Skims in FastSim

Matteo Rama
Laboratori Nazionali di Frascati

20 March 2012

definition of skim at BaBar

- Skim: a subset of the AllEvents collection according to a set of selection criteria
- Examples:
  - *BSemiExcl*

    pre-selection of $B^0 \to D^{(*)-} X^+$, $B^+ \to \bar{D}^{(*)0} X^+$ X=combination of pions and kaons
  - *Tau11*

    pre-selection of tau→1prong vs tau→1prong
  - *DmixD0ToKPiPi0*

    pre-selection of $D^{*+} \to D^0\pi^+$, $D^0 \to K\pi\pi^0$ for D-mixing analysis

Skims are produced in central productions called *Skim cycles*

skims at BaBar

- Let's suppose one wants to select $B^+ \rightarrow \tau^+ \nu$ with $B^- \rightarrow D^{(*)0} X^-$ on the other side ("recoil analysis")
  - running the selection over the skim BSemiExcl saves a factor $O(10^3)$ in the number of processed events compared to running over AllEvents

- All BaBar physics analyses are performed by selecting the events on the appropriate skim

BaBar vs FastSim

- In BaBar all processed events are saved on disk. A skim is a subset of the *AllEvents* collection
- In FastSim, events are not saved on disk
  - The output ROOT file of a FastSim job is ***not*** a collection of events, but rather a collection of variables describing some aspects of the event

How can skims be implemented in FastSim?

Skims in FastSim

- Underlying idea:
  Skim: collection of random number generator seeds

"All Events"

event1	seed1
event2	seed2
event3	seed3
event4	seed4
…	
eventN-3	seedN-3
eventN-2	seedN-2
eventN-1	seedN-1
eventN	seedN

"Skim XXX"

event1	seed1
event2	seed2
event3	seed3
event4	seed4
…	
eventN-3	seedN-3
eventN-2	seedN-2
eventN-1	seedN-1
eventN	seedN

- Skim XXX identified by the sequence of seeds:
  {seed3,…,seedN-3,seedN-1}

Requirement

- **Basic requirement:** two FastSim jobs with same configuration and same seed produce exactly the same reconstructed event

Problems

- **Basic requirement:** two FastSim jobs with same configuration and same seed produce exactly the same reconstructed event

not satisfied

example:
(from old version of PacEmcClusterMeas::recursiveFillCluster())

Loop over a list of pointers. The elements of the list are ordered by the pointer value

```
std::set<AbsDetIndex*>::const_iterator iter = tci->itsNeighbours()->begin();
while ((iter != tci->itsNeighbours()->end())) {
  const TwoCoordIndex* tci2 = dynamic_cast<const TwoCoordIndex*> (*iter);
  recursiveFillCluster(cluster, tci2, hit, scale, method, RMxp, RMxm, RMyp, RMym);
  ++iter;
}
```

Since the pointer values can be different in two jobs with the same event seed, the order in the iterator can be different as well

recursiveFillCluster(…) uses random numbers

If the order of pointers in the iterator change then sequences of random numbers, that are the same in both events, are applied to different elements.
➔**The final result can be different in two events with the same seed**

Fix

- **Basic requirement:** two FastSim jobs with same configuration and same seed produce exactly the same reconstructed event

not satisfied

example:

Loop over a list of pointers. The elements of the list are ordered by the pointer value

```
std::set<AbsDetIndex*>::const_iterator iter = tci->itsNeighbours()->begin();
while ((iter != tci->itsNeighbours()->end())) {
  const TwoCoordIndex* tci2 = dynamic_cast<const TwoCoordIndex*> (*iter);
  recursiveFillCluster(cluster, tci2, hit, scale, method, RMxp, RMxm, RMyp, RMym);
  ++iter;
}
```

Since the pointer values are different in different events with the same seed, their order in the iterator can be different as well

recursiveFillCluster(…) uses random numbers

If the order of pointers in the iterator change, then sequences of random numbers, that are the same in both events, are applied to different elements.

➔**The final result can be different in two events with the same seed**

Fixed

Procedure

1. Create the skim(s):

   run the skim selection (for example B→D(*)X) on generic events and store the seed of selected events in output ROOT file (for example *skimCollection.root*)

   seed: runNumber (set by RUNNUM tcl var)
   evtNumber (in typical configuration it goes from 1 to NEVENT)

2. Run on the skim 'collection'

   Use the module **PmcSkimEvents** to 'read' (i.e. to generate) the skimmed events

```
module input PmcSkimEvents
module disable RacTestInput
talkto PmcSkimEvents {
   inputFile set skimCollection.root
   treeName set T
   runnumVar set runNumber
   evtnumVar set evtNumber
   verbose set false
}
```

Code and instructions

- Code committed and partially validated
  - In V0.3.1+ Patches_devel
  - Follow instructions for developers in FastSim User guide

http://mailman.fe.infn.it/superbwiki/index.php/FastSimDoc/InstructionsForDevelopers

more validation needed