Optimal Mappings of m Dimensional FFT Communication to k Dimensional Mesh for Arbitrary m and k

Z. GEORGE MOU and XIAOJING WANG

Department of Computer Science and Center for Complex Systems
Brandeis University, Waltham, MA 02254, USA
{mou,wang}@cs.brandeis.edu

Abstract. The FFT communication patterns are important to not only FFT algorithms, but also many other algorithms over one or higher dimensional. The mapping of m dimensional FFT communication to k dimensional mesh has previously been considered only for the following special cases (a) m = 1 or 2, k =1 or 2, (b) m = 1 or 2, $k = \log(n)$ where n is the size of the machine. In this paper, we present the optimal mappings of m dimensional FFT communication onto k dimensional mesh for arbitrary m and k. The mappings are optimal since the communication distances in the logarithmic steps sum to exactly the diameter of the mesh regardless of the dimension or the shape of the mesh. An m-k shuffle permutation, which subsumes perfect shuffle, is introduced and used to derive some of the optimal mappings. As a by-product, an optimal broadcast algorithm over any dimensional mesh, including binary hypercube as a special case, is presented.

1 Introduction

The butterfly communication patterns found in FFT algorithms, which we will refer to as *FFT communications*, are important to not only the FFT algorithms but also many other divide-and-conquer algorithms for a broad class of numerical problems including reduction, scan, polynomial evaluation, linear difference equations, and bitonic sort. On the other hand, mesh is an important underlying topology for a wide range of parallel architecture, ranging from the one dimensional linear array to the binary hypercube which is no more than a mesh of logarithmic dimensional mesh. The study of the mappings from FFT communication to mesh architectures however has been very limited in the past. Let m be the dimension of the FFT communications, and k the dimension of the mesh, then the previous studies only cover the following special cases:

- $m = 1, 2, k = 1, 2$ [3, 1, 2].
- $m = 1, 2, k = \log(n)$ [11, 13, 5, 12, 10].
- $m = 1, 1 \leq k \leq \log(n)$ [7, 8].

For the case of $k = \log(n)$, we have included Stone's work on perfect shuffle [12] and Preparata's work on cube-connected-shuffle [10] since the well-known isomorphism between the two architectures and a binary hypercube [14]. It thus can be observed that the mapping has not been studied for high dimensional FFT and the meshes with the dimensionality between the two extremes - k = 1 and 2 and $k = \log(n)$. Moreover, there is a lack of general approach to the problem for different dimensions of the FFT and the mesh, and the inherent connections between the results for different dimensions cannot be easily seen.

In this paper, we study the mapping from m dimensional (m-d) FFT to k dimensional (k-d) mesh for arbitrary m and k. The main results generalize the previous work in [1, 3, 2, 11, 13, 7, 8] and include

- A unified framework to study the mapping for arbitrary m and k.
- A family of $(\log(n))!$ mappings for arbitrary dimensions of the FFT and the mesh with size n.
- Establish the lower bound of the communication cost, and prove the optimality of the family of mappings.

As a by-product of the proof in Section 3, we present optimal broadcast algorithms which can be used to (1) broadcast over k-d mesh for any k, including binary hypercube as a special case; (2) broadcast over a given dimension of an m dimensional array distributed over a k-d mesh for any m and k. This result thus generalizes the well-known result on broadcasting in [4].

This paper is organized as follows. We begin by introducing some basic notions about mesh and FFT communication in Section 2. In Section 3, we establish the lower bound of the FFT communication on mesh. Section 4 and 5 introduces two specific mappings from FFT communications to mesh for arbitrary dimensions, present the implementations of the communications on mesh, and prove their optimality. In Section 6 we introduce a family of optimal mappings for which the previous two are members. In Section 7, we generalize the results in previous sections to allow arbitrary shape of the FFT array and the mesh. Some of the related results that cannot be reported hear are mentioned in Section 8, where other comments are made as well.

2 Preliminary

2.1 k dimensional mesh

The topology of the mesh A *k dimensional mesh* with the *shape* $P_{k-1} \times \ldots, P_0$ consists of processors in the set P^k of the form

$$P = \{p = (p_{k-1}, \ldots, p_0) \mid 0 \leq p_i \ P_{i-1}, \text{ for } i = 0 \text{ to } k-1\}$$

where P_i is the size of the mesh along the *ith dimension*, p_i the coordinate of processors p along the ith dimension. Two processors $u, v \in P^k$ are *directly connected*, denoted by uLv, if only only if their coordinates differ along one dimension by one, namely

$$uLv \Leftarrow |u_i - v_i| = 1, \text{ and } u_j - v_j = 0 \leq i, j < k-1, j \neq i,$$

The total number of processors $s = |P| = \prod_{i=0}^{k-1}$ is referred to as the *size of the mesh*.

A *metric function* $\mathcal{D} : P^k \times P^k \rightarrow INT$ is introduced to measure the *distance between processors*. Given two processors $u = (u_0, u_1, \ldots, u_{k-1})$ and $v = (v_0, v_1, \ldots, v_{k-1})$ in P^k the function D is

$$\mathcal{D}(u, v) = \sum_{i=0}^{k-1} |u_i - v_i|$$

Univ. of Washington Libraries

The *diameter* of a mesh is defined to be

$$Min\{\mathcal{D}(x,y) \mid x,y \in P\}$$

It follows from the definitions that

Theorem 1.

- *the function D is a well-defined metric function over space P.*
- *when $k = 2$, D reduces to the Manhattan Distance over two dimensional grid space.*
- *when $k = log(n)$ D reduces to the Hamming Distance over the space of k-bit binary numbers.*
- *when the mesh is viewed as a graph, D is consistent with the length of shortest path between any two processors.*
- *The diameter of a k dimensional mesh is $\sum_{i=0}^{k-1}(P_i - 1)$ where P_i is the size of its ith dimension.*

A mesh is *regular* if all of its dimensions has the same size. It follows that a regular k dimensional size has the diameter of $k(n^{1/k} - 1)$.

Communications over mesh A *communication between two processors* u and v denoted by $z(u,v)$ means that a message is sent from processor u to processor v. The communication is *symmetric* if at the same time a message is sent from processor v to processor u.

A *communication over the mesh* with processor set P^k with respect to binary relation R over P^k denoted by Z^R is the collection of communications between two processors

$$Z^R = \{z(u,v) | uRv\}$$

The communication Z^R is symmetric if all its members are symmetric. It is asymmetric if none of its members is symmetric [1].

A d-step communication over P^k is a sequence of d communications of the form

$$(Z^{R_0}, Z^{R_1}, \ldots, Z^{R_{d-1}})$$

such that Z^{R_i} is performed before $Z^{R_{(i+1)}}$ for $0 \leq i < d-1$

An asymmetric communication Z^R over a k−d mesh is *uniform* if there exists a constant vector, referred to as the *communication vector*, $w = (w_0, \ldots, w_{k-1})$ such that pRp' if and only if $p' = p + w$. A symmetric communication is uniform if the relation R can be partitioned into two sub-relations R_1 and R_2 such that $R = R_1 \cup R_2$, and both Z^{R_1} and Z^{R_2} are uniform.

A special case of uniform communication is *single dimensional communication*, where the communication vector has the form of

$$(0, \ldots, w_i, 0, \ldots, 0)$$

In other words, the communication is sent from all the senders to all the receivers only along dimension i by distance w_i.

[1] A communication Z^R thus can be neither symmetric nor asymmetric by the above definition.

Cost of communications We measure the cost of a communication $z(u,v)$ between two processors (alone) denoted by $\mathcal{C}(z(u,v))$ by the distance between the two processors, namely

$$\mathcal{C}(z(u,v)) = \mathcal{D}(u,v)$$

The cost of a general communication over the entire mesh is difficult to determine because it depends on not only the distances the messages travel, but also the routing algorithms and if there is message congestion under the routing algorithm.

However the *cost of a uniform communication* with communication vector $w = (w_{k-1}, \ldots, w_0)$ is simply the norm of

$$|w| = \sum_{i=0}^{k-1} |w_i|$$

which reflects the minimum number of communication links that the messages go through. This cost is realistic under many possible routine algorithms. A straightforward one is that route the messages from dimension 0 by distance w_0, dimension 1 by distance $w_1, \cdots$, dimension $(k-1)$ by distance w_{k-1}. The communication cost along dimension i costs exactly w_i since all messages can travel with no interferences with each due to the uniformity. It follows that the cost of single dimensional communication with the communication vector $(0, \ldots, w_i, 0, \ldots, 0)$ is simply w_i.

Finally, the cost of a d-step communication over P^k is the sum of the cost of all the d steps, i.e.

$$\mathcal{C}(Z^{R_0}, Z^{R_1}, \ldots, Z^{R_{d-1}}) = \sum_{i=0}^{d-1} \mathcal{C}(Z^{R_i})$$

Note that although the order of the communication steps are generally important in computing, it is of no importance as far as the cost is concerned since addition is commutative. Therefore, two d-step communications

$$(Z_0^{R_0}, Z_0^{R_1}, \ldots, Z_0^{R_{d-1}}) \text{ and } (Z_1^{R_0}, Z_1^{R_1}, \ldots, Z_1^{R_{d-1}})$$

are *cost-equivalent* if and only if they contain the same set of constituents.

Communication instructions The above discussion is true for both MIMD and SIMD architectures. In the case of k-d SIMD architecture, we assume the *communication instruction*

$$d\ (w_{k-1}, \ldots, w_0)$$

which sends a message from each active processors processor $u = (u_0, \ldots, u_{k-1})$ to processor $v = (u_{k-1}+w_{k-1}, \ldots, u_{k-1}+w_{k-1})$. A uniform communication with communication vector $w = (w_{k-1}, \ldots, w_0)$ thus can be *implemented* by $d(z) = d(z_0, \ldots, z_{k-1})$ [2] Single dimensional communication on mesh can be expressed more concisely by the instruction

$$sd\ (i, w_i)$$

[2] Note that this implementation may cause messages sent between unrelated processors but will definitely deliver all the messages to be sent. The unwanted messages can be either prevented by a predicate over the processors or simply ignored after the reception, which is often the cheaper approach in practice

CMU Washington Libraries

where the $0 \leq i < (k-1)$ is a dimension, $0 \leq w_i < P_i$ is the communication distance along that dimension i with size P_i.

Finally, we would like to emphasize that the discussions of this section applies to binary hypercube, which is no more than a regular k dimensional mesh with n processors where $k = \log(n)$.

2.2 M dimensional FFT communication

An m dimensional FFT communication is defined over an *m dimensional index set* I^m of the form

$$I^m = \{(i_0, i_1, \ldots, i_{m-1}) \mid 0 \leq i_j < A_j, \text{ for } j = 0 \text{ to } m\}$$

where A_j is the size along the dimension j. The *shape* of the index set is $A_0 \times \cdots A_{m-1}$, and the *size* of the index set is $n = \sum_{j=0}^{m-1} A_j$. The index set is said to be *regular* if all dimensions have the same size, namely, $A_j = A_i$ for all $0 \leq i, j < m$.

A communication between two indices $x, y \in I$, denoted by $c(i, j)$, means a value is sent from x to y. The communication is bidirectional if at the same time a message is sent from y to x.

A communication over an m dimensional index set I^m with respect to a binary relation over I^m denoted by C^R is a collection of communication between two indices $x, y \in I$ of the form

$$C^R = \{c(x, y) \mid xRy\}$$

A d-step communication over the index set I^m is a sequence of communications over I^m of the form

$$(C_{R_0}, C_{R_1}, \ldots, C_{R_{d-1}})$$

such that C_{R_i} is performed before $C_{R_{i+1}}$ for $0 \leq i < d-1$.

We next define a special binary relation over the index set I^m. Given two indices $x, y \in I^m$

$$x = (x_0, \ldots, x_p, \ldots x_{m-1}) \text{and} \quad y = (y_0, \ldots, x_p, \ldots, y_{m-1})$$

we say x and y are *p-q related* denoted by $x\ R_p^q y$ if and only the binary numbers of their pth dimensional index x_p and y_p differ in their qth least significant bits. Observe that this relation is symmetric by definition. It can be decomposed into two disjoint relations $R_p^q(0)$ and $R_p^q(1)$, where x is $R_p^q(0)((1))$ related to y if they are p-q related, and the qth bit of x_p is 0 (1), of y_p is 1 (0).

A *p-q communication* denoted by $C^R(p, q)$ over an index set is a collection of communications (to be performed in parallel) of the form

$$C^R(p, q) = C^{R_p^q} = \{c(x, y) \mid x R_p^q y\}$$

An *m dimensional FFT communication over* index set I^m with dimension size A consists of $(m \times \log(A))$-step p-q communications. The communication can be of *postmorphism* or of *premorphism* [9]. A postmorphism m dimensional communication is performed from the least significant bit to the most significant bit for each dimension

$$\begin{array}{l}(C^R(0,0), \ldots, C^R(0, a_0 - 1),\\ C^R(1,0), \ldots, C^R(1, a_1 - 1),\\ \ldots, \ldots\\ C^R(m-1, 0) \ldots C^R(m-1, a_{m-1} - 1))\end{array}$$

where $a_j = \log(A_j)$

In contrast, a premorphism m dimensional communication is performed from the most significant bit to the least significant for each dimension:

$$\begin{array}{l}(C^R(0, a_0 - 1), \ldots, C^R(0, 0),\\ C^R(1, a_1 - 1), \ldots, C^R(1, 0),\\ \ldots, \ldots\\ C^R(m-1, a_{m-1} - 1) \ldots C^R(m-1, 0))\end{array}$$

It should be pointed out that the m-d FFT communication is symmetric. More precisely, each step of the communication based on the relation R_p^q can be decomposed into two asymmetric communication, one is based on the relation $R_p^q(0)$, and another on the relation $R_p^q(1)$.

2.3 Mappings

Given an m-d index set I^m and the processor set P^k of a k dimensional mesh, a *mapping* $f : I^m \rightarrow P^k$ is an assignment from the set of indices to the set of processors. For the discussion of this section we assume the *mapping ratio* $R = |I^m|/|P^k| = 1$ and f is bijective.

A mapping f maps a communication c(x, y) between two indices $x, y \in I^m$ to a communication $z(f(x), f(y))$ between two processors $u, v \in P^k$ where u = f(x), and v = f(y), which we denote by $z(u, v) = f(c(x, y))$. Similarly, f maps a communication over an index set C^R to a communication Z^R over the mesh

$$Z_R = \{(z(f(x), f(y)) \mid xRy\}$$

which we denote by $Z^R = f(C^R)$.

The mapping f is a *uniform mapping* for a communication C^R over an index set if $Z^R = f(C^R)$ is uniform.

Given a d-step communication over an index set I^m

$$(C^{R_0}, C^{R_1}, \ldots, C_{R_{d-1}})$$

and a mapping f from I^m to P^k, we say f maps the d-step communication over I^m to a d-step communication over P^k

$$(f(C^{R_0}, f(C^{R_1}), \ldots, f(C^R_{d-1}))$$

The following proposition allows us to focus on the study of one instead of both of the two types of m-d FFT communications.

Theorem 2. *Let C and C' be respective the post- and pre- morphism FFT communications over the index set I^m, $Z = f(C)$ and $Z' = f(C')$ are the communications over P^k where f is a mapping function from I^m to P^k, then Z and Z' are cost equivalent.*

Proof Follows from the definitions.

Note that there was no notion of cost for communication over index set. With a mapping from index set to processors in meshes, we can now define the *cost of communications over index set under the mapping*. Given a mapping $f : I^m \rightarrow P^k$, the cost of a communication between two indices $x, y \in I^m$ is the cost of the communication between the two processors $f(x), f(y) \in P^k$, the cost of a communication C^R over I^m is the cost of the communication $f(C^R)$ over P^k, the cost of a d-step communication C is the cost of the d-step communication $f(C)$.

Univ. of Washington Libraries

3 The lower bound of m-d FFT communication on k-d mesh

To show the optimality of the mappings, we need to establish the lower bound of the FFT communications on a k-d mesh. This section is to establish the lower bound for FFT communication through the problem of broadcast over m-d arrays, and show how broadcast can be reduced to the FFT communication.

Given an index set I^m, and an index $x \in I^m$, the problem of broadcast is to send a value associated with x to all other indices directly or indirectly. For simplicity of the discussion, we first assume x = $(0, \ldots, 0)$, the index with all 0's along all dimensions.

We next define a simpler problem over m-d index set called *broadcast along a given dimension*. Given an index set I^m, the problem of broadcast along its pth dimension is to send a value from each index x of the form

$$x = (x_{m-1}, \ldots, 0_p, \ldots, x0)$$

where $0 \leq x_j < A$ to the A indices in the set

$$\{x = (x_{m-1}, \ldots, X_p, \ldots, x0) \mid 0 \leq X < A\}$$

Lemma 3. *The problem of broadcast over I^m can be solved by solving m broadcasts along the m dimensions in any order.*

Proof Without loss of generality, we assume the m broadcast is done in the order of dimension 0, dimension 1, $\cdots$, and finally dimension $(m-1)$.

After the first step, all indices in the following set

$$\{(0, \ldots, 0, X) \mid 0 \leq X < A\}$$

will have the value. And after the ith step, all indices in the index set of

$$\{(0, \ldots, X_{i-1}, \ldots, X_0) \mid 0 \leq X < A\}$$

will have the value. Therefore, after m steps, all indices will have the value.

In the following, we refer to the $\log(A)$-step communication based on the relation $R_(p, q)$ for a fixed p, and $q = 0$ to $(\log(A)-1)$ the m-d FFT communication (restricted) on the pth dimension.

Lemma 4. *The m-d FFT communication over I^m along the pth dimension can be used to achieve broadcast along the pth dimension.*

Proof The pth demensional FFT communication over array of shape A^m consists of A parallel FFT communications over vector of size A, each of them can be used to broadcast along one vector (see Figure 1 and [6].)

In order to complete the discussion of this section, we need to further establish the following fact

Lemma 5.

- *A one dimensional FFT communication can be used to broadcast from any index to all other in the vector.*

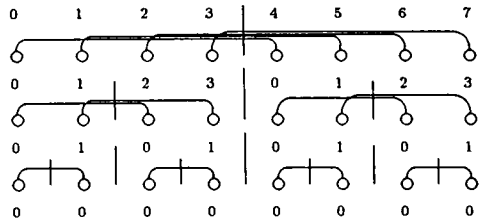

Fig. 1. A broadcast over vector with FFT communication

- *The m dimensional FFT communication can be used to broadcast from any index to all others in the array.*

Note that the second part of the lemma follows from the first. The first part can be proven by the construction of a algorithm with the FFT communication, which we omitted to save the space.

Theorem 6. *The lower bound for the m dimensional FFT communication over I^m on a k dimensional mesh P^k of shape $S_{k-1} \times \cdots, S_0$, where $|I^m| = |P^k|$,is the diameter of the mesh, i.e. $\sum_{i=0}^{k-1}(S_{k-1} - 1)$.*

Proof Since FFT communication can be used to broadcast any index to all others in I^m, no matter what the mapping $f : I^m \rightarrow P^k$ is, it can be used to send a value from a corner processor to a processor at the opposite corner with the distance equal to the diameter.

4 The Identical Mapping

The mapping f from an m-d index set I^m to a k-d mesh with processors P^k has the functionality of $f : I^m \rightarrow P^k$. In this section, we assume

- The mapping ratio is one, thus $|I^m| = |P^k|$.
- The mesh is regular, thus it has the shape of $S \times \cdots \times S = |P^k|$.

We introduce the following operations over binary numbers and integers:

Coding $B(i)$ where i is an integer returns i's binary number.
Decoding $B^{-1}(b)$ where b is a binary number returned the integer such that $B(B^{-1}(b)) = b$.
Concatenation $\text{cat}(b_{m-1}, \ldots, b_0) = b$, where b_i are binary numbers of w bits, b is a binary number of m*w bits whose first leftmost w bits are those of b_0, second leftmost w bits are those of b_1, and so on.
Decompose $\text{dec}(b, m) = (b_{m-1}, \ldots, b_0)$, the inverse of the concatenation operation cat.
Set $\text{set}(b, k, x) = b'$, where b, b' are binary numbers, k an integer, x a binary bit, b' is the same as b except that the kth least significant bit is set to x.
Quotient $Q(i, j)$ returns the quotient of the division i/j.
Remainder $R(i, j)$ returns the remainder of the division i/j.

Scalable Concurrent Libraries

The mapping we introduce is given by Algorithm 1. It can be thought as taking the bits from the indices along all dimensions, concatenate them into one binary number, and then take it apart along the dimensions. It is further illustrated in Figure 4. Since the bit positions are not shuffled in any way before the concatenation, it is referred to as the *identical mapping.*

Algorithm 1 *The identical mapping*
Input: index $(i_{m-1},\ldots,i_0)$ *in* I^m *of shape* A^m.
Output: coordinate $(x_{k-1},\ldots,x_0)$ *in* P^k *of shape* S^k.

$s = \log(S)$, $I = cat(B(i_{m-1},\ldots,B(i_0))$ $(x_{k-1},\ldots,x_0) = dec(I,s)$,
return $(x_{k-1},\ldots,x_0)$

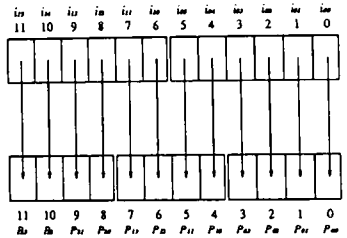

Fig. 2. An example of the identical mapping for n=12, m=2, k=3

The following proposition answers how the FFT communication is mapped to the communication on mesh under the identity mapping.

Theorem 7. *Let A and S be the dimension sizes of the index set* I^m *and mesh* P^k *respectively,* $a = \log(A)$, $s = \log(S)$, *then the identical mapping* $f : I^m \Rightarrow P^k$ *maps a step of FFT communication* $C^R(p,q)$ *over* I^m *to the single dimensional uniform mesh communication of the form* $sd(i,w)$, *where*

$$i = Q(p*a+q)/s \text{and} \quad w = 2^{R(p*a+q)/s}$$

Proof By the definition of relation $R(p,q)$, two communicating indices differ only in their qth bit of pth dimension. The identity mapping maps that bit to the wth bit of the ith dimension, the communication is thus sd (i,w).

We can then define a function $g_{id}(p,q) = (i,w)$ called the *translating function* associated with the mapping f from I^m to P^k, which takes p and q, returns p and w by the algorithm given in Theorem 7. The following algorithm then implement the postmorphism m-d FFT communication on a k dimension mesh under the identical mapping

Algorithm 2 *m-d FFT postmorphism communication under identity mapping with translating function*

for $p = 0$ *to m-1*
  for $q = 0$ *to* $\log(A) - 1$
    $sd \circ g_{id}\ (p,q)$

A disadvantage of the above algorithms is that the translating function is being computed for each value of p and p, and the computing is fairly expensive. The following algorithm exploit the regularity of the translation when p and q changes from one value to the next, and reduces the computation of the translation function to some simpler operations:

Algorithm 3 *m-d FFT postmorphism communication without translating function*

Max_Bit $=\log(P)$, *dim = 0, dis = 1 bit_ptr = 0;*
for $i = 0$ *to* $m * \log(A) - 1$
  sd (dim, dis)
  bit_ptr = bit_ptr +1
  if bit_ptr = Max_Bit then dim = dim +1, bit_ptr = 0, dis = 1
  else bit_ptr = bit_ptr +1, dis = 2 dis*

A similar program can be written for a premorphism m dimensional FFT communication.

We give an example in Figure 4 where a two dimensional postmorphism FFT communication is translated into the communications of communications on a three dimensional regular mesh for $n = 4k$ under the identical mapping.

	q = 0	q = 1	q = 2	q = 3	q = 4	q = 5
p = 0	sd(0,1)	sd(0,2)	sd(0,4)	sd(0,8)	sd(1,1)	sd(1,2)
p = 1	sd(1,4)	sd(1,8)	sd(2,1)	sd(2,2)	sd(2,4)	sd(2,8)

Fig. 3. The identical mapping for m = 2, k=3, and size = 4k

It can be easily verified the total communication distance in the 12 steps of the communication of the above example is

$$3 \times \sum_{i=0}^{3} 2^i = 3 \times 15 = 45$$

which exactly equal to the diameter of the three dimensional mesh of the shape of $16 \times 16 \times 16$, the mapping is thus optimal. More general, we have

Theorem 8. *The cost of the m-d FFT communication over index set of size n on a k-d regular mesh of the same size under the identical mapping is the diameter of the mesh -* $k(n^{1/k} - 1)$.

Proof: The cost is simply the sum of all the single dimensional communications, namely

$$k \times \sum_{x=1}^{\log(n/k)} 2^x = k(n^{1/k} - 1)$$

Boston University Libraries

It follows from Propositions 8, 1, and 6 that

Theorem 9. *The identical mapping from m-d FFT communication over I^m to a k-d mesh P^k of the same size is optimal.*

5 The m-k Shuffle Mappings

In this section, we introduce another mapping from m-d FFT communication to k-d mesh of the same size based on what we call *m-k shuffle permutation.*

A *m-k* shuffle can be thought as a way of dealing n cards $0 \ldots (n-1)$ from m dealers $M_0, \ldots, M_{m-1}$ to k gamblers $K_0, \ldots, K_{k-1}$. To begin, the n cards numbered from 0 to (n-1) are partitioned into m decks and dealer M_i holds the cards from $w_1 * itow_1 * (i+1) - 1$. The dealing of the cards then proceeds as follows:

1. The dealer M_0 starts dealing the k cards on the top of his deck to the k gamblers, the ith card to gambler K_i.
2. After dealer M_j finishes his turn, dealer M_{j+1} deals his cards the same way to the k gamblers, for $j = 0$ to $m-1$.
3. After dealer M_{k-1} finishes his turn of dealing, dealer M_0 starts again.
4. The dealing stops when all the cards are given out.

Now there are w_2 cards in each of the k gamblers, and the cards are ordered by the time they are received. We then merge the k decks of cards from the k gamblers into one deck. So that the cards from gambler K_i proceeds the cards from gambler K_{i+1} from $i = 0$ to $(k-2)$. This process clearly maps each card from a position number x before the dealing to a new position number y in the merged pile. It should be pointed out that *m-k* shuffles are generalizations of the *perfect shuffle* permutation where $m = 2$ and $k = 1$. An example for $m = 2$, and $k = 3$ is given in Figure 4.

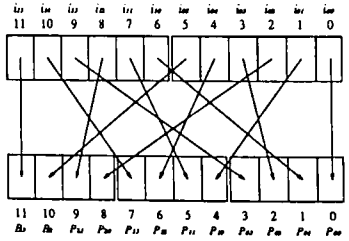

Fig. 4. An example of S_k^m for $n = 12, m = 2, k = 3$

Formally, an *m-k* shuffle S_k^m is a permutation over integers from 0 to $(n-1)$.

Theorem 10. *An m-k shuffle S_k^m over n integers, assuming that n is a multiple of m and k, n/m is a multiple of k, is the permutation over the integers $S_k^m(x) = y$ where*

$$y = R(x,k)w_2 + Q(R(x,w_1),k)m + Q(x,w_1)$$
where $w_1 = n/m, w_2 = n/k$

We next define a mapping of binary numbers based on *m-k* shuffle. Given a binary number $b = (b_{n-1}, \ldots, b_i, \ldots, b_0)$, the *m-k* shuffle of b is

$$S_k^m(b) = (b_{S_k^m(n-1)}, \ldots, b_{S_k^m(i)}, \ldots, b_{S_k^m(0)})$$

The *m-k shuffle mapping* from m-d FFT communication over index set I^m to k-d mesh with processors in P^k is different from the identical mapping only in that a *m-k* shuffle is performed over the concatenated binary number before it is partitioned into k dimensions:

Algorithm 4 *The m-k Shuffle Mapping*
Input: index $(i_{m-1}, \ldots, i_0)$ in I^m of shape A^m.
Output: coordinate $(x_{k-1}, \ldots, x_0)$ in P^k of shape S^k.

$s = \log(S)$, $I = cat(B(i_{m-1}, \ldots, B(i_0))$, $J = S_k^m(I)$, $(x_{k-1}, \ldots, x_0) = dec(J, s)$,
return $(x_{k-1}, \ldots, x_0)$

The following theorem answers how the FFT communication is mapped to the communication on mesh under the *m-k* shuffle mapping.

Theorem 11. *Let the m-k shuffle be the mapping from I^m of shape of shape A^m to mesh processor set P^k of shape S^k, $a = \log(A)$, $s = \log(S), n = \log(|I^m|), w_1 = n/m, w_2 = n/k$, then a step of FFT communication $C^R(p,q)$ over I^m is mapped to the single dimensional uniform mesh communication of the form $sd(i,w)$, where*

$$i = R(x,k)$$
$$w = Q(R(x,w_1),k)m + Q(x,w_1)$$

Proof Omitted.

The above theorem also defines the translating function for the *m-k* shuffle mapping, namely

$$g_{(m,k)}(p,q) = (i,w)$$
where
$$i = R(x,k), w = Q(R(x,w_1),k)m + Q(x,w_1)$$

The *m*-d FFT communication can then be implemented by the following algorithm

Algorithm 5 *m-d FFT postmorphism communication under m-k shuffle mapping with translating function*

for $p = 0$ *to m-1*
  for $q = 0$ *to* $\log(A) - 1$
    $sd \circ g_{m,k}\ (p,q)$

To take advantages of the regularity of the translation, a postmorphism, the algorithm can be rewritten as

Algorithm 6 *m-d FFT postmorphism communication under m-k shuffle mapping without translating function*

Boston University Libraries

Max_Dim =k, dim = 0, dis = 1, bit_ptr = 0
for $i = 0$ *to* $m * \log(A) - 1$
  sd (dim, dis), dim = dim + 1, bit_ptr = bit_ptr +2
  if bit_ptr = Max_Bit then dim = dim +1, bit_ptr = 0, dis = 1
  else bit_ptr = bit_ptr +1, dis = 2 dis*

A similar program can be written for a premorphism m-d FFT communication.

We give an example in Figure 5 where a two dimensional postmorphism FFT communication is translated into communications on three dimensional regular mesh for $n = 4k$ under the m-k shuffle mapping.

	q = 0	q = 1	q = 2	q = 3	q = 4	q = 5
p = 0	sd(0,1)	sd(1,1)	sd(2,1)	sd(0,4)	sd(1,4)	sd(2,4)
p = 1	sd(0,2)	sd(1,2)	sd(2,2)	sd(0,8)	sd(1,8)	sd(2,8)

Fig. 5. The m-k shuffle mapping translation for m = 2, k=3, and size = 4k

To determine the cost the m-d FFT communication under the m-k shuffle mapping, we first show that

Theorem 12. *Let F be the m-d FFT communication over an index set I^m. f_{id} and $f_{(m,k)}$ respectively the identical mapping and the m-k shuffle mapping from I^m to a regular mesh of the same size, $f_{\text{id}}(F)$ and $f_{(m,k)}(F)$ the mapped mesh communications from the FFT communication. Then $f_{(m,k)})(F)$ and $f_{\text{id}}(F)$ are cost-equivalent.*

Proof Since they contain the same set of single dimensional communications, only in different orders.

It follows from Theorems 12 and 9 that

Theorem 13. *The m-k shuffle mapping from m-d FFT communication over I^m to a k-d regular mesh P^k of the same size is optimal.*

6 The bit-permuting family of optimal mappings

In the previous two sections we introduced two optimal mappings from m-d FFT communication to k-d mesh of the same size, and showed that both of them are optimal. In this section, we generalize the previous results by introducing a family of optimal mappings of which the two mappings – identical mapping and m-k shuffle mapping – are members.

Let I^m be an m-d index set of shape A^m, P^k the set of processors in a k-d regular mesh, $|I^m| = |P^k| = W, w = \log(W)$, Π a arbitrary permutation over w integers, and can also be applied to a w-bit binary number to obtain another w-bit binary number given by

$$\Pi(b_{w-1}, \ldots b_i, \ldots b_0) = (b_{\Pi(w-1)}, \ldots, b_{\Pi(i)}, \ldots, b_{\Pi(0)}$$

Algorithm 7 is a mapping based on the permutation Π, which we refer to as a *bit-permutation mapping based on* Π

Algorithm 7 *The bit-permuting mapping f_Π*
Input: index $(i_{m-1}, \ldots, i_0)$ in I^m of shape A^m.
Output: coordinate $(x_{k-1}, \ldots, x_0)$ in P^k of shape S^k.

$s = \log(S)$, $I = cat(B(i_{m-1}, \ldots, B(i_0))$, $J = \Pi(I)$, $(x_{k-1}, \ldots, x_0) = dec(J, s)$,
return $(x_{k-1}, \ldots, x_0)$

The following proposition tells how each step of FFT communication is translated into the mesh communications under the bit-permutation mapping.

Theorem 14. *Let the f_Π be the mapping from I^m of shape A^m to mesh processor set P^k of shape S^k, $a = \log(A)$, $s = \log(S)$, $\Pi(x) = y$, then a step of FFT communication $C^R(p,q)$ over I^m is mapped to the single dimensional uniform mesh communication of the form $sd(i,w)$, where i and w is the solution of the following linear equation group*

$$pa + q = x, \quad is + w = y$$

Proof Obvious.

We next show the above mapping is cost equivalent to identical mapping regardless of what the permutation Π is

Theorem 15. *Let F be the m-d FFT communication over an index set I^m. f_{id} and f_Π respectively the identical mapping and the mapping based on permutation Π from I^m to a regular mesh of the same size, $f_{\text{id}}(F)$ and $f_\Pi(F)$ the mapped mesh communications from the FFT communication. Then $f_{(m,k)})(F)$ and $f_{\text{id}}(F)$ are cost-equivalent.*

Proof Since they contain the same set of single dimensional communications in different orders.

It follows that for any permutation over $\log|I^m|$ integers we have

Theorem 16. *The mapping f_Π from m-d FFT communication over I^m to a k-d mesh P^k of the same size is optimal.*

Since an index set I^m of size N has $(\log(N))!$ different permutations we have

Theorem 17. *There are $(\log(N))!$ optimal bit-permutation mappings from an m-d FFT communication over I^m of size n to a k-d regular mesh P^k of the same size.*

Clearly, identical and m-k shuffle mappings are simply two members of the large family.

7 Generalizations

In the previous three sections, we have assumed the regularity of both index set and the mesh. In this section, we show that bit-permutation mappings can be generalized from arbitrarily shaped m-d FFT communication to arbitrarily shaped k-d mesh and are still optimal.

Let us first introduce an operation

$$\text{dec2}(b, i_{k-1}, \ldots, i_0) = (b_{k-1}, \ldots, b_0)$$

Libraries

It takes a binary number b and k integers $i_{k-1}, \ldots, i_0$ as arguments, returns $k-1$ binary numbers, $b_{k-1}, \ldots, b_0$ such that b_0 contains the i_0 leftmost bits of b, b_1 the next i_1 leftmost bits of b, and so on. Clearly, the operation dec in Section 4 can be viewed as a special case of the operation dec2 where all the k integers take the same value.

We now define a mapping from the index set I^m of the shape $A_0 \times A_{m-1}$ to a k-d mesh with processors in P^k of the shape $S_0 \times S_{k-1}$ based on an arbitrary permutation over $\log(N)$ integers where $N = |I^m| = |P^k|$.

Algorithm 8 *The mapping for arbitrary shaped index set and mesh Input: index* $(i_{m-1}, \ldots, i_0)$ *in* I^m *of shape* A^m.
Output: coordinate $(x_{k-1}, \ldots, x_0)$ *in* P^k *of shape* S^k.
Notation: $a_i = \log(A_i)$for $0 \leq i < m, s_i = \log(S_i)$for $0 \leq i < k$, Π *a permutation over* $(\sum_{i=0}^{m} A_i)$ *integers.*

$s = \log(S)$, $I = cat(B(i_{m-1}, \ldots, B(i_0))$, $J = \Pi(I)$, $(x_{k-1}, \ldots, x_0) = dec2(J, s_{k-1}, \ldots, s_0)$,
return $(x_{k-1}, \ldots, x_0)$

We make the following claims about the the above bit-permutation mapping from I^m to P^k of arbitrary shape,

Theorem 18.

- *it maps each step of m-d FFT communication* $C^R(p, q)$ *over* I^m *to a single dimensional uniform communication over* P^k.
- *the cost of the m-d FFT communication under the mapping equal to the diameter of the k-d mesh, namely* $\sum_{i=0}^{k-1} S_i$, *and thus optimal.*

Proof It is easy to see that if the qth bit of pth dimension for an index in I^m is permuted to the jth bit of ith dimension in P^k, then a step of the FFT communication $C^R(p, q)$ is translated into the single dimensional communication sd $(i, 2^j)$ on the mesh. The second part follows from the fact that the each bit of each dimension will be mapped to exactly once, the total cost is thus

$$\sum_{i=0}^{k-1} \sum_{j=0}^{\log(S_j)} 2^j = \sum_{i=0}^{k-1} (S_j - 1)$$

in other words, the diameter of the mesh.

8 Conclusion

We have studied the mapping from m dimensional FFT communication to k dimensional for arbitrary m and k under a unified framework. We established the lower bound of the communication, and give $(\log(n))!$ bit-permutation mappings, all are optimal. The results in this paper thus generalizes the previous work in [1, 3, 2, 11, 13, 7, 8]. Our broadcast algorithms in Sec. 3 also generalized generalizes the well-known results on broadcasting in [4].

There are a number of related issues and results that can be further studied

- the applications of the mapping to algorithms other than FFT but use FFT communication patterns.
- array operations over specific dimension using FFT communication patterns, e.g., row-wise broadcast for $m = 2$.

Limited by the space, we will discuss the above issues elsewhere. It should be pointed out that the m-k shuffle subsumes that perfect shuffle (m=2, k=1), and the permutation used in [7, 8] ($m = 1, k$ arbitrary). Although the m-k shuffle mapping is cost-equivalent to the other mappings in the bit-permutation mapping family it can be shown that for operations over some but not all dimensions of an array, it outperforms other mappings such as the identical mapping, which is unfortunately widely used in implementations.

References

1. Peter M. Flanders. A unified approach to a class of data movements on an array processor. *IEEE Transactions on Computers*, C-31(9):809–819, September 1982.
2. Peter M. Flanders and Dennis Parkinson. Data mapping and routing for highly parallel processor arrays. *Future Computing Systems*, 2(2):184–224, 1987.
3. Donald Fraser. Array permutation by index-digit permutation. *Journal of ACM*, 23(2):298–309, April 1976.
4. S. Lennart Johnsson and Ching-Tien Ho. Spanning graphs for optimum broadcasting and personalized communication in hypercubes. *IEEE Trans. Computers*, 38(9):1249–1268, September 1989.
5. R. A. Kamin and G. B. Adams. Fast fourier transform algorithm design and tradeoffs on the cm-2. *International Journal of High Speed Computing*, 1(2):207–231, 1989.
6. Z. G. Mou. Divacon: A parallel language for scientific computing based on divide-and-conquer. In *Proceedings of the Third Symposium on the Frontiers of Massively Parallel Computation*, pages 451–461. IEEE, October 1990.
7. Z. G. Mou, C. Constantinescu, and T. Hickey. Divide-and-conquer on a 3-dimensional mesh. In *Proceedings of the European Workshops on Parallel Computing*, pages 344–355, Barcelona, Spain, March 1992.
8. Z. G. Mou, Cornel Costantinescu, and T. Hickey. Optimal mappings of divide-and-conquer algorithms to mesh connected parallel architectures. In *Proceedings of International Computer Symposium*, pages 273–284, Taiwan, December 1992.
9. Z. G. Mou and P. Hudak. An algebraic model for divide-and-conquer algorithms and its parallelism. *The Journal of Supercomputing*, 2(3):257–278, November 1988.
10. F. P. Preparata and J. Vuillemin. The cube-connected cycles: A versatile network for parallel computation. *Communications of the ACM*, 8(5):300–309, May 1981.
11. S.L.Johnsson, C-T Ho, M. Jacquemin, and A. Ruttenberg. Computing fast fourier transforms on boolean cubes and related networks. *SPIE Advanced Algorithms and Architectures for Signal Processing*, 826(II):223–230, 1987.
12. H. S. Stone. Parallel processing with the perfect shuffle. *IEEE Transactions on Computers*, C-20(2):153–160, February 1971.
13. C. Tong and P. N. Swarztrauber. Ordered fast fourier transforms on a massively parallel hypercube multiprocessor. *Journal of Parallel and Distributed Computing*, (12):50–59, 1991.
14. J. D. Ullman. *Computational Aspect of VLSI*. Computer Science Press, 1984.

Digitized by Google Libraries