高效能巨量資料與人工智慧系統 - HW2

You are asked to implement a padded 2-D convolution program (serial and parallelized) and analyze the experiment results.

Requirements

1. Write a serial padded 2-D convolution code in C.
2. Run the code on your PC with the following inputs and kernels:
   - Input: 256x256, 512x512, 1024x1024, 2048x2048, 4096x4096
   - Kernel: 3x3, 5x5, 7x7, 9x9
3. How the code performs? Report and analyze the performance.
4. Parallelize the code with OpenMP.
5. Verify if your code is correct.
6. Re-do step 2 with 1, 2, 4, 8, 16, 32 threads.
7. Report and analyze the performance.

Some general tips about your report:

- Please provide the details about your experiment platform, e.g., how many cores does it have?
- Elaborate what optimization techniques you applied to parallelized the code.
- Illustrate your experiment results with some charts or tables.
- Other things you want to share.

Execution

A template code and testing data will be provided for you to get started.

We will run your serial code like this:

```
gcc conv.c -o conv
./conv mat-256.txt ker-3.txt ans-256-3.txt
```

and parallel code like this:

```
gcc conv-openmp.c -o conv-openmp -fopenmp
OMP_NUM_THREADS=2 ./conv-openmp mat-512.txt ker-3.txt ans-512-3.txt
```

The first argument is the input matrix file, the second argument is the input kernel file, and the last argument is the answer file containing the convolution result of the specified matrix and kernel.

> ✎ **Note**
>
> By definition, you have to flip the kernel when calculating 2-D convolution.
> For easier implementation, the provided kernels are already filped.

Submission

- Deadline: **03/17 (Mon) 23:59 p.m.**
- Submit a zip file named `<Student ID>_HW2.zip` with the following files:
  - `conv.c`
  - `conv-openmp.c`
  - `<Student ID>_HW2_report.pdf`
- Please make sure your source codes can be compiled and executed without any problem. If some modifications are required to compile and run your code, there will be a small points deduction.

Grading Policy

- Correctness:
  - serial implementation: 10%
  - parallel implementation: 10%
- Scalability: 20%
  - you will get full points as long as your parallelized code can show good scalability based on your experiment platform
- Report: 60%

> ⚠ **Late Submission Penalty:**
>
> - Your score will be multiplied by 70% if you submit your homework after the deadline.

References

- [Kernel (image processing) - Convolution: Wikipedia]

- Using OpenMP with C: CURC User Guide