Casey Adams, Kevin Connell
ECEC 622
Project 6- CUDA Jacobi Solver
3/2/2021

CUDA: Jacobi Solver

Implementation

The Jacobi solving operation is one which can be significantly sped up through the use of parallelization. In this assignment, the operation was parallelized using CUDA on the xunil server. The process works by copying the matrix objects from the CPU to the GPU which has thousands of threads which can work on the problem simultaneously.

This specific implementation involved two techniques. One was the "naive" approach and the other was the "optimized" approach. The main difference between these approaches comes down to the structure of the matices. The optimized version was designed to ensure that accesses to the matrix elements in memory were coalesced. This is done by reordering the matrix in "column major" form.

Performance

Performance was tested on a CPU implementation (Gold) as well as both CUDA implementations discussed above. Times were recorded for each method operating on matrices of 512x512, 1024x1024, and 2048x2048 in size. The results of these tests can be seen in Table I below.

Table I Performance results for Jacobi Solver

Matrix Size	Gold Compute Time (s)	Naive Compute Time (s)	Optimized Compute Time (s)
512x512	3.65	18.73	18.13
1024x1024	29.71	46.75	41.14
2048x2048	241.49	96.41	91.60

These results include some notable patterns. One point of interest is that the CPU version performed better for the smaller matrix sizes of 512x512 and 1024x1024. This is likely because of the extra overhead associated with transferring data between the CPU and GPU for the CUDA solutions. However, as the matrix size increased, this overhead became less significant when compared to the total amount of computation time required. This resulted in improved performance for the CUDA approaches when the matrix was of size 2048x2048 and would likely continue to be the case for even larger matrices. A slight performance boost for the optimized version of the code was also seen although this was less significant than the improvement seen simply by parallelizing the gold example.

Conclusion

By parallelizing the jacobi solver with CUDA to take advantage of GPU parallel compute resources, the processing time for large matrices can be significantly reduced. This becomes more noticeable as the matrix size increases. It should be noted that for small matrices, the overhead associated with transferring data to the GPU and performing computation on the GPU results in worse performance.