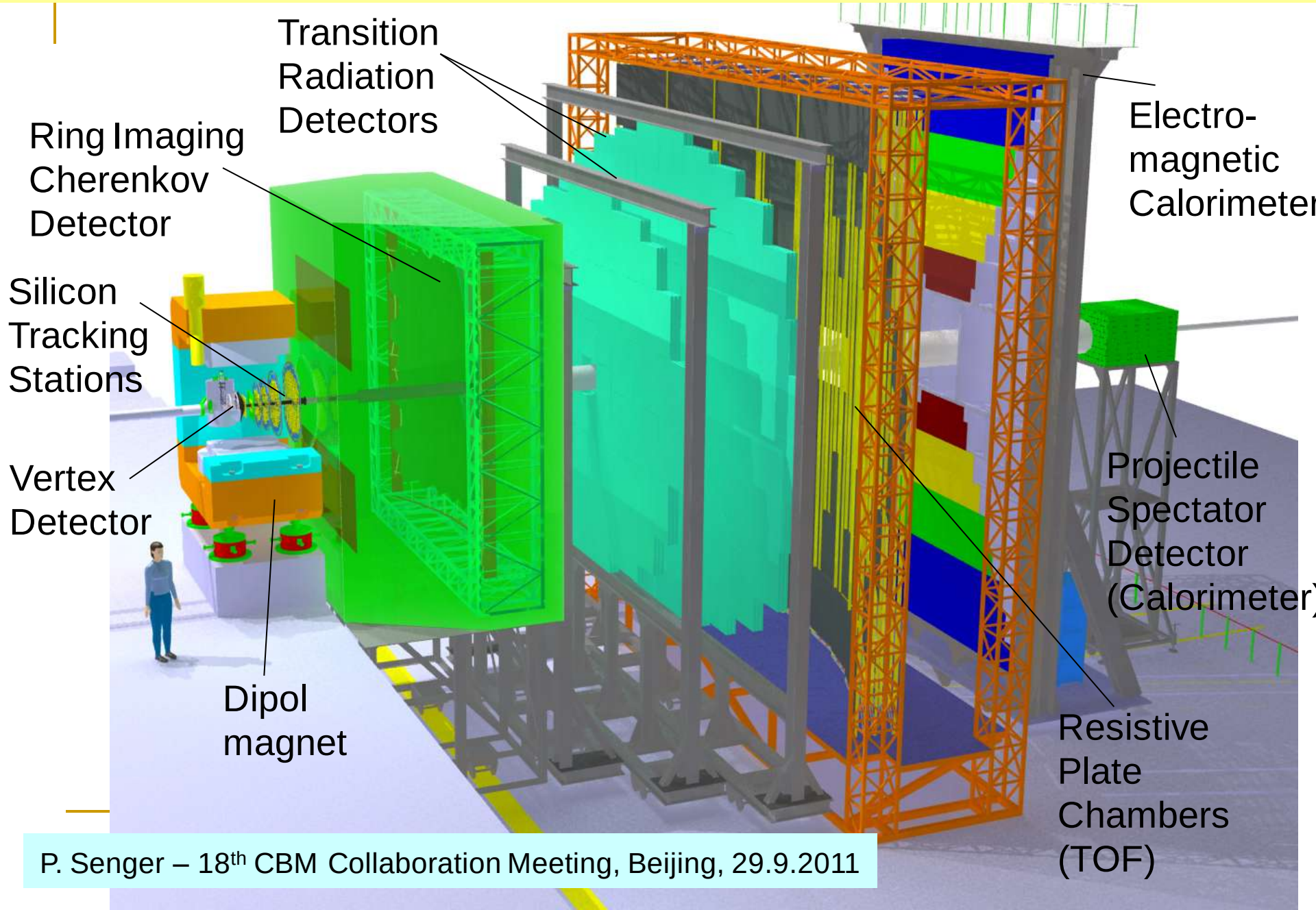
DAQ and online software development for CBM experiment

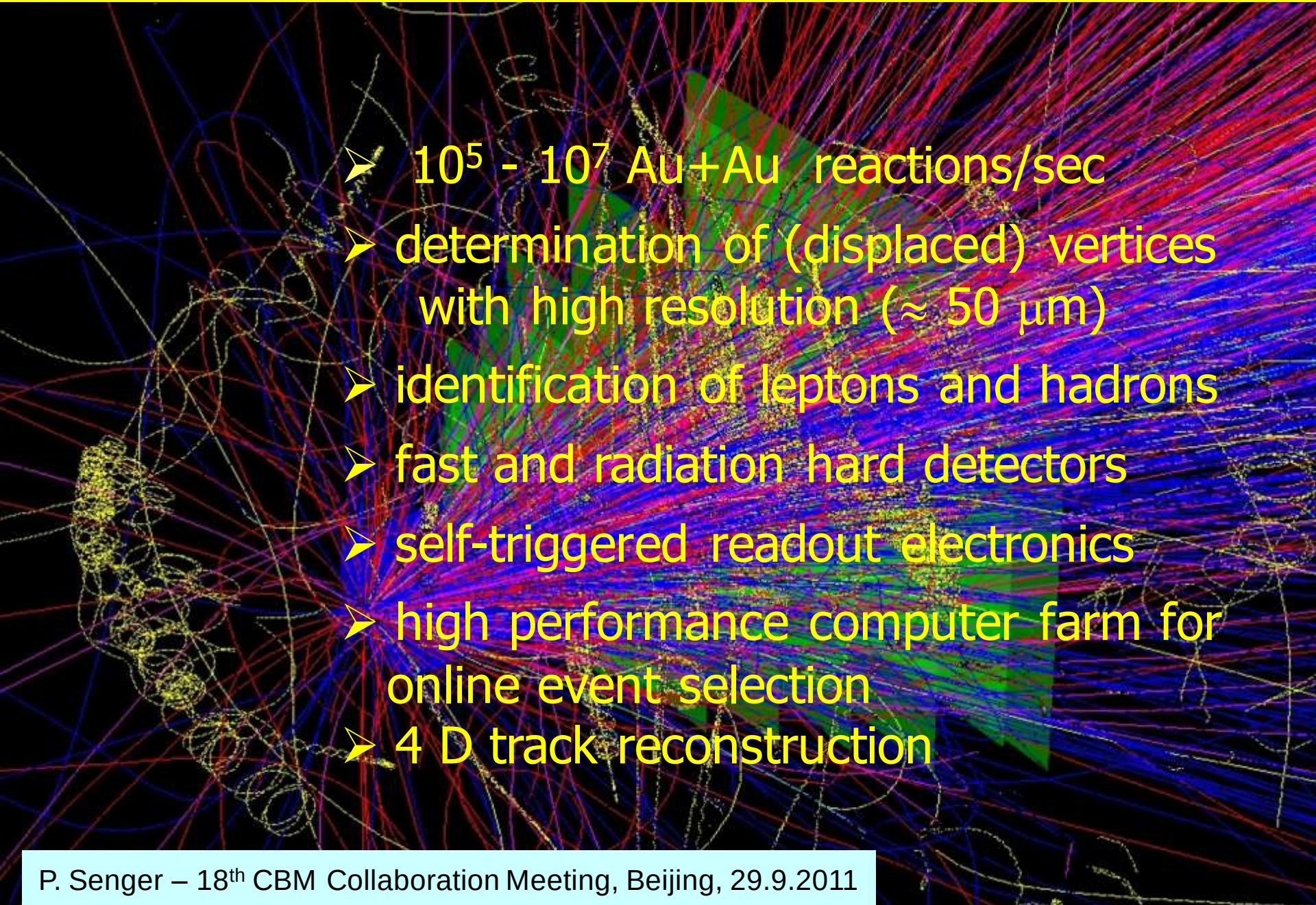
Sergey Linev,

Experiment Electronic, GSI

The Compressed Baryonic Matter Experiment

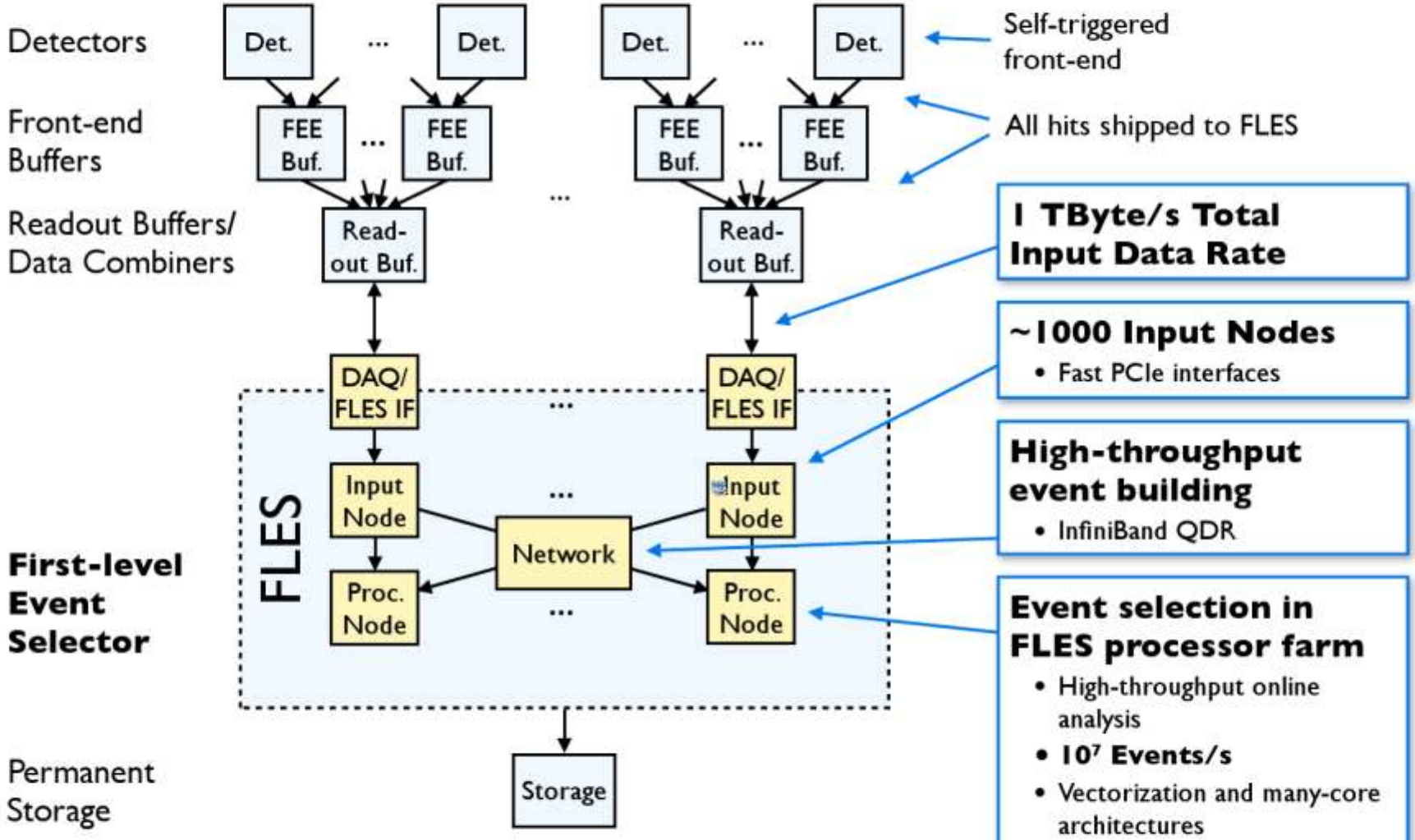


Experimental challenges



- 10^5 - 10^7 Au+Au reactions/sec
- determination of (displaced) vertices with high resolution ($\approx$ 50 μm)
- identification of leptons and hadrons
- fast and radiation hard detectors
- self-triggered readout electronics
- high performance computer farm for online event selection
- 4 D track reconstruction

CBM Online Computing and Readout



Main challenges for CBM DAQ

- Triggerless readout
  - precise timing system
  - custom-designed readout chips
- High data rates
  - high-speed interconnect to the front-ends
  - FPGA-based signals processing
- Event building with 1 TB/s
  - high-performance (but low cost) network

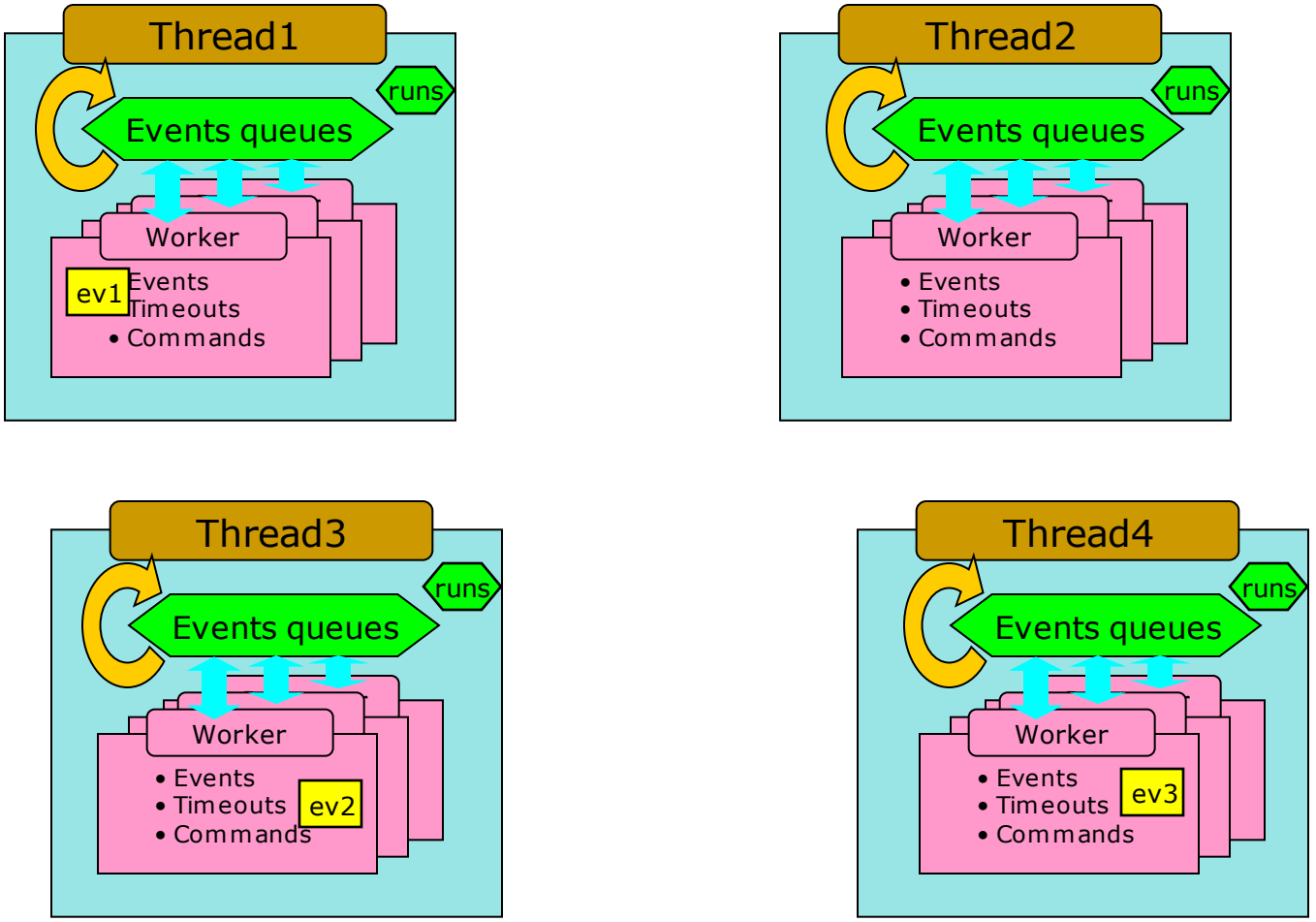
What kind of DAQ software is required?

- Connect (near) any frontend
- Handle (together) triggered and self-triggered data
- Merge / split / distribute data-streams over many compute nodes
- Provide interfaces for application code
- Probe key techniques of future CBM DAQ system

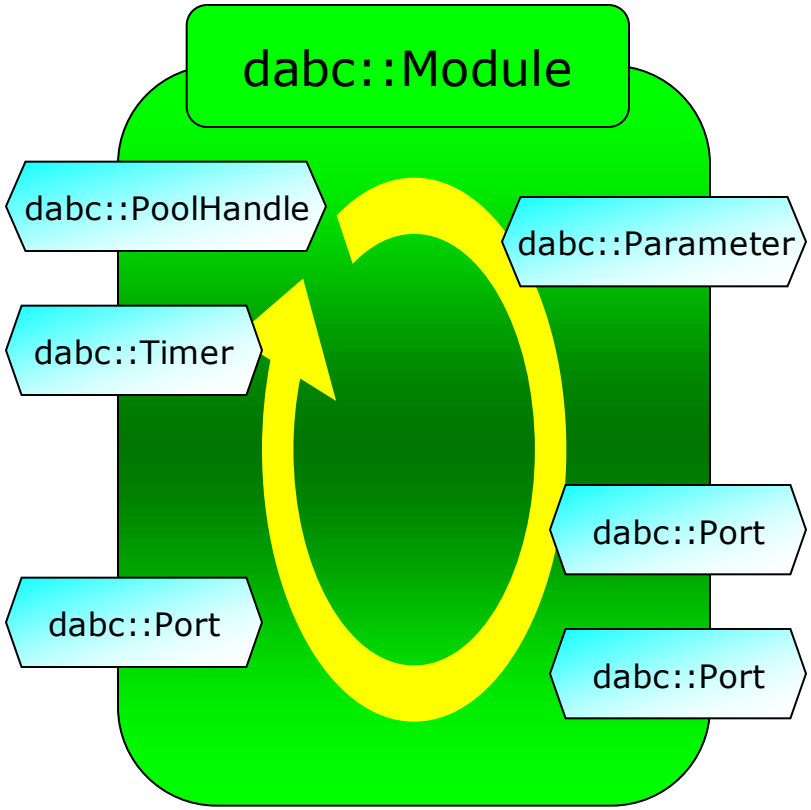
DABC: Main features

- Compact multi-threaded data-flow core
- Number of device/application specific add-ons
- TCP/IP (sockets) and InfiniBand (OFED verbs) as data transports
- Plugins for user-specific components
- BNET – components for constructing event-building network
- Flexible configurations with xml files
- DIM and EPICS as interface for control system
- generic Java GUI

Multithreading in DABC



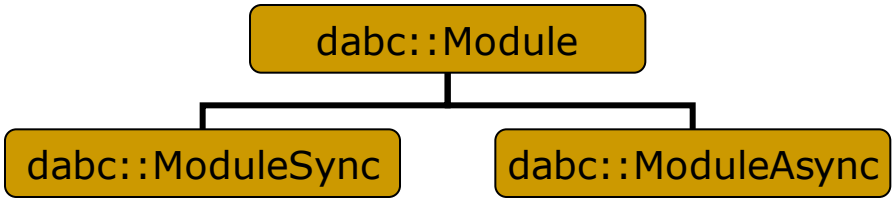
dabc::Module – place for user code

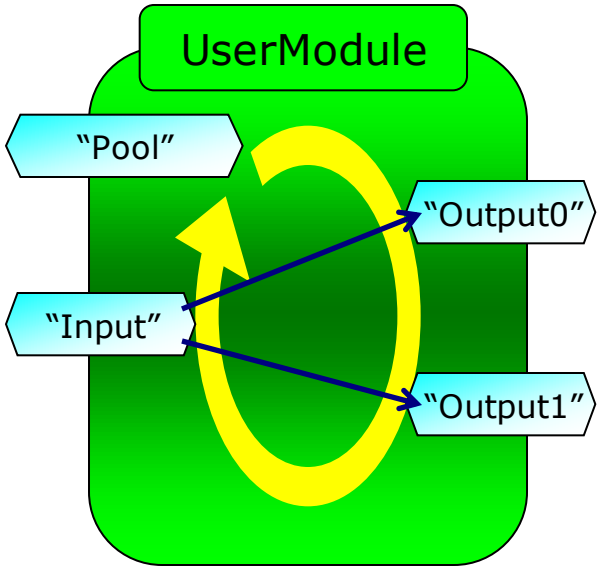


dabc::Module class provides:

- I/O ports for communications
- Pools handles to request memory
- Timers for timeouts processing
- Configuration & monitoring parameters

Synchronous and asynchronous modules





Constructors mainly the same

```
CreatePoolHandle("Pool", 2048, 1);
CreateInput("Input", Pool(), 5);
CreateOutput("Output0", Pool(), 5);
CreateOutput("Output1", Pool(), 5);
fCnt = 0;
```

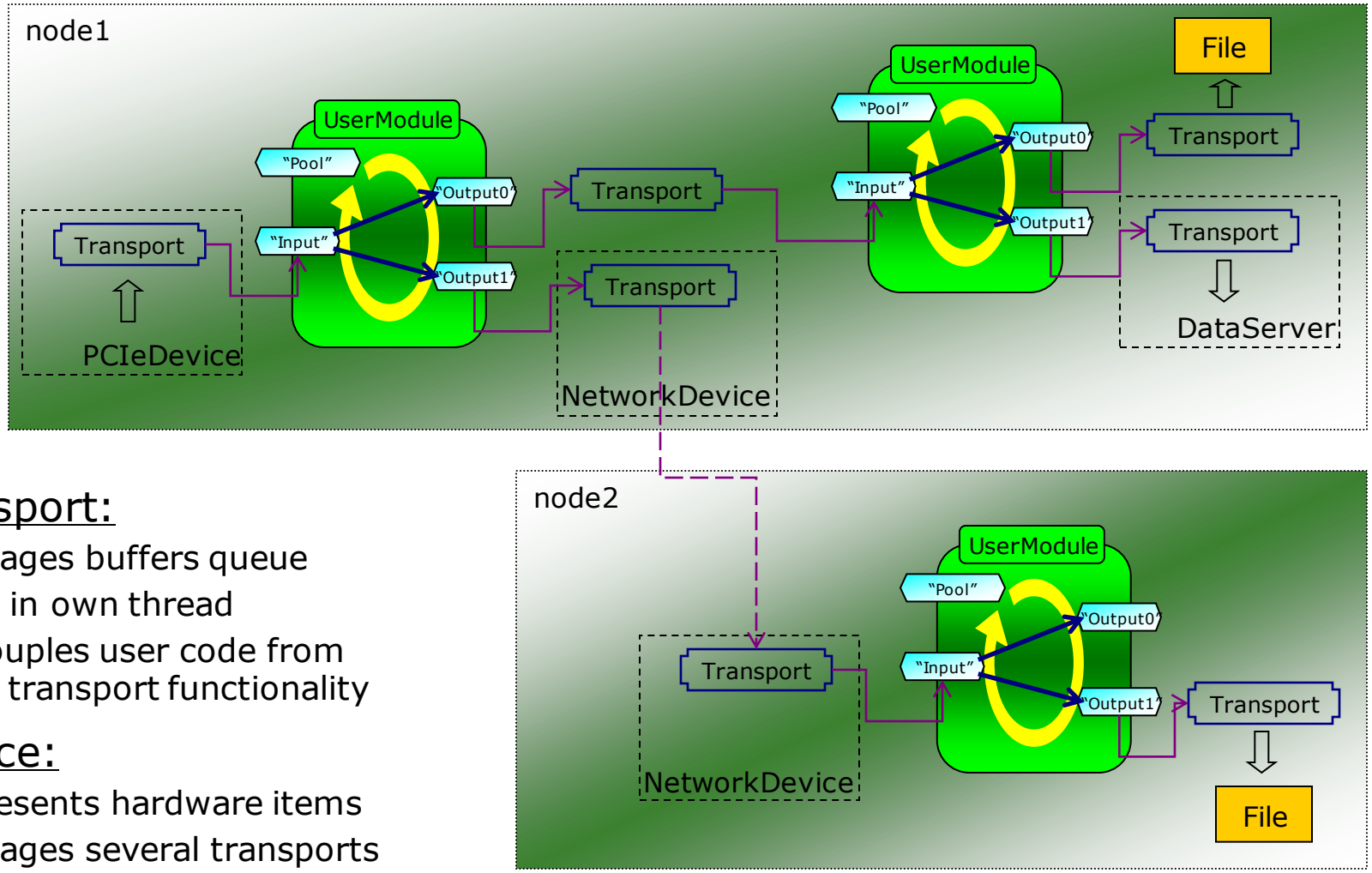
Explicit loop in ModuleSync

```
void UserModule::MainLoop()
{
  while (ModuleWorking()) {
    dabc::Buffer buf = Recv(Input());
    if (fCnt++ % 2 == 0) Send(Output(0), buf);
                    else Send(Output(1), buf);
  }
}
```

Event processing in ModuleAsync

```
void UserModule::ProcessIOEvent(dabc::Port*)
{
  while (Input()->CanRecv() &&
         Output(fCnt % 2)->CanSend()) {
    dabc::Buffer buf = Input()->Recv();
    Output(fCnt++ % 2)->Send(buf);
  }
}
```

Devices and transports - dataflow



Transport:

- manages buffers queue
- runs in own thread
- decouples user code from actual transport functionality

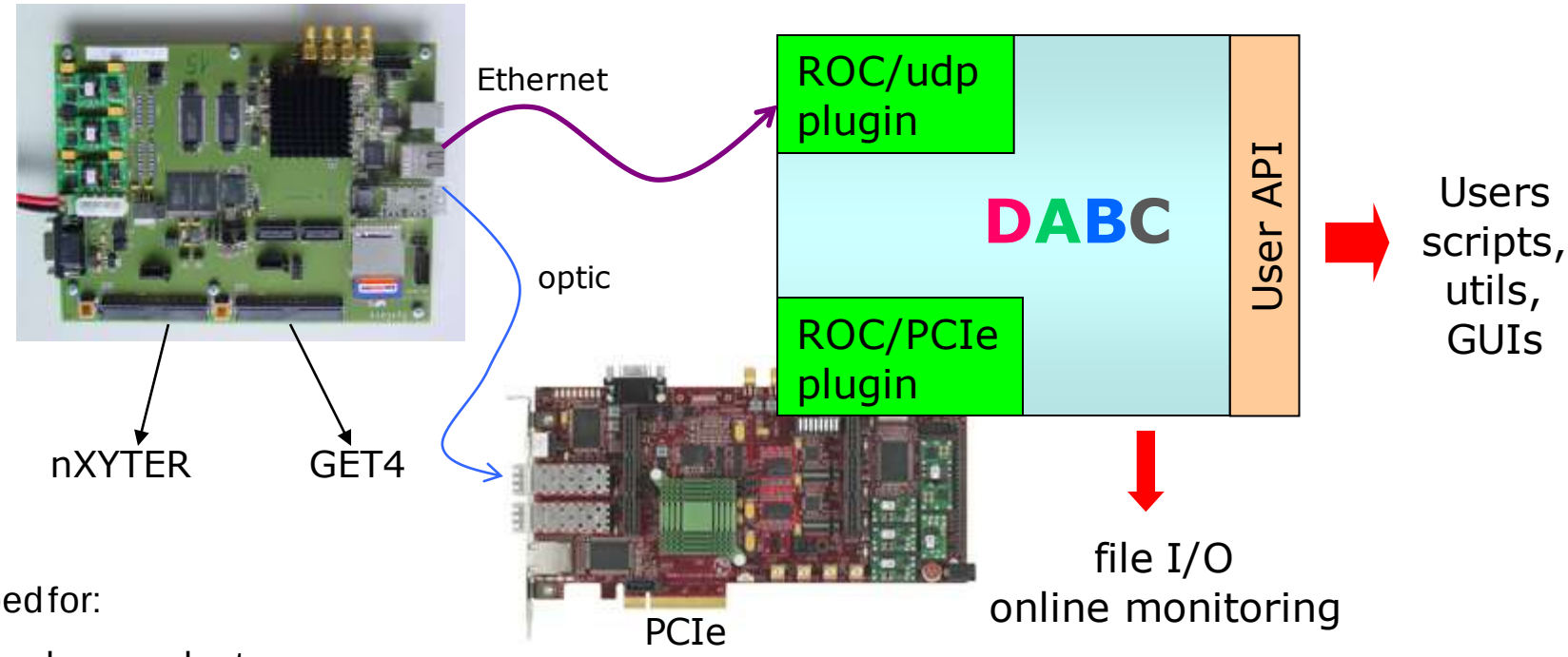
Device:

- represents hardware items
- manages several transports

DABC – status and plans

- C++ software framework, developed since 2008
- Works on 32/64 bit Linux computers
- Current stable release is 1.1, available on http://dabc.gsi.de
- Since mid-2011 version 1.9 (DABC2 beta) is available, accessible via repository: https://subversion.gsi.de/goofy/dabc/brunches/ver2

DABC as access layer to CBM ROC



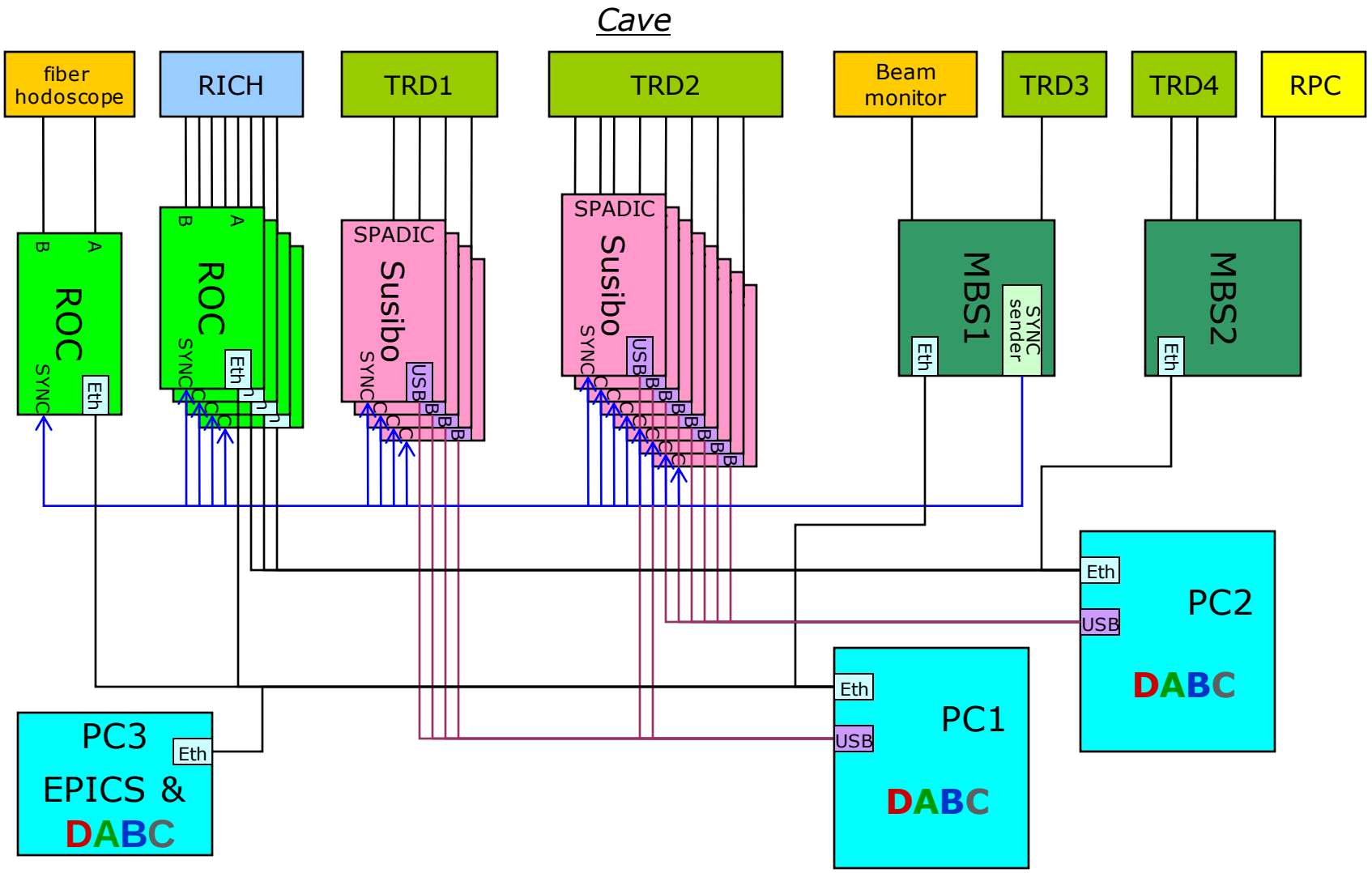
Testbed for:

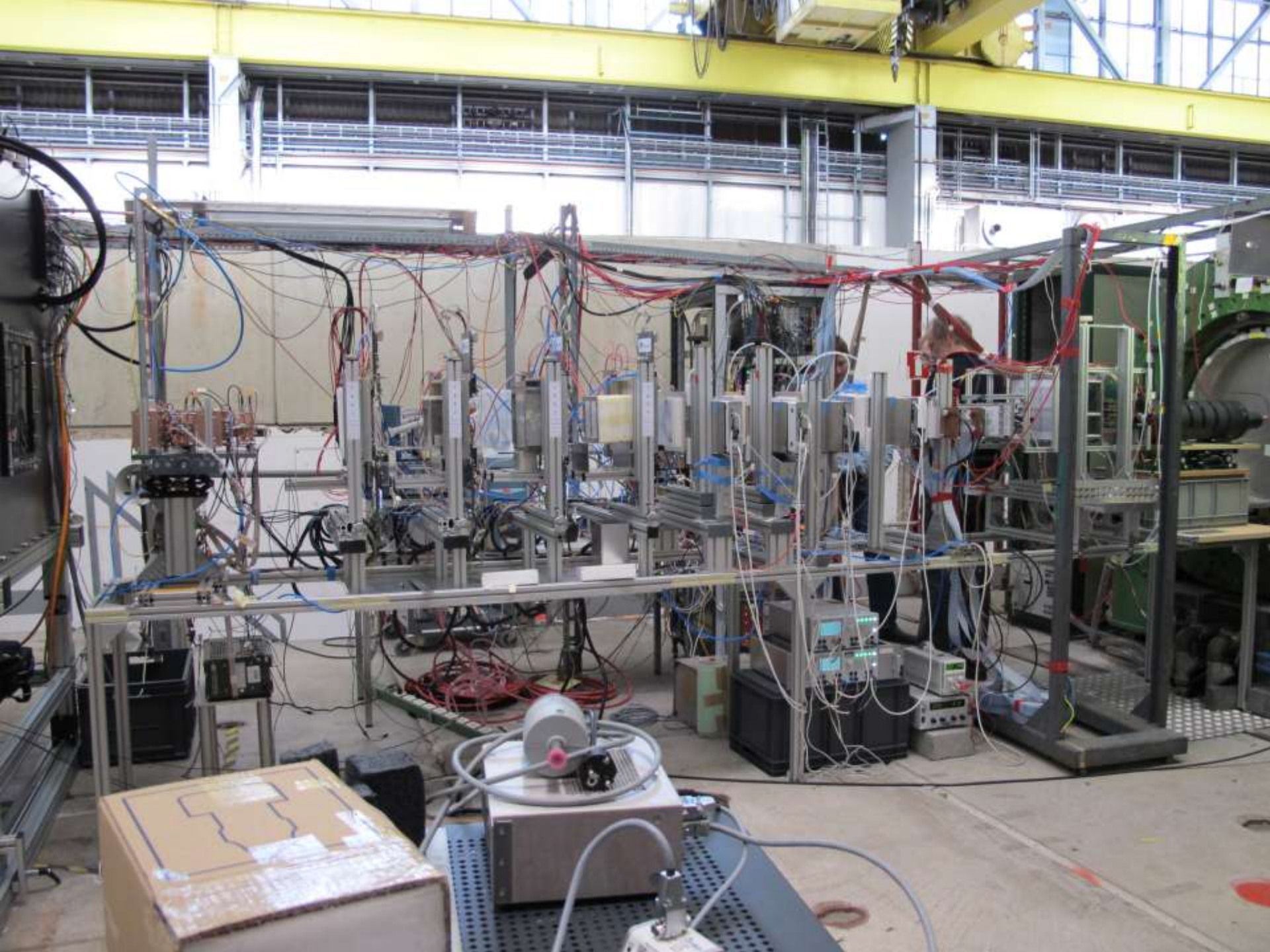
- triggerless readout
- message-oriented data format
- clock/time synchronization

DABC plugins for for ROC readout

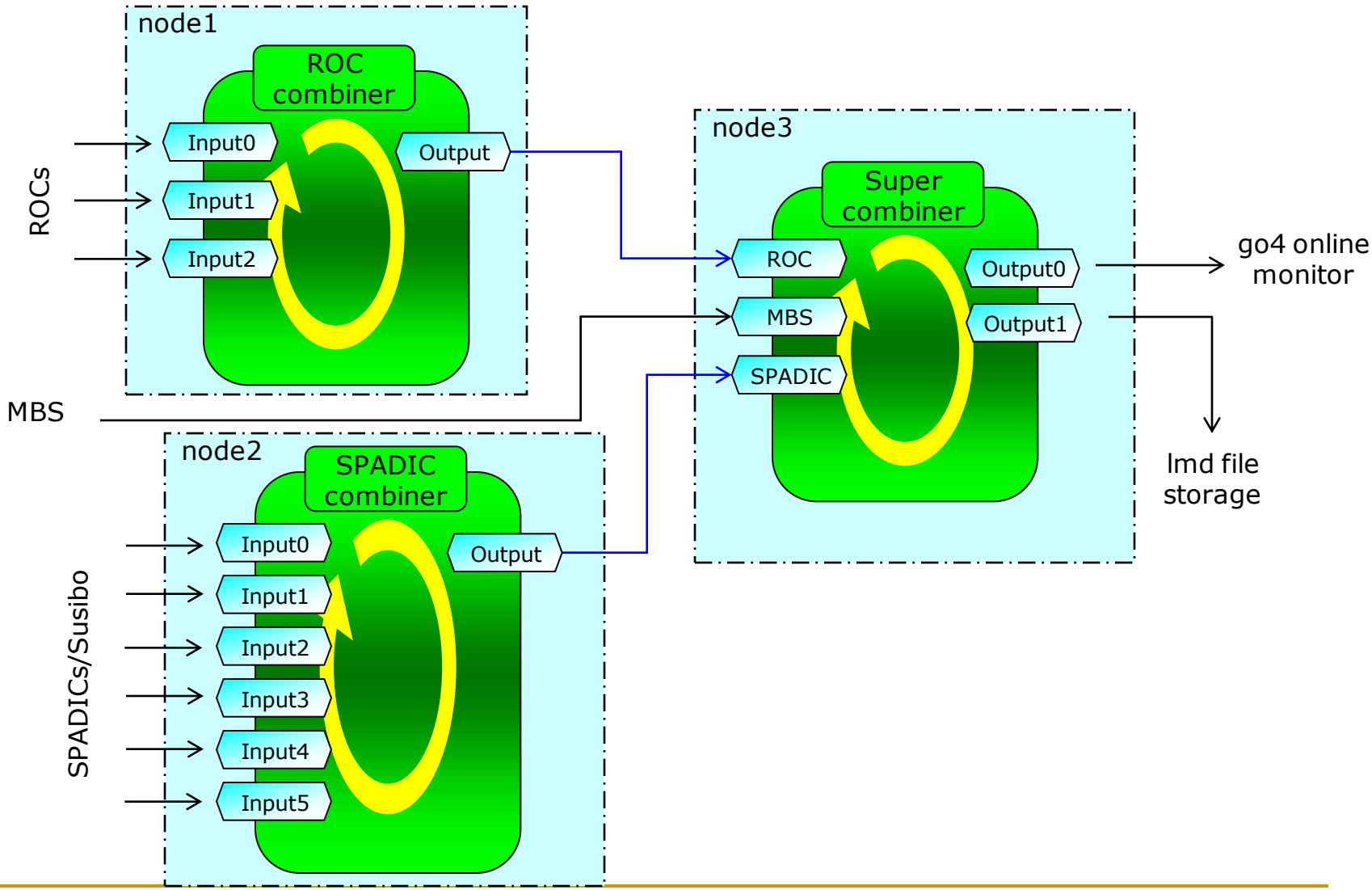
- UDP-based
  - uses *dabc::SocketWorker* class
  - implementation based on *select()* method
    - many connections can be treated in single thread
- Optic-based
  - *mprace* library from Uni Mannheim
  - based on simple *dabc::DataTransport* class

Planned setup for CERN (Oct 11)

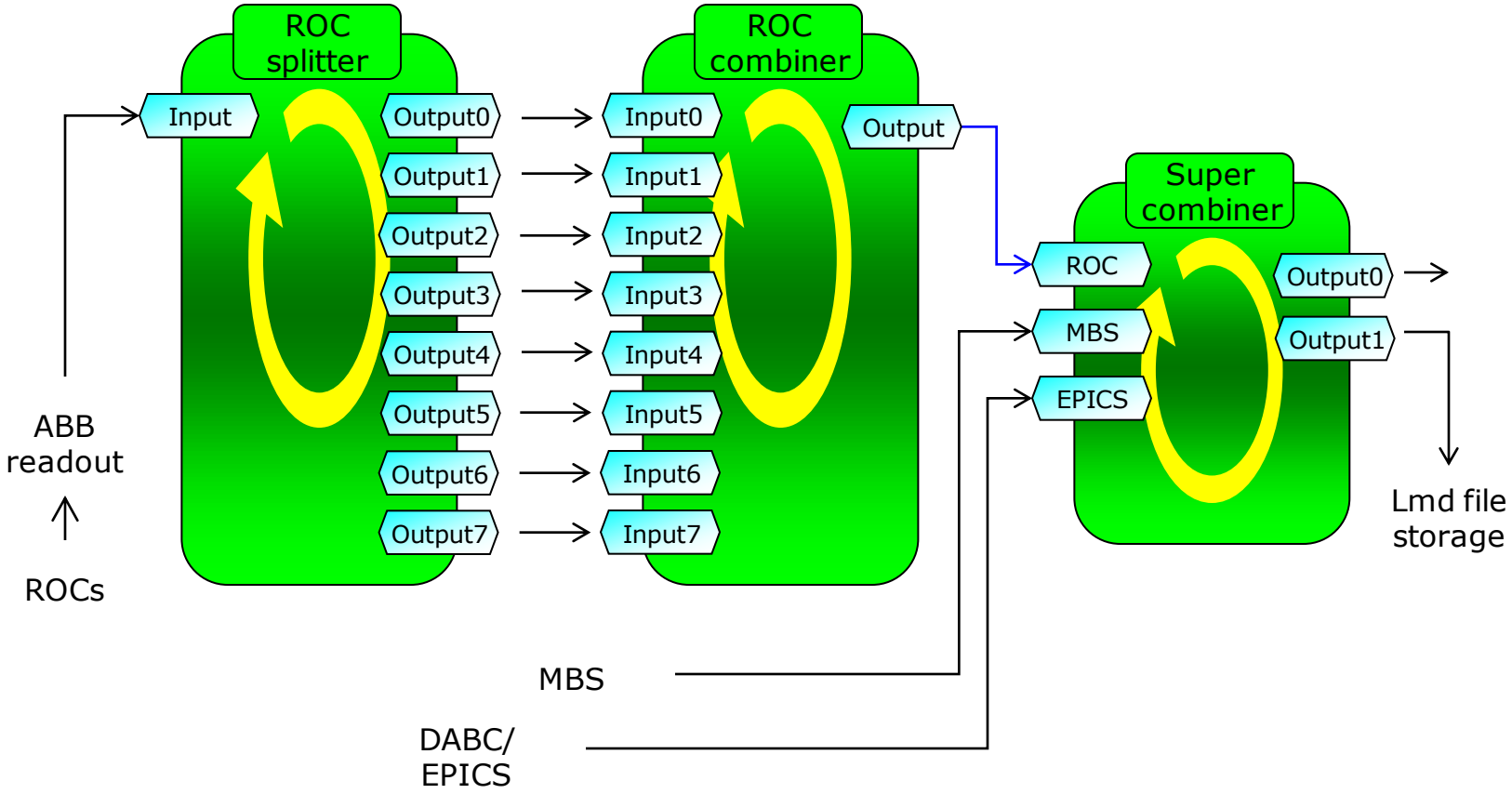




Readout application

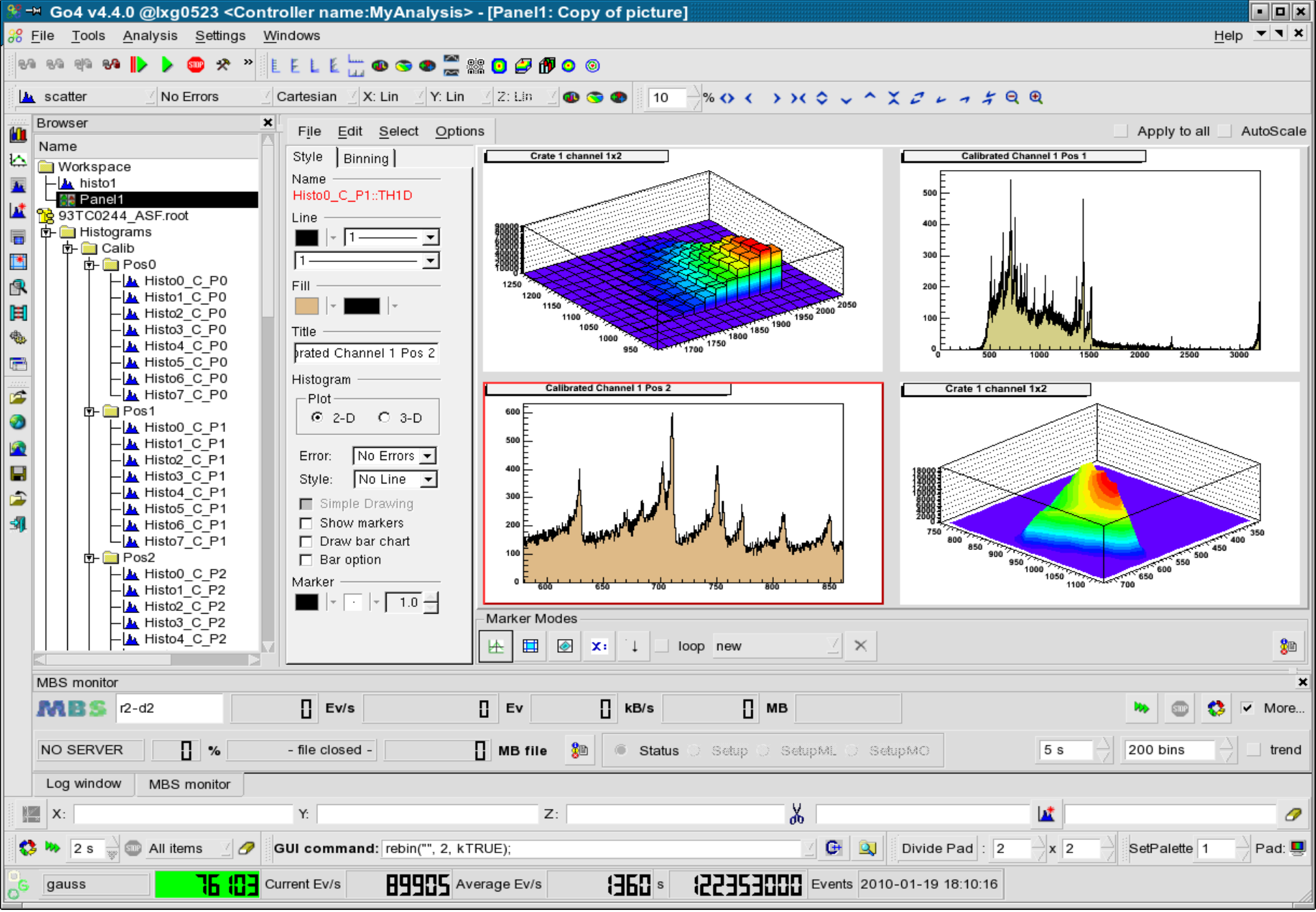


Data splitter for optic data

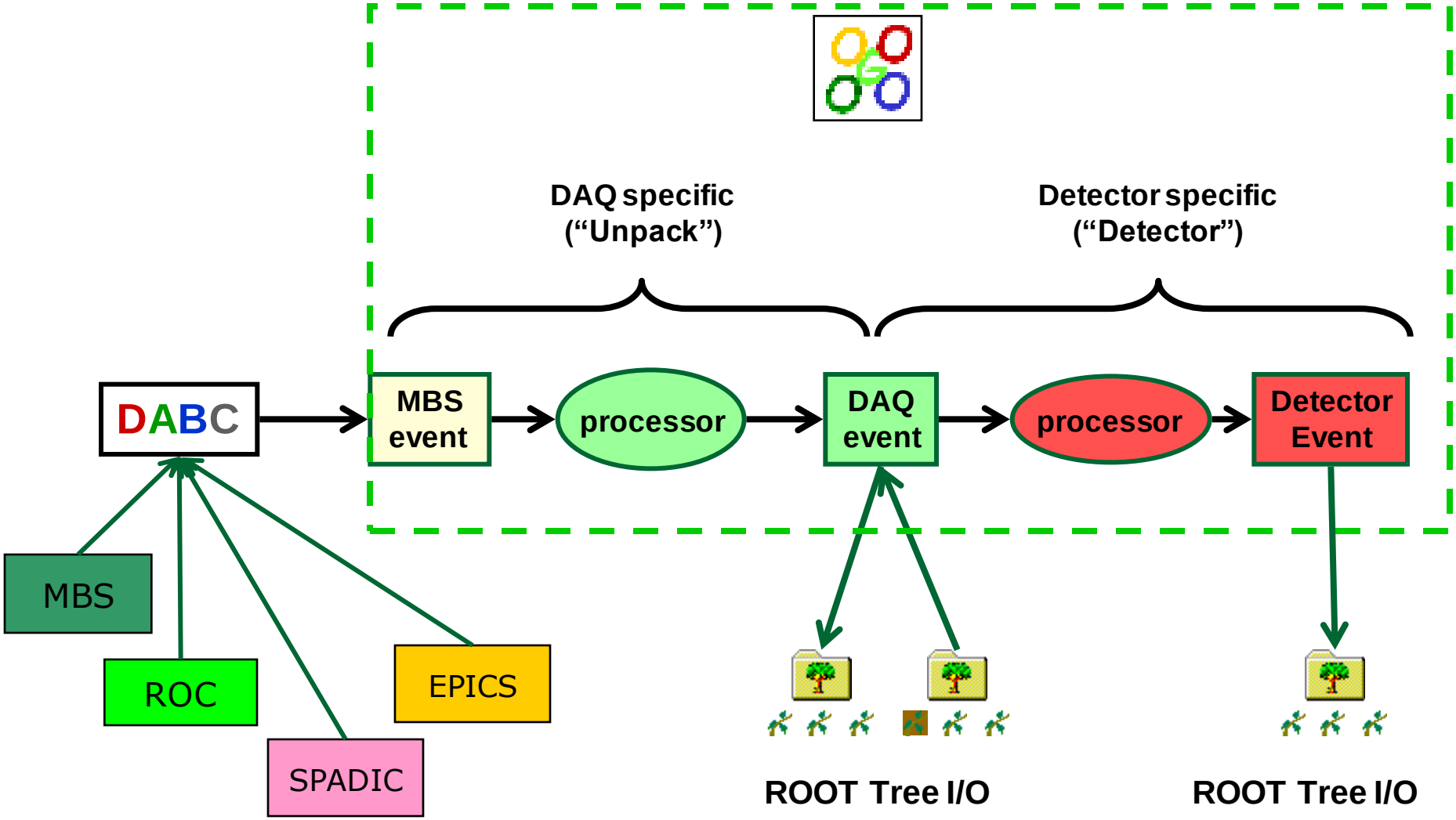


Go4 – GSI analysis framework

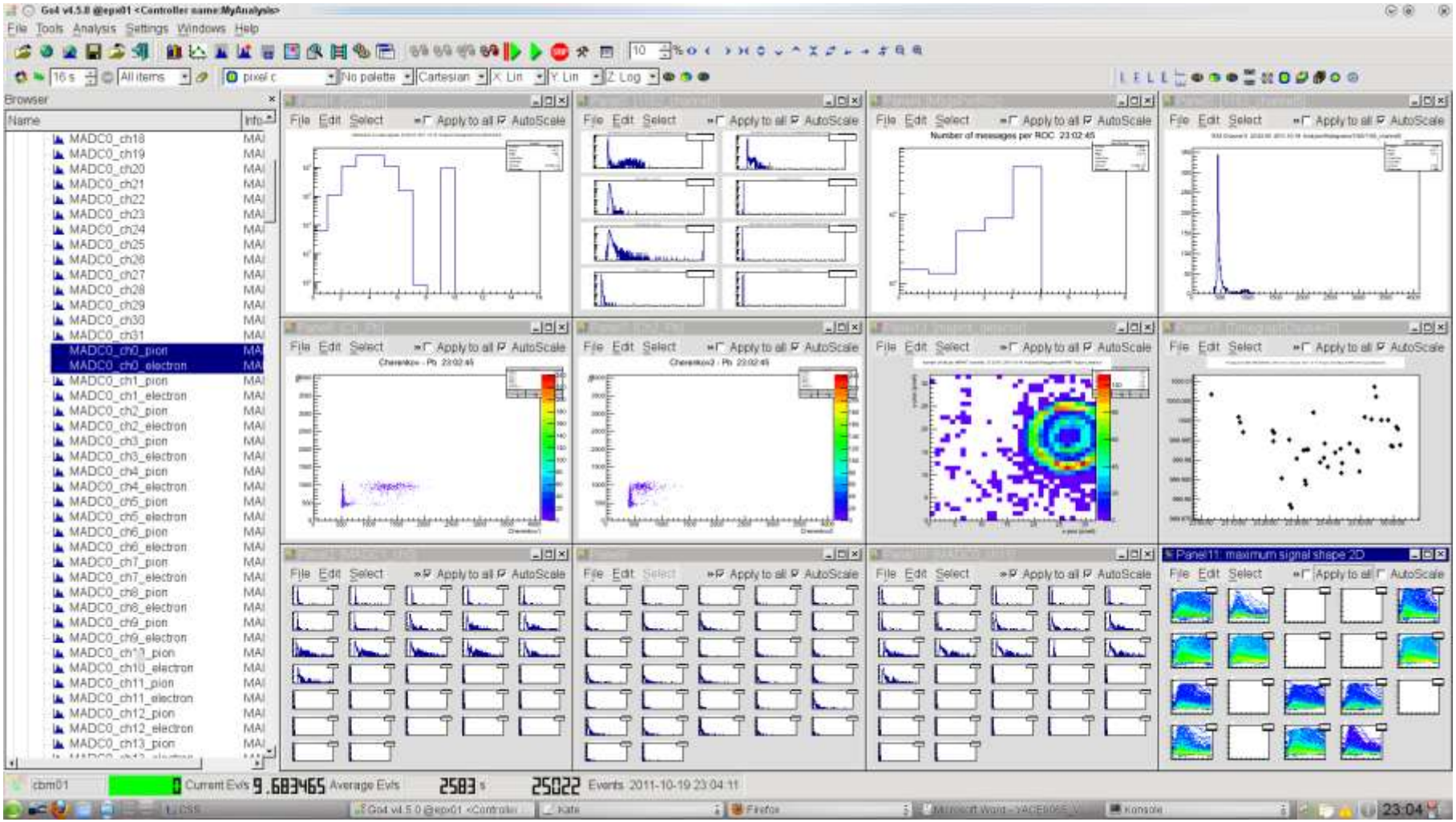
- Framework for many kinds of experiments (Atomic & Nuclear Physics)
- Based on C++, ROOT (CERN) and Qt (Nokia)
- Provides services and interfaces for user written analysis
- Batch mode (CINT or compiled, online/offline)
- Interactive mode (online/offline):
  - A non blocking GUI controls and steers the analysis
  - GUI interfaces ROOT and Qt graphics
  - Analysis can update graphics asynchronously: live monitoring
  - User can create and add specific GUIs (Qt designer)


Go4 v4.4.0 @lxg0523 <Controller name:MyAnalysis> - [Panel1: Copy of picture]
File Tools Analysis Settings Windows Help
scatter No Errors Cartesian X: Lin Y: Lin Z: Lin
Browser
Name
Workspace
histo1
Panel1
93TC0244_ASF.root
Histograms
Calib
Pos0
Histo0_C_P0
Histo1_C_P0
Histo2_C_P0
Histo3_C_P0
Histo4_C_P0
Histo5_C_P0
Histo6_C_P0
Histo7_C_P0
Pos1
Histo0_C_P1
Histo1_C_P1
Histo2_C_P1
Histo3_C_P1
Histo4_C_P1
Histo5_C_P1
Histo6_C_P1
Histo7_C_P1
Pos2
Histo0_C_P2
Histo1_C_P2
Histo2_C_P2
Histo3_C_P2
Histo4_C_P2
File Edit Select Options
Apply to all AutoScale
Style Binning
Name
Histo0_C_P1::TH1D
Line
Fill
Title
Histogram
Plot
2-D 3-D
Error: No Errors
Style: No Line
Simple Drawing
Show markers
Draw bar chart
Bar option
Marker
Crate 1 channel 1x2
Calibrated Channel 1 Pos 1
Calibrated Channel 1 Pos 2
Crate 1 channel 1x2
Marker Modes
loop new
MBS monitor
MBS r2-d2 Ev/s Ev kB/s MB More...
NO SERVER % - file closed - MB file Status Setup SetupML SetupMO 5 s 200 bins trend
Log window MBS monitor
X: Y: Z:
2 s All items GUI command: rebin("", 2, kTRUE); Divide Pad : 2 x 2 SetPalette 1 Pad:
gauss 76 103 Current Ev/s 89905 Average Ev/s 1360 s 122353000 Events 2010-01-19 18:10:16

Analysis organization for CBM beamtimes



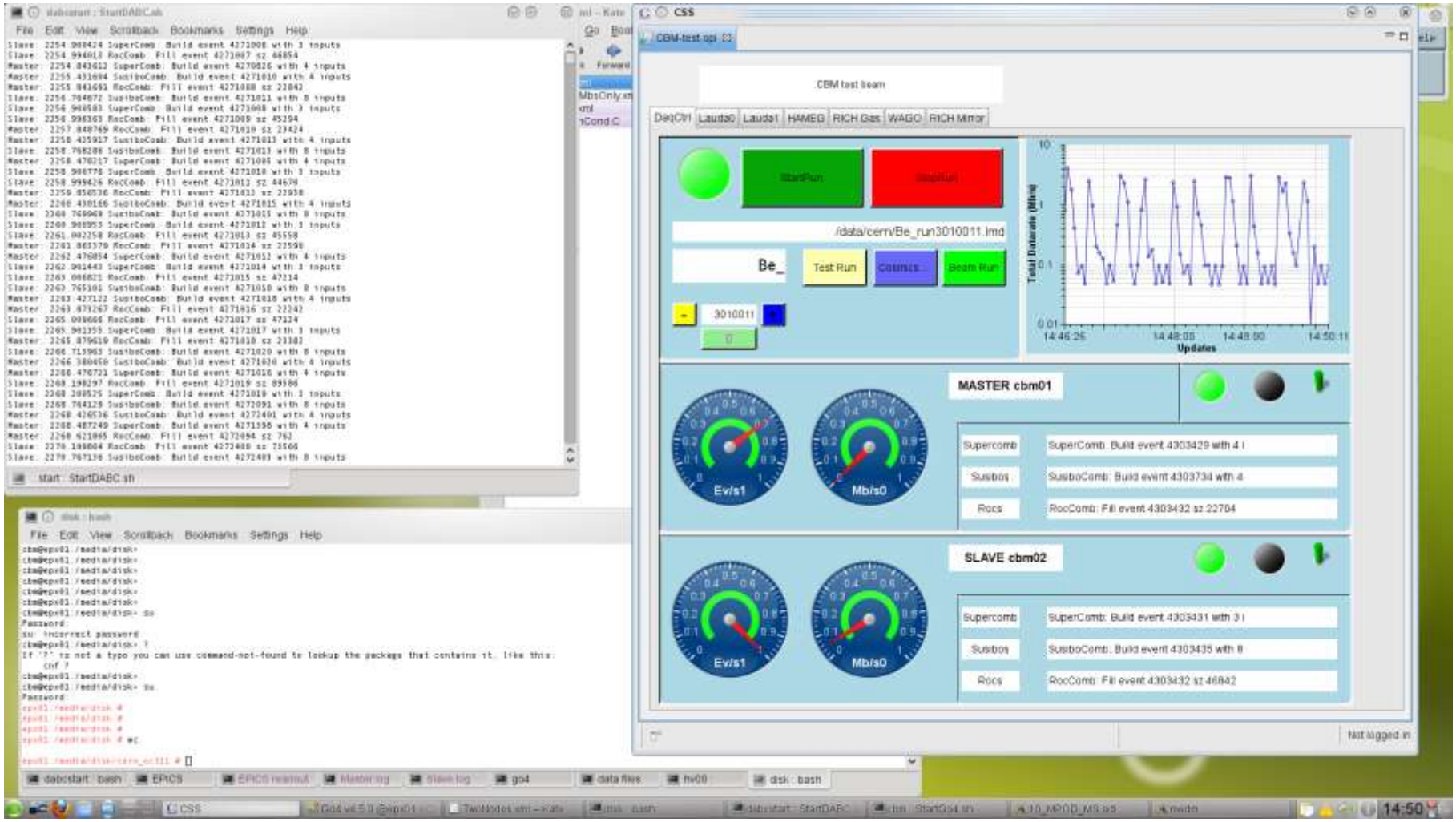
Online monitoring during beamtime



EPICS – DABC – Go4 integration

- EPICS as slow control for DABC
  - using DIM/EPICS interface DABC nodes can be controlled via EPICS-based GUIs
- DABC as readout of EPICS variables
  - as alternative storage of slow-control data
- Both DAQ and slow-control data available in Go4 analysis

EPICS control GUI during beamtime



Conclusion

- DABC is developed as general-purpose DAQ software framework, running on any Linux PC
- Since 2008 used by CBM collaboration as DAQ system in many test beams and electronic/detectors tests
- Easily can be integrated with other DAQ systems like MBS
- Provides connection to online analysis and control systems