Proceedings of the Second IASTED International Conference
European Parallel and Distributed Systems (Euro-PDS'98)
July 1-3, 1998, Vienna, Austria

AN EFFICIENT ALGORITHM FOR MULTI-INDEXED RECURRENCE RELATIONS WITH APPLICATIONS TO COMPRESSION OF MULTIDIMENSIONAL SIGNALS *

ABDOU YOUSSEF
Department of EE & CS
The George Washington University
Washington, DC 20052
USA
email: youssef@seas.gwu.edu

Abstract: Parallel solution of 1-indexed recurrence relations has received much attention, but no effort has been made to design parallel algorithms for multi-indexed recurrence relations. Multi-indexed relations arise in JPEG-standard DPCM compression of images, and thus merits attention. In this paper we design and analyze a parallel algorithm for solving q-indexed recurrence relations of arbitrary order and arbitrary number of indices q. The approach is *dimension shifting*, which involves reducing the number of indices to 1 while clustering the terms of the relation into vectors. This is accomplished by means of special vector operators that we define and study. Our parallel algorithm for doing the dimension shifting and for solving the resulting 1-indexed recurrence relation takes $O(\log^2 N)$ time on meshes of partitionable buses or hypercubes, where N is the data input-output size.

1 Introduction

The massive amounts of imagery data in many applications demand that images be compressed to reduce their storage and transmission requirements. In certain applications, most notably medical imaging, the compression has to be lossless. One of the well-known lossless image compression techniques is differential pulse-code modulation (DPCM), which was adopted by the Joint Photographic Expert Group (JPEG) as an international standard [7]. Since when an image is retrieved, locally or remotely, it has to be decompressed (decoded) online for display, decoding must be as fast as possible. Therefore, fast parallel algorithms for DPCM decoding merit attention.

DPCM decoding is essentially the computation of a 2-indexed scalar recurrence relation of low order; every term in the recurrence relation is indexed with two indices, the row and column positions of pixels. Similarly, DPCM of 3D and 4D images, which is conceivable in applications involving 3D visual modeling with or without motion, leads to 3-indexed and 4-indexed recurrence relations.

Much work has been done on parallel algorithms for solving 1-indexed recurrence relations of arbitrary order [1, 4, 5], due to their applicability in numerical computations; the fastest parallel algorithms for solving those equations take logarithmic time, and are based on parallel prefix computation [6]. However, until recently no work has been reported on multi-indexed recurrence relations. Considering the importance of DPCM coding/decoding of imagery, and the need for fast decoding, parallel algorithms for solving multi-indexed recurrence relations are needed.

In this paper we will develop a new approach, called *dimension shifting*, to design a parallel algorithm for solving multi-indexed recurrence relations of arbitrary order and arbitrary number of indices. The algorithm takes square-logarithmic times on hypercubes and multidimensional meshes of partionable busses.

The paper is organized as follows. The next section presents multi-indexed scalar recurrence relations. Sections 3 develops the dimesnion shifting technique for 2-indexed recurrence relations. Section 4 develops dimension shifting for multi-indexed recurrence relations of arbitrary order, where the number of indices is greater than 2. Finally, the last section closes the paper with conclusions and future directions.

2 Multi-indexed Recurrence Relations and DPCM

The most widely used form of DPCM is presented first as a 2-indexed recurrence relation of order 1. Afterwards, higher order and higher dimension DPCM are presented as a multi-indexed recurrence relation.

DPCM assumes that the interpixel redundancies and correlations can be modeled as a 2D Markov model of some order [3]. For order 1, the model involves three given parameter, a, b and c, as described next. Let $X(0:n-1, 0:n-1)$ be a matrix of pixels, representing an image. The model assumes that every pixel $X(i,j)$ can be largely "predicted" from its three neighbors in the north, west, and north-west as follows:

$$X(i,j) = aX(i,j-1)+bX(i-1,j)+cX(i-1,j-1)+E(i,j) \quad (1)$$

where $E(i,j)$ is *the prediction error*, or *residual*. Note that when i and j are outside their appropriate range, the corresponding value for $X(i,j)$ is assumed to be zero.

DPCM coding computes E from X and then codes it into a bit stream using an entropy coder such as Huffman coding, run-length encoding, or arithmetic coding, which are lossless [8]. DPCM decoding decodes

*This work was performed in part at the National institute of Standards and technology.

the bit stream back to the residuals $E(i,j)$, and then computes the pixel values $X(i,j)$ according to equation 1. Clearly, equation 1 is a 2-indexed scalar recurrence relation of order one.

The generalization of equation 1 comes from higher order DPCM and from multidimensional imagery. DPCM of order $(t_1, t_2, ..., t_q)$ and parameter array $a(0 : t_1, 0 : t_2, ..., 0 : t_q)$, for q-dimensional imagery data, involves a q-indexed recurrence relation of order $(t_1, t_2, ..., t_q)$ in the form:

$$X(i_1, ..., i_q) = \sum_{r_1=0}^{t_1} ... \sum_{r_q=0}^{t_q} a(r_1, ..., r_q) X(i_1 - r_1, ..., i_q - r_q) + E(i_1, ..., i_q), \quad (2)$$

where $a(0, 0, ..., 0)$ is equal to 0 so that $X(i_1, i_2, ..., i_q)$ does not depend on itself. As before, any term is assumed to be zero if its indices are out of their defined bounds.

In this paper we will develop a parallel algorithm for solving equation 2 using the general dimension shifting technique. For simplicity of presentation, we start with developing dimension shifting for 2-indexed relations first. The more general q-indexed relations for arbitrary q will be treated afterwards.

3 Dimension Shifting of 2-Indexed Relations

One fundamental idea in dimension shifting is the use of the *shift operator* $\mathcal{S}$ on column vectors. If $V(0 : n-1)$ is a column vector, then $\mathcal{S}V$ is another vector $W(0 : n-1)$ derived from V by shifting it down one position: for all $i > 0$, $W(i) = V(i-1)$, and $W(0) = 0$. The operator $\mathcal{S}$ can be applied repeatedly (say, r times) on a vector. This is equivalent to composition of operators. Denote by $\mathcal{S}^r$ the resulting operator. Clearly, $\mathcal{S}^r V$ is a vector derived from V by shifting the latter down r times, that is, for all $i = 0, 1, ..., n-1$, $(\mathcal{S}^r V)(i) = V(i - r)$, where a term is 0 when its index is out of bounds. Two special cases of r are worth noting: $r = 0$ and $r = n$. $\mathcal{S}^0$ is by convention the *identity operator* $\mathcal{I}$, that is, $\mathcal{I}V = V$ for any vector V. For convenience, we will sometimes use 1 instead $\mathcal{I}$. When $r = n$, observe that $\mathcal{S}^n V = 0$, the zero vector of length n. Accordingly, we simply denote the operator $\mathcal{S}^n$ by 0. Note that for all $r \geq n$, $\mathcal{S}^r = 0$. Polynomial operators in $\mathcal{S}$, that is, any linear combination of the powers of $\mathcal{S}$, can be defined as follows.

Definition 1 *Given a real sequence $a_0, a_1, ..., a_{n-1}$, a polynomial operator $\mathcal{T} = \sum_{l=0}^{n-1} a_l \mathcal{S}^l$ is defined to be such that for any vector V, $\mathcal{T}V = \sum_{l=0}^{n-1} a_l (\mathcal{S}^l V)$. The vector $T(0 : n-1)$, where $T(i) = a_i$ for all i, is called the characteristic vector of operator $\mathcal{T}$.*

It is important to note that $\mathcal{T}V$ is some form of convolution of vectors T and V. We elaborate on this important observation before we proceed with converting 2-indexed relations to 1-indexed relations.

Recall that the convolution of two vectors $U(0 : n-1)$ and $V(0 : n-1)$ is a vector $W(0 : 2n-1)$ where $W(i) = \sum_{r=0}^{i} U(r)V(i-r)$, using again the convention that when indices are out of their bounds, the corresponding terms are zeros. Only the first n components of W will be of use to us. Therefore, we denote by $\otimes$ the partial convolution operation that returns the first n components of the full convolution. For ease of reference, we will define partial convolution formally.

Definition 2 *Let $U(0 : n-1)$ and $V(0 : n-1)$ be two vectors. The partial convolution of U and V, denoted $U \otimes V$, is a vector $W(0 : n-1)$ of the same length as U and V such that $W(i) = \sum_{r=0}^{i} U(r)V(i-r)$, for $i = 0, 1, ..., n-1$.*

Since full convolution is an associative operation, partial convolution $\otimes$ is associative too.

Theorem 1 *Let $\mathcal{T} = \sum_{r=0}^{n-1} a_r \mathcal{S}^r$ be a polynomial operator of characteristic vector T, where $T(i) = a_i$, and let $V(0 : n-1)$ be a vector. Then $\mathcal{T}V = T \otimes V$.*

Proof: Let $W = \mathcal{T}V = \sum_{r=0}^{n-1} a_r \mathcal{S}^r V$, and recall that $(\mathcal{S}^r V)(i) = V(i - r)$. Therefore, $W(i) = \sum_{r=0}^{n-1} a_r (\mathcal{S}^r V)(i) = \sum_{r=0}^{n-1} a_r V(i - r) = \sum_{r=0}^{n-1} T(r)V(i - r)$, for all $i = 0, 1, ..., n-1$. It follows that $W = T \otimes V$, and thus $\mathcal{T}V = T \otimes V$. □

Now we are in position to carry out dimension shifting on 2-index relations. Throughout this section, the following notational convention will be followed. For any matrix $M(0 : n-1, 0 : n-1)$, denote by M_j the j-th column of M, so that $M_j(i) = M(i, j)$.

Recall that a general 2-indexed recurrence relation of order (k, t) and parameter array $a(0 : k, 0 : t)$ is

$$X(i,j) = \sum_{l=0}^{k} \sum_{s=0}^{t} a(l,s) X(i-l, j-s) + E(i,j). \quad (3)$$

Using the notation $M_j(i)$ for $M(i, j)$, equation 3 becomes

$$X_j(i) = \sum_{l=0}^{k} \sum_{s=0}^{t} a(l,s) X_{j-s}(i-l) + E_j(i)$$

Since $X_{j-s}(i - l) = (\mathcal{S}^l X_{j-s})(i)$, the above equation transforms to

$$X_j(i) = \sum_{l=0}^{k} \sum_{s=0}^{t} (a(l,s) \mathcal{S}^l X_{j-s})(i) + E_j(i), \quad \textit{for all } i.$$

Because the last equation involves the i-th components of the various column vectors, the index i can be dropped, yielding a condensed vector equation:

$$X_j = \sum_{s=0}^{t} \sum_{l=0}^{k} (a(l,s) \mathcal{S}^l X_{j-s}) + E_j,$$

where the order of the double summation is reversed. By splitting the summation over s into two parts, one for $s = 0$ and the other for $s \neq 0$, and by recalling that $a(0, 0) = 0$, we obtain

$$X_j = (\sum_{l=1}^{k} a(l,0) \mathcal{S}^l X_j) + \sum_{s=1}^{t} \sum_{l=0}^{k} a(l,s) \mathcal{S}^l X_{j-s} + E_j.$$

Next, move the first sum on the right hand side to the left, and factor X_j out, resulting in

$$(1 - \sum_{l=1}^{k} a(l,0)\mathcal{S}^l)X_j = \sum_{s=1}^{t}(\sum_{l=0}^{k} a(l,s)\mathcal{S}^l)X_{j-s} + E_j. \quad (4)$$

To simplify the notation, let

$$\mathcal{T}_0 = 1 - \sum_{l=1}^{k} a(l,0)\mathcal{S}^l, \quad \mathcal{T}_s = \sum_{l=0}^{k} a(l,s)\mathcal{S}^l$$

so that equation 4 becomes $\mathcal{T}_0 X_j = \sum_{s=1}^{t} \mathcal{T}_s X_{j-s} + E_j$, leading to

$$X_j = \sum_{s=1}^{t} \mathcal{T}_0^{-1}\mathcal{T}_s X_{j-s} + \mathcal{T}_0^{-1}E_j. \quad (5)$$

The main question now is whether the polynomial operator $\mathcal{T}_0$ has indeed an inverse, and if so, compute its inverse. As the next theorem will show, $\mathcal{T}_0^{-1}$ does exist, and it is a polynomial operator whose coefficients will be the solution of a scalar recurrence relation of order k.

Theorem 2 *1) $\mathcal{T}_0^{-1} = \sum_{l=0}^{n-1} C(l)\mathcal{S}^l$, where the $C(l)$'s satisfy the following 1-indexed scalar recurrence relation of order k:*

$$C(0) = 1,\ C(r) = \sum_{l=1}^{k} a(l,0)C(r-l),\ \textit{for } 1 \le r \le n-1. \quad (6)$$

2) For every $s = 1, 2, \ldots, t$, $\mathcal{T}_0^{-1}\mathcal{T}_s = \sum_{l=0}^{k} B_s(l)\mathcal{S}^l$, where

$$B_s = C \otimes a(0 : n-1, s). \quad (7)$$

Note that for $m > k$, $a(m,s) = 0$.

Proof: 1) The $C(l)'s$ satisfy
$(1 - \sum_{l=1}^{k} a(l,0)\mathcal{S}^l)(\sum_{l=0}^{n-1} C(l)\mathcal{S}^l) = 1.$
In that identity relation, the coefficient of $\mathcal{S}^0$ is $C(0)$ on the left side and 1 on the right side, leading to $C(0) = 1$. For $r = 1, 2, \ldots n-1$, the coefficient of $\mathcal{S}^r$ is $C(r) - \sum_{l=1}^{min(k,r)} a(l,0)C(r-l)$ on the left hand side, and 0 on the right hand side. Thus, $C(r) = \sum_{l=1}^{min(k,r)} a(l,0)C(r-l)$. By taking $C(m) = 0$ for $m < 0$, the value of $C(r)$ simplifies to $C(r) = \sum_{l=1}^{k} a(l,0)C(r-l)$.
2) Thr proof of this part follows immediately from observing that operator polynomials multiply in the same way as real polynomials, that polynomial multiplication is equivalent to convolution of the coefficients, and that $\mathcal{S}^r = 0$ for all $r \ge n$. □

Using Theorem 2, and recalling from Theorem 1 that applying a polynomial operator to a vector is a convolution $\otimes$, equation 5 becomes

$$X_j = \sum_{s=1}^{t} B_s \otimes X_{j-s} + F_j \quad (8.a)$$
$$F_j = C \otimes E_j. \quad (8.b)$$

Clearly, equation 8.a is a 1-indexed vector recurrence relation of order t. This completes the dimension shifting from index dimension to term dimension. It remains to shift the order dimension to term dimension.

The standard method for solving 1-indexed recurrence relations of order $t > 1$ is to convert the equation into a vector equation of order 1 such that every vector has t terms, where each term has the same dimensionality as the terms of the original recurrence relation. Specifically, equation 8.a will be converted to the following form

$$\overline{X}_j = A \odot \overline{X}_{j-1} + \overline{F}_j \quad (9)$$

where $\overline{X}_j$ is a column vector of t vectors

$$\overline{X}_j = [X_{tj-1}\ X_{tj-2}\ X_{tj-2}\ \ldots X_{tj-t}]^T,\ j = 1, 2, \ldots, n/t,$$

and A, $\overline{F}_j$ and operation $\odot$ are to be defined. Note that in the definition of $\overline{X}_j$, the exponent T stands for matrix transpose. Note also that $\overline{X}_0$ is the zero vector of length $t \times n$.

The conversion from order t to order 1 is usually straightforward, but because each term in equation 8.a is a long vector and the operations involved, especially $\otimes$, are fairly costly, the conversion becomes a time-consuming task, and thus warrants special attention to parallelize it. The conversion entails computing the matrix A and the vectors $\overline{F}_j$'s, and defining the operation $\odot$.

The values of A and the $\overline{F}_j$'s are derived by repeated application of equation (8.a) so that each X_{tj-r}, for $r = 0, 1, \ldots, t-1$, is expressed in terms of $X_{t(j-1)-1}$, $X_{t(j-1)-2}$, ..., $X_{t(j-1)-t}$ and F_{tj-r}, F_{tj-r-1},...,F_{tj-t}. We summarize next the outcome of that derivation/conversion process, leaving out the derivation details for the sake of brevity.

- The matrix A in equation 9 is a $t \times t$ matrix where each term $A_{i,s}$ is a vector of length n, for $i = 1, \ldots, t$ and $s = 1, 2, \ldots, t$. Specifically, $A_{i,s}$ satisfies the following equation

$$A_{t,s} = B_s,\ A_{i-1,s} = B_s \otimes A_{i,1} + A_{i,s+1},$$
$$i = t, t-1, \ldots, 2,\ s = 1, 2, \ldots, t \quad (10)$$

  where B_s was defined in equation 7.

- For every column vector $Y = [Y_1\ Y_2\ \ldots\ Y_t]^T$ where each Y_i is a vector of length n,

$$A \odot Y = Z,\ \textit{where } Z(i) = \sum_{s=1}^{t} A_{i,s} \otimes Y_s. \quad (11)$$

  Note that $A \odot Y$ is similar to a matrix-vector product except that the scalar multiplication operation is replaced by the vector convolution operation $\otimes$. Note also that if $\odot$ is performed on a system consisting of t subsystems, where subsystem i computes $Z(i)$, then the time for $\odot$ is $T_\odot = t \times T_\otimes + (t-1) \times T_+$. In particular, if each subsystem has n processors, then $T_\otimes = O(\log n)$ time by performing parallel convolution through FFT, and $T_+ = O(1)$ because it is a parallel vector addition. Hence, $T_\odot = O(tT_\otimes) = O(t \log n)$.

- Each column vectors $\overline{F}_j$ in equation 9 is of the form

$$\overline{F}_j = [F_{j,1}\ F_{j,2}\ \ldots\ F_{j,t}]^T, \quad (12)$$

where the terms $F_{j,r}$ are themselves vectors of length n, satisfying the following equation

$$F_{j,t} = F_{tj-t}, \ and for \ r = 1, 2, ..., t-1,$$

$$F_{j,r} = F_{tj-r} + \sum_{s=r+1}^{t} A_{t+r+1-s,1} \otimes F_{tj-s}, \ j = 1, 2, ..., n/t. \tag{13}$$

Note that the F_{tj-r}'s are defined in equation 8.b.

4 Generalization to Multi-indexed Relations

In this section we treat the most general case, namely, q-indexed relations of arbitrary order, where $q \geq 2$. The treatment will require a generalization of: (1) the 1D shift operator to $(q-1)$-dimensional shift operators, (2) the subsequent generalization of the "single-variable" polynomial operator to something that resembles a multivariable polynomial operator, and (3) finally the use of $(q-1)$-dimensional convolution rather than 1D convolution. The conversion process from q-indexed arbitrary-order relations to 1-indexed relations of order 1 is otherwise the same as in the previous section.

Recall that a q-indexed recurrence relation of order $(t_1, ..., t_q)$ has the following form:

$$X(i_1, ..., i_q) = \sum_{r_1=0}^{t_1} ... \sum_{r_q=0}^{t_q} a(r_1, ..., r_q) X(i_1 - r_1, ..., i_q - r_q) + E(i_1, ..., i_q) \tag{14}$$

where $i_l = 0, 1, ..., N_l - 1$ for $l = 1, 2, ..., q$, the array $E(0 : N_1 - 1, 0 : N_2 - 1, ..., 0 : N_q - 1)$ and the array $a(0 : t_1, 0 : t_2, ..., 0 : t_q)$ of parameters are given, $a(0, 0, ..., 0)$ is equal to 0 so that $X(i_1, i_2, ..., i_q)$ does not depend on itself, and $t_l << N_l$ for $l = 1, 2, ..., q$. As before, any term is assumed to be zero if its indices are out of their defined bounds.

To simplify the multi-index cumbersome notation, we will introduce some notations and conventions. Let

- $\mathcal{R} = \{(r_1, r_2, ..., r_q) \mid r_l = 0, 1, ...t_l \ for \ all \ l = 1, 2, ..., q\}$, and any R in $\mathcal{R}$ is a vector $R = (r_1, r_2, ..., r_q)$.
- $\mathcal{R}' = \{(r_1, r_2, ..., r_{q-1}) \mid r_l = 0, 1, ...t_l \ for \ all \ l = 1, 2, ..., q-1\}$.
- $\mathcal{R}^* = \mathcal{R} - \{(0, 0, ..., 0)\}$, that is, the set $\mathcal{R}$ except of the zero-tuple.
- $\mathcal{R}'^* = \mathcal{R}' - \{(0, 0, ..., 0)\}$.
- $\mathcal{I} = \{(i_1, i_2, ..., i_q) \mid i_l = 0, 1, ...N_l - 1 \ for \ all \ l = 1, 2, ..., q\}$, and any I in $\mathcal{I}$ is by convention a vector $I = (i_1, i_2, ..., i_q)$.
- $\mathcal{I}' = \{(i_1, i_2, ..., i_{q-1}) \mid i_l = 0, 1, ...N_l - 1 \ for \ all \ l = 1, 2, ..., q-1\}$.
- Convention: If $R = (r_1, r_2, ..., r_q)$ is an index vector in $\mathcal{R}$, then R' is vector derived from R by dropping the last term of R. Clearly, $R' = (r_1, r_2, ..., r_{q-1}) \in \mathcal{R}'$, and $(R', r_q) = R$. Similarly, for any index vector I in $\mathcal{I}$, denote by I' the vector derived from I by dropping the last term from the latter.
- Convention: For any q-dimensional array M and any integer i_q, denote by M_{iq} the $(q-1)$-dimensional array such that $M_{i_q}(i_1, i_2, ..., i_{q-1}) = M(i_1, i_2, ..., i_{q-1}, i_q)$. Equivalently, $M_{i_q}(I') = M(I)$.

Accordingly, equation 14 becomes

$$X(I) = \sum_{R \in \mathcal{R}^*} a(R) X(I - R) + E(I). \tag{15}$$

Using the last two conventions indicated above, the last equation translates to

$$X_{i_q}(I') = \sum_{R \in \mathcal{R}^*} a(R) X_{i_q - r_q}(I' - R') + E_{i_q}(I').$$

Split the multiple summation on the right hide side of the previous equation into two parts: one for $r_q = 0$ and another for $r_q \neq 0$. This leads to

$$X_{i_q}(I') = \sum_{R' \in \mathcal{R}'^*} a(R', 0) X_{i_q}(I' - R') + \sum_{r_q=1}^{t_q} \sum_{R' \in \mathcal{R}'} a(R', r_q) X_{i_q - r_q}(I' - R') + E_{i_q}(I'). \tag{16}$$

Now we introduce the generalization of the shift operator. For each vector $R' = (r_1, r_2, ..., r_{q-1})$ of non-negative components, define the shift operator $\mathcal{S}_{R'}$ to operate on $(q-1)$-dimensional arrays V of size $N_1 \times N_2 \times \cdots \times N_{q-1}$ as follows: $\mathcal{S}_{R'} V$ is vector of the same dimensional structure as V such that $(\mathcal{S}_{R'} V)(I') = V(I' - R')$ for all $I' \in \mathcal{I}'$. Therefore, $X_{i_q}(I' - R') = (\mathcal{S}_{R'} X_{i_q})(I')$. Note that if $r_l \geq N_l$ for some l, then $\mathcal{S}_{R'}$ is the zero operator, that is, $\mathcal{S}_{R'} V$ is the zero array of size $N_1 \times N_2 \times \cdots \times N_{q-1}$. Note also that if R' and S' are two index vectors in $\mathcal{R}'$, then $\mathcal{S}_{R'} \mathcal{S}_{S'} = \mathcal{S}_{R'+S'}$. Indeed, we can correspond to $\mathcal{S}_{R'}$ a polynomial term $x_1^{r_1} x_2^{r_2} ... x_{q-1}^{r_{q-1}}$ of $q-1$ variables; the product (i.e., the composition) of two operators $\mathcal{S}_{R'} \mathcal{S}_{S'}$ is equivalent to the product of their corresponding polynomial terms. Hence, one can define "multivariable" polynomial operators.

Definition 3 *Given a $(q-1)$-dimensional real array $T(I')$, $I' \in \mathcal{I}'$, the operator $\mathcal{T} = \sum_{I' \in \mathcal{I}'} T(I') \mathcal{S}_{I'}$, called the multivariable polynomial operator of characteristic array T, is such that for any array V of size $N_1 \times N_2 \times \cdots \times N_{q-1}$, $\mathcal{T}V = \sum_{I' \in \mathcal{I}'} T(I')(\mathcal{S}_{I'} V)$.*

Equation 16 can now be put in the following form:

$$X_{i_q}(I') = (\sum_{R' \in \mathcal{R}'^*} a(R', 0) \mathcal{S}_{R'} X_{i_q})(I') +$$

$$\sum_{r_q=1}^{t_q}\Big(\sum_{R'\in\mathcal{R}'} a(R',r_q)\mathcal{S}_{R'}X_{i_q-r_q}\Big)(I') + E_{i_q}(I'). \quad (17)$$

Since the vector-index I' is the same in all the terms of the previous equation, we can drop it resulting in a compact $(q-1)$-dimensional array equation

$$X_{i_q} = \sum_{R'\in\mathcal{R}'^*} a(R',0)\mathcal{S}_{R'}X_{i_q} + \sum_{r_q=1}^{t_q}\sum_{R'\in\mathcal{R}'} a(R',r_q)\mathcal{S}_{R'}X_{i_q-r_q} + E_{i_q}. \quad (18)$$

Next, move the first summation on the right hand side to the left, and factor X_{i_q} out:

$$\Big(1-\sum_{R'\in\mathcal{R}'^*} a(R',0)\mathcal{S}_{R'}\Big)X_{i_q} = \sum_{r_q=1}^{t_q}\sum_{R'\in\mathcal{R}'} a(R',r_q)\mathcal{S}_{R'}X_{i_q-r_q} + E_{i_q}. \quad (19)$$

We simplify the notation by defining

$$\mathcal{T}_0 = 1 - \sum_{R'\in\mathcal{R}'^*} a(R',0)\mathcal{S}_{R'},$$

$$\mathcal{T}_{r_q} = \sum_{R'\in\mathcal{R}'} a(R',r_q)\mathcal{S}_{R'}, \ for\ r_q = 1,2,...,t_q.$$

As a result, equation 19 becomes $\mathcal{T}_0 X_{i_q} = \sum_{r_q=1}^{t_q}\mathcal{T}_{r_q}X_{i_q-r_q} + E_{i_q}$, leading to

$$X_{i_q} = \sum_{r_q=1}^{t_q}\mathcal{T}_0^{-1}\mathcal{T}_{r_q}X_{i_q-r_q} + \mathcal{T}_0^{-1}E_{i_q} \quad (20)$$

The next theorem combines and generalizes Theorems 1 and 2 to multivariable polynomial operators. We first define partial multidimensional convolution.

Definition 4 *The partial convolution of two $N_1\times N_2\cdots\times N_{q-1}$ arrays U and V is an $N_1\times N_2\cdots\times N_{q-1}$ array W, denoted $W = U\otimes V$, such that* $W(i_1,...,i_{q-1}) = \sum_{r_1=0}^{i_1}\cdots\sum_{r_{q-1}=0}^{i_{q-1}} U(r_1,...,r_{q-1})V(i_1-r_1,...,i_{q-1}-r_{q-1})$.

Theorem 3 *1) Let T and V be two $N_1\times N_2\times\cdots\times N_{q-1}$ arrays, and $\mathcal{T}$ the multivariable polynomial operator of characteristic array T. Then, $\mathcal{T}V = T\otimes V$.*
2) $\mathcal{T}_0^{-1} = \sum_{I'\in\mathcal{I}'} C(I')\mathcal{S}_{I'}$, where C is an $N_1\times N_2\times\cdots\times N_{q-1}$ array that satisfies the following $(q-1)$-index recurrence relation of order $(t_1,t_2,...,t_{q-1})$:

$$C(0,0,...,0) = 1, \quad C(I') = \sum_{R'\in\mathcal{R}'} a(R',0)C(I'-R') \quad (21)$$

3) For every $i_q = 1,2,...,t_q$, $\mathcal{T}_0^{-1}\mathcal{T}_{i_q} = \sum_{I'\in hIp} B_{i_q}(I')\mathcal{S}_{I'}$, where

$$B_{i_q} = C\otimes a_{i_q}, \quad (22)$$

and a_{i_q} is an $N_1\times\cdots\times N_{q-1}$ array such that $a_{i_q}(i_1,...,i_{q-1}) = a(i_1,...,i_{q-1},i_q)$ if $(i_1,...,i_{q-1})$ is in $\mathcal{R}'$, and equal to zero otherwise.

Proof: The proof of (1) is similar to the proof of Theorem 1. The proof of (2) and (3) follows the same reasoning as in the proof of Theorem 2. □

Now, by using the previous theorem, we convert equation 20 to a form identical to equations (8) (correspond i_q to j and r_q to s):

$$X_{i_q} = \sum_{r_q=1}^{t_q} B_{r_q}\otimes X_{i_q-r_q} + F_{i_q} \quad (23.a)$$
$$F_{i_q} = C\otimes E_{i_q}. \quad (23.b)$$

Clearly, equation (23.a) is a 1-indexed recurrence relation of order t_q, where every term is an $N_1\times N_2\times\cdots\times N_{q-1}$ array. This completes the dimension shifting from index dimension to term dimension. The shifting of the order dimension to term dimension is the same as in the previous section starting from equation 9 onward. The resulting 1-index recurrence relation of order 1, similar to equation 9, is

$$\overline{X}_{i_q} = A\odot\overline{X}_{i_q-1} + \overline{F}_{i_q} \quad (24)$$

The definitions of A, $\overline{X}_{i_q}$, $\overline{F}_{i_q}$, F_{i_q,r_q}, and $\odot$ are formally the same as in the previous section, with the proper substitutions. Note in particular that A is a $t_q\times t_q$ matrix where every term $A_{i,s}$ is an $N_1\times N_2\times\cdots\times N_{q-1}$ array, and the convolution $\otimes$ involved in the definition of $\odot$ is now a $(q-1)$-dimensional convolution on $N_1\times N_2\times\cdots\times N_{q-1}$ arrays. The architecture, data assigment and the parallel algorithm follow.

Architecture

The architecture is naturally an $N_1\times N_2\times\cdots\times N_q$ mesh of hypercubes or of partitionable buses. The mesh can be viewed as N_q *submeshes*, each being an $N_1\times N_2\times\cdots\times N_{q-1}$ mesh. The submeshes are labeled $0,1,...,N_q-1$. On the other hand, the whole mesh can be viewed as partitioned into N_q/t_q contiguous *subsystems* of t_q contiguous submeshes each. The subsystems are labeled $1,2,...,N_q/t_q$, and the submeshes within each subsystem are labeled locally $1,2,...,t_q$.

Data Assignment

Assume that each subsystem has the parameter array a such that the (sub)array a_{r_q} is stored in the r_q-th submesh of each subsystem, one term per processor. In addition to array a_1, submesh 0 of the mesh has the array a_0. The arrays E_{i_q} and F_{i_q} are stored in submesh i_q, that is, $E(i_1,i_2,...,i_q)$ and $F(i_1,i_2,...,i_q)$ are in processor $(i_1,i_2,...,i_q)$. Each local submesh r_q of each subsystem i_q hosts arrays B_{r_q}, A_{i,r_q} for all $i = 1,2,...,t_q$, and F_{i_q,r_q}.

Algorithm General(**input:** E,a; **output:** X)
begin

1. Submesh 0 calls the algorithm "General" recursively to compute $C(0:N_1,...,0:N_{q-1})$ by solving the $(q-1)$-indexed scalar recurrence relation 21 of order $(t_1,t_2,...,t_{q-1})$. Afterwards, C is broadcast along dimension q to all the submeshes.
2. **for** i_q=0 **to** N_q-1 **pardo** /* Compute F */
3. Submesh i_q does: $F_{i_q} = C\otimes E_{i_q}$ by a parallel convolution algorithm;
   endfor
4. **for** $r_q = 1$ **to** t_q **pardo**
5. Local submesh r_q of each subsystem does: $B_{r_q} = C\otimes a_{r_q}$; $A_{t_q,r_q} = B_{r_q}$;

endfor

6. **for** $i = t_q$ **down to** 2 **do** /* Compute A */
7. Local submesh 1 of each subsystem broadcasts $A_{i,1}$ along dimension q to all the submeshes of its subsystem;
8. Each submesh $r_q = 2, 3, ..., t_q$ in each subsystem sends the vector array A_{i,r_q} one step down along dimension q in the mesh; /* Steps 7 and 8 deliver data to processors to compute A next. */
9. **for** $r_q = 1$ **to** t_q **pardo**
10. Local submesh r_q of each subsystem does: $A_{i-1,r_q} = B_{r_q} \otimes A_{i,1} + A_{i,r_q+1}$;

endfor

endfor

11. **for** i_q=1 **to** N_q/t_q **pardo** /* Compute $\overline{F}$ */
12. **for** r=1 **to** t_q **pardo**
13. Local submesh r of subsystem i_q does: it gets $F_{t_q i_q - r_q}$ from local submesh r_q of subsystem i_q, for $r_q = r, r+1, ..., t_q$; it then computes $F_{i_q,r} = F_{t_q i_q - r} + \sum_{r_q=r+1}^{t_q} A_{t_q+r+1-r_q,1} \otimes F_{t_q i_q - r_q}$;

endfor

endfor

14. All the subsystems of the mesh work together to solve the 1-indexed recurrence relation $\overline{X}_{i_q} = A \odot \overline{X}_{i_q-1} + \overline{F}_{i_q}$ in parallel, where $\overline{X}_{i_q}$ resides in subsystem i_q;

end

Analysis

Let $T(N_1, t_1, N_2, t_2, ..., N_q, t_q)$ be the parallel time complexity of the whole algorithm, and let $N = N_1 N_2 ... N_q$. Step 1, being a recursive step, takes $T(N_1, t_1, N_2, t_2, ..., N_{q-1}, t_{q-1})$ time. The time of all the remaining steps is dominated by the last step, step 14, which is the solution of the 1-indexed recurrence relation of order 1, equation 24, and thus takes $O((T_\odot + T_+) \log \frac{N_q}{t_q})$ time. Note that $\odot$ is a matrix-vector product where the matrix is $t_q \times t_q$ and the multiplication operation is replaced by $(q-1)$-dimensional $\otimes$. Multidimensional convolution can be implemented by multidimensional FFT, performed one dimension after another, in $O(\log N_1 + \log N_2 + \cdots + \log N_{q-1}) = O(\log(N_1 N_2 ... N_{q-1}))$ time because FFT in dimension l takes $O(\log N_l)$ time. Since $T_\odot = O(t_q T_\otimes)$, it follows that $T_\odot = O(t_q \log(N_1 n_2 ... N_{q-1}))$. Since also $T_+ = O(1)$, it follows that step 14 takes

$$O(t_q \log(N_1 N_2 ... N_{q-1}) \log(N_q/t_q)).$$

Consequently, we have

$$T(N_1, t_1, ..., N_q, t_q) = T(N_1, t_1, ..., N_{q-1}, t_{q-1}) + O(t_q \log(N_1 ... N_{q-1}) \log(N_q/t_q)).$$

This recurrence relation easily yields that

$$T(N_1, t_1, ..., N_q, t_q) = T(N_1, t_1) + O(\sum_{l=2}^{q} t_l \log(N_1 ... N_{l-1}) \log(N_l/t_l))$$

Clearly, $T(N_1, t_1) = O(t_1 \log \frac{N_1}{t_1})$ because it corresponds to solving a scalar 1-indexed recurrence relation of order t_1. Since $\log(N_1 ... N_{l-1}) \log(N_l/t_l) \leq \log N \log(N_l/t_l) \leq \log N \log N_l \leq \log^2 N$, the time formula for the whole algorithm simplifies to

$$T(N_1, t_1, N_2, t_2, ..., N_q, t_q) = O((\sum_{l=1}^{q} t_l) \log^2 N).$$

Since the sequential time of solving equation 14 is $O(t_1 ... t_q N)$, it follows that the speedup is $O(\frac{t_1 t_2 ... t_q}{\sum_{l=1}^{q} t_l} \frac{N}{\log^2 N})$, and the efficiency is $O(\frac{t_1 t_2 ... t_q}{(\sum_{l=1}^{q} t_l) \log^2 N})$, where N is the size of the output array X. In the typical cases where the t_l's are small constants, the overall time becomes $O(\log^2 N)$, and the speedup and efficiency become $O(N/\log^2 N)$ and $O(1/\log^2 N)$, respectively.

5 Conclusions

In this paper we developed a dimension-shifting approach for novel parallel solution of multi-indexed recurrence relations of arbitrary order in square-logarithmic time, using a mesh of hypercubes or of partitionable buses. Multi-indexed recurrence relations have not been addressed before, but hold special significance due to their application to DPCM, which is a standard image compression technique.

For future work, it is of practical interest to consider how best to load-balance and possibly pipeline the algorithm "General" on a system of fewer processors than the size of the output X, for the purpose of minimizing the parallel time complexity.

References

[1] S. Chen and D. J. Kuck, "Time and Parallel Processor Bounds for Linear Recurrence Systems," *IEEE Trans. Comput.*, C-24(7), July 1975.

[2] J. W. Cooley and J. W. Tuckey, "An Algorithm for the Machine Calculation of Complex Fourier Series," *Math. of Comput.*, Vol. 19, pp. 297–301, 1965.

[3] R. Gonzales and R. Woods, *Digital Image Processing*, (Chapter 3) Addison-Wesley, 1992.

[4] L. Hyafil and H. T. Kung, "The Complexity of Parallel Evaluation of Linear Recurrences," *JACM*, 24(3), pp. 513–521, July 1977.

[5] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Trans. Comput.*, C-22(8), pp. 786–793, Aug. 1973.

[6] C. P. Kruskal, L. Rudolph and M. Snir, "The power of parallel prefix," *IEEE Trans. Comput.*' Vol. 34, pp. 965–968, 1985.

[7] B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*, Van Nostrand Reinhold, New York, 1993.

[8] K. Sayood, *Introduction to Data Compression*, Morgan Kauffmann Publishers, San Fransisco, California, 1996.