# Exercício 1: Paralelização do Jogo da Vida com OpenMP

**Nome:** Daniel Araujo Cavassani (COMP 25)

**Data:** 02/09/2024

Durante este exercício, a ideia foi paralelizar a função `UmaVida` do Jogo da Vida usando OpenMP para rodar no supercomputador Santos Dumont. A função originalmente processava o tabuleiro de forma sequencial, mas eu modifiquei o código para que ele pudesse usar várias threads e, assim, acelerar o cálculo.

## Modificação no Código

A modificação foi feita na função `UmaVida`. Adicionei a diretiva `#pragma omp parallel for private(i, j, vizviv) shared(tabulIn, tabulOut)` para paralelizar o loop que percorre o tabuleiro:

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

 #pragma omp parallel for private(i, j, vizviv) shared(tabulIn, tabulOut)
 for (i=1; i<=tam; i++) {
   for (j= 1; j<=tam; j++) {
     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;
```

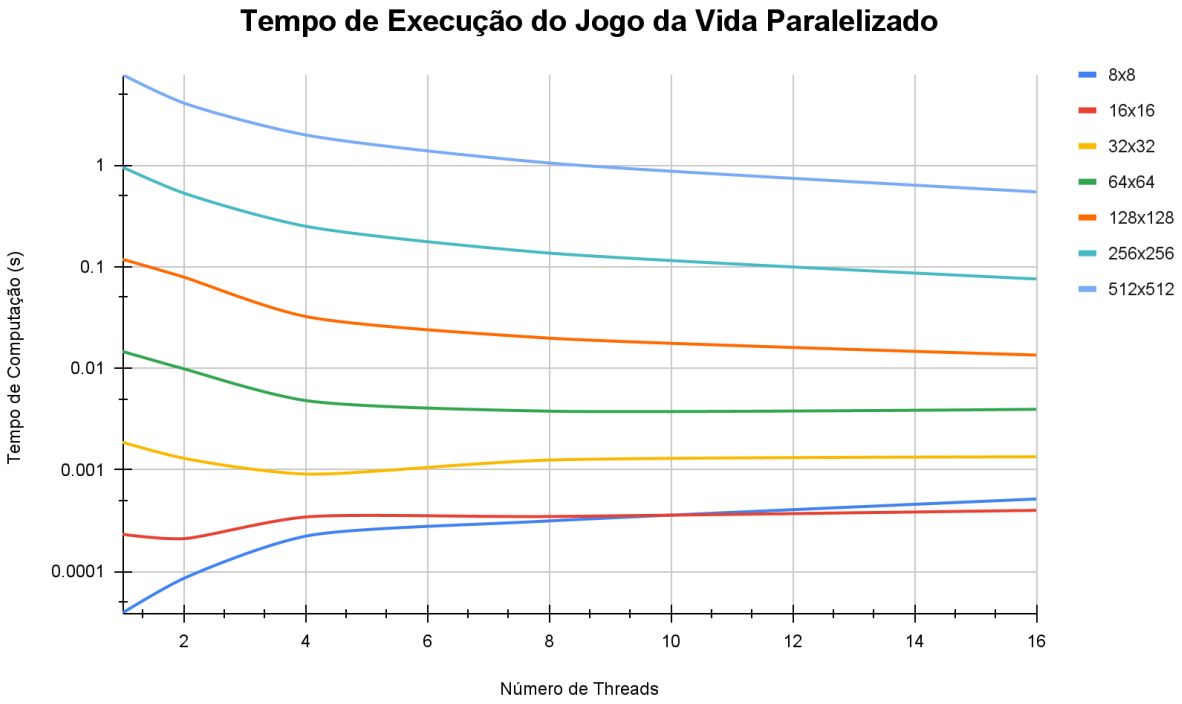
```
    else if (!tabulIn[ind2d(i,j)] && vizviv == 3)
 tabulOut[ind2d(i,j)] = 1;
    else
 tabulOut[ind2d(i,j)] = tabulIn[ind2d(i,j)];
   }
 }
}
```

Essa mudança permite que o loop que processa cada célula do tabuleiro seja executado em paralelo por múltiplas threads, o que deve reduzir o tempo de execução, especialmente para tabuleiros maiores.

## Resultados

Depois de garantir que o código estava correto, rodei as execuções com diferentes números de threads (1, 2, 4, 8, 16) para medir o tempo de computação. Aqui estão os tempos de execução para cada configuração:

| Número de Threads | 8x8 | 16x16 | 32x32 | 64x64 | 128x128 | 256x256 | 512x512 |
|---|---|---|---|---|---|---|---|
| 1 | 0.000039 | 0.000232 | 0.00187 | 0.014624 | 0.118044 | 0.953555 | 7.749303 |
| 2 | 0.000085 | 0.00021 | 0.001301 | 0.009887 | 0.079019 | 0.530404 | 4.087536 |
| 4 | 0.000222 | 0.000343 | 0.000909 | 0.004812 | 0.032337 | 0.249901 | 1.978594 |
| 8 | 0.000314 | 0.000347 | 0.001247 | 0.003777 | 0.019775 | 0.135931 | 1.046656 |
| 16 | 0.000516 | 0.000399 | 0.001343 | 0.003939 | 0.013477 | 0.075549 | 0.544217 |




## Cálculo do Speed-up

O speed-up foi calculado dividindo o tempo de execução com 1 thread pelo tempo de execução com múltiplas threads. Isso nos dá uma medida de quanto a paralelização melhorou a eficiência do programa.

| Número de Threads | 8x8 | 16x16 | 32x32 | 64x64 | 128x128 | 256x256 | 512x512 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 0.46 | 1.1 | 1.44 | 1.48 | 1.49 | 1.8 | 1.9 |
| 4 | 0.18 | 0.68 | 2.06 | 3.04 | 3.65 | 3.82 | 3.92 |
| 8 | 0.12 | 0.67 | 1.5 | 3.87 | 5.97 | 7.02 | 7.4 |
| 16 | 0.08 | 0.58 | 1.39 | 3.71 | 8.76 | 12.62 | 14.24 |

## Análise

Os resultados mostram que, em geral, o tempo de computação diminui à medida que o número de threads aumenta, especialmente para tabuleiros maiores. No entanto, o ganho de

desempenho não é linear, e o speed-up varia significativamente dependendo do tamanho do tabuleiro e da quantidade de threads utilizadas.

Em alguns casos, especialmente com tabuleiros menores, o overhead de gerenciamento das threads e a comunicação entre elas se sobrepõem aos ganhos esperados com o paralelismo. Isso resulta em um speed-up menor que 1, indicando que o tempo de execução com mais threads foi, na verdade, maior do que com uma única thread. Esse fenômeno demonstra que, para problemas de menor escala, o custo adicional de gerenciamento do paralelismo pode superar os benefícios, levando a uma piora no desempenho.

Essa observação é crucial, pois destaca que o paralelismo nem sempre traz melhorias, especialmente quando o problema não é suficientemente grande para justificar a divisão do trabalho entre várias threads. Para problemas maiores, o paralelismo mostrou-se eficaz, mas é essencial considerar o impacto do overhead ao decidir quantas threads utilizar.

## Conclusão

A paralelização com OpenMP trouxe ganhos significativos em tabuleiros maiores, mostrando que o uso de múltiplas threads pode acelerar o Jogo da Vida. No entanto, para tabuleiros menores, o overhead do paralelismo às vezes superou os benefícios, resultando em um desempenho pior do que o sequencial. Esses resultados mostram que o paralelismo é eficaz, mas deve ser aplicado com cuidado, especialmente em problemas menores, para evitar que o overhead prejudique o desempenho.