**Third AO4ELT Conference - Adaptive Optics for Extremely Large Telescopes**
Florence, Italy. May 2013
ISBN: 978-88-908876-0-4
DOI: 10.12839/AO4ELT3.13354

# Building a reliable, scalable and affordable RTC for AO instruments on ELTs

Damien Gratadour[1a], Arnaud Sevin[1], Denis Perret[1], and Julien Brulé[1]

Laboratoire d'Etudes Spatiales et d'Instrumentation en Astrophysique (LESIA), Observatoire de Paris, CNRS, UPMC, Universite Paris Diderot, 5 places Jules Janssen, 92195 Meudon France[c]

**Abstract.** Addressing the unprecedented amount of computing power needed by the ELTs AO instruments real-time controllers (RTC) is one of the key technological developments required for the design of the next generation AO systems. Throughput oriented architectures such as GPUs, providing orders of magnitude greater computational performance than high-end CPUs, have recently appeared as attractive and economically viable candidates since the fast emergence of devices capable of general purpose computing. However, using for real-time applications a I/0 device which cannot be scheduled nor controlled internally by the operating system but is sent commands through a closed source driver comes with a number of challenges. Building on the experience of almost real-time end-to-end simulations using GPUs, and relying on the development of the COMPASS platform, a unified and optimized framework for AO simulations and real-time control, our team has engaged into the development of a scalable, heterogeneous GPU-based prototype for an AO RTC. In this paper, we review the main challenges arising when utilizing GPUs in real-time systems for AO and rank them in terms of impact significance and available solutions. We present our strategy, to mitigate these issues including the general architecture of our prototype, the real-time core and additional dedicated components for data acquisition and distribution. Finally, we discuss the expected performance in terms of latency and jitter on the basis of realistic benchmarks and focusing on the dimensioning of the MICADO AO module RTC.

## 1 Introduction

Several key technological developments for the E-ELT instrumentation have been identified during the initial design studies as top priorities for the European astronomical community. Among these, several are related to AO and require realistic numerical simulations to be studied at full scale. Additionally, the foreseen AO systems will require an unprecedented amount of computing power to be driven in real-time. Accelerator based architectures using GPUs are very attractive solutions to provide the required power. Moreover, the emergence of GPGPU programming is, for the first time, proving a mean of addressing both simulations and real-time control developments on a unified architecture.

Our team is engaged into the COMPASS project[1] aiming at providing a full scale end-to-end AO development platform, able to address the E-ELT scale and including a real-time core that can be directly integrated on a real system. The development of this platform is based on a full integration of software with hardware and will rely on an optimized implementation on heterogeneous hardware using GPUs as accelerators. The end product, a unified and optimized

[c] Groupement d'Interet Scientifique PHASE (Partenariat Haute resolution Angulaire Sol Espace) between ONERA, Observatoire de Paris, CNRS, Universite Paris Diderot, IPAG and LAM

[a] damien.gratadour@obspm.fr

[1] This work is supported by the ANR grant ANR-12-MONU-0022 of the French Ministry of Research

computing framework (software and hardware), will address several major needs for research in AO, as it will provide:

- An efficient computing environment to run large scale AO systems simulations in almost-real-time in order to test the design of the E-ELT instrumentation and prepare the scientific return of the foreseen instruments
- A real-time core, integrated in the simulation environment, fully expandable and capable of driving a real system at the appropriate speed using various command /control algorithms depending on the system (SCAO, MCAO, MOAO)
- A high bandwidth low latency interface between such a real-time core and a standard interconnect such as 40 GbE
- A pathfinder for accelerator-based AO control under the form of a prototype for a low latency image acquisition and processing system including the real-time core and its interface to a camera

In this paper, we review the main challenges arising when utilizing GPUs in real-time systems for AO and rank them in terms of impact significance and available solutions. We present our strategy for unifying efficient simulations and real-time control frameworks, for prototyping an accelerator based RTC and for the development of the appropriate interconnect.

## 2 Fast AO simulations with GPUs

The angular stone of the design studies of a complex system is the numerical simulation. It allows to validate the conceptual design and to test the behavior of the various system components under realistic conditions. OCTOPUS [4] is a fully parallel simulation code designed to run on a cluster of CPUs. It relies on a large and costly infrastructure to provide appropriate execution times at the scale of the E-ELT.

Our team as developed a GPU-based approach codenamed YoGA_AO[2] able to address the scale of the E-ELT and providing quasi-real-time performance for the simulation of a Shack-Hartmann based SCAO system on a single GPU. We recall in the table below the simulation profiles in ms for the various parts of the AO system physical model and various telescope diameters. A comparison of the performance with the sequential CPU code YAO, written by F. Rigaut[5] shows a speedup of 7 on the global execution[1] time and even greater on some parts of the code such as control.

**Table 1.** Simulation profiles in ms obtained on a NVIDIA Tesla M2090.

| Tel. diam. | Turbulence | Ray-tracing | WFS | COG | Control | DM |
|---|---|---|---|---|---|---|
| 4m | 0.107 | 0.008 | 0.138 | 0.013 | 0.019 | 0.137 |
| 8m | 0.192 | 0.022 | 0.459 | 0.031 | 0.060 | 0.562 |
| 20m | 0.550 | 0.135 | 3.07 | 0.079 | 0.363 | 3.22 |
| 30m | 0.927 | 0.299 | 6.73 | 0.168 | 0.915 | 7.39 |
| 40m | 1.44 | 0.526 | 11.9 | 0.320 | 2.263 | 13.62 |

Column 6 and 7 of table 1 correspond to the task a RTC should execute in a real system and the performance obtained are compatible for control at about 1000 frames per second on a

30m telescope and 400 frames per second on the E-ELT. Moreover, the code relies on custom scalable kernels for centroiding and standard BLAS libraries for control. Performance on this part of the code is thus likely to increase with hardware developments in the coming years.

## 3 Unified framework for simulations and real-time control

AO is a versatile technique with many possible system designs for different applications, from the simplest classical AO system with a single WFS and a single DM operated in closed-loop to the multi-object AO system with multiple WFSs and DMs providing tomographic correction in an opened-loop fashion.

While a significant amount of work has already been done to port key physical models for the simulation of AO systems on parallel platforms, such as atmospheric turbulence generation or Shack-Hartmann sensor image formation; one of the main remaining technical challenges is porting the various system control algorithms on such platforms. This core component of any AO system is constantly evolving to provide the systems with better performance and more robust behavior, this increasing complexity being multiplexed, in the case of the E-ELT, by the dramatic increase in the number of degrees of freedom. We propose a novel and original approach in which the simulation models and the real-time core are developed in a unified framework so as to provide optimal simulation performance and full flexibility for the integration of complex control schemes in real systems. Such approach could provide a dramatic improvement in terms of robustness, development cost and updatability of AO real-time computers. The concept is presented in the figure below.

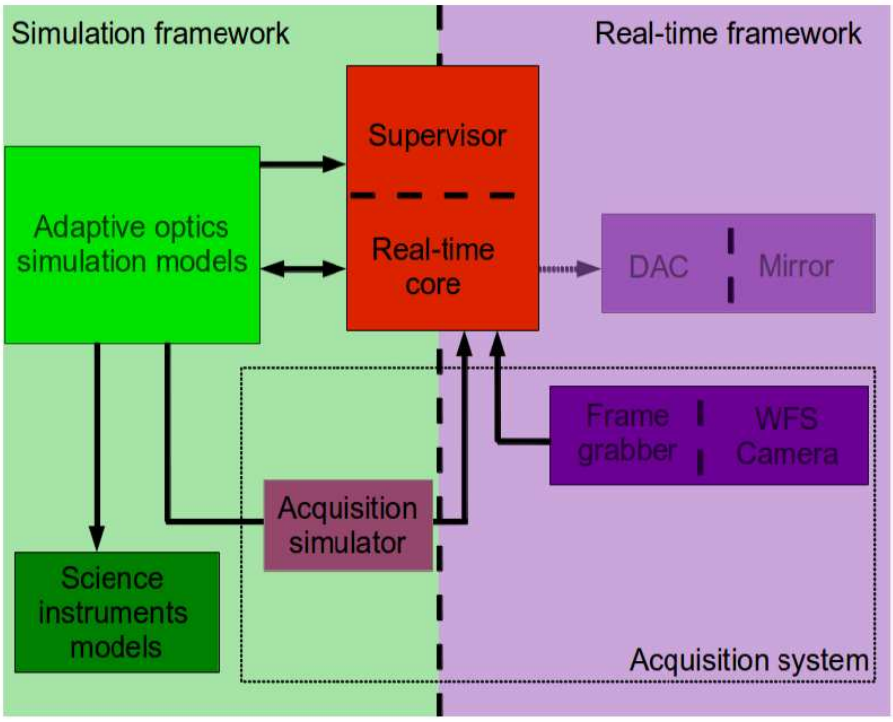


**Fig. 1.** The COMPASS framework unifying simulations and real-time control.

As demonstrated by this figure, the real-time core is a shared component between the simulation and real-time frameworks. In order to make the developments fully compatible, the data acquisition interface should be addressed in the simulation framework under the form of an acquisition simulator to provide the real-time core with a single entry point. Another benefit of such approach is the intrinsic compatibility between the real-time core and the simulation models allowing for the implementation of optimal supervision strategies.

## 4 RTC simulator

While accelerator-based architectures provide the perfect tools for simulations, the main technological bottle-neck for real-time applications is their external device / accelerator nature and intrinsically introduced latency in data transfers [3]. In the simulation framework, this latency can efficiently be hidden using compute-copies overlapping and unified memory addressing. In the real-time framework, such technique, to be efficiently applied, requires a full control over the data acquisition protocol.

We implemented a AO RTC simulator, based on simple assumptions and using the control core of the simulation code, in order to evaluate the potential performance of a GPU-based RTC under conditions consistent with real-time operations on a 40m telescope. The principle of the simulator has already been described in [2] and is recalled here. Pixel data residing on the system memory are asynchronously transferred to GPU memory using an optimized memory copy protocol and pixels values are used on the GPU to compute centroids. The obtained centroids residing on the GPU are then multiplied by the command matrix (preloaded on the GPU) to obtain the DM commands that are transferred synchronously to the system memory to ensure frame by frame synchronization. The profile of the overall process in the case of a typical dimensioning for the MICADO SCAO module on the E-ELT (80x80 Shack-Hartmann with 6x6 pixels per sub-apertures) is illustrated in Fig. 2.

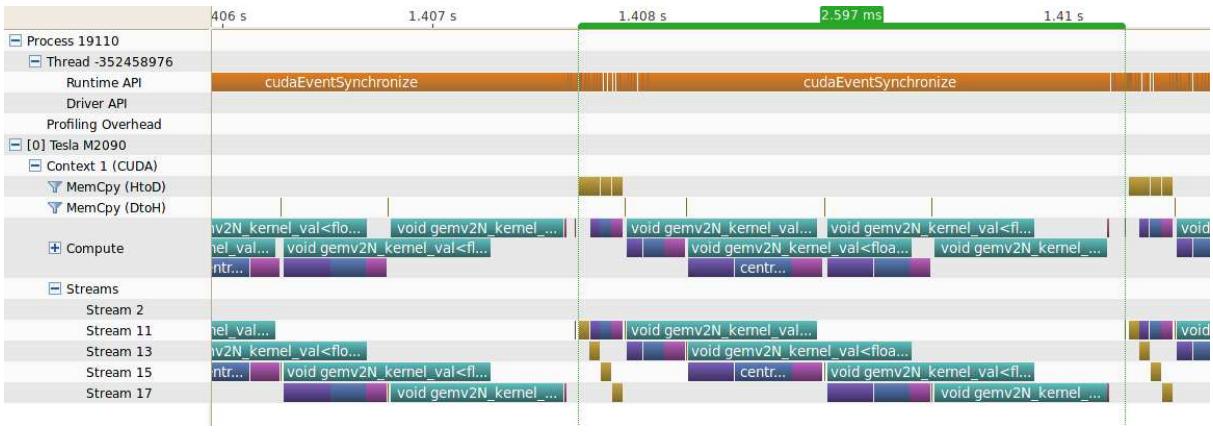


**Fig. 2.** Profile of the RTC simulator for a 80x80 Shack-hartmann system obtained on a single NVIDIA Tesla M2090. The dark yellow blocks represent data copies to a from the system memory, dark and light violet and dark blue represent the centroiding process and light blue represent the command vector computation. The profile was measured using NVidia Visual Profiler NVVP.

As shown in this figure, the various operations of this process are shared in blocks and sent to the GPU under the form of streams. This method allows for minimizing the latency introduced

by the pixels data copy to the GPU and for maximizing the GPU occupation by overlapping the reduction of several blocks. The obtained performance is thus optimized. This is made possible thanks to two features of CUDA, the NVidia framework for GPGPU computing:

– the GPU hosts several copy engines independent from the compute engine
– CUDA allows for the concurrent execution of several kernels on the same multi-processor

Another feature of the CUDA framework provides the ability to extend this approach to a multi-GPU configuration: unified virtual addressing (UVA) allowing several GPUs to share the same memory space. In this case, the sub-apertures are shared among the several GPUs and the command vectors obtained each contribute two the final results. An additional synchronization step between the GPUs is thus required to sum the various contributions.

The performance obtained for the MICADO SCAO dimensioning are illustrated in Fig. 3 for computer hosting respectively one and two NVidia Tesla M2090 and for different number of streams. The operating system used for this benchmark relies on a Linux kernel with real-time patches and corresponding patched NVidia driver and the master RTC simulator process on the CPU that sends instructions to the GPU has been shielded.

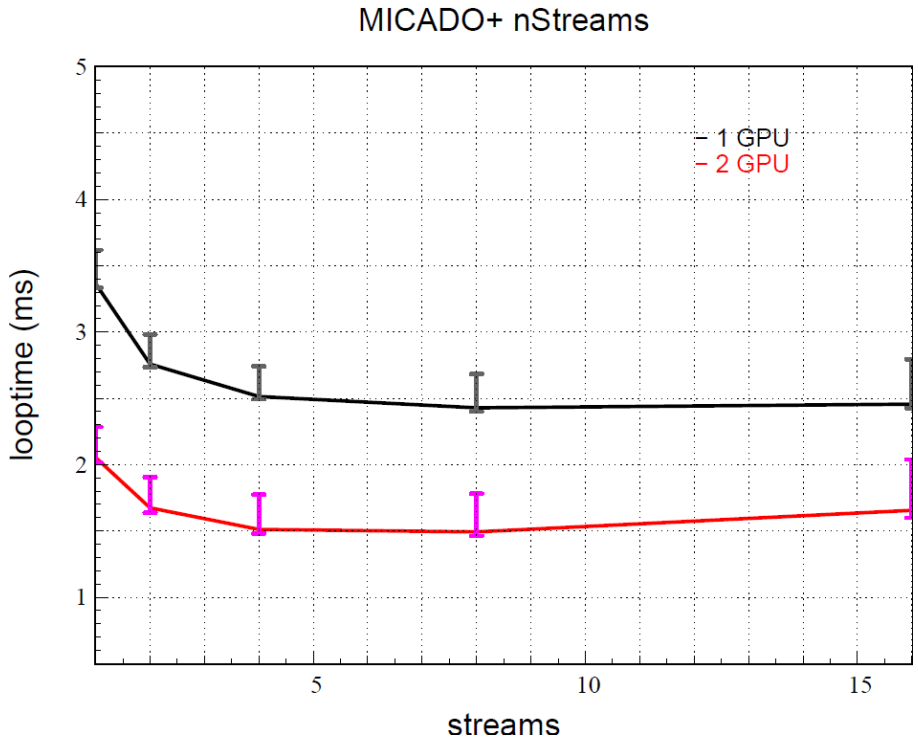


**Fig. 3.** Performance of the RTC simulator for a 80x80 Shack-Hartmann system and various number of streams on a single GPU system (black) and a dual GPU system (red). Jitter, represented as vertical bars, is estimated on $10^5$ iterations of the process.

In this figure, the vertical bars represents the jitter, as estimated using the largest and smallest execution times measured over $10^5$ iterations. An example of such a sequence is illustrated in Fig. 4 in the case of 4 streams.

This figure demonstrates the stability of the process, with the proposed system configuration: real-time Linux kernel and shielded master process, which is able, when using two GPUs, to

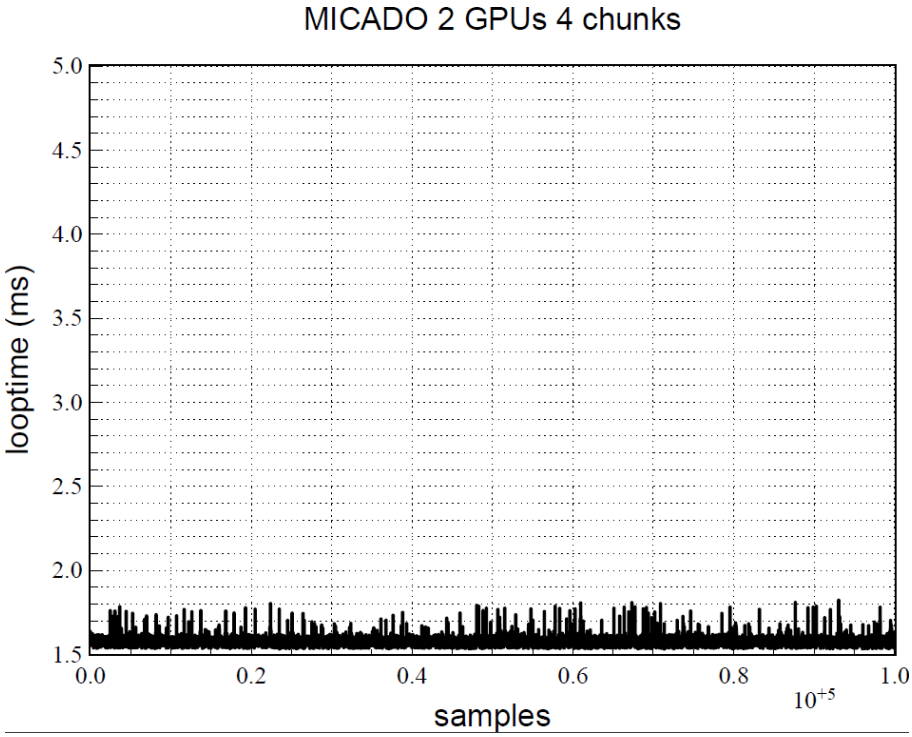


**Fig. 4.** $10^5$ iterations of the RTC simulator process for 4 streams. The system was running a Linux kernel with real-time patches and a shielded master process to send instructions to the GPU.

deliver about 400 updates per second to the deformable mirror with an overall latency of about 2ms.

## 5 Low latency data acquisition on GPUs

To drive turbulence compensation, the real-time core retrieves data from a camera with a finite readout speed using a frame grabber with limited bandwidth, computes the current error with a given throughput, and sends commands to a deformable mirror with a finite response time through another limited bandwidth protocol.

Optimizing the global throughput of this process, following our GPU-based approach, translates into minimizing the latency introduced by peer-to-GPU and GPU-to-peer transfers. To do so, it is required to implement a direct memory access (DMA) protocol over the interconnect between the camera and the GPU in the RTC (remote DMA to the GPU). This will provide two advantages:

- avoid a double copy mechanism between the frame-grabber, the system memory and the GPU as DMA engines on frame-grabbers are optimized to write the data on the system memory
- minimize the interaction between the GPU and the CPU hence minimizing the impact of any OS interrupt policy, leading to jitter reduction

The principle of RDMA to GPU memory is depicted in Fig. 5. In a classical scenario (left), the DMA engine on the frame-grabber is optimized to send data to the system memory through the PCIe bus. These data then need to be copied to a buffer on the system memory pre-allocated using special kernel flags and proper alignment to allow for fast transfer to the GPU memory

(also called *pinned* memory). In an optimized scenario (right), the DMA engine on the frame-grabber directly transfers data to GPU memory through the PCIe bus, avoiding OS interrupts interference and the double copy mechanism.

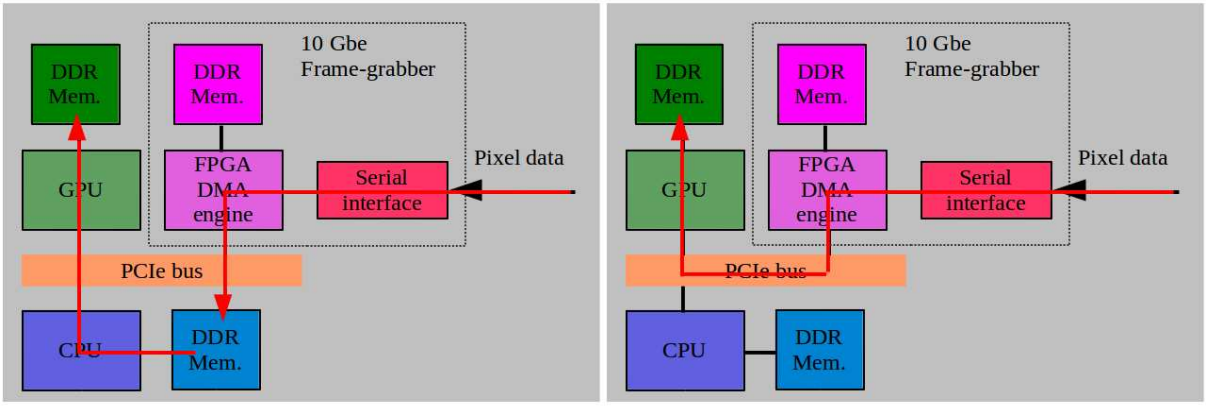


**Fig. 5.** Schematic view of data transfer from a camera to a GPU using a frame-grabber without (left) and with (right) RDMA.

Again, the CUDA framework allows for such a protocol to be implemented thanks to the GPUdirect API which provides kernel and user levels functions to map a buffer in GPU memory onto the PCIe memory space and retrieve the corresponding physical address. In this scheme, a buffer is pre-allocated on the GPU to hold the pixel data and mapped to the PCIe address space using a kernel module. The physical address of the mapped area is then sent to the DMA engine of the frame-grabber to be used as the receiving address for pixels data.

Initial concepts for the LGS wavefront sensors for the MCAO and LTAO systems on the E-ELT are based on Shack-Hartmann designs with 80x80 apertures of 20x20 pixels. Reaching the kHz frame-rate, if the pixels are encoded on 16 bits, leads to a theoretical bandwidth requirement of 40 Gb/s. Such bandwidth is now available on commodity hardware thanks to QSFP+ transceivers able to support various serial communication standards including up to four 10 GbE connection at full speed and the implementation of RDMA to GPU memory on a 40 Gb/s has already been demonstrated for the Infiniband communication standard[6].

Our team has started the development of a dedicated interconnect based on a PCIe development board holding a QSFP+ transceiver and an Altera Stratix V FPGA, an optimized PCIe IP core (QuickPCIe) including several DMA engines and an IP core for the UDP stack (QuickUDP) all from the PLDA company. The first performance benchmarks using the data generator of the reference design provided with the FPGA shows a peak bandwidth of about 42 Gb/s over the PCIe bus to the GPU memory. The integration of this interconnect and our RTC simulator in ongoing. We expect a GPU-based RTC prototype including a COTS GPU and a custom serial interconnect at 40 Gb/s to be fully implemented in the coming months according to the COMPASS road-map.

## 6 Conclusion

For the first time GPUs provide the unique combination of an extremely high computing throughput and a comprehensive C-based programming framework filling the requirements for both the

simulation and the real-time control of high density AO systems. The COMPASS project aims at developing and maintaining such a unified framework based on the use of GPUs as accelerators for the development of this next generation of AO systems. While the simulation code is already allowing to lead large scale simulation campaigns targeting SCAO systems on the E-ELT[1], the next evolution will introduce support for large scale clusters containing multiple GPUs on multiple nodes so as to target LGS MCAO and MOAO systems.

In parallel, our team is developing a GPU-based RTC demonstrator, able to target the MICADO SCAO module dimensioning and based on the use of commodity hardware and a custom serial to PCIe interconnect supporting RDMA to GPU memory at 40 Gb/s.

## References

1. Y. Clénet et al., This conference
2. D. Gratadour et al., SPIE **8447** (2012) id. 84475D
3. W. Ketchum et al., 2nd conf. on technology and instrumentation in particle physics (2011)
4. M. Le Louarn et al., SPIE **8447** (2012) id. 84475D
5. F. Rigaut, http://frigaut.github.com/yao/index.html
6. G. Shainer, et al. Computer Science-Research and Development **26.3-4** (2011) 267