Programação Concorrete no Jogo da Vida

Relatório de Desempenho de Threading

Luiz Otávio de Medeiros Portella Passos
Departamento de Ciência e Tecnologia
Universidade Federal de São Paulo
São José dos Campos, Brasil
luizopassos33@gmail.com

***Abstract*—Esse documento é um relatório de performance do uso de um diferente número de theads para a resolução do mesmo problema, programado em C e Java;**

I. Introdução

O Jogo da Vida, criado por John H. Conway, utiliza um autômato celular para simular gerações sucessivas de uma sociedade de organismos vivos. É composto por um tabuleiro bi-dimensional, infinito em qualquer direção, de células quadradas idênticas. Cada célula tem exatamente oito células vizinhas (todas as células que compartilham, com a célula original, uma aresta ou um vértice). Cada célula está em um de dois estados: viva ou morta (correspondentes aos valores 1 ou 0). Uma geração da sociedade é representada pelo conjunto dos estados das células do tabuleiro. Sociedades evoluem de uma geração para a próxima aplicando simultaneamente, a todas as células do tabuleiro, regras que estabelecem o próximo estado de cada célula. As regras são:

A. Células vivas com menos de 2 (dois) vizinhas vivas morrem por abandono

B. Cada célula viva com 2 (dois) ou 3 (três) vizinhos deve permanecer viva para a próxima geração

C. Cada célula viva com 4 (quatro) ou mais vizinhos morre por superpopulação.

D. Cada célula morta com exatamente 3 (três) vizinhos deve se tornar viva.

II. Sobre o Experimento

Como visto anteriormente duas versões serão programadas, uma em C utilizando PThreads e outra em Java utilizando JavaThreads. Será criado uma matriz de tamanho 2048x2048 onde cada posição se refere a uma célula, logo poderá apenas assumir 1 ou 0. Tais posições serão geradas aleatoriamente onde na primeira geração se esperam 2096241 células vivas e após 2000 novas gerações espera-se 146951 células vivas. Para comparação dos resultados os programas serão executadas com 1, 2, 4 e 8 threads, o tempo de execução será medido em milisegundos e depois comvertido para segundos, vale ressaltar que 1 thread diz respeito ao próprio processo. Serão calcudados 2 tempos, um para o processo todo e outro apenas para a parte de execução das threads, que é o laço de criação de sucessivas gerações.

III. Metodologia

Para criar uma nova geração é necessário percorrer toda matriz verificando cada célula e se essa está dentro das condições para permanecer viva, morrer ou ressuscitar. Com o auxílio das threads pretende-se dividir o trabalho de percorrer a matriz melhorando assim o desempenho, essa divisão é feita baseada no número de threads. Enquanto 1 thread percorrerá a matriz desde a linha 0 até a linha 2047, ao usar 2 threads uma percorrerá da linha 0 até a 1023 e a outra da linha 1024 até 2047 e assim sucessivamente conforme aumentada a concorrência.

IV. Especificações da Máquina

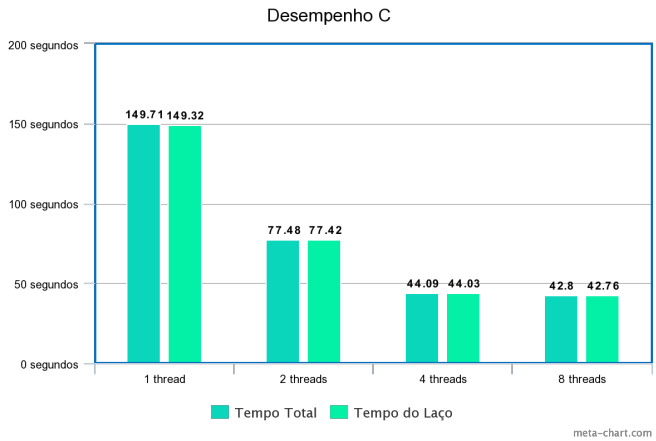
A. Processador: i7 4770k

1) *4 núcleos verdadeiros:*
2) *8 threads:*
3) *Frequência Clock: 3.5 GHz:*
4) *Memória Cache: 8 MB:*

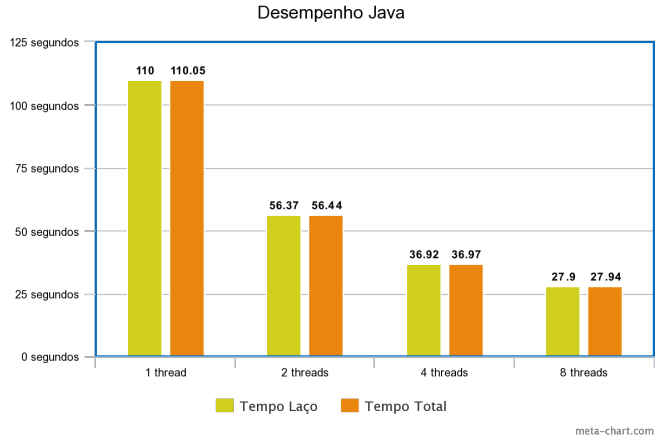
B. Memória RAM: 8 Gb

V. Resultados

A. C



B. *Java*



VI. Conclusão

Percebe-se em ambos os programas que o tempo quase diminiu pela metade, com excessão do resultado de 4 para 8 threads, isso se deve ao fato do processador em que esse experimento foi executado ter apenas 4 núcleos. O tempo não é exatamente diminuido pela metade uma vez que a parte de alocação das matrizes quanta a contagem final das células vivas é feita de forma serial. Nota-se também que o programa em Java rodou mais rápido que o em C.