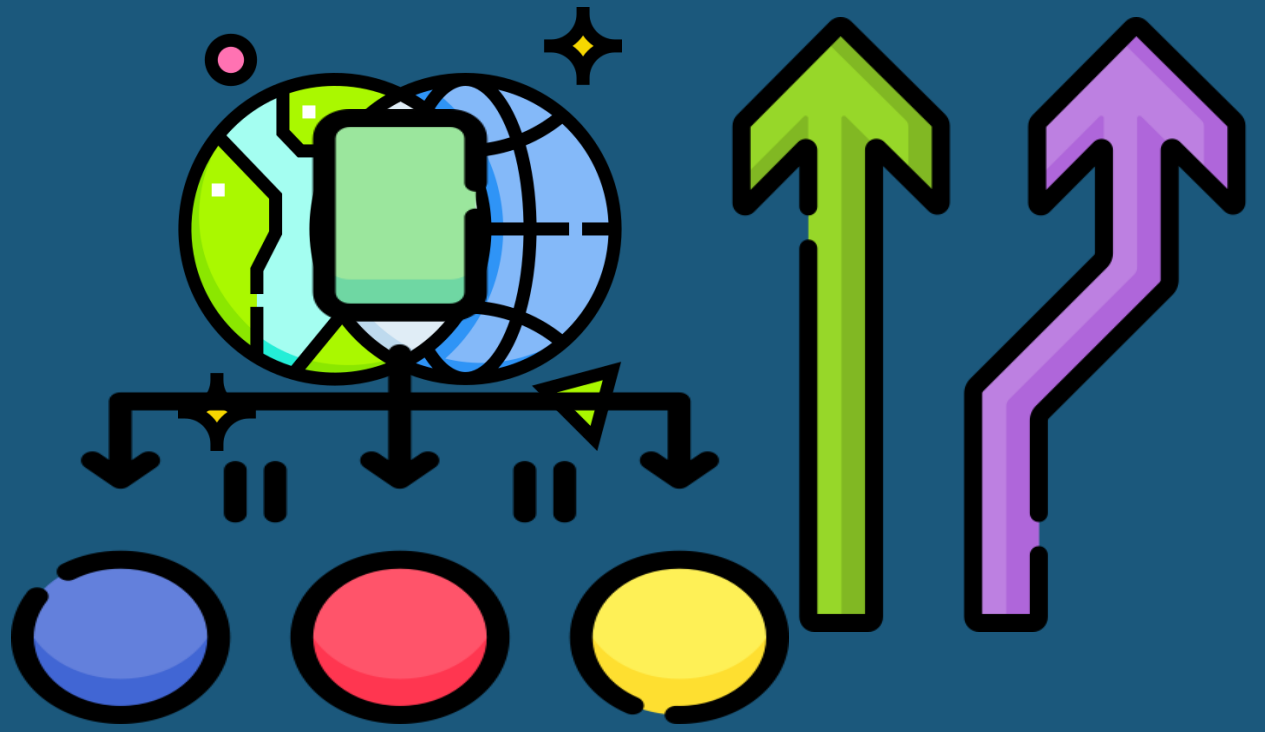


Parallel and Distributed Computing

Mr. Hassan Sardar

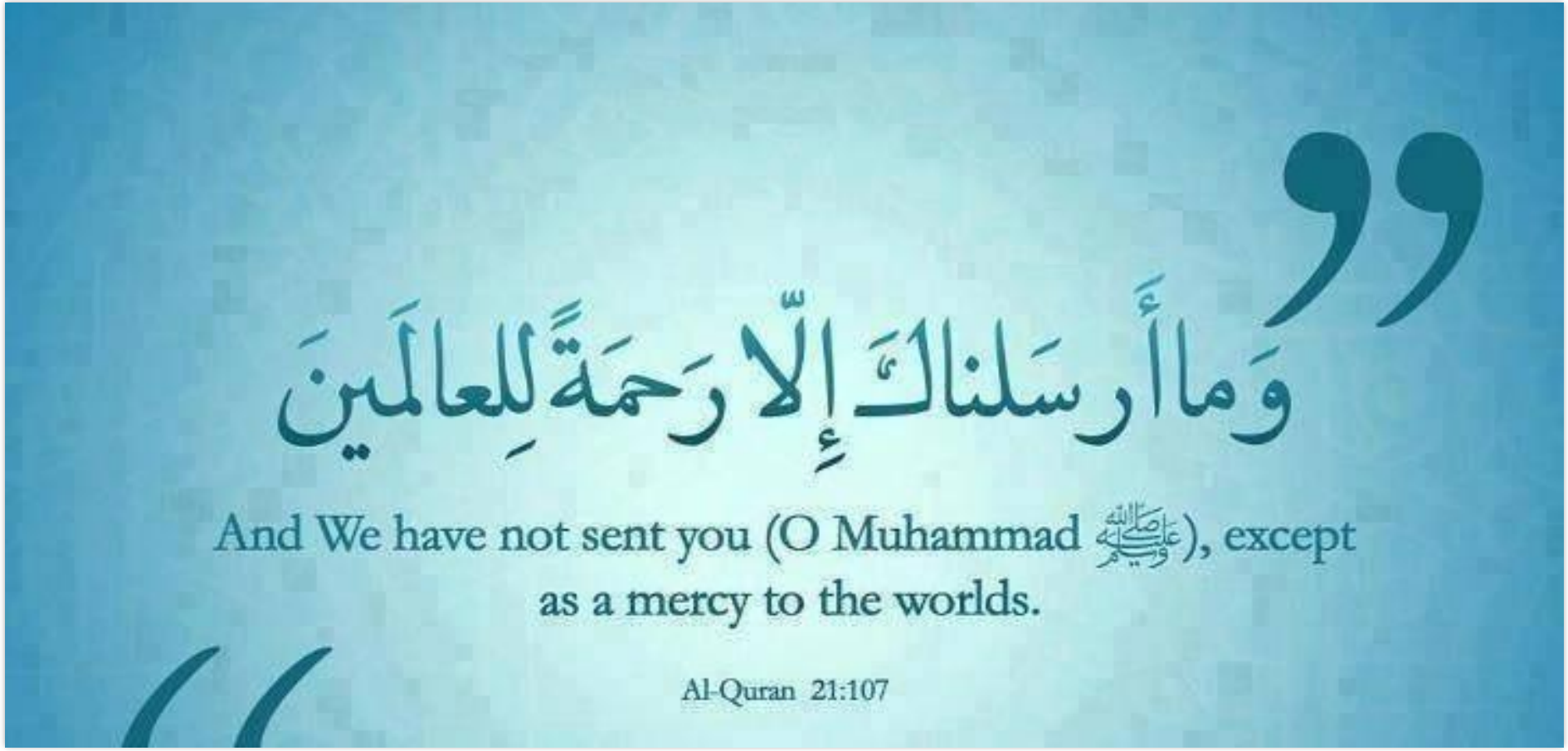
Lecturer, Department of Computer Science

COMSATS University Islamabad, Wah Campus

Email: hassan@ciitwah.edu.pk

Lesson

WAH Campus



Today' Lecture

WAH Campus

- Why Parallel Computing
  - Why should you learn about parallel computing
  - The Fundamentals laws of parallel computing
  - How does parallel computing work?
  - Categorizing parallel approaches
  - Parallel speedup vs comparative speedups

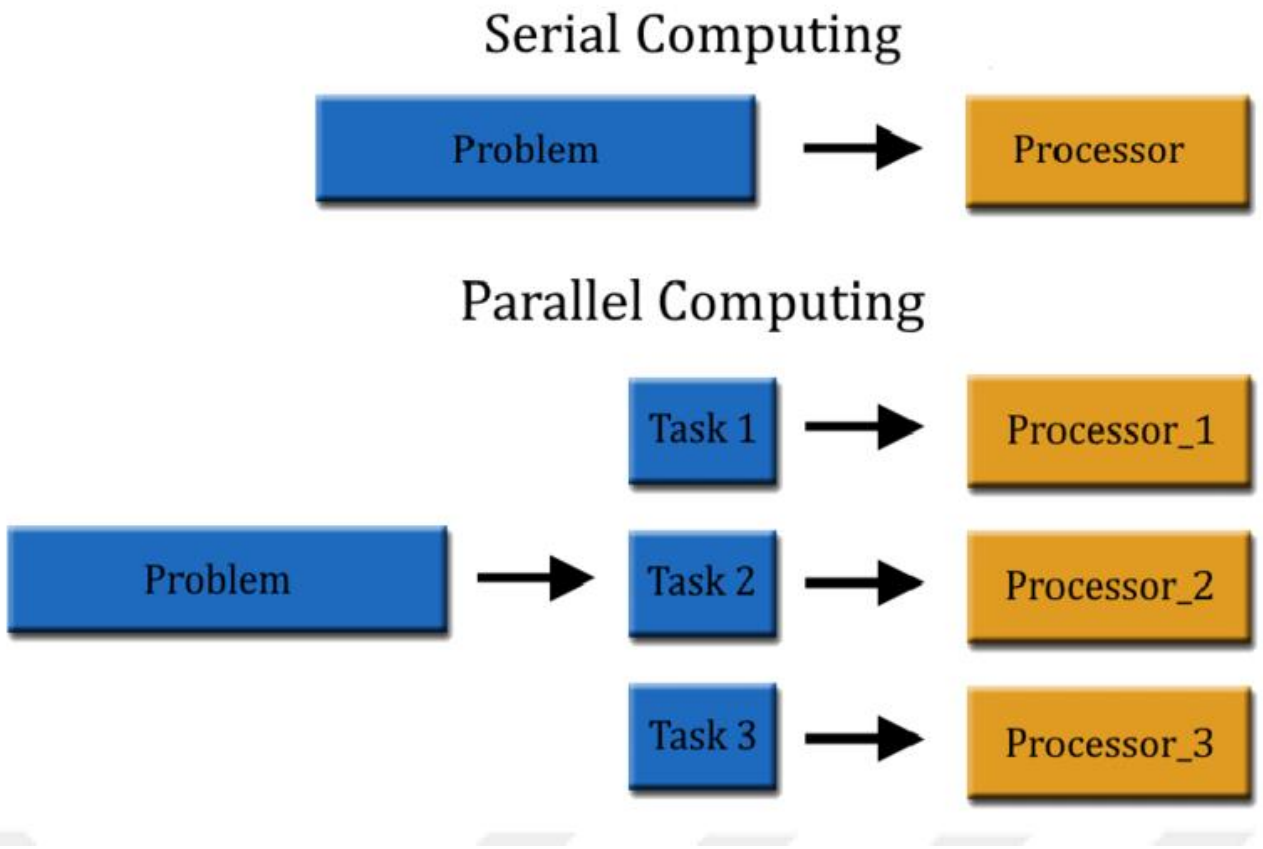
Why Parallel Computing?(1/2)

WAH Campus

- **Emerging Challenges**:
  - Extensive use of computing resources in fields like AI, machine learning, and scientific research.

  Application Include:

  - Voice recognition and image recognition (e.g., driverless cars).
  - Virus modeling, vaccine development, and climate modeling.
- **Parallel Computing Overview**
  - Execution of many operations at the same time.
  - **Why it's important**: Unlocks unused compute power, handles larger problems and datasets, improves simulation speed (up to 1000x faster).



Why Parallel Computing?(2/2)

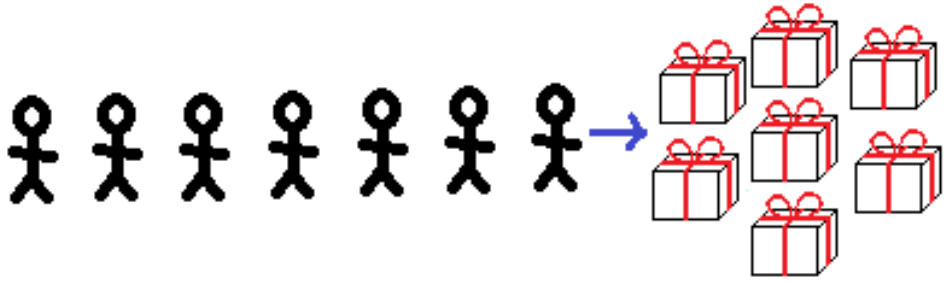
WAH Campus

How Parallel Computing Works?

- **Exposing Parallelism**: Identifying tasks that can be performed simultaneously.
- **Concurrency**: Operations that can run in any order as resources become available.

Applications & Benefits:

- Faster processing of large-scale problems.
- Reducing power consumption for mobile devices.
- Enhancing system performance beyond speed, including problem size and energy efficiency.
- Real-World Example: Coffee shop having single and multiple coffee maker



Concurrency: 7 kids queueing for presents from 1 heap

Parallelism: 7 kids getting 7 labeled presents, no queue

Why Learn Parallel Computing?(1/3)

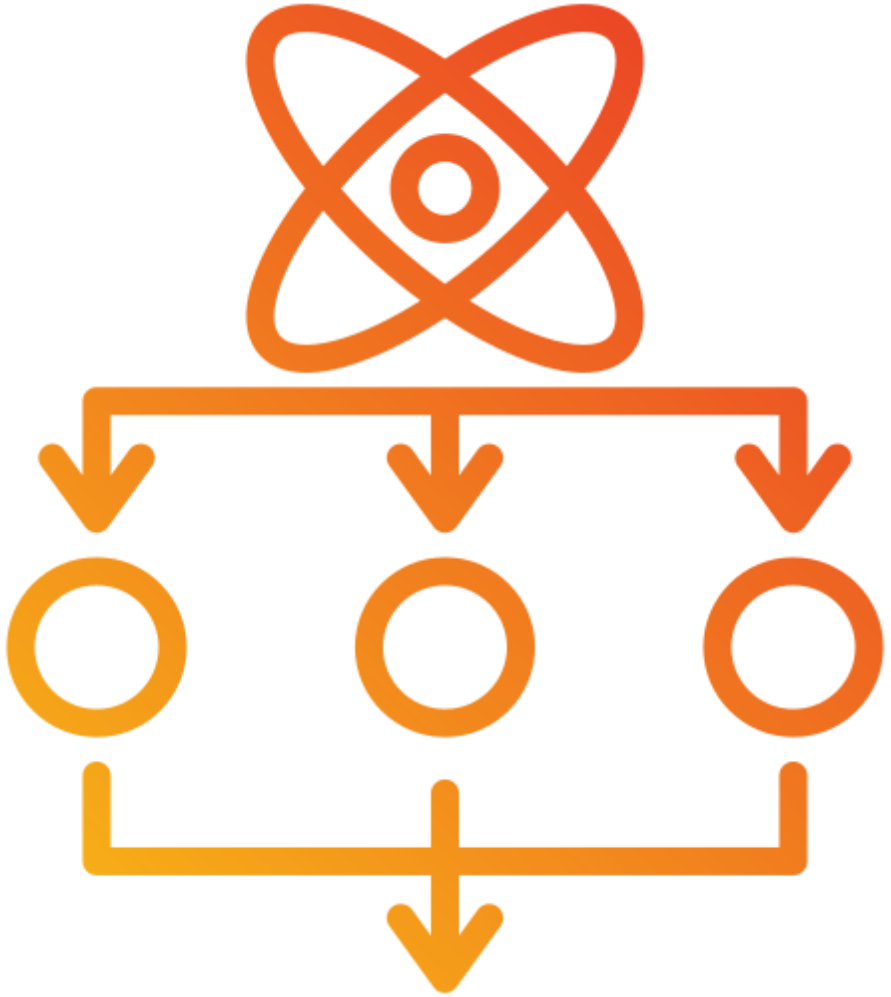
WAH Campus

Slowing Progress:

- Serial performance improvements have slowed due to limits in miniaturization(making things smaller), clock frequency, power, and heat.
- **Trend (2005 onwards)**: Multiple cores replace increasing clock frequency as the key to better CPU performance.

Core Count Rise:

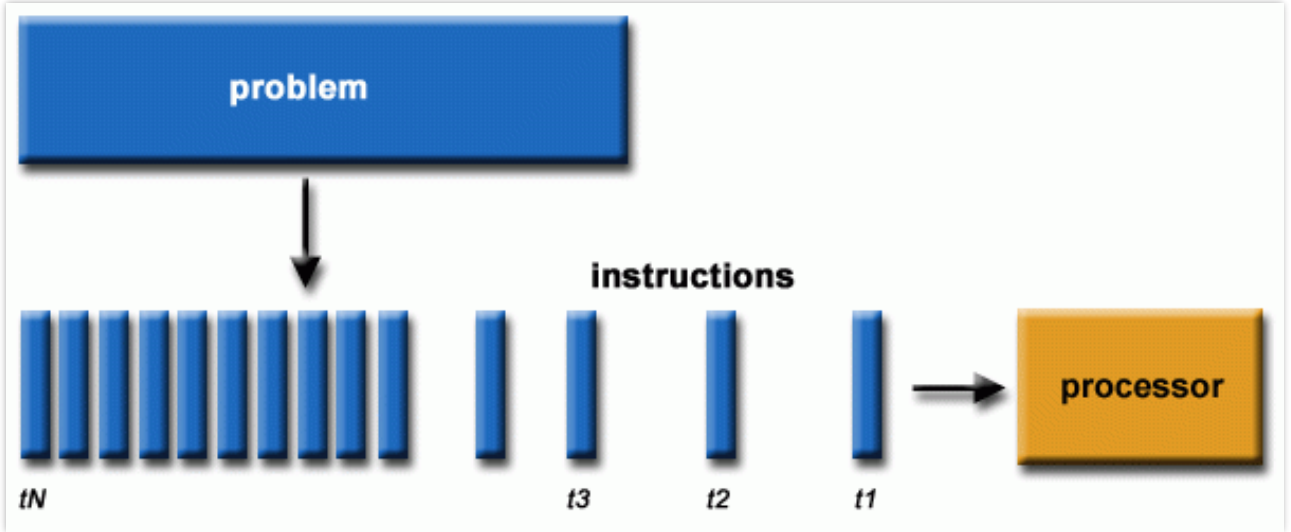
- **Performance Growth**: The number of CPU cores increased, leading to better performance through parallelism.
- Modern Hardware: Today's laptops, desktops, and smartphones are as powerful as supercomputers from 20 years ago.



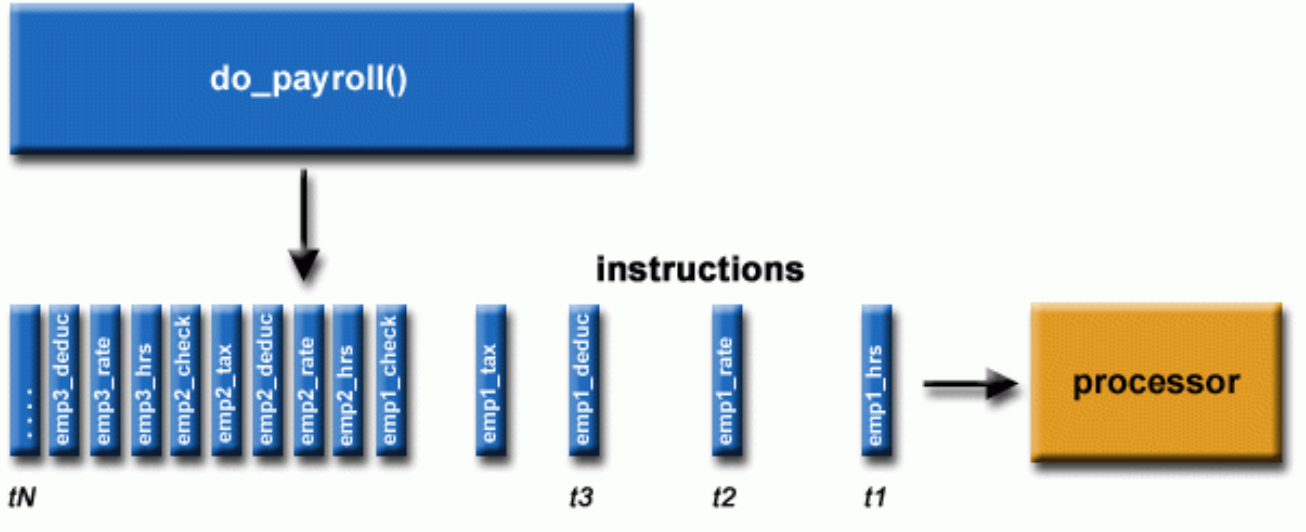
Why Learn Parallel Computing?(2/3)

WAH Campus

Serial computing generic example



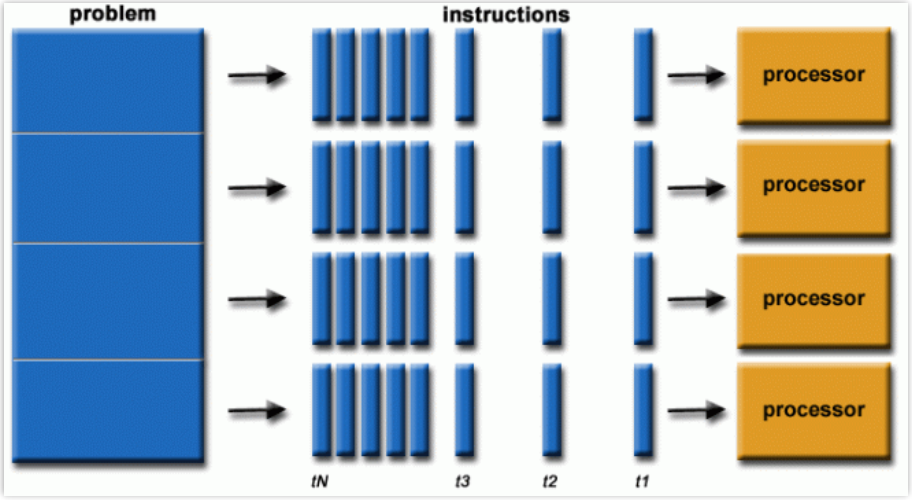
Serial computing example of processing payroll



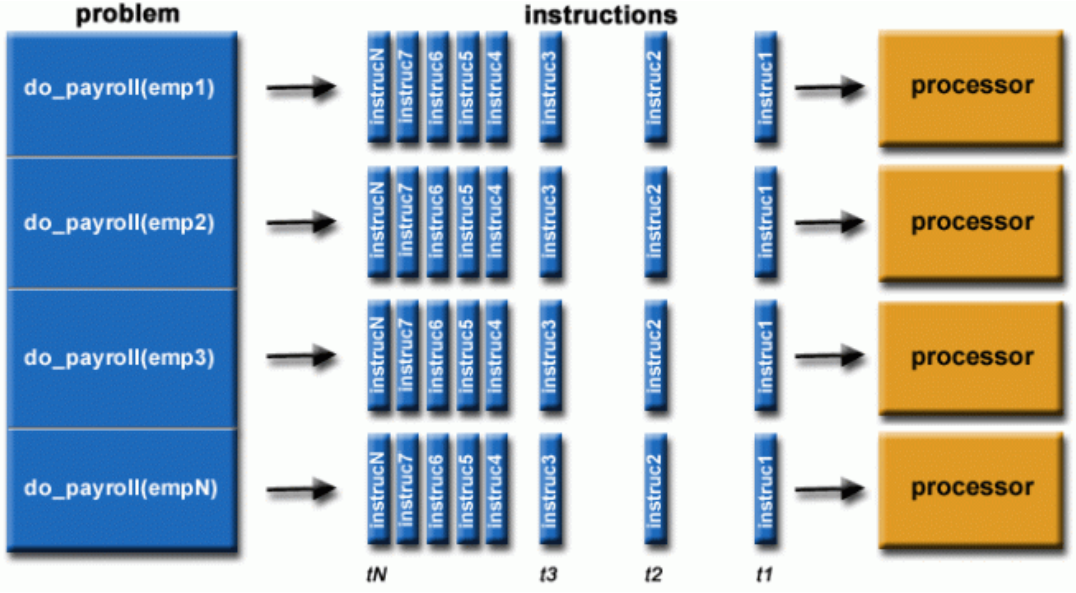
Why Learn Parallel Computing?(3/3)

WAH Campus

Parallel computing generic example



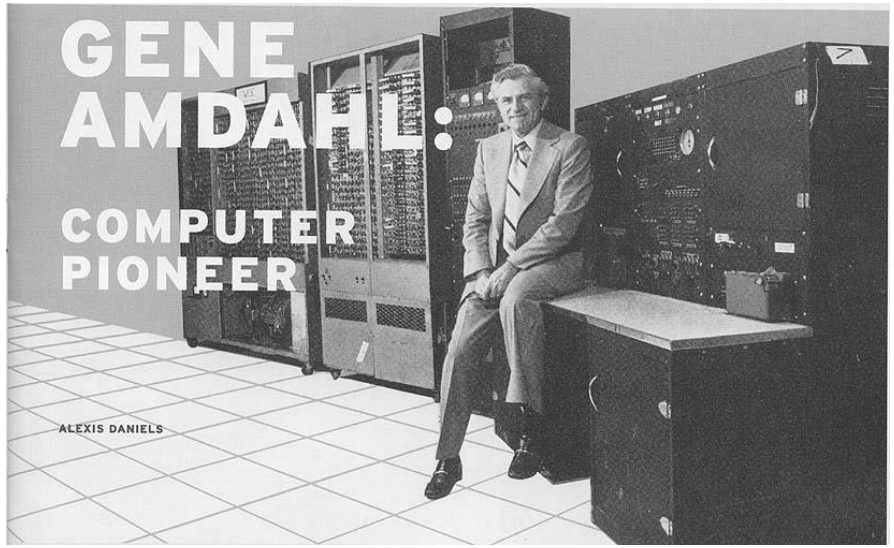
parallel computing example of processing payroll



Fundamental Laws of parallel computing(1/4)

Amdahl's Law

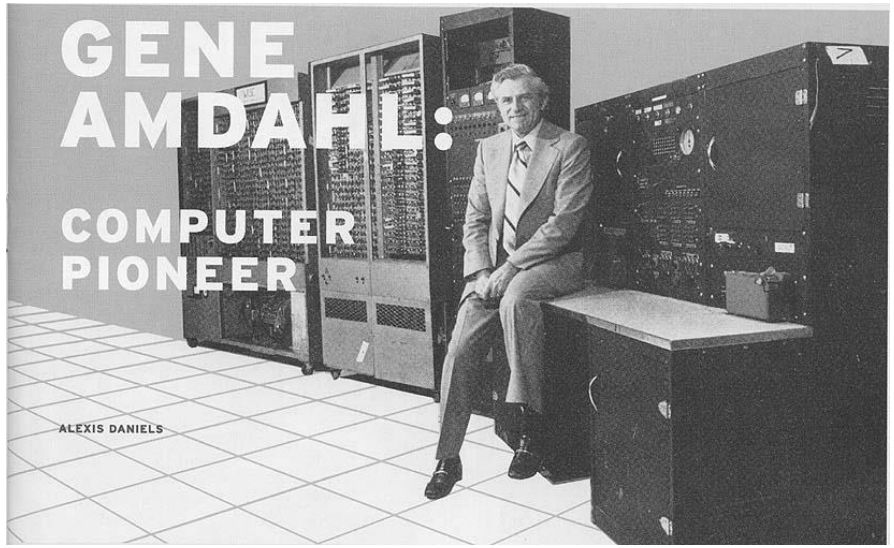
- Amdahl's Law, proposed by Gene Amdahl in 1967, describes the potential speedup of a fixed-size problem as the number of processors increases. It helps us calculate the limit of parallel computing performance.
- The equation is: Speedup (N) = 1 / (S + P/N)
- Key Components of Amdahl's Law:
  - 1. P (Parallel Fraction): The part of the code that can run in parallel.
  - 2. S (Serial Fraction): The part of the code that runs sequentially.
  - 3. N: The number of processors.
  - 4. P + S = 1, meaning the total of parallel and serial parts equals 1.



Fundamental Laws of parallel computing(2/4)

Amdahl's Law

- **Limitation of Parallelism:**
  - Amdahl's Law highlights that no matter how much we increase the speed of the parallel part, the overall speedup is limited by the serial fraction.
  - This limitation is visualized through strong scaling.
- **Strong Scaling:**
  - Strong scaling refers to the time to solution in relation to the number of processors for a fixed-size problem. The performance gain is limited by the serial portion of the program.



Fundamental Laws of parallel computing(3/4)

Gustafson-Barsis's Law :

- In 1988, Gustafson and Barsis proposed that increasing problem size as more processors are added can provide a better way to calculate speedup. This allows us to exploit additional parallelism in larger problems.
- The formula is as follows:
- SpeedUp (N) = N - S * (N - 1)
- Where:
- N = number of processors
- S = serial fraction of the program

Fundamental Laws of parallel computing(4/4)

Benefits of Gustafson-Barsis's Law:

- By growing the problem size with the number of processors, larger problems can be solved in the same amount of time. This opens new opportunities for parallelism and improved scalability.

Definition of Weak Scaling:

- Weak scaling refers to the time to solution with respect to the number of processors, keeping the size of the problem per processor fixed.

Strong Scaling vs Weak Scaling:

- Strong scaling: Fixed total problem size divided across processors to speed up calculation.
- Weak scaling: Fixed problem size per processor, increasing overall problem size with more processors.

How does parallel computing work?(1/3)

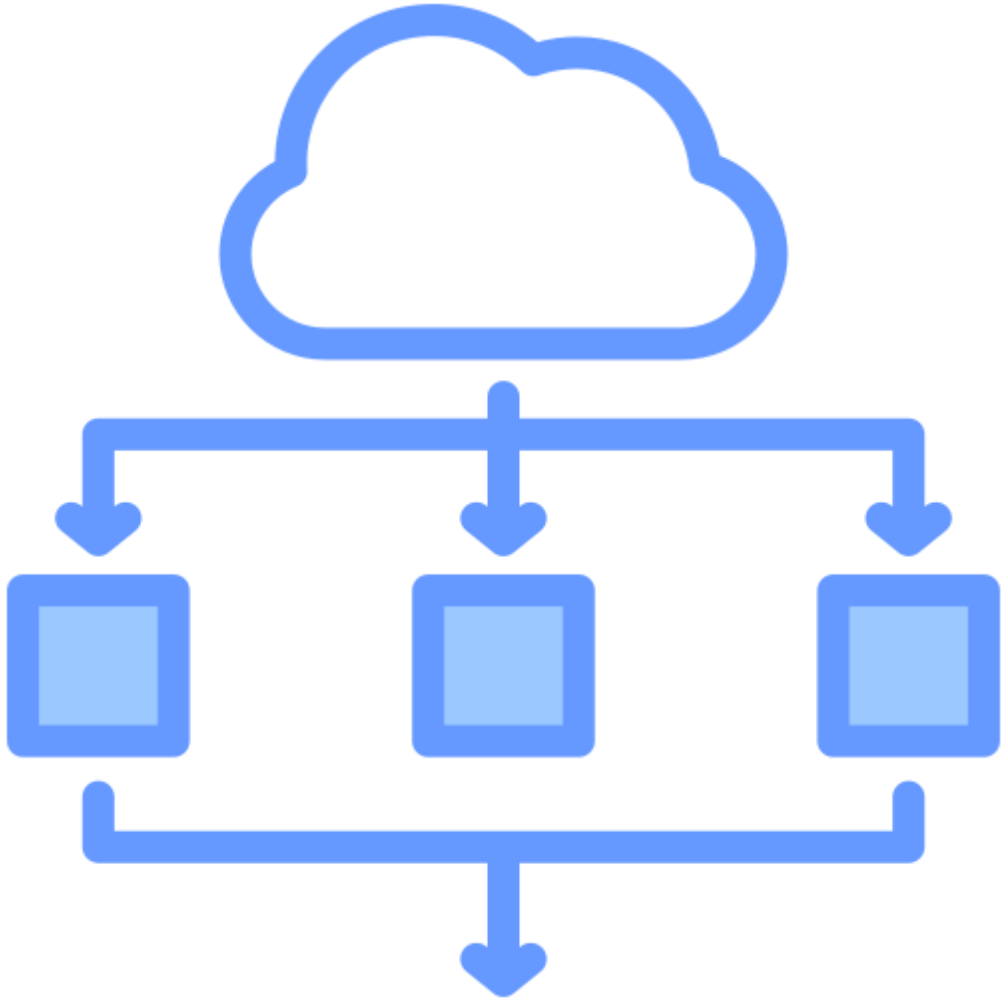
WAH Campus

What is Parallel Computing?

- Parallel computing requires combining hardware, software, and parallelism to develop efficient applications. It involves more than just threading or message passing and offers multiple ways to increase efficiency by splitting work across processors.

Layers of Parallel Computing:

- In parallel computing, your application goes through multiple layers:
- 1. Source Code: Where parallelism is exposed
- 2. Compiler: Translates code for hardware
- 3. Operating System: Manages execution on hardware
- Understanding these layers is key to effectively utilizing hardware resources.

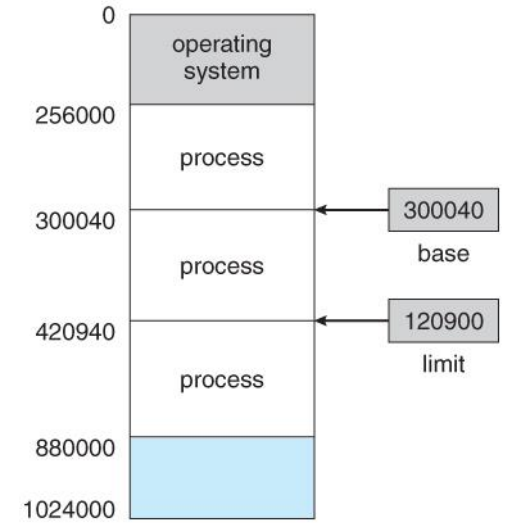


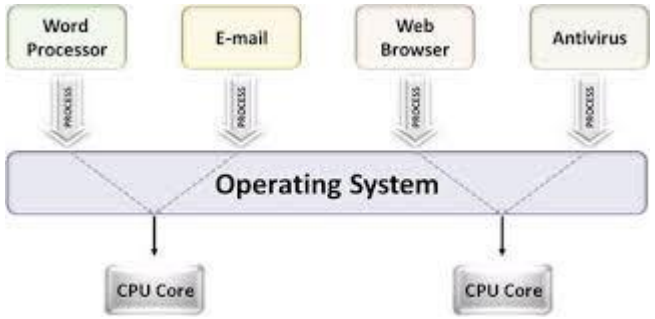
How does parallel computing work?(2/3)

WAH Campus

Parallelization Strategies:

- Common approaches to parallelization include:
- 1. Process-based: Uses separate memory spaces
- 2. Thread-based: Utilizes multi-core CPUs
- 3. Vectorization: Operates on multiple data units simultaneously
- 4. Stream Processing: Commonly used by GPUs





How does parallel computing work?(3/3)

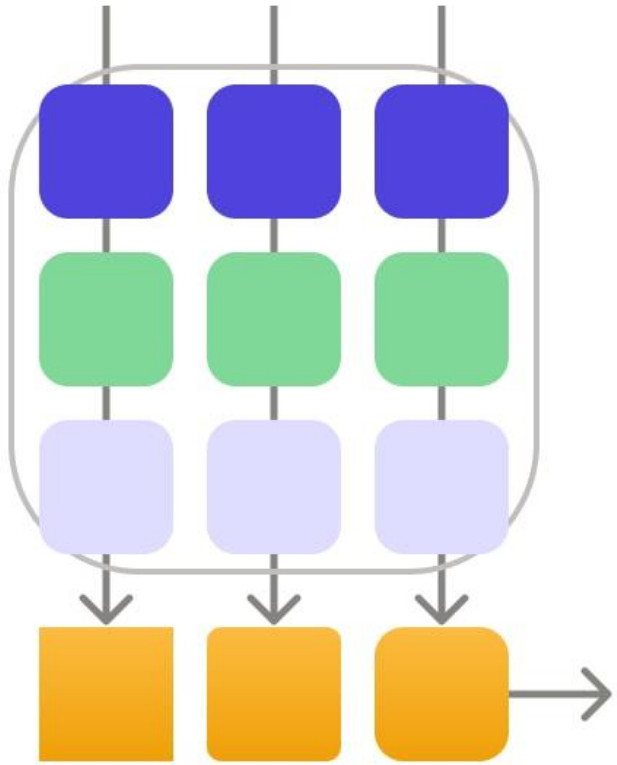
WAH Campus

Data Vectorization:

- Vectorization improves performance by applying a single operation to multiple data points simultaneously, instead of one at a time (scalar computing). This parallelizes computations, particularly with large datasets, using modern CPU/GPU architecture.

Benefits of Vectorization:

- **Faster Computation**: Handles large datasets more efficiently, providing significant speedups.
- **Optimized for Array Operations**: Ideal for mathematical/statistical tasks (e.g., NumPy in Python).



Categorizing Parallel approaches

WAH Campus

Flynn's Taxonomy (1966)::

Categorizes parallel computer architectures based on instructions and data processing.

Categories of Flynn's Taxonomy:

SISD (Single Instruction, Single Data):Serial architecture; one instruction on one data item at a time.

SIMD (Single Instruction, Multiple Data): One instruction executed across multiple data points (e.g., vectorization).

- Common in GPUs and array processing.

MISD (Multiple Instruction, Single Data): Rare architecture, mostly used in fault-tolerant systems like spacecraft controllers.

- Multiple instructions on the same data for redundancy.

MIMD (Multiple Instruction, Multiple Data): Parallelization of both instructions and data, typical in multi-core processors and large parallel systems.

Categorizing Parallel approaches

SIMT (Single Instruction, Multi-Thread):A variation of SIMD used in GPU architectures to describe multiple threads performing the same operation on different data.

Application of Flynn's Taxonomy :Useful for recognizing patterns and challenges in parallelization, such as the handling of conditionals in SIMD.

Parallel speedup vs comparative speedups

- **Parallel Speedup**: Measures improvement from running a serial task in parallel on the same architecture (e.g., CPU to GPU).
- **Comparative Speedup**: Compares performance across different parallel architectures (e.g., multi-core CPU vs. GPU).

Contextualizing Speedup:

- Use hardware release years in parentheses to clarify performance comparisons:
- **Best 2016**: Highest-end hardware from a specific year.
- **Common 2016**: Mainstream parts released in a specific year.
- **Mac 2016**: Hardware in a 2016 Mac system.

Conclusion: Always provide context for speedup comparisons to avoid misleading results, especially due to the rapid advancement of CPU and GPU models.