Analysis of Parallelized Memory Algorithms in High Performance Computing

***Index Terms*—Internet of Things; IoT Technologies; IoT Applications; Energy Sources for IoT; IoT Challenges.**

I. Introduction

II. Literature Survey

Kotyra *et al.* [1] developed a faster method for calculating flow accumulation, resulting in shorter execution times. They discussed two approaches in parallelizing flow accumulation algorithms: the bottom-up approach and the top-down approach. The study compared six flow accumulation algorithms in sequential, parallel, and task-based implementations, using 118 different data sets. The top-down algorithm was found to be the fastest, with an average execution time of less than 30 seconds. The parallel top-down implementation was the most suitable for flow accumulation calculations. The task-based top-down implementation was less efficient, with an average processing time of 21.1% longer for subcatchments and 32.7% longer for rectangular frames. The algorithm's linear time complexity was measured in various settings, including frame 58 and frame 17 with 240 threads per core. The results showed a strong relationship between the number of cores used and the speedup compared to the sequential version, indicating that increasing the number of cores up to 60 still reduces the average computation time.

Jong *et al.* [2] parallelized and distributed the set of flow accumulation algorithms to determine how the material flows downstream. They used the asynchronous many-task (AMT) approach for parallel and concurrent computations, avoiding synchronization points and promoting composability of modelling operations. The AMT-based algorithms were evaluated for performance, scalability, and composability. It is observed that they can efficiently use additional hardware and perform well when combined with other operations. However, further research is needed to optimize the algorithms for specific hardware architectures and evaluate their performance on larger datasets. The limitation is that the performance of the algorithms was evaluated on a limited set of datasets and that further research is needed to analyze the impact of different hardware architectures, programming languages, flow direction algorithms, and scheduling strategies.

Kotyra *et al.* [3] designed seven fast raster-based algorithms for finding the longest flow paths in flow direction grids using a linear time complexity approach. The algorithms were evaluated using eight large datasets, generated using a hydrological model, and compared with existing GIS software. The algorithms achieved significant speedups, up to 30 times faster on Windows and 17 times faster on Ubuntu, depending on the dataset and algorithm used. The authors concluded that their approach was effective in achieving fast and accurate results for finding the longest flow paths in flow direction grids. However, they noted that their approach is based on raster data and assumes a steady-state flow regime, which may not be applicable to unsteady flow conditions. Future research should explore the development of algorithms based on more complex models and the scalability and portability of their algorithms to other platforms and architectures.

Cho *et al.* [4] suggested longest flow path algorithm which compute few rasters that reduce computational time and improves efficiency. The algorithm is based on depth-first search and breadth-first search, and using equations derived from Hack's law to estimate the longest flow length. The algorithm also employs a branching strategy to eliminate inferior neighbor cells and speed up traversal. The suggested method is evaluated through benchmark experiments conducted in Georgia and Texas, comparing the performance of the algorithm with the Arc Hydro Longest Flow Path tool for ArcGIS Pro. The results showed that the algorithm's performance is affected by disk type and memory size, with solid-state drives and larger memory sizes resulting in faster computation times. The authors conclude that the proposed algorithm is a valuable addition to environmental modelling and software. One limitation is that the experiments were conducted on a limited set of data from two states in the United States, and the results may not be generalizable to other regions or datasets.

References

[1] B. Kotyra, Łukasz Chabudziński, and P. Stpiczyński, "High-performance parallel implementations of flow accumulation algorithms for multicore architectures," *Computers & Geosciences*, vol. 151, p. 104741, 2021.

[2] K. de Jong, D. Panja, D. Karssenberg, and M. van Kreveld, "Scalability and composability of flow accumulation algorithms based on asynchronous many-tasks," *Computers & Geosciences*, vol. 162, p. 105083, 2022.

[3] B. Kotyra and Łukasz Chabudziński, "Fast parallel algorithms for finding the longest flow paths in flow direction grids," *Environmental Modelling & Software*, vol. 167, p. 105728, 2023.

[4] H. Cho, "A recursive algorithm for calculating the longest flow path and its iterative implementation," *Environmental Modelling & Software*, vol. 131, p. 104774, 2020.