PROCEEDINGS OF SCIENCE

Data Analysis with PROOF

Gerardo Ganis*
CERN
E-mail: gerardo.ganis@cern.ch

Jan Iwaszkiewicz
CERN and *Institute of Informatics, University of Warsaw*
E-mail: jan.iwaszkiewicz@cern.ch

Fons Rademakers
CERN
E-mail: fons.rademakers@cern.ch

The Parallel ROOT Facility (PROOF) is a software system enabling ROOT-based analysis and processing in parallel on distributed resources. PROOF optimizes the execution time by implementing data parallelism at event-level. Furthermore it provides real-time feedback. The scalable Master-Worker architecture allows efficient analysis on computing clusters but it is also a natural way to exploit multicore desktops and laptops. We discuss the PROOF approach in the context of LHC data analysis, in particular for medium-size tasks requiring short response time, in comparison to the conventional batch approach. We also discuss the main issues which emerged in the current major test installations from ALICE and ATLAS: performance, dataset management and scheduling.

XII Advanced Computing and Analysis Techniques in Physics Research
November 3-7 2008
Erice, Italy

*Speaker.

PoS(ACAT08)007

1. Introduction

The Large Hadron Collider (LHC) is designed to collide protons at a center-of-mass energy of 14 TeV and an instantaneous luminosity of 10^{34} cm^{-2} s^{-1} corresponding to about $40 \cdot 10^6$ collisions per second [1]. Under reasonable assumptions [1] these numbers translate into an amount of raw data of about 10 PB/year collected in total by the four major experiments installed on the ring. Since the beginning of the LHC Computing Grid project (LCG), it was clear that this unprecedented amount of data required a special handling and distribution model [1]. In particular, the efficient analysis of these amounts of data, by a large number of users, in a complex analysis environment is not a trivial task and needs to efficiently exploit the available resources.

In this paper we review the end-user analysis scenarios and discuss how some of the issues are addressed by the Parallel ROOT Facility (PROOF [2, 3, 4]).

1.1 The LHC data analysis problem

To set the scale of the data to be analyzed, the additional transformations applied to the raw data have to be taken into account. The format dedicated to physics analysis, Analysis Data Objects or AOD, is the result of two reduction steps: the reconstruction, which extracts the physics and data-quality control information into the Event Summary Data format (ESD), and the skimming of all information not directly related to analysis. The total size of the AOD samples is about 30÷200 TB/year. Further reductions, at physics group or individual level, may take place to produce the Derived Physics Data (DPD), with a total size of the order of 10 TB/year. In addition to the real data, there is between 20 and 100% of simulated raw data which undergo the same treatment, potentially even doubling the sizes.

These large amounts of data are handled with a *hierarchical multi-tier distribution model*, structured in four tiers [1]. The exact functionality provided by each tier depends, of course, on the experiment. A typical partitioning of tasks is the following:

- *Tier-0* Unique and located at CERN; it stores one copy of raw data, and typically it runs the first reconstruction and the data quality checks. Some prompt analysis on selected data samples could also be run at this level.
- *Tier-1* About 10 centers distributed worldwide; they typically store a second copy of raw data, run the additional reconstruction cycles, and produce the ESD and AOD formats.
- *Tier-2* About five centers per Tier-1, corresponding to national geographical areas; typically they act as ESD/AOD repository and they are the place where the Monte Carlo simulations are run; depending on the experiment, end-user analysis jobs may be run at Tier-2 centers.
- *Tier-3* The least defined of the four; Tier-3 should correspond to the computing centers at department level where the bulk on end-user analysis is done; their size is order of 30 nodes and they should be able to store data in AOD and DPD format.

[1] LHC will also run in heavy ion mode to collide lead ions at 2.76 TeV/n and 10^{27} cm^{-2} s^{-1}.

1.1.1 End-user analysis scenarios and the proposed solution

End-user analysis will therefore take place at Tier-3 or Tier-2 level on analysis facilities of about 100 nodes and accessing up to ≈100 TB of data. The typical end-user activities can be classified in three categories [5]:

Interactive tasks	Browsing histograms, making final fits, visualization tasks; these are typically performed on private desktops or laptops.
I/O bound tasks	Basically data mining; to get down to reasonable execution times these typically require hardware I/O able to give rates much higher than those provided by conventional I/O systems; for example, to get 10 TB processed in about 1 hour, a sustained speed of about 2 GB/s is required, which is reached with about 10 commodity hard drives.
CPU bound tasks	fast or full private simulations, toy-Monte Carlo for systematic studies.

A common feature of the I/O or CPU bound tasks in High Energy Physics (HEP) is that they can typically be formulated as *embarrassingly* parallel tasks, i.e. a parallel speed-up can be obtained by just splitting the job in sub-jobs. Another way to classify the end-user analysis activities is to consider the degree of interactivity as a function of the response time.

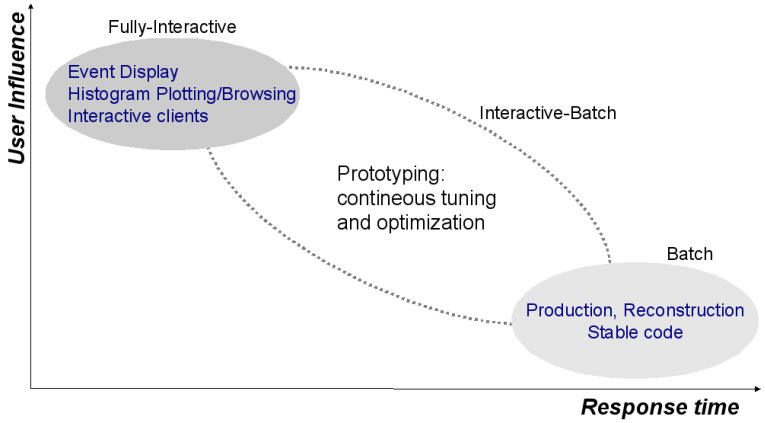

Figure 1: End-User analysis scenarios

This is shown in figure 1. One can clearly identify three regions in the plot:

Fully-interactive	Interactive tasks mentioned above; very short response times and full user control;
Batch	Monte Carlo production, reconstruction, format reduction, requiring sporadic tuning or optimization; these typically have long execution times and require very little interactivity;
Interactive-batch	Prototyping of selection or reduction algorithms, requiring repeated refinement cycles on relatively large data samples; varying level of interactivity.

At LHC, the *fully-interactive* tasks will be run on local desktops or laptops, using experiment specific applications (like event displays) or pure ROOT [2]. The *batch* tasks will be run on dedicated batch farms or the Grid using the submission and handling interfaces developed by each

experiment [6], or general purpose interfaces like Condor [7]. However, for the *interactive-batch* there is no common solution.

1.1.2 The traditional batch approach

In the traditional batch case we exploit the intrinsic embarrassing event parallelism to reduce the execution time by splitting the job in sub-jobs, which are run in parallel on the farm and whose outputs are merged at the end.

The advantage of this approach is that it implements job-level parallelism; therefore, typically there is no need to modify the user program: the same code can be run locally, on the batch system, and on grids.

The main disadvantage of this approach is that it uses a *push* architecture, which means that the sub-job partitioning is done *a priori*. Also, a batch system typically submits jobs sequentially thereby extending the total time by the submission time. In addition, the overall execution time with a push architecture is determined by the execution time of the slowest sub-job. It may therefore be subject to long tails, because there is no way to automatically redistribute the work of a under-performing worker.

Another disadvantage is the fact that real-time feedback about the status of processing would require special instrumentation to interface with a monitoring system. Also, exploiting multi-core machines requires a modification of the program.

1.1.3 The PROOF approach

PROOF addresses in particular the problem of long tails and realtime feedback. It implements a *master-worker* architecture schematically shown in figure 2, with the master in charge of distributing the work and merging the results. Efficient event level parallelism is achieved using a *pull*

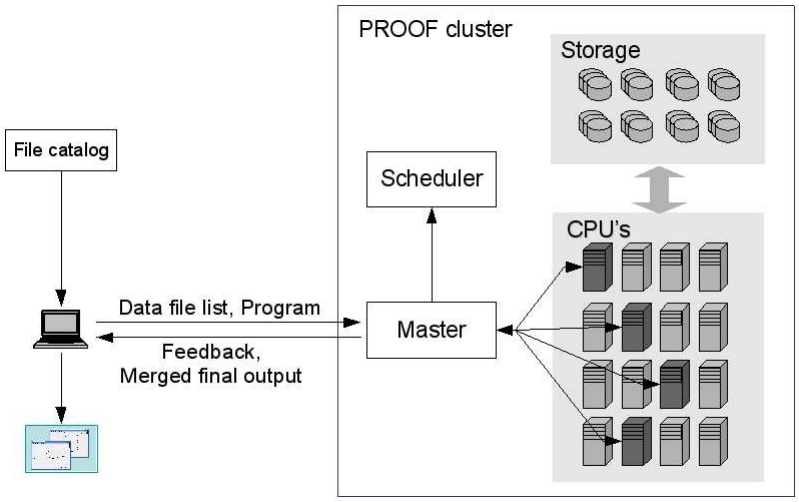

Figure 2: PROOF approach to the interactive-batch case: the master receives the job from the client, distributes it dynamically to the workers and collects and merges the results.

architecture to dynamically load-balance the available resources and make sure that all workers

finish at the same time. In this way the number of wasted CPU cycles is reduced, and the slower workers (the ones potentially creating the long tails) automatically receive less load and eventually are cut out. By design PROOF has real-time access to the latest job output and can give real-time feedback. It can also naturally exploit any additional available CPUs.

In the rest of this paper we discuss in more detail the PROOF system, its status and the main issues currently being addressed. Before that we need to recall briefly the ROOT approach to end-user data analysis.

2. Analysis with ROOT

The ROOT system [2] provides a software framework developed and optimized for HEP experiments but which also fits the needs of other analysis environments. Among many other things, ROOT provides: i) a storage system optimized for HEP data; ii) visualization tools (2D, 3D, event display, ...); iii) a complete set of mathematical and statistical functions and tools; v) a C++ interpreter. The C++ interpreter allows to run the same code in the shell and in custom applications, significantly facilitating code prototyping.

Apart form its shell-related facilities, the relevance of ROOT for the LHC analysis comes from the fact that all the experiments will store their data using the ROOT format [8, 9, 10, 11].

2.1 Model: trees, selectors

The HEP data are typically written once and read many times. Also, in the large majority of cases the needed information is only a small fraction of the total event record. To handle this case, ROOT organizes the information in tree-like structures which allows efficient access to individual data members of the event objects. In this model, that data is stored vertically allowing only the data actually used in a query to be read.

ROOT provides a framework for tree analysis which can be easily parallelized; this framework is called *selector framework*. A selector is composed basically of three main parts: *Begin()* where the job input parameters are parsed and the job output defined; *Process()* where the algorithm to be applied to each event record in the tree is run; *Terminate()* where the final part of the analysis (fitting,...) is applied on the output objects build during Process().

The Process() part of the selector can be easily parallelized, since it is called independently for each event-record and events are independent.

Selectors are implemented by inheriting from a dedicated ROOT class TSelector.

2.2 Data Access Issues

As mentioned above, the data to be analyzed will be residing in dedicated repositories at the storage elements of the Tier-2 centers. Efficient access to these data is required. Two solutions are currently envisaged: i) caching locally the files in, non-backed-up, disk pools using using dedicated file transfer tools; ii) use low latency remote access techniques. The first approach is more favored when the same files have to be used many times by more applications. The second is preferred when the file is used only a few times, typically by only one application.

Solutions for efficient remote access to data exists as discussed elsewhere in this workshop [12, 13]. The most favored in conjunction with PROOF is the one based on SCALLA/XROOTD [14],

an efficient remote file server system, providing Virtual Mass Storage capabilities, which has been recently adopted as data access protocol for LHC [15]. As we will see below, in PROOF clusters XROOTD is used for remote files access.

3. PROOF

The main goal of the PROOF system, is to run the ROOT analysis in parallel in a number of worker processes. Each worker process runs on its own CPU core or machine. PROOF aims at being a transparent extension of the ROOT single user session, so that the syntax differences to run on PROOF and locally are minimal. Due to minimization of the serial overhead PROOF achieves very good scalability and runs efficiently with many 100's of workers. PROOF optimizes the work distribution for *data locality*, meaning that a worker gets first assigned local data, thereby drastically reducing data transfer (over)loads in the cluster. Finally PROOF uses dynamic load-balancing to reduce CPU cycles.

PROOF uses the XROOTD [14] infrastructure to start its master and workers. This PROOF specific part is handled by a special XROOTD plugin, XrdProofd.

The PROOF system has already been presented at past editions of this workshop [3, 4]; below we only recall the main aspects of it.

3.1 Basic architecture

The PROOF system implements a multi-tier architecture sketched in figure 3.

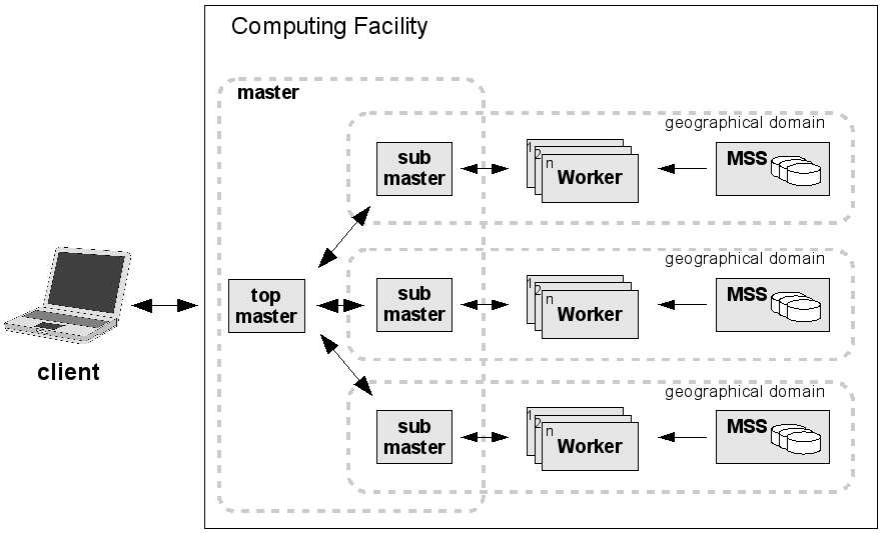

Figure 3: PROOF multi-tier architecture

The *client* runs standard ROOT and starts a PROOF session by contacting the *master*, the entry point to the computing facility. The role of the master is: i) to start a set of workers; ii) to parse the client requests and distribute the work to the workers; iii) to merge the result of the workers and return it to the client. The master tier can be multi-layered. This allows to distribute

the merging work, which may become the bottle-neck in the case of many workers. It also allows to federate geographically separated clusters by optimizing the access to distributed mass storages. In particular, it allows to adapt to a wide range of cluster sizes from multi-core desktops to the grid (gLite interface, gLitePROOF [16]).

Actually, to address the case of multi-core desktops and laptops, there is a dedicated version of PROOF, called PROOF-Lite, implementing a two-tier architecture where the master is merged into the client, controlling directly the workers. PROOF-Lite does not require any configuration and it is a convenient way to exploit the additional CPU resources of modern desktops/laptops.

3.2 Main Features

Interactive parallel execution of independent tasks Despite the fact that it has been developed initially to face the data-mining case, PROOF can be used for other problems that can be formulated as sets of independent tasks. An example are Monte-Carlo generations or studies based on toys Monte-Carlos. For example, an interface with the PYTHIA Monte-Carlo [17] exists and is available from the ROOT distribution.

Interactive-Batch mode Long PROOF jobs can be run in asynchronous mode, that is with the client getting back control of the local session; the connection client-master is stateless in this case, and the client can disconnect with the PROOF system continuing processing her/his job. The result is kept on the master and can be picked-up at any later time.

Real-time feedback The user can define a subset of the output objects to be send back at a tunable frequency for checks.

Package management Additional software required by the algorithm to be run can be loaded to the system in optimized way.

Flexible environment setting Special experiment-specific environment settings can be defined dynamically or statically.

Flexible authentication infrastructure PROOF makes use of the flexible, plug-in based, security infrastructure of XROOTD system [14].

3.3 Impact on existing analysis frameworks

To fully exploit the benefits of event-level parallelism, PROOF requires the code to follow the TSelector paradigm. This may interfere with the experiment frameworks, and in some cases it is seen as an obstacle to start working with PROOF. The TSelector framework is, however, quite flexible and a smooth transition should typically be possible. A solution adopted by some experiments, including CMS [18], is to modularize the way users use the frameworks, clearly separating out the parts of initialization, processing, termination. In this case a templated TSelector can be written to automatically use the same code used in the experiment framework. Another approach, currently followed by ATLAS [19], is to use a special TSelector to just start external tasks, which may be the same as those run in batch jobs (however here one looses some of the load-balancing optimizations of PROOF as each job runs to conclusion without reporting back its progress).

More generally, we believe that some changes in the user code are a reasonable price to pay for a better exploitation of the available resources. Also, moving to the selector framework implies a modularization of the code which typically improves the quality and allows users to concentrate on their algorithms.

4. Experience from PROOF users and installations

The PROOF system has been initially developed as a joint project between CERN and MIT, within the context of the PHOBOS experiment. The PHOBOS experiment used PROOF in production from 2003 to the end of their analysis program [20].

At LHC, the main PROOF users are ALICE, which requires a PROOF service as integral part of its computing model [8], and ATLAS, which is developing an alternative analysis model based on PROOF [21]. Some CMS groups are testing the CMS software framework on PROOF [18].

The ALICE and ATLAS facilities and users are currently the major sources of feedback driving the development of PROOF. Some more details about the current main installations are given in the appendix.

4.1 Performance tests

A series of performance tests were run to understand the scalability in CPU bound and I/O bound scenarios. Scalability test for CPU bound task were run on a 24-core machine [2]; not surprisingly, scalability is almost ideal, as it can be seen in figure 4 .

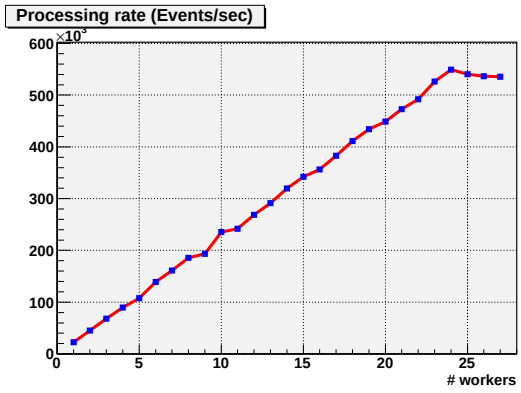

Figure 4: PROOF scalability for a CPU bound job on a 24-core machine [22]. The scalability breakdown when more than 24 workers are used is clearly visible.

For I/O bound cases, scalability tests have been run in several setups. Figure 5 summarizes the results at the ALICE CAF (see appendix A.1). Figure 5a shows the results obtained forcing one worker process per node, accessing local data; in this way one effectively gets a distributed I/O system and the scalability is very good. On the other hand, figure 5b shows saturation effects when using many workers on the same machine, indicating that the bottle-neck is hardware I/O. The plots in the figure show an important feature: the system is predictable, i.e. knowing the number of

[2]The machine was kindly provided by the CERN Openlab project [22]

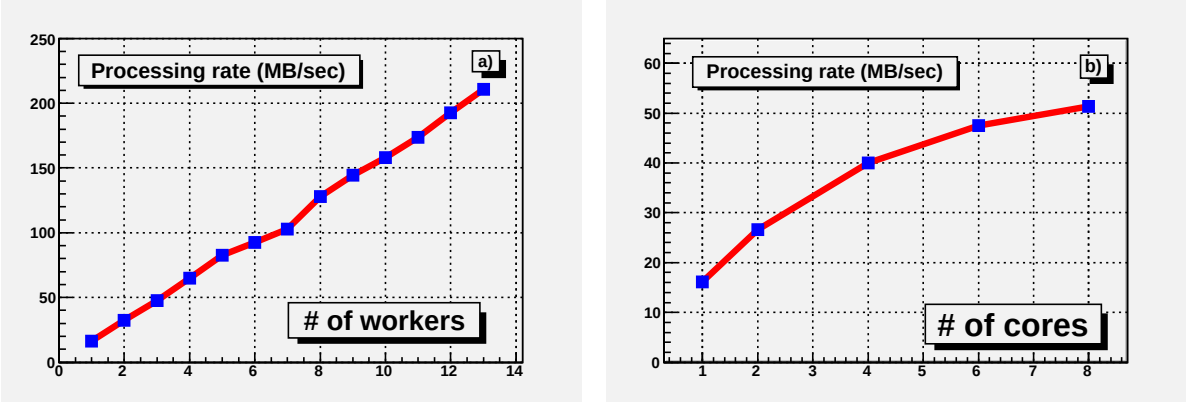

Figure 5: PROOF processing rate scalability for a I/O bound job on a ALICE CAF cluster (13 8-core machines; see appendix A.1): a) forcing one worker per machine; b) using up to 8 workers on one machine only. In this test the data were always read from the local disk of the machines. The rates include decompression and are consistent with a max raw I/O rate of about 200 MB/s.

machines and the number of workers per machines one knows the global throughput; this is very important for resource scheduling.

Preliminary results of the performance tests run at BNL with Solid-State-Disks show that SSD's allow to get very good scalability with many workers on the same machine [23]; if confirmed this result may indicate the way to go in the future to fully exploit the available CPU's for I/O bound tasks.

4.2 Data locality and Dataset management

Despite the significant improvements in network performance and in the efficiency of remote data access protocols, data locality is still an advantage, especially for repeated concurrent access to the same set of files, a typical use-case for end-user analysis. Currently, the direction is to go for data access models foreseeing a certain amount of local *cache*, either based on HDD or SSD or both. This approach, however, adds the problem of managing the files on this pool according to the policy decided by the experiment.

A dataset management system integrated with PROOF has been developed together with ALICE. Each physics group gets assigned a fraction of the available pool and it is fully responsible for the datasets in its partition. The system is up and running and was presented at CHEP07 [24]. ATLAS is working at a solution based on an SQL database managed pool following similar ideas, and the possibility to introduce an expiration token to automatically cleanup unused files is being considered [19].

4.3 Scheduling

Scheduling issues in PROOF were discussed in detail during the previous edition of this workshop [4]. Scheduling is needed to optimize the response time and the total throughput; this is achieved by controlling how many resources are assigned to each session and how resources are shared between running sessions.

In [4] we have shown how a technique based on re-nicing the sessions allows to get fair-sharing according to some experiment specific policy. As far as resource assignment is concerned, there is evidence that the Linux scheduler [3] works reasonably well if the number of sessions in not larger than the number of CPU cores in the worker machine. So what PROOF needs is a way to control the number of concurrent running sessions. The features of the PROOF scheduler, currently under test by ALICE and ATLAS, include dynamic worker assignment, priority management, FIFO queues, a worker re-nicing mechanism and a number of scheduling policies including load-based ones.

5. Summary

In this paper we have outlined some of the issues of the data analysis at LHC and how PROOF provides an alternative approach addressing use-cases where fast, (near) interactive, turn around is needed. We have also underlined how the different experiment driven developments on PROOF address issues of general interest, like data access, resource sharing and multi-core exploitation. The PROOF system is steadily maturing and becoming a full-featured solution for LHC analysis, in particular for Tier-2/Tier-3 clusters and/or multi-core machines.

References

[1] C. Eck, *et al.*, *LHC computing Grid : Technical Design Report*, CERN. Geneva. LHC Experiments Committee.

[2] R. Brun and F. Rademakers, *ROOT - An Object Oriented Data Analysis Framework*, in proceedings of AIHENP'96 Workshop, Lausanne, Nucl.Instrum.Meth. A389 81-86 (1997). See also http://root.cern.ch/.

[3] M. Ballintijn *et al.*, *Parallel Interactive Data Analysis with PROOF*, in proceedings of ACAT 05, DESY, Zeuthen, Germany, Nucl.Instrum.Meth. A559 13-16 (2006). See also http://root.cern.ch/drupal/content/proof.

[4] G. Ganis, J. Iwaszkiewicz and F. Rademakers, *Scheduling and Load Balancing in the Parallel ROOT Facility (PROOF)*, in proceedings of ACAT 2007, Amsterdam, The Nederlands; published in PoS ACAT:022,2007.

[5] F. Carminati et al., Common Use Cases for a HEP Common Application Layer, LHC-SC2-20-2002.

[6] GANGA, http://ganga.web.cern.ch/ganga/, used by ATLAS and LHCb;
PANDA, https://twiki.cern.ch/twiki/bin/view/Atlas/Panda, used by ATLAS;
CRAB, https://twiki.cern.ch/twiki/bin/view/CMS/SWGuideCrab, used by CMS;
AliEn, http://alien.cern.ch/twiki/bin/view/AliEn/Home, used by ALICE.

[7] *The Condor System for High Throughput Computing*, http://www.cs.wisc.edu/condor/; Condor is mostly popular among the US ATLAS and CMS groups.

[8] ALICE Collaboration, *T.D.R. of the Computing*, CERN-LHCC-2005-018

[9] ATLAS Collaboration, *Computing T.D.R.*, CERN-LHCC-2005-022

[10] CMS Collaboration, *The Computing Project T.D.R.*, CERN-LHCC-2005-023

[3] All the major PROOF installation are Linux based.

[11] LHCb Collaboration, *Computing T.D.R.*, CERN-LHCC-2005-019

[12] A. Hanushevsky, "Are SE Architectures Ready For LHC?", presented at this workshop.

[13] F. Furano, "The ALICE Global Redirector. A step towards real storage robustness", presented at this workshop.

[14] A. Hanushevsky et al., *The Scalla Software Suite*, http://xrootd.slac.stanford.edu/.

[15] A. Peters, "XCFS - An analysis disk pool and filesystem based on FUSE and XROOT protocol", presented at this workshop.

[16] A. Manafov, *PoD, PROOF on Demand, an implementation of the PROOF distributed data analysis on different batch systems and/or on the Grid.*, https://subversion.gsi.de/trac/dgrid/wiki/PoD.

[17] T. Sjostrand, *The PYTHIA Monte Carlo*, http://home.thep.lu.se/ torbjorn/pythiaaux/present.html.

[18] See, for example, https://twiki.cern.ch/twiki/bin/view/CMS/HamburgWikiComputingNAFPROOF .

[19] Neng Xu and G. Carrilo, University of Wisconsin, private communication.

[20] See M. Ballintijn, *PROOF at Phobos*, Application Area Meeting, CERN, May 24, 2006, http://indico.cern.ch/conferenceDisplay.py?confId=a062074

[21] See N. xu and S.Panitkin talks at PROOF 2007, CERN, Nov 2007, http://indico.cern.ch/conferenceDisplay.py?confId=23243

[22] *CERN Openlab*, http://proj-openlab-datagrid-public.web.cern.ch/proj-openlab-datagrid-public/.

[23] S.Panitkin, BNL, private communication.

[24] J.F. Grosse-Oetringhaus, *The CERN analysis facility: A PROOF cluster for day-one physics analysis*, presented at CHEP07, Vancouver, Canada; published in J.Phys.Conf.Ser.119:072017,2008.

[25] A. Gheata, "ALICE Analysis Framework", presented at this workshop.

[26] A. Kreshuk, "Interactive Data Analysis with PROOF: Experience at GSI", presented at this workshop.

Appendix

A. The main PROOF installations

A.1 ALICE

CAF Located at CERN, the CAF is the embryo of what will become the CERN Analysis Facility (CAF) for ALICE, a cluster of about 500 cores and a few 100 TB of local storage, intended for prompt analysis, calibration, alignment and fast-simulation. Currently the cluster has 112 cores and 35 TB of local storage. It is used regularly by 5÷10 people out of about 50 registered. The activities at this facility are described in detail elsewhere at this workshop [25].

GSIAF Located at GSI-Darmstadt, the GSIAF (GSI Analysis Facility) has currently 160 cores and 150 TB of storage under Lustre. It is used by 5÷10 people for data analysis and ALICE-TPC calibration. The activities at this facility are described in detail elsewhere at this workshop [26].

A.2 ATLAS

BNL There are two facilities at BNL: one for testing, with 72 cores, 25 TB of local storage and 192 GB of storage on Solid-State-Disks (SSD); one for data analysis with 40 cores and 20 TB of local storage. These facilities have been used to test I/O performances with RAIDS and SSDs, and to test cluster federation.

Wisconsin The Wisconsin facility has 200 cores, 100 TB of local storage, multi-RAID5. This is used for data analysis prototyping (Higgs searches) and I/O performance tests w/ multi-RAIDs.

Additional ATLAS facilities are present at LMU, Munich and UTA.