Centre de Calcul de l'Institut National de Physique Nucléaire et de Physique des Particules

GPUs at CC-IN2P3

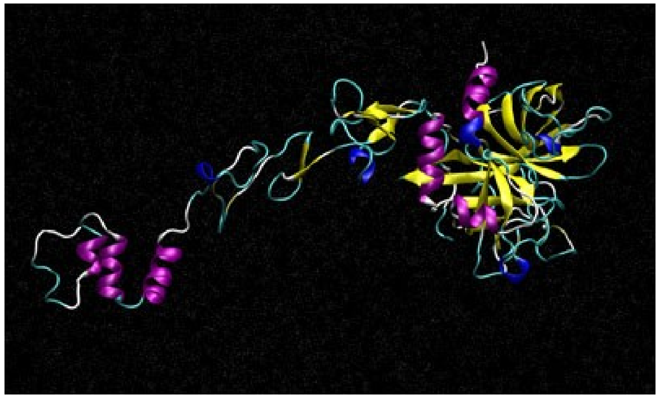
February 2016

Introduction

- Why GPUs?
- What kind of GPUs?
- How to integrate GPUs?
- Some considerations about how to use them
- Some feedback

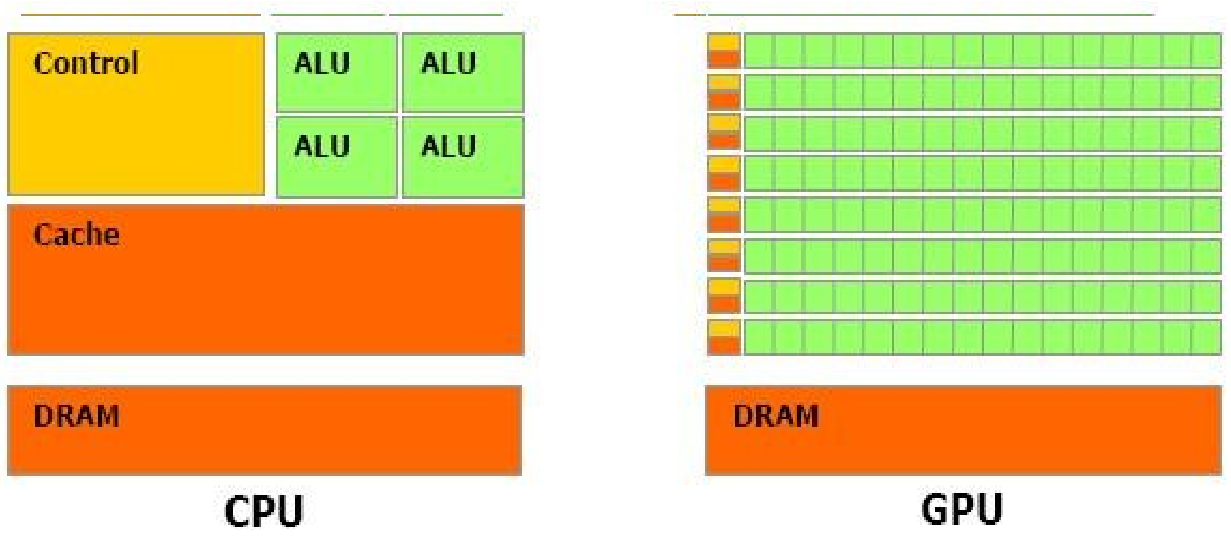
Why would we like to have GPUs?

- Because users want to try it
  - Signal processing
  - Simulating particles propagation
- Because users want to use it for production
  - Biomolecular dynamics



Why would we like to have GPUs?

- It's suited for highly parallel tasks



- It's more energy-efficient
  - Lower frequencies
  - More ALUs than control structures

What kind of GPUs?

- NVIDIA and AMD have products
- Main choice to do: OpenCL or CUDA?
- Most users wanted to have CUDA available
- AMD announced CUDA support at SC15 but…
  - Not yet ready
  - Partial
  - What about performances ?
- For us, it will be NVIDIA

OpenCL

NVIDIA CUDA

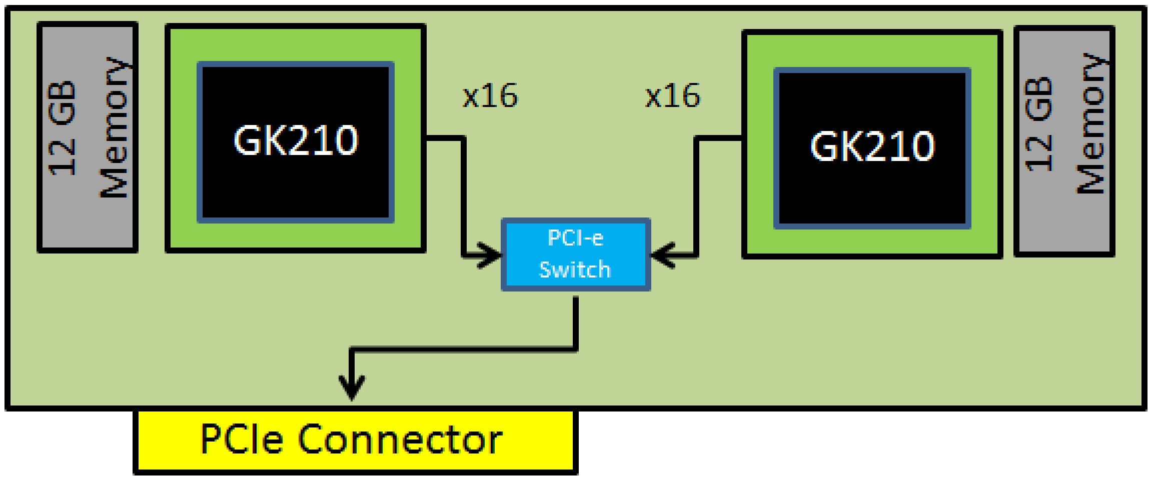
What kind of GPUs?

- 3 categories of products:
  - GeForce: gaming, but supports CUDA
  - Quadro: graphics
  - Tesla: GPGPU
- Dell only supports Tesla
- Tesla advantages over GeForce:
  - Double precision
  - ECC
  - Passive cooling/board design adapted to servers
  - A few more tools to manage GPUs
- Price makes a big difference

What kind of GPUs?

- Tesla K40 board
  - 1 GPU GK110
- Tesla K80 board
  - 2 GPUs GK210
  - Lower frequencies
  - A few less active ALUs
  - Bigger caches
  - OS detects 2 GPUs

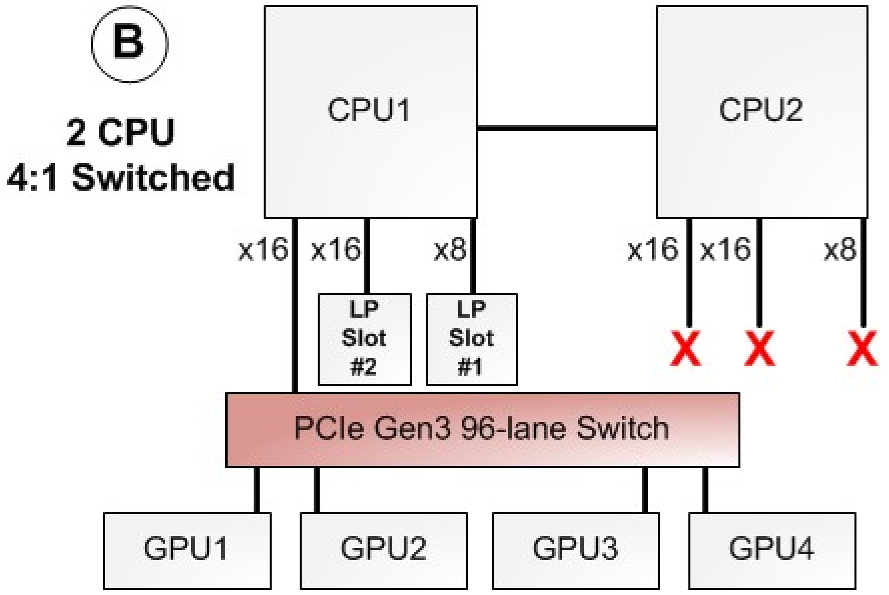
Tesla K80 Block Diagram

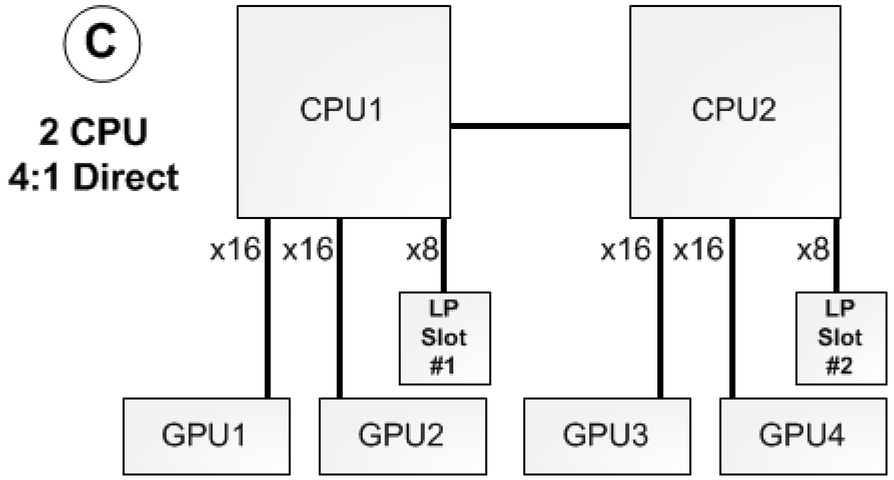


- Tested with some OpenMM based simulation

How do we want the host to be?

- copy between host memory and GPU memory: time expensive
- Several possible host/GPU interconnect schemes:





=> Single GPU vs multi-GPU choice

- Infiniband can help with multi GPU on multiple hosts

How do we want the batch scheduler to be?

- Univa Grid Engine integrates GPU
  - allocates jobs to 1 or more GPU
  - Can manage NUMA topology (topology masks)
  - But currently lacks accounting

How to use GPUs/a few things to be aware of

- Try to spot where you can parallelize, but...
- … to write an efficient GPU code, it might take redesigning from scratch
- To be efficient, fill up the GPU!
- Intermediate results should never be copied to the host
- Optimize memory usage/buses usages
- Choose carefully the operations
- Use optimized operations when possible
- Be careful about the precision/check the results

Some feedback from other centers...

- Optimization is very GPU dependent: better to have an homogeneous platform
- Not more than 1 process per GPU at a time
- Intense memory utilization is what makes the GPU boil
- Be careful with variable casting, very tricky to find the error

Conclusion

- Usecases for GPUs exist in HEP
- CPUs can't compete when GPU fits the computation
- Difficult to use GPU seamlessly with existing code
- Ours users have different amounts of experience with GPUs
- We will provide a production cluster
- But we must keep in mind that many users first need a test platform
- And their needs might evolve!