Protein Matching in Python using MPI

1. Introduction

This project highlights a program that finds matches of a specific pattern in protein sequences. It compares the performance of the matching algorithm using a serial (serial-proteins.py) and a parallel version (mpi-proteins.py) of the program. The protein data is included in the *proteins.csv* file, which has two columns (id and sequence) and many rows.

2. Protein Matching Algorithms

The serial and parallel version of the protein matching algorithm have a very similar structure which involves requesting a pattern from the user and finding occurrences of pattern in the protein sequences file. In the parallel (i.e. MPI) version, the data is split across processes, each of which scans a portion of the protein sequences. The results are then aggregated by the root process to obtain the final count. This approach allows for performance (run time) to be greatly improved, highlighting the benefit of parallelism.

4. Running the programs

To run the mpi version of the protein matching algorithm it is required to install MPICH. MPICH is appropriate for this application because it is an open-source, high-performance MPI implementation that enables efficient parallel communication across distributed systems. The serial and parallel versions of the program can be run using the following commands:

- python serial-proteins.py
- mpiexec -n 4 python mpi-proteins.py

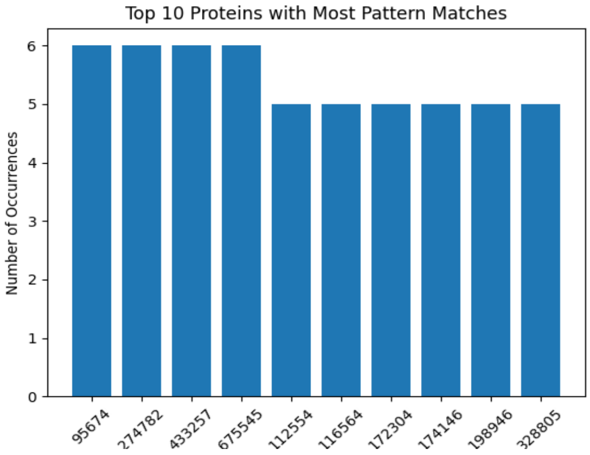
Where the number (in this case 4) determines the number of processes to use.

5. Results and Conclusion

In this section the execution time is from a computer with 10 cores processing a proteins.csv file with 2 million lines. The table below shows the execution times (in seconds) using the serial program for different protein patterns.

ProteinID	Execution time (seconds)	ProteinID max occurrences
A	5.63	1994400
AA	5.38	78203
CCB	5.36	95674
AAAD	5.02	212428
ABCDE	5.53	579

The graph below shows the results for ProteinID = CCB. The same plot is produced by the serial and MPI versions of the program. The x-axis shows the id of the protein sequence and y-axis shows the number of occurences of the pattern (in this case "ccb") in the protein sequence.



The table below shows the execution times (in seconds) using the MPI program for different protein patterns and number of processes.

ID \ Size	1	2	3	4	5	6	7
A	6.60	3.96	3.72	3.58	3.55	3.75	3.68
AA	5.77	3.88	3.46	3.36	3.36	3.50	3.53
CCB	5.63	3.31	3.30	3.18	3.06	3.26	3.26
AAAD	5.29	3.16	3.08	2.97	2.97	3.04	3.20
ABCDE	5.73	3.75	3.34	3.16	3.21	3.45	3.53

Considering the above table, the ideal number of processes N is 5. Now let's consider speedup. Since this problem size can be scaled with the number of processors and the serial portion of the program is very small, Gustafson’s law is appropriate to calculate the theoretical speedup. Timing the MPI program found that essentially the whole program can be parallelized, thus p=1. Let’s take the number of processors to be N = 4. Thus, theoretical speedup is:

$$S = 1 + (N - 1)p$$

$$S = 1 + (4 - 1)1 = 4$$

Considering the serial and MPI program execution times for N=4 and ProteinID = CCB, observed speedup is:

$$S = 5.36/3.18 = 1.69$$

Which means that according to Gustafson’s law more performance could be achieved.