Programação concorrente

2021/2022

Projeto – Parte B

Na segunda parte do projeto os alunos continuar o desenvolvimento de alternativas de parelelização do processamento de imagens.

Na parte A do projeto, foram desenvolvidas duas soluções, em ambas, as *threads* eram responsável pelo processamento de um conjunto pré-determinado de imagens.

Neste projeto os alunos irão implementar duas outras soluções em que cada *thread* não sabe quais as imagens a processar, mas recebem informação dessas imagens através de *pipes* e durante a sua execução.

Também nestas duas novas aplicações, para cada imagem original, são gerados os correspondentes ficheiros *watermark*, *thumbnail* e *resize*.

1 Descrição do projeto

Neste projeto os alunos deverão usar *threads* e *pipes* para implementar duas versões paralelas das aplicações. Nestas versões cada *thread* é responsável por realizar apenas uma das operações (*watermark*, *thumbnail* ou *resize*):

- **thread_thumbnail**
- **thread_watermark**
- **thread_resize**

O utilizador poderá definir quantas réplicas de cada um destes tipos de *threads* existe.

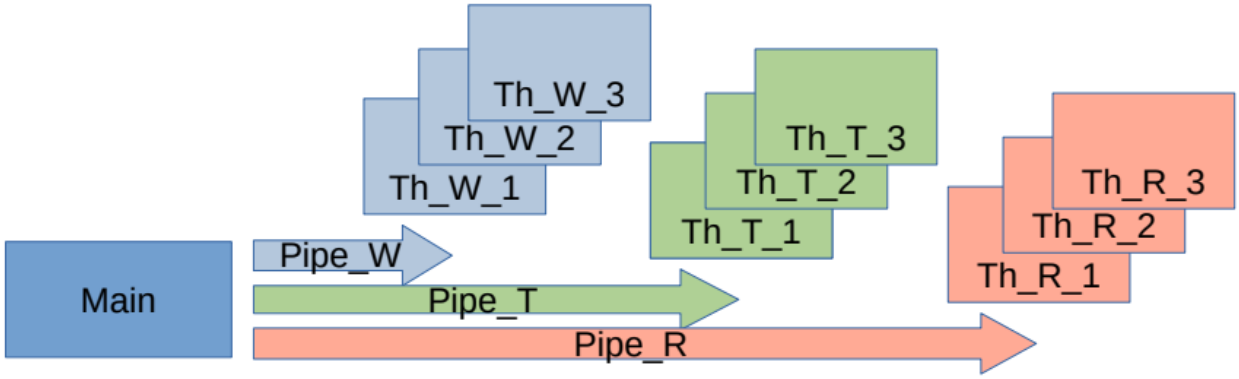
Nesta verão, os nomes das imagens a serem processadas deve ser armazenado num vetor (como nas aplicações da Parte A), mas este vetor não deverá ser acedido pelas

threads. Cada *thread*, processa as imagens cujo nome leu de um *pipe*. Cada *thread*, depois de processar uma imagem deverá voltar a ler o nome de uma nova imagem.

Cada aplicação continua a realizar as três transformações a cada uma das imagens armazenadas na diretoria (produzindo três novos conjuntos de imagens) e realiza-as em paralelo para a acelerar o processo e reduzir o tempo de processamento.

1.1 Paralelização dinâmica

Nesta aplicação paralela (chamada **ap-paralelo-dinamico**) para cada imagem as três diferentes *threads* poderão trabalhar em simultâneo. A primeira imagem deverá ser processada simultâneamente por cada *thread*. Cada *thread*, após a conclusão do seu processamento deverá ler o nome da imagem seguinte.



Para conseguir este tipo de processamento deverão existir 3 *pipes*, em que cada tipo de *thread* lê os nomes dos ficheiros a partir de um dos *pipes*. O *main* escrever os nomes das imagens nos diversos *pipes*, como ilustrado na Figura 1.

1.2 Paralelização pipeline

Na segunda aplicação (chamada **ap-paralelo-pipeline**), também existem três tipos de *threads* e três *pipes* distintos. Também aqui cada um dos tipos de *thread* lê os nomes

dos ficheiros a partir de um dos *pipes* existentes, mas o modo como os nomes são escritos nos *pipes* difere da primeira aplicação, como se pode observar na Figura 2.

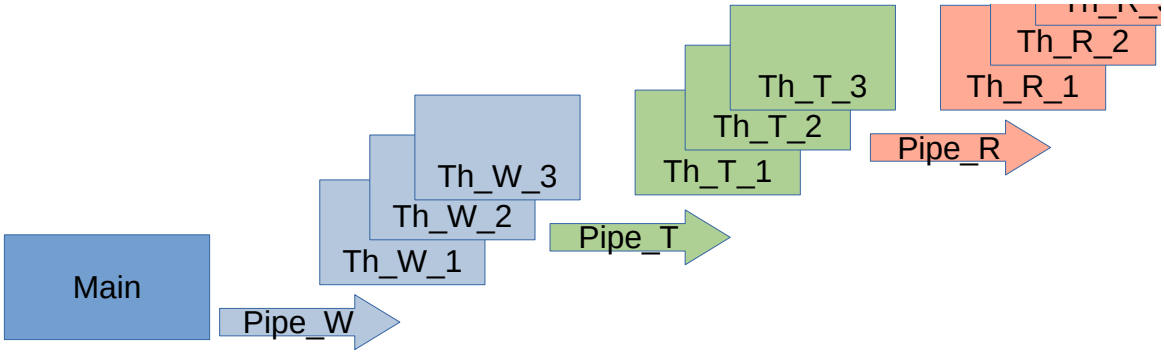


Figura 2: Paralelização pipeline

O *main* apenas escreve os nomes das imagens num dos *pipes* (o *pipe* da watermark). As *threads* responsáveis pela watermark leem os nomes das imagens escritos pelo *main* e depois de processarem cada uma dessas imagens escrevem o nome correspondente no *pipe* associado ao thumbnail. Assim as *threads* responsáveis pelo thumbnail podem começar o seu processamento. De igual forma os nomes das imagens lidos pelas *threads* do resize são escritos no *pipe* correspondente pelas *threads* do Thumbnail após a conclusão do processamento de cada imagem.

Desta forma para cada imagem apenas uma das transformação está a ser realizada em cada instante, como um pipeline normal.

1.3 Número de *threads*

O numero de réplicas de *threads* de cada tipo é definido pelo utilizador num dos argumentos da linha de comando e igual para todos os tipos de *threads*.

Todas as *threads* de um determinado tipo leem os nomes das imagens do mesmo *pipe*, como o **read** pode ser atómicos, cada imagem é processada apenas por uma *thread*.

1.4 Terminação

O *main* deverá terminar ordeiramente após todas as imagens terem sido processadas (watermark, thumbnail e resize).

Existem diversas formas de implementar esta funcionalidade. As duas aplicações desenvolvidas (**ap-paralelo-dinamico** e **ap-paralelo-pipeline)** deverão implementar dois mecanismo distintos de deteção do fim do processamento.

2 Funcionamento das aplicações

As aplicações serão desenvolvida em C e executarão em Linux (ou Cygwin).

Para cada execução, o utilizador deverá indicar através dos argumentos da linha de comando:

- a diretoria onde se encontram as imagens originais,
- o número de *threads* de cada tipo.

Os programas aplicarão as transformações atrás descritas, armazenando as imagens num conjunto de pastas pré-definido. Serão também produzidas estatísticas acerca do tempo de processamento.

2.1 Dados de entrada

A diretoria onde se encontram as imagens originais e o número de *threads* são definidos pelo utilizador na linha de comando aquando da execução das aplicações:

```
ap-paralelo-dinamico dir-1 4   //    ap-paralelo-pipeline dir-2 8
```

O primeiro argumento corresponde à diretoria e o segundo ao número de *threads* de cada tipo.

Se, por exemplo, o utilizador executar o comando

```
ap-parallel-dinamico . 1
```

a aplicação processará imagens na diretoria onde foi executada e utilizará apenas uma *thread* de cada tipo (para além do *main*).

A diretoria indicada pelo utilizador poderá conter mais imagens do que aquelas a serem processadas. O ficheiro `img-process-list.txt`, contém a lista das imagens a serem processadas (um nome de ficheiro por linha). As aplicações deverão ler este ficheiro e só as imagens aí listadas serão processadas.

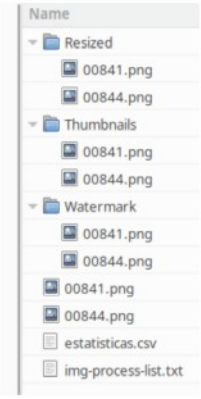
Apenas deverão ser processadas imagens do formato PNG.

Serão fornecidos conjuntos de imagens (com correspondente ficheiro `img-process-list.txt`) de modo a os alunos terem conjuntos de dados variáveis e comparáveis.

2.2 Resultados

A execução das aplicações produz um conjunto de imagens, correspondentes à transformações de cada uma das imagens iniciais. O nome das novas imagens será o mesmo da imagem original, mas cada imagem será colocada numa sub-diretoria específica:

- Resized – sub-diretoria que contém os resultado da redução das imagens
- *Thumbnails* – sub-diretoria que contém os thumbnail produzido
- Watermarks – sub-diretoria que contém a imagens com o watermark aplicado.



2.3 Interrupção de execução

Se as aplicações forem interrompidas a meio do processamento dos ficheiros, apenas parte dos resultados terão sido produzidos e guardados no disco.

Se o utilizador voltar a executar a aplicação, não deverá ser necessário voltar a produzir os ficheiros resultado já existentes. A aplicação só deverá processar e gastar tempo na criação dos ficheiros em falta.

Para verificar se um ficheiro existe os alunos poderão usar as funções *`access()`* como no seguinte exemplo:

```
if( access( nome_fich, F_OK ) != -1){
   printf("%s encontrado\n", nome_fich);
}else{
   printf("%s nao encontrado\n", nome_fich);
}
```

3 Estatísticas

Durante a execução das aplicações, deverão ser recolhidos os seguintes dados:

- instante de início e fim de cada *thread*
- instante de início e fim de processamento de cada imagem original (no caso da aplicação **ap-paralelo-pipeline**)
- instante de início e fim de processamento de cada transformação (ambas as aplicações)
- instante de início e fim de execução da aplicação

Estas estatísticas deverão ser armazenadas num ficheiro chamado `estisticas.csv` na diretoria onde se encontram as imagens.

Este ficheiro deverá ser do tipo *Comma-separated values*[1].

1 https://pt.wikipedia.org/wiki/Comma-separated_values

4 Submissão do projeto

O prazo para submissão da resolução da parte B do projeto é dia **4 de Fevereiro de 2022 às 19h00** no FENIX.

Antes da submissão, os alunos devem criar grupos de 2 e registá-los no FENIX.

Os alunos deverão submeter um ficheiro **.zip** contendo o código de ambas as aplicações. O código para cada uma das aplicações deverá ser colocado numa das seguintes diretorias: **ap-paralelo-dinamico/** ou **ap-paralelo-pipeline/**. Se possível, os alunos deverão entregar também uma `Makefile` para a compilação das aplicações.

Os alunos deverão submeter um pequeno documento/relatório (chamado **pconc-relatorio-final.pdf**).

5 Relatório final

O modelo do relatório será fornecido posteriormente, mas deverá listar as funcionalidades implementadas em ambas as partes do projetos e apresentar os seguintes resultados produzidos com base no ficheiro `estatisticas.csv` para várias execuções das **4 aplicações**:

- tempo total de execução da aplicação
- tempo médio de processamento de cada imagem
- speedup

Para calcular o *speedup*, os alunos devem apenas usar os dados armazenados no ficheiro `estatisticas.csv`.

Os alunos deverão executar as quatro aplicações no computador **sigma** com as várias combinações de:

- dois conjuntos de dados usados na Parte A

- 1, 2, 4, 8 número de *threads*. (para as aplicações da Parte B este valor corresponde ao número de réplicas)

Os alunos deverão garantir que as aplicações são executadas nos mesmos ambientes que as aplicações da Parte A do projeto.

6 Avaliação do projeto

A nota final do projeto será dada tendo em consideração o seguinte:

- Avaliação da Parte A
- Avaliação da Parte B
  - Número de aplicações e funcionalidades implementadas
  - Modo de gestão das *threads* e recursos
  - Estrutura e organização do código
  - Tratamento de erros
  - Comentários
  - Relatório