Published online DD MM YYYY *Nucleic Acids Research, YYYY, Vol. xx, No. xx* ***1–4***
doi:10.1093/nar/gkn000

SIMD implementation of Sequence Alignment Algorithms

Mazyar Ghezelji

Department of Computer Science, University of Victoria, Victoria, Canada.

ABSTRACT

Sequence alignment is one of the most crucial steps in bioinformatics analysis. Many algorithms have been developed with the goal of generating optimal solutions to this problem in an efficient manner. But, optimality in this context, which is more important in some use cases, comes with higher orders of complexity, meaning that the execution time for large datasets would be a bottleneck. One of the ways to reduce the execution time for these algorithms is to utilize parallelization. SIMD (single input multiple outputs) instruction-level parallelization is one of the powerful tools that can reduce the run time of the algorithms substantially. There are tools like SEQ programming language that use this tool to increase their efficiency. In this report we are implementing global, local, and semi-global sequence alignment algorithms using SIMD and we compare the results with SEQ alignment modules.

INTRODUCTION

With the advent of new sequencing technologies, a vast amount of genetic sequences have been available. These sequences can be used to uncover many aspects of humankind's attributes and diseases. Therefore, many researchers from different fields of study are working actively in this area to develop new algorithms and methods for efficient and correct analysis of these data.

Aligning or matching different biological sequences like DNA, RNA, or Protein is referred to as the process in which we are trying to find the best arrangement of subsequences that can help us find relations between these sequences.This process is a crucial step in many downstream analyses of genetic data such as sequence database searching, multiple sequence alignment, genome assembly, and short-read mapping(1). Several algorithms have been designed for this purpose which can be classified into three main groups: global-alignment algorithms that are the end-to-end alignment of two sequences, local-alignment algorithms that will align all substrings of the first sequence with all substrings of the second sequence, and semi-global (glocal) alignment algorithms that align a substring of the first sequence with a substring of the second sequence. Each of these classes can have classic and heuristic algorithms.

As the classic sequence alignment algorithms give the optimal solutions, they have special use cases. But, their high order of complexity makes them impossible to use in large datasets(1). In order to fix this issue, heuristic algorithms have been proposed which can be faster but, they do not guarantee to return the optimal solution(2). Another solution is to use parallelization in the implementation of classic algorithms. Computations of alignment of separate pairs of sequences can be done in a parallel manner as these pairs of sequences have no dependency on each other(3). SIMD (Single-instruction Multiple-Data) Instruction level parallelism is one of the powerful tools that can be used for faster implementation of alignment algorithms(1). It is accessible through most modern processors such as Intel processors that include instruction set extensions like SSE and AVX. These instruction sets work with longer registers that can apply a single instruction to multiple data in one CPU cycle thus, making the implementation faster. Many bioinformatic-specific libraries like ksw2(4) and programming languages like SEQ(5) are utilizing this toolkit to enhance their performance and efficiency.

In this report, we will work with classic algorithms and we will show how these classic algorithms can be much faster using SIMD implementations. We also compare our results with built-in functions in the SEQ programming language.

MATERIALS AND METHODS

Three different sequence alignment algorithms are implemented and benchmarked in this paper. For each algorithm, we compare the normal implementation with the SIMD implementation. Furthermore, SEQ implementation for global alignment has been applied to the data and compared with other implementations.

Sequence alignment algorithms are considered dynamic programming methods and Each of them has three main stages; initializing the scores matrix, recursive completion of the matrix, and trace back the matrix. In the first stage, the initial values of the first row and first column of the matrix are determined. Then based on the recursive formulation of each algorithm, gap and match/mismatch penalties, the rest of the matrix is calculated. Finally, after computing all the values of the matrix we can trace back the matrix based on the specific type of the algorithm in order to find the exact alignment of the two sequences. Gap penalties play an important role in getting optimal results using these algorithms. Linear and Affine gap penalties are the most commonly used methods. Therefore, in this study, we use the linear gap model.

Algorithms

Global-alignment (also known as Needleman-Wunsch)(6) algorithm is a well-known method for determining the degree of similarity between two biological sequences. It aligns two sequences from end to end. Initial step of this algorithm is as follows:

$$M[0,j] = gap \times j$$
$$M[i,0] = gap \times i$$

The reccursion step for global alignment is:

$$M[i,j] = max \begin{cases} M[i-1,j] + gap \\ M[i,j-1] + gap \\ M[i-1,j-1] + sim(s[i],t[j]) \end{cases}$$

where s and t are sequences being aligned. After building the complete scoring matrix, the traceback process begins. In global alignment, the traceback starts at $M[m,n]$ where m is the length of the first sequence and n is the length of the second sequence and finishes at $M[0,0]$.

Local-alignment (also known as Smith-Waterman)(7) algorithm is very similar to the Needleman-Wunsch algorithm and has small adjustments. Its goal is to find substrings of two larger sequences which higher similarity. This algorithm can be used to find preserved parts of genes in an evolutionary process.
Initalising the matrix would be done as shown below:

$$M[0,j] = 0$$
$$M[i,0] = 0$$

We set the first row and first column to zero then compute the matrix entries in the following recurrence manner:

$$M[i,j] = max \begin{cases} M[i-1,j] + gap \\ M[i,j-1] + gap \\ M[i-1,j-1] + sim(s[i],t[j]) \\ 0 \end{cases}$$

The only difference with global alignment is that we would not have negative numbers which means that at any point it can start a new alignment between the remaining parts of two sequences. The traceback process starts at a matrix cell that has the maximum value and stops at a cell that has zero value.
Glocal alignment (also known as semi-global)(8) algorithm is a modification of global alignment that does not penalize gaps at the beginning and end of sequences. It has applications in sequence assembly where the end of one sequence overlaps with the beginning of another sequence.
In the initializing step, we have two cases. If we want to allow gaps at the beginning of the first sequence (query) the first row should be zero and the first column the same as global alignment. If we want to allow gaps at the beginning of the second sequence the first column should be zero and the first row would be the same as global alignment. The recurrence process is the same as global alignment but the traceback is slightly different. Traceback can either start at the maximum entry of the last column (allowing a gap at the end of the first sequence) and finish at the zero-initialized row or column or can start at the maximum entry of the last row (allowing a gap at the end of the second sequence) and finish at the zero-initialized row or column.

Modes of Implementation

Each one of the algorithms mentioned above can be executed on one pair of sequences. If one wants to compare many sequences together, for example in finding homologue sequences in a large dataset, the running time would be a bottleneck. There are two ways (modes) to do these alignments. The first way is to compare all pairs of sequences one by one. It can be done in a scalar way or in parallel which is called intra-alignment. Another way is to align multiple pairs of sequences in parallel, called inter-alignment. Instruction level parallelism in sequence alignment can be applied using longer registers, shown in Fig (these registers are normally 128 or 256 bits long). In this report, we compare scalar and inter-alignment modes in c++ and intra-alignment and inter-alignment in SEQ language.

Benchmark

In order to test sequence alignment algorithms, we generated a dataset of 10,000 pairs of 200 bp sequences where each pair are 10% to 20% similar to each other. Implementations were tested on a Linux machine with 6 core Intel Core i7-8700 CPU and 65GB of main memory. Algorithms were implemented in the c++ programming language using SSE and AVX intrinsic instruction sets. Running times shown in the table are an average of 10 repeats of the algorithms. Gap penalty, mismatch score, and match score are -2, -1, and 1 respectively for all algorithms. Each algorithm was tested in two different modes; with and without the CIGAR (Compact Idiosyncratic Gapped Alignment Report) string which is the final result of the traceback process.
We have compared our implementation with internal global sequence alignment libraries of SEQ language. SEQ is a domain-specific language designed specifically for bioinformatics application purposes. Two different alignment functions of SEQ language are *align*, which uses ksw2(4) as the default alignment kernel for SIMD parallelization in one pair of sequences, and *inter-align*, which uses the BWA-MEM2(9) kernel for inter-sequence parallelizations. Both of these functions were tested and compared with other implementations.

RESULTS

Evaluation of implementations is shown in tables 1 and 2. In general, algorithms with CIGAR strings take more execution time. Scalar implementation of algorithms has the highest execution time as expected. Inter-sequence alignments in c++ had less execution time than scalar implementations. For all of the algorithms, AVX has better performance than SSE as it can use longer registers. In our implementation of algorithms, SSE codes are the same as the AVX codes except for using different AVX/SSE-specific intrinsic libraries and instructions. SEQ

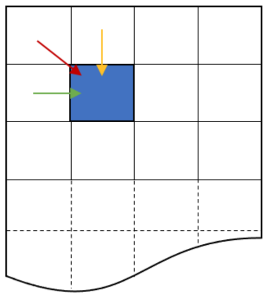

Figure 1. In scalar mode, one entry in the score matrix of each pair of sequences is computed at a time. The value of each cell is dependent on the top, left, and diagonal cells, so they should be computed before we can compute the desired cell.

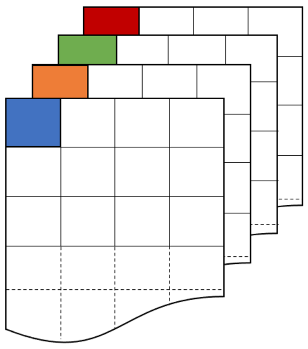

Figure 2. In inter-alignment mode normally 8 to 16 pairs of sequences can be computed using SSE or AVX registers. Each pair of sequences is independent so parallel computing could be applied easily.

	Global	Local	Semi-Global
Scalar	8.819	10.835	8.921
SSE 4	6.333	7.724	6.277
AVX 2	5.729	**7.685**	**6.039**
SEQ (Ksw2)	**0.304**	N/A	N/A
SEQ (BWA-MEM2)	0.312	N/A	N/A

Table 1. Execution time of algorithms without CIGAR

language does not support local and semi-global algorithms yet but Its intra-sequence and inter-sequence global alignment modules (based on ksw2 and BWA-MEM2) have up to 20× better performance in terms of execution time. Although we expect Inter-sequence alignment to perform better in a large body of biological sequence data, the Inter-sequence alignment module of SEQ in our experiments took slightly more execution time than Intra-sequence alignment.

	Global	Local	Semi-Global
Scalar	9.091	10.900	8.959
SSE 4	6.345	7.770	6.314
AVX 2	6.032	**7.714**	**6.079**
SEQ (Ksw2)	**0.363**	N/A	N/A
SEQ (BWA-MEM2)	0.390	N/A	N/A

Table 2. Execution time of algorithms with CIGAR

DISCUSSION

Results of algorithms implementations show that executing them without building CIGAR strings is faster than expected because there is no need for a traceback process. Each SSE register can hold 128 bits of data(four 32-bit integers) and each AVX register can hold 256 bits of data (eight 32-bit integers) so, our expectation is that it can be $4-8\times$ faster than scalar mode but evaluations show that difference between scalar mode execution time and SIMD mode execution time was less than expected. One of the reasons would be that the overload of SIMD instructions are more as the data is larger and need to be loaded and fetched. On the other hand, SEQ module use inter-sequence and intra-sequence parallelisations efficiently and are highly optimised. Evaluations demonstrate SEQ’s high capacity in processing large genomic data in and efficient manner. The most efficient tool in these comparisons is SEQ’s intra-sequence module which is based on ksw2 library which is an implementation of Suzuki-Kasahara diagonal formulation.

CONCLUSION

This project is a comparison of different implementations of classic sequence alignment algorithms. Three main sequence alignment algorithms, global alignment, local alignment, and semi-global alignment were imlpemented in scalar and SIMD modes using c++ programming language. Implementations were compared with global alignment modules, with inter-sequence and intra-sequence parallelizations, in SEQ programming language which is a domain-specific language designed for bioinformatics applications. Evaluations demonstrated that using SIMD instruction-level parallelization can be beneficial for both inter-sequence and intra-sequence alignments.

Conflict of interest statement. None declared.

REFERENCES

1. T. Rognes, “Faster smith-waterman database searches with inter-sequence simd parallelisation,” *BMC bioinformatics*, vol. 12, no. 1, pp. 1–11, 2011.
2. S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, “Gapped blast and psi-blast: a new generation of protein database search programs,” *Nucleic acids research*, vol. 25, no. 17, pp. 3389–3402, 1997.
3. B. Alpern, L. Carter, and K. S. Gatlin, “Microparallelism and high-performance protein matching,” in *Supercomputing’95: Proceedings of the 1995 ACM/IEEE Conference on Supercomputing*. IEEE, 1995, pp. 24–24.
4. H. Suzuki and M. Kasahara, “Introducing difference recurrence relations for faster semi-global alignment of long sequences,” *BMC bioinformatics*, vol. 19, no. 1, pp. 33–47, 2018.

5. A. Shajii, I. Numanagić, R. Baghdadi, B. Berger, and S. Amarasinghe, "Seq: a high-performance language for bioinformatics," *Proceedings of the ACM on Programming Languages*, vol. 3, no. OOPSLA, pp. 1–29, 2019.
6. S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of molecular biology*, vol. 48, no. 3, pp. 443–453, 1970.
7. T. F. Smith, M. S. Waterman *et al.*, "Identification of common molecular subsequences," *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981.
8. M. Brudno, S. Malde, A. Poliakov, C. B. Do, O. Couronne, I. Dubchak, and S. Batzoglou, "Glocal alignment: finding rearrangements during alignment," *Bioinformatics*, vol. 19, no. suppl_1, pp. i54–i62, 2003.
9. M. Vasimuddin, S. Misra, H. Li, and S. Aluru, "Efficient architecture-aware acceleration of bwa-mem for multicore systems," in *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2019, pp. 314–324.