Fakultät für Informatik und Mathematik
Dr. Armin Größlinger
Summer semester 2025

Exercises for Lecture: Practical Parallel Programming
Project 1 (Image Conversion)

Problem 1 (Image Conversion) [10+2 Points]

Write a program that loads a color image in PPM (portable pixmap) format, estimates the color temperature of the image, converts the image from a given color temperature to a new color temperature and saves the converted image to an output file. Use the skeleton in `/scratch/ppp2025/1/colortemp.tar`. After extracting the skeleton (`tar xvf colortemp.tar`), create a separate directory for compiling (e.g., `mkdir build`) and configure the project with

```
cmake ../colortemp        # specify path to the source directory
```

in this directory. After building the project using

```
make
```

the generated binary can be found in `bin/colortemp` and can be run with, e.g., `srun bin/colortemp` on the cluster. Use `colortemp -h` to get a brief summary of the command-line options.

The skeleton contains a sequential implementation of all operations in `src/colortemp/single.c`; see this file for the definition of the operations.

(a) Implement the image conversion using MPI in `src/colortemp/parallel.c`. [5 Points]

Load the image in a single process (e.g. process 0), distribute the image in an efficient manner (for example using `MPI_Scatterv`), perform the estimation of the color temperature and the conversion to a new color temperature, collect the resulting image again (in an efficient manner) and save it to the output file.

(b) As an alternative to loading the image in a single process and distributing it, the image can be loaded in a parallel fashion. The user can request parallel loading with `-L` on the command line.

Extend your program such that when `-L` is specified, the image is loaded using the function `ppp_pnm_read_part` to load a part of the image in each process (see also the example program `invert_pgm_mpi`). [2 Points]

(c) Add OpenMP directives to `src/colortemp/parallel.c` for shared-memory parallelism. [3 Points]

Use shared memory parallelism when estimating the color temperature and when converting the image to the new color temperature.

(d) (Only for master:) Implement parallel saving of the output image using MPI parallel I/O. [2 Points]

Have a look at function `ppp_pnm_write` in `src/ppp_pnm/ppp_pnm.c` for the image format. The PPM format has a simple header followed by the raw image data. Write the header in a single process (e.g., process 0) and then the image data in a parallel fashion using, e.g., `MPI_File_write_at_all`.

When the option `-S` is specified on the command line, the program should use this MPI-based implementation to save the output image instead of calling `ppp_pnm_write`.

Test your program using the images in `/scratch/ppp2025/1`. For benchmarks, you can use the (quite big) image `world.ppm`; use `/dev/null` as output file in this case (to discard the output).

Upload your solution (file `parallel.c`) in ILIAS. Do not miss the deadline! Late submissions cannot be accepted due to legal regulations!

Notes:

PPM Images:

Loading and saving PPM images can be done using the functions from the library `ppp_pnm` provided in the project skeleton. In `src/invert_pgm` you can find the example programs `invert_pgm.c` and `invert_pgm_mpi.c` showing how to use the library (for grayscale images, PGM, portable graymap).

Color images (PPM, portable pixmap) represent the image row by row with 3 bytes per pixel. The three bytes of a pixel contain the red, green and blue components (in this order).

OpenMP:

The number of MPI processes and cores/hyperthreads for OpenMP threads can be controlled using `srun`:

```
srun -n 1 -c 4 ./program
```

This starts one process on 4 processor cores/hyperthreads.

```
srun --cpu_bind=thread -n 6 -c 4 ./program
```

This starts 6 MPI processes and each MPI process uses 4 OpenMP threads (24 threads in total). The option `--cpu_bind=thread` ensures that each MPI process can only use "its" (hyper-)threads of the assigned cores.

You can use the example program `/scratch/ppp2025/0/mpi-example.c` to see the effect of these options on the number of processes and threads.

Use MPI functions only outside of `omp parallel` regions to avoid undesirable interactions between OpenMP and MPI.

Submission due: Monday, May 19, 2025 at 18:00 via ILIAS