The GRAND off-line software

Lech Wiktor Piotrowski

Faculty of Physics, University of Warsaw, Poland

05.VI.2025, GRAND meeting, Warsaw

GRANDlib

GRANDlib is our off-line software for (among others):

- Data reading and writing (DOI/AOI)
- Adapting simulations to GRAND (sim2root, RFChain, see Matias' talk)
- Topography and coordinates conversions
- Database for storing data files information and the web interface for downloading (see Francois' talk)
- Event viewers (however, they need updates)
- Extra functions (that's needed, but missing)

GRANDlib: new dev branch!

Last week we've pushed a new dev branch, with significant changes

- Works with the newest ROOT 6.36.00 (no Conda yet, but hopefully soon)
- The latest and greatest RFChain
- Reads all the data (that I can read)
- Same stuff inside as used to generate GRAND sims by Lukas & Matias
- Some cleanup (but much more pending)

We are still solving some instabilities, thus no official announcement yet

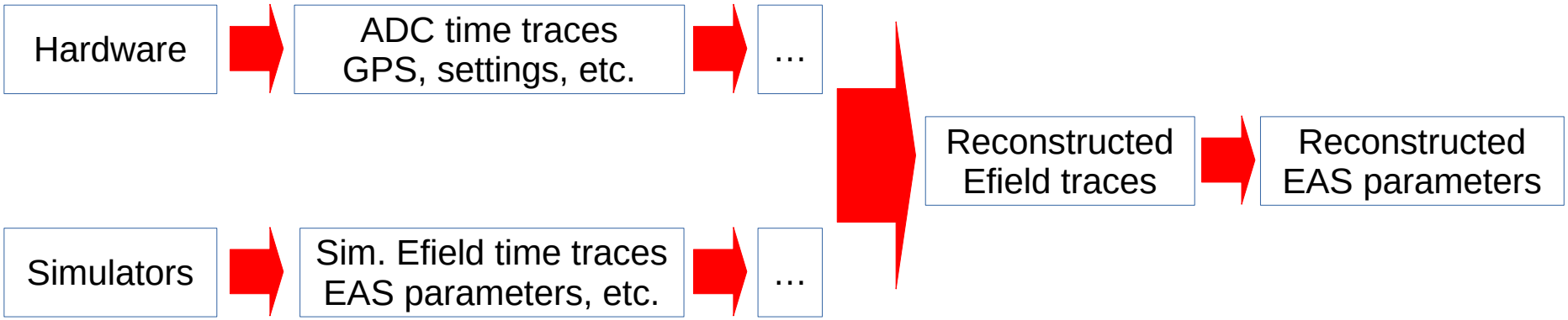
Software tasks and status

	Title	Assignees	Status	Priority
1	Merge of dev_sim2root_merge__merge_with_dev_fix_fields into dev #113		Done	
2	Fields incorrectly filled or mismatch with GrandRoot definition in simulations #99	mjtueros	Done	P3
3	Improving Analysis Oriented Interface	lwpiotr	In Progress	P3
4	Review merge dev_sim2root and dev_io_root → dev → master	lwpiotr and mjtuer...	Done	P2
5	Updating the (Ramesh's) EventViewer for GRANDROOT file format	claireguepin	In Progress	P2
6	Pipeline for RFChain		Todo	P1
7	Developing unit-tests for GRANDlib		Todo	P1
8	Continuous integration of GRANDlib on Github	luckyjim	In Progress	P1
9	Improving DataFile/DataDirectory in DOI	lwpiotr	Done	P1
10	Add multi files handling to sim2root	lwpiotr	Done	P1
11	Maintaining the docker image	lwpiotr and mjtuer...	In Progress	P1
12	Writing/updating the documentation		In Progress	P1
13	Develop an automatic pipeline for search for and analysis of UHECR events		Todo	P1
14	Developing functions for GRANDlib		Todo	P0
15	Dumb-proofing GRANDlib		Todo	P0
16	Type-hinting GRANDlib		In Progress	P0
17	Moving sim2root to descriptors		Todo	P0
18	Estimating if GtoT would be feasible in Python		Todo	P0
19	Validation tests for physics in continuous integration		Todo	P0
20	Trigger + digital filter implementation in GRANDlib		In Progress	P0

New task: Convert GULL/Turtle to Python (Ramesh)

The data processing chain

The structuring of data in the experiment is dictated by the data sources and processing steps:



This leads to a physical structure of trees that we have implemented.

Data Structure

We have many TTrees. They can be divided along 3 axes:

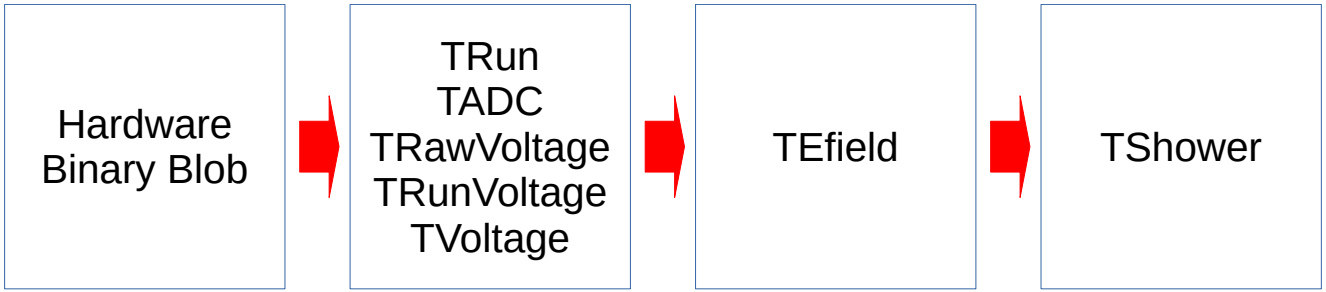
Variability constant through a run (trun*), constant through an event (t*(without "run"))

Type of data ADC counts (t*adc), (raw) voltage (t(raw)*voltage) or electric field measurements (t*efield), shower parameters (t*shower) information about noise (t*noise)

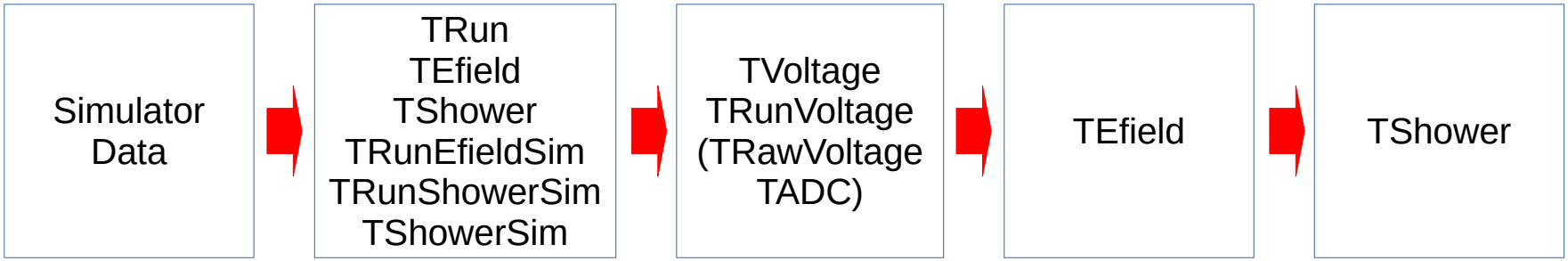
Source of data common to both hardware and simulators (t*(without "sim")), or simulator only (t*sim)

Analysis chains

Hardware data analysis chain:



Simulated data analysis chain:



The idea is, that the (most important part of) the data is generator-agnostic.

(see talk by Matias for more info)

Data Oriented Interface (DOI)

Direct Python interface to TTrees (fast access)

- A "Dataclass" for each tree type (camel-case: tefieldsim → TEfieldSim)

Data Oriented Interface (DOI)

Direct Python interface to TTrees (fast access)

- A "Dataclass" for each tree type (camel-case: tefieldsim $\rightarrow$ TEfieldSim)
- Class hierarchy: DataClass $\rightarrow$ MotherRunTree, MotherEventTree $\rightarrow$ TRun*, T($\sim$Run)*

Data Oriented Interface (DOI)

Direct Python interface to TTrees (fast access)

- A "Dataclass" for each tree type (camel-case: tefieldsim → TEfieldSim)
- Class hierarchy: DataClass → MotherRunTree, MotherEventTree → TRun*, T(∼Run)*
- Traces are 3D lists
  - The indexing is: trace[DU_number][arm/channel][time_bin]
  - The trace length not necessarily constant
  - → the array not necessarily rectangular
  - → NumPy array not working well

Traces are Python 3D lists now

- Keep it as is
- Could be numpy array of objects
- Could be awkward arrays

Awkward arrays are my favorite

Data Oriented Interface (DOI)

Direct Python interface to TTrees (fast access)

- A "Dataclass" for each tree type (camel-case: tefieldsim $\rightarrow$ TEfieldSim)
- Class hierarchy: DataClass $\rightarrow$ MotherRunTree, MotherEventTree $\rightarrow$ TRun*, T($\sim$Run)*
- Traces are 3D lists
- Mostly custom fields

Data Oriented Interface (DOI)

Direct Python interface to TTrees (fast access)

- A "Dataclass" for each tree type (camel-case: tefieldsim $\rightarrow$ TEfieldSim)
- Class hierarchy: DataClass $\rightarrow$ MotherRunTree, MotherEventTree $\rightarrow$ TRun*, T($\sim$Run)*
- Traces are 3D lists
- Mostly custom fields
- Whole file/directory access: DataFile/Directory

Data Oriented Interface (DOI)

Direct Python interface to TTrees (fast access)

- A "Dataclass" for each tree type (camel-case: tefieldsim → TEfieldSim)
- Class hierarchy: DataClass → MotherRunTree, MotherEventTree → TRun*, T(∼Run)*
- Traces are 3D lists
- Mostly custom fields
- Whole file/directory access: DataFile/Directory
- Analysis levels

We can have several trees of the same type in a file/directory

- They differ in "analysis level"
- L1 data is the (like) the data that comes from hardware without processing
- L0 data is clean data from sims
- L>1 data would be processed

For example, in DC2 sims:

- efield_l0 is clean signal from Zhaires/Coreas (with regularised trace length)
- efield_l1 pretends (would be in the future) to be reconstructed

Data Oriented Interface (DOI)

Direct Python interface to TTrees (fast access)

- A "Dataclass" for each tree type (camel-case: tefieldsim → TEfieldSim)
- Class hierarchy: DataClass → MotherRunTree, MotherEventTree → TRun*, T(∼Run)*
- Traces are 3D lists
- Mostly custom fields
- Whole file/directory access: DataFile/Directory
- Analysis levels
- Event indices

- An event is defined by (run_num, event_num) index pair
- Such indexes in a file do not have to be continuous
- get_event(run_num, event_num) requires knowing what event you are looking for
- get_entry(i) just loads an event with an ordinal number i in the file
- To iterate consecutively, better use built-in iterators

GtoT – Grand to Trees

Grand to Trees (GtoT) – A C++ program for converting hardware binary blob to GRANDROOT files

- Needs to be really fast: thus C++
- Handles both firmware v1 (GP13) and v2/3 (GRAND@Auger) hardware blobs

Problems:

- Some of the GRANDROOT classes written twice: in Python and C++
- More difficult maintanance, checking compatibility
- More difficult to monitor the data format version
- One day I'll check if GtoT could be as fast in Python as in C++ (any help?)

From GRAND files to GRAND directories

One very important change is pending:

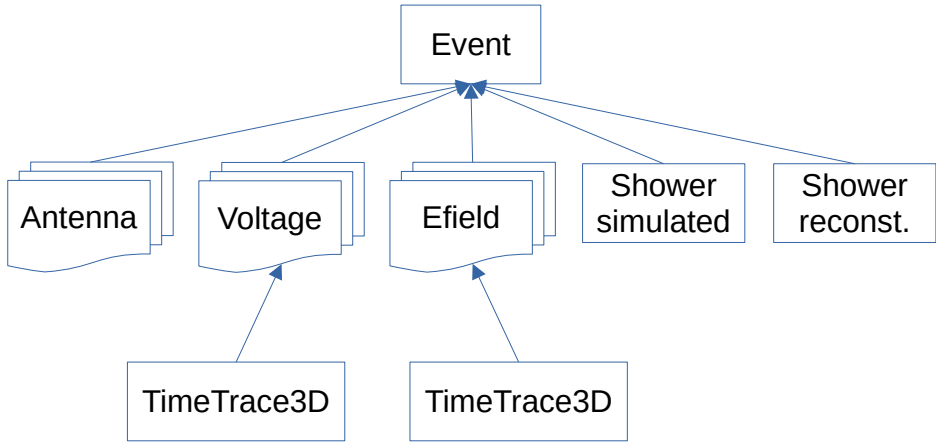
- Now GRAND experimental data distributed (still) as single ROOT files with many trees
- GRAND simulation data distributed as GRAND directories with many ROOT files, one tree type per file
- We want to move the experimental data to GRAND directory format
- **The problem:** I already work with GRAND Directory (see confirmed_events), but the data is distributed as files

Hopefully the data distribution can catch up soon!

Analysis Oriented Interface (AOI)

An interface for event-by-event analysis

- Assumption: analysis operations slow $\rightarrow$ AOI does not have to be fast
- The main class – Event – contains the most important data for analysis



Analysis Oriented Interface (AOI)

An interface for event-by-event analysis

- Assumption: analysis operations slow $\rightarrow$ AOI does not have to be fast
- The main class – Event – contains the most important data for analysis
- EventList – the main class for iterating through events

Analysis Oriented Interface (AOI)

An interface for event-by-event analysis

- Assumption: analysis operations slow $\rightarrow$ AOI does not have to be fast
- The main class – Event – contains the most important data for analysis
- EventList – the main class for iterating through events
- "The most important data" – what data is missing? Need your feedback!

Summary

We have a lot:

- GRANDlib is moving forward
- Members use GRAND data (exp and sims) in the GRAND ROOT format
- GtoT converts hardware blobs to GRANDROOT on daily basis
- GRAND sims with RFChain has been created and are being both used and updated
- Our files are downloadable from a DB

But:

- We need UHECR dedicated functions in GRANDlib (have you checked how to add your function?)
- We need testing and reports
- We need documentations and automatic tests

"Everyone wants the software to be ready and tested, nobody wants to do the software, or to test it"

Matias Tueros

We need **you** to help us with software development and testing!