Applied Mathematics and Computation 220 (2013) 429–438

Contents lists available at SciVerse ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Multiple regular graph embeddings into a hypercube with unbounded expansion

Jen-Chih Lin

Department of Digital Technology Design, National Taipei University of Education, No. 134, Sec. 2, Heping E. Rd., Da-an District, Taipei City 106, Taiwan

ARTICLE INFO

Keywords:
Hypercube
Linear array
Mesh
Ring
Torus

ABSTRACT

As topological structures, interconnection networks play an important role in parallel and distributed computer systems, particularly in multicomputer systems, which provide an effective mechanism for exchanging data between processors. In this paper, we study the node-fault-tolerant capability of an n-dimensional hypercube with respect to multiple regular graph embeddings into a hypercube with unbounded expansion. We present a fault-tolerant method for multiple regular graph embeddings into a hypercube with dilation 3, congestion 1, and load 1. These results show that $O\left(n^2 - \lfloor log_2 l \rfloor^2\right)$ faults can be tolerated where n is the number of dimensions in a hypercube and l is the number of the nodes of the regular graph. The presented embedding methods are mainly optimized for balancing the processor loads, while minimizing load and congestion as far as possible. Furthermore, our method expands on some previously known findings. Also, this technology can be applied in grid computing and cloud computing.

1. Introduction

Parallel and distributed computer systems need the processors, memory, and switches to be able to communicate with each other. The connections between these elements define the interconnection network. The interconnection network provides an effective mechanism for exchanging data between processors in a parallel and distributed computing system. The interconnection networks takes on important roles in parallel and distributed computer systems because they determine the performance of the systems on a large scale. Many such interconnection networks have been proposed in the literature [2,7,12,16,20,29]. The linear array, ring, mesh, and torus are important topologies of interconnection network for parallel and distributed computing. In the design and analysis of an interconnection network, its graph embedding ability is a major concern. An ideal interconnection network (host graph) is expected to possess excellent graph embedding ability, which helps efficiently execute parallel and distributed algorithms with regular task graphs (guest graphs) on this network. Therefore, when evaluating an interconnection network, one major concern is a graph embedding ability. It is well known that an interconnection network can be modeled by a connected graph $G = (V,E)$, where $V = V(G)$ is the vertex-set and $E = E(G)$ is the edge-set of G, in which vertices represent processors and edges represent communication links between processors. Graph embedding [16] allows us to use an algorithm developed for an existing network on a new or different one, by simply letting the new/different network follow the same steps as in the old network. We say that the new network (guest graph) is embedded into the old one (hose graph).

The embedding of a graph G into a host graph H is a one-to-one function with N nodes (vertices) of G into the nodes of H, combined with a mapping E of each edge $e = (v,w)$ of G to a path of H between $N(v)$ and $N(w)$. The path $E(e)$ is called an image path. Efficiency of an embedding (N,E) is generally measured by *load*, *dilation*, *congestion*, and *expansion*. *Load* is the

E-mail address: yachih@tea.ntue.edu.tw

0096-3003/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.amc.2013.06.033

maximum number of nodes of G mapped to any node of H. *Dilation* is the length of the longest image path. *Congestion* is the maximum number of image paths including an edge e' for every edge e' of H. *Expansion* is the ratio of the number of the nodes of H to the number of the nodes of G.

The hypercube [22] has recently provided a cost effective and feasible approach to supercomputing through parallelism at the processor level by directly connecting a large number of low-cost processors with local memories which communicate by passing messages instead of shared variables. The hypercube concept has been the focus of various research studies in parallel and distributed computing because of its well-defined properties with the modularity, regularity, and low diameter, etc. It is important advantages are high data bandwidth and low message latency. Moreover, a hypercube may contain as many other networks as its subgraphs such as linear arrays, rings, meshes, and tori, etc. Therefore, the hypercube is one of the most versatile interconnection networks yet discovered for parallel and distributed computation. It can efficiently simulate many other networks of various sizes. Because nodes and/or links in a network may fail accidentally, the fault tolerance of a network must be considered. Hence, many research studies have addressed the issue of faulty hypercubes [11,13,25,26,30].

The linear arrays and rings are two important types of embedded graphs because many parallel and distributed algorithms have been developed using their structures [3,5,8,13,17]. In particular, the embedding of linear arrays and rings in a faulty interconnection network is of great significance. For example, path embedding in a faulty n-cube was addressed in [13,19,25]. However, one should notice that each component of a network may not be equally reliable. Thus, the probability that all faulty components would be close to one another seems low. With this observation, Harary [10] first introduced the concept of conditional connectivity. Later, Latifi et al. [15] defined the conditional-node-faults, which require each node of a network to at least have fault-free neighbors.

Since the mesh and torus are two of the most fundamental structures for parallel and distributed computation, a variety of efficient algorithms were developed based on these two topologies [6,14,18,27,31]. Actually, meshes and tori are also two popularly used interconnection networks in distributed-memory parallel and distributed computers. For instance, the Alpha 21364-based HP GS 1280 machine [4], the Cray X IE vector computer [24], and 3-dimensional torus were designed in the Cray T3D and T3E [23]; a 2-dimensional torus is the common choice on the interconnection networks of the iWarp [9] and 2-dimensional mesh and torus topologies are usually adopted for networks-on-chips [21]. Due to these advantages, the problem of how to embed a family of disjointed tori (or meshes) into a host graph has received a great deal of consistent attention in academia as well as industries in recent years.

Load balancing, communication locality, communication congestion, and node utility in process graphs can be studied abstractly as problems related to embedding [28]. In a process graph, the nodes represent processes comprising a distributed program or a parallel program and the edges represent communications between processes. The efficiency of a reconfiguration scheme is strongly affected by how tasks are initially mapped to a parallel computer. If a task graph (representing the task) is embedded in a proper way, the reconfiguration scheme can be simple and involve only local movements. Such initial embeddings, called fault-tolerant embedding, however, require more nodes than embeddings with no fault tolerance. Thus, the idea of fault-tolerant embedding is to intentionally leave some spare nodes in the initial embedding such that, when faults occur, the faulty nodes can be quickly replaced by nearby spare nodes. Therefore, this paper relaxes the limit of the expansion. The main design question raised in fault-tolerant embedding is how to distribute the spare nodes and minimize their number so that more faults can be tolerated. In this investigation, we discuss our embedding function such above multiple regular graph embeddings into a hypercube with unbounded expansion, congestion 1, dilation 3, load 1. As a result, we can transmit the parallel and distributed algorithms developed under these regular graph structures to the hypercube.

The rest of this paper is organized as follows. Section 2 introduces basic definitions and notations. Section 3 investigates these multiple regular graph embeddings. Section 4 presents the embedding of multiple regular graphs into a faulty hypercube with unbounded expansion. Finally, Section 5 presents the conclusions.

2. Preliminary discussion

For our purpose, an interconnection network is represented by graph, where nodes and edges represent processors and communication links between processors, respectively.

Definition 2.1. The Hamming distance between two nodes with labels $x = x_{n-1}x_{n-2}\ldots x_0$ and $y = y_{n-1}y_{n-2}\ldots y_0$ is defined as

$$HD(x,y) = \sum_{i=0}^{n-1} hd(x_i, y_i) \quad \text{where} \quad hd(x_i, y_i) = \begin{cases} 0, & \text{if } x_i = y_i, \\ 1, & \text{if } x_i \neq y_i. \end{cases}$$

Definition 2.2. Let two nodes with labels $x = x_{n-1}x_{n-2}\ldots x_0$ and $y = y_{n-1}y_{n-2}\ldots y_0$, then $Dim(x,y) = \{i \text{ in } (0\ldots n-1) | x_i \neq y_i\}$

An n-dimensional hypercube H_n is a graph with 2^n vertices, in which each vertex is denoted by an n-bit binary string $x = x_{n-1}x_{n-2}\ldots x_0$. Two vertices are adjacent if and only if their strings differ in exactly one bit position. Fig. 1 depicts the 4-dimensional hypercube H_4.

The *Binary-Reflected Gray Code* (BRGC) [1] is defined recursively as follows. $C_{n+1} = \{0C_n, 1(C_n)^R\}$, where $C_1 = \{0,1\}$, $(C_1)^R = \{1,0\}$ and $C_2 = \{0C_1, 1(C_1)^R\}$. The following is a method of constructing longer Binary-Reflected Gray codes from shorter