Parallel Data Processing Engine: CPU–GPU Correlation Matrix Computation

Objective

Design and evaluate parallel implementations of a computationally intensive data-processing task on both CPU and GPU, using correlation matrix estimation from a large collection of time series as a concrete case study. The goal is to understand the benefits, limits, and trade-offs of CPU-based and GPU-based parallelism as data dimensionality scales.

Core Tasks

- Construct or obtain a dataset consisting of **N parallel time series**, with N ranging from small to very large values.
- Implement a **single-threaded CPU baseline** for estimating the full correlation matrix.
- Implement a **multi-core CPU parallel version** using multiprocessing or multithreading.
- Implement a **GPU-accelerated version** using an appropriate library (e.g., CuPy, PyTorch, or similar).
- Ensure numerical correctness and consistency of results across implementations.
- Benchmark runtime and memory usage across implementations as N and time-series length increase.

Additional Tasks

- Analyse and compare **data transfer overheads** between CPU and GPU, and their impact on end-to-end performance.
- Identify **problem-size regimes** where GPU acceleration provides meaningful speedups and where it does not.
- Compare at least **two GPU computation strategies** (e.g., full matrix computation vs block-wise computation).
- Include at least one scenario where GPU memory constraints require redesigning the computation.
- Relate empirical results to **theoretical computational and memory complexity** ($O(N^2)$ correlations).
- Revise at least one implementation after profiling reveals performance or memory bottlenecks.

Expected Outcome

A systematic, evidence-based comparison of CPU and GPU parallelisation strategies for large-scale correlation matrix estimation, providing practical insight into when GPU acceleration is advantageous and how parallelism interacts with data size, memory limits, and system architecture.