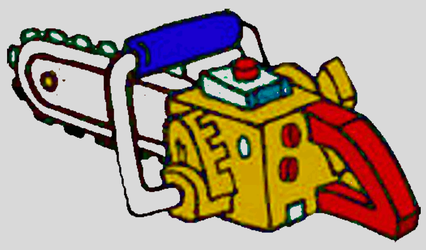


TSKIM : a tool for skimming ROOT trees

Motivation

As in many experiments, *Fermi* is storing its data within ROOT trees. A common activity of physicists is the tuning of selection criteria which define the events of interest, thus cutting and pruning the ROOT trees in order to extract all the data linked to those specific events. TSkim has been designed to facilitate this task. Initially a pair of PERL and ROOT scripts, TSkim is today a fully compiled C++ application, offering a panel of features going far beyond the original *Fermi* requirements.

Assumptions about Event Data Model

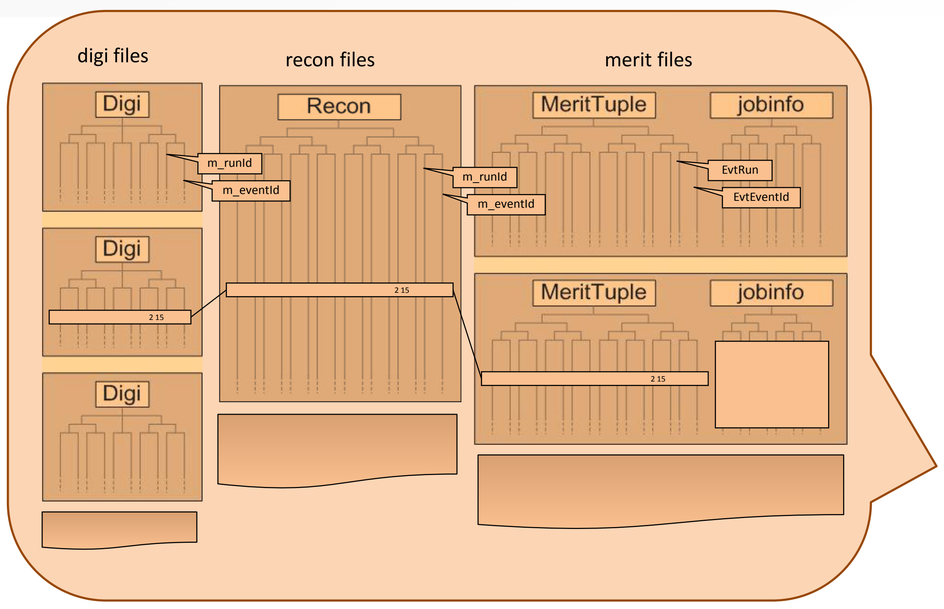
All the information about a physical event does not stay in a single tree. Each kind of information is stored in a dedicated kind of files and trees. In the example below :

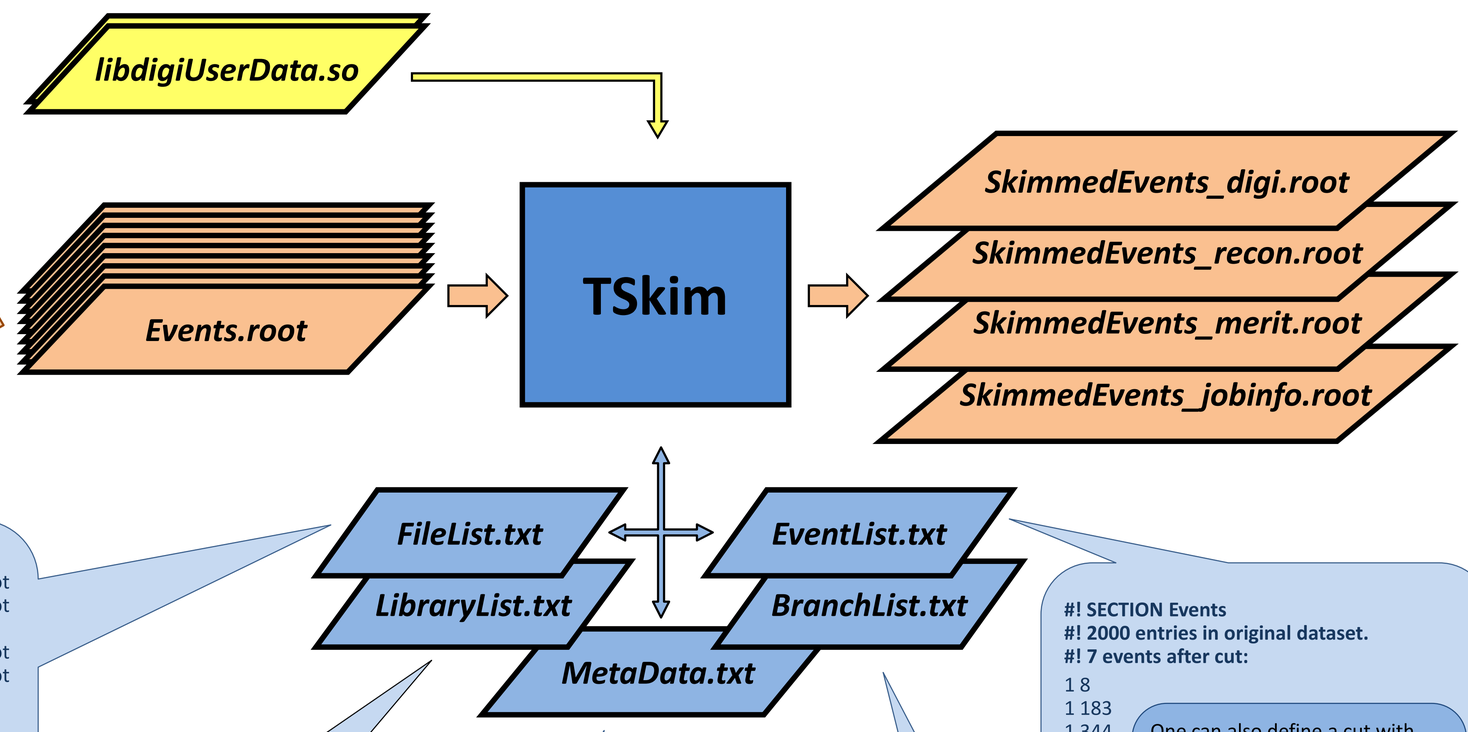
- raw measurements are collected in “Digi” trees, within “digi” files.
- reconstructed data is collected in “Recon” trees, within “recon” files.
- typical charasteristics are collected in “MeritTuple” trees, within “merit” files, together with “jobinfo” trees.

The different kind of trees are not expected to be aligned: the nth entry in a given tree does not refer the same physical event as the nth entry in another kind of tree (that’s why ROOT friend trees will not help us). Relevant entries are matched thanks to the branches which contain the run and event identifiers.

Why use TSkim rather than a simple home-made ROOT script ?

- Thanks to a meta-data file which list the names of run and event identification branches for the different kinds of trees (prepared once for all), the user can define a TCut for a single kind of tree and let TSkim extract the associated entries from all kinds of trees (see “MetaData.txt” below).
- TSkim selects the fastest ROOT features when possible (for example fast merging) and checks for known ROOT issues (for example if the range of event ids is compatible with TTreeIndex)
- For a given experiment, we can implement specific plugins. For *Fermi* we provide :
  - A connection with the experiment data catalog, which can return the list of input files for a given “task”.
  - An inspector which can interpret the TKey object “header” available in all *Fermi* data files, understands which release of the code was used when the data file was generated, and deduces which user-data shared library should be loaded.





#! SECTION Files
(**merit:jobinfo**)root://glast-rdr.slac.stanford.edu//glast/Data/1.70/merit/r0256187154_v001_merit.root
(**merit:jobinfo**)root://glast-rdr.slac.stanford.edu//glast/Data/1.70/merit/r0256192883_v001_merit.root
(**merit:jobinfo**)…
(**merit:jobinfo**)root://glast-rdr.slac.stanford.edu//glast/Data/1.70/merit/r0256198612_v001_merit.root
(**merit:jobinfo**)root://glast-rdr.slac.stanford.edu//glast/Data/1.70/merit/r0256204341_v000_merit.root
(**digi**)/nfs/u35/MC-tasks/BeamTest-10_0000_digi.root
(**digi**)/nfs/u35/MC-tasks/BeamTest-10_0001_digi.root
(**digi**)…
(**digi**)/nfs/u35/MC-tasks/BeamTest-10_0008_digi.root
(**digi**)/nfs/u35/MC-tasks/BeamTest-10_0009_digi.root
(**recon**)/nfs/…
…

If relative paths are used, files are searched for in directories defined by **TS_DATA_DIRS**

#! SECTION Events
#! 2000 entries in original dataset.
#! 7 events after cut:
1 8
1 183
1 344
1 553
2 117
2 517
2 980

One can also define a cut with Unix variables. For example :
TS_TCUT = « TkrEnergy>200 »
TS_TCUT_DATA_TYPE = « merit »

#! SECTION Libraries
/nfs/u09/builds/rh9_gcc32/Beamtest/v3r0907p0/libcommon.so
(**mc**)/nfs/u09/builds/rh9_gcc32/Beamtest/v3r0907p0/libmc.so
(**digi**)/nfs/u09/builds/rh9_gcc32/Beamtest/v3r0907p0/libdigi.so
(**recon**)/nfs/u09/builds/rh9_gcc32/Beamtest/v3r0907p0/librecon.so

If relative paths are used, files are searched for in directories defined by **TS_LIB_DIRS**

#! SECTION MetaData
(**merit.treeName**) MeritTuple
(**merit.runIdBranchName**) EvtRun
(**merit.eventIdBranchName**) EvtEventId
(**jobinfo.treeName**) jobinfo
(**digi.treeName**) Digi
(**digi.runIdBranchName**) m_runId
(**digi.eventIdBranchName**) m_eventId
(**digi.topBranchName**) DigiEvent
(**digi.topBranchType**) DigiEvent
(**digi.libName**) libdigiRootData.so
…

#! SECTION Branches
(**merit**) -*
(**merit**) +PtT*
(**merit**) +Cal*
(**digi**) +m_eventId
(**digi**) +m_runId
(**digi**) +m_acd
(**digi**) -m_cal
(**digi**) …

Future Work

- **Better support for very simple use-cases** : *Fermi* users have reached some limits when dealing with many small files, or very large files. We must probably drop TEventList/TEntryList for something more scalable, and reduce the number of loops through all data files.
- Manage **Meta-Data evolution**.
- Implement a new way to associate the entries from different types of trees, based on **timestamps**.
- **Deliver an API** : we have many ROOT utilities which are certainly worth sharing with users.

The Composite Event List

After it has localized all the user data of interest, and if requested to do so, TSkim can produce a special kind of ROOT file which does not contain the skimmed data, but the file names, tree names, and entries of each piece of skimmed data. We call this a “Composite Event List”. Such a light weight list can be communicated to collegues. It can also be reused as input for a subsequent TSkim job. This is how one can apply successive cuts on different kind of trees, while never duplicating real user data.

http://llr.in2p3.fr/trac/tskim/