Calculating Genomic Distances in Parallel Using OpenMP

Vijaya Smitha Kolli[1], Hui Liu[1], Jieyue He[1, 2], Michelle Hong Pan[3], and Yi Pan[1]

[1] Department of Computer Science, Georgia State University, Atlanta, GA 30303, USA
hui.anitaliu@gmail.com, pan@cs.gsu.edu
[2] Department of Computer Science, Southeast University, Nanjing, Jiangsu, China
[3] Centers for Disease Control and Prevention, Office of Workforce and Career Development, Career Development Division, Public Health Informatics Fellow Program, 1600 Clifton Rd, Atlanta, GA 30333, USA
hdp1@cdc.gov

Abstract. By finding the corresponding shortest edit distance between two signed gene permutations, we can know the smallest number of insertions, deletions, and inversions required to change on string of genes into another, where insertion, deletion and inversion are the process of genome evolutions. However, it is NP-hard problem to compute the edit distance between two genomes. Marron et al proposed a polynomial-time approximation algorithm to compute (near) minimum edit distances under inversions, deletions, and unrestricted insertions. Our work is based on Marron's et al algorithm, which carries out lots of comparisons and sorting to calculate the edit distance. These comparisons and sorting are extremely time-consuming, and they result in the decrease of the efficiency. We believe the efficiency of the algorithm can be improved by parallelizing. We parallelize their algorithm via OpenMP on Intel C++ compiler for Linux 7.1, and compare three levels of parallelism: coarse grain, fine grain and combination of both. The experiments are conducted for a varying number of threads and length of the gene sequences. The experimental results have shown that either coarse grain parallelism or fine grain parallelism alone does not improve the performance of the algorithm very much, however, the combination of both fine grain and coarse grain parallelism have improve the performance to a great extent.

1 Introduction

As the need for comparing genomes of different species has grown dramatically with the fast progress of the Human Genome Project, the evolution at the level of whole genomes has attracted more and more attentions from both biologists and computer scientists. They are especially interested in the scenarios in which the genome evolves through insertions, deletions, and movements of genes along its chromosomes.

A gene is the fundamental physical and functional unit of heredity. Each chromosome can be represented by an ordering of signed genes. The gene orders can be rearranged via evolutionary events like inversions and transpositions. The motivation of studying the gene sequencing arises in molecular biology. The ability to

C. Priami, A. Zelikovsky (Eds.): Trans. on Comput. Syst. Biol. II, LNBI 3680, pp. 113 – 123, 2005.

compare genomes of different species has grown considerably with the rapid advancement of Human Genome Project, genetic and DNA data on different species. One of the most effective methods of finding the similarity between genomes is to compare the order of appearance of identical genes in the two species. By finding the corresponding shortest edit distance between two signed gene permutations, we can know the smallest number of insertions, deletions, and inversions required to change one string of genes into another. However, it is very difficult to compute the edit distance between two genomes; for example, this problem is NP-hard for unsigned permutations even with equal gene content and only inversion allowed.

Many researchers have proposed various algorithms on finding the minimum edit distances [3,4,5,6,10]. Most of these algorithms involve lots of comparisons and sorting while computing edit distances. Marron et al's algorithm [2] is of particular interest for our implementation purposes. This algorithm handles duplications as well as insertions and presents an alternate framework for computing (near) minimal edit sequences involving insertions, deletions, and inversions. They produce a new canonical form in which the shortest edit sequences can be transformed into equivalent sequences of equal length in which all insertions are performed first, following by all inversions, and then by all deletions.

Marron et al's algorithm is implemented sequentially, which is not efficient in terms of computation time, because the algorithm carries out many comparisons and sorting. Parallelism can be employed in the time-consuming comparisons and sorting, thus increasing the efficiency of the algorithm. We have performed profiling on the whole algorithm and identified the functions that are utilizing maximum time. We parallelize the algorithm through OpenMP on Intel® C++ for Linux compiler 7.1. The Intel® Compiler provides optimization technology, threaded application support, features to take advantage of Hyper-Threading technology. At the same time it produces optimal performance for the applications [9]. OpenMP, the industry standard for portable multi-threaded application development, is powerful at fine grain (loop level) and large grain (function level) threading. The Intel C++ Compiler supports OpenMP API version 2.0 and performs code transformation for shared memory parallel programming [9].

The rest of this paper proceeds as follows. Section 2 provides preliminaries on the sequential algorithm by Marron et al. We illustrate the details of our parallel implementation in Section 3. The experimental results are analyzed in Section 4. Section 5 draws the conclusion and proposes future works.

2 Preliminary Existing Algorithm

Marron et al's algorithm [2] is based on a new canonical form for edit sequences. They show that shortest edit sequences can be transformed into equivalent sequences of equal length in which all insertions are performed first, followed by all inversions, and then by all deletions. This canonical form allows taking advantage of El-Mabrouk's exact algorithm for inversions and deletions, which can be extended by finding the best possible prefix of insertions, and producing an approximate solution with bounded error.