Use of Root I/O Trees for CMS Crossings

Benefits and deficiencies of Root I/O trees when :
- NOT dealing with TObjects,
- reading the trees entries NOT sequentially,
- processing them NOT one by one.

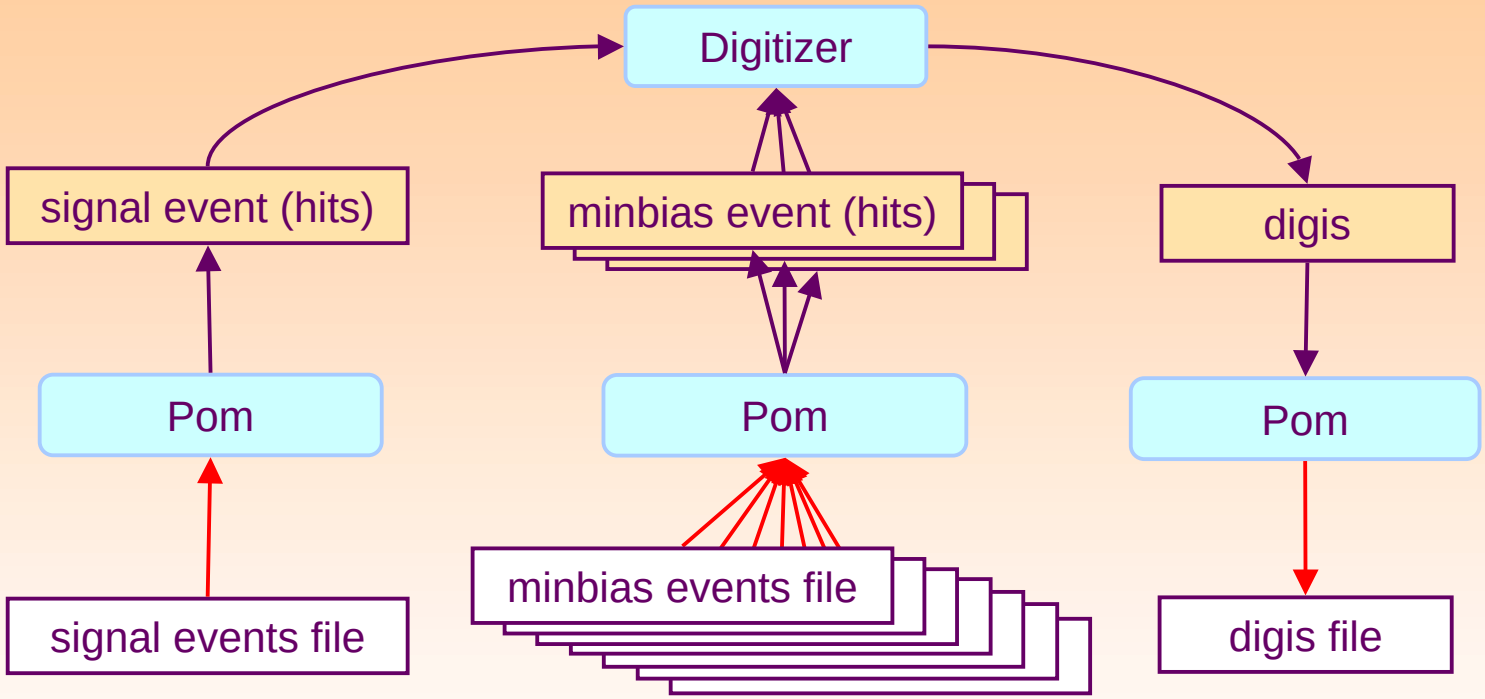
Outline

3. Goal & scope.
4. Main use-case.
5. Crossing data model.
6. Persistent objects managers.
7. Four persistency strategies.
8. From foreign to Root classes.
9. Implementation issues.
10. Performances.
11. Conclusions.
12. Future work.

Goal & scope

- Evaluate the use of TTree for the persistency of CMS event data (whose classes heavily rely on templates and external packages).
- Focus on the generation of crossings (pile-up of about 200 simulated events chosen randomly).
- Not covered yet : meta-data and references.

Main Use-Case



Crossing Data Model

- The folders ***//root/pool/**** represent the events composing the current crossing.
- Each event folder contains collections of **RtbTrackHit**, **RtbCaloHit**,…
- These collections should be kind of **RtbVArray<>** :
  - **RtbCArray<>** (home-made C-like array).
  - **RtbClonesArray<>** (wrap a **TClonesArray**).

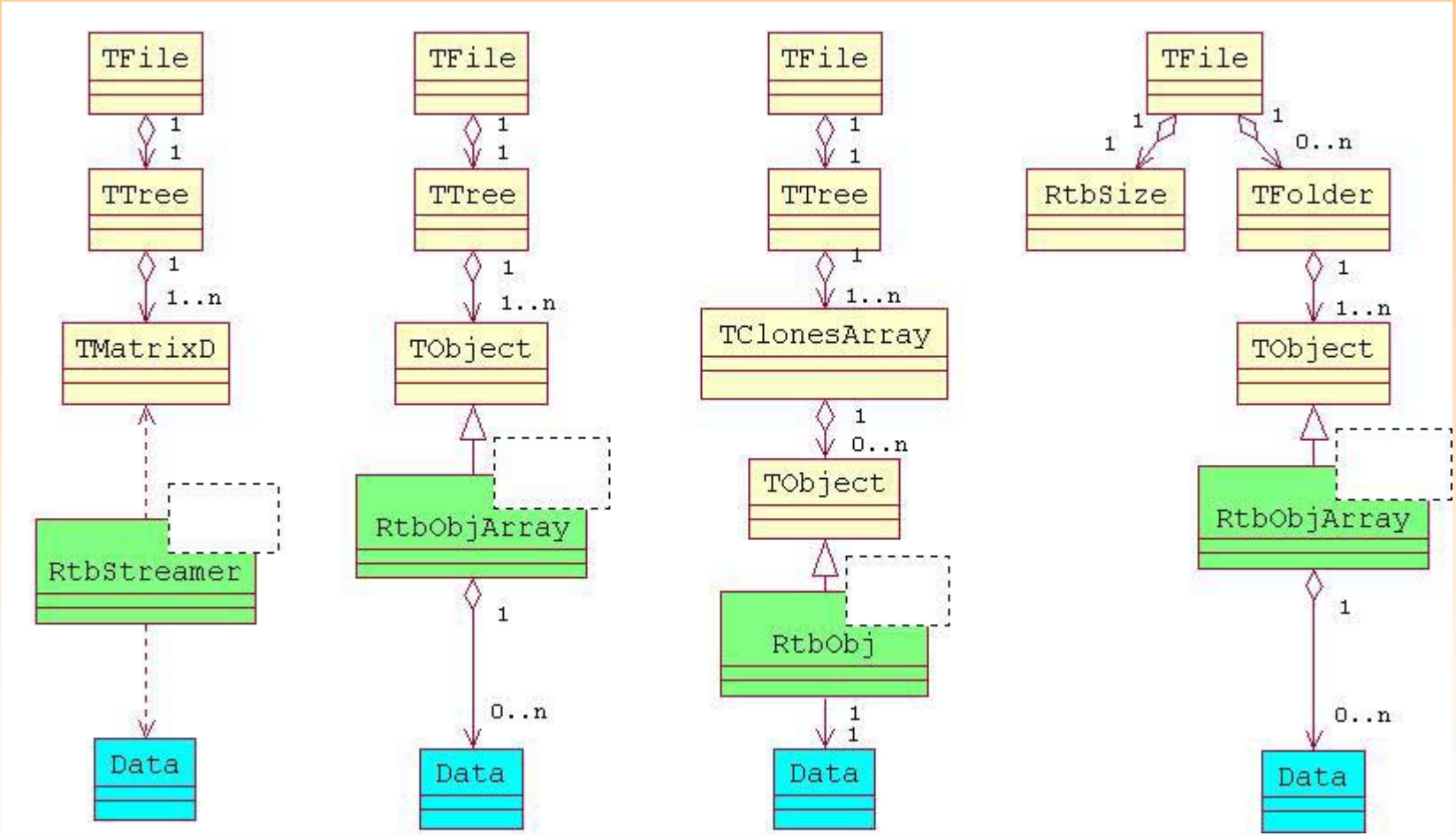
Persistent Objects Managers

- Persistency managers are able to transfer event data (set of RtbVArray<>) between a TFolder and a TFile.
- We implemented three kinds of RtbVPom :
  - **RtbDirectPom** : attach each RtbVArray<> of the folder to a branch of a TTree.
  - **RtbMatrixPom** : for each RtbVArray<>, create a TMatrixD and attach it to a branch of a TTree.
  - **RtbKeysPom** : directly store the **TFolder** in the **TFile**, each time with a different meaningful name.

Four persistency strategies

1. RtbCArray and RtbMatrixPom (matrix)
2. RtbCArray and RtbDirectPom (carray)
3. RtbClonesArray and RtbDirectPom (clones)
4. RtbCArray and RtbKeysPom (keys)

Storage of non-TObjects



Implementation issues

- Tips for scram
  - add **-p** to rootcint (why not the default ?)
  - add **-fPIC** for scram link step
  - remove **-ansi –pedantic**
- Typical problems with Root I/O :
  - collections sizes and operator[],
  - storage of empty collections,
  - redirection of pointers attached to branches,
  - tuning of branchs of a Tchain.

Performances

	Matrix	CArray	Clones	Keys
500 events file size (Mb)	119.6 63.3	119.3 63.8	106.6 57.7	118.9 63.5
500 events write time (s)	153 74	197 96	147 72	191 95
200 random events read time (s)	4.57	5.82	5.88	6.93

Conclusions

Given the specific use-case & the heavy use of templates and foreign classes…

…Use of TTree appeared more complex to tune than “direct” storage.

…It improved slightly the performance.

…It opens the door to ROOT analysis features.

…We failed to really take profit of TClonesArray.

Future work

- Solve few remaining memory leaks (rather in CMS digitization code).
- Provide root team with unscramed demonstration of what we observed.
- Implement a fifth strategy using TObjArray.
- Look inside ROOT classes implementations, optimize strategies accordingly, and perhaps change the conclusions.
- Add references between objects.