PARALLEL IMAGE PROCESSING PROJECT REPORT

Parallelism is a key technique used to improve application performance by leveraging multiple processing cores. In this assignment, we explore different methods of parallelism: process-based and thread-based approaches, as well as various communication and synchronization mechanisms such as pipes, shared memory, semaphores, and atomic operations. Our goal is to analyze and compare their performance when processing an image through three sequential stages (S1, S2, and S3).

1. Baseline Application (Part I)

The baseline version of our image processing application was implemented without parallelism. It processes the image sequentially, performing the stages S1, S2, and S3 in order, for each pixel. The image is read from a file, processed, and the result is written back to the file.

Performance Results:

Read image: 0.00264593 seconds
Smoothen image: 8.42552 seconds
Find details: 1.58902 seconds
Sharpen image: 1.54104 seconds
Write image: 0.00280034 seconds
Total Time:11.57716 seconds

2. Parallel Implementations (Part II)

2.1 Process-based Parallelism with Pipes (Part II.1)

In this approach, we use three separate processes to execute the three stages (S1, S2, S3) of image processing. S1 computes each row and then transfers to S2 and S2 computes that row and transfers it to S3 ....We achieve parellelism at row level.

Performance Results:

Read image: 0.00269204 seconds
Sharpen image: 11.1037 seconds
Find details: 11.1043 seconds
Smoothen image: 11.105 seconds
Write image: 0.00226868 seconds
Total Time:11.105 seconds
SPEED UP:1.04x

2.2 Process-based Parallelism with Shared Memory and Semaphores (Part II.2)

In this approach, shared memory is used to pass data between processes, and three semaphores ensure synchronization between reads and writes and also between stages.

Performance Results:

Read image: 0.00287508 seconds
Smoothen image: 11.5021 seconds
Find details: 11.5021 seconds
Sharpen image: 11.4994 seconds
Write image: 0.00160992 seconds
Total Time:11.5021 seconds
SPEED UP:1.006x

2.3 Thread-based Parallelism with Atomic Operations (Part II.3)

In this version, we use 3 threads within a single process to parallelize the three stages. The threads communicate directly through Global Atomic varaibles, and state and data synchronization is achieved using atomic operations.

Performance Results:

Read image: 0.00259509 seconds
Smoothen image: 10.972 seconds
Find details: 10.9717 seconds
Sharpen image: 10.9238 seconds
Write image: 0.00229816 seconds
Total Time:10.972 seconds
SPEED UP:1.05x

3 Performance Comparison and Analysis

3.1 Runtime Analysis

The thread-based approach using atomic operations yielded the best performance, with a speed-up of 1.05x compared to the baseline.Process-based approaches using pipes and shared memory also showed significant improve- ments, though they were slower due to inter-process communication over- head.

3.2 Challenges

• Pipes: Managing data flow through pipes required careful handling to avoid deadlocks and ensure order of data flow since the row data was being sent.Therefore careful syncronisation was needed,but since we used unnamed pipes they got synchronised precisely.

• Shared Memory: Making sure that S2 and S3 will not start running before S1 and to solve this 3 semaphores were needed to make sure there wont be any data race and that the processes get the correct data from the shared memory.

- Atomic Operations: Although the thread-based approach was eas- ier to implement,It was needed to implement synchronisation between them because the data was being transferred row wise there is chance of S1 writing another row before S2 reading which was solved by using another atomic state variable.

3.3 Integrity Check

We used Checksum hash function to make sure that the transfer of data is correct in between the 3 functions,While sending the row of data we add the sum at last of the data and send the data,The reciever calculates the sum of the recieved data and verifies it with the recieved sum.If both are equal then it will continue,If not it will exit the process.

Conclusion

Our experiments highlight the benefits of parallelism in reducing the total execution time for image processing tasks. While all parallel approaches demonstrated improvements over the baseline sequential version, the thread- based parallelism with atomic operations achieved the best speed-up, offering a balanced approach between ease of implementation and performance effi- ciency.

The process-based methods using pipes and shared memory were also effective but introduced additional complexity due to the overhead of inter- process communication and the need for careful synchronization. Despite these challenges, both process-based approaches still provided significant speed-up, demonstrating their potential for more complex, larger-scale par- allel applications.