André Luis Gonçalves Bien - RA: 111829
Andrew Medeiros de Campos - RA: 111775
Washington Holanda de Oliveira - RA: 112268

Atividade de Programação 1 - Jogo da Vida

Introdução

Nesse projeto foi implementado o sistema de Jogo da Vida, criado por John H. Conway para desenvolver melhor o entendimento sobre programação paralela. Foi proposto o desenvolvimento de códigos sequenciais e paralelos, utilizando as linguagens C e Java e suas rotinas de paralelização (bibliotecas OpenMP e PThreads para C e interface Runnable para Java).

O tabuleiro do jogo contava com uma matriz de 2048x2048 posições onde estavam presentes os indivíduos e foram feitas 2000 gerações.

Desenvolvimento

Para maior fidelidade de medição, os tempos de execução foram aferidos utilizando a mesma máquina e com nenhum outro aplicativo rodando concorrentemente. As configurações do computador utilizado são:

- Processador Intel i5
  - 8ª Geração
  - 4 Núcleos
  - 8 Processadores Lógicos
- Memória RAM DDR3
  - 8 GB

Inicialmente foram desenvolvidos os códigos seriais nas linguagens C e Java, que funcionariam como base para a construção das implementações em que seriam usados os métodos de paralelismo fornecidos. Esses códigos ajudaram a desenvolver uma ideia dos lugares nos códigos em que poderiam ser usados os métodos de programação paralela, de modo que não interferisse nos resultados e trouxesse algum ganho em desempenho e velocidade de execução dos programas.

Cada método usado gerou suas dificuldades e facilidades nos momentos de adaptação ao código original. Devido a essas diferenças, alguns métodos tornaram possível a paralelização de mais partes do código, trazendo, talvez, um pouco mais de eficiência que os outros. A partir do momento em que soube-se como trabalhar com cada um deles, a implementação tornou-se mais fácil, sendo possível alcançar os resultados desejados.

Os códigos paralelos foram rodados utilizando 2, 4 e 8 threads simultâneas e os resultados de medição e desempenho feitos se encontram na seção abaixo.

Resultados

Após a execução de todos os códigos, os resultados foram anotados e se encontram abaixo. Todas as medições de tempo da tabela foram feitas em segundos.

Configurações da Máquina: Intel (R) Core (TM) i5-8265U CPU @ 1.60GHz-1.80GHZ com 4 núcleos físicos, 8 processadores lógicos e 8GB de memória RAM												
	OpenMP				Pthread				JavaThread			
	1 Thread	**2 Threads**	**4 Threads**	**8 Threads**	**1 Thread**	**2 Threads**	**4 Threads**	**8 Threads**	**1 Thread**	**2 Threads**	**4 Threads**	**8 Threads**
Laço Gerações (s)	172,298	94,270	56,115	55,722	179,290	104,724	81,830	68,424	96,390	60,183	39,354	35,677
Tempo Total (s)	172,402	94,356	56,180	55,789	179,388	104,821	81,933	68,525	96,424	60,217	39,390	35,711

Como é possível observar, existe um significativo ganho de desempenho entre as diferentes situações e o código sequencial (1 thread), porém à partir do momento em que o número de threads supera o número de núcleos físicos o ganho para de ser tão acentuado, uma vez que essas threads excedentes passam a ser processadas concorrentemente porém sem paralelismo, fazendo assim com que não haja ganho de tempo processamento com processamento simultâneo.

Outro ponto que é importante ser ressaltado é que dentre os métodos feitos na linguagem C, o OpenMP foi mais eficiente (mesmo que com pouco diferente) porém não supera o método feito em java, mesmo contendo uma porcentagem de código paralelizado maior.