Universitat de València
Departament de Física Teòrica




# Distributed computing and farm management with application to the search for heavy gauge bosons using the ATLAS experiment at the LHC (CERN)

Tesis de Doctorado
Juan Antonio Lopez-Perez
2007

# CONTENTS

# LIST OF FIGURES

# Preface

The Standard Model of particle physics (SM), presented in the introduction of this work, describes the strong, weak, and electromagnetic forces between the fundamental particles of ordinary matter. The SM describes with high precision the interactions at energies reached by present accelerators. However, it presents several problems, also discussed in this work, and some questions remain unanswered so the SM cannot be considered a complete theory of fundamental interactions.

Since the completion of the SM, many extensions have been proposed in order to address these problems. Some important recent extensions are the Extra Dimensions (ED) theories. These theories unify gravity with the other fundamental forces and solve the hierarchy problem. An introduction on the motivation, history and main ideas of ED theories is given in this work.

In the context of some models with ED of size about 1 $TeV^{-1}$, in particular in the ADD model with only fermions confined to a D-brane, heavy Kaluza-Klein (KK) excitations are expected, with the same properties as SM gauge bosons but more massive. In this work, three hadronic decay modes of some of such massive gauge bosons, $Z^*$ and $W^*$, are investigated using the ATLAS experiment at the Large Hadron Collider (LHC), presently under construction at CERN.

The LHC will start to operate in 2007 and collect data in 2008. It will produce roughly 15 Petabytes (15 million Gigabytes, i.e. more than 20 millions of CDs) of data per year. Access to this experimental data has to be provided for some 5,000 scientists working in 500 research institutes and universities. In addition, all data need to be available over the estimated 15-year lifetime of the LHC. The analysis of the data, including comparison with theoretical

simulations, requires a huge amount of computing power.

The computing challenges that scientists have to face are: the huge amount of data and calculations to perform and the large number of collaborators. The Grid has been proposed as a solution for those challenges.

In particular, regarding the high CPU requirements required for LHC studies, a particular kind of Grid that is useful and has been considered in this work is BOINC. The feasibility of porting some simulation and reconstruction physics tools has been studied as well.

This work is presented in five different parts, including seven sections and four appendices. In short, there are two main topics. One refers to data analysis and another one to distributed computing. In addition, there is an introduction, conclusions and appendices.

In the first part of this work, an introduction to the physics and computing tools is presented.

In the first chapter of this part an introduction to CERN and to its main collider, the LHC, is given. The ATLAS detector and the main computing tools are also introduced.

In the second chapter, the Standard Model of particle physics is introduced and some extensions are proposed. In particular, an introduction to Extra Dimensions Models is given.

In the third chapter, a general introduction to Grid computing is presented. In particular, two important Grid projects, LCG and BOINC, are discussed.

The second part involves one single chapter that includes the analysis of $Z^*$ and $W^*$ decays using the ATLAS experiment. In particular, the search for the decay modes $Z^* \longrightarrow b\bar{b}$, $Z^* \longrightarrow t\bar{t}$ and $W^* \longrightarrow t\ b$ is presented. This search is performed using data simulated with the ATLAS Fast Simulation and reconstruction package, Atlfast.

In the third part of this work, details of the distributed computing environments LCG and BOINC are given in three different chapters. Their technology and infrastructure mechanisms are studied in depth. In addition, the main activity areas concerning ATLAS computing are discussed. The work performed on the porting of some physics applications, like Atlfast, to the BOINC Grid environment is presented. The data used for our analysis is also generated and reconstructed again using this environment and is compared with the one obtained without the use of the Grid. Finally, a comparison between LCG and BOINC Grid environments is given.

Finally, in the fourth part, the most important conclusions of this work are summarised.

A final part includes some appendices with details about the porting of LCG software to non-linux platforms and of Atlfast and Geant4 to the BOINC Grid environment. A glossary including the acronyms and abbreviations used in this work is also given.

This work has been possible thanks to a close collaboration between the IFIC research institute in Valencia and CERN and it was supported by both IFIC and CERN.

# Prefacio

En el contexto de la física de partículas, el Modelo Estándard (Standard Model, SM, en inglés) presentado en la introducción, describe tres de las fuerzas existentes entre las particulas fundamentales de la materia ordinaria: la fuerza electromagnética, la fuerte y la débil. El SM ha demostrado ser capaz de describir con alta precisión dichas interacciones hasta los niveles de energías capaces de ser alcanzadas por los aceleradores de particulas actuales. Sin embargo, presenta varios problemas, también descritos en esta tesis, y varias cuestiones permanecen sin respuesta todavía, de modo que el SM no puede ser considerado una teoria completa sobre las interacciones fundamentales.

Desde que se completó la descripción del SM, varias extensiones han sido propuestas con el objetivo de tratar los problemas comentados. Una reciente e importante extension son las teorias de Dimensiones Extra (Extra Dimensions, ED, en inglés). Dichas teorias unifican la gravedad con las otras fuerzas fundamentales y resuelven también el problema de la jerarquia. En este trabajo se incluye una introducción a la motivación, la historia y las ideas principales en las que se basan las teorias de Dimensiones Extra.

En el contexto de ciertos modelos con Dimensiones Extra de tamaños de alrededor de 1 $TeV^{-1}$, en concreto en el modelo llamado ADD con solo fermiones confinados en la D-brana, se esperan encontrar excitaciones pesadas de kaluza-Klein (KK) con las mismas propiedades que los bosones gauge del SM pero con mayor masa. En este trabajo se investigan tres modos de desintegración de algunos de dichos bosones gauge masivos, $Z^*$ y $W^*$, usando el experimento ATLAS en el Gran Colisionador de Hadrones (Large Hadron Collider, LHC, en inglés) que se encuentra actualmente en construcción en el CERN.

El LHC comenzará a funcionar en el 2007 y a recoger datos en el 2008. Producirá alrededor de 15 Petabytes (15 millones de Gigabytes, más de 20 millones de CDs) de datos por año. Unos 5000 cientificos trabajando en más de 500 institutos de investigación y universidades deben de tener acceso a dichos datos. Además, toda esa información debe de estar disponible durante el tiempo de vida estimado del LHC, alrededor de 15 años. El analisis de dichos datos, incluyendo la comparación con las simulaciones teóricas, requiere una cantidad inmensa de capacidad de computación.

Los cientificos deben enfrentarse a los siguientes retos tecnológicos: la inmensa cantidad de datos y cálculos necesarios y el gran número de colaboradores. El Grid ha sido propuesto como una solución para dichos retos.

En concreto, para solucionar el problema de las inmensas necesidades de CPU que necesitarán los experimentos en el LHC, se ha estudiado en este trabajo un tipo particular de Grid llamado BOINC. Se ha estudiado también en detalle la viabilidad de la adaptación a este entorno de ciertas herramientas de simulación y reconstrucción en física usadas por ATLAS actualmente.

Este trabajo se presenta en cinco partes diferentes, incluyendo siete secciones y cuatro apéndices. En resumen, se tratan fundamentalmente dos temas. Uno de ellos sobre análisis de datos y otro sobre computación distribuida. Además, hay una introducción, conclusiones y apéndices.

En la primera parte de este trabajo, se presenta una introducción a las herramientas de física y computacion en física usadas por ATLAS.

En el primer capítulo de dicha parte, se ofrece una introducción al CERN y a su principal colisionador, el LHC. También se introduce el detector ATLAS y sus principales herramientas de computación.

En el segundo capítulo, se introduce el Modelo Estándar de la física de partículas y ciertas extensiones propuestas. En concreto, se da una introducción a los modelos de Dimensiones Extras.

En el tercer capítulo, se presenta una introducción a computación Grid y se tratan dos proyectos Grid importantes: LCG y BOINC.

La segunda parte incluye un capítulo único en el que se expone el analisis de desintegraciones de bosones $Z^*$ y $W^*$ en el experimento ATLAS. En concreto, se trata la búsqueda de los modos de desintegración $Z^* \longrightarrow b\bar{b}$, $Z^* \longrightarrow t\bar{t}$ y $W^* \longrightarrow t\ b$ . Este análisis se realiza usando datos simulados con el paquete de simulación y reconstrucción rápida de sucesos de ATLAS (en inglés, ATLAS Fast Simulation and reconstruction, Atlfast).

En la tercera parte de este trabajo, se dan detalles sobre los entornos de computación distribuída LCG y BOINC y se estudia en detalle la tecnología e infrastructura usadas por ambos. Se presenta también el trabajo

realizado en la adaptación al entorno Grid BOINC de ciertas aplicaciones de física usadas por la colaboración ATLAS, como Atlfast, y se generan y reconstruyen de nuevo en dicho entorno los datos simulatos utilizados en nuestro análisis, comparándolos con los obtenidos sin utilizar entorno Grid. Se realiza, además, una comparación entre los entornos Grid de LCG y BOINC.

Finalmente, en la cuarta parte, se resumen las conclusiones más importantes de este trabajo.

Al final de la tesis se incluye una parte adicional con varios apéndices dando detalles sobre la adaptación de software de LCG a plataformas no-Linux y la adaptación al entorno Grid BOINC de dos programas de generación y reconstrucción de sucesos usados en ATLAS: Atlfast y Geant4. También se incluye un glosario incluyendo los acrónimos y abreviaturas usadas.

Este trabajo ha sido posible gracias a una colaboración estrecha entre el centro de investiación IFIC en Valencia y el CERN y ha sido apoyado por ambos.

# Part I

# Introduction

CHAPTER

# 1

# CERN, LHC and ATLAS

In this first chapter we give an introduction to CERN, its main collider, the LHC and the ATLAS experiment.

## 1.1 CERN

After the Second World War, the European physics research was in a very poor state. Louis de Broglie (Nobel prize-winner in 1929) proposed the creation of an European Science Laboratory in 1949, in the "Conférence Européenne de la Culture" at Lausanne.

Currently, the European Organisation for Nuclear Research, known as CERN [1], is the world's largest particle physics laboratory. It is situated in Geneva on the border between France and Switzerland. It is also known for being the birthplace of the World Wide Web (WWW).

CERN has 20 European Member States, but many non-European countries are also involved in different ways. It employs 3000 people and about 6500 visiting scientist coming from over 500 universities and research institutes from more than 80 countries. Apart from physicists, CERN's staff also includes highly specialised engineers and technicians.

The aim of CERN is to provide the particle accelerators and other infrastructure needed for high energy physics research. Numerous experiments have been performed at CERN by international collaborations. CERN

also owns a large computer centre with powerful data processing facilities, primarily devoted to experimental data analysis.

The acronym CERN [2] stands for Conseil Européen pour la Recherche Nucléaire (European Council for Nuclear Research). This provisional council was approved by 11 European governments in 1952 in order to set up the laboratory[2] [3].

In 1954, the provisional council was dissolved and the organisation was given the name European Organisation for Nuclear Research, although the acronym CERN was retained. More information about the history of CERN can be found in [3].

At the beginning, research concentrated in nuclear studies, hence the word "nuclear". Very soon, the work of the laboratory sifted to higher energies. Therefore, CERN is today a High Energy Physics (HEP) institute devoted to the study of fundamental particles and interactions. For this reason, CERN is also commonly known as the "European Laboratory for Particle Physics".

### 1.1.1 Discoveries and research fields

Several important discoveries in particle physics have been made at CERN. They include:

- The discovery of neutral currents in 1973 by the Gargamelle bubble chamber experiment.
- The discovery of W and Z bosons in 1983 by the UA1 and UA2 experiments.

Some important technical achievements made at CERN were the construction of the Intersecting Storage Rings (ISR) commissioned in 1971, and the Super Proton Synchrotron (SPS), which came into operation in 1981 and produced the massive W and Z particles, confirming the unified theory of electromagnetic and weak forces. Another revolutionary technological development is the invention of the multi-wire proportional chamber in the 60s. In the 90s very precise measurements were made using the Large Electron-Positron Collider (LEP), in particular the measurement of the number of neutrino families. The results obtained at LEP confirmed with high precision the Standard Model.

The research program at CERN, apart from Particle Physics in large colliders, includes projects in Nuclear Physics like the Isotope Separation OnLine DEvice (ISOLDE) project, or in Neutrino Physics, like the CERN Neutrinos to Gran Sasso (CNGS) project. Other active research fields are technological developments in accelerators, detectors and computer science.

Concerning computing, a revolutionary development at CERN was the development of the World Wide Web ("WWW" or simply the "Web") in the 80s by Tim Berners-Lee and others. On April 30, 1993, CERN announced that the World Wide Web would be a free tool, available to everybody. A recent study found that there are over 11.5 billion indexed web pages in the Web as of January 2005.

An important active research field is currently the Grid project, with the goal of handling the huge amount of data expected at the LHC.

Grid computing is covered in depth in section 3.

## 1.2 The LHC

The accelerator complex of CERN (shown schematically in figure 1.1) consists of several machines where particles are accelerated successively to higher energies. The last stage of the complex is the Large Hadron Collider (LHC) [4] [5], presently under construction. In addition, the LHC injectors are used for experiments at lower energies.

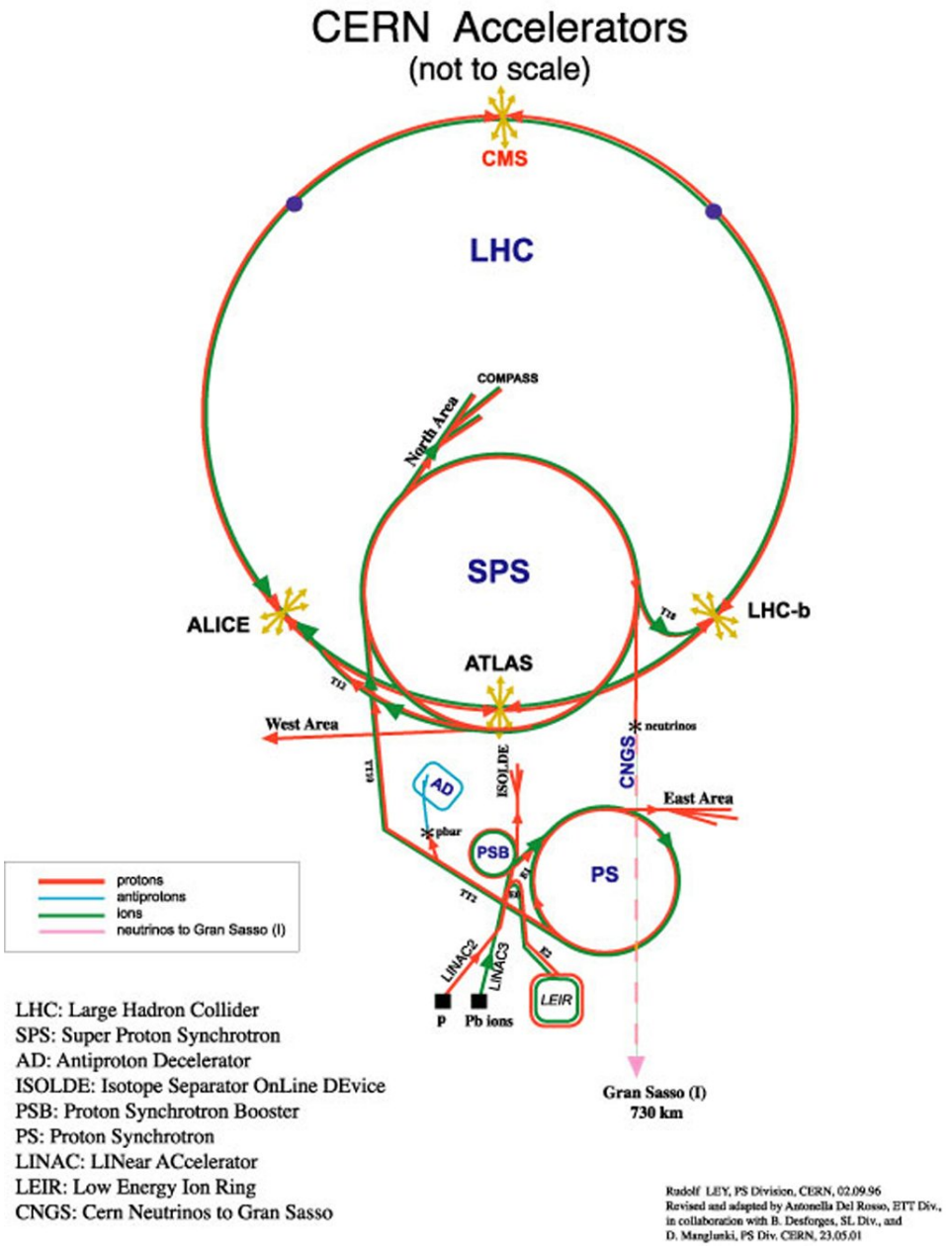


Figure 1.1: Scheme of the CERN accelerator complex.

The LHC accelerates protons and lead ions at high energies never achieved before. Two proton beams collide with a center-of-mass energy of 14 TeV and heavy ions with a center-of-mass energy of 1250 TeV. The accelerator is placed in the tunnel of 27 km of circumference used by LEP, 100 meters underground. The existing accelerator facilities at CERN are used as pre-accelerators (see figure 1.1).

Each LHC beam contains 2835 bunches, each of them with $10^{11}$ particles. Since the beam crossing frequency is 25 ns, a luminosity of $10^{34}$ cm $^{-2}$s$^{-1}$ can be achieved in the so-called 'high luminosity' phase. To achieve such a challenging performance, LHC uses the most advanced super-conducting magnet and accelerator technologies.

Figure 1.2 shows the four experiments planned for the LHC along the accelerator ring. These experiments are ALICE (shown in figure 1.3) dedicated to the study of heavy-ion physics and quark-gluon plasma, LHCb (also shown in figure 1.3) for CP violation studies in B meson decays and finally ATLAS and CMS (shown in figures 1.4 and 1.5), which are general purpose experiments. ATLAS is described in detail in the next section.

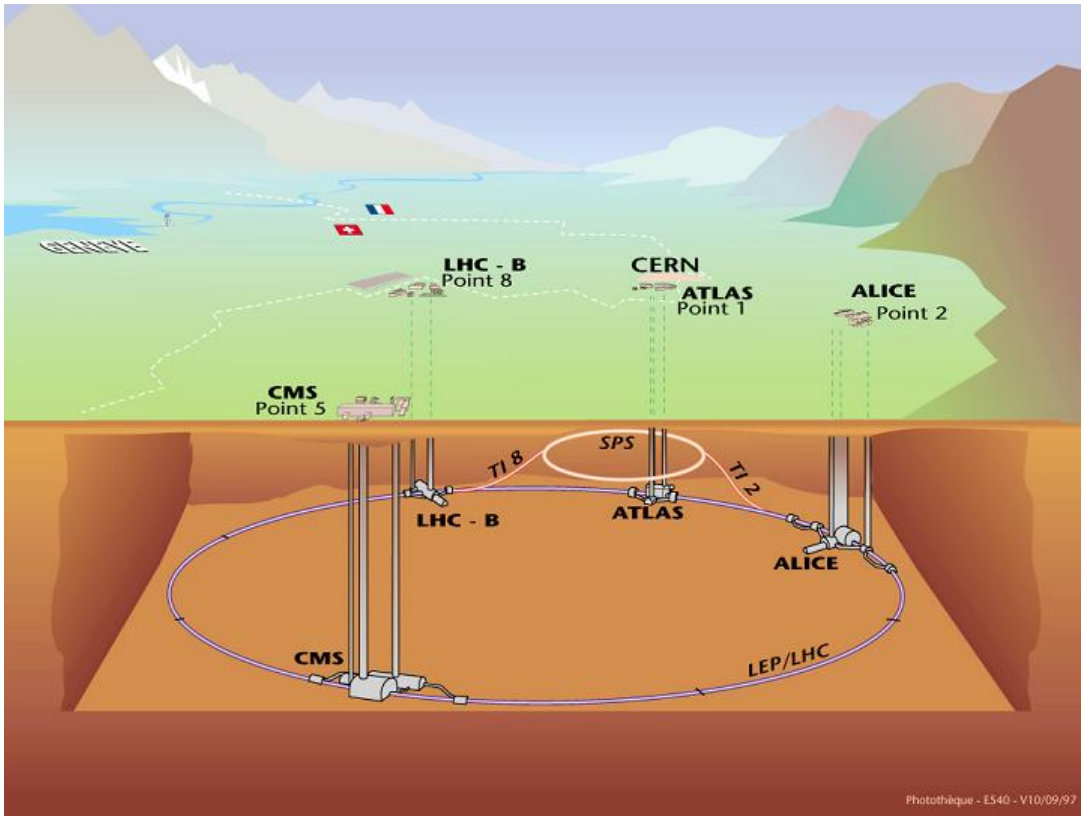


Figure 1.2: The 4 LHC experiments.

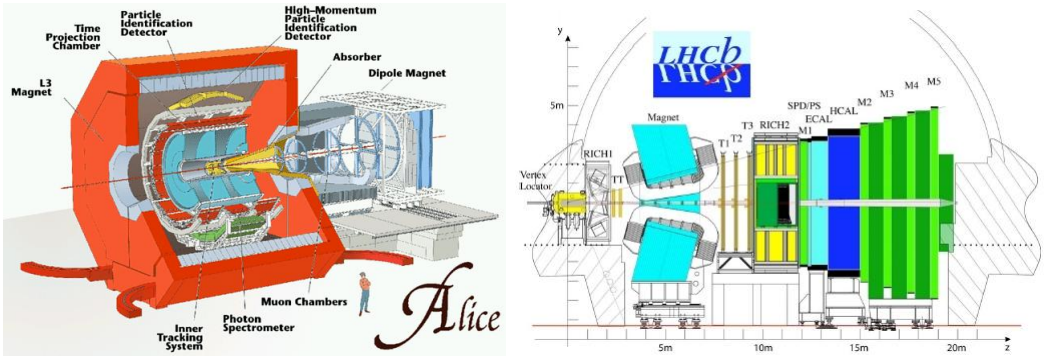


Figure 1.3: The Alice and LHCb detectors.

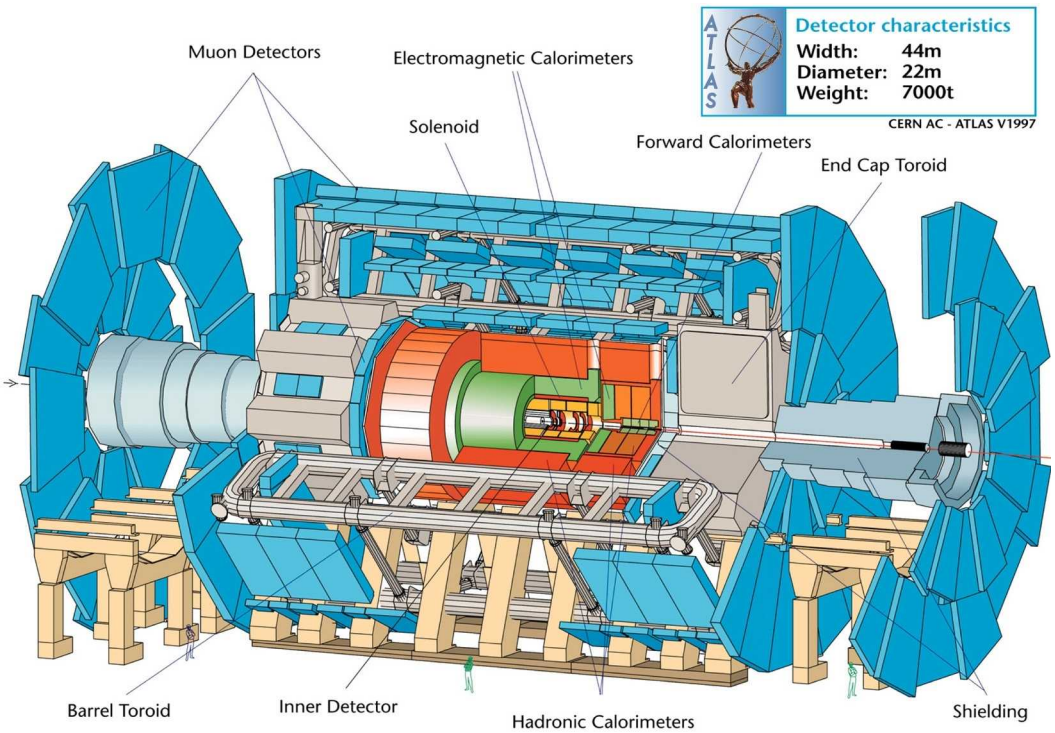


Figure 1.4: Sketch of the ATLAS detector.

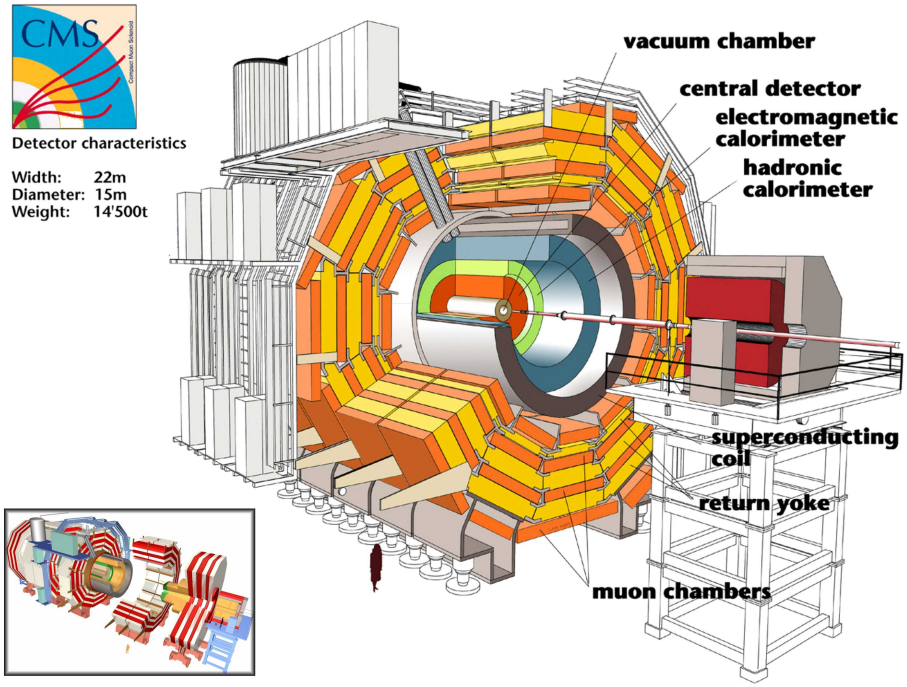


Figure 1.5: The CMS detector.

## 1.3 The ATLAS experiment

ATLAS (A Toroidal LHC ApparatuS) [6] is a general-purpose detector designed to exploit the full discovery potential of the LHC. The detector design is optimised to analyse expected processes in the SM and new physics as well. ATLAS includes a very good electromagnetic calorimeter (for electron and photon identification and energy measurements), complemented by full-coverage hadronic calorimetry (for accurate jet and missing transverse energy measurements) and finally a high-precision muon tracking system. The technical details are described in the ATLAS Technical Proposal [7] and in the ATLAS Technical Design Report [8].

Figure 1.6 shows a photograph of the ATLAS detector under construction.

ATLAS has a structure with several layers, called sub-detectors, in order to measure particle properties. The main goals of the sub-detectors are summarised below.

- Observation of charged particles tracks. This implies measuring the charge, trajectory and momentum of the particles. It also implies the detection of secondary vertices from short-lived decaying particles.
- Measurement of the energy carried by electrons, photons and hadrons produced in each collision.

Figure 1.6: Photograph of the ATLAS detector under construction.

- Indirect detection using transverse momentum conservation of weak-interacting neutral particles, such as neutrinos.
- Identification of the detected particles.
- Capability for maintaining a long and reliable operation in a very hostile radiation environment.

The different parts of the ATLAS detector are listed below

- **Tracking Detector:** The inner region of the detector is filled with highly segmented position sensitive devices in order to measure charged particle tracks with high accuracy.
- **Calorimetry System:** The calorimeters measure electromagnetic and hadronic energy. They have enough thickness to fully absorb the electromagnetic or hadronic showers produced by the particles. Electromagnetic calorimeters measure the energy of electrons, positrons and photons interacting with charged particles inside matter. Hadronic calorimeters measure the energy of hadrons interacting with atomic nuclei.
- **Muon Chambers:** The outer layers of the detector are able to detect charged particles. They consist of gas-filled chambers. As only muons and neutrinos can reach these outer layers, they are called muon chambers. The presence of neutrinos can only be inferred from transverse missing energy.

- **Magnet Systems:** They are intended to bend the trajectories of charged particles and allow the measurement of their momenta.

In fixed target experiments, the particles are produced mostly in the forward direction and the detectors are placed in the beam downstream direction. In colliding beam experiments like ATLAS, particles are produced in all directions so a $4\pi$ stereo radian detector geometry coverage is needed. As a result, ATLAS has a cylindrical structure.
The main physics goal of ATLAS (and also CMS, the other general purpose experiment) is to explore SM and beyond the SM processes. In particular:

- To discover or exclude the Standard Model Higgs or the various Higgs bosons of super-symmetric theories.
- To discover or exclude super-symmetry.
- To discover or exclude any new electroweak gauge bosons with masses of the order of 1 TeV.
- To discover or exclude any new quarks or leptons with masses of the order of 1 TeV.
- To study the production and decays of the top quark.
- To study B-physics, the decay of B-baryons and mesons.

### 1.3.1 ATLAS sub-detectors

In the following, the different ATLAS sub-detectors are presented. A schematic view of the ATLAS detector is presented in figure 1.4 showing the various sub-detectors.

Some of the ATLAS subsystems have been partly constructed in Spain. In particular, the Instituto de Física Corpuscular (IFIC) [10] in Valencia hosts three working groups involved in the ATLAS experiment:

- The **SCT group** is involved in the Atlas Forward Silicon Tracker construction. In total, 220 Semi-Conductor Tracker (SCT) modules and the corresponding read out electronics have been assembled and tested. This work has been done in collaboration with the Centro Nacional de Microelectrónica (CNM) in Barcelona.
- The **TileCal group** is responsible for the design, assembly, test and commissioning of the Read Out Drivers electronic boards (RODs) for the Hadronic Tile Calorimeter (TileCal) of the ATLAS detector.

  The group has also constructed 50% of the modules for one of the extended barrel parts of the hadronic calorimeter.

A photomultiplier test setup was installed at IFIC where more than 1750 photomultipliers (PMs) were tested. This represents 17.5% of the total number of PMs.

- The **ATLAS Grid Computing group** is in charge of a Tier-2 Grid centre and is involving in research and development of ATLAS Grid tools and software.

### Inner detector

The goal of the Inner Detector (ID) is to reconstruct tracks and vertices with high efficiency, contributing with the calorimeter and muon systems to electron, photon, muon and jet measurements. The ATLAS ID covers the pseudo-rapidity range $|\eta| < 2.5$. A view of the Inner detector is shown in figure 1.7 .

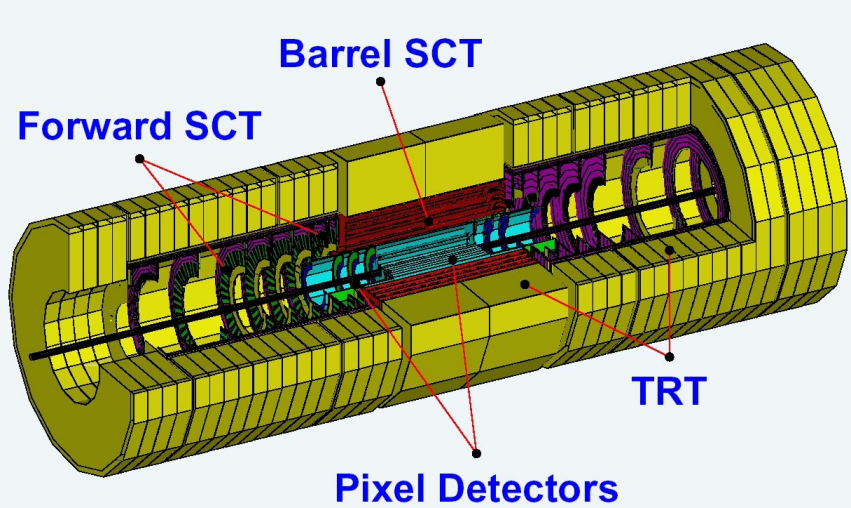


Figure 1.7: A view of the ATLAS Inner Detector.

Silicon micro-strip and pixel detectors are used to achieve high-precision measurement close to the interaction point. Around the vertex region, Pixel Detectors are used, providing a very high granularity. In the second layer, the Semi-Conductor Tracker (SCT) uses silicon micro-strip detectors. To increase the number of tracking points, a third layer is used: the Transition Radiation Tracker (TRT), with straw detectors providing the possibility of continuous tracking and electron identification. The combination of the two techniques (silicon detectors and straw detectors) gives very robust pattern recognition and high precision in both $\phi$ and z coordinates, with an average of 11 precision space-points measurements and 36 hits in the straws per track.

The Pixel Detector is designed to provide high precision measurements as close as possible to the interaction point. The detector consists of three layers

at average radii of 5 cm, 10 cm, and 12 cm, and four disks on each side, between radii of 11 and 20 cm, in order to complete the angular coverage. It contains approximately 1500 identical barrel modules and 1000 identical disk modules with a total amount of 140 million channels.

The SCT system is designed to provide eight precision measurements per track, contributing to the measurement of momentum, impact parameter and vertex position. The barrel SCT consists of four double-sided layers of silicon micro-strip detectors and provides precision points in the R$\phi$ and z coordinates. This sub-detector contains 61 $m^2$ of silicon detectors, with 6.2 millions channels.

The TRT consists of straw detectors. Each straw is a small cylindrical proportional chamber, with an anode wire in the center and the straw wall acting as a cathode. Electron identification capability is ensured by employing xenon gas to detect transition-radiation photons created in a radiator located between the straws. The TRT barrel contains about 50000 straws and the end caps contain 320000 radial straws, with 420000 electronic channels. The TRT detector is also shown in figure 1.7.

### Calorimeters

At the LHC, about twenty soft collisions per bunch crossing are produced. As a consequence, fast detector response and fine granularity are required to minimise the impact of pile-up on the physics performance.

ATLAS has an electromagnetic calorimeter covering the rapidity region $|\eta| < 3.2$, a barrel hadronic calorimeter covering $|\eta| < 1.7$, hadronic end cap calorimeters covering $1.4 < |\eta| < 3.2$, and forward calorimeters covering $3.2 < |\eta| < 4.8$.

The electromagnetic calorimeter is a lead Liquid-Argon (LAr) detector with accordion geometry. The hadronic barrel calorimeter (Hadron Tile Calorimeter) is based on a sampling technique with plastic scintillator plates (tiles) embedded in an iron absorber. At larger rapidities, where high radiation resistance is required, the radiation-hard LAr technology is used for all calorimeters: the hadronic end cap calorimeter and the forward calorimeter. A scheme with all the calorimeters is shown in figure 1.8 .

The **Liquid Argon** sampling calorimeter technique with "accordion-shaped" electrodes is used for electromagnetic calorimetry covering the pseudo-rapidity interval $|\eta| < 3.2$. This technique is also used for hadronic calorimetry in the range $1.4 < |\eta| < 4.8$.

In the barrel, the electromagnetic calorimeter consists of two identical half-barrels covering the rapidity range $|\eta| < 1.4$. For each half-barrel (divided into 16 modules) the calorimeter is made of 1024 accordion-shaped absorbers alternating with 1024 read-out electrodes. Between each pair of absorbers, there are two liquid argon gaps, separated by a read-out electrode.

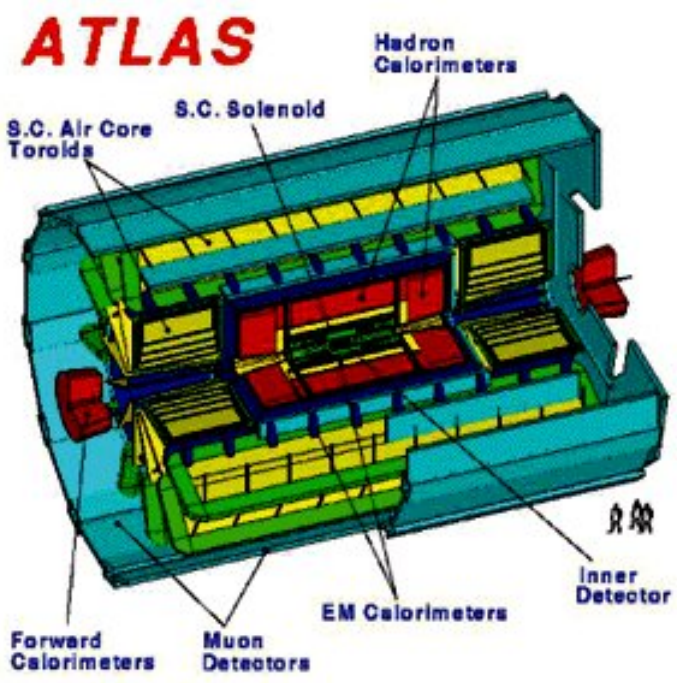


Figure 1.8: Scheme including the calorimeters of the ATLAS detector.

The electromagnetic EndCap calorimeter (EMEC), the Hadronic EndCap calorimeter (HEC) and the Forward Calorimeter (FCAL) are placed inside a common EndCap cryostat. The EMEC, covering the range $1.375 < |\eta| < 3.2$, uses the same technique as the barrel part.

The HEC covers the range $1.5 < |\eta| < 3.2$ and uses copper-plates as absorber, with parallel geometry in this case. The FCAL covers the range $3.2 < |\eta| < 4.9$ providing coverage for electromagnetic and hadronic showers by using copper and tungsten as absorbers, respectively.

The electromagnetic calorimeter is segmented in three longitudinal samplings in the $|\eta| < 2.5$ region and in two samples in the $|\eta| > 2.5$ region. Its total thickness is above 24 radiation lengths for the barrel and above 26 for the end caps.

The **Tile Calorimeter** is a sampling device made out of steel and scintillating tiles as absorber and active material, respectively. It is divided in three sections: the Central Barrel (CB) and two Extended Barrels (EBs). The barrel covers the region $|\eta| < 1.0$, and the extended barrels cover the region $0.8 < |\eta| < 1.7$. Azimuthally, the Barrel and Extended Barrels are divided into 64 modules. The full depth of the TileCal is above 7 absorption lengths in the CB and about 10 in the EB.

The tiles are placed perpendicular to the colliding beams. This is an inno-

vation of the ATLAS TileCal, since in most hadronic calorimeters the active elements are placed parallel to the beams. This design ensures homogeneity in the signal sampling. The structure is periodic along z. The tiles are 3 mm thick and the total thickness of the iron plates is 14 mm.

The light produced in the scintillating tiles has a wavelength in the ultraviolet region and intensity proportional to the energy deposited by the particles. Both sides of the scintillating tiles are read-out via WaveLength Shifting (WLS) fibres by two PhotoMultiplier Tubes (PMTs) to achieve redundant read-out. The WLS fibres shift the light to longer wavelengths, in order to match the sensitivity of the PMTs.

The use of fibres allows to define a three-dimensional cell read-out, with projective geometry to the interaction region for triggering and energy reconstruction. The TileCal has a three samplings longitudinal segmentation, with a $\Delta\eta \times \Delta\phi$ granularity equal to $0.1\times0.1$ in the first two samplings and $0.1\times0.2$ in the last sampling. A compact electronic read-out is housed in the girder of each module.

### Muon system

The ATLAS Collaboration has designed a high-resolution muon spectrometer with stand-alone triggering and momentum measurement capability over a wide range of transverse momentum, pseudo-rapidity and azimuthal angle. Four chamber technologies are employed in the detector. The positions of the chambers are optimised to achieve good hermeticity and optimum momentum resolution.

For precision measurement of muon tracks in the principal bending direction of the magnetic fields, Monitored Drift Tube (MDT) chambers are used ,except in the innermost ring of the inner station of the end caps where particle fluxes are highest. In this region, covering the pseudo-rapidity range $2< |\eta| <2.7$, Cathode Strip Chambers (CSCs) are employed.

The trigger function in the barrel is provided by three stations of Resistive Plate Chambers (RPCs). They are located on both sides of the middle MDT station, and either directly above or directly below (depending on $\phi$) the outer MDT station. In the end caps, the trigger is provided by three stations of Thin Gap Chambers (TGCs) located near the middle MDT station.

### Magnet system

An optimised magnetic field configuration for particle bending around the various detectors, within a structure which minimises scattering effects, has been chosen by ATLAS. The final arrangement consists of a central solenoid servicing the inner detector trackers with an axial magnetic field, surrounded by a system of three large air-core toroids generating a tangential magnetic field for the muon spectrometer. A niobium-titanium superconductor in a

copper matrix technology is used. The magnet system weighs 1300 tons and is cooled by liquid helium down to 4.5 K (needing 40 days to reach this temperature).

The subsystems are the Central Solenoid (CS), Barrel Toroids (BTs) and two EndCap Toroids (ECTs). The CS is a superconducting solenoid made as a single layer coil. It is magnetically decoupled from the toroid magnets and is mounted in the same cryostat as the LAr Calorimeter. This solenoid provides a 2 Tesla magnetic field.

The BTs cover the central region and provide a 2-6 $T \cdot m$ magnetic field integral. It is build up by eight flat racetrack magnets each of them consisting of two double pancake windings housed in a common aluminum case. The two ECTs are positioned inside the Barrel Toroid at the end of the Central Solenoid, providing a 4-8 $T \cdot m$ magnetic field integral. In contrast to the Barrel Toroid, the eight coils of each End Cap are assembled inside a single cryostat.

### 1.3.2 ATLAS performance

#### Inner detector

The track momentum resolution per isolated muons can be described approximately by

$$\sigma\left(\frac{1}{p_T}\right) = 3.6 \times 10^{-4} \oplus \frac{1.3 \times 10^{-2}}{p_T\sqrt{sin\theta}} \; [GeV^{-1}] \tag{1.1}$$

where $p_T$ is the transverse momentum (in GeV) and $\theta$ the polar angle. The first term is the intrinsic detector resolution and the second term is due to multiple scattering. This result implies that the momentum resolution is dominated by multiple scattering for momenta below 36 GeV.

The track impact parameter resolution for muons is:

$$\sigma(d_0) = 11 \oplus \frac{73}{p_T\sqrt{sin\theta}} \; [\mu m] \tag{1.2}$$

where $d_0$ is the impact parameter and $p_T$ is in GeV as before. The first term is the intrinsic detector resolution and the second term is due to multiple scattering. This result implies that the impact parameter resolution is dominated by multiple scattering for momenta below 6.6 GeV.

Similar results are obtained for isolated pions. Worse results are expected for electrons, since electron tracks suffer from bremsstrahlung. At high momenta, muons can be reconstructed with 99% efficiency. Pions and electrons can be found with 95% efficiency. For $p_T = 1\ GeV$, the efficiency drops to 97%, 84% and 76% for $\mu$, $\pi$ and $e$, respectively. These efficiencies are fairly robust to the effects of detector inefficiencies and noise, as well as pile-up at high luminosity.

The TRT allows a $\pi/e$ separation. For 90 % electron efficiency, the pion rejection is about 40.

Algorithms have been developed to allow the reconstruction of primary vertices in the context of high luminosity. For $H \rightarrow \gamma\gamma$ events (a difficult case since there are no changed tracks associated to the Higgs boson), an efficiency of 70% can be achieved. In $B^0 \rightarrow J/\psi K_s$ decays, the $K_s$ vertex can be reconstructed with 40% efficiency.

Of special interest is the capability of the inner detector to detect $b$-jets, i.e. jets originating from the decay of $b$-hadrons. Typical values are efficiencies of 60%, 10% and 1% for $b$-jets $c$-jets and other jets, respectively, using impact parameter based algorithms. This performance applies to jets in the $p_T$ range of 50 to 250 GeV.

Jets originating from $\tau$-decays can also be identified assuming $\tau$ hadronic decays. If the $\tau$-hadronic products are hard enough (typically $p_T > 10\ GeV$), the $\tau$ labeling efficiency is of the order of 90%. The alignment and calibration of the ID has also been considered in detail. An uncertainty in momentum scale of 0.02% seems challenging but possible to achieve.

**Electromagnetic calorimeter**

The electromagnetic calorimeter offers for electrons and photons a good energy resolution, excellent uniformity and angular resolution, and powerful particle identification capability. The main performance can be summarised as follows. The energy resolution for isolated electrons and photons can be described approximately by:

$$\frac{\sigma(E)}{E} = \frac{10\%}{\sqrt{E}} \oplus 0.7\% \tag{1.3}$$

where E is the particle energy in GeV. The first term is the sampling term and the second term depends on the calorimeter uniformity and calibration errors. The calibration of the calorimeter, leading to a constant term of 0.7%, can be achieved using $Z \rightarrow e^+e^-$ events.

The use of the fine longitudinal and lateral segmentation of the calorimeter allows several precision measurements of the shower position and angle to be performed. For example, the primary vertex in $H \rightarrow \gamma\gamma$ events can be measured with an accuracy of 1.3 cm. An average rejection factor of 3 against $\pi^0$ should be obtained for a photon efficiency of 90%.

The combination of the ID and the electromagnetic calorimeter provides the potential to identify and measure the energy and measurement of electrons and photons in the presence of background. Electrons and photons are however significantly affected by the material in front of the calorimeter. Nevertheless, the effects of bremsstrahlung and conversions can be partially compensated by the use of dedicated algorithms. Using $E/p$ from $W \rightarrow e\nu$

events, it should be possible to calibrate the calorimeter to 0.1 % in cells of size $\triangle\eta \triangle \phi = 0.2 \times 0.4$.

Low energy electrons coming from $b \rightarrow e$ or $J/\psi \rightarrow e^+e^-$ can also be identified. Electrons with $E_T > 20\ GeV$ can be identified with 70% efficiency and jet rejection rates of $10^5$. Photons with $E_T > 20\ GeV$ can be identified with 80% efficiency and jet rejections of $10^3$.

The invariant mass resolution for a light Higgs boson with mass of 100 GeV decaying into photon pairs is 1.1 GeV. The mass resolution for a Higgs of 130 GeV decaying into four electrons is 1.5 GeV.

### Hadronic calorimeter

The hadronic calorimeter, normally in association with the electromagnetic calorimeter, is relevant for the measurement of jet and missing transverse energy. Since the calorimeters are non-compensating, an algorithm for jet energy reconstruction is applied to correct for this effect and to add corrections for energy loss in the dead material. The intrinsic performance of the detector, in the region that extends up to $|\eta| = 3$, is of the order of:

$$\frac{\sigma(E)}{E} = \frac{50\%}{\sqrt{E}} \oplus 3\% \tag{1.4}$$

where E is the jet energy in GeV. The constant term reflects in this case the intrinsic calorimeter behavour rather than calibration or uniformity errors. The resolution degrades when the jet reconstruction is limited to a cone or when the jet points to a crack region. The response of the calorimeter is nearly linear in the range of 20 to 1000 GeV, with deviations of less than 3%.

Low $p_T$ jets can be reconstructed down to 15 GeV (25 GeV at high luminosity). The performance for forward jet tagging in the rapidity range $2 < |\eta| < 5$ is a 90% efficiency up to $|\eta| = 4$, decreasing to 50% at $|\eta| = 4.8$, in both cases with fake rates smaller than 10%.

Hadronic $\tau$ decays can be efficiently reconstructed by using the information from the calorimeters and the ID. For $\tau$-tagging efficiencies of 20%, rejection factors of $10^2$ to $10^3$ can be achieved for jets from $W$, top or $b$-quarks.

The most relevant issues for the $E_T^{miss}$ performance are: calorimeter calibration and coverage, cuts applied to sum cell energies and presence of noise and pile-up. At low luminosity, the missing energy resolution is described by:

$$\sigma(E_T^{miss}) = 0.46\sqrt{E_T} \tag{1.5}$$

where $E_T$ is the total transverse energy in GeV. At high luminosity the resolution degrades by a factor of 2.

Various cases of invariant mass reconstruction have been investigated. The typical mass resolution for $W$ bosons decaying into hadrons is 8 GeV. The mass resolution for a 100 GeV Higgs boson decaying to $b$-quarks is 15

GeV. Finally, the mass resolution of a $Z$ boson decaying to $\tau$ pairs is typically 10 GeV at low luminosity.

**Muon Spectrometer**

The combined muon measurements provide a highly performant reconstruction over a very large momentum range, from 6 GeV up to 2 TeV. The efficiency is larger than 85%. The correct matching of muon spectrometer and ID tracks allows rejection of muons from $\pi/k$ decays and identification of muons from heavy flavour decays. Muons with $p_T$ below 6 GeV can be identified using the ID and the hadron calorimeter.

The accurate momentum reconstruction allows a precise invariant mass measurement from multi-muon final states. The $Z$-boson mass can be reconstructed with a 2.5 GeV resolution, comparable to the natural width. Higgs boson decays into four muons can be reconstructed with a resolution of about 1.1% for Higgs masses below 200 GeV.

### 1.3.3 ATLAS software

In present HEP experiments like ATLAS, computing physics tools are as important as any other part of the detector. This is due to the expected huge amount of data, calculations to be performed and large number of collaborators involved in the experiment.

Information and references concerning the online and offline software to be used by the ATLAS collaboration can be found in [6] and [7], and in particular in the Computing Technical Design Report (TDR) [9].

The main software activities are described below.

- Computing Coordination and Management
- Software Project
  - Coordination and Management
  - Simulation: Generators, Geant4 framework, Digitization, fast simulation
  - Core Services: Athena framework, Databases, Geometry, Event Data Model (EDM), Graphics
  - Event Selection, Reconstruction and Analysis Tools
- Database and Data Management (including on-line activities)
- Computing Operations

– Grid, Data Challenges and World-wide operations
– Grid Tools and Services development and deployment
– Operations Management

We describe briefly the most important software tools that are used in this work.

### Athena framework

Athena is part of the Core services of the ATLAS Software Project. It is a control framework and an enhanced version of the Gaudi framework, that was originally developed by the LHCb collaboration, but is now a common ATLAS / LHCb project and is also used by other experiments .

Athena is a specific implementation of an underlying architecture, also called Gaudi, that was designed for a wide range of physics data processing applications.

Nowadays Gaudi is a kernel of software common to both ATLAS and LHC-b, while Athena includes ATLAS-specific developments.

In particular, the Athena framework

- provides a skeleton where developers plug in their code,
- gives most of the common functionality and communications between different components,
- embodies the underlying design and philosophy of the software,
- encourages a common approach and
- factors out common functionality for re-use.

Athena is written in the Python programming language and is used for running most of the ATLAS software, in particular Atlfast.

### Graphics and event display software

The ATLAS event display is called Atlantis.

The primary goal of Atlantis is visual investigation and understanding of complete events. In addition, it facilitates developing reconstruction and analysis algorithms, and allows debugging during detector commissioning. The Atlantis program is based on DALI, the event display used by the ALEPH experiment.

Atlantis is written in Java and runs on different operating systems. It uses the experience obtained from DALI, which showed that 2D projections provide excellent information for understanding events. The choice of 2D projections is driven by data and the detector layout. Sometimes nonlinear projections are helpful for track identification or extrapolation.

**Simulation software**

This part of the ATLAS Software includes the programs described below.

- Generators
  Event generators are essential tools for understanding the complex physics processes that lead to the production of hundreds of particles per event at LHC energies. Generators are used to optimise detector designs, to formulate analysis strategies, or to calculate acceptance corrections. They also allow to calculate uncertainties in physics results.

  Generators allow to model the physics of hard processes, initial- and final-state radiation, multiple interactions and beam remnants, hadronization and decays, and the interplay between all these processes.

  One of the supported generators for the ATLAS experiment is called PYTHIA. In our study, we use the PYTHIA event generator to generate events, as described in chapter 4. We describe PYTHIA in more detail later and we study its possible use in Grid environments, in particular in BOINC, in section 6.4 .

- Fast Simulation
  The ATLAS fast simulation program, Atlfast, simulates physics events, including some effects due to detector response and the software reconstruction chain. The input to the program is the collection of four-vectors from a physics event, provided by a physics event generator. Four-vectors corresponding to isolated electrons and muons are smeared and the resulting four-vectors are the output to be used in physics analysis. The calorimeter response to photons and jets is simulated as well. Jet finding algorithms are applied to the energy deposits in the calorimeter, and the resulting jets are the output for physics analysis. Other quantities calculated by Atlfast are track helix parameters and global event quantities such as the total transverse energy and missing momentum.

  The original version of Atlfast is a stand-alone FORTRAN program. Currently it has been rewritten in C++, and the structure has been modified to fit within the Athena framework. The physics results are however the same as in the original FORTRAN version.

  In our study, we use Atlfast coupled to PYTHIA to generate and reconstruct events, as explained at chapter 4. We describe Atlfast and we study its use in the BOINC Grid environment, in section 6.4 .

- Geant4
  The ATLAS full simulation programs have been based on the Geant3 simulation package since the conception of the experiment. With the development and implementation of the Geant4 program in the year

2000, ATLAS decided to use this new program. The switch-over happened in 2003. Since then, Geant4 has become the main simulation tool of ATLAS, and all new developments have been done in this environment.

The Geant4 program provides a framework and the necessary functionality for running detector simulation in particle physics experiments. The functionalities provided include optimized solutions for geometry description and navigation through the geometry, the propagation of particles through detectors, the description of materials, the modeling of physics processes (a huge effort has been invested in recent years into the development and improvement of hadronic-physics models), visualization, and many more. Geant4 allows the definition of active detector elements, performs actions within them, and writes out hits which carry information like position, energy deposit, identification of the active elements, etc. Geant4 is part of the common LCG application project, and its development is pursued as a world-wide effort, coordinated by a strong development team based at CERN.

Geant4 is not directly used in our study because we use the ATLAS Fast simulation software. Nevertheless its study is interesting since Geant4 is used in the full simulation of ATLAS events.

We discuss in depth Geant4 later and we study its possible use in Grid environments, in particular in BOINC, in section 6.4 .

- Garfield
  Garfield is a Monte Carlo computer program to simulate gaseous detectors and is programmed in FORTRAN. It is included in the LCG project and is used by most CERN experiments, in particular by ATLAS.

  Garfield is not used directly in our study but its use in Grid environments is interesting due to similarities with PYTHIA and the interest for ATLAS research.

  The work related to Garfield and its use in Grid environments is described in section 6.4 as well.

**Data Management**

Currently, a very active ATLAS area is Distributed Data Management (DDM). In particular, ATLAS has recently started to develop the final DDM system for LHC startup, called Don Quijote 2 (DQ2). The scope of this program is the management of all kind of data (event data, conditions data, user-defined file sets containing files of any type).

The DQ2 system design is based on an analysis of data management in ATLAS including managed production, global data search, managed distribution operations, and user analysis support including usages outside Grid.

Initially, Don Quijote was developed as a small program to locate files and issue file movement by users of Grid middleware. It was later expanded in the light of experience to compensate for missing functionality and finally Don Quijote eventually became the single point of failure since all production jobs depended on its functionality.

The ATLAS computing model includes the following pieces that are described in detail in [9]:

- Data Acquisition and Managed Production,
- Data Aggregation and Splitting,
- Group-level Data Production (analysis group production, calibration teams),
- Managed Data Distribution,
- File Migration,
- Data Discovery and Access,
- Physics Analysis and finally
- Regional Semi-Autonomous Grids.

CHAPTER

# 2

# The Standard Model and Extra Dimensions

## 2.1 The Standard Model

The Standard Model of particle physics (SM) describes the strong, weak, and electromagnetic forces between the fundamental particles of ordinary matter. An introduction to the SM can be found in [11].

The SM is a quantum field theory, consistent with both quantum mechanics and special relativity. All experimental tests of the SM are at present in agreement with theoretical predictions. However, the SM is not a complete theory of fundamental interactions, primarily because it does not describe the gravitational force.

The SM contains both fermionic and bosonic fundamental particles. Fermions possess half-integer spin and obey the Pauli exclusion principle. Bosons possess integer spin, do not obey the Pauli exclusion principle and are responsible for the interactions between fermions.

The SM includes a theory of the electroweak interaction, unifying the weak and electromagnetic interactions, and a theory of the strong interaction called quantum chromodynamics (QCD). All these theories are gauge theories, coupling fermions to intermediate bosons. The Lagrangian is invariant under various gauge transformations and therefore these mediating

bosons are called gauge bosons. The bosons of the Standard Model are listed in table 2.1 .

The Higgs boson, also included in the table, is the only fundamental boson which is not a gauge boson. It has not been observed, and its discovery is a major goal of present experiments, in particular of LHC experimenta.

Also the gravitons, the mediators of the gravitational interaction, are included in the table, although they are not included in the SM.

| **interaction** | **boson** | **symbol** | **spin** | **mass** ($\mathbf{GeV}/c^2$) | **charge** ($q_e$) |
|---|---|---|---|---|---|
| **electromagnetic** | photon | $\gamma$ | 1 | 0 | 0 |
| **weak** | intermediate | Z | 1 | 91.2 | 0 |
| | bosons | W | 1 | 80.4 | $\pm 1$ |
| **strong** | gluons | g | 1 | 0 | 0 |
| **gravitational** | graviton | G | 2 | 0 | 0 |
| | Higgs | H | 0 | $m_H > 114$ | 0 |

Table 2.1: The SM fundamental bosons.

The SM is invariant under gauge transformations that are elements of a unitary group called a "gauge group". The gauge group of the strong interaction is $SU(3)$, and the gauge group of the electroweak interaction is $SU(2) \times U(1)$. Therefore, the complete gauge group of the Standard Model is $SU(3) \times SU(2) \times U(1)$.

There are twelve different fundamental fermion types , or "flavours", in the SM. The proton and the neutron are made by two of these fundamental fermions: the up quark and the down quark, bound together by the strong nuclear force. Together with the electron (bound to the nucleus in atoms by the electromagnetic force), these fermions are the building blocks of ordinary matter. All the fundamental fermions of the Standard Model are listed in table 2.2.

The experimental mass values and other properties of the SM fermions and bosons can be found in reference [12].

Each particle of the SM has a corresponding antiparticle, not shown in the previous tables.

Concerning quarks masses, what is actually measured experimentally is the mass of baryons and mesons, since quarks cannot be isolated due to confinement. The mass attributed to quarks is the mass of the quark at a given renormalization scale using QCD. There are various techniques in

| ***quarks*** | **aprox. mass (GeV)** | **charge ($q_e$)** | | **leptons** | **aprox. mass (GeV)** | **charge ($q_e$)** |
|---|---|---|---|---|---|---|
| ***u*** | 0.003 | -1/3 | | ***e*** | 0.0005 | -1 |
| ***d*** | 0.005 | +2/3 | | $\nu_e$ | $\sim 0$ | 0 |
| ***s*** | 0.1 | -1/3 | | $\mu$ | 0.106 | -1 |
| ***c*** | 1.3 | +2/3 | | $\nu_\mu$ | $\sim 0$ | 0 |
| ***b*** | 4.3 | -1/3 | | $\tau$ | 1.8 | -1 |
| ***t*** | 175 | +2/3 | | $\nu_\tau$ | $\sim 0$ | 0 |

Table 2.2: Fundamental fermions of the SM

order to compute quark masses. For example, the hadron spectrum can be computed using QCD in a lattice and the input quark masses are varied until the results agree with experimental data.

Fermions can be ordered in three generations, the first one consisting of the electron, the up and down quarks, and the electron neutrino. All ordinary matter is made of first-generation fermions; all the other fundamental fermions, except neutrinos, decay into first-generation fermions and can only be produced in high-energy interactions. The second and third generations are a replica of the first generations, including particles with the same fundamental interactions, but with a different mass. For example, the electron and the muon have both half-integer spin, unit electric charge and are not sensitive to the strong interaction. The muon is however about 200 times more massive than the electron.

The electron, the electron neutrino, and the corresponding particles from the other generations, are called "leptons". Unlike quarks, they do not have strong interaction, only weak and electromagnetic, that decrease with distance. On the contrary, the strong force between quarks increases with distance so quarks can only exist in colourless combinations called hadrons. This phenomenon is known as quark confinement. These colourless combinations are either baryons composed of three quarks (the proton and neutron being the most familiar examples) or mesons composed of quark-antiquark pairs, such as pions. Baryon masses exceed largely quark masses due to the strong binding energy.

## 2.2 Beyond the Standard Model

Some examples of important predictions of the SM that have been tested experimentally are:

- The existence of the weak $W$ and $Z$ intermediate bosons, the gluon, the charm quark, the bottom quark and the top quark. All these particles have been observed experimentally after they were predicted.

- The expected properties of all these particles were experimentally confirmed, for example the mass of the $W$ and $Z$ bosons, calculated after the discovery of neutral currents.

- The Large Electron-Positron collider (LEP) at CERN tested with extremely good accuracy various predictions concerning the production and decay properties of $Z$ bosons.

The Standard Model has been therefore very successful in explaining experimental results, but cannot be considered a complete theory of fundamental interactions. The main reasons are listed below.

- *Gravity*: The SM does not include the gravitational interaction.

- *Parameters*: The SM contains 19 free parameters, including particle masses and coupling constants. These parameters cannot be calculated using the model itself.

- *Neutrino oscillations*: The first experimental deviation from the SM (as proposed in the 1970's) was the observation of neutrino oscillations, implying massive neutrinos, since massless neutrinos cannot oscillate. The SM model can however be modified to include non-zero neutrino masses. This may be simply achieved by adding 10 more free parameters, in addition to the initial 19. The pattern of all these new parameters remains however unexplained.

- *Hierarchy problem*: The SM does not answer the question why the weak force is $10^{32}$ times stronger than gravity. More technically, the question is why the Higgs boson is so much lighter than the Planck mass, since the large (quadratically divergent) quantum corrections to the the Higgs boson mass should push this mass to a huge value, unless an incredible fine-tuning between the bare mass and the quadratic corrections produces the required cancelation.

In addition, the Higgs boson, predicted by the SM, has not been observed experimentally. One of the goals of the LHC is to observe the Higgs boson.

Since the completion of the Standard Model, many efforts have been made to modify it in order to address all these problems.

Grand Unification theories appear as an attempt to reduce the large number of free parameters. These grand unified theories (GUTs) imply that the $SU(3)$, $SU(2)$, and $U(1)$ groups are actually subgroups of a single larger

symmetry group. At high energies (far beyond the reach of current experiments), the symmetry of the unifying group is recovered; at low energies, the larger symmetry reduces to the SM symmetry $SU(3) \times SU(2) \times U(1)$ by spontaneous symmetry breaking. A distinguishing prediction of these GUTs is that, unlike in the Standard Model, protons should decay. Proton decays have not been observed to date, excluding many GUT theories, in particular $SU(5)$.

Another extension of the SM intended to solve the hierarchy problem is Supersymmetry. This theory implies a massive supersymmetric "partner" for each particle of the conventional SM. These partners have the same couplings but a a different spin than conventional SM particles. The lightest supersymmetric particle has been proposed as a candidate for dark matter. Although supersymmetric particles have not been observed experimentally, this theory is at present rather attractive as a building block for other theories. One of the main research topics for LHC experiments is the search for such supersymmetric particles.

Other important extensions to the SM are Extra Dimensions (ED) theories. These theories can unify gravity with the other fundamental forces and solve the hierarchy problem as well. They are discussed in the next section.

## 2.3 Extra dimensions

A short introduction on the motivation, history and main ideas of extra dimension theories is given below. A more extensive discussion can be found in references [13] [14].

Already in the 19th century, two basic forces of nature (electricity and magnetism) were unified by Maxwell following the work of Ampère and Faraday. Once the relativistic invariance of Maxwell's theory was established, it became clear by the work of Minkowski, Lorentz, Einstein and others, that the unification of electricity and magnetism entails a unification of three-dimensional space and time into a four-dimensional space.

As soon as a relativistic theory of gravitation was available, the question of its unification with Maxwell's theory became relevant. Even before Einstein's general relativity, Nordström [15] proposed a relativistic theory with gravity described by a scalar field coupled to the trace of the energy momentum tensor. In 1914, still before the publication of general relativity, Nordström [16] proceeded to unify his theory of gravitation with Maxwell's theory adding another space dimension, inspired by Minkowski's four-dimensional space-time. The result was a flat five-dimensional world. He noticed that under certain hypothesis, the equations of this five-dimensional "Maxwell" theory include the gravitational theory. He concluded that scalar gravity in our four-dimensional world is a remnant of an abelian gauge electromagnetic theory in a five-dimensional flat space-time.

In 1919 Kaluza [17] demonstrated that the Einstein theory of gravity in five dimensions, by imposing a circular constraint on the 4th spatial dimension, yields the ordinary four-dimensional Einstein gravity and Maxwell electromagnetism. He studied the case of a five-dimensional manifold $\mathbf{M} = M_4 \times S^1$, the product of a four-dimensional space-time $M_4$ with a circle $S^1$.

To understand the principle of this unification, one can use the following toy example: a free scalar $\Phi(x_\mu, y)$ in five dimensions. The lagrangian has the form

$$L_\Phi = -\frac{1}{2}\partial_A \partial^A \Phi, \qquad A = 0, 1, 2, 3, 4 \tag{2.1}$$

In $\mathbf{M} = M_4 \times S^1$ the 5th coordinate $y$ is curled into a circle of radius R so one can impose the boundaty condition

$$\Phi(x_\mu, y) = \Phi(x_\mu, y + 2\pi R) \tag{2.2}$$

and by expanding the 5th component in harmonics, one obtains

$$\Phi(x_\mu, y) = \sum_{n=-\infty}^{+\infty} \Phi(x_\mu)_n e^{iny/R} \tag{2.3}$$

After redefining the field in the way $\phi_n = \sqrt{2\pi R}\ \Phi_n$, the action becomes

$$S_\Phi = \int d^4x \left(-\frac{1}{2}\partial_\mu \phi_0 \partial^\mu \phi_0\right) - \int d^4x \sum_{n=1}^{+\infty} \left(\partial_\mu \phi_n \partial^\mu \phi_n^* + \frac{n^2}{R^2}\phi_n \phi_n^*\right) \tag{2.4}$$

and the conclusion is that in $M_4 \times S^1$ a free massless scalar becomes

- a massless scalar $\phi_0$ and
- an infinite tower of massive scalars with masses

$$m_n = \frac{|n|}{R}; \qquad n = 1, 2, 3 \ldots \tag{2.5}$$

This infinite number of scalars is called a *Kaluza-Klein tower* (KK) and the additional scalars obtained are called KK replica. For energies much lower than $1/R$ only the zero mode is visible. This explains why the extra dimensions have not been observed since current experiments have not reached the required energies.

At the LHC, the two proton beams will collide with a center-of-mass energy of 14 TeV so the new experiments will be sensitive to KK replica with masses of 1 TeV, or equivalently to curled extra dimensions with a compactification scale $R$ of the order of $1TeV^{-1}$.

If the action of Einstein gravity is considered, instead of a simple scalar field, then an important result is obtained, namely the unification of electromagnetism and gravity. In four dimensions the action is

$$S_4 = \frac{1}{2} M_{Planck}^2 \int d^4x \sqrt{g} R_4(g) \tag{2.6}$$

where $M_{Planck} \cong 1.2 \times 10^{19}$ GeV, $g$ is the determinant of the metric $g_{\mu\nu}$ and $R_4$ is the scalar curvature. If we incluye an additional space dimension, the action becomes

$$S_5 = \frac{1}{2} M_5^3 \int d^4x dy \sqrt{G} R_5 \tag{2.7}$$

where $G = det(G_{AB})$ with $A, B = 0, 1, 2, 3, 4$, and $R_5$ is the scalar curvature in 5 dimensions. By compactifying the 5th dimension into a circle with Radius $R$, as in the case of the scalar field, the metric tensor can be expanded in harmonics in the following way:

$$G_{AB}(x_\mu, y) = \sum_{n=-\infty}^{+\infty} G_{AB}(x_\mu)_n e^{iny/R} \tag{2.8}$$

yielding again massless particles and an infinite tower of massive gravitons. The massless components of the five dimensional metric can be written in the following way

$$G_{MN}^0 = \begin{pmatrix} g_{\mu\nu} e^{\sigma/\sqrt{3}} + A_\mu A_\nu & e^{-2\sigma/\sqrt{3}} A_\mu \\ e^{-2\sigma/\sqrt{3}} A_\mu & e^{-2\sigma/\sqrt{3}} \end{pmatrix} \tag{2.9}$$

and therefore the metric $G_{MN}$ in five dimensions includes

- a massless graviton in four dimensions $g_{\mu\nu}$
- a gauge vector boson $A_\mu$, the photon, and
- a scalar field $\sigma$, called the radion.

So, the four dimensional gravity and a gauge photon are unified in five dimensions.

By expanding the five dimensional action of eq. (2.7) and keeping only the zero mass fields, it is possible to obtain the four dimensional gravity action, with the additional constraint

$$M_{Planck}^2 = M_5^3 2\pi R \tag{2.10}$$

Those ideas were further elaborated by Klein in 1926. In particular, he calculated the electric charge of the KK replica and noticed that the

charge is quantized [18]. He also obtained the relativistic generalization of Schrödinger's equation (carried out independently by many others) [13], now known as the Klein-Gordon equation. He arrived to this result starting from Kaluza's theory: a zero mass wave equation in five dimensions yields four-dimensional Klein-Gordon equations for the individual harmonics [19]. Klein also discussed the higher harmonics and the size of the radius $R$.

One can notice that the fundamental scale of the fifth dimension, $M_5$, can be arbitrarily low if the radius $R$ is sufficiently large, opening a way to explain why the Planck scale is so large. One has to take into account, however, that eq. (2.5) implies that $m_n \propto 1/R$ so if $R$ is too large, then the masses of the unobserved KK replica would be too small.

A solution to this problem is provided by theories including branes. The idea is that non-gravitational, i.e. SM, fields are confined in a sub-manifold of the full space called brane, and therefore do not possess KK replica. On the other hand, the gravitational field occupies the full space, filling the bulk space outside the brane. In some models, all the SM fields are forced to live inside the brane. In other models only SM fermions are confined inside the brane and bosons can propagate freely in the bulk.

Many different extensions of the KK theory are available. A popular example is the so called ADD model [20], with fermions confined to a brane but all gauge bosons propagating in the bulk between branes.

In this case, the couplings of KK gauge replica to fermions are equal to the SM couplings between gauge bosons and fermions except for the presence of an additional $\sqrt{2}$ factor [21].

The phenomenology of such models is discussed for example in [22]. The observation of KK excitations of the weak gauge bosons ($Z$ and $W$) using the ATLAS experiment has already been discussed [23] [24]. It has been shown that, if KK excitations $Z^*$ and $W^*$ exist, they can be reconstructed in the ATLAS detector using leptonic decays as discovery channels. In this work we discuss the possibility to detect hadronic decays of $Z^*$ and $W^*$ (see chapter 4).

CHAPTER

# 3

# Distributed Grid computing

After discussing the LHC, the ATLAS collaboration and the physics that can be extracted, we study in depth the important part of the computing challenges involved and the proposed solution to them: the Grid. In particular, we define the Grid and its different flavours together with the objectives to reach and other areas where it can be useful.

Then, we describe the Grid used by LHC experiments, the LHC Computing Grid (LCG), focusing on the ATLAS collaboration. We explain its computing model, distributed in tiers, and its infrastructure. We discuss the EGEE I project, which involves most of the current resources of LCG, its follow up EGEE II and its predecessor EDG. We describe Quattor as well, the software tool created in the scope of EDG Grid project and currently used to install and maintain the Grid nodes.

Finally we also introduce the BOINC distributed computing environment as another flavour of Grid that has been recently proposed at CERN to perform physics calculations using the spare time of desktop computers.

## 3.1 The Grid

### 3.1.1 Definition

Whereas the Web is a service for sharing information over the Internet, the Grid is a service for sharing computer power and data storage capacity over the Internet. The Grid goes well beyond simple communication between

computers, and aims ultimately to turn a global network of computers into one vast computational resource. An introduction to Grid computing is given in [25]. More detailed information can be found in [27].

The Grid takes its name from an analogy with the electrical "power grid". It is expected that "the Grid would let users exploit processing power off the Internet as easily as electrical power can be drawn from a wall socket". The meaning of this sentence is:

When you plug-in something to the electrical power, you never worry about where the electricity you are using comes from, if it is from coal, from wind or from a nuclear plant. You know that whatever you plug into a wall socket, it will get the electrical power you need to do the job. In the Grid, when you sit in front of your computer your only concern is to run your job (a simulation, analysis...). You just will that you will get the computing power and storage capacity you need to do the job and you do not worry about where it comes from.

Anyway it is important to notice that there is no exact definition for the term Grid, as it is very recent and many different kinds of distributed computing environments have been called Grids by different authors.

Ian Foster, considered as one of the 'fathers' of the Grid, tries to assess this issue giving a formal definition [26]. For him, a Grid is a system that

- *coordinates resources that are not subject to centralised control ...*
  A Grid integrates and coordinates resources and users that live within different control domains. For example, the user's desktop vs. central computing; different administrative units of the same company; or different companies; and addresses the issues of security, policy, payment, membership, and so forth that arise in these settings. Otherwise, we are dealing with a local management system.

- *... using standard, open, general-purpose protocols and interfaces*
  A Grid is built from multi-purpose protocols and interfaces that address such fundamental issues as authentication (check of the identity of the user), authorisation (check if the user has the rights to use the Grid resources), resource discovery, and resource access. It is important that these protocols and interfaces be standard and open. Otherwise, we are dealing with an application specific system.

- *... to deliver non trivial qualities of service.*
  A Grid allows its constituent resources to be used in a coordinated way to deliver various qualities of service, relating for example to response time, throughput, availability, and security, and/or co-allocation of multiple resource types to meet complex user demands, so that the utility of the combined system is significantly greater than that of the

sum of its parts. This feature is very important in order to solve complex scientific and technical problems involving collaborative environments (what has recently being called e-Science).

For David Anderson, considered as the 'father' of BOINC, that is a distributed computing environment making profit of unused desktop computers, those should not be included in the general definition of Grid and the term Distributed Computing System is more appropriate [28]. BOINC is discussed in depth in section 3.3 .

In this work we consider Distributed Computing Systems like BOINC as a flavour of Grid, since this is the point of view of CERN [25] and it is also explicitly considered as such by Ian Foster [26].

### 3.1.2 Some key concepts

The following concepts are common to most kinds of Grids:

**Virtual Organisation (VO)**
: They are made by people who share a common goal, like the people collaborating in each of the LHC experiments. To achieve their goal, they need to perform several types of demanding calculations which can not be handled with the resources belonging to just one of the participants, or they need to access each others' databases in a well-defined and secure way. Then, a certain Grid can have many different VOs that choose to share their resources, meaning direct access to computers, programs, files, data, sensors and networks. This sharing must be arranged in a controlled, secure, and flexible way, usually for a limited period of time.

**Middleware**
: It is the software that organises and integrates the different computational facilities belonging to a Grid. Its main role is to automate all the "machine to machine" negotiations required to interlace the computing and storage resources and the network into a single computational "fabric". It enables the various elements (servers, storage, networks, etc.) to participate in a unified Grid environment.

**Metadata**
: This is essentially "data about data". Metadata play a crucial role as they contain all information about, for example, how, when and by whom a particular set of data was collected, how the data is formatted, and where in the world it is stored (sometimes at several locations). It is a key ingredient for middleware.

**Testbeds**

A testbed is a dedicated Grid infrastructure implemented and deployed to test middleware and application development. It is a "real Grid", whose limit is mainly the restricted access, limited to small groups of developers and scientists during limited periods of time. It is made up of one or more nodes (computer centres contributing resources to the testbed). Each node contains a certain number of computers, which may be playing different roles.

### 3.1.3 Middleware components

The main components of the middleware used by most of the current Grid projects are described below.

- **User Interface (UI)**: allows users to access the Grid facility and receives the input Grid jobs written in Job Description Language (**JDL**).
- **Resource Broker (RB)**, the module that receives users' requests and queries the Information Index to find suitable resources.
- **Information Service (IS)**, which can reside on the same machine as the Resource Broker, keeps information about the available resources.
- **Storage Element (SE)**: provides storage space.
- **Replica Manager (RM)**: coordinates file replication across the Grid from one Storage Element to another. This is useful for data redundancy but also to move data closer to the machines which perform the computation.
- **Replica Catalogue (RC)**: keeps information about file replicas. A logical file can be associated to one or more physical files that are replicas of the same data. Thus a logical file name can refer to one or more physical file names.
- **Worker Node (WN)**: processes input data.
- **Computing Element (CE)**: receives job requests and delivers them to the Worker Nodes, that perform the real work. The Computing Element provides an interface to the local batch queueing systems. A CE can manage one or more WNs. A Worker Node can also be installed on the same machine as the Computing Element.

We have to keep in mind that the details and names of this middleware components may change slightly from one Grid project to another. For instance, we study in section 6.1 the case of BOINC Grids middleware.

### 3.1.4 The Globus toolkit

Most of the Grid projects are being built on protocols and services provided by the Globus Toolkit [29], a software being developed by the Globus Alliance, which involves primarily Ian Foster's team at Argonne National Laboratory and Carl Kesselman's team at the University of Southern California in Los Angeles.

The toolkit provides a set of software tools to implement the basic services and capabilities required to construct a computational Grid, such as security, resource location, resource management, and communications.

Globus includes programs such as:

- **GRAM** (Globus Resource Allocation Manager), which figures out how to convert a request for resources into commands that local computers can understand
- **GSI** (Grid Security Infrastructure), which provides authentication and authorisation.
- **MDS** (Monitoring and Discovery Service) to collect information about resources (processing capacity, bandwidth capacity, type of storage, etc)
- **GRIS** (Grid Resource Information Service) to query resources for their current configuration, capabilities, and status
- **GIIS** (Grid Index Information Service) which coordinates arbitrary GRIS services
- **GridFTP** (Grid Service for File Transfer) which provides a data transfer mechanism with FTP functionality and GSI security
- The **Replica Catalogue** *, a catalogue that allows other Globus tools to find where other replicas of a given dataset can be found on the Grid
- The **Replica Management system**, which ties together the Replica Catalogue and GridFTP technologies, allowing applications to create and manage replicas of large datasets.

There are two main reasons for the strength and popularity of the Globus toolkit: its object oriented approach and the fact that it is available under an open source licensing agreement which allows to use the software freely and add improvements to it.

For instance, the current middleware being developed for the Grid used by the LHC experiments originated from Globus and it uses most of the services described before.

*'Catalog' in US spelling

### 3.1.5 Overview of a Grid job

The life cycle of a Grid job is explained in figure 3.1.

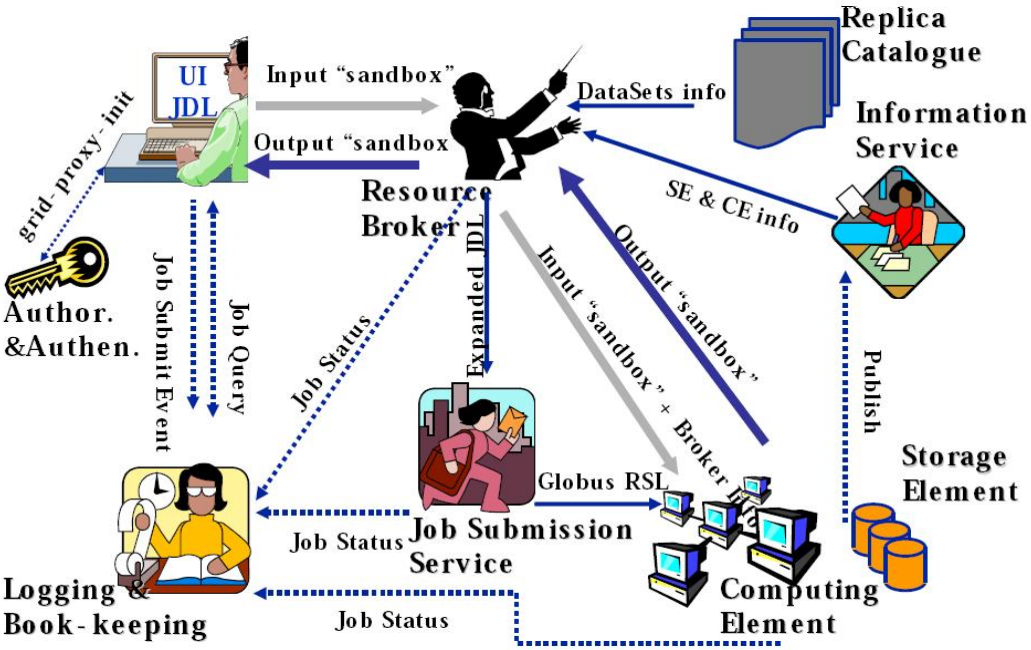


Figure 3.1: Overview of the life cycle of a Grid job

- The user creates the job written in JDL language or uses a Graphical User Interface (GUI) that creates it.
- The user sends the job to the Grid using the User Interface (UI).
- At this stage, it is checked the identity (authentication) and the rights of the user to access to the Grid resources (authorisation).
- If it is accepted, the job is registered by the Logging and Book-keeping service (LB).
- Then the Resource Broker (RB) receives the job, queries the Information Index (II) to find suitable resources and sends it to the allocated resources.
- The job is sent to a worker node with computing elements, storage elements and connected to a replica catalogue to retrieve and use the required files. At this stage, the status of the job is registered from time to time by the LB service to be queried by the user if desired.
- Finally, the result comes back to the RB, which sends it back to the UI and logs the result to the LB service.

### 3.1.6 Challenges and benefits of the Grid

The Grid has been proposed as a solution to the current challenges that scientists of different research fields have to face regarding computing [25]:

- The amount of data needed is huge and the data is stored in different institutions. It might take a long time to copy the data to one central computer in order to analyse it. Ideally the computation should be performed where the data are.
- The amount of calculations to perform is huge. It would take too much time in one computer, or even in a cluster of computers.
- A scientific team has members around the globe and wants to share large amounts of data and to perform complex analysis of the data in a short time.

In addition, other key benefits that a computing Grid can provide are the following [30]:

- The cost of maintaining and upgrading the necessary resources for such computing challenges are more easily handled in a distributed environment, where individual institutes and participating national organisations can fund local computing resources and retain responsibility for these, while still contributing to the global effort.
- In a distributed system, there are no single points of failure. Multiple copies of data and automatic reassigning of computational tasks to available resources ensures a load balancing of resources and facilitates access to the data for all the scientists involved, independently of geographical location. Spanning zones also facilitates round-the-clock monitoring and support. This is important since in world wide collaborations there is always some research institutes using the Grid at any time, so the facility has to be available 24/24h.

As an example of such a round-the-clock Grid involving different scientific communities distributed geographically, the ATLAS collaboration has approximately 1800 physicists participating, coming from more than 150 universities and laboratories in 34 countries, as we show in figure 3.2.

The Grid can be very useful for many different kinds of fields. Currently, some of the fields where Grid computing can be used are:

- **High Energy Physics (HEP)**: To explore fundamental particles and their interactions, HEP experiments, in particular the LHC (see 1.2), will soon produce about 15 Petabytes of data per year, corresponding to 30 million CDs. Thousands of physicists from many universities around the world will work in the analysis of these data.


Figure 3.2: The different countries involved in the ATLAS collaboration.

- **Flood forecasting**: The recent extreme floods in Europe resulted in scientific and societal concerns about the reliability of short-term quantitative meteorological and flood forecasts. The utility of Grid technology to support flood crisis teams in international rivers is currently being studied. Flood forecasting requires quantitative precipitation forecasts based on meteorological simulations that require high-performance computing resources and infrastructure that normally are not available locally on the appropriate scale.

- **Climate prediction**: Climate change, and our response to it, are issues of global importance, affecting food production, water resources, ecosystems, energy demand, insurance costs... The currently running experiments need to perform climate model simulations thousands of times to produce a forecast of the climate for the XXI century. In the past, estimates of climate change were made using a very small ensemble (tens rather than thousands) of model runs. By using Grid technologies the performance can be considerably improved.

- **Ozone studies**: Earth scientists keep track of the level of atmospheric ozone with satellite observations. For this task alone, they download, from space to ground, about 100 Gigabytes of raw images per day (the equivalent of about 150 CDs).

- **Medicine**: Vascular diseases are a major medical problem, particularly in the developed countries and are of major concern in Europe. Treatment often involves surgery. Grid computing provides a modern way of addressing problems of this kind and currently Grid-based prototypes are being developed for pretreatment planning in vascular interventions and surgical procedures through real-time interactive simulation of vascular structure and flow. The system consists of a distributed real-time simulation environment, with user interaction in Virtual Reality. A 3D model of a patient's arteries serves as input to the environment for blood flow calculations.

- **Biomedical studies**: Ten years ago, biologists simulated single small molecules on computers. Now, using Grid technologies, some projects work on the simulation of thousands of molecular drug candidates to see how they would interact with specific proteins. Other projects simulate and predict the structure of proteins to be used in medical studies of protein-related diseases. Examples of such studies are the malaria drug discovery survey called WISDOM (Wide In Silico Docking On Malaria) [31] made by the EGEE project (see 3.2.2) and the Predictor@home project [32] using the BOINC environment [33] [34] (see 3.3) to predict structure of proteins to address critical biomedical questions.

- **Genome**: When human genome and protein structures are known, scientists can use them to research disease treatments and cures. Unlocking the secrets of the human genome would be impossible without the computerised analysis of massive amounts of data, including the sequence of the three billion chemical units that comprise our DNA. An example of one of such studies is the Human Proteome Folding project [35], one of the projects of the IBM's World Community Grid project [36], which uses the BOINC environment.

In particular, the Instituto de Física Corpuscular (IFIC) in Valencia is part of the ATLAS collaboration. It has contributed to the CrossGrid project [37], that involves four different projects related to particle physics, medical research, flooding prediction and pollution and weather forecasting.

### 3.1.7 Different kinds of Grids

Concerning the different kinds of Grids, they have been classified in the following way [25]:

- **National Grids**: The idea behind National Grids is to couple high-end resources across a nation. This provides a strategic "computing

reserve" and allows substantial computing resources to be applied to large problems in times of crisis, such as to plan responses to a major environmental disaster, earthquake, or terrorist attack. Furthermore, such a Grid will act as a "national collaboratory", supporting investigations of complex scientific and engineering problems, such as global climate change, space station design, and environmental cleanup. For example, the UK has a major e-Science program [38] dedicated to develop a major national Grid: the UK Grid for Particle Physics (GridPP) [39].

- **Private Grids**: sometimes called local-Grids or intra-Grids, they can be useful in many institutions (hospitals, corporations, small firms, etc). They are characterised by a relatively small scale, central management and common purpose and, in most cases, they probably need to integrate low-cost commodity technologies. In fact, commercial solutions for such private Grids are already available, such as Entropia's DC Grid[†] [40], and likely to grow in sophistication over the next years.

- **Project Grids**: They are created to meet the needs of a variety of multi-institutional research groups and multi-company "virtual teams", to pursue short or medium-term projects (scientific collaborations, engineering projects). A Project Grid is typically built ad hoc from shared resources for a limited time, and focuses on a specific goal. Typically, this is something that a self-motivated team could set up, without need to apply to any major Public Grid infrastructure for permission. The LCG Project at CERN (see section 3.2.1) is an example of such a Grid for a particular high-energy physics community.

- **Goodwill Grids**: This kind of Grids are specialised on distributing computing power. They are subscribed by anyone owning a computer at home who wants to donate some computer capacity to a good cause. For this reason these type of Grids are also called "Public Resource Computing (PRC)" environments since they typically use non private resources to perform the calculations [‡]. To date, activities in this area has been limited to the various "@home" projects using the BOINC distributed computing environment (see 3.3). In most cases, they provide an attractive screen saver and give credits for the performed work to motivate the participants. Examples of such a Grid are the famous SETI@home project [41] (looking for extraterrestrial signals) and also the CERN-managed one, LHC@home, discussed in depth in section 6.3.

[†]Entropia ceased commercial operations in 2004, although no formal announcement to that effect was ever made. Currently, the official web site is offline.

[‡]Project Grids, like CERN's LCG, also use public resources in the sense that they are not privately funded.

More information, in Spanish, about this kind of Grids can be found in reference [42].

- **Peer-to-peer Grids**: They are specialised in distributing data and depend on people sharing data between computers. The name peer-to-peer suggests that there is no central control, requiring no third-party intervention. Compared to Goodwill Grids, the idea is that you get in kind for what you give: access to data files for sharing your own, for instance. Examples of this kind of sharing data Grids are Napster and Gnutella. It is important to notice that currently the BOINC Grid environment is studying to use also a peer-to-peer Grid, in particular BitTorrent [43], to share data replicas among the different Worker Nodes to decrease the load on the data servers.

- **Consumer Grids**: The resources are shared on a commercial basis, rather than on the basis of goodwill or mutual self-interest. Companies or other organisations rent distributed resources, and the owners of these resources are paid for the computing power or data storage capacity they provide, by a "middleman" in charge of the middleware.

ATLAS uses LCG, which is a kind of Project Grid in this classification. We study in depth this LCG Grid in the following work.

The BOINC Grid, which is a kind of Goodwill Grid, is also used at CERN and it is used also for ATLAS research. Although BOINC is a Goodwill Grid in the sense that is mostly used to execute jobs in computers from volunteers, it can also be used to take profit of the spare time of private computers. In particular in this work we show the execution of Grid jobs in a private farm at CERN and in computers from several Spanish institutions as well.

## 3.2 LHC Computing Grid

### 3.2.1 The LCG project

The Large Hadron Collider (LHC), which is expected to start to operate in 2007. It will start at a center of mass energy of 900 GeV and it will ramp up to 7 TeV per beam in the second half of 2008. It will collect approximately 15 Petabytes of data annually. Access to this experimental data needs to be provided for about 5,000 scientists in some 500 research institutes and universities worldwide who are participating in LHC experiments. In addition, all the data needs to be available over the estimated 15-year lifetime of the LHC. The analysis of the data, including comparison with theoretical simulations, requires of the order of 100,000 CPUs using 2004 measures of processing power.

The aim of the **LHC Computing Grid (LCG)** project[§] [30] [44] is to develop, build and maintain a distributed computing infrastructure for the storage and analysis of data from the four LHC experiments.

A traditional approach would be to centralise all this capacity in one location near the experiments, as for LEP. In case of the LHC, however, the Computing Grid distributed model for data storage and analysis was chosen due to the benefits obtained by a Grid environment (see 3.1.6).

Of course, a distributed system presents also a number of significant challenges. These include ensuring adequate levels of network bandwidth between the contributing resources, maintaining coherence of software versions installed in various locations, copying with heterogeneous hardware, managing and protecting the data so they are not lost or corrupted over the lifetime of the LHC, and providing accounting mechanisms so that different groups have fair access, based on their needs and contributions to the infrastructure. These are some of the challenges that the LCG Project is addressing.

### LCG hierarchical dataflow

The LCG Project uses a distributed four-tiered model.

**Tier-0 :** The original raw data emerging from the data acquisition systems of the experiments is recorded by the Tier-0 centre at CERN. The first-pass reconstruction takes place at the Tier-0, where a copy of the reconstructed data is stored. The Tier-0 distributes a second copy of the raw data to the Tier-1 centres associated with the experiment. Additional copies of the reconstructed data are also distributed to the Tier-1 centres.

**Tier-1 :** The role of the Tier-1 centres varies depending on the experiment, but in general they have the responsibility for managing the permanent data storage (raw, simulated and processed data) and providing computational capacity for re-processing and for analysis processes that require access to large amounts of data. At present 12 Tier-1 centres have been defined, most of them serving several experiments. In particular, 10 of them serve the ATLAS collaboration.

**Tier-2 :** The role of Tier-2 centres is to provide computational capacity and appropriate storage services for Monte Carlo (MC) event simulation and for end-user analysis. The Tier-2 centres obtain data as required from Tier-1 centres, and the data generated at Tier-2 centres is sent to Tier-1 centres for permanent storage. More than 100 Tier-2 centres have been proposed.

[§]Recently, the LCG collaboration changed its name to Worldwide LHC Computing Grid (WLCG) but the project kept its name. Then, currently the LCG project is part of the WLCG collaboration.

**Tier-3 :** Other computing facilities in universities and laboratories will take part in the processing and analysis of LHC data. They are called Tier-3 centres. Any group of scientists associated to a Tier-2 centre (with minimum requirements for computation power and data storage) can be a Tier-3. These centres are outside the scope of the LCG Project, although they must have access to the data and analysis facilities, as decided by the experiments.

In Spain, there is one Tier-1 multi-experiment facility at Barcelona, the Port d'Informació Científica (PIC), giving support to ATLAS, CMS and LHC-b.

Concerning the Tier-2 Spanish centres, the solution has been federated Tier-2 facilities involving different research institutes.

There is one Tier-2 for ATLAS involving the Instituto de Física Corpuscular (IFIC) in Valencia, the Instituto de Física de Altas Energías (IFAE) in Barcelona and the Universidad Autónoma of Madrid (UAM).

There is another Tier-2 for CMS involving the Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT) in Madrid and the Instituto de Física de Cantabria (IFCA) in Santander. There is finally another Tier-2 for LHC-b formed by the University of Barcelona (UB) and the University of Santiago de Compostela (USC).

#### LCG infrastructure

LCG consists of a set of services and applications running on the Grid infrastructures provided by the LCG partners. These infrastructures at present are provided by the EGEE project (see section 3.2.2) in Europe, the Open Science Grid (OSG) [¶] project [45] mostly in US and the Nordic Data Grid Facility (NorduGrid) [46], mostly in the Nordic European countries.

We focus in the following on the EGEE infrastructure as it is used by the West European sites, in particular by the Spanish ones.

### 3.2.2 The EGEE project

The Enabling Grids for E-sciencE (EGEE) project[‖] [49] is intended to provide access to major computing resources, independent of geographic location. The LCG project is the primary production environment that uses the

[¶] The OSG project is a continuation of Grid3, a community Grid built in 2003 as a joint project of the US Grid projects iVDGL-VDT, GriPhyN and PPDG, and the US participants in the LHC experiments ATLAS and CMS.

[‖] At the beginning, the acronym stood for 'Enabling Grids for E-science in Europe' but it was changed when the project was extended to non-European sites.

EGEE project infrastructure. Then, from the WLCG collaboration point of view, we can see EGEE as "the operational Grid instance integrating many national Grids and the majority of the sites that will provide capacity for LHC experiments" [50].

The EGEE project primarily concentrates on three core areas:

- To build a consistent, robust and secure Grid network that will attract additional computing resources.
- To continuously improve and maintain the middleware in order to deliver a reliable service to users.
- To attract new users from industry and science and to ensure the high standard of training and support needed.

The project started in April 2004 with the objective of building a permanent European Grid infrastructure that could serve a broad spectrum of scientific applications reliably and continuously, providing Grid services to scientists throughout Europe.

Funded by the European Commission, the EGEE project community has been divided into 12 partner federations (led by CERN), consisting of over 70 contractors and over 30 non-contacting participants covering a wide-range of both scientific and industrial applications and involving major computer centres in Europe and leading American and Russian centres.

The work being carried out in the project is organised into 11 activities. Two pilot application domains were selected to guide the implementation and certify the performance and functionality of the evolving infrastructure. One is the LCG project, supporting physics experiments, and the other involves Biomedical Grids, regarding bio-informatics and health-care.

### EGEE II

The EGEE project was conceived as the first two-year phase of a four-year programme. EGEE was due to end on 31 March 2006 and a follow up project, EGEE-II [51], started on 1 April 2006.

By the end of 2005, the 800 scientists and engineers working on EGEE (from five different continents) were managing an infrastructure sharing the power and storage from more than 10.000 dedicated computers located at over 200 sites worldwide. From October 2004 to October 2005 about two million jobs were successfully run on this Grid.

Currently, EGEE-II consists of some 20,000 CPUs available in addition to 5 Petabytes of storage capacity and maintains 20,000 concurrent jobs on average.

More than 20 applications from scientific domains including Earth observation, climate prediction, petroleum exploration and drug discovery are running on this infrastructure.

EGEE-II continues the work of EGEE in order to build an international computing infrastructure for science, extending this infrastructure to other countries and projects. It also increases the number of scientific disciplines supported, with plans to include fusion science and continuing to look for new communities with high performance computing needs as well.

The EGEE-II Consortium consists of more than 90 partners from 32 countries, grouped into 12 federations and representing almost all major and national Grid efforts in Europe, and projects from the USA and Asia as well. In addition, a number of related projects being submitted to FP6 calls will extend the infrastructure further. Examples of such areas are:

- Latin American countries with European partners (Argentina, Brazil, Chile, Cuba, Mexico, Peru and Venezuela together with Italy, Portugal and Spain), by the e-Infrastructure shared between Europe and Latin America (**EELA**) project [52]
- South Eastern European countries (Albania, Bosnia- Herzegovina, Bulgaria, Croatia, FYR of Macedonia, Greece, Hungary, Serbia - Montenegro, Romania, Turkey), by the South Eastern European Grid (**SEE-Grid**) project [53]
- Mediterranean countries (Turkey, Algeria, Morocco, Italy, Cyprus, UK, Egypt, Arabian Republic of Syria, Israel, Jordan, Tunisia, Palestine, Spain and Malta, apart from CERN), by the (**EUMedGRID**) Project [54]
- Baltic States (Lithuania, Latvia and Estonia), by the **BalticGrid** project [55]
- China, by the **EUChinaGRID** Project [56]

Combined with other related projects spun out from or affiliated with EGEE and EGEE-II, this demonstrates the incubator role of the EGEE project.

### 3.2.3 The EDG project

The widely recognised success of past Grid projects coordinated by CERN has been a key factor in generating support to follow-up projects like EGEE.

This is the reason why it is worth mentioning at least one of them: the European DataGrid (EDG) project [57]. It began in 2000 and finished in 2004.

For instance, the middleware of LCG is based on the one of the EDG project together with the Virtual Data Toolkit (VDT) middleware [47], used by the iVDGL Project [48].

In the year 2000, one of the challenges for building a Grid was the lack of software needed to keep it ticking over (the middleware). This is the reason why CERN, together with a host of leading European research centres, took the initiative for the European DataGrid (EDG) project, in order to develop a testbed for Grid technologies.

EDG was built on a software toolkit for Grid technology known as Globus, developed in the US, as well as other software packages, and used these to build a functioning Grid testbed (see 3.1.2). The project involved more than 100 computer engineers, who generated some 300 000 lines of code. In 2002, EDG middleware managed to connect computing resources at some 40 major centres, including extra-European sites in Russia, South-Korea and Taiwan.

In collaboration with the LHC experiments ATLAS and CMS, a number of highly demanding computational challenges were successfully carried out. This proved that many components of the EDG software were ready for use in future projects.

### 3.2.4 The Quattor software management tool

One of the projects started in the scope of the EDG project is the Quattor [58] administration toolkit system. Its name is a recursive acronym which stands for "quattor is an administration toolkit for optimizing resources". Quattor is currently being used at CERN and in many other Grid facilities to manage and configure the Grid nodes.

Quattor is a large scale fabric management system intended for managing medium to very large (more than 1000 nodes) clusters. It provides a portable and modular software for the automated installation, configuration and management of clusters and farms running UNIX flavours, like Linux and Solaris ** [59]. Development and maintenance is coordinated by CERN (IT department) in collaboration with other partner institutes.

Quattor was created to extend the functionality and to solve the problems of LCFG [117]. LCFG, which stands for "Local ConFiGuration system", was developed at the University of Edinburgh and a modified version, called EDG LCFG, was used initially to configure and manage prototype testbed clusters for the EDG project. Finally, the Quattor toolkit was developed using an architecture similar to LCFG and was used in most

**Solaris is a computer operating system developed by Sun Microsystems. It is certified against the Single Unix Specification as a version of UNIX. Then, it is considered a UNIX system-like, as well as Linux.

EDG sites. Currently Quattor is used to manage and configure a majority of LCG sites.

### Quattor design and architecture

The Quattor information model is based on the distinction between the desired state and the actual state. The desired state is registered in a fabric-wide Configuration Database (CDB), using a specially designed configuration language to express and validate configurations, composed of reusable hierarchical building blocks called templates. Configurations are propagated to the managed nodes.

Quattor involves the following parts:

- CDB: database to manage the configuration of the nodes and clusters composed by text entries organized in text files.
- Software packages: they involve binaries, scripts, configuration files, etc. In the case of Linux they are packaged in RPM Package Manager (RPM) files for distribution. In Solaris they are packaged as Solaris Package (PKG) files.
- Components: they are configuration scripts written in Perl programming language that can be executed automatically by Quattor or in any other way, as decided by the administrator.

The following subsystems running on the nodes handle the managing of software packages and the configuration of local services:

- The Software Package Management Agent (SPMA) handles local software installations using the system packager (RPM or PKG). Packages can be stored and managed centrally in Software Repositories (SWRep).
- The Node Configuration Manager (NCM) subsystem configures and reconfigures local system and Grid services using a plug-in component framework.
- A subsystem called Automated Installation Infrastructure (AII) handles the initial node installation, configuring the system installer (KickStart/Anaconda, JumpStart).

We discuss Quattor in more detail in section 5.3.

## 3.3 The BOINC distributed computing environment

The Berkeley Open Infrastructure for Network Computing (BOINC) project [33] can be seen as a good example of "Goodwill Grid" or "Public Resource Computing" environment, in the classification presented in section 3.1.7 .

It is an open source software platform for distributed computing that uses volunteered computer resources. In short, it uses idle CPU cycles from the computers that participate, i.e. the CPU power that is not in use by the computer, to perform scientific calculations.

It can therefore be seen as a specialised Grid suitable to run applications that are "pleasantly parallel" so that it is possible to distribute the calculations to a number of machines that do not communicate with each other. Anyway, some new projects like Feynman@home have announced that they will incorporate this feature. In the case of Feynman@home, this is mandatory since in order to be able to perform high order Feynman diagrams calculations, a communication is required, as higher order diagrams require the results of the previous ones.

In addition, BOINC suitable applications have high CPU requirements relative to the size of the input and output data. The reason is that at present BOINC is focused towards CPU calculations instead of data transfers .

On the other hand, the infrastructure required to set up a BOINC Grid is very simple, requiring only a single server, although it can be distributed to different machines for performance reasons, since client machines are typically provided and maintained by volunteers.

The applications that are run in a BOINC project must have public appeal so that participants are willing to volunteer their resources. Outreach is a very important part of this kind of project to give the public a feeling of direct participation in a scientific project. Attractive screen savers are often provided together with a credit based ranking system for the participants.

It is important to distinguish between 'users' and 'participants' as in the BOINC-related papers they are commonly used indistinctively. In BOINC, a 'user' is typically a participant who provides computer power installing the BOINC software in his or her private computer to execute the Grid jobs voluntarily, i.e. a user is the owner of one or more Grid Worker Nodes. In LCG, the concept 'user' refers to scientists who send the jobs to be executed in the Grid Worker Nodes. Here we use the LCG meaning as the most widely used in HEP and we use the word 'participant' to refer to volunteers who own Worker Nodes connected to the Grid.

The first project running on BOINC was the well known SETI@home

[41], to analyse data taken from radio telescopes looking for extra-terrestrial intelligence. It has delivered more than 9 million years of aggregate computing time and it has attracted more than 5 million volunteered CPUs. Currently, BOINC is being used by many different projects in domains like Physics, Medicine and Climate Prediction.

One of the leading BOINC projects is LHC@home, managed by CERN. It simulates particles circulating around the LHC ring in order to study the long-term stability of their orbits. We discuss in more detail this project in section 6.3.

### 3.3.1 Overview of a BOINC job

We describe below a typical cycle for sending, computing and receiving a BOINC job (see figure 3.3).

We can compare it with the equivalent process involved in a generic LCG job (see section 3.1.5). We have to take into account that, in BOINC, most Grid middleware components (see section 3.1.3) are in the so called BOINC servers except the Worker Nodes, which provides computational power, located in the BOINC clients, the machines of the volunteer participants. The different parts of the BOINC server and client are discussed in depth in section 6 .

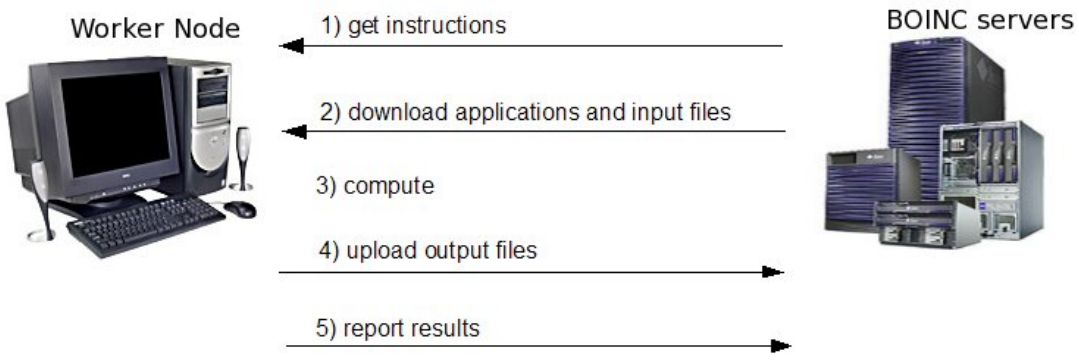


Figure 3.3: Overview of the life cycle of a BOINC job.

1. The BOINC client (Grid Worker Node) detects that there are no jobs to process in the queue or that the amount of work is too small. Then, it contacts the project's BOINC scheduling server to get instructions. This set of instructions depends on the client machine (architecture, RAM, free disk space...) and may include several Grid jobs at the same time. In addition, BOINC projects can support several "applications", and the server may send work to the client from any of them.

2. The client downloads the BOINC job(s) to compute including the executable and the required input files from the Data Server. If any of those binaries or input data have been already delivered by another executed job, they will not be downloaded.

3. The client machine runs the job application programs, producing output data.

4. The output files are uploaded to the BOINC Data Server.

5. Finally, the BOINC Scheduling Server is contacted again by the client, the results are reported and more work is requested. This cycle is repeated indefinitely until the user chooses to select a different BOINC project or to uninstall the BOINC client software entirely from his or her computer, stopping to contribute to this Grid project.

### 3.3.2 BOINC credit system

The BOINC Database keeps track of the work done by each computer; this is called credit. The credits can be used as a reference for the 'Quality of Service' of each participant's CPU.

Credits are useful for BOINC managers because they give information about submitted jobs and about BOINC clients. They can be also used to detect and correct problems. For instance, if all the jobs give always too many credits for a given client machine, then there is probably a problem with this machine and BOINC can be instructed to not to use it.

In addition, credits are important due to the motivation they provide to participants. This motivation is a key concept in BOINC, as most of the CPU used for Grid computations is provided by volunteers.

Those participants can 'compete' to have as many credits as possible. They can check the obtained credits for their donated CPU time using the Participants Web Interface or the BOINC manager utility. They can also compare themselves with the other participants, and their ranking position, at different monitoring and statistics web sites [60] like for instance the BOINC Synergy one [61].

Participants can join BOINC 'teams' to combine their credits and compete with the other teams. Some of those teams support web sites with advanced monitoring and statistics facilities to compare the credits and ranking of participants and teams (see for instance [61]). Some of those web sites were used at CERN, as a complement to the BOINC monitoring and management tools, in order to monitor daily the completed jobs and the machines status of a farm working for the LHC@home project at CERN. We give more details about this in sections 6.3 and 6.5.

To ensure that Credit is granted fairly and properly, most BOINC-powered projects work as follows:

- Each BOINC job may be sent to several computers.
- When a computer reports a result, it claims a certain amount of Credit, based on how much CPU time was used.
- When sufficient results have been returned, this number being set by each BOINC project, a BOINC Validator compares them and if results agree, this output is called 'canonical' and the participants are granted Credit based on the project rules.

In general, at least three Results are returned and BOINC calculates the average value of claimed Credits. An exact explanation of the current default rules and those specified by the different BOINC projects can be found in reference [34].

The process is shown in figure 3.4 where two different clients receives a copy of the same job at different times and taking different times to compute. Then, they claim different credits but the same final value is assigned to both of them.

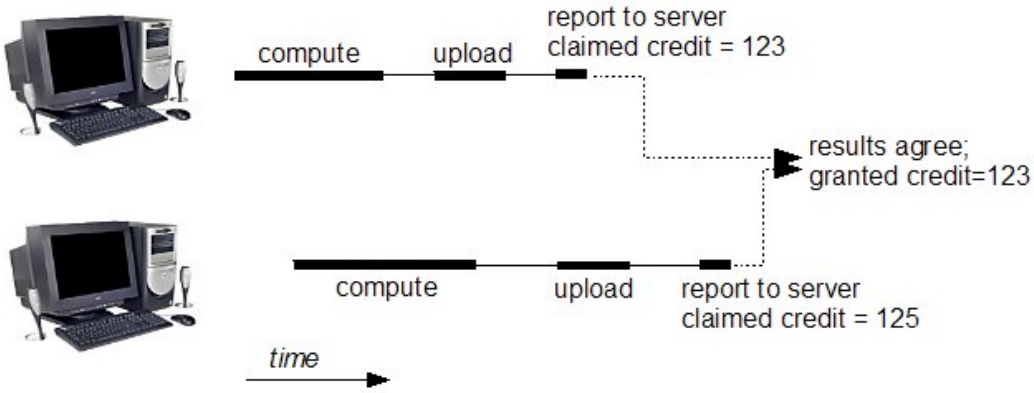


Figure 3.4: Sketch about describing BOINC credits are granted.

### 3.3.3 The future of BOINC

The combination of Public Resource Computing and Project Grid computing, in particular BOINC and LCG, has been studied. Bridges for sending

jobs between Project Grids and BOINC have been tested and built for the LCG [44] and NorduGrid [46] middleware.

For instance, there is a bridge from BOINC to LCG-2 middleware (see 5.1) allowing BOINC jobs to run on LCG resources [62]. In particular, researchers at CERN have set up a system where jobs submitted to the Grid are sent either to a BOINC project or to a GRAM job manager (see 3.1.4). This code has been already fed back and included in the official BOINC software sources [33].

The idea behind this bridge is to improve the overall performance of a BOINC platform using a mix of cycle-scavenging [††] software (in this case, BOINC) and dedicated resources (LCG sites in the example). In reference [63] a description of the possibility to guarantee a 'hard stochastic quality of service' mixing those two kinds of Grid technologies is given.

Another recent example, is the implementation in the Condor [‡‡] middleware [64] of a method to allow to run BOINC when there are no Condor jobs to process or interactive users occupying a given machine. This capability is described in the Condor manual [65].

We can also mention that the Lattice project [66], from the University of Maryland, is developing a Grid system that integrates Globus, BOINC, and several other software components.

It is also worth mentioning that BOINC is currently studying the use of a peer-to-peer Grid, in particular BitTorrent [43], in order to share data replicas among the different Worker Nodes (there are not Storage Elements in BOINC so BOINC clients are at the same time Worker Nodes and Computing Elements). In that way, when a BOINC client requires data to perform a computation, instead of looking for the BOINC data servers it will ask other clients to try to download these data from them. This procedure will decrease significantly the load of the BOINC data servers and will normally boost the download speed of the data. In addition, this procedure will allow BOINC projects to have replicas of the data to recover them if needed.

It has also been considered the possibility of storing permanently data on the clients having different replicas of the data. In this case, the clients would be at the same time Grid Worker Nodes, Computing Elements and Storage Elements.

In chapter 6, the BOINC environment is discussed in detail and the applications ported to BOINC as well as current projects, installations and

[††]This is the name for the software that takes profit of non used CPU cycles, i.e. the computation power which is not used when the computer is idle.

[‡‡]Condor is a Grid Project started in 1988 at the University of Wisconsin. The LCG-2 middleware originated from Condor, EDG and Globus among others.

testbeds are also described. A comparison between the LCG Grid and BOINC is made in chapter 7 .

# Part II

# Data analysis

CHAPTER

# 4

# Search for $Z^*$ and $W^*$ decay modes

## 4.1 Introduction

As discussed before, in the context of some models with extra-dimensions of size about 1 $TeV^{-1}$, in particular in the ADD model with only fermions confined to a D-brane [20], heavy KK excitations are expected, with the same properties as the SM gauge bosons, but more massive.

In the following, three hadronic decay modes of massive Z and W gauge bosons are investigated, using the ATLAS experiment at the LHC. These decay modes are the following:

$$\begin{aligned} Z^* &\longrightarrow b\,\bar{b} \\ Z^* &\longrightarrow t\,\bar{t} \\ W^* &\longrightarrow t\,b \end{aligned} \tag{4.1}$$

These decays are more difficult to detect than the corresponding leptonic decays, but may provide very useful information about $Z^*$ and $W^*$ couplings. In particular, if a resonance is found in any of the leptonic channels, the detection of these hadronic decays should allow a measurement of the couplings between heavy gauge bosons and quarks.

## 4.2 $b$-tagging

The identification of $b$ quarks is one of the most important tasks of the inner detector of the ATLAS experiment. For example, $b$ quark identification is required in the study of $t \rightarrow W\ b$ decays, or in the search for new particles strongly coupled to heavy quarks, like the Higgs boson [67].

Jets originated by $b$ quarks can be identified by taking into account the large lifetime and mass of $B$ hadrons. Tracks resulting from $B$ hadron decays have typically large impact parameters (minimal distance between the track and the primary event vertex) whereas tracks from decays of hadrons involving just $u$, $d$ or $s$ quarks originate in general from the primary vertex. It is also possible to study the semi-leptonic decays of the $b$ quark to identify $b$-jets, but this second method is less efficient than the first one.

The $b$-tagging algorithm involves the following two variables:

- The efficiency ($\epsilon$) to identify a jet generated by a $b$ quark.
- The rejection factor ($R$) for jets generated by a non $b$ quark. It is defined as the inverse of the efficiency, i.e., $R = 1/\epsilon$.

In general, it is enough to specify $R_u$ and $R_c$ because the rejection factor is approximately equal for all the other light quarks. $R_u$ also includes the contribution of jets generated by gluons fragmenting into light quarks. The rejection factor for a given value of $\epsilon_b$ depends mainly on two jet variables: the transverse momentum $p_T$ and the rapidity $\eta$.

In all decays studied in the following, the tagging of very high $p_T$ $b$ jets is required. The average $p_T$ of these jets is shown in table 4.1

| $< p_T >$ | $Z^* \rightarrow b\bar{b}$ | $Z^* \rightarrow t\bar{t}$ | $W^* \rightarrow tb$ |
|---|---|---|---|
| M = 1 TeV | 400 | 200 | 250 |
| M = 2 TeV | 800 | 400 | 500 |

Table 4.1: Average $p_T$ (in GeV) of final state $b$ jets for the various channels studied in this work.

The tagging of $b$ jets has been studied in detail within ATLAS (see for example [68]), but in all these studies the average $p_T$ of $b$-jets was typically below 200 GeV. In order to investigate the b-tagging performance for larger $p_T$ values, the following event samples have been generated and analysed

$$Z^*(2\,TeV) \rightarrow b\,\bar{b},\ c\,\bar{c},\ u\,\overline{u}$$

Each sample contained 20 000 events and was processed using the full reconstruction provided by ATLAS software.

The standard $b$-tagging algorithm was used. Since this algorithm is not optimised for high $p_T$ jets, the results should be considered conservative, and further improvements might be expected in the future.
The efficiencies ($\epsilon$) and rejections ($R = 1/\epsilon$) reported in table 4.2 have been applied in our analysis.

| mass | decay | $\epsilon_b$ | $R_c$ | $R_u$ |
|---|---|---|---|---|
| 1 TeV | $b\bar{b}$ | 0.1 | 140 | 1000 |
| | $t\bar{t}$ | 0.5 | 10 | 100 |
| | $tb$ | 0.5 | 7 | 40 |
| 2 TeV | $b\bar{b}$ | 0.1 | 45 | 90 |
| | $t\bar{t}$ | 0.2 | 28 | 130 |
| | $tb$ | 0.2 | 26 | 75 |

Table 4.2: Efficiencies and rejections applied in the $b$-tagging analysis.

## 4.3 Search for $Z^* \longrightarrow b\bar{b}$

### 4.3.1 Simulation

The events were generated using the ATLAS fast simulation and reconstruction MC program Atlfast [69] coupled to the Monte Carlo generator PYTHIA (version 6.157). In section 6.4 these two programs are described in more detail. As explained there, Atlfast was ported to the BOINC distributed computing platform and was used to generate events.

In table 4.3 some information about the generated events for a $Z^*$ mass of 2 TeV is presented.

| | **Signal** | **Reducible *Backg.*** | **Irreducible *Backg.*** |
|---|---|---|---|
| **Process** | $Z^* \to b\bar{b}$ | gg $\to$ jj | gg $\to b\bar{b}$ |
| **Cross section** | 0.9 pb | 5700 pb | 7.7 pb |
| **Generated events** | 10000 | 100000 | 100000 |

Table 4.3: List of generated events for the study of $Z^* \to b\bar{b}$ with $M(Z^*) = 2\ TeV$

Events were generated assuming an integrated luminosity of $\mathcal{L} = 3 \times 10^5\ pb^{-1}$, corresponding to 3 years of LHC running at high luminosity.

### 4.3.2 Selection cuts

The following selection cuts were applied.

- The $b$-type *jets* have $\mathrm{p}_T \geq 500\ GeV$ and $|\eta| \leq 2.5$.
- The invariant mass of $Z^*$ is $2000 \pm 400\ GeV$.

The *number of events* (N) were calculated using the following equation

$$N = \mathcal{L}\ \sigma\ BR(Z^* \longrightarrow b\bar{b})\ \epsilon \tag{4.2}$$

where

- $\mathcal{L}$ = integrated luminosity (we assume $3 \times 10^5\ pb^{-1}$)
- $\sigma(Z^*)$ = cross section (see table 4.3)
- $BR(Z^* \to b\bar{b})$ = decay branching ratio (in our case: $1/8 = 12.5\%$)
- $\epsilon$ = detection efficiency

$$\epsilon = \epsilon_{kin} \times \epsilon_b^2 \tag{4.3}$$

- $\epsilon_{kin}$ = number of events in the selected mass window divided by the total number of generated events
- $\epsilon_b$ = $b$-tagging efficiency (chosen to be 10%)

For the 10000 signal MC events, we obtain 5145 events passing the selection cuts so we obtain the following number of expected signal events according to equation 4.2

$$N_{sig} = 173 \tag{4.4}$$

The equation used to calculate the irreducible background is

$$N = \mathcal{L} \cdot \sigma(gg \to b\bar{b})\epsilon \tag{4.5}$$

where $\epsilon$ is calculated in the same way as the signal (eq. 4.3) since in both cases we have real $b$-type jets. The result for the irreducible background is 6946 of the 100000 simulated events passing the cuts, so the number of expected events in the data is

$$N_{irr} = 1604 \tag{4.6}$$

The equation used to calculate the reducible background is

$$N = \mathcal{L} \cdot \sigma(gg \to jj)\epsilon' \tag{4.7}$$

where $\epsilon'$ is calculated considering all the different jet types. There are 5 possible combinations: $bc$, $bu$, $cc$, $cu$ and $uu$. The details of the calculations can be found in ref. [70].

Finally we obtain $\epsilon = 2.2 \times 10^{-4}$ and, applying eq. 4.7, the following expected number of reducible events in real data

$$N_{red} = 19120 \tag{4.8}$$

### 4.3.3 Results

The following significance is obtained from the previous results

$$\frac{S}{\sqrt{B}} = \frac{N_{sig}}{\sqrt{N_{irr} + N_{red}}} = 1.2 \tag{4.9}$$

where $S$ is the signal and $B$ the total background.

The reconstructed mass peak for the simulated signal is shown in figure 4.1

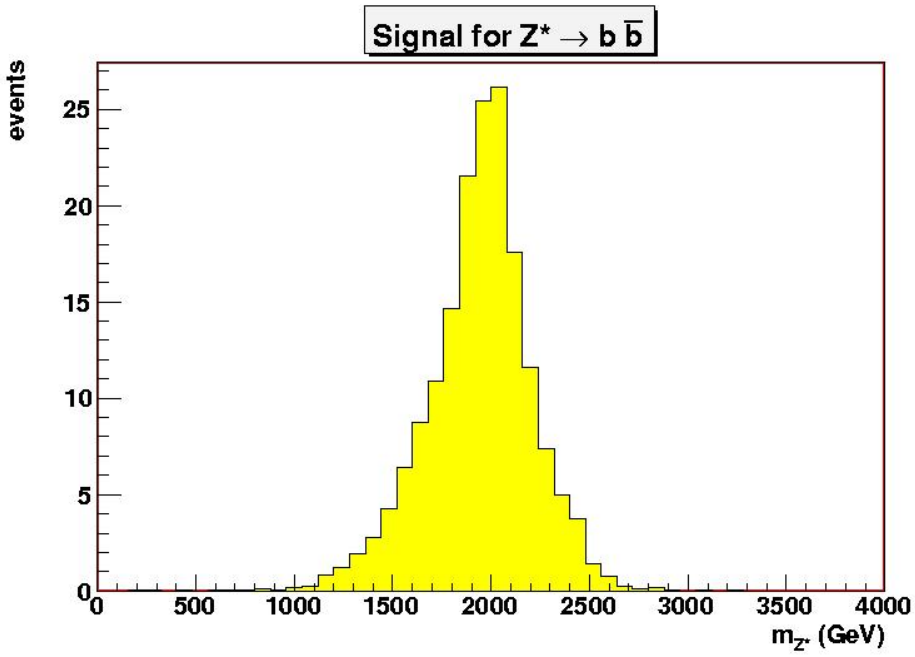


Figure 4.1: Simulated signal obtained in the study of $Z^* \to b\bar{b}$

In figure 4.2 the histogram including both signal and background events is presented as well.

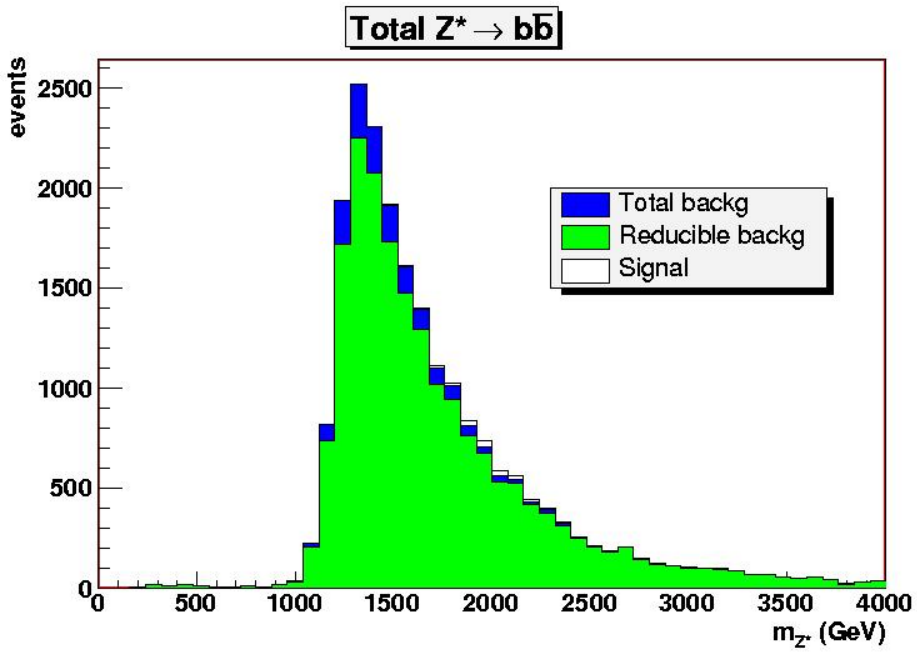


Figure 4.2: Signal and background obtained in the study of $Z^* \rightarrow b\bar{b}$

The signal is very small compared with the expected background. A signal is considered to be observable if the significance is larger than 5, but in our case it is much smaller. We conclude then from this analysis that, in general, the channel $Z^* \rightarrow b\bar{b}$ is not observable at the LHC, with an integrated luminosity of $3 \times 10^5\ pb^{-1}$ for a $Z^*$ mass of 2 TeV.

## 4.4 Search for $Z^* \longrightarrow t\bar{t}$

In this case the final state is more complex and therefore more difficult to identify. We consider only events with the following $t$ and $\bar{t}$ decays:

$$\begin{aligned} t &\rightarrow W + b \\ \bar{t} &\rightarrow \bar{W} + \bar{b} \end{aligned} \tag{4.10}$$

followed by the $W$ and $\bar{W}$ decays

$$\begin{aligned} W &\rightarrow jj \\ \bar{W} &\rightarrow l + \nu \end{aligned} \tag{4.11}$$

where $j$ is a jet and $(l, \nu)$ is a lepton with its associated neutrino. Therefore, the final state observed in the detector is

$$\begin{aligned} t &\rightarrow jj + b \\ \bar{t} &\rightarrow l + \nu + \bar{b} \end{aligned} \tag{4.12}$$

or, more precisely

$$Z^* \rightarrow l + \nu + \bar{b} + b + jj \tag{4.13}$$

### 4.4.1 Simulation

As in the previous study, the events were generated and reconstructed using Atlfast and the luminosity is assumed to be $\mathcal{L} = 3 \times 10^5\ pb^{-1}$. The details of the analysis are shown in table 4.4.

| | **Signal** | **Reducible *Backg.*** | **Irreducible *Backg.*** |
|---|---|---|---|
| **Process** | $Z^* \rightarrow t\bar{t}$ | $W$ + jets | $t\bar{t}$ events |
| **Cross section** | 0.9 pb | 6.35 pb | 1.77 pb |
| **Number of events** | 10000 | 100000 | 100000 |

Table 4.4: List of generated events obtained in the study of $Z^* \rightarrow t\bar{t}$ with $M(Z^*) = 2\ TeV$

.

### 4.4.2 Selection cuts

The following selection cuts were applied

- The muons have $p_T \geq 20\ GeV$ and $|\eta| \leq 2.5$.
- The electrons and the $b$-type *jets* have $p_T \geq 25\ GeV$ and $|\eta| \leq 2.5$.
- The invariant mass of the reconstructed $Z^*$ is $2000 \pm 400\ GeV$.

In total, 545 out of 10000 generated events pass the selection cuts. The cross sections are presented in table 4.4, the branching ratio of the decay mode is 1/8 as in the previous case and the efficiency of the detection 0.2. Then, using eq. 4.2 the following number of events is obtained

$$N_{sig} = 74 \tag{4.14}$$

Concerning the irreducible background 582 out of the 100000 generated events pass the selection cuts, and applying equation 4.5 we obtain

$$N_{irr} = 443 \tag{4.15}$$

In the case of the reducible background, we find

$$N_{red} = 2 \tag{4.16}$$

### 4.4.3 Results

Using the previous results the following significance is obtained

$$\frac{S}{\sqrt{B}} = \frac{N_{sig}}{\sqrt{N_{irr} + N_{red}}} = 3.5 \tag{4.17}$$

The reconstructed mass peak for the simulated signal is presented in figure 4.3

In figure 4.4 the histogram including signal and background events is presented as well.

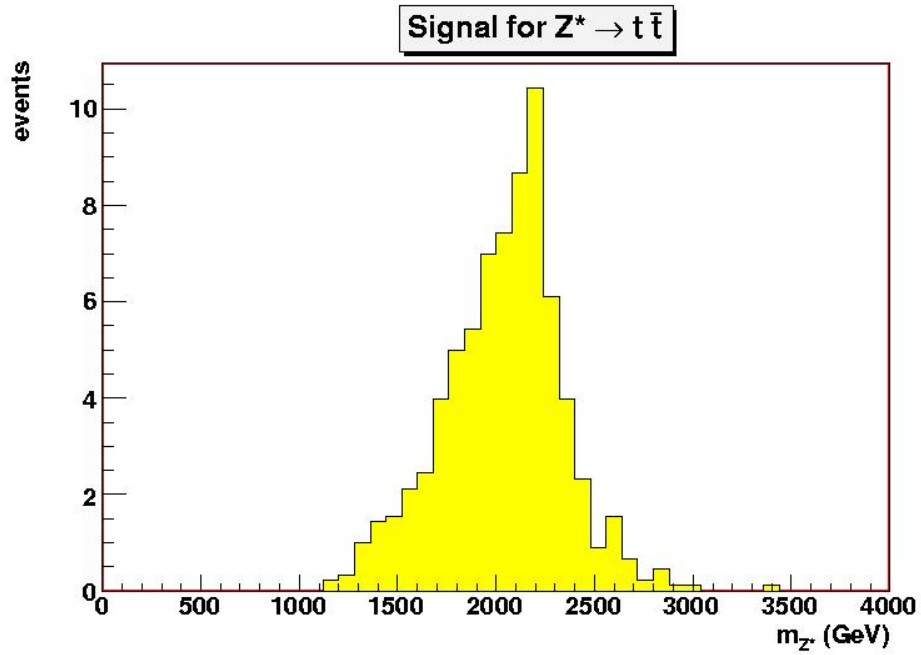


Figure 4.3: Simulated signal obtained in the study of $Z^* \rightarrow t\bar{t}$ with $M(Z^*) = 2\ TeV$

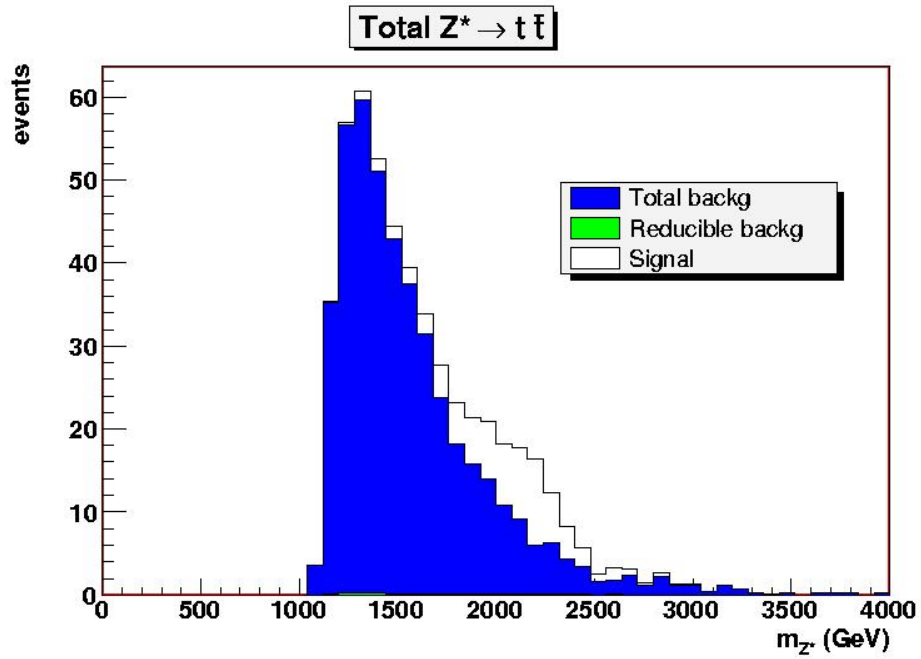


Figure 4.4: Signal and background obtained in the study of $Z^* \rightarrow t\bar{t}$ with $M(Z^*) = 2\ TeV$

In this case, the dominant background is irreducible ($t\ \bar{t}$) whereas in the previous decay mode ($Z^* \rightarrow b\bar{b}$) the dominant background was reducible.

As in the previous case, the signal is very small compared with the expected background and, although the significance is larger, the value is again smaller than 5. Then, we conclude from our analysis that, in general, the channel $Z^* \rightarrow b\bar{b}$ will be difficult to detect at the LHC with an integrated luminosity of $3 \times 10^5\ pb^{-1}$ for a $Z^*$ mass of 2 TeV.

## 4.5 Search for $W^* \longrightarrow t\ b$

As in the $Z^* \rightarrow t\bar{t}$ case, the final state, involving a large multiplicity of particles, cannot be identified easily. For this reason, we consider only the case where the $t$ quark decays leptonically.

### 4.5.1 Simulation

The following events (see table 4.5) were generated and reconstructed using Atlfast, assuming a integrated luminosity of $\mathcal{L} = 3 \times 10^5\ pb^{-1}$.

| | **Signal** | **Reducible *Backg.*** | **Irreducible *Backg.*** |
|---|---|---|---|
| **Process** | $Z^* \rightarrow t\ b$ | $W$ + jets | $t\ \bar{t}$ events |
| **Cross section** | 1.9 pb | 6.35 pb | 1.77 pb |
| **Generated events** | 10000 | 100000 | 100000 |

Table 4.5: List of generated events for the study of $W^* \longrightarrow t\ b$ with $M(Z^*) = 2\ TeV$

### 4.5.2 Selection cuts

In this case, the selection is applied in the same way as in the $Z^* \rightarrow t\bar{t}$ case, except that no energy is collected around the second $b$-quark, since in this case this $b$-quark is the only component of the second jet.

From the 10000 signal generated events, we have 396 passing the selection cuts. The cross sections are presented in table 4.5, the branching ratio of the decay mode is 1/4 and the efficiency of the detection is 0.4. Then, using eq. 4.2 the following number of events is obtained

$$N_{sig} = 226 \tag{4.18}$$

For the irreducible background we obtain 146 events, out of 100000 generated, passing the selection cuts so we obtain from equation 4.5

$$N_{irr} = 445 \tag{4.19}$$

For the reducible background, we obtain 1582 out of the 100000 generated events passing the selection cuts so the result is

$$N_{red} = 74 \tag{4.20}$$

### 4.5.3 Results

We obtain the following significance

$$\frac{S}{\sqrt{B}} = \frac{N_{sig}}{\sqrt{N_{irr} + N_{red}}} = 9.9 \tag{4.21}$$

The reconstructed mass peak obtained for the simulated signal is presented in figure 4.5

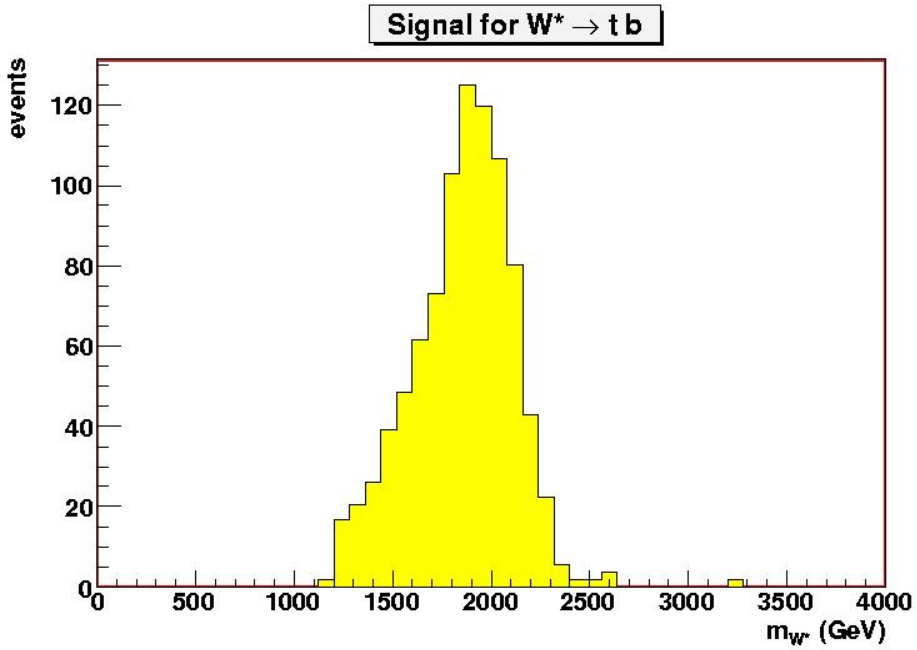


Figure 4.5: Simulated signal obtained in the study of $W^* \rightarrow t\ b$ with $M(Z^*) = 2\ TeV$

In figure 4.6 the histogram including signal and background events is presented

As in the previous case, the dominant background is irreducible.

However, in this case we find that the $W^* \rightarrow t\ b$ decay mode might yield a signal separable from the background. In addition, we also find a significance larger than 5. We conclude that it would be possible to detect this mode at the LHC, for an integrated luminosity of $3 \times 10^5\ pb^{-1}$ and a $Z^*$ mass of 2 TeV.

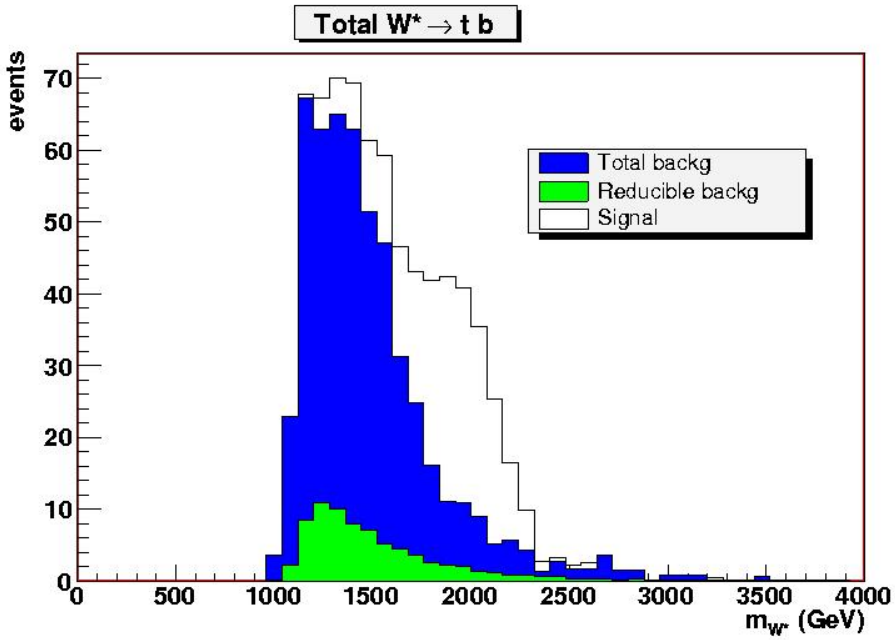


Figure 4.6: Signal and background obtained in the study of $W^* \rightarrow t\ b$ with $M(Z^*) = 2\ TeV$

## 4.6 Mass dependence

In table 4.6 we summarize the results from the previous analysis.

| M = 2 TeV | $Z^* \to b\bar{b}$ | $Z^* \to t\bar{t}$ | $W^* \to tb$ |
|---|---|---|---|
| signal | 174 | 74 | 226 |
| irreducible bkg. | 1605 | 443 | 445 |
| reducible bkg. | 19120 | 2 | 74 |
| significance | 1.2 | 3.5 | 9.9 |

Table 4.6: Signal and background inside a mass window of $\pm 400\ GeV$ around $M = 2\ TeV$ for $\mathcal{L} = 3 \times 10^5\ \ pb^{-1}$.

The analysis was also performed for masses of $M = 1\ TeV$. The results are presented in table 4.7

| M = 1 TeV | $Z^* \to b\bar{b}$ | $Z^* \to t\bar{t}$ | $W^* \to tb$ |
|---|---|---|---|
| signal | 3461 | 13170 | 34178 |
| irreducible bkg | 63764 | 116130 | 23231 |
| red. bkg | 24388 | 126 | 257 |
| significance | 11.7 | 35.2 | 364.5 |

Table 4.7: Signal and background inside a mass window of $\pm 200\ GeV$ around $M = 1\ TeV$ for $\mathcal{L} = 3 \times 10^5\ pb^{-1}$.

In figures 4.7, 4.8 and 4.9 the dependence of the significance on the mass is displayed for the three decays studied in this analysis. An exponential dependence on the mass is assumed.

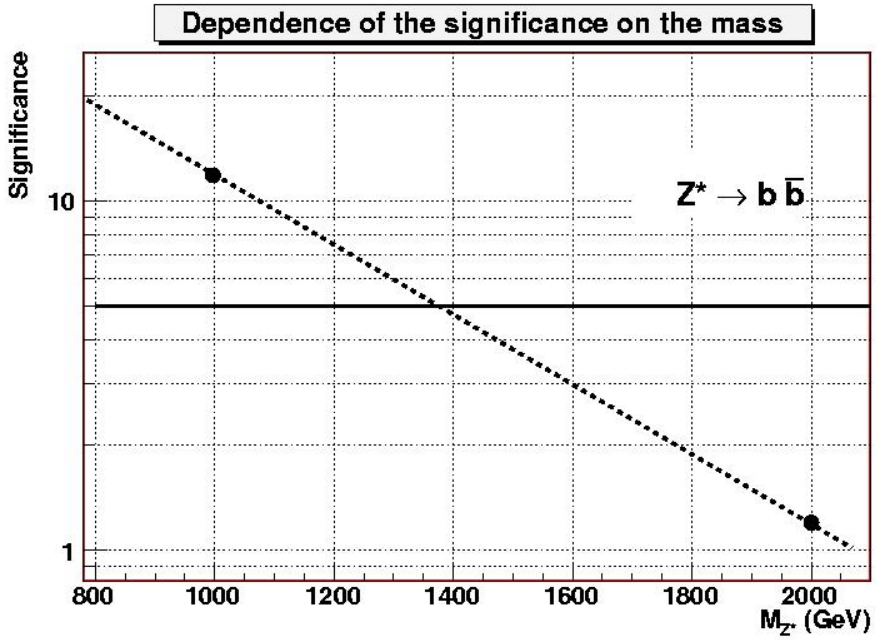


Figure 4.7: Significance as a function of mass for the study of the decay channel $Z^* \rightarrow b\bar{b}$ and assuming $\mathcal{L} = 3 \times 10^5\ pb^{-1}$

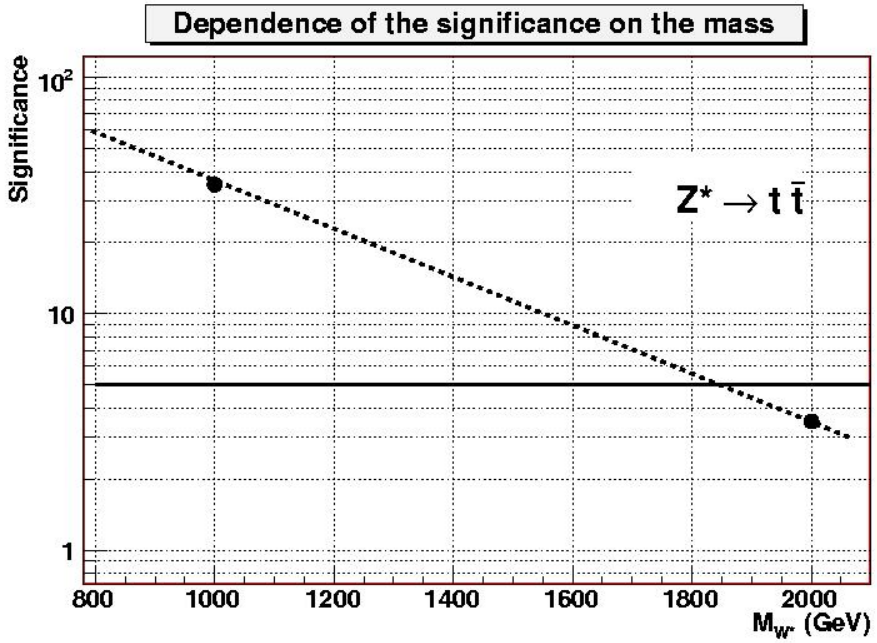


Figure 4.8: Significance as a function of mass for the study of the decay channel $Z^* \rightarrow t\bar{t}$ and assuming $\mathcal{L} = 3 \times 10^5\ pb^{-1}$

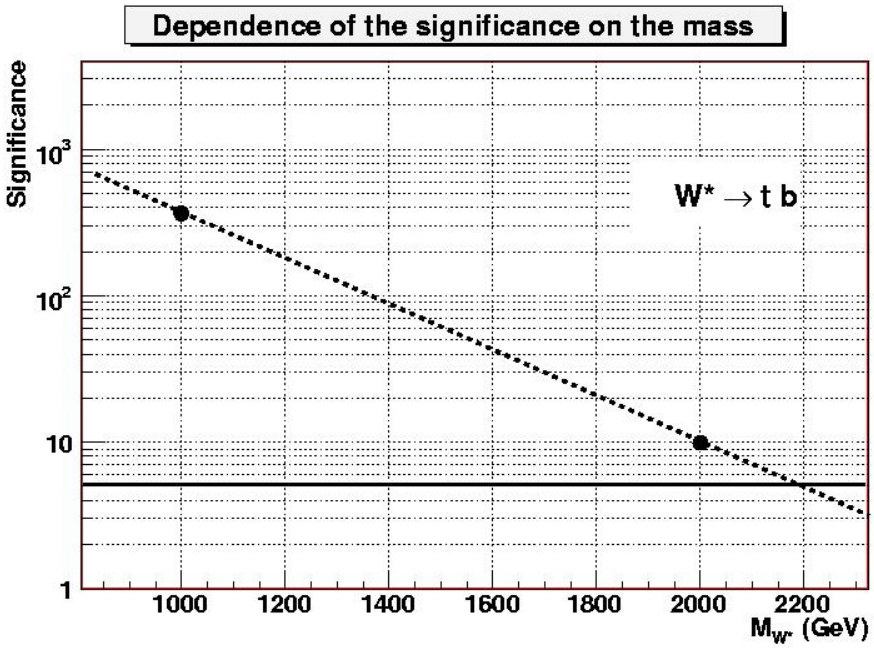


Figure 4.9: Significance as a function of mass for the study of the decay channel $W^* \rightarrow t\ b$ and assuming $\mathcal{L} = 3 \times 10^5\ pb^{-1}$

From the previous results we conclude that a significance of 5 may be achieved below a certain mass for each decay mode as shown in table 4.8.

| | $Z^* \to b\bar{b}$ | $Z^* \to t\bar{t}$ | $W^* \to tb$ |
|---|---|---|---|
| $M(TeV)$ | 1.4 | 1.9 | 2.2 |

Table 4.8: Maximum heavy boson mass in order to achieve a significance larger to 5 for an integrated luminosity of $\mathcal{L} = 3 \times 10^5\ pb^{-1}$.

# Part III

# Distributed Computing

CHAPTER

# 5

# The LCG project

In the past chapter we have studied the search of heavy gauge bosons in ATLAS at CERN. Now it is worth to talk about the computational tools that all CERN experiments will require to successfully process, distribute and analyse the obtained data.

The LCG Project Grid (see 3.2.1) is an important example of a well-known Grid environment in HEP and is the one used by the ATLAS experiment so it is studied in depth in this Chapter. In addition, we study the physics applications using this infrastructure like the required libraries, the data management tools and the simulation software. All of them are presented in general and also from the point of view of the ATLAS collaboration.

This description of the LCG Grid is important as in the following chapters we describe as well the BOINC Grid environment and we compare it with LCG.

We also describe the software tools used by LCG to deploy and manage the Grid software: Quattor. The work performed writing, improving and correcting Quattor components is also mentioned.

## 5.1 LCG technology and infrastructure

We describe the LCG middleware and services (see 3.2.1) deployed on the EGEE infrastructure (see 3.2.2), as it is the one used in Europe (except in

the Nordic countries where NorduGrid is used), in particular in Spain. We focus on the case of the ATLAS collaboration.

EGEE began to work using the LCG-2 middleware package, provided by the LCG project. This middleware was based on the one from the EDG project (see 3.2.3), EGEE's predecessor, among other ones like Globus (see 3.1.4) and Condor [64]).

In parallel it produced the Lightweight Middleware for Grid Computing (**gLite**) [71], re-engineered using components from a number of sources to produce lightweight middleware that provides a full range of basic Grid services. As of September 2006, gLite is at version 3.0, and comprises some 220 packages arranged in 34 logical deployment modules. The gLite middleware is also used by a number of groups outside of EGEE.

As gLite becomes available it is certified by the deployment team, and installed first on the pre-production service, where it is further tested by applications groups before being more widely distributed. As far as possible the new components should be able to co-exist on the same Grid system with the current middleware, enabling a progressive deployment strategy to be adopted. This is an essential feature for new middleware to be introduced into a large operational Grid.

The current middleware package is LCG-2. It is planned to be phased out as soon as possible after the gLite package has been adopted by the major applications.*.

This Grid middleware (LCG-2 / gLite at EGEE) can be classified into the following different services:

- Security
- Information Service
- Data management
- Workload management
- Computing & storage elements
- File Transfer Service

They are based on the ones used by the EDG project (see 3.1.3). Below we describe in depth each of them.

*At the time of finishing this work, end of 2006, gLitle has already been adopted by most of the LCG sites. In particular all Tier 1 centres should have gLite installed in production from June 2006 [44]. Nevertheless, it is also stated there that the current official middleware package continues being LCG-2.

### 5.1.1 Security

All EGEE middleware services rely on **GSI** (see 3.1.4). To be able to use the Grid facility, users must get and renew their (long-term) **certificates** from an accredited **Certification Authority (CA)**. Short-term proxies are then created and used throughout the system for authentication (check of the user's identity) and authorisation (check of user's rights to access to the Grid resources). These short-term proxies may be annotated with VO membership (see 3.1.2) and group information obtained from the **Virtual Organisation Membership Services (VOMS)**. When longer-term proxies are needed, MyProxy services can be used to renew the proxy. The sites maintain **Certificate Revocation Lists (CRLs)** to invalidate unauthorised usage for a revoked Grid user.

There is no significant functional difference between the VOMS in LCG-2 and in gLite.

In the ATLAS collaboration the VOMS middleware package allows the definition of user groups and roles within the ATLAS VO. There is initially a VOMS group for each Physics Working Group, Combined Performance group and Detector System. There is also a VOMS generic group and another one for software testing, validation and central production activities.

### 5.1.2 Information service

Information services publish and maintain data about resources in Grids. This information in LCG is modelled by the **Grid Laboratory Uniform Environment (GLUE) schema**. The aim of the GLUE schema is to define, publish and enable the use of common schemas for interoperability between the EU physics Grid project efforts (LCG, EDG...) and the US physics Grid project efforts (iVDGL, PPDG, GriPhyN...). The GLUE schema have the following main components:

- The **Berkeley Database Information Index (BDII)** is an implementation of the GIIS (see 3.1.4), but allowing for more scalability. Information provided by the BDII adheres to the GLUE information model. Interfacing with BDII is made of Lightweight Directory Access Protocol (LDAP) operations for which commands and an application programming interface (API)[†] exist. Both LCG-2 and gLite currently rely on BDII for proper operation.

- The **Relational Grid Monitoring Architecture (R-GMA)** [72] presents a relational view of the collected data. It is basically a producer/consumer service with command line interfaces as well as an API

[†]An API is a source code interface that a computer system or program library provides in order to support requests for services by a computer program.

for Java, C, C++ and Python and a Web interface. R-GMA models the information infrastructure of a Grid as a set of consumers (that request information), producers (that provide information) and a central registry which mediates the communication between producers and consumers. R-GMA can use the same information providers as used by BDII. Recently, a Service Discovery mechanism using R-GMA has been implemented (see references at [30]). R-GMA is currently also used to collect LCG accounting records.

- The **Logging & Book-keeping services (LB)** tracks jobs during their lifetime in term of events (important points of job life, such as submission, starting execution, etc.). It takes the information from the the Workload Managers (WMs) and the CEs (they are instrumented with LB calls). The events are first passed to a local logger then to book-keeping servers.

- **Job Provenance Services** keep track of submitted jobs (completed or failed), including execution conditions and environment, and important points of the job life-cycle for longs periods (months to years). Currently they are being prototyped. Then, this information can be reprocessed for debugging, post-failure analysis, comparison of job execution and re-execution of jobs.

### 5.1.3 Workload management System (WMS)

The WMS is responsible for the distribution and management of tasks across Grid resources, in such a way that applications are conveniently, efficiently and effectively executed. It essentially provides the facilities to manage jobs (submit, cancel, suspend/resume, signal) and to enquire about their status.

In LCG-2, it makes use of Condor and Globus technologies and relies on GSI security. It dispatches jobs to appropriate CEs, depending on job requirements and available resources. BDII and the Replica Location Service (RLS) are used for retrieving information about the resources. The user interfaces to the WMS using a **Job Description Language (JDL)** based on the Condor one.

The gLite's WMS is based on the LCG-2 one and it is interoperable with LCG-2 CEs. The main difference is that in gLite CEs can ask the WMS for work. Then, LCG-2 works using a push model (where a job is pushed to a CE for its execution) while gLite can use also a pull model (where a CE asks a Workload Manager for jobs).

Then, gLite's WMS includes a **Workload Manager (WM)** or Resource Broker (see 3.1.3) which is responsible of accepting and satisfying job management requests coming from its clients. The WM will pass job submission requests to appropriate CEs for execution, taking into account requirements

and preferences on the resources expressed in the job description but also on the policies that the sites or the VO administrators have put on the CEs.

Other new features in gLite are the addition of a Web service interface to the WMS as well as bulk submission and parametrised job capabilities .

### 5.1.4 Storage Element (SE)

An SE is a logical entity that provides storage space to the Grid. In general it will be a Mass Storage System (MSS), either disk cache or disk cache front-end backed by a tape system.

Each Grid site may provide multiple SEs providing different qualities of storage. For example, it may be considered convenient to provide an SE for data intended to remain for extended periods and a separate SE for data that is needed only for the lifetime of a job (or set of jobs).

In LCG-2, SEs support GridFTP (see 3.1.4) and SRM interfaces[‡] to transfer data. gLite itself does not provide a particular SRM interface nor a GridFTP server. The main difference with LCG-2 is that gLite has a POSIX-like I/O service for access to Grid files via their Logical File Names (LFNs). It provides open/read/write/close style of calls to access files while interfacing to a file catalogue. gLite I/O currently interfaces to the File and Replica Manager (FiReMan) and the LCG-RLS catalogues.

Regarding authentication, authorisation and audit/accounting features, the SE should provide and respect ACLs for files and datasets that it owns, with access control based on the use of proxy certificates with a user Distinguish Name (DN) and attributes based on VOMS roles and groups. It is essential that a SE provide sufficient information to allow tracing of all activities for an agreed period. It should also provide information and statistics on the use of the SE.

### 5.1.5 Compute Resource Services

A **Computing Element (CE)** is a set of services that provide access to a local batch system running on a computing farm. Typically the CE provides access to a set of job queues within the batch system.

In LCG-2, GRAM (see 3.1.4) is used for submitting jobs to the LRMS[§]. As we said in the WMS description, the main difference between LCG-2 CEs and gLite ones is that the second ones can ask a WM for jobs.

In LCG-2 as well as in gLite, the CE interfaces to the LB services to keep track of the jobs during their lifetimes. Also in both cases, it uses GSI

[‡]The Storage Resource Manager (SRM) defines a set of functions and services that a storage system provides and allows the Grid to access the SE.

[§]A Local Resource Management System (LRMS) is a system to distribute and balance the load of the CPU resources. LCG CEs support, among other ones, Load Sharing Facility (LSF), Portable Batch System (PBS) and Condor

security, the Globus gatekeeper and Grid Information System according to the GLUE schema. The authentication and authorisation mechanisms at the CEs are based on the VOMS model.

### 5.1.6 Data Management

Files on Grids can be replicated in many places. The users or applications do not need to know where the files actually are, and use **Logical File Names (LFNs)** to refer to them. In order to ensure that a file is uniquely identified, **Global Unique IDentifiers (GUIDs)** are usually used.

It is the responsibility of file catalogues services to locate and access the data. The most important ones are:

- **EDG Replica Management Service (RMS)**: The services provided by the RMS, originating from EDG, are the Replica Location Service (RLS) and the Replica Metadata Catalogue (RMC). The RLS maintains information about the physical location of the replicas. The RMC stores mappings between GUIDs and LFNs. A last component is the Replica Manager offering a single interface to users, applications or Resource Brokers. Currently, EDG RMS are gradually be phasing out.

- **LCG File Catalogue (LFC)**: It offers a hierarchical view of logical file name space. The two functions of the catalogue are to provide LFN and to locate the site at which a given file resides. The LFC provides Unix style permissions and POSIX Access Control Lists (ACL). The catalogue exposes a **Data Location Interface (DLI)** that can be used by applications and Resource Brokers. Simple metadata can be associated with file entries. The LFC provides a command line interface and can be interfaced through Python.

- **gLite File and Replica Manager (FiReMan) Catalogue**: Based on LFC. The catalogue provides Unix-style permissions and ACLs support via Distinguished Names or VOMS roles. File access is secured via these ACLs. The Fireman catalogue also provides Web Services Interfaces. The catalogue exposes to so-called Storage Index interface used by the gLite Workload Management System to dispatch jobs at the relevant site.

### 5.1.7 File Transfer Service (FTS)

Basic-level data transfer between the SE and the Grid is provided by GridFTP. This is the essential mechanism by which data is imported to and exported from the SE. Normally the GridFTP transfer will be invoked indirectly via FTS or through SRM.

LCG-2 does not provide a particular FTS (it was up to the user) while gLite has one called **File Placement Service (FPS)**. It receives data movement requests and executes them according to defined policies. It maintain a persistent transfer queue thus providing reliable data transfer and interacts with the FiReMan catalogue. The FPS can be used without the interaction with the catalogue and is then referred to as the gLite FTS.

## 5.2 LCG Activity Areas

The LCG project is classified into the following four activity areas:

- Fabric Area
- Grid Deployment Area
- Distributed Analysis
- Applications Area

In the following we describe briefly their aims and related projects. More information as well as details and references can be found at [44] [30] [105].

### 5.2.1 Fabric Area

This activity area is responsible of organising the LHC computing services at CERN and the architecture of the Tier 0 and the CERN Analysis Facility (CAF) installations. It also has to verify and manage the scalability and performance of the architecture in the Data Challenges (DCs) as well as the sharing of technical information with systems administration experts at other regional centres. It is also meant to re-evaluate regularly the evolving technologies in areas such as fabric management, storage and computation and to perform the regular re-assessment of the cost of the LHC computing facilies.

### 5.2.2 Grid Deployment Area

It organises and operates the global Grid service for LHC and coordinates with regional centre managers and with the production managers from the experiments. It also has to build and certify the distribution packages for installation at regional centres and to operate the Grid infrastructure (information services, registration services, call centre, operations centre, user consultancy and support).

### 5.2.3 Distributed Analysis (ARDA)

The ARDA area, which stands for A Realisation of Distributed Analysis for LHC, is responsible for working with teams within the experiments that are involved in prototyping distributed analysis systems, helping them to interface to Grid services and coordinating between the experiments, the middleware developers and regional centres.

### 5.2.4 Applications Area

The aim of this area is to provide the tools and infrastructure for physics application software development as well as to organise sub-projects to implement the common solutions identified by the different testbeds and achieved challenges.

In particular, the work of the applications area is conducted within projects. There are currently four projects. Their official web sites as well as the details, references and documentation can be found at [105].

- Software process and infrastructure (SPI)
- Core libraries and services (ROOT)
- Persistency framework (POOL)
- Simulation (SIMU)

#### Software Process and Infrastructure (SPI)

The software projects of the LCG Applications Area share a single development infrastructure, which is provided by the SPI project. A set of basic services and support are provided for the various activities of software development. The definition of a single project managing the infrastructure for all the development projects is crucial in order to foster homogeneity and avoid duplications in the way the Applications Area projects develop and manage their software.

The goal of the SPI project is to provide to the development projects of the LCG the following:

- basic environment for physics software development
- general scientific libraries and class libraries
- software development tools
- documentation tools and document templates

- compiler expertise

- support activity necessary to ensure that a common Grid-enabled environment is available at all Grid sites

The aim of SPI is to achieve those roles while taking care of consistency and homogeneity in the development of the different packages of the LCG Application Area.

In addition, configuration management support is provided for all LCG projects for both Configuration Management Tool (CMT) [87] and Software Configuration, Release And Management (SCRAM) [106] configurations such that LCG software can be used in the various build environments of the experiments. LCG software is distributed using Web-downloadable tarfiles of all binaries. Recently, Pacman [88] repositories of both sources and binaries are being provided by SPI.

In our work, the CMT, SCRAM and Pacman tools were needed to install and/or configure some ATLAS software, like Atlfast.

Another part of SPI is the External Software Service [104], which provides open source and public domain packages required by the LCG projects and experiments. Presently, more than 50 libraries and tools are provided on the set of LCG-supported platforms. All packages are installed following a standard procedure and are documented on the web and a set of scripts has been developed to automate new installations.

As explained in appendix B, most of this External Software was ported by us to Solaris because it was required by the SEAL.

### Core libraries and services (SEAL-ROOT)

The aim of the SEAL project (Shared Environment for Applications at LHC) is to provide the software infrastructure, basic frameworks, libraries and tools that are common among the LHC experiments. The project should address the selection, integration, development and support of foundation and utility class libraries. These utilities cover a broad range of unrelated functionalities and it is essentially impossible to find a unique optimum provider for all of them. They should be developed or adapted as the need arises. In addition to these foundation and utility libraries, the project should develop a coherent set of basic framework services to facilitate the integration of LCG and non-LCG software to build coherent applications.

Recently, the SEAL project has become just a work package of the ROOT project.

ROOT[¶] [116] is an object-oriented analysis framework aimed at solving the data analysis challenges of high-energy physics.

It was originally programmed by the same team of researchers who lead successful projects such as PAW, PIAF, and Geant. They knew that FORTRAN libraries had reached their limits as they cannot scale up to the challenges offered by the Large Hadron Collider, where the data is a few orders of magnitude larger than in LEP, to compare with the most recent experiment of similar size.

ROOT was developed in the context of the NA49 experiment at CERN. NA49 has generated an impressive amount of data, around 10 Terabytes per run. This rate provided the ideal environment to develop and test the next generation data analysis with ROOT.

A particular version of SEAL as well as ROOT were ported by us to the Solaris operating system to address the interest of LHC researchers to have the analysis environment working on this platform and to assess the difficulty of porting this important code, required by most of the ATLAS software. The details can be found in appendix B .

### Persistency framework (POOL)

The POOL project (acronym for POOL Of persistent Objects for LHC) has been created to implement a common persistency framework for LCG. POOL can store multi-Petabyte experiment data and metadata in a distributed and Grid enabled way. The project follows a hybrid approach combining C++ Object streaming technology, such as ROOT I/O, for the bulk data with a transaction safe relational database store, such as MySQL. POOL is based a strict component approach providing navigational access to distributed data without exposing details of the particular storage technology.

### Simulation (SIMU)

The simulation project of the LCG Applications Area encompasses common work among the LHC experiments on the development of a simulation framework and infrastructure for physics validation studies, CERN and LHC participation in Monte Carlo generator services, Geant4, Fluka and Garfield. Its work is guided by the reports of the simulation, the Monte Carlo generators and the detector description.

In our work, the Garfield simulation package was studied in the context of a Grid environment as a useful FORTRAN simulation software test case. It is studied in depth in section 6.4.

[¶]ROOT is not an acronym but a name, although it contains the acronym OO of Object Oriented. The name has been proposed to show the idea of making a system that provides a solid ROOT on which other systems can grow.

## 5.3 Quattor and farm management

At CERN, we performed several tasks related with Quattor:

- we set up a working Quattor environment under Solaris. This required the porting, correction and creation of several Quattor components and packages.
- we corrected and improved functionalities of the Quattor automated system to install, uninstall and configure software.
- we used all this experience to install and manage a farm of computers that we used to execute Grid jobs.

The two first tasks regarding the porting, correction and creation of Quattor components and also of the Quattor automated software management system are presented in detail in appendix A.

The installation and management of a farm of computers at CERN and its use in running Grid jobs and as part of the BOINC testbed is presented in section 6.5, as it is part of the performed work regarding BOINC.

CHAPTER

# 6

# BOINC

After talking about the LHC Computing Grid project, the related ATLAS physics applications and its software management tools, we present the BOINC distributed computing environment as a possible complement to LCG.

After describing its general infrastructure we show the work performed on porting physics applications of interest for ATLAS, like the PYTHIA event generator, the Atlfast reconstruction and simulation software and the Garfield gas simulation software.

The deployment of more than 200 CPUs at CERN to work on those applications using the BOINC environment is also presented.

In addition, we discuss as well about the testbed performed regarding the framework protocol of collaboration signed by CERN and the regional government of Extremadura region at Spain.

## 6.1 Middleware components

The main components of the middleware used by BOINC Grids are similar to the components used by the LCG one (see section 3.1.3) although there are some small differences due to the different approaches of both Grid flavours.

In particular, in BOINC, these middleware components are distributed in mainly two kind of machines: server and clients. They are organised in the way we show at figure 6.1 .

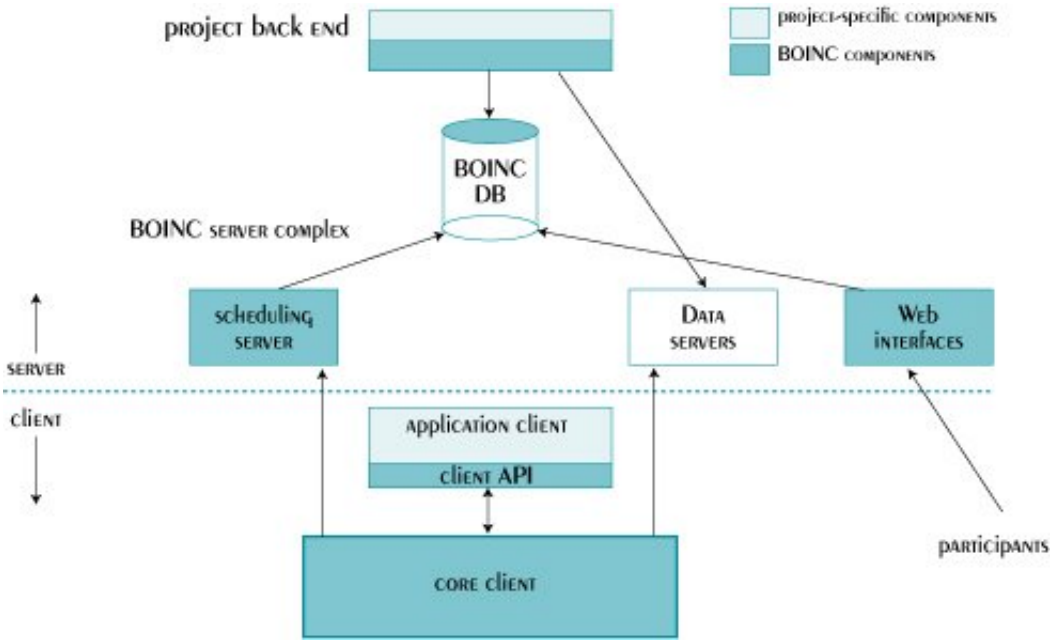


Figure 6.1: BOINC client-server infrastructure showing the different parts of each kind of components.

- **BOINC server**: Involves all Grid services except the computation itself.
  - **User Interface (UI)**: allows users to access to the Grid facility and receives the input Grid jobs. In the BOINC, the job information is provided by two different input files written in Extensible Markup Language (XML). One of them gives details about how to perform the job and the other one informs about how to retrieve the results. In LCG, a single input file written in JDL is used.
  - **Web interfaces**: In addition to the UI that we found in most Grids, in BOINC Grids there is the need also of a **Participants Web Interface**. It allows the participants who donate their computer power to register their computers and to follow up their work and update preferences like the percentage of memory and CPU from their computers that they wish to donate. BOINC provides also a **Management Web Interface** which allows the Grid managers to monitor, manage and debug the Grid jobs and their results, the registered participants and hosts, the BOINC applications. It also allows the Grid managers to perform other operations like the management of the participants forum.
  - **BOINC Database (DB)**: it is a MySQL database [73]. It stores all information regarding sent jobs, received results and registered client machines, participants and applications to run. It can be

accessed by the participants using the Participants Web Interface and by the Grid users using the Management Web Interface. It can also be accessed and managed by the BOINC administrators using other MySQL tools. For instance, LHC@home managers use a web tool called phpMyAdmin [74]. We can comment that LCG can use MySQL but also Oracle databases.

  - **Data servers**: provide storage space so they are equivalent to the Storage Elements we find in other kinds of Grids. In addition, they also act as servers for the clients to send and retrieve input data and results.

  - **Scheduling Server**: sends jobs to the BOINC clients and receives their results. It performs the roles of the Resource Brokers in LCG Grids. It does not need to contact any Information Service as in BOINC are the clients or Worker Nodes which ask for work to the Scheduling Server when they foresee that the can run out of work.

- **BOINC clients**: Performs the computation of the jobs. It involves what is called Worker Nodes and Computer Elements in other kinds of Grids. In this way, each one of the BOINC clients is itself a single Worker Node and Computer Element.

  There have been some discussions and development regarding the possibility of allowing BOINC clients to work as Storage Elements as well as a complement and replica copies of the BOINC Data Servers. Currently this feature has not been implemented.

## 6.2 BOINC technology and infrastructure

A BOINC server includes a set of daemons for generating and handling work. These daemons are not present in the BOINC clients. It is important to notice that all BOINC code is open source so it is available to be modified and adapted to particular needs and some of those daemons have been adapted for different projects. In particular some of them were modified to match our needs regarding the applications ported by us, as it is described in section 6.4 .

They involved daemons are shown in figure 6.2 and are the following ones:

- Work generator
  There is one work generator per application. It creates BOINC Grid jobs (called 'workunits' or WUs in BOINC) and the corresponding input files. It is application-specific, and uses BOINC library functions for registering the workunits in the database.

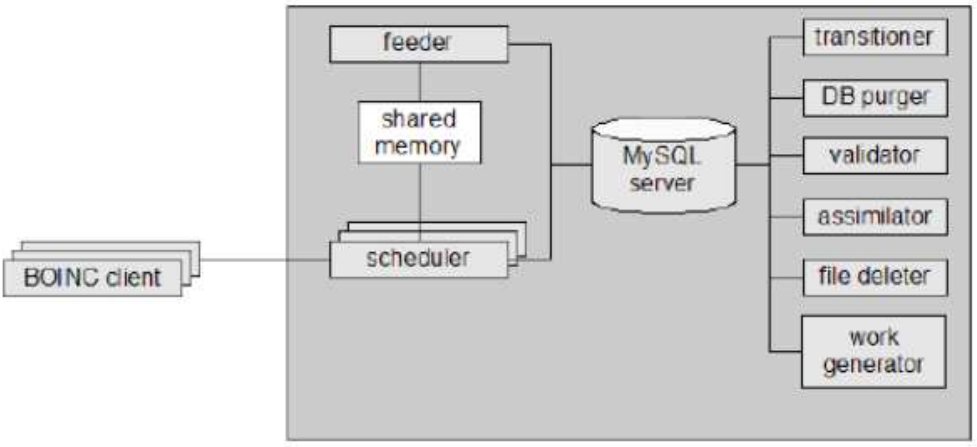


Figure 6.2: BOINC daemons

- Feeder
  It is supplied by BOINC and is application independent. It creates a shared-memory segment used to pass database records to the BOINC scheduler processes. This data includes applications, application versions, and 'work items' (an unsent result and its corresponding workunit).

- Transitioner
  This daemon is supplied by BOINC and is application independent. It handles state transitions of workunits and results. It generates initial results for workunits, and generates more results when timeouts or errors occur.

- Validator
  There is one validator per BOINC Grid application used to 'validate' the results returned from the BOINC clients. It compares redundant results and selects a 'canonical' result, see section 3.3.2, representing the correct output, and a 'canonical' credit granted to participants and hosts that return the correct output. Depending on your application, you can use the BOINC-supplied validator, or you may have to develop a customised validator. The default BOINC-supplied validator just checks if the files, containing the results, returned by different clients for the same workunit are exactly the same ones. We have to remember that this is important as the workunits are sent typically to volunteer client machines that in some cases may give wrong computation results so this validator checks if the results of same jobs sent to different clients agree. Then, this default validator does not analyse

the information obtained from the calculations, although a customised one can be implemented by the BOINC managers of the project.

- Assimilator
  There is one assimilator per application. It handles workunits that are 'completed', i.e. that have a 'canonical' result, or for which an error condition has occurred. Handling a successfully completed result might involve record results in a database and perhaps generating more work.

- File deletion
  This application-independent daemon deletes input and output files when they are no longer needed. This is made in a completely automated way.

- Database purging
  This daemon is also independent from the BOINC application. It removes work-related database entries when they are no longer needed. This keeps the BOINC database at a constant size even when your project runs for a long time.

### 6.2.1 Generating work

As described earlier, a BOINC Grid job or workunit involves the inputs to a computation. The steps followed to create a workunit in BOINC are:

- To write XML 'template files' that describe the workunit and its corresponding results. In short, they just contain the names of the input and output files, the application to be executed and the required options. Generally the same templates will be used for a large number of workunits as in BOINC the applications to be executed typically do not change and are the input files provided with each job which are different.

- To create the workunit's input file(s) and place them in the download directory.

- To invoke the BOINC function, or a customised script, that creates a database record for the workunit and sends the job to the Worker Nodes.

Once this is done, BOINC takes over: it creates one or more results for the workunit and distributes them to client hosts. These clients execute the jobs and return the results back to the server. It collects the output files, finds a canonical result, assimilates the canonical result, and deletes files.

During the testing phase of a project, you can use the make_work daemon to replicate a given workunit as needed to maintain a constant supply of work. This is useful while testing and debugging the application.

### 6.2.2 Validation

In BOINC, the process of validation performs the following two tasks:

- It compares the redundant results and decides which ones are to be considered correct. By default, the different results for the same job, executed on different Worker Nodes, are just compared to check if they are the same ones. This can be customised to perform more tests to validate the results. In any case, the performed tests are completely done in an automated way so they just check the coherence and consistency of them and they are not analised in depth.

- It decides how many credits to grant to each correct result. The default behaviour is explained in section 3.3.2 . Is important to notice that it can be customised. In particular, the BOINC project Climate Prediction made major changes to the default schema and does not grant credits for given results but for computation time, without taking into account if the job finish or not, as their jobs typically take too long time compared with the ones of the other projects.

A validator is a daemon program. You must supply a validator for each application in your project, and include it in the "daemons" section of your project configuration file.

Depending on various factors, you may be able to use a standard validator that comes with BOINC (it provides two different validator frameworks), or you may have to develop a customised validator.

- If your application generates exactly matching results then you can use what in BOINC is called 'sample bitwise validator', which requires a strict majority and regards results as equivalent only if they agree byte by byte. An application typically generates exactly matching results either because it does no floating-point arithmetic, or because you use the BOINC 'homogeneous redundancy' feature which configures the BOINC server to send results for a given workunit only to hosts with the same operation system name and CPU vendor.

- If you are using BOINC for 'desktop Grid' computing (i.e. you trust all the participating hosts) then you can use the 'sample trivial validator', which regards any two results as equivalent if their CPU time exceeds a given minimum.

- Otherwise, you will need to develop a customised validator for your application. BOINC supplies a simple validator framework in which you plug in three short application-specific functions. This is sufficient for most projects. If you need more control over the validation process, you can use BOINC's advanced validator framework.

You can find more details about the available validators and frameworks that come with the BOINC environment at [33].

### 6.2.3 Result assimilation

Projects must create one assimilator program per application. This is called when one of the following conditions is fullfiled:

- The workunit finish with error status. In this case the assimilator can write a message to a log or send an email to the application developer.
- The workunit has a canonical result. In this case it might, for example, parse the canonical result's output file and write its contents to a separate database.

If the assimilator finish successfully , the workunit record will be marked as assimilated. Otherwise the assimilator will log an error message and exit. In this way the system administrator can fix the problem before any completed or erroneous workunits are mis-handled by BOINC.

Is also important to notice that there exist some BOINC's back-end utility functions to get file pathnames and open files to manage the successful obtained results or to debug the ones returned with error.

### 6.2.4 Server-side file deletion

Files are deleted from the data server's upload and download directories by the file_deleter daemon. Typically you do not need to customise this [33].

The default file deletion policy is:

- A workunit's input files are deleted when all results are 'over' (reported or timed out) and the workunit is assimilated.
- A result's output files are deleted after the workunit is assimilated. The canonical result is handled differently, since its output files may be needed to validate results that are reported after assimilation; hence its files are deleted only when all results are over, and all successful results have been validated.

### 6.2.5 Database purging

As a BOINC project operates, the size of its workunit and result tables increases. Eventually they become so large that adding a field or building an index may take hours or days. To address this problem, BOINC provides a utility db_purge that writes result and WU records to XML-format archive files, then deletes them from the database. Workunits are purged only when

their input files have been deleted. Because of BOINC's file-deletion policy, this implies that all results are completed. So when a workunit is purged, all its results are purged too.

## 6.3 LHC@home

LHC@home [75] [76] [77] is the official BOINC CERN project to which volunteers can attach their computers to donate their idle CPU to execute jobs attached to CERN researches.

Currently, the LHC@home CERN BOINC project runs the SixTrack simulation program [78]. It simulates protons circulating around the Large Hadron Collider (LHC) ring in order to study the long-term stability of the particle orbits. Each job typically tracks 105 turns of 60 particles in the LHC and requires about 110 hours of CPU time on a modern PC.

Is also worth to mention that the SixTrack program is part of the software used in the CPU benchmarks test performed by the version 2000 of the Standard Performance Evaluation Corporation Organisation (SPEC CPU2000), used to assess the relative performance of a given CPU in units of sfp2K CPU, i.e. in SPEC CPU2000 flotant point CPU units.

The application is written in FORTRAN. It was the first one written on that language ported officially to the BOINC platform and at that time the BOINC libraries, written in C, were not able to be used on FORTRAN applications without some kind of wrapper. Then, a C wrapper was developed at CERN for LHC@home and successfully used in the porting of SixTrack. After that, the code was fed back to the BOINC project team and currently it is used by other BOINC applications.

The SixTrack application was ported to BOINC on August 2004 by a team involving researchers from Copenhagen University, the Helsinki Institute of Physics, the University of Basel, U.C. Berkeley and CERN's IT Department [79].

In April 2006, the system was about 14000 active volunteer participants *, 25000 active hosts and could provide more than five sustained Tera-FLOPS processing rate [77]. This has allowed the computation of more than 700 CPU-years in few months, assuming as typical a 1 Ksfp2K CPU that is approximately equivalent to a 2.8 GHz Xeon.

Figure 6.3 shows the number of participants and BOINC credits at that time. We can also see at this figure how the installation of the BOINC farm

*Remember that, as explained in section 3.3, in BOINC-related papers a 'participant' is not a Grid user but a person who donates computing resources by installing the BOINC client software in their computers

installed with Quattor at CERN at the summer of 2005 raised up highly the amount of performed work. This will be presented at section 6.5 .

So, even if not totally free, the cost/performance ratio is very positive. This is without taking into account the outreach effects, like the publicity obtained when this huge amount of people is able to participate in some way in the project.

We note that LHC@home still has the potential of becoming much larger if enough work is available to use the computing capacity that is offered.

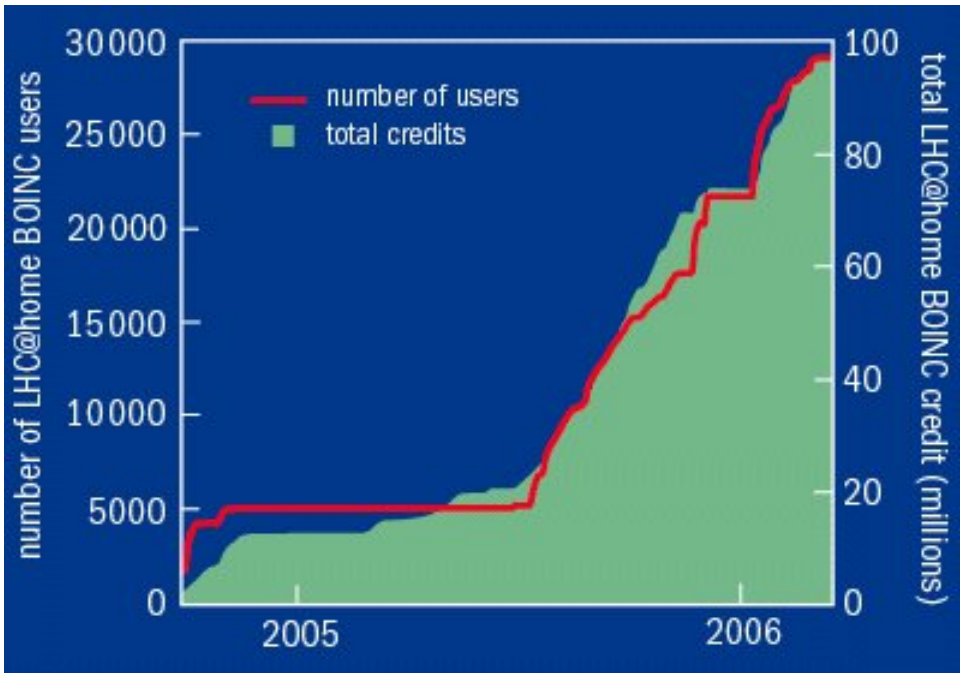


Figure 6.3: Statistics of participants on the LHC@home project and total credits.

Currently SixTrack is the only application ran by LHC@home but other CERN applications of interest to high-energy physics have been studied [77] and are discussed in the following, see section 6.4. In particular, the infrastructure to execute under BOINC a version of the Garfield application is ready and it is planned to implement it to send jobs to LHC@home.

This work was presented at CHEP 2006 [80] and the details, as well as many documents and references, can be found at [81].

## 6.4 Applications ported to BOINC

Here we present the work performed at CERN porting several physics applications with interest to the ATLAS experiment to the BOINC Grid infras-

tructure.

### 6.4.1 PYTHIA

PYTHIA [82] is a program developed and maintained at the Lund University, at Sweden. It is a generator of high-energy physics events, i.e. for the description of collisions at high energies between elementary particles such as $e$, $e^-$, $p$ and $\bar{p}$ in various combinations. It contains theory and models for a number of physics aspects, including hard and soft interactions, parton distributions, initial and final state parton showers, multiple interactions, fragmentation and decay.

The PYTHIA program is based on JETSET, the first of the "Monte Carlo" generators programmed at the Lund University. Over the years, the JETSET and PYTHIA generators grew too much to be maintained in common and in 1997 they were therefore merged to one, under the PYTHIA label. The current version is PYTHIA 6.3.

The physics in PYTHIA is gradually further developed in a number of directions. Additionally, the major current project is the complete rewriting[†] of PYTHIA in C++, while all previous versions have been in FORTRAN 77. It is used also in combination with other tools like Atlfast, which we describe in this section as well.

It is important to notice, as stated in the ATLAS Computing TDR [9], that the C++ version of PYTHIA currently lacks all the functionality of the FORTRAN version. This implies that, given the time needed for development, deployment, testing and validation of new generators, it is clear that FORTRAN support will be required for some considerable time after LHC data is available.

#### BOINC porting

The feasibility of the BOINC porting of simulations involving stand alone PYTHIA programs was studied at CERN. You can find the details of the performed work at [83].

The usual way in which this FORTRAN version of PYTHIA is used is the following one: to modify/create a FORTRAN program including the PYTHIA code and parameters, to compile it and finally to execute the output binary.

The main problem found with this approach is that in this way the created BOINC application would not allow to execute different Grid jobs with different input files. In this approach each workunit would need to send the whole application each time, requiring a different version of the software with each different workunit sent and consuming much more bandwidth

[†]The rewriting of PYTHIA in C++ is known as PYTHIA version 7 or Phytia7

than needed. BOINC is designed to work in the opposite way, i.e. using a single application which typically does not change much and receiving different input files for it.

Then, it seems that a first simple approach to show the portability of PYTHIA can be to take as BOINC input the binary already compiled. In this way, each job will imply a different PYTHIA binary, which is contrary to the BOINC philosophy. Then we will have to take as BOINC application a simple C++ program to execute the input binary (with a system call). Finally we would have to take as BOINC result the output of the input binary. This output can be files or the standard output. In the examples of the PYTHIA web page, the output is sent to the screen as standard output. We have to notice that in this way we cannot do check-pointing. Anyway it would prove the feasibility of the porting.

To be able to check-pointing, we have to modify the code to append the simulation of each event to a file and to call the proper BOINC function after that. In that case, the BOINC application will be different for each job unless we modify our code to make similar simulations but with different input parameters. Then, we have also to modify that code to read those input files. Finally, the BOINC application would be modified and linked with the BOINC libraries code. The input would be our parameters file and the output would be the file in which we appended the generated events. The example programs typically produce the output using PYTHIA functions which send all to the standard output. Nevertheless, is possible to modify the code to write one event to the file each time, together with the simulation seed. In this way, if the application is stopped at any point, the recorded events are not lost and the check-pointing feature will continue generating processes without the need of repeating the generation of the previous ones.

In any case, the Windows port should be relatively easy because it just implies to compile our FORTRAN file using a Windows FORTRAN compiler. The code is completely a stand-alone file and it does not need external compiled software. This is an important advantage.

Finally the porting of standalone PYTHIA code was not continued because of the mentioned problems as well as because of the fact that currently, in ATLAS, PYTHIA is in most cases used together with the Atlfast Fast Simulation sofware. Then, we concentrated our efforts on the Atlfast package, which is described in the following section.

### 6.4.2 Atlfast

Atlfast [69] is the Atlas Fast Simulation package. Depending on the physics problem under study, Atlfast can be used as a good approximation to the full detector simulation and reconstruction phases of the Monte Carlo (MC) reconstruction chain. Fast simulation is performed by smearing the MC truth information directly with resolutions measured in full simulation studies.

Atlfast can run using different even generators that delivers output in n-tuple format. You can find more information about the compatible event generators and Atlfast features at the documentation [69]. While the speed at which Atlfast runs depends on many factors (available CPU, output file format, input file format, complexity of physics channel, etc..), it is in general 4 or 5 orders of magnitude faster than running the full chain.

Currently there are two versions, a FORTRAN one [84] and a C++ one [69] using Athena (see section 1.3.3). The latest FORTRAN version is the 2.60 one, using PYTHIA 6.2 . Currently the development is focused on the C++/Python version using Athena.

As we mentioned in part 4, the reconstructed simulations used for our analysis were done using the Atlfast FORTRAN version coupled with the Monte Carlo generator PYTHIA. Then, we considered the port of that code to the BOINC environment to check the results with and without using it.

In addition, another reason to perform firstly the port to BOINC of the FORTRAN version is that it is much simpler than the port of the one written in C++, as the Athena framework is also used in that case.

The details about the installation, configuration and porting of the FORTRAN version can be found in appendix C.

Finally, the port to BOINC was successful and a number of jobs were sent to three different clients. As BOINC clients, standard desktop machines running Scientific Linux 3 were chosen at CERN IT's department. All the jobs sent to them were returned correctly and their results were verified to be exactly the same, byte by byte, using the BOINC standard validator.

They were compared as well byte by byte with the results obtained using Atlfast in the standard stand-alone way, i.e. without BOINC, obtained that the results were exactly the same ones with and without the use of the BOINC Grid environment.

#### C++ version

We also worked on the porting of the C++ version of Atlfast and some advances were done in that direction. You can find at references [85] [86] the details about the installation of that release as well as links pointing to useful

documentation and many comments about the advances made and problems found.

The current experience makes us pessimistic about the chance of getting any "Athena" application running on BOINC, mostly because

> The lack of portability. It is exemplified on the reliance on a very precise of the C and C++ compilers (it works only on version 3.2.3 of GCC/G++). In addition, in the code there is the general assumption regarding that the code is programed to work in a very specific Linux flavour: Scientific Linux CERN 3. There are also many directory paths hard coded inside the code, mainly Andrew File System (AFS) ones only accessible from CERN machines or research centres with access to that AFS directories. This makes difficult the modification of the code to be able to create stand alone applications which could be able to be executed on any machine from a volunteer.
>
> The use of different external management tools such as Configuration Management Tool (CMT) [87], together with its required files, and Pacman [88] plus some setup scripts to be run per used package. In addition, Python and Gaudi frameworks are used to Glue together different parts of the software. There is as well a large number of environment variables required to be set.
>
> In addition, the size of the whole kit is over 5 GB (very big to be sent with each single BOINC job to the volunteer machines) and it is difficult to find the dependencies to separate the different parts.

The possibility of creating a BOINC application that could be used only inside CERN and in collaborating research institutes was also considered. In that case we avoid the main problem found, the dependency on code and libraries stored on AFS cells but we have to take into account that we loose the huge volunteer computing power coming from the hundreds of thousands of machines donated by the BOINC participants. In the other hand, in the case of having an Atlfast BOINC application running only in CERN and in collaborator institutes then we have an schema similar to a Project Grid, like LCG, with less computing power but with managed and reliable worker nodes.

### 6.4.3 Geant4

Geant4 [89] [90] is a toolkit for the simulation of the passage of particles through matter. Its areas of application include high energy, nuclear and accelerator physics, as well as studies in medical and space science.

The use of Geant4 in a public resource computing project has been studied at CERN. After contacts with some of the main Geant4 developers we found

that BOINC could be useful to run Geant4 simulations, as they typically consume many CPU time while need a relatively small amount of input and output data.

In particular, the "*Geant4 Release Test*" simulation was found to be a good case to explore what we could do for more complex high-energy physics simulations. This is a simple test beam set-up to compare physics results produced by different program versions which is used to validate new versions of the Geant toolkit.

It simulates a beam of particles and their interactions with a detector involving the steps of propagation, interaction and detection. The detector is made of several slices of two different materials, one of them sensitive. It is possible to customize many particle propierties like the type of particle, its energy, momentum and direction as well as the detector propierties like dimensions and materials. We can also customize the physics model to use between the available ones made for Geant4.

The details about the installation, configuration and porting to Linux and Windows of this Geant4 Release Test code can be found in appendix D.

Finally, we ported the Geant4 release testing software to the BOINC environment both in Windows and Linux and set up a BOINC server to demonstrate a production environment.

The experience gained in this process, and the benefits and limitations of BOINC based projects for running high-energy physics applications were presented at the Geant4 workshop 2005 [91] and at CHEP 2006 [92]. The details can be found at [93]. The next step is to consider more realistic and useful Geant4 models, and some work and contacts have already be done in order to achieve that.

### 6.4.4 Garfield

Garfield [94] is a Monte Carlo computer program for simulating gaseous detectors programmed in FORTRAN. It is worth to notice that it is included in the LCG project.

In particular it performs the detailed simulation of two- and three-dimensional drift chambers.

An interface to external simulation programs is provided for the computation of electron transport properties in nearly arbitrary gas mixtures. This is achieved by solving the Boltzmann transport equations for electrons in gas mixtures under the influence of a set of electric and magnetic fields that you specify.

Garfield also has an interface with other simulation programs in order to simulate ionisation of gas molecules by particles traversing the chamber.

Transport of particles, including diffusion, avalanches and current induction is treated in three dimensions irrespective of the technique used to compute the fields.

It is currently used at CERN. In particular, by the experiments ATLAS, CMS and LHC-b.

After some talks with the main Garfield developers at CERN, they proposed us to port his software to the BOINC environment as to solve the high CPU requirements of the model in use for the ATLAS experiment. Some researchers from ATLAS were interested in using BOINC for this case as quickly as possible. Other CERN experiments also showed interests on the project.

After some contacts with them, they provided us the source code plus some example code for testing. The process is presented in detail at reference [95].

Finally, the Garfield program has been ported to the BOINC framework in co-ordination with the software's authors.

After achieving that, real-use jobs were provided by ATLAS researchers and executed with BOINC. A high number of jobs were executed successfully not only in the test machines but also in the 200 CPUs farm that was set up at CERN as well as in the BOINC testbed, presented in detail in section 6.5.2.

## 6.5 Installations performed

### 6.5.1 lxboinc cluster

At CERN's main Computing Centre one hundred dual core old farm PCs (i.e. 200 CPUs) have been recycled to run as BOINC clients. In particular they have dual core Intel Xeon CPUs working at 2.80 GHz, 2 Gb of RAM and the operating system installed on them is Scientific Linux CERN release 3 (SLC3) [96], a CERN customized version of Scientific Linux (SL) [97] which is assembled from freely available Red Hat Enterprise Linux [98] sources . This cluster was called *lxboinc* and it is located at CERN's main Computer Center. Its computing power can be stimated as 200 Ksfp2K, taking into account that 1 Ksfp2K CPU is approximately equivalent to a 2.8 GHz Xeon.

In figure 6.3 we can see how the number of total BOINC credits raised in mid-2005 due to the attachement of those computers.

We are managing these machines using Quattor. The required BOINC client software is installed using a RPM package [81]. This RPM was created by us with to install BOINC as a service, i.e. with the feature of starting automatically when the computer boots. It also installs RPM in the way

that the jobs are executed in the client machines by a particular UNIX user called *boinc* to increase the security of the system. In addition, the BOINC configuration is also performed in the post-install script attaching the installed client to a related BOINC LHC@home account. Another pre-uninstall script was also added to perform the required uninstallation steps deleting the software but also the service and the created user.

Apart from contributing to LHC@home executing BOINC Grid jobs, the PCs are also enabling studies of meta-scheduling mechanisms that should make it possible to implement applications with volunteer resources complemented by limited dedicated ones to guarantee quality of service.

They have also been used as clients for the Garfield testbed as we explain in section 6.5.2 .

A BOINC server RPM has also been produced to automate the installation and management of BOINC servers [81]. The RPMs and the bug corrections have being fed back to the BOINC project.

### 6.5.2 Extremadura testbed

CERN and the regional government of Extremadura region at Spain signed a framework protocol of collaboration in February 2005. Since then, both institutions collaborate in several IT projects.

One of them is the use of BOINC Grid technologies to make available to Extremadura and CERN researchers local unused computing capacity. Indeed, the regional secondary schools host some 80.000 computers (running a Linux Debian flavour called "gnuLinEx" [99]) with abundant spare CPU cycles. BOINC has been studied as a way of helping to put such large potential at the disposal of a number of scientific projects.

In this context, a testbed [100] was set up joining many heterogeneous kinds of machines from those Extremadura region secondary schools (the ones of two different secondary school centres were used with testing purposes), Extremadura University, CIEMAT (in particular from the CETA-CIEMAT research institute) and CERN into a BOINC Grid infrastructure with the aim of running production research jobs for the ATLAS collaboration, in particular using the Garfield simulation program.

Is important to notice that several kinds of different operating systems (different Linux flavours plus Windows) were used. This implied some development on BOINC's job submission and assimilation systems due to some problems found sending the same jobs to different platforms.

In this testbed it was used one machine acting as BOINC server at CERN, in particular a dualcore Xeon working at 3 GHz with 2 Gb of RAM. As

BOINC clients, following clusters of machines distributed at several locations were used:

- the 100 recycled dualcore PCs from the lxboinc cluster at CERN, running SLC3.
- 30 dualcore PCs from participant groups at Extremadura (University and CETA-CIEMAT) running SLC4 and different flavours of Debian.
- 45 desktop PCs from two schools in Extremadura running gnuLinEx.

This involves 305 CPUs, most of them being 2.8 GHz Xeon ones. Then, the approximate computing power of the testbed can be estimated to be about 300 Ksfp2K CPU.

In addition, many jobs were also sent to some test desktop machines running Windows XP at CERN.

The main results obtained are the following ones:

- Delivered jobs: 26,597
- CPU days (as Pentium IV 2.8 GHz): 2,579
- 100% of the jobs delivered and processed successfuly

CHAPTER

# 7

# Comparing LCG and BOINC

After talking about the LCG Computing Grid project and the BOINC distributing computing environment, it is worth to study together both systems to clarify their different possibilities and roles. Due to their different design approaches, they have been created with different purposes and then they are worth for different kinds of jobs, as we show in this chapter.

## 7.1 Technology and infrastructure

In the following lines we compare the LCG services (see 5.1) with the equivalent BOINC ones.

### 7.1.1 Installation, configuration and management

Currently, it is possible to install the LCG middleware using RPMs. There is also the possibility of installing and maintaining them using Quattor. In particular, some Quattor components have been developed to manage and configure LCG [58]. The installation and configuration of this middleware has been made much easier during the past year but it continues being not trivial and it takes time to have a running optimal configuration.

The current version of the LCG Middleware runs on Scientific Linux 3 (SL3) and most middleware testing has been carried out on CERN Scientific Linux 3 (SLC3) [101].

Regarding BOINC, it has been created from the beginning to be installed and managed by volunteers without important computing knowledge so its installation and management is in general much easier. In Windows it uses the Windows Installers and installs itself as a screensaver so it is easy to use. It can also be installed as a service. In UNIX system-like flavours, like Linux or Solaris, binaries are distributed and they work with almost no configuration required after being unpacked. Some third parties have created RPMs for particular projects.

BOINC is officially maintained currently for Windows as well as for Linux and Mac OS X. It has been ported also to other UNIX system-like OS as Solaris, HP-UX, and FreeBSD and the binaries can be found at the official BOINC site [33].

At CERN, we created a RPM package [81] to deploy LHC@home easily across our lxboinc farm, presented in section 6.5, but also in the testbed carried out using this farm and other computers in public schools and research institutions in Spain, as seen in section 6.5.2 .

This RPM was installed and managed using Quattor.

### 7.1.2 Security

There are major differences regarding security between LCG and BOINC due to their different approach.
In LCG there are much more users (researchers of many different research groups and institutes) than providers (private Grid nodes which are maintained and shared by those groups). Then, security is focused in granting and checking certificates to those users. For that, the users are authenticated using Globus' GSI, grouped in VOs managed with VOMS, and CAs are needed to provide the certificates.
In BOINC there is typically just one research collaboration per project so there is no concern on authenticating the users who send the jobs to be executed. The reason is that the Grid managers have to be logged into the BOINC server machine in order to send jobs. Then, there is no need for VOs nor certificates, so neither for using VOMS or CAs.

On the other side, the BOINC providers of computing power (clients) are in most cases volunteered machines so the security is focused in the authentication of those client machines and in the verification of the received results, as well as in the sandboxing* of the executed applications.
Then, in BOINC, each file exchanged between server and clients is digitally signed using the system of public/private keys. In LCG this is not set by default although it can be implemented. In he other hand, in LCG there is

*In computer security, a sandbox is a security mechanism for safely running programs. The sandbox typically provides a tightly-controlled set of resources for guest programs to run in, such as scratch space on disk and memory. Network access, the ability to inspect the host system or read from input devices is usually disallowed or heavily restricted.

no concern on the confidence of the CEs, as they are private and managed by the same Grid collaborators, so there is not need for sandboxing of the executed software nor in the authentication of each single CE.

### 7.1.3 Information service

Regarding information management, there is also an important difference due to the different orientation of the two approaches.

In LCG, there are many different computing and storage sites and in many cases the data and the required software is already in place in the sites in which the jobs are executed. Then, an Information Service is required to identify the right sites which already can have the required data and software.

In BOINC, all the required data and software are sent to the clients by default except in the case in which it is already on them. The system just checks if the required binaries and data is already at the client and it sends it if it is not the case. Then, there is no need for having an Information Service.

This is why one of the desirable features of an application to be executed in a BOINC environment is the requirement of a small amount of data to transfer, although some projects like Climate Prediction require input files of about 500 Mb. In the other hand, this behaviour makes a BOINC client to be easy to install as there is no need to worry about the required software or data.

Regarding the LCG Logging & Book-keeping Service, in BOINC the jobs are also tracked and the information related with job submission, execution, etc. is stored in the MySQL database. This data can be monitored and managed with the included web interface as well as with any MySQL managing tool we wish.

In particular, when a BOINC work unit is created, all the related information is registered into the database. When it is sent by the BOINC server to a client, it registers as well the details regarding the particular client where it was sent, the exact time, the deadline to receive the results back... When the result comes back, successfully or not, this fact is also registered by the server together with many other information which can be used to monitor the process as well as to debug the situation in case of problems. Finally, the useless information is deleted automatically by BOINC or manually, using the management web interface or any other MySQL tool.

### 7.1.4 Workload management System (WMS)

In BOINC, workload management is simpler than in LCG because it works only in pull mode. It means that the BOINC server does not contact the clients to cancel or to suspend/resume already sent jobs as well as to

enquire about their status. Those volunteered machines connect automatically to the server on predefined time intervals and store their status in the database, asking them if there is any action to perform. Managing this database, we can monitor job status and we can change it to cancel if we wish. It will be changed the next time that the CE will contact the server.

Then, there is no need for a mechanism to identify available resources matching the requirements. In addition, BOINC does not need an Information Service to look for them.

Regarding requirement checking, when we register an application into the BOINC database, some related information like the execution platform, operating system and memory requierements are also registered. Then, when a client ask for work to the BOINC server, it informs as well about its capabilities and takes only jobs that matches them.

Then, in BOINC there is no need neither for a mechanism like BDII or RLS to retrieve information about the available resources.

### 7.1.5 Storage Elements and Data Management

Currently, in BOINC there is not the concept of Storage Element, as all the data is stored in the server (or servers) and sent to each Computer Element together to the application and the job to be executed.

Then, it does not exists the concept of LCG replicas as the required data is sent each time together with the job to be executed. Then, there is no need for Replica Catalogues.

As we said in section 3.3.1, it has been considered also the possibility of storing permanently data on the clients having different replicas among them, giving also another kind of credits for 'storage space donation' different from the credits given per 'CPU donation'. In this case, the clients would be at the same time Grid Worker Nodes, Computing Elements and Storage Elements.

In addition, we mentioned as well that currently BOINC is studying to use also a peer-to-peer Grid to share data replicas among the different Worker Nodes and this will allow BOINC projects to have replicas of the data stored in the server distributed on the different clients. It is worth to notice that the concept is slightly different from the LCG Storage Element one.

### 7.1.6 Compute Resource Services

As we said, the BOINC equivalent to LCG Computer Elements are the BOINC clients, which are Worker nodes as well, i.e. where the BOINC Grid jobs are sent by the BOINC server.

In BOINC there is no need for Local Resource Management System (LRMS) as the clients ask for work to the server, working in pull mode. Then, it is not necessary to balance the load of the CPU resources and there is no need of using software like LSF, PBS and Condor, used by LCG.

Gatekeepers are neither necessary in BOINC as the clients work independently, not as a group. The jobs are just sent to individual clients when they ask for job and fit the requirements.

Regarding the Logging and Bookkeeping services, all the informationg regarding the jobs sent by the BOINC server are stored automatically at the BOINC database. It is updated each time that the Worker Node contacts the server so the user can look at the job status looking at the server, without any need of contacting the client directly. Then, there is no need of using, as LCG, GSI security.

In LCG, authentication and authorisation mechanisms based on the VOMS model are used at the CE. In BOINC there is no need as it is the CE who asks for work to the server. Then, by default, i.e. without adding particular software to BOINC, the jobs can only be sent contacting directly the BOINC server and running there the necessary commands so the authentication is performed when the server machine is accessed and the authorisation is not managed as anyone with access to the server can send jobs to any client attached to it.

### 7.1.7 File Transfer Service (FTS)

In LCG, LCG-2 does not provide a particular FTS (it is up to the user) while gLite has one called File Placement Service (FPS) which receives data movement requests and executes them according to defined policies.

In BOINC, the data is sent by HTTP using BOINC functions so there is no particular File Transfer Service.

It is important to notice that BOINC always sign all the data sent to the Worker Nodes while in LCG it is up to the user to do that and most of the times this is not done. Anyway in LCG the nodes are private and are installed in private networks. In addition, it is required authentication and authorisation to be able to send jobs so there is no much worry about the security implications of executing remote code. In BOINC this security enhacements are necessary as most participants are volunteers not attached in any way to the research group and that are connected to the general Internet network.

### 7.1.8 User Interfaces (UIs)

Both kinds of Grid, LCG and BOINC, have UIs but, due to their different approach, they have different features.

In LCG, any user of the collaboration is able to send jobs to the Grid using the UI, provided that he or she has the rights to do that, i.e. the required certificates provided by the CA from a VO. In BOINC, in the default schema, you have to be logged directly into the server machine in order to send jobs so typically all the Grid users send the jobs to the Grid administrator and is this person who send the jobs to the Grid, using the UI.

The UI of LCG interfaces the Workload Management System and it allows to send, to monitor, to manage and to receive results of Grid jobs. It involves a command line tool although some experiments have their Graphical User Interfaces (GUI) to interface the command line UI and make easier to use it.

In BOINC, the command line UI only allows you to send the jobs while it exist in addition a Management Web User Interface which provides some management and monitoring features. The management capabilities of this can be highly increased using additional interfaces to the MySQL database like phpMyAdmin [74].

In addition, in the BOINC Worker Nodes there is a different UI which allows them to attach, to monitor, to manage and to deattach the subcriptions to the projects for which it is working. There exist a command line tool, a GUI and also what is called "User Web Interface". They are used by the participants to register their machines to work for the Grid, i.e. to transform them in clients/Worker Nodes, as well as to manage their preferences and to monitor their performed work.

## 7.2 Summary of the comparison

We present in the following table 7.2 the results fo the comparison performed between the two kinds of Grid studied in depth in this work.

| | LCG | BOINC |
|---|---|---|
| Platforms | Some Linux flavours (mainly SLC) | most versions of Windows, Linux, Mac, Solaris & other |
| Installation & configuration | Not easy | Very simple |
| Management with Quattor | Ok. Quattor was developed for it. | Very easy. |
| Security | Focused on authentication and authorisation: Certificates/CA, VOs/VOMS, GSI | Focused in securing the executed software: Digital sign, sand-boxing |
| Information management | Grid Information Service | No need |
| Logging & Bookkeeping | GLUE, DBII, GMA, LB using Oracle or MySQL | MySQL database |
| Workload | WMS | No need |
| job submitting way | push mode (gLite also pull) | pull mode |
| job language | JDL | XML |
| CE | private managed Grid nodes | mostly volunteered but also can use private nodes |
| SE and DM | data already distributed at Grid nodes & Replicas | data sent to each CE with each single job |
| FTS | FPS in gLite, authentication and authorisation | simple HTTP with code signing |
| Grid UIs | command line UI to send, to manage and to monitor jobs | command line UI to send and Web UI to manage and monitor jobs |
| CE's UIs | No need | command line UI, GUI and Web UI to register the CEs, manage and monitor them |



Table 7.1: Comparison between LCG and BOINC Grids.

# Part IV

# Conclusions

# Conclusions

This work is divided in two parts. In the first one we study the decay of some heavy bosons at the ATLAS detector at LHC, using data simulated and reconstructed with the Atlfast software. At the second part, a study and comparison are performed between the LCG Grid, used by ATLAS and the other LHC experiments, and a complementary kind of Grid called BOINC. Finally, a number of jobs, including the simulation and reconstruction of the data used in our physics analysis, have been executed using BOINC, to study the feasibility of the use of this Grid to perform useful physics calculations.

Regarding the first part, the observability of the decay of some heavy bosons predicted by Extra Dimentions theories is studied. All the studies were done assuming an integrated luminosity of $3 \times 10^5\ pb^{-1}$ for a $Z^*$ mass of 2 TeV.

The conclusions obtained can be summarised as follows:

- We conclude from our analysis that the channel $Z^* \rightarrow b\bar{b}$ would not be observable because we obtain that the signal would be very small compared with the expected background, which in this case is dominated by the reducible one.
- In the case of the decay channel $Z^* \rightarrow t\bar{t}$, the dominant background is irreducible. As in the previous case, the signal is very small compared with the expected background, although the significance is larger. We conclude as well that, in general, the channel $Z^* \rightarrow t\bar{t}$ will be difficult to be detected at the LHC.
- In the decay channel $W^* \rightarrow t\ b$, as in the previous case, the dominant background is irreducible. However, in this case we find that this decay

mode might yield a signal separable from the background. In addition, we also find a significance larger than 5 so we conclude that it would be possible to detect this particular mode at the LHC.

Finally, the analysis was also performed for masses of 1 $TeV$ and the mass dependence of the previous results are studied. We conclude that the significance of the decay modes, i.e. their observability, decreases with the mass. In particular, a significance of 5 (a signal is considered to be observable if the significance is larger than 5) may be achieved below approximately the following masses.

| | $Z^* \to b\bar{b}$ | $Z^* \to t\bar{t}$ | $W^* \to tb$ |
|---|---|---|---|
| $M(TeV)$ | 1.4 | 1.9 | 2.2 |

At the second part of this work, a comparison between the LCG Grid and another complementary kind of Grid called BOINC has been presented.

The LCG project Grid is the official Grid used by ATLAS and the other CERN experiments to store and analyse the huge amount of data that the LHC collider is going to gather, as well as to run other required physics applications like simulation software. It is a public founded project which uses managed computer facilities distributed worldwide, connected to a huge number of different research groups and collaborations.

BOINC is a Grid environment focused on running CPU intensive software, like simulation or reconstruction one. Typically it is configured to execute its calculations on the idle time of desktop computers and it uses distributed computers mostly donated by volunteers although also private clusters as well as any computer connected to a network can be used to perform part of the calculations. This way of working can make this Grid useful to use any desktop or cluster machine at a research institute to contribute to make physics calculations as part of a Computing Grid. It is interesting to notice that it has been shown possible to execute LCG jobs into BOINC Grids, to complement the LCG available resources using the idle time of private nodes or volunteer machines in an inexpensive and easy to manage way.

In order perform our study, different tasks were done at different levels:

- **Previous study:** A deep study of both kinds of Grids is presented at this work, including technology and infrastructure details. Although the way in which those two Grids work is different due to their different approach, the study is presented keeping the same structure to be able to compare their features and similarities in an easier way.

- **Installation, configuration and management:** The installation and configuration of both Grids is presented. In particular, its management using the Quattor farm management software is studied. An introduction, description and some performed work regarding Quattor is also presented. Many Quattor components and packages were ported, corrected or created and a working Quattor environment was set up in Solaris together with a Quattor automated software management system. This experience was used to install and manage a cluster of computers in Linux that we used to execute BOINC Grid jobs at CERN and in a performed BOINC testbed.

- **Adaptation of physics applications:** In BOINC, the applications have to be adapted to be able to use them at the Grid environment. This is due to the fact that they are able to be executed at any kind of volunteered computer which, in general, we do not manage and cannot even know which software it has installed. An study of the feasibility of adaptation to the BOINC Grid environment of a number of physics applications was performed. A brief description of them as well as their importance and use is also given. The following conclusions were obtained:

  - **PYTHIA:** It is a event generator very used in HEP to obtain simulations. The main problem found with this approach is that the usual way in which PYTHIA is used to generate the events implies the modification or creation of a different program to perform different simulations. This means that if you want to execute different simulations in a computer not managed by you, like in a volunteer one, you have to send the whole application with each simulation that you would want to execute. In addition, PYTHIA is usually used in ATLAS as part of the Atlfast Fast Simulation package so we concentrated our efforts on this other one.

  - **Atlfast:** It is the ATLAS Fast Simulation software, which executes an event generator, PYTHIA in our case, and performs the simulation and reconstruction phases of the MC chain in a faster way than the full simulation one. The adaptation to the BOINC Grid environment was successful and a number of jobs were sent to different clients. In particular, the set of events used in our analysis of the decays of the heavy bosons were simulated and reconstructed using Atlfast. They were executed first in stand alone mode and later as Grid jobs. All the jobs were sent to the BOINC Grid, executed and returned correctly. The results were verified to be the same, byte by byte, to the ones obtained by running the simulation without using the BOINC environment.

- **Geant4:** It is a toolkit for the simulation of the passage of particles through matter. It has several areas of application apart from high energy, nuclear and accelerator physics and it is the event generation software used by ATLAS in the full simulation process. The particular test beam set-up, called "*Geant4 Release Test simulation*" and used to validate new versions of the Geant toolkit was successfuly adapted to run under the BOINC Grid environment both in Linux and Windows operating systems. Finally, several jobs were finally sent, executed and returned successfully.
- **Garfield:** It is a MC software to simulate gaseous detectors used by ATLAS and other CERN experiments. It also was ported succesfuly to the BOINC environment. Many real-use jobs provided by ATLAS researchers were executed with success using a BOINC Grid infrastructure. In addition, a BOINC testbed was set up and executed a high number of ATLAS jobs.

- **Installations and test beds:** A cluster of one hundred machines was installed and managed with Quattor and the BOINC Grid software was installed and managed on them using that cluster management software. It executed many of our BOINC test jobs and it also was part of a BOINC test bed that we set-up, together with a big number of machines from some schools, a university and a research center located in Spain, involving a huge number of heterogenous machines. The testbed was successful and a total of 26,597 Grid jobs were delivered, executed and received successfully.

- **Porting to other platforms:** An important part of the LCG Grid software, in particular SEAL and its required external software, was ported to a non-Linux operating system, Solaris, to study its portability. Although Solaris is a Unix implementation, as well as Linux, and they use the same POSIX standards, we found many problems in the way but finally the adaptation of the software was successful. In the BOINC side, some applications were ported to the Windows operating system successfully, as well as to other Linux versions and flavours.

During this time, I was part of the CERN BOINC team at CERN IT's department and some works were published, some presentations were given and some documentation was written [42] [77] [79] [80] [81] [83] [85] [91] [92] [93] [95] [103] [110] [102].

We can conclude that BOINC and LCG are complementary kinds of Grid that can be useful to perform computing tasks for ATLAS and the other LHC experiments.

LCG has very good data distribution, management and storage capabilities that BOINC does not have currently. This means that BOINC cannot be used at present for data storage or distribution.

This also implies that BOINC does not require the important bandwidth and internet speeds that LCG needs. A typical home internet connection is enough so, taking into account that BOINC can be installed and managed easily in most operating systems, we can conclude that it is possible to use almost any kind of computer with a modest internet connection as a part of a powerful Computing Grid. Then, BOINC can provide a huge and inexpensive amount of computing power for running Grid jobs.

In the other hand, such jobs must require low input and output data to take profit of the advantages of BOINC. Typical Grid jobs of this kind are, for instance, physics simulation jobs like event generators. We have proved in this work the feasibility of the use of such kind of jobs in BOINC.

We notice as well that the implementation of a new application to be executed in a BOINC Grid environment requires some previous work to adapt or port such computing application, although this is a work that has to be done only once at the beginning and it is relatively easy after having some experience.

One possible framework in which both kinds of Grid can be complementary is to use LCG as main Grid flavour and to configure it to send jobs to BOINC when their Worker Nodes have too much work or when those jobs have high CPU requirements. Those extra jobs could be sent to use the idle time of other Grid machines like Worker Nodes but they also can be sent easily to any other computer, like to desktop machines in research centers or in universities, schools or other public institutions, as well as to computers provided by volunteers all around the world. An approach of this kind is interesting as a lot of additional computing power can be obtained with almost zero cost. This could be interesting for small research groups with low budget but also for big ones with high computation needs. In the particular case of small groups, BOINC would allow them, for instance, to compute important simulation works without having much resources. The technology to make this possible exists already, as bridges to send LCG jobs to BOINC have been already developed, as seen at section 3.3.3 .

Other possible framework would be to use BOINC only for LCG tasks with high CPU requiremnets, like physics simulation jobs, to execute them completely outside LCG, in volunteer machines. This would reduce significantly the work load of LCG sites. In order to do this, the only requirement is to port the required software, like Atlfast, to the BOINC environment, performing some changes in the code.

In any of those frameworks, the Quattor toolkit has been shown to be

useful

- to manage easily and quickly a large number of Grid machines
- to have a control about the installed software in each one
- to reconfigure it automatically in any set of machines
- to be able to reproduce quickly a clean state in a machine

# Conclusiones

Este trabajo esta dividido en dos partes. En la primera se estudia la desintegración de ciertos bosones pesados en el detector ATLAS en el LHC, usando datos simulados y reconstruídos con el software Atlfast. En la segunda parte, se realiza un estudio y comparación entre el Grid LCG, usado por ATLAS y los otros experimentos del LHC, y un tipo de Grid complementario llamado BOINC. Finalmente, se han realizado con BOINC un cierto número de tareas (en ingés, jobs) incluyendo simulación y reconstrucción de datos usados para nuestro análisis en física, con el objetivo de estudiar la viabilidad de este tipo de Grid para realizar calculós en física.

Respecto a la primera parte, se ha estudiado la observabilidad de la desintegración de ciertos bosones pesados predichos por teorías de Dimensiones Extra. Todos los estudios se han realizado asumiendo una luminosidad integrada de $3 \times 10^5\ pb^{-1}$ para una masa del $Z^*$ de $2\ TeV$.

Las conclusiones obtenidas se pueden resumir de este modo:

- Concluimos de nuestro análisis que el canal $Z^* \rightarrow b\bar{b}$ no sería observable debido a que obtenemos que la señal sería muy pequeña comparada con el fondo (background), dominado en este caso por el de tipo reducible, y sería enmascarada por él.

- En el caso del canal de desintegración $Z^* \rightarrow t\bar{t}$, el fondo dominante es irreducible. Al igual que en el caso previo, la señal es muy pequeña comparada con el fondo esperado, aunque la significancia calculada es mayor. Concluímos entonces que, en general, el canal $Z^* \rightarrow t\bar{t}$ también será difícil de ser detectado en el LHC.

- En el canal $W^* \to t\,b$, al igual que en el caso anterior, el fondo observado es irreducible. Sin embargo, en este caso encontramos que este modo de desintegración puede ofrecer una señal claramente distinguible del fondo. Además, encontramos una significancia mayor que 5, de modo que concluímos que en general sería posible de detectar este modo particular en el LHC.

Finalmente, el análisis se realizó también para masas de 1 $TeV$ y se ha estudiado la dependencia con la masa de los resultados previos. Concluímos que la significancia de los modos de desintegracion, es decir, su observabilidad, disminuye con la masa. En concreto, una significancia de 5 puede ser obtenida por debajo de aproximadamente las siguientes masas (recordemos que una señal es considerada observable si la significancia es mayor que 5).

| | $Z^* \to b\bar{b}$ | $Z^* \to t\bar{t}$ | $W^* \to tb$ |
|---|---|---|---|
| $M(TeV)$ | 1.4 | 1.9 | 2.2 |

En la segunda parte de este trabajo, se presenta una comparación entre la Grid LCG y otro tipo complementario de Grid llamado BOINC.

El proyecto Grid LCG es la Grid oficial usada por ATLAS y los otros experimentos del CERN para almacenar y analizar la inmensa cantidad de datos que el LHC va a tomar, permitiendo usar, además, los programas de física necesarios como por ejemplo el software de simulación. Es un proyecto pagado con fondos publicos que usa instalaciones computacionales administradas y distribuídas por todo el mundo, conectadas a un gran numero de grupos de investigacion y colaboraciones diferentes.

BOINC es un entorno Grid especializado en la ejecución de software que requiere un uso intensivo de capacidad de cálculo, como el de simulación o reconstrucción. Típicamente esta configurado para ejecutar cálculos en ordenadores de sobremesa cuando estos no estan realizando trabajo y usa ordenadores distribuídos donados en su mayoría por voluntarios. También puede usar grupos de ordenadores (clusters) privados, o en general, cualquier ordenador conectado a una red puede ser usado para realizar parte de los cálculos. Este modo de trabajar hace esta Grid útil para usar cualquier tipo de ordenador de sobremesa o de una instalación de un centro de investigación para contribuir a hacer cálculos de física formando parte de una Grid de computación. Es interesante saber que ya se ha demostrado la viabilidad de ejecutar trabajos LGC en Grids BOINC, de modo que los recursos de LCG

pueden ser complementados usando el tiempo no utilizado de ordenadores privados o máquinas de voluntarios de un modo barato y fácil de gestionar.

Para nuestro estudio, diferentes tareas fueron realizadas a diversos niveles:

- **Estudio previo:** Se presenta en este trabajo un estudio profundo sobre ambos tipos de Grid, incluyendo detalles sobre su tecnología e infrastructura. Aunque el modo en que trabajan ambos tipos de Grid is muy diferente debido a su enfoque distinto, el estudio presenta la misma estructura para poder comparar sus características y similitudes en un modo sencillo.

- **Instalación, configuración y administración:** Se estudia la instalación y configuración de ambos tipos de Grid. En concreto, se estudia su administración usando el software de administración de grajas de ordenadores llamado Quattor. Se presenta también una introducción, descripción y trabajo realizado relacionado con Quattor. Varios componentes y paquetes de software de Quattor fueron adaptados, corregidos o creados y un completo entorno Quattor fue configurado, junto con un sistema de administración automático de software adicional integrado en el sistema. Este trabajo fue presentado en un taller (workshop) y publicado. Esta experiencia fue usada para instalar y administrar una granja de ordenadores en Linux que fue usada para ejecutar tareas Grid BOINC en el CERN y en un banco de pruebas, presentado mas adelante en esta sección.

- **Adaptación de software de física:** En BOINC, el software utilizado debe de ser adaptado para poder usarlo dentro del entorno Grid. Eso es debido al hecho de que dicho software puede ser ejecutado en cualquier ordenador ofrecido por un voluntario y que, en general, no es administrado por nosotros ni podemos saber que software tiene ya instalado. Se ha realizado, además, un estudio sobre la viabilidad de adaptación de software al entorno Grid BOINC de un cierto número de aplicaciones. También se proporciona una breve descripción de dichas aplicaciones y su importancia en física. Se obtuvieron las siguientes conclusiones:

  - **PYTHIA:** Es un generador de sucesos muy utilizado en física de altas energías para obtener simulaciones. El problema principal encontrado es que el modo de trabajo típico en que PYTHIA es usado implica la modificación o creacion de diferentes programas para realizar diferentes simulaciones. Eso significa que si queremos generar diferentes simulaciones en un ordenador no administrado por nosotros, como por ejemplo en uno ofrecido por un voluntario, debemos enviar el programa completo con cada simulación que deseemos generar. Además, PYTHIA normalmente es usado en ATLAS como parte del paquete de simulación Atlfast de modo

que nosotros concentramos nuestros esfuerzos en la adaptación de éste último.

- **Atlfast:** Es el software de simulación rápida de ATLAS, el cual utiliza un generador de sucesos, PYTHIA en nuestro caso, y realiza las fases de simulación y reconstrucción Monte Carlo en un modo más rápido que el programa de simulación completa (full simulation). La adaptación al entorno Grid BOINC se completó con éxito y un cierto número de trabajos fueron enviados y procesados en diferentes clientes BOINC. En concreto, se simularon y se reconstruyeron usando Atlfast en BOINC los sucesos usados en nuestro análisis de las desintegraciones de bosones pesados. Primero fueron generados en modo autónomo y después en modo distribuído como tareas Grid. Todas ellas fueron enviadas a la Grid BOINC, procesadas y devueltas correctamente. Los resultados fueron verificados de modo que fueran idénticos, byte a byte, a los obtenidos ejecutando la simulación fuera del entorno BOINC.
- **Geant4:** Es un conjunto de herramientas para simular el paso de partículas a través de la materia. Tiene diversas areas de aplicación aparte de física de altas energías, nuclear y de aceleradores. Se adaptó con éxito para ser ejecutado en entornos Grid BOINC, tanto en diversos sistemas operativos Linux como en Windows, la simulacion oficial de Geant llamada "*Geant4 Release Test simulation*" que es usada para validar las nuevas versiones del programa. Finalmente, varios trabajos fueron enviados, ejecutados y devueltos con éxito. Éste trabajo fue presentado en diversos talleres y publicado.
- **Garfield:** Es un software Monte Carlo de simulación de detectores gaseosos usado por ATLAS y los otros experimentos del CERN. Fue también adaptado con éxito al entorno BOINC. Se ejecutaron con éxito en una infrastructura Grid BOINC un gran número de tareas de utilidad real proporcionadas por investigadores de ATLAS. Además, se puso a punto un banco de pruebas BOINC en el que se ejecutaron un gran numero de tareas ATLAS.

- **Instalaciones y bancos de pruebas:** Se instaló y administró con Quattor un conjunto de cien ordenadores y el software de Grid de BOINC fue instalado y administrado en ellas usando dicho software de administración de ordenadores. Se ejecutaron muchos de nuestros trabajos de test de BOINC en él y también fue parte del banco de pruebas que configuramos, junto con un gran número de máquinas adicionales de escuelas, una universidad y un centro de investigación español, incluyendo un inmenso número de máquinas heterogéneas. Las pruebas

fueron exitosas y el total de 26,597 trabajos grid fueron enviados, ejecutados y recibidos con éxito.

- **Adaptación a otras plataformas:** Una parte importante del software de Grid de LCG, en concreto SEAL y el software externo requerido por él, fue adaptado a un sistema operativo no-linux, Solaris, para estudiar la viabilidad de su adaptación. A pesar de que Solaris es una implementación Unix, al igual que Linux, y que ambos usan los mismos estándares POSIX, nosotros encontramos varios problemas en el proceso pero finalmente la adaptación fue completada satisfactoriamente. Con respecto a BOINC, varias aplicaciones fueron adaptadas con éxito al sistema operativo Windows, además de a diversas versiones y tipos de Linux.

Durante este tiempo, estuve trabajando como parte del grupo BOINC del CERN en su departamento IT (Information Technology) y varios trabajos fueron publicados, varias presentaciones fueron ofrecidas en talleres y cierta documentación fue redactada [42] [77] [79] [80] [81] [83] [85] [91] [92] [93] [95] [103] [110] [102].

Podemos concluir que BOINC y LCG son tipos de Grid complementarios que pueden ser útiles para realizar tareas de computación para ATLAS y para el resto de los experimentos del LHc.

LCG tiene unas capacidades muy buenas de distribución, administración y almacenamiento de datos que BOINC no tiene actualmente. Eso significa que a dia de hoy BOINC no puede ser usado para almacenamiento o distribución de datos.

Eso también implica que BOINC no requiere el gran ancho de banda o velocidades de conexión a internet que LCG necesita. Una conexín casera típica es suficiente de mode que, teniendo en cuenta que BOINC puede ser instalado y administrado facilmente en la mayor parte de los sistemas operativos, podemos concluir que es posible usar casi cualquier tipo de ordenador con una conexión a internet modesta como parte de una potente Grid de computación. De ese modo, BOINC puede proveer una inmensa y barata cantidad de poder de computación para realizar trabajos Grid.

Por otra parte, dichos trabajos deben requerir poca cantidad de datos de entrada y salida para aprovechar bien las ventajas de BOINC. Trabajos de Grid típicos de ese tipo son, por ejemplo, trabajos de simulación de física como generadores de sucesos. Hemos probado en este trabajo la viabilidad de ejecución de dicho tipo de trabajos en BOINC.

Hemos notado tambien que la implementación de una nueva aplicación a ejecutar en una Grid BOINC requiere un trabajo previo para adaptar dicha aplicación, aunque esta es una tarea que debe ser hecha solo una vez al comienzo y es relativamente sencilla despues de tener cierta experiencia.

Un posible entorno de trabajo en el cual ambos tipos de Grid pueden ser complementarios sería el uso de LCG como tipo de Grid principal configurándolo para enviar tareas a BOINC cuando sus nodos de trabajo tengan demasiado trabajo o cuando dichas tareas tengan requisitos altos de CPU. Dichos trabajos adicionales podrían ser enviados con el objetivo de usar el tiempo no utilizado de otras máquinas en la Grid, como nodos de trabajo, pero también podrian ser enviados fácilmente a cualquier otro ordenador como por ejemplo ordenadores de escritorio en centros de investigación o en universidades, escuelas o otras instituciones públicas, al igual que a ordenadores proporcionados por voluntarios en cualquier parte del mundo. Un enfoque de este tipo es interesante debido a que una gran cantidad de capacidad computacional puede ser conseguida con prácticamente coste cero.

Esto podria ser interesante para grupos de investigacion pequeños con bajo presupuesto pero tambien para grandes grupos con grandes necesidades de computación. En el caso particular de grupos pequeños, BOINC podría permitirles, por ejemplo, realizar importantes trabajos de simulación sin tener muchos recursos propios.

La tecnología necesaria para hacer esto posible existe ya, debido a que ya se han desarollado puentes para enviar trabajos de LCG a BOINC, como vimos en la sección 3.3.3 .

Otro posible entorno de trabajo seria el usar BOINC solo para tareas LCG con altos requisitos de CPU, como trabajos de simulación de física, para ejecutarlos completamente fuera de LCG, en maquinas proporcionadas por voluntarios. Ello reduciría significantemente la carga de trabajo de las instalaciones LCG.

Para ello, el unico requsito es la adaptación del software necesario, como Atlfast, al entorno BOINC, realizando los cambios en el código explicados en este trabajo.

En cualquiera de dichos entornos de trabajo, la herramienta Quattor se ha demostrado útil para:

- administrar fácil y rápidamente un gran numero de máquinas Grid
- tener un control del software instalado en cada ordenador
- reconfigurarlo automáticamente en un conjunto dado de máquinas
- ser capaz de reproducir rápidamente un estado de trabajo limpio en un ordenador dado

# Part V

# Appendices

APPENDIX

# A

# Quattor tasks performed at CERN

We present in this appendix the tasks performed at CERN's IT department regarding the porting, correction and creation of Quattor components and also of the Quattor automated software management system.

Part of the results of this work as well as the experience obtained were used to install and manage the lxboinc cluster, presented in section 6.5 and to create the BOINC RPM presented in the same section.

## A.1 Use case. Implementation of Quattor in Solaris

To set up a working Quattor environment running under the Solaris OS at CERN, we ported, corrected and created several Quattor components and packages.

### A.1.1 Porting

Regarding porting, the following ones were modified by us:

- Ported components: ncm-afsclt, ncm-krb4clt, ncm-krb5clt, ncm-srvtab, ncm-access_control
- Ported packages: cdp-listend, cdispd, spmacf, arc script)

The differences between Linux and Solaris raised the following issues:

- Missing required scripts (like /etc/rc.d/init.d/functions)
- Use of different paths (location of the binaries, lock directories, startup scripts, log files, configuration files...)
- Need of different environments
- Use or not use of some package features (like the configuration of the firewall in Solaris)
- Check of different Quattor CDB entries
- Differences in the startup scripts (sh instead of bash, different way to declare the scripts to be started...)
- Different logging methods and log paths (in Linux is used initlog)
- Different configuration information (like the one in /etc/syslog.conf)

### A.1.2 Corrections

The following components and packages required different corrections:

- Fixed components: ncm-etcservices, ncm-automounter, ncm-postfix, edg-caf-perl
- Fixed packages: afssetup, operator, spmacf, ccm, pkgbuild (pkgt), automounter, srvtab, spma, ncm-ncd, arc, cdispd, cdp-listend

The most common problems corrected includes:

- Bugs reported by users
- Programming mistakes
- Correction and optimization of configuration files
- Fatal errors instead of warnings showed in case of non existing but not required files
- Missing tests about the existence or access rights to files or directories
- Missing of important system checks before making changes in the system
- Wrong hard coded paths
- Change of run levels to avoid shutdown errors

- Unused or confusing variables
- Messy and confusing code which needed to be rewritten
- Other errors (misprints, wrong end of lines, spaces instead of tabs...)
- Creation, improvement, update and/or correction of the documentation

### A.1.3 Creation

Several components and packages were created or deleted.

- Created: ncm-postfix, pineconf, boinc-client
- Deleted: srvtab (files moved to arc and functionality added to ncm-srvtab)

This boinc-client package is the one which was used for installing boinc in the lxboinc farm, presented in section 6.5 .

## A.2 Quattor software management system

The Quattor automated software management system makes possible to install, unistall and configure software in an automated manner when we make changes on the CDB. These changes can be, for instance:

- a version of a package (then, it is installed at this moment or upgraded/downgraded if a different version is already installed)
- any information that a component can get from the CDB (then, the component will take into account this changed information when it is configured at the end of the process)

This system includes the following packages and components: cdp-listend, ccm, ncm-cdispd, ncm-spma, spma and ncm-ncd. The porting and correction of some of them was required and performed by us in order to have a working environment to be used for our needs. More detailed information as well as the slides presented in an ELFms meeting can be found at [102].

We present below how this system works. It is shown as well in figure A.1 .

- We edit a profile in the CDB and we make a commit
- Then, the CDB sends an UDP packet informing about it
- If the cdp-listend daemon is running, it

  - detects the packet,
  - logs what it does to /var/adm/messages,
  - runs the ccm-fetch utility (installed by the ccm package)

- ccm-fetch retrieves the XML configuration profiles to /var/lib/ccm
- If the cdispd daemon is running, it
  - detects the new incoming profiles
  - logs what it does to /opt/edg/var/log/ncm-cdispd.log
  - makes a list of the components to dispatch (+)
  - configures the ncm-spma component (with "ncm-ncd –configure spma") (+)
    - the ncm-spma component automatically executes the "spma" program (+)
  - configures all the components included in the previous list (with "ncm-ncd –configure all")

The steps marked with a plus sign (+) are executed only if there are certain entries at the CDB so we can customize them at any moment.

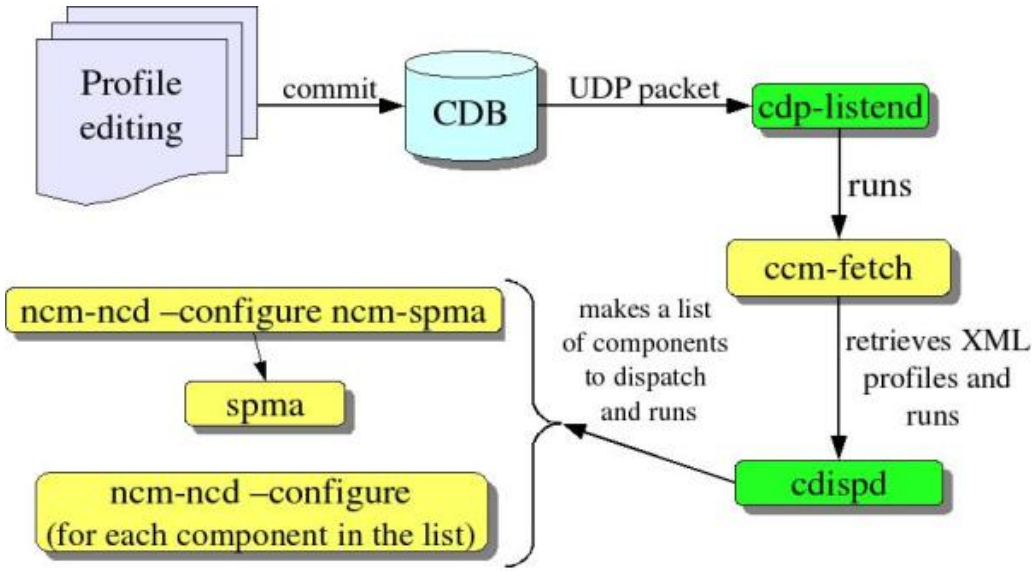


Figure A.1: Schema of the Quattor automated software management system.

Then, we have shown that modifying Quattor CDB entries we can configure Quattor to manage a determinate set of machines

- keeping up-to-date all the software and configuring themselves it in an automated manner
- having those machines never updated by Quattor
- updating automatically only the configuration, not the version of the installed software
- configuring and/or upgrading with Quattor only a particular set of components

APPENDIX

# B

# Porting of LCG software

At CERN, we ported to Solaris the LCG Physics analysis environment and its external software [103]. Our goal was twofold. On one hand we wanted to address the interest of LHC researchers to have the analysis environment working on Solaris. On the other hand we wanted to assess the difficulty of porting the code, written in Linux without particular attention to portability to other UNIX implementations .

We started with the LCG External software [104] in order to use it to build the LCG applications [105] such as POOL and SEAL and we continued with the port of SEAL.

We chose Solaris 8 as the target environment because it is the leading commercial Unix implementation and CMS users had expressed interest in having the LCG environment on it. Although the interest of the LCG community in this platform seems to be decreasing, it still is a good example of a Unix implementation other than Linux and most of the problems found would apply in porting to any other commercial Unix or POSIX 1003.1 1003.2 compliant operating system. We chose the GNU compilers because they have been used to develop most of the code and their portability guarantees porting with minimal code changes. The Sun compilers seem to provide better performance on Solaris, but lack of support for some C++ template features, such as templates with template arguments would make the port more complicated. We do not have the GNU binutils package available on Solaris so we have to use the Sun linker to make shared libraries and dynamic executables. We also have to use the Solaris dynamic linking mechanisms.

In this environment the GNU options, such as "shared", are not understood by the linker, so we have to use the "-Wl" gcc option to pass options to the sun linker directly. The typical options required are "-G" to make shared libraries and "-h" to set the internal name of the libraries. We have built all object files, including the External Software ones, as pure text using the "-fPIC" gcc option in order to avoid errors due to relocations against non-writable, allocatable sections when building shared libraries and dynamic executables. We use the default instruction set architecture available in our environment, ILP32. This is SPARC V8 architecture as defined by Version 8 of the SPARC Architecture Manual. However we have to emulate the behaviour of the Sun compiler (ANSI C plus K&R C compatibility extensions) by defining __STDC__ = 0 for the system headers to define the explicit 64 bit types, such as int64_t and uint64_t required by SEAL in 32 bit environment. These types are implemented using the "long long" type of gcc. We also had to make sure that other "longlong" types defined by the ReflectionBuilder in SEAL are identified with the explicit 64 bit types as otherwise they were not recognised as identical when building the dictionary. In this environment we have ported SEAL 1.3.3 and all the required External Software.

## B.1 LCG External Software

Most of the following ported programs are installed using the "configure + make + make install" method with the appropriate options. There are many variations in which they also use some "make check" or "make test" step or no "configure" or "make install" command is necessary so the method cannot be guessed before reading carefully the installation instructions. In other cases the installation method is completely different involving other tools, ad hoc scripts, compilation of special files in some directories, links, etc. In principle we ported just the version required by SEAL and POOL but we also ported other program versions in the cases in which the SPI project supports them or when the required version gave some problem.

The following required packages were finally ported without major problems: Bison, BLAS, CMake, Expat, GSL, Jam, MySQL, Otl, Oval, Pcre, QMTest, SLOCCount, XercesC and Zlib.

The following ones required modifications in the code because of bugs, problems in the installation and / or non-trivial compilation options. Some of them also needed additional non-trivial software: Boost, bz2lib, CLHEP, CppUnit, Doxygen, GCC-XML, LAPACK, MySQL++, Python, PyXML, Root, UnixODBC, Uuid and wxPython.

There are also some packages that cannot be ported because they are profiling tools or memory debuggers working only on x86 linux machines. Nevertheless, it is possible to configure SEAL and other packages requiring this packages to ignore them. They are: IgProf, OProfile and Valgrind.

A special external software is the Software Configuration, Release And Management (SCRAM) tool [106], required by all the LCG analysis tools. It is the software configuration, release and management tool used by the CERN applications and it is used to build SEAL and the other LCG software. It is installed using a Perl script. We just had problems finding the sources and the installation instructions due to the lack of documentation. You can find them at [107]. Currently, the web site has been updated and improved [108].

SCRAM downloads the required sources from a CVS repository and then performs the required installation commands generating the required Makefiles from different locations, using the configuration files at the SEAL CVS repository, written in xmlXMLyle. These Makefiles are not stored locally so in many cases we have no access to the compilation commands which may complicate debugging.

## B.2 SEAL

The installation instructions [109] inform us that SEAL is installed and configured using the SCRAM tool. The installation method involve a Concurrent Versions System (CVS)* check out of some SEAL configuration files, a SCRAM bootstrap command and a SCRAM build command in the suitable directory. To modify the SEAL source files, we had to create our own CVS repository and check the entire SEAL CVS repository in.

For each change in the code we had to check it out, modify it and then check it in with the appropriate CVS tag (SEAL_1_3_3), because this tag is also in the configuration files and is the one which SCRAM will download.

We also had to create our own CVS repository for the SCRAMToolBox. This is a set of configuration files where SCRAM finds all the information about the compilers and external programs that SEAL will use.

Afterwards, we had to change the SEAL configuration files to point to these repositories instead of to the default ones. In the check out step, we hit a CVS bug where it complains about not being able to open a temporary file. We circumvent this by creating an empty file with this name. The bug was be solved in the following version of the package.

Many SCRAM configuration files have a separated block for each installation architecture so we just had to add the missing Solaris ones. In some of the configuration blocks the Solaris architecture was already there but only with the Sun CC compiler so we had to add a new one using gcc.

We had to change all the required external software paths as well as the compiler paths and options. The compiler options are distributed on different files that we had to modify.

*CVS is an open-source version control system. It keeps track of all work and all changes in a set of files and allows several developers to collaborate.

We had to configure the link options to use the Sun linker. As part of the configuration, we also had to add all the required system libraries to the link command line in order to avoid symbol referencing errors.

The lcgdict script used tcsh syntax when invoking /bin/csh, while scripts within gccxml used bash syntax when invoking /bin/sh. Csh and sh are not identical to tcsh and bash in Solaris so the references had to be corrected.

We had to change wrong "defines" in several places in order to cater for Solaris, in particular we had to replace "ifdef __linux" by "ifndef _WIN32" in seal/Scripting/ PyLCGDict/src/LCGDictWrapper.cpp.

We found that seal/Foundation/SealBase/src/SharedLibrary.cpp hard-coded the Linux method to find the link map for dynamic linking, so we had to add support for Solaris using the dlinfo() library call.

Some SEAL files in the CVS repository were missing the version tag, therefore SCRAM was not downloading them and SEAL could not find them. All the files under "seal/Scripting/PyLCGDict2" only had the SEAL_1_4_0 tag. Similarly the seal/Documentation/WebSite/doxygen.css file, and the seal/Documentation/WebSite/workbook directory did not have any tag.

APPENDIX

# C

# Atlfast porting to BOINC

## C.1 Atlfast compilation and configuration

Firstly we downloaded and compiled the code from the official web site [84]. To compile properly, it required the correction of some paths in the Makefile. Then, we got some undefined reference errors due to many missing libraries and we corrected them one by one adding some flags to the proper variable inside the Makefile. We also had to take care of pointing to the static libraries instead of the default shared ones. This is because a BOINC application typically is sent to a computer from a volunteer and we cannot know in advance which software is already installed on it. Then, we have to compile all statically to avoid dependencies on any kind of external software. We also found some additional errors which were solved setting up the right environment variables for the compilation.

After solving those problems we were able to compile the code and to produce and execute successfully the demo application to test the code, producing a output file with the results. The required steps are explained in detail at [85].

Then, to run real code we had to create a new KEYPRO condition modifying the code. In particular, to perform our simulation and reconstruction in the ATLAS detector of the $Z^* \longrightarrow b\bar{b}$ decay at 1000 GeV we had to add the following KEYPRO 'if' condition

```
C ------ NEW PROCESSES----------------------------
```

```
IF(KEYPRO.EQ.12) THEN
   MSEL = 0
   MSUB(141) = 1
   PMAS(32,1) = 1000.
   MSTP(44) = 3

   MDME(289,1) = 0
   MDME(290,1) = 0
   MDME(291,1) = 0
   MDME(292,1) = 0
   MDME(293,1) = 0
   MDME(294,1) = 1
   MDME(295,1) = 0
   MDME(296,1) = 0
   MDME(297,1) = 0
   MDME(298,1) = 0
   MDME(299,1) = 0
   MDME(300,1) = 0
   MDME(301,1) = 0
   MDME(302,1) = 0
   MDME(303,1) = 0
   MDME(304,1) = 0
   MDME(305,1) = 0
   MDME(306,1) = 0
   MDME(307,1) = 0
   MDME(308,1) = 0
   MDME(309,1) = 0
   MDME(310,1) = 0

   PARU(121) = 0.878
   PARU(122) = 0.878
   PARU(123) = 0.878
   PARU(124) = 0.878
   PARU(125) = 0.878
   PARU(126) = 0.878
   PARU(127) = 0.878
   PARU(128) = 0.878

   PARJ(180) = 0.878
   PARJ(181) = 0.878
   PARJ(182) = 0.878
   PARJ(183) = 0.878
   PARJ(184) = 0.878
   PARJ(185) = 0.878
```

```
      PARJ(186) = 0.878
      PARJ(187) = 0.878
      PARJ(188) = 0.878
      PARJ(189) = 0.878
      PARJ(190) = 0.878
      PARJ(191) = 0.878
      PARJ(192) = 0.878
      PARJ(193) = 0.878
      PARJ(194) = 0.878
      PARJ(195) = 0.878

      MSTP(88) = 4
      MSTJ(11) = 3
      PARJ(54) = -0.07
      PARJ(55) = -0.006
      PARP(82) = 1.8
      PARP(84) = 0.5
      MSTP(128) = 0
      PYLISTI = 0
      PYLISTF = 0
   ENDIF
```

in the same place in which the other KEYPRO conditions are. We also had to modify a data file to tell ATLFast which KEYPRO we want to simulate. In our case it is the number 12 so we put

```
12      ----KEYPRO
```

Performing that steps we simulate and reconstruct our $Z^*$ decay instead of the demo one.

### BOINC porting

The feasibility of the BOINC porting of Atlfast using PYTHIA as event generator to simulate and reconstruct events was studied at CERN. The details of the performed work can be found at [85].

The first step is to find in the code the calls to open and close input and output files and to replace them by the proper BOINC functions.

Regarding the required data files that the application needs to work properly, it needs a big amount of them and it would be annoying to declare one by one when we create the jobs. Then, the option of creating a compressed ZIP file including all the required data files was chosen.

In order to use ZIP files within BOINC we had to use the BOINC ZIP internal function. At that time, such BOINC functionality could be used only with programs written in C but Atlfast is written in FORTRAN. A wrapper function to use ZIP features within BOINC with FORTRAN programs had to be implemented by us and it was used successfully in the porting. After that, it was included inside the BOINC code, which had also previous contributions by the LHC@home team, and fed back to the BOINC project team which currently has it available and used by other FORTRAN applications ported to BOINC. You can find the details about it at [110].

It was also neccesary to modify the PYTHIA subroutine used to accept input data to change our simulation depending on an input file. It is not optimal to be ran under BOINC an application which performs always the same simulation. To study the feasibility of the port, the code was modified to read the Mass of the Z* by reading this parameter from an input file. This can be easily changed to allow us to completely change the simulation depending on our needs. All the details of the process can be found at [85].

Finally, to be able to compile the binary, we had to include the required BOINC libraries in the Makefile.

Then, we can execute the obtained binary to run the BOINC application in standalone mode. At that point we check that it works properly looking at the output and checking that the results shown by our ported BOINC Atlfast application are exactly the same as the ones obtained by the original one without any modification.

Finally we can register the application in the server and create the jobs. For that we just need:

- the binary: obtained after compiling the Atlfast code after the modifications performed to use the BOINC functions and to compile properly
- the required data files: in our case, just the ZIP file created by us which includes all of them
- input files: in our case, a text file including our defined variables which change for each simulation which we decided to be just the mass of the $Z^*$ in our test
- output file

At that point, a problem regarding the maximum CPU bound was found. BOINC includes a mechanism to stop automatically jobs which consume more than a given CPU time, to prevent runaway processes. Our application

consumed more CPU time that the default limit so we had to add a proper BOINC flag increasing that limit.

After solving that problem and checking that the application runs properly on standalone mode, we registered it properly on the BOINC database using using the BOINC scripts. In order to do that, it was necessary to create our work unit and result templates to define the job. In those XML files we have to declare the name of the binary file to execute and their required options as well as the names of the input and output files.

Here we show the work unit template, in which we just declare the input file, called "variables", and we re-define the maximum CPU time bound:

```
<file_info>
   <number>0</number>
</file_info>
<workunit>
    <file_ref>
        <file_number>0</file_number>
        <open_name>variables</open_name>
    </file_ref>
    <rsc_foops_bound>100000000</rsc_fpops_bound>
</workunit>
```

As an example of result template, we have here the following one in which we see that ATLFast produces just an "output.txt" file:

```
<file_info>
    <name><OUTFILE_0/></name>
     <generated_locally/>
     <upload_when_present/>
    <max_nbytes>10000000</max_nbytes>
    <url><UPLOAD_URL/></url>
</file_info>
<result>
    <file_ref>
        <file_name><OUTFILE_0/></file_name>
        <open_name>output.txt</open_name>
    </file_ref>
```

APPENDIX

# D

# Geant4 porting to BOINC

## D.1 The Simulation

The simulation runs 5000 times the following event: the interaction of one single particle with a detector involving its propagation, interaction and detection. The detector is made of several slices of two different materials, one of them sensitive. We can customise the type of particle, its energy, momentum, direction as well as the detector dimensions and materials. We can also customise the physics model.

Is important to notice that the time to perform each simulation is more or less proportional to the energy we select for the incoming particles. This time also scales approximately linearly with the number of events we simulate. It also depends on the used physics model we choose.

The code is written in C++ and most of the times we care about modifying a single file. We need Geant4 and the CLHEP (Class Library for High Energy Physics) libraries to compile the binary but if we compile it statically, we can make it dependent only on the C++ libraries. To compile the binaries we just need to go to the StatAcceptTest directory, set the right environment variables in the buildSetup.sh file and run gmake to execute the Makefile. It is very simple and it does not use external programs like autoconf or automake.

The binary sends the output to the screen. The output was originally HBOOK histograms, but it has been modified to produce just plain text to ease portability to Windows. Its format is defined in the StatAccepTestAnal-

ysis.cc file. To address Windows and BOINC constraints, we had to modify the code to write the results directly into a predefined file rather than redirect the standard output as originally done in Linux. The main results are deposited energy and momentum, and they are divided in transversal (perpendicular to the direction of the incident particle) and longitudinal (parallel). The Geant4 random number generator uses also a time variable for the seed so each time we run the simulation we should have different results.

The current binary should work on any Linux version where the gcc libraries compatible with version 3 are installed. In particular, it was compiled on SLC3 (Scientific Linux CERN version 3.05) using gcc 3.2.3. It also works in Debian 3.0 with gcc 3.3.5. It was also executed on Windows XP under the Cooperative Linux (coLinux)* environment [111].

This SLC3 binary does not work on SLC4 as gcc 3.4.3 is the default compiler. Installing gcc 3.2.3 on SLC4 provides the missing libraries and makes it work (the test was done also in Windows XP under the coLinux environment).

## D.2 Boincification

The current stable version of the BOINC client libraries is taken from the CVS server:

```
cvs -d  :pserver:anonymous:@alien.ssl.berkeley.edu:\
/home/cvs/cvsroot checkout -r stable boinc
```

The versions of required packages for building BOINC on SLC3 are too old. Specific versions had to be generated. Some SLC3 libraries are not at the recommended level but nevertheless did not cause any problem. For other packages, it was necessary to install the newer versions in a non-system directory because just replacing the old version would break many other applications. The newer ones were placed to be found first by the BOINC generation procedure. As BOINC client and libraries are built statically, the libraries are not needed at execution time so it is worth to do that.

The Geant4 programs were built statically, but they still depend on some system shared libraries. The dependencies are shown below:

```
$ ldd mainStatAccepTest
libstdc++.so.5 => /usr/lib/libstdc++.so.5 (0x0019b000)
libm.so.6 => /lib/tls/libm.so.6 (0x00111000)
libgcc\_s.so.1 => /lib/libgcc\_s.so.1 (0x002a4000)
libpthread.so.0 => /lib/tls/libpthread.so.0 (0x00266000)
```

*coLinux is software that lets Microsoft Windows cooperate with the Linux kernel to run both in parallel on the same machine. It allows to execute Linux software in Windows machines without important changes in the code.

```
libc.so.6 => /lib/tls/libc.so.6 (0x00b1d000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x00d62000)
```

These libraries are the default SLC3 and gcc 3 libraries and these induce the compatibility restrictions mentioned above.

The input to the program is a Geant4 macro and no direct user interaction is required. Thus, this program fits nicely into the BOINC framework. To complete the integration in BOINC, the Geant4 test code was slightly modified to always accept a fixed file name as input script. Thus, the Geant4 script is provided with the job by the server (and it can be different for different jobs). The standard output where results are written is redirected to a file with fixed name using the C++ constructor shown below:

```
// save original sbuf
std::streambuf* cout\_sbuf = std::cout.rdbuf();
std::ofstream   fout(resolved\_name\_out);

//redirect 'cout' to a 'fout'
std::cout.rdbuf(fout.rdbuf());
  (...)

// restore the original stream buffer
std::cout.rdbuf(cout\_sbuf);
```

Other methods were considered. It is possible to overload the Geant4 descriptors G4cout and G4cin but because of the lack of documentation, the idea was abandoned. One could also set some BOINC flags to force the BOINC system to put the standard output in the given file but this is neither well documented and could not be made to work. The calls used to open the data files needed by the Geant4 binary other than for the input file were not touched. In fact, a lot of fstream C++ style IO is present in the code and are not trivial to wrap. What was done, from within the C++ program, was to set environment variables that make sure that all the needed files are under the current directory.

Some BOINC calls have to be inserted in the code: boinc_init() and boinc_finish() as prologue and epilogue, boinc_zip() to package, compress and decompress the input and output data files. Compression is used to optimise file transfers and the boinc_zip() function makes the program independent of the zip program on Windows and any other BOINC supported platform. These calls to BOINC internals are resolved by adding the BOINC libraries and POSIX threads in the Geant4 Makefile.

```
LDFLAGS += -L/test/boincclient/install/lib
LDLIBS += -lboinc_api -lboinc
CXXFLAGS += -pthread
INCFLAGS += -I/test/boincclient/install/include
```

The order in which the BOINC libraries are inserted in the Makefile is relevant. Wrong ordering leads to unresolved references. The results, output of the Geant4 test code, contains a line informing about the time needed to perform the simulation (user, real and system time) like

```
Number of events processed : 10
User=10.48s Real=11.07s Sys=0.05s
```

Thus, different jobs always have different results, even with the same seed, just because of that line. As we mentioned previously when describing BOINC, it is necessary that identical jobs send back identical results, byte by byte. This is why BOINC sends the same job to 2 or more clients. This allows to check if the results are correct. BOINC servers uses any BOINC client machine provided that it is attached to the project. This machine may be in a bad state leading to incorrect results and it also may happen that it will never send back results.

Then, this problem should be corrected, modifying the code, in order to use this application in BOINC. The offending line in the Geant4 output was found to be generated by line 236 of G4RunManager.cc:

```
{ G4cout << "  Number of events processed : "
         << n\_event << G4endl; }          // line 235
  G4cout << "  "  << *timer << G4endl;    // line 236
  (...)
```

The solution was then to just comment out the line 236 in the source file.

## D.3 Porting to Windows

As suggested by the Geant4 documentation in [112] the Cygwin POSIX environment [113] from http://www.cygwin.com/ was installed so that we could use UNIX tools to manage the code, such as make.exe as a make tool g++.exe as a tool to analyse source file dependencies and create dependency (.d) files Several other UNIX tools like cp, mv, rm, touch, etc.

At the same time a recipe is provided by the Geant4 team to be able to use the Microsoft Visual C++ compiler in this environment. The recipe in question consists of adding to the cygwin.bat start-up file of Cygwin32 all the MS DOS commands found in the file vcvars32.bat provided in MS Visual C++ (in the VC++ .NET compiler installation directory, and usually located inside the Common7/Tools directory). This sets the environment for MS Visual C++, i.e. set the paths to libraries, include files, and executables for MS Visual C++.

In our experience, the Cygwin startup file appeared from Windows in c:\cygwin\cygwin.bat and the path to the VC++ script was the following

```
C:\Program Files\Microsoft Visual Studio.NET 2003\Common7\
Tools\vsvars32.bat
```

This way of working could be very interesting for other BOINC projects as it allows you to produce pure Windows executables while using UNIX tools to manage your code.

Geant4 was installed as described in the doc in [114] and CLHEP 1.9.2.1 for VC++ 7.1 was downloaded from

```
http://proj-clhep.web.cern.ch/proj-clhep/DISTRIBUTION/
clhep-1.9.html
```

We had to set the following environment variables:

```
G4SYSTEM=WIN32-VC
CLHEP_BASE_DIR=c:/local/dirLHCAtHome/clhep-1.9.2.1-win32-vc71
```

before running make from $G4INSTALL/source. This builds one library for each "leaf" category (maximum library granularity) and automatically produces a map of library use and dependencies.

Then we have to run

```
make global
```

in order to build global libraries, one for each major category The hadronic physics lists were compiled by doing

```
cd $G4INSTALL/physics_lists/hadronic;
make
```

Once the Geant4 libraries were built, we were able to make the Release Test program. To do so the buildSetup.sh setup script had to be edited as follows

- Change G4SYSTEM from Linux-g++ to WIN32-VC.
- Change CLHEP_BASE_DIR to point to the CLHEP that was downloaded.
- Change CLHEP_LIB from CLHEP to libCLHEP-1.9.2.1.lib. CLHEP is the UNIX name. Although the default Windows name is CLHEP.lib, this name appeared only as a broken symbolic link.
- Comment out GUI_USE_TCSH because this option is not supported in Windows.

- Change G4WORKDIR from $PWD to the explicit directory in Windows syntax: c:\local\dirLHCAtHome\StatAccepTest . This is because $PWD in a Cygwin bash shell gives the path as \cygdrive\c\local\dirLHCAtHome\StatAccepTest and this syntax is not understood by the Microsoft compiler.

- We also had to add the time.h include file to mainStatAccepTest.cc to be able to use the time() function.

- To port the program to BOINC environment the same code changes as in Linux were done.

- Additionally the referenced BOINC include and source files had to be added to the project. For instance to find boinc_win.h we would do

```
cd /cygdrive/c/local/dirLHCAtHome/boinc
du -a |grep boinc_win
Or to find were md5_init is defined or referenced
cd /cygdrive/c/local/dirLHCAtHome/boinc
grep -r md5_init *
```

The source files moved to the project were taken into account as dependencies for compilation and linking provided that the .C or .c extension is changed to .cc.

As the right versions of autoconf, automake, Curl, SSL, etc. were available in the Cygwin environment; we tried to use the _autosetup program to automate the porting and configuration for Windows. Unfortunately configure did not work because it could not find the SSL libraries and it insisted in using g++. We later found that this is a known bug in the Cygwin environment. When this bug is fixed _autosetup would allow us to build "real" libraries rather than copying the relevant source files thus saving a lot of time.

Although most of the code worked without change, a nasty problem was found in the hostinfo.h BOINC header file. There there was a reference to a Windows system type

```
extern HINSTANCE g_hIdleDetectionDll;
```

that produced syntax errors until we prefixed with

```
#include <Windows.h>
```

However, this produced a clash of the Windows system type PSIZE defined in

```
C:\Program Files\Microsoft Visual Studio.NET 2003\Vc7\
   PlatformSDK\Include\WinDef.h
```

With the Geant4 type defined in

```
$G4INSTALL/source/processes/hadronic/cross_sections/
 include/G4HadronCrossSections.hh
```

Fortunately we could remove the clash modifying WinDef.h as follows

```
#ifndef G4VERBOSE
typedef struct tagSIZE
{
    LONG        cx;
    LONG        cy;
} SIZE, *PSIZE, *LPSIZE;
#else
typedef struct tagSIZE
{
    LONG        cx;
    LONG        cy;
} SIZE, *LPSIZE;
#endif
```

We also had to add

```
LDFLAGS += Winmm.lib
```

to the GNUmakefile in order to link to the Windows system multimedia library to resolve the timeSetEvent() and timeKillEvent() calls used in boinc_api.cc. These calls are described in [115].

## D.4 BOINC templates

A work unit template and a result template have been created in the templates directory under the main BOINC server installation directory. The templates are plain text files containing XML tags. We created the following work unit template:

```
<file_info>
   <number>0</number>
</file_info>
<file_info>
   <number>1</number>
</file_info>
<workunit>
    <file_ref>
```

```
        <file_number>0</file_number>
        <open_name>run.g4</open_name>
    </file_ref>
    <file_ref>
        <file_number>1</file_number>
        <open_name>seed.txt</open_name>
    </file_ref>
</workunit>
```

This template gives to BOINC the list of files that your application needs to be executed. In particular, the files that change in each job execution have to be specified. Other files required by your binary that are the same for all job executions are better placed in the applications directory to prevent them to be downloaded each time.

In our work unit the run.g4 script is used. It gives details about the type and energy of the simulated particle and about the materials of the detector, among other things. A seed.txt file is also sent with the seed to be used for the simulation. Is important that all the job instances use the same seed because BOINC may send the same job to different machines and verify the results by comparing them, so the results are expected to be the same for the same job.

The result template used is the following:

```
<file_info>
    <name><OUTFILE_0/></name>
     <generated_locally/>
     <upload_when_present/>
    <max_nbytes>102400</max_nbytes>
    <url><UPLOAD_URL/></url>
</file_info>
<result>
    <file_ref>
        <file_name><OUTFILE_0/></file_name>
        <open_name>my_stdout.txt</open_name>
    </file_ref>
</result>
```

It specifies the name (and number) of the file we want to be sent by the client as result. It is important to add the tags generated_locally and upload_when_present.

Please note that BOINC will find the output file only if we "resolve" its name inside the application's code (as well as the input ones) so we sent with the job an empty output file to be filled.

APPENDIX

# E

# Glossary

Here we present a glossary including the acronyms and abbreviations used in this work. It includes the ones used in High Energy Physics as well as the ones used in the ATLAS experiment and in Grid Computing.

You can find a more extensive list of ATLAS acronyms at [118] and another list including most Grid acronyms at [119].

| | |
|---|---|
| ACL | *Access Control List* |
| AFS | *Andrew File System* |
| AII | *Automated Installation Infrastructure* |
| ALICE | *A Large Ion Collider Experiment* |
| API | *Application Programming Interface* |
| ARDA | *A Realisation of Distributed Analysis for LHC* |
| ATLAS | *A Toroidal LHC ApparatuS* |
| Atlfast | *ATLAS FAST simulation* |
| BalticGrid | *Baltic States Grid* |

| | |
|---|---|
| BDII | *Berkeley Database Information Index* |
| BOINC | *Berkeley Open Infrastructure for Network Computing* |
| BT | *Barrel Toroids* |
| CA | *Certification Authority* |
| CAF | *CERN Analysis Facility* |
| CB | *Central Barrel* |
| CDB | *Configuration DataBase* |
| CE | *Computing Element* |
| CERN | *European Organisation for Nuclear Research* |
| CIEMAT | *Centro de Investigaciones Energéticas, MedioAmbientales y Tecnológicas* |
| CMS | *Compact Muon Solenoid* |
| CMT | *Configuration Management Tool* |
| CNGS | *CERN Neutrinos to Gran Sasso* |
| CNM | *Centro Nacional de Microelectrónica* |
| CPU | *Central Processing Unit* |
| CRL | *Certificate Revocation List* |
| CS | *Central Solenoid* |
| CSC | *Cathode Strip Chambers* |
| CVS | *Concurrent Versions System* |
| DB | *Database* |
| DC | *Data Challenge* |

| | |
|---|---|
| DN | *Distinguish Name* |
| DDM | *Distributed Data Management* |
| DLI | *Data Location Interface* |
| DQ2 | *Don Quijote 2* |
| EB | *Extended Barrels* |
| ECT | *EndCap Toroid* |
| ED | *Extra Dimensions* |
| EDG | *European DataGrid* |
| EELA | *E-infrastructure shared between Europe and Latin America* |
| EDM | *Event Data Model* |
| EGEE | *Enabling Grids for E-SciencE* |
| EMEC | *Electromagnetic EndCap calorimeter* |
| EUMedGRID | *EU Mediterranean Grid* |
| FCAL | *Forward Calorimeter* |
| FiReMan | *File and Replica Catalogue* |
| FTP | *File Transfer Protocol* |
| FPS | *File Placement Service* |
| GIIS | *Grid Information Index Server* |
| gLite | *Lightweight Middleware for Grid Computing* |
| GRAM | *Grid Resource Allocation and Management* |
| GridFTP | *Grid Service for File Transfer* |
| GridPP | *The UK Grid for Particle Physics* |

| | |
|---|---|
| GRIS | *Grid Resource Information Service* |
| GSI | *Grid Security Infrastructure* |
| GUI | *Graphical User Interface* |
| GUID | *Global Unique IDentifiers* |
| GUT | *Grand Unified Theory* |
| HEC | *Hadronic EndCap calorimeter* |
| HEP | *High Energy Physics* |
| ID | *Inner Detector* |
| IFAE | *Institut de Física d'Altes Energíes* |
| IFCA | *Instituto de Física de Cantabria* |
| IFIC | *Instituto de Física Corpuscular* |
| II | *Information Index* |
| IS | *Information Service* |
| ISOLDE | *Isotope Separation OnLine DEvice* |
| ISR | *Intersecting Storage Rings collider* |
| iVDGL | *International Virtual Data Grid Laboratory* |
| JDL | *Job Description Language* |
| KK | *Kaluza-Klein* |
| LAr | *Liquid-Argon calorimeter* |
| LB | *Logging & Bookkeeping* |
| LCFG | *Local ConFiGuration system* |
| LCG | *LHC Computing Grid* |

| | |
|---|---|
| LDAP | *Lightweight Directory Access Protocol* |
| LEP | *Large Electron-Positron collider* |
| LFC | *LCG File Catalogue* |
| LFN | *Logical File Name* |
| LHC | *Large Hadron Collider* |
| LHCb | *Large Hadron Collider beauty experiment* |
| LRMS | *Local Resource Management System* |
| MC | *Monte Carlo* |
| MDS | *Metacomputing Directory Service* |
| MDT | *Monitored Drift Tube* |
| MSS | *Mass Storage System* |
| NCM | *Node Configuration Manager* |
| NorduGrid | *Nordic Data Grid Facility* |
| OO | *Object Oriented* |
| OSG | *Open Science Grid* |
| PBS | *Portable Batch System* |
| PIC | *Port d'Informació Científica* |
| PKG | *Solaris Package* |
| PM | *Photomultiplier* |
| PMTs | *Photomultiplier Tubes* |
| POOL | *POOL Of persistent Objects for LHC* |
| PRC | *Public Resource Computing* |

| | |
|---|---|
| PS | *Proton Synchrotron collider* |
| RB | *Resource Broker* |
| RC | *Replica Catalogue* |
| R-GMA | *Relational Grid Monitoring Architecture* |
| RLS | *Replica Location Service* |
| RM | *Replica Manager* |
| RMC | *Replica Metadata Catalogue* |
| RMS | *Replica Management Service* |
| ROD | *Read Out Drivers electronic board* |
| RPC | *Resistive Plate Chamber* |
| RPM | *RPM (Red Hat Package Manager)* |
| SCRAM | *Software Configuration, Release And Management* |
| SCT | *Semi-Conductor Tracker* |
| SE | *Storage Element* |
| SEAL | *Shared Environment for Applications at LHC* |
| SEE-Grid | *South Eastern European Grid* |
| SL | *Scientific Linux* |
| SLC3 | *Scientific Linux CERN release 3* |
| SM | *Standard Model* |
| SPEC | *Standard Performance Evaluation Corporation* |
| SPMA | *Software Package Management Agent* |
| SPS | *Super Proton Synchrotron collider* |

| | |
|---|---|
| SRM | *Storage Resource Manager* |
| SUSY | *Super-Symmetry* |
| SwRep | *Quattor Software Repository* |
| TDR | *Technical Design Report* |
| TGC | *Thin Gap Chamber* |
| TileCal | *Hadronic Tile Calorimeter* |
| TRT | *Transition Radiation Tracker* |
| UAM | *Universidad Autónoma de Madrid* |
| UB | *Universitat de Barcelona* |
| UI | *User Interface* |
| USC | *Universidad de Santiago de Compostela* |
| VDT | *Virtual Data Toolkit* |
| WISDOM | *Wide In Silico Docking On Malaria* |
| VO | *Virtual Organisation* |
| VOMS | *Virtual Organisation Membership Services* |
| WLS | *WaveLength Shifting* |
| WN | *Worker Node* |
| WM | *Workload Manager* |
| WMS | *Workload Management System* |
| WU | *Workunit* |
| WWW | *World Wide Web* |
| XML | *Extensible Markup Language* |

# Agradecimientos

Inevitablemente, en un trabajo que ha requerido varios años de esfuerzo y dedicación, se debe gratitud a mucha gente. Durante los años en que he trabajado para esta tesis para el IFIC mientras estaba en el CERN, he conocido a gran cantidad de gente que me ayudó y apoyó de la que tengo agradables recuerdos.

Primeramente debo agradecer a mis tutores Jose Salt y Eduardo Ros por toda su ayuda y apoyo con la tesis, tan importante al principio como al final. Su ayuda y tutela ha sido desde luego crucial, no solo desde el punto de vista de la investigación sino también desde el pusto de vista de la motivación, organización y logística, permitiendo además realizar esta tesis siendo apoyado por el IFIC mientras la mayor parte del trabajo se realizó teniendo una beca en el departamento de IT del CERN. La ayuda de Jose Salt en temas de Grid fue fundamental, asi como la de Eduardo Ros en toda la parte de física.

Es importante expresar mi agradecimiento también a Ignacio Reguero, del departamento de IT del CERN, quien fue mi supervisor allí y me ofreció una muy importante ayuda y tutela tanto profesional como personal durante toda mi estancia en Ginebra. Él colaboró muy activamente en la parte de computación, especialmente en todo lo referente a BOINC, a la instalación y adaptación a otras plataformas de LCG y al trabajo referente a Quattor. También fue muy importante la ayuda de Miguel Marquina, también del CERN, principalmente en temas de BOINC. Gracias él pude meterme de lleno en el tema, no solo trabajando sino impartiendo un curso en el CIEMAT de Madrid y varios seminarios en el CERN.

También debo agradecer mucho a Nuria Rius el haberme permitido comenzar el doctorado, finalizando con ella mi tesina en "Masas de neutrinos

y bariogénesis a la escala de Gran Unificación" y publicando un articulo y dando una charla sobre dicho tema, aunque por diversos motivos finalmente debiera de cambiar la línea de investigación.

No debo olvidar agradecer a Luis March y a Belen Salvachua, del IFIC, su ayuda y colaboración durante toda la tesis, en temas de análisis de física y en computación Grid, mientras hacían también sus respectivos doctorados. Los dos estuvieron siempre ahí en caso de cualquier pregunta o duda que pudiera surgir y me ayudaron incondicionalmente en todo.

Quiero agradecer también a todos los compañeros y compañeras del IFIC con quienes he pasado buenos momentos y han compartido su tiempo conmigo aquí. Especialmente a Loli, con quien compartí despacho, al igual que con Albert antes, y además a Natxo, Juanjo y Diego, con quienes he pasado mucho tiempo durante mi estancia en el IFIC.

También me gustara dar las gracias a mis compañeros del CERN durante mi estancia en IT. Especialmente a Michel Manent, Philippe Defert, Juan Manuel Guijarro y Hege Hannsbank, por todo el apoyo tanto laboral como personal durante mi estancia allí, y por los buenos ratos pasados.

Deseo agradecer también el apoyo incondicional que todos mis amigos y amigas me han dado durante estos años de doctorado y de estancia en el CERN, aunque sería imposible citar a toda la gente que conozco y que me ha estado animando y apoyando durante este tiempo. Deseo citar al menos a unas cuantas personas más que me apoyaron y me animaron especialmente: Agnieszka, Andrea, Cristina, Esteban, Geri, Herta, Jesús, Jonatan, Juste, Natalia, Nacho, Pablo, Paula, Robin, Tatiana, Vero y Yuan. Quiero dar las gracias en concreto a Elena quien estuvo conmigo y me apoyó enormemente durante la carrera y el doctorado, además de durante mi estancia en el CERN.

Para finalizar, quiero citar también especialmente a mi familia: mis padres, mi hermano y mi hermana, mi cuñado y mis sobrinos, por su confianza en m y su apoyo durante todo ese tiempo.

# BIBLIOGRAPHY


[1] CERN's official web site
http://www.cern.ch

[2] About CERN's name
http://public.web.cern.ch/Public/Content/Chapters/
AboutCERN/WhatIsCERN/CERNName/CERNName-en.html

[3] CERN's chronology web site
http://library.cern.ch/archives/chrono/chrono_2002_cern.php

[4] The Large Hadron Collider (LHC), official web site
http://lhc.web.cern.ch/lhc

[5] LHC Design Report
http://ab-div.web.cern.ch/ab-div/Publications/LHC-
DesignReport.html

[6] ATLAS collaboration public web site
http://atlas.ch
ATLAS experiment private web site
http://atlas.web.cern.ch/Atlas/index.html

[7] ATLAS Technical Proposal, CERN/LHCC/94-43, LHCC/P2
http://atlas.web.cern.ch/Atlas/TP/tp.html

[8] ATLAS Technical Design Reports web site
http://atlas.web.cern.ch/Atlas/internal/tdr.html

[9] ATLAS Computing Technical Design Report
ATLAS TDR–017, CERN-LHCC-2005-022
http://cern.ch/atlas-proj-computing-tdr/Html/Computing-TDR-4.htm
http://cern.ch/atlas-proj-computing-tdr/PDF/Computing-TDR-final-July04.pdf

[10] Instituto de Física Corpuscular (IFIC) research institute web site
http://ific.uv.es

[11] S.F. Novaes, Proceedings of 10th Jorge Andre Swieca Summer School: Particle and Fields, Sao Paulo, Brazil, 31 Jan - 12 Feb 1999. [arXiv:hep-ph/0001283]

[12] K. Hagiwara et al. (the Particle Data Group). *Review of particles physics*. Phys. Rev. D 66 (2002)
Official web site with the latest and most accurate values:
http://pdg.lbl.gov

[13] Th. Appelquist, A. Chodos, P. Freund, *Modern Kaluza-Klein Theories*, Addison-Wesley 1987

[14] L. Ibañez, *Physics beyond the Standard Model*, Academic Training Lecture For Postgraduate Students, 29th March 2004
http://agenda.cern.ch/fullAgenda.php?ida=a036621

[15] G. Nordström, Phys. Zeitsh. **13**, 1126 (1912); Ann. d. Physik **40**, 872 (1913); **42**, 533 (1913)

[16] G. Nordström, Phys. Zeitsh. **15**, 504 (1914)

[17] Th. Kaluza, Sitzungsber. Preuss. Akad. Wiss. Phys. Math. Klassse 996 (1921)

[18] O. Klein, Nature, **118**, 516 (1926)

[19] O.Klein, Z.F. Physik, **37**, 895 (1926)

[20] N. Arkani-Hamed, S. Dimopoulos and G.R. Dvali,
Phys. Rev. **D59** (1999) 086004

[21] D.A. Dicus, C.D. McMullen, S. Nandi, Phys. Rev. **D.65**, 076007 (2002)

[22] G. Rizzo, Phys. Rev. **D61** (2000) 055005

[23] G. Polesello and M. Prata, Eur. Phys. j. **C32S2** (2003) 55-67

[24] G. Azuelos and G. Polesello, Eur. Phys. j. **C39S2** (2005) 11-1

[25] CERN's Grid Café web site
http://www.gridcafe.org

[26] Ian Foster, *What is the Grid? A Three Point Checklist*
http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf

[27] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc. (1999)

[28] David Anderson, *Volunteer Computing*
http://boinc.berkeley.edu/volunteer.php

[29] The Globus Toolkit
http://www.globus.org/toolkit

[30] LHC Computing Grid Technical Design Report
http://lcg.web.cern.ch/LCG/tdr

[31] Wide In Silico Docking On Malaria (WISDOM) project web site
http://wisdom.eu-egee.fr

[32] Predictor@home project's web site
http://predictor.scripps.edu

[33] BOINC (Berkeley Open Infrastructure for Network Computing)
http://boinc.berkeley.edu

[34] BOINC wiki documentation
http://boinc-wiki.ath.cx

[35] The Human Proteome Folding project web site
http://www.grid.org/projects/hpf

[36] World Community Grid project web page
http://www.worldcommunitygrid.org

[37] CrossGrid project web site
http://www.crossgrid.org

[38] UK e-Science Programme web site
http://www.rcuk.ac.uk/escience

[39] UK Computing for Particle Physics project web site
http://www.gridpp.ac.uk

[40] Entropia's web site
http://www.entropia.com

[41] Seti at Home project web page
http://setiathome.ssl.berkeley.edu

[42] J.A. Lopez-Perez, M. Marquina, I. Reguero, C. Ungil *Computación Ciudadana: una expresión de democracia en Ciencia y Tecnología*, p.114-119, Anthropos, n.214, 2007

[43] BitTorrent project web site
http://www.bittorrent.com

[44] LHC Computing Grid web page
http://www.cern.ch/lcg

[45] Open Science Grid web site
http://www.opensciencegrid.org

[46] NorduGrid web site
http://www.nordugrid.org

[47] Virtual Data Toolkit (VDT) web site
http://vdt.cs.wisc.edu

[48] International Virtual Data Grid Laboratory (iVDGL) Project web site
http://www.ivdgl.org

[49] Enabling Grids for E-sciencE (EGEE) project web page
http://eu-egee.org

[50] Report of the LCG Project Leader to the POB, 20 June 2005,
http://lcg.web.cern.ch/LCG/peb/documents/pob_20jun05/
LCG_status_050620.doc

[51] EGEE II at the EGEE information sheet,
http://public.eu-egee.org/files/EGEE II final 1.pdf
EGEE II at the EGEE news,
http://public.eu-egee.org/news/fullstory.php?news_id=58

[52] e-Infrastructure shared between Europe and Latin America (EELA) project web site
http://www.eu-eela.org

[53] South Eastern European Grid (SEE-Grid) project web site
http://www.see-grid.org

[54] EUMedGRID Project web site
http://www.eumedgrid.org
http://www.grid.org.tr/eng/etkinlikler/eumedgrid

[55] BalticGrid web site
http://www.balticgrid.org

[56] EUChinaGRID Project web site
http://www.euchinagrid.org

[57] The European DataGrid (EDG) Project web page
http://www.cern.ch/eu-datagrid

[58] Quattor project web page
http://www.cern.ch/quattor

[59] Solaris Operating System Official home page
http://www.sun.com/software/solaris

[60] List of Web sites for BOINC statistics
http://boinc.berkeley.edu/stats_sites.php

[61] BOINC Synergy team web page
http://www.boincsynergy.com

[62] Christian Soettrup, *Developing Distributed Computing Solutions Combining Grid Computing and Public Computing* (M.Sc. Thesis)
https://twiki.cern.ch/twiki/pub/LHCAtHome/LinksAndDocs/ChristianSoettrupBOINCThesis.pdf
Software download web site: http://www.fatbat.dk/thesis

[63] Ch. Kenyon and G. Cheliotis, *Creating services with hard guarantees from Cycle-Harvesting systems*, IBM Research Zurich, October 30, 2002.

[64] Condor Project web site
http://www.cs.wisc.edu/condor

[65] Condor Manual
http://www.cs.wisc.edu/condor/manual/v6.7

[66] Lattice BOINC project web site
http://lattice.umiacs.umd.edu

[67] R. Hawkings, *The ATLAS inner detector and flavour tagging performance*. ATLAS scientific note, SN-ATLAS-2003-026.

[68] M. Vos et al., *The b-tagging performance of the complete ATLAS DC1 layout using $W_H$ events*, ATL-COM-INDET-2003-17.

[69] ATLAS Fast Simulation software (Atlfast) official web site
http://www.hep.ucl.ac.uk/atlas/atlfast

[70] S.González de la Hoz, L. March, E.Ros, *Search for hadronic decays of $Z_H$ and $W_H$ in the Little Higgs model*, ATL-PHYS-PUB-2006-003

[71] E. Laure, F. Hemmer, et al., *Middleware for the Next Generation Grid Infrastructure*, Computing in High Energy and Nuclear Physics (CHEP), Interlaken, Switzerland, September 2004.
Lightweight middleware for Grid Computing (gLite) web site
http://glite.org

[72] Relational Grid Monitoring Architecture Project web site
http://www.r-gma.org

[73] MySQL database web site
http://www.mysql.com

[74] phpMyAdmin Project web site
http://www.phpmyadmin.net

[75] Introduction to LHC at Home
http://www.cern.ch/athome

[76] LHC at Home Project web page
http://lhcathome.cern.ch

[77] CERN BOINC team, *BOINC activities at CERN*, CERN Courier Computing Newsletters, April 2006
http://cerncourier.com/articles/cnl/3/4/14/1

[78] The SixTrack simulation
http://athome.web.cern.ch/athome/LHCathome/whatis.html

[79] CERN BOINC team, *LHC@home takes centre stage*, CERN Bulletin No.13/2005, 28th March
http://bulletin.cern.ch/eng/articles.php?bullno=13/2005&base=art

[80] P.Defert, M.Degerholm, F.Grey, J.Klem, J.A.Lopez-Perez, E.Mcintosh, J.Pedersen, I.Reguero, F.Schmidt, B.Segal, C.Soettrup, *Public Resource Computing at CERN - LHC@home*, Computing in High Energy and Nuclear Physics (CHEP), Mumbai, India, February 2006

[81] J.Klem, J.A.Lopez-Perez, I.Reguero, *LHC@home: Links, Documentation and Publications*, CERN Twiki document.
https://uimon.cern.ch/twiki/bin/view/LHCAtHome/LinksAndDocs

[82] Official web site of the PYTHIA event generator
http://www.thep.lu.se/ torbjorn/Pythia.html

[83] J.A. Lopez-Perez, *Pythia port to BOINC*, CERN Twiki document.
https://uimon.cern.ch/twiki/bin/view/LHCAtHome/BOINCandPythia

[84] Official web site of the FORTRAN version of Atlfast
http://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/HIGGS/Atlfast.html

[85] J.A. Lopez-Perez, *ATLFast port to BOINC*, CERN Twiki document.
https://uimon.cern.ch/twiki/bin/view/LHCAtHome/BOINCandATLFast

[86] L. Oakes, *Porting Atlfast C++ to BOINC: Feasibility study and ground work*, CERN Twiki document.
https://uimon.cern.ch/twiki/bin/view/LHCAtHome/PortingAtlfastC

[87] Configuration Management Tool (CMT), official web site
http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/sit/Policy

[88] Pacman management tool, official web site
http://physics.bu.edu/pacman

[89] Geant4
http://www.cern.ch/geant4

[90] S. Agostinelli et al., *Geant4 - a simulation toolkit*, Nuclear Instruments and Methods in Physics Research A 506 (2003) 250-303
J. Allison et al., *Geant4 developments and applications*, IEEE Transactions on Nuclear Science 53 No. 1 (2006) 270-278

[91] P.Defert, J.Klem, J.A.Lopez-Perez, I.Reguero, A.Ribon, C.Soettrup, *Public Resource Computing and Geant4*, Geant4 User Conference, November 2005

[92] P.Defert, J.Klem, J.A.Lopez-Perez, I.Reguero, A.Ribon, C.Soettrup, *Public Resource Computing and Geant4*, Computing in High Energy and Nuclear Physics Conference (CHEP), February 2006

[93] J.A. Lopez-Perez, *BOINC and Geant4*, CERN set of Twiki documents.
https://uimon.cern.ch/twiki/bin/view/LHCAtHome/BOINCandGeant4

[94] Garfield simulation program, official web site
http://garfield.web.cern.ch/garfield

[95] J.A. Lopez-Perez, *Porting of Garfield to BOINC*, CERN Twiki documents.
https://uimon.cern.ch/twiki/bin/view/LHCAtHome/BOINCandGarfield

[96] Scientific Linux CERN official web site
http://linux.web.cern.ch/linux
CERN Linux Installation and Software Repository Service
http://linuxsoft.cern.ch

[97] Scientific Linux official web site
https://www.scientificlinux.org

[98] Red Hat Enterprise Linux official web site
http://www.redhat.com/rhel

[99] Official web site of the gnuLinEx Linux distribution (only in Spanish)
http://www.linex.org
Background information in English (from official site)
http://www.linex.org/linex2/linex/ingles/index_ing.html

[100] C. García Orellana, *Public Resource Computing projects at Extremadura*, Computing Seminar at CERN, Tuesday 12 September 2006
http://agenda.cern.ch/fullAgenda.php?ida=a063365&header=none

[101] LCG middleware installation instructions at CERN's Grid Deployment official web site
http://grid-deployment.web.cern.ch/grid-deployment/documentation/LCG2-Manual-Install

[102] J.A. Lopez-Perez, *Quattor software management system*, ELFms meeting, Geneva (CERN), July 2005
https://twiki.cern.ch/twiki/bin/viewfile/DESgroup/SolarisHowto?filename=quattor13.pdf

J.A. Lopez-Perez, M. Manent *How to install Solaris 9 at CERN*, CERN Twiki document.
https://twiki.cern.ch/twiki/bin/view/DESgroup/SolarisHowto

[103] J.A. Lopez-Perez, I. Reguero, *Porting LCG Applications*, Computing in High Energy and Nuclear Physics Conference (CHEP), Interlaken, Switzerland, September 2004

[104] LCG External Software
http://spi.cern.ch/extsoft

[105] LCG Applications
http://lcgapp.cern.ch/project

[106] JP. Wellisch, C. Williams , S. Ashby, Computing in High Energy and Nuclear Physics, (2003)

[107] SCRAM documentation web site
http://service-spi.web.cern.ch/service-spi/app/spi/scram/V0_20_0/doc/html/SCRAM.html

[108] SCRAM web site
http://spi.cern.ch/scram

[109] SEAL installation instructions
http://seal.web.cern.ch/seal/snapshot/devguide/howtorelease.html

[110] J.A. Lopez-Perez, *FORTRAN Applications in BOINC*, CERN Twiki document.
https://twiki.cern.ch/twiki/bin/view/LHCAtHome/FortranApplicationsInBoinc

[111] Cooperative Linux environment official web page
http://www.colinux.org

[112] Geant4 documentation,
http://geant4.web.cern.ch/geant4/G4UsersDocuments/UsersGuides/InstallationGuide/html/PCMachines/pcMachines.html

[113] Cygwin POSIX environment official web page
http://www.cygwin.com

[114] Geant4 installation
http://geant4.web.cern.ch/geant4/G4UsersDocuments/UsersGuides/InstallationGuide/html/UnixMachines/unixMachines.html

[115] Geant4, timeSetEvent function
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/multimed/htm/_win32_timesetevent.asp

[116] ROOT official web site
http://root.cern.ch

[117] LCFG official web site
http://www.lcfg.org

[118] ATLAS Acronym Glossary (AARG) web at ATLAS Twiki web site
http://manchester.web.cern.ch/manchester/twiki/bin/view/Atlas/AtlasAcRonymGlossary

[119] Grid Acronym Soup (GAS) web at GridPP web site
http://www.gridpp.ac.uk/gas