Parallel Matrix Multiplication Performance Evaluation

This report documents the performance evaluation of four matrix multiplication implementations: Sequential, MPI-Only, MPI + OpenMP (Hybrid), and MPI + OpenCL (Hybrid).
System Configuration:

- **Processor (CPU):** Macbook\ M3\ Pro\ (ARM64)
- **Operating System:** macOS\ ARM64
- **OpenCL Device (GPU/CPU):** Integrated\ GPU\ (Apple\ M3\ Pro)
- **Max Cores Tested:** 4\ MPI\ Processes\ (np=4)\ and\ 11\ OMP\ Threads\ per\ Process

1. Program Implementations Summary

Program	Parallel Model	Key Technology	Parallelization Strategy
Sequential	Serial Baseline	C++	N/A
MPI-Only	Distributed	MPI	Scatter rows of A to processes; each process computes its local C block.
MPI + OpenMP	Hybrid	MPI (inter-node) + OpenMP (intra-node)	MPI scatters rows; OpenMP threads within each MPI process compute the local C block.
MPI + OpenCL	Hybrid	MPI (inter-node) + OpenCL (intra-node)	MPI scatters rows; OpenCL offloads local C block computation to a GPU/CPU device.

2. Performance Data

All timing measurements are in **Milliseconds (ms)**.

Table 2.1: Sequential vs. MPI-Only (CPU Cores Only)

Matrix Size (N)	Sequential Time (ms)	MPI Time (np=4) (ms)	Speedup (Seq / MPI)
512	**375.0**	**95.972**	**3.91**
1000	**2862.0**	**502.597**	**5.69**
2048	**46261.0**	**15797.9**	**2.93**

Table 2.2: Comprehensive Hybrid Performance Comparison

Matrix Size (N)	Sequential (ms)	MPI-Only (ms)	MPI + OpenMP (ms)	MPI + OpenCL (ms)	Best Speedup
512	**375.0**	95.972	**46.181**	70.379	**8.12x (OMP)**
1000	**2862.0**	502.597	314.115	**103.811**	**27.57x (OCL)**
2048	**46261.0**	15797.9	11163.5	**141.853**	**326.11x (OCL)**

3. Findings and Conclusion

3.1. Comparison of Parallel Models

MPI-Only vs. Hybrid Models: The pure distributed model (MPI-Only) achieved a 5.69x speedup for N=1000, but both hybrid models significantly outperformed it.

OpenMP vs. OpenCL (GPU Acceleration): The MPI + OpenCL implementation demonstrated superior scaling as the problem size increased.

Two Regimes of Efficiency:

- **Small Problems (**N=512**):** The MPI + OpenMP model is the most efficient. The overhead of initial **OpenCL setup** and data transfer (70.379 ms) slightly outweighs the benefit, making the pure CPU-based, shared-memory approach (46.181 ms) the optimal choice for quick, smaller tasks.
- **Large Problems (**N=1000 **and** N=2048**):** The MPI + OpenCL model becomes overwhelmingly dominant. Its speedup explodes from 27.57x at N=1000 to an incredible 326.11x at N=2048.

- Failure of CPU Scaling:
  Both MPI-Only and MPI + OpenMP show severe performance degradation at N=2048. The Sequential time increased by a factor of ~16 between N=1000 and N=2048, but the MPI and OpenMP times increased even faster, resulting in declining speedup (from 5.69x to 2.93x for MPI). This confirms that the CPU cores and memory system are heavily saturated by the computational load, causing massive contention and negating the benefit of parallelism.
- OpenCL Dominance:
  The MPI + OpenCL implementation demonstrates near-perfect scaling and is entirely bottlenecked by the CPU's memory capacity for the purely CPU-bound models. The OpenCL time only increased from 103.8 ms to 141.9 ms between N=1000 and N=2048, confirming the integrated GPU's superior ability to handle high-density parallel workloads without suffering from the CPU's memory limitations.

3.2. Final Recommendation

Based on the complete performance data:

1. **For Small Problems (N < 1000):** Use **MPI + OpenMP**. It offers the highest speedup () due to its minimal overhead.
2. **For Large Problems (N 1000):** The **MPI + OpenCL** implementation is the **mandatory choice**. It is the only model that scales effectively, providing hundreds of times the performance of the CPU-based models by leveraging the massive parallelism of the integrated on the platform.