Performance of SPMV Calculation using OpenMPI

Charlie Flux

***Abstract*—This paper investigates the performance of sparse matrix-vector multiplication (SPMV) using naive and structured approaches using OpenMPI. This paper focuses on the COO-to-CSR conversion and workload partitioning.**

I. Introduction

The purpose of this experiment is to compare the performance of naive and structured implementations of SPMV using OpenMPI. Specifically, it examines how varying rank counts affect the performance of distributed sparse matrix-vector multiplication algorithms.

II. Methodology

2 Approaches of calculating SPMV were used:

- **Naive**
- **Strucuted COO-to-CSR conversion with reduction**

For experiments, measuring time, rank counts R were chosen from the set:

$$R = \{1, 2, 4, 6, 8, 12, 16, 24, 32\}$$

All experiments used the arguments the matricies provided by cant.mtx & b.mtx files.

A. Experimental Setup

All tests were ran on Talepas. Experiments were repeated 10 times and the average was calculated for each experiment. Data was filtered using the Interquartile Range Filter to remove points that do not lie between the first and third quartile as a consequence of the the unreliable data caused by the unlocked variable frequencies on Telepas.

III. Results

A. Broadcast

The first stage of both methods is to broadcast data out to all nodes. Both the naive and structured approach perform similarly. This is expected as the data that is broadcasted is the same therefore, with no differences in implementation, similar results are expected.

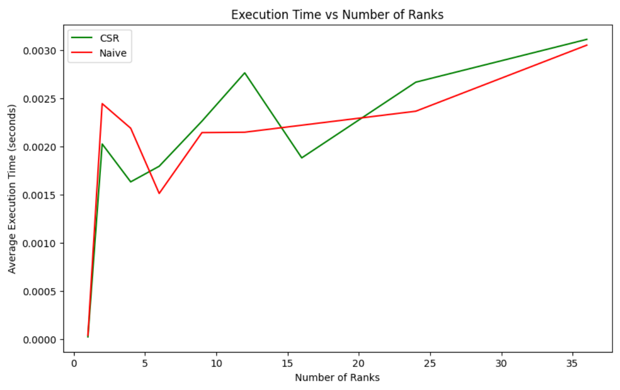

Fig. 1: SPMV GPU Implementations

B. Scatter

The next stage is to scatter the rows that each node will be processing out. The CSR format performs better than the COO format during the scatter due to the reduced amount data that is needed to store the matrix. I implemented load balancing on the rows by calculating an equal amount of non zero values for each node to process, this should result in a speedup during the SPMV calculation.

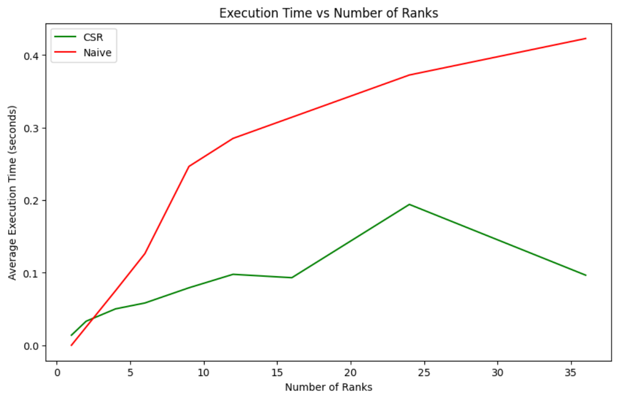

Fig. 2: CSR implementation

C. SPMV

The SPMV section of the code is similar in distribution however, the structured approach uses load balancing which is apparent in Figure 3. The structured approach sees an improvements as the number of ranks increases whereas, the naive approach suffers as more nodes are introduced. I believe that the naive approach suffered because of the lack of load balancing therefore, some nodes will have a more work to do than other which increases the wait time on a single node.

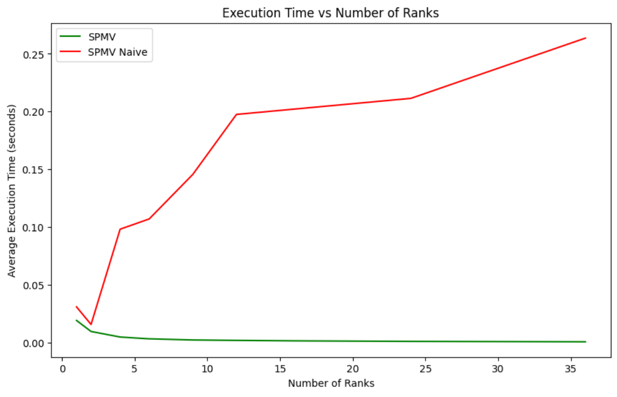

Fig. 3: SPMV Calculation

D. *Reduction*

The final stage of the SPMV calculation is the reduction on rank 0. The structured approach performs better than the naive approach as the number of nodes increases. I did not implement reduced local result array sizes which would offer a greater speedup on the structured compared to naive approach.

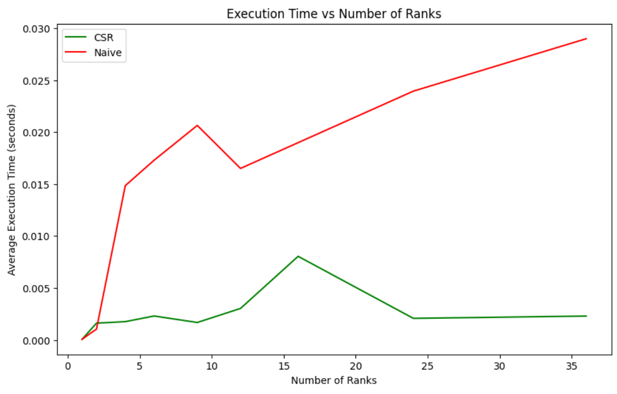

Fig. 4: COO to CSR Conversion

IV. Conclusion

This experiment compared the performance of SPMV matrix calucations using OpenMPI focusing on a structured vs naive approach. In conculsion the structured approach performed better across the board. This performance difference is due to the reduced data used by the CSR matrix format, effective load balancing which minimizes idle time and ensures more efficient workload distribution. However, further optimization, such as reducing local result array sizes, could enhance performance.