A data-flow modification of the MUSCLE algorithm for multiprocessors and a web interface for it

Alexey N. Salnikov (salnikov@cs.msu.su)
Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University, 2-nd educational building, Leninskie Gory, Moscow, 119992, Russia

AbstractNucleotide and amino acid sequences research is actual for molecular biology and bioengineering. An important aspect of analysis of such sequences is multiple alignment. This article describes the implementations of the MUSCLE and ClustalW programs on multiprocessors and a web interface to them. The modification of the MUSCLE algorithm realize a data-flow manner of sequence alignment. It uses the PARUS system to build a data-flow graph and execute it on one multiprocessor. The data-flow algorithm has been tested on the sequences of human Long Terminal Repeats class five (LTR5) and several other examples.

Keywords. sequence alignment, data-flow, clusters

Introduction

Nucleotide and amino acid sequences research is actual for molecular biology and bioengineering. An important aspect of analysis of such sequences is multiple alignment. Its purpose is to arrange the sequences of DNA, RNA, or protein to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences. Aligned sequences are typically represented as rows within a matrix. Gaps are inserted between the residues so that residues with identical or similar characters are aligned in successive columns. There are several algorithms of multiple sequence alignment: ClustalW [1], MUSCLE [3], DIALIGN-TX [7]. The most popular method for multiple sequence alignment of nucleotides and amino acids is ClustalW. But ClustalW has several disadvanteges and a new algorithm MUSCLE that tries to solve these disadvantages was developed. But on a large number of sequences or on long sequences the MUSCLE algorithm, originally non-parallel, can consume rather long time for the execution — up to several days, and can demand a large amount of memory. It should be noted that several algorithms has parallel implementation (for example: ClustalW-MPI [2], DIALIGN P [8]).

For the pairwise alignment the dynamic programming method allows to get the precise result in a reasonable time, but for the multiple alignment the time complexity of the multidimensional analog of dynamic programming is an exponential function with respect to the number of sequences. Usually heuristic algorithms are used, which do not give precise solution in the terms of optimization of some score function, but have a less

complexity and give an eligible result in the context of biology. We will consider the algorithm described in the article [3]. It is realized in the MUSCLE software product available at http://www.drive5.com/muscle.

The web interface has been developed to facilitate availability of multiprocessor for researcher. The web interface requires the task creation by reseacher. Earch task binds the source data represented as a sequences in FASTA format, algorithm (ClustalW-MPI, modified MUSCLE) and multiprocessor recources.

1. The MUSCLE algorithm

The MUSCLE algorithm is divided into several stages.

On the first stage the algorithm counts the degrees of similarity between all aligned sequnces and aggregates them into a matrix. Then the sequences are clustered by their similarities, and a binary cluster tree is built from the similarity matrix using the UPGMA[4] algorithm.

The similarity degree of sequences included in alignment on the first stage is calculated by the following method: all sequences are divided into sets of fragments with the length of k (k-mers), then the occurrence of each k-mer in all the sequences is calculated. Similarity calculation uses this formula:

$$similarity(S_i, S_j) = \sum_{m=1}^{M} \frac{min(frequency(S_i, \tau_m), frequency(S_j, \tau_m))}{min(length(S_i), length(S_j)) - k + 1}$$

where τ_m is one of the k-mers distinguished from the sequences, $frequency(S_i, \tau_m)$ is the number of times τ_m was met in sequence S_i, M — the number of all different revealed k-mers for all sequences in alignment.

After the similarity counting, the alignments for the pairs of leaves having a common parent in the tree are built. The alignments for the pairs of sequences are built using the dynamical programming method. Then we get subalignments (alignments of sequences subset) for the all interior tree nodes that is done by pairwise alignment of two subalignments. The sequences that are involved in such subalignment cannot be shifted, so the gaps are inserted simultaneously into all lines of subalignment.

On the second stage the tree is refined using the Kimura distance [3], a more accurate metric than the k-mer distance used on the first stage. Then the aligment is rebuilt by the method similar the first stage. Practically, algorithm needs to realign the subalignments that corresponds to the vertices which have been changed in tree.

The third stage is iterative. The alignment is “refined”.

2. The data-flow algorithm for multiple sequence alignment

For using the MUSCLE algorithm on multiprocessor systems, the original method has been modified on the stage of building subalignments using the cluster tree. We build a parallel program as a graph-program by means of PARUS [6](PARUS is a parallel programming language that allows to build parallel programs in a data-flow graph notation (http://parus.sf.net)).

Our algorithm is split into several stages. On the first stage we use the original MUSCLE algorithm to count similarities between sequences and to build the cluster tree. On the second stage we build a graph-program. The third stage concentrates on multiprocessor execution of the graph-program.

The graph-program is built from the cluster tree in the following way: each source vertex in a PARUS graph-program corresponds to one or several aligning sequences. The graph-program edges are directed from the source vertices to the inner ones. Inner vertices align pairs of subalignments to create a new subalignment that can be sent to a next inner vertex or to a drain vertex of the graph-program. The algorithm goes to the drain node when the number of sequences in the current subalignment of global alignment is equal to the whole number of sequences to be aligned.

Therefore, the graph-program is built from the leaves of cluster tree to the root. To improve the efficiency of the parallel program the bottom (closer to leaves) layers of the cluster tree are compressed into one layer of graph-program. To do that, a new parameter is introduced into the algorithm. It defines a number of layers of the cluster tree to be compressed. As a result of the compression, the source vertices of the graph-program can contain more than one sequence, unlike the leaves in the cluster tree.

The time of program execution for the modified MUSCLE algorithm is reduced due to the possibility to execute each node of graph program on a different processor. The maximal number of concurrently handled vertices is limited by the number of vertices in one layer of the graph-program and the number of processors accessible for running the parallel program.

3. Web interface

The web interface has been developed to bind needs of the researcher to align the sequences in a shortest time with a multiprocessor and job scheduling system on it (for example LoadLeveler). So, there has been introduced an abstraction of task. For each task, the reseacher must chose one of the multiprocessors, then for the chosen mutiprocessor show the required processor time and the number of processors. After marking task as ‘ready’, during one hour the source data will be transmitted to the multiprocessor and the task will be submited to the multiprocessor’s queue. The task abstraction will be implemented on multiprocessor as the MPI-application executing on many processors.

Once per hour the task on the multiprocessor is automatically checked by means of Python script. If the task finished its work, then the researcher will be notified by email about success or errors. In any case, the result on success and the temporary data on error will be transfered back to the web site.

The web interface is available at: http://angel.cs.msu.su/aligner.

4. Testing results

The data-flow algorithm has been tested on sequences of LTR’s class five in human genome. LTR (Long Terminal Repeat) is a family of so-called repeats (sequences occuring in the genome in a number of nearly identified copies); there are several classes of LTR. The class 5 (LTR5) contains approximately 1500 sequences approximately 1200

nucleotydes each. Aligning a collection (described in [5]) of all known LTR5 objects in human genome requires 1h 8m for one-processor configuration on PrimePOWER 850. The data-flow implementation with 12-processor PrimePOWER 850 requires 28 minutes, which is 2.4 times faster then the original sequential MUSCLE algorithm. On the IBM pSeries 690 the data-flow modification's running time is 24 minutes. The one-processor original MUSCLE algorithm (with imposed limit of 3 iterations) requires more than 7 hours to run. On the cluster mvs100k (cluster of 470 nodes with four Intel Xeon 5160 processors which are connected through Infiniband network) has been aligned 1088 protein sequences approximately 300 amino acids each. The data-flow modification takes: 30 seconds on 100 processors, 201 seconds on 500 processors, and 181 seconds on 16 processors. So, the actual task is to find the optimal number of processors. At the first sight effectiveness of multiprocessor implementation is not high but it should be taken into consideration that the cluster tree for all LTR5's is rather unbalanced, and the algorithm is most efficient for balanced cluster tree.

References

[1] Julie D. Thompson, Desmond G. Higgins, Toby J. Gibson CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice //Nucleic Acids Research, 1994, vol. 22 No. 22 , pp. 4673-4680. ISSN: 0305-1048 (Print), ISSN: 1362-4962 (Electronic).

[2] Kuo-Bin Li ClustalW-MPI: ClustalW analysis using distributed and parallel computing //Bioinformatics Vol. 19, No. 12, 2003, pp. 1585-1586. ISSN: 1460-2059 (Electronic),ISSN: 1367-4803 (Print)

[3] Robert C. Edgar MUSCLE: a multiple sequence alignment method with reduced time and space complexity //BMC Bioinformatics 2004, 5:113, ISSN: 1471-2105 (Electronic).

[4] P.H.A. Sneath, Robert R. Sokal Numerical Taxonomy //Nature 193, pp. 855-860 (03 March 1962), ISSN: 0028-0836, EISSN: 1476-4687.

[5] Alexeevski A.V., Lukina E.N., Salnikov A.N., Spirin S.A. Database of long terminal repeats in human genome: structure and synchronization with main genome archives //Proceedings of the fours international conference on bioinformatics of genome regulation and structure, Volume 1. BGRS 2004, pp. 28-29 Novosibirsk.

[6] Alexey N. Salnikov PARUS: A Parallel Programming Framework for Heterogeneous Multiprocessor Systems //Lecture Notes in Computer Science (LNCS 4192) Recent Advantages in Parallel Virtual Machine and Message Passing Interface, Volume 4192, pp. 408-409, 2006, ISBN-10: 3-540-39110-X ISBN-13: 978-3-540-39110-4.

[7] Amarendran R Subramanian, Michael Kaufmann and Burkhard Morgenstern DIALIGN-TX: greedy and progressive approaches for segment-based multiple sequence alignment // Algorithms for Molecular Biology, 2008, 3:6, ISSN: 1748-7188 (electronic).

[8] Martin Schmollinger, Kay Nieselt, Michael Kaufmann and Burkhard Morgenstern DIALIGN P: Fast pair-wise and multiple sequence alignment using parallel processors // BMC Bioinformatics, 2004, 5:128, ISSN: 1471-2105 (Electronic).