Project Proposal

Laura Wieme and Karl Voelker

December 14, 2009

1 Problem Description

The goal of the longest common subsequence (LCS) problem is to find the longest subsequence that is common to each member of a given set of sequences. LCS is useful for making comparisons and finding differences. The LCS problem is NP-hard for an arbitrary number of inputs, but is $O(n^2)$ for two inputs, which is the problem we will solve, using an adaptation of the traditional dynamic programming algorithm. Since a problem that can be solved by dynamic programming tends to easily be broken down into subproblems, the longest common subsequence problem will work well as a parallel program. [3] [2] [1]

2 Programs

Both the sequential and parallel programs will accept the same input and produce the same output. The input will be two lines of text which are the two sequences for which to find the LCS. The output, on standard output, will be a single line consisting of an LCS of the two inputs. The output, on standard error, will be a textually-represented integer giving the number of milliseconds during which the program ran.

2.1 Sequential Program

For each pair of input strings, x and y, a matrix C will be filled in. The dimensions of C are $(|x|+1, |y|+1)$. All cells $C_{0,j}$ and $C_{i,0}$ are initialized to 0. The remaining cells are filled in by diagonals which parallel the one from $C_{|x|,0}$ to $C_{0,|y|}$, with the first such diagonal consisting only of $C_{1,1}$ and the last only of $C_{|x|,|y|}$. Since this program will be sequential, each cell will be individually calculated before moving on to the next.

The sequential program will be run with the command `java LcsSeq < in.txt > out.txt`.

2.2 Parallel Program

For each pair of input strings, x and y, a matrix C will be filled in. The dimensions of C are $(|x|+1, |y|+1)$. All cells $C_{0,j}$ and $C_{i,0}$ are initialized to 0.

The remaining cells are filled in by diagonals which parallel the one from $C_{|x|,0}$ to $C_{0,|y|}$, with the first such diagonal consisting only of $C_{1,1}$ and the last only of $C_{|x|,|y|}$. The cells of any such diagonal depend only on the cells of the previous two diagonals, so all the cells of a diagonal can be calculated in parallel.

The parallel program will be run with the command `java LcsSmp < in.txt > out.txt`.

3 Metrics

For two input strings x and y, the problem size $N = |x| \cdot |y|$. Let $T_{seq}(N, K)$ and $T_{par}(N, K)$ be the sequential and parallel running times for a problem of size N on K processors, and let $N_{seq}(T, K)$ and $N_{par}(T, K)$ be the sequential and parallel problem sizes that can be solved with a running time of T on K processors. The metrics we will report are:

- $Speedup(N, K) = \frac{T_{seq}(N,1)}{T_{par}(N,K)}$
- $Efficiency(N, K) = \frac{Speedup(N,K)}{K}$
- $Sizeup(T, K) = \frac{N_{par}(T,K)}{N_{seq}(T,1)}$
- $SizeupEfficiency(T, K) = \frac{Sizeup(T,K)}{K}$
- $EDSF(N, K) = \frac{K \cdot T_{par}(N,K) - T_{par}(N,1)}{K \cdot T_{par}(N,1) - T_{par}(N,1)}$, the experimentally-determined sequential fraction

References

[1] L. Bergroth, H. Hakonen, and T. Raita. A survey of longest common subsequence algorithms. In *SPIRE '00: Proceedings of the Seventh International Symposium on String Processing Information Retrieval (SPIRE'00)*, page 39, Washington, DC, USA, 2000. IEEE Computer Society.

[2] Brenda Hinkemeyer and Bryant A. Julstrom. A genetic algorithm for the longest common subsequence problem. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 609–610, New York, NY, USA, 2006. ACM. `doi:http://doi.acm.org/10.1145/1143997.1144105`.

[3] Wei Liu, Ling Chen, and Lingjun Zou. A parallel lcs algorithm for biosequences alignment. In *InfoScale '07: Proceedings of the 2nd international conference on Scalable information systems*, pages 1–8, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).