

FLIT-level InfiniBand network simulations of the DAQ system of the LHCb experiment for Run-3

Tommaso Colombo*, Paolo Durante*, Domenico Galli^{†‡}, Matteo Manzali[‡], Umberto Marconi[‡], Niko Neufeld*, Flavio Pisani^{*†‡}, Rainer Schwemmer*, Sbastien Valat*

*CERN, Geneva, Switzerland

[†]Alma Mater Studiorum - Universit di Bologna, Bologna, Italy

[‡]Istituto Nazionale di Fisica Nucleare - INFN, Sez. di Bologna, Bologna, Italy

Email flavio.pisani@cern.ch

Abstract—The Large Hadron Collider beauty (LHCb) experiment is designed to study differences between particles and anti-particles as well as very rare decays in the charm and beauty sector at the Large Hadron Collider (LHC). The detector will be upgraded in 2019 and a new trigger-less readout system will be implemented in order to significantly increase its efficiency and take advantage of the increased machine luminosity. In the upgraded system, both event building and event filtering will be performed in software for all the data produced in every bunch-crossing of the LHC. In order to transport the full data rate of 32 Tb/s we will use custom FPGA readout boards (PCIe40) and state of the art off-the-shelf network technologies. The full event building system will require around 500 nodes interconnected together. From a networking point of view, event building traffic has an all-to-all pattern, therefore it tends to create high network congestion. In order to maximize the link utilization different techniques can be adopted in various areas like traffic shaping, network topology and routing optimization. The size of the system makes it very difficult to test at production scale, before the actual procurement. We resort therefore to network simulations as a powerful tool for finding the optimal configuration. We will present an accurate low level description of an InfiniBand based network with event building like traffic. We will show comparison between simulated and real systems and how changes in the input parameters affect performances.

I. INTRODUCTION

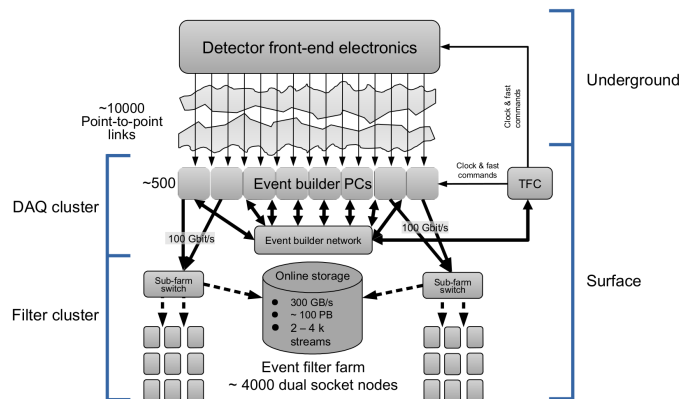


Fig. 1: The architecture of the upgraded LHCb readout system.

The LHCb experiment [1] will receive a substantial upgrade [2] during the Long Shutdown 2 (LS2) of the LHC. One of the major changes during this upgrade process will be the

installation of a completely new DAQ system without any low level hardware trigger, providing higher trigger yield at the luminosity foreseen after LS2. To implement a trigger-less readout, the full bandwidth of ~ 32 Tb/s produced by the detector must be forwarded by the event building network, in order to achieve this total throughput we are targeting a system composed of ~ 500 nodes interconnected together using 100 Gb/s networking technology, as shown in Fig. 1.

In order to design and build a system with the above mentioned complexity we need extensive planning and testing, for this reason we developed DAQ Protocol-Independent Performance Evaluator (DAQPIPE). This software generates real event building traffic and can be configured in multiple ways in order to experiment with different network configurations and technologies. By only using DAQPIPE, in order to test the scalability of the system, we need to access to High Performance Computing (HPC) clusters equipped with 100 Gb/s capable interconnection networks. Because of the relative small number of suitable systems available in world, the waiting time can be very long and the network configuration may be suboptimal for event building tests.

In this work, we present a low level simulation model that can be used, in parallel with tests on real systems, to speed up the process of designing the event building network for a trigger-less readout system.

II. LHCb EVENT BUILDER ARCHITECTURE

In this section, we briefly describe the DAQ's architecture of the LHCb experiment for the Run-3 of the LHC, because a full view is out of the scope of this paper we will focus on the network side of the system, a comprehensive view can be found in the Technical Design Report (TDR) [2].

A. Event building architecture

The LHCb event building is composed of three main logical units:

- **Builder Unit (BU)** receives and aggregates the fragments into full events
- **Readout Unit (RU)** collects the fragments from the DAQ board and sends them to the BUs
- **Event Manager (EM)** assigns which event is built on which BU

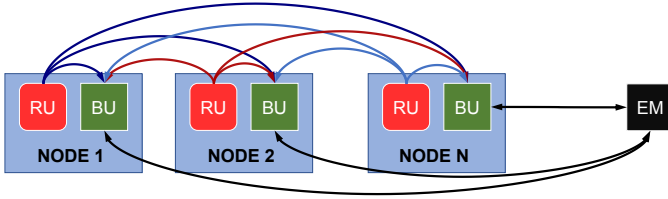


Fig. 2: Event building architecture. The different arrows represent the multiple fragments gathered by the BU while the black ones the control messages to and from the EM.

As depicted in Fig. 2 a BU and a RU are aggregated into one single node generating a ‘folded’ event builder, because the data traffic is always flowing from the RUs to the BUs, this architecture is used to fully exploit the full-duplex nature of the network and to reduce by a factor two the number of physical machines needed in the final system compared to a one-directional event builder.

In the collective communication schema the traffic pattern of a folded event builder can be compared to an all-to-all with different data size for every fragment.

In order to reduce the network congestion, generated by an all-to-all personalized exchange, we use the *linear shifting* traffic scheduling technique, which can be explained as follows:

- We divide the all-to-all exchange into N phases, where N is the total number of nodes
- In every phase every node sends data to one destination and receives from one source
- During phase n node i sends to node $(n + i) \% N^1$

If the aforementioned conditions are respected for all the phases then we have a linear shifting scheduling. In a real world scenario a mechanism for defining phases and synchronizing all the nodes must be provided.

B. Event building network

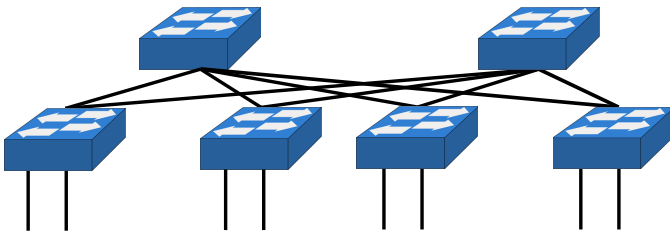


Fig. 3: Fat-tree network build using switches with a radix of four. The two switches in the upper part are called spine switches, while the four in the lower part are called leaf switches.

From the networking point of view, the event building traffic tends to create congestion and high link utilization among all the nodes, therefore the selected network topology has to be non-blocking and provide full bisection bandwidth.

For the implementation of the LHCb event building network, we decided to use a folded Clos network as the one

¹The % symbol indicates the modulo operation

depicted in Fig. 3; often referred to as fat-tree². We selected this particular topology because: it fulfils the aforementioned requirements; it is widely adopted and it is supported by switch vendors. In particular, the OpenSM subnet manager used in InfiniBand-based networks provides optimized routing for fat-tree topologies [3]. This algorithm uses a constant one-to-one correspondence between the spine switch selected and the switch port used by the destination node. This particular routing algorithm provides a conflict-free path for all the packets that are following a perfect linear shifter.

C. Event building benchmark: DAQPIPE

DAQPIPE [4][5][6] is a small benchmark application to test network fabrics for the future LHCb upgrade. It emulates an event builder based on a local area network and it supports multiple network technologies through different communication libraries like: MPI, LIBFABRIC, VERBS and PSM2.

DAQPIPE can be used either in a PUSH or PULL schema and it supports different traffic shaping strategies to reduce network congestion. Technologies and protocols can be mixed in a plug-and-play way.

The software provides an implementation of all the logical blocks required by the LHCb event building and emulates reading data from a real DAQ board connected to the detector. All the fragments of the same emulated event are then sent through the network using the desired communication library and protocol, and then aggregated into the BU selected by the EM.

In order to take advantage of the available bandwidth and reduce the CPU overhead, DAQPIPE sends multiple fragments of multiple events in parallel. The number of fragments in flight and the number of events processed in parallel can be tuned via two parameters:

- **Credits:** number of events processed in parallel by the BU
- **Parallel sends:** number of fragments of the same event in flight

In order to reduce the traffic congestion DAQPIPE provides a barrel shift-like traffic shaping, without enforcing strong synchronization among the nodes³.

III. SIMULATION MODEL

The simulations model we developed is implemented using the Objective Modular Network Testbed in C++ (OMNeT++) framework [7]; this discrete event simulator primarily targets network simulations and offers multiple tools that can be used to accomplish different tasks: from describing the network topology to gathering advanced statistics from the simulated design. In order to simulate the LHCb DAQ system, we mainly need two components: an accurate description of the network and a precise modelling of the DAQ traffic.

²From a rigorous point of view the network topology shown in Fig. 3 is a folded Clos network, nevertheless in the industry and data center world, it is frequently referred to fat-tree. Even if the network topologies are not exactly the same from this point on we will use the industry standard naming ‘fat-tree’ instead of ‘folded Clos’.

³There is a version of DAQPIPE with enforced timing but it will not be considered for the purpose of this paper

Mellanox technologies has already contributed to an OMNeT++ based InfiniBand Flow Control Unit (FLIT) level simulation model. This model supports: link level flow control, static lookup-table-based routing, arbitration between multiple Virtual Lanes (VLs)⁴, packet generation and fragmentation and packet arbitration; however, it is not updated and does not support the 100 Gb/s flavour of InfiniBand (i.e. EDR). Therefore we decided to expand the library capabilities to fulfil our requirements and to make it as accurate as possible. In order to obtain a realistic model behaviour, we performed a fine tuning of the parameters using information collected from real hardware available in our test laboratory. In particular, we focused on: buffer sizes, network latency, link flow control, packet arbitration, latency and jitter of our entire software stack including PCIe communication overheads.

A. Modules Description

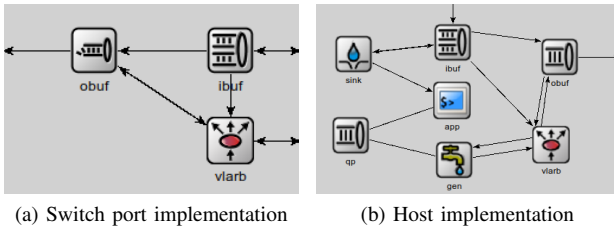


Fig. 4: Internal structure of a switch port and an host

OMNeT++ uses *modules* as fundamental building blocks, hereinafter we provide a brief description of the main ones implemented:

- **IBOutBuf:** buffer for outgoing FLITs
- **IBInBuf:** buffer for incoming FLITs
- **IBVLArb:** it implements arbitration among the different VLs
- **PktFwdIfc:** it provides destination ports to packets according to the static routing table
- **SwitchPort:** it combines input and output buffers with the VL arbitration logic
- **IBApp:** it generates messages according to the selected traffic pattern
- **IBWorkQueue:** queue for the different message coming from one or more applications
- **IBGenerator:** it arbitrates all the work queues and generates the packets and the FLITs accordingly
- **IBSink:** it receives the packets and notifies the IBApp module

Fig. 4 depicts how module can be interconnected together and generate more complex units.

B. Topologies

In order to implement network topologies, OMNeT++ provides the Network Description (NED) language which can

⁴A VL is the InfiniBand implementation of a Virtual Channel [8] - i.e. a set of multiple flow control independent channels multiplexed on to the same physical one -

be used to generate hierarchical and parametric networks. By using this powerful and flexible tool we implemented a parametric description of a fat-tree network. In view of analysing and comparing against real data collected on HPC clusters we also implemented a Python script that generates NED code by parsing the subnet manager information of the real cluster topology. In this way, we can study ideal topologies and compare them against real world systems with small imperfections like: missing nodes, swapped cables and suboptimal routing.

C. Traffic injectors

Accurate traffic modelling is a key component for obtaining precise and realistic network simulation; in particular, in this work we used both synthetic and real application traffic. Our main target is to simulate the event building system of the LHCb experiment, therefore a particular effort was put in an accurate replication of the DAQPIPE traffic. Moreover we implemented two linear shifters with a different phase definition. A list and a brief description of the traffic injector implemented follows:

- **Fixed-size linear shifter:** it shifts destination after a fixed-size injection.
- **Time-window linear shifter:** it shifts destination after a fixed time interval. This injector uses a fixed grace period to absorb jitter, during this period the nodes are not allowed to send data, resulting in increased stability at the expense of a lower theoretical throughput.
- **DAQPIPE:** an injector that replicates the real DAQPIPE traffic. This traffic generator allows the user to change all the relevant parameters as in the real software.

IV. PARAMETER TUNING

The simulation model has several different parameters that need to be tuned and optimized to replicate the behaviour of real InfiniBand systems. For our event building studies we are interested in 100 Gb/s networking solutions, therefore we tuned the model to replicate InfiniBand EDR hardware. In particular we use a Mellanox SB7700 EDR switch and Mellanox ConnectX-5 Host Channel Adapters (HCAs).

Most of the basic parameters can be extracted from the InfiniBand architecture specification [9], e.g.: real bandwidth, header overhead, encoding overhead, link flow control behaviour, ecc. Advanced and hardware specific ones can be estimated performing real measurements and reverse engineering on the actual hardware.

Crucial values for our simulations are: switch buffer size, link layer latency and PCIe latency.

A. Switch buffer estimation

In order to measure the switch buffer size we can use two different techniques [10]: analysing the link level flow control packets or generating congestion and monitoring the congestion indicator⁵ on the various ports.

⁵The congestion indicator is the PortXmitWait counter which indicates the time, expressed in clock ticks, that a given port has been idling because of insufficient credits on the receiving buffer

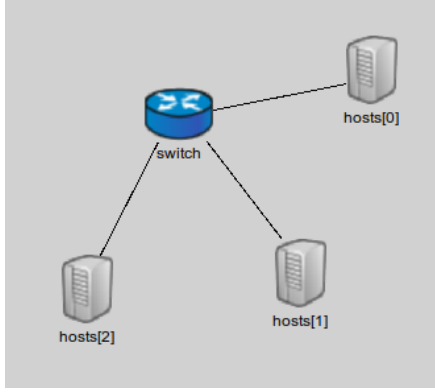


Fig. 5: Setup used to generate congestion and estimate the switch buffer size. host0 sends at full speed data to host2, at the same time host1 sends packets of different sizes to create controlled congestion.

Decoding the information from the flow control packets produces a more accurate measure, but it requires a low-level InfiniBand protocol analyser. Because there are no EDR capable protocol analysers available on the market, we decided to use the second strategy, and estimating the amount of buffering available in every switch port by measuring the performance counters.

The setup used is depicted in Fig. 5 and the procedure used to create congestion is the following:

- host0 sends continuously to host2 at full speed
- host1 sends to host2 packets of increasing size at regular intervals, to create congestion
- by reading the PortXmitWait counter and knowing the packet size we can estimate the buffer size of the switch

Following this procedure we estimated a buffer size of 64 KiB per port per VL with 4 VLs enabled.

B. Link layer latency estimation

In order to measure the link layer latency, without using external protocol analysers, we decided to use the hardware timestamping feature of the IEEE 1588-2008 standard - i.e. Precision Time Protocol (PTP) - implementation in the Mellanox HCAs.

The path latency measure using PTP produced an estimation of 170 ns full delay using a direct attached copper cable, between two directly connected hosts.

C. PCIe latency modelling

The final piece in our model tuning is a realistic model PCIe and InfiniBand software stack latency; because of the non real time nature of modern computing systems and software, we decided to perform real world latency measures and replicate this behaviour in our simulation model.

The latency has been measured using the `ib_write_lat` benchmark and subtracting the link layer latency, therefore this measurement include all the time needed from the hardware and software chain to make a packet available to the link layer.

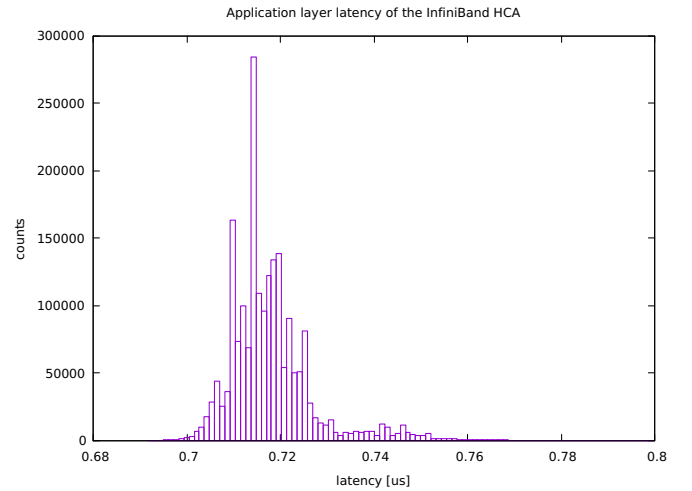


Fig. 6: Application and PCIe latency of an InfiniBand EDR HCA

Fig. 6 shows the histogram of the latency measurements, the simulation model generates random number generates from this distribution to replicate latency and jitter of the real system.

V. RESULTS

In this section we present some results obtained by simulating DAQPIPE with the aforementioned simulation model. In particular we provide a comparison between the simulation and real data and a comparison of two different network topologies. Fig. 7 shows a comparison between the simulated

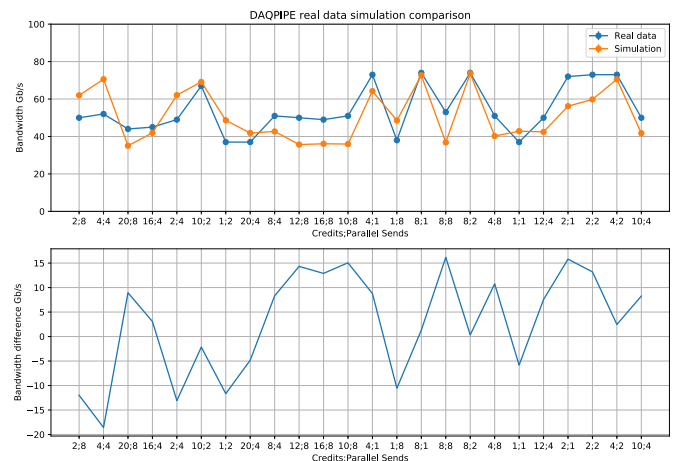


Fig. 7: Comparison between real and simulated DAQPIPE on a real HPC cluster topology of 64 nodes

and the real DAQPIPE for different values of the `credits` and `parallel_sends` parameters. The real data are collected on an HPC cluster of 64 nodes interconnected via a fat-tree-like network with: missing nodes, swapped cables and non-ideal routing. The simulation uses a replica of the same topology and the same routing of the real system.

From this plot we can confirm that the simulation can replicate the trend and the absolute value of the measurements performed on the real system.

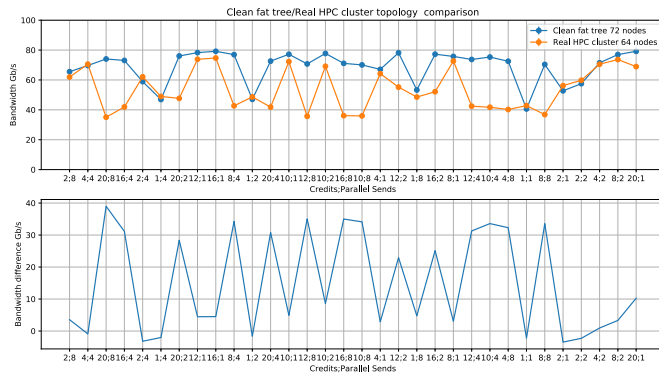


Fig. 8: Comparison between simulated DAQPIPE on a real HPC cluster topology of 64 nodes and on a fat tree of 72 nodes

In Fig. 8 we present a comparison of the performances of the simulated DAQPIPE on two different topologies: a clean fat tree of 72 nodes and the real HPC cluster of 64 nodes.

As we can see from the plot the performance loss is highly dependant on the parameters and can be as high as 40 Gb/s, nevertheless the bandwidth drop for the best configuration is ~ 5 Gb/s.

We can conclude that a non ideal topology affects the performances of DAQPIPE and makes it more unstable, the performance drop can vary significantly and it is highly influenced by the configuration parameters and the topology itself.

VI. CONCLUSIONS AND FUTURE WORK

We have implemented an accurate low level model of our event building traffic based on the InfiniBand EDR fabric. We have tuned the model to achieve realistic results.

We have validated our model against real data obtained on an HPC cluster and we measured the impact of non ideal fat-tree topologies.

We will run an extensive simulation campaign to evaluate the scalability of the system up to the required of ~ 500 nodes.

REFERENCES

- [1] The LHCb Collaboration et al, “The LHCb detector at the LHC,” *Journal of Instrumentation*, vol. 3, no. 08, S08005, 2008.
- [2] “LHCb Trigger and Online Upgrade Technical Design Report,” Tech. Rep. CERN-LHCC-2014-016. LHCb-TDR-016, 2014. [Online]. Available: <https://cds.cern.ch/record/1701361>.

- [3] Z. Eitan, J. Gregory, K. D. J., and L. Michael, “Optimized infinibandtm fattree routing for shift alltoall communication patterns,” *Concurrency and Computation: Practice and Experience*, vol. 22, no. 2, pp. 217–231, DOI: 10.1002/cpe.1527. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.1527>. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.1527>.
- [4] A. Otto, D. H. C. Prez, N. Neufeld, R. Schwemmer, and F. Pisani, “A first look at 100 gbps lan technologies, with an emphasis on future daq applications.,” *Journal of Physics: Conference Series*, vol. 664, no. 5, p. 052 030, 2015. [Online]. Available: <http://stacks.iop.org/1742-6596/664/i=5/a=052030>.
- [5] D. H. C. Prez, R. Schwemmer, and N. Neufeld, “Protocol-independent event building evaluator for the lhcb daq system,” *IEEE Transactions on Nuclear Science*, vol. 62, no. 3, pp. 1110–1114, 2015, ISSN: 0018-9499. DOI: 10.1109/TNS.2015.2428891.
- [6] F. Pisani, D. H. C. Prez, and N. Neufeld, “High-speed zero-copy data transfer for daq applications,” *Journal of Physics: Conference Series*, vol. 608, no. 1, p. 012 029, 2015. [Online]. Available: <http://stacks.iop.org/1742-6596/608/i=1/a=012029>.
- [7] A. Varga and R. Hornig, “An overview of the omnet++ simulation environment,” in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, ser. Simutools ’08, Marseille, France: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, 60:1–60:10, ISBN: 978-963-9799-20-2. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1416222.1416290>.
- [8] J. Duato, S. Yalamanchili, and N. Lionel, *Interconnection networks: An engineering approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, ISBN: 1558608524.
- [9] InfiniBand SM Trade Association., *Infiniband architecture specification volume 1 and 2*. 2015. [Online]. Available: http://www.infinibandta.org/content/pages.php?pg=technology_public_specification.
- [10] Q. Liu, *Analyzing InfiniBand Packets*. [Online]. Available: https://www.openfabrics.org/images/eventpresos/workshops2015/UGWorkshop/Thursday/thursday_09.pdf.