AD/A-002 248

ON THE EFFICIENT COMPUTATION OF RECURRENCE RELATIONS

Don E. Heller

Carnegie-Mellon University

Prepared for:

Office of Naval Research
National Science Foundation
National Aeronautics and Space Administration

13 June 1974

DISTRIBUTED BY:

NTIS

National Technical Information Service
U. S. DEPARTMENT OF COMMERCE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	AD/A-002 248
4. TITLE (and Subtitle)	ON THE EFFICIENT COMPUTATION OF RECURRENCE RELATIONS
5. TYPE OF REPORT & PERIOD COVERED	Final
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s)	Don E. Heller
8. CONTRACT OR GRANT NUMBER(s)	N00014-67-A-0314-0010 Nr 044-422; CMU 1-51039
9. PERFORMING ORGANIZATION NAME AND ADDRESS	Carnegie-Mellon University Department of Computer Science Pittsburgh, PA 15213
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS	
11. CONTROLLING OFFICE NAME AND ADDRESS	Office of Naval Research Arlington, VA 22217
12. REPORT DATE	June 13, 1974
13. NUMBER OF PAGES	7
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	
15. SECURITY CLASS. (of this report)	UNCLASSIFIED
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
US Department of Commerce
Springfield, VA. 22151

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A new parallel algorithm for the solution of a general linear recurrence is described. Its relation to the work of Kogge and Stone is discussed.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ON THE EFFICIENT COMPUTATION OF RECURRENCE RELATIONS

Don E. Heller

June 13, 1974

Carnegie-Mellon University
Department of Computer Science
Pittsburgh, Pa. 15213

This paper was prepared in part while the author was in residence at ICASE, NASA Langley Research Center under NASA Grant NGR47-102-001 and in part at Carnegie-Mellon University under National Science Foundation Grant GJ32111 and Office of Naval Research Contract N00014-67-A-0314-0010, NR 044-422.

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

ON THE EFFICIENT COMPUTATION OF RECURRENCE RELATIONS

Recently much progress has been made in the formulation of parallel algorithms which compute the terms of a sequence (y_i) defined by

$$(1)\qquad \begin{aligned} &y_0 \text{ given,} \\ &y_i = f_i(y_0, y_1, \ldots, y_{i-1}),\ i = 1, \ldots, N. \end{aligned}$$

The germinal point of this work is the now well-known "log-sum" algorithm which computes $\Sigma_{i=1}^{N} a_i$ in $\lceil \log_2 N \rceil$ parallel addition steps, given $\lceil N/2 \rceil$ processors. Here the underlying recurrence is

$$y_0 = o$$

$$y_i = y_{i-1} + a_i,\ i = 1, \ldots, N;$$

y_N is the desired result.

Two apparently distinct generalizations of the log-sum algorithm have appeared. Kogge and Stone [1] have considered the case

$$(2)\qquad \begin{aligned} &y_0 = b_0 \\ &y_i = f(b_i, g(a_i, y_{i-1})),\ i = 1, \ldots, N, \end{aligned}$$

where f is associative, g distributes over f, and there is a function h such that $g(x,g(y,z)) = g(h(x,y),z)$. Seemingly restricted to first order recurrences, by a suitable mapping m^{th} order recurrences are also treated. Heller [2] has studied the case

$$(3)\qquad \begin{aligned} &y_0 = h_0 \\ &y_i = \Sigma_{j=o}^{i-1}\ a_{ij} y_j + h_i,\ i = 1, \ldots, N. \end{aligned}$$

This problem is equivalent to the solution of a lower triangular linear system of equations. In this note we give an improved parallel algorithm for (3) and show a relationship between the two generalizations.

Rewrite (3) as $(I-L)y=h$, where L is a strictly lower triangular matrix, and I is the identity. y and h are $(N+1)$-vectors. Since $L^{N+1} = 0$,

$$\begin{aligned}(I-L)^{-1} &= (I+L+L^2+\cdots+L^N)\\ &= (I+L^{2^m})(I+L^{2^{m-1}})\cdots(I+L)\end{aligned}$$

where $2^m \leq N < 2^{m+1}$. Thus we have the algorithm:

$\{x_0 \leftarrow h;\ LI \leftarrow I\};$

$\underline{\text{for}}\ i \leftarrow 0\ \underline{\text{step}}\ 1\ \underline{\text{until}}\ m-1\ \underline{\text{do}}$

$\quad \{x_{i+1} \leftarrow (I+L^{2^i})\, x_i;$

$\quad\quad L^{2^{i+1}} \leftarrow L^{2^i}\, L^{2^i};$

$\quad\quad LI \leftarrow (I+L^{2^i})LI\};$

$\{y \leftarrow (I+L^{2^m})\, x_m\ ; LI \leftarrow (I+L^{2^m})LI\}.$

The algorithm is sequential in i, and within the braces operations are performed concurrently. When completed, we have the desired y, and $(I-L)^{-1}$ is stored in LI. LI may now be used to compute y' given h'. It is easily shown that, with $O(N^3)$ processors, the calculation may be done in $m^2 + 3m + 1$ parallel steps of addition and multiplication. (We use the fact that matrix products may be computed in logarithmic time with sufficiently many processors.) The previous result required $O(N^4)$ processors and $m^2 + 4m + 2$ operation steps.

We now turn to the Kogge - Stone results. Rewrite (2) as

$$\begin{aligned} &y_0 = b_0 \\ (2') \quad & \\ &y_i = a_i \otimes y_{i-1} \oplus b_i,\ i=1,\ldots,N.\end{aligned}$$

Here g is replaced by the binary operation $\otimes$, and f by $\oplus$. Assume that $\oplus$ is associative, $\otimes$ distributes over $\oplus$, and there is a $\otimes'$ such that $a \otimes (b \otimes c) = (a \otimes' b) \otimes c$. Let α be a symbol distinct from all others, and define $\alpha \oplus x = x \oplus \alpha = x$, $\alpha \otimes x = x \otimes \alpha \quad \alpha$ for all x. Define an operator L on (N+1)-vectors by

$$(Lz)_0 = \alpha$$

$$(Lz)_i = a_i \otimes z_{i-1},\ i = 1,\ldots,N.$$

Then $y = Ly \oplus b$ by (2'). It is observed that L is an additive operator since $\otimes$ distributes over $\oplus$ and by the definition of α. Moreover, $L^{N+1} = \alpha$, since for any z, and $i = 1,\ldots,N+1$, $(L^i z)_0 = \alpha$ and for $1 \le j < i$, $(L^i z)_j = (L(L^{i-1}z))_j = a_j \otimes (L^{i-1}z)_{j-1} = a_j \otimes \alpha = \alpha$. Therefore,

$$\begin{aligned} y &= Ly \oplus b = L(Ly \oplus b) \oplus b = L^2 y \oplus (L \oplus I)b \\ &= \cdots = L^{N+1} y \oplus (L^N \oplus L^{N-1} \oplus \cdots \oplus I)b \\ &= (L^N \oplus L^{N-1} \oplus \cdots \oplus I)b \\ &= (L^{2^m} \oplus I)(L^{2^{m-1}} \oplus I)\cdots(L \oplus I)b. \end{aligned}$$

Since $L^3 = (L^2)L = L(L^2)$, $\otimes'$ behaves as an associative operation, and so

$$\begin{aligned} (L^{2^i} y)_j &= \alpha,\ 0 \le j < 2^i \\ &= a_j \otimes (a_{j-1} \otimes (\ldots \otimes (a_{j-2^i+1} \otimes y_{j-2^i})\cdots) \\ &= (a_j \otimes' a_{j-1} \otimes' \cdots \otimes' a_{j-2^i+1}) \otimes y_{j-2^i},\ 2^i \le j \le N. \end{aligned}$$

Similarly,

$$\begin{aligned} (L^{2^{i+1}} y)_j &= \alpha,\ 0 \le j < 2^{i+1} \\ &= [(a_j \otimes' \cdots \otimes' a_{j-2^i+1}) \\ &\quad \otimes' (a_{j-2^i} \otimes' \cdots \otimes' a_{j-2^{i+1}+1})] \otimes y_{j-2^{i+1}},\ 2^{i+1} \le j \le N, \end{aligned}$$

and the "coefficients" of $L^{2^{i+1}}$ may be computed from the "coefficients" of L^{2^i} in one $\otimes'$ operation step. Thus an algorithm similar to the previous one may be used to compute y. If the operator $(L^N \cdots \oplus I)$ is not formed, the computation time is $O(\log_2 N)$ with $O(N)$ processors. In fact, if $y' = Ly' \oplus b'$, it is less efficient to directly apply $(L^N \oplus \cdots \oplus I)$ than to use the above method.

The general recurrence (1) may be written as $y = L_1 y$, where L_1 is a strictly lower triangular operator in the sense that, for any z, $(L_1 z)_i$ is independent of $z_i, z_{i+1}, \ldots, z_N$. By an induction argument L_1^{N+1} is a constant operator, and so the solution may be found by $y = L_1^{n+1} z$ for any z. The special cases (2) and (3) allow the simple computation of the powers of L_1 when $L_1 z = Lz \oplus b$, and L is linear. Kung [3] has shown that for non-linear recurrences, it is not possible, in general, to decrease the computation time by more than a constant factor by the use of parallelism.

REFERENCES

1. Kogge, P. M. and Stone, H. S., "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations." IEEE Trans. on Computers, C-22 (1973), pp. 786-793.

2. Heller, D., "A Determinant Theorem with Applications to Parallel Algorithms." To appear, SIAM Journal Num. Anal., 1974.

3. Kung, H. T., "New Algorithms and Lower Bounds for the Parallel Evaluation of Certain Rational Expressions." Proceedings, ACM Symposium on Theory of Computing, 1974.