Universidade do Vale do Itajaí
Curso de Graduação em Sistemas Para Internet
Disciplina: Sistemas Operacionais
Professor: Felipe Viel

AIRTON BORGES DE SOUZA

Avaliação 01 – Threads e Paralelismo

Itajaí, SC
2º período; 19/09/2023

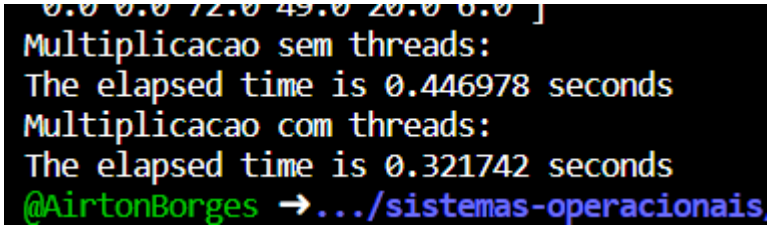
Implementação feita em C, usando Pthreads, e o time.h para realizar as medições de tempo de execução.

Implementação da Multiplicação de matrizes

Ao analisar os resultados, é possível notar uma diferença clara no processamento, porém, ao diminuir e aumentar o número de threads, o resultado se torna mais ou menos eficiente. Isso se dá pela implementação do pthreads na linguagem selecionada, o c. Sem carga suficiente, o pthreads acaba executando as threads sequencialmente, e não é possível ver grandes ganhos de desempenho.

Os resultados são os listados a seguir:

- Maior tempo de execução em single thread, usando números aleatórios de 1 a 9, em uma matriz 30x30, loggando no console a matriz a cada iteração 0.446978 segundos
- Maior tempo de execução em duas threads, usando números aleatórios de 1 a 9, em uma matriz 30x30, loggando no console a matriz a cada iteração. Onde cada thread é responsável por metade da matriz: 0.321742 segundos



Foi possível notar que as duas threads conseguiram trabalhar em paralelo, para multiplicar os vetores.

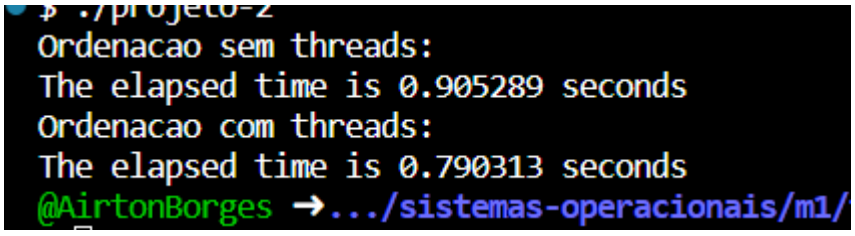
Implementação do Bubble Sort

Porém, o Bubble Sort, acaba sendo um caso onde não é possível ganhar muita performance ao implementar multithreading, principalmente pelo fato, de que as threads vão organizar metade da lista, porém, ao juntar as duas listas, é necessário reordenar para obter o resultado esperado. Salvo neste caso, onde metade da lista estará ordenada corretamente, e o if de checagem de ordem não precisará executar ao passar para juntar a lista, já que a mesma foi ordenada por uma thread anteriormente.

Os resultados são os listados a seguir

- Maior tempo de execução em single thread, ordenado de maneira descendente, em um vetor de 18000 posições: 0.905289 segundos

- Maior tempo de execução em duas threads, ordenado de maneira descendente, em um vetor de 18000 posições: 0.790313 seconds



Reiterando, o vetor está ordenado de maneira decrescente, então o if da validação de trocar as posições, não executou algumas vezes, devido a ordenação na primeira thread, logo, pode-se notar um ganho de desempenho.