Relatório do Trabalho de Sistemas Operacionais

Professor André Leon S. Gradvohl
Danilo Barcellos Corrêa 193740
Samuel Rodrigues Da Silva 249564

Introdução

Threads são processos executados pelo processador; é uma sequência de instruções. Na programação *multithread*, há a utilização de mais de uma thread, a fim de se obter um melhor desempenho e de se alcançar determinado objetivo de maneira mais rápida quando comparada à programação comum.

No trabalho em questão, a implementação de conceitos da programação citada se dá na linguagem C, resolvendo o problema proposto.

O problema proposto consiste na implementação de processamento de tarefas utilizando-se threads. Os passos seguidos do enunciado são: 1) ler duas matrizes (A e B), utilizando duas threads; 2) somar as matrizes A e B, sendo essa nova matriz a matriz D (o número de threads varia, incluindo 1, 2 e 4); 3/4) gravar a matriz D e ler a matriz C, com apenas um thread para cada; 5) multiplicar as matrizes C e D, resultando na matriz E (o número de threads varia, incluindo 1, 2 e 4); 6/7) gravar a matriz E e calcular a redução dessa mesma matriz (para a última tarefa, o número de threads varia, incluindo 1, 2 e 4).

Código-fonte

Link para acesso ao repositório no GitHub: DaniloBarcellos/TrabalhoSO: Trabalho de Sistemas Operacionais 2024 | Multithread (github.com).

Além disso, o vídeo para a explicação do código-fonte e a exibição de sua execução: https://youtube.com/@user-lw1vy6rx8f?feature=shared.

Instruções de Compilação

Para a compilação correta do programa é necessário que no mesmo diretório do arquivo .c esteja 3 arquivos .dat contendo matrizes quadradas. Os arquivos devem ter os respectivos nomes “arqA.dat”, ”arqB.dat” e “arqC.dat”.

Para criação do executável será necessário passar no terminal, estando no mesmo diretório do arquivo, a seguinte linha de comando “gcc -o main main.c -l pthread”. onde “-0 main” para o nome do executável, “main.c” para o arquivo fonte e “-l pthread” para referenciar e usar a biblioteca pthread.

Por fim só passa para o terminal a seguinte linha de comando “./main T n arqA.dat arqB.dat arqC.dat arqD.dat arqE.dat”. “n” é a dimensão da matriz quadrada, testada neste experimento com os valores 100 e 1000, e “T” para o número de threads desejados na compilação, testado neste experimento com os valores 1, 2 e 4.

Todas linhas de comando:

```
gcc -o main  main.c -l pthread
./main T n arqA.dat arqB.dat arqC.dat arqD.dat arqE.dat
```

Experimentos

Os experimentos realizados a seguir consideram 6 cenários, nos quais há duas variáveis principais: o número de threads de processamento T e o tamanho das matrizes quadradas n. Os cenários são: Cenário 1) $T = 1$ e $n = 100$; Cenário 2) $T = 1$ e $n = 1000$; Cenário 3) $T = 2$ e $n = 100$; Cenário 4) $T = 2$ e $n = 1000$; Cenário 5) $T = 4$ e $n = 100$; Cenário 6) $T = 4$ e $n = 1000$.

Espera-se que, a medida em que o número de threads aumenta, o tempo total de processamento, considerando um valor de n fixo, diminua, justamente pela divisão das tarefas entre as diferentes threads. A seguir, os resultados obtidos são expressos em cada cenário, e a discussão e conclusão sobre o experimento encontram-se no capítulo Conclusão.

Cenário 1

```
samuel@DESKTOP-U7DTG2V:~/dan$ ./main 1 100 arqA.dat arqB.dat arqC.dat arqD.dat arqE.dat
Redução: 2000000
Tempo soma: 0.0000870000 segundos.
Tempo multiplicação: 0.0097140000 segundos.
Tempo redução: 0.0000490000 segundos.
Tempo total: 0.0177500000 segundos.
```

Figura 1: cenário 1

Cenário 2

```
samuel@DESKTOP-U7DTG2V:~/dan$ ./main 2 100 arqA.dat arqB.dat arqC.dat arqD.dat arqE.dat
Redução: 2000000
Tempo soma: 0.0002110000 segundos.
Tempo multiplicação: 0.0053750000 segundos.
Tempo redução: 0.0002350000 segundos.
Tempo total: 0.0104060000 segundos.
```

Figura 2: cenário 2

Cenário 3

```
samuel@DESKTOP-U7DTG2V:~/dan$ ./main 4 100 arqA.dat arqB.dat arqC.dat arqD.dat arqE.dat
Redução: 2000000
Tempo soma: 0.0004270000 segundos.
Tempo multiplicação: 0.0004560000 segundos.
Tempo redução: 0.0005000000 segundos.
Tempo total: 0.0056440000 segundos.
```

Figura 3: cenário 3

Cenário 4

```
samuel@DESKTOP-U7DTG2V:~/dan$ ./main 1 1000 arqA.dat arqB.dat arqC.dat arqD.dat arqE.dat
Redução: 2000000000
Tempo soma: 0.0083110000 segundos.
Tempo multiplicação: 11.6745740000 segundos.
Tempo redução: 0.0050650000 segundos.
Tempo total: 12.5797900000 segundos.
```

Figura 4: cenário 4

Cenário 5

```
samuel@DESKTOP-U7DTG2V:~/dan$ ./main 2 1000 arqA.dat arqB.dat arqC.dat arqD.dat arqE.dat
Redução: 2000000000
Tempo soma: 0.0051180000 segundos.
Tempo multiplicação: 12.1786160000 segundos.
Tempo redução: 0.0036540000 segundos.
Tempo total: 12.8958570000 segundos.
```

Figura 5: cenário 5

Cenário 6

```
samuel@DESKTOP-U7DTG2V:~/dan$ ./main 4 1000 arqA.dat arqB.dat arqC.dat arqD.dat arqE.dat
Redução: 2000000000
Tempo soma: 0.0333850000 segundos.
Tempo multiplicação: 13.5619580000 segundos.
Tempo redução: 0.0004690000 segundos.
Tempo total: 14.2678970000 segundos.
```

Figura 6: cenário 6

Diante dos cenários expostos, obtém-se o gráfico de barras a seguir com a somatória dos tempos de processamento resultantes, quando $n = 100$.

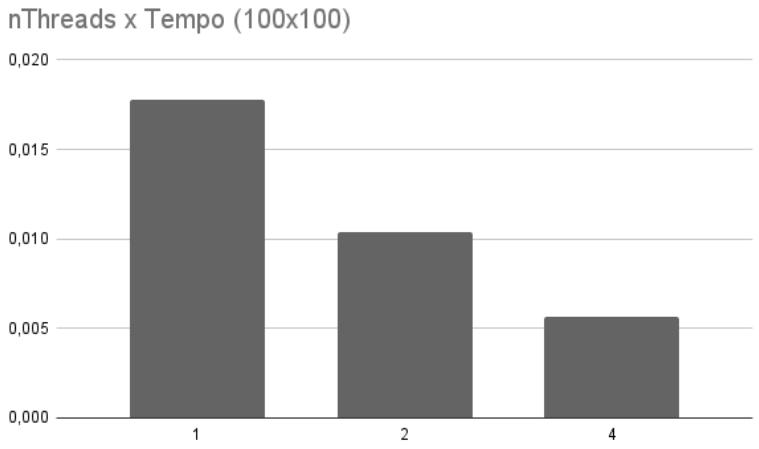

Figura 7: nThreads x Tempo para matrizes quadradas de tamanho 100.

Obtém-se, também, o gráfico de barras para o seguinte caso $n = 1000$.

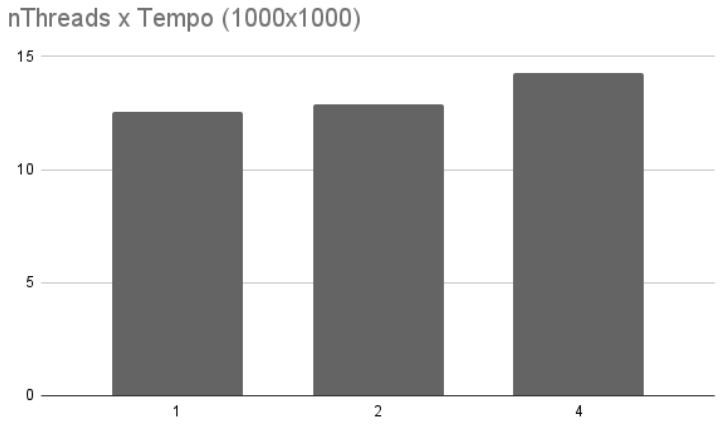

Figura 8: nThreads x Tempo para matrizes quadradas de tamanho 1000.

Conclusão

Observa-se nitidamente que há uma grande disparidade entre os tempos de processamento quando $n = 1000$ e $n = 100$. No primeiro caso, a matriz possui 10^6 elementos, enquanto a segunda possui 10^4; a diferença é de 100 vezes, enquanto que a diferença entre os maiores tempos é de quase 1000 vezes.

A **figura 7** exibe o resultado esperado para todos os testes, ao evidenciar que o maior número de threads reduz o tempo de processamento; entretanto, a **figura 8** apresenta um resultado diferente do esperado, podendo haver uma possível competição entre os threads ou uma distribuição que poderia ser mais otimizada. No caso de matrizes 100x100, a programação multithread, com 4 threads, mostrou-se muito mais eficiente, reduzindo o tempo de processamento pela metade, aproximadamente. Para matrizes de 1000x1000, o aumento do número de threads resulta em um aumento no tempo total de execução do programa. Isso pode ser consequência do gerenciamento de threads pelo sistema operacional, incluindo mudanças de contexto, alocação de CPU, bloqueios e esperas ativas devido à sincronização de acesso a recursos compartilhados pelas threads. A sobrecarga dessas tarefas acaba por aumentar o tempo total de execução do programa conforme o aumento de threads.