Dynamic PROOF clusters with PoD: architecture and user experience

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 tool-set, which sets up a PROOF cluster on any resource management system. PoD is a user oriented product with an easy to use GUI and a command-line interface. It is fully automated. No administrative privileges or special knowledge is required to use it. PoD utilizes a plug-in system, to use different job submission front-ends. The current PoD distribution is shipped with LSF, Torque (PBS), Grid Engine, Condor, gLite, and SSH plug-ins. The product is to be extended. We therefore plan to implement a plug-in for AliEn Grid as well. Recently developed algorithms made it possible to efficiently maintain two types of connections: packet-forwarding and native PROOF connections. This helps to properly handle most kinds of workers, with and without firewalls. PoD maintains the PROOF environment automatically and, for example, prevents resource misusage in case when workers idle for too long. As PoD matures as a product and provides more plug-ins, it's used as a standard for setting up dynamic PROOF clusters in many different institutions. The GSI Analysis Facility (GSIAF) is in production since 2007. The static PROOF cluster has been phased out end of 2009. GSIAF is now completely based on PoD. Users create private dynamic PROOF clusters on the general purpose batch farm. This provides an easier resource sharing between interactive local batch and Grid usage. The main user communities are FAIR and ALICE.

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 [5] enabling interactive analysis of large sets of files in parallel on clusters of computers. Normally users get PROOF

provided by administrators as a pre-installed shared cluster. To avoid certain disadvantages of "static" PROOF clusters PROOF on Demand (PoD) [6] has been developed.

2. Overview

PROOF on Demand is a tool-set, which dynamically sets up a PROOF cluster at a user request, on any resource management system (RMS).

PoD is a user oriented product with an easy to use GUI and a command-line interface. It is fully automated, and no administrative privileges, special knowledge or preconfigured nodes are required to use it.

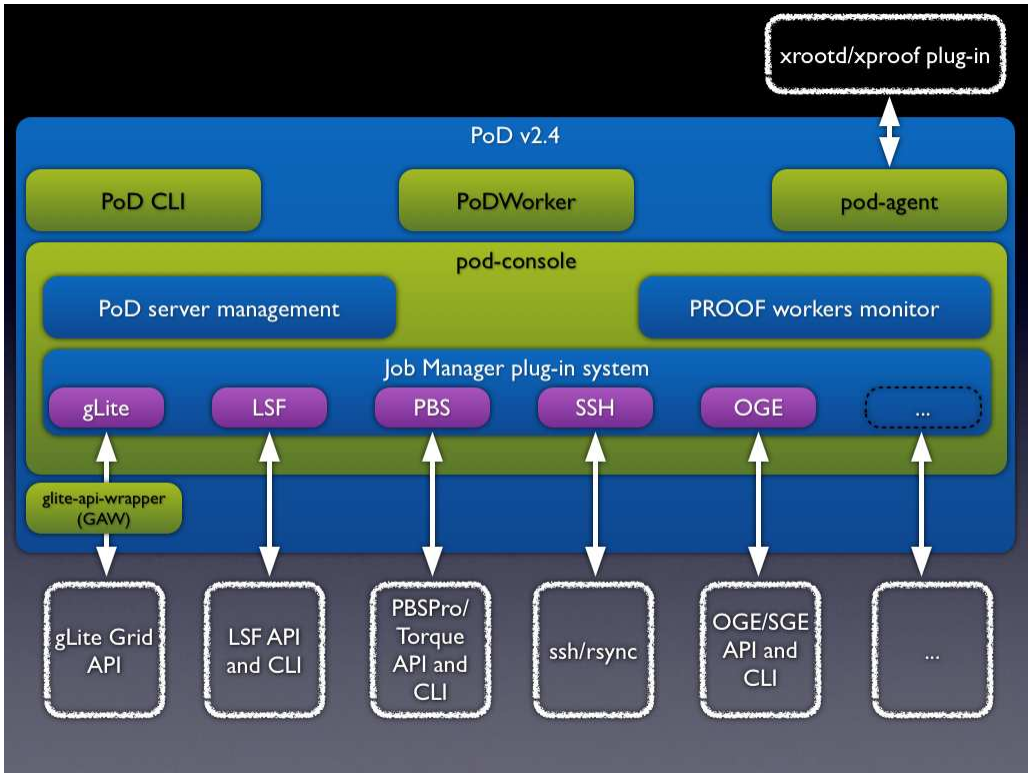

Figure 1. A generic schema of PoD .

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

- *pod-agent* — a lightweight, standalone C++ daemon. Acts as a multifunctional proxy, client/server application and helps to use proof/xproofd [9] on the remote worker nodes possibly behind a firewall. Also *pod-agent* has a number of additional useful features which help to start, to process, and to control a PoD/PROOF interactive analysis.
- *PoD GUI (pod-console)* — a standalone C++ application. It provides a GUI and aims to simplify the usage of PoD.
- *PoD CLI* — a number of utilities, which provide a PoD command line interface.
- *PoDWorker script* — a generic job script, executed on remote machines. This script is responsible for enabling PoD environment and starting PoD processes on worker nodes.
- *PoD utilities* — these are default job scripts for plug-ins, a number of configuration files and helper utilities.

- *Job Manager* — a plug-in based system for both GUI and CLI. It helps to use different job submission front-ends.

3. Main features of PoD

- **Easy to use**. The process of installation is very simple and fully automated. PoD works out of the box. Its distribution contains preconfigured modules and everything users need to just immediately start to work with it right after the installation.

- **GUI & Command-line.** PoD provides a simple and intuitive graphics user interface in order to simplify access to its functionality. For user's convenience there is also a command line interface, it helps to manage a PoD cluster remotely or use it in a batch mode.

- **Native PROOF connections.** Whenever possible, PoD setups direct PROOF connections between nodes. It results in a full functional PROOF cluster. Users get native speed and the whole range of PROOF features. To use native connections an incoming traffic must be allowed on PoD workers for a defined port. Otherwise PoD uses packet-forwarding algorithms.

- **Packet-forwarding.** When worker nodes are behind a firewall then PoD uses its packet-forwarding algorithms to maintain the PROOF traffic. The algorithms are very efficient, there will be no speed penalty, but some PROOF functions are limited.

- **Multiuser/-core environment.** PoD implements automatic port mapping algorithms to properly handle cases when several users start PoD instances (servers/ workers) on the same machine. PoD also automatically manages situations when multiple PoD workers are started on the same node. Private PoD instances can't disturb each other.

- **Different job managers.** PoD supports different job managers via a plug-in system. It is a very easy to extend system. PoD is currently shipped with the following plug-ins:

    - SSH,
    - LSF (Load Sharing Facility) [8],
    - PBS Pro/OpenPBS/Torque (Portable Batch System) [13],
    - Grid Engine (Oracle/Sun Grid Engine) [12],
    - Condor [11],
    - gLite [7] [10].

4. Use case

The main use case of PoD is to set up a distributed PROOF cluster on an RMS and/or the Grid. This is described in the following easy steps:

1. First of all users need to start PoD server, which later will became a PROOF muster. PoD server can be started using PoD CLI: ***pod-server start***
   or using GUI, see Figure 2.

Currently PoD server and PoD user interface always run on the same machine. Future PoD versions will allow user to start remote PoD servers and use a laptop, for example, as a user interface.

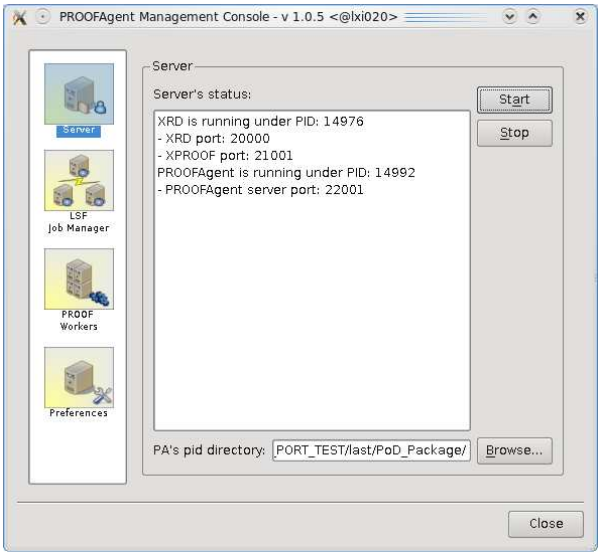

Figure 2. pod-console: Server page

2. The next step is to submit remote PoD workers using PoD's job manager. These PoD workers will automatically setup PROOF workers on remote hosts. Starting with version 2.0.7 the PoD project provides a plug-in based system, in order to use different job submission front-ends. PoD is currently shipped with gLite, LSF, PBS (PBSPro/OpenPBS/Torque), Grid Engine (OGE/SGE), Condor, and SSH plug-ins. PoD makes it possible just within a few seconds to get a private PROOF cluster on any RMS. If there is no RMS, then SSH plug-in can be used, which dynamically turns a bunch of machines to PROOF workers. The SSH plug-in is also a perfect solution for a Cloud based PROOF clusters. It also possible to use a combination of plug-ins to get PROOF workers on Grid worker nodes and local batch machines in the same time. Let's take, for example an LSF plug-in and setup our PROOF cluster with 200 workers on LSF farm's queue "proof".
   PoD CLI: **pod-submit –r lsf –q proof –n 200**
   or using GUI, see Figure 3.

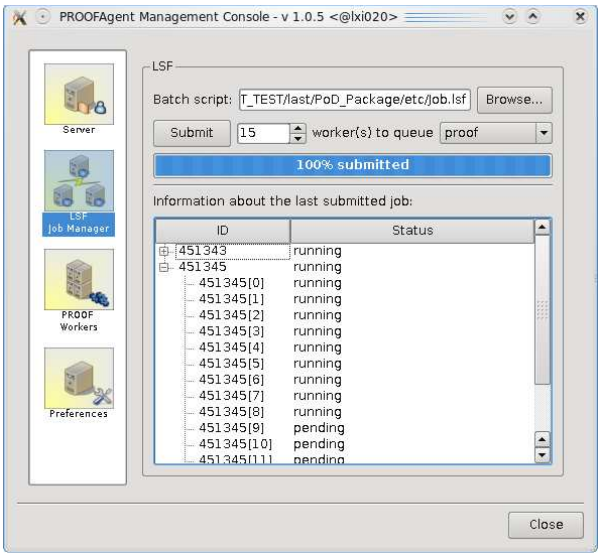

Figure 3. pod-console: LSF job manager

3. As soon as a single job reaches remote worker node (WN), it tries to connect to PoD server to transfer information about itself and environment of WN. When negotiations are done and PoD server accepts WNs, it became a normal PROOF Worker for the user. It is not required to wait until all requested workers will be connected. Users could start analysis after a reasonable number of workers are on-line, even after the first connected worker one could start the analysis. When other workers arrive, the ROOT (PROOF) session must be restarted in order to reconnect to the newly arrived workers.
   A number of available PROOF workers users can check using

   PoD CLI: ***pod-info –n***

   or using GUI, see Figure 4.

Now when your remote PROOF workers (PoD workers) are on-line, you can process you ROOT/PROOF analysis normally, if it would be a usual PROOF session.

PoD supports reconnection. That means if your analysis has a bug or a root session crashed you don't need to resubmit PoD jobs. You just need to close current root session, open it again. PoD will manage reconnection with its worker nodes automatically. Worker nodes will be on-line until the pod-agent service is on-line or until s Grid and/or batch queue time is over.

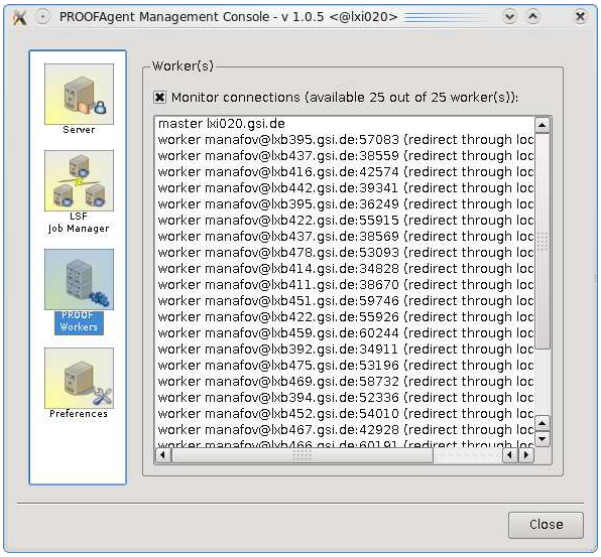

Figure 4. pod-console: PROOF workers

5. Summary

As PoD matures as a product, it is used more and more as a standard for setting up dynamic PROOF clusters in many different institutions in HEP community. Additionally there are already several Cloud based installations, which use PoD as a PROOF cluster solution. With PoD there is no need to maintain a dedicated PROOF analysis facility. PoD users create themselves private dynamic PROOF clusters on general purpose batch farms, Grid or Cloud systems.

Upcoming versions of PoD are going to support an out-of-server user interface. Users will be able to select a remote computer acting as PoD server (PROOF master). In this case PoD UI will be just a lightweight control center and could run on different OS types. Also an AliEn plug-in is going to be developed in collaboration with the ALICE Offline team. This cooperation will help PoD to provide a fast interactive PROOF experience on the AliEn Grid.

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] The Parallel ROOT Facility, PROOF http://root.cern.ch/drupal/content/proof
[5] ROOT - An Object Oriented Data Analysis Framework
for more information see http://root.cern.ch
[6] PROOF on Demand http://pod.gsi.de
[7] gLite project http://glite.web.cern.ch/glite/
[8] Load Sharing Facility (LSF) http://www.platform.com/
[9] The Scalla Software Suite: xrootd/cmsd http://xrootd.slac.stanford.edu/
[10] gLite WMProxyAPI https://edms.cern.ch/document/674643/1
[11] Condor project http://www.cs.wisc.edu/condor/
[12] Grid Engine http://gridengine.sunsource.net/
[13] Portable Batch System http://en.wikipedia.org/wiki/Portable_Batch_System