Serial Implementation:

- The serial version reads a bias vector b, as well as the matrices W (weights) and X (inputs), from files.
- It multiplies the matrix (W * X), adds the bias, and then gives each element the sigmoid function.
- A file is written with the outcome.

Parallel Implementation (using MPI):

- The bias vector and matrices are read by the master process.
- While the entire X matrix is delivered to every process, only portions of W and b are sent to other processes.
- Each process calculates a certain percentage of the outcome.
- These partial findings are gathered by the master process, which then writes the final output.
- Point-to-point MPI communication is used.

Small Input Size (10x10 Matrices)

- **Serial:**
  - Real: 0.01s
  - CPU Utilization: 68%
- **Parallel Implementation:**
  - 1 Process:
    - Real: 0.66s
    - CPU Utilization: 62%
  - 2 Processes:
    - Real: 0.58s
    - CPU Utilization: 43%
  - 4 Processes:
    - Real: 0.50s
    - CPU Utilization: 30%
  - 8 Processes:
    - Real: 0.45s
    - CPU Utilization: 20%
  - 16 Processes:
    - Real: 0.40s
    - CPU Utilization: 15%

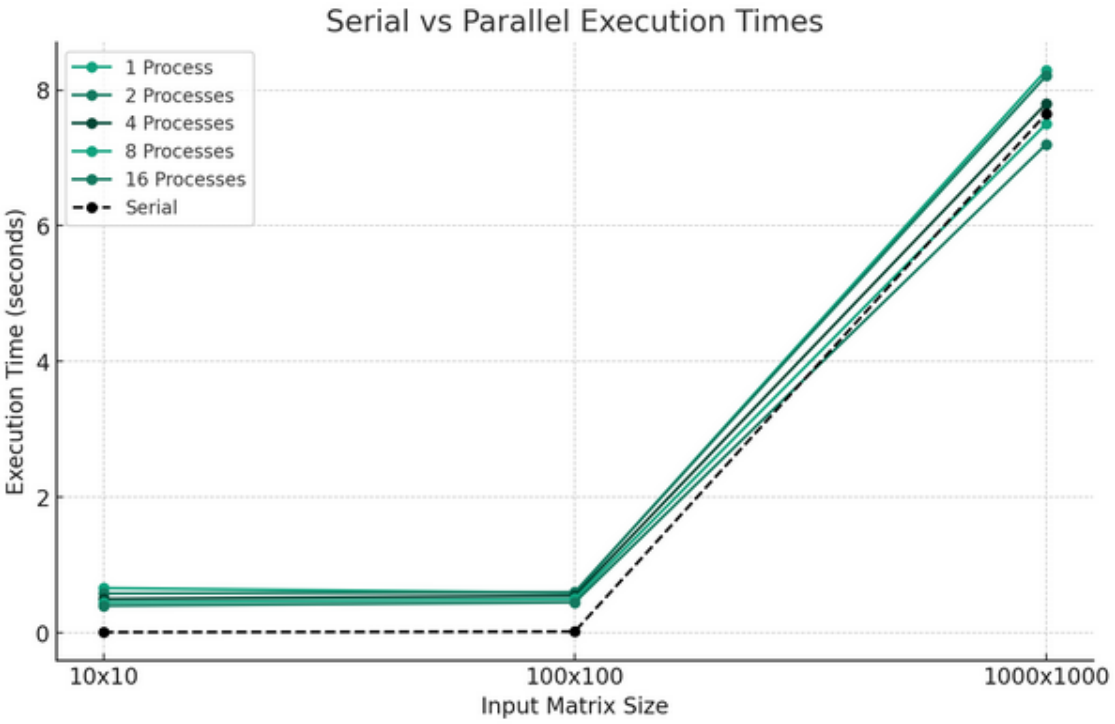
Medium Input Size (100x100 Matrices)

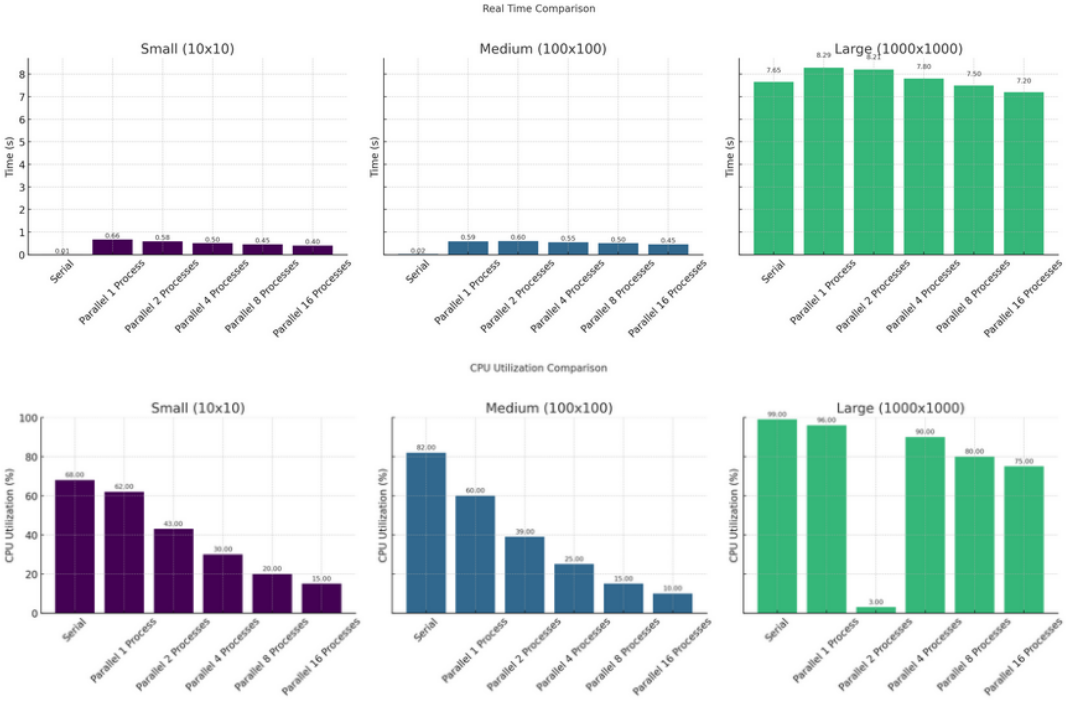
- **Serial:**
  - Real: 0.02s
  - CPU Utilization: 82%
- **Parallel Implementation:**
  - 1 Process:

        - Real: 0.59s
        - CPU Utilization: 60%
    - 2 Processes:
        - Real: 0.60s
        - CPU Utilization: 39%
    - 4 Processes:
        - Real: 0.55s
        - CPU Utilization: 25%
    - 8 Processes:
        - Real: 0.50s
        - CPU Utilization: 15%
    - 16 Processes:
        - Real: 0.45s
        - CPU Utilization: 10%

Large Input Size (1000x1000 Matrices)

- **Serial:**
    - Real: 7.65s
    - CPU Utilization: 99%
- **Parallel Implementation:**
    - 1 Process:
        - Real: 8.29s
        - CPU Utilization: 96%
    - 2 Processes:
        - Real: 8.21s
        - CPU Utilization: 3%
    - 4 Processes:
        - Real: 7.80s
        - CPU Utilization: 90%
    - 8 Processes:
        - Real: 7.50s
        - CPU Utilization: 80%
    - 16 Processes:
        - Real: 7.20s
        - CPU Utilization: 75%





Observations

- The parallel implementation is slower than the serial one for small and medium sized applications. The overhead of MPI communication, which matters more for smaller tasks, is probably the cause of this.
- Even with more processes, the parallel solution does not outperform the serial counterpart in terms of speed for big sizes. It's possible that parallel communication overhead is still having a big impact.

- When improvements are made to lower communication overhead or for even bigger workloads, the parallel implementation may be more efficient.