Alma Mater Studiorum · University of Bologna

School of Science
Department of Physics and Astronomy
Master Degree in Nuclear and Subnuclear Physics

Deployment of a Data Analysis Workflow of the ATLAS Experiment on HPC Systems

Supervisor:
Prof. Lorenzo Rinaldi

Co-supervisor:
Dr. Giuseppe Carratta

Submitted by:
Federico Andrea Guillaume Corchia

Academic Year 2021/2022

Abstract

LHC experiments produce an enormous amount of data, estimated of the order of a few PetaBytes per year. Data management takes place using the *Worldwide LHC Computing Grid* (WLCG) grid infrastructure, both for storage and processing operations. However, in recent years, many more resources are available on High Performance Computing (HPC) farms, which generally have many computing nodes with a high number of processors. Large collaborations are working to use these resources in the most efficient way, compatibly with the constraints imposed by computing models (data distributed on the Grid, authentication, software dependencies, etc.).

The aim of this thesis project is to develop a software framework that allows users to process a typical data analysis workflow of the ATLAS experiment on HPC systems. The developed analysis framework shall be deployed on the computing resources of the Open Physics Hub project and on the CINECA Marconi100 cluster, in view of the switch-on of the Leonardo supercomputer, foreseen in 2023.

Sommario

Gli esperimenti dell'LHC producono un'enorme quantità di dati, stimata dell'ordine di alcuni PetaByte all'anno. La gestione dei dati avviene mediante l'uso dell'infrastruttura grid *Worldwide LHC Computing Grid* (WLCG), sia per l'archiviazione dei dati sia per il loro processamento. Negli ultimi anni, molte più risorse sono disponibili su farm High Performance Computing (HPC), che generalmente prevedono molti nodi di calcolo con un grande numero di processori. Le grandi collaborazioni lavorano per utilizzare queste risorse nel modo più efficiente, compatibilmente con i vincoli imposti dai modelli computazionali (dati distribuiti sulla Grid, autenticazione, dipendenze dei software, ecc.).

L'obiettivo di questo progetto di tesi è sviluppare un framework software che permetta agli utenti di processare un tipico workflow di analisi dell'esperimento ATLAS su sistemi HPC. Il framework di analisi sviluppato sarà impiegato sulle risorse di calcolo del progetto Open Physics Hub e sul cluster Marconi100 del CINECA, in vista dell'attivazione del supercomputer Leonardo, prevista nel 2023.

Contents

Introduction

LHC experiments produce an enormous amount of data (a few petabytes per year), managed with the WLCG grid infrastructure for storage and processing. This latter resource is, however, turning out not to be sufficient for the needs of contemporary and future High Energy Physics experiments. Recently, further resources have become available on High Performance Computing (HPC) farms, featuring many compute nodes with a high number of processors, to which analysis work may be submitted. Large collaborations are working to employ these resources for their research in the most efficient way, compatibly with the constraints imposed by computing models (data distribution on the Grid, authentication, software dependencies, etc.).

The aim of this thesis project is to develop a software framework that allows users to process typical data analysis workflows of the ATLAS experiment on HPC systems. For this work, the ATLAS analysis workflow *SeeSawAnalysis* exploited to perform the already published Type-III SeeSaw searches ([1, 2, 3, 4]) is considered. The developed analysis framework shall be deployed on the computing resources of the Open Physics Hub project at the Department of Physics and Astronomy "Augusto Righi" of the University of Bologna and on the Marconi100 cluster at CINECA, also in view of the forthcoming switch-on of the Leonardo supercomputer also at CINECA.

Chapter 1 presents the Physics theory and the experimental context of this work, covering for the latter both the ATLAS experiment and its computing model. Chapter 2 provides an introduction to High Performance Computing theory and to why it turns out to be indeed important in High Energy Physics, while Chapter 3 features details on the clusters and software solutions used for this work. The concrete thesis work is exposed in detail in Chapter 4, with the description of the original analysis workflow and the building steps of the framework, and its results are finally discussed in Chapter 5, together with possibilities for this work and the ideas behind it to be further developed.

Chapter 1

Physics Analysis and Computing Context

1.1 Basics on the Standard Model

The Standard Model (SM) of Particle Physics is the foundation of modern High Energy Physics [5]. According to this model, all matter is built from a small number of fundamental spin 1/2 particles, called *fermions*, divided into six *leptons* and six *quarks*. Leptons carry integral electric charge: three of them are charged (of unit negative charge), being the *electron* (e), the *muon* (μ) and the *tauon* (τ), and three are neutral, the so-called *neutrinos*. There is a neutrino type, or "flavour", for each charged lepton: *electron neutrino* (ν_e), *muon neutrino* (ν_μ) and *tau neutrino* (ν_τ). Quarks carry fractional charges: $+\frac{2}{3}|e|$ or $-\frac{1}{3}|e|$, where e is the electron charge ($1.602176634 \cdot 10^{-19}$ C, [6]). There are six quark flavours: *up* (u), *down* (d), *charm* (c), *strange* (s), *top* (t) and *bottom* (b), which are grouped into three pairs. These pairs are defined as: (u,d), (c,s), (t,b), following the ($+\frac{2}{3}|e|$,$-\frac{1}{3}|e|$) syntax; protons are made of two up quarks and one down quark, while neutrons are made of one up quark and two down ones [7].

The Standard Model also includes the *interactions* between these particles. They are described in terms of the exchange of fundamental integer-spin particles, called *gauge bosons*, between fermions. There are four types of interactions in nature, of which the SM only covers the first three:

- the *strong* interaction, responsible for binding quarks in protons and neutrons, and the latter two in nuclei, mediated by *gluons*;
- the *weak* interaction, responsible for radioactive decays, mediated by the $W^\pm$ and Z^0 bosons;
- the *electromagnetic* interaction, mediated by the *photon*;

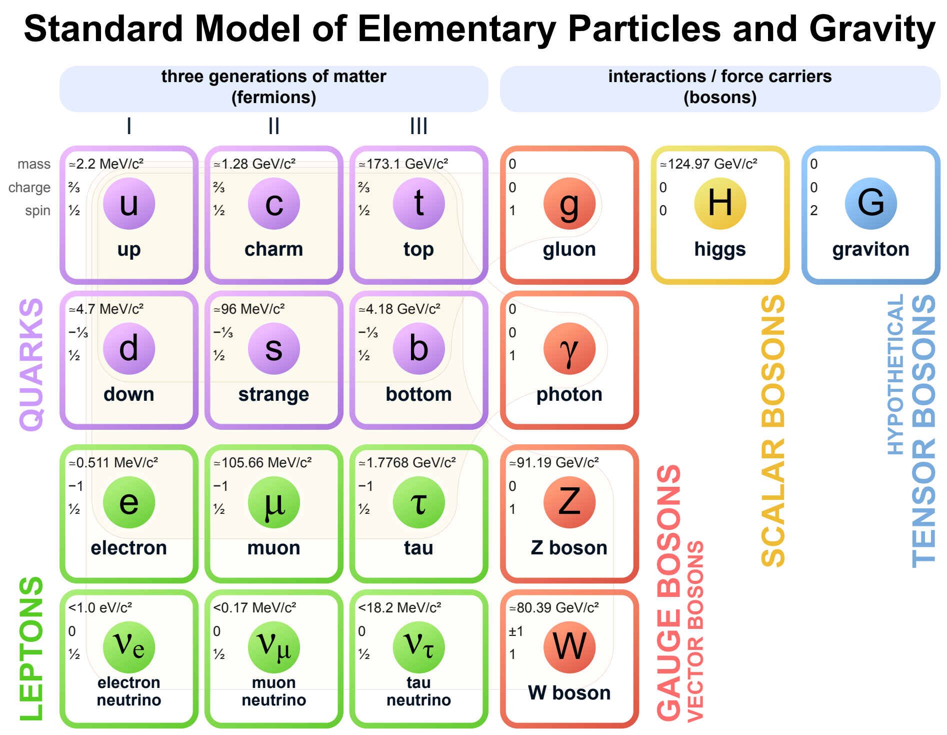

Standard Model of Elementary Particles and Gravity
three generations of matter (fermions)
interactions / force carriers (bosons)
I
II
III
mass
charge
spin
≈2.2 MeV/c² ⅔ ½ u up
≈1.28 GeV/c² ⅔ ½ c charm
≈173.1 GeV/c² ⅔ ½ t top
0 0 1 g gluon
≈124.97 GeV/c² 0 0 H higgs
0 0 2 G graviton
≈4.7 MeV/c² −⅓ ½ d down
≈96 MeV/c² −⅓ ½ s strange
≈4.18 GeV/c² −⅓ ½ b bottom
0 0 1 γ photon
≈0.511 MeV/c² −1 ½ e electron
≈105.66 MeV/c² −1 ½ μ muon
≈1.7768 GeV/c² −1 ½ τ tau
≈91.19 GeV/c² 0 1 Z Z boson
<1.0 eV/c² 0 ½ νe electron neutrino
<0.17 MeV/c² 0 ½ νμ muon neutrino
<18.2 MeV/c² 0 ½ ντ tau neutrino
≈80.39 GeV/c² ±1 1 W W boson
QUARKS
LEPTONS
GAUGE BOSONS
VECTOR BOSONS
SCALAR BOSONS
HYPOTHETICAL
TENSOR BOSONS

Figure 1.1: The Standard Model of Particle Physics, including the hypothetical mediator of gravitation [8].

- the *gravitational* interaction, thought to be mediated by the *graviton*.

Studies have been and are being conducted to look for unification of these interactions into a single one. This has already been done for the electromagnetic and the weak interaction (in the *electroweak* theory): at low energy, the symmetry is broken, making them appear as different interactions.

The Standard Model also includes a further field, the *Brout-Englert-Higgs field* (BEH), which gives mass to weak interaction mediators and to fermions. From this field one further spin-0 particle arises, the *Higgs boson* [9, 10, 11].

All SM particles are show in Fig. 1.1.

The SM is defined by a local gauge symmetry of the group:

$$SU(3)_C \otimes SU(2)_L \otimes U(1)_Y \tag{1.1}$$

where $SU(3)_C$ corresponds to the strong interaction, while $SU(2)_L \otimes U(1)_Y$ corresponds to the Glashow-Weinberg-Salam (GWS) theory, which describes the electroweak interaction. All of the above is formally expressed through the SM Lagrangian, which is the following:

$$\mathcal{L} = -\frac{1}{4}F_{\mu\nu}F^{\mu\nu} + i\bar{\psi}\not{D}\psi + h.c. + \psi_i y_{ij}\psi_j\phi + h.c. + |D_\mu\phi|^2 - V(\phi) \tag{1.2}$$

with the various terms describing different parts of the theory:

- $-\frac{1}{4}F_{\mu\nu}F^{\mu\nu}$: it describes gauge bosons and the interactions of each one with itself (for those who have a self-interaction, i.e. all gauge bosons but the photon);
- $i\bar{\psi}\not{D}\psi$: it describes how interaction particles interact with matter particles;
- *h.c.*: the *hermitian conjugate* of the previous term;
- $\psi_i y_{ij}\psi_j\phi$: it describes how matter particles couple to the BEH field ϕ and, therefore, get mass;
- *h.c.*: the *hermitian conjugate* of the previous term. This one in particular describes the same interaction of the term above, but with antimatter particles;
- $|D_\mu\phi|^2$: it describes how interaction particles couple to the BEH field;
- $-V(\phi)$: it describes the potential of the BEH field.

The SM has been successful in verifying most of its predictions, but for some like the neutrino mass it still has not (according to it, neutrinos should have no mass, while there is experimental evidence that they do). Moreover, gravity is not included in the model, so the SM is not the final theory of everything.

1.2 The SeeSaw Mechanism

1.2.1 Fundamentals of the SeeSaw Mechanism

The workflow chosen to run on the projected framework was developed to look for evidence of the Type-III SeeSaw mechanism. The SeeSaw mechanisms [12, 13] aim at explaining theoretically the small masses of neutrinos with respect to the other charged leptons, as experimentally verified and in contrast with SM predictions. There are three SeeSaw models, all of them with the exchange of new, non-SM heavy particles with mass values able to generate tiny neutrino masses:

- *Type-I SeeSaw*, employing at least two neutral fermions, being right-handed neutrinos with Yukawa couplings λ, to which neutrino masses are related;
- *Type-II SeeSaw*, employing a new $SU(2)_L$ scalar weak triplet ($\Delta^{\pm\pm}$, $\Delta^{\pm}$, Δ^0) and featuring lepton number violation;

- *Type-III SeeSaw*, employing at least an extra $SU(2)_L$ heavy fermionic triplet Σ, with mass eigenstates N^0 and $L^\pm$ coupling to charged and neutral SM leptons through mixing. Neutrino masses are generated through Yukawa couplings of the SM Higgs field, heavy leptons and SM neutrinos. Due to their gauge couplings, heavy leptons can be produced in pairs directly from gauge interactions and through plenty of production mechanisms at proton-proton colliders.

Neutrino masses may possibly also arise from combinations of the above, e.g. in *Left-Right Symmetric models*, containing both Type-I and Type-II, or in the case of the adjoint representation of $SU(5)$, containing both Type-I and Type-III.

The workflow focused in particular on the Type-III mechanism, described in the next section.

1.2.2 Type-III SeeSaw Mechanism

As said above, in this type of the SeeSaw mechanism the SM field content is only extended by the addition of fermions being triplets of SU(2) with zero hypercharge, which shall be denoted here with $\vec{\Sigma}$, where the vectorial character refers to its three SU(2) components, i.e. $\vec{\Sigma} = (\Sigma^1, \Sigma^2, \Sigma^3)$. Being $\vec{\Sigma}$ in the adjoint representation of the gauge group, its Majorana mass term is gauge invariant and interactions are described by the following Lagrangian (Eq. 1.3) [14]:

$$\mathcal{L}_\Sigma = i\vec{\Sigma}_R \not{D} \vec{\Sigma}_R - \left[\frac{1}{2}\vec{\Sigma}_R M_\Sigma \vec{\Sigma}^c_R + \vec{\Sigma}_R Y_\Sigma \left(\tilde{\phi}^\dagger \vec{\tau} l_L\right) + h.c.\right] \tag{1.3}$$

Here the three $SU(2)_L$ components of the field $\vec{\Sigma}$ have identical Majorana mass terms and are not eigenstates of the electric charge, given instead by the combinations:

$$L^\pm \equiv \frac{\Sigma^1 \mp i\Sigma^2}{\sqrt{2}} \tag{1.4}$$

$$N^0 \equiv \Sigma^3 \tag{1.5}$$

The basis used hereafter is one in which M_Σ is a diagonal matrix in generation space. After electroweak symmetry breaking, the Yukawa coupling Y_Σ induces Majorana neutrino masses for the left-handed neutrino fields of the SM through the exchange of $\vec{\Sigma}$ particles [14].

Solving the equations of motion, at low energies the light neutrino mass is the following:

$$m_\nu = -\frac{v^2}{2} Y_\Sigma^T \frac{1}{M_\Sigma} Y_\Sigma \tag{1.6}$$

where v is the vacuum expectation value of the Higgs field.

The Type-III SeeSaw heavy leptons $L^{\pm}$ and N^0 can be produced in *pp* collisions and have the following possible decay channels:

$$L^{\pm} \to W^{\pm}\nu$$

$$L^{\pm} \to Z^0 l^{\pm}$$

$$L^{\pm} \to H l^{\pm}$$

$$N^0 \to W^{\pm} l^{\mp}$$

$$N^0 \to Z^0 \nu$$

$$N^0 \to H \nu$$

The work for which SeeSawAnalysis has been created, being the one at [1, 2, 3, 4], is focused on the search for the Type-III SeeSaw heavy leptons *in leptonic final states* (Fig. 1.2), using the full ATLAS Run 2 dataset, combining the results of the topologies with two, three and four leptons. Among all production mechanisms, in Fig. 1.2 the following are reported:

- in the three-lepton channel, two opposite sign charge heavy leptons are produced through a virtual Z boson, with one of them decaying into a lepton and a real Z^0 boson producing a lepton pair, and the other one decaying into a neutrino and a real $W^{\pm}$ boson producing a quark-antiquark pair;
- in the four-lepton channel, a virtual $W^{\pm}$ boson produces neutral and charged heavy leptons, with N^0 decaying into a lepton and a real W boson producing a quark-antiquark pair, and $L^{\pm}$ producing a lepton and an opposite sign lepton pair through a Z^0 boson [3, 15].

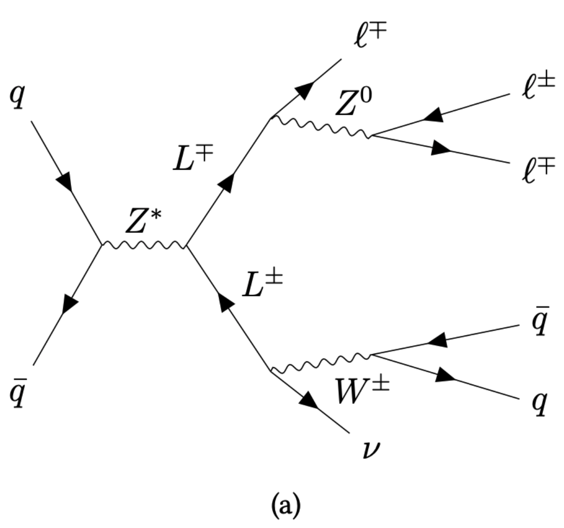
The $L^{\pm}$ and N^0 total width and their decay branching ratios (BR) into SM leptons depend on the leptons mixing matrix elements V_i ($i = e, \mu, \tau$). The latter only affect heavy leptons decays, while production is independent because of coupling with electroweak bosons. The branching ratio is directly proportional to the mixing angles values according to Eq. 1.7:

$$BR \propto \frac{|V_e|^2}{|V_e|^2 + |V_\mu|^2 + |V_\tau|^2} \tag{1.7}$$

The analysis performed in [1, 2, 3, 4] assumed the specific BR for the three lepton flavours have the same value, equal to 1/3. This is called the *flavour-democratic scenario*. The following values are provided by theory and independent experimental measurements:

$$|V_e| < 5.5 \cdot 10^{-2}$$

$$|V_\mu| < 6.3 \cdot 10^{-2}$$



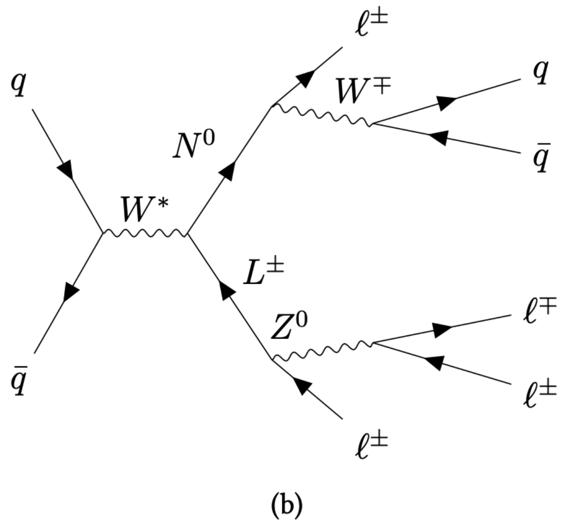


Figure 1.2: Feynman diagrams for the considered Type-III SeeSaw heavy leptons pair production ((a) three-lepton final state, (b) four-lepton final state) [3].

$$|V_\tau| < 6.3 \cdot 10^{-2}$$
$$|V_e V_\mu| < 1.7 \cdot 10^{-7}$$
$$|V_e V_\tau| < 4.2 \cdot 10^{-4}$$
$$|V_\mu V_\tau| < 4.9 \cdot 10^{-4}$$

All of the above implies constraints on heavy leptons production and decay modes. The assumption of a flavour-democratic scenario leads to the possibility to get small masses for the three SM neutrinos [3].

1.3 The LHC and the ATLAS Experiment at CERN

1.3.1 The Large Hadron Collider

The *Large Hadron Collider* (LHC) is a proton-proton collider with a 27 km circumference in use at the CERN (*Conseil Européen pour la Recherche Nucléaire*, European Organisation for Nuclear Research) laboratory near Geneva, Switzerland, designed to provide collisions with a maximum centre of mass energy of 14 TeV and to give answers to several SM open issues. The LHC performed a huge amount of precise measurements and put constraints on a wide area of the SM and beyond the SM parameters. The most important event in LHC history was the discovery of the Higgs boson in 2012 [9, 10].

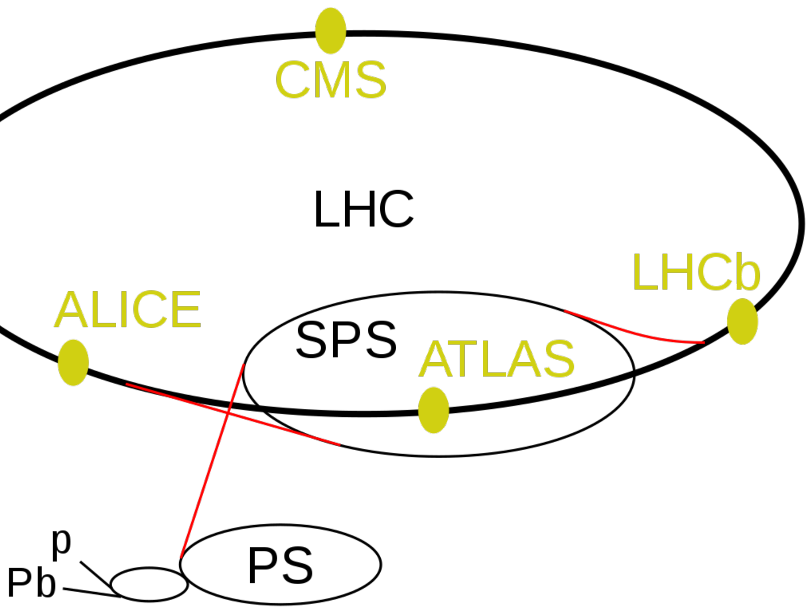


Figure 1.3: Schematic picture of the Large Hadron Collider with its four main experiments. Before reaching the LHC, beams are accelerated in a chain of linear and circular accelerators, with the Proton Synchrotron (PS) and the Super Proton Synchrotron (SPS) as the last stages [16].

The LHC is located in a tunnel about 100 m underground. After being accelerated in a chain of accelerators, composed of both linear and smaller circular accelerators, beams circulate in the ring inside vacuum pipes, guided by superconducting magnets. Particle bunches are made collide at the location of the four main experiments (Fig. 1.3):

- *A Toroidal LHC ApparatuS* (ATLAS): one of the two large general multi-purpose experiments [17];
- the *Compact Muon Solenoid* (CMS): the other general multi-purpose experiment [18];
- LHCb: designed to study the b-quark sector, CP violation and rare decays [19];
- *A Large Ion Collider Experiment* (ALICE): a heavy ion experiment, studying the nature of the *quark-gluon plasma* [20].

1.3.2 The ATLAS Detector

ATLAS (*A Toroidal LHC ApparatuS*) is a general multi-purpose detector in use at the Large Hadron Collider (LHC) since 2009, probing proton-proton and heavy ion-heavy ion collisions [17]. It has been built, as seen above, to confirm the theory of the Standard Model (e.g. the existence of the Higgs boson) and seek for signatures of New Physics.

ATLAS collects events coming from proton-proton collisions at 13 TeV of centre-of-mass energy and about $10^{34} cm^{-2} s^{-1}$ of luminosity, with bunches of up to 10^{11} protons colliding 40 million times per second. It also collects events from heavy ion collisions (Pb-Pb) at about 5 TeV of centre-of-mass energy. In order to achieve all these features in the best way, the detector must have:

- excellent measurement resolution in its parts;
- fast radiation hard electronics and sensor elements, because of the hard experimental conditions at the LHC;
- high granularity, to handle particle fluxes and lower the influence of overlapping events;
- large acceptance in pseudorapidity;
- highly efficient trigger for low transverse-momentum objects with sufficient background rejection, to have an acceptable trigger rate for most physics processes of interest [17].

Fig. 1.4 shows the ATLAS components. From the centre outwards it is made of:

- the *inner detector*: it is immersed in a 2 T solenoidal magnetic field produced by superconductors and features high resolution semiconductor pixel and strip detectors in the inner part of the tracking volume and straw-tube tracking detectors in the outer part, in order to have good charged particle momentum resolution and reconstruction. It also features vertex detectors close to the interaction region, to observe secondary vertices and for offline tagging of τ-leptons and b-jets;
- *electromagnetic calorimeter*: of liquid argon (LAr) type, for electron and photon identification and measurements of their energies;
- full-coverage *hadronic calorimeters*: of both LAr and scintillator-tile type, for accurate measurements of hadronic particles energy;
- the *muon spectrometer*: it provides good muon identification and momentum resolution; it is also able to determine unambiguously the charge of high transverse momentum muons. It is an air-core toroid system, with a long barrel and two inserted end-cap magnets and three layers of high precision tracking chambers.

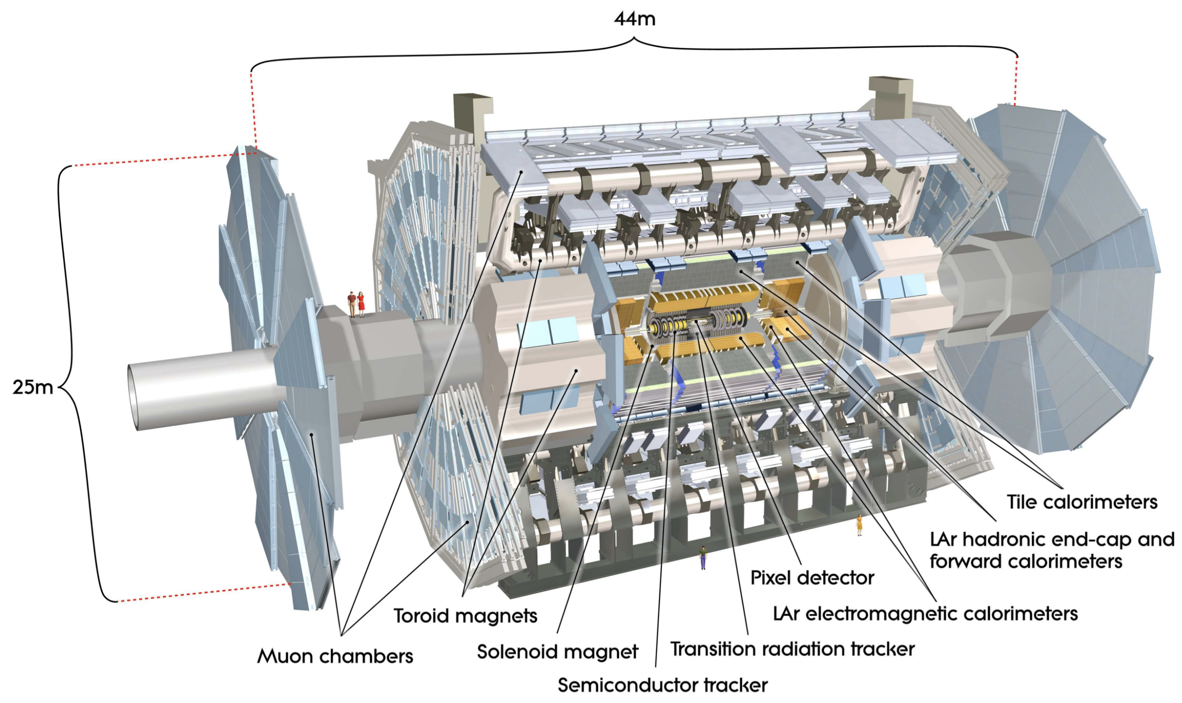

44m
25m
Tile calorimeters
LAr hadronic end-cap and forward calorimeters
Pixel detector
LAr electromagnetic calorimeters
Toroid magnets
Muon chambers
Solenoid magnet
Transition radiation tracker
Semiconductor tracker

Figure 1.4: Cut-away view of the ATLAS detector. It weighs 7000 tonnes [17].

Collected data are passed to a highly efficient multi-level trigger system, which must deal with a very high rate of incoming events, the proton-proton interaction rate at design luminosity of $10^{34}cm^{-2}s^{-1}$ being approximately 40 MHz. Nevertheless, events can only be recorded at about 100 Hz, so a large reduction is required. The Level 1 (L1) trigger system uses a subset of total detector information to select potentially interesting events reducing the data rate to about 100 kHz, followed by the *High Level Trigger* (HLT). Accepted events are finally stored at Tiers 0 and 1 of the Worldwide LHC Computing Grid (see next section).

1.3.3 The ATLAS Computing Model and the Worldwide LHC Computing Grid (WLCG)

The ATLAS Computing Model has been developed and made evolve to cope with the challenging requests of the experiment. The following refers to the computing model during LHC Run 2, from which experimental data used for this thesis work come [21].

LHC Run 2 at ATLAS introduced, because of the higher energy and luminosity, a doubling of the trigger rate, a further complexity of the events and major detector improvements. This had to be handled properly, considering that computing resources would not increase considerably (a 20% growth of computing power per year was expected).

Data are stored and replicated on a Grid system. Grid computing [22] may be defined as a pervasive computing resources supply; with this technology, users get access to computing resources (e.g. processors, storage, data and applications) according to their needs. An important point is that users do not need to know the actual location or the advanced technical details of such resources. Usually, a comparison is made with power grids, where users connect their appliances to electricity through a socket and electricity is used *de facto* ignoring where or how it is actually generated. Only the fact that it is actually available and it is satisfactorily (i.e. with a given voltage and without interruption) matters. Grid computing is based on the underlying distributed computing infrastructure technologies and on virtualisation techniques. For the latter, instead of dealing with physical resource devices, grid users interact with virtual devices: depending on the grid system, it may be either just the virtualisation of CPUs or machines (i.e. the creation of virtual machines) or the virtualisation of higher level components, like homogeneous resources, networks, applications and even more. In any case, this is only achievable through the use of open standards, ensuring that applications may transparently harness the potential of any appropriate resource given to them. Grid computing may also be defined, therefore, as distributed computing across virtualised resources: users get to interact with a large and powerful virtual computer made up of a combination of various resources. This technique provides plenty of advantages, such as the possibility to exploit under-used resources, harness the potential of parallel CPU capacity, access additional resources not available locally and even do the latter only when actually needed, all this in a reliable and cost-effective way.

In particular, the Grid system used by the ATLAS collaboration is the *Worldwide LHC Computing Grid* (WLCG), a global collaboration of about 170 computing centres in more than 40 countries linking national and international Grid infrastructures. The WLCG provides global computing resources for storage, distribution and analysis of the LHC. Users make job requests from one of the multiple entry points of the system, the computing Grid checks the identity of the user and looks for available sites able to provide requested resources. After being accepted, jobs entail the processing of the requested set of data, which are analysed using software provided by the experiments. All this is done without the user having to know the actual origin of the computing resources, just that this service makes such resources available to them. These resources turn out to be particularly useful, since the accelerator produces a great amount of data (about 200 PB per year) and this is only expected to increase in the coming years, with the next runs of the LHC [21, 23].

From a more technical side, the WLCG has a hierarchical structure of four layers, called *tiers*, each tier providing a specific set of services:

- Tier 0: the CERN Data Centre in Geneva, Switzerland. All data pass through it, but it only provides about 20% of the total compute capacity. It is responsible for safe-keeping of raw data, first reconstruction, distribution of raw data and reconstruction output to Tier 1s and reprocessing of data when the LHC is off;

- Tier 1: thirteen large computing centres around the world. One Tier 1 cluster is at INFN-CNAF in Bologna. They are responsible for safe-keeping of a share of raw and reconstructed data, large scale reprocessing and safe-keeping of its output, data distribution to Tier 2s and safe-keeping of a share of simulated data produced at Tier 2s;

- Tier 2: about 160 university and other scientific institutions devices around the world, handling analysis requirements and simulated event production and reconstruction;

- Tier 3: local computing resources, either local clusters in universities or individual PCs. These are the devices accessing the WLCG but without a formal engagement with it (apart from the possession of a user account).

From the point of view of data management, ATLAS uses Rucio (see Section 3.7.1) as distributed data management system. Rucio can manage the transfer of more than one million files per day (i.e. 4 PB of data per week) and delete more than ten million files per day; it also implements different protocols for data transfer and data access, including HTTP and XRootD ([24]), improved handling of metadata and advanced features for ownership, permissions and quotas. Jobs can thus run on sites with free CPUs other than the one on which the data they use are physically located, which can be accessed by said jobs remotely through the storage federation. Files are replicated on multiple systems with a given lifetime, established by a Data Lifecycle management model, which is in place to permit an intelligent data handling and also considers the popularity of the data (i.e. how many times they are needed).

The workload is handled by a Workload Management system. The ATLAS production system, Prodsys-2 [25], is made of four major components:

- a *Request Interface*, allowing production managers to define and handle production requests;

- DEfT (Database Engine for Task), translating production requests into tasks;

- JEDI (Job Execution and Definition Interface), creating jobs from task definitions;

- PanDA, executing jobs in the distributed environment.

Resource use has been improved, either for detector simulation (which takes up to 40% of CPU resources) and event reconstruction (a very memory intensive process). The

ATLAS analysis system is based on a file format (xAOD) directly turnable into ROOT Ntuples.

A fundamental point is that ATLAS heavily uses beyond-pledge resources, especially for Monte Carlo simulation. It adds as further resources *cloud-based* ones, *High Performance Computers* and even *Volunteer Computing* (the ATLAS@home project, born as an outreach activity, letting customer PCs run simulations when idle). Of particular interest for this work is *High Performance Computing* (HPC): supercomputers turn out to be very useful for adding further powerful resources and ATLAS has already succeeded in getting resource allocation in many centres in the world, even commercial ones (e.g. Amazon services). Their use poses some issues to be solved: while the fact that HPC has been designed for massively parallel applications (High Energy Physics is instead an embarrassingly parallel use case, therefore not the best to fit their architecture) in practice is no relevant issue, the collaboration had instead to work out a way through the heterogeneity of the centres in terms of processor architecture (x86-compatible or not), access policies and provisioning schemes (limits on the number of granted jobs). A focus on HPC shall be provided in Chapter 2. For the ATLAS collaboration to fulfil its needs, further resources would be needed: this is the issue addressed by this thesis work.

Chapter 2

High Performance Computing (HPC) Systems

2.1 Parallel Computing: High Throughput Computing (HTC) and High Performance Computing (HPC)

Parallel computing is a form of computation where multiple calculations are performed simultaneously. In order to analyse a process in a parallel way, it must be divided into smaller and independent sub-processes. If, for any reason, one task depends on another one, they are said to be concurrent. The logic is to adopt pure parallelism as long as it is possible, switching to concurrent tasks if this is not possible. A parallel workflow must be designed properly: if multiple tasks act redundantly there is just a waste of resources. Tasks should be handled such that work distribution among tasks and resources is the most efficient [26].

Several types of architectures exist for devices dealing with parallel computing: one single node with one single processor, one single node with multiple processors, multiple nodes with one single processor and multiple nodes with multiple processors. The choice of the architecture depends on the problem to analyse: some approaches are better than others. When designing a parallel workflow, it must also be considered whether the single operations making it up, if applied to a part of the dataset, do not interfere with other parts of the dataset (i.e. they are *independent*) or depend instead on other sections of the dataset or another user's operation (i.e. *dependent*). Computer clusters (see next section) provide great advantage for independent operations, while dependent ones run slower because communication between them is time consuming when compared to the time that might be spent doing independent operations. For the latter to be performed faster, a coordination and communication mechanism must be set up: this is done by interfaces such as *OpenMP API* and *MPI* [26].

However, there is an upper limit on how much a problem can be solved more efficiently: there are always portions of the problem that cannot be solved in a parallel way. This upper limit is given by Amdahl's law, which states that, when enhancing a fraction f of a computation by a speedup S, the overall speedup $S_{overall}$ is given by [26, 27]:

$$S_{overall}(f, S) = \frac{1}{(1-f) + \frac{f}{S}} \tag{2.1}$$

Depending on how resources are handled, the paradigms of High Throughput Computing and High Performance Computing are introduced:

- *High Throughput Computing* (HTC): is the computational paradigm of having a (huge) amount of substantially uncoupled computational tasks that can be distributed and run on independent CPUs. This is the case when multiple copies of a program are made run, each copy with a variation in the input variables: the copies run independently, since they can also do on their own, but at the same time, so results over the range of interest of the input variables are returned must faster than on a sequential, i.e. one after the other, execution;

- *High Performance Computing* (HPC): is the computation paradigm of handling tightly coupled calculations with demanding requirements, i.e. having a (huge) amount of computational tasks depending on each other that can be be distributed and run on various CPUs. This is the case of very complex analysis tasks which are divided into smaller ones according to a logical criterion, which continuously exchange information because each task also depends on the work conducted by the other ones.

In particular, the latter has relevant requirements in e.g. processing power, memory size and storage: for them to be satisfied, the most powerful computers in the world find application. The next section shall delve in detail into this aspect; in the following it shall also be demonstrated how HPC techniques also find useful application to HTC computing tasks [28].

2.2 General Ideas on HPC and Use Cases

High Performance Computing (HPC) may also be defined as the field of study of the most powerful computers in the world (supercomputers), both from a scientific and a technical point of view. It encompasses several computing resources, such as the dedicated software tools, programming languages and parallel programming [29].

The 500 most powerful supercomputers are placed under the "Top500" ranking, regularly updated and available on the Internet (at [30]). This ranking changes quickly: in

Rank	Name	Manufacturer	Site	Country	Cores (millions)	Peak performance (PFlops)	Power consumption (MW)	Previous Ranking (Nov 2021)
1	Frontier	HPE	DOE/SC/Oak Ridge National Laboratory	U.S.	8.73	1686	21.1	-
2	Supercomputer Fugaku	Fujitsu	RIKEN Center for Computational Science	Japan	7.63	537	29.9	1
3	LUMI	HPE	EuroHPC/CSC	Finland	1.11	214	2.94	-
6	Sunway TaihuLight	NRCPC	National Supercomputing Center in Wuxi	China	10.6	125	15.4	4
12	HPC5	DELL EMC	Eni S.p.A.	Italy	0.670	51.7	2.25	9

Table 2.1: Relevant entries in the latest (June 2022) TOP500 ranking. Here are shown the first three entries and the most powerful devices installed in the United States, Japan, China, the European Union and Italy [30].

2014, the world's most powerful computer was Tianhe-2 at the National Super Computer Center in Guangzhou (China) [30], while in November 2021 it was Fugaku at RIKEN Center for Computational Science in Kobe (Japan) and in the latest ranking (June 2022), with the latter falling at the second position, it is Frontier at DOE/SC/Oak Ridge National Laboratory in the United States [30]. A fast pace of progress may be seen: the first one had 3.12 million cores and a peak performance of 54.9 PFlops (one FLOPs being one Floating-Point Operation Per second), while the second reaches respectively 7.63 million cores and 537 PFlops of peak performance and Frontier tops at 8.73 million cores and 1686 PFlops of peak performance. Supercomputer design, in addition to processing power, must also take into account aspects of storage and connection, like overall memory size and network bandwidth of the interconnection network. These machines are power-consuming: the Chinese supercomputer required 17.8 MW of electrical power, while the Japanese one needs 29.9 MW and the American one 21.1 MW. This implies that design and usage of these devices must also consider economic cost and eco-friendliness [29]. Tab. 2.1 shows notable devices in June 2022 TOP500 ranking.

HPC is of the utmost importance for its speed, subsequent precision and ability to solve resource-demanding problems. Complex phenomena modelling, like weather forecasting and simulations with a finer mesh and/or bigger data sets, get relevant progress with HPC techniques. HPC is also useful for energy saving (at a constant performance, lower-profile and so less consuming processors are preferred) and for those algorithms that are intrinsically parallel, like image and video processing applying a filter on each pixel in a parallel way. GPUs (*Graphics Processing Unit*) turn out very useful, not only for high-demanding image processing, but also for general computing (this latter approach being known as the GPGPU, *General Purpose Graphics Processing Unit* paradigm) [29].

Below are some use cases of HPC techniques:

- Models for simulations:
  - Wind tunnel: a wind tunnel may get too difficult to build;
  - Crash tests: an airplane crash test is undeniably too expensive;
  - Climate or galaxy evolution: such simulations are so demanding that common computers are too slow;
  - Tests on nuclear weapons, drugs, pollution, epidemics: phenomena that must obviously be simulated because of their intrinsic danger;
- Fast, delay-free obtention of results:
  - Weather forecasts: weather data are usually only relevant if they can be used to make forecasts for the future. Data are only meaningful for a little time, so algorithms, said to be *on-line algorithms*, must deliver results fast;
  - Stock market: being the first in getting a result with high-frequency trading algorithms turns into an advantage in being able to take decisions faster;
- Big Data processing:
  - Genome analysis: DNA analysis implies dealing with huge datasets.

The last point mentions *Big Data*. This expression designates the technologies for massive datasets processing and large-scale data processing. This is usually described by using the *four Vs*, which are the characteristics under which a data analysis situation falls under the Big Data category:

- *Volume*: large datasets;
- *Variety*: data of various types;
- *Velocity*: data continuously collected, e.g. from sensors;
- *Value*: value of the obtainable insight.

These aspects may be employed in many fields, since many situations imply the four Vs, and this explains why this expression is widely used nowadays. In the following section application of all these ideas to Physics will be shown [29, 30].

2.3 HTC and HPC Application in High Energy Physics

Experiments like the ones at the LHC are an important use case of high performance devices. Particles collide in detectors approximately one billion times per second, which would generate collision data of the order of the petabyte each second. Such amounts of data would be impossible to handle and record for current computing systems, so they must be filtered to only save interesting ones. This filtering operation is done by each experiment individually, using dedicated trigger systems. Filtered data are then aggregated into the CERN Data Centre, where a copy is produced and archived for long-term storage on special tapes; afterwards, data reconstruction is performed. Still, even after such a reduction in data, the CERN Data Centre processes 1 PB of data a day on average [23].

"Stress" on computing resources is expected to increase: the four main LHC experiments (ATLAS, CMS, ALICE, LHCb) produce data for a long time (in 2016, during Run 2, data taking lasted 7.5 million seconds, 50% more than the original prediction of 5 million seconds); luminosity is also increasing in newer experiments, so events are more complex and require more sophisticated reconstruction and analysis techniques. Computing and storage resources are used at a very high level, continuously solving issues regarding data acquisition, data rates and data volumes.

A first solution to deal with increasing requirements is to upgrade the local computing infrastructure at CERN; another one is to use remote extensions of the data centre, like the one at the Wigner Research Centre for Physics (RCP) in Hungary, connected with three very high speed fibre optic links to Geneva.

This is, however, not enough: in the end, only 20% of CERN data are processed at CERN own data centre. This is where HPC techniques can intervene, overcoming all these limits.

The ATLAS experiment is actively pursuing collaboration with HPC enterprises to make use of HPC resources. The latest supercomputer joined to ATLAS processing activities is Vega, the first new petascale machine from the EuroHPC project, in Maribor, Slovenia. Vega contains on its own 122 800 physical cores or 245 760 logical cores, while the entire Grid network provides ATLAS with around 300 000. This successful experience shows that the ATLAS collaboration can obtain further resources by harnessing the potential of HPC clusters, a strategy with plenty of room for upscaling [31].

A very important final consideration is that, technically, High Energy Physics (HEP) tasks are actually *mainly of HTC*. The typical use case is having running in a parallel way an enormous number of copies of a workflow, each copy with a variation in the values of the input variables, to perform a scan over the range of interest: there is usually no interaction between different copies, so most HEP computing tasks fall under the HTC category, not HPC. However, out of the HEP research world, the other fields mainly need the submission of tasks falling instead under HPC (e.g., for fluid dynamics studies or neutral network training), so research in high performance devices mainly focuses on HPC-optimised techniques and devices made available are HPC-optimised. HTC-designed jobs may be run on HPC clusters without any issue, but with the understanding that performance does not improve as much as it would if the cluster had been designed with an optimisation for HTC; the performance increase is, nevertheless, still remarkable, fully justifying its use, especially in a context where Grid resources turn out not to be sufficient for the needs of HEP research. When designing a workflow for HPC clusters, physicists must therefore also think of how to better use HPC resources to increase performance of HTC-designed programs [32].

Chapter 3

Computing Setup

3.1 Introduction on Computing Clusters

In HPC a relevant role goes to clusters. A *computer cluster* is a set of connected computers working together, which can be seen, for many aspects, as a single system. Connection between computers can be loose, i.e. they communicate with each other but work and behave independently, or tight if this is not the case. On most modern clusters, loose connection is used. Benefits and issues of using a cluster instead of a single computer strictly depend on the program being run, its dataset size and operation types [26].

Computers making up a cluster are called *nodes*. Usually three kinds of them are defined:

- *entry* or *login* nodes: the ones to which users connect in order to use the cluster;
- *worker* nodes: the ones running programs in the cluster environment;
- *storage* nodes: the ones dedicated to long-term data storage.

This configuration and number of nodes of each type may vary depending on the individual cluster.

For this work, three clusters have been used: *MARCONI100* at CINECA [33], *Open Physics Hub* at the Department of Physics and Astronomy "Augusto Righi" of the University of Bologna [34] and *LXPLUS* at CERN [35]. These systems are presented later in this chapter.

3.2 Cluster Software

For the software design, the following aspects must be taken into account when working on HPC clusters:

- libraries: nowadays, there are plenty of libraries for all purposes. However, they do differ for both implementation style and approach, parallelisation support and implementation, recentness etc., aspects that must be studied thoroughly;
- compilers: they differ for performance of the produced executable; moreover, the usage of the right flags may make a relevant difference;
- compilation process: it becomes difficult to handle by hand when a program is made of many components with many dependencies. Solutions like *GNU Make* [36] provide automation for this task;
- performance: to be studied on all components. Many solutions exist, the *GNU time* [37] utility already being enough for most cases.

Also fundamental in the HPC context are good common programming practices like version control and documentation.

3.2.1 Software Distribution with CernVM File System (CVMFS)

The *CernVM File System* (CernVM-FS or CVMFS) is a read-only file system delivering scientific software into virtual machines and physical worker nodes in a fast, scalable and reliable way. Requested files and file metadata are downloaded on demand via a standard HTTP protocol and cached. Data authenticity and integrity are guaranteed; the CVMFS software includes both a client version to mount CVMFS repositories and a server version to create distributable CVMFS repositories [38].

CVMFS can be intended as a virtual file system, only loading data when requested and with software releases hosted on a CVMFS repository on a web server. Offline working is also possible, as long as all files required for software run are cached [38].

In the context of this thesis, CVMFS has been used to have important software ready for use without the need to install them on an own installation (e.g. ROOT [39]).

3.2.2 Conda

Conda is an open source package and environment management system, able to quickly install, run and update packages and their dependencies, and let users switch between environments on their devices. Available for Windows, macOS and Linux, it can package and distribute software for any language. It is available in two versions, Anaconda and Miniconda, while packages can be downloaded from online repositories [40].

3.2.3 ROOT

ROOT is a framework for data processing, developed at CERN for HEP research. ROOT lets users save data (using a file format, the ROOT tree, able to powerfully handle huge amounts of data), access data (locally or remotely, using e.g. the Grid), analyse them with powerful mathematical and statistical tools, output publication-quality results, run and build applications. Developed in C++, it can also be bound to other languages such as Python and R [39].

3.3 Container Technology

3.3.1 Basics on Containers

Unix operating systems are based on a *kernel* space and a *user* space. The former handles the hardware and provides core system features, while the latter is where applications, libraries and system services run. Applications can only run on a given user space if they were packaged on an operating system (OS) with a user space compatible with the one of the operating system employed by the final user. Using containers, the user space becomes independent of the running OS, since they package up everything needed to run desired applications. The latter can thus be run inside the container with everything they require to work properly. This procedure solves issues involving dependencies and requirements and also lets users install multiple versions of the same software at the same time, useful for a large variety of works [41].

3.3.2 Singularity

Singularity is an open source container platform, allowing users to create and run containers [41]. First developed at Lawrence Berkeley National Laboratory, it quickly gained popularity among other HPC and academic sites. It guarantees reproducibility and security, easy harnessing of GPU, high speed networks and parallel filesystems power, and an easy sharing among users [41].

Containers may be started from image files with `.sif` suffix. Its command line interface lets users interact with containers and use them to run programs. Below are some of the most important commands [42]:

- `singularity shell IMAGE.sif`: starts a new shell within the container from the image file IMAGE.sif, allowing the user to interact with it as a virtual machine;

- `singularity exec IMAGE.sif COMMAND ARGUMENT(S)`: executes command COMMAND with argument(s) ARGUMENT(S) within the container from image file IMAGE.sif;

- `singularity run IMAGE.sif`: containers include user-defined scripts which define the actions the container performs when run. This command activates the scripts of the container from image file IMAGE.sif. This can also be done by calling the container as if it were an executable (`./IMAGE.sif`).

Singularity also lets users create their own container images with the `build` command (see [42]).

3.4 Interaction with Clusters

The foundation of the interaction between users and clusters is *remote server access*, which lets users interact with them from their own computers, even if not belonging to their network. The most common method of connecting to remote servers is the *Secure Shell* (SSH, [43]). SSH uses Public Key Cryptography (PKC, [44]) for client authentication on the server and for establishment of a secure channel between client and server [26].

When connection is established, users interact with the cluster using a shell: the most common one is the *Bourne Again Shell* (bash, [45]). A Graphical User Interface (GUI) may be available based on the *X Window System* (X11, [46]) [26].

Remote access via Computing Element can also be employed and is described in Section 3.5.3.

3.5 Batch Systems for Scheduling

Multiple programs run at the same time on clusters, so rules must be established. This is done by the *scheduling* process: in this way, available resources are fully used while optimally reducing concurrency for resources. Many scheduling programs or *schedulers* exist, e.g. *SLURM* and *HTCondor* [47, 48].

All scheduling solutions require the definition of jobs and queues. A *job* is an instruction list to run programs, while a *queue* is a list of jobs, ordered by definite criteria, waiting to be run. When users want to run a job on a cluster, they submit an instruction list to it, normally a bash script, which shall run as soon as possible. Submission should usually include the quantity of resources needed by the program to run. After submission, the job stays in a queue until requested resources are available: when this happens, the scheduler allocates them to run the job on any available worker node [26].

Below is the description of two scheduling solutions, being the ones used on the supercomputers used for this thesis work: SLURM and HTCondor.

3.5.1 SLURM

SLURM is an open source, fault-tolerant and highly scalable cluster management and job scheduling system for Linux clusters. It has three main functions: allocating access to resources to users for a given time, providing a framework to manage and monitor work on allocated resources, and arbitrating resource use among users by managing a queue of pending work. No kernel modifications are required for its operation and it also supports additional plugins.

Interactive Jobs with SLURM

An interactive SLURM batch job lets the user employ an interactive environment to run their programs, using the power of compute nodes. In this way, parallel jobs and all long jobs may also be used interactively, without having to switch to batch mode and use batch scripts (see next).

The user can send requests of resource allocation on compute nodes to SLURM with commands starting with `salloc` or `srun`. The request is queued and scheduled like a batch job; when accepted, the standard input, output and error streams of the job are connected to the terminal session from which came the request, making the user able to use them in an interactive way.

Below is an example of `salloc` command syntax:

```
salloc -N1 --exclusive --gres=gpu:4 -A myGroup -p partition_A --time=01:00:00
```

With this expression, the user requests for their interactive job 1 node (`-N1`) for exclusive use (`--exclusive`) including 4 GPUs (`--gres=gpu:4`) for the group account ”myGroup” (i.e. which project pays for this work, `-A myGroup`) on partition ”partition_A” (`-p partition_A`) for 1 hour (`--time=01:00:00`). When the space for the job is allocated, the user can run their programs interactively using the resources for which they have asked. Users can close the job and come back to the login node through the `exit` command.

SLURM automatically exports environmental variables defined in the source shell, making it easier for users to run programs with specific environmental requirements [33].

Batch Jobs with SLURM

Batch mode is the appropriate way to run large production runs. First, users prepare a file with a list of commands, which are either commands for the machine and requests like number of nodes, of GPUs, computation time, then they submit it to the scheduler, which looks at the system in order to satisfy the requirements and, as soon as they are fulfilled, runs a job executing the specified commands. Results are finally sent back to the user. Script files (with the `.x` suffix) have a structure like the one below:

```
#!/bin/bash
#SBATCH -A myGroup
#SBATCH -p myPartition
#SBATCH --time 00:10:00     # format: HH:MM:SS
#SBATCH -N 1                # 1 node
#SBATCH --ntasks-per-node=8 # 8 tasks out of 128
#SBATCH --gres=gpu:1        # 1 GPU per node out of 4
#SBATCH --mem=7100          # memory per node out of 246000 MB
#SBATCH --job-name=myBatchJob
#SBATCH --mail-type=ALL
#SBATCH --mail-user=myemail@mydomain.com

srun ./myExecutable
```

Job specifications are specified on the lines beginning with `#SBATCH`, which are not ignored as comments from the scheduler. Executing this script, the user requests a job for the group account "myGroup" (`-A myGroup`) on partition "myPartition" (`-p myPartition`) for ten minutes (`--time 00:10:00`) on one node (`-N 1`), with eight tasks (i.e. eight logical CPUs, `--ntasks-per-node=8`), one GPU (per node, `--gres=gpu:1`) and 7100 MB of memory (`--mem=7100`). The job is given the name "myBatchJob" (`--job-name=myBatchJob`) and the user gets notifications on job execution at mail address "myemail@mydomain.com" (`--mail-user=myemail@mydomain.com`). Commands to be executed by the job are given with the `srun` command: here the job expects its allocated resources to run `myExecutable` [33].

The job can be finally started by giving the command `sbatch myScript.x`, where the argument is the name of the script file [33].

3.5.2 HTCondor

HTCondor [48] is a software system that creates a HTC environment, allocating access to resources to users according to given criteria and providing a system to manage work. It is optimised for HTC-oriented jobs (the situation where a job must be run many times with different datasets being its typical use case), contrarily to SLURM, which is more HPC-oriented; moreover, being they different software, the script syntax also differs, as shown below.

The job management logic behind HTCondor is, however, the same as for SLURM: the user prepares a submit file with their resource requests and commands to be executed and submits it to the scheduler, which allocates requested resources and runs given commands. Below is an example of an HTCondor submit file:

```
executable = myExecutable.sh
arguments = $(ClusterId) $(ProcId)
```

```
output = output/myExecutable.$(ClusterId).$(ProcId).out
error = error/myExecutable.$(ClusterId).$(ProcId).err
log = log/myExecutable.$(ClusterId).log
queue
```

With this file, the user requests an execution of the script ”myExecutable.sh” (`executable = myExecutable.sh`, it may also be a command), passing to the command the arguments listed under `arguments`, writing the standard output on the file at position `output/myExecutable.$(ClusterId).$(ProcId).out`, the standard error at position `error/myExecutable.$(ClusterId).$(ProcId).err` and the logs (not the ones of the job, but of HTCondor, showing submission times, execution host and times) at position `log/myExecutable.$(ClusterId).log`. `queue` schedules the job. `ClusterId` and `ProcId` mark respectively the cluster number and the job number in each cluster: for a job as simple as this one the value of `ProcId` is 0. Any folder specified in given paths must already be existing: HTCondor cannot create them on its own and raises an error. Many other options also exist, dealing with e.g. system requirements and maximum runtime. In particular, option `universe` lets users submit to different platforms or architectures, handled by HTCondor as a concept it calls ”universe”. Default is the one called ”vanilla” (`universe = vanilla`): other ones exist, for e.g. container running (see Section 3.3).

Assuming the submit file is called `myScript.sub`, the job(s) may be started with `condor_submit myScript.sub` and monitored with `condor_q` (showing current status) or `condor_wait -status log/myExecutable.CLUSTER_ID.log`. `CLUSTER_ID` is actually a number, denoting the cluster actually attributed to the job at job start (and also set into `ClusterId`), and is written in the output of `condor_submit`.

3.5.3 Remote Submission

In a Grid environment, the execution of jobs workload is usually performed by installing an intermediate server, the *Computing Element* or *Computing Entrypoint*, that mediates the job request from users to local batch systems. As an additional functionality, an existing pool of HTCondor can be dynamically extended thanks to *HTCondor Exec Nodes* securely joining the central services of HTCondor. In this way, it is possible to perform an extension of a computing farm, including worker nodes that could become available, even for a limited interval of time. From the user perspective, there will just be one entry point, a HTCondor CE, which can be configured to route specific jobs to a specific type of resources belonging to the same pool.

3.6 Authentication and Authorisation

3.6.1 The VOMS System

In Grid computing environments, authorisation plays a key role in giving access to resources. Let us first introduce the following concepts:

- *Virtual Organisation* (VO): an abstract entity that groups users, institutions and resources into the same administrative domain;
- *Resource Provider* (RP): a facility that offers resources such as CPUs, network and storage to other parties, like VOs, according to specific rules.

They work together to handle information regarding the relationship of the user with their VO (group, roles, etc.) and what that user is allowed to do at a RP because of their membership of a given VO [49].

The *Virtual Organisation Membership Service* (VOMS) is a service managing authorisation information in the VO scope, developed to grant users authorisation to access resources depending on group membership, roles (administrator, student, etc.) and capabilities. It is composed of the following components:

- *User Server*: receives requests from a client and returns information about the user;
- *User Client*: contacts the server presenting the user certificate, receiving a list of groups, roles and capabilities of that user. All this procedure is executed in a secure and authenticated way;
- *Administration Client*: is used by the VO administrators to e.g. add users, create new groups, change roles and even more;
- *Administration Server*: accepts requests from clients and updates the database with all information on users and their roles and privileges [49, 50].

In order to use VOMS-controlled resources, users employ VOMS tools to generate a user proxy certificate, containing, in addition to the typical details included in an X.509 certificate [51], information from the VOMS server(s). All this information is returned in a structure also including the credentials of both the user and the VOMS server and a time validity (typically of twelve hours), with digital signature of the VOMS server. This system enjoys the security expected from a certificate authentication and authorisation system, e.g. certificates must be signed by a trusted Certificate Authority and be still valid [49, 50].

3.6.2 The INDIGO Identity and Access Management (IAM) Service

Further solutions for authentication and authorisation exist, as an alternative and evolution of the VOMS system. An example is the *INDIGO Identity and Access Management* (IAM) Service, a system providing a layer where identities, enrollment, group membership, other attributes and other authorisation policies on distributed resources are managed in a homogeneous and more straightforward way. Implementing the Virtual Organisation (VO) concept, IAM provides the following:

- a registration service, with support for periodic Acceptable Usage Policy (AUP) enforcement;
- an administration dashboard to handle common administrative tasks in an intuitive way;
- support for multiple authentication systems, not only X.509 certificates, but also JWT token exchange (see [52]), username-password systems and even social logins;
- support for account linking, letting users link multiple identities to their accounts;
- Application Programming Interfaces (APIs) to query and modify membership information [53, 54].

In the context of this thesis, the token exchange system has been used.

3.7 Data Access

3.7.1 The Rucio Framework

Rucio is an open-source software framework providing scientific collaborations with the possibility to organise, manage and access data, even if particularly heavy or not stored locally [55]. Originally developed for the ATLAS collaboration, it finds now useful application wherever are instruments generating heavy loads of data and heterogeneous storage and computing resources, even belonging to different administrative domains or organisations. Rucio incorporates all data management tasks (access, transfer, synchronisation, monitoring, usage statistics...) into a single easy-to-use solution and satisfies the needs of data-intensive sciences for functionality, scalability, robustness and efficiency [55].

In the context of the ATLAS collaboration, Rucio provides detector, simulation and user data handling, a unified interface for data management across heterogeneous storage and network infrastructures, data recovery and adaptive replication [55].

Rucio introduces systems for:

- data addressing (called *namespaces*);
- authorisation, authentication and permission handling (accounts);
- unified storage interface to distributed data centres;
- large-scale dataflow policies (*subscriptions*);
- consistent distributed state of the namespace on storage (*replication rules*).

From the architectural side, it is based on the following layers:

- the *clients'* layer, for user interaction;
- the *server*, offering authentication, a common API for interaction and further applications;
- the *core*, being the abstraction of all Rucio concepts;
- the *daemons*, taking care of background workflows.

In addition to these four main layers there are also storage resources, storage tools, queuing systems and databases [55].

3.7.2 The XRootD Framework

The XRootD software framework is a suite for remote access to files, with fast, low latency and scalable data access [24]. Developed at CERN and SLAC, it is compatible with all data formats and organised in a hierarchical filesystem-like way. It also supports authentication through several certificates [24].

XRootD functionalities can be invoked using command-line instructions, among which the most important ones are shown below:

- `xrdcp [options] source destination`: copies one or more files from location `source` to location `destination`, both either local or remote. XRootD paths begin with `root://` and also contain host (and sometimes port) information (paths look like `root://eosuser.cern.ch//eos/user/myfolder/myfile`);
- `xrdfs host[:port] [command [args]]`: executes typical file handling commands (e.g. `ls`, `mv`, `rm`) on XRootD resource paths. N.B.: the XRootD path must be split into its host and port information, to be inserted at arguments `host` and `port`, and the effective path, to be given as an argument to command `command` [24].

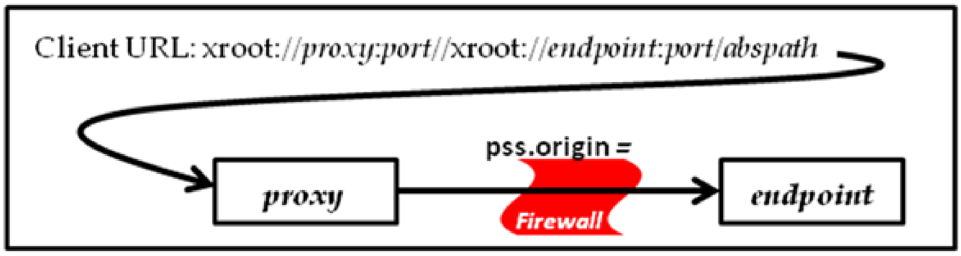

Client URL: xroot://proxy:port//xroot://endpoint:port/abspath
proxy
pss.origin =
Firewall
endpoint

Figure 3.1: Diagram of a forwarding mode proxy [56].

XRootD paths can also be opened from inside ROOT using the regular file-opening function (i.e. `TFile::Open("root://eosuser.cern.ch//eos/user/myfolder/myfile")`).

Also available is the possibility to establish a *XRootD proxy server*. This service is required whenever it is needed to provide a XRootD access to systems or parts of systems which have a limited connectivity, either because they can only access a private network or because they sit behind a firewall. Proxies act then as a "bridge" between the system and the external network. One proxy type is the *forwarding mode proxy* (Fig. 3.1), the one specifically used for this thesis work: the client prefixes the destination URL with the proxy location and then the proxy server contacts the specified endpoint on behalf of the client [56].

3.8 Description of Used Clusters

For this work, three clusters have been used: *MARCONI100* at CINECA [33], *Open Physics Hub* at the Department of Physics and Astronomy "Augusto Righi" of the University of Bologna [34] and *LXPLUS* at CERN [35]. This chapter presents these systems, with their hardware and software features.

3.8.1 MARCONI100

General Information

MARCONI100 (M100) is a HPC system in use at CINECA [57]. It is built on a IBM Power 9 AC922 architecture, with 8 Login IBM Power9 LC922 (similarly to compute nodes) login nodes, and employs Red Hat Enterprise Linux Server 8.1 as Operating System (OS) [33]. When activated it was the ninth most powerful device in the world (June 2020 TOP500 ranking [30]), while currently it is the twenty-first most powerful (June 2022 TOP500 ranking [30]). Tab. 3.1 shows its basic features [33].

Login nodes can be accessed through SSH at hostname `login.m100.cineca.it`. Graphic sessions may be run using the RCM (Remote Connection Manager) tool [33].

Model	IBM Power AC922 (Whiterspoon)
Racks	55 total (49 compute)
Nodes	980
Processors	2x16 cores IBM POWER9 AC922 at 2.6(3.1) GHz
Accelerators	4 x NVIDIA Volta V100 GPUs/node, NVLink 2.0, 16GB
Cores	32 cores/node, Hyperthreading x4
RAM	256 GB/node (242 usable)
Peak Performance	about 32 PFlops, 32 TFlops per node
Internal Network	Mellanox IB EDR DragonFly++
Disk Space	8PB raw GPFS storage
Global Peak Performance (theoretical)	31.6 PFlops
Single Node Performance	32 TFlops
of which CPU part (nominal/peak frequency)	691/791 GFlops
of which GPU part	31.2 TFlops
Memory Bandwidth (nominal/peak frequency)	220/300 GB/s

Table 3.1: Basic technical details of M100. Single node data refer to the AC992 CN node with 2 Power9 processors and 4 NVIDIA V100 GPUs [33].

Name	Total Dimension (TB)	Quota (GB)	Permanent	Backed Up	User/project specific
$HOME	200	50	Yes	Yes	User
$CINECA_SCRATCH	2000	None	No (temporary)	No	User
$WORK	4000	1024	Yes	No	Project

Table 3.2: M100 storage areas. $WORK area can be extended if needed. $CINECA_SCRATCH also features an automatic cleaning procedure deleting data older than 40 days [33].

Work on M100 has been carried out using the allocation obtained by INFN for tests on WLCG stack, coordinated by Dr. Tommaso Boccali (INFN-Pisa).

Disks and Filesystems

Users may employ three storage areas, listed in Tab. 3.2. Since processes running on login nodes have a time limit of 10 CPU-minutes (CPU time being the time spent by a process for logic and arithmetic operations on the CPU), users may use a dedicated data transfer virtual machine, accessible, either with SSH or RSYNC [58], at `data.m100.cineca.it` [33].

Nodes

As seen in Tab. 3.1, M100 is made of 980 compute nodes and 8 login nodes, connected though a Mellanox Infiniband EDR network arranged into an architecture called DragonFly++. Each node, be it login or compute, is made of 2 Power9 sockets, with 16 cores and 2 Volta GPUs each (i.e. 32 cores and 4 GPUs per node). Multi-threading is active

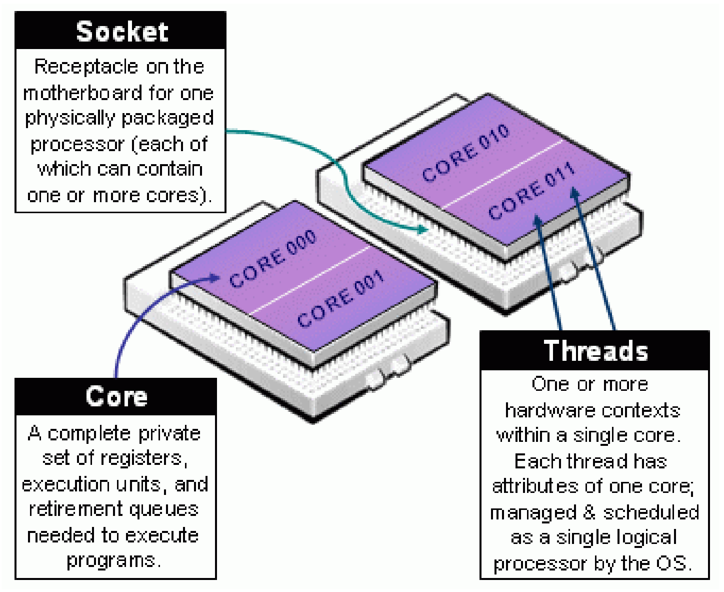

Socket
Receptacle on the motherboard for one physically packaged processor (each of which can contain one or more cores).
CORE 010
CORE 011
CORE 000
CORE 001
Core
A complete private set of registers, execution units, and retirement queues needed to execute programs.
Threads
One or more hardware contexts within a single core. Each thread has attributes of one core; managed & scheduled as a single logical processor by the OS.

Figure 3.2: Socket structure [33].

with 4 threads per physical core, or 128 total threads (i.e. "logical CPUs") per node (see Figs. 3.2 and 3.3).

CPUs are numbered from 0 to 127. The two Power9 sockets are connected via a 64 GBps X bus and each of them is connected via NVLink 2.0 with 2 GPUs.

Scheduling

M100 is a general purpose system used by many users at the same time, therefore long production jobs are submitted through the SLURM scheduler (see Section 3.5.1). M100 usage is based on a policy of node sharing among different jobs: one physical node can be allocated to multiple jobs of different users, with always the guarantee of exclusivity at the level of the single core and GPU through low-level mechanisms. Jobs may be submitted either interactively or in batch mode.

SLURM sees each node as having 128 virtual CPUs, 4 GPUs and 246 000 MB of memory, and assigns a node in a shared way, as seen. Users may request the full node in exclusivity with option `--exclusive` together with `--gres=gpu:4` [33].

SLURM jobs may use various partitions on M100, each with different features shown in Tab. 3.3 [33].

Compilers

M100 programming environment includes various compilers for the main languages of scientific usage (e.g., FORTRAN, C and C++), debuggers and profilers for code optimi-

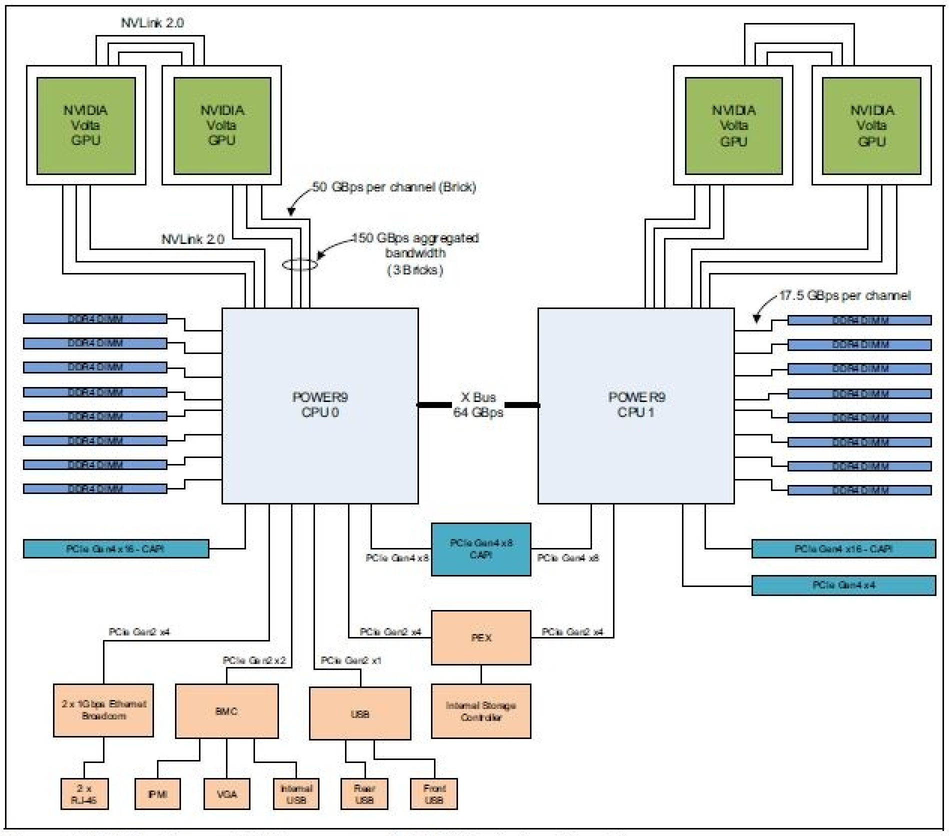


Figure 2-5 The Power AC922 server model GTH logical system diagram

Figure 3.3: The Power AC922 server model GTH logical system diagram [33].

SLURM partition	Job QOS	Cores/GPUs per job	Max wall-time	Max running jobs (or cores/nodes/GPUs) per user	Priority	Notes
m100_all_serial	normal	1 core, 1 GPU, max mem=7600 MB	04:00:00	4 CPUs/1 GPU	40	
	qos_install	16 cores	04:00:00	16 cores, 1 job per user	40	To be requested
m100_usr_prod	normal	16 nodes	24:00:00		40	Running on 880 nodes
	m100_qos_dbg	2 nodes	02:00:00	2 nodes/64 cores/8 GPUs	80	Running on 12 nodes
	m100_qos_bprod	256 nodes (min 17 nodes)	24:00:00	256 nodes	60	Running on 512 nodes, min 17 full nodes
m100_usr_preempt	normal	16 nodes	24:00:00		1	Running on 99 nodes
m100_fua_prod (EUROFUSION)	normal	16 nodes	24:00:00		40	Running on 87 nodes
	m100_qos_fuadbg	2 nodes	02:00:00		45	Running on 12 nodes
	m100_qos_fuabprod	32 nodes	24:00:00		40	Running on 64 nodes at same time
ALL PARTITIONS	qos_special	> 32 nodes	$> 24 : 00 : 00$		40	To be requested
ALL PARTITIONS BUT EUROFUSION	qos_lowprio	16 nodes	24:00:00		0	Active projects with exhausted budget

Table 3.3: M100 SLURM partitions. Core refers to a physical CPU, CPU to a logical one [33].

sation. Available compilers (together with libraries and other useful tools) can be listed with the command `module available` and loaded with `module load PATH`, where `PATH` is the path of the compiler, obtainable with `module available`.

For this thesis work, since custom installations of software were needed, a setup based on Conda has been used.

3.8.2 Open Physics Hub

General Information

The OPH (Open Physics Hub) cluster is a parallel computing facility at the Department of Physics and Astronomy "Augusto Righi" of the University of Bologna [34]. It is made of two distinct node groups, both based on the x86 architecture: "BladeRunner", for low-parallelism jobs, and "Matrix", for highly parallel jobs. It adopts as an OS Debian 5.10: for the purposes of this thesis work, a Singularity container replicating CentOS 7 (the OS of LXPLUS, see next) has been used.

Each node generally contains two sockets, each with a certain number of cores representing the basic computing units of this system and with said cores which may host one or two threads (hyperthreading). Contrarily to M100 and LXPLUS, this cluster is heterogeneous, i.e. contains nodes differing for core number, available memory and CPU producer (Intel or AMD); nevertheless, jobs are only given homogeneous nodes, unless requested otherwise. Job priority depends on already used resources on a fair share principle: the higher the amount of resources used by members of a certain group is, the lower job priority for this group becomes. Total storage amounts at 282 TB, divided into areas (home, scratch...) with a logic similar to the one of the other clusters.

OPH does not come with CVMFS installed: since it is needed for this thesis work, it has been mounted through the *cvmfsexec* package, a tool able to entirely manage mounting and unmounting of CVMFS repositories without being a privileged user [59].

Scheduling

The installed scheduler is SLURM, for which the same considerations exposed in Sections 3.5.1 and 3.8.1 apply, and the cluster supports MPI parallelisation.

3.8.3 LXPLUS

General Information

LXPLUS (Linux Public Login User Service) is the interactive logon service to Linux for CERN users [35]. It gives access to the homonymous cluster, consisting of public machines, based on the x86 architecture, running 64-bit CERN CentOS 7 (CC7) Linux.

LXPLUS is accessible via the SSH protocol at the domain `lxplus.cern.ch`. A set of machines running CentOS 8 is also available at `lxplus8.cern.ch`.

A wide range of shells is available, with bash being the most popular and the recommend one. LXPLUS machines can access temporary storage directories (`/tmp`, regularly cleaned), AFS file system (for home directories, group space and some scratch space) [60], EOS service [61] and CASTOR service [62]. Users may also employ various compilers and interpreters (such as gcc, g++ and the ones for Python, Ruby, Perl and Go), the CERN Mail Server, the CERN Print System and network services (like WWW, ssh and scp). The service is meant for interactive runs of non-intense jobs (other ones may get killed): intensive ones must be run with the CERN batch system, which may be invoked from LXPLUS [35].

The CERN Batch Service. Scheduling

The CERN Batch Service is the High Throughput Computing (HTC) batch system in use at LXPLUS. It is based on the HTCondor scheduler (see Sec. 3.5.2) [63].

HTCondor ensures that resources are attributed to users on a fair share, based on attributed quotas. Users with a lower number priority are given a higher priority and more machines than users with a higher number priority; depending on the number of used machines and run jobs, priority numbers are modified after a period of time. There is no limit to the number of jobs a user may submit, while a maximum of 10 000 jobs may run simultaneously (further ones are executed afterwards) and they are removed if running for too long, put in hold for more than 24 hours or restarted for more than ten times. Jobs are given by default one slot of a CPU core, 2 GB of memory and 20 GB of disk space; more resources can be asked for with the proper options.

Chapter 4

Framework Building

4.1 Goals and Issues

The goal of this thesis project is to develop a software framework allowing users to process typical data analysis workflows of the ATLAS experiment on HPC systems.

This work requires an analysis workflow as a test bed for the workflow and HPC systems for the workflow to run on. The choice fell on a workflow called *SeeSawAnalysis* (SSA), which shall be presented in the following section. For the choice of the devices on which to have this software run, MARCONI100 and OPH were adopted, all of them already described in Chapter 3.

SSA was first developed for LXPLUS (also presented in Chapter 3), on which the version of SSA is used as a reference to compare results and guarantee that the output is the same on all systems. Execution on LXPLUS requires the activation of the ATLAS general environment and of a specific LCG (a software stack on CVMFS containing external packages and tools, see [64]).

One aiming at adapting an analysis software like SSA on other devices encounters the following obstacles:

- SSA was developed for LXPLUS, therefore considering software and libraries installed on it. M100 and OPH software offers differ, also including different job schedulers (as seen, HTCondor on LXPLUS and SLURM on the other two);
- M100 even has a wholly different architecture (PowerPC instead of x86);
- SSA requires input files stored on the local file system, also with a determinate relationship between folders, and uses file names for naming produced output files. For example, datasets are collected using instructions like this one, which searches the folder at environmental variable `DATASET_DIRECTORY` for file(s) containing `FILE_NAME_SEARCHED_FOR` (also an environmental variable) in the name (so that files get recognised even with name variations, i.e. "DATA_2018" also

gets collected when searching for it even if it is called "DATA_2018_full") and already implies one constraint, i.e. that files have a "dataset directory/dataset files" structure with no subdirectories (`-mindepth 1 -maxdepth 1`):

```
find "$DATASET_DIRECTORY" -mindepth 1 -maxdepth 1 -iname
$FILE_NAME_SEARCHED_FOR
```

Moreover, it does not support data collection from files via XRootD, as a direct consequence of this; in fact, the program crashes if given XRootD paths, because the required folder structure may not be implemented on the XRootD server and, simply, taking XRootD paths as names for output files means there are slashes and columns (coming from the `root:` prefix) in the output file name, which is rejected by the system. This is an example of an issue arising not from architectural or software difference, but from conception of the software that may not be applied to use cases other than the ones for which the software was originally designed. Developers did not need to get files from XRootD servers, having them on the same device, nor was the imposed folder structure an obstacle, making file handling easier in the original software. All these ideas must be revised when adapting this software for other devices or needs: XRootD is a further option providing a useful feature, the file not having to be hosted locally anymore, and even becomes fundamental when there are storage constraints (this is the case of OPH, for which the files to be submitted for analysis in SSA exceed storage quota limits).

The framework must be able to solve these issues, overcoming the architectural and software limits and introducing new functions whenever needed. As a validation, it must be able to produce exactly the same output on all systems.

Next sections present SSA and the operations performed to adapt SSA to these needs and have it run on more devices than originally intended when first conceived. Next chapter discusses the final outcome and "certifies" the successful adaptation with comparison of the outputs.

4.2 The SeeSawAnalysis Data Analysis Workflow

SeeSawAnalysis is the analysis framework used for this study, developed by the ATLAS collaboration in the context of the work at [1, 2, 3, 4] for searches of exotic particles. It requires as an input either Ntuple files (containing data as ROOT trees) and metadata files, containing the metadata related to them. The code structure is optimised to look at three different kinds of topologies: final states with two leptons ("dilepton", divided for opposite sign and same sign), three leptons ("trilepton", with different jet multiplicity and Z boson requirements) and four leptons ("fourlepton", divided for final states charge). Three kinds of analysis region are defined, as we aim at looking for excess of the predicted particles with respect to the SM expectations, inspecting various observables and comparing with the SM backgrounds:

- control region (CR): rich of background events, it is used to study dominant backgrounds. From here a normalisation factor is extracted. It can be described, in a first approximation, as the ratio between the number of events in data and the number of Monte Carlo (MC)-generated ones. The normalisation factor is used to correct MC disagreement, if present;

- validation region (VR): similar to the control region, again with enough background events, it is used to validate the study of the background performed in the control region;

- signal region (SR): rich of signal events and poor of background ones, it is used to extract the signal strength (see [65] for a definition of signal strength).

From the point of view of the phase space, the following regions of interest are defined depending on the final states being with three or four leptons:

- Trilepton channel:
  - *ZL*: requiring a Z decaying leptonically in the final state;
  - *ZLveto*: vetoing leptonically decaying Z's;
  - *JNLow*: having low jet multiplicity (less than two).

- Four-lepton channel:
  - *Q0*: looking for two same-sign lepton pairs with a total charge of the system equal to zero;
  - *Q2*: having one lepton of different sign giving a total charge equal to 2 [66].

We look for new heavy particles, producing leptons, high energy bosons and large missing energy transverse. The normalisation factor worked out from the control region is first applied at the validation region as a check and then used in the signal region. Finally, a profile likelihood is employed to compare signal events with SM predictions and exclusion limits are imposed if an excess is not found. Data are not put immediately into the signal region, in order to avoid biasing.

From the point of view of the code, it is divided into modules, the actual frameworks executing the program. Commands to be executed are defined into a submodule containing all the algorithms to perform the search, such as the definition of objects, analysis observables and even more. Useful definitions of e.g. particle pairs and variable ranges are also included. Before use, the user defines relevant (environment) variables such as build, input and output folders into a configuration shell script file. The main commands used for this thesis work are the following:

- `list_samples`: returns a list of the Ntuples inside the input directory;
- `process ANALYSIS_REGION`: performs actual processing on given `ANALYSIS_REGION` (whose job name also includes the distinction among control, validation and signal), with the operations exposed above. Options for specific processing requests are also available;

For example, if a user wants to inspect the ZL control region, they first invoke the configuration shell script for important environment variables, then invoke the commands listed above, giving as an argument the job name (in this case `CR_3l2j_2VB`) associated to the corresponding analysis region. First, they check the presence of the input file with `list_samples CR_3l2j_2VB`, then they can execute the command `process CR_3l2j_2VB` to start the analysis. At the end, only events satisfying the requirements shown above about the regions of interest, with the addition of other selection criteria explained in [66], are present in the output file. Plots are already made available as an output of `process`: the user may get further ones, formatted with ATLAS official cosmetics for publication, with command `plot`.

4.3 Realisation

4.3.1 Installation on MARCONI100

In order to have SSA installed on M100, I have performed the following:

- rebuilt SSA and added missing libraries and software, either required by recompilation itself or just during execution. Among missing software were also included ROOT and *ONNXRuntime* ([67]);
- adapted HTCondor template scripts, made available to users to let them create jobs with SSA, into SLURM ones, converting the HTCondor syntax into the one of SLURM;
- added a Singularity container with the VOMS client for proxy generation. Such container is run on a compute node (since it requires CVMFS, only installed on compute nodes on M100) and it has been used to generate the proxy. This Singularity container also has the custom build of HTCondor used for the activation of the HTCondor node for remote access (see Section 3.5.3);
- M100 compute nodes have a limited external connectivity, contrarily to login nodes. In order to make it possible for XRootD to run, a XRootD proxy server as presented in Section 3.7.2 has been installed and configured at INFN-CNAF Tier 1 (since M100 nodes can access the machines within INFN-CNAF at domain

`cnaf.infn.it`), establishing a sort of "bridge" between compute nodes and the external network, circumventing this limitation.

Running SSA on M100 implies, in the end, the following steps:

1. launching the Singularity container (with command `exec`) on a compute node creating the VOMS proxy for ATLAS;

2. running SSA in its running steps, as exposed in Sec. 4.2.

For proxy generation, the user may also choose to import a proxy generated elsewhere and set the `X509_USER_PROXY` environmental variable with its path. The Singularity container provides, however, the possibility of doing this directly on M100, without relying on external "sources".

4.3.2 Installation on OPH

In order to have SSA installed on OPH, I have performed the following:

- implemented a Singularity container replicating the LXPLUS OS (CentOS 7), since OPH architecture is compatible with the one of LXPLUS. Operations have also been performed to also have CVMFS installed on OPH via *cvmfsexec*, which is fundamental for a successful compilation of SSA;

- rebuilt SSA into a fresh build in the container;

- adapted HTCondor scripts into SLURM ones.

Running SSA on OPH implies, in the end, the following steps:

1. running (with command `run`) the Singularity container with the LXPLUS environment emulator and CentOS 7 (the OS on LXPLUS);

2. performing the same preliminary operations of LXPLUS (i.e. ATLAS environment and LCG release setup, as seen in Section 4.1);

3. creating a proxy certificate, having it associated to environmental variable `X509_USER_PROXY` and sourcing a shell script to set up certificate handling;

4. running SSA in its running steps, as exposed in Sec. 4.2.

4.3.3 Function Addition to SeeSawAnalysis

With respect to the original version of SSA, I have also added the possibility for it to also use XRootD paths, to solve the software limitations exposed in Section 4.1. The problem was in particular not in the ROOT macros inside SSA, but in the complex structure of scripts making up SSA, which make their intervention on file paths (i.e. assume that files are in a certain folder structure and use file names to name output files, as seen before) that in the new operating context just interferes.

Concrete solution consisted in removing all this "chain of assumptions" and replacing it with a new structure, with proper commands and environment variables such that XRootD paths could go directly from the SSA configuration file to ROOT without being unjustifiably altered. Such a solution also included the implementation of a new naming scheme for output files, always using file names but in a way such that special symbols like slashes do not interfere.

For example, the code that had been shown in Section 4.1 has been converted into this one:

```
cat $DATASET_LIST_TEXT_FILE | egrep "$FILE_NAME_SEARCHED_FOR"
```

Files to inspect are listed in a text file at environmental variable `DATASET_LIST_TEXT_FILE`, which is searched for file(s) containing in the name the string `FILE_NAME_SEARCHED_FOR` (also an environmental variable). The command `find` had to be fully replaced, since it is not fully compatible with XRootD filename standard.

This solution has first been implemented on the SSA version on LXPLUS, so that LXPLUS could also work on the same data files, which are hosted on a XRootD server, used by the versions on M100 and OPH. This allows results comparison in the end to ensure that all versions yield the same results. The solution has then been copied to the M100 and OPH versions, implementing this new functionality also on these new systems.

4.3.4 Packing the Framework

Operations depicted in the previous sections have finally been packed in an actual framework, making it possible for all operations above to run automatically, with a single command invocation, and thus with more ease.

For all systems, this involved the creation of a script running at first preliminary operations required for SSA to run on the specific system and then SSA actual commands. Below are specific details for the three devices:

- M100: since a VOMS proxy generator system is not installed by default on M100, proxy creation has been left independent (the user may thus use the Singularity container, use a tool of their choice or import a proxy of their own). The script uses the proxy at environmental variable `X509_USER_PROXY`;

- OPH: as seen on Section 4.3.2, SSA is run on a Singularity container replicating the LXPLUS environment. Therefore, the script replicates the one of LXPLUS, the only difference being that proxy creation is independent as on M100 (since the image, contrarily to the original LXPLUS system, does not include the VOMS proxy generator); again, the script uses the proxy at environmental variable `X509_USER_PROXY`;
- LXPLUS: the script sets the ATLAS environment up, also creating the proxy via VOMS, fully supported on this system.

4.3.5 Remote Submission from Grid Endpoints

In our testbeds, we followed the approach described in [68] and [69]: the strategy has been to run a containerised instance of the Condor service on the worker node (which could be either a node of M100 or OPH), as an unprivileged user. In the Condor instance, it has also been possible to set a custom configuration (like node architecture, the maximum number of cores and processes per core, etc.) and a routing configuration. The latter has then been used by the HTCondor Computing Element for dispatching the jobs to the external nodes. On the other side, the user could perform a standard HTCondor submission request to the HTCondorCE from a generic user interface machine, adding all needed specifications, like executables and their parameters, and authentication (which could be either with a X.509 proxy certificate or JWT tokens).

The full submission chain has been successfully tested for remote submission of the SSA framework, both on M100 and OPH, using either a X.509 proxy certificate or JWT token authentication.

Chapter 5

Results

5.1 General Comments. Proper Functioning Checks

For the creation of the framework to be considered successful, returned results on the new systems (M100 and OPH) must be found to be equal to the ones returned on LXPLUS. The following tests have been conducted:

- checking general statistics. This is output returned on the shell by the software, providing information about running processes, job management, amount of read and written data, number of elaborated events and of the ones passing SSA selection filters. It must be the same on all systems. Even though architectures and software foundations used by SSA to run are different on the three systems, the fact that results are perfectly the same means that, computationally, functioning is equivalent, fully preserving the physical-mathematical logical workflow;
- checking output graphs (see next section);
- performing all of the above on more use cases, in order to have a more solid proof of the success of the adaptation. This test must also be passed.

The three tests above have all been passed, meaning that the framework development can be considered successful. SSA is now able to run on devices other than the one for which it had been originally designed (LXPLUS) and results are perfectly the same on all systems, making the framework of viable usage. Users are now able to use other devices, harnessing their further benefits (e.g., the higher computational speed of M100) with the guarantee that results be the same as if the study were conducted on the original system.

One relevant consideration is the relative ease with which the framework has been realised. Adopted solutions are neither particularly hard nor "exotic" (i.e. using non-standard libraries). Moreover, no administrator privileges are required to perform all these operations: a common user account to log in the systems and a VOMS authentication, either via proxy certificate or token exchange, are the only requirements. This makes workflow deployment on new systems particularly convenient on this side.

5.1.1 Validation Graphs

SSA returns as an output multiple graphs, containing relevant analysed observables. For this test to be considered successful, output graphs must be equal on all systems. Such a test provides a large statistics, being the number of entries considerable (of the order of 10^6 on some graphs), and is, therefore, much more quantitative.

This test has been found to be successful: an equivalence of graphs has been observed on all systems, providing a further proof to the correct creation of the framework.

In Fig. 5.1 is an example of this comparison, performed on the graphs referring to the trilepton transverse mass m_T (using input data "diboson"). This observable is defined as the magnitude of the vector sum of the trilepton momentum $\vec{p_T}$ and the MET (Missing Energy Transverse) E_T^{miss} of the trilepton system (Eq. 5.1) [1, 2, 3, 4]:

$$m_T = \left| \sum_{i=1}^{3} \vec{p_{T,i}} + E_T^{miss} \right| \tag{5.1}$$

On the y axis is the weighted number of events for each m_T value, weighted to consider e.g. cross sections and selection efficiencies. In Graphs (a) and (b) of Fig. 5.1, the graphs obtained on M100 and OPH are overlaid with the one obtained on LXPLUS. For a better presentation (i.e. to avoid that fluctuations in count values make it hard to appreciate the overlaying), graphs have been rebinned from 1 GeV bins to 10 GeV bins. It may be noticed that compared graphs match perfectly: the graphs obtained on M100 and on OPH overlay perfectly on all bins with the one obtained on LXPLUS, which, being it the original system on which SSA has been developed, acts as the reference.

For a more solid testing, the bin-to-bin count ratio of said graphs has also been calculated and plotted: this would underline possible deviation that might escape a purely visual inspection. Ratios are shown in Graphs (c) for M100 data over LXPLUS data and (d) for OPH data over LXPLUS data: in both cases the ratio is of the "new" system data over LXPLUS reference data. The ratio is unitary for all mass values, denoting the equality of obtained results. Unusually large error bars are noticed, however, for high mass values and, in a smaller measure, for the lowest ones. They can be traced back to issues with floating-point numerals, whose behaviour at the least significant digit, as known, must be handled with particular care on processors (because the two entries to compare might turn out to differ even if generated by the very same input and computational processes), or to the case where the ratio for two bins with zero entries or for two in which the entry for LXPLUS (at the denominator) is zero is calculated (the two entries differing again because of the issues with floating point numerals), which is not

Device	Average Execution Time [s]	Notes
LXPLUS	250 ± 50	-
M100	51.6 ± 1.1	-
OPH "Blade-Runner"	96.1 ± 0.7	Node for low parallelisation processes [34]
OPH "Matrix"	57.5 ± 0.4	Node for massively parallel jobs [34]

Table 5.1: Benchmark test of the framework, performed using 4 CPUs on all systems. All values are the average of 10 runs. Since OPH nodes are heterogeneous (Section 3.8.2), the test has been conducted on two different-type nodes on it.

possible and on a computing system yields an error. All these considerations, however, do not refute the successful passing of the test, being due to intrinsic computational issues.

5.2 Further Evaluation

5.2.1 Speedup Benchmarking

For a better appreciation of the opportunity offered by HPC systems, a benchmark speedup test has been conducted, measuring the framework execution time (for input data "raretop") on the different systems. Results are shown in Table 5.1. These results only refer to the strict data processing part of the framework, excluding preliminary environment configuration operations, which, being base environments different, necessarily differ on the three systems. It may be noticed the great improvement in running time on M100 and OPH. This underlines the relevance of finding a way to have one's analysis workflow run on systems that are more powerful than the one originally used.

5.3 Possibilities for Follow-Up

This work may be extended in a variety of ways. Its key ideas can be also made apply to the full computing production system of the ATLAS experiment at CERN. First of all, the computing frame may be extended to the case of particle-collision event generation, detector simulation and data derivation (from RAW data to analysis-oriented format). In addition, the framework can be interfaced to the ATLAS production and analysis system (PANDA) [25]. In HPC there is a large usage of techniques like machine learning and deep learning: the deployment of such tools within the framework may be evaluated, in order to fully exploit the capabilities of HPC systems. Moreover, a study of the

Modules	3
Computing Nodes	5000
Peak Performance	>200 PFlops
RAM Memory	>3 PB
Input/Output Memory	150 PB
Storage	150 PB
Bandwidth	1 TB/s
Interconnection Bandwidth	200 Gb/s
Power Consumption	9 MW
Power Usage Effectiveness	1.08
Investment	240 M€
Footprint	>1500 m^2

Table 5.2: Basic features of LEONARDO [72].

user experience may be performed, providing physicists with a direct access to HPC computing resources, hiding the complexity of the computing environments [70].

All of the above may be performed not only on the devices employed for this work, but also on next generation ones, continuously appearing with ever increasing performance. This is the case, for example, of LEONARDO (see next), whose activation is foreseen in 2023 at the *Tecnopolo* in Bologna [71].

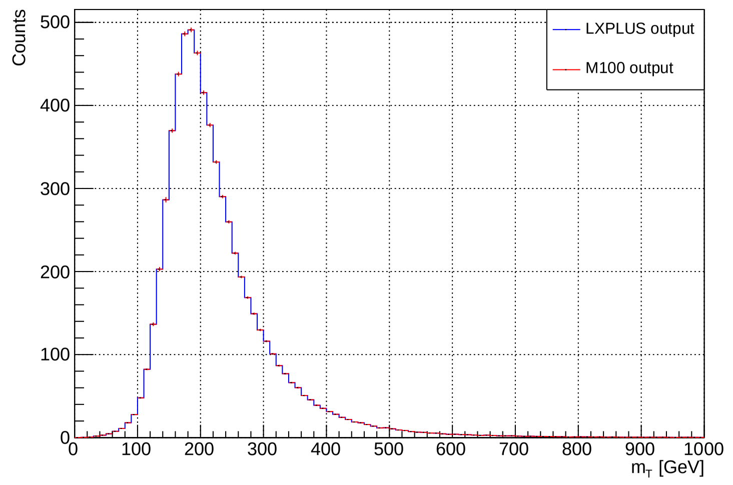
A follow-up of this thesis work may find useful application in Physics and even more, either in other strictly scientific domains (e.g. climate studies and genome analysis, as seen in Section 2.2) or general data-intensive fields (e.g. finance), i.e. generally all fields where Big Data paradigms apply.

5.3.1 The New Leonardo Supercomputer

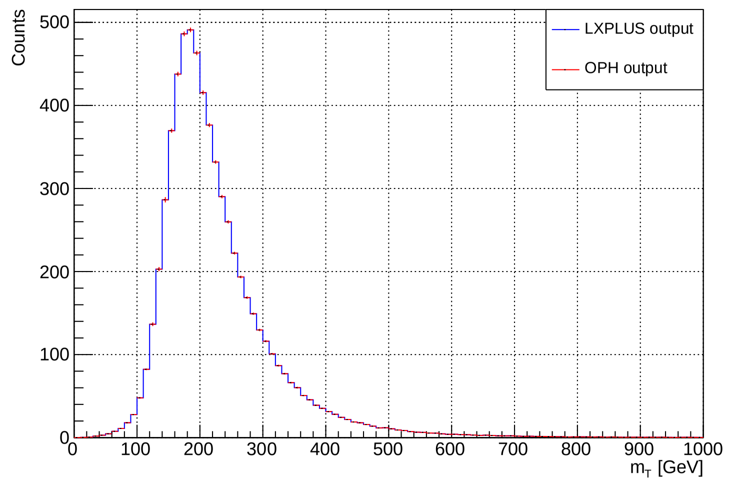
The city of Bologna (Fig. 5.2) and Italy will have the privilege to host one of the pre-exascale class computers funded by the European Commission as part of EuroHPC, the joint collaboration between European countries and the European Union to develop and support exascale supercomputing by 2022/2023. The Italian computer (the other ones being LUMI in Kajaani, Finland, and - yet to be installed - in Barcelona, Spain) is LEONARDO. The LEONARDO project was presented by CINECA, with collaboration of the Italian Ministry of Education, University and Research, INFN (*Istituto Nazionale di Fisica Nucleare*, National Institute of Nuclear Physics) and SISSA (*Scuola Internazionale Superiore di Studi Avanzati*, International School of Advanced Studies). In Tab. 5.2 are its main features [72].

This new device will be the second most powerful computing device in the world, providing a breakthrough for many scientific fields, not only Physics, but also pharmaceutical research, pandemics studies (e.g., LEONARDO will only take a few hours to evaluate how millions of molecules interact with COVID19 Spike protein, much faster than the days taken by M100), weather forecasting, handling of natural disasters and industrial development. It shall also greatly improve Italy's and the European Union position in the HPC field (compare with the current TOP500 positions in Tab. 2.1).

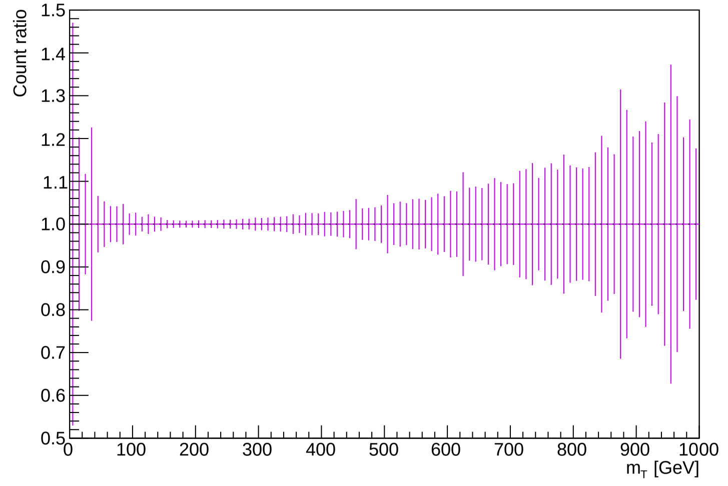
In conclusion, the activation of this device, together with the other ones foreseen by the EuroHPC project, shall give plenty of new opportunities for Physics and scientific analysis, Big Data processing and more.



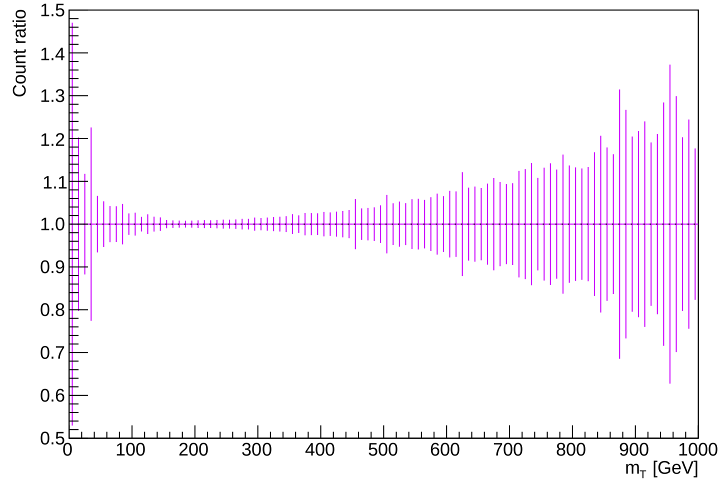
(a)



(b)



(c)



(d)

Figure 5.1: Comparison of graphs referring to the trilepton transverse mass for input data "diboson". In (a) is the overlay of results returned by the program run on M100 on results obtained on LXPLUS; in (b) is the overlay of OPH results on LXPLUS ones; in (c) is the ratio between weighted counts on M100 and the ones on LXPLUS; in (d) is the ratio between weighted counts on OPH and the ones on LXPLUS.

Figure 5.2: LEONARDO is to be hosted in Bologna, on the redeveloped site of the former *Manifattura Tabacchi* (tobacco factory) [73].

Chapter 6

Conclusions

The development of a software framework allowing users to process a typical data analysis workflow of the ATLAS experiment (*SeeSawAnalysis*) on HPC systems has been presented. The context behind this work has been shown in detail, covering the Physics issues the workflow addresses, the experimental context, the concrete functioning of the workflow and the theory on High Performance Computing, with technical details on used HPC clusters (MARCONI100, OPH and LXPLUS) and software. The development had to address multiple issues, either due to hardware and software differences between the three systems or to new needs that have emerged. Discussion of results has proven the development of the framework to be successful: the output has been found to be equal on all systems and in many use cases, guaranteeing the conservation of the physical-mathematical workflow of SSA on all systems and the possibility to actually harness much more computational power than before. Speedup benchmarking and remote submission tests have also been performed, permitting a further appreciation of the opportunities offered by HPC clusters, especially M100. This work may be continued in many ways, with the use for example of new generation devices like LEONARDO, providing further benefit on a larger scale.

Acknowledgements

This is the conclusion of a path along which I enjoyed the assistance and company of many relevant people. First, I would like to thank Prof. Lorenzo Rinaldi, who, first with his course then as my supervisor, introduced me to the fascinating field of HPC and how this may be employed as a breakthrough for research in Physics, and Dr. Giuseppe Carratta, who assisted me in the adaptation of the SeeSawAnalysis framework for the purposes of this work. I would also like to thank Lucia Morganti (CNAF) for the setup of the XRootD proxy server and Stefano Dal Pra (CNAF) and Daniele Spiga (INFN-Perugia) for the setup of the remote submission system. Then, I thank my parents, always at my side and fundamental in letting me become the person I am today. I would also like to thank my friends, for being such a merry and inspiring company. Finally, I thank my scout group, a long time company who let me live plenty of experiences and forge valuable bonds.

Bibliography

[1] The ATLAS Collaboration, *Search for type-III seesaw heavy leptons in dilepton final states in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector*, Eur.Phys.J.C 81 (2021) 3, 218

[2] The ATLAS Collaboration, *Search for type-III seesaw heavy leptons in leptonic final states in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector*, arXiv:2202.02039v1

[3] Carratta G., *Search for Type-III SeeSaw heavy leptons in leptonic final states using proton-proton collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector*, Ph.D. Thesis in Physics, University of Bologna (2022)

[4] Novak T., *Search for type-III seesaw heavy leptons with the ATLAS detector at the LHC*, Ph.D. Thesis, University of Ljubljana (2020)

[5] Goldstone J., Salam A., Weinberg S., *Broken Symmetries*, Phys.Rev. 127 (1962) p. 965-970

[6] Workman R.L. *et al.* (Particle Data Group), to be published in *Prog. Theor. Exp. Phys. 2022*, 083C01 (2022)

[7] Perkins D. H., *Introduction to High Energy Physics - 4th Edition*, Cambridge University Press (2000), p. 7-12

[8] https://commons.wikimedia.org/wiki/File:Standard_Model_of_Elementary_Particles_%2B_Gravity.svg

[9] The ATLAS Collaboration, *Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC*, Physics Letters B, Volume 716, Issue 1, 17 September 2012, p. 1-29

[10] The CMS Collaboration, *Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC*, Physics Letters B, Volume 716, Issue 1, 17 September 2012, p. 30-61

[11] Halzen F., Martin A.D., *Quarks & Leptons: An Introductory Course in Modern Particle Physics*, J. Wiley & Sons. (1984), p. 326-330

[12] Franceschini R., Hambye T., Strumia A., *Type-III seesaw mechanism at CERN LHC*, Phys. Rev. D 78 033002 (2008)

[13] Han T., Liao J., Liu H., Marfatia D., Ruiz R., *BSM ν physics: complementarity across energies - a white paper for Snowmass 2021*, Contribution to Snowmass 2021 - arXiv:2203.06131v1

[14] Abada A. *et al.*, *Low energy effects of neutrino masses*, JHEP12 061 (2007)

[15] Goswami D., Poulose P., *Direct searches of Type-III seesaw triplet fermions at high energy e^+e^- collider*, Eur. Phys. J. C 78 42 (2018)

[16] https://commons.wikimedia.org/wiki/File:LHC.svg

[17] The ATLAS Collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider*, 2008 *JINST* 3 S08003 (2008), p. 1-5

[18] The CMS Collaboration, *The CMS experiment at the CERN LHC*, 2008 *JINST* 3 S08004, Citeseer (2008)

[19] The LHCb Collaboration, *The LHCb detector at the LHC*, 2008 *JINST* 3, 08, S08005, IOP Publishing (2008)

[20] The ALICE Collaboration, *The ALICE experiment at the CERN LHC*, 2008 *JINST* 3, 08, S08002, IOP Publishing (2008)

[21] The ATLAS Collaboration, *ATLAS Computing: technical design report*, Technical design report. ATLAS, CERN, Geneva (2005)

[22] Jacob B., Brown M., Fukui K., Trivedi N., *Introduction to Grid Computing* (IBM Redbooks) (2005), pp. 3-17

[23] WLCG, *LHC Run 2 (2014-2018)*, WLCG (2019) at https://wlcg-public.web.cern.ch/about

[24] https://xrootd.slac.stanford.edu

[25] Maeno T. *et al.*, *PANDA*, J. Phys.: Conf. Ser. 898 052002 (2017)

[26] Almeida S., *An Introduction to High Performance Computing*, International Journal of Modern Physics A, Vol. 28, No. 22n23, 1340021 (2013)

[27] Hill M.D., Marty M.R., *Amdahl's Law in the Multicore Era*, Computer, vol. 41, no. 7 (July 2008), p. 33-38,

[28] Manuali C. *et al.*, *Efficient Workload Distribution Bridging HTC and HPC in Scientific Computing* in Murgante, B. *et al.*, *Computational Science and Its Applications – ICCSA 2012*. ICCSA 2012. Lecture Notes in Computer Science, vol 7333 (2012), Springer, Berlin, Heidelberg https://doi.org/10.1007/978-3-642-31125-3_27

[29] Nielsen F., *Introduction to HPC with MPI for Data Science* (Springer) 2016, p. 3-17

[30] *TOP500 Supercomputer List* at top500.org

[31] The ATLAS Collaboration, *Harnessing a supercomputer for ATLAS* at https://home.web.cern.ch/news/news/computing/harnessing-supercomputer-atlas

[32] Hollowell C. *et al.*, *Mixing HTC and HPC Workloads with HTCondor and Slurm*, J. Phys.: Conf. Ser. 898 082014 (2017)

[33] *MARCONI100* at https://www.hpc.cineca.it/hardware/marconi100

[34] *Open Physics Hub* at https://site.unibo.it/openphysicshub/en

[35] *The LXPLUS Service* at https://lxplusdoc.web.cern.ch/#features-offered

[36] *GNU Make* at gnu.org/software/make

[37] *GNU Time* at gnu.org/software/time

[38] *CernVM-FS* at cvmfs.readthedocs.io

[39] *ROOT* root.cern

[40] *CONDA* docs.conda.io/en/latest/

[41] *Introduction to Singularity* at sylabs.io/guides/3.5/user-guide/introduction.html

[42] *Quick Start* of Singularity at sylabs.io/guides/3.5/user-guide/quick_start.html

[43] Shirey R., *Internet Security Glossary, Version 2* (2007) at man.openbsd.org/ssh

[44] datatracker.ietf.org/doc/html/rfc4949

[45] *GNU Bash* at gnu.org/software/bash/

[46] x.org/wiki/

[47] *SLURM Overview* at slurm.schedmd.com/overview.html

[48] *HTCondor Overview* at htcondor.org/htcondor/overview/

[49] Alfieri R., Cecchini R., Ciaschini V., Dell'Agnello L., Frohner Á., Gianoli A., Lőrentey K., Spataro F., *VOMS, an Authorization System for Virtual Organizations*, in Fernández Rivera F., Bubak M., Gómez Tato A., Doallo R., *Grid Computing - First European Across Grids Conference - Santiago de Compostela, Spain, February 2003 - Revised Papers*, Springer (2004)

[50] Alfieri R., Cecchini R., Ciaschini V., Dell'Agnello L., Frohner Á., Lőrentey K., Spataro F., *From gridmap-file to VOMS: managing authorization in a Grid environment*, Future Generation Computer Systems, Volume 21, Issue 4 (April 2005), p. 549–558

[51] Chadwick D.W., Otenko A., *The PERMIS X.509 role based privilege management infrastructure*, Future Generation Computer Systems, Volume 19, Issue 2 (February 2003), p. 277-289

[52] Jones M., Nadalin A., Campbell B., Bradley J., Mortimore C., *OAuth 2.0 Token Exchange*, RFC 8693 (2020), https://rfc-editor.org/rfc/rfc8693.txt

[53] *Overview* of INDIGO IAM at indigo-iam.github.io/v/v1.7.2/docs/overview

[54] Ceccanti A., Vianello E., Giacomini F., *Token-based authentication and authorization in practice*, EPJ Web of Conferences 245, 03021 (2020)

[55] Barisits M., Beermann T., Berghaus F. *et al.*, *Rucio: Scientific Data Management*, Comput Softw Big Sci 3, 11 (2019)

[56] *XRootD - Proxy Storage Services (Caching, Non-Caching, & Server-less Caching) Configuration Reference* at xrootd.slac.stanford.edu

[57] *CINECA* cineca.it

[58] *RSYNC* rsync.samba.org

[59] *The cvmfsexec repository* at github.com/cvmfs/cvmfsexec

[60] cern.service-now.com/service-portal?id=service_element&name=afs-service

[61] eos-web.web.cern.ch/eos-web/

[62] castor.web.cern.ch/castor/

[63] *CERN Batch Service User Guide* at https://batchdocs.web.cern.ch/index.html

[64] *LCG Releases Introduction* at lcgdocs.web.cern.ch/lcgdocs/lcgreleases/introduction/

[65] Gross E., *Practical Statistics for High Energy Physics*, Proceedings of the 2015 European School of High-Energy Physics, Bansko, Bulgaria, 2 - 15 September 2015, edited by M. Mulders and G. Zanderighi, CERN Yellow Reports: School Proceedings, Vol. 4/2017, CERN-2017-008-SP (CERN, Geneva, 2017)

[66] Carratta G. on behalf of the ATLAS Collaboration, *Search for Type-III SeeSaw heavy leptons in leptonic final states in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector*, PoS(PANIC2021)139

[67] *ONNXRUNTIME* at onnxruntime.ai

[68] Boccali T. *et al.*, *Extension of the INFN Tier-1 on a HPC system*, EPJ Web Conf. 245 09009 (2020) arXiv:2006.14603

[69] Rodrigues A.M., Spiga D., Boccali D., *Enabling CMS Experiment to the utilization of multiple hardware architectures – a Power9 Testbed at CINECA*, poster presented at ACAT 2021 Conference, https://indico.cern.ch/event/855454/contributions/4604962

[70] jupyter.org

[71] cineca.it/en/hot-topics/Leonardo-announce

[72] cineca.it/en/our-activities/data-center/hpc-infrastructure/leonardo

[73] *Tecnopolo di Bologna, via all'ultima fase dei lavori nell'ex Manifattura Tabacchi* on the Emilia-Romagna region website (fesr.regione.emilia-romagna.it/notizie/2022/marzo/tecnopolo-di-bologna-via-allultima-fase-dei-lavori-nellex-manifattura)

List of Figures

List of Tables