

Computação em Larga Escala — Trabalho 3

P3 G1

Diogo Carvalho – 92969 – MEI

Rafael Baptista – 93367 – MEI

Programa 1 Open file Read number of matrices and order of matrices Get block id and thread id Allocate memory for storing matrices and determinants in host Adjust the matrix and in device pointer to the start of the matrix Read matrices from file and initialize determinants to 1 There are still Nο iterations to End process? Copy the host data to Yes device memory Run computational kernel with specific The diagonal Find a column to Save determinant Yes block and grid Column found? coefficient is 0? equal to 0 and end switch dimensions Copy the kernel results to the host Yes Switch the two elements of the two Sync threads Free the memory of columns of the line device equal to the thread id The thread id is Get the term of the There are more equal to the current current line to lines to update? Compute iteration? calculate determinants in CPU Yes Sync threads Show determinant Multiply the diagonal results of GPU and value to the CPU determinant Change the coefficient of this column and line Free the memory of host

Resultados programa 1

Relativamente aos resultados dos cálculos dos determinantes das matrizes comparámos com os resultados que o professor forneceu e verificámos que os resultados eram os mesmos.

Tempos de execução (s):

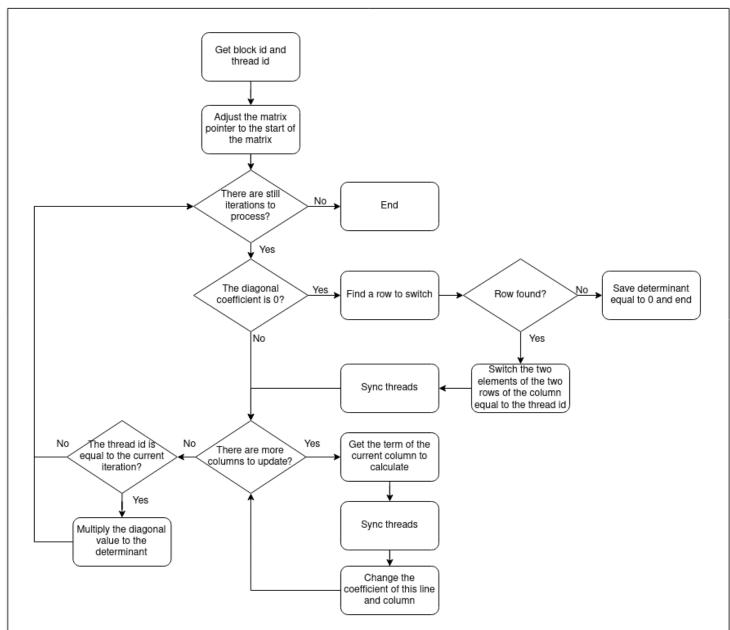
modo ficheiro	Cpu (Single Thread)	GPU (CUDA)	Speed up
mat128_32.bin	0.0007578	0.0003236	~2.34
mat128_64.bin	0.005771 0.002549		~2.26
mat128_128.bin	0.04804	0.01874	~2.56
mat128_256.bin	0.3747	0.1535	~2.44
mat512_32.bin	0.003097	0.001179	~2.63
mat512_64.bin	0.02327	0.008552	~2.72
mat512_128.bin 0.1928		0.06849	~2.82
mat512_256.bin	1.451	0.443	~3.28

Tempos de execução para o ficheiro mat512_32.bin com diferentes dimensões:

blockDim.x	blockDim.y	gridDim.x	gridDim.y	Time(s)
32	1	512	1	0.00116
16	2	512	1	0.00118
8	4	512	1	0.00117
4	8	512	1	0.00117
2	16	512	1	0.00116
1	32	512	1	0.00116
32	1	256	2	0.00118
32	1	32	16	0.00117
32	1	16	32	0.00117
32	1	2	256	0.00116
32	1	1	512	0.00115

Open file Read number of matrices and order of matrices Allocate memory for storing matrices and determinants in host and in device Read matrices from file and initialize determinants to 1 Copy the host data to device memory Run computational kernel with specific block and grid dimensions Copy the kernel results to the host Free the memory of device Compute determinants in CPU Show determinant results of GPU and CPU Free the memory of host

Programa 2



Resultados programa 2

Relativamente aos resultados dos cálculos dos determinantes das matrizes comparámos com os resultados que o professor forneceu e verificámos que os resultados eram os mesmos.

Tempos de execução (s):

modo ficheiro	Cpu (Single Thread)	GPU (CUDA)	Speed up
mat128_32.bin	0.001248	0.000441	~2.83
mat128_64.bin	0.01025	0.008527	~1.20
mat128_128.bin	nat128_128.bin 0.1316		~0.61
mat128_256.bin 4.595		1.766	~2.60
mat512_32.bin	0.004957	0.005442	~0.91
mat512_64.bin	0.0412	0.06955	~0.60
mat512_128.bin 0.5428		0.906	~0.60
mat512_256.bin 18.62		7.937	~2.35

Tempos de execução para o ficheiro mat512_32.bin com diferentes dimensões:

blockDim.x	blockDim.y	gridDim.x	gridDim.y	Time(s)
32	1	512	1	0.00539
16	2	512	1	0.00550
8	4	512	1	0.00565
4	8	512	1	0.00550
2	16	512	1	0.00563
1	32	512	1	0.00554
32	1	256	2	0.00577
32	1	32	16	0.00545
32	1	16	32	0.00555
32	1	2	256	0.00568
32	1	1	512	0.00558

Conclusões

Gerais:

- De uma forma geral, a principal conclusão é que o desempenho dos programas através do GPU é significativamente maior.
- Ao variar as dimensões do bloco e da grid, não verificámos praticamente nenhuma diferença nos tempos de execução. Isto deve-se ao facto que no contexto destes dois programas, a posição das threads não tem grande influência.
- Comparando os dois programas conseguimos verificar que o programa 2 (em que o algoritmo segue uma abordagem por linhas) possui um desempenho pior do que o programa 1. Isto deve-se ao facto de que como no algoritmo do programa 2 cada thread corresponde a uma linha da matriz, o número de cache misses que acontecem é maior, pelo que a leitura de dados da memoria é realizada com um tempo maior, o que afeta a performance.

Considerando a lei de Amdahl:

- Conseguimos verificar também que de acordo com os valores do speed up calculado entre as versões de CPU e GPU, o
 melhoramento do desempenho é maior quanto maior for a complexidade do problema. Isto porque, nos resultados do
 programa 1, o speed up vai aumentando à medida que o tamanho do ficheiro também vai aumentando.
- Contudo, o mesmo não é tão visível nos resultados do programa 2, e isto deve-se ao facto de que neste programa existe um maior número de cache misses, o que afeta significativamente o desempenho do programa tanto do CPU como GPU. O facto de por vezes o tempo de execução ser pior no GPU, deve-se ao facto de que como a memória cache do GPU é inferior à do CPU, leva a que provavelmente existam mais cache misses no GPU do que no CPU, o que afeta a performance.