Thread pooling is a technique used in many applications to manage and ensure the tasks are executed in a timely and efficient manner.

When a task needs to be executes, a thread is assigned to it. Once the task is completed, the thread is returned back to the pool for reuse.

How it works ?

* Fixed number of threads are created and kept idleuntil task to be executed.
* When task in submitted to thread pool, a thread is assigned to it.
* Once the task is completed the thread is returned back to pool.
* Ensures system resources are used efficiently and task executed in timely efficient manner.

Benefits :

* Thread pooling is implemented by ExecutorService interface and various implementation provided in jav.util.concurrent package.
* Provides thread pool configuration
  + Fixed thread pool Size
  + Dynamic resizing of thread pool size
  + Prioritization of task

When thread pool is required ?

* Thread pooling is necessary in application that handle a large number of short-lived task or need to scale a large number of concurrent tasks.





Real -time comparison

* Thread pooling can be compared to a restaurant kitchen. In a restaurant kitchen, there are several chefs working simultaneously to prepare different dishes. Each chef has a specific task, and they work together to ensure that all dishes are prepared quickly and efficiently. Similarly, in thread pooling, a fixed number of threads are assigned to specific tasks, and they work together to ensure that all tasks are executed efficiently.

Conclusion :

By understanding the benefits of thread pooling and how to implement it using Java’s ExecutorService interface, developers can build applications that are efficient, scalable, and performant.