Data Acquisition Backbone Core library

Jörn Adamczewski-Musch, Hans G.Essel, Nikolaus Kurz, Sergey Linev
GSI, Experiment Electronics: Data Processing group

Performance
Design
Release v1.0
Use cases

Work supported by EU RP6 project JRA1 FutureDAQ RII3-CT-2004-506078

Software developments at GSI (EE)

- **Self-triggered front-end components**
- **Event building over fast networks**
- **1 GByte/s bi-directional @ ~1000 nodes**

Software packages developed:

1. 2005 Simulation with SystemC (flow control, scheduling)
   - Meta data on data network

2. 2006 Real dataflow core (round robin, with/without sychronization)
   - Linux, InfiniBand, GB Ethernet
   - Simulates data sources

Software developments at GSI (EE)

- **Self-triggered front-end components**
- **Event building over fast networks**
- **1 GByte/s bi-directional @ ~1000 nodes**

Software packages developed:

1. 2005 Simulation with SystemC (flow control, scheduling)
   - Meta data on data network

2. 2006 Real dataflow core (round robin, with/without sychronization)
   - Linux, InfiniBand, GB Ethernet
   - Simulates data sources

   - **> 500 Mbyte/s per node bi-directional @ 110 nodes**

Recall: InfiniBand basic tests

Hardware for testing:

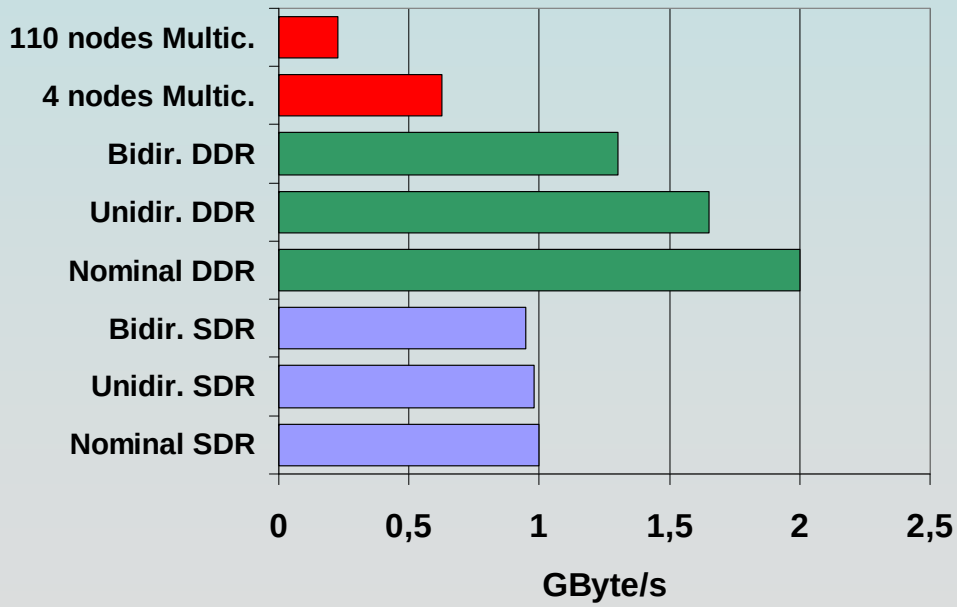
- GSI (November 2006) - 4 nodes, single data rate SDR
- Forschungszentrum Karlsruhe* (March 2007) – 23 nodes, double data rate DDR
- UNI Mainz** (August 2007) - 110 nodes, DDR

Point-to-point tests

	SDR (GSI)	DDR (Mainz)
Unidirectional	0.98 GByte/s	1.65 GByte/s
Bidirectional	0.95 GByte/s	1.3 GByte/s

Multicast tests

2 KByte	Rate per node
GSI (4 nodes)	0.625 GByte/s
Mainz (110 nodes)	0.225 GByte/s

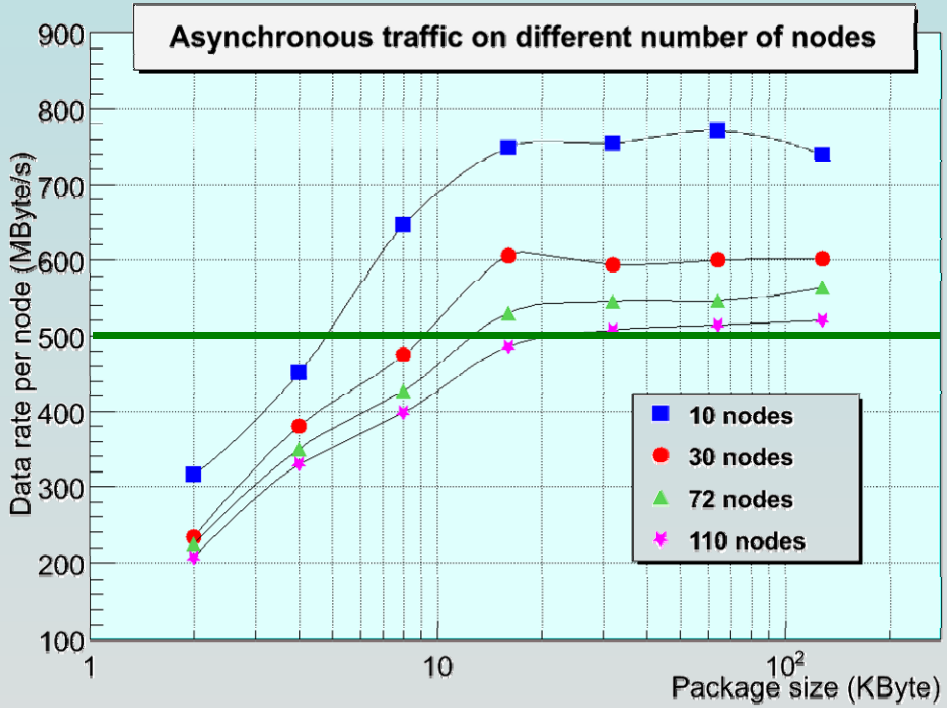


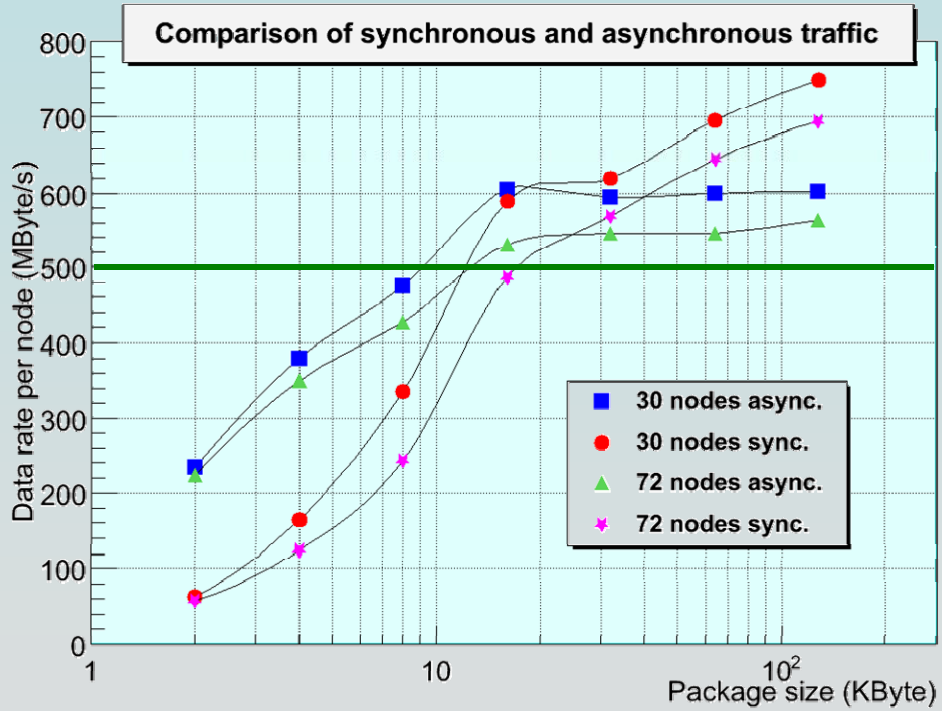
* thanks to Frank Schmitz, Ivan Kondov and Project CampusGrid in FZK
** thanks to Klaus Merle and Markus Tacke at the Zentrum für Datenverarbeitung in Uni Mainz

Recall: Performance scaling

> 500 MBytes/s bidirectional

+ 25%





Topology optimization for 110 nodes would have required different cabling of switches!

Software developments at GSI (EE)

- Self triggered front-end components
- Event building over fast networks
- 1 GByte/s bi-directional @ ~1000 nodes

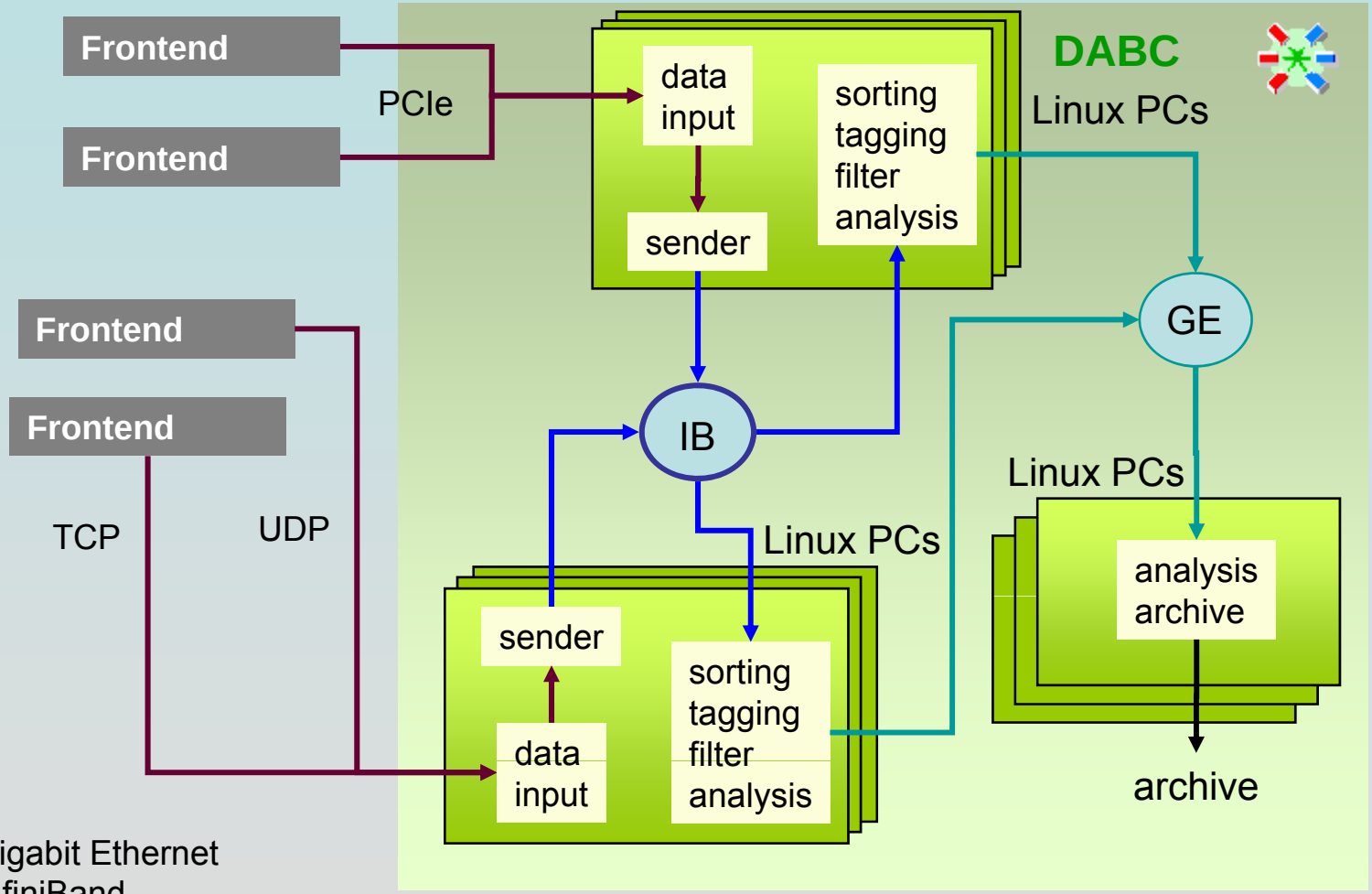
Software packages developed:

1. 2005 Simulation with SystemC (flow control, scheduling)
   - Meta data on data network

2. 2006 Real dataflow core (round robin, with/without sychronization)
   - Linux, InfiniBand, GB Ethernet
   - Simulates data sources

3. **2007/8 Data Acquisition Backbone Core DABC (includes dataflow core)**
   - **Real data sources (arbitrary)**
   - **Applications implemented by plug-ins**
   - **Configuration, Controls, Monitoring, GUI ...**
   - **⇨ General purpose DAQ framework**

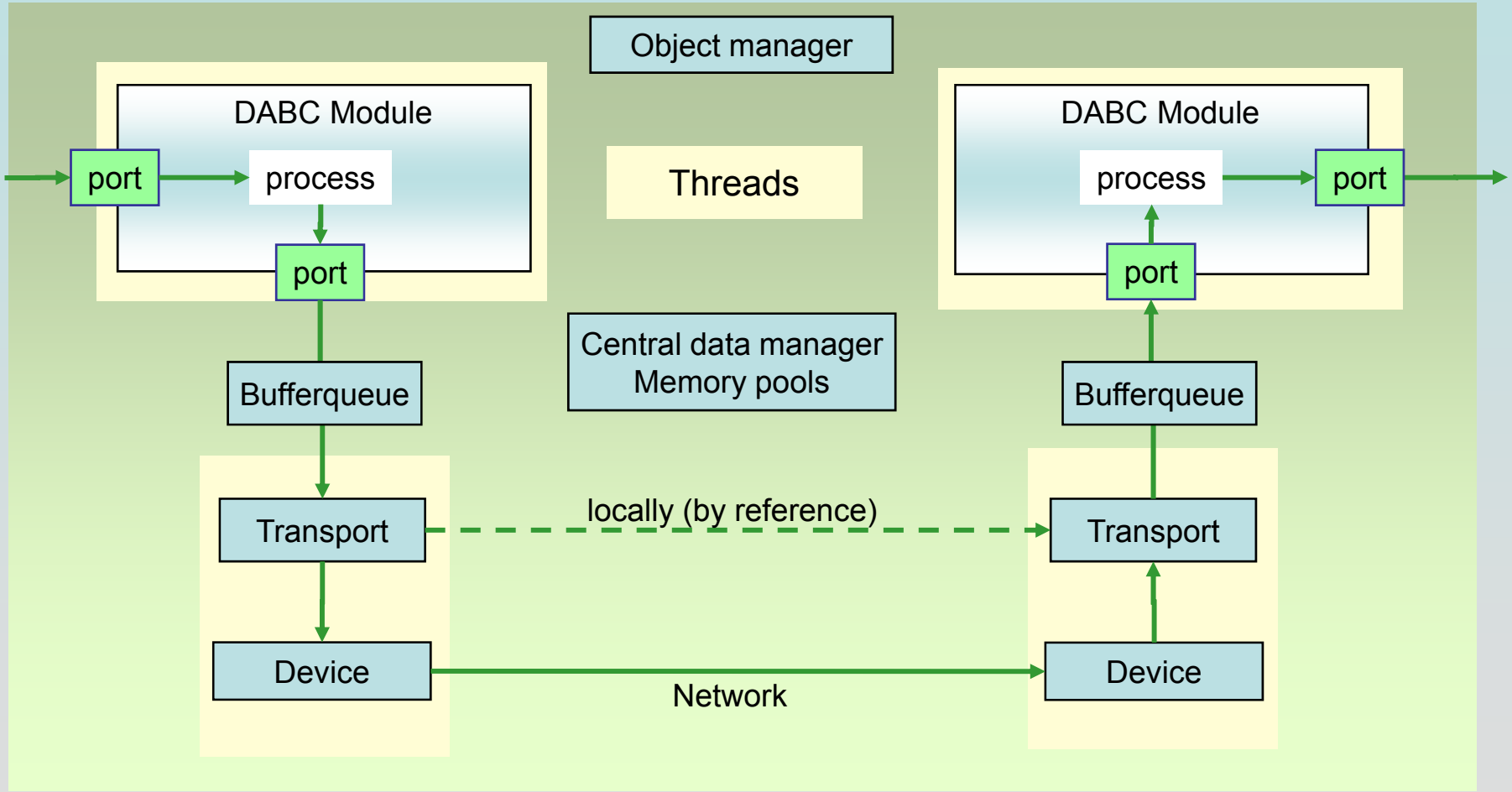
DABC design: global overview



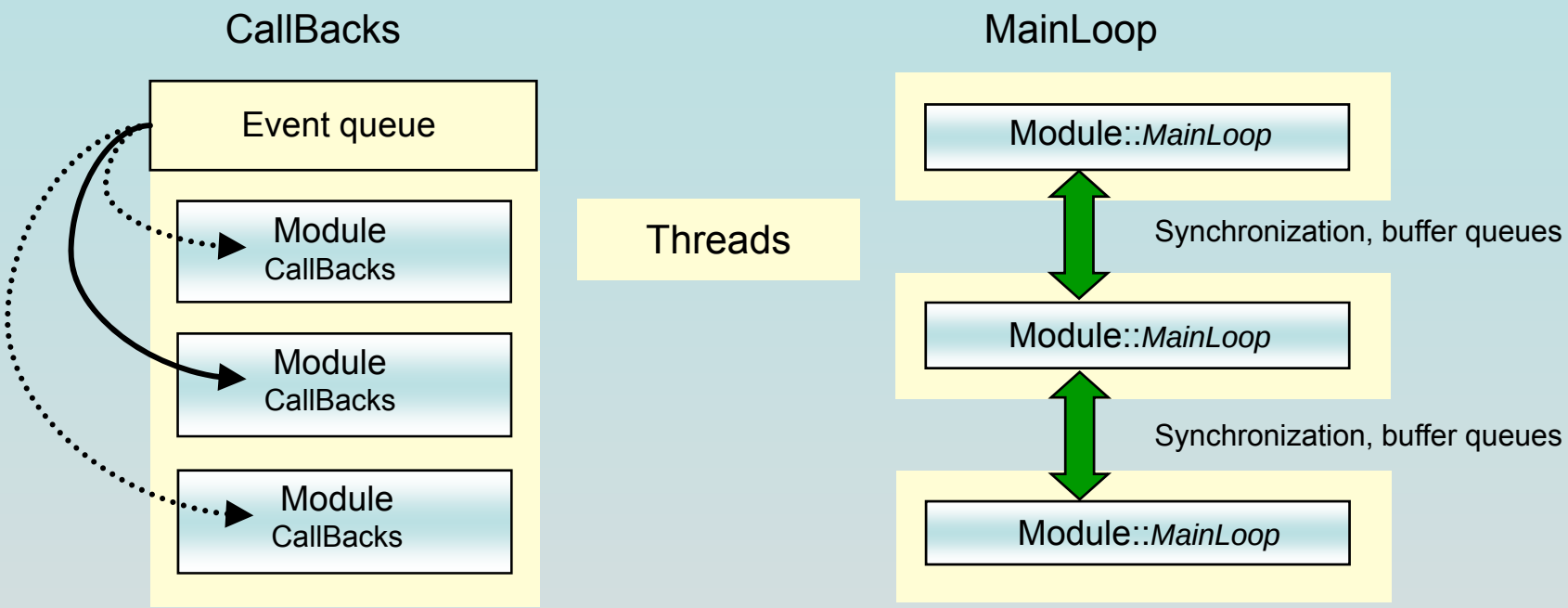
GE: Gigabit Ethernet
IB: InfiniBand

DABC design: software objects of data flow engine

A *module* processes data of one or several data streams.
Data streams propagate through ***ports***, which are connected by ***transports*** and ***devices***

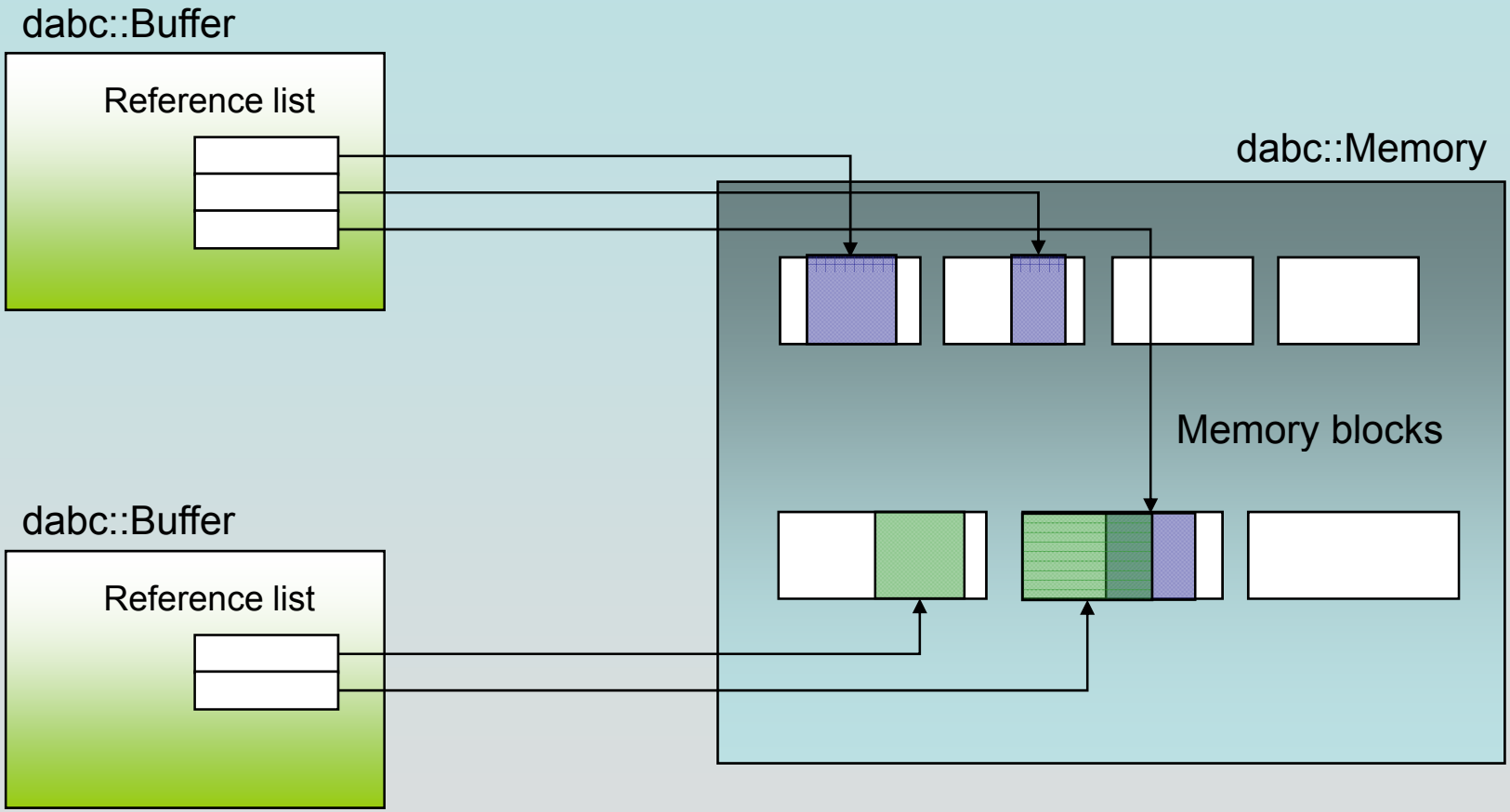


DABC design: single versus multiple threads



CallBacks	MainLoop
Several per thread (assigned by setup)	One per thread
Sequential execution	Parallel execution
Non-blocking resource requests (faster)	Blocking resource request (slower)
For low CPU requirements	For high CPU requirements (multicore!)
For high data flow (no context switch)	For low data flow (context switch)

DABC design: smart buffers



Multiple references of resources (blocks)
Copy buffer without copy of data
PCI/DMA → buffer → IB/DMA

Data flow components and Plug-ins

- **Module** (owned by manager)
  - Each **synchronous module** is executed by a dedicated **working thread**. The thread executes a method *MainLoop()* with arbitrary code, which may block the thread.
  - Several **asynchronous modules** may be run by a **shared working thread**. The thread processes an event queue and executes appropriate callback functions of the module.
- **Port** (owned by module)
  - Buffers are entering and leaving a module through Port objects.
  - Each port has a buffer queue of configurable length.
  - A module may have several **input**, **output**, or **bidirectional ports**.
  - The ports are owned by the module.
- **Transport** (owned by device)
  Outside the modules the ports are connected to Transport objects.
  - Local transport (zero copy)
  - Transport over network
  - Input/output from/to data source/sink (PCI, TCP, file)
- **Device** (owned by manager)
  - Owns and manages transports
  - Controls **transport thread**
  - usually represents an I/O component (e.g. network card, PCI card)
- **Buffer management**
  - Preallocated buffers in pools
  - Counted reference lists (overlapping)

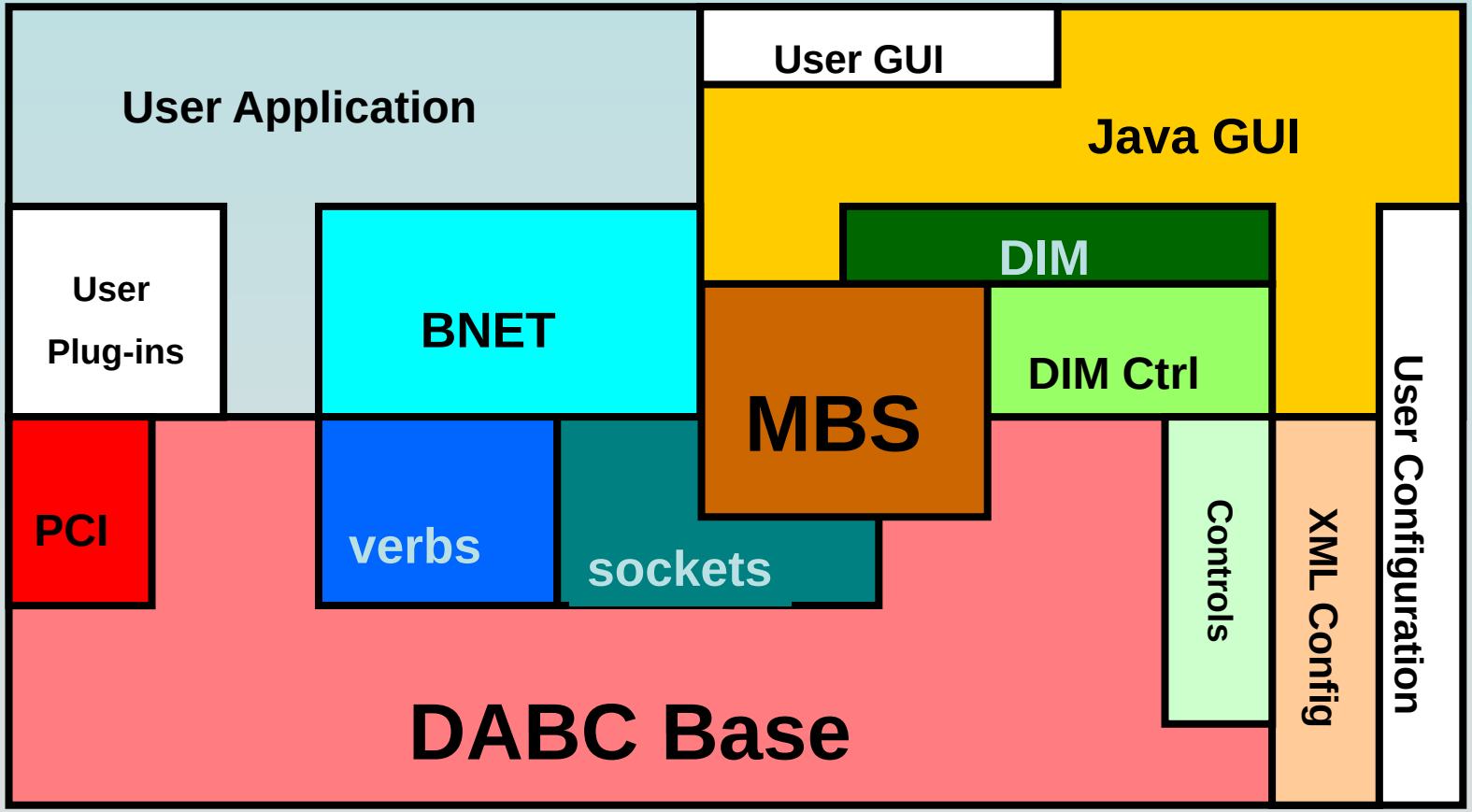
Control components and Plug-ins

- **Finite state machine**
  - Application states
  - Controlled by manager (state transitions) -> application methods
- **Parameters**
  - Structures (predefined)
  - Initialized by setup files
  - Exported to monitoring (update)
  - Maybe changed by commands (GUI)
- **Commands**
  - Managed by manager
  - Execution from control system (GUI) or via *Execute* method
  - Can be dispatched to module or device threads
- **Manager** (singleton, working thread)
  - Object manager
  - Implements control system (commands, parameters)
  - Default state machine
  - DIM server
- **Application** (singleton)
  Setting up the overall topology of a DAQ process

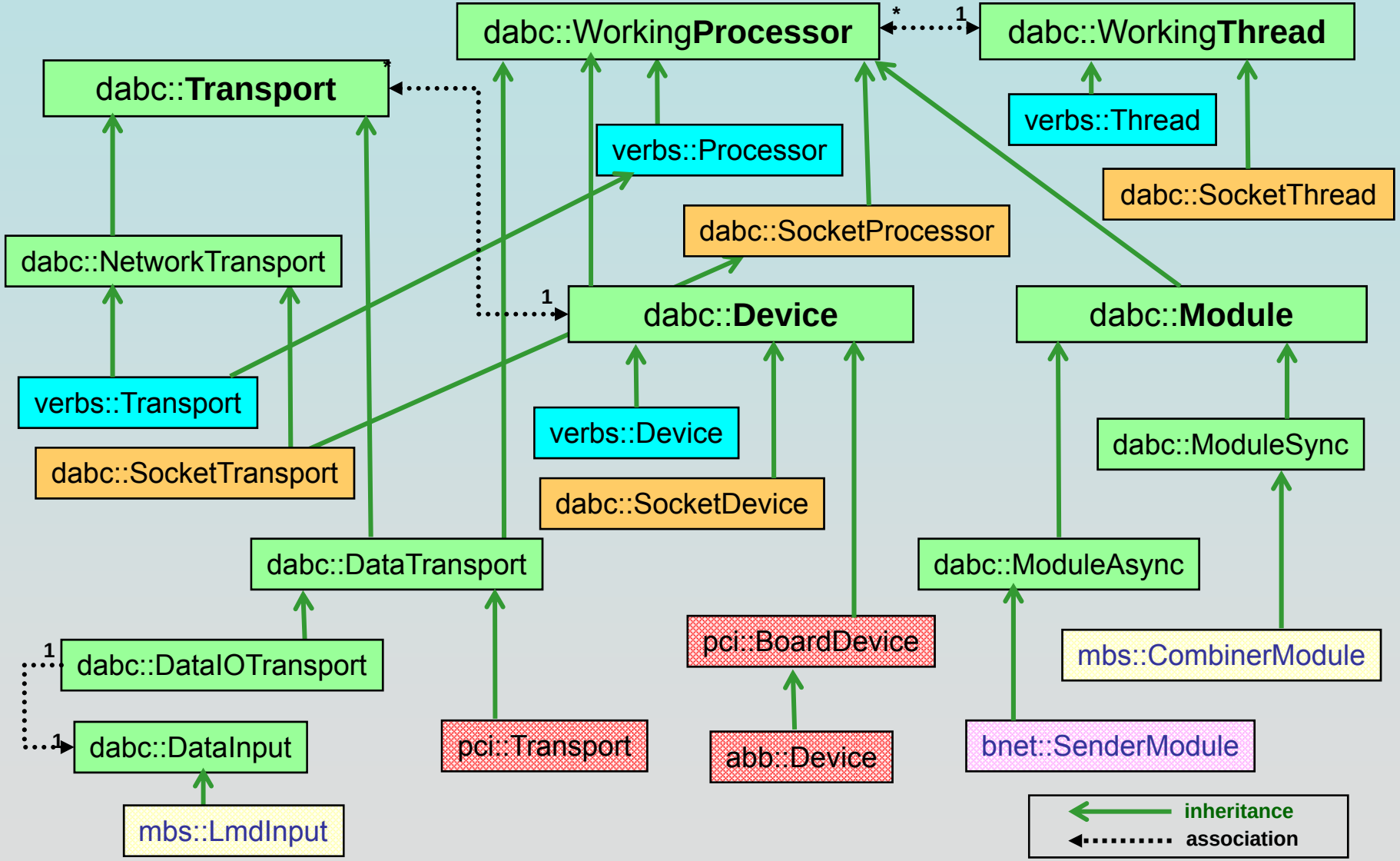
Plug-ins

- **Plug-ins**
  - Modules, transports, devices, application by subclasses and interfaces
  - Compiled and linked into shared library
  - Loaded by name by DABC executable
  - Interface methods called at state transitions
- **Design**
  - Flexibility by combination of code and setup
  - Data flow topology established by methods of application
  - Use of threads depending on performance requirements

Packages

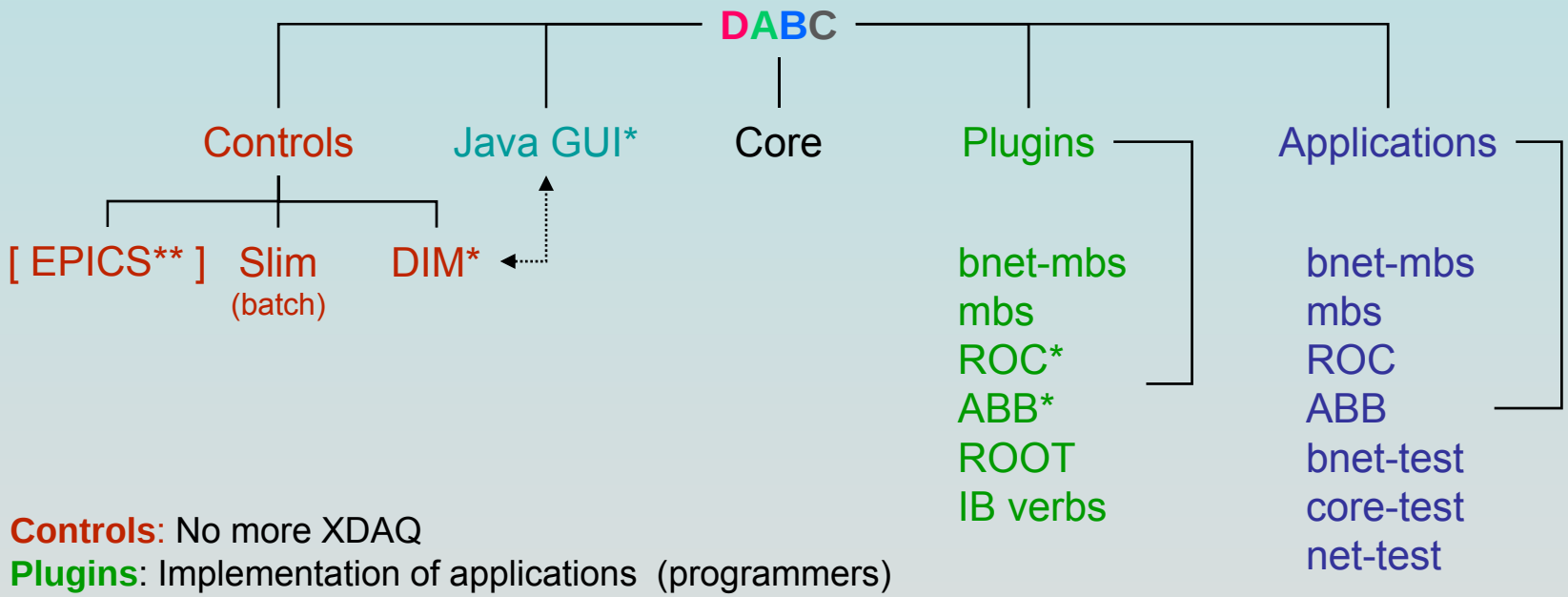

User Application
User GUI
Java GUI
User Plug-ins
BNET
DIM
MBS
DIM Ctrl
PCI
verbs
sockets
Controls
XML Config
User Configuration
DABC Base

Classes


dabc::WorkingProcessor
dabc::WorkingThread
dabc::Transport
verbs::Thread
verbs::Processor
dabc::SocketThread
dabc::SocketProcessor
dabc::NetworkTransport
dabc::Device
dabc::Module
verbs::Transport
verbs::Device
dabc::ModuleSync
dabc::SocketTransport
dabc::SocketDevice
dabc::DataTransport
dabc::ModuleAsync
pci::BoardDevice
mbs::CombinerModule
dabc::DataIOTransport
dabc::DataInput
pci::Transport
abb::Device
bnet::SenderModule
mbs::LmdInput
inheritance
association

DABC release v1.0.01

Download via dabc.gsi.de



Controls: No more XDAQ
Plugins: Implementation of applications (programmers)
Applications: Mainly setup or testing programs (users)

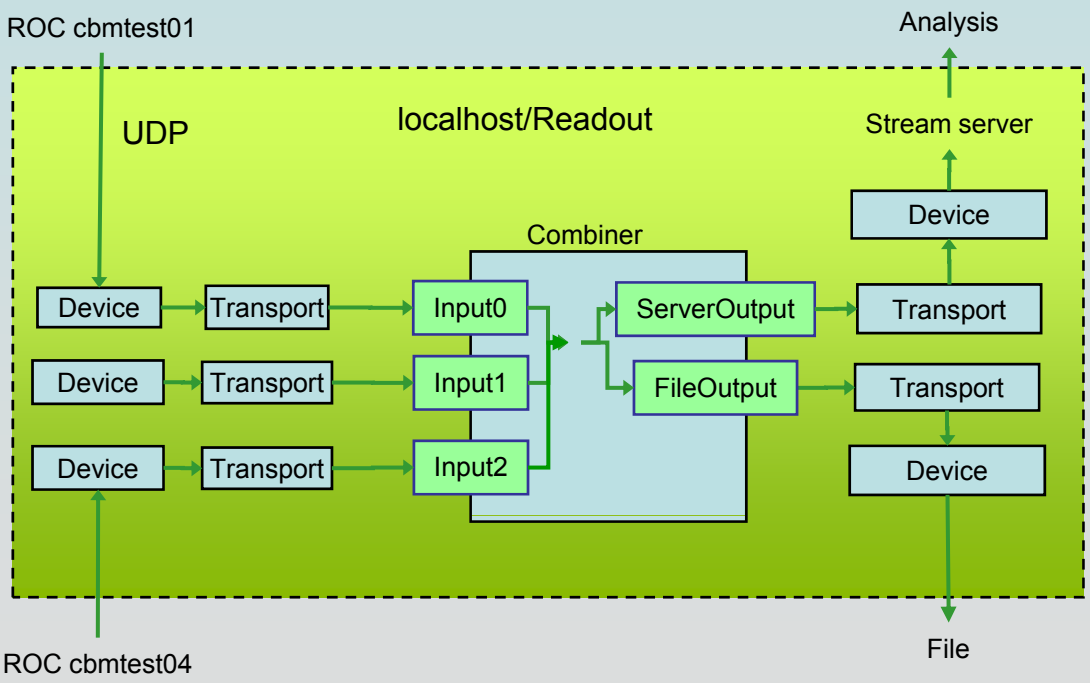
* external packages needed
** future?

ROC: ReadOutController board (UDP)
ABB: ActiveBufferBoard (PCIe)
IB: InfiniBand
mbs: MultiBranchSystem (GSI DAQ)

ROC: Three ROCs to raw file and stream server

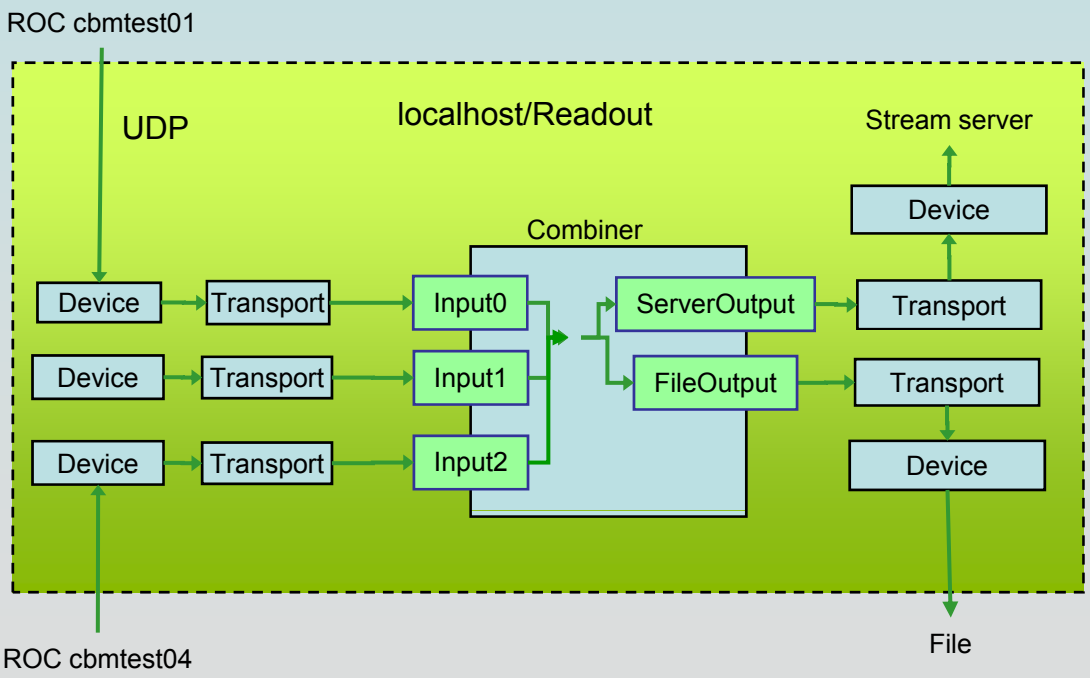
Testbeam for nXYTER readout chain

Poster 302, Thursday:
Integration of CBM readout controller into DABC framework



ROC: Three ROCs to raw file and stream server

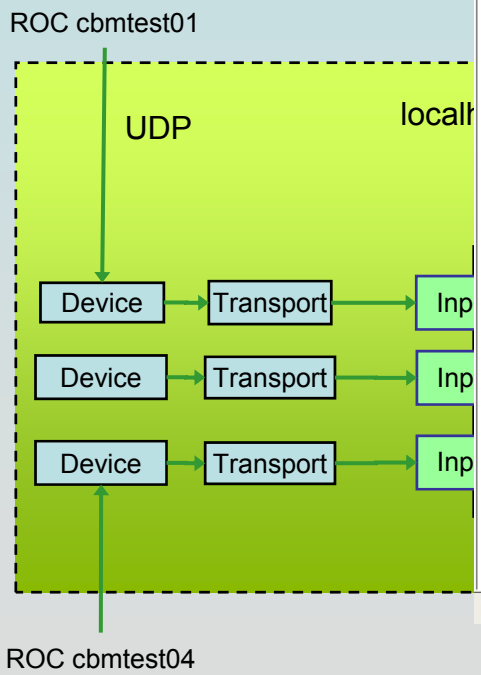
- Class ***Readout*** implements ***CreateAppModules***
- **Application** parameters known to all modules

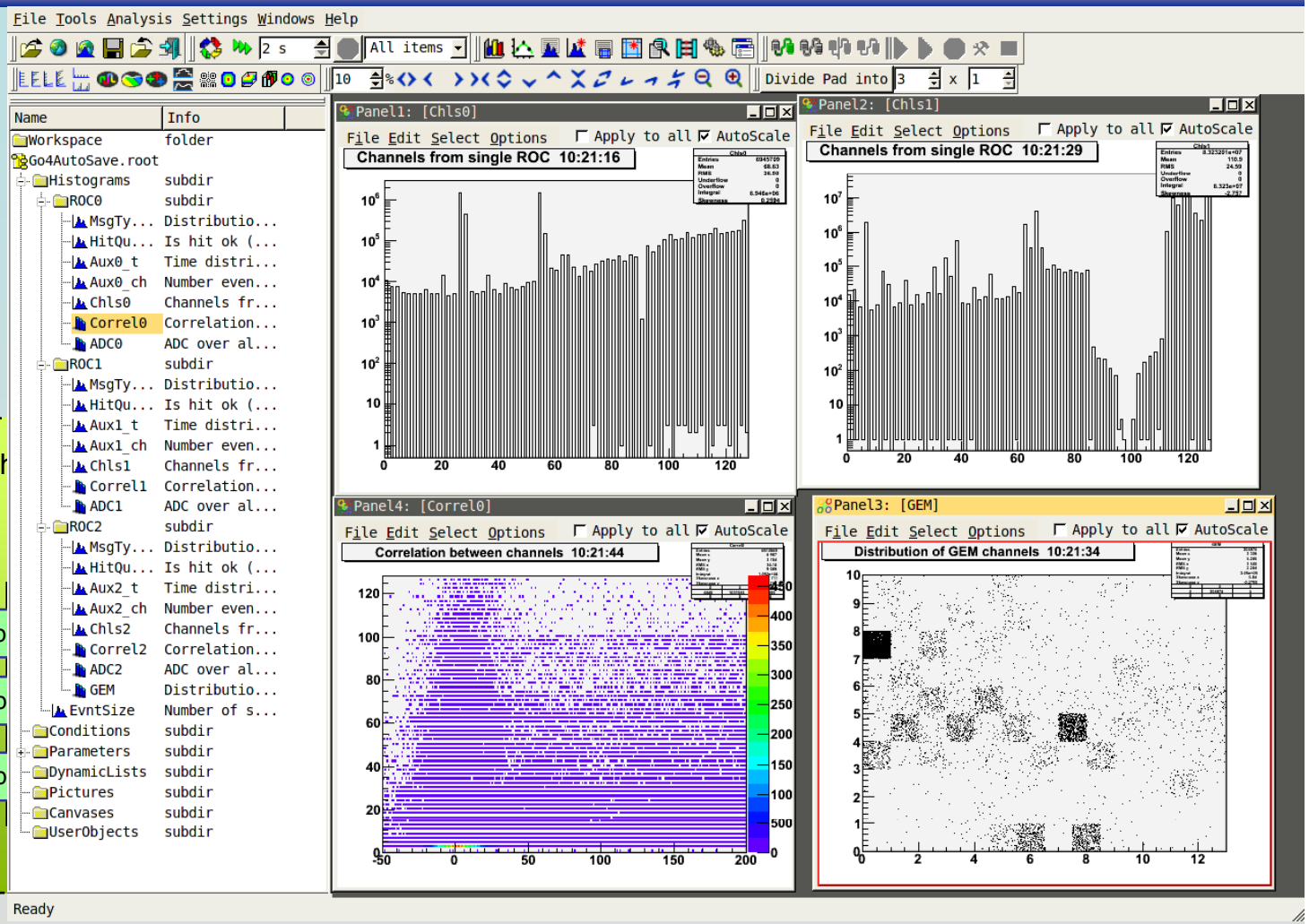



```xml
<?xml version="1.0"?>
<dabc version="1">
<Context name="Readout">
  <Run>
    <lib value="libDabcMbs.so"/>
    <lib value="libDabcKnut.so"/>
  </Run>
  <Application class="roc::Readout">
    <DoCalibr value="0"/>
    <NumRocs value="3"/>
    <BufferSize value="65536"/>
    <NumBuffers value="100"/>
    <RawFile value="run090.lmd"/>
    <MbsFileSizeLimit value="110"/>
    <RocIp0 value="cbmtest01"/>
    <RocIp1 value="cbmtest02"/>
    <RocIp2 value="cbmtest04"/>
    <TransportWindow value="30"/>
    <MbsServerKind value="Stream"/>
  </Application>
 </Context>
</dabc>
```

ROC: Three ROCs to raw file and stream server

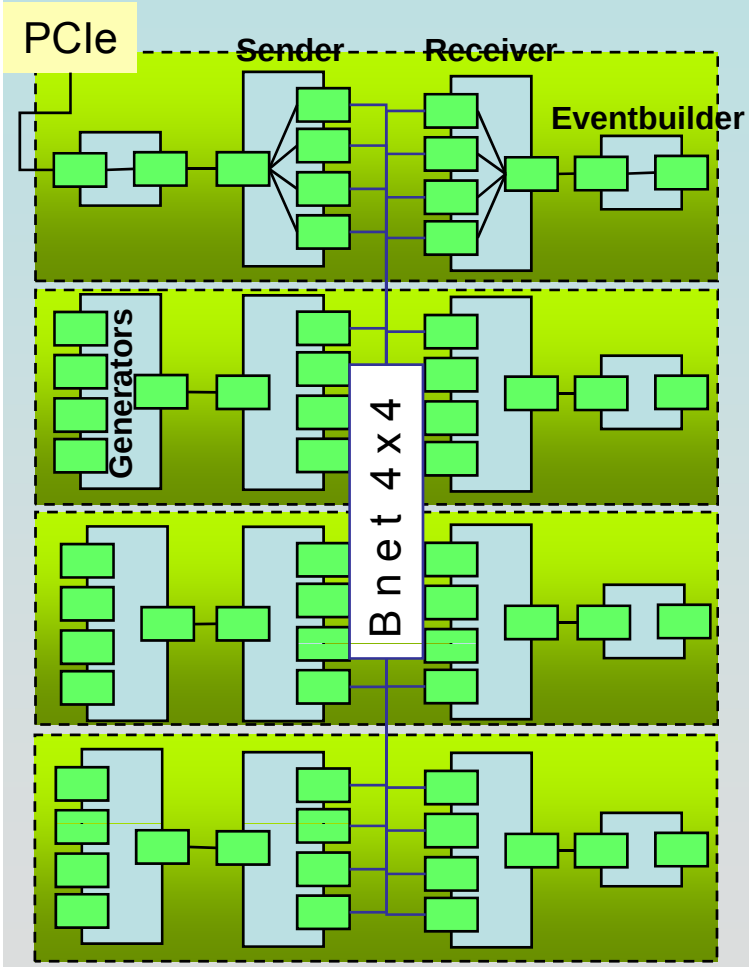
On-line analysis with Go4





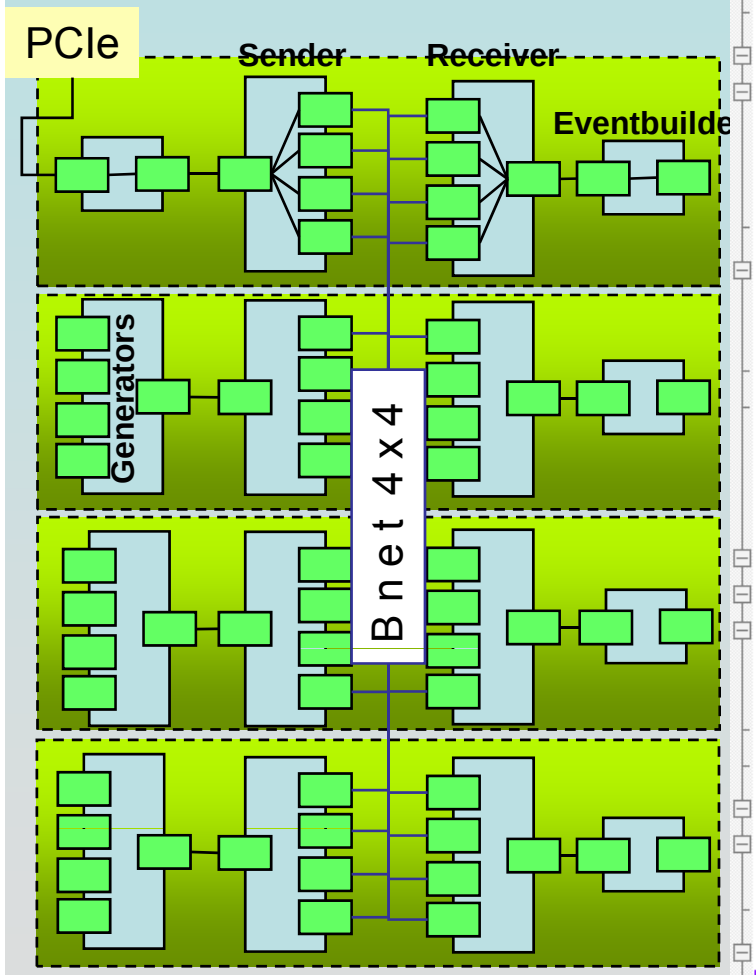
File

PCIexpress board and Bnet



GSI DABC

PCIexpress board ar




```
<Context host="master" name="Controller">
  <Run>
    <runfunc value="RunTestBnet"/>
  </Run>
  <Application class="bnet::Cluster">
    <NetDevice value="verbs::Device"/>
  </Application>
</Context>
<Context host="master" name="Worker1">
    <Run>
      <lib value="${DABCSYS}/lib/libpcidriver.so"/>
      <lib value="${DABCSYS}/lib/libmprace.so"/>
      <lib value="${DABCSYS}/lib/libDabcAbb.so"/>
    </Run>
  <Application class="bnet::TestWorker">
    <NumReadouts value="1"/>
    <InputOCfg value="ABB"/>
  </Application>
</Context>
<Context host="node01" name="Worker2"/>
<Context host="node02" name="Worker3"/>
<Context host="node03" name="Worker4"/>
<Defaults>
  <Context name="*">
    <Run>
      <lib value="libDabcVerbs.so"/>
      <lib value="libDabcBnet.so"/>
    </Run>
  </Context>
  <Context name="Worker*">
    <Run>
      <lib value="${DABCSYS}/applications/bnet-test/libBnetTest.so"/>
    </Run>
    <Application class="bnet::TestWorker">
      <IsSender value="true"/>
      <IsReceiver value="true"/>
      <NumReadouts value="4"/>
    </Application>
  </Context>
</Defaults>
```

PCI board

Three data generators

For all contexts

For all workers

FOPI experiment on-line