Subject: Minutes of Meeting 2004-05-26
Posted by NormanGraf on Mon, 28 Jun 2004 16:14:28 GMT
View Forum Message <> Reply to Message

9:08 AM 5/26/2004
TJ, NG, RC, FG, HV

Agenda Meeting 26-05-2004
---

1) Status f77, java, cpp

2) Reconstruction data model:

a) TJs comments on Tracks and LCRelations

b) user extensions (see message #94 in this forum)

c) extension to Track: getRadiusOfInnermostHit()

3) Transient LCCollections and multiple containment of one object
Discussing with users from DESY it turned out that they would need
the possibility to create subcollections of existing collections,
e.g. to analyze e+e->HH->l+l-,jets one would pass a list of all
tracks except the two identified leptons to a jet algorithm.
Copying the tracks is prohibitive in terms of memory and cpu,
simply creating a collection with pointers to existing tracks
will cause a segmentation violation (in C++,f77) when the event
is deleted, as the assumption is made that every object is
contained uniquely in one collection. One way around that is
to keep an additional reference to every object in an event
and delete objects to wrt. that list. Another way would be to
tag the collection as 'transient' and then not delete 'objects'
in transient collections. To my mind these are independent
concepts and we should implement both: allow one object to be
contained in more than one collection and allow for users to
flag collections as transient, i.e. they are only used as
input to some subsequent analysis/reconstruction module.

4) Other issues

Minutes:

TJ: updated java version of RECO, cluster, track and some parts of Relation (not IO)
also working on bporting fastMC and reco to use LCIO and newer JAVA version.
FG: was not able to read a file written in C++ reading with Java.
TJ still under development.
HV no updates on fortran version. will update documentation.
try to write lcio output from brahms after reconstruction. eta few weeks
FG: C++ started to implement changes for the track class. not yet checked in.
wanted to use strings for collection types, seems wasteful of space.
could just use an integer type, where user has defined bits.
TJ: no way to interpret bits in an arbitrary file, which is limiting.
would prefer to have something in header to define what the bits mean.
FG: can set only two bits for a track containing VTX and TPC hits.
TJ: would like to be able to use these files even if we don't necessarily
know what the content is or who used it.
TJ: header provides a detector (string), use this name to find out what the detector is
lookup is easier if it's defined by string and not some bitmap.
RC: put bits in for each track, use header to map bits onto names.
FG: had talked about putting some structue into run header.
TJ at a minimum, map bits onto detector names
FG: concerned about disk space, so would like to replace the doca boolean with another bit.
currently proposed to allow user to dynamically define the number of hits per detector.
e.g. getNumHits(string name of subdetector.
NG: should this be in run or event header. probably event, since we ocassionally will strip out events.
TJ names needn't be defined in a generic way, since we need to look up the detector geometry anyway.
how to predefine usable bits, e.g. tracker bits 0-15. so user needn't dump the run header to decode.
FG; doesn't solve problem of how to add a list of number of hits to a track.
currently we would need a string for the detector plus an integer defining the number of hits.
this is expensive.
TJ: comes up primarily for the fastmc or full reco where we drop the hits.
Requires more thought.

2a.) TJ had posted comments on relations.
should we have method to remove relationships?
writing into file relationships between types of class a and b with some weight.
sld found it useful to go in boh directions. current interface sees it as a one-way

relationship. proposes making this two-way. could implement using two maps, one for each direction, with some additional overhead. can perhaps allow user to define if relationship is two-way, or use lazy instantiation.
FG;what happens if we relate objects of the same type? be explicit about what direction we are going in.
TJ; make method names somewhat more explicit.
some discussion about api
TJ,FG: will review, propose more meaningfull methods names and functionality
TJ in constructor, not clear on what arguments are: interface name

2b.)
TJ: request #94 wasn't really generic, so prototype wouldn't be a solution.
FG: prototype + lcrelation would. should we try to implement this simple solution?
introducing generic object, leaving out macros
keeping in mind we want a better more felxible solution in the future..
TJ: sounds sensible. so user still needs to some coding, but targets the generic object, not sio explicitly.
FG: could intorduce mapping to define the data layout of the object.
TJ if string is purely descriptinve would be a lot happier.

3.)
two concepts ,
a.) having collections point to subsets of other collections.
b.) flag collection as transient
FG; think some more aboutthe C++ implementation

TJ in implementing java reco package on top of lcio, started by exposing lcio collections to user. not convenient, so have now put a thin layer above the lcio collections. thightly couple those to the IO, but will want to have some layer between the IO and the user.
problems, e.g lcio returns float[] since this makes sense for io, but transient event shouldn't have to be that tightly bound to the io.
downside is that there is a layer between user and lcio.
FG: would prefer to expose the lcio to the final user.
tj: low-level data interfaces tied to io file format, and higher level event interfaces
both defined by aid files to specify java and C++ implementations. would like to make user interfaces easier to allow them to use natural features of the languages.
leads to divergence, but this gets around coding to the lowest common denominator.

RC: looked at status bits, looks fine. how soon can we make a release?
TJ, FG: propose tagging a version which can be used for lcs developers.

TJ: updated java MCParticle.