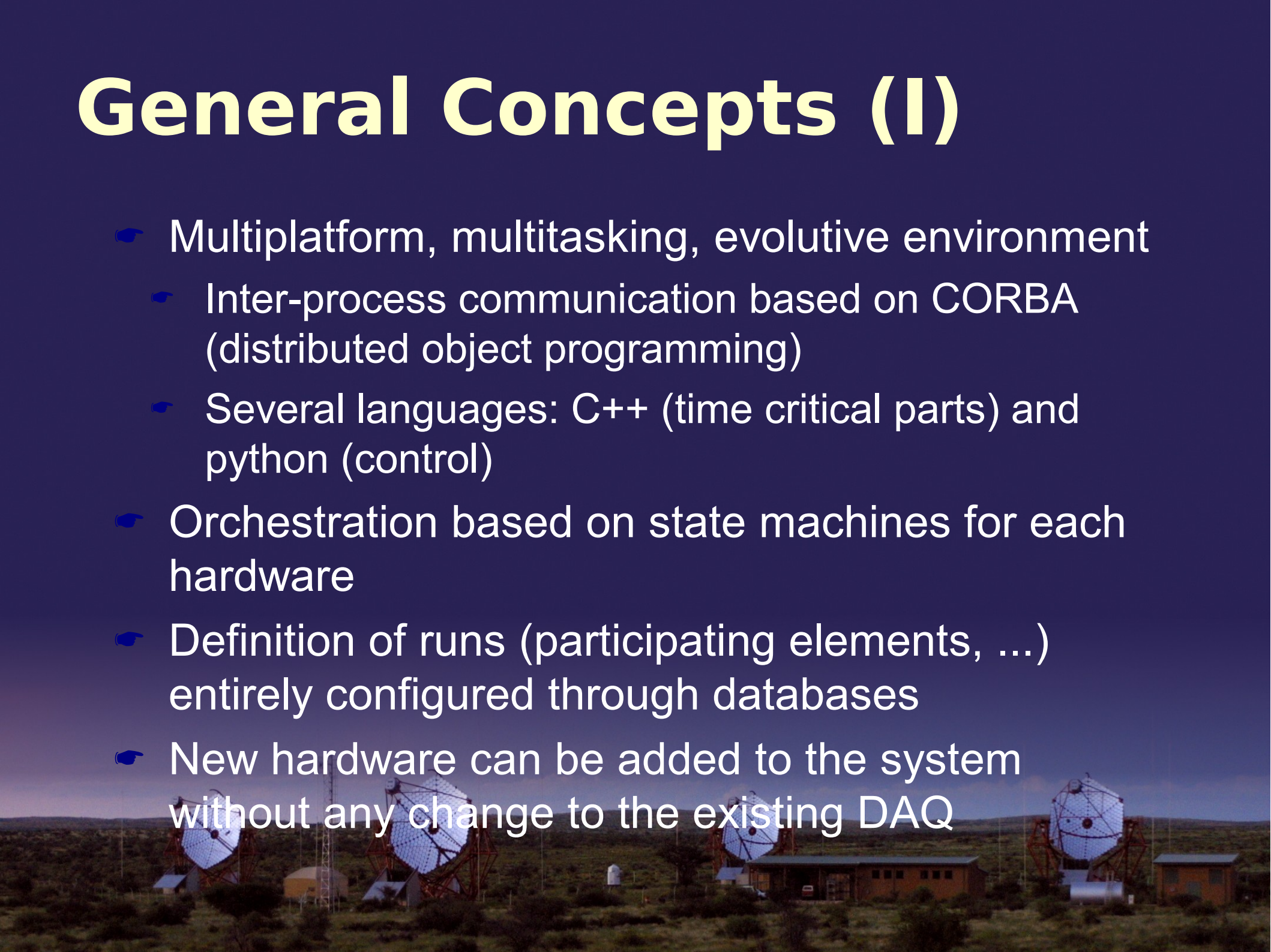
L'expérience de HESS

- Vue générale de l'acquisition
- Camera DAQ
- Event Builder

Data Acquisition

General Concepts (I)

- Multiplatform, multitasking, evolutive environment
  - Inter-process communication based on CORBA (distributed object programming)
  - Several languages: C++ (time critical parts) and python (control)
- Orchestration based on state machines for each hardware
- Definition of runs (participating elements, ...) entirely configured through databases
- New hardware can be added to the system without any change to the existing DAQ



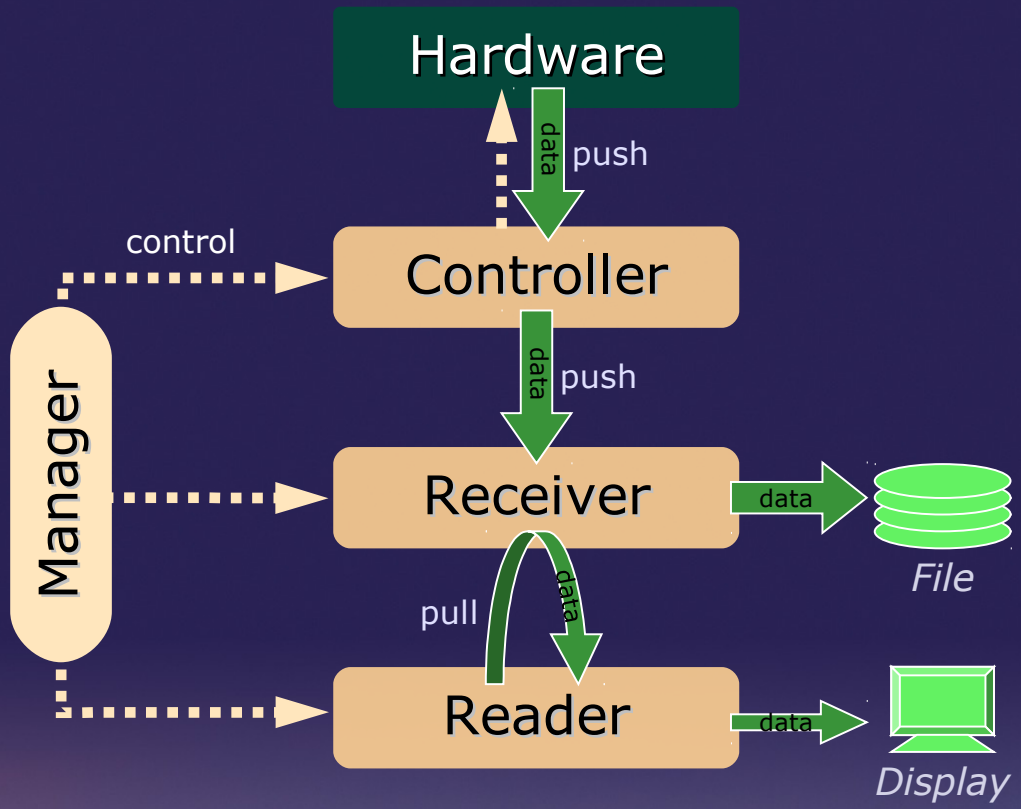
General Concepts (II)

- Data format based on ROOT
  - Introspection capabilities allows to access any member (for display, analysis, ...) without recompilation
  - No distinction between offline and online algorithms

General Concepts (II)

- Data format based on ROOT
  - Introspection capabilities allows to access any member (for display, analysis, ...) without recompilation
  - No distinction between offline and online algorithms

DAQ Building blocks



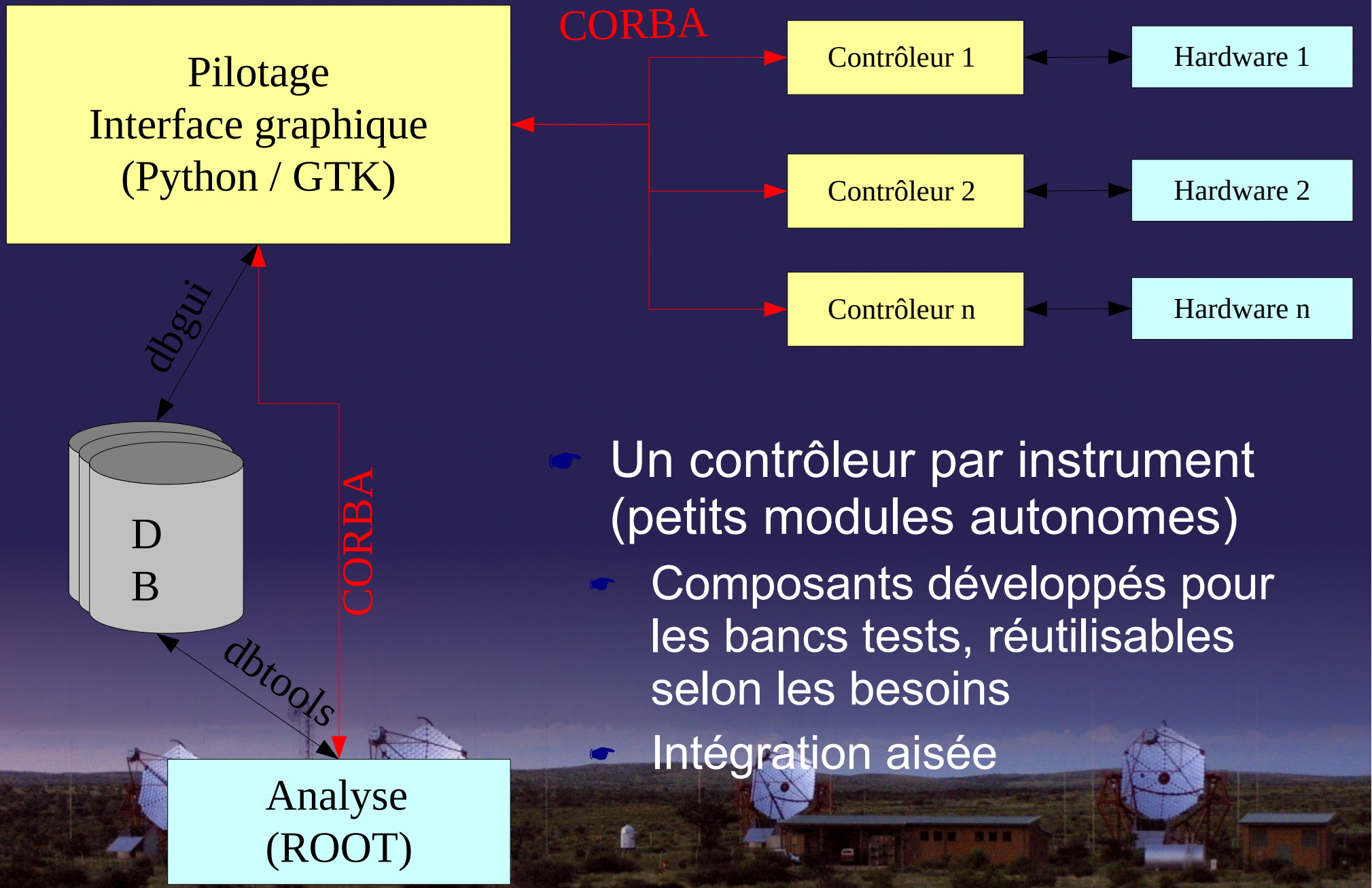
Dash provides all necessary classes and interfaces for the processes in the DAQ

Interprocess communication: CORBA (open source implementation omniORB)

Data processing chain:

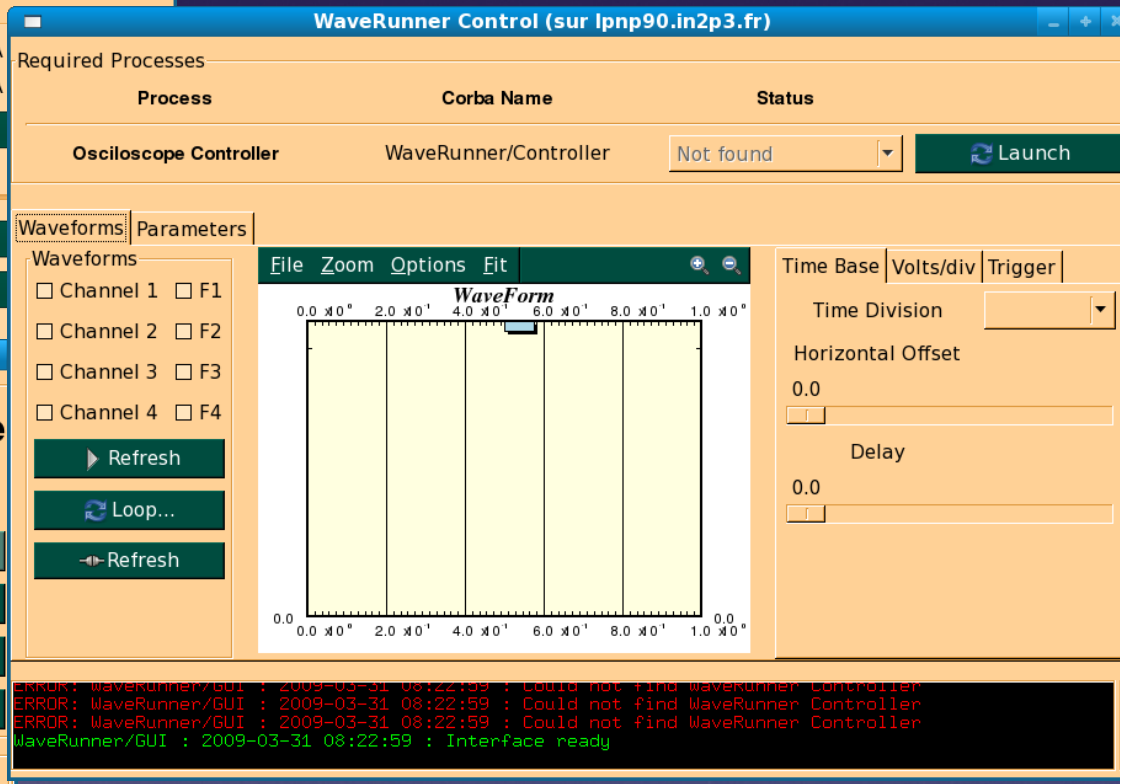
- *Controllers*
- *Receivers*
- *Readers*
- *Managers*

Philosophie générale



- Un contrôleur par instrument (petits modules autonomes)
  - Composants développés pour les bancs tests, réutilisables selon les besoins
  - Intégration aisée

Exemples d'interfaces



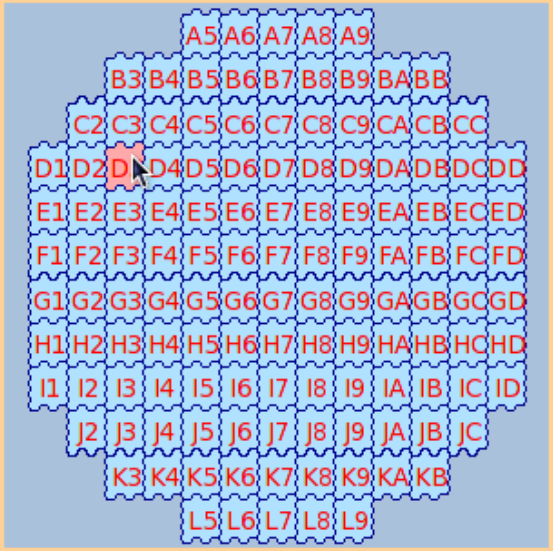
CT5 Camera Controller

File Options

Stopped

- GeneralWindow
- Config
- Pixels
- SAM Config
- Drawer Config
- Drawer
- Trigger Configurati
- TimeStamp Config
- Pre L2
- L2 Trigger
- Calibration
- Pedestal
- NDSearch
- Delay Calibration
- Pulse Shape
- Gain Search
- Gain Scan
- SinglePE
- Linearity
- AfterPulse Rate
- Main(Obsolete)
- Power & LID(Obsc
- Fan & Temp(Obsol
- Slow control(Obso
- Trigger(Obsolete)

Please click on a Drawer to see or modify the configuration of its pixels



Load from dbase

Save to dbase

Valid config

Valid current drawer config

SET HV ON All Drawers

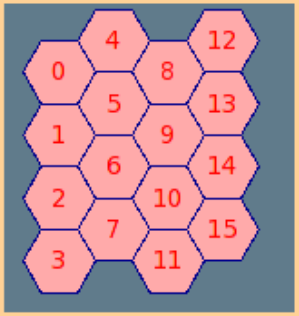
SET HV OFF All Drawers

SET HV ON selected Drawers

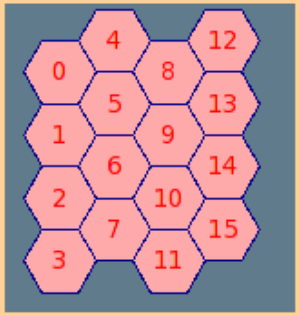
SET HV OFF selected Drawers

PM #	Serial #	HV (Set)	HV (Reac	DCI	HVI	Scaler	HV ON/OF	Trigger ON/OF
		All					ON OFF	ON OFF
D300	23387	907	0.0	0	0.0	0	ON	ON
D301	23567	863	0.0	0	0.0	0	ON	ON
D302	22041	853	0.0	0	0.0	0	ON	ON
D303	21502	870	0.0	0	0.0	0	ON	ON
D304	21631	867	0.0	0	0.0	0	ON	ON
D305	22727	882	0.0	0	0.0	0	ON	ON
D306	21745	861	0.0	0	0.0	0	ON	ON
D307	23207	876	0.0	0	0.0	0	ON	ON
D308	21490	872	0.0	0	0.0	0	ON	ON
D309	21754	877	0.0	0	0.0	0	ON	ON
D310	21864	873	0.0	0	0.0	0	ON	ON
D311	23103	874	0.0	0	0.0	0	ON	ON
D312	22998	866	0.0	0	0.0	0	ON	ON
D313	23173	868	0.0	0	0.0	0	ON	ON

HV ON/OFF



TRIGGER ON/OFF




```
CT5/Camera : 2010-01-14 17:36:27 : Starting GPIB/PowerController on n1n31
CT5/Camera : 2010-01-14 17:36:27 : Starting GPIB/GUI
CT5/Camera : 2010-01-14 17:36:27 : Starting utilities/Fitter_01 on n1n41
CT5/Camera : 2010-01-14 17:36:27 : Starting utilities/Fitter_02 on n1n41
```

DBMa... CT5 C...

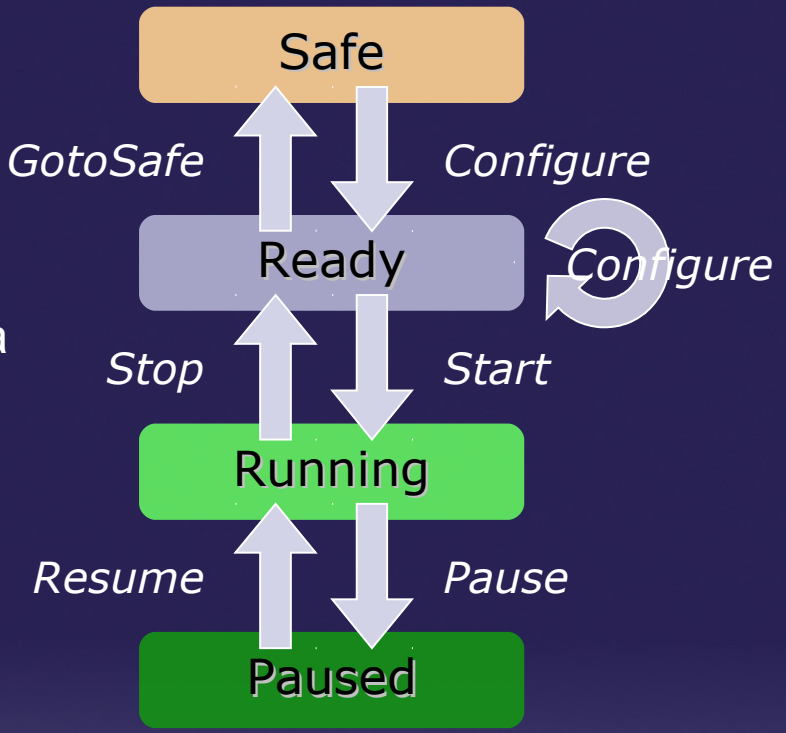
Process State Control

`StateControllers`:

- each process is a StateController
- generic states for all processes
- only certain transitions allowed

Example: Camera

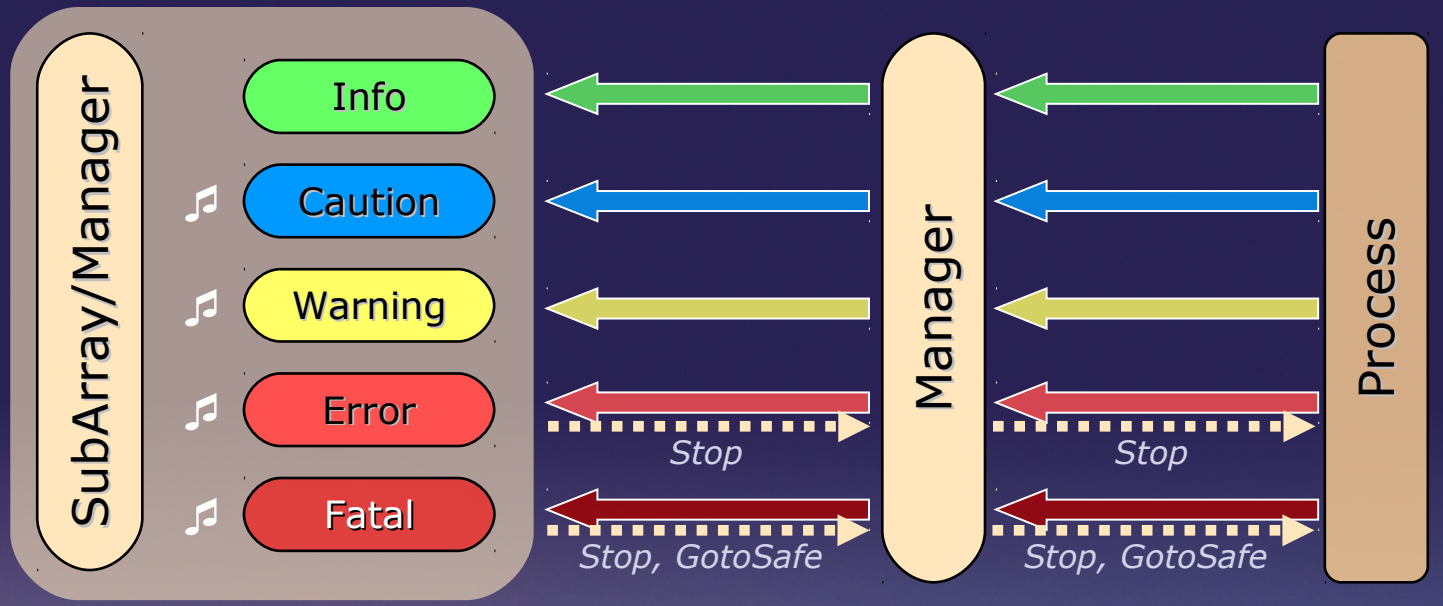
- States
  - *Safe*: power off
  - *Ready*: power on, lid open, send monitoring data
  - *Running*: HV and trigger enabled, send event data
- Transitions
  - *Configure*: read configuration from database, power on, open lid, reactivate PMTs
  - *Start*: enable HV, contact *Receiver*, start trigger
  - *Stop*: disable HV and trigger, send event data
  - *GotoSafe*: switch power off, close lid



Message Passing, Error Handling

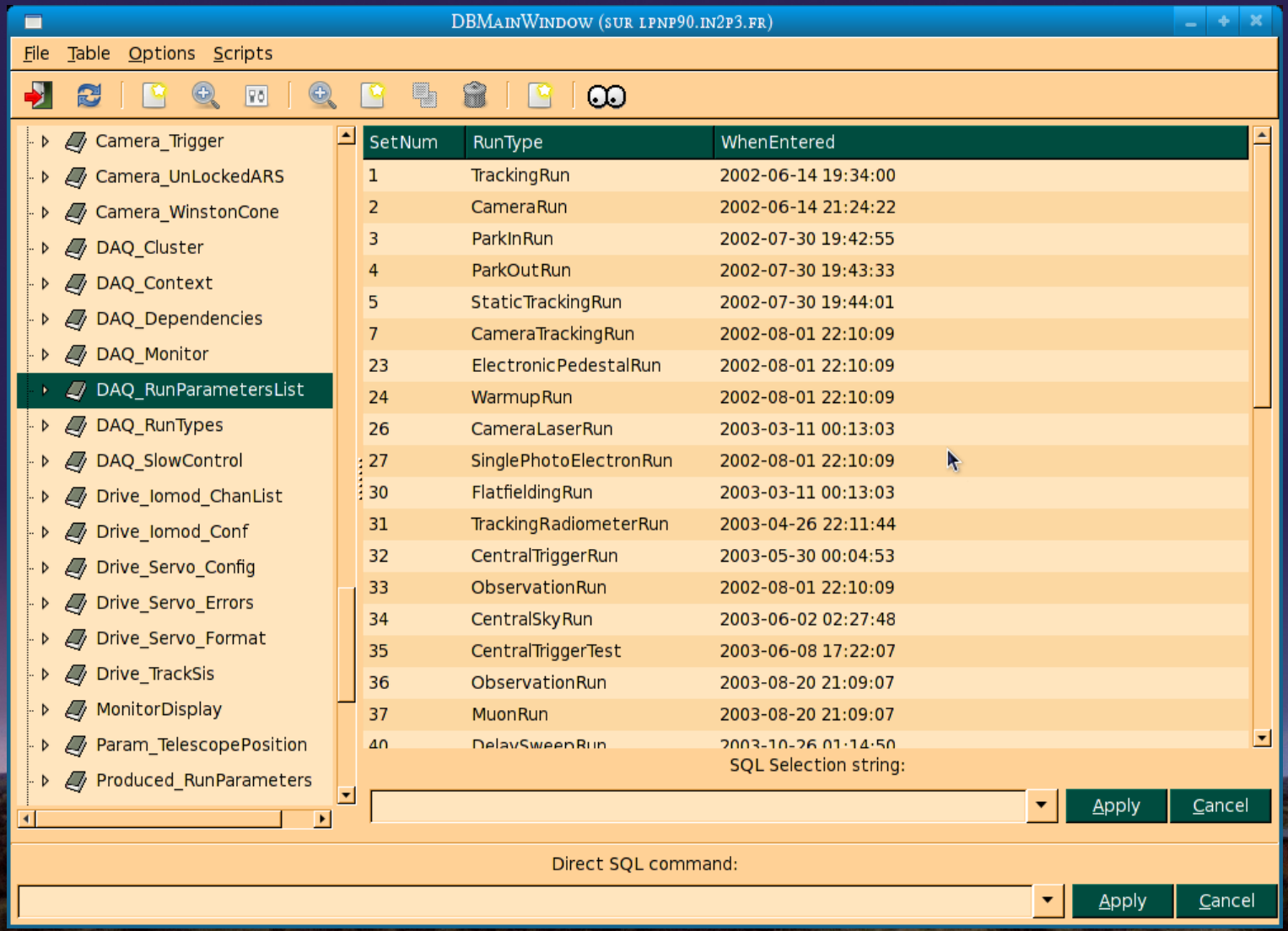
Message categories:

- Assign shift crew feedback or automatic action to certain message type
- Messages are passed through Manager hierarchy

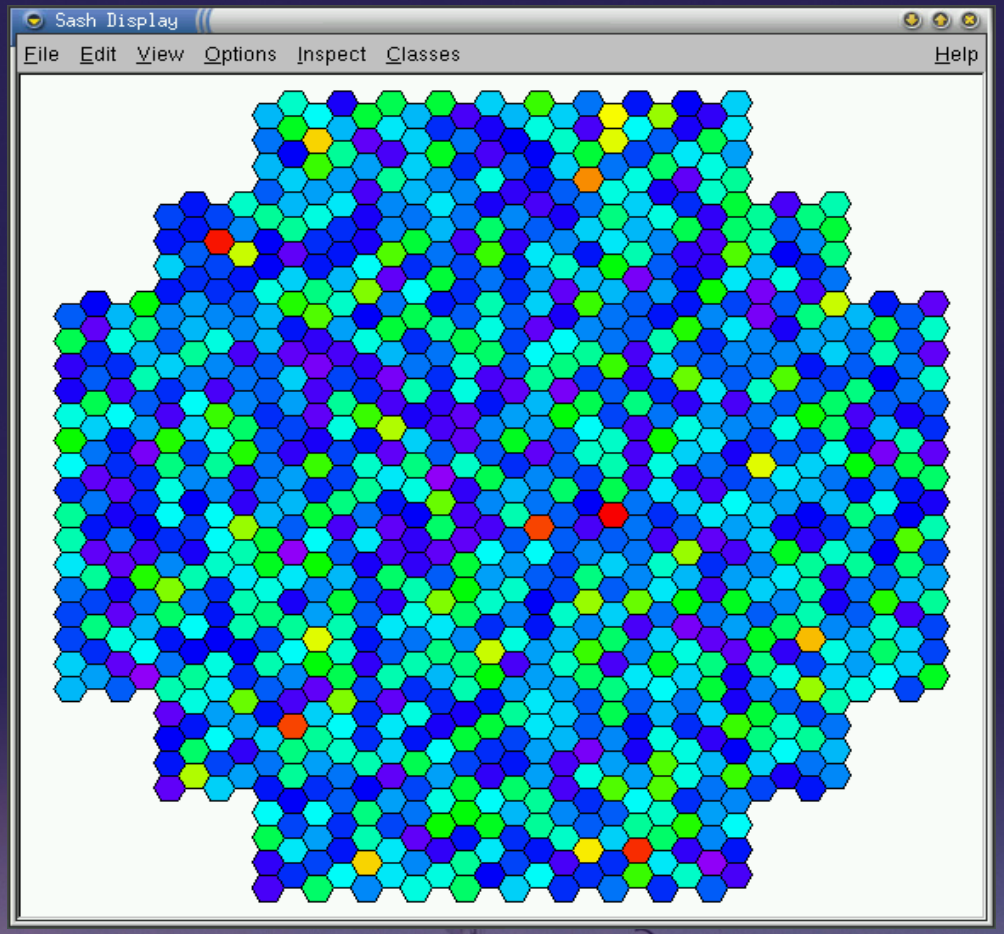


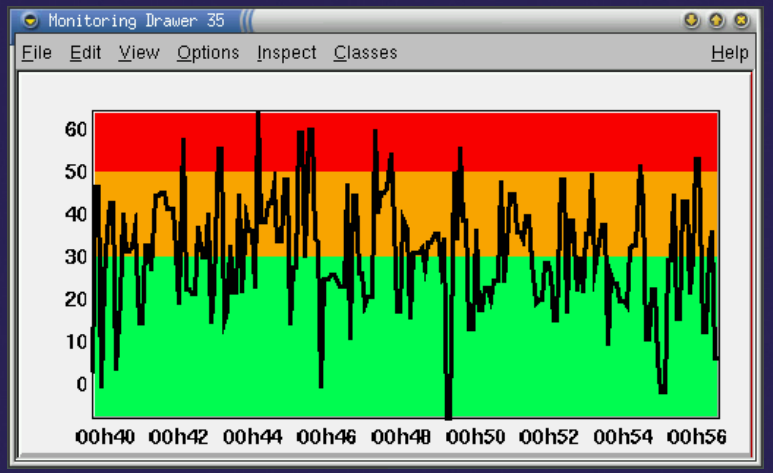
Configuration

- Paramètres de configuration associés à chaque contrôleur:
  - Valeurs par défaut spécifique de chaque type de run:



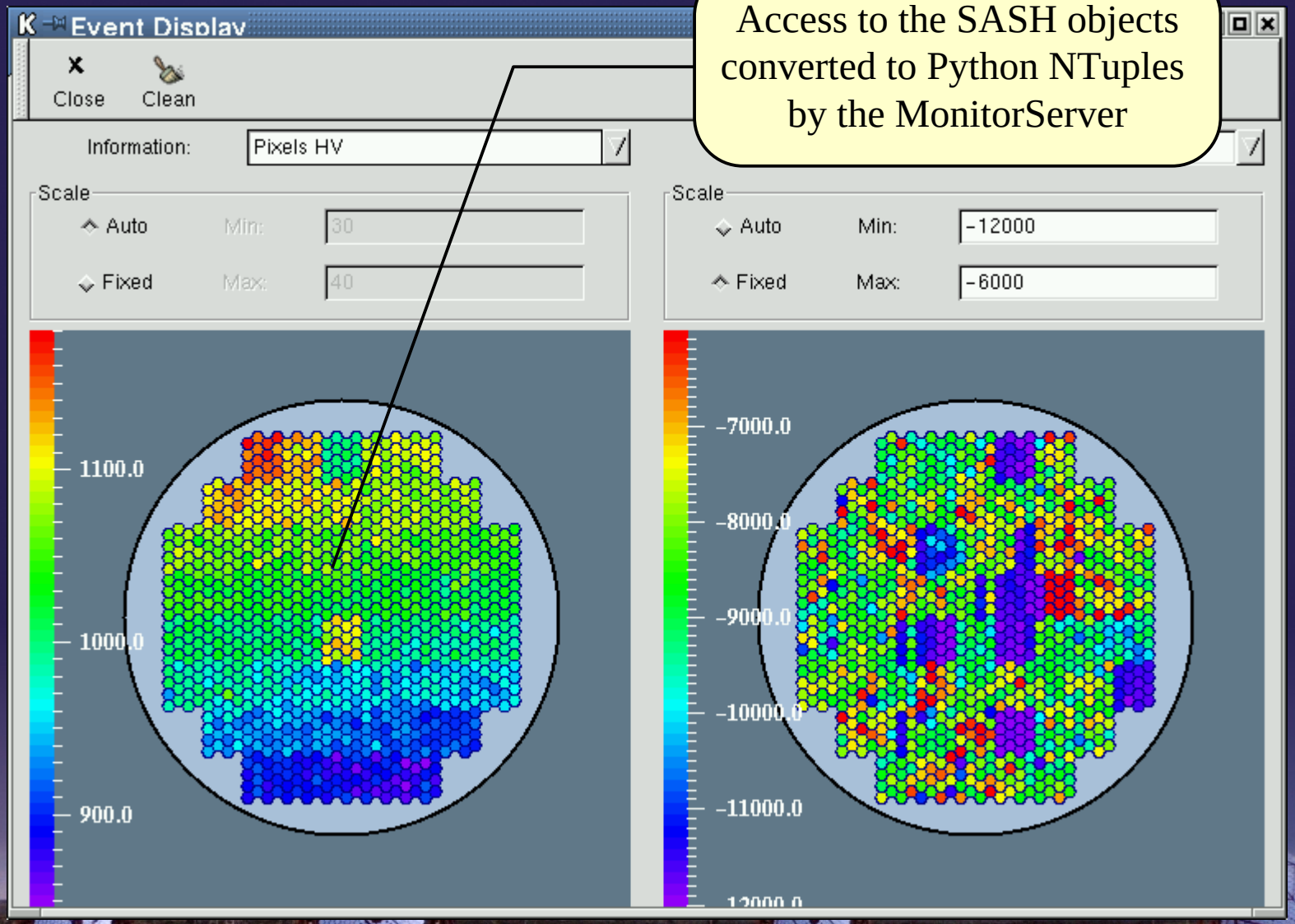
Monitor client: ROOT





- Uses Sash framework (Sash::CameraMonitorEvent,...)
- Handles history
- Configurable through databases

Monitor client: python



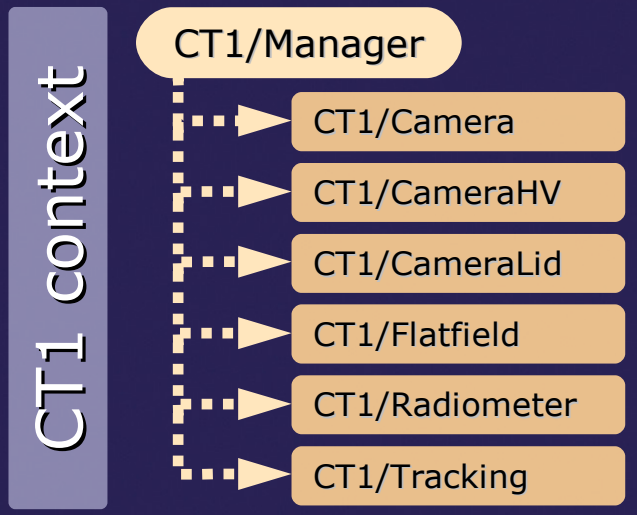
Access to the SASH objects converted to Python NTuples by the MonitorServer

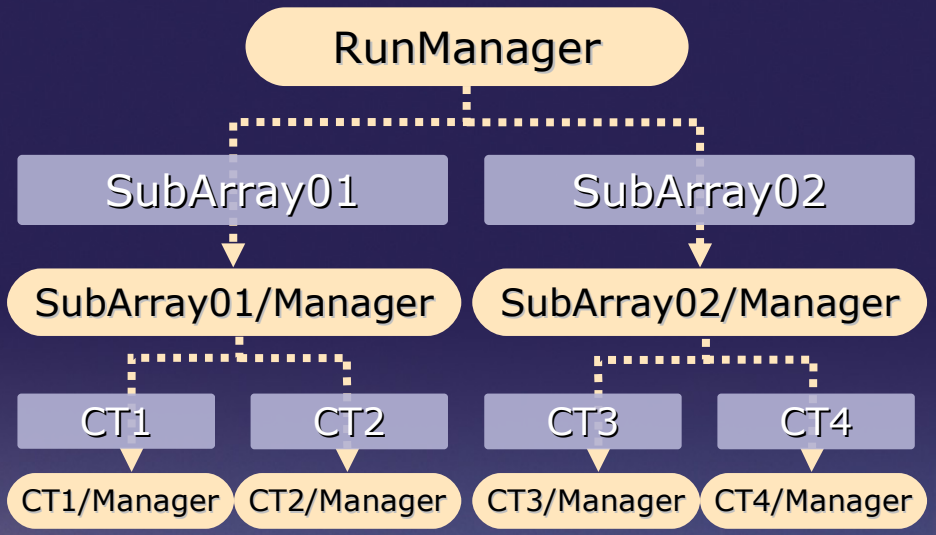
Run Control

Control of data taking:

- measurement organized in ***runs*** of 30 min duration
- assign set of configuration parameters ➲ ***run types*** e.g. *ObservationRun, FlatfieldingRun, …*
- Processes read their configuration according to run parameters in database
- *Managers* take over control of processes in *contexts*
- Parallel runs possible by using different contexts

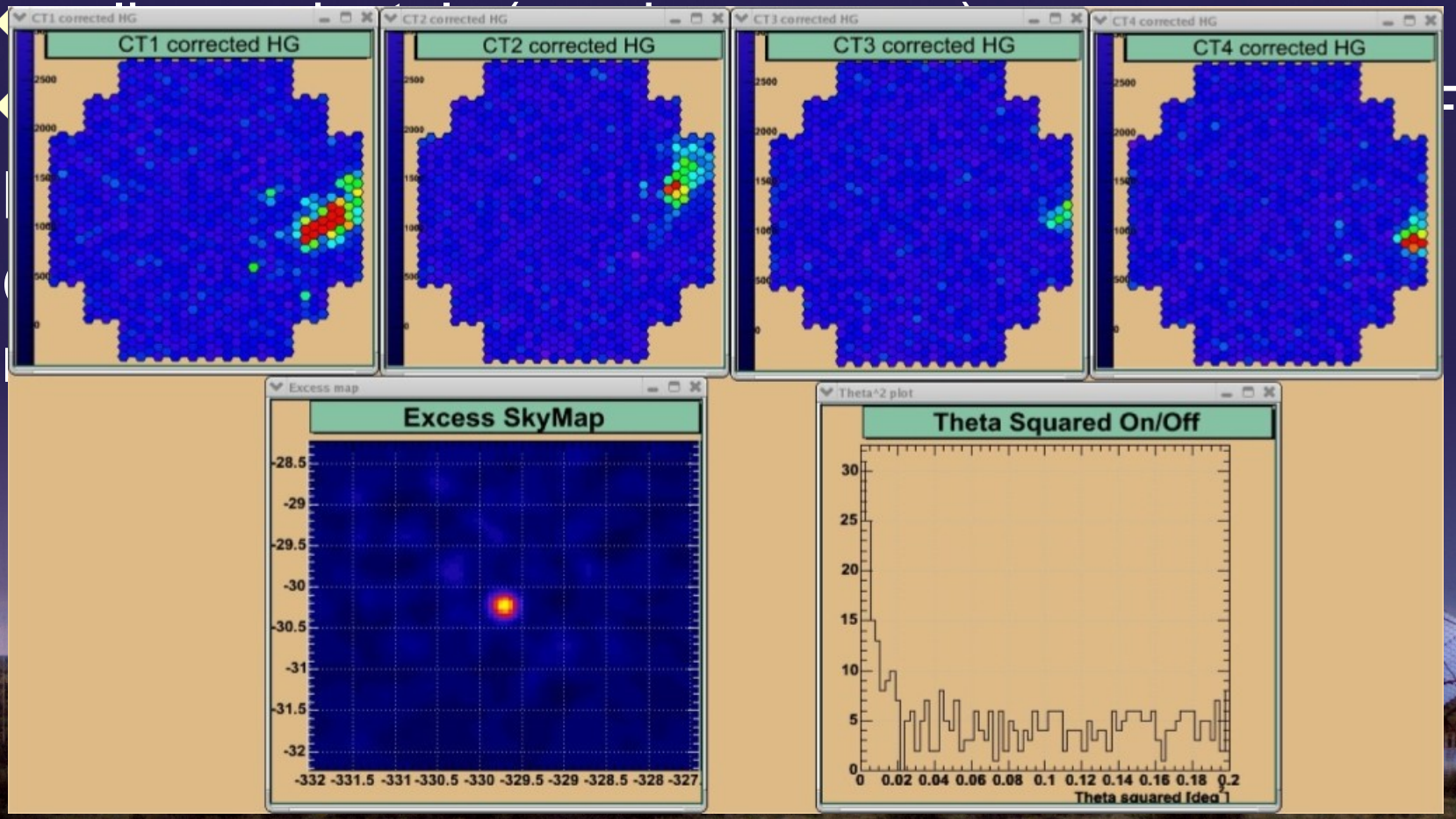
➲ SubArray/Manager controls all contexts in the run



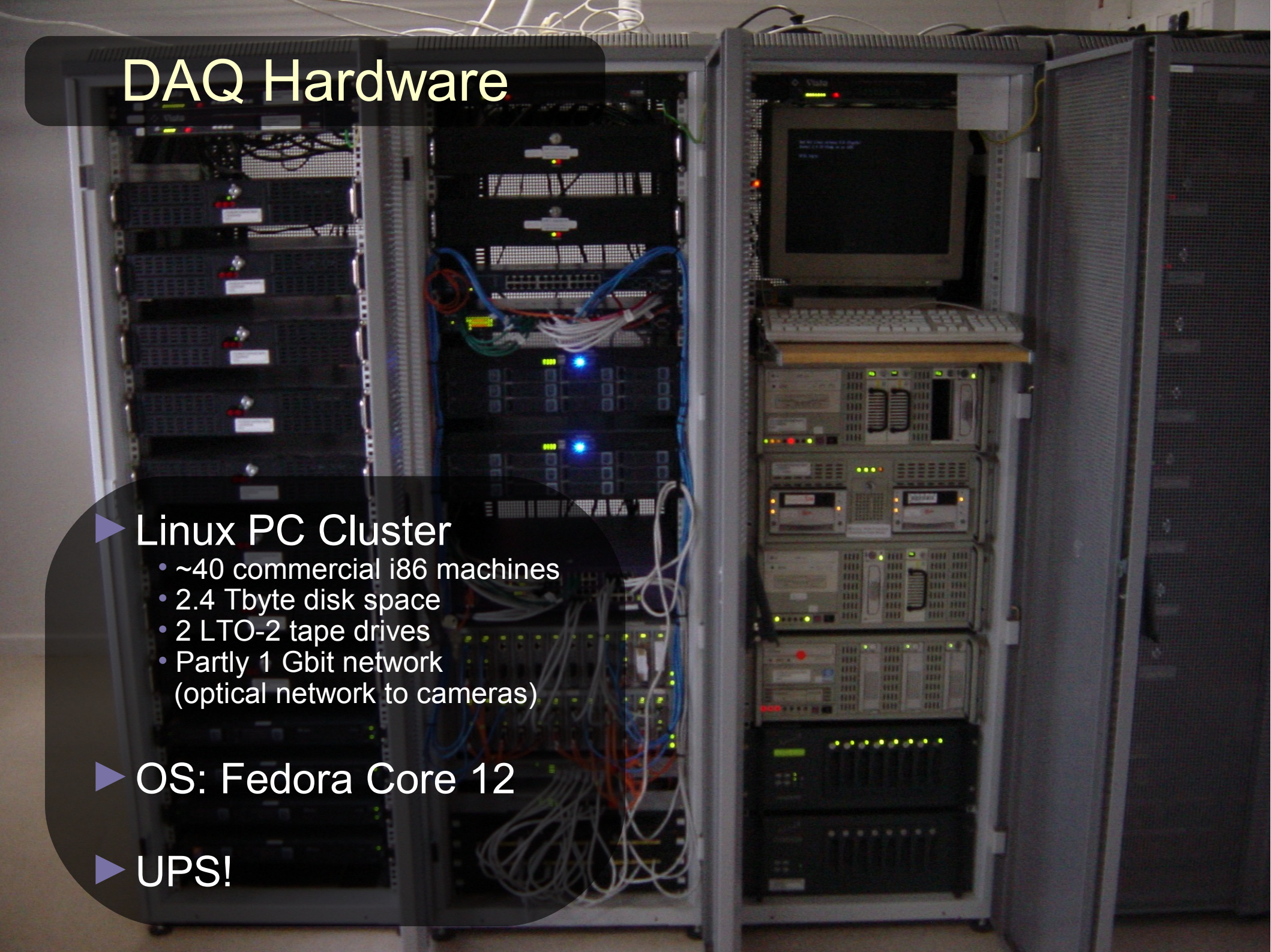


Online Analysis

- Running stable in parallel to data taking
- Performs online calibration:



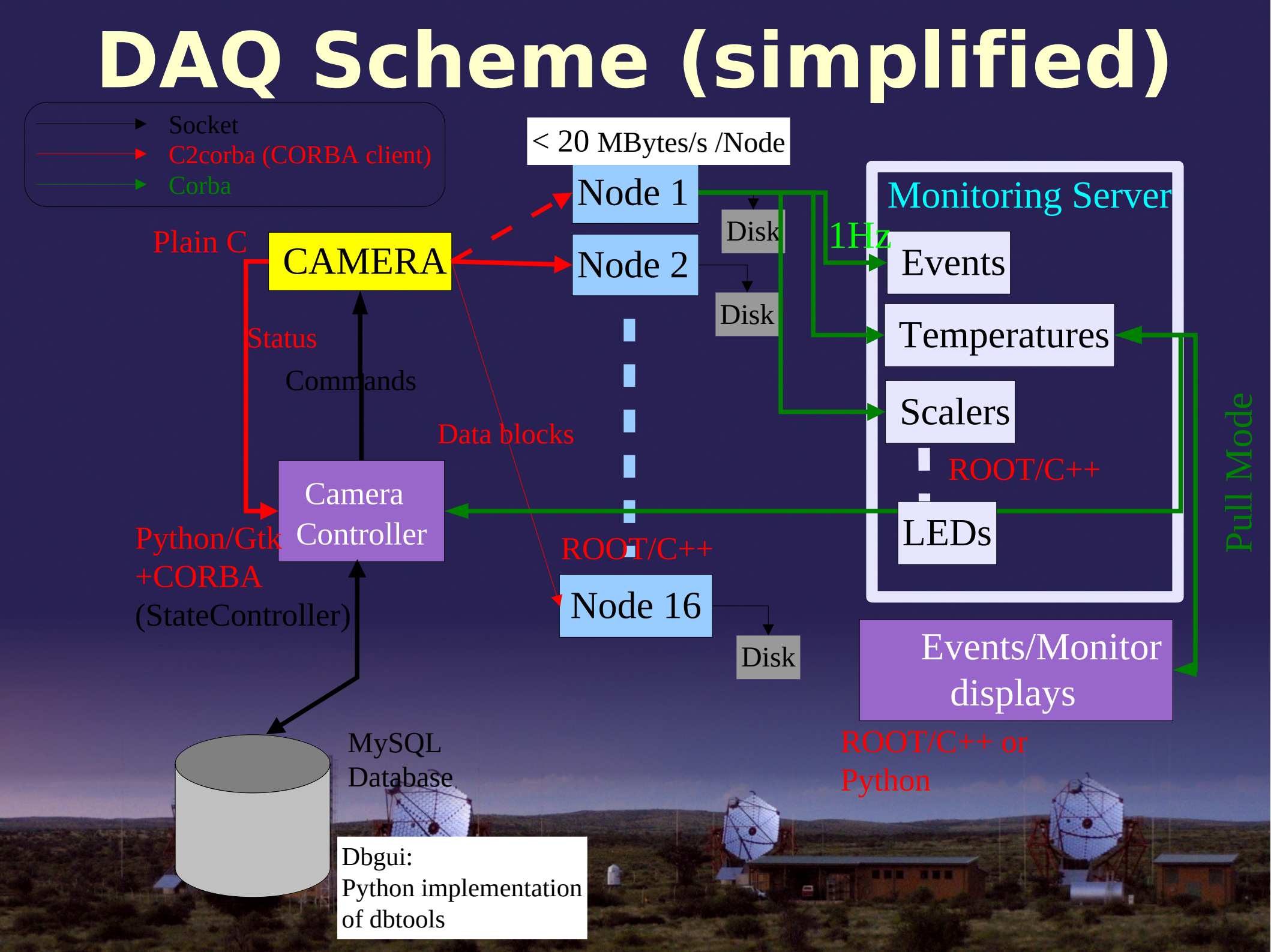
DAQ Hardware



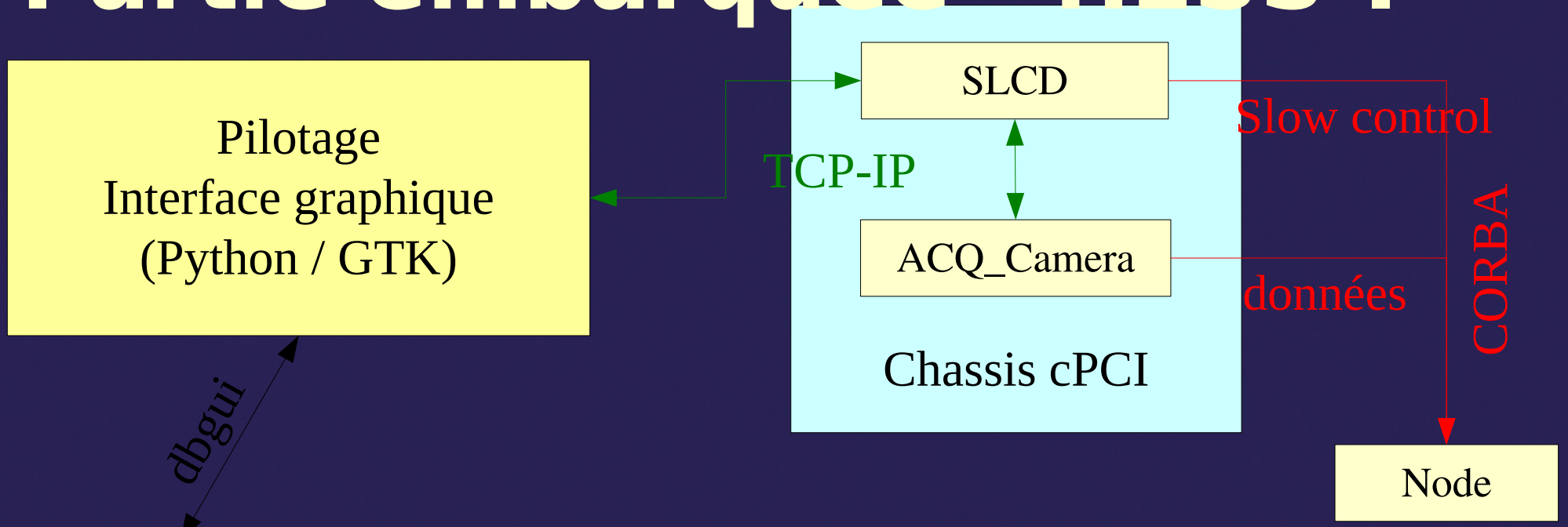
- Linux PC Cluster
  - ~40 commercial i86 machines
  - 2.4 Tbyte disk space
  - 2 LTO-2 tape drives
  - Partly 1 Gbit network (optical network to cameras)
- OS: Fedora Core 12
- UPS!

Camera DAQ

DAQ Scheme (simplified)

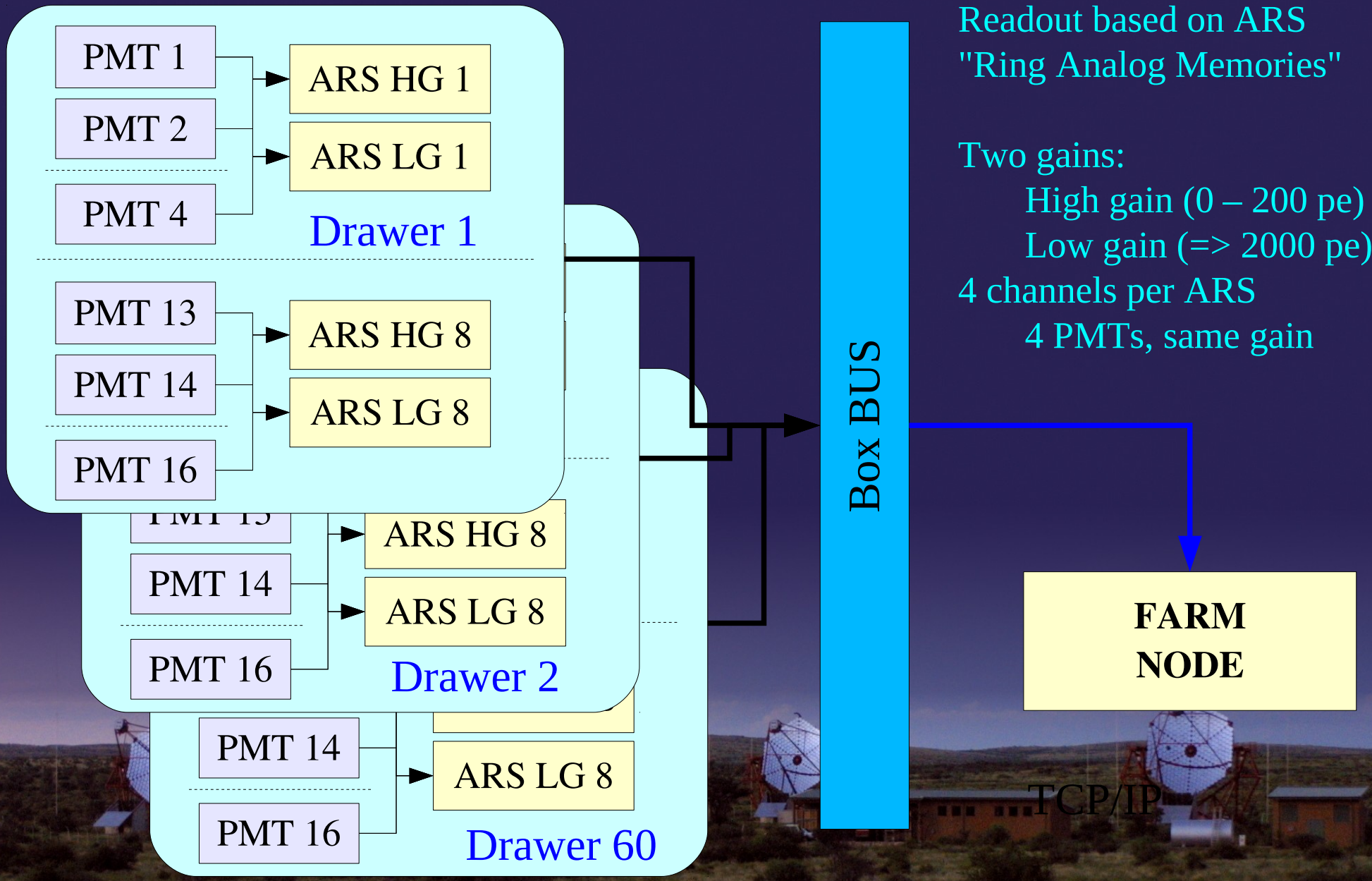


Partie embarquée - HESS-I



- Un démon slow contrôle (slcd)
  - Interpréteur de commande, piloté par une interface graphique au sol
  - Boucle de monitoring (HV, températures,...)
- Un démon d'acquisition
  - « data pooling », envoi vers la ferme

Data Flow – I - Camera



Readout based on ARS "Ring Analog Memories"

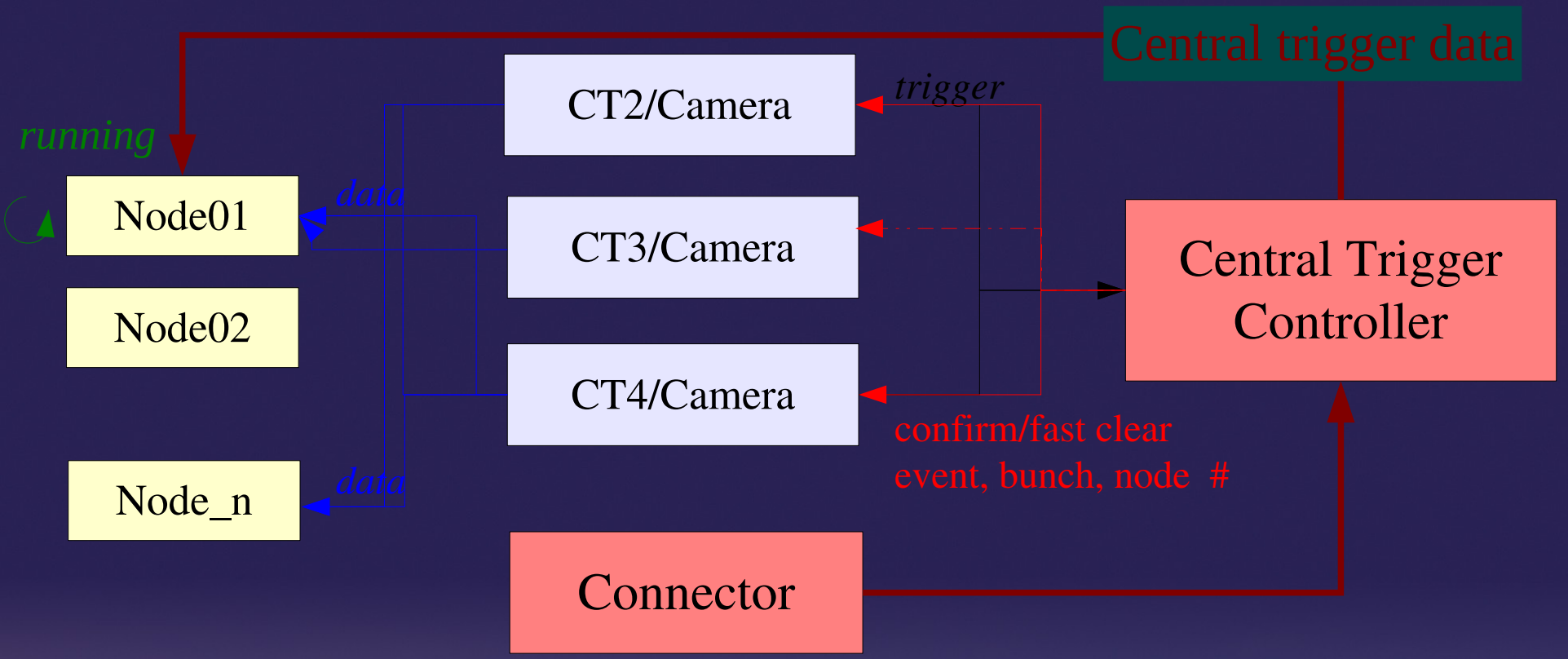
Two gains:

- High gain (0 – 200 pe)
- Low gain (=> 2000 pe)

4 channels per ARS

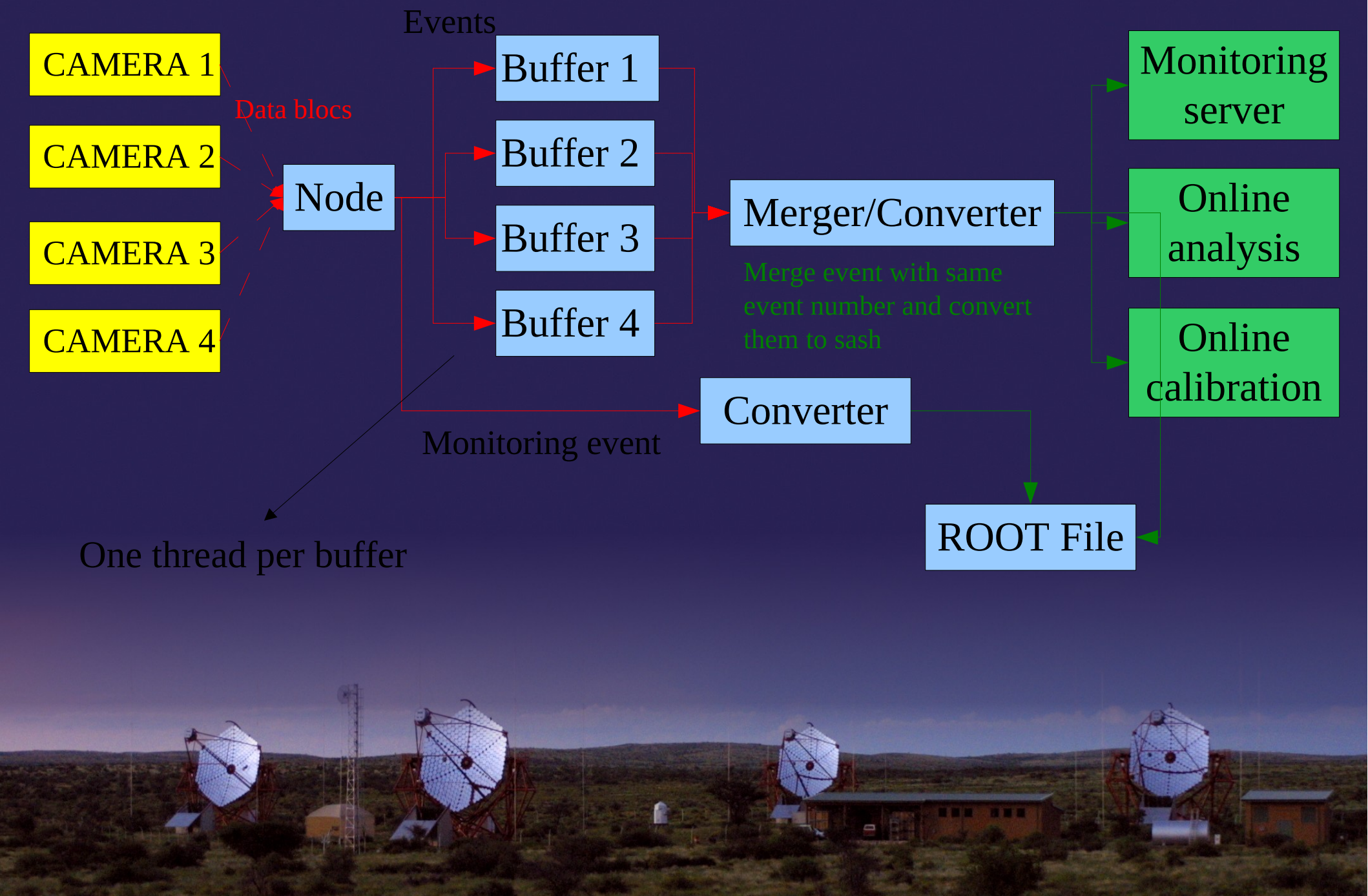
- 4 PMTs, same gain

Data flow – II – Central Acquisition & Online Merging



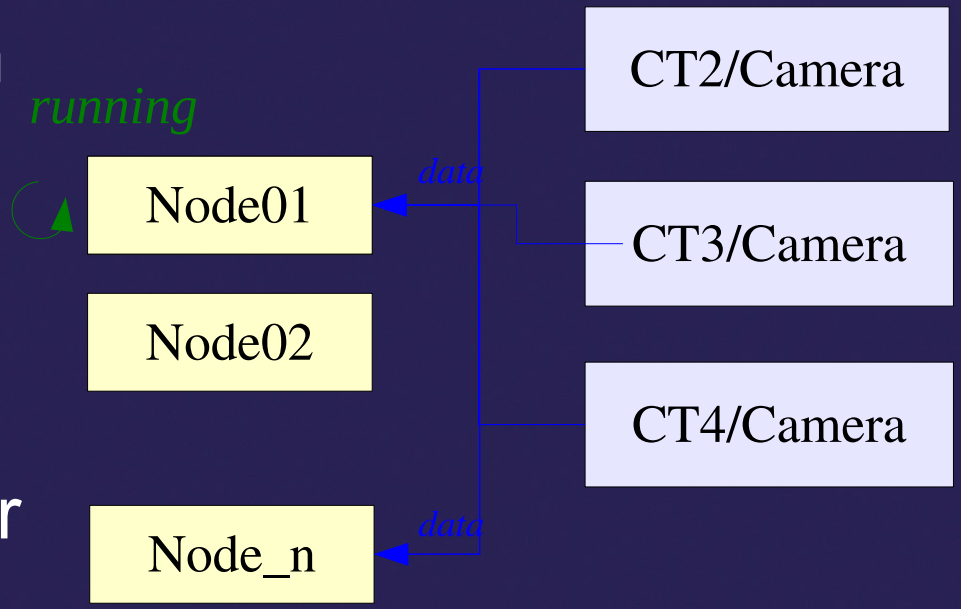
- Each node receives data during 4 seconds
- Automatic node switching
- One data file per node, with all telescopes

CameraReader scheme

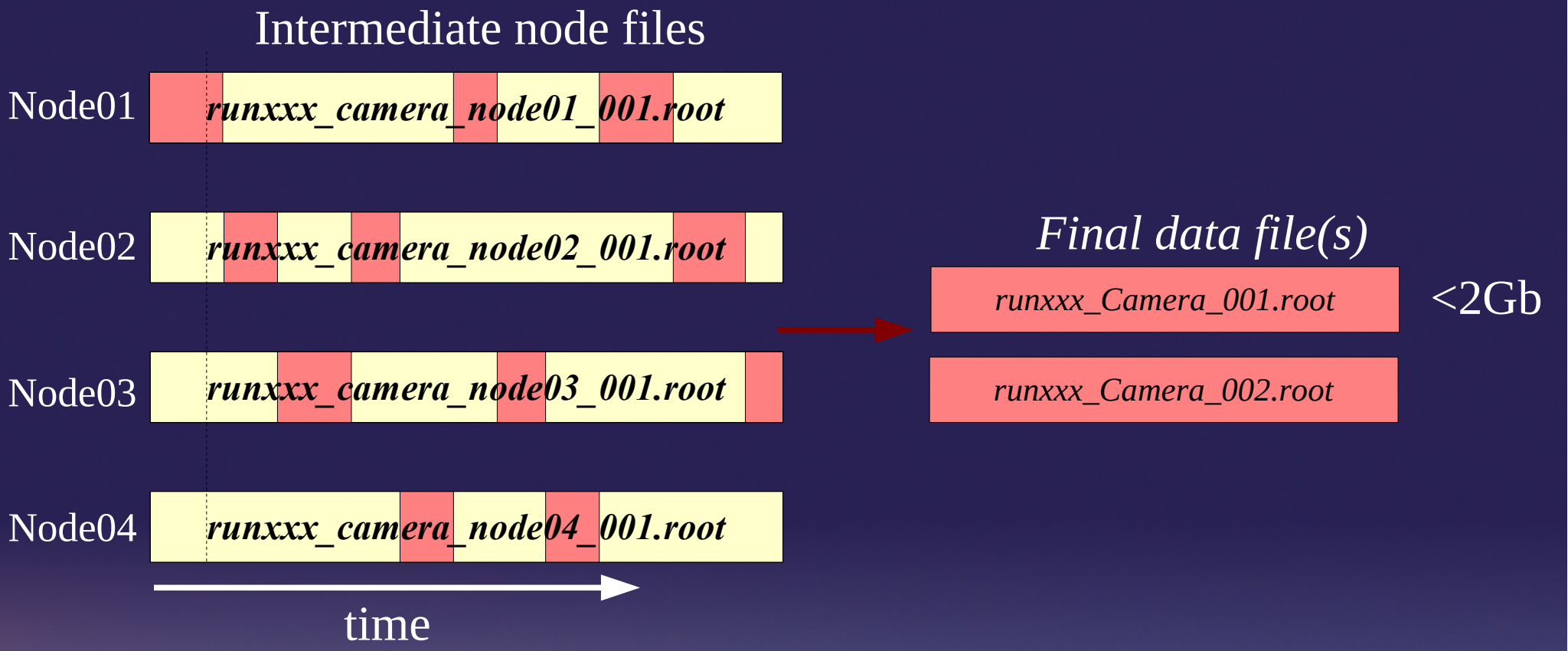


Online Data Merging

- Process running on each farm node (camerareader)
- Merge data from all camera together an with central trigger data
- Relies on bunch/event number (sent by central trigger to all camera @ trigger confirm)
- Buffers the data during 4 second (receiving time)
- Process the data while the other nodes are receiving data
  (< 1 mn) – Speed issue is critical
- Requires perfect consistency between information coming from each source



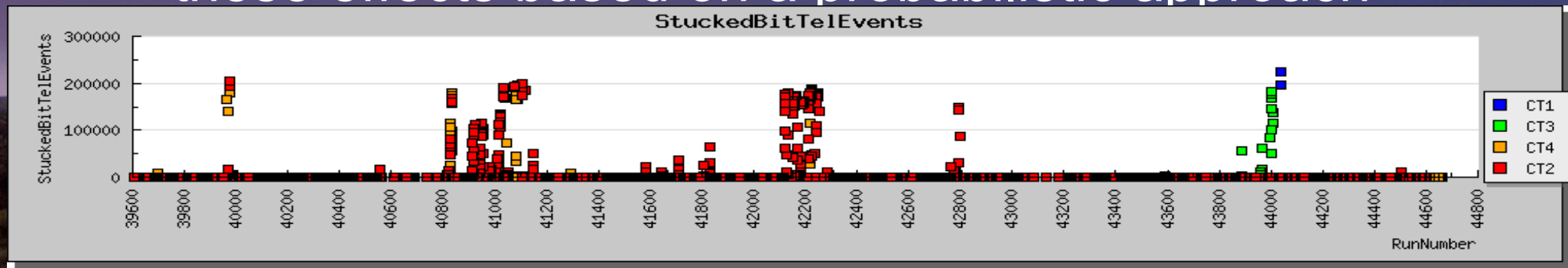
Off line data sorting



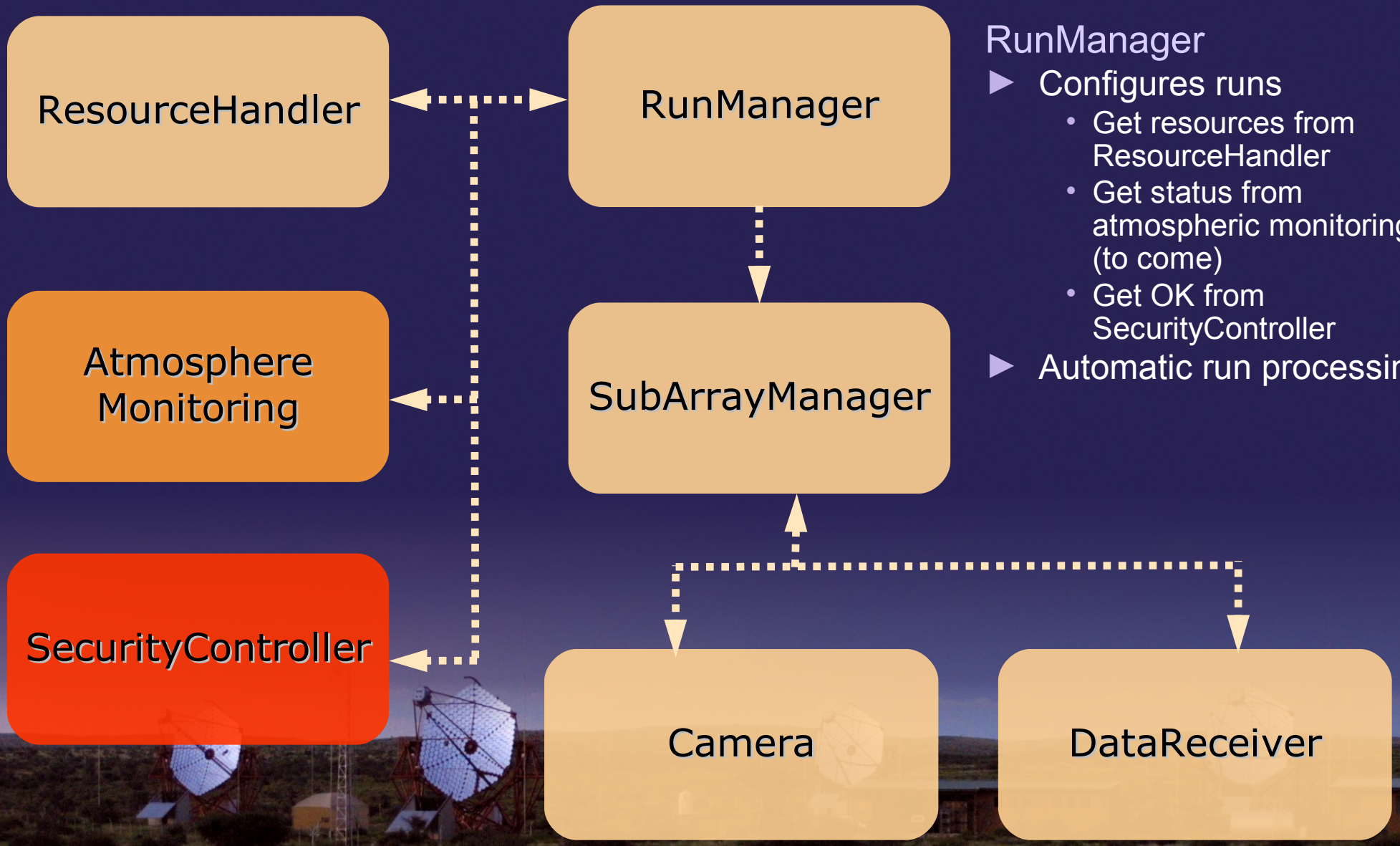
- Runs offline (during day time) in Nambia
- Relies on the central trigger time to produce a time-ordered sequence of events

Problems in event building

- Constant shift in Central GPS:
  - Due to missing antenna or software problemOffset can be large always entire number of seconds (?)
- Freeflying Cameras GPS
- Ghosts events produced by a second trigger generated by noise during the data transfer
  - Not-matching global and local counter
- Incorrect data read from local module
  - Corrupted data, stucked bit
- Need for a complicated software that corrects for these effects based on a probabilistic approach



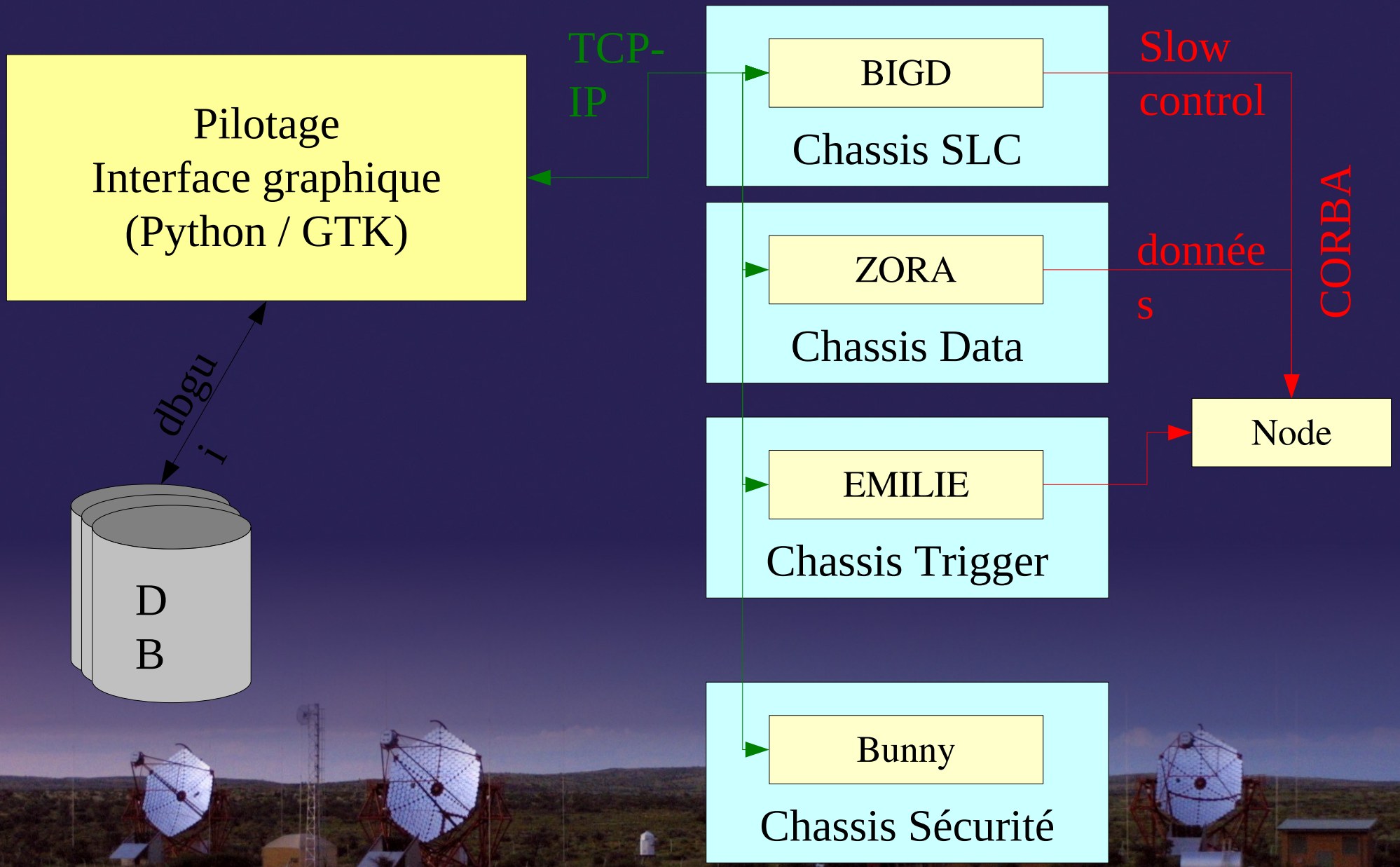
DAQ for H.E.S.S. II



RunManager

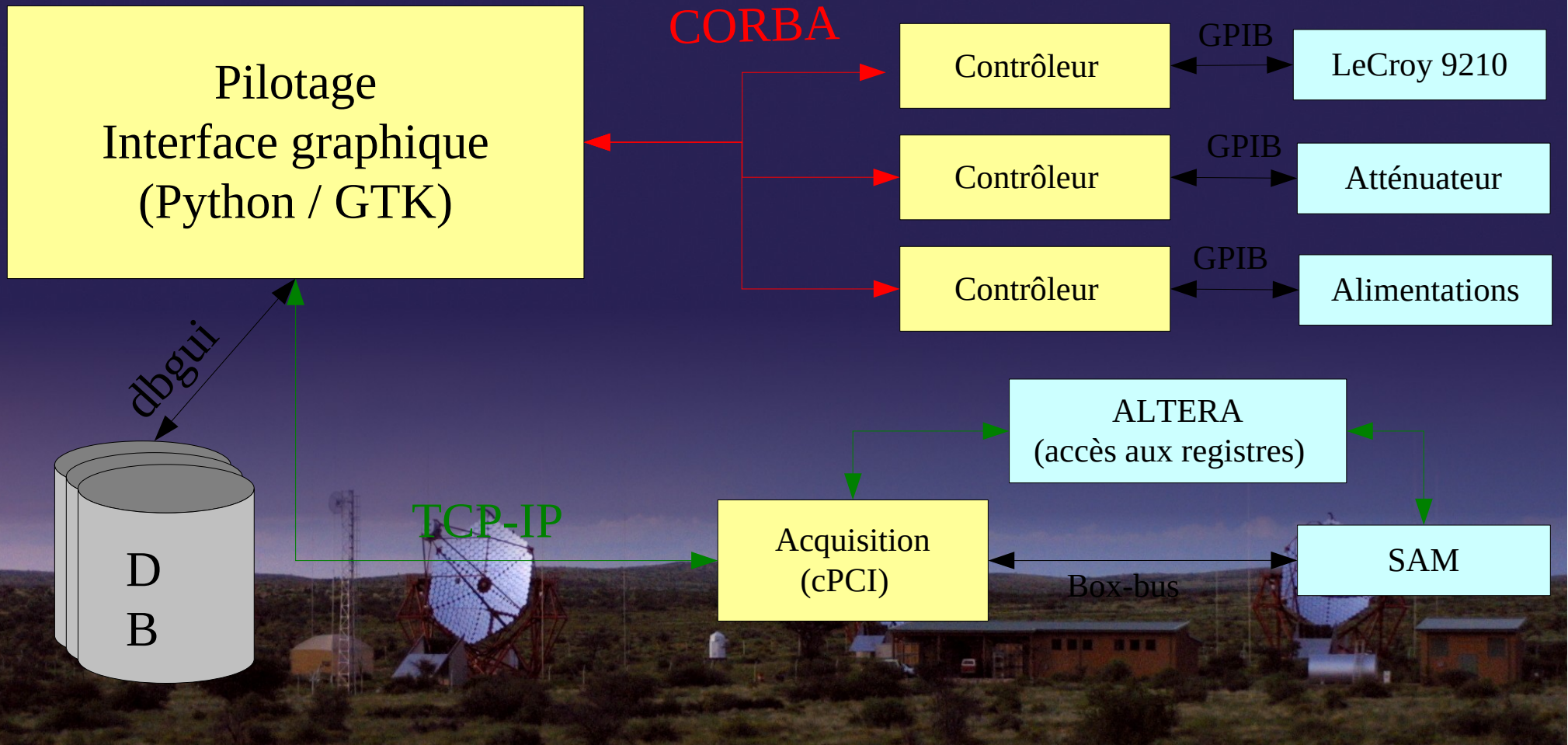
- Configures runs
  - Get resources from ResourceHandler
  - Get status from atmospheric monitoring (to come)
  - Get OK from SecurityController
- Automatic run processing

Partie embarquée - HESS-II



Extenstions: SAMTEXT

- Interface avec SAM
- Pilotage LeCroy 9210 (Interface GPIB), Alimentations et atténuateur
- Résultats systématiquement sauvés dans la base de données
- Sera installé en Namibie pour une maintenance aisée des tiroirs



Conclusion

- Le système d'acquisition de HESS est évolutif et flexible
- Peut accomoder de nombreux systèmes
- Quelques difficultés au démarrage (apprentissage, mais aussi rendre ROOT multitaches)
- L'intégration de HESS-II s'est faite rapidement.
- Volume de données important: 30GB/s sans problèmes avec HESS-II
- Mais certaines améliorations possible, notamment en ce qui concerne la tolérance aux pannes et la réaction à des erreurs
- Les instruments n'ont pas de driver unifiés => Multiplicité de sous systèmes indépendants (tracking, météo, caméra, ...)