EUCALL

The European Cluster of Advanced Laser Light Sources

Grant Agreement number: 654220

Work Package 5 – UFDAC

Deliverable D5.2

Report on high speed data-transfer and data injection

Lead Beneficiary: ELI

Authors: Kwinten Nelissen, Balazs Bago, Petr Pivonka, Pavel Bastl, Nikolas Janvier, Bernd Schmitt, Carlos Lopez Cuenca, Martin Brückener, Wassim Mansour, Patrick Geßler, Michael Bussmann

Due date: 30.09.2018
Date of delivery: 29.09.2018

Project webpage: www.eucall.eu

Deliverable Type	
R = Report DEM = Demonstrator, pilot, prototype, plan designs DEC = Websites, patents filing, press & media actions, videos, etc. OTHER = Software, technical diagram, etc.	R
Dissemination Level	
PU = Public, fully open, e.g. web CO = Confidential, restricted under conditions set out in Model Grant Agreement CI = Classified, information as referred to in Commission Decision 2001/844/EC	PU

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

Contents

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

1. Introduction

Data transfer from detectors and their front-end systems into FPGAs and GPUs for online processing is crucial as well as demanding due to the required high data bandwidth and/or low latency. Data is transferred between detector, server, GPU/CPU memory in various ways. Modern technologies like 10/40 GBps Ethernet as well as PCIe along with DMA and zero copy are important aspects of the possible solutions. Given the major effort implementing different computer algorithms for this ever increasing complex hardware, a joint development is important. The goal here is the identification of common needs and establishing common standards and hardware topologies depending on the given requirement of the application. Some applications require an ultra-low latency while others require an ultra-high bandwidth. Example applications which require ultra large bandwidth are the Jungfrau Detector at PSI (Please specify here) and the pixel detector at PSI (BANDWIDTH). While at the other hand some applications require ultra-low latency such as trigger distribution systems and monitoring devices. Bandwidth is often trade to the expense of latency demanding a complete different strategy depending on the user case.

Data transfer and injection are important for ultrafast data acquisition in four distinct ways:

1. **Latency** of the solution. For real time feedback applications latencies must be as short as possible to avoid pile up or loss of important data. Examples of real time feedback applications are beam stabilization applications and trigger distribution. For high data rate application latency might exist, but can be hidden by buffering incoming data while performing calculations on previously acquired data in parallel.
2. **Throughput** of the solution. Throughput is foremost determined by the capabilities of the hardware that is used to transfer the data but also by the software stack used for data transfer.
3. **Scalability** of the solution. Scalability means that with growing demands the data transfer solution provided can be adapted, so that there is at best a linear relation between increase in throughput and increase in resources needed for data transfer. Scalability also means that solutions can be scaled to different hardware platforms if growing demands in throughput require this.
4. **Resilience** of the solution. If data production is expensive as is the case at many light sources, data loss during transfer or injection is inacceptable. As such, resilience of solutions is important to minimize the possibility of data loss.

In the following the status of data transfer and injection within UFDAC is summarized and discussed with reference to latency, throughput, scalability and resilience if applicable. Contributions from ELI-ALPS, ESRF, PSI, European XFEL, and HZDR are included.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

2. Overview of basic technologies and Infrastructure

2.1 Technologies for connecting devices

PCI-e

Peripheral Component Interconnect Express, officially abbreviated as PCIe or PCI-e, is a high-speed serial computer expansion bus standard.

RDMA

Remote direct memory access (RDMA) is a direct memory access from the memory of one computer into that of another without involving either one's operating system. This permits high-throughput, low-latency networking, which is especially useful in massively parallel computer clusters.

Ethernet

Ethernet is a family of computer networking technologies commonly used in local area networks (LAN), metropolitan area networks (MAN) and wide area networks (WAN). Systems communicating over Ethernet divide a stream of data into shorter pieces called frames. Each frame contains source and destination addresses, and error-checking data so that damaged frames can be detected and discarded; most often, higher-layer protocols trigger retransmission of lost frames.

Infiniband

InfiniBand (abbreviated IB) is a computer-networking communications standard used in high-performance computing that features very high throughput and very low latency. It is used for data interconnect both among and within computers. InfiniBand is also used as either a direct or switched interconnect between servers and storage systems, as well as an interconnect between storage systems.

2.2 Key technology challenges of computing hardwares

GPUs

Historically GPUs were made for image processing with the desire for fast parallel processing of individual pixels. Today's frameworks like OpenCL, CUDA or HCC enable todays GPUs for the wider use in highly parallelizable problems. One of the main concerns with this technology is that the average project live spans vary between 10 and 15 years, while at the other hand it is very hard to foresee which architectures, frameworks will be supported over this time span. Project requirements like scalability are not always clear at the beginning of the project and do vary often during the course of a project. This project enabled us to

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

identify similarities between different in house projects and perhaps maybe even more important across different facilities. Large parts of computer codes could be reused for different purposes without to re-invent the wheel over again

FPGA

Historically FPGA's could be seen as a kind of gluing element between specialized integrated circuits (IC). Its principle is based on thousands of programmable elements with connections that can be programmed in order to achieve the desired functionality. Today's FPGA are evaluated towards highly versatile interfacing and processing elements which are able to combine different technologies (e.g. CPUs, DSPs, Logical Blocks, Clocks, …) on a single piece of silicon. The huge advantage of a FPGA with respect to GPSs and CPUs is that its architecture is not fixed but completely configurable optimised for a certain processing task. For instance data types, memory structures, specialized processors can be implemented arbitrary. Typical applications of FPGAs can be found in prototyping, image and signal processing and even in artificial intelligence.

Conventional computing platforms

Despite GPUs and FPGAs in general enable faster data acquisition and processing than classical CPU architectures, modern CPUs may be also suited and cheaper than specialized hardware. Recent developments in CPU architectures with the inclusion of fast vector extensions like SSE, AVX or VSX enable to become more powerful and bring GPU technology towards the CPU from generation to generation. One of the big advantages of CPUs is the easy for development, debugging and profiling, tasks which are extremely hard on GPUs and FPGAs. Therefore, it would be desirable to be able to develop application on generic CPUs and within the same framework being able to scale the application towards high performance GPU cluster. A good example for project aiming this goal is Alpaka currently being developed at HZDR.

2.2 Computing infrastructures

The data acquisition system of ELI-Beamlines, as described in PBa18lg is based on a low latency network which enables sharing of memory pools between nodes of data acquisition servers.

In ELI-Beamlines, we focus on the actual practical implementation of a scalable data acquisition system with low latency and high throughput.

There is significant work being done on the implementation of data processing services, but due to the nature of our facility (multiple, complex, but often independent experiments; comparatively low volume of "global" online-processing), the current focus is on providing

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

the strongest-possible infrastructure the actual acquisition (from detector / digitizer /.. into storage.)

The underlying foundation is an optical network with >15.000 single-mode fibres running at 10 GB/s (scalable to 100 GB/s) (see Figure 1), which is divided it into three physically separated networks, for control, synchronization and data acquisition; following the scientific DMZ network architecture.

At the moment, we see first applications with throughputs of 10 GS/s and more from digitizers (for example: ADQ7-DC-F10-MTCA) and 2D-detectors (this corresponds to 1 GPixel/s). Several of those are already implemented, and the number is rapidly growing as installation progresses.

The system we designed (Figure 2) is divided into two parts: Top level and Local level.

Top level data acquisition system

In the server room, data is acquired, processed and stored using three main components:

- a blade server for DAQ which contains a 2x Infiniband FDR switch in the rear, a 2x Ethernet switch 10/40GBASE-X in the rear and currently 14 blades, each blade has 24 cores and 768 GB of memory for the buffer pool.
- the Infiniband network (network interface cards, cabling and switches) acting as a low-latency interconnection inside the blade server and to the data storage
- the multi-tier data storage where the tier-1 is based on flash drives, tier-2 is based on standard hard drives and tier-3 is based on tape library. (Implemented in stages, initially 6 PB)

The RAM of the DAQ server, which acts as a memory buffer, is considered tier-0.

Figure 1: View of server room / optical network installation

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

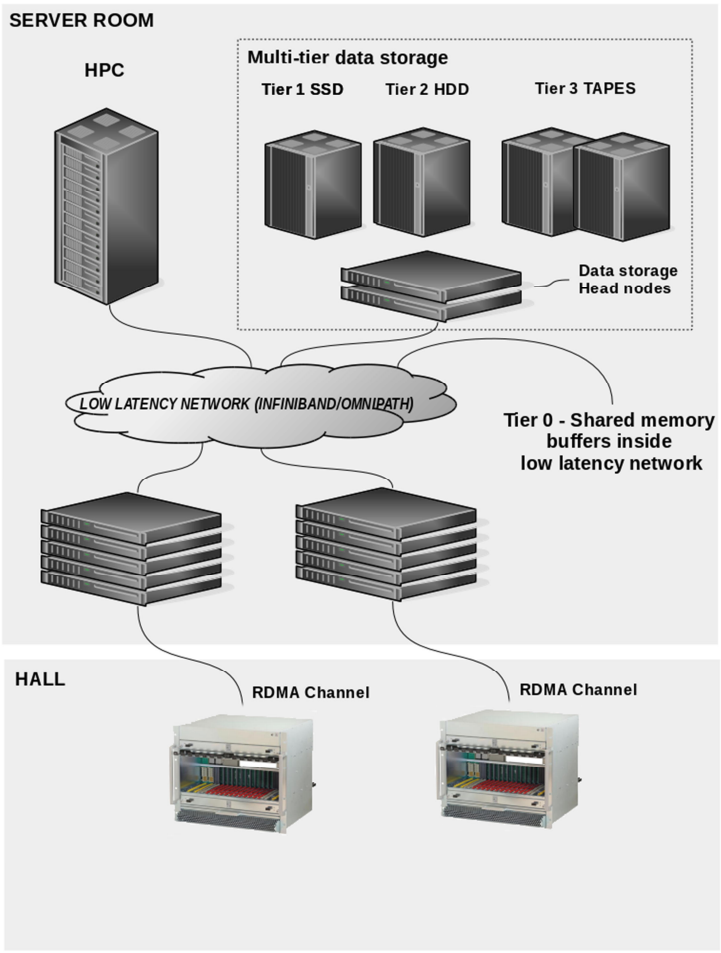

Figure 2: Overview of the Data Acquisition System in ELI-Beamlines

The DAQ servers are hereby used for three purposes:

- Data aggregation from the DAQ hardware installed in the hall using either pure NICs (Mellanox MCX4121A-XCAT: 2x 10GBASE-X, RDMA support) or FPGA cards (Alpha-Data ADM-PCIE-KU3: FPGA XCKU060, 2x QSFP, SDAcell support, 8 GB DDR; Mellanox Innova Flex LX-4 :FPGA XCKU060, 1x QSFP, RDMA support, 2 GB DDR)
- as a memory buffer pool (tier-0) that is safe from EMP and can be shared across processing units using Infiniband, giving also access to associated systems such as our HPC
- as a host for online data processing using both computer cores and the above described FPGA cards for acceleration. We do prefer FPGA as core accelerator (directly incoming data, no bottleneck from PCIe), but provide some Xeon Phi / GPU for users.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

Local data acquisition hardware

On the local level, we have two types of local DAQ hardware:

- standard PCIe-based DAQ systems (Supermicro) with 128 GB of RAM, 24 cores and 10 PICex8 slots. These servers are low-cost, can provide large memory buffers (terabytes) and there is a wide variety of PCIe-cards for different applications
- MTCA-based DAQ systems, which have the advantage of clock support (timing system, see next section) in the backplane and allow card-to-card-connection without involving the CPU. They have one limit: Due to the card size, the AMCs can only have 16 GB of buffer.

However, when we want to use the advantages of both (clock support, card-to-card-connections, large memory buffers), our MCH (NAT-MCH-PHYS80) allows to connect its internal PCIe switch through optical cable to PCIe based local DAQ server. The connection setup (shown in Figure 3) provides PCIe x16 interface and can be implemented using PCIe on MTCAs' agnostic backplane and its fat pipes.

Performance of the implementation

The installation of all major components of the ELI DAQ system was finalized in 2017. The data acquisition needs of the users are still quite modest, but expected to rapidly grow once the secondary sources are fully operational.

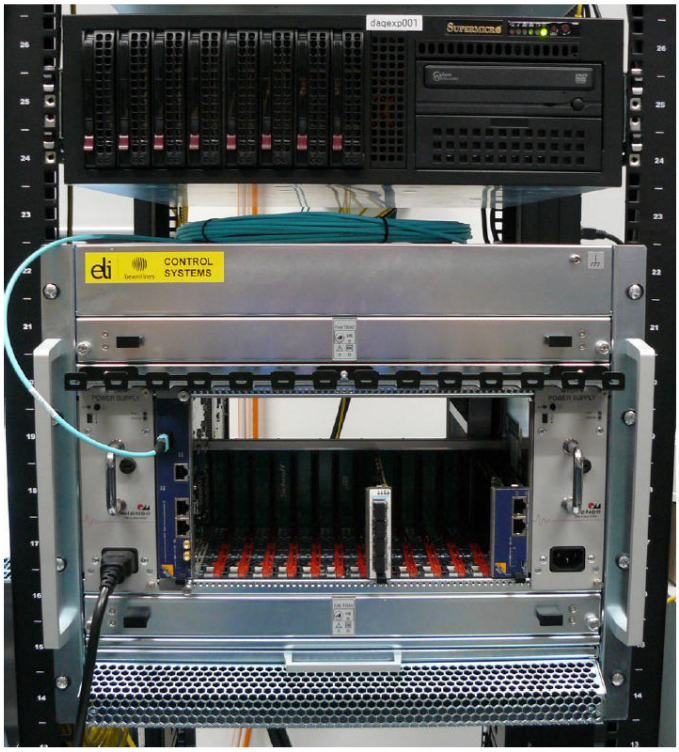

Figure 3: Standard PCIe DAQ system (top) connected with MTCA DAQ system (bottom) via optical fibre -PCIe

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

Furthermore, we are starting with small simulations and pilot projects regarding a new approach to network load management. Like most facilities, we see high fluctuations in network load and anticipate these to worsen with increasing data loads. Many network component vendors (including CISCO – which we are mainly working with) are starting to offer technologies for automatic balancing by directing the traffic on switch level, which we believe to be foolish: Such systems can only work well in homogeneous environments, and lead to vendor lock-in. Big physics research, especially user facilities, has a highly diverse landscape of "data producers".
Since our entire network is highly performant, we want to use our top level DAQ server hardware as a buffer for network load balancing. We are currently evaluating the capability of Infiniband (Mellanox), OmniPath (Intel), and CAPI (OpenPOWER foundation) together with our Tier-0 storage as network RAM buffer for load flattening.

3. Identified and developed solutions bridging technologies and performance results

3.1 Performance report on remote direct memory access (ELI-ALPS)

3.1.1 Introduction

Modern High Performance Computing (HPC) centers provide high-speed/low-latency network interconnects for the compute nodes. However, in order to harness the performance of this interconnects in an efficient and effective way a better understanding of data-injection handling is required, which depends on the available hardware and installed software stack.
The main aim of the report is to investigate the bandwidth and latency of these high-speed interconnects for different usage modes as function of the data packet size. For this purpose, two different benchmark programs are used. One them is the OSU benchmarks (OSU Micro-Benchmarks, 2017) based on the popular MPI-library and the other is the perftest benchmark (perftest, 2017) which perform data-injection at the lowest level. The dependence of the hardware stack is investigated by performing the benchmark programs on two different cluster, i.e. HZDR – hypnos (NIIF - 'KIFÜ/NIIF' LEO HPC, 2017) and NIIF - debrecen2 (HZDR, 2017).
At the end of the report software design guidelines are formulated allowing future software developers to select the most optimal software architecture for their use case. Still, a common programming model for multi- and many-core hardware was missing. This report introduces the Alpaka, developed within UFDAC, as a possible way to have single source development for a large variety of target platforms.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

3.1.2 Test platforms

For measuring the bandwidth and latency of the network interconnects two HPC centers, NIIF –debrecen2 and HZDR – hypnos were chosen. The benchmarks were performed with two different software performances tests, i.e. OSU and perftest. The software and hardware configuration are given below.

HPC center – NIIF

Software components:

- CUDA 8.0.61
- NVIDIA Driver 375.26
- Mellanox Driver 2.33.5000
- OpenMPI 1.8.5
- Red Hat Enterprise Linux Server release 6.8 (Santiago)

Hardware components:

- GPUs: 3 x Tesla K20Xm, 6 GB
- Network Interface card: Mellanox Technologies MT27600
- CPU: 2 x Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60 GHz
- System Memory: 12 GB RAM

The theoretical bandwidth of data transfer between GPUs and GPU - HOST is 8 GB/s since the GPUs are operating in PCI-e 2.0 mode over 16 lanes. The theoretical limit of data transfer of the NICs is determined to be 8 GB/s since the data transfer is established over PCI-e 3.0 with 8 lanes.

HPC center – HZDR

Software components:

- CUDA 8.0
- NVIDIA Driver 367.48
- Mellanox Driver 2.33.8000
- OpenMPI 1.8.6
- Ubuntu 14.04.5 LTS

Hardware components:

- GPUs: 4 x Tesla K20Xm, 4 GB
- Network Interface card: Mellanox Technologies MT27500 Family [ConnectX-3]

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

- CPU: 2 x Intel(R) Xeon(R) CPU E5-2609 0 @ 2.40GHz
- System Memory: 64 GB RAM

The theoretical bandwidth of data transfer between GPUs is determined to be 8 GB/s over PCI-e 2.0 with 16 lanes. The theoretical limit of data transfer from the Network Interface Cards (NIC) and host is given to be 8 GB/s since the data transfer is established over PCI-e 3.0 with 8 lanes.

3.1.3 Data transfer methods

For testing the data transfer performance of HPC centres two different kinds of data sources and targets are tested. One of them is the 'Host to Host' transfer. In this case data transfer is performed between the system memories of two compute nodes. As shown in the Figure 4 the data transfer (write or read) goes from the memory of Node A to the memory of Node B through nodes NIC (red arrows).

The other tests are the Device to Device tests, where the data goes from the memory of a PCI-e device of one node to the memory of a PCI-e device of another node. In our test environments the considered devices are NVIDIA GPUs. The data flow of this kind of tests is shown on the Figure 5 with red arrows. The main advantage here is that during the data transfer process the System memory is not accessed and data passes the PCI-e buss only for one time.

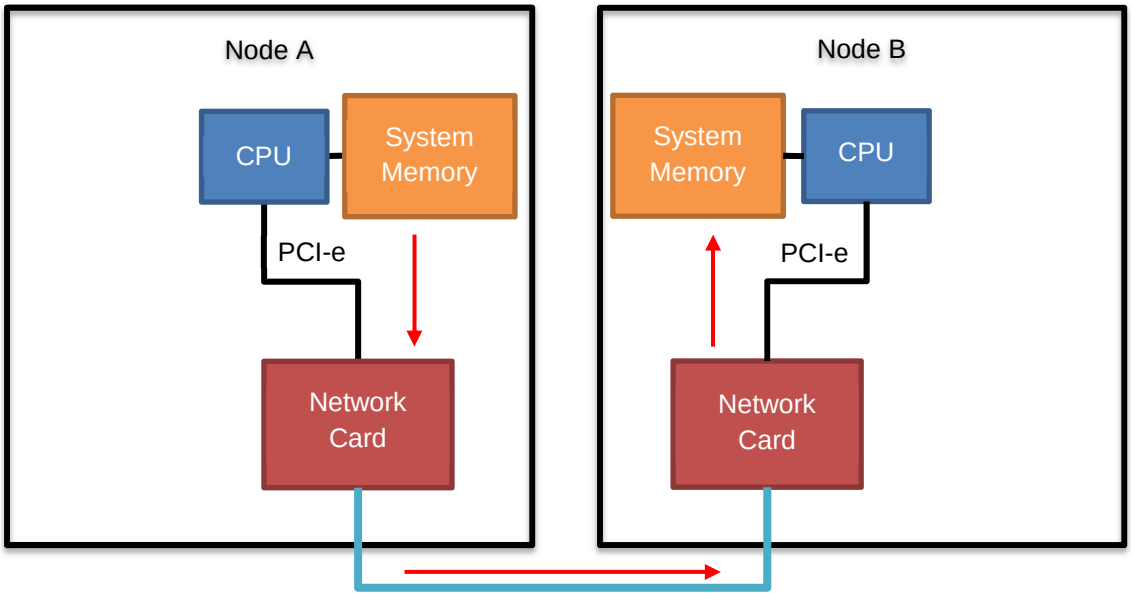

Figure 4: Data flow of Host to Host data transfer

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

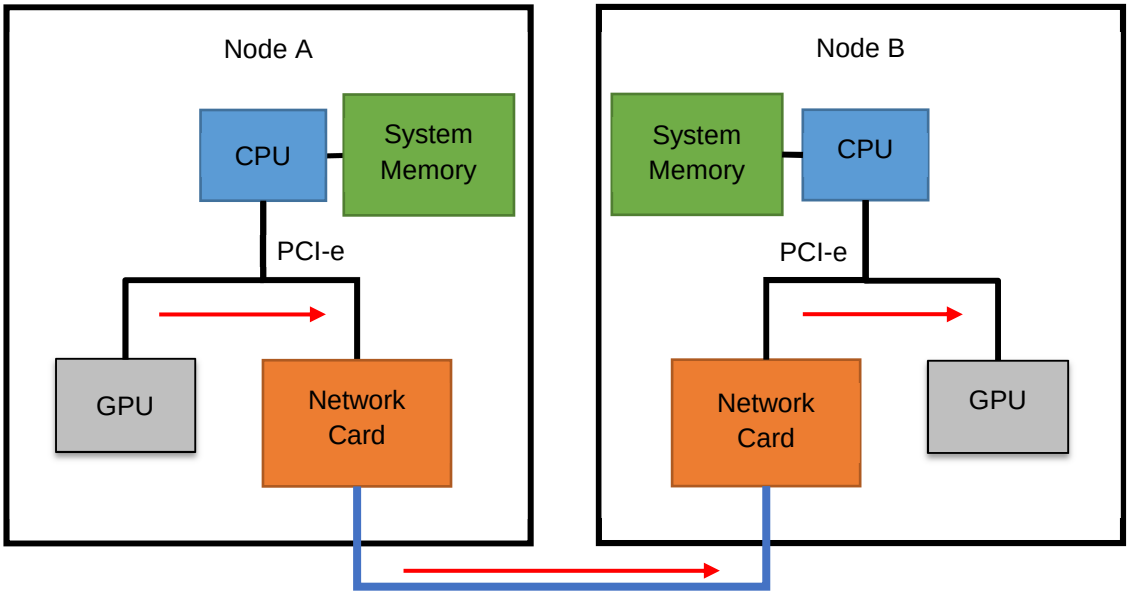

Figure 5: Data flow for Device to Device data transfer

3.1.4 OSU Micro-Benchmarks

The OSU Micro-Benchmark (OMB) software package is a tool for testing bandwidth and latency for network connection (http://mvapich.cse.ohio-state.edu/benchmarks/).

Host to host benchmarks

Two types of tests are performed on the different HPC centers. One of them was the bandwidth test (osu_bw) and the other was the latency test (osu_latency).

In bandwidth tests the sender sends back-to-back messages to the receiver and then waits for a reply from the receiver. The receiver sends the reply only after receiving all these messages. This process is repeated for several times and the bandwidth is calculated based on the elapsed time and the number of bytes sent by the sender (OSU Micro-Benchmarks, 2017).

The bandwidth for lower message size (size smaller than 8 KB) is 1.5 times higher for the HZDR (Figure 6), however the maximum bandwidth is nearly identical. A possible explanation for this behavior may be caused by driver differences.

In latency tests the sender sends a message to the receiver and waits for the response from receiver. The receiver sends back a message after receiving. After many iteration of this ping-pong tests the average latency values are calculated (OSU Micro-Benchmarks, 2017).

We discovered an analogous behavior at NIIF for small message sizes (smaller than 32 bytes). The exact reason at the time of writing this report is unknown but is currently under investigation.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

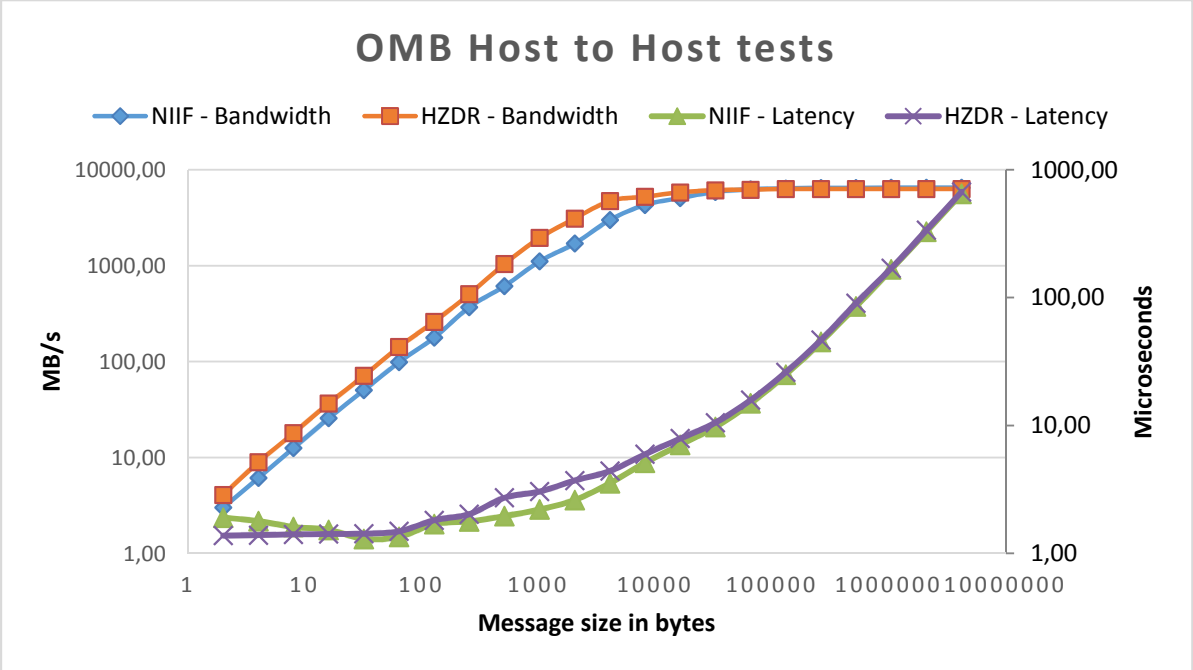

Figure 6: OSU Micro-Benchmarks Host to Host tests at NIIF and HZDR

Device to device benchmarks

The GPU to GPU message passing with CUDA-RDMA could be performed at the HZDR. At NIFF currently the Mellanox OFED GPUDirect RDMA package is not (yet) available. Similar to the host to host benchmark, bandwidth and latency tests were carried out.

In the bandwidth tests between device to device data transfer another interesting behaviour was observed. For message sizes between 1 KiB and 16 KiB the increment of bandwidth speed slows down and at 32 KiB the bandwidth rises significantly (3 times larger than at previous message size).

Conclusions

The OSU Micro-Benchmarks tests have shown the power of high bandwidth interconnections and by using GPU Direct RDMA technology of NVIDIA the GPU based computation can benefit from this interconnection as well. By using GPU RDMA latency is reduced for smaller message size (smaller than 32 KiB).

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

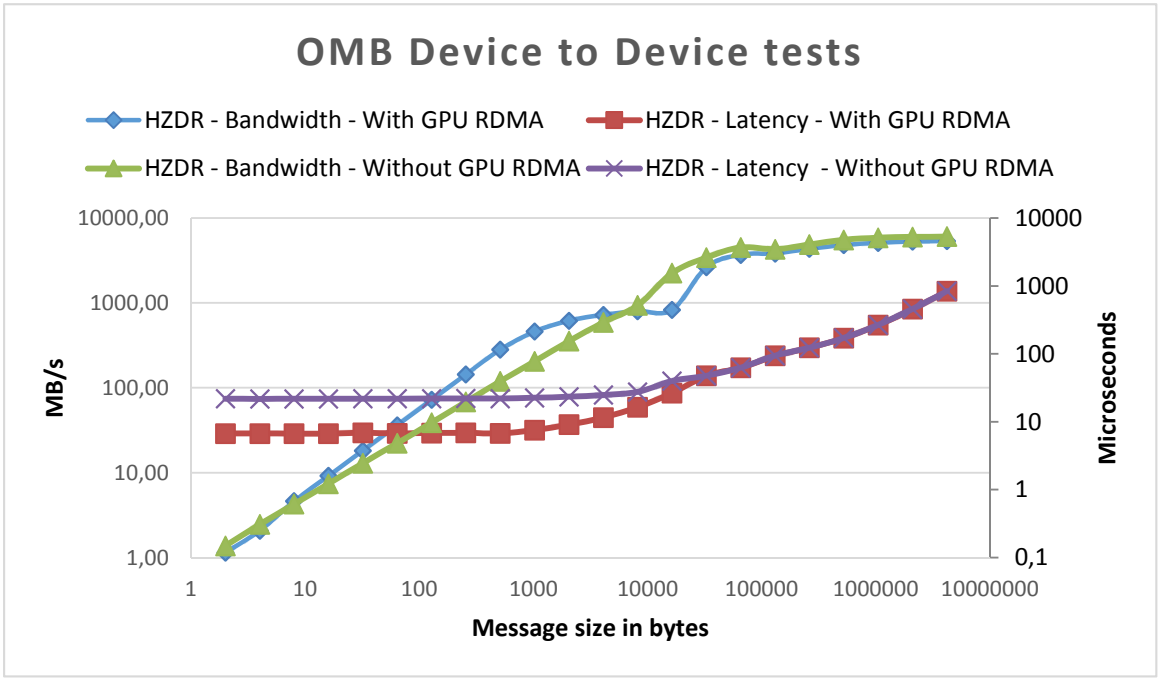

Figure 7: OSU Micro-Benchmarks Device to Device tests at HZDR

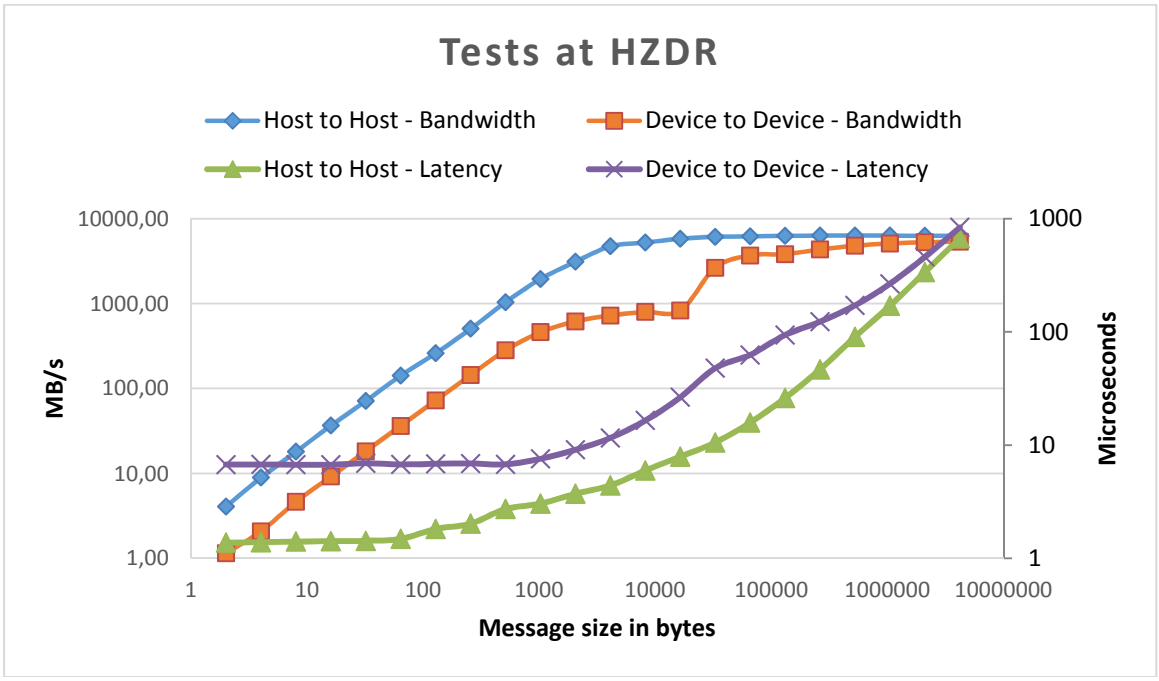

Figure 8: OSU Micro-Benchmark tests for HZDR

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

3.1.4 Perftest

A second benchmark software was used to confirm the results of the OMB tests. The 'perftest' is a software tool to test network performance. This tool was selected because of the performance test of NVIDIA for modern HPC platforms (NVIDIA Accelerated Computing, 2017). The perftest benchmark uses the directly ibverbs library (libibverbs, n.d.) to send and receive messages with RDMA (RDMA, n.d.), unlike at OMB tests, where RDMA is supported through Open MPI (Open MPI, n.d.).

Host to host

Two types of tests were used like at OMB tests, the bandwidth (ib_send_bw) and the latency (ib_send_lat).

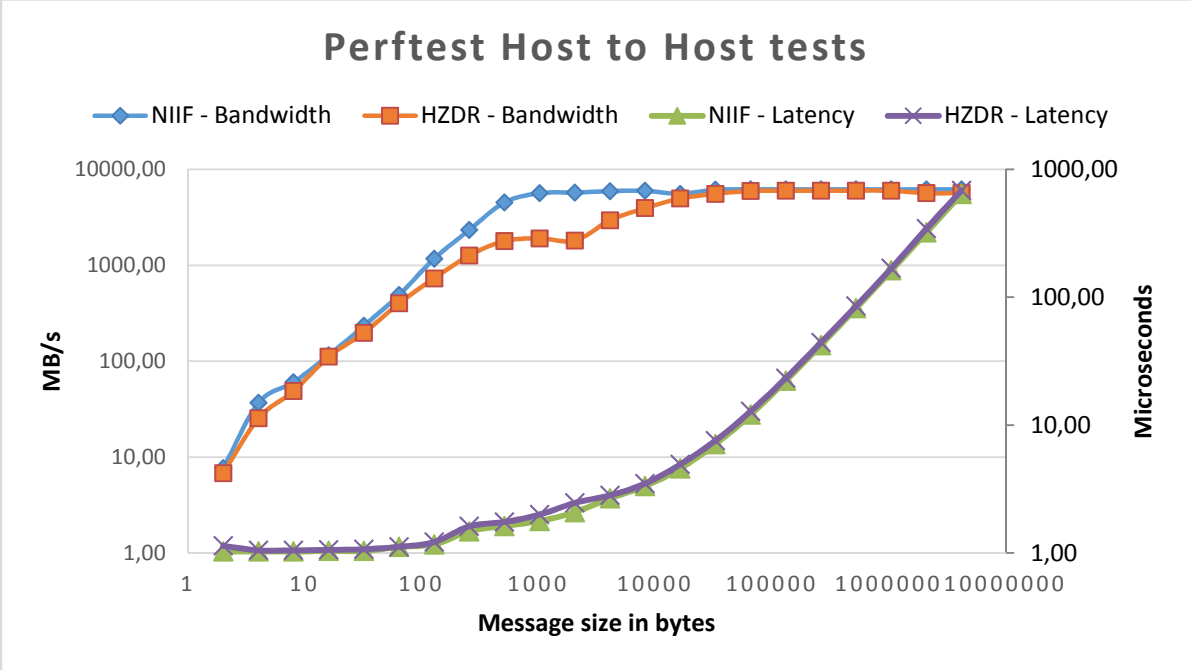

Figure 9: 'Perftest' tests Host to Host at NIIF and HZDR

Device to device

Only at the compute nodes at HZDR we could perform the GPU to GPU tests using 'perftest' benchmark since the package supports only bandwidth tests with CUDA GPU Direct RDMA. This is unfortunately not yet implemented at NIIF.

Conclusion

The results confirm the outcome of the OMB between host to host, but at the device to device tests an unexpected behaviour was discovered. Currently the poor performance of GPU to GPU tests is under investigation.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

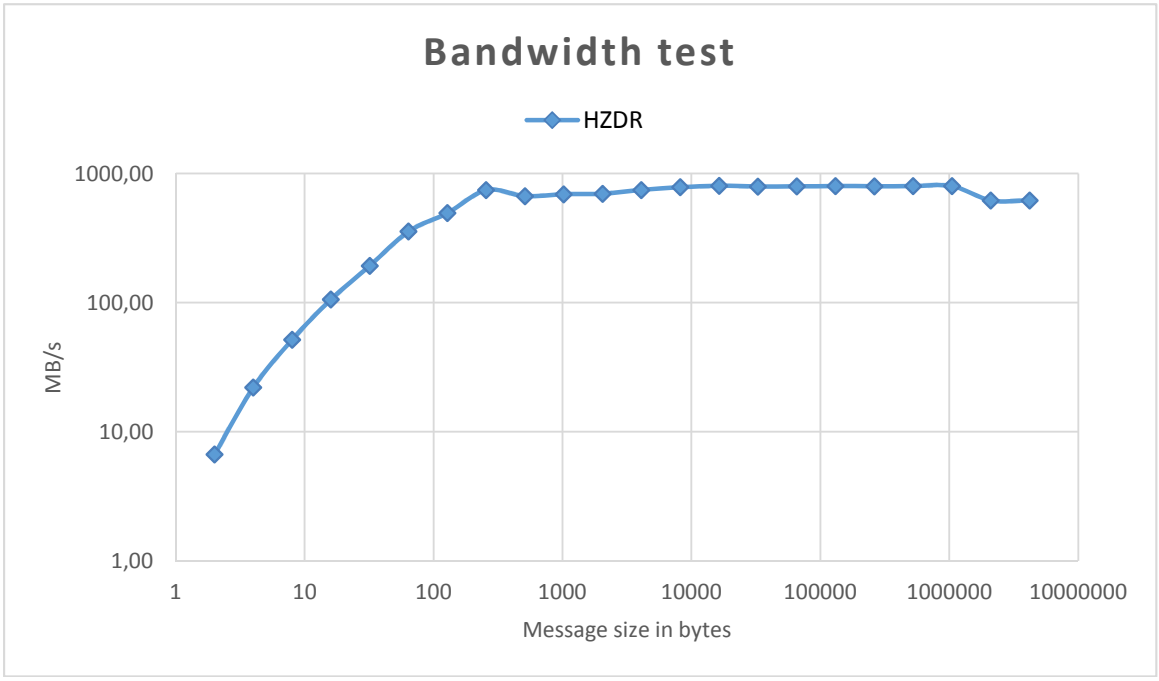

Figure 10: 'Perftest' bandwidth test Device to Device at HZDR

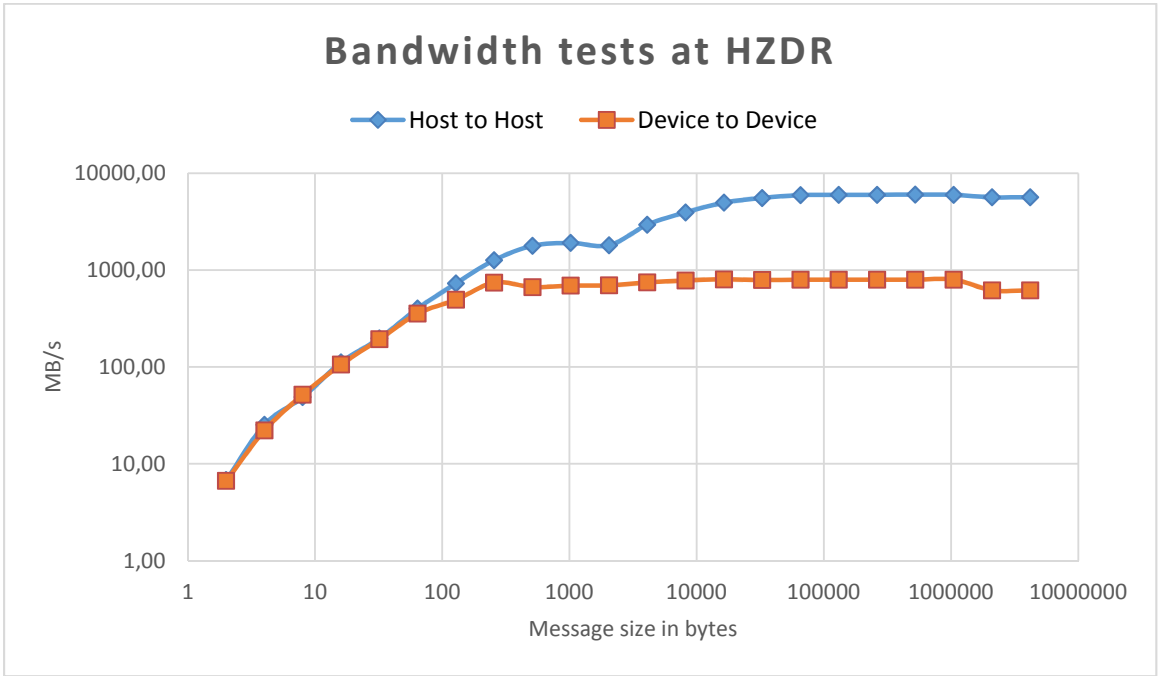

Figure 11: 'Perftest' bandwidth tests at HZDR

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

3.1.5 Conclusions

The results of Host to Host tests show that the Open MPI based data transfer has lower bandwidth than the low-level ibverbs based data transfer for smaller package sizes (size smaller than 16 KiB), however the maximum bandwidth is very similar for both transfer types. Concerning latency, the tests show that the Open MPI based solution has a higher latency than the ibverbs based method. The optimal bandwidth of Host to Host data transfer for Open MPI based solution is found for 16 KiB and 2 KiB for ibverbs.

The Device to Device tests show that the maximum bandwidth is smaller 6317 MB/s vs. 5388 MB/s and latency is higher with the Open MPI based solution. The optimal message size is quite large, i.e. 1 MiB. The OMB tests for Device to Device with and without enabling GPU RDMA showed (Figure 7) that the use of a CUDA based solution is worth, especially for smaller message size (smaller than 32 KiB) where latency can be three times smaller and bandwidth can be ten times larger. For the ibverbs based solution an anomaly was found and therefore the peek bandwidth is significantly smaller 6032 MB/s vs. 835 MB/s.

One may note that the development process for Open MPI based solutions is less complicated for Host to Host and for Device to Device (CUDA based solution) and the Open MPI is well known for developers. Therefore, we would recommend, based on the current results, the use of the Open MPI based solution if the message size reach the optimum level of the Open MPI based transfer (16 KiB) and latency is not critical. For GPU based computational tasks it is recommended to use the Open MPI based solution. The low-level ibverbs based solution is preferable, when the message size is small and latency is critical.

Acknowledgment

We would like to thank 'KIFÜ/NIIF' LEO HPC and HZDR for given access to their HPC facilities. Furthermore, we would like to thank Michael Bussmann, Ferenc Bartha and Zoltan for their support.

3.2 RASHPA (ESRF)

3.2.1 Introduction

Data production and analysis are the essential components at the heart of any scientific experimental application. The produced data rates by x-ray detectors increases significantly, as the technology of the photon sources evolves.

Modern high performance 2D detectors are able to produce very high throughput in the range of 1-100 GBps. Eiger 9-Mpixel and Jungfrau-10 MPixel detectors developed at the Paul Shearer Institute (PSI) are examples of throughputs that can go up to 360 GBps and 400 GBps respectively. Such data streams are challenging to transfer, manipulate and process in acceptable times.

Traditionally, efforts on detector development for photon sources have focused on the properties and performance of the detection front-ends. In many cases, the data acquisition chain was treated as a complementary component of the detector system that was added at a late stage of the project. In some cases, the data acquisition subsystems, although achieving minimum bandwidth requirements were kept relatively simple in term of functionalities, in order to minimize design effort, complexity and implementation cost.

This approach is changing in the last years as it does not fit new high performance detectors; industrial data acquisition protocols do not provide the required data throughput and implementing high performance schemes becomes much more difficult and resource consuming. Detector developers are changing their paradigm and moving into the development and implementation of reusable high performance data acquisition schemes that can be applied to different kind of detector devices.

The design and implementation of efficient data acquisition scheme with multi-gigabyte per second capabilities become a mandatory and unavoidable solution to deal with such data rates. RASHPA (RDMA-based Acquisition System for High Performance Applications) is the generic data acquisition framework currently under development at the ESRF. It is optimized for the transfer of 2D detector data, i.e. images, metadata, etc., relying completely on Remote Direct Memory Access (RDMA) mechanism. Therefore, RASHPA is able to push data, at maximum throughput into the address space of one or several destination backend computers. RASHPA scheme provides a high standardization level in the data transmission pipeline from the detector up to the software application for further processing, visualization or storage.

3.2.2 Objectives

The design of RASHPA framework pursues three main objectives discussed as follows:

Promoting standardization and reusability

The design and implementation of an efficient data acquisition schemes with multi-gigabyte per second capabilities and high-level functionality is far from trivial. Therefore, it is of

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

overriding importance that the invested effort can target a variety of detector systems and over a long-lasting period of several decades. One of the key reasons behind that is the long-term development for x-ray detectors that can easily take a decade from concept to product. With this purpose, the data acquisition scheme must focus on conceptual and functional aspects and should minimize the dependency on a given technology. This is particularly important in what respects data link hardware and protocols so that RASHPA should be able to adopt future standards, following and profiting from the evolution of data communication technology.

The proposed framework should be independent of the particularities of the detection front-end and other instrument features that are not directly related to the data transmission process. In this way it must be possible to design detectors, in which, the RASPHA related functionalities could be included in a reusable functional block with well-defined interfaces.

In order to simplify the system development without compromising the performance target, the rich functionalities provided by RASHPA must be managed at the backend side by detector independent code that provides an interface sufficiently high-level to be easily exploited by the diversity of detector software applications that may be used in the full detector systems. This approach is expected to greatly reduce software development efforts and make the data acquisition framework much more attractive for detector developers.

Addressing a wide range of implementations

Most detector developments for photon sources do not target a single scientific application. It is therefore frequent and convenient that the same detector technology and components are used for various applications. These applications may have rather different data-throughput and performance requirements as well as their own set of technical and cost constraints. In practice, this is addressed by producing different implementations of the detection systems. It is therefore very important that a generic data acquisition framework as RASHPA can fit in such a scheme and that the future compliant detectors are not restricted only to very high end configurations.

Scalability is therefore a major issue. In the most demanding experiments, it must be possible to build high performance RASHPA based detectors able to send data to computing farms by using large number of the fastest data links available at the time of the development. At the other hand, it must be possible to scale down the systems to rather simple and relatively inexpensive configurations. One important goal behind RASHPA is that the system could be rescaled by reconfiguration of the components and not requiring any hardware redevelopment.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

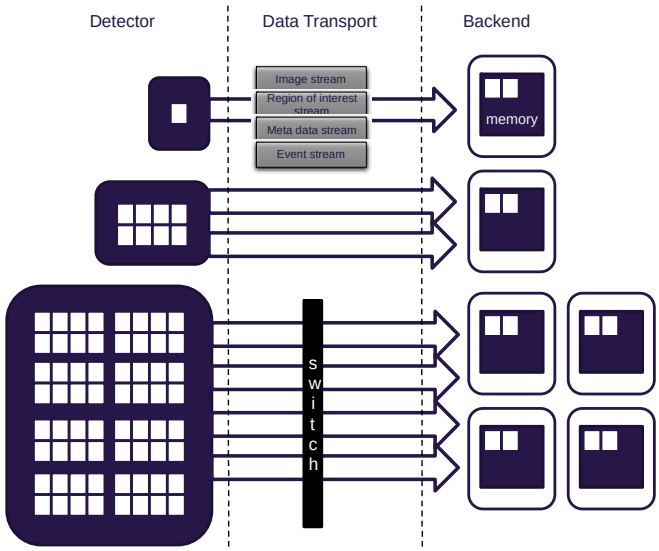

Figure 12: Block diagram of RASHPA's architecture

Alleviating the data management challenge

Moving data at high rates from the detector head to memory buffers in the computer infrastructure is a necessary step to take advantage of many of the most advanced detector systems. This does not solve other difficulties related to data management and manipulation. The effective exploitation of such very high-throughput data streams is extremely challenging and must consider all the on-line data management aspects from data generation and transmission, to data dispatching, visualization, analysis and storage.

Therefore, RASHPA data acquisition framework should support the following features depicted in Figure 12:

1) Multiple sources and destinations via routable network topology based on high speed data links.
2) Parallel and simultaneous data streams.
3) Remote direct memory access (RDMA) zero copy transfer features
4) Potential support of image manipulations and other hardware accelerator algorithms at hardware level

As previously mentioned, RASHPA is conceived to serve as basis in the design of the data acquisition mechanism for a diversity of detectors. For the sake of reusability and longevity, the framework itself imposes very few constraints on the choice of technologies employed for the implementation of the actual detector devices. With this approach, it is expected that RASHPA will be applicable to a larger scope of detector developments and will take advantage of future progress in electronics and data communication technology. It is also foreseen that detector software implementations that use RASHPA as its internal machinery for data acquisition purposes will provide interoperability with different detector designs, and will be usable with various generation of detectors.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

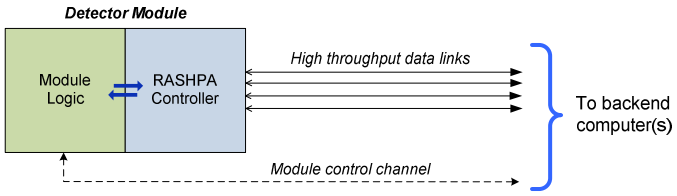

Figure 13: Basic scheme of RASHPA detector module

3.2.3 Key Concepts

RASHPA relies on several actors in order to be implemented. The descriptions and terminologies defining those actors are the following:

1) Detector Module (DM): is a core unit in detectors that includes functional block, called RASHPA Controller (RC) responsible of transferring data to Backend Computers (BC) via high-speed data links. Figure 13 shows a basic scheme of a RASHPA detector module.
2) Multi Module Detector (MMD): Is a detector consisting of one or more identical DMs.
3) Backend Computers (BC): These are the final destinations of the data. In RASHPA terminology, all the computers that receive detector data are called data receivers (DR). The backend computer that is in charge of the configuration and initialization of the data acquisition subsystem is called the system manager (SM). Figure 14 shows a basic scheme of a RASHPA backend computer. A C-Library responsible for RASHPA's operation, called LIBRASHPA, is integrated in each BC.
4) RASHPA Telegrams (RT): XML telegrams used to retrieve the capabilities and configuration of DM and DRs. RTs are sent to the software library prior to the configuration of the RCs present in each DM.
5) RASHPA Address Space (RAS): it is a single common address space that remaps all the memory areas of all the DRs. It appears to be a contiguous block even if it may include regions from more than one computer. It could also mix system RAM memory buffers with other memory even from an address space of one DR. It is constructed by dedicated functions in the RASHPA software library.
6) RASHPA Buffers (RB): consists of a set of address areas or blocks named local buffers (LB) that reside in the address space of one or more DR. The simplest case of RB would be a conventional buffer declared in the system RAM area of one DR.
7) Data slices, data sets and data blocks: Detector data are organized in sequences, produced and numbered consecutively, called data slices (DS). A DS can be consisted of data produced by DM during a time interval.
8) Every type or class of data produced by DM and treated separately is called a dataset (ex. Image, metadata, etc.).
9) All the data from a given dataset, produced within a data slice are called Data Blocks (DB).

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

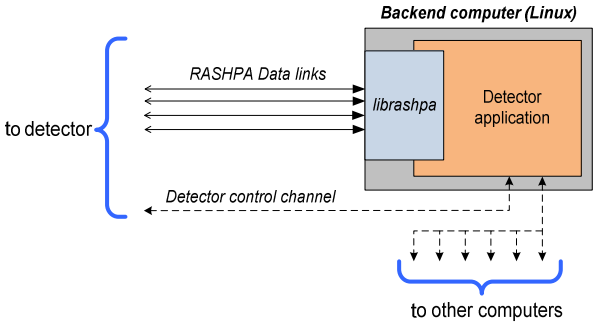

Figure 14: Basic scheme of RASHPA backend computer

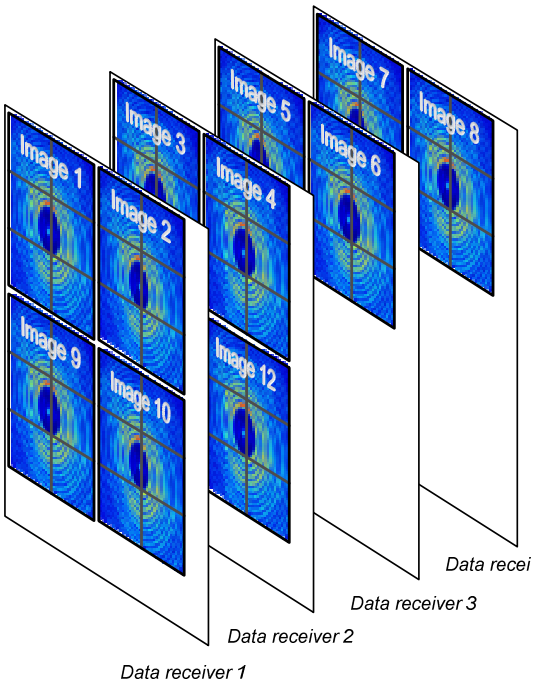

Figure 15: Example of a DTP result from a segmented 2D detector composed by six independent modules to four RASHPA BCs

10) Data Transfer process (DTP): is a description of the direct transfer from DM to RB in a given DR. It supports RDMA and guarantees data integrity. In other words, DTP specifies what to send, where to send and how to send data. An example of a DTP result from a segmented 2D detector composed by six independent modules to four RASHPA BCs is shown in Figure 15.
11) Data Channels (DC): These are functional RASHPA blocks, responsible for data transfer from DM to destination buffer. A DC is configured by DTP. In most general case, a DTP requires the activation of at least one DC for each DM.
12) Event channels: These are functional RASHPA blocks responsible of sending events to the BCs. Events are asynchronous messages generated by the RC in the DM. They are used to signal the detector software application about errors, change of status, progress of the running DTP, etc.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

3.2.4 Hardware requirements

In order to build a RASHPA compliant detector and software application, some requirements should be respected at the data link, the detector module and the backend computers.

Data Links

The transfer of data between the detector modules and the backend computers is achieved by fast links that fulfil the following requirements:

1) Direct memory access (DMA): The full address space of the backend computers should be accessible from the RASHPA controllers by RDMA mechanisms. In RASHPA scheme, the effort and complexity is put on the initialization and configuration of the system; once the data transfer operations started there is no need of any intervention by the data receiver's CPUs.
2) Asynchronous event signalling: The selected data links must provide a low-level asynchronous mechanism to signal conditions that trigger RASHPA events. The conditions activated in each case are selected by LIBRASHPA as part of the DTP configuration.
3) Data integrity: RASHPA requires that the data link layer implement all the mechanisms necessary to achieve data integrity during the transfer process. It may include any packet ordering, retransmission or error correction schemes as needed.
4) Bidirectional operation: The data links operate as unidirectional channels from the detector towards the backend computers. However, the use of bidirectional links would allow future functional extensions of the framework such as high-level flow control mechanism or embedded control channels. In any case, the data bandwidth requirements would be very asymmetric and in order to minimize misuse of the links, they should be used as write only channels and avoid read operations.
5) Data switches: The link technology must be compatible with the implementation of data switches for multiple-host configurations. The data switches must provide mechanisms to both write data and transmit asynchronous events to the backend computers.

There are no specific requirements on data bandwidth or other minimum performance figures for the capacity of the data links. However, a main goal of the RASHPA framework is to allow very high throughput data transfers. It is therefore expected that the RASHPA based systems will use advanced data links for which the implementations will evolve in the future following the evolution of the data communication technology.

From a purely functional point of view, the link technology considered the most suitable to be used in a RASHPA system is PCI Express over cable, and it is the choice adopted for the implementations of the first prototype. PCIe is the natural extension of the internal bus of the backend computers and it does not need any software protocol. The hardware and the operating systems of the computers take in charge the full initialization of the data links that

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

provide data integrity and minimum latency. In addition, PCIe endpoint can be integrated in the detector modules with a reasonable design effort. Disadvantages of using this link includes the limited availability of commercial off-the-shelf data switches and PCIe links based on optical cables as well as the small packet transfer size. Therefore, other candidates based on more widespread technologies, such as Infiniband or Ethernet, should not be discarded. Although Ethernet would need to be extended with remote DMA protocols (RDMA) such as iWARP or RoCE.

Detector Modules

The RASHPA framework specifies the functionality of the detector modules but not the internal resources required to achieve such functionality.

In the case of DM with multiple data links, the DM must either, know in advance or detect at runtime, how many data links are physically connected and in operational state. A module must be able to operate even if only part of the data links is operational during DTP initialization. All the operational data links must be able to access the full RAS. The framework does not specify how the data write operations are shared among the various links, this is the responsibility of the module developers.

A detector module must implement the required functionality of a RASHPA controller and integrate, in addition to the fast data link interfaces, high-level configuration features and powerful data manipulation capabilities.

Backend Computers

All the backend computers in a RASHPA based data acquisition system must be based on the Linux operating system and run detector software that include and use one instance of the LIBRASHPA library. The backend computer that acts as SM plays a central role for initialization, configuration and overall system monitoring. The SM must manage or at least have access to the control link.

All the BCs must also include the data link hardware interfaces that are required to implement the configuration selected for the particular application. Both the SM and DR must be able to accept and treat the asynchronous messages from the DMs that trigger RASHPA events.

The DRs must be able to map RASHPA buffer in their system address space. The data destination buffers will be in most of the cases large memory areas in system RAM but they may also consist of buffers in extension boards such as GPU or FPGA coprocessors or disk controllers.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

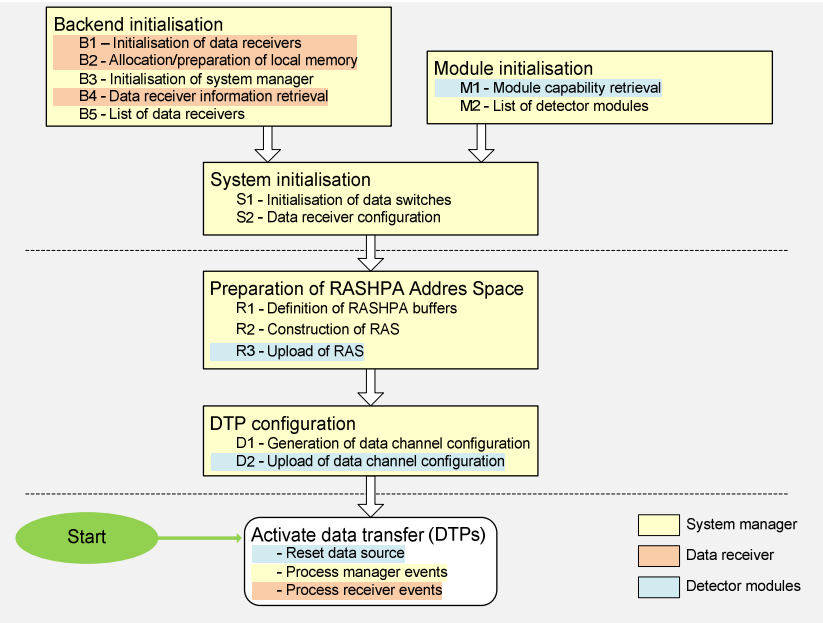

Figure 16: Functional steps in RASHPA system

3.2.5 System functional description

RASHPA framework defines the conventions, procedures and functionality that a compliant data acquisition system must implement. DMs must be configured at the initialization phase and operate as expected by the DTP. At the backend side, the detector application software uses the high-level specific functionalities provided by LIBRASHPA library to comply with the framework. It is the responsibility of the detector applications to orchestrate the full operation of the complete system. This is done by properly combining the control of the detector with the calls to LIBRASHPA functions. The detector application is also responsible of the intercommunication with various computers in the case of multiple backend configurations.

Figure 16 illustrates the different steps and partial operations in a RASHPA system. The figure shows the dependencies of the various steps and the order in which the detector application has to handle them.

When a detector application repeats acquisition and data transfer sequences not all the operations have to be done again and the application will normally loop across the operation flow in Figure 16.

The initialization of the system must start by the initialisation of the main system components: the backend computers (B1 to B5) and all the detector modules (M1, M2). Once all those individual components are initialised the system manager may proceed to complete this first phase by initialising the data switching system (S1) and dispatching the configuration of the communication system to all the data receivers (S2). At this point the system is ready to start the configuration phase.

The first step in the configuration phase is the definition of the RASHPA buffers (R1) that will be used to receive the data streams during the data transfer operations. This step is

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

necessary to generate and upload the RAS map (R2, R3) into all the detector modules. The next configuration step is the generation of the configuration of the DTPs (D1) that must be uploaded into the detector modules (D2).
At this point the system is ready to start data transfer by activating the configured data channels.

3.2.5 Software library

LIBRASHPA is the piece of software in charge of managing all the specific aspects of a RASHPA based system. LIBRASHPA is thought to be a Linux library written in C language that must be included in the detector software applications such as LIMA data acquisition and control library used at the ESRF and other synchrotron radiation facilities [10].
The main duties of a LIBRASHPA manager are:

1) Compiling the capabilities of all the DMs and the information from all the DRs in the system.
2) Initializing backend components, such as data switches in case of routable network.
3) Building and maintaining the RASHPA global address space.
4) Managing the configuration of the data reception buffers and provide the information required by the DRs.
5) Providing the configuration of the detector modules from the definition of the requested DTPs.
6) Managing and triggering system wide events and passing information to the detector application.

The duties of a LIBRASHPA for data receivers include allocating physical memory for data reception, triggering receiver events and passing the information to the detector application.

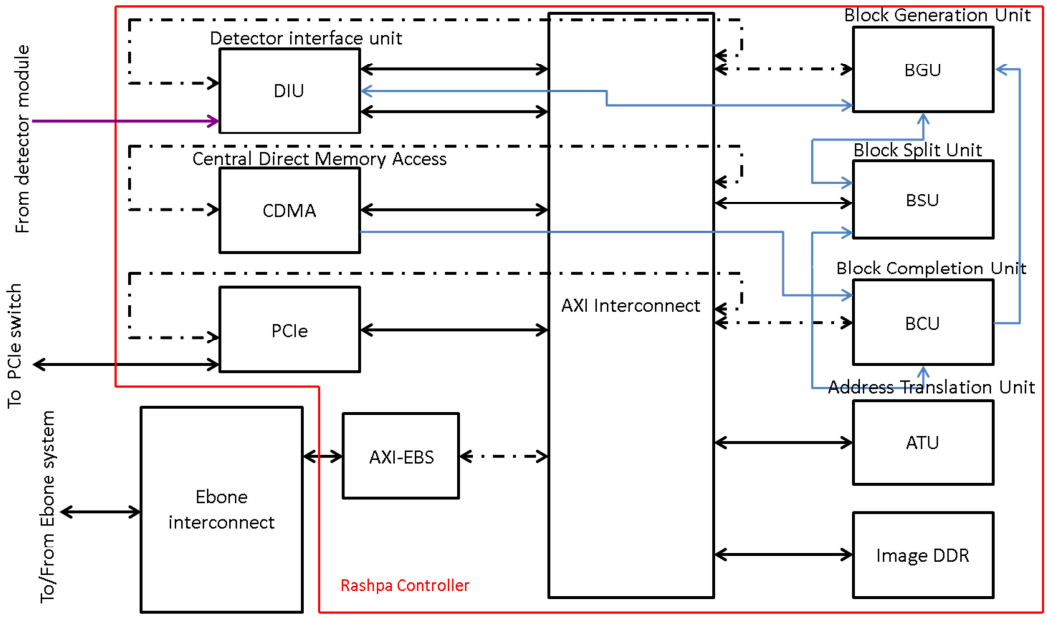

Figure 17: Architecture of a RASHPA FPGA implementation

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

3.2.6 RASHPA prototypes and results

Two types of implementation should be identified: hardware and software implementations. From the hardware point of view, the FPGA implementation of RASHPA is independent of the type of high speed data link used to perform the data acquisition process.

The FPGA implementation presented in the following paragraphs is based on an AXI subsystem implemented on a Xilinx KC705 development board. It is considered as a separate intellectual property (IP) that can be easily integrated within the ESRF control systems. In this implementation, a PCIe over cable data link was chosen for the reasons mentioned in the introduction.

The AXI subsystem implementing RASHPA on the KC705 development board, Figure 17, consists of several IPs, some from Xilinx and others designed ad-hoc:

- Detector Interface Unit (DIU): This unit reads the image data and store them in a DDR3 memory on board.
- Block Generation Unit (BGU): It contains all the data channels that are configured prior starting the data transfer process. This unit outputs 256-bit packets containing all the information about each transfer, such as source address in the DDR, destination address index of a physical address in an address translation unit, bytes to transfer, etc.
- Address Translation Unit (ATU): This is an internal memory containing the physical address of the available memory buffers on each DR.
- Block Split Unit (BSU): It is responsible of reading and analysing BGU's output data, getting corresponding addresses from the ATU, and configuring the Central Direct Memory Access (CDMA), and the PCIe IPs.
- CDMA: a Xilinx built-in direct memory access IP.
- PCIe: a Xilinx built-in transfer layer IP

All the previously described IPs are configured through the e-bone interconnect [10] which is the ESRF standard interconnect used for control applications.

A RASHPA system must be able of sending data to multiple destinations, thus it requires a routable network. As this implementation has adopted PCIe for data transmission, one requires to use PCIe switches for packet routing. Such components are not so common, but one could find some of them such as the Dolphin IXS600 from dolphinics and OSS-PCIe-1U-SW-x4-2.0 from one stop systems.

For the demonstration purpose, we made the decision to use the PXH810 board which integrates a PLX8749 PCIe switch.

The board comes with a Linux driver that is not compatible for RASHPA's application. We have then re-implemented the driver of the PLX switch in order to fit the RASHPA needs. The PXH810 board is plugged in a BC, with one PCIe cable adapter board. The adapter board links the current BC to the DM, whereas the PXH810 connects the current BC to another BC. This way, RASHPA can send data to two BCs via the PCIe switch.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

The workstation middleware called librashpa, a Linux library, is in charge of providing the client application with data transmission services through a well-defined application programming interface (API). An example of client is the LIMA data acquisition and detector control library widely used at ESRF and other synchrotron radiation facilities. The middleware layers depicted in Figure 17 support the following characteristics:

- Configuration: this layer is in charge of building description of the hardware such as when and where to transmit data.
- Device abstraction: Since RAHSPA is link independent, the PCIe endpoint discussed earlier can be replaced with any other data link such as Ethernet, Infiniband, etc... The device abstraction layer hides low-level details and provides the software with a generic interface to access the underlying devices. While most of the layer is implemented in user space, special operations such as interrupt handling require a thin driver.
- Memory management: RDMA transfer requires to work with the BC's physical address space, however typical client application buffers are allocated in the process virtual address space. The middleware thus provides a virtual memory allocator on top of an internally managed physical memory allocator.
- Unified Event Model: There are multiple sources of events that would make sequential programming difficult. For instance, data transmission completion is signalled by the detector using an interrupt along with auxiliary data; LINUX informs about PCIe device adding or removal using system notifications; Timers and callbacks may be registered by the client application itself. For those reasons, the middleware programming model is largely event based and provides software abstractions that unify the different event sources.

The first RASHPA prototype was developed in the frame of the European project CRISP. In the first version, RASHPA supported the data transfer from a DM to one BC. It has been tested and validated using a data generator emulating the detector behaviour.

In the current version, Figure 18, RASHPA has been integrated to the SMARTPIX, a Medipix3 based detector currently under development at the ESRF. In that perspective, the KC705 development board has been replaced with a commercial PFPKX7 board from Techway. The new of prototype version also supports the multiple destinations feature thanks to the use of PCIe switches. Copper cables were used to build and test the routable network although fibre optics cabling is also available. In this implementation, two types of PC are used: a so-called detector PC having 64 GB of internal DDR, and Gen3x16 PCIe endpoint, and an industrial PC with only 4 GB on internal DDR and Gen1x1 PCIe endpoint. The PFPkx7 supports Gen2x4 PCIe, thus the link connecting the detector to the detector PC is negotiated to Gen2x4 whereas the virtual link connecting the detector to the industrial PC is limited to Gen1x1.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

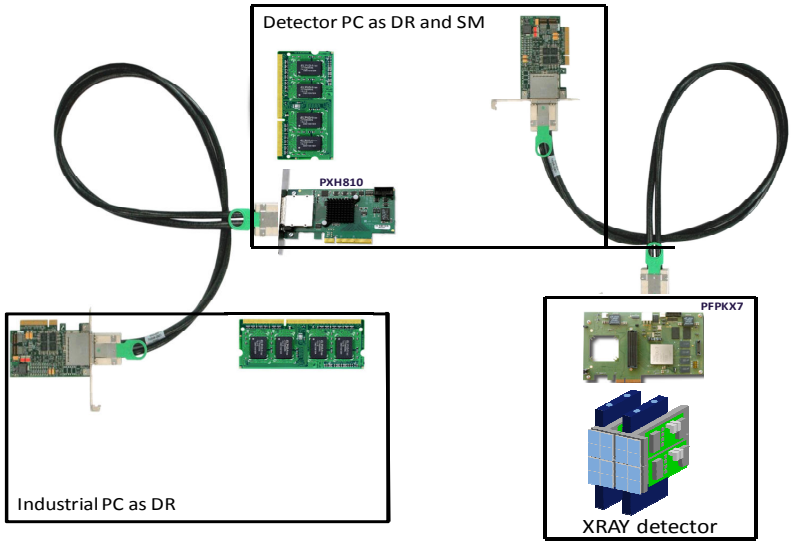

Figure 18: RASHPA prototype

Due to the difficulty of producing real detector images with the current SMARTPIX hardware without an x-ray source, a Java image generator application was developed, where the ESRF logo was used as a reference image. A screen shot of an experiment is illustrated in Figure 19. In this experiment two data channels where configured to send the original image to the detector PC and a region of interest of that image to the industrial PC based on some configuration sent by the system manager to the RASHPA controller.

Measured throughput when Gen2x4 PCIe endpoint is the target is 68% of the Gen2x4 maximum bandwidth when measured within the FPGA and 51.15% when measured at the application level. Similarly measured bandwidth at the Gen1x1 is 71.99% and 56.28% when measured at the FPGA and application level respectively. These losses in the data throughput are due to the configuration of the DMA which is imposed by RASHPA as well as the restriction of the PCIe packet size limited to 4 Kbytes. Some other latency can be added when throughput is measured at the application level due to the processor execution time.

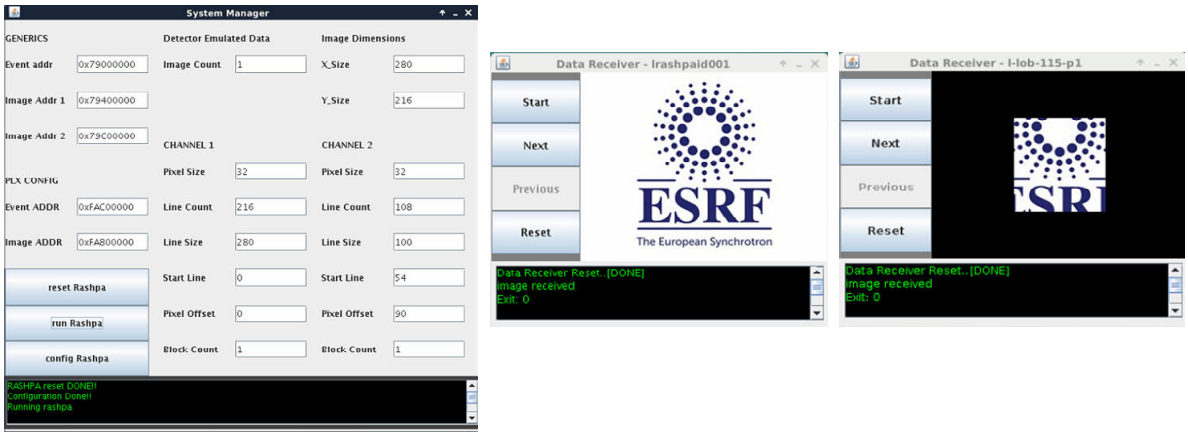

Figure 19: RASHPA prototype

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

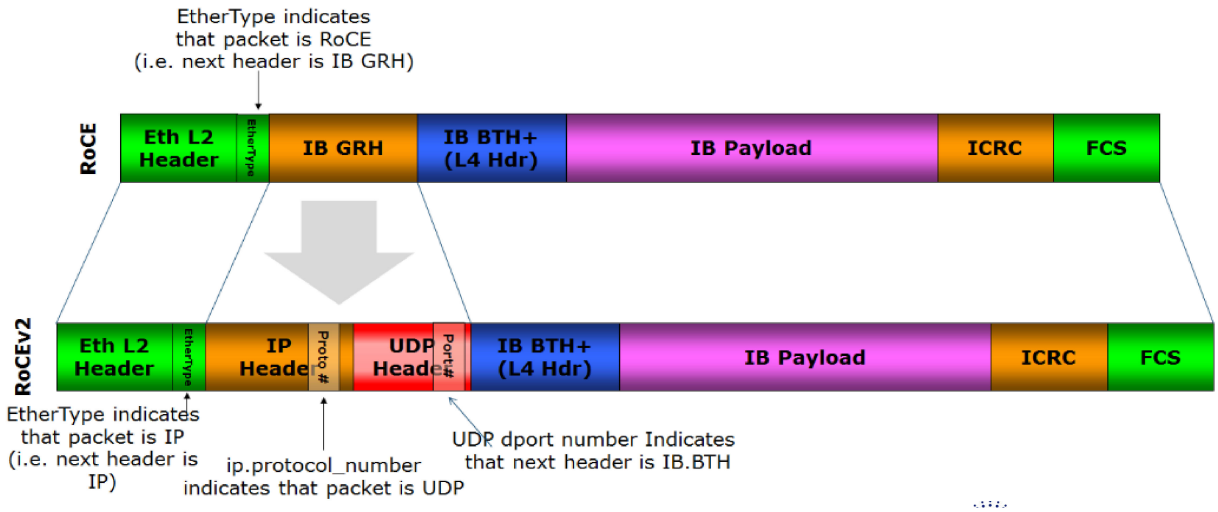

Figure 20: RoCE-v1 vs RoCE-v2 packets

3.2.7 RDMA-based data acquisition over 100GbE

Ethernet is a computer networking protocol introduced in 1983 and standardized as IEEE 802.3. It divides the data stream into shorter pieces called frames. Each frame contains source and destination Media Access Controller (MAC) addresses, Ethernet type, data and error-checking code for the frame data.

The Ethernet type field specifies which protocol is to be included in the frame. Internet Protocol (IP) is one of these communication protocols and is the level 3 in the Open Systems Interconnection (OSI) model which constitute the Ethernet communication standard. User Datagram protocol (UDP) is one of the essential communication protocols used by the IP protocol. The UDP frame consists of several fields in addition to the Ethernet header and the IP header: source port, destination port, length, checksum and payload data.

RoCE (RDMA over converged Ethernet) is an ethernet protocol based on the Infiniband specification, and available in two different versions: RoCE-v1 and RoCE-v2 or routable RoCE. RoCE-v1 is an Ethernet layer non routable protocol whereas the routable version is the most interesting for RASHPA's implementation.

RRoCE is an RDMA capable, layer 3 network based on UDP/IPv4 or UDP/IPv6, and relying on congestion control and lossless Ethernet. It is currently supported by several off-the-shelf network adapters as well as the latest Linux kernel drivers.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

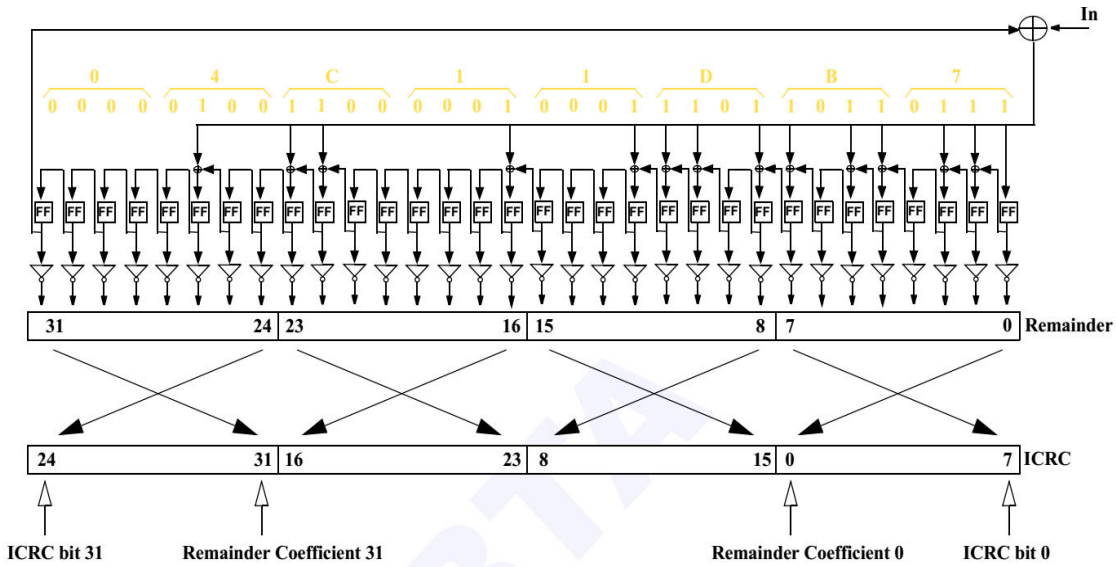

Figure 21: Calculation of invariant CRC for RoCE protocol

RASHPA allows detectors to push data (images, regions of interest (ROI), metadata, events etc...) produced by 2D x-ray detectors directly into one or more backend computers. RASHPA's main properties are its scalability, flexibility and high performance. It is intended to have an adjustable bandwidth that can be compatible with any backend computer.

The UDP payload data of a RRoCE protocol, illustrated in Figure 21, contains an infiniband header, the actual data payload in addition to an invariant cyclic redundancy check (iCRC) field that is mandatory for the RoCE packets in order to be accepted by the network adapter. The iCRC field is retained from the Infiniband specifications. Note that, an Ethernet frame does also contain another CRC field for the global packet.

The calculation of the iCRC algorithm for RoCE-V2/IPv4 is performed following the below steps:

- Extract RoCEv2: IP+UDP+InfiniBand.
- Add Dummy LRH field, 64 bits of 1s.
- For RoCEv2 over IPv4
- Time to Live = 1s
- Header Checksum = 1s
- Type of Service (DSCP and ECN) = 1s
- UDP checksum = 1s.
- Resv8a = 1s
- CRC calculation is based on the crc32 used for Ethernet networking, 0x04C11DB7.
- CRC calculation is done over the UDP frame starting from the most significant bit of the most significant byte.
- Inversion and byte swap has to be applied in order to get the invariant arc to be integrated in the RRoCE frame

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

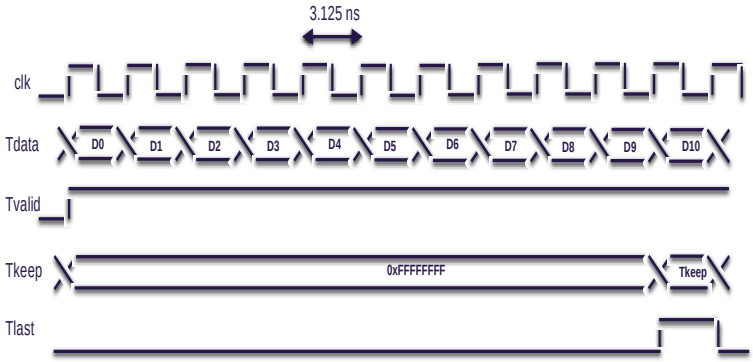

Figure 22: Timing diagram of the iCRC calculation

A first FPGA implementation trial of the RRoCE has been performed using the unreliable datagram mode (UD). In this mode data are sent in streams without any acknowledgement from the receiver side. The target FPGA board was the KCU116 by Xilinx and the target network adapter was a Mellanox ConnectX-4 (MCX415A-CCAT) board. It is important to note that in Ultrascale+ families, the 100G CMAC IP core is a hard IP having LBUS (Local BUS) as input/output, which is converted into AXI stream bus to be integrated in system on chip designs.

In fact, the basic challenge in the FPGA implementation of a RRoCE algorithm is the optimal implementation of an iCRC algorithm. Figure 22 depicts the timing diagram of the design. Data of 64 bytes are streamed at each 3.125 ns clock cycle period except the last cycle that may contain partial data that requires multiplexing via the AXI stream “tkeep” signal for byte selection.

A pipelined iCRC design requires 64 clock cycles in order to calculate the iCRC over the 64-bytes input. After 64 clock cycles, the design will be allowed to continue the calculation over the second 64-bits input data. That means that 200 ns are lost for each data calculation of 64 bytes. Supposing that the transmitter sends 12.5 GB (100 Gbits) of data, that will theoretically take one second to be transferred over a 100 Gbps Ethernet link, the actual theoretical transfer delay caused by the iCRC calculation will be 42 ms that is 4.2%.

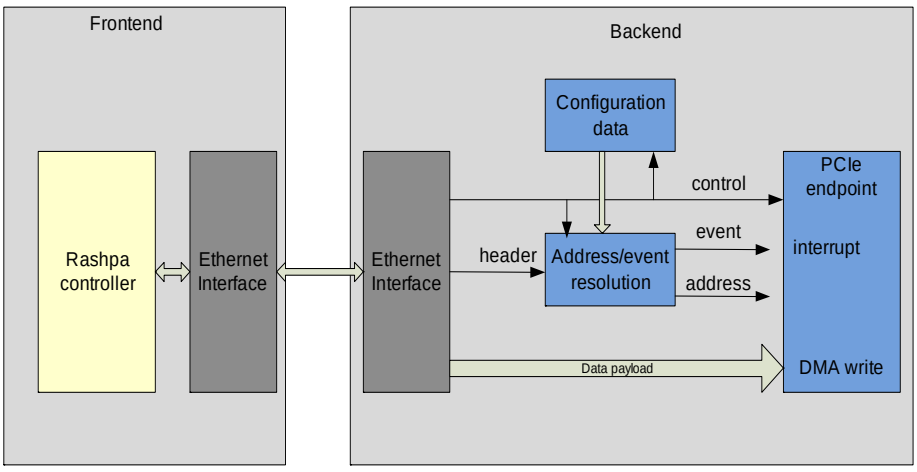

Figure 23: Architecture of the proposed RDMA over Ethernet protocol

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

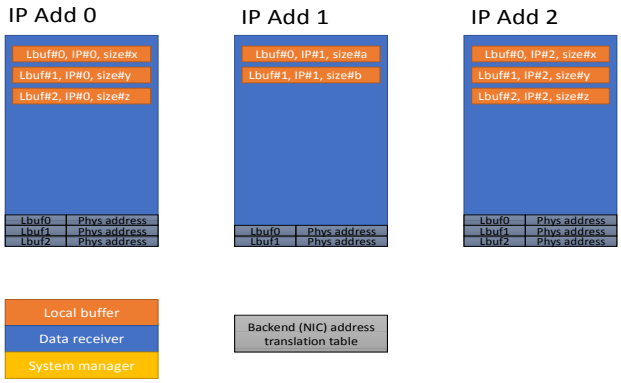

Figure 24: Representation of local memory buffer in the backend computers

RRoCE is a well-developed commercial protocol supported by the ib-verbs library available in the latest Linux kernels. However, one can even go faster in data transfer due to the iCRC calculation problem and the overhead used for the Infiniband header. In addition to the previously mentioned reasons, controllability and observability over an in-house developed protocol is a major advantage for an ESRF RDMA over Ethernet protocol over RoCE.

The proposed RDMA over Ethernet standard proposed in this paper will mainly use the UDP/IP protocol for routability, and information about each transfer in the unused source and destination ports of the UDP header.

The proposed standard relies on the interactions of three major actors. The first one is the RASHPA controller on the x-ray detector side which is the data transmitter. The second one is the FPGA board acting as a data receiver, which will transform UDP packets coming from the transmitter side into PCIe DMA-based packets and sent to some buffers on the data receiver computer which is the third actor in the system. Figure 23 illustrates the architecture of the overall system.

There will be a software library called LIBRASHPA installed on the data receiver side that will help allocating memory buffers of different sizes to be used as final data destinations. These buffers will be identified by an identification number (ID), a size, and the IP address of the data receiver as depicted in Figure 24. The RASHPA controller, which is the transmitter, should have enough knowledge about these three parameters, however the receiver FPGA board should store the real physical address of the allocated buffers for address translation.

Figure 25, shows the FPGA implementation of the Ethernet transmitter side via the Xilinx 100G cmac IP. Data streams coming from the detector are stored in a DDR4 memory. Whenever a full image is written to the DDR, the RASHPA controller will configure a Direct Memory Access (DMA) IP allowing it to read the data via an AXI4 interconnect, and sends it as stream of data (AXI stream bus) to the header insertion IP. The header insertion IP gets its configuration from the RASHPA controller. In fact, the configuration of the header insertion unit is nothing but the UDP header and the destination local buffer represented by the following parameters stored in an internal Block RAM (BRAM) during the configuration

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

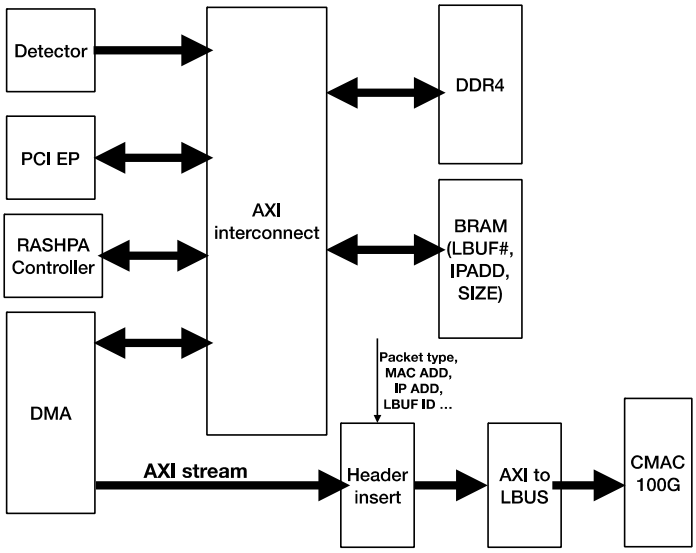

Figure 25: FPGA implementation of the RDMA over 100G Ethernet transmitter side

phase: the ID, the IP address, as well as an offset respecting the size of the buffer calculated by RASHPA controller. The constituted header will be concatenated with the data stream coming from the DMA. Since the CMAC IP has a local bus (LBUS) input/output interface, a bridge between the AXIS to LBUS has been implemented and used as an intermediate stage between the header insertion unit and the CMAC IP. The configuration of the whole process can be done using the same Ethernet link or via an external link such as 1 GB Ethernet, PCIe over cable, etc.

At the receiver side, Figure 26, the CMAC output data as LBUS are bridged to an AXI stream interface before it gets analyzed in order to resolve the physical address of the final destination buffer. Actually, during the configuration phase, LIBRASHPA should store the physical address of each local buffer in a BRAM inside the receiver's FPGA. The output data of the header analyzer unit can be stored in a DDR4 or FIFO for synchronization, then sent to the PCI express endpoint for DMA transaction to the final destination. The whole process is controlled by a finite state machine implemented in the driver IP.

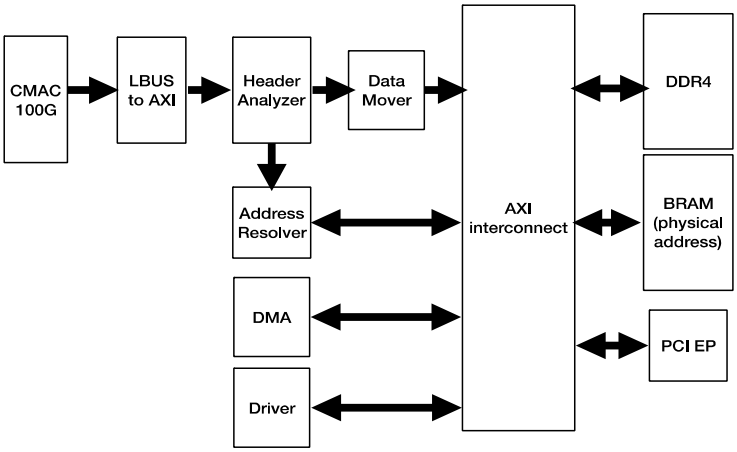

Figure 26: FPGA implementation of the RDMA over 100G Ethernet receiver side

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

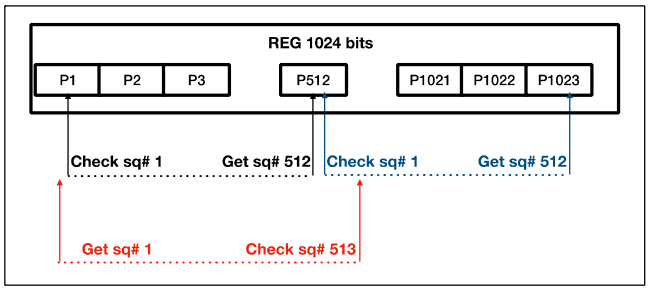

Figure 27: Packet loss detection algorithm

In order to guarantee no packet loss, one can use a converged network, but in case of lost packets, the data receiver should be informed. For that, a simple packet loss detection algorithm has been implemented. It consists of a 1024-bit shift register, each bit in this shift register represents one packet number represented by its sequence number. When packet sequence number "512" is received, the receiver looks for packet 1 if it is missing than it generates an event to inform the data receiver. The same process repeats for each received packet, which means that the receiver can identify a lost packet after 512 packets. The process is illustrated in Figure 27.

The implementation of the proposed prototype as well as RoCE-V2, targets a Xilinx FPGA development board (KCU116) which is based on the XCKU5P Kintex Ultrascale+ family. In case of the proposed prototype, the receiver implementation targets an industrial board called XpressVUP developed by Reflexces. It is based on a XCVU9P virtex ultrascale+ FPGA with an integrated Gen3x16 PCIe endpoint. The PCIe endpoint is comparable to the integrated one in the Mellanox network adapter card, MCX415A-CCAT, used as a RoCEV2 backend. The FPGA boards integrate a hard MAC IP core supporting 100 GB Ethernet. A UDP stack has been implemented on the FPGA allowing the RASHPA controller to construct the frames of data and the back-end to read these packets and analyze them before transforming them into DMA configurations. Post route of the front-end (transmitter) FPGA implementation show that the design occupies around 50% of the total CLBs and 21 % of BRAM of the selected XCKU5P FPGA.

To confirm the correctness of the constructed packets and to test the transfer bandwidth, the Mellanox NIC was used together with wireshark software on a PC running on Linux debian distribution.

The realized experiments allow building correct UDP packets, however the UDP receive buffer overloaded when measuring UDP bandwidth due to the high transfer rate without the ability to empty it. Hardware RoCE-V2 as well as soft-RoCE were also tested between two Mellanox boards running at 100 GBps.

In order to provide a fair comparison of the transfer throughput of both protocols, one should exclude the CRC implementation because it will terribly affect the transfer rate.

First of all, and in order to have an idea about the transfer one could achieve with the 100G link itself, FPGA to FPGA UDP transfers were selected. Different configurations of the MAC

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

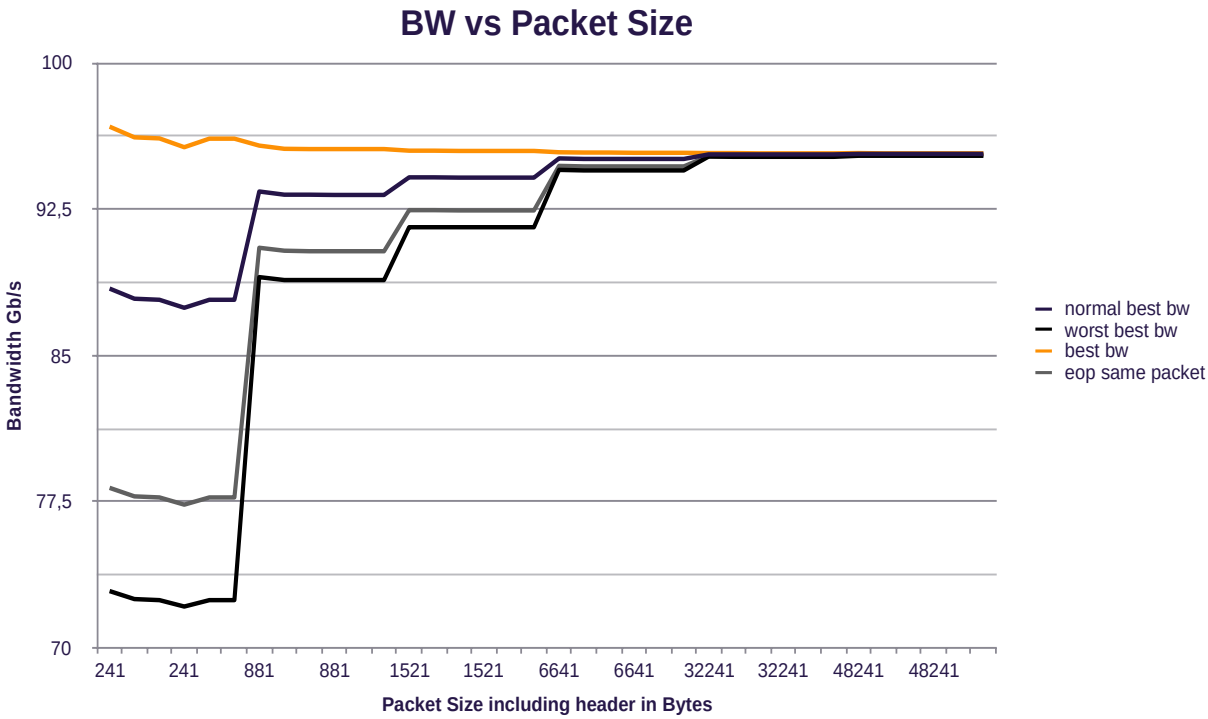

Figure 28: Bandwidth Vs Packet size for the 100GbE link

IP including packet sizes and number of packets to send were selected. Figure 28 illustrates the obtained results, and shows that the 100G transfer can reach a rate of 90 GBps for a minimum packet size of 1 kB and becomes stable at 95 GBps for packet sizes of 32 Kb and above. Small packet sizes decrease significantly the throughput.

The throughput comparison between RoCE and the proposed algorithm was based on pre-constructed data packets of 598 bytes. The same configuration was adapted for both algorithms where a computer was used to configure the DMA on the transmitter side for each transfer. Note that this is not the ultimate throughput to measure because of the CPU interaction at each packet. Table 1 shows the measured bandwidth for both algorithms using the adopted strategy.

Results show that the proposed algorithm is more than 1.5 times faster than the RoCE-V2 protocol considering that the iCRC is pre-calculated and only the link is tested together with the receiver side, i.e the Mellanox network adapter versus the FPGA implementation of the suggested protocol. Both receivers are connected via PCIe x16 lanes.

It is important to note that while performing these end-to-end tests, either from one FPGA to another or from an FPGA to Mellanox board, no lost packets were detected.

Protocol	Data Transfer	Bandwidth	Latency
RoCE-V2	598Bytes	6.1 Gbps	1 Mx → 10^{-8} Wb = 10^{-8} V·s
Proposed protocol	598Bytes	10.3 Gbps	1 Mx → 10–8 Wb = 10–8 V·s

Table 1: RoCE-V2 vs the proposed prototype measured bandwidth

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

3.2.8 Conclusions and Further developments

RASHPA will be finalized and used for the first time in the final version of smartpix by mid-2019. This will be based on the PCIe over cable network tested and validated within the EUCALL project.

Future development will focus on the integration of both the proposed RDMA over Ethernet protocol and RoCE all together in the RASHPA framework. Selection between these protocols will be based on the price/throughput requirements for each experiment

3.3 Intelligent Ethernet to PCIe bridging (PSI)

As described in EUCALL Deliverable D5.1: 'Report on online 2D image processing', Eiger and Jungfrau are module based detectors. Therefore the readout characteristics (number of frames per second) are independent of the detector size and the total data rate scales linearly with the size.

Each of these detector modules send out UDP packets independently and in parallel containing a section of the image. Since the packet size remains constant the image section sent in one packet depends on the bits per pixel.

On the packet receiving host several receiver threads are running. Each thread is listening to one detector module. Due to the flexible architecture the receivers can also be spread over several hosts to distribute the load.

To differentiate between the modules of the detector the UDP header contains information about the module id, the frame id and packet id. With this information each receiver thread inserts the packet data at the right place in the host memory to build up a complete sub-image.

Eiger 4 Bit/Pixel	Eiger 8 Bit/Pixel	Eiger 16 Bit/Pixel	Eiger 32 Bit/Pixel
16 lines/packet	8 lines/packet	4 lines/packet	2 lines/packet

Table 2: Content of a UDP packet for Eiger

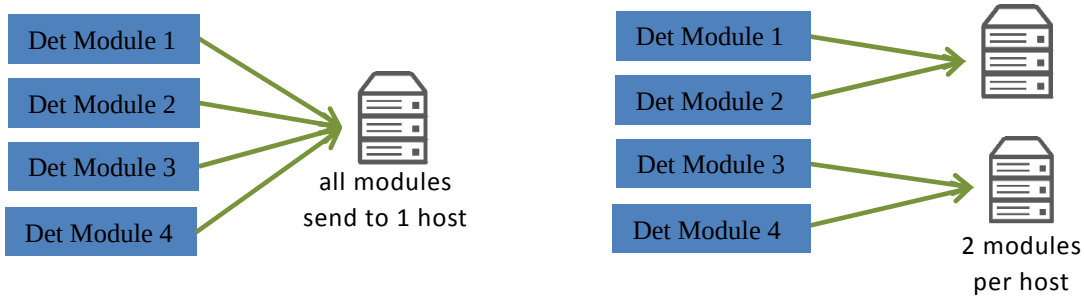

Figure 29: All detectors can either send to one or to multiple hosts

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

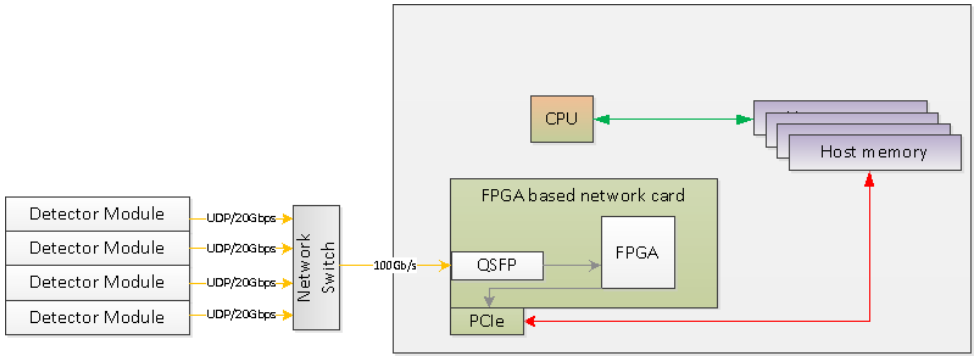

Figure 30: Data flow with the FPGA based network card

As said before every UDP packet contains only a part of the image. Each part consists of several lines of the image. Therefore the receiver thread has to generate several memory addresses. Analyzing the UDP header, generating memory addresses and copying data around for several threads increases significantly the CPU load. For an Eiger 0.5M (a one module detector) running at full speed a four core CPU is fully occupied. In addition to that also missing packets have to be detected and handled. The current software solution allows discarding the whole image, to store the received data or to pad the missing data. Besides storing only the received data, all other solutions mean additional CPU load for data handling.

This is where the intelligent Ethernet to PCIe bridging comes into play. Instead of occupying the CPU with these tasks, an FPGA can easily handle several packets in parallel, generate memory addresses and copy the data. Being master on the PCIe bus the FPGA can write directly into host memory with full PCIe speed. Padding missing data by just leaving this memory region free and indicating this in a bitmask as well as discarding images by just overriding them in host memory are tasks which can be handled easily by an FPGA.

After finalizing one or more images the FPGA can issue an interrupt to inform the CPU.

For a first implementation the Xilinx VCU108 board with a 100 Gbit/s Ethernet and 8 lanes of PCIe Gen 3 is used. A design has been implemented which receives network data packets from the 100 Gbit/s Ethernet interface and analyses the payload header to generate the corresponding memory address. Up to now one memory address per header has been implemented.

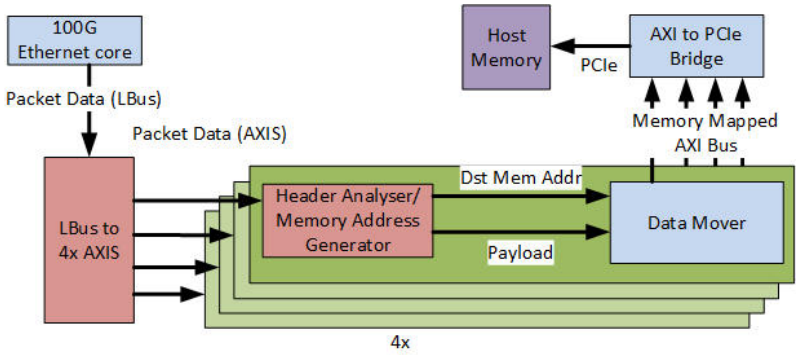

Figure 31: The FPGA design of the FPGA based network card

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

The design makes use of the available Xilinx IP cores. Beside the 100 Gbit/s Ethernet core it uses a transparent AXI-to-PCIe bridge as well as the data mover core to copy the incoming payload to the generated memory address. A custom core parallelizes the incoming data stream into four independent packet processing pipelines including the custom made packet header analysing and memory address generation core.
Unfortunately the maximum transfer into host memory is limited by the PCIe 3.0 x8 with its maximum data rate of 64 Gbit/s.
To test the maximum data rate as well as the CPU load during the PCIe transfer measurements with the Intel Performance Counter Monitor have been done. This tool reads the CPU performance registers containing among other things the memory data transfer rate.
Depending on the packet size the transfer rate is 5.6 GByte/s (4096 Byte packet size) or 6.1 GByte/s (8192 Byte and 16384 Byte packet size).
The CPU load during this memory transfer is negligible and below 1%.
For the future it is foreseen that the FPGA is communicating directly with the GPU for further more complex image processing tasks similar to what is described in Section 4.3 “Data conversion of Jungfrau at PSI”.

3.4 Train Builder (European XFEL)

As described in EUCALL Deliverable D5.1: ‘Report on online 2D image processing’, Eiger and Large and fast 2D image detectors do not only produce high data rates (e.g. for 1 Megapixel detectors at European XFEL, this is about 10 Gbytes/s), but also acquire and transmit data on multiple channels. Therefore each individual channel only provides a certain region of the images. However, in many applications, the data of the individual channels have to be combined (concentrated) on a central system in order to allow processing of complete images. The so called Train Builder is a technical solution, which solves the data concentration, re-arrangement of the fragmented data into correct series of full images, and to pre-process the data before it is forwarded to further processing or storage systems. In that role it also serves as a deterministic high-speed data transfer interface between the detector and the storage and/or computing systems.
The train builder was developed by the Rutherford Appleton Laboratory (UK) and is based on the Advanced Telecommunication Computing Architecture (ATCA). Figure 32 shows an image of a single train builder module, which could be used to handle data of smaller detectors. Figure 33 depicts the complete assembly to handle the data of larger detector systems, which includes four train builder modules and one cross-point switch module. Finally, in Figure 34, a simplified overview of the basic data transfer chain is shown, including a detector, the train builder and a following computing layer.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

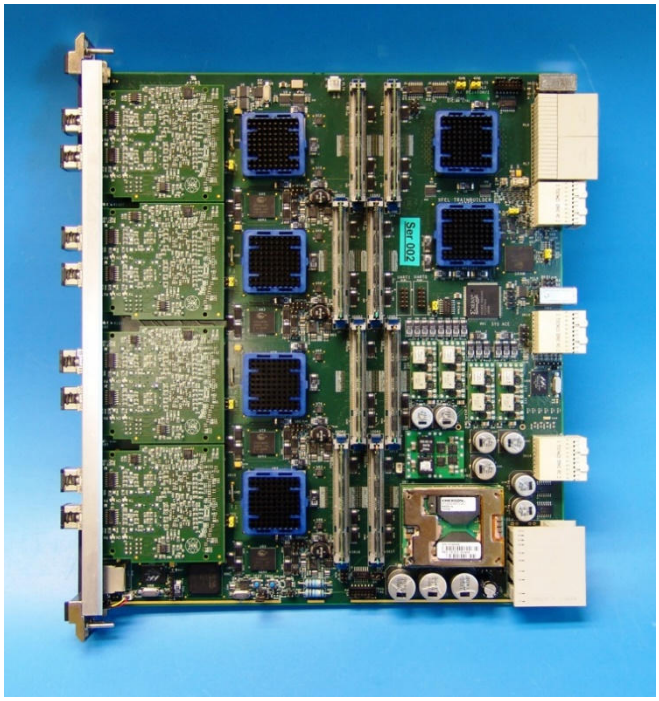

Figure 32: Image of a single train builder ATCA module, which could either be used for smaller detectors (e.g. ¼ Megapixels) or combined with three further modules of the same kind and one cross-point switch interfacing module for larger detectors (e.g. 1 Megapixels). On the left side eight 10 GB Ethernet interfaces provide input and output of the detector data. In the center are five FPGAs and one smaller switch hidden under the blue framed heat sinks attached to memory banks. On the right high-density connectors provide a high-speed interface to further train builder modules and cross-point switch module.

Figure 33: Image of an ATCA crate including train builder and cross-point switch modules to handle data of large 2D detectors. The blue fiber cables connect to the detector and to the further computing systems. The black cables provide high-speed interfaces between the train builder modules. The grey network cables provide configuration, monitoring and control of the train builder system via the control system.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

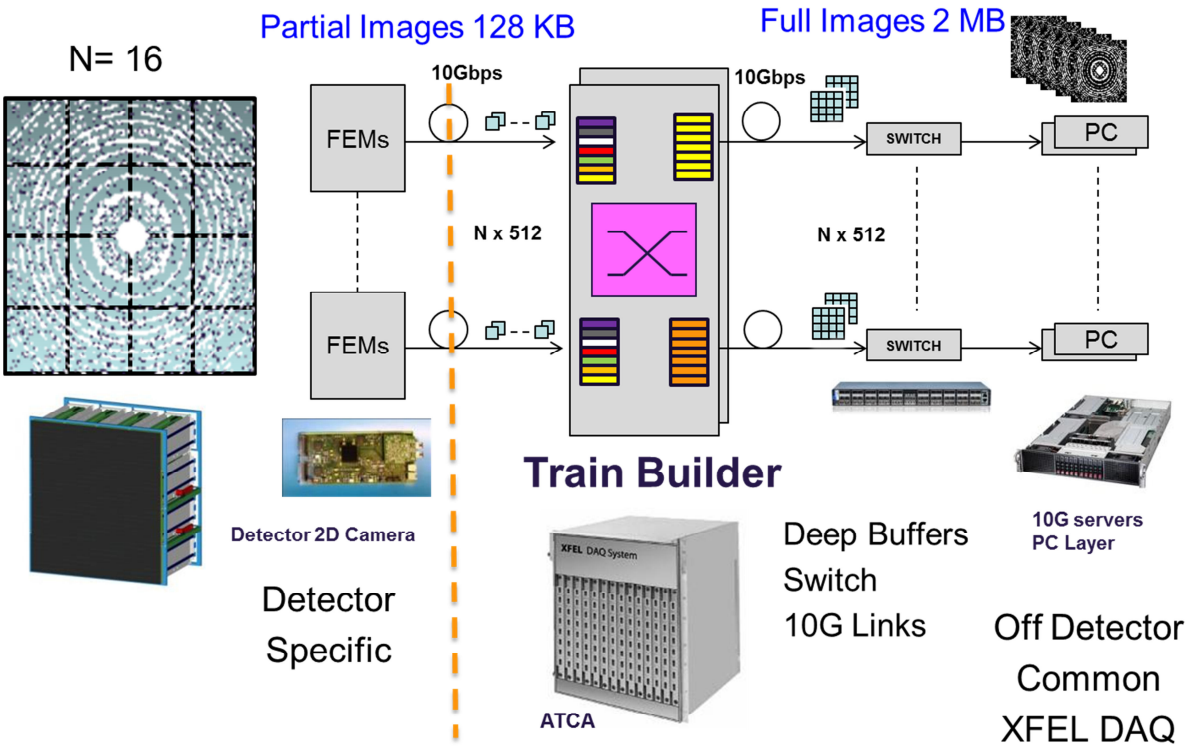

Figure 34: Simplified overview of the environment around the train builder as used in the tests at European XFEL. On the left the data acquired and generated by the 2D x-ray image detectors front-end electronics/modules (FEM) provide the partial and pixel-unordered data streams to the inputs of the train builder. Within the train builder the data transfers and assembly into full image series (trains) is depicted. On the right side the processed data id forwarded by standard Ethernet switches to further computing systems and storage (Data Acquisition System – DAQ).

The train builder system provides up to 16 channels of 10 Gbits/s Ethernet (10GbE) data inputs from a detector and up to 16 channels of 10GbE outputs to following computing layers. The system combines multiple Field Programmable Gate Arrays (FPGAs) and high-speed cross-point switches in order to cope with the high data rates and allows flexible distribution and concentration of data. In order to allow buffering of the data as well as re-organization of the data different Memory blocks are attached to all FPGAs. Due to the large number of interfaces as well as FPGAs, the system consists of multiple ATCA modules, which are connected via multi-channel high-speed interconnects.

The typical usage and data path of the train builder is as following: a 2D detector is producing partial images on 16 channel of 10GbE with an unordered pixel arrangement due to technical reasons (efficient design of the underlying detector and ASICs). These channels are received by the 16 input of the train builder. On arrival the pixels will be ordered. After that all the partial images will be forwarded in a round robin fashion to destination FPGAs and the resulting series of images buffered in memory. At that point further processing of the images is possible before the series of images is forward (each series on one 10GbE output) to the next computing layer for storage to disc or further processing (e.g. GPU, HPC, FPGA, etc.).

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

An important aspect to allow correct data transmission and decoding is the definition of a generic set of protocols. Since the communication interface is 10 GB Ethernet, the UDP was chosen as standard communication protocol. Since UDP is a datagram based protocol and does not provide tracking of order of packets and detection of data losses and additional simple layer of protocol was developed to add this functionality (called PPT). On top of this a generic data format protocol was defined, which provides fields for all kind of information describing the properties of the image data, the images (pixel data), information about the data source as well as further information of the train builder system.
As mentioned above, the key functionalities related to data processing and transfer are:

- Immediate reception of data stream from detector and local buffering
- Re-ordering of pixels within the partial images
- Concentration of the partial images in one place (FPGA/buffer)
- Optional: Providing further possibilities for processing of the images and injecting gathered information
- Transmission of data to next computing layer (incl. distribution to multiple systems)

The complete functionality of the train builder, as described above, had been implemented and the different aspects of the functionality and performance were successfully tested in real environments (e.g. 2D detector and computing layer behind the train builder at European XFEL laboratories).

3.5 CRACEN - Resilient Communication for High Throughput Applications in Heterogeneous Networks (HZDR)

Large scale data acquisition strongly benefits from efficient distribution of the data streams to variable network topologies. The abstraction library Cracen was developed for this purpose. It allows for the flexible distribution of data streams and the communication in arbitrary networks. The CRACEN solution aims at providing a resilient and scalable solution for data transfer and injection that allows for optimizing communication based on topologies, supports load balancing and a variety of communication software stacks. Currently, UDP, TCP and MPI are implemented as back ends for communication, which can be used and mixed in one application with a single communication interface.
CRACEN allows for inherent serialization and deserialization of data. Communication within applications is described via graph-based topologies that can be mapped to existing interconnect topologies to optimize throughput. As such, the communication logic inherent to a certain software solution can be optimized for a chosen underlying hardware interconnect solution and is scalable between hardware solutions. The fundamental working

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

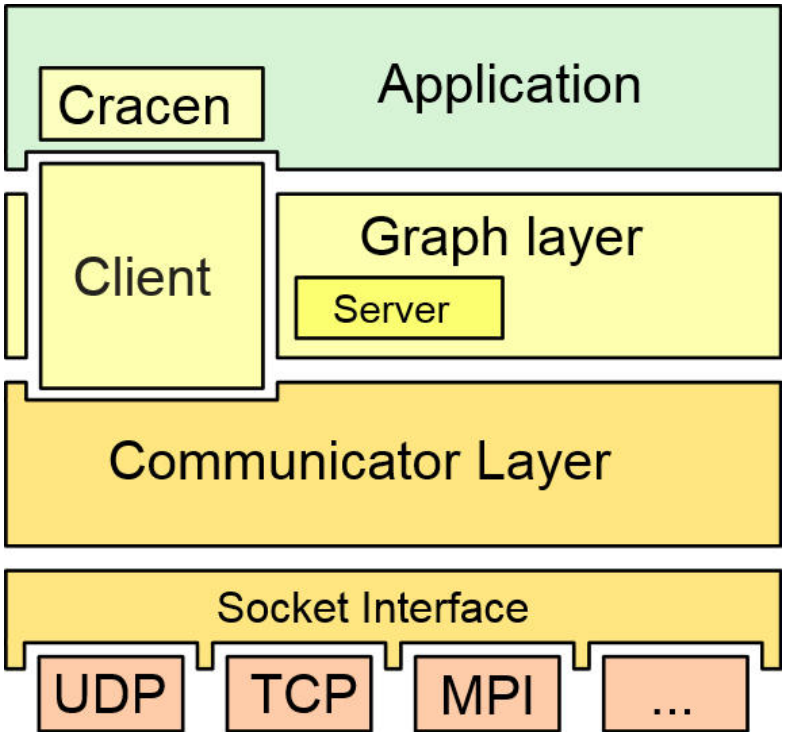

Figure 35: The CRACEN design includes a communication layer on top of a socket interface and a graph layer for describing communication topologies and their mappings to network topologies.

principle of CRACEN is thus as follows: the dataflow is considered as directed graph, the nodes of the graph correspond to transition functions and the edges to communication. Resilience can thus be easily achieved by directing the incoming data another route using available edges and nodes, providing for redundancy as long as the remaining physical hardware allows to support the maximum throughput.

A variety of communication libraries are applied to enable the flexible use of different network topologies. The user has to define a mapping from the logical to the physical nodes. As backends for the communication boost::asio, asio::datagam_socket and boost::mpi were implemented. For the processing of the nodes, input and output ring buffers are used to compensate for peak loads. One or multiple worker threads consume the data and apply the transition function. The framework consumes the output and sends it, using a send policy (e.g. round robin, broadcast or least workload first).

The basis for both load balancing and resilience lies within parallel buffering streaming data between data analysis steps, allowing to describe any application by a graph-interconnected succession of tasks. Oversubscription and buffering allows for scalability, load balancing and resilience at the same time.

Scalability in terms of throughput was measured to be linear on the Hypnos cluster at HZDR. These benchmark tests proved the linear increase of bandwidth from source to sink node with the number of cores. This of course depends on the underlying hardware configuration and selected communication topology, but the data shows that the CRACEN software stack does not prevent linear strong scaling.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

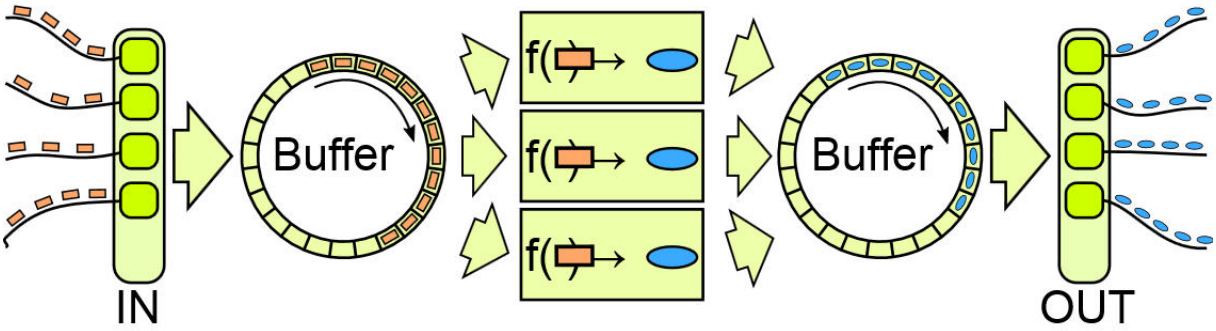

Figure 36: Applications using CRACEN can describe data transfer and analysis by a graph-interconnected sequence of tasks with inherent buffering, allowing for resilience and scalability without a need to include explicit communication calls in the application.

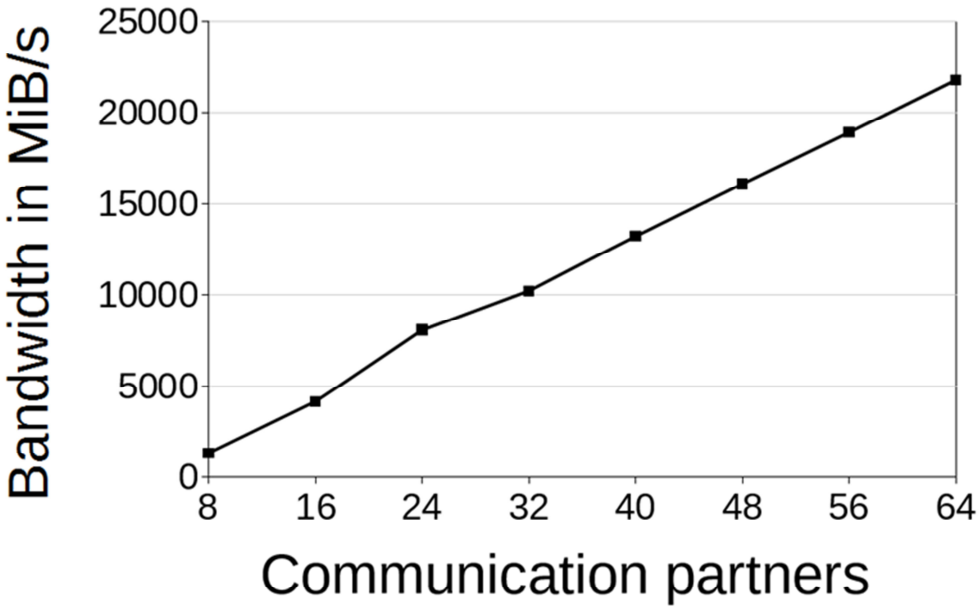

Figure 37: Strong scaling of throughput shows that it scales linearly with the number of communication partners

Resilience and load balancing are achieved by asynchronous communication and penalty-based scheduling of communication operations depending on local buffer size and response time for each communication partner within the topology. With close to empty buffer state and quick response time communication partners receive more data than partners with close to full buffer states and long response times. This means that malfunctioning partners will not be provided with data. Resilience was tested with respect to throughput and demonstrated by removing inner nodes from the graphs which had no significant influence on the throughput in case the total number of cores remained sufficiently high.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

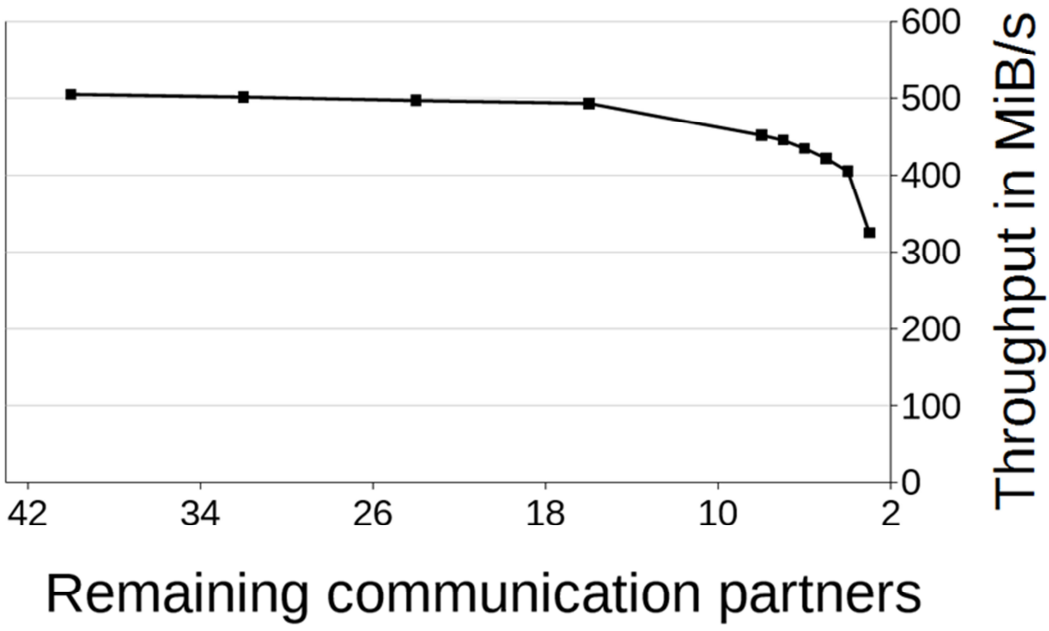

Figure 38: Resilience test for CRACEN: By intentionally decreasing the number of working communication partners throughput gradually decreases, as more partners experience failures.

CRACEN is implemented in C++11 and is available as Open Source [10]. It will be used in a first test be to scale the single-node based solution for the PSI Jungfrau calibration algorithm (see Deliverable D5.1).

4. Synergy Aspects

The work done at ELI-ALPS in collaboration with HZDR is a comprehensive write up of measurements of data throughput and latency for CUDA-based NVIDIA GPUs and their respective host systems as well as of the Infiniband interconnect. Such configurations are a prototypical and very general solution to high data rate imaging solutions. As such the results given here are of extreme importance also for high data rate applications at light sources.

RASHPA is a versatile PCIe based solution that can be adapted by other RIs. As it is already prototyped using a smartpix detector used also at other RIs, portability between RIs is ensured. It is scalable and its hardware stack is based on the widely available PCIe interface.

The PSI application of Jungfrau detector calibration developed together with HZDR is a good example of the interplay between data transfer and analysis that lies at the heart of UFDAC, viewing ultrafast data acquisition as an integrated task of transfer and analysis of data. The solution developed here is currently being evaluated by ELI-ALPS for possible reuse.

The HZDR CRACEN solution aims at a scalable, resilient data transfer solution that is flexible in both communication topology and communication stack. It is GPL Open Source and can be adapted to work for different hardware solutions, with emphasis on DAQ chains connecting

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

a high data rate experiment to a compute centre for online analysis. It is foreseen to use CRACEN for scaling the PSI detector calibration presented here to 32 Jungfrau modules using CRACEN and to test it at ELI-ALPS.
The various applications presented here also shows the different aspects of data transfer and injection with GPU and FPGA based solutions, giving a comprehensive overview of the various setups envisioned for such solutions at the various RIs.

5. Summary and Outlook

We present four prototype setups for determining efficient data transfer and injection, one using FPGAs (ESRF), two on GPUs (ELI-ALPS, HZDR) and one that uses a frontend FPGA with subsequent GPU-based analysis (PSI).
The data throughput typically seen are close to what can be expected from the underlying hardware solution and are in the range of few GBps with freely available hardware.
Latency can become an issue and has to be mitigated by more complex approaches that also increase scalability.
First studies of scalability suggest that scaling in terms of throughput produces new challenges and subsequent analysis is needed to better optimize scalability. Scaling in terms of portability is possible, but depends on the final throughput achievable with the software stack used and the underlying hardware.
Finally, resilience can be achieved for some communication software stacks, but especially those like MPI over Infiniband optimized for throughput and latency pose inherent problems for resilient solutions.
With INTEL Omnipath and IBM/NVIDIA Nvlink under development it is mandatory to stay open for future solutions, which is a great challenge for long-term development at current RIs.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

Publications and References

[1] P. Pangaud et al., XPAD3-S: a fast hybrid pixel readout chip for x-ray synchrotron facilities, Nucl. Instrum. Meth. A 591 (2008) 159.

[2] P. Kraft et al., Performance of single-photon-counting PILATUS detector modules, J. Synchrotron Radiat. 16 (2009) 368.

[3] J. Marchal et al., EXCALIBUR: a small-pixel photon-counting area detector for coherent x-ray diffraction — front-end design, fabrication and characterisation, J. Phys. Conf. Ser. 425 (2013) 062003.

[4] J. Jakubek et al., Large area pixel detector WIDEPIX with full area sensitivity composed of 100 Timepix assemblies with edgeless sensors, 2014 JINST 9 C04018.

[5] R. Ballabriga et al., The Medipix3RX: a high resolution, zero dead-time pixel detector readout chip allowing spectroscopic imaging, 2013 JINST 8 C02016.

[6] T. Tick and M. Campbell, TSV processing of Medipix3 wafers by CEA-LETI: a progress report, 2011 JINST 6 C11018.

[7] X. Wu, J. Kalliopuska, S. Eranen and T. Virolainen, Recent advances in processing and characterization of edgeless detectors, 2012 JINST 7 C02001.

[8] X. Llopart, R. Ballabriga, M. Campbell, L. Tlustos and W. Wong, Timepix, a 65k programmable pixel readout chip for arrival time, energy and/or photon counting measurements, Nucl. Instrum. Meth. A 581 (2007) 485 [Erratum ibid. A 585 (2008) 106].

[9] C. Ponchut, M. Ruat and J. Kalliopuska, x-ray imaging characterization of active edge silicon pixel sensors, 2014 JINST 9 C05017.

HZDR. (2017, 9 7). Retrieved from High Performance Computing at HZDR: https://www.hzdr.de/db/Cms?pOid=12231&pNid=852

libibverbs. (n.d.). Retrieved 9 7, 2017, from www.rdmamojo.com/2012/05/18/libibverbs

NIIF - 'KIFÜ/NIIF' LEO HPC., https://niif.hu/en/supercomputing

NVIDIA Accelerated Computing. (2017, 9 7). Retrieved from Benchmarking GPUDirect RDMA on Modern Server Platforms: (https://devblogs.nvidia.com/parallelforall/benchmarking-gpudirect-rdma-on-modern-server-platforms/)

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220

Open MPI. (n.d.). Retrieved 9 7, 2017, from https://www.open-mpi.org/

OSU Micro-Benchmarks. (2017, 9 6). Retrieved from http://mvapich.cse.ohio-state.edu/static/media/mvapich/README-OMB.txt

perftest. (2017, 9 7). Retrieved from https://github.com/lsgunth/perftest/blob/master/README

RDMA. (n.d.). Retrieved 9 7, 2017, from https://en.wikipedia.org/wiki/Remote_direct_memory_access

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 654220