The InfiniBand based Event Builder implementation for the LHCb upgrade

IOP Conf. Series: Journal of Physics: Conf. Series **898** (2017) 032036 doi:10.1088/1742-6596/898/3/032036

The InfiniBand based Event Builder implementation for the LHCb upgrade

A Falabella [1], F Giacomini [1], M Manzali [1 2], U Marconi [3], N Neufeld [4], S Valat [4] and B Voneki [4]

[1] INFN CNAF
[2] Università degli Studi di Ferrara
[3] INFN Bologna
[4] CERN

E-mail: matteo.manzali@cnaf.infn.it

Abstract. The LHCb experiment will undergo a major upgrade during the second long shutdown (2019 - 2020). The upgrade will concern both the detector and the Data Acquisition system, which are to be rebuilt in order to optimally exploit the foreseen higher event rate. The Event Builder is the key component of the DAQ system, for it gathers data from the sub-detectors and builds up the whole event. The Event Builder network has to manage an incoming data rate of 32 Tb/s from a 40 MHz bunch-crossing frequency, with a cardinality of about 500 nodes. In this contribution we present an Event Builder implementation based on the InfiniBand network technology. This software relies on the InfiniBand verbs, which offers a user space interface to employ the Remote Direct Memory Access capabilities provided by the InfiniBand network devices. We will present the performance of the software on a cluster connected with 100 Gb/s InfiniBand network.

1. Introduction

LHCb [1] is one of the four main experiments at the Large Hadron Collider (LHC). A major upgrade of the detector is foreseen during the second long shutdown of the LHC (2019-2020). The upgrade will concern both the detector and the Data Acquisition (DAQ) system. They will be reviewed in order to optimally exploit the collision rate of about 40 MHz. In fact one of the main limitations of the way the LHCb detector is currently operated is that the collision rate must be reduced to match the maximum readout rate of 1.1 MHz. The rate reduction is achieved by a hardware trigger operating within a fixed latency of a few microseconds. Due to its implementation the hardware trigger causes the largest inefficiencies in the entire trigger chain. One of the main objectives of the LHCb upgrade is to remove this bottleneck, thanks to the adoption of an approach characterized by a trigger-less readout and a fully software-driven event selection [2].

2. The upgraded DAQ design

The Event Builder (EB) is the enabling component of the upgraded DAQ system. It collects and reassembles the event fragments delivered by the sub-detector readout boards, called PCIe40 [3], at collision rate. Each PCIe40 is equipped with up to 24 optical links coming from the detector and it is directly connected to a node of the EB through a Peripheral Component Interconnect

Express (PCIe) slot. Consecutive event fragments transmitted from the front-end electronics are received and buffered by the PCIe40 and then copied into the EB node memory by means of Direct Memory Access (DMA) operations.

Given the foreseen nominal event size of 100 KB and the maximum event rate of 40 MHz, the aggregated bandwidth of the EB network can be estimated to be of the order of 32 Tb/s. Assuming an input rate, through the PCIe40 board, of 100 Gb/s per server, the size of the EB cluster can be estimated of the order of 500 nodess. The final foreseen read-out system for the upgrade is shown in Figure 1, in which each of the estimated 500 nodes of the EB cluster will receive data from the front-end electronics and will exchange data with its peers at about 100 Gb/s full-duplex. Moreover, in case of no data filtering performed on an EB node, the built events will be sent out to an Event Filter Farm (EFF) again at about 100 Gb/s.

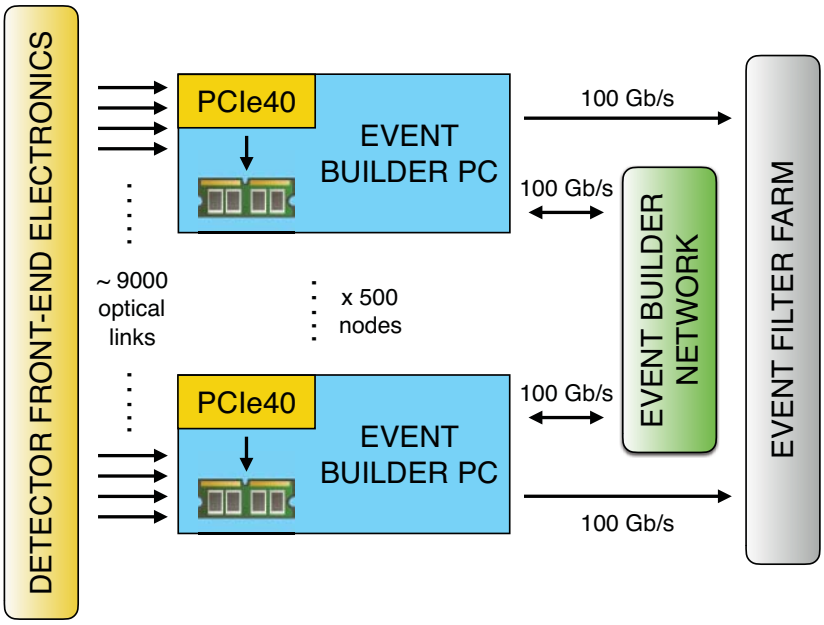

Figure 1. The architecture of the upgraded LHCb read-out system.

The EB network can be effectively implemented by using commercial LAN technologies such as 100G Ethernet [4], OmniPath [5] or InfiniBand [6]. In the following we first present the design choices adopted for the Large Scale Event Builder (LSEB) software [7], an EB software implementation designed for an InfiniBand interconnect infrastructure. Then we show some scalability tests aimed at validating those choices.

One of the key capabilities of InfiniBand is the support for Remote Direct Memory Access (RDMA), that is the ability to transfer data directly between applications over the network with no operating system involvement and while consuming negligible CPU resources on both sides (zero-copy transfers). This makes InfiniBand a high-speed, low-latency and low CPU-overhead interconnect technology.

3. The Event Builder implementation

In the EB design foreseen for the upgrade, each EB node includes two distinct logical components: the Readout Unit (RU) and the Builder Unit (BU). A RU receives event fragments from the detector and ships them to a receiving BU in a many-to-one pattern. Each BU gathers the event fragments and assembles them in full events, which are then sent out to the EFF

for processing. The LSEB software implementation reflects this design and its main blocks are represented in the schematic view of Figure 2.

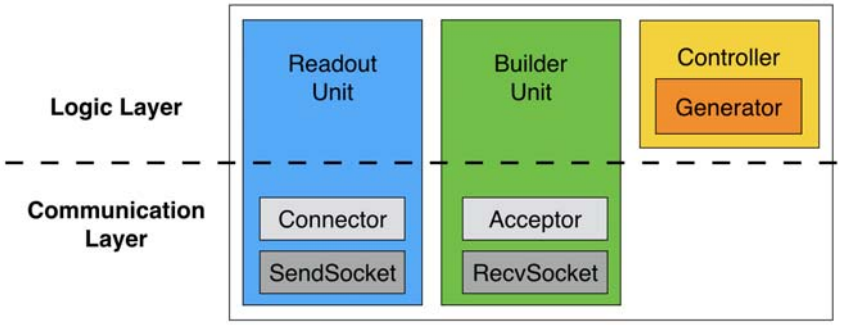

Figure 2. Schematic view of the main blocks of LSEB.

In order to keep the communication management separated from the event-building logic, LSEB can be logically split into two distinct layers, namely the Communication Layer and the Logic Layer. The Communication Layer includes primitives for data communication between nodes and relies on the InfiniBand verbs [8] library, which offers a user-space interface to access the RDMA capabilities of the network device. On top of the Communication Layer sits the Logic Layer, a set of software components performing the actual event-building under realistic conditions.

3.1. Options for RDMA programming

RDMA programming offers many more options and is more complex than traditional sockets programming, as it requires the programmer to directly manipulate data structures defined by the network interface in order to directly control all aspects of RDMA message transmission. Therefore, the programmer must take decisions which may drastically affect performance. The following sections explain which implementation choices have been taken in order to develop the Communication Layer of LSEB.

3.1.1. Completion detection strategy. The basic principle of RDMA is the ability to access memory on a remote machine without involving the CPU, leaving it free to perform other tasks. The verbs implement this principle and they are designed to be asynchronous. Each transfer call is performed by posting a work request on a dedicated First-In First-Out (FIFO) queue, named Work Queue (WQ); the call returns immediately instead of waiting for the completion of the requested operation. There are two ways for an application to know about the completion of a work request:

- *Busy polling.* The first completion detection strategy is to poll a dedicated FIFO queue, named Completion Queue (CQ), for a work completion.
- *Event notification.* The second strategy is to set up a dedicated channel, named Completion Channel, that allows an application to wait until a notification is signaled on it.

Repeatedly polling the CQ allows immediate reaction to completions at the cost of full CPU utilization, but requires no operating system intervention. On the other hand, the event notification approach is somewhat less efficient than the polling approach since it introduces the overhead of an interrupt handler and context switches between user and kernel modes [9]. Furthermore, the second strategy forces to adopt an event oriented design for the whole application in order to avoid a busy waiting call somewhere in the logic of the program.

The main aim of the Readout Units in the EB design is to send data to the Builder Units as soon as it is available. When all the send operations are started, the Readout Units cannot do anything but wait for a work completion and, with the high data rates expected, it is likely that the poll operation is almost always successful. Thus, LSEB implements the busy polling strategy, privileging the performance and keeping a linear and simple design.

3.1.2. Memory management. The DMA engine responsible for transferring data from main memory to the network device handles only physical addresses. Thus, the virtual addresses of the communication buffer have to be translated into physical ones; this process is called memory registration. Registering a Memory Region (MR) is a time-consuming process which requires several operations to be performed by the operating system and the driver [10]. In order to avoid registration and deregistration of memory for every transfer, it is preferable to use designated memory buffers, which are registered only once. Furthermore, registering physical contiguous memory can allow the low-level drivers to perform better since fewer memory address translations are required [11]. Thus, it is a good practice to register a large MR and to access a subset of it each time.

These considerations fit well with the design of the EB, in which a large and contiguous memory area of the Readout Units is written by the readout boards and subsets of it are sent remotely to the Builder Units. Therefore, LSEB allows the registration of a single MR, keeping all the memory contiguous, and avoids temporary registrations/deregistrations.

3.1.3. Work request type. Starting from all the data transfer options foreseen by the InfiniBand architecture [12], we identified two viable paradigms to implement the communication logic of the EB:

- *send/receive*. In this paradigm a receiver first posts a *receive* work request that describes a MR into which the Host Channel Adapter (HCA) should place a single message. The sender then posts a *send* work request which refers to a MR containing the message to be sent. The HCAs transfer data directly from the sender's memory to the receiver's memory without any intermediate copy. Since both sides of the transfer are required to post work requests, this is called a "two-sided" transfer.
- *rdma_write/receive*. The second paradigm is a "one-sided" transfer in which a sender posts an *rdma_write* request that pushes a message directly into a MR that the receiving side previously communicated to the sender. The receiving side is completely passive during the transfer. When the sender needs to notify the receiver, it posts a *rdma_write_with_immediate* request to push a message directly into the receiving side's MR, as for *rdma_write*, but in this case the work request also includes 4 bytes of immediate (out-of-band) data that is delivered to the receiver on completion of the transfer. The receiving side needs to post a *receive* work request to catch these 4 bytes, and the work completion for the *receive* indicates the status and the amount of data written with the *rdma_write_with_immediate* operation.

In the EB logic each Builder Unit needs to know the address and the size of each fragment (or collection of fragments) sent by the Readout Units in order to build complete events. With the *send/receive* paradigm each Builder Unit implicitly obtains such information, being notified of the memory address and the transferred size of each transfer. With the *rdma_write/receive* paradigm the immediate data can be used to communicate the address, but this requires to post a *receive* work request by the Builder Unit for each data transfer. Moreover, using this second paradigm requires the Readout Units to be aware of the remote memory in order to write only on those MRs no longer used by the Builder Units. Efficiency studies have shown that the performance of these two approaches are equivalent [13]. However, the use of the first paradigm

has the advantage of avoiding the implementation of memory management and synchronization mechanisms. For these reasons LSEB adopts the *send/receive* paradigm to perform RDMA data transfers.

4. Performance tests

LSEB has been tested on a variety of clusters [14]. Here we describe the tests carried out on a 84-node High Performance Computing cluster in order to study the performance and the scalability of the software to validate the design choices described above.

4.1. Testbed details

We had the possibility to run LSEB on a cluster composed by 84 nodes and fully connected with InfiniBand EDR interconnect. Each node of the cluster is a "C6320 PowerEdge" server [15] and it contains two Intel Xeon Haswell E5-2697 v4 processors, each with 18 cores (36 hardware threads) and a clock of 2.3 GHz. The EDR (Enhanced Data Rate) standard is the highest data rate currently available with InfiniBand; it foresees 25 Gb/s of raw signaling data rate per link and in a classic four-link configuration it allows to reach a raw throughput of 100 Gb/s. Considering the overhead induced by the 64b/66b encoding scheme, the maximum theoretical bandwidth allowed by the InfiniBand EDR interconnect is of 96.97 Gb/s.

4.2. Measurements

Generically speaking, before testing an application on one or more nodes, it is required to execute standard benchmarks on those machines in order to establish how that application performs and how it can be improved. In case of applications that make use of RDMA, there is a set of micro benchmarks provided by the OFED package [16] that allows for the verification of the effective point-to-point network capacity. One of these micro benchmarks, the so called *ib_write_bw*, was chosen and used to identify the real maximum bandwidth attainable between two random nodes belonging to the cluster. The tests performed with *ib_write_bw* foresee the execution of 5000 bidirectional RDMA data transfer operations for each different buffer size from 2^1 to 2^{22} bytes. Figure 3 shows the average bandwidth obtained running the **ib_write_bw** tool on two nodes of the cluster: the benchmark reaches about 95 Gb/s with buffer sizes greater than 32 KB (2^{15} bytes).

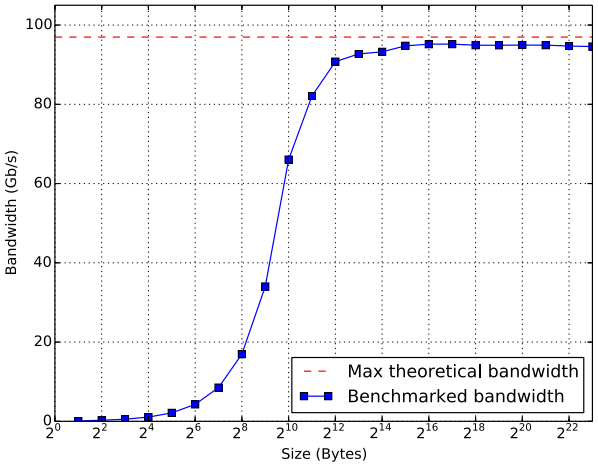

Figure 3. Bandwidth benchmarked with *ib_write_bw* on two nodes of the cluster.

After the benchmark with *ib_write_bw*, we started running LSEB on different scales. The scalability plot is shown in Figure 4, where the average bandwidth is plotted as a function of the number of nodes. In all the tests LSEB runs with the same fragment aggregation cardinality of 600 fragments; this implies that each data transfer has an average size of about 128 KB. This size is big enough to potentially saturate the bandwidth, as shown by the benchmark previously described. Due to availability issues we were able to exploit only up to 64 of the 84 nodes of the cluster. Starting from a 2-node setup, several tests were performed doubling each time the number of nodes, up to 64 nodes. Moreover, in order to better understand the bandwidth gap between 32 and 64 nodes, additional tests were performed with 36, 42, 48 and 56 nodes.

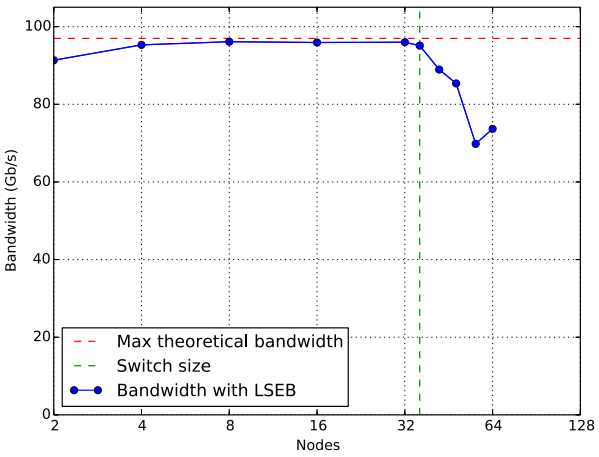

Figure 4. Bandwidth measured running LSEB on a different number of nodes.

The network topology of the cluster is a two-level non-blocking fat-tree network with 5 edge switches and 3 core switches. Both the edge and the core switches are Mellanox EDR InfiniBand SB77X0 systems [17] with 36 ports each. In a fat-tree network [18] the servers represent the leaves. In a two-level fat-tree network there are two layers of switches: servers are connected to the edge switches and the edge switches are interconnected through the core switches. When the up-links and down-links in an edge switch are in a 1:1 proportion the network is non-blocking. The presumed switch hierarchy present in the cluster is sketched in Figure 5. The plot in Figure 4 clearly shows an unexpected bandwidth drop while moving across the switches. This might be explained in case of a discontinuous node allocation on the cluster, which might create link contention. However, we do not expect such contention in a non-blocking fat-tree network. More studies are needed in order to understand the ultimate causes of this bottleneck and if the implementation of a software aware of the specific network topology can reduce the performance drop with an increasing number of nodes.

5. Conclusions

LHCb will under a major upgrade during the second long shutdown. Apart from detector upgrades, also the readout system will be redesigned in order to allow for a trigger-less data acquisition. We developed an evaluator in order to study the performance of the EB with high-throughput network technologies and we tested this software on a cluster connected with an InfiniBand EDR network, able to provide 100 Gb/s full-duplex bandwidth. The performed scalability tests show that the solution we implemented is promising and that the InfiniBand EDR interconnect could cope with the event-building requirements of the upgraded LHCb experiment. However, a complete and deep knowledge of the network topology is needed in

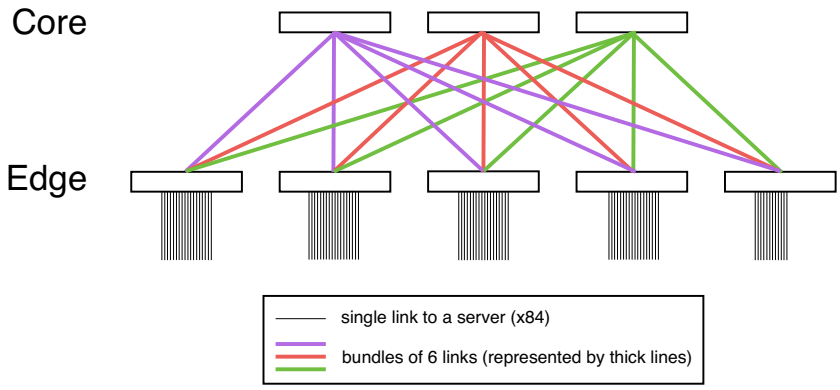

Figure 5. Presumed switch hierarchy of the cluster.

order to optimally exploit the bandwidth capacity with a number of nodes that require a switch hierarchy.

References

[1] LHCb Collaboration, *The LHCb detector at the LHC*, JINST,3,2008,S08005
[2] LHCb Collaboration, *LHCb Trigger and Online Upgrade Technical Design Report*, CERN,LHCC,2014,C10026
[3] M. Bellato, G. Collazuol, I. D'Antone, P. Durante, D. Galli, B. Jost, I. Lax, G. Liu, U. Marconi, N. Neufeld, R. Schwemmer and V. Vagnoni, *A PCIe Gen3 based readout for the LHCb upgrade*, Journal of Physics: Conference Series (2104).
[4] 100 Gigabit Ethernet Technology Overview, `http://www.ethernetalliance.org/wp-content/uploads/2011/10/document_files_40G_100G_Tech_overview.pdf`.
[5] Intel Omni-Path Architecture, `http://www.intel.com/content/www/us/en/high-performance-computing-fabrics/omni-path-architecture-fabric-overview.html`.
[6] R. Buyya, T. Cortes, H. Jin, *An Introduction to the InfiniBand Architecture*, Wiley-IEEE Press (2002).
[7] M. Manzali, *lseb 2.0*, DOI 10.5281/zenodo.46935.
[8] RDMA Protocol Verbs Specification (Version 1.0), `http://www.rdmaconsortium.org/home/draft-hilland-iwarp-verbs-v1.0-RDMAC.pdf`.
[9] A. Cohen, *A Performance Analysis of 4X InfiniBand Data Transfer Operations*, Parallel and Distributed Processing Symposium (2003).
[10] F. Mietke, R. Rex, R. Baumgartl, T. Mehlan, T. Hoefler and W. Rehm, *Analysis of the Memory Registration Process in the Mellanox InfiniBand Software Stack*, Proceedings of Euro-Par 2006 Parallel Processing (2006).
[11] Tips and tricks to optimize your RDMA code, `http://www.rdmamojo.com/2013/06/08/tips-and-tricks-to-optimize-your-rdma-code/`.
[12] InfiniBand Architecture Specification (Release 1.3), March 2015, `https://cw.infinibandta.org/document/dl/7859`.
[13] P. MacArthur and R. Russell, *A Performance Study to Guide RDMA Programming Decisions*, Proceedings of the 2012 IEEE 14th International Conference on High Performance Computing and Communication (2012).
[14] A. Falabella, M. Manzali, F. Giacomini, U. Marconi, B. Voneki, N. Neufeld and S. Valat, *Large-scale DAQ tests for the LHCb upgrade*, 2016 IEEE-NPSS Real Time Conference (2016).
[15] Dell PowerEdge C6320 Datasheet, `http://i.dell.com/sites/doccontent/shared-content/data-sheets/en/Documents/Dell-PowerEdge-C6320-Spec-Sheet.pdf`.
[16] OpenFabrics Enterprise Distribution (OFED), `https://www.openfabrics.org/index.php/openfabrics-software.html`.
[17] Mellanox 1U EDR 100Gb/s InfiniBand Switch Systems Hardware User Manual Models: SB7700/SB7790, `http://www.mellanox.com/related-docs/user_manuals/1U_HW_UM_SB77X0.pdf`.
[18] C. Leiserson, *Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing*, IEEE Transactions on Computers (1985).