DAQ and online software development for CBM experiment

Sergey Linev,
Experiment Electronic, GSI
The Compressed Baryonic Matter Experiment

- Dipol magnet
- Ring Imaging Cherenkov Detector
- Silicon Tracking Stations
- Transition Radiation Detectors
- Vertex Detector
- Resistive Plate Chambers (TOF)
- Electro-magnetic Calorimeter
- Projectile Spectator Detector (Calorimeter)

P. Senger – 18th CBM Collaboration Meeting, Beijing, 29.9.2011
Experimental challenges

- $10^5 - 10^7$ Au+Au reactions/sec
- determination of (displaced) vertices with high resolution ($\approx 50 \, \mu m$)
- 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


- Self-triggered front-end
- All hits shipped to FLES

First-level Event Selector

Permanent Storage

FLES

1 TByte/s Total Input Data Rate

~1000 Input Nodes
- Fast PCIe interfaces

High-throughput event building
- InfiniBand QDR

Event selection in FLES processor farm
- High-throughput online analysis
- $10^7$ Events/s
- Vectorization and many-core architectures
- ~ 60,000 Cores
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

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

Explicit loop in ModuleSync

```cpp
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

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

Sergey Linev, DAQ and online software for CBM experiment, 5.11.2011, Zagreb
Data splitter for optic data

Abb

Input

Output0

Output1

Output2

Output3

Output4

Output5

Output6

Output7

ABB readout

ROCs

MBS

DABC/EPICS

ROC splitter

ROC combiner

Super combiner

Lmd file storage

Input

Input0

Input1

Input2

Input3

Input4

Input5

Input6

Input7

Output

Output0

Output1

Output2

Output3

Output4

Output5

Output6

Output7

ROC

MBS

EPICS
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)
Analysis organization for CBM beamtimes

**Diagram:**

1. **DABC** 
2. **MBS event** 
3. **processor** 
4. **DAQ event** 
5. **Detector event** 
6. **Detector**

**Links and Tools:**

- **DAQ specific ("Unpack")**
- **Detector specific ("Detector")**
- **ROOT Tree I/O**
- **EPICS**
- **ROC**
- **SPADIC**
- **MBS**
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