Performance Comparison Report

Processing System	Time (seconds)
Sequential Processing	0.31
Parallel (2 Workers)	0.42
Parallel (4 Workers)	0.38
Parallel (8 Workers)	0.55
Distributed (2 Nodes)	0.45

Best Number of Workers:

The best performance in parallel mode was achieved using **4 workers** (0.40s), although **sequential processing remained the fastest overall (0.31s)**.
This happened because the dataset is small, and the time required to start parallel processes and manage communication introduced overhead.
So, parallelism did not provide a speed benefit here.

Discussion:

In this experiment, parallel and distributed processing were compared with sequential execution. While parallelism typically improves performance by allowing multiple tasks to run simultaneously, in this case the dataset was small and the image processing operations were lightweight. As a result, the overhead of creating multiple processes, transferring data between them, and synchronizing their results was greater than the actual computation work. Therefore, sequential processing turned out to be the fastest. Parallel and distributed approaches did not show performance improvement because the workload per image was too small to benefit from CPU parallelization. This demonstrates that parallelism is helpful mainly when the dataset or computation is sufficiently large to offset multiprocessing overhead.