RAM optimization for digitizing long tracks

Pietro Meloni and Igor Abritta Costa
18-04-2023

Problem

The digitization code required too much RAM (~30 GB) for long tracks (~100 keV). This made impossible the digitization of background images (jobs killed)

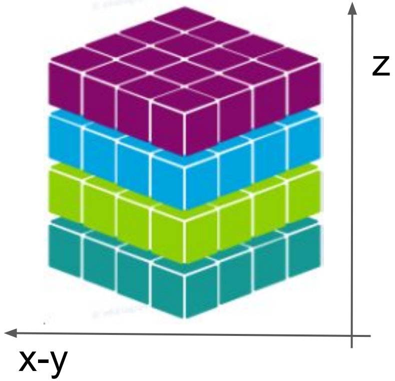
This is due to the saturation effect that requires the use of a 3D histogram, in which each voxel represents a GEM channel at a given time. The number of primaries in each voxel must be computed to apply the saturated gain.

Solution

We now apply the saturation effect in layers along the z-axis:

1. we introduce a new parameter **V_{max}= max volume of the 3D histo** (max number of voxels);
2. the number of layers **N** is given by the volume of the smallest cuboid containing the track, divided by V_{max}
3. for each layer, we fill the 3D-histo and we apply the saturated gain;
4. we sum all the results along the z.
5. finally, we apply the optical factors (solid angle, photons per electron, etc…)

example
V_{max}=16 voxels



Results on LNGS cluster

E < 50 keV -> no big differences in RAM and time

50 keV < E < 200 keV -> now: ~8 GB, ~1 min (before: ~32 GB , ~1 min)

E ~ 1000 keV -> now: ~32 GB, ~2 min (before: practically impossible)

The images are the same as before: same linearity plot (integral vs energy) before and after the optimization

Code here: https://github.com/CYGNUS-RD/digitization/pull/17

Further improvements if needed (in order of complexity)

- parallelize new saturation loop (speed up)
- reduce x-y dimension of single layer in saturation loop (save RAM, for oblique tracks)
- use sparse object for saturation (at the moment the numpy object is taking memory for zeros)
- use cython to compile code as C and define datatype (int16)