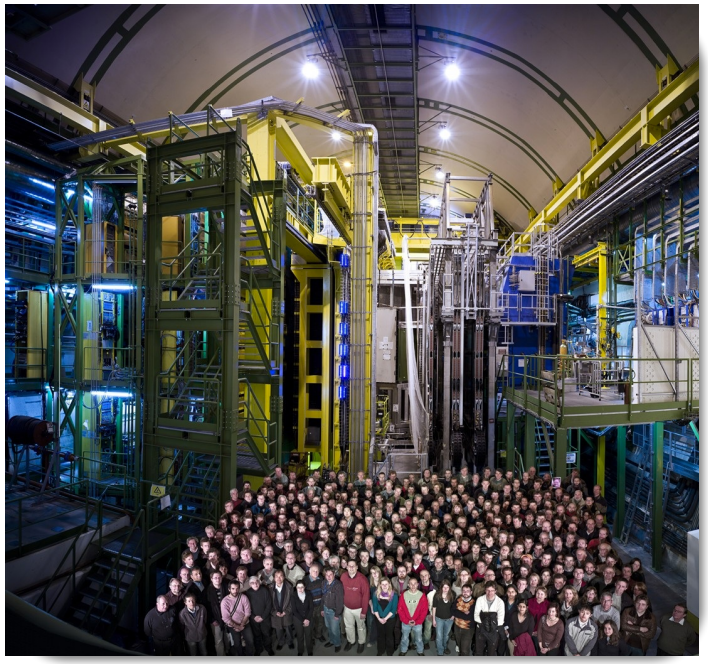
LHCb Event Building on high-performance interconnections

Matteo Manzali
INFN - University of Ferrara

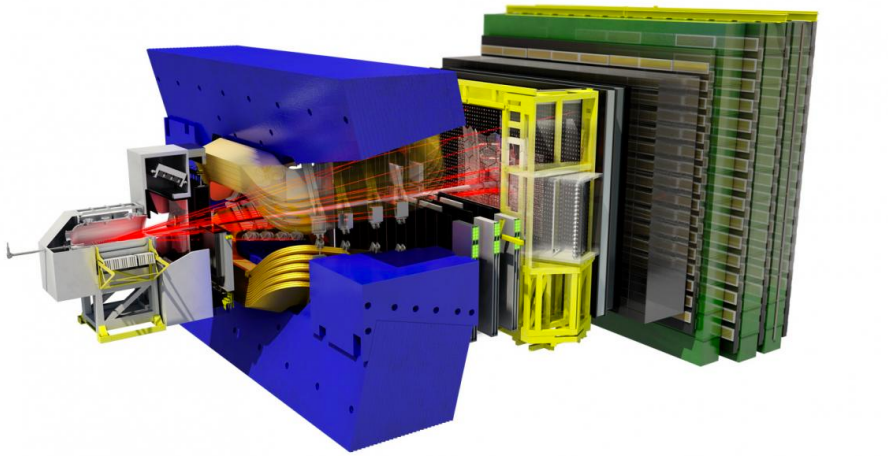
The LHCb experiment

- The LHCb experiment is one of the four large experiments based at CERN
- A major upgrade is scheduled in the 2019-2020 period:
  - Upgrade of the detector
  - Upgrade of the Data Acquisition system (DAQ)



- Currently the primary event filter is performed using custom FPGAs, cutting off the acquisition frequency to 1 MHz
- For the upgrade a full software filter is foreseen, allowing to acquire data at the maximum frequency available of 40 MHz

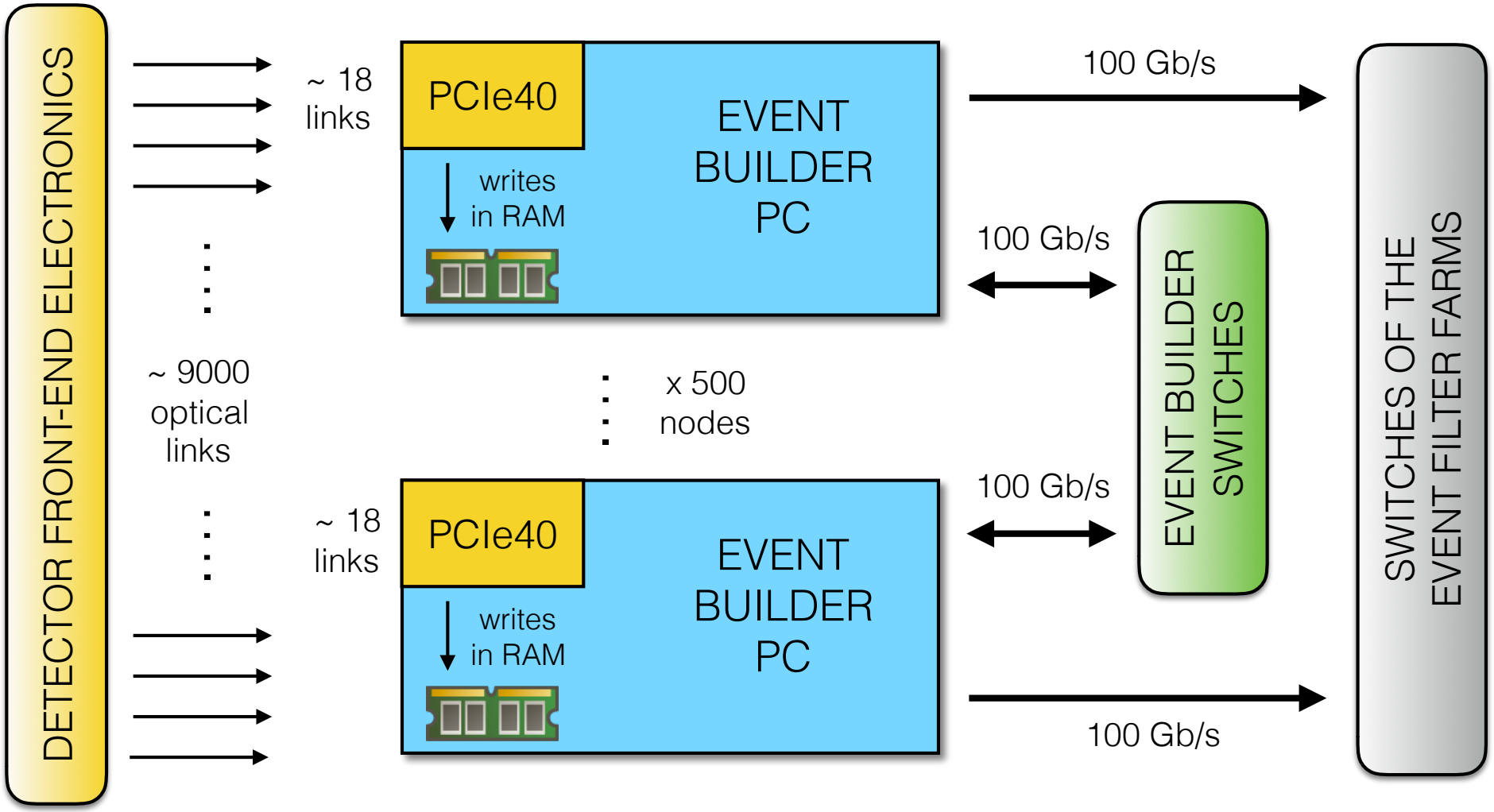
The LHCb experiment



conservative values

	present		future
Event size	65 KB	→	100 KB
Event rate	1 MHz	→	40 MHz
Aggregate bandwidth	520 Gb/s	→	32 Tb/s
Readout boards	300	→	500

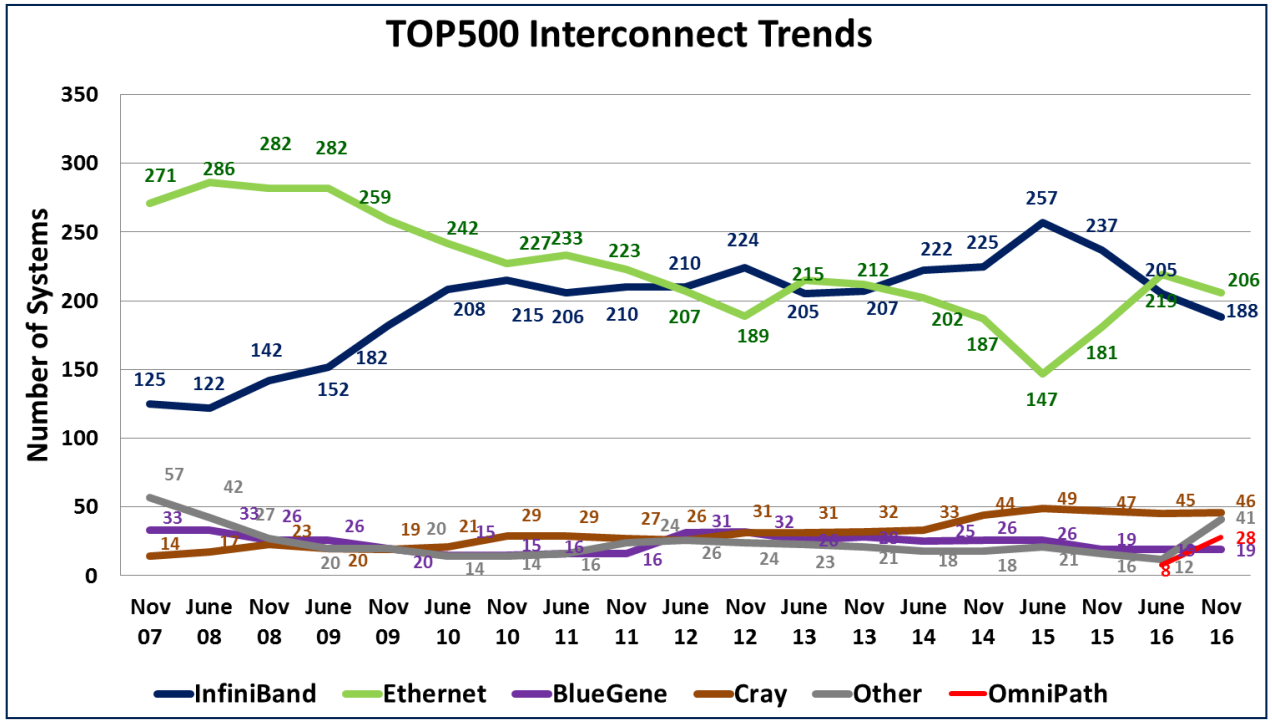
Upgraded DAQ design



Network technologies

- Different 100Gb/s network technologies under study by the LHCb online working group (Ethernet, InfiniBand, Intel OmniPath)

- Remote Direct Memory Access (RDMA) feature required:
  - remote memory access without involving OS and CPUs
  - low CPU utilization, low latency, zero-copy support



Software technologies

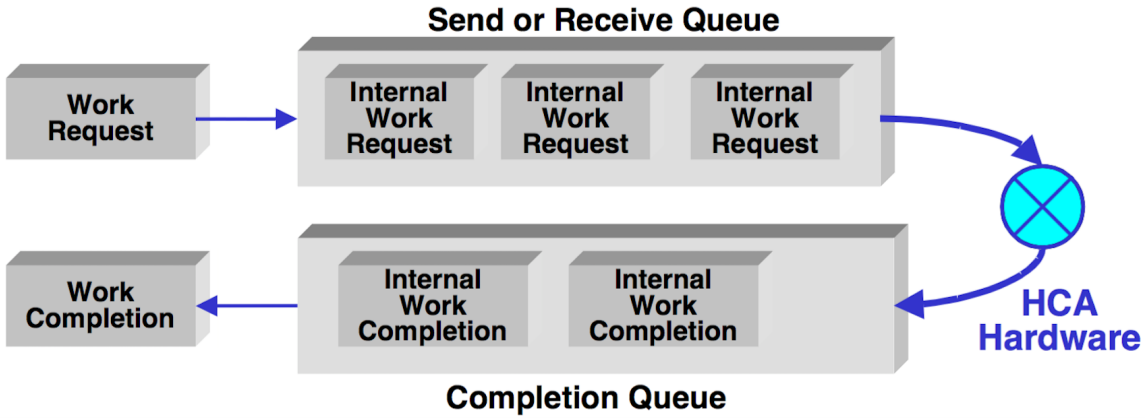
- Each interconnect provides at low level a specific interface (sockets, verbs, PSM2, etc…)

- MPI can be used to run over all these interconnects:
  - simpler software development
  - hidden complexity
  - it is difficult to investigate inefficiencies
  - natively it aborts all the distributed processes in case of failure
  - is is difficult to implement any fault tolerance mechanism

- Libfabric is a framework that exposes a unified API over all these interfaces:
  - it doesn't hide the underlying complexity

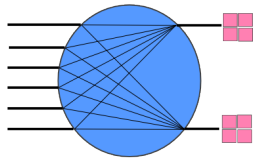
How RDMA works

- Each RDMA communication is identified by a Queue Pair (Send and Receive)
- Asynchronous operations:
  - the host posts Work Requests into the Queue Pair (read, write, send, recv)
  - once the operation is completed, a Work Completion is posted into a Completion Queue
  - check of completion with busy polling or event notification approach



EB implementations

- There are currently two EB implementations:
  - **DAQPIPE v2** (https://goo.gl/glFI0M)
    - developed to test different approaches and protocols
    - official starting point for the upgraded Event Builder
  - **LSEB** (https://goo.gl/Er3rfV)
    - developed to benchmark interconnects with the least overhead possible
    - developed by INFN



DAQPIPE

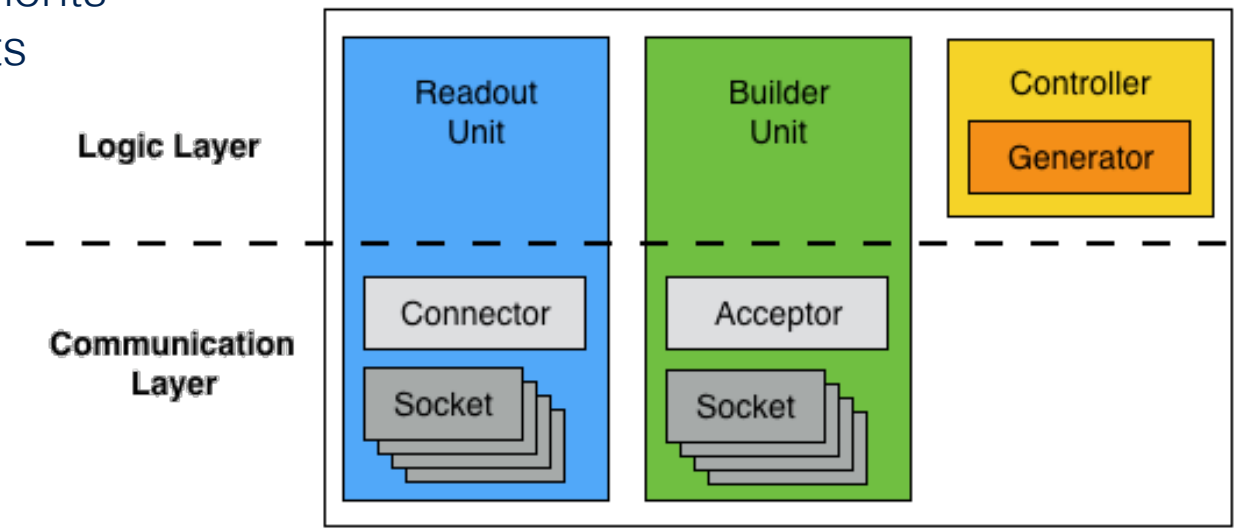
- LSEB - “Large Scale Event Builder”:
  - based on the verbs library with busy polling approach
  - C++11 and Boost libraries (~3400 lines of code)

The Large Scale Event Builder

- A LSEB process is mainly composed of two distinct logical components: the Readout Unit (RU) and the Builder Unit (BU)

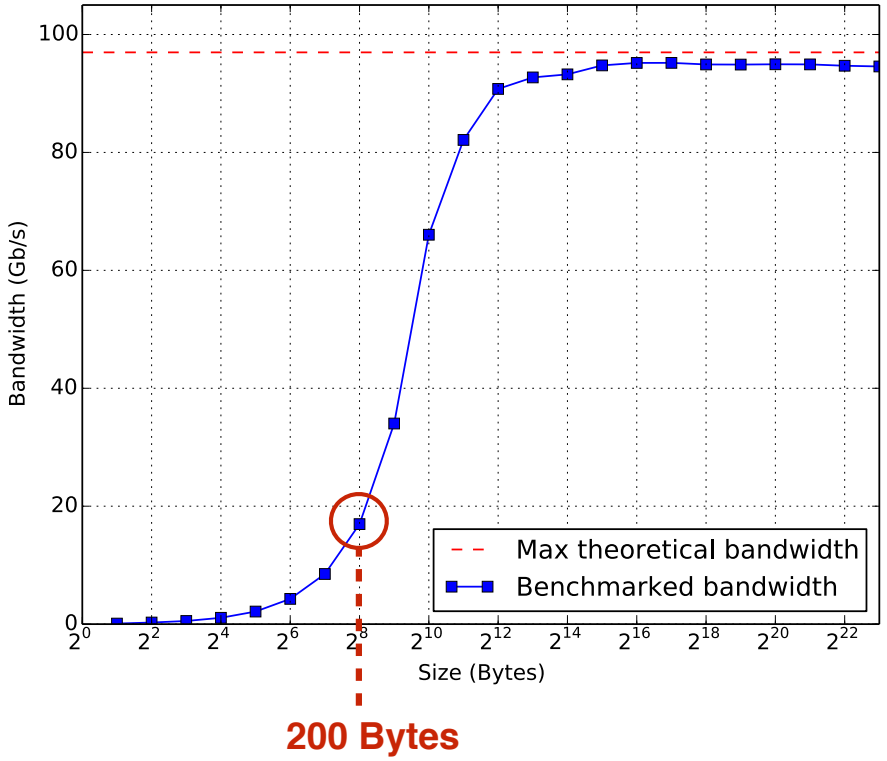
- Each RU:
  - receives the event fragments from a generator
  - ships them to the receiving BU in a many-to-one pattern

- Each BU:
  - gathers event fragments
  - generates full events



Buffering

- Fragment size: ~200 Bytes
- A single fragment could not be enough for bandwidth saturation
- Solution: SEND / RECV of bulk of contiguous fragments

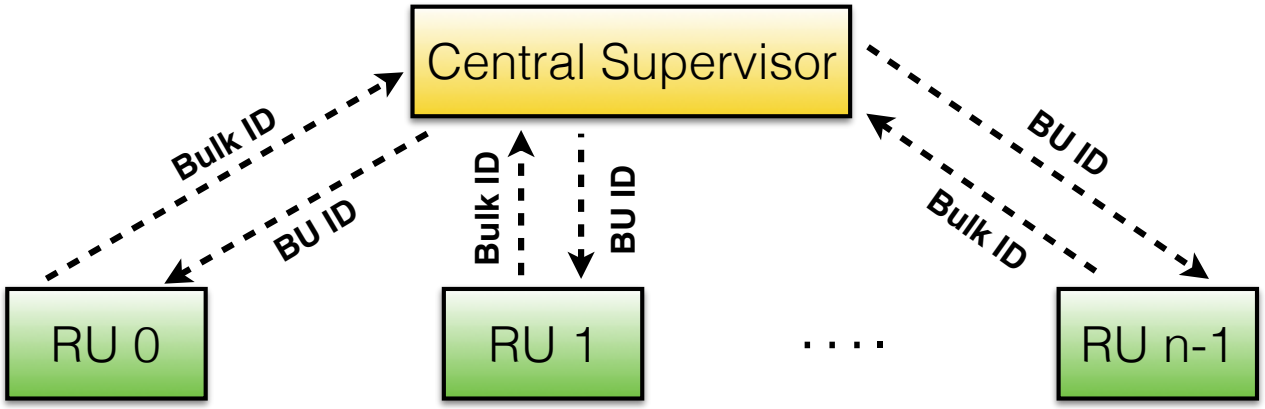


Scheduling strategy

- A central supervisor may be used to decide which BU has to build which range of events

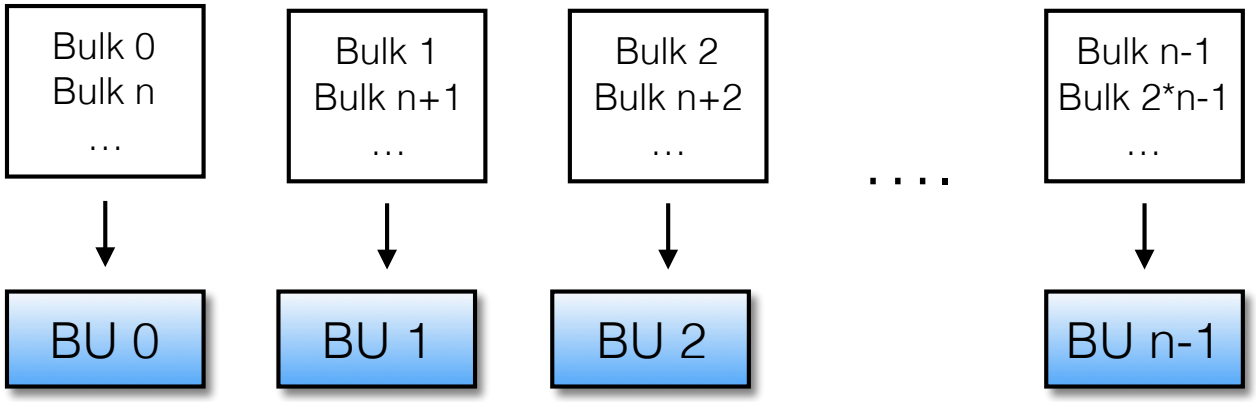
- **PROS -** Using a central supervisor allows to:
  - perform load balancing
  - perform fault tolerance mechanism

- **CONS -** It may cause an overhead in terms of:
  - software complexity
  - latency
  - network traffic

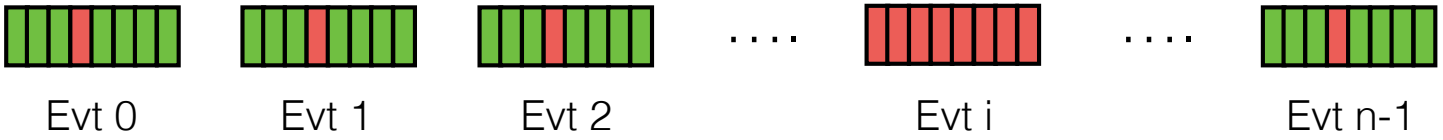


Scheduling strategy

- In LSEB it is not used a central supervisor, preferring a pre-defined Round Robin scheduling strategy:

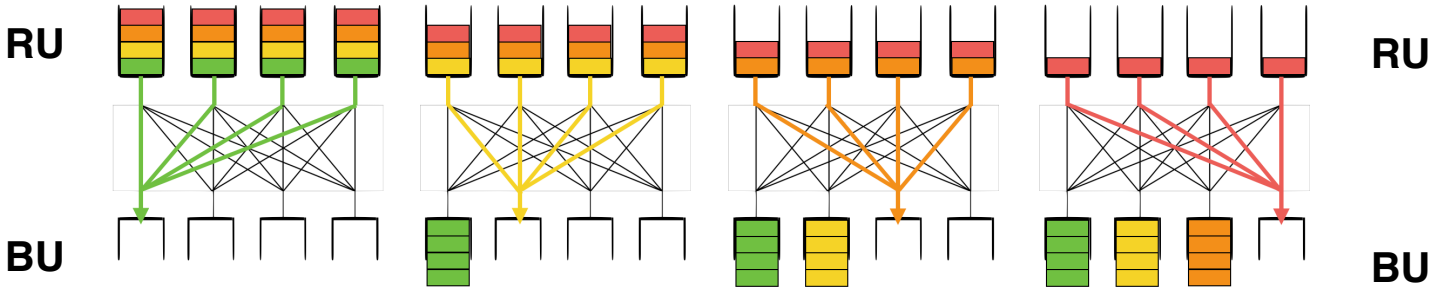


- In case of fault of a single node there is the loss of:
  - 1 / N events (missing BU)
  - 1 / N fragments for each event (missing RU)

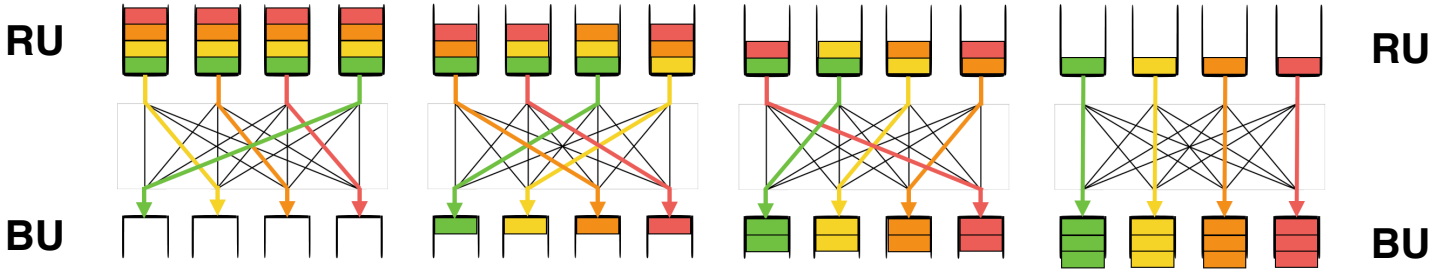


Traffic shaping

- Ideally each RU sends data to the same BU at the same time
- This may produce a traffic congestion:



- One possible solution is to introduce a traffic-shaping strategy
- In LSEB each RU starts to send data to the BU with subsequent ID:

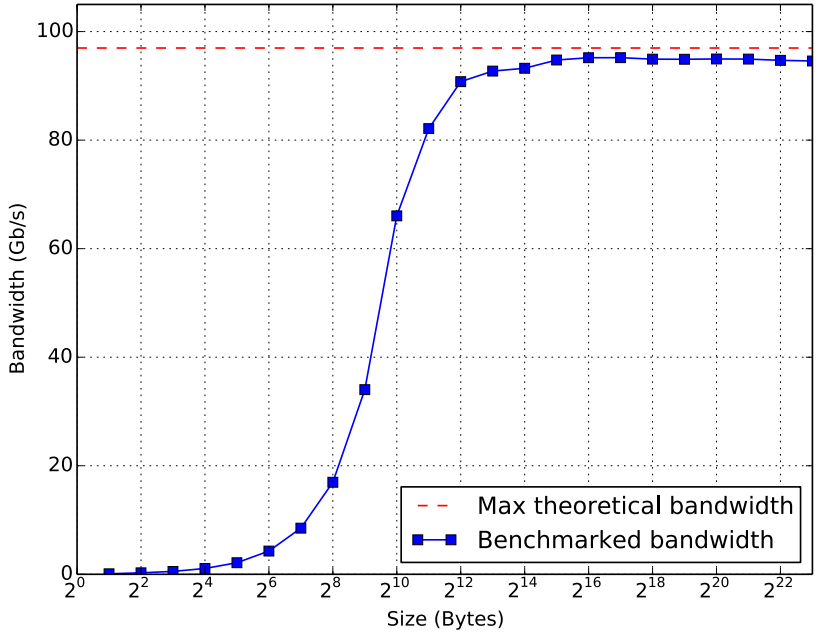


InfiniBand clusters

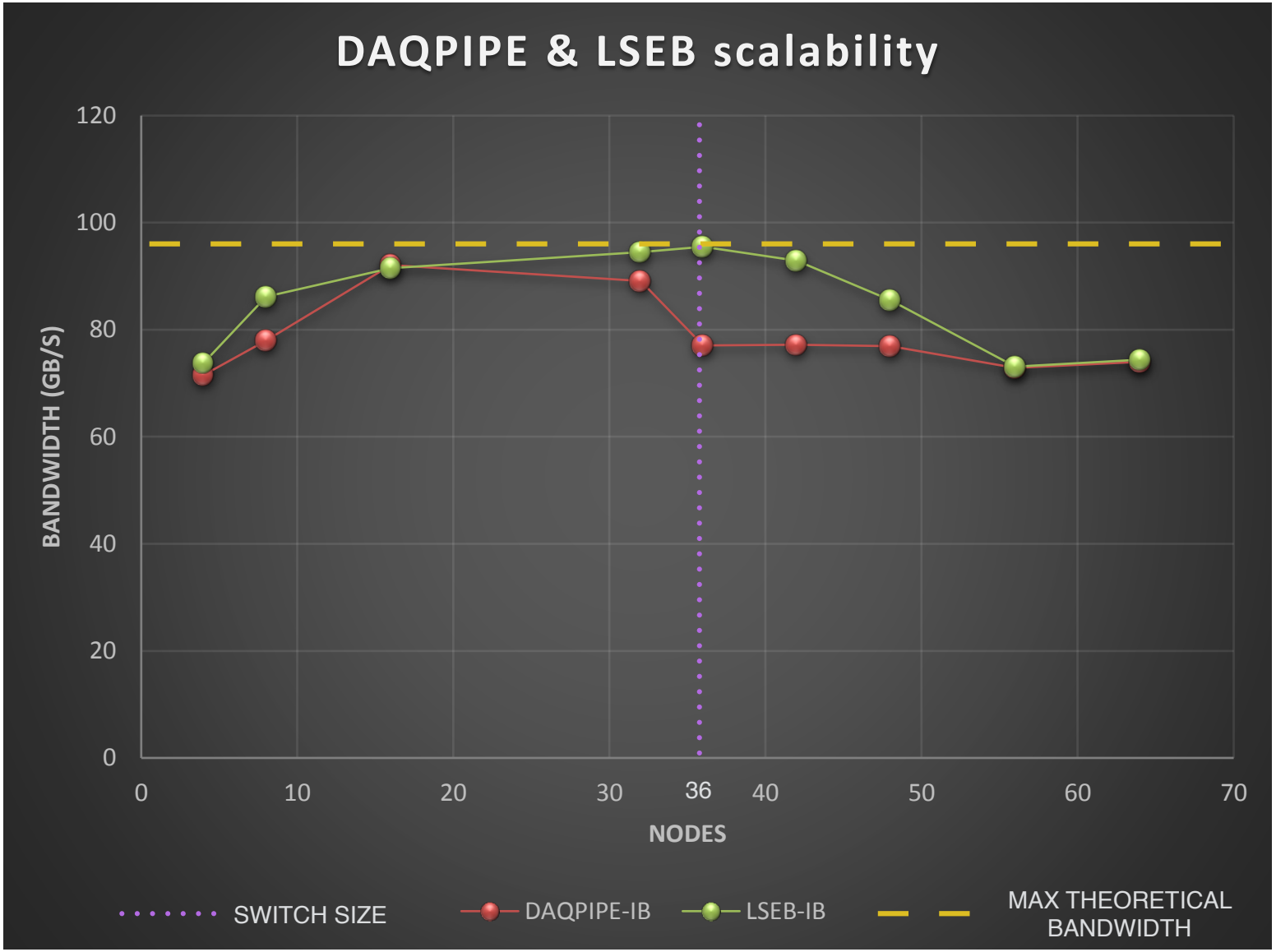
- Several clusters with different IB technologies tested:
  - QDR - 40 Gb/s
  - FDR - 56 Gb/s
  - EDR - 100 Gb/s
  - range of nodes: from 4 to 128
- Most significant test done on a 84-node cluster with IB EDR interconnect:
  - 2 x 18-cores Xeon Haswell E5-2697 v4 processors
  - only 64 nodes available

InfiniBand EDR

- InfiniBand EDR standard:
  - 100 Gb/s (4 lanes, 45 Gb/s each one)
  - 64b/66b encoding → 96.97 Gb/s of max theoretical bandwidth
- Performed benchmark with ib_write_bw tool (OFED package):
  - one-to-one bidirectional test
  - ~ 95 Gb/s of bandwidth
  - saturation with buffers > 32 KB

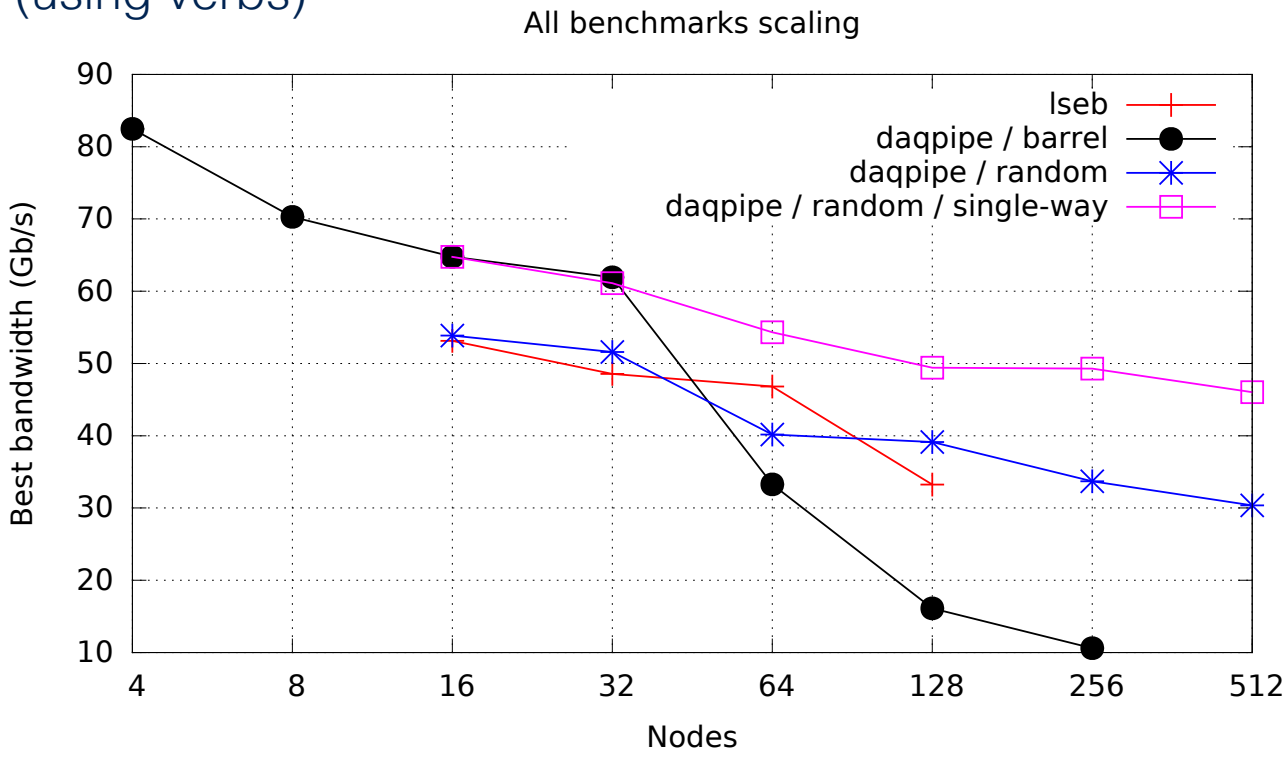


InfiniBand EDR



OmniPath - 500 nodes

- Really quick test on Marconi (A1 partition) before the production phase:
  - no time to perform fine tuning
- Different software interfaces for communication:
  - DAQPIPE (using MPI)
  - LSEB (using verbs)

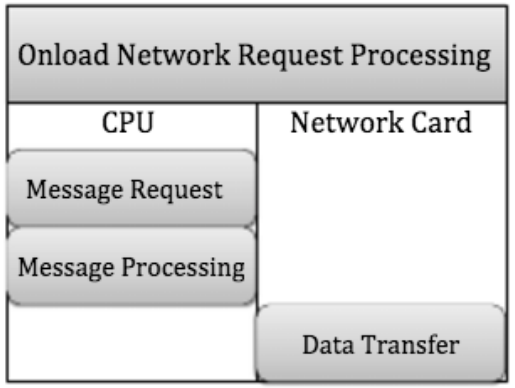


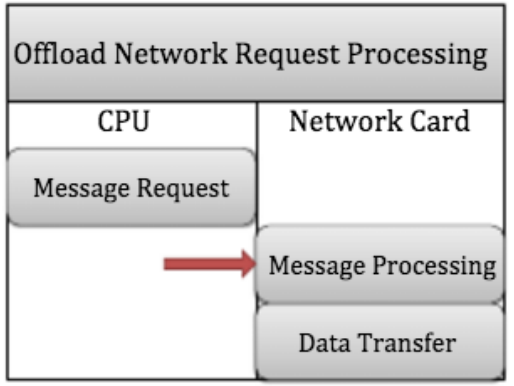
Near future

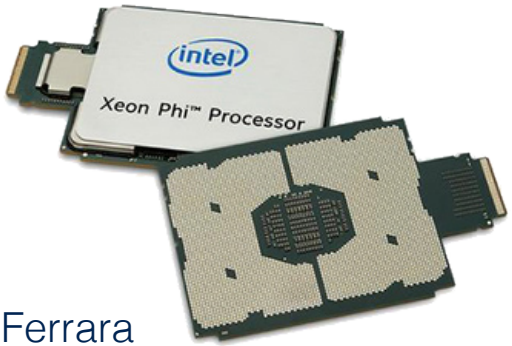
- Scalability tests:
  - run on large cluster with EDR interconnect (LENOX of LENOVO)
  - second run on Marconi cluster with OmniPath interconnect
- Study of 100Gb ethernet with RDMA support (iWARP and RoCE):
  - small testbed at CERN

Near future

- The choice of the CPU technology can be affected by the used interconnect:
  - OmniPath → onload approach
  - InfiniBand → offload approach
- Less-power processors with offload approach?
- Data processing during the event-building?
- Intel Xeon Phi with integrated OmniPath on SOC:
  - how much can it improve the performance?








Thank
You