Unifying simulations and real-time frameworks to reduce development cost

YOrick with Gpu Acceleration

Damien Gratadour

LESIA, Observatoire de Paris

Outline

- AO end-to-end simulations at the ELT scale
- Hardware accelerators : GPUs
- A common framework for simulations and RT
- The COMPASS project
- The missing link : a low latency acquisition interface
- Conclusions & perspectives

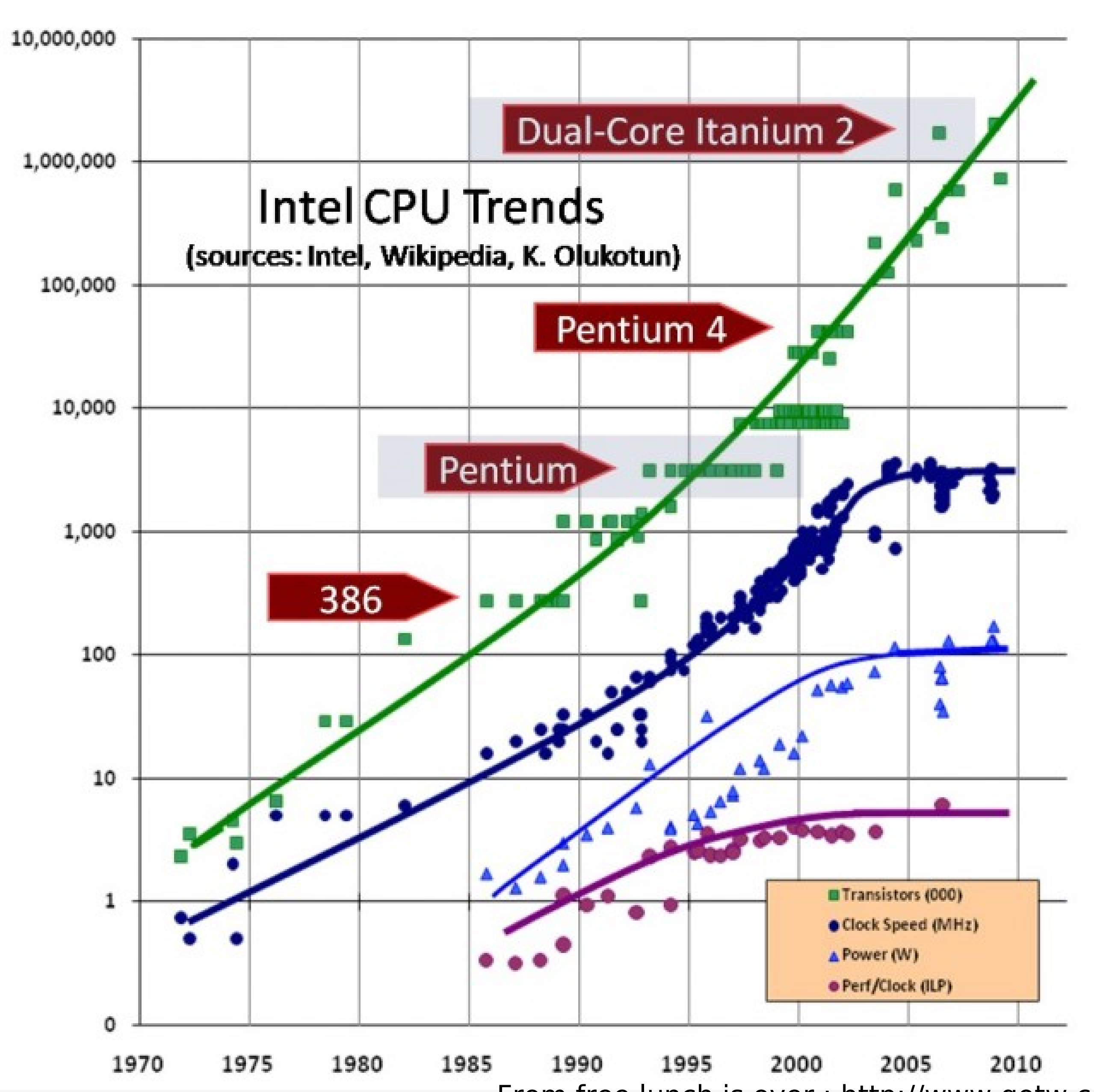
AO end-to-end simulations

- Multiple physics from turbulence generation to control theory
  - Stochastic phenomena so Monte-Carlo gives the most realistic results
- ELT makes it a large scale problem :
  - Simulated pupil : ~2k x 2k pixels (hence 20k x 20k phase screens)
  - FFT support size for image formation : ~4k x 4k
  - ~5k sub-apertures and ~5k DM commands (several 100 GFLOPs needed for MVM in a simple real-time control scheme)
  - Large number of iterations to reach convergence (10-100k)
- Advanced concepts for ELT AO modules (LGS MCAO, LTAO)
  - Several WFS and DMs, use of LGS : larger sub-apertures, more pixels, need to generate high-resolution images for LGS
- Change of paradigm :
  - From *home-brewed* models to unified high performance models
  - AO simulations must enter the HPC era !

Hardware acceleration

◎ Current trends in the HPC industry

- Processors just not running faster since beginning of the century (clock speed cannot increase : heat, power, current leakage)

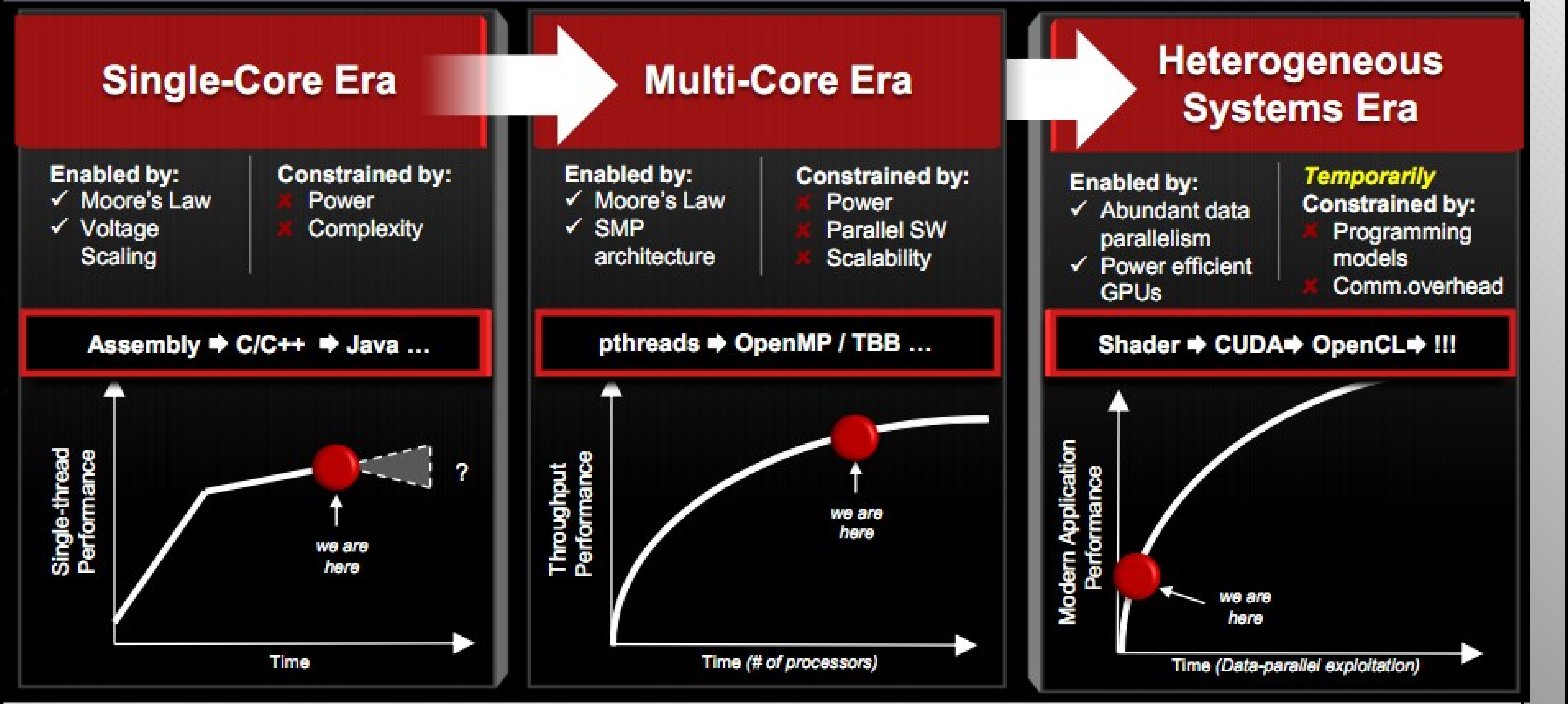


From free lunch is over : http://www.gotw.ca/publications/concurrency-ddj.htm

Hardware acceleration

- Current trends in the HPC industry
  - Processors just not running faster since beginning of the century (clock speed cannot increase : heat, power, current leakage)
  - Concurrency is the new paradigm : hyper-threading, multi-core, many-core
  - The next revolution is primarily software : introduce concurrency in our software
- Heterogeneous architecture
  - General purpose multi-core processor + special purpose co-processor (DSP, GPU, MIC) or custom acceleration logic (FPGA) + high speed / low latency network interconnection
  - Programming can intrinsically be tricky (different hardware = different programming models)
- Many-core : several approaches
  - Use scaled down vector processors (Intel MIC, imminent)
  - Use large amount of small scalar processors in a SIMD model (NVIDIA GPU, available)
  - Requires fine grain parallelism (+ vectorization for Intel MIC): “no free lunch” !
  - Co-processors: attached through PCIe bus (bandwidth + latency + memory model)

Hardware acceleration



Single-Core Era

Enabled by:
- ✓ Moore's Law
- ✓ Voltage Scaling

Constrained by:
- ✗ Power
- ✗ Complexity

Assembly ➡ C/C++ ➡ Java …

Multi-Core Era

Enabled by:
- ✓ Moore's Law
- ✓ SMP architecture

Constrained by:
- ✗ Power
- ✗ Parallel SW
- ✗ Scalability

pthreads ➡ OpenMP / TBB …

Heterogeneous Systems Era

Enabled by:
- ✓ Abundant data parallelism
- ✓ Power efficient GPUs

Temporarily Constrained by:
- ✗ Programming models
- ✗ Comm.overhead

Shader ➡ CUDA➡ OpenCL➡ !!!

Modern Application Performance
Time (Data-parallel exploitation)
we are here

From “The death of CPU scaling” : http://www.extremtech.com

GPU

- 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 NVIDIA / GPUs)
- Developments in GPU architectures
  - Streaming Multiprocessors (SM) = clusters of processing cores. GPU = group of SMs + scheduler
  - Inside a SM : SIMD units = group of executable threads (warps in CUDA)
  - Several memory levels: low latency at the SM level (registers, shared memory) / higher latency at the chip level (global memory)
- High performance per W and per $ ratio
  - Peak throughput > 1TFLOPs in single precision for few k$ and few 100W
  - As of nov. 2012 : Fastest computer on the planet : Titan (17 PetaFLOPs !) equiped with NVIDIA K20 (62 supercomputers in the top500 using GPUs)

YoGA

YOrick with Gpu Acceleration

- 2-years development at LESIA
  - Interfacing a high-level programming language (Yorick) with CUDA to build an optimized end-to-end simulation
  - ~X10 in performance as compared to single thread simulations
  - Parts of the code show even larger speedups (control, supervision)
  - Comprehensive interface for data reduction / debugging
- Loop closed ! Now adding features ...
  - Multiple phase screens, multiple SH WFS, NGS and LGS, multiple DMs, multiple targets available
  - Simple LS control algorithm
  - 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)

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

- Profile dominated by pure simulation tasks (WFS and DM models)
- Performance of core algorithms on a single GPU almost ensure real-time
- Not optimized for a specified GPU (auto-tuning)

Simulations & RT

- Core algorithms in simulations are the core of a RTC
  - Simulations : usually used to evaluate new control strategies (lot of time spent to develop, debug and test new algorithms)
  - RTCS : needs to be tested using simulated data / needs to be supervised using simulated data
- Different goals
  - Simulations : strive for high end-to-end throughput (reduce computational time) for quick diagnosis or to lead large scale parametric studies
  - RT : needs high throughput but primarily driven by low latency & jitter
- Different constraints
  - Simulations : all internal (generates its own data and play with it)
  - RTC : interacts with the system (feeding and being fed)

Common framework

- A lot of money (and brains !) could be saved if we had a common development framework
  - Throughput is already there (if optimized): see Arnaud's talk later today
  - First time an architecture could be used for both simulations and RT applications thanks to a comprehensive development framework (CUDA / OpenCL)
  - Rather cheap solution using high-end off-the-shelf hardware with broad range market and free development framework (as compared to FPGA / DSP solutions)
  - Development cost reduced to a minimum (only done once)
  - Risk decreases while robustness and upgradeability increase significantly
- Need to address fundamental discrepancies between RT applications and simulations goals / constraints
  - Minimize latency and jitter
  - Interact with the outside world optimally

Common framework

- Unifying simulation and RT frameworks

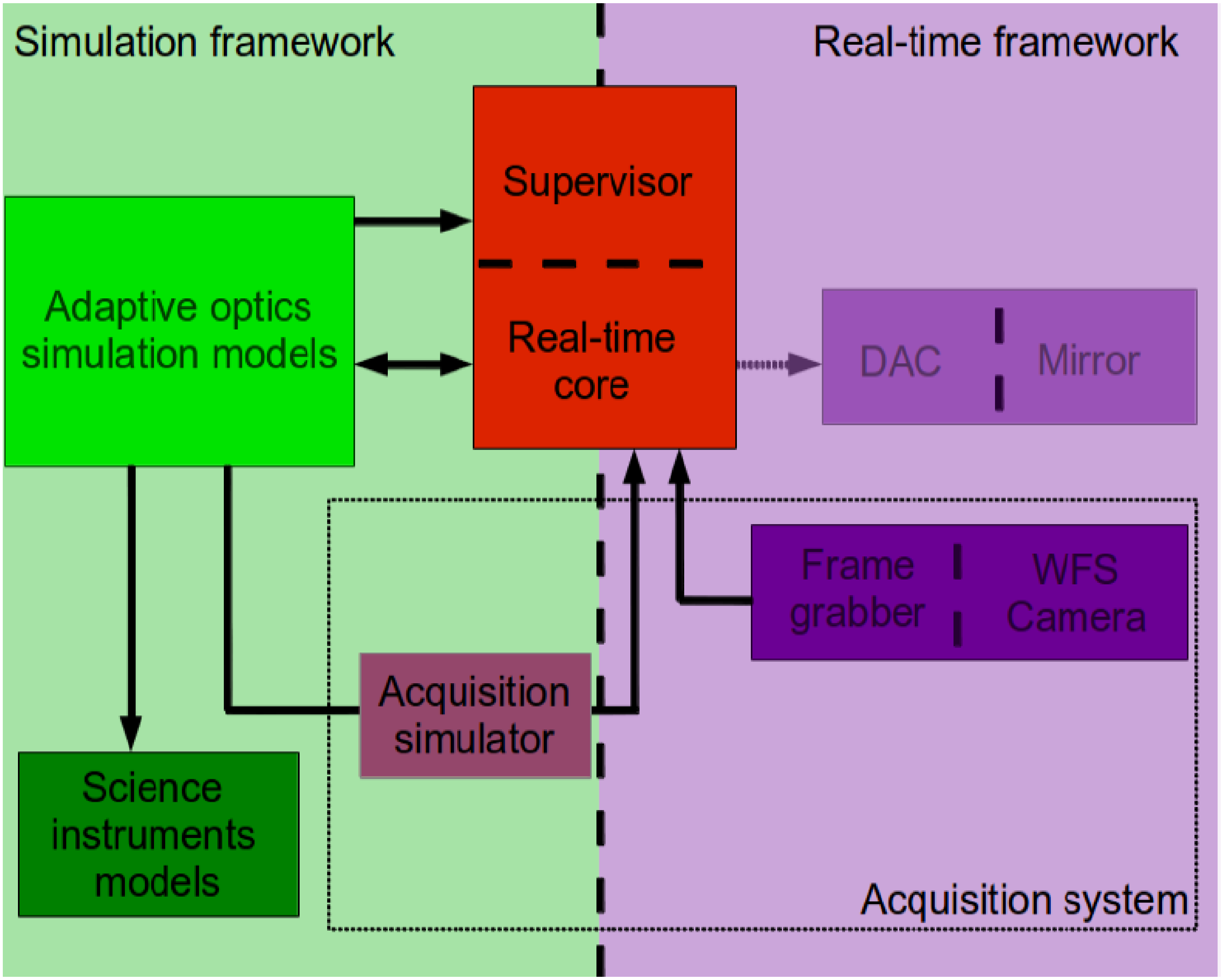


Illustration 1: COMPASS framework

The COMPASS project

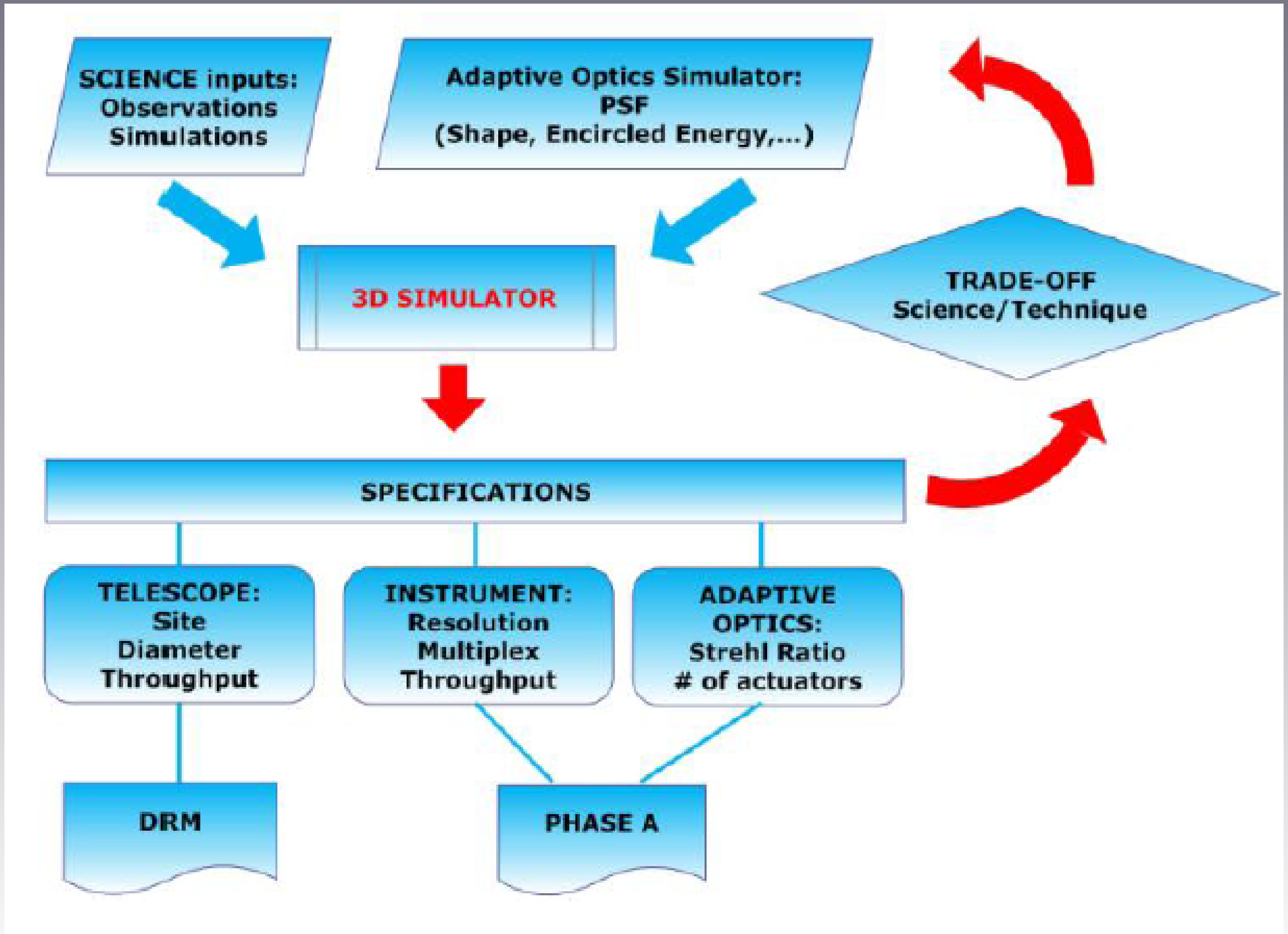
- COMputing Platform for Adaptive opticS Systems
- Build a unified framework on scalable heterogeneous architecture
  - Federate efforts in the PHASE partnership (French HAR labs) to develop and maintain a numerical development platform for AO
  - Associate partner: *Maison de la simulation* a joint 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)

The COMPASS project

- COMPASS: make the link with E-ELT instrumentation

- Full scale end-to-end simulation platform from astrophysical objects to AO corrected data

- Generalized decision tool for ELT-CAM, ELT-IFU, ELT-MOS



The COMPASS project

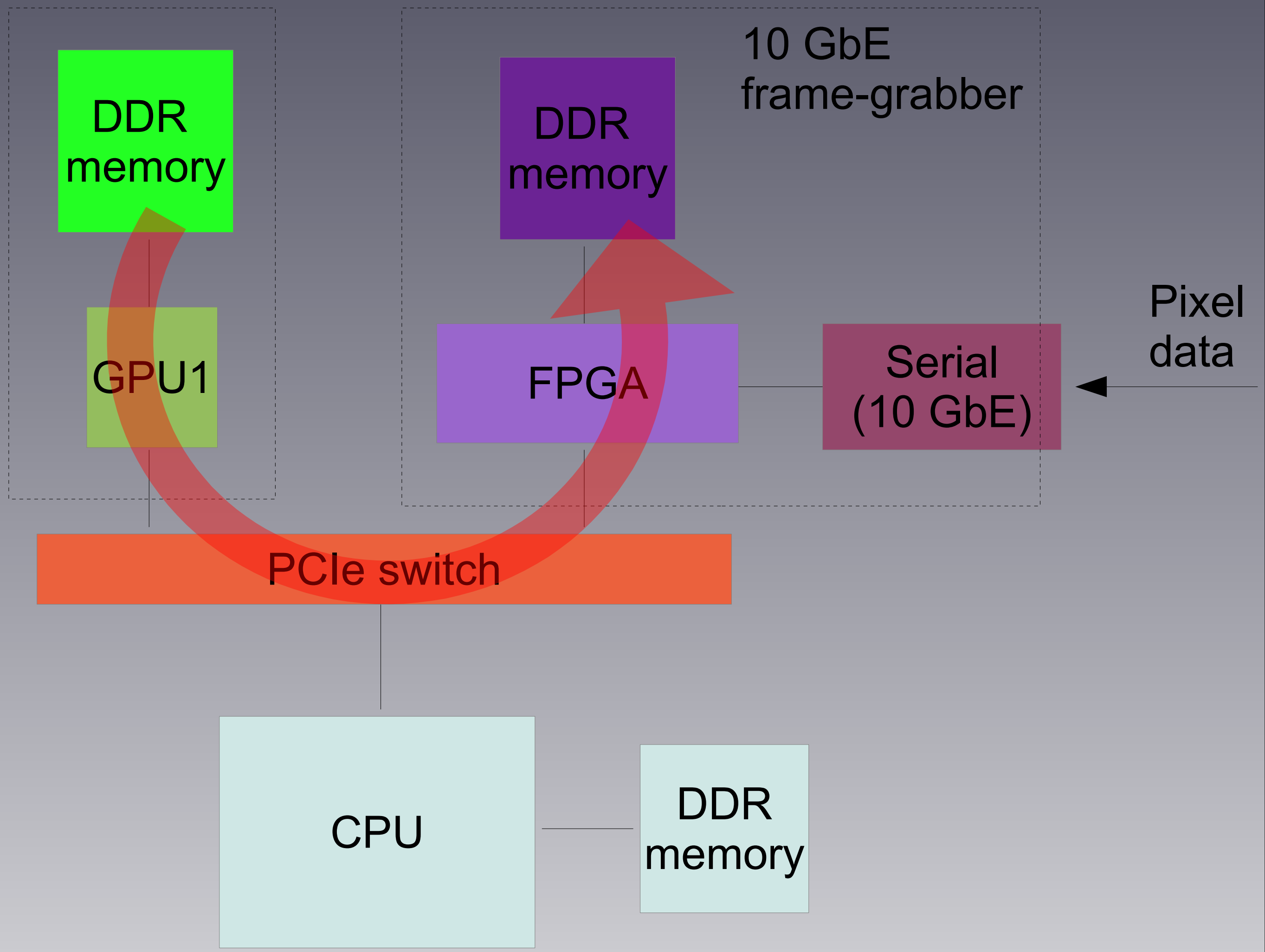
- COMPASS: port key algorithms to many-core
  - Models: Realistic deformable mirror model, pyramid WFS model
  - Control strategies: Minimum variance, LQG, Learn & Apply, iterative methods (FRiM), Fourier methods
  - Supervision strategies: sparse matrix inversion, conjugate gradient
- The missing link: low latency data transfer
  - Key requirement for an AO RTC using coprocessors (GPU, MIC) is the ability to transfer data at high bandwidth and low latency
  - Bandwidth is there (128Gb/s for PCIe x16 Gen3)
  - Limited by latency in transaction (cam. controller copies to the host memory and host copies to co-processor)
  - Need to implement RDMA between the cam. controller and the co-processor

Low latency data transfer

- Interface between cam. controller and GPU
  - DMA: use pinned memory on the host (i.e. that cannot be moved or swapped by the system)
  - CUDA: this host pinned memory is mapped on the GPU address space. GPU can access this memory asynchronously
  - Heterogeneous architecture: can we transfer at minimum latency (PCIe bandwidth) data from a 3rd party device to the GPU ?
- GPU: opaque layer of vendor-supplied driver
  - GPU must always be the master
  - The 3rd party device driver should map his memory to somewhere mappable by the GPU
  - 3rd party device memory must be pinned to ensure DMA

Low latency data transfer

- CUDA: GPUdirect & Unified Virtual Addressing
  - GPU can map addresses on the PCIe bus in their address space
  - Potentially enables RDMA from 3rd party devices through PCIe
  - Already used by Infiniband manufacturers (Mellanox) to provide *CUDA-friendly* communications in large scale clusters



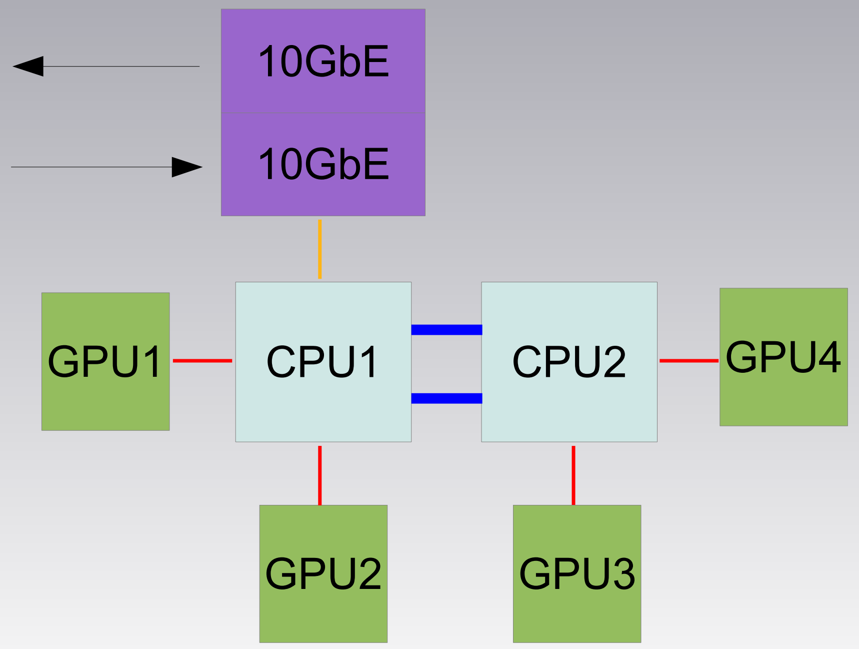
- Could be implemented on PCIe development boards (PLDA, HighTech Global, ...)
  - Requires a limited amount of development on top of the PCIe core on the FPGA
  - Easy access to standard serial protocols (10 GbE for instance)

GPU-based RTC

- COMPASS: building a prototype for a GPU-based RTC

- Using this fast serial interface

- Commercial hardware: dual CPU socket + 4 GPUs



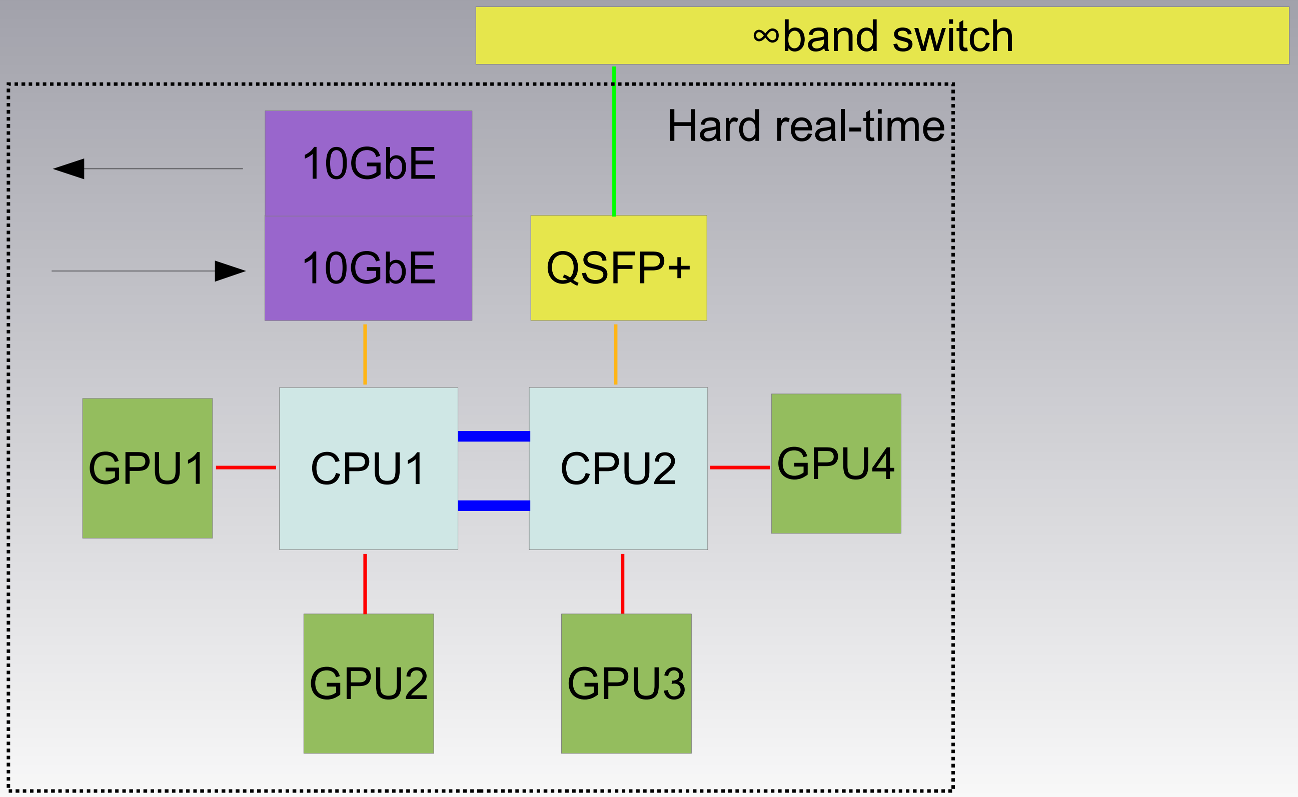
GPU-based RTC

- COMPASS: building a prototype for a GPU-based RTC

- Using this fast serial interface

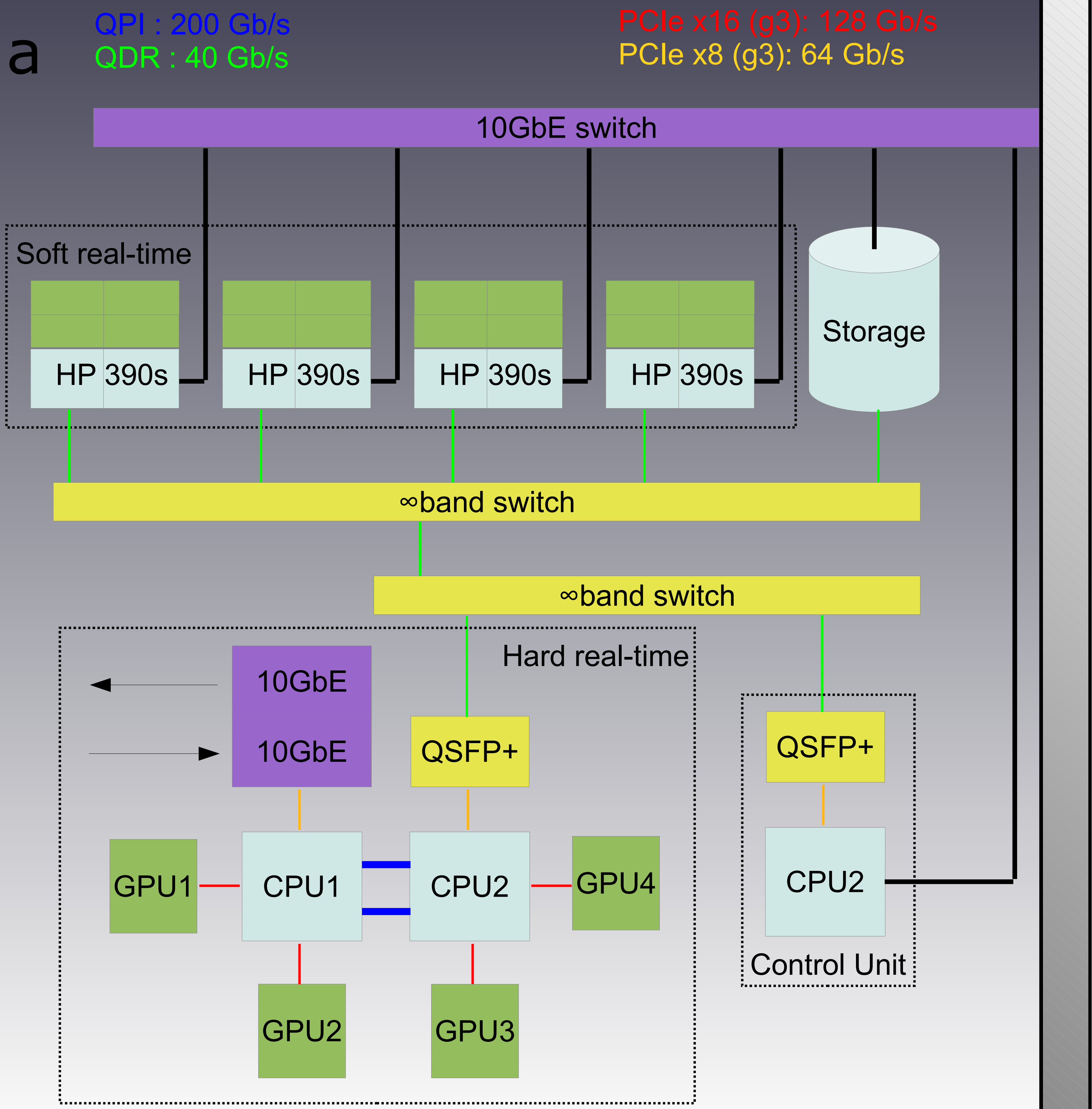
- Commercial hardware: dual CPU socket + 4 GPUs

- Scalable for increased throughput



GPU-based RTC

- COMPASS: building a prototype for a GPU-based RTC
  - Using this fast serial interface
  - Commercial hardware: dual CPU socket + 4 GPUs
  - Scalable for increased throughput
  - Interfaced with soft real-time and telemetry at high bandwidth



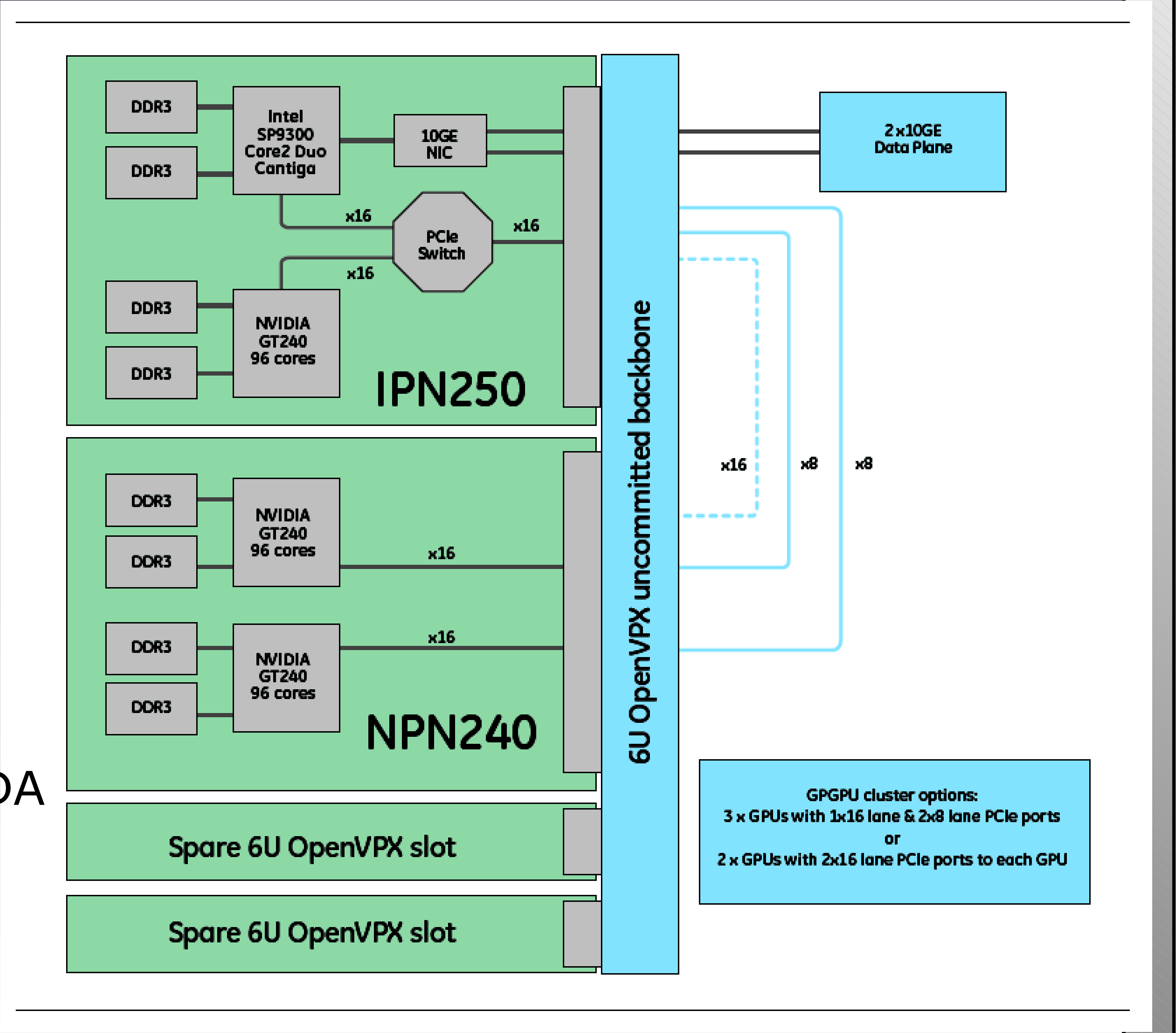
The long-term approach

- GPU manufacturers: aggressive development schedule
  - A new board every 6 months, new evolution of the architecture every 2 years.
  - Long term hardware procurement not ensured with broad market products
  - Market not primarily driven by HPC products
- Need for tailored COTS products with long-term maintenance and upgrade strategy
  - VME specs are compatible with GPU requirements : asynchronous transfers, DMA, master / slave boards
  - VPX : new high bandwidth evolution of the VMEbus with 10GbE, PCIe and RapidIO integration
  - GPGPU is coming to VPX : GE products

The long-term approach

- Example of COTS GPGPU system based on VPX

- GE intelligent Systems VPX 6U CUDA starter Kit
- 3 GPUs systems
- Next generation will embark Kepler chips
- Native DMA with 10 GbE through VPX
- Linux environment
- Same framework for core computations : CUDA
- Specialized framework for communications



GPGPU cluster options:
3 x GPUs with 1x16 lane & 2x8 lane PCIe ports
or
2 x GPUs with 2x16 lane PCIe ports to each GPU

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
  - Need for a long-term approach : VPX COTS solutions ?

Perspectives

- Intel Knights: the new competitors
  - Many Integrated Core architecture inherits from project Larrabee: 64 bit x86 vector (512 bit wide) processor for graphics
  - PCIe co-processor, 60 cores @ 1.2GHz, limited on-board memory (8GB)
  - Very similar to GPUs in some ways … well it's an hybrid architecture !
- No magic compiler = “no free lunch” !
  - Still need to parallelize the code to reach peak performance
  - Vectorization: additional constraint. What fraction of our codes is vectorizable ?
- Better integration with the GPP
  - Sure it's an Intel product !
  - How fast optimized standard numerical methods / optimized development tools will emerge for MIC ?
  - Remember NVIDIA won the first round thanks to CUDA