l'Observatoire de Paris — LESIA
Laboratoire d'Études Spatiales et d'Instrumentation en Astrophysique

Durham University

MICRO GATE

PLDA

RTC4AO 2018

Green Flash

AO real-time control with GPUs and FPGAs

Julien BERNARD

European Commission

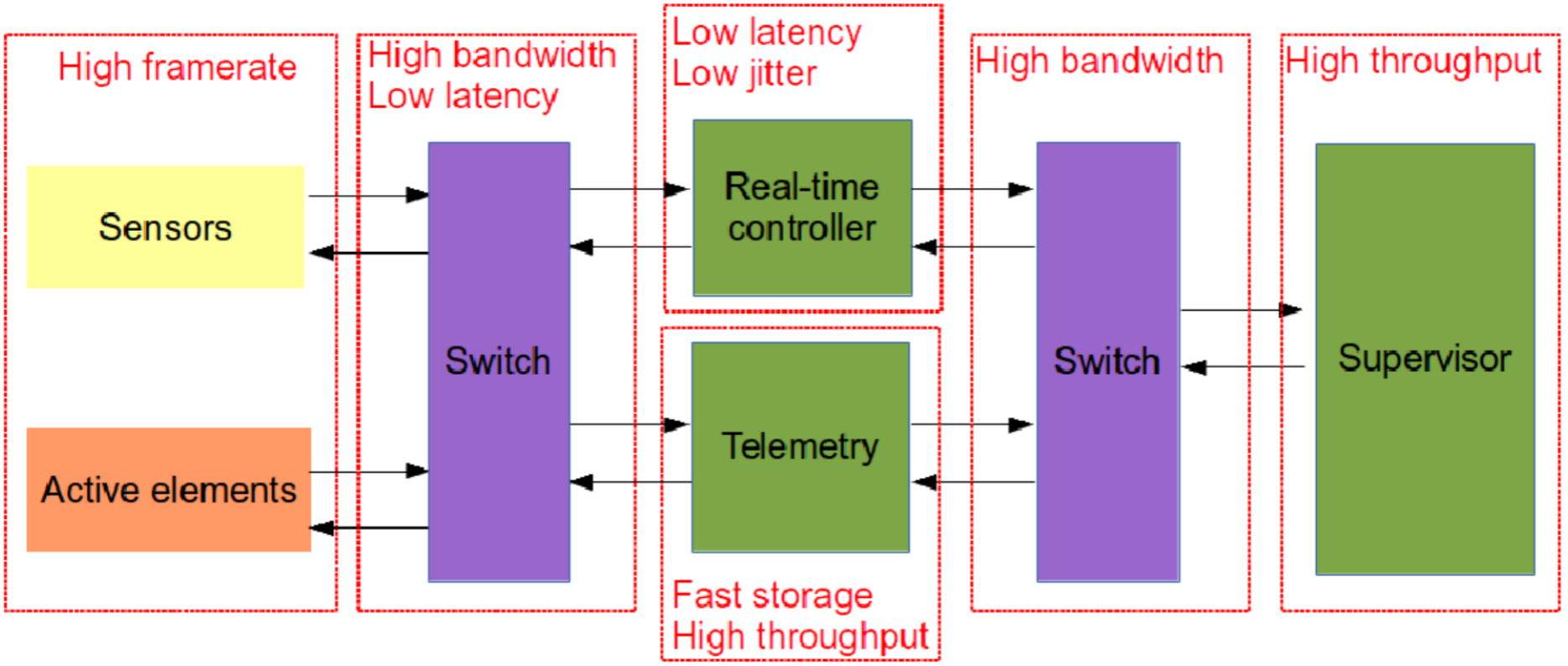
Project #671662 funded by European Commission under program H2020-EU.1.2.2 coordinated in H2020-FETHPC-2014

Summary

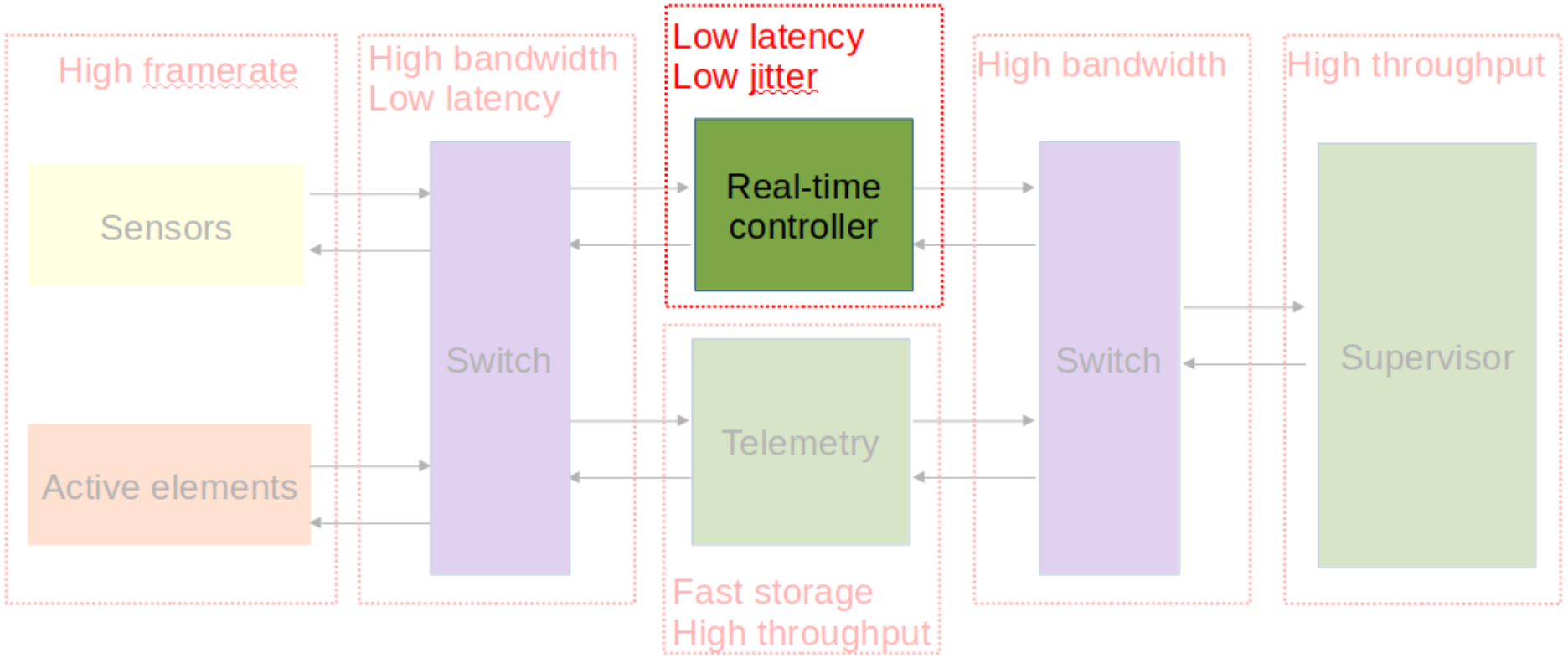
- Architecture solution for ELT AO RTC
- Software solution
  - Legacy
  - GPU direct & I/O Memory mapping
  - Persistent Kernel
- Results
- Conclusion

Architecture solution

RTC concept for ELT AO : Proposed solution



RTC concept for ELT AO : Proposed solution

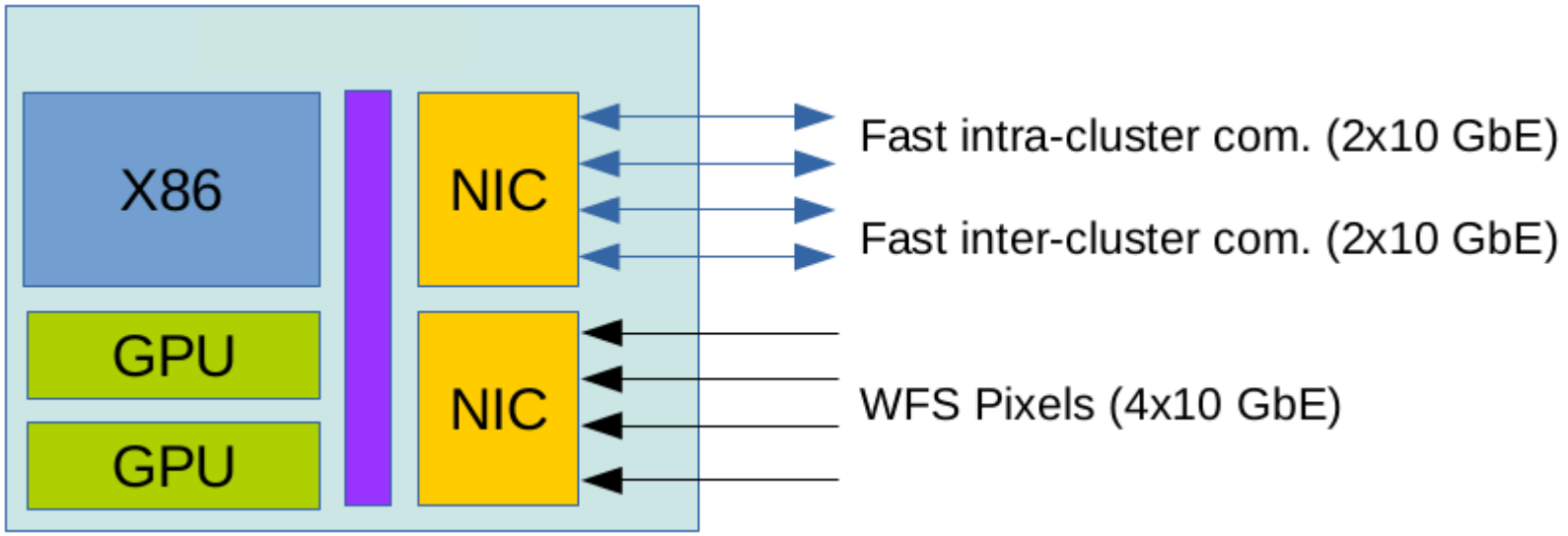


SCAO RTC architecture

SCAO prototype **Brahmin**

Configuration:

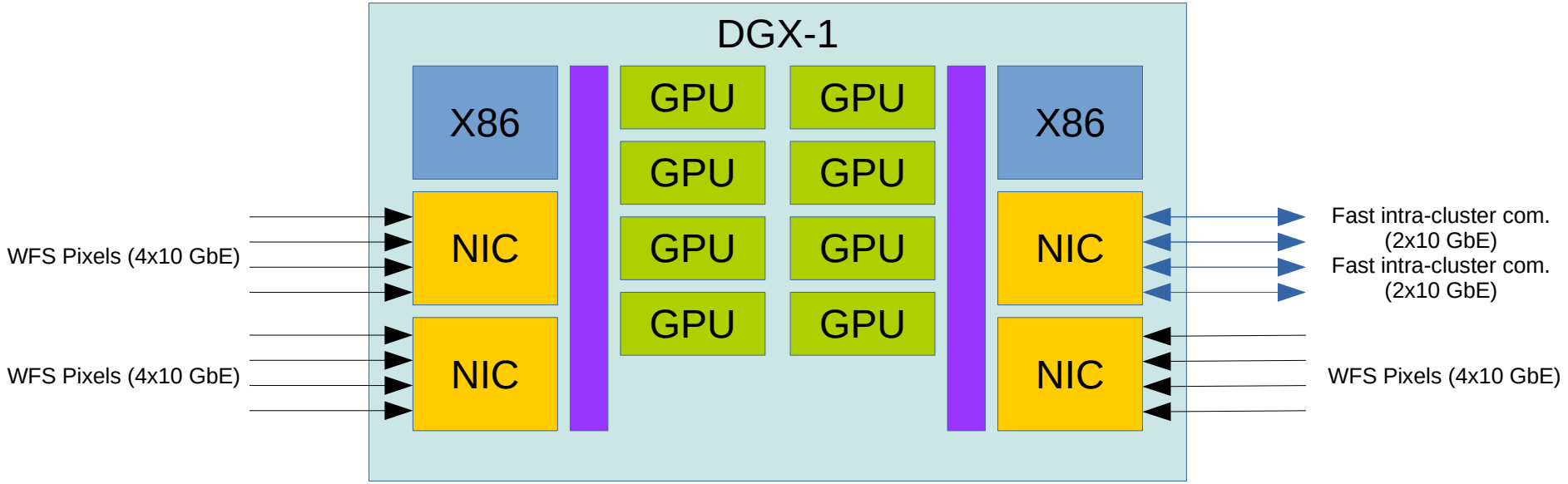
- CPU: Intel i7-5930K 6 cores @ 3.50GHz
- GPU: Nvidia Quadro K6000
  15 Multiprocessors @ 902MHz
  12 GB GDDR5 @ 288GB/s
- FPGA:
  Microgate Xcomp



MCAO RTC architecture

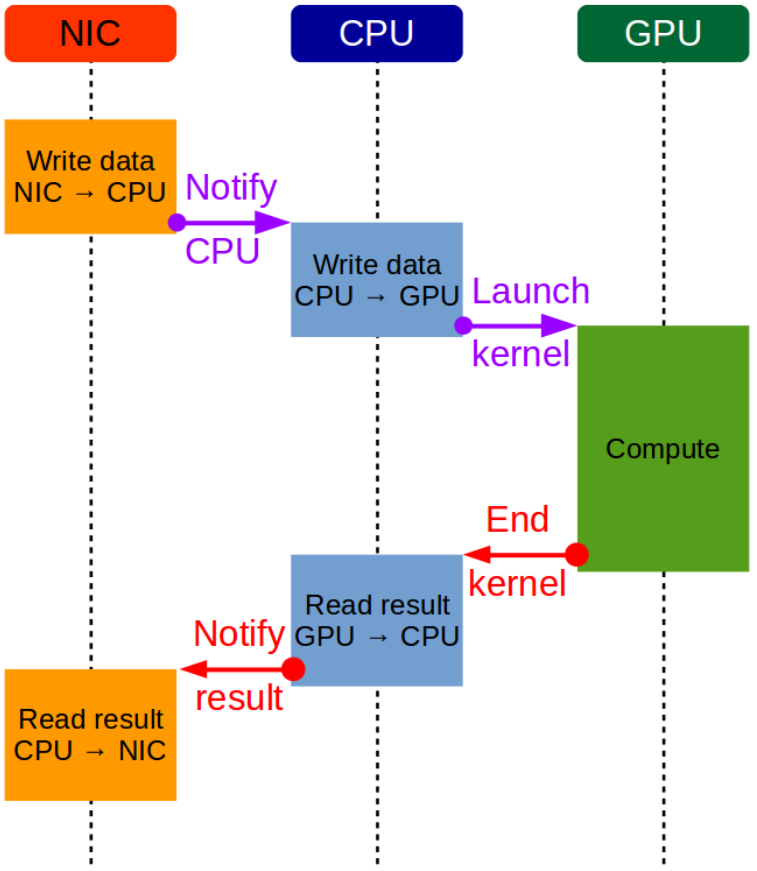
MCAO prototype **Nirvana**
DGX-1 server Configuration:

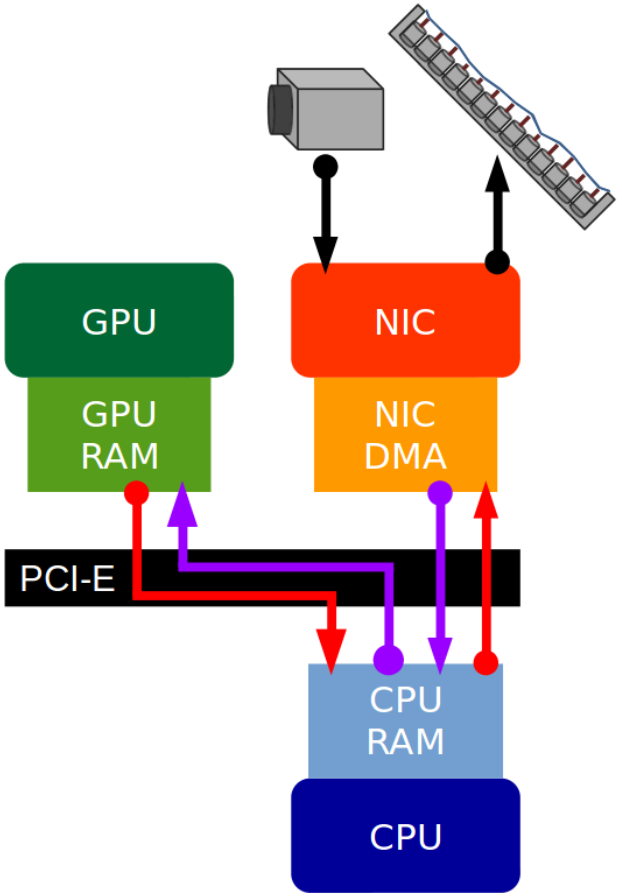
- CPU: Intel Xeon 40 cores @ 2.2GHz
- GPU: 8 Tesla P100
  56 Multiprocessors @ 1.48GHz
  16 GB HBM2 @ 732GB/s



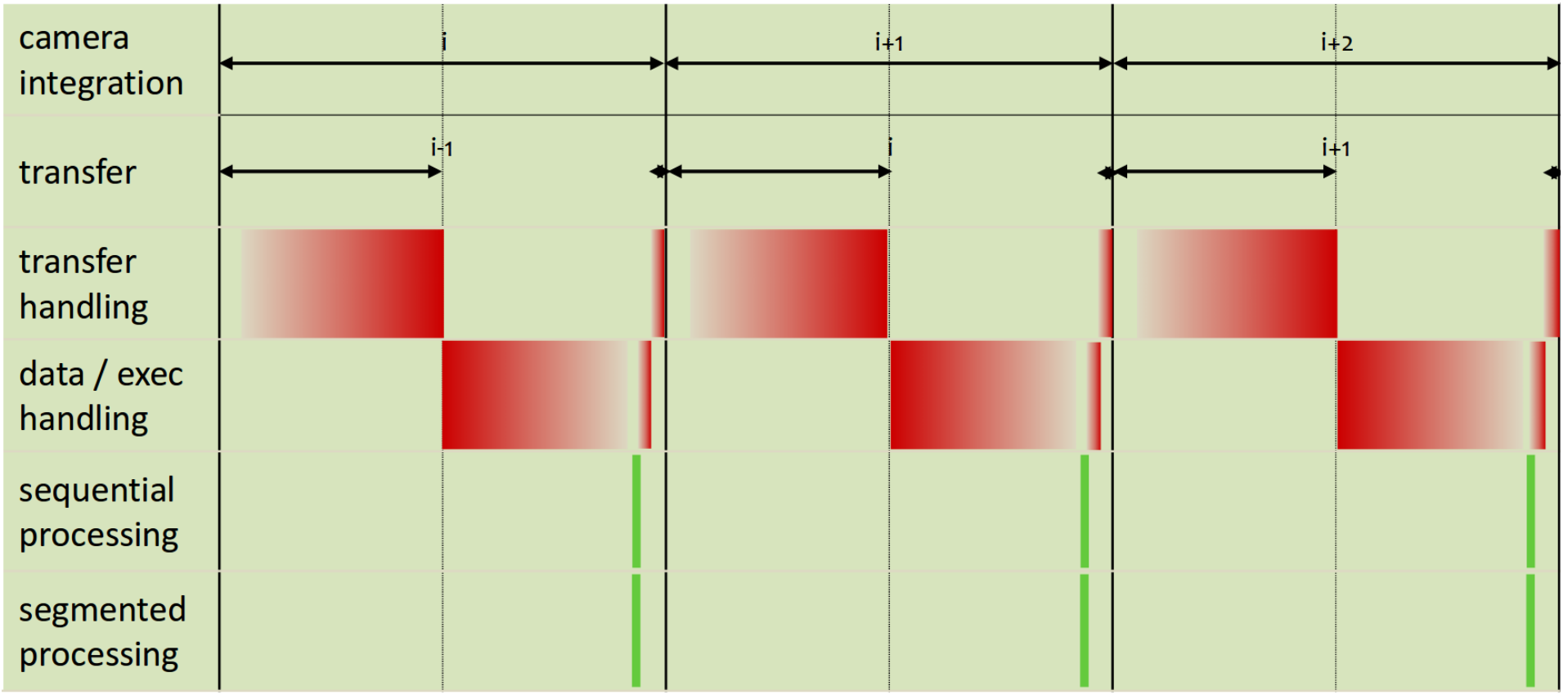
Software solution

Legacy solution

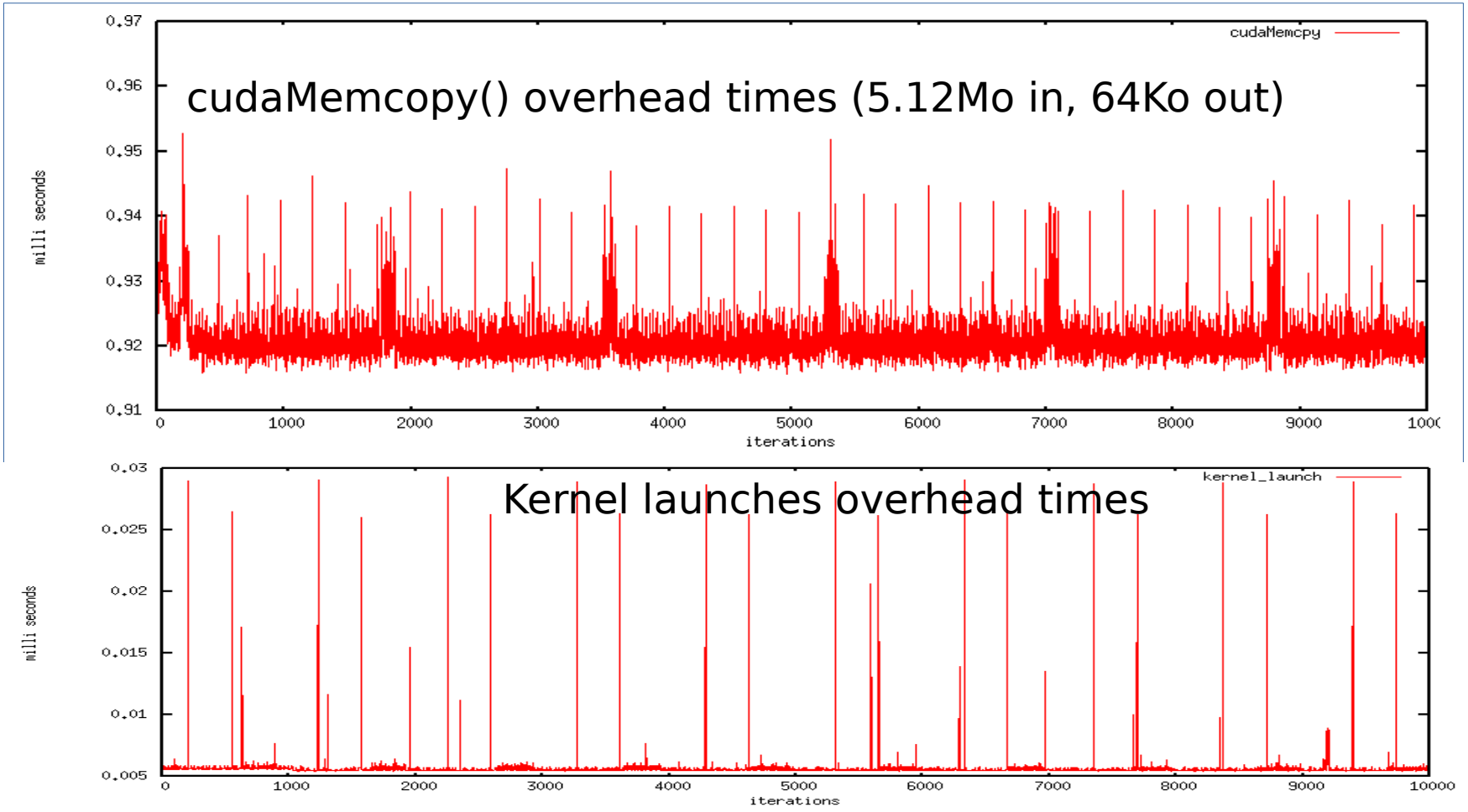




Legacy solution timing



Legacy solution jitter



Improvement

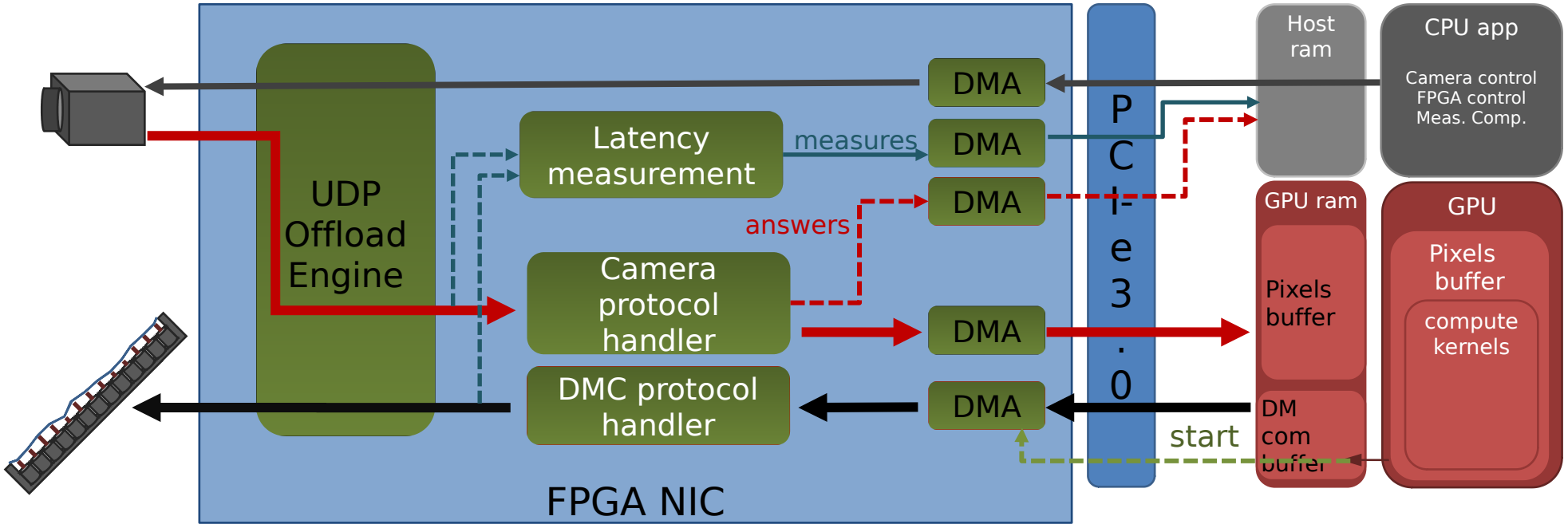
GPU direct & IO
Memory mapping

Persistent Kernel

GPU direct & I/O Memory mapping

I/O memory mapping

- The CPU gets the address of dedicated GPU buffers and use it to configure the NIC DMA engine
- The data are written/read directly to/from the GPU memory



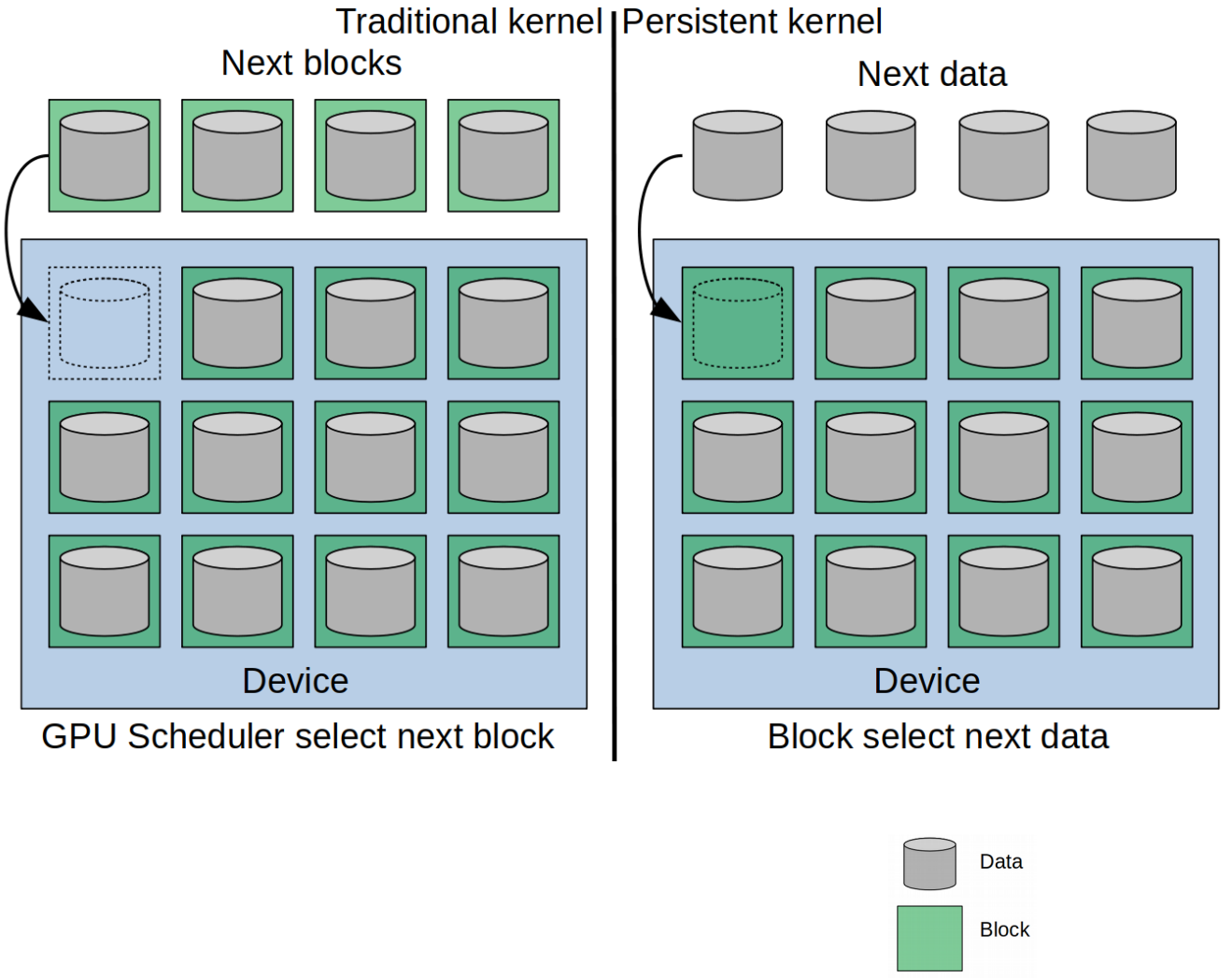
I/O memory mapping implementation

- FPGA need physical address to access GPU buffers
- Use Nvidia gdrcopy
  - Provide kernel module to expose physical address
- Retrieve physical bus address
- Configure FPGA DMA engine

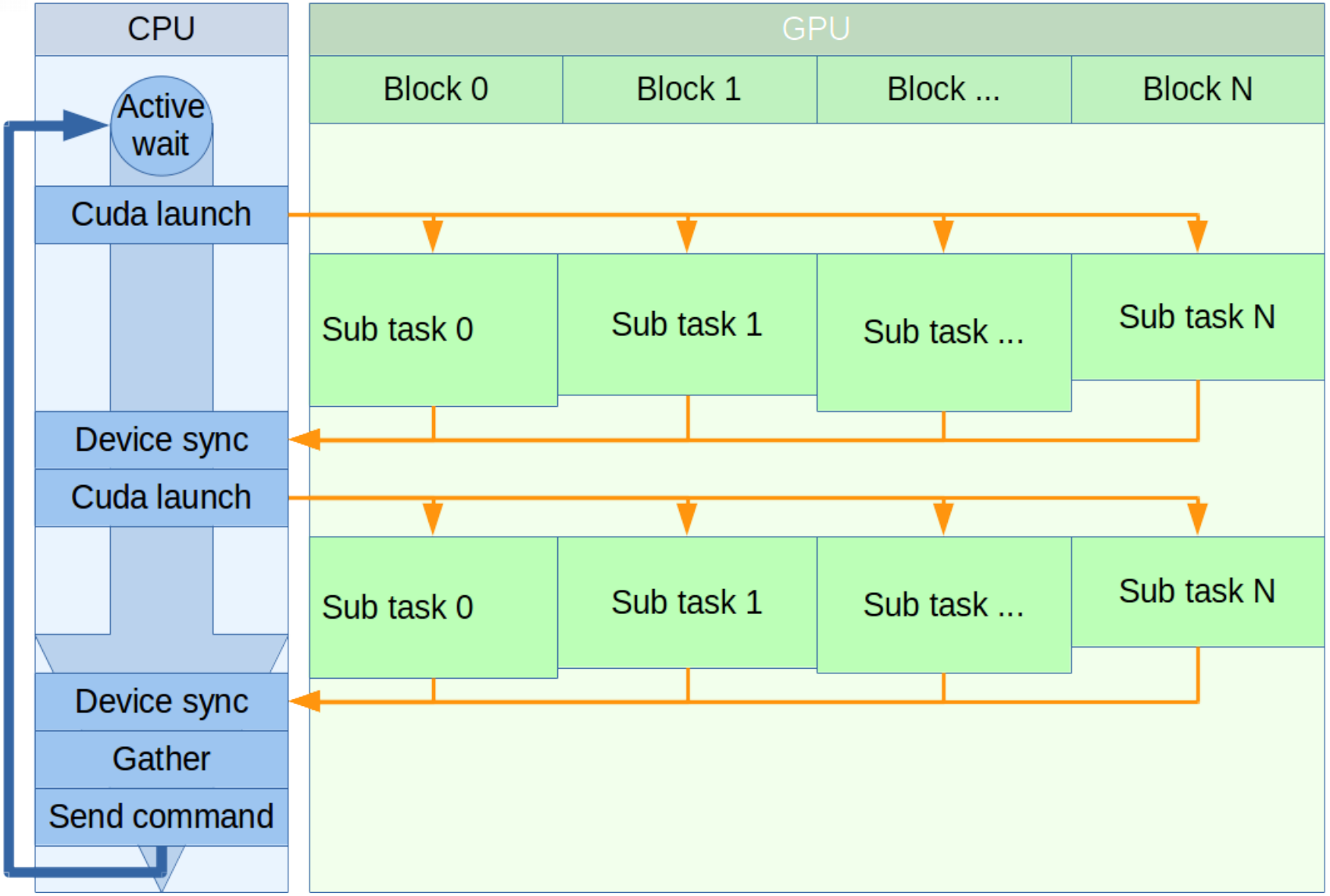
Persistent Kernel

Persistent kernel theory

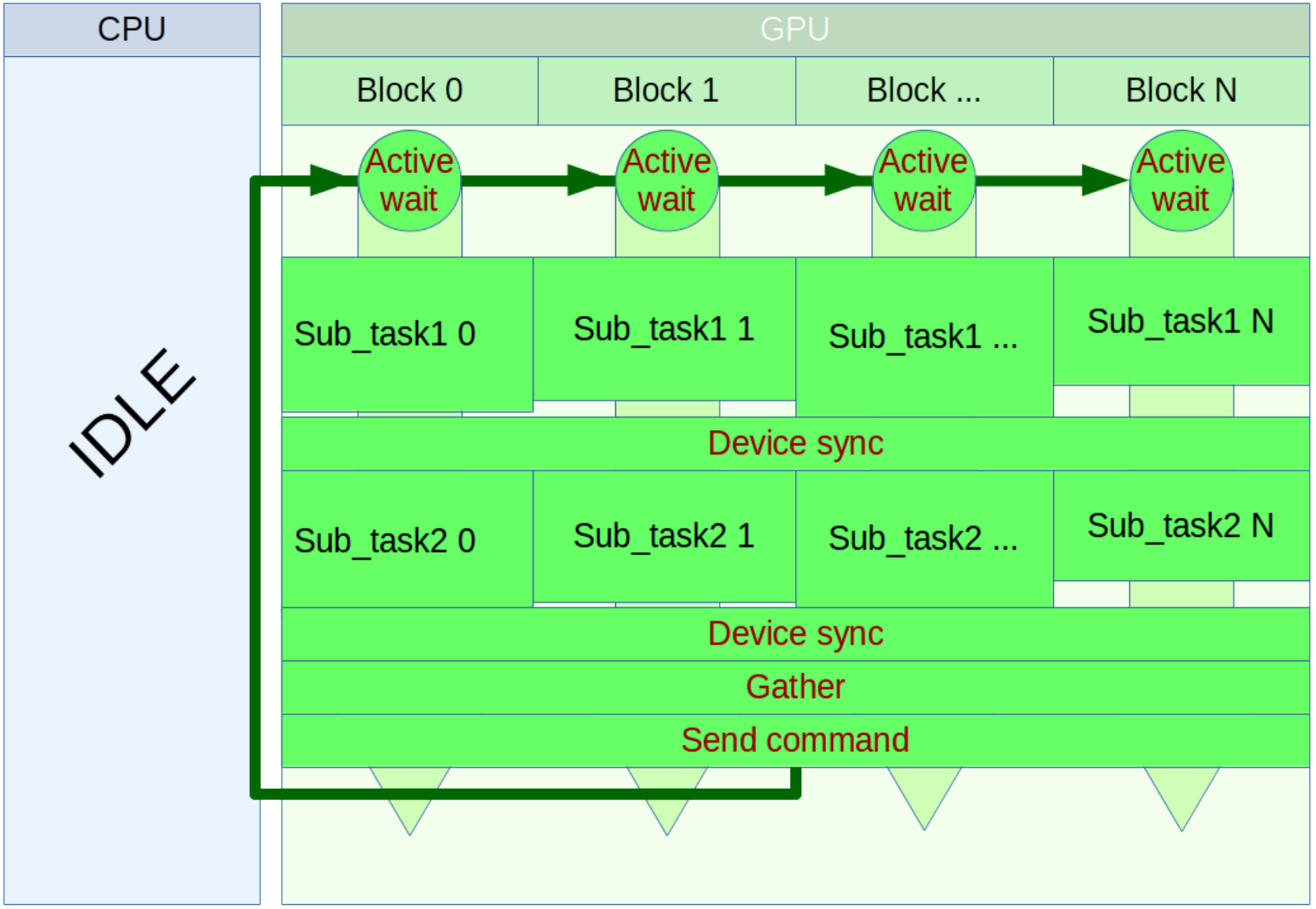
- Traditional kernel
  - Number of block depends on the problem size
  - Data is assigned to a specific block
  - Blocks are independent
  - GPU scheduler assign remaining block to available processing cores
- Persistent kernel
  - Number of block depends on the GPU capabilities
  - Blocks process data as necessary
  - Blocks are all executed simultaneously
  - GPU scheduler is not used after the initialization



Classic implementation



Persistent kernel implementation



Memory polling

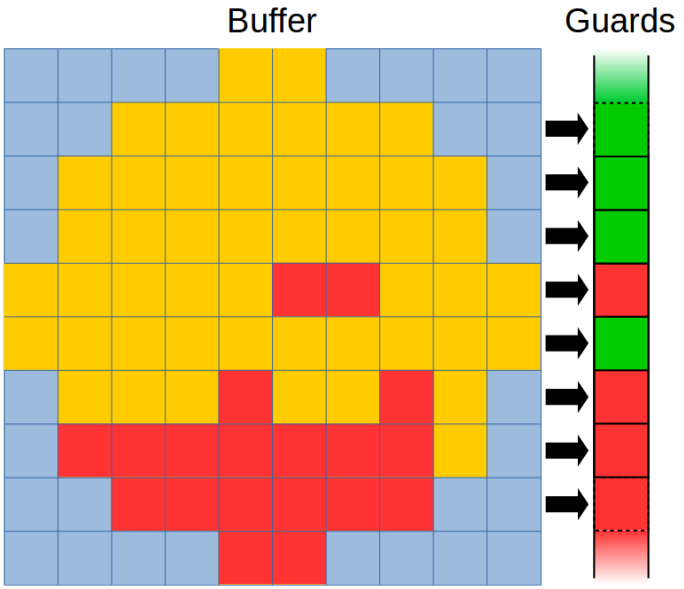
- Detecting data using unused state
  - 10-bit integer on 16-bit word : 4-bit remaining
  - IEEE 754 float : NaN
- Using extern guard associate with specific chunk of data

GPU

```
__device__ uint16_t waitPixel(
    const uint16_t volatile * frame
) {
  uint16_t res;

  while ((res = *frame) == 0xfc00);

  return res;
}
```



GPU notifying

- GPU need to access the FPGA register in order to notify the command availability
- Require superuser privilege

1) Get FPGA register address using Quickplay api

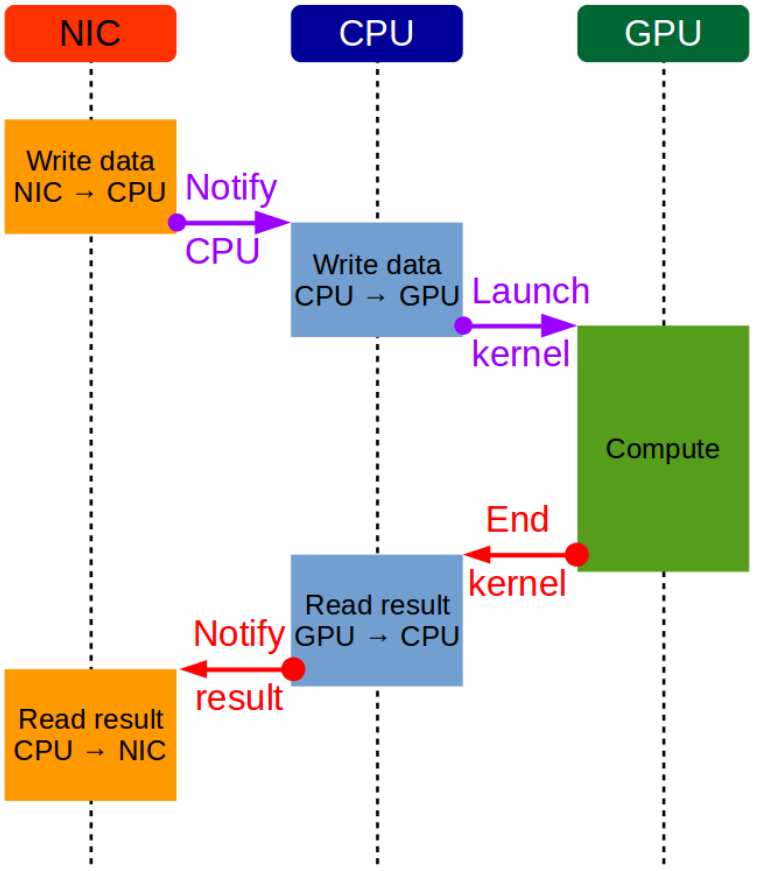
2) Map the address to the GPU Universal Virtual Address (UVA) Space

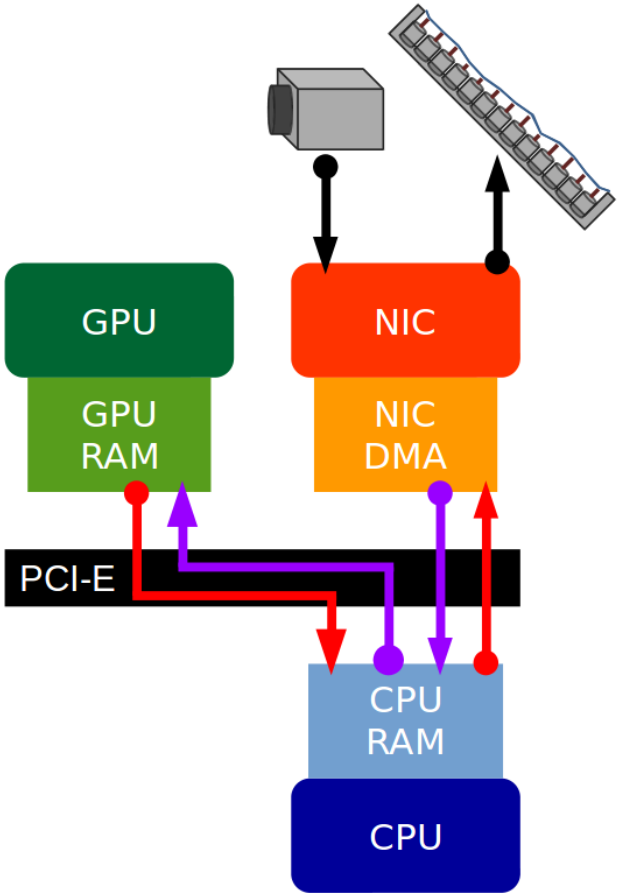
➔ cudaHostRegister()

3) May need to get device address of register address

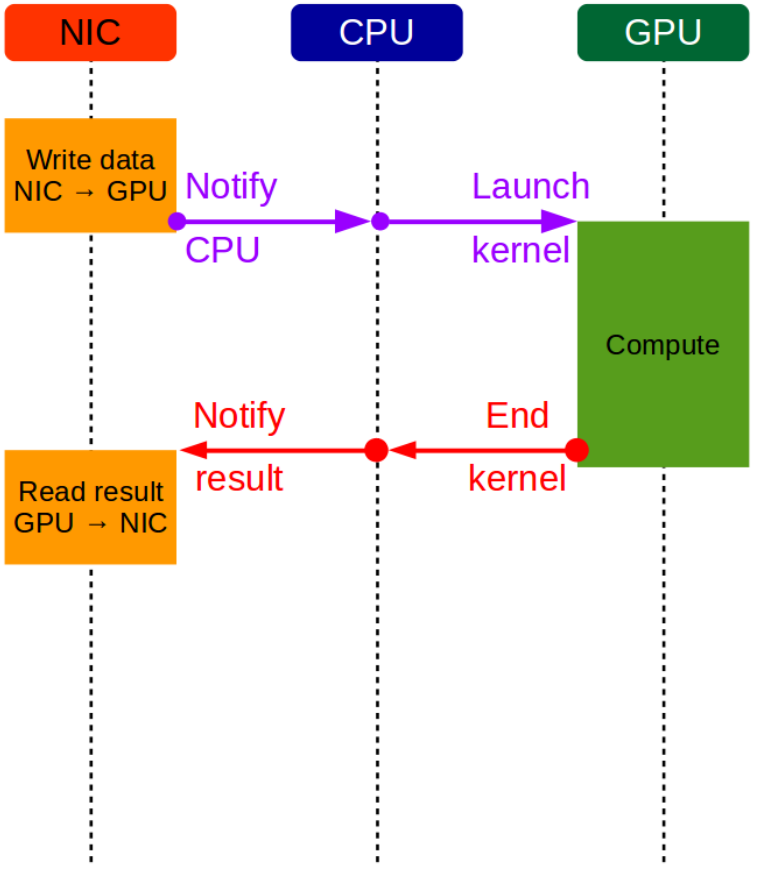
➔ cudaHostGetDevicePointer()

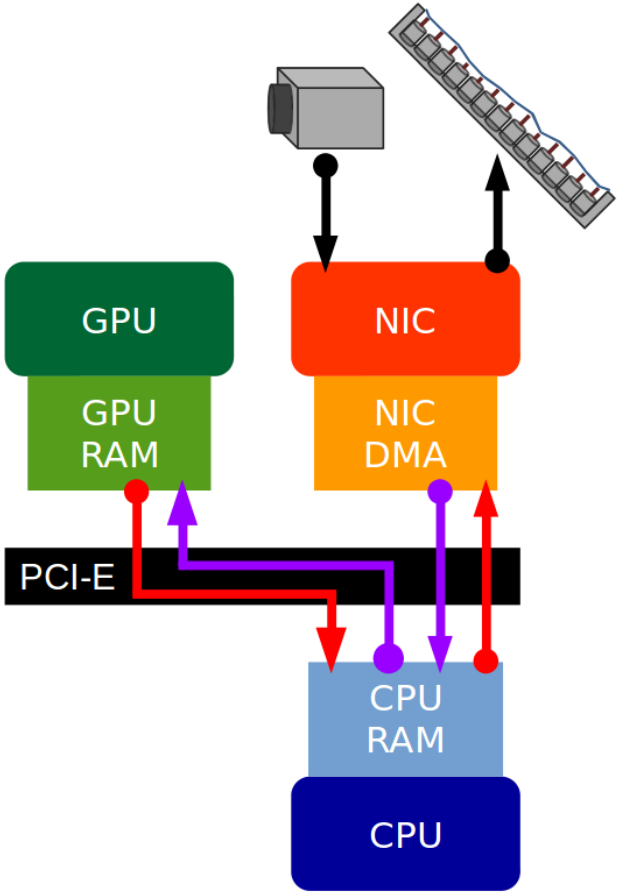
Legacy solution



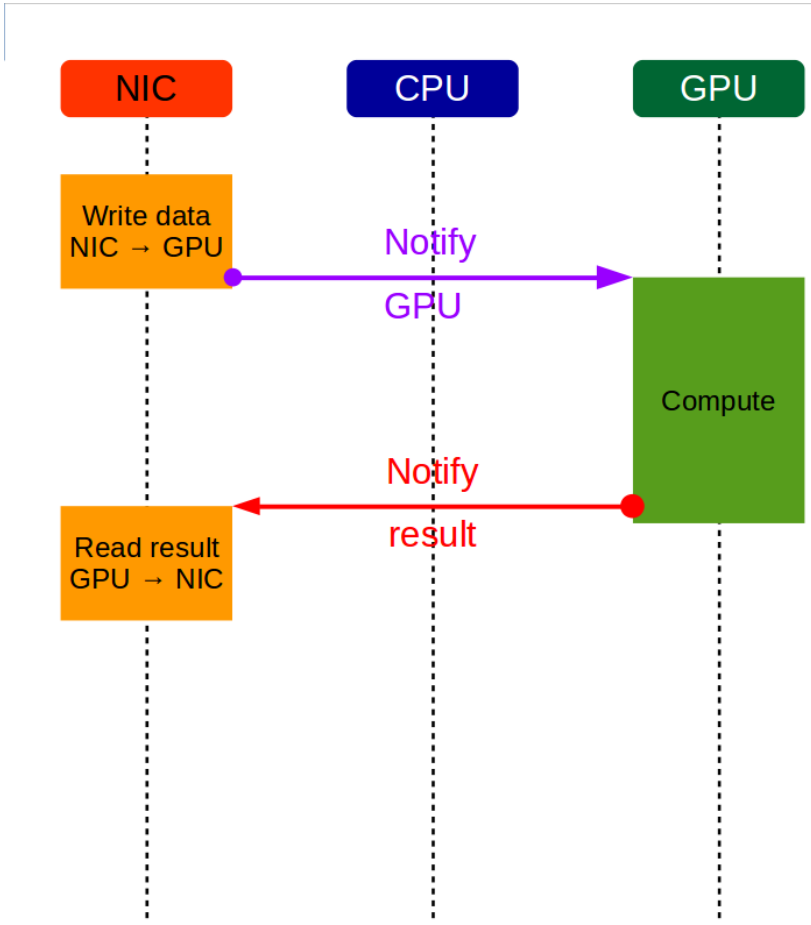


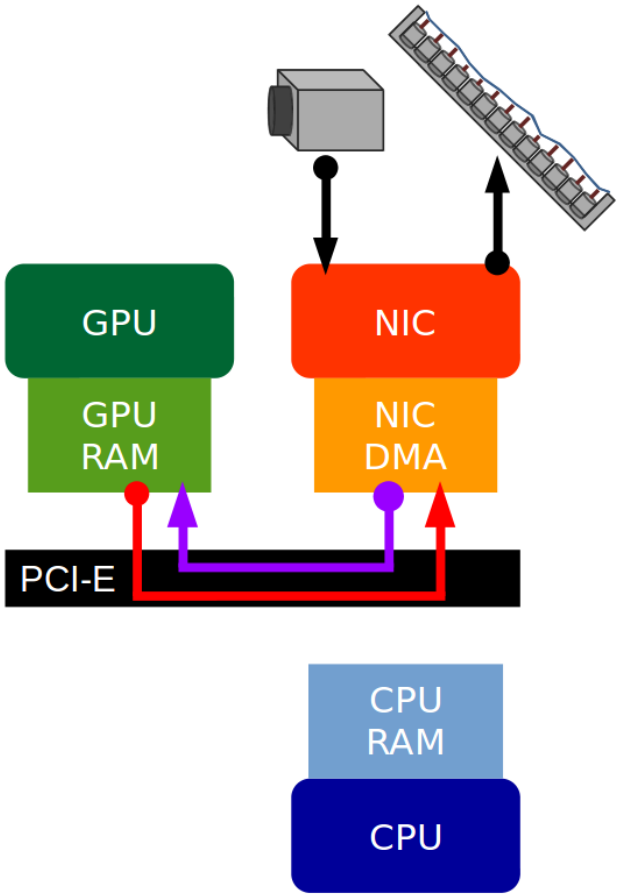
GPU direct , I/O Memory mapping solution



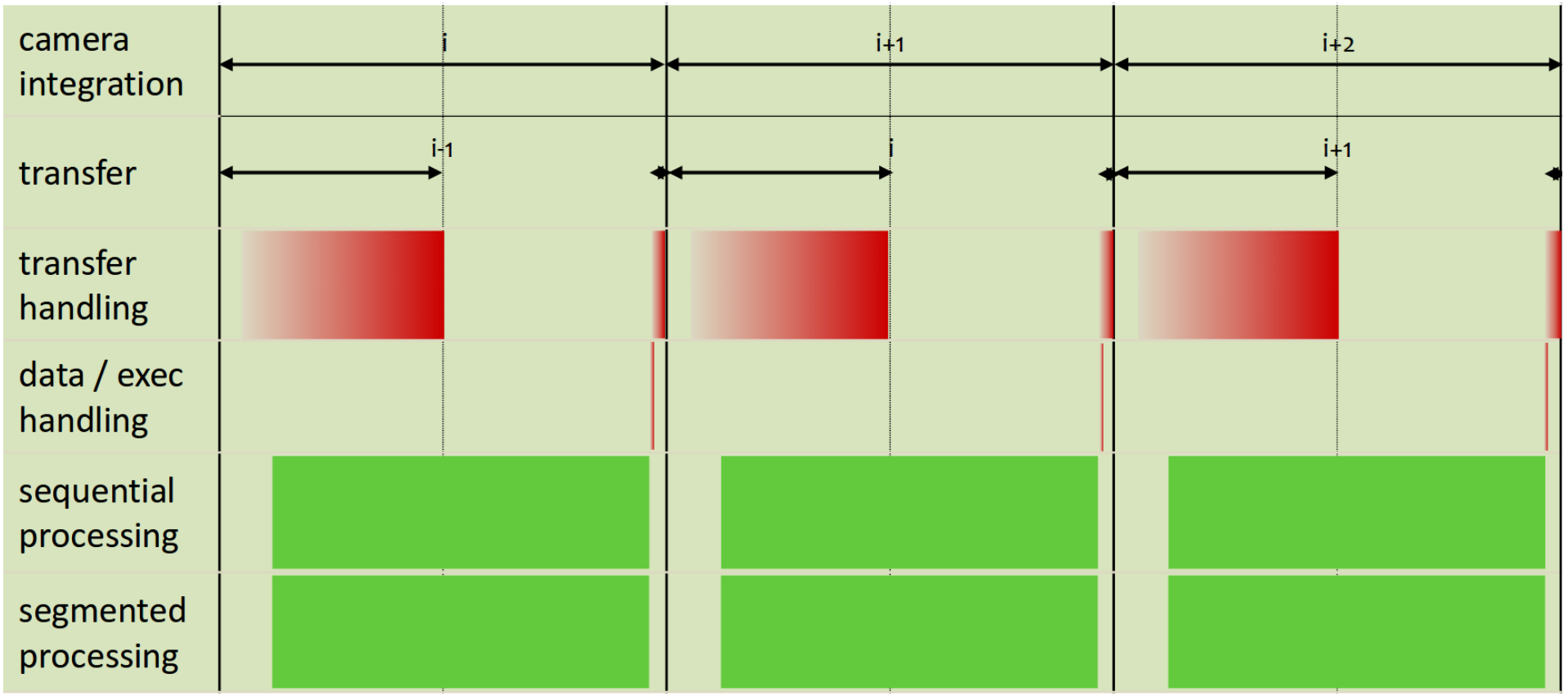


GPU direct, I/O Memory mapping & Persistent kernel solution





GPU direct, I/O Memory mapping & Persistent kernel timing




Green
Flash

Results


l'Observatoire
de Paris
Durham
University
MICRO GATE
PLDA

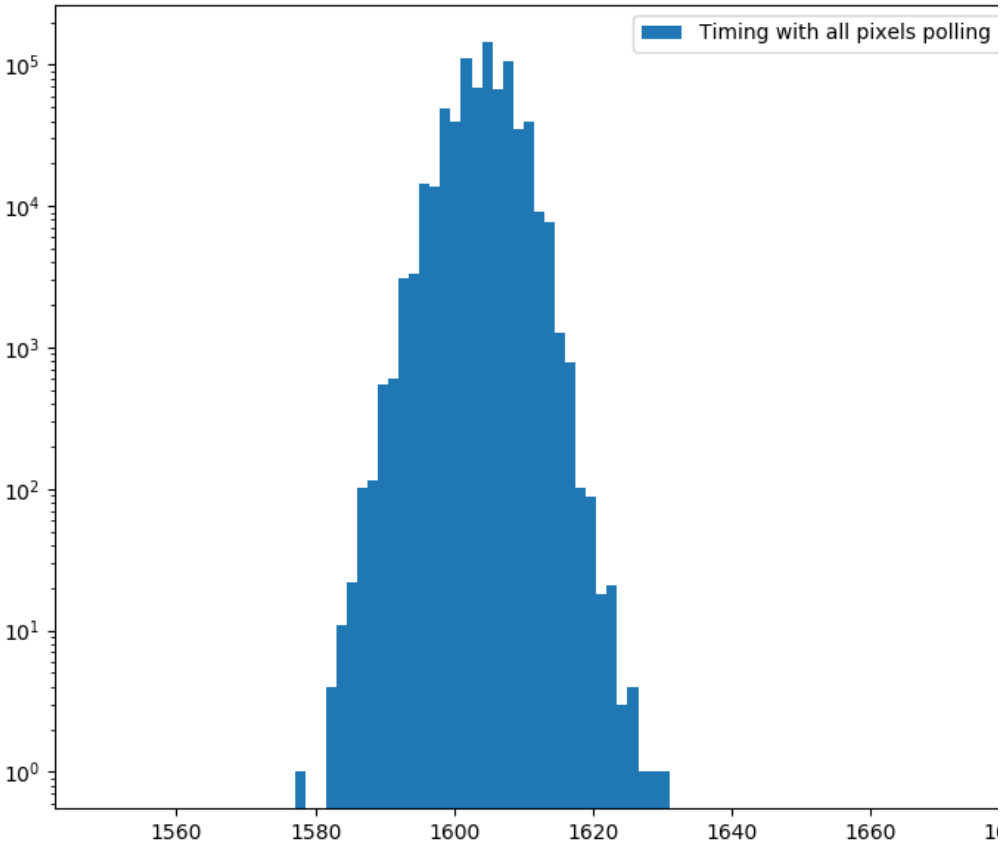
SCAO case

RTC configuration:

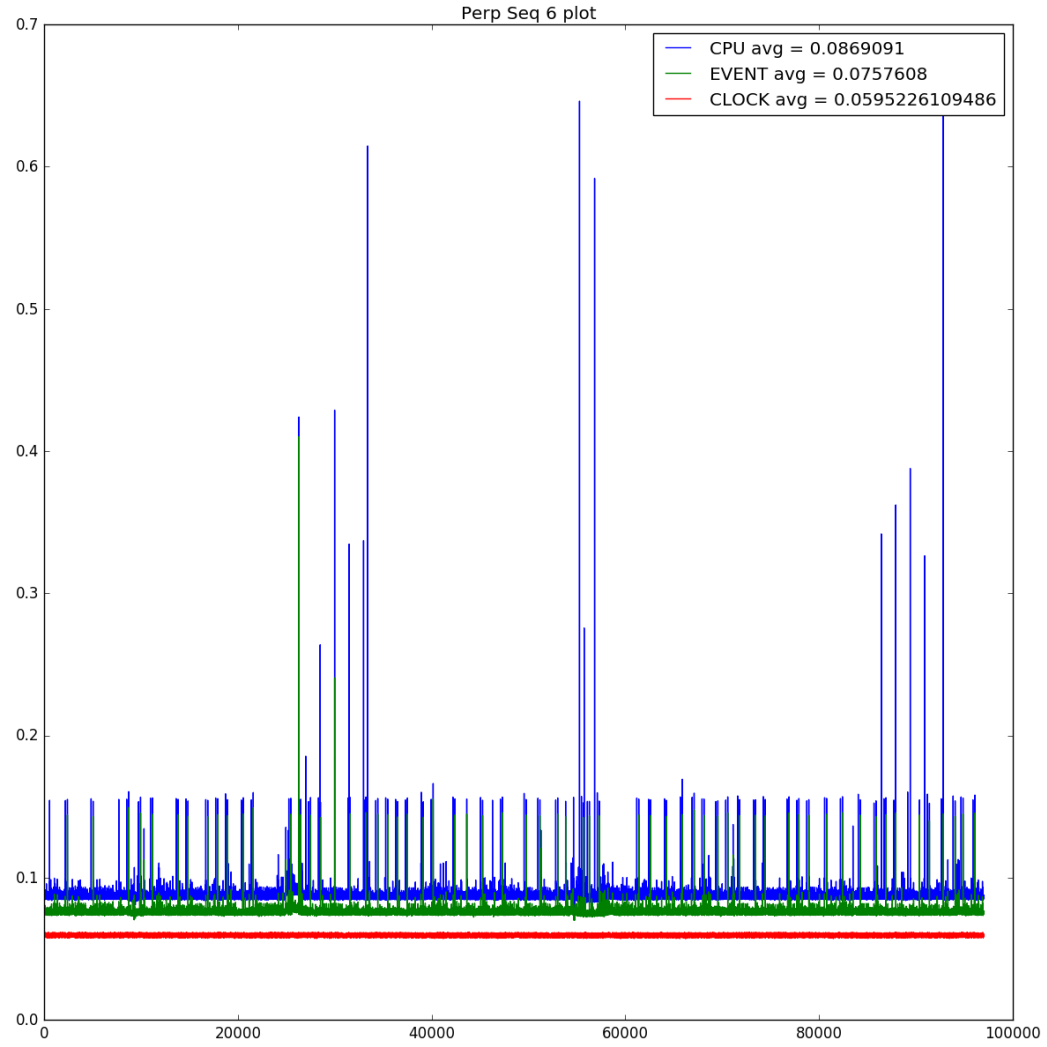
- Perform on a Quadro K6000
- Camera : 262k px @ 1024 fps
- Slope : 5000 (eq 70x70)
- Command : 2048
- 700k iterations

- Results :
  - Avg : 1604 µs
  - Peak to peak jitter : 52 µs

✔ Computations validated with COMPASS



Time measurement strategies 1/2



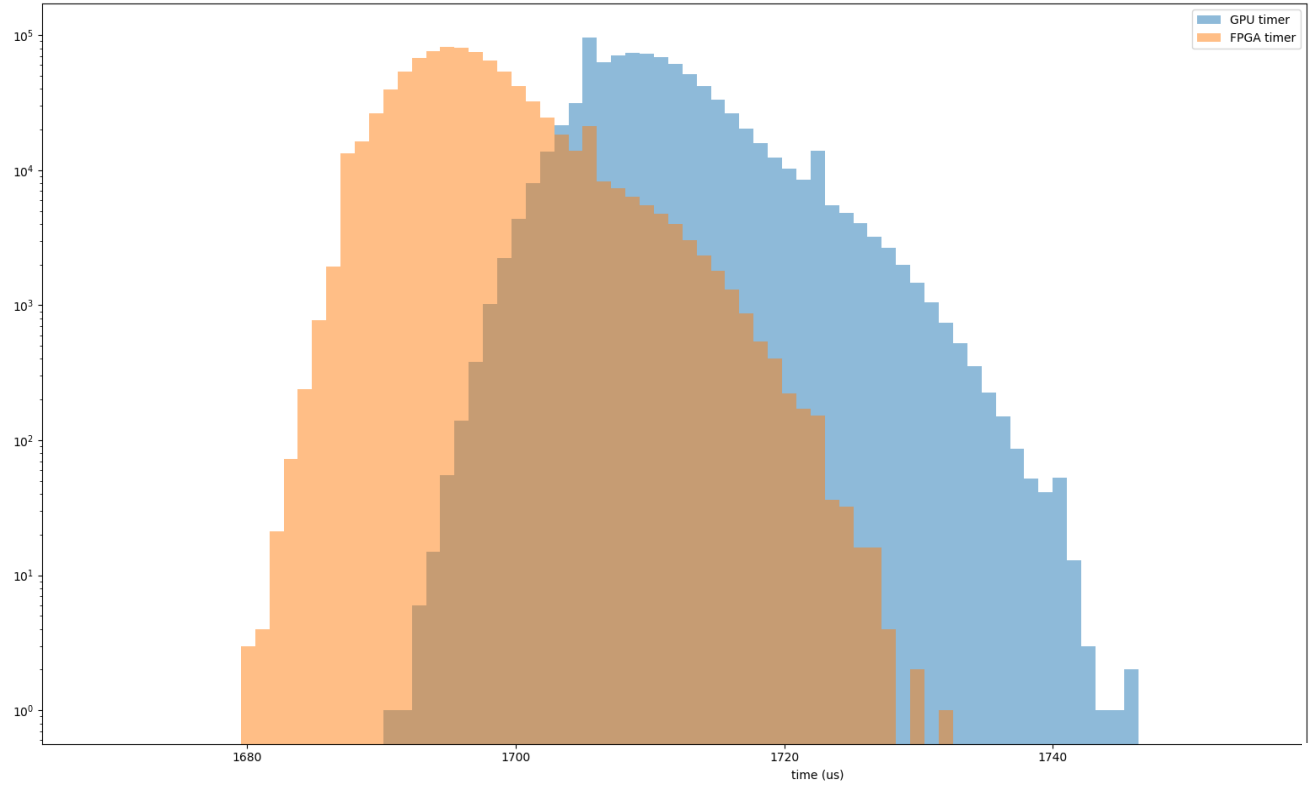
3 way measurements

- CUDA event
- C++ std high_resolution_clock
- CUDA clock64

Event & C++ clock show jitter due to CPU utilization

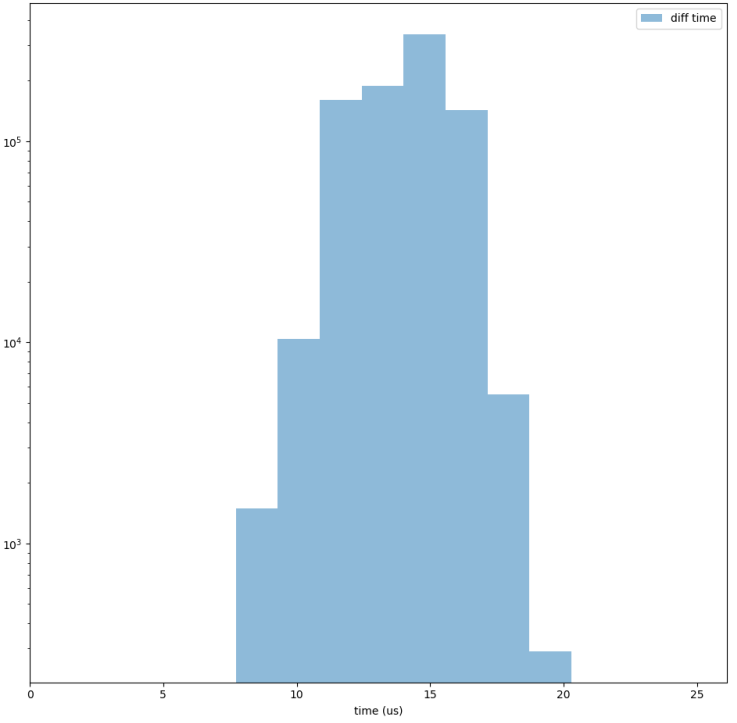
CUDA clock only depend on GPU execution

Time measurement strategies 2/2



Compared to FPGA timing, clock timing is still relevant:

- Doesn't hide jitter
- Small error: 20µs max + FPGA overhead

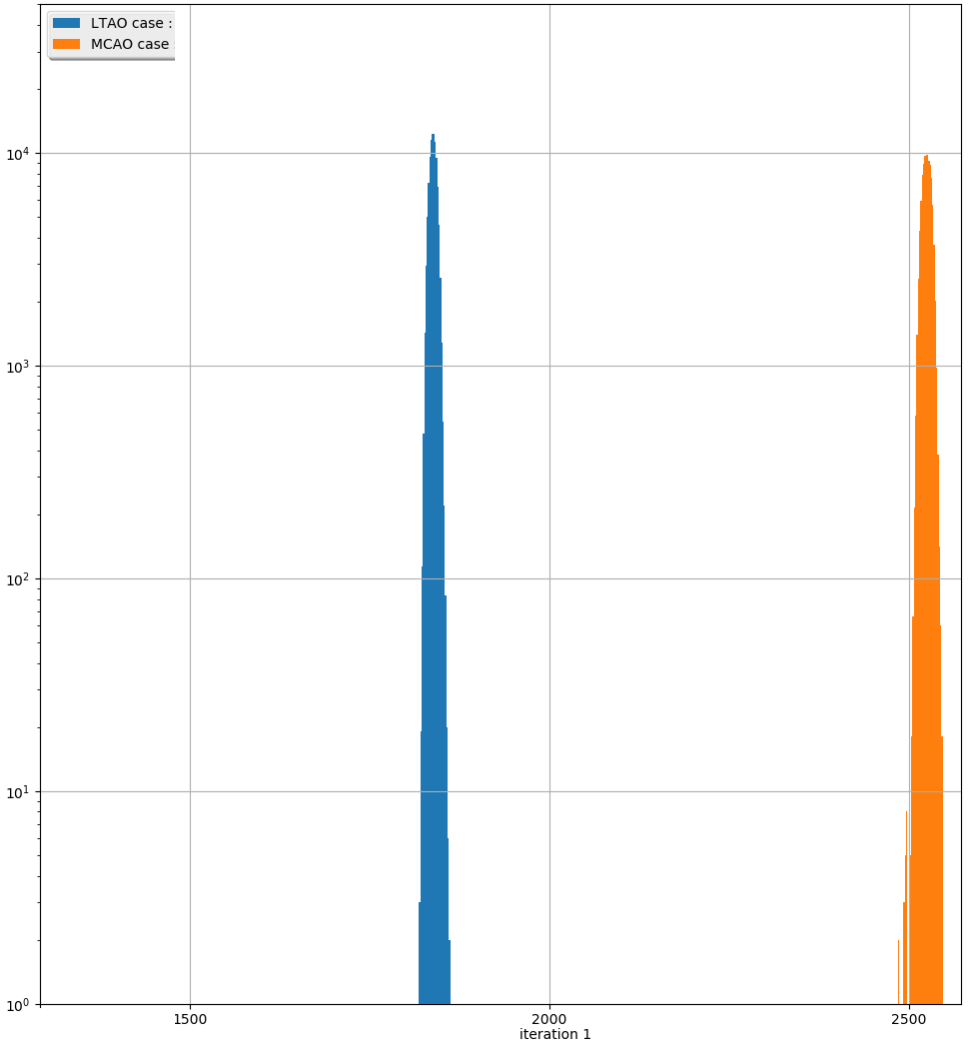


LTAO & MCAO

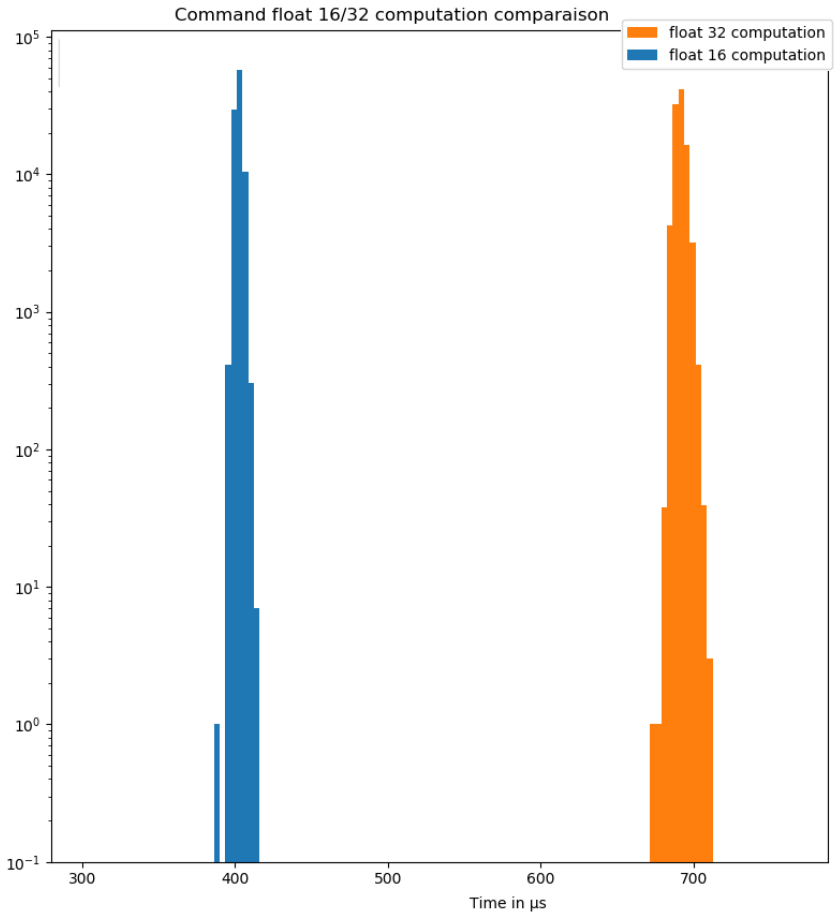
RTC configuration :

- LTAO
  - 6 wfs x 10k slopes
  - 5k commands
  - 4 devices

- MCAO
  - 6 wfs x 10k slopes
  - 15k commands
  - 8 devices



Float vs half (float 16)



Matrix vector computation using IEEE 754-2008 16-bit floating point

- Need to use vectorized half: half2 for performance
  - Each thread compute element two by two
  - CUDA support all operations
    - Simple arithmetic
    - FMA (Multiply-add)
    - Math functions (exp, log, trigo.)

Conclusion

- Using GPU direct & Persistent kernel: avoid all CPU constraints
  - No shielding
  - No real time OS
- Our solution is real time by design

l'Observatoire de Paris — LESIA
Laboratoire d'Études Spatiales et d'Instrumentation en Astrophysique

Durham University

MICROGATE

PLDA

Thank you

Question ?

European Commission

Project #671662 funded by European Commission under program H2020-EU.1.2.2 coordinated in H2020-FETHPC-2014