A Novel Approach for Performance Improvement in Parallel Processing

Dr. Ananth Prabhu G, Department of CSE, SCM, Mangalore, educatorananth@gmail.com
Dr. Ganesh Aithal, Research Dean, MITE, Mangalore, India, deanresearch@mite.ac.in
Dr. G.F. Ali Ahammed, Department of CSE, VTU PG CENTRE, Mysore, aliahammed78@gmail.com
Dr. Sarika Tale, Department of DECS, VIAT Muddenahalli, raga.sarika@gmail.com
Mustafa Basthikodi, Research Scholar, Department of CSE, BIT, Mangalore, mbasthik@gmail.com

ABSTRACT

With the high requirements of gene sequencing in the field of scientific research, it is essential to make faster sequencing process. The one of the main sequencing operation is done using Smith Waterman algorithm. This algorithm is used in two conventional ways to evaluate the matrix elements. They are i. Sequential processes ii. Conventional parallel processes. Since the work is to consider these both of these approaches and evolve a new so that these there are three main objectives are met a) Should take less time for the computation of the matrix elements b) optimize the processor used. Even though later one conventional parallel (O(2m)) is faster than the former one sequential processing O(mn), this work tried an attempt for reducing the timing still further, with a challenge to reduce to an extent of O(m) by introducing cross diagonal element wise parallel processing approach. Also as a part of the work processor optimization of the processor for the conventional parallel and the cross diagonal element wise parallel approach is completed with a satisfactory result. The Cross Diagonal Element Wise Parallel Processing Approach (CDEWPPA) performs better than the conventional parallel approach for the query execution time test as well as for the speed up ratio.

Keywords: Gene Sequencing, Smith Waterman algorithm, Cross Diagonal Element Wise Parallel Approach.

1. INTRODUCTION

In the present day’s scenario, the approaches of genomics have played a vital role in optimizing parallel processing systems, since the volume of data is huge it compels the operators to go for parallel operation. Genomics is an emerging field, constantly presenting many new challenges to researchers in both biological and computational aspects of applications. Genomics applications can be very computationally intensive, due to the magnitude of the data sets involved, such as the three billion base pair human genome. Many of these applications involve either searching a database for a target sequence, as done with Hidden Markov Models [HMM] [1].The HMM statistical Markov model assumes that the modeled system is a Markov process which has unobserved states. Genomics applications therefore require immense quantities of computation to explore these hidden states. These data sets are in the form of a database to be searched, as in sequence searching with HMM. These challenges tend to lead to long execution times for the software programs. Therefore, there is a need to develop a novel approach where the execution time can be reduced; also considering the number of processors that are used in these processes to be optimized. It is a well-known fact that in the field of High-Performance computing, the complexity and growth escalates with every day and also the data in the bioinformatics field has seen tremendous growth, which stands as posts guiding. Since the presented work emphasizes on the development of parallelization approach for gene sequencing and thus coming up with an element wise parallel computing approach which could facilitate optimum performance. It is essential to understand the fundamentals of the computation for the conventional processing on how this algorithm works and how it can be improvised with parallel architecture and computing strategies for even more efficient processing. The genome sequencing is done using several approaches, out of which, one of the approach proposed by Smith Waterman is elaborated and used in this work.

Among the various sequential search algorithms, Smith–Waterman (Smith and Waterman, 1981) [2] algorithm is an important and widely used algorithm. Smith–Waterman algorithms have been developed using. Single-Instruction Multiple-Data (SIMD) where a single instruction is capable of executing the same operation on many pieces of data all together in parallel. Single Instruction Multiple Data (SIMD) calculation proposed by Farrar is utilized for crossover Symmetric Multiprocessing - SIMD usage. Compared to using only Symmetric Multiprocessing (SMP) framework, the SMP-SIMD framework offered better efficiency with respect to time. Element programming strategy is utilized to focus similarity between DNA, RNA or protein arrangements. Smith-Waterman calculation was initially proposed as an element programming system by Temple F. Smith and Michael S. Waterman [2]. Parallelization of organic arrangement investigation utilizes restrictive parallel equipment [3], [4]. Numerous endeavors have been carried out for parallel examination of protein correlation. For instance parceling the database of successes to be sought against, and dividing the quantity of clump occupations to be run [5]. To process various protein groupings, the Smith-Waterman calculation ought to be called by every individual succession. The calculation obliges a gigantic measure of processing time, because at whatever point of time, the quantities of successes are huge [7]. For example, if one framework handles 500 groupings, the Smith-Waterman must be called 500 times to perform this undertaking especially, in the usage of parallel Smith-Waterman calculation in SMP [Symmetric Multi-Processing] framework [8]-[10].

2. RELATED WORK

Parallelization approach for gene sequencing is a computing approach that could facilitate optimum performance for sequencing. It is required to have a complete knowledge about the normal processing and its relation with parallel architecture, computing strategies and study gene function and regulation of the nucleic acids or their corresponding proteins that are to be sequenced. To study all these issues we elaborate on some of the research papers as literature survey in this chapter.

In paper [13] the work forms a Fast Local Similarity Analysis; this is also called as Local Similarity Analysis (LSA). LSA algorithm is mainly based on Divide and Conquer method and this can be used to solve the base case of the problem in which the matrix computation has to be done in memory. If this
computation is not possible, then given problem is broken into simpler sub-parts (using divide and conquer method) which are then processed recursively to find out and solve the base case. If the results of all sub parts are integrated together, it yields the full alignment. LSA is not suitable when single instruction and multiple data is considered. The Local Similarity Analysis (LSA) statistic is primarily used to identify the presence of local and lagging relationships. It must also be noted that, determining the significance using the p-value has been very difficult using algorithms as it involves exhaustive permutation tests, the rows have to be shuffled along with the columns and the need to calculate the statistic in iteration. Further, assuming the normality, the p-value is to be calculated, which is a unit used in statistics that is not associated with most of the real world datasets.

In the paper [14] the authors present Farrar’s Streaming SIMD Extensions 2 (SSE2) - Single instruction, multiple data (SIMD) Algorithm SSE2 which is a set comprising of Single Instruction, Multiple Data which is shown in Figure 2.1. Normally, for the Intel processors supporting SSE2 directions, SIMD are composed. These utilization of 128-bit wide SIMD registers are aided by these guidelines. Farrar utilized these directions to expand the velocity of Smith-Waterman (SW) during usage; a question profile will be made in this approach. This dispenses with the need of performing look-ups (reference from the look up table) of substitution network amid interior cycles of Smith Waterman calculations. The question profile will be put away in 16 byte portion, with the goal that a solitary SIMD guideline can get to 16 components from that profile on the external circle of the calculation. On the off chance that the cell size is more than 255, recalculation happens with higher exactness by separating SIMD registers to 8 – 16 bit components. In case, if the length is short of 16 components, then cushion remaining bits utilize nonpartisan qualities. Calculation’s pace is expanded by utilizing single read to get to 16 pre-computed components. We ought to just cushion the remaining bits in the SIMD register with nonpartisan qualities if the remaining inquiry length gets to be short of what 16 components [14]. The execution pace of the algorithm has significantly expanded by getting to these 16 already computed components read from a register. The cells which future figuring’s put away in a buffer. Since Farrar’s approach uses the intra task parallelism approach, the same if performed using the inter task parallelism (multiple alignment tasks are run in parallel) would yield better results.

### 3. PROPOSED SYSTEM

There are two types of solutions for filling the matrix elements of Smith Waterman algorithm for evaluation of the score of the matrix elements:
1. Sequential approach
2. Conventional Parallel approach

In the sequential approach, the matrix elements are filled one by one as shown in Figure 1.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
</tbody>
</table>

Fig 1 Time requirement of sequential evaluation of matrix for a given example of 5 x 4 matrix

For the matrix of $M(4,5)$, the time taken to calculate till the last element is 20 time units. In the conventional parallel approach, the matrix is filled as below in Figure 2.

<table>
<thead>
<tr>
<th>1</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>12</td>
</tr>
</tbody>
</table>

Fig 2 Time requirement of conventional parallel approach for a given example of 5 x 4 matrix

The time taken is 12 units to calculate the last element using conventional parallel approach, which is more efficient than the sequential approach. Afterthis, the sequence of maximum score in the matrix elements are found as a sequence, by tracing back from the cell containing the maximum element called back tracking. This back tracking runs till top left of the matrix, till it encounters a zero, to get sequences of maximum length. This back tracking is used for finding the match of two strings.

Here we present both filling the matrix element and back tracking together, also second method that is parallel approach is presented in a refined way such that the matrix flooding can be processed with a better timing and concurrency which is also called as Cross Diagonal Element Wise Parallel Processing Approach (CDEWPPA).

In the proposed system it improves the performance of Smith-Waterman algorithm by developing 3 novel techniques for the parallel approach. They are
1. Cross Diagonal Element Wise Parallel Processing Approach
2. Parallel back tracking also coined as forward tracking
3. Processor optimization

To find the similarity / match between these two gene sequences, Smith Waterman Algorithm is applied. In this approach, instead of computing it sequentially or conventional parallel approach, it is computed using the proposed Cross Diagonal Element Wise Parallel Processing Approach. In this method, after initialization of the matrix, first row and first column elements, further process of scoring of matrix element starts. In case of existence of score for M(i-1,j-1),M(i-1,j) and M(i,j-1) a processor starts calculating M(i,j). Parallelization starts as more than one exists ofth element, that is M(i-1, j-1), M(i-1,j) and M(i,j-1) score tuple. For the existence of each tuple, it is possible to consider a separate event and a process runs in parallel with the existing process. For each tuple which is generated will call a parallel process and one processor is used for the same. This grows with the number of tuples, as prerequisite for the evaluation and also evaluation grows accordingly. This growth moves in the cross diagonal way and ends at the bottom right corner of the matrix. As it has been seen further that total number of processors will be maximum, when the column reaches equal to number of rows. (considering number of columns is more than number of rows).

### 3.1 Algorithm for the Cross Diagonal Element Wise Parallel Processing Approach (CDEWPPA)

As it was considered in the earlier chapter, take two sequences $A = a_1 a_2 a_3 ... a_n$ and $B = b_1 b_2 b_3 ... b_j$. Sequence elements a and b are to be compared. $S(a_i, b_j)$ is the score of the match in the sequence. After calculation of the score, the traceback process is initiated. Here the cell value having the maximum value is identified and backtracking is done in the direction of the pointer of the cells till a score having zero as the value is encountered. This will yield the local
alignment having the highest score. To get the continuity in matching a gap is used to compensate for the insertion or deletion, whenever a gap is introduced, a score W is levied, this is called gap penalty. Two sequences of length m and n are to be compared, where n > m, where m is number of rows and n is number of columns. In the processes of Smith Waterman algorithm first a matrix of dimension m rows and n columns are considered. The first row and first column is initialized. Till this, the algorithm runs as in the case of earlier method. Further the process of element wise parallization starts. Since three matrix elements M(0,0), M(0,1) and M(1,0) exist, for calculation of M(1,1). This is the starting point of the Cross Diagonal Element Wise Parallel Processing Approach (CDEWPPA). As it completes evaluation of M(1,1) there is a provision for evaluation of M(2,1) and M(1,2), since M(1,0), M(2,0), M(1,1), M(0,1) and M(0,2) are readily available for the calculation. The processor which was used for the evaluation of M(1,1) will continue with the evaluation of M(2,1). A new processor starts for the evaluation of M(1,2). This indicates one existing for the evaluation of M(2,1) and one more processor, for the evaluation of M(1,2), totally there are two processors initiated for the work. This is the second cross diagonal of the matrix. This justifies the term “cross diagonal element wise parallel processing”. Now, the processor which evaluates M(1,2) will continue processing all the elements in that row till it encounters the M(1, n)th element. Also, the processor that calculated M(2,1) will continue processing all the elements of the column till it encounters the M(m,1)th element. Also, it must be noted that, when M(1,2) and M(2,1) are calculated, apart from the above mentioned values till M(1, n) and M(m, 1) respectively. There will also be one more processor that will be computed, which is calculation of M(2,2) because the values M(1,1), M(2,1), M(1,2) are already available.

3.2 Optimization of Processors for the conventional parallel processing approach

Since the first row and first column is filled by zeros, M(1,1) will be evaluated by the processor 1. Once M(1,1) is computed, it gives the provision for the evaluation of M(2,1). This can be processed by Processor 1, because it is free after computation of M(1,1). Now, upon computation of M(2,1), M(3,1) is available for computation and the processor 2 will evaluate M(3,1) and processor 1 will continue to evaluate cross diagonally M(1,2). Now, upon evaluation of M(3,1), M(4,1) is ready for computation. Meanwhile, Processor 1 is free after computing M(1,2). This processor can be reused for computing M(4,1) and meanwhile, M(3,2) will be computed by processor 2. This process will go on till (m, n). Therefore, it is seen by the above argument that for any conventional parallel processing approach, a maximum of 2 processors are sufficient to complete the computation of all the values in the matrix. That is two processors are running in parallel whenever the next processor’s first element becomes eligible then this processor starts for the evaluation in parallel with the earlier, and so on it completes the entire matrix.

3.3 Evaluation of the time constraint of the proposed system

Let us consider a matrix Let M is a matrix with m rows and n columns, where 1: m > n. Let us assume, the time taken to complete evaluation of one score of the matrix element is 1 time unit. The matrix is populated using the Cross Diagonal Element Wise Parallel Processing Approach. The time taken to evaluate the value of M(m, n)th element is the last or bottom right most corner in case of Cross Diagonal Element Wise Parallel Processing Approach evaluated as O (m).

It is easy to calculate cross diagonal of a square matrix. In case of square matrix n x n total numbers of cross diagonals are (n + n – 1) where n + n indicates number of row and number of columns minus 1 indicating the main cross diagonal is considered in both rows as well as columns, so it is necessary to subtract one from the total result. Extending the result to the rectangle matrix. In case of a matrix number of cross diagonals of a rectangle matrix, is calculated as number of rows + (number of columns -1) or number of columns + (number of rows -1) it is written as n + (m – 1) also as m + (n – 1). It is observed that time taken is the function of n or m whichever is high in this case we considered greater than n therefore the time taken is function of m or O (m).

Example: Consider the arbitrary values m=6 and n=7.

Fig 3 Time taken for evaluation of matrix elements using the Cross Diagonal Element Wise Parallel Processing Approach representing the time needed to compute individual cells to a given matrix of 6 x 7.

The matrix has 6 rows and 7 columns respectively. The total number of cross diagonals are 6 + (7-1) = 12.

Table 1: Computation time comparison between sequential approach, conventional approach and proposed Cross Diagonal Element Wise Parallel Processing Approach

4. RESULTS AND ANALYSIS

In this work biological gene sequencing using cross diagonal element wise parallel processing approach in Smith Waterman algorithm is used. In the earlier chapter the details of the Cross Diagonal Element Wise Parallel Processing Approach (CDEWPPA) and its performance has been evaluated analytically. In this chapter the practical evaluation of the performance of CDEWPPA is presented.

In the practical analyses, there are two different type of evaluation.

i. Query execution time analysis
ii. Speed up ratio analysis
4.1 Query execution time analysis

Query analysis is a process used in databases which make use of Structured Query Language (SQL) in order to determine how to further optimize queries for performance. Query analysis is an important aspect of query processing as it helps improve overall performance of query processing, which will speed up many database functions and aspects.

4.2 Speed up ratio analysis

Speed up ratio analysis: Speed up ratio is a quintessential factor in determining the performance related to parallel computing. It focuses on the increasing degree of the programming computational speed from serial computing to parallel optimization. Assuming the time required to run a code on a number of processors is \( T_a \) and the \( T_n \) being the time required where the number of processors to run the same code are \( n \) then the speed up from serial to parallel is given by
\[
S = \frac{T_a}{T_n}
\]

The proposed work is compared with

4.3 Query Analysis for Serial sequencing

Serial sequencing is the process of doing one task at a time, also called serial threading. To compare two sequences of lengths \( m \) and \( n \), as it is shown in Chapter 6, \( O(mn) \) required. The conventional Smith Waterman approach uses serial sequencing technique for computation of matrix elements. By this conventional approach, it takes more time compared to the conventional parallel approach.

4.4 Various Parallel sequencing approaches

There are two approaches in this parallel approach

1. Conventional parallel
2. Proposed cross diagonal element wise parallel processing approach.

To align two sequences of lengths \( m \) and \( n \) using the conventional parallel approach, as discussed in the chapter 6, it requires \( O(2m) \) time. This parallel approach is entirely different from our proposed approach. Wherein, our proposed approach requires \( O(m) \) for the total computation of matrix elements. By the asymptotic notation, it is evident that the proposed method is comparatively better. This is shown practically by the following evaluation techniques. However it is better to go for the practical approach. In this practical approach, the tests are performed using sample data from 1000 genomes website [101] of different sequence lengths of size 2000, 4000, 6000, 8000 and 10000 respectively and the results are presented below. This code is developed on C# programming language on .NET platform and executed on windows 7 system having 4 gigabyte RAM and core i5 quad core processor. The result is shown in the next diagram figure 4.

![Query execution time analysis](image)

**Fig. 4** Query execution time analysis

Figure 4 demonstrates the execution time with query sequences of different lengths as mentioned earlier. The relative execution among serial handling, conventional parallel system and CDEWPPA is shown in figure 4, it can be found here the execution time for CDEWPPA is faster for query length of smaller and larger sequences because the number of processors for computation gradually increase in this process. Moreover, the growth difference in execution time is not very substantial with the increase of sequence length because, the function used for calculating is not exponential. This shows clearly a large difference in the result of CDEWPPA approach.

![Speed up ratio analysis](image)

**Fig. 5** Speed up ratio of sequential approach with analysis for conventional parallel and Proposed CDEWPPA

Figure 5 outlines the number of computations required as the query sequence length increases. It may be noted that, the serial and conventional parallel approach increase the number of computations gradually with sequence length, but in the proposed parallel diagonal approach, the growth in computation is relatively slower.

After analyzing the above results, it is evident that the proposed CDEWPPA is comparatively more efficient than the existing system.

5. CONCLUSION

This exploration work has been carried out to devise an effective parallel programming approach that can prepare a computational strategy for enhancing the time efficiency of gene sequencing in Smith Waterman Algorithm. In order to determine the similarity or similar regions between the two strings, nucleotides or the protein sequences, the Smith Waterman algorithm is used. This dynamic programming algorithm compares the segments of various possible lengths and then optimizes the measure for similarity instead of comparing the whole sequence at a time. In order to figure out
the A’s, C’s, G’s and T’s that are responsible in making up the DNA of an organism, the sequencing of the genomes are carried out. This helps in identifying the order of the DNA nucleotides, bases etc. The prime reason for carrying out this research work is to optimize and accelerate the existing time consuming gene sequencing process. Upon the identification and analysis of the gene from the sequence information, the researchers can look forward for any mutations and crossovers which are directly or indirectly responsible in causing the diseases. Thus, valuable medical information can be mined from the sequence. This research work is carried out with a perspective to improve the quality of lives of human beings all around the globe to enhance better research by the bioinformatics researchers. For matrix filling, the sequential approach fills the individual rows of the matrix with one cell at a time. Only one processor is sufficient for matrix filling. In the conventional parallel system, the values of the cells are computed diagonally and two processors are required for the matrix filling when it is optimized. Whereas, in case of the proposed CDEWPPA approach, it fills the values of individual matrix element considering the evaluation of matrix elements of rows and columns in parallel. It is seen that requires \(n + m - 1\) processors to compute all the values where \(m\) and \(n\) are row and column of the matrix. While the evaluation of time complexity for computing the matrix elements, it has been observed that, for a matrix having \(m\) rows and \(n\) columns, it takes \(O(mn)\) to compute all the values in case of sequential approach, \(O(2m)\) to compute all the elements in case of conventional parallel approach and \(O(m)\) in case of the proposed CDEWPPA. In all cases we assume that \(m > n\).

Instead of the conventional approach of finding the maximum element and tracing back till a zero is encountered for the backtracking process, in this research work, a novice approach is proposed called forward tracking. Here, the process of linking sequences is carried out during the matrix filling process itself, so that upon calculation of the element of the matrix, the result for optimum match is ready as the sequence sorting and comparisons are carried out in parallel. The conventional backtracking process takes \(O(mn)\) units of time because it requires to scan the entire matrix \(M(m,n)\) to identify the maximum element as mentioned earlier in chapter 6. From there onwards, the backtracking is done until a zero is encountered in the direction of the pointer from where the value is generated. This process is continued until the left top corner of the matrix. In forward tracking, the tracking is processed immediately after matrix element is evaluated, at the end of the evaluation of the matrix element; the best match along with other similarity matches is already available. After this, the above 3 approaches are computed for query execution time and speed up ratio. It is observed that, the proposed CDEWPPA is comparatively better than the existing approaches, by figure 4 and figure 5 respectively. Thus, this new approach makes itself as a potential contender to be used for gene sequencing and parallel programming application necessities. The proposed method can also be utilized for string matching and string comparison applications as well.

REFERENCES


