Concorrência e Paralelismo 2016-17 - HW 2
Game of Life paralelo

Aluno: Gonçalo Sousa Mendes, n.º 42082, Turno p.3
Prof: João Lourenço
Dept. de Informática
FCT/NOVA
24 de Dezembro de 2016

1 Introdução

O objectivo deste trabalho de casa era paralelizar, com eficiência, a implementação do *game of life* realizada no primeiro trabalho de casa, usando a biblioteca *cilk plus*.

2 Implementação

Neste trabalho era essencial entender as funções dadas pelo *cilk plus* e utilizadas corretamente para paralelizar o *game of life*. A função usada foi o *cilk_for*, que faz paralelização implícita de um ciclo *for*, isto é, avalia o número de processadores disponíveis num computador e tenta dividir um ciclo *for* em partes iguais, dando uma parte a cada processador, caso não seja possível, não é feita a paralelização. Esta função pode ser usada neste trabalho, pois não existem dependências entre as iterações. Para se percorrer as células todas recorre-se a dois ciclos aninhados, um para percorrer as linhas, outro para percorrer as colunas. O local escolhido para o *cilk_for* foi o *loop* de fora, que percorre as linhas. Este pormenor é bastante importante, pois é necessario garantir que se introduz o menor *overhead* possível com a criação das *threads*, se tivéssemos colocado no ciclo de dentro, estaríamos a criar um grande *overhead*, pois, para cada linha, seriam criadas novas *threads*, ou seja, num tabuleiro com 8 linhas e 4 processadores criavam-se 32 *threads*, comparado com às 4 se criam colocando o *cilk_for* no ciclo de fora.
Foram ainda necessárias outras alterações face à implementação sequencial, as variáveis usadas nos *loops* tinham de ser declaradas e inicializadas na declaração do *loop*, pois se fossem declaradas antes tornavam-se partilhadas entre as *threads*, sem que a biblioteca do *cilk plus* consegui-se no final fazer o *join* das variáveis, resultando num tabuleiro errado. As variáveis que eram usadas dentro dos ciclos tiverem de ser declaradas e inicializadas depois dentro dos ciclos, pelo mesmo motivo.

3 Avaliação e conclusão

O primeiro reparo que se faz é que as operações aritméticas envolvidas para cada célula são bastantes simples, pelo que para casos onde o tabuleiro era pequeno, obteve-se o efeito contrário, o *overhead* causado pela inicialização e lançamento das *threads* aumentou bastante o tempo de execução do jogo, pois a cada *tick* do jogo é necessário iniciar estas *threads*. Para se conseguir obter uma melhoria foi necessário usar tabuleiros maiores, na ordem das 256 linhas por 256 colunas (fornecidos por um colega no fórum do grupo da cadeira). Só neste caso se conseguiu registar uma diminuição nos tempos quando comparado com a implementação sequencial.
Neste trabalho foi possível perceber as dificuldades envolvidas na paralelização de um programa, mesmo quando se usa uma biblioteca que realiza todo esse trabalho pelo programador. Questões ligadas à paralelização de um programa nem sempre são triviais de responder, questões como, qual o melhor sítio para se paralelizar e quando é que a paralelização compensa o *overhead* causado pela paralelização, foram algumas das questões que tiveram uma importância vital para se conseguir paralelizar este trabalho e entender o custo de paralelizar um programa. Por último, o contacto com uma biblioteca que realizada a paralelização pelo programador foi bastante importante para se compreender que estas soluções vêm sempre com um *trade-off* associado, assunto que também será abordado no último trabalho de casa.