JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 8, 301–302 (1990)

Special Issue on Algorithms for Hypercube Computers

Guest Editor's Introduction

QUENTIN F. STOUT

Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, Michigan 48109-2122

Hypercube computers are of significant commercial and theoretical interest. A survey of this and other journals shows many recent articles dealing with hypercube computers, and the participation in the Conference on Hypercube Concurrent Computers (which has been renamed the Distributed Memory Computing Conference) has continued to rapidly expand. Because of this interest, the time is appropriate for a special issue. In response to the call for papers, 19 papers were submitted, and 7 were selected.

Formally, a *hypercube computer of dimension d* has 2^d processing elements, labeled $0 \cdots 2^d - 1$. Processing element i has a communication link to processing element k if and only if $|i - k|$ is an integral power of 2. Equivalently, if the labels of the processing elements are viewed as d-bit strings, then two processing elements have a communication link between them if and only if their labels differ by a single bit. Figure 1 shows some small hypercubes. Note that d is the logarithm, base 2, of the number of processing elements, and hence algorithms finishing in $\Theta(d)$ time are completed in a time which is logarithmic in the number of processing elements.

The hypercube interconnection network has several properties that make it attractive. In a hypercube of dimension d, any two processors can communicate by sending messages along a path of at most d links, so the worst-case time for communication for a single message is $\Theta(d)$. A variety of simple tasks, such as broadcasting one data value to all processing elements, or forming a sum or maximum of data values in all processing elements, can also be easily completed in $\Theta(d)$ time.

The first design of a hypercube computer was apparently that of Squire and Palais [2], who proposed a gargantuan 12-dimensional hypercube in 1963. Despite a few other studies, hypercubes did not become popular until 20 years later, when the Cosmic Cube was built at the California Institute of Technology [1]. Motivated by the Cosmic Cube, several companies quickly started work on hypercube computers, with two of them, Intel and NCUBE, still manufacturing such machines, each having sold more than a hundred hypercubes. The Cosmic Cube and the Intel and NCUBE machines are medium-grained multiple-instruction multiple-data (MIMD) machines in which each processing element is a microprocessor with its own memory and program. Another hypercube machine, the Connection Machine from Thinking Machines Corporation, is a fine-grained single-instruction multiple-data (SIMD) machine with a controller which issues the same instruction to all processing elements, each of which is a bit-serial processor with its own data memory.

The papers in this issue illustrate the wide range of problems related to using hypercube computers to efficiently solve problems. The paper "Embedding Meshes in Boolean Cubes by Graph Decomposition," by Ho and Johnsson, is concerned with mapping rectangular grids onto the processors of a hypercube so that neighbors on the grid are close to each other in the hypercube. Concerns of this sort arise when a problem has some fixed, natural communication pattern specified by a graph where nodes represent computation and links represent communication. In this case the problem graph is a grid, and the goal is to map the problem graph onto the target machine so that communication delay is minimized. The paper "Squashed Embedding of E-R Schemas in Hypercubes," by Baru and Goel, takes a different approach to the mapping problem in that it requires that neighbors in the problem graph are mapped to neighbors in the hypercube, but now nodes are mapped to subcubes or pairs of adjacent subcubes. This approach to the mapping problem was suggested by their work on implementing an Entity-Relationship database model on a hypercube. The paper "Run-Time Scheduling and Execution of Loops on Message Passing Machines," by Saltz, Crowley, Mirchandaney, and Berryman, also addresses a problem of minimizing communication delay, where a program is given and the goal is to parallelize the inner loops. Here the data dependencies are analyzed so that processors can send the data needed by other processors without an explicit request, resulting in a significant speedup.

The remaining four papers each deal with a different application area, showing some of the range of problems for which hypercubes can give efficient solutions. The paper by Dehne and Rau-Chaplin, "Implementing Data Structures on a Hypercube Multiprocessor, and Applications in Paral-

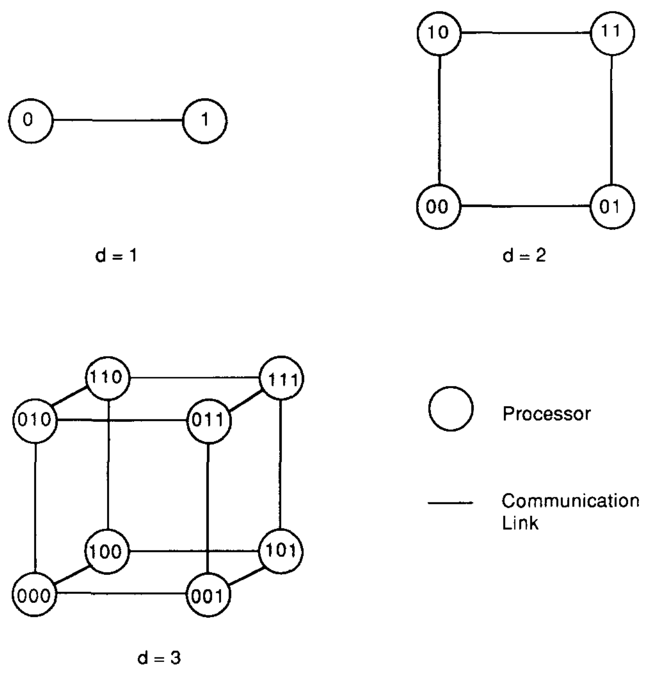

FIG. 1. Some small hypercubes.

lel Computational Geometry," constructs distributed data structures which permit simultaneous queries, and then uses these data structures to give hypercube algorithms for triangulating a simple polygon and for constructing the trapezoid map of a set of line segments. The paper by Davis, Mannix, Shaw, and Hartrum, "Distributed Discrete-Event Simulation Using Null Message Algorithms on Hypercube Architectures," also uses a distributed data structure, the distributed events list, to achieve speedups on discrete-event simulations. The paper by Eberlein and Park, "Efficient Implementation of Jacobi Algorithms and Jacobi Sets on Distributed Memory Architectures," gives both the data structures and the communication needed for efficient implementation of the Jacobi algorithm to compute the eigenvalues and eigenvectors of a matrix. Finally, the paper by Olukotun and Mudge, "Hierarchical Gate Array Routing on a Hypercube Multiprocessor," uses a top-down approach to perform the computationally intensive task of routing in gate arrays.

ACKNOWLEDGMENTS

I thank the referees for their considerable help in selecting and improving the papers for this special issue.

REFERENCES

1. Seitz, C. L. The cosmic cube. *Comm. ACM* **28** (1985), 22–33.
2. Squire, J. S., and Palais, S. M. Programming and design considerations for a highly parallel computer. *AFIPS Conf. Proc.* **23** (1963), 395–400.

Received February 8, 1989; revised November 12, 1989