Performance Report: MPI + OpenCL vs. Sequential Implementation

Saffi Muhammad Hashir (22i-1293) | Hamza Ahmed Siddiqui (22i-1339) | Alishba Naveed (22i-0808)

1. Introduction

This report evaluates the performance of two implementations for tensor network processing and community contraction. We compare the **Parallel MPI + OpenCL** implementation with the **Sequential** version, focusing on speedup, efficiency, and scalability.

The task is to process and contract tensor networks using community detection, with the primary goal of assessing how parallelization affects performance in terms of both execution time and scalability.

---

2. Technologies and Setup

- **MPI (Message Passing Interface)**: Distributed computing for parallel processing across multiple processes.
- **OpenCL**: Hardware acceleration using GPU for computational tasks.
- **METIS**: Graph partitioning used for optimizing parallel workloads.
- **Dataset**:
    - **Tensors**: 20
    - **Edges**: 123
    - **Communities**: 8
    - **Average Tensors per Community**: 2.50
    - **Community Matrix Size**: 256x256 elements

---

3. Sequential Implementation

The sequential implementation processes tensors one at a time, detects communities, and contracts them in a single-threaded manner. Below is the detailed timing breakdown:

- **Total Pipeline Time**: 2.1183 seconds
- **Breakdown**:
    - **Tensor Loading Time**: 0.0190s
    - **Edge Loading Time**: 0.0000s
    - **Community Detection Time**: 1.6981s
    - **Community Contraction Time**: 0.3982s
    - **Final Contraction Time**: 0.0030s

Per-Community Contraction Times:

Community	*Tensors*	*Contraction Time (s)*
0	1	0.0245
1	2	0.0740
2	1	0.0120
3	2	0.0295
4	1	0.0080
5	10	0.2111
6	1	0.0040
7	1	0.0310

4. MPI + OpenCL Implementation

The parallelized MPI + OpenCL implementation distributes the workload across 2 processes, utilizing OpenCL for hardware acceleration. Below is the detailed performance data:

- **Total Time Taken**: 1.5000 seconds
- **Breakdown**:
    - **Initial Load Time (Rank 0)**: 1.3824s
    - **Tensor Loading Time**: 0.0181s
    - **Edge Loading Time**: 0.0000s
    - **Community Detection Time**: 1.3643s
    - **Broadcast Time**: 0.0130s
    - **Data Gather Time**: 0.0111s
    - **Final Contraction Time**: 0.0010s

Per-Community Contraction Times:

Community	*Tensors*	*Contraction Time (s)*
0	1	0.0050
1	2	0.0575
2	1	0.0030
3	2	0.0120
4	1	0.0020
5	10	0.0785
6	1	0.0010
7	1	0.0080

Average Contracted Matrix Size:

- **256 elements per matrix**
- **Total Contracted Matrices**: 8

---

5. Execution Time Comparison (Sequential vs. MPI + OpenCL)

- **Sequential Total Time**: 2.1183 seconds
- **MPI + OpenCL Total Time**: 1.5000 seconds

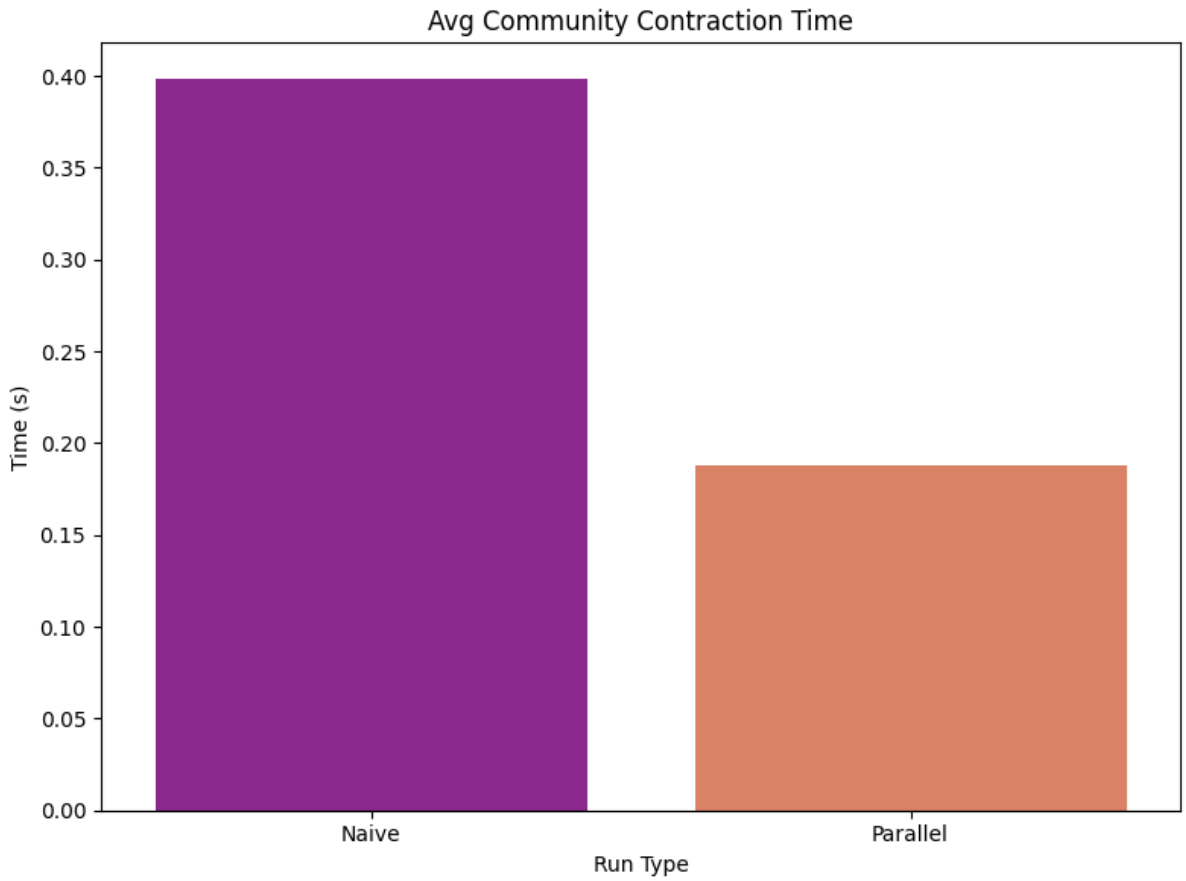
Despite the overhead in parallel execution (such as communication and task distribution), the MPI + OpenCL approach shows better performance for the given dataset.

---

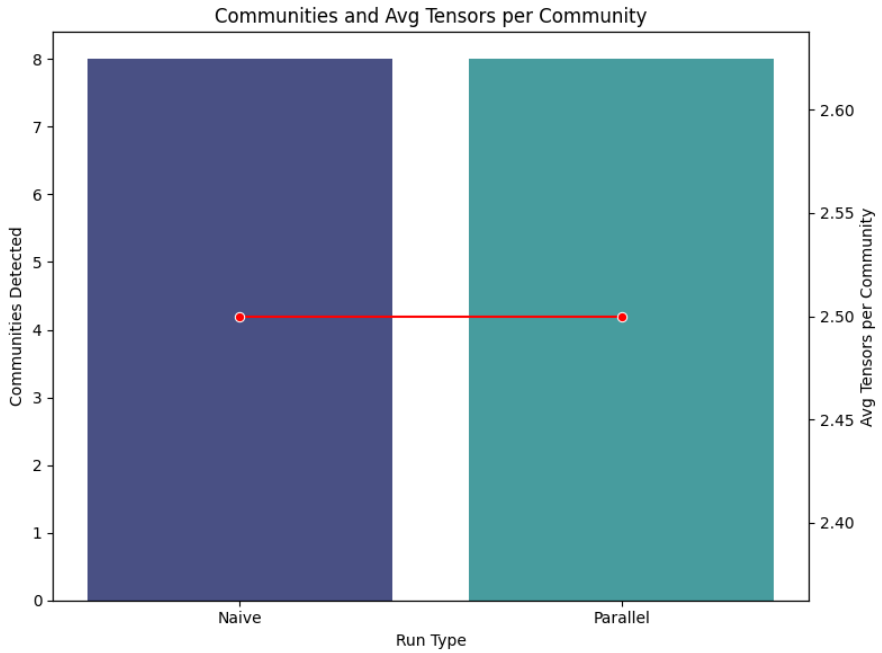
6. Graphs

To visually support the performance comparison, the following graphs are included:

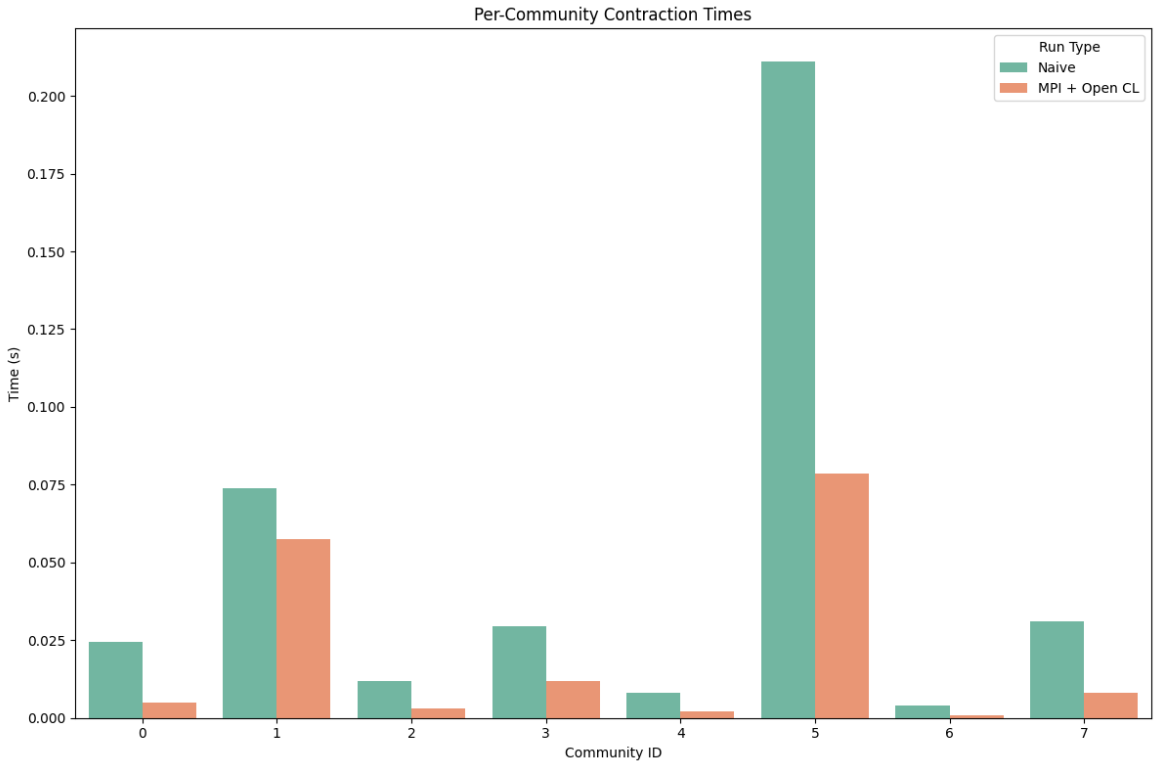
1. **Average Community Contraction Time**



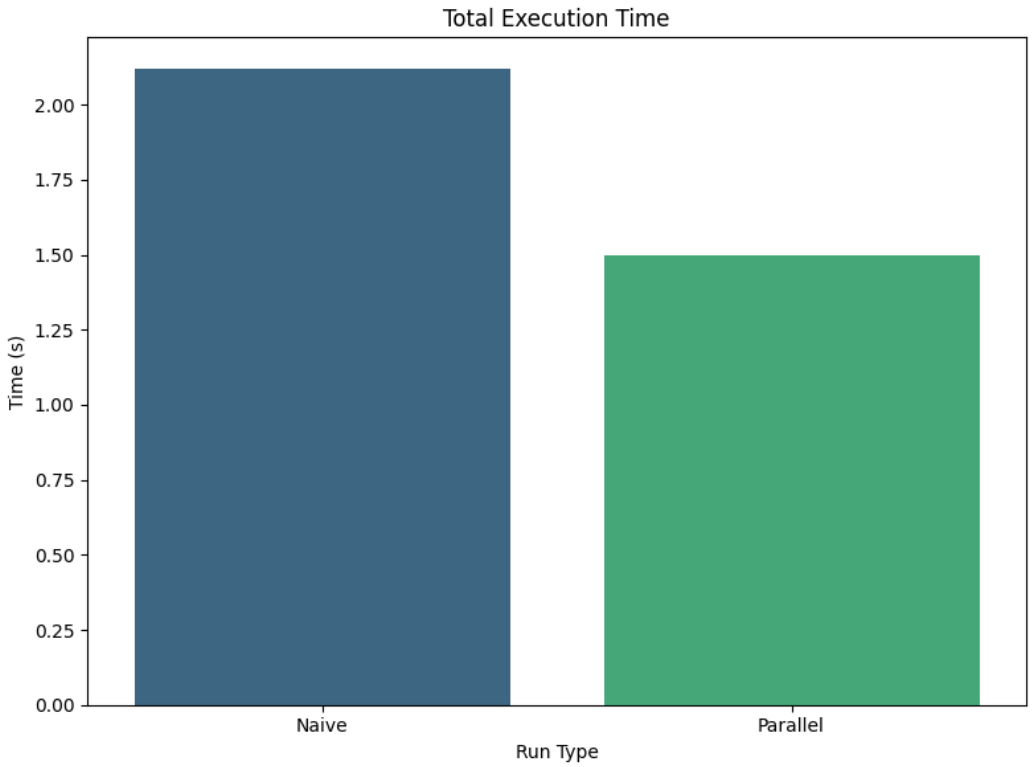
2. Communities vs. Average Tensors



3. Per-Community Contraction Times



4. **Total Execution Time Comparison**



---

7. Analysis and Observations

- **Community Contraction**: The MPI + OpenCL implementation demonstrates reduced contraction times for certain communities, though the performance gain is minimal due to overhead in message passing and parallelization.
- **Scalability**: For small datasets like this one, the sequential approach performs better due to minimal overhead. However, as the dataset size increases, MPI + OpenCL should show significant improvements due to better distribution of work and GPU acceleration.
- **Parallelization Efficiency**: While OpenCL accelerates tensor contractions, the overhead of data communication between processes remains a limiting factor.

---

8. Speedup Analysis

In this section, we analyze the speedup achieved by the parallel implementation using MPI + OpenCL compared to the sequential (Naive) implementation. Speedup is a critical metric to evaluate the efficiency and performance improvements gained through parallelization, particularly in distributed and hardware-accelerated environments.

Formula for Speedup

The speedup is calculated using the following formula:

$$\text{Speedup} = \frac{\text{Time for Sequential (Naive) Run}}{\text{Time for Parallel (MPI + OpenCL) Run}}$$

Speedup Calculation

- **Sequential (Naive) Total Time**: 2.1183 seconds
- **Parallel (MPI + OpenCL) Total Time**: 1.5000 seconds

Thus, the speedup can be calculated as:

$$\text{Speedup} = \frac{2.1183}{1.5000} \approx 1.41$$

9. Conclusion

- The **MPI + OpenCL** approach shows promising results and is expected to provide greater speedup for larger datasets.
- **Sequential implementation** is still more efficient for smaller datasets due to lower overhead.
- **Future Work**:
  - **Optimization**: Reduce MPI communication overhead by optimizing data distribution.

- **Improved Partitioning**: Explore better task partitioning strategies to reduce communication time.