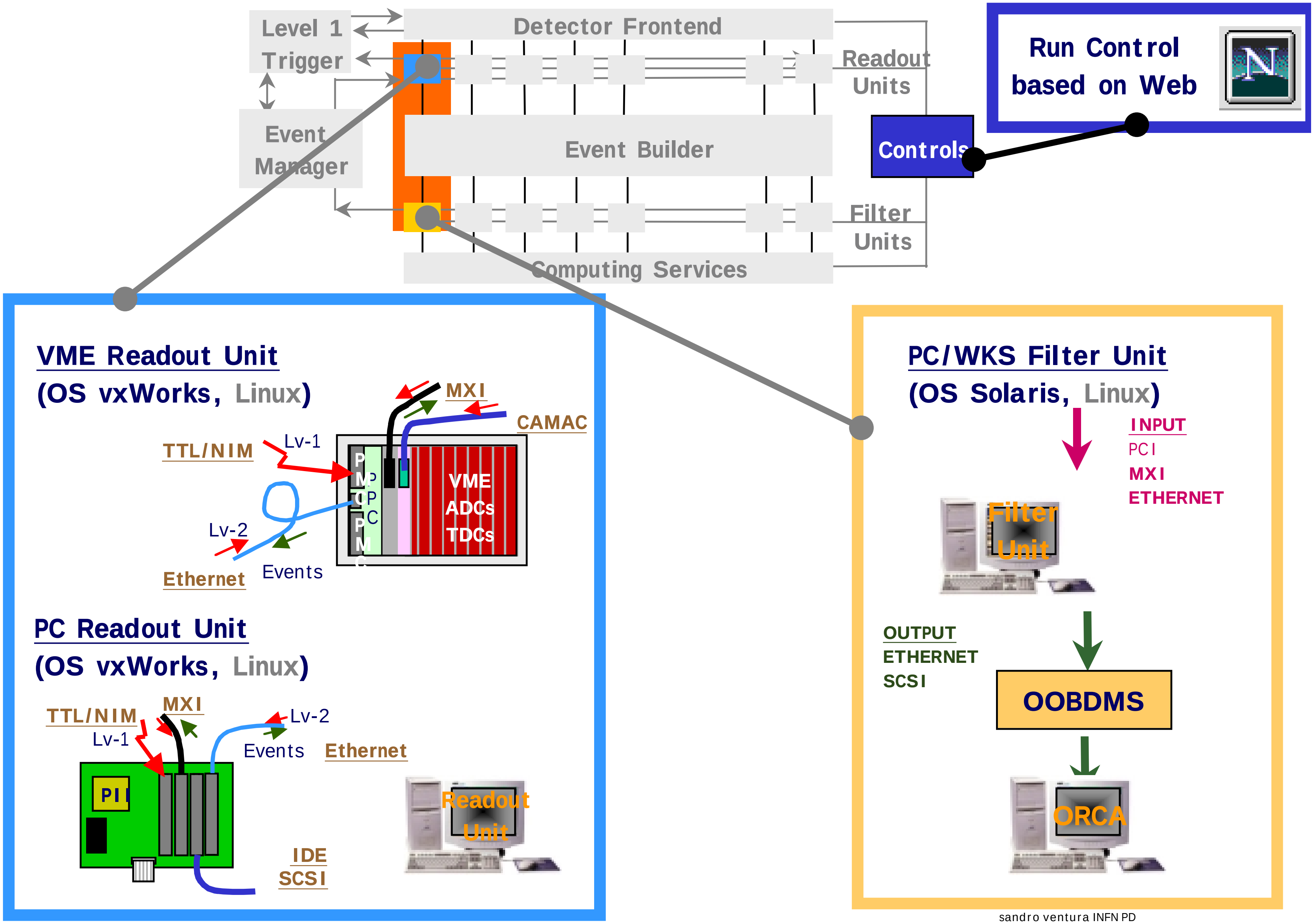
Experience with H2 testbeam daq

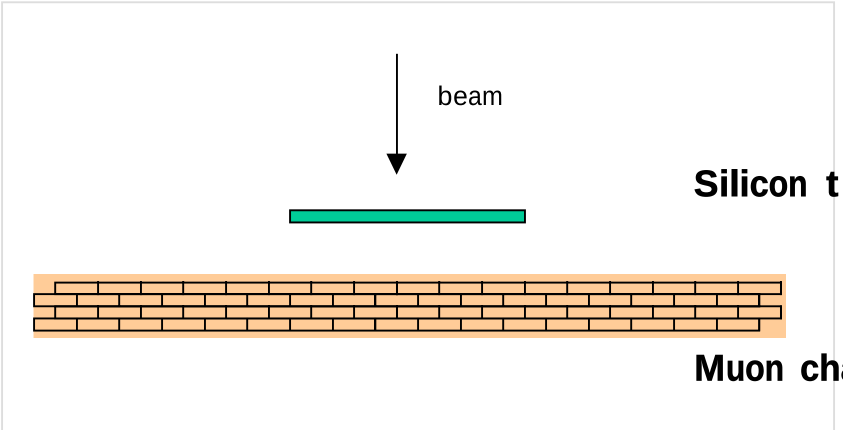
sandro ventura
INFN padova

Small daq system based on daq column



sandro ventura INFN PD

H2 july '99 testbeam setup



Silicon telescope

to be read through dual port ram snooping (old H2 daq system)

Muon chamber

64 TDC channels

1 PU for BTI output recording

Rates ≈ 800 trig/spill - 400 hz

8 ktrigs/spill - 4 khz (no silicon)

Typ. sizes ≈ 50 kB/spill

80 kB/spill (no silicon)

Setup:

a "parallel" daq system based on daq column components

Goals:

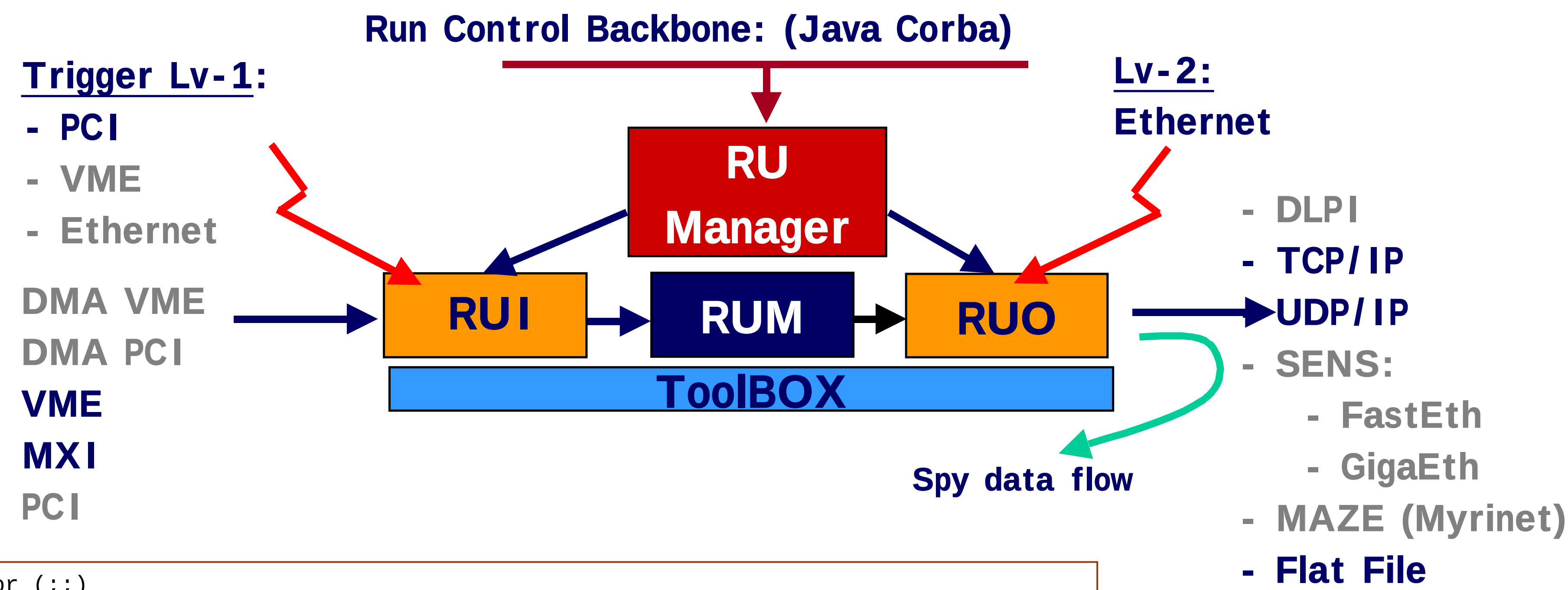
Provide real data to analysis through ORCA

Verify needed resources to customize the generic DAQ column

Validate building protocols

Verify portability and code reusability

Daq software architecture



generic daq loop

```
 for (;;)
{
        try {
            // Waiting trigger
              *ruiTrgStream >> setl(sizeof(trigger)) >> (char*)&TBtrg;
            //Read Event
              *ruiInputStream >> setl(1) >> (char *)evt_data;
            //Write to RUM memory
              rumStream_->open(&event,vxios::write);
              *rumStream_ << setl(evt_data[0]*sizeof(int)) << (char *)evt_data;
              rumStream_->close();
            }

 }
```

RU measured rates:

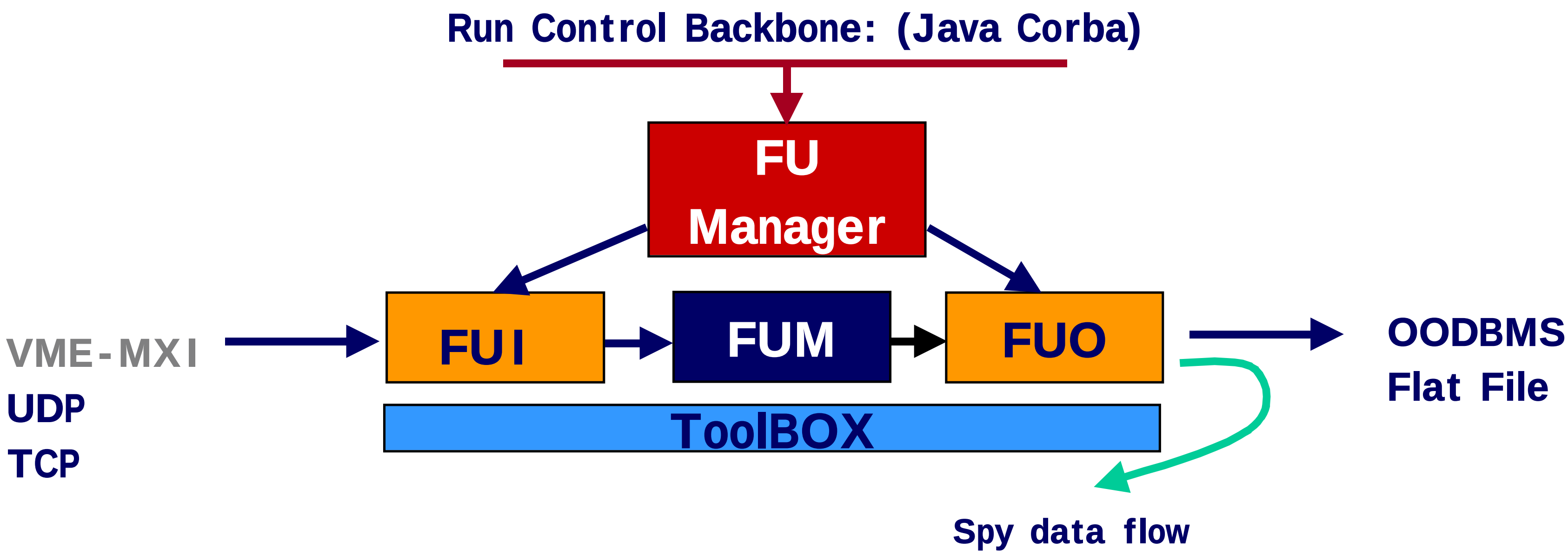
RUI ≈ 100 khz

RUI+RUO

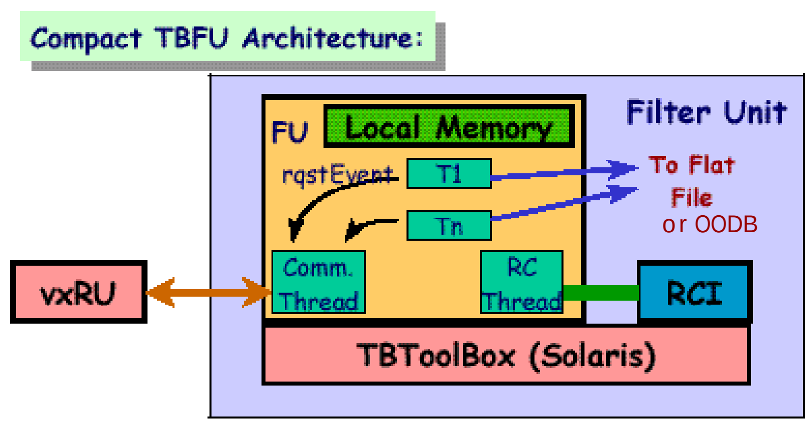
≈ 9 khz (256 B/ev)

≈ 6 khz (4 kB/ev)

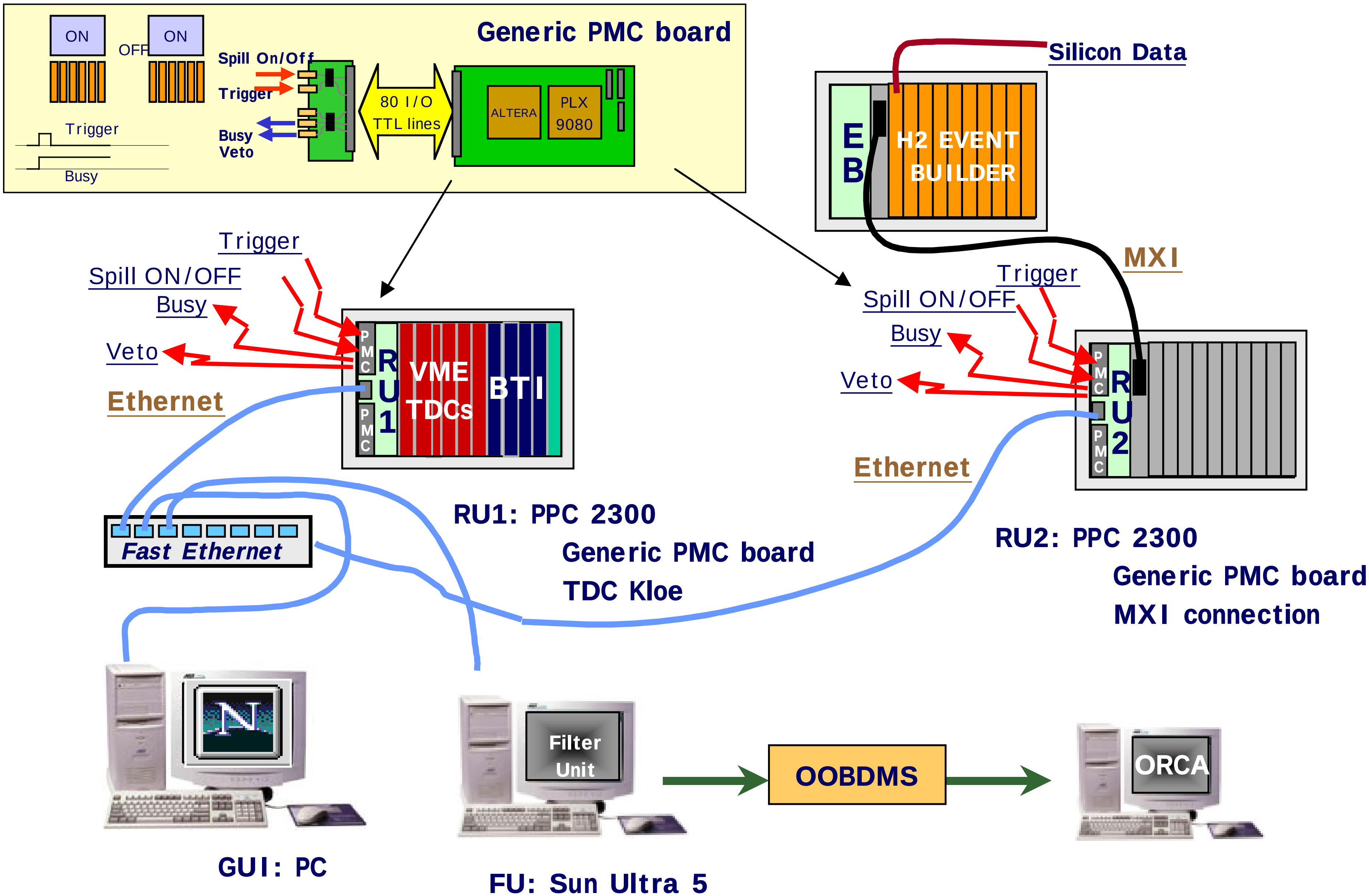
Daq software architecture



FU measured rates:
RU+CFU
≈ 1 khz (256 B/ev)
≈ 700 hz (4 kB/ev)



System Components: hardware setup



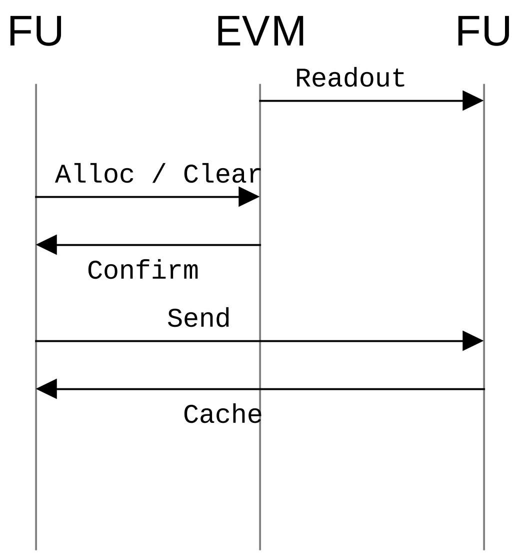
sandro ventura INFN PD

System Components : EVM

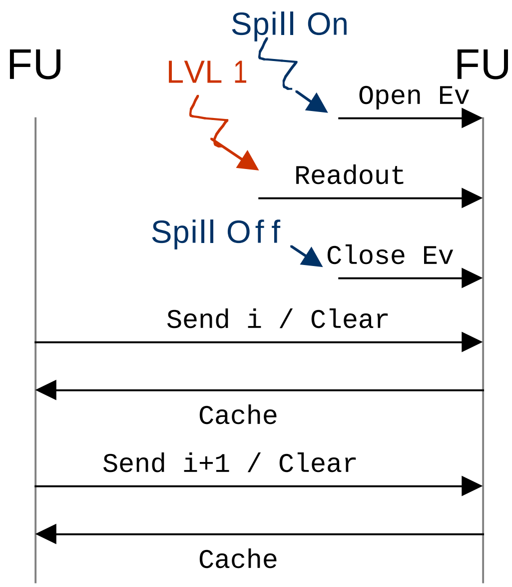
- No software component.
- Hardcoded logic for the synchronization (BUSY's and VETO's).
- Sequential super event numbering drives requests.

Due to silicon data snooping, data were collected as super events (1 per spill): LVL-1 triggers were appended up to the end of the spill

Full Building Protocol

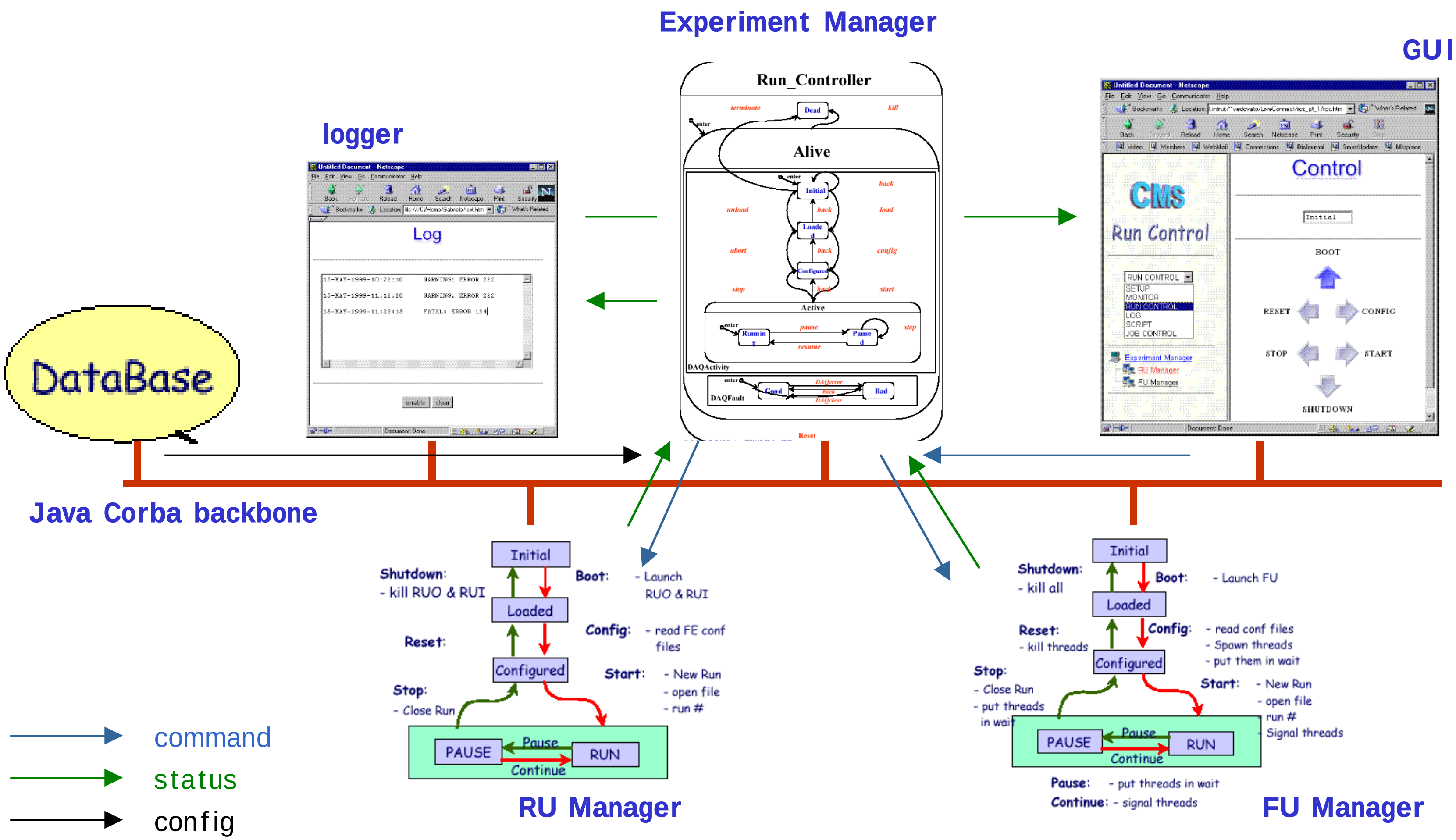


TestBeam Simplified Protocol



Effective super event rate 1/14.2 s

System Components : Run Control



Working as a spy daq, the RCS actually didn’t provide any front end configuration, nor run setup logging.

System evaluation

Performances

Total throughput wasn't a big issue (1-100 kB/s) due to spill cycle.

Level-1 trigger handling within requirements (> 500 hz)

Uptime

60% of the two weeks run (mostly on single RU configuration). Half of the runs only on flat file storage.

Required Manpower

	this setup	to a new front end
customization	3 man months	≈ 10 days
integration	2 man months	-
final setup debugging	3 man weeks	probably same

Major inconveniences

Bugs: found quite a few during integration and running, both on inherited code and on custom code. Systematic deadlock on RUI/RUO sync hang RU. Memory leaks on the FU side. Software exceptions handling problems (compiler?).

System evaluation

Major inconveniences (continued)

Inadequate RU model: the RU classes had to be modified to allow use of specialized RUI's, with different trigger handling.

Online Event Display: the lack of running tools to spy OODB data flow resulted in DB being filled without any check. Unacceptable condition. Although raw data spies had been added, at least a rough event display (whether OO or not) to qualify data will be necessary during future runs.

Database Population: following the previous lack, problems with raw data encoding to DB objects gave much more troubles than they should have.

Run Control: Run Control System unable to handle asynchronous error conditions. GUI had several misbehaviour (and was too slow). Switched to alphanumeric user interface. ORB interoperability problems forced the move of RU manager away from the RU cpu.

System evaluation

Future steps

RU/BU API: a major revision of the whole toolbox went through, resulting on a new software model, based on remote method invocation, aimed to a higher flexibility. Testing is now being done, and possibly a new integration will proceed on a next small daq system.

Run Control: the tracked bugs have been worked around. While the architecture isn't going to be modified, a new release of the RCS provides a cleaner interface between components.

Event Display: while no display can be generic enough to cover every setup, some basic general purpose tool (e.g. histo server) could be embedded on the builder.

System evaluation

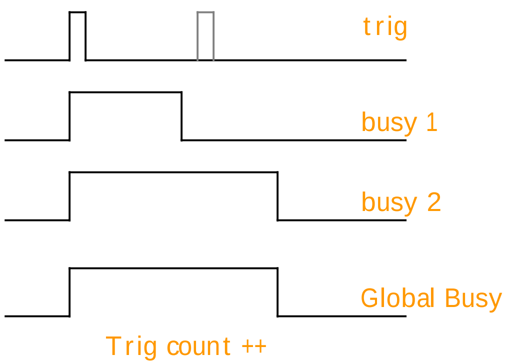
Future steps (continued)

Database support: a local, lightweight database will be integrated on the system to address all the issues related to system partitionning, run configuration and bookeeping. Among various products we are evaluating **minisql** (public domain), **mysql** (linux 6.1 distr.), **Jdatastore** (Borland), last two being JDBC compliant.

Multi front-end integration

The lack of pipeline in present testbeamfront-end involves a revision of the EVM-RU-BU protocol to insure proper trigger synchronization in a multi RU's setup (e.g. integration of silicon data required spill sync).

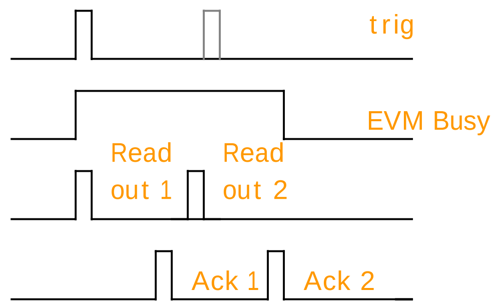
HW oriented sync:



A RU can loose trigs due to time alignment problems.

EVM trig ID broadcast:

Acknowledged Readout Invocation:



Every Readout (or broadcast) needs to be acknowledged.

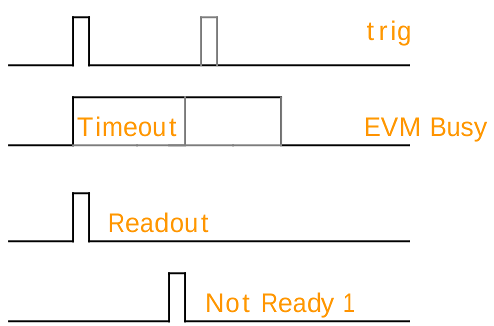
Deadtime sums up.

Band limited when trig rate increases due to n-ack's

Multi front-end integration

EVM trig ID broadcast:

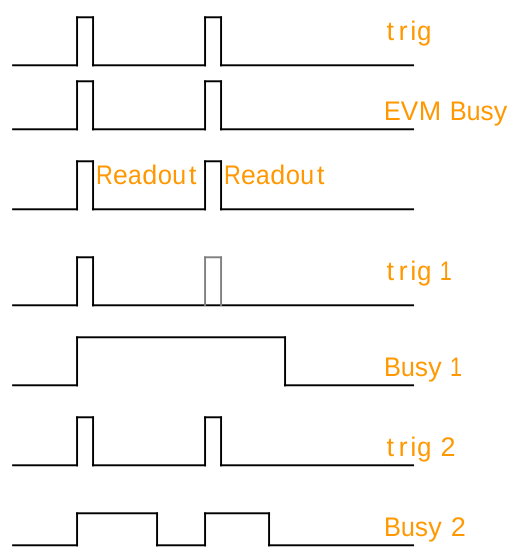
Timed Out Busy:



Only unaccepted trigs are signaled to EVM. If none after timeout busy is cleared.

Trig Rate limited.

Independent RU's:



Every trig ID is broadcasted.

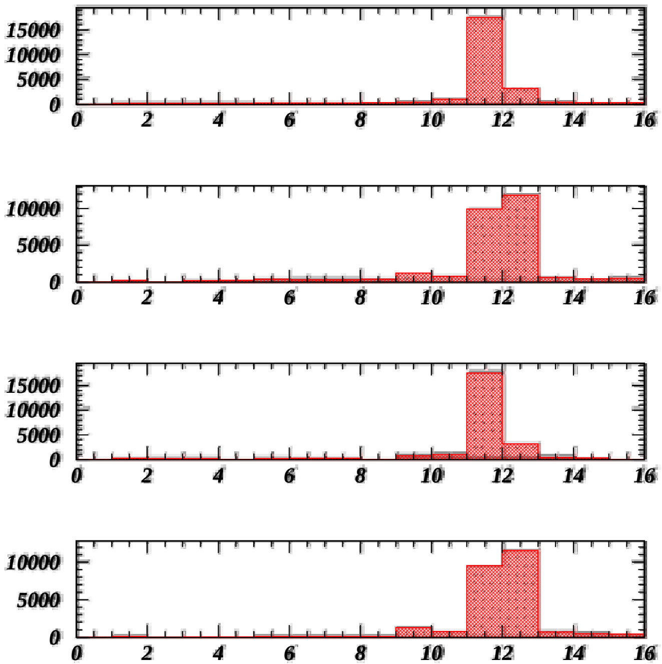
RU's can accept or reject trig.

(empty trig entries might be pushed on DPM for proper merging).

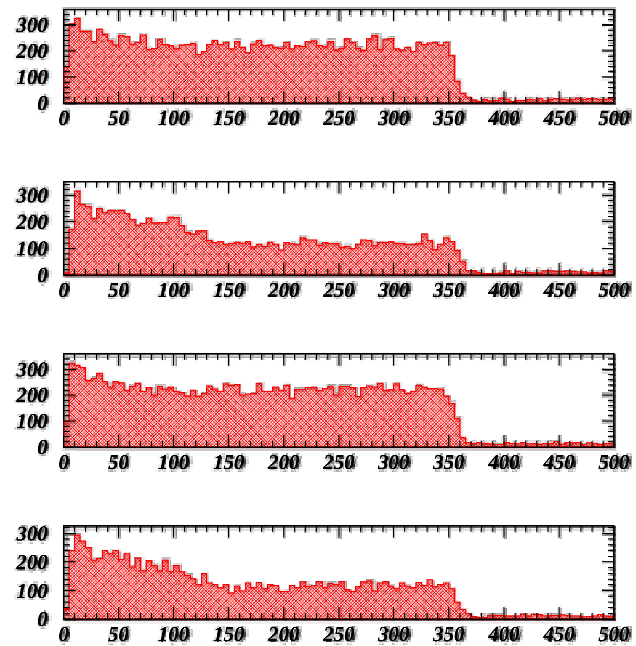
First data analysis through OODB

by Annalina Vitelli andClaudio Grandi

Cell Occupancy



Drift Time Boxes



Chamber Resolution 200 μm

Efficiency 90%