COSC 420
High Performance Computing

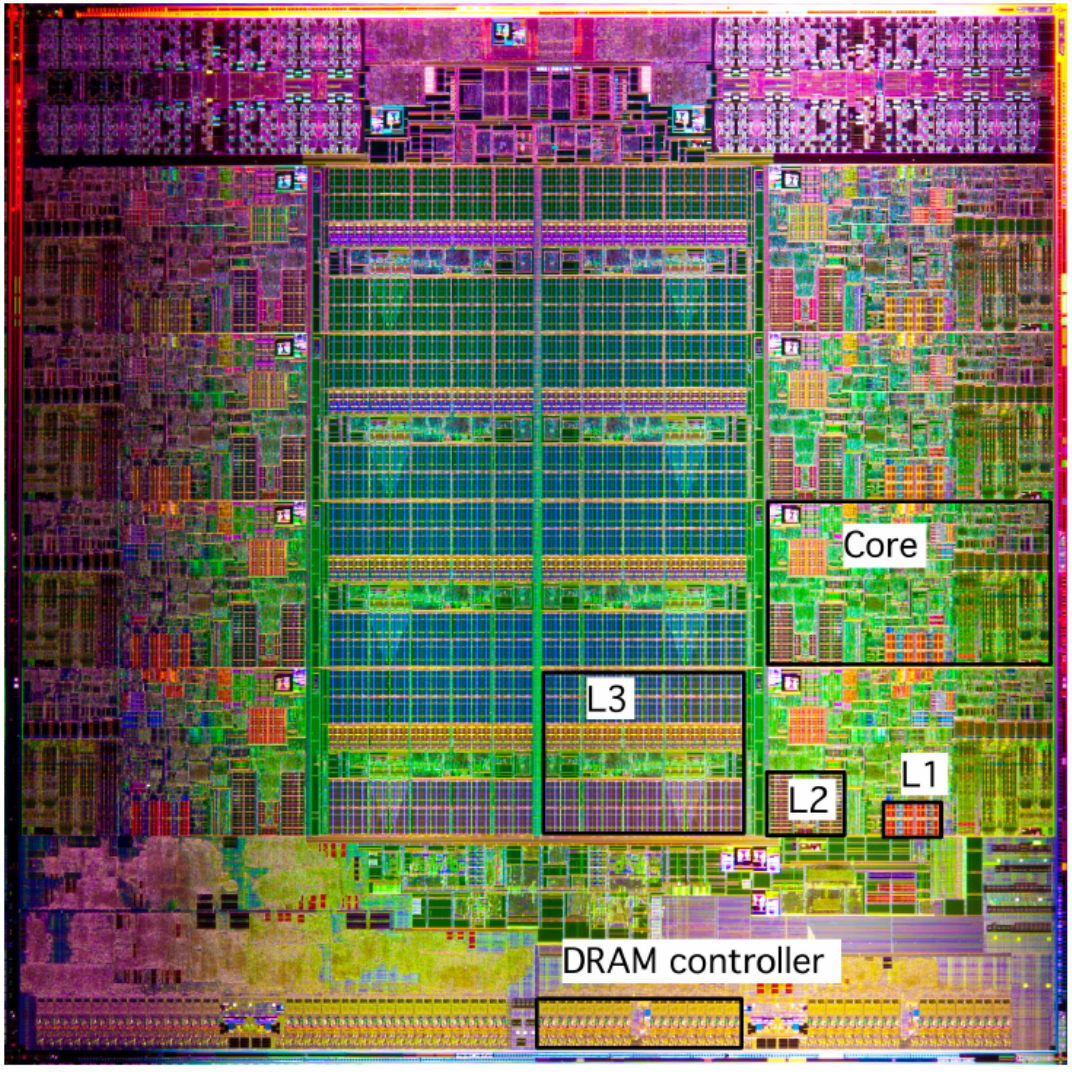
High Performance Computing: A Definition

High Performance Computing *is a field of endeavor that relates to all facets of technology, methodology, and application associated with achieving the greatest computing capability possible at any point in time and technology.*

Moore's Law & Transition to Parallel Computation

- Moore's Law: An observation made by Gordon Moore of Fairchild Semiconductor (and cofounder of Intel Corporation) stating that the number of transistors in large integrated circuits doubles approximately every 2 years.
- Simply stated, the computational ability of computers would double every two years.
- This held true for several decades but physical constraints started limiting IS density. Supercomputers started incorporating multiple processing units working in parallel.
- In the early 2000s commercial chip manufactures started moving to multiple processor units on a single IS, i.e. multiple core processors.

Moore's Law & Transition to Parallel Computation



8 Core Processor

Moore's Law & Transition to Parallel Computation

- With the commercial chip architecture now matching (albeit on a smaller scale) supercomputer architecture, methods of super-computing are also applicable to consumer grade applications.
- Although there are many aspects of hardware and software design that contribute to the overall performance of the machine on a calculation, the biggest, at this time, is the construction of efficient parallel programs. This will be the main focus of the class.

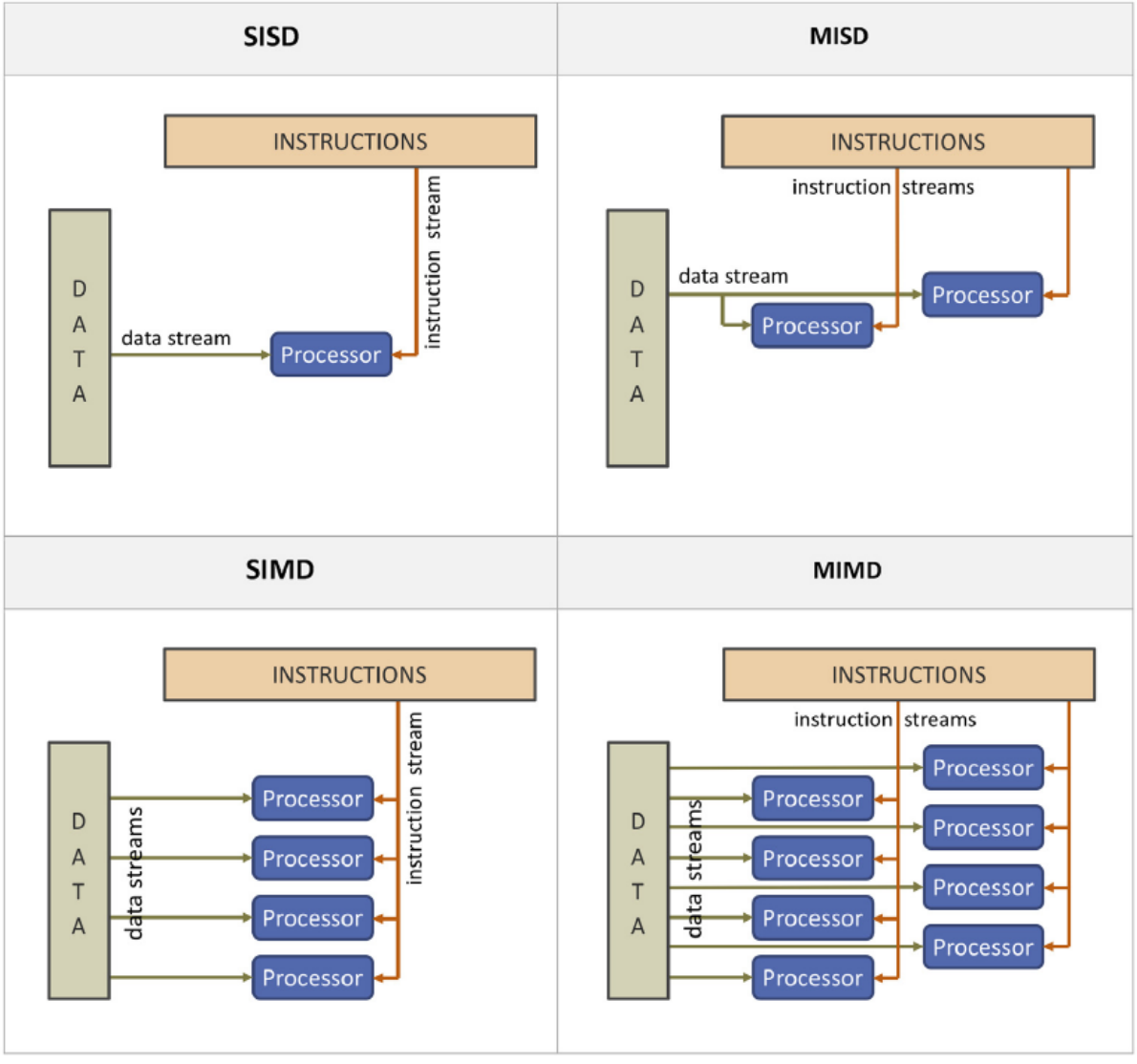
Flynn's Taxonomy (1976)

- SISD: Single instruction stream, single data stream (pronounced "sisdee"). Traditional CPU architecture, that is Von Neumann architecture. At any one time only a single instruction is executed, operating on a single data item.
- SIMD: Single instruction stream, multiple data stream (pronounced "simdee"). Multiple processors, each operating on its own data item, but they are all executing the same instruction on that data item. Older vector computers and some components of GPU processors.
- MISD: Multiple instruction stream, single data stream (pronounced "misdee"). Architecturally not used but certain pipeline systems and shared data parallel programs have this methodology.

Flynn's Taxonomy (1976)

- MIMD: Multiple instruction stream, multiple data stream (pronounced "mimdee"). Multiple CPUs operate on multiple data items, each executing independent instructions. Most current parallel/super computers are of this type.

Flynn's Taxonomy (1976)

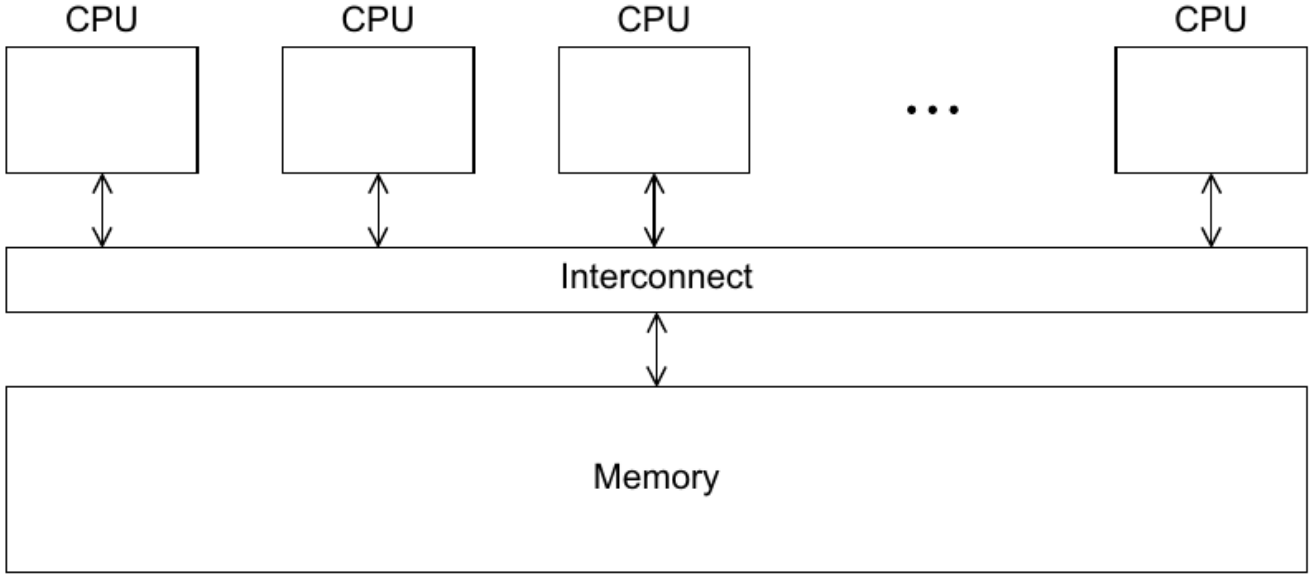


Flynn's Taxonomy (1976)

- SPMD: Single program, multiple data stream (pronounced "spimdee").
  - Not strictly part of Flynn's taxonomy, but inspired by it.
  - A more practical variation of the SIMD/MIMD models.
  - Processors execute multiple, possibly differing instructions, each on their own data.
  - The programmer starts up the same executable on the parallel processors.
  - Instead of issuing and broadcasting one instruction at a time to all the processing units, the SPMD model sends a function call or an entire program that is to be performed by all the processing units of the parallel machine.

SPMD Modes

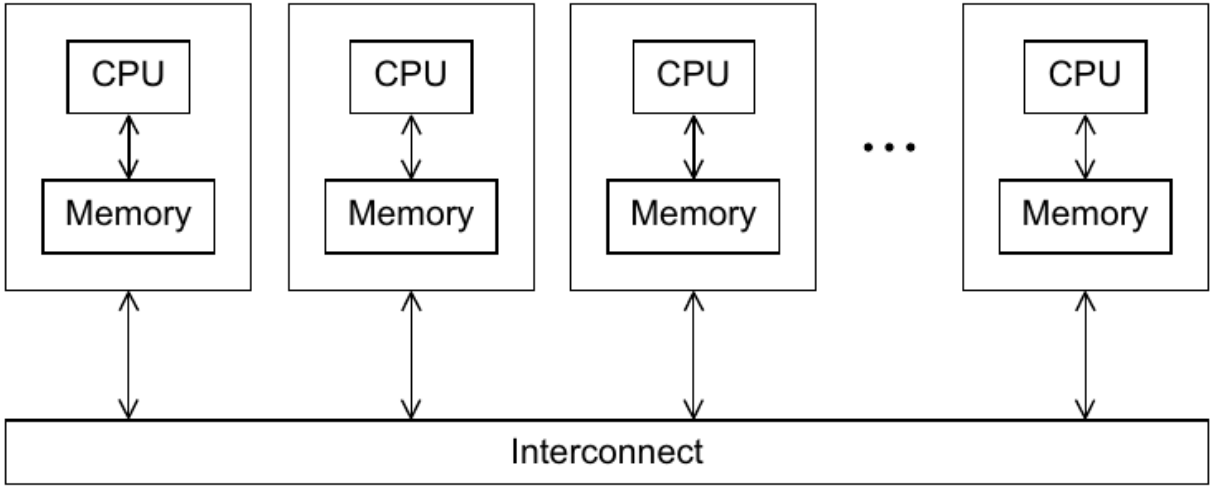
- Shared Memory Systems



  - Expensive for large scale systems.
  - Not easy to expand interconnect.
  - Mostly utilized on single chip processors where the CPUs listed above are cores to the single CPU chip/socket.
  - Multi-threading approach to program utilization, pthreads and OpenMP.

SPMD Modes

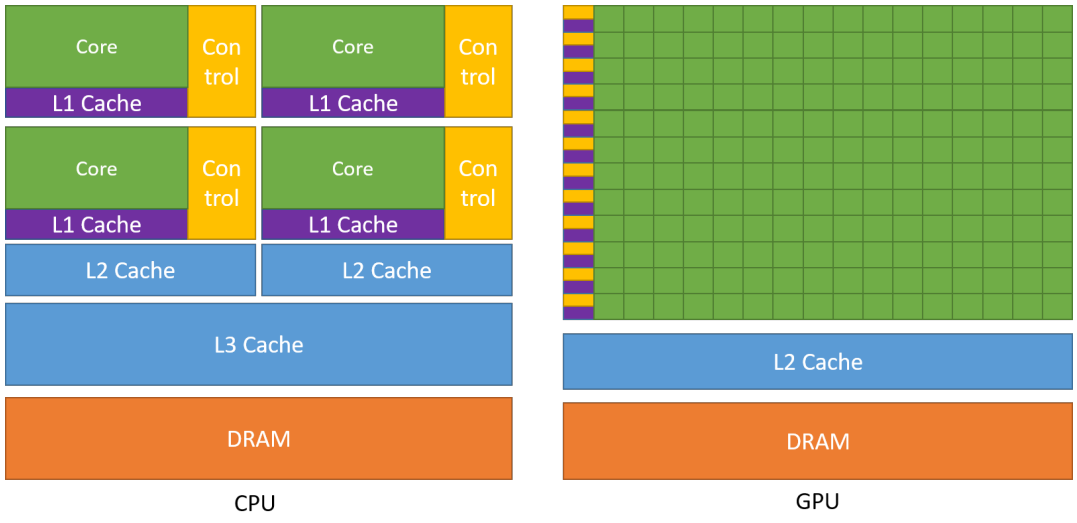
- Distributed Memory Systems



  - Basic architecture of large-scale clusters and supercomputers.
  - Easy to expand interconnect. It is usually an Ethernet or InfiniBand switch.
  - Message-passing approach to program utilization, MPI.

SPMD Modes

- Accelerators & GPUs



  - GPU has a different architecture, it should, it is designed for graphics processing and not general processing.
  - Can be configured to solve certain parallel computational problems.
  - Massive numbers of processing cores, 100's to 1000's.
  - SIMD-like model of programming, CUDA, OpenCL, OpenACC.

Top 500 List

- GigaFlops: 1,000,000,000 FLOPS
- TeraFlops: 1,000,000,000,000 FLOPS
- PetaFlops: 1,000,000,000,000,000 FLOPS
- ExaFlops: 1,000,000,000,000,000,000 FLOPS
- ZetaFlops: 1,000,000,000,000,000,000,000 FLOPS

`https://www.top500.org`

Top 5 — https://www.top500.org

1. **Frontier**: Oak Ridge National Laboratory, 8,699,904 cores, HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X accelerators. 1.194 Exaflop/s.
2. **Fugaku**: RIKEN Center for Computational Science Japan, 7,630,848 cores, A64FX 48C 2.2GHz. 442.01 Petaflop/s.
3. **LUMI**: EuroHPC/CSC Finland, 2,220,288 cores, HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X accelerators, 309.1 Petaflop/s.
4. **Leonardo**: EuroHPC/CINECA Italy, 1,824,768 cores, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB accelerators, 238.7 Petaflop/s.
5. **Summit**: Oak Ridge National Laboratory, 2,414,592 cores, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100 accelerators, 148.6 Petaflop/s.

Course Focus

- Programming and Parallel Algorithms
  - Base programming language of C/C++.
  - Shared Memory Systems: OpenMP and/or pthreads.
  - Distributed Memory Systems: MPI — Message Passing Interface.
  - GPU Accelerators: CUDA — Compute Unified Device Architecture
  - Conversion or complete rewriting of sequential algorithms into parallel algorithms.
- Cluster Creation and Use
  - Create your own Beowulf cluster of COTS to run MPI applications.
  - Test run and time your MPI applications on this system.

Course Focus

- Analysis
  - Timing analysis on single machine, your cluster, and larger-scale university systems.
  - Analyze algorithms and their implementations for,
    - Theoretical Parallelism
    - Raw Speed & Parallel Speedup
    - Scaling
    - Efficiency & Reliability
- Our General Approach
  - Cover the three main APIs (OpenMP, MPI, CUDA) at introductory and intermediate levels along with most of the theoretical concepts of the class.
  - Revisit the three main APIs with more advanced applications.
  - If any time remains, investigate other technologies such as OpenCL, OpenACC, hybrid methods, ...