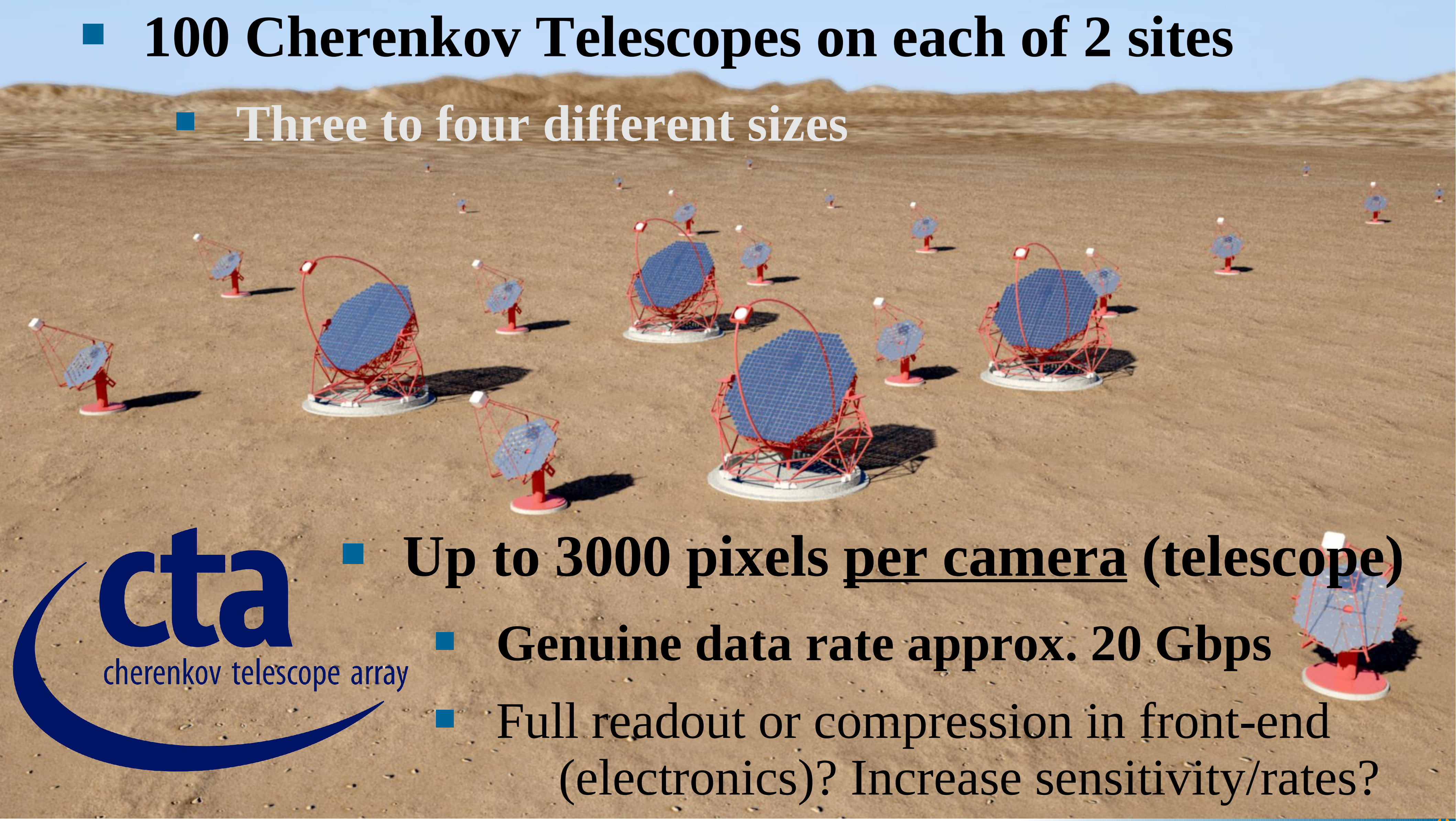
10Gigabit-Ethernet Event-Builder for a Cherenkov Telescope Array

Dirk Hoffmann, Julien Houles
Centre de Physique des Particules de Marseille

Outline

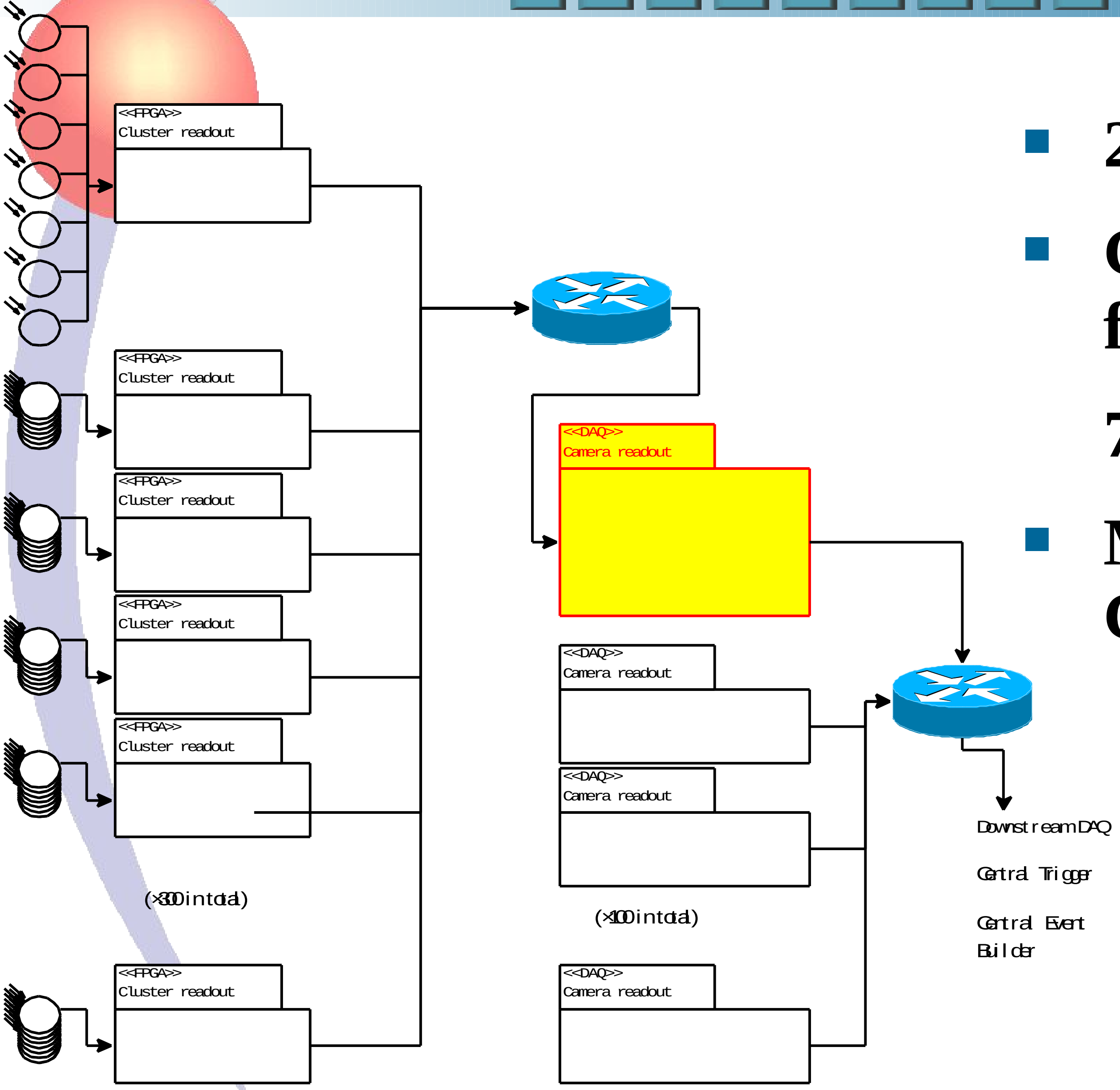
- Experimental Context, Constraints
- Hardware Choice
- Event-Builder Design
- Data Generation (test-bench stimulator)
- First Results in standard Linux
- Consequences, Interpretation, Prospects

How the experiment may look like



- **100 Cherenkov Telescopes on each of 2 sites**
  - **Three to four different sizes**
- **Up to 3000 pixels <u>per camera</u> (telescope)**
  - **Genuine data rate approx. 20 Gbps**
  - Full readout or compression in front-end (electronics)? Increase sensitivity/rates?

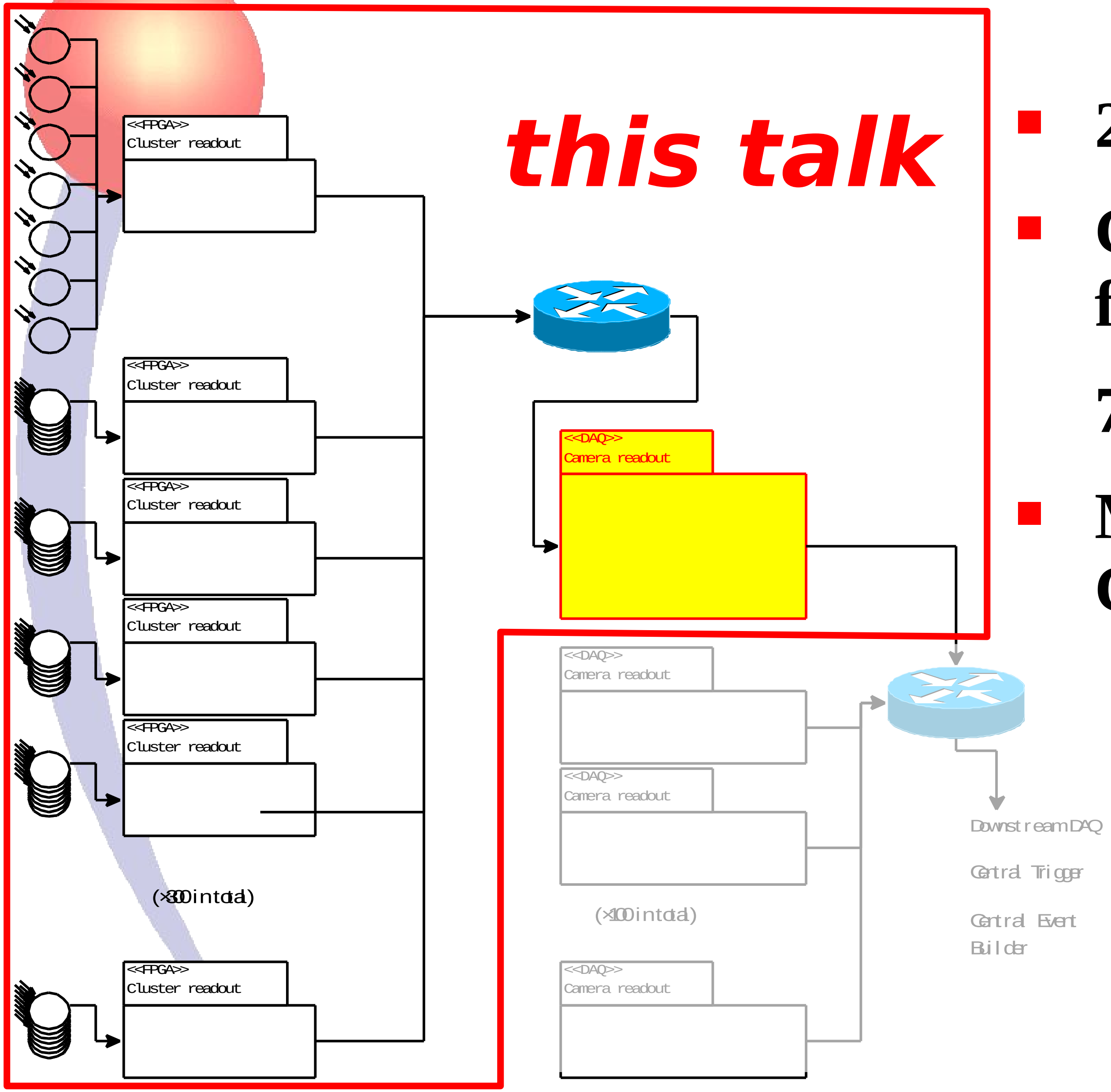
Schematic View



- **20 Gbps total**
- **Groups of 7 pixels per front-end board:**

  70 Mbps per board
- **Merged by Camera switch**
  - **Collected by Camera Server**
  - **Occupancy 1%**
  - **200 Mbps down-stream** (20 Gbps)

Schematic View



- 20 Gbps total
- Groups of 7 pixels per front-end board:

  70 Mbps per board
- Merged by Camera switch
  - Collected by Camera Server
  - Occupancy 1%
  - 200 Mbps downstream (20 Gbps)

DAQ Requirements

- **Average full data stream of 20 Gbps**
- **Needs reduction.**
  - Trigger selection (obviously done)
  - Compression on board (fit, parameters): ToT, amplitude
  - Reconstruction in camera and second level filter
  - Compression (lossless?) in Camera-Server
- **O(*n*·100) datasources**
  - Reliable event-building
- **Cohabitation with Slow Control Traffic possible?**
- **Optimised cost, industrialisation for the array!**

Hardware choice

- **Selection of COTS hardware for cost ALARA**
- **Generic test of state-of-the-art technologies**

- **Precision T7500 Server**
  - 2×Xeon X5650
    2.7GHz, 6.4GT/s,
    12MB, 6 cores
  - Intel X520 DA2
    10GbE dual port SFP+
    on PCIe×8
  - GPU (PCIe×16) option

12 cores @ 2.7GHz

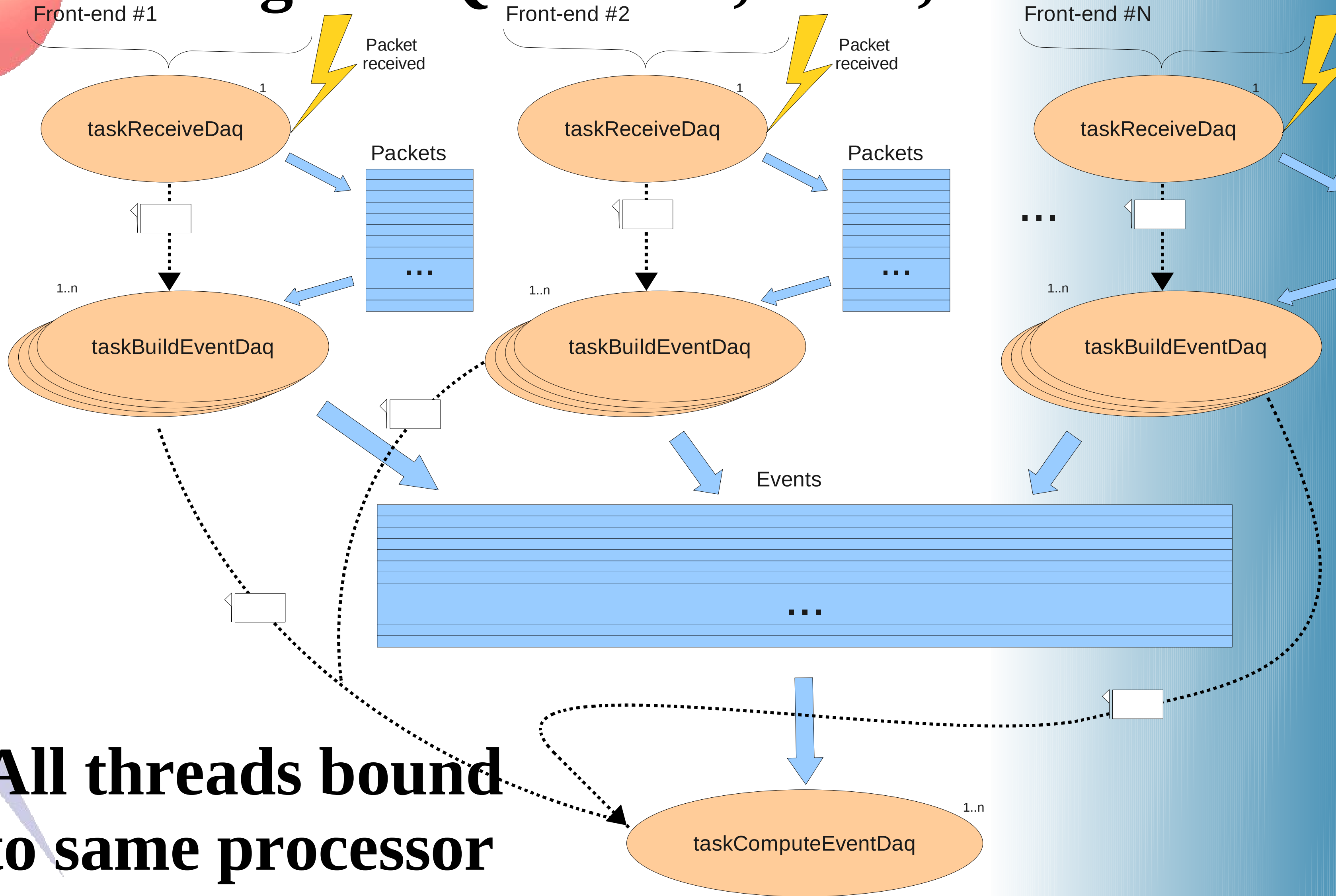
- **Powerconnect 6248**
  - 48× 1Gbps (RJ45)
  - 4× 10Gbps (SFP+)
  - 4× 1Gbps (SFP)
  - stackable (max. 12)
    with backplane ic
  - Jumbo frame support

Event Builder

- **Collect event fragments:**
  - Typical event per board has 1kB size
    Bundle them in front-end? May vary!
  - UDP protocol chosen for transfer
  - Jumbo frame support (MTU>1518)
- **Build events**
  - 20 (later 24) Gbps input / 200Mbps output
- **Minimize workload** (cost and cohabitation!)

Software design #1

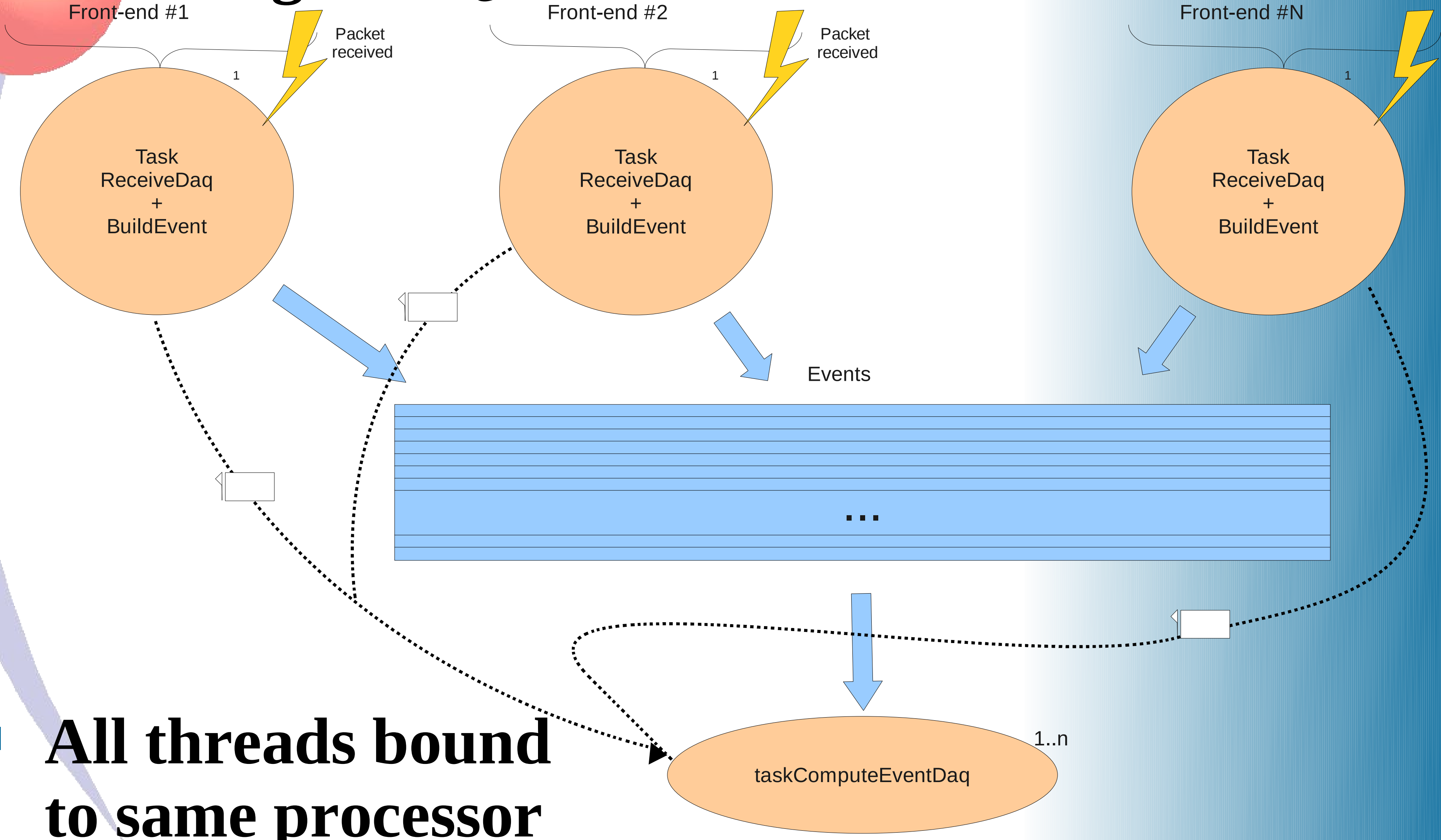
- Three-stage DAQ: receive, build, send



- All threads bound to same processor

Software design #2

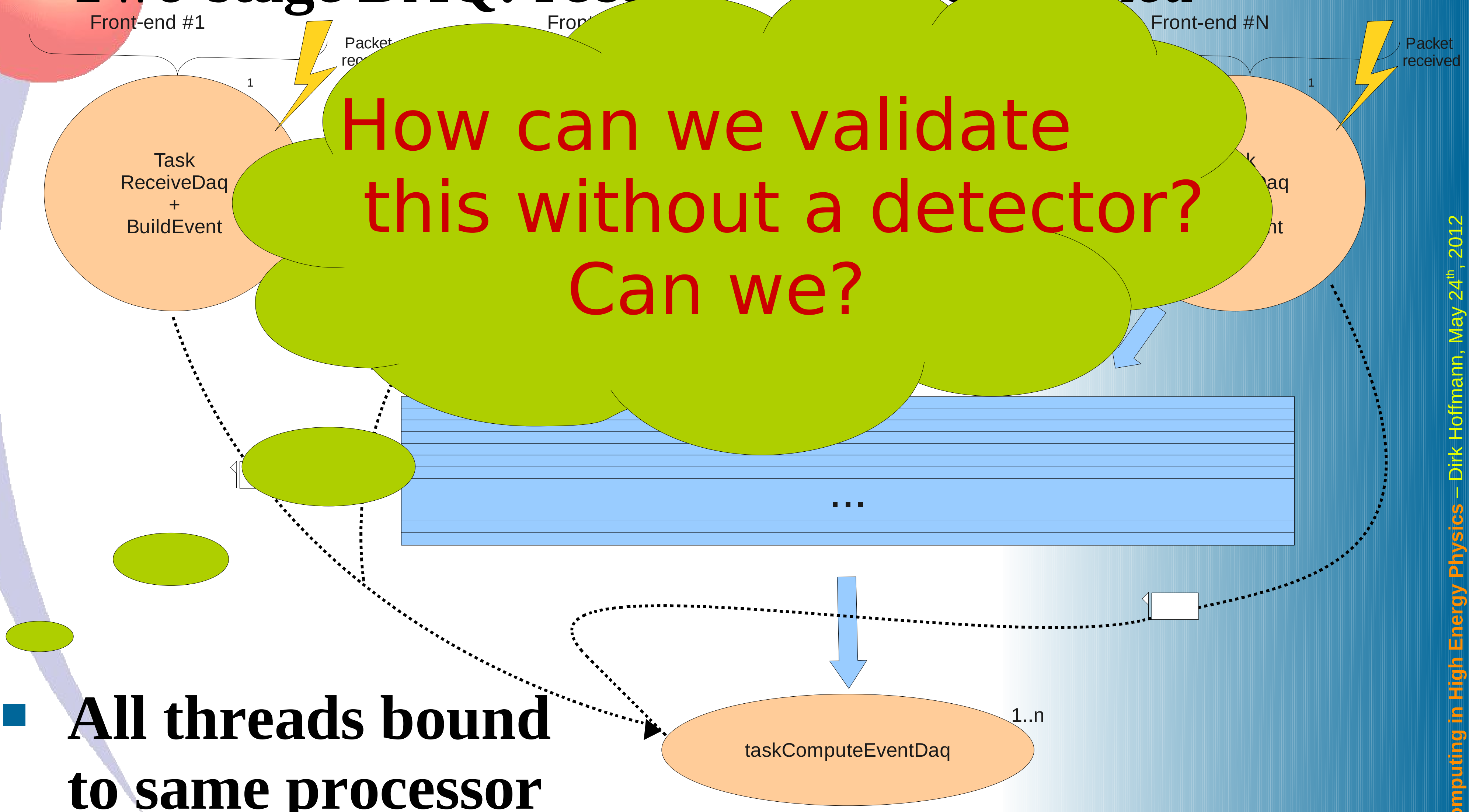
- **Two-stage DAQ: receive + build *combined***



- **All threads bound to same processor**

Software design #2

- **Two-stage DAQ: receive + build *combined***



How can we validate this without a detector? Can we?

- **All threads bound to same processor**

Data Generation, DAQ S[t]imulator

- **DAQ is prototype, electronics as well.**
- **Simulate camera on site at lowest cost**
  ⇒ *Side-effect and real requirement!*
- **Hence build a “camera simulator” to stimulate the Event-Builder DAQ**

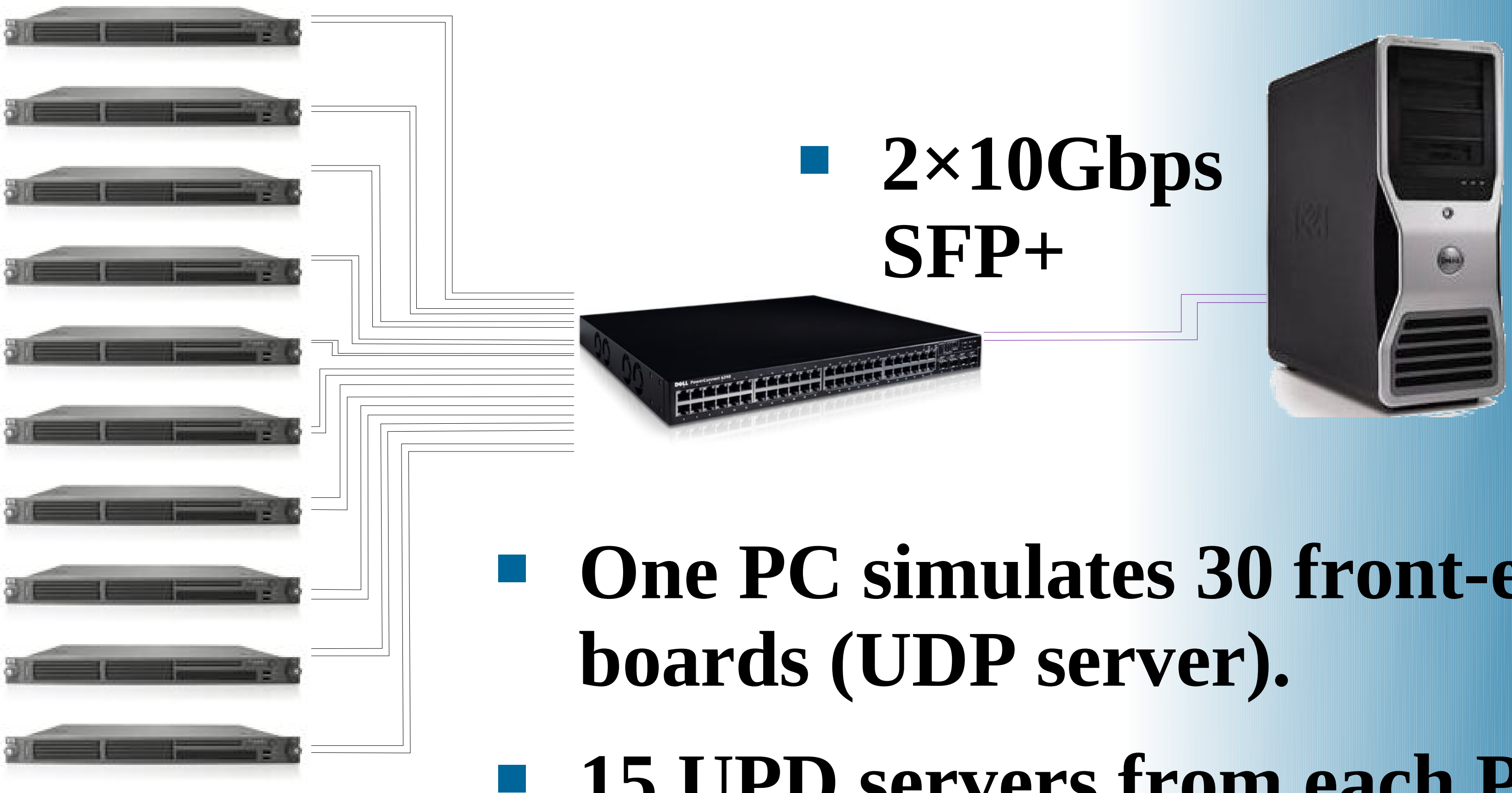
Stimulator: optimum

- **High-bandwidth is standard, many ports isn't!**
- **Found a 50€/port candidate (EVOC)**

- **6×1000baseT (via PCIe each)**
- **Internal architecture is relevant.**
  - PCI 32/64 = 133/266 MBps
  - PCIe = 500 MBps (here: PCIe v1.1 = 250 MBps)

Stimulator: reality (for now)

- 10 “borrowed” GRID PCs = 10×2×1000baseT



- 2×10Gbps SFP+

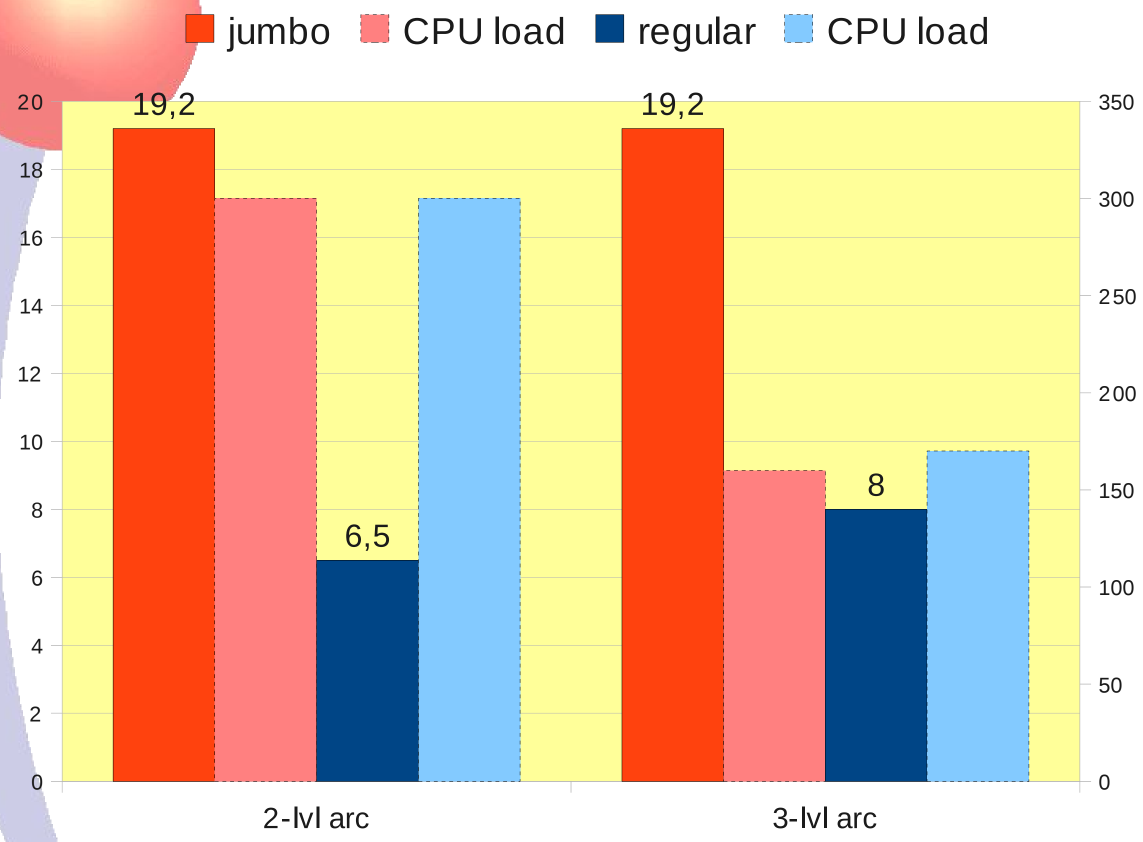
- One PC simulates 30 front-end boards (UDP server).
- 15 UPD servers from each PC per SFP+ interface

Results

Packet size	Three-stage architecture	Two-stage architecture
Jumbo (8192 bytes)	**19.2 Gbps** (2.4 GBps) CPU load: 300%	**19.2 Gbps** (2.4 GBps) CPU load: 160%
Regular (1024 bytes)	**6.5 Gbps** (820 MBps) CPU load: 300%	**8 Gbps** (1.0 GBps) CPU load: 170%

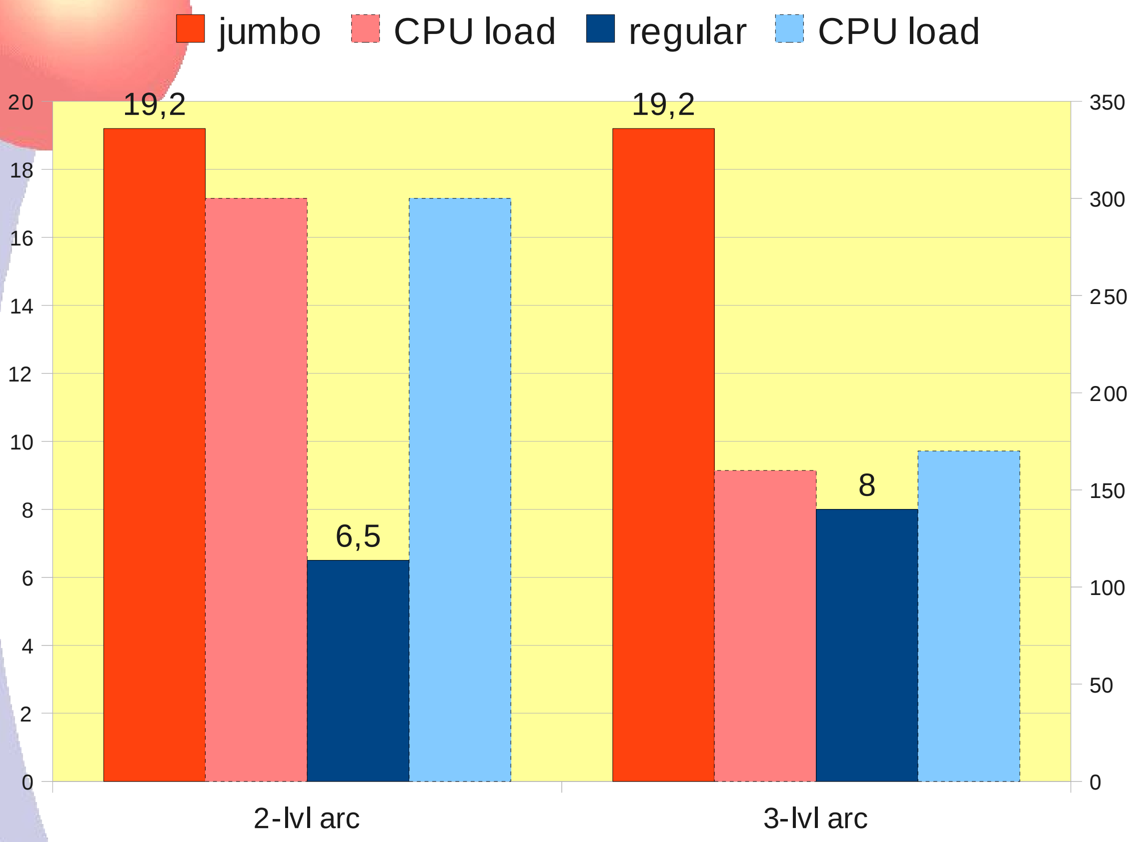
- **All events assembled and checked (no I/O)**
- **No loss of packets**
- **Standard h/w**
- **Standard s/w (SL6 drivers, libraries)**

Interpretation



- **Significant loss of performance for “small” frames**
- **2-lvl architecture outperforms 3-lvl architecture: Less than 2 cores needed**

Interpretation



- **Significant loss of performance for “small” frames**
- **2-lvl architecture outperforms 3-lvl architecture: Less than 2 cores needed**

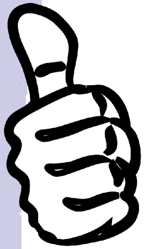
- **Where is the bottleneck?**

Limitations and possible Solutions

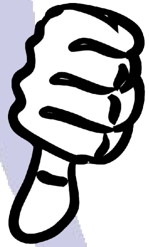
- Standard libraries / drivers provide optimal performance (assuming optimal data formats).
- Moore's law helps to overcome wildest dreams (or bad design).
- But CPU / IC design hits the limit of power dissipation before the limit of 1 Å or *c*.

“Free lunch is over.”

- Computing power is increased by

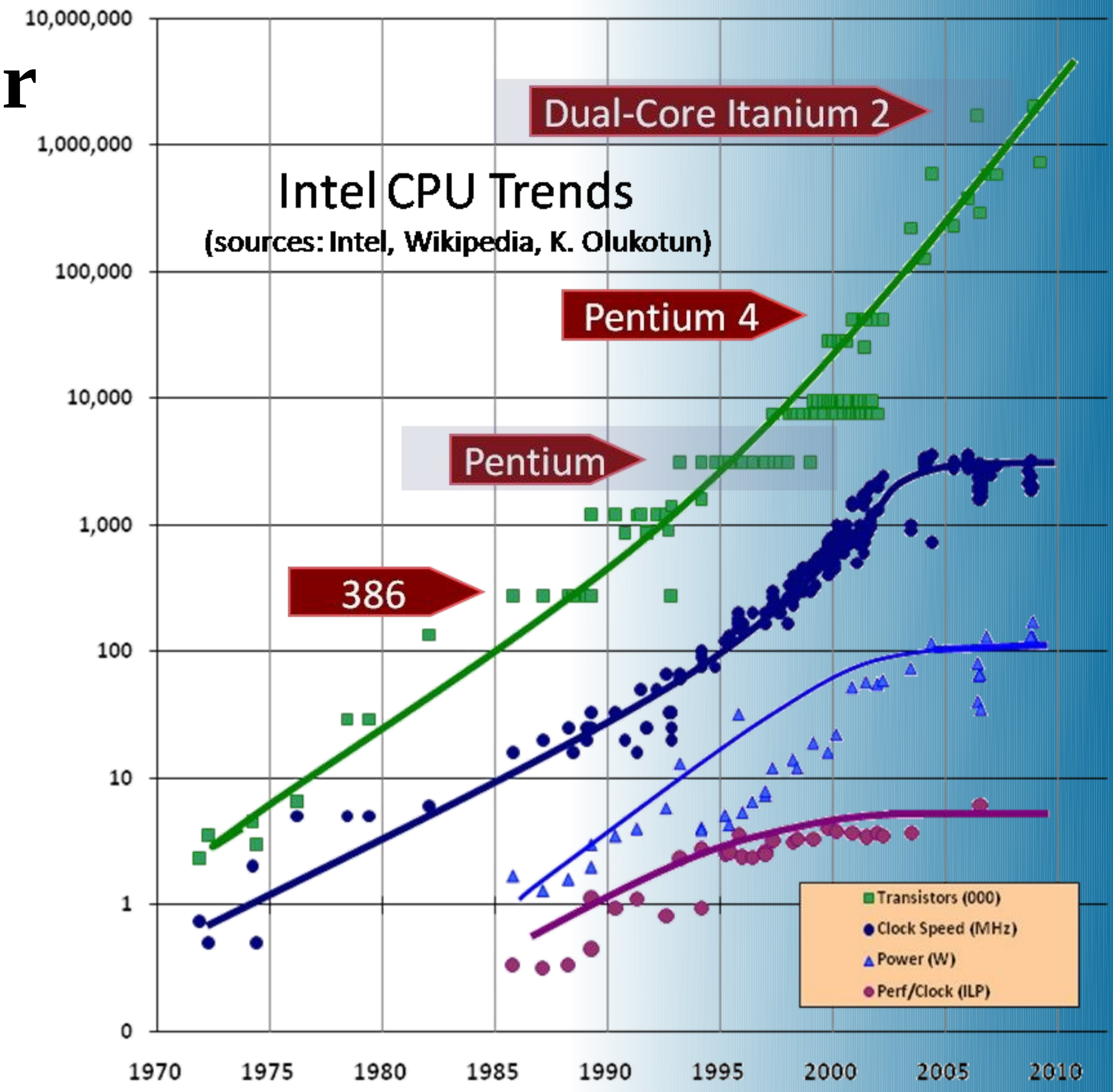


multiplying the number of cores and CPUs



rather than increasing clock frequency

- UNLIKE NETWORKS!



A loong way to 10 Gbps

Courtesy L. Rizzo, U Pisa

- **1980-2010:**
  - 4 Mbps (token ring)
  - 10Gbps (25 soon?)

- **But software architectures are still the same.**
  - raw socket, BPF, libpcap
  - mbuf/skbuf/NdisPacket encapsulation
  - one system call per packet, poor parallelism
- **Even with faster clock speeds, some things do not scale:**
  - memory and bus latency, system calls

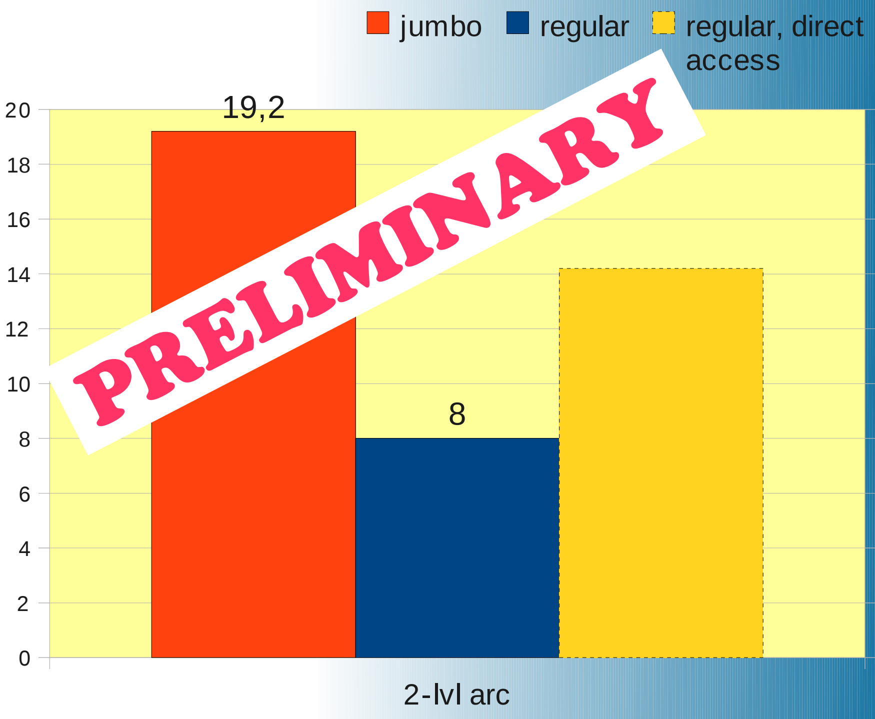
What next?

- **Recent work on libraries to replace 30-year old Unix/Linux driver technology,**

  Using direct access to network components (h/w – memory map)
  (This is critical by default, due to access of kernel memory!)

- **Need work on both sides!**
  TX/RX

- **Increased to 7.1 Gbps in first tests on single link with regular packets**



- **More about this in Amsterdam 2013?**

Conclusion

- **It is relatively easy to build a 10Gbps data transfer and collection system (Event-Builder).**
  - With COTS hardware
  - Combining multiple data sources
  - With reasonably low CPU load (2-3 cores)
  - Using standard Linux drivers and libraries
  - *Packaging data in maximum sized packets.*

- **Discrepancy between progress in CPU/IC and network technology necessitates new h/w access methods.**