Online & Offline software at H.E.S.S.

ROOT User Workshop
15 October 2001

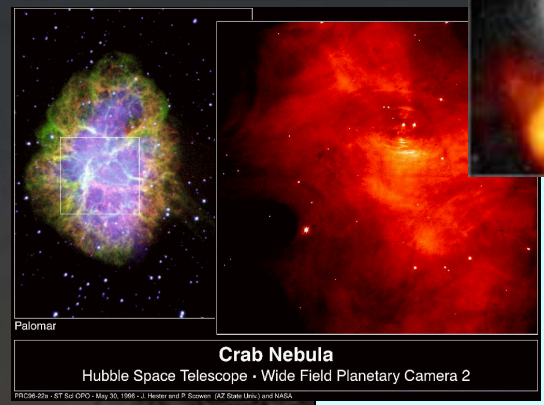
Mathieu de Naurois, LPNHE *Paris University VI/VII*
Christoph Borgmeier, Christian Stegmann, *Humboldt University Berlin*

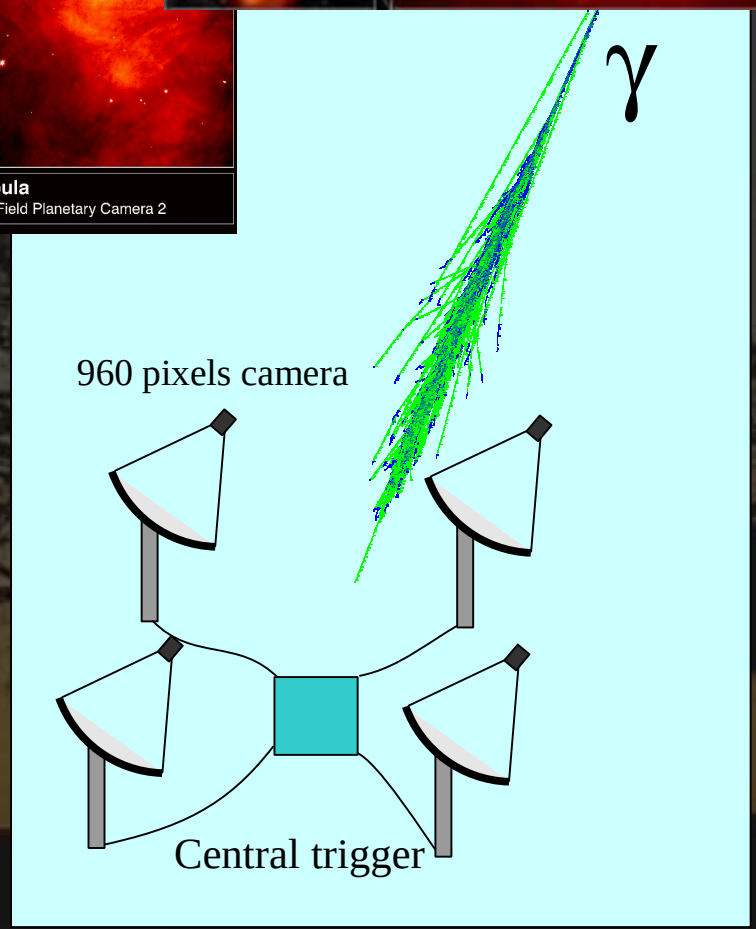
- The H.E.S.S. Experiment
- Data storage & Off-line software
- Acquisition software
- ROOT Problems/Wishlist

http://www.mpi-hd.mpg.de/hfm/HESS/HESS.html

The High Energy Stereoscopic System (H.E.S.S.)

- Observe γ-induced showers above 100 GeV
  - Active galactic nuclei
  - Pulsars & Plerions
  - Micro-quasars
  - ...
- Stereoscopy for 3D reconstruction
- Fast camera & DAQ for threshold
- Installed in Namibia (Gamsberg)



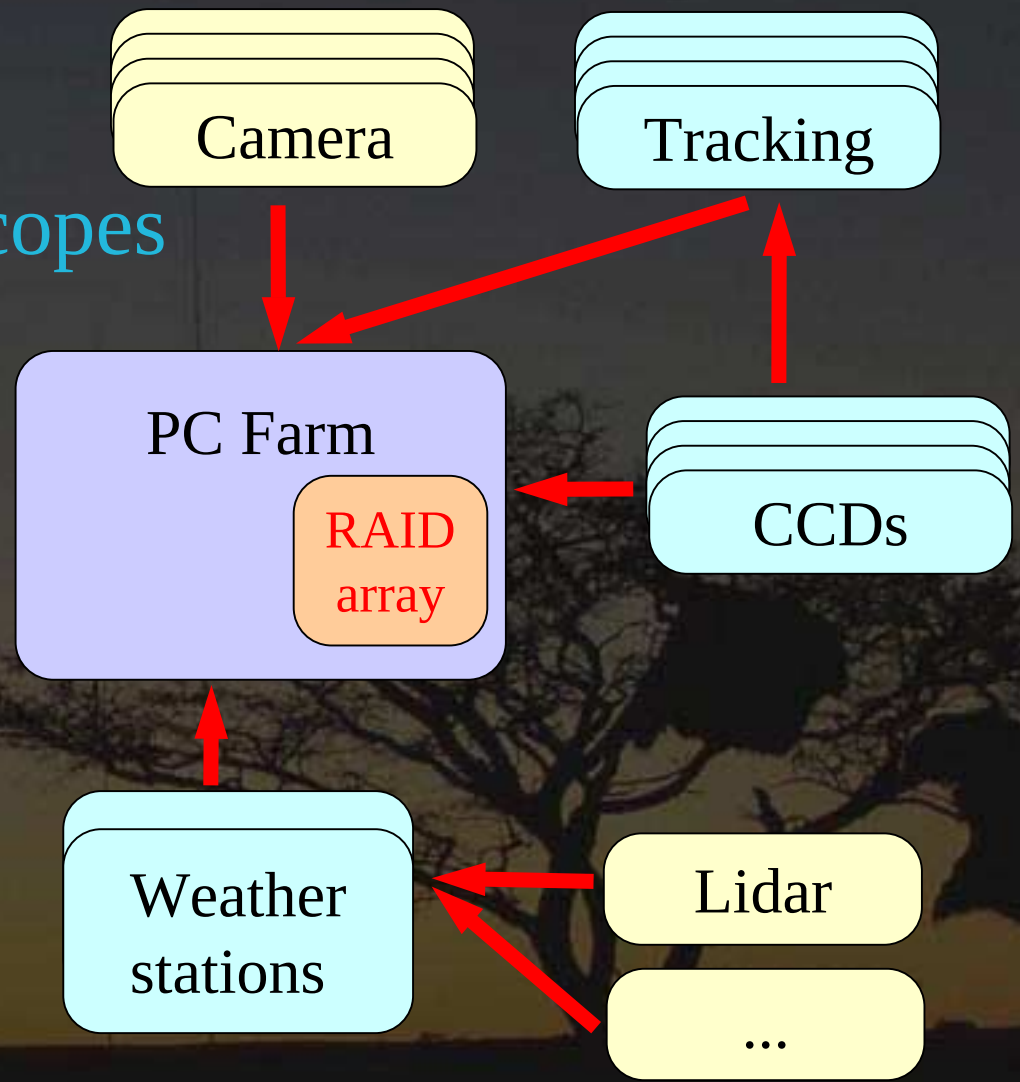


Current Status

- First telescope operational since June 2002
- Structure for the next 3
- Second camera : early 2003

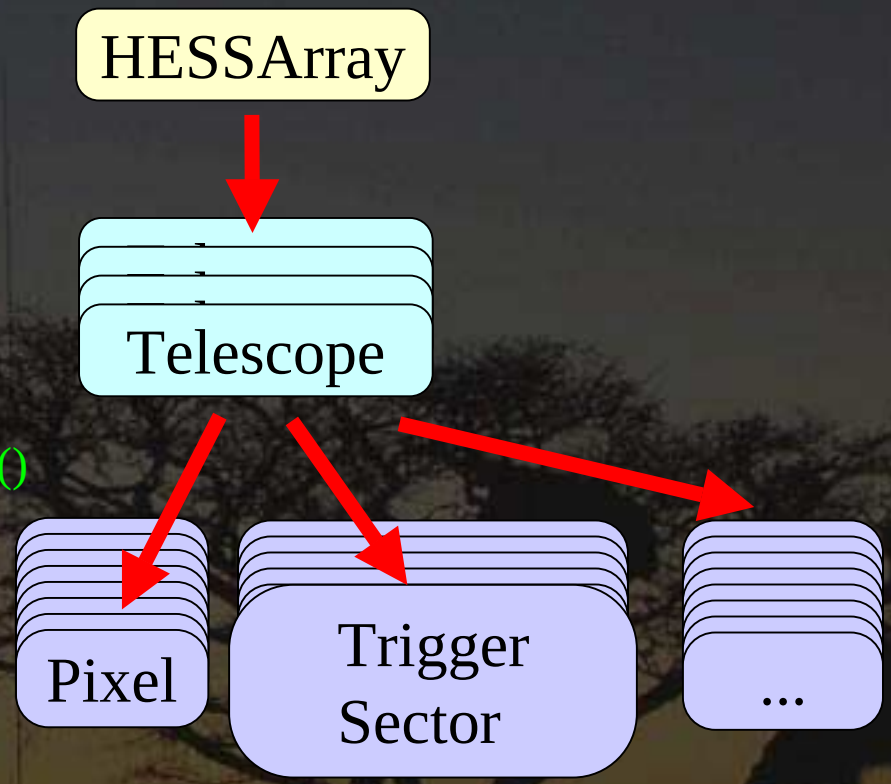
Data sources @ H.E.S.S.

- Camera events: 1kHz (6MB/s per camera)
- ~100 GB/night @ 4 telescopes
- Monitoring data (taken independently)
  - CCDs
  - Cloud scanner
  - Optical Telescopes
  - Telescope drives
  - ....



SASH: (*Storage & analysis software at H.E.S.S.*)
I – Container hierarchy

- Fixed container hierarchy
- No fixed numbering scheme
- All elements accessible from the top level container via iterators (no C pointers)
  Sash::Pointer<Sash::Pixel> Sash::Telescope::beginPixel()
- List, Sets, ... and Iterator provided for looping over elements
  Sash::List<Sash::Pixel> Pixel::Neigbours()

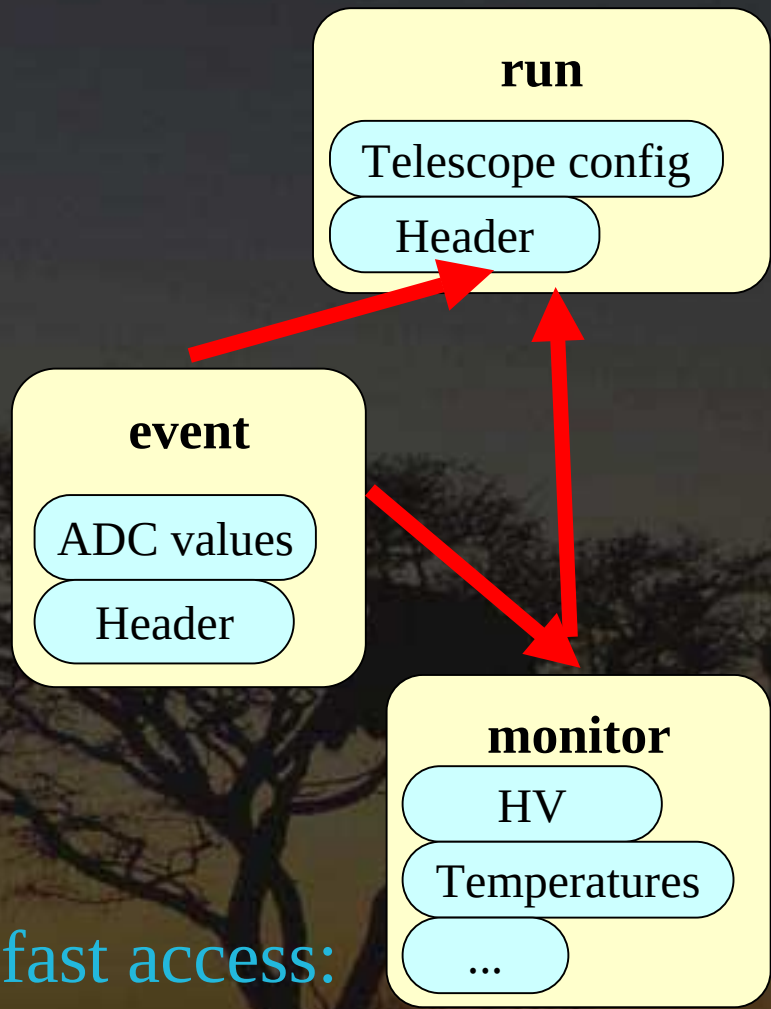


SASH
II – Data access

- Each data class knows its container type
- A helper template class registers a slot in the container for each data class
  Sash::EnvelopeEntry<T>
- Additional named slots allows coexistence of several instances ($\Rightarrow$ comparison of analysis methods,...)
- Data classes are created and accessed by the container (using CINT)
  Container::Handle<T>(parameters) and Get<T>()
- Data classes store their creation parameters (argument of Handle<T>) to enable automatic recreation at file readout
- Expandable (user class libraries)

SASH
III – Sash::DataSet

- Extension of ROOT TTree
  - Maps the data class at their proper location in the container hierarchy
  - Organises the data from the same TTree in TFolder (run,events,...)
  - Provides iterators (iterate on events)
- Provides a dependencies mechanism based on time stamp from several files
  (monitor information is loaded for each event)
- Uses the branch splitting mechanism for fast access: data are loaded only when accessed.

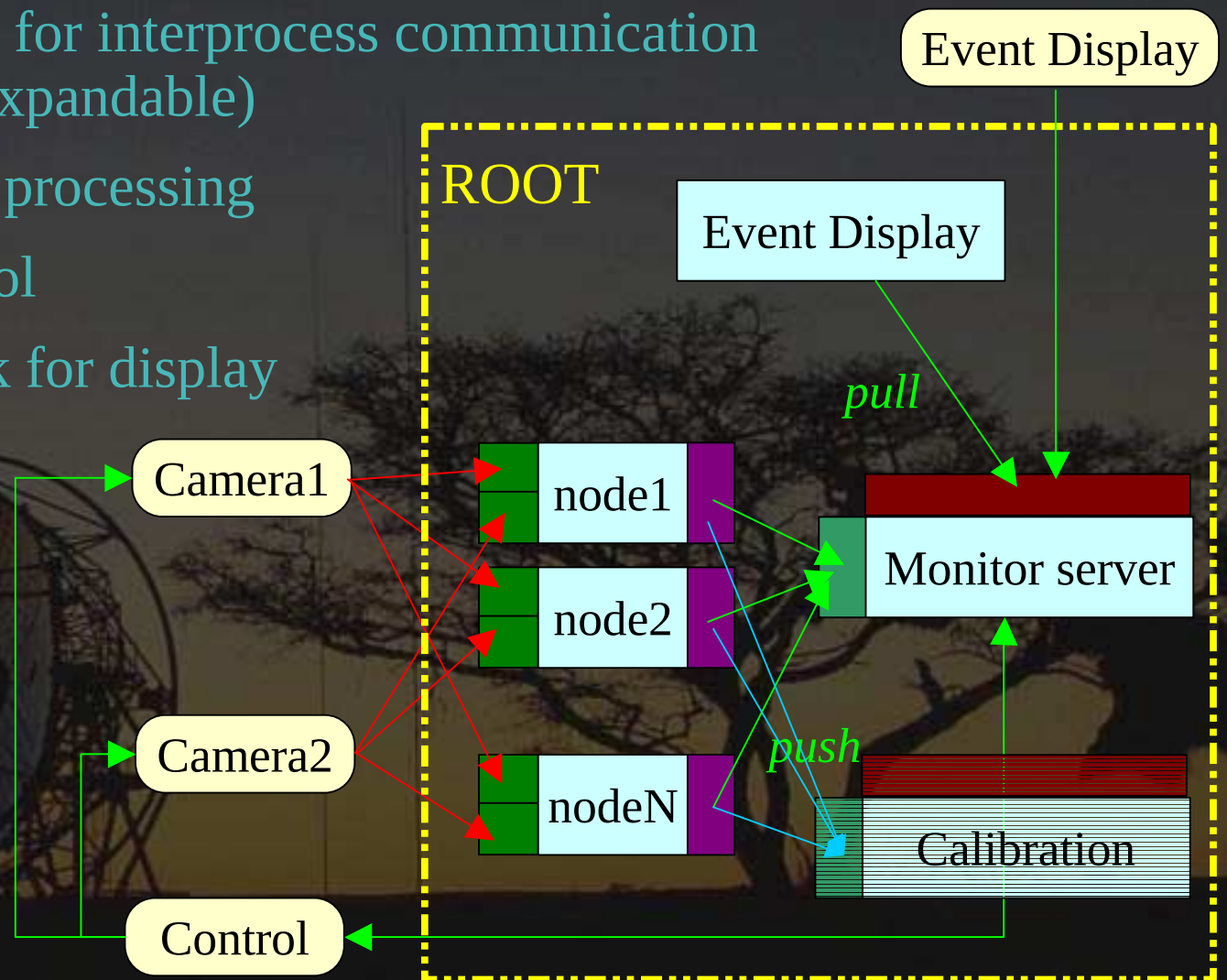


SASH
IV – Sash::Makers

- Common interface for data manipulation
  - Receive a Sash::DataSet and operates on it
  - Full access to the container hierarchy
  - Can be chained + call-back mechanism
- Two specific Makers:
  - Sash::DataSetIterator: loads an event and synchronises other DataSets (monitor,…)
  - Sash::TreeWriter: writes a event to disk

DASH: *Data acquisition software*

- Multi-processes, multi-threaded & distributed acquisition
  - CORBA (omniORB) for interprocess communication ($\Rightarrow$ modular, easily expandable)
  - ROOT for storage & processing
  - Python/Gtk for control
  - ROOT or Python/Gtk for display
  - Building blocks
    - Buffer
    - Server
    - Sender
    - Processor
    - ...
  - Push & Pull

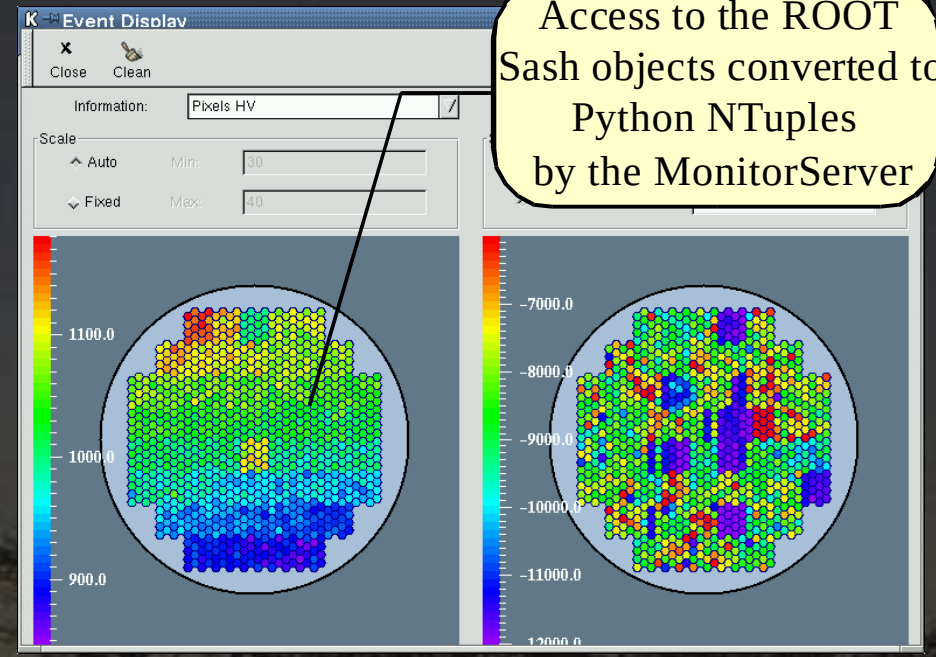


DASH II – Data Block Processor

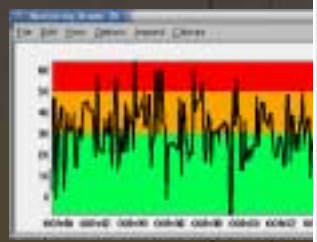
- Arbitrary byte sequence data (⇒ polymorphism)
  - CCD images or C Structures
  - Streamed ROOT objects
  - Whole folders
  - Automatic conversion into python arrays if corresponded function provided
- Data processed by Processor_i
- Three types of RootProcessor objects:
  - RootProcessor_i: arbitrary ROOT object, new memory location for each event
  - FixedRootProcessor_i<T>: non polymorphic but constant memory location
  - HESSArrayProcessor_i: maps the received objects/folders into the container hierarchy and call the registered Sash::Maker

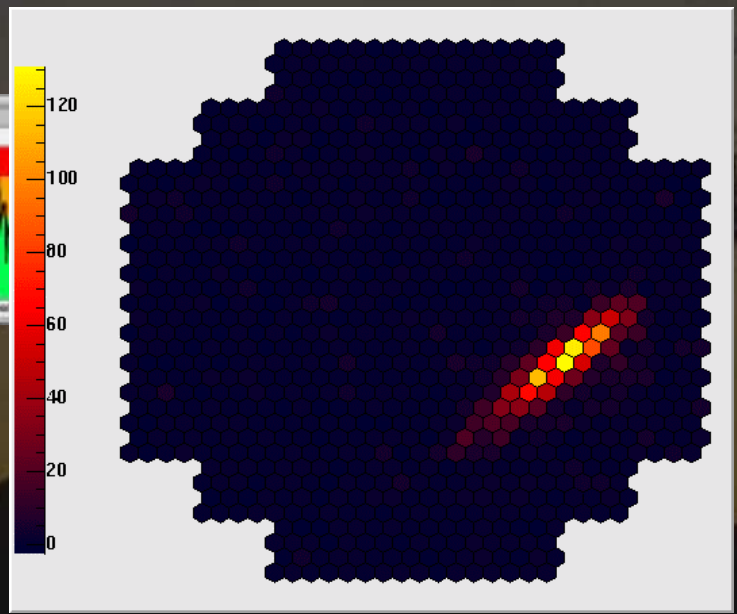
Dash III - Displays

- Pull mode
- Process ask for a folder or a specific Sash object
- Use the HESSArrayProcessor to run Sash::Maker's for incoming objects



Access to the ROOT Sash objects converted to Python NTuples by the MonitorServer





Summary

- Sash implements a general way to combine different ROOT trees (Sash::DataSet) from different files at a fixed memory location & automate tasks (Sash::Maker).
- Analysis/Calibration software consist of ~ 15 CVS modules based on Sash, ~ 10 developers
  (many people develop their own analysis module)
- DASH provides a class hierarchy of building blocks to organize the H.E.S.S. DAQ in a general way
  - CORBA protocol ⇒ transparently expandable ressources
  - Modularity (building blocks connected by CORBA)
  - Transport of ROOT objects
  - Highly multithreaded data transport, display and analysis software
  - Good performances (6MB/s on a single machine)

ROOT Problems/Wishlist

Last year problems:

- Namespace support
  Fixed
- Template support
  Almost fixed
- STL support
  much better
- Thread stability
  improved

WhishList:

Better thread stability

Threads created by CORBA, not by ROOT.
Too much use of global variables in ROOT (gFile,fgFitter,...)
Is Qt communication protocol thread-safe?

Support for templates member functions

```
class A {
  template <class T> f();
};
#pragma link C++ function A::f<MyClass>();
```

Support for class member type change in ROOT I/O

(when cast operator or conversion constructor provided)

Experience with ROOT

- Impressive support from the ROOT team:
  - Quick fixes
  - Fast developpement of requested features

Thank you