Faculdade de Ciências da Universidade de Lisboa

Programação Paralela e Concorrente

José Eduardo Madeira
fc51720

Assignment 3

Este relatório foi escrito com base em testes executados no Google colab, numa gpu Tesla K80.

Para podermos executar funções na gpu, temos de criar uma nova função com a keyword __global__, onde a iremos chamar na função floyd_warshall_gpu, com uma sintax special <<<blocks, threadsPerBlocks>>>, que por sua vez ira ser chamada na função main(pelo cpu).

Inicialmente pensei em abordar este problema com ciclos while na função "__global__ void floyd_warshall_kernel", dois whiles encadeados (while (i<graph_size){ (while (j<graph_size){ // do something }}), mas com ciclos while só é vantagoso para um numero pequeno de threads, e não é muito rápido, abordei de uma maneira diferente , nós queremos aproveitar o máximo possível de threadse para tal iremos escolher o número de threads por bloco e quantos blocos baseados no graph_size=2000.

Para alem desta escolha, também iremos organizar melhor os as nossas threads, visto que estamos a trabalhar com matrizes e não arrays iremos organizar as threads em 2D usando a função dim3, para o numero total de blocos iremos usar o 256 como "aconselhado" pelo "Cuda Occupancy Calculator" logo dim3 blocks (16,16) pois 16*16 dá os 256, para a quantidade de threads de cada block pensei em dividir o graph_size pelos 16, resultaria para este graph_size, contudo não funciona com tamanhos não divisíveis por 16, então teria de ser dim3 threads((graph_size+16-1)/16 , (graph_size+16-1)/16).

Com isto verifiquei que o tempo de execução do programa em cpu rondava a média dos 32 segundos, enquanto que a média de tempo da execução do programa em gpu rondou 1,30 segundos, sendo que o valor mais baixo foi 0,71segundos e o mais alto foi 1,97segundos. Não consegui perceber o porque de uma diferença tao grande entre o valor mínimo e o máximo da execução em gpu, mas foi um melhoramento enorme em questão de tempo.

A solução para garantir que todas as iterações no gpu são synchronized, foi sincronizarando os blocks, ao ter usando o for , (for (i = 0; i < graph_size; i++)), outside the threads (fora da função __global__) , porque todas as kernel calls sao synchronized.