STATUS of ATLFAST++ into LHC++

From generation to analysis using Atlfast++ algorithms into LHC++:

→ Objectivity : version 4.0.2 (IBM platform)
version 5.1 (SUN platform)

→ HEPExplorer (running with Objectivity 4.0.2)

→ HTL

→ HepODBMS

→ CLHEP

Structure of ATLFast++ into ROOT :

- pipeline of modules called “Makers”
- ATLFast class = main class to control the program, it permits to loop on all Makers
- ATLFMaker class = common base class that defines main basic rules followed by all Makers
- each Maker is responsible for creating its branch of the output ROOT Tree

The following Makers are implemented:

ATLFMCMAker: to invoke event generator (Pythia)

ATLFClusterMaker: to reconstruct clusters

ATLFElectronMaker: to reconstruct electrons

ATLFMuonMaker: to reconstruct muons

ATLFPhotonMaker: to reconstruct photons

ATLFJetMaker: to reconstruct jets

ATLFTrackMaker: to reconstruct tracks

ATLFTriggerMaker: to build various trigger types

ATLFMiscMaker: to compute some event parameters (missing energy..)

MAIN STEPS followed:

A) Eliminate ROOT dependencies from ATLFast++ classes (➞ deeply integrated into ROOT)

B) Check results of reconstruction with FORTRAN version of ATLFast

C) Organize data into Objectivity

D) Read data from Objectivity and visualize them with HEPExplorer

A - Eliminate ROOT dependencies from ATLFast++ classes:

→ use of the container class TClonesArray which consists of a special array designed to store a large quantity of objects

↓

→ replaced with :

STL vectors/lists in the transient version of program

Objectivity VArray to store persistent objects

- ATLFMCMaker class organises the generation of the physics processes by Pythia, it presents deep dependencies from ROOT classes (TObject, TNamed , TPythia, TClonesArray… .)

⬇

- replaced with a Pythia wrapper made of 3 main classes:

  - Pythia → reads datacards and call Pythia routines
  - HepParticle → defines particle quantities (px, py…) and calculate rapidity, pT…
  - AtlasEvent → organizes list of particles in a STL vector

B - Check results of reconstruction with FORTRAN version of ATLFast:

Check on a statistic sample of 10000 events on two physics channels:

$Z^o \rightarrow e\ e$

$A^o \rightarrow \tau\ \tau \rightarrow (\ \mathrm{jet}\ \upsilon\ )\ (\ \mu\ \upsilon\ \upsilon)$

good channel for benchmarks :

- many signatures in event (jets, μ, missing E)
- missing E reconstruction permits a check of all reconstructed energies

Some plots of comparison

C - Organize data into Objectivity
(with help of Dino Ferrero Merlino)

- Followed Tag/Event data model proposed by LHC++ project

- Simple architecture :
  - one single container for Event, no associations
  - sketch of Event and Tag

D - Read data from Objectivity

- Visualize data with HEPExplorer and perform analysis map :

  an Explorable Collection has been registered on the Tag to make data visible to HepExplorer

- Read data from Objectivity with a simple C++ program, perform analysis and fill some histograms with Histomanager (hbook wrapper by A. Dell'Acqua) ➞ developped for MONARC purposes

For Monarc tests → Objy/DB populated with 100.000 evts from Atlfast++ :

- ~ 0.11 sec/evt
- 1 evt ~ 40 Kilobytes ~ 5 pages
- open a new Events.DB each 40.000 evts
  (max DB size = 2 Gigabytes)
- Event.DB ~ 1.7 Gigabytes (for 40.000 evts)
- Tag.DB ~ 8.4 Megabytes (for 100.000 evts)

In Milan started first retrieving tests of Objectivity on local federation with configuration:

System = SUN Ultra 5 (333 Mhz)
OS = Solaris 2.7
Compiler = Sun C++ 4.2
Objectivity version = 5.1
Objectivity page size = 8192 Bytes

Final remarks:

- This work needs further refinements and debugging :
  - extract last ATLFast++ classes from ROOT
  - check reconstruction on more physics channels

- It could be a usefull tool to learn/test LHC++ components (Objectivity, HEPExplorer...) and eventually it could be interfaced with other analysis tools in order to make comparisons of performances

- All information can be found at the following adress:
  http://www.cern.ch/GROUPS/SOFTWARE/OO/domains/analysis/analysis_tools.html