Distributed Dynamic Programming

Parallel Programming in MPI and OpenMP – Project Report

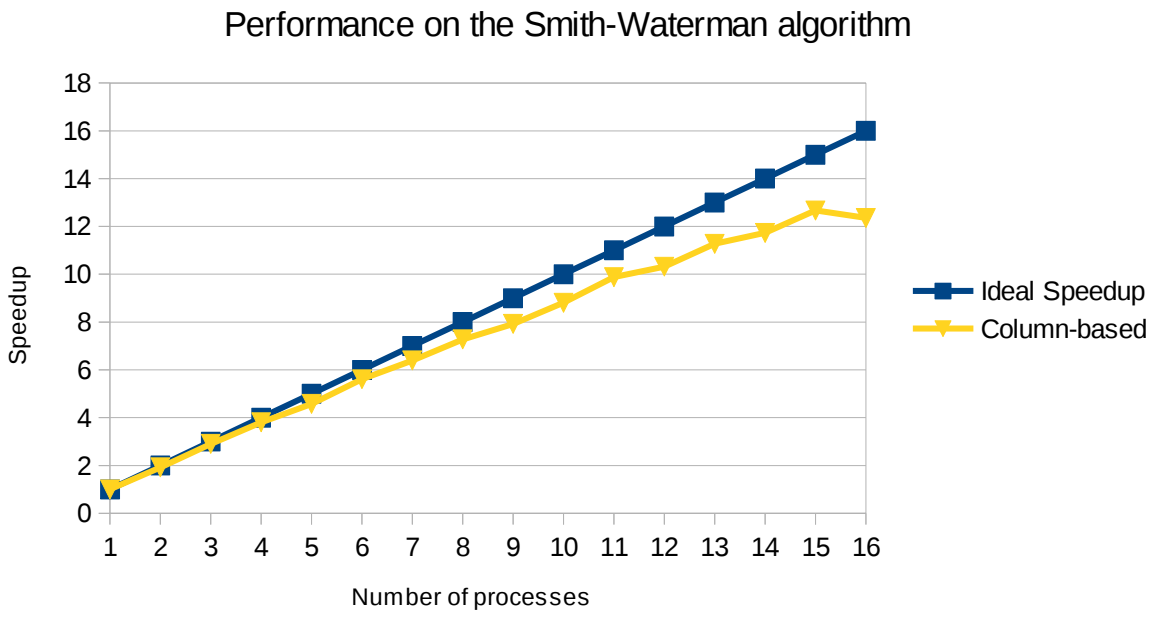
Team members:

- Franky (1110339128)
- Jirka Marsik (1110339127)

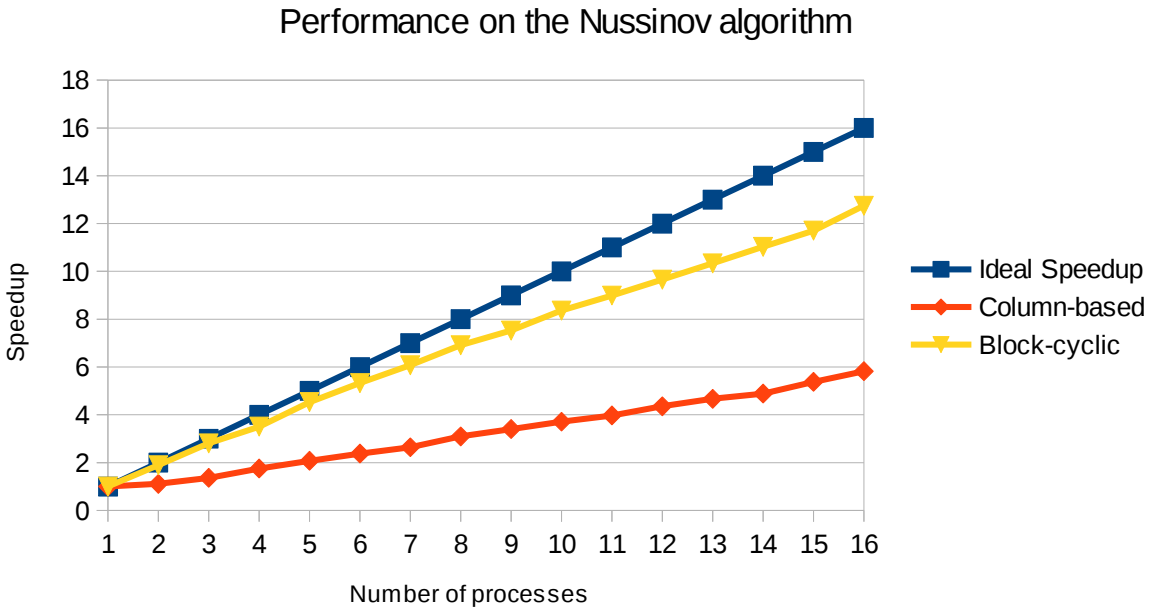
We have succeeded in implementing the block-cyclic based approach to parallel dynamic programming described in “Parallel Design Pattern for Computational Biology and Scientific Computing Applications”. As a consequence of implementing the more general block-cyclic method, we also have an implementation of the simpler column-based method. To test and evaluate our implementation, we have included functions for solving problems using the Smith-Waterman algorithm and the Nussinov algorithm. This let us reproduce and verify the claims from the article cited above.

The implementation ended up being dead simple to write in Python, the short program dprunner.py is included in the attachment. We ended up with a generic implementation which can be easily parameterized by providing three simple functions to get a working dynamic programming solver for a given problem. Examples of such functions are in the files sw.py and nus.py which implement the Smith-Waterman algorithm and the Nussinov algorithm. The implementation turned out just the way we anticipated it, with the sole exception being the dropping of the block height parameter. Since the individual rows we need to transmit form disparate segments of the DP array, there is no benefit from grouping the individual Send commands into blocks.

We evaluated the Smith-Waterman algorithm on sequences of 3000 characters and measured the speedup of the column-based method (the block-cyclic method does not have any effect on this problem, since the complexity of computing the matrix cells is uniform over the entire matrix). Our results are in unison with the paper cited above.



The Nussinov algorithm is more interesting since the time complexity of computing the value in a cell of the DP matrix is linear in the difference of the coordinates (i.e. asymmetric). This means that the simple column-based method of parallelizing the solution is not very suitable, as we can see below. However, the block-cyclic method solves this problem to a great extent, as is confirmed by our trials.



We were pleased to see such performance on the benchmarks and relieved that we were able to reproduce the article's results without too much effort. We will post the resulting source code on GitHub so others who might wish to verify the paper's results or experiment with the algorithm can do so more easily.