High-throughput Low-latency Online Image Processing by GPU/FPGA Data Coprocessors using RDMA Techniques

Thesis Canvas

{raphael.ponsard PhD Student, nicolas.janvier}@esrf.fr

{dominique.houzet, vincent Fristot}@gipsa-lab.grenoble-inp.fr

OUTLINE

The European Synchrotron Radiation Facility

Problem statement

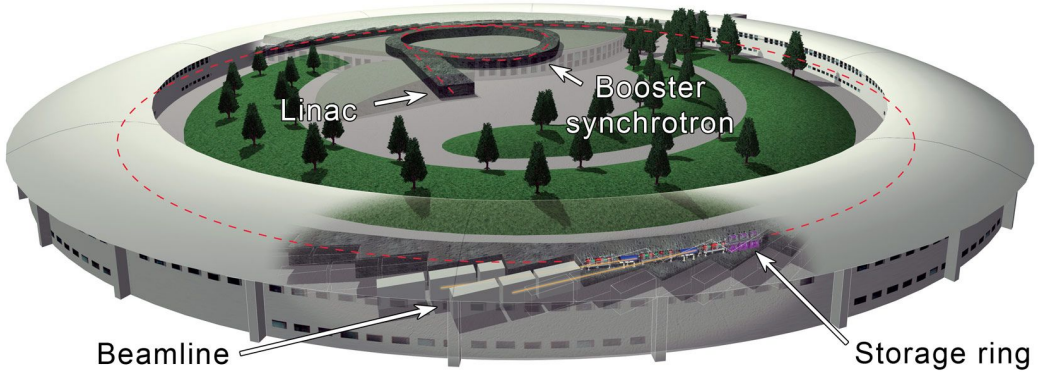
- High-throughput data transfer (RoCE)
- Directly into GPU/FPGA coprocessors for Low latency data analysis

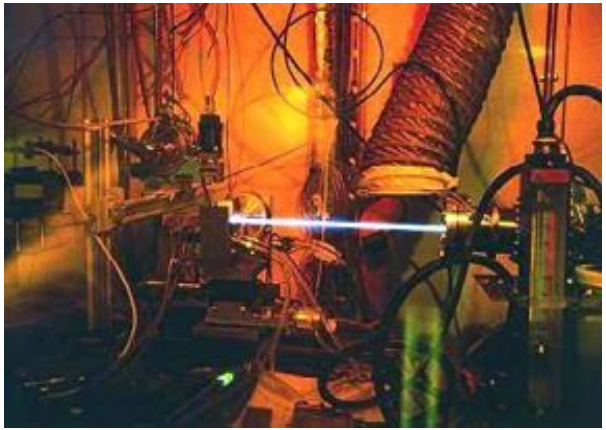
Outcome

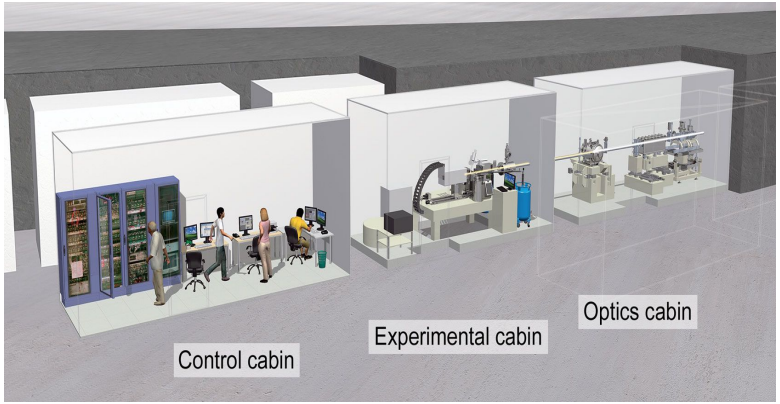
- Detector simulators
- Data analysis pipeline in GPU/FPGA

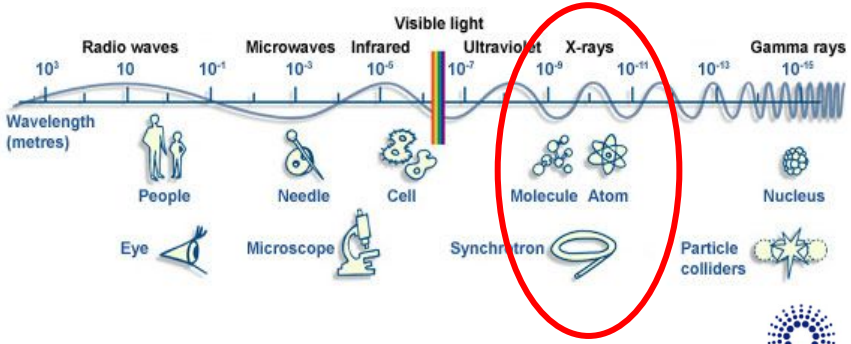
European Photon & Neutron Science Campus
Grenoble, FR

X RAY SYNCHROTRON RADIATION



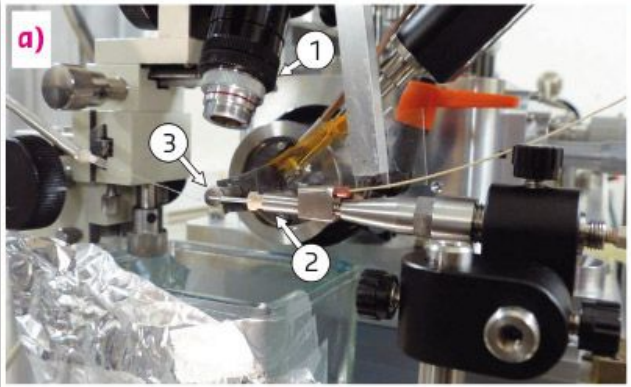


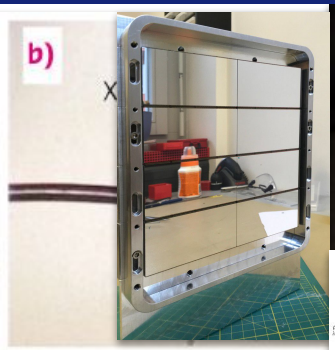


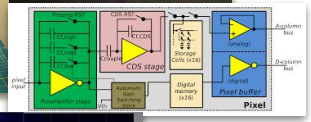


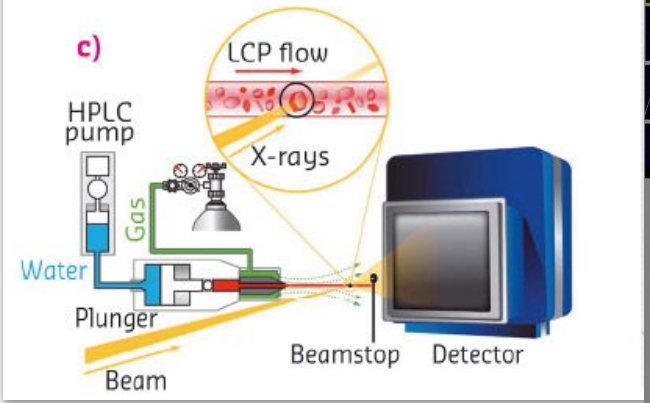
The European Synchrotron | ESRF

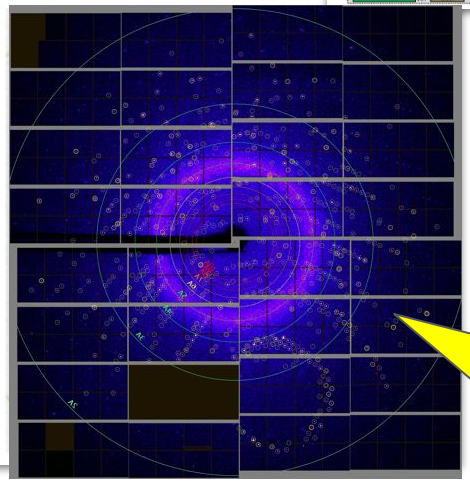
PHOTON SCIENCE AT BIG DATA ERA











Challenging use case: X ray Serial Crystallography is one of the most demanding science.

Precludes storage of the raw data and batch processing as done traditionally.
Online processing challenges:

- Complex raw data pre-processing…
- Fast-feedback,
- Image rejection,
- Compression,
- Reconstruction, ...

PSI Jungfrau 32M detector =
32 modules x 1024x512 pixels
16 bits 2.2 kHz **68 GB/s 4T/mn**

ESRF RASHPA DAQ FRAMEWORK

High-throughput, low-latency, detector driven, data transfer

Spatially and temporally selectable RoI

Multiple concurrent data flows

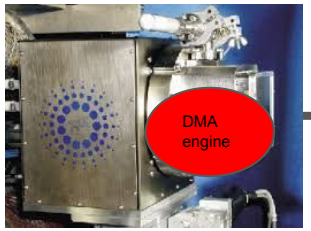
RDMA / Zerocopy techniques

FPGA/GPU Accelerators as data processing units

Scalable and technology agnostic: RoCEv2 (GBE), PCI-e

RASHPA canonical case (RoCE) using Memory Mapped to Stream (MM2S) and Stream to Memory Mapped (S2MM) DMA engine

RASHPA PCI-e: degenerated case with single MM2MM DMA engine



RASHPA
DETECTOR

12.5 GB/s

RNIC

DMA
engine

12.5 GB/s

PCI-e

<16 GB/s

a)

Mem
channel

nT x 10 GB/s

3 x 12 GB/s

Mem ctrl
I/O
Cache mem
core core
core core
core core
core core

4 GB/s

QPI

Mem ctrl
I/O
Cache mem
core core
core core
core core
core core

b)

12 GB/s

GPU
ACCELERATOR

DMA
engine

RASHPA
RECEIVER

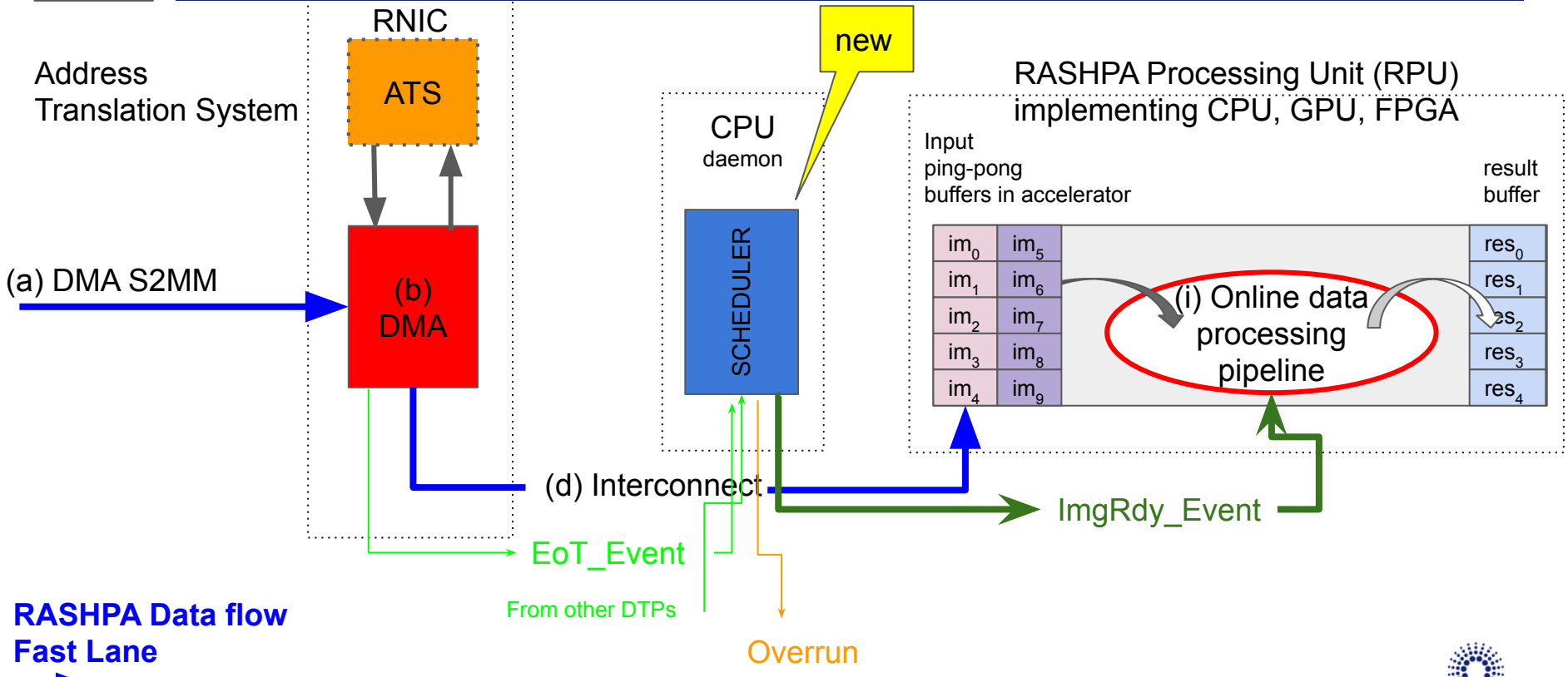
c)

12 GB/s

DMA
engine

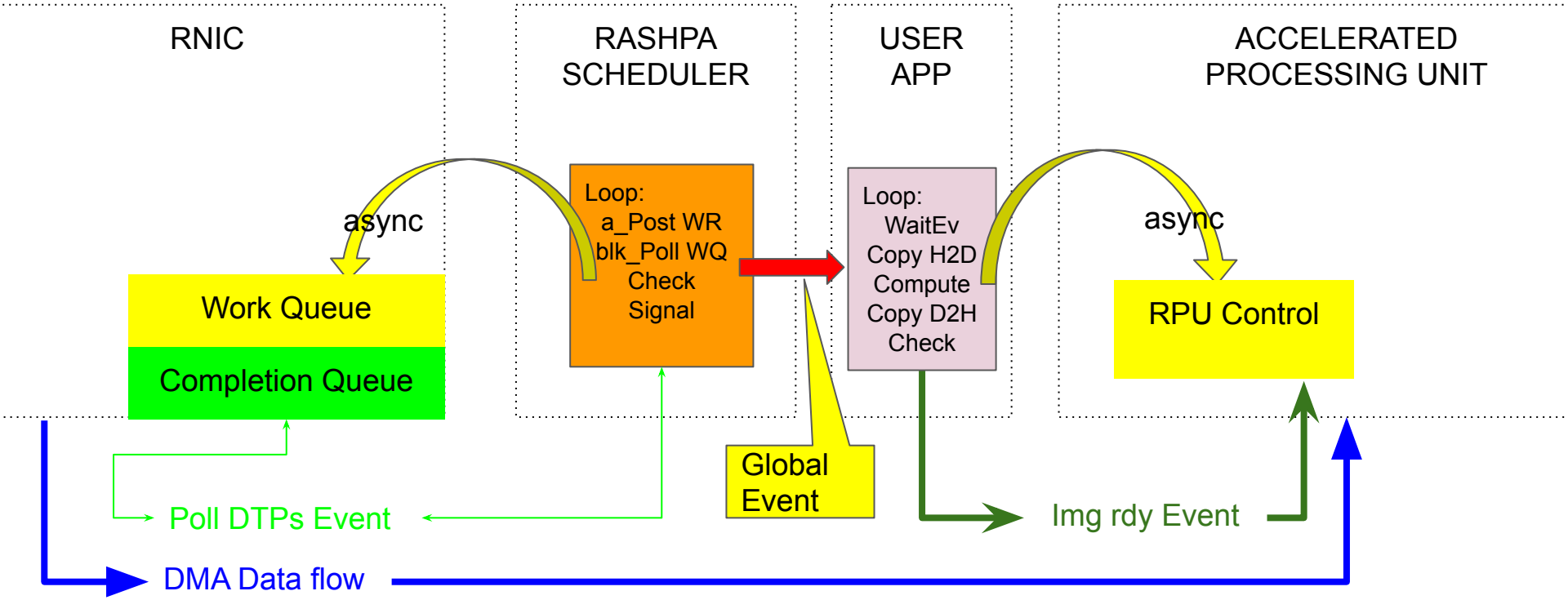
FPGA
ACCELERATOR

RASHPA PROCESSING UNIT



RASHPA Data flow
Fast Lane

RASHPA SCHEDULER



GPU&FPGA AS RASHPA PROCESSING UNIT

Streaming Multiprocessor / HLS IP

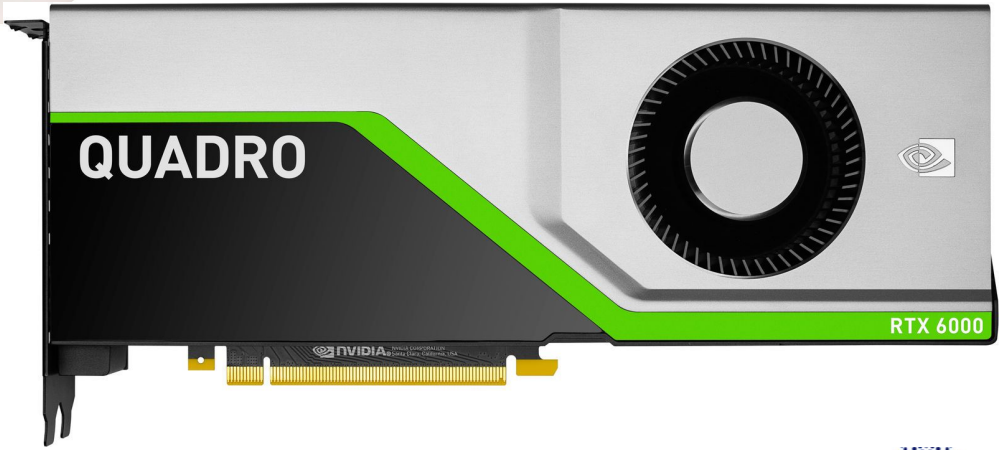
CUDA / C + #pragma

GPU / FPGA

PCI-e, DMA Bypass

Global Memory DDR6 / DDR4

Shared Memory / BRAM



RASHPA EMULATORS & ROCEv2

REMU: DETECTOR SIMULATOR

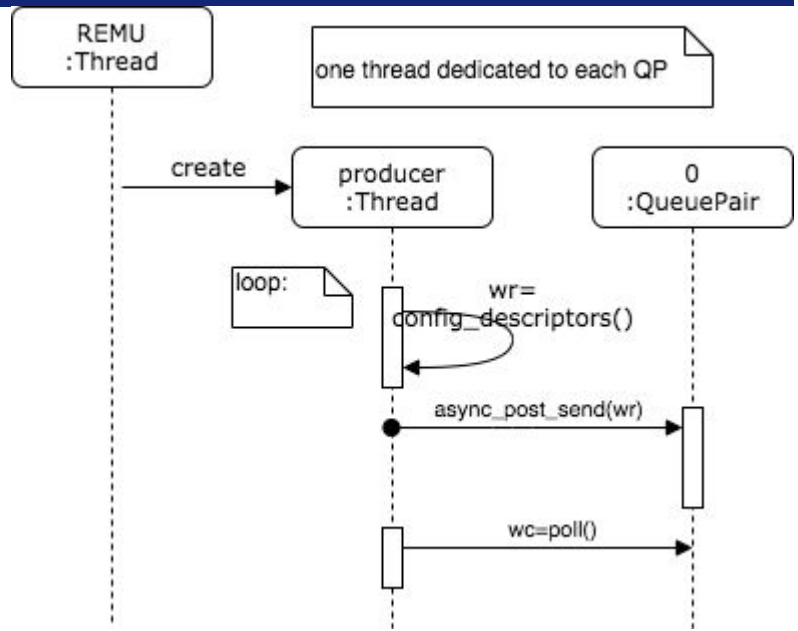
- Custom dataset upload
- RASHPA Manager

ROMULU: ONLINE DATA ANALYSIS

- CUDA, OpenCL / NVIDIA, AMD
- OpenMP, Persistent kernel

COMMON: IBVERBS

- Write Verb, Unreliable Connected Queue Pair (Libibverbs), Connectx-5



FIRST PUBLICATION IN JSR (IUCr)

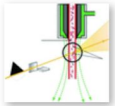
Not Secure | scripts.iucr.org/cgi-bin/paper?S1600577520008140

Journal of Synchrotron Radiation

home archive editors for authors for readers submit subscribe open access

RESEARCH PAPERS

J. Synchrotron Rad. (2020). **27**, 1297-1306
https://doi.org/10.1107/S1600577520008140
Viewed by 137



RDMA data transfer and GPU acceleration methods for high-throughput online processing of serial crystallography images

R. Ponsard, N. Janvier, J. Kieffer, D. Houzet and V. Fristot

The continual evolution of photon sources and high-performance detectors drives cutting-edge experiments that can produce very high throughput data streams and generate large data volumes that are challenging to manage and store. In these cases, efficient data transfer and processing architectures that allow online image correction, data reduction or compression become fundamental. This work investigates different technical options and methods for data placement from the detector head to the processing computing infrastructure, taking into account the particularities of modern modular high-performance detectors. In order to compare realistic figures, the future ESRF beamline dedicated to macromolecular X-ray crystallography, EBSL8, is taken as an example, which will use a PSI JUNGFRAU 4M detector generating up to 16 GB of data per second, operating continuously during several minutes. Although such an experiment seems possible at the target speed with the 100 Gb s^{-1} network cards that are currently available, the simulations generated highlight some potential bottlenecks when using a traditional software stack. An evaluation of solutions is presented that implements remote direct memory access (RDMA) over converged ethernet techniques. A synchronization mechanism is proposed between a RDMA network interface card (RNIC) and a graphics processing unit (GPU) accelerator in charge of the online data processing. The placement of the detector images onto the GPU is made to overlap with the computation carried out, potentially hiding the transfer latencies. As a proof of concept, a detector simulator and a backend GPU receiver with a rejection and compression algorithm suitable for a synchrotron serial crystallography (SSX) experiment are developed. It is concluded that the available transfer throughput from the RNIC to the GPU accelerator is at present the major bottleneck in online processing for SSX experiments.

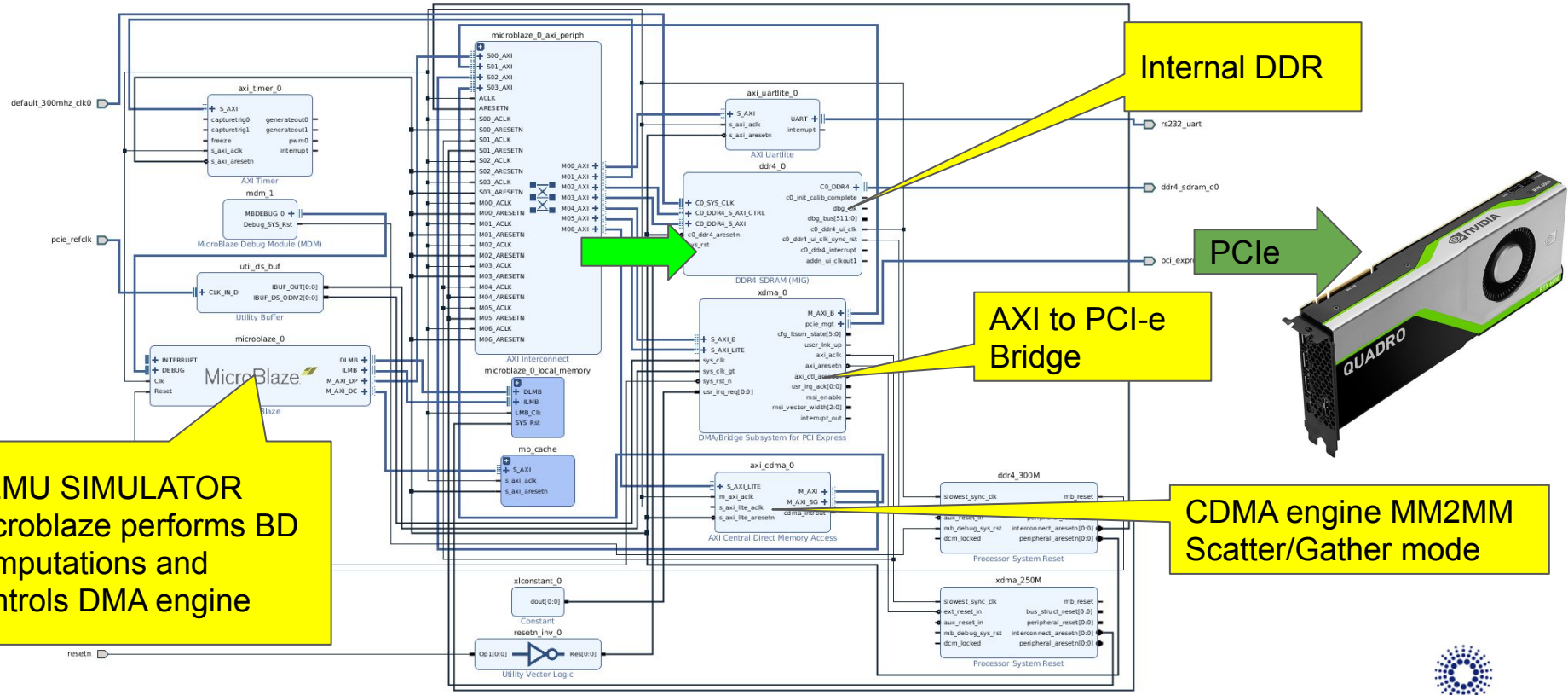
Keywords: online data processing; **RDMA**; **RoCEv2**; **GPU**; **SSX**; **online data analysis**.

Read article Similar articles

Follow J. Synchrotron Rad. E-alerts Twitter Facebook RSS

Search IUCr Journals doi
Author All journals volume page
Advanced search

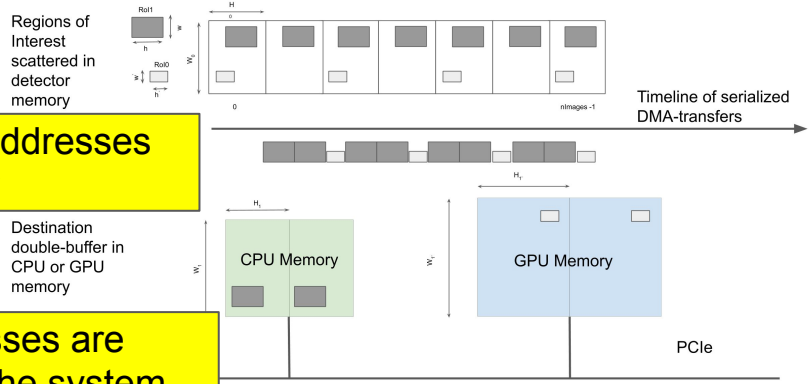
PCIE EMULATOR SOFTCORE CONTROLLED



ONLINE DMA BDs CONFIGURATION

BD[i].**src_addr** = ***FPGA_DDR0_L + DDR_OFFSET***
+ n * d->w0*d->h0
+ d->x0
+ d->y0 * d->w0
+ i * d->w0;

Describes source addresses in detector memory



BD[i].**dest_addr** = (!b) ? ***PA_DDR_L***
+ i * d->w1
+ (n%(2*d->bunch)) * d->w1*d->h1
+ d->x1
+ d->y1*d->w1

: ***PA_DDR_L***
+ i * d->w1
+ (n%(2*d->bunch)) * d->w1*d->h1
+ d->x1+d->y1*d->w1
+ d->w1*d->h1*2 * d->bunch;

Destination addresses are also managed by the system

Problem Statement:

Could next buffer list be computed during previous DMA transfer by a softcore application ?

Online GPUAnalysis using Adaptive DMA Controlled by Softcore for 2D Detectors

R. Ponsard*
ESRF
GIPSA-LAB
Grenoble, France
ponsard@esrf.fr

N. Janvier
ESRF
IEEE
Grenoble, France

D. Houzet
GIPSA-LAB
Grenoble, France

V. Fristot
GIPSA-LAB
Grenoble, France

W. Mansour
ESRF
IEEE
Grenoble, France

***Abstract*— New generation X-ray detectors enables cutting-edge experiments that can produce very high throughput data streams that are challenging to manage and store. This paper presents an evaluation of a configurable data placement mechanism from an FPGA device collecting detector raw data to a burst-cache memory and concurrently to a GPU accelerator, bypassing hardware and software extraneous copies and bottlenecks via PCI-Express. It includes a DMA controller dynamically configured in real-time by a Microblaze soft-processor. A low-latency synchronization mechanism using GPUDirect technology is presented as well. Multi-GB, DMA-able memory buffer allocation, leveraging Linux contiguous memory allocator is investigated. As illustrative workloads, real-time raw-data correction as foreseen in Serial Synchrotron X-ray experiments were processed. Obtained results showed that if one could reach a data throughput of 12.7GB/s to CPU memory when using PCIe gen3 x16, a 12-cores OpenMP CPU application processes the raw data only up to 2.7GB/s and is outperformed by a GPU accelerator (NVIDIA RTX 6000).**

FPGA; PCIe; DMA; CMA; GPUDirect; Microblaze

I. Introduction

Many photon science experiments are producing huge amount of data at high repetition rates. Using X ray detectors at full capacity requires not only a high throughput data links, but also challenges traditional workflow (acquisition, transfer, storage, batch processing) putting tremendous pressure on storage and computing infrastructures. Therefore, this enforces online data analysis in real-time pipeline and immediate rejection of non-pertinent data or compression to alleviate the pressure on the storage system.

Serial Synchrotron Crystallography (SSX) is one of the most demanding use case as it produce large amount of data for a long time and will be used in this paper as a realistic challenge as far as data processing is concerned. The foreseen detector has an adaptive gain feature: the processing of a single 4M pixels image requires 8 million 32-bit floating-point operations and will produce nearly one Terabyte of data in one minute [1].

A. Overview of Bottlenecks in Computer Architecture

In traditional design, CPUs are the main bottleneck source when data transfer is concerned. This problem can be mitigated using Direct Memory Access engine (DMA).

Other bottlenecks in the operating system itself might affect the throughput of the data transfer:

- during multiple copies of data from the kernel space to user space,
- when the processor goes from user to supervisor state and vice-versa, at each system call or interrupt handling.

B. Background of Online Data Processing in GPU

During the limited time slot available between two images (1ms), a GPU accelerator is the best candidate to perform detector raw-data correction, followed by image geometrical reconstruction and then by data compression or rejection.

This data treatment requires transferring the detector data into the GPU accelerator and a low-latency trigger mechanism to synchronize the computation and DMA data flow as well.

C. Proposed FPGA/GPU system

Figure 1. shows the proposed system. An X-ray detector Front-end electronics outputs data on Xilinx Aurora serial link. In the backend computer, an FPGA board serves as an Aurora/PCIe bridge.

A DMA engine push data to their final location in main memory for storage and batch processing or straight into GPU memory for fast-feedback. In this paper, we are only discussing the right side of the design.

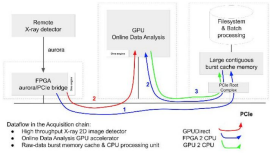


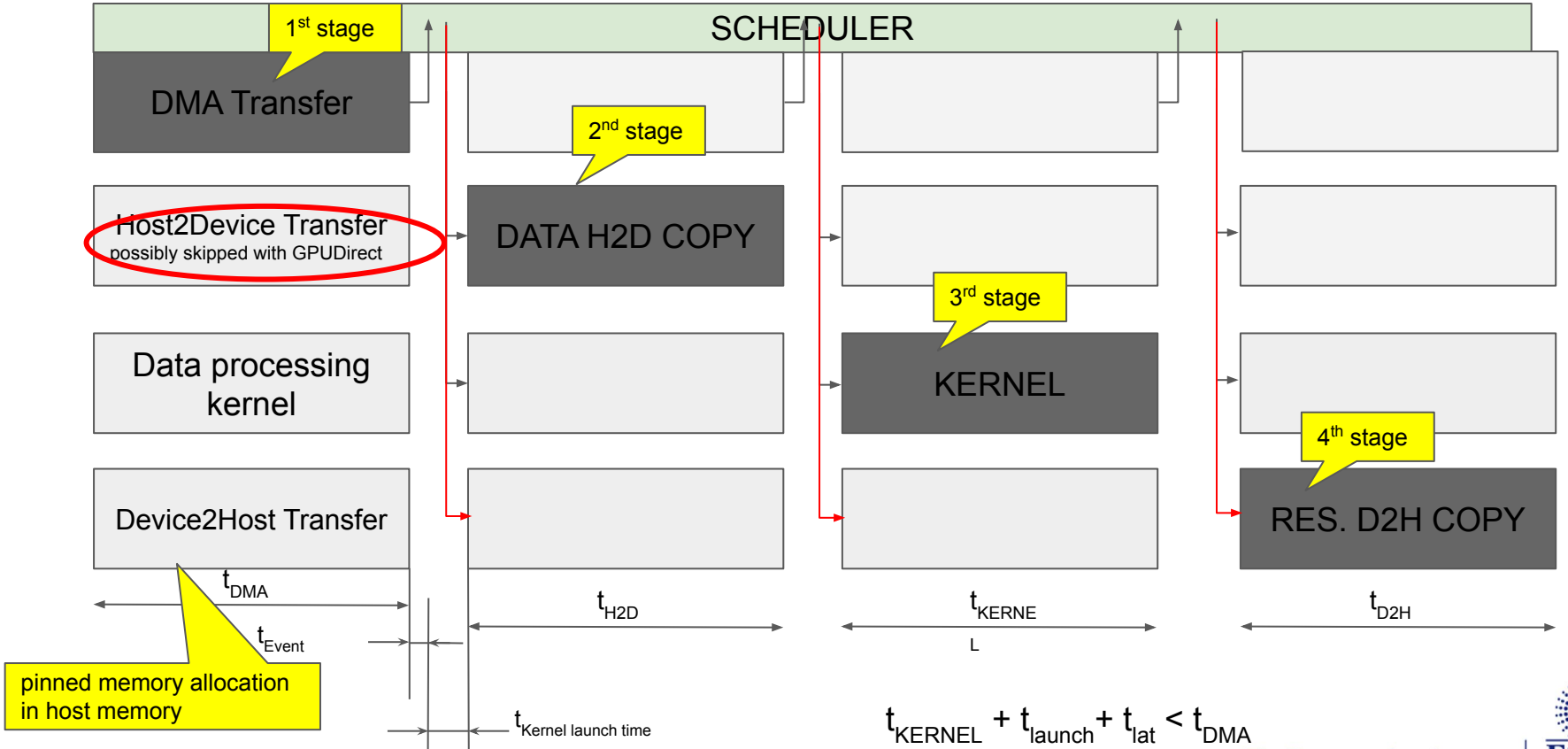
Figure 1. Architectural overview of the data acquisition system.

D. State of the Art

RDMA transfer is routinely done by using Mellanox Technologies adapters. Our framework address the other use cases, not compliant with Ethernet or Infiniband links. Many authors have proposed frameworks leveraging different GPU accelerators and programming API such as AMD/OpenCL [2] or GPUDirect/CUDA [3].

978-1-7281-9535-3/20/$31.00 ©2020 IEEE
DOI 10.1109/DSD51259.2020.00075

DATA PROCESSING PIPELINE: ROMULU



ONLINE DATA ANALYSIS

Raw data preprocessing

3 gain level/pixel
=>Up to 6 parameters
for each pixel

$$Pixel_{i,j}\,[keV\,] = \frac{\left(Raw_{i,j}\,[ADU] - Pede_{i,j}[ADU]\right)}{Gain_{k,i,j}\left[\frac{ADU}{keV}\right]}$$

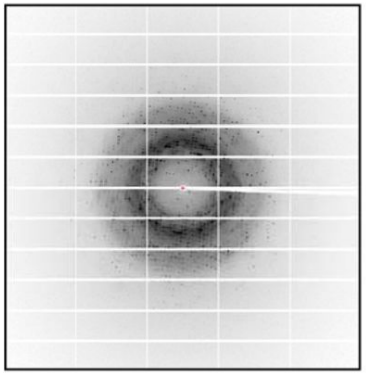
Rejection algorithm based on
Bragg's Peak counter (< 1 - 10% hit expected)

- AtomicInc
- Shared memory

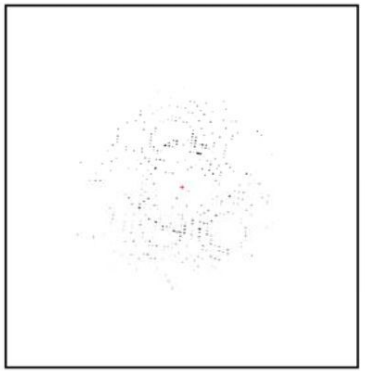
Compressed Sparse Row Matrix format (CSR)

- Based on Cum-scan
- PyCUDA Meta compiler generated code

Total Scattering



Bragg Peaks



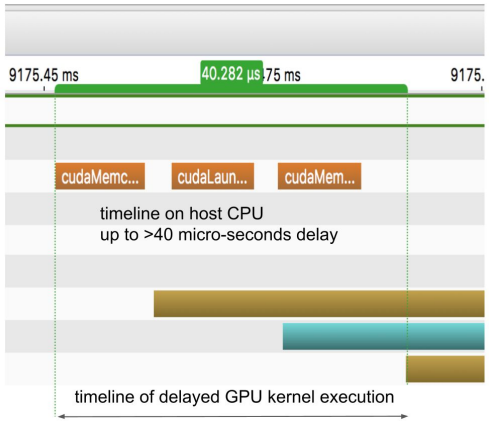
LOW LATENCY GPU SYNC. MECHANISM

CUDA kernels launch time not negligible when done from host driver (40 µs)

Reduced to 5 µs

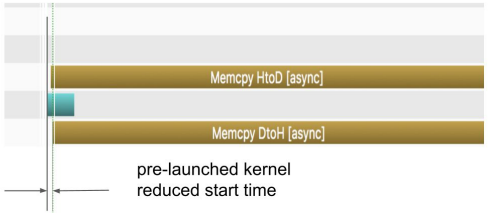
CUDA stream memory operations

- Kernels are pre-launched and put on hold on CUDA streams (cuStreamWaitValue32)
- Triggered DMA end of transfer



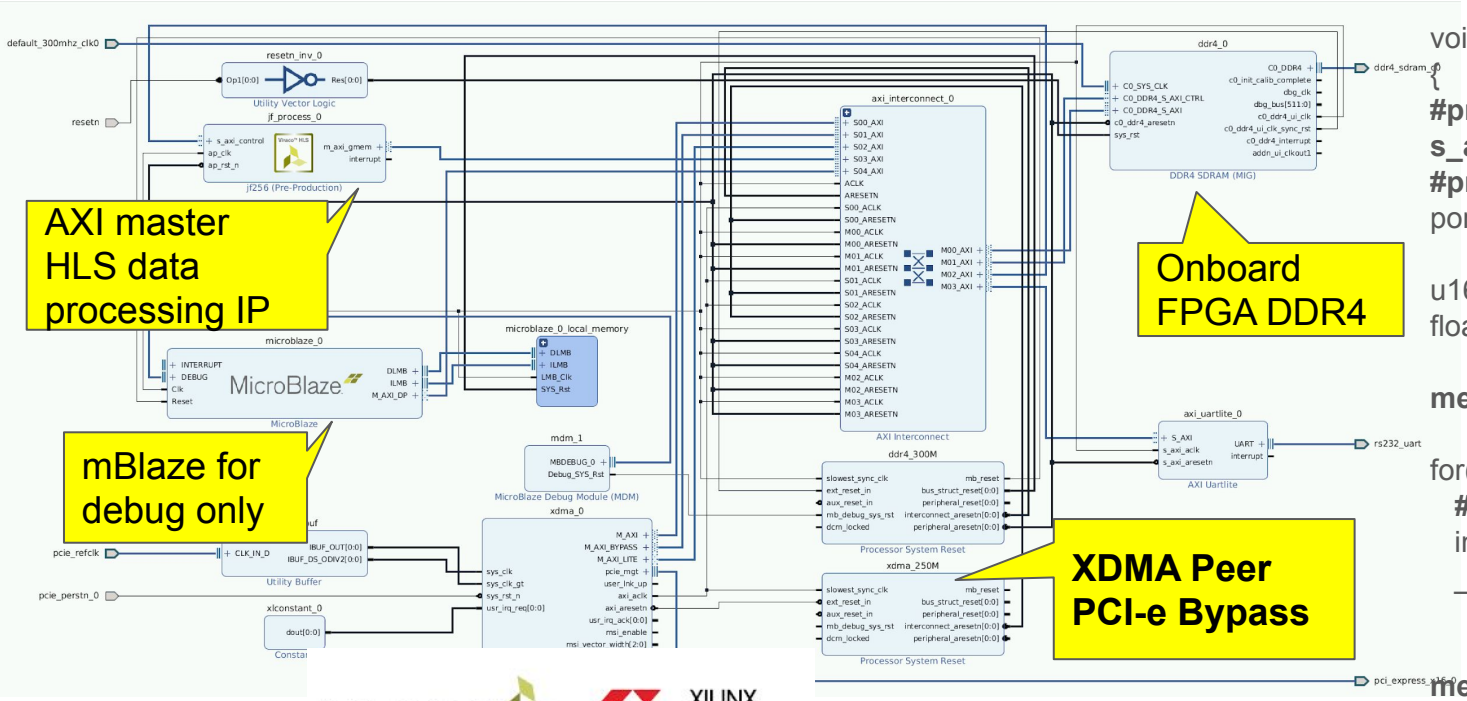
using proposed synchronization mechanism:

- host CPU pre-launches GPU commands on CUDA streams
- cuStreamWaitValue32 CUDA memory operation puts stream on hold
- upon RDMA completion, CPU triggers memory lock polled by GPU
- GPU processing starts within <5 micro-seconds



Decreased launch time

DATA PROCESSING IN FPGA




```
void jf_process(u16 raw[N],res[N])
{
#pragma HLS INTERFACE
s_axilite port=raw bundle=control
#pragma HLS INTERFACE m_axi
port=raw depth=N

u16 _raw[N],;
float _res[N];

memcpy(_raw, raw,

for( i=0; i < N; i+=4)  {
  #pragma HLS UNROLL
  int l=(_raw[i] >> 14);
  _res[i] = ((_raw[i] & 0x3fff) -
                  _pede[i]) / _g[i+l*N];

memcpy(res,_res,
}
```

VIVADO HLS

XILINX VITIS

The European Synchrotron | ESRF

Thank you for your attention

www.esrf.eu

Take home message

RASHPA

RDMA-based **A**cquisition **S**ystem
for **H**igh **P**erformance **A**pplications.

Zerocopy / RDMA / RoCEv2 / GPUDirect

Correspondence email

raphael.ponsard@esrf.fr[a,b] PhD Student

Acknowledgements

N. Janvier[a], D. Houzet[b], V. Fristot[b], W. Mansour[a]

[a]ESRF, Grenoble, FR
[b]GIPSA-LAB, Grenoble Alpes University, FR