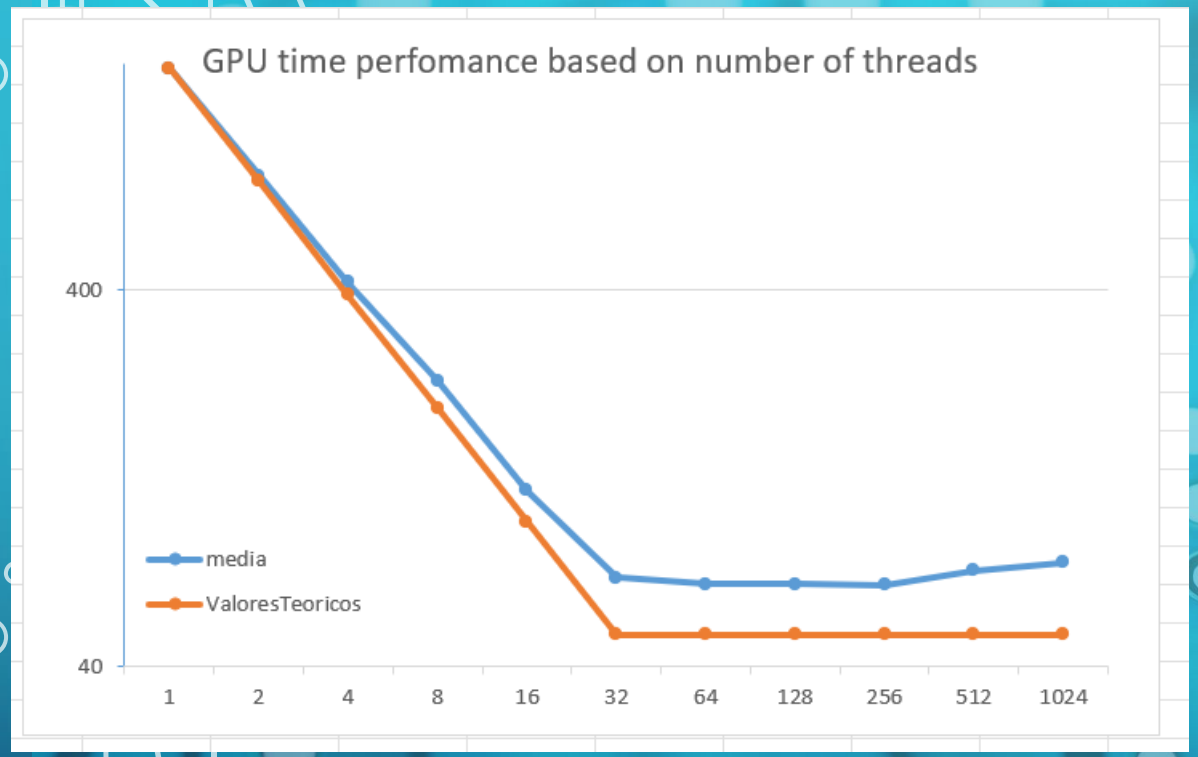
CCC_T3G4

FEITO POR: ALDEBERTO ROSÁRIO (105589)

RUI LAMEIRAS 102817



Knowing this definicion, looking at our grath, we can awnser that we are preforming paralism programing. When we exacute the program the information we be devided in diferente threads and then, those threads, we be handle by de GPU in parallalism. Not only that our grath contitues a good example to preform prgraming in prallism sinse we created to find the best number of threads to exaxute this polacy in in most efficient manuer.

In short, we are taking advantage of parallelism, because in our case all CEs are active. And for this, the BT thread number must be equal to ws. In our case, BT number T is greater than ws and as a result all threads will be split into multiple warps.
Nº T do BT = 256; ws = 32;

blocDimX	blocDimY	gridDimX	T1	T2	T3	T4	T5	media	disvioPadr	Conclusão
8	0	8	65,84	65,83	65,84	65,84	65,83	65,83	0,005477	Melhor x,y
7	1	8	65,88	65,89	65,87	65,88	65,89	65,88	0,008367	
6	2	8	65,95	65,95	65,96	65,96	65,96	65,95	0,005477	
5	3	8	66,2	66,19	66,21	66,2	66,21	66,2	0,008367	
4	4	8	66,51	66,51	66,51	66,29	66,47	66,4	0,095499	
3	5	8	66,84	66,83	66,83	66,81	66,86	66,8	0,018166	
2	6	8	86,29	86,26	86,24	86,24	86,2	86,2	0,032863	
1	7	8	138	146,9	146,9	146,9	137,9	143	4,902244	
0	8	8	246,8	236,1	236,2	226,7	236,8	236	7,114563	

Combinations of blocdimx and blocdimy.

According to our experience, we concluded that the best combination of blocdim is blocdimx = 8 and blocdimy = 0, keeping griddimx = 8. In other words, we maintain the best combination found on the first slide.

gridDimX	gridDimY	blcDX|y	T1	T2	T3	T4	T5	media	disvioPadr	conclusão
8	0	8 0	65,84	65,83	65,84	65,84	65,83	65,83	0,005477	Melhor x,y
7	1	8 0	65,9	65,91	65,88	65,86	65,91	65,8	0,021679	
6	2	8 0	65,9	65,9	65,89	65,87	65,9	65,8	0,013038	
5	3	8 0	66	65,85	65,87	65,87	65,87	65,8	0,060992	
4	4	8 0	65,87	65,88	65,87	65,87	65,89	65,87	0,008944	
3	5	8 0	65,87	65,85	65,86	65,94	65,88	65,8	0,035355	
2	6	8 0	65,87	65,85	66,02	65,88	65,87	65,8	0,069065	
1	7	8 0	65,91	65,9	65,83	65,84	65,89	65,8	0,036469	
0	8	8 0	65,89	65,86	65,89	65,89	65,86	65,8	0,016432	

Combinations of gridDimx and gridDimY.

According to our experience, we concluded that the best combination of gridDim is gridDimX = 8 and gridDimY = 0, keeping blocDimX = 8 and blocDimY = 0. In other words, we maintain the best combination found on the first slide.

The best:

blocDimX = 8; blocDimY = 0; gridDimX = 8; gridDimY = 0. We don't change the Z combinaction.

- In CUDA programming, the optimal number of threads often depends on various factors, including the specific GPU architecture, the problem being solved, and memory limitations. The choice of the number of threads per block (like 256) is influenced by the following considerations:
  1. Warp Size (32);
  2. Resource Limitations: Selecting 256 threads might fit well within these resource constraints, balancing parallelism without exceeding resource limits.
  3. Data Parallelism:
  4. Occupancy;
  5. Memory Access Patterns;
  6. Kernel Complexity;

- Nº T do BT = 256; ws = 32; 256 > 32; Nº Warps = 8. Each Warps with 32 BT. And 4 blocks will be executed concurrently. In total, 32 are executed in parallel, but as we have 8 warps, only 4 are executed in parallel (32/8 = 4).

SPEED UP

- speed_up = avg_CPU_exec_time / (avg_GPU_exec_time + avg_TTH-D + avg_TTD-H);
- avg_GPU_exec_time = 65,83 ms;
- avg_CPU_exec_time = 3106 ms;
- avg_TTH-D = 132 ms;
- avg_TTD-H = 0,12 ms;
- speed_up = 3106/ (65,83 + 132 + 0,12) = 15,69;
- Conclusion:
- We concluded that it is worth using the GPU because the calculated speed up (15,69) is greater than 1ns.
- If this were less than 1, then it would not be viable to use the GPU.

N	T1	T2	T3	T4	T5	media	disvioPadr	gridDimX	
256	3107	3104	3112	3108	3102	3106	3,847077	8	CPU Kenel
256	65,84	65,83	65,84	65,84	65,83	65,83	0,005477	8	GPU Kenel
256	134,9	135,4	133,7	127,8	130,2	132	3,276431	8	TTH-D
256	0,1272	0,1231	0,1251	0,1221	0,1272	0,12	0,002329	8	TTD-H