# Performance timing and analysis using MPI

PDC Assignment 2 Part 1 2023

Gia Bao Hoang – a1814824

# I. Objective:

The primary goal of this study is to determine the minimum data size for which it is advantageous to parallelize the histogram program on a four-core system such as USS.

# II. Methodology:

In this study, we used the provided histogram.c program and inserted code segments to measure the efficiency of the parallel program. To ensure the accuracy of our results, we chose the longest time among all processes as the representative measure for the runtime of the parallel program. This is because in a parallel computing environment, the overall execution time is determined by the slowest process, as all other processes need to wait for it to finish before the program can be considered complete.

For each combination of dataset size, number of bins, and number of processes used, we ran the histogram.c program ten times, recording ten different runtimes. To minimize the impact of anomalies on our calculation, we use the minimum runtime instead of the median or average of the ten trials for our calculations. The anomalies could be sporadic delays in the computing environment. By selecting the minimum runtime, we aim to obtain a more reliable representation of the actual performance of the parallel program under ideal conditions.

Based on these runtimes, we calculated the efficiency for each dataset size, number of bins, and number of processes used. Efficiency indicates how effectively the resources (i.e., processors) are being utilized. It can be calculated using the following formula:

$$E = \frac{T_{serial}}{p * T_{parallel}}$$

Where E is efficiency, $T_{serial}$ is the serial runtime, $T_{parallel}$ is the parallel runtime, and p is the number of processes used in the parallel program. In this study, we use the runtime on one process as the serial runtime of histogram.c, since it would execute the program sequentially.

To summarize, our methodology involves the following steps:

1. Modify the histogram.c MPI program to include code segments for measuring runtime.
2. Execute the modified program ten times for each combination of dataset sizes, number of bins, and number of processes (1, 2, and 4 processes), and record the runtimes.
3. Select the minimum runtime from each set of ten runs to mitigate the impact of any anomalies.
4. Calculate efficiency for each combination of parameters based on the minimum runtimes.

This methodology enables us to systematically evaluate the performance of the parallel histogram.c program with various dataset size and determine whether we can run it on a four-core system.

# III. Primary Results:

In our study, we collected data on the efficiency of the histogram.c program when run on 2 and 4 processes (parallelize) compared to just one process (sequential). The results are summarized in **Table 1**.

| Efficiencies of Parallel histogram.c | | | | | | | |
|---|---|---|---|---|---|---|---|
| Number of bins | Number of processes | Dataset size | | | | | |
| | | 100 | 1000 | 10000 | 100000 | 1000000 | 10000000 |
| 5 | 1 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| | 2 | 0.65 | 0.75 | 0.79 | 0.78 | 0.81 | 0.80 |
| | 4 | 0.30 | 0.43 | 0.49 | 0.47 | 0.56 | 0.49 |
| 10 | 1 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| | 2 | 0.68 | 0.79 | 0.82 | 0.81 | 0.86 | 0.80 |
| | 4 | 0.38 | 0.51 | 0.58 | 0.58 | 0.62 | 0.54 |
| 20 | 1 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| | 2 | 0.68 | 0.71 | 0.69 | 0.68 | 0.71 | 0.69 |
| | 4 | 0.34 | 0.44 | 0.48 | 0.49 | 0.53 | 0.49 |

**Table 1: Efficiencies of Parallel histogram.c**

We established an efficiency threshold of 0.7, which means that any combination of the number of processes and dataset size with an efficiency above 0.7 is considered worthwhile. This indicates a significant improvement in performance due to parallelization.

From our analysis, it appears that running the program in parallel for 5, 10 and 20 bins using 2 processes is beneficial, particularly when the dataset contains at least 1000 data points. The efficiency remains consistently above 0.7 threshold even as we increase the number of bins. However, when we extend the dataset size to a significant 10,000,000 data points, we do see a minor decrease in efficiency, dipping below the threshold to 0.69 with 20 bins. Despite this slight deviation from our set benchmark, we understand that practical scenarios can afford some degree of flexibility. Considering the marginal difference from our threshold, we still strongly recommend utilizing 2 processes for datasets with at least 1000 data points and above, for up to 20 bins. This strategy ensures a balance efficient resource use and enhanced performance.

In the case of using 4 processes, the efficiency values are significantly lower, mostly falling below the 0.7 threshold. This means that parallelizing the program with 4 processes is not worthwhile, as the benefits of parallelization are outweighed by factors such as increased overhead and workload imbalances.

## IV. Conclusion:

Our analysis concludes that parallelizing the histogram program using 2 processes and a dataset size of at least 1000 data points, with up to 20 bins, is advantageous. The efficiency consistently remains above the 0.7 threshold in this configuration, validating the worth of parallelization. It is important to note a slight efficiency dip to 0.69 at 20 bins and 10,000,000 data points, falling just below the threshold. However, this minor deviation does not outweigh the overall benefits of parallelization. Conversely, employing 4 processes consistently results in efficiency below the 0.7 threshold, indicating limited benefits from parallelization.

As a result, we recommend running the histogram program on the USS system with 2 processes, a dataset size of at least 1000 data points, and up to 20 bins to achieve optimal efficiency. Further research is needed to explore the relationship between the number of bins and the efficiency of parallelization in greater detail.