Performance of SPMV Calculation

Charlie Flux

***Abstract*—This paper investigates the performance of sparse matrix-vector multiplication (SPMV) using different matrix representations, specifically Coordinate List (COO) and Compressed Sparse Row (CSR), under varying thread counts. Both parallel and serial implementations are explored, with a focus on the effect of parallelization on execution time. Results show that CSR outperforms COO in a parallel environment however, COO is more effective in serial applications.**

I. Introduction

The purpose of this experiment is to compare the performance of different matrix representations. Specifically, it examines how varying thread counts affect the performance of parallel sparse matrix-vector multiplication algorithms.

II. Methodology

2 Methods of storing matricies were used:

- **Coordinate List (COO)**
- **Compressed Sparse Row (CSR)**

For experiments measuring time, core counts C were chosen from the set:

$$C = \{1, 2, 4, 6, 8, 12, 16, 24, 32, 48, 64, 96, 128\}$$

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 20 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. Serial implementation

Firstly, a serial calculation algorithm was implemented. Figure 1 shows the performance difference between the CSR and COO formats while calculating the Sparse Matrix-Vector (SPMV) multiplication. As expected, the CSR format performs worse on average due to the $O(n^2)$ time complexity of the algorithm implemented. This time complexity is inherent to the CSR method; however, it should become an advantage in a parallel application.

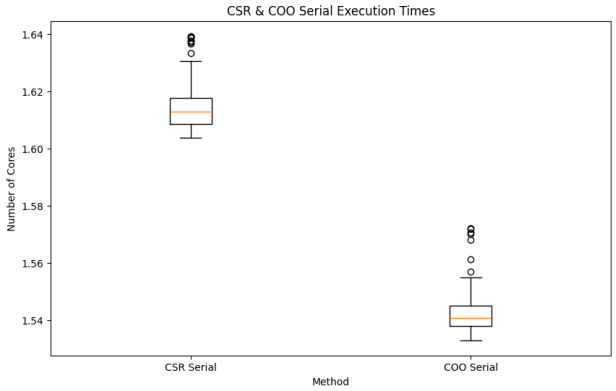

Fig. 1: Serial SPMV Implementations

B. Parallel implementation

Figure 2 has a logarithmic y-axis and compares the execution time of the parallel implementations of the COO & CSR sparse matrix-vector multiplication algorithms. In the CSR algorithm, each thread processes an entire row at a time, which improves cache locality and enables more efficient sequential memory access, leading to better performance. In contrast, the COO algorithm uses locks for each row to prevent race conditions when multiple threads access the same row. However, this locking mechanism introduces additional execution overhead. In the CSR algorithm, race conditions are avoided because each row is assigned to an individual thread, eliminating the need for locks.

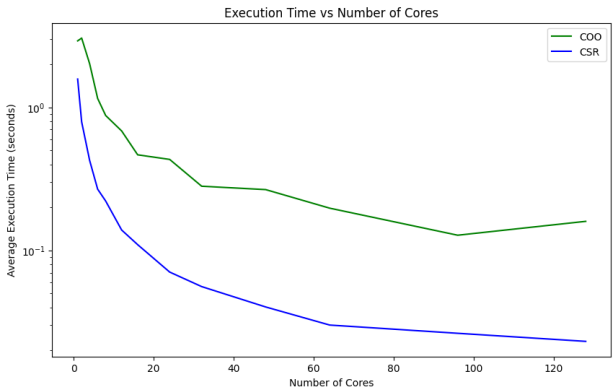

Fig. 2: Parallel SPMV Implementations

C. Conversion

Figure 3 illustrates the average execution time for the conversion from COO to CSR formats compared to the number of cores in a parallelized environment. A substantial speedup

can be found with an increase in core count up to a certain threshold, beyond which the performance gain diminishes, indicating limited scalability of the conversion process at higher core counts.

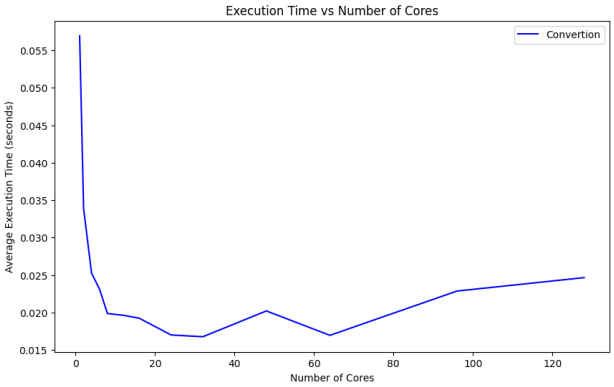

Fig. 3: COO to CSR Conversion

IV. Conclusion

In conclusion, the results show that the CSR format is more favourable in a parallel environment because of the optimised memory fetch operations coming from each row being assigned a thread. On the other hand, the COO format performs better in a serial environment because of the reduced time complexity of the algorithm used to calculate a sparse matix vector multiplication.