Primeiro Trabalho - Relatório Parcial

Computação Concorrente (MAB-117) — 2021/1 REMOTO

Merge-Sort Concorrente

Thalles Nonato Leal Santos | DRE: 119058809

Matheus Augusto Vargas da Silva | DRE: 119039821

1. Descrição do problema

Descrever o problema escolhido:

- *do que se trata e como deve funcionar;*
    - O " merge-sort" trata-se de um algoritmo de ordenação de vetores, ele utiliza do método de divisão e conquista e é chamado recursivamente, o vetor principal é dividido em diversos subgrupos, até chegar a um conjunto unitário, após isso ocorre a junção desses subgrupos, e em cada mesclagem o subgrupo resultante é ordenado, até restar o vetor original ordenado. Em termos de complexidade, o algoritmo entrega desempenho em tempo logarítmico. O uso de um vetor auxiliar faz a complexidade dele em relação a memória ser um pouco mais alta que os demais algoritmos de ordenação, todavia seu tempo de execução compensa.

- *quais são os dados de entrada e qual a saída que deve ser gerada;*
    - Entrada: Quantidade de elementos n do vetor e número de threads, com esses dados será gerado um vetor de tamanho n com números aleatórios.
    - Saída: O mesmo vetor, só que ordenado, além do tempo de execução do programa.

- *apontar que o problema tem paralelismo de dados, com independência de processamento.*
    - No algoritmo sequencial, ocorre a divisão do vetor em subgrupos menores, e a função principal tem que lidar com os diversos tipos para o contexto concorrente, cada thread será responsável por lidar com os diferentes subgrupos e ordená-los.

2. Projeto inicial da solução concorrente

Descrever o projeto inicial da solução concorrente para o problema:

- *listar as estratégias que podem ser usadas para dividir a tarefa principal entre fluxos de execução independentes;*
    - Uma das estratégias possíveis, é a de dividir o vetor em partes, onde cada thread terá seu início e fim calculado, baseado no identificador local da thread (similar ao que foi mostrado na Semana 3). Calcularemos o tamanho de cada bloco dividindo o tamanho total do vetor pelo número de threads. O início dos subgrupos será o índice local de cada thread (0,1,2), multiplicado pelo tamanho do bloco, e para pegar o limite final do vetor será somado ao início de cada subgrupo, o tamanho do bloco. Após isso chamaremos o algoritmo habitual do merge-sort para lidar com as ordenações e, com o fim dos processos feitos pela thread, mesclamos o vetor.

    - Outra estratégia possível, é criarmos uma thread para fazer mergesort(), e toda vez que precisarmos ordenar as metades da esquerda e direita, criamos duas novas threads para ordenar cada metade, depois esperamos elas terminarem com a pthread_join (). E quando as threads terminarem, mesclamos o vetor.

- *apontar qual estratégia será escolhida e por qual motivo;*
    - A estratégia que iremos utilizar é a primeira das listadas acima, isso porque dependendo do tamanho da dimensão do vetor, e da quantidade de vezes que ordenarmos (tanto da direita, quanto para esquerda), a nossa segunda resolução pode ter mais threads sendo requisitadas pelos sistema do que disponíveis na nossa máquina. Então pensando na complexidade de espaço, o primeiro é muito mais viável.

- *descrever as estruturas de dados principais que serão usadas e em qual escopo (global ou local);*
    - Um vetor de entrada global, além do vetor auxiliar que já é usado no merge-sort sequencial.

- *descrever quais argumentos serão passados para as threads e quais valores serão retornados por elas.*
    - Serão passados para as threads os intervalos em que elas irão ordenar o vetor. Não há valor de retorno, pois a ordenação será feita pela modificação do vetor original.

3. Projeto inicial dos casos de teste

Descrever como o programa será testado:

- *descrever o conjunto de possíveis casos de teste para avaliação da corretude da solução proposta;*
    - O conjunto de casos testes será um vetor de dimensão N, que será passado por linha de comando, com números aleatórios (pela função rand). A corretude será avaliada por meio de uma função que percorrerá todo o vetor e avaliará se ele está ordenado.
- *descrever o conjunto de possíveis casos de teste para avaliação de desempenho da solução proposta;*
    - Serão testados vetores de tamanhos 10^6, 10^7, 10^8

4. Referências bibliográficas

Listar as referências bibliográficas utilizadas:

https://discourse.world/h/2020/03/31/Multithreaded-sorting-using-a-thread-pool-in-Java

https://pt.m.wikipedia.org/wiki/Merge_sort

https://www.diva-portal.org/smash/get/diva2:839729/FULLTEXT0