- *Note: the following slides are from the Hardware Architectures course. It is recommended that PPL students read them to help understand the GPU programming lectures.*

Accelerated Architectures

Alan Gray

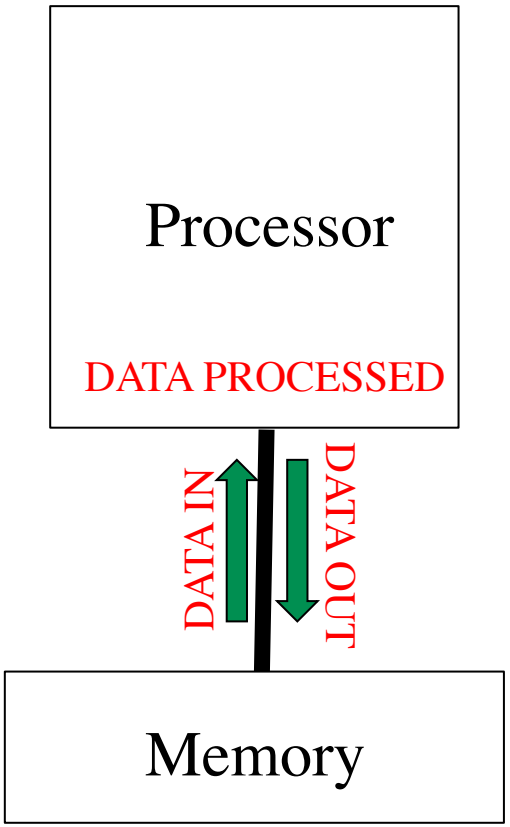
EPCC

The University of Edinburgh

Outline

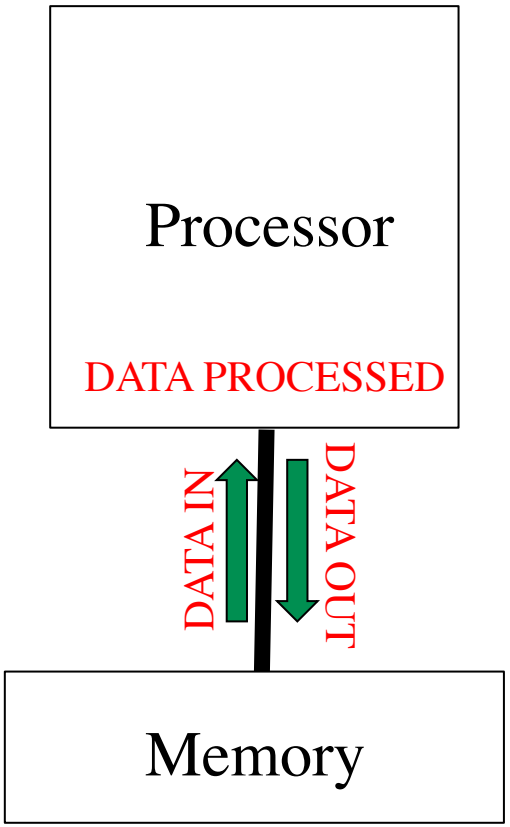
- Why do we want/need accelerators such as GPUs?
- Architectural reasons for accelerator performance advantages
- Latest accelerator Products
  - NVIDIA and AMD GPUs
  - Intel Xeon Phi
- Accelerated Systems

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

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

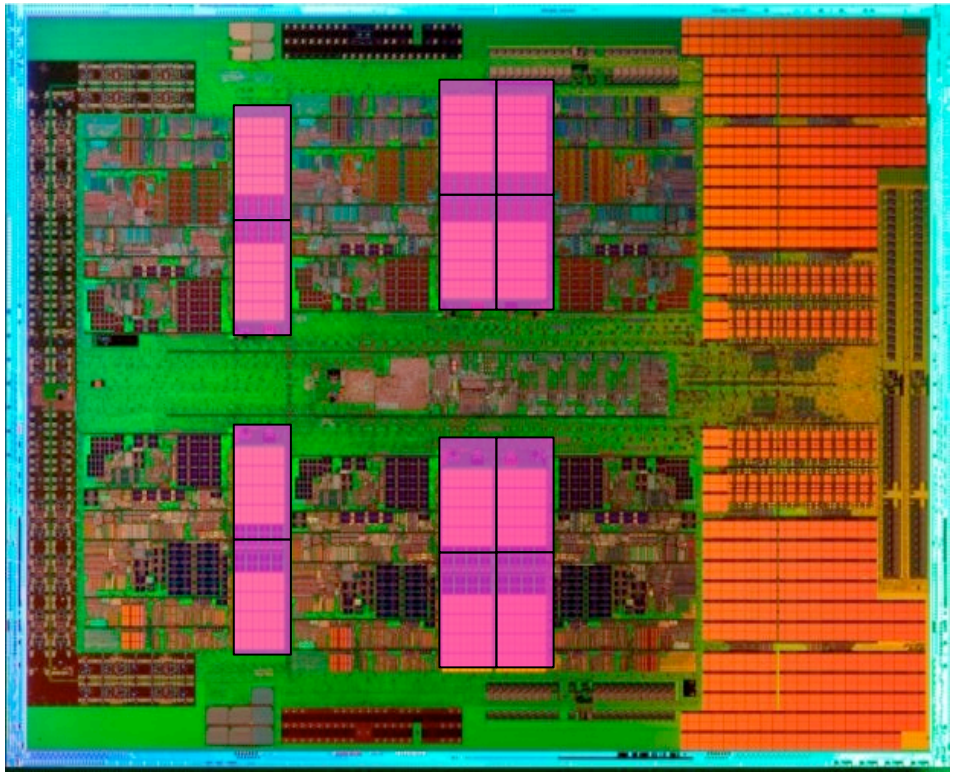
Accelerators

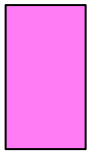
- It costs a huge amount of money to design and fabricate new chips
  - Not feasible for relatively small HPC market
- Luckily, over the last few years, Graphics Processing Units (GPUs) have evolved for the highly lucrative gaming market
  - And largely possess the right characteristics for HPC
    - 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

- 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, Knights Landing)
  - Intel prefer the term “coprocessor” to “accelerator”
- Comprises old Pentium CPU cores from 1993
  - Augmented with wide vector units
- So again uses concept of many simple low-power cores
  - Each performing multiple operations per cycle

AMD 12-core CPU

- Not much space on CPU is dedicated to compute

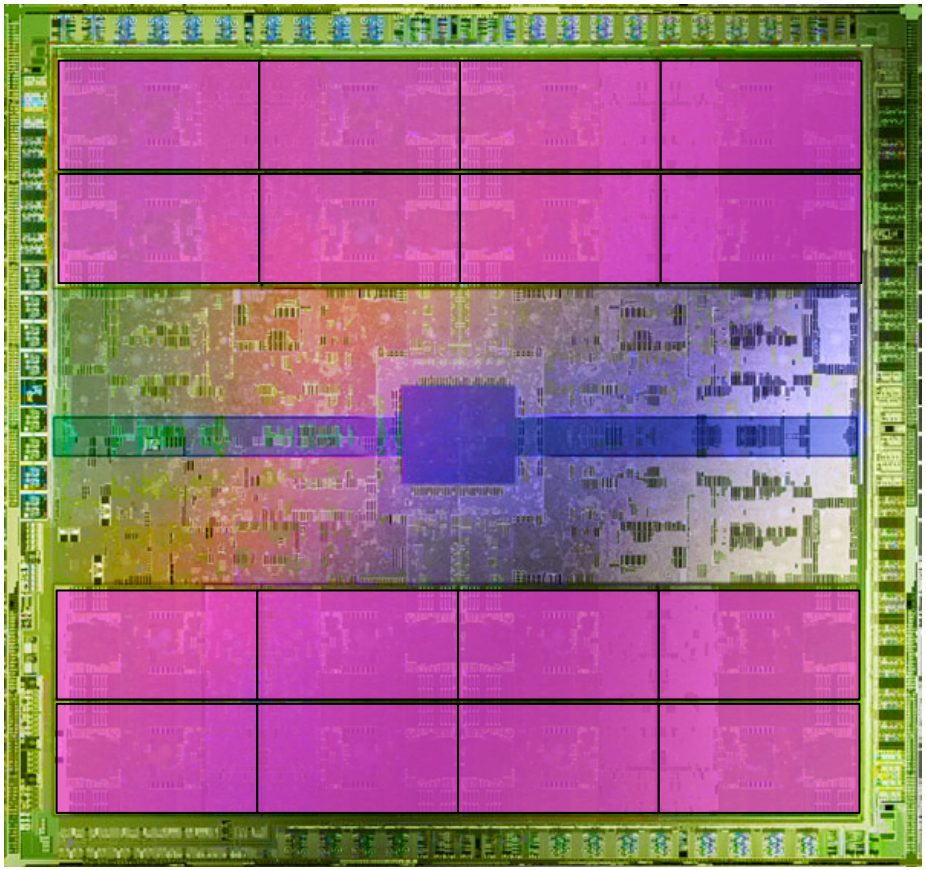




= compute unit
(= core)

NVIDIA Fermi GPU

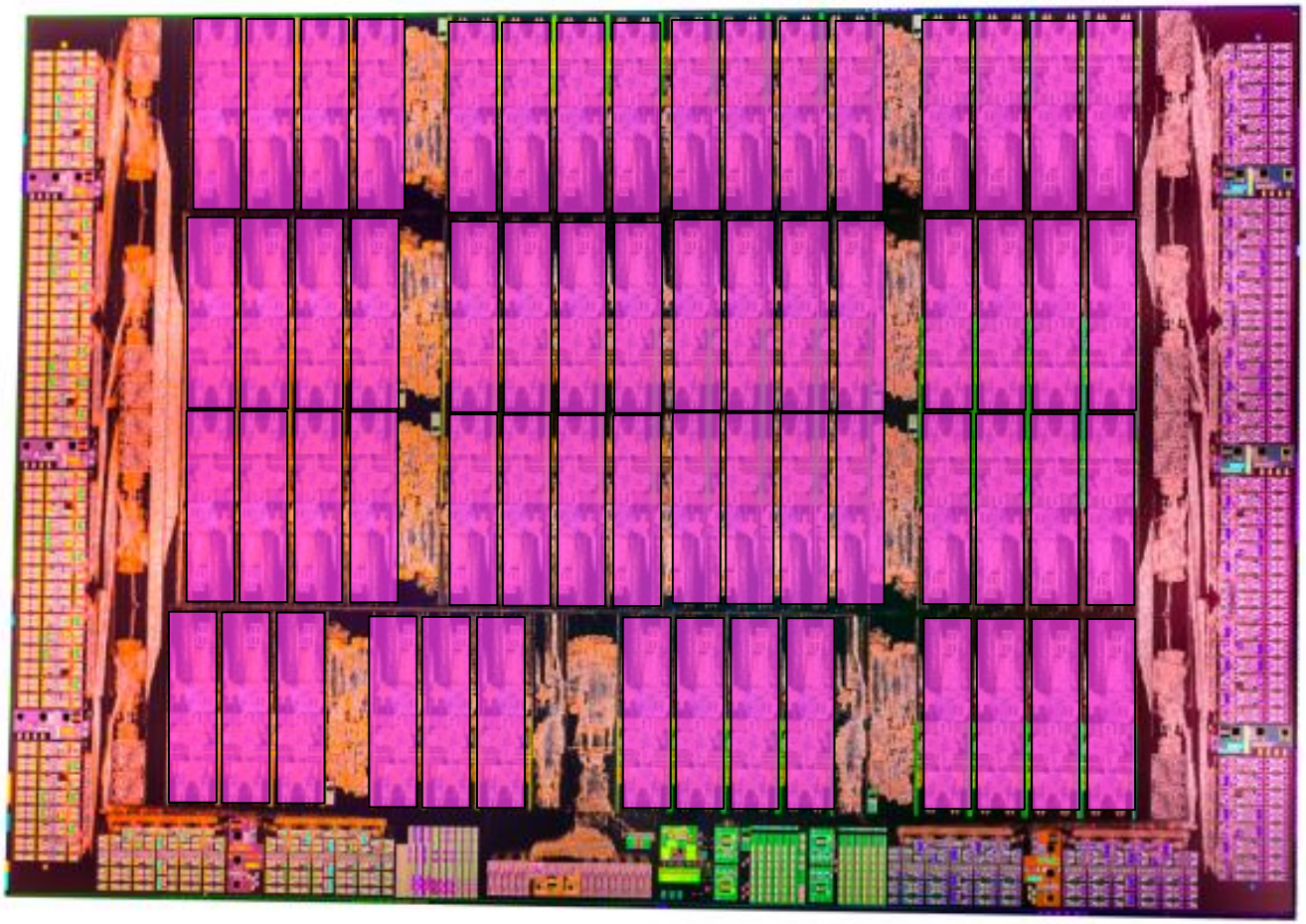
- GPU dedicates much more space to compute
  - At expense of caches, controllers, sophistication etc



= compute unit
(= SM
= 32 CUDA cores)

Intel Xeon Phi

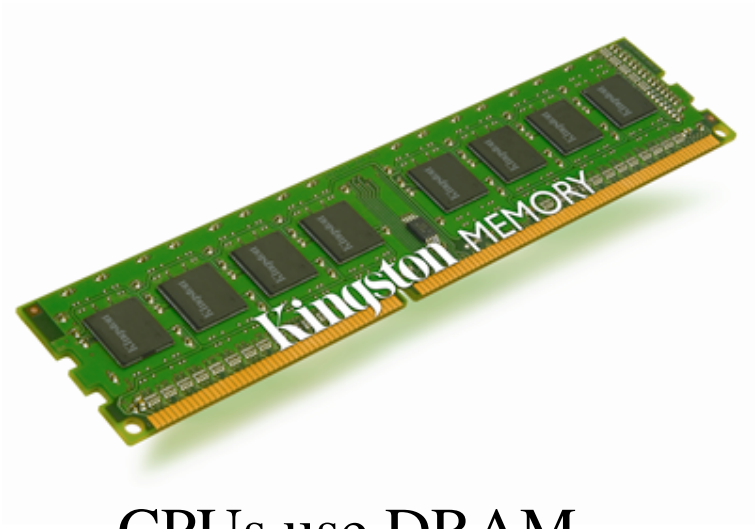
- As does Xeon Phi



= compute unit (= core)

Memory

- GPUs and Intel Xeon Phi both use Graphics memory: much higher bandwidth



CPUs use DRAM

GPUs and Xeon Phi use Graphics DRAM

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

Accelerators: 4 key factors

- *Parallel processing*
  - Accelerators have a much higher extent of parallelism than CPUs. Many more cores and/or operations per core
- *Clock frequency*
  - Accelerators typically have lower clock-frequency than CPUs, and instead get performance through parallelism
- *Memory bandwidth*
  - Accelerators use high bandwidth GDDR memory
- *Memory latency*
  - Memory latency from GDDR is similar to DDR
  - GPUs hide latency through very high levels of multithreading
  - Xeon Phi hides latency in a similar way to CPUs, although the caches are smaller and no out-of-order execution (in current models).

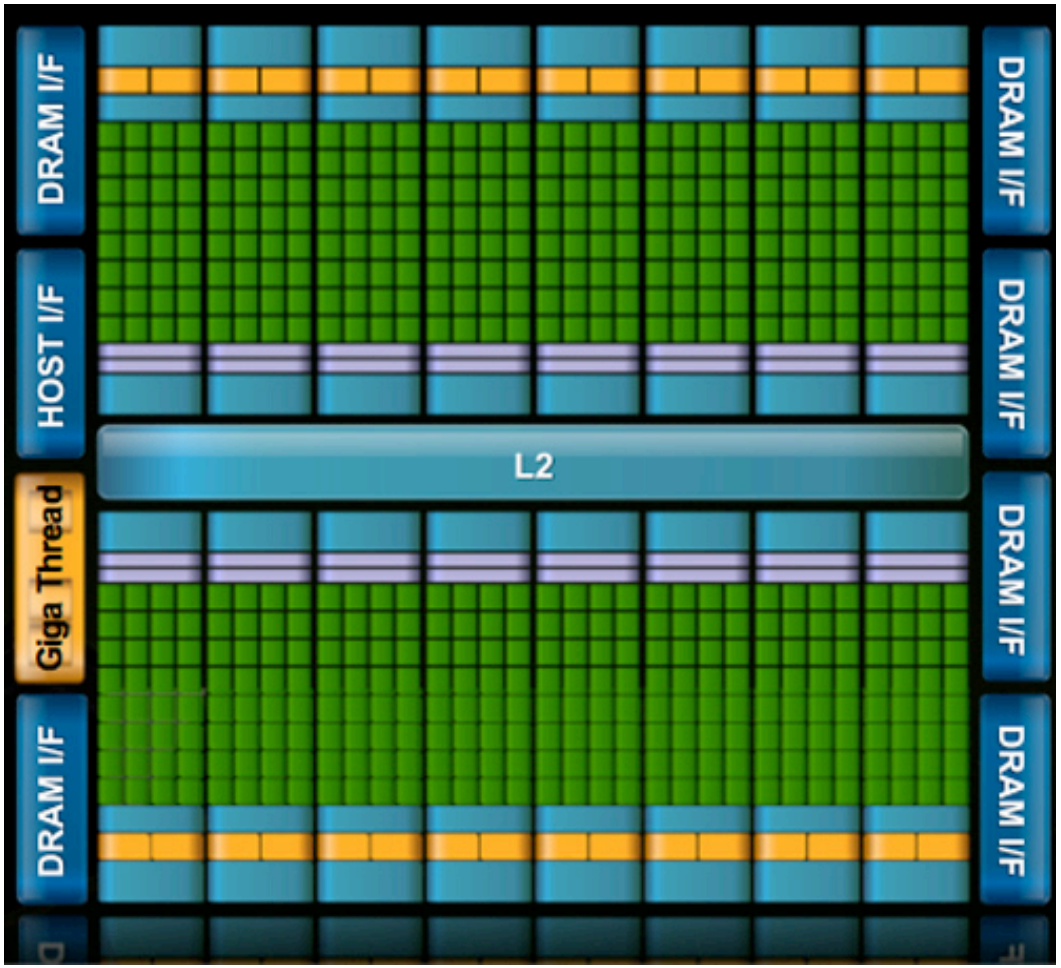
Latest Technology

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

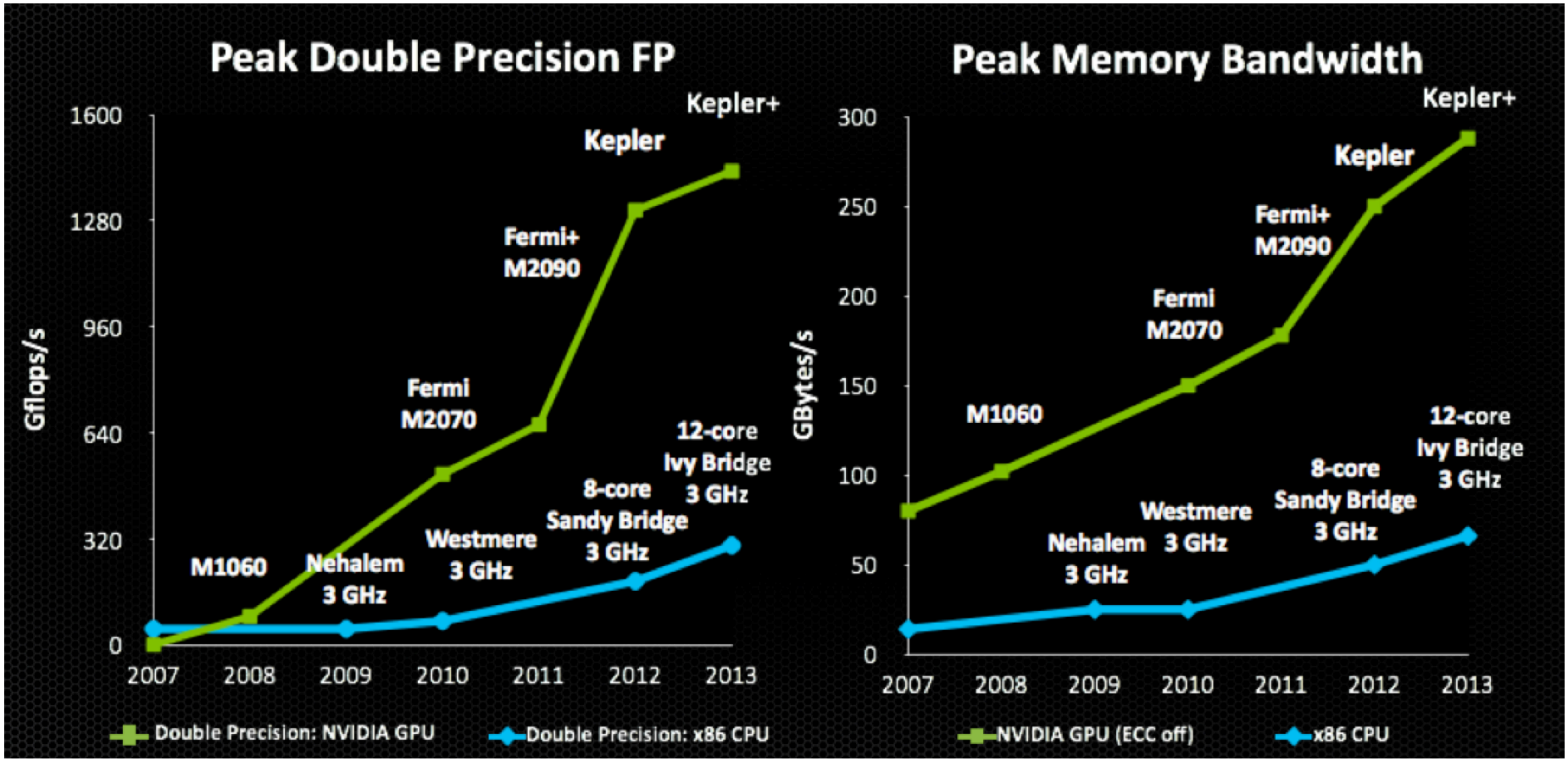
- AMD
  - *FirePro* HPC specific GPUs have evolved from (ATI) *Radeon* series

- Intel
  - *Xeon Phi* has recently emerged to compete with GPUs for general purpose computation

NVIDIA Tesla Series GPU

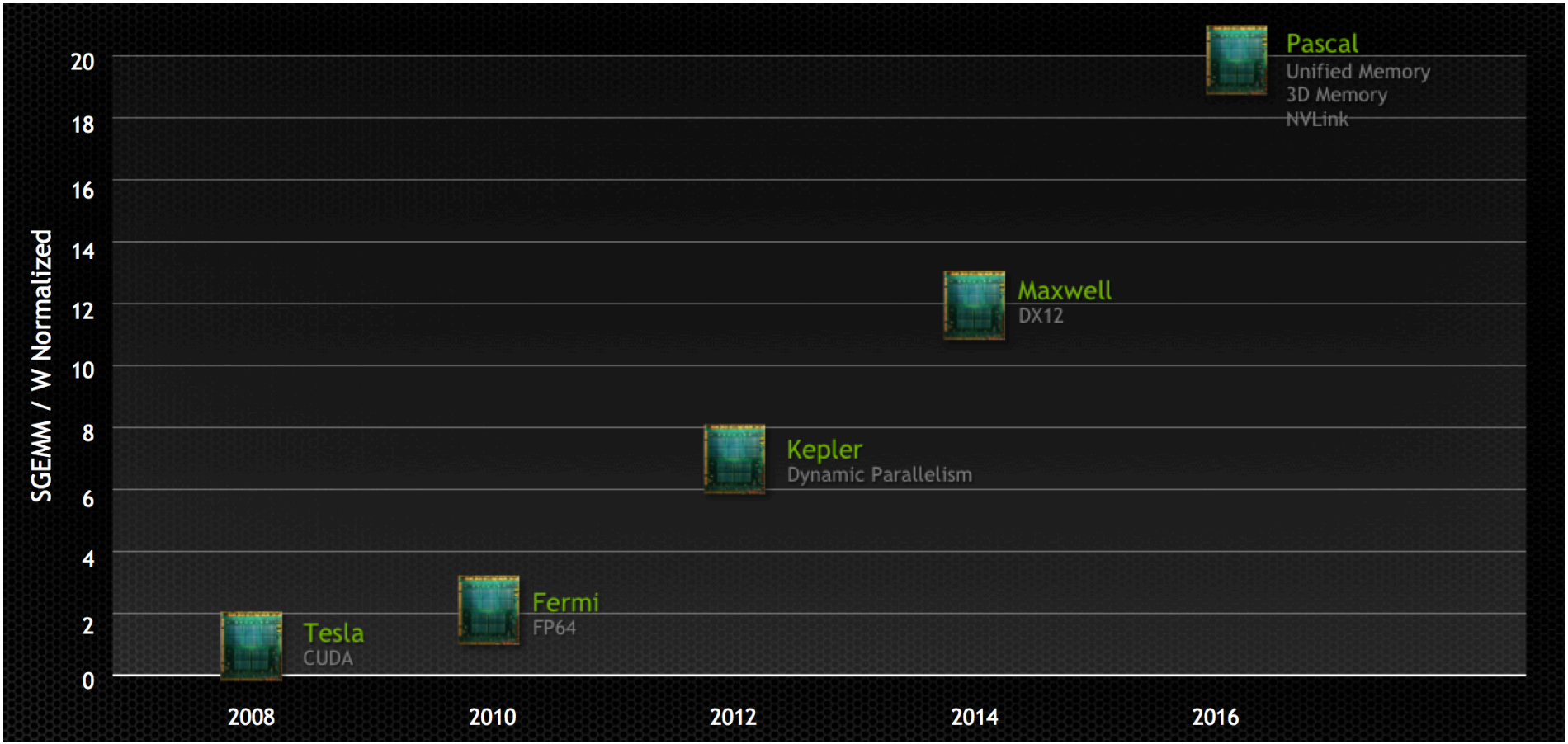


- Chip partitioned into *Streaming Multiprocessors* (SMs)
  - That act independently of each other
- Multiple cores per SM
  - Within each SM, groups of cores act in “lock-step”: they perform the same instruction on different data elements
- Number of SMs, and cores per SM, varies across products. High-end GPUs have more than 1,000 total cores



- GPU performance has been increasing much more rapidly than CPU

NVIDIA Roadmap

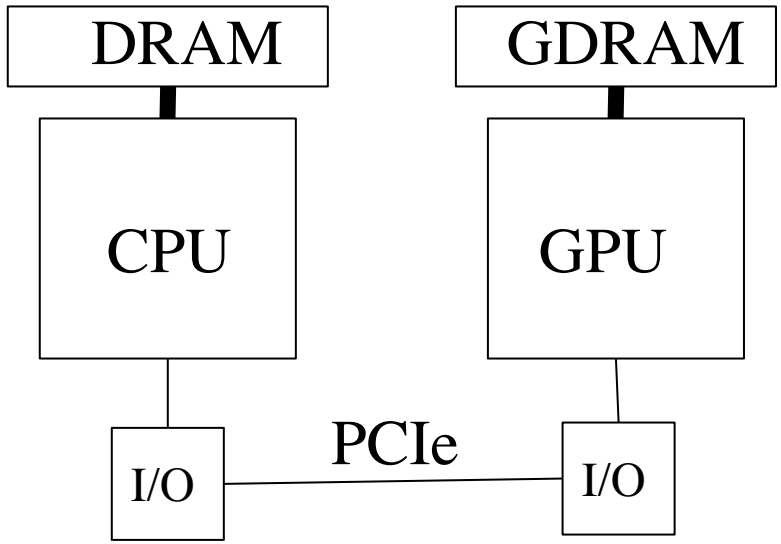


AMD FirePro

- 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

GPUs accelerated systems

- 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



Intel Xeon Phi

- Intel Pentium P54C cores were originally used in CPUs in 1993
  - Simplistic and low-power compared to today's high-end CPUs
- Philosophy behind Phi is to dedicate large fraction of silicone to many of these cores
- And, similar to GPUs, Phi uses Graphics GDDR Memory
  - Higher memory bandwidth that standard DDR memory used by CPUs

Intel Xeon Phi

- Each core has been augmented with a wide 512-bit vector unit
- For each clock cycle, each core can operate vectors of size 8 (in double precision)
  - Twice the width of 256-bit “AVX” instructions supported by current CPUs
- Multiple cores, each performing multiple operations per cycle
- Peak performance and memory bandwidth similar to GPUs

Xeon Phi Systems

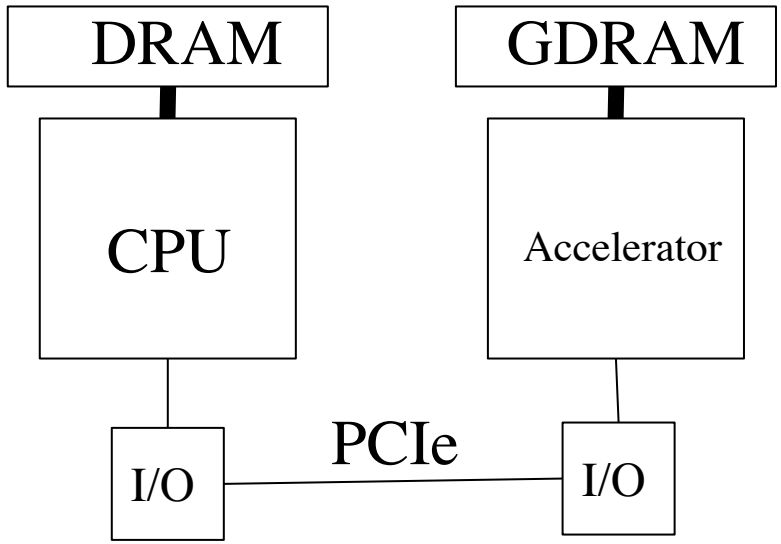
- Unlike GPUs, Each Xeon Phi runs an operating system
- User can log directly into Xeon Phi and run code
  - "native mode"
  - But any serial parts of the application will be very slow relative to running on modern CPU
- Typically, each node in a system will contain at least one regular CPU in addition to one (or more) Phis.
- Phi Acts as an "accelerator", in exactly the same way as already described for GPU systems.
- "Offload mode": run most source code on main CPU, and offload computationally intensive parts to Phi

Programming

- GPUs
  - 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
- Xeon Phi
  - Can just use regular CPU code with OpenMP
    - Typically needs work to allow compiler to auto-vectorise efficiently
    - Intel specific directives allow offloading to phi, such that fast CPU core can be used for serial parts of code
  - Also Intel Cilk and Intel Thread Building Blocks

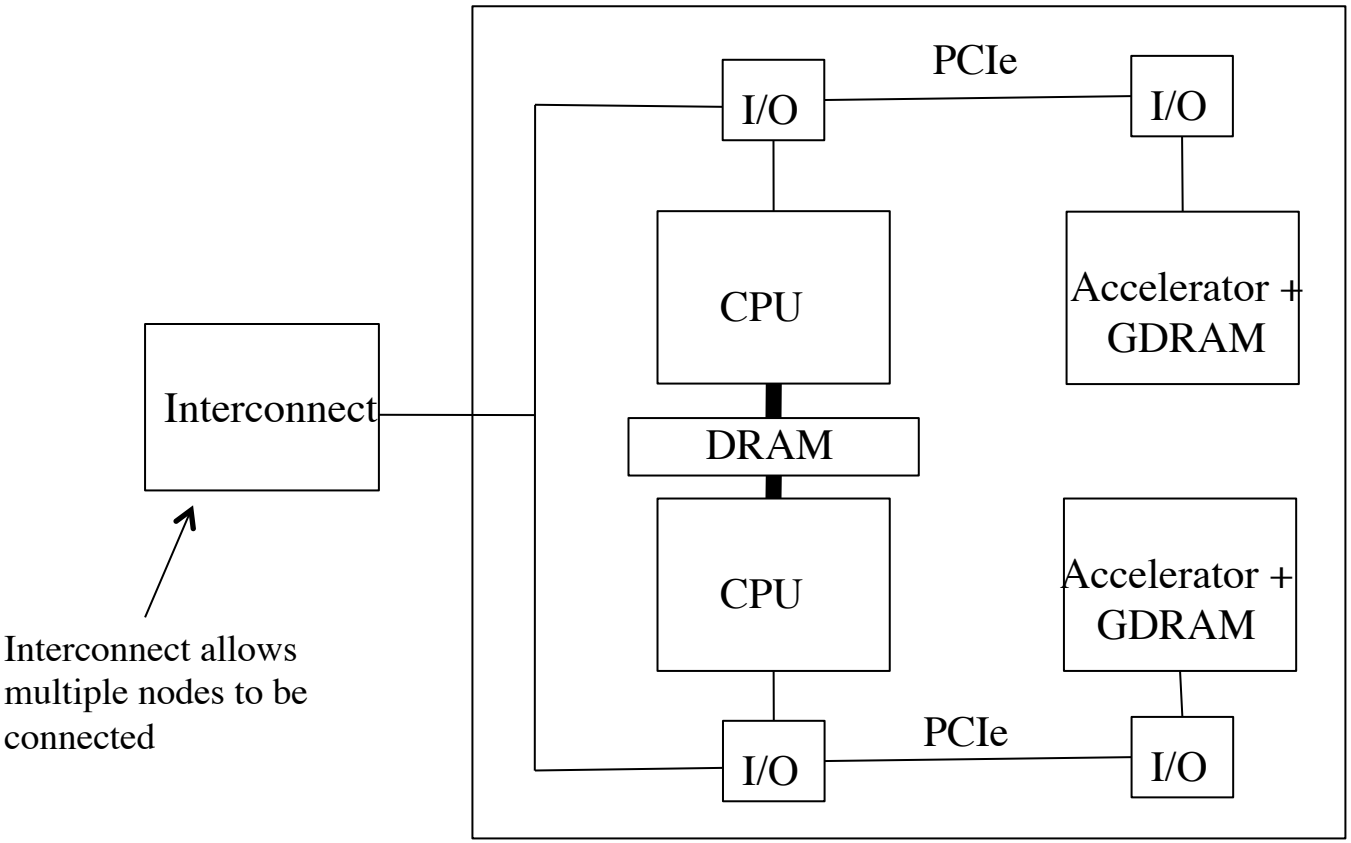
GPU Accelerated Systems

- CPUs and Accelerators are used together
  - Communicate over PCIe bus

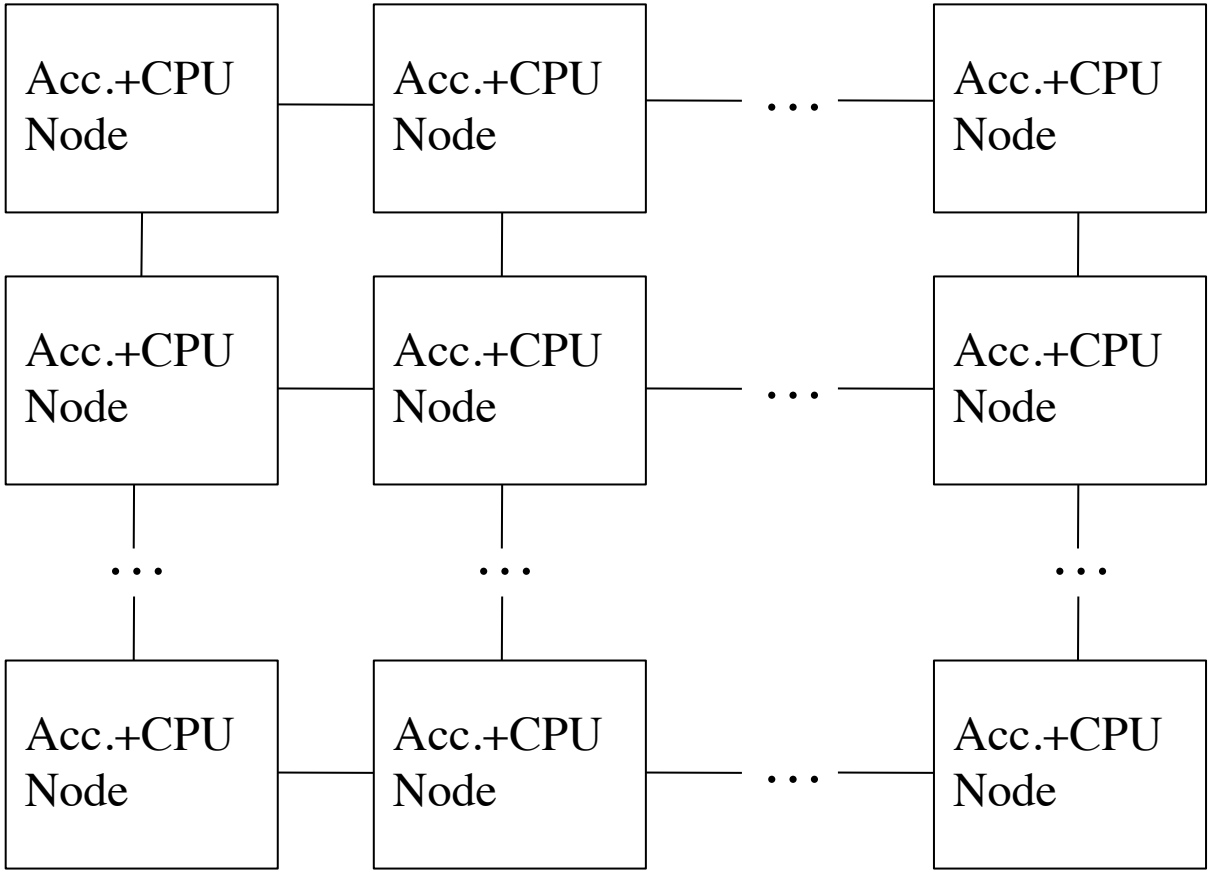


Scaling to larger systems

- Can have multiple CPUs and accelerators within each “workstation” or “shared memory node”
  - E.g. 2 CPUs +2 Accelerators (above)
  - CPUs share memory, but Accelerators 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

The future

- Some very interesting developments on the horizon
- In 2016, both GPUs and Xeon Phi will adopt stacked 3D memory as an alternative to GDDR.
  - Higher bandwidth, lower latency
  - NVIDIA Pascal: High Bandwidth Memory (HBM)
  - Intel Knights Corner (KNC): Hybrid Memory Cube (HMC)

The future

- Pascal will also introduce NVLINK: an alternative to PCIe with several-fold performance benefits
  - To closely integrate fast dedicated CPU with fast dedicated GPU
- KNC is moving from Pentium to Silvermont architecture cores (adapted from Atom mobile range)
  - Faster single thread performance
  - Also integrating network controllers on chip
  - HPC systems will likely just use KNC in “native mode”

Summary

- Accelerators have higher compute and memory bandwidth capabilities than CPUs
  - Silicon dedicated to many simplistic cores
  - Use of graphics memory
- Accelerators are typically not used alone, but work in tandem with CPUs
- Most common are NVIDIA GPUs and Intel Xeon Phis.
  - Architectures differ
  - 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