Relatório de Execução

Testes de execução:

Foram realizados testes de execução com o objetivo de aferir, entre outras coisas, o desempenho das soluções sequencial e concorrente para diferentes casos de teste. Foram dadas como entrada para o algoritmo matrizes de números inteiros aleatórios pertencentes ao intervalo [-50, 50], de diferentes dimensões. Estas matrizes foram geradas automaticamente através da execução de um pequeno programa implementado pela dupla (segue anexo junto aos demais códigos fonte).

Todos os testes foram realizados em uma máquina com processador Pentium 4 3,00 GHz de 2 núcleos. Ao todo, foram efetuados 26 testes comparativos, em que, para uma mesma matriz de entrada foram executadas as versões sequencial e concorrente do problema. Abaixo são listados alguns dos casos de teste e seus resultados correspondentes.

Casos de Teste:

1. *Matriz 8x8*

    a. *Solução sequencial:*

    Soma: 85
    Maior: 49
    Menor: -50
    Tempo de processamento: 2 ms

    b. *Solução concorrente:*

    Soma: 85
    Maior: 49
    Menor: -50
    Tempo de processamento: 159 ms

    c. *Conclusão:* Versão *sequencial* foi mais eficiente.

2. *Matriz 24x24*

    a. *Solução sequencial:*

    Soma: 590
    Maior: 49
    Menor: -50
    Tempo de processamento: 9 ms

    b. *Solução concorrente:*

Soma: 590
Maior: 49
Menor: -50
Tempo de processamento: 1110 ms

c. *Conclusão:* Versão *sequencial* foi mais eficiente.

3. *Matriz 40x40*

   a. *Solução sequencial:*

      Soma: -662
      Maior: 49
      Menor: -50
      Tempo de processamento: 22 ms

   b. *Solução concorrente:*

      Soma: -662
      Maior: 49
      Menor: -50
      Tempo de processamento: 2477 ms

   c. *Conclusão:* Versão *sequencial* foi mais eficiente.

4. *Matriz 80x80*

   a. *Solução sequencial:*

      Soma: -2660
      Maior: 49
      Menor: -50
      Tempo de processamento: 82 ms

   b. *Solução concorrente:*

      Soma: -2660
      Maior: 49
      Menor: -50
      Tempo de processamento: 12244 ms

   c. *Conclusão:* Versão *sequencial* foi mais eficiente.

5. *Matriz 160x160*

a. *Solução sequencial:*

   Soma: -5767
   Maior: 49
   Menor: -50
   Tempo de processamento: 321 ms

b. *Solução concorrente:*

   Soma: -5767
   Maior: 49
   Menor: -50
   Tempo de processamento: 54053 ms

c. *Conclusão:* Versão *sequencial* foi mais eficiente.

6. *Matriz 800x800*

   a. *Solução sequencial:*

      Soma: -319264
      Maior: 49
      Menor: -50
      Tempo de processamento: 8047 ms

   b. *Solução concorrente:*

      Soma: -319264
      Maior: 49
      Menor: -50
      Tempo de processamento: 1.11225e+06 ms

   c. *Conclusão:* Versão *sequencial* foi mais eficiente.

7. *Matriz 1280x1280*

   a. *Solução sequencial:*

      Soma: -846361
      Maior: 49
      Menor: -50
      Tempo de processamento: 20554 ms

   b. *Solução concorrente:*

Soma: -846361
Maior: 49
Menor: -50
Tempo de processamento: 2.69355e+06 ms

c. *Conclusão:* Versão ***sequencial*** foi mais eficiente.

8. *Matriz 1440x1440*

a. *Solução sequencial:*

Soma: -1046686
Maior: 49
Menor: -50
Tempo de processamento: 26098 ms

b. *Solução concorrente:*

ERRO: pthread_create()

c. *Conclusão:* Versão ***sequencial*** foi mais eficiente.

Conclusão Final:

Em todos os testes efetuados, a versão sequencial se mostrou a mais eficiente, fato que nos surpreendeu, pois esperávamos que para grandes dimensões a versão concorrente levasse vantagem sobre a sequencial.

Este fato pode ser explicado pelo grande número de threads que é criado por conta da proposta de solução. Por mais que a carga de processamento seja bem distribuída entre as threads, o custo para criação de um número alto de threads, acrescido a muitas trocas de contexto geradas por conta da concorrência de uma CPU que não é a ideal para este tipo de aplicação, acaba sendo superior ao possível ganho obtido através da distribuição do processamento.