Building a reliable, scalable and affordable RTC for ELTs AO systems

Damien Gratadour, Arnaud Sevin, Julien Brulé
LESIA, Observatoire de Paris
Outline

- Graphical Processing Units
- YoGA
- Feeding a RTCS with simulated data
- COMPASS
- Simulating a RTCS
- Acquisition interface
- Conclusions & perspectives
Graphical Processing Units

**NVIDIA Compute Unified Device Architecture (CUDA)**

- **2000:** programmable hardware for graphics = unified processor architecture, with scalar cores (NVIDIA). GPU term appears
- **2003:** the idea of General Purpose GPU appears (brookGPU API)
- **2007:** NVIDIA releases the CUDA framework for GPGPU
- **2009:** release of the OpenCL framework (not limited to GPUs)

**High performance per W and per $ ratio**

- Peak throughput > 1TLOPs in single precision for few k$ and few 100W
- Fastest computer on the planet: Titan (17 PetaFLOPs !) equipped with NVIDIA K20 (52 supercomputers in the top500 using GPUs)
Graphical Processing Units

From NVIDIA's CUDA-C programming guide
3-years development at LESIA

- Interfacing a high-level programming language (Yorick) with CUDA to build an optimized end-to-end simulation

- High-level code inspired by YAO

- ~x7 in performance as compared to multi-threaded simulations (see poster #13240 by Y. Clénet et al.)

- Parts of the code show even larger speedups (control, supervision)

- Comprehensive interface for data reduction / debugging

- Good agreement with results obtained with other simulation tools (YAO)

Now adding features ...

- Prototype model for a pyramid WFS (under testing ...)

- Adding multi-GPU mode (peer-to-peer + MPI support) in progress

- Stabilizing, testing, debugging, etc .. thanks to users
YoGA performance

- SCAO profiles in ms on a Tesla M2090 (single GPU mode)

<table>
<thead>
<tr>
<th>Telescope diam.</th>
<th>Turbu generation</th>
<th>Raytracing turbu</th>
<th>Raytracing DM</th>
<th>WFS</th>
<th>COG</th>
<th>Control</th>
<th>DM shape computation</th>
<th>Raytracing target</th>
</tr>
</thead>
<tbody>
<tr>
<td>4m</td>
<td>0.107</td>
<td>0.008</td>
<td>0.008</td>
<td>0.138</td>
<td>0.013</td>
<td>0.019</td>
<td>0.137</td>
<td>0.008</td>
</tr>
<tr>
<td>8m</td>
<td>0.192</td>
<td>0.022</td>
<td>0.023</td>
<td>0.459</td>
<td>0.031</td>
<td>0.060</td>
<td>0.562</td>
<td>0.023</td>
</tr>
<tr>
<td>20m</td>
<td>0.550</td>
<td>0.135</td>
<td>0.136</td>
<td>3.07</td>
<td>0.079</td>
<td>0.363</td>
<td>3.22</td>
<td>0.137</td>
</tr>
<tr>
<td>30m</td>
<td>0.927</td>
<td>0.299</td>
<td>0.300</td>
<td>6.73</td>
<td>0.168</td>
<td>0.915</td>
<td>7.39</td>
<td>0.302</td>
</tr>
<tr>
<td>40m</td>
<td>1.44</td>
<td>0.526</td>
<td>0.525</td>
<td>11.9</td>
<td>0.320</td>
<td>2.263</td>
<td>13.62</td>
<td>0.527</td>
</tr>
</tbody>
</table>

- Profiles dominated by pure simulation tasks (WFS and DM models)
- Performance of core algorithms on a single GPU almost ensure real-time @ ELT scale
- Not optimized for a specified GPU (auto-tuning)
Feeding a RTCS with YoGA data

- Distribute YoGA data over the network using openDDS
- YoGA: quasi real-time performance if data remains on the GPU
- But: openDDS is not (yet!) CUDA-aware
  - Need to copy the data from the GPU memory to the system memory
- SPHERE case (40x40 subaps, 6x6 pixels per subaps)
  - YoGA alone: quasi-real-time performance 750 it/s
  - Copying data to the system memory: 450 it/s
  - Switching on DDS: 430 it/s
- MICADO on the E-ELT (80x80 subaps, 6x6 pixels per subaps)
  - YoGA alone: 70 it/s
  - Copying data to the system memory: 37 it/s
  - Switching on DDS: 35 it/s
- No significant impact when switching DDS on
- Similar results when sending the data over the network (UDP) or on the system shared memory
Common framework

- Unifying simulation and RT frameworks
The COMPASS project

COMputing Platform for Adaptive opticS Systems

Build a unified framework on scalable heterogeneous architecture

- Federate efforts in the PHASE partnership (French HAR labs) to develop and maintain a numerical development platform for AO
- Associate partner: Maison de la simulation a joint laboratory between 5 partners (including CNRS, CEA and INRIA) for research in HPC
- Multi-disciplinary collaboration: AO + astrophysics + HPC
- End product: a high performance platform based on a total integration of software with hardware to run on scalable heterogeneous platforms

Goals:

- Software development platform: validate key components / test new concepts
- Efficient computing environment: run large scale simulations
- Unified and optimized framework for PHASE
- Enable real-time applications: pathfinder for accelerator-based AO control

30 months, funding secured thanks to an ANR grant : 800k€ (total investment : 2.5 M€ from partners = 260 men.months + equipment)
GPUs for hard real-time tasks

- GPUs are not system programmable: the OS cannot schedule nor control the GPU, only offload tasks
  - GPU tasks are non-preemptive: no notion of GPU task priority at the OS level
  - Professional GPU: i/o device with closed source driver (pros and cons)
  - Interrupt handling depends on the interaction of this closed source driver with the OS (non tweakable)
- => Minimize interaction between the OS and the GPU

- Challenge: transfer data on the GPU from a 3rd party device at low latency
  - Enabling technology for real-time applications with GPUs
  - Very hot topic in the GPGPU community (mil applications, automotive industry, etc..)
  - Because of the intrinsic black-box nature of GPUs, they have to be the masters in transactions
RTC simulator

Realistic simulator of MICADO SCAO RTC

- Copy the data, compute centroids, do MVM, transfer back commands
- MICADO: 80x80 subaps with 6x6 pixels / subaps
RTC simulator

- Maximizing throughput = maximize occupancy + minimize impact of memory transactions
  - Stream computing: overlap memcopies and compute (2 copy engines + 1 compute engine queue)
  - Concurrent kernel launches (up to 16 concurrent kernels on Fermi)

- Several options:
  - Dedicated streams
    - Requires a form of explicit synchronization
RTC simulator

- Maximizing throughput = maximize occupancy + minimize impact of memory transactions
  - Stream computing: overlap memcopies and compute (2 copy engines + 1 compute engine queue)
  - Concurrent kernel launches (up to 16 concurrent kernels on Fermi)

- Several options:
  - $N$ streams
  - Implicit synchronization
RTC simulator

- Performance for various numbers of chunks

![Graph showing performance for various numbers of chunks with MICADO+ StreamPerOps]
RTC simulator

- Performance with non-RT OS, no shielding: huge jitter
RTC simulator

- Performance with RT OS + shielding: reduced jitter
RTC simulator

- Performance for various numbers of chunks

**MICADO+ 1StreamPerOps**

![Graph showing performance with various numbers of chunks](chart.png)

- 1 GPU
- 2 GPU

- looptime (ms)
- Chunks
RTC simulator

- Performance for various numbers of streams
GPU-based RTC

- Not limited by frame transfer time
  - Achieved (measured) bandwidth: 4.8GB/s (above requirement of ~2GB/s)
  - Depends on the number of chunks / streams (need to saturate the bus: cannot hide completely the frame transfer time)

- But:
  - Framegrabber-to-host + host-to-device => bandwidth / 2! (IF framegrabber is as efficient as GPU)
  - More interaction with the OS: potentially more jitter
  - If we want less jitter, we need to tweak the RT OS (proper scheduling for GPU tasks, stronger shielding, interrupt handling, etc...)

- What can we do about that?
  - Launch a single perpetual kernel on the GPU (no interaction at all with the OS)
  - Framegrabber has to RDMA to the GPU
GPU-based RTC

- DDR memory
- GPU1
- PCIe switch
- CPU
- DDR memory
- FPGA
- 10 GbE frame-grabber
- Serial (10 GbE)

Pixel data
GPU-based RTC

- GPU1
- DDR memory
- PCIe switch
- CPU
- DDR memory
- FPGA
- Serial (10 GbE)
- 10 GbE frame-grabber
- Pixel data
GPU-based RTC

- CPU
- DDR memory
- PCIe switch
- GPU
- DDR memory
- FPGA
- 10 GbE frame-grabber
- Serial (10 GbE)
- Pixel data
Can we RDMA to the GPU?

- Yes! Technology is available: GPUdirect RDMA from NVIDIA.
- Devices on the PCIe root complex use PCIe BARs to communicate.

Several groups have demonstrated ~2GB/s bandwidth between a 3rd party device and GPU (PCIe gen 2.0):

- GE: mil applications (x5 less latency in DMA with GPUdirect).
- APE group from INFN (APEnet+ board for APE supercomputers): 1.5GB/s on commercial FPGA development board (Fermi GPU, PCIe gen 2 x8).
Conclusions

- GPUs provide for the first time a scalable solution to unify simulations and RT frameworks at the ELT scale
  - Commercial hardware, high programmability, high throughput
  - Reduce cost and risk while increasing robustness and upgradeability
  - Need to address fundamental discrepancies between simulations and RT goals / constraints (throughput / latency / jitter)

- COMPASS project: federate efforts of the French AO community to develop a high performance platform based on this unified framework

- Main challenge: low latency, GPU-friendly data transfer with serial protocols
  - Based on commercial hardware with limited amount of development
Perspectives

- Developing the next generation GPU-friendly interconnect
  
- LESIA is starting a collaboration with APE group in Rome to develop the next generation GPU-friendly 10GbE interconnect based on PCIe gen3.0
  
- Benefit from the logic already developed for APEnet+ (1GbE, PCIe gen2.0)
  
- Common development platform:
  
  - PLDA Stratix V development board (with QSFP cage, up to 4 10GbE ports)
  
  - PLDA QuickPCIe IP core (designed for PCIe gen3.0)
RTC simulator

- Performance for various numbers of streams (64x64 subaps)
Some numbers ...

**Typical E-ELT (39m diameter) SCAO module**

- 64x64 subapertures, 8x8 pixels / subap = 512x512 pixels (16bits) : 2MB
- 1kfps : 2GB/s data transfer
- 6k slopes, 3k DM commands : 36MB command matrix
- Single MVM : 2MxN-M flops : 36 Mflop, @1kHz : 36 Gflop/s
- Singular Value Decomposition : MxN² : 54 Gflop, but memory-bound!

**Typical MCAO module**

- 6x LGS WFS (64x64 subpas, 12x12 pixels/subap) @0.5kfps : 7GB/s
- 36k slopes, 7k command : 500MB command matrix
- Single MVM @0.5kHz : 250 Gflop/s
- Matrix inversion (LU : N³) : 350 Gflop