Parallel and Distributed Computing

Lecture 1

Introduction

Aim of the course

to understand the fundamental concepts of parallel and distributed computing

design and analysis of Parallel algorithms

analyze different problems and develop parallel programming solutions of those problems

Study the challenges of Parallel and Distributed systems and how to cope with them

Outline

- Motivating Parallelism
- Computing vs Systems
- Parallel vs Distributed Computing
- Practical Applications of P&D Computing

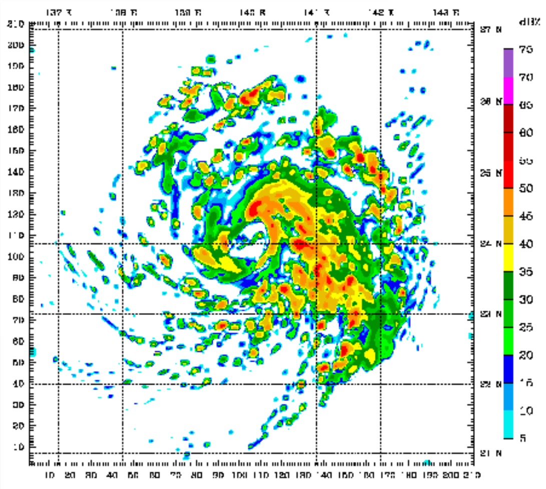
Lecture1: Introduction to Parallel Computing

Motivating Parallelism

Motivation for Parallel and Distributed Computing

- Uniprocessor are fast but
  - Some problems require too much computation
  - Some problems use too much data
  - Some problems have too many parameters to explore

- For example
  - Weather simulations, gaming, web Servers, code breaking



Why Parallel Computing

- **Problems with large computing complexity**
  - Computing hard problems (NP-complete problems)
    - exponential computing time.
  - Problems with large scale of input size
    - quantum chemistry, statistic mechanics, relative physics, universal physics, fluid
    - mechanics, biology, genetics engineering, …

For example, it costs 1 hour using the current computer to simulate the procedure of 1 second reaction of protein molecule and water molecule. It costs

1.14×10^8 years to simulate the procedure of 1 hour reaction.

Why parallel Computing

- Physical limitation of CPU computational power
- In past 50 years, CPU was speeded up double every 2.5 years. But, there are a physical limitation. Light speed is $3 \times 10^8 m/\sec$.
- Therefore, the limitation of the number of CPU clocks is expected to be about 10GHz.

- To solve computing hard problems
  - Parallel processing
  - DNA computer
  - Quantum computer

…

Parallel Computing

- **Parallel computing** is a computing where the jobs are broken into discrete parts that can be executed concurrently.
- Each part is further broken down to a series of instructions. Instructions from each part execute simultaneously on different CPUs.
- Parallel systems deal with the simultaneous use of multiple computer resources that can include a single computer with multiple processors, a number of computers connected by a network to form a parallel processing cluster or a combination of both.
- Parallel systems are more difficult to program than computers with a single processor because the architecture of parallel computers varies accordingly and the processes of multiple CPUs must be coordinated and synchronized.

Motivating Parallelism

- Developing parallel hardware and software has traditionally been time and effort intensive.
- If one is to view this in the context of rapidly improving uniprocessor speeds, one is tempted to question the need for parallel computing.
- Latest trends in hardware design indicate that uni-processors may not be able to sustain the rate of *realizable* performance increments in the future .
- This is the result of a number of fundamental physical and computational limitations.
- The emergence of standardized parallel programming environments, libraries, and hardware have significantly reduced time to develop (parallel) solution.

Motivating Parallelism

Moore's Law

- Proposed by Gorden E. Moore in 1965 and revised in 1975.

- **It states that [Simplified Version]**

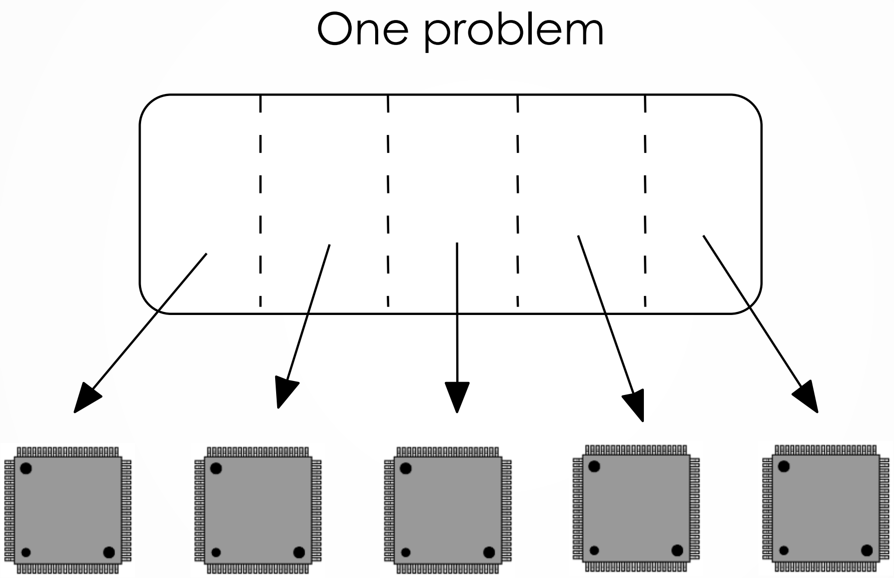
  "Processing speeds, or overall processing power for computers will double every18 months."

- **A more technically correct interpretation**

  "The number of transistors on an affordable CPU would double every two years [18 months]."

What is parallel computing

- Using a number of processors to process one task
- Speeding up the processing by distributing it to the processors



Classification of parallel computers

- M.J. Flynn proposed a classification for the organization of a computer system by the number of instructions and data items that are manipulated simultaneously.
- The sequence of instructions read from memory constitutes an **instruction stream**.
- The operations performed on the data in the processor constitute a **data stream**.

Two kinds of classification

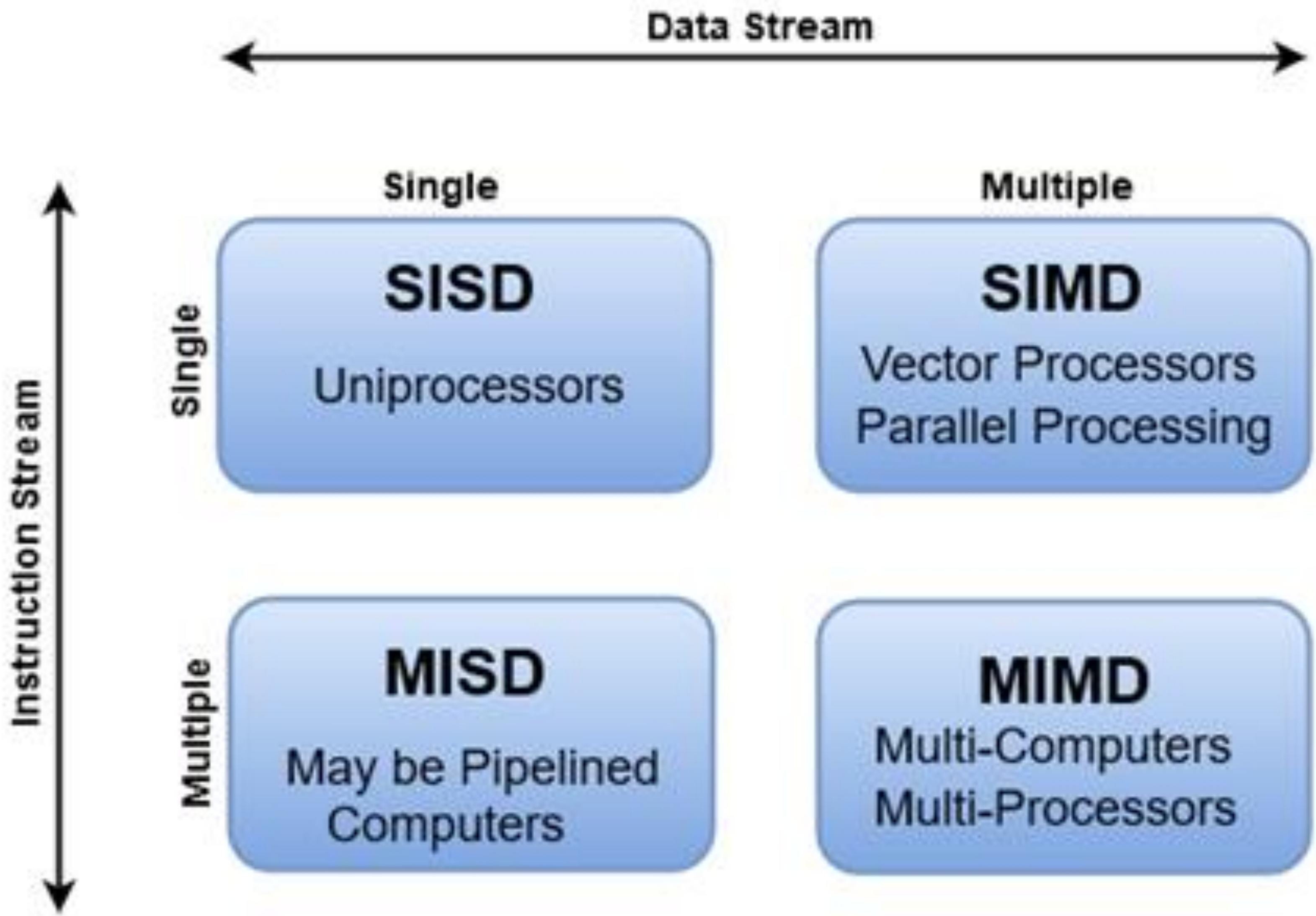
1. Flann's Classification

- SISD (Single Instruction stream, Single Data stream)
- MISD (Multiple Instruction stream, Single Data stream)
- SIMD (Single Instruction stream, Multiple Data stream)
- MIMD (Multiple Instruction stream, Multiple Data stream)

2. Classification by memory status

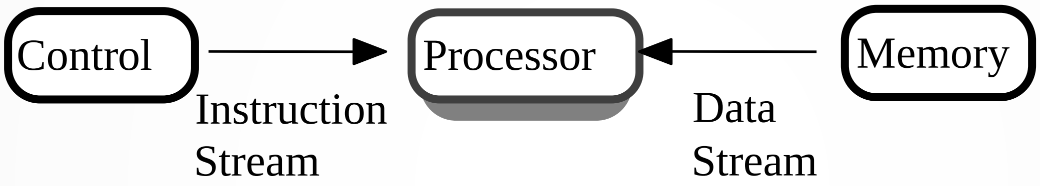
- Share memory
- Distributed memory

Flynn's Classification of Computers



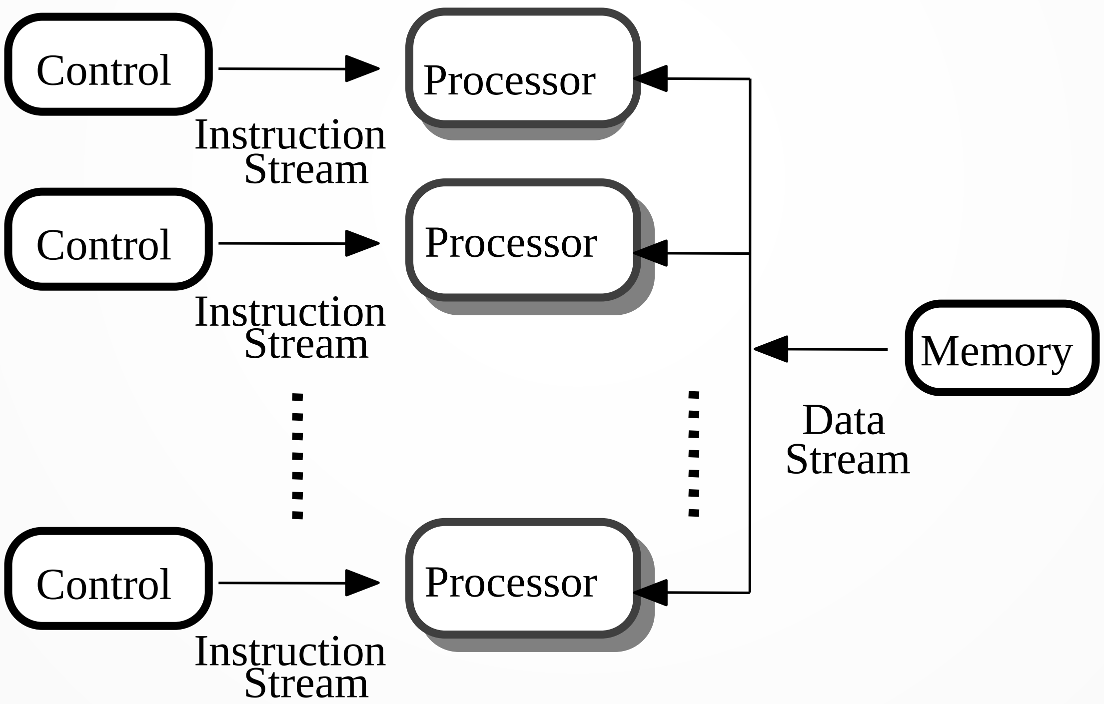
Flynn's classification(1)

- SISD (Single Instruction Single Data) computer
  - Von Neuman's one processor computer



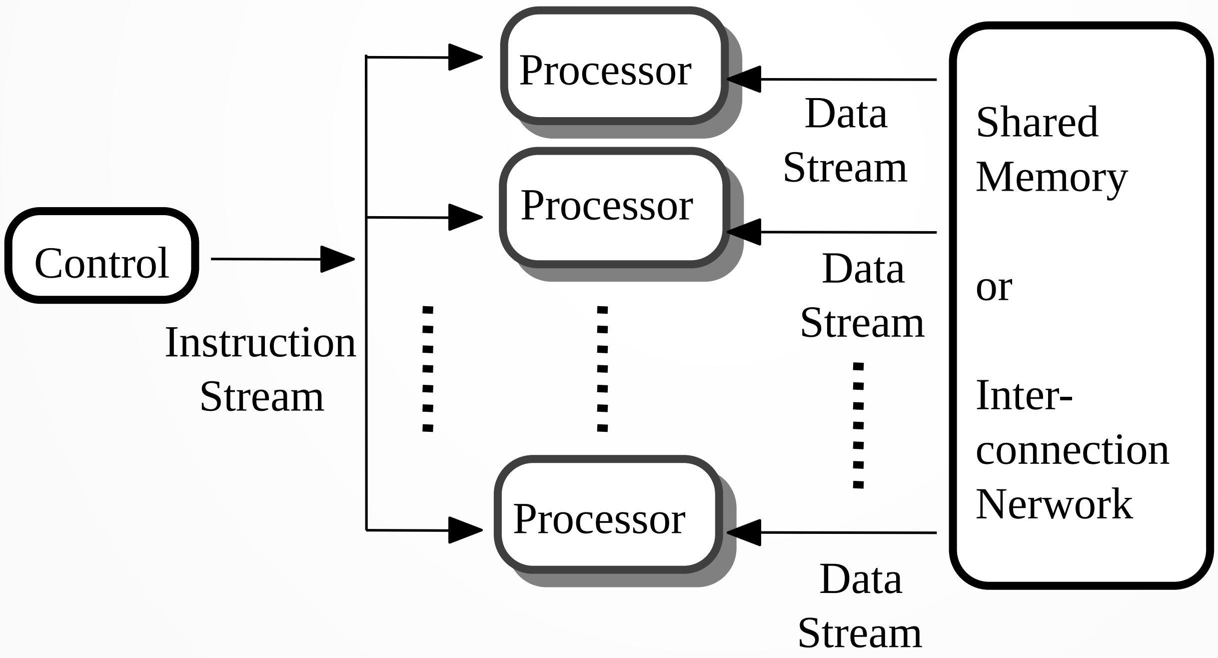
Flynn’s classification (2)

- MISD (Multi Instructions Single Data) computer
- All processors share a common memory, have their own control devices and execute their own instructions on same data.



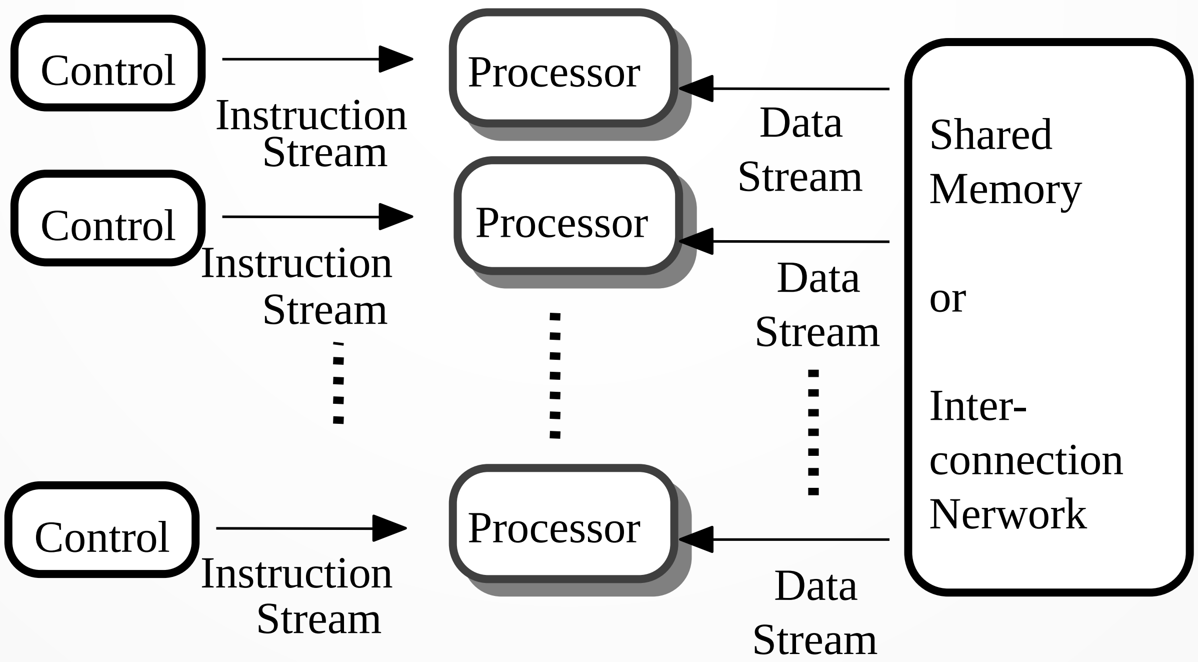
Flynn's classification (3)

- SIMD (Single Instructions Multi Data) computer
  - Processors execute the same instructions on different data
  - Operations of processors are synchronized by global clock.



Flynn’s classification (4)

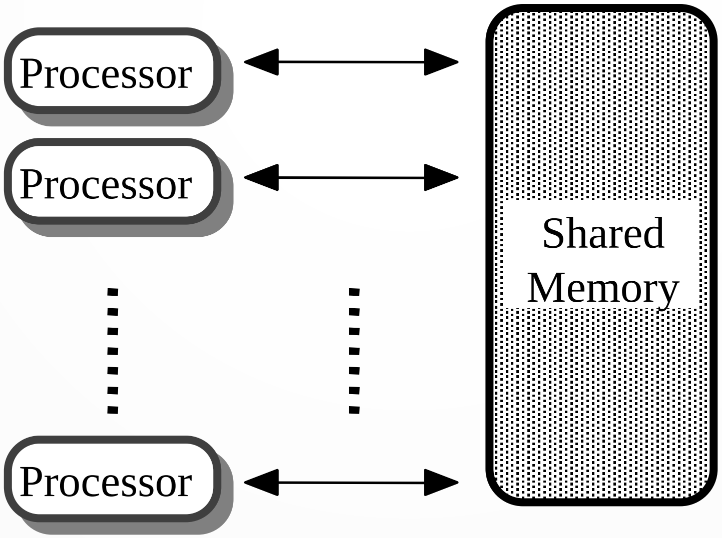
- MIMD (Multi Instruction Multi Data) Computer
  - Processors have their own control devices, and execute different instructions on different data.
  - Operations of processors are executed asynchronously in most time.
  - It is also called as distributed computing system.



Classification by memory types (1)

- 1. Parallel computer with a shared common memory
  - Communication based on shared memory

For example, consider the case that processor *i* sends some data to processor *j*. First, processor *i* writes the data to the share memory, then processor *j* reads the data from the same address of the share memory.



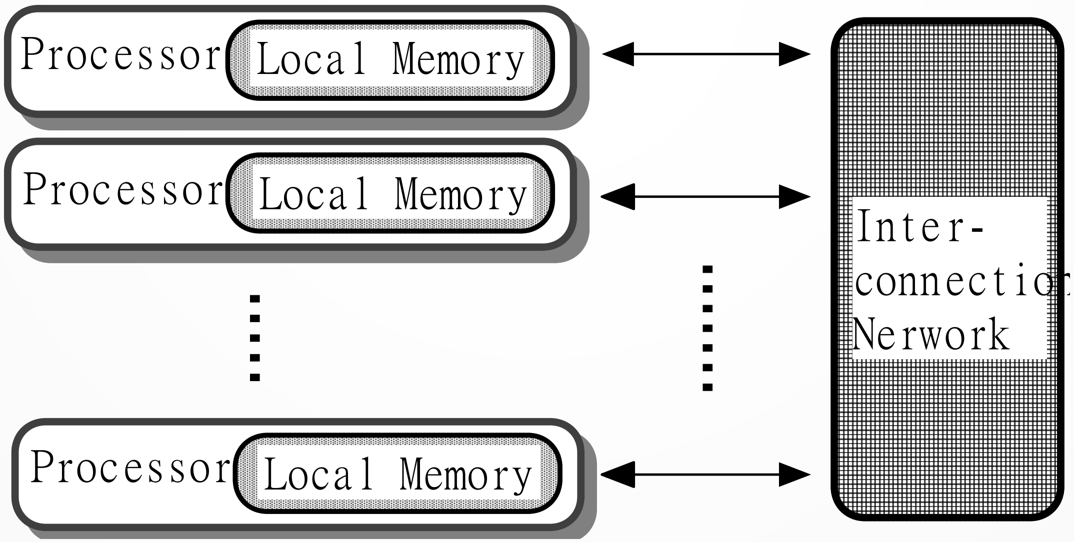
Classification by memory types (2)

- Features of parallel computers with shared common memory
  - Programming is easy.
  - Exclusive control is necessary for the access to the same memory cell.
  - Realization is difficult when the number of processors is large.

    （Reason) The number of processors connected to shared memory is limited by the physical factors such as the size and voltage of units, and the latency caused in memory accessing.

Classification by memory tapes (3)

- 2. Parallel computers with distributed memory
  - Communication style is one to one based on interconnection network.
  - For example, consider the case that processor *i* sends data to processor *j*. First, processor *i* issues a send command such as “processor *j* sends xxx to process *i*”, then processor *j* gets the data by a receiving command.



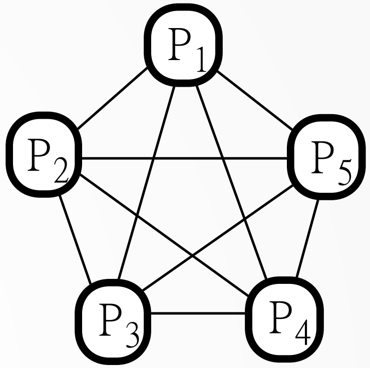
Classification by memory types (4)

- Features of parallel computers using distributed memory

- There are various architectures of interconnection networks (Generally, the degree of connectivity is not large.)
- Programming is difficult since comparing with shared common memory the communication style is one to one.
- It is easy to increase the number of processors.

Types of parallel computers with distributed memory

- Complete connection type



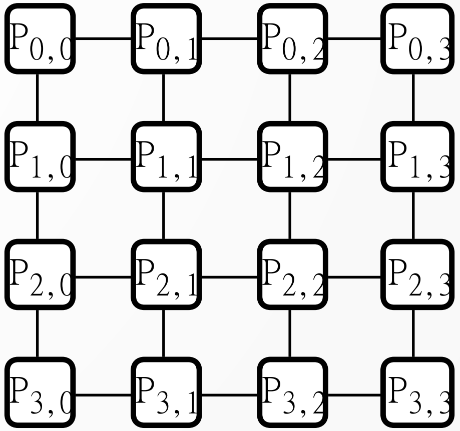
Any two processors are connected

Features

- Strong communication ability, but not practical
- (each processor has to be connected to many processors).

Mash connection type

Processors are connected as a two-dimension lattice.



Features

- Connected to few processors. Easily to increase the number of processors.

$\sqrt{n}$

- Large distance between processors:
- Existence of processor communication bottleneck.

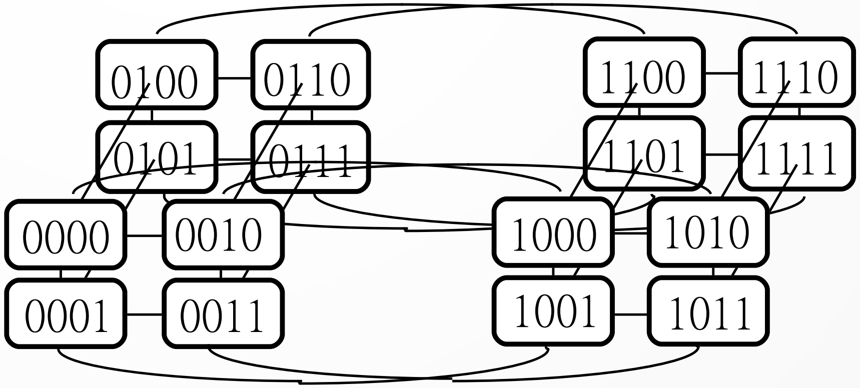
Types of parallel computers with distributed memory

Hypercube connection type

Processors connected as a hypercube (each processor has a binary number. (Processors are connected if only if one bit of their number are different.)

Features

- Small distance between processors: log *n*.
- Balanced communication load because of its symmetric structure.
- Easy to increase the number of processors.



Types of parallel computers with distributed memory

<u>Other connected type</u>

Tree connection type, butterfly connection type, bus connection type.

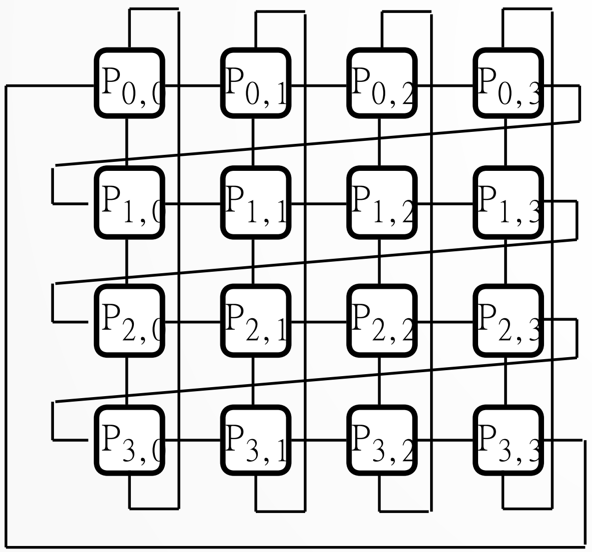
<u>Criterion for selecting an inter-connection network</u>

- Small diameter (the largest distance between processors) for small communication delay.
- Symmetric structure for easily increasing the number of processors.

The type of inter-connection network depends on application, ability of processors, upper bound of the number of processors and other factors.

Real parallel processing system (1)

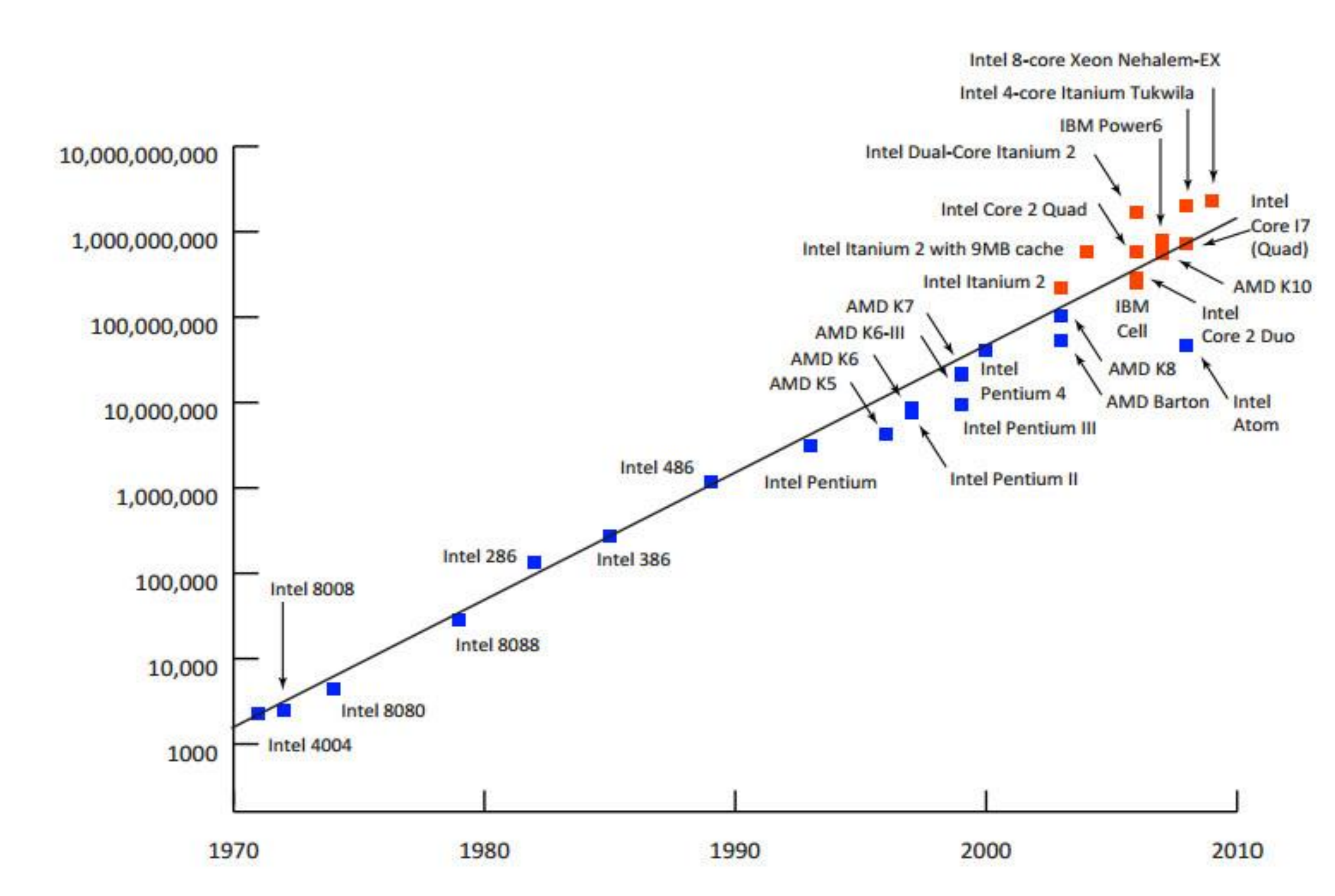
- Early days parallel computer (ILLIAC IV)
  - Built in 1972
  - SIMD type with distributed memory, consisting of 64 processors



Transformed mash connection type, equipped with common data bus, common control bus, and one control Unit.

Moore's law

- Number of transistors incorporated in a chip will approximately double every two years.



Motivating Parallelism

Moore's Law

- More computational power implicitly means more transistors.
- Then why need second interpretation?
- Let's have a look on empirical data from 1970 to 2009
  - In 1970's (i.e., from 1970 to 1979), processor speeds ranged from 740 KHz to 8 Mhz. Difference shows that both the interpretations are correct.
  - From 2000 to 2009, Speeds ranged from 1.3 GHz to 2.8 GHz.
  - Speed difference is too low but, number of integrated transistors ranged from 37.5 million to 904 million.
  - So, second interpretation is more accurate.

Motivating Parallelism

Moore's Law

- Why doubling the transistors does not doubles the speed?
  - The answer is increase in number of transistor per processor is due to multi-core CPU's.
  - It means, to follow Moore's law, companies had to:
    - Introduce ULSI(ultra large-scale integrations)
    - And multi-core processing era.
- Will Moore's law hold forever?
  - Adding multiple cores on single chip causes heat issues.
  - Furthermore, increasing the number of cores, may not be able to increase speeds [Due to inter-process interactions].
  - Moreover, transistors would eventually reach the limits of miniaturization at atomic levels

Motivating Parallelism

Moore's Law

- So, we must look for efficient parallel software solutions to fulfill our future computational needs.
- As stated earlier, number of cores on a single chip also have some restrictions.
- Solution[s]?
  - Need to find more scalable distributed and hybrid solutions

Motivating Parallelism

The Memory/Disk Speed Argument

- While clock rates of high-end processors have increased at roughly 40% per year over the past decade, DRAM access times have only improved at the rate of roughly 10% per year over this interval.
- This mismatch in speeds causes significant performance bottlenecks.
- Parallel platforms provide increased bandwidth to the memory system.
- Parallel platforms also provide higher aggregate caches.
- Some of the fastest growing applications of parallel computing utilize not their raw computational speed, rather their ability to pump data to memory and disk faster.

Motivating Parallelism

The Data Communication Argument

- As the network evolves, the vision of the Internet as one large computing platform has emerged.
- In many applications like databases and data mining problems, the volume of data is such that they cannot be moved.
- Any analyses on this data must be performed over the network using parallel techniques

Computing vs Systems

Distributed Systems

- A collection of autonomous computers, connected through a network and distribution middleware.
  - This enables computers to coordinate their activities and to share the resources of the system.
  - The system is usually perceived as a single, integrated computing facility.
  - Mostly concerned with the hardware-based accelerations

Distributed Computing

- A specific use of distributed systems, to split a large and complex processing into subparts and execute them in parallel, to increase the productivity.
  - Computing mainly concerned with software-based accelerations (i.e., designing and implementing algorithms)

Parallel vs Distributed Computing

Parallel (shared-memory) Computing

- The term is usually used for developing concurrent solutions for following two types of the systems:
  1. Multi-core Architecture
  2. Many core architectures (i.e., GPU's)

Distributed Computing

- This type of computing is mainly concerned with developing algorithms for the distributed cluster systems.
- Here distributed means a geographical distance between the computers without any shared-Memory.

Practical Applications of P&D Computing

Scientific Applications

- Functional and structural characterization of genes and proteins
- Applications in astrophysics have explored the evolution of galaxies, thermonuclear processes, and the analysis of extremely large datasets from telescope.
- Advances in computational physics and chemistry have explored new materials, understanding of chemical pathways, and more efficient processes
  - e.g., Large Hydron Collider (LHC) at European Organization for Nuclear Research (CERN) generates petabytes of data for a single collision.

Practical Applications of P&D Computing

Scientific Applications

- Bioinformatics and astrophysics also present some of the most challenging problems with respect to analyzing extremely large datasets.

- Weather modeling for simulating the track of a natural hazards like the extreme cyclones (storms).

- Flood prediction

Practical Applications of P&D Computing

Commercial Applications

- Some of the largest parallel computers power the wall street!
- Data mining-analysis for optimizing business and marketing decisions.
- Large scale servers (mail and web servers) are often implemented using parallel platforms.
- Applications such as information retrieval and search are typically powered by large clusters.

Practical Applications of P&D Computing

Computer Systems Applications

- Network intrusion detection: A large amount of data needs to be analyzed and processed
- Cryptography (the art of writing or solving codes) employs parallel infrastructures and algorithms to solve complex codes.
- Graphics processing
- Embedded systems increasingly rely on distributed control algorithms. E.g. modern automobiles

Limitations of Parallel Computing:

- It requires designing the proper communication and synchronization mechanisms between the processes and sub-tasks.
- Exploring the proper parallelism from a problem is a hectic process.
- The program must have low coupling and high cohesion. But it's difficult to create such programs.
- It needs relatively more technical skills to code a parallel program.

Questions