



Università degli Studi di Ferrara

DOTTORATO DI RICERCA IN INFORMATICA
CICLO XXVI

COORDINATORE PROF. MASSIMILIANO MELLA

**Enabling parallel and interactive distributed
computing data analysis for the ALICE
experiment.**

SETTORE SCIENTIFICO DISCIPLINARE INF/01

TUTORE:
Prof. Eleonora Luppi

DOTTORANDO:
Cinzia Luzzi

CO-TUTORE:
Dr. Latchezar Betev

ANNI 2011/2013

To my uncle.

Contents

Introduction	1
1 Computing for High Energy Physics	3
1.1 LHC and the ALICE experiment at CERN	3
1.1.1 The ALICE Experiment	7
1.2 HEP computing	8
1.2.1 Grid computing	9
1.2.2 Cloud computing in High Energy Physics	10
1.3 Data analysis technique	14
1.3.1 Interactive data Analysis	15
1.3.2 Batch data Analysis	15
1.3.3 Interactivity on batch resources on the Grid and the Cloud: sta- tus of the field	17
1.3.3.1 Interactivity through the PROOF framework	18
2 The Grid Technology and the ALICE Environment: AliEn	21
2.1 The Grid Paradigm	21
2.1.1 Grid Architecture	22
2.1.2 The Worldwide LHC Computing Grid	24
2.2 ALICE computing model	28
2.3 The ALICE Grid environment	28
2.3.1 External software components	30
2.3.2 AliEn core components and services	30

CONTENTS

2.3.2.1	File and Metadata Catalogue	31
2.3.2.2	Workload Management System	32
2.3.2.3	Additional services	34
2.3.2.4	Distributed analysis	35
2.3.2.5	Data management tools and services for data transfer, replication and storage	36
2.3.2.6	Authentication and Authorization	37
2.3.2.7	Monitoring	38
2.3.3	User Interfaces	41
2.3.4	Future AliEn development: JAliEn	43
3	The ROOT parallel analysis facility and the dynamic PROOF frame- work	47
3.1	ROOT analysis framework	47
3.1.1	The access and storage data system	48
3.1.2	The C++ interpreter	50
3.1.3	Mathematical and statistical analysis algorithms	50
3.1.4	The GUI and the Event Visualization Environment	50
3.1.5	The Virtual Monte Carlo Interface	51
3.2	The PROOF framework	52
3.2.1	System architecture	52
3.2.2	Use case	54
3.2.3	PROOF features	55
3.2.3.1	PROOF load balancing engine	55
3.2.3.2	PROOF resource scheduling	57
3.3	The PROOF cluster at CERN: CAF	57
3.4	The PROOF On Demand framework	60
3.4.1	PROOFAgent	62
3.4.2	PAConsole	63
3.4.3	PoDWorker script	63
3.4.4	PoD utilities, configuration files and use cases	64

4	Enabling PROOF on the AliEn System	69
4.1	Benefit of having PROOF on the Grid	70
4.2	Architecture and Implementation	71
4.2.1	PoD on AliEn: first prototype	72
4.2.1.1	PoD on AliEn: first prototype - workflow	74
4.2.2	PoD on AliEn: second prototype	77
4.2.2.1	PoD on AliEn: second prototype - workflow	78
4.3	Results	82
4.4	Summary	86
5	The CAD interface for the ROOT geometry modeller	87
5.1	The ROOT geometry modeller	87
5.1.1	ROOT Shapes	89
5.2	The TGeoCad Interface	89
5.2.1	The STEP Standard	91
5.2.1.1	Components of STEP	92
5.2.1.2	Geometry Data Exchange: AP203 - AP214	93
5.2.2	The Open CASCADE Technology	93
5.2.2.1	Foundation Classes	94
5.2.2.2	Modelling Data	95
5.2.2.3	Modelling Algorithm	95
5.2.2.4	Extended Data Exchange	95
5.2.2.5	Application Framework	96
5.2.3	Details of the conversion	96
5.2.4	Structure of the interface	98
5.2.4.1	TOCCToStep class	99
5.3	Technical details	101
5.4	Results	103
	Conclusion	105
	Bibliography	107

CONTENTS

Introduction

This thesis has been carried out under the auspices of the University of Ferrara in collaboration with the Offline group of the ALICE experiment at CERN.

The thesis aims to provide interactivity on the Grid and to extend the capabilities of ROOT interactive data analysis framework used by High Energy Physics (HEP) experiments.

A ROOT interface to CAD systems has been developed in order to improve the level of collaboration between physicists and engineers who are working, respectively, on the detector simulation and on the mechanical design of the detector, often using different softwares.

The PROOF framework, which is an extension of ROOT to provide parallel and interactive data analysis on static cluster, has been integrated in the AliEn Environment i.e. the distributed system used by the ALICE experiment. In particular, PoD, the dynamic version of PROOF, has been integrated in AliEn providing interactive and parallel data analysis on the Grid.

The work has been divided in five chapters. Chapter 1 presents the scientific goals of the LHC and ALICE experiment. In this context, it introduces the HEP computing architectures and specifically the Grid and Cloud computing paradigms. Finally, the principles and techniques of data analysis are discussed, as well as the batch and interactive solutions used to perform it.

Chapter 2 introduces the Grid architecture, the largest scientific distributed computing tool, its infrastructure and middleware. In particular, the ALICE computing model and the AliEn system used to implement it are described.

Chapter 3 explains the ROOT framework and the PROOF architecture. Two ways

INTRODUCTION

of exploiting PROOF are described: the static PROOF cluster at CERN and the dynamic PROOF solution at GSI.

Chapter 4 presents the “*PoD on AliEn*” project that is a Grid service to provide interactive and parallel jobs execution in a distributed environment. The advantage of performing interactive and parallel analysis on the Grid is highlighted. The results obtained are shown.

Chapter 5 describes the implementation of the “*TGeoCad Interface*” that consists of an interface to improve the compatibility between the ROOT framework and the CAD systems by providing the possibility to export a ROOT geometry in a STEP format which is supported by many CAD systems.

Chapter 1

Computing for High Energy Physics

The chapter explains briefly the scientific goals of LHC [1] and the ALICE experiment [2]. It gives an overview on the computing architectures adopted by High Energy Physics (HEP) experiments. Different requirements for the physics data analysis will be discussed, leading up to the introduction of interactive and distributed data analysis.

1.1 LHC and the ALICE experiment at CERN

The main aim of *High Energy Physics* (HEP) experiments is to understand what the universe is made of. To reach this goal the elementary constituents of matter and the interactions between them are studied.

The *Large Hadron Collider* (LHC) is a particle accelerator built at CERN that straddles the Franco-Swiss border near Geneva (Switzerland) and it is in operation since December 2009. The LHC occupies a tunnel of 27 kilometers in circumference, up to 175 meters below ground.

Two beams of particle bunches travelling in opposing directions are injected into the LHC from a series of smaller machines and accelerated close to the speed of light. The beams of protons or heavy ions are brought into collision at unprecedented energy at the centres of the four major LHC experiments: ATLAS [3], CMS [4], ALICE and LHCb [5].

During the last run, terminated in 2013, LHC accelerated particles to a centre of mass energy for proton-proton (pp) collisions of $\sqrt{s} = 8$ TeV, $\sqrt{s_{NN}} = 2.76$ TeV for

1. COMPUTING FOR HIGH ENERGY PHYSICS

lead-lead (Pb-Pb) collisions and $\sqrt{s} = 5.02$ TeV for proton-lead (pPb) collisions. Fig.1.1 shows the system of accelerators that inject ions and protons into the LHC.

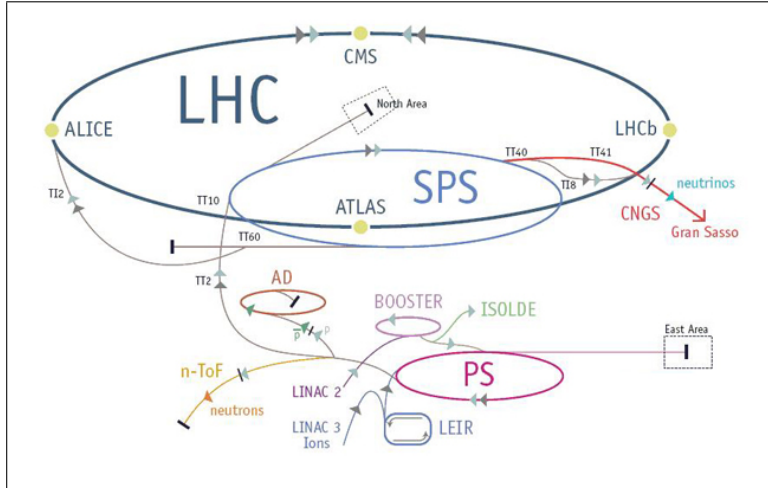


Figure 1.1: Each machine of the CERN accelerator system (Booster, PS, SPS, LHC) injects the beam into the next one progressively increasing the beam energy. The last element in the chain, is the LHC itself where the beam is accelerated up to the energy of 8 TeV. [6]

The main aim of the LHC is to help scientists to find answers to such questions as:

- *What is the origin of mass?*

The *Standard Model* gives a theoretical answer to this question. It assumes the existence of the *Higgs field* that, interacting with particles, gives them their mass.

Developed in the early 1970s, the model describes the process according to which everything in the universe is made starting from fundamental particles, governed by four forces: the strong force, the weak force, the electromagnetic force and the gravitational force.

The Standard Model defines three major groups of particles: *quarks*, *leptons* and *gauge bosons* (See Fig. 1.2). These last ones mediate the forces.

Each group consists of six particles, which are related in pairs called *generations*. The first generation represents the lightest and most stable particles, while the second and third generations include the heavier and less stable particles. The six leptons are the *electron*, *muon*, *tau* and their corresponding *neutrinos*.

The three different generations of *Quarks* are includes the *up* and *down* quark,

1.1 LHC and the ALICE experiment at CERN

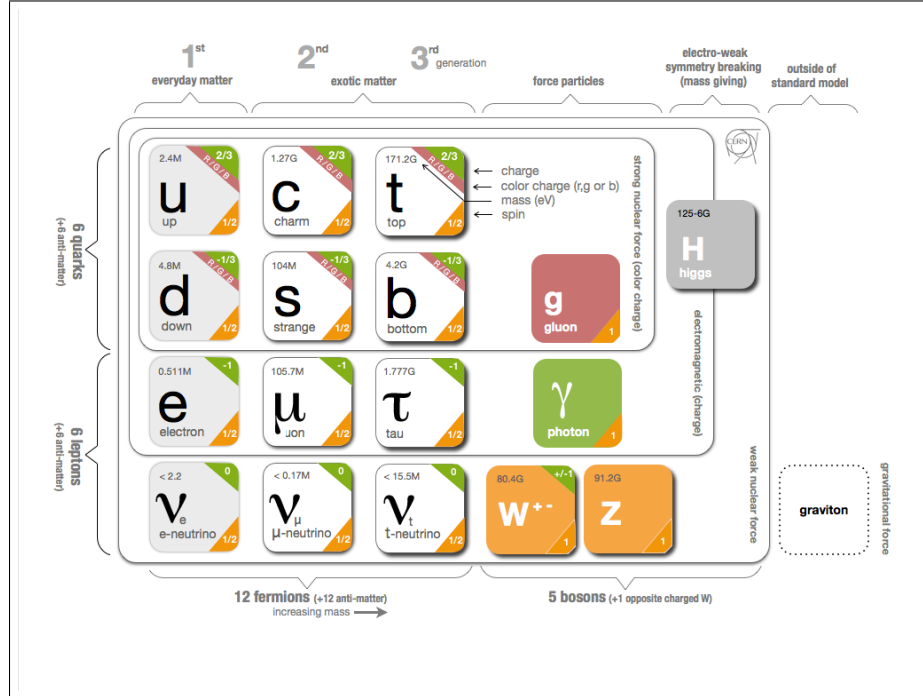


Figure 1.2: The elementary particles in the Standard Model.

which, building up neutrons and protons, constitute ordinary matter. The second and third generation include *charm* and *strange* quarks and *top* and *bottom* quarks respectively.

Matter particles transfer discrete amounts of energy by exchanging gauge *bosons* with each other. Three of the four fundamental forces are carried by a corresponding boson: the electromagnetic force is carried by the *photon*, the strong force is carried by the *gluon* and the weak force is carried by the W and Z bosons. The fourth fundamental force, gravity, does not fit in the Standard Model, but at the minuscule scale of particles its effect is fully negligible.

An essential component of the Standard Model is a particle called the *Higgs boson* which explains which is the boson of the field through which particles acquire their mass.

The mass of all particles depend on their interaction with the *Higgs field*. Thus, for example, *Electrons* scarcely interact with the field, acquiring little mass. The *Quarks* that make up *protons* and *neutrons* interact strongly with the field so they have more mass. W and Z bosons plunge in the *Higgs field* therefore they have thousands of time

1. COMPUTING FOR HIGH ENERGY PHYSICS

more mass. *Photons* and *Gluons* are massless because they do not interact with the field.

In 1964 two papers, published independently by Belgian physicists Robert Brout and Francois Englert and by British physicist Peter Higgs, proposed the *Brout-Englert-Higgs* (BEH) mechanism as the origin of the masses of fundamental particles.

A key prediction of the model is the existence of a new type of *boson*, the discovery of which was announced by the ATLAS and CMS experiments at CERN on 4 July 2012.

On 8 October 2013 Prof. Francois Englert and Prof. Peter W. Higgs were awarded the Nobel Prize in physics for the discovery of the BEH mechanism, which was confirmed by the detection of the *Higgs Boson* by LHC experiments. One of the main tasks of the LHC has been allowing scientists to discover the Higgs.

Although the Standard Model describes well the fundamental particles of matter, there are other important questions that it does not answer, such as:

- *Why is there more matter than antimatter in the universe?*

Antimatter particles have the same mass as matter particles but opposite charges. If a pair of matter and antimatter particles come into contact with each other, they annihilate, producing pure energy.

It is commonly supposed that pairs of particle-antiparticle were produced during the first fraction of a second of the big bang. An unknown mechanism must have interfered with the particles avoiding the destruction of all the matter-antimatter pairs so that a tiny portion of matter managed to survive.

High-energy proton collisions at the Large Hadron Collider lead to the creation of matter and antimatter particles. Physicists try to obtain more information on the relations between matter and anti-matter by studying the behaviour of these particles.

- *What is dark matter?*

At the moment, the particles that we know account for only 4 % of the universe. The rest is believed to be made up of *dark matter*.

Several theories attempt to explain what the dark matter is made of. One of these is the *Supersymmetry theory*, an extension of the Standard Model, that predicts a partner

particle for each fundamental particle to help explain why particles have mass.

Following this theory, the dark matter could contain *supersymmetric particles* and this is one of the things the LHC experiments are looking for.

Physicists try to answer to all these questions by means of the LHC experiments. ATLAS and CMS are the largest. They were mainly designed to look for the Higgs boson, however they can be used to study all kinds of physics phenomena at the LHC energy range. The ALICE detector is a dedicated heavy ion detector to study the properties of the *Quark Gluon Plasma* formed in the collisions of lead ions. LHCb has been specifically designed to study the asymmetry between matter and antimatter.

1.1.1 The ALICE Experiment

ALICE (A Large Ion Collider Experiment) is a dedicated heavy-ion detector designed to study the physics of strongly interacting matter at extreme energy densities. At such densities a state of matter called *Quark-gluon plasma* (QGP) has been predicted to exist. Physicists in ALICE aim to study characteristics of the QGP arising from nucleus-nucleus collisions.

The study of QGP helps to understand the *confinement principle* that reflect the experimental fact that *quarks* are never been observed in isolation, instead they are bound together by mean of the strong interaction force mediate by the exchange of *gluons* and confined inside composite particles such as, e.g., a *proton* or a *neutron*.

According to the *Quantum Chromo-Dynamics* theory (QCD), *quarks* and *gluons* should no longer be confined inside composite particles when very high temperatures (around 2000 billion degrees) and very high densities are reached. Matter in this state is known as *Quark-gluon plasma* and can be formed when heavy nuclei collide at almost the speed of the light. In this conditions the strong interaction force becomes very small (asymptotic freedom). To characterise and understand the behaviour of QGP, ALICE is focused on the study of *hadrons*, *electrons*, *muons*, and *photons* produced in the collision of heavy nuclei (PbPb).

ALICE is also studying proton-proton and proton-nucleus collisions to collect reference data for the heavy-ion programme and to address several QCD topics for which ALICE is complementary to the other LHC detectors. The ALICE collaboration is composed of 36 countries, 131 institutes and has more than 1200 members.

The ALICE detector (See Fig. 1.3) has a weight of 10.000 tons, is 26 meter long,

1. COMPUTING FOR HIGH ENERGY PHYSICS

16 meters high, and 16 meters wide and it is build in a cavern situated 56 meters below ground in France.

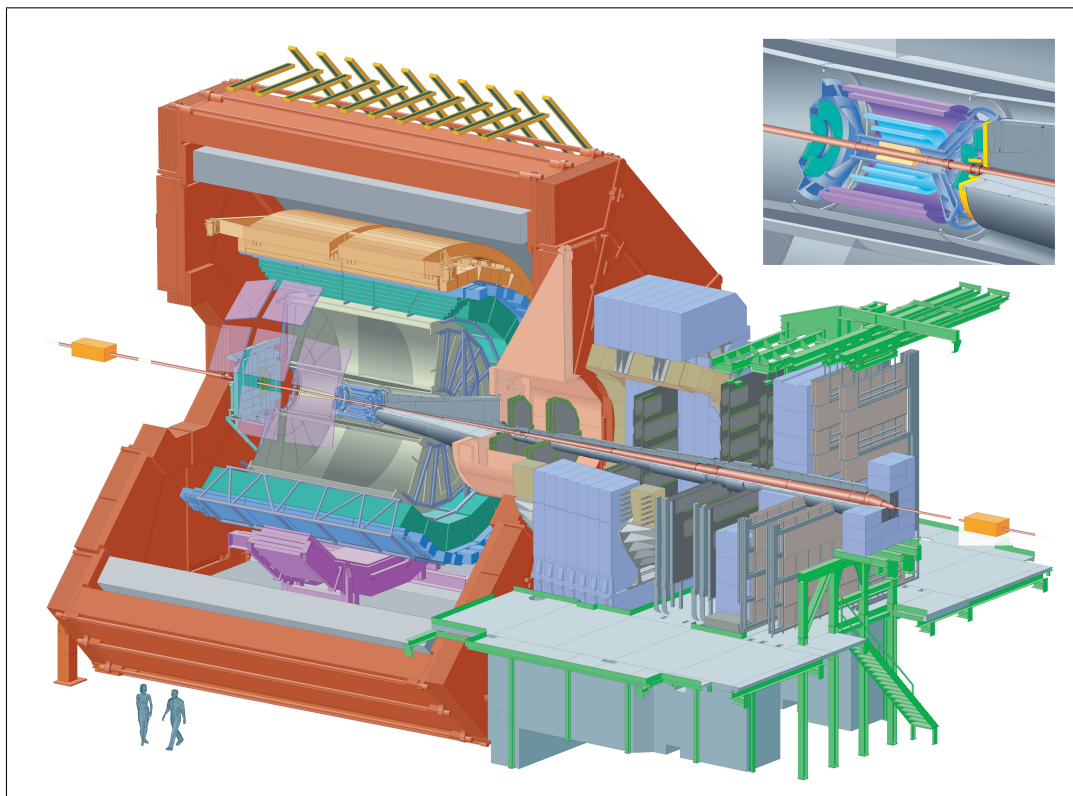


Figure 1.3: A schematic view of the ALICE detector.

1.2 HEP computing

Physicists from all over the world are analysing data from the LHC experiments. The analysis involves reading and mining large volumes of data with algorithms tuned to extract specific physics signals.

Analysing physics data is done via a sequence of steps starting by processing raw detector data in order to identify and reconstruct physics objects such as charged particle tracks. The analysis includes the processing of data generated via Monte Carlo simulations to predict the detector response and to calculate the efficiency of observing physics processes.

All these tasks are data and computing intensive. Across the full scope of the HEP

experimental programs, scientists require large-scale computing resources that exceed the computing power of single institutes or even large data centres.

Therefore, part of the computing needs are met by dedicated high-energy physics (HEP) computing centres like the one located at the CERN laboratory but another substantial part of the computing demand relies on institutes and universities that are members of the experiment collaboration. In the case of ALICE, at present around 85 centres contribute to computing resources.

A Grid architecture was identified as a solution to share computing and storage resources spread worldwide and to distribute physics data for reconstruction and analysis. During the past ten years, the Grid has given excellent results in terms of performance and quality of services.

However, in the last few years, the concept of Cloud computing is gaining ground and it starts to be adopted as new solution to meet the always growing demand for computing resources of the HEP experiments.

1.2.1 Grid computing

At the beginning of LHC, the volume of data to be analysed in order to discover new physics phenomena was already estimated to be about 15 PB per data taking year.

The number of the processor cores needed to process this amount of data was estimated to be about 200.000.

The answer to these requirements was found in a distributed computing and data management infrastructure called the Grid.

*“A **Computational Grid** is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities. It aims to enable coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations.”* [7] [8]

A **Virtual Organization** (VO) can be defined as a goal-oriented collection of multiple members who reside in different locations around the world and are working together sharing distributed computing resources. [9]

1. COMPUTING FOR HIGH ENERGY PHYSICS

In 2002, the *Worldwide LHC computing Grid* (WLCG) [10] was postulated to provide the production and analysis environments for the LHC experiments. It now integrates thousands of computers and storage systems in hundreds of data centres worldwide.

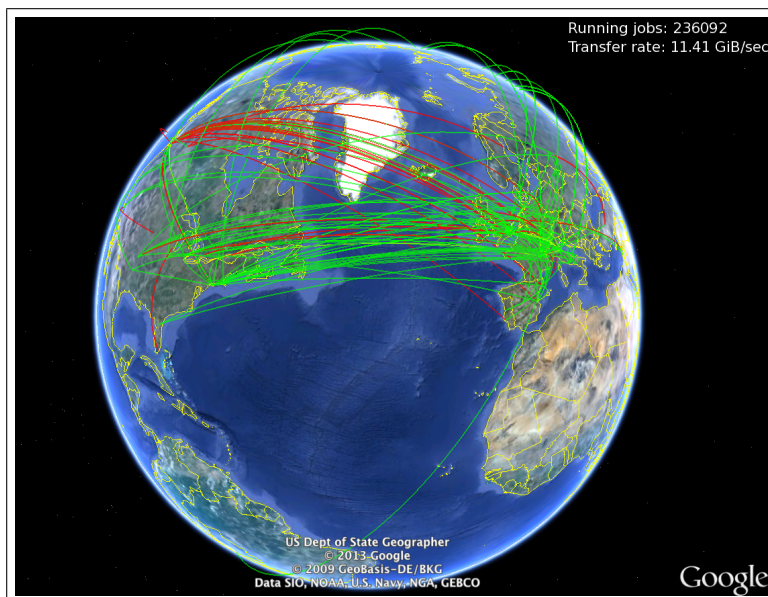


Figure 1.4: A view of the WLCG worldwide network in 2013.[11]

CERN itself provides about 20% of the resources needed to manage the LHC data. The rest is provided by the member states national computing centres and research network infrastructures supported by national funding agencies.

The WLCG allows the resource sharing between all the physicists participating in the LHC experiments, which is the basic concept of a Computing Grid. The Grid computing and the WLCG will be discussed in Chapter 2.

1.2.2 Cloud computing in High Energy Physics

During the last years, Cloud computing has been widely considered as a possible solution to provide a more flexible and on-demand computing infrastructure than the Grid. The Cloud can be defined as:

A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power,

storage, platforms, and services are delivered on demand to external customers over the Internet. [12]

Although Grid computing has proven to be a good solution for managing distributed resources and allows for the treatment of large scale computational problems, this approach still has some limitations.

One of the limitations of the Grid is the dynamic management of resources sharing. For example, an available resource may not satisfy the requirements of a virtual organization, therefore it can be underused. In the other hand, a specific resource may not be available at the time it is needed, leaving the user to wait or to seek an alternative. To overcome these limitations a new technology named Cloud computing has been evolved out of Grid computing.

While the Grid computing infrastructure provides computing and storage resources, the Cloud computing infrastructure delivers more abstract resources and services following a more economy based model.

Cloud Computing can be viewed as platforms providing easy access to applications, storage and computing resources through web services. It offers a way to increase capacity or add capabilities on the fly without investing in new infrastructure or in new software through the concept of Virtualisation. Virtualisation means to create a virtual version of a device or resource, such as a server, storage device, network or an operating system where the framework divides the resource into one or more execution environments. The virtualisation of the operating system allows to run multiple operating systems simultaneously on a single machine.

As for the Grid, one of the aims of Cloud computing is to hide the complexity of underlying resources from its users. In addition, Cloud computing platforms provide reliability, very good scalability, high performance, and settable configurability.

Virtualisation is widely used by Cloud architecture to create an environment independent of the underlying resources. It can be applied to memory, networks, storage, hardware, operating systems, and applications.

Virtualisation makes possible the running of multiple applications and operating systems in a single physical system by using partitioning. Virtual machines can be dynamically configured before deployment leading to the reduction of inefficient resource

1. COMPUTING FOR HIGH ENERGY PHYSICS

allocation and of excessive overheads.

The Cloud architecture delivers services at three different levels to users:

- *Software as a Service* (SaaS): applications that are remotely accessible by users through the Internet.
- *Hardware/Infrastructure as a Service* (H/IaaS): services providing data storage and computer processing to outside users.
- *Platform as a Service* (PaaS): services providing remote computing capacity along with a set of software-development tools to build, test and deploy users applications.

The HEP community has started to profit of the benefit from the IaaS Cloud computing that allows the virtual delivery of computing resources in the form of hardware, networking, and storage services.

The *CERN Private Cloud* [13] provides an IaaS solution integrated with CERN's computing facilities. Using self-service portals or Cloud interfaces, users can rapidly request virtual machines for production, test and development purposes. The IaaS Cloud computing has been adopted at CERN because it allows a more efficient use of hardware by enabling the remote management for new data centres and by consolidating the support to a single service.

However, the Cloud computing approach is not devoid of limitations. A network must be set-up, the use of virtualisation makes difficult the correct monitoring of resources because the lower level resources are rendered opaque to the user. The limited information available may make it challenging to figure out what the resource status is.

Another challenging problem of the Cloud is security. The Cloud user should make sure that sensitive data will only be accessible to privileged users. The Cloud provider has external audits and security certifications and it should also have an efficient replication and recovery mechanism to restore data in case of problems.

The Grid business model is a one-time payment of a high cost for the full availability of the resource to the VO for a certain amount of time until a new resource has to be purchased. On the other hand, using Cloud computing, it is possible to purchase dynamically a resource or a service for a short time and provided at a relatively low

cost compared to dedicated infrastructures. However such Cloud costs are incurred each time the resource is used. Since 2008, Cloud computing has been taking its first steps forward also at CERN. An example is the *CernVM project* [14]. Cern VM (See Fig.1.5) is a virtual machine designed to support the LHC physics computing on several hypervisors and platforms such as end-user laptops, Grid and cluster nodes, volunteer PCs running BOINC [15], and nodes on the Amazon Elastic Compute Cloud (EC2) [16].

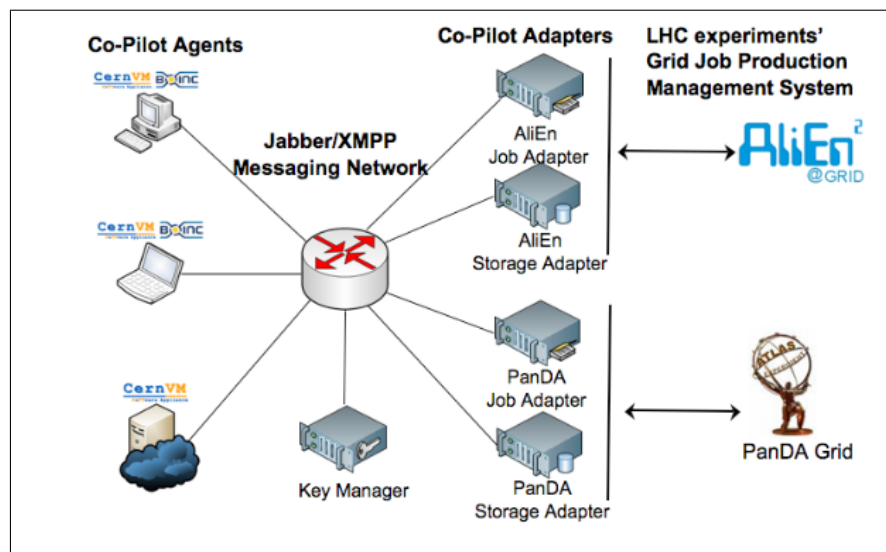


Figure 1.5: The CernVM Co-Pilot architecture: an interface to extra computing resources as dynamically configurable clouds.

Another LHC Cloud project is the *CERN LXCloud pilot cluster* [17] that can be used as a Cloud interface to the underlying Grid or to access other public or commercial Clouds.

Although a significant part of LHC physics computing can benefit from the use of the Cloud, the rest is not suitable for Cloud operation. This is the case for I/O intensive operations or physics analysis that need widely distributed datasets.

However, since the performance and monetary cost-benefits for scientific applications are not yet clear, the adoption of Cloud computing solutions by the scientific community is proceeding slowly.

Nevertheless the strategy towards deployment over the WLCG resources of Cloud interfaces, managed with high level of virtualisation, is under evaluation to better un-

derstand how to implement a performing Cloud architecture and what are the costs involved.

1.3 Data analysis technique

Traditionally, a physics analysis job uses a small fraction of the production results as input and then performs selections and other data manipulations to get a conclusion about the existence or properties of a physics process.

The fundamentals of high energy physics analysis are the pre-selection of events from the data storage, the analysis itself performed by means of programs aimed to single out event properties and the presentation of histograms, graphs, numbers and tables from the analysis.

The end-user analysis activities can be classified in three categories, according to the degree of interactivity as a function of the response time:

- Interactive tasks which are usually performed on private laptops and require very short response times. Examples are the browsing of histograms, the production of final fits and tasks related to the visualization processes.
- Operations that require sporadic tuning or optimization such as batch Monte Carlo production, reconstruction and format reduction. These often have long execution times and require very little interactivity.
- Interactive-batch operations such as prototyping of selection or reduction algorithms, requiring repeated refinement cycles on large data samples. They require different level of interactivity.

At CERN, interactive tasks run on local desktops or laptops, using experiment-specific applications such as event display or pure ROOT [18], while batch tasks run on dedicated batch farms or on the Grid, using the experiments submission interfaces or general purpose interfaces like Condor [19].

For interactivity on batch resources there is not a common solution. Often, solutions are implemented by each experiment according to their particular Grid or batch system.

1.3.1 Interactive data Analysis

The aim of interactive data analysis is to extract information through a process where the user issues a query, receives a response and formulates the next query based on the response. Often, the level of interactivity available is a limit for the size of the input dataset.

Several interactive analysis environments have been developed inside and outside the context of HEP. Examples within HEP are the *Physics Analysis Workstation (PAW)* [20], ROOT and *Java Analysis Studio (JAS)* [21]. Experiments wishing to use these analysis frameworks have to store summary data in the format defined by these systems in order to provide an easy access to the data.

ROOT is the data analysis framework developed and used at CERN in the context of HEP experiments; it allows interactive analysis by using three methods:

- C++ code linked against ROOT libraries, a method suitable for large analysis programs;
- Using the ROOT GUI which is a quick way to display and manipulate histograms;
- Through macros using the CINT (C++ interpreter) which evaluates C++ code inside ROOT;

An extension of ROOT, developed to perform parallel analysis as an interactive alternative to batch systems for Central Analysis Facilities and departmental workgroups is the PROOF framework [22]. This is explained in Chapter 3.

PROOF is used for interactive parallel execution of independent data analysis tasks on a set of PROOF enabled servers. It provides real-time feedback, therefore the user can define a subset of the output objects to be sent back at a tunable frequency for checking. In addition, PROOF provides also an interactive-batch mode which allows to run long jobs in asynchronous mode. In this mode, the client can disconnect from the PROOF master with the PROOF system continuing to process its task. The result can be retrieved from the master at any time.

1.3.2 Batch data Analysis

In a traditional batch data analysis system, the user issues queries, the system processes them asynchronously, and then it returns the result.

1. COMPUTING FOR HIGH ENERGY PHYSICS

Physics data are organized into events that can be processed independently on different computing nodes and even on different sites of a batch cluster or a Grid. Therefore, the intrinsic event parallelism is exploited by splitting the large queries into smaller pieces which are run in parallel and whose outputs are merged at the end. This mechanism reduces significantly the execution time. In addition, the implementation of a query-level parallelism lets the same code run locally, on the batch system, and on the Grid without modification.

The response time is improved by distributing the processing of large datasets, both because multiple processors work on the problem at the same time and, since the data are distributed, there is the possibility of sending the process to the data rather than moving the data.

The overall execution time is determined by the execution time of the slowest query-job. Therefore, it may be subject to long tails, since there is no mechanism to automatically redistribute the work of a slow worker.

Furthermore real-time feedback about the status of processing would require special instrumentation to interface with a monitoring system.

The *CERN batch computing service* currently consists of around 30.000 CPU cores running *Load Sharing Facility* (LSF) [23]. It provides computing power to the CERN experiments for tasks such as physics event reconstruction, data analysis and physics simulations.

CERN computing resources are exposed to users through a local batch system and a Grid share. The local batch system can be used by users with local accounts to submit jobs for local processing. The batch resources are provided by the general WLCG Grid via the Grid share, they represent around two thirds of CERN computing resources and all users of the Grid can access these resources via their experiment job submission frameworks.

For user-analysis, the majority of WLCG experiments prefer Grid job submission systems to local CERN job submission systems because the Grid provides a far faster turnaround than the local batch solution.

More precisely, ALICE and LHCb have no local share in the CERN batch system, all their users use the Grid. ATLAS and CMS users have a very small share on the CERN batch system. However, all their users are recommended to use the Grid.

Although most of the work done on computing Grids uses a batch-oriented mode of operation, the extension to an interactive use is very attractive.

1.3.3 Interactivity on batch resources on the Grid and the Cloud: status of the field

During the last few years several solutions to provide interactivity on batch resources, Grid and Cloud have been proposed.

On the Grid, one solution is the creation of a special *Globus job-manager* [24] with interactive response. It provides event based processing of dynamic data created while a Grid job is running.

Another solution to provide interactive applications in Grid environments is the use of virtual machines in order to run applications [25]. This approach does not treat the problem of virtual machine allocation.

IGENV [26] is an architecture for the execution of interactive applications. It allows the creation of interactive sessions on remote resources. However, the resource scheduling problem is not considered by this proposal.

The *CrossGrid* [27] project allows the execution of parallel applications compiled with the *MPICH library* [28] on Grid resources in a transparent and automatic way.

Three main products have been developed in the context of the *CrossGrid* project. The *Migrating Desktop* [29] is a user-friendly environment that provides a method of accessing Grid resources from any web-enabled computer from anywhere on the Internet. It uses a Java-based GUI designed specifically for mobile users, and it is platform independent.

The *CrossBroker job manager* [30] provides services as part of the *CrossGrid* middleware and it allows the execution of parallel Message Passing Interface (MPI) applications on Grid resources in a transparent and automatic way.

The *Interactive European Grid (i2g)* [31] project provides a production quality e-infrastructure, interoperable with those using gLite [32], enabling advanced support for interactivity, parallel execution and graphical visualization.

The project has exploited and consolidated the main *CrossGrid* achievements, such as the *CrossBroker* and the *Migrating Desktop*, to manage both interactive and batch jobs of both sequential and parallel types.

New developments from the *i2g project* are *mpi-start* [33] and *i2glogin* [34].

1. COMPUTING FOR HIGH ENERGY PHYSICS

Mpi-start is a software layer that hides all the heterogeneity inherent to Grid hardware and software set-up (file system, MPI implementation, etc.). It allows for flexible and transparent use of MPI parallel applications.

The *i2glogin* middleware allows fully interactive connections to the Grid, with functionality comparable to that of SSH, but without the overhead of a server side running on top of every Grid node.

1.3.3.1 Interactivity through the PROOF framework

Different approaches allow the integration on the PROOF framework in a Grid Environment, on a batch system or on the Cloud.

HEP experiments usually adopt ad-hoc solutions for PROOF use on Grid or batch resources according to their Grid middleware or batch system. A common solution has been provided integrating PROOF on the Cloud.

A mechanism to integrate PROOF with the local batch systems of small institutes of the CMS experiment [35] provides transparent access to a PROOF cluster. It allows local users to profit from an interactive queue through which individual PROOF clusters can be dynamically enabled.

Another ad-hoc solution [36] provides PROOF on batch resources by enabling PROOF cluster on the *Oracle Grid Engine* (OGE) batch system [37].

Tools like *Proof on Demand*, which is covered in Chapter 3, allow PROOF to be set-up on batch resources.

A framework able to deploy the PROOF infrastructure [38] uses different platforms such as PROOF-Lite or batch systems using PoD and Cloud starting virtual machines on demand.

A dynamic PROOF-based Cloud analysis facility based on *PoD* and *OpenNebula*, managed by a control daemon that makes virtualisation transparent for the user provides interactivity on Cloud resources [39]. *OpenNebula* [40] is an open source *IaaS* (Infrastructure as a Service) framework to set-up, administrate and monitor public and private clouds. *OpenNebula* can dynamically provide interactive PROOF resources even on small-sized computing facilities.

Finally, a *Virtual PROOF-based Analysis Facility* [41] has been provided at CERN. This is a cluster appliance combining the CernVM software and PoD for the set-up of dynamic PROOF clusters on the Cloud.

1.3 Data analysis technique

This thesis presents a prototype to provide interactivity on the ALICE Grid by integrating *Proof on Demand* in the *AliEn* system.

1. COMPUTING FOR HIGH ENERGY PHYSICS

The Grid Technology and the ALICE Environment: AliEn

This chapter will highlight the basic concept of the Grid architecture, using the ALICE Grid as the main example. A detailed description of the AliEn middleware will be given.

2.1 The Grid Paradigm

The LHC experiments vast computing, storage and network requirements have created a new computational reality, which could not be supported by the standard high performance computing centre layout and relative isolation.

A single organization is not able anymore to provide enough resources to suit the growing computational needs. In addition, the research institutions needed to handle dynamically the frequent changes of workloads and to be flexible to provide computing power where it is needed most. Many scientific, engineering and business problems can not be managed using the existing high performance computers.

A good answer to the computing power demand and to the necessity to use and manage heterogeneous resources has been found in the *Grid Computing* concept. The main purpose of the Grid is to build a computing infrastructure that provides reliable, consistent and inexpensive access to computational resources.

Through the concept of *Virtual Organization*, a grid provides a single point of access to distributed and heterogeneous resources.

2. THE GRID TECHNOLOGY AND THE ALICE ENVIRONMENT: ALIEN

As well as researchers who are using the power of the Grid computing to work on many of the most difficult scientific tasks, the business community is recognizing the utility and the power of distributed systems applied to problems such as data mining and economic modelling.

Using a Grid, the users can use and share networked, storage and computational distributed resources in a scalable and efficient way, combining them to increase the computing power when and where it is needed.

A really important example of the use of a Grid to share scientific instruments is the CERN WLCG , which we will discuss in the next section. WLCG allows physicists from all over the world to analyse and manage data coming from the experiments performed with the *Large Hadron Collider* (LHC) particle accelerator.

2.1.1 Grid Architecture

The typical Grid architecture consists of four layers (Fig. 2.1) :

- *Fabric layer*: it provides distributed resources such as computational resources, storage media and networks. The fabric layer implements the local operations on specific resources that occur as a result of sharing operations at higher levels.
- *Resource and connectivity layer*: it provides communication protocols to exchange data between fabric layer resources, authentication protocols to verify the identity of users and resources, remote resource management, co-allocation of resources and storage access. These services hide the complexity and heterogeneity of the fabric level.
- *Collective layer*: it contains services and protocols not resource-specific and provides higher level abstractions and services. These include application development environments and Grid-enabled programming tools.
- *Grid applications and portals layer*: it consists of user applications that require access to remote datasets, computational power, management of huge amounts of data and may need to interact with scientific instruments. Grid portals provide web applications to allow users to submit and collect results for their jobs on remote resources through the Web.

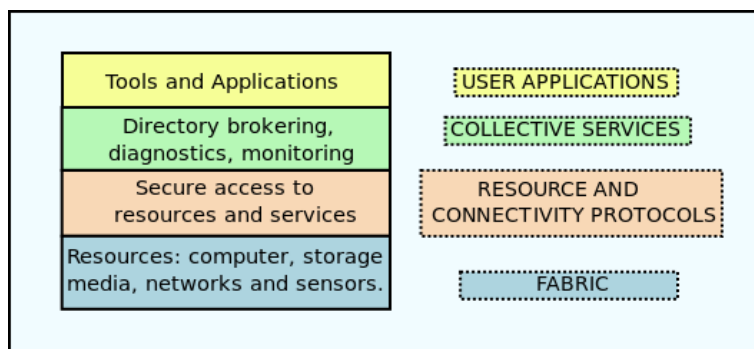


Figure 2.1: The layered Grid Architecture.

The *middleware system* [42] can be defined as an interface between the user and the heterogeneity of the Grid distributed resources owned by different organizations.

The aim of the middleware systems is to provide an infrastructure where users benefit from distributed computational resources, without knowing the underlying system architecture. In addition to the management of heterogeneous distributed resources, the middleware should also ensure scalability to avoid performance degradation as the size of Grids increases. A middleware system must choose the available resources dynamically and use them in an efficient way.

An early example of Grid middleware is the *Globus toolkit* [43]. This is an open source software toolkit whose purpose is to support the building of computational Grids. Figure 2.2 shows the structure of Globus toolkit 5 which provides components for basic runtime and execution management, data management and security services.

The resource management tools are represented by the *Grid Resource Allocation and Management* (GRAM) interface in charge of the management and scheduling of remote computations. In the version 5 of the *Globus Toolkit* the *Monitoring and Discovery Service*, used for publishing and querying of resources information, is part of the GRAM.

The data management tools are referred to the location, transfer, and management of distributed data. GT5 provides various basic tools, including *GridFTP* for high-performance and reliable data transport and the *replica location service* (RLS) for maintaining location information for replicated files.

The security tools provide methods for authentication of users and services, protecting communications, authorization, as well as with supporting functions such as

2. THE GRID TECHNOLOGY AND THE ALICE ENVIRONMENT: ALIEN

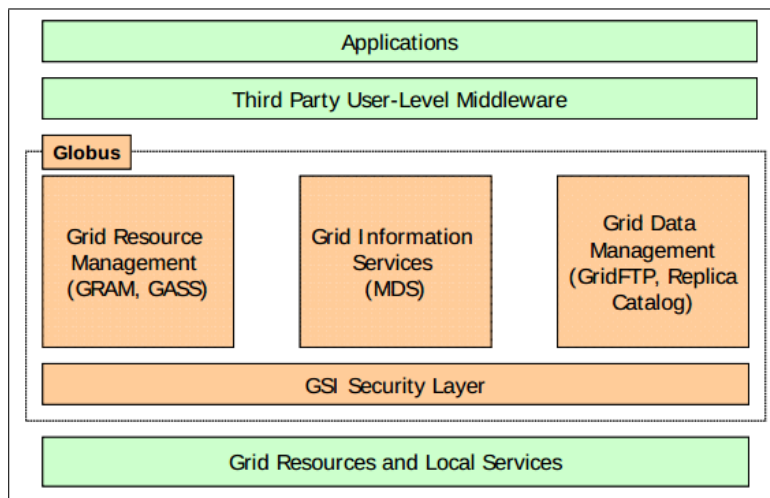


Figure 2.2: The Globus Architecture.

managing user credentials and maintaining group membership information. They are based on *Secure Sockets Layer (SSL)*, *Public Key Infrastructure (PKI)* and *X.509 Certificate Architecture*. The *MyProxy* service is an online credential repository used to store X.509 Proxy credentials, protected by a passphrase, for later retrieval over the network. *GSI-OpenSSH* is a modified version of OpenSSH that supports for X.509 Proxy certificate authentication and delegation, providing a single sign-on remote login and file transfer service.

On the top of the main services, third party user-level middleware can be created by developing applications.

2.1.2 The Worldwide LHC Computing Grid

The requirement to manage large amounts of data originating from the LHC experiments, to process and analyse them is one of the main reasons for the development of the Grid.

The WLCG project, started in 2002, is today the largest existing Grid infrastructure and involves more than 250 sites spread over 45 countries on 5 continents. The mission of the WLCG is to provide a global access to distributed computing resources in order to store, distribute and analyse the huge amount of data annually generated by the LHC at CERN.

2.1 The Grid Paradigm

The structure of WLCG is hierarchical, each site is classified into “*Tiers*” according to the level of services and resources provided.

The first model of a potential distributed computing system for LHC, created in 1999, was the so-called *MONARC model* [44]. Figure 2.3 shows the tiers of the model, and the interaction between them.

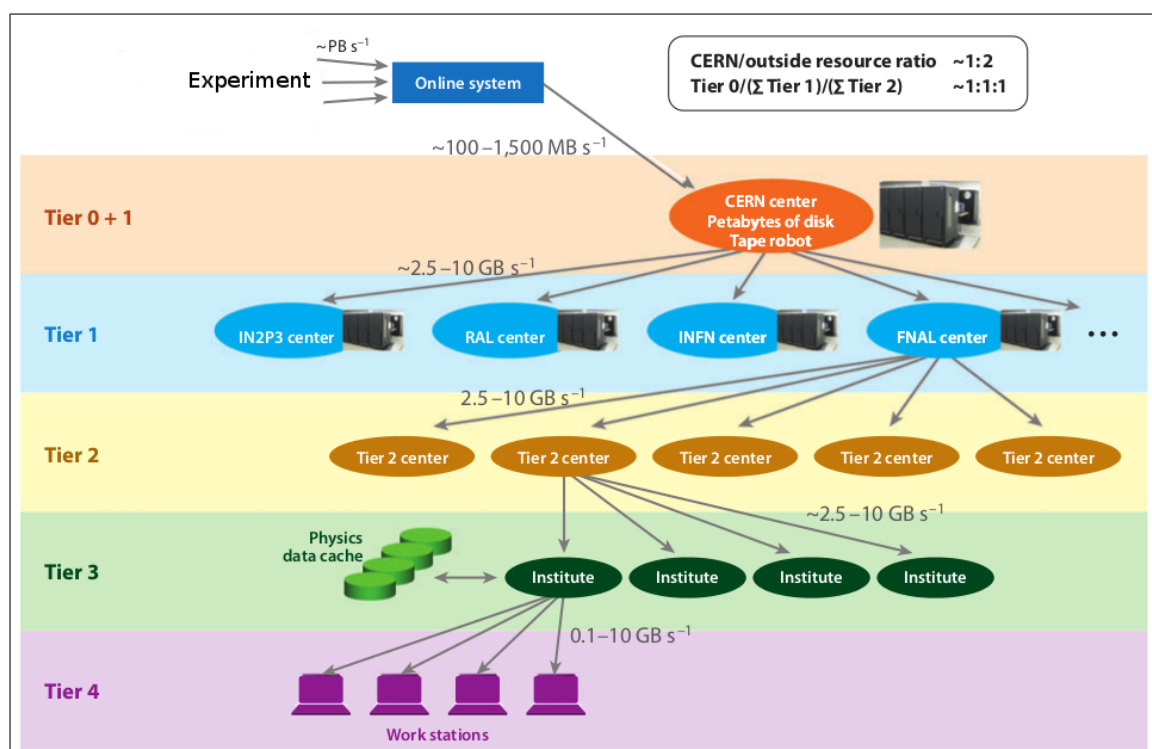


Figure 2.3: The MONARC model: the original concept for distributed LHC computing in 1999.

The *Tier-0* site is CERN, where the raw data from LHC are stored for the first time on the tape system called *CERN Advanced STORAGE manager (CASTOR)*, here the first level of processing is performed. The data are also replicated and re-processed over several *Tier-1* centres which are large computer centres in Europe, the USA, Canada, Korea and Taiwan, all connected to CERN through dedicated network links of at least 10 Gbit/s. Then there are currently about 140 *Tier-2* spread worldwide, they process Monte Carlo simulations of the collision events in the LHC detectors and perform end-user analysis jobs. The last tiers are the *Tier-3* and *Tier-4*, which are small universities or research institutes computing clusters and work stations.

2. THE GRID TECHNOLOGY AND THE ALICE ENVIRONMENT: ALIEN

A fast and reliable connection between tiers and end-users is made possible by the WLCG networking. It relies on the *Optical Private Network* (OPN) backbone which provides dedicated connections with the capacity of 10 Gbit/s between CERN *Tier-0* and each of the *Tier-1s*.

As stated above, a middleware suite is a set of protocols and services used to make accessible and usable resources distributed all over the world to the LHC experiments in a transparent way. Figure 2.4 shows the middleware of WLCG that is a collection of services including:

- Data Management Services.
- Security Services.
- Job Management Services.
- Information Services.

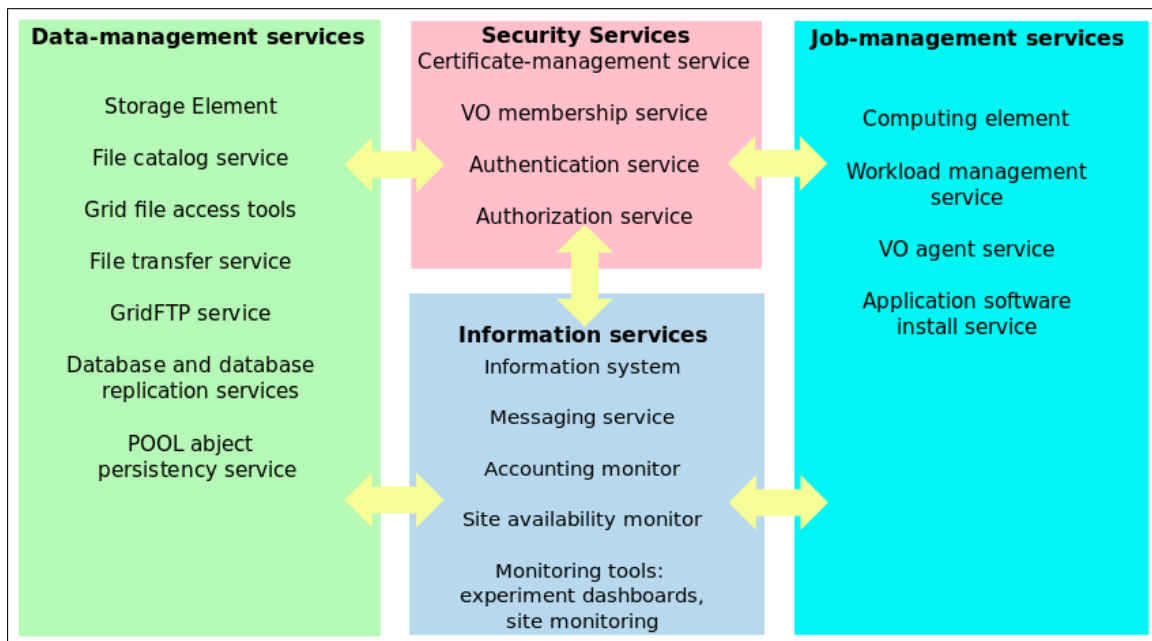


Figure 2.4: Grid middleware components shown in terms of their functionality.

The WLCG middleware has been built using and developing some packages produced by other projects including:

- *EMI (European Middleware Initiative)* [45] : combining the key middleware providers of ARC, gLite, UNICORE and dCache.
- *Globus Toolkit*: developed by the Globus Alliance.
- *OMII* [46] : from the Open Middleware Infrastructure Institute.
- *Virtual Data Toolkit* [47] .

In addition to the standard Grid middleware, the LHC experiments have developed their own specific components to better meet the demands of the experiment-specific computing models.

The ALICE experiment has developed a Grid middleware called *ALICE Environment* (AliEn) [48] , which is an interface between ALICE users and the distributed resources with the aim to provide a transparent access to the complex underlying distributed system.

In addition to selected services of WLCG middleware such as *CREAM-CEs* and *WLCG-VOBox*, AliEn provides other services in order to manage and process the data produced by the ALICE experiment.

The *Computing Resource Execution And Management* (CREAM) CE [49] is a lightweight service for job management operations at the *Computing Element* (CE) level. CREAM is an interface between AliEn and the local batch system or another Grid system. Developed for the gLite middleware (now part of EMI) and adopted by the WLCG, CREAM is a job management component that can be used to submit, cancel, and monitor jobs for execution on suitable computational resources.

The *VOBox* [50] is the entry point of ALICE to the WLCG environment. Its aim is to meet the experiment's requirement for an environment to run services (*CE*, *PackMan*, *Monitor*, *CMReport*) providing a Proxy renewal mechanism and the automatic refreshing of the user proxies registered on the *VOBox*. *VOBox* services requiring certificate authentication are always provided with valid credentials and direct access to the shared software area, with access restricted to the *VOBox* administrator. Other LCG-developed services used by ALICE are *Disk Pool Manager (DPM)* [51], *CASTOR* [52] and *dCache* [53] storage elements (SEs).

The ALICE-specific services of AliEn (*packman*, *CMReport*, *MonAlisa*, *central Task Queue* etc) are explained in the next sections.

2. THE GRID TECHNOLOGY AND THE ALICE ENVIRONMENT: ALIEN

2.2 ALICE computing model

In line with the other LHC experiments, ALICE is using a multi-tier computing model [54] to manage computational and storage resources. Following the so-called *MONARC model*, the resources are provided by CERN (*Tier-0*), six *Tier-1* centres (GSDC, RAL, NDGF, CNAF, CC-IN2P3, GridKA) and regional *Tier-2* centres.

However, given the good performance of the network and considering the data placement issues, the tiered structure is dissolving and the *MONARC model* is evolving into a more flexible architecture.

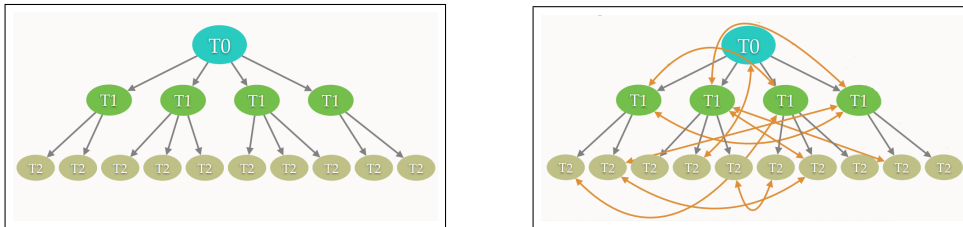


Figure 2.5: a) The original MONARC model; B) The current ALICE model with a more democratic data flow allowed sharing the data flow between centres.

Figure 2.6 shows the ALICE *Tier-0* which stores the raw data coming from the detector, performs the first pass reconstruction resulting in the *Event Summary Data* (ESD) and stores calibration data, ESD and *Analysis Object Data* (AOD) derived from the ESD. A copy of the RAW data is also stored in the ALICE *Tier-1* centres, that are the major computing centres with mass storage capability. Consecutive reconstruction passes are performed at the *Tier-1* centres and the ESDs and AODs are replicated. The ALICE *Tier-2* are smaller regional computing centres where simulation and user analysis are performed.

2.3 The ALICE Grid environment

AliEn is a Grid middleware suite developed by the *ALICE Offline Project*, it enables the full offline computational work-flow of the experiment, simulation, reconstruction and data analysis in a distributed and heterogeneous computing environment.

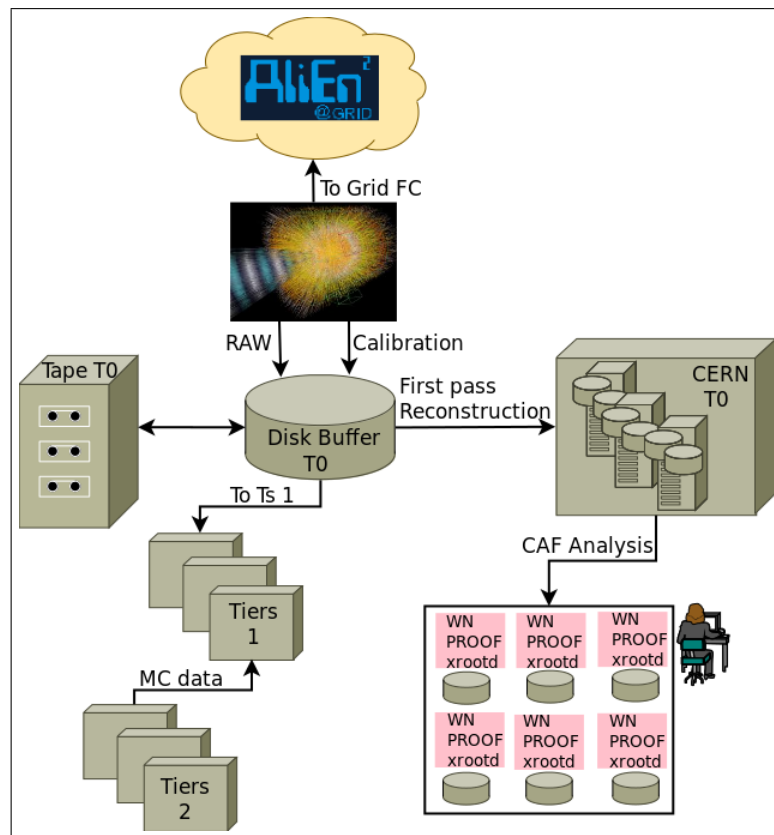


Figure 2.6: The ALICE data flow.

2. THE GRID TECHNOLOGY AND THE ALICE ENVIRONMENT: ALIEN

Alien has been developed since the year 2000 for distributed Monte Carlo productions. Each year more functionality has been added to the system and since 2005, AliEn has been used also for data analysis.

AliEn has been using standard network protocols and Open Source components based on the web services concept. All the communications between AliEn web services are handled using the *Simple Object Access Protocol* (SOAP) [55]. The basic parts of AliEn can be grouped into three categories: external software components, AliEn core components and services and user interfaces.

2.3.1 External software components

AliEn is using several external components, the most important are:

- Perl modules: the perl language was selected to build AliEn as a number of open source core modules were readily available. The entire AliEn core has been built using such modules. Examples are the crypto and SOAP client-server modules.
- *Lightweight Directory Access Protocol* (LDAP): it is used by AliEn in order to describe the configuration of VOs. By using LDAP, it is possible to define people, packages, sites and to configure services on remote sites.

2.3.2 AliEn core components and services

In the following sections a brief description of the basic components and services of AliEn is given. These components are:

- File and metadata catalogue.
- Workload Management System: job execution model, computing elements, information services and site services.
- Data management tools and services for data transfer, replication and storage.
- Authentication and Authorization.
- Monitoring.

2.3.2.1 File and Metadata Catalogue

One of the basic components of AliEn is the *File and Metadata Catalogue* [56]. The interface provided by the catalogue is a hierarchical structure of files and directories similar to the standard UNIX file system. In addition, it allows users to store metadata information to describe the files content.

The catalogue is implemented in a set of relational databases (MySQL). Scalability is achieved by splitting the logical directories and users into different tables and further onto separate databases on multiple hosts.

The catalogue annotates the physical files, which are contained in various storage elements across Grid sites and keeps only location and metadata information.

Figure 2.7 shows the internal catalogue structure. It is divided between *Logical File Name* (LFN) and *Grid Unique Identifier* (GUID) tables, which are kept in different and independent databases containing respectively the translation between LFN and GUID and the translation between GUID and PFN.

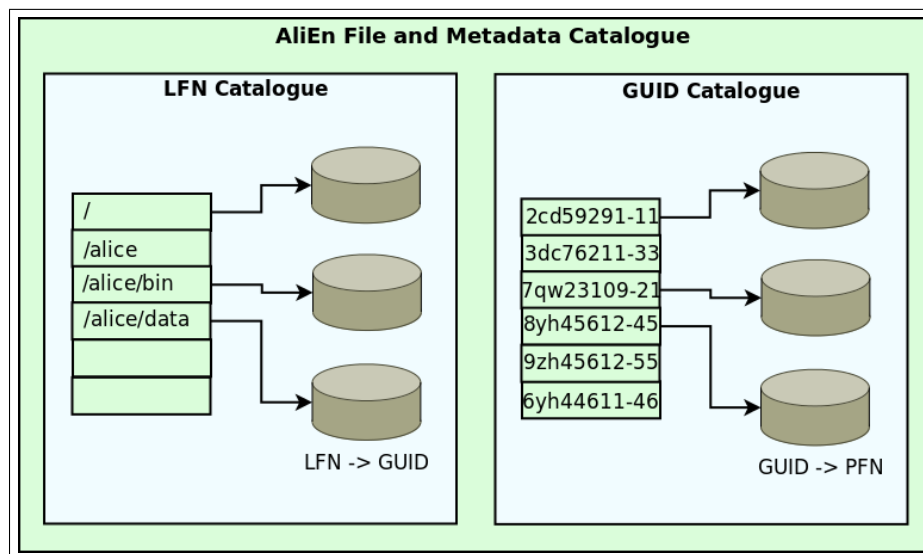


Figure 2.7: Structure of the AliEn File and Metadata Catalogue.

In order to obtain the PFN corresponding to a LFN, the look up in the index table of the LFN database must be performed. The corresponding GUID must be used to look up in to the GUID index table of the GUID database to get all the PFNs associated with the GUID.

2. THE GRID TECHNOLOGY AND THE ALICE ENVIRONMENT: ALIEN

Currently, the AliEn File Catalogue contains about 443 million files spread in 60 disk storage elements and 10 tape storage elements. The actual available storage and the used storage sizes are approximately 30 PB each. Figure 2.8 shows the growth of the number of file written in the catalogue in the last 7 years.

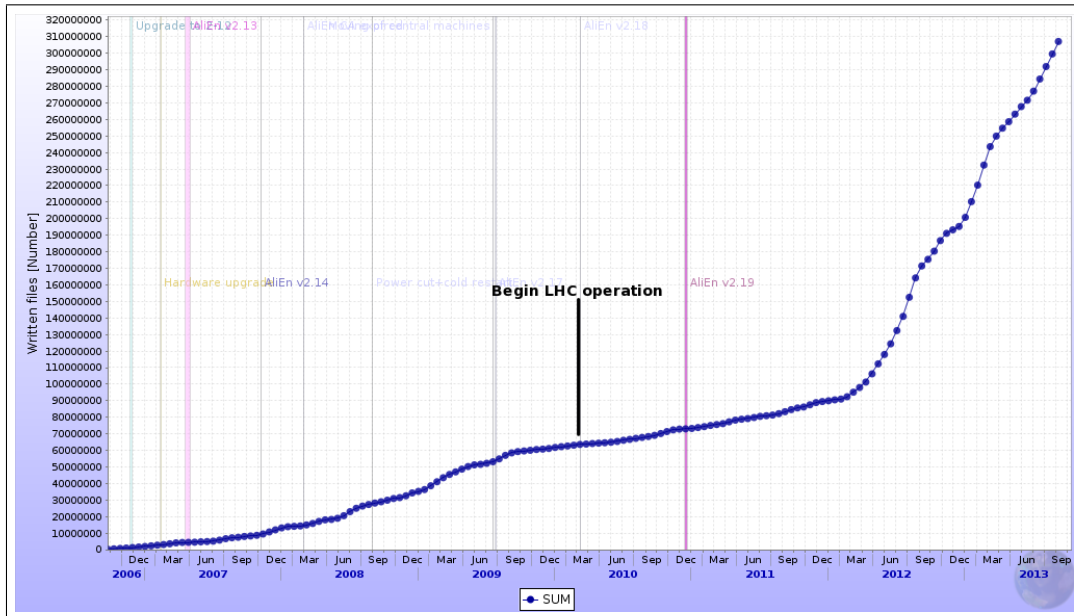


Figure 2.8: The number of files written in the AliEn File Catalogue from 2006 to 2013.

2.3.2.2 Workload Management System

The AliEn *Workload Management System* (WMS) is composed of site services in charge of managing the interfacing of central services to local resources and central services in charge of managing the whole system and distribute the workload. The AliEn site services involved in the WMS are:

- *Computing Element*(CE): runs on every site providing computing resources, it is the interface between AliEn and the local batch system or another Grid system. The task of a CE is to get hold of jobs descriptions, translate them into the syntax appropriate for the local batch system and execute them.
- *Job Agent* (JA): is a pilot job that runs on the nodes and is in charge of getting and running jobs.

The AliEn central services part of the WMS are:

- *Task Queue*: is a database holding all jobs submitted to the Grid.
- *Job Broker*: it is the service in charge of distributing the jobs waiting in the *Task Queue* between the CEs of each site.

In the AliEn WMS, the distribution of jobs is based on a pull model. In general, in the push model it is the broker that gathers information about free resources, and it is the one who pushes the jobs to the free nodes. In the pull model, it is a CE that asks the WMS if there are any jobs to be done that are “matched” to the CE.

According to the pull model concept, the AliEn CEs monitor local resources and, if there are some available, they make themselves known to the job broker by presenting their characteristics [57] .

As shown in the Figure 2.9, the CE sends a description of its capabilities to a *JobBroker* running on the AliEn central servers, which performs the matching with the list of jobs waiting in the *Task Queue*. If there is a match, the *Job Broker* tells the CE to submit pilot jobs (Job Agents). If another CE matches the same job at the same time, the *JobBroker* also instructs the CE to start *Job Agents* (JAs) concurrently.

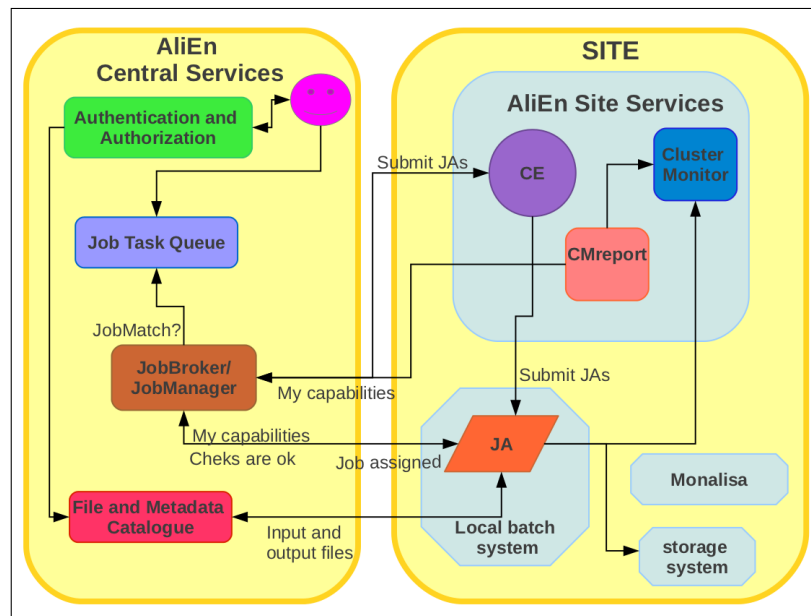


Figure 2.9: The AliEn Workload Management System and interaction between the various elements.

2. THE GRID TECHNOLOGY AND THE ALICE ENVIRONMENT: ALIEN

When the JA starts on the worker node it executes a set of sanity checks on the hardware and software available at the site and if the checks are successful it sends a job request to the *Job Broker*. This request is more detailed than the one sent by the CE, it contains, among other things, the available disk space, the memory, information about the platform and the OS of the *Worker Node* (WN). If the description satisfies the requirements of the job waiting in the *Task Queue*, the *Job Broker* assigns the job to the JA. The JA analyses the *Job Description Language* (JDL) of the job, downloads the necessary input files, prepares the environment for execution, including the software packages requested, and executes the job. Once the user job terminates the JA initiates the transfer of the output files to the storage system and registers the files in the AliEn file catalogue. After that the JA tries to get another job for execution and if there are no more jobs to execute, it exits.

2.3.2.3 Additional services

In addition to the CE, each AliEn site runs the *ClusterMonitor service*, the *CMreport service* and a *MonALISA client*. These services are running on the *VOBox* dedicated machine.

The *Cluster Monitor* manages the connections between the site and central services, it handles communication with the AliEn *Job Broker* and configures JAs.

The *CMreport service* collects the messages from the *Cluster Monitor* and sends them in regular intervals to the *JobBroker/JobManager* in order to minimize asynchronous traffic from the *VOBox* to the central services.

Another service, which is currently integrated in the JA, is the *alien package manager* (packman) that allows the automatic installation, upgrade, configuration and removal of the ALICE software packages on the site. The predefined packages are installed when requested by a JA running on a WN or during the central software deployment over the Grid sites. It will be shown in the following chapters how this functionality has been useful for the realization of this thesis project.

2.3.2.4 Distributed analysis

The distributed analysis process provided by AliEn is shown in the Figure 2.10

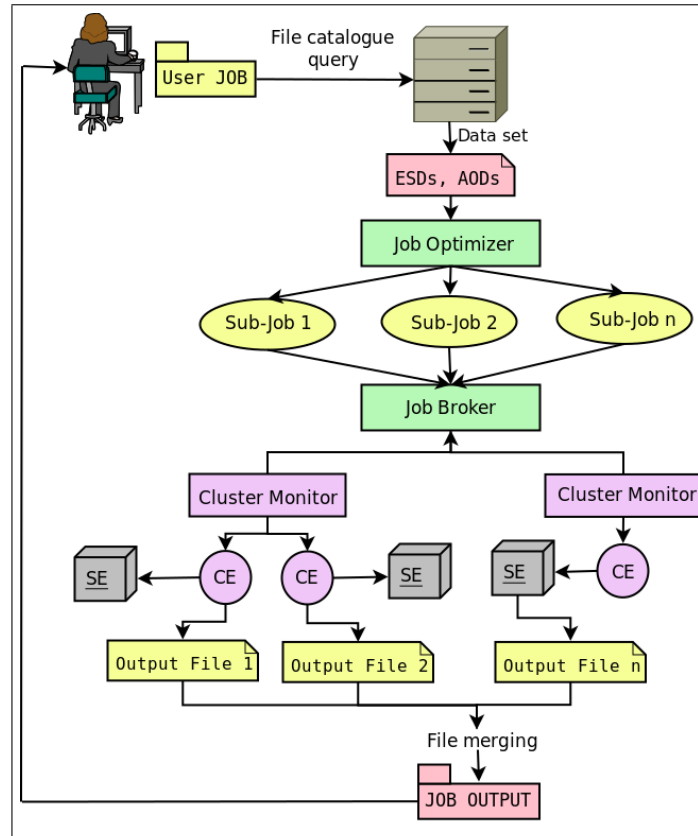


Figure 2.10: AliEn Distributed Analysis.

Jobs are described by using the *Job Description Language* (JDL) , which is based on *Condor ClassAd language*[58] . The only mandatory field for the job description is the executable to run. The AliEn *Job Manager* central service will fill the other fields in case they are empty when the job is submitted to the system. Passing appropriate arguments to the executable, the user can in addition specify a set of optional parameters, like requirements that the worker nodes need to meet (e.g. available disk space, name of the CE etc), the name of the input data file needed by the job, the names of the output files, the name of the packages necessary for the job execution and so on.

2. THE GRID TECHNOLOGY AND THE ALICE ENVIRONMENT: ALIEN

2.3.2.5 Data management tools and services for data transfer, replication and storage

AliEn provides also data management tools and services for data transfer, replication, retrieval and storage.

The files on all *Storage Elements* (SEs) are accessible both locally and remotely over the Wide Area Network (WAN). Although the remote access is contrary to the philosophy of the hierarchical *MONARC model*, the prevalence of networks with sufficient throughput and good inter-site connectivity makes efficient remote data access a reality. The basic rule for job scheduling still remains that the data must be local to the job, i.e. the job is sent to the data. However, remote data access in certain cases simplifies the data management system and increases the overall throughput of the Grid.

Good local and remote access is achieved through the use of the *extended root daemon* protocol (*xrootd*), which is a part of the *Scalla clustering suite* [59]. The framework provides a fast, low latency and scalable data access, which can serve any kind of data, organized as a hierarchical filesystem-like namespace, based on the concept of directory.

Figure 2.11 shows the storage strategy adopted by ALICE that uses *CASTOR2*, *dCache*, *DPM* and *Scalla (xrootd)* as storage technologies and *xrootd* for file access.

CASTOR2 and *dCache* provide a tape backend, and they are used in *Tier-0* and *Tier-1's*. While the *Tier-2* centres use *DPM* and *Scalla* as disk pool managers. *CASTOR2* is a hierarchical storage management system developed at CERN and used to store physics production files and user files. Files can be stored, listed, retrieved and accessed using command line tools or applications built on top of the different data transfer protocols including *xrootd*.

dCache is a system for storing and retrieving data, distributed on heterogeneous server nodes, under a single virtual filesystem tree. It supports also *xrootd* as data access protocol (in emulation mode).

ALICE implements a *Virtual Mass Storage System* according to which all the *xrootd*-based ALICE storage elements are aggregated into a unique meta-cluster, managed by the *ALICE global redirector* host (Fig. 2.12). Each site that does not have tapes or similar mass storage systems considers the meta-cluster as its mass storage

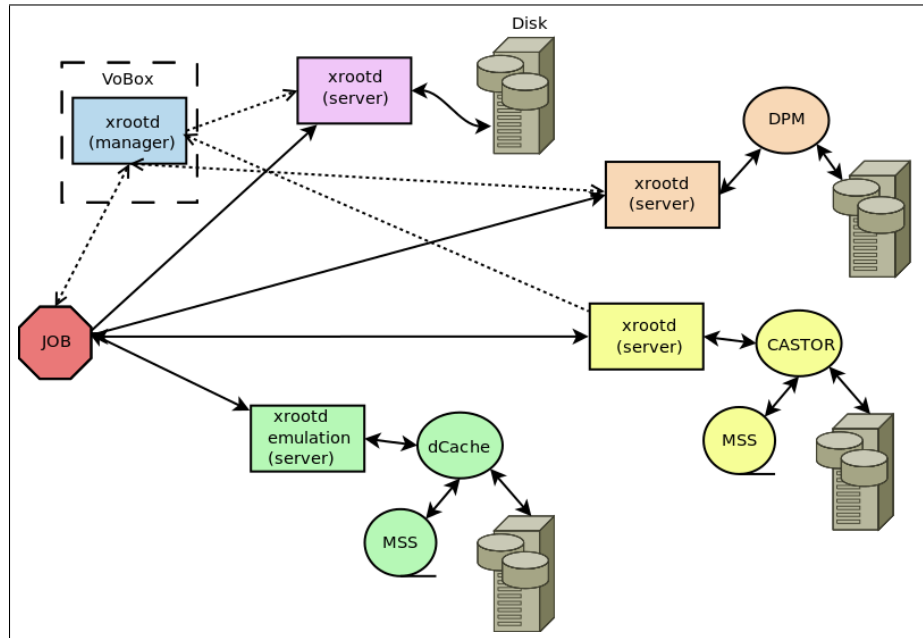


Figure 2.11: Storage strategy used by ALICE.

system. When an authorised request for a missing file comes to a storage cluster, the latter tries to get it from one of its neighbours.

The *xrootd* protocol provides a high and scalable transaction rate, it allows the opening of many files per second and if the system grows the rate grows in a linear way. Furthermore, it keeps CPU usage low and is WAN friendly allowing efficient remote POSIX-like direct data access through WAN. It can also set-up WAN-wide large repositories by aggregating remote clusters or making them cooperate.

2.3.2.6 Authentication and Authorization

The AliEn Authentication and Authorization system is based on the *Virtual Organization Management Service* (VOMS) [60], that is a Grid technology used to provide user access to Grid resources.

The main component of VOMS is a central repository containing authorization information of users as valid Grid certificates. Each user is a member of the *Virtual Organizations* (VOs). VOMS provides interfaces for administrators to manage the users.

AliEn does not use VOMS directly. The authentication is based on *x509 certificates*

2. THE GRID TECHNOLOGY AND THE ALICE ENVIRONMENT: ALIEN

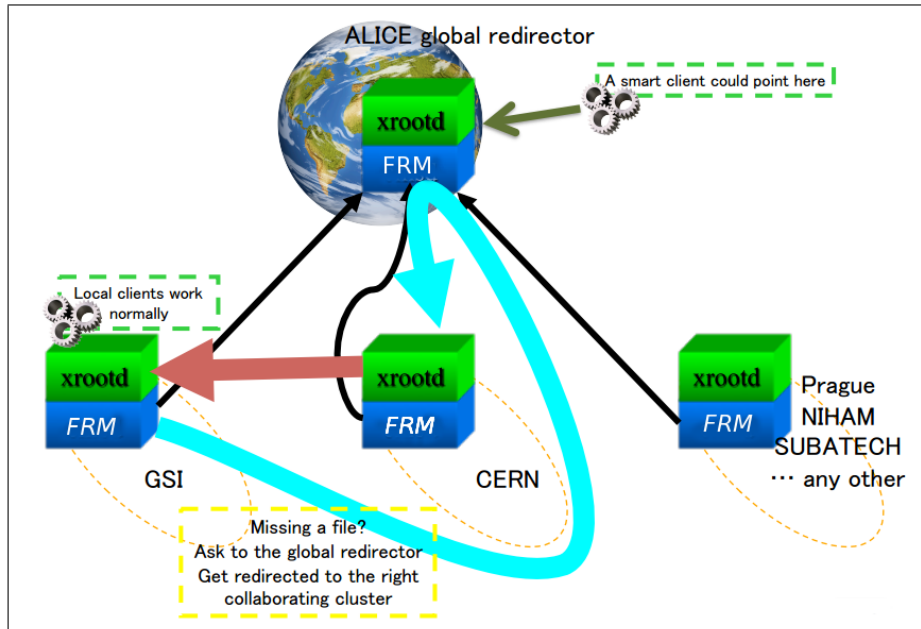


Figure 2.12: An exemplification of the Virtual Mass Storage System.

and is done through *Globus* and *openssl* [61], while the authorization is based on groups and roles which are embedded in the catalogue.

2.3.2.7 Monitoring

The monitoring solution adopted by ALICE is the *MONitoring Agents using a Large Integrated Services Architecture* (MonALISA) [62] framework shown in the Figure 2.13.

The MonALISA system has a hierarchical architecture that allows the monitoring of services, operating systems and network on distributed remote computing sites. The MonALISA system is designed as a group of multi-threaded agent-based subsystems able to collaborate in order to monitor, control and optimize distributed systems.

The structure of the system can be described as a layered architecture where each layer provides a specific group of services.

The basic layer is the *LookUp Services (LUS)* network in charge of providing dynamic registration and discovery for all other services and agents. The registration is implemented using a lease mechanism. A service that fails to renew its lease, is removed from the LUS and a notification is sent to all the services or other applications that subscribed to such events.

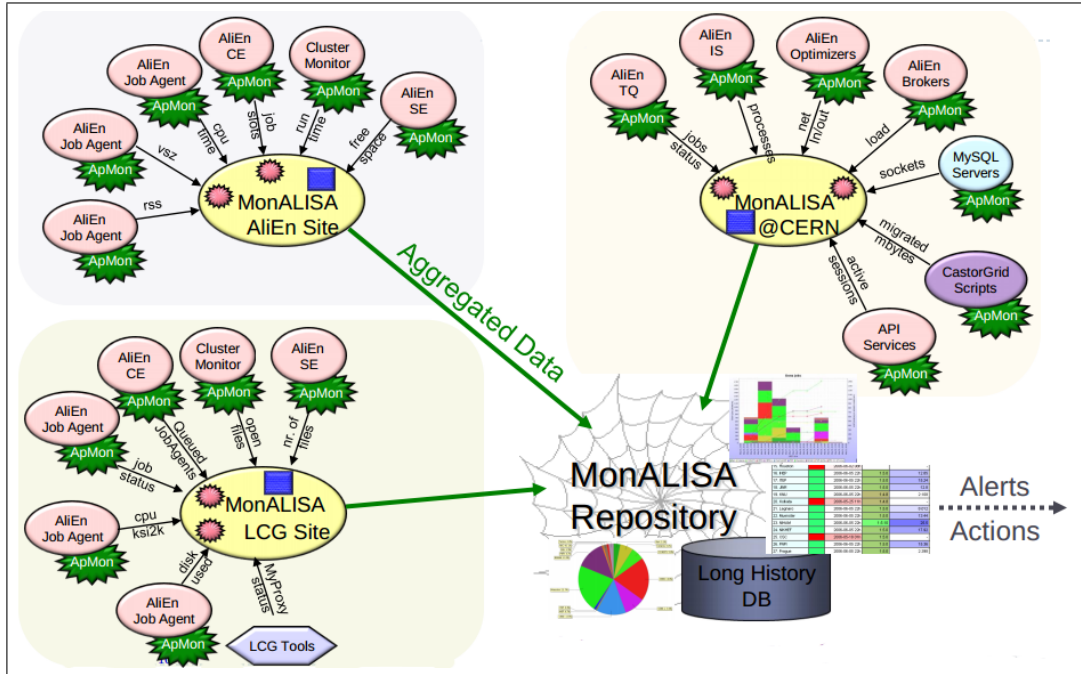


Figure 2.13: The publisher subscriber messaging system used in MonALISA.

The second layer of the system is composed of the MonALISA services. They execute several monitoring tasks in parallel through the use of a multi-threaded execution engine and analyse the collected information in real time by using a variety of agents.

Proxy services are in the third layer of the MonALISA framework and provide an intelligent multiplexing of the information requested by the clients or other services. Proxy services are used for reliable communication among agents.

Higher-level services and clients form the last layer of the system. They use the Proxy layer in order to access the collected information. A location-aware, load-balancing mechanism is used to dynamically allocate these services to the best Proxy service. MonALISA provides also the *MonALISA Application Programming Interface* (ApMon) that can be used by any application to transmit job and host information and services specific parameters. All the monitoring information is published via Web Service.

For the ALICE experiment, the MonALISA service runs on each ALICE *VOBox* and on all the machines running central services. It collects monitoring information from all computing nodes, storage systems, data-transfer applications, and software

2. THE GRID TECHNOLOGY AND THE ALICE ENVIRONMENT: ALIEN

running in the local cluster.

Furthermore, an ALICE-specific mechanism aggregates the raw parameters to produce values giving an overview of the system in real time. An example of aggregation, shown in Figure 2.14, is a single parameter to describe the entire CPU usage by all the jobs on every cluster, obtained by summing up CPU time used by every individual job and subsequently aggregate it per site.

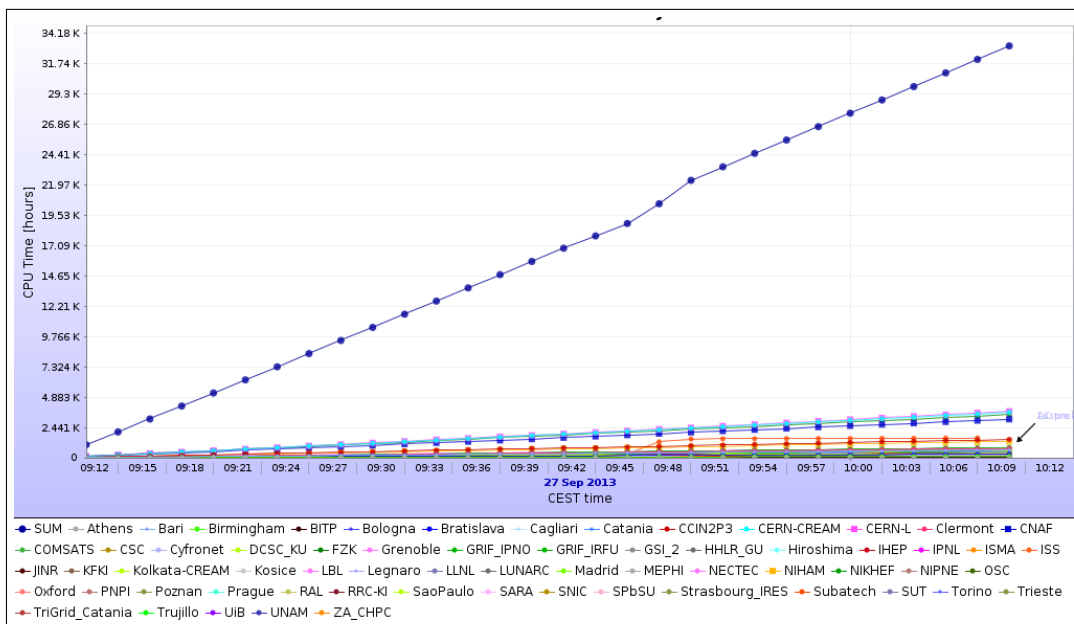


Figure 2.14: Daily CPU time usage by ALICE.

These values are also used for taking automatic actions both locally in the MonALISA service and globally in a MonALISA client. In this way, decision-making agents, available on a central client, help to manage the system by allowing actions such as: restarting remote services when they fail the functional tests; sending notification emails when automatic restart procedures do not resolve problems; coordinating network-bandwidth tests between pairs of remote sites; managing load balancing of the central machines based on the Domain Name System (DNS); automatically executing standard applications when CPU resources are idle.

The actions framework in ALICE is also used to automate the processes of generating Monte Carlo data for simulation of the detector response and reconstruction or analysis of data. MonALISA achieves this by monitoring the central *Task Queue*

(Fig.2.15) taking action when ALICE jobs fail or the number of waiting jobs goes below a certain threshold. First, it tries to reschedule jobs to run and, if the queue length is still too short, it schedules new bunches of 1,000 jobs. The same framework is used to copy data automatically to remote sites and to continuously test the network connectivity among all endpoints.

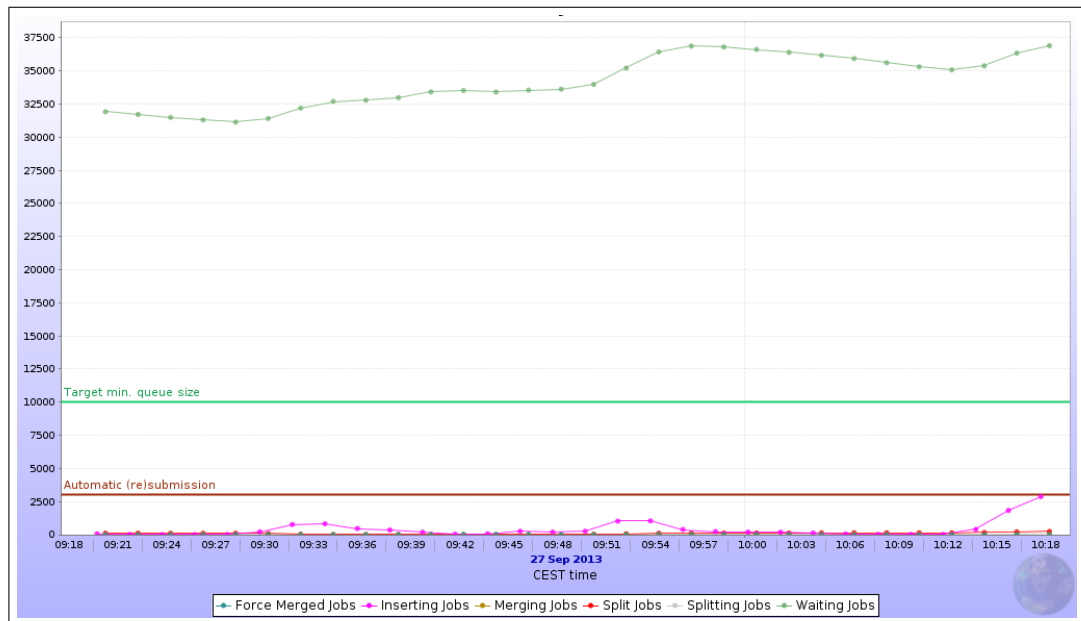


Figure 2.15: Jobs status in the Task Queue

In order to provide long term persistency of the monitoring information, MonALISA keeps all the data collected on each AliEn site in a web repository, implemented by a PostgreSQL database.

2.3.3 User Interfaces

- Command line interface: the AliEn shell `aliensh` provides a UNIX-shell-like interactive environment with an extensive set of commands which can be used to access AliEn Grid computing resources and the AliEn virtual file system.

Using these interfaces, it is possible to access the catalogue, submit jobs and retrieve the output. There are three categories of commands:

- Informative and Convenience commands such as `whoami`;

2. THE GRID TECHNOLOGY AND THE ALICE ENVIRONMENT: ALIEN

- File Catalogue and Data Management commands which include all the UNIX shell commands (`cp`, `rm`, etc);
- TaskQueue/Job Management commands as `top` and `ps`:
 - `top`: checks the status of submitted jobs, `top` accepts arguments in order to display jobs from a particular user, or jobs with a certain status.
 - `ps`: while `top` gives the list of all submitted jobs, `ps` gives the list of jobs of the current user. It can be used to show the jobs JDL during or after the jobs runtime: `ps -jdl Job-ID`.

Figure 2.16 shows the basic structure of the `aliensh` interface to AliEn. In addition to a shell-like interface, a C library allows the Grid functions to be used directly from the ROOT interface. On the right hand side (AliEn box) are the AliEn central services, introduced elsewhere in the text.

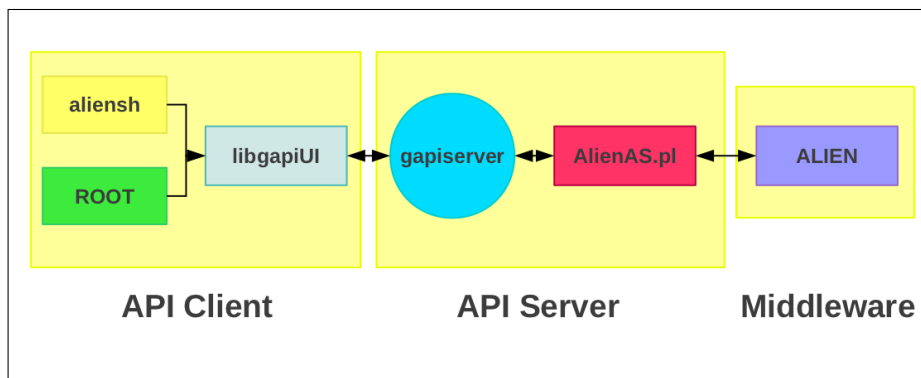


Figure 2.16: Basic elements of the `aliensh` user interface.

The API for all client applications is implemented in a library called `libgapiUI.so`. Every client interface can communicate over a session-based connection with an API server. For the authentication to the server, a session token is used. Tokens have a limited lifetime and are modified every time they have been used. Each token has a predefined role and is sent by the server to the client over an SSL connection.

- ROOT interface: the basic part of the ROOT interface is the `TGrid` class implemented using the AliEn C++ API. The `TAlien` class (`TGrid` inherited), implements all the methods to connect and disconnect from the Grid and to

browse the virtual file catalogue.

Several other classes (`TAlienFile`, `TAlienJobIO`, `TAlienAnalysis` etc) have been implemented using the AliEn C++ API to allow the submission of jobs to AliEn from ROOT and to access the files stored in the global AliEN catalogue.

- Interfaces to other Grid infrastructures (ARC, EMI, OSG): currently, AliEn provides three interfaces to interoperate with the ARC, EMI and OSG Grids. Using this feature, the AliEn CE can be configured to serve as a gateway to an entire foreign Grid infrastructure.

2.3.4 Future AliEn development: JAliEn

The interface between users and central AliEn services has been recently re-implemented using the Java language. At the beginning, the framework was developed to control, in an automatic way, central productions, data registration and replication according to MonALISA monitoring information.

Currently, this interface called *JAliEn* [63], provides a full implementation of the AliEn objects and their interactions with the central databases. Figure 2.17 shows the architectures both of AliEn and JAliEn.

Three new components have been added:

- JCentral: implements the AliEn components in Java objects (Job, SE, users, file catalogue objects etc) and provides direct connections to the DBs for all operations. The minimal set of commands is implemented as API calls through serialisable objects encapsulating the requester's identity.
- JSite: runs on the site *VOBox* and it is in charge of forwarding requests to the central services. Some of the JSite objects are cacheable and can be quickly returned to the requester (E.g. OCDB file locations). It can be cascaded indefinitely in order to reduce the number of sockets per server to a reasonable number.
- JBox: provides authentication and authorization avoiding the use of proxies for the users, the certificate is loaded asking for the password at the time of the first request. *JBox* provides also secure communication services with Java SSL sockets such as listing, opening and writing files, submitting and accessing job information. It consists of 3 components:

2. THE GRID TECHNOLOGY AND THE ALICE ENVIRONMENT: ALIEN

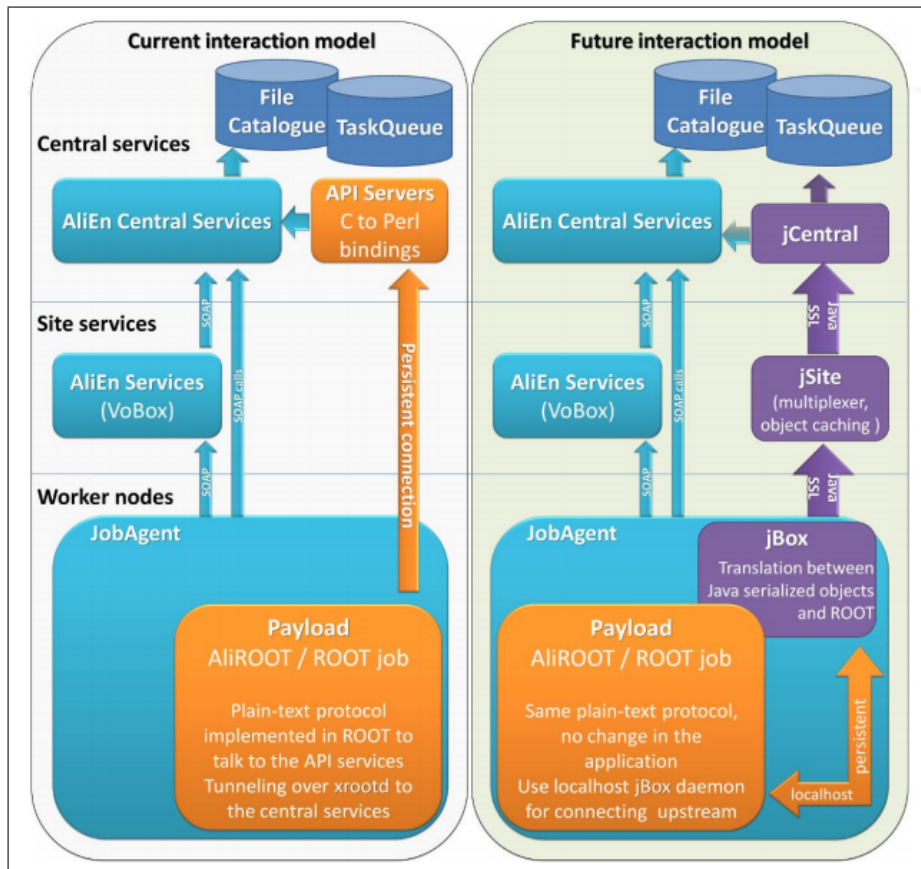


Figure 2.17: The AliEn middleware and the new JAliEn.

2.3 The ALICE Grid environment

- Java daemon responsible for authentication, authorization and communication between user applications and central services.
- ROOT implementation of TGrid functionality which is a small independent C++ library responsible for communication between ROOT and Java daemon.
- Java shell client which has the same features of the AliEn client, except that commands are implemented as classes using introspection.

2. THE GRID TECHNOLOGY AND THE ALICE ENVIRONMENT: ALIEN

Chapter 3

The ROOT parallel analysis facility and the dynamic PROOF framework

This chapter will briefly introduce the ROOT data analysis framework, followed by a thorough discussion of PROOF, the ROOT parallel analysis facility framework. Two different ways of using PROOF will be outlined: the static PROOF cluster provided at CERN and the dynamic PROOF solution provided at the *Helmholtz Centre for Heavy Ion Research* (GSI) [64] .

3.1 ROOT analysis framework

ROOT is an object-oriented C++ framework developed by the HEP community in order to allow the storage, retrieval, visualisation and analysis of physics data.

The ROOT framework includes a hierarchical Object-Oriented I/O sub-system to provide an efficient data storage, access and query system, a C++ interpreter, statistical analysis algorithms like multi dimensional histogramming, multivariate analysis, statistical and cluster finding algorithms, a visualization tool, a virtual Monte Carlo interface, a geometrical modeller and a parallel analysis facilities engine (PROOF).

The geometrical modeller is used for the simulation, visualisation and reconstruction of the experiments detectors geometries, it has been part of a project executed during the thesis on which a detailed description can be found in the appendix.

3. THE ROOT PARALLEL ANALYSIS FACILITY AND THE DYNAMIC PROOF FRAMEWORK

3.1.1 The access and storage data system

ROOT provides an access data system based on a selective sparse scanning of data implementing a vertical data partitioning of objects, defined by using the *TTree* class (Fig.3.1).

The *TTree* containers can span a large number of files on local disks, on the Grid or on a number of different shared file systems. They are partitioned into branches which store consecutive objects or data members of a class or other *TBranches*. Following the so-called *splitting process*, branches can be composed by entire objects of a class or they can be divided into sub-branches, where each sub-branch contains individual data members belonging to the original object. During the reading phase each branch can be accessed independently.

Since all branches stored in a *TTree* are written into separate buffers in a file, iterating over the data stored in a branch implies only the reading of these associated buffers.

This approach to the storage based on branches called *vertical* or *column-wise storage* (CWS) is the opposite of the *horizontal* or *row-wise storage* (RWS), which is usually found in *Relational Database Management System* (RDBMS) databases. The CWS method adopts a block-wise reading of several entries at the same time because data members that should be read are consecutive on the storage medium. This approach leads to the reduction of the number of I/O operations as well as to the reduction of the amount of transferred data.

Usually, the really large amount of data processed by ROOT users leads to the creation of *TTrees* that do not fit into a single file and that needs to be split. In addition, in (parallel) batch system-based analyses, splitting *TTrees* across several files facilitates the distribution of data. The splitting of *TTrees* is done in ROOT by means of *TChain*, which are a collection of *TFiles* that all contain a part of the same *TTree*.

ROOT implements a pre-fetch mechanism according to which the reading of the next entry is performed while the previous entry is still being processed reducing the effect of high latency networks.

Since ROOT is used for storing Petabytes of data, the I/O layer is extremely important. It is used to store objects in storage systems such as file systems and databases or common protocols for storage elements such as xrootd or dCache.

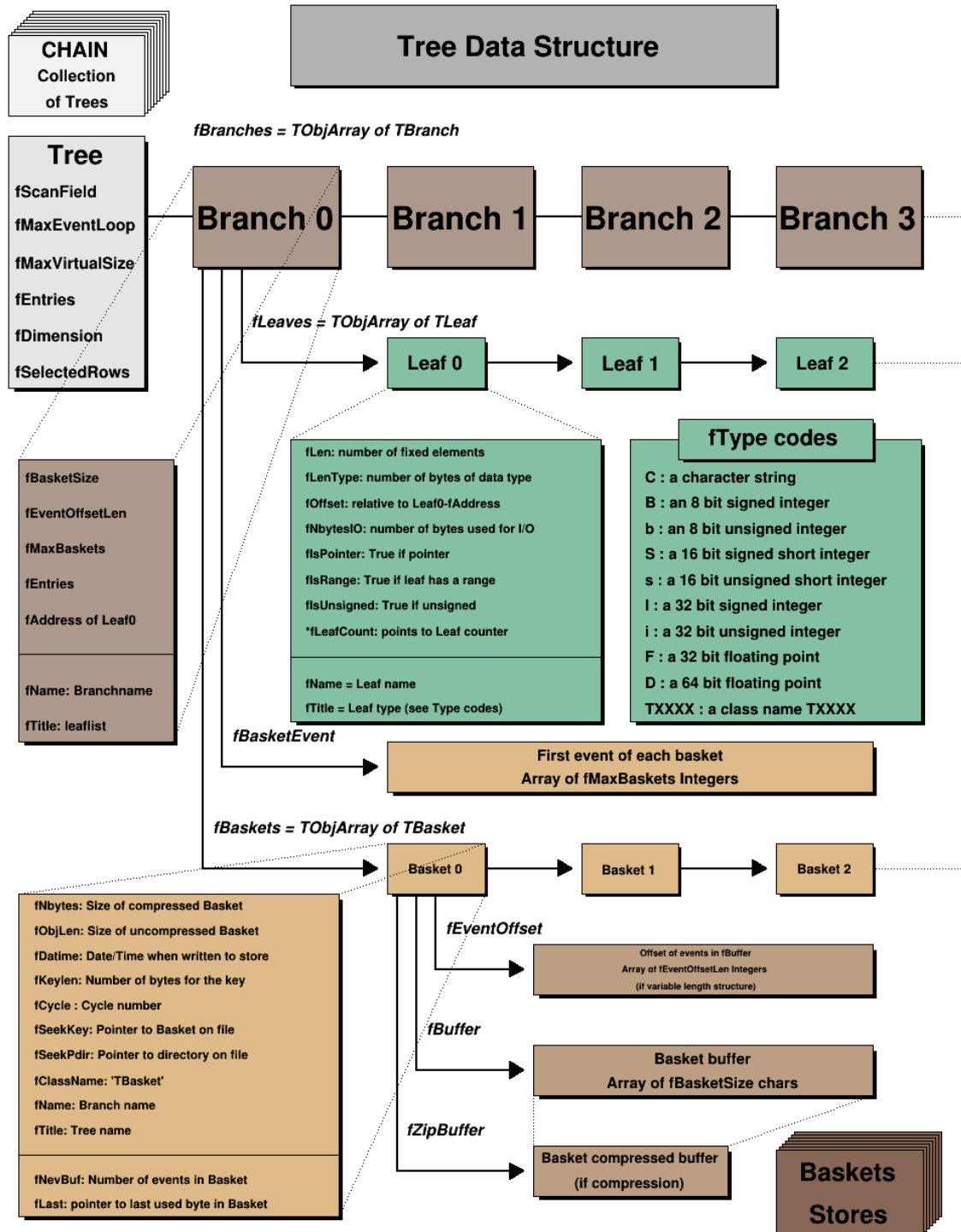


Figure 3.1: The *TTree* data structure.

3. THE ROOT PARALLEL ANALYSIS FACILITY AND THE DYNAMIC PROOF FRAMEWORK

3.1.2 The C++ interpreter

By means of CINT, the interactive C++ interpreter, users can create their analysis macros step by step starting with small data samples. Afterwards, they can run the macros over big datasets performing the compilation process during the interactive session by by means of *The Automatic Compiler of Libraries for CINT* (ACLiC) and benefiting from the full speed and the reliability of the C++ compiler. Currently, the ROOT team is migrating from CINT interpreter to the Cling interpreter, which is built on the top of LLVM [65] and Clang libraries [66] . Its advantages over the standard interpreters are that it has a command line prompt and uses just-in-time (JIT) compiler for compilation.

3.1.3 Mathematical and statistical analysis algorithms

To analyse data, ROOT provides a large set of mathematical and statistical functions, like linear algebra classes, numerical algorithms such as integration and minimization, and regression analysis methods (fitting). The *MathCore library* provides tools for numerical computing, it contains mathematical functions not provided by the C++ standard and it provides statistical functions like probability density, cumulative and its inverse for each statistical distribution. The *MathCore library* contains also classes for random number generation, implementation of numerical algorithms, like integration or derivation. Classes required for fitting all the ROOT data objects are also provided.

Another module of ROOT is the histogram package that provides classes for binning of one and multi-dimensional data. Operations like addition, subtraction, multiplication and division are supported between histograms (see Fig. 3.2) . To plot histograms ROOT provides the Draw() method which displays histograms allowing interactive manipulation.

3.1.4 The GUI and the Event Visualization Environment

The ROOT framework comes with a user-friendly *Graphical User Interface* (GUI) rich in functionalities providing tools for developing user interfaces based on the ROOT GUI classes. ROOT can also be used by command line or by batch scripts.

To display 2D plots ROOT is interfaced with X11 graphics engine for Unix systems,

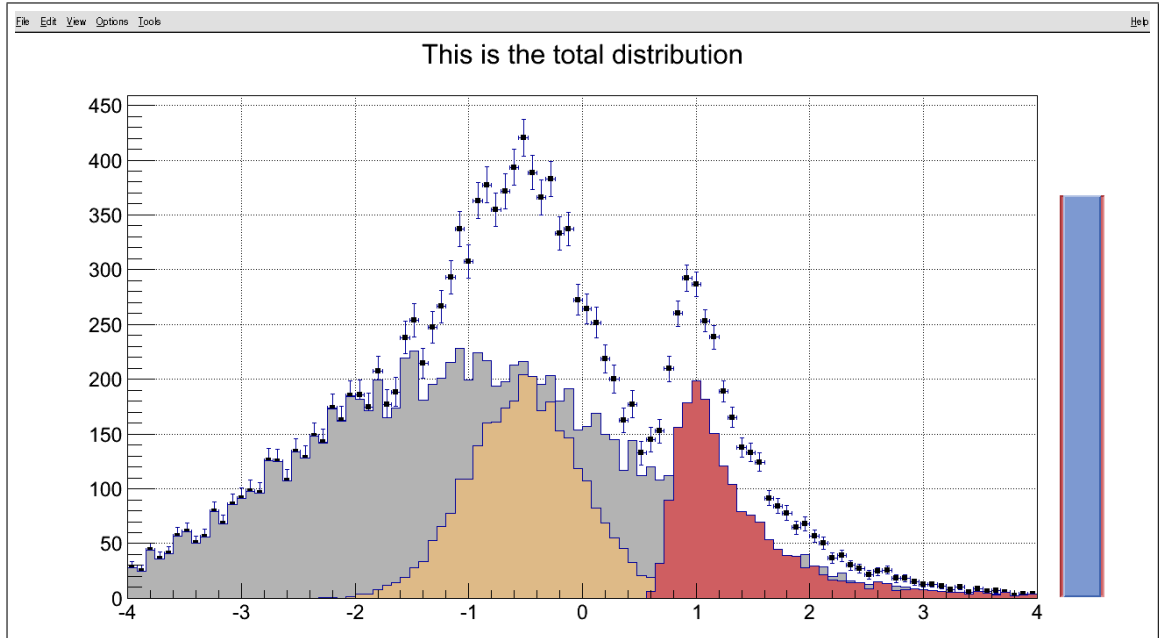


Figure 3.2: Example illustrating how to use ROOT to fill histograms in a loop and show the graphics results.

with Win32 for Windows systems and Cocoa for Mac. The rendering of 3D graphics is provided by using the OpenGL graphics library, which is very useful for representing large detector geometries.

ROOT provides also an *Event Visualization Environment* (EVE) which permits the organisation of data in a hierarchical structure and the management of object interaction and visualization via GUI and OpenGL representations. EVE is also used for visualization of detector geometry, simulation and reconstruction data like hits, clusters, tracks and calorimeter information.

3.1.5 The Virtual Monte Carlo Interface

The *Virtual Monte Carlo* (VMC) is a ROOT interface to several HEP simulation software like Geant4, Geant3 [67] and Fluka [68]. By means of the VMC interface it is possible to build an application to simulate the passage of particles through matter and their propagation in a magnetic field independent of the actual transport Monte Carlo used.

The VMC interface gives the possibility of switching between different simulation

3. THE ROOT PARALLEL ANALYSIS FACILITY AND THE DYNAMIC PROOF FRAMEWORK

software packages simulating the behaviour of the particle detector. In this way, it is possible to compare the results of different simulation software in order to estimate systematic uncertainties.

3.2 The PROOF framework

PROOF (*Parallel ROOT Facility*) is an extension of the ROOT data analysis framework enabling interactive analysis of large sets of ROOT files in parallel on clusters of computers or many-core machines.

PROOF enables the inherent event parallelism of independent HEP events by distributing the analysis over many CPUs and disks of a multi-core machine or a computing farm.

The PROOF system has been designed taking into account three main basic concepts. Firstly, it has to be transparent, in the sense that the difference between a local ROOT session and a remote parallel PROOF session has to be minimized as much as possible, consequently the analysis macros have to be executed in both systems without the necessity to perform any change on the code. Secondly, there should be no limitations on the number of computers that can be used in parallel, this means that PROOF should not have scalability problems. Thirdly, the system must be adaptable providing a prompt answer to any variations in the remote environment such as a network interruption.

A PROOF user has the possibility of running ROOT analysis jobs on large clusters in a transparent and interactive way.

In addition to the interactive mode, PROOF provides also an interactive-batch mode that allows users to start very long jobs, disconnect the client and reconnect to the job at any time from any location to monitor the progress or retrieve the result. The main advantage of this feature is that the user does not need to wait until all the sub-jobs have been finished and merged to get an answer.

3.2.1 System architecture

The PROOF system is based on a Multi-Tier Master-Worker architecture (Fig. 3.3), consisting of the master server, the worker machines and the client session.

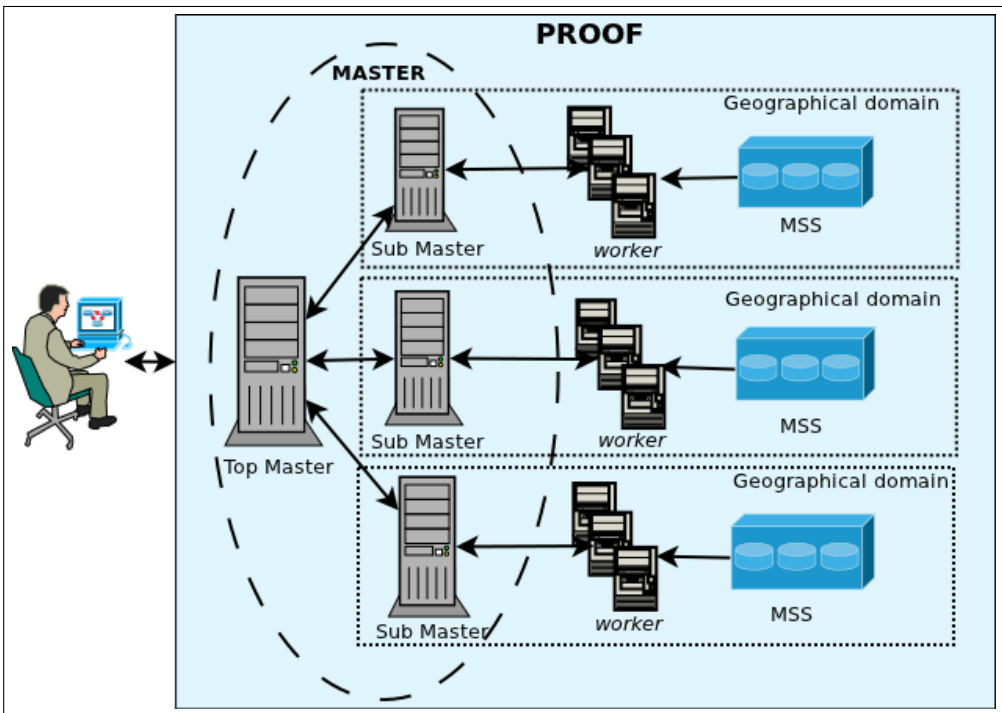


Figure 3.3: The Multi-Tier PROOF Architecture: the client sends jobs to the master by using a ROOT session. The master distributes the jobs to the workers which might be connected to a Mass Storage System

3. THE ROOT PARALLEL ANALYSIS FACILITY AND THE DYNAMIC PROOF FRAMEWORK

The ROOT session tier allows the connection between the user and the master server located on a remote cluster, the master server creates the worker processes on all the nodes in the cluster. The task of the workers is to execute all the queries in parallel. Each worker asks the master for work packets. The master manages the workloads by sending smaller work packets to slower workers and larger packets to faster workers.

Due to the fact that the bandwidth and latency of a networked cluster are fixed, the parallel processing performance in this case depends on the packet size. If the size of the work packets are too small, the number of packets sent over the network between master and workers will increase thus decreasing the performance of the parallelism due to the communication overhead. On the other hand, if the packet size is too large the performance of each node is not sufficiently equalised. This approach allows PROOF to manage the load on each individual cluster node and to optimize the performance and the job execution time.

In addition, the master tier can be multi-layered. This allows the implementation of geographical federation of distributed clusters thus optimizing the access to auxiliary resources, like mass storage elements. It also allows the spreading of the distribution and merging work, which can be responsible for a decrease of the performance if there are several workers involved.

PROOF has been designed as an alternative to batch systems for Central Analysis Facilities and work groups like *Tier-2* and *Tier-3*. The multi-tier architecture that allows multiple levels of masters simplifies the use of PROOF working with virtual clusters distributed over domains that are spread out physically as in the GRID.

Modern PROOF implementation works in close cooperation with *Xrootd*, a file serving protocol used for communication, clustering, data discovery and file serving. When a PROOF session is started the *xproofd daemon* needs to be running on a set of machines. *Xproofd* is just a plug-in for *Xrootd* and it has been implemented to simplify the configuration when data-serving is not needed or it is handled differently.

3.2.2 Use case

When the user starts the session, a *Transmission Control Protocol (TCP)* socket to the master is created and the client will be connected to the *master xproofd*.

Once logged in, the master process *proofserv* is forked by *xproofd* and it calls back *xproofd* to set-up the connection with the client by means of a UNIX socket. The

proofserv process reproduces the ROOT framework, with the only difference that it reads commands from a socket instead of from the terminal.

When the client is connected, the master starts to set-up the workers and connects to them one by one using the same procedure used to connect to the client. Figure 3.4 shows the individual elements, services and communication layer of a PROOF cluster.

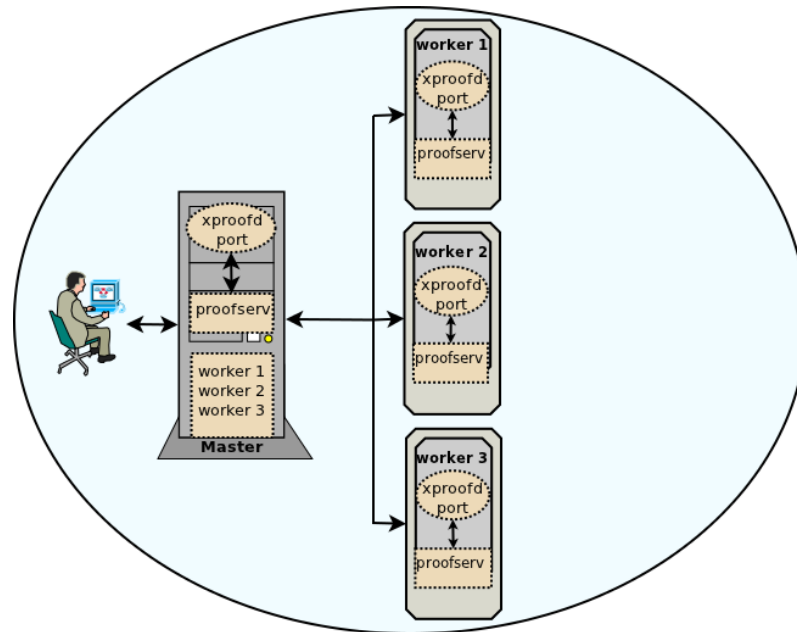


Figure 3.4: A schematic view of a PROOF cluster. The two main elements PROOF master (on the left) and the PROOF workers (on the right) communicate through the *rootd* protocol. This protocol is also used for data I/O.

3.2.3 PROOF features

The PROOF system needs to take care of two important aspects: the *resource scheduling* that is the application of strategies for the assignment of the right share of resources to jobs and the *load-balancing* for the optimization of the workload distribution within jobs.

3.2.3.1 PROOF load balancing engine

The PROOF load balancing engine is called *Packetizer* and it is used to distribute the workload among nodes.

3. THE ROOT PARALLEL ANALYSIS FACILITY AND THE DYNAMIC PROOF FRAMEWORK

In order to optimize the execution time by getting all the tasks terminated at the same time, the PROOF master divides each task into pieces called *packets*, which are sent to workers. The amount of packets assigned to each worker is dynamically determined according to the real time performance of the workers.

The packet does not contain any data but only a name of a file with HEP events local or remote and a range of events to be processed.

Each event is independent from the others, so it can be processed independently on any node at any time. In this way, PROOF exploits the inherent parallelism of HEP data.

There is an instance of the *Packetizer* on the master node for each job. If a multi-master configuration is used, then there is one *Packetizer* on each of the sub-masters.

A pull approach is used in order to dynamically balance the work distribution, workers ask for new packets when they are ready for further processing. The aim of this approach is to have all the nodes finish at the same time. Once they have executed their packets they send the results to the master.

In addition, the *Packetizer* has to manage packet executions so that the workers finish using the available resources as soon as possible. To achieve this goal there are multiple solutions depending on the type of jobs and the system must be able to use a different strategy according to the requirements of the situation.

According to the general algorithm of the data-driven *Packetizer*, the locations of all the files are checked as first step. Secondly, the files are validated in parallel by the workers and, finally, there is the processing phase during which the *Packetizer* keeps assigning packets to the workers in order to process the whole data set. The packets can be assigned according to two different strategies:

- The *basic Packetizer* which takes into account that the analysis of files at their location is more efficient than remote analysis. Under this approach the worker analyses the local part of the data set as first step and then analyses remote files.
- The *new adaptive Packetizer* strategy dynamically predicts how much time each worker takes to analyse the local part of the data set. The worker that is expected to analyse its local data faster than the average can be asked to process other data before its local analysis is completed. This strategy tries to avoid the situation in which some nodes become bottlenecks at the end of the job.

Performance measurements show good efficiency and scalability for the basic Packetizer when the datasets are uniformly distributed. On the other hand the efficiency decreases when the datasets are randomly distributed. The adaptive Packetizer strategy using the predictions minimizes the processing time and improves the resource utilization.

3.2.3.2 PROOF resource scheduling

The resource scheduling is the second important aspect to be considered. This is the service that assigns workers to the jobs thus maximizing the efficient use of resources and ensuring the best response time possible on multi-core machines as well as on big clusters. The best resource utilization is obtained when there is a good balance between the load of the machines and the location of the datasets.

PROOF resource scheduling operates both at the worker node level and at the central level. On each worker node it checks which resources are used by each job, according to the user priority. The central scheduler sets the user priorities and assigns workers to the jobs according to the current state of the cluster and the usage history. PROOF provides some scheduling options that can be combined together:

- *Load based scheduling* with which the decision of the number of workers assigned to a job is based on the current load of the system.
- The *Priorities and fair-share* according to which the number of workers assigned to a job can also depend on the priority of the user or group. The priorities can be static or calculated dynamically in conjunction with a monitoring system like MonALISA.
- The *Queueing option* that can be used to handle the situation of congestion where the volume of submitted jobs exceeds the processing capability. The alternative is to reject new queries until the system load decreases. The *First in First Out (FIFO)* queue can be combined with the load based scheduling or the static scheduling with a limit on the maximum number of running jobs/sessions set.

3.3 The PROOF cluster at CERN: CAF

The *CERN Analysis Facility (CAF)* [69] is a cluster of machines running PROOF in order to provide to ALICE users the possibility to perform interactive parallel process-

3. THE ROOT PARALLEL ANALYSIS FACILITY AND THE DYNAMIC PROOF FRAMEWORK

ing of data.

The `xrootd` and the *analysis facilities dataset manager daemon* (`afdsmgrd`) software are part of the CAF cluster and used for data management. They allow operations like copying files from Grid to CAF storage or remove files from CAF.

CAF is also used by the ALICE physicist community, consequently it is running AliEn and it uses the `packman` in order to install and update experiment analysis software (AliRoot, ROOT, Geant etc).

The aim of CAF is not only to provide interactive parallel data analysis but also calibration and alignment of detector components and make it possible to perform fast simulation and event reconstruction.

Figure 3.5 shows the CERN Analysis Facility that currently consists of about 400 CPU cores and 162 TB of space for local staging.

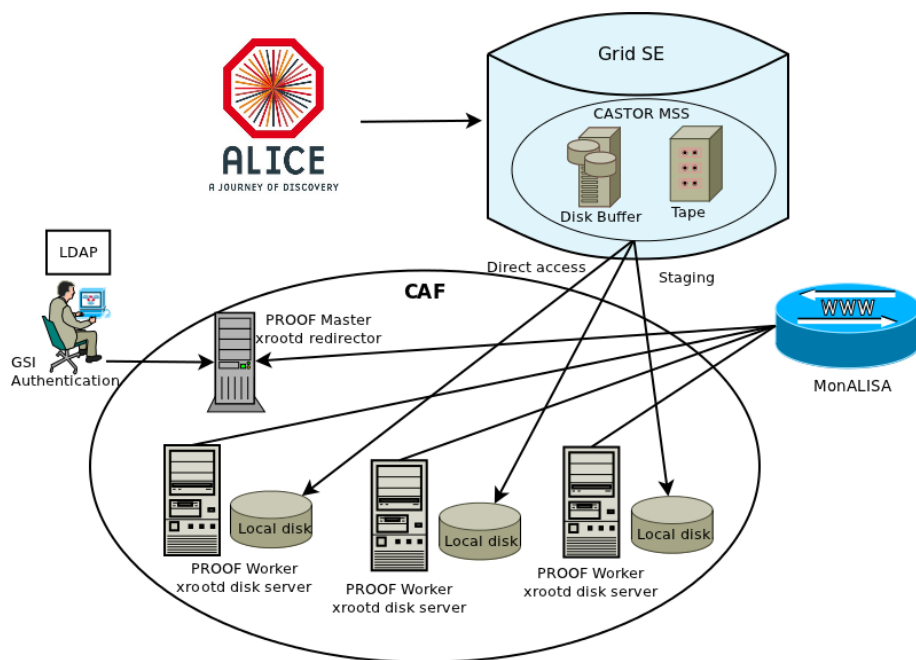


Figure 3.5: The CERN Analysis Facility architecture.

One machine of the cluster runs the PROOF master and the `xrootd` redirector for file serving. Another machine is used as backup master and the rest are used as PROOF workers and `xrootd` disk servers. Having the data on the disk server of the worker node makes it possible to perform the local analysis of data that is more efficient than remote data analysis.

3.3 The PROOF cluster at CERN: CAF

The CAF cluster is monitored by the MonALISA system, where information like disk quota, CPU priority or user sessions can be inspected. The host monitoring provides a table of the resource utilization (Fig. 3.6). For each machine the status of the services, the current load, the number of PROOF users, the CPU usage, the memory consumption, the network traffic and the number of hosted files with total size are shown. Furthermore, histories of user session are provided and an average of seven connected users is shown.

Machines status																
Machine	Machine status				CPU				Networking			Disk space				
	Online	xproofd	xrootd	cmsd	load1	# proof	usr	sys	nice	idle	IN	OUT	Total	Free	Used	%
1. lxbsq1130					-	-	-	-	-	-	-	-	-	-	-	-
2. lxbsq1131					-	-	-	-	-	-	-	-	-	-	-	-
3. lxbsq1134					1.66	6	14.07	0.408	0	85.24	7.559 MB/s	25.52 KB/s	2.412 TB	155 GB	2.261 TB	93.73
4. lxbsq1137					-	-	-	-	-	-	-	-	-	-	-	-
5. lxbsq1142					1.08	6	11.83	0.342	0	87.71	432.3 KB/s	7.223 KB/s	2.498 TB	145 GB	2.356 TB	94.33
6. lxbsq1216					0.5	6	0.578	0.276	0	98.47	530.1 KB/s	9.798 MB/s	2.498 TB	131.1 GB	2.37 TB	94.87
7. lxbsq1226					-	-	-	-	-	-	-	-	-	-	-	-
8. lxbsq1227					1.25	6	12.12	0.359	0	87.29	4.269 MB/s	13.81 KB/s	2.498 TB	133.5 GB	2.368 TB	94.78
9. lxbsq1229					0.67	6	9.263	0.327	0	90.17	11.09 MB/s	41.55 KB/s	2.412 TB	175.8 GB	2.241 TB	92.88
10. lxbsq1232					4.02	6	0.686	0.61	0	85.84	0.81 KB/s	0.667 KB/s	2.498 TB	122 GB	2.379 TB	95.23
11. lxbsq1234					1.12	6	12.67	0.463	0	86.63	15.34 MB/s	70.59 KB/s	2.412 TB	2.287 TB	128.6 GB	5.208
12. lxbsq1235					1.31	6	9.983	0.312	0	88.92	86.17 KB/s	90.17 KB/s	2.412 TB	151.1 GB	2.265 TB	93.88
13. lxbsq1238					2.06	5	2.599	0.969	0	86.9	10.27 MB/s	5.704 MB/s	2.498 TB	122.5 GB	2.378 TB	95.21
14. lxbsq1240					-	-	-	-	-	-	-	-	-	-	-	-
15. lxbsq1244					0.97	-	4.277	0.358	0	92.3	342.3 KB/s	4.032 KB/s	2.498 TB	133.3 GB	2.368 TB	94.79
16. lxbsq1409					1.22	3	1.828	12.6	0	85.39	70.42 KB/s	5.074 MB/s	-	-	-	-
17. lxbsq1410					-	-	-	-	-	-	-	-	-	-	-	-
18. lxbsq1411					0.02	6	0.469	0.179	0	99.26	282.3 KB/s	3.198 KB/s	2.498 TB	906.7 GB	1.612 TB	64.55
19. lxbsq1412					0.11	6	0.165	0.137	0	99.58	0.726 KB/s	0.833 KB/s	2.412 TB	152 GB	2.264 TB	93.85
20. lxbsq1413					0.78	6	12.35	0.356	0	87.01	13.42 MB/s	44.94 KB/s	2.498 TB	143.3 GB	2.358 TB	94.4

Figure 3.6: The machines status of the ALICE PROOF cluster.

Not all the data produced by the experiment is significant for interactive parallel analysis therefore, the CAF disk space is not used as a permanent storage but rather as cache space on which the user can import a selection of data from the storage elements.

The *staging operation* is an automatic process in CAF which is performed by the *Cluster Management Service Daemon (CMSD)* service, a part of the xrootd software.

To describe the input for analysis or for data staging from AliEn the concept of *dataset* is used. A *dataset* is a list of files selected by the user to be processed with PROOF. When a new *dataset* is registered by the user, the xrootd redirector on the PROOF master machine selects the xrootd disk servers and forwards the request to stage each file. Each disk server sends the request to its data-stager that performs the

3. THE ROOT PARALLEL ANALYSIS FACILITY AND THE DYNAMIC PROOF FRAMEWORK

staging.

CAF has also a *garbage collector* mechanism that deletes files with the oldest last access time when the disk usage is high (90%) and stops when it is about 80%.

On the PROOF master, the description of each *dataset* is stored in a file. A data manager daemon checks regularly the *datasets* to see if the files listed in the *datasets* are available. If files are missing, staging is triggered by sending a stage request to the redirectors CMSD service. Once a file is staged, meta information is extracted and stored in the corresponding *dataset*, e.g. the files location, the size and the number of the contained events.

Files are regularly consulted thus preventing their clearance by the garbage collector on the disk servers. If a *dataset* is removed, files are not deleted automatically but just not touched: in this way the garbage collector will permanently remove files after a defined amount of time. This simple approach has the advantage of not requiring to loop over the existing *datasets* to verify whether a file is used or not.

The CAF users are split in groups, each group has its assigned level of priority for managing the concurrent queries. The priority algorithm assures a certain amount of CPU time for each group. The priorities of the groups are defined in the CAF configuration, while the consumed CPU time is retrieved from MonALISA. The algorithm calculates the difference between usage and quotas and computes the new priorities, privileging groups using less than the defined quota and demoting those groups using too much CPU. New priorities are sent to PROOF, which executes the processes accordingly.

User authentication is based on the *Globus Security Infrastructure* (GSI) and uses X509 certificates and an LDAP-based configuration management. Grid certificates are used to authenticate the users to the system. In this way, the same means of authentication is used for the Grid and for CAF, giving the added advantage of possible direct access to the Grid files via the CAF workers.

3.4 The PROOF On Demand framework

The default installation of PROOF is a static cluster. In order to become a PROOF worker or master, a node has to run the `proofd` daemon. This well-proven approach, has its limitations, especially for analysis of larger datasets or when the installation of

3.4 The PROOF On Demand framework

a dedicated cluster is not possible. A more flexible solution of a local cluster can be adopted using the *PROOF on Demand* (PoD) framework [70].

PoD was implemented at GSI in Darmstadt to give the possibility to users to dynamically set-up a PROOF cluster on request, on any resource management system. It provides a command-line interface, or a simple graphics user interface, in order to simplify access to its functionality.

PoD allows the use of PROOF directly on Grid-enabled clusters or batch systems, by dynamically assigning interactive nodes on user request. This makes it possible to design and build a system, which will automatically start and use the PROOF framework in a dynamic way, creating workers upon request and without any of the potential workers having to run any daemons to become part of the PROOF cluster.

The PoD design takes into account that the user interface and the PoD workers could be multicore machines. Many concurrent processes of PoD from different users can be running on the same physical machine.

PoD has an internal mechanism to decide which port to use for which service, the user only needs to define a port range. Using this mechanism if PoD finds an existing xrootd process running on a worker node under the current user account, it will force all the services to use it.

Figure 3.7 shows the PoD plug-in based architecture that supports different job submission systems, like *Portable Batch System* (PBS) [71] and LSF. An SSH plug-in is also provided which can be used when there are no *Resource Management systems* (RMSs) available.

PoD supports both native PROOF and packets-forwarding connections. If the workers are behind a firewall, PoD automatically uses the packet forwarding connection to maintain the PROOF traffic between server and workers. When PoD can connect its workers on xproofd port, a native PROOF connection is set-up. In this last case, it is necessary to open a xproofd port range for incoming connections on the worker nodes. Furthermore, PoD supports reconnections, this means that if there is a crash of the ROOT analysis sessions, users do not need to resubmit PoD jobs. They have to restart ROOT and PoD automatically reconnects to its workers. Worker nodes are on-line until the podagent service is on-line.

The basic components of PoD are the *PROOFAgent* and the *PAConsole*. The *PROOFAgent* is a C++ daemon that manages the communication tunnel between the

3. THE ROOT PARALLEL ANALYSIS FACILITY AND THE DYNAMIC PROOF FRAMEWORK

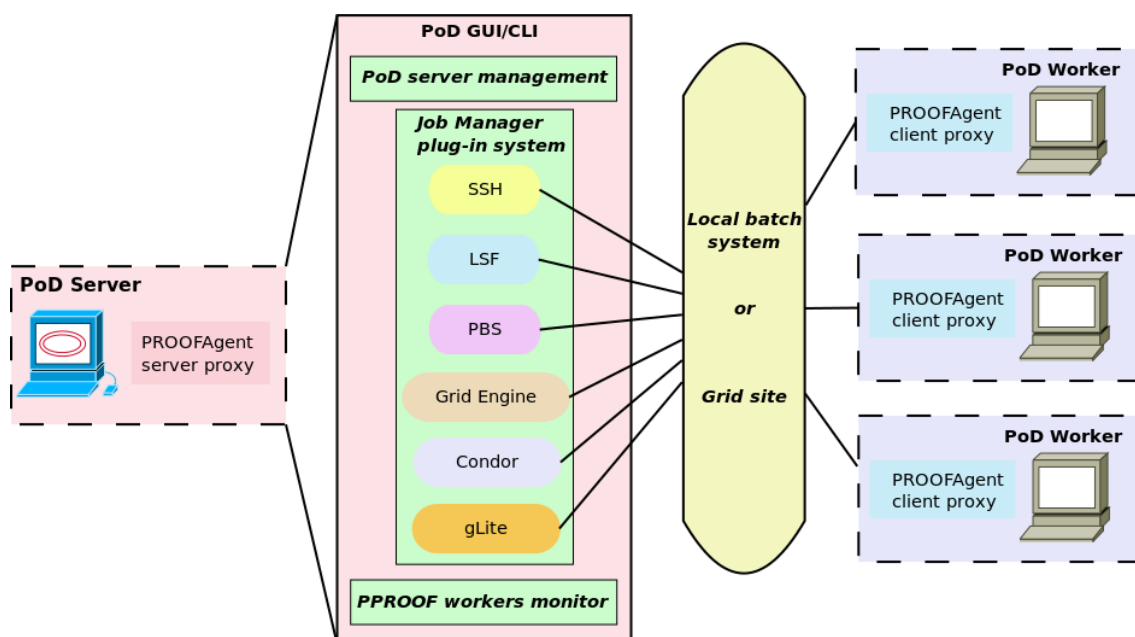


Figure 3.7: Connection between server and workers through the Job-Manager plug-in system part of the PoD GUI/CLI.

PROOF master on the local machine and the PROOF workers on the remote nodes. The *PAConsole* implements the PoD GUI to allow the easy set-up, management and shutdown of the PROOF server that can be a local or a remote server. When the server is remote, an SSH tunnel allows the communication with the users. It is automatically managed by PoD creating a background daemon, which regularly checks the status of the remote PoD server.

The last component of PoD is the *PoDWorker script* that is executed on the remote machines and it is responsible for the workers set-up.

3.4.1 PROOFAgent

The *PROOFAgent daemon* acts as a client Proxy on the remote workers while it becomes a server Proxy on the user machines. This daemon sets up the xproofd communication channel between the server and the workers.

PoD was designed taking into account that worker nodes are blocked for incoming traffic, thus the workers are contacting and connecting to the PoD server. Unlike the PROOF framework, this makes PoD suitable for the set-up of PROOF worker processes

on Grid workers that do not accept incoming traffic.

Furthermore, the client part of the *PROOFAgent* sends its environment information to the part of the server that decides whether the node is suitable to be a PROOF worker.

This is extremely important when the PROOF workers are Grid nodes. In this case, to run a PROOF worker the environment settings must be known as well as the user ID on which the PROOF worker will accept connections. Normally, this information is not known until the job is executed on a remote worker node using a Grid ID.

The *PROOFAgent* is also responsible for the management of bad workers. This means that if a worker fails, also during the analysis session, the *PROOFAgent* will remove it from the list of workers. The user can start analysis when a certain number of workers are on-line. As soon as other nodes are ready, they will be added to the cluster automatically.

Finally, the *PROOFAgent* is monitoring itself and it shuts down the PoD workers after a certain amount of idle time.

3.4.2 PAConsole

The *PAConsole* (Fig. 3.8) is the PoD main GUI that makes it possible to start, stop or monitor the PoD server status.

Using the *PAConsole*, users can access various kind of information such as the PIDs and sockets used by the PoD daemons or the list of the currently available PROOF workers.

In order to support different job submission frontends, the *PAConsole* was designed and implemented as a plug-in based system which allows the use of PoD on different resources such as Grid, Cloud, RMS or simple machines using the SSH access.

The *PAConsole* provides a server and workers information page, a page to set-up preferences, a Grid information page, the jobs monitoring page and a page to visualize the logging information.

3.4.3 PoDWorker script

The *PoDWorker script* is started by a job submission system on the worker resources in order to set-up all the PoD worker services on the worker node. As shown in Figure 3.9, the *PoDWorker script* contains a binary payload attachment, which brings to

3. THE ROOT PARALLEL ANALYSIS FACILITY AND THE DYNAMIC PROOF FRAMEWORK

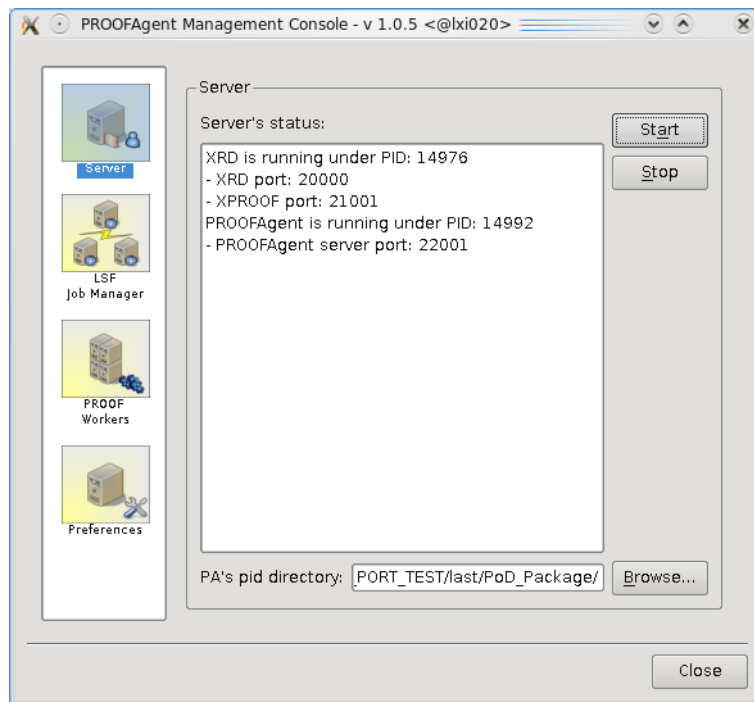


Figure 3.8: The server information page of the *PROOFAgent* management console.

the worker node also the pre-compiled worker binaries for different platforms and the configuration files.

The previous versions of PoD do not include a binary payload attachment and the *PoDWorker script* together with the precompiled workers binaries and configuration files are sent to the worker node with the help of the RMS. To make PoD as automatic as possible, a payload attachment to the *PoDWorker script* has been introduced.

3.4.4 PoD utilities, configuration files and use cases

The installation of PoD provides also predefined configuration values. However, users can change default values in the `PoD.cfg` file to have a specific configuration. Several parameters can be set-up on the `PoD.cfg` file:

- PoD server and worker configurations:
 - Parameters related to file management and locations such as: the PoD server and workers working directory used by PoD modules to store temporary files,

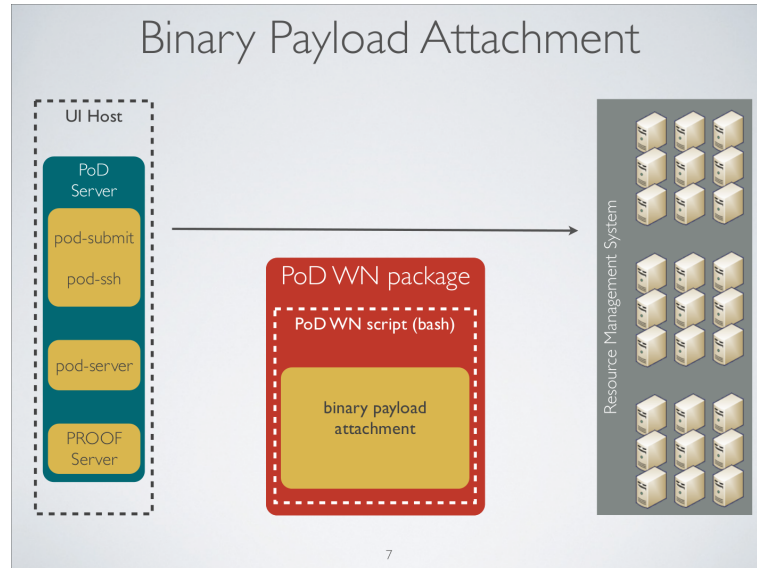


Figure 3.9: A binary payload attachment to the *PoDWorker* script is the strategy adopted to bring pre-compiled workers binaries, configuration files and server information to the worker node.

the path for PoD server and workers log files. It is also possible to decide if PoD should overwrite its log files when starting a new session.

- Parameters related to the server and workers management such as the number of seconds during which the server or the workers can be idle without being shut down automatically.
- A range of ports used by PoD to dynamically assign ports to xproofd when restarting a PoD server or a PoD worker.
- PoD server configuration:
 - Parameters related to the connections such as the port range used by the pod-agent locally on the server host, when in the packet-forwarding mode. Each PROOF client gets its Proxy redirected via the ports from that range. Furthermore, it is possible to set a range of ports used to dynamically assign ports to a pod-agent when restarting a PoD server.
 - Parameters to set-up the use of the packet-forwarding connection even when workers are not behind a firewall.

3. THE ROOT PARALLEL ANALYSIS FACILITY AND THE DYNAMIC PROOF FRAMEWORK

- PoD workers configuration:
 - Parameters to set-up the ROOT installation directory. If no ROOT directory is present, PoD downloads a default, pre-compiled version of ROOT according to the worker node environment.
- Parameters related to the PoD plug-in configurations (LSF, PBF, Condor, SSH, etc). For example, using LSF, a parameter can be specified in order to decide whether the jobs output is sent to the user by mail.

Figure 3.10 shows how PoD can be used to set-up a distributed PROOF cluster on the Grid or on batch systems.

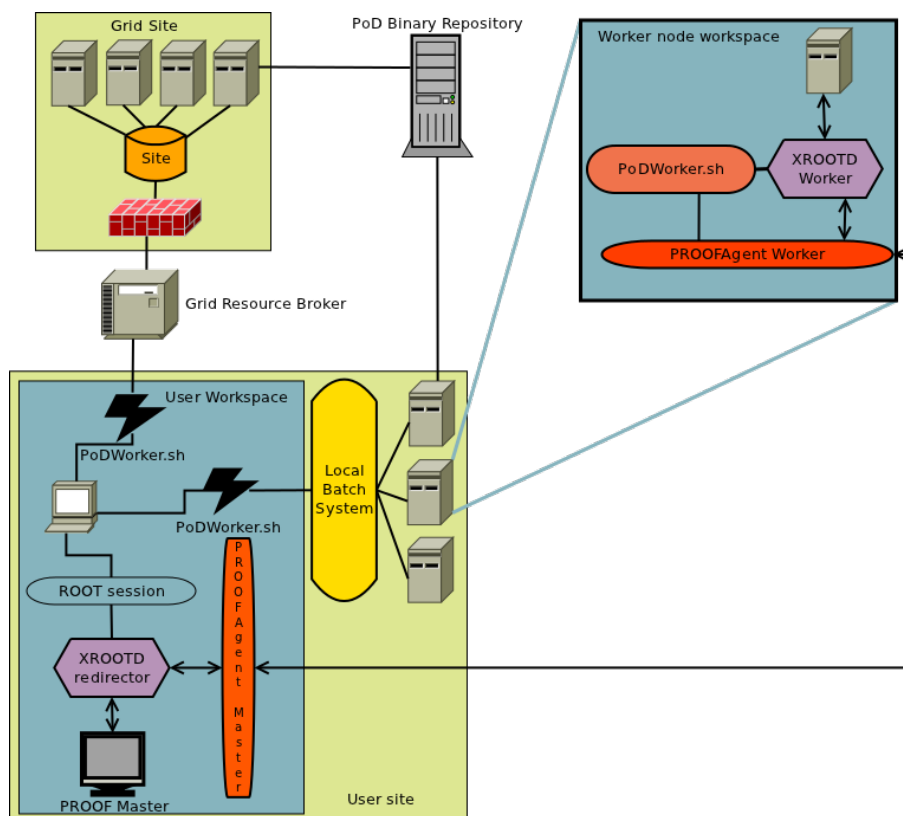


Figure 3.10: The building blocks of the PoD framework.

- 1) As a first step, the user needs to start the PoD server on the user machine that could be local or remote. The *PROOFAgent* server is listening on the PROOF

3.4 The PROOF On Demand framework

port of the xrootd redirector and waits for connection from *PROOFAgent* clients. The user will then need to submit a predefined PoD job to the remote resource that could be an LSF farm or a workload management system. The job will start the *PROOFAgent* client and will configure the environment.

- 2) The *PROOFAgent* client contacts the *PROOFAgent* server and sends its environment information. The *PROOFAgent* server decides if the node is a good candidate to be a worker, accepts the connection and adds the worker to the PROOF configuration file, the client is now a PoD worker.
- 3) When the user obtains the requested number of workers, he can start a ROOT session, connect to the PROOF master and use it for the data analysis.

3. THE ROOT PARALLEL ANALYSIS FACILITY AND THE DYNAMIC PROOF FRAMEWORK

Chapter 4

Enabling PROOF on the AliEn System

In this chapter we describe an architecture to provide interactive and parallel jobs execution in a Grid environment. We explain what has been done in order to make the PROOF framework available to the user as a Grid service.

The local implementation (processing capacity) of Grid systems, including AliEn, relies on batch farms with various types of jobs scheduling (PBS, LSF, OGE, Condor) and does not offer any interactive methods for job execution and control. The choice of using batch processing comes from the fact that the Grid middleware was developed for computing intensive jobs, which may run for a long time before a result becomes available. This approach, which may take many minutes or hours, does not provide interactivity and the user needs to wait for the termination of the jobs to have any information about the execution.

An interactive analysis allows the user to analyse a dataset and have back partial results on a really short time scale. Through an interactive analysis tool, the user must have a feedback on the processing state of the job and should be able to stop and restart an analysis and to select or change the dataset during the analysis process, all while interacting with the job.

In order to increase the speed of data analysis, parallel analysis is usually adopted by the HEP community, either on the Grid or on interactive systems. When a parallel analysis is performed, the task is split into sub-tasks that are executed on many computing nodes at the same time.

A possible approach to provide interactivity and parallel analysis in a Grid envi-

ronment is described in the following paragraphs.

4.1 Benefit of having PROOF on the Grid

Each year the LHC physics detectors and simulations produce about 20 PB of data. These large datasets can be analysed by batch analysis or by interactive analysis.

In the case of the ALICE experiment at CERN, the distributed interactive parallel processing of data on clusters of computers or multi-core machines is provided by PROOF-enabled clusters, called *ALICE Analysis Facilities* (AAFs)[72]. These are mostly used by individual users for fast data analysis.

However, the AAF clusters can hold only a fraction of the 5PB of data accumulated by ALICE yearly. The remaining part of data is held using the AliEn Grid. The extension of the PROOF concept to the Grid paradigm would provide interactivity on the Grid and the possibility for the user to take advantage of a private PROOF cluster. Therefore, the user can use a greater number of processors in comparison to the processors available for the local analysis.

One of the subjects of this thesis is the exploitation of the capability of the Grid to provide more computing power for interactive analysis.

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 the analysis of larger datasets or when the installation of a dedicated cluster is not possible.

As we described in Chapter 2, the Tier-2 centres of the ALICE Computing Model, are computing centres where end-user analysis takes place. Often, Tier-2 infrastructures do not have the possibility to use their resources to set-up statically facilities for interactive and parallel data analysis. An alternative approach is to dynamically allocate some of the available Grid resources for interactive analysis.

The *PoD on AliEn* service allows distributed parallel analysis of large volumes of data on remote nodes while maintaining interactivity.

This service allows the setting up of a private PROOF cluster dynamically within the AliEn system. It enables parallel data analysis on a set of worker machines dynamically chosen from the system in order to execute jobs in a short time scale exploiting data locality, while presenting results to the users with a graphical client interface running on their desktop workstation.

Therefore, using PoD, PROOF can be used directly on Grid-enabled clusters by dynamically assigning interactive nodes on user request. This means that the resources of the Grid are used to set-up a PROOF cluster each time this is needed by the user and they are released as soon as the user has finished. In this case, there is no need to dedicate a cluster to the PROOF analysis facility.

In the following paragraphs we will show two possible and different approaches on how to provide this service within the AliEn Grid.

4.2 Architecture and Implementation

In order to provide a more flexible solution than the static set-up of the PROOF cluster, the PoD framework has been developed at GSI.

As described in Chapter 3, PoD allows a more dynamic use of PROOF, giving the possibility to set-up a local cluster on request on any resource management system.

PoD integrates an automatic process to set-up a PROOF cluster that does not require human intervention unless a change in the default configuration file of PoD is needed. For example, it may be necessary to change the default lifetime of the PoD server or of the PoD workers.

This automatic set-up of PROOF makes PoD a very good candidate for the integration into the AliEn system. Therefore, by using PoD, we have implemented a new service able to provide parallel and interactive analysis on the Grid.

During the design phase of PoD on AliEn several problems were taken into account:

- **Data locality:** in a distributed computing environment the data to be analysed is located at several computing centres spread all over the world. In order to avoid data movement, the analysis tasks must be executed in the computing centre hosting the data. This can be achieved by starting PoD Server and workers at the computing nodes of these centres.
- **Transparency:** the service needs to be transparent to the user who should be able to submit interactive jobs to the AliEn system without having to perform any other operation. The easiest way to achieve this objective is to allow the user to start a PoD Server submitting a request job to AliEn. In this way, the user does not need to perform any special action than a normal job submission.

4. ENABLING PROOF ON THE ALIEN SYSTEM

- **Scalability:** the basic architecture does not put any implicit limitations on the number of computers that can be used in parallel. It is possible to start several PoD Servers spread over the sites of the AliEn Grid; the user can decide how many workers to be used for each PoD Server.
- **Network latency:** in order to avoid connection latency, the PoD Worker nodes should be created on the same site as that of the PoD Server.
- **Easy to set-up:** particular efforts have been made in the system design to automate the set-up of PoD on the Grid.
- **Easy to use:** many of the existing job management components of AliEn are used in the design. In particular, the interaction with the PoD framework is done through a standard job submission.
- **Security:** Grid nodes are firewall protected from incoming connections, this prevents the starting of PROOF workers from the corresponding PROOF master running on the site front-end machine. According to the PoD on AliEn design, these nodes are required to advertise their presence and initiate an outgoing connection towards the PROOF master.
- **Interactivity:** PROOF is an interactive system, while the Grid is based on a batch system where a job starts running after an unpredictable delay with respect to the submission time. In this study, two different solutions to provide interactivity on the Grid are proposed which have lead to the implementation of two different prototypes.

4.2.1 PoD on AliEn: first prototype

The first prototype of PoD on AliEn foresees a modification of the AliEn JobAgent code. It consists of integrating the procedure to initiate the PoD Server in the JobAgent, using the MonALISA system as a trigger to start the PoD Workers. For each site it uses a dedicated AliEn CE for the PoD Server installation.

The architecture of the first prototype of the PoD on AliEn service is shown in Figure 4.1

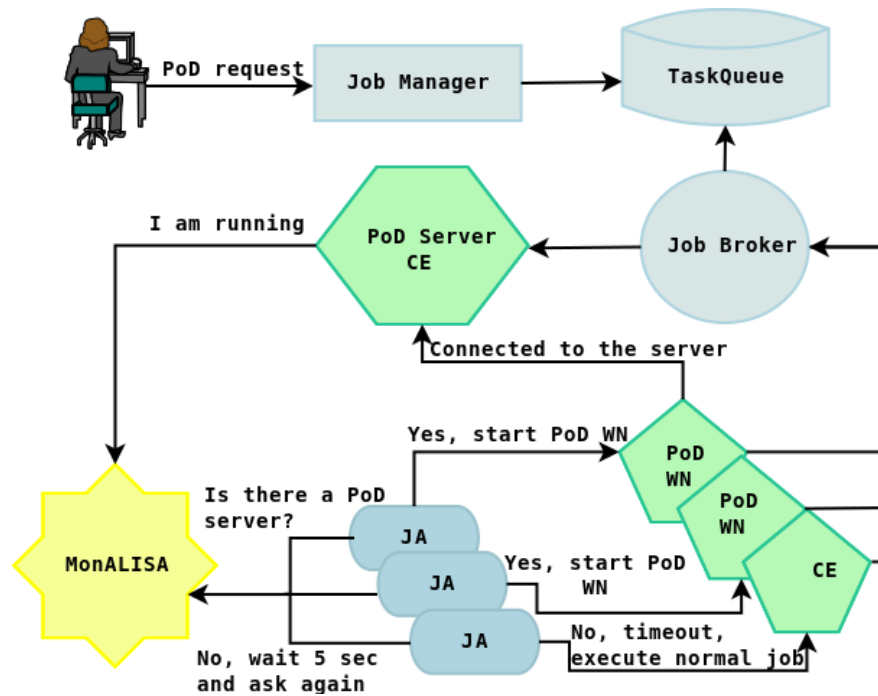


Figure 4.1: The PoD on AliEn architecture: first prototype.

In order to request the set-up of a PoD Server, the user submits a job request on AliEn. When the user request is detected, the dedicated AliEn CE installs the ROOT and PoD packages and executes the `poDStartServer.sh` script on a dedicated AliEn worker node.

Once the server is running, it starts to advertise itself to the MonALISA central repository by publishing information such as the hostname and the port number on which it is listening and the PoD version being used. The publication of the information from the PoD Server to MonALISA is done by using the ApMon API [73]. AliEn JobAgent, by default, checks MonALISA to see if there are any PoD Servers running.

The JobAgent checks MonALISA at regular intervals and for a predefined amount of time. When the timeout expires without finding any PoD Servers, the JobAgent starts to execute a normal AliEn job from the Central Task Queue.

The timeout value and the number of idle seconds before the next check are variables that can be defined by the site administrator in the LDAP configuration database. If an available server is detected, the procedure to create PoD Workers is started. The

4. ENABLING PROOF ON THE ALIEN SYSTEM

system submits to AliEn as many PoD Workers JDLs as are required by the user.

The PoD Worker JDL defines as executable the `CreatePoDWorker.sh` script and it is executed by a normal AliEn CE. It requires the installation of PoD and ROOT packages on the worker node; these packages have to be predefined in the AliEn system.

The `CreatePoDWorker.sh` script creates a default configuration file containing the PoD Server information (hostname, port number and PoD Server version) taken from MonALISA. The configuration file together with other files (for example, pre-compiled workers binaries) is packed creating the PoD Worker node package used to set-up PoD Workers on AliEn WNs.

The `CreatePoDWorker.sh` script executes in turn a PoD WN script which starts the xproof daemon on the worker node and creates the connection between PoD Worker and PoD Server.

The PoD Server stops to send information to MonALISA either when it has acquired the pre-defined set of workers or after a pre-defined amount of time during which it is idle. After this timeout the server is shut down automatically.

This version foresees that the first free PoD Server on the MonALISA list is the one that will execute the next PoD job. In order to reduce the network traffic and the system overhead, it would be better to select the nearest PoD Server. A possible solution to this problem would be to implement a web service with a similar layout of MonALISA (PoDMonALISA) to be started by each VOBox. The PoD Server could send its information to the closest PoDMonALISA service in the environment as well as the `CreatePoDWorker.sh` script could ask for the information from the closest PoDMonALISA service. This design relies on a constant stream of freshly starting Job Agents on the WNs, and each of them attempts to find and connect to an active PoD Server.

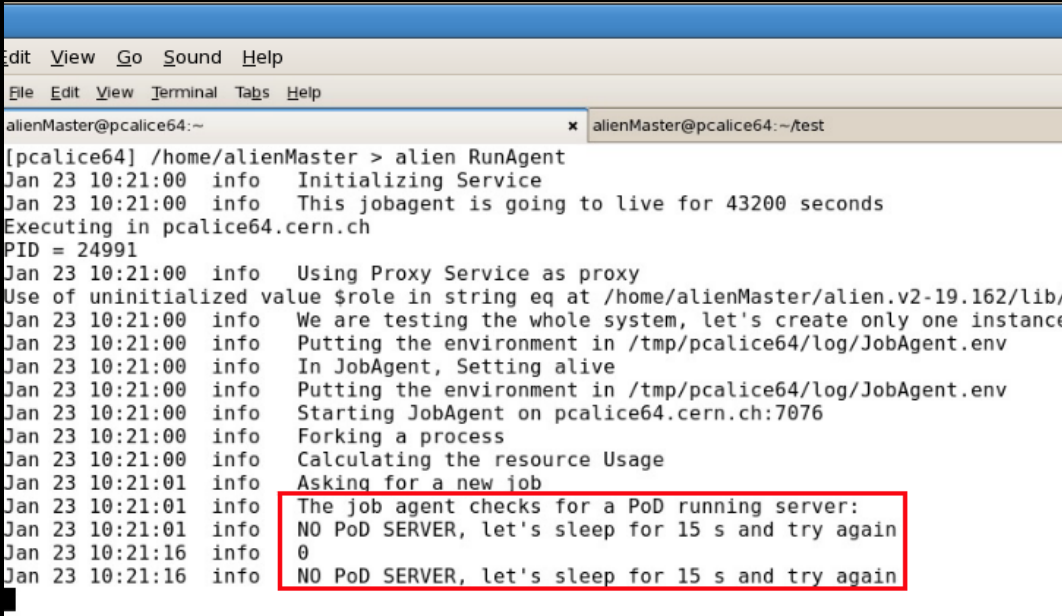
This approach has been implemented at an early stage and then replaced because considered more complicated and time-consuming to set up than the second prototype of PoD on AliEn described later in this Chapter.

4.2.1.1 PoD on Alien: first prototype - workflow

This paragraph shows the workflow of the first prototype, integrating the descriptions of steps with screen-shots of the execution of the PoD on AliEn service.

4.2 Architecture and Implementation

- 1) The user requests a PoD Server submitting a request job on AliEn. The minimum and the maximum number of workers needed and the maximum waiting time to get the maximum number of workers can be specified in the JDL.
- 2) AliEn executes the JDL and starts a PoD Server on a PoD-aware server machine.
- 3) The PoD Server starts to send its information to MonALISA every X seconds.
 - a) If there are no servers, as shown in Figure 4.2 , the AliEn JobAgent waits for a predefined period of time, after which it will execute a normal job from the AliEn Task Queue.

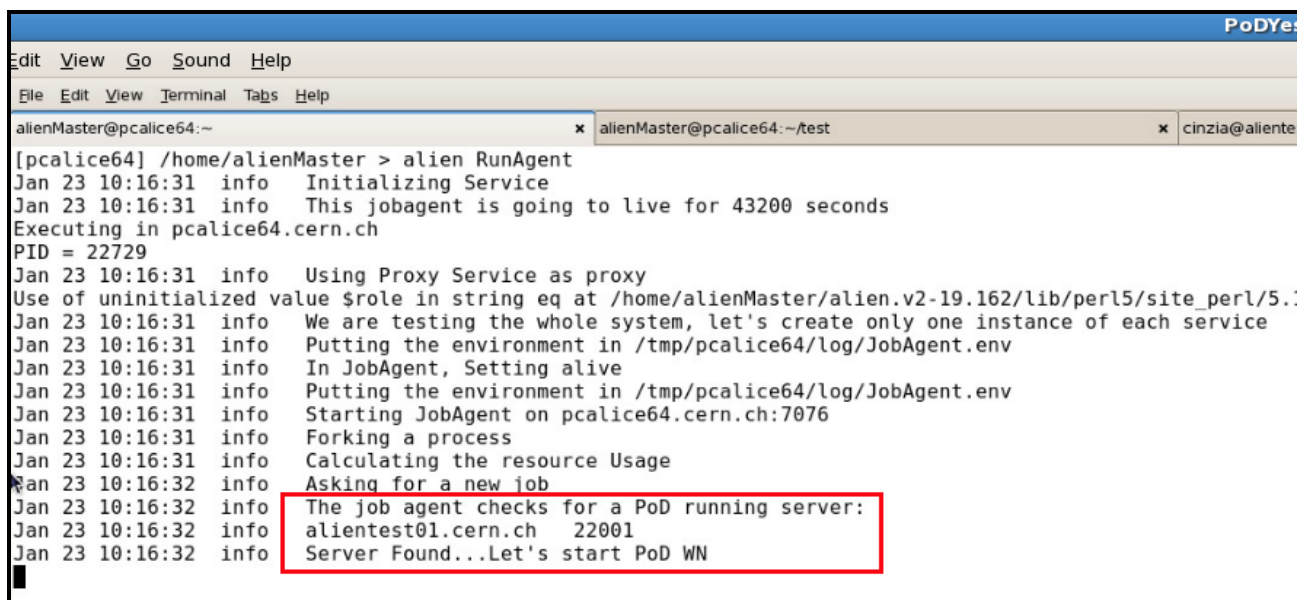


```
alienMaster@pcalice64:~
[pcalice64] /home/alienMaster > alien RunAgent
Jan 23 10:21:00 info Initializing Service
Jan 23 10:21:00 info This jobagent is going to live for 43200 seconds
Executing in pcalice64.cern.ch
PID = 24991
Jan 23 10:21:00 info Using Proxy Service as proxy
Use of uninitialized value $role in string eq at /home/alienMaster/alien.v2-19.162/lib/
Jan 23 10:21:00 info We are testing the whole system, let's create only one instance
Jan 23 10:21:00 info Putting the environment in /tmp/pcalice64/log/JobAgent.env
Jan 23 10:21:00 info In JobAgent, Setting alive
Jan 23 10:21:00 info Putting the environment in /tmp/pcalice64/log/JobAgent.env
Jan 23 10:21:00 info Starting JobAgent on pcalice64.cern.ch:7076
Jan 23 10:21:00 info Forking a process
Jan 23 10:21:00 info Calculating the resource Usage
Jan 23 10:21:01 info Asking for a new job
Jan 23 10:21:01 info The job agent checks for a PoD running server:
Jan 23 10:21:01 info NO PoD SERVER, let's sleep for 15 s and try again
Jan 23 10:21:16 info 0
Jan 23 10:21:16 info NO PoD SERVER, let's sleep for 15 s and try again
```

Figure 4.2: The AliEn JobAgent checks for available PoD Server.

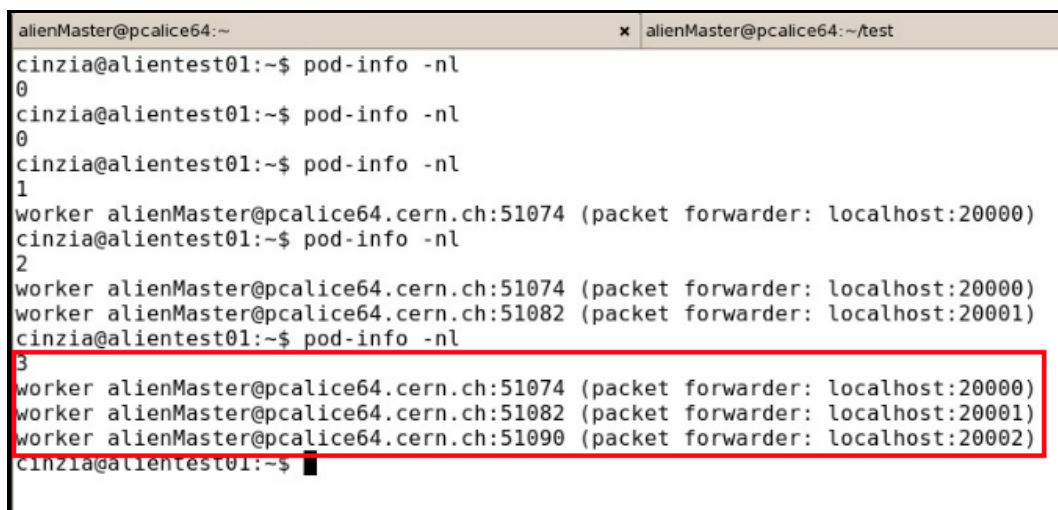
- b) If the PoD Server is available, as shown in Figure 4.3, the procedure to create PoD Workers is started.
- 4) When the PoD Workers are running, as shown in Figure 4.4, they connect to the available PoD Server.
- 5) When the PoD Server gets enough workers, it stops publishing information to MonALISA and it does not accept workers anymore.
- 6) The Server is automatically shut down after a predefined period of idle time.

4. ENABLING PROOF ON THE ALIEN SYSTEM



```
alienMaster@pcalice64:~  
[pcalice64] /home/alienMaster > alien RunAgent  
Jan 23 10:16:31 info Initializing Service  
Jan 23 10:16:31 info This jobagent is going to live for 43200 seconds  
Executing in pcalice64.cern.ch  
PID = 22729  
Jan 23 10:16:31 info Using Proxy Service as proxy  
Use of uninitialized value $role in string eq at /home/alienMaster/alien.v2-19.162/lib/perl5/site_perl/5.1  
Jan 23 10:16:31 info We are testing the whole system, let's create only one instance of each service  
Jan 23 10:16:31 info Putting the environment in /tmp/pcalice64/log/JobAgent.env  
Jan 23 10:16:31 info In JobAgent, Setting alive  
Jan 23 10:16:31 info Putting the environment in /tmp/pcalice64/log/JobAgent.env  
Jan 23 10:16:31 info Starting JobAgent on pcalice64.cern.ch:7076  
Jan 23 10:16:31 info Forking a process  
Jan 23 10:16:31 info Calculating the resource Usage  
Jan 23 10:16:32 info Asking for a new job  
Jan 23 10:16:32 info The job agent checks for a PoD running server:  
Jan 23 10:16:32 info alienest01.cern.ch 22001  
Jan 23 10:16:32 info Server Found...Let's start PoD WN
```

Figure 4.3: The AliEn JobAgent finds an available PoD Server.



```
alienMaster@pcalice64:~ x alienMaster@pcalice64:~/test  
cinzia@alienest01:~$ pod-info -nl  
0  
cinzia@alienest01:~$ pod-info -nl  
0  
cinzia@alienest01:~$ pod-info -nl  
1  
worker alienMaster@pcalice64.cern.ch:51074 (packet forwarder: localhost:20000)  
cinzia@alienest01:~$ pod-info -nl  
2  
worker alienMaster@pcalice64.cern.ch:51074 (packet forwarder: localhost:20000)  
worker alienMaster@pcalice64.cern.ch:51082 (packet forwarder: localhost:20001)  
cinzia@alienest01:~$ pod-info -nl  
3  
worker alienMaster@pcalice64.cern.ch:51074 (packet forwarder: localhost:20000)  
worker alienMaster@pcalice64.cern.ch:51082 (packet forwarder: localhost:20001)  
worker alienMaster@pcalice64.cern.ch:51090 (packet forwarder: localhost:20002)  
cinzia@alienest01:~$
```

Figure 4.4: PoD Workers running.

4.2.2 PoD on AliEn: second prototype

The design of the second prototype does not foresee the use of MonALISA anymore and the PoD Server is not running on a dedicated machine but can be executed by any AliEn CE. Furthermore, all the files containing information about the server and needed to set-up the PoD Workers are now sent to the PoD worker nodes via a `PoDWorker.sh` script to which a payload is attached.

As before, in order to request a PoD Server the user has to submit a request job on AliEn. This request job is a JDL with a specified configuration, it contains the procedure in charge of starting the server (`StartPoDServer.sh`) as an executable file, ROOT and PoD as necessary packages. A minimum and a maximum number of workers and a maximum amount of waiting time to get the maximum number of workers can also be specified in the JDL. An example of JDL is:

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

The architecture of the PoD on AliEn framework is shown in Figure 4.5.

The detection of the request for a PoD Server is a trigger for AliEn to install and start the server on an AliEn CE.

In addition to the procedure of starting the server, the `StartPoDServer.sh` script submits to AliEn as many worker JDLs as are those requested by the user, specifying that the workers have to run on the same site of the PoD Server.

When the PoD Server starts running, it creates the `PoDWorker.sh` script that has a binary attachment containing the PoD configuration file, the PoD version used, a configuration file containing the host name and the port number of the PoD Server machine and the pre-compiled worker binaries. This information is sent to the WN in order to set-up the PoD Worker.

Each worker JDL, as soon as the CE executes it, starts the procedure to create a PoD Worker on the WN.

The execution of each `PoDWorker.sh` script creates a PoD Worker that is connected to the server and it is ready to execute the user's tasks. To go into more details, when this script is running and the set-up of all the environment variables is done on the WN and the payload is unzipped. Afterwards the precompiled binaries are unzipped,

4. ENABLING PROOF ON THE ALIEN SYSTEM

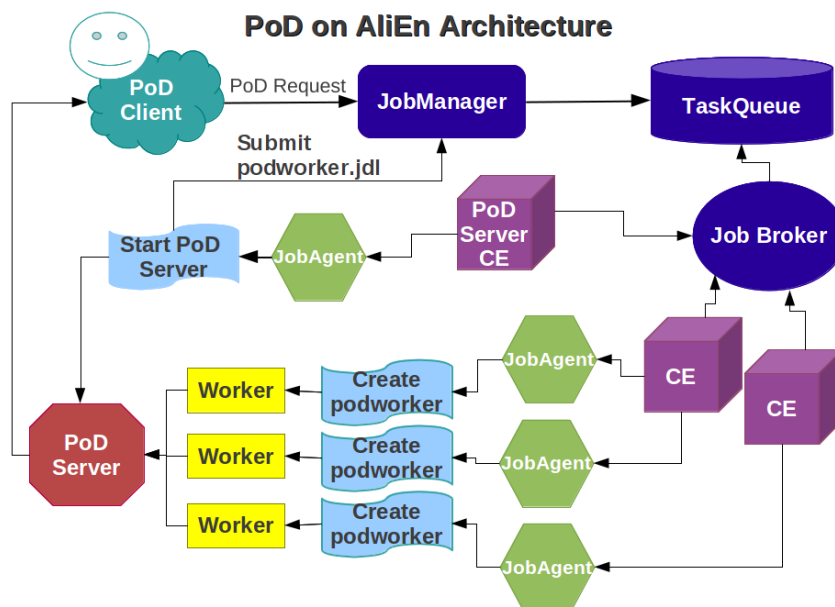


Figure 4.5: The PoD on AliEn architecture: second prototype.

the xproofd process and the pod-agent are started. At this point the PoD Worker is connected to the PoD Server and is ready to be used.

When several users start their own private PoD Server on an AliEn WN, they do not disturb each other. Figure 4.6 shows two different users that submit a PoD Server request to AliEn getting two different servers running. Each user can use the PROOF connection string (E.g. `alisgm15@cern.ch:21001`) to connect to the PoD Server from a ROOT session.

The server is automatically stopped after a predefined amount of idle time. At the same time all the workers are killed automatically and the JobAgents go to the saving step, thereby saving the output files properly in the AliEn system.

This design relies on fewer services than the one described in section 4.2.1, however it requires a dedicated high-priority queue for the PoD Server requests.

4.2.2.1 PoD on AliEn: second prototype - workflow

This paragraph shows the workflow of the second prototype, integrating the descriptions of steps with screen-shots of the execution of the PoD on AliEn service.

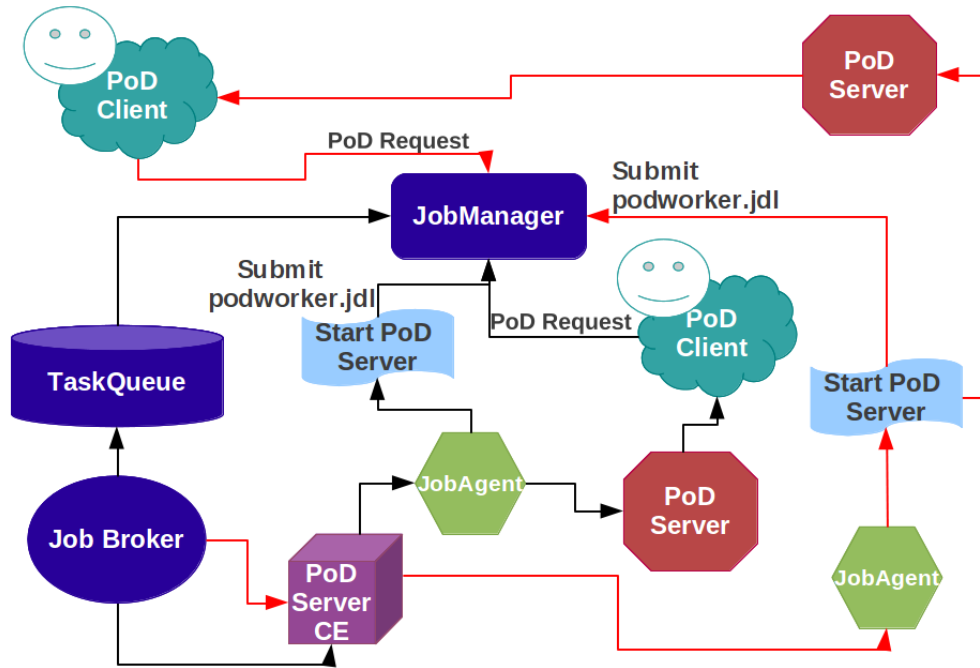


Figure 4.6: The PoD on AliEn architecture provides a private PoD Server for each user.

- 1) The user requests a PoD Server by submitting a job on AliEn in which the needed number of workers can be specified.
- 2) An AliEn CE picks up the job for the execution and it becomes the PoD Server. Figure 4.7 shows the server being started on the AliEn machine `lxbsp0848.cern.ch`. The port number of the `xproofd` process, the port number of the PoD Agent and the PROOF connection string are also shown.
- 3) The request job submits as many JDLs as the number of requested workers. Figure 4.8 shows the submission of one of the 10 workers requested by the user. The submission is automatically made by AliEn after the server JDL execution.
- 4) The workers are started on the same site of the PoD Server and they are connected to the server. Figure 4.9 shows the starting process of a worker.
- 5) The server and workers jobs running can be checked from AliEn, as shown in Figure 4.10. In this figure, the first process ID is the PoD Server while the others are PoD Workers.

4. ENABLING PROOF ON THE ALIEN SYSTEM

```
File Edit View Terminal Tabs Help
Jul 31 13:47:21 info We are supposed to contact the cluster at lxbsp0848.cern.ch:8092
Jul 31 13:47:21 info Got Test: ClusterMonitor is at voalice14.cern.ch:8084
Execution machine: lxbsp0848.cern.ch
Setting the environment for ROOT
Setting ROOTSYS to /home/grid/alismg15/home_cream_630412960/CREAM630412960/alien_installation.29720/al
Setting POD_LOCATION TO /home/grid/alismg15/home_cream_630412960/CREAM630412960/alien_installation.297
Source PoD_env.sh
Generating a default PoD configuration file...
Generating a default PoD configuration file - DONE.
la directory e' /home/grid/alismg15/home_cream_630412960/CREAM630412960/alien_installation.29720/alie
0
WE HAVE DEFINED THE ENVIRONMENT VARIABLE CE_LCGCE = ' '
WNs pre-compiled binaries are missing.
Downloading WNs pre-compiled binaries...
Starting PoD server...
updating xproofd configuration file...
starting xproofd...
starting PoD agent...
preparing PoD worker package...
selecting pre-compiled bins to be added to worker package...
PoD worker package: /pool/grid/alismg15/home_cream_630412960/CREAM630412960/.PoD/wrk/PoDWorker.sh
-----
XPROOFD [23838] port: 21001
PoD agent [23861] port: 22001
PROOF connection string: alismg15@lxbsp0848.cern.ch:21001
```

Figure 4.7: PoD Server started using the AliEn submission process.

```
submit 10 worker
Jul 31 13:45:45 info In checkFileQuota for user: cluzzi, request file size:0
Jul 31 13:45:45 info In checkFileQuota cluzzi: Allowed
Jul 31 13:45:45 info Submitting job '/alice/cern.ch/user/c/cluzzi/bin/StartWorker.sh '...
Jul 31 13:45:45 info There is no time to live (TTL) defined in the jdl... putting the default '6 h
Jul 31 13:45:45 info There is no price defined for this job in the jdl. Putting the default '1.0'
Jul 31 13:45:45 info *** calling PackMan with arguments list -silent -all
Jul 31 13:45:45 info Calling directly getListPackages (list -silent -all)
Jul 31 13:45:45 info Asking the PackMan for the packages that it knows
Jul 31 13:45:45 info Input Box: {LF:PoDWorker.sh.23796}
Jul 31 13:45:45 info OK, all right!
Jul 31 13:45:45 info Command submitted (job 307799373)!! ---> Worker submitted
Jul 31 13:45:45 info Job ID is 307799373 - 0
We have just done the submission of alien login -exec submit podworker.jdl PoDWorker.sh.23796
```

Figure 4.8: Automated worker submission from the PoD Server.

4.2 Architecture and Implementation

```
*** [Wed, 31 Jul 2013 13:46:37 +0200]      +++ PoD Worker START +++
*** [Wed, 31 Jul 2013 13:46:37 +0200]      Current working directory: /pool/grid/alisgm15/hor
*** [Wed, 31 Jul 2013 13:46:37 +0200]      Untar payload...
xpd.cfm
PoD.cfg
version
server_info.cfg
pod-wrk-bin-3.12-Darwin-universal.tar.gz
pod-wrk-bin-3.12-Linux-amd64.tar.gz
pod-wrk-bin-3.12-Linux-x86.tar.gz
*** [Wed, 31 Jul 2013 13:46:37 +0200]      host's CPU/instruction set: amd64
*** [Wed, 31 Jul 2013 13:46:37 +0200]      PoD worker runs on Linux-x86_64
pod-agent v3.12
protocol: v6
Report bugs/comments to A.Manafov@gsi.de
*** [Wed, 31 Jul 2013 13:46:37 +0200]      using ROOTSYS: /home/grid/alisgm15/home_cream_2582
*** [Wed, 31 Jul 2013 13:46:37 +0200]      Attempt to start pod-agent (1 out of 3)
*** [Wed, 31 Jul 2013 13:46:37 +0200]      Attempt to start and detect xproofd (1 out of 10)
*** [Wed, 31 Jul 2013 13:46:37 +0200]      trying to use XPROOF port: 21001
*** [Wed, 31 Jul 2013 13:46:37 +0200]      starting xproofd...
*** [Wed, 31 Jul 2013 13:46:38 +0200]      xproofd is running. pid=[22539] port=[21001]
*** [Wed, 31 Jul 2013 13:46:38 +0200]      starting pod-agent...

[xbsu1347.cern.ch:8091----> PoD Worker started
```

Figure 4.9: Process to start a worker.

```
[aliendb06c.cern.ch:3307] /alice/cern.ch/user/c/cluzzi/ > ps
cluzzi  307794079  R   00:09:06
cluzzi  307799373  R   00:11:07
cluzzi  307800758  R   00:05:02
cluzzi  307801911  R   00:03:01
cluzzi  307803359  R   00:05:05
cluzzi  307804420  R   00:09:04
cluzzi  307805455  R   00:04:03
cluzzi  307806613  R   00:07:01
cluzzi  307807756  R
cluzzi  307808995  R
cluzzi  307809965  R   00:09:05
[aliendb06c.cern.ch:3307] /alice/cern.ch/user/c/cluzzi/ >
```

Figure 4.10: Process list of PoD Server and PoD Workers running on AliEn.

4. ENABLING PROOF ON THE ALIEN SYSTEM

- 6) The user can get and use the PROOF connection string in the ROOT session in order to connect to the remote workers. Figure 4.11 shows the connection of the user to the PROOF server from ROOT using the PROOF connection string `alisgm15@cern.ch:21001`.

```
*****
*
*      W E L C O M E  t o  R O O T      *
*
*   Version   5.34/04   10 January 2013  *
*
*   You are welcome to visit our Web site *
*      http://root.cern.ch                *
*
*****

ROOT 5.34/04 (tags/v5-34-04@48259, Jan 10 2013, 17:15:27 on linuxx8664gcc)

CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0] TProofBench pb("alisgm15@lxbsp0848.cern.ch:21001");
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)
Run description: PROOF at lxbsp0848.cern.ch, 10 workers
Info in <TProofBench::SetOutFile>: using default output file: 'proofbench-lxbsp
root [1] □
```

Figure 4.11: Connection to the PoD cluster done.

4.3 Results

The PoD framework has been successfully integrated in the AliEn system. Figure 4.12 shows the execution of a CPU test on the AliEn PROOF cluster using 10 remote workers.

The performance of the PoD on AliEn prototype has been tested estimating the PoD Workers and PoD Servers start-up latencies. The time to start the server and to obtain the requested workers depends on many factors such as:

- Available job slots on the site.
- Site occupancy rate.
- User priority in AliEn.

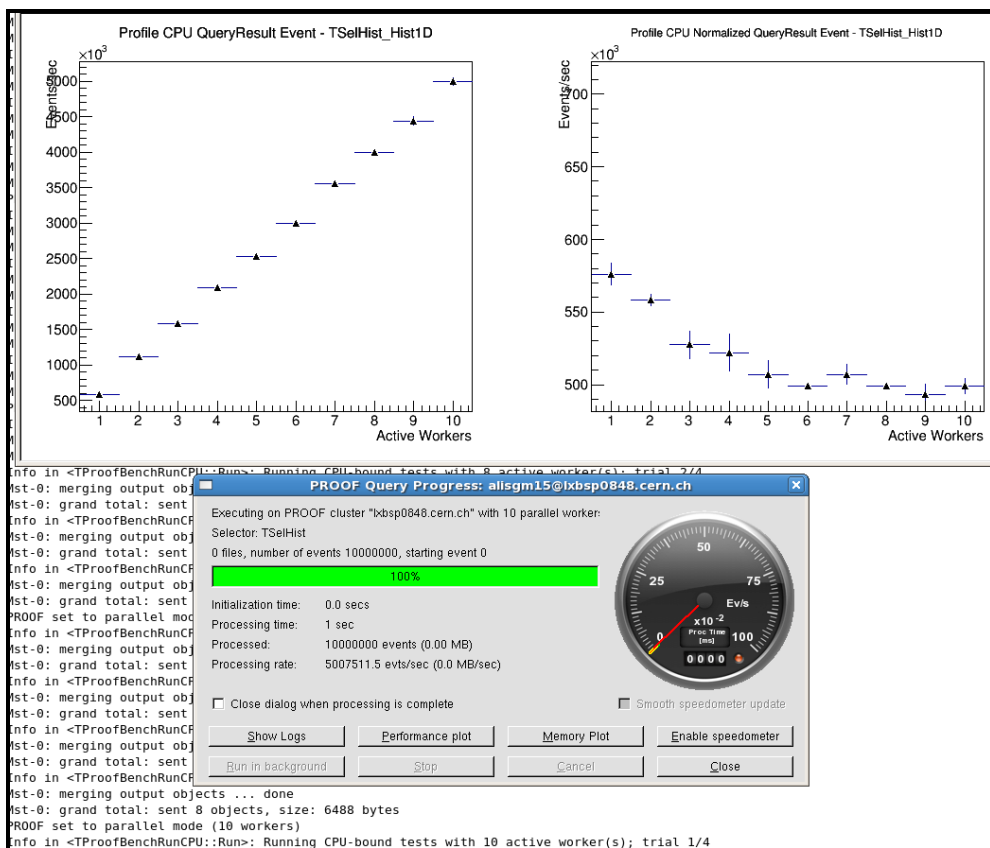


Figure 4.12: CPU test on the AliEn PROOF cluster with 10 workers

4. ENABLING PROOF ON THE ALIEN SYSTEM

The PoD jobs have been treated by AliEn as normal jobs for which the priority of execution is calculated every hour using the following formula:

$$Priority = 100 * Price * (1 - \frac{Running}{MaxRunning}) - \frac{Jobid}{MaxJobid}$$

Price = level of priority, it is set up by the user;

Running = current number of jobs running;

Max Running = maximum number of running jobs per user, it is set up by the administrator;

Jobid = current job id;

MaxJobId = the maximum job id;

The start-up time for a worker is the time interval from when the server starts until the time the worker is running. The start-up of a server is the time interval from when the user job to request the server is submitted until when the PoD Server is running.

Each single test consists of submitting 100 workers for 100 times:

- The start-up time of the first worker is on average 237 seconds (3 min 57 sec).
- The start-up time for 100 workers is on average 1963 seconds (32 min 43 sec), one PoD Worker is set-up each 20 seconds (Fig.4.13).
- The start-up time of PoD Servers is on average 319.07 seconds (5 min 32 sec) (Fig. 4.14).

These tests have been performed using the requirements of running the PoD Server and the PoD Workes on the CERN site. Firstly because this allows to avoid communication latency between server and workers. Secondly because the PoD Server needs to write on a temporary directory during the set-up phase. Currently, this is possible at CERN, while for other sites, it needs to be allowed by the administrators when PoD on AliEn is in production. The start-up time tests will surely improve either when the PoD Server and the PoD Workers are running on the whole AliEn system or if a dedicated and high priority AliEn queue for the PoD jobs scheduling is set-up.

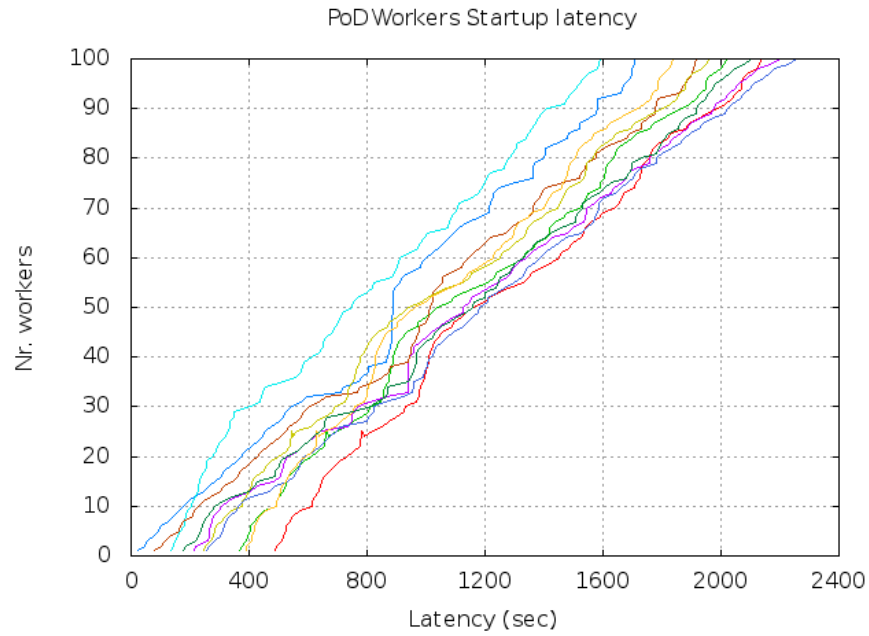


Figure 4.13: PoD Workers start-up latency. For readability the plot shows only 10 out of 100 tests.

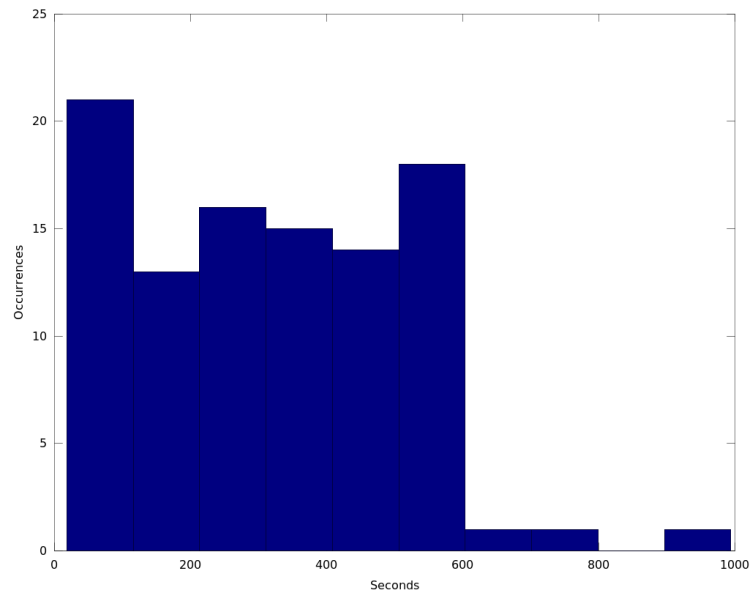


Figure 4.14: Start-up time of PoD Servers.

4.4 Summary

The second prototype of the PoD on AliEn service is easy to use, maintain and install.

Using of PoD on AliEn is transparent from the user point of view. After a JDL submission, the user will be granted of all the advantages of using distributed PROOF for data analysis.

From the AliEn administrator point of view, the installation and maintenance are simple. It is only necessary to define the PoD and ROOT packages in the AliEn system and to make available a predefined PoD request JDL which can be subsequently modified by the users.

Both the procedure to start the server and the procedure to create PoD Workers are managed by means of the job submission system in AliEn therefore no additional installation actions are required by the site administrator.

The CAD interface for the ROOT geometry modeller

5.1 The ROOT geometry modeller

ROOT provides a geometry package [74] used to build, browse and visualize detector geometries. This tool is extremely important for the simulation of detectors.

The ROOT geometry architecture is based on the concept of *Volumes*, which are the basic components used for building the logical hierarchy of the geometry.

A volume can be seen as a container, or a virtual volume used to group and position together other volumes, or a real volume defining geometric objects having a given shape and material and containing a list of nodes (positioned volumes).

The top volume of the geometry contains all the other volumes and it defines the global reference system in which the others are positioned. Except to the largest, each volume is associated with a medium which can be a mixture of different materials.

Geometries are built by positioning volumes inside the others using spatial transformations with respect to the mother (container volume) reference system. Two volumes have to be linked together providing the relative transformation matrix. The positioned instance of a volume inside a container volume is called *Node*. A Node is not defined by users but it is created automatically as a result of adding one volume inside others. Each node points to a volume, which in turn points to a list of nodes. Each element of the geometry is made by a pair volume-node. Each volume can have one or more

5. THE CAD INTERFACE FOR THE ROOT GEOMETRY MODELLER

daughters and nodes can be viewed as bi-directional links between containers and contained volumes.

The same volume can be positioned several times in the geometry and it can be divided into new volumes. Since volumes can be replicated, the resulting structure is a graph where every volume is a branch of the graph. A node is unique and identified by a complete branch of nodes defined up to the top node in the geometry. Its global transformation matrix can be obtained by aggregating all the local transformations in its branch.

The hierarchy defined by nodes and volumes is called *the logical graph* while the extension of the logical graph by all possible paths is defined as *the physical tree*, a tree structure where all nodes are unique objects.

Assemblies are a special kind of volumes, used to manage structures of positioned volumes that have to be grouped and handled together. This is useful when, for example, the structure has to be replicated in several parts of the geometry or when the grouped volumes needs to represent a single object, too complex to be described by a primitive shape.

The geometrical transformations define the position of a local volume with respect to its container. If T is a transformation used for positioning daughter volumes, then: $MASTER = T * LOCAL$.

T is used to perform a local to master conversion, while T^{-1} for a master to local conversion.

The geometrical transformations are also used to define the global transformation of a given object in the geometry; it represents the aggregation of all the local transformations in a specific branch.

A transformation is defined as a 4x4 matrix enclosing a rotation, a translation and a scale (5.1).

Volumes are built using primitive shapes defined by ROOT (box, sphere, tube) or combining them using boolean operations such as union or intersection. Geometries can be saved as a C++ macro or in a ROOT file.

<p>(a) Rotation</p> $\begin{vmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$	<p>(b) Translation</p> $\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{vmatrix}$
<p>(c) Scale</p> $\begin{vmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$	<p>(d) Inverse rotation</p> $\begin{vmatrix} r_{11} & r_{21} & r_{31} & 0 \\ r_{12} & r_{22} & r_{32} & 0 \\ r_{13} & r_{23} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$
<p>(e) Inverse translation</p> $\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -t_x & -t_y & -t_z & 1 \end{vmatrix}$	<p>(f) Inverse scale</p> $\begin{vmatrix} 1/s_x & 0 & 0 & 0 \\ 0 & 1/s_y & 0 & 0 \\ 0 & 0 & 1/s_z & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$

Table 5.1: Transformation matrices: rotation, translation, scale and their inverse.

5.1.1 ROOT Shapes

Shapes are the geometrical objects that, when associated with a material or mixture, create a volume. They define the local coordinate system of the volume.

Materials are defined by single elements with given atomic mass and charge, while mixtures are derived from the material class and represent combinations of elements.

The modeller provides a set of 20 basic shapes called primitives (boxes, general trapezoids, tubes etc) and classes to create shapes (composite shapes) as a result of boolean operations between primitives. The composition operation can be recursive allowing the creation of a huge number of different shape topologies.

5.2 The TGeoCad Interface

For high-energy physics experiment, in addition to the data analysis, a very high precision in the description of the detector geometry is essential to achieve the required performances.

5. THE CAD INTERFACE FOR THE ROOT GEOMETRY MODELLER

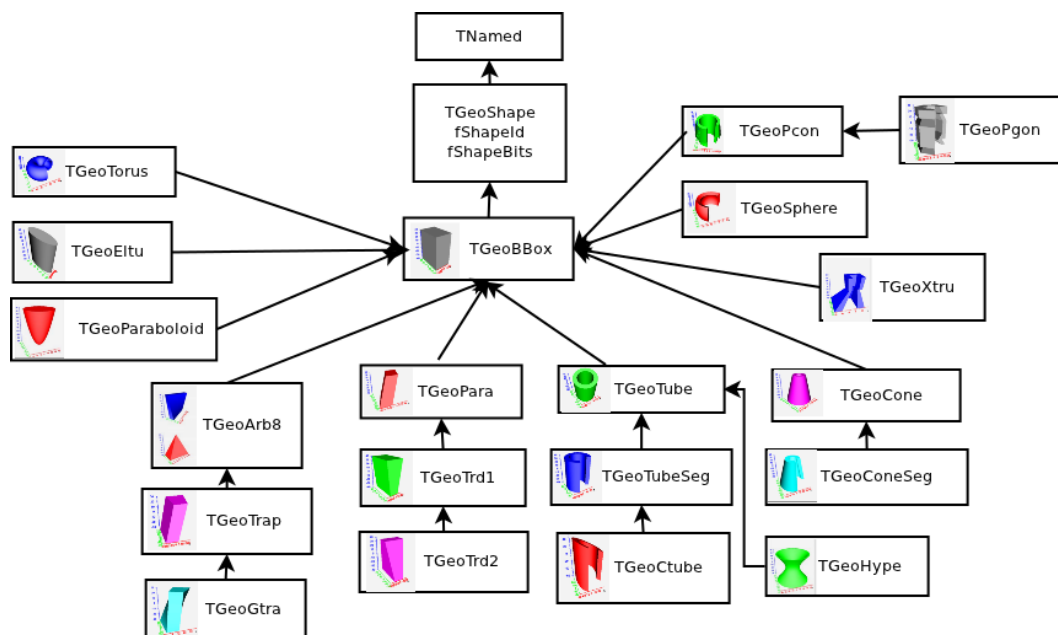


Figure 5.1: The ROOT primitive shapes inheritance schema.

The physicists carrying out the Monte Carlo Simulation of the detector need to strictly collaborate with the engineers working on the mechanical design of the detector. Often, this collaboration is complicated because of different and incompatible software.

The necessity to improve the level of communication between physicists and engineers led to the implementation of an interface between the ROOT simulation software and the CAD systems. The *TGeoCAD Interface* is able to convert ROOT files into the STEP (ISO 10303) [75] format which can be imported and used by many CAD systems.

The incompatibility between the ROOT framework and the CAD systems comes from the different mechanisms adopted to represent shapes. In a CAD system shapes are treated as *Boundary Represented Objects (BREPS)*. They are made up of a set of basic bounded surfaces (plane, conical or spherical). Boundary representation models are based on two elements: topology and geometry. The topology gives information about the relationships between the main topological items, i.e. faces (bounded portion of a surface), edges (bounded pieces of a curve) and vertices (points). The geometry describes the integration of these elements in space. Each solid is placed by a transla-

tion/rotation matrix in an assembly of many solids, the different assemblies forming a complete hierarchy of geometrical objects.

On the other hand, ROOT adopts the *Constructive Solid Geometry (CSG)* model. According to this concept: each solid is defined directly as a three-dimensional primitive and created out of a set of simple shapes like cuboids, cylinders, prisms, pyramids, spheres. Complex shapes are created by using boolean operation between simple shapes. Each shape can then be placed by translation/rotation matrices in assemblies. For more information about the solid modelling see [76].

Another substantial difference between ROOT and CAD systems is the hierarchy of assemblies. In a CAD system the basic unit is a solid part that can have an arbitrary number of nested assemblies. A ROOT geometry is composed by a number of volumes. The largest one is used as a container for all other volumes in the detector geometry. Each created volume is placed inside the previous one, called the mother volume. A volume is composed by its shape and all its physical characteristics such as the material of the volume and any sensitive detector elements, for example the magnetic field.

The main task of the TGeoCad Interface is to analyse the ROOT shape representation, extract all the information about the solid and reproduce it in a BREPS format using the *OpenCASCADE Technology* [77] libraries. This also provides the possibility to write BREPS assemblies in a STEP file.

5.2.1 The STEP Standard

STEP is the acronym for: *STandard for the Exchange of Product model data*. It is the ISO 10303 Standard intended to handle a wide range of product-related data including electronic and mechanical characteristic covering the entire life cycle of a product through design, analysis and planning to manufacture.

The STEP format was designed to provide a common set of CAD/CAM tools for the needs of most companies. To achieve this goal, a common neutral format for the file exchange has been adopted. This has led to a reduction in the number of translators required.

Finally, the exchange process is implemented using the file approach. According to this, as a first step, the data is translated from the data format of the originating system into the ISO 10303 format (neutral ASCII file). In a second step, the data is translated into the format of the receiving system.

5. THE CAD INTERFACE FOR THE ROOT GEOMETRY MODELLER

The entities to be exchanged using STEP and their relationships, are defined in schemas written in *EXPRESS* [78], an information modelling language.

5.2.1.1 Components of STEP

The components of the STEP standard are divided into several series of parts. Each part series contains one or more types of ISO 10303 parts. Figure 5.2 provides an overview of the architecture of the STEP standard.

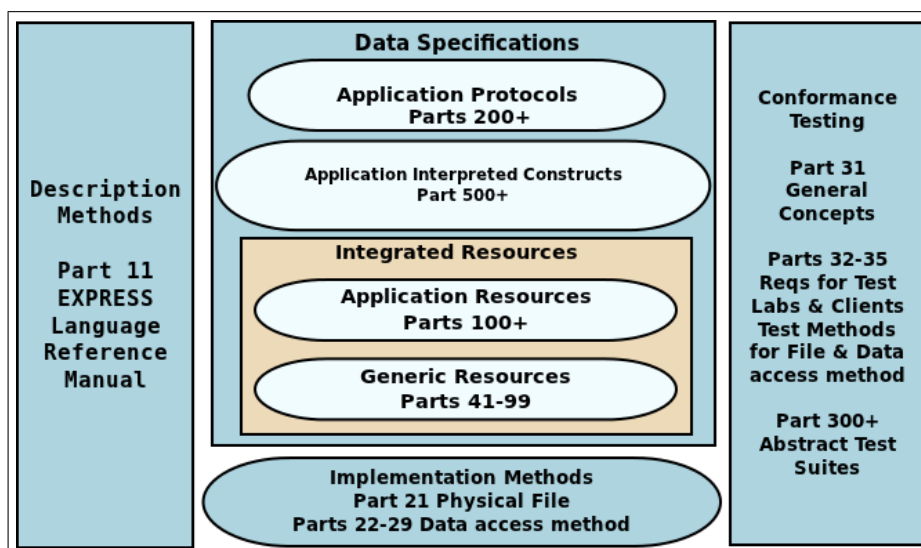


Figure 5.2: STEP standard components description with the specific parts numbers.

The *description method* series of STEP parts are mechanisms used for specifying the data constructs of STEP such as the formal data specification language developed for STEP, known as EXPRESS.

The *implementation method* series are standard implementation techniques to define how the data constructs previously specified using the STEP description methods are mapped to a specific implementation method. This series includes the physical file exchange structure, the standard data access interface, and its language bindings. Separating the data specification from the implementation method enables upward and downward compatibility of implementations of STEP.

The *conformance testing* is covered by two series of 10303 parts: conformance testing methodology and framework, describing how testing of implementations of various

STEP parts is accomplished and abstract test suites which are test cases used for conformance testing of an implementation of a STEP application protocol. Each abstract test case specifies input data to be provided for the implementation under test, along with information on how to assess the capabilities of the implementation.

The final major component of the STEP architecture are the *data specifications*. The constructs within the *integrated resources* are the basic semantic elements used for the description of any product at any stage of the product life cycle.

Application protocols (APs) are the implementable data specifications of STEP. APs include an EXPRESS information model that satisfies the specific product data needs of a given application context and they can be implemented using the implementation methods.

Many of the components of an application protocol are intended to document the application domain in application specific terminology in order to facilitate the review of the application protocol by domain experts.

The Open CASCADE technology gives the possibility to create a file according to the STEP parts AP203 and AP214.

5.2.1.2 Geometry Data Exchange: AP203 - AP214

The *Configured-controlled design protocol* (AP203) manages the transfer of product shape models, assembly structure as well as configuration control information such as part versioning and release status.

The assembly models that can be exchanged using AP203 are collections of positioned and oriented part models.

The *Core data for automotive mechanical design process protocol* (AP214) defines core data for automotive mechanical design processes. It is an extension of AP203 and it also manages manufacturing tools, tolerance data, 2D drawing, and product data management information.

5.2.2 The Open CASCADE Technology

The *Open CASCADE Technology* (OCCT) is an open source software development platform written in C++. It has been designed for rapid production of sophisticated domain-specific design applications.

OCCT allows to develop applications dealing with two or three-dimensional (2D

5. THE CAD INTERFACE FOR THE ROOT GEOMETRY MODELLER

or 3D) geometric modelling in general-purpose or specialized *Computer Aided Design (CAD)* systems, manufacturing or analysis applications, simulation applications, or illustration tools.

The OCCT libraries are grouped into six modules shown by the blue part of Figure 5.3, those used by the TGeoCad Interface are listed in the following sections.

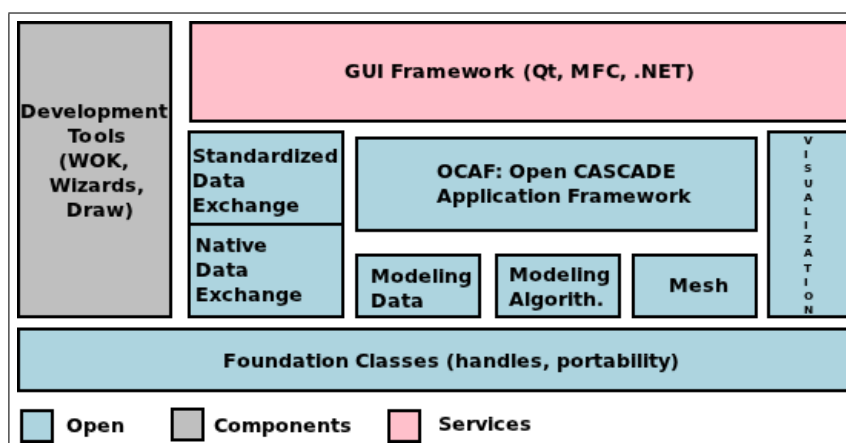


Figure 5.3: The OCCT architecture.

5.2.2.1 Foundation Classes

The Foundation Classes module includes general-purpose services such as kernel classes and math utilities. The kernel classes implement primitive types, strings and various types of quantities, automated management of heap memory, exception handling and classes for manipulating data collections.

The maths utilities provide:

- Description of elementary geometric shapes such as points, vectors, lines, circles and conics, planes and elementary surfaces.
- Means for positioning geometry in space or on a plane using an axis or a coordinate system.
- Definition of geometric transformations such as translation, rotation and symmetries.

5.2.2.2 Modelling Data

The modelling data module provides data structures to represent 2D and 3D geometric and topological models. The topological library allows to build pure topological data structures and to define relationships between simple geometric entities. It gives the possibility to model complex shapes as assemblies of simpler entities.

The abstract topological data structure allows to describe shapes which can be divided into the following component topologies:

- Vertex: a zero dimensional element (point).
- Edge: an element created from a curve and vertices (1D).
- Wire: a set of edges connected by their vertices.
- Face: part of a surface bounded by a number of closed wires (2D).
- Shell: set of faces connected by some of the edges of their wire boundaries.
- Solid: part of 3D space limited by closed shells.
- Compsolid: set of solids connected by their faces.
- Compound: group of any type of topological object.

5.2.2.3 Modelling Algorithm

The modelling algorithm module provides geometric and topological algorithms used in modelling for:

- Creating vertices, edges, faces, solids.
- Building primitive objects such as boxes, wedges and rotational objects.
- Using sweeping operations such as linear, rotational or general sweep and boolean operations.

5.2.2.4 Extended Data Exchange

The Extended data exchange (XDE) module provides tools to write STEP (AP203, AP2014) file from XDE format allowing software based on Open CASCADE to exchange data with various CAD software.

5. THE CAD INTERFACE FOR THE ROOT GEOMETRY MODELLER

5.2.2.5 Application Framework

The Open Cascade Application Framework (OCAF) is a Rapid Application Development (RAD) framework used for specifying and organizing application data according to which data structure are organized by mean of a reference model. It is based on an application/document architecture and provides an infrastructure to attach any kind of data to any topological element by mean of a reference implemented in the form of label. Application data such as a shape itself or a shape location are attached to these labels as attributes.

The OCAS application is used to manage an XDE document which is a container for the reference keys composed by a set of labels organized in a tree structure (Fig. 5.4). Each label has a tag expressed as an integer value and it is identified by a string build by concatenation of tags from the root of the tree to the label node, for example [0:1:2].

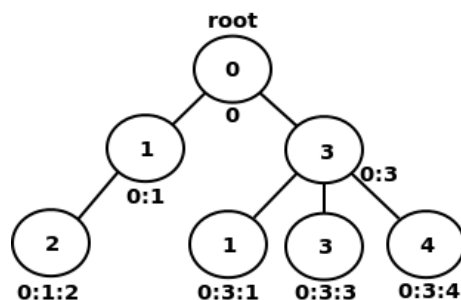


Figure 5.4: The OCCT tree of labels.

5.2.3 Details of the conversion

The TGeoCad Interface has different ways of reproducing a ROOT shape in the Open CASCADE platform:

- Shapes created step by step starting from points (edge, wire, face, shell and solid) such as box, parallelepiped, trapezoid etc.
 - For example, the ROOT TGeoTrd1 shown in Figure 5.5, is translated by creating edges from points, wires from edges, faces (planar surfaces) from wires, shells from faces and solid from shells.

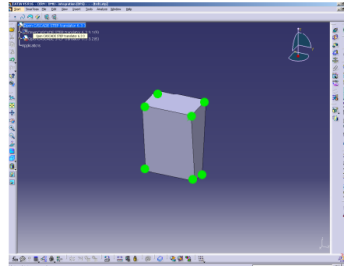


Figure 5.5: The TGeoTrd1 represented in STEP format and visualized by the CATIA software.

- Shapes created using OCCT capabilities for solid primitives creation and boolean operations such as tubes, cones, spheres.
 - As shown in Figure 5.6, the TGeoCone is created using the OCCT libraries which giving the height and radius information allow to create inner and outer cones. Using boolean operations the inner cone is subtracted from the outer cone.

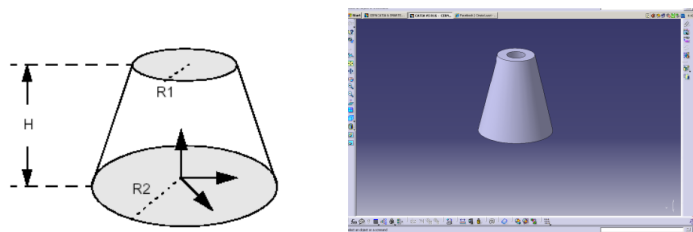


Figure 5.6: a) OCCT cone representation; b) A TGeoCone translated in a STEP file and loaded in CATIA.

- Shapes created by using modelling algorithms (extrusions, revolutions, lofts) applied to basic geometries such as hyperboloids.
- Composite shapes created using OCCT boolean operations between two or more shapes.
 - The TGeoCompositeShape shown in Figure 5.7 is obtained as result of boolean operations applied to a TGeoBox, a TGeoTube and a TGeoPgon.

5. THE CAD INTERFACE FOR THE ROOT GEOMETRY MODELLER

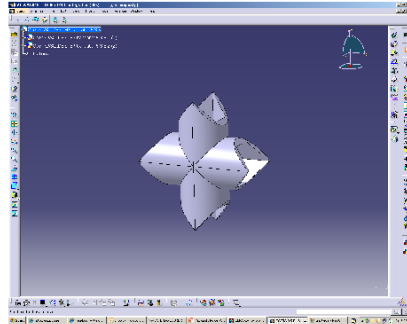


Figure 5.7: A TGeoComposite shape translated in a STEP file and loaded in CATIA.

The conversion method is selected according to the type of each shape. When it is possible, the OCCT libraries for solid primitives are used, sometimes combined with the OCCT modelling algorithms. When there is no suitable OCCT library for the creation of a specific solid, the creation is done starting from point.

5.2.4 Structure of the interface

The TGeoCad Interface has been implemented as a module of ROOT (Fig. 5.8). It uses the OCCT libraries in order to translate a ROOT geometry file in a STEP file.

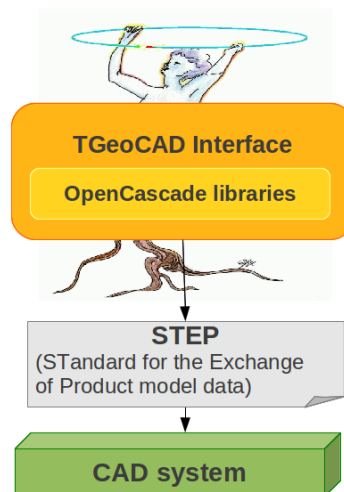


Figure 5.8: The TGeoCad Interface.

The TGeoCad Interface is composed by three main classes:

- `TGeoToStep`: a `TGeoObject` based class that takes a `TGeoManager` pointer to a ROOT geometry and produces the geometry STEP file.
- `TGeoToOCC`: implements all the methods shown in the Table 5.2 to translate each ROOT shape in the corresponding OCCT shape.

ROOT	TGeoToOCC	ROOT	TGeoToOCC
<code>TGeoBBox(..)</code>	<code>OCC_Box(..)</code>	<code>TGeoTrd2(..)</code>	<code>OCC_Trdr(..)</code>
<code>TGeoSphere(..)</code>	<code>OCC_Sphere(..)</code>	<code>TGeoTubeSeg(..)</code>	<code>OCC_Tube(..)</code>
<code>TGeoArb8(..)</code>	<code>OCC_Arb8(..)</code>	<code>TGeoCtub(..)</code>	<code>OCC_Cuttub(..)</code>
<code>TGeoConeSeg(..)</code>	<code>OCC_Cones(..)</code>	<code>TGeoTube(..)</code>	<code>OCC_TubeSeg(..)</code>
<code>TGeoCone(..)</code>	<code>OCC_Cones(..)</code>	<code>TGeoPcon(..)</code>	<code>OCC_Pcon(..)</code>
<code>TGeoPara(..)</code>	<code>OCC_ParaTrap(..)</code>	<code>TGeoTorus(..)</code>	<code>OCC_Torus(..)</code>
<code>TGeoGtra(..)</code>	<code>OCC_ParaTrap(..)</code>	<code>TGeoPgon(..)</code>	<code>OCC_Pgon(..)</code>
<code>TGeoTrd1(..)</code>	<code>OCC_Trdr(..)</code>	<code>TGeoEltu(..)</code>	<code>OCC_Eltu(..)</code>
<code>TGeoHype(..)</code>	<code>OCC_Hype(..)</code>	<code>TGeoXtru(..)</code>	<code>OCC_Xtru(..)</code>
<code>TGeoComposite(..)</code>	<code>OCC_Composite(..)</code>		

Table 5.2: Methods implemented to translate ROOT shapes in OCCT shapes.

- `TOCCToStep`: reproduces the ROOT geometry tree (mother-children relationship) in the XDE document and writes it to the STEP file.

5.2.4.1 TOCCToStep class

Firstly, an OCAF application is created in order to manage the XDE document, afterwards shapes are created and memorized on the XDE document without reporting any information about the relationship between shapes.

As shown in Figure 5.9, starting from the top of the ROOT geometry tree the `OCCShapeCreation(..)` method translates each ROOT shape in the OCCT format using all the methods shown in the Table n.5.2.

For each translated shape a new label is created and memorized into the XDE document. The correspondence shape-label is also stored in a map of volumes and labels in order to keep memory of the shapes already translated.

In this way, a check is done for each shape taken from the ROOT file. If the shape-label correspondence is present in the map, it means that the shape has already been translated then the label is copied from the map to the XDE document using the up-to-date location of the current ROOT shape. If there is no correspondence present in

5. THE CAD INTERFACE FOR THE ROOT GEOMETRY MODELLER

the map, the shape needs to be translated.

Secondly, Figure 5.10 shows the `OCCTreeCreation(..)` method which starts to

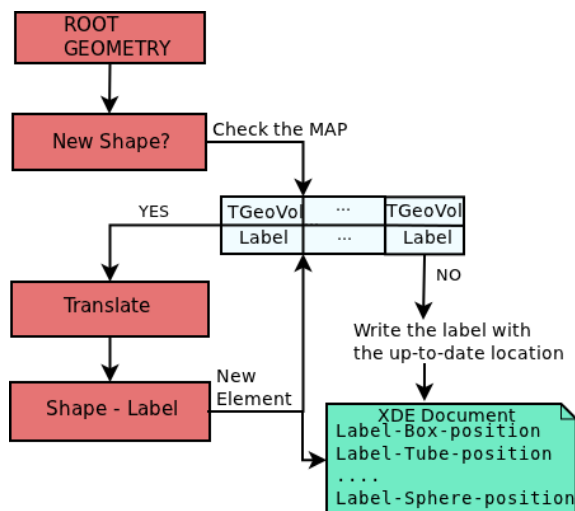


Figure 5.9: The `OCCShapeCreation(..)` method.

browse the geometry and, for each node from the end to the top of the ROOT physical tree, finds the mother and daughter label references in the map. The daughter location matrix is obtained from the ROOT file. These parameters are then used to calculate the position of the daughter with respect to the mother. The connection between the daughter label and the mother label is then created, resulting in a new label, which is added to the XDE document. The OCCT shapes are added, step by step, to an

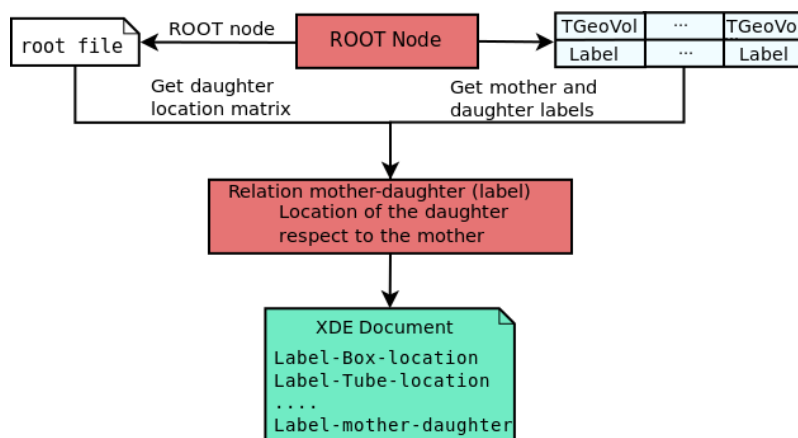


Figure 5.10: The `OCCTreeCreation(..)` method.

assembly entity where each of its sub-shapes defines a sub label so that each node of the assembly refers to its sub-shapes. At last, the XDE document containing the whole geometry is written to the STEP file. Figure 5.11 shows an example of STEP file while Figure 5.12 shows the whole architecture of the TGeoCAd Interface.

```

#3107 = SHAPE_DEFINITION_REPRESENTATION(#3108,#2783);
#3108 = PRODUCT_DEFINITION_SHAPE('', '#3109);
#3109 = PRODUCT_DEFINITION('design', '#3110,#3113);
#3110 = PRODUCT_DEFINITION_FORMATION('', '#3111);
#3111 = PRODUCT('bar1', 'bar1', '#3112);
#3112 = MECHANICAL_CONTEXT('#2,'mechanical');
#3113 = PRODUCT_DEFINITION_CONTEXT('part definition', '#
2, 'design');
#3114 = CONTEXT_DEPENDENT_SHAPE_REPRESENTATION(#3115, #3117);
#3115 = ( REPRESENTATION_RELATIONSHIP('', '#2783.#1551)
REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION(#3116)
SHAPE_REPRESENTATION_RELATIONSHIP() );
#3116 = ITEM_DEFINED_TRANSFORMATION('', '#11,#1568);
#3117 = PRODUCT_DEFINITION_SHAPE('Placement', 'Placement of an
item',
#3118);
#3118 = NEXT_ASSEMBLY_USAGE_OCCURRENCE('14', '=>[0:1:1:5]', '#
1546,#3109
,S);

```

Relationship
definition
between labels
#2783 and #1551

Figure 5.11: Relationship between two shapes reproduced in the STEP file.

5.3 Technical details

In order to have the TGeoCAd Interface available in ROOT, Open CASCADE must be installed and ROOT needs to be compiled using the configuration options:

```

./configure --enable-geocad;
--with-occ-incdir: location of OpenCascade inc files
--with-occ-libdir: location of OpenCascade lib files

```

The TGeoCAd has to be used from ROOT by loading a geometry in memory, creating an object of the TGeoToStep class and calling the method CreateGeometry(). A STEP file called geometry.stp containing the geometry will be created on the current directory.

```

root[0] gSystem->Load("libGeoCAd.so");
root[1] .x rootttest.C
root[2] TGeoToStep *myStep = new TGeoToStep (gGeoManager);
root[3] myStep->CreateGeometry();

```


5. THE CAD INTERFACE FOR THE ROOT GEOMETRY MODELLER

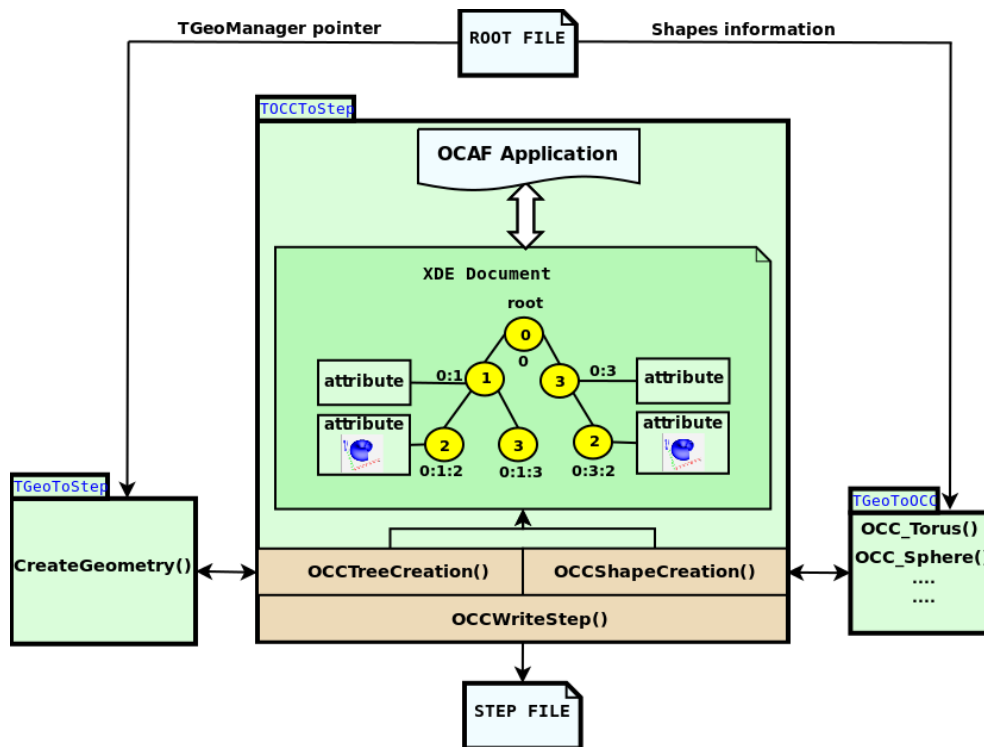


Figure 5.12: The TGeoCad architecture.

5.4 Results

The table 5.3 shows the number of volumes and nodes of the ROOT geometry, the size of the ROOT files and of the related STEP files, as well as the execution time of each conversion.

Nr. Volumes	Nr. Nodes	ROOT file	STEP File	Execution time
13	485	5.3 KB	143.1 KB	0.18 s
90	184	19.7 KB	1.4 MB	792.53 s
645	646	36 KB	10.3 MB	10.27 s
936	936	13.1 KB	15.6 MB	16.22 s
7143	29046966	328.2 KB	13.9 MB	6303.58 s

Table 5.3: Information related to the conversion of several geometries.

The execution time is not directly proportional to the number of volumes or to the size of the ROOT file. It can be asserted that, in the majority of the cases, the execution time depends on the number of boolean operations required to reproduce the geometry. These boolean operations are time consuming in Open CASCADE.

The dimension of the STEP file depends on the size of the ROOT file and on how many times a same node is repeated in the geometry. Each time a node is repeated in the ROOT geometry, the implementation of the shape does not take place but only the reference to the object is re-written in the file by modifying the location.

5. THE CAD INTERFACE FOR THE ROOT GEOMETRY MODELLER

Conclusion

This thesis has presented two contributions in the context of physics data analysis for HEP experiments.

A prototype to provide interactivity on the Grid has been presented in Chapter 4. It allows to merge the advantage of interactive execution provided by PROOF with the possibility of dynamically allocated resources on the Grid. Section 4.1 highlight the advantages of using PROOF on the Grid and Section 4.2 addresses the problems taken into account in the design phase of the prototype such as data locality, transparency, scalability. The *PoD on AliEn* architecture and behaviours are shown and the results related to performance tests are presented. The start-up latency of PoD Servers and PoD Workers has been calculated by executing 100 PoD Servers which in turn required 100 PoD Workers on the CERN site. The start-up time to get all the 100 workers is of several minutes, depending on the site occupancy rate at the moment of the submission, on the user priority in AliEn and on the number of jobs slot on the site. The start-up time can be improved using a dedicated queue for the PoD jobs or assigning high priority to these jobs. However, the user can start the analysis as soon as the first worker is available, the other workers will be connected to the server as soon as they are ready. The user can benefit of the interactive data analysis on Grid when the remote PROOF cluster is set up and ready to be used.

In Chapter 5 an extension of the ROOT framework, the *TGeoCad Interface*, is presented. It allows to translate ROOT geometries, used by physicists, into the STEP format, and thus into CAD systems, commonly used for detector design. After a briefly introduction of the ROOT geometry modeller, the TGeoCad Interface architecture and implementation has been shown.

The execution time of the conversion has been calculated showing that it is not directly correlated to the number of volumes contained in the geometry or to the ROOT

CONCLUSION

file size. Instead it mostly depends on the number of boolean operations required to reproduce the geometry which are time consuming operations in Open CASCADE. The size of the STEP file depends on the size of the ROOT file and on how many times the same node is repeated in the geometry. A node that appears several times in the geometry is converted only once and the reference to that shape is used to insert a new entry in the OCCT file reporting the shape reference itself and the new location of the shape within the geometry.

Bibliography

- [1] **The Large Hadron Collider.** <http://home.web.cern.ch/about/accelerators/large-hadron-collider>.
- [2] THE ALICE COLLABORATION ET AL. **The ALICE experiment at the CERN LHC.** *Journal of Instrumentation*, **3**, 2008. <http://aliceinfo.cern.ch/>.
- [3] **The ATLAS experiment.** <http://atlas.ch/>.
- [4] **The CMS experiment.** <http://cms.web.cern.ch/>.
- [5] **The LHCb experiment.** <http://lhcb.web.cern.ch/lhcb/>.
- [6] CERN COMMUNICATION GROUP. **LHC:the guide.** <http://cds.cern.ch/record/1165534/files/CERN-Brochure-2009-003-Eng.pdf>.
- [7] IAN FOSTER; CARL KESSELMAN. **The grid: blueprint for a New Computing.** ISBN:1-55860-475-8, 1998.
- [8] STEVEN TUECKE IAN FOSTER; CARL KESSELMAN. **The Anatomy of the Grid. Enabling scalable virtual organization.** *The Intl. Jnl. of High Performance Computing Applications*, pages 15(3):200-222, 2001.
- [9] IAN FOSTER; CARL KESSELMAN. **The Grid 2: Blueprint for a New Computing.** ISBN 1-55860-933-4, 2003.
- [10] **The Worldwide LHC Computing Grid.** <http://wlcg.web.cern.ch/>.
- [11] **WLCG Google Earth Dashboard.** <http://wlcg.web.cern.ch/wlcg-google-earth-dashboard>.
- [12] IOAN RAICU SHIYONG LU IAN FOSTER, YONG ZHAO. **Cloud Computing and Grid Computing 360-Degree Compared.** *Proceedings of the IEEE Grid Computing Environments Workshop*, pages 1-10, 2008.
- [13] **CERN Cloud Infrastructure User Guide.** <http://information-technology.web.cern.ch/book/cern-private-cloud-user-guide>.
- [14] **The CernVM project.** <http://cernvm.cern.ch/portal/>.
- [15] **BOINC.** <http://boinc.berkeley.edu/>.
- [16] **Amazon Elastic Compute Cloud.** <http://aws.amazon.com/ec2/>.
- [17] SEBASTIEN GOASGUEN ET AL. **Lxcloud: a prototype for an internal cloud in HEP. Experiences and lessons learned.** *J. Phys.: Conf. Ser.*, **396 Part 3**, 2012.
- [18] **The ROOT analysis framework.** <http://root.cern.ch/>.
- [19] **Condor Classified Advertisements.** <http://research.cs.wisc.edu/htcondor/classad/>.
- [20] **Physics Analysis Workstation.** <http://wwwasd.web.cern.ch/wwwasd/paw/>.
- [21] **Java Analysis Studio.** <http://jas.freehep.org/jas3/>.
- [22] **PROOF.** <http://root.cern.ch/drupal/content/proof>.
- [23] **Load Sharing Facility.** <http://www-03.ibm.com/systems/technicalcomputing/platformcomputing/products/lsf/>.
- [24] JENS VOLKERT HERBERT ROSMANITH, DIETER KRANZLMULLER. **An Interactive Job Manager for Globus.** *In Proceedings of Computer Aided Systems Theory, EUROCAST 2007*:431442, 2007.
- [25] E. DE LARA M. SATYANARAYANAN H. A. LAGAR-CAVILLA, N. TOLIA AND D. R. OHALLARON. **Interactive resource-intensive applications made easy.** *In Proceedings of the ACM/IFIP/USENIX 8th International Middleware Conference*, volume **4834** of *Lecture Notes in Computer Science*:pages 143163, 2007.
- [26] S. BASU V. TALWAR AND R. KUMAR. **Architecture and environment for enabling interactive grids.** *Journal of Grid Computing*, page 1(3):231251, 2003.
- [27] MIQUEL A. SENAR JOS SALT ELISA HEYMANN, ALVARO FERNANDEZ. **The EU-CrossGrid Approach for Grid Application Scheduling.** *In proceeding of: Grid Computing, First European Across Grids Conference, Santiago de Compostela, Spain.*, 2003.
- [28] **High-Performance Portable MPI.** <http://www.mpich.org/>.
- [29] N. MEYER B. PALAK M. PL CIENNIK M. KUPCZYK, R. LICHWALA AND P. WOLNIEWICZ. **Applications on demand as the exploitation of the Migrating Desktop.** *Future Generation Computer Systems.*, **21(1)**:3744, 2005.
- [30] ELISA HEYMANN MIQUEL A. SENAR ENOL FERNANDEZ, ANDRS CENCERRADO. **CROSSBROKER: A GRID METASCHEDULER FOR INTERACTIVE AND PARALLEL JOBS.** *Computing and Informatics*, **27**:187197, 2008.
- [31] I. CAMPOSA ET AL J. MARCOA. **THE INTERACTIVE EUROPEAN GRID: PROJECT OBJECTIVES AND ACHIEVEMENTS.** *Computing and Informatics*, Vol. **27**:161171, 2008.
- [32] **gLite.** <http://glite.web.cern.ch/glite/>.
- [33] R. KELLER K. DICHEV, S. STORK AND E. FERNANDEZ. **Mpi support on the grid.** *Computing and Informatics*, **27(3)**:213222, 2008.

BIBLIOGRAPHY

- [34] H. ROSMANITH AND J. VOLKERT. **Interactive techniques in grid computing: A survey.** *Computing and Informatics*, **27**::199211, 2008.
- [35] ANA YAIZA RODRIGUEZ MARRERO ET AL. **Interactive Analysis using PROOF in a GRID Infrastructure.** *J. Phys.: Conf. Ser.*, **331**, 2011.
- [36] J SAMSON W BEHRENOFF, W EHRENFELD AND H STADIE. **PROOF on a Batch System.** *J. Phys.: Conf. Ser.*, **331**, 2011.
- [37] **Oracle Grid Engine.** <http://www.univa.com/oracle>.
- [38] ANA Y RODRIGUEZ-MARRERO ET AL. **Integrating PROOF Analysis in Cloud and Batch Clusters.** *J. Phys.: Conf. Ser.*, **396**, 2012.
- [39] D BERZANO ET AL. **PROOF on the Cloud for ALICE using PoD and OpenNebula.** *J. Phys.: Conf. Ser.*, **368**, 2012.
- [40] **OpenNebula.** <http://opennebula.org/>.
- [41] D BERZANO ET AL. **PROOF as a Service on the Cloud: a Virtual Analysis Facility based on the CernVM ecosystem.** *J. Phys.: Conf. Ser.*, **In publication**, 2013.
- [42] MAARTEN VAN STEEN ANDREW S. TANENBAUM. **Distributed systems: principles and paradigms.** *Prentice Hall*, ISBN / ASIN: 0130888931:803, 2002.
- [43] **The Globus toolkit.** <http://www.globus.org/toolkit/>.
- [44] ET AL. ADERHOLZ M. **Models of networked analysis at regional centres. MONARC phase 2 rep.** *CERN-LCB-2000-001*, 2000.
- [45] **European Middleware Initiative.** <http://www.eu-emi.eu>.
- [46] **OMII.** <http://www.omii.ac.uk/>.
- [47] **VDT.** <http://vdt.cs.wisc.edu/>.
- [48] **The AliEn.** <http://alien2.cern.ch/>.
- [49] **CREAM.** http://www.eu-emi.eu/products/-/asset_publisher/1gkD/content/cream-3.
- [50] E LANCIOTTI P MNDEZ LORENZO N MAGINI V MICCIO R SANTINELLI A SCIAB S CAMPANA, A DI GIROLAMO. **Testing and integrating the WLCG/EGEE middleware in the LHC computing.** *Journal of Physics, Conference Series* **119**:9, 2008.
- [51] **Disk Pool Manager.** http://www.eu-emi.eu/products/-/asset_publisher/1gkD/content/dpm-2.
- [52] **CASTOR.** <http://castor.web.cern.ch/>.
- [53] **dCache.** <http://www.dcache.org/>.
- [54] ALICE COLLABORATION. **ALICE Computing Model. CERN-LHCC-2004-038/G-086**:26, 2005.
- [55] **The Simple Object Access Protocol.** <http://www.w3.org/TR/soap/>.
- [56] P. BUNCIC, A. J. PETERS, P. SAIZ, AND J.F. GROSSE-OTRINGHAUS. **The architecture of the AliEn system.** CHEP 2004 - Interlaken, Switzerland.
- [57] PETERS A J SAIZ P, BUNCIC P. **AliEn Resource Brokers.** *Conf. for Computing in High-Energy and Nuclear Physics*, 2003.
- [58] MARVIN SOLOMON. **The ClassAd Language Reference Manual Version 2.4.** 2004.
- [59] **XRootD.** <http://xrootd.slac.stanford.edu/>.
- [60] **VOMS.** http://www.globus.org/grid_software/security/voms.php.
- [61] **OpenSSL.** <http://www.openssl.org/>.
- [62] **The MonALISA.** <http://alimonitor.cern.ch/>.
- [63] **JAliEn.** <http://jalien.cern.ch/>.
- [64] **Helmholtz Centre for Heavy Ion Research.** <http://www.gsi.de/>.
- [65] **LLVM.** <http://www.llvm.org/>.
- [66] **Clang.** <http://clang.llvm.org/>.
- [67] **Geant4.** <http://geant4.cern.ch/>.
- [68] **Fluka.** <http://www.fluka.org/fluka.php>.
- [69] **CAF.** <http://aliweb.cern.ch/Offline/Activities/Analysis/CAF/index.html>.
- [70] **PoD.** <http://pod.gsi.de/>.
- [71] **PBS.** <http://www.pbsworks.com/>.
- [72] **ALICE Analysis Facilities.** <http://aaf.cern.ch/>.
- [73] **ApMon User Guide.** http://monalisa.cern.ch/monalisa_Documentation_ApMon_User_Guide.htm.
- [74] **The ROOT Geometry Package.** <ftp://root.cern.ch/root/doc/18Geometry.pdf>.
- [75] **ISO 10303: STandard for the Exchange of Product model data.** <http://www.iso.org/>.
- [76] W. H. FREEMAN AND CO. **Introduction to Solid Modeling. ISBN:0-88175-108-1**, 1988. <ftp://root.cern.ch/root/doc/18Geometry.pdf>.
- [77] **The OpenCASCADE Technology.** <http://www.opencascade.org/>.
- [78] **The EXPRESS data modelling language.** [http://en.wikipedia.org/wiki/EXPRESS_\(data_modeling_language\)](http://en.wikipedia.org/wiki/EXPRESS_(data_modeling_language)).