Instruções para o projeto final

Vocês irão precisar transformar o vídeo em imagens, convertê-las para tons de cinza com o código disponibilizado, processá-las de acordo com o seu projeto, transformar as imagens processadas novamente em vídeo. A versão sequencial é a versão que deverá ser utilizada como critério de corretude para a versão paralela (ex.: Norma de frobenius da diferenca entre as duas).

O vídeo utilizado foi retirado daqui: https://creativecommons.org/about/videos/

Converter video em imagens

1. Instalar o ffmpeg (Se tiver rodando no cluster, pular essa etapa)
    - sudo apt-get install ffmpeg libopenh264-dev (ubuntu, mint, popOs)
    - sudo dnf install ffmpeg openh264-devel (fedora)
2. O video está uma pasta separada (video/). Os frames do video deverão ser gravados em outra pasta (frames/)
3. Abrir o terminal e ir para a pasta base_projeto/
4. Usar o ffmpeg para separar video em imgs (5 eh a ordem de grandeza do num de frames):
    - ffmpeg -i video/1080p.mp4 -qscale:v 10 frames/frame%05d.jpg
5. Criar um arquivo com a lista de imagens a serem processadas. Você deve ir na pasta frames/ e digitar:
    - ls -1 . > ../imagens_para_processar

O arquivo criado deverá ter N linhas, onde N é o total de frames do vídeo. No exemplo em questão N=1800.

Pronto, as imagens estarão na pasta “frames/” e prontas a serem processadas pelo seu projeto.

Executando o projeto

Em seguida, vocês devem observar o arquivo **codigo_base.cpp**. Para compilá-lo você deverá executar e gerar um executável com o nome “nomeexecutavel”:

- g++ **codigo_base.cpp** -o *nomeexecutavel*

Para executá-lo você deverá passar três parâmetros:

- **dir_in/**: Diretório de entrada onde estão os frames a serem processados;
- **dir_out/**: Diretório de saída onde serão salvos os frames processados;
- **list_files**: Lista de arquivos a serem processados

Assim, no terminal, você deverá executar:

- ./*nomeexecutavel* **dir_in/ dir_out/ list_files**

Implementação do seu projeto

Esse código base, atualmente, efetua a leitura das imagens definidas em **list_files**, que estão no diretório **dir_in/**. Em seguida converte-as para tons de cinza e as salva no **dir_out/**.

Você deverá utilizar o **codigo_base.cpp** para auxiliar na implementação do seu projeto. Adicionar código relacionado ao seu tema para gerar tanto a versão sequencial quanto as duas versões paralelas.

Não esqueça de medir os tempos de processamento tanto na versão sequencial como na paralela nos mesmos lugares, para que sejam comparáveis.

Salve as imagens após processamento na pasta “frames_processados/”.

Converter imagens geradas em vídeo

Após o processamento feito no seu projeto, você deverá fazer o contrário: transformar as imagens processadas em video novamente. Para isso, no terminal vá para a pasta do projeto.

Para criar video a partir dos frames processados, no terminal, digite:

- ffmpeg -framerate 24 -pattern_type glob -i ‘frames_processados/*.png’ -c:v h264 -pix_fmt yuv420p video/video_processado.mp4

Cuidado para não sobreescrever os resultados gerados. Sugestão: Crie uma pasta de frames_processados para cada execução. Exemplo:

- **frames_processados_seq**: Para a versão sequencial;
- **frames_processados_mpi**: Para a versão paralela com OpenMPI (Caso implemente essa estratégia);
- **frames_processados_omp**: Para a versão paralela com OpenMP (Caso implemente essa estratégia);
- **frames_processados_cuda**: Para a versão paralela com CUDA (Caso implemente essa estratégia).

Dessa forma, você deve gerar um vídeo para cada implementação dessas que vocês fizerem (ex.: video_sequencial_seq.mp4, ...).

Compilação paralela

Para deixar o código mais legível para vocês, não utilizei ANSI C, mas sim C/C++. Assim, no lugar de compilar com gcc, vocês devem utilizar g++.

OpenMP

g++ codigo_base.cpp -fopenmp -o nomeexecutavel

MPI

mpic++ codigo_base.cpp -o nomeexecutavel

Não esqueçam de validar as versões paralelas com a sequencial. Tampouco de medir os tempos para calcular os speedups e eficiências em cada cenário.