Programação Paralela e Concorrente
2021/2022
Mestrado em Engenharia Informática

Report - Assignment 3
Floyd-Warshall

Tomás Dias nº57582

Sendo que para a realização deste trabalho foi utilizado CUDA (através de um ambiente Google Collab), foi necessário adaptar o código do programa aos principais conceitos da aplicação.

Os procedimentos como a declaração das variáveis a serem utilizadas durante a execução no GPU bem como o envio do grafo pesado do CPU para o GPU e vice-versa (utilizando-se as funções *cudaMalloc()* e *cudaMemcpy()* para o efeito), foram efetuados na função *floyd_warshall_gpu()*. A computação realizada no GPU foi efetuada no kernel *__global__ fw_gpu_kernel()*.

Em relação ao particionamento do trabalho, foi utilizada uma grelha de uma dimensão com 32 blocos (o máximo aconselhado, de forma a manter o GPU ocupado) e 256 threads por bloco. Os valores destes parâmetros tiveram como base a informação disponibilizada no ficheiro *CUDA_Occupancy_Calculator*, com o objetivo de maximizar a performance do programa. O ajuste dos parâmetros neste ficheiro foi feito a partir das especificações do dispositivo utilizado, apresentadas a quando da execução do ficheiro *deviceQuery*.

Como esperado, foi de notar uma diferença bastante significativa no que diz respeito aos tempos de execução entre a versão sequencial no CPU e a versão em paralelo no GPU, sendo esta última imensamente mais rápida para valores de N (correspondente ao tamanho do grafo). Eis alguns exemplos para diferentes valores de N:

Para a matriz *not_random_graph*:

N = 100: CPU time = 0.005s; GPU time = 0.2016s
N = 2000: CPU time = 32.800s; GPU time = 0.5984s
N = 5000: CPU time = 525.4s; GPU time = 6.069s

Sobre *thread divergence*, o programa não manifestou nenhum comportamento nesse sentido. Uma explicação possível deve-se à estrutura dos dados, permitindo que existam sempre breakpoints lógicos nos limites de *warp* (*warp boundaries*), fazendo com que todas as threads executem o mesmo código simultaneamente.