PROOF on Demand

Peter Malzacher, Anar Manafov

GSI Helmholtzzentrum für Schwerionenforschung GmbH, Planckstr. 1, 64291 Darmstadt, Germany

E-mail: A.Manafov@gsi.de

Abstract. PROOF on Demand (PoD) is a set of utilities, which allows starting a PROOF cluster at user request, on any resource management system. It provides a plug-in based system to use different job submission frontends, such as LSF or gLite WMS. Main components of PoD are the PROOFAgent and the PAConsole. PROOFAgent provides the communication layer between the PROOF master on the local machine and the PROOF workers on the remote resources, possibly behind a firewall. PAConsole provides a user-friendly GUI, which is used to setup, manage, and shutdown the dynamic PROOF cluster. Installation of PoD is simple and doesn't require administrator privileges, and all the processes run in user space. PoD gives users, who don't have a centrally-administrated static PROOF cluster at their institute, the possibility to enjoy the full power of interactive analysis with PROOF.

1. Introduction

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.

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.

With all its advantages the Grid model anyway introduces a delay in obtaining the results. Users receive them after a given time which is the execution time of the program itself plus an overhead that tends to be bigger than in case of batch systems.

In addition to the Grid-like analysis many experiments provide a local interactive analysis using the Parallel ROOT Facility (PROOF) [4]. PROOF is an extension of the ROOT system for parallelizing the application execution. The default installation of PROOF is a static cluster. But there is an idea to use PROOF in more user-friendly and convenient way – a dynamic cluster on request. This is the reason PROOF on Demand (PoD) [5] development has been started.

PoD is a specially designed solution to provide a PROOF cluster on the fly.

2. Overview

PoD is a set of utilities, scripts, and configuration files. It provides a possibility to set up a PROOF cluster dynamically on Grid and batch systems.

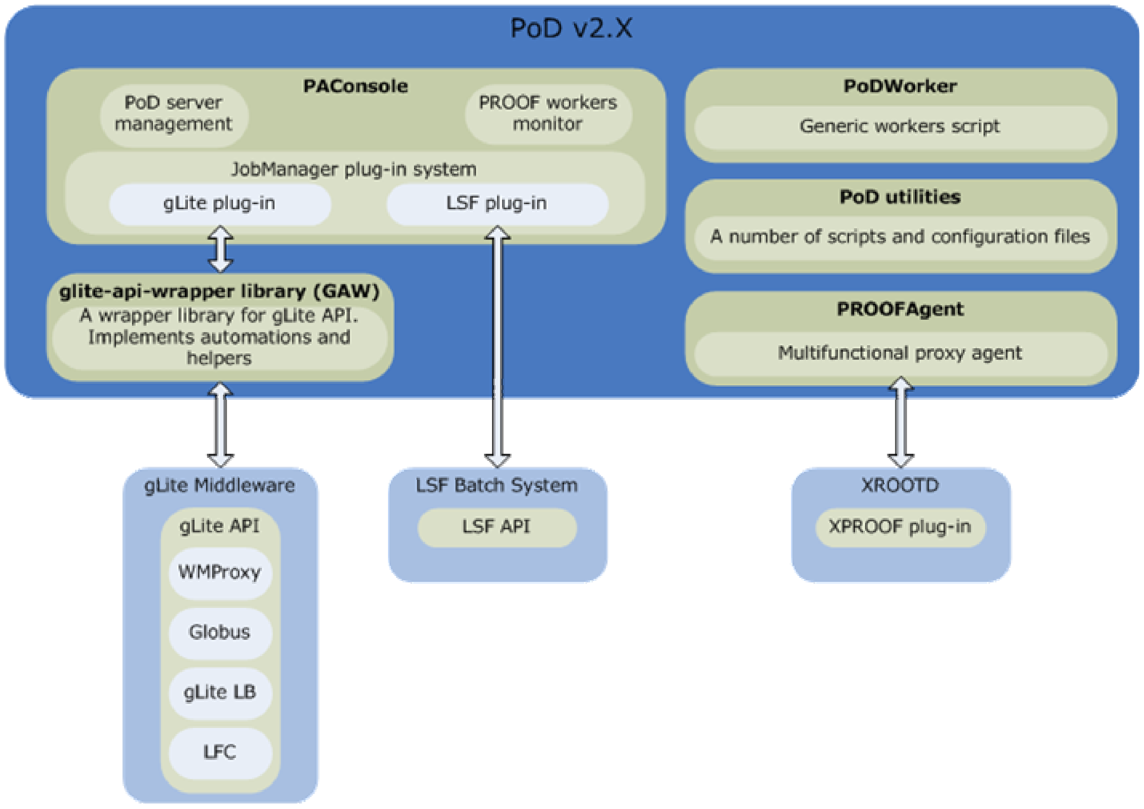


Figure 1. A generic schema of PoD

PoD consists of the following main components (see Figure 1):

- *PROOFAgent* — a lightweight, standalone C++ daemon. Acts as a multifunctional proxy, client/server application and helps to use proof/xrootd [8] on the remote worker nodes possibly behind a firewall. Also PROOFAgent has a number of additional useful features which help to start, to process, and to control a PROOF interactive analysis.
- *PAConsole* — a standalone C++ application, provides a graphics user interface and aims to simplify the usage of PROOFAgent and PoD configuration files. It is a PoD management console.
- *PoDWorker script* — a generic job script, executed on remote machines.
- *PoD utilities* — these are default job scripts for plug-ins, a number of configuration files and helper utilities.

2.1. PROOFAgent

PROOFAgent is a client/server proxy daemon. On remote worker nodes it is used in a client mode. On a user interface machine it runs in a server mode. The main responsibility of PROOFAgent is to provide communication tunnels for xproofd network packets. PROOFAgent actually makes PROOF think that everything is running locally. PROOF server sees its workers as local host workers, and workers themselves think that PROOF server is also running on the local host. PoD always considers that remote worker nodes are blocked for incoming traffic. As you may know, PROOF works on such a way, that PROOF server connects to its workers and not other way around. This is why it is not possible to setup PROOF slaves on Grid workers, for example, or any other machine which doesn't accept incoming traffic.

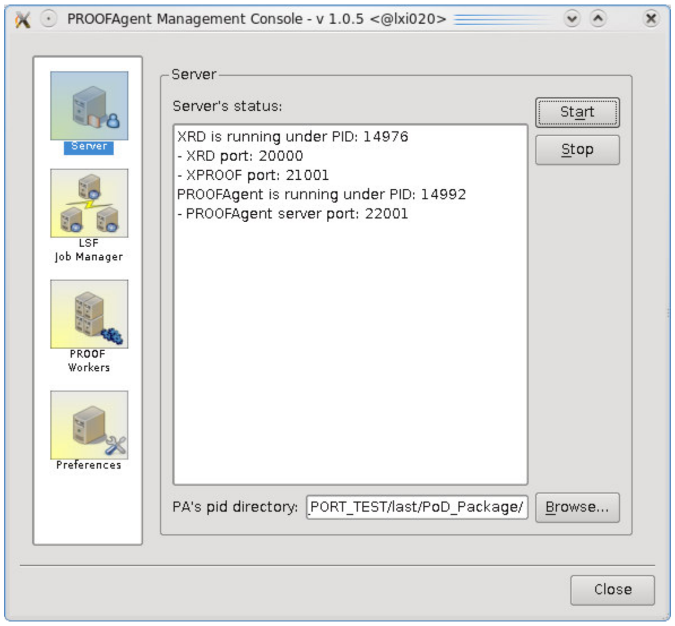
Another issue, where PROOFAgent can help us, is that user IDs under which our jobs running on remote worker nodes and environment settings of the node most of the time are unknown. When you work with Grid workers, for example, then your job could arrive to some Grid site and on the worker nodes you will be mapped to some grid user ID, under which your jobs will be executed. In order to run PROOF worker there we need to know environment settings and a user ID on which PROOF worker will accept connections. This is exactly the case when the second important feature of PROOFAgent helps us. Each PROOFAgent client sends its environment information to a PROOFAgent server. So that PROOFAgent server can make decisions whether to continue to use that particle worker or not.

Before starting each session PROOFAgent checks whether each worker node is "healthy" or not to run PROOF on it. If PROOFAgent detects a bad worker, then it will be immediately removed from PROOF workers list, so that user can safely continue to process analysis. If PoD worker fails during the analysis session, then PROOFAgent will manage this situation and remove a bad worker from the list as well.

In the next version of PROOFAgent it will monitor itself, and if in the client mode it is idle for some defined amount of time, then it will gracefully shut down PoD worker in order to unblock Grid/resource slot.

2.2. PAConsole

PAConsole is PoD's graphics user interface. Using PAConsole one can start/stop PoD server. Also monitor its status. Users can check at any time under which PIDs services are running and which sockets are currently in use (see Figure 2).

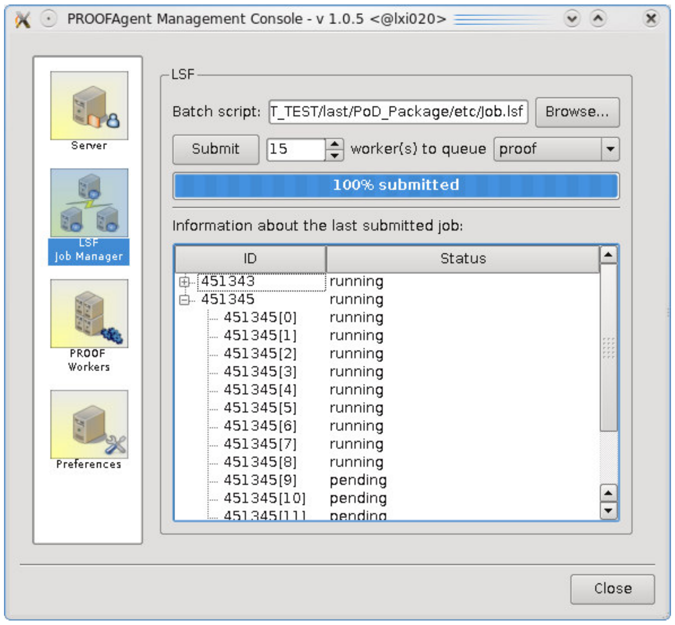


```
XRD is running under PID: 14976
- XRD port: 20000
- XPROOF port: 21001
PROOFAgent is running under PID: 14992
- PROOFAgent server port: 22001
```

PA's pid directory: PORT_TEST/last/PoD_Package/

Figure 2. PAConsole - PoD Server status

In order to support different job submission frontends, PAConsole was designed and implemented as a plug-in based system. Thanks to the job manager plug-ins, users are able to submit PoD workers to different resources, like Grid or local batch systems, which can later function as one uniform PROOF cluster. Currently PoD provides and distributes gLite [6] and LSF [7] plug-ins. Figure 3 shows the LSF plug-in view of the PAConsole.



Batch script: T_TEST/last/PoD_Package/etc/Job.lsf

Submit 15 worker(s) to queue proof

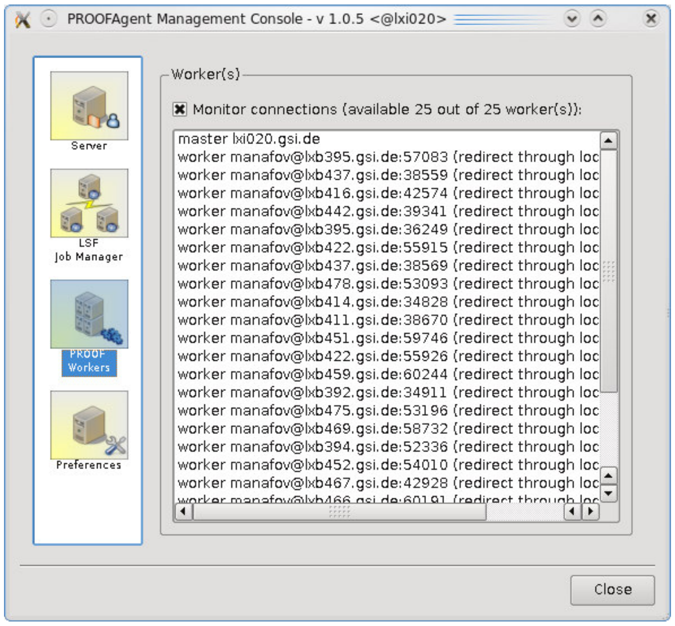
100% submitted

Information about the last submitted job:

ID	Status
451343	running
451345	running
451345[0]	running
451345[1]	running
451345[2]	running
451345[3]	running
451345[4]	running
451345[5]	running
451345[6]	running
451345[7]	running
451345[8]	running
451345[9]	pending
451345[10]	pending
451345[11]	pending

Figure 3. PAConsole - LSF job manager plug-in

PAConsole also helps to examine a list of currently available “good” PROOF workers (see Figure 4) and to tune several PoD configuration settings.



Monitor connections (available 25 out of 25 worker(s)):

```
master lxi020.gsi.de
worker manafov@lxb395.gsi.de:57083 (redirect through loc
worker manafov@lxb437.gsi.de:38559 (redirect through loc
worker manafov@lxb416.gsi.de:42574 (redirect through loc
worker manafov@lxb442.gsi.de:39341 (redirect through loc
worker manafov@lxb395.gsi.de:36249 (redirect through loc
worker manafov@lxb422.gsi.de:55915 (redirect through loc
worker manafov@lxb437.gsi.de:38569 (redirect through loc
worker manafov@lxb478.gsi.de:53093 (redirect through loc
worker manafov@lxb414.gsi.de:34828 (redirect through loc
worker manafov@lxb411.gsi.de:38670 (redirect through loc
worker manafov@lxb451.gsi.de:59746 (redirect through loc
worker manafov@lxb422.gsi.de:55926 (redirect through loc
worker manafov@lxb459.gsi.de:60244 (redirect through loc
worker manafov@lxb392.gsi.de:34911 (redirect through loc
worker manafov@lxb475.gsi.de:53196 (redirect through loc
worker manafov@lxb469.gsi.de:58732 (redirect through loc
worker manafov@lxb394.gsi.de:52336 (redirect through loc
worker manafov@lxb452.gsi.de:54010 (redirect through loc
worker manafov@lxb467.gsi.de:42928 (redirect through loc
worker manafov@lxb466.gsi.de:60191 (redirect through loc
```

Figure 4. PAConsole - PROOF workers monitor

2.3. PoDWorker script

The PoDWorker script is a job script, which is actually started by a job submission frontend system on worker nodes. It is the first what PoD executes on remote worker nodes and the last what “leaves” worker nodes after PoD job is done.

This script is responsible for the first environment recognition and for the start of all PoD worker services. Its responsibility is also a graceful shut down of PoD, and it doesn't matter whether PoD finished correctly, or failed, or received a signal. In all cases PoDWorker script tries to shut down gracefully and collect all possible log files. The log files delivered later to the PoD server.

2.4. PoD utilities and configuration files

PoD distributes also a number of helpers-utilities and configurations files. It is important to mention that default installation of PoD made on such a way that it works out of the box. PoD provides default configuration files for core services, such as xrootd, PROOFAgent and PAConsole. Each job manager plug-in is distributed with its configuration and its job scripts as well, which make it possible to start to use these plug-ins without any modification.

Normally users don't need to modify anything to start using PoD. But since PoD components are very well configurable some users may want to tune settings for a specific configuration.

3. Use Case

The main use case of the PoD is to set up a distributed PROOF cluster on any resource management system. In case of the Grid, the cluster can be distributed over several sites. A schematic picture of this use case can be seen in Figure 5.

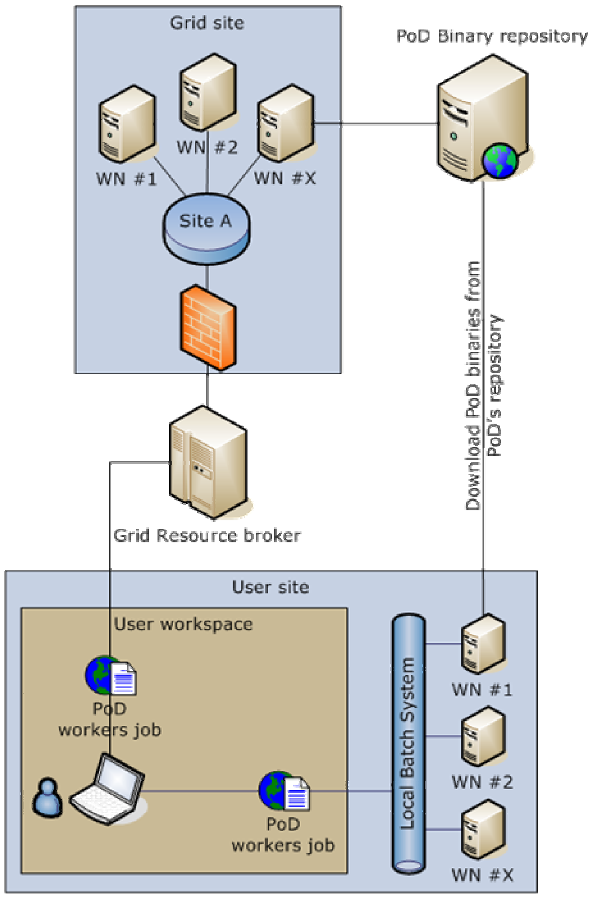


Figure 5. Setting up a distributed PROOF cluster by using the PoD

The first thing, a user has to do, is to start the server side processes on a central machine; in terms of PoD it is a user interface machine. These are the PROOFAgent server daemon and the xrootd redirector. The PROOFAgent server then connects to the xproofd (a PROOF plug-in for xrootd) on PROOF port of the xrootd redirector and waits for PROOFAgent clients to connect.

The next step is to submit PoD jobs to worker nodes. Using job manager plug-ins of PAConsole the user submits a predefined PoD job to a chosen resource. It could be a gLite workload management system or a local LSF farm. It also could be both at the same time if the user has can access both of types of resources. Each job manager plug-in uses the most efficient way of job submission. gLite plug-in submits parametric jobs using WMProxy [9] and LSF uses array jobs.

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 worker and a PROOFAgent in client mode. The PROOFAgent client on the remote cluster, possibly behind a firewall, communicates with the PROOFAgent server, exchanges environment data, and initiates a network tunnel (see Figure 6). 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 PoD 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 he connects to the PROOF master, registers the data, and runs the analysis script. 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 PoD jobs.

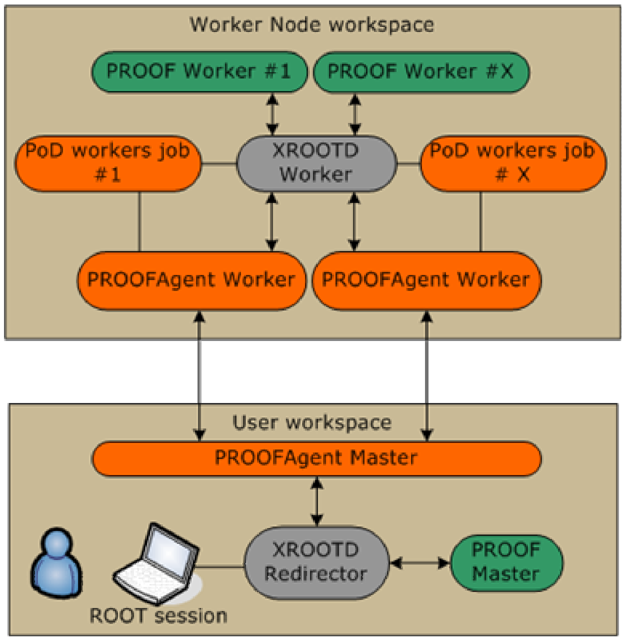


Figure 6. PROOFAgent clients deliver environment information and create network tunnels with the PROOFAgent server

4. Summary

The main goals of the project are to have a tool set which is:

- easy in use,
- transparent,
- extensible,
- very well documented.

The following features of PoD are making it possible to achieve the goals of the project.

- *One click installation*. The process of installation of PoD is very simple and fully automated. In some cases users may be required to let PoD know locations of some external libraries, and if PoD build system has a problem to automatically detect some of them.
- *Easy in use*. PoD provides a simple graphics user interface in order to simplify access to its functionality. Actually, using PoD GUI users need just a couple of mouse clicks to get a private PROOF cluster on the fly setup and ready. PoD also provides a command line interface, which duplicates most of the features of GUI. The command line interface could be used as a substitution for GUI, but using GUI is very much recommended. To use PoD it is not required to have any xrootd/proof installation and configuration knowledge. It is worth to mention that PoD runs in a user workspace and doesn't require root privileges.
- *It works out of the box*. The PoD distributive contains everything users need to just immediately start using PoD right after the installation. All services come with the default configuration files.
- *Dynamic ports on worker nodes and user interfaces*. Nowadays multi-core machines more and more coming to the game. One physical machine normally has now several job slots. While designing PoD it was always kept in mind, that a user interface and PoD workers are multi-core machines, and several concurrent processes of PoD from different users could be running on the same physical machine. PoD has an internal, smart mechanism that decides which port to use for which service. For example, on worker nodes PoD looks for existing xrootd process of the current user and binds all services to use it or start a new one to let other PoD workers on this machine to use it. This and other self-configuration processes happen on the fly at runtime. A user defines only a port range and everything else PoD does automatically on user interfaces and on worker nodes.
- *PoD supports reconnections*. This is essential feature of PoD. Whatever happens with ROOT/PROOF analysis sessions even a crash, users don't need to resubmit PoD jobs. PoD will automatically refresh the environment and reconnect its workers.
- *Supports different job submission frontends*. Since PAConsole supports job manager plug-ins and the interface of the plug-in system is very simple, it is very easy to extend PoD to support different kinds of resources. As it was already mentioned, PoD supports gLite and LSF plug-ins. In the future versions some more plug-ins will be implemented, such as Condor [10] and SGE [11].

The nightly and stable releases of all software packages as described above can be downloaded from the project web page [5], where detailed project documentation, code documentation, and various project metrics can also be found.

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)
[4] ROOT - An Object Oriented Data Analysis Framework
for more information see http://root.cern.ch
[5] PROOF on Demand https://subversion.gsi.de/trac/dgrid
[6] gLite project http://glite.web.cern.ch/glite/
[7] Load Sharing Facility (LSF) http://www.platform.com/
[8] The Scalla Software Suite: xrootd/cmsd http://xrootd.slac.stanford.edu/
[9] gLite WMProxyAPI https://edms.cern.ch/document/674643/1
[10] Condor project http://www.cs.wisc.edu/condor/
[11] Sun Grid Engine (SGE) http://gridengine.sunsource.net/

Acknowledgments
Authors wishing to gratefully acknowledge GSI's system administrator Victor Penso for his useful comments and recommendations on design and implementation of PROOF on Demand.