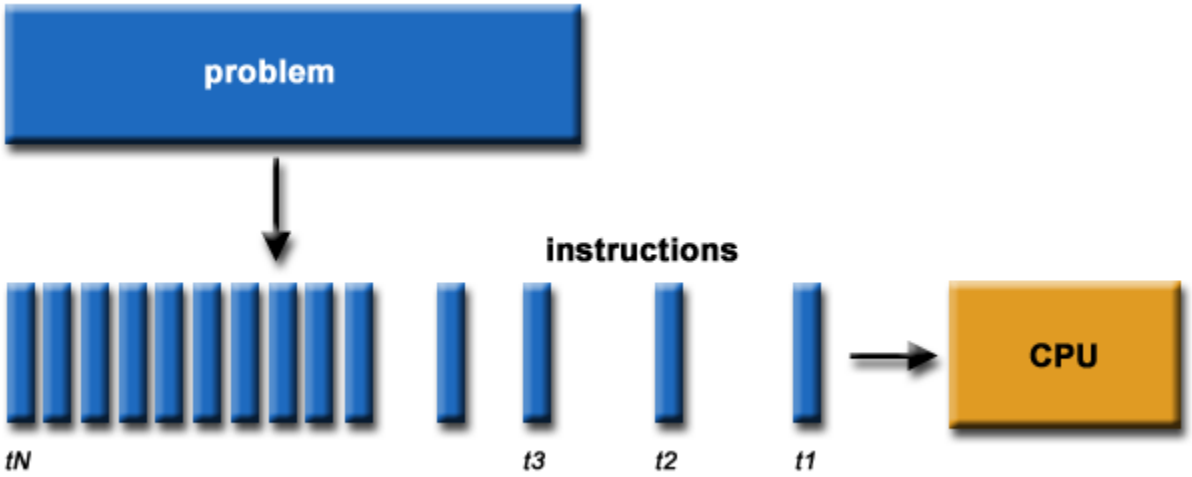
Introduction to Parallel Computing

Mohammad Javad Rashti

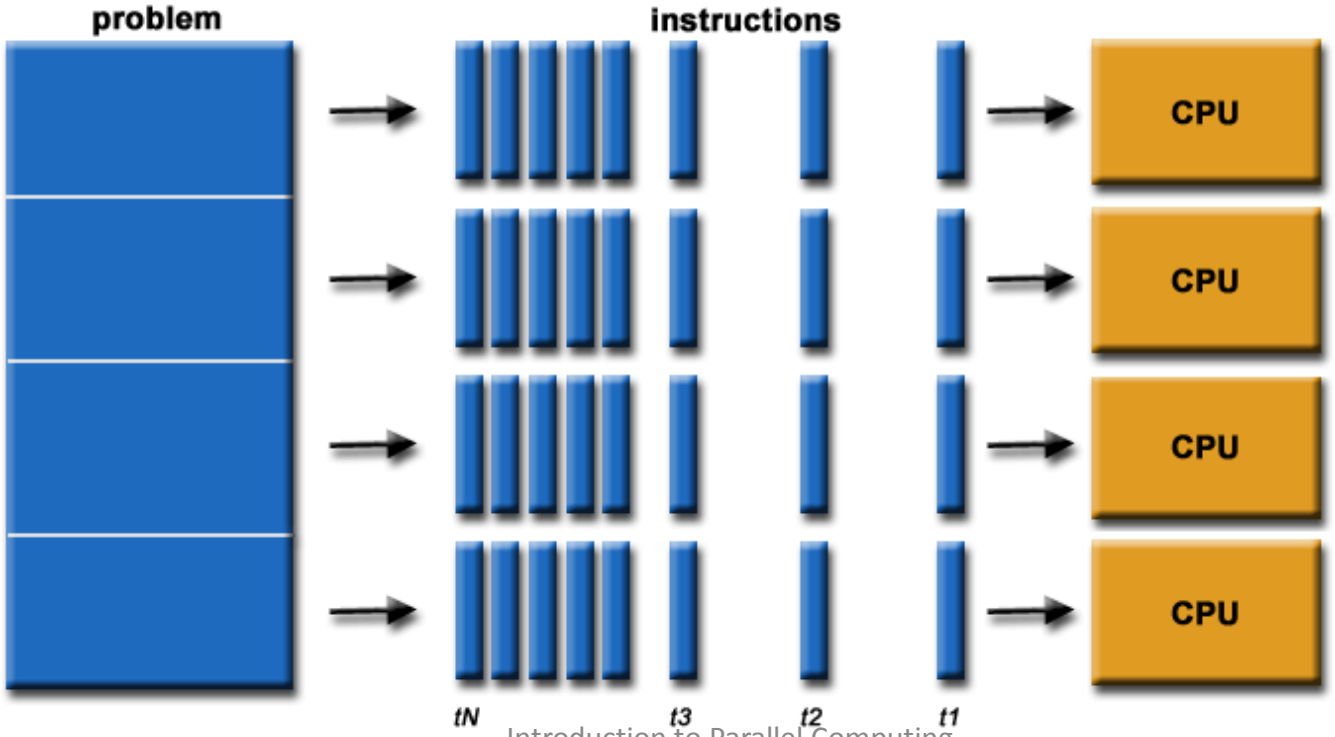
What is Parallel Computing?

- Traditionally a computer has one CPU (Central Processing Unit)
  - User program is a sequence of instructions
  - One CPU can execute program instructions in sequence



What is Parallel Computing?

- Using multiple CPUs to solve the problem(s) in parallel
- Partitioning: Domain or functional decomposition
- **Supercomputer:** A highly-parallel computing system used for running parallel programs



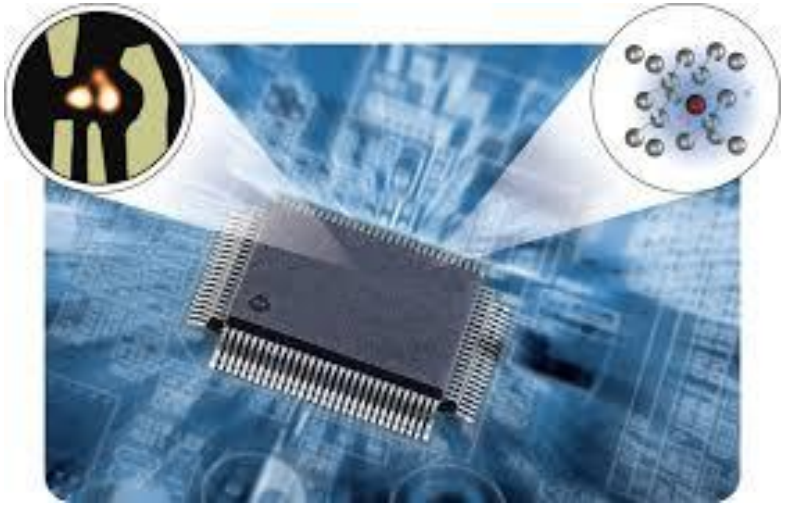
Why do we need Parallel Programming?

- Need to run the program faster
- Need to run the program with higher precision
- Need to run multiple programs simultaneously
- The program does not fit a single computer's resources

IBM BlueGene Supercomputer

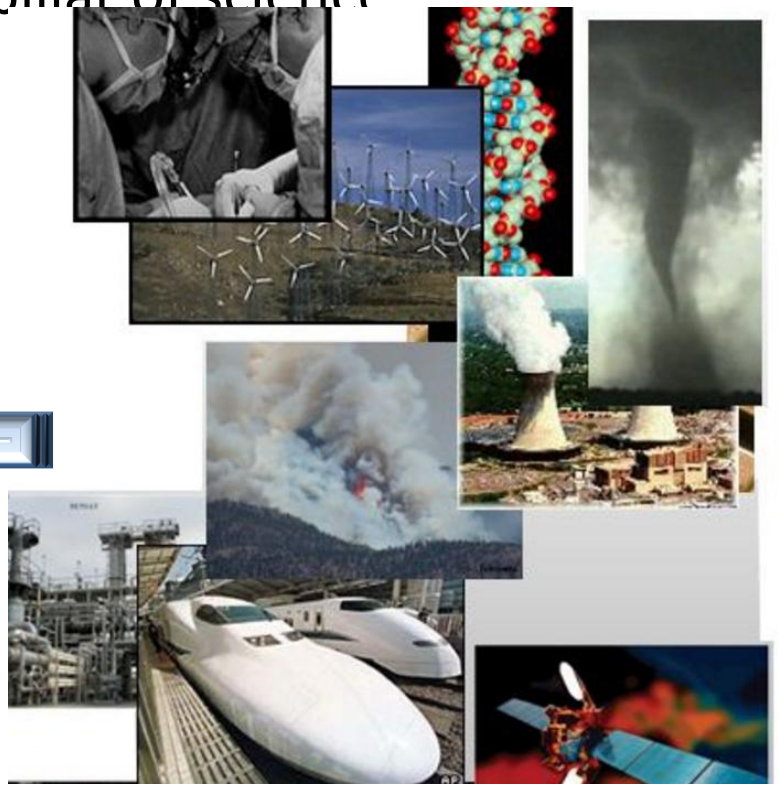
What Limits a Sequential Computer?

- Power limits
  - More frequency means much more power $P \cong f^3$
- Transmission speeds
  - Light: 30cm/ns
  - Cooper: 9cm/ns
- Physical limits
  - 7nm transistors ~ 15 atoms
- Economical limits

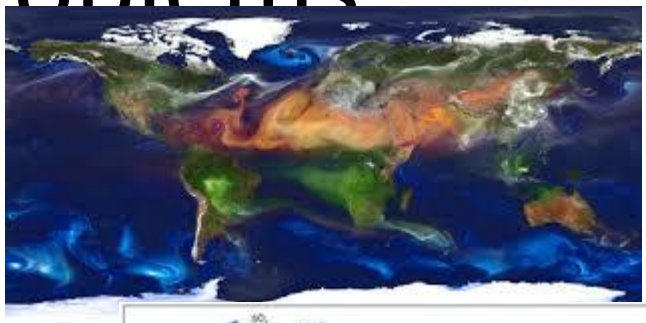


Applications of Parallel Programming

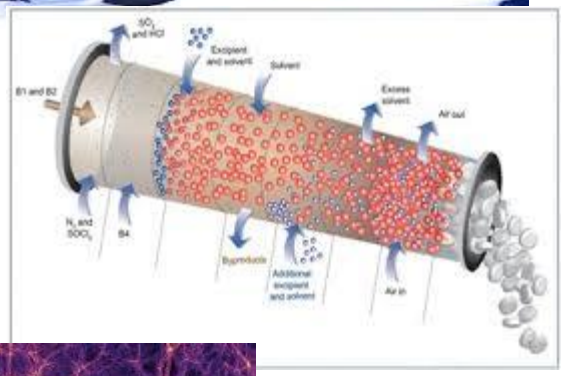
- High Performance Computing (HPC)
  - Grand Challenge Problems (Scientific and industrial)
  - Scientific simulation is the third pillar of science
- Big Data & Business Problems

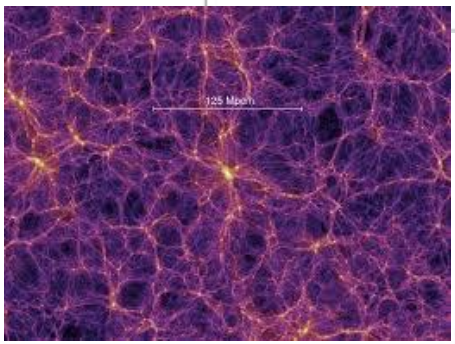


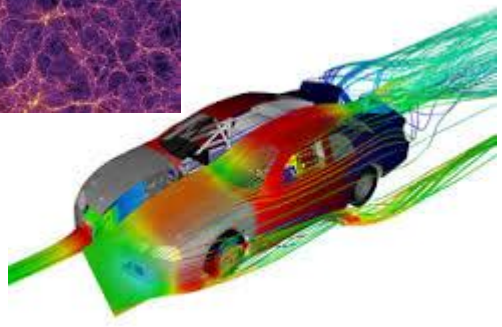
Grand Challenge Problems



- Weather and Climate Simulation
  - Weather forecast, Global warming, ...
- Pharmaceutical Simulations
  - Simulation of new drugs and their effects
- Physics, Astronomy and Molecular Dynamics
  - Simulation of cosmic phenomena
- Fluid and Structural Dynamics
  - Building a bridge
  - Building an aircraft or rocket







Big Data Applications

- Data Mining

- Web Search

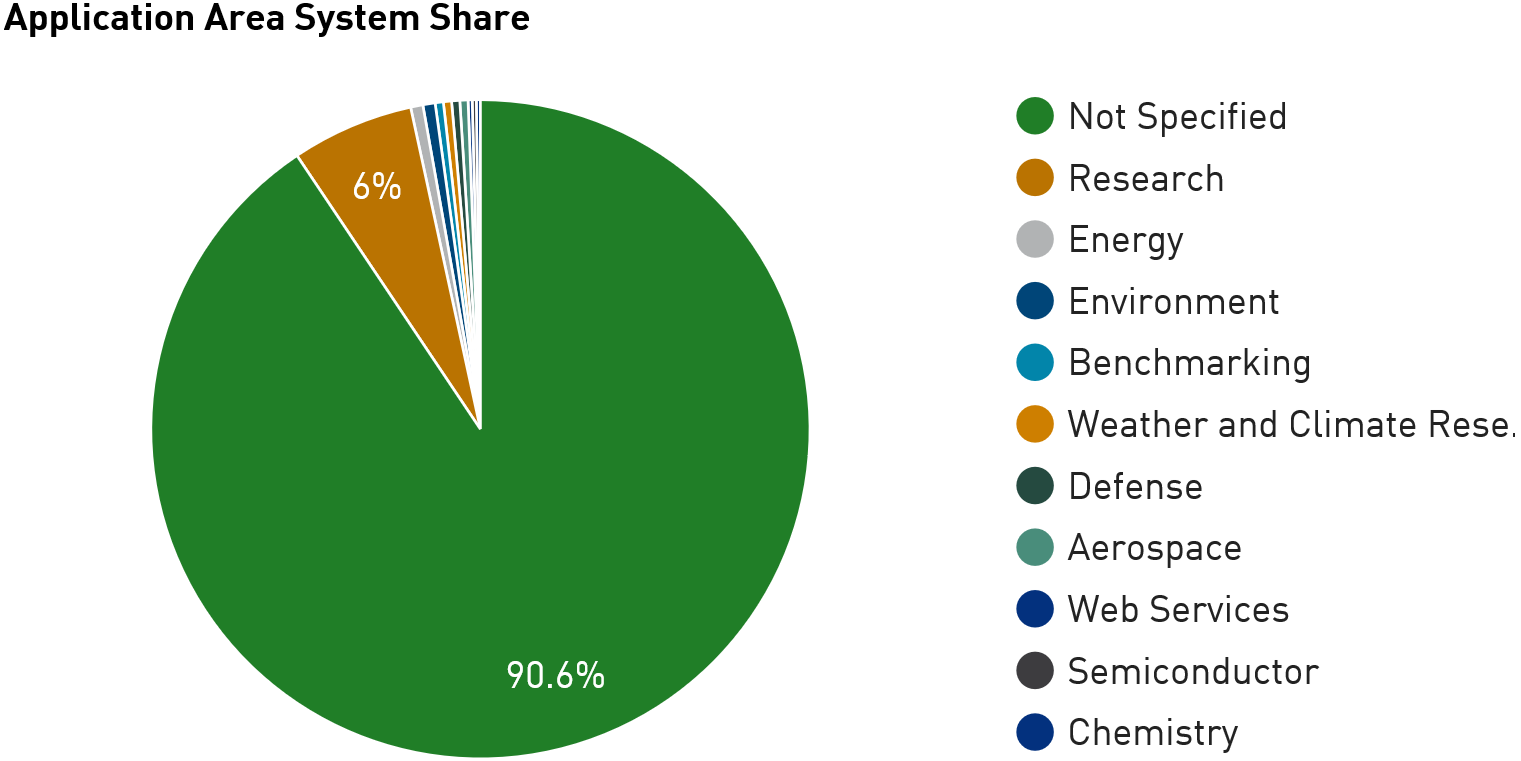
- Medical Data Analysis

Big Data Applications

- Social Networks
- National Security
- Business Intelligence
- Retail

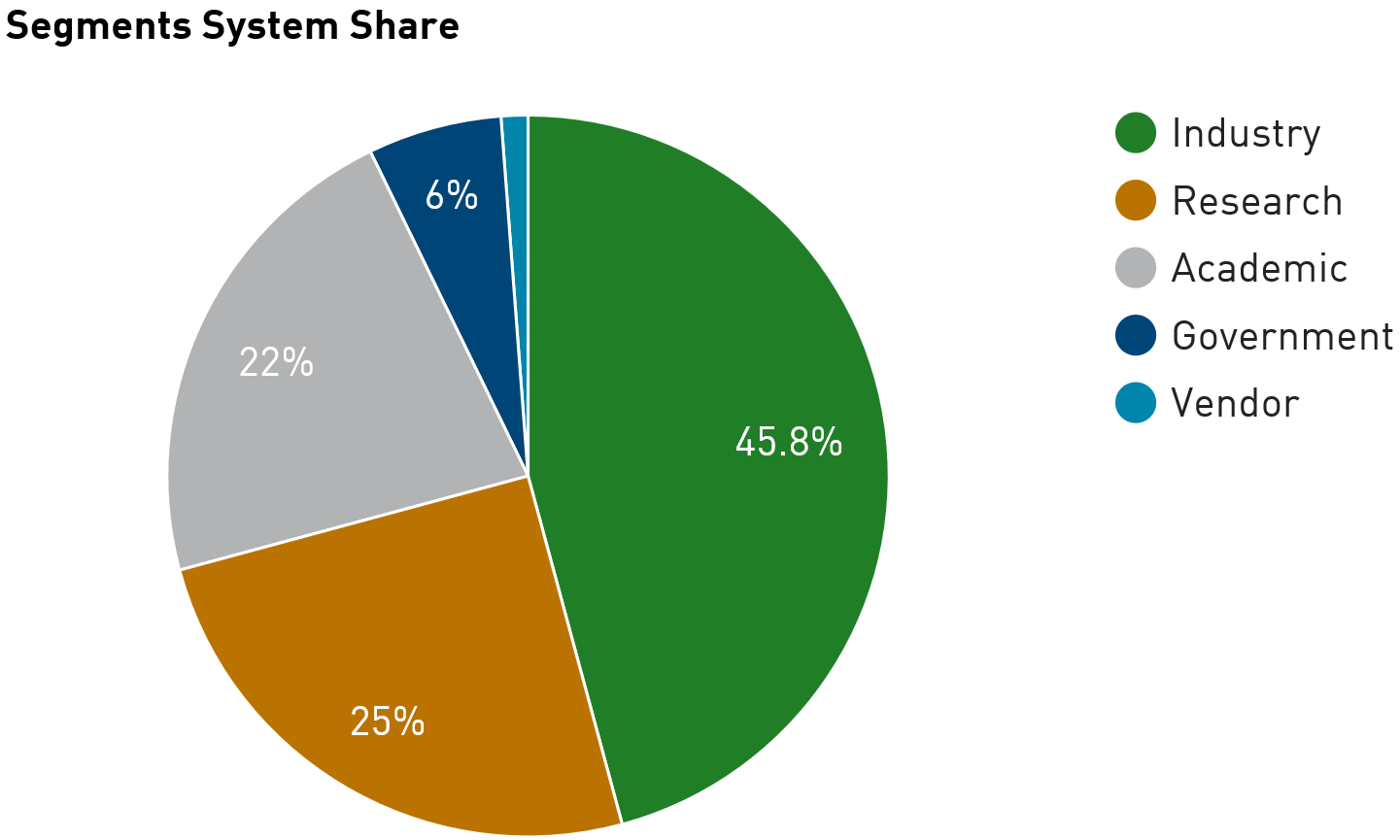
Supercomputer Use Cases

- **Top500.org** Supercomputer List: updated semi-annually



Big Users of Supercomputers

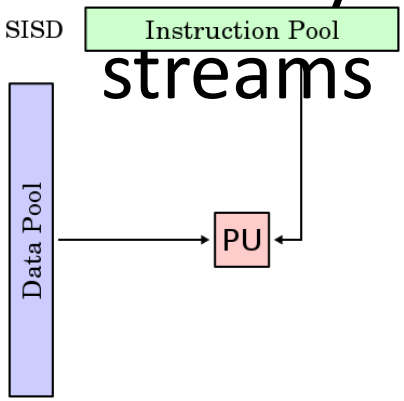
- Top500 Supercomputers by Segment

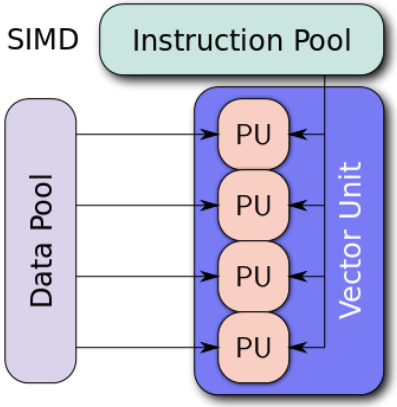


PARALLEL COMPUTING ARCHITECTURES

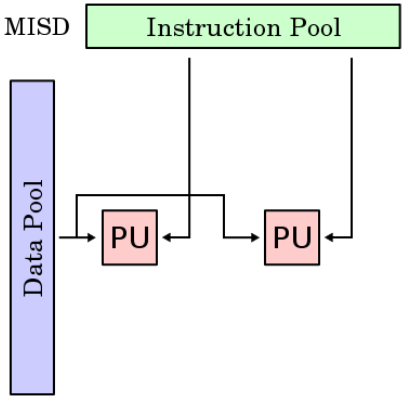
Flynn's Matrix

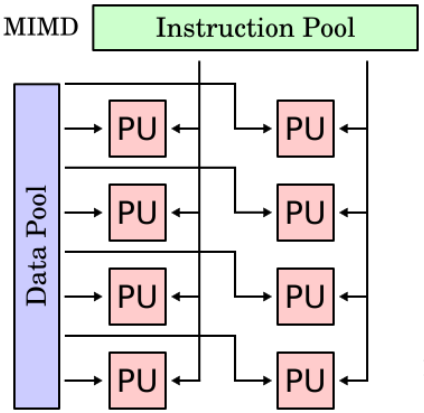
- Classify based on data and instruction streams





	Regular PCs	Vector processors
	SISD Single Instruction stream Single Data stream	**SIMD** Single Instruction stream Multiple Data stream
	MISD Multiple Instruction stream Single Data stream	**MIMD** Multiple Instruction stream Multiple Data stream
	Fault-tolerant systems	Distributed systems



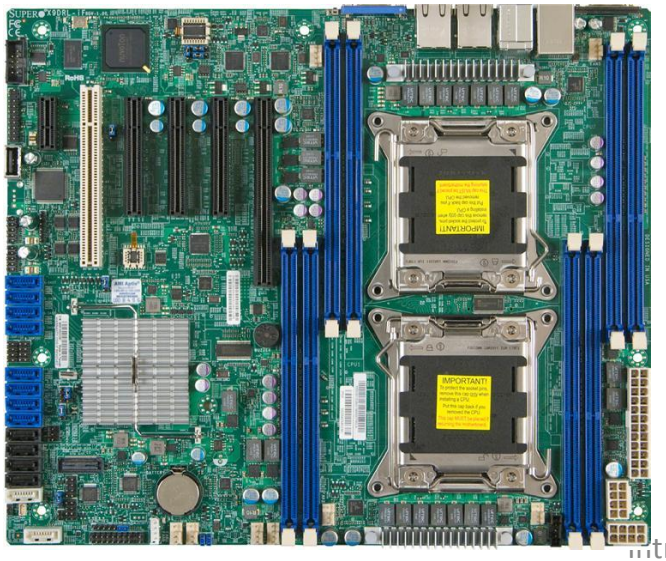


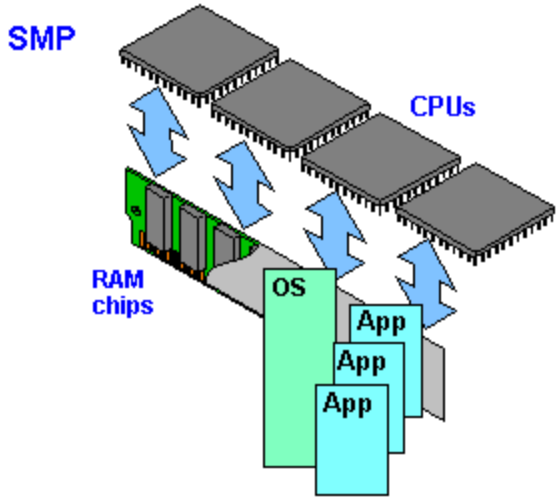
Memory Classification of Parallel Architectures

- Shared Memory
  - Global Address Space (GAS)
  - Uniform Memory Access (UMA)
  - Non-uniform Memory Access (NUMA)
- Distributed Memory

Supercomputer Topology Models

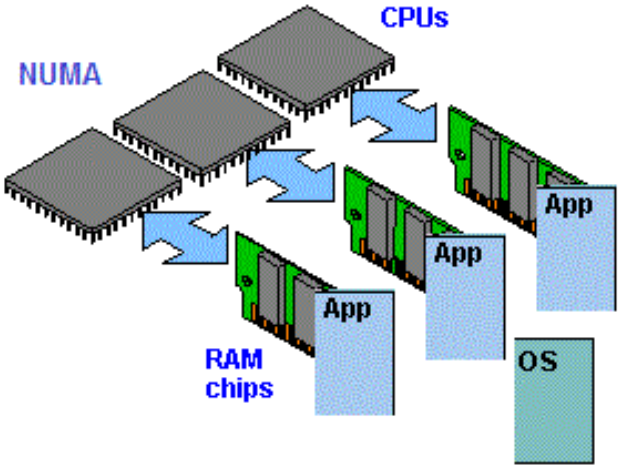
- Shared Memory - Symmetric Multi-processing (SMP)
  - Multiple CPU sockets on the MB
  - Memory is symmetrically shared among them
  - Single OS manages the whole node
  - Example: regular multi-socket server nodes
  - AMD Hyper-transport and Intel Quick-path

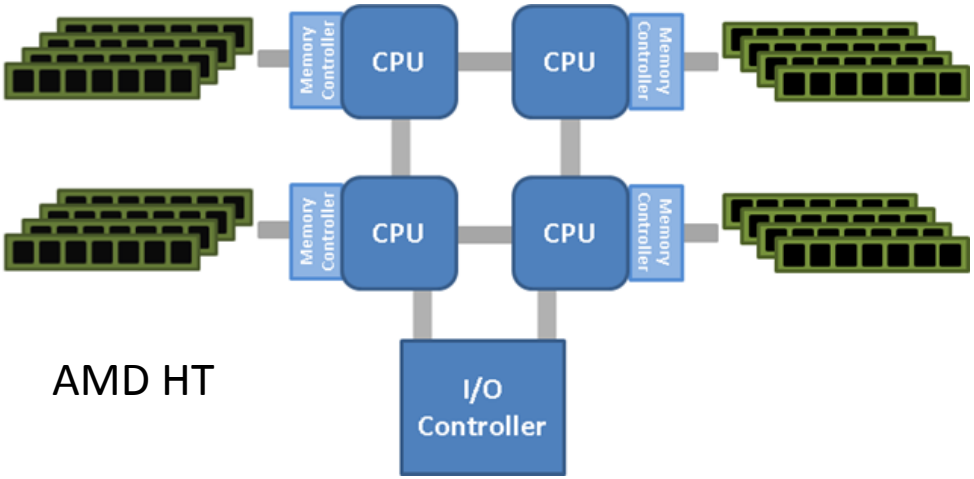




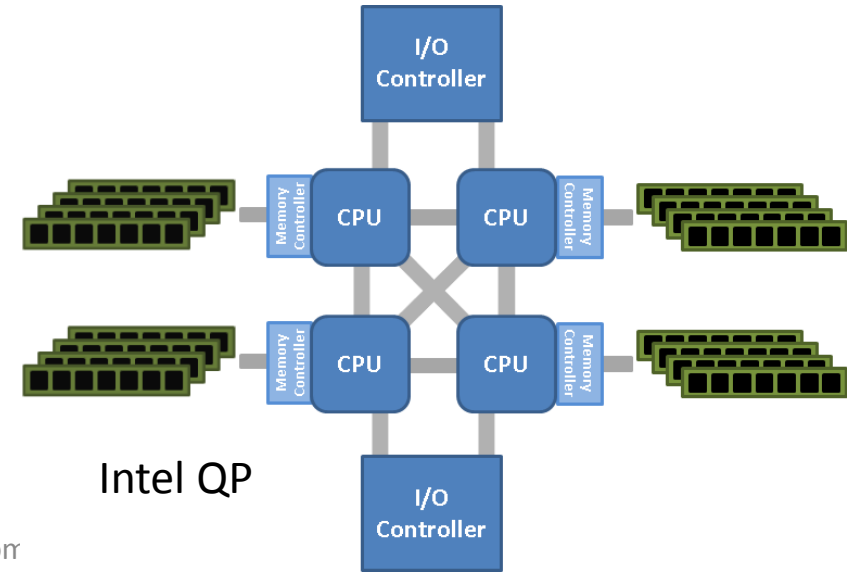
Supercomputer Topology Models

- Shared Memory - Non-uniform Memory Access (NUMA)
  - Each NUMA node has its own "close" memory banks





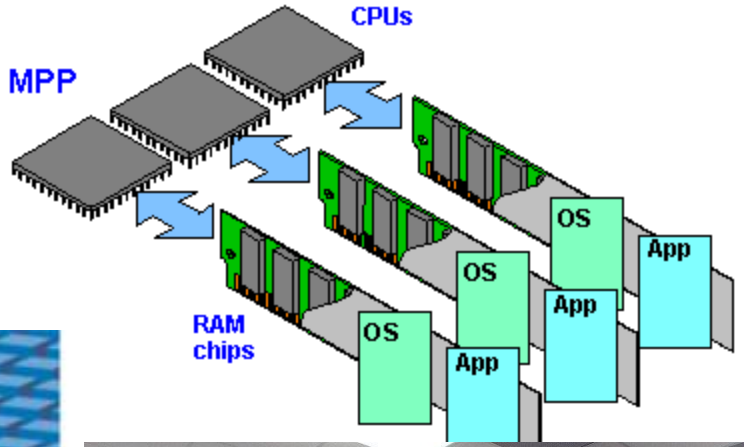
AMD HT

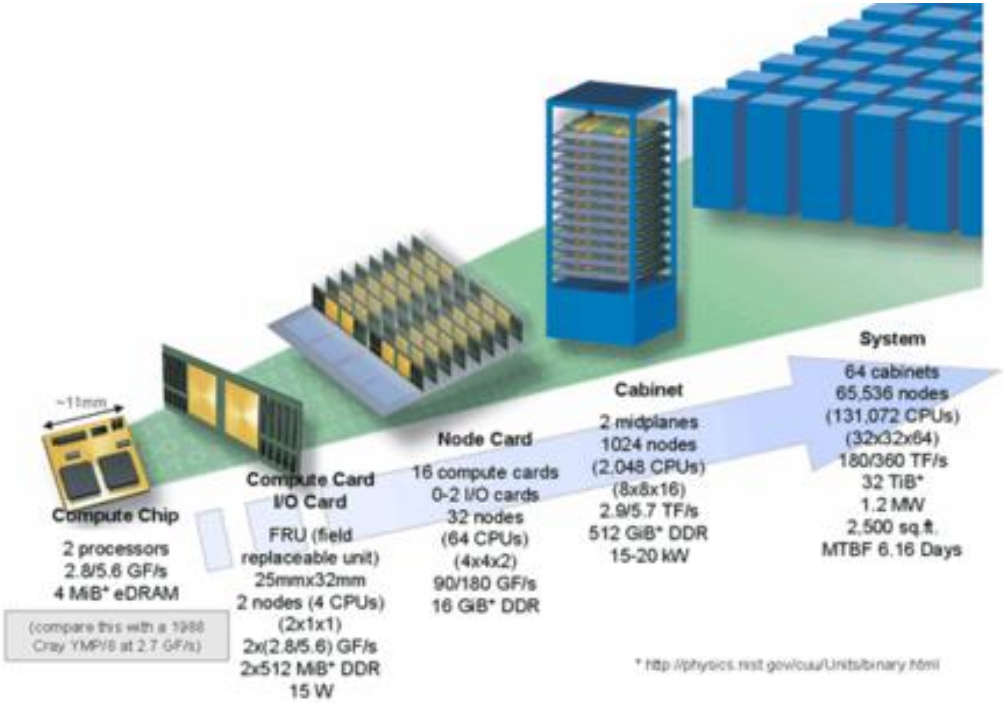


Intel QP

Supercomputer Topology Models

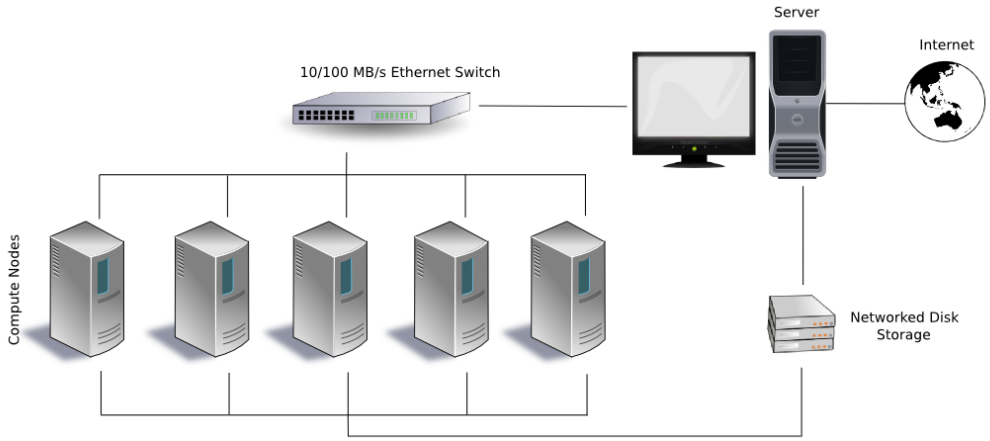
- Distributed Memory -
  - Massively Parallel Processors (MPP)





Supercompute Topology Mode

- Distributed Memory – Clusters
- 426 out of top 500 supercomputers



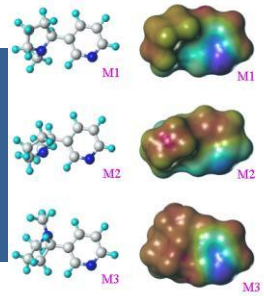
Tianhe2 – World's fastest supercomputer cluster – 33 PFLOPS, 3M cores

HOW TO PROGRAM IN PARALLEL?

Steps in Developing Parallel Programs

- Understand the problem
  - …. and the existing sequential code
- Is the problem parallelizable?

Calculate the potential energy for each of several thousand independent conformations of a molecule. When done, find the minimum energy conformation.



Parallelizable

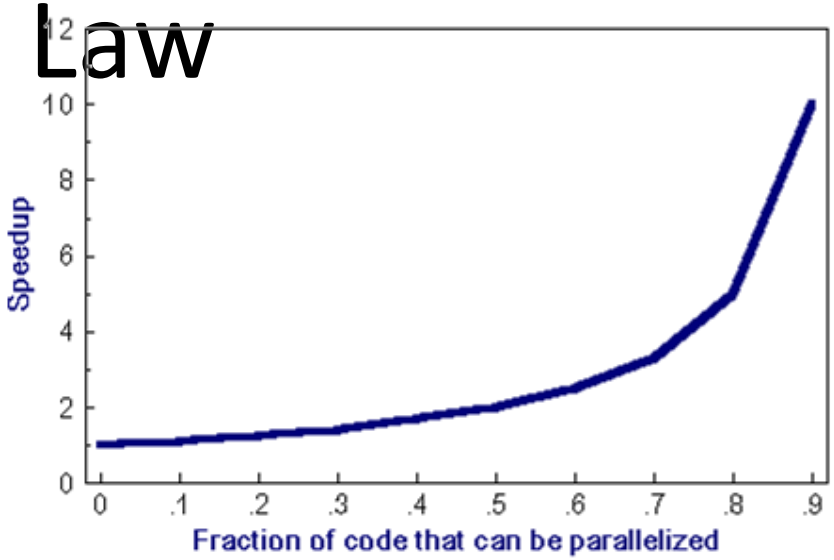
Calculation of the Fibonacci series (1,1,2,3,5,8,13,21,...) by use of the formula:

```
F(k + 2) = F(k + 1) + F(k)
```

Non-parallelizable

Amdahl's Law

- Amdahl's Law states that for a fixed workload, potential program speedup is defined by the fraction of code (q) that cannot be parallelized:



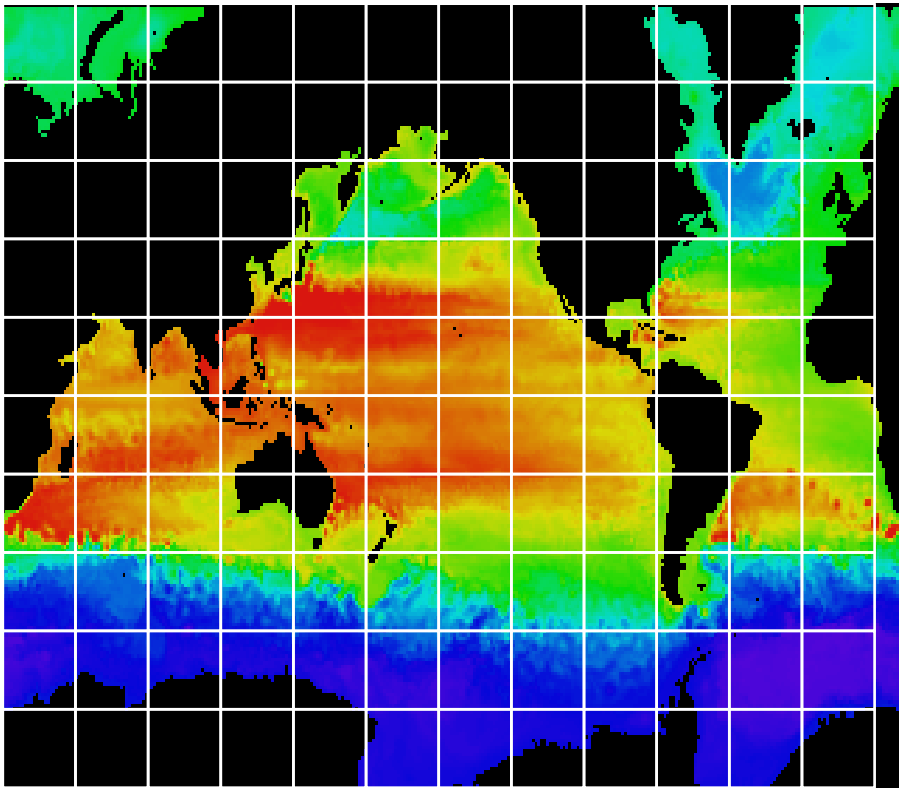
$$Speedup = \frac{1}{q}$$

- If none of the code can be parallelized, q = 1 and the speedup = 1 (no speedup). If all of the code is parallelized, q = 0 and the speedup is infinite (in theory).
- If 50% of the code can be parallelized, maximum speedup = 2, meaning the code will run twice as fast.

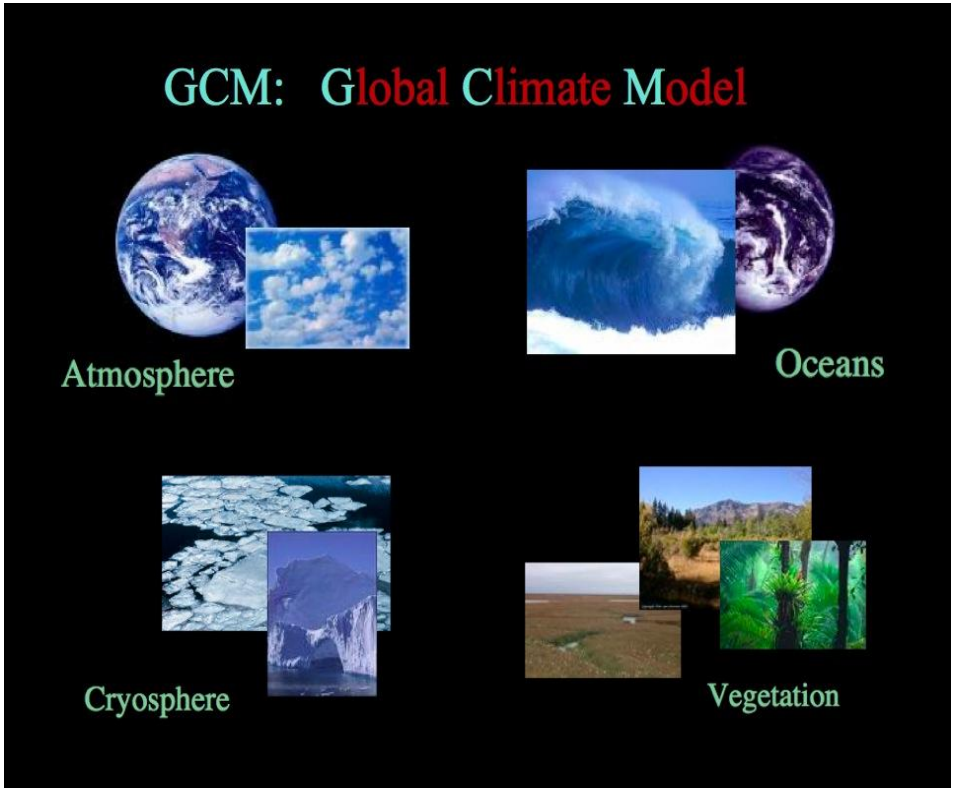
Steps in Developing Parallel Programs

- Find the critical work flow / hotspots
  - Where most of the work is done
  - Use profilers and performance analyzers on the sequential code
- Identify data dependencies
  - Inhibitors to parallelization
- Restructure the sequential program
  - Remove bottlenecks such as I/O out of the hotspots
- Partition the problem
  - Domain or functional decomposition

Example of Problem Partitioning



Domain Decomposition
Processors in adjacent blocks communicate their result.



Functional Decomposition

Steps in Developing Parallel Programs

- Arrange for Inter-process Communications
- Do we need communication?
  -  Embarrassingly Parallel problems
    - Example: Inverting image color
  -  Most of the problems need communication
    - Example: 3D Heat diffusion
- Communications are Overhead
  - Reduce the data to be moved
  - Avoid unnecessary communication in the code
  - Overlap communication with computation or communication

Concepts of Communication

- Latency & Bandwidth
- Explicit vs. implicit
- Synchronous vs. Asynchronous
- Scope
  - Point-to-point
  - Collective