An Evaluation of 100-Gb/s LAN Networks for the LHCb DAQ Upgrade

Sébastien Valat, Balázs Vőneki, Niko Neufeld, Jonathan Machen, Rainer Schwemmer, and Daniel Hugo Cámpora Pérez

Abstract—**The Large Hadron Collider Beauty experiment (LHCb) experiment is preparing a major upgrade resulting in the need for a high-end network for the data acquisition system. Its capacity will grow up to a target speed of 40 Tb/s, aggregated by 500 nodes. This can only be achieved reasonably by using links that are capable of coping with 100-Gb/s line rates. The constantly increasing need for more and more bandwidth has initiated the development of commercial 100-Gb/s networks. There are three candidates on the horizon that need to be considered: Intel Omni-Path, 100-G Ethernet, and EDR InfiniBand. We present test results with such links using both standard benchmarks (e.g., iperf) and a custom benchmark called Data AcQquisition (DAQ) Protocol Independent Performance Evaluator (DAQPIPE). With DAQPIPE, we mainly evaluate the ability to exploit the targeted network for a kind of all-to-all communication pattern. The key benefit of these measurements is that it helps us to tune our benchmark and improves our understanding of the relevant parameters. It will now permit us to prepare and motivate some upcoming tests at scale on existing supercomputers offering the targeted hardware.**

Index Terms—**Benchmark, data acquisition system, detector, Ethernet, high-performance computing (HPC), InfiniBand, MPI, networkds, Omni-Path.**

I. Introduction

THE Large Hadron Collider (LHC) has four big experiments, and one of them is LHCb. It has an underground detector that gathers information from particle collisions at high energies. The designed collision rate is 40 MHz. In order to be able to deal with the large quantities of data generated at the collider, one needs to reject the irrelevant events and keep only events that are interesting to us for later analysis. This procedure is called triggering.

Currently the LHCb operates by applying two levels of triggers: a low-level trigger (LLT) and a high-level trigger (HLT), where the first is realized by field-programmable gate array (FPGA)-based custom hardware, reducing the 40-MHz input rate to 1 MHz. Getting this hardware trigger makes the maintainability very difficult due to usage of custom hardware exposed in the radiation area. After triggering, the data are sent as User Datagram Protocol (UDP) packet generated by FPGAs to aggregate them on PCs. This also adds to the complexity to maintain those FPGA programs. For data aggregation, the UDP packets are sent by burst (synchronously) to the target PC. This requires large buffers in the switches to handle the traffic congestion, making us dependent on costly switches. After getting the data, the HLT can start to partially build the event and select the interesting one for long-term storage.

The LHCb experiment will be upgraded during the LHC Long Shutdown 2 starting from 2018 to 2019 [2]. One of the key features of this upgrade is the removal of the LLT. Hence, it will end with a trigger-free readout and fully software-driven trigger. This improvement increases the event-rate from 1 to 40-MHz introducing a total throughput close to 40 Tb/s (zero suppressed) to be handled by the readout nodes located at point 8 (the LHCb site) at the surface. Then the data will be sent to a filter farm that will be placed at point 8, close to the readout nodes or in a shared place with another experiment on the Prevessin site. It was also decided to use as much as possible commercial off-the-shelf hardware for the data transmission. To lower switches price we are also looking to make the buffering inside standard PCs.

On the network hardware side, over the past years, the high-speed fabrics have greatly been improved for the high-performance computing (HPC) applications and now provides 100-Gb/s network hardware. For the next LHCb upgrade, those network technologies will permit to set up an event building cluster of 500 nodes to read and aggregate the 40 Tb/s of data going out of the detector. Hence we consider the ability to use 80% of the link speed to proceed. Remark that the communication pattern is all-to-all which is likely to encounter some issues while scaling on a large number of nodes.

This paper will shortly describe the network communication that will be needed on those 500 nodes. Then, we will discuss the benchmarking results we had, going from simple available bandwidth benchmark to more realistic custom-made benchmarks. This analysis will end up with a full-running test on 14 nodes equipped with 100-Gb/s InfiniBand EDR boards.

II. 100-Gb/s Technologies Overview

The LHCb experiment is considering three different 100-Gb/s technologies for the upgrade [3].

A. 100-Gb/s Ethernet

The technology was first defined by IEEE 802.36ba-2010 [8]. We had the privilege to test several early releases.

Manuscript received June 24, 2016; revised November 13, 2016 and January 26, 2017; accepted March 10, 2017. Date of publication March 24, 2017; date of current version June 26, 2017.

S. Valat, B. Vőneki, N. Neufeld, and R. Schwemmer are with CERN, 1217 Meyrin, Switzerland (e-mail: sebastien.valat@cern.ch).

J. Machen is with DCG IPAG, EU Exascale Labs, Intel Semiconductor AG, INTEL, 8048 Zürich, Switzerland.

D. H. Cámpora Pérez is with the Universidad de Seville, 41004 Seville, Spain.

Digital Object Identifier 10.1109/TNS.2017.2687124

It is an interesting solution for both event building as well as the filter farm. From the user point of view it is used through the widely used Ethernet stack. It comes with all the advantages of Ethernet (reliability, disconnection detection, etc.). But, due to the presence of the kernel in the communication path and intermediate copies, it induces higher CPU usage.

B. EDR InfiniBand

InfiniBand is one of the most used interconnect for large supercomputers for several years now. EDR is an acronym for Enhanced Data Rate. It is the next stage on the InfiniBand roadmap after fourteen data rate (FDR) which is currently used in 28% of existing supercomputers in 2016 [4]. EDR increases capable bandwidth from 38 to 100 Gb/s with latency lower than 500 ns. An EDR link consists of four lanes, each able to transfer 25 Gb/s. The latest generation is PCIe 3 compliant. Compared to Ethernet, InfiniBand provides data transfers directly to/from the remote memory without using intermediate buffers nor required kernel actions. This remote direct memory access (RDMA) is now widely used in the HPC field to bypass the kernel performance limitation. It reduces the CPU usage and handles communications with extremely low latency. For the programmer, InfiniBand is used through the VERBS API [9]. One can remark that InfiniBand networks are already in use into the current data acquisition system of compact muon solenoid experiment [5].

C. Intel Omni-Path

The Intel Omni-Path is a new high-speed interconnect technology optimized for HPC deployments. It is supposed to offer 100-Gb/s bandwidth and a 56% gain in lower latency over InfiniBand fabrics. As for InfiniBand, it provides an RDMA protocol to limit the CPU usage. For developers, Intel Omni-Path provides a VERBS support but its native usage is based on the new PSM2 [10] interface inherited from QLogic, or the new vendor common effort: libfabric [11]. In the next generations, Intel Omni-Path will also be embedded directly into Intel future processors, possibly leading to cost, space reduction, and performance improvement.

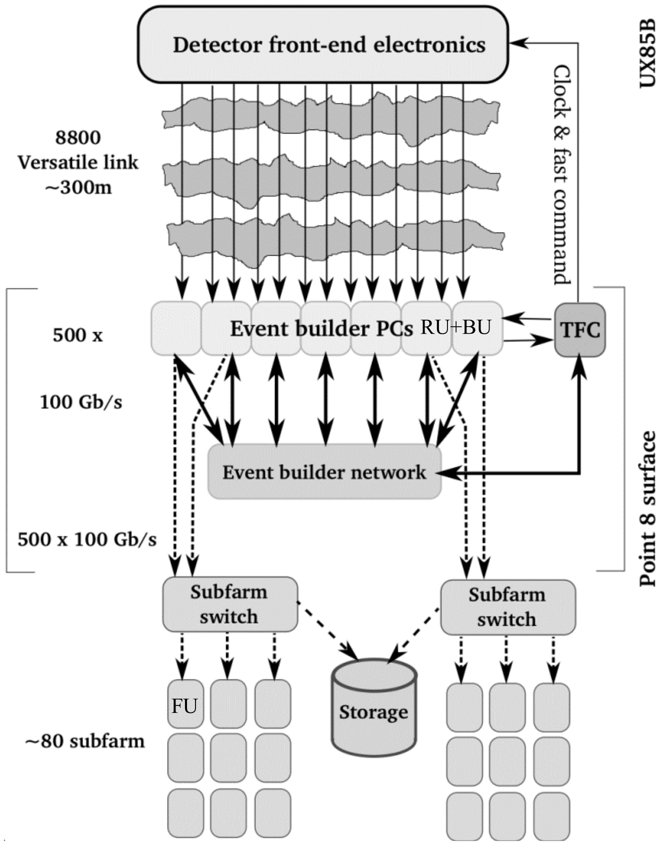


Fig. 1. Architecture of the upgraded LHCb readout system.

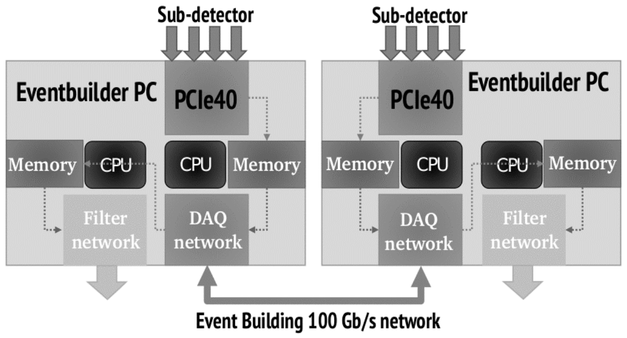


Fig. 2. Dataflow per event builder node.

III. Event Building and Data Flow

For the next LHCb upgrade, we will set up an event building cluster of 500 nodes to read and aggregate the 40 Tb/s of data going out of the detector. As shown in Fig. 1, the data will arrive from the subdetectors through 8,800 optical links running from the LHCb cavern to the surface. Those fibers will be connected to dedicated PCIe Gen3 boards based on FPGA with up to 48 links and 100 Gb/s per board [6]. Using standard servers to host those readout units permits to manage easily the buffering thanks to the large available memory. It also eases the event building (aggregation of the subdata parts) by using 100-Gb/s standard fabrics from the HPC field.

Once the data have been aggregated by the builder units (BUs) they have to be sent to one of the 3,500 filter units (FU) from the filter fam onto a second network, still at 100 Gb/s not to accumulate data inside the node. Those FU will apply the software triggering rules to decide whether the event must be sent to long-term storage or not.

We decided to host the Readout and BUs on the same host in order to reduce the cost of the event building farm by requiring fewer nodes. But, it induces two stressing conditions for the hardware. First, it requires an inner memory throughput up to 400 Gb/s for each node (see Fig. 2) to handle the data movements. This last requirement seems to be achievable considering the already available hardware. This is shown in Fig. 2 by using the stream benchmark [7] (see Fig. 3) on a dual socket Intel Xeon E5-2690 server with a total bandwidth of 600 Gb/s. This makes a huge advantage for the RDMA-based networks as it avoids internal memory copies inside the kernel, avoiding to make us running at the limit. It might let rooms for making some data reshaping inside the BU. Second, each node

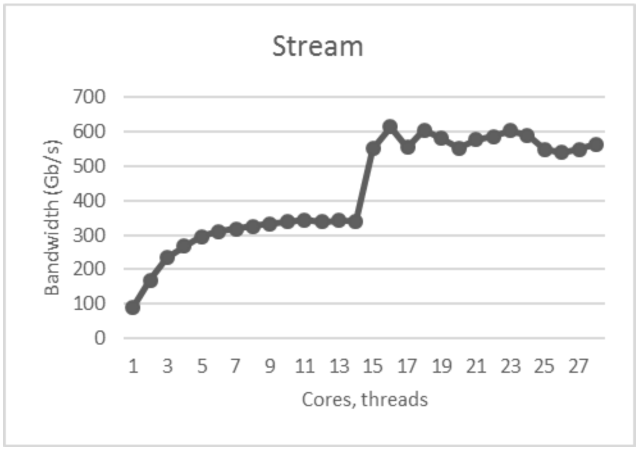


Fig. 3. Stream benchmark on a dual socket server with Xeon E5-2690.

will receive and send data on the event building network, hence we need to achieve the required bandwidth in full duplex. The event building communication pattern can also be compared to a kind of all-to-all communications which certainly stress the switches. The purpose of our current benchmarking effort is to find possible bottlenecks on this approach.

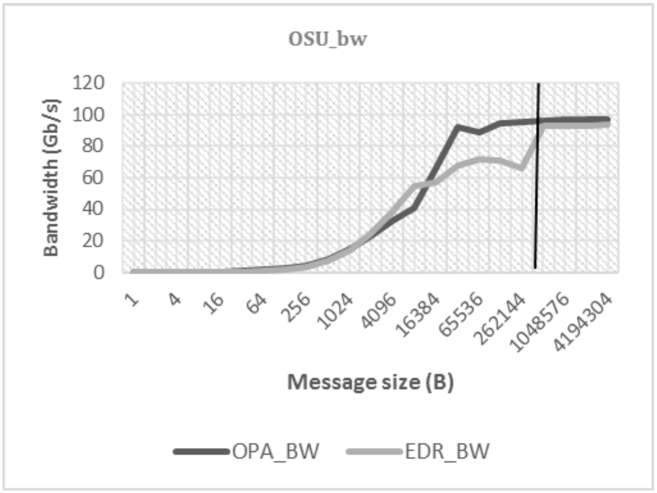
IV. Simple Benchmarks

As a first step we wanted to evaluate the previously mentioned network technologies with simple benchmarks. It gives some reference number about the best bandwidth we can achieve in a trivial case. On HPC networks we can use the Message Passing Interface (MPI) Ohio State University (OSU) [7] benchmark which simply applies a ping-pong communication pattern between nodes to evaluate the bandwidth. This ping-pong can be in one way (OSU_BW) or two ways (OSU_BIBW). The results of Intel Omni-Path and IB EDR are provided in Fig. 4. It shows a full duplex bandwidth of 160 Gb/s on Intel Omni-Path and 120 Gb/s on InfiniBand. It shows us that we are already close the 80% link usage for this simple communication pattern. It also shows us that using segments larger than 512 kB is required to achieve the required bandwidth.

In the same idea we can also evaluate the 100-Gb/s Ethernet solution by using Intel Boulder Rapids cards. We can evaluate those cards with the iperf2 benchmark as the OSU is based on MPI which is available but not optimized for Ethernet. The results are shown in Fig. 5 and shows the ability to reach a bandwidth of 93 Gb/s with this simple benchmark. But remark that it requires 30 threads fully using the CPUs to reach this bandwidth.

InfiniBand cards from Mellanox can also be set up in Ethernet mode. Hence we can also use iperf2 on them. Running iperf2 on those cards gives a bandwidth of 97 Gb/s which shows that usage of Ethernet might be seriously considered. But it was by fully using the 32 cores of the machine, which shows the major interest of the Omni-Path and InfiniBand technologies which only uses one for the same performance.

The CPU usage could be a limitation for us such that we can only consider Ethernet for one of the two networks otherwise the CPU will be the limitation if we use Ethernet for both.



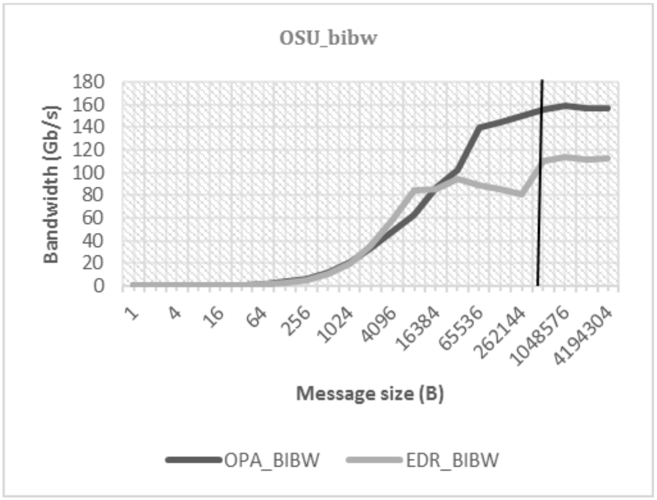


Fig. 4. Simple MPI OSU benchmark between two nodes.

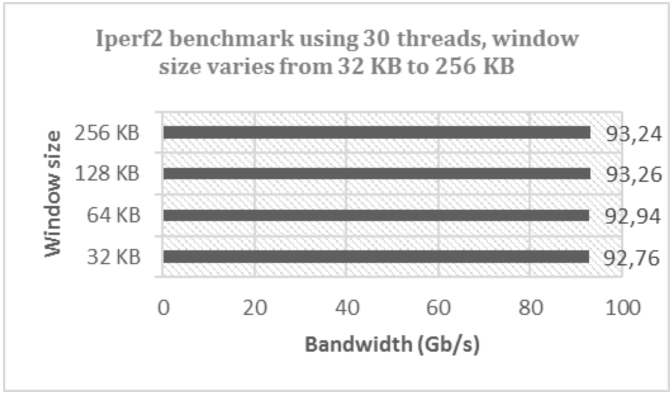


Fig. 5. Iperf test on Intel Ruby Rapids evaluation boards.

Hence we currently favor usage of IB or Intel Omni-Path for the event building due to the full-duplex requirement and Ethernet for the filter farm network to have more flexibility.

One can propose the usage of the data plane development kit (DPDK) [13] framework which aimed at being used for switch implementation. This framework offer a network packet management air position indicator (API) in userspace, bypassing the kernel and working directly with the board if it has the required support. Thanks to such a framework we

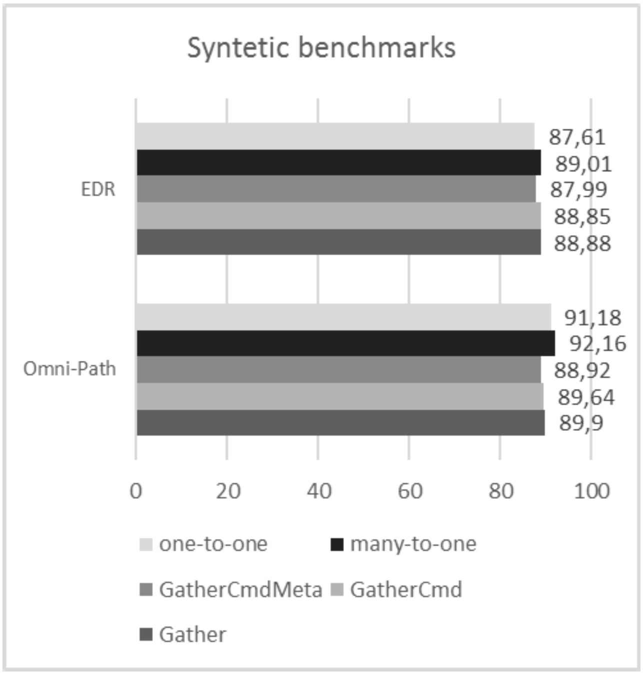


Fig. 6. Exploring the benefits of various communication patterns.

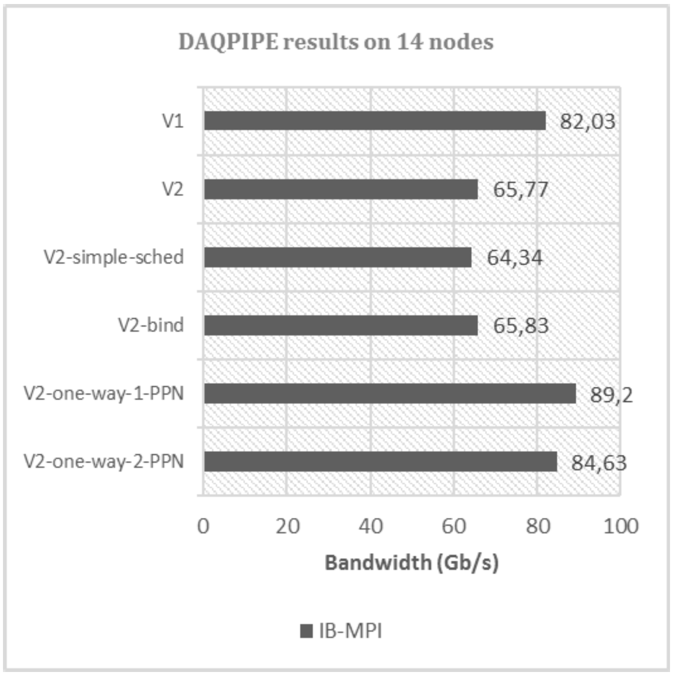


Fig. 7. DAQPIPE results on 14 nodes by running various modes.

can expect a better CPU efficiency by avoinding the memory copies done by the kernel. Altrhough it is interesting for us, it has not yet been studied while writing this paper.

V. Using Synthetic Benchmark

Before moving to a more realistic benchmark, we wanted to explore some synthetic ones to evaluate the potential issues which can arise with our communication pattern. To this end, we implemented three benchmarks to simulate various communication patterns without any synchronization between the communications.

1) *One-to-One:* Each odd process sends packets to the next even process. This is the simplest and ideal pattern we can consider on a large network.
2) *Many-to-One:* This is closer to our real pattern. Every process sends its data to another process (always the same). This is a simple reduction pattern. It might help to detect potential overloading of the network. In this case, we report the bandwidth on the receiving process.
3) *Gather:* This is a simulation of the real pattern without the synchronization. Every process sends packets to all the other processes in round robin. Here we tested different variants. GatherCmd mix large messages (1 MB) with some small command messages (64 B). GatherCmdMeta also add some medium size messages (10 kB) to simulate the meta-data from DAQPIPE. This match with the implementation of the DAQPIPE benchmark described in the next section.

Those benchmarks also permit to handle a controlled number of simultaneous messages. The best results obtained on these benchmarks are given in Fig. 6. It shows that we achieve bandwidth larger than 87 Gb/s in any case with Intel Omni-Path and InfiniBand by selecting the right configuration. Those benchmarks show there is no noticeable effect due to the various message size used by the application. We now can used those parameters (requirement to have multiple on-the-fly communications and message size) to tune to more realistic benchmark. These benchmarks also show that the all-to-all pattern does not appears to be an issue at this scale. But it will be useful to reproduce those results on 500 nodes where we might see some divergence between those benchmarks due possible scalability issues.

VI. DAQPIPE Benchmark

In order to evaluate the real event building solution, we built [1] DAQ Protocol Independent Performance Evaluator (DAQPIPE). This benchmark reproduces the realistic DAQ scenario by implementing the three main units. A readout unit simulating the incoming data and send them on the network. A BU fetching the data from readout unit and aggregating them. An event manager balancing the work between the BUs. It also implements support for various network drivers to compare them with the same software. DAQPIPE also provides various communication scheduling scenarios which are outside of the topic of this article. It has been developed to be easily run on existing supercomputer. This benchmark has been tested on a cluster of 14 InfiniBand EDR nodes in various running modes.

1) *V1:* Version one is an older version of the DAQPIPE benchmark. It mostly has the property of using two processes per node, one to send the data and one to receive.
2) *V2:* Version two uses by default only one process per node making the read and write in parallel in this unique process.
3) *V2-Simple-Sched:* By default the version two uses a round robin scheduler with a limited number of

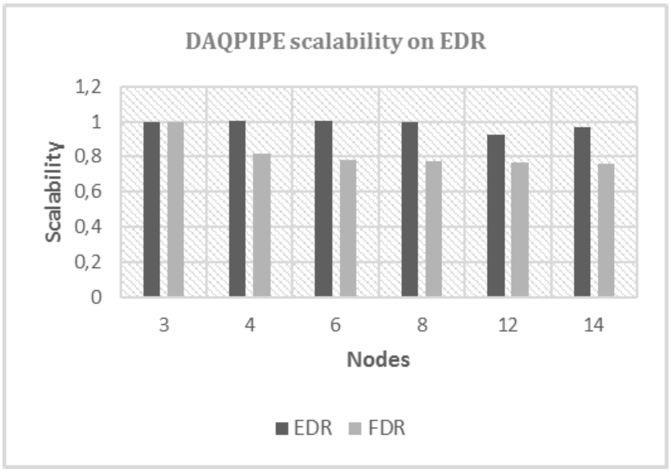


Fig. 8. DAQPIPE scalability on 14 nodes considering 3 nodes as reference.

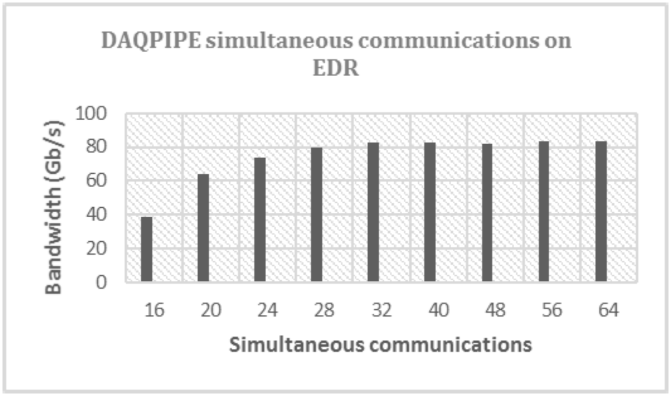


Fig. 9. DAQPIPE effect of simultaneous communications on 14 EDR nodes.

simultaneous communications. A variant is to send all the messages at same time, letting the network handling the scheduling.

4) *V2-Bind:* Another variant consists of binding with hwloc (Portable Hardware Locality) the threads of the running process close to the network board to avoid possible Nonuniform Memory Access (NUMA) effects.
5) *V2-One-Way-1-PPN:* Apply the same running mode that version one with one task (read or write) per process. In this case, we can use one process per node (so one task per node), this make a single way communications on each node. It is used to be compared with the full-duplex approach and determine how much we lose by fully loading the links in two ways.
6) *V2-One-Way-2-PPN:* Same as the previous one by using two processes per nodes, hence we generate a full-duplex communication scheme on each one (equivalent to v1). This shows better performance than the full-duplex approach with a unique process per node. It is close to what we consider for the realistic scenario.

The results are given in Fig. 7 and shows achievement of 84 Gb/s for the most with two task per process. This shows that we need two cores to fully saturate the network.

We can also check the scalability of our current benchmark up to 14 nodes to evaluate possible performance losses due to the switches. Here the scalability is defined as the ratio of the local bandwidth using N nodes over the local bandwidth with 2 nodes. This is shown in Fig. 8. Remark that InfiniBand EDR scale well with our pattern starting from 84 Gb/s on 3 nodes to 81.5 Gb/s on 14 nodes. This makes a loss of only 3%. This contrasts with FDR. As seen in Fig. 8, FDR has a performance drop on 4 nodes going down from 20 to 15.3 Gb/s which is a 25% loss. Of course the open question is now how it scales beyond 16 nodes.

Some internal operations are required between two communications. This introduces some latency reducing the achieved bandwidth. In theory it is interesting in such a design to manage many communications simultaneously to recover those latencies. Hence, DAQPIPE can handle such a parameter and can produce the results of Fig. 9 by always having multiple pending communications on the network. It shows that we gain a lot by using at least 28 simultaneous communications per process on 16 nodes.

VII. Conclusion

As explained, for the next LHCb major upgrade we will need to handle 40 Tb/s going out of the detectors and to be handled and aggregated over 500 nodes using an HPC fabric. We have seen the capability to reach 95 Gb/s with some simple benchmarks. Then we showed the capability to reach around 88 Gb/s with some synthetic benchmarks. We finally proved that we can achieve good bandwidth usage up to 14 nodes with 85 Gb/s per process with the realistic benchmark. This fulfills our expectation to exploit 80% of the link bandwidth. However, we also showed that the Ethernet approach requires 30 cores to achieve g The CPU usage could be a limitation for us such that we can only consider Ethernet for one of the two networks otherwise the CPU will be the limitation if we use Ethernet for both. Hence we currently favor usage of IB or Intel Omni-Path for the event building due to the full-duplex requirement and Ethernet for the filter farm network to have more flexibility.

One can propose the usage of the DPDK [13] framework which aimed at being used for switch implementation. This framework offer a network packet management API in userspace, bypassing the kernel and working directly with the board if it has the required support. This might provide *good* bandwidth which has to be compared to InfiniBand and Intel Omni-Path which requires only one or two core for similar performances.

VIII. Future Work

The next step will be to evaluate our benchmark at a larger scale on existing HPC clusters up to 500 nodes to pinpoint potential issues when going through larger and two level switches. Tunings are ongoing with DAQPIPE on the Intel Omni-Path architecture; hence, we need to obtain similar results than IB at small scale before running on larger scale. We are also looking for failure recovery support to handle failure of nodes during data taking which is affordable if we use the low-level API (verbs, psm2) but not the standard MPI API. Here we notice that we at least need to manage the failure mitigation to continue running if a node dies to flush all the buffers in the pipeline. The second step is to support the reconnection of a failed node. A final point will be to test the usage of DPDK framework for the Ethernet mode to possibly reduce the CPU usage.

REFERENCES

[1] D. H. C. Pérez, R. Schwemmer, and N. Neufeld, "Protocol-independent event building evaluator for the LHCb DAQ system," *IEEE Trans. Nucl. Sci.*, vol. 62, no. 3, pp. 1110–1114, Jun. 2015.

[2] The LHCb Collaboration *et al.*, "LHCb trigger and online upgrade technical design report," CERN, Meyrin, Switzerland, Tech. Rep. CERN-LHCC-2014-016 and LHCB-TDR-016, 2014. [Online]. Available: https://cds.cern.ch/record/1701361

[3] A. Otto, D. H. C. Pérez, N. Neufeld, R. Schwemmer, and F. Pisani, "A first look at 100 Gbps LAN technologies, with an emphasis on future DAQ applications," in *Proc. 21st Int. Conf. Comput. High Energy Nucl. Phys.*, 2015, p. 052030.

[4] J. Dongarra, "Performance of various computers using standard linear equations software," Dept. Comput. Sci., Univ. Tennessee, Knoxville, TN, USA, Tech. Rep. CS-89-85, 2014.

[5] T. Bawej *et al.*, "The new CMS DAQ system for run-2 of the LHC," in *Proc. 19th IEEE-NPSS Real Time Conf. (RT)*, Nara, Japan, May 2014, p. 1.

[6] P. Durante, N. Neufeld, R. Schwemmer, U. Marconi, G. Balbi, and I. Lax, "100 Gbps PCI-express readout for the LHCb upgrade," *IEEE Trans. Nucl. Sci.*, vol. 62, no. 4, pp. 1752–1757, Aug. 2015.

[7] J. D. McCalpin, "Memory bandwidth and machine balance in high performance computers," *IEEE Comput. Soc. Tech. Committee Comput. Archit. Newslett.*, Dec. 1995. [Online]. Available: https://www.researchgate.net/publication/51992086_Memory_bandwidth_and_machine_balance_in_high_performance_computers

[8] J. Mcdonough, "Moving standards to 100 Gbe and beyond," *IEEE Commun. Mag.*, vol. 45, no. 11, pp. 6–9, Nov. 2007.

[9] *RDMA Aware Networks Programming User Manual*, Mellanox, Yokne'am Illit, Israel, May 2015.

[10] *Intel Performance Scaled Messaging 2 (PSM2)*, Intel, Santa Clara, CA, USA, Nov. 2015.

[11] *OFIWG, Libfabric*, accessed on Oct. 2016. [Online]. Available: https://ofiwg.github.io/libfabric/

[12] *InfiniBand Architecture Specification, Release 1.2.1*, InfiniBand Trade Assoc. Admin., Beaverton, OR, USA, Oct. 2006.

[13] *Data Plane Development Kit: Programmer's Guide, Revision 6*, Intel, Santa Clara, CA, USA, Jan. 2014.