DAQ Demonstrators

Overview on available setups
Activities on various setups

FEDBuilder Status

TDRDemonstrator performance analysis

Demonstrators Overview

- Old farm (cannibalised)
  - approx 50 PCs, 1 processor, PIII (800 and 1000 MHz)
  - Networking:
    - partly Myrinet Lanai9
    - Gigabit Ethernet
  - Plan: make a small farm with 32 nodes and 2 CPUs per ndoe

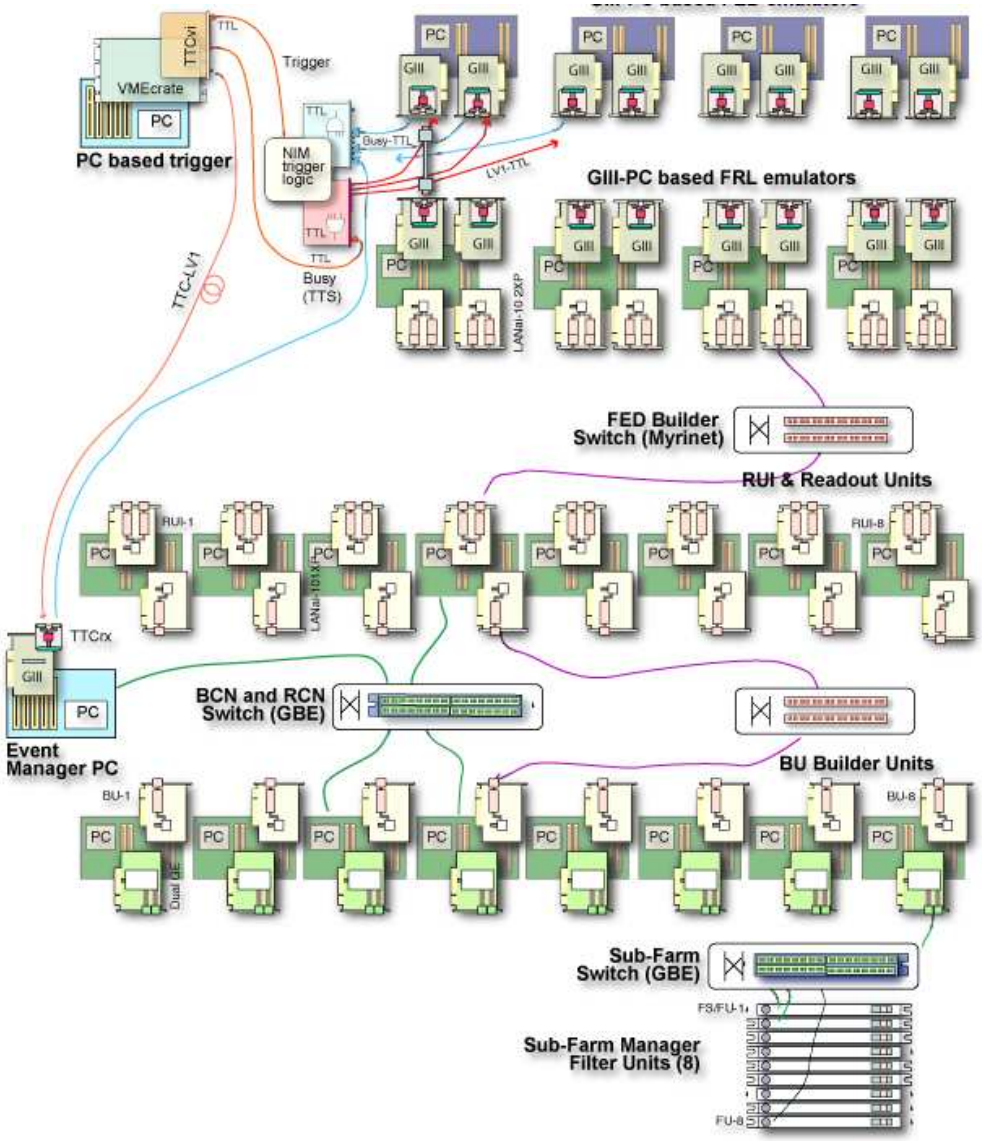
contd.: Demonstrators

- TDRDemonstrator
  - 8x8 FEDBuilder on Myrinet (Lanai9)
  - 8x8 RU Builder
    - Myrinet (Lanai 9)
    - Gigabit Ethernet
  - Filter Farm
    - 8 FUs on Gigabit Ethernet
    - 1 subfarm controler
  - 14 FED-emulators based on GIII
    - with SLINK 64
  - 8 FRL emulators (based on GIII)

contd.: TDRDemonstrator

- 1 EVM PC
- some FED-emulator control PCs
- RU/BU/EVM (and FRL control) PCs:
  - 2 XEON 2.4 GHz CPUs (hyperthreaded: 2 logical processors per CPU)
  - 512 MB RAM
  - 4 PCI buses, 6 PCI slots (64bits 66MHz)
- FU and Subfarm-controler PCs
  - 2 XEON 2.4 GHz CPUs, 2 GB RAM

TDRDemonstrator



Trigger modification in TDRDemonstrator

- Features:
  - No use of TTCvi/TTCrx anymore
  - Use GIII to generate triggers of programmable rate
  - GIII implements busy logic
  - Transfer of data is done with FEDkit
  - Programmable trigger frequency (sub-Hz to MHz)
  - Data Record freely programmable
- Missing:
  - No transfer over dedicated trigger-FRL
  - Software for input is not yet equivalent to RUI

Will be used until the Trigger emulator is available (next week?)

contd: Demonstrators

- 8 extra nodes
  - same as PCs of TDRDemonstrator (RU/BU/EVM)
  - 4 nodes with Myrinet Lanai 9 (optical)
  - Gigabit Ethernet on all nodes
  - managed in TDRDemo cluster
- DCS test bench
  - ca 10 nodes
  - for DCS software evaluation (PVSS, performance, scaling, ...)

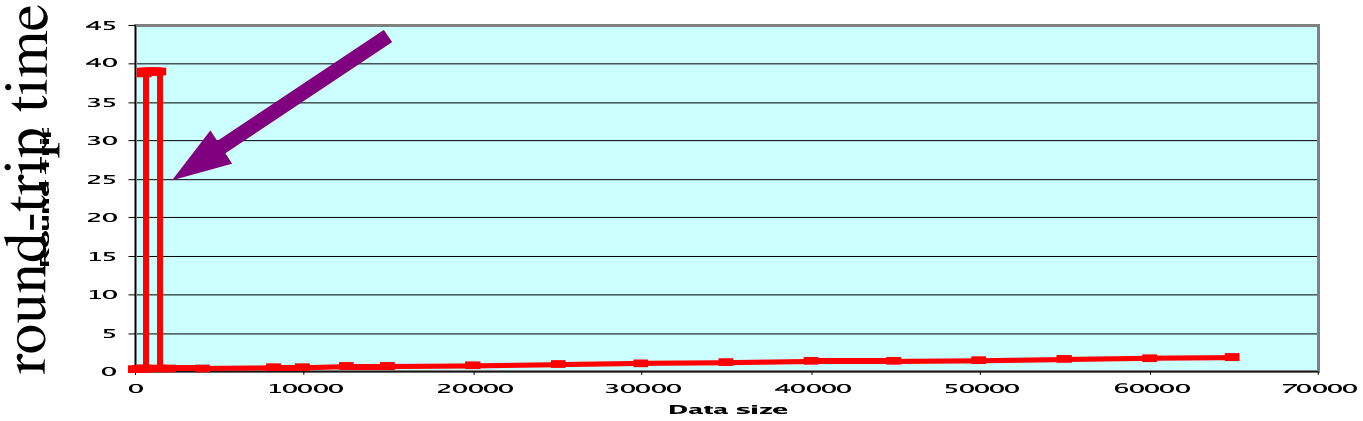
Activities

- Old Farm
  - RCMS integration and development (Michele and Alexander)
  - TCP/IP performance measurements (Samim):

roundtrip measurements: new kernel behaves different than old kernel:

large latency at small fragment sizes

Conclusion: new kernel implements CORRECTLY Nagle algorithm
(the old one not ... although it was better for us)



fragment size

Activities

- Still not understood: memory effect for streaming measurements:
  The result of the measured throughput at a given fragment size depends on the the fragment-sizes which have been streemed through the sockets previously.
- UDT evaluation (Michi)
  - UDT is UDP with a protocol for reliability on top.
  - had been designed for WAN networks (large packet sizes, long latencies)
  - seems to be unadequate for us (max. 46 MB/s at ca 12kB fragment size after tweaking of parameters)
- Event Builder development

- Extra nodes
  - Multi-rail measurements (Marco)
  - Folded EVB (Marco)

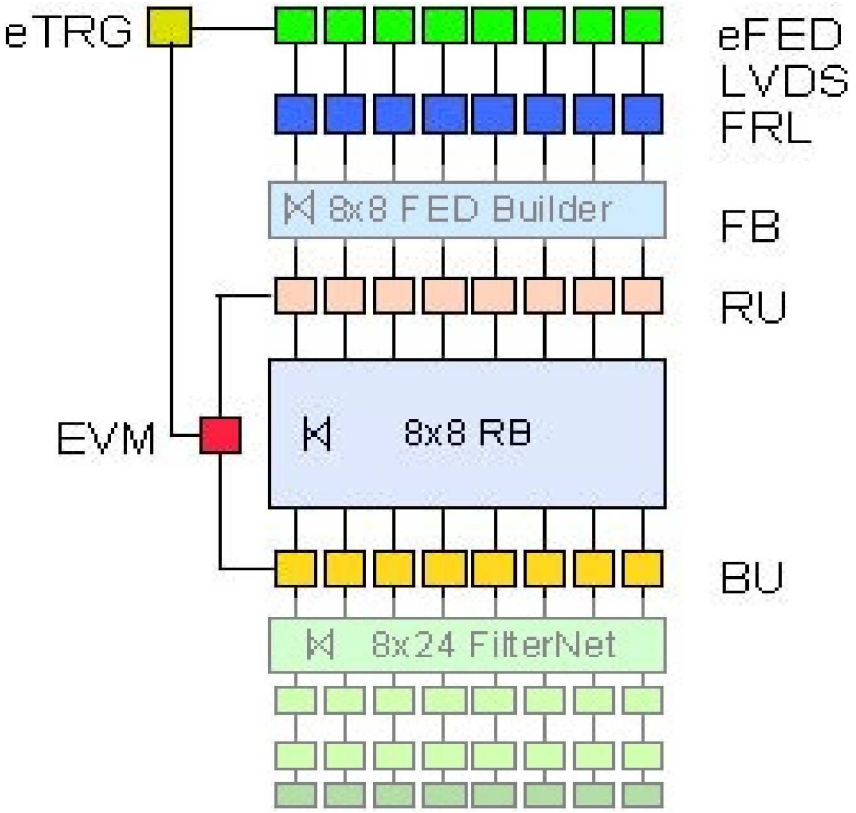
contd: Activities

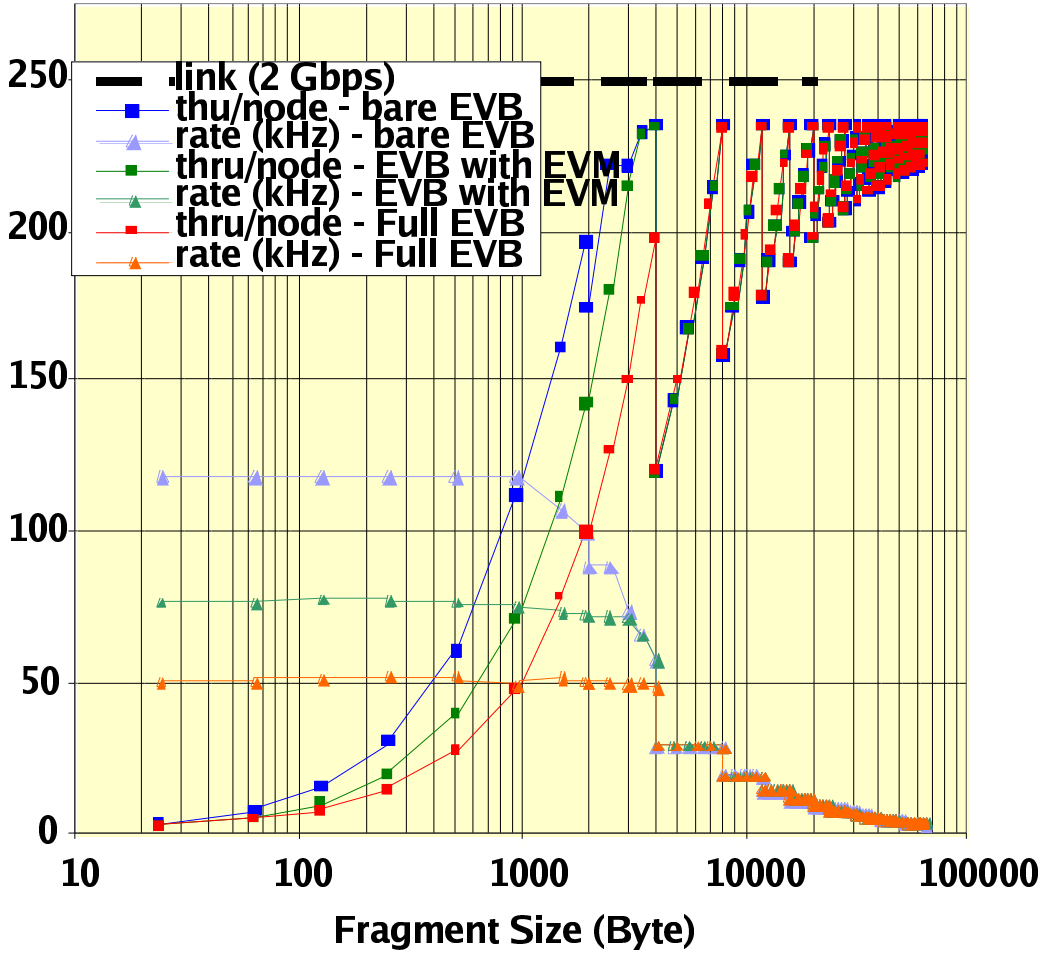
- Filter Farm
  - Development of Filter software framework (“online meets offline”) (Emilio)
- TDRDemonstrator
  - FED-Builder development
  - System integration (“Annual Review” - results)
  - Detailed performance investigation

FED Builder development

- First version of firmware available
- Integrated in TDRDemo software
  - not yet stable though
- Functionalities:
  - Super-fragment building in NIC (FBO)
  - credit based (to avoid NIC memory overflow in FBO)
  - allows for different block sizes in FRL and EVB
- Current restriction
  - EVB blocks must be larger/equal than max super-fragment size

TDRDemonstrator: Annual Review Results

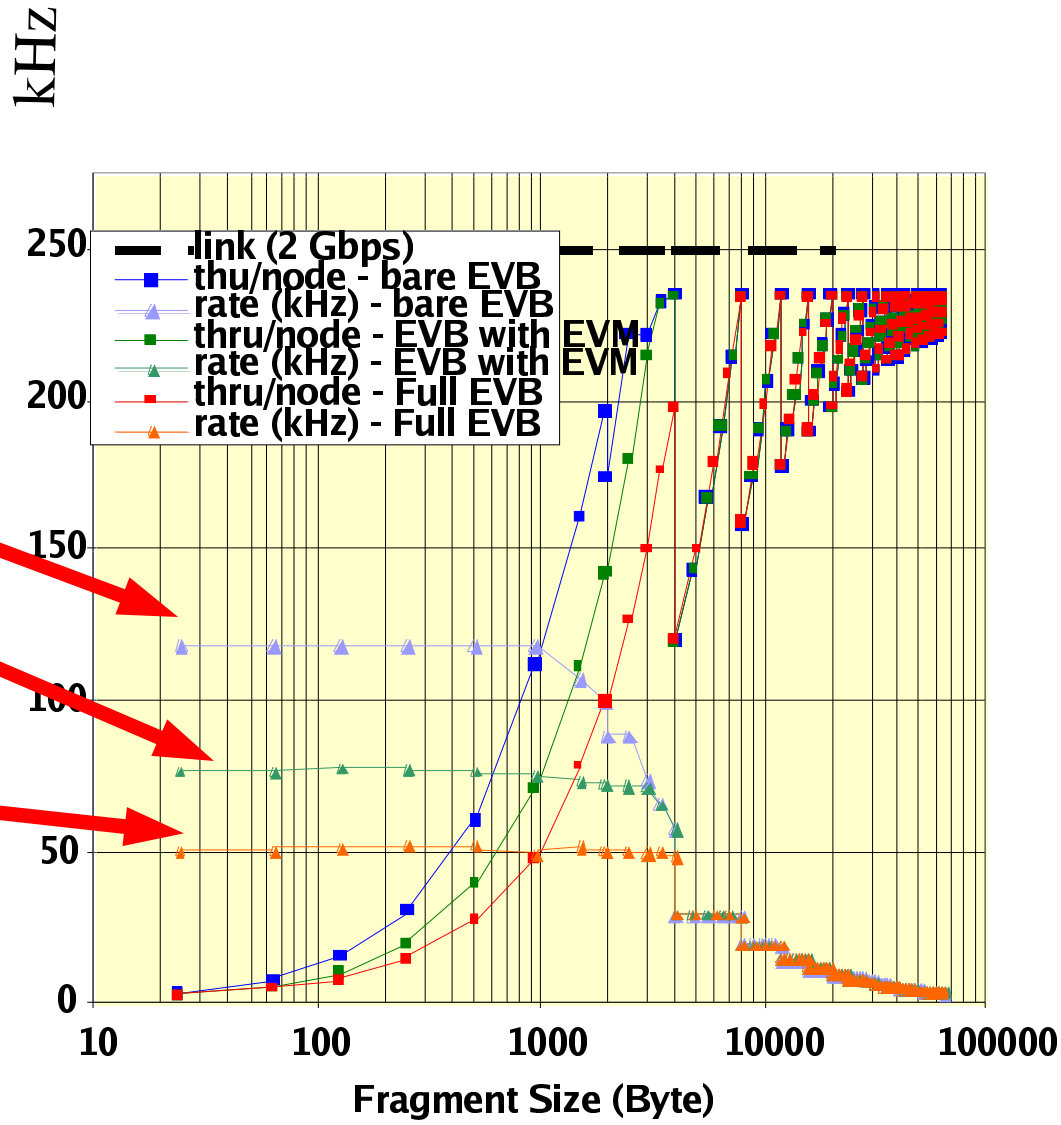




Performance for small fragments

- Expectation
  - BS cycle : 16.5 us
  - max. 2 packets / barrel
  - **121 kHz max**
- Measured:
  - **120 kHz** for bare EVB (no EVM)
  - **77 kHz** for EVB with EVM (overhead of Control Msgs, event generation)
  - **50 kHz** for full EVB (EVM and data input: RUI)

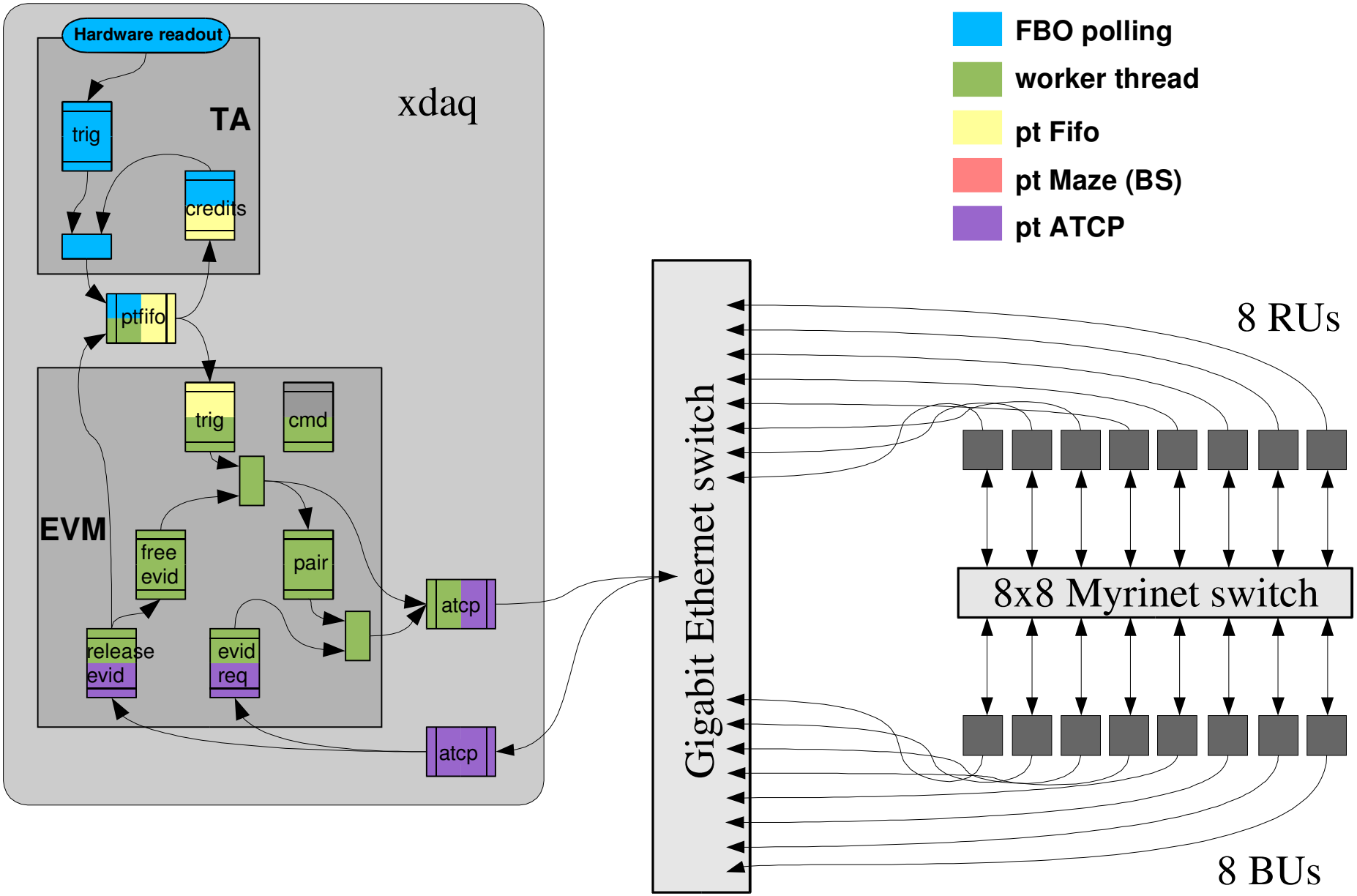
full EVB: large variation when observed for many seconds



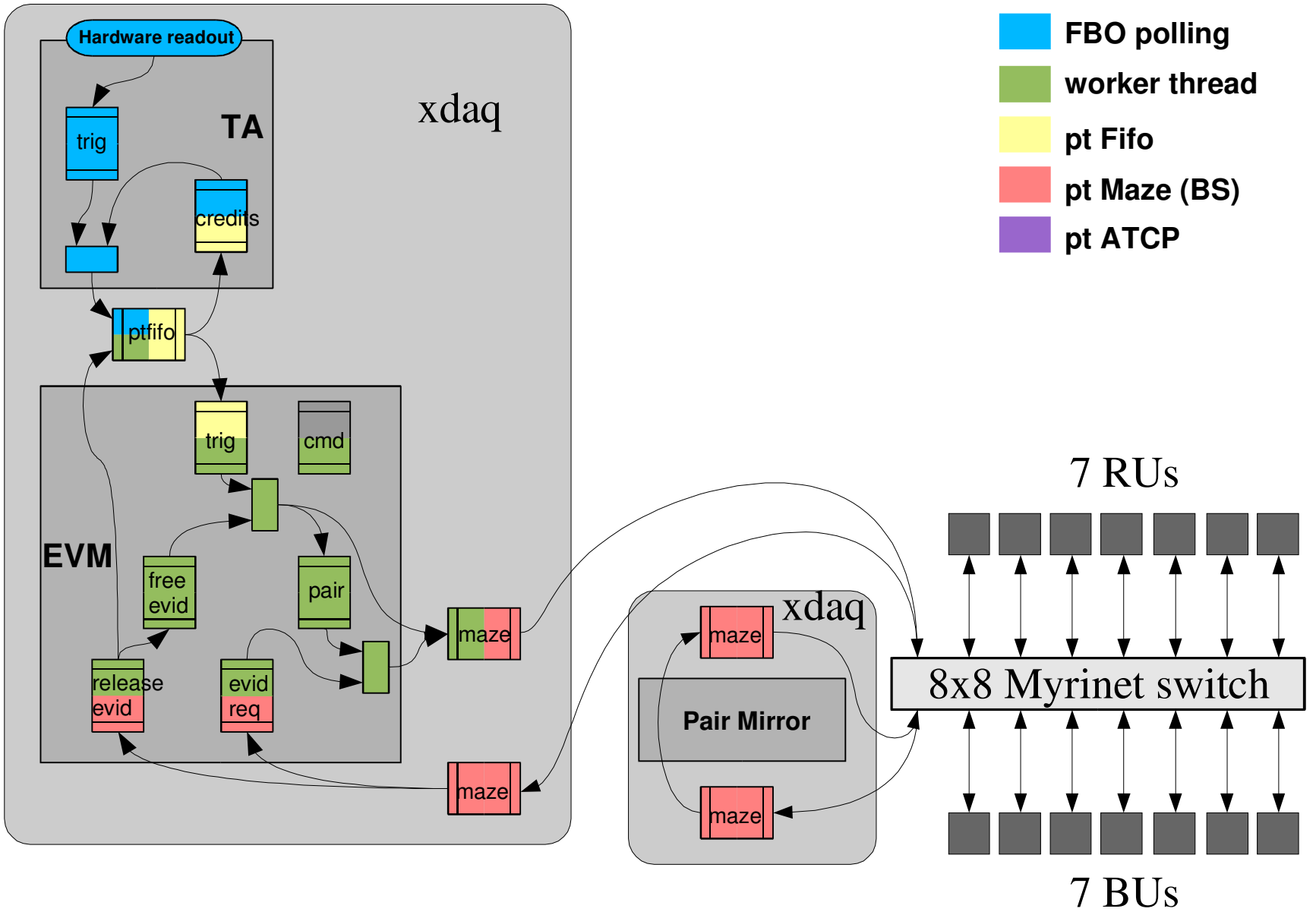
Investigation: small frag performance (full)

- one bugfix : rate from 50 kHz to 70 kHz
  - CPU was consumed by preparing a string for every fragment in RUI
- Where is the bottleneck? Investigations on a column:
  - EVB without data (only control messages) : 170 – 260 kHz
  - EVM is NOT the bottleneck
  - BU is doing almost nothing in full EVM (no Filter yet)
  - RU is the bottleneck in the system
- To understand variations and locate the bottleneck:
  - look at model of the EventBuilder : EVM RU BU
    - number of active threads
    - FIFOs (decouple threads, buffer data-descriptors)
  - look at architecture of XEON CPU (hyperthreading)

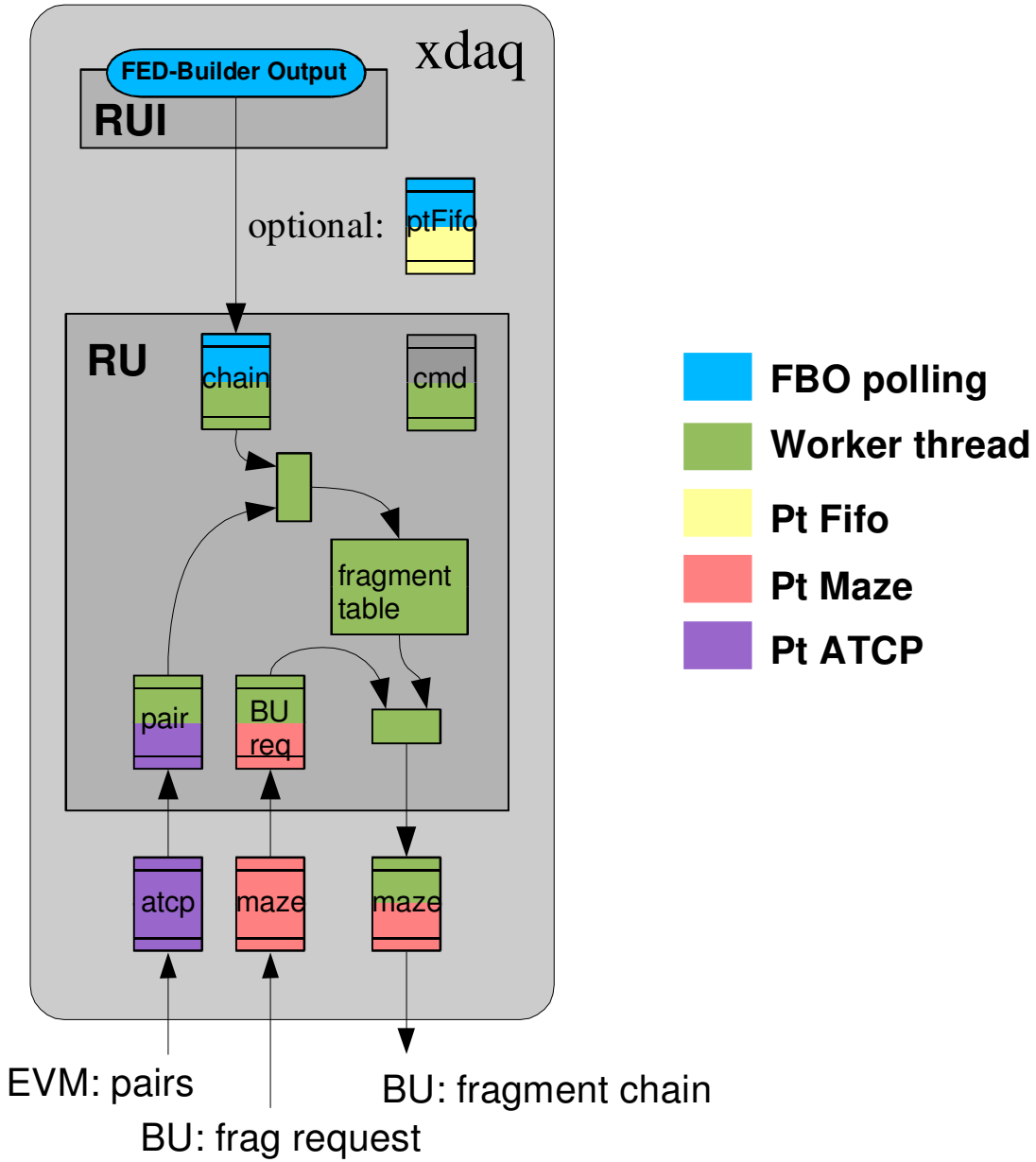
EVM model (hybrid)



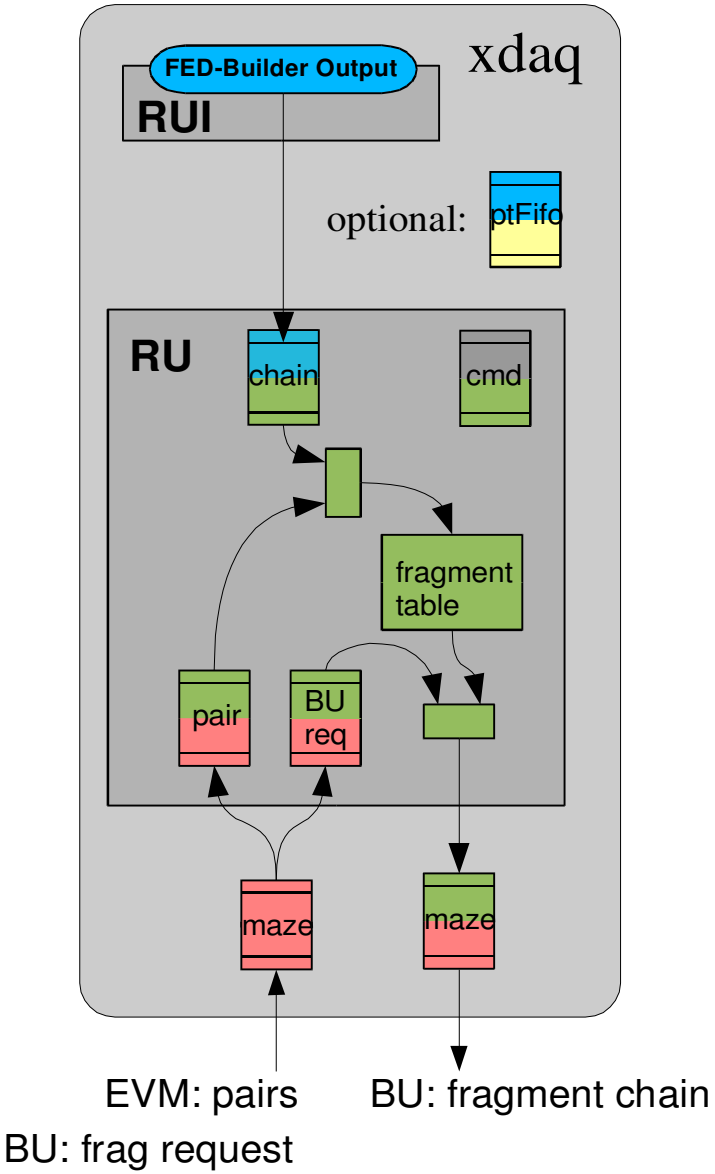
EVM model (Maze only)



RU model (hybrid)



RU model (Maze only)



FBO polling

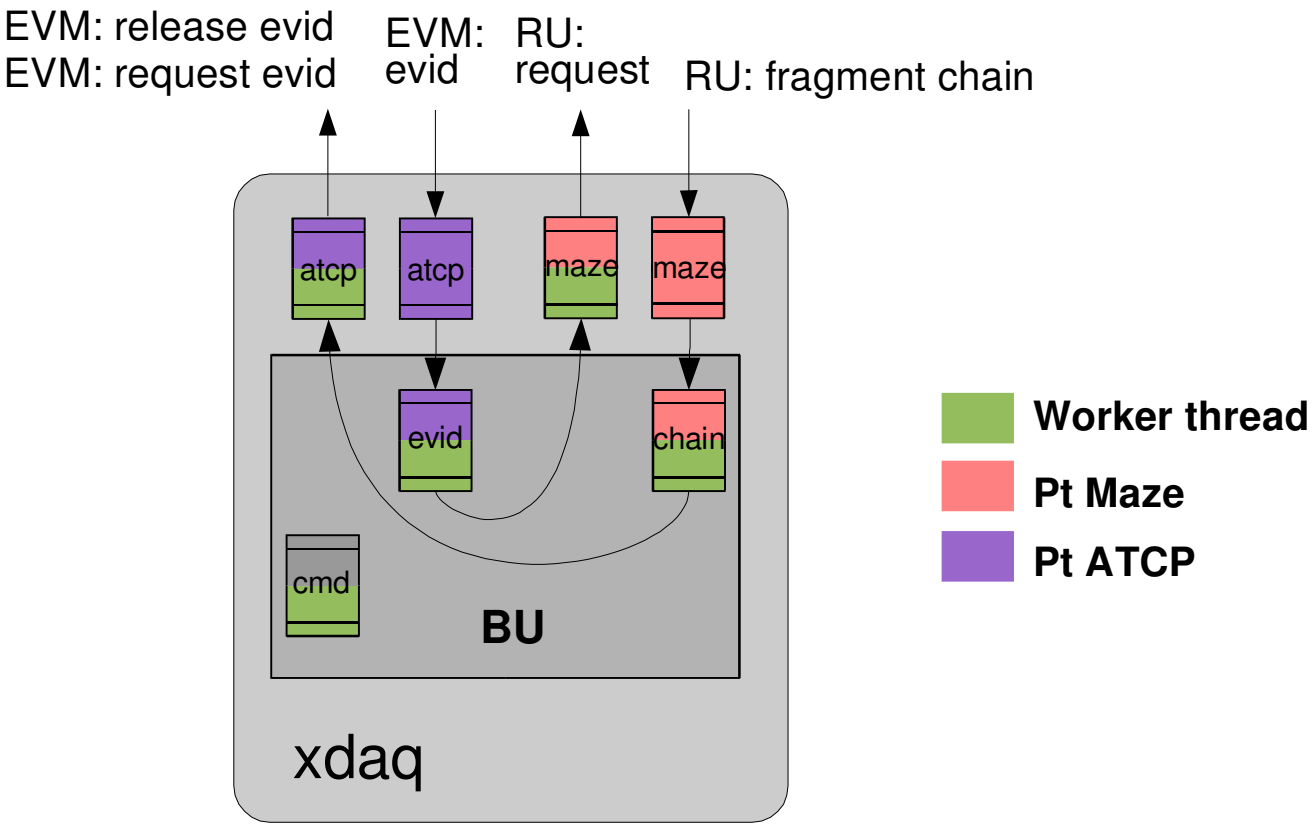
Worker thread

Pt Fifo

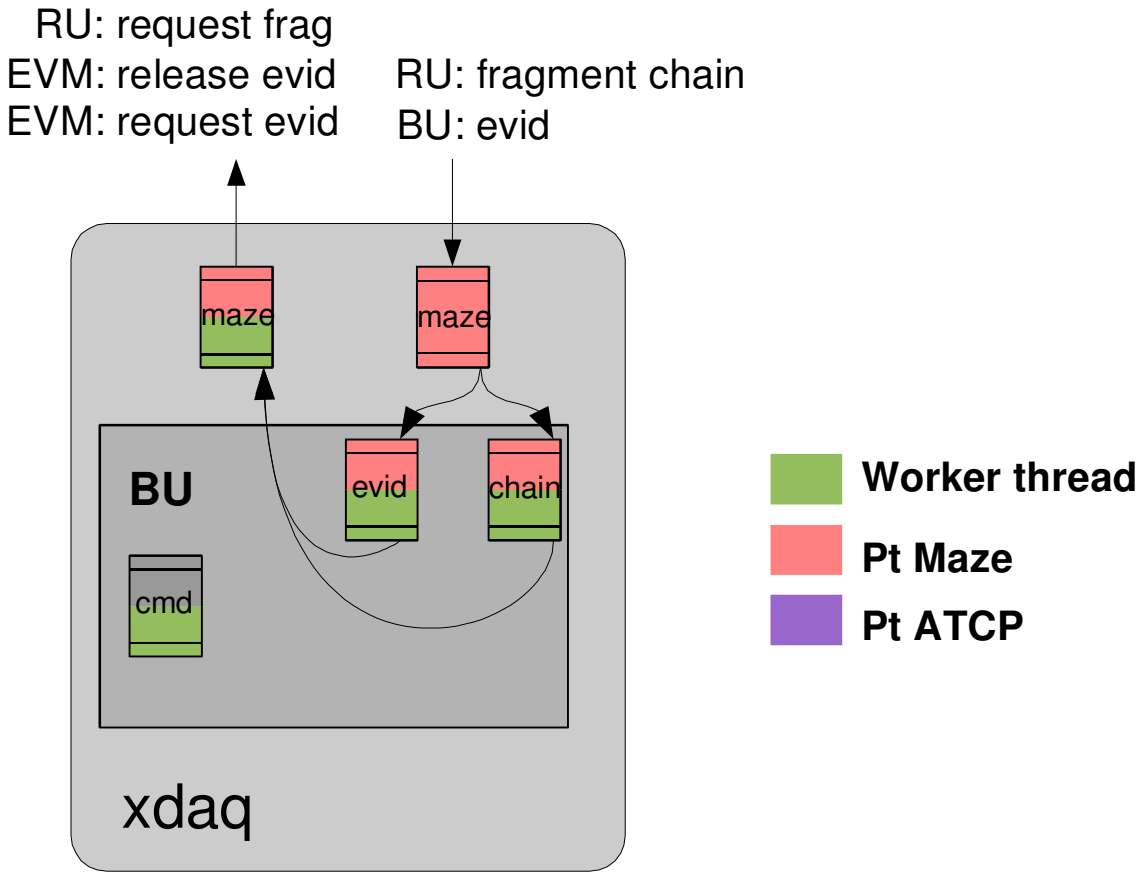
Pt Maze

Pt ATCP

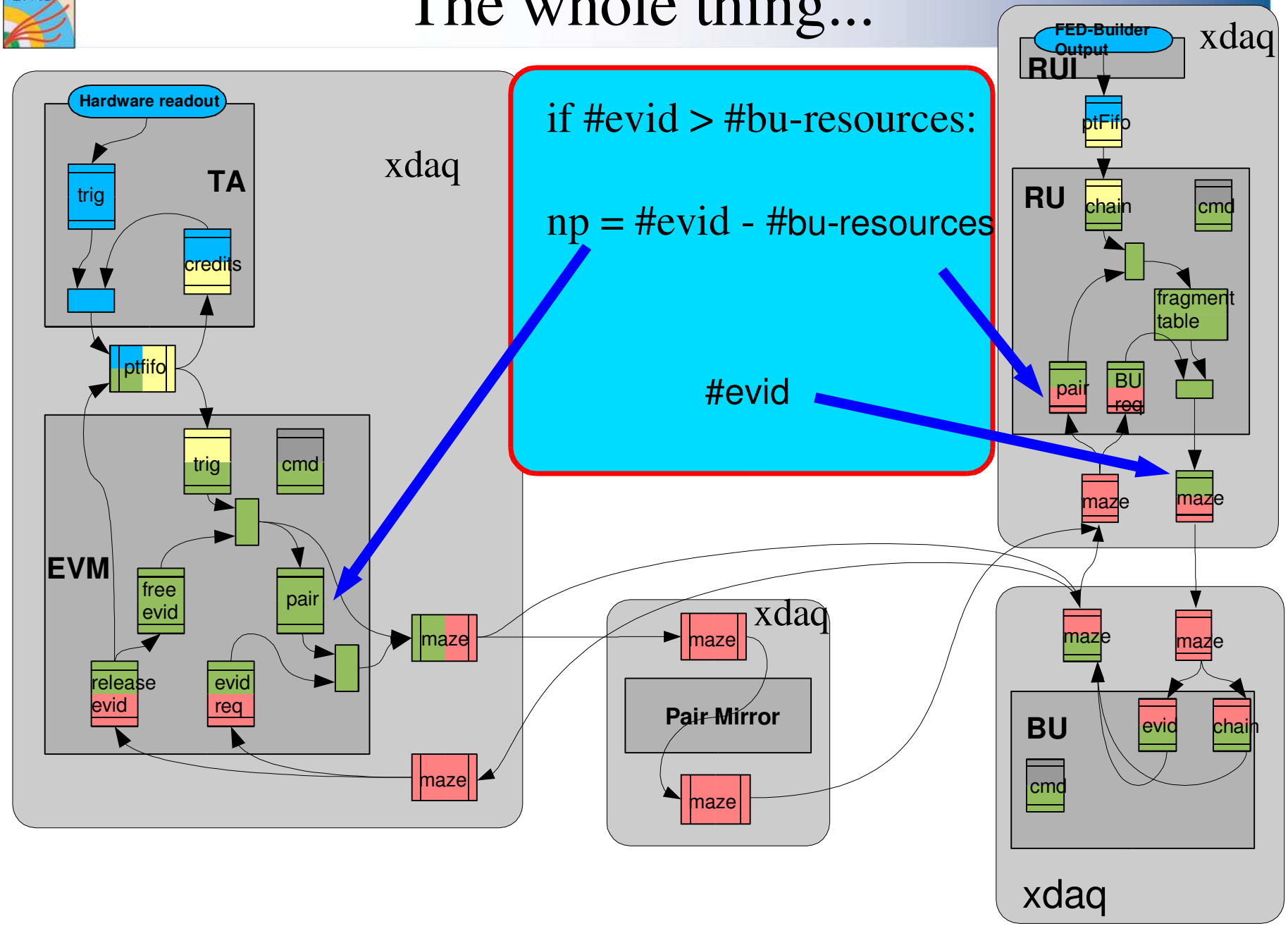
BU (hybrid)


EVM: release evid
EVM: request evid
EVM: evid
RU: request
RU: fragment chain
atcp
atcp
maze
maze
evid
chain
cmd
BU
xdaq
Worker thread
Pt Maze
Pt ATCP

BU (Maze only)


RU: request frag
EVM: release evid
EVM: request evid
RU: fragment chain
BU: evid
maze
maze
BU
evid
chain
cmd
xdaq
Worker thread
Pt Maze
Pt ATCP

The whole thing...



Conclusion: fragment rate limit

- RUI design
  - If ptFifo is used instead of “directDispatch” :
    - rate drops from 70kHz to 45khz
    - Linux puts the ptFifo thread on same logical processor than RuiRead
- Bottleneck of fragment rate:
  - For small fragments the fragments are “sitting” in the Maze output queue of the RU (ready to be sent).
  - Maze does not succeed to always fill a packet with 2 fragments. (the measured rate is less than 121 kHz)
  - Could be due to higher CPU load in full EVB.
- Nice feature would be to have parameters in the peer transports which show the occupancy of the internal buffers.

Fluctuation of fragment rate

- In RU there are three active (100%) threads:
  - RUI : FED builder reader thread (polling)
  - RU: Worker thread
  - RU: ptMaze
- Linux changes assignments of threads to CPUs
  - can be observed via “top”
  - each RU contains
    - 2 physical XEON processors
    - shown as 4 logical processors

Hyperthreading in the XEON

- XEON contains 2 logical processors
- Shared resources
  - execution engine
  - bus interface
  - all caches (including data TLBs (64 entries). Translation Look-aside Buffers cache the address-translation of the MMU for a given memory page: memory page descriptors are located in RAM)
  - MTRRs (registers which contain configuration parameters for the memory pages. They anyway must be the same for all processors in a system.)
- Duplicated resources
  - all (relevant) processor registers
  - APIC (Advanced Programmable Interrupt Controller)
  - instruction TLBs (128 entries)

Conclusion: performance fluctuations

Observations:

- Linux sometimes puts two active (100%) threads in the same logical processor for seconds ==> drastic performance drop (fragment rate drops to 60%)
- No performance variation if the thread which has a physical processor on its own changes.

Conclusions:

- The active threads in the RU are currently balanced (no needs significant more CPU time than the other)
  - this was not the case before the mentioned bug-fix: if the RUI thread had a physical processor on its own, the performance was significantly better.
- Hyperthreading is efficiently exploited by the EVB software.
  - Otherwise the performance drop if Linux puts 2 threads in the same logical processor would be less significant.