Real-time fault-tolerant hypercube multicomputer

B.A. Izadi and F. Özgüner

Abstract: A real-time fault-tolerant design for a d-dimensional hypercube multiprocessor with two modes of operation is presented and its reconfigurability is examined. The augmented hypercube, at stage one, has a spare node connected to each node of a subcube of dimension i, and the spare nodes are also connected as a $(d - i)$-dimensional hypercube. At stage two, the process is repeated by assigning one spare node to each $(d - i - j)$-dimensional spare subcube of stage one. Two modes of operations are considered, one under heavy computation or hard deadline and the other under light computation or soft deadline. By utilising the capabilities of wave-switching communication modules of the spare nodes, faulty nodes and faulty links can be tolerated. Both theoretical and experimental results are presented. Compared with other proposed schemes, the proposed approach can tolerate significantly more faulty components with a low overhead and no performance degradation.

1 Introduction

In the quest to attain petascale computing, researchers are designing parallel machines with hundreds of thousands of processing elements [1]. The hypercube, due to its rich interconnection properties [2], is an attractive topology to implement such parallel machines. To keep the hypercube multicomputer operational and sustain the same level of performance in the presence of faults, researchers have proposed schemes that augment the hypercube with spare nodes and/or spare links to replace the faulty ones [3–13]. Petascale multicomputers have been cited as having the capacity to solve a number of complex applications, including some with real-time implication [1]. Hence, in a real-time environment faulty components have to be replaced with spares in a manner that also satisfies the required completion deadline of active tasks. Two modes of operation are generally considered: the 'strict mode' and the 'relaxed mode'. The strict mode pertains to tasks whose computational requirements are heavy or have a hard completion deadline. The relaxed mode, on the other hand, consists of tasks with a soft completion deadline or a light computational load. Therefore, in the strict mode of operation, in order to allow for fast reconfiguration, spare replacement of faulty components should result in very few changes in the system interconnections. A common approach to accommodate this mode of operation is to replace each faulty component with the local spare using a distributed reconfiguration algorithm [14]. On the other hand, in the relaxed mode of operation, a global reconfiguration algorithm is applied to maximise the probability that in the next strict mode of operation there exists a local spare for every faulty component.

We present a two-stage redundant scheme for the hypercube. The objectives of the scheme are two fold. First, to facilitate real-time fault tolerance by allowing the system to operate in either the strict mode or the relaxed mode. Second, to utilise the spare network to tolerate a large number of faulty nodes and faulty links.

2 Definitions and notation

Each regular node of a d-dimensional hypercube is denoted by 'd-tuple' $(b_{d-1}, \ldots, b_i, \ldots, b_0)$, where $b_i \in \{0, 1\}$. A subcube is defined by a unique d-tuple $\{0, 1, X\}^d$ where '0' and '1' are the 'bound' coordinates, and 'X' represents the free co-ordinates. A $(d - k)$-dimensional subcube is represented by a d-tuple with k bound co-ordinates and $(d - k)$ free co-ordinates. Each spare node is uniquely defined by d-tuple $\{0, 1, S\}^d$ where '0' and '1' represent the bound co-ordinates and S corresponds to the free co-ordinates of the spare node's assigned cluster. A regular link is specified uniquely by d-tuple $\{0, 1, -\}^d$ where '-' can be substituted by '0' or '1' to identify its connecting nodes. An intra-cluster spare link (a link connecting the spare node of stage one to a regular node of its cluster, or a link connecting the spare node of stage two to a spare node of its cluster at stage one) is defined by a $(d+1)$-tuple $\{0, 1, S\}^{(d+1)}$ where S is inserted to the left of the $(i - 1)$th bit of the regular node ID, to which the spare node is connected. For example, the intra-cluster spare link connecting the spare node 00SS and node 0001 is identified as 00S01. An inter-cluster spare link (a link connecting two spare nodes at either stage one or stage two) is defined by a d-tuple $\{0, 1, S, -\}^d$ where '-' can be substituted by '0' or '1' to identify its connecting spare nodes. Hence, the inter-cluster spare link connecting spare nodes 01SS and 00SS is labelled 0-SS. Finally, we define the 'connection requirement' (C_R) of a spare node in a cluster with multiple faulty nodes as the number of edge-disjoint paths that must be constructed within the spare cube from that spare node to other spare nodes in the fault-free clusters so that faulty nodes can be tolerated.

IEE Proceedings online no. 20020720

DOI: 10.1049/ip-cdt:20020720

Paper first received 8th August 2001 and in revised form 19th August 2002

B.A. Izadi is with the Department of Electrical and Computer Engineering, State University of New York, 75 South Manheim Blvd., New Paltz, NY 12561, USA

F. Özgüner is with the Department of Electrical Engineering, Ohio State University, 2015 Neil Ave., Columbus, OH 43210-1277, USA

3 Overview of the TECH

In our scheme, at stage one, the d-dimensional hypercube is divided into $2^{(d-i)}$ subcubes of dimension i; we call each of these subcubes a cluster. One spare node is assigned to each cluster; the spare node is connected to every regular node of its cluster via an intra-cluster spare link. Spare nodes are also interconnected to form a $(d-i)$-dimensional spare cube using inter-cluster spare links. We call the resultant structure the 'enhanced cluster hypercube' (*ECH*) [3]. Fig. 1 depicts an ECH of dimension $d=4$ with clusters of dimension $i=2$. At stage two, the process is repeated: the spare nodes of stage one are also divided into $2^{(d-i-j)}$ clusters of dimension j and one spare node from stage two is assigned to each of these clusters. Moreover, the spare nodes at stage two are interconnected as a $(d-i-j)$-dimensional spare cube. We call the resultant structure the 'two-stage enhanced cluster hypercube' (*TECH*). Fig. 2 depicts a TECH of dimension $d=4$ with $i=2$ and $j=1$. Each regular node in a TECH is connected to its d neighbouring regular nodes and the local spare node at stage one. Each spare node at stage one is connected to 2^i regular nodes of its local cluster, its $(d-i)$ neighbouring spare nodes at stage one, and its assigned spare node at stage two. Each spare node at stage two is connected to 2^j spare nodes of its local cluster of stage one and its $(d-i-j)$ neighbouring spare nodes at stage two. Therefore, the degree of each regular node, each spare node at stage one, and each spare node at stage two are $(d+1)$, $(2^i+d-i+1)$, and $(2^{i-j}+d-i-j)$, respectively.

We next describe how the TECH tolerates faulty nodes and faulty links. Each node is made of a computation module and a wave-switching communication module [15]. Waveswitching implements circuit-switching and wormhole-switching concurrently; permanent connections and long messages use the circuit-switched segment while short messages are transmitted using the wormhole-switching. We assume that faulty nodes retain their ability to communicate. This is a common assumption since the hardware complexity of the communication module is much lower than that of the computational module. Therefore, the probability of failure in the communication module is much lower than in the computation module.

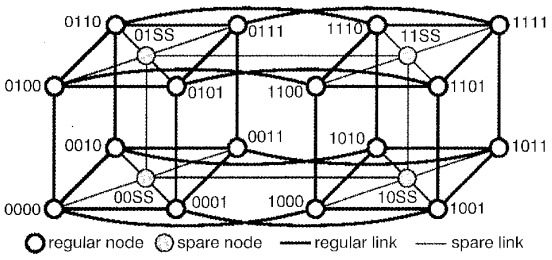

Fig. 1 *ECH of dimension 4*

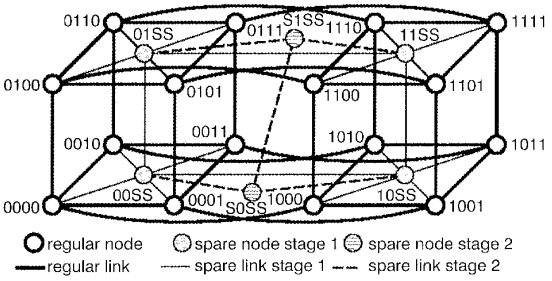

Fig. 2 *TECH of dimension 4*

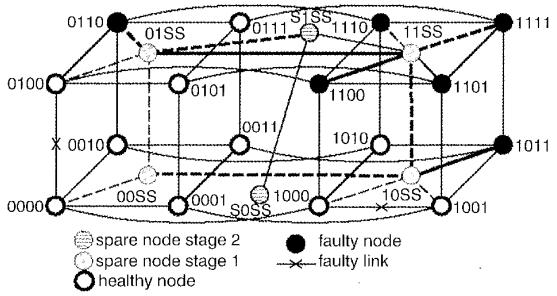

Fig. 3 *Reconfiguration of TECH*

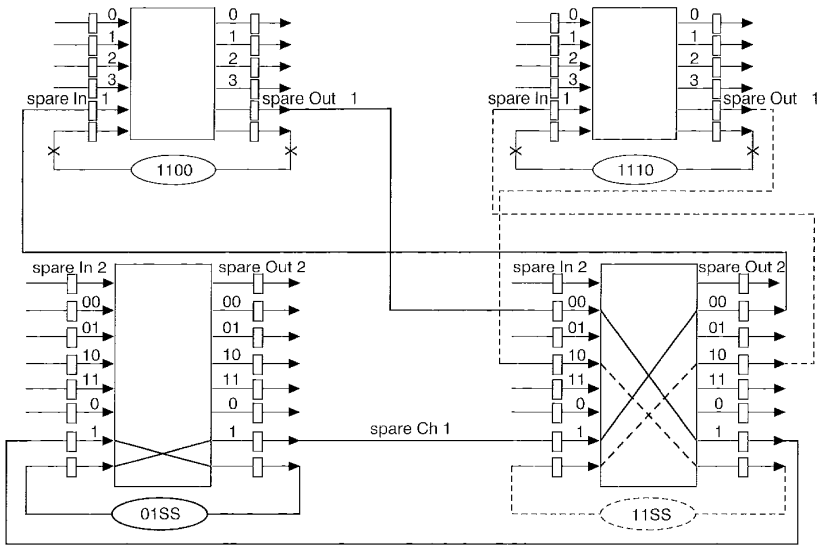

Fig. 4 *Replacing faulty nodes 1100 and 1110 with spare nodes 01SS and 11SS, respectively*

This assumption may be avoided by duplicating the communication module in each node. To tolerate a faulty node, the computation module of the spare node logically replaces the computation module of the faulty node. In addition, if the spare node resides in the cluster of the faulty node, the new communication module consists of the functional communication module of the faulty node merged with the appropriate routing channel of the local spare node. If the assigned spare node and the faulty node belong to different clusters, a dedicated path is constructed by linking the appropriate routing channels of the intermediate spare nodes. Once such a path is established, due to the circuit-switched capability of the wave-switching communication modules, the physical location of the faulty node and its assigned spare node becomes irrelevant. Moreover, no modification of the available computation or communication algorithm is necessary. Similarly, faulty links are bypassed by establishing parallel paths using spare links. Fig. 3 illustrates the reconfiguration of a TECH with $d=4$, $i=2$, and $j=1$ in the presence of indicated faulty nodes and faulty links. For the sake of clarity, in Fig. 3, non-active spare links are deleted and active spare links are drawn in a variety of line styles to distinguish the bypass paths. Note that by utilising spare nodes from other fault-free clusters, in effect, four logical spare nodes are present in cluster 11*XX*. Fig. 4 illustrates how spare nodes 01*SS* and 11*SS* replace faulty nodes 1100 and 1110, respectively, by merging their communication module. The heavy and dashed lines in Fig. 4 pertain to similar lines in Fig. 3 and represent effective permanent circuit-switched connections after the reconfiguration.

4 Reconfigurability of the TECH

To allow for fast reconfiguration in the strict mode of operation, the reconfiguration algorithm should result in minimum changes in system interconnections. Hence, under the strict mode of operation, each faulty node of a cluster is replaced by the local spare node at stage one. The algorithm is applied distributively, allowing each spare node at stage one to monitor the status of the regular nodes within its cluster, and replace the faulty node as outlined in the previous Section. The reconfiguration algorithm under the strict mode of operation fails if more than one node becomes faulty in a cluster.

The reconfiguration algorithm in the relaxed mode of operation first tries to assign each detected faulty node to a spare node at stage two. This is done to make the spare nodes at stage one available for the next strict mode of operation. For example, in Fig. 2, in the relaxed mode of operation, the task of faulty nodes 0110 and 0000 are assigned to spare nodes *S*1*SS* and *S*0*SS*, while in the strict mode of operation they would be assigned to the local spare nodes 01*SS* and 00*SS*, respectively. Under the relaxed mode of operation, if there is no available unassigned spare node at stage two, a spare node from stage one is assigned to the faulty node; details of the reconfiguration algorithm under the relaxed mode of operation is discussed in Section 4.2.

4.1 Theoretical results

From the previous Section it follows that the reconfigurability of the TECH is a function of the number of dedicated and edge-disjoint paths, within the spare network at stages one and two, that can be established between the local spare node of a cluster with multiple faulty nodes and the available spare nodes in the fault-free clusters. The availability of these edge-disjoint paths is a connectivity issue within the spare network. The following theorems examine the connectivity of the spare network and establish bounds on the number of faulty nodes that a TECH can tolerate.

Lemma 1: A d-dimensional TECH with clusters of dimension i, at stage one, can at most tolerate $(d-i+2)$ faulty nodes in one cluster.

Proof: Given a cluster, at stage one, with multiple faulty nodes, the local spare node can replace one of them. Since the local spare node, at stage one, has a degree of $(d-i+1)$ within the spare network, at most $(d-i+1)$ edge-disjoint paths may be constructed from it to unassigned spare nodes. □

Theorem 1: A d-dimensional TECH with clusters of dimension i, at stage one, can tolerate $(d-i+2)$ faulty nodes regardless of the fault distribution.

Proof: We first show, by induction, that a TECH at stage one can tolerate $(d-i+1)$ faulty nodes regardless of the fault distribution. The TECH at stage one is simply an ECH of dimension d with clusters of dimension i. The base case is shown for $d=i$. There exists one spare node which is connected to each of the 2^d regular nodes. Upon failure of any one regular node, the spare node can replace it directly. Therefore, the ECH can tolerate $d-i+1=1$ faulty node.

Next, let us consider an ECH of dimension $d=i+(k-1)$: each cluster has a dimension $i=d-(k-1)$ and the dimension of the spare cube is $d-i=(k-1)$. Let us assume that the $(i+k-1)$-dimensional ECH can tolerate $(d-i)+1=(k-1)+1=k$ faulty nodes. Keeping the cluster size constant, consider an ECH of dimension $d=i+k$. The dimension of the spare cube is then $d-i=k$. Let us split the $(i+k)$-dimensional ECH along a dimension l such that $l>i$. Consequently, the regular nodes of the two $(i+k-1)$-dimensional subcubes are connected along the dimension l; they form a match along the dimension l. Likewise, the spare nodes of the two $(k-1)$-dimensional spare cubes are connected along the dimension $(l-i)$ of the spare cube. By the induction hypothesis, each $(i+k-1)$-dimensional ECH can tolerate k faulty nodes. Suppose there exist $(k+1)$ faulty nodes. If the distribution of faulty nodes is such that at least one of them is in one $(i+k-1)$-dimensional ECH while the rest reside in the other half, the system can tolerate the faulty nodes by the induction hypothesis. Consider the case where all faulty nodes reside in the same $(i+k-1)$-dimensional ECH. k faulty nodes can be tolerated in the same subcube. Since every faulty node has an unused spare link to its local spare node and the local spare node has an unused spare link in the dimension $(l-i)$ to an unassigned spare node within the fault-free half of the ECH, a dedicated path within the spare cube between the $(k+1)$th faulty node and the unassigned spare node can be established. The system can therefore tolerate $(k+1)$ faulty nodes, and it follows by induction that $(d-i+1)$ faulty nodes can be tolerated within the spare cube at stage one.

Finally, consider the TECH at stages one and two. Since the $(d-i+2)$th faulty node has an available edge connecting it to the local spare node, at stage one, and the local spare node has an unused spare link connecting it to a spare node at stage 2, a dedicated path between the $(d-i+2)$th faulty node and the spare node at stage 2 can be established. The system can therefore tolerate $(d-i+2)$ faulty nodes regardless of the distribution of faulty nodes. □

We next set the bounds on the number of faulty nodes per cluster, under the maximum number of faulty nodes ($2^{(d-i)}+2^{(d-i-j)}$), that the TECH can tolerate. Our proof uses the following theorem from [3].

Theorem 2: A d-dimensional ECH with clusters of dimension i can tolerate $2^{(d-i)}$ faulty nodes with up to three faulty nodes per cluster regardless of the fault distribution. □

Theorem 3: A d-dimensional TECH with clusters of dimension i, at the first stage, and clusters of dimension j, at the second stage, can tolerate $2^{(d-i)}+2^{(d-i-j)}$ faulty nodes, regardless of the fault distribution, provided that the maximum number of faulty nodes in each subcube of dimensions i and $i+j$ is 4 and $3(2^j+1)$, respectively.

Proof: Consider the regular hypercube and the spares of the TECH at stage one only. The topology is that of an ECH of dimension d with clusters of dimension i; let us refer to it as the ECH-1. Likewise, the ECH-2 of dimension d with clusters of dimension $i+j$ can be formed by utilising the regular hypercube and the spares of the TECH at stage two. Under the structure of the ECH-2, spare nodes of the TECH at stage one simply function as intermediate hops, and we can ignore the inter-cluster spare links of stage one. By theorem 2, the ECH-2 can tolerate $2^{(d-i-j)}$ faulty nodes with up to three faulty nodes per subcube (cluster) of dimension $i+j$. Likewise, the ECH-1 can tolerate $2^{(d-i)}$ faulty nodes with up to three faulty nodes per subcube of dimension i. Hence, the total number of faulty nodes that the TECH can tolerate is $2^{(d-i-j)}+2^{(d-i)}$.

Next, consider a subcube of dimension $i+j$. Within this subcube, there exists 2^j subcubes of dimension i. Hence, the maximum number of tolerated faulty nodes within a subcube of dimension $i+j$ is 3×2^j (due to the ECH-1) plus 3 (due to the ECH-2) for a total of $3(2^j+1)$. Within this constraint, each subcube of dimension i can tolerate three and one faulty nodes due to the ECH-1 and the ECH-2, respectively, for a maximum total of four. □

From theorem 3 it follows that the reconfiguration of the TECH, under the maximum number of faulty nodes, is guaranteed provided that the number of edge-disjoint paths that must be initiated from each spare node at stage one and stage two be limited to three and two, respectively; the maximum C_R of each spare node at stage one be three and at stage two be two. The result of theorem 3 could be extended to a hypercube with multi-stage redundancies.

Theorem 4: A d-dimensional enhanced cluster hypercube with k-stage redundancies can tolerate $\sum_{j=1}^{k} 2^{(d-\sum_{i=1}^{j} i_i)}$ faulty nodes provided that the maximum C_R of every spare node at stage k is two and every other stage is three, respectively. □

4.2 Simulation results

From our theoretical results, it follows that, in the relaxed mode of operation, some patterns of five faulty nodes per cluster can cause the reconfiguration of the TECH to fail. However, the probability that the faulty nodes can form such patterns is very low. Therefore, a more realistic measure of the reconfigurability of the TECH would be under random fault distributions. We next examine the simulation results based on the following reconfiguration algorithm. An optimal reconfiguration algorithm can be developed by utilising the maxflow algorithm [16]. The main drawback to a reconfiguration using the above algorithm is that a digraph representation of the spare network has to be constructed [17] and the spare node assignment has to be done by the host processor. To overcome these deficiencies, we next present a near-optimal reconfiguration algorithm. The algorithm consists of four parts as specified below:

(i) *Early abort:* The following solvability checks are performed to determine whether the reconfiguration is feasible. If the total number of faulty nodes is greater than the number of spare nodes ($2^{(d-i)}+2^{(d-i-j)}$), the reconfiguration fails. If the C_R of a spare node at stage one is greater than $(d-i+1)$, the reconfiguration fails due to lemma 1.
(ii) *Assignment at stage two:* We utilise Lee's path-finding algorithm [18] to find a set of candidate spare nodes at stage two that can be assigned to the faulty node. The algorithm begins by constructing a breadth-first search of minimum depth k ($1\le k\le 2^{(d-i-j)}-1$) in the spare cube of stage two from the local spare node of a faulty cluster. If a free spare node is found, a path is formed to the faulty node. The algorithm guarantees that a path to a spare node will be found if one exists and the path will be the shortest possible [18]. Once a path is formed, the links associated with that path are deleted from the spare tree, resulting in a new structure. If there still remain some uncovered faulty nodes, a solvability test similar to step 1 is performed on the new structure and this step is repeated for a higher depth k in stage two of the spare cube.
(iii) *Local assignment:* If all spare nodes at stage two are assigned and there still remain some faulty nodes, the local spare node of every faulty cluster is assigned to a faulty node within the cluster.
(iv) *Assignment at stage one:* If there remain additional faulty nodes, we apply Lee's path-finding algorithm to both stages one and two from the local spare node of a faulty cluster with an unassigned faulty node. Reconfiguration fails if $k>2^{(d-i)}+2^{(d-i-j)}$, which is the longest acyclic path in the spare network.

To illustrate our reconfiguration algorithm, let us consider Fig. 5. The algorithm avoids the early abort of step one, since the total number of faulty nodes is not greater than the number of spare nodes and the C_R of no spare node is greater than $(d-i+1)=3$. During the second step, the algorithm randomly selects the faulty node 0110. A breadth-first-search of depth one, at stage two, from the local spare node 01*SS* yields the available spare node *S1SS*. Hence, the spare node *S1SS* is assigned to the faulty node 0110. Similarly, the faulty node 1110 is assigned to the spare node *S0SS* using a breadth-first-search of depth two from the local spare node 11*SS*. Since no other unassigned spare node is available at stage two, the algorithm moves on to step three, where randomly faulty nodes 1011 and 1110 are assigned to their local spare nodes 10*SS*

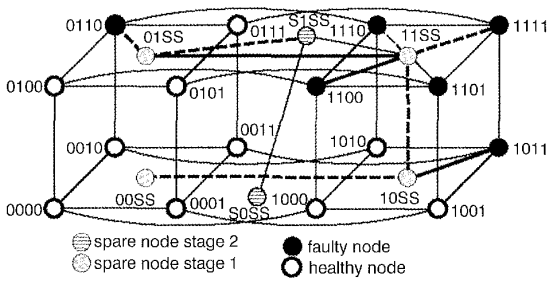

Fig. 5 *Reconfiguration of TECH*

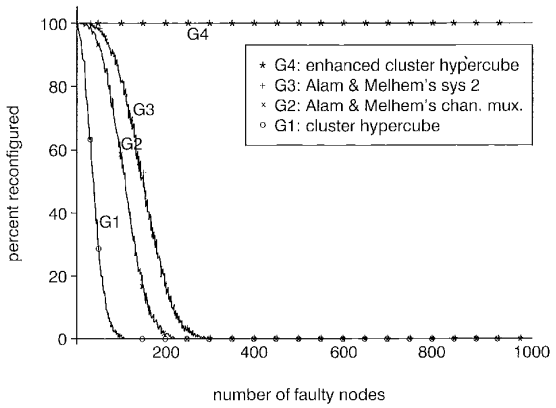

Fig. 6 *Reconfigurability of ECH under random fault distribution*
dimension of hypercube = 20
dimension of each cluster = 10
dimension of spare nodes = 1024

and 11*SS*, respectively. Finally, during step four, faulty nodes 1100 and 1111 are assigned to spare nodes 01*SS* and 00*SS* using a breadth-first-search of depth one and two, respectively, from the local spare node 11*SS*.

We implemented the reconfiguration algorithm for a TECH with $d = 20$, $i = 10$, and $j = 3$. 1000 simulation runs were performed for each given number of faulty nodes. The result of our simulations, under random fault distribution, indicate 100% reconfiguration in the presence of up to 1152 faulty nodes (the maximum). To compare the fault tolerant capability of the TECH with other schemes, we first simulated the reconfigurability of an ECH (a TECH with spare nodes only at stage one) of $d = 20$ and $i = 10$; this is done since most fault-tolerant hypercube schemes in the literature have only one level of redundancy. The simulation result of the ECH for up to 1024 randomly placed faulty nodes is shown in Fig. 6 as plot G4. Plots G1, G2, and G3 in the figure pertain to the schemes proposed by [5, 19], [9], and [8], respectively; compared to other schemes in the literature, the selected ones tolerate more faulty nodes for similar hardware overhead. The result indicates 100% reconfiguration of the ECH in the presence of up to maximum number of faulty nodes.

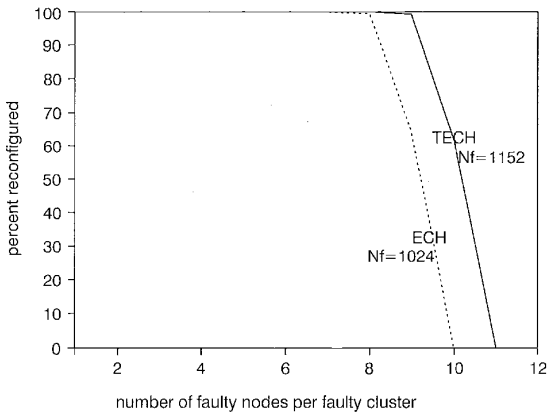

Fig. 7 *Reconfigurability of TECH versus ECH under random fault distribution*
dimension of hypercube = 20
dimension of spare cube at stage 1 = 10
dimension of spare cube at stage 2 = 7
Nf = total number of faulty nodes

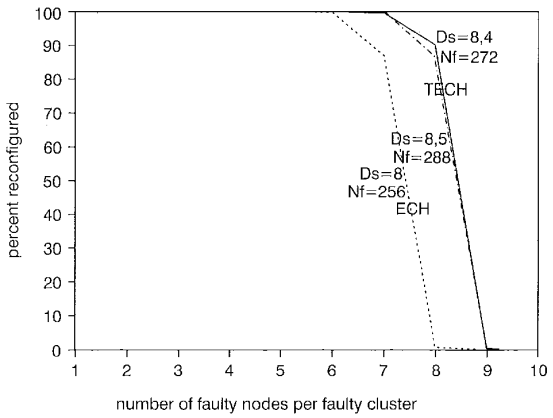

Fig. 8 *Reconfigurability of TECH versus ECH under random fault distribution*
dimension of hypercube = 20
DS = dimension of spare cube
Nf = total number of faulty nodes

We next compared the reconfigurability of the TECH and the ECH with $2^{(d-i)}$ and $2^{(d-i)} + 2^{(d-i-j)}$ faulty nodes, respectively, such that each cluster contains a fixed number of faulty nodes. Fig. 7 depicts the simulation results for the hypercube of dimension ten; the solid and the dashed lines in the Figure pertain to the result of the TECH under 1152 faulty nodes and the ECH under 1024 faulty nodes, respectively. The result indicates that, under the maximum number of faulty nodes, the TECH can handle one more faulty node per cluster than the ECH. The result is intriguing since the degree of the spare node of the TECH at stage one is also higher than the ECH by one. Similar results were attained under different dimension sizes of the spare cubes at stage one and stage two. Fig. 8 depicts the simulation results of an ECH and two TECHs, where the dimensions of the spare cube of the ECH and the spare cube of the TECH at stage one are eight. Moreover, the dimensions of the second-stage spare cube of the two TECHs are five and four. The results illustrate that the existence of a second stage of redundancy is more critical to the higher reconfigurability of the TECH than the size of the spare cube at the second stage. Hence, multi-stage redundancy should only marginally enhance the reconfigurability of the hypercube over the TECH, since the degree of the spare node at stage one is $(d - i + 1)$ regardless of the number of spare stages or their dimensions.

As indicated in the previous Section, the TECH can also tolerate faulty links. However, some faulty links can only have one bypass path. Therefore, no distinction between their relaxed and strict mode of operations can be made. Also, two or more faulty links sharing a node in a TECH will cause the reconfiguration to fail. Hence, no theoretical lower bound on the number of tolerated faulty links can be established. Our simulation results of the TECH, based on random distribution of faulty links, is slightly better than the simulation results of the ECH [3].

5 Conclusion

We have proposed a scheme to allow a hypercube multiprocessor to tolerate faulty nodes in real time. During the strict mode of operation, the scheme uses local reconfiguration, which is the fastest and involves the fewest

switch changes. Then, in the next relaxed mode of operation the tasks of local spare nodes, at stage one, are transferred to the spare nodes at the second stage by applying a global reconfiguration scheme. If a node becomes faulty during the relaxed mode of operation, the scheme tries to assign a spare node at stage two to replace it. This is done to maximise the probability that in the next strict mode of operation local spare nodes may be available to replace potential faulty nodes. Our theoretical results indicate that, in the relaxed mode of operation, our scheme can always tolerate the maximum number of faulty nodes with up to four faulty nodes per cluster at stage one. Our experimental results suggest that, under random fault distribution, the maximum number of faulty nodes can be tolerated with a very high probability.

6 References

1 DONGARRA, J., and WALKER, D.: 'The quest for petascale computing', *IEEE Comput. Sci. Eng.*, 2001, pp. 32–39
2 SAAD, Y., and SCHULTZ, M.H.: 'Topological properties of hypercubes', *IEEE Trans. Comput.*, 1988, **37**, pp. 867–872
3 IZADI, B., ÖZGÜNER, B., and ACAN, A.: 'Highly fault-tolerant hypercube multicomputer', *IEE Proc., Comput. Digit. Tech.*, 1999, **146**, pp. 77–82
4 BANERJEE, P., and PEERCY, M.: 'Design and evaluation of hardware strategies for recon-figuring hypercubes and meshes under faults', *IEEE Trans. Comput.*, 1994, **43**, pp. 841–848
5 CHAU, S.C., and LIESTMAN, A.L.: 'A proposal for a fault-tolerant binary hypercubes architecture'. Proceedings of the IEEE International symposium on fault tolerant computing, 1989, pp. 323–330
6 BANERJEE, P.: 'Strategies for reconfiguring hypercubes under faults'. Proceedings of the IEEE International symposium on fault tolerant computing, 1990, pp. 210–217
7 BANERJEE, P., RAHMEH, J., STUNKEL, C., NAIR, V., ROY, K., BALASUBRAMANIAN, V., and ABRAHAM, J.: 'Algorithm-based fault tolerance on a hypercube multiprocessor', *IEEE Trans. Comput.*, 1990, **39**, pp. 1132–1145
8 ALAM, M., and MELHEM, R.: 'An efficient modular spare allocation scheme and its application to fault tolerant binary hypercubes', *IEEE Trans. Parallel Distrib. Syst.*, 1991, **2**, pp. 117–126
9 ALAM, M., and MELHEM, R.: 'Channel multiplexing in modular fault tolerant multiprocessors'. Proceedings of the IEEE International conference on Parallel processing, 1991, pp. 1492–1496
10 HAYES, J.P.: 'A graph model for fault-tolerant computing systems', *IEEE Trans. Comput.*, 1976, **25**, pp. 875–884
11 BRUCK, J., CYPHER, R., and HO, C.T.: 'Efficient fault-tolerant mesh and hypercube architectures'. Proceedings of the 22nd Annual international symposium on Fault tolerant computing, 1992, pp. 162–169
12 BRUCK, J., CYPHER, R., and HO, C.T.: 'Wildcard dimensions, coding theory and fault-tolerant meshes and hypercubes'. Proceedings of the 23rd Annual international symposium on Fault tolerant computing, 1993, pp. 260–267
13 DUTT, S., and HAYES, J.P.: 'Designing fault-tolerant systems using automorphisms', *J. Parallel Distrib. Comput.*, 1991, **12**, pp. 249–268
14 LIBESKIND-HADAS, R., SHRIVASTAVA, N., MELHEM, R., and LIU, C.: 'Optimal reconfiguration algorithms for real-time fault-tolerant processor arrays', *IEEE Trans. Parallel Distrib. Syst.*, 1995, **6**, pp. 498–510
15 DUATO, J., LOPEZ, P., and YALAMANCHILI, S.: 'Deadlock- and livelock-free routing protocols for wave switching'. Proceedings of the 11th International parallel processing symposium, April 1997, pp. 570–577
16 TUCKER, A.: 'Applied combinatorics', (Wiley, 1984, 2nd edn.)
17 IZADI, B.: 'Design of fault-tolerant distributed memory multiprocessors'. PhD thesis, The Ohio State University, 1995
18 LEE, C.Y.: 'An algorithm for path connection and its applications', *IRE Trans. Electron. Comput.*, 1961, **ec-10**, pp. 346–365
19 IZADI, B., and ÖZGÜNER, F.: 'Spare allocation and reconfiguration in a fault tolerant hypercube with direct connect capability'. Proceedings of the sixth Conference on Distributed memory computing, 1991, pp. 711–714