Primeiro Trabalho - Relatório Final

Computação Concorrente (MAB-117) - 2021/1 Remoto

Busca em arquivo - usando abordagem concorrente

Ana Claudia Ribeiro dos Santos
DRE: 118060382

Rebeca Batista Medeiros da Fonseca
DRE: 114156733

1. Projeto Final e implementação da solução concorrente

Quando é feita uma busca por uma palavra em grandes textos, os programas devem fazer uma busca nele todo para mostrar a quantidade de ocorrências do mesmo. Isso pode ser custoso e esse custo está diretamente relacionado com o tamanho do texto em si.

Esse trabalho procura explorar o uso das Threads dividindo as tarefas para filtrar pela quantidade de ocorrência de palavras em grandes textos.

Como entrada teremos a quantidade de Threads a serem utilizadas, o caminho do arquivo e a palavra ou expressão a ser encontrada.

Para uma solução sequencial, os passos são:

- Fazer a leitura dos parâmetros de entrada e verificar se estão certos
- Abrir o arquivo e fazer a busca pela palavra ou expressão do parâmetro
- Encontrando, incrementa uma variável local chamada contador
- Após a leitura de todo o arquivo, retorna para o usuário a quantidade de vezes em que essa palavra foi encontrada.

Para uma solução concorrente, os passos serão parecidos mas com algumas distinções:

- Fazer a leitura dos parâmetros de entrada e verificar se estão certos
- Abrir o arquivo e ver a quantidade de linhas dele
- Dividir, o mais igualmente possível, a quantidade de Threads pela quantidade de linhas.
- Permitir que cada Thread criada faça a busca por um bloco de texto único para ele.
- Cada Thread criada irá executar uma parte do texto e receberá como parâmetro uma struct com alguns dados como o id dela e o número de linhas que deve iniciar a busca. Seu retorno será o número de ocorrências encontradas.

- Encontrando, incrementa uma variável local chamada contador
- Assim que as Thread terminarem, somar as variáveis encontradas
- Após a leitura de todo o arquivo, retorna para o usuário a quantidade de vezes em que essa palavra foi encontrada.

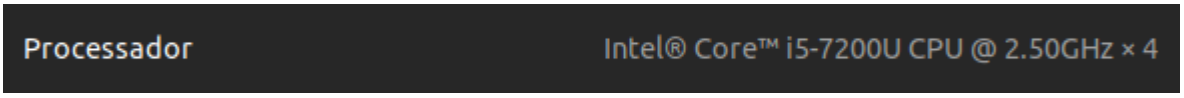
2. Casos de Teste

Os testes foram feitos com um arquivo consideravelmente grande de 300MB e por volta de 1 milhão de linhas.

As cargas das threads foram divididas entre a quantidade de linhas, por exemplo, se o arquivo tiver 200 linhas e são 2 threads, cada thread fica responsável por 100 linhas.
Para facilitar nos testes, a carga foi dividida igualmente nos próprios arquivos, ou seja, as linhas do arquivo tiveram aproximadamente o mesmo tamanho, garantindo que não haja sobrecarga em apenas uma das threads, o trabalho seja separado igualmente.

3. Avaliação de Desempenho

Configuração da máquina testada:



	1 Thread	2 Threads	3 Threads	5 Threads	10 Threads
Tempo da busca	0.1828168480	0.1786664650	0.1873349050	0.2433147500	0.4005069080
Tempo total	1.2424692010	1.2299587390	1.2427134150	1.3023984730	1.4801903780
Aceleração	1.0785	1.0895	1.0783	1.0289	0.9053

Cada teste foi realizado pelo menos 5 vezes e buscamos o menor valor entre eles.
Nota-se que nem sempre aumentar a quantidade de threads é uma tarefa que melhora o desempenho, isso pois existe a sobrecarga de tempo já que é necessário um processamento a mais para criar o mecanismo e a criação das threads em si.

4. Discussão

Por mais que tenha sido um arquivo consideravelmente grande (300MB), o programa executou bem rápido já no sequencial, então não foi possível perceber

bem as acelerações nas threads criadas. Por esse motivo, criar muitas threads não é uma boa estratégia enquanto o sequencial rodar rápido.

5. Referências Bibliográficas

http://wiki.icmc.usp.br/images/8/82/Manipulacao_arquivos.pdf
https://www.geeksforgeeks.org/fgets-gets-c-language/