Laboratório de Programação Concorrente

Lab1

Warmup - 25.1

Objetivo

Neste laboratório, o objetivo é que vocês se familiarizem com a sintaxe e o estilo de diferentes linguagens de programação ao desenvolver soluções concorrentes utilizando múltiplas threads. Para isso, disponibilizamos uma implementação do **cálculo de sum de arquivos** em três linguagens diferentes: **Python, Java e C**. A partir dessas versões sequenciais, você deve desenvolver uma versão concorrente que explore o uso de múltiplas threads para melhorar o desempenho. O sum de um arquivo é uma representação numérica do seu conteúdo. Existem diversas formas de calculá-lo, mas neste laboratório adotamos uma abordagem simples: a soma de todos os bytes do arquivo.

A implementação fornecida atualmente realiza o cálculo de forma sequencial, processando os arquivos um por vez. **Sua tarefa é adaptar essa solução para que o cálculo dos sums dos arquivos seja feito de forma concorrente**, utilizando threads, de modo a explorar paralelismo e reduzir o tempo total de execução.

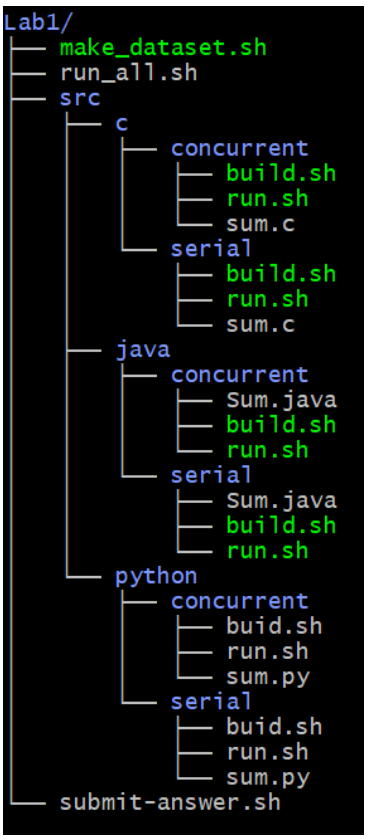
Visão geral do código base

No código base vocês encontrarão implementações seriais em Python, Java e C. Cada implementação recebe como argumentos os “path” de arquivos que devem ter o sum calculado. A sua implementação deve adicionar concorrência a esse processo.

A entrega, detalhada nas seções seguintes, envolverá o código fonte. Iremos avaliar tanto as possibilidades de plágio entre os alunos quanto a geração automática de código.

https://github.com/giovannifs/fpc/tree/master/2025.2/Lab1

O código está organizado na seguinte hierarquia:



- src/*/serial/:
  São os diretórios com as implementações serial em cada linguagem, esse é o ponto de partida para entender o funcionamento do cálculo do sum dos arquivos.

- src/*/concurrent/:
  São os diretórios onde você implementará as versões concorrentes. **Note que, inicialmente, esse diretório contém a mesma implementação que a serial, então você deve alterá-la para implementar a concorrência!**

- make_dataset.sh:
  Script para gerar um conjunto de *n* arquivos que podem ser considerados para input das execuções. Os arquivos serão gerados em um diretório chamado dataset. Para executar esse script, informe o número de arquivos a serem gerados como argumento, por exemplo:
  bash make_dataset.sh 10

- run_all.sh:

Script para compilar e executar as implementações serial e concorrente em todas as linguagens de programação, em seus respectivos diretórios. Este script considera que os arquivos do diretório dataset serão passados como argumentos das execuções.

- submit-answer.sh:
  Script que será utilizado para a submissão de sua resposta.

Preparação

1. Clone o repositório do código base

   ```
   git clone [link do repositório]
   ```

Faça uma comparação de desempenho entre as versões serial e concorrente (detalhes abaixo):

2. Execute o script make_dataset.sh para gerar arquivos

   ```
   bash make_dataset.sh 10
   ```

3. Execute o script run_all.sh para executar todos os cenários considerando os arquivos dataset como argumentos das execuções

   ```
   bash run_all.sh
   ```

Comparação de Desempenho

Entendendo o output do script run.sh:

- real: o tempo total decorrido
- user: o tempo total que o processo gastou utilizando a CPU em modo usuário
- sys: o tempo total que o processo gastou utilizando recursos do kernel

Interpretação

- real: é o tempo que você veria em um cronômetro
- user + sys: representa o tempo efetivamente gasto pela CPU no processamento

Se o programa usar múltiplas threads em um sistema com vários núcleos, o valor de user pode ser maior que real, já que múltiplas threads podem trabalhar simultaneamente.

Prazo

17/06/2024 às 16h00

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 lab1_matr1_matr2.tar.gz somente com o "src" do repositório que vocês trabalharam. Para isso, supondo que o diretório raiz de seu repositório privado chama-se lab1_pc, você deve executar:

   ```
   tar -cvzf lab1_matr1_matr2.tar.gz lab1_pc/src
   ```

2) Submeta o arquivo lab1_matr1_matr2.tar.gz usando o script submit-answer.sh, disponibilizado no mesmo repositório do laboratório:

   ```
   bash submit-answer.sh lab1 path/lab1_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.