A high-performance data acquisition on COTS hardware for astronomical instrumentation

Julien Plante, Damien Gratadour, Lionel Matias, Cédric Viou, Elena Agostini

A high performance data acquisition on COTS hardware for astronomical instrumentation

Julien Plante[a], Damien Gratadour[a], Lionel Matias[b], Cédric Viou[c], and Elena Agostini[d]

[a]LESIA, Observatoire de Paris, Université PSL, CNRS, Sorbonne Université, Université de Paris, Meudon, Hauts-de-Seine, France
[b]Thales LAS France, Limours, Essonne, France
[c]Observatoire Radioastronomique de Nançay, Observatoire de Paris, PSL Research University, CNRS, Université d'Orléans, Nançay, Cher, France
[d]NVIDIA, Santa Clara, CA, USA

ABSTRACT

Data throughput in modern telescopes instrumentation have been steadily increasing over the last decade. The few gigabits per second range is now the lower bound, and bandwidths as high as tens of terabits per second are expected with the Square Kilometer Array. We present a new approach based on DPDK, and its support for GPUDirect recently introduced by Nvidia to perform DMA from Network Interface Controller (NIC) to GPU memory, to answer very high throughput data acquisition in astronomy.

Keywords: Data acquisition, GPU, DPDK, GPUDirect

INTRODUCTION

Telescopes, such as other cyber-physical systems, share a similar high-level architecture: information from the physical world is captured through sensors (e.g. cameras or antennas), sent to a processing unit, and an action is taken (e.g. storing an image or a spectrum, steering an antenna or activating an actuator).

Much progress has been made on the processing side, with systems increasingly hybrid, combining multicore CPUs, GPUs, and others to reach the required computing power. The advent of the CUDA programming language especially made for GPUs has become a first-class general purpose computing hardware, bringing together very high throughput and user-friendly programming and low cost maintainability.

Although the processing side has seen many actionable improvements, we cannot say so for the data acquisition side. Even though some major progress on the hardware side has to be acknowledged (10/40/100/200/... GbE, Infiniband), their acceptance and use is still limited. Some high level programming models, such as MPI, are already able to use these features at their full power, but little work has been made to fit the needs of general purpose data acquisition.

In this paper, we present a high performance data acquisition prototype and provide performance estimates. This prototype is based on Commercial-off-the-Shelf (COTS) hardware, targeted at astronomical instrumentation together with GPU computing. We will begin in section 1 with a more in-depth explanation of the issue of data acquisition, covering the mechanism and the current approaches. Then, we present the different applications that motivated this work in section 2, and that will be used as examples and test cases during the rest of the article. We describe the current prototype in section 3, and show its performance in section 4. Finally, we state further possible improvements to the current prototype in section 5, and conclude.

Further author information:
Julien Plante: E-mail: julien.plante@obspm.fr

1. DATA ACQUISITION IN CYBER-PHYSICAL SYSTEMS

1.1 Ubiquity of the UDP protocol

To transfer data between different parts of a system which processes a continuous stream of data, a commonly used approach is to rely on the UDP (User Datagram Protocol) network protocol, together with some application specific information. The UDP protocol is very lightweight, containing only a checksum as security mechanism. It does not guarantee delivery nor ordering, resulting in less overhead and simpler emitter/receiver, but requiring some security mechanisms to be implemented on top.

This simple design explains its ubiquity in cyber-physical and real-time systems, together with some historical factors. Let's now see how a data acquisition system can be built for this kind of protocol when using GPUs.

1.2 Naive approach and limitations

The naive approach to perform computations on streaming data on a platform hosting a number of CPUs and GPUs can be described as the following:

- Receive data through the Linux network stack (recv / recvmsg / recvmmsg)
- Process the application specific header on CPU, and store the payload in a ring buffer in RAM
- Copy a chunk of the ring buffer to GPU memory
- Process this chunk of memory on the GPU
- Take action

The issue with this design is two-fold:

- Each call to a Linux networking function requires a system call, which has a moderate to high impact on performance. The worst case is when this system call requires a context switch. A better case is when vDSO is used, but overall, it is hard to receive more than 10 Gbit/s per CPU core because of this.
- Data has to be copied twice across the PCIe (NIC $\rightarrow$ RAM, RAM $\rightarrow$ GPU), which induces additional unwanted latency.

1.3 Related work

RDMA is possible over an Infiniband network fabric, or over an Ethernet fabric using RoCE (RDMA over Converged Ethernet). These solutions are arguably the most widespread, and provide good overall performance. However, this is mostly used for node-to-node communication in HPC, as it requires a level of logic on the emitter side. In cyber-physical systems, data often comes from sensors, or very simple, high-throughput hardware. The level of control over this emitting side can be very low, and implementing RoCE was not conceivable for the applications motivating this work.

Sending packets directly from a network controller to the GPU memory has been achieved for Adaptive Optics (AO) using a custom FPGA board,[1] achieving line throughput and very low latency at the price of dedicated hardware and firmware, rather difficult to maintain. The motivation behind our work is to reiterate this result on Commercial-off-the-shelf (COTS) hardware, to reduce the development time, help maintenance and ease the development of future high-performance data acquisition systems.

Many projects have been initiated in order to overcome the performance bottleneck of the Linux networking stack, the most adopted solutions being the Data Plane Development Kit (DPDK)[2] and Programming Protocol-independent Packet Processors (P4).[3] Some projects in astronomy already use such frameworks as a data acquisition method, such as the CHIME detector[4] and YAO's 40-m telescope.[5]

However, this has only ever been used to replace calls to the Linux networking stack, still using a RAM-based ring buffer. NVIDIA recently introduced a new DPDK library named gpudev[6] to help GPU real-time packet

processing applications where CPU is in the critical path to coordinate the network card to receive packets in GPU memory (technology branded as "GPUDirect RDMA" by NVIDIA) and notifying a packet-processing CUDA kernel waiting on the GPU for a new set of packets. The main requirement is to maximise the zero-packet loss throughput at the lowest latency possible. The gpudev library, not only introduces GPUDirect RDMA in the DPDK ecosystem but also offers utilities like the so called "communication list" to provide a real-time direct information exchange tool to CPU and GPU. We wanted to apply this technique to astronomy in order to remove all the costly operations mentioned in subsection 1.2.

2. TARGETED APPLICATIONS

2.1 Adaptive Optics

Our work has been chosen as a solution for the acquisition of wavefront sensor (WFS) image frames in the context of MICADO's SCAO (Single-Conjugate Adaptive Optics),[7] one of the adaptive optics systems of the ELT. This application is the main driver for very low latency, as the round-trip time through the controller in this context must be below 300µs. Ingress bandwidth is of about 2 Gbit/s, and as such is not the main challenge.

This application is also the only one explored with burst emission: in SCAO, a WFS image frame is broken down into 1 leader packet, N payload packets and 1 trailer packet. The case addressed by this work relates to the ESO-ALICE WFS camera, where N=60. This burst emission mode can be compared to a continuous emission type, where every packet is a payload packet, without discretization of separate image frames.

Reception has already been validated for this use on a first prototype, and is being fully qualified. The processing part of the pipeline has already been implemented as modules in the COSMIC framework.[8]

2.2 Radioastronomy

The original exploration ground for this work is a FRB pre-detection pipeline for NenuFAR.[9] The goal of this project is to perform a continuous Fast Radio Burst (FRB) survey on NenuFAR, while only triggering storage of the full-resolution time-frequency data through candidate FRB pre-detection. This would make it possible to detect a larger number of new FRBs, without a huge memory footprint.

A collaboration has also started with CSIRO to use this work for the ongoing upgrade of ATCA-BIGCAT. This application is especially challenging because of the higher bandwidth need, from 60 Gbit/s to 120 Gbit/s depending on the final design choices. The multiple streams configuration of this setting is also interesting, as we rely on hardware packet filtering to distribute the whole data stream to multiple compute nodes.

2.3 Radar

The final application of this work currently in development relates to radar systems, through a collaboration with Thales LAS France. Radar systems at Thales share the same high-level architecture than radio-telescopes on the reception side, and can take advantage of a high performance data acquisition system. Bandwidth requirements range from 40 Gbit/s to 1.6 Tbit/s, with an overall timing budget of a few hundreds of microseconds per algorithm iteration, implying very low latency in this application as well.

A functioning prototype has been developed for this project, with which we have demonstrated the reception up to 100 Gbit/s. This maximum bandwidth is currently limited by the hardware used in this setup.

2.4 Reusability

This work is being implemented as a library, and offers an optional wrapper with the COSMIC framework.[8] Both solutions are being developed to help building high performance data acquisition for GPU-based computing nodes. While the library itself is not ready to be distributed yet, a first glance at the code for the NenuFAR application can be made in this repository: `https://gitlab.obspm.fr/jplante/NenuFRB`. The Acquisition folder implements the work exposed in this article.

The main difficulty in building this work into a reusable component is the wide variety of protocols. One idea would be to build upon P4[3] for the protocol description syntax, and to generate CUDA + DPDK code from there.

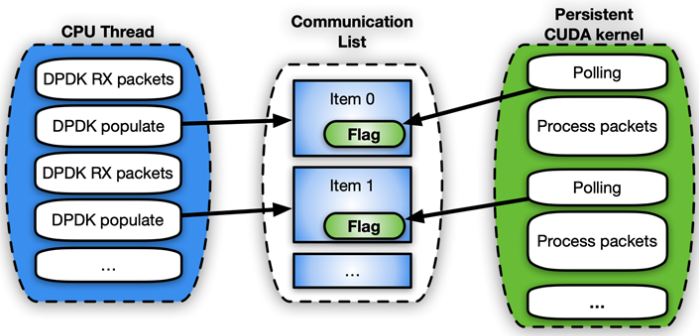

Figure 1. DPDK gpudev communication list mechanism to coordinate in real-time CPU receiving packet and notifying the GPU and GPU processing them. In this way, network and processing can be parallelized.

3. CURRENT PROTOTYPE

3.1 Design and limitations

As mentioned in subsection 1.3, by means of the DPDK gpudev library in order to reproduce the sequence in subsection 1.2 it's possible to:

- Initialise network card to use GPU memory for network packets creating a GPU memory DPDK mempool
- Pre-launch a CUDA kernel, waiting on GPU to receive a new set of packets to process (persistent or semi-persistent CUDA kernel)
- Create a communication list to coordinate CPU and GPU progresses
- Receive packets in GPU memory using DPDK `rte_eth_rx_burst` function

Once these items are in place the interaction between CPU and GPU can be described as in Figure 1:

- CPU thread keeps receiving packets in GPU memory and populate communication list items with packets info (number, memory addresses, etc..)
- CUDA packet processing kernel waits on the next item of the communication list to get CPU notification about new packets received in GPU memory
- Process the new subset of packets received in GPU memory as in Figure 2

Pros of this approach are many. Pre-launching a CUDA kernel waiting on the communication list items for a CPU notification avoids to launch a new CUDA kernel for each subset of receiving packets (less CPU work in critical path). Additionally, the whole Ethernet packet is received in GPU memory so protocol termination can be done directly within the CUDA kernel where packets' headers and payload can be analysed and real-time decisions can be made (as in Figure 2).

Moreover, packet processing is a GPU software approach so whatever type of analysis an application has to do (data placement, traffic analysis, etc..), it can be implemented regardless the subsystem (type of CPU, type of network card, etc..) without relying on specific network card offloads or CPU capabilities taking the advantage of the high degree of parallelism of GPUs being able to process tons of packets in parallel.

A first limitation of this design is that the CPU is still on the critical path. One core is required to poll the NIC for new ingress packets. Although this core is doing very little work, it is looping infinitely in order to reduce jitter as much as possible, resulting in a 100% CPU load. Similarily, this solution requires by design a subset of a GPU to perform packet processing. This contributes to the great flexibility and performance of this approach, but has to be taken into account as it could interfere the overall GPU performance.

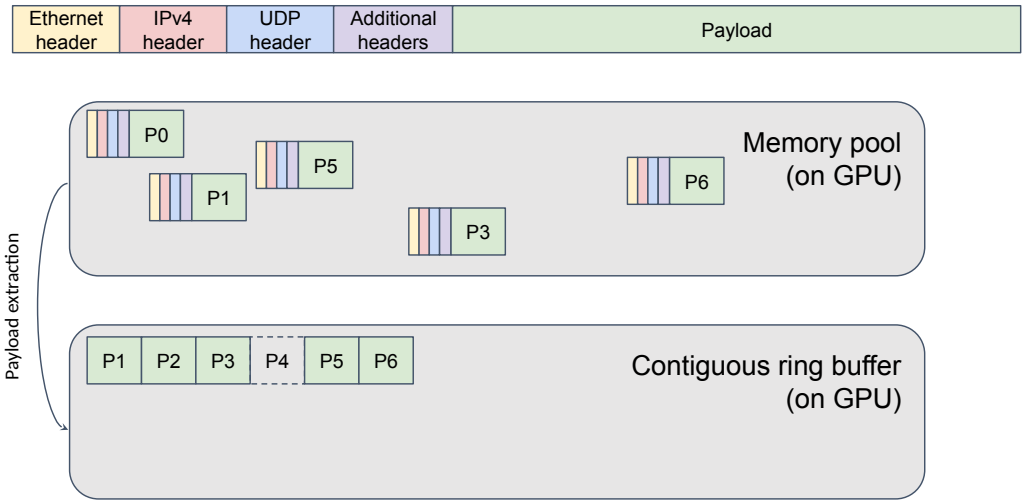

Figure 2. General representation of packet reconstruction. Packets arrive in a GPU memory pool managed by DPDK, which does not guarantee any ordering. We thus have to parse each incoming packet to extract the payload to a contiguous memory location.

A second limitation is that this technology has been introduced by Nvidia, and only been tested on Nvidia hardware (Mellanox NICs and Nvidia GPUs). While the concept of DMA is a standard PCIe feature, and should be usable on other hardware, this has not been verified yet.

Finally, a third limitation is that the user needs to write a GPU kernel in CUDA to process the application specific header on GPU. Each new protocol needs a part of rather low-level code to be written, which reduces portability. However, writing this packet processing kernel allows the user to have complete control about this packet processing operation, which enables the implementation of many features: complex networking protocols, preprocessing (time integration, power law, ...), signal transposition and so on.

4. BENCHMARKING

Two benchmarking machines have been used to verify the functional behaviour and measure the performance in terms of latency and throughput of our data acquisition method. The only major difference between the two machines is the version of PCIe bus. We observed that the PCIe gen 3 limits the maximum bandwidth at 80 Gbit/s, contrary to the 4th generation with which we reached 100 Gbit/s.

To facilitate latency measurement, experiments have been made with a physical loopback Ethernet link connecting the NICs of the machine handled by different CPU cores. This avoids clock synchronization issues, which would be predominant at our time scale ($< 100\mu s$). Experiments are planned to check the functional behaviour on two different machines, but will not be addressed in this article.

During our experiments on "moksha" we learnt about the importance of the PCIe topology when using this data acquisition technology. Going through multiple PCIe switches can decimate the achievable throughput, and much care must be taken to the PCIe topology because of this.

4.1 Packet processing kernel bandwidth

A preliminary experiment was to check whether the packet processing GPU kernel (see Figure 2) was able to keep up with the ingress throughput. We measured the throughput of this kernel, executed on one CUDA block and a varying number of CUDA threads per block, as well as a varying size of memory transactions (Figure 5).

Figure5 helps sizing the required GPU resources to run the persistent packet processing kernel. We observe a sweet spot for 512 threads per block, which can be explained by the high cost of memory transactions (hundreds of clock cycles per transaction). Using more threads allows the hardware to better hide these latencies, at the price of higher contention on the memory controller. Up to 512 threads per block, the latency hiding mechanism

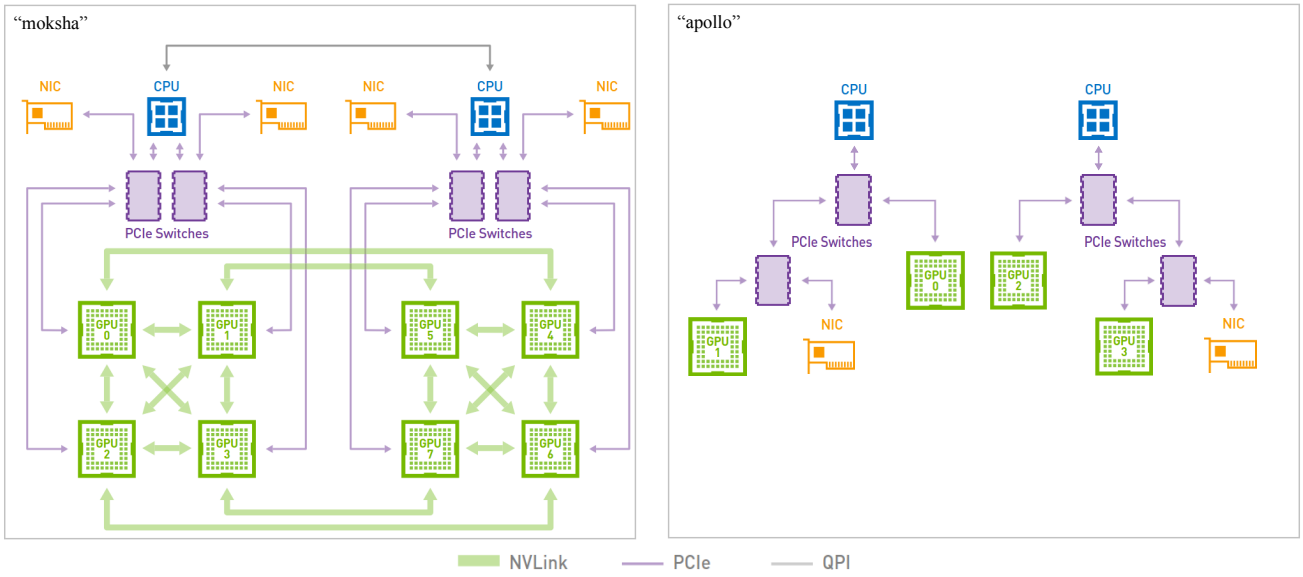

Figure 3. The two test machines, "moksha" and "apollo". Moksha features a PCIe Gen 3 bus, while apollo features a PCIe Gen 4 bus. On moksha, the leftmost NIC is connected to the rightmost NIC through a 100GbE Ethernet. On apollo, the two NICs are interconnected using a 100GbE Ethernet cable, that can be upgraded to a 200GbE Ethernet cable.

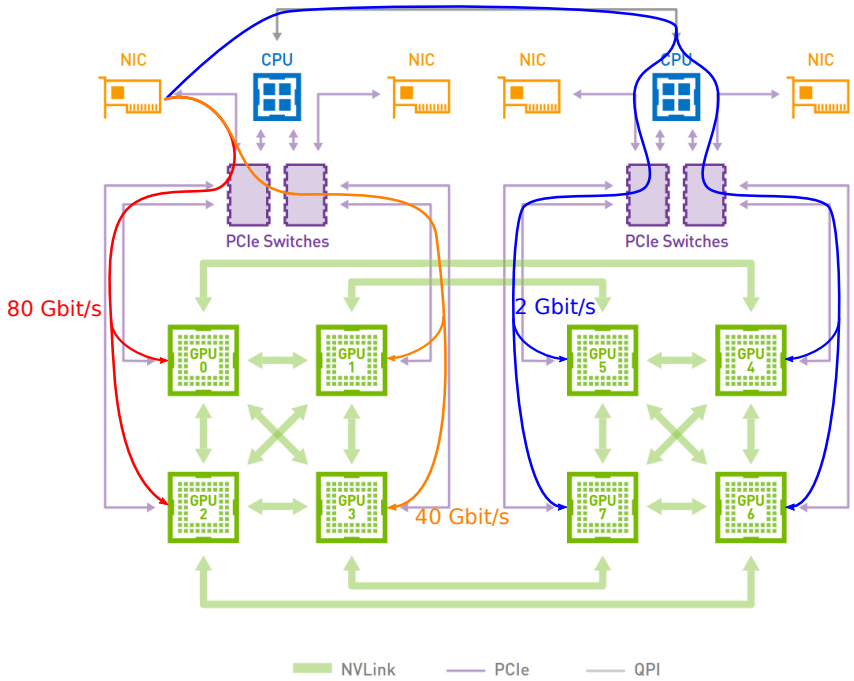

Figure 4. Mean bandwidth on PCIe generation 3 for DMA between a NIC and a GPU varying through different paths. The best bandwidth is achieved for devices that are on the same PCIe bridge.

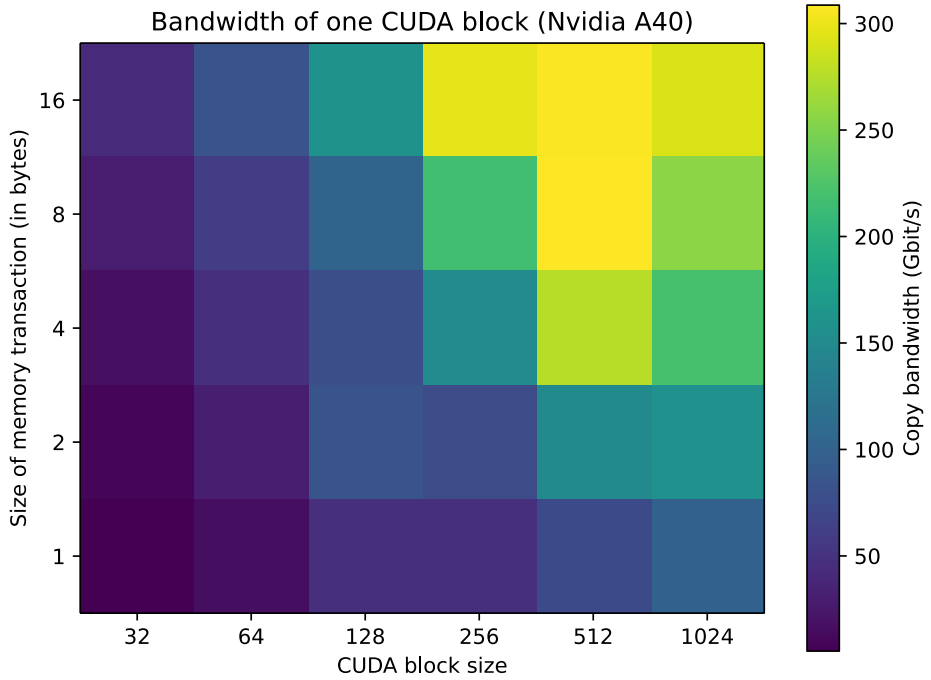

Figure 5. Measured copy bandwidth of one CUDA block on Nvidia A40. Memory transactions can be vectorized on GPU up to 16-byte transactions, as long as the memory accessed is aligned to the same boundary.

prevails, leading to an increased bandwidth. After this limit, the higher hardware contention becomes limiting, resulting in worse performance for 1024 threads per block.

We also confirm that vectorization using bigger memory transactions improve the copy bandwidth. Finally, we notice that most configurations can handle 100 Gbit/s, and many of them can sustain 200 Gbit/s. We deduce that a kernel of only one CUDA block is enough to perform packet processing. Using only one block will help keeping the performance impact of this approach limited.

4.2 Adaptive Optics experiment

Other experiments were focused on simulating the reception of the multiple real-life applications exposed in section 2. The Adaptive Optics application was tested on the "moksha" machine, with the leftmost NIC sending ALICE WFS-like packets, and the rightmost NIC receiving. Processing is done on GPU 4, which is on the same PCIe bridge as the receiving NIC.

The functional behaviour for this application is correct, with image frames being reconstructed as expected and no packet being lost. We measured the latency between the reception of the trailer packet and the end of the packet processing for each frame (Figure 6).

Two implementations are compared. A reference one acquiring packets with DPDK on CPU, and performing the packet processing on CPU before uploading data to the GPU afterwards using a cudaMemcpy. Basic optimizations were used to make a fair comparison (vectorization, pinned memory). The second implementation relies on DPDK with GPUDirect, as described in the other parts of this article. Both implementations are based on the COSMIC framework.

We observe a mean latency five times smaller with our prototype than with a more naive approach. However, we notice a relatively high jitter: even though the standard deviation of this recording is small, the peak-to-peak latency is very high with regular spikes of latency (4x mean latency). This is most likely due to the presence of the CPU on the critical path. While this result is acceptable and shows the feasibility of a COTS acquisition system, it could be improved and especially optimized for jitter, as discussed in 6.1.

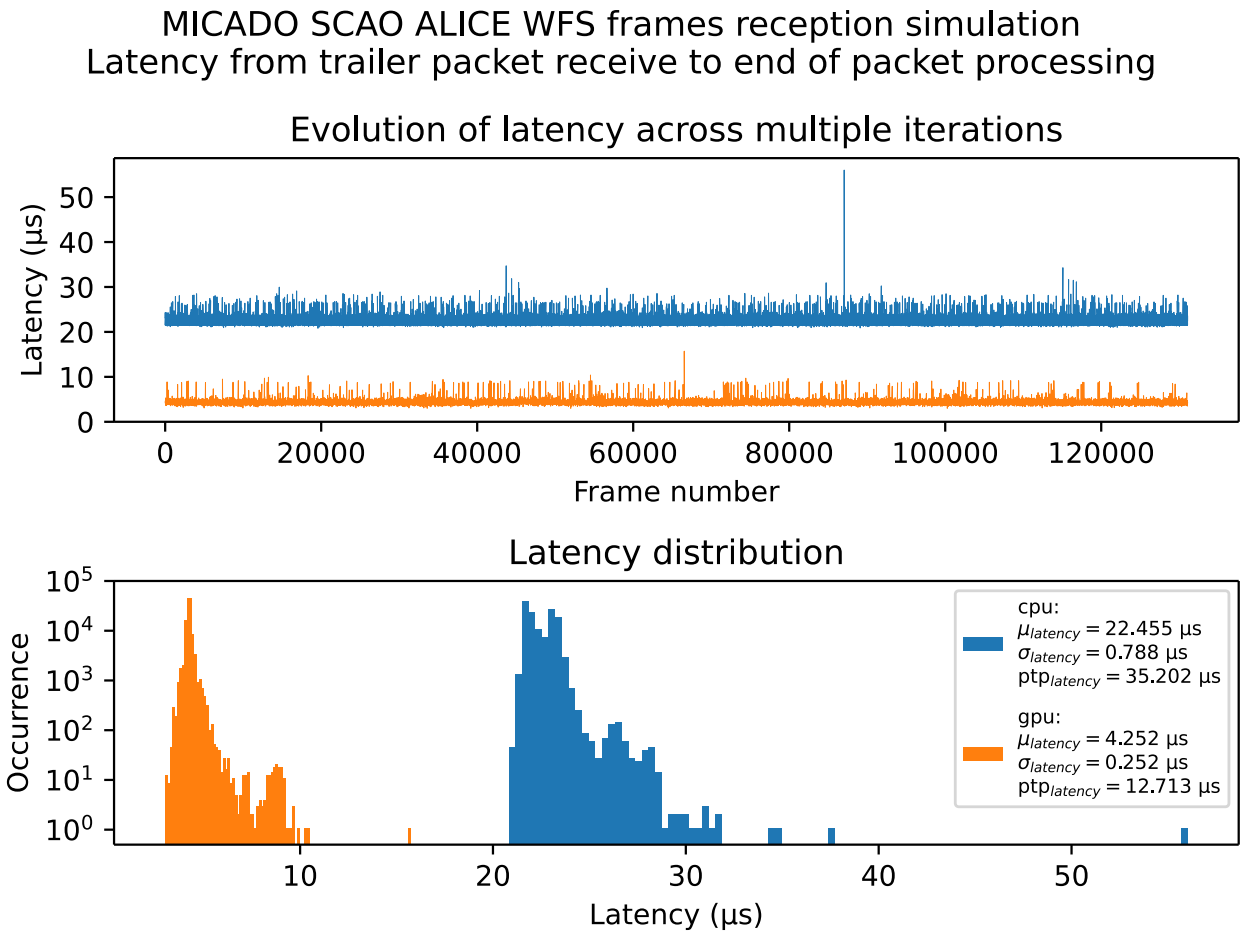

Figure 6. Latency measured for 2^{17} frame transfers

4.3 Other experiments

For other applications, the functional behaviour has been validated, up to 100 Gbit/s for radar, with no packet loss. This demonstration has been made using only one CPU core, and one CUDA persistent kernel of one CUDA block. Latency measurements have not been performed as of now, since the latency requirement is weaker than for adaptive optics.

For the NenuFAR application, the behaviour has been verified using packet captures of the actual network traffic of the telescope, stored under the PCAP format. This is a great step in the writing of the FRB predetection pipeline for NenuFAR,[9] and will allow us to focus on the processing part.

For radars, the behaviour has only been verified on a custom packet generator written with DPDK, because of the lower technlogy readiness level, but show very promising results. Indeed, these first results show that this acquisition method can be used to achieve line throughput on COTS hardware.

5. FUTURE WORK

5.1 GPU-driven DMA

One of the biggest limitation of the current design is that CPU is still on the critical path, since the packet reception is driven by the CPU. Previous work show that CPUs are the main source of jitter in real-time hybrid systems, even when using CPU shielding. Moreover, our design only requires the presence of the CPU as an orchestrator because of DPDK, it does not serve any other purpose currently.

One of the future goals would be to get rid of the CPU in the main loop of reception, and to drive the DMA transfer from the GPU itself. The expected gain of this shift would be to further reduce the latency and jitter observed, as well as freeing the CPU core running the host-side reception loop. Depending on the capabilities of such a GPU-driven DMA, it could also reduce the complexity of the data acquisition system, by reducing the race condition hazards and the number of required synchronizations.

5.2 OS-level optimizations

The “moksha” machine runs Ubuntu 20.04 LTS, together with the Linux kernel 5.4.0-97-generic. Further experiments will focus on CPU shielding and using a real-time kernel.

5.3 Security mechanisms

The current implementation feature little security mechanisms. Ingress data is stored in a ring buffer, without any meta information. This implies that old data is overwritten without any notice, and it is not possible to detect a missing packet.

A solution based on attaching an additional ring buffer containing this meta information to the data ring buffer is being investigated. The nature of this information could be made protocol-independent.

5.4 Targeted applications

We are gaining maturity in this data acquisition method with each of the various application we already have. All of them are only explored through simplistic packet emitters currently. The next step regarding these applications is to perform the data acquisition on a real stream of data, or on a better simulator depending on the cases. We especially plan to deploy the FRB predetection pipeline on NenuFAR by the end of the year, after completing the processing chain.

CONCLUSION

We presented a new high-performance data acquisition solution for hybrid systems based on COTS hardware, using DPDK and GPUDirect. The current prototype achieves low latency (30 µs) up to line bandwidth, and has been tested up to 100 Gbit/s with no packet drop. This work will further simplify the use of GPUs in computing nodes of cyber-physical systems, and will help reaching real-time performance on hybrid systems.

We are working to make this work usable for any protocol, looking at high-level protocol descriptions such as those proposed by P4.[3] As of now, the prototype developed can be adapted manually to any protocol, but requires a good understanding of the DPDK. Further iterations will aim at making this work easy to use as a replacement of more standard acquisition methods.

ACKNOWLEDGMENTS

Many thanks to Florian Ferreira from LESIA for his help on understanding the protocols used in MICADO-SCAO.

The PhD project of Julien Plante at Observatoire de Paris is sponsored through a grant from Paris Region PhD program (PRPHD 2020).

This work is sponsored through a grant from project 873120, a.k.a. Rising STARS, funded by European Commission under program H2020-EU.1.3.3 coordinated in H2020-MSCA-RISE-2019.

REFERENCES

[1] Patauner, C., Biasi, R., Andrighettoni, M., Angerer, G., Pescoller, D., Porta, F., and Gratadour, D., "Fpga based microserver for high performance real-time computing in adaptive optics," *Proceedings of the Adaptive Optics for Extremely Large Telescopes 5* (2017).

[2] Leger, J. S., "Myth-busting dpdk in 2020," *AvidThink* (2020).

[3] Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., and Varghese, G. e. a., "P4," *ACM SIGCOMM Computer Communication Review* **44**(3), 87–95 (2014).

[4] Amiri, M., Bandura, K., Berger, P., Bhardwaj, M., Boyce, M. M., Boyle, P. J., Brar, C., Burhanpurkar, M., Chawla, P., and Chowdhury, J. e. a., "The chime fast radio burst project: System overview," *The Astrophysical Journal* **863**(1), 48 (2018).

[5] Dai, W. and Wang, F., "Study on processing performance of a dpdk and gpu combined pulsar data reduction system," *International Journal of Mechatronics and Applied Mechanics* **1**(6) (2019).

[6] Agostini, E., "Boosting inline packet processing using dpdk and gpudev with gpus | nvidia technical blog," (2022).

[7] Clénet, Y., Buey, T. M., Rousset, G., Cohen, M., Feautrier, P., Gendron, E., Hubert, Z., Chemla, F., Gratadour, D., and Baudoz, P. e. a., "Overview of the micado scao system," *SPIE Proceedings* (2014).

[8] Ferreira, F., Sevin, A., Bernard, J., Guyon, O., Bertrou-Cantou, A., Raffard, J., Vidal, F., Gendron, E., and Gratadour, D., "Hard real-time core software of the ao rtc cosmic platform: architecture and performance," *Adaptive Optics Systems VII* (2020).

[9] Plante, J., Gratadour, D., Bondonneau, L., and Viou, C., "A novel frb detection pipeline for nenufar," (2021).