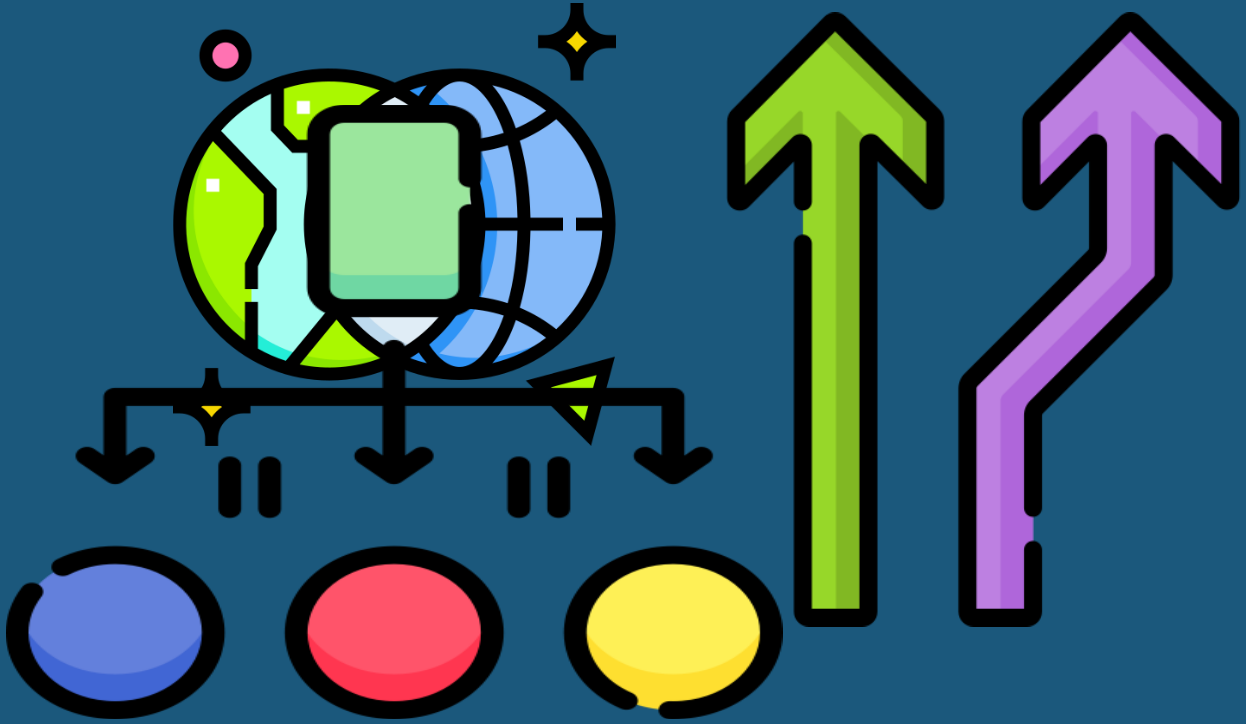


Parallel and Distributed Computing (CSC334)

Mr. Hassan Sardar

Lecturer, Department of Computer Science

COMSATS University Islamabad, Wah Campus

Email: hassan@ciitwah.edu.pk

WAH Campus

لَآ اِلٰهَ اِلَّآ اَنتَ سُبحٰنَکَ اِنِّی کُنتُ مِنَ الظّٰلِمِینَ

There is no god but You (O My Lord!). Glory be to You! Verily, I have been of the wrongdoers.

Today' Lecture

What is Parallelism?

- Parallelism in Computing
- Task Parallelism
- Characteristic ,Advantages and Disadvantages
- Data Parallelism
- Characteristic ,Advantages and Disadvantages
- Key Differences Between Task and Data Parallelism
- Hybrid Parallelism

Introduction to Parallelism

WAH Campus

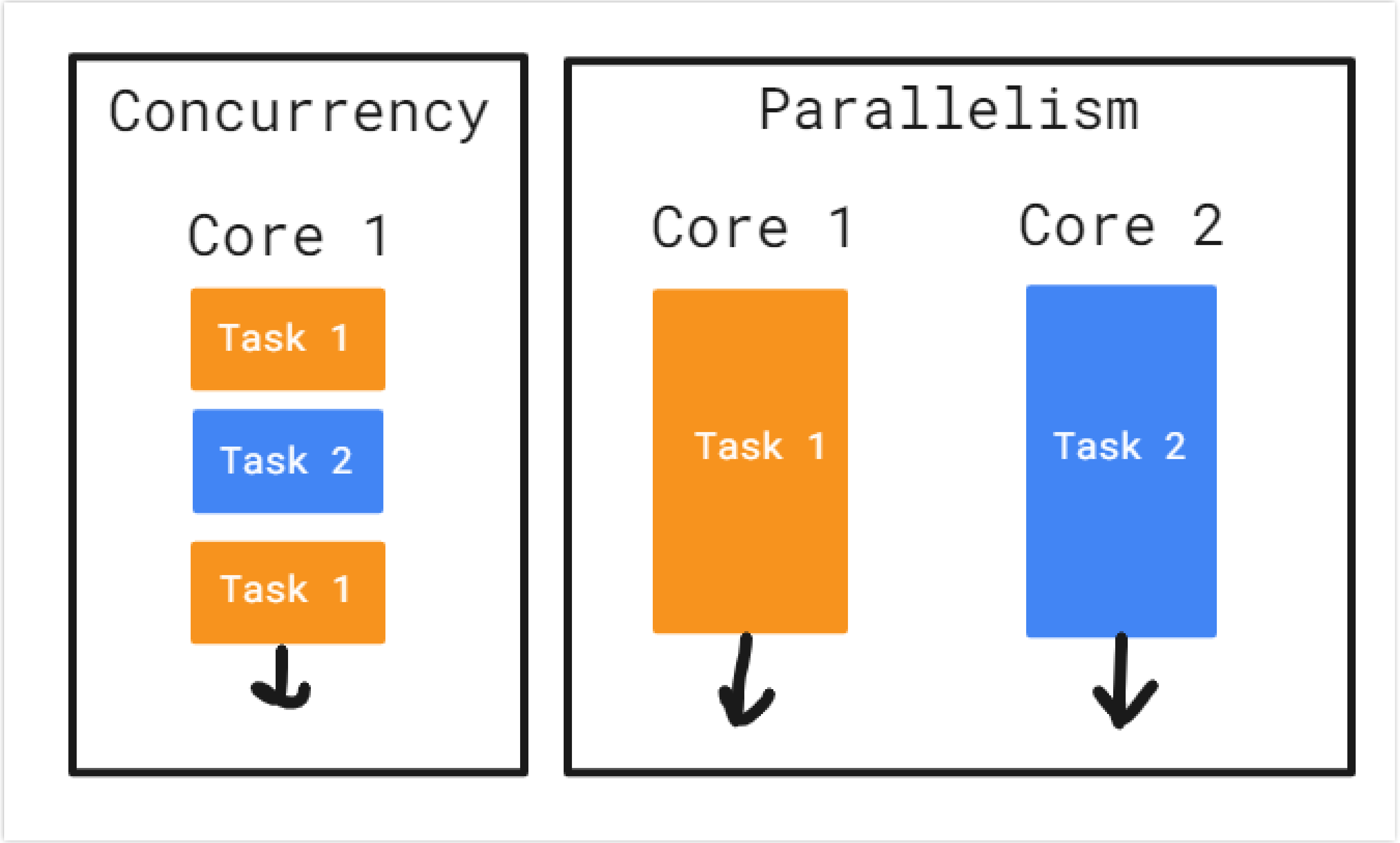
- Parallelism is the ability of a computer to perform multiple tasks or operations simultaneously. It aims to speed up computation and improve efficiency by distributing tasks across multiple processors or cores.
- With the increasing demand for high-performance computing in applications like scientific simulations, data analysis, and machine learning, parallelism has become a crucial concept in computer science.

Types of Parallelism
Parallelism is primarily categorized into two types:

1. **Task Parallelism**
2. **Data Parallelism**

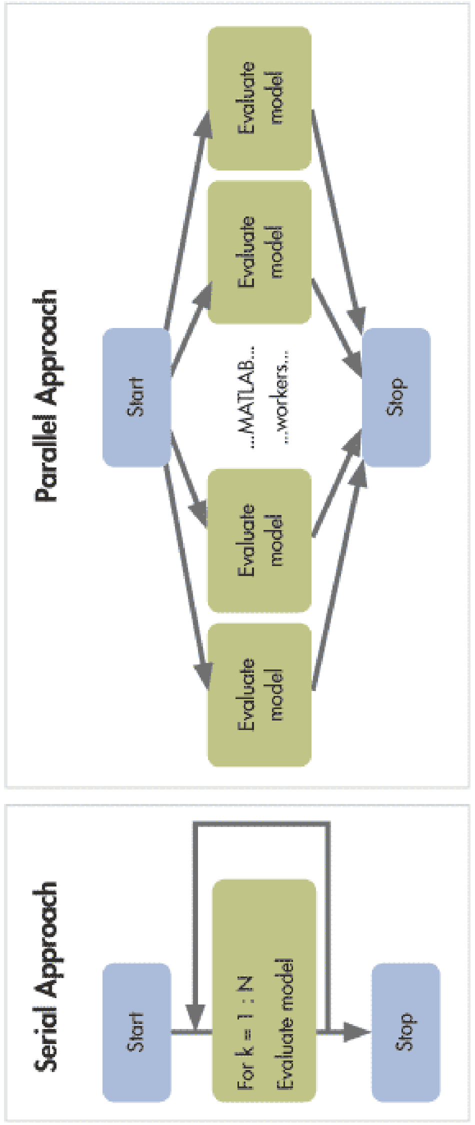
- Before diving into these types, let's understand the broader concept of parallelism.

Introduction to Parallelism



Parallelism in Computing

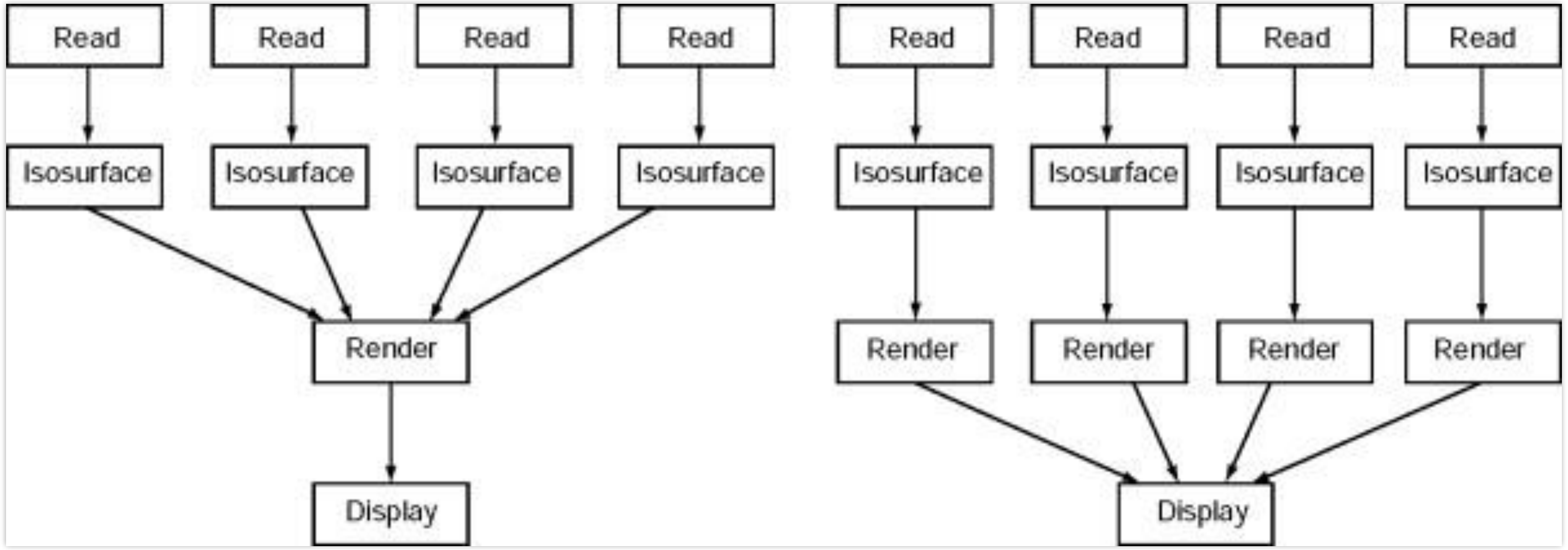
- Parallel computing refers to a computational process in which multiple processors execute or process an application or task simultaneously. Parallel computing helps to solve larger problems, reduce computation time, and improve the efficiency of systems.
- Parallel computing can be achieved at various levels:
  - **Bit-level Parallelism**: Working with larger data types to reduce the number of operations.
  - **Instruction-level Parallelism**: Executing multiple instructions in one clock cycle.
  - **Task and Data Parallelism**: Distributing tasks and data across multiple processors for faster execution.



Task Parallelism

- **Definition**
  Task parallelism, also known as *control parallelism*, is a form of parallelization in which different tasks or processes are executed concurrently. Each task runs independently on a separate processor or core.
- **Concept**
  In task parallelism, the problem is divided into distinct tasks that do not necessarily have to perform the same operation. Each processor may execute a different function on different data. The tasks may communicate with one another but are largely independent.
- For example, in a video processing application:
- Task 1: Reading video frames from a file.
- Task 2: Applying filters to each frame.
- Task 3: Encoding the frames into a new video format. All these tasks can be run in parallel on different processors.

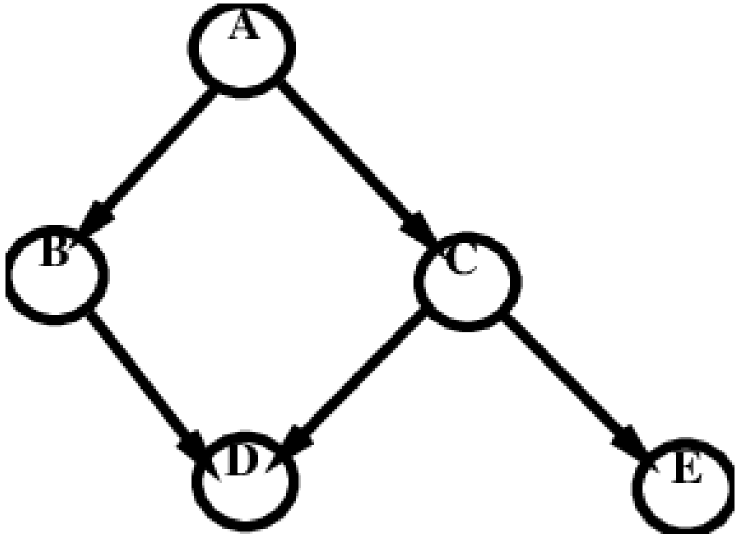
Task Parallelism



Key Characteristics:

WAH Campus

1. **Independent Tasks**: Each task can run on different processors with minimal synchronization.
2. **Asynchronous Execution**: Tasks can be completed at different times.
3. **Heterogeneous Workloads**: Different processors may perform different operations.
4. **Communication**: Tasks may need to exchange data or communicate, but it is limited compared to data parallelism.



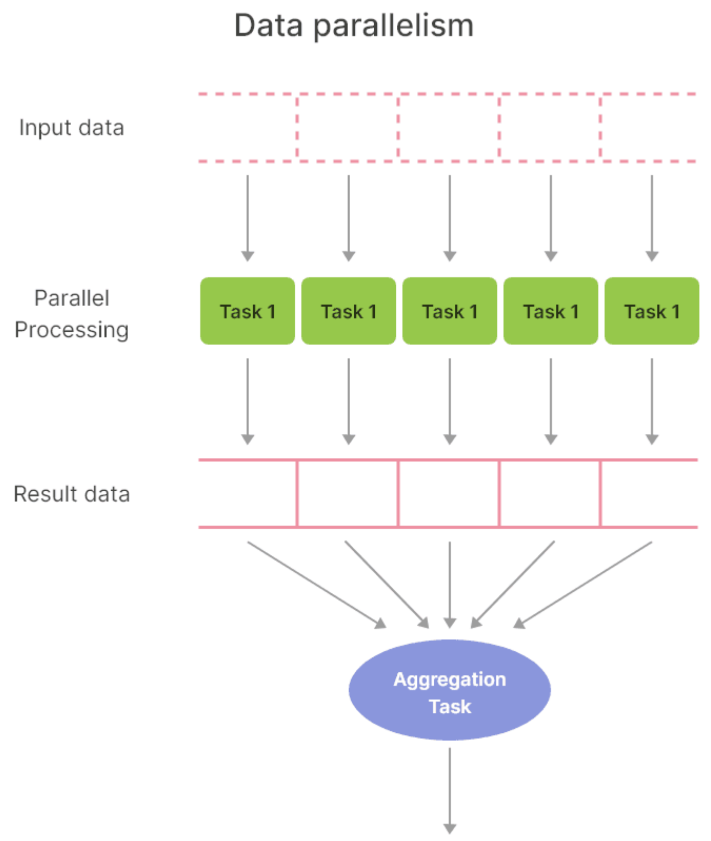
Advantaged and Challenges

- **Advantages**:
  - Efficient utilization of multi-core processors.
  - Suitable for problems where different tasks can be identified and run concurrently.
  - Can handle heterogeneous workloads well.
- **Challenges**:
  - Managing inter-task communication and synchronization.
  - Task scheduling becomes complex when tasks depend on each other.
- **Example**: In a web server, multiple client requests can be processed in parallel, where each request may be an independent task running concurrently on different threads or cores.

Data Parallelism

WAH Campus

- **Definition**
  Data parallelism focuses on distributing data across multiple processors and performing the same operation on each data subset simultaneously.
- **Concept**
  In data parallelism, the same task or operation is applied to different data chunks. Instead of each processor performing a different task (as in task parallelism), all processors perform the same operation but on different portions of the data.



Data Parallelism Continue

- For example, in matrix multiplication:
- A large matrix can be divided into smaller sub-matrices.
- Each processor multiplies a different sub-matrix simultaneously.

Key Characteristics:

1. **Same Operation, Different Data**: All processors perform the same task but on different pieces of data.
2. **Synchronous Execution**: Processors execute the same instruction simultaneously, but on different data sets.
3. **Scalability**: Works well when large datasets need to be processed in parallel.
4. **Shared Memory/Distributed Memory**: Can be implemented in both shared-memory systems (using threads) and distributed-memory systems (using processes).

Advantages and Challenges

WAH Campus

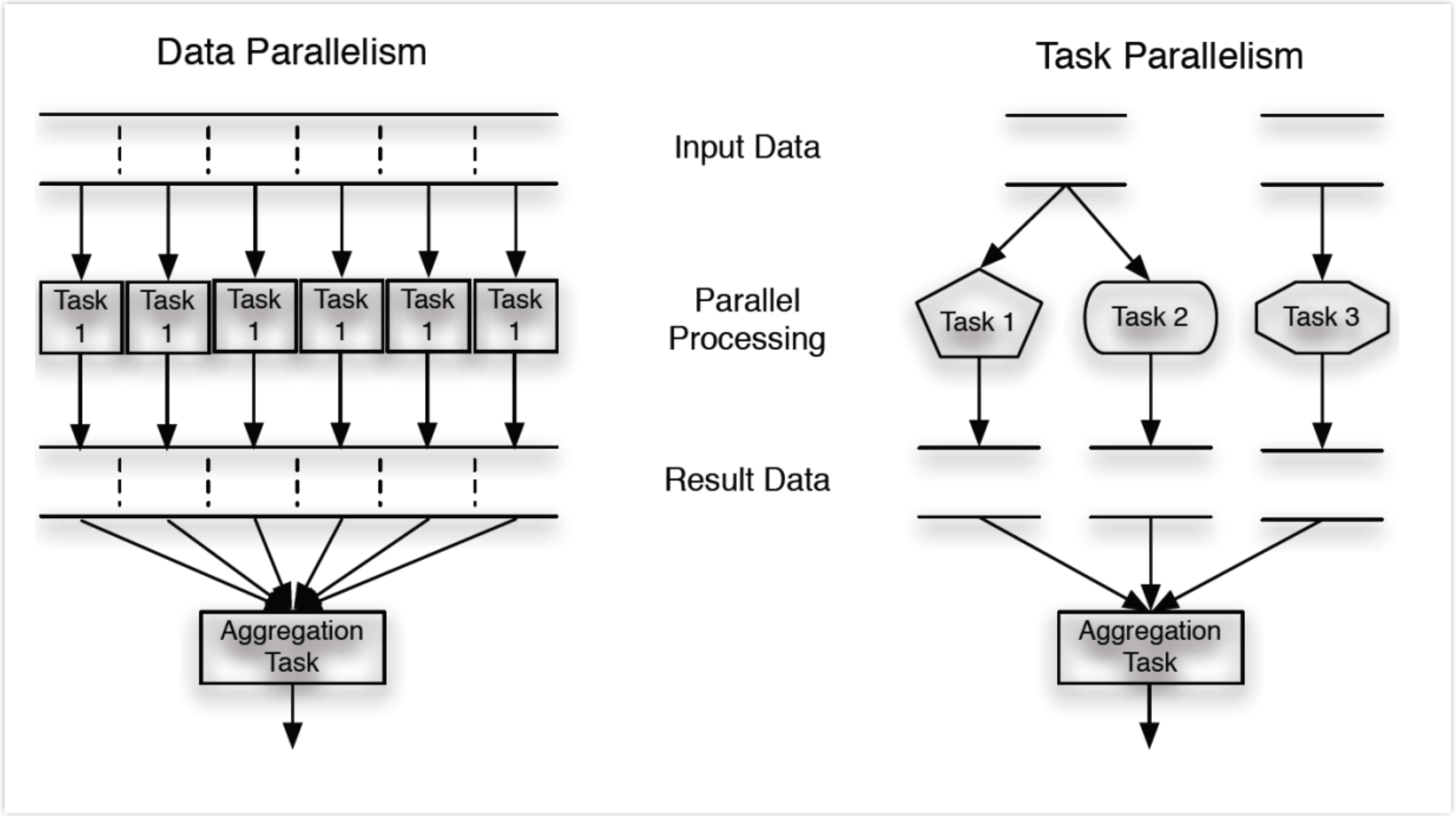
- **Advantages**:
  - Highly scalable, especially with large datasets.
  - Efficient for problems where the same computation is applied to a large amount of data.
  - Easy to implement when the data can be divided evenly among processors.
- **Challenges**:
  - Ensuring the data is divided evenly across processors.
  - Synchronization overhead if processors finish at different times.
  - Data dependency issues if parts of the dataset depend on each other.
- **Example**: Image processing algorithms like applying filters to images can be parallelized by dividing the image into smaller blocks, and each processor applies the same filter to different blocks of the image concurrently.

Key Differences Between Task and Data Parallelism

Feature	Task Parallelism	Data Parallelism
Execution	Different tasks running concurrently	Same task on different data simultaneously
Independence	Tasks are largely independent	Data is divided, and each processor works on its subset
Processor Usage	Heterogeneous workload across processors	Homogeneous workload across processors
Communication	Communication between tasks can be complex	Minimal communication between processors
Example	Web server handling multiple requests	Image processing with filters

Key Differences Between Task and Data Parallelism

WAH Campus



Hybrid Parallelism

WAH Campus

- In some cases, applications may benefit from combining both task and data parallelism, known as *hybrid parallelism*. Here, some processors may handle different tasks while others process the same task on different data. This combination can improve performance, especially in applications that have both distinct tasks and large datasets.
- For example, a scientific simulation may involve multiple phases where each phase can be treated as a task (task parallelism), while within each phase, large datasets can be processed using data parallelism.

Conclusion

WAH Campus

- Parallelism has revolutionized computing in both hardware and software systems.
- Task parallelism is effective for applications where different tasks can be performed concurrently.
- Data parallelism is ideal for processing large datasets with the same operations.
- Understanding when and how to use task and data parallelism is essential for optimizing performance in modern computing systems.

Practical Examples and Exercises

WAH Campus

Task Parallelism Example:

- Implement a program that reads data from multiple files simultaneously, processes each file's data in a different way, and writes the output to separate files.

Data Parallelism Example:

- Create a matrix multiplication program that divides the matrix into smaller sub-matrices and multiplies them concurrently using threads.