Aluno: *Sérgio Luciano de Oliveira Soares*

RA: *263560*

Faculdade de Tecnologia

Relatório dos exercícios solicitados da Matéria de Processamento de Alto Desempenho FT077A P

Prof. André Leon S. Gradvohl, Dr.

Exercício

Crie um programa serial e um programa paralelo com PThreads que calcule a operação matricial D = A * B + C, onde todas as matrizes (A, B, C e D) têm dimensões n x n. Prepare-se para gerar gráficos para cada um dos itens a seguir.

Otimizações utilizadas para o cálculo:
D = A*B+C = C+A*B

soma de matrizes: $s_{ij} = a_{ij} + b_{ij}$
multiplicação de matrizes: $m_{ik} = a_{i1} . b_{1k} + a_{i2} . b_{2k} + ... + a_{in} . b_{nk}$

Para o cálculo do resultado D (=A*B+C), temos a fórmula para cada elemento:
$D_{ik} = C_{i1} + A_{i1} . B_{1k} + A_{i2} . B_{2k} + ... + A_{in} . B_{nk}$

A matriz foi alocada em uma única etapa (como um vetor).
O acesso aos elementos para o loop acima (1..n) causa o acesso aos elementos em B de forma não sequencial na memória.
Como o acesso a posições próximas de memória são mais interessantes, foi utilizada a abordagem de calcular a matriz transposta de B, de forma que o loop cause o acesso aos elementos de B de forma sequencial, assim como em A.
Bt = transposta de B
Com isso, temos:

$D_{ik} = C_{i1} + A_{i1} . B_{k1} + A_{i2} . B_{k2} + ... + A_{in} . B_{kn}$

Essa abordagem permitiu que o cálculo fosse feito sem que fossem criadas condições de corrida, com apenas a necessidade de uma barreira.
Se tivesse sido calculado A*B para a matriz inteira e depois calculado (A*B)+C, haveria necessidade da criação das threads duas vezes (uma para a multiplicação, e outra para a adição), com o uso de uma barreira para cada etapa.

O código todo foi feito em um único arquivo (exerc1.c).
Para compilação:
gcc -pthread exerc1.c -o exerc1

A validação do algoritmo foi feita com n=10, com 2 threads, pegando as matrizes produzidas pelo programa e refazendo o cálculo no excel, com o resultado do programa ficando exatamente igual ao cálculo executado no excel.

Calcule o tempo de execução do programa serial para matrizes de tamanho n x n, onde n = 10, 100 e 1000.
Calcule o tempo de execução do programa paralelo para matrizes de tamanho n x n, onde n = 10, 100 e 1000, cada uma com 2, 4, 8, 16 e 32 threads.

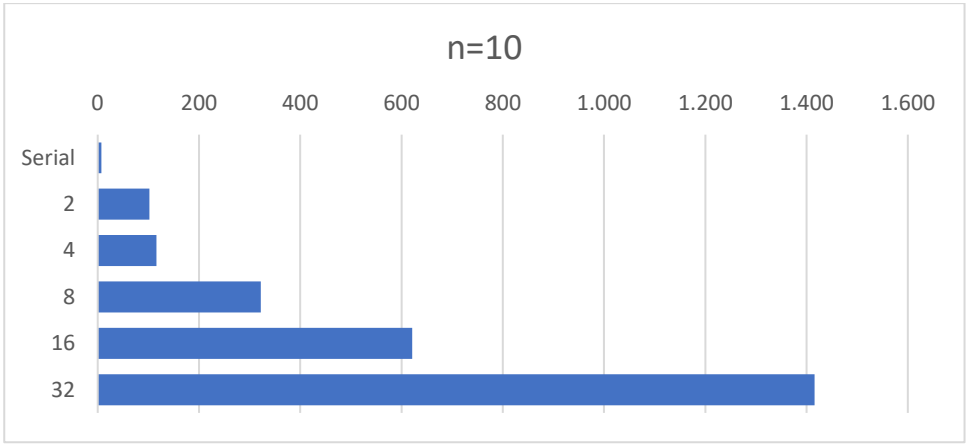
Tempos de execução em microssegundos:

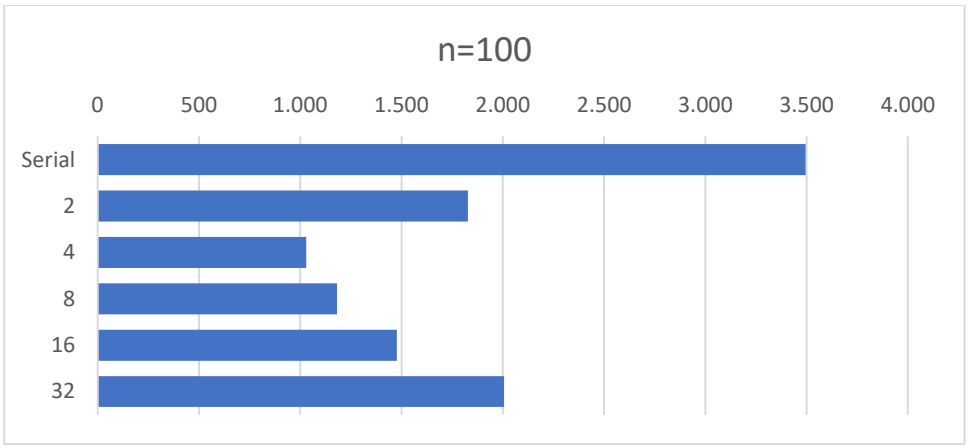
	Threads					
n	Serial	2	4	8	16	32
10	7	102	116	322	621	1.416
100	3.496	1.828	1.030	1.181	1.477	2.007
1000	3.384.285	1.699.999	861.204	864.175	862.453	869.223

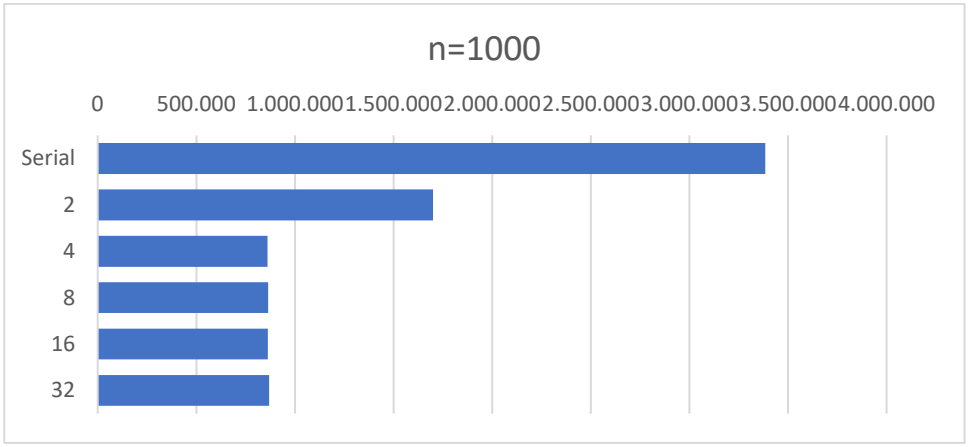
Aluno: *Sérgio Luciano de Oliveira Soares*

RA: *263560*

UNICAMP Faculdade de Tecnologia







Responda as perguntas a seguir no relatório:

Há necessidade de sincronização entre as threads para resolver as operações?
Não. Do modo como o algoritmo foi implementado, não existem condições de corrida. Apenas foi necessária a barreira para aguardar a execução de todas as threads.

Aluno: *Sérgio Luciano de Oliveira Soares*

RA: *263560*

Faculdade de Tecnologia

Qual foi o speedup em relação ao programa serial em cada uma das execuções?
Speedup:

	Threads				
n	2	4	8	16	32
10	0,07	0,06	0,02	0,01	0,005
100	1,91	3,39	2,96	2,37	1,74
1000	1,99	3,93	3,92	3,92	3,89

Houve algum caso em que não houve speedup em relação ao programa serial? Se houve, qual a razão para isso?
Sim.
Para n=10, não houve speedup (speedup<1) em nenhuma situação.
A razão é que a quantidade de processamento para o cálculo do resultado é menor que o overhead de criação e controle das tarefas.
Para n=100, esse overhead tornou-se maior a partir do uso de 8 threads, onde o tempo total de processamento passou a crescer com o aumento do número de threads.
Para n=1000, o speedup se manteve constante a partir de 4 threads, provavelmente por ter atingido o limite de threads reais que o host podia executar simultaneamente.