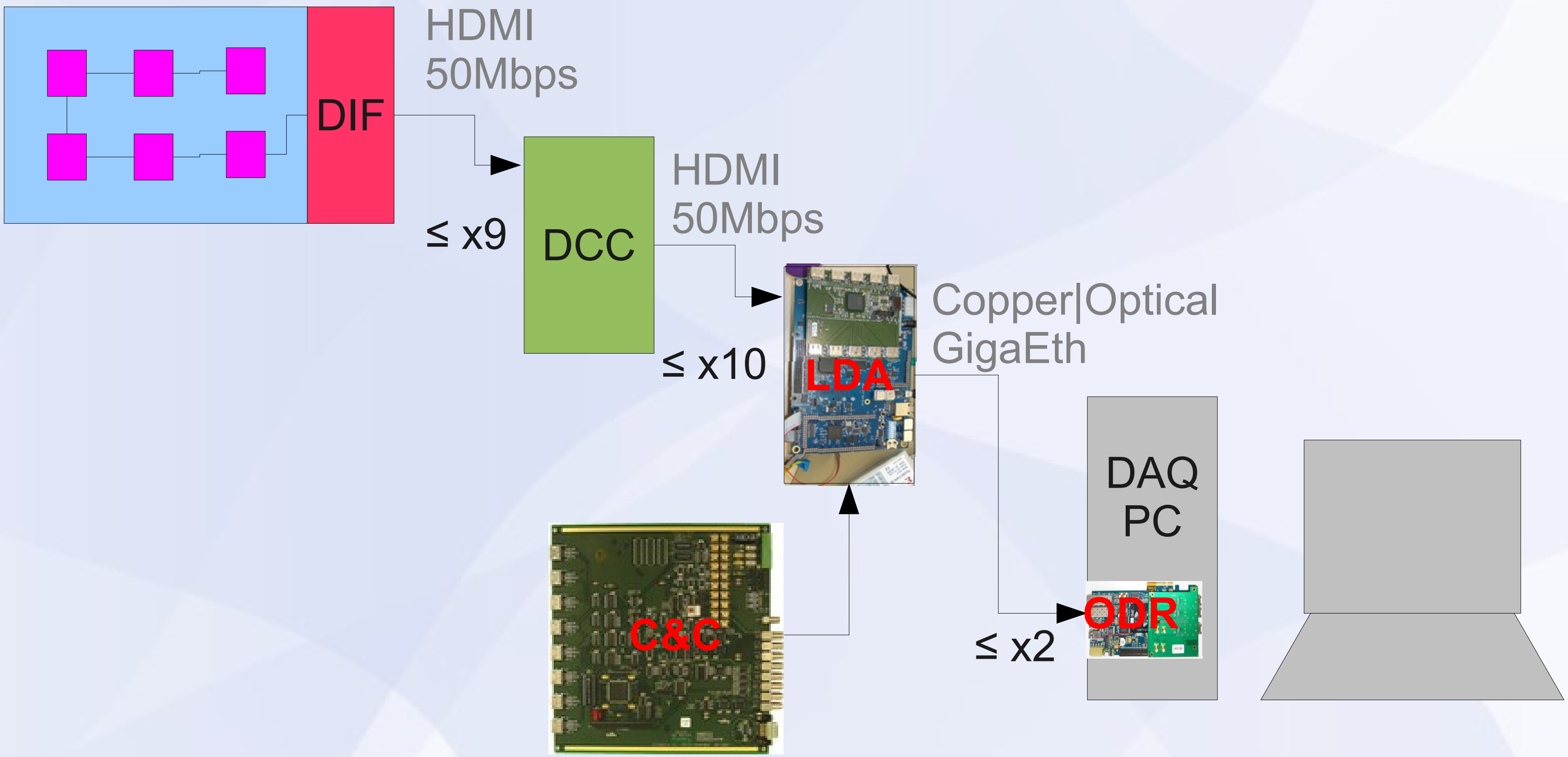
Calice DAQ & testbeams: status and perspectives

David Decotigny (LLR)
First part in collaboration with Valeria Bartsch (UCL)
With notes from Paul Dauncey (Imperial)
“BIF” slides from the BIF team

Overview

- Current Calice “DAQ2” Software status
- Foreseen full setup
- Slow control/supervision: plans and open questions
- Data-readout path: plans and open questions
- Monitoring the beam: “BIF” proposals
- Interfacing with the machine

Calice "DAQ2" System overview



A network-oriented setup: a tree of data concentrator cards

Hardware/Firmware status

- LDA+ODR+C&C+DAQ PC:
  - Hardware Provided to LLR by UCL/RHUL/Manchester
  - Firmware developed by UCL/RHUL/Manchester

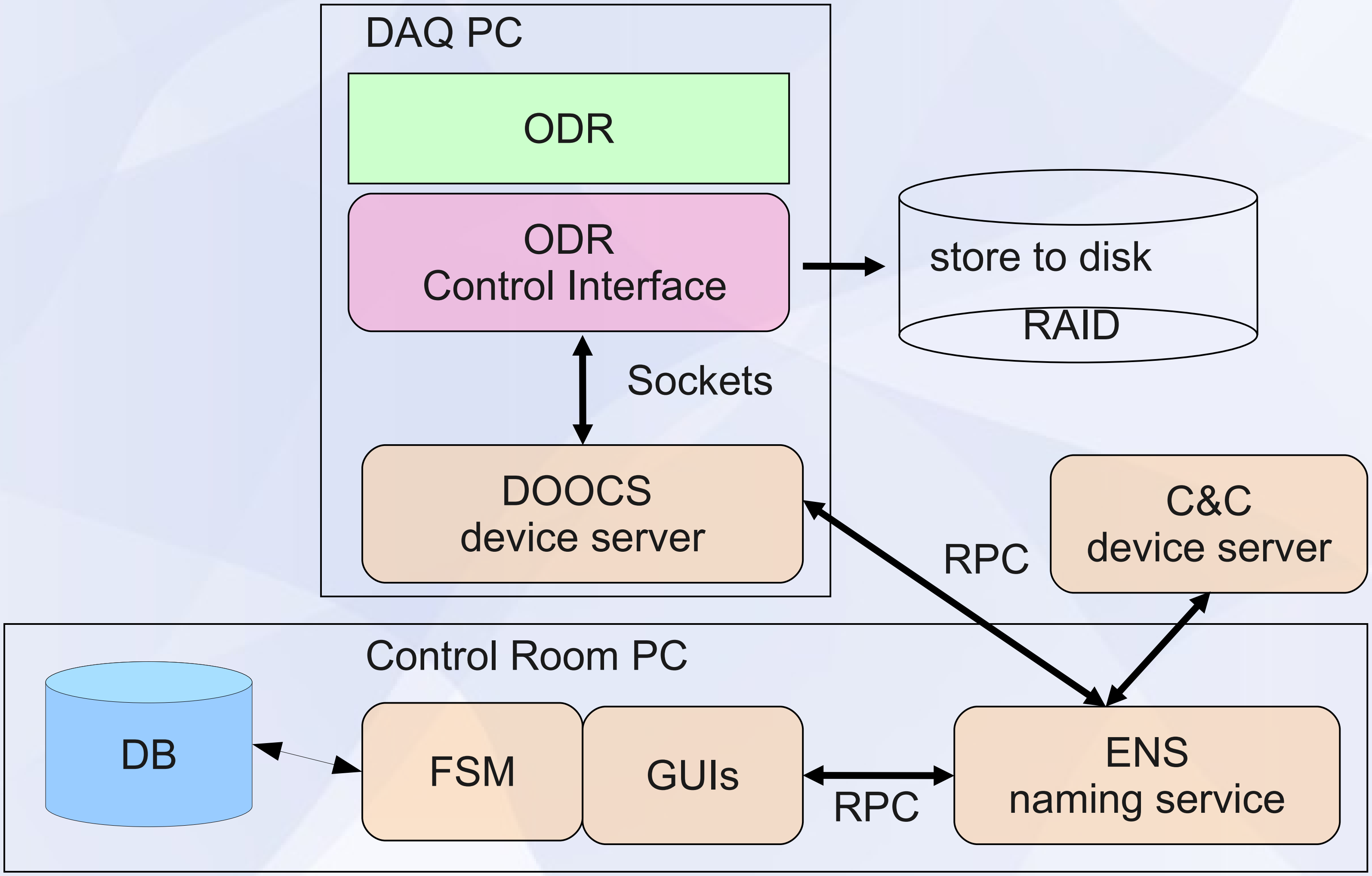
- DCC:
  - Hardware available @LLR
  - Firmware developed by F. Gastaldi

- DIF:
  - Several prototypes available @LLR
  - DIF Task Force for the firmware

Software status

- Prototype running on DOOCS
  - ODR kernel driver by A. Misiejuk (Manchester)
    - Able to store data packets to disk
  - CCC interface for DOOCS
    - Register accesses
  - DAQ Software by V. Bartsch (UCL)
    - GUIs to control the ODR + CCC
    - FSM to control the device servers
    - DB framework to retrieve configs

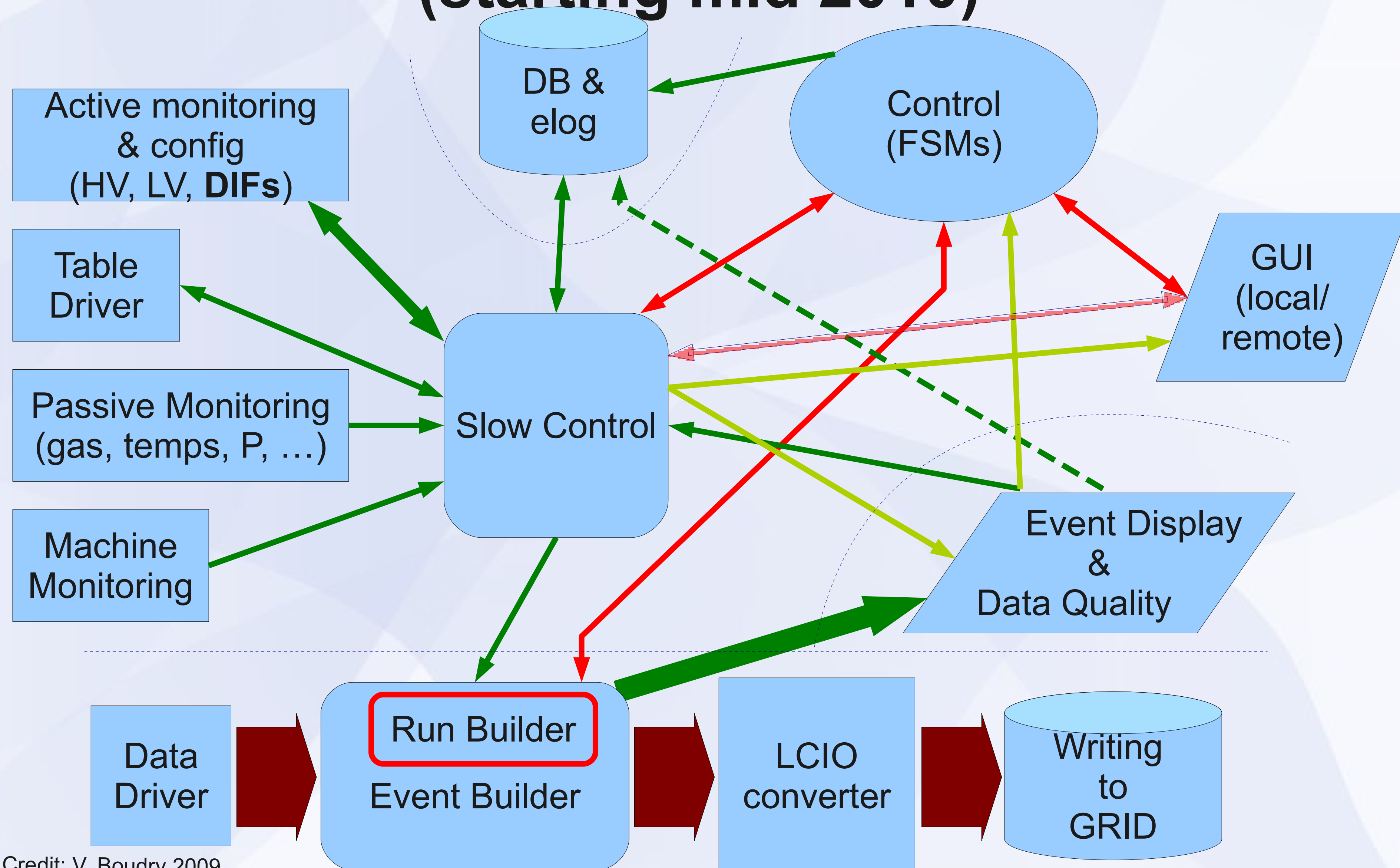
DAQ SW architecture



DAQ Prototype: short-term perspectives (by 2010)

- Integrate/debug the whole chain
  - Validate ODR/LDA connection
    - Could test PC-LDA with a basic Eth card
  - Validate LDA/DCC connection
  - Validate DCC/DIF connection
- Assess the performances of read-out reassembly algos

Longer-term: foreseen setup
(starting mid 2010)



Credit: V. Boudry 2009

Slow-control: plans and open questions

Slow-control/supervision Frameworks

- Many SCADA frameworks available for HEP:
  - DAQ1, EUDAQ, Xdaq, Tango, EPICS, Labview, PVSS, MIDAS, home-made, …
- Many different frameworks used in ILC testbeams
- For Calice:
  - Xdaq+Labview used at IPNL
  - DAQ2: keep DOOCS for now and will evaluate Xdaq and/or Tango

Slow-control/supervision Preparing for integration issues

- **Integration problem**: various subdetectors using different frameworks need to be integrated together for the same testbeam
- This may not seem a hard problem:
  - Some of the frameworks rely on the same basic concepts (DOOCS, Epics, Tango, ...)
    - Usually: subscribe-update semantics
  - Others are generic enough to “theoretically” be interfaced with others (Xdaq, ...)
- <u>But:</u> glue code / lengthy development and tests needed for each testbeam

Slow-control/supervision Preparing for integration issues

- Another solution: general consensus on a framework for ILC
- Another solution (P. Dauncey):
  - require the sub-detectors to follow a general configuration method. The main software calling it is not dependent on the front-end hardware to configure
  - Generalize this approach to other slow-control aspects ?

Slow-control/supervision Preparing for integration issues

- (Some other) slow-control subsystems to consider, same integration problems:
  - Configuration
  - System dynamics (FSM)
  - Error handling, logging
  - Online monitorimg: rendering with ROOT ? FROG ? Other ?
  - Trending database
  - ...

Data-readout path: plans and open questions

Data-readout path Frameworks

- Fewer frameworks available:
  - EUDaq, DAQ1, Xdaq, ICE, Narval, home-made (file/socket-based), …
- Many different frameworks used in ILC testbeams:
  - Examples: EUDaq, DAQ1, Narval, home-made
- For Calice:
  - File-based (text+binary) at IPNL
  - File-based (“simple serialization”) by P. Dauncey
  - DAQ2: will *try* to create LCIO event files directly
    - Performance evaluation will be required

Data-readout path
Preparing for integration issues

- **Integration problem**: various subdetectors using different frameworks need to be integrated together for the same testbeam
- The goal is always the same: transfer data
  - “Only” reformatting needed if frameworks change
  - Impairs the performances
- Might not be a hard problem for Calice (eg. max 460kB per readout at LDA in DHCAL). Might be a hard problem for others

Data-readout path
Preparing for integration issues

- Another solution: general consensus on a data-readout framework
  - including hardware (eg. DAQ2 ? CAPTAN ?) ?
- Another solution: use at least a common way to transport the data to ease plugging the testbeam softwares onto each-other ?
  - Could be a subset of EUDaq: Emlyn Corin thinking about this
  - Should also decide on a common data format ?

Data-readout path
Preparing for integration issues

- (Some) other aspects to consider, new integration issues:
  - Correctly identify readout data coming from different subdetectors as belonging to the same physical event
    - Uniquely identify each trigger ? Specialized hardware (TLU/CCC ?)
    - P. Dauncey: order the readout phases...
      - Need to evaluate this solution for network-oriented DAQ

Data-readout path
Preparing for integration issues

- (Some) other aspects to consider, new integration issues (ctd.):
  - Data volumes: single machine able to store ?
    - Specific hardware needed ?
    - "Intelligent" load-balancing switches to transfer data to PC clusters ?
    - Zero-suppression, selective readout, on-the-wire lossless compression ? Other ?
    - ➔ Not really a concern for Calice... how about the other subdetectors ?
  - Storage data format ?

Monitoring the Beam: BIF proposals

Monitoring the beam

- ➔ To determine the accurate actual characteristics of the beam
- Usually: readout of Scintillators, Cerenkov, MWPC, fiber hodoscopes, etc. specific and/or supported by the hosting testbeam facility
  - Need to adapt the setup for each testbeam location
- “Beam Interface Card” (BIF): common hardware to monitor the beam
  - Proposals by V. Boudry, R. Cornat, F. Gastaldi (LLR) and J. Prast (LAPP)
  - Hardware card + associated driver software
    - Compatible with the ILC acquisition modes
      - Including auto-triggered: would allow to have an accurate insight into the beam characteristics for each recorded event
    - Readout of a wide range of signals used by the devices above

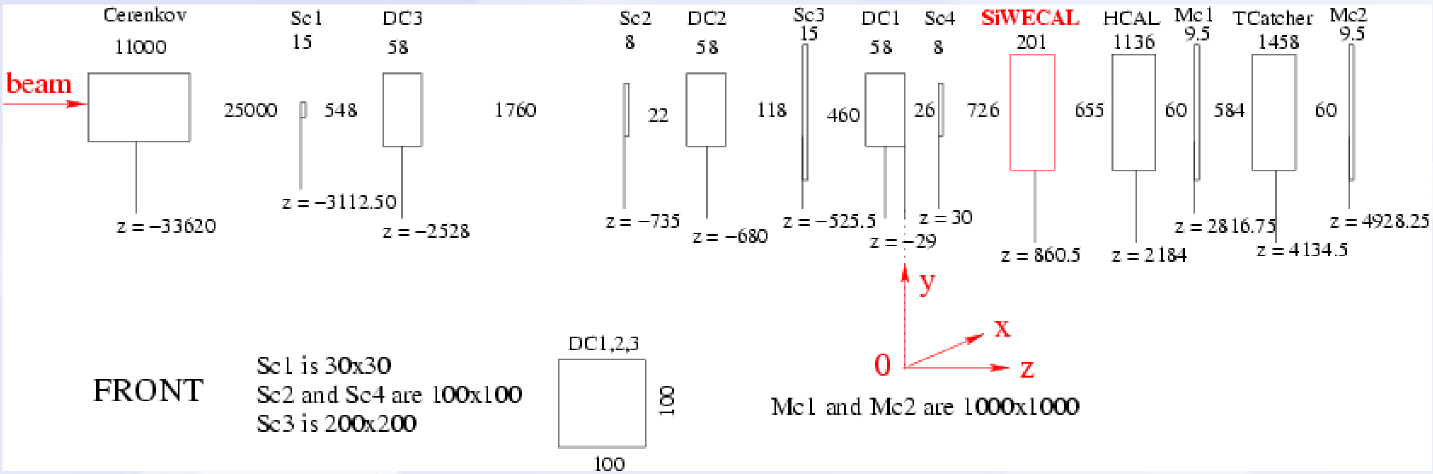
A BIF proposal

- A ROC-based solution:
  - 1 or 2 ROC receive the external signals and store them
  - Pros/cons:
    - ROC Number of channels seem large enough (eg. 10 scint, 16 C PM, 8 MWPC)
    - SPIROC stores time+amplitude: needed for MWPC
    - Memory size might be too short for DHCAL
      - use 1 HR for digital signals (Scint, C PM), 1 SPIROC for MWPC ?
    - Trigger logic provided by the outside world
  - Readout by a standard DIF
  - More demanding in hardware design manpower

Another BIF proposal

- A DIF-based solution
  - Adapter board + modified DIF
    - Existing hardware, FPGA-based
  - Pros/cons:
    - Cheap+simple adaptation board (~ connectors)
    - Large memory needed onboard FPGAs
    - No amplitude recording
    - “Easily” reconfigurable trigger logic
  - More demanding in VHDL manpower

Interfacing with the Machine



CERN H6 (Credit:ECAL 2006 TB@CERN / M. Faucci Giannelli, 9/2009)

Interfacing with the machine Monitoring the environment

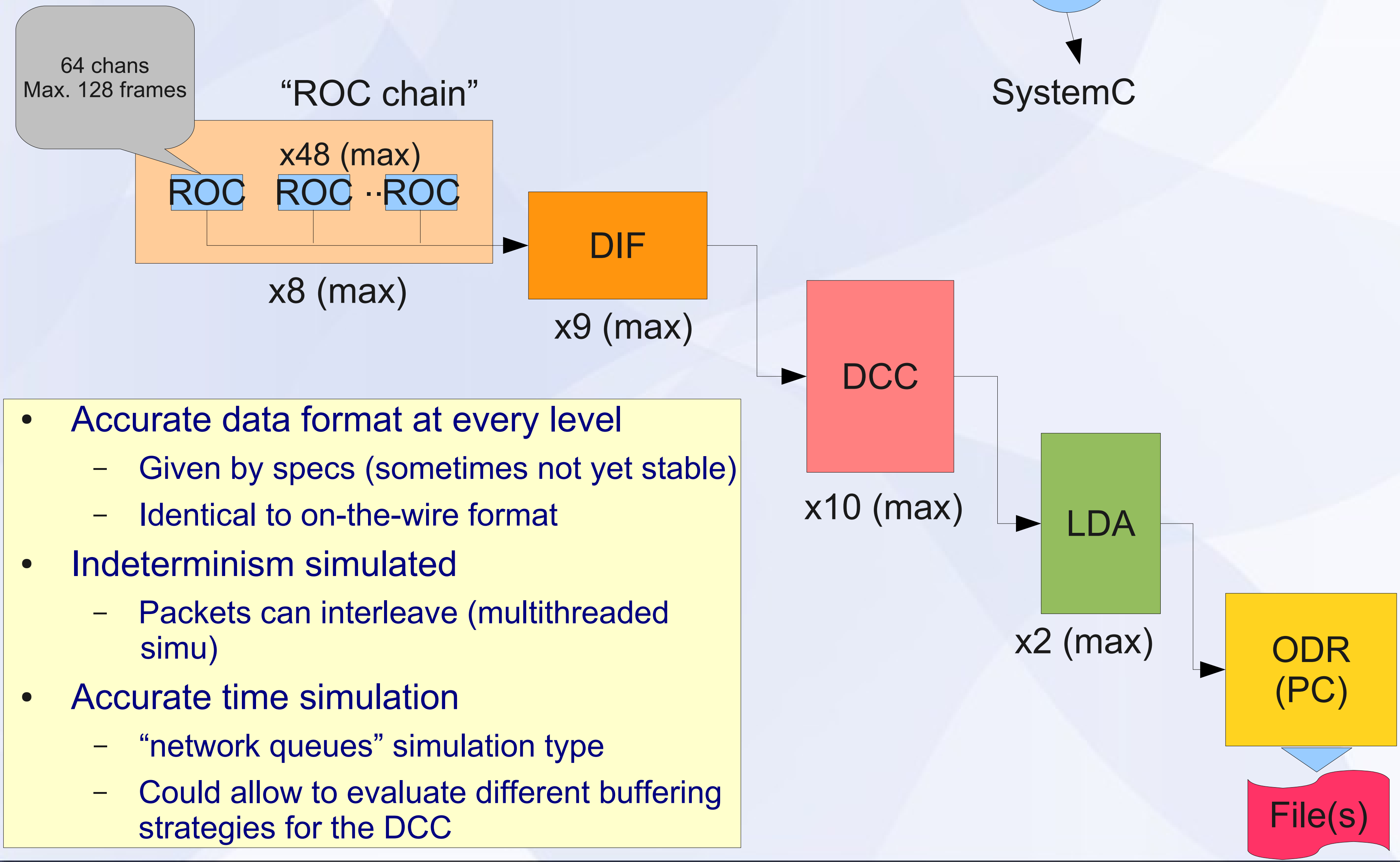
- To get get knowledge about:
  - beam energy & particle type (←→ magnets current & upstream collimators position)
  - beam intensity ("counting devices")
  - final collimators position
  - beam profile (monitoring chambers histograms)
  - Cherenkov pressure
  - and others: existing environmental measures (P, t, humidity), …
- Have a common API for all the machines ?
- All this with fast access (≤ 30s)
- An open question ?

Thank you

Backup slides

A simulation of the read-out chain

Read-out simulation: scsim



- Accurate data format at every level
  - Given by specs (sometimes not yet stable)
  - Identical to on-the-wire format
- Indeterminism simulated
  - Packets can interleave (multithreaded simu)
- Accurate time simulation
  - "network queues" simulation type
  - Could allow to evaluate different buffering strategies for the DCC

Read-out simulation: scsim

- Example:
  - 1 ODR: 2 LDA * 2 DCC * 2 DIF * 7 chains * 48 ROCs

```
0 s: send read-out signal...
Received pkt (1052 bytes) at 4312616 ns
Received pkt (1052 bytes) at 4312616 ns
Received pkt (1052 bytes) at 4312616 ns
Received pkt (1052 bytes) at 4312616 ns
Received pkt (1052 bytes) at 4321032 ns
Received pkt (1052 bytes) at 4321032 ns
Received pkt (1052 bytes) at 4321032 ns
Received pkt (1052 bytes) at 4321032 ns
Received pkt (1052 bytes) at 4415216 ns
...
```

1 simulated second (5 read-outs)
34160 ODR packets
File = 39MB

  - Full setup: 878400 packets, File = 989MB

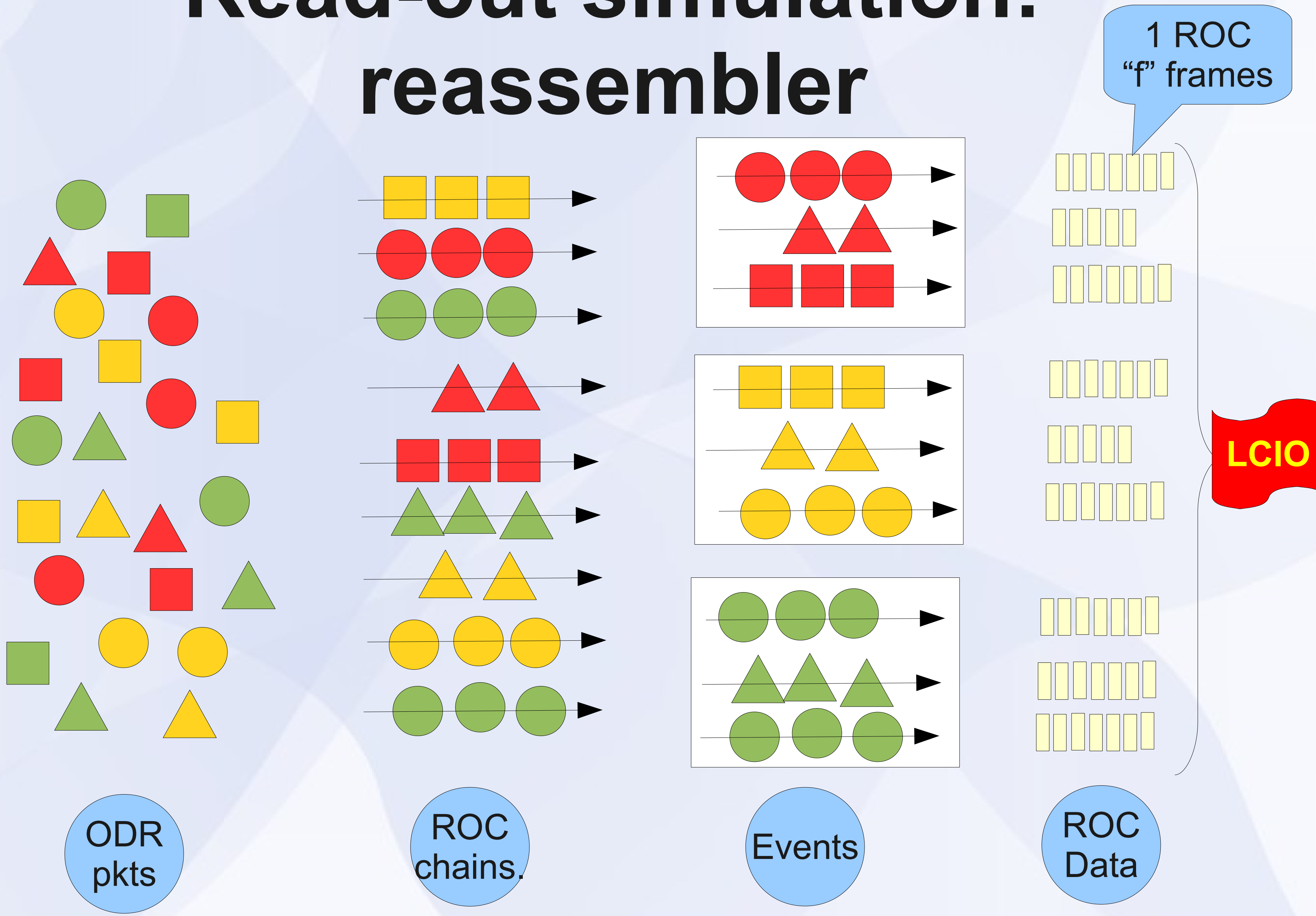
Read-out simulation: reassembler

- Goal: from a stream of un-ordered interleaved packets, reconstruct the original ROC read-out data sequences
- Parse and store the re-assembled data as structured data (LCIO format)
  - For now: HR2 format only, but should be easily generalized (Spiroc, Skiroc, ...)

Read-out simulation: reassembler

- Prototype: 3-level pipeline
  - Identification of the ROC chain data sequences (containing the ROC hit data)
  - Assemble the ROC chain data belonging to the same “event” (ie. Start-readout event) together
  - Parse the ROC data sequences to get the ROC hit data
- Meant to be parallelized (with threads)

Read-out simulation: reassembler



Read-out simulation: reassembler

- LCIO Format:
- 1 Event =
  - Detector Name, Run#, Event#, Timestamp
  - 1 LCCollection per DIF
    - Unique DIF ID (ODR/LDA/LDAlink/DIF Id)
    - 1 LCGenericObject per ROC Data
      - Unique ROC ID (ROC Chain ID/ROC Id)
      - Chip config (type, acqMode)
      - Number of frames
      - Blocks of frame data (for HR2: 20B / frame)

Read-out simulation: reassembler

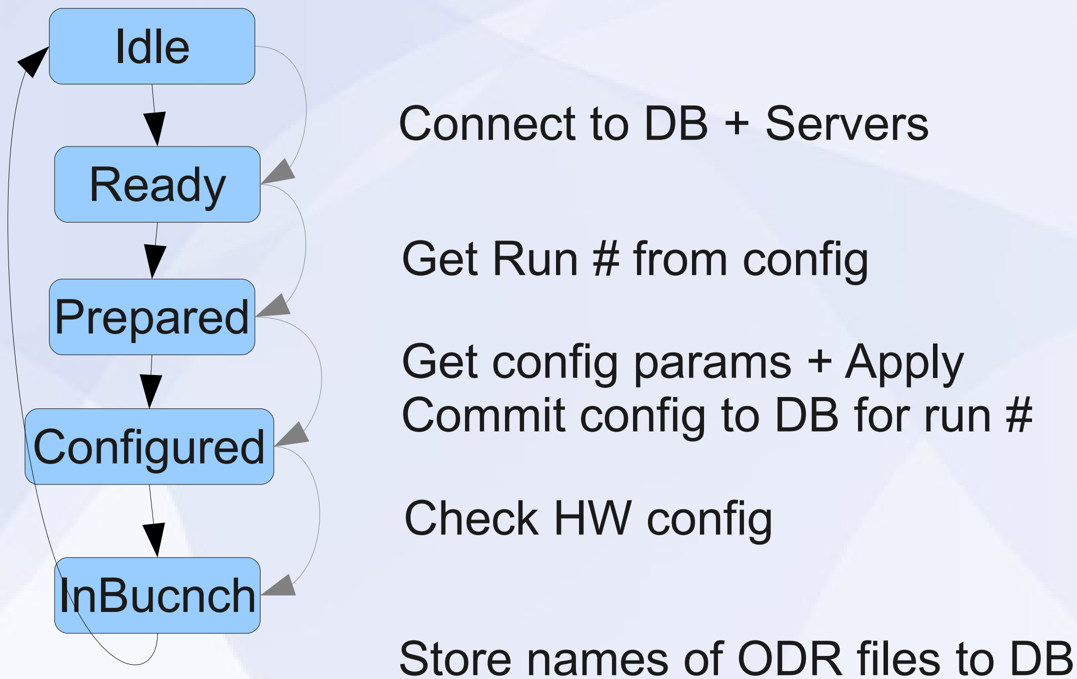
- Calice packet formats described in
  https://svn.in2p3.fr/calice/online-sw/trunk/daq/calice_packets/calice_raw_formats.h?view=markup
- LCIO Format detailed in
  https://svn.in2p3.fr/calice/online-sw/trunk/daq/reassembler/lcio_dump.hpp?view=markup

(Open ?) questions

- Online event-reconstruction + LCIO conversion/storage realistic ?
- Use a common DAQ framework ?
  - ➔ Integrate smoothly with the existing DAQ chains:
    - C. Combaret @IPNL (Xdaq)
    - EUDAQ
    - Si tracking DAQ
    - ALICE-based
    - Other testbeams
    - ....
  - – EUDAQ ? Xdaq ? DOOCS ? Tango ? PVSS ? Custom ?

DAQ SW: FSM & States

- Current implementation: 2-level hierarchical FSM
  - “Super” FSM + Device servers
- State Machines:



Software: read-out simulation

- In svn: online-sw/trunk/daq (https://svn.in2p3.fr/calice/online-sw/trunk/daq/)
  - calice_packets: C++ library to parse/generate ODR/LDA/DCC/DIF (+USB) read-out packets
  - scsim: SystemC simulation to accurately simulate readout on ROC → DIF → DCC → LDA → ODR → PC
  - reassembler: C++ framework to reorder/reassemble ODR packets and save them in LCIO format with a simple structure “run/event/DIF/ROC_chain data”

DAQ Framework: Tango ?

- http://www.tango-controls.org/
- Used by many synchrotron exp. (ESRF, DESY, Soleil…)
- But: the base is NOT synchrotron-specific
- Generic DAQ distributed system (slow-control)
  - Distributed configuration stored in DB
  - Service interaction through an ORB (~ RPC bus)
- Nothing really technically original, but more modern, nicer, more mature, less DESY-centric than Doocs

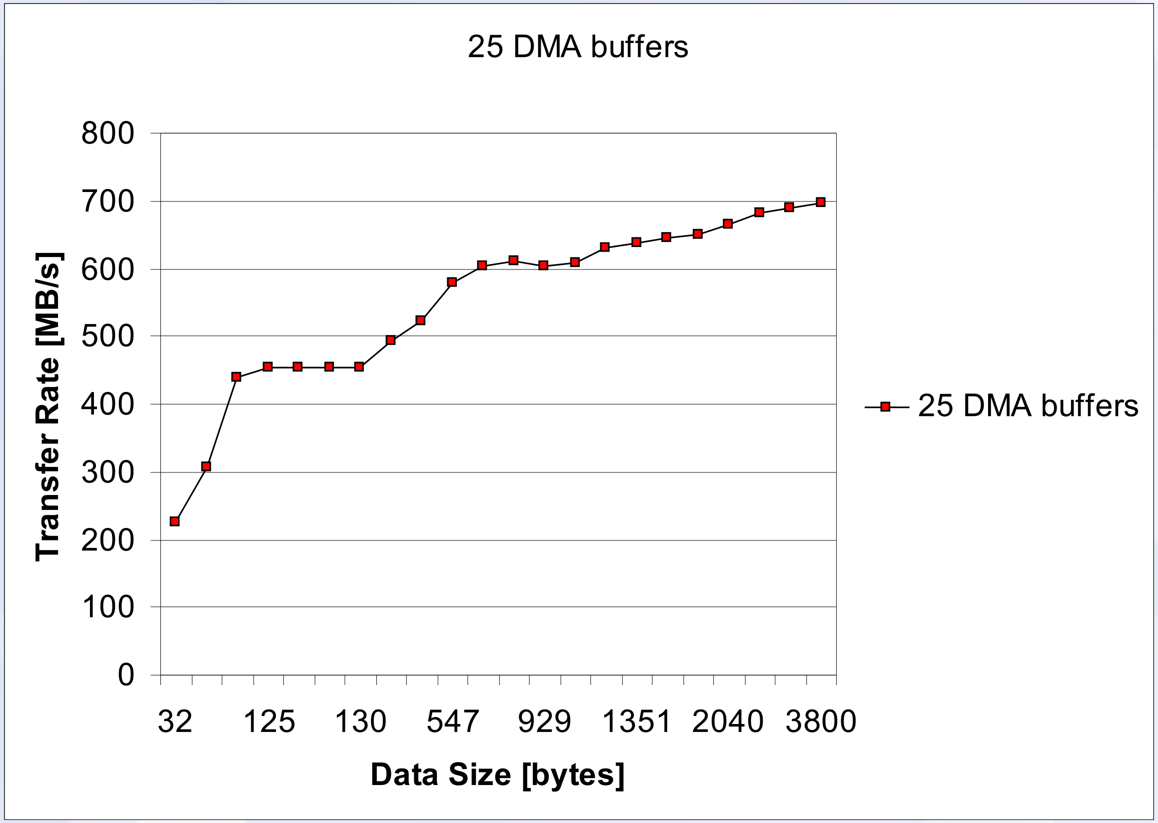
DAQ Framework: Tango ?

- The things that made me enthusiastic:
  - A real 3rd-party blob: we are *users* of the framework, we don't go *inside* the framework
  - Simple generic & composable message data types between services
  - Reasonably language-agnostic: C++, java, python
  - Can use jddd (Doocs GUI builder)
  - Management can be achieved via GUIs
  - A fast "device server design" approach (GUI code-generator)

DAQ Framework: Tango ?

- Tango or not Tango ?
  - Generated only 3 device servers (C++, java, python) and played with them
  - Had a positive experience with the Tango mailing-list
- To be continued ?...

DAQ Prototype: Performances



Evaluation by V. Bartsch/T. Wu/UCL/Manchester