A Parallel Wavefront Algorithm for Efficient Biological Sequence Comparison

C. E. R. Alves, E. N. Cáceres, F. Dehne, S. W. Song

ICCSA 2003 - Technical Session on Coarse Grained Parallel Algorithms For Scientific Applications

May 19, 2003

String Editing or Similarity Problem

Find the edit distance between two strings A and C.

Operations: insertion, deletion, substitution.

Edit Distance = Sum of the costs of each edit operation.

Applications in search for similarities in biosequences.

CGM/BSP Model

CGM (Coarse Grained Multicomputer) model: a small number p of processors, each with its own local memory, communicating through a network.

The algorithm alternates between

- Computation rounds: each processor computes independently.

- Communication rounds: each processor sends/rece data to/from other processors.

Goals:

- Obtain a speed-up linear on p (for a range of values of p).

- Minimize the number of rounds.

A Simple String Alignment Example

Alignment (a):

A	a	c	t	t	c	a	−	t	
C	a	t	t	c	−	a	c	g	
Score	1	0	1	0	0	1	0	0	3

Alignment (b):

A	a	c	t	t	c	a	−	t	
C	a	−	t	t	c	a	c	g	
Score	1	0	1	1	1	1	0	0	5

Consider strings A and C.

Insert spaces such that they become equal in length.

Assign score 1 when symbols match; 0 otherwise.

Total score of case (a): 3

Total score of case (a): 5

A More General Score Assignment

Consider strings A and C: $|A| = n$ and $|C| = m$.

In the alignment, consider a column consisting of symbols r of A and s of C. The score $S(r, s)$ is defined as:

- If $r = s$: score $S(r, s) = f(r, s) (> 0)$ (match)
- If $r \neq s$: score $S(r, s) = f(r, s) (< 0)$ (mismatch)
- If we insert a space: $S(r, s) = -k$

Dynamic Programming Approach

We can compute $S(r, s)$ by using $S(r-1, s)$, $S(r-1, s-1)$ and $S(r, s-1)$ because there are only three ways of computing an alignment between $A[1 \ldots r]$ and $C[1 \ldots s]$: We can

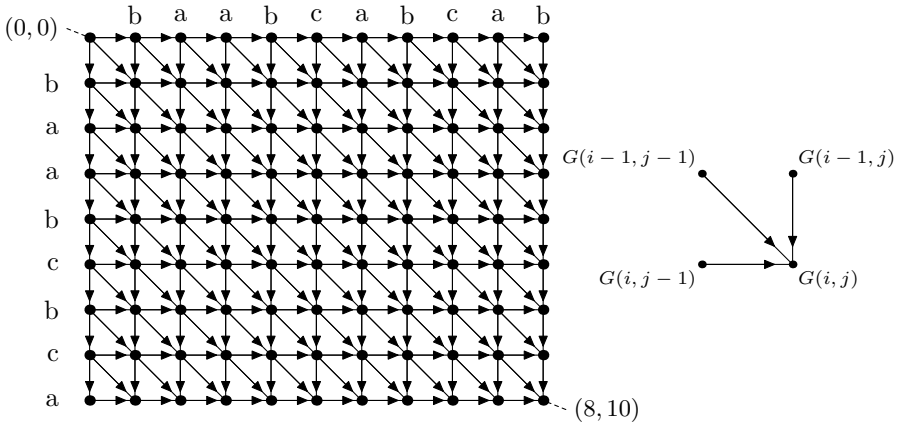
- align $A[1..r]$ with $C[1..s-1]$ and match a space with $C[s]$, or
- align $A[1..r-1]$ with $C[1..s]$ and match a space with $A[r]$, or
- align $A[1..r-1]$ with $C[1..s-1]$ and match (or mismatch) $A[r]$ with $B[s]$. Thus:

$$S(r, s) = \max \begin{cases} S[r, s-1] - k \\ S[r-1, s-1] + f(r, s) \\ S[r-1, s] - k \end{cases}$$

Grid Directed Acyclic Graph (GDAG)

$$S(r,s) = \max \begin{cases} S[r, s-1] - k \\ S[r-1, s-1] + f(r,s) \\ S[r-1, s] - k \end{cases}$$

This can be illustrated by a grid directed acyclic graph (GDAG):



Highest scoring path from (0,0) to (8,10)= best alignment.

Sequential algorithm: $O(mn)$ time.

Previous Parallel Algorithms for this Problem

Apostolico et al. 1990:

- CREW: $O(\log m \log n)$ time
  with $O(mn/\log m)$ processors $(n \geq m)$
- CRCW: $O(\log n(\log\log m)^2)$ time
  with $O(mn/\log\log m)$ processors
- in both case: $O(mn)$ space

Galil and Park 1992:

- CREW: $O(\sqrt{n}\log n)$ time

Alves, Cáceres, Denhe, Song 2002

- CGM algorithm $O(\log p)$ communication rounds

Here we present an $O(p)$ communication rounds CGM algorithm that requires communication with few neighbor processors and is very simple to implement.

An $O(p)$ Commun. Rounds Algorithm

$A = \{a_1 \ldots a_m\}$, $C = \{c_1 \ldots c_n\}$ with $|A| = m$ and $|C| = n$.

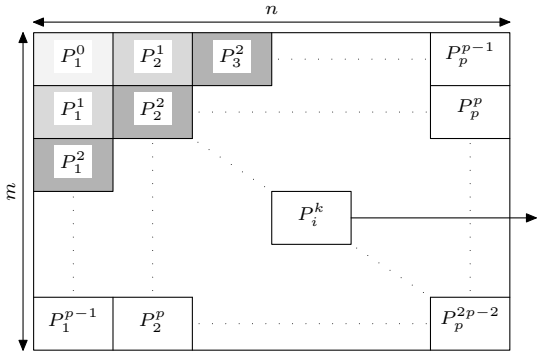
C is divided into p pieces of size $\frac{n}{p}$.

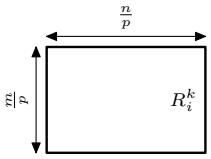
Each processor P_i receives A and the i-th piece of C.

Each P_i computes $S_i(r, s)$ of the submatrix S_i based on $S_i(r-1, s)$, $S_i(r-1, s-1)$ and $S_i(r, s-1)$.

Processor P_i can only start to compute $S_i(r, s)$ after P_{i-1} has computed $S_{i-1}(r, s)$.

Idea of the Algorithm





P_i^k denotes execution of processor i at round k.

R_i^k, $1 \leq i, k \leq p$, elements of the right boundary (rightmost column) of the k-th part of submatrix S_i.

$$R_i^k = \{S_i(r, i\frac{n}{p}), (k-1)\frac{m}{p} + 1 \leq r \leq k\frac{m}{p}\}.$$

After computing the k-th part of the submatrix S_i, processor P_i sends the elements R_i^k to processor P_{i+1}.

Using R_i^k, processor P_{i+1} can compute the k-th part of the submatrix S_{i+1}.

An Improvement using Parameterized Algorithm

Observation: The last processor P_p only starts its work at round $p-1$ when processor P_1 is finishing its computation.

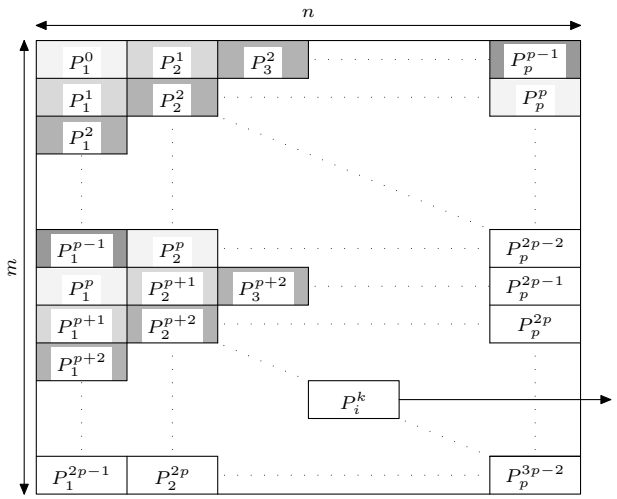
Therefore we have a very bad load balancing.

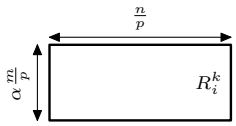
We can improve the load balancing by assigning work to the processors sooner.

We can decrease the size of the messages that processor P_i sends to processors P_{i+1}.

Instead of message size $\frac{m}{p}$ consider sizes $\alpha\frac{m}{p}$ ($\alpha \leq 1$) and explore several sizes of the parameter α.

Choosing Parameter α





Exemple: Using $\alpha = \frac{1}{2}$

Parameter $\alpha \leq 1$ expresses the trade-off between the workload of each processor and the number of communication rounds required.

Small α means smaller workload and more communication rounds.

Case when $\alpha = 1/2$: Processors start to work earlier but requires $3p-2$ communication rounds.

The $(1+\frac{1}{\alpha})p-2$ Rounds Similarity Algorithm:

Input: Value i of the processor P_i; entire string A and the substring C_i (size $\frac{n}{p}$); the parameter α.

Output: Score $S(r,s)$, $(i-1)\frac{m}{\sqrt{p}}+1 \leq r \leq i\frac{m}{\sqrt{p}}$ and $(j-1)\frac{n}{p}+1 \leq s \leq j\frac{n}{p}$.

1: **for** $1 \leq k \leq \frac{p}{\alpha}$ **do**
2: **if** $i = 1$ **then**
3: **for** $\alpha(k-1)\frac{m}{p}+1 \leq r \leq \alpha k\frac{m}{p}$ **and** $1 \leq s \leq \frac{n}{p}$ **do**
4: `compute` $S(r,s)$;
5: **end for**
6: `send(`R_i^k`,`P_{i+1}`);`
7: **end if**
8: **if** $i \neq 1$ **then**
9: **receive**(R_{i-1}^k, P_{i-1});
10: **for** $\alpha(k-1)\frac{m}{p}+1 \leq r \leq \alpha k\frac{m}{p}$ **and** $1 \leq s \leq \frac{n}{p}$ **do**
11: `compute` $S(r,s)$;
12: **end for**
13: **if** $i \neq p$ **then**
14: `send(`R_i^k`,`P_{i+1}`)`
15: **end if**
16: **end if**

17: **end for**

Main Theoretical Result

Theorem 1 *The proposed algorithm with parameter α solves the string editing problem in $(1+\frac{1}{\alpha})p-2$ communication rounds with local computation time of $O(\frac{mn}{p})$ in each processor.*

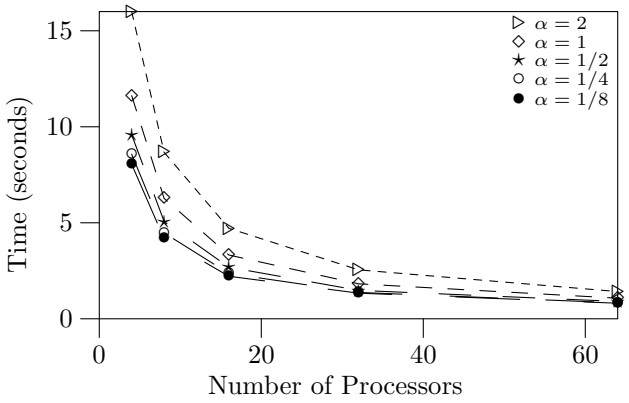
Implementation Results

64-node Beowulf: each with 256MB RAM, 256MB swap memory, CPU Intel Pentium III 448.956 MHz, 512KB cache, connected with a 100 Mb fast-Ethernet switch. Code is written in standard ANSI C using the LAM-MPI library.

. Parameter Tuning (see curves)

. Execution Times (see curves)

Parameter Tuning

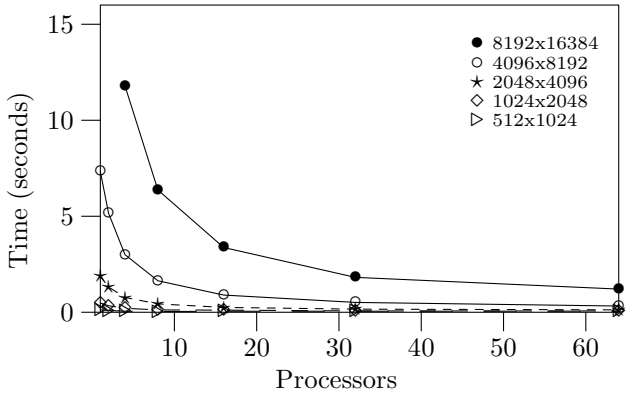


Using input strings: m=8000 and n=16000

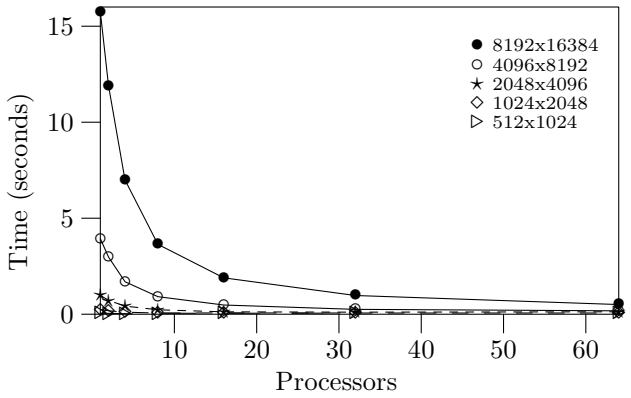
Best value for α between 1/4 and 1/8.

Execution Times ($\alpha = 1$)

We tried two different implementations.



Each processor stores the entire submatrix it computes: quadratic space.



Each processor stores one row at a time of the submatrix it computes: linear space.

Conclusion

- $O(p)$-communication-round CGM algorithm.
- Each processor communicates with two neighbors.
- Apropriate for implementation in *grid computing*.
- Parameter α expresses the trade-off between the workload of each processor and the number of communication rounds required; can be tuned empirically in a given computing platform.