LHCb Starterkit Notes

* Typically the ‘stable’ particles that have a long enough lifetime to traverse the full detector are:

1. Charged pions
2. Charged kaons
3. Protons
4. Electrons
5. Muons
6. Photons
7. Deuterons

* Many properties of these particles are intrinsic, but can be inferred based on the signals they create in the detector as they traverse through it

**Reconstruction**

* The process of inferring the properties of the particles produced in collisions
* For a given proton-proton collision, the reconstruction of all the objects is performed simultaneously
* Output of reconstruction is a set of tracks, each of which represents the trajectory of a particle through the detector. One can assign a momentum estimate to a track based on its curvature, as induced by the LHCb dipole magnet. One can infer the charge of the particle as the polarity of the magnet, as well as its field strength are known
* A vertex represents some source of particle production in space. Primary vertices are those that are created in the interaction region upon the collision of proton-proton bunches.

**Building Decay Candidates**

* All reconstructed objects contain contributions from detector effects (e.g. resolution)
* Contributions from ‘ghost’ tracks could also contribute to this. Hence, one must infer properties statistically, based on ensembles of objects and events
* **SEE EXAMPLES IN LHCB STARTERKIT**

**LHCb Data Flow**

* LHCb records approximately 1 TB of data every second. This is too much to store, and hence, one needs to *filter* the data and keep only the interesting events
* The selection process must be done quickly and is required to be sufficiently complex to extract the relevant events
* The data flow is as follows:
  + **Data from the detector is filtered to the detector, which consists of the L0 (hardware), and the High-Level trigger (HLT), which is implemented in software. The HLT is designed in Moore**
  + **The triggered, raw data are then reconstructed to transform the detector hits into objects such as tracks and clusters. This is done by Brunel, and the objects are stored into an output file in ‘DST’ format**
  + **Data is then filtered further through selections known as stripping, controlled by the DaVinci application. The output files are then grouped into streams that contain similar selections in order to save disk space and to promote accessibility for analysts**
  + **A lot of simulated events are also produced (Monte Carlo data), and this is processed in a very similar way to real data. The similarity is very beneficial as the simulated data is subject to same deficiencies as processing of real data.**
  + **The proton-proton collisions and decay of resulting particles are controlled by the Gauss application**
  + **Simulated hits made in the virtual detector are converted to signals that mimic the real detector by the Boole application. The output of Boole matches the output of the actual detector closely. This way, the simulated data can then be passed through the usual processing chain (i.e. trigger, reconstruction, etc.)**
* Only DaVinci application is run by users, with everything else being run centrally

**Changes to the Data Flow in Run 2**

* The selection of candidates made in the second stage of the high level trigger, HLT2, is saved to disk and used directly by analysts, without a further offline reconstruction (Turbo stream)
* Turbo stream contains only the candidates that were reconstructed in the trigger.
* Turbo stream runs in parallel with the regular data flow

**Analysis Flow and Analysis Preservation**

* After preselection of data in stripping, sprucing, or triggering, one can produce ROOT files containing *ntuples* running the DaVinci application.
* Ntuple is a data structure stored within a ROOT file which contains information about events or candidates in the data sample, such as the candidate mass or trigger decision flags.
* After getting the ntuples, the user develops new analysis code, or expands existing code. Analysis code is usually based on HEP software tools or general data analysis tools (e.g. numpy, pandas, etc.)
* A typical analysis consists of the following steps:
  + **Defining and applying selections, including kinematic, particle identification, MVA, signal/background separations by fitting**
  + **Computing and applying calibrations and corrections.**
  + **Computing efficiencies, acceptances and measuring detector resolution effects**
* Snakemake workflow management system allows you to write a pipeline for analysis that will determine the hierarchy of the steps and changes in input.

**Introduction to LHCb Software**

* Software is based on Gaudi framework. The key Gaudi concepts that one must be familiar with include:
  + **Event Loop:** Because the events are almost completely independent of each other, it makes sense to process them one by one. Gaudi enables this to be performed
  + **Transient Event Store:** A single event contains a lot of different data objects (i.e. pcles, vertices, tracks, hits). These are organised by the TES.
  + **Algorithm:** A C++ class that can be inserted into the EventLoop. These allow one to perform a certain function for each event (like filtering according to trigger decision, reconstructing particles)
  + **Tools:** Often, algorithms will want to make use of a common function (i.e. vertex fitting, calculating distances, etc.). These can be implemented as tools and shared between algorithms
  + **Options:** To make all of this configurable, Gaudi allows one to set properties of algorithms and tools from a Python script, known as an *option* file, within which one can specify the order in which algorithms are run, and also set their properties.

**The Simulation Framework**

***Gauss***

* Gauss is the LHCb simulation framework which manages the creation of simulated events by interfacing to multiple external applications. An event is created via the following procedure:
  + **ProductionTool (e.g. Pythia, GenXicc, etc.) generates an event with the required signal particle. Either by generating minimum bias events until a matching particle is found, or by ensuring one is produced in every event. The resulting event is comprised of either stable or unstable particles which are known to either EvtGen or Geant4 and can be decayed**
  + **The signal particle is decayed using the DecayTool (EvtGen) to the desired final state, all remaining unstable particles are decayed independently**
  + **The signal and its decay products might be required to pass generator level cuts implemented as a CutTool**
  + **Particles are transported through the detector simulation**