GPU-Accelerated Sequence Alignment Algorithms for Bioinformatics

GPU Programming Project Proposal

Lorenzo Ruotolo, s313207

PROJECT OVERVIEW

This project will focus on developing and benchmarking a GPU-accelerated implementation of the **Smith-Waterman algorithm**, widely used for local sequence alignment in bioinformatics. This algorithm is crucial for identifying similarities between DNA or protein sequences, detecting mutations, and comparing homologous genes. It is commonly used in real-world applications, such as sequence similarity searches (e.g., **BLAST**).

Traditional CPU-based sequence alignment methods are slow and inefficient when processing large genomic datasets due to the computational intensity of the dynamic programming matrix. The goal of this project is thus to exploit the parallel processing capabilities of the NVIDIA Jetson Nano GPU, provided during the lectures, to accelerate the Smith-Waterman algorithm and demonstrate the advantages of the employment of GPU computing in the bioinformatics field.

Relevance of GPU Acceleration

Without going much into details, the Smith-Waterman algorithm is computationally expensive because it involves filling a **dynamic programming matrix**, where each element requires calculation based on its neighbouring elements, making it a time-consuming process, especially when working with usually large datasets or long sequences. Each element in the matrix can be computed independently, allowing for multiple simultaneous calculations on a GPU.

PLANNED METHODS

CUDA-Based Algorithm Implementation

The project will focus on implementing the Smith-Waterman algorithm using CUDA on the Jetson Nano's GPU. The matrix computations involved in the algorithm will be parallelized, distributing each cell of the dynamic programming matrix to individual GPU threads to achieve maximum efficiency. In addition, another layer of speedup could be achieved by performing multiple comparisons at once with several different sequences, further improving the performance of large-scale datasets.

Depending on the time available, an interesting option could be to also implement some optimized version of the algorithm to compare with its standard implementation.

Memory Usage Optimization

Since the dynamic programming matrix for long sequences requires significant memory, several strategies seen during the lectures - such as **shared memory** usage and block-wise computation (**tiling**) - will be used to ensure efficient memory access patterns.

Benchmarking

The performance of the GPU-accelerated Smith-Waterman implementation (alongside its optimized version, if possible) will be tested and compared to traditional CPU-based versions of the algorithm. To make the simulation as realistic as possible, real-world genomic datasets, like human DNA sequences and microbial genomes, will be used to evaluate performance. During the evaluation, execution time, speedup factor, and memory usage will be used as performance metrics.

REFERENCES

1. **K. Dohi, K. Benkridt, C. Ling, T. Hamada and Y. Shibata**, "Highly efficient mapping of the Smith-Waterman algorithm on CUDA-compatible GPUs" *ASAP 2010 - 21st IEEE International Conference on Application-specific Systems, Architectures and Processors*, Rennes, France, 2010, pp. 29-36.

2. **L. Ligowski and W. Rudnicki**, "An efficient implementation of Smith Waterman algorithm on GPU using CUDA, for massively parallel scanning of sequence databases" *2009 IEEE International Symposium on Parallel & Distributed Processing*, Rome, Italy, 2009, pp. 1-8.

3. **Liu Y, Maskell DL, Schmidt B**. "CUDASW++: optimizing Smith-Waterman sequence database searches for CUDA-enabled graphics processing units" *BMC Res Notes*. 2009 May 6;2:73.