CHIMERA

AN OFFLINE PROGRAM FOR DATA RECONSTRUCTION OF SABRE POP AND FSE

L. Bignell, I. Bolognino, S. Copello, G. D'Imperio, D. D'Angelo, A. Duffy, W. J. Dix, G. Lane, G. Lawrence, C. Li, P. Montini, F. Nuti, Suerfu, V. Toso, P. Urquijo

SABRE coll. meeting
LNGS, 4-6 Oct 2017

What are we talking about?

- SaberDAQ program saves a rawdata file composed of:
  - a run header with coded config info.
  - plain stream of digitizers: a list of 12 bit signal sample.
- SaberDAQ status:
  - Suerfu is restructuring the code.
  - We are debugging the run header and the buffer reading.

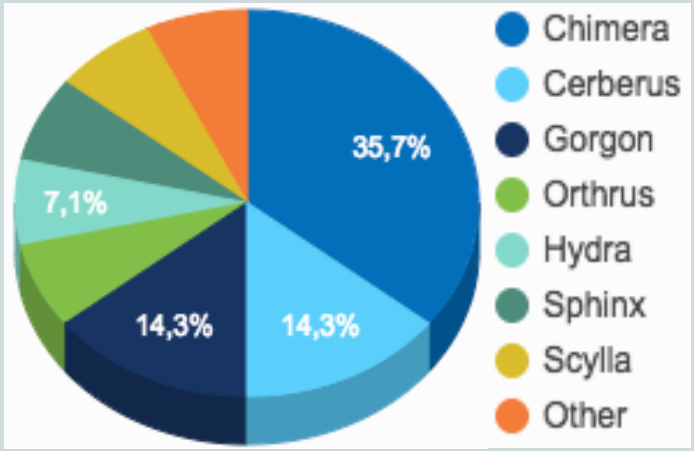
  } We're in good shape
- An offline reconstruction is needed to:
  - read rawdata file(s) and decode information.
  - read from database calibration parameters (pedestals, gains, time offsets, …) and applies all necessary corrections.
  - extract physics events from waveforms.
    - Flags noise pulses.
  - Computes (anti-)coincidences among channels and sub-detectors.
  - Computes parameters for analysis:
    - pulse shape
    - asymmetry
  - Computes the event energy estimators.

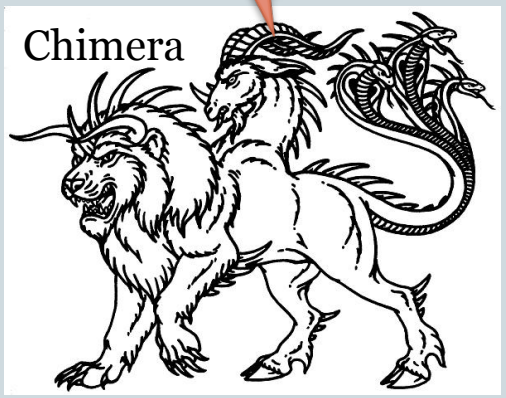
Requirements for reconstruction code

- Modular design
- Allow concurrent algorithms
- Data encapsulation (easy debug)
- User friendly and developer's friendly
  - Documented
- DB interface provided
- User interface provided (I/O: config and messages)
- Eventually serve as online monitor

The genesys

- We started in July 2017 from the empty framework of Echidna
  - (Borexino reconstruction code. Framework by A. Razeto, D. D'Angelo)
- Echidna in greek mythology is the mother of all monsters
- We hold a poll to decide the name, among Echidna's offspring:
  - 14 people voted
  - most voted monster: Chimera (5 votes)



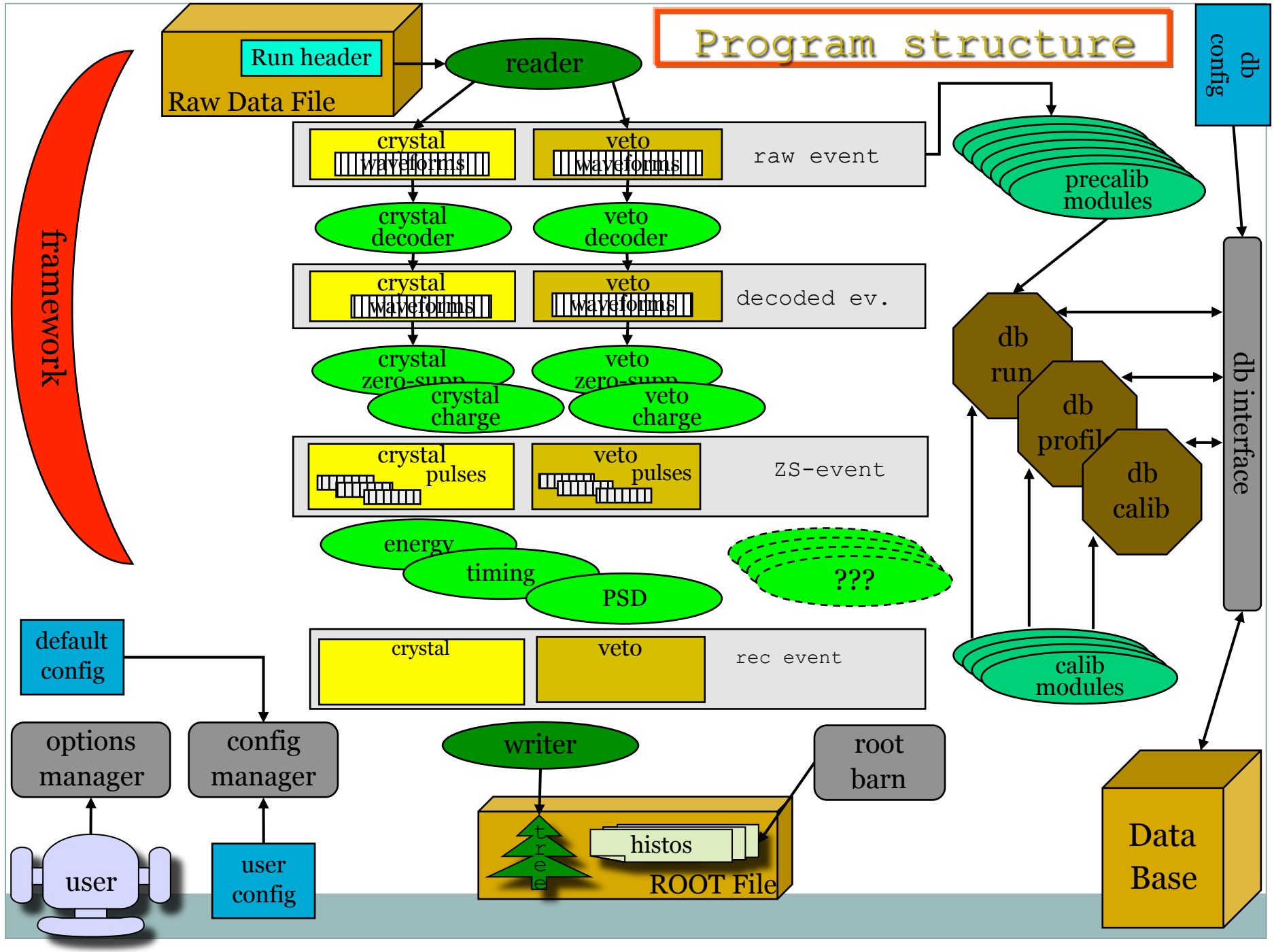


The genesys

- Is this too scary?
- But Echidna is also a cute anteater
  - common in Australia!
- Perhaps we can use a more friendly icon…

Echidna

Chimera


Program structure
framework
Run header
Raw Data File
reader
crystal waveforms
veto waveforms
raw event
crystal decoder
veto decoder
crystal waveforms
veto waveforms
decoded ev.
crystal zero-supp
veto zero-supp
crystal charge
veto charge
crystal pulses
veto pulses
ZS-event
energy
timing
PSD
???
crystal
veto
rec event
db config
precalib modules
db run
db profil
db calib
db interface
calib modules
default config
options manager
config manager
user
user config
writer
root barn
t r e e
histos
ROOT File
Data Base

Chimera key features

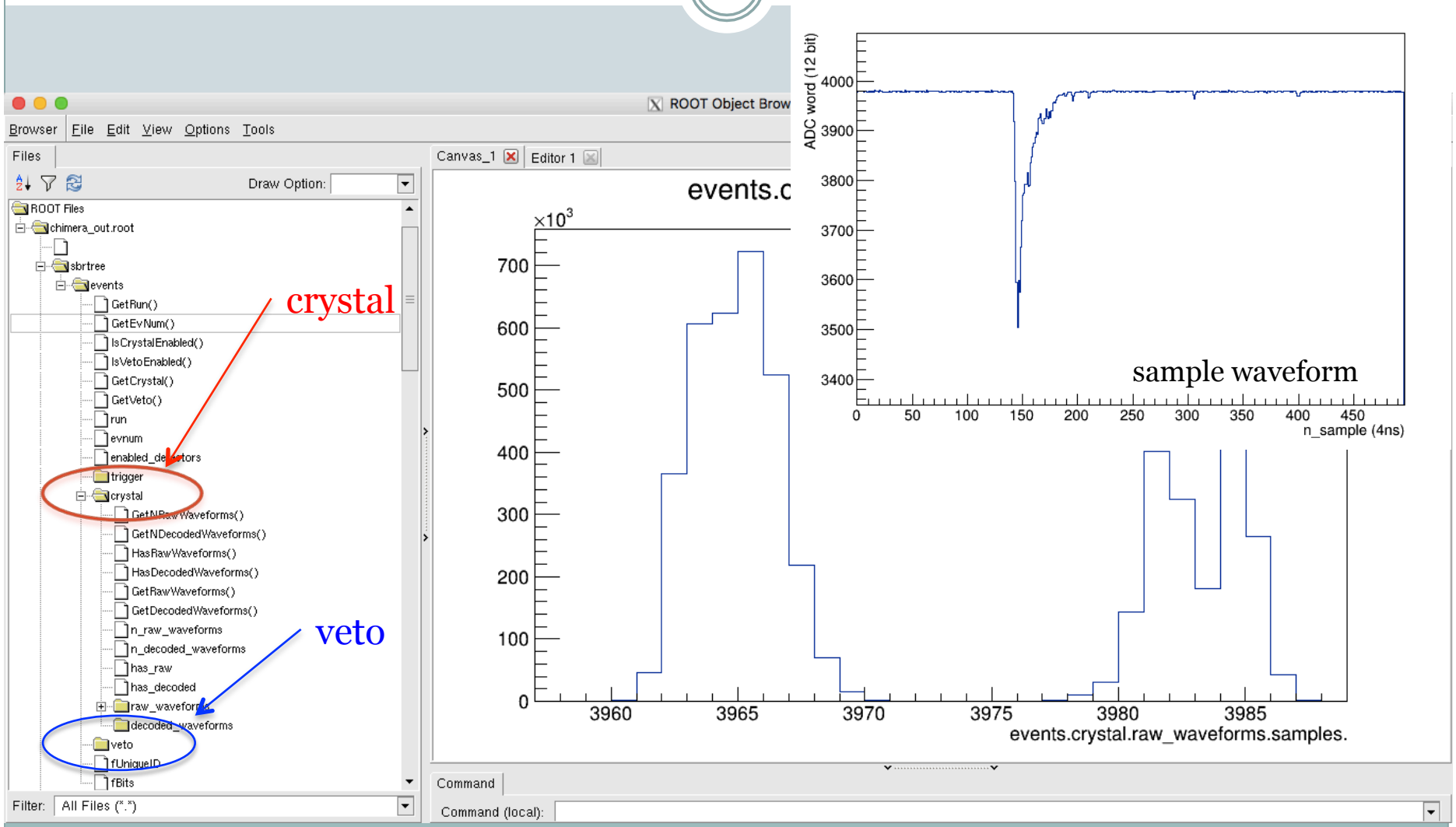
- Object-oriented (C++)
- Modular design
  - Allow concurrency of key algorithms
- Data encapsulation
  - every module can write only a dedicated part of the event.
  - debugging a module is independent from others and from framework.
- User friendly and developer's friendly
- User interface provided
  - Input: config parameters -> customize your module's behavior without recompiling.
    - Multi-layer: default file, user file, command line
  - Output: screen messages and log file.
    - Multi-level: log -> critic with exception handling
- ROOT-based.
  - Tested with ROOT 5.34 and 6.02.
  - Output file containing
    - ROOT tree. Event reflects detector sub-structure and reconstruction stages.
    - ROOT barn (histograms)
- Scalable: with different configuration can process data from: PoP and FSE (North and South).

To be done:

- Documentation
  - User's guide
  - Developer's guide
- DB interface
- All physics modules
- Precalibration (run based) and calibration (periodic) modules
- Also run as online monitor?

The fun starts now!

Embryo version working



How to proceed

A large group of people from every institution contributes to the physics modules:

- Decoders (crystal, veto)
  - from ADC samples to mV and time.
- Baseliner
  - baseline computation and subtraction
- Zero-suppression
  - from waveforms to pulses
- Energy estimators
- Timing (crystal, veto and relative)
- Pulse shape
- Precalibration tasks:
  - channel alignment
  - pulse-height calibration
- Calibration tasks
  - TBD
- Modules to run as online monitor?

Please do not write your own program:
implement all your algorithms in Chimera, it's super easy!

How to write a module

- Infrastructure makes your life easy!
  - Read "Module's Programmer Guide"
  - Watch test_module example.
- You only need to implement 3 methods:
  - begin(): resource allocation
  - doit(): event-level computation
  - end(): resource deallocation, run-level computation, info save.
- Event interface:
  - A pointer to event is received in doit()
  - Write only in dedicated areas.
- Database interface (to be done)
  - I/O with dedicated methods
- User interface:
  - get_parameter("par_name");
  - get_message(level) << "I love you" << dispatch;
- ROOT histograms storage in ROOT barn.

Choices being discussed

- Where do we store rawdata?
- Where does Chimera execute?
  - Lngs computing cluster? (current tests)
  - CNAF?
  - dedicated machine(s)?
- Where do we store rootfiles?
- Where is the DB and which architecture?
  - LNGS can provide this service in the Computing Centre
    - but their policy is not host daq-critical services in the outside lab.
  - Foresee a future mirror?
- How do we handle backups?
- How do we handle file transfers?
  - need a dedicated fiber from the lab?
- Where do we keep the code?
  - Currently use GIT on several repositories (Princeton, Gitlab). Can we unify?
- Data mirror with Australia and/or Princeton?

Conclusions

- Team is forming, please join!
- An working framework exists
  - available in offline git repostory: https://gitlab.com/SABRE-EXPERIMENT/Chimera (ask Giulia for permissions)
  - Check it out and send feedback
- Physics modules to be written by different developers
- Urgent to finalize rawdata format
  - effort ongoing
- Coordinate with SABRE-South DAQ choices to ensure Chimera can read those data too.