Laboratório de Programação Concorrente

Lab2
Image
Restorer - 25.1

Objetivo

Nesse laboratório, vocês irão trabalhar com o melhoramento de desempenho de um programa para restaurar imagens. A deconvolução é uma técnica computacional utilizada para restaurar imagens que sofreram desfoque. Esse desfoque pode ocorrer por fatores como movimento da câmera, desfoco óptico ou baixa qualidade da lente. Entre os algoritmos mais conhecidos para esse processo está o Richardson-Lucy, um método iterativo baseado em convoluções sucessivas com o chamado "ponto de dispersão" (PSF - Point Spread Function). O Richardson–Lucy recupera detalhes de uma imagem borrada por meio de correções iterativas baseadas em convoluções com o kernel conhecido e sua inversa, refinando progressivamente uma estimativa da imagem original. Abaixo, apresentamos um exemplo de imagem digital que possui desfoque, e a mesma imagem após o processamento de restauração da solução disponibilizada no código base.

Neste laboratório, o objetivo é que vocês desenvolvam uma solução concorrente utilizando múltiplas threads para o problema proposto com um desempenho melhor em comparação a solução serial. Para isso, disponibilizamos uma implementação de deconvolução (restauração de imagens) em na linguagem **Java**. A partir dessa versão sequencial, você deve desenvolver uma versão concorrente que explore o uso de múltiplas threads para melhorar o desempenho.

A implementação fornecida atualmente realiza a deconvolução da imagem de forma sequencial, gerando uma imagem final restaurada. **Sua**

tarefa é adaptar essa solução para que a restauração da imagem seja feita de forma concorrente, utilizando threads, de modo a explorar paralelismo e reduzir o tempo total de execução. Ao final da execução concorrente, é importante que você verifique se a imagem final foi gerada corretamente.

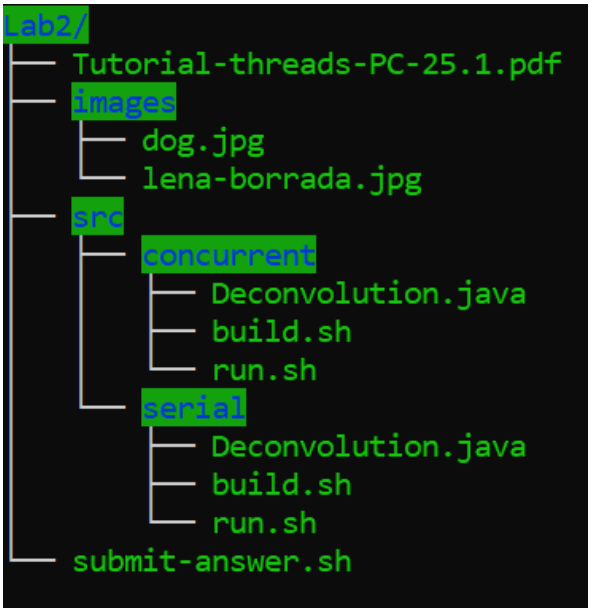
Visão geral do código base

No código base vocês encontrarão a implementação serial em da solução em Java. A implementação recebe como argumento o "path" do arquivo da imagem borrada a ser restaurada. 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/2025.1/Lab2

O código está organizado na seguinte hierarquia:



- src/serial/:
  É o diretório com a implementação serial em Java, esse é o ponto de partida para entender o funcionamento do algoritmo de restauração e planejar sua solução concorrente.

- src/concurrent/:
  É o diretório onde você implementará sua versão concorrente. **Note que, inicialmente, esse diretório contém a mesma implementação que a serial, então você deve alterá-la para implementar a concorrência!**

- images/:
  Diretório com exemplos de imagens borradas que podem ser usadas para testes. **Sugestão: fazer testes iniciais com a imagem lena-borrada.jpg por ser menor e executar mais rápido.**

- src/*/build.sh:
  Scripts para compilar a implementação do respectivo diretório (serial ou concurrent)
  bash build.sh
- src/*/run.sh:
  Scripts para executar a implementação do respectivo diretório (serial ou concurrent). Este script deve receber como argumento o path do arquivo da imagem borrada.
  bash run.sh path/to/image.jpg

- 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. No diretório src/serial, execute os scripts abaixo

   bash build.sh

   bash run.sh ../../images/lena-borrada.jpg

3. No diretório src/concurrent, execute os scripts abaixo

   bash build.sh

   bash run.sh ../../images/lena-borrada.jpg

Como, inicialmente, o conteúdo do diretório concurrent é uma cópia do diretório serial, espera-se desempenho muito próximos de ambas as execuções. No entanto, executando os comandos acima você garante que o código disponibilizado está funcionando em seu ambiente de desenvolvimento.

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

01/07/2025 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 lab2_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 lab2_pc, você deve executar:

```
tar -cvzf lab2_matr1_matr2.tar.gz lab2_pc/src
```

2) Submeta o arquivo lab2_matr1_matr2.tar.gz usando o script submit-answer.sh, disponibilizado no mesmo repositório do laboratório:

```
bash submit-answer.sh lab2 path/lab2_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.