PARALLELIZING THE SMITH-WATERMAN ALGORITHM FOR GENOME SEQUENCING

Kosta Stojiljkovic, Han Yao Choong, Langwen Huang, Benjamin Langer

Progress report - Design of Parallel and High-Performance Computing - Group 15

1. INTRODUCTION

Genome sequencing has become a promising field tool that is hypothesized to facilitate researchers to find the root cause (and subsequently the cure) for a number of genetic diseases in the future. Of central concern for this ambitious task is the similarity between two sequence portions (effectively two strings). Finding an a close alignment of two sequences indicates to a researcher the possibility of a mutation. The Smith-Waterman algorithm was originally invented for that purpose and different (usually heuristic) variations of it power standard software such as BLAST and FASTA [1].

The investigation into three different algorithms listed in the initial proposal (see attached below) was discarded in favor of a more thorough and deeper look into a single one of them (namely Smith-Waterman). From a computer science perspective all of these algorithms are rather similar, consequently the team decided to focus its efforts on the one which is most relevant in the given domain.

2. PARALLELIZATION STRATEGY

In our first attempts two different strategies were pursued that either exploit the inherent structure of the problem or the overall nature of the problem.

Fine-grained approach. A major step of the Smith-Waterman algorithm is the calculation of a similarity matrix which measure how closely parts of the tow sequences resemble each other. Each entry in the matrix depends only on its neighbors to the west, north, and north-west. For example, calculating the value encircled in red in Figure 1 only depends on the black encircled entries. Exploiting these data dependencies, each anti-diagonal in the similarity matrix may be calculated independently from each other. This is general approach taken by a number of papers [2, Sec. 2]. We plan to run our benchmark also against those projects as soon as a more representative test example is determined.

The current implementation utilizes OpenMP to split calculations along the anti-diagonal among multiple cores.

Coarse-grained approach. Usually researchers are not solely interested in how one particular sequence can be aligned optimally to another. Biologically more interesting is the comparison of one sequence towards a large reference database. For this purpose, we split our mock library file among MPI (message passing interface) nodes and let each one run the algorithm separately. This task is insofar optimal for parallel computers as that no communication between nodes is ever necessary with the exception of generating the output file.

		T	**G**	**T**	**T**
	0	0	0	0	0
G	0	0	3	1	0
G	0	0	③	①	0
T	0	3	①	⑥	4
T	0	3	1	4	9

Fig. 1: Data dependencies between entries

3. WORK SPLIT

Name	Main focus*
Kosta	Coarse-grain approach, MPI
Langwen	CMake setup, Matrix transformation
Han	Bioinf input, OpenMP
Benjamin	Testing, base algorithm implementation

*Various teammates also contributed actively and significantly to other fields.

4. REFERENCES

[1] Veli Mäkinen, Djamal Belazzougui, Fabio Cunial, and Alexandru I. Tomescu, *Genome-Scale Algorithm Design: Biological Sequence Analysis in the Era of High-Throughput Sequencing*, Cambridge University Press, 2015.

[2] Bo Chen, Yun Xu, Jiaoyun Yang, and Haitao Jiang, "A new parallel method of smith-waterman algorithm on a

heterogeneous platform," in *Algorithms and Architectures for Parallel Processing*, Ching-Hsien Hsu, Laurence T. Yang, Jong Hyuk Park, and Sang-Soo Yeo, Eds., Berlin, Heidelberg, 2010, pp. 79–90, Springer Berlin Heidelberg.

DPHPC group: Kosta Stojiljkovic, Han Yao Choong, Langwen Huang, Benjamin Langer

PROJECT PROPOSAL - Parallelization of popular genome sequencing algorithms

In bioinformatics, algorithms that compare to strings for similarity play a decisive role in genome sequencing. Although today's research is dominated by programs like BLAST that produce approximate solutions on large datasets, they fundamentally rely on dynamic algorithms such as:

1. Smith-Waterman algorithm[1]
2. Needleman-Wunsch algorithm[2]
3. Hirschberg's algorithm[3]

These three, closely related algorithms all fall under the category of dynamic programming algorithms for finding the edit distance of two strings (i.e. a measure for their dissimilarity). Our project will consist of exploring different methods for parallelization of these fundamental algorithms while using different techniques (MPI, threading, combination etc.).

In a second step, our research will try to look into the more advanced methods utilized in bioinformatics. The widely used BLAST toolbox[4] (which relies on heuristics due to the generally large data amounts) and minimap[5] which specializes in processing long and noisey (i.e. error-prone) sequences will both serve as a benchmark and loose point of reference for our implementation.

Where possible we will try to relate our work to existing research on parallelization of the aforementioned algorithms and benchmark against those projects if applicable.[6]

Rough project timeline (Oct 11, 2019 - Dec 17, 2019) - 10 weeks

- **Weeks 1-3 (Oct 14 - Oct 31):** Research into dynamic programming parallelization and current implementations and parallelization of algorithms of interest
- **Weeks 4-8:** Work/testing of different implementations
- **Weeks 9-10:** Benchmarking different implementations and prepare for the presentation
- **Until Jan 17, 2020:** Preparing the final report

---

[1] https://en.wikipedia.org/wiki/Smith%E2%80%93Waterman_algorithm
[2] https://en.wikipedia.org/wiki/Needleman%E2%80%93Wunsch_algorithm
[3] Hirschberg, D.S., Algorithm for Computing Maximal Common Subsequences, 1975, http://www.mathcs.emory.edu/~cheung/Courses/323/Syllabus/DynProg/Docs/Hirschberg-LCS-1975.pdf
[4] https://blast.ncbi.nlm.nih.gov/Blast.cgi
[5] Li, H., Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences, 2017, https://arxiv.org/pdf/1512.01801.pdf and https://github.com/lh3/minimap2
[6] E.g. Rashid, N.A. et al., Parallel Homologous Search with Hirschberg Algorithm: A Hybrid MPI-Pthreads, 2007, Solutionhttp://eprints.usm.my/9497/1/Parallel_Homologous_Search_with_Hirschberg_Algorithm_A_Hybird_MPI-Pthreads_Solution.pdf