Working Paper

Allocation of Complex Objects in Hypercube

Motoyasu Nagata

WP-92-20
February 1992

IIASA International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 715210 □ Telex: 079 137 iiasa a □ Telefax: +43 2236 71313

Allocation of Complex Objects in Hypercube

Motoyasu Nagata

WP-92-20
February 1992

International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 715210 □ Telex: 079 137 iiasa a □ Telefax: +43 2236 71313

Foreword

This paper presents allocation of complex objects in hypercube. It is shown that the data structure of a complex object that is an object-oriented database can be embedded into the hypercube with expansion 1 and dilation 1. The result about some properties of this allocation will be a guideline for parallel processing of the complex object in the hypercube multiprocessor.

Allocation of Complex Objects in Hypercube

Motoyasu Nagata

1 Introduction

The operation of a large database requires parallel processing. Several researches about parallel processing of the relational database have been developed in the hypercube multiprocessors [2], [8]. Recently researches of the parallelism for the object-oriented database have been started concurrently with development of the object-oriented database [1], [6], [7]. Yet, hypercube parallel processings of the object-oriented database have scarcely been explored.

This paper explores two problems for allocation and operation of the complex objects in the hypercube. The first problem is concerned with the embedding of the class-attribute hierarchy of the complex object into hypercube. The data structure to be embedded is a class-hierarchy indexing of the complex object. The purpose of this embedding is how to allocate classes in the class-hierarchy indexing to nodes of the hypercube with minimum distance between enbedded adjacent classes. We present some algorithms for embedding class-hierarchy indexing into the hypercube. We also present some propositions about distance between embedded superclass and its subclass. The second problem relates to internode communication for query operation of the complex object. We also discuss the relevant propositions about communication channels. These results can lead to efficient parallel processing of the complex objects.

The preliminary topological property of the hypercube is described in Section 2. Section 3 discusses the class-attribute hierarchy of the complex object. In Section 3, the indexing of the complex object is described. In Section 4, the transformation of the class-hierarchy indexing into complete binary tree is studied as preprocessing of the embedding. In Section 5, allocation of the complex object is explored. Section 6 presents the internode communication for operation of the complex object. Concluding remarks are given in Section 7.

2 Preliminary

In this section we discuss topological structure of the hypercube from viewpoints of addressing and graph [9].

Definition 2.1 The n-dimensional hypercube Q_n has 2^n nodes. Addresses of these nodes are from 0 to $2^n - 1$. Any two nodes are adjacent if and only if two binary addresses differ by one and only one bit.

Definition 2.2 Let $a_n a_{n-1}.....a_1$, $b_n b_{n-1}.....b_1$ be binary addresses of two nodes a and b of n-dimensional hypercube Q_n. The two nodes a and b are adjacent if the Hamming distance between two binary addresses $H(a,b) = \sum_{i=1}^{n} |a_i - b_i|$ is one.

Definition 2.3 Addresses of the n-dimensional hypercube Q_n are recursively constructed as follows:

(1) Addresses of two nodes of one-dimensional hypercube Q_1 are 0 and 1.

(2) Let $a_{n-1}.....a_1$ be the binary address of any node of $(n-1)$-dimensional hypercube Q_{n-1}. For two Q_{n-1}s, concatenate 0 and 1 to the leftmost bit positions of two nodes with the same address $a_{n-1}.....a_1$, and connect these two nodes.

Definition 2.4 Let $G = (V, E)$ be a graph where V is a set of nodes and E is a set of edges. Let $G_p = (V_p, E_p)$ be a product of two graphs $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$, denoted by $G_1 \times G_2$, where the set of nodes $V_p = V_1 \times V_2$. Two nodes $u = (u_1, u_2)$ and $v = (v_1, v_2)$ are adjacent in G_p if and only if $u_1 = v_1 (u_2 = v_2)$ and $u_2 (u_1)$ is adjacent to $v_2 (v_1)$.

Definition 2.5 Graph of n-dimensional hypercube Q_n is recursively constructed as follows:

(1) Q_0 is a trivial graph with one node.

(2) $Q_n = K_2 \times Q_{n-1}$, where K_2 is a complete graph which consists of two nodes.

Some lemmas about topological properties of the hypercube have been proposed from the above definitions.

Lemma 2.6 There are no cycles of odd length in Q_n.

Proof Consider a cycle A_1, A_2, ..., A_m of Q_n, where $A_1 = A_m$. Length of the cycle $m-1$ is the sum of the bit reversing and its reversing again for some a_i $(1 \leq i \leq n)$.□

Definition 2.7 The graph is connected if there exists a path that connects any two nodes of the graph. Maximum distance between two nodes of the graph is called the diameter.

Lemma 2.8 Q_n is a connected graph of diameter n.

Proof Consider a node A with address 0.....0 of Q_n and a node B whose address has k 1s and $(n-k)$ 0s. By Definition 2.3 there exists a path with k edges that connects A and B, then Q_n is a connected graph and $k \leq n$.□

Lemma 2.9 The minimum distance between any two nodes A and B of Q_n is the Hamming distance $H(A,B)$.

Lemma 2.10 Let A and B be any two nodes of Q_n and assume that $H(A,B) < n$. Then there are $H(A,B)$ parallel paths of length $H(A,B)$ between nodes A and B.

$$a_n.....a_{n-i+2}a_{n-i+1}a_{n-i}.....a_{n-k+1}a_{n-k}.....a_1 = A$$

$$a_n.....a_{n-i+2}b_{n-i+1}a_{n-i}.....a_{n-k+1}a_{n-k}.....a_1$$

$$a_n.....a_{n-i+2}b_{n-i+1}b_{n-i}.....a_{n-k+1}a_{n-k}.....a_1$$

.....

$$b_n a_{n-1}a_{n-2}...a_{n-i+2}b_{n-i+1}b_{n-i}...b_{n-k+1}a_{n-k}...a_1$$

$$b_n b_{n-1}a_{n-2}...a_{n-i+2}b_{n-i+1}b_{n-i}...b_{n-k+1}a_{n-k}...a_1$$

.....

$$b_n b_{n-1}.....b_{n-k+2}b_{n-k+1}a_{n-k}.....a_1 = B.$$

Then there exists $H(A,B)$ parallel paths.□

Lemma 2.11 Let A and B be any two nodes of Q_n and assume that $H(A,B) < n$. Then there are n parallel paths between the nodes A and B. Moreover, the length of each path is at most $H(A,B)+2$.

Proof Assume that addresses of nodes A and B are the same as those in the proof of Lemma 2.10. We have already proved that there exist k parallel paths with length $H(A,B)$ in Lemma 2.10. We prove the existence of $(n-k)$ parallel paths with length $H(A,B)+2$ besides k parallel paths with length $H(A,B)$. First reverse a_i of address of node A, next take a procedure of Lemma 2.10, and finally reverse a_1 again. We apply these procedures for a_i, $2 \leq i \leq n-k$. Then the result holds.□

Definition 2.12 The n-dimensional binary-reflected Gray code (BRGC) G_n is recursively constructed as follows:

$$G_1 = (0,1)$$

$$G_n = (0G_{n-1}, 1\overline{G}_{n-1}),$$

where $0G_{n-1}$ is a concatenation of 0 and G_{n-1}, and $\overline{G}_{n-1}$ is a backward-sorted code of G_{n-1}.

3 Complex Object

3.1 Class-attribute hierarchy [7]

First, any entity of the real world is modeled as an object, and the object has a unique object identifier. Second, every object has a set of values for the attributes of the object and a set of methods. The methods are procedures that operate on the set of values for the attributes of the object. The value of an attribute of an object is also an object. Third, objects that have the same set of attributes and methods are grouped as a class. Any object belongs to one and only one class. Classes are divided into primitive classes and non-primitive classes. A primitive class has no attributes, that is, a primitive class is an integer, character, or Boolean. A non-primitive class has attributes. Since value of an attribute of an object is also an object, its value belongs to some class. This class is termed the domain of the attribute of the object.

Thus a new class is created as specialization of an existing class. The new class inherits all attributes and methods of the existing class. Additional attributes and methods can be specified in the new class. The new class and existing class correspond to the subclass of the existing class and the superclass of the new class, respectively. There are two kinds of inheritances, that is, single inheritance and multiple inheritance. The single inheritance means that subclass inherits attribute and methods from only one superclass. The multiple inheritance means that subclass inherits attribute and methods from multiple superclasses. The classes form a hierarchy in single inheritance. The classes form a root-directed graph in multiple inheritance. The hierarchy and root-directed graph are called class hierarchy.

The schema of the object-oriented database is represented by a class-attribute graph. The class-attribute graph consists of class/subclass links and attribute/domain links. In the case of the primitive class, the value of an attribute of an object is stored in the object. In the case of non-primitive class, the object identifier of the instance of the class is stored. The object identifier is uniquely generated when one object is created. In the object-oriented database an object refers to another object via identifiers of the instances

of the domains of the attributes of the object. Finally, the cycle is described. If B is a domain of an attribute of an object of a class A, C is a domain of an attribute of an object of a class B, and A is also a domain of an attribute of an object of a class C, then this is called a cycle. In this paper we restrict the conditions: class hierarchy is hierarchy without cycle, that is, only single inheritance without cycle exists.

3.2 Indexing [7]

An index is a data structure of the object-oriented database. An index is maintained on an attribute of a class. An index is a list of pairs (key value, list of object identifiers). The key value is a value of the indexed attribute of the object. The object identifier helps to refer an instance of the domain of the attribute of the object. The object identifier is the identifier of an object in which the indexed attribute holds the key value. There are two kinds of indexing, that is, class-hierarchy indexing and nested-attribute indexing. The class-hierarchy indexing maintains one index on the attribute for all the classes on the class-hierarchy rooted at the target class.

We consider k-ary tree, which can be regarded as class-hierarchy indexing without cycle. The k-ary tree consists of non-leaf nodes and leaf nodes. The non-leaf node contains a set of pairs (key value, list of object identifiers) of a superclass; the object identifier assigns the object of a subclass. The leaf node also contains a set of pairs (key value, list of object identifiers) of a class where the object identifier assigns the object belonging to its own class. Both the non-leaf node and leaf node have information about page addresses of objects.

4 Transformation of Indexing into Binary Tree

We obtain two propositions about transformation of the k-ary tree of the class-hierarchy indexing of the complex object into a complete binary tree. These procedures are pre-processings for the embedding of the k-ary tree into the hypercube. With no loss of generality, it is assumed that every subclass has k subclasses.

Algorithm 4.1 Transformation of class-hierarchy indexing into binary tree

The k-ary tree of the class-hierarchy indexing of the complex object is transformed into a binary tree by the following procedure:

Step 1 Assume that the k-ary tree indexing consists of one superclass node and k subclass

nodes where $2^{m-1} < k \leq 2^m$. Then add $(2^{m+1}-1)-(k+1) = 2^{m+1}-k-2$ dummy nodes to the k-ary tree.

Step 2 Construct a complete binary tree downward from a root node corresponding to the superclass by binary splitting of m times. Allocate k nodes corresponding to the subclasses into 2^m leaf nodes of the transformed complete binary tree.

In general case, the transformation is obtained by the following algorithm.

Algorithm 4.2 Transformation of class-hierarchy indexing into binary tree
Assume that k-ary tree of the complex object is of n height and every superclass has k subclasses where $2^{m-1} < k \leq 2^m$. Then the k-ary tree is transformed into a binary tree by the following procedure:

Step 1 Construct a complete binary tree downward from a root node corresponding to a superclass by binary splitting of m times. Allocate k nodes corresponding to subclasses into 2^m leaf nodes of the binary tree with $2^{m+1}-1$ nodes.

Step 2 Construct a binary tree downward from 2^m leaf nodes by binary splitting of m times. Allocate k^2 subclass nodes, whose superclasses are subclasses in the previous step, into 2^{2m} leaf nodes of the binary tree with $2^{2m+1}-1$ nodes. These k^2 nodes belong to k subtrees that are rooted at k subclass nodes in the previous step. Recursive iteration by $n-1$ times generates a complete binary tree with $2m(n-1)+1-1$ nodes. Number of the added nodes is $2^{m(n-1)+1}-1-(k^n-1)/(k-1)$.

Corollary 4.3 Distance between the superclass node and its subclass node in the transformed complete binary tree is m. The distance between the rooted superclass node and leafed subclass node in the transformed complete binary tree is $m(n-1)$.

5 Allocation of Complex Object into Hypercube

Three methods of embedding of the class-hierarchy indexing of the complex object into hypercube are presented. These are applications of previous results [4], [10] to k-ary tree as the data structure of the complex object. Some results are presented on the distance between embedded superclass and its subclass of the class-hierarchy indexing.

Definition 5.1 Consider an embedding f of a graph $G = (V, E)$ into a graph $G' = (V', E')$. Expansion is the ratio of the number of nodes in V' to the number of nodes in V. Dilation 1 is defined if $\max f(v_1, v_2) = 1$ for arbitrary vertices v_i $(i = 1, 2)$.

Lemma 5.2 [4] An embedding of complete binary tree of $2^n - 1$ nodes in n-dimensional

hypercube Q_n, by labeling the tree nodes in inorder (according to Johnson's literature) and embedding the tree by a binary encoding of the node indices, yields an embedding in which a parent node and its left descendant are at distance 1; the parent and its right descendant are at distance 2; and the right and left descendants are at distance 1 from each other.

Proof By assumption of the induction, this proposition holds for k-height complete binary tree. The root of the k-height tree is addressed by $2^{k-1}-1$. If $(k+1)$-height tree is constructed based on the two k-height trees, the root of the $(k+1)$-height tree is addressed as 2^k-1. The addresses of the left and right children of the root, 2^k-1, are $2^{k-1}-1$ and $2^{k-1}-1+2^k$, respectively. Binary representations of these three addresses are as follows:

$2^k-1 = 011\cdots\cdots1$ $(k1s)$

$2^{k-1}-1 = 001\cdots\cdots1$ $((k-1)1s)$

$2^{k-1}-1+2^k = 101\cdots\cdots1$ (rightmost $(k-1)1s$),

then $H(2^k-1, 2^{k-1}-1) = 1$, $H(2^k-1, 2^{k-1}-1+2^k) = 2$, $H(2^{k-1}-1, 2^{k-1}-1+2^k) = 1$. This proof is completed.□

From Lemma 5.2, we obtain a lemma and its relevant algorithm about distance between embedded superclass and its subclass of the class-hierarchy indexing of the complex object.

Lemma 5.3 Consider a complete binary tree of $2^{m+1}-1$ nodes in which the root node corresponds to a superclass and 2^m leaf nodes contain k, $(2^{m-1} < k \leq 2^m)$ subclasses. Then embedding of complete binary tree of $2^{m+1}-1$ nodes in Q_{m+1} with inordered labeling yields a distance between embedded root node and its leaf nodes by the following relation:

distance(root, leaf)=j with C_{j-1}^m paths, $1 \leq j \leq m+1$.

Proof The $(k+1)$-height tree has a root node addressed by 2^k-1, where addresses of leaf nodes are $2^{k+1}-2i$, $1 \leq i \leq 2^k$. We prove this lemma by induction. For 2-height tree, it is obvious that distance(root, leaf)=$H(01,00) = 1$ with 1 path and distance(root, leaf)=$(01,10) = 2$ with 1 path. For 3-height tree, the root address 011 is a concatenation of bit 1 to the rightmost bit of the root address 01 of the 2-height tree. Then

distance(root, leaf)=$H(011,000) = 2$,

distance(root, leaf)=$H(011,010) = 1$,

distance(root, leaf)=$H(011,100) = 3$,

distance(root, leaf)=$H(011, 110) = 2$.

This satisfies the result, that is, distance=1 with C_0^2 path and distance=2 with C_2^2 path. From the result of the distance between the root and leaves in the 2-height tree, $(1, 2)$, the result in the 3-height tree is obtained by the following rule:

$(1+1, 2-1, 1+1+1, 2-1+1) = (2, 1, 3, 2)$.

Assume the result holds for the $(m+1)$-height tree, that is, distance(root, leaf)=j, with C_{j-1}^m paths, $1 \leq j \leq m+1$. For construction of $(m+2)$-height tree,

distance*(root, leaf) $\leftarrow$ distance(root, leaf)+1

(for the i-th leaf node, $0 \leq i \leq 2^m - 2$ in $(m+1)$-height tree)

distance**(root, leaf) $\leftarrow$ distance(root, leaf)+1

(for the i-th leaf node, $2^m \leq i \leq 2^{m+1} - 2$ in $(m+1)$-height tree)

distance(root, leaf) $\leftarrow$ distance*(root, leaf)+1

distance(root, leaf) $\leftarrow$ distance**(root, leaf)+1.

Then, in the $(m+1)$-height tree, the number of paths satisfying distance(root, leaf)=j is $C_{j-1}^m + C_{j-2}^m$, that is, C_{j-1}^{m+1}. □

Algorithm 5.4 Selection of k-leaf nodes

Consider embedding of a complete binary tree of $2^{m+1} - 1$ nodes into Q_{m+1} with ordered labeling. The allocation of k subclasses to 2^m leaf nodes is done by the following procedure:

Step 1 Allocate the address $2^m - 1$ to the superclass. Addresses of the leaf nodes are $0, 2, 4, ..., 2^{m+1} - 2$.

Step 2 Select k, $(2^{m-1} < k \leq 2^m)$, leaf nodes corresponding to the first k ones in ascendant ordering of Hamming distances $H(2^m - 1, 2^{m+1} - 2i)$ where $1 \leq i \leq 2^m$.

Johnsson's Lemma 5.2 clarifies tree embedding into hypercube with expansion 1. Johnsson also presents a similar result using BRGC. We give alternative proof of his result.

Lemma 5.5[4] An embedding of a complete binary tree of $2^n - 1$ nodes in Q_n, by labeling the tree nodes in inorder and embedding the tree by BRGC encoding of the nodes indices, yields an embedding in which a leaf node is at distance 1 from its parent node and all other nodes are at distance 2 from their respective parent node. Left and right descendants of a node are always at distance 2 from each other.

Proof The notation $G_n(i)$ stands for the i-th code of the n-dimensional BRGC. In the proof of Lemma 5.2, it is shown that addresses of the root, its left child and right child

are $2^k - 1$, $2^{k-1} - 1$ and $2^{k-1} - 1 + 2^k$ in the $(k+1)$-height tree. Since

$G_{k+1}(2^k - 1) = 01G_{k-1}(0) = 010G_{k-2}(0)$

$G_{k+1}(2^{k-1} - 1) = 00G_{k-1}(2^{k-1} - 1) = 001G_{k-2}(0)$

$G_{k+1}(2^{k-1} - 1 + 2^k) = 11G_{k-1}(2^{k-1} - 1) = 111G_{k-2}(0)$,

then $H(010G_{k-2}(0), 001G_{k-2}(0)) = 2$,

$H(010G_{k-2}(0), 111G_{k-2}(0)) = 2$,

$H(001G_{k-2}(0), 111G_{k-2}(0)) = 2$.

This proof is completed.□

From Lemma 5.5, we obtain a lemma about distance between root and leaf nodes, and its relevant algorithm.

Lemma 5.6 Embedding of complete binary tree of $2^{m+1} - 1$ nodes in Q_{m+1} with BRGC labeling in inorder, yields odd distances between embedded root node and its leaf nodes by the following relation:

distance(root, leaf)=m for Q_{m+1} ($m+1$: even)

distance(root, leaf)=$m+1$ for Q_{m+1} ($m+1$: odd)

To prove Lemma 5.6, we use the following lemma.

Lemma 5.7 [4] The BRGCs that encode i and $i + 2^k \mod 2^n$ differ in 2 bits.

Proof For two integers i and j whose binary representations are

$$i_n i_{n-1} \ldots\ldots\ldots i_1$$

$$j_n j_{n-1} \ldots\ldots\ldots j_1,$$

two BRGCs that correspond to i and j are assumed as follows:

$$g_n g_{n-1} \ldots\ldots\ldots g_1$$

$$h_n h_{n-1} \ldots\ldots\ldots h_1.$$

Then

$$i_m = j_m, m = \{1, 2, \ldots\ldots\ldots, k\}$$

$$j_m = \bar{i}_m, m = \{k+1, \ldots\ldots\ldots, s\},$$

where carry stops propagation at bit position s. From Gray's theorem, the result is obtained

$$h_m = g_m, m = \{1, 2, \ldots\ldots, k-1, k+1, \ldots\ldots, s\}$$

$$h_k = \overline{g}_k, h_s = \overline{g}_s,$$

where the overline stands for complement of the bit.□

Proof of Lemma 5.6 BRGCs of the leaf nodes in the $(m+1)$-height complete binary tree are $G_{m+1}(2i)$, $0 \leq i \leq 2^m - 1$. Since $G_{m+1}(2^m - 1) = 01G_{m-1}(0)$ and $G_{m+1}(0) = 00G_{m-1}(0)$, then $H(G_{m+1}(2^m - 1), G_{m+1}(0)) = 1$. As the result of Lemma 5.7, $G_{m+1}(2)$ and $G_{m+1}(0)$ $mod 2^{m+1}$ differ in 2 bits. Then $H(G_{m+1}(2^m - 1), G_{m+1}(2)) = 1$ or 3, that is, odd. Similarly, $H(G_{m+1}(2^m - 1), G_{m+1}(2i))$ =odd, $2 \leq i \leq 2^m - 1$. □

Algorithm 5.8 Selection of k leaf nodes

Consider embedding of a complete binary tree of $2^{m+1} - 1$ nodes into Q_{m+1} with BRGC labeling in inorder. Allocation of k subclasses to 2^m leaf nodes is done by the following procedure:

Procedure Select k $(2^{m-1} < k \leq 2^m)$ leaf nodes which correspond to first k ones in ascendant ordering of Hamming distances

$H(G_{m+1}(2^{m-1}), G_{m+1}(2^{m+1} - 2i))$(=odd) where $1 \leq i \leq 2^m$.

Next we consider the embedding with dilation 1.

Definition 5.9 Assume that f_p is a map from binary tree of height p into hypercube Q_{p+1}. The free-free neighbor property is defined if $R = f_p$(binary tree of height p) has a free neighbor R_1 and R_2 has a free neighbor R_2, then $\{R_1, R_2\}$ is not a subset of f_p(nodes of binary tree of height p).

As the straightforward application of the tree embedding [10], we can obtain the following result.

Lemma 5.10 The k-ary tree of the complex object such that every subclass has k $(2^{m-1} < k \leq 2^m)$ subclasses, with height n, can be embedded into a hypercube $Q_{m(n-1)+2}$ with dilation 1.

Proof The following inductive algorithm proves the result.

Algorithm 5.11 Embedding of k-ary tree of complex object into $Q_{m(n-1)+2}$

Step 1 Left subtree of height $m(n-1)$ is embedded by $f_{m(n-1)}$ into $0Q_{m(n-1)+1}$ of $Q_{m(n-1)+2}$. A node $0L = f_{m(n-1)}$(root of left subtree) has a free neighbor $0L_1$ which has a free neighbor $0L_2$.

Step 2 Right subtree of height $m(n-1)$ is also embedded by $g_{m(n-1)}$ into $1Q_{m(n-1)+1}$ of $Q_{m(n-1)+2}$. A node $1R = g_{m(n-1)}$(root of right subtree) has a free neighbor $1R_1$ that has a free neighbor $1R_2$.

Step 3 A hypercube $Q_{m(n-1)+2}$ is constructed by combination of $0Q_{m(n-1)+1}$ and $1Q_{m(n-1)+1}$ in such a way that $0L_1$ is a neighbor of $1R$ and $0L_2$ is a neighbor of $1R_1$.□

Lemma 5.12 Embedding of the k-ary tree in the above proposition into $Q_{m(n-1)+2}$ with dilation 1 yields distance m between superclass and its subclass, and distance $m(n-1)$ between rooted superclass and leafed subclass.

6 Hypercube Operation of Complex Object

We present some results about communication channels, i.e., parallel paths, in the hypercube for query operation of the complex object, when the k-ary tree is embedded with expansion 1. Consider the following query problem [3] [5]:

Query complex object that satisfies a query condition such that

$$C(1)C(2).....C(n) = A(1)A(2).....A(n),$$

where $C(1)C(2).....C(n)$ is a sequence of classes of the complex object. $C(i)$ is a class of the i-th level and $A(i)$ is an instance corresponding to a class $C(i)$.

Lemma 6.1 Under the embedding of k-ary tree into $Q_{m(n-1)+1}$ with inordered labeling, consider a query $C(1).....C(k+1) = A(1).....A(k+1)$ from the accessed result $C(1).....C(k) = A(1).....A(k)$, $k+1 \leq n$. Then the number of parallel paths for this query is $(m+2)2^{m-1}$ with distance(superclass, subclass)= m in transformed binary tree.

Proof From Lemma 2.10 and Lemma 5.3, this query requires $C_{j-1}^{m} j$ parallel paths with distance(root, leaf)= j, where the number of leaf nodes is the sum of C_{j-1}^{m}, $1 \leq j \leq m+1$, that is, $\sum_{j-1}^{m+1} C_{j-1}^{m} = 2^m$. Then the sum of parallel paths is calculated as follows: $\sum_{j=1}^{m+1} C_{j-1}^{m} j = (m+2)2^{m-1}$.□

Lemma 6.2 Under the embedding of the k-ary tree into $Q_{m(n-1)+1}$ with inorder labeling, consider a query $C(1).....C(k+1) = A(1).....A(k+1)$ from the accessed result $C(1).....C(k) = A(1).....A(k)$, $k+1 \leq n$. Then number of parallel paths for this query is $m2^m$ where the length of each path is at most distance(root, leaf)+2.

Proof From Lemma 2.11 and Lemma 5.3, this query requires $C_{j-1}^{m} m$ parallel paths with at most $j+2$ length, where distance(root, leaf)= j. Then the sum of parallel paths is calculated as follows: $\sum_{j=1}^{m+1} C_{j-1}^{m} m = m2^m$.□

We also obtain a proposition about parallel paths of communication for the query of the complex object embedded with dilation 1.

Lemma 6.3 Under the embedding of the k-ary tree into $C_{m(n-1)+2}$ by Algorithm 5.11, consider the above-mentioned query of complex object. Then the number of parallel paths

for this query is $m2^m$.

7 Concluding Remarks

This paper investigates allocation of a data structure of the complex objects into the hypercube. Several theoretical properties were obtained concerning the distance between embedded superclass and its subclass of the class-hierarchy indexing of the complex object. The allocation of an arbitrary tree as a data structure of the complex object into the hypercube, the query processing, and concurrency control of the complex object in the hypercube are still open problems.

References

[1] M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier and S. Zdonik, The Object-Oriented Database System Manifesto, *Proceedings First International Conference on Deductive and Object-Oriented Databases*, 40-57, 1989.

[2] C. K. Baru and O. Frieder, Database Operations in a Cube-Connected Multicomputer System, *IEEE Transaction on Computers*, **38**, (6), 920-927, 1989.

[3] E. Bertino and W. Kim, Indexing Techniques for Queries on Nested Objects, *IEEE Transaction on Knowledge and Data Engineering*, **1**, (2), 196-214, 1989.

[4] S. L. Johnsson, Communication Efficient Basic Algebra Computations on Hypercube Architectures, *Journal of Parallel and Distributed Computing*, **4**, 133-172, 1987.

[5] W. Kim, H. T. Chou and J. Banerjee, Operations and Implementation of Complex Objects, *IEEE Transaction on Software Engineering*, **14**, (7), 985-996, 1988.

[6] K. C. Kim, Parallelism in Object-Oriented Query Processing, *Proc. Sixth International Conference on Data Engineering*, 1990.

[7] W. Kim, Architectural Issues in Object-Oriented Databases, *Journal of Object-Oriented Programming*, 1990.

[8] E. R. Omiecinski and E. T. Lin, Hash-Based and Index-Based Join Algorithms for Cube and Ring-Connected Multicomputers, *IEEE Transaction on Knowledge and Data Engineering*, **1**, (3), 329-343, 1989.

[9] Y. Saad and M. H. Schultz, Topological Properties of Hypercubes, *IEEE Transaction on Computers*, **37**, (7), 867-872, 1988.

[10] A. Y. Wu, Embedding of Tree Networks into Hypercube, *Journal of Parallel and Distributed Computing*, **3**, (2), 238-249, 1985.