FAULT–TOLERANT ALGORITHM FOR FAST FOURIER TRANSFORM ON HYPERCUBES

Yu–Wei Chen

Department of Computer and
Information Science
Aletheia University

No. 32, Chen–Li Street, Tamsui,
Taipei, 25103 Taiwan, R. O. C.
Email: ywchen@email.au.edu.tw

Kuo–Liang Chung

Department of Information Management and
Institute of Information Engineering
National Taiwan University of
Science and Technology
No. 43, Section 4, Keelung Road,
Taipei, Taiwan 10672, R. O. C.
Email: klchung@cs.ntust.edu.tw

ABSTRACT

This paper presents an efficient fault–tolerant algorithm for Fast Fourier Transform (FFT) with 2^n data on an n–dimensional hypercube with $n-1$ faulty nodes in $9n-15$ communication steps and $O(n)$ computation steps. To the best of our knowledge, this is the first time that such a fault–tolerant algorithm for FFT on hypercubes is proposed in the literature.

Keywords: Fast Fourier Transform (FFT), fault tolerance, faulty hypercube, free dimensions, parallel algorithm.

1. INTRODUCTION

Fast Fourier Transform (FFT) is one of the most useful algorithms in the community of computer science and electrical engineering [**?**]. Hypercube is one of the most versatile and popular networks due to its low degree of diameter, good connectivity, and symmetry [**?**, **?**]. Therefore, how to design an efficient FFT algorithm on hypercubes is a significant research topic. Without providing fault–tolerant capability, some efficient FFT algorithms [**?**, **?**, **?**, **?**] have been designed on hypercubes. Considering faulty hypercubes, it is an important issue to design an efficient fault–tolerant FFT algorithm and it leads to this research.

Consider an input sequence of 2^n data and an n–dimensional hypercube, H_n, with $n-1$ faulty nodes. Employing the free–dimension concept [**?**], this paper presents an efficient fault–tolerant algorithm for FFT with these 2^n data on the faulty H_n in $9n-15$ communication steps and $O(n)$ computation steps. To the best of our knowledge, this is the first time that such a fault–tolerant algorithm for FFT on hypercubes is proposed in the literature.

The remainder of this paper is organized as follows. The next section describes some preliminaries. Section 3 presents our fault–tolerant algorithm for FFT. Finally, some conclusions are addressed in Section 4.

2. PRELIMINARIES

This section consists of three subsections. The first subsection introduces some notations of hypercubes and the fault model used. The second subsection describes the FFT algorithm on the fault–free H_n. The third subsection reviews the free–dimension (FD) concept [**?**] in hypercubes.

2.1 Notations and Fault Model Used

An n–dimensional H_n has 2^n nodes and $n2^{n-1}$ edges. Each node in H_n is labeled by $b_n b_{n-1} \cdots b_2 b_1$, $b_j \in \{0,1\}$ for $1 \leq j \leq n$, where j denotes the corresponding dimension. Two nodes are connected via an edge if and only if their binary strings differ in exactly one bit. For example, node $b_n b_{n-1} \cdots b_{j+1} b_j b_{j-1} \cdots b_2 b_1$ and node $b_n b_{n-1} \cdots b_{j+1} \bar{b_j} b_{j-1} \cdots b_2 b_1$ are adjacent along dimension j, where $\bar{b_j}$ denotes the complement of b_j. Fig. 1 illustrates an H_4.

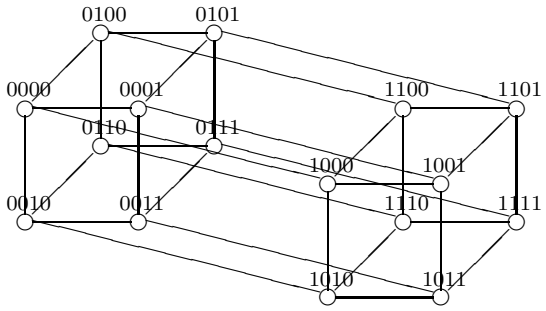

Fig. 1. An H_4.

H_n can be partitioned into 2^{n-k} SH_k's, where each SH_k is a k–dimensional subcube, spanned by the same k dimensions. For example, H_4 can be partitioned into four SH_2's labeled by $00**$, $01**$, $10**$, and $11**$. Two SH_2's are adjacent if their ternary representations differ in exactly one symbol.

The fault model used in this paper follows the total fault model [**?**]. In this fault model, it assumes that the functions for computation and communication in the faulty node are all lost.

2.2 FFT on a Fault–Free H_n

Consider an input sequence $X =< X_0, X_1, \cdots, X_{N-1} >$ of $N = 2^n$ data. The corresponding discrete Fourier transform of the sequence X is the sequence $Y =< Y_0, Y_1, \cdots, Y_{N-1} >$, where

$$Y_i = \sum_{k=0}^{N-1} X_k w^{ki}, \qquad 0 \leq i < N. \tag{1}$$

In Equation (1) [**?**], w is the Nth complex root of unity; that is, $w = e^{2\pi\sqrt{-1}/N}$, where e is the base of the natural logarithm.

The basic concept of FFT algorithm is that the computation for the discrete Fourier transform with N data can be split into two discrete Fourier transforms, each with $N/2$ data. Therefore, Equation (1) can be rewritten as:

$$\begin{aligned} Y_i &= \sum_{k=0}^{N/2-1} X_{2k} w^{2ki} + \sum_{k=0}^{N/2-1} X_{2k+1} w^{(2k+1)i} \\ &= \sum_{k=0}^{N/2-1} X_{2k} e^{2(2\pi\sqrt{-1}/N)ki} + \\ &\qquad \sum_{k=0}^{N/2-1} X_{2k+1} w^i e^{2(2\pi\sqrt{-1}/N)ki} \\ &= \sum_{k=0}^{N/2-1} X_{2k} e^{2\pi\sqrt{-1}ki/(N/2)} + \\ &\qquad w^i \sum_{k=0}^{N/2-1} X_{2k+1} e^{2\pi\sqrt{-1}ki/(N/2)}. \end{aligned} \tag{2}$$

Let $\bar{w} = e^{2\pi\sqrt{-1}/(N/2)} = w^2$ which is the primitive $(N/2)$th complex root of unity. Thus, Equation (2) can be rewritten as:

$$Y_i = \sum_{k=0}^{N/2-1} X_{2k} \bar{w}^{ki} + w^i \sum_{k=0}^{N/2-1} X_{2k+1} \bar{w}^{ki}. \tag{3}$$

The FFT algorithm can be easily implemented on H_n [**?**, **?**]. Fig. 2 illustrates an FFT with 8 data $< X_0, X_1, \cdots, X_7 >$ on a fault–free H_3. Initially, each node $b_3 b_2 b_1$ for $b_3, b_2, b_1 \in \{0,1\}$ in H_3 holds X_i for $0 \leq i = b_1 b_2 b_3 \leq 7$ as shown in the first row in Fig. 2. For example, letting $b_3 b_2 b_1 = 100$, node 4 holds X_1. In Fig. 2, such an FFT with 8 ($=2^3$) data performed on H_3 takes three stages as shown in the second, third, and fourth rows. In each stage, each node does an exchange–compute (EC) operation which is defined as: in Stage_j for $1 \leq j \leq n$, the pair nodes $b_n b_{n-1} \cdots b_{j+1} b_j b_{j-1} \cdots b_2 b_1$ and $b_n b_{n-1} \cdots b_{j+1} \bar{b_j} b_{j-1} \cdots b_2 b_1$ on H_n exchange their intermediate values and compute their new intermediate values, respectively.

For example, in Stage_2 as shown in Fig. 2, nodes 0 and 2 exchange their intermediate values $X_0 + X_4$ and $X_2 + X_6$ and then compute their new intermediate values $X_0 + X_2 + X_4 + X_6$ and $X_0 + w^4X_2 + X_4 + w^4X_6$, respectively. An EC operation takes one communication step and O(1) computation steps. Finally, the desired result Y_i resides in node i ($= b_1b_2b_3$) in H_3. The interesting readers are referred to [**?**, **?**] for detail.

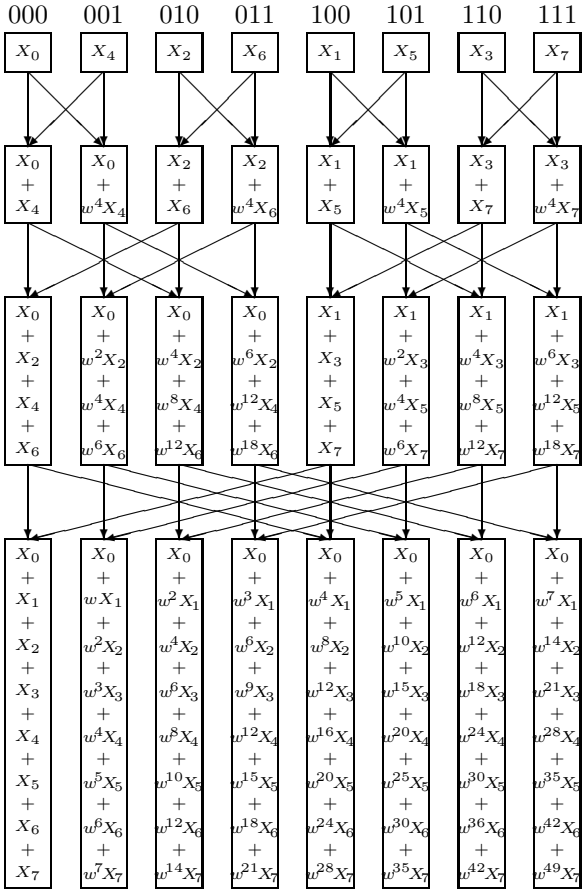

Fig. 2. An FFT with eight data on a fault–free H_3.

In general, an FFT with N ($=2^n$) data performed on H_n takes n stages, i.e., n communication steps and O(n) computation steps [**?**, **?**].

2.3 The Free–Dimension (FD) Concept

Previously, Raghavendra *et al.* [**?**] defined that a dimension d in H_n is called a free dimension (FD) if there is no pair of faulty nodes across a dimension d. For example, suppose the faulty nodes of H_4 in Fig. 1 are $\{0000, 1000, 1001, 1010\}$ such that there exists only one FD, i.e., dimension 3. Based on the FD concept, they presented the following result.

Lemma 1. [**?**] *Given f faulty nodes, $0 \le f \le n$, H_n can be partitioned into 2^{f-1} SH_{n-f+1}'s such that each SH_{n-f+1} is spanned by the same $n - f + 1$ FDs and contains at most one faulty node.*

In addition, they also presented a distributed algorithm in O(n) for finding the FDs in the H_n with f faulty nodes such that each fault–free node can identify these FDs [**?**].

3. FAULT–TOLERANT ALGORITHM FOR FFT

In this section, we present a fault–tolerant algorithm for FFT with an input sequence $X =< X_0, X_1, \cdots, X_{N-1} >$ of $N = 2^n$ data on H_n with $n-1$ faulty nodes in $9n-15$ communication steps and O(n) computation steps.

3.1 Two Preprocessing Steps: Relabeling Process and Data Assignment

Before presenting the proposed algorithm, we first relabel the faulty H_n. Using Lemma 1 and the FD–finding algorithm [**?**], we can find two FDs, say dimensions i and j, on the faulty H_n. Each node $b_nb_{n-1}\cdots b_{i+1}b_ib_{i-1}\cdots b_{j+1}b_jb_{j-1}\cdots b_2b_1$, $b_k \in \{0,1\}$ and $1 \le j < i \le n$, in the faulty H_n is relabeled as $b_nb_{n-1}\cdots b_{i+1}b_{i-1}\cdots b_{j+1}b_{j-1}\cdots b_2b_1b_ib_j$. For clarity, hereafter, we assume that the label $b_nb_{n-1}\cdots b_{i+1}b_{i-1}\cdots b_{j+1}b_{j-1}\cdots b_2b_1b_ib_j$ of each node is reindexed as $b_nb_{n-1}\cdots b_2b_1$ such that the dimensions 1 and 2 are two FDs.

Because dimensions 1 and 2 are two FDs, by Lemma 1, we know that each SH_2 labeled by $b_nb_{n-1}\cdots b_4b_3{*}{*}$ contains at most one faulty node. Therefore, if node $b_nb_{n-1}\cdots b_2b_1$ in H_n is faulty, then fault–free node $b_nb_{n-1}\cdots b_2\bar{b_1}$ takes over the

task of the faulty node $b_n b_{n-1} \cdots b_2 b_1$.

Consider an input sequence $X =< X_0, X_1, \cdots, X_{N-1} >$ of $N = 2^n$ data. Each X_i for $0 \leq i = b_1 b_2 \cdots b_{n-1} b_n \leq 2^n - 1$, $b_k \in \{0, 1\}$ and $1 \leq k \leq n$, is assigned to node $b_n b_{n-1} \cdots b_2 b_1$ in H_n if such a node is fault–free. If node $b_n b_{n-1} \cdots b_2 b_1$ is faulty, then X_i is assigned to node $b_n b_{n-1} \cdots b_2 \bar{b_1}$.

3.2 The Main Body of the Proposed Algorithm

The basic concept of the proposed fault–tolerant algorithm for FFT is described below. There are n stages in the proposed algorithm. In Stage_j for $1 \leq j \leq n$, if both nodes $b_n b_{n-1} \cdots b_{j+1} b_j b_{j-1} \cdots b_2 b_1$ and $b_n b_{n-1} \cdots b_{j+1} \bar{b_j} b_{j-1} \cdots b_2 b_1$ are all fault–free, then they do the EC operations defined in Subsection 2.2. After performing the EC operations, the fault–free nodes without obtaining the new intermediate values must perform the update operations to obtain the corresponding intermediate value for FFT at this stage. The detailed update operation will be defined later. Continuing this way, after performing n stages, the desired $Y =< Y_0, Y_1, \cdots, Y_{N-1} >$ will be obtained.

In Stage_1, if both nodes $b_3 b_2 b_1$ and $b_3 b_2 \bar{b_1}$ are all fault–free, then they do the EC operations. If node $b_3 b_2 b_1$ is fault–free and node $b_3 b_2 \bar{b_1}$ is faulty, then node $b_3 b_2 b_1$, which keeps two data sequentially, computes the two corresponding intermediate values for FFT. Specifically, it is impossible that both nodes $b_3 b_2 b_1$ and $b_3 b_2 \bar{b_1}$ are all faulty because dimension 2 and 1 are two FDs such that each SH_2 labeled by $b_3 * *$ contains at most one faulty node. That is, each pair nodes contain at most one faulty node. Consequently, Stage_1 takes one communication step and O(1) computation steps.

In Stage_2, if both nodes $b_3 b_2 b_1$ and $b_3 \bar{b_2} b_1$ are all fault–free, then they first do the EC operations. Because node 011 (110) can not do the EC operation with the faulty node 001 (100) whose task is taken by node 000 (101), node 011 (110) exchanges its own intermediate value $X_2 + w^4 X_6$ $(X_3 + X_7)$ with the intermediate value $X_0 + w^4 X_4$ $(X_1 + X_5)$ of node 000 (101). In addition, both nodes 011 and 000 (110 and 101) compute their new intermediate values. The above process is called an update operation. It is easy to know that the update operation at Stage_2 is performed in two communication steps and O(1) computation steps. Consequently, Stage_2 takes 3 (=1+2) communication steps and O(1) computation steps.

In Stage_3, if both nodes $b_3 b_2 b_1$ and $\bar{b_3} b_2 b_1$ are fault–free, then they first do the EC operations. Because each of both SH_2's labeled by $0 * *$ and $1 * *$ contains only one faulty node, at most two fault–free nodes in each SH_2 are needed to perform the update operations. Both nodes 000 and 101, each of which holds two intermediate values, do the update operations *twice* such that node 000 obtains Y_0 and Y_1 and node 101 obtains Y_4 and Y_5. For analyzing the communication steps of the update operation in this stage, we need the result [**?**]: the diameter of H_3 with two faulty nodes is 4 (=3+1). Consequently, Stage_3 takes 9 (=1+2*4) communication steps and O(1) computation steps. As a result, an FFT of eight data $X_0, X_1, \cdots, X_7$ can be performed on a faulty H_3 with two faulty nodes in 13 (=1+3+9) communication steps and O(1) computation steps. Although the above small example has eight data and eight processors, it is applicable to extend it to the general case with 2^n data.

In general, consider an FFT with 2^n data on H_n with $n - 1$ faulty nodes. In Stage_j for $4 \leq j \leq n$, if both nodes $b_n b_{n-1} \cdots b_{j+1} b_j b_{j-1} \cdots b_2 b_1$ and $b_n b_{n-1} \cdots b_{j+1} \bar{b_j} b_{j-1} \cdots b_2 b_1$ are all fault–free, then both nodes do the EC operations. If any one of the two nodes is faulty, then the update operations will be performed. Both SH_2's labeled by $b_n b_{n-1} \cdots b_{j+1} b_j b_{j-1} \cdots b_4 b_3 * *$ and $b_n b_{n-1} \cdots b_{j+1} \bar{b_j} b_{j-1} \cdots b_4 b_3 * *$ can be treated as the two SH_2's labeled by $0 * *$ and $1 * *$ at Stage_3. Thus, the time required in Stage_j for $4 \leq j \leq n$ is the same as that in Stage_3.

As a result, Stage_1 takes 1 communication step;

Stage_2 takes 2 communication steps; by the result [**?**], the number of communication steps required from Stage_3 to Stage_n is $9(n-2)$. Consequently, the proposed algorithm takes $9n-15$ $(=1+2+9(n-2))$ communication steps and $O(n)$ computation steps. Based on the above analysis, we have the following main result.

Theorem 2. *The proposed fault–tolerant algorithm for FFT with 2^n data can be performed on H_n with $n-1$ faulty nodes in $9n-15$ communication steps and O(n) computation steps.*

4. CONCLUSIONS

The main contribution of this paper is presenting a fault–tolerant algorithm for FFT with 2^n data on an H_n with $n-1$ faulty nodes in $9n-15$ communication steps and $O(n)$ computation steps. In addition, using the same communication scheme of the proposed fault–tolerant algorithm for FFT, all algorithms designed on the n–dimensional butterfly network [**?**] can be directly performed on an H_n with $n-1$ faulty nodes under the same complexity.

References

[1] E. O. Brigham, The Fast Fourier Transform, Prentice–Hall, Englewood Cliffs, NJ., 1974.

[2] R. M. Chamberlain, Gray codes, Fast Fourier Transforms and hypercubes, *Parallel Computing*, **6** (1988) 225–233.

[3] J. Håstad, T. Leighton, and M. Newman, Reconfiguring a hypercube in the presence of faults, *ACM Symp. on Theory of Comput.*, (1987) 274–284.

[4] S. L. Johnsson, M. Jacquemin, and C. T. Ho, High radix FFT on boolean cube networks, Technical Report NA89–7, Thinking Machines Corporation, 1989.

[5] M. S. Krishnamoorthy and B. Krishnamurthy, Fault diameter of interconnection networks, *Comput. Math. Appl.*, **13** (1987) 577–582.

[6] F. T. Leighton, Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes, Morgan Kaufmann Pub., CA., Chap. 3, 1992.

[7] C. S. Raghavendra, P. J. Yang, and S. B. Tien, Free dimensions—an effective approach to achieving fault tolerance in hypercubes, *IEEE Trans. on Computers*, **44**, 9 (1995) 1152–1157.

[8] Y. Saad, and M. H. Schultz, Topological properties of hypercube, *IEEE Trans. on Computers*, **37** (1988) 867–872.

[9] P. N., Swartztrauber, Multiprocessor FFT's, *Parallel Computing*, **5** (1987) 197–210.