PARALLEL AND DISTRIBUTED COMPUTING

Shahzad Ali Rana
Lecturer GCUF
03006641562
/shahzad.rana47

PARALLEL AND DISTRIBUTED COMPUTING

INTRODUCTION

The simultaneous growth in the availability of big data and in the number of simultaneous users on the Internet places particular pressure on the need to carry out computing tasks "in parallel," or simultaneously. Parallel and distributed computing occur across many different topic areas in computer science, including algorithms, computer architecture, networks, operating systems, and software engineering. During the early 21st century there was explosive growth in multiprocessor design and other strategies for complex applications to run faster. Parallel and distributed computing buildon fundamental systems concepts. Such as concurrency, mutual exclusion, consistency in state/memory manipulation message-passing, and shared-memory models.

PARALLEL COMPUTING:

The simultaneous execution of the same task on multiple processors in order to obtain faster results is called parallel computing.

- HPC: High Performance/Productivity Computing
- Technical Computing
- Cluster computing

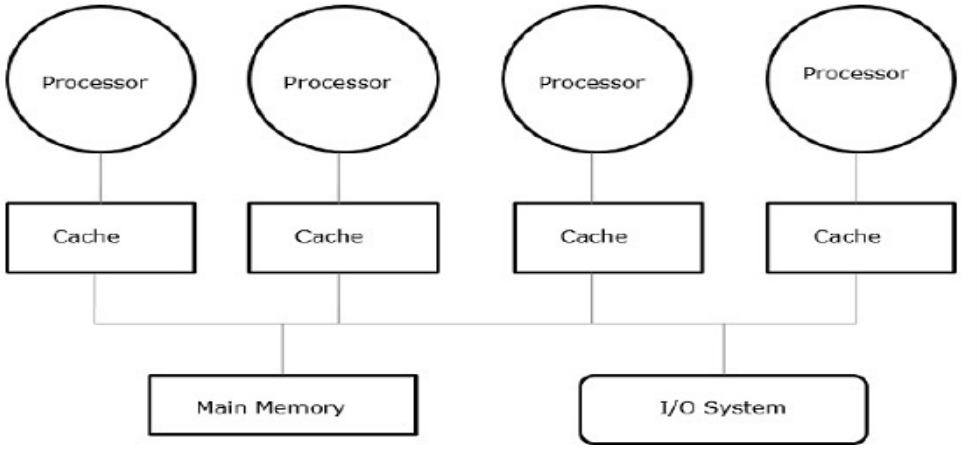
"In parallel computing, multiple processors perform multiple tasks assigned to them simultaneously. Memory in parallel systems can either be shared or distributed. Parallel computing provides concurrency and saves time and money."

DISTRIBUTED COMPUTING:

Uses or coordinates physically separate computing resources.

- Grid computing
- Cloud Computing

"In distributed computing, we have multiple autonomous computers which seem to the user as a single system. In distributed systems, there is no shared memory and computers communicate with each other through message passing. In distributed computing, a single task is divided among different computers."



DIFFERENCE BETWEEN PARALLEL COMPUTING AND DISTRIBUTED COMPUTING

Sr No.	PARALLEL COMPUTING	DISTRIBUTED COMPUTING
1	Many operations are performed simultaneously	System components are located at different locations
2	A single computer is required	Uses Multiple Computers
3	Multiple processors perform multiple operations	Multiple computers perform multiple operations
4	It may have shared or distributed memory	It has only distributed memory
5	Processors communicate with each other through the bus	Computers communicate with each other through message passing
6	Improves the system performance	Improves system scalability, fault tolerance, and resource-sharing capabilities

ADVANTAGES OF PARALLEL COMPUTING

The advantagesof Parallel Computing over Serial Computing are as follows:

1. It saves time and money as many resources working together will reduce the time and cut potential costs.
2. It can be impractical to solve larger problems on Serial Computing.
3. It can take advantage of non-local resources when the local resources are finite.
4. Serial Computing 'wastes' the potential computing power, thus Parallel Computing makes better work of hardware.

TYPES OF PARALLELISM

1. **BIT-LEVEL PARALLELISM:**
   It is a form of parallel computing which is based on the increasing processor's size. It reduces the number of instructions that the system must execute in order to perform a task on large-sized data.
   Example: Consider a scenario where an 8-bit processor must compute the sum of two 16-bit integers. It must first sum up the 8 lower-order bits, then add the 8 higher-order bits, thus requiring two instructions to perform the operation. A 16-bit processor can perform the operation with just one instruction.

2. **INSTRUCTION-LEVEL PARALLELISM:**

   A processor can only address less than one instruction for each clock cycle phase. These instructions can be re-ordered and grouped which are later on executed concurrently without affecting the result of the program. This is called instruction-level parallelism.

3. TASK PARALLELISM:

Task parallelism employs the decomposition of a task into subtasks and then allocating each of the subtasks for execution. The processors perform the execution of sub-tasks concurrently.

WHY PARALLEL COMPUTING?

- The whole real world runs in dynamic nature i.e. many things happen at a certain time but at different places concurrently. This data is extensively huge to manage.
- Real-world data needs more dynamic simulation and modeling, and for achieving the same, parallel computing is the key.
- Parallel computing provides concurrency and saves time and money.
- Complex, large datasets, and their management can be organized only and only using a parallel computing approach.
- Ensures the effective utilization of the resources. The hardware is guaranteed to be used effectively whereas in the serial computation only some part of the hardware was used and the rest was rendered idle.
- Also, it is impractical to implement real-time systems using serial computing.

APPLICATIONS OF PARALLEL COMPUTING:

- Databases and Data mining.
- Real-time simulation of systems.
- Science and Engineering.
- Advanced graphics, augmented reality, and virtual reality.

LIMITATIONS OF PARALLEL COMPUTING:

- It addresses such as communication and synchronization between multiple sub-tasks and processes which is difficult to achieve.
- The algorithms must be managed in such a way that they can be handled in a parallel mechanism.
- The algorithms or program must have low coupling and high cohesion. But it's difficult to create such programs.
- More technically skilled and expert programmers can code a parallelism-based program well.

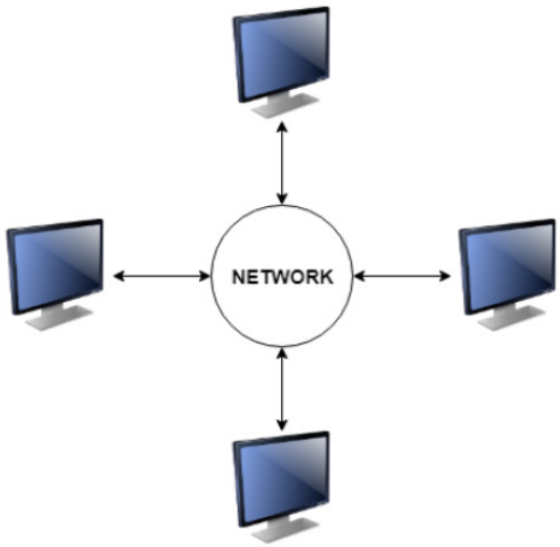
FUTURE OF PARALLEL COMPUTING:

The computational graph has undergone a great transition from serial computing to parallel computing. Tech giant such as Intel has already taken a step towards parallel computing by employing multicore processors.

Parallel computation will revolutionize the way computers work in the future, for the better good. With the world connecting to each other even more than before, Parallel Computing does a better role in helping us stay that way. With faster networks, distributed systems, and multi-processor computers, it becomes even more necessary.

WHAT IS DISTRIBUTED SYSTEM?

A distributed system contains multiple nodes that are physically separate but linked together using the network. All the nodes in this system communicate with each other and handle processes in tandem. Each of these nodes contains a small part of the distributed operating system software.
A diagram to better explain the distributed system is –



DISTRIBUTED OPERATING SYSTEM

TYPES OF DISTRIBUTED SYSTEMS

The nodes in the distributed systems can be arranged in the form of client/server systems or peer to peer systems. Details about these are as follows –

CLIENT/SERVER SYSTEMS:

In client-server systems, the client requests a resource and the server provides that resource. A server may serve multiple clients at the same time while a client is in contact with only one server. Both the client and server usually communicate via a computer network and so they are a part of distributed systems.

PEER-TO-PEER SYSTEMS:

The peer-to-peer systems contain nodes that are equal participants in data sharing. All the tasks are equally divided between all the nodes. The nodes interact with each other as required to share resources. This is done with the help of a network.

ADVANTAGES OF DISTRIBUTED SYSTEMS

Some advantages of Distributed Systems are as follows:
All the nodes in the distributed system are connected to each other. So nodes can easily share data with other nodes.
More nodes can easily be added to the distributed system i.e. it can be scaled as required.
Failure of one node does not lead to the failure of the entire distributed system. Other nodes can still communicate with each other.
Resources like printers can be shared with multiple nodes rather than being restricted to just one.

DISADVANTAGES OF DISTRIBUTED SYSTEMS

Some disadvantages of Distributed Systems are as follows:

- It is difficult to provide adequate security in distributed systems because the nodes as well as the connections need to be secured.
- Some messages and data can be lost in the network while moving from one node to another.
- The database connected to the distributed systems is quite complicated and difficult to handle as compared to a single-user system.
- Overloading may occur in the network if all the nodes of the distributed system try to send data at once.

DISTRIBUTED COMPUTING MODELS

There are certain technologies working behind the cloud computing platforms making cloud computing flexible, reliable, and usable. These technologies are listed below:

- Virtualization
- Service-Oriented Architecture (SOA)
- Grid Computing
- Utility Computing

WHY USE PARALLEL AND DISTRIBUTED SYSTEM?

When to Use Parallel Computing:

This computing method is ideal for anything involving complex simulations or modeling. Common applications for it include seismic surveying, computational astrophysics, climate modeling, financial risk management, agricultural estimates, video color correction, medical imaging, drug discovery, and computational fluid dynamics.

When to Use Distributed Computing:

Distributed computing is best for building and deploying powerful applications running across many different users and geographies. Anyone performing a Google search is already using distributed computing. Distributed system architectures have shaped much of what we would call "modern business," including cloud-based computing, edge computing, and software as a service (SaaS).

WHERE ARE THEY USED?

Parallel computing is often used in places requiring higher and faster processing power. For example, supercomputers.

Since there are no lags in the passing of messages, these systems have high speed and efficiency.

Distributed computing is used when computers are located at different geographical locations.

In these scenarios, speed is generally not a crucial matter. They are the preferred choice when scalability is required.

DIFFERENT USES OF PARALLEL AND DISTRIBUTED COMPUTING SYSTEMS

Parallel Computing	Distributed Computing
Typically consists of a network of computers	Mainly uses several processors
Used to share resources	Used for high performance
Used for consistency and availability	Used for concurrency
Designed to tolerate failures	Designed for massive calculations

WHICH IS BETTER: PARALLEL OR DISTRIBUTED COMPUTING?

It's hard to say which is "better"—parallel or distributed computing—because it depends on the use case (see section above). If you need pure computational power and work in a scientific or other types of highly analytics-based field, then you're probably better off with parallel computing. If you need scalability and resilience and can afford to support and maintain a computer network, then you're probably better off with distributed computing.

APPLICATIONS OF PARALLEL AND DISTRIBUTED COMPUTING?

APPLICATIONS OF PARALLEL COMPUTING

Notable Applications of Parallel Processing (Also known as Parallel Computing) Include:

- **Databases and Data Mining.**
  The concept of parallel computing is based on dividing a large problem into smaller ones and each of them is carried out by one single processor individually. In addition, these processes are performed concurrently in a distributed and parallel manner.

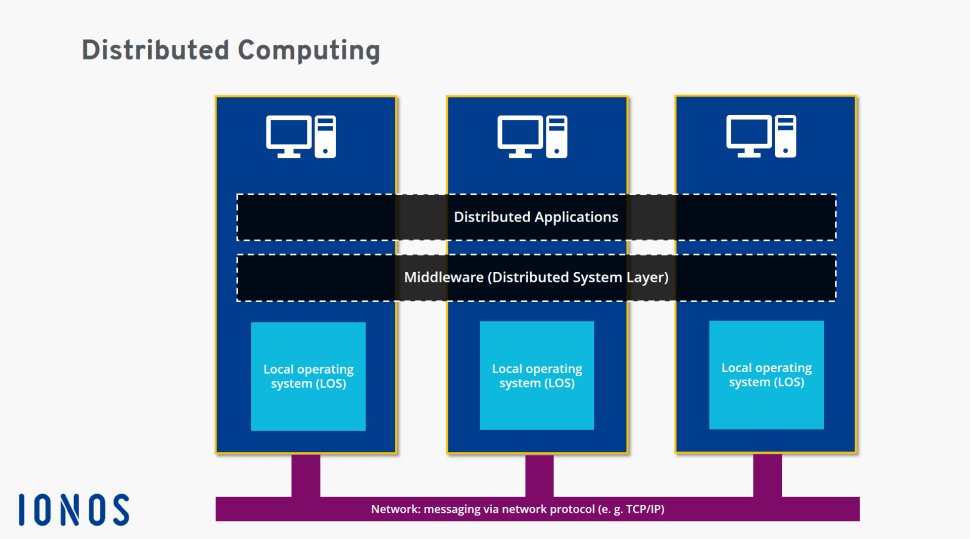
- **Real-Time Simulation of Systems.**
  Space Parallel Simulation is a more flexible and widely applicable strategy for reducing the execution time, but also for increasing the simulation scalability. It partitions the network to simulate in subnetworks and assigns logical processes to those that can be concurrently executed on different processors.

- **Science and Engineering.**
  The advantages of parallel computing are that computers can execute code more efficiently, which can save time and money by sorting through "big data" faster than ever. Parallel programming can also solve more complex problems, bringing more resources to the table.

- **Advanced Graphics, Augmented Reality, and Virtual Reality.**
  The cost in computing cycles for state-of-the-art imagery continues to rise exponentially. Fortunately, so does the cost-effectiveness of state-of-the-art computer processors. However, that leaves us continually several orders of magnitude short on processing power if we wish to make state-of-the-art images in real-time or even interactive time. An obvious answer to this dilemma lies in parallel processing.

APPLICATIONS OF DISTRIBUTED COMPUTING

Social networks, mobile systems, online banking, and online gaming (e.g. multiplayer systems) also use efficient distributed systems. Additional areas of application for distributed computing include e-learning platforms, artificial intelligence, and e-commerce.



OVERVIEW OF THE CHAPTER AND EXAMPLES

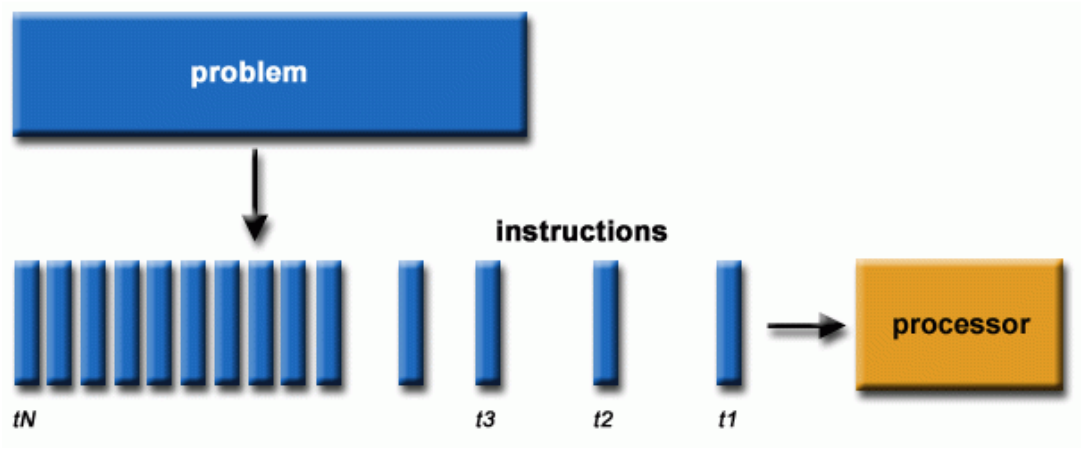
OVERVIEW

WHAT IS PARALLEL COMPUTING?

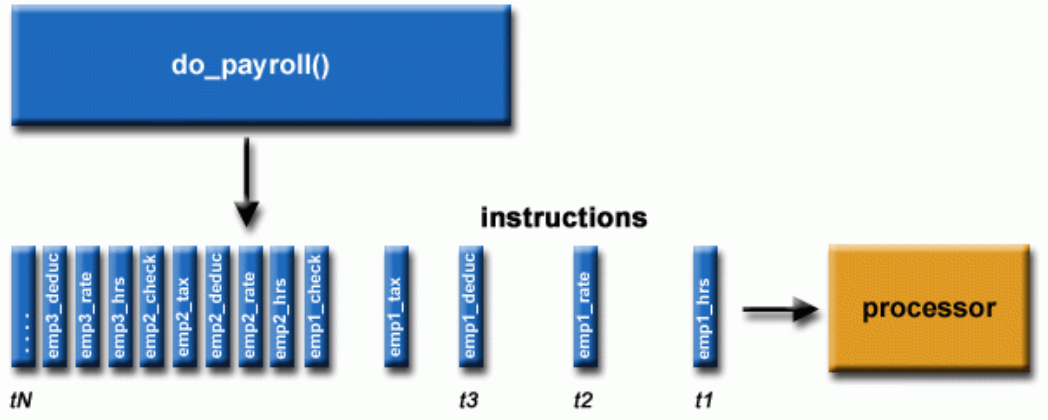
SERIAL COMPUTING

Traditionally, the software has been written for *serial* computation:

- A problem is broken into a discrete series of instructions
- Instructions are executed sequentially one after another
- Executed on a single processor
- Only one instruction may execute at any moment in time



FOR EXAMPLE:



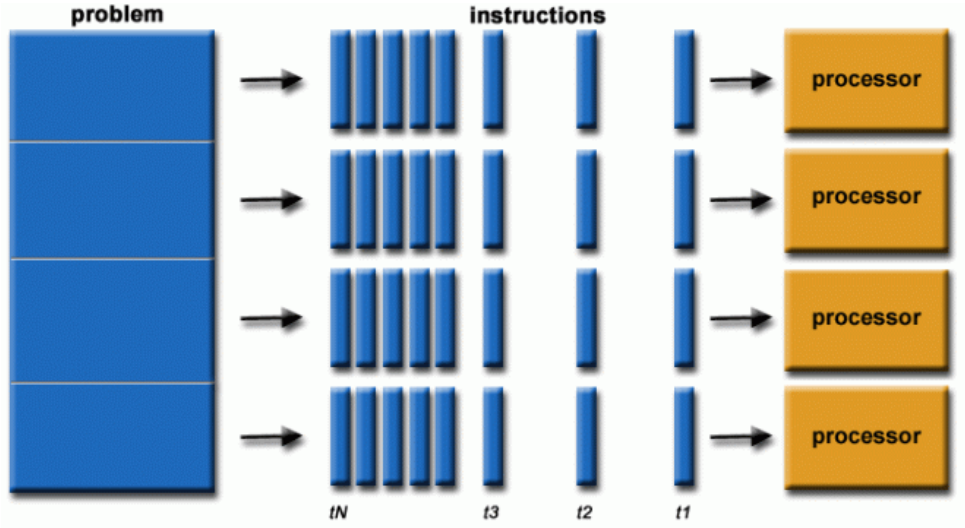
Serial computing example of processing payroll

PARALLEL COMPUTING

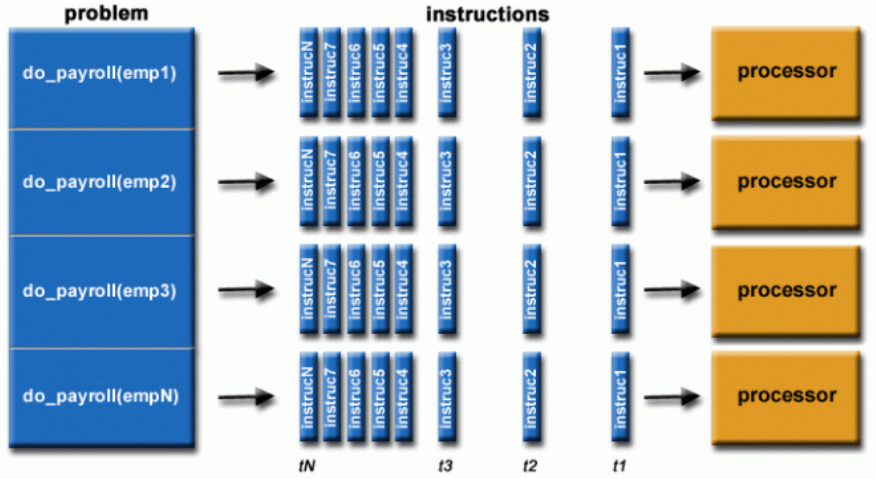
In the simplest sense, ***parallel computing*** is the simultaneous use of multiple computing resources to solve a computational problem:

- A problem is broken into discrete parts that can be solved concurrently
- Each part is further broken down to a series of instructions
- Instructions from each part execute simultaneously on different processors

- An overall control/coordination mechanism is employed



FOR EXAMPLE:

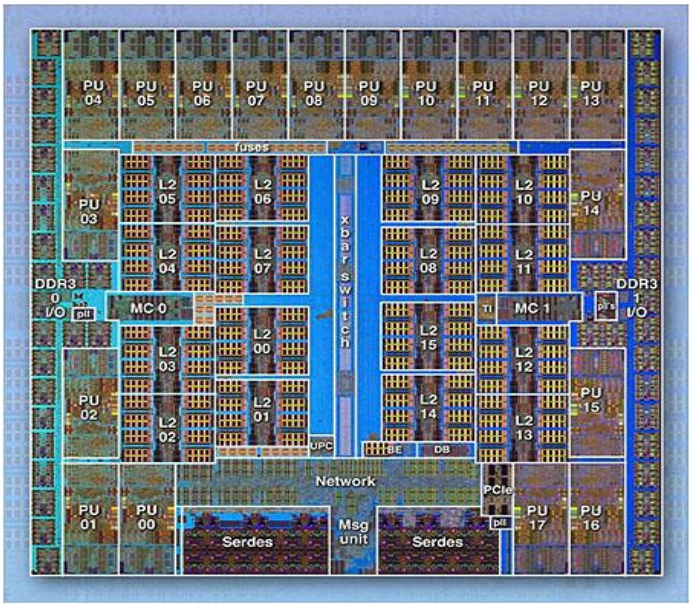


Parallel computing example of processing payroll

- The computer resources are typically:
  - A single computer with multiple processors/cores
  - An arbitrary number of such computers connected by a network

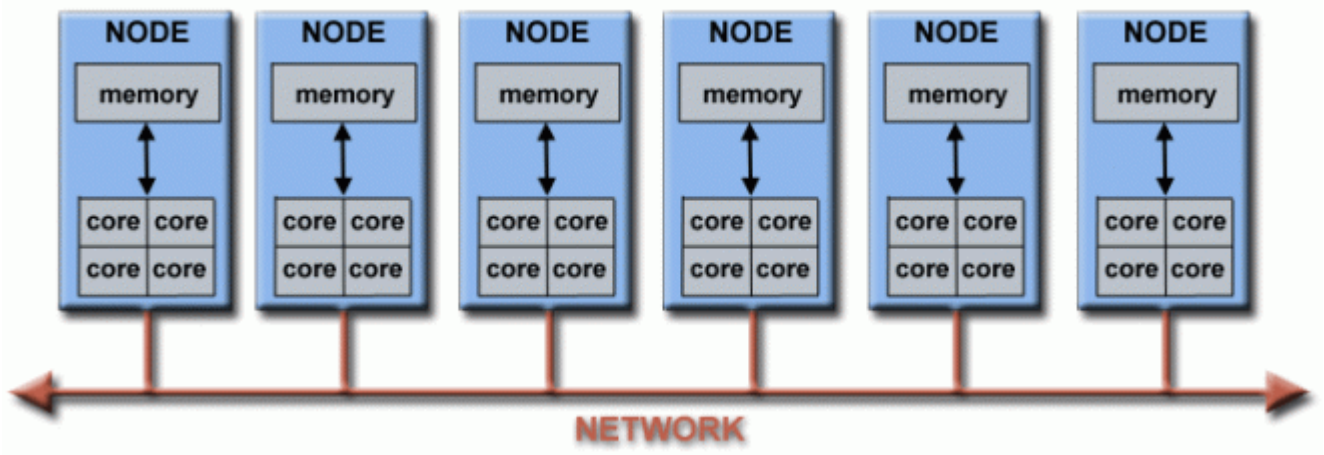
PARALLEL COMPUTERS

- Virtually all stand-alone computers today are parallel from a hardware perspective:
  - Multiple functional units (L1 cache, L2 cache, branch, pre-fetch, decode, floating-point, graphics processing (GPU), integer, etc.)
  - Multiple execution units/cores
  - Multiple hardware threads



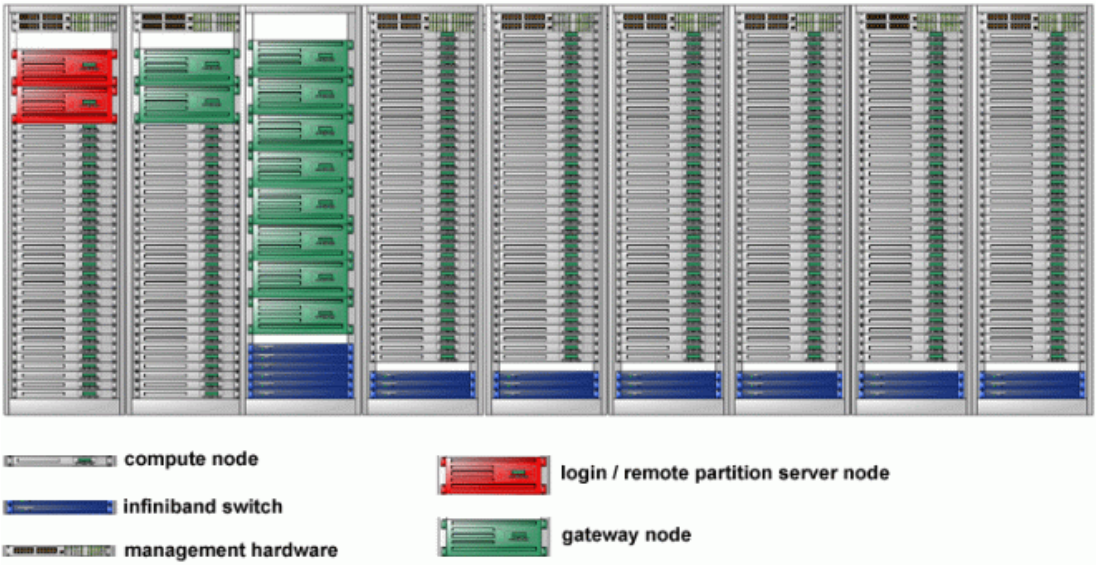
IBM BG/Q Compute Chip with 18 cores (PU) and 16 L2 Cache units (L2)

- Networks connect multiple stand-alone computers (nodes) to make larger parallel computer clusters.



NETWORK CONNECTIONS

- **For example**, the schematic below shows a typical LLNL parallel computer cluster:
  - Each compute node is a multi-processor parallel computer in itself
  - Multiple compute nodes are networked together with an Infiniband network
  - Special purpose nodes, also multi-processor, are used for other purposes



Example of typical parallel computer cluster

- The majority of the world's large parallel computers (supercomputers) are clusters of hardware produced by a handful of (mostly) well known vendors.

THE REAL WORLD IS MASSIVELY COMPLEX

- In the natural world, many complex, interrelated events are happening at the same time, yet within a temporal sequence.
- Compared to serial computing, parallel computing is much better suited for modeling, simulating and understanding complex, real world phenomena.
- For example, imagine modeling these serially:

Real world phenomena can be simulated with parallel computing

Real world phenomena can be simulated with parallel computing

MAIN REASONS FOR USING PARALLEL PROGRAMMING
SAVE TIME AND/OR MONEY

- In theory, throwing more resources at a task will shorten its time to completion, with potential cost savings.
- Parallel computers can be built from cheap, commodity components.

Working in parallel shortens completion time

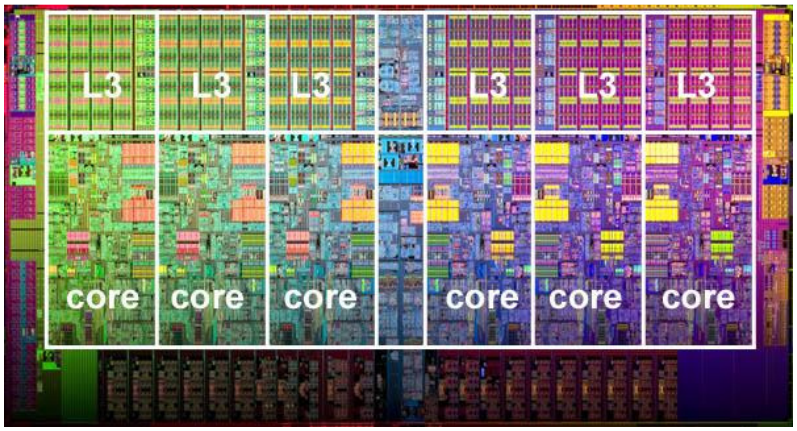
PROVIDE CONCURRENCY

- A single compute resource can only do one thing at a time. Multiple compute resources can do many things simultaneously.
- Example: Collaborative Networks provide a global venue where people from around the world can meet and conduct work "virtually."

Collaborative networks

MAKE BETTER USE OF UNDERLYING PARALLEL HARDWARE

- Modern computers, even laptops, are parallel in architecture with multiple processors/cores.
- Parallel software is specifically intended for parallel hardware with multiple cores, threads, etc.
- In most cases, serial programs run on modern computers "waste" potential computing power.



THE FUTURE

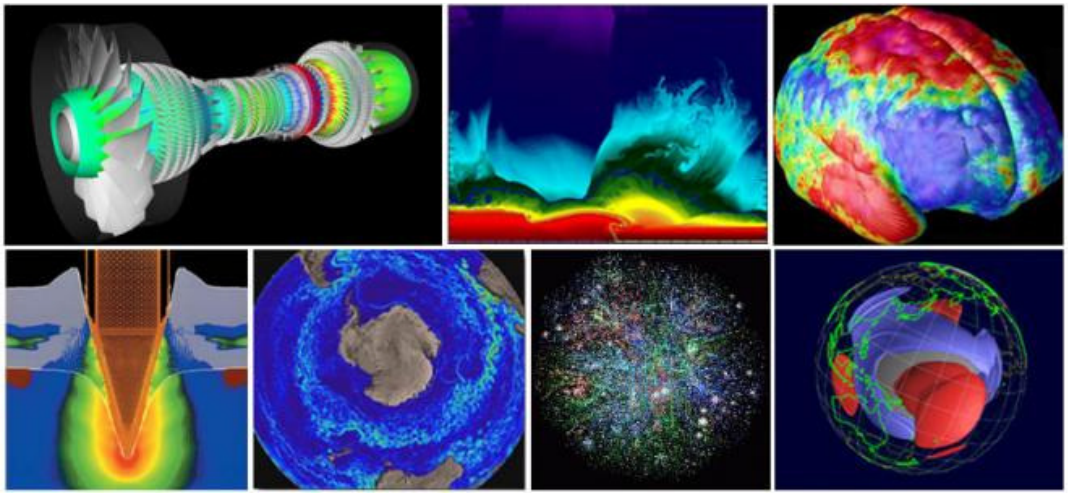
- During the past 20+ years, the trends indicated by ever faster networks, distributed systems, and multi-processor computer architectures (even at the desktop level) clearly show that ***parallelism is the future of computing***.
- In this same time period, there has been a greater than **500,000x** increase in supercomputer performance, with no end currently in sight.

WHO IS USING PARALLEL COMPUTING?

SCIENCE AND ENGINEERING

Historically, parallel computing has been considered to be "the high end of computing," and has been used to model difficult problems in many areas of science and engineering:

- Atmosphere, Earth, Environment
- Physics - applied, nuclear, particle, condensed matter, high pressure, fusion, photonics
- Bioscience, Biotechnology, Genetics
- Chemistry, Molecular Sciences
- Geology, Seismology
- Mechanical Engineering - from prosthetics to spacecraft
- Electrical Engineering, Circuit Design, Microelectronics
- Computer Science, Mathematics
- Defense, Weapons

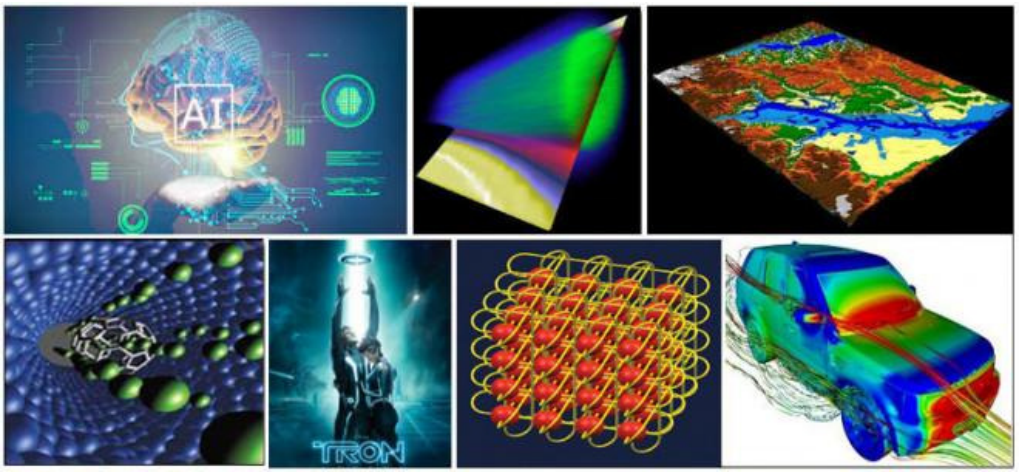


Parallel computing is key to simulating a range of complex physical phenomena

INDUSTRIAL AND COMMERCIAL

Today, commercial applications provide an equal or greater driving force in the development of faster computers. These applications require the processing of large amounts of data in sophisticated ways. For example:

- "Big Data," databases, data mining
- Artificial Intelligence (AI)
- Oil exploration
- Web search engines, web based business services
- Medical imaging and diagnosis
- Pharmaceutical design
- Financial and economic modeling
- Management of national and multi-national corporations
- Advanced graphics and virtual reality, particularly in the entertainment industry
- Networked video and multi-media technologies
- Collaborative work environments



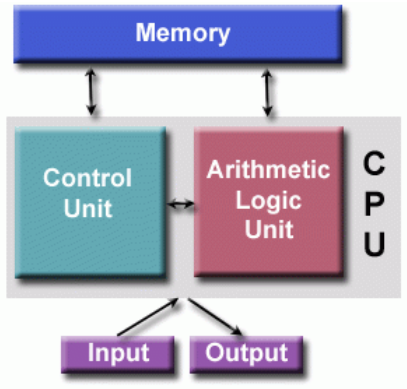
Parallel computing is used in many commercial applications

CONCEPTS AND TERMINOLOGY

von Neumann Computer Architecture

John von Neumann circa 1940s
(Source: LANL archives)

- Named after the Hungarian mathematician John von Neumann who first authored the general requirements for an electronic computer in his 1945 papers.
- Also known as "stored-program computer" - both program instructions and data are kept in electronic memory. This differs from earlier computers which were programmed through "hard wiring".
- Since then, virtually all computers have followed this basic design:



Basic computing architecture

- Comprised of four main components:

1. Memory
2. Control Unit
3. Arithmetic Logic Unit
4. Input/Output

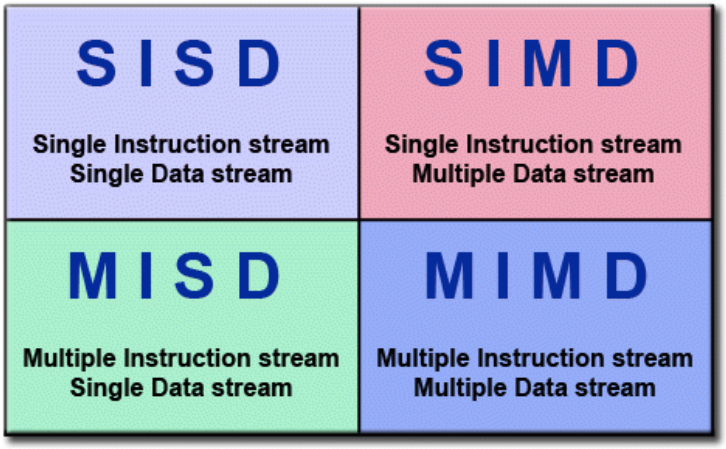
- Read/write, random access memory is used to store both program instructions and data
- Program instructions are coded data which tell the computer to do something
- Data is simply information to be used by the program

- Control unit fetches instructions/data from memory, decodes the instructions and then ***sequentially*** coordinates operations to accomplish the programmed task.
- Arithmetic Unit performs basic arithmetic operations
- Input/Output is the interface to the human operator

Parallel computers still follow this basic design, just multiplied in units. The basic, fundamental architecture remains the same.

FLYNN'S CLASSICAL TAXONOMY

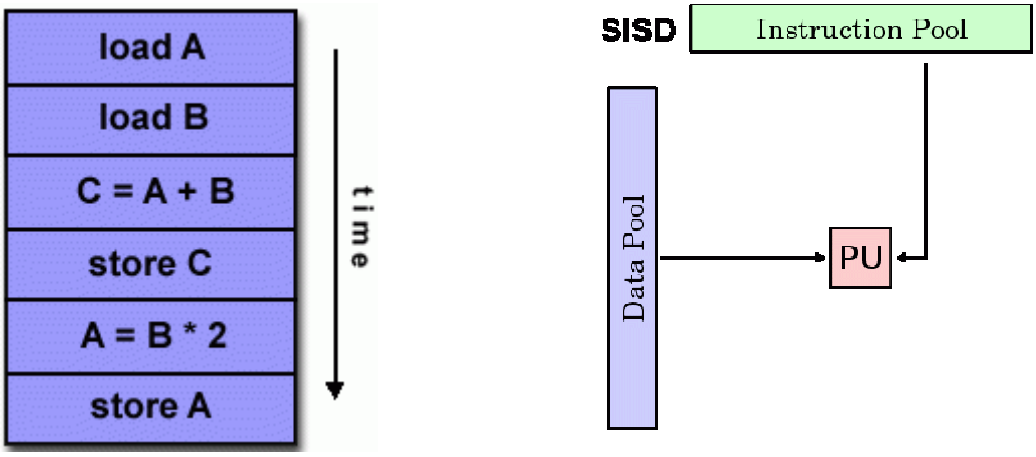
- There are a number of different ways to classify parallel computers. Examples are available in the references.
- One of the more widely used classifications, in use since 1966, is called Flynn's Taxonomy.
- Flynn's taxonomy distinguishes multi-processor computer architectures according to how they can be classified along the two independent dimensions of ***Instruction Stream*** and ***Data Stream***. Each of these dimensions can have only one of two possible states: ***Single*** or ***Multiple***.
- The matrix below defines the 4 possible classifications according to Flynn:



Flynn's taxonomy

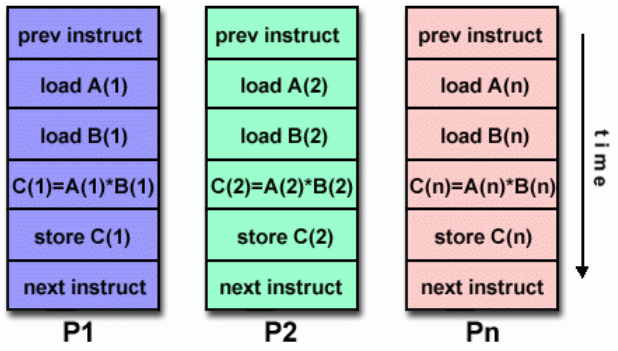
SINGLE INSTRUCTION, SINGLE DATA (SISD)

- A serial (non-parallel) computer
- **Single Instruction:** Only one instruction stream is being acted on by the CPU during any one clock cycle
- **Single Data:** Only one data stream is being used as input during any one clock cycle
- Deterministic execution
- This is the oldest type of computer
- Examples: older generation mainframes, minicomputers, workstations and single processor/core PCs.



SINGLE INSTRUCTION, MULTIPLE DATA (SIMD)

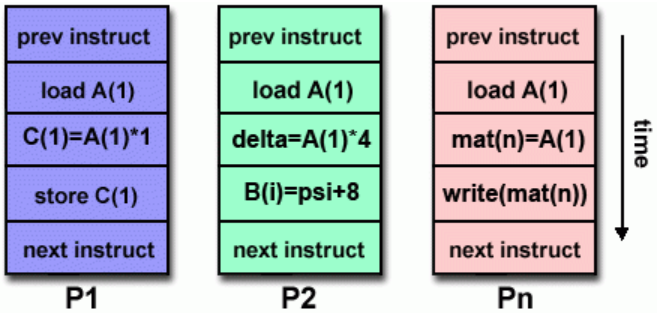
- A type of parallel computer
- **Single Instruction:** All processing units execute the same instruction at any given clock cycle
- **Multiple Data:** Each processing unit can operate on a different data element
- Best suited for specialized problems characterized by a high degree of regularity, such as graphics/image processing.
- Synchronous (lockstep) and deterministic execution
- Two varieties: Processor Arrays and Vector Pipelines
- Examples:
    - Processor Arrays: Thinking Machines CM-2, MasPar MP-1 & MP-2, ILLIAC IV
    - Vector Pipelines: IBM 9000, Cray X-MP, Y-MP & C90, Fujitsu VP, NEC SX-2, Hitachi S820, ETA10
- Most modern computers, particularly those with graphics processor units (GPUs) employ SIMD instructions and execution units.

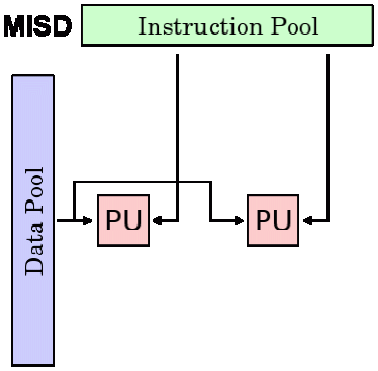


MULTIPLE INSTRUCTION, SINGLE DATA (MISD)

- A typ+
- e of parallel computer

- **Multiple Instruction:** Each processing unit operates on the data independently via separate instruction streams.
- **Single Data:** A single data stream is fed into multiple processing units.
- Few (if any) actual examples of this class of parallel computer have ever existed.
- Some conceivable uses might be:
    - multiple frequency filters operating on a single signal stream
    - multiple cryptography algorithms attempting to crack a single coded message.





MULTIPLE INSTRUCTION, MULTIPLE DATA (MIMD)

- A type of parallel computer
- **Multiple Instruction:** Every processor may be executing a different instruction stream
- **Multiple Data:** Every processor may be working with a different data stream
- Execution can be synchronous or asynchronous, deterministic or non-deterministic
- Currently, the most common type of parallel computer - most modern supercomputers fall into this category.
- Examples: most current supercomputers, networked parallel computer clusters and "grids", multi-processor SMP computers, multi-core PCs.
- **Note:** Many MIMD architectures also include SIMD execution sub-components

GENERAL PARALLEL COMPUTING TERMINOLOGY

- Like everything else, parallel computing has its own jargon. Some of the more commonly used terms associated with parallel computing are listed below. Most of these will be discussed in more detail later.

CPU

Contemporary CPUs consist of one or more cores - a distinct execution unit with its own instruction stream. Cores with a CPU may be organized into one or more sockets - each socket with its own distinct memory. When a CPU consists of two or more sockets, usually hardware infrastructure supports memory sharing across sockets.

Node

A standalone "computer in a box." Usually comprised of multiple CPUs/processors/cores, memory, network interfaces, etc. Nodes are networked together to comprise a supercomputer.

Task

A logically discrete section of computational work. A task is typically a program or program-like set of instructions that is executed by a processor. A parallel program consists of multiple tasks running on multiple processors.

Pipelining
Breaking a task into steps performed by different processor units, with inputs streaming through, much like an assembly line; a type of parallel computing.

Shared Memory
Describes a computer architecture where all processors have direct access to common physical memory. In a programming sense, it describes a model where parallel tasks all have the same "picture" of memory and can directly address and access the same logical memory locations regardless of where the physical memory actually exists.

Symmetric Multi-Processor (SMP)
Shared memory hardware architecture where multiple processors share a single address space and have equal access to all resources - memory, disk, etc.

Distributed Memory
In hardware, refers to network based memory access for physical memory that is not common. As a programming model, tasks can only logically "see" local machine memory and must use communications to access memory on other machines where other tasks are executing.

Communications
Parallel tasks typically need to exchange data. There are several ways this can be accomplished, such as through a shared memory bus or over a network.

Synchronization
The coordination of parallel tasks in real time, very often associated with communications.

Synchronization usually involves waiting by at least one task, and can therefore cause a parallel application's wall clock execution time to increase.

Computational Granularity
In parallel computing, granularity is a quantitative or qualitative measure of the ratio of computation to communication.

- ***Coarse:*** relatively large amounts of computational work are done between communication events
- ***Fine:*** relatively small amounts of computational work are done between communication events

Parallel Overhead
Required execution time that is unique to parallel tasks, as opposed to that for doing useful work. Parallel overhead can include factors such as:

- Task start-up time
- Synchronizations
- Data communications
- Software overhead imposed by parallel languages, libraries, operating system, etc.
- Task termination time