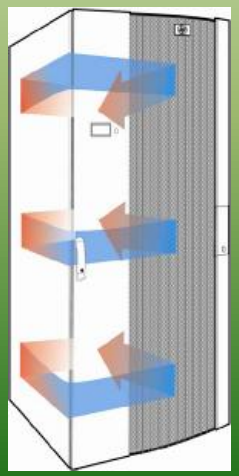
designing a GPU Computing Solution

Patrick Van Reeth – EMEA HPC Competency Center - GPU Computing Solutions
Saturday, May the 29th, 2010

Current Computing Challenges

Accelerators supplement multi-core with more silicon devoted to computation



What is an "HPC Accelerator"?

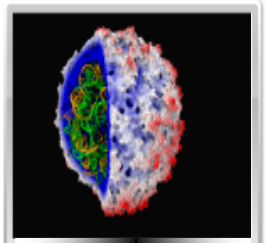
- Hundreds of functional units executing in parallel
- Speedup applications by 2x, 10x, 30x, or even 100x!
- Really fast
- Really cheap
- Was hard to program, but getting easier now
- Specific applications benefit
  - Not useful for most applications
  - Excellent for a growing number of highly parallel applications
    - When it works, it can really fly!

NVIDIA Tesla Industry Results

– Speedups are 20x to 150x

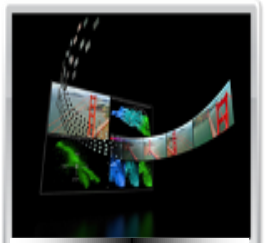
146X

Interactive visualization of volumetric white matter connectivity



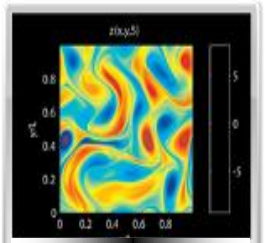
36X

Ionic placement for molecular dynamics simulation on GPU



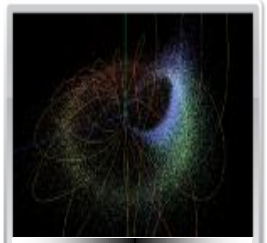
19X

Transcoding HD video stream to H.264



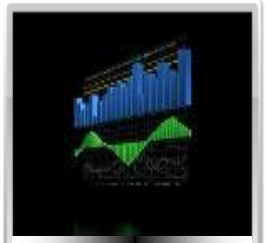
17X

Simulation in Matlab using .mex file CUDA function



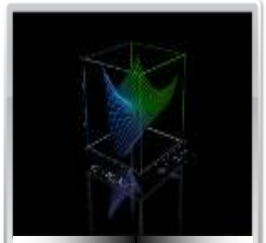
100X

Astrophysics N-body simulation



149X

Financial simulation of LIBOR model with swaptions



47X

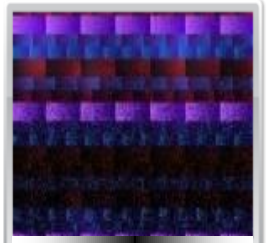
GLAME@lab: An M-script API for linear Algebra operations on GPU

20X

Ultrasound medical imaging for cancer diagnostics

24X

Highly optimized object oriented molecular dynamics

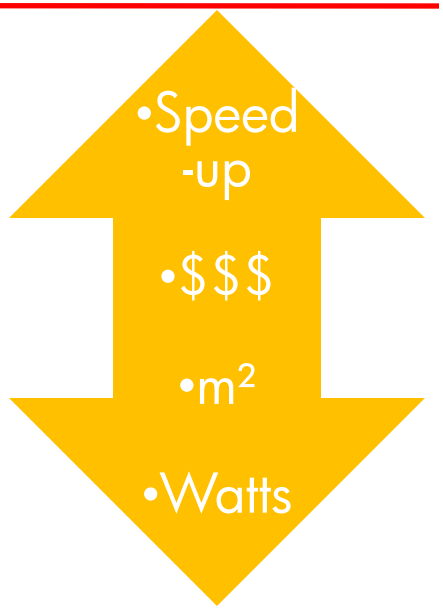


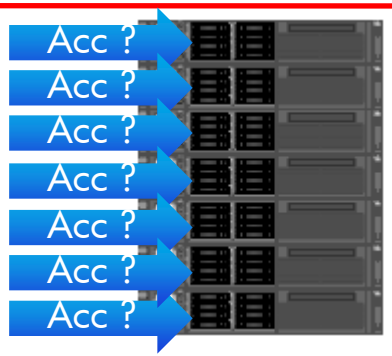
30X

Cmatch exact string matching to find similar proteins and gene sequences

How Accelerators can improve the Applicative Environment ?

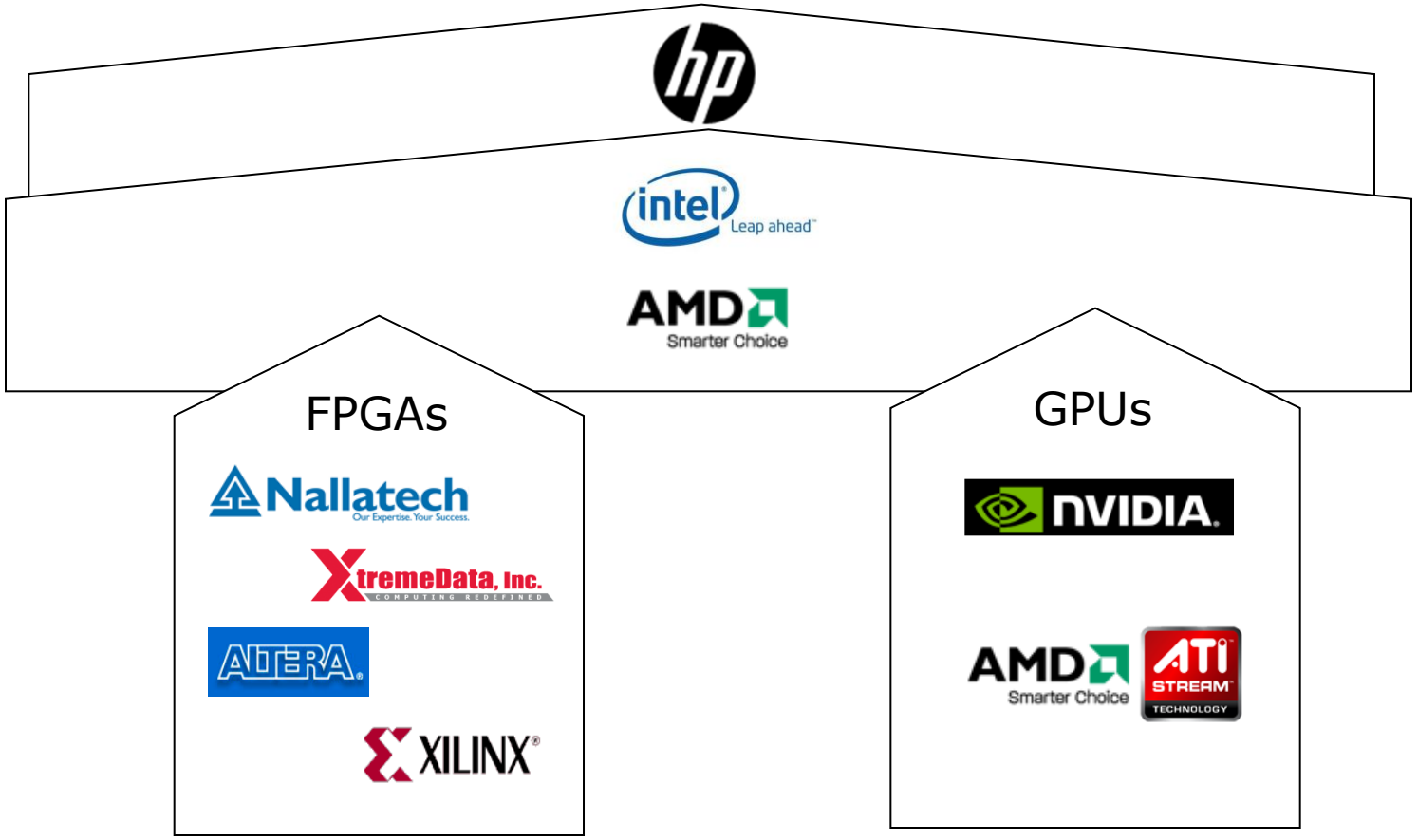
- Multi Nodes
- Multi CPUs
- Multi-cores
- Many cores : brings Hundreds of functional units executing in parallel





HP Accelerator program

Hybrid Platforms



- Large codes and operating systems
- Floating point

- Text or integer
- Performance per Watt

- Floating point (engineering computations)

Application Speed-up key factors

Understand the existing code :

- Track the most computationally expensive areas (inner loops…)
- Probe the load-balancing on nodes & cores
- Examine the data set splits
- Probe the communications
- …

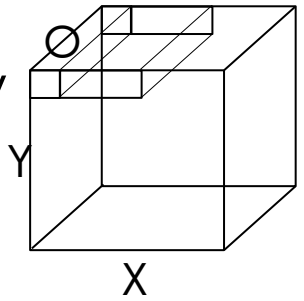
The tool box is rich :

- Multi-core CPUs
- Memory bwth/Latency
- InterConnect
- Accelerators
  - PCI-E speed
  - SW Environments : C & Fortran compilers, OpenCL, CUDA, HMPP, Allinea, TotalView…

Be Agnostic first !!!

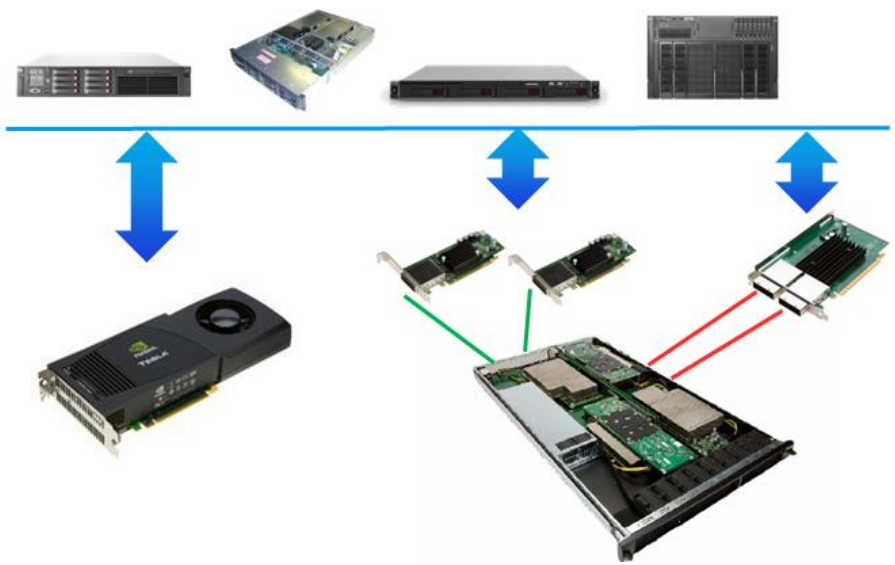
Accelerator Success Factors
TRACK THE MINES !!!

- GPU rules :
  - Embarrassingly parallel is good !
  - Minimize memory access versus calculation. Too few calculations per memory read/write is bad.
  - use cache when possible (use memory hierarchy !)
  - Thread/core mapping is very important
- Redesign the code architecture to scale to the right levels : Node(s), core(s), GPU(s)
- Minimize CPU-GPU transfers
  - Partition the data sets to stay in the onboard memory boundary
  - Data set split : shift along the correct X, Y, Z axis !
  - Hide communications



- Balance CPU (e.g. summations, less flops…) and GPU (e.g. convolutions, transpositions, …) execution time appropriately.

Determine the right platform

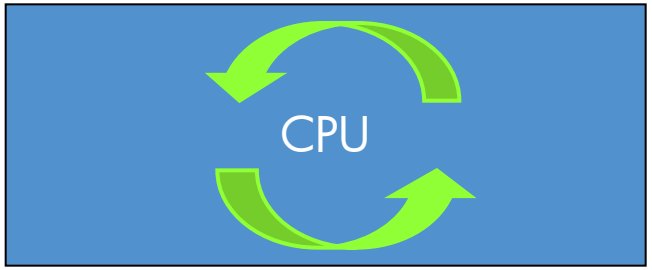


- CPU / GPU execution time ratio → will determine the type of node cores, the # GPU per node
- PCI-E communication between CPU & GPU will determine if can be shared by multiple GPUs
- Single or double precision will determine the GPU type
- Data set size (SMP, Cluster, …)
- $$$, Watts, perf

GPU Solution architectures (at node level)

Applicative environment topology 1

High CPU activity

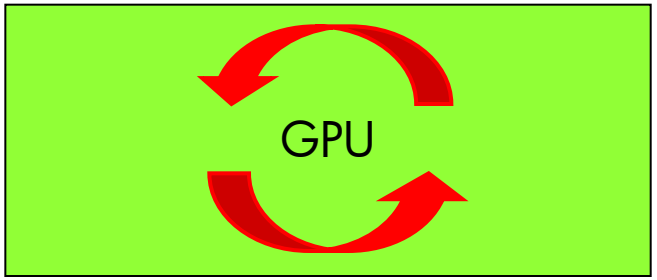


CPU activity limited to communications: Low-end bi-socket servers

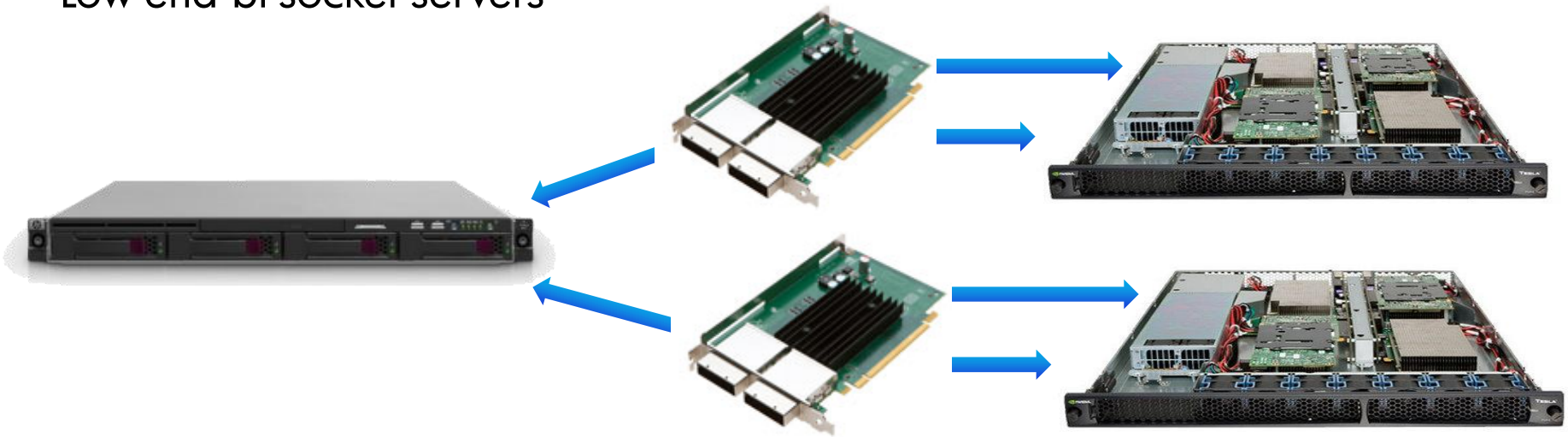
Few CPU/GPU communications

Low communication : PCI-E can be shared by GPUs

High GPU activity



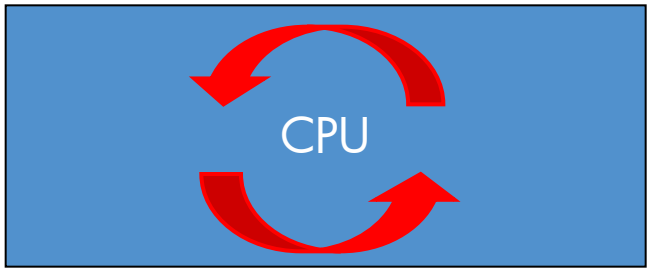
many GPU cores requested



GPU Solution architectures (at node level)

Applicative environment topology 2

High CPU activity

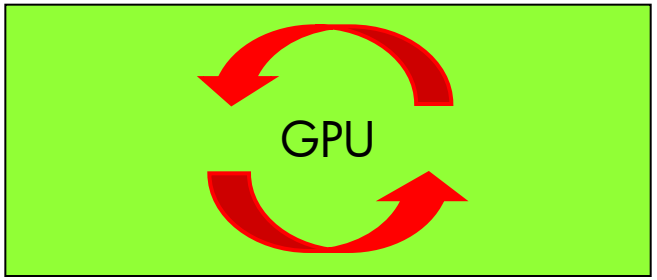


CPU intense activity,
Large Data sets :
N x multicore
Tbytes Mem

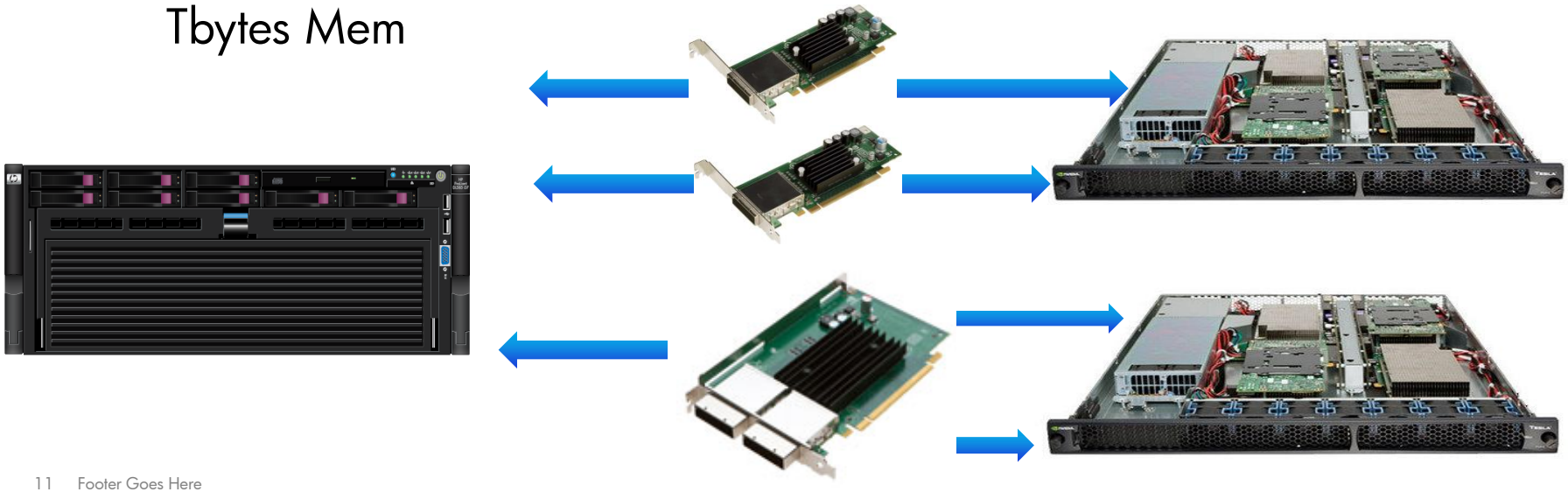
Few CPU/GPU communications

Low communication : PCI-E can be shared by GPUs

High GPU activity



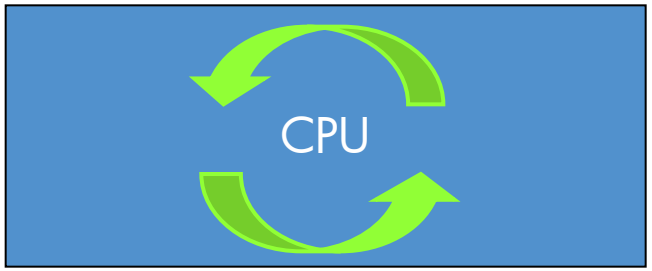
many GPU cores requested



GPU Solution architectures (at node level)

Applicative environment topology 3

Few CPU activity

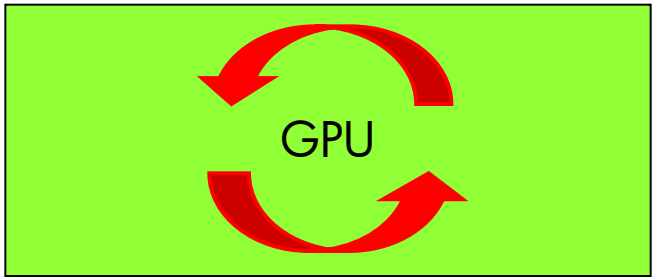


CPU activity limited to communications : Bi-socket Servers

Few CPU/GPU communications

GPU Dedicated PCI-E 16x

High GPU activity



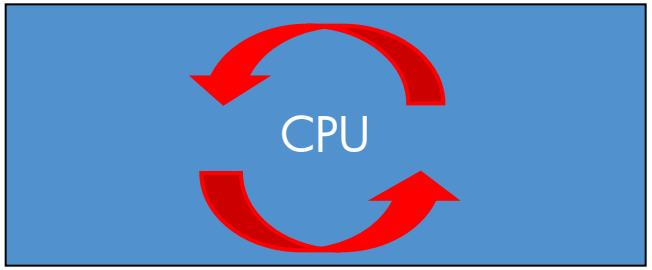
Direct access to many GPUs

PCI-E 16x

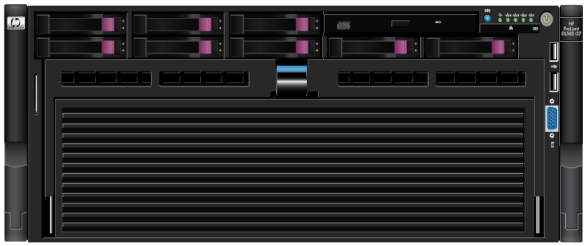
GPU Solution architectures (at node level)

Applicative Environment topology 4

High CPU activity



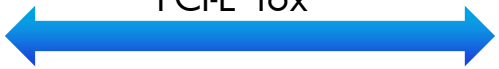
CPU intense activity,
Large Data sets :
N x multicore
Tbytes Mem



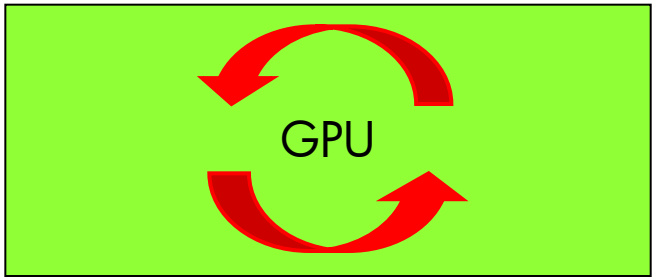
Critical CPU/GPU communications

GPU Dedicated
PCI-E 16x

PCI-E 16x



High GPU activity



Direct access to many GPUs

Conclusions

When request to speed-up an existing environment :

- analyze & probe the existing code,
- Identify if/where accelerators fit
- Adding GPU is not exactly playing LEGO !
- Define the right CPU/GPU split, and data set partitioning
- Adapt the HW architecture :
  - At node level
  - At cluster level

Thank You !