International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 02 | Feb-2016 www.irjet.net p-ISSN: 2395-0072

"AN OPTIMIZED PARALLEL ALGORITHM FOR LONGEST COMMON SUBSEQUENCE USING OPENMP" – A Review

[1]Hanok Palaskar, [2]Prof. Tausif Diwan

[1] M.Tech Student, CSE Department, Shri Ramdeobaba College of Engineering and Management, Nagpur, India
[2]Assistant Professor, CSE Department, Shri Ramdeobaba College of Engineering and Management, Nagpur, India

---

Abstract - *The LCS problem is to find the maximum length common subsequence of two or more given sequences. Finding the Longest Common Subsequence has many applications in the areas of bioinformatics and computational genomics. LCS problem has optimal substructure and overlapping sub problems, problems with such properties can be approached by a dynamic programming problem solving technique. Due to growth of database sizes of biological sequences, parallel algorithms are the best solution to solve these large size problems in less amount of time than sequential algorithms. In this paper we have carried out a brief survey of different parallel algorithms and approaches to parallelize LCS problem on the multi-core CPUs and as well as on GPUs. We have also proposed our optimized parallel algorithm to solve LCS problem on multi-core CPUs using a tool OpenMP.*

Key Words: *LCS, Dynamic Programming, Parallel Algorithm, OpenMP.*

1. INTRODUCTION

One of the classical problems in computer science is the longest common subsequence. In LCS problem we are given two sequences A = (a_1, a_2,….a_m) and B = (b_1,b_2,….b_n) and wish to find a maximum length common subsequence of A and B. By using the Dynamic Programming technique LCS can be solved in O(mn) time. Dynamic Programming algorithms recursively break the problem up into overlapping sub problems, and store the answer to the sub problems for later reference. If there is an enough overlapping of sub problems, then the time complexity can be reduced drastically, typically from exponential to polynomial. LCS has the application in many areas, such as speech recognition, file comparison, and especially bioinformatics.

Most common studies in the bioinformatics field have evolved towards a more large scale, for example, analysis and study of genome/proteome instead of a single gene/protein. Hence, it becomes more and more difficult to achieve these analyses using classical sequential algorithms on a single computer. The bioinformatics requires now parallel algorithms for the massive computation for their analysis. Parallel algorithms, are different from a traditional serial algorithms, and can be executed a piece at a time on many different processing devices, and at the end to get the correct result can be combined together.

Due to the spread of multicore machines and multithreading processors in the marketplace, we can create parallel programs for uniprocessor computers also, and can be used to solve large scale instances problems like LCS. One of the best tools to do parallel processing on multi-core CPUs is OpenMP, which is a shared-memory application programming interface (API) and can be used to describe how the work is to be shared among threads that will execute on different processors or cores.

2. THE LONGEST COMMON SUBSEQUENCE PROBLEM

The deduction of longest common subsequence of two or more sequences is a current problem in the domain of bioinformatics, pattern matching and data mining. The deduction of these subsequences is frequently used as a technique of comparison in order to get the similarity degree between two or more sequences.

2.1 Definition

A sequence is a finite set of characters or symbols. If P = <a_1,a_2, ..,a_n> is a sequence, where a_1,a_2, ..,a_n are characters, the integer n is the magnitude of P. A sequence Q = <b_1,b_2, ..,b_n> is a subsequence of P = <a_1,a_2, ..,a_n> if there are integers i_1, i_2, .., i_m($1 \le i_1 < i_2 < .. < i_m \le n$) where $b_k = a_{ik}$ for $k \in [1,m]$.
For example, X = <B,C,D,E> is a subsequence of Y = <A,B,C,D,E,F>. A sequence W is a common

subsequence to sequences X and Y if W is a subsequence of X and of Y. A common subsequence is largest subsequence if it is having maximum length. For example: sequences <C,D,C,B> and <C,E,B,C> are the longest common subsequences of <B,C,D,C,E,B,C> and of <C,E,D,B,C,B>.

2.2 Score Matrix

A classical approach for solving the LCS problem is the dynamic programming. This technique is based on the filling of a score matrix by using a scoring mechanism. The last calculated score is the length of the LCS and by tracing back the table we can get the subsequence.
Consider n and m be the lengths of two strings which are to be compared. We determine the length of a largest common subsequence in X = $<x_1,x_2, ..,x_n>$ and Y = $<y_1,y_2, ..,y_m>$.
We find L(i, j) the length of a largest common subsequence in $<a_1,a_2, ..a_i>$ and $<b_1,b_2, ..,b_j>(0 \le j \le m, 0 \le i \le n)$.

$$L(i,j) = \begin{cases} 0 \text{ if } i=0 \text{ or } j=0 \\ L(i-1,j-1) + 1 \text{ if } a_i = b_j \\ \text{Max}(L(i,j-1), L(i-1,j)) \text{ else} \end{cases}$$

We use the above scoring function in order to fill the matrix row by row (fig. 1).

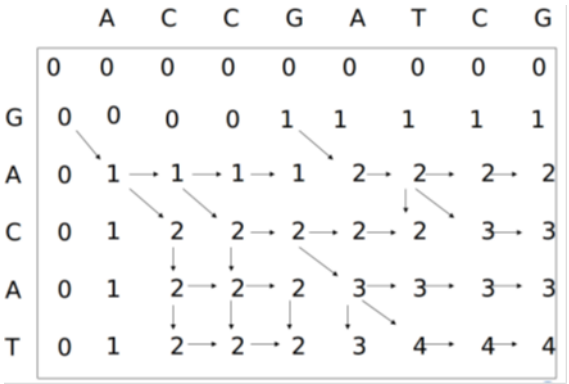

Fig. 1: Example of Filling LCS Score Matrix

The length of the LCS is the highest calculated score in the score matrix. In Fig. 1, the length is 4. To find the LCS we trace back from the highest score point (4) to the score 1 in the score matrix.

3. LITERATURE SURVEY

To find an optimized solution, lot of research is going on for over thirty years for the LCS problem. The solutions are based on Dynamic Programming, Divide-and-conquer, or Dominant Point technique etc. also, many attempts has been made to parallelize the existing algorithms. This section briefly explains the techniques and approaches used by the various authors to parallelize the LCS problem in order to get the optimized solution.

- **Amine Dhraief, Raik Issaoui and Abdelfettah Belghith in "Parallel Computing the Longest Common Subsequence (LCS) on GPUs: Efficiency and Language Suitability"** focused on parallelization of LCS problem using Dynamic Programming approach. They have presented parallelization approach for solving LCS problem on GPU, and implemented their proposed algorithm on NVIDIA platform using CUDA and OpenCL. To parallelize their algorithm they have computed the score matrix in the anti-diagonal direction instead of row wise. They have also implemented their proposed algorithm on CPU using OpenMP API. Their algorithm achieves good speedup on GPU than CPU, and for their proposed algorithm CUDA is more suitable, for NVIDIA platform, than OpenCL[1].

- **Quingguo Wang, Dmitry Korkin, and Yi Shang in "A Fast Multiple Longest Common Subsequnce (MLCS) Algorithm"** presented an algorithm for general case of Multiple LCS and its parallelization. Their algorithm is based on dominant point approach and a fast divide-and-conquer technique is used to compute the dominant points. The parallelization of the algorithm is carried out using multiple processors having one master processor and other as slaves. Master processor starts the divide-and-conquer algorithm, splits the dominant points and assigns them evenly to two children processor. The program is recursively executed at the children processor to form binary tree based on parent-children relationship. This algorithm shows asymptotically linear speed-up with respect to sequential algorithm [2].

- **Amit Shukla and Suneeta Agrawal in "A Relative Position based Algorithm to find out the Longest Common Subsequence from Multiple Biological Sequnces"** proposed a parallel algorithm for LCS based on the calculation of the relative positions of the characters from any number of given sequences. The speed-up in this algorithm has been achieved by recognizing and rejecting all those subsequences which cannot generate the next character of the LCS. For this algorithm it is required to have the number of characters being used in the sequences in advance. Parallelization approach of this algorithm uses multiple processors where number of processors is equal to the number of characters in the finite symbol set. Calculations are done at each processor and the results are stored at local memories of each processer which are then combined to get the final LCS. The complexity for the sequential algorithm is O(n) where n is length of the sequence and the complexity for the parallel algorithm is O(k) where k is the slowest processor. Complexity for the parallel algorithm is independent of the number of sequences [3].

- **R. Devika Ruby and Dr. L. Arockiam in "Positional LCS: A position based algorithm to find Longest Common Subsequence (LCS) in Sequence Database (SDB)"** in their paper presented a position based algorithm for LCS which is useful in Sequence Database Applications. Their proposed algorithm focuses only on matched position, instead of unmatched positions of sequences, to get LCS. Primary idea for their algorithm is to remove backtracking time by storing only the matched position, where LCS occurs. Also to reduce the time complexity, instead of searching entire score matrix, matched position array is used. In their proposed algorithm score matrix is computed for entire sequences, but to find the LCS their algorithm scans only the matched positions. Time complexity of their proposed algorithm is reduced to half than time complexity of dynamic LCS [4].

- **Jaimei Liu and Suping Wu in "Research on Longest Common Subsequence Fast Algorithm"** proposed a fast algorithm for LCS, for two sequences having length 'm' and 'n', by transforming the problem of LCS into solving the problem of matrix m[p,m] (where p< min(m,n)). They have also presented the parallelization of their proposed algorithm based on OpenMP. For optimizing computation of each element in the score matrix, they have used their proposed theorems. To find the LCS instead of backtracking the score matrix, they have used a simple formula which gives the required LCS in constant time. Time complexity of their proposed algorithm is O(np) and the space complexity is O(m+n). They have realized the parallelization of their proposed algorithm by using OpenMP constructs on the nested outer loops [5].

4. PROPOSED WORK

We propose the new optimized parallel LCS algorithm. Major factor in finding the LCS is the computation of score matrix hence we will optimize the calculation of elements in the score matrix by using theorems, instead of classical method. We will implement our parallel algorithm on the multi-core CPUs using OpenMP API constructs. To increase the performance of our parallel in terms of speed and memory optimization we will divide the load among the threads by applying load balancing techniques and cache optimization respectively. We expect that our proposed algorithm will be faster than the existing Parallel LCS algorithms. In future we will expand our algorithm, to support the Multiple Longest Common Subsequence (MLCS) and also to make the hybrid version of our algorithm by combining OpenMP and MPI.

5. CONCLUSION

Problem of LCS have the variety of applications in the domain of pattern matching, data mining and bioinformatics. Due to the recent developments in the multi-core CPUs, parallel algorithms using OpenMP are one of the best ways to solve the problems having large size inputs. This paper presented a review of parallel algorithm for the Longest Common Subsequence problem and approaches to parallelize LCS problem on the multi-core CPUs and as well as on GPUs. We also have proposed our parallel algorithm and the optimizations in order to increase the performance, which we expect to be the faster algorithm in comparison to the existing parallel algorithms for solving LCS.

6. REFERENCES

[1] Amine Dhraief, Raik Issaoui, Abdelfettah Belghith, "Parallel Computing the Longest Common Subsequence (LCS) on GPUs: Efficiency and Language Suitability", The First International Conference on Advanced Communications and Computation, 2011.

[2] Quingguo Wang, Dmitry Korkin, Yi Shang, "A Fast Multiple Longest Common Subsequnce (MLCS) Algorithm",IEEE transaction on knowledge and data engineering, 2011.

[3] Amit Shukla, Suneeta Agrawal, "A Relative Position based Algorithm to find out the Longest Common Subsequence from Multiple Biological Sequnces ", 2010 International Conference on Computer and Communication Technology, pages 496 – 502.

[4]R. Devika Ruby, Dr. L. Arockiam, "Positional LCS: A position based algorithm to find Longest Common Subsequence (LCS) in Sequence Database (SDB)", IEEE International Conference on Computational Intelligence and Computing Research, 2012.

[5] Jiamei Liu, Suping Wu "Research on Longest Common Subsequence Fast Algorithm", 2011 International Conference on Consumer Electronics, Communications and Networks, pages 4338 – 4341.

[6] Zhong Zheng, Xuhao Chen, Zhiying Wang, Li Shen, Jaiwen Li "Performance Model for OpenMP Parallelized Loops ", 2011 International Conference on Transportation, Mechanical and Electrical Engineering (TMEE), pages 383-387.

[7] Rahim Khan, Mushtaq Ahmad, Muhammad Zakarya, "Longest Common Subsequence Based Algorithm for Measuring Similarity Between Time Series: A New Approach" World Applied Sciences Journal, pages 1192-1198.

[8] Jiaoyun Yang, Yun Xu,"A Space-Bounded Anytime Algorithm for the Multiple Longest Common Subsequence Problem", IEEE transaction on knowledge and data engineering, 2014.

[9] I-Hsuan Yang, Chien-Pin Huang, Kun-Mao Chao, "A fast algorithm for computing a longest common increasing subsequence", Information Processing Letters, ELSEVIER, 2004.

[10] Yu Haiying, Zhao Junlan, Application of Longest Common Subsequence Algorithm in Similarity Measurement of Program Source Codes. Journal of Inner Mongolia University, vol. 39, pp. 225–229, Mar 2008.

[11] Krste Asanovic, Ras Bodik, Bryan, Joseph, Parry, Samuel Williams, "The Landscape of Parallel Computing Research: A view from Berkeley" Electrical Engineering and Computer Sciences University of California at Berkeley, December 2006.

[12] Barbara Champman, Gabriel Jost, Ruud Van Der Pas "Using OpenMP", 1-123