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

Durham University

MICRO GATE

PLDA

RTC4AO 2016

Green Flash

A Real Time Controller For E-ELT

Julien BERNARD

European Commission

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

Green Flash

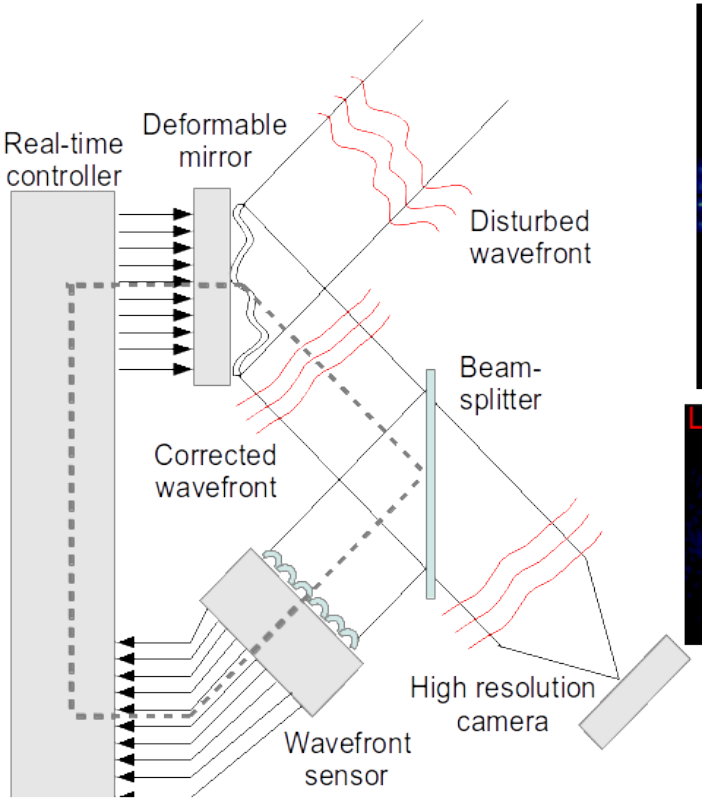
- Public and private actors
  - Paris Observatory
  - University of Durham
  - Microgate
  - PLDA

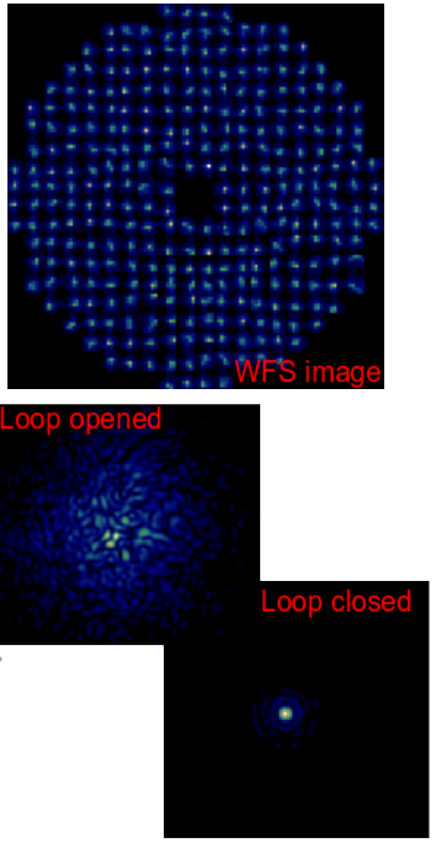
- Part of Horizon 2020 : EU Research and Innovation programme
- 3 years project
- 3,8 million €
- Involve about 30 people

- Research axes
  - Real time HPC with accelerators and smart interconnects
  - Energy efficient platform based on FPGA
  - Real Time Controller (RTC) prototype for European – Extremely Large Telescope Adaptive Optics (AO) system

Adaptive Optics

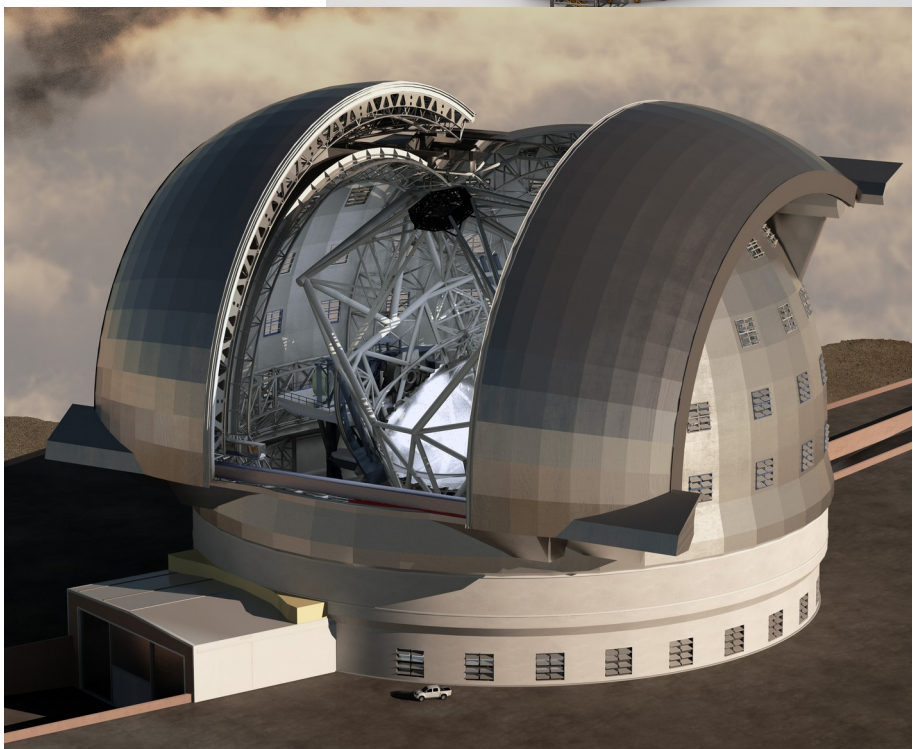
- Compensate in real-time the wavefront perturbations
- Using a wavefront sensor - WFS to measure them
- Using a deformable mirror – DM to reshape the wavefront
- Commands to the mirror must be computed in real-time (~ms rate)





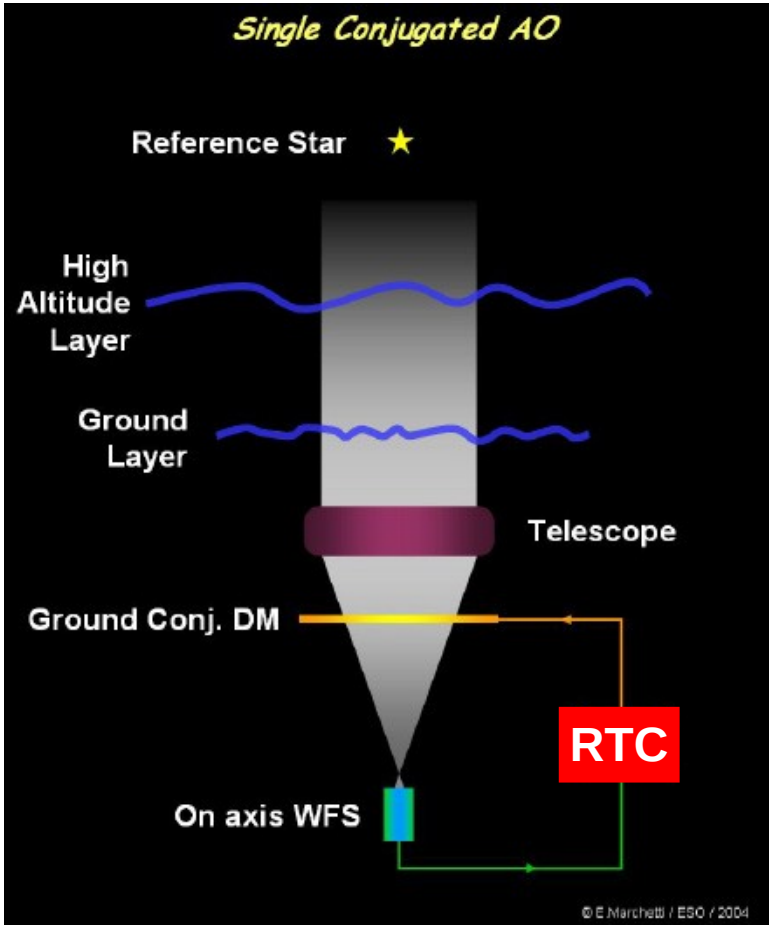
RTC AO prototype for E-ELT

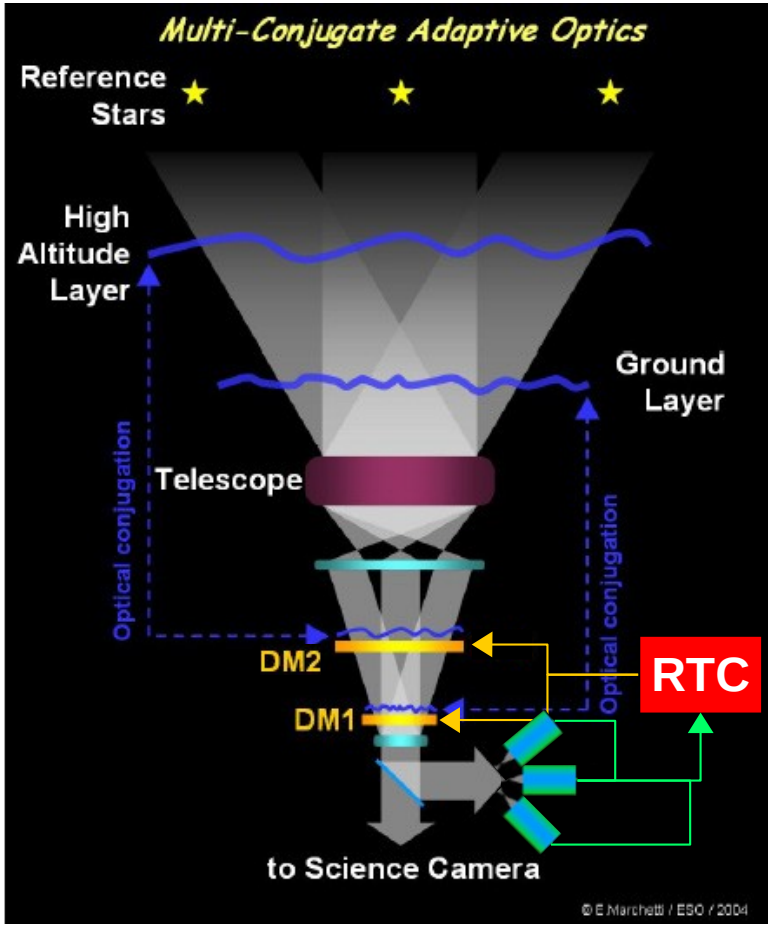
- European – Extremely Large Telescope
  - Next telescope generation
  - 39 m diameter telescope
  - 100m dome, 2800 tones structure rotating at 360°, seismic safe (Chile)
  - first light foreseen in 2024
  - European project led by ESO funded by 15 European countries
  - 1.2 billion € project



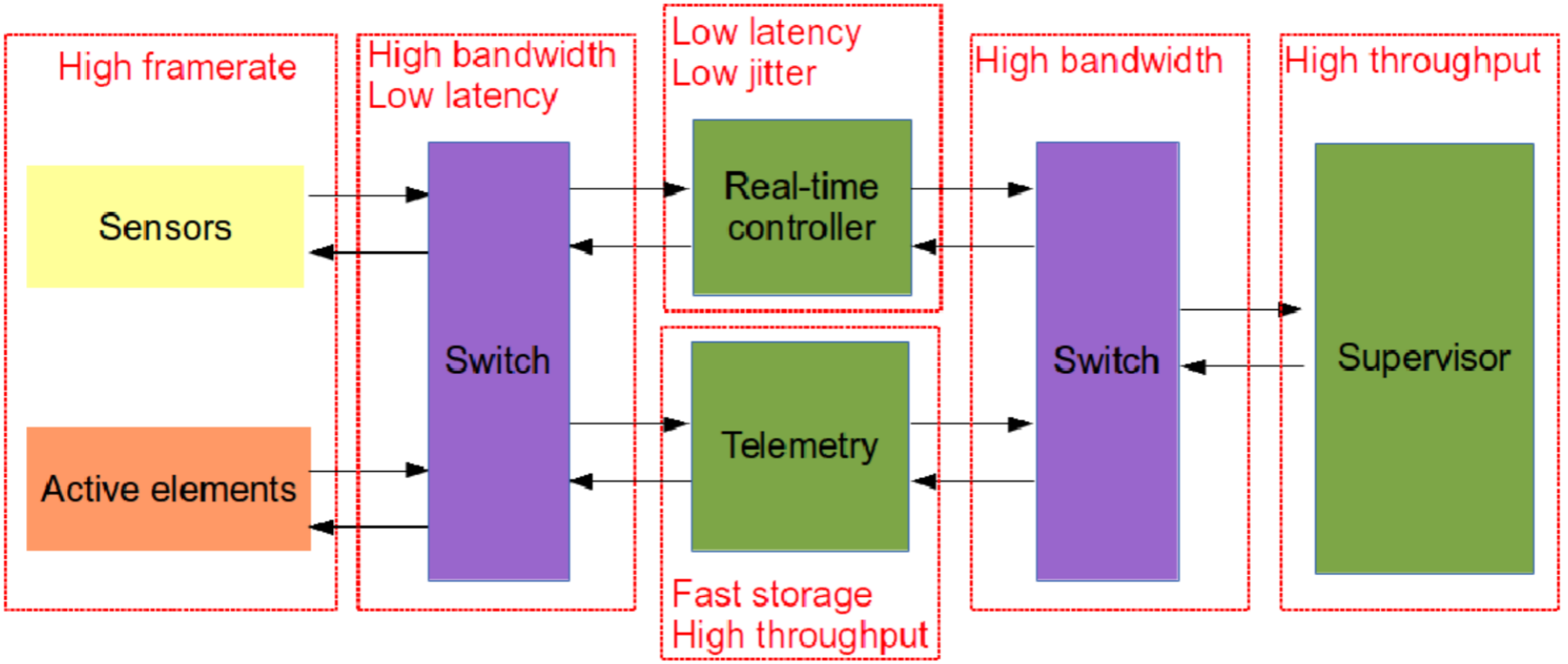
SCAO / MCAO

	Simple Conjugate Adaptive Optics	Multiple Conjugate Adaptive Optics
Wave Front Sensor	1 (10k measures)	6 (60k measures)
Deformable mirror	1 (5.3k degree of freedom)	3 (16k degree of freedom)
Frequency	1kHz	500Hz





RTC concept for ELT AO



Computation : MCAO

- MCAO process
  1) Calculate the pseudo-open loop measurement vector with the two last command vector and the interaction matrix

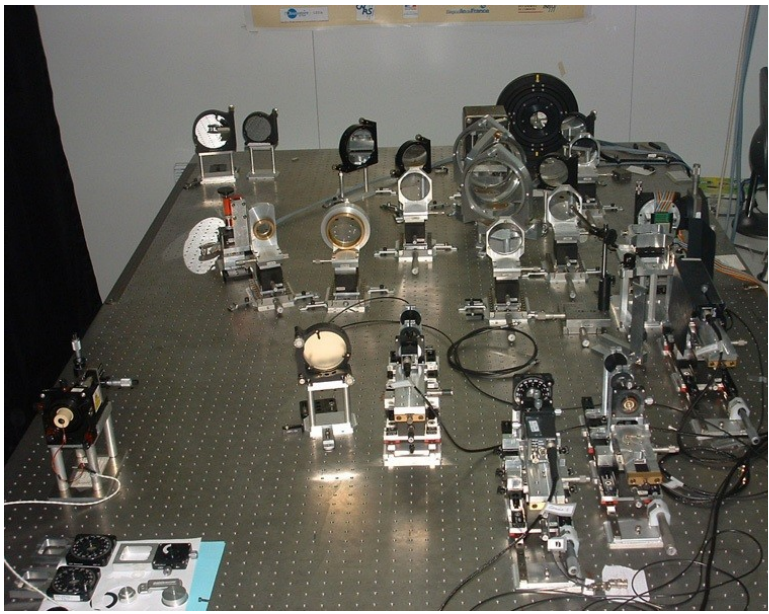
$$\vec{M}_{ol}[k] = \vec{M}[k] - D\left(a\,\vec{c}[k-2] + (1-a)\,\vec{c}[k-1]\right)$$

  2) Calculate the raw tomographic vector

$$\vec{e}[k] = R\,\vec{M}_{o1}[k]$$

  3) Get the vector command by smoothing the raw vector with the last command

$$\vec{c}[k] = g\,\vec{e} + (1-g)\,\vec{c}[k-1]$$



- Sizing :
  - SCAO : 1 matrix x 10 048 x 5316 = 53 MMAC per iteration and **53 GMACS @ 1kHz**
  - MCAO : 2 matrix x 60288 x 15000 = 1.8GMAC per iteration and **1 TMACS @ 500 Hz**

GPU sizing

- Matrix Vector Multiply – MVM take more than 90% of the time
- I/O bound computation
- Depends of the sustained bandwidth of the device

	K20C	K40	K80	P100
B_{theo}	208	288	240 (x2)	732
$B_{no\ ECC}$	175 (84%)	236 (82%)	200 (x2, 83%)	460 (62%)
B_{ECC}	150 (72%)	208 (72%)	173 (x2, 72%)	460 (62%)

Bandwidth of GPU (theoretical, real without ECC, with ECC) in giga byte per second

$$CEIL\left(\frac{CR}{\widetilde{B}\ x\ OI}\right) = N$$

OI (Operation Intensity) : number of MACS by quantity of non cached data byte to compute the MACS

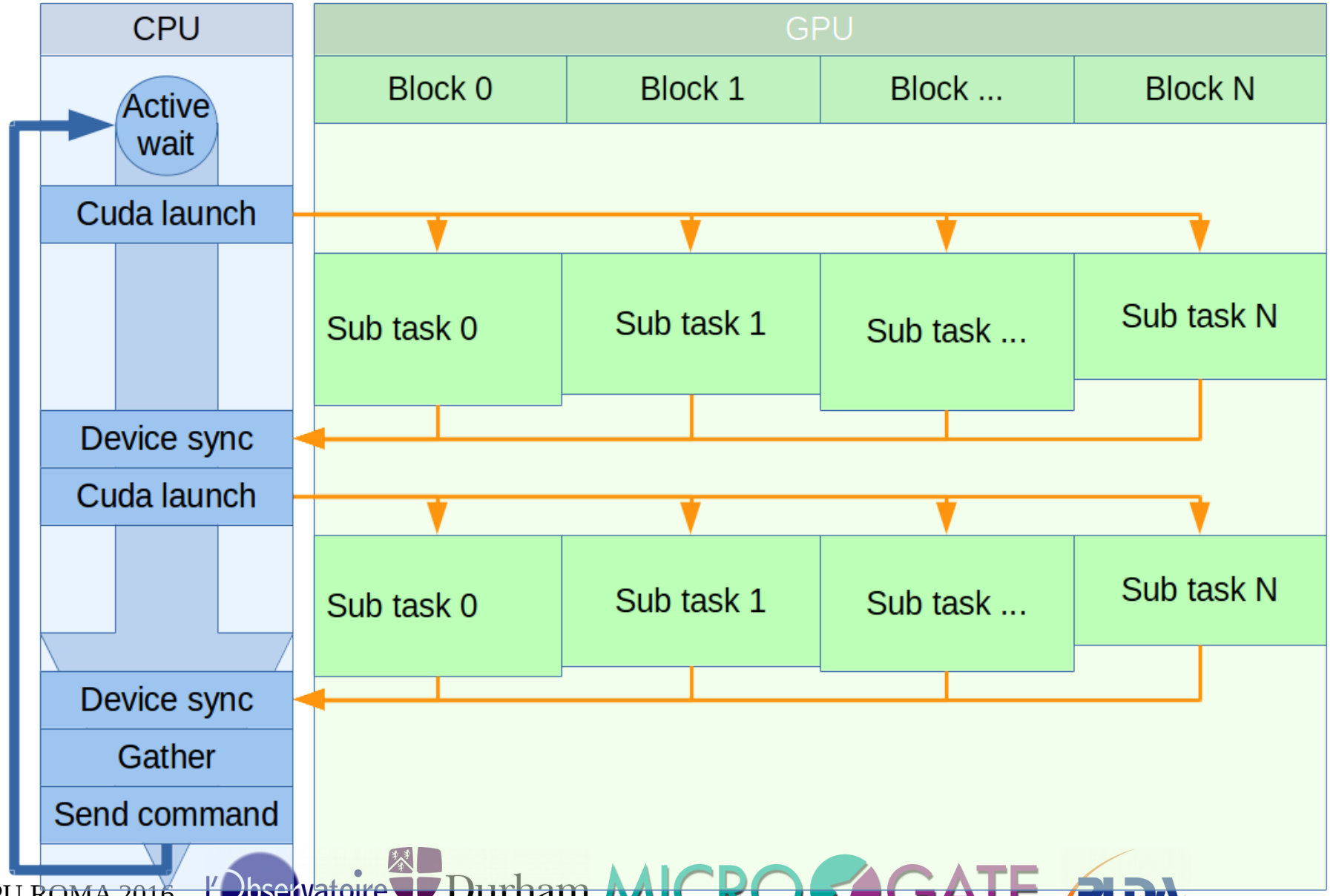
I.E. GEMV processing in :

- f32 $\frac{2n^2+2n}{4(n^2+3n)} \approx 0.50$
- f64 $\frac{2n^2+2n}{8(n^2+3n)} \approx 0.25$

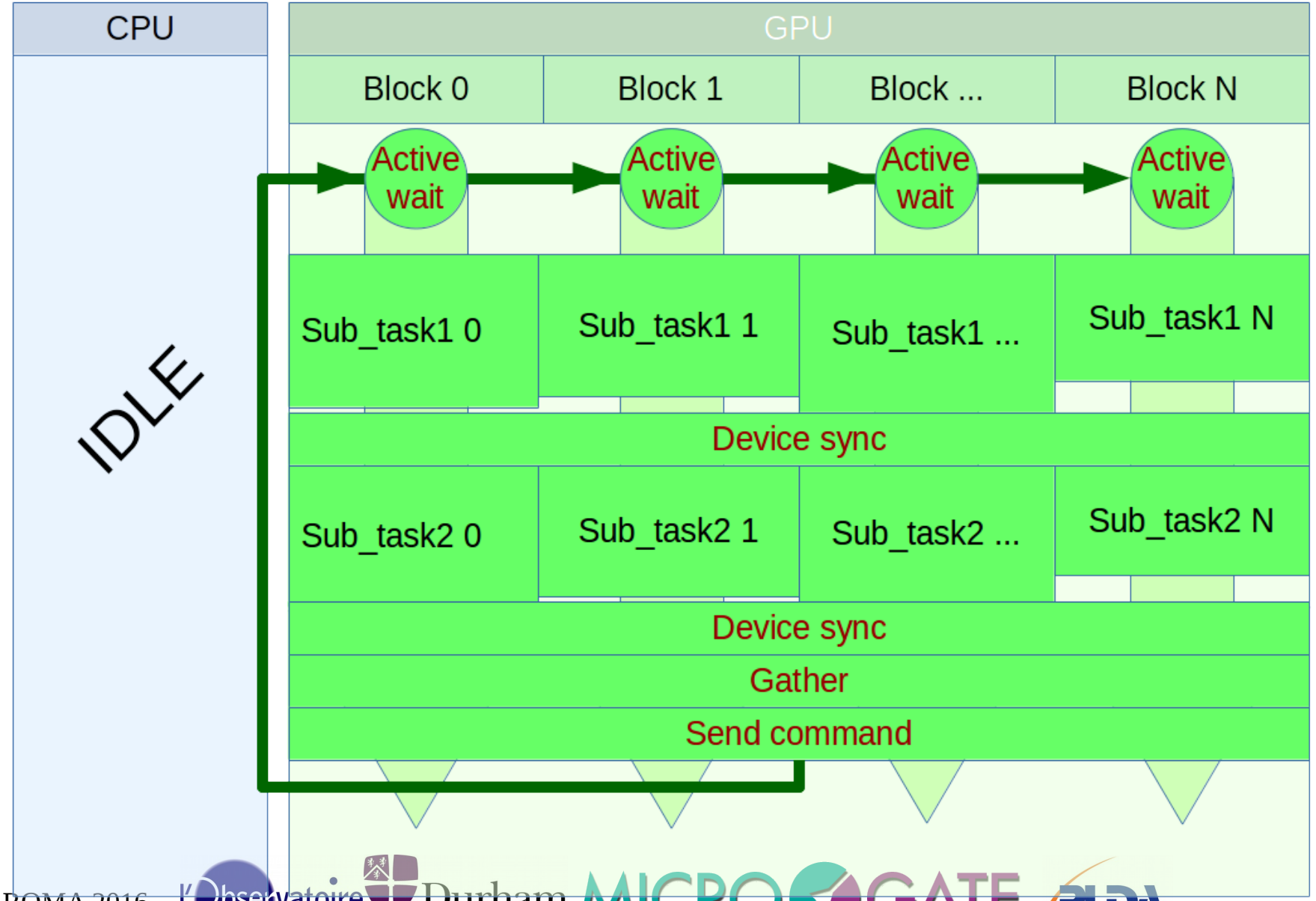
Number of GPU needed for the MCAO case computed in f32

ECC	K20C	K40	K80	P100
Off	12	9	6	5
On	14	10	6	5

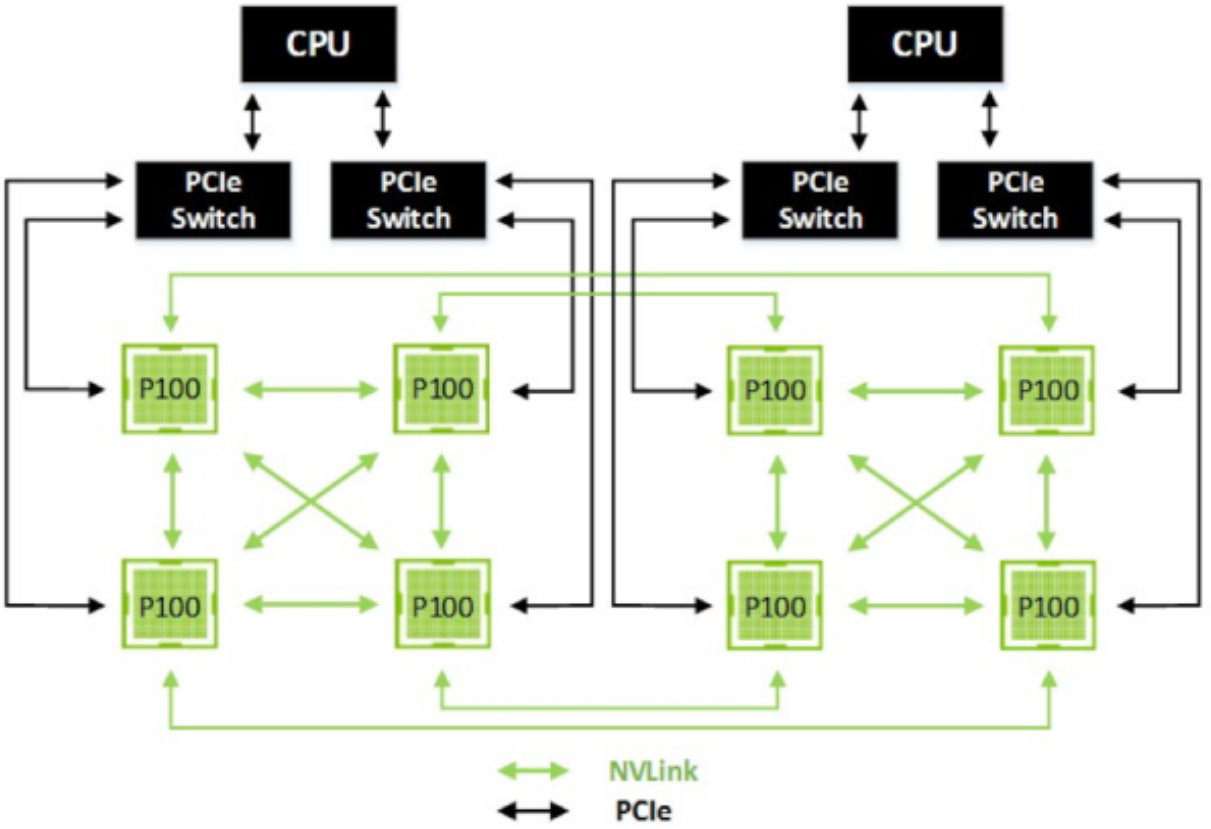
Classic implementation



Persistent kernel implementation



Hardware



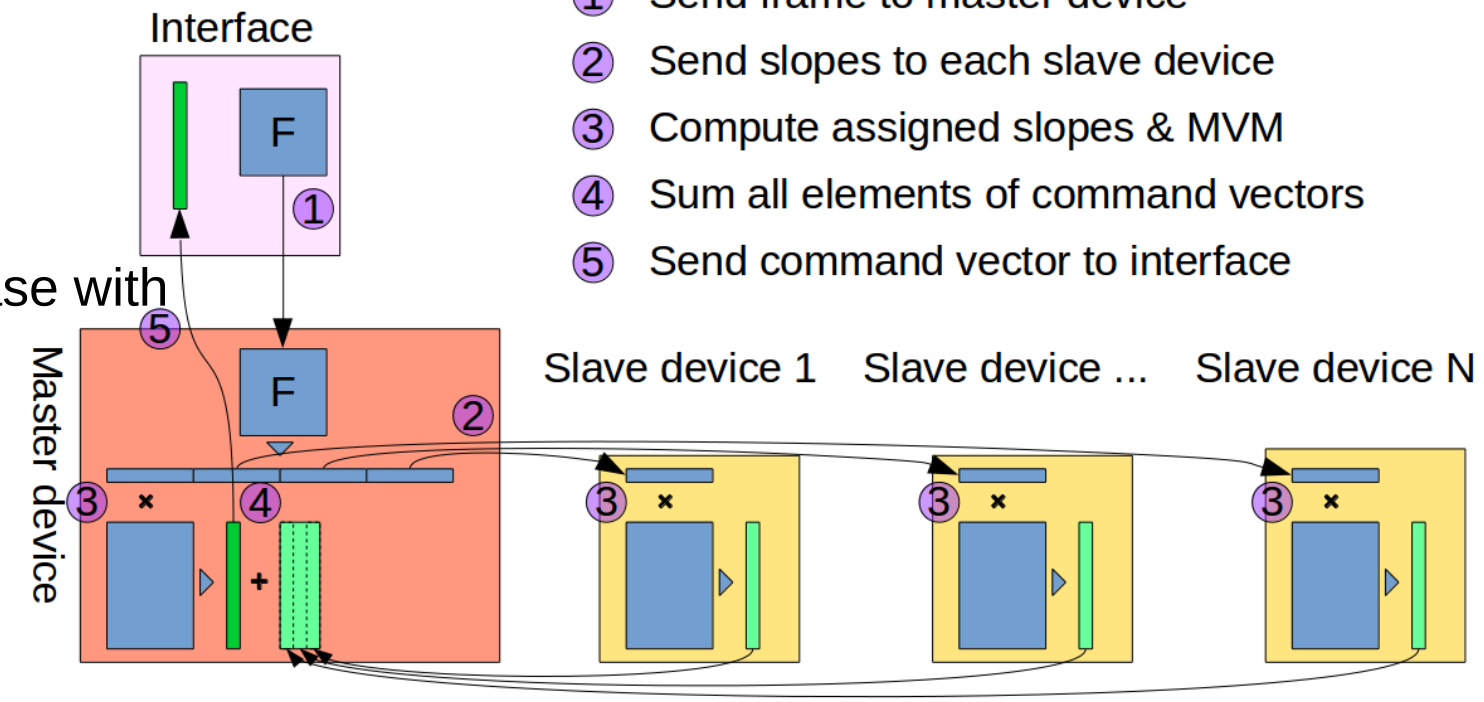
DGX-1 server

- Ubuntu 14.04
- Dual Intel Xeon E5-2698 @ 2.2 GHz
- 8 Nvidia P100 @ 1480:715 MHz
  - ECC disabled
  - CUDA 8.0

Test pipeline

Test on SCAO based case with

- 16 bit x 512² pixel frame
- 2 x 5024 slopes
- 15k commands
- 1-4 GPU for computing

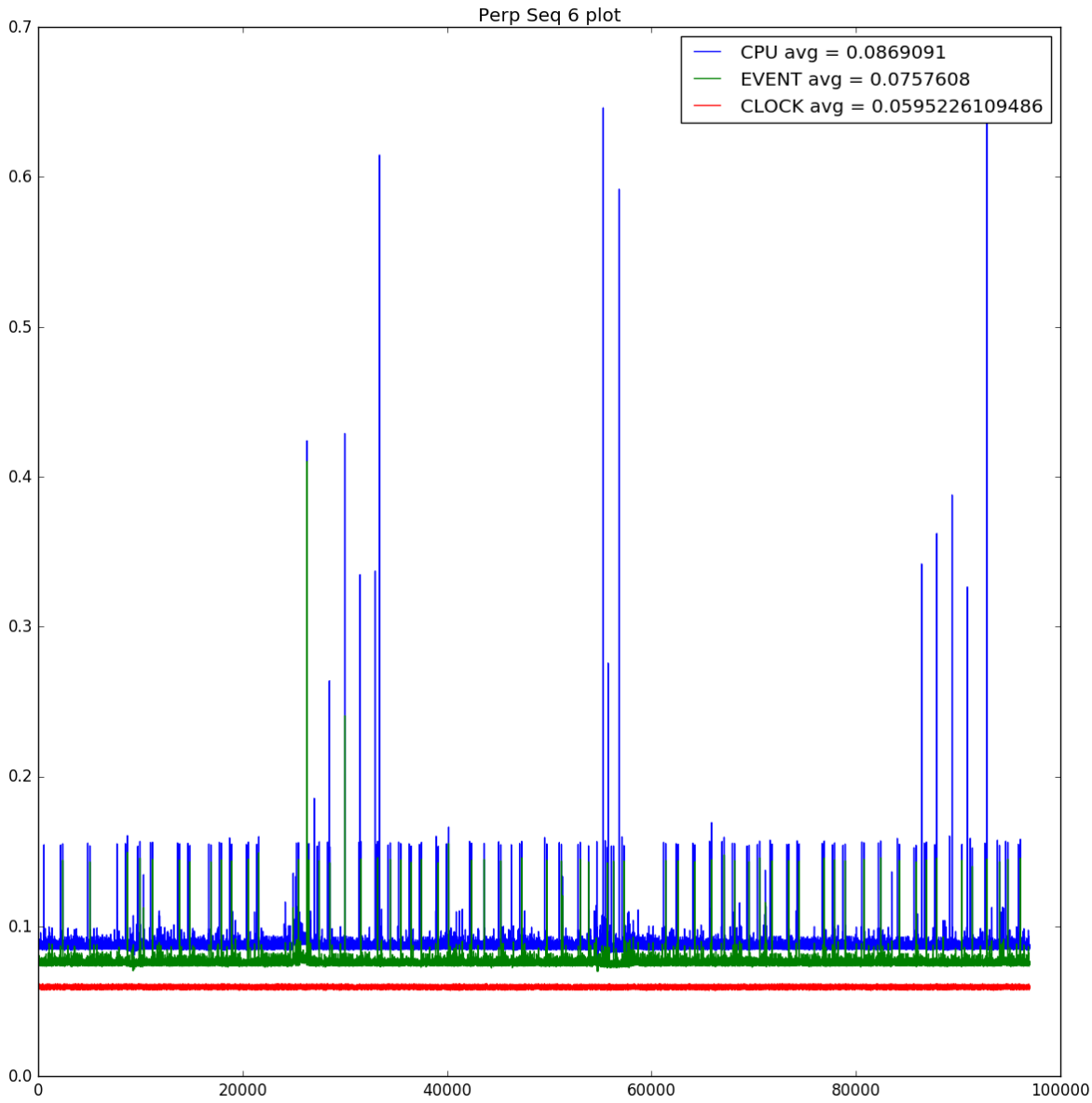


1. Send frame to master device
2. Send slopes to each slave device
3. Compute assigned slopes & MVM
4. Sum all elements of command vectors
5. Send command vector to interface

Time measurement strategies

3 way measurements

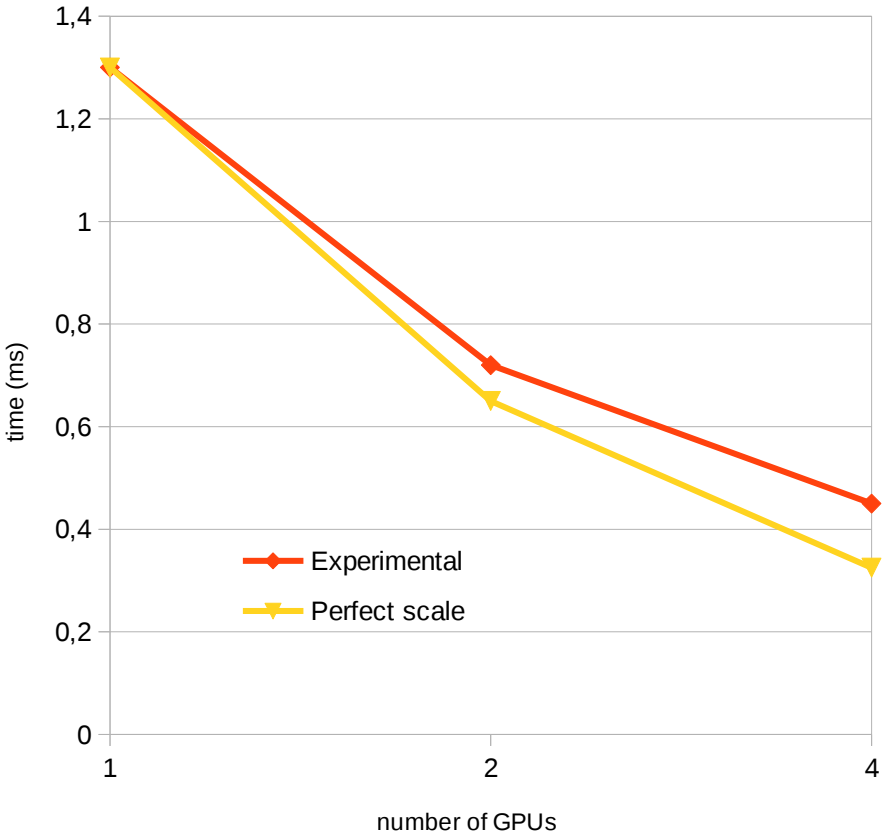
- CUDA event
- C++ std high_resolution_clock
- CUDA clock64



Result 1/2 : Scalability and jitter

Strong scalability

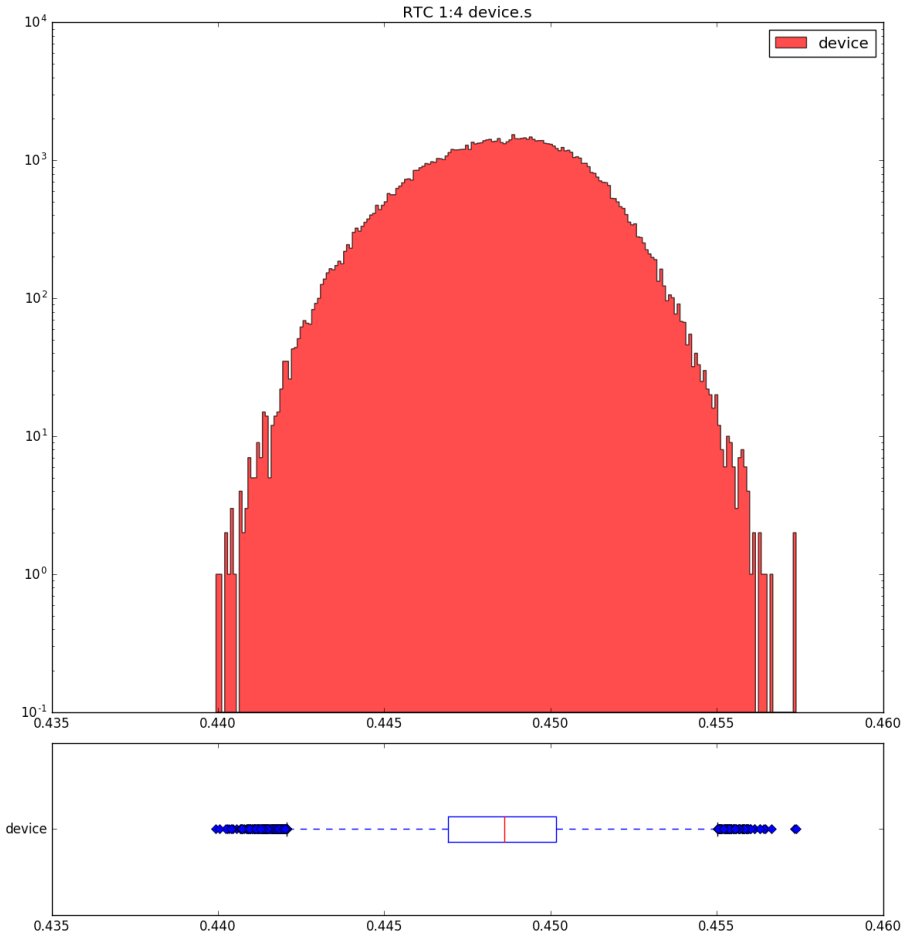
Constant case with **10,048 slopes x 15,000 commands**



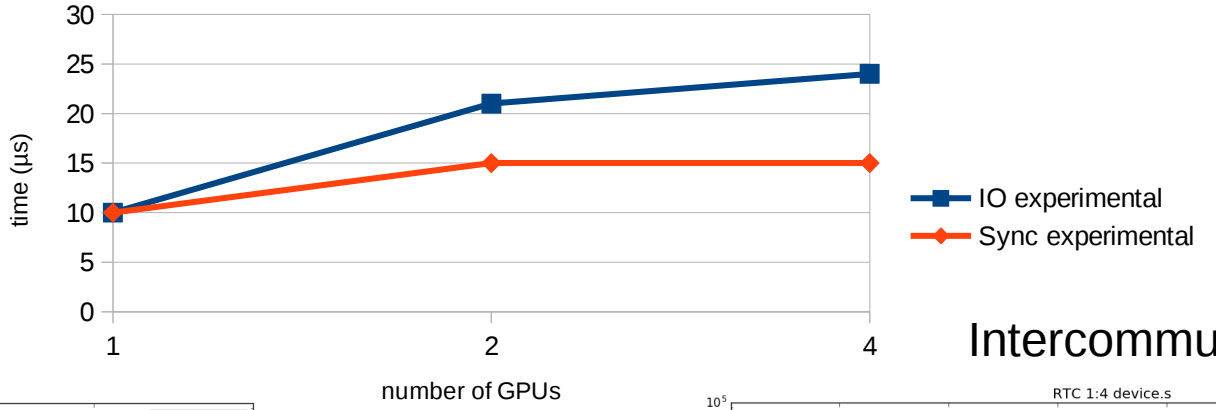
Histogram

Case with **10,048 slopes x 15,000 commands** on 4 devices

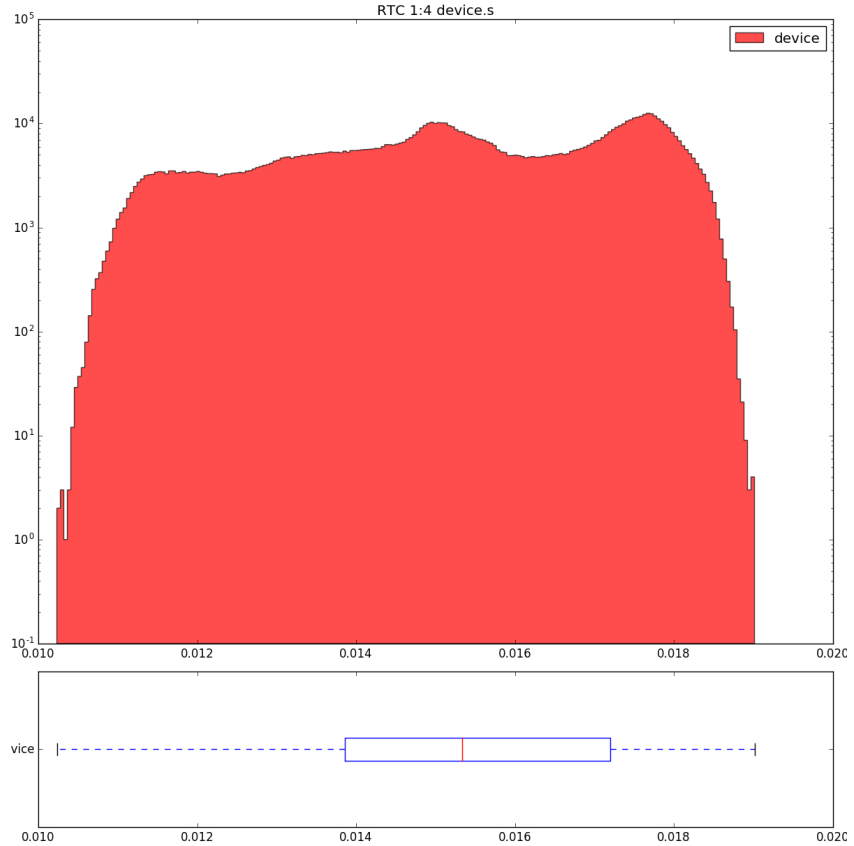
Average : 0.45ms Jitter : 17µs



Result 2/2 : Sync & Intercom time

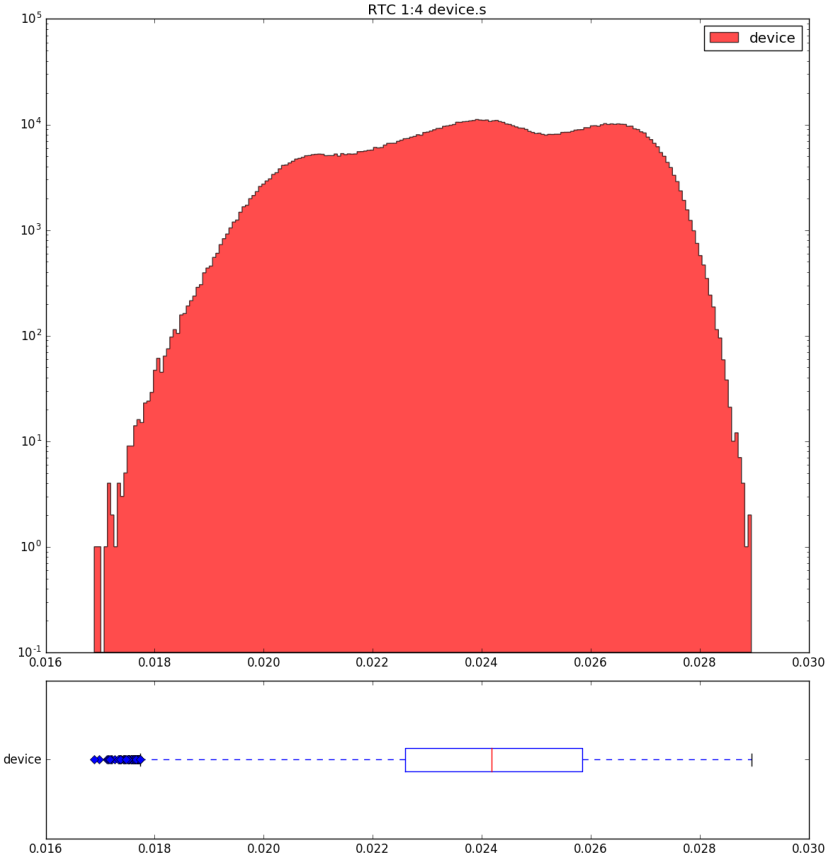


Synchronize jitter



Average : 15µs Jitter : 8.8µs

Intercommunication jitter



Average : 24µs Jitter : 12µs

Conclusion : Persistent kernel

- Pros
  - Much lower jitter
  - Simpler execution stream
- Cons
  - Beware of kernel mapping
  - No concurrency
  - Hard to debug
  - Prohibited features
    - No cuda library using kernel (i.e. cuBlas)
    - No cuda runtime synchronization
  - Useless features
    - stream, memcpy
  - No intercommunication through QPI

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