1998 International Computer Symposium
Workshop on Algorithms
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

Parallel Two-dimensional Prefix Computation on the PRAM

Pou-Yah Wu
Department of Mathematics Education
National Tainan Teachers College,Tainan, Taiwan, R.O.C.
E-mail address: pywu@ipx.ntntc.edu.tw

ABSTRACT

This paper provides a two-dimensional prefix computation and its parallel algorithms on the CREW PRAM machine. The two-dimensional prefix computation is to compute the values $\sum_{0\le i<p}\sum_{0\le j<q} x_{ij}$ for all up-left submatrices $[x_{ij}]_{p\times q}$, which contain the first row and first column element x_{00}, of a matrix. This computation can be applied to compute grayscale sums of subimages for the fractal image compression and to find the maximum sum submatrix. First, we design a faster parallel algorithm with time complexity $O(\log n)$ using $O(n^2)$ processors. Then, a cost-optimal algorithm is found to run in $O(\log n)$ time using $O(\frac{n^2}{\log n})$ processors.

1. INTRODUCTION

Parallel prefix computation is an important parallel computation method which has many applications, for example, maximum sum subsequence, job sequencing with deadlines, etc. Parallel two-dimensional prefix computation computes the values $\sum_{0\le i<p}\sum_{0\le j<q} x_{ij}$ for all up-left submatrices $[x_{ij}]_{p\times q}$ of a matrix $[x_{ij}]_{m\times n}$, where up-left submatrices are those submatrices which contain the first row and first column element x_{00}. It can be applied to precompute grayscale sums of subimages for fractal image compressions[5] and to find the maximum sum submatrix of a matrix.

Parallel prefix computation is a one-dimensional case of parallel two-dimensional computation. There are many researches on the parallel prefix computation[1-4, 6-17]. Akl[1] summarized as the following. A sequential algorithm of the prefix computation had time complexity $O(n)$. On the CREW PRAM, a parallel prefix algorithm ran in $O(\log n)$ time using $O(n)$ processors and a cost-optimal (or work-optimal) parallel prefix algorithm had the time complexity $O(\log n)$ using $O(\frac{n}{\log n})$ processors. Then, it could be improved by the divide-and-conquer method with the time complexity $O(\log\log n)$ using $O(\frac{n}{\log\log n})$ processors.

In this paper, we design the parallel two-dimensional prefix computation on the CREW PRAM. We provide the two-dimensional prefix computation and its sequential algorithm in Section 2. Then, Section 3 gives a faster parallel two-dimensional prefix computation and a cost-optimal parallel two-dimensional prefix computation. Finally, conclusions are in Section 4.

2. TWO-DIMENSIONAL PREFIX COMPUTATION

Given a set χ and define an operation $\circ$ on χ such that

(1) The operation $\circ$ is a binary operation, i.e. $x_i \circ x_j$ where $x_i, x_j \in \chi$.

(2) The operation $\circ$ satisfies the closure, that is, if $x_i, x_j \in \chi$ then $x_i \circ x_j \in \chi$.

(3) The operation $\circ$ satisfies the associative law, that is, if $x_i, x_j, x_k \in \chi$ then
$(x_i \circ x_j) \circ x_k = x_i \circ (x_j \circ x_k) = x_i \circ x_j \circ x_k$.

Consider matrix $X = [x_{ij}]_{m \times n}$, where $x_{ij} \in \chi$ and $0 \le i < m, 0 \le j < n$. To compute the following quantities :

$s_{00} = x_{00}$,
$s_{01} = x_{00} \circ x_{01}$,
$s_{02} = x_{00} \circ x_{01} \circ x_{02}$,
$\vdots$
$s_{0,n-1} = x_{00} \circ x_{01} \circ \cdots \circ x_{0,n-1}$,
$s_{10} = x_{00} \circ x_{10}$,
$\vdots$
$s_{1,n-1} = x_{00} \circ x_{01} \circ \cdots \circ x_{0,n-1} \circ x_{10} \circ x_{11} \circ \cdots \circ x_{1,n-1}$,
$\vdots$
$s_{m-1,n-1} = x_{00} \circ x_{01} \circ \cdots \circ x_{0,n-1} \circ \cdots \circ x_{m-1,0} \circ x_{m-1,1} \circ \cdots \circ x_{m-1,n-1}$.

That is, $s_{ij} = \sum_{0 \le u \le i} \sum_{0 \le v \le j} x_{uv}$ where $0 \le i \le m-1$ and $0 \le j \le n-1$.

We can also express the above expressions as following :

$s_{00} = x_{00}$,
$s_{0j} = s_{0,j-1} \circ x_{0j}$ where $1 \le j < n$,
$s_{i0} = s_{i-1,0} \circ x_{i0}$ where $1 \le i < m$ and
$s_{ij} = s_{i-1,j} \circ x_{i0} \circ \cdots \circ x_{ij}$ where $1 \le i < m$ and $1 \le j < n$.

The process, which obtain $S=[s_{ij}]_{m \times n}$ from $X=[x_{ij}]_{m \times n}$, is called the two-dimensional prefix computation. Similarly, the two-dimensional suffix computation is defined as the following expressions:

$a_{m-1,n-1} = x_{m-1,n-1}$,
$a_{m-1,j} = x_{m-1,j} \circ a_{m-1,j+1}$,
$a_{i,n-1} = x_{i,n-1} \circ a_{i+1,n-1}$ and
$a_{ij} = x_{ij} \circ x_{i,j+1} \circ \cdots \circ x_{i,n-1} \circ a_{i+1,j}$.

Example 1 : Suppose χ be the set of integers and $\circ$ be the addition of integers. Given a matrix

$$X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$

As the above definitions, we can obtain matrices S and A by the two-dimensional prefix computation and the two-dimensional suffix computation respectively, where

$$S = \begin{bmatrix} 1 & 3 & 6 \\ 5 & 12 & 21 \\ 12 & 27 & 45 \end{bmatrix} \text{ and } A = \begin{bmatrix} 45 & 33 & 18 \\ 39 & 28 & 15 \\ 24 & 17 & 9 \end{bmatrix}.$$ Q.E.D.

Suppose the operation $\circ$ take the constant time. Since the element $s_{m-1,n-1}$ in the matrix S is computed from x_{ij} for all $0 \le i < m$ and $0 \le j < n$, the sequential two-dimensional prefix computation has the time complexity $O(n^2)$, as the algorithm in Fig.1.

Algorithm RAM TWO-DIMENTIONAL PREFIX COMPUTATION
Step 1 : $s_{00} \leftarrow x_{00}$
Step 2 : for j = 0 to n-2 do
  $s_{0,j+1} \leftarrow s_{0j} \circ x_{0,j+1}$
  end for
Step 3 : for i = 0 to m-2 do
  $s_{i+1,0} \leftarrow s_{i0} \circ x_{i+1,0}$
  $r \leftarrow x_{i+1,0}$
  for j = 1 to n-2 do
    $r \leftarrow r \circ x_{i+1,j}$
    $s_{i+1,j} \leftarrow s_{ij} \circ r$
  end for
end for

Figure 1. The algorithm of the sequential two-dimensional prefix computation

In order to process the submatrix in the multiprocessor system, we discuss the relations of the two-dimensional prefix computation between the submatrix and the original matrix in Theorem 1. Without loss of generality, we take $\circ$ as the addition operation of real numbers and take the inverse operation of $\circ$ as the subtraction operation of real numbers.

Theorem 1. Suppose M be a submatrix of a matrix X as the following:

$$M = \begin{bmatrix} x_{pq} & x_{p,q+1} & \cdots & x_{pv} \\ \vdots & & & \\ x_{uq} & x_{u,q+1} & \cdots & x_{uv} \end{bmatrix}.$$

Let S^X and S^M be the two-dimensional prefix sums of X and M, respectively. Then

$$s^M_{ij} = s^X_{i+p,j+q} - s^X_{p-1,j+q} - s^X_{i+p,q-1} + s^X_{p-1,q-1}.$$

In particular, the submatrix sum

$$s^M_{u-p,v-q} = s^X_{uv} - s^X_{p-1,j} - s^X_{i,q-1} + s^X_{p-1,q-1}.$$ Q.E.D.

Example 2 : Suppose the matrices X and S be as in Example 1. Now, given $M = \begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix}$, s^M_{11} =28. S^M_{11} can also be computed by

$$s_{22} - s_{02} - s_{20} + s_{00} = 45 - 6 - 12 + 1 = 28.$$ Q.E.D.

3. PARALLEL TWO-DIMENSIONAL PREFIX COMPUTATION ON THE PRAM

3.1. A Faster Algorithm

We describe an algorithm to find $S = [s_{ij}]$ from $X = [x_{ij}]$ on the PRAM. Suppose $P_0, P_1, \cdots, P_{m\times n-1}$ be m×n processors on the PRAM. Initially, the matrix X is stored in shared memory by having $P_{i\times n+j}$ reads x_{ij} from the input, for $0 \le i \le m-1$ and $0 \le j \le n-1$. Processor $P_{i\times n+j}$ sets the variable s_{ij} equal to x_{ij}. These processes take the constant time.

The algorithm is composed by $\log n + \log m$ iterations; during each step, the binary operation is performed by pairs of processors. First, the indices of these processors, which read elements in the same row of the matrix X, are separated by a distance of 2^{k-1} at the k-th iteration, for $1 \le k \le \log n$. Then, the indices are separated by a distance of $2^{k-1} \times n$ at the $(\log n + k)$-th iteration, for $1 \le k \le \log m$.

Hence, in the first iteration, we compute

$$s_{01} \leftarrow s_{00} \circ s_{01}, s_{02} \leftarrow s_{01} \circ s_{02}, \cdots, s_{0,n-1} \leftarrow s_{0,n-2} \circ s_{0,n-1},$$
$$s_{11} \leftarrow s_{10} \circ s_{11}, s_{12} \leftarrow s_{11} \circ s_{12}, \cdots, s_{1,n-1} \leftarrow s_{1,n-2} \circ s_{1,n-1},$$
$$\vdots$$
$$s_{m-1,1} \leftarrow s_{m-1,0} \circ s_{m-1,1}, \cdots, s_{m-1,n-1} \leftarrow s_{m-1,n-2} \circ s_{m-1,n-1}.$$

Then, in the second iteration, we compute

$$s_{02} \leftarrow s_{00} \circ s_{02}, s_{03} \leftarrow s_{01} \circ s_{03}, \cdots; s_{0,n-1} \leftarrow s_{0,n-3} \circ s_{0,n-1},$$
$$s_{12} \leftarrow s_{10} \circ s_{12}, s_{13} \leftarrow s_{11} \circ s_{13}, \cdots; s_{1,n-1} \leftarrow s_{1,n-3} \circ s_{1,n-1},$$
$$\vdots$$
$$s_{m-1,2} \leftarrow s_{m-1,0} \circ s_{m-1,2}, s_{m-1,3} \leftarrow s_{m-1,1} \circ s_{m-1,3}, \cdots; s_{m-1,n-1} \leftarrow s_{m-1,n-3} \circ s_{m-1,n-1}.$$

In the log(n)-th iteration, we compute

$$s_{0,\frac{n}{2}} \leftarrow s_{00} \circ s_{0,\frac{n}{2}}, s_{0,\frac{n}{2}+1} \leftarrow s_{01} \circ s_{0,\frac{n}{2}+1}, \cdots; s_{0,n-1} \leftarrow s_{0,\frac{n}{2}-1} \circ s_{0,n-1},$$
$$s_{1,\frac{n}{2}} \leftarrow s_{10} \circ s_{1,\frac{n}{2}}, s_{1,\frac{n}{2}+1} \leftarrow s_{11} \circ s_{1,\frac{n}{2}+1}, \cdots; s_{1,n-1} \leftarrow s_{1,\frac{n}{2}-1} \circ s_{1,n-1},$$
$$\vdots$$
$$s_{m-1,\frac{n}{2}} \leftarrow s_{m-1,0} \circ s_{m-1,\frac{n}{2}}, s_{m-1,\frac{n}{2}+1} \leftarrow s_{m-1,1} \circ s_{m-1,\frac{n}{2}+1}, \cdots; s_{m-1,n-1} \leftarrow s_{m-1,\frac{n}{2}-1} \circ s_{m-1,n-1}.$$

And, in other iterations, similar expressions can be computed to obtain the finial values in s_{ij}. Figure 2 illustrates the two-dimensional prefix computation while m=4 and n=4 as the algorithm in Figure 3.

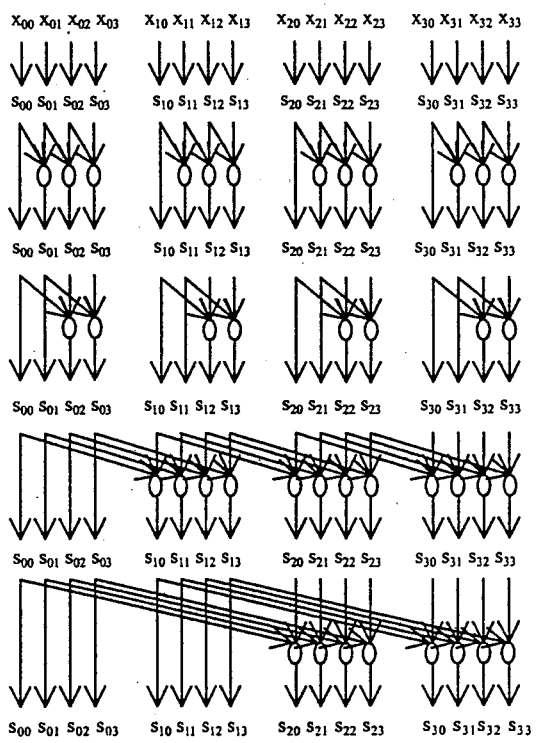

Figure 2.The two-dimensional prefix computation on the PRAM

Algorithm PARALLEL TWO-DIMENSIONAL PREFIX COMPUTATION ON THE PRAM

```
for i = 0 to log n-1 do
  for j = 0 to m × n-1 do in paraellel
    l = j mod n
    k = (j-l)/n
    if (2^i ≤ l) then s_kl ← s_{k,l-2^i} ∘ s_kl
  end for
end for
for i = 0 to log m-1 do
  for j = 0 to n × m-1 do in parallel
    l = j mod n
    k = (j-l)/n
    if (2^i ≤ k) then s_kl ← s_{k-2^i,l} ∘ s_kl
  end for
end for
```

Figure 3. The parallel algorithm of two-dimensional prefix computation on the PRAM

The time complexity of the parallel two-dimensional prefix computation is $O(\log n)$. Since $p(n)=O(n^2)$, the cost of this algorithm is

$$\begin{aligned} c(n) &= p(n)\times t(n) \\ &= O(n^2)\times O(\log n) \\ &= O(n^2 \log n) \end{aligned}$$

This cost is not optimal while the sequential two-dimensional prefix computation on the RAM requires the time complexity $O(n^2)$

3.2. A Cost-Optimal Algorithm

Suppose $p=\sqrt{\log m}$, $q=\sqrt{\log n}$, $u=\frac{m}{p}$ and $v=\frac{n}{q}$. The PRAM has $u\times v$ processors $P_{00}, P_{01}, \cdots, P_{u-1,v-1}$. Consider that the matrix $X=[x_{ij}]_{m\times n}$ is decomposed into $u\cdot v$ submatrices with size $p\times q$, that is,

$$Y_{00}=\begin{bmatrix} x_{00} & \cdots & x_{0,q-1} \\ \vdots & & \\ x_{p-1,0} & \cdots & x_{p-1,q-1} \end{bmatrix}=\begin{bmatrix} y_{00}^{00} & \cdots & y_{0,q-1}^{00} \\ \vdots & & \\ y_{p-1,0}^{00} & \cdots & y_{p-1,q-1}^{00} \end{bmatrix},$$

$$Y_{01}=\begin{bmatrix} x_{0,q} & \cdots & x_{0,2q-1} \\ \vdots & & \\ x_{p-1,q} & \cdots & x_{p-1,2q-1} \end{bmatrix}=\begin{bmatrix} y_{00}^{01} & \cdots & y_{0,q-1}^{01} \\ \vdots & & \\ y_{p-1,0}^{01} & \cdots & y_{p-1,q-1}^{01} \end{bmatrix},$$

$$\vdots$$

$$Y_{u-1,v-1}=\begin{bmatrix} x_{m-p-1,n-q-1} & \cdots & x_{m-p-1,n-1} \\ \vdots & & \\ x_{m-1,n-q-1} & \cdots & x_{m-1,n-1} \end{bmatrix}=\begin{bmatrix} y_{00}^{u-1,v-1} & \cdots & y_{0,q-1}^{u-1,v-1} \\ \vdots & & \\ y_{p-1,0}^{u-1,v-1} & \cdots & y_{p-1,q-1}^{u-1,v-1} \end{bmatrix},$$

First, processor P_{ij} reads Y_{ij}, where $0\le i\le u-1$ and $0\le j\le v-1$, and stores it into shared memory of the PRAM. Then, apply RAM TWO-DIMENSIONAL PREFIX COMPUTATION to obtain the matrix

$$\begin{bmatrix} s_{00}^{ij} & \cdots & s_{0,q-1}^{ij} \\ \vdots & & \\ s_{p-1,0}^{ij} & \cdots & s_{p-1,q-1}^{ij} \end{bmatrix} \quad \text{where} \quad s_{kl}^{ij}=\sum_{0\le r\le k}\sum_{0\le t\le l} y_{rt}^{ij},$$

$0\le i\le u-1$ and $0\le j\le v-1$.

Since each processor executes $\sqrt{\log m}\times\sqrt{\log n}$ iterations, this step requires $O(\log n)$.

Secondly, compute the following matrix by the PARALLEL TWO-DIMENSIONAL PREFIX COMPUTATION on processors $P_{00}, P_{01}, \cdots, P_{u-1,v-1}$.

$$\begin{bmatrix} s_{p-1,q-1}^{00} & \cdots & s_{p-1,q-1}^{0,v-1} \\ \vdots & & \\ s_{p-1,q-1}^{u-1,0} & \cdots & s_{p-1,q-1}^{u-1,v-1} \end{bmatrix}$$

This step can obtain the final values of $s_{p-1,q-1}^{ij}$ and runs in $O\left(\log\left(\frac{n}{\sqrt{\log n}}\right)\right)$ time.

Then, each processor processes the last column $\begin{bmatrix} s_{0,q-1}^{ij} \\ \vdots \\ s_{p-2,q-1}^{ij} \end{bmatrix}$ as the following. For each element in this column, compute $[s_{r,q-1}^{i0} \quad s_{r,q-1}^{i1} \quad \cdots \quad s_{r,q-1}^{i,v-1}]$ where $0\le r\le p-2$ by parallel prefix computation on processors $P_{i0}, P_{i1}, \cdots, P_{i,v-1}$. This step requires

$O\left(\sqrt{\log n}\log\log\left(\frac{n}{\sqrt{\log n}}\right)\right)$. And, add $s^{i-1,j}_{p-1,q-1}$ to each element in this column with time complexity $O\left(\sqrt{\log n}\right)$.

Similarly, each processor processes the last row $[s^{ij}_{p-1,0} \cdots s^{ij}_{p-1,q-2}]$ and requires $O\left(\sqrt{\log n}\log\log\left(\frac{n}{\sqrt{\log n}}\right)\right)$. Add $s^{i,j-1}_{p-1,q-2}$ to each element in this row with time complexity $O\left(\sqrt{\log n}\right)$.

Finally, process other elements on each processor. Compute the following matrix on each processor by

$s^{ij}_{rt} = s^{ij}_{rt} + s^{i-1,j}_{p-1,t} + s^{i,j-1}_{r,q-1} - s^{i-1,j-1}_{p-1,q-1}$, $0 \le r \le p-2$ and $0 \le t \le q-2$.

$$\begin{bmatrix} s^{ij}_{00} & \cdots & s^{ij}_{0,q-2} \\ \vdots & & \\ s^{ij}_{p-2,0} & \cdots & s^{ij}_{p-2,q-2} \end{bmatrix}$$

Due to Theorem 1, s^{ij}_{rt} has the two-dimensional prefix sum. This step requires the time complexity $O(\log n)$.

Figure 4 shows this algorithm. The time complexity is

$$O(\log n)+O\left(\log\left(\frac{n}{\sqrt{\log n}}\right)\right)+O\left(\sqrt{\log n}\log\log\left(\frac{n}{\sqrt{\log n}}\right)\right)+O\left(\sqrt{\log n}\right)$$

We need to compare $O(\log n)$ with $O\left(\sqrt{\log n}\log\log\left(\frac{n}{\sqrt{\log n}}\right)\right)$ in the following theorem. And, the time complexity is $O(\log n)$.

Theorem 2 : $O(\log n) \ge O\left(\sqrt{\log n}\log\log\left(\frac{n}{\sqrt{\log n}}\right)\right)$.

Proof : We only show that $\sqrt{\log n} \ge \log\log n$. Let $f(x) = \sqrt{\log x} - \log\log x$.
$f'(x) = \frac{1}{2x\log x}[\sqrt{\log x} - 2]$. When $x \ge 16$, $f'(x) \ge 0$. And, $f(16) = 0$.

Hence $O(\log n) \ge O\left(\sqrt{\log n}\log\log\left(\frac{n}{\sqrt{\log n}}\right)\right)$ Q.E.D.

Since $p(n) = O\left(\frac{n^2}{\log n}\right)$, the cost of the algorithm is

$$\begin{aligned} c(n) &= p(n) \times t(n) \\ &= O\left(\frac{n^2}{\log n}\right) \times O(\log n) \\ &= O(n^2) \end{aligned}$$

Hence this cost is optimal. We give Example 3 to illustrate this cost-optimal parallel two-dimensional prefix algorithm as in Figure 5.

Example 3 : Let $\circ$ be +, m=4, n=4 and χ be the set of integers. Here p=2, q=2, u=2 and v=2. We use four processors : P_{00}, P_{01}, P_{10} and P_{11}. Initially, decompose the matrix $\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$ into four submatrices so that processors P_{00}, P_{01}, P_{10} and P_{11} have $\begin{bmatrix} 1 & 2 \\ 5 & 6 \end{bmatrix}$, $\begin{bmatrix} 3 & 4 \\ 7 & 8 \end{bmatrix}$, $\begin{bmatrix} 9 & 10 \\ 13 & 14 \end{bmatrix}$ and $\begin{bmatrix} 11 & 12 \\ 15 & 16 \end{bmatrix}$ respectively. Then, according to the cost-optimal two-dimensional prefix computation, the steps can be processed as the following.

Step 1 : Processors P_{00}, P_{01}, P_{10} and P_{11} compute $\begin{bmatrix} 1 & 3 \\ 6 & 14 \end{bmatrix}$, $\begin{bmatrix} 3 & 7 \\ 10 & 22 \end{bmatrix}$, $\begin{bmatrix} 9 & 19 \\ 22 & 46 \end{bmatrix}$ and $\begin{bmatrix} 11 & 23 \\ 26 & 54 \end{bmatrix}$ respectively.

Step 2 : $\begin{bmatrix} 14 & 22 \\ 46 & 54 \end{bmatrix}$ can be computed to obtain $\begin{bmatrix} 14 & 36 \\ 60 & 136 \end{bmatrix}$ on processors P_{00}, P_{01}, P_{10} and P_{11} parallelly.

Step 3 : Process $[3]$, $[7]$, $[19]$ and $[23]$ on P_{00}, P_{01}, P_{10} and P_{11} respectively.

Step 3.1 : $[3 \quad 7]$ can be computed to obtain $[3 \quad 10]$ on processors P_{00} and P_{01}. Simultaneously, $[19 \quad 23]$ can be computed to obtain $[19 \quad 42]$ on processors P_{10} and P_{11}.

Step 3.2 : Add 14 to 19 on processor P_{10} and add 36 to 42 on processor P_{11} concurrently.

Algorithm COST-OPTIMAL PARALLEL TWO-DIMENSIONAL PREFIX COMPUTATION

Step 1 : Apply RAM TWO-DIMENSIONAL PREFIX COMPUTATION to obtain the following matrix on each processor.

$$\begin{bmatrix} s^{ij}_{00} & \cdots & s^{ij}_{0,q-1} \\ \vdots & & \\ s^{ij}_{p-1,0} & \cdots & s^{ij}_{p-1,q-1} \end{bmatrix}$$

Step 2 : Apply PARALLEL TWO-DIMENSIONAL PREFIX COMPUTATION to yield the following matrix on processors $P_{00}, P_{01}, \cdots, P_{u-1,v-1}$.

$$\begin{bmatrix} s^{00}_{p-1,q-1} & \cdots & s^{0,v-1}_{p-1,q-1} \\ \vdots & & \\ s^{u-1,0}_{p-1,q-1} & \cdots & s^{u-1,v-1}_{p-1,q-1} \end{bmatrix}$$

Step 3 : Process $\begin{bmatrix} s^{ij}_{0,q-1} \\ \vdots \\ s^{ij}_{p-2,q-1} \end{bmatrix}$ on each processor.

Step 3.1 : Compute $[s^{i0}_{r,q-1} \quad s^{i1}_{r,q-1} \quad \cdots \quad s^{i,v-1}_{r,q-1}]$ by PARALLEL PREFIX COMPUTATION on processors $P_{i0}, P_{i1}, \cdots, P_{i,v-1}$.

Step 3.2 : Add $s^{i-1,j}_{p-1,q-1}$ to each element in this column.

Step 4 : Process $[s^{ij}_{p-1,0} \quad \cdots \quad s^{ij}_{p-1,q-2}]$ on each processor.

Step 4.1 : Compute $\begin{bmatrix} s^{oj}_{p-1,r} \\ s^{1j}_{p-1,r} \\ \vdots \\ s^{u-1,j}_{p-1,r} \end{bmatrix}$ by PARALLEL PREFIX COMPUTATION on the processors

$P_{0j}, P_{1j}, \cdots, P_{u-1,j}$.

Step 4.2 : Add $s^{i,j-1}_{p-1,q-1}$ to each element in this row.

Step 5 : Obtain the following matrix on each processor by computing

$$\begin{cases} s^{ij}_{rt} = s^{ij}_{rt} + s^{i-1,j}_{p-1,t} + s^{i,j-1}_{r,q-1} - s^{i-1,j-1}_{p-1,q-1} & \text{if } i \neq 0 \text{ or } j \neq 0 \\ s^{ij}_{rt} = s^{ij}_{rt} + s^{i,j-1}_{r,q-1} & \text{if } i = 0 \text{ and } j > 0 \\ s^{ij}_{rt} = s^{ij}_{rt} + s^{i-1,j}_{p-1,t} & \text{if } i > 0 \text{ and } j \neq 0 \end{cases}$$

where $0 \leq r \leq p-2$ and $0 \leq t \leq q-2$.

$$\begin{bmatrix} s^{ij}_{00} & \cdots & s^{ij}_{0,q-2} \\ \vdots & & \\ s^{ij}_{p-2,0} & \cdots & s^{ij}_{p-2,q-2} \end{bmatrix}$$

Figure 4. The optimal-cost two-dimensional prefix algorithm

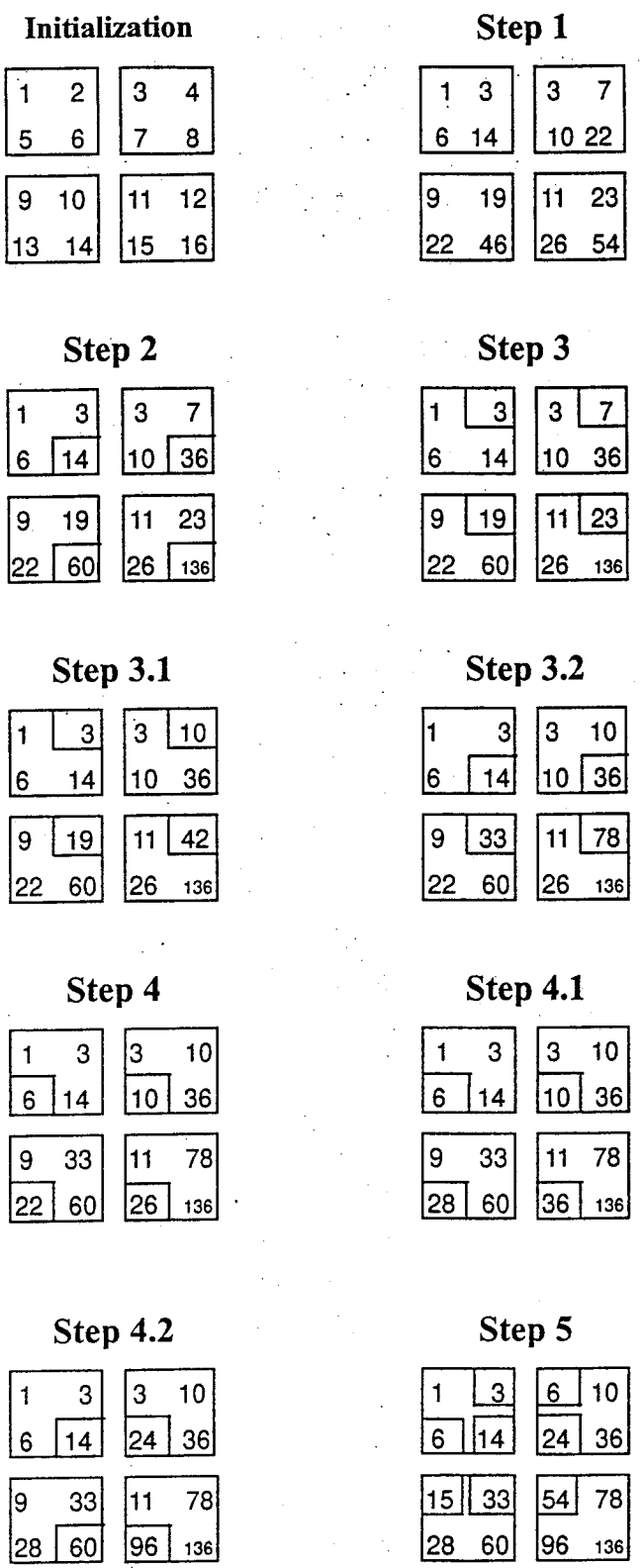

Figure 5. Cost-optimal two-dimensional prefix computation on the PRAM

Step 4 : Process $[6]$, $[10]$, $[22]$ and $[26]$ on processors P_{00}, P_{01}, P_{10} and P_{11} respectively.

Step 4.1 : $\begin{bmatrix}6\\22\end{bmatrix}$ can be computed to obtain $\begin{bmatrix}6\\28\end{bmatrix}$ on processors P_{00} and P_{10}. Simultaneously, $\begin{bmatrix}10\\26\end{bmatrix}$ can be computed to yield $\begin{bmatrix}10\\36\end{bmatrix}$ on processors P_{01} and P_{11}.

Step 4.2 : Add 14 to 10 and add 60 to 36 concurrently.

Step 5 : Processor P_{01} computes 3+3=6. Processor P_{10} computes 6+9=15. Simultaneously, processor P_{11} computes 11+33+24-14=54. Hence the following result matrix can be obtained. $\begin{bmatrix}1&3&6&10\\6&14&24&36\\15&33&54&78\\28&60&96&136\end{bmatrix}$ Q.E.D.

4. CONCLUSIONS

In this paper, we provide the two-dimensional prefix computation and its parallel algorithm on the CREW PRAM. A cost-optimal algorithm has the time complexity $O(\log n)$ using $O(\frac{n^2}{\log n})$ processors. This computation can be applied to the fractal image compression and the maximum sum submatrix. In the future, we shall apply the divide-and-conquer method to solve this computation and implement other data structures to observe this computation.

REFERENCES

[1] Akl, S. G. *Parallel Computation: models and methods.* Prentice-Hall Inc., New Jersey, 1997.

[2] Akl, S. G. *The design and analysis of parallel algorithms.* Prentice-Hall Inc., New Jersey, 1989.

[3] Chung, K. L., *Prefix computation on a generalized mesh-connected computer with multiple buses.* IEEE Transactions on Parallel and Distributed Systems, Vol. 6, 1995, pp.196-199.

[4] Cole, R., and Vishkin, U. *Faster optimal parallel prefix*

sums and list ranking. Information and Control, Vol. 81, 1989, pp.334-352.

[5] Fisher, Y. *Fractal image compression*, New York, Spring-Verlag, 1994.

[6] Hagerup, T. *The parallel complexity of integer prefix summation.* Information Processing Letter, Vol. 56, 1995, pp.59-64.

[7] Han, Y. *Parallel algorithms for computing linked list prefix.* Journal of Parallel and Distributed Computing, Vol. 40, 1991, pp.1149-1153.

[8] Kruskal, C. P., Madej, T., and Rudolph, L. *Parallel prefix on fully connected direct connection machine.* Proceedings of the International Conference on Parallel Processing, 1986, pp. 278-283.

[9] Kruskal, C.P., Rudolph, L., and Smir, M. *The power of parallel prefix.* IEEE Transaction on Computers, Vol. 34, 1985, pp.965-968.

[10] Lander, R. E., and Fisher, M. J. *Parallel prefix computation.* Journal of ACM, Vol. 27, 1980, pp.831-838.

[11] Lakshmivarahan, S., and Dhall, S. K. *Parallel computing using the prefix problem.* Oxford University Press, New York, 1994.

[12] Lubachevsky, B. D., and Greenberg, A. G. *Simple efficient asynch runous parallel prefix algorithms.* Proceedings of the International Conference on Parallel Processing, 1987, pp.66-69.

[13] Meijer, H., and Akl, S. G. *Optimal computation of prefix sums on a binary tree of processors.* International Journal of Parallel Programming, Vol. 16, 1987, pp.127-136.

[14] Perf, J. H. *Probabilistic parallel prefix computation.* Proceedings of the International Conference on Parallel Processing, 1984, pp.291-298.

[15] Saxena, S., Bhatt, P. C. P., and Prasad, V. C. *On parallel prefix computation.* Parallel Processing Letters, Vol. 4, 1994, pp.429-436.

[16] Snir, M. *Depth-size tradeoffs for parallel prefix computation.* Journal of Algorithms, Vol. 7, 1986, pp.185-201.

[17] Springsteel, F., and Stoimenoric, L. *Parallel general prefix computations with geometric, algebraic and other applications.* International Journal of Parallel Programming, Vol. 18, 1989, pp.485-503.