Universidade Federal
do Espírito Santo

CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

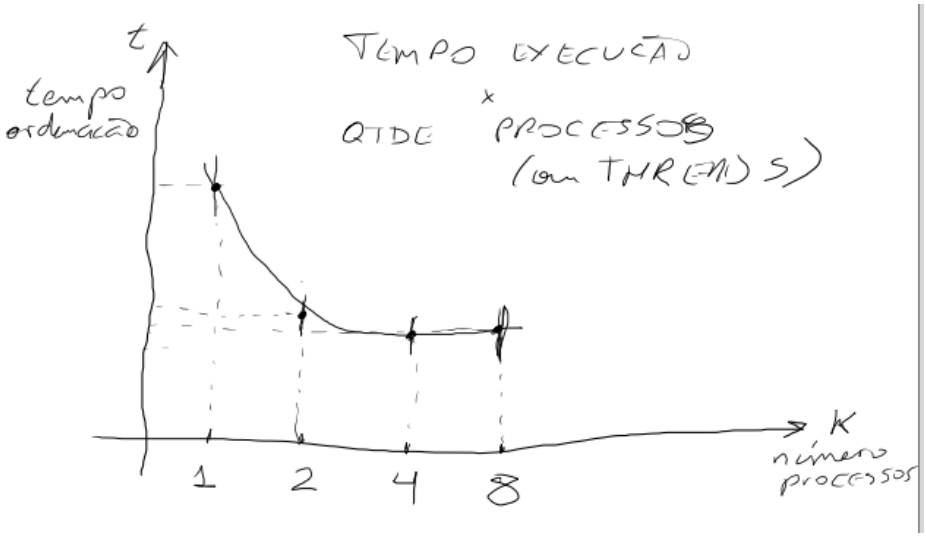
Processamento Paralelo e Distribuído – Turmas 01 e 02 (EARTE) – 2021/1
Prof. Rodolfo da Silva Villaça – rodolfo.villaca@ufes.br
Trabalho T1 – Parte 1 – Paralelismo de Processos e Threads

Objetivo:
Experimentar o paralelismo por meio de processos e *threads* na ordenação de um vetor de grandes dimensões. Comparar o tempo de execução da tarefa de ordenação de vetores com diferentes quantidades de processos ou *threads*.

Instruções:
1. Gerar um vetor v com n=50 milhões de números aleatórios;
2. Criar uma função/procedimento que receba um vetor como entrada, faça a ordenação deste vetor e retorne o vetor ordenado. Pode-se usar qualquer algoritmo de ordenação, inclusive usar algoritmos prontos, importados de bibliotecas nativas das linguagens de programação;
3. O seu programa deverá paralelizar a tarefa de ordenação do vetor v por meio de k processos (k=1, 2, 4, 8) associados à função de ordenação criada no passo 2;
4. Cada processo p_i, sendo i = [1, 2, …, k], recebe um vetor v_i com n/k inteiros e $v_1 + v_2 + \ldots + v_k = v$. Isto é, dividir v em k partes iguais, sendo cada vetor v_i atribuído a uma parte do vetor original, v,;
5. Cada processo ou *thread* deverá ordenar o seu vetor v_i e aguardar o término dos demais k-1 processos ou *threads*;
6. Após os k primeiros processos ou *threads* terminarem suas tarefas, reduzir o número de processos ou *threads* ativas pela metade (fazer k=k/2);
7. Repetir o procedimento a partir do passo 3 enquanto k>0, quando, se tudo tiver sido executado corretamente, o vetor v estará completamente ordenado;
8. **Após a criação do vetor v** (passo 1), inserir marcador de tempo (*timestamps*) no seu código que guardem o instante inicial de execução da tarefa de ordenação do vetor v. **Após o vetor v estar completamente ordenado**, medir o tempo decorrido para ordenação do vetor a partir da instante inicial de execução;
9. Avaliar o tempo de execução para os diferentes valores de k.

Requisitos:
1. O trabalho pode ser feito em grupos de 2 ou 3 alunos: não serão aceitos trabalhos individuais ou em grupos de mais de 3 alunos;
2. Os grupos poderão implementar os trabalhos usando qualquer uma dentre as três linguagens de programação: C, Java ou Python. Os grupos também poderão usar quaisquer algoritmos ou bibliotecas para implementar os algoritmos de ordenação,

exceto algoritmos presentes nativamente paralelos, pois os resultados ficarão prejudicados;
3. O paralelismo pode ser implementado por meio de processos ou *threads*, a critério do grupo, porém observem as limitações de cada biblioteca linguagem de programação (algumas não suportam corretamente, verifiquem a sua escolha);
4. A submissão deverá ser feita por meio de um *link* com a disponibilização do código no Github, em modo de acesso público ou, minimamente, que meu e-mail rodolfo.villaca@ufes.br tenha direito de acesso ao código;
5. A documentação para uso/teste do seu programa deverá ser feita na própria página do Github através do arquivo README[1];
6. Como resultado final do trabalho, cada grupo deverá executar o trabalho 10 vezes para cada valor de k, mantendo o vetor v em cada execução. Deve-se extrair a média e o desvio padrão dessas 10 execuções para cada valor de k. O resultado dessa avaliação pode ser apresentado junto com o README, no Github, ou anexado em formato PDF anexado durante a submissão. Exemplo de gráfico para o relatório, e não deixe de informar no relatório a configuração de processamento do hardware usado na avaliação:



7. Avaliação: Adequação aos Requisitos (30%), Legibilidade do Código (30%), Documentação (40%);
8. Data de Entrega: 09/07/2021 pela Sala Virtual da disciplina no Google Classroom;

Bom trabalho!

1 https://guides.github.com/features/wikis/