CUDA Matrix Determinant Calculator

CLE - Computação em Larga Escala
Assignment 3
June 2022

Mário Silva - 93430
Pedro Marques - 92926

Matrix Determinant - Column Elimination

1. Read and process the command line.
2. Read the number of matrices in the file.
3. Read the order of the matrices in the file.
4. Initialize the array of matrices and determinants.
5. Load all the matrices.
6. Copy the matrices from the host to the GPU global memory.
7. Process and calculate the matrices determinants.
8. Retrieve the array of determinants from the GPU back to the host.
9. Print results.
10. For each matrix, calculate determinant using the CPU.
11. Print total elapsed time for both CPU and GPU operations.

Calculating the matrices determinants:

1. The thread corresponding to the current iteration calculates the pivot, and, multiplies the pivot to the determinant of that matrix.
2. Threads Synchronize.
3. Each thread, that is responsible for a row below the current pivot’s row, does the Gaussian Elimination on its row only.
4. Threads Synchronize.

Matrix Determinant - Each thread Processes a Row

Timing results for processing files, total of **20 tries**, using a GTX1660Ti

File	CPU		GPU	
	Avg. Elapsed Time (s)	Standard Deviation (s)	Avg. Elapsed Time (s)	Standard Deviation (s)
mat128_32.bin	0.0010526	0.000199412136	0.0003047	0.000001031095483
mat128_64.bin	0.007693	0.0001406567303	0.00210415	0.0000108641128
mat512_128.bin	0.21610135	0.003772436059	0.2275545	0.001084113438
mat512_256.bin	1.54656865	0.02553825263	2.0364482	0.003413536731

Matrix Determinant - Row Elimination

1. Read and process the command line.
2. Read the number of matrices in the file.
3. Read the order of the matrices in the file.
4. Initialize the array of matrices and determinants.
5. Load all the matrices.
6. Copy the matrices from the host to the GPU global memory.
7. Process and calculate the matrices determinants.
8. Retrieve the array of determinants from the GPU back to the host.
9. Print results.
10. For each matrix, calculate determinant using the CPU.
11. Print total elapsed time for both CPU and GPU operations.

Calculating the Determinant

- The thread corresponding to the current iteration calculates the pivot, and, multiplies the pivot to the determinant of that matrix.
- Threads Synchronize.
- Each thread, that is responsible for a column to the right of the current pivot's column, does the Gaussian Elimination on its column only.
- Threads Synchronize.

Matrix Determinant - Each Thread Processes a Column

Timing results for processing files, total of **20 tries**, using a GTX1660Ti

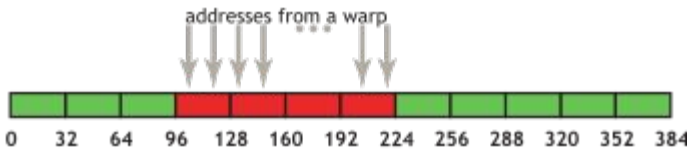
File	CPU		GPU	
	Avg. Elapsed Time (s)	Standard Deviation (s)	Avg. Elapsed Time (s)	Standard Deviation (s)
mat128_32.bin	0.0009545	0.0002358526525	0.00018205	0.000002762054918
mat128_64.bin	0.0059165	0.0001141059065	0.00103965	0.000004356181331
mat512_128.bin	0.4604272	0.008098785442	0.02513095	0.00002959636357
mat512_256.bin	31.26492845	0.6890058022	0.2012886	0.000433889678

Conclusion

“Is it worthwhile to use the GPU to run this kind of problem?”

On **CPU**, the **row reduction is usually faster** because the **memory access** during the subtraction of each row is **sequential** (+1), but in the **column reduction**, **for each column** it has to **access the value of the next row** (+order) which is not sequential memory access.

As for the **GPU**, the situation is the opposite, having a more **sequential memory access**, as the figure below shows, for the **column reduction method**, since the block threads are **writing values from the same row on each synchronized iteration**, and **for the row reduction, they are writing different rows for each synchronized iteration**.



Even with part of this problem having to be sequential, such as the partial pivoting, we believe **it is worth** to have this kind of problem run on the **GPU** to parallelize the matrix subtractions, but it has to be with the **column reduction method**, for the reason mentioned above.