Quadtrees and Hypercubes: Grid Embedding Strategies Based on Spatial Data Structure Addressing

L. A. BREENE

Department of Computer Science, Texas A&M University, TX 77843, USA

A uniform toroidal addressing scheme for k-trees, or spacial data structures, is given. These include bintrees, for decomposing the line, or partitioning linear arrays; quadtrees, for two-dimensional structures; octrees, for three dimensions; etc. Reinterpretation of these addresses as hypercube node identifiers affords simple conceptualization of processor grids of arbitrary Euclidean dimension. Use of gray code produces a hierarchy of topological neighborhoods reflected in the addresses themselves and, with this, fast multicast algorithms for various multiple-processor subgroups.

Received November 2, 1992, revised February 12, 1993

1. INTRODUCTION

Quadtrees began as hierarchical spacial data structures which store two-dimensional array, or lattice data [10], although recently the term has become generic applying now to any number of dimensions. These structures are used primarily in image processing and sparse matrix algorithms [11]. They were first introduced by Hunter and Steiglitz [6], with Gargantini [3] subsequently giving a locational code, or tree address, assignment strategy which associates each node with its position in the space/array.

In this paper we present a new addressing strategy and interpret the addresses so defined so as to embed grids of various dimension in hypercubes [1, 2, 4, 9, 12]. Our addressing strategy requires a minimal number of bits to store [5] and makes use of gray code not only to characterize the geometry of the plane by preserving topological neighborhood relationships at each level of decomposition, but also to directly relate this geometry to the hypercube neighborhoods.

Reinterpretation of the addresses as hypercube node id's gives embeddings of arbitrary dimension and facilitates the development of $\log p$ (p processors) multicasts, i.e. single source broadcasts to 'subspaces' of processors. While all topologically nearest neighbors can be reached in at most $\log p$ steps, half can be reached in one. The distance and path is determined from the node address and characterizes the relative positions of the neighbors in the tree. Our approach generalizes naturally from one (the bintree) to k dimensions (the k-tree).

In the next sections, methods for assigning tree addresses and for establishing a correspondence between these and hypercube node addresses are described, example multicasts for the one and two dimensional embeddings are presented, and application sketches are given.

2. ADDRESSING

The k-tree is a hierarchical spacial data structure used to represent the discrete k-dimensional grid, or lattice, with all dimensions of size n, where $n=2^m$ and m is the height of the tree. The root node represents the grid taken as a whole. It has 2^k children corresponding to a decomposition of the space into 2^k subspaces of equal extent with all sides of length $n/2$. When $k=1$, the grid covers a line segment and the tree is a bintree. The left and right halves of the line segment correspond to the left and right children of the root and are labeled *0* and *1*. When $k=2$, the tree is a quadtree, and the nodes (from left to right) representing the subsquares are labeled *00*, *01*, *11*, *10* (0 through 3 in gray code, defined in more detail below), starting at the lower left, and proceeding in a counterclockwise direction (see Figure 1), as Karnaugh [7] labeled the circuit simplifying charts described by Veitch [13]. Gray code labeling here ensures that neighboring subsquares (and the subspaces in higher dimensions) differ in at most a single bit at the current level. When $k=3$, a octree, subcubes at front and back are labeled as in the quadtree, with the front cube addresses prefixed by *0* and back addresses by *1*. When $k>3$, successive dimensions contribute additional digits to the address on the left in the same way.

This procedure gives a k digit label to each of the subspaces resulting from a single decomposition. The decompositions continue until each node (now leaf) represents a subspace at the resolution limit which is determined by the number of processors (for compression, decomposition ends when the subspace is of one colour or value). These subsequent decompositions concatenate substrings of length k on the right to the current label, giving a label, or address, of length mk.

The labels map to lattice points as follows. We fix the origin, $(x_1, \ldots, x_i) = (0, \ldots, 0)$, at one of the lattice points whose neighborhood is of minimum cardinality, i.e. an