Laboratório de Programação Concorrente

Lab4 - Computing Sum Equality - 24.2

Objetivo

O sum de um arquivo é uma representação numérica de seu conteúdo. Há várias maneiras de calcular o sum de um arquivo. Neste laboratório, implementamos uma versão simples que retorna a soma dos valores de cada byte do conteúdo do arquivo. O código disponibilizado realiza o cálculo sequencialmente para cada arquivo passado como parâmetro pelo usuário. Vocês precisarão desenvolver uma solução concorrente para a implementação fornecida para o cálculo de sums de arquivos. O objetivo é usar múltiplas threads para processar os arquivos de forma concorrente e diminuir o tempo total de processamento.

Na implementação concorrente, vocês precisam sincronizar threads para implementar três requisitos:

1) retornar o valor da soma de todos os arquivos;

2) limitar o consumo de memória processando no máximo N/2 arquivos concorrentemente (onde n é a quantidade de arquivos passados como parâmetro); e

3) encontrar arquivos que tenham somas iguais.

Para isso, você poderá/precisará implementar exclusão mútua, uso de um multiplex (para controlar um número máximo de threads concorrentes) e esperar que a soma de todos os arquivos tenha sido calculada para, em seguida, encontrar arquivos de mesma soma.

Comportamento esperado - Etapa 1

Para a primeira etapa, o programa deverá escrever na saída padrão, além do sum de cada arquivo individualmente como acontece na solução serial, o valor total com a soma de todos os arquivos. Internamente, você deve criar uma variável global para armazenar esta soma.

Comportamento esperado - Etapa 2

Na etapa dois, mantenham um número máximo de N/2 threads calculando o **sum** de cada arquivo (com N sendo o número de arquivos passados para o programa). É importante destacar que você **deve criar N threads (uma para cada arquivo), no entanto, deve implementar** o controle de admissão (com multiplex) que envolve somente a execução do **sum.** O código da etapa 2 deve incluir o comportamento da etapa 1,

ou seja, o programa deve continuar escrevendo na saída padrão o sum de cada arquivo individualmente e a soma total de todos os arquivos.

Comportamento esperado - Etapa 3

Na última etapa, será necessário escrever na saída padrão os arquivos que tem o mesmo valor de **sum.** Para testar seu código, basta criar alguns arquivos de mesmo conteúdo (por exemplo, criando uma cópia, de nome diferente, de um arquivo já criado). Essa etapa deve manter o comportamento das etapas anteriores. Adicionalmente, na saída padrão deve-se indicar os arquivos de mesmo sum da seguinte forma:

35 filenameM filenameX filenameY

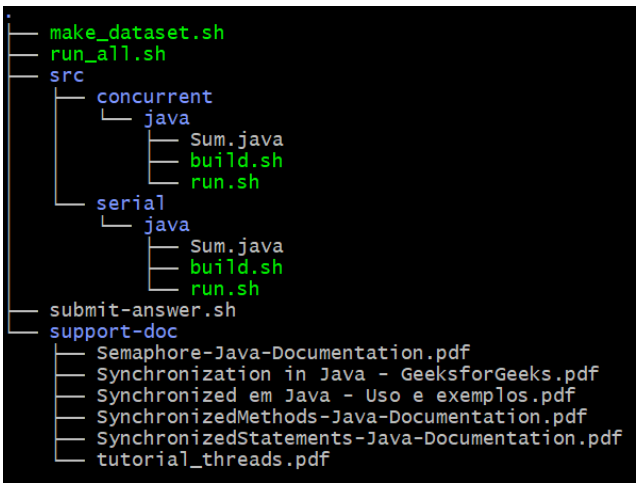
No caso, o primeiro token indica o valor de sum 35 e os demais tokens indicam os nomes dos arquivos que tem sum igual à 35.

Você deve escrever na saída padrão para os casos de arquivos com mesmo sum. Ou seja, se um arquivo for único (não tiver outro arquivo com o mesmo sum) ele não deve ser impresso.

Visão geral do código base

https://github.com/thiagomanel/fpc/tree/master/2024.2/Lab4

O código está organizado na seguinte hierarquia:



- ***src/serial/:*** Diretório que disponibiliza a implementação serial para o cálculo do sum de um conjunto de arquivos.
- ***src/concurrent/:*** Diretório onde você implementará a versão concorrente para o cálculo do sum de um conjunto de arquivos. **Note que, inicialmente, este diretório contém uma cópia da implementação serial, então você deve alterá-la para implementar o que foi solicitado!**
- **scripts auxiliares nos diretórios serial e concurrent:**
  - *build.sh:* script para compilar o código do diretório
  - *run.sh:* script para executar a implementação do diretório. Este exige um ou mais filepath como argumento(s).
- ***submit-answer.sh:*** Script que será utilizado para a submissão da resposta do laboratório.
- ***support-doc:*** Disponibiliza documentação e tutorial sobre o uso de threads, semáforos e sincronizações em Java.
- **script auxiliar *make_dataset.sh*:**
  - Para facilitar o teste do seu código use o script **make_dataset.sh.** Este script recebe um argumento que indica a quantidade de arquivos a ser criada em um diretório padrão. Ou seja, se executado assim:

    bash make_dataset.sh 10
  - O script criará 10 arquivos com conteúdo aleatório no diretório **dataset/** (que também será criado pelo script).
- **script auxiliar *run_all.sh*:**
  - Ainda, você pode usar o script **run_all.sh** que executará as implementações desenvolvidas para processar o dataset gerado pelo script anterior (e, indicará o tempo de execução de cada uma das versões). O código base chama os script *run.sh* serial e concurrent.

Execução do código

1. Clone o repositório do código base

   git clone [link do repositório]

2. Execução da versão serial do código
    a. Navegue até o diretório da implementação serial:

       cd fpc/2024.2/lab4/src/serial/java

    b. Compile o código

       bash build.sh

    c. Execute a versão serial especificando os parâmetros:

       bash run.sh Sum.java

3. Execução da versão concorrente do código
    a. Navegue até o diretório da implementação concorrente:

       cd fpc/2024.2/lab4/src/concurrent/java

    b. Compile o código

       bash build.sh

    c. Execute a versão concorrente especificando os parâmetros:

       bash run.sh Sum.java

Entrega

Você deve criar e manter um repositório privado no GitHub com a sua solução. No entanto, a entrega do laboratório deverá ser realizada por meio de submissão online utilizando o script submit-answer.sh, disponibilizado na estrutura de arquivos do próprio laboratório. Uma vez que você tenha concluído sua resposta, seguem as instruções:

1) Crie um arquivo lab4_matr1_matr2.tar.gz somente com o “src” do repositório que vocês trabalhou. Para isso, supondo que o diretório raiz de seu repositório privado chama-se lab4_pc, você deve executar:
   tar -cvzf lab4_matr1_matr2.tar.gz lab4_pc/src

2) Submeta o arquivo lab4_matr1_matr2.tar.gz usando o script submit-answer.sh, disponibilizado no mesmo repositório do laboratório:

```
bash submit-answer.sh lab4 lab4_matr1_matr2.tar.gz
```

Lembre-se que você deve manter o seu repositório privado no GitHub para fins de comprovação em caso de problema no empacotamento ou transmissão online. Alterações no código realizadas após o prazo de entrega não serão analisadas.

Prazo

13/mar/25 18:00