Instituto Tecnológico de Aeronáutica

CE-265: Processamento Paralelo

Exercício 02: Paralelizar Jogo da Vida no SDumont em OpenMP

Bruno Benjamim Bertucci - Turma 23.2

1 Implementação de paralelismo OpenMP no Jogo da Vida

Um dos componentes principais do Jogo da Vida, implementado sequencialmente no último exercício, é a computação das gerações, que correspondem aos estados de cada elemento de um tabuleiro quadrado de tamanho N.

No processamento de uma geração, cada elemento do tabuleiro é visitado individualmente, o que é feito com dois laços. Uma vez que a atualização do estado de um elemento não depende de nenhum outro estado, pois os estados atualizados são gravados em uma matriz distinta daquela que armazena os estados da geração anterior, pode-se dizer que essa tarefa é paralelizável.

Para realizar essa paralelização, a biblioteca OpenMP foi importada na página que contem a função que avança uma geração. Imediatamente antes do início dos laços, o seguinte comando de pré-processador foi incluso:

```
#pragma omp parallel for private(vizviv, j)
```

A diretiva *parallel* indica a criação de um trecho do código onde haverá paralelismo de execução, enquanto a diretiva *for* efetua a paralelização da execução desse laço. As variáveis *vizviv* e *j* foram inclusas na cláusula *private*, para indicar que cada *thread* deve receber uma cópia separada dessas variáveis, uma vez que elas representam, respectivamente, a quantidade de vizinhos "vivos" para cada elemento do tabuleiro e o contador do segundo laço, enquanto o contador do primeiro laço é privatizado automaticamente pelo OpenMP.

Observa-se que, sem essa privatização das variáveis citadas, haveriam erros no resultado da execução, pois para cada elemento deve ser atribuído um valor diferente a elas.

O comando OpenMP usado implica que, logo após o término do laço externo da função, as *threads* adicionais criadas devem ser destruídas. Em outras palavras, após esse laço, o resto da execução da função é feita de forma sequencial.

2 Teste de correção do Jogo da Vida paralelizado

Para garantir que a implementação paralelizada do Jogo da Vida foi feita sem comprometer a correção dos resultados, um teste inicial foi feito, no qual o jogo foi executado para um tabuleiro de tamanho 6. Esse teste foi feito usando desde um até seis *threads*, e para cada execução, indicou que os resultados estavam corretos em todos eles, demonstrando que a implementação está correta.

O Anexo A mostra a saída produzida pelo executável desse teste para todos os números de *threads* testados.

3 Teste de tempos de execução e análise de *speed-up*

Após garantir a correção da implementação paralelizada do Jogo da Vida, um segundo teste foi executado, no qual tabuleiros de tamanhos variando em potências de 2, desde 8 até 512, foram combinados com quantidades de *threads* variando, também em potências de 2, desde 1 até 16,

totalizando 30 testes distintos. Os tempos de processamento de cada teste, medidos em segundos, encontram-se na Tabela 1.

	Threads				
Tamanho	1	2	4	8	16
8	0.000076	0.000120	0.000175	0.000272	0.081626
16	0.000407	0.000268	0.000197	0.000205	0.210403
32	0.003381	0.001835	0.001079	0.000816	0.472217
64	0.027526	0.014066	0.007314	0.004158	0.994440
128	0.223542	0.112372	0.056720	0.029331	0.347624
256	1.805324	0.904092	0.452200	0.228031	0.118860
512	14.495039	7.256608	3.629822	1.817614	0.920666

Table 1: Tempos de execução para cada número de *threads* e tamanho do tabuleiro

Observando somente os tempos, não é imediata a percepção da diferença de desempenho entre um número de *threads* e outro. Nesse viés, uma métrica comum para medir tal desempenho é o denominado *speed-up*, cuja fórmula está dada na Equação 1, onde p é o número de processadores, $T(1)$ é o tempo de execução com 1 processador, e $T(p)$ é o tempo de execução com p processadores. Os valores de *speed-up* correspondentes aos tempos obtidos encontram-se na Tabela 2.

$$S(p) = \frac{T(1)}{T(P)} \tag{1}$$

	Threads			
Tamanho	2	4	8	16
8	0.633333	0.434286	0.279412	0.000931
16	1.518657	2.065990	1.985366	0.001934
32	1.842507	3.133457	4.143382	0.007160
64	1.956917	3.763467	6.620010	0.027680
128	1.989303	3.941150	7.621356	0.643057
256	1.996837	3.992313	7.917011	15.188659
512	1.997495	3.993320	7.974762	15.744080

Table 2: Speed-up.

Observando-se a tabela de valores de *speed-up*, pode-se extraír algumas conclusões importantes sobre a implementação paralelizada do Jogo da Vida. Primeiramente, é possível observar que o aumento do número de *threads* leva a ganhos maiores de velocidade de execução quanto maior for o volume de processamento da tarefa paralelizada, uma vez que, para um mesmo número de *threads*, o *speed-up* aumenta conforme o tamanho do tabuleiro aumenta.

Além disso, a tabela mostra que, para tarefas menores, ou seja, tamanhos de tabuleiros menores, um número elevado de *threads* prejudica o desempenho, resultando em tempos consideravelmente maiores do que o tempo de processamento com um *thread*. Esse fato é intuitivo, uma vez que, quando não há uma tarefa complexa o suficiente para o número de *threads* usado, o custo computacional adicional gasto para a criação e destruição das *threads*, bem como a divisão do trabalho entre elas, superará os ganhos de tempo gerados pela execução paralela.

Finalmente, é importante notar que, para uma tarefa complexa o suficiente, como é o caso do tabuleiro de tamanho 512, o *speed-up* é capaz de se aproximar bem da quantidade de *threads* usada, o que pode ser verificado até para 16 *threads* nesse caso. Isso indica que a tarefa proposta, de visitar e operar sobre cada elemento de uma matriz de forma independente, se beneficia de forma eficiente do uso de processamento paralelo.

Anexo A – Saídas dos testes de correção do Jogo da Vida paralelizado

1 *Thread*

```
Execucao com 1 threads em 1 processadores
Inicial; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
.X....
..X...
XXX...
......
......
......
=======
Iter 001; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
X.X...
.XX...
.X....
......
......
=======
Iter 002; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
..X...
X.X...
.XX...
......
......
=======
Iter 003; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
.X....
..XX..
.XX...
......
......
=======
Iter 004; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
..X...
...X..
.XXX..
......
......
=======
Iter 005; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
.X.X..
..XX..
```

```
..X...
......
=======
Iter 006; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
...X..
.X.X..
..XX..
......
=======
Iter 007; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
..X...
...XX.
..XX..
......
=======
Iter 008; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
...X..
....X.
..XXX.
......
=======
Iter 009; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
..X.X.
...XX.
...X..
=======
Iter 010; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
....X.
..X.X.
...XX.
=======
Iter 011; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
...X..
....XX
...XX.
=======
```

```
Iter 012; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
....X.
.....X
...XXX
=======
**RESULTADO CORRETO**
```

2 *Threads*

```
Execucao com 2 threads em 2 processadores
Inicial; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
.X....
..X...
XXX...
......
......
......
=======
Iter 001; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
X.X...
.XX...
.X....
......
......
=======
Iter 002; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
..X...
X.X...
.XX...
......
......
=======
Iter 003; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
.X....
..XX..
.XX...
......
......
=======
Iter 004; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
..X...
...X..
.XXX..
......
......
```

```
=======
Iter 005; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
.X.X..
..XX..
..X...
......
=======
Iter 006; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
...X..
.X.X..
..XX..
......
=======
Iter 007; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
..X...
...XX.
..XX..
......
=======
Iter 008; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
...X..
....X.
..XXX.
......
=======
Iter 009; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
..X.X.
...XX.
...X..
=======
Iter 010; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
....X.
..X.X.
...XX.
=======
Iter 011; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
```

```
......
......
......
...X..
....XX
...XX.
=======
Iter 012; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
....X.
.....X
...XXX
=======
**RESULTADO CORRETO**
```

3 *Threads*

```
Execucao com 3 threads em 3 processadores
Inicial; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
.X....
..X...
XXX...
......
......
......
=======
Iter 001; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
X.X...
.XX...
.X....
......
......
=======
Iter 002; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
..X...
X.X...
.XX...
......
......
=======
Iter 003; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
.X....
..XX..
.XX...
......
......
=======
Iter 004; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
```

```
=======
......
..X...
...X..
.XXX..
......
......
=======
Iter 005; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
.X.X..
..XX..
..X...
......
=======
Iter 006; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
...X..
.X.X..
..XX..
......
=======
Iter 007; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
..X...
...XX.
..XX..
......
=======
Iter 008; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
...X..
....X.
..XXX.
......
=======
Iter 009; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
..X.X.
...XX.
...X..
=======
Iter 010; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
```

```
......
....X.
..X.X.
...XX.
=======
Iter 011; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
...X..
....XX
...XX.
=======
Iter 012; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
....X.
.....X
...XXX
=======
**RESULTADO CORRETO**
```

4 *Threads*

```
Execucao com 4 threads em 4 processadores
Inicial; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
.X....
..X...
XXX...
......
......
......
=======
Iter 001; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
X.X...
.XX...
.X....
......
......
=======
Iter 002; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
..X...
X.X...
.XX...
......
......
=======
Iter 003; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
```

```
.X....
..XX..
.XX...
......
......
=======
Iter 004; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
..X...
...X..
.XXX..
......
......
=======
Iter 005; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
.X.X..
..XX..
..X...
......
=======
Iter 006; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
...X..
.X.X..
..XX..
......
=======
Iter 007; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
..X...
...XX.
..XX..
......
=======
Iter 008; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
...X..
....X.
..XXX.
......
=======
Iter 009; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
..X.X.
```

```
...XX.
...X..
=======
Iter 010; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
....X.
..X.X.
...XX.
=======
Iter 011; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
...X..
....XX
...XX.
=======
Iter 012; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
....X.
.....X
...XXX
=======
**RESULTADO CORRETO**
```

5 *Threads*

```
Execucao com 5 threads em 5 processadores
Inicial; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
.X....
..X...
XXX...
......
......
......
=======
Iter 001; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
X.X...
.XX...
.X....
......
......
=======
Iter 002; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
..X...
X.X...
```

```
.XX...
......
......
=======
Iter 003; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
.X....
..XX..
.XX...
......
......
=======
Iter 004; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
..X...
...X..
.XXX..
......
......
=======
Iter 005; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
.X.X..
..XX..
..X...
......
=======
Iter 006; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
...X..
.X.X..
..XX..
......
=======
Iter 007; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
..X...
...XX.
..XX..
......
=======
Iter 008; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
...X..
....X.
..XXX.
......
```

```
=======
Iter 009; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
..X.X.
...XX.
...X..
=======
Iter 010; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
....X.
..X.X.
...XX.
=======
Iter 011; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
...X..
....XX
...XX.
=======
Iter 012; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
....X.
.....X
...XXX
=======
**RESULTADO CORRETO**
```

6 *Threads*

```
Execucao com 6 threads em 6 processadores
Inicial; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
.X....
..X...
XXX...
......
......
......
=======
Iter 001; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
X.X...
.XX...
.X....
......
```

```
......
=======
Iter 002; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
..X...
X.X...
.XX...
......
......
=======
Iter 003; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
.X....
..XX..
.XX...
......
......
=======
Iter 004; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
..X...
...X..
.XXX..
......
......
=======
Iter 005; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
.X.X..
..XX..
..X...
......
=======
Iter 006; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
...X..
.X.X..
..XX..
......
=======
Iter 007; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
..X...
...XX.
..XX..
......
=======
Iter 008; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
```

```
=======
......
......
...X..
....X.
..XXX.
......
=======
Iter 009; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
..X.X.
...XX.
...X..
=======
Iter 010; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
....X.
..X.X.
...XX.
=======
Iter 011; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
...X..
....XX
...XX.
=======
Iter 012; Dump posicoes [1:6, 1:6] de tabuleiro 6 x 6
=======
......
......
......
....X.
.....X
...XXX
=======
**RESULTADO CORRETO**
```