CS 6023 - GPU Programming

GPU’s place in a PC

01/11/2018

Setting and Agenda

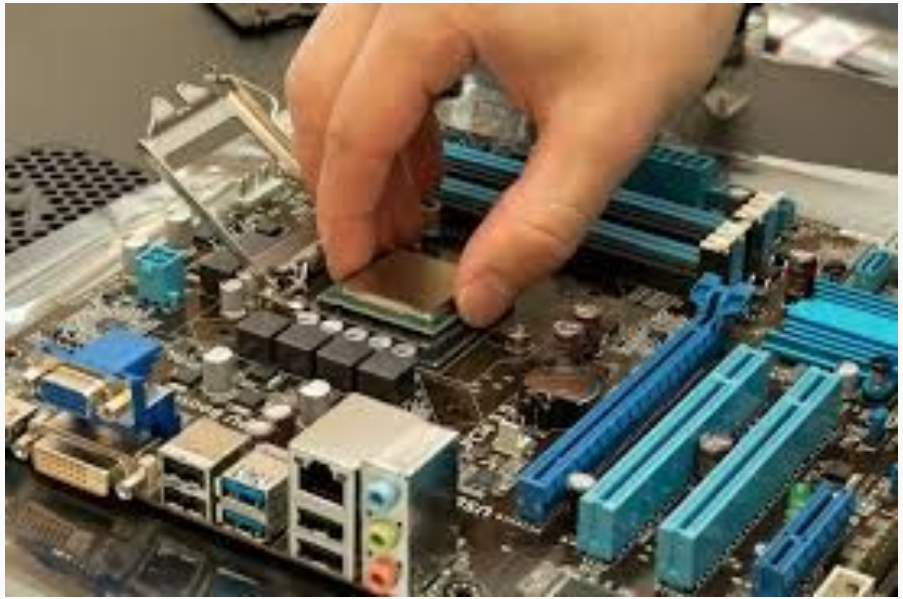
- Understand a few things about PC architecture
- How are GPUs connected in a PC’s architecture
  - Legacy
  - Recent trends
- Multi-GPU execution

Role of bandwidth in parallel systems

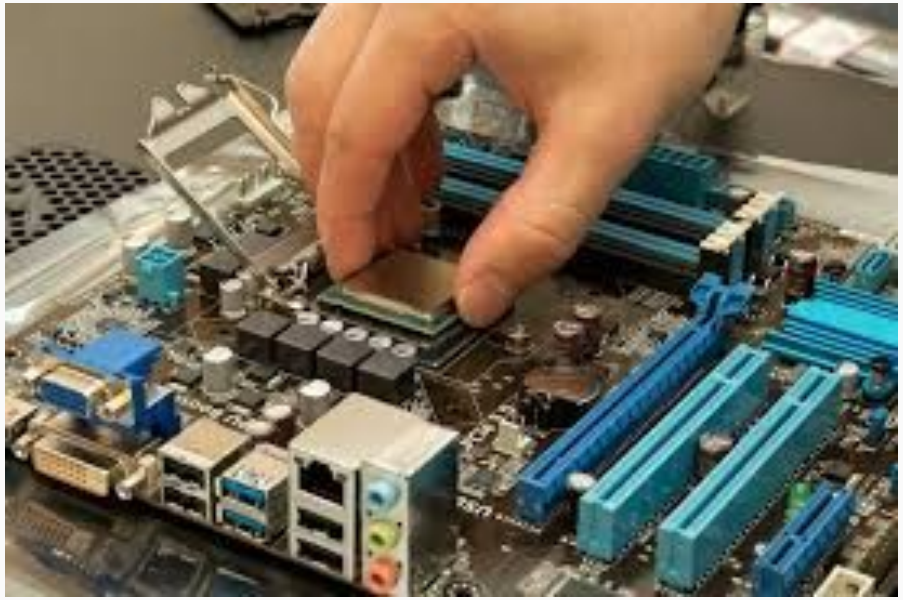
- In most parallel systems, bandwidth between compute units is a key metric that affects performance
- Very significant for modern workloads such as deep learning, video processing, networking
- Dataflow optimisation, caching, buffering, reordering play a significant role in performance

- In this context, its important to understand and exploit how we connect GPUs to the CPU

What is in your motherboard?

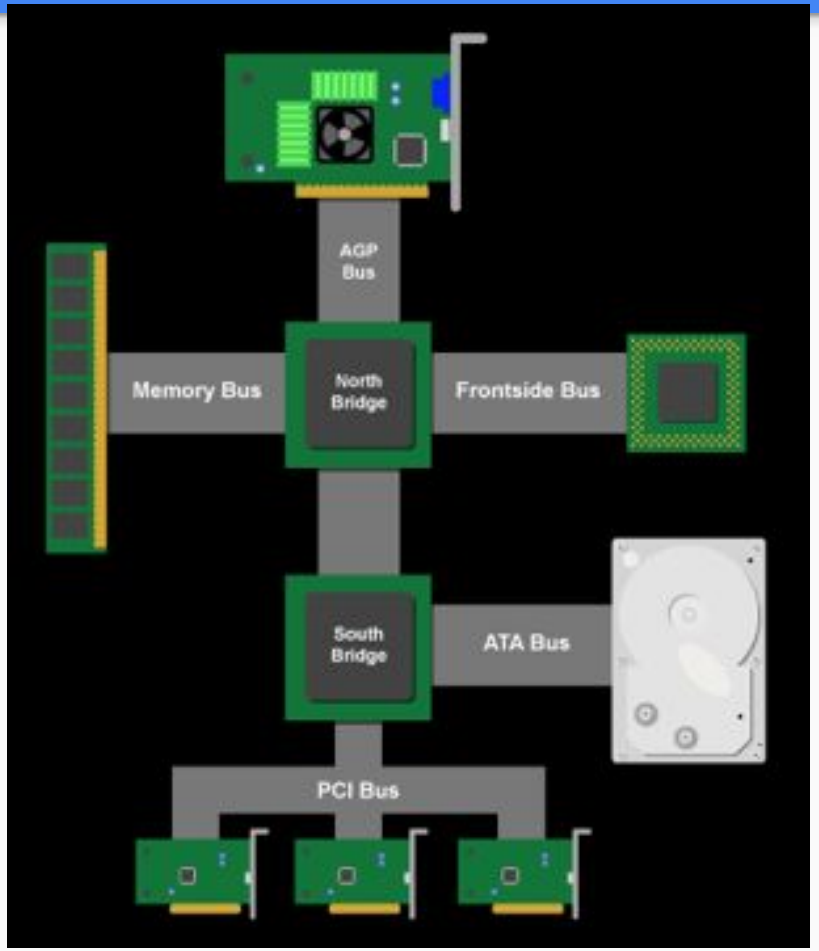


What is in your motherboard?



- CPU slot
- Chipset - manages communication between processor, other components of PC
- Buses
- Expansion slots
- Graphics card
- Memory slots
- Clock generator
- Power connector

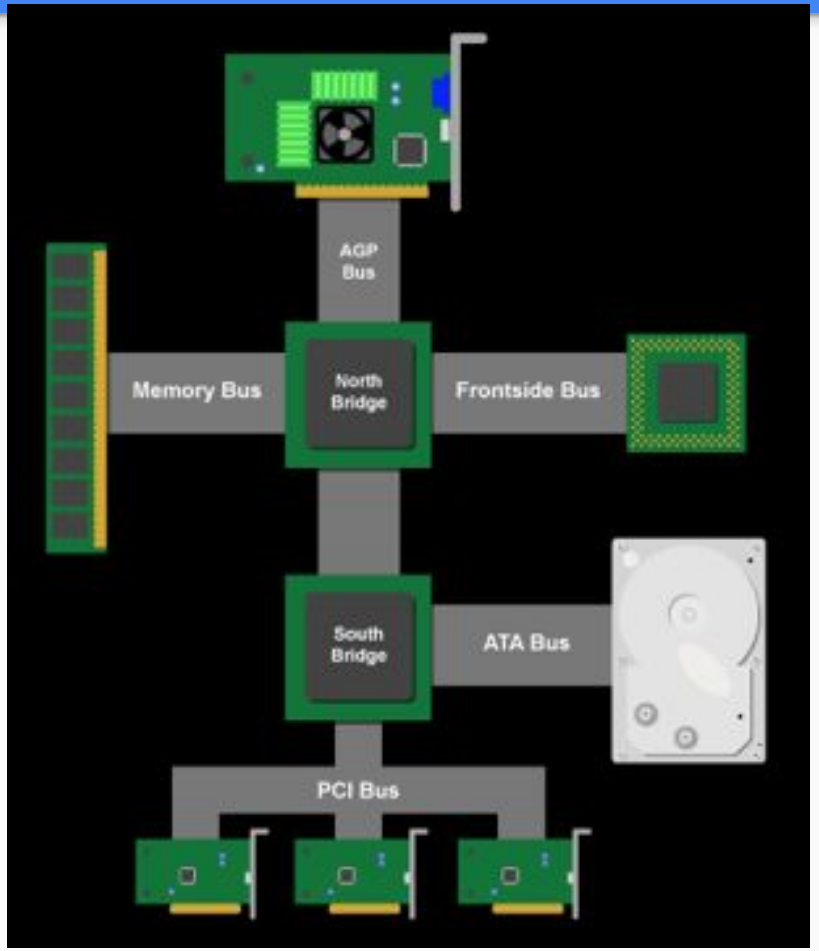
Chipset



- Designed for a given family of processors
- As per legacy, we have a chipset architecture with two bridges
  - Northbridge
  - Southbridge
- Think of these as “routers” moving data from one bus to another

http://arstechnica.com/articles/paedia/hardware/pcie.ars

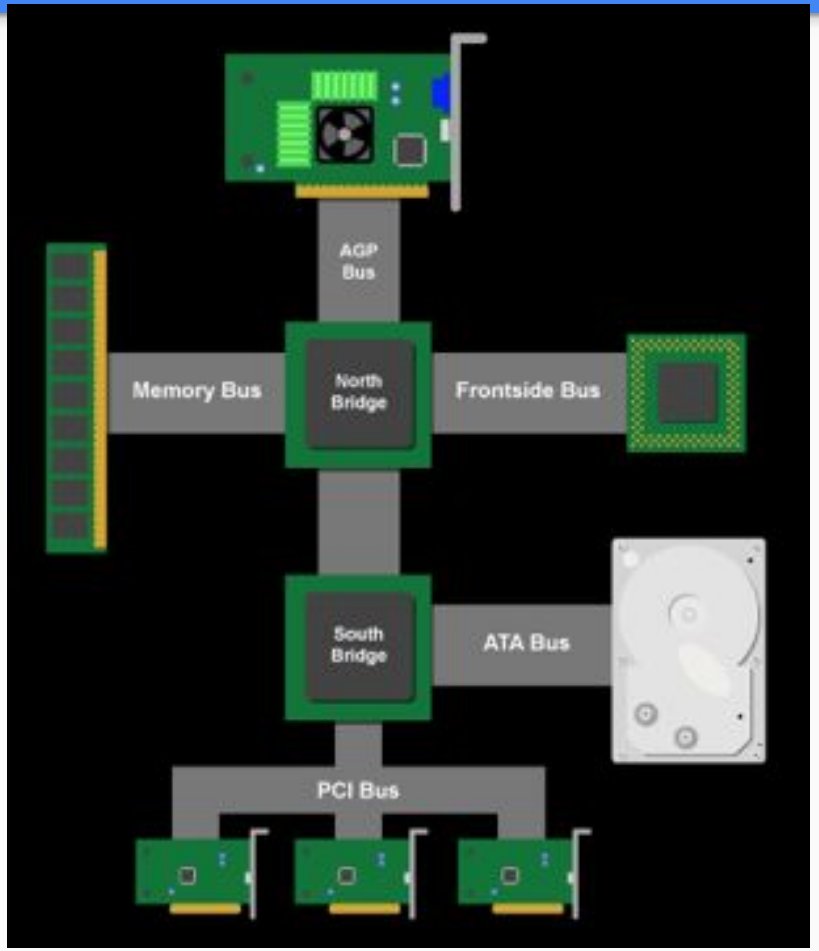
Chipset



- Northbridge
  - Connects the CPU to high speed components such as
    - DRAM
    - Video
  - NB-CPU bus is called Front Side Bus (FSB)
  - Earlier graphics/video cards were connected to Northbridge

http://arstechnica.com/articles/paedia/hardware/pcie.ars

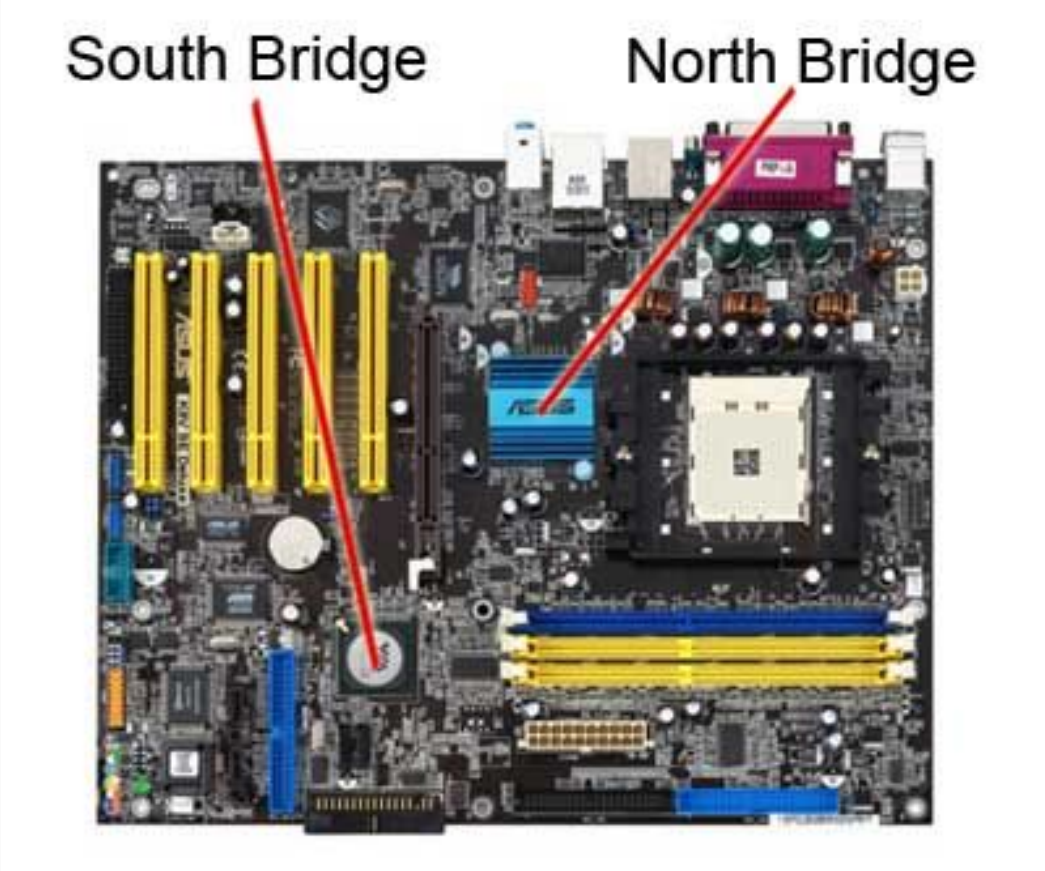
Chipset



- Southbridge
  - Acts as a hub for low-speed devices (mostly I/O) such as
    - USB
    - SATA
    - PCI bus
  - Connects to the CPU through link to Northbridge
    - Example: Intel link has 266MB/s

http://arstechnica.com/articles/paedia/hardware/pcie.ars

Source: howstuffworks.com

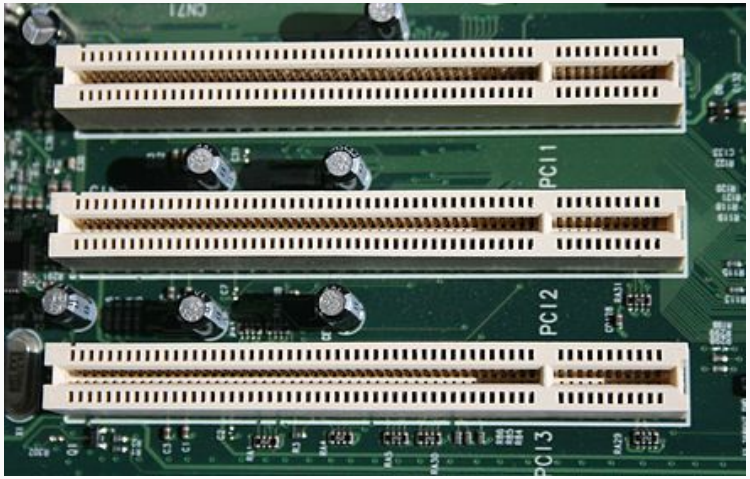


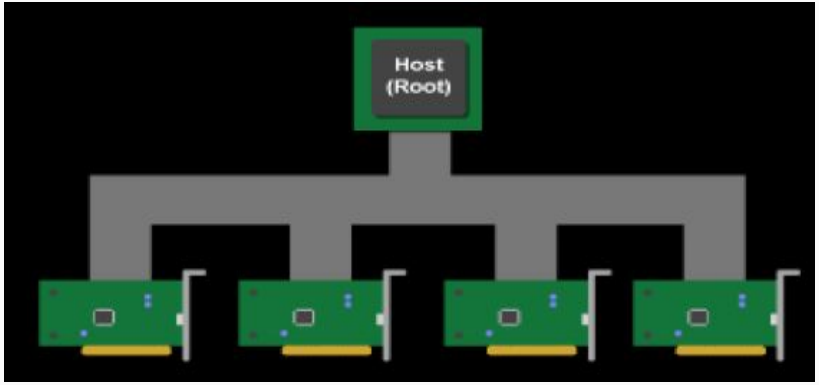
Buses

- Designed to be interoperable: Different devices and different buses
  - Need bus protocols
  - Sometimes requires pin mapping decoder chip

- Buses can be synchronous or asynchronous
- Synchronous bus is driven by a line from an oscillator
  - All transfers take place on edges of this line
  - Example: FSB, Memory
- Asynchronous bus is based on a handshake protocol
  - Example: Universal Serial Bus (USB)

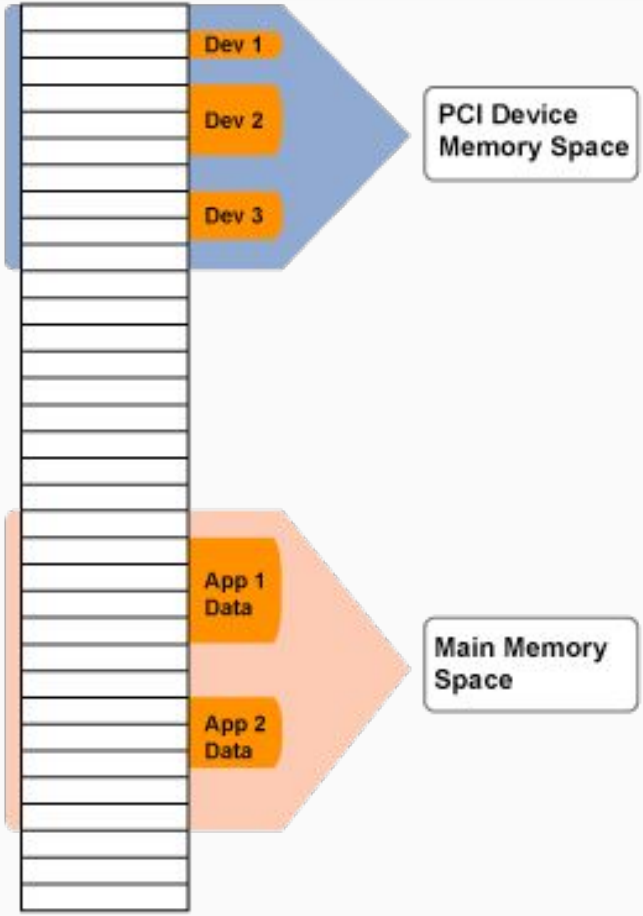
The PCI Bus





- PCI stands for Peripheral Component Interconnect - **Shared** parallel bus
- Connected to Southbridge
- Earlier: 33MHz, 32-bit wide, 132 MB/s theoretical peak transfer rate
- Now: 66 MHz, 64-bit wide, 528 MB/s
- Attach multiple devices which contend to become master by arbitration
- The winner gets to connect to CPU/DRAM via both bridges

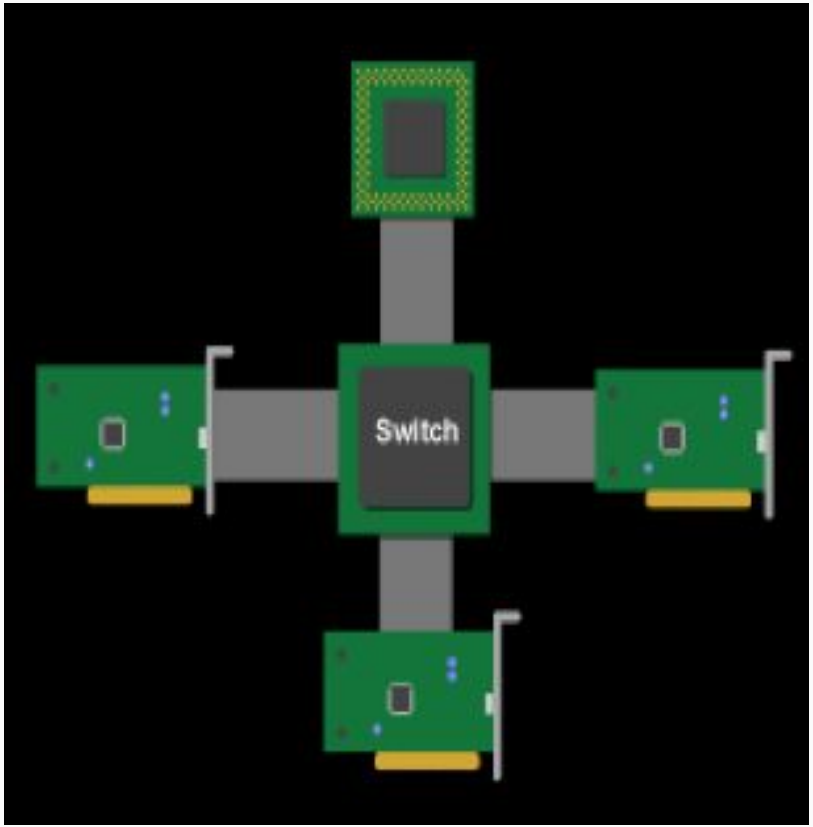
Memory mapping for PCI



- PCI devices are memory mapped I/O
- The device registers of PCI devices are mapped on to the physical address space of the CPU
- This assignment happens at boot time

http://arstechnica.com/articles/paedia/hardware/pcie.ars

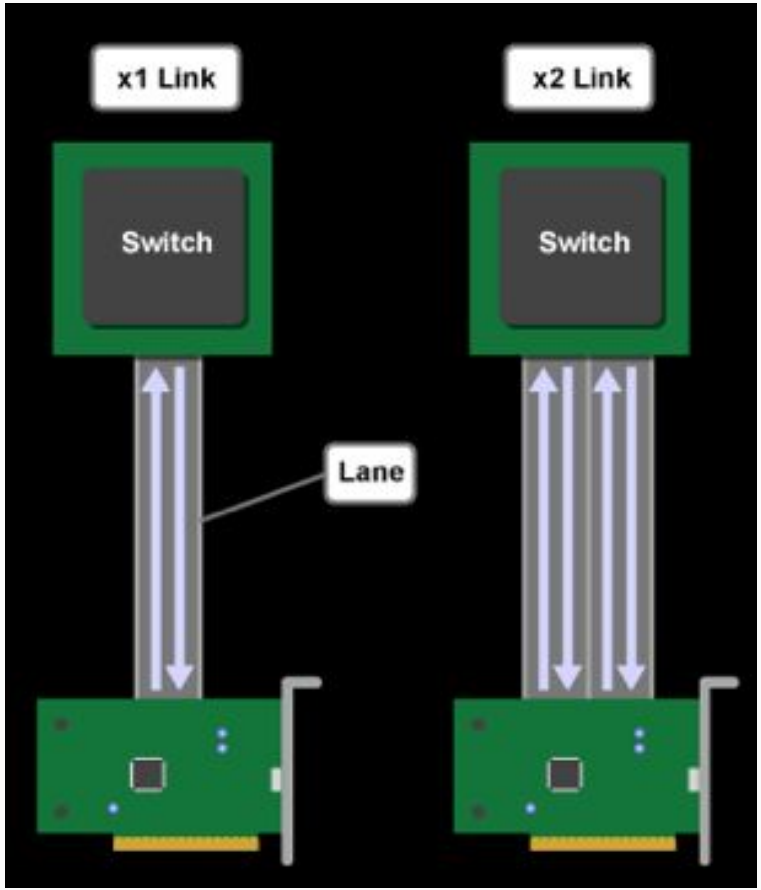
PCI-Express PCI-e



- Focus on supporting high performance peripherals
- Point-to-point serial bus. Packet switched
- Switch creates a virtual channel between device - no arbitration
- Can support 1, 2, 4, 8, or 16 lanes
- Latest version: 1969 MB/s (duplex) per lane

http://arstechnica.com/articles/paedia/hardware/pcie.ars

PCI-e lanes



- Link of a device to a switch can reserve more than one lane
- Each lane supports 1-bit simultaneous upstream and downstream
- Thus, each device has a guaranteed bandwidth depending on number of reserved lanes
    - Necessary for applications with QoS, eg. real-time video streaming

http://arstechnica.com/articles/paedia/hardware/pcie.ars

8b/10b encoding

- We do not want to send too many consecutive 0’s and 1’s. Why?

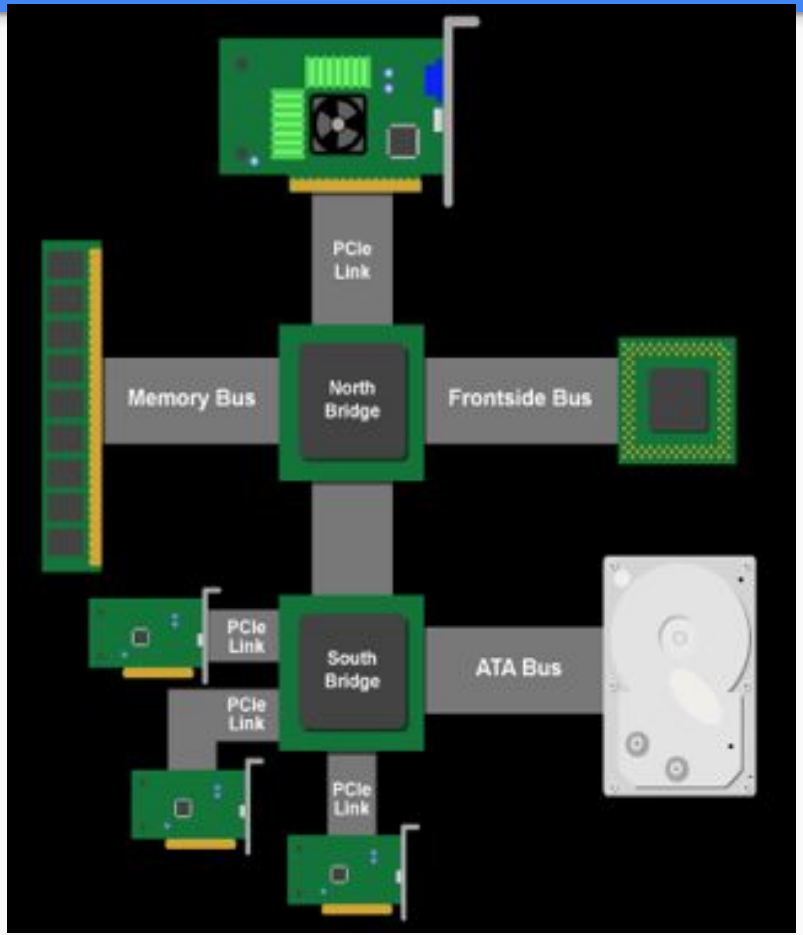
8b/10b encoding

- We do not want to send too many consecutive 0’s and 1’s. Why?
- So patterns like 0000 or 1111 are bad
- Typical physical layers specify constraints like
  - The maximum difference between 0s and 1s in any 20 consecutive characters <= 2
  - In no case can there be more than 5 consecutive bits of the same value
- What can we do?

8b/10b encoding

- We do not want to send too many consecutive 0’s and 1’s. Why?
- So patterns like 0000 or 1111 are bad
- Typical physical layers specify constraints like
  - The maximum difference between 0s and 1s in any 20 consecutive characters <= 2
  - In no case can there be more than 5 consecutive bits of the same value
- What can we do?
- Encode every 8 bits sequence (256 possibilities) to 10 bits (1024 possibilities) chosen to have the above properties

PC architecture based on PCIe

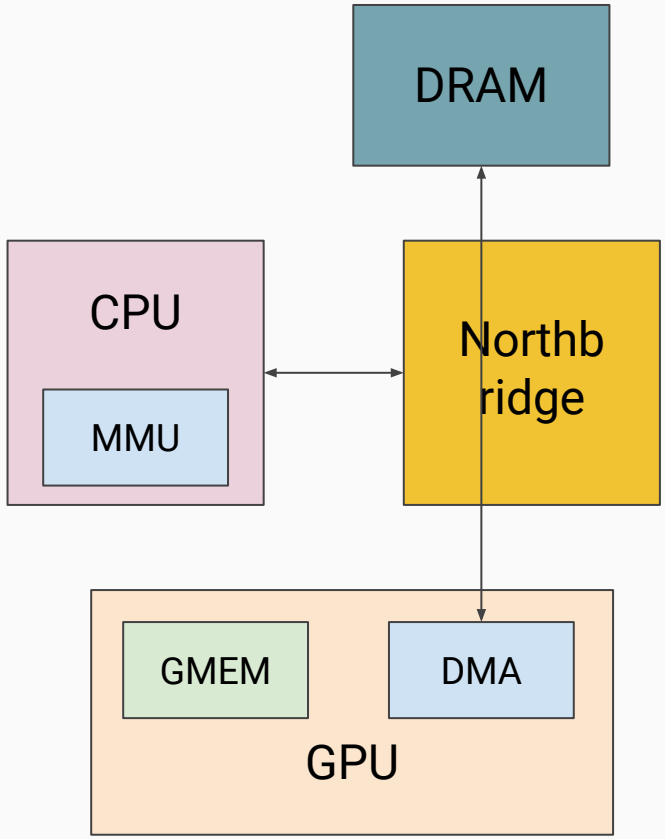


- The base architecture is PCIe
- Both NB and SB are PCIe switches
- Compatible with slower PCI devices with a PCI-PCIe bridge

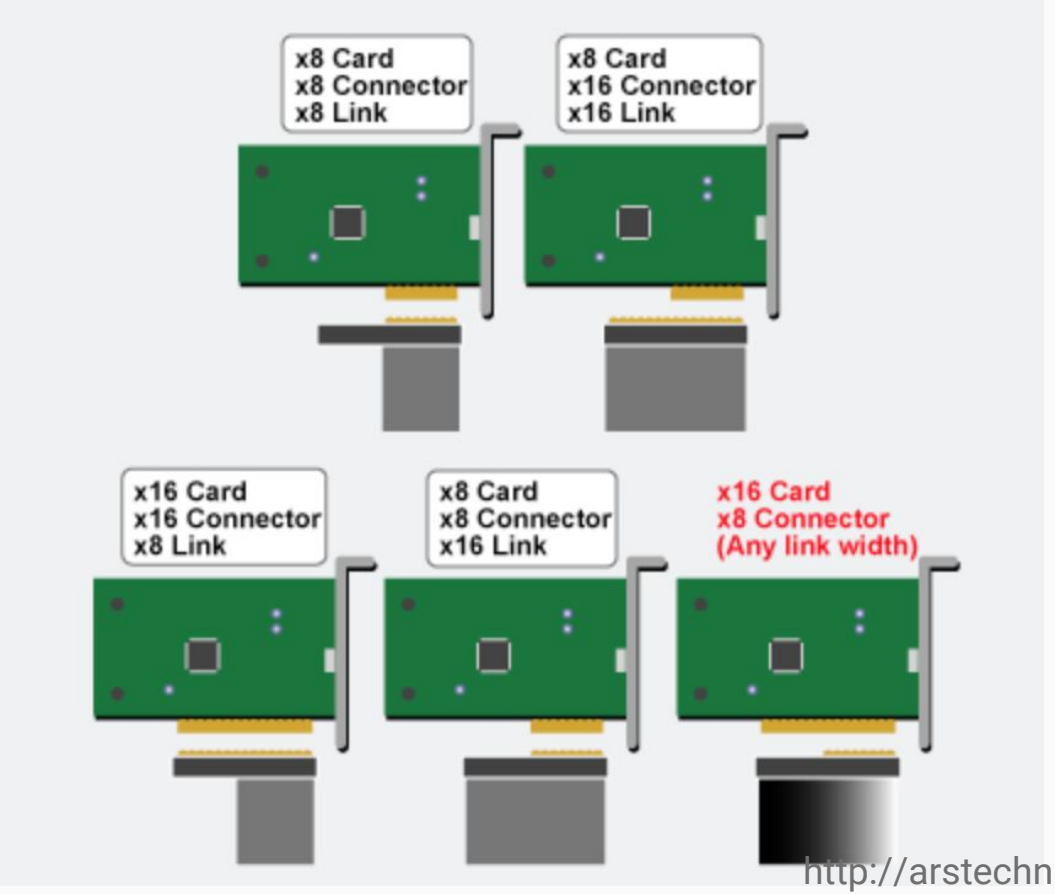
http://arstechnica.com/articles/paedia/hardware/pcie.ars

DMA access to main memory

- Instead of the CPU co-ordinating the memory transfer to/from, it can program the Direct Memory Access (DMA) engine instead
- This works when physical addresses can be provided
  => pinned memory
- If not pinned, then it needs to be copied to a pinned location
  => More overhead for MMU, CPU



Lane negotiation



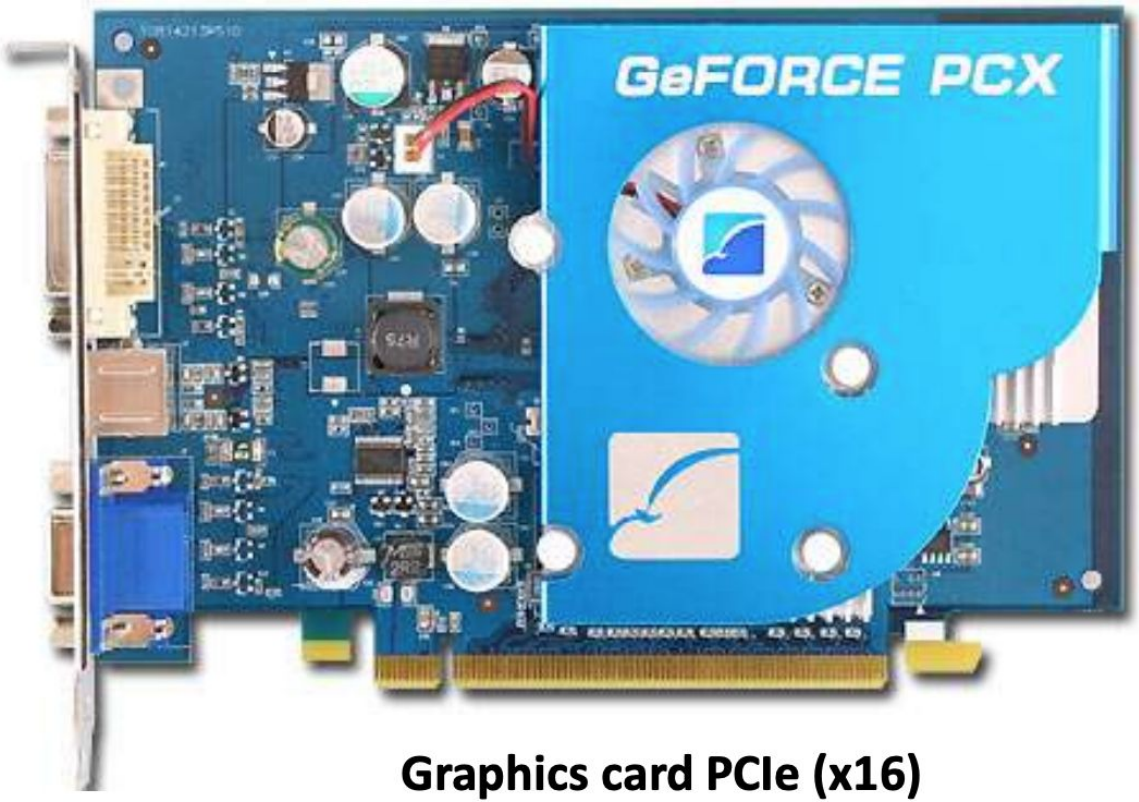
- Depending on the size of the card (number of copper lanes), it will fit on a specific connector
- However, at bootup, the devices negotiate how many **links** they receive

http://arstechnica.com/articles/paedia/hardware/pcie.ars

SLI / Crossfire

- These are for inter-GPU communication
- Scalable Link Interface (SLI) by Nvidia
  - Enables up to 1GB/s communication between 2, 3, or 4 GPU cards
  - Consumes no bandwidth on PCIe
- Similar technology from AMD - Crossfire

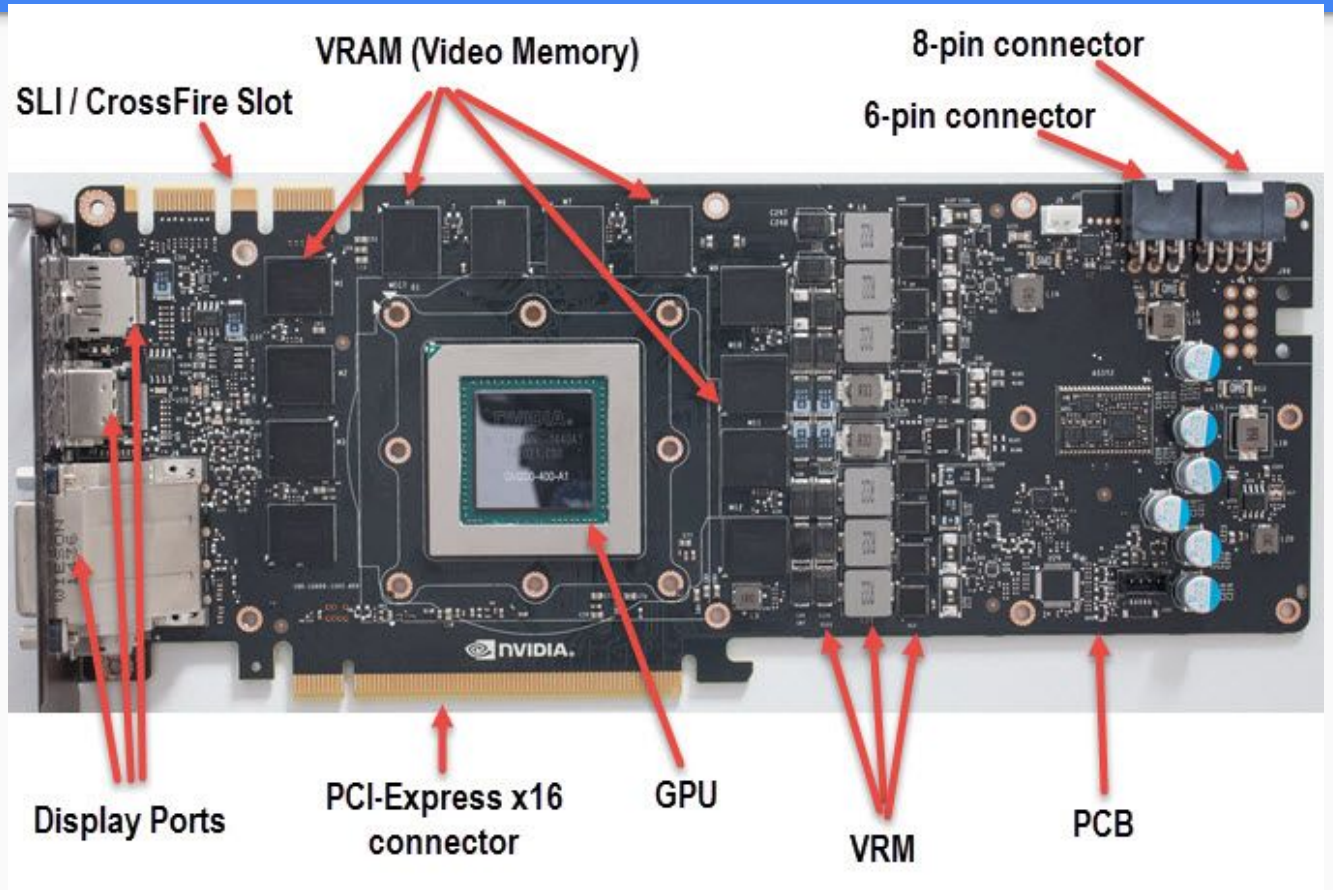
An older card



Graphics card PCIe (x16)

Source: National Instrument

A more recent card



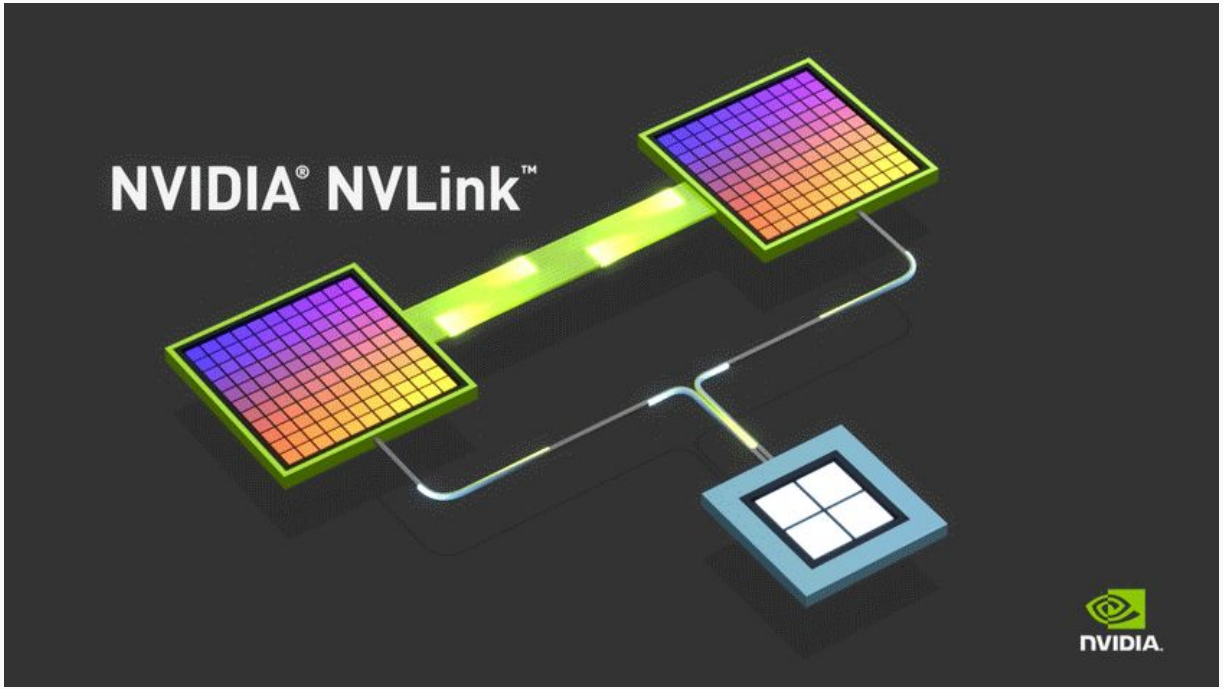
https://graphicscardhub.com/graphics-card-component-connectors/

Multi-GPU setup

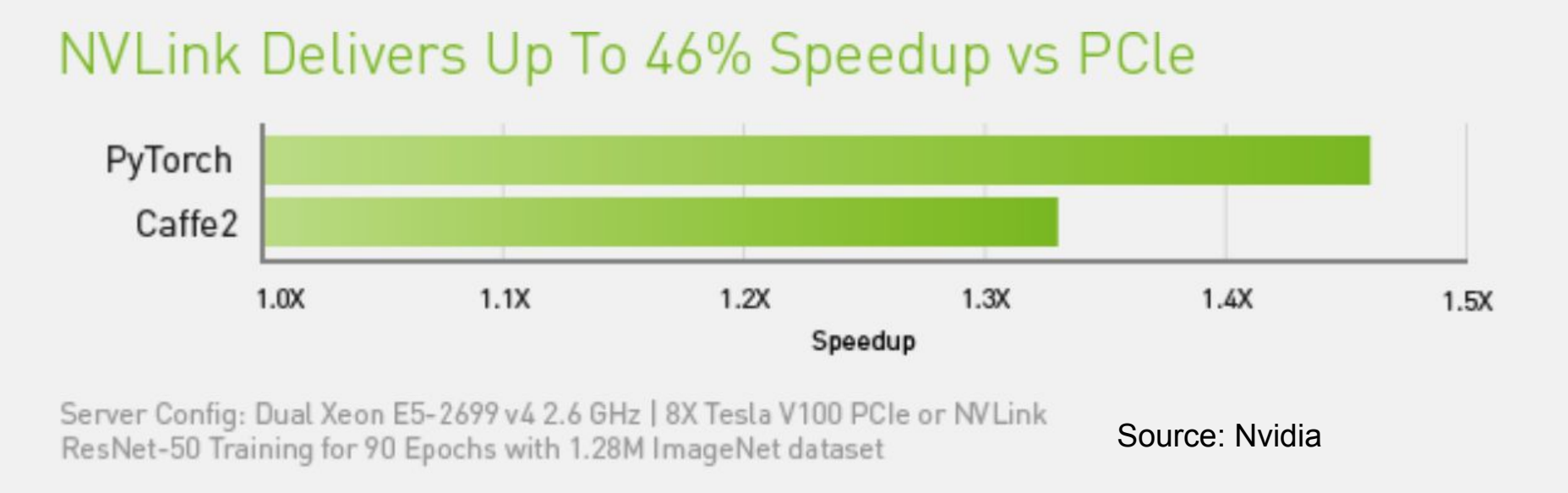
- High end applications (eg. autonomous driving, distributed DL training) require multiple GPUs
- Sometimes memory available in one GPU is a constraint
- Interfacing each GPU to a (separate) CPU server would be very expensive => Identify GPU-GPU communication

NVLink

- An upgrade on SLI from Pascal series onwards
- Developed by Nvidia for GPU-GPU communication without burdening PCIe



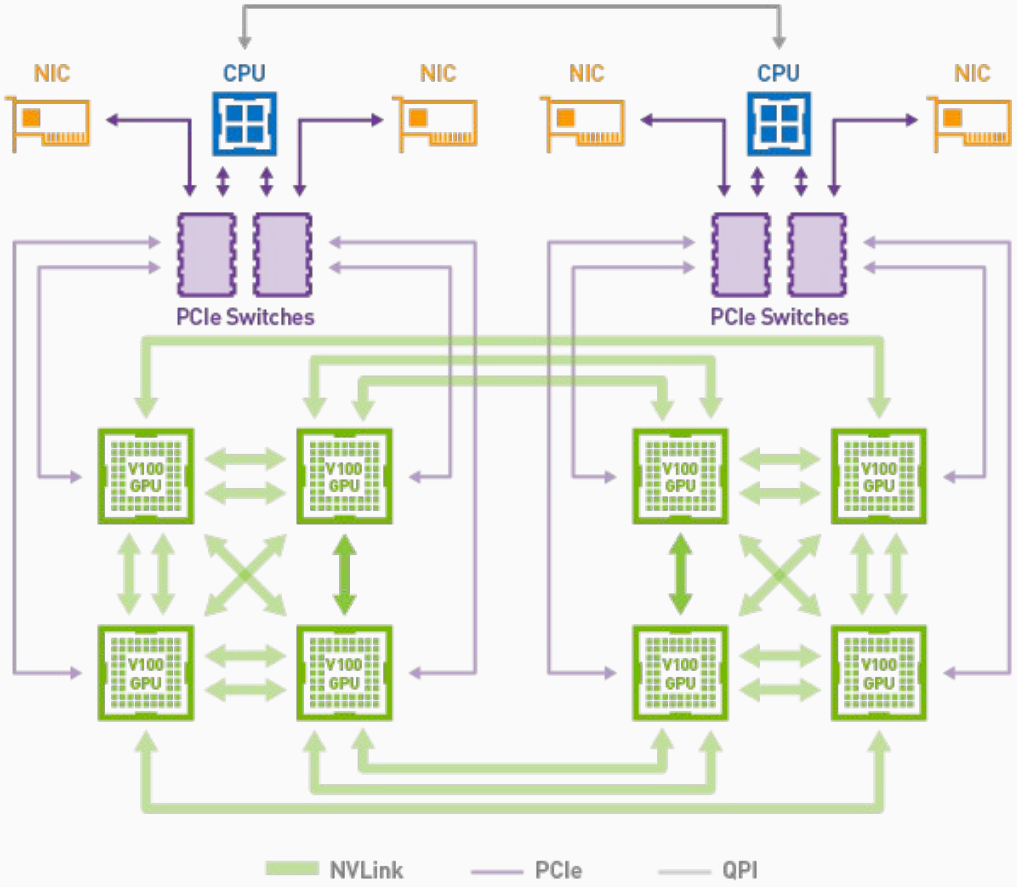
NVLink



Source: Nvidia

- NVLink can support 5x-12x higher throughput than PCIe

NVFabric


NIC
CPU
NIC
NIC
CPU
NIC
PCIe Switches
PCIe Switches
V100 GPU
V100 GPU
V100 GPU
V100 GPU
V100 GPU
V100 GPU
V100 GPU
V100 GPU
NVLink
PCIe
QPI

Multi-GPU Programming

- Multiple GPU devices can be connected to one CPU
  - Need not be the same card
  - Need not even support the same CUDA compute level
- There is one device identified to be **current** to which all CUDA api calls go to
- CUDA api call to change current device: `cudaSetDevice(devId)`
- Changing device can happen interleaved between async calls

```
cudaSetDevice(0);
cudaMemcpyAsync();myKernel<< ... >> ();cudaMemcpyAsync();
cudaSetDevice(1);
cudaMemcpyAsync();myKernel<< ... >> ();cudaMemcpyAsync();
```

GPU-GPU communication

- `cudaDeviceEnablePeerAccess (devId, 0)`
  - Enables current device to dereference pointers on device `devId`
- `cudaDeviceCanAccessPeer (&val, dev1, dev2)`
  - Checks whether dev1 and dev2 can have peer-to-peer communication
- `cudaMemcpyPeerAsync (void* dst_add, int dst_dev, void* src_add, int src_dev, size_t n, cudaStream_t stream)`
  - Queued to stream which runs on some device
  - Performance is better if the stream is assigned to the src_dev
  - CPU is not involved if peer access is available