Interactive Analysis using PROOF in a GRID Infrastructure

2011 J. Phys.: Conf. Ser. 331 072061

(http://iopscience.iop.org/1742-6596/331/7/072061)

Interactive Analysis using PROOF in a GRID Infrastructure

Ana Yaiza Rodríguez Marrero[1], Isidro González Caballero[2], Alberto Cuesta Noriega[2], Francisco Matorras Weinig[1]

[1] Instituto de Física de Cantabria (UC-CSIC), [2] Universidad de Oviedo

E-mail: arodrig@ifca.unican.es

Abstract. Current high energy physics experiments aim to explore new territories where new physics is expected. In order to achieve that, a huge amount of data has to be collected and analyzed. The accomplishment of these scientific projects require computing resources beyond the capabilities of a single user or group, thus the data is treated under the grid infrastructure. Despite the reduction applied to the data, the sample used in the last step of the analysis is still large. At this phase, interactivity contributes to a faster optimization of the final cuts in order to improve the results. The Parallel ROOT Facility (PROOF) is intended to speed up even further this procedure providing the user analysis results within a shorter time by simultaneously using more cores. Taking profit of the computing resources and facilities available at Instituto de Física de Cantabria (IFCA), shared between two major projects LHC-CMS Tier-2 and GRID-CSIC, we have developed a setup that integrates PROOF with SGE as local resource management system and GPFS as file system, both common to the grid infrastructure. The setup was also integrated in a similar infrastructure for the LHC-CMS Tier-3 at Universidad de Oviedo that uses Torque (PBS) as local job manager and Hadoop as file system. In addition, to ease the transition from a sequential analysis code to PROOF, an analysis framework based on the TSelector class is provided. Integrating PROOF in a cluster provides users the potential usage of thousands of cores (1,680 in the IFCA case). Performance measurements have been done showing a speed improvement closely correlated with the number of cores used.

1. Introduction

The Parallel ROOT [1] Facility (PROOF [2]) allows researchers to analyze and understand much larger data sets on a shorter time scale by enabling interactive executions in parallel on clusters of computers or many core machines. The main goal concerning this work is to construct and provide a user friendly environment to fully profit from the PROOF system.

PROOF is integrated transparently with the used local batch system. The framework developed can be used with the CMS analysis software. The final configuration offers:

- Centralized integration between PROOF and the batch system.
- Transparent access to the PROOF cluster for the users.
- An analysis framework to ease the transition from the user analysis code to the required structure by the PROOF usage.

2. PROOF in the batch system

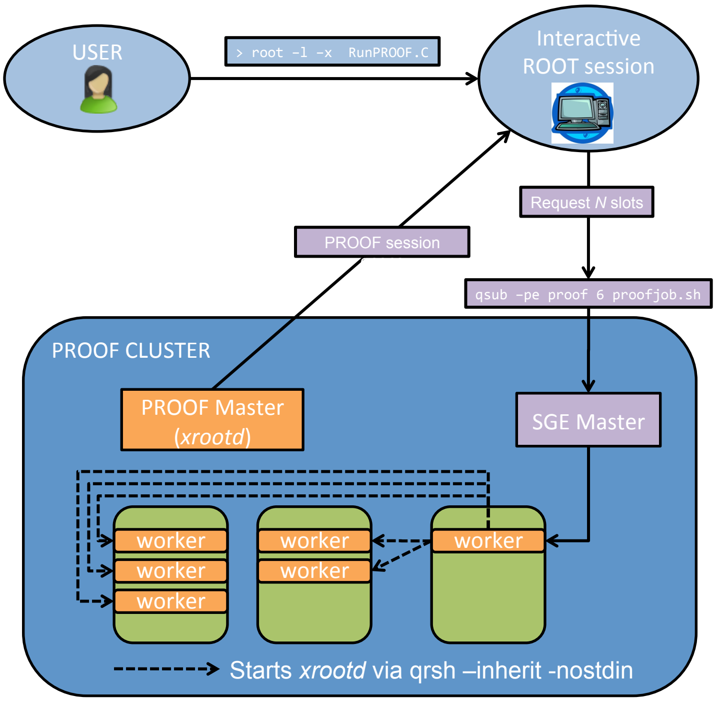
The most common way to set up a PROOF cluster is to configure a set of machines to run the PROOF servers permanently. However, a dedicated PROOF cluster is incapable of adapting to the dynamic demand of the resources made by users.

By taking profit of established infrastructures as, for example, the ones at Instituto de Física de Cantabria (IFCA) and at Universidad de Oviedo (UO), our system provides a fast response for the users and at the same time makes an efficient use of resources by only requesting the needed nodes. Those institutions give support to many researchers that ultimately execute their jobs at the corresponding batch queues. Local users at these centres profit from an interactive queue through which individual PROOF clusters can be dynamically enabled.

When the users start a PROOF session the following process occurs:

- A *startproof* script initializes and exports all the configuration variables that are needed at the master and workers servers, and submits a single SGE [1] or Torque (PBS [2]) job *proofjob* requiring the number of desired slots that act as workers.
- *Proofjob* configures and starts the *xrootd* daemon on the primary worker and spawns through an interactive session the same process at the secondary workers. Once the process is successfully completed, a signal is sent to the client and captured by the user analysis ROOT macro.
- The client connects to the master initializing a PROOF session on the dynamic cluster just built where the analysis can be executed using the PROOF facility.
- The master is independent to the batch system. It runs its own *xrootd* daemon permanently.

Figure 1 shows a schematic view of the process just described.



```
> root -l -x  RunPROOF.C
```

```
qsub -pe proof 6 proofjob.sh
```

Figure 1. Schematic view of the different steps carried out when a user starts a PROOF session.

[1] See http://www.gridengine.sunsource.net

[2] See http://www.clusterresources.com

There are some other aspects of the dynamic PROOF cluster that are dependent on the PROOF Cluster usage policy:

- Since the PROOF job is submitted to an interactive queue, the waiting time to get the slots is of the order of a few seconds.
- The number of slots assigned by the job manager depends on the current load of the batch system. The user requirements are used as an upper limit but fewer resources may be provided, a minimum of two slots is guaranteed.
- In case the cluster is fully loaded some jobs from other users may be suspended for a while in order to fulfill the previous features.
- Typically, the dynamic PROOF cluster jobs will run in the batch system on a short queue. We have set the limit of a PROOF cluster to one hour from its initialization. Several PROOF sessions may be run throughout that time period.

3. A CMS Analysis Framework using PROOF

Based on the needs of our users to analyse plain `TTree` ROOT files with a selection of events and information extracted from the official CMS datasets we have built a user friendly and flexible PROOF analysis framework to simplify the migration of the large amount of sequential user code to a PROOF based environment. This kind of HEP analysis is specially suited for PROOF. The framework is slightly more general providing means to run the exact same code in sequential mode or through PROOF.

We have gone one step forward avoiding the manual recreation of the branches every time the exact content of the data files changes by automatically producing and loading that information. The user can, therefore, concentrate the efforts on the coding of the actual selection and reconstruction algorithms for the analysis.

A set of extra tools specialized for its usage in PROOF are also provided: counters, input parameters, and collections. The handling of typical analysis objects (histograms, counters, collections) is optimized through dedicated methods in order to take care of repetitive call sequences. Figure 2 shows a scheme of this structure at the user and developers level.

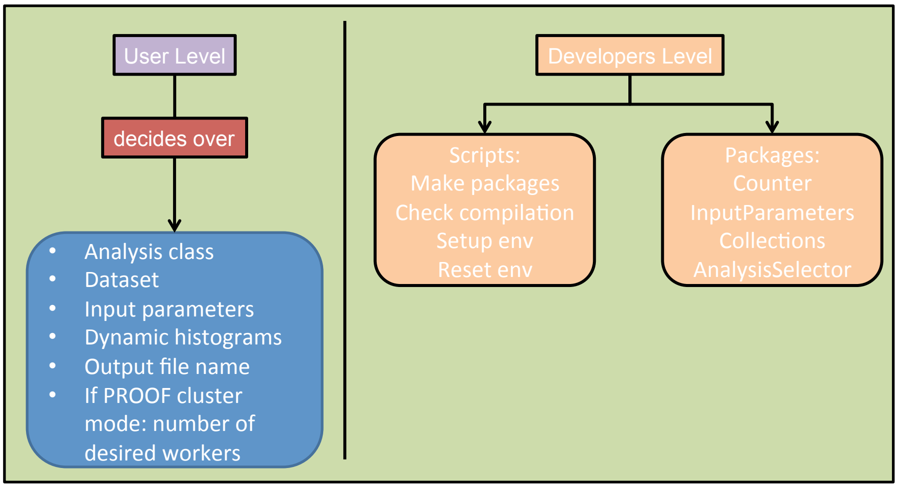
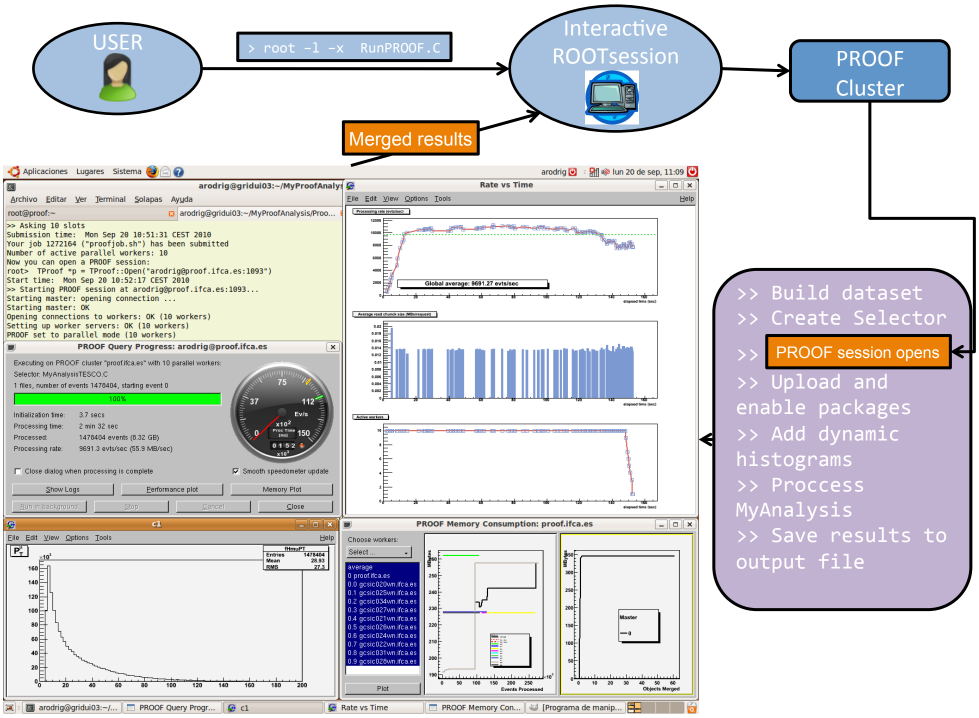


Figure 2. Basic scheme showing the decoupling of the funcionality between users and developers.

The configuration of the PROOF session is handled through a single entry point macro. The objects created in the PROOF session are automatically stored in a ROOT file that can be later explored. The users may also select some of the histograms to be interactively plotted as they are filled during a PROOF session. This may be very useful in early spotting mistakes in the analysis code. Figure 3 shows the flow from the ROOT session start to the end of the analysis execution.


USER
> root -l -x RunPROOF.C
Interactive ROOTsession
PROOF Cluster
Merged results
>> Build dataset
>> Create Selector
>> PROOF session opens
>> Upload and enable packages
>> Add dynamic histograms
>> Proccess MyAnalysis
>> Save results to output file
Aplicaciones Lugares Sistema
arodrig@gridui03:~/MyProofAnalys
Archivo Editar Ver Terminal Solapas Ayuda
root@proof:~
arodrig@gridui03:~/MyProofAnalysis/Proo...
Rate vs Time
Global average: 9691.27 evts/sec
PROOF Query Progress: arodrig@proof.ifca.es
Ev/s
Proc Time [ms]
c1
fHmuPT
PROOF Memory Consumption: proof.ifca.es
Choose workers:
Plot

```
>> Asking 10 slots
Submission time:  Mon Sep 20 10:51:31 CEST 2010
Your job 1272164 ("proofjob.sh") has been submitted
Number of active parallel workers: 10
Now you can open a PROOF session:
root>  TProof *p = TProof::Open("arodrig@proof.ifca.es:1093")
Start time:  Mon Sep 20 10:52:17 CEST 2010
>> Starting PROOF session at arodrig@proof.ifca.es:1093...
Starting master: opening connection ...
Starting master: OK
Opening connections to workers: OK (10 workers)
Setting up worker servers: OK (10 workers)
PROOF set to parallel mode (10 workers)
```

Executing on PROOF cluster "proof.ifca.es" with 10 parallel workers:
Selector: MyAnalysisTESCO.C
1 files, number of events 1478404, starting event 0
100%
Initialization time: 3.7 secs
Processing time: 2 min 32 sec
Processed: 1478404 events (8.32 GB)
Processing rate: 9691.3 evts/sec (55.9 MB/sec)
□ Close dialog when processing is complete
☑ Smooth speedometer update

Figure 3. Simplified flow from the ROOT session start to the end of the analysis execution.

4. Testbeds: IFCA and Universidad de Oviedo

IFCA [3] and Universidad de Oviedo [4] provide computing resources for the Spanish participation at the LHC-CMS experiment, they do this task as federated Tier-2 and Tier-3 respectively. Our implementation of a dynamic PROOF cluster makes use of these computing resources.

Figure 4 describes the submission mechanism to the batch system for PROOF jobs. Once the jobs are submitted to the batch system they are executed through the interactive queue in the corresponding worker nodes. Those jobs access data and write the output via the shared file system. All these resources are also used by the jobs submitted via grid. In this later case the job execution in the batch system is managed by the computing elements and the access to the file sytem by the storage element.

IFCA uses the SGE batch system, GPFS [5] as file system and StoRM [6] as storage element. IFCA counts with 1680 cores at its grid cluster, and has 600 TB RAW for CMS storage. Universidad de Oviedo uses the Torque (PBS) batch system, and Hadoop [7] as file system and storage element. Universidad de Oviedo counts with 100 slots, and 100 TB RAW for CMS storage.

[3] See http://grid.ifca.es/Tier2
[4] See http://www.hep.uniovi.es
[5] See http://www-03.ibm.com/systems/software/gpfs
[6] See http://storm.forge.cnaf.infn.it
[7] See http://hadoop.apache.org

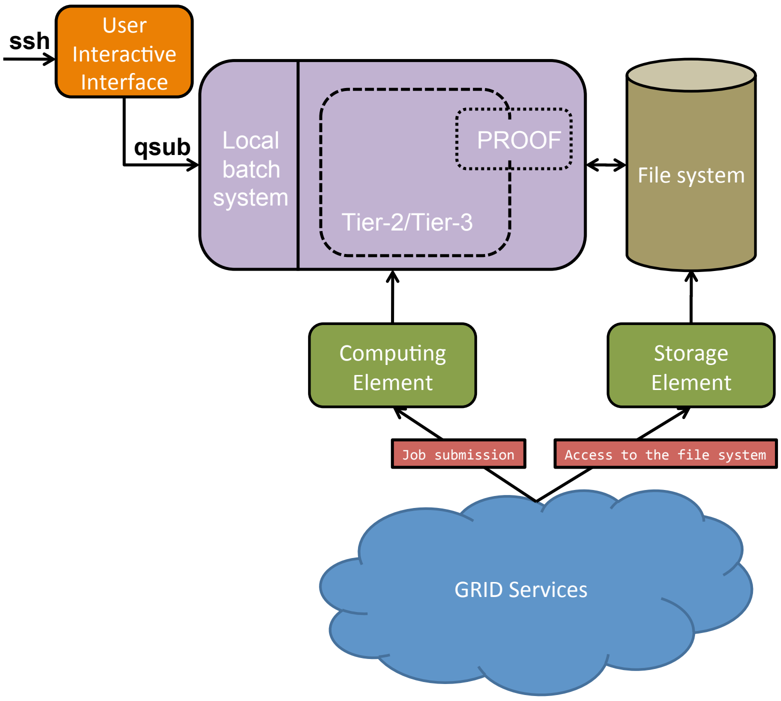


Figure 4. PROOF submission jobs to the batch system.

5. Performance

In order to study the performance of the PROOF cluster, several tests have been carried on. An I/O bounded analysis that applies cuts to the events and fills a few histograms was used to study how the application scales with the number of workers. The possible dependence of the performance with the size of the analyzed data was also taken into account. Two sets of CMS data were used, with a size of 11 GB and 44 GB, these sets consisted of 1GB data files.The analysis run over each of them 10 times for each assignation of worker nodes. The mean value for each of the 10 repetitions and the general behaviour obtained at IFCA, with SGE as batch system and GPFS as file system, is shown in Figure 5.

Our measurements show that the PROOF cluster performs almost linearly up to 20 workers. CPU-bounded analysis are expected to show a better scalability. More detailed studies are planed in order to understand the speed-up behavior.

6. Conclusions

Dynamic PROOF clusters can be currently created at IFCA through the SGE batch system, and at Universidad de Oviedo through PBS.

CMS local users at both institutions have been using this new facility since its integration very recently. The analysis framework favored the migration from sequential analysis to parallel analysis executions.

The full setup (configuration files, scripts, etc) to integrate PROOF into a SGE or PBS batch system is available upon request to the authors.

References

[1] ROOT: A C++ framework for petabyte data storage, statistical analysis and visualization. Computer Physics Communications; Anniversary Issue; Volume 180, Issue 12, December 2009, Pages 2499-2512.

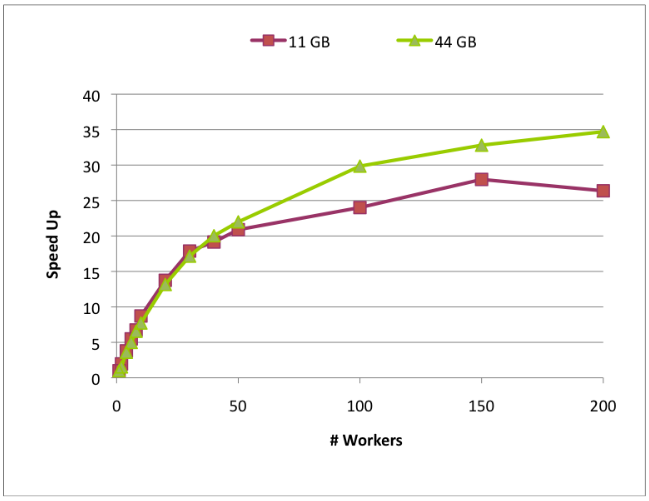


Figure 5. The processing time speeds up almost linearly with the number of workers up to 20 workers. The processing time decreases from 40 minutes (reading the 44 GB dataset) to 5 minutes for 10 workers and to 1 minutes for 200.

[2] The PROOF Distributed Parallel Analysis Framework based on ROOT. 2003 Conference for Computing in High-Energy and Nuclear Physics, La Jolla, CA, USA, 24 - 28 Mar 2003.