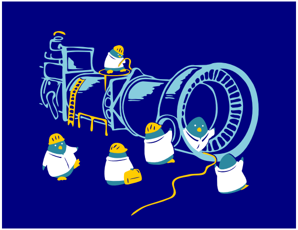
NA62 Online PC Farm Design and Implementation

Jonas Kunze

Universität Mainz

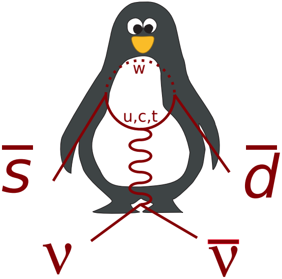
29.02.2012



SPONSORED BY THE

Federal Ministry
of Education
and Research

1. Motivation
   - The physics
   - The experiment
2. DAQ and Trigger
   - Three level online trigger
3. Merging L1 and L2
   - New proposal
   - Event building @ L1
4. Performance
5. Outlook and conclusion



$K^+ \to \pi^+ \nu \bar{\nu} \Leftrightarrow V_{td}$ of CKM matrix

SM branching ratio: $(8.5 \pm 0.7) \cdot 10^{-11}$

NA62 aims $\sigma < 10\%$ with 100 events

$\approx 10^{13}$ K^+ decays required

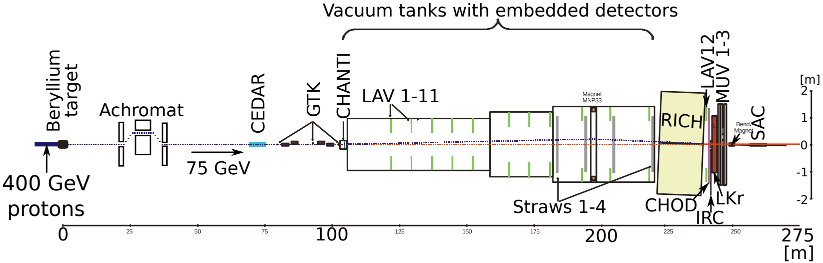
Data taking planned for 2014-2015, first test run this year

Signal-to-noise ratio of $1/10$ planned with an event rate of 10 MHz

High efficiency needed ⇒ high data rate

⇨ High-performance DAQ and Trigger necessary

NA62 Experiment at CERN



Data rates

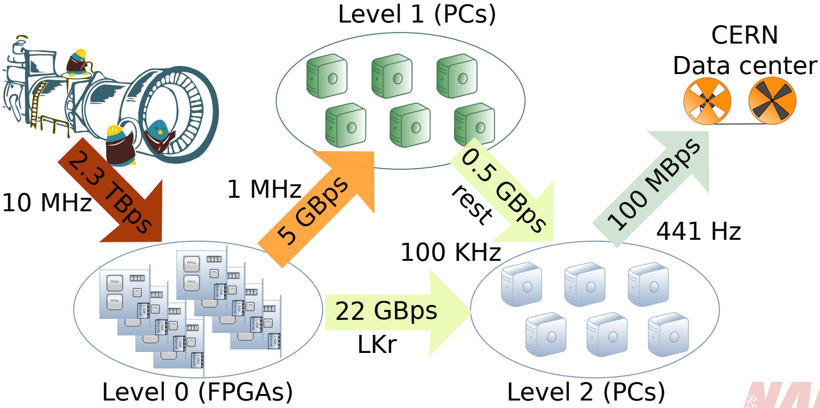
10 MHz event rate

Detector	Event size [B]	Data rate [GBps]
CEDAR	216	2.16
GTK	2250	22.50
CHANTI	192	1.92
LAV	160	1.60
STRAW	768	7.68
RICH	160	1.60
CHOD	$\ll$ 1000	$\ll$ 10
MUV	768	7.68
IRC & SAC	576	5.76
LKR	**222 k**	**2220**
Sum	**≈227 kB**	**≈2.3 TBps**

DAQ and Trigger system

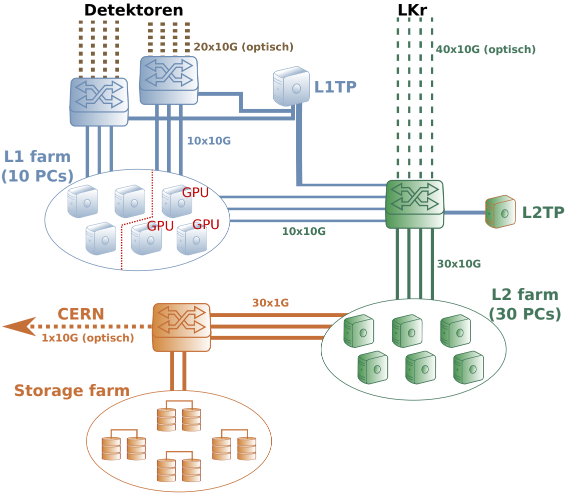
Three levels to filter data

Data transmission via ordinary 10 gigabit ethernet and **UDP/IP**:



First topology proposal

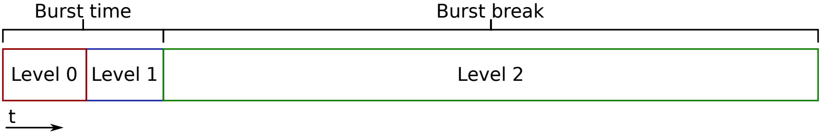
Original concept:



Burst time and duty-cycle

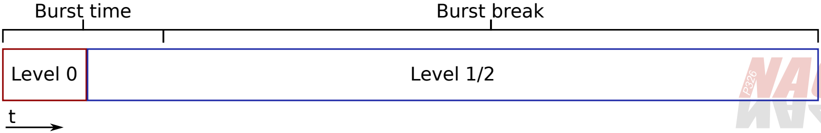
Only 3-9 sec. burst and long break

Duty cycle: $T_{Burst} / T_{Break} \approx 0.3$

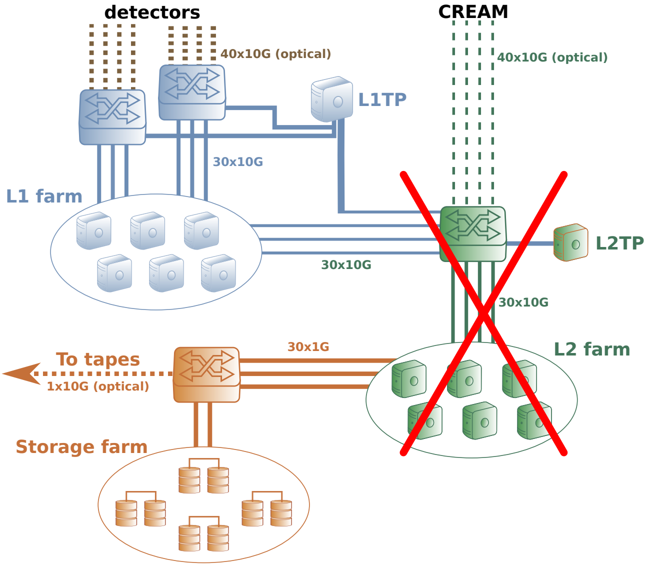


My proposal to use resources more efficiently

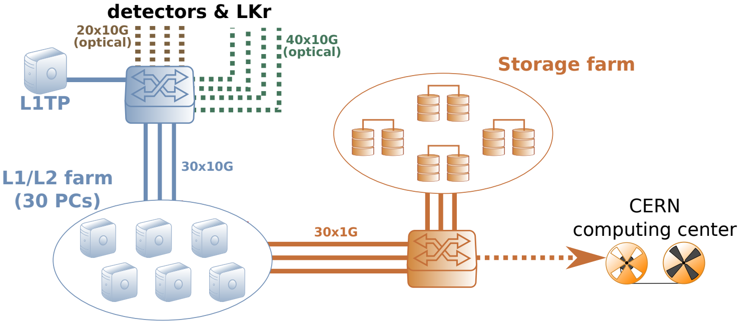
Reuse L1 PCs during burst break for L2 computation by combining L1 and L2 to one farm



Don’t separate L1 and L2!



Combine L1 and L2 to one farm

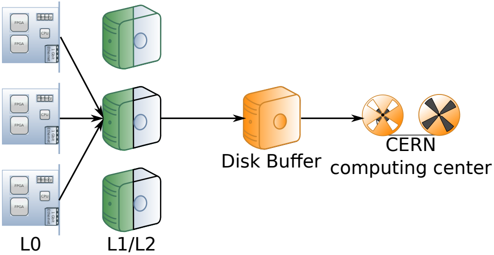


We save about 80k

- No L1 PCs anymore
- Less switches, less network cards

New proposal

Event building @ L1



Every subdetector sends data of an event to one single PC

- + No broadcast of a L1 decision needed anymore (no L1TP)
- + Easier to implement load balancing (self-sustaining PCs)
- − Every farm PC must serve every subdetector ⇒ needs GPUs

pf_ring - new type of network socket

Bad performance with standard Kernel sockets

Interrupt based transmission causes packet loss

Special socket: pf_ring DNA by ntop

- Direct access to the NIC memory (avoids system calls)
- Only ≈40% CPU @ full speed 10G receiving 1kB packets
- No packet loss at all

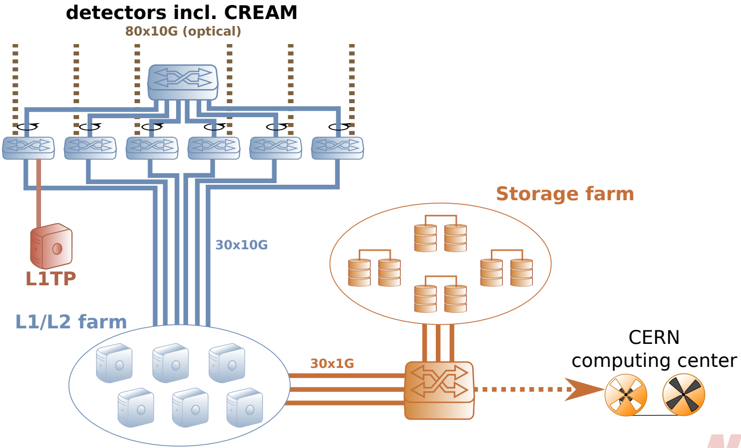
270kHz Eventbuilding rate with only 5 virtual cores
⇒ 19 cores left for L1 and L2 trigger

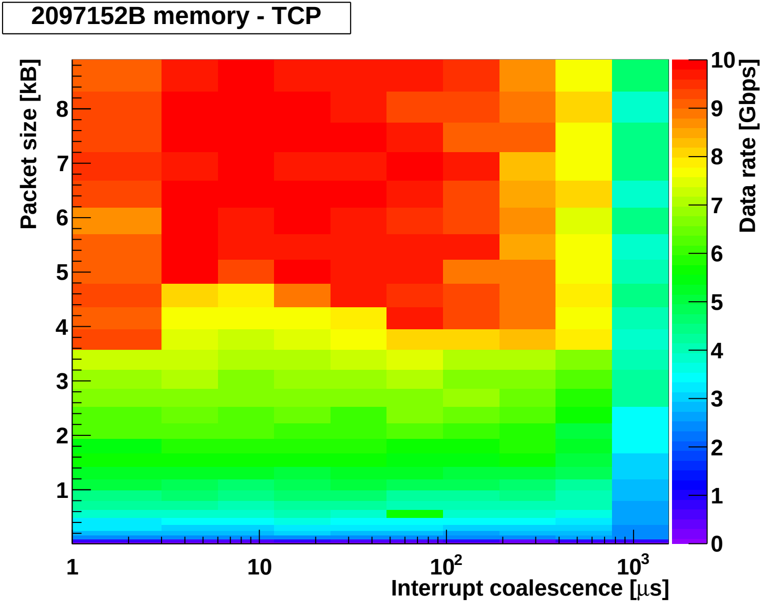
Conclusion

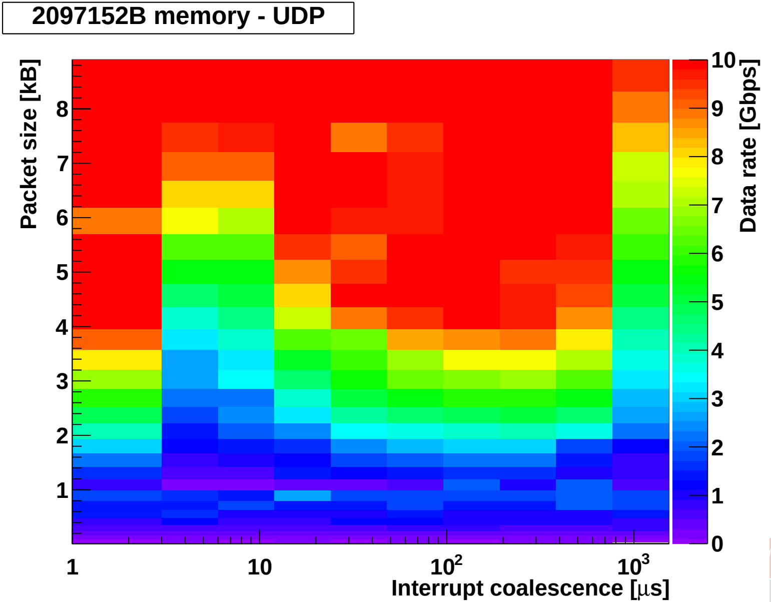
- High energy and high precision ⇒ a lot of data
- Using ordinary ethernet saves money and time and gives you the ability to quickly switch between different approaches
- Special driver needed for lossless communication: pf_ring
- Unsteady data production allows new approaches
  - Considering trigger levels as logical object, not as real farms saves a lot of money
- The new farm design allows us to have a central software architecture which is much easier to implement and maintain

Thank you!

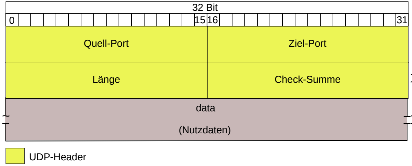
Tree topology (Hexapus)




2097152B memory - TCP
Packet size [kB]
1
2
3
4
5
6
7
8
1
10
10^2
10^3
Interrupt coalescence [µs]
Data rate [Gbps]
0
1
2
3
4
5
6
7
8
9
10


2097152B memory - UDP
Packet size [kB]
8
7
6
5
4
3
2
1
1
10
10^2
10^3
Interrupt coalescence [µs]
10
9
8
7
6
5
4
3
2
1
0
Data rate [Gbps]

TCP vs. UDP



32 Bit

0 15 16 31

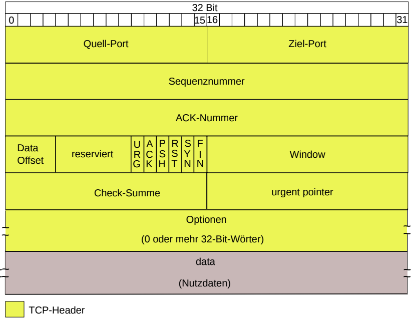
Quell-Port | Ziel-Port

Länge | Check-Summe

data

(Nutzdaten)

🟨 UDP-Header



32 Bit

0 15 16 31

Quell-Port | Ziel-Port

Sequenznummer

ACK-Nummer

Data Offset | reserviert | URG | ACK | PSH | RST | SYN | FIN | Window

Check-Summe | urgent pointer

Optionen

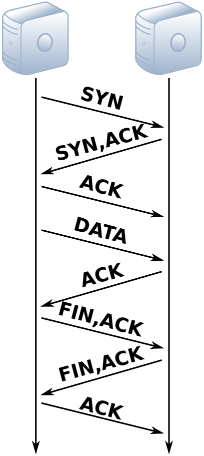
(0 oder mehr 32-Bit-Wörter)

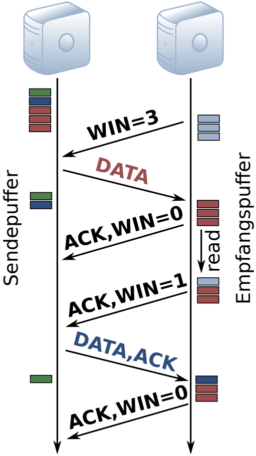
data

(Nutzdaten)

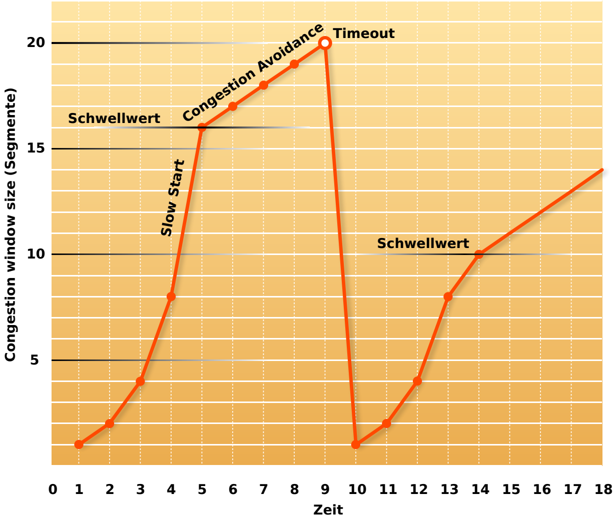
🟨 TCP-Header

TCP: Reliability and flow control





Congestion Avoidance



Performance tests

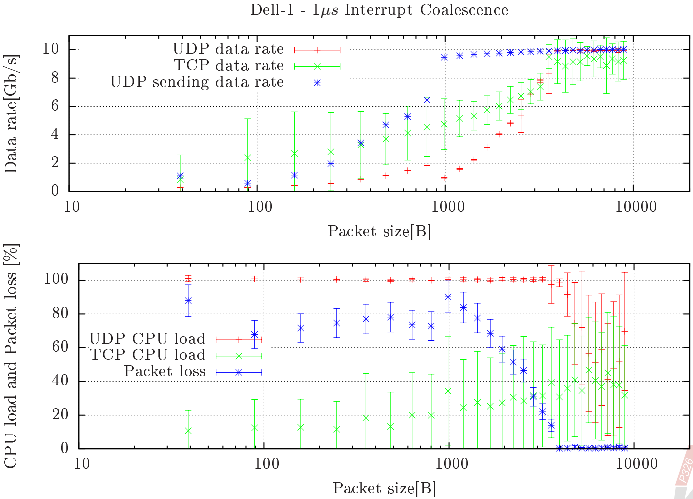
⇨ TCP optimizes the usage of network resources

But what does this cost?

Intuitionally one would guess: Higher CPU usage and longer latencies.

but. . .

Data rate and CPU usage



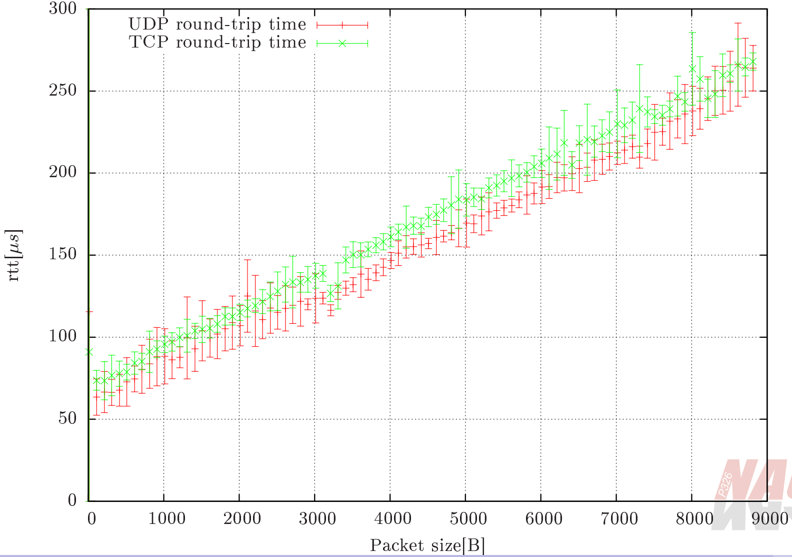
TCP Offload Engine

Network cards and drivers are optimized for TCP

- Checksumms and fragmentation calculated on hardware
- Some drivers ignore interrupt coalescence of $0\mu s$

⇨ Using TCP reduces CPU usage $\Rightarrow$ more space for computation

Timing


UDP round-trip time
TCP round-trip time
rtt[μs]
Packet size[B]
0
50
100
150
200
250
300
0
1000
2000
3000
4000
5000
6000
7000
8000
9000

Results

- TCP reduces CPU usage ⇒ more space for computation
- TCP has flow control and congestion avoidance
- TCP is reliable

TCP in FPGAs

TCP means high payload in hardware!
⇒ TCP can only be used for PC to PC communication at NA62!

TCP/UDP vs. basic IP

Using standard interrupt/kernel based socket programming. . .

- is optimized for TCP (drivers)
- + is easy and many libraries can be used (e.g. boost::asio)
- – induces high latency ($\approx 30 - 150 \mu s$)
- – induces high packet loss??!!

Programming own Kernel modules. . .

- – is hard stuff (only few small libraries)
- – is bound to hardware
- + highest performance possible