
# Jogo da Vida

*Programação Multithread*

Programação Concorrente e Distribuída
Prof. Álvaro Fazenda

Daiana K. S. Santos 120357
Isadora Muniz 120431

São José dos Campos
Dezembro de 2020

# Objetivo

Implementar o Jogo da Vida utilizando programação concorrente em três versões: PThreads e OpenMP utilizando linguagem C/C++ e JavaThreads em Java.

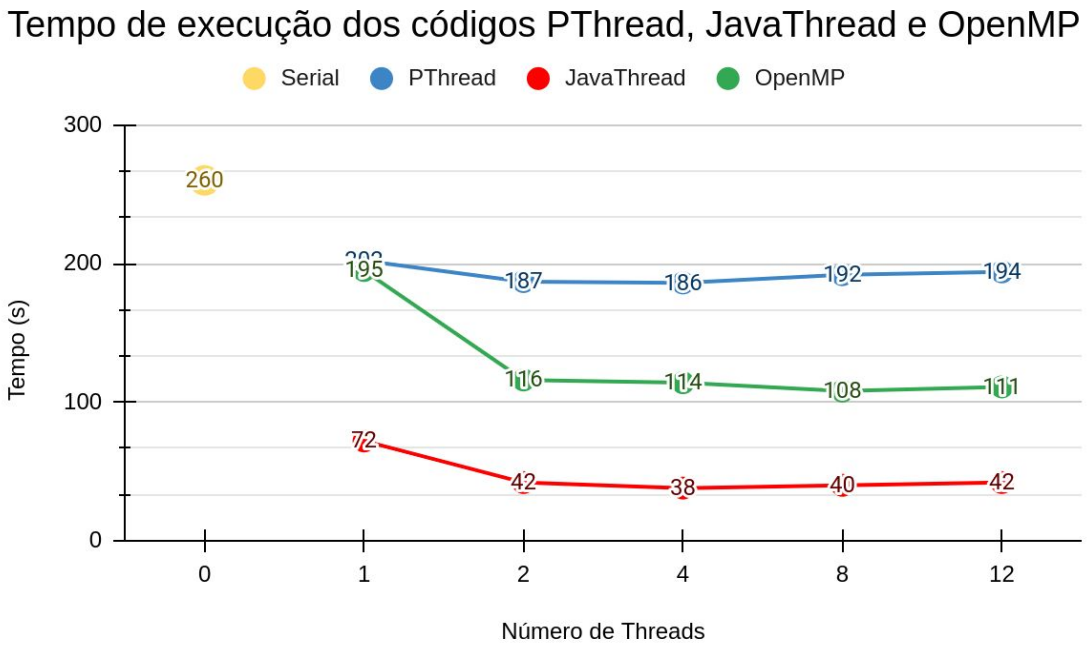
## Especificações da máquina

Para a execução dos códigos, realizou-se em um laptop HP 246G6, com processador Intel Core I5-7200U, de 4 núcleos, 2,5GHz, e 16GB de RAM.

## Execução dos códigos

Desenvolveram-se os programas para cada versão utilizando programação concorrente, em que se realizou testes usando os seguintes números de threads: 1, 2, 4, 8. A seguir serão apresentados os valores obtidos durante a realização desses testes. Para a realização das medidas de tempo, utilizou-se a função gettimeofday(), da biblioteca <sys/time.h>, nos programas desenvolvidos em linguagem C.

| N threads | Serial | PThread | JavaThread | OpenMP |
|---|---|---|---|---|
| 0 | 260 | - | - | - |
| 1 | - | 202 | 72 | 195 |
| 2 | - | 187 | 42 | 116 |
| 4 | - | 186 | 38 | 114 |
| 8 | - | 192 | 40 | 108 |
| 12 | - | 194 | 42 | 111 |



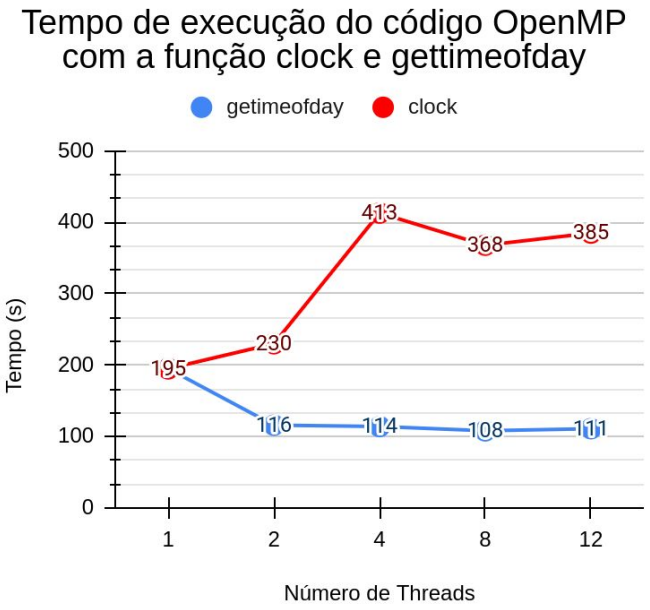

Para a execução dos testes em Java, os resultados dos tempos de desempenho se mostraram de certa forma condizente, em que ao utilizar apenas 1 thread o comportamento do programa é menos eficiente, quando comparado às execuções que utilizam duas ou mais threads.

Assim como esperado com o JavaThreads, obteve-se para OpenMP. Os resultados do seu desempenho de maior eficiência é bastante notório ao comparar sua execução com 1 thread e com 2 threads. Para maiores quantidades de threads, não observou-se grandes diferenças em tempo de execução.

Por outro lado, a execução dos testes com PThreads, o desempenho de execução com diferentes quantidades de threads se manteve muito aproximado uns dos outros. Sendo assim, a implementação de paralelismo com PThreads obteve menor eficiência, quando comparado aos anteriores.

Outra medida de desempenho realizada, foi utilizando a função de tempo, sendo clock(), da biblioteca <time.h> e gettimeofday(), da biblioteca <sys/time.h> no código do OpenMP. A diferença entre as duas funções é que a clock() mostra o tempo gasto pelas CPUs no processo e a gettimeofday() é como se fosse um cronômetro, mostrando o tempo real em que foi executado o código.

| N threads | clock OpenMP | gettimeofday OpenMP |
|---|---|---|
| 1 | 195 | 195 |
| 2 | 230 | 116 |
| 4 | 413 | 114 |
| 8 | 368 | 108 |
| 12 | 385 | 111 |




Assim, é possível observar que o tempo de execução do clock() fica maior que do gettimeofday() a partir de 2 threads, pois ele soma o tempo de execução das threads, resultando em tempos maiores.

Outro detalhe observado é que tanto no código serial e nos concorrentes obtiveram o mesmo número de vivos inicial e após as 2000° gerações.