Fundamentos de Sistemas Operacionais

Trabalho 02 Processos & Threads

Iago Rodrigues Gonçalves - 13/0010219 Jonathan Henrique Maia de Moraes - 12/0122553

Universidade de Brasília - Faculdade Un
B $\operatorname{Gama} 11/09/2016$

1 Alarme de Relógio

1.1 Ambiente de Desenvolvimento

• Sistema Operacional: Linux Mint Sarah (17.3)

• Editor de Texto: Sublime Text 2

• Compilador: gcc 4.8.4

• Flags do Compilador: -Wall -g

1.2 Instruções

• Diretório: question 01

• Comandos do Makefile:

- make: compila o código, gerando o executável de nome q01

- make dist-clean: remove o executável de nome q01

• Comando de Execução: bin/q01

• Operações: Nenhuma. Após a chamada da execução do programa, o mesmo irá esperar durante 5 segundos e depois disparar uma mensagem informando que o alarme foi disparado.

2 Máximo de uma Sequência

2.1 Ambiente de Desenvolvimento

• Sistema Operacional: Linux Mint Sarah (17.3)

• Editor de Texto: Sublime Text 2

• Compilador: gcc 4.8.4

• Flags do Compilador: -Wall -g - lpthread

2.2 Instruções

• **Diretório**: question 02

• Comandos do Makefile:

- make: compila o código main.c, gerando o executável de nome q02
- make sequential: compila o código sequential.c, gerando o executável de nome sequential_q02
- make dist-clean: remove o executável de nome q02 e sequential_q02
- Comando de Execução: bin/q02 e bin/sequential_q02
- Operações:
 - Entrada do vetor de inteiros: Como parâmetro do programa, deverá ser enviado o número n que indique a quantidade total de inteiros, e em seguida os n números que irão preencher o vetor V.
 - Sequencial: caso essa seja a versão sequencial do programa, ele já irá realizar a programação e mostrar os resultados. Caso contrário, ele continuará nos passos a seguir.
 - Preenchimento do Vetor W: São disparadas então n threads que irão preencher o vetor W de n posições com o número 1 em todas elas.

- Comparação dos números: Em seguida são disparadas $\frac{n(n-1)}{2}$ threads que irão fazer as comparações. A thread $T_{(i,j)}$ irá comparar os números v_i e v_j e escrever 0 na posição do vetor W correspodente, isto é, se $v_i < v_j$, $W_i = 0$, e caso contrário $W_j = 0$.
- Verificação do maior número: Na terceira etapa são disparadas n threads que irão ler o vetor W. Caso a thread identifique o valor 0, ela irá se fechar automaticamente. Caso seja encontrado o valor 1 ela imprime a posição e o valor do maior número.

2.3 Respostas das Questões de Análise

No roteiro é perguntado porque são utilizadas $\frac{n(n-1)}{2}$ threads e não n^2 . Isso se deve ao fato de que se fosse n^2 as threads estariam comparando a própria posição com ela mesma, além de repetir combinações de números já realizadas por outras threads. Assim, devemos analisar o fato de que devemos realizar as combinações de N elementos sem repetição dessas combinações, que é definido pela fórmula: $\frac{n!}{p!(n-p)!}$. No nosso contexto n é a quantidade de números entrados pelo usuário, e p é igual a 2, pois estamos combinando de 2 em 2. Assim ficamos:

$$\frac{n!}{2!(n-2)!} = \frac{n(n-1)(n-2)!}{2(n-2)!} \\
= \frac{n(n-1)}{2} \tag{1}$$

3 Multiplicação de Matrizes

3.1 Ambiente de Desenvolvimento

• Sistema Operacional: Debian Jessie (8.5)

• Editor de Texto: Atom 1.0.19

• Compilador: gcc 4.9.2

• Flags do Compilador: -O2 -Wall -lpthread

3.2 Instruções

• **Diretório**: question_03

• Comandos do Makefile:

- make: compila o código q03a.c, gerando o executável de nome q03
- make a: compila o código q03a.c, gerando o executável de nome q03
- make b: compila o código q03b.c, gerando o executável de nome q03
- \mathbf{make} \mathbf{c} : compila o código q03c.c, gerando o executável de nome q03
- make tc1: executa o q03 com a entrada tc1.in, que consiste em uma matriz $A_{(2,3)}$ e outra matriz $B_{(3,2)}$. O valor esperado de saída é uma matriz $R_{(2,2)}$, além do tempo utilizado para realizar a operação (em nanossegundos ns). A operação pode ser vista a seguir:

$$A = \begin{bmatrix} 10 & 20 & 30 \\ 40 & 50 & 60 \end{bmatrix}$$

$$B = \begin{bmatrix} 10 & 20 \\ 30 & 40 \\ 50 & 60 \end{bmatrix}$$

$$R = A \times B = \begin{bmatrix} 2200 & 2800 \\ 4900 & 6400 \end{bmatrix}$$
(2)

- **make tc2**: executa o q03 com a entrada tc2.in, que consiste em uma matriz $A_{(5,2)}$ e outra matriz $B_{(2,3)}$. O valor esperado de saída é uma matriz $R_{(5,3)}$, além do tempo utilizado para realizar a operação (em nanossegundos - ns). A operação pode ser vista a seguir:

$$A = \begin{bmatrix} 1 & 2 \\ 2 & 4 \\ 4 & 8 \\ 8 & 16 \\ 16 & 32 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 3 & 6 \\ 2 & 6 & 9 \end{bmatrix}$$

$$R = A \times B = \begin{bmatrix} 5 & 15 & 24 \\ 10 & 30 & 48 \\ 20 & 60 & 96 \\ 40 & 120 & 192 \\ 80 & 240 & 384 \end{bmatrix}$$
(3)

- **make tc3**: executa o **q03** com a entrada **tc3.in**, que consiste em uma matriz $A_{(10,5)}$ e outra matriz $B_{(5,8)}$. O valor esperado de saída é uma matriz $R_{(10,8)}$, além do tempo utilizado para realizar a operação (em nanossegundos - ns). A operação pode ser vista a

seguir:

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 0 \\ 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 0 \\ 0 & 9 & 8 & 7 & 6 \\ 5 & 4 & 3 & 2 & 1 \\ 0 & 9 & 8 & 7 & 6 \\ 5 & 4 & 3 & 2 & 1 \\ 0 & 1 & 9 & 2 & 8 \\ 3 & 7 & 4 & 6 & 5 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 8 & 1 & 7 & 2 & 6 & 3 & 5 & 4 \\ 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}$$

$$R = A \times B$$

$$E = \begin{bmatrix} 78 & 57 & 75 & 60 & 72 & 63 & 69 & 66 \\ 198 & 132 & 170 & 120 & 142 & 108 & 114 & 96 \\ 78 & 57 & 75 & 60 & 72 & 63 & 69 & 66 \\ 198 & 132 & 170 & 120 & 142 & 108 & 114 & 96 \\ 198 & 132 & 170 & 120 & 142 & 108 & 114 & 96 \\ 198 & 132 & 170 & 120 & 142 & 108 & 114 & 96 \\ 78 & 57 & 75 & 60 & 72 & 63 & 69 & 66 \\ 198 & 132 & 170 & 120 & 142 & 108 & 114 & 96 \\ 78 & 57 & 75 & 60 & 72 & 63 & 69 & 66 \\ 198 & 132 & 170 & 120 & 142 & 108 & 114 & 96 \\ 78 & 57 & 75 & 60 & 72 & 63 & 69 & 66 \\ 104 & 46 & 105 & 65 & 106 & 84 & 107 & 103 \\ 144 & 111 & 130 & 105 & 116 & 99 & 102 & 93 \end{bmatrix}$$

- make clean: remove o executável de nome q03
- Comando de Execução: ./q03
- Operações:
 - Entrada de duas Matrizes: Existente nas três versões de código, a entrada é recebida da seguinte forma:
 - 1. Dois números inteiros devem ser fornecidos. Tais números são utilizados para definir o tamanho da primeira matriz, $A_{(n,m)}$.

- O primeiro valor é a quantidade de linhas (n) e o segundo valor é a quantidade de colunas (m) da matriz A;
- 2. Com o tamanho da matriz A definido, devem ser fornecidos $n \times m$ números inteiros. Tais números são os valores contidos na matriz A;
- 3. Com a matriz A definida, dois outros números inteiros devem ser fornecidos. Tais números são utilizados para definir o tamanho da segunda matriz, $B_{(p,q)}$. O primeiro valor é a quantidade de linhas (p) e o segundo valor é a quantidade de colunas (q) da matriz B;
- 4. Com o tamanho da matriz B definido, devem ser fornecidos $p \times q$ números inteiros. Tais números são os valores contidos na matriz B.
- Cálculo do Tempo Utilizado para Executar o Algoritmo de Multiplicação de Matrizes: Existente nas três versões de código. É registrado o tempo inicial logo antes do primeiro cálculo da célula da matriz resultante. Após o término do cálculo da última matriz resultante, é registrado a diferença do tempo atual pelo tempo inicial.
- Multiplicação de Matrizes com Algoritmo Sequencial: Existente na versão q03a, o algoritmo de multiplicação é realizado sequencialmente, utilizando apenas de estruturas de repetição para calcular cada célula da matriz resultante.
- Multiplicação de Matrizes com Algoritmo Concorrente, Sem Restrição do Número Total de threads: Existente na versão q03b, o algoritmo de multiplicação é realizado concorrentemente, utilizando de um número total de threads equivalente ao número total de células da matriz resultante.
- Multiplicação de Matrizes com Algoritmo Concorrente, Com Restrição do Número Total de threads: Existente na versão q03c, o algoritmo de multiplicação é realizado concorrentemente, utilizando de quantitades necessárias de lotes de threads, cada lote possui um número máximo de threads equivalente ao número total de processadores no sistema computacional em execução.
- Informação de Resultados: Existente nas três versões de có-

digo. É informado o tempo utilizado para executar o algoritmo de multiplicação de matrizes em nanossegundos e a matriz resultante. Será informado Invalid Input. caso a quantitade de colunas da primeira matriz seja diferente da quantidade de linhas da segunda matriz. Entradas de valores inesperados provocarão erros.

3.3 Resposta da Questão de Análise

A versão q03a (sequencial) possui o tempo mais curto de execução dos cálculos da célula da matriz resultante, com tempos na ordem de grandeza de 10^2 ns para os casos de teste fornecidos. Em seguida, a versão q03b (concorrente, limitado pela quantitade de processadores no sistema computacional) apresenta tempos na ordem de grandeza de 10^4 à 10^6 ns. Por fim, a versão q03c (concorrente, não limitado) apresenta tempos na mesma ordem de grandeza que a versão q03b, de 10^4 à 10^6 ns, porém, seus valores apresentados são, em média, 14% maiores que a versão q03b quando o número total de células da matriz resultante é superior ao número total de processadores no sistema computacional. As figuras a seguir descrevem os resultados de três execuções de cada versão de código em cada o caso de teste fornecido pelo Makefile.

Figura 1: Resultados da versão q03a com o caso de teste tc1.

Figura 2: Resultados da versão q03b com o caso de teste tc1.

Figura 3: Resultados da versão q03c com o caso de teste tc1.

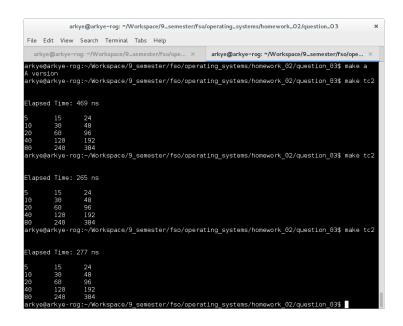


Figura 4: Resultados da versão q03a com o caso de teste tc2.

Figura 5: Resultados da versão q03b com o caso de teste tc2.

Figura 6: Resultados da versão q03c com o caso de teste tc2.

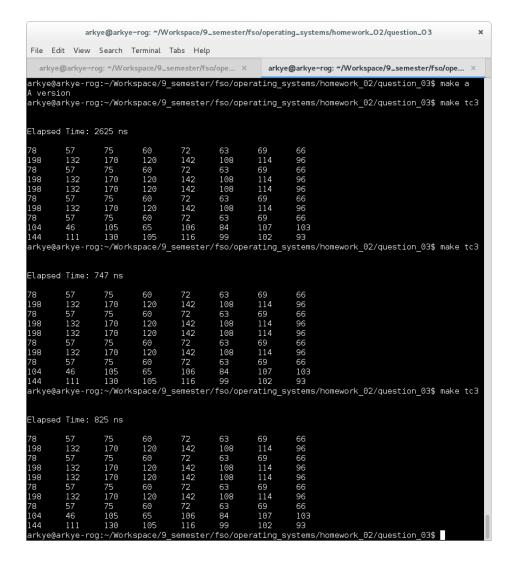


Figura 7: Resultados da versão q03a com o caso de teste tc3.

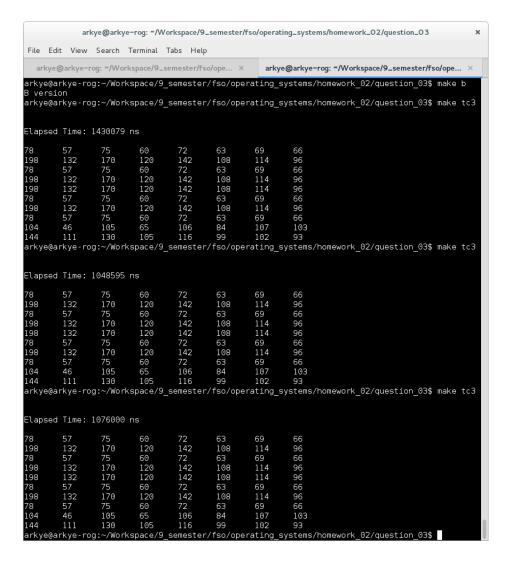


Figura 8: Resultados da versão q03b com o caso de teste tc3.

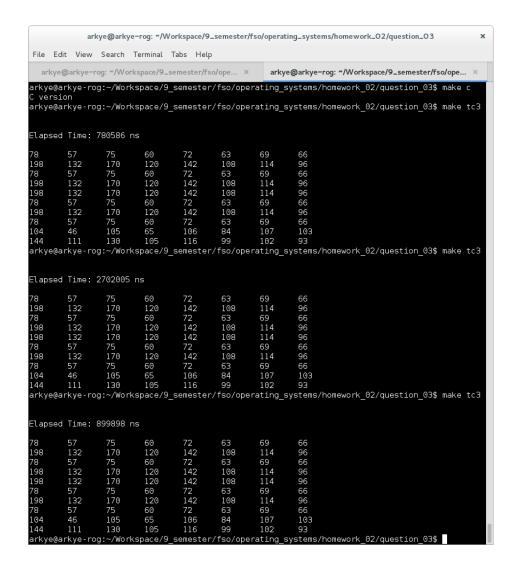


Figura 9: Resultados da versão q03c com o caso de teste tc3.