Exploring RapidIO technology within a DAQ system event building network

Simaolhoda Baymani, Konstantinos Alexopoulos, Sébastien Valat

***Abstract*—Exploring RapidIO RapidIO (http://rapidio.org/) technology is a packet-switched high-performance fabric, which has been under active development since 1997. The technology is used in all 4G/LTE basestations worldwide. RapidIO is often used in embedded systems that require high reliability, low latency and deterministic operations in a heterogeneous environment. RapidIO has several offloading features in hardware, therefore relieving the CPUs from time-consuming work. Most importantly, it allows for remote DMA and thus zero-copy data-transfer. In addition it lends itself readily to integration with FPGAs.**

In this paper we investigate RapidO as a technology for high-speed DAQ networks, in particular the DAQ system of an LHC experiment. We present measurements using a generic, multi-protocol event-building emulation tool which was developed for the LHCb experiment.

Event building using a local area network, such as the one foreseen for the future LHCb DAQ puts heavy requirements on the underlying network as all data sources from the collider will want to send to the same destinations at the same time. This leads to an instantaneous overcommitment of the output buffers of the switches.

We will present results from implementing a event building cluster based on RapidIO interconnect, focusing on the bandwidth capabilities of the technology as well as the scalability.

I. Introduction

RapidIO is a high-performance, low pin count, packet switched system level interconnect standard. RapidIO promises the combined strengths of PCI Express and Ethernet while at the same time offering support for heterogeneous systems. The above together with the inherent support for error handling at the hardware level, motivates a closer investigation of the technology and in particular its suitability in the high-speed network domain, where speed and robustness are essential. [1]

Simaolhoda Baymani is with the IT Department, European Organization for Nuclear Research, Geneva, CERN CH-1211 Geneva 23, Switzerland (e-mail: sima.baymani@cern.ch)

Konstantinos Alexopous is with the IT Department, European Organization for Nuclear Research, Geneva, CERN CH-1211 Geneva 23, Switzerland. He is also with the Electrical and Computer Engineering Department, National Technical University of Athens, Heroon Polytechniou 9, 15780 Zografou, Greece (e-mail: konstantinos.alexopoulos@cern.ch)

Sébastien Valat is with Experimental Physics Department, European Organization for Nuclear Research, Geneva, CERN CH-1211 Geneva 23, Switzerland (e-mail: sebastien.valat@cern.ch)

This project aims to explore RapidIO in different domains of network applications. In this paper we will present the work done in the fields of data analytics and data acquisition.

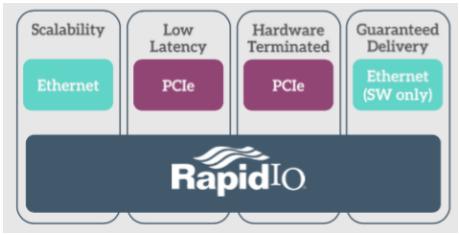

Fig. 1. RapidIO vs PCIe vs Ethernet. *Image source: IDT.*

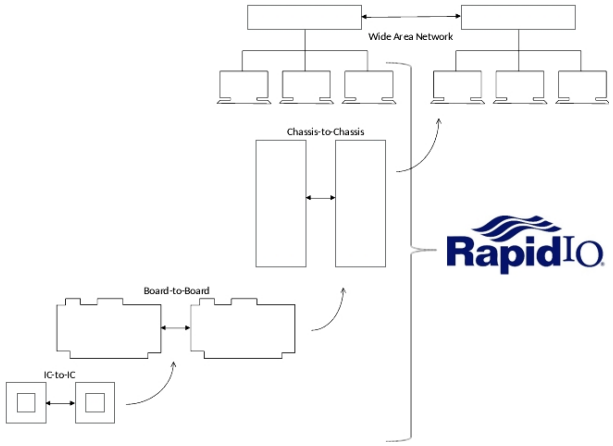

Fig. 2. Interconnect Application Domains[6]

A. The Protocol

The RapidIO protocol is based on request and response transactions. The communication elements between the various end point devices in the system are packets. Control symbols are used for packet acknowledgment, flow control information and maintenance functions. All error handling is done at the hardware level. Notable logical operations introduced in the protocol specification include Read/Write operations (Direct Memory Access) and Messaging (Channelized Messaging). Speeds of 1.24-10.3125 GBaud per lane are achievable according to the latest protocol specification. [2]

B. The Interface

User-space programs normally access the interconnect via library calls. The individual library calls interface the un-

978-1-5090-2014-0/16/$31.00

derlying driver, which handles the creation and termination of connections, as well as the orchestration of *Channelized Messaging* (CM) and *remote DMA* operations (from now referred to as rDMA).

C. The Setup

Our hardware setup consists of a 2U Quad unit with four Intel Xeon L5640@2.27GHz nodes, each with 48GB of RAM. Each server is equipped with an IDT Tsi721 PCIe to RapidIO Bridge, offering speeds at 14.5 Gbps. The nodes are connected to a 38-port Top of Rack (ToR) RapidIO Generation 2 switch box using QSFP+ cables. The switch ports offer speeds at 20Gbps. The speed difference between the switch port and the PCIe bridge are due to PCIe bus limitations.

The servers run on CERN CentOS release 7.2.1511. The RapidIO software stack used is Linux kernel drivers and libraries provided by IDT. This software package is under development.

II. ROOT

ROOT[3] is a data processing framework developed at CERN. Originally targeted at data analysis and simulations related to high-energy physics, ROOT is now used by numerous teams around the world in applications ranging from data mining to astronomy. For the project described ROOT has been extended to use the RapidIO protocol instead of TCP/IP, for transactions between two or more of its instances. For our purposes two implementations have been developed: one is utilizing CM and the other rDMA. Two important factors that influenced the respective designs were:

1) ROOT's internal bookkeeping and layering conventions
2) Limitations imposed by the current RapidIO library implementation

A. Channelized Messaging Implementation

The Channelized Messaging implementation is comprised of consecutive send and receive function calls for the maximum discrete buffer size until the entirety of the data has been propagated to the other side. Every message is acknowledged in order to achieve synchronization.

B. Direct Memory Access Implementation

The rDMA implementation is based on the same principle as the CM implementation. Consecutive writes of the maximum possible size are performed until the entirety of the data has been written to the target memory destination. Messages have been used for inititating and acknowledging every rDMA operation.

C. Benchmarking

In order to evaluate RapidIO's performance for this particular extension of ROOT, the following scenario was outlined. A single server - multiple clients topology was set up. The server is running an instance of ROOT, which is accepting connections and, consecutively, data. The clients are sending data to the server concurrently.

The above has been used to evaluate the performance of both the CM and the rDMA implementation.

D. Results

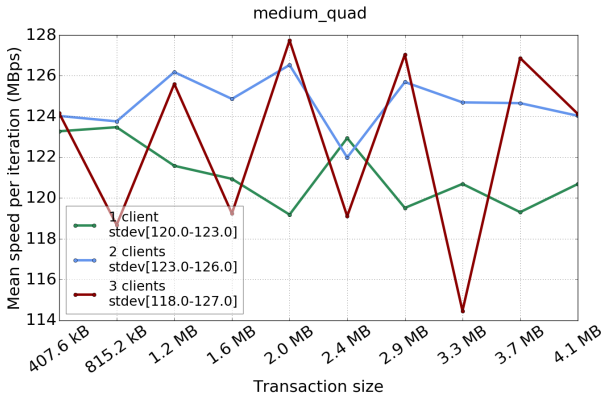

Fig. 3. ROOT - CM

Fig. 4. ROOT - rDMA

On figures 3 and 4 the results for a certain range of transaction sizes for both implementations are shown.

E. Conclusions

In order to correctly evaluate benchmarking results several factors need to be taken into account.

- ROOT utilizes its internal bookkeping. The various internal operations that are performed by ROOT can introduce an important overhead, which may skew the results.
- ROOT is a user-space application, managed by the Linux kernel. Scheduling operations can affect the performance of different transactions, introducing minor differences between each run.

Due to factors such as those stated above utilization close to the nominal was not achieved. However, a significant difference in speed was observed between the CM and the rDMA implementation, with the first maxing at around 120MBps and the second around 1000MBps. This is expected, as CM operations are targeted to orchestration, whereas rDMA operations are targeted to data transfers.

III. DAQPIPE

LHCb-DAQPIPE, DAQ Protocol-Independent Performance Evaluator, is a benchmark application to test network fabrics

for the future upgrade of the LHCb experiment at CERN. DAQPIPE emulates an event-builder based on a local area network, such as the one envisioned in the LHCb upgrade. The application is protocol, topology and transport agnostic, allowing for multidimensional testing of an interconnect. Several technologies have already been ported to DAQPIPE for evaluation.[4]

The event-building network itself is a fully connected network where nodes receive data from the readout system. This data is subsequently sorted into events. Data for one event may be spread out over several nodes and first has to be aggregated. The aggregation can be configured in many ways, with different aggregation protocols, memory layouts and network topologies. DAQPIPE has a large set of configuration parameters which allow for thorough testing of which parameter set that gives the most optimal results for a specific interconnect.

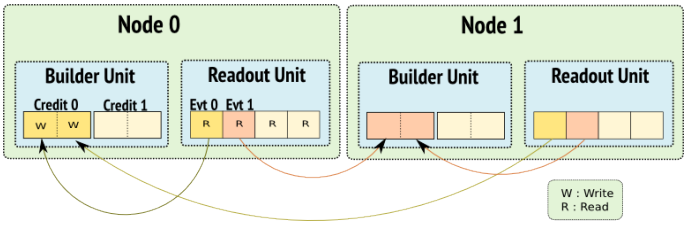

Fig. 5. Data aggregation in DAQPIPE[5]

A. Implementation

DAQPIPE was ported to RapidIO using both Channelized Messages and rDMA. Channelized Messages were used for commands, which are used to organize the event building and rDMA buffers were used as data buffers. Threads were used to synchronously handle connection operations.

Following the main porting work, the following limitations in the RapidIO library created the need for two variants of the port:

1) Due to a combination of restrictions in the Linux kernel and the current RapidIO library implementation, there is a maximum size of 2MB for each rDMA allocation
2) Due to hardware constraints, there can be maximum 8 rDMA allocations at one time

In the standard implementation of DAQPIPE, each node will have a read and write buffer. The read buffer contains the readout data, i.e. parts of an event. The write buffer is for aggregating this event data into a full event. Nodes use offsets to write the event data into the corresponding address of the recipient node. With limited buffer size, each node can only collect a certain number of events.

In the alternative implementation, each node will instead have *multiple* write buffers, also called segments, which allows for a larger accumulated buffer size. However, the number of allocations need to obey the maximum limit. Thus, this solution limits the number of readout cards that can be connected to each unit.

B. Benchmarking

DAQPIPE was run with both implementation versions across several sets of configurations, alternating between three and four nodes. Benchmarking parametrization was influenced by library limitations.

C. Results

On figure 6 results are shown for the multiple segments implementation. As can be seen, the curve has not yet reached a stable state but is still reaching upwards pointing to a high probability of higher speeds if the allocation constraints are removed.

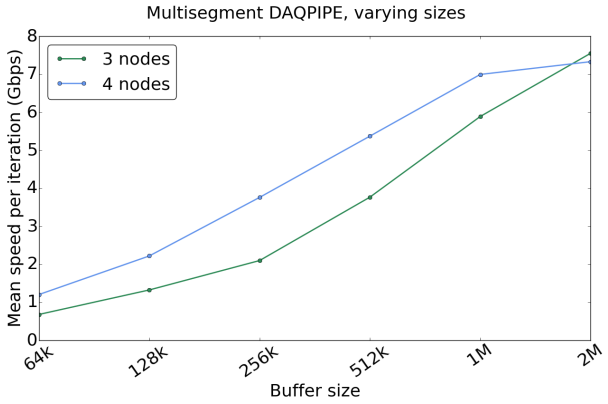

Fig. 6. Multisegment DAQPIPE with varying sizes

D. Conclusions

The DAQPIPE architecture uses commands to orchestrate the event data aggregation. Event data is subsequently stored in memory segments. In this sense, RapidIO CM and rDMA features fit well with the paradigms used by DAQPIPE. The need for an alternative write segment layout brings another interesting aspect into view, namely how the interconnect handles multiple, concurrent allocations.

ACKNOWLEDGMENT

The authors would like to thank Mohammad Akhter, Alexandre Bounine, Devashish Paul, Barry Wood from Integrated Device Technology and Olof Bärring, Niko Neufeld and Alexandru Grigore from the European Organization for Nuclear Reasearch, for their support and collaboration. The authors would also like to thank IDT for its membership and technical collaboration with CERN openlab.

REFERENCES

[1] Architecture and Systems Platform, *RapidIO™: An Embedded System Component Network Architecture*. Motorola Semiconductor Product Section. [Online]. Available: http://www.cs.ucr.edu/ mart/CS260/rapidIO.pdf

[2] RapidIO.org, *RapidIO™ Interconnect Specification Version 3.1*. [Online]. Available: http://www.rapidio.org/wp-content/uploads/2014/10/RapidIO-3.1-Specification.pdf

[3] Rene Brun and Fons Rademakers, *ROOT - An Object Oriented Data Analysis Framework*, Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86. See also http://root.cern.ch/

[4] D. H. Campora Perez, R. Schwemmer, N. Neufeld *Protocol-Independent Event Building Evaluator for the LHCb DAQ System*, Realtime Conference, 2014.

[5] D. Campora, S. Valat, B. Voneki, S. Baymani, *LHCB-DAQPIPE Wiki*. [Online]. Available: https://gitlab.cern.ch/svalat/lhcb-daqpipe-v2/wikis/home

[6] G. Shippen, *System Interconnect Fabrics: Ethernet versus RapidIO™ Technology*, Freescale Semiconductor Inc. [Online]. Available: http://cache.freescale.com/files/32bit/doc/app_note/AN3088.pdf