Eur Phys J C **34**, s01, s425–s431 (2004)
Digital Object Identifier (DOI) 10.1140/epjcd/s2004-04-046-2

EPJ C direct
electronic only

LHC physics analysis – Worldwide distribution

N.H. Brook

H.H. Wills Physics Lab., University of Bristol, Bristol BS8 1TL, United Kingdom, e-mail: n.brook@bristol.ac.uk

Received: 13 August 2003 / Accepted: 17 November 2003 /
Published Online: 13 July 2004 – © Springer-Verlag / Società Italiana di Fisica 2004

Abstract. The generic needs and requirements of performing analysis in the LHC era are examined. These requirements are addressed assuming a distributed computing environment such as that offered by Grid technologies. The developments and the tools being evolved in the individual LHC experiments are presented.

1 Introduction

The LHC [1] will commence in 2007 and will collide protons with protons at a centre of mass energy of 14 TeV. The collision rate of the bunches will be 40MHz, corresponding to 10^9 events every second, at design luminosity. In addition the LHC will operate in a heavy ion mode producing collisions at 5.7 TeV per nucleon.

There are five LHC experiment: two generic detectors (ATLAS [2] and CMS [3]) and three specialised detectors. These are ALICE [4], whose primary aim is the study of heavy ion physics, LHCb [5], uniquely designed for the investigation of the B-physics and in particular CP-violation, and TOTEM [6], dedicated to measuring the total cross section, elastic scattering and diffractive dissociation at the LHC.

There are over 5000 physicist from all over the world working on the LHC experiments. It is essential that all the physicist have access to the data stored and processed by each of the experiments. The necessary computing resources have to be available to process the data and to perform a subsequent analysis. Practically, and politically, it is not possible to have all the needed computing resources based at the accelerator laboratory, therefore a distributed computing resource environment will need to be developed. The 'so-called' Grid computing [7] paradigm seems adaptable to the computing needs of the LHC.

1.1 Complexity of the problem

ATLAS alone has 150 million electronic read-out channels that will have to be processed during data taking. This is approximately two orders of magnitude greater than present day High Energy experiments. The original interaction rate of 40MHz will have to be reduced, through a series of sophisticated "trigger" algorithms, to the order of 100Hz or less before the events are made persistent. Many of the higher-level trigger algorithms will be software based. Even at the start up of the LHC, at lower luminosities than design, the experiments will have to store several Terabytes, if not Petabytes, of data per year. In addition to collating the data as it comes off the detector, it will be necessary to generate huge amounts of Monte Carlo simulated data. Both the 'real' data and the simulated data will have to be reconstructed to form physics object on which a physicist will perform analysis. The latest estimates of the computing resources needed at CERN alone in 2008 are given in Table 1. The current working assumption is that at least twice the computing capacity of CERN will be available externally.

Table 1. Computing resources needed by the LHC experiments in 2008. A typical 2 GHz processor is about 700 SI2000 units

Experiment	CPU (kSI2000)	Disk (TB)	Tape (PB)
ALICE	7416	384	2.3
ATLAS	5200	1300	12.6
CMS	5684	1769	9.15
LHCb	810	330	1.0

The worldwide nature of the LHC collaborations means any adopted computing model must be distributed in nature. Computing outside of CERN is developing around large computing centres or Tier-1's (such as Brookhaven or Fermilab in the US and Karlsruhe, France and RAL in Europe.) The tier's classification [8] will be based on the quality of service each centre can offer to the LHC computing problem. In many places, a 'cloud' structure is emerging that sees individual computing facilities being advertised as a single, coherent structure. An example of one such 'cloud' structure is ScotGrid [9], in the UK, which has a compute engine based in Glasgow with a mass store in Edinburgh.

Physicists, to be able to work in this environment, have to adapt. Already tools for collaborating and communicating at distances, such as the Virtual Room Videoconferencing system (VRVS) [10] and AccessGrid [11] have been developed and are being adopted as common practice. It is only through further development of such tools will teams of physicists be able to interact in a normal, natural environment limiting the need of air travel and maximising their collaborative effort.

2 Analysis

Computing in Particle Physics can be classified in two distinct types. The first is production jobs such as Monte Carlo simulation or event reconstruction. These jobs are planned in advance and perform a homogenous set of tasks. The input is a pre-determined set of events accessed sequentially, processed and then written out. In general, this activity is centrally organised by the experiments. The second is of a more chaotic nature and covers the analysis activity. Analysis jobs are submitted by many users acting, to first order, independently of each other. The input is typically a selection/analysis algorithm to be applied to large samples of data. There is a high probability that there will be a 'sparse' data access pattern as opposed to the sequential access encountered for production jobs. Physicists will be submitting analysis jobs at any time and could well be attempting to access simultaneously a dataset required by another job.

Figure 1 illustrates a typical dataflow involved in an analysis. The Monte Carlo simulation and the reconstruction are the centrally organised task involving relatively few people; these are relatively CPU intensive activities. The Monte Carlo simulation task will be an ongoing collaboration activity with the relative priority between the various simulation needs being decided within the collaboration. A first pass of data reconstruction will be performed essentially as the raw data is processed from the detector. Additional reconstruction of the data will be performed as the alignment and calibration of the detectors is refined and improved. This enhanced reconstruction step will be performed 2-3 times a year. A number of physics analysis groups will exist in each experiment, in the large collaborations there could be of the order of 20 such groups. Each of these groups will have a set of algorithms that they will apply to the data in order to (pre-)select the events of interest to their particular physics topic. A first pass with these algorithms could well occur in the production steps of the simulation and the reconstruction. The CPU needs of this selection step will be relatively small and as the algorithms are improved and selection criteria changed there will be a need to re-run the group selection jobs on almost a monthly basis. It is hoped that the pre-selection will reduce the number of events of interest by a factor of 10-100. The final step will be individual physicists running over the group datasets. The aim will be to enhance and finalise the physics selection criteria and produce the physics plots for publication. Whilst these jobs will be the least CPU intensive there will be the most chaotic, with many users running jobs simultaneously.

The ability to access data files is the cornerstone of any distributed computing model. The data available will be characterised by 'metadata.' This metadata will allow datasets to be located and accessed efficiently. Dataset metadata will include information such as the year the data was taken, the provenance (history) of the process creating it. In addition, there will be event-level metadata that will characterise an event within the dataset. This metadata will include information such as the triggers the event passed, high-level particle information. A typical analysis would consist of performing a query on the dataset metadata catalogue to determine which logical datasets meet a broad criteria (e.g. all data from 2008) and then performing a query to select the events and their components that may be of interest based on the event-level metadata.

The analysis environment will be designed around the distributed data store and computing infrastructure. In general, experimental software will not build directly on this infrastructure but will build upon generic software ("middleware") that performs the low level tasks. This generic Grid software will be provided through projects such as the Virtual Data toolkit [12] and the European DataGrid (EDG) [13] and its successor. This middleware, building on the low level tools provided by Globus [14], will provide the basic building blocks for the experiments' software development for a distributed environment. The LHC Computing Grid (LCG) [15] will provide an upper layer of middleware that is experiment independent but provides a higher level of functionality that matches the generic needs of the LHC physics community. In extreme cases, the experiments may well develop their own tools, based on the basic toolkit, to provide an application that meets a specific requirement. It is important for the experiments that the generic middleware developments provide a robust and coherent set of tools and mechanisms.

Once the experiments software framework is integrated with the Grid middleware the physics applications, such as the simulation, reconstruction and analysis, will be able to function in the distributed environment. In addition to integrating the experiments' applications, the user interfaces to the Grid (for the non-expert) need to be developed in parallel, allowing easy and transparent access to the resources, data and applications. Ideally this user interface will be independent on whether the final analysis is in a Grid environment or just utilising local resources e.g. local batch farm or even the PC's own CPU. Such a user interface should be stable, any changes to the underlying architecture should be invisible to the end user and any increased functionality should be incorporated in a manner that is not intrusive to the already established environment.

Many of the tools that will be required in the distributed environment are already familiar to physicists performing analysis e.g. generic analysis tools and detector/event display. It is essential that the ease of use of these tools in the Grid environment is as simple as that of the non-Grid environment. Other tools, that to

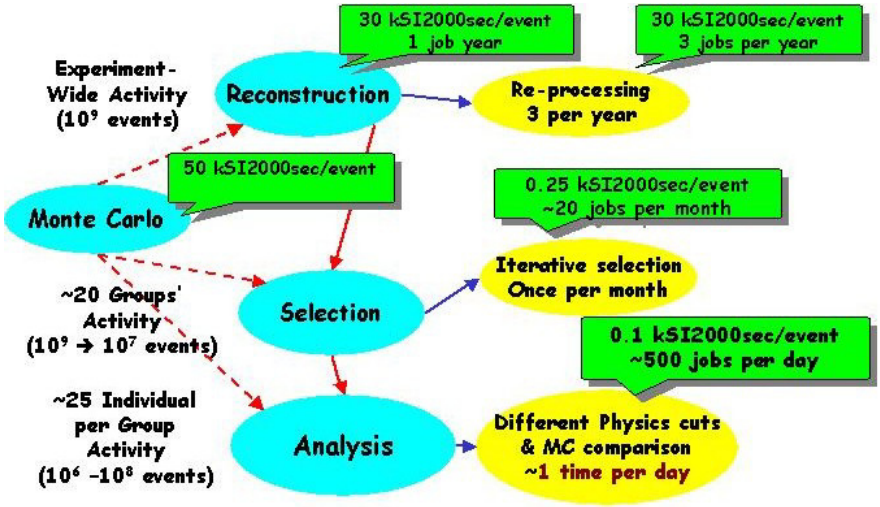

Fig. 1. Schematic of steps in a Particle Physics Analysis

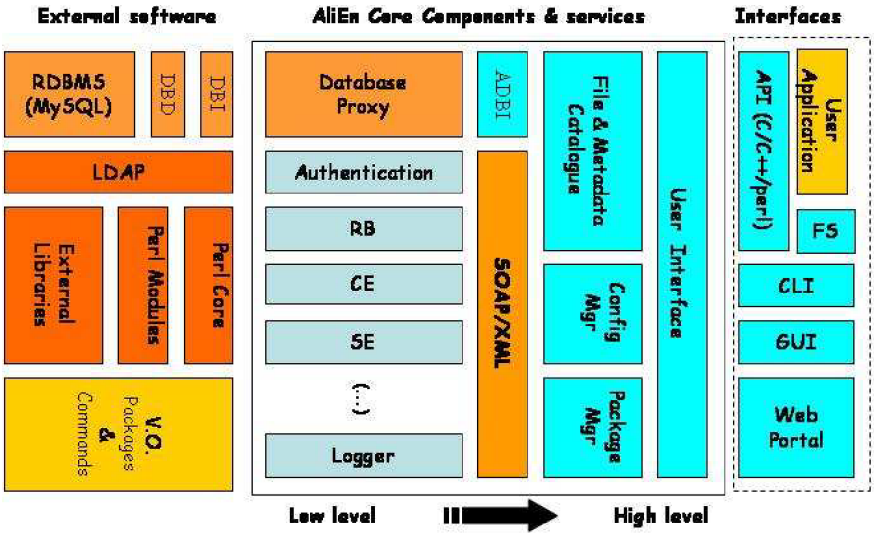

Fig. 2. A schematic view of the AliEn components

a lesser or greater extent already exists for analysis, will become increasingly important in a distributed environment e.g. data browser, an analysis job wizard, data management tools and software development and installation tools. These tools will further hide the complexities of the Grid environment from the user and maximize the productivity of the physicist. A job wizard would allow the straightforward creation and configuration of a job with automatic generation of the job script. The management of the data is critical in a distributed environment and therefore it is crucial tools for data manipulation are provided. One example of the need for data management is the physicist preparing the data (alignment and calibration data, data sets, etc.) that would have to be downloaded to a laptop from the Grid in order to allow an analysis to be pursued whilst there is no internet connection.

3 Role of the LHC computing grid

The role of the LCG Project is to provide the computing infrastructure for the processing of the LHC data for all of the LHC collaborations. This includes both the software support for the physics application software, and the development and deployment of the computing services needed to store and process the data, providing batch and interactive facilities for the worldwide community of physicists involved in LHC. The project is organised around four areas: applications; computing fabrics; Grid technology and Grid deployment. A production version of the LCG Grid will be ready for use by the experiments in the Data Challenges scheduled for 2004.

Within the LCG applications area, there currently exists five sub-projects: software process and infrastructure (SPI), persistency framework (POOL), core libraries and services (SEAL), physicist interface (PI), and simulation.

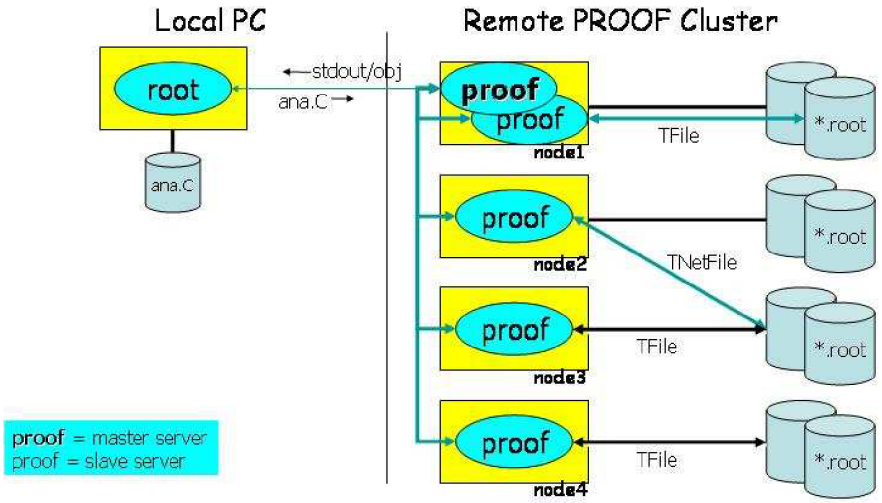

Fig. 3. A schematic view of the PROOF system

It also provides assistance in the integration of the physics applications in the Grid environment. The overall aim is to provide common software infrastructure, frameworks, libraries and tools along with common applications such as simulation and analysis toolkits. In addition studies are ongoing looking at the Grid interfaces for the experiments, this will build upon the work already ongoing in the individual experiments, which is discussed in the following sections.

4 AliEn

AliEn [16] has been developed by the ALICE experiment, though it is not ALICE specific. The aim within ALICE is that all computing resources that are part of the ALICE virtual organisation (VO) can be used as a single entity in a transparent manner by the user. AliEn consists of an authentication service; a querying system; a file catalogue and replication service; the storage (SE) and computing (CE) elements and a resource broker (RB.) AliEn has been built on top of common Open Source components.

AliEn has a modular design. This enables the components to be group together in packages which are self-contained and have no external dependencies. A schematic view of the AliEn components is shown in Fig. 2. AliEn can be installed without system privileges and the components can be configured using the Configuration Manager. The details of the configuration are stored in a LDAP (Lightweight Directory Access Protocol) [17] directory. The various services shown communicate via the extensible markup language (XML) [18] using SOAP (Simple Object Access Protocol) [19]. Users can interact with AliEn through several methods such as application programme interfaces (APIs), a command line interface (CLI), a graphical interface (GUI) or a web portal. As new Grid middleware is developed and becomes available it will implemented and evaluated within AliEn.

Datasets associated with a job are registered in the AliEn file catalogue. This is a virtual file system that maps one or more physical file names (PFN) to a logical file name (LFN.) Users see only LFNs and AliEn will translate them into the most appropriate PFN dependent on the location of the client. The interface to this catalogue looks like a UNIX file system to the user. In addition, it allows the user to replicate data and also to store data describing file contents ("metadata.") Underlying the file catalogue is a relational database (RDB) that is interfaced to the AliEn core components and services.

The Package manager allows contributions from different VO's to be managed e.g. including experiment specific software. The package manager's tasks is to install the appropriate packages and prepare the execution environment. It will be aware of different versions and any interdependencies between packages.

A SE is responsible for managing physical files and the interface to the local mass storage system(s). It deals with space allocation, file caches as well as file integrity issues. The CE service is an interface to the local batch system. When a CE has available resources to execute a job, it notifies the resource broker (RB) service of its availability and capabilities. The RB will attempt to match the needs of a job, registered in a database table, with those being advertised and flag the job for execution on a particular CE if a match is made. The AliEn RB uses a pull architecture, as opposed to the push mechanism adopted by the EDG, and as such does not need to know the status of all resources in the system.

Once the data have been produced and catalogued by the AliEn framework, they can be analysed via a distributed tool based on PROOF [20], the parallel ROOT [21] facility.

5 PROOF

The Parallel ROOT facility, PROOF, is a system for the parallel interactive analysis of large datasets on clusters of heterogeneous computers. PROOF is built upon the ROOT framework making use, for example, of its object

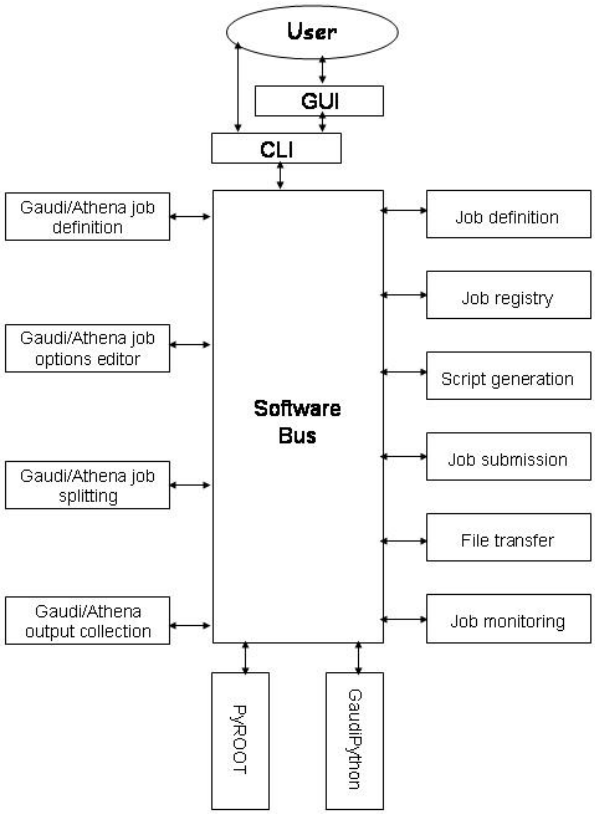

Fig. 4. A schematic representation of the GANGA design, which is based on components interacting via a software bus

containers, object streaming, scripting, networking classes and remote file access.

PROOF consists of a 3-layer architecture: a ROOT client session, a PROOF master server and PROOF slave servers. This is illustrated in Fig. 3. The ROOT session client session creates a master server on a remote cluster and the master server in turns creates slave servers on all the nodes in the cluster. The analysis script *ana.C* is sent to the master and then the slave servers. All the slave servers execute the analysis script(s) and queries on a chain of ROOT trees in parallel. The system is based around a "pull" mechanism, each slave server asks the master for work packets. The master distributes work packets, of differing size, dependent on the speed of the particular slave. This approach ensures that PROOF is adaptable to the performance and load of the individual cluster nodes. The packet size is tuned taking into account the duration of each small job (packet), bandwidth and latency of the network. It is necessary to carefully balance the performance of each node with the communications overhead.

The location of the input data is also essential for maximising the performance of the system. In general, a large number of data files will want to be analysed. These files could well be distributed over the different nodes in a cluster or even geographically distributed over computing resources connected by the local (or wide) area network. ROOT allows files to be group together in a single logical entity. To optimise performance, packets that contain data local to the node are distributed to the slave servers first. Only when all local data has been processed will a packet be assigned to a slave server that needs to access remote data. The master maintains knowledge of all generated packets per slave, thus allowing other slaves to reprocess packets "lost" if a particular slave dies during processing. Each slave sends an object (histogram) back to the master, which adds the individual histograms from each slave together.

In the Grid context, the PROOF model will be extended from local clusters to a distributed "virtual cluster." It envisaged that the slave servers are started via the Grid RB. It will need to interact not only with the RB but also the Grid file catalogues and monitoring services in order to optimise the resources for the job parallelisation. This will further enhance the transparent access to resources.

6 GANGA

The GAUDI/ATHENA software framework [22] is used by LHCb and ATLAS as the framework for all event processing applications such as simulation, reconstruction and

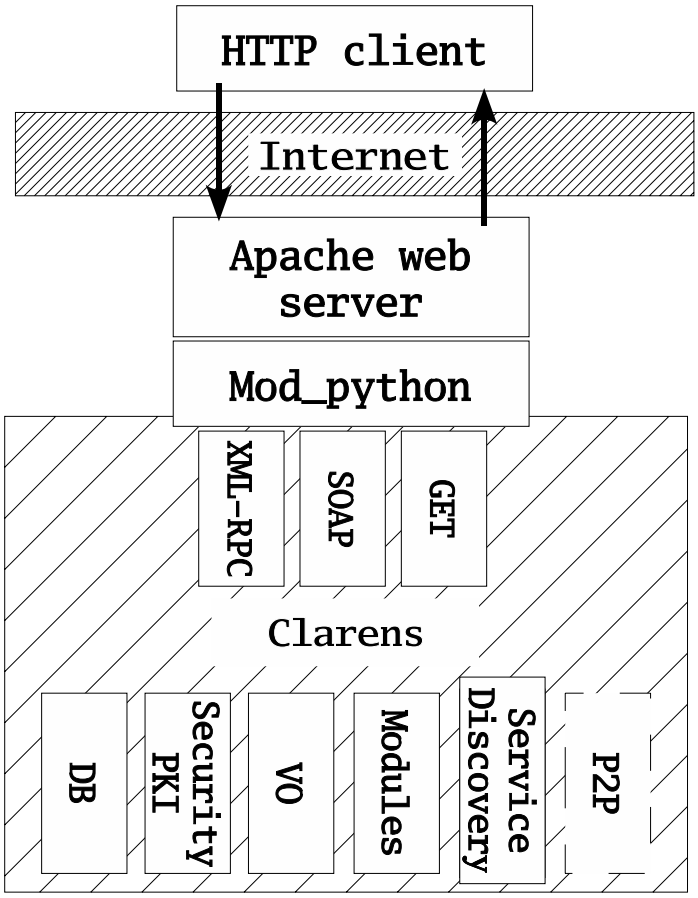

Fig. 5. Clarens architecture diagram

physics analysis. The front end to this framework is known as GANGA[1] [23].

GANGA spans all aspects of the cycle of a job: creation, configuration, splitting and recollection, script generation, file transfer to and from worker nodes, submission, run-time setup, monitoring and reporting. In the case of GAUDI jobs it also allows configuration of the algorithms to be run and the specification of the inputs and outputs. GANGA relies on Grid middleware from other project but aims to make the functionality of the middleware transparent.

GANGA is being implemented using the PYTHON OO-scripting language [24]. The modular design of GANGA is well-suited to the PYTHON language. The components of GANGA interact with one another through, and managed by, a so-called "software bus." The PYTHON software bus does not have any privileges over other modules and hence components written for the bus can be used as PYTHON modules without the bus. The interplay between the components and the bus is shown schematically in Fig. 4. The functionality of the GANGA components can be accessed through a CLI and/or a GUI built on common APIs.

GANGA has a set of core components suitable for job handling tasks in a wide range of application areas. These components provide the implementation for the job definition, the editing of job options, the splitting of jobs based on user provided configuration and job templates, and the output collection. Other GANGA components of general applicability performs operation on, for or using job objects e.g. file transfers, job registry or job submission. These general components are illustrated on the right hand side of Fig. 4.

For the current user groups, ATLAS and LHCb, specialised components exist that incorporate knowledge of the GAUDI framework, illustrated on the left hand side of Fig. 4. For example, applications based on GAUDI are packaged using a configuration tool (CMT [25]) that requires its own elements. Specialised components for other applications areas are easily added. The sub-division into generalised and specialised components allows new user groups to identify quickly the components that match their requirements. In addition GANGA is supplemented by the functionality of external components, including non PYTHON components such as ROOT and the GAUDI framework itself. These are shown along the bottom of Fig. 4.

The GANGA design, including possibilities for creating, saving, modifying, submitting and monitoring jobs, has been partly implemented and released. The current release allows submission to the EDG as well as local PBS

[1] GAUDI/ATHENA and Grid Alliance

and LSF queues. The first release does not yet fulfil all requirements but it already helps the user to perform a number of otherwise routine, manual tasks.

7 Clarens

Clarens [26] is a Grid-enabled web service infrastructure developed for use within CMS. Clarens acts as a broker between services and clients. For the client side it deals with authorisation, routing service requests, deserialisation of information and any encryption or compression needs. Whilst for the server it is responsible for contacting the service, passing the parameters from the client, the serialization of the output from the service and managing client session and service process. The Clarens servers is implemented as an extension to the Apache web server [27] using the mod_python extension of the PYTHON language. It uses XML-RPC [28] and SOAP protocols to act as the intermediary for the distributed clients to access the services.

The Clarens architecture is shown in Fig. 5. The Apache server receives a HTTP request from the client and invokes Clarens based on the form of the URL specified by the client. After the request has been processed, a response is sent back to the client.

For authentication Clarens make use of the Public Key Infrastructure [29] and relies on certificates issued by a Certificate Authority. The authentication protocol of the server is implemented at the application level, thus removing the requirement for a custom security layer on the client side. This has the benefit for the user not having to propagate their credentials to every computing system they wish to use to access resources. In addition to authentication a user needs authorisation to access server resources. Within Clarens this is done with the concept of a tree-like virtual organisation of groups (e.g. ATLAS VO, CMS VO) with members identified by unique distinguished names (DNs) issued by the Certificate authority as part of the certificate.

To provide a uniform interface for an external application to access various storage media, CLARENS will be interfaced to the San Diego SuperComputing Storage Resource Broker (SRB) [30]. An interface has been designed that allows a simple access to the job scheduler through a jobID. This interface is used within CMS to develop interactive remote analysis. Interactivity is possible with a long running analysis process on a cluster. Another interface has been developed that allows remote users to query large datasets using standard relational database queries. The results are returned in the form of object file formatted as a ROOT tree. In addition, CLARENS allows remote access of ROOT files, thus opening up the full functionality of ROOT for transparent analysis of remote files.

8 Summary

The analysis needs of the physicists in the LHC era provide a very challenging problem. These needs are thought to be best addressed through the concept of Grid computing. Computing centres from around the world are beginning to collaborate under the auspices of the LCG project, bringing together the computing resources needed to meet the needs of the LHC.

Effort has currently focussed on delivering the tools and resources for centrally coordinated, production tasks such as Monte Carlo event simulation. As first collisions at the LHC moves ever closer, the focus will shift to meet the needs of the more challenging, chaotic environment of analysis with strong emphasis on data management. Tools needed to exploit these distributed resources in a transparent manner are beginning to be developed. The ideas and needs for analysis and how best to access the appropriate resources will continue to develop rapidly over the coming years.

Acknowledgements. I would like to thank Predrag Buncic (ALICE/AliEn), Fons Rademackers (ALICE/PROOF) and Vincenzo Innocente (CMS) for their help and patience as I put this presentation together. I still claim full responsibility for any mistakes that have propagated to the final version.

References

1. http://user.web.cern.ch/user/Index/LHC.html
2. http://atlas.web.cern.ch/Atlas/Welcome.html
3. http://cmsinfo.cern.ch/Welcome.html
4. http://alice.web.cern.ch/Alice/
5. http://lhcb.web.cern.ch/lhcb/
6. http://totem.web.cern.ch/Totem/
7. I. Foster, C. Kesselman, and S. Tuecke: Int J. Supercomputer Application 15,3 (2001)
8. L. Bauerdick et al.: RTAG5 LCG report, http://lcg.web.cern.ch/LCG/SC2/RTAG6/finalreport.doc
9. http://www.scotgrid.ac.uk/
10. http://www.vrvs.org/
11. http://www.accessgrid.org/
12. http://www.lsc-group.phys.uwm.edu/vdt/project.html
13. http://eu-datagrid.web.cern.ch/eu-datagrid/
14. http://www.globus.org/
15. http://lcg.web.cern.ch/
16. http://alien.cern.ch
17. http://www.openldap.org/
18. http://www.w3.org/TR/1998/REC-xml-19980210
19. http://www.w3.org/TR/SOAP/
20. M. Ballintijn et al.: 'The PROOF Distributed Parallel Analysis Framework Based on ROOT', physics/0306110
21. http://root.cern.ch
22. http://proj-gaudi.web.cern.ch/proj-gaudi/
23. http://ganga.web.cern.ch/
24. http://www.python.org/
25. http://www.cmtsite.org/
26. C. Steenberg et al.: 'The Clarens Web Services Architecture', cs.DC/0306002; C. Steenberg et al.: 'Clarens Client and Server Applications', cs.DC/0306001, http://clarens.sourceforge.net
27. http://www.apache.org/
28. http://www.xmlrpc.com/
29. http://www.pkiforum.org/pdfs/PKI_Basics-A_technical_perspective.pdf
30. http://www.npaci.edu/DICE/SRB/