Journal of Mathematical Research and Exposition
Vol.10 No.1 Feb. 1990

Parallel Algorithm for FGFT and Its Application*

Li Lei *You Zhaoyong*

(Xian Jiaotong University)

1. Introduction

The generalizations of usual discrete Fourier transform (DFT) are all defined on a commutative ring. In [1] and [2], a generalization of DFT on a non-commutative ring (FGFT) is given and is applied to some fast linear computation problems. In this paper, parallel computation efficiencies for these fast algorithms have been given without being proved. These computing problems have very good parallelism. But it is difficult to further decrease complexities in this paper. The parallel steps mentioned in this paper are all times of four arithmetic operations, and the orders k of matrix mentioned are all much smaller than n.

2. Parallel Algorithm for FGFT

Let $\varepsilon_N = e^{\frac{2\pi i}{N}}$ or $\varepsilon = e^{\frac{2\pi i}{N}}$ for short when do not being confused, where $i = \sqrt{-1}$. Let E be a diagonal matrix of k order:

$$E = \mathrm{diag}(\varepsilon, \varepsilon, \cdots, \varepsilon)$$

where k is a natural number.

Definition 1 Given a k-th order matrix sequence (real or complex) $A(0)$, $A(1), \cdots, A(N-1)$, the generalization Fourier transfrom is given by

$$X(j) = \sum_{n=0}^{N-1} E^{jn} A(n), \quad (j = 0, \cdots, N-1). \tag{1}$$

with inverse

$$A(n) = \frac{1}{N} \sum_{j=0}^{N-1} E^{-jn} X(j) \quad (n = 0, \cdots, N-1). \tag{2}$$

where $E^{-1} = \varepsilon^{-1} I$. When $k = 1$, the generalized Fourier transfrom is usual discrete Fourier transfrom.

Theorem 1 The parallel steps of computing the generalized Fourier transfrom (1) and its inverse (2) are not more than $O(\log N)$ when using $k^2 N$ proces-

* Received Jau. 26, 1988.

sors.

3. Parallel Algorithm for Matrix Sequence Convolution

Definition 2 Given two k order matrix sequences $A(i)$ and $B(i)$ for $i=0, \cdots, N-1$, and then extended them, i.e.

$$A(i+\mu N)=A(i),\quad B(i+\mu N)=B(i)$$

where $i=0,\cdots,N-1$ and μ is an arbitrary integer, the convolution of matrix sequences $A(i)$ and $B(i)$ is gived by

$$C(j)=\sum_{i=0}^{N-1}A(i)B(j-i)\quad (j=0,\cdots,N-1) \tag{3}$$

If we use usual algorithm, the parallel steps of computing (3) are $O(\log kn)$ when using k^3N^3 processors. If we use FGFT, we can prove following theorem

Theorem 2 It needs $O(\log kN)$ parallel steps to compute (3), the convolutiou of matrix sequences $A(i)$ and $B(i)$ for $i=0,\cdots,N-1$, when using k^3N processors.

According to [3], we can further get following results.

Corollary 1 It needs $O(\log kN)$ parallel steps to compute (3) when using $\frac{k^3N}{\log k}$ processors.

Corollary 2 It needs $O(\log kN)$ parallel steps to compute (3) when using $\frac{k^{2.81}N}{\log k}$ processors.

4. The Partitioned Cyclical Parallel Algorithm

Definition 3 Let $A_i, i=0,\cdots,n-1$ be matrices of k order, partitioned cyclical matrix is given by following nk-th order matrix:

$$A(k,n)=\begin{bmatrix} A_0 & A_1 & \cdots & A_{n-1} \\ A_{n-1} & A_0 & \cdots & A_{n-2} \\ \cdots & \cdots & \cdots & \cdots \\ A_1 & A_2 & \cdots & A_0 \end{bmatrix} \tag{4}$$

Usually, $k \ll n$. If $A(k,n)$ and $B(k,n)$ are all partitioned cyclical matrices like (4), computing $A(k,n)*B(k,n)$ by applying usual inner product algorithm needs $\log kn$ steps and k^3n^3 processors. In this paper, we get:

Theorem 3 It needs $O(\log kn)$ parallel steps at most to compute the product of two partitioned cyclical matrices like (4) when using k^3n processors.

If $A(k,n)$ is inverse, it needs $O(k^4n^4)$ processors and $O(\log^2 kn)$ steps to compute inverse of $A(k,n)$ by applying Gsanky equivalence Theorem [5]. In this paper, we get

Theorem 4 It needs $O(\log n+\log^2 k)$ parallel steps at most to compute inverse

of partitioned cyclical matrix $A(k,n)$ like (4) when using k^4n processors.

5 Parallel Algorithm for Partitioned Ttriangular Matrix

Definition 4 Let A_i be k-th order matrix for $i=0,\cdots,n-1$. Partitioned T-formed (upper) triangular matrix is given by nk-th order matrix

$$A(k,n)=\begin{bmatrix} A_0 & A_1 & \cdots & A_{n-1} \\ 0 & A_0 & \cdots & A_{n-2} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & & A_0 \end{bmatrix} \tag{5}$$

Theorem 5 It needs $O(\log kn)$ parallel steps at most to compute the product $A(k,n)*B(k,n)$ of two partitioned T-formed (upper) trianguler matrices like (5) when using k^3N processors.

Theorem 6 It needs $O(\log^2 kn)$ parallel steps at most to compute inverse of $A(k,n)$ like (5) when using $k^3(n+k)$ processors.

6. Parallel Algorithm for Polynomial Matrix

Definition 6 Polynomial matrix is a rectangle matrix whose elements are polynomials of λ, i.e.

$$A(\lambda)=(a_{ik}(\lambda))_{mn}=(a_{ik}^{(0)}\lambda^{\mu}+a_{ik}^{(1)}\lambda^{\mu-1}+\cdots+a_{ik}^{(\mu)})_{mn}$$
$$i=1,\cdots,m,\quad k=1,\cdots,n.$$

where μ is maxmum number of degrees of polynomial $a_{ik}(\lambda)$.

If let

$$A_j=(a_{ik}^{(j)})_{mn}\quad (j=0,\cdots,\mu).$$

we can get

$$A(\lambda)=A_0\lambda^{\mu}+A_1\lambda^{\mu-1}+\cdots+A_{\mu-1}\lambda+A_{\mu}$$

Now, we will give parallel complexity upper bound for computing product of two above mentioned polynomial matrices.

Theorem 7 Given two polynomials of scalar variable x whose coefficients are M-th order matrices:

$$P(x)=\sum_{i=0}^{n}A(i)x^i,\quad Q(x)=\sum_{j=0}^{m}B(j)x^j$$

then computing $P(x)*Q(x)$ needs $O(\log MN)$ parallel steps and M^3N processors at most, where N is a minimum integer satisfing $N=2^p\geqslant n+m-1$.

Let

$$A(\lambda)=\sum_{i=0}^{m}\tilde{A}_i\lambda^i,\quad B(\lambda)=\sum_{j=0}^{n}\tilde{B}_j\lambda^j.$$

where $\tilde{A}_i$ and $\tilde{B}_i$ are all matrices of k order. If $A(\lambda)$ is regular, i.e. $\tilde{A}_m\neq 0$, we consider parallel algorithm for computiong left quotient $Q(\lambda)$ and left remainder

$R(\lambda)$ of dividing $B(\lambda)$ by $A(\lambda)$:

$$B(\lambda)=Q(\lambda)A(\lambda)+R(\lambda).$$

Thcorem 8 Let $n=2m-1$. It needs $O(\log^2 km)$ parallel steps at most when using $k^3(m+k)$ processors to compute left quotient and left remainder of diving $B(\lambda)$ by $A(\lambda)$.

When $k=1$, we will get parallal complexity upper bound on division of monadic polynomial.

7. Remarks

If the order M and length N of matrix sequence in Theorem 2 satisfy following inequality:

$$M\leqslant\log N.$$

then numbers of processor can be decreased to M^2N and parallel steps to $O(\log N)$. So all results in this paper can be improved similarly.

References

[1] You Zhaoyong, Li Lei, J. of Hanzhong Normal College, 1(1985), 1—10.

[2] You Zhaoyong, Li Lei, J. of Xian Jiaotong University, 6(1987), 37—42.

[3] You Zhaoyong, Li Lei, Mathematica Numerica Sinica, 1(1988), to appear.

[4] You Zhaoyong, Li Lei, J. of Computer Engineering and Science, 3(1986), 1—6.

[5] Zhang Lijun, Chen Zengrong, Gao Kunmin, Design and Analysis of Parallel Algorithm, Hunan Technical Publishing House, China, 1984.

[6] Li Lei, Fast Devision for Polynomial Matrix, be submitted to Mathematics in Practice and Theory.

FGFT的并行算法及其应用

游兆永　　李　磊

（西安交通大学）

摘要　常见的离散Fourier变换（DFT）的推广均定义在一个交换环上．我们在[1]、[2]中给出了DFT在一类非交换环上的推广（FGFT），并将它应用于一些快速线性计算问题。本文将不加证明地列出这些快速算法的并行计算效率。结果表明，这些计算问题亦具有很好的并行性．