**Common Approaches to Parallelization:**

* **Process-based Parallelization:** This method involves running multiple independent processes simultaneously, each with its own dedicated memory. These processes communicate via protocols like message passing or by sharing resources. *Example*: Running different applications in parallel across a computer cluster, as seen in large-scale scientific simulations or in handling independent services on web servers.
* **Thread-based Parallelization:** This approach uses multiple threads within a single process, all sharing the same memory. Each thread can run on separate CPU cores, enhancing performance, particularly on multi-core systems. *Example*: In video editing software, one thread might handle rendering, another manages playback, and a third takes care of file input/output, all functioning concurrently.
* **Vectorization**: This technique enables a single instruction to operate on several data points simultaneously using vector registers, making it highly efficient for repetitive mathematical tasks. *Example*: Financial systems can leverage vectorization to analyse multiple stock prices at once during portfolio evaluations.
* **Stream Processing:** Stream processing breaks data into smaller pieces, processing them concurrently across many GPU cores. This allows for significant parallelism. *Example*: GPU acceleration can be used for real-time data analysis, where vast datasets are processed in parallel for immediate insights.
* **How GPUs Enable Stream Processing:**

GPUs divide tasks into smaller units called streams, distributing them across numerous small cores for parallel execution. This architecture is particularly effective for large-scale, repetitive tasks, such as real-time 3D rendering or deep learning model training.