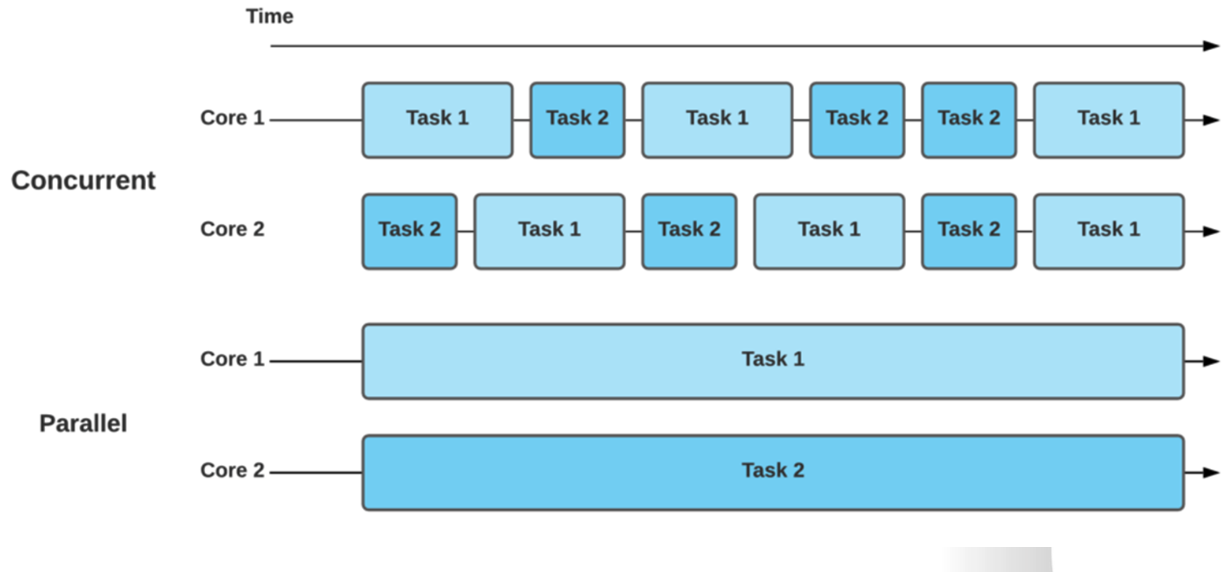
Parallel Programming

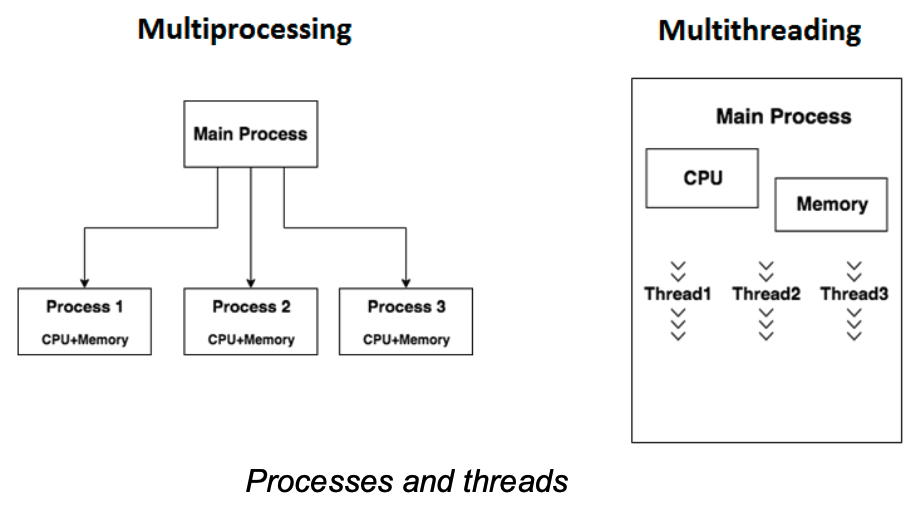


**Concurrency**

* Concurrency means **executing multiple tasks at the same time** in an overlapping manner.
* It is a way to structure a program to handle multiple operations independently.
* Doesn’t necessarily run in parallel; may use a single core with task switching.
* Useful for I/O-bound operations.

**Threading**

* Threading allows a program to run multiple threads (lightweight processes) within a single process.
* Threads share the same memory space, enabling faster context switching.
* Best suited for I/O-bound tasks.
* Python provides a threading module for creating and managing threads.



**Multiprocessing**

* Involves using **multiple CPU cores** to run tasks in **true parallelism**.
* Each process has its own memory space (unlike threads).
* Suitable for CPU-bound operations.
* Python’s multiprocessing module allows spawning processes using a simple API.

**Multithreading**

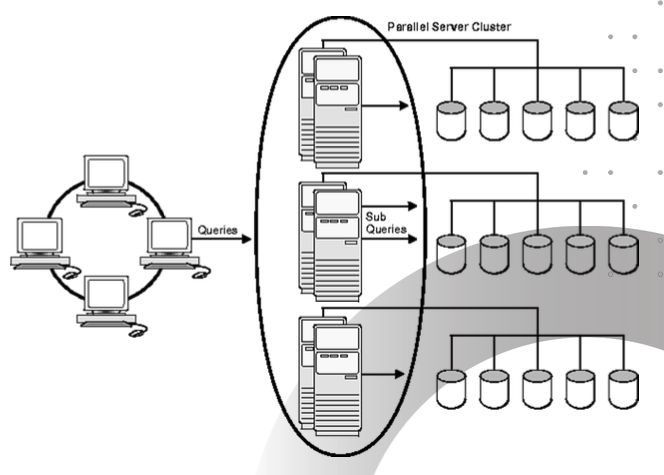
* A specific form of concurrency where multiple threads run in a single process.
* Python's Global Interpreter Lock (GIL) can limit multithreading performance for CPU-bound tasks.
* Ideal for tasks like web scraping, network I/O, and user interfaces.

**Parallelism**

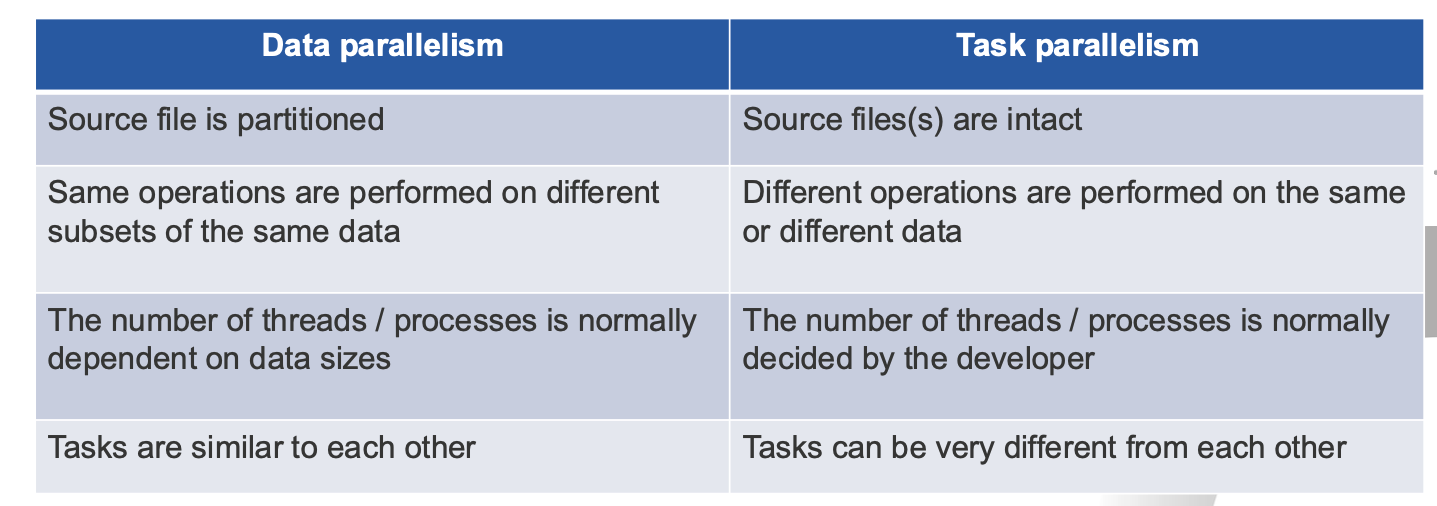
* A type of concurrency where **multiple tasks are executed simultaneously** on multiple processors.
* Requires dividing a problem into subtasks that can run concurrently.
* Results in significant performance improvements for large-scale computations.

**Distributed Computing**

* Tasks are executed on multiple machines (nodes) over a network.
* Used for large-scale data processing and high-performance computing.
* Examples include Apache Spark, Hadoop, and cloud computing.



**Task Parallelism Vs Data Parallelism**



**asyncio Library**

* Built-in Python library for writing asynchronous code using coroutines.
* Works on a single thread using event loops.
* Ideal for I/O-bound and high-level structured network code.
* Allows non-blocking code using async, await.

**threading Library**

* Python’s standard library for managing threads.
* Provides tools to create, start, and manage threads.
* Good for lightweight concurrent tasks but limited by the GIL.

**Dask**

* A flexible parallel computing library for analytics in Python.
* Scales from a single machine to a cluster.
* Works with NumPy, pandas, and scikit-learn.
* Allows handling of larger-than-memory datasets and complex workflows.