Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 6, June 2014, pg.741 – 753

RESEARCH ARTICLE

DECIPHERING THE SEQUENCE ALIGNMENT BY NEEDLEMAN-WUNSCH ALGORITHM ON TO REDUCE COMPUTATIONAL TIME VIA HIGH PERFORMANCE COMPUTING

Chandrasai Potladurthi[1*], Shyam Perugu[2*], Durga Bhavani S[1]

[1]School of Information Technology, JNT University Hyderabad, Kukatpally, Hyderabad, India

[2]Bio Medical Informatics Centre, National Institute of Nutrition, Hyderabad, India

chandrasaip@hotmail.com, shyambiochem@gmail.com, sdurga.bhavani@gmail.com

***Corresponding Author:**

Chandrasai PotlaDurthi

M.Tech Bioinformatics Student

School of Information Technology (SIT)

JNT University Hyderabad 500085, +91-9985886716

Email: chandrasaip@hotmail.com

ABSTRACT

In accumulation to the old fashioned vastly parallel computers, distributed computer terminal clusters nowadays play an significant role in systematic calculating perhaps due to the introduction of article of trade high performance workstations, low latency and high-band width grids as well as potent improvement tools. In Bio-Informatics, DNA sequence information is vital to indulgent genetic variations. DNA sequencing is the progression of decisive the exact order of the chemical edifice blocks in a sample. This computation requirement is the greatest technical confront in the Human Genome Project. In this approach bioinformatics implements can speediness the exploration of huge scale sequence string data, specifically as regards sequence alignment. The Needleman Wunsch algorithm is used designed for configuration of DNA Sequences under Global Alignment grouping. A high-speed computation resolution is proposed through a parallel adaptation of this algorithm and use of the Alchemi Grid as the processing engine.

In the present study we are going to pursue the Needleman Wunsch algorithm to pairwise alignment designed for configuration of large scale analysis of DNA Sequences. The main scope of this project is to parallelize the Needleman Wunsch algorithm so that it can reduce the computational time and enhance the performance using high performance computing technology.

Keywords: Parallel Computing, Parallel Computing, Genes, DNA, Pairwise Alignment

Technologies used: Java, C, MPI, OPENMP

1. INTRODUCTION

In our present era, a biological data explosion has occurred and also a great hastening in the amassing of biological information began. The reasons for the biological data explosion are the revolutionary recombinant DNA technology used for DNA sequencing and the latest revolution of Genome Sequencing Projects. So, it is at ease to attain the DNA sequence of the gene consistent towards RNA or proteins than it is to experimentally govern its structure or its function. Because of this, we find that the size of sequence databases (e.g. Genbank maintained by NCBI, USA) is larger than the size of structure databases (e.g. PDB, maintained by RCSB, USA), to date. This provides a strong inspiration for emerging computational approaches that can deduce biological evidence from sequence unaccompanied. With the advent of modern computers and information technology, the biological data have not only been stored onto the computer in the form of databases but also processed using computational techniques to get useful results and connections among them.

Bioinformatics is a newly emerging field and it is an integration of statistical, mathematical and computer methods to investigate biological statistics. We use computer package programs towards make inference commencing from the biological data, to make acquaintances among them besides to derive useful as well as interesting predictions. There are several bioinformatics tasks and applications on sequence and structural analysis of biological data. The most important one is computational sequence analysis. In this we will discuss in detail about the sequence alignment algorithms and their applications in bioinformatics tool development. We demonstrate that a protein homology modeling tool development requires the Needleman Wunsch sequence alignment algorithm. In the present study we are going to pursue the Needleman Wunsch algorithm to pairwise alignment designed for configuration of large scale analysis of DNA Sequences. The main scope of this project is to parallelize the Needleman Wunsch algorithm so that it can reduce the computational time and enhance the performance using high performance computing technology.

Sequence Alignment

A sequence alliance is a pattern of writing one arrangement on top of another wherever the residues in one location are thought to have a shared evolutionary origin. If the identical letter ensues in both sequences formerly this position has been well conserved in process of evolution. If the literatures differ it is presumed that the two originate from an ancestral letter. Homologous sequences might have diverse length, though, which is usually elucidated through deletions or insertions in sequences. Therefore, an expanse of letters or a single letter can be paired active through dashes in the supplementary sequence to imply such deletion or an insertion. Meanwhile a deletion in one sequence can continually be seen as an insertion in the supplementary one frequently uses the term "indel".

BANANA-

-ANANAS

In such a humble evolutionarily interested arrangement, an alignment intercedes the definition of a distance on behalf of two sequences. One usually assigns negative figure to a mismatch, 0 to a match and a larger negative figure to an indel. By means of adding these values along an arrangement one gains a score for this alignment. A distance gathering for two sequences can be distinct by looking for the alignment which produces the minimum score. By means of dynamic programing this minimization may be effected deprived of explicitly enumerating all probable alignment of two sequences.

The furthermost elementary sequence analysis job is to align two sequences in a pairwise manner and to find whether the two sequences are related or not. In general, newfangled sequences are reformed from preexistent sequences somewhat than invented de novo. Hence there subsists weighty similarity among a new sequence besides previously known sequences. This is precise privileged for computational sequence examination. Although similarity is a measure of similitude and transformations, independent of the basis of similitude, homology is to the similar sequences then the organisms in which they follow are inclined commencing a common ancestor. If two interrelated sequences be there homologous, formerly we can handover information nearly function and/or structure by means of homology. So, it will be very much useful to find whether the two given sequences are similar, using pairwise sequence alignment algorithms.

Types of alignment

- Global alignment is an end to end matching of two sequences using Needleman-Wunsch algorithm. To find the best overall alignment between sequences.
- Local alignment is a portion or subsequence matching using Smith-Waterman algorithm

The length of a normal DNA Sequence makes about 40KB to 60KB long string in FASTA format [5]. Aligning two DNA sequences [2][3][4][6] requires a long time on a single processor. The algorithm has a complexity of O(NxM) where N is length and M is depth of 2D array. The Parallel Needleman-Wunsch algorithm proposed is based on Needleman-Wunsch algorithm [9] to globally align [7] [8] two DNA Sequences using multiple processors, it reduces the time to O(N+M).

Grid Computing [1] is an emerging technology to provide high performance computing in a virtual organization composed of a large number of computers connected through web based technologies. We have implemented a parallel version of the Needleman-Wunsch algorithm for handling the DNA matching and alignment problem. The Alchemi grid has been used to the run the algorithm in grid environment.

A sequence arrangement is a system of writing one sequence on top of additional wherever the residues in one location are well-thought-out to have a shared evolutionary origin. If the similar letter happens in together sequences then this location is preserved in evolution. Homologous sequence stretches might have dissimilar lengths. Therefore, a stretch of letters or a letter might be matching up with dashes in the additional sequence to indicate such a deletion or an insertion. An insert in one sequence can continuously be seen as a deletion in the additional one; we custom the term ideal intended for such operation.

In such a humble evolutionarily encouraged system, an alignment mediates the description of a distance intended for two sequences. In general assigns some negative number to a mismatch, 0 to a match and a larger negative figure to an indel. By addition these values alongside an alignment one gets a score for an alignment. An expanse purpose for two sequences can be well-defined by observing for the alignment, which produces the minimum mark score. By dynamic programming this minimization might be effected deprived of overtly tallying all possible alignments of two sequence stretches.

A. Global Alignment [15]

Global Alignment shoulders that the two proteins are fundamentally similar over the whole length of another. The alignment tries to match them towards each other as of end to end, even nevertheless parts of the alignment be there not very considerable.

```
RHKPSTKDFGKISESR EFDNQ
        ||||     |
QGIQLERSFGKINMRLEDALV
```

B. Local Alignment [15]

Local alignment examines for alignment segments of the two sequence stretches that match thriving. There is no endeavor to vigor whole sequences keen onto an alignment, just those portions that seem to have worthy likeness, bestowing to particular criterion are considered. By means of the similar sequences by way of above, solitary might get:

```
RHKPSTKDDFGKILGPSTKDDQ
         ||||
QGIQLERSSNFGKINQLERSSNN
```

Most commonly used algorithm for local alignment is Smith-Waterman algorithm [16].

2. RELATED WORK

Needleman-Wunsch algorithm [9][10][11]

All possible pairs of residues (DNA bases or protein amino acids) - one from each sequence - are represented in a 2-dimensional array. The sequences are written across the top and down the left side of the matrix, except that an extra row (row #0) and column (column #0) are added to allow the alignment to begin with a gap of any length in either sequence. The gap rows are filled with penalty scores for gaps of increasing lengths. Maximum possible values are calculated for all other boxes below, to the right of the top row and left column using the above scoring functions. All possible alignments are represented by pathways through this matrix. Each cell is the maximum possible score for an alignment ending at that point. For each cell, look at all possible pathways back to the beginning of the sequence (allowing gaps) and give that cell the value of the maximum scoring pathway.

Figure 1 show the matrix filled with values and pointers. In implementation there are two matrices, one to store the calculated values and another is used to store the pointers which will be used later to trace back for the optimal alignment

F(i,j)		i=0	1	2	3	4
			A	G	T	A
j=0		0	-1	-2	-3	-4
1	A	-1	1	0	-1	-2
2	T	-2	0	0	1	0
3	A	-3	-1	-1	0	2

Figure 1: Filled Needleman-Wunsch Matrix with Traceback

Every non-decreasing path from (0, 0) to (M,N) corresponds to a global alignment of the two sequences.

The succeeding is an instance of global sequence alignment by means of Needleman/Wunsch techniques. Intended for this instance, the two sequences stretches to be globally allied are

GAATTCAGTTA (stretch sequence #1)

GGATCGA (stretch sequence #2)

Thus N = 7 and M = 11 (the length of stretch sequence #2 as well as stretch sequence #1, correspondingly)

An progressive scoring outline is expected wherever Si,j = 2 if the residue at location i of sequence stretch #1 is the corresponding as the residue at location j of sequence stretch #2. Otherwise

Si,j = -1 (mismatch score)

w = -2 (gap penalty)

Initialization Step

The principal step in the global arrangement dynamic encoding approach is to generate a matrix with M + 1 columns as well as N + 1 rows everywhere M in addition to N resemble to the size of the stretch sequences to be allied. The first row as well as first column of the matrix can be primarily filled with 0.

Matrix Fill Step

One probable resolution of the matrix fill step treasure trove the maximum global alignment mark cut by opening in the upper left indicator junction in the matrix in addition finding the maximal mark score Mi,j intended for every location in the matrix. To find Mi,j aimed at any i,j it is slightest to know the mark score for the matrix locations to the left, diagonal and above to i, j. With respect to matrix positions, it is essential to know Mi-1,j, Mi,j-1 in addition to Mi-1, j-1.

For each location, Mi,j is well-defined towards be the maximum mark score at location i,j; i.e.

Mi,j = MAXIMUM [

Mi-1, j-1 + Si,j (mismatch/match in the diagonal),

Mi-1,j + w (gap in stretch sequence #2)

Mi,j-1 + w (gap in stretch sequence #1),

].

On that note in the instance, Mi,j-1 will be green, Mi-1,j-1 will be red and Mi-1,j will exist as blue. With this information, the mark score at location 1, 1 in the matrix can be measured. Meanwhile the 1st residue in both sequences stretches is a G, S1,1 = 2 besides by the suppositions stated earlier w = -2. Consequently, M1, 1 = MAX [M 0, 0 + 2, M1, 0 -2, M0, 1 - 2] = MAX [2, -2, -2].

A value of 2 is formerly placed in location 1, 1 of the scoring matrix. On that note there is likewise an projectile placed hind into the cell that occasioned in the maximum mark score, M[0,0].

Moving to the down the first column towards row 2, we can understand that there is once more a match is there in both sequences. Therefore S1, 2 = 2. Consequently M1,2 = MAX[M0,1 + 2, M1,1 - 2, M0,2 -2] = MAX[2, 0, -2].

An assessment of 2 is then positioned in location 1, 2 of the scoring matrix then an arrow is positioned to point hind to M [0, 1] which leads to the maximum score.

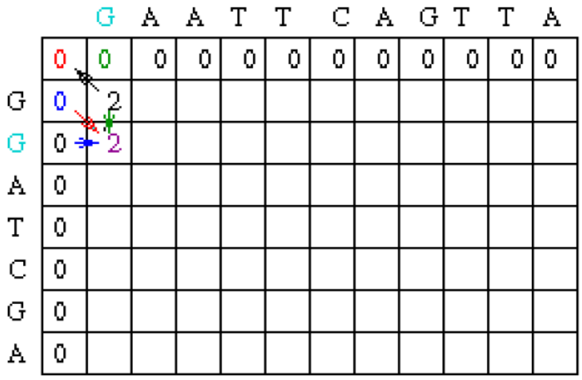


Figure 2 – Filling of Matrix

Observing next to column 1 row 3, there is not identical in the stretch sequences, so S 1, 3 = -1. M1, 3 = MAX [M0, 2 - 1, M1, 2 -2, M0, 3 - 2] = MAX [-1, 0, -2]. A assessment of 0 is then positioned in location 1,3 of the matrix then an arrow is positioned to point hind to M[1,2] which leads to the maximum mark score. We can endure filling in the cells of the matrix by means of the same reasoning. Finally, in the column 3 row 2. Since there is not matching in the sequences at this location, S3, 2 = -1. On that note in the overhead case, there are two unlike ways to acquire the maximum mark score. In that case, pointers are positioned hind to all of the cells that can yield the maximum mark score. Then the rest of the matrix can be filled in. The completed score mark matrix will be as below:

Trace back Step

Afterward the score matrix fill step, the maximum global alignment mark score intended for the two sequences is 3. The trace back footstep will regulate the actual alignment that outcome in the maximum mark score. The trace back footstep starts in the M, J location in the matrix, i.e. the location wherever both sequences be situated globally aligned. Since we have preserved pointers back to all likely predecessors, the trace back step is simple. At every cell, we look to see wherever we move succeeding rendering to the pointers. To instigate, the only possible predecessor is the slanting (diagonal) match.

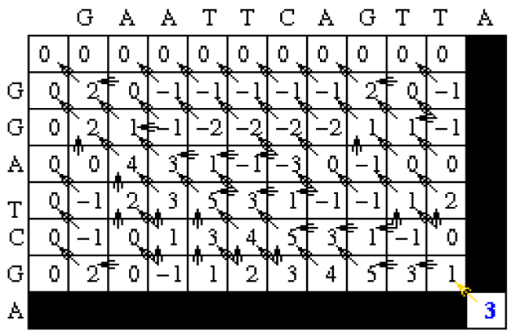


Figure 3 – Initialization of Traceback

This contributes us an arrangement of

A
|
A

On that note the blue letters besides gold projectiles specify the path foremost to the maximum mark score. We can endure to follow the route using a single pointer till we get to the resulting situation.

The alignment by the side of this point is

```
T C A G T T A
|  |     |       |
 T C _ G _ _ A
```

On that note there are now two probable neighbors that might end result in the current score. In that case, in one of the neighbors is arbitrarily selected. Formerly the trace back is completed, it can be understood that there are merely two possible paths prominent to a maximal global alignment. One feasible track is as surveys:

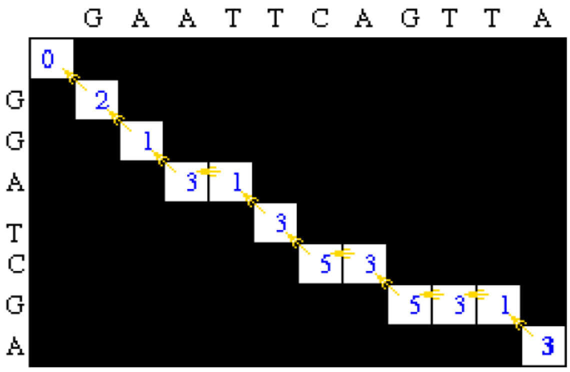


Figure 4 – Complete Traceback of Matrix

Identification that the mark scoring scheme is -1 for a mismatch, +2 for a match and 2 for a gap, together sequences can be experienced to make unquestionable that they end outcome in a mark of 3.

```
G A A T T C A G T TA
 |     |    |  |    |       |
 G G A _ T C _ G _ _ A
 +  -  +  - +  + -  +  -  - +
   2 1 2 2 2 2 2 2 2 2 2
```

So together of these arrangements do indeed result in the maximum alignment mark score.

3. MATERIALS AND METHOD

3.1. REQUIREMENTS

We have used 16 I3 processors with dual core and each system having 4GB RAM and 500GB Hard disk. CISCO Network Routers are used to build a Clusters on Linux Environment

3.2. METHODOLOGY

Parallel Needleman-Wunsch algorithm

A parallel version of Needleman-Wunsch algorithm [9] has been developed; which uses multiple processors for initializing, Calculating and filling the DataMatrix (Stores the DNA Sequences and their calculated values) and the PointerMatrix (Stores DNA Sequences and the pointer values to be used later in backtracking). Our algorithm doesn't include backtracking process to keep record for values to be calculated in each iteration on parallel machines. This algorithm has been implemented on Grid using Alchemi Framework [14]. All the matrices in parallel version of Needleman-Wunsch algorithm are places in global memory space so that all available processors can access them at the same time to perform initialization and other calculations. A. Parallel Initialization of Matrices using different CPUs. In our example we will show the implementation of our algorithm on 3 CPUs. One of which contains global memory and rest of two are used for calculations.

The DataMatrix and PointerMatrix are initialized with DNA Sequences and the gap values are inserted, mathematically. This step is handled in parallel on the participating machines.

PROPOSED SYSTEM

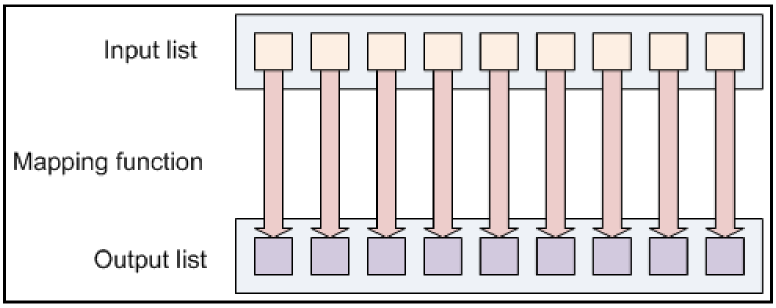


Figure 5: Mapping function

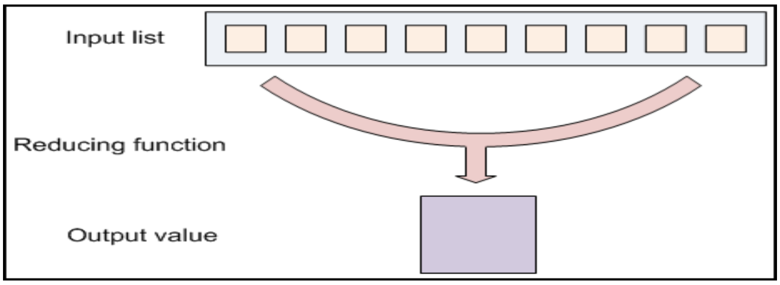


Figure 6: Reducing a list iterates over the input values to produce an aggregate value as output.

It is noticeable that this initialization of both matrices with DNA sequences can be performed in parallel, provided there are four CPUs available, which means that Step 4.2.1 and Step 4.2.2 can be further performed in parallel.

After above step, same is the case in initializing the two matrices with Gap values. Which means Step 4.2.4 and Step 4.2.6 can be performed in parallel as well, provided four CPUs are available.

Parallel For-Loops to fill DataMatrix with two sequences Seq1 and Seq2 using two different CPUs executing each

For-Loop

```
For i=2 to Length of DataArray
        DataArray [0,i] = Seq1[i-2]
For j=2 to Depth of DataArray
        DataArray [j,0] = Seq1[i-2]
```

Parallel For-Loops to fill PointerMatrix with two sequences (seq1 and seq2) using two different CPUs executing each

For-Loop

```
For i=2 to Length of PointerArray
        PointerArray [0,i] = Seq1[i-2]
For j=2 to Depth of PointerArray
        PointerArray [j,0] = Seq1[i-2]
```

Initializing the anchor point of the DataMatrix

```
DataArray [1,1] = 0
```

Parallel For-Loops to fill DataMatrix with GAP values using two different CPUs executing each For-Loop

```
Temp = 0
For i=2 to Length of DataArray
        Temp = Temp + GAP
        DataArray [1,i] = Temp
Temp = 0
For j=2 to Depth of DataArray
        Temp = Temp + GAP
        DataArray [j, 1] = Temp
```

Initializing the anchor point of the PointerMatrix

```
PointerArray [1,1] = 0
```

Parallel For-Loops to fill PointerMatrix with GAP values using two different CPUs executing each For-Loop

```
Temp = 0
```

```
        For i=2 to Length of PointerArray
                Temp = Temp + GAP
                PointerArray [1,i] = Temp
        Temp = 0
        For j=2 to Depth of PointerArray
                Temp = Temp + GAP
                PointerArray [j,1] = Temp
```

SEQUENTIAL ASSIGNING FOR PARALLEL CALCULATION

This step of algorithm is assuming that 2 CPUs are available.

```
CPU1 = 0 // shows CPU 1 is free
CPU2 = 0 // shows CPU 2 is free
For i=0 to Depth of MyArray
        For j=0 to Length of MyArray
                If MyArray [i,j] < > null
                        While (CPU1 < > 0 OR CPU2 < > 0 )
                        {       If CPU1 == 0
                                DataArray [i,j] = MaxofCPU1( MyArray [i,j] )
                                Else
                                DataArray [i,j] = MaxofCPU2( MyArray [i,j] )
                        }       Else
                                Exit j Loop
```

For-i loop check for Parallel Values to be calculated

For-j assigns CPUs the indexes for which they will calculate values

INSTRUCTIONS ON EACH CPU - INT MAXOFCPUN (INT I,INT J)

The variables i and j are the coordinates of the DataMatrix's value to be calculated.

```
int max(int i, int j)
{       Diagonal = DataArray [i-1,j-1]
        Up = DataArray [i-1,j]
        Left = DataArray [i,j-1]
```

```
If ( DataArray [i,0] == DataArray[0,j] )
        Diagonal = Diagonal + MATCH
Else
        Diagonal = Diagonal+ NoMATCH
Up = Up + GAP
Left = Left + GAP
If ( Diagonal > Left AND Diagonal > Up )
        PointerArray[i,j]="3"
        return Diagonal
Else If( Up > Left )
        PointerArray[i,j]="2"
        return Up
Else
        PointerArray[i,j]="1"
        return Left        }
```

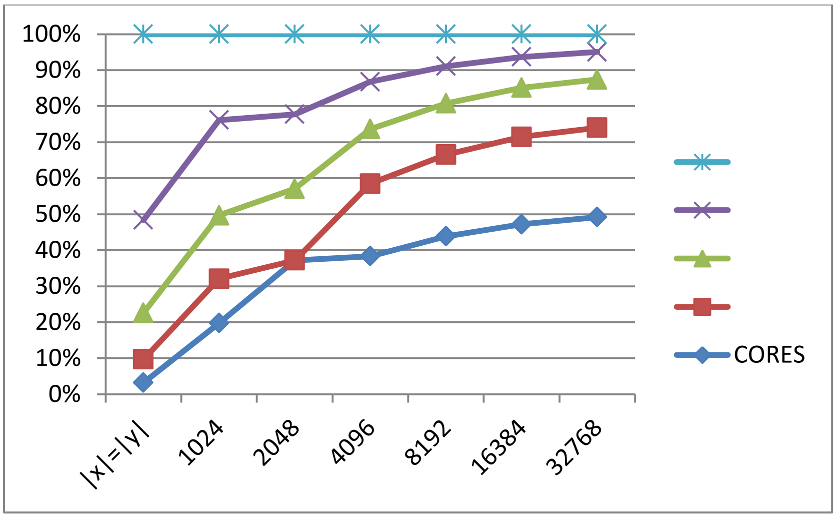
4. RESULTS & DISCUSSION

The program proceeds as participation two sequences strings in FASTA arrangement. It was used to calculate an optimal overall alignment of two strings. We have generalized usual dynamic programming algorithms grounded on Needleman-Wunsch to compare sequences. The suggested process accompaniments fast current methods for matching sequences strings. The run time reults are showed in table 1 with respect to the cores and the data size in Kbytes. The comparative results are shown in grapg 1.

Run Time in Seconds

	CORES				
|x|=|y|	1	2	4	8	16
1024	0.044	0.02743	0.03916	0.05893	0.05313
2048	0.188280	0.1.1960	0.100859	0.104743	0.112706
4096	0.748237	0.393671	0.295803	0.256242	0.258605
8192	2.981053	1.541502	0.965154	0.703047	0.604804
16384	11.85034	6.093001	3.429063	2.149426	1.58627
32768	46.2913	23.37108	12.57231	7.231707	4.646397

Table 1: Shows the results of parallel program execution time



Graph 1: Shows the performance in number of cores

The matrices are global and are accessible to the CPUs on a Grid. While implementing the above parallel Needleman-Wunsch algorithm using Alchemi framework [14], we faced the problem of increased network traffic. For small size of matrix it is not significant. However with typical sizes of DNA sequences the network traffic overhead has to be reduced. To handle these problems two formulas as under were used:

No. of Threads = Cell (Number of values in the present diagonal / Threshold [Higher limit]

Wherever Threshold is the range of values as of which we choice the number of values to be resolved per thread.

Workload = Ceil (Number of values in the present diagonal / Number of Threads)

Load is the number of values to be resolved per thread.

Sessions

Meant for every new diagonal a newfangled session is generated. Every period consists of one or added threads dependent on the length or measurement of the diagonal and the threshold (range of values from which the workload is chosen with the help of formulae). Each new session is dependent on the result of its previous session. As long as the threads of the session are running, new session cannot be created.

Threads

A thread [13] is assigned a certain workload with respect to the number of values in the current diagonal and the threshold. All the threads that belong to the same session are totally independent of each other and thus can be solved in a parallel fashion.

5. CONCLUSION

By developing Needleman's parallel algorithm we have reduced the calculation time from O(NxM) to O(N+M) [12]. The Alchemi framework for grid computing has been used to demonstrate the usefulness of the concept for large sequences like DNA. Initialization steps are already parallel in this new algorithm and a slight change in algorithm can enhance initiation to double of current speed provided more processors are available.

Preparation for parallel calculation step also indicates that more CPUs means more calculations to be performed in parallel as in the above example, eight CPUs are required to obtain the best of this algorithm.

- In the current project the computational time has been decreased drastically.
- Allowing larger sequences to align.
- It works robustly.

The result shows effectively when compared to single architecture

REFERENCES

1. Krishna N. and Akshay L. and Dr. Rajkumar B., 2002, Alchemi v0.6.1 Documentation [online], University of Melbourne.
2. About the Human Genome Project [online], Oak Ridge National Laboratory.
3. The Science behind the Human Genome Project [online], Oak Ridge National Laboratory.
4. Facts about Genome Sequencing [online], Oak Ridge National Laboratory.
5. FASTA Format Description [online], NGFN-BLAST by Nationale Genomforschungsnetz. Available: http://ngfnblast.gbf.de/docs/fasta.html
6. Source of DNA Sequences [online], National Center for Biotechnology Information.
7. Pairwise Sequence Comparison [online], Lab of Bioinformatics, Institute of Computing Technology (ICT), Chinese Academia of Sciences (CAS).
8. Introduction to Bioinformatics - Chapter 5 - Introductory Sequence Analysis [online], Human Genome Mapping Project Resource Centre (HGMP-RC) by UK Medical Research Council. %20sequence%20alignment
9. Rong X, Jan 2003, Pairwise Alignment - CS262 - Lecture 1 Notes [online], Stanford University.
10. Bin Wang, 2002, Implementation of a dynamic programming algorithm for DNA Sequence alignment on the Cell Matrix Architecture [online], Utah State University, Logan, Utah.
11. Chand T. John, April 2004, CS273: Algorithms for Structure and Motion in Biology, Stanford University.
12. DNA Sequence Comparison [online], The BioWall by Swiss Federal Institute of Technology in Lausanne (EPFL).
13. Krishna N. and Akshay L. and Dr. Rajkumar B., 2002, Alchemi v0.6.1 Documentation [online], University of Melbourne.
14. Krishna N. and Akshay L. and Dr. Rajkumar B., 2002, Alchemi v0.6.1 Documentation [online], University of Melbourne.
15. BioInformatics Educational Resources Documentation [online], European Bioinformatics Institute United Kingdom. Chitta Baral, Computational Molecular Biology, CSE 591 Arizona State University, United States of America.

