RDataFrame-based analysis in *k4megat*

Yong Zhou

Megat Weekly Meeting

May 2, 2023

Outline

Persistence Format of *edm4hep*

Columnar layout direct accessible in *RDataFrame* (hits in `std::vector`)

xxxHitData the underlying **`struct`** saving hit information

```
Vector3f:
  - float x
  - float y
  - float z

Vector3d:
  - double x
  - double y
  - double z
```

```
SimTrackerHitData:
  - uint64_t cellID
  - float EDep
  - float time
  - Vector3d position
  - Vector3f momentum
  - ...
```

```
SimCalorimeterHitData:
  - uint64_t cellID
  - float energy
  - Vector3f position
  - ...
```

More explanation can be found in *k4megat* doc and edm4hep official doc.

ID Specification

- **cellID** in *xxxHitData*: 64-bit (cell id and segmentation policy)
- Encode/decode format specified by a segmentation string

CZT calo	`'system:6,section:6,layer:6,row:32:8,column:8,x:8,y:8'`
TPC (strip)	`'system:6,layer:6,strip:16:48'`
TPC (pixel)	`'system:6,x:16:24,y:24'`

- Sub-detector ID: `'system'` `(TPC:1, CZT:2)`
- CZT Calo
  - `'section'` `(-Z: 0, +Y: 1, +X: 2, -Y: 3, -X: 4)`
  - `'layer'` `(0~3)`: layer id inside each section
- TPC (strip)
  - `'layer'` `(X: 0, Y: 1)`
  - `'strip'`: strip id inside each layer
- TPC (pixel)
  - `'x'`,`'y'`: row and column id of each pixel

The *analysis* Package (latest in *ana-dev* branch)

Motivation:

- *Gaudi* is powerful and flexible, but over-skilled for last-mile analysis
- Lightweight & agile solution is needed for explanatory analysis
- **RDataFrame** is the best solution
  - same idea as *R* and *pandas*, but in C++
  - implicit multi-threading (both data reading and processing)
  - convertible to *numpy*

Features available:

- Helper functions/functors to be used in *RDataFrame* analysis workflow
  - cellID decoder
  - cellID -> cell position
  - edm4hep structure -> *ROOT::RVec* container
- *megat*: a Python-binding package
  - Alternative to ROOT C++ macro
  - Auto-load the utility libraries
- *mgana*: a light-weight analysis framework (under development)

Declarative Programming

Declarative Programming:

- Just specifies what you wanna do
- No explicit loop management, all operations are column-wise (i.e. branches)
- Common column operators provided by ROOT
- Custom column operations defined by end-user, in any of callable objects in C++:
  - function
  - functor class
  - lambda

A related concept is Functional Programming, which is an programming paradigm focusing on data immutability/locality. The paradigm is mostly leveraged for easier parallelization. *Gaudi* supports Functional Programming as well, but not easier to use for end-user.

Python script

```python
import ROOT
from megat import simcalo

ROOT.EnableImplicitMT()

df = ROOT.RDataFrame("events","megat.root")

df2 = (
   df.Define("x1","SimCalo::hit_x(CztHits)")
     .Define("x2","CztHits.position.x")
     .Define("dx", "x1-x2"))

h1 = df2.Histo1D('dx')

c = ROOT.TCanvas()
h1.Draw()
c.SaveAs('demo_fillx.png')
```

C++ macro

```cpp
void demo() {
  LoadMegat();

  ROOT::EnableImplicitMT();

  ROOT::RDataFrame df("events","megat.root");

  auto df2 =
    df.Define("x1","SimCalo::hit_x(CztHits)")
      .Define("x2","CztHits.position.x")
      .Define("dx", "x1-x2")

  auto h1 = df2.Histo1D("dx")

  auto c = new TCanvas();
  h1->DrawClone();
}
```

`LoadMegat()` or `import megat` is mandatory to activate *k4megat*.

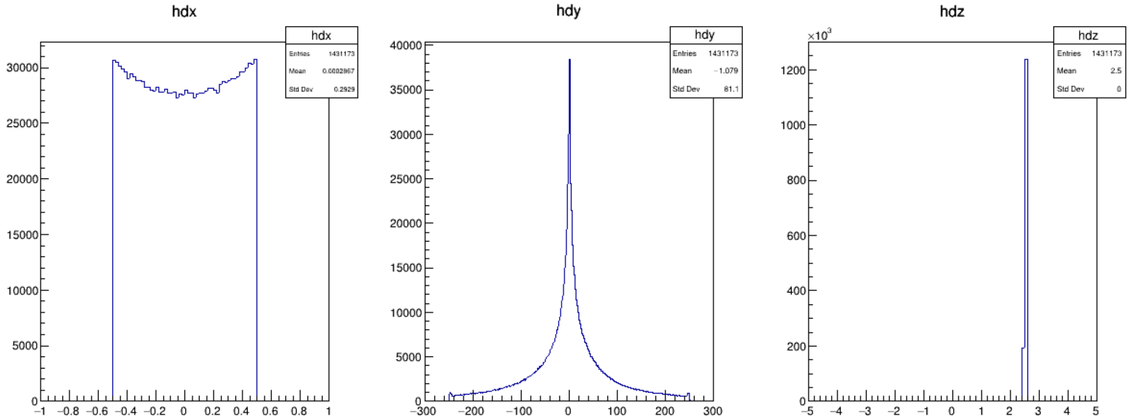
loadGeometry, IdConverter & CellPosition

```
using namespace megat::utility;

auto mgRoot = std::getenv("MEGAT_ROOT"); // configured by thismegat.sh
auto xmlGeom = fmt::format("{}/geometry/compact/Megat.xml",mgRoot); // master xml
auto xmlTpc  = fmt::format("{}/geometry/compact/TPC_readout.xml",mgRoot); // tpc readout xml

loadGeometry({xmlGeom, xmlTpc}, ro_name); // ro_name may be: TpcStripHits or TpcPixelHits

IdConverter idConv(ro_name); // Strip and Pixel readouts have different id converter
bool  is_strip = idConv.isStrip("TPC"); // check the TPC readout pattern

auto  decoder  = idConv.decoder("TPC"); // get decoder; 'TPC' or 'Calorimeter'
auto  layer_id = decoder->get( cell_id, "layer" ); // decode the field value from a cell id

CellPosition<edm4hep::TrackerHitData> cell_pos(ro_name); // functor to get cell position
```

fmt is a high-performance string formating library integrated into *analysis* package.

A full demo to compare TPC sim and recon position

μ, 10 GeV, *Isotropic*, $(0, 0, 0)$



https://github.com/MegMev/k4megat/blob/sim-dev/analysis/scripts/demo_tpc_pos.C