Disciplina: INF16179 - Sistemas Distribuídos
Alunos: Thiago da Silva Meireles de Souza / Marcio Merçon de Vargas

Laboratório I

Metodologia

Este trabalho realiza uma comparação da tarefa de ordenação de um vetor com uso do método selection sort, com execução em 2, 4, 8 e 16 threads. A implementação final foi realizada em C com o uso exclusivo de threads por motivos que serão apresentados adiante. A execução do teste foi realizada com vetores de tamanhos variando de (1-10)*10^5 e a coleta do tempo de execução foi realizada por meio do comando time do linux que permite a execução de outra tarefa retornado informações sobre os recursos utilizados. A execução foi automatizada com uso de um script e os resultados salvos em um arquivo. Por fim os dados obtidos foram analisados com auxílio do google colab.

Problemas encontrados

Durante o desenvolvimento desta atividade alguns problemas foram encontrados. A primeira versão desenvolvida foi implementada em python com uso de threads para o método de ordenação e processos para a concatenação do vetor. Problemas de deadlock surgiram com vetores de tamanho acima de 20000 na troca de mensagens entre os processos. A segunda alternativa tomada foi efetuar apenas o uso de threads, o programa passou a operar sem problemas, porém a execução, embora em multitarefa não executou em paralelo. A terceira alternativa foi refazer o código em C, com fork e pipe para a ordenação do vetor e thread para a concatenação, de maneira semelhante ao observado no python, com um número de elementos acima de 20000 o pipe não era capaz de efetuar a troca de mensagens, contudo agora os processos estavam sendo executados em paralelo. A última alternativa foi a execução dos processos fazendo uso de exclusivamente threads, desta maneira o programa passou a se comportar como deveria.

Resultados obtidos

As figuras 1 e 2 abaixo apresentam os resultados obtidos após a execução do programa, de imediato já é notório a significativa redução no tempo de execução das tarefas à medida que o número de threads aumentou. A execução do programa obteve o resultado esperado, a medida que o problema foi dividido em mais partes a serem executadas em paralelo, o tempo de execução foi reduzido. Um dos fatores que contribui para esse fenômeno é a granularidade grossa desse problema, no qual ele pode ser facilmente dividido de forma equilibrada entre as partes, além disso não há necessidade de constantes sincronizações e troca de mensagens durante sua execução, o que permite uma redução significativa no tempo de execução. E isso pode ser observado na figura 2 onde dobrar o número de processos traz praticamente a redução do tempo de execução na metade, a

particularidade ocorre no aumento de 2 para 4 processos, onde a redução do tempo de execução foi de quase 4 vezes.

Outro ponto importante também se dá pela complexidade do algoritmo selection sort que é $O(n^2)$, reduzir o *n* em cada thread também reduzirá significativamente o número de operações de $O(n^2)$ $O((n/p)^2)$.

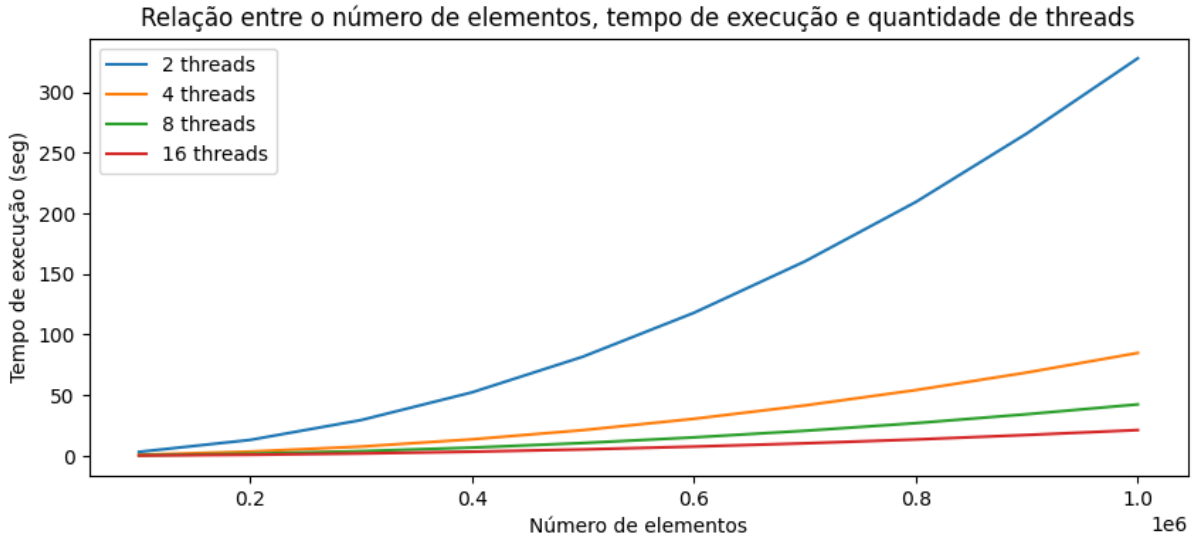

Figura 1

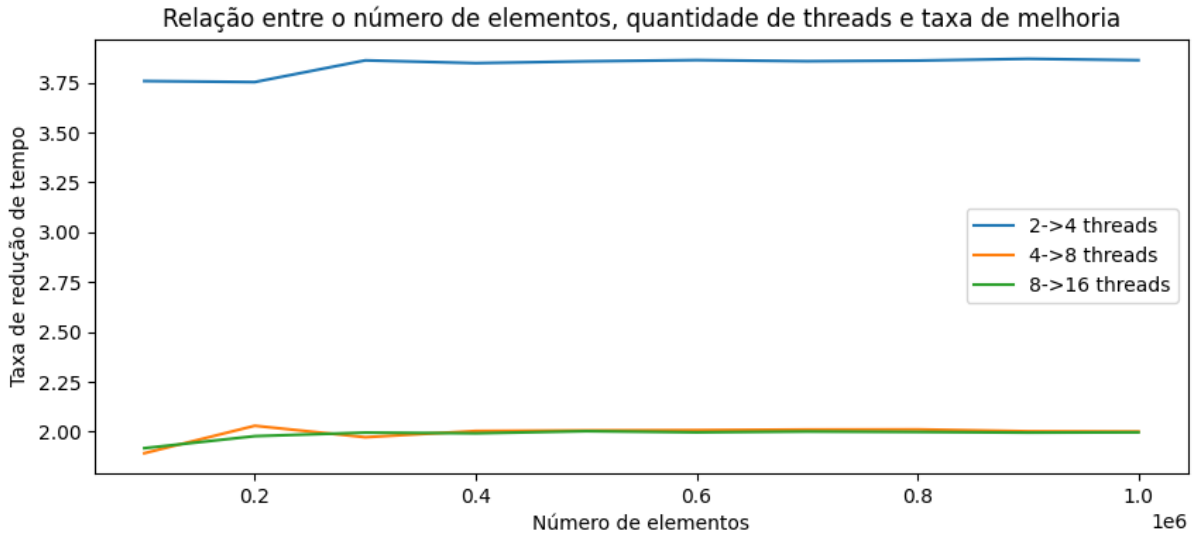

Figura 2