1. I’m going to tell you about the Rich Alignment.
2. I will start with explaing the general alignment strategy and then show how it was implemented for Run 2 and how it has performed so far. At the end I will give a very small overview of the general RICH maintenance and operation.
3. So we have two ring imaging Cherenkov detectors at LHCb where the charged particles travel through the gas and emit Cherenkov photons. These are then reflected by a primary mirror onto a secondary mirror and then onto the detector planes here. In order to correctly reconstruct the Cherenkov angle under which the photons were emitted, the mirrors here have to be properly aligned - otherwise the PID might suffer.
4. When I talk about “misalignment” or “aligning the mirrors” I don’t actually mean that any of the mirrors are being physically moved. Misalignment means that the orientation of the mirrors in the conditions database is incorrect and when we align them we find out the correct orientation and update the database.   
   Here you can see what happens in case you have a misalignment: Here is a charged particle track that you want to projected onto the detector plane. If there is a misalignment of the primary mirror for example, then the predicted position will differ from the actual position.   
   Here is what this would look like on the detector plane: here are the photon hits that form the Cherenkov ring and the actual track position right here in the middle. From this you can see that for the real track position the Cherenkov angle is independent of where you are on the ring but for the incorrectly projected track position the Cherenkov angle will vary with the azimuthal angle.   
   In order to find the misalignment, what we do now is to select a track whose Cherenkov angle we know, and then look at the measured minus the predicted angle as a function of the azimuthal angle. Here you can see the histograms for an example mirror pair in RICH2. We then fit a sinusoidal to the 2D histogram and get the factors in front of the cosine and sine which correspond to the misalignment on the detector plane - for a perfectly aligned mirror-pair you get a straight line and the misalignment constants are 0.
5. Now for each mirror pair you get one of these histograms which give you the misalignment in y and z on the detector plane. What we need though is the misalignment of the individual primary and secondary mirror on both y and z, meaning that we have twice as many unknowns as equations.   
   For RICH1 this problem is easily solved, since one given secondary mirror only receives photons from one primary mirror. So what we do is to keep the primary mirrors fixed and only apply alignments to the secondary mirrors.
6. For RICH2 this is a bit more complicated since one secondary mirror can receive photons from several primary mirrors as you can see here at the example of the secondary mirror 10 that receives photons from 6 different primary mirrors. SO for RICH2 the strategy is to align all mirrors with respect to one mirror.   
   That means we start at primary mirror 12 and keep that one fixed, then we align the other mirrors along these arrows. For example we start with the mirror-pair where 12 is the primary mirror and the secondary mirror 9. Since 12 is fixed we only need to align the secondary mirror 9. From there we go to the next mirror-pair where 9 is the secondary mirror and for example primary mirror 17. We know 9’s misalignment so we can align the primary mirror 17. And then so on along this chart.
7. Here is an overview of the entire alignment procedure. So we start off with some data and the default conditions database. We use those to reconstruct high energy tracks where we can assume that the Cherenkov angle is saturated.   
   In the next step we fill the Cherenkov angle vs the azimuthal angle for each mirror-pair and then fit those to find the misalignment on the detector plane.   
   From there we get the individual mirror misalignments and update the database.
8. Then we compare the initial database to the new one and if any mirror has moved for more than 0.1 mrad
9. we replace the initial database with the new one and do a next iteration.
10. If at the end of one of these iterations the alignment of none of the has moved by mire than 0.1 mrad
11. We consider the alignment to be converged and then we can use the last database for the conditions database.
12. Now we come to how this all was implemented in run 2.
13. In Run 1 the alignment was done offline after the data taking with ganga and then applied to the reconstruction at the end of year reprocessing.  
    In Run 2 the entire procedure is online as Giulio just showed. At the beginning of every fill the HLT1 lines for the Rich alignment collect events and once there are enough the alignment is started. The Rich Alignment will run in monitoring mode only, which means that the alignment is run but the database is not automatically updated.   
    In order for all the nodes on the HLT farm to work together the entire thing is steered by a finite state machine, which I will tell you about in a bit.
14. But first let me tell you a bit about the HLT farm. The farm consists of one central node called the iterator and of about 1700 nodes called the analyzers.   
    The data from the HLT1 lines is equally distributed over the analyzers and all they do is to reconstruct the data that is stored on them with a given conditions database and fill the 2D histograms.  
    The central does all the rest. First the iterator provides the database that the farm nodes use to reconstruct the events. After the reconstruction the iterator collects the histograms and performs the 2D fits and then produces the new database. It then decides weather or not to continue and provides the new database to the analyzers.  
    Now the iterator doesn’t know about the analyzers, it just knows where to put files and where to collect them. The same counts for the analyzers, they don’t even know about each other. In order for this iterative process to work we use the principle of a finite state machine.
15. Finite state machine means that each member can only be in one of a finite number of states. A state can be for example “running” or “paused”. For the alignment we have the iterator on one side and all the analyzers on the other side and in the middle we have the run control to coordinate. Here is what the sequence looks like for the mirror alignment: First the run control will give the command to configure to both the iterator and the analyzers. The iterator will for example get the default conditions database and put it where the analyzers can pick it up. For the analyzers ‘configuring’ means they will read in and set up the options for the Brunel reconstruction.  
    Once they are done configuring the iterator and the analyzers will both go into the state ‘ready’.  
    The run control will wait until the iterator and the analyzers are all in state ‘ready’ and then give them the command ‘start’ which will set them into the state ‘running’. During this time the analyzers will reconstruct the events and fill the histograms. Each analyzer will go into state ‘paused’ when finished processing the events and when all analyzers are paused the run control will put them into state ready. At the same time the run control will send the command ‘pause’ to the iterator. The iterator now does all his work while in state paused, meaning fitting the histograms and getting the mirror alignments and updating the database. If it decides that the alignment is not converged yet it will place the new database where the analyzers can pick it up and go into state ‘running’. The run control will see this and send the command ‘start’ to all analyzers that will then again reconstruct the events using the new database and fill the histograms and so on.