15618 - Parallel Computer Architecture and Programming

Project Proposal

Agastya Sampath (agastyas), Shijie Bian (sbian)

TITLE : PIPL - Parallel Image Processing Library

URL : https://agastya-sampath.github.io/618-project-s23/

Summary :

We plan to explore parallel implementations of several image processing techniques in the areas of image denoising, color correction and data augmentation. We plan to create an image processing pipeline using these techniques and compare the techniques/applications in terms of their parallelizability.

Background:

As a team, we believe that image processing is a critical area of study due to the ubiquity of images in our daily lives. From social media to medical imaging, images are essential in a variety of fields, and ensuring their quality is of utmost importance. However, image processing can be a computationally intensive task, especially when dealing with large images. As such, we recognize the importance of leveraging parallelism to accelerate image processing techniques. This project provides us with a unique opportunity to contribute to the development of a parallel image processing library, where we can apply and analyze various parallel programming techniques, such as OpenMP, CUDA, and MPI. The non-trivial nature of this project lies in the fact that different processing techniques may benefit from parallelism differently, and thus require careful profiling and analysis to ensure optimal performance

One important aspect of the problem that can benefit from parallelism is the pixel-level operations, such as denoising and color augmentations. For example, in the case of using Ising and Gibbs sampling to do denoising, we can apply

parallelism across pixels to accelerate the process. Similarly, for scaling operations, parallelism can be applied to distribute the workload among multiple threads or processes to reduce the overall processing time. As mentioned, different processing techniques may benefit from different parallelism approaches and units (i.e., units other than pixels). For example, in image scaling, we probably can divide the image into smaller blocks and perform parallel scaling on each block. In color augmentations, we think that we can perform parallel operations on individual color channels. In image segmentation, we should be able to use parallel algorithms to partition the image into segments. Therefore, the major goal of this project is to identify the specific aspects of the multifaceted problem we are trying to solve that can benefit from parallelism, and leverage the correct tools to speed up the image processing library.

The Challenge:

The image processing problem is challenging because different processing techniques may benefit from parallelism differently, which requires careful analysis and profiling to determine the optimal parallelization strategy. Moreover, some image processing algorithms have dependencies between pixels or between neighboring regions, which can make it difficult to parallelize them efficiently. The workload may have varying memory access characteristics, such as locality or high communication-to-computation ratio, which can impact the effectiveness of parallelization. Additionally, some image processing techniques involve divergent execution, where different parts of the code follow different execution paths, which can also make parallelization more challenging. The constraints of the system, such as the available resources and communication bandwidth, may also make mapping the workload to the system challenging. By tackling these challenges, we hope to learn about the trade-offs and best practices in parallelizing image processing algorithms, and ultimately improve the performance of our library.

Resources:

Our resources for the project will include the code and templates from assignments 1 to 4, including CUDA, OpenMP, and MPI, as well as access to the PSC computer cluster. We will also use basic algorithms for image processing, starting with the Ising and Gibbs sampling methods from the MC_Ising GitHub repository (https://github.com/g0bel1n/MC_Ising), and then adding MPI and parallelism. We also intend to reference basic sequential and serialized implementations of common image processing algorithms (https://neptune.ai/blog/image-processing-python-libraries-for-machine-learning) and to start by reproducing results, analyzing performance, and add our parallelism profiling and optimizations. We have not identified any other specific resources needed at this time (other than research papers on some of the algorithms we might choose to implement for color conversion and augmentation), but we will keep an open mind for any additional tools or resources that may be beneficial to the project.

Goals and Deliverables:

- **Plans to achieve:**
    - Implement and parallelize the image denoising module using Ising and Gibbs sampling with CUDA, OpenMP, or MPI. This goal is important because we consider image denoising as our first toolset in our library, and is also something we had in mind in the first place.
    - Implement and parallelize the grayscale to color module with CUDA, OpenMP, or MPI. This goal is important because a grayscale to color conversion is highly powerful and useful in modern digital art, and we would like to explore ways to improve its efficiency without undermining the performance. We consider this as an advanced image processing tool provided in our library.
    - Implement and parallelize the color augmentation module with CUDA, OpenMP, or MPI. This goal is what we had in mind in the first place, and we consider this to be a module that can be attached to

the prior two modules into an image processing pipeline where we first denoise, then color, and if necessary, augment coloring.
    - Conduct performance analysis and profiling of each module with different parallelization methods and determine the best approach for each module.
    - Write a report summarizing the results of our analysis and profiling, detailing the advantages and disadvantages of each parallelization method, and providing recommendations for future improvements.
- **Hope to achieve:**
    - Implement additional image processing modules such as edge detection, image augmentation, and image segmentation. These are some ideas we gained from referencing past projects on the class website and discussed with the instructors. However, these are advanced techniques that require a lot of time and effort.
    - Try a combination of parallelism instead of using only one for each module. This requires extensive parameter and method tuning.
- **What we have in mind for the demo**:
    - We want to demonstrate by showing a raw image that is noisy and with bad coloring, and then our program can use different modules to denoise and augment the color efficiently
    - We are unsure if we can create an effective UI for displaying and profiling the parallelism boosting performance in real-time, but that would be a great add-on to the demo.

Platform Choice:

Our choice of C++ as our primary programming language is well-suited to the parallel programming methodologies that we will be using, such as CUDA, OMP, and MPI (note: we might choose to exclude MPI if the manual overhead becomes clearly significant compared to the other paradigms, since an unfair comparison would not help our project goal). Additionally, C++ is a low-level language that

allows us to optimize our code for memory access and cache locality, which are important considerations in image processing. We will be running our code on both CPU and GPU architectures on Linux, which will allow us to explore different levels of parallelism depending on the task at hand. To aid in our development and testing, we will also be utilizing GHC machines, which provide a convenient environment for debugging and running sanity checks. Finally, we will have access to Bridges PSC supercomputer, which will allow us to perform extensive profiling and optimization of our code. Overall, our choice of platforms is well-suited to the requirements of our project and will enable us to achieve our goals efficiently and effectively.

Schedule:

1. We plan to finish setting up the framework for our library (very basic CUDA/openMP/maybe MPI programs operating on images) by the first week of April.
2. Remaining conservative for the milestone, we plan to implement at least one of our identified image processing algorithms in openMP/CUDA/maybe MPI (aiming for at least a naive implementation of one denoising algorithm, but will attempt more if possible).
3. Finally, we plan to implement at least two more algorithms for image processing (most likely the ones for color conversion and color augmentation) by the end of April.
4. In the last week, we plan to just collect numbers, with maybe a few final touches on our implementations if something goes wrong.

Keeping the above in mind, a tentative weekly schedule is presented in the table below (counting from the submission of this document):

Week	Deadline	Task
1	April 7	Set up the framework for library implementation
2	April 14	Implement one image processing algorithm (denoising), possibly two
3	April 21	Implement second image processing algorithm (color conversion)
4	April 28	Implement third image processing algorithm (color augmentation)
5	May 4	Final touches + Record performance numbers on all implementations. Stretch goal: Combine into a pipeline, or attempt multiple implementations across frameworks

---