4005-735 Parallel Computing I
Project Proposal:
Longest Common Subsequence

Adam Risi
Rochester Institute of Technology
ajr7708@rit.edu

Richard Sarkis
Rochester Institute of Technology
res7008@rit.edu

December 13th, 2010

1 Computational Problem

The Longest Common Subsequence (LCS) problem is common to bioinformatics and a classic in computer science. The problem is one in which given two sequences $X =< x_1, x_2, \cdots, x_m >$ and $Y =< y_1, y_2, \cdots, y_n >$ we attempt to find the maximum-length common subsequences of X and Y. A subsequence is not the same as a substring (subsequences do not need to be consecutive in the string). Given the string $X =< x_1, x_2, \cdots, x_m >$, string $Z =< z_1, z_2, \cdots, z_k >$ is then a subsequence of X if there is a strictly increasing sequence $< i_1, i_2, \cdots, i_k >$ of indices of X such that for all $j = 1, 2, \cdots, k$ we have $x_{i_j} = z_j$ [2].

The problem is commonly encountered in bioinformatics in the comparison of DNA which is represented by the four DNA bases adenine, guanine, cytosine, and thymine represented relatively by the letters A, C, G, T. This results in strings that are a combination of these letters representing an organism's DNA sequence. These sequences are typically very long in length resulting in a computationally intensive problem.

The problem is generally considered NP-hard problem with an arbitrary number of input sequences but with a constant number of inputs the solution time is polynomial through the use of dynamic programming [1].

Parallelizing the LCS problem results in great increases in speed in the decoding of the genetic-derived character strings being compared. By determining the longest common subsequence of DNA, biologists can determine the relative closeness of two organisms.

The goal of this project is to determine the speed-up of this problem when using parallelization techniques and compare these performance metrics with a sequential single-CPU experiment.

2 Code Deliverables

The project will involve two sets of deliverables. The first is a sequential code set and the second is a parallelized version of the sequential code.

2.1 Sequential

We will investigate the single-processor, single-threaded version of an LCS algorithm, whether it is based off of methods like memoization, recursive,

Hirschberg's algorithm or dynamic programming or another algorithm yet to be determined through our research. Benchmarks using metrics described in section 3 will be recorded for the chosen sequential algorithm. This code will be written in C.

2.2 Parallelized

Research into the problem will help us determine the best LCS algorithm for parallelization. A modification of one of the algorithms chosen in the sequential portion of the project would be preferred for the most accurate analysis. The code will be written in C and parallelized using MPI (either OpenMPI or MPICH). Again, metrics described in section 3 will be used for our analysis of the chosen algorithm in its sequential and parallelized forms. It's not is not yet determined if we will investigate a pure MPI solution or consider a hybrid approach using MPI and PThreads.

3 Performance Metrics

With regard to the sequential algorithm we will likely determine basic metrics such as running time versus the size of the input strings (both chosen to be equal in length for simplicity).

We'll calculate a relative speed-up ratio where the running time of the sequential version (i.e. our LCS algorithm running on one processor) is divided by the running time of the parallelized version on the cluster. As we increase the length of the input sequences the growth of the relative speedup will be measured. Factors to be considered in the parallelized MPI version is the impact of communication between nodes and the overall rate of growth of the speed-up ratio.

Bibliography

[1] Longest common subsequence problem. *Wikipedia*. http://goo.gl/5bMjT.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001.

[3] Jinxian Lin and Yiqing Lv;. Computing all longest common subsequences on mpi cluster. *Computational Intelligence and Natural Computing Proceedings (CINC), 2010 Second International Conference on*, 1:386 – 389, 2010.

[4] Wei Liu and Ling Chen;. A parallel algorithm for solving lcs of multiple bioseqences. *Machine Learning and Cybernetics, 2006 International Conference on*, pages 4316 – 4321, 2006.

[5] B Ben Mabrouk, H Hasni, and Z Mahjoub. Parallelization of the dynamic programming algorithm for solving the longest common subsequence problem. *Computer Systems and Applications (AICCSA), 2010 IEEE/ACS International Conference on*, pages 1 – 8, 2010.

[6] Nuraini Rashid, Rosni Abdullah, and Abdullah Talib. Parallel homologous search with hirschberg algorithm: a hybrid mpi-pthreads solution. *ICCOMP'07: Proceedings of the 11th WSEAS International Conference on Computers*, Jul 2007.