GPU Architecture

Alan Gray

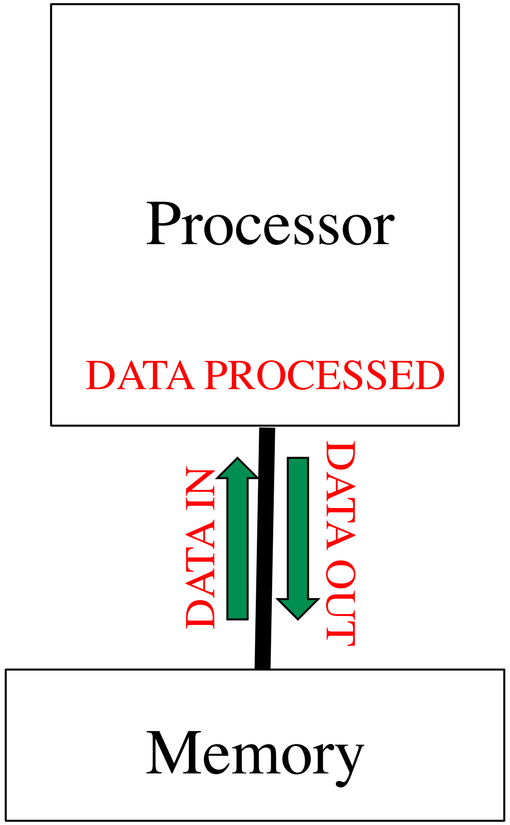
EPCC

The University of Edinburgh

Outline

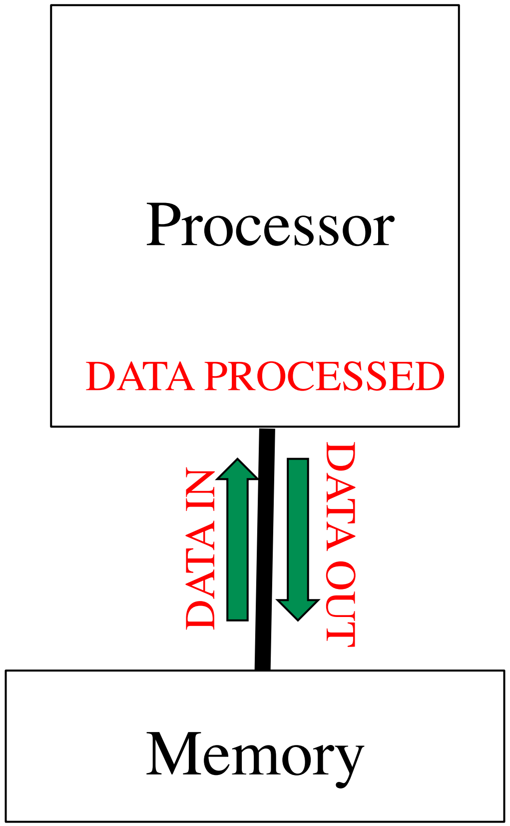
- Why do we want/need accelerators such as GPUs?
- Architectural reasons for accelerator performance advantages
- Latest GPU Products
  - From NVIDIA and AMD
- Accelerated Systems

4 key performance factors



1. Amount of data processed at one time (*Parallel processing*)
2. Processing speed on each data element (*Clock frequency*)
3. Amount of data transferred at one time (*Memory bandwidth*)
4. Time for each data element to be transferred (*Memory latency*)

4 key performance factors



1. *Parallel processing*
2. *Clock frequency*
3. *Memory bandwidth*
4. *Memory latency*

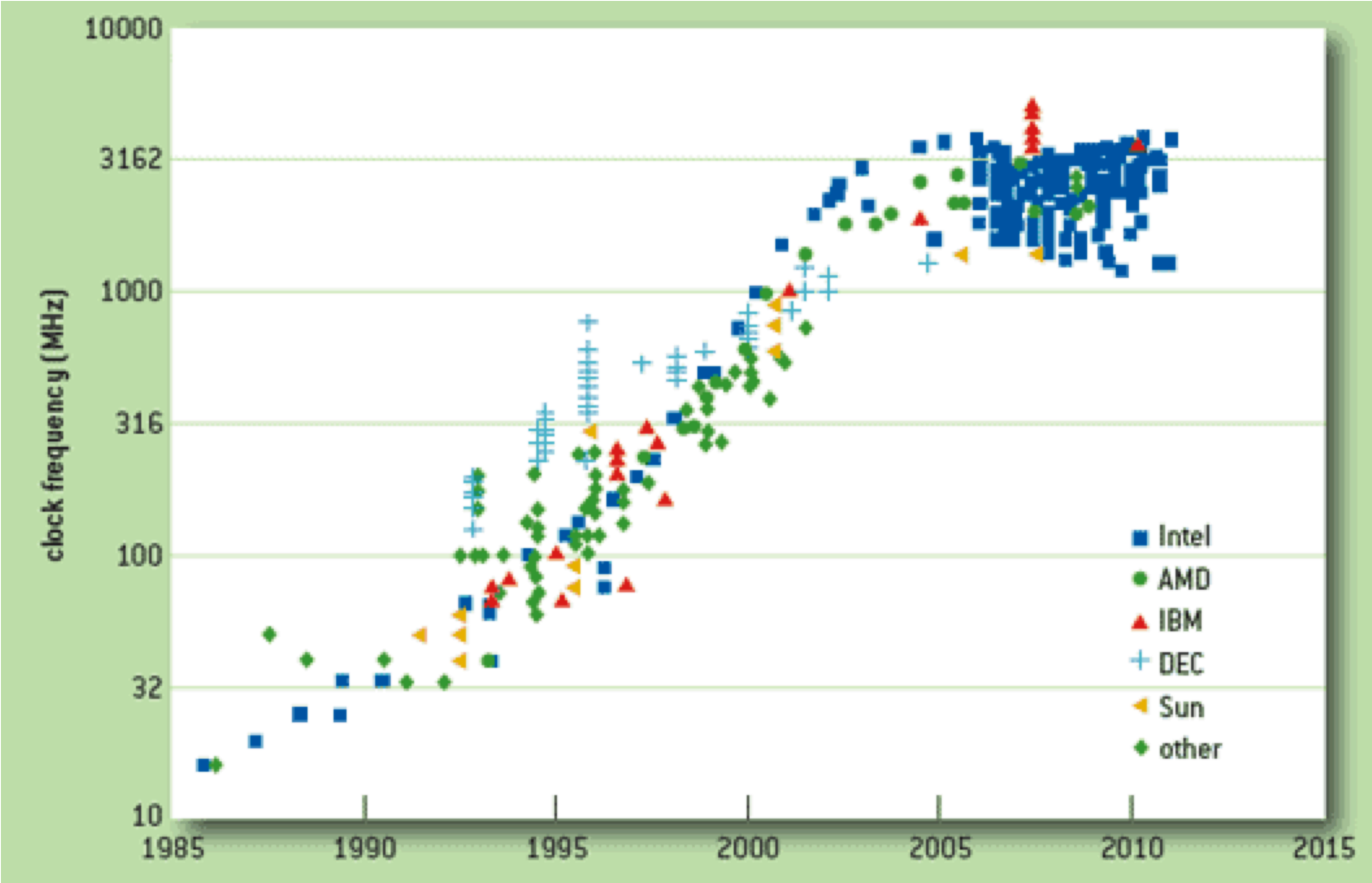
- Different computational problems are sensitive to these in different ways from one another
- Different architectures address these factors in different ways

CPUs: 4 key factors

- *Parallel processing*
  - Until relatively recently, each CPU only had a single core. Now CPUs have multiple cores, where each can process multiple instructions per cycle
- *Clock frequency*
  - CPUs aim to maximise clock frequency, but this has now hit a limit due to power restrictions (more later)
- *Memory bandwidth*
  - CPUs use regular DDR memory, which has limited bandwidth
- *Memory latency*
  - Latency from DDR is high, but CPUs strive to **hide** the latency through:
    - Large on-chip low-latency caches to stage data
    - Multithreading
    - Out-of-order execution

The Problem with CPUs

- The power used by a CPU core is proportional to Clock Frequency x Voltage^2
- In the past, computers got faster by increasing the frequency
  - Voltage was decreased to keep power reasonable.
- Now, voltage cannot be decreased any further
  - 1s and 0s in a system are represented by different voltages
  - Reducing overall voltage further would reduce this difference to a point where 0s and 1s cannot be properly distinguished



Reproduced from http://queue.acm.org/detail.cfm?id=2181798

The Problem with CPUs

- Instead, performance increases can be achieved through exploiting parallelism
- Need a chip which can perform many parallel operations every clock cycle
  - Many cores and/or many operations per core
- Want to keep power/core as low as possible
- Much of the power expended by CPU cores is on functionality not generally that useful for HPC
  - e.g. branch prediction

Accelerators

- So, for HPC, we want chips with simple, low power, number-crunching cores
- But we need our machine to do other things as well as the number crunching
  - Run an operating system, perform I/O, set up calculation etc
- Solution: “Hybrid” system containing both CPU and “accelerator” chips

Accelerators

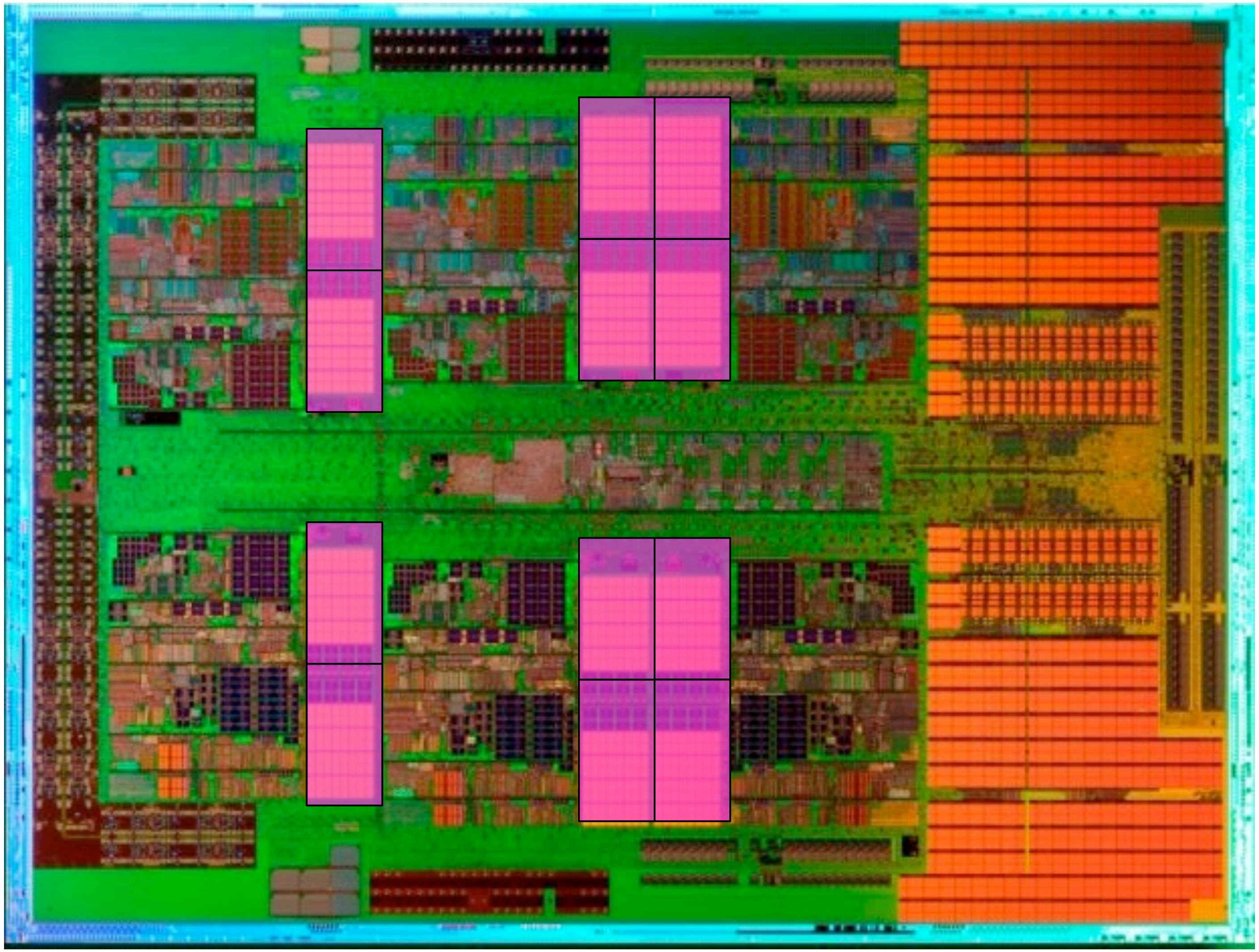
- It costs a huge amount of money to design and fabricate new chips
  - Not feasible for relatively small HPC market
- Luckily, over the last few years, Graphics Processing Units (GPUs) have evolved for the highly lucrative gaming market
  - And largely possess the right characteristics for HPC
    - Many number-crunching cores
- GPU vendors NVIDIA and AMD have tailored existing GPU architectures to the HPC market
- GPUs now firmly established in HPC industry

Intel Xeon Phi

- More recently, Intel have released a different type of accelerator to compete with GPUs for scientific computing
  - Many Integrated Core (MIC) architecture
  - AKA Xeon Phi (codenames Larrabee, Knights Ferry, Knights Corner)
  - Used in conjunction with regular Xeon CPU
  - Intel prefer the term “coprocessor” to “accelerator”
- Essentially a many-core CPU
  - Typically 50-100 cores per chip
  - with wide vector units
  - So again uses concept of many simple low-power cores
    - Each performing multiple operations per cycle
- But latest “Knights Landing (KNL)” is not normally used as an accelerator
  - Instead a self-hosted CPU

AMD 12-core CPU

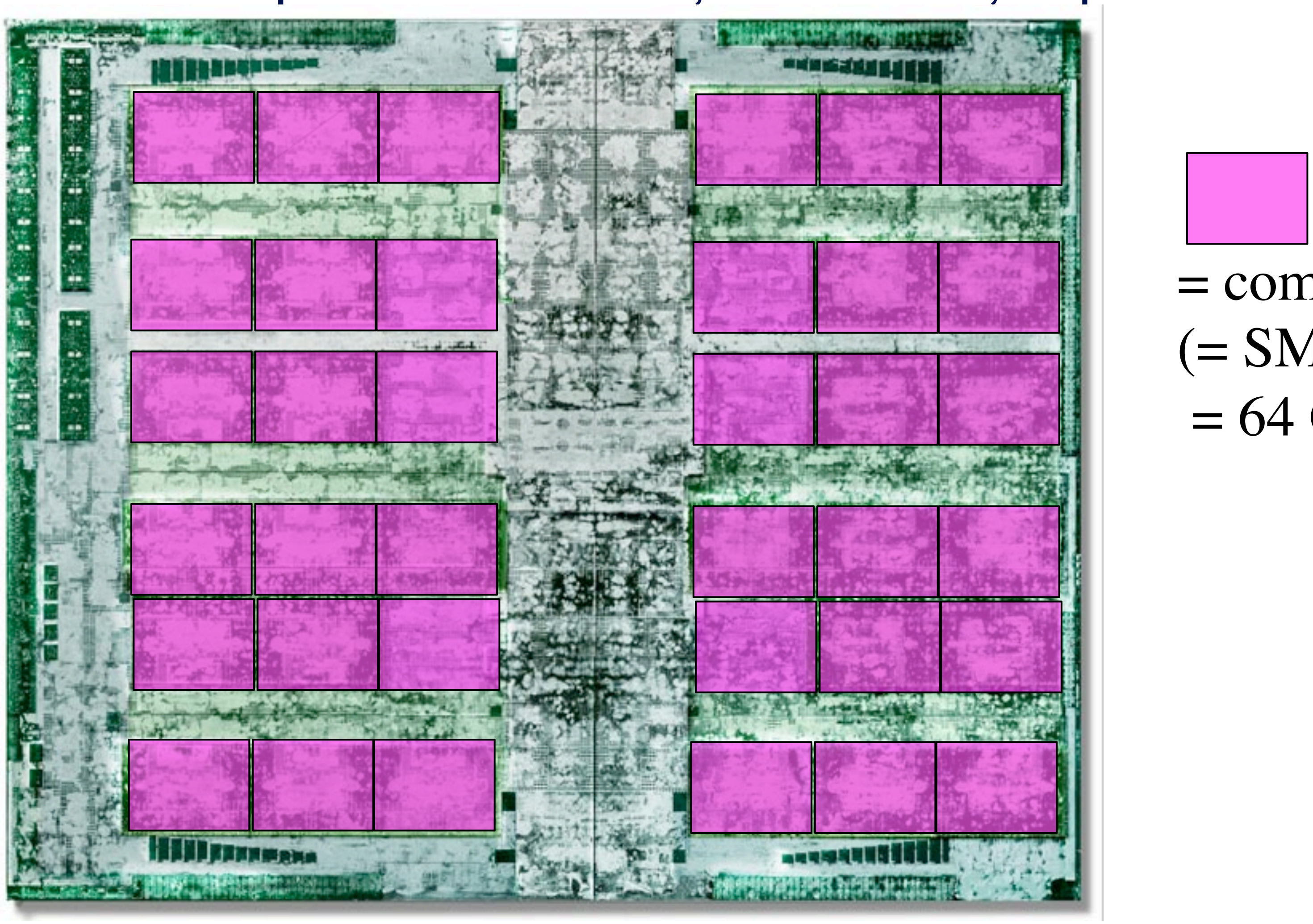
- Not much space on CPU is dedicated to compute



= compute unit
(= core)

NVIDIA Pascal GPU

- GPU dedicates much more space to compute
  - At expense of caches, controllers, sophistication etc

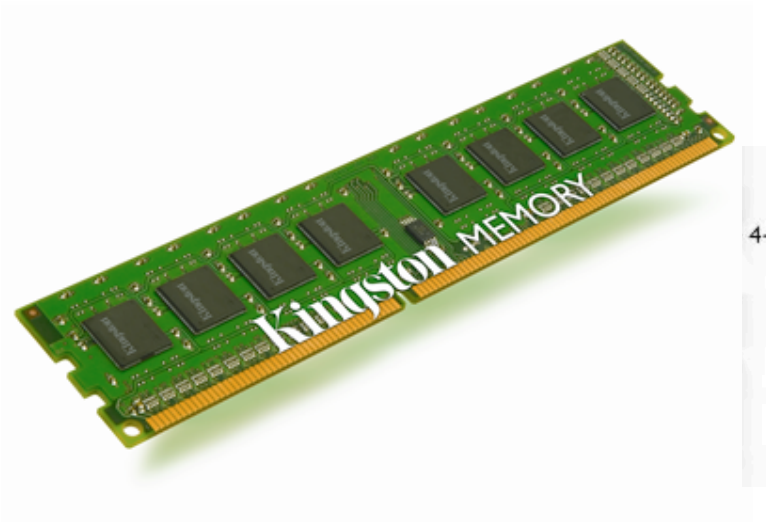


= compute unit
(= SM
 = 64 CUDA cores)

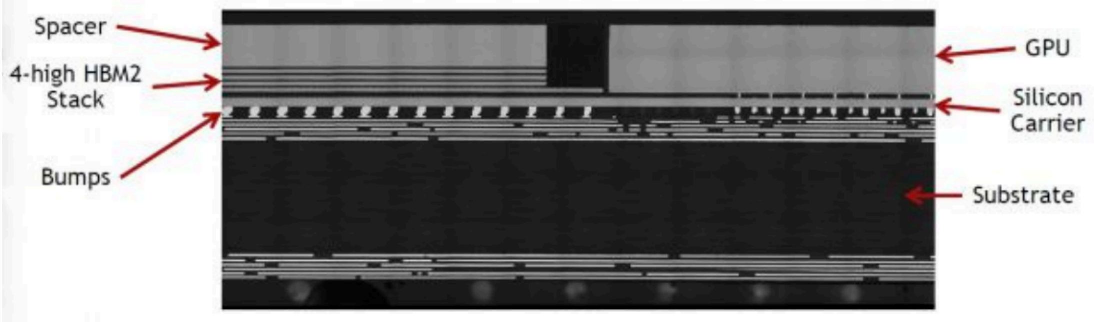
Memory

- For many applications, performance is very sensitive to memory bandwidth
- GPUs use high bandwidth memory

GPUs use Graphics DRAM (GDDR)



CPUs use DRAM



or HBM2 stacked memory (new Pascal P100 chips only)

GPUs: 4 key factors

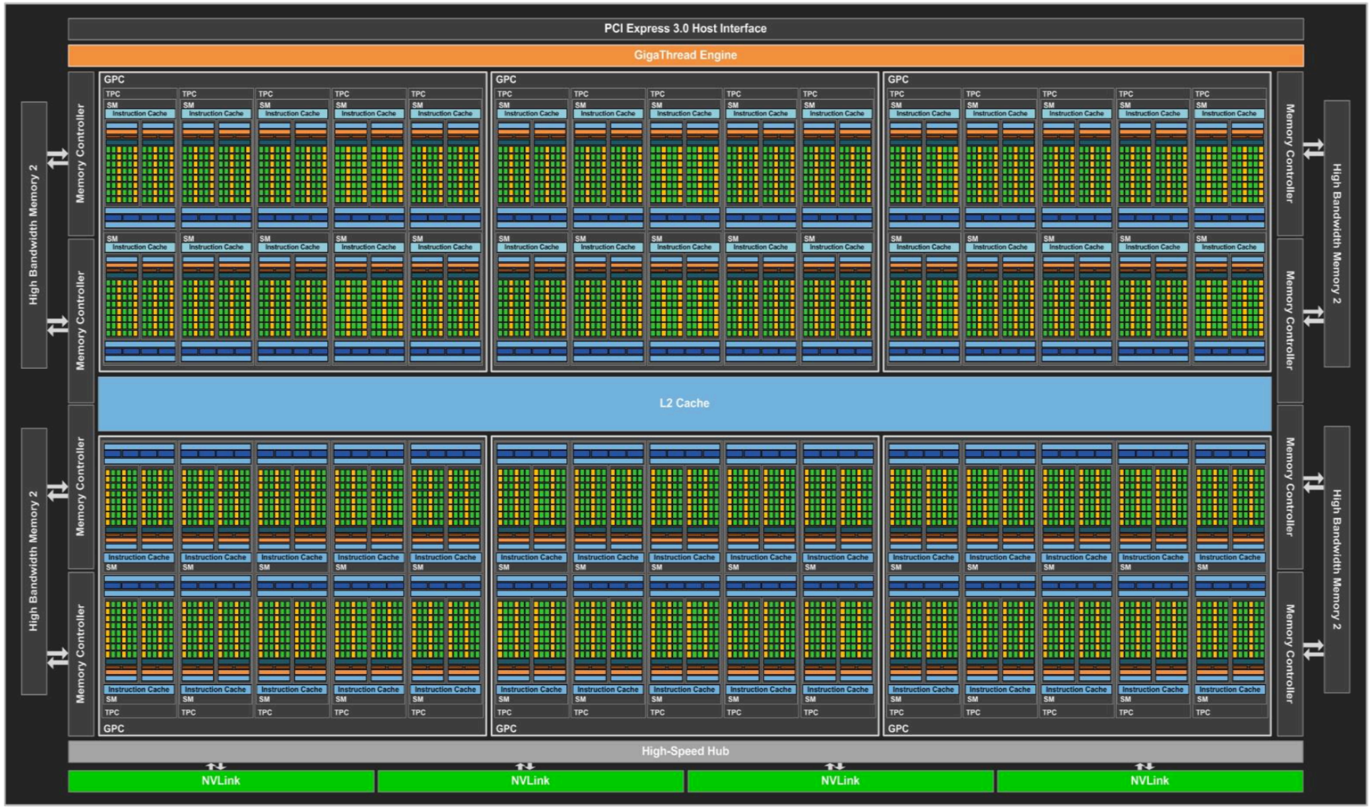
- *Parallel processing*
  - GPUs have a much higher extent of parallelism than CPUs. Many more cores and/or operations per core
- *Clock frequency*
  - GPUs typically have lower clock-frequency than CPUs, and instead get performance through parallelism
- *Memory bandwidth*
  - GPUs use high bandwidth GDDR or HBM2 memory
- *Memory latency*
  - Memory latency from is similar to DDR
  - GPUs hide latency through very high levels of multithreading

Latest Technology

- NVIDIA
  - *Tesla* HPC specific GPUs have evolved from *GeForce* series

- AMD
  - *FirePro* HPC specific GPUs have evolved from (ATI) *Radeon* series

NVIDIA Tesla Series GPU

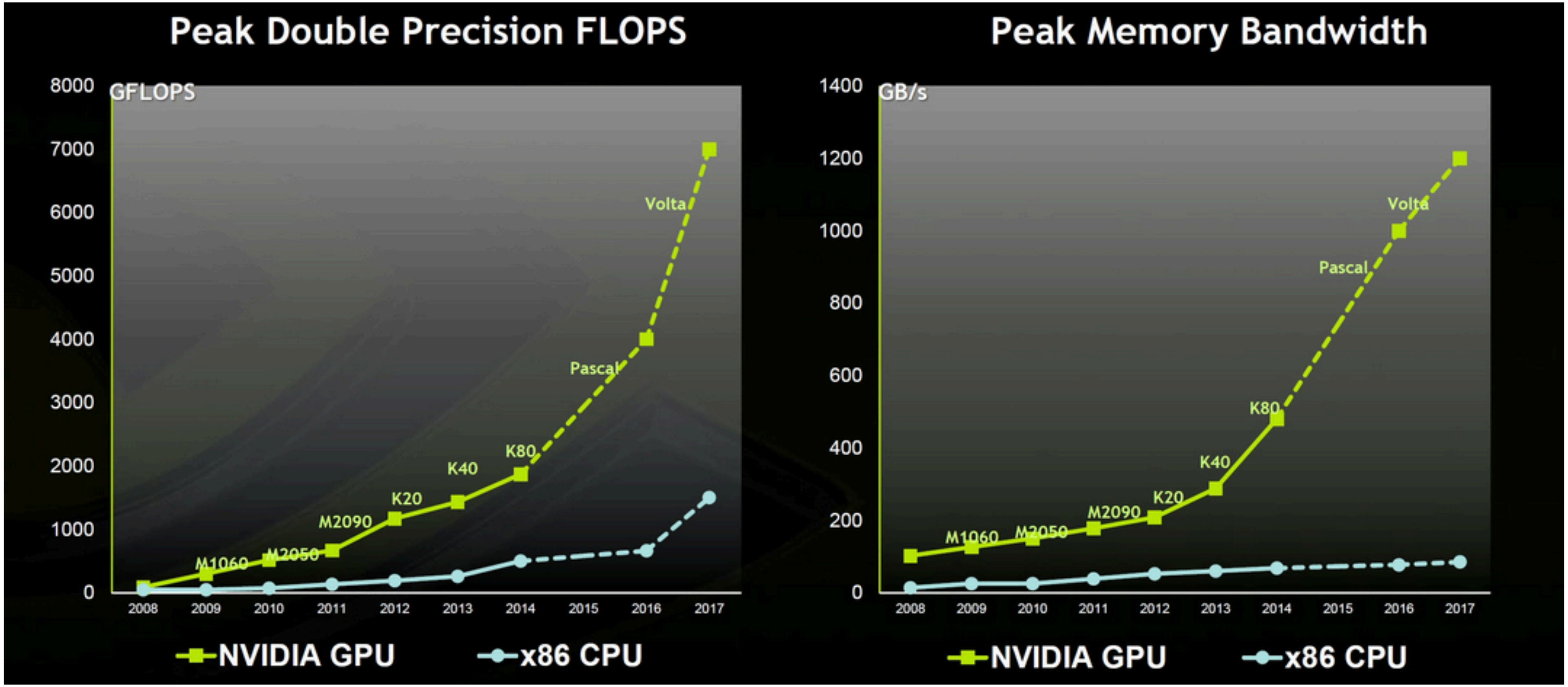


- Chip partitioned into *Streaming Multiprocessors* (SMs) that act independently of each other
- Multiple cores per SM. Groups of cores act in “lock-step”: they perform the same instruction on different data elements
- Number of SMs, and cores per SM, varies across products. High-end GPUs have thousands of cores

NVIDIA SM

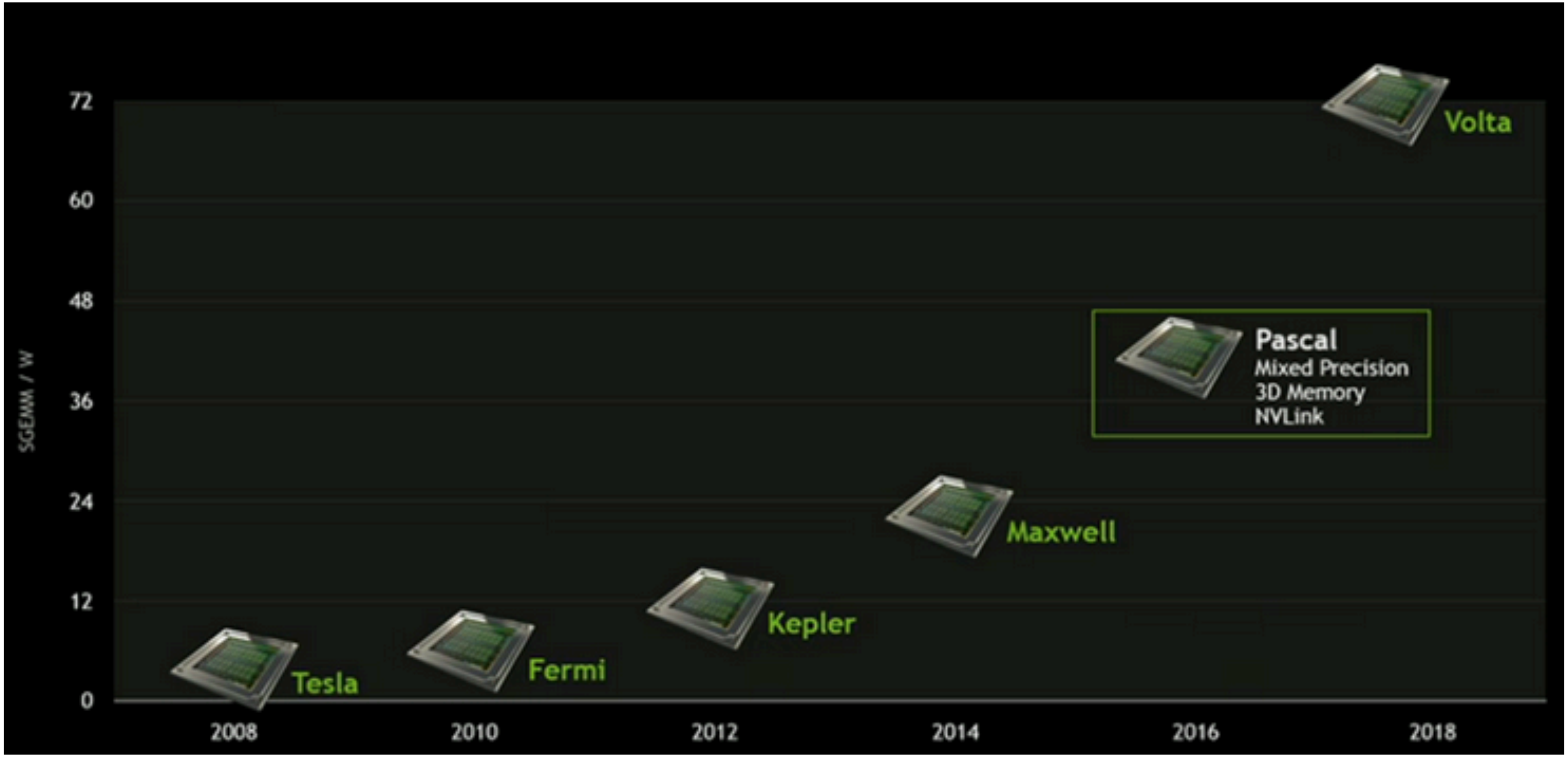

SM
Instruction Cache
Instruction Buffer
Instruction Buffer
Warp Scheduler
Warp Scheduler
Dispatch Unit
Dispatch Unit
Dispatch Unit
Dispatch Unit
Register File (32,768 x 32-bit)
Register File (32,768 x 32-bit)
Core
DP Unit
LD/ST
SFU
Texture / L1 Cache
Tex
Tex
Tex
Tex
64KB Shared Memory

Performance trends



- GPU performance has been increasing much more rapidly than CPU

NVIDIA Roadmap



AMD FirePro

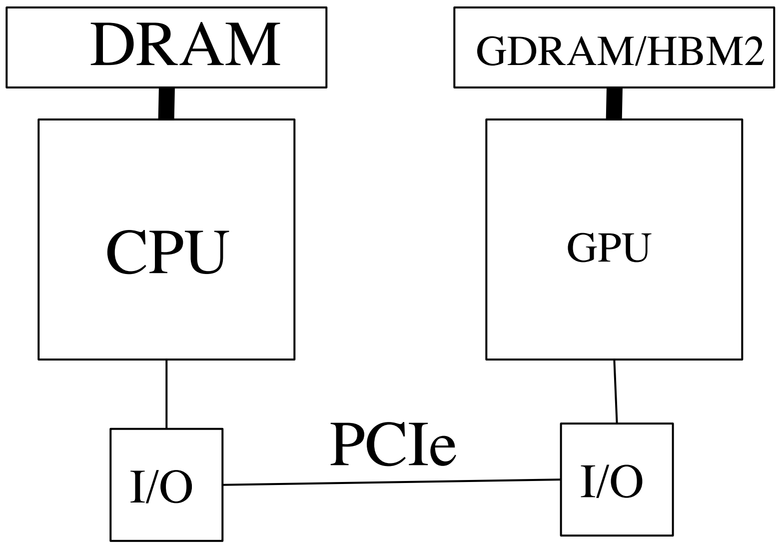
- AMD acquired ATI in 2006
- AMD FirePro series: derivative of Radeon chips with HPC enhancements
- Like NVIDIA, High computational performance and high-bandwidth graphics memory
- Currently much less widely used for GPGPU than NVIDIA, because of programming support issues

Programming GPUs

- GPUs cannot be used *instead* of CPUs
  - They must be used together
  - GPUs act as accelerators
    - Responsible for the computationally expensive parts of the code
- CUDA: Extensions to the C language which allow interfacing to the hardware (NVIDIA specific)
- OpenCL: Similar to CUDA but cross-platform (including AMD and NVIDIA)
- Directives based approach: directives help compiler to automatically create code for GPU. OpenACC and now also new OpenMP 4.0

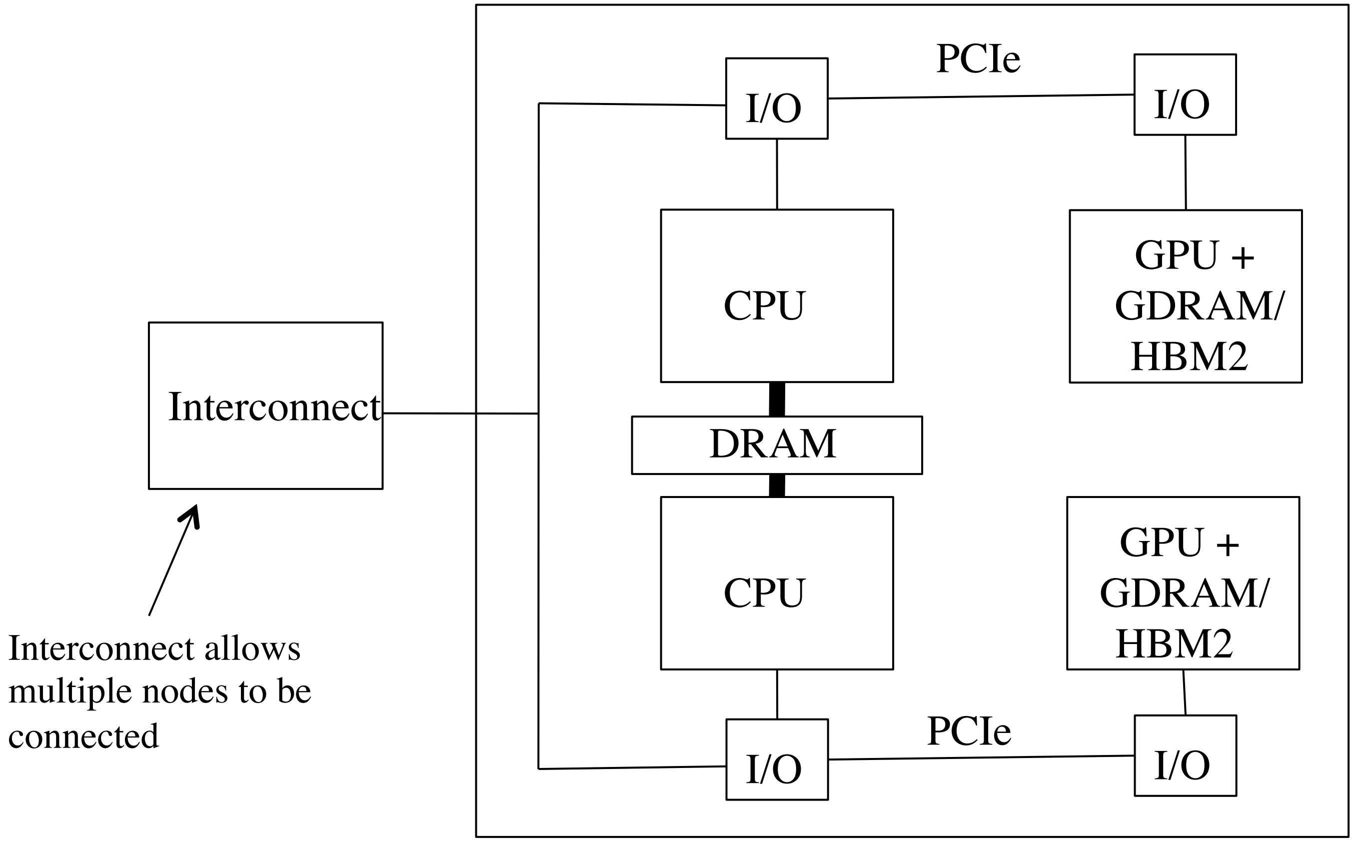
GPU Accelerated Systems

- CPUs and GPUs are used together
  - Communicate over PCIe bus
    - Or, in case of newest Pascal P100 GPUs, NVLINK (more later)

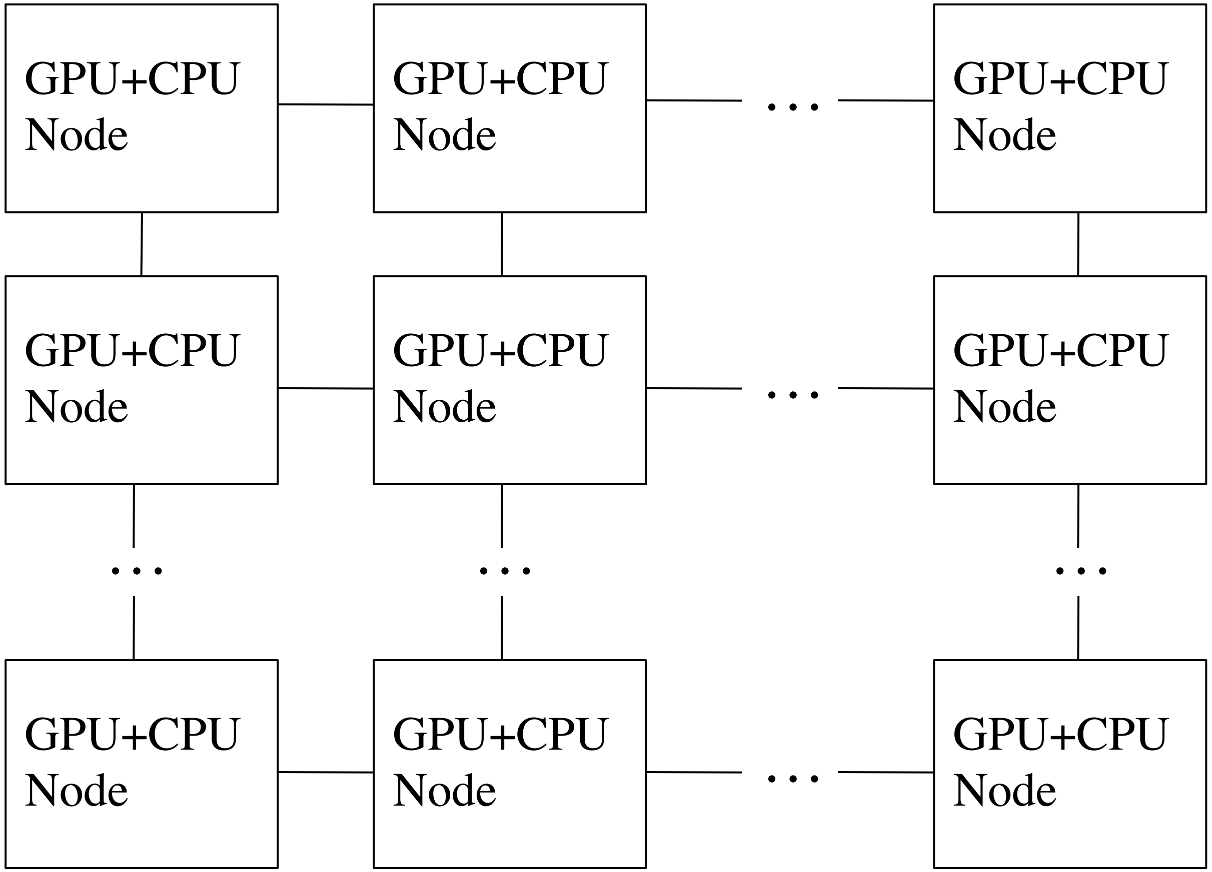


Scaling to larger systems

- Can have multiple CPUs and GPUs within each "workstation" or "shared memory node"
  - E.g. 2 CPUs +2 GPUs (below)
  - CPUs share memory, but GPUs do not



GPU Accelerated Supercomputer



DIY GPU Workstation

- Just need to slot GPU card into PCI-e
- Need to make sure there is enough space and power in workstation

GPU Servers

- Multiple servers can be connected via interconnect

- Several vendors offer GPU Servers
- Example Configuration:
  - 4 GPUs plus 2 (multi-core) CPUs

Cray XK7

- Each compute node contains 1 CPU + 1 GPU
  - Can scale up to thousands of nodes

NVIDIA Pascal

- In 2016 the Pascal P100 GPU was released, with major improvements over previous versions
- Adoption of stacked 3D HBM2 memory as an alternative to GDDR.
  - Several times higher bandwidth
- Introduction of NVLINK: an alternative to PCIe with several-fold performance benefits
  - To closely integrate fast dedicated CPU with fast dedicated GPU
  - CPU must also support NVLINK
    - IBM Power series only at the moment.

Summary

- GPUs have higher compute and memory bandwidth capabilities than CPUs
  - Silicon dedicated to many simplistic cores
  - Use of high bandwidth graphics or HBM2 memory
- Accelerators are typically not used alone, but work in tandem with CPUs
- Most common are NVIDIA GPUs
  - AMD also have high performance GPUs, but not so widely used due to programming support
- GPU accelerated systems scale from simple workstations to large-scale supercomputers