TB 2021: Remarks and Questions

Veta Ghenescu, Alina Neagu, Mihai Potlog
Institute of Space Science, Bucharest

What we focused on …

- Telescope data – alignment using Corryvreckan code
- Sensor data – first look over the data format
- Computing resources

Telescope data

What we've learned

- Since we do not have an DESY account we had to install all software on local servers;
- Corryvreckan is not very straightforward to install;
- It requires some workaround to solve de dependencies, especially with Eudaq2 and Root;
- Configuration files: while easy to read/understand the format, require time to really understand the method used for each module;

What we've done

- A lot of compilations and recompilations to make the code work;
- Redesign Shaun's TrackingInfo() module in order to get the information about the positions of the particles both at the exit of telescope plane and entering the sensor;
- Alignment done in 5 steps: prealignment + 3 x alignment + run;
- Testing on several runs with different configurations of modules

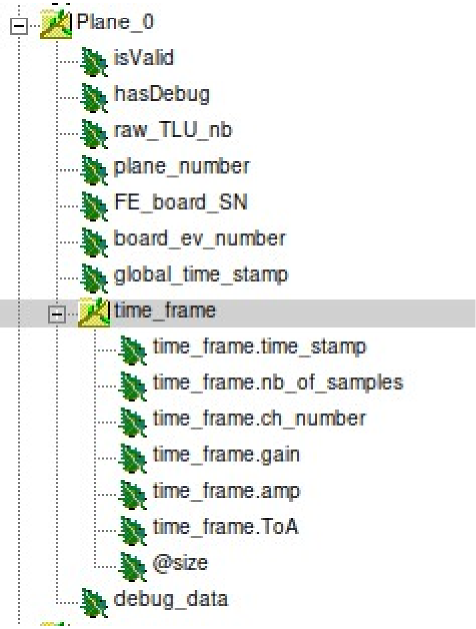
Open questions …

- Since the telescope hasn't been reconfigured / moved during the testbeam, do we need to perform alignment for each run or it's enough to do it once?
- There are several parameters which are set in the config files that are taken by granted such as: time_cut_abs, track_model, shift_triggers, momentum, etc. … A deeper understanding of this would be useful.

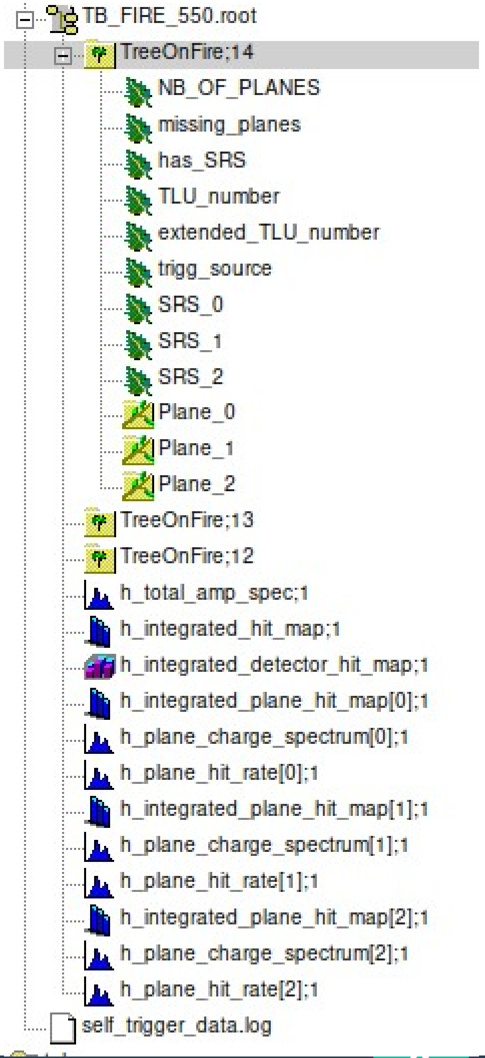
Sensor data

What we've learned

- For each run a pair of files are created: TB_FIRE_run-nb.root and TB_FIRE_run-nb.log – corelated with the telescope number, ran000nbrun.raw ;
- Sensor data are stored in the .root file;
- Each .root file contains a few Trees and several histograms;
- Tried to identify where all the useful information is stored;



```
Plane_0
    isValid
    hasDebug
    raw_TLU_nb
    plane_number
    FE_board_SN
    board_ev_number
    global_time_stamp
    time_frame
        time_frame.time_stamp
        time_frame.nb_of_samples
        time_frame.ch_number
        time_frame.gain
        time_frame.amp
        time_frame.ToA
        @size
    debug_data
```



```
TB_FIRE_550.root
    TreeOnFire;14
        NB_OF_PLANES
        missing_planes
        has_SRS
        TLU_number
        extended_TLU_number
        trigg_source
        SRS_0
        SRS_1
        SRS_2
        Plane_0
        Plane_1
        Plane_2
    TreeOnFire;13
    TreeOnFire;12
    h_total_amp_spec;1
    h_integrated_hit_map;1
    h_integrated_detector_hit_map;1
    h_integrated_plane_hit_map[0];1
    h_plane_charge_spectrum[0];1
    h_plane_hit_rate[0];1
    h_integrated_plane_hit_map[1];1
    h_plane_charge_spectrum[1];1
    h_plane_hit_rate[1];1
    h_integrated_plane_hit_map[2];1
    h_plane_charge_spectrum[2];1
    h_plane_hit_rate[2];1
self_trigger_data.log
```

Sensor data

What we’ve done

- Write an script in root to read the data from the .root file;
- To properly read the data an dictionary must be created; this provide the definitions of structures used for root file composition;
- Write a script in root to identify the empty-events; also many attempts were made, we cannot comply with the numbers from the logbook;
- Started to reproduce in Geant4 the geometry configuration used in Testbeam with GaAs type of sensor.

Open questions …

- Each TreeOnFire from .root file is created more then once; is this due to the SetAutoSave () number which is set after some amount of data is collected or is due to some other issue? Which one is the one which should be used?
- Is there any tutorial which explains the content of each branch/leaf?

Computing resources

What we’ve done

- Purchase and install in a rack a new server dedicated to Fcal-Luxe;
- Tried to comply with the configuration already existing at DESY for a smooth transition

Actual configuration:

CPU model/make:	AMD EPYC 7713P 64-Core Processor
CPU Core:	64
Thread per core:	2
Total threads:	128
Total memory:	258 G
Storage :	6 x 12 T drives in RAID6
Total storage:	44 T

- Operating system: CentOS Linux
- CernVM File System installed (cvmfs)
- Access via SSH using Public Key Authentication – details in a separate email
- Data from TB_2021 stored in a location accessible to everyone with an account