Dynamic parallel ROOT facility clusters on the Alice Environment

C Luzzi 1, L Betev 2, F Carminati 2, C Grigoras 2, A Manafov 3 and P Saiz 2

1 University of Ferrara, Via Giuseppe Saragat, 1, 44020 FE, Italy
2 CERN, 1211 Geneva 23, Switzerland
3 GSI Helmholtzzentrum für Schwerionenforschung GmbH, Planckstr. 1, 64291 Darmstadt, Germany
E-mail: cluzzi@cern.ch

Abstract. The ALICE collaboration has developed a production environment (AliEn) that implements the full set of the Grid tools enabling the full offline computational work-flow of the experiment, simulation, reconstruction and data analysis, in a distributed and heterogeneous computing environment.

In addition to the analysis on the Grid, ALICE uses a set of local interactive analysis facilities installed with the Parallel ROOT Facility (PROOF). PROOF enables physicists to analyze medium-sized (order of 200-300 TB) data sets on a short time scale. The default installation of PROOF is on a static dedicated cluster, typically 200-300 cores. This well-proven approach, has its limitations, more specifically for analysis of larger datasets or when the installation of a dedicated cluster is not possible. Using a new framework called PoD (Proof on Demand), PROOF can be used directly on Grid-enabled clusters, by dynamically assigning interactive nodes on user request.

The integration of Proof on Demand in the AliEn framework provides private dynamic PROOF clusters as a Grid service. This functionality is transparent to the user who will submit interactive jobs to the AliEn system.

1. Introduction

In order to describe the integration of PoD [1] framework in AliEn [2], we explain briefly all the components.

PROOF [3] is part of the ROOT [4] analysis framework and its goal is to allow the parallel analysis of data. PROOF consists of a three tier architecture: the ROOT client session, the PROOF master server and the PROOF slave servers. The user connects from his ROOT session to a master server on a remote cluster and the master server in turn creates slave servers on all the nodes in the cluster. Queries are processed in parallel by all the slave servers. The slave asks the master server for work packets by using a pull protocol which allows the master to distribute customized packets for each slave server.

AliEn is the ALICE [5] grid environment that allows to manage and use data in a distributed way. In a static scenario, the integration of PROOF in the AliEn system is performed by using a PROOF cluster for each AliEn site. To become a PROOF master or a slave server a farm has to run a proofd process. These daemons are always running on dedicated machines on each site.

This last feature constrains to the use of a limited number of machines. To increase the number...
of possible workers and offer a more flexible solution than the static cluster, a new software named PoD, has been developed at GSI [6].

PoD consists of two components: the PROOFAgent and the PAConsole. The PROOFAgent provides the communication layer between the PROOF master on the local machine and the PROOF workers on the remote nodes. The PAConsole provides a user-friendly GUI, which is used to setup, manage, and shutdown the dynamic PROOF cluster. PoD provides also a PoDWorker script that needs to be executed on remote machines. It is started by a job submission frontend system on worker nodes. In our case, we use the PoDWorker script to start PoDWorkers within the AliEn system.

The integration of PoD in AliEn will let us avoid the drawbacks of a static system. In a dynamic environment, a PoD service can start proofd processes dynamically at the user request within AliEn. In this way the system can react to an increasing number of requests for PROOF sessions by starting a higher number of proofd processes.

2. PROOF on the Grid

In order to request a PoD Server the user has to submit a job on AliEn.

The user’s jdl needs to be configured using the StartPoDServer.sh as executable file, ROOT and PoD as necessary packages and PoDServer as partition where the job will be executed. A minimum and a maximum number of workers and a maximum of waiting time to get the maximum number of workers can also be specified on the jdl.

An example of jdl is:

```plaintext
Executable: "StartPoDServer.sh";
Partition = "PoDServer";
Packages: {"ROOT","PoD"};
Arguments = 5; Number of Workers
```

The detection of the request for a PoD server is a trigger for AliEn to start the server on a dedicated AliEn Computing Element (CE). In addition to the procedure of starting the server, the StartPoDServer script submits a new jdl that will start the procedure to create a PoD worker. This jdl will be executed by a normal AliEn CE. While it is running the server starts to publish information on the MonALISA [7] Information System.

On the worker node side, the procedure to create a PoD worker starts to check MonALISA in order to find an available PoD server. In case of availability, a PoD worker will be created and connected to the server. If there are no available servers, the procedure will check again for a predefined number of times and as soon as the timeout has expired the procedure will exit and the system will execute other AliEn jobs. The Figure 1 shows the whole architecture of the integration of PoD in the AliEn system.
2.1. PoD Server

When a PoD user request is detected, the dedicated AliEn CE will install the PoD predefined package and will execute the PoDStartServer.sh script. This script contains only the command to start the PoD server because all the environment variables are set up within AliEn during the PoD installation process. If the starting server procedure is successful then the script submits a new jdl to AliEn in order to start the PoD worker creation procedure. As soon as the server is running, it starts to advertise itself to MonALISA by publishing the hostname and the port number on which it is listening and the used PoD version (see Fig. 2). The publication of the information from the PoD server to MonALISA has been done by using the ApMon API [8].

The PoD server stops to send information to MonALISA either when it has acquired the predefined set of workers or after a predefined amount of time during which it is idle. After this timeout the server will be shut down automatically.
2.2. PoD Workers

The PoDWorker jdl, submitted by the PoDStartServer.sh, requires the installation of PoD and ROOT packages and will be executed by a normal AliEn CE. This jdl defines as executable the CreatePoDWorker procedure. As soon as the job is executed, the procedure starts to check MonALISA to know if there is a PoD server available. Once this procedure detects the requests from the PoD server, it will stop to check MonALISA and will start the PoD worker.

In particular, the CreatePoDWorker procedure creates a configuration PoD server file by processing the information about the PoD server (hostname, port number and PoD server version) taken from MonALISA and by using it to create the PoD worker node (WN) package on the AliEn WN. As last thing, the procedure executes the PoD WN script that will start the xproof daemon and will create the connection between PoD worker and PoD server (see Fig. 3). If the CreatePoDWorker procedure does not detect an available PoD server then it will check again MonALISA for a predefined number of times. As soon as the timeout has expired the procedure exits and AliEn WN starts to execute other AliEn jobs.

3. Conclusion and Outlook

The integration of PoD in the AliEn system is easy to use, maintain and install.

The use of PoD from the client point of view is completely transparent. After a jdl submission the user will be granted all the advantages of using PROOF for the data analysis.

From the site administrator point of view, the PoD package will be predefined in the AliEn system and installed automatically after the user request. Both the procedure to start the server and the procedure to create PoD workers are managed by means of the jobs submission in AliEn. In this way no additional installation is required by the site administrator who might only have the necessity to set up the timeout variables in the LDAP database.

The AliEn PoD worker acquires a list of free server nodes from MonALISA. Currently, the first free PoDserver that is on the list is the one that will execute the next PoD job. In the future a new optimized procedure to choose the available PoD server will be integrated. For example the nearest PoD server can be chosen in order to reduce the network traffic and the system overheads. In order to select this PoD server, each MonALISA VOBox service could start a web service with a similar layout as MonALISA (PoDMonALISA). The PoD server will send its information to the closest PoDMonALISA service in the environment and the CreatePoDWorker procedure will ask for the information to the closest PoDMonALISA service too.

Figure 3. Creation and connection of the PoD worker with the PoD server
Acknowledgements
We would like to thank the Prof. Eleonora Luppi for her support and advices throughout this work.

References
[1] Malzacher P, Manafov A. 2009, Computing in High Energy Physics and Nuclear Physics (Prague: Czech Republic) PROOF on Demand http://pod.gsi.de
[2] Buncic P, Peters A J, Saiz P and Grosse-Oetringhaus J F 2004 Computing in High Energy Physics and Nuclear Physics (Interlaken: Switzerland) The architecture of the AliEn system http://alien2.cern.ch
[3] Ballintijn M, Roland G, Brun R, Rademakers F 2003 Computing in High Energy Physics and Nuclear Physics (La Jolla: California) The PROOF Distributed Parallel Analysis Framework based on ROOT
[4] An Object-Oriented Data Analysis Framework http://root.cern.ch
[5] The ALICE collaboration http://aliceinfo.cern.ch
[6] GSI Helmholtzzentrum für Schwerionenforschung GmbH http://www.gsi.de
[7] Newman H B, Legrand I C, Galvez P, Voïcu R and Cirstoiu C 2003, Computing in High Energy Physics and Nuclear Physics (La Jolla: California) MonALISA : A Distributed Monitoring Service Architecture. http://alimonitor.cern.ch
[8] ApMon User Guide: http://monalisa.cern.ch/monalisa_Documentation_ApMon_User_Guide.htm