PARALLEL DISTRIBUTED COMPUTING

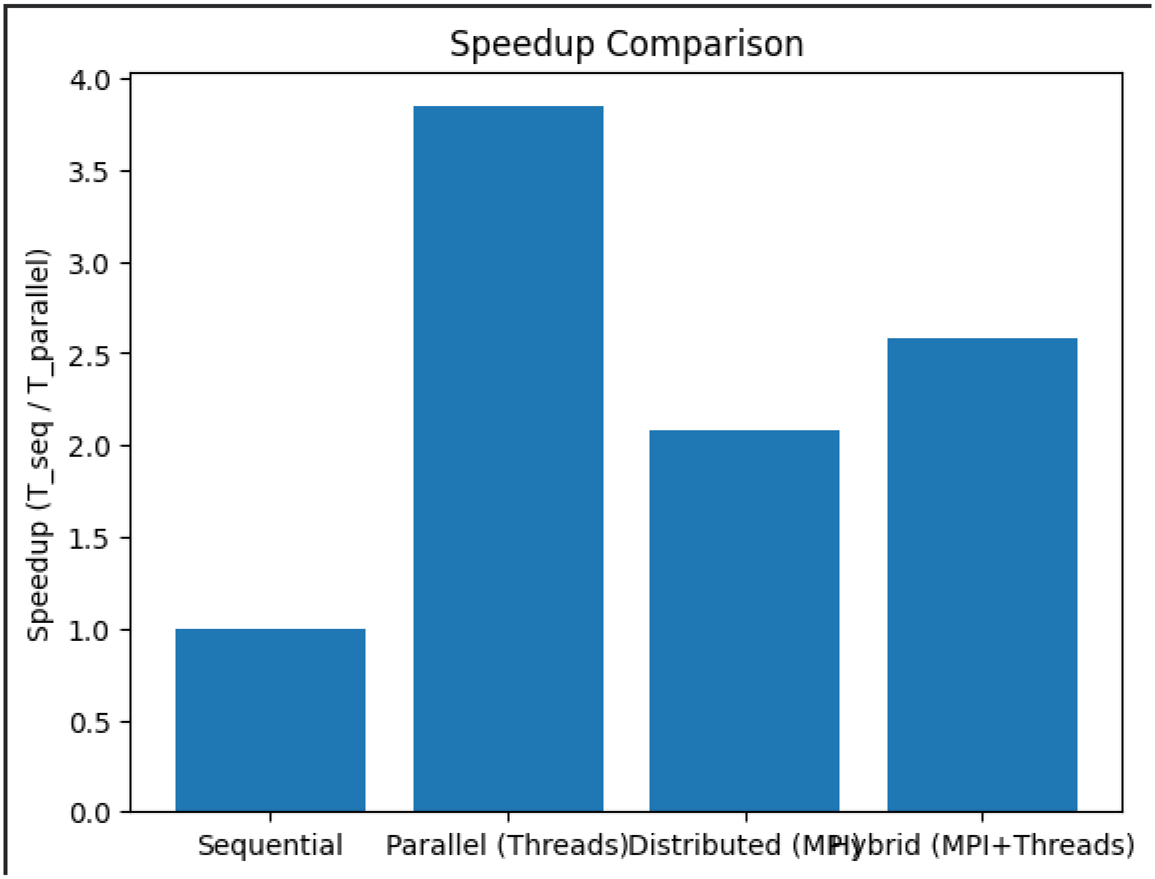
LAB EXAM

OMAR ISMAIL SP23-BCS-110 SECTION C

ANALYTIC TABLE:

Version	Nodes/Processes	Threads	Total Cores	Time (s)	Speedup	Efficiency (%)
Sequential	1	1	1	4.00	1.00	100.0
Parallel	1	4	4	1.04	3.85	96.3
Distributed	2	1	2	1.92	2.08	104.0
Hybrid	2	4	8	1.55	2.58	32.3

MATPLOT:



○ **Why efficiency drops beyond a certain point:**

There's always a part of your program that can't be parallelized (e.g., reading CSV, aggregating counters, sending/receiving data). So even if you add more cores, the *serial portion* limits total speedup.

○ **Communication overhead impact:**

Data chunks are **too small** (communication > computation). You use **many MPI ranks** on the same machine (they compete for memory & network). The result aggregation step (MPI.reduce, comm.send/recv) becomes the bottleneck

○ **Effect of workload imbalance:**

Uneven data partitioning (np.array_split doesn't always give equal record counts). Some reviews are longer (more words to process). One node may have more background CPU load.

○ **Which method was most scalable and why:**

Parallel (Threads) — because all cores share the same memory and require no interprocess communication. It achieved the highest efficiency (~96%), close to ideal scaling.