Paralelização do Jogo da Vida com OpenMP

Modificações no código

A principal alteração feita foi a transformação de dois loops for aninhados em um único loop:

```
// Loops aninhados
for(int i = 1; i <= tam; i++) {
    for(int j = 1; j <= tam; j++) {
        // ... codigo usando i e j
    }
}

// Loop unico
for (k=0; k<sqtam; k++) {
    int i=k%tam + 1;
    int j=k/tam + 1;
    // ... mesmo codigo usando i e j
}
```

Como no caso do jogo da vida o código executado dentro do loop não depende de resultados anteriores do mesmo, a união torna possível paralelizar o código em até tam*tam (sqtam) threads, ao contrário do original que só poderia ser divido em tam threads.

Após a união dos loops bastou garantir que nenhuma váriavel pudesse causar conflito entre as diversas threads, especificamente as variaveis i, j e vizviv foram declaradas privadas à thread, garantindo assim, que o loop seja verdadeiramente paralelizável e sem conflitos.

Procedimento UmaVida Paralelizado

```
void UmaVida(int* tabulIn, int* tabulOut, int tam) {
  int i, j;
  int vizviv;
  int k;
  int sqtam = tam*tam;

#pragma omp parallel for private(i, j, vizviv)
  for (k=0; k<sqtam; k++) {
    i=k%tam + 1;
    j=k/tam + 1;
    vizviv =
      tabulIn[ind2d(i-1,j-1)] +
      tabulIn[ind2d(i-1,j  )] +
```

```
      tabulIn[ind2d(i-1,j+1)] +
      tabulIn[ind2d(i  ,j-1)] +
      tabulIn[ind2d(i  ,j+1)] +
      tabulIn[ind2d(i+1,j-1)] +
      tabulIn[ind2d(i+1,j  )] +
      tabulIn[ind2d(i+1,j+1)];
    if (tabulIn[ind2d(i,j)] && vizviv < 2)
      tabulOut[ind2d(i,j)] = 0;
    else if (tabulIn[ind2d(i,j)] && vizviv > 3)
      tabulOut[ind2d(i,j)] = 0;
    else if (!tabulIn[ind2d(i,j)] && vizviv == 3)
      tabulOut[ind2d(i,j)] = 1;
    else
      tabulOut[ind2d(i,j)] = tabulIn[ind2d(i,j)];
  }
}
```

Arquivos de Saída Funciona.c

- OutFunciona_1OMP

```
Execucao com 1 threads em 24 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**
```

- OutFunciona_2OMP

```
Execucao com 2 threads em 12 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**
```

- OutFunciona_3OMP

```
Execucao com 3 threads em 24 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**
```

- OutFunciona_4OMP

```
Execucao com 4 threads em 24 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**
```

- OutFunciona_5OMP

```
Execucao com 5 threads em 24 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**
```

- OutFunciona_6OMP

```
Execucao com 6 threads em 24 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**
```

Tabelas de Tempo de Execução e Speed-Ups

- Tamanho 8

Threads	Tempo (s)	Speed-Up
1	0.00007	-
2	0.000126	0.5555555555555555
4	0.000226	0.30973451327433627
8	0.000339	0.20648967551622416
16	0.000489	0.14314928425357873

- Tamanho 16

Threads	Tempo (s)	Speed-Up
1	0.000542	-
2	0.000376	1.4414893617021276
4	0.000537	1.009310986964618
8	0.00059	0.91864406779661
16	0.000597	0.9078726968174203

- Tamanho 32

Threads	Tempo (s)	Speed-Up
1	0.004657	-
2	0.002873	1.6209537069265574
4	0.003674	1.267555797495917
8	0.003413	1.3644887196015236
16	0.003366	1.3835412953060011

- Tamanho 64

Threads	Tempo (s)	Speed-Up
1	0.038864	-
2	0.021209	1.8324296289311144
4	0.032529	1.1947493006240586
8	0.021967	1.769199253425593
16	0.018912	2.054991539763113

- Tamanho 128

Threads	Tempo (s)	Speed-Up
1	0.317793	-
2	0.162205	1.9592059430966988
4	0.103375	3.0741765417170495
8	0.144506	2.1991682006283475
16	0.077896	4.0797088425593095

- Tamanho 256

Threads	Tempo (s)	Speed-Up
1	2.59461	-
2	1.306746	1.985550367095059
4	0.735442	3.5279600566734013
8	0.386641	6.710643723764422
16	0.222676	11.651951714598788

- Tamanho 512

Threads	Tempo (s)	Speed-Up
1	21.138297	-
2	10.640818	1.9865293250951197
4	5.494318	3.8473013393109032
8	2.867999	7.370399013388777
16	1.61829	13.062119274048534

Análise e Comentários

Do gráfico que o speedup não é uma linha reta, ou seja, existe algum custo envolvido na paralelização, relacionado provavelmente à escrita e leitura em memória compartilhada entre as CPU's (respectivamente às variáveis tabulIn

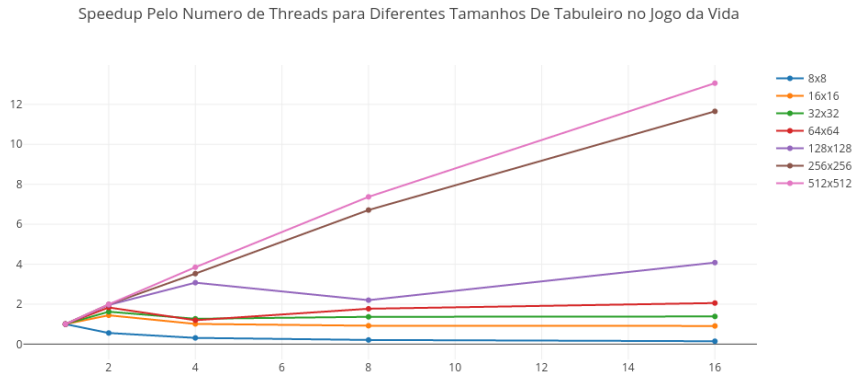

Figure 1: Comparação entre speedup para cada tamanho de tabuleiro

e tabulOut). Para alguns casos o speedup inclusive diminui com o numero de threads (muito notável no caso do tabuleiro 8x8).

Além disso, do exercício pode-se entender a simplicidade de se tornar uma parte do código paralelizada a partir do uso do OpenMP, ao mesmo tempo que existe uma complexidade envolvida em garantir que o código seja paralelizável. Planejar o código para que variaveis 'globais' não entrem em conflito em multiplas threads foi o maior desafio desse exercício.:wa