A "look-see" at data from Run 3 at ALICE

Miles Kidson

Supervisors: Prof. Zinhle Buthelezi, Dr. SV Fortsch, & Prof. Tom Dietel
Assisted By: Dr. B Naik (Postdoctoral fellow)

UCT Honours 2022

Abstract

We investigate pilot-beam data from proton-proton collisions in Run 3 at ALICE, checking if the new Muon Forward Tracker, Inner Tracking System, and Online-Offline analysis framework are working as intended.

Contents

1 Introduction

Run 3 is the latest period of data capture at the LHC, with an intended centre of mass energy per collision of $\sqrt{s} = 13.6\,\text{TeV}$ and increased luminosity of collisions—a factor 10 increase in integrated luminosity for Pb-Pb collisions. For Run 3, ALICE is moving from a triggered readout system to a combination of triggered and continuous readout. In order to achieve this, many detectors and their front-end electronics were upgraded, some new detectors were added, and the analysis framework was overhauled entirely.

The Inner Tracking System (ITS) was upgraded with an entirely new pixel detector technology, hoping to greatly increase the resolution when determining the primary collision vertex. The Muon Forward Tracker (MFT) is one of the new detectors added. Its primary use is to assist the Muon Spectrometer (MCH) with vertexing and tracking in the forward region of ALICE and was developed with the same technology as the ITS. To deal with the increased volumes of data, a new analysis framework was introduced called Online-Offline (O2).

This report aims to use O2 to have a "look-see" at data coming from the ITS and MFT to see if the detector and analysis tools are working correctly. The data used in this investigation is from two proton-proton collision runs performed in October 2021, at a centre-of-mass energy of 900 GeV. This is not an energy we expect to use for physics data analysis but is good enough for this purpose.

2 Background

The ALICE (A Large Ion Collider Experiment) detector is a detector experiment at the Large Hadron Collider (LHC) at CERN. Its primary goal is the investigation of "strongly interacting matter at extreme energy densities, where a formation of a new phase of matter, the quark-gluon plasma, is expected" [5]. It achieves this goal by studying the products of head-on collisions of heavy ions such as lead, called Pb-Pb collisions for short. It also studies proton-lead (p-Pb) and proton-proton (p-p) collisions.

2.1 Coordinates

The coordinate system used at ALICE needs to be discussed in order to fully explain the scope of this report. A modified cylindrical coordinate system, shown in figure 2.1, is used as most detectors in the experiment are cylindrically symmetric about the beamline of the LHC.

We place the z-axis along the beamline with its origin at the interaction point (IP). The IP is the point at which collisions happen, right in the center of the detector. The angle around the z-axis is called the azimuthal angle, denoted by φ. Sometimes in the literature φ ranges from 0 to 2π and sometimes it ranges from $-\pi$ to π. We will try stay consistent and use the latter in this report, but we may need use the other convention at times. The angle from the z-axis to the x-y plane is called the polar angle, denoted by θ, and runs from 0 to π. We are interested in the standard 3-momentum of particles that we track in the detector, which we call $\vec{p} = (p_x, p_y, p_z)$, but we also define the transverse momentum as

$$p_{\text{T}} = \sqrt{p_x^2 + p_y^2}. \tag{2.1}$$

We define the rapidity, often denoted as y, as

$$y = \frac{1}{2} \ln \left(\frac{E + p_z}{E - p_z} \right) \tag{2.2}$$

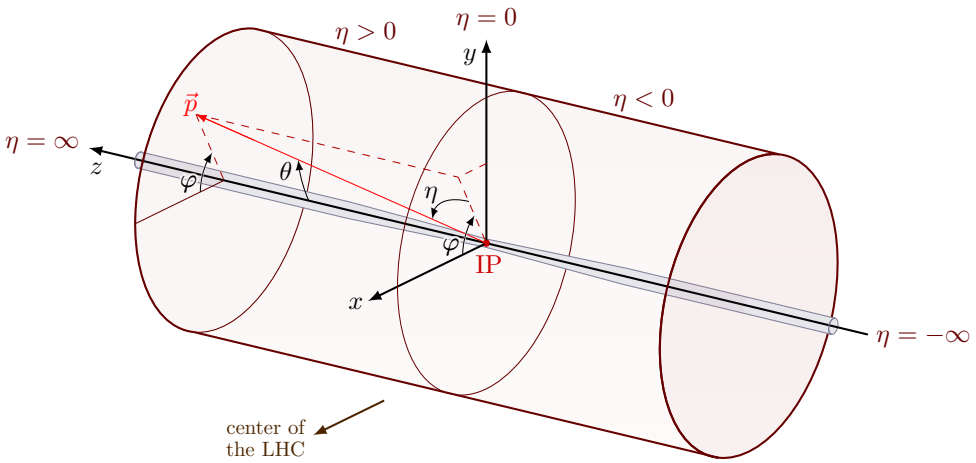

Figure 2.1: Modified cylindrical coordinate system used at the LHC [11].

where E is the total energy of the particle being considered and p_z is the momentum in the z direction [9]. This quantity is useful as differences in rapidity are Lorentz invariant for boosts along the z-axis. One issue, however, is that the energy of a particle is hard to measure, so we instead use pseudorapidity, denoted as η. Rapidity and pseudorapidity are equivalent for massless particles, and near equivalent for particles with total 3-momentum magnitude p much greater than their mass m. Pseudorapidity is much easier to measure as it is defined in [9] as

$$\eta = -\ln \tan \frac{\theta}{2}. \tag{2.3}$$

From figure 2.1 we see that for z positive, η is also positive, and similarly for z negative. Confusingly, we define the "forward region" of the ALICE detector as the region for which z, and thus η, are negative. The forward region is where our interest lies.

The four coordinates that we use most often are z, φ, p_{T}, and η.

2.2 ALICE Run 3

In 2018, the LHC shut down for what was called Long Shutdown 2 (LS2). During this time, the ALICE experiment was being prepared for Run 3, where it will be taking data at much higher luminosities and much higher energies than before, from 2022 until 2025 [3]. Figure 2.2 shows the detector configuration for Run 3. The intent of these upgrades was in large part to prepare ALICE for a higher luminosity of collisions in both Pb-Pb and p-p cases.

Part of the upgrades for Run 3, the details of which can be found in [3], were a whole new Inner Tracking System and a brand new detector called the Muon Forward Tracker. These detectors are both silicon-based and their primary purpose is tracking particles and determining the collision vertex, which is the best estimation of where the collision that resulted in these particles happened.

The readout electronics for many detectors were also upgraded to allow for continuous readout where necessary. The MCH also had its readout and front-end electronics upgraded but will still work on a triggered readout system.

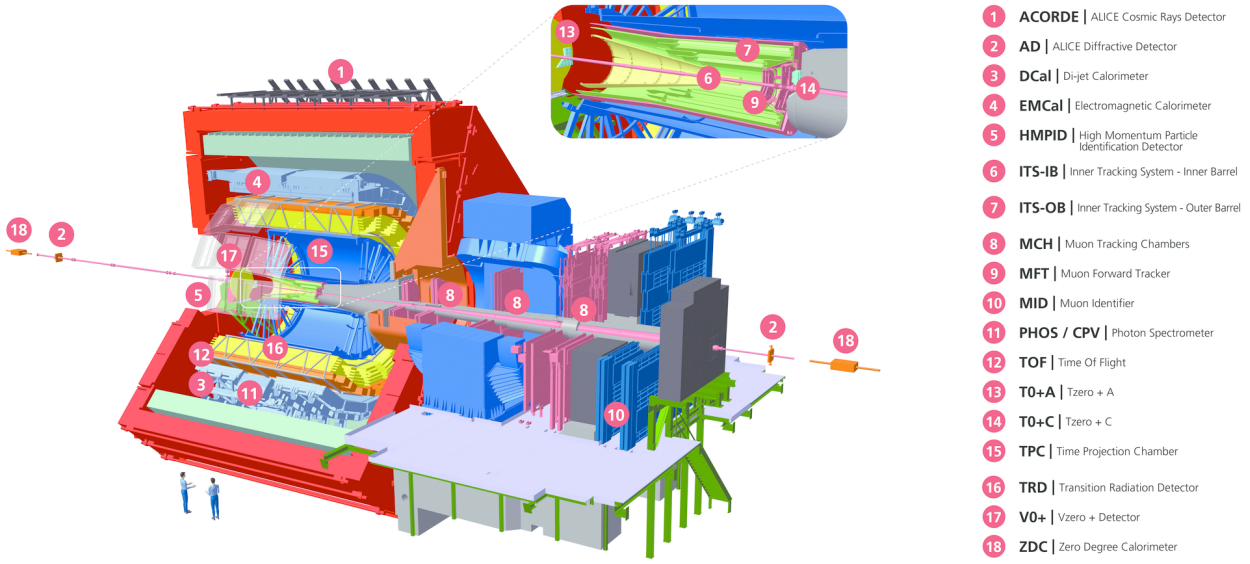

Figure 2.2: Schematic view of the ALICE detector setup for Run 3 of the LHC [1]. Note here that the MCH is shown separate from the MID, which act as the triggering mechanism for the MCH. For the purposes of this report, the MID will be considered part of the MCH.

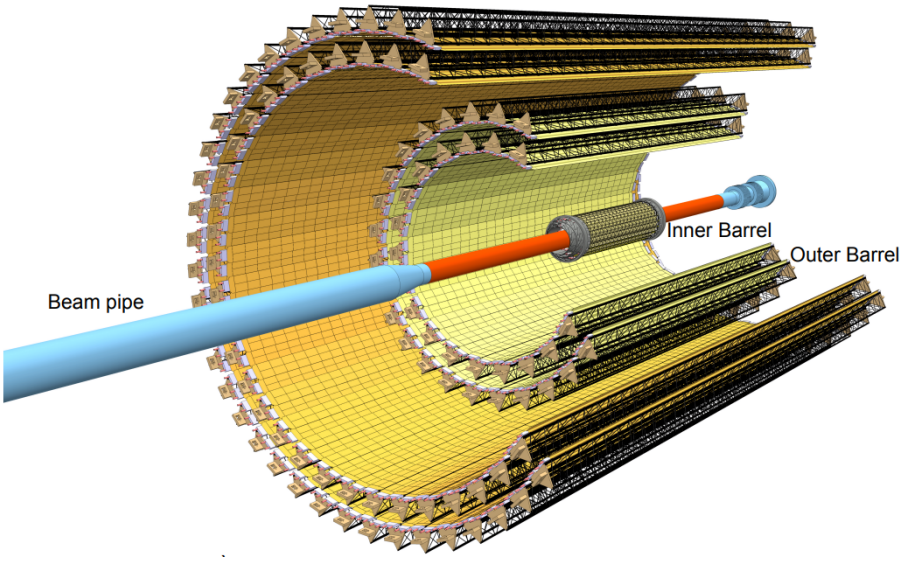

Figure 2.3: Schematic view of the Inner Tracking System [2]. Note the thinner beam pipe and extremely close Inner Barrel.

2.3 The Inner Tracking System

The Inner Tracking System (ITS) sits in the main barrel of ALICE, as seen in figure 2.2, and covers the range $|\eta| < 1.22$ [2]. For Run 3 it has been upgraded significantly by replacing the old detector with a new layout and new pixel detector technology, leading to an improvement in track position resolution at the primary vertex of a factor of 3 or greater [2]. The ITS's main purpose is

to track the particles resulting from the collisions and determine the position of the primary vertex of collisions. It also serves to "reconstruct secondary vertices, track and identify particles with low momentum, and improve the momentum and angle resolution for particles reconstructed by the Time Projection Chamber (TPC)" [8].

The new ITS consists of 7 layers of pixel detectors; 3 in the "Inner Barrel" and 4 in the "Outer Barrel". The pixel detectors used are 0.18 µm CMOS chips from TowerJazz. When a charged particle passes through the silicon in the active volume, it liberates the charge carriers in the material, which then collect on electrodes connected to the silicon, telling the detector that a particle has been detected. The fine segmentation of the detectors also allows the detector to determine the point at which the particle hit the detector, up to a resolution of 4 µm in both the $r\varphi$ and z directions [2].

The innermost layer sits at a radius of only 22.4 mm from the IP thanks to a reduction in beam pipe radius for Run 3 and the outermost layer sits at a radius of 391.8 mm from the IP. Figure 2.3 shows the layout more clearly.

2.4 The Muon Spectrometer

The MCH sits in the forward region of ALICE, as seen in figure 2.2, and covers $-4 < \eta < -2.5$. It is designed to study heavy quark resonances through their single- and di-muon decay channels.

As is shown in figure 2.4, it is composed of a hadronic absorber, 5 tracking chambers, a dipole magnet, and finally the 2 trigger chambers. The MFT is often also considered part of the MCH but is not shown in figure 2.4. The absorber, made of carbon and concrete, serves to filter out all particles travelling towards the MCH that are not muons as muons interact very rarely with matter. The 5 tracking chambers track the path of the muons before, during, and after they are deflected by the dipole magnet. These tracking chambers used to also perform tracking and vertexing in Run 1 and Run 2, but not with great accuracy as they sit behind the hadronic absorber. In order to remedy this, as well as deal with the increase in luminosity and energy of the LHC in Run 3, the MFT was added. Lastly, the trigger system consists of two large gas chamber detectors which trigger the MCH to take data only when a muon is detected.

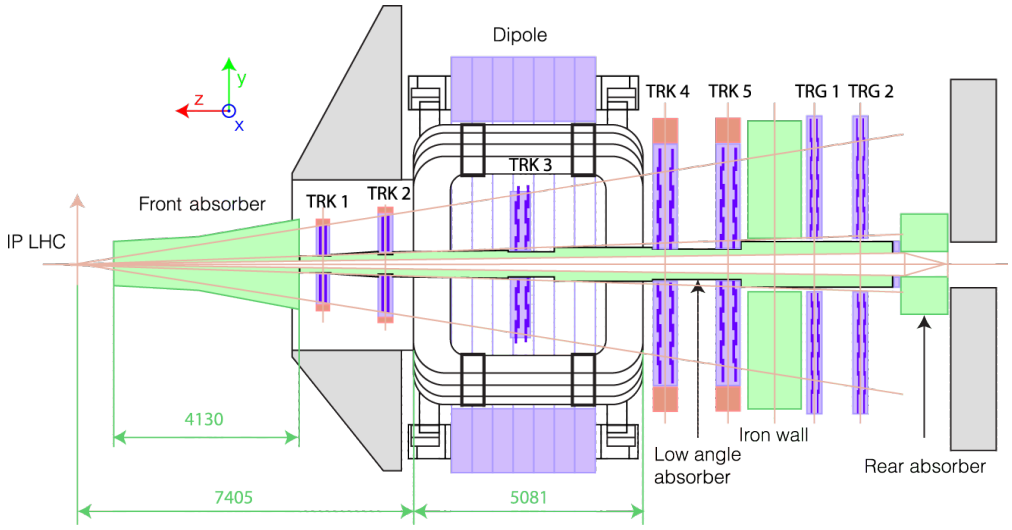

Figure 2.4: Diagram of the layout of the Muon Spectrometer [10]. Muons pass through the absorber, are deflected by the dipole magnet, and hit the trigger chambers at the back.

2.5 The Muon Forward Tracker

The MFT is a brand new detector added to ALICE for Run 3. It serves as a tracking detector for the MCH and covers the range $-3.6 < \eta < -2.45$. The MFT was made in conjunction with the ITS and uses precisely the same CMOS pixel sensors but in a conical configuration surrounding the beam pipe to better suit the geometry of the problem. Due to the MFT being placed in front of the absorber, it detects a lot more particles than make it through to the MCH, allowing it to be much better at finding the primary vertex of collisions.

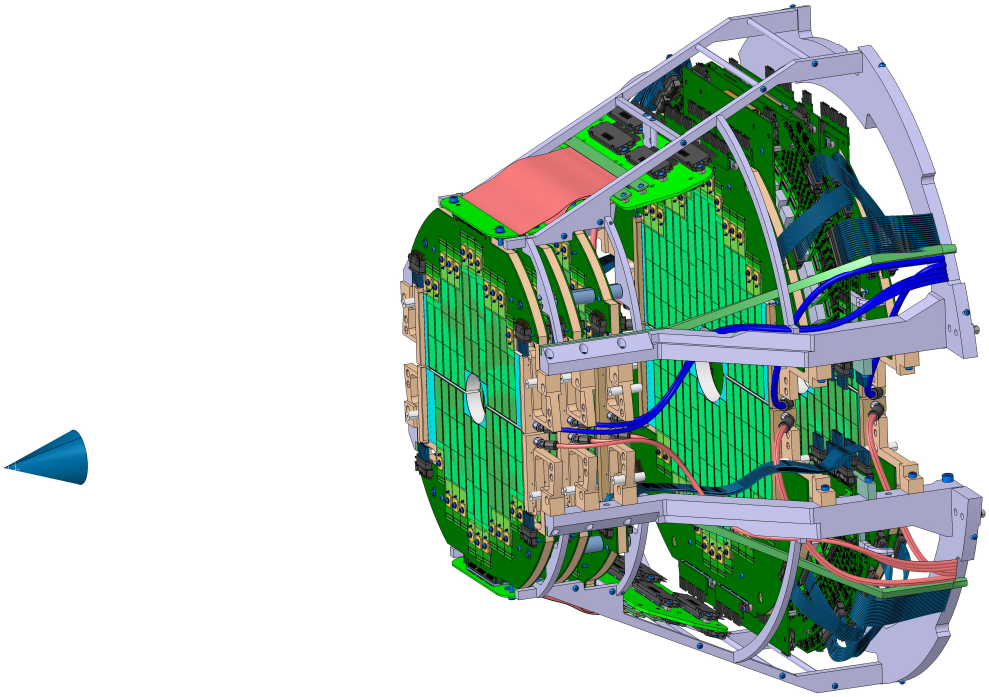

Figure 2.5: Schematic view of the Muon Forward Tracker [4]. The small cone on the left shows the IP.

The MFT Technical Design Report [12] gives all the details on the construction and implementation of the MFT, but the points relevant to this report will be discussed here.

The MFT is made up of 5 disks, each made of 2 half-disks. Each half disk has two planes of detectors, one on the front and one on the back. The disks sit at z-positions -46.0, -49.3, -53.1, -68.7, and -76.8 cm respectively and each disk is 1.4 cm thick, leading to detector planes at ± 0.7 cm from each of those positions.

2.6 The Online-Offline Analysis Framework

With the increased interaction rate expected for Run 3, a new system for real-time processing, as well as offline analysis, needed to be constructed [3]. The Online-Offline (O2) framework was developed for this purpose. The "Online" portion of the framework is the real-time processing, where continuously captured data from ALICE is split into 10 ms chunks, called timeframes. These timeframes are then later processed, or "reconstructed" into Event Summary Data (ESD) and then Analysis Object Data (AOD) files—this is the "Offline" part—which contain only the data of interest. Different passes of reconstruction can be done for purposes and are often iterated upon if

something was missed in a previous pass. Because of this, the same data can often look different between reconstruction passes. The AOD files can then be analysed—also Offline—through an "Analysis Task", written in C++ and ROOT.

We distinguish between data taken in Run 3 and data taken in Run 1 and 2 by calling Run 3 AOD's "AO2D" files, and Run 1 and 2 just "AOD" files. The data is stored on the `alimonitor` system, requiring a certificate to access, which is obtained by joining the ALICE collaboration. Access to all data and most analysis tools used in this report is restricted behind this wall.

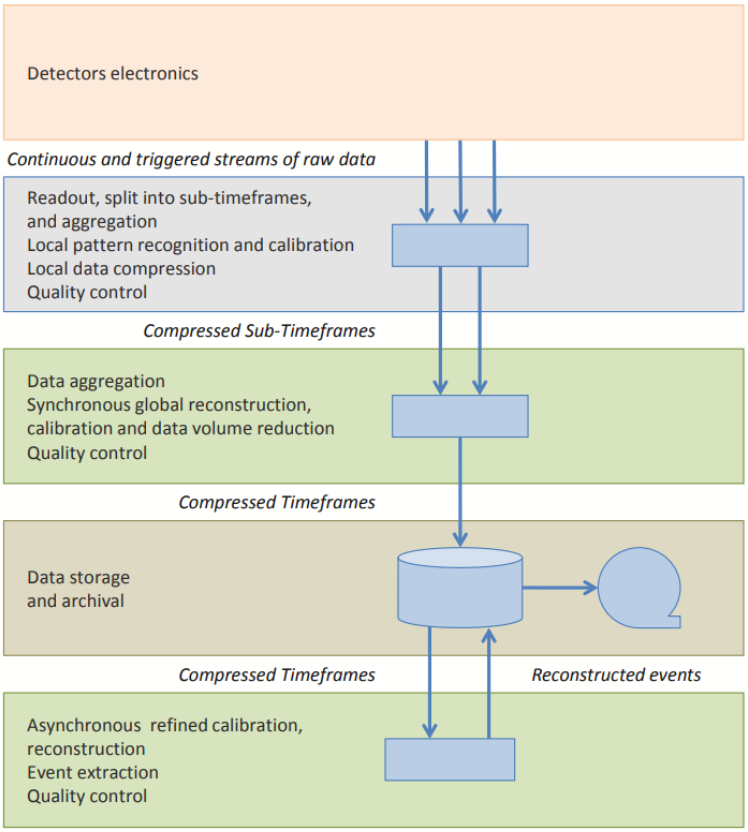

Figure 2.6: Functional flow of the O2 framework [6].

The focus of the upgraded analysis framework was to reduce disk space usage when processing and analysing, as well as making sure all analysis takes advantage of all processing power available to it at all times. Figure 2.6 shows the general flow of data in the O2 processing pipeline from particle hits in detectors until usable data in the form of AOD files.

3 Writing an Analysis Task

Writing an analysis task in O2 is structured quite strictly, due to it needing to be run by O2. We will outline the steps needed to go from reconstructed data to histograms of, in our case, kinematic variables. It must be noted that learning how to do this, and how to deal with the idiosyncrasies of the O2 software, were what took up the majority of the time spent on this project. It is written to do one thing very very well, but unfortunately that comes with the side-effect of it being extremely picky about the conditions in which the software will actually work.

3.1 AOD Structure

The data that we use in our analysis comes in the form of Analysis Object Data. These are in the form of ROOT files, which are based on a tree structure. In the tree are a number of tables corresponding to reconstructed tracks and collisions, among a few others. We call these entries or rows. For each of these tables, there are a number of reconstructed variables associated with each entry, such as their p_{T}, η, or φ, which we can access. There are 4 types of variables:

- Static variables are saved to disk during the reconstruction process and are available at any time. φ is a static variable.
- Dynamic variables are calculated on demand, using static variables as input. η is a dynamic variable, calculated from θ.
- Index variables point from one table to another, such as from a track to its associated collision.
- Expression variables no clue

All variables associated with a track, for example, could be included in a single table. However, since we often only need a few of the variables, the tables are split up into sections that contain variables often used together. If needed, these tables can be joined together when doing analysis, using `o2::soa::Join<Table1, Table2>`. Importantly, only tables which correspond row-to-row and have the same number of rows can be joined in this way.

3.2 Analysis Task Structure

Analysis tasks are written in C++ and need to be structured in a specific way for O2 to use them properly. Each task is written as a `struct` object which is then called at the end of the task. Below is an outline of what is needed for a task.

```
#include "Framework/runDataProcessing.h"
#include "Framework/AnalysisTask.h"

struct MyTask {
    // Define things here, such as histogram registries, filters for data, or new tables

    void init{o2::framework::InitContext&} {
        // Here we initialise histograms and other things used in the analysis
    };

    void process(aod::Collisions const& collisions, aod::Tracks const& tracks) {
        // Here we can do any processing that we need, calculating things etc, and then fill the
↪   histograms we defined earlier
    };
};

// This is what O2 looks at to run the task
WorkflowSpec defineDataProcessing(ConfigContext const& cfgc)
{
  return WorkflowSpec{
    adaptAnalysisTask<MyTask>(cfgc),
  };
}

```

With the task written, it then needs to be compiled and added to O2 so that it can be run. O2Physics has a number of analysis tasks written by people at ALICE which get compiled automatically. These are sorted into physics working groups such as Heavy Flavour (PWGHF) and Electromagnetism (PWGEM). If we want to add our own task to O2Physics, we need to create our own folder with the same structure as the working groups. Below shows the structure of the file system.

```
alice
├── alidist
├── O2
├── O2Physics
│   ├── Functional Things
│   ├── ...
│   ├── myTasks
│   │   ├── CMakeLists.txt
│   │   ├── Tasks
│   │   └── myTask1.cxx
│   │       ├── CMakeLists.txt
│   │       ├── myTask2.cxx
│   │       └── ...
│   ├── Other Working Groups
│   ├── ...
│   └── CMakeLists.txt
└── sw
```

Here `alidist` is the git repository that handles the versioning of O2 and O2Physics. O2 is what handles the backend of the analysis framework, compiling the tasks written in O2Physics. The `CMakeLists.txt` files are needed at every level of the O2Physics structure to tell O2 what to compile and which commands to use to refer to things.

See below an example of what the `CMakeLists.txt` file in the `myTasks` folder would look like.

```
1 add_subdirectory(Tasks)   # Ensures O2 can see the Tasks folder
2 o2physics_add_dpl_workflow(my-task1          # The command assigned to the task. Note only
  ↪ lowercase letters, hyphens, and numbers are allowed
3                  SOURCES myTask1.cxx   # The source file for the task
4                  PUBLIC_LINK_LIBRARIES O2::Framework
5                  COMPONENT_NAME Analysis)
```

3.3 Compiling O2

O2 is built on your system using `aliBuild` [7]. It prefers to be built on UNIX systems and requires at least 8 GB of RAM, preferably more. We used both O2 and O2Physics as we wanted to create and run analysis tasks.

Once O2 is built (those four words are doing some *very* heavy lifting) we can enter the O2Physics environment with `alienv enter O2Physics` and this will place us in a new terminal shell. The magic of O2 is that it compiles all the analysis tasks in O2Physics, as well as tools for simulation and the like, such that everything can be done by running commands in that shell. Before running our own tasks, however, we need to tell O2 to build our tasks into O2Physics. To do this we use `ninja`.

If we have our tasks written and files structured as shown in section 3.2, we can enter an O2Physics environment and load ninja alongside using `alienv enter O2Physics ninja/latest`. In the shell we can then navigate to the build of O2Physics, which should be in `alice/sw/BUILD/O2Physics-latest/O2Physics` and run `ninja install myTasks/all` which will rebuild only those parts of O2.

Ninja is the quick way of doing things so every now and then it's a good idea to rebuild O2 and O2Physics entirely, pulling the latest release. We can do this while also making sure our own tasks don't get overwritten using the following steps. We first need to make sure that our work does not get overwritten, which is done by making sure git knows they're there with `git add path/to/files` and then committing with `git commit -m "commit message"`. With that done, we can systematically update `alidist`, `O2`, and `O2Physics` by navigating to each and running `git pull --rebase`. Finally we can rebuild by navigating to `alice` and running `aliBuild build O2Physics --defaults o2`. This will take a few hours to complete (if it's even successful) and then will be able to be used again.

3.4 Running a Task with O2

Once we have our tasks created and built in O2Physics, we can then run them. For our purposes, the only commands we need to know are how to run a task and the options that come along with that. All analysis tasks in O2Physics get assigned a unique command that can be used to run that task. They all begin with `o2-analysis...` followed by the name assigned to it in the relevant `CMakeLists.txt` file as shown in section 3.2. In the case of that task, we would run it with `o2-analysis-my-task1`.

Most analysis tasks are run on `AOD.root` or `AO2D.root` files so in order to tell the task which file to use, we use the flag `--aod-file AO2D.root`. We could also supply a list of files in a text file and use `--aod-file @AO2D_list.txt` where `AO2D_list.txt` contains the path to the files we want to run on.

Lastly on the topic of running tasks is piping the output of one task into another. Often we want to run multiple tasks in succession on the same data, feeding the output of one into another. To do this, we simply use the pipe symbol `|` between the tasks: `o2-analysis-trackselection | o2-analysis-ud-mytask --aod-file AO2D.root`. Here the ordering of the tasks doesn't matter as the input and output format of a task is known before it runs, so O2 does some quick thinking to arrange the workflow such that the tasks get fed the correct format of data.

For the most part, the output of an analysis task is either a `AnalysisResults.root` file or a `QAResults.root` file, with the former being the most common. This output type is chosen when defining the histogram registry.

References

[1] *3D ALICE Schematic RUN3 - with Description — ALICE Figure*. URL: https://alice-figure.web.cern.ch/node/11220 (visited on 07/15/2022).

[2] B Abelev et al. *Technical Design Report for the Upgrade of the ALICE Inner Tracking System*. Tech. rep. Nov. 2013. DOI: 10.1088/0954-3899/41/8/087002. URL: https://cds.cern.ch/record/1625842.

[3] B Abelev et al. *Upgrade of the ALICE Experiment: Letter of Intent*. Tech. rep. Geneva: CERN, Aug. 2012. DOI: 10.1088/0954-3899/41/8/087001. URL: https://cds.cern.ch/record/1475243.

[4] MFT Collaboration ALICE. *ALICE Muon Forward Tracker (MFT)*. CERN Bulletin. Number: OPEN-PHO-EXP-2020-004. Dec. 25, 2020. URL: https://cds.cern.ch/record/2748310 (visited on 07/17/2022).

[5] *Letter of Intent for A Large Ion Collider Experiment [ALICE]*. Tech. rep. Geneva: CERN, 1993. URL: https://cds.cern.ch/record/290825.

[6] P Buncic, M Krzewicki, and P Vande Vyvre. *Technical Design Report for the Upgrade of the Online-Offline Computing System*. Tech. rep. Apr. 2015. URL: https://cds.cern.ch/record/2011297.

[7] *Installation via alibuild · ALICE Analysis Tutorial*. URL: https://alice-doc.github.io/alice-analysis-tutorial/building/custom.html (visited on 09/26/2022).

[8] *ITS Info Page*. URL: https://alice-collaboration.web.cern.ch/menu_proj_items/its (visited on 07/18/2022).

[9] Deepak Kar. *Experimental Particle Physics*. 2053-2563. IOP Publishing, 2019. ISBN: 978-0-7503-2112-9. DOI: 10.1088/2053-2563/ab1be6. URL: https://dx.doi.org/10.1088/2053-2563/ab1be6.

[10] *Muon Spectrometer - ALICE Collaboration*. URL: https://alice-collaboration.web.cern.ch/menu_proj_items/Muon-Spect (visited on 07/17/2022).

[11] Izaak Neutelings. *CMS coordinate system*. URL: https://tikz.net/axis3d_cms/ (visited on 07/14/2022).

[12] *Technical Design Report for the Muon Forward Tracker*. Tech. rep. Jan. 2015. URL: https://cds.cern.ch/record/1981898.

Appendix

A Downloading data from the GRID

```
# The directory in alimonitor where you want to get the data from. Should contain a load of numbered directories
# For example, /alice/data/2021/OCT/505548/AOD. Note that it's AOD not AO2D
sourceMotherDir=/alice/data/path/to/AOD
nFiles=$1

# This is the directory on your local machine where you want to store your data
targetMotherDir=/path/to/alice/data/${sourceMotherDir}
mkdir -p ${targetMotherDir}

for ((i=1; i<=nFiles; i++))
do
    if (( i < 10 )); then
        pref=00
    fi
    if (( i > 9 )); then
        pref=0
    fi
    iDir=${pref}${i}
    iSourceDir=${sourceMotherDir}/${iDir}
    iTargetDir=${targetMotherDir}/${iDir}
    mkdir -p ${iTargetDir}

    echo "copying from ${iSourceDir} to ${iTargetDir}"

    alien_cp -retry 5 ${iSourceDir}/AO2D.root file://${iTargetDir}/AO2D.root
done
```