Event Mixing: Requirements for the Analysis Framework

A. Morsch

WOM – 22/11/2007

Motivation

Experience from PHENIX

It is essential to have one single event mixing framework since there are many pitfalls to make regarding how one sets up the event pools and does the normalization etc.

There are never ending discussions on these things unless everyone uses the standard framework.

(D. Silvermyr)

Objective

- Collect requirements for event mixing in ALICE
- Mine for existing code
- Derive requirements for the Analysis Framework and implement possible common solutions

Thanks for your input !

A. Pulvirenti, R. Vernet, P. Christakoglou, M. Ivanov, M. Vala, Kisiel, P. Hristov, F. Carminati, Y. Kharlov, A. de Falco, D. Tapia, J. Faivre, D. Silvermyr, ….

General Requirements

- 2 nested loops over events which are “close”
- “Close” means similar
  - z-vertex
  - multiplicity
  - reaction plane
  - but also same geometrical acceptance of the detector, ….
- Re-use every event *N* times, where *N* is a small number $\mathcal{O}(10)$

Main questions

- How to group (pool) events according to a predefined metric ?
- How to access the events in the event loop ?
- How can this work in a distributed environment ?
- Do we need special event formats ?

Event grouping

- Two proposed solutions
  - Equidistant bins
  - Define around each event a region (hyper-sphere) in which partners can be accepted
    - Use of KD-Trees

Data access: extreme solutions

- Pre-bin the events according to one or more criteria and copy into pools
  - Not very flexible ?
- Produce event lists for each bin and send one job per bin
  - Random access of storage elements, files and events by concurrent jobs
  - Orthogonal our original philosophy: “Send the analysis code to the data”
- Sequential processing of events, buffer each event until *N* partners have been found
  - Memory !
  - Could work in special cases (muons, electrons, high p_T, …)

A possible solution

- Pre-run:
  - Use tag metadata to define the event binning
  - Add file id as axis to the binning
  - Use implementation of a sparse histogram and store pointer to event list in each bin.
- Run:
  - Replace id by storage element.
  - Select the storage elements optimising bin coverage for required statistics
  - Split jobs and send to sites with storage elements. Histogram is sent with storage element bin projected out.
  - For each site two possibilities
    - Loop over bins and then over events in each bin
    - Loop over events and determine partners from binning
- Event buffering
  - At least two events in memory, ideal would be *N*
    - To be supported by the InputEventHandler
    - Correct replication of AODEvents
    - New event formats, less memory ?