Relatório do: Projeto 1 Sistemas Operacionais - 2018
“Ordenação de Alto Desempenho"

Grupo: Golden Wind

Eliseu Pereira Henrique de Paula - 215293

O código feito para a resolução do dado problema, pode ser dividido em duas partes, uma de leitura e escrita dos arquivos e ordenação.

Para o funcionamento do código, o usuário deve executá-lo desta maneira: “./'nome do executável' 'número de threads que deseja usar' 'n arquivos de entrada' 'arquivo de saída'”. Onde utilizamos dos atributos argc e argv do main, sendo respectivamente suas funções a de contar quantos atributos foram inseridos na execução do programa, e argv é um vetor com o que foi inserido.

A leitura de arquivo foi feita da seguinte maneira:

a) Pegar o valor de argc, entrar em um loop indo de 2 até argc-1 (argc - 1 e o arquivo de saída),
b) Realizar uma contagem de quantos dados tem no arquivo (início do arquivo até end of file), moldar dinamicamente um vetor para que os dados caibam nele, e inserir os dados nesse vetor.
c) A inserção no arquivo de saída, foi feita após a ordenação, onde essa pegava o último argumento de argv e o número contado na leitura, e assim realizava a inserção no arquivo estabelecido pelo usuário.

A ordenação foi feita da seguinte forma:

a) Foi pego a argumento número 1 do argv (número de threads) e convertido para inteiro;
b) Foi feito um vetor de registros, onde o mesmo continha o endereço para o vetor coletado, e os números de começo e final das

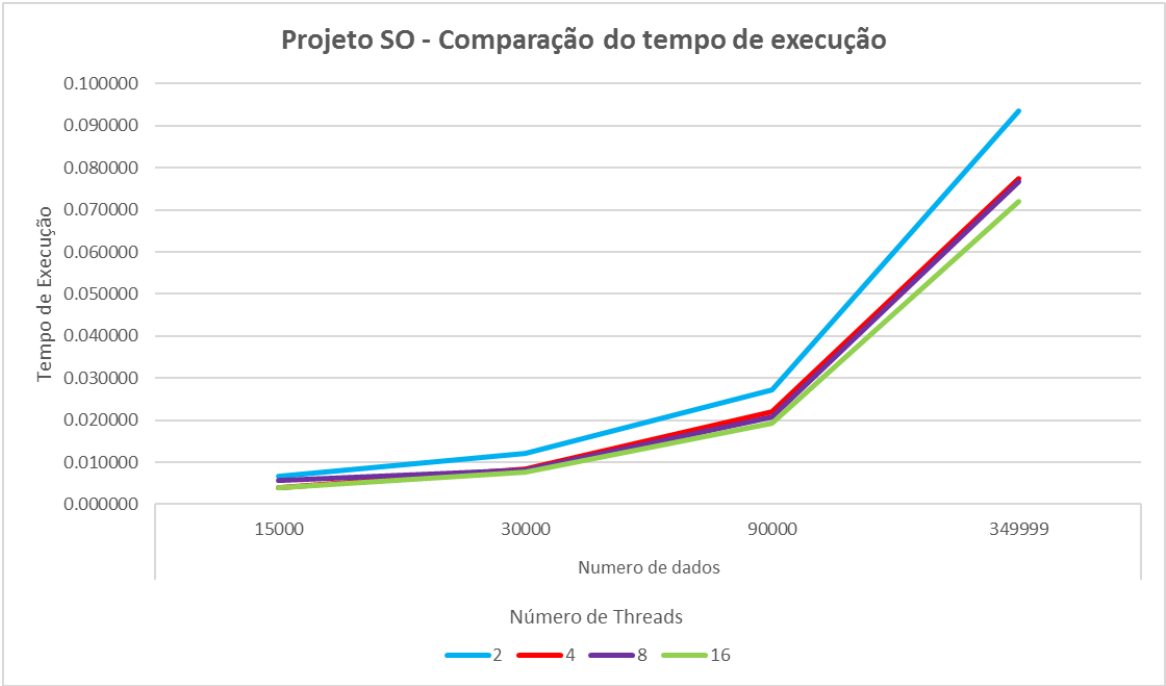
posições divididas para realização da ordenação utilizando vários threads;

c) Foi criado um vetor de threads do tamanho inserido pelo usuário, onde a mesma criava threads dentro de um vetor, passando como argumento, uma função para iniciar os threads e o registro para determinada partição;

d) A função de iniciar a thread chama a função de mergesort, onde a mesma consiste em dividir o vetor, através da recursividade, tornando o em uma árvore, e juntando suas últimas folhas de maneira ordenada, e assim ordenando o vetor;

e) Após a ordenação utilizando os threads, devemos juntar os threads através de um loop, obtendo um vetor semi ordenado, onde deve ser feita uma última chamada do mergesort para realizar a ordenação das parcelas deixadas pelos tamanhos de cada partição, assim obtendo o vetor já ordenado, e inserindo no arquivo.

Comparação do desempenho de cada quantidade threads



Como podemos observar, com o aumento de threads o processo de ordenação se torna mais rápido como apontado no gráfico acima.

Link para o repositório no GitHub: https://github.com/EliseuPHP/projSO

Link para o vídeo: https://drive.google.com/file/d/1k_sLal8-MKGKRtuvdvmrTtrdqlNZkRCO/view?usp=sharing