# Table of Contents

Discussion on the Muon Event Model

- TES containers per hit type
- Relations
- Visualisation
- Use cases
Discussion on the Muon Event Model

To help in following the discussion, the speaker name is recalled before each paragraph, with the following convention:

MC= Marco Cattaneo; AS= Alessia Satta; CRJ= Chris Jones; OL= Olivier Leroy; GC= Gloria Corti; EP= Erica Polycarpo;

MC: This class is unnecessary, could be superseded by a modified MCHit class (see similar discussion for MCVeloHit in Velo pages)

AS: Such idea is already contained in the documentation of the Muon event model.

CRJ: Just a comment, but with some experience from the RICH, I think we should be careful in making sure using a common MCHit class really makes sense and simplifies things. I.e does every data member make sense for all sub-dets, and are they no detector specific fields that are needed (such as flags etc). If I understand the proposal correctly, then it is to store these flags as part of the key? Does this complicated using these flags? Will the user have to write custom accessor/setter methods or can GOD still do this for us?

GC: I have some comments regarding the MCMuonHit and how the specific detector information is filled in Gauss.

The information about the chamber and gap numbers does NOT come for free in Geant4 where it is not packed in a single word: the packing is explicitly done in the MCMuonHit class. While processing the hit the information available regarding the geometry in Geant4 is what is available in a G4Step: the position and the volume at the beginning and at the end of the step. At the beginning is the entry to the sensitive volume and at end it is in reality the information at the entry of the next volume. The information readily available are the coordinate at the beginning (end) of the step in the global reference system (LHCb) and the name (string) of the volume in which the current step is occurring, as well as the copy number of the volume. Given the current hierarchical structure of the muon system and what is defined as the sensitive volume, the copy number provided has no relation with the gap or chamber number: to obtain the number of the gap one has to step up in the geometry history tree and then move one additional step up to obtain the number of the chamber. In addition the chamber copy number is an arbitrary consecutive number (starting from 1000) and as positioned by Geant4: there is some awkward hard-coded subtraction to then obtain the chamber number:

```cpp
if(CurrentChamNumber<1012) {
    ChamNumber=CurrentChamNumber - 999;
}
if(CurrentChamNumber>1011 && CurrentChamNumber<1036) {
    ChamNumber=CurrentChamNumber - 1011;
}
...
```

This implies not only relies on the geometry structure but also assumes all chambers to be always present in the simulation and arbitrary chambers cannot be removed. The numbering depends on the order of the conversion from the XML to the detector description to the Geant4 detector geometry: hence on the order of the lines in the XML, and the order of the looping over the chambers in the detector converters. >From my point of view this is very dangerous since it depends on code not necessarily under our control: a different behaviour of deprecated Geant4 code we were using to get the copy numbers while processing the hits lead to the (in)famous muon bug last spring. One can force the copy number in Geant4 to be a given value by forcing it also in our XML detector description (still the navigation up the detector history tree is unavoidable) but to have a unique numbering the forcing of a unique copy/replica number must be done for all volume at the same level, including supporting structures. The dependency of the knowledge of the
geometry structure in the code to retrieve the information cannot be avoided: hence this code can only be under the responsibility of the muon system. The only way to guarantee that the geometry information can be obtained without relying on Geant4 classes and knowledge of the geometrical tree structure is to get it from the DetectorElement but then I don't see why the MCHit could not be used and then the chamber/gap number handled as for the other detectors either externally by the DetecetorElement or via a physical volume identifier in the MCHit as currently under discussion in Velo and ST.

CRJ: I fully agree with your concerns regarding the muon ID detector number in Gauss, and in particular the implicit detector knowledge that this implies. I strongly support your view that the muon system should not use such a construction, and use the detector elements instead, as is done by the other detectors.

AS: I think I miss something. Let's assume that we remove the piece of code that in Geant4 identify the chamber and the gap in which the hit is produced. Then we should evaluate the chamber and gap at the end of gauss or at the beginning of boole searching the gap using the hit coordinate. Is it correct? In such case the search of volume in which the hit is contained would be quite time consuming. It was just for such reason that the volume id was evaluated in geant4. let me know.

CRJ: I don't see why the location of the correct detector element from the hit coordinate should be any more time consuming for the MUONs that for any other sub-detectors, which do things this way? The timing of Boole proves I think that it is not excessively slow and in any case if I understand Gloria's argument correct the calculations in Gauss are not for free anyway, and already take some time. Do you have some evidence to support the claim that doing things like the other sub-detectors in Boole really would be that much slower?

Note that removing this information from Gauss has benefits from the persistency side, as it allows the removal of the bit-packed word from MCMuonHit, reducing the persistent size, and also potentially allowing the muons to drop the MCMuonHit class entirely and just use the MCHit class directly.

AS: The muon detector has about 6000 sensitive volumes. I do not the corresponding numbers from OT or IT but I believe that the do not instantiate individua tubes or strips., For sure in Velo the sensitive volumes are very few.

In the past, before G4, we have to calculate the chamber and gap during the digitization and it was the largest time consuming part even if David H. used a lot of tricks to speed up the code btw such tricks are not compatible with the misalignment so ..... I do not think the the calculation of chamber and gap ID in gauss is time consuming. Am I right Gloria? Even if I agree with Gloria that such code is not the final one I propose o keep the existing code for MCMuonHit. In the near future a better implementation in G4 can be found. In case of failures we can calculate the chamber and gap id and the end of Gauss before storing the hits.

MC: I think this conversation is going round in circles. Both Gloria and I do not think that time spent in Gauss or in Boole should be an issue - the total time (Gauss+Boole) is essentially unchanged. If this is the only reason for keeping MCMuonHit, then it should be dropped.

We should concentrate on technical issues - is it desirable to have a Gauss output that depends on detector elements, is the code simplified by doing the conversion in Boole or at the end of Gauss, is there an advantage in having a Gauss output that consists of just hits in sensitive areas without reference to detectors (e.g. the same Gauss input could be used for several alignments).

In any case, the time argument seems to be the only one that distinguishes the Muon detector from the others. If it is decided that it is useful to store the detector IDs in the Gauss output, then it should be done for all sub-detectors. Conversely, if it is decided to do the correspondence in Boole, it should again be done for all sub-detectors. This makes the code much more uniform, easier to read, easier to maintain.
Finally, I don't like the comment "in the near future a better implementation in G4 can be found". We are talking about our final event model, and an implementation that must be ready by the end of the year. We should build on the experience of the last few years, not hope for some bright ideas in some not well defined future.

AS: I do not understand. If you move the sensitive volume we cannot reuse the hits produced with a different alignment. If the detectors have large differences I do not see why the should be treated in the same way. For instance in the Rich they prefer to use a modified hit class without the exit point and I do not see the problem with that.

The present implementation in G4 was strongly suggeste by Witold. Stefano De Capua who wrote the code followed his directives without a good knowledge of G4 and Gauss. Now I believe that in LHCB much more experience in G4 is gained so probably a better solution could be find.

Modifing the G4/Gauss procedure to evaluate the detector ID is transparent to event model, so I think it can be modified and made more robust even after the event model review if the interface (the class) are kept the same.

MC: The whole point of the event model review is to look back at decisions made a few years ago when we had less experience, and see if we can do things better. Different sub-detectors chose to do things differently, this was good because now we can compare different implementations - and now is the time to determine the best way and implement it in the same way for everybody.

It is almost sure that all sub-detectors will have to change some small part of the event model. This is the whole point of the review - how can the event model evolve so that the code becomes simpler and more uniform.

20 TES containers per hit type

MC: This is an absurdity. Not only is the argument for their existence fallacious (navigation in the TES to find the right container by name is probably as time consuming as searching in a single container), but also it makes the code and integration more complicated (2 nested loops everywhere hits have to be searched, 20 lines in any options file that has to refer to these containers). Similar comment for the 5 MuonCoords containers. Comment on TSA: If I understand correctly, TSA uses the standard keyed containers, but builds specialised iterators for partitioning the contained objects.

AS: While I think it is not a problem in the software to manage the 20 containers I agreed already in the documentation that if required by the persistency the containers can be reduced to one using an alternative way to separate hits from the 20 regions of the detectors.

OL: Concerning the "20 TES container per hit type", how long do you think it will take to reduce the number of containers to one. Can it be done before Christmas?

It depends on the amount of all the modification to the event model that we must implement. My proposal is the following:

1) reduce to 1 container for hits and coords

2) keep the chamber and gap ID in MCMuonHit.

3) keep MuonDigit in the running of boole, convert them to raw buffer, store only the raw buffer. In Brunel convert back the raw buffer to digit that create the coords and fill the associator. It should be based on a linker between MuonTileID (digit) and MCParticle.
All such modifications are not trivial, but probably before end of year they can be implemented. In case many more modifications will be required I have serious concern about the time scale...

CRJ: I have some reservations on this proposal (2). To my mind there are two issues.

1. Is it appropriate for any detector to store detector information in the MCHits. This question is not isolated to the MUON and should be discussed by all. If the general consensus is that it is not appropriate, the MUONS should remove this information.

2. How the information is generated in Gauss should not imply any hard-coded assumptions on the geometry or XML numbering etc. This I think is mandatory, and not the case for the current muon code in Gauss. If LHCb does decide keeping this information is appropriate, then the code needs changing into something completely robust against detector changes. (I would also add that I would agree that if it is not possible to do this, then the answer to point one should be no).

MC: I completely agree with this analysis. Either the detector identifier in the MCHit is useful, in which case it should be implemented for all sub-detectors, or it is not, in which case it should be dropped also by the Muon detector.

AS: I believe that the silicon (Velo and TT and IT) requires the sensitive volume ID just as muon. I copied below a part of the VELO and ST review

VELO/ST review: All agreed identifying which material a hit is in by its x,y,z position is best avoided. Velo currently does this by adding an extra word with sensor ID. Suggestion to use instead key of keyed object to store a material identifier in some bits + hit number in other bits. The use of this information in the construction of ChannelIDs should be configurable, ensuring de-coupling of MCHits with detector numbering schemes. This solution would do away with the specialised MCVeloHit. Should be adopted LHCb wide. How to identify active volumes uniquely in XML and pass the information to Gauss is under discussion. Use of Geant4 copy numbers has been considered and tried in the past, and is deemed not to be robust.

AS: Also in the RICH even if with different aim the hit contains some "reference" to the detector

RICH review: A bit packed 32bit integer containing information on the history of the hit, such as whether it was produced by a Cherenkov hit or a charged track, or which radiator the hit came from (needed in RICH1 for example, to distinguish aerogel and C4F10 hits).

AS: So I agree with Chris that such discussion should be lhcb wide and since it doe not regards the muon event model only. Concerning the code that fill the ID as I told you it was proposed by Witold, I am not a G4 user so some study is needed before a better solution can be implemented but honestly I am pretty sure that it must exist.

CRJ: Concerning point (3) of AS above, do you need the digits ? Can you go straight from rawbuffer to the coords ?

AS: You should consider the muondigit an internal class, not to be written on storage. It is a useful high level interface among the raw buffer and the high level code and it can be modified whenever we like without spoiling the backward/forward compatibility of the produced data. I propose to keep the existing strategy since honestly I do not see any drawback. On the contrary I find such class useful. For instance during the study of the performance of the detector, time alignment cluster size, noise, space alignment etc. having the digit on the TES will for sure make more user friendly the procedure.

Let me stress that already in the HLT muonID the code skip the muondigit creation and goes from raw buffer to list of coord tileID. So I have no objection that the coords can be created starting from ra buffer. I only prefer to keep such high level interface as additional possibility.

20 TES containers per hit type
MC: Is there any use for the MuonDigit other than the use described? Clearly the associator should be links between MuonTileID and MCParticle, but why is the MuonDigit needed for this? It is sufficient (and far more robust) to replace the SmartRefVector inside MuonCoord with a std::vector.

MC: I have no objection to having MuonDigit as an internal class. But this implies that you modify MuonCoord and replace SmartRefVector with std::vector - otherwise the public, persistent class refers to a private transient only class. I can only see advantages with this change, as it minimises the coupling between classes in the event model, and does not lose any information (in fact, it adds information, because it allows you to navigate from MuonCoord to MCMuonDigit without going via MuonDigit).

AS: As you pointed out the replacement of smartref with std::vector is needed and of course it will be done.

MC: Also, if the code exists to go directly from RawBuffer to MuonCoord, I would argue that this code should be the default code running in Brunel as well as the HLT. The step via MuonDigit can be kept in the specialised code used by the Muon experts to study the detector performance.

AS: I believe such modification can be implemented. So at the end the MuonDigit will be almost a private class of the muon detector, which I think is a good compromise between all requirements.

CRJ: Concerning point (2) of AS above: If I understand the Velo/ST proposal correctly, then it is to use the key of the hit to store the detector information. Something like X of the 32bits to store the detector info, in whatever format you want, and the remaining 32-X bits to account for multiple hits in the same regions. Detector specific class could be written in XML to encode the information.

This is better than the current scheme in MCMuonHit as it does not introduce an additional word. Also, you would no longer need the MCMuonHit class and could use the MCHit class directly, with the MUON specific data stored in the templated key class. Do you think this is possible?

MC: This should be possible without too much new coding, because the detector identifiers already exist in part of the MuonTileID bit field.

My suggestion for a recommendation would be:

- Give a meaning to the "Key" of MCHit: some bits reserved for detector identifier, the rest for a sequential number of hit in the detector (cannot use full channelID, as there can be several MCHits in ne channel). The detector identifier should be the one coded in the various xxChannel/TileID, i.e. consistent with the Detector Element.

- Suppress MCVeloHit and MCMuonHit, use instead MCHit extended as above.

- Fill the detector identifier from when the MCHits are created (at the end of Gauss?) using the same, safe, method for all sub-detectors.

- MCRichHit should have a key with a similar meaning, but can remain a different class because it is smaller (no exit point).

AS: Yes, I think this could be a good solution since it fulfills all requirements of maintenance of whole LHCB code and in the mean time specific detector needs. The muon ID will require 13 bits out of the 32, I believe that it is ok, but some checks are needed to check that the 19 free bits in the key is ok to identify uniquely all hits in the events without problem. At first glance 256k hits in the muon detector are safe upper bound - otherwise more than software problem we will have "real life" problem.
MC: This should be more than sufficient. The "hit serial number" is per detector ID, I can't imagine you will have more than 256K hits per chamber!!

MC: This was designed for the L0 trigger, is it optimal also for use in the offline? Can it be packaged so that it sits in LHCbKernel package, like all other xxID classes? Also, should it be possible to store the MuonTileID as a LHCbID (in which case a scheme has to be found to pack all the info in 28 bits or less)? As a more general comment, are there synergies between L0Muon and Muon simulations that could be facilitated by a more integrated event model?

AS: For the muon software the addressing schema of MuonTileID satisfies all our needs. Concerning a more compact schema I think that it is conceivable for the muon software since 3 bits out of the 31 used in the MuonTile are not strictly required. I do not know about the L0 Muon simulation. I do not fully understand the last comment.

MC: Last comment is just a question to the review: are there places in either the L0Muon simulation or Muon digitization that could benefit from mutually desirable changes to the Muon event model classes, in particular MuonTileId and the MonteCarlo truth classes

CRJ: As Marco points out below, I do not think we should worry about the effort needed to implement any changes as part of this review (Look for instance at the tracking changes - this is a major rewrite). The point is to identify if anything could be improved. You say things "satisfy" your needs. Fair enough, but can things be made even better...

CRJ: on the point regarding the LHCbID (which I am also reviewing....) the use-case for this is primarily in the tracking event model, and there I think it is important that the MuonTileID is made compatible. It seems to me a reasonable idea that the muon hits could at some point be added to the tracks themselves as additional hits, to improve the track fit? At this point it will become essential.

MC: Is such a class needed at all in the event model, given that the same information exists in RawBuffer? How is it used? Could one not use the RawBuffer information directly to build the MuonCoords, without creating these additional objects?

AS: It is widely used in the code so that removing it is painfull. Which is the drawback of having it since it is not saved on storage? Since the digits are the objects sent to DAQ I suspect that even for studying the system the MuonDigit objects can be usefull especially if the raw buffer is not easy to access due to strange format to achieve a better compression or faster access by the mound in the trigger.

MC: I don't think pain of migration should be a criterion of this review. I am just wondering whether creating Digits on the way to making coordinates isn't adding complexity to the code, and also CPU time, given that these are tiny keyedObjects (only 4 bits of useful info other than the TileID) that are created with new to be put in the TES

CRJ: In general I agree in removing them if they are not essential to the processing model. Also, the muon id is already very fast and it is not inconceivable to rerun it in DaVinci if needed (see comments below regarding the MuonID and MuonCoord classes). In this case any improvements in speed will be very useful.

MC: if MuonDigit are suppressed from the event model, MuonCoord should contain a vector of MuonTileID identifying the digits, instead of the SmartRefs. This is in fact more efficient for the persistency and helps with truth relations (see below)
AS: Substituting ref with key is possible if required by the persistency.

MC: It is not required by the persistency but would allow to, e.g. navigate directly to RawBuffer digits without having to rebuild the MuonDigits container. Similarly it would allow direct navigation to MCParticle via the linker table described below.

**Relations**

MC: It was decided some time ago that all persistent associations would be based on Linker objects, not Relations. Given that Digits are uniquely identified by a MuonTileID, probably the simplest implementation would be a Linker table between MuonTileID and MCParticle (similar to what is done for all the tracking detectors). Each MuonCoord can then be associated to one or more MCParticles by looping over the MuonTileIDs of the corresponding digits.

AS: In my documents I suggested to modify the associator to improve understanding the MC truth. I am waiting for comments on such proposal.

MC: I'd like a comment on this counterproposal...

CRJ: If I can comment on both.... It seems the proposals come down to two decisions - Whether to use a relations table or linker, and whether to link to MuonDigits or MuonTileIDs. On the first, both relations and linkers can technically do the job (perhaps relations given more options and are more configurable..) but linkers have real benefits as regard persistency. I agree muon group should follow the general consensus and use linkers for persistency. On the second issue, it seems to me nothing to do with the MC relations themselves, as MuonTileIDs and MuonDigit both represent a single channel and thus are essentially interchangeable. In this sense, the decision is more if the MuonDigit classes have a use at all in the event model or should they be removed for simplicity. I would vote to try and remove them if conceivable (Accepting changes will have to be made to the rest of the muon code to adapt...)

MC: My point is that, if MuonCoords contain a list of MuonTileIDs rather than SmartRefs to MuonDigits, then a single Linker table between MuonTileID and MCParticle can be used to find the truth for both MuonCoords and MuonDigits; for MuonCoords this could then be done directly, without the intermediate hop via MuonDigits. This is then independent of the decision on whether to remove MuonDigits altogether.

CRJ: Why does the MuonID object contain references to MuonCoord objects - Are these really needed by DaVinci users ? I am guessing they are only needed in order to perform some additional processing in DaVinci after the main reconstruction in Brunel ? Would it not make more sense for the MuonID object to contain only the numerical (likelihood, shared hits etc.) data characterising the muon ID for the given track. All the important Muon ID processing should be done in the offline reconstruction (Brunel) and stored as a summary in the MuonID objects for use in DaVinci etc. In this case the MuonCoord objects become purely internal to the muon ID reconstruction code, and do not need making persistent.

Miriam: I think that at some point we thought it would be good to be able to re-do the muonID in DaVinci. But it is not needed by the DaVinci users in general - just the DaVinci users that make the muonID (us..). I think it was a question of not having to run Brunel, but maybe there is some technical reason I am not aware of.

CRJ : OK. In that case I would very much suggest removing it. I think in the rather rare cases that people want to rerun the muon reconstruction in DaVinci, for specialist studies by yourselves, or if a new tuning is available, then it is better to then simply rerun the full offline reconstruction chain, as in Brunel. I do not really see how having this smartref helps you do this ? All the information is available in the dsts to do this (RawBuffer and tracks I guess ?) so technically it is trivial to do. (It is done already for the CALO...)

MuonCoord
"on-demand" reconstruction and also I do it regularly for the RICH. This is the approach I believe is followed by other sub-detector reconstructions. Note that the keeping the smartrefs and also the additional size of the muoncoords on the dsts increases the dst event size.

Comment from Physics Event model review. For consistency with the CALO and RICH models, it would be nice if the "MuonID" object was renamed "MuonPID"

CRJ: Can I go back to the question of the use of the MuonCoords in the MuonID objects - There seems to be some confusion over why this is there ? My proposal would be it can be removed, with perhaps a new linker/associator created to go from MuonIDs to MuonTileIDs to cope with the fact that the MuonCoord -> MuonTileIDs can no longer be used ? Are these MuonCoords used for anything in DaVinci other than this association ? If they are only there to aid some secondary reconstruction can this not be done by going back to the rawbuffer and doing it all again ?

AS: Miriam should comment on this, but I believe that at the moment they are used for the calculation of the DLL by means of the appropriate tool. I imagine that as soon as the new muonID will be available the muoncoords will be no longer needed. Unfortunately I do not know which is the schedule for the new code in RIO

EP: I believe that for DC06, in the end of this year, we will have new MuonID objects, which will contain directly the DLL and the number of shared hits. The MuonCoord will be used in Brunel in order to calculate this DLL and nSharedHits, and will no longer be needed in DaVinci.

and

CR: Are these classes written out during normal productions, or only during private productions for expert monitoring ? If only in private, would it be useful to always have this info for debugging ? Also, why are these flags kept in a separate class, and not contained directly in the MCMuonHit and MCMuonDigit classes themselves (if they are always written out wouldn't it be simpler to just add these flags to the main classes).

AS: They are written always, they are member of the MCMuonHit and MCMuonDigit objects so they are written whenever the hits or mdigits are stored. I find easier to write code, to debug it and to read if a main class is slit in some smaller "closed" blocks.

Visualisation ???

CRJ: Are any of the above classes used in Panoramix for the visualisation of the Muon detector data ?

Use cases

OL: When running the L0 muon trigger, we are always interested in knowing:

- why a pad or a strip has been fired

- why a pad or a strip has not been fired (resulting in a muon lost). i.e understanding in details the physics performance of the detector.

With the current Event model, could you please remind me what is most efficient way to access to the following information, and if you think the implementation is optimal. I want to know why a MuonCoord has been hit:

- real muon (or pion, kaon or other real particle)
- origin of this real particle (beam halo, pileup, spillover, interaction point, ..)

- cross-talk electronic (from x ? from y ?)

- cross-talk geometric

- low energy background

- electronic noise

AS: The present implementation is far from optimal both concerning technical and "information content" items. Since some technical issues have been already discussed in the web page I will restrict to the "use case" you propose. With the present implementation the link between coord and MCParticle is quite crazy since the associator links the first in time hits crossing one of the H or V strips with the coords. If one of the strip is due to cross talk the information is lost. Same for low energy background, noise and beam halo muons, spillover. Please note that the correct information are stored in the MCMuonDigit, so the problem is only in the associator not in the digitization.

Please note that even changing the associator it is not possible to answer all your questions without using the MCMuonDigit which unfortunately are not stored in Brunel output. Unfortunately the problem is not simply solved storing the MCMuonDigit since they link to MCMuonHit which I believe cannot be stored ...

CRJ: Note that there is another way to access MC information that does not require all MC data to be copied to the dst (or whatever) file. If you use the POOL file catalogues then you can load objects in other files transparently. I use this all the time in my private RICH studies, and it allows me to load MC objects in, for example, the sim files, from jobs only running on the dsts.

One big problem with this is it does require the POOL catalogues to be maintained, and this is not done for the main productions and is non-trivial. Maybe Marco can comment, but I've never really been sure what LHCb's policy is on these files - Are we going to use them in public productions or not? I understand there issues, such as "what if someone's grid job triggers the loading of many files behind their back, and not locally available?"..... It seems to me that this question does have an impact on how we handle MC information...

MC: It is certainly the intention to keep the POOL catalogues. In fact, on the grid, file access will only be by logical file name and catalogue. But we should not base our event model on the assumption that we can read data on "ancestor" files, as this is likely to be very expensive. All data needed for a mainstream analysis should be available directly on the DST, access to the SIM or DIGI should be the exception (detailed studies of a sub-detector response, pattern recognition studies etc.) - it is with this in mind that, up to now, we have not been keeping the .sim tapes except for a limited sample necessary for such studies.

This is why a clever scheme for MC Truth is desirable. Do you really need the MCMuonDigits or MCMuonHits to keep the information needed by Olivier?

MC: It seems to me such information (OL above) could be coded in a single "MC Summary info" word, which links an int or to a MuonCoord, something similar to the MCTrackInfo object.

OL: When I look for a muon track in the muon trigger, I find, for example 4 hits "aligned" and one hit is missing in M4 FOI. I want to know why this hit is missing:

- chamber inefficiency?

- dead time?

- outside time window?
I would like to be sure that all these information will be accessible with the new event model.

AS: Such information are available using the MCMuonDigit only at boole and brunel level, not in Davinci. It will be not modified by new associator. I believe trace back the inefficiency is not possible in DaVinci for all detectors.

OL: It happens that a MuonCoord is fired because of several reasons (true muon + true electronic noise). In that case, how the information is stored in MCtruth?

AS: The preference is always to the true hit (real particle)

OL: Concerning Relations, is the LoKi problem solved? I thought that if we use exclusively Linker and no Relation, LoKi could not work anymore. But I have certainly misunderstood something?

CRJ: I could be wrong but I believe that whilst internally LoKi may use associators, it does take input from linkers and converts them to associators (I think it does this for trigger tracks). So I guess LoKi will break, but can be fixed... Note that LoKi will likely break in many many other ways due to all the other changes anyway, so I don't think this should be an issue.

MC: Correct. Relations are used extensively inside LoKi (and also inside the Calo reconstruction), and will continue to be. The only change is that we have decided that Relations are not allowed in classes to be made persistent, but they can be used in transient classes.

OL: Finally a general comments: Would it be possible to have a link between long tracks and hits in the muon detector. If I take a hit in M3, is there a simple way to know the long track associated to it (if any)? And if I take a long track identified as muon, how can I get the muons hits associated to it?

AS: I believe this use case is not muon only so a discussion lhcb wide should be required. Some features should be needed also the the other sbdetectors.

CRJ: One comment though, the MuonID objects do contain links to the associated track, so in some of the use cases above you can replace "track" with "MuonID". I.e. is it possible to get from a MuonID object to the hits that produced it? (If you can do this then the opposite is always possible). This is a MUON question only.

MC: I think this question came from Olivier Leroy. This is a question for the muon reconstruction. Of course, if someone bothers to extend the long tracks to the muon chambers, it is possible to do pattern recognition there. There is no problem adding muon hits to the tracks, as long as a MuonTileID can be converted to an LHCbID (which is not the case right now - and this is a MUON problem).