Universidade Estadual de Campinas – UNICAMP
Faculdade de Tecnologia – FT

T_TT005B_2020S2 - Tópicos Especiais em Telecomunicações I

(Programação de Alto Desempenho)

Grupo: Gáta moustáki
Paloma Eduarda Salvador de Mello - RA 260610
Henrique Arakaki Fonseca - RA 236540

Laboratório 01 - OpenMP

1. Enunciado

Dadas três matrizes, A_{yxw}, B_{wxv}, C_{vx1}, calcule a matriz D_{yx1} tal que $D_{yx1} = (A_{yxw}\ x\ B_{wxv})\ x\ C_{vx1}$. Além disso, calcule a redução pela soma dos elementos na matriz D_{yx1}, isto é, a soma de todos os elementos em D_{yx1}.

As dimensões das matrizes são estabelecidas pelas variáveis y, w, v, cujos valores deverão ser informados pela linha de comando do programa. Além disso, os dados que compõem as matrizes A_{yxw}, B_{wxv}, C_{vx1} devem ser números reais, com até duas casas decimais, que variam entre -10 e +10, gerados aleatoriamente.

Esses dados devem estar dispostos sequencialmente em cada arquivo texto, isto é, o arquivo deve ter um valor real em cada linha, sem espaços antes ou depois de cada número. Os nomes dos arquivos que contêm os dados também deverão ser informados pela linha de comando.

A matriz resultante da computação, isto é, a matriz D_{yx1}, deve estar em um arquivo texto, cujos dados também devem estar dispostos sequencialmente, com duas casas decimais. O nome desse arquivo também deve ser especificado na linha de comando. Já a redução pela soma dos elementos na matriz D_{yx1} deve ser mostrado na tela. Esse valor deve ser exibido sem nenhuma string nem antes, nem depois do número.

2. Resolução

O problema explicado no enunciado acima foi resolvido utilizando a biblioteca OpenMP, usando 4 núcleos de processamento com auxílio de threads, e compilado e executado na instância virtual da Amazon Web Services (utilizou-se o MobaXterm para se conectar a essa instância). Para facilitar sua compilação e evitar erros, seu compilador (*gcc -fopenmp prog.c -o prog*) foi salvo no arquivo "compilador.sh", que encontra-se salvo no repositório criado pelo grupo no GitHub *https://github.com/Shokitex/-Labs-Hight-Performance-Programming* juntamente com o arquivo prog.c, que contém o código fonte e uma cópia deste PDF.

Para compilar e executar o programa, deve-se baixar os arquivos compilador.sh e prog.c para o computador. No prompt de comando, após selecionar a pasta onde os arquivos foram salvos deve-se digitar a seguintes linhas de comando:

chmod +x compilador.sh

./compilador.sh

Em seguida pode-se executar o programa seguindo a sintaxe:

./prog y w v arqA.dat arqB.dat arqC.dat arqD.dat

Na imagem abaixo, pode-se observar o processo de compilação e execução do programa, assim como explicado anteriormente, bem como suas saídas. Fez-se os seguintes testes:

y = 10, w = 10 e v = 10

y = 100, w =100 e v =100

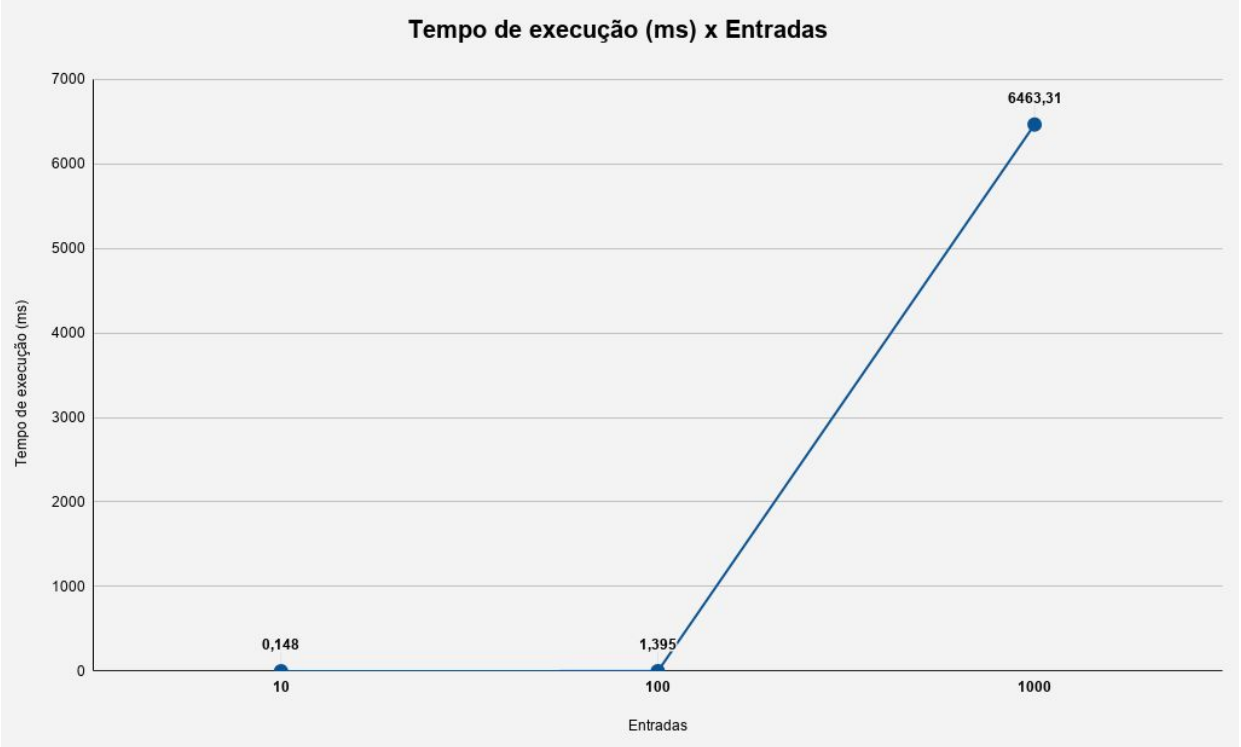
v = 1000, w = 1000 e v = 1000

```
[h236540@ip-172-31-46-145 ~]$ chmod +x compilador.sh
[h236540@ip-172-31-46-145 ~]$ ./compilador.sh
[h236540@ip-172-31-46-145 ~]$ ./prog 10 10 10 arqA.dat arqB.dat arqC.dat arqD.dat
Tempo gasto: 0.148 ms.
-4574.70
 [h236540@ip-172-31-46-145 ~]$ ./prog 100 100 100 arqA.dat arqB.dat arqC.dat arqD.dat
Tempo gasto: 1.395 ms.
-250979.50
 [h236540@ip-172-31-46-145 ~]$ ./prog 1000 1000 1000 arqA.dat arqB.dat arqC.dat arqD.dat
Tempo gasto: 6463.31 ms.
2178035.00
 [h236540@ip-172-31-46-145 ~]$
```

Na saída do programa tem-se a redução da matriz D, gerada por (matriz A x matriz B) x Matriz C, e os tempos de execução em ms (milissegundo) para cada valor de y, w e v, descontadas as operações de I/O.

Abaixo há o gráfico de *valor de variáveis x tempo*:

	Entradas	Tempo de execução (ms)
y, w, v	10	0,148
y, w, v	100	1,395
y, w, v	1000	6463,31



Para a compilação e execução do programa usando os arquivos de números testes, disponibilizados no GitHub, deve-se comentar a linha 82 `createArquivo(argc,argv,y,w,v);` do código fonte, pois essa é responsável pela chamada da função *createArquivo()*, a qual sorteia números aleatórios e grava-os nos arquivos A, B e C.

3. Conclusões

Foi possível notar que o tempo de execução aumenta proporcionalmente às dimensões das matrizes, ou seja, quando aumenta-se as dimensões, o tempo também de execução (especificamente execução dos cálculos) também aumenta.

O uso de threads em conjunto com um ambiente que oferece uma ótima capacidade de processamento e a biblioteca OpenMP fez com que o cálculo - que por sua vez seria demorado devido às dimensões das matrizes se não houvesse threads e o uso da biblioteca OpenMP - fosse feito rapidamente. Portanto, o uso de threads aliado com o OpenMP otimizou o tempo de processamento para cálculos de multiplicação entre matrizes e reduções de matrizes.