to people unskilled - and uninterested to such details - how to get away from those; this is typically a killer.

In order to classify software requirements, PEAC extends the concept of *Virtual Organization* (VO) and introduces VO inheritance. Each new cluster is a new VO identified by a unique name and can inherit its settings from an existing VO; for example, ALICE users will certainly inherit the VO_ALICE default settings distributed with PEAC, eventually adding private *ad hoc* extensions. Packages are identified by tags having the following format:

VO_*[VO_name]*@*[package_name]*::*[version]*

For example, for ROOT 5.34 and ATLAS settings, VO_ATLAS@ROOT::v5-34-00 .

Details about the VO handling can be found in the PEAC documentation [7].

4. Third-party software

In order to achieve its purposes PEAC installs and configures several third-party components which we will briefly review in this section.

4.1. XROOTD

The XROOTD package [3] has been initially developed to improve the efficiency and smoothness of accessing ROOT files and therefore, since its beginnings, it has been intimately connected with ROOT. ROOT has been, *de facto*, for a long time its primary distribution channel. However, XROOTD is ROOT-agnostic and, having grown stable and popular, is now distributed through the most popular standard channels[1]. As a consequence, it has been removed from ROOT starting with version 5.32/00. XROOTD has nonetheless remained an important external component of ROOT: it is needed to build the related `TFile` implementation plug-in, the PROOF daemon and its client.

PEAC makes an extensive usage of XROOTD as data access system. XROOTD is used to provide a common view of the storage available on the nodes. It is also used as file residency manager of such storage, including copying from other storage systems. In addition, XROOTD is also used to serve binaries available on the cluster to the cluster members or to other PEAC installations.

4.2. PROOF-ON-DEMAND

PROOF-ON-DEMAND, POD [4], is a toolkit to set up PROOF on any resource management system using plug-in technology. Plug-ins for the most common back-ends are available. A special plug-in, labelled SSH, addresses the case of a set of nodes accessible via password-less SSH and provides a convenient way to handle the relevant daemons on any set of connected nodes. The POD-SSH plug-in is used by PEAC to manage the PROOF cluster it sets up.

4.3. AFDSMGRD

AFDSMGRD is a open-source daemon [5] in charge of making sure that the files belonging to the registered datasets are available for opening on the cluster. Here the concept of dataset is the one represented in ROOT by the `TFileCollection` class. PROOF provides a dataset manager class which works in conjunction with a file-based repository, where each dataset `TFileCollection` object is saved in a separate ROOT file. The daemon monitors the repository and acts whenever a dataset is added or modified, issuing the relevant *staging* operation when required. The way dataset managements works is schematically shown in Figure 1.

4.4. MONALISA

The MONALISA system [6] is a fully distributed monitoring system based on agent technology and featuring a dynamic distributed service architecture. Widely used at LHC, MONALISA comes with a set sensors to monitor the node resources which, integrated with the information posted by the application being run, provide a complete view of the status of the cluster during running and of the cluster usage.

4.5. PARALLEL-SSH

PSSH [8] provides a convenient way to run all the OPENSSH-related applications in parallel.

[1] Since version 3.0.2, XROOTD is included in the major LINUX distributions and provides repositories for SLC systems [3].

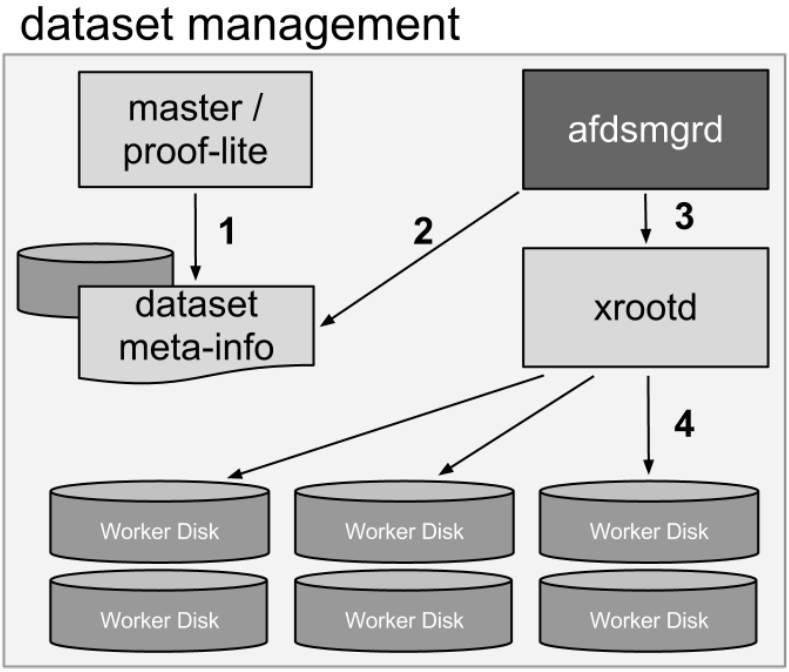

Figure 1. The way the dataset manager works. 1: the user registers the dataset via the master or a local PROOFLITE session; 2: the AFDSMGRD daemon analysis the dataset repositories and finds out the a dataset has been added or modified; 3: the daemon queries XROOTD about the new files; 4: eventually, XROOTD issues the required actions.

5. Use-cases addressed by PEAC

PEAC has been developed having in mind two main use-cases. The first use-case, which we will be referring to as PROOFLITE in the following, is the case of a end-user with a standard multicore machine on its desk, possibly with some local storage. The second use-case, the *standard* case, is the case of analysis group having a cluster of machines to be used for analysis and local storage.

5.1. PROOFLITE

The PROOFLITE use-case is the case of a user having a powerful private multicore machine - a desktop or a laptop - where she/he would run PROOFLITE to analyze the data or to perform the tasks needed by her/his analysis. In addition to setup ROOT - and therefore PROOFLITE - what one needs in this case is the ability to install smoothly the additional code required, for example, by her/his experiment and to be able to access the data, which typically are stored remotely. The situation is depicted in Figure 2.

The local storage can be used to cache the data files needed by the analysis. It is known that a simple XROOTD daemon can be used for *staging*, but its configuration may be complicated for the average end-user analyst. The additional software may be already available in the binary form from shared file systems or binary servers, a smooth access to which is required.

5.2. Standard PROOF

The *standard* use-case is the case of a cluster of - typically homogeneous - worker nodes, with a non-negligible amount of local storage. This case is depicted in Figure 3.

Also in this case access to potentially available software binaries needs to be smooth. However,

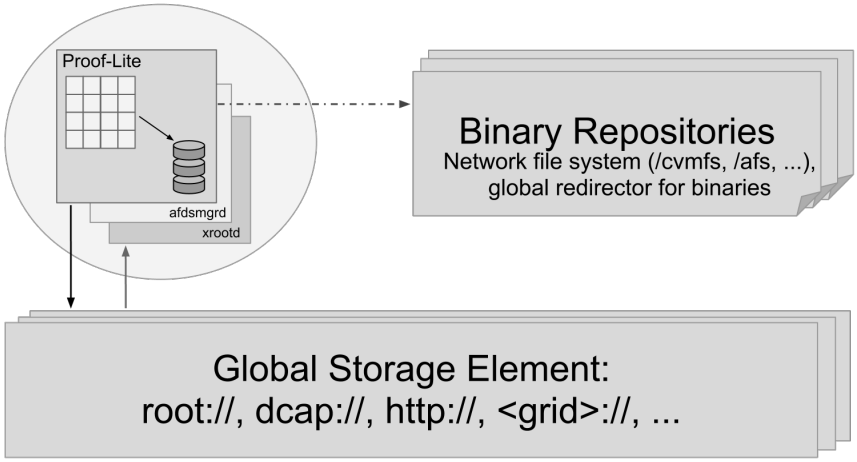

Figure 2. The PROOFLITE case.

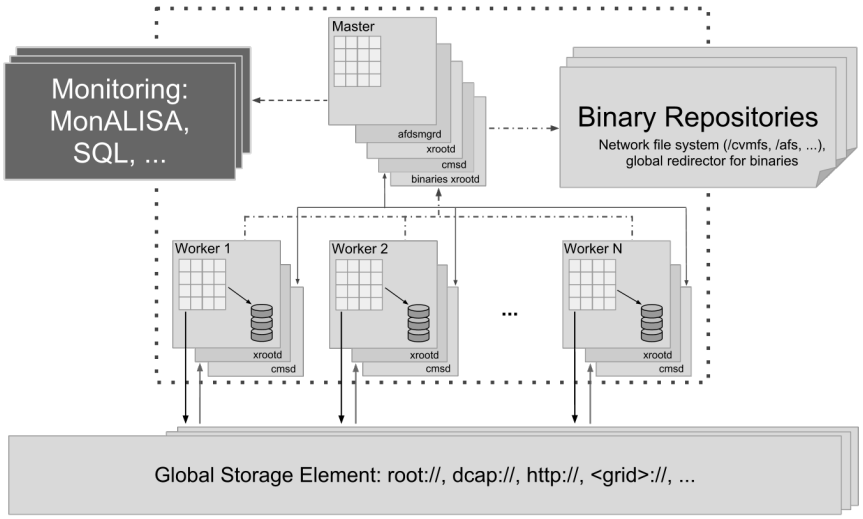

Figure 3. The standard PROOF case.

the master, i.e. the node chosen to be the entry point to the system, will serve also as build/retrieve machine and will make binaries available to the workers.

The local storage will be used as cache for the incoming data. However, the XROOTD system will be in this case more complex, because it has to aggregate the different storage systems in one single virtual entity.

In addition, a cluster of nodes will be used by more than one user. Setting up a monitoring system for the machine parameters relevant to analysis (CPU, RAM, network, disk, ...) and for the PROOF internal activities, is therefore also required.

6. The `peac` command

As mentioned above, the idea is to be able to do all the configuration and control operations with a simple and effective command line interface. In the case of PEAC the command is called `peac` and has the following syntax:

$ `./peac` *command option [args]*

The main commands are the following:

sw	allows to manage software synchronization and installation on the master machine;
repos	allows to manage the local VO configuration repositories;
init	used to setup the different nodes, *lite*, master or workers;
proof	used to control the PROOF daemons;
xrd	used to control the XROOTD daemons;
stager	used setup and control the dataset manager daemon
mon	used to manage the MONALISA services.

PEAC provides online help for each of the commands by just typing `./peac` *command*, e.g.

```
./peac sw
   ./peac sw list
   ./peac sw install VO_ALICE@ROOT::v5-28-08d
   ./peac sw list-to-remove
   ./peac sw autoremove
   ./peac sw add-package LITE VO_ALICE@AliRoot::v5-03-15-AN
   ./peac sw defaults [show|set]
```

7. Eight step way to a Peac managed cluster

In this section we should an example of how to setup a *standard* cluster with PEAC in 8 steps. The commands are run from an unprivileged account on the master machine and it is assumed that the master can logon password-less via SSH on the the other nodes.

7.1. Step 1: Get the `peac` *script on the master node*

The first step consists in creating a dedicated directory on the master and retrieving the `peac` script in there from the ROOT FTP server. We will name this directory PEAC:

```
master:~/ mkdir PEAC; cd PEAC
master:~/PEAC wget ftp://root.cern.ch/root/peac/peac
```

7.2. Step 2: Configure it

Next we need to run the `peac` script to enter some relevant information about the facility: a (unique) name, the coordinates of the administrator, the mount points of the storage disks on the nodes[2]:

```
master:~/PEAC ./peac
Enter your PEAC type[master,lite]: master
PEAC_TYPE = master
Enter your PEAC name: MyPeac
Enter your admin name: admin
Enter your admin email: admin@my.place
Enter your data dirs: /data0 /data1
Doing init of peac scripts ...
```

[2] It is assumed that by default all the nodes have the same storage structure; it is, of course, possible to customize the XROOTD config files to support non-homogeneous storage configuration on the nodes.

7.3. Step 3: Change the ROOT *version (optional)*

This optional step allows to choose a specific ROOT version. By default PEAC installs the latest stable version, i.e. the last tagged patch on the latest production release. The default can be changed with:

```
master:~/PEAC ./peac sw defaults set
```

Optionally, at this point one can define a remote repository or a shared file system where to look for the binaries.

7.4. Step 4: Setup the master

The next step is to set up the master, which means making sure that the software required is available (built, if needed) and installed. To do this just issue the *init* command:

```
master:~/PEAC ./peac init
```

This will also setup and start the relevant MONALISA sensors.

7.5. Step 5: Setup the workers

After having set up the master we need to do the same for the workers; to this end PEAC will use as much as possible what it already used on the master. The first thing is to define the list of worker nodes; this must be done in a file named `wk.cf`, containing one worker FQDN per line, e.g.

```
cernvm26.cern.ch
cernvm28.cern.ch
cernvm30.cern.ch
cernvm32.cern.ch
cernvm34.cern.ch
cernvm36.cern.ch
```

We then need to issue *init workers* to insure that everything is installed correctly on those machines:

```
master:~/PEAC ./peac init workers
Doing init workers ...
Generating workers config ...
Starting xrootd on MASTER ...
status xrootd [sw] [ RUNNING ]
```

Again, this includes setting up and starting the relevant MONALISA sensors.

7.6. Step 6: Bringing up PROOF

At this point we are ready to start PROOF. As mentioned above, for this purpose PEAC uses POD and therefore starts PROOF using the POD defaults. If specific configuration directives are required, the snippets of configuration files including those directives can be registered in `proof/proof.config.list` and added under `proof/proof.config`, so that they are automatically picked-up by POD.

Once happy with the configurations we just issue *proof start*:

```
master:~/PEAC ./peac proof start
Starting PoD server...
```

We can then test the installation:

```
master:~/PEAC root -b
root [] TProof::Open(localhost)
Starting master: opening connection ...
...
PROOF set to parallel mode (48 workers)
(class TProof*)0x16d4db80
```

7.7. Step 7: Bringing up XROOTD

To finalize the setup of the XROOTD system we issue *xrd start*. The configuration files are located under `etc`:

```
master:~/PEAC ls etc/xrd*.cf
etc/xrd-cns.cf  etc/xrd-global-rdr.cf  etc/xrd-manager.cf  etc/xrd-simple.cf
```

these files can/should be edited for any advanced configuration.

To simplify the configuration of the File Residency Manager (FRM) component enabling staging of files from a external data servers - e.g. via wget or via xrdcp - PEAC provides dedicated options to the *xrd* command: *gen-frm* and *update-frm*. For example:

```
master:~/PEAC ./peac xrd gen-frm http://root.cern.ch/files
PREFIX= http://root.cern.ch/files
CMD = wget
ARGS = -q
Script ./etc/xrd-frm.sh was generated .
master:~/PEAC ./peac xrd update-frm
```

sets up staging from the ROOT HTTP server.

Once all configurations are in place, the XROOTD system needs to started (or restarted, if already running and the configuration changed) by issuing the appropriate command, e.g.

```
master:~/PEAC xrd restart
```

7.8. Step 8: Setting up and starting the stager

The last step is to configure and start the AFDSMGRD daemon. This is done using *peac stager*. The default configuration file must be generated first:

```
master:~/PEAC ./peac stager generate-config
Config './PEAC/etc/afdsmgrd.conf' created
```

Configuration can be file-tuned by editing `etc/afdsmgrd.conf`. We can then start the daemon:

```
master:~/PEAC ./peac stager start
```

The daemon will start parsing regularly the PROOF dataset directory, making sure that the files required are available on the cluster.

8. Advanced configurations

While the default configurations should sufficient in most of the cases, *ad-hoc* setups may be required, especially when working with experiment frameworks. These ad-hoc settings are typically related to dedicated environment settings, but they may also consist in the necessity to provide private packages (or private versions of given packages) as PAR files.

8.1. Environment
PEAC uses PoD to start daemons, all PoD techniques described in [4] to configure the daemon environments are directly available. For example, for ATLAS this is typically used to set up the environment from CVMFS.

8.2. Packages as PAR files
Additional software can be provided via PAR files. Examples of ALICE and ATLAS packages are distributed with PEAC˙ATLAS example: the following commands

```
master:~/PEAC ./peac sw add-package MyPeac  VO_ATLAS@RootCore::v0.0.2
Repository MyPeac doesn't exist.
    Do you want to create it? (y or Ctrl-C to exit) y
Package VO_ATLAS@RootCore::v0.0.2 was added.

master:~/PEAC cat sw/repos/VO_MyPeac/soft.conf
VO_ATLAS@RootCore::v0.0.2
```

activate, during init, the creation of a PAR file enabling the ATLAS package `RootCore` and its dependencies.

9. Access to monitoring information
As mentioned in Sections 1 and 7, PROOF is configured by PEAC to send monitoring information to MONALISA; standard MONALISA sensors are started on the nodes so that detailed information about CPU load, RAM usage, disk I/O, I/O wait, network I/O, etc ... is automatically collected from each cluster started with PEAC. A dedicated test machine - `peacmon.cern.ch` - has been instrumented to collect the information[3], so that the collected information can be browsed through dedicated MONALISA pages right after PROOF is started on the cluster.

10. Summary and future work
We have described the current status of a set of tools - collectively named PEAC - to manage a fully-featured PROOF-based analysis facility. These tools have been derived from existing similar ones used in the ALICE Analysis Facilities [2], and are already being used to manage a few clusters in Russia, Košice and CERN. PEAC documentation is available at [7].

The main current development activities for PEAC are related to the general consolidation and refinements of scripts options, in particular to what relates the PROOFLITE case, the usage of externally build binaries (e.g. `/cvmfs`) and integration of MONALISA with existing builds, the setup of the stager and the flexibility of the FRM setup.

References
[1] See http://root.cern.ch/drupal/content/proof. See also I.Antcheva et al., Comput.Phys.Commun.180:2499-2512,2009. See also http://root.cern.ch/.
[2] The ALICE Analysis Facilities web site can be found at http://aaf.cern.ch .
[3] See http://xrootd.org .
[4] See http://pod.gsi.de .
[5] See http://code.google.com/p/afdsmgrd/ .
[6] See http://monalisa.caltech.edu/monalisa.
[7] See http://root.cern.ch/drupal/content/peac.
[8] See http://code.google.com/p/parallel-ssh/.

[3] The `peacmon.cern.ch` machine is currently directly accessible from inside the cern.ch only; it is planned to have the collector machine accessible from anywhere for a near future.