Parallel *CLUSTAL W* For PC Clusters

James Cheetham[1], Frank Dehne[2], Sylvain Pitre[3]
Andrew Rau-Chaplin[4], Peter J. Taillon[5]

[1] Institute of Biochemistry, Carleton University,
Ottawa ON Canada, jcheetha@ccs.carleton.ca
[2] School of Computer Science, Carleton University,
Ottawa, ON Canada, frank@dehne.net
[3] School of Computer Science, Carleton University,
Ottawa, ON Canada, spitre@scs.carleton.ca
[4] Faculty of Computer Science, Dalhousie University,
Halifax, NS Canada, arc@cs.dal.ca
[5] School of Computer Science, Carleton University,
Ottawa, ON Canada, ptaillon@scs.carleton.ca

Abstract. This paper presents a parallel version of CLUSTAL W, called *pCLUSTAL*. In contrast to the commercial SGI parallel Clustal, which requires an expensive *shared memory* SGI multiprocessor, *pCLUSTAL* can be run on a range of distributed and shared memory parallel machines, from high-end parallel multiprocessors (e.g. Sunfire 6800, IBM SP2, etc.) to PC clusters, to simple networks of workstations. We have implemented *pCLUSTAL* using C and the MPI communication library, and tested it on a PC cluster. Our experimental evaluation shows that our *pCLUSTAL* code achieves similar or better speedup on a *distributed memory* PC clusters than the commercial SGI parallel Clustal on a *shared memory* SGI multiprocessor.

Key Words: Multiple Sequence Alignment, CLUSTAL W, Parallel Computing, Computational Biochemistry.

1 Introduction

The alignment of DNA or protein sequences is by far the most common task in Bioinformatics. Procedures relying on sequence comparison are diverse and range from database searches and studies of evolution to protein structure prediction. Pairwise alignment of sequences is the most simple form of sequence alignment and is mainly used for searching sequence databases. More than two sequences can also be aligned, and this multiple sequence alignment has many uses. Sequences can be aligned along their entire length (global alignment) or only in certain regions (local alignment). Global and local alignment can be used for both pairwise and multiple alignments. Global alignments use gaps (insertions/deletions) while local alignments don't have to and only align regions between gaps (Figure 1). A global alignment compares the two sequences over their entire lengths, and is appropriate when comparing sequences that are expected to share similarity over their entire lengths. Global alignment maximizes regions of similarity and minimizes gaps using scoring matrices and gap parameters set by the user.

Protein sequences can be related by homology or convergence. Multiple sequence alignments are useful in both cases. Homologous proteins, by definition,

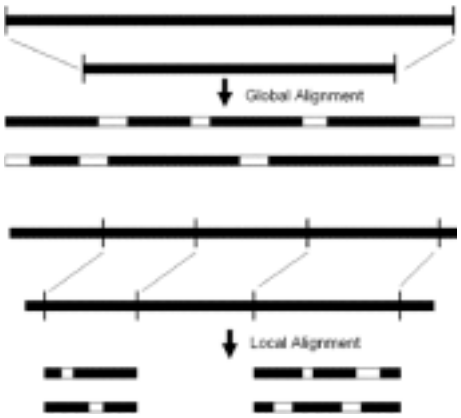

Fig. 1. A comparison of local and global sequence alignments.

have a common ancestor and usually a common function. Converged proteins have evolved independently and have similar amino acid sequences and usually similar structures and/or functions. The helix-turn-helix DNA binding domain is an example of convergence in archael, eubacterial and eukaryotic DNA binding proteins.

The alignment of novel DNA or protein sequences with well characterized homologous sequences can provide information on the potential functions of the novel sequences. A multiple alignment of protein sequences determines the position and nature of conserved regions in each member of the group. Conserved amino acid sequences usually correspond to structurally and/or functionally important parts of a protein. It is a common occurrence that we only know the structure or function for one or two members of a group of protein sequences. Constructing multiple alignments allows us to infer structures and functions for other members of the group and to generate hypotheses about the functional importance of specific sequences which can be tested experimentally.

Phylogenetic analysis provides a conceptual framework for understanding evolution but involves several computational challenges. The generation of evolutionary trees can be seen as two distinct NP-complete problems: multiple sequence alignments and phylogenetic tree searchs. Phylogenetic analysis of sequence data depends strongly on accurate multiple alignments (Figure 2). In addition, there are other problems, such as orthologs and paralogs. Orthologs are sequences derived from a common ancestor through vertical descent. In more direct terms, this means the same gene in different species. Paralogs are genes within the same genome that have been generated by duplication. Distinguishing between orthologs and paralogs is important is we hope to build accurate phylogenetic trees.

One objective of multiple sequence alignments can be the generation of fully annotated phylogenetic trees. (An example is shown in Figure 2.) This is difficult for two reasons. (1) To considers all possible multiple sequence alignments and then, all possible phylogenetic trees and pick the best one, would be impossible. Thus, most phylogenetic programs use previously aligned sequences. (2) The result will be influenced by the criteria used to determine the best tree.

There are three common types of tree building algorithms: distance matrix, maximum likelihood and parsimony. Distance matrix methods estimate pairwise distances between the sequences (which reduces the information in the alignment to a single number). Other methods build several trees from the information in the multiple alignment and select the best tree. The guide tree in CLUSTAL is derived from the distances between pairwise aligned sequences. These distances may not be equal to the distances between sequence pairs in the multiple sequence alignment.

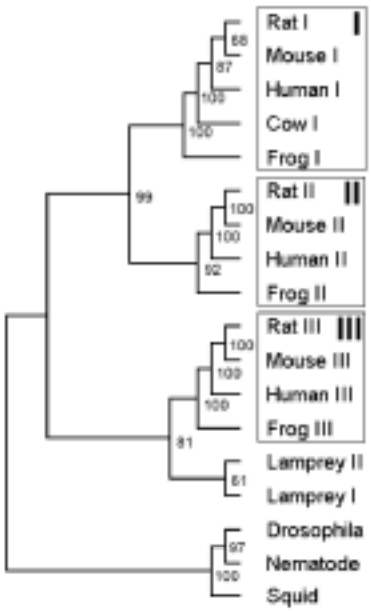

Fig. 2. An example phylogenetic tree constructed from a multiple sequence alignment. The tree was made using protein sequences from the synapsin family. Bootstrap values are percentages for 1000 trials. The tree was calculated using a multiple alignment from the ClustalW program and drawn with the TreeView program.

As the number of DNA and protein sequences in databases increases, it is increasingly important to be able to create multiple sequence alignments for very large numbers of sequences. Using standard multiple sequence alignment tools like Clustal W [10] (see Figure 3) a set of 100 sequences can be aligned in under an hour on a fast workstation. However, given that the underlying alignment algorithm requires $O(n^2)$ steps, where n is the number of sequences to be aligned, it is not surprising that these standard tools soon begin to take many hours to run.

In this paper, we describe a parallel version of Clustal W, called *pCLUSTAL*, that can be run a range of distributed memory parallel machines, from high-end parallel multiprocessors (e.g. Sunfire, IBM SP, SGI Origin, etc.), to PC clusters, to simple networks of workstations. We have implemented *pCLUSTAL* using C

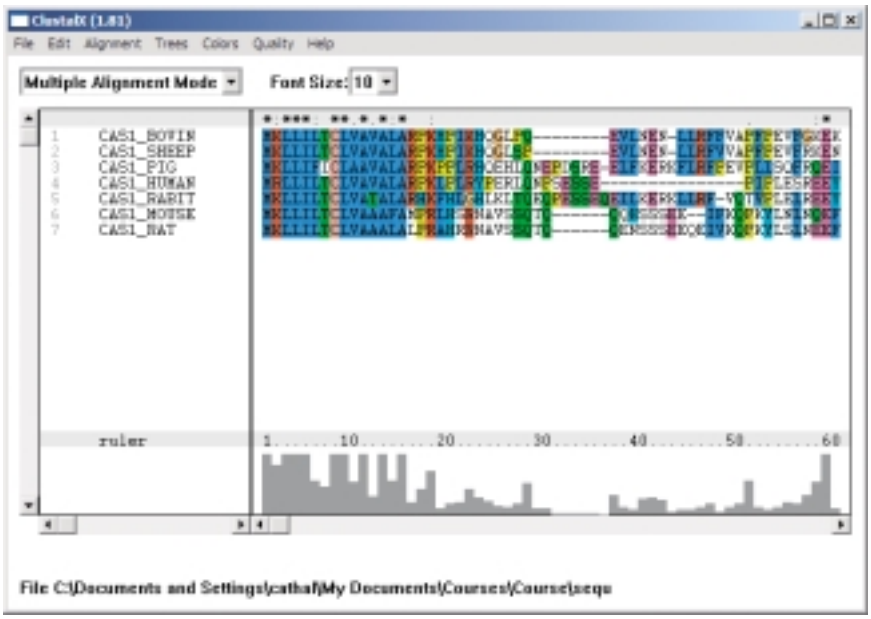

Fig. 3. CLUSTAL W.

and the MPI communication library, and tested it on a PC cluster. Our experiments, presented in Section 4, investigate the speedup and scalability of our method on gene sequences obtained from the National Center for Biotechnology Information.

A parallelization of Clustal for shared memory SGI multicomputers, like the SGI Origin 3000, was previously developed by the SGI ChemBio group [8]. Their implementation was based on SGI OpenMP shared memory compilers and runs only on *shared memory* SGI machines. They report achieving a maximum speedups of 10 on 16 processor machines [8].

The main contribution of this paper is to provide an efficient *distributed memory* implementation of CLUSTAL W that can be run on a wide range of distributed memory PC clusters and parallel multicomputers. Our experimental evaluation shows that our *pCLUSTAL* code achieves similar or better speedup on *distributed memory* PC clusters than the SGI parallel Clustal on a *shared memory* SGI Origin 3000.

The remainder of this paper is organized as follows. Section 2 reviews the multiple sequence alignment problem and sequential Clustal W algorithm. In Section 3, we present a parallelization of CLUSTAL W for distributed memory multicomputers. Section 4 presents the experimental performance results for our *pCLUSTAL* code on a PC cluster, and Section **??** concludes the paper.

2 Review: Sequential CLUSTAL W

CLUSTAL W has become the most popular algorithm for multiple sequence alignment [4, 5, 3, 10, 6, 9, 7]. This program implements a progressive method for multiple sequence alignment. A high level description of the three basic phases in the CLUSTAL W alignment algorithm are given in Algorithm 1.

Algorithm 1 *Sequential CLUSTAL W*
Input: A set S of n sequences. **Output:** A multiple alignment of S.

(1) *Pairwise alignment*: Compute pairwise alignments for all sequences against all other sequences and store the result in a similarity matrix. Convert the values in the sequence similarity matrix to distance measures which reflect the evolutionary distance between each pair of sequences.
(2) *Guide-tree*: Construct a guide-tree which defines the order in which pairs of sequences are aligned and combined with previous alignments using the sequence similarity matrix and a neighbour-joining algorithm.
(3) *Multiple alignment*: Align progressively following guide tree. Start by aligning most closely related pairs of sequences and at each step align two sequences or one to an existing subalignment.

— End of Algorithm —

Scoring

There are two main types of scoring used in CLUSTAL W: pairwise scores and multiple alignment scores.

When aligning two sequences, scoring matrices (e.g. PAM250 or BLOSUM62) are used in order to determine a similarity score for matches and mismatches for each amino acids or nucleotides. A penalty is incurred when gaps are inserted, as well as smaller penalties for extending those gaps. The pairwise score between a pair of sequences is calculated as the sum of similarity scores for all aligned pairs of characters minus the gap penalties introduced in either sequences. These scores are necessary for building the guide tree used in the multiple alignment phase.

The multiple alignment score is a sum-of-pair score or SP [1]. For an alignment of N sequences, each of M columns (length), we will denote the i-th column in the alignment by A_{i1}, A_{i2}, ..., A_{iN}. We define $P_{ijk} = 1$ for every pair A_{ij} and A_{ik} which are aligned with each other and $P_{ijk} = 0$ otherwise. The score S_i for the i-th column is

$$S_i = \sum_{j=1, j \neq k}^{N} \sum_{k=1}^{N} P_{ijk}$$

and the SP for the alignment is

$$SP = \frac{\sum_{i=1}^{M} S_i}{\sum_{i=1}^{M_r} S_{ri}} .$$

Here M_r is the number of columns in the reference alignment and S_{ri} is the score S_i for the i-th column in the reference alignment. Other attempts have

been made to improve this scoring method, such as Circular tours and Traveling Salesman Problem [2].

3 Parallel CLUSTAL for Distributed Memory Multicomputer

In the following, we describe a parallelization of the tree basic phases of the CLUSTAL W alignment algorithm. Our method assumes a set of p processors, P_0, P_1, ..., P_{p-1} where each processor.

Algorithm 2 *pCLUSTAL*
Input: set of sequences, $\langle S \rangle$. **Output:** guide tree, T_S; alignment, A_S.
(1.0) Processor P_0 reads the sequence set S. It then determines the mapping of sequence-pairs to processors, based on the number of processors, p, and the number of sequences, $n = |S|$.
(1.1) Processor P_0 sends the precomputed subset of sequences to the corresponding processors, P_i, $1 \leq i \leq p-1$.
(1.2) Each processor P_i, $0 \leq i \leq p-1$, performs a pairwise alignment on its assigned sequences using the CLUSTAL W sequential algorithm.
(1.3) Each processor P_i, $1 \leq i \leq p-1$, sends its relative pairwise alignment scores to P_0.
(2.0) Processor P_0 builds the alignment guide tree, T_S, using the scores gathered in Step (1.3).
(3.0) Processor P_0 analyzes T_S to identify sequence pairings that can be evaluated independently in Step (3.1). Sequences that can be evaluated independently are assigned to processors, P_i, $1 \leq i \leq j-1$.
(3.1) Processor P_0 gathers resulting pairs evaluated in Step (3.0) and, using T_S, sequentially completes the multiple alignment of the sequences, A_S.
— End of Algorithm —

Correctness

The correctness of Algorithm *pCLUSTAL* follows from the correctness of the sequential CLUSTAL W. The reported results are identical.

Performance

We assume that n is the number of input sequences, each of which is bounded in length by m, and that p is the number of processors.

Phase One: Step (1.0) requires $O(nm)$ time to read the sequence file and $O(n^2)$ time to map the sequence pairs to the processors. In Step (1.1), specific subsets of sequences are sent to specific processors requiring $O(1)$ h-relation operations (MPI_AllToAllv). Each processor sends it scores to P_0 in Step (1.3), using $O(1)$ h-relation operations. In Step (1.2), each processor performs a sequential pairwise alignment on its sequence subsets, requiring $O(n^2m^2/p)$ time.

Phase Two: Building the guide tree in Step (2.0) requires $O(n)$ time. There is no communication required in this phase.

Phase Three: The analysis of T_S in Step (3.0) requires $O(n)$ time. The multiple alignment operation in Step (3.1) requires $O(nm^2)$ time. The dispersal

and gathering of independent candidate pairs in Steps (3.0) and (3.1) require $O(1)$ h-relation operations.

Algorithm Complexity: From the previous analysis, the total time for Algorithm *pCLUSTAL* is $O(n^2m^2/p)$ local computation and $O(1)$ h-relation operations for communication. This shows the effectiveness, in theory, of Algorithm *pCLUSTAL*. The local computation shows linear speedup, and the communication overhead consists only of a small, fixed, number of data permutations. Our performance analysis in Section 4 below shows that this property of Algorithm *pCLUSTAL* does indeed translate into very good practice performance of our *pCLUSTAL* code.

4 Experimental Performance Analysis

We implemented Algorithm 2 (*pCLUSTAL*), using C and the MPI communication library. For our experiments, we used a 64 node PC cluster with 1.8 GHz Intel Xeon processors, 512 MB RAM per node and 60 GB of disk storage per node. Every node is running Linux Redhat 7.2 with gcc 2.95.3 and MPI/LAM 6.5.6. The nodes are interconnected via a gigabit Ethernet switch. All parallel times are measured as the wall clock time between the start of the first process and the termination of the last process. We will refer to this as *parallel wall clock time*. All times include the time taken to read the input graph from a file and write the solution into a file. Furthermore, all wall clock times were measured with no other user except us on the parallel machine.

Our experiments measured the following: (1) Parallel wall clock times as a function of the number of processors. (2) Relative speedup (ratio of parallel wall clock time on one processor and parallel wall clock time on p processors) as a function of the number of processors.

As test data, we obtained protein sequence data sets from the National Center for Biotechnology Information (NCBI). The sequences were, on average, about 300 amino acids in length. We used the following sequence sets:

- SYNAPSIN: 28 sequences
- SYNAPTOTAGMIN: 66 sequences
- RELAXIN: 426 sequences
- GLUCAGON: 478 sequences
- KINASE: 647 sequences
- TRYPSIN: 878 sequences

Figures 4 to 6 show the parallel wall clock times as a function of the number of processors for SYNAPSIN, SYNAPTOTAGMIN, RELAXIN, GLUCAGON, KINASE, and TRYPSIN, respectively. The figures show the parallel wall clock times for the three phases of the algorithm (pairwise alignment, guide tree construction, and multiple alignment) as well as the total measured parallel wall clock time. We observe, that all six sets of curves are very similar. The guide tree construction and multiple alignment times are small compared to the pairwise alignment times. The pairwise alignment requires time $O(n^2m^2/p)$ in theory and the times measured are indeed proportional to $1/p$. Furthermore, the pairwise alignment times dominate the guide tree construction and multiple alignment times and, therefore, the total measured parallel wall clock times are similar to the pairwise alignment times and proportional to $1/p$.

Figure 7 shows the measured relative speedup (ratio of parallel wall clock time on one processor and parallel wall clock time on p processors), as a function of the number of processors, for SYNAPSIN, SYNAPTOTAGMIN, RELAXIN, GLUCAGON, KINASE, and TRYPSIN, respectively. Except for the SYNAPSIN case (small data set), the speedup is very good for up to 8 processors, good for up to 16 processors, and still reasonable for up to 32 processors.

Figure 8 is taken from [8] and shows the speedups reported in [8] for the commercial SGI parallel CLUSTAL on a *shared memory* SGI Origin 3000. Despite the much more powerful machine, the speedups reported are similar or less than the speedups obtained by *pCLUSTAL* on a much simpler *distributed memory* PC cluster.

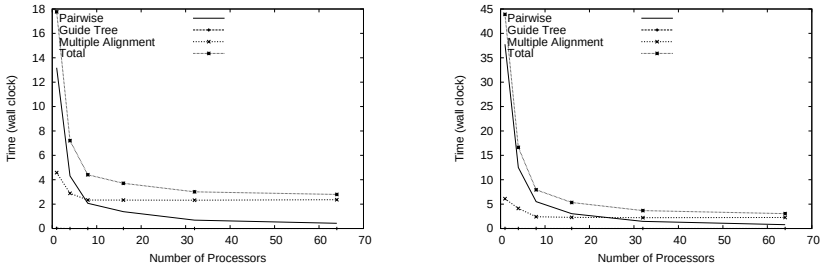

Fig. 4. Parallel Wall Clock Times Measured For SYNAPSIN And SYNAPTOTAGMIN.

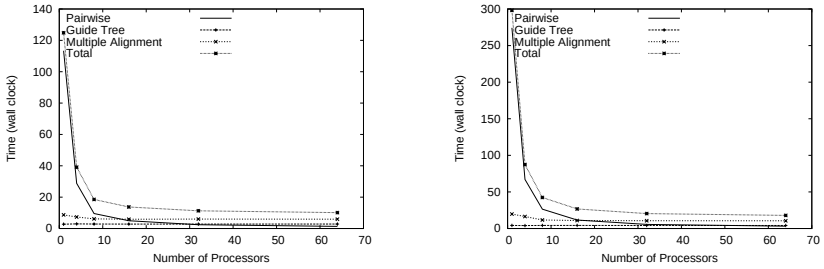

Fig. 5. Parallel Wall Clock Times Measured For RELAXIN And GLUCAGON.

References

1. H. Carillo. and D. Lipman. The multiple sequence alignment problem in biology. *SIAM J. Appl. Math*, 48(5):1073–1082, 1998.

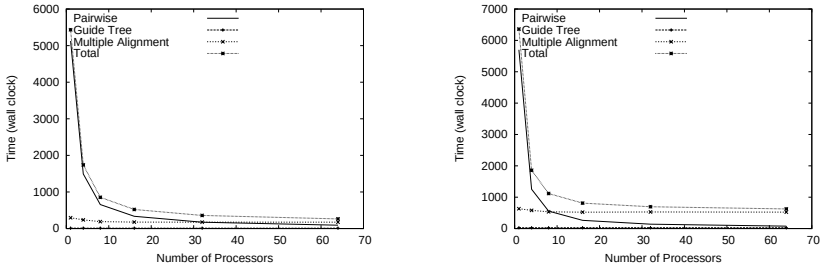

Fig. 6. Parallel Wall Clock Times Measured For KINASE And TRYPSIN.

2. G.H. Gonnet, C. Korostensky, and S. Benner. Evaluation measures of multiple sequence alignments. *J Comput Biol.*, 7(1-2):261–276, 2000.
3. H.G. Higgins, A.J. Bleasby, and R. Fuchs. Clustal v: improved software for multiple sequence alignment. *CABIOS*, 8:189–191, 1992.
4. H.G. Higgins and P.M. Sharp. Clustal: a package for performing multiple sequence alignment on a microcomputer. *Gene*, 73:237–244, 1988.
5. H.G. Higgins and P.M. Sharp. Fast and sensitive multiple sequence alignments on a microcomputer. *CABIOS*, 5:151–153, 1989.
6. H.G. Higgins, J.D. Thompson, and T.J. Gibson. Using clustal for multiple sequence alignments. *Methods Enzymol*, 266:383–402, 1996.
7. F. Jeanmougin, J.D. Thompson, M. Gouy, D.G. Higgins, and T.J. Gibson. Multiple sequence alignment with clustal x. *Trends Biochem Sci*, 23:403–405, 1998.
8. Performance optimization of clustal w: Parallel clustal w, ht clustal, and multi-clustal. http://www.sgi.com/ industries/ sciences/ chembio/ resources/ papers/ Clustal_DevNews/ Clustal_DevNews.html.
9. J.D. Thompson, T.J. Gibson, F. Plewniak, F. Jeanmougin, and H.G. Higgins. The clustalx windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucleic Acids Research*, 24:4876–4882, 1997.
10. J.D. Thompson, H.G. Higgins, and T.J. Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.

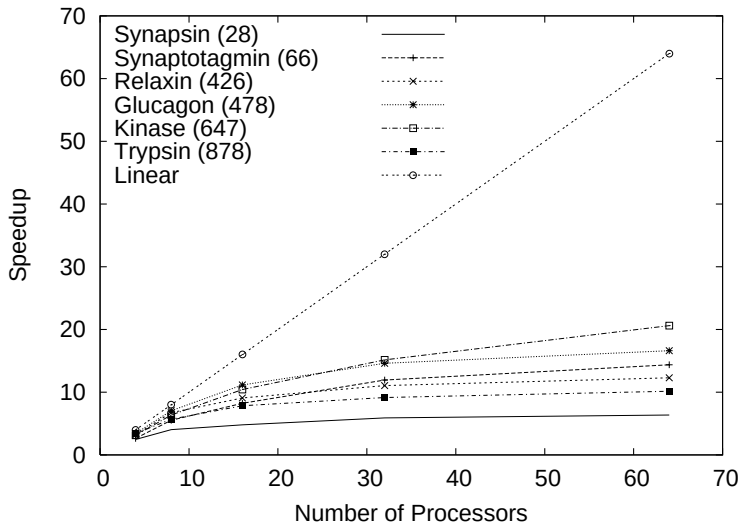

Fig. 7. Relative Speedup For *pCLUSTAL* On A PC Cluster As Measured In Figures 4 To 6

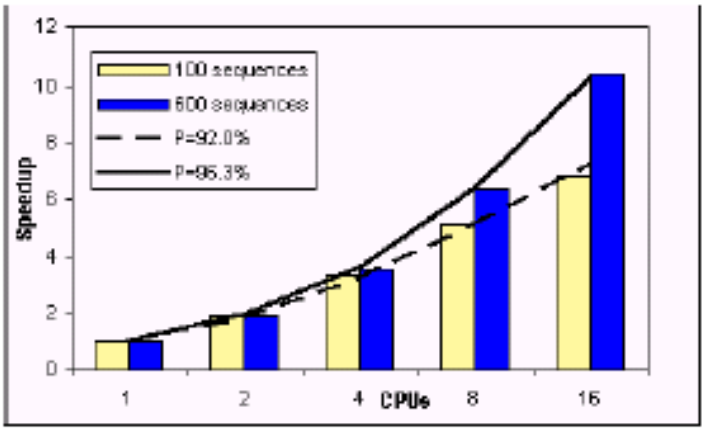

Fig. 8. Relative Speedup of SGI's parallel CLUSTAL on a SGI Origin 3000. From [8]: "Parallel Clustal W scaling. Calculation was done for 100 and 600 GPCR protein sequences with the average length 390 amino acids."