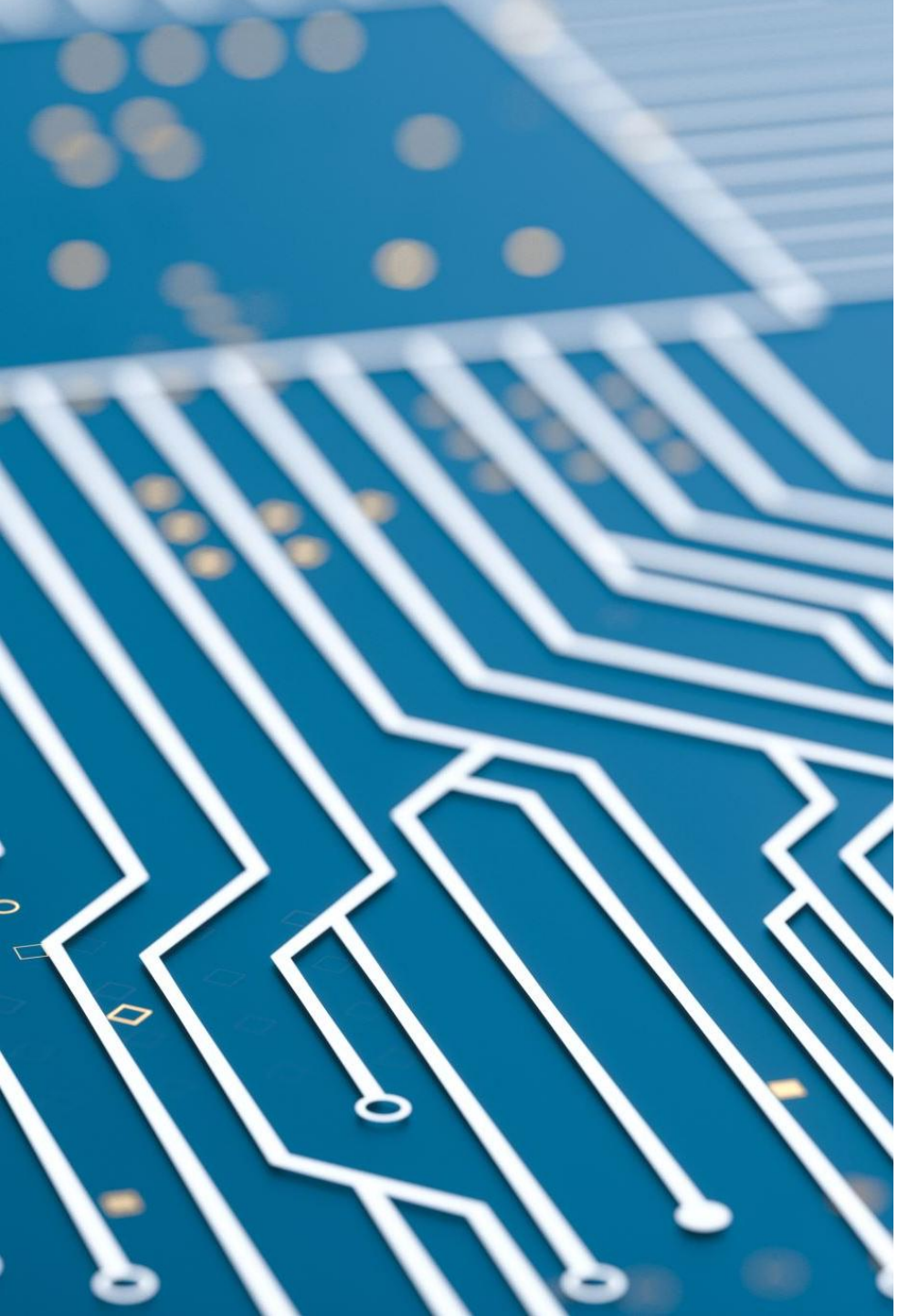
Parallel & Distributed Computing Lecture # 4

BY,

DR. ALI AKBAR SIDDIQUE



Introduction to Performance Metrics

What are Performance Metrics?

Performance metrics are **quantitative measurements** used to evaluate the effectiveness of a parallel or distributed computing system. These metrics help determine how well a system utilizes resources such as CPUs, GPUs, memory, and bandwidth to solve computational problems efficiently.

Definition

- In parallel and distributed computing, performance metrics are used to measure and analyze how efficiently algorithms and systems perform when using multiple processing elements.
- These metrics are crucial for identifying bottlenecks, comparing systems, and optimizing applications.

Why Are They Important?

- Help determine whether parallelization improves performance.
- Allow comparison between different algorithms, architectures, or implementations.
- Guide developers to optimize resource usage and minimize computation time.
- Essential for scaling applications across larger systems.

Common Scenarios Where Performance Metrics Are Used

- Evaluating parallel algorithms in HPC (High Performance Computing) environments.
- Monitoring distributed workloads in cloud computing.
- Optimizing real-time systems and simulations.
- Benchmarking tools and profilers to identify critical code paths.

Efficiency in Parallel Computing

What is Efficiency?

- Efficiency measures how well a parallel system uses its resources. It is the ratio of speedup to the number of processors used. A high-efficiency value indicates that the system is using its processors effectively.
- Efficiency (E) is a performance metric that evaluates the degree of resource utilization in a parallel computing system.
- It shows how much useful work each processor is doing relative to the work done in the sequential version of the program.

$$\text{Efficiency (E)} = \frac{\text{Speedup (S)}}{\text{Number of Processors (P)}}$$

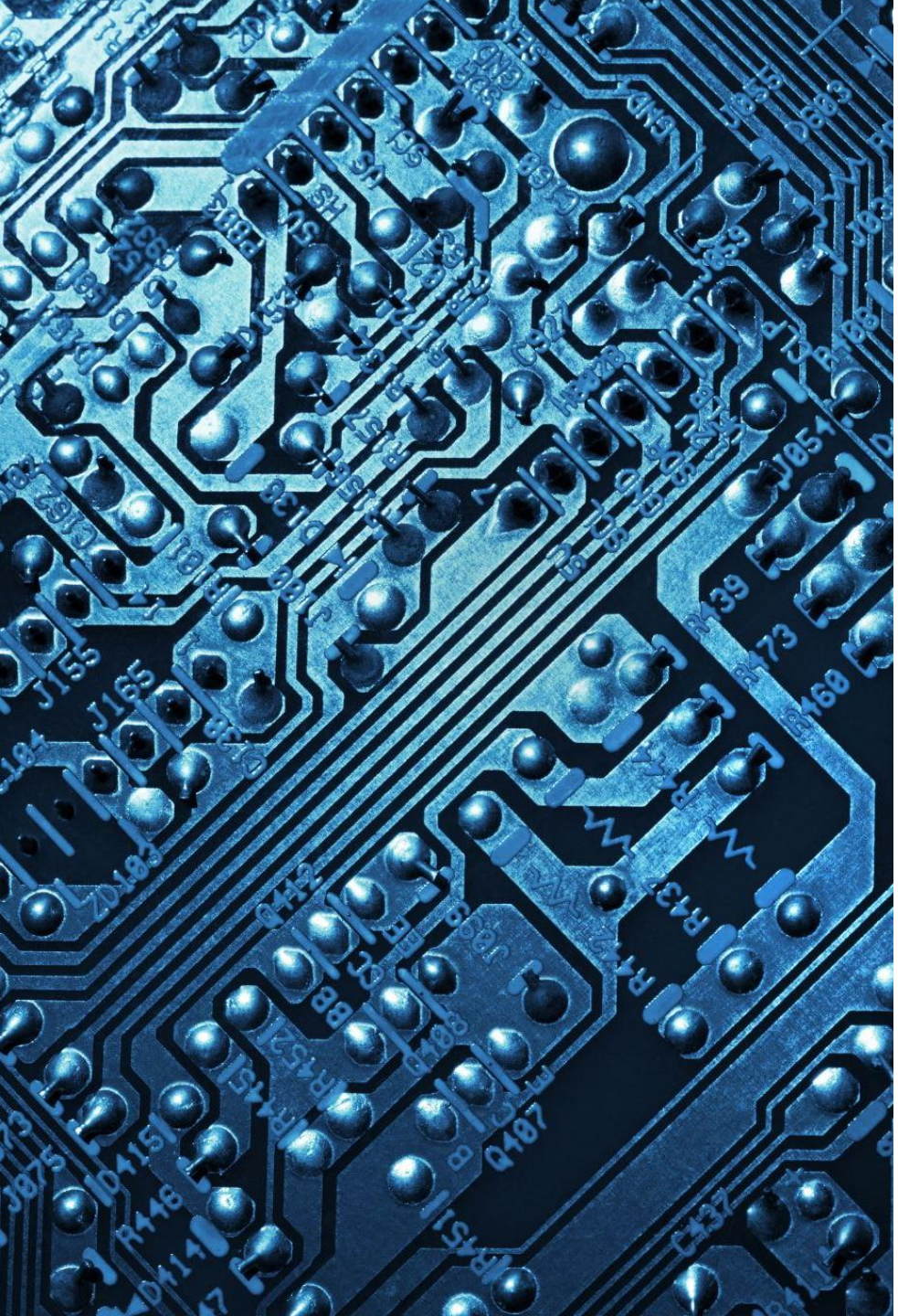
- Where:

  S = Speedup = $\frac{T_1}{T_P}$

  P = Number of processors

  T_1 = Time taken by sequential execution

  T_P = Time taken by parallel execution on P processors



Interpretation

- E = 1 (or 100%): Perfect efficiency (each processor contributes equally).
- E < 1: Some degree of waste or overhead, such as communication or synchronization delays.
- Efficiency usually decreases as the number of processors increases due to these overheads.

Why Efficiency Matters:

Helps identify if adding more processors is beneficial or just adding overhead.

Assists in resource allocation and cost justification in parallel/distributed systems.

Important when working on scalable applications—you want to maintain efficiency even as you scale up.

Real-World Analogy

- Think of a group project: If four students complete a project in 1/4th the time it takes one student alone, they’re 100% efficient. If they only save a little time, they might be stepping on each other's toes (inefficient!).

Numerical

Q: A program takes 100 seconds to execute on a single processor and 50 seconds on 2 processors. Calculate the efficiency.

Solution:

- Speedup $S = \frac{100}{50} = 2$
- Efficiency $E = \frac{2}{2} = 1.0 \rightarrow 100\%$

Numerical

Q: A parallel program takes 120 seconds on 1 processor and 40 seconds on 4 processors. Find the efficiency.

Solution:

- Speedup $S = \frac{120}{40} = 3$
- Efficiency $E = \frac{3}{4} = 0.75 \rightarrow 75\%$

Numerical

Q: A job takes 90 seconds on a single core and 30 seconds on 5 cores. Compute efficiency.

Solution:

- Speedup $S = \frac{90}{30} = 3$
- Efficiency $E = \frac{3}{5} = 0.6 \rightarrow 60\%$
- **Interpretation**: There is some overhead or idle time among processors.

Numerical

Q: A 6-processor system is operating at 80% efficiency. What is the speedup?

Solution:

- Efficiency $E = \frac{S}{P} \Rightarrow S = E \times P = 0.8 \times 6 = 4.8$
- **Speedup = 4.8**

Numerical

Q: A program has 80% of its code parallelizable. What is the efficiency when using 4 processors? (Use Amdahl's Law: $S = \frac{1}{(1-f)+\frac{f}{P}}$)

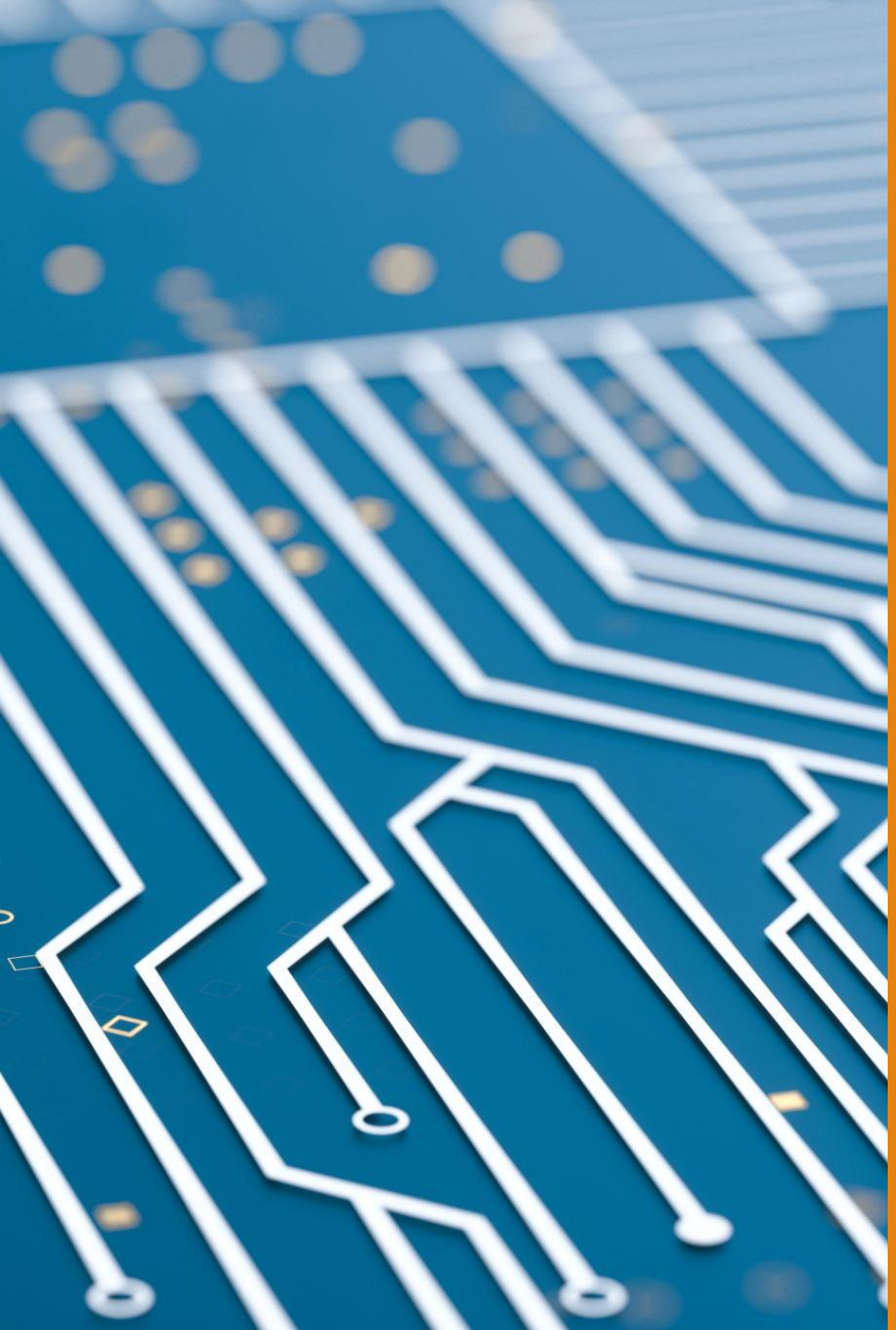
Where:

- $f = 0.8$ (parallel fraction)
- $P = 4$

Solution:

$$S = \frac{1}{(1 - 0.8) + \frac{0.8}{4}} = \frac{1}{0.2 + 0.2} = \frac{1}{0.4} = 2.5$$

- Efficiency $E = \frac{2.5}{4} = 0.625 \rightarrow 62.5\%$



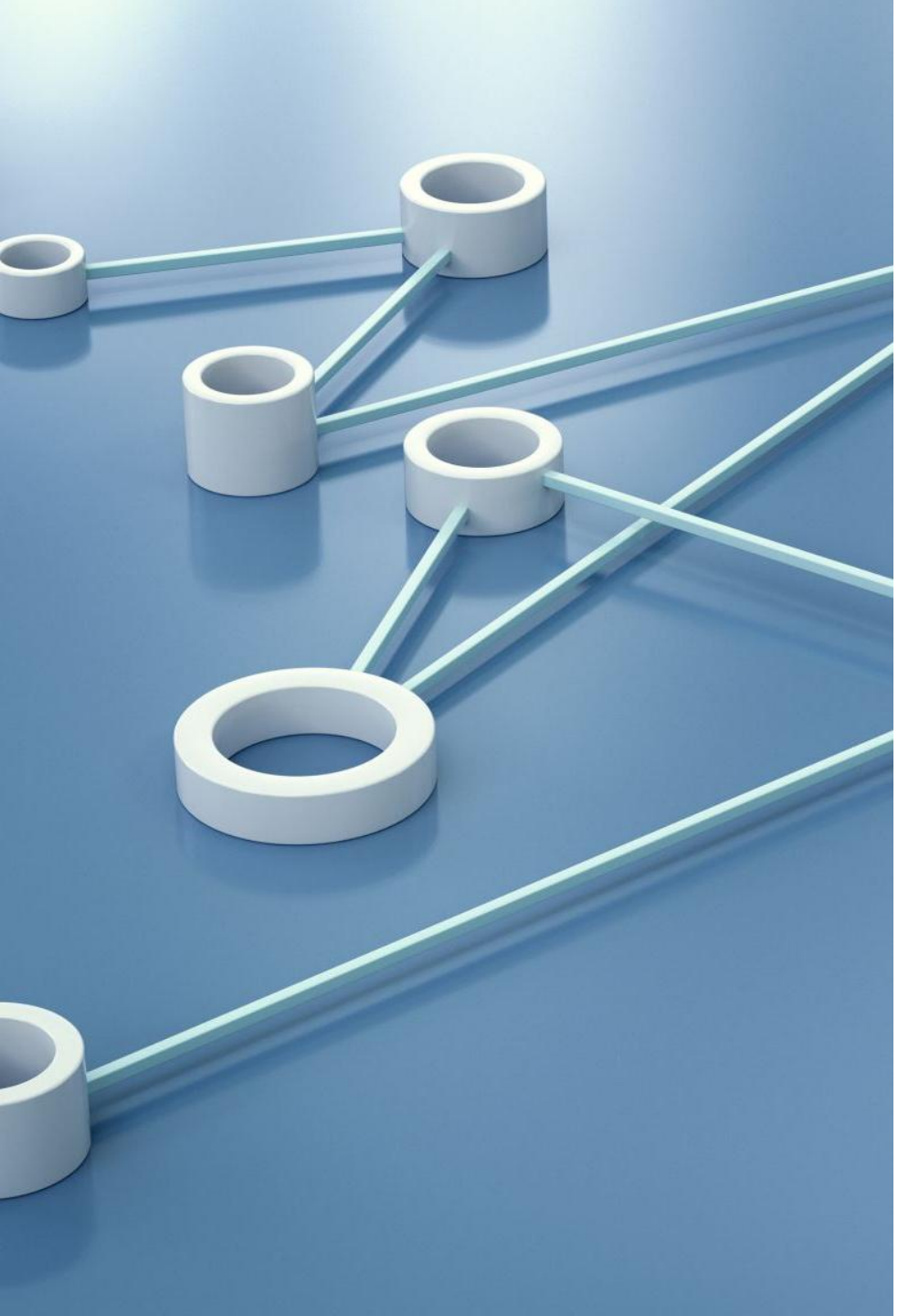
Scalability in Parallel Systems

Scalability refers to the ability of a parallel system to achieve improved performance (typically, speedup) as the number of processors increases. It measures how effectively a system can scale up in terms of performance with the addition of computing resources.

Types of Scalability:

1. **Strong Scalability**:
   - The problem size remains **fixed**, and performance improves as more processors are added.
   - Ideal for tasks with heavy computation and low communication overhead.
   - Example: Weather simulation using a fixed-size model on more cores.

2. **Weak Scalability**:
   - The problem size **increases proportionally** with the number of processors.
   - Measures how well the system can handle a growing workload.
   - Example: Processing more images in a distributed image processing task as more machines are added.



Why Scalability Matter?

- Ensures systems can handle larger problems or more users.
- Helps maximize hardware utilization and optimize cost-performance ratio.
- Critical for high-performance computing (HPC) applications and cloud-based infrastructures.

Real-World Insight:

A system with poor scalability may perform worse when more resources are added—known as **parallel slowdown**.

Amdahl's Law

Amdahl's Law provides a theoretical limit to the maximum speedup achievable when only part of a program can be parallelized. It highlights the impact of the sequential portion of a task on overall performance improvement.

$$\text{Speedup}(S) = \frac{1}{(1 - P) + \frac{P}{N}}$$

Where:

- **P** = Proportion of the program that can be parallelized
- **1 - P** = Proportion of the program that remains sequential
- **N** = Number of processors

Interpretation

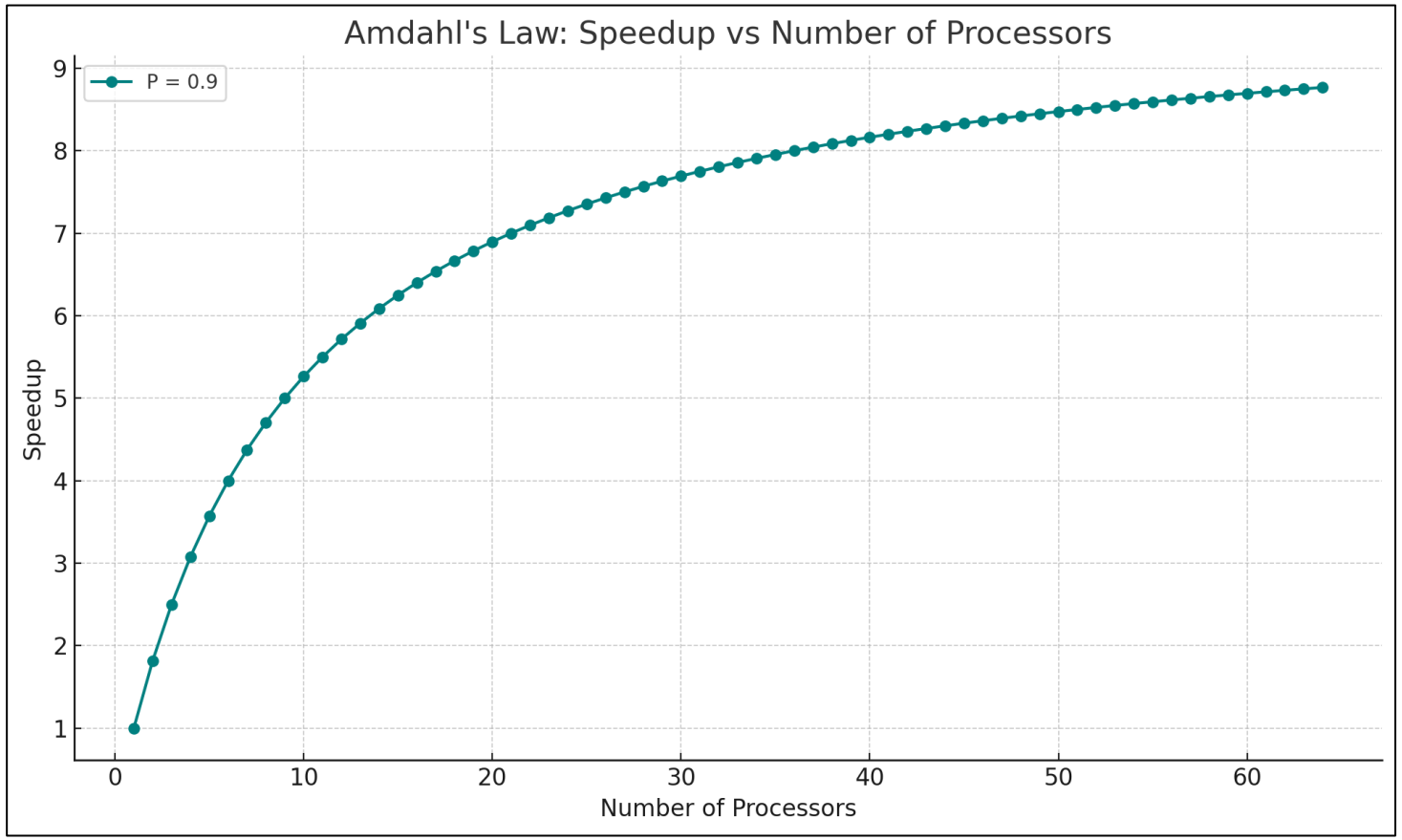
- No matter how many processors are added, the **sequential part** limits speedup.
- If **P = 0.9 (90%)** and **N = 10**, then:

$$S = \frac{1}{(1 - 0.9) + \frac{0.9}{10}} = \frac{1}{0.1 + 0.09} = \frac{1}{0.19} \approx 5.26$$

- Even with **infinite processors**, the max speedup is:

$$S = \frac{1}{1 - P}$$

Amdahl’s Law (Speed-up Vs Nos of Processors)



Here's a graphical representation of Amdahl's Law showing how speedup improves with the number of processors when 90% (P = 0.9) of the code is parallelizable. As shown, the speedup increases with more processors but plateaus due to the serial portion of the code, illustrating the law’s limitations. Let me know if you'd like graphs for other values of P or for scalability comparisons too.

Real-World Analogy

Imagine cooking dinner:

- Boiling rice (parallel task): Can be done on many burners.
- Serving (sequential task): One person has to do it.
- Even with 10 chefs, if only one can serve, your total time won’t improve significantly.