# Data Placement Strategy in Hierarchical Symmetrical Multiprocessor Systems [♣]

© Andrey V. Lepikhov
South-Ural State University
lepihov@susu.ru

© Leonid B. Sokolinsky
South-Ural State University
sokolinsky@acm.org

## Abstract

In this paper we propose a new approach to data placement and load balancing in multiprocessor relational hierarchical database systems. The described strategy is based on the symmetrical multiprocessor hierarchical system model. It proposes using horizontal fragmentation within each database relation and partial disk mirroring between the nodes and the levels of the multiprocessor hierarchy. Simultaneously, each fragment on a logical level is divided into equal size segments, which are the least unit of data replication and load balancing. The number of segments contained in the replica is specified by a replication factor. Proposed data placement techniques are cluster and Grid systems oriented.

## 1. Introduction

At present, hierarchical architectures of parallel database systems are becoming widely distributed [11]. This is related to the fact that modern multiprocessor systems are frequently organized according to the hierarchical principle. In fact, most supercomputers have a two-level cluster architecture today. According to this architecture a multiprocessor system is designed as a number of homogenous computing units connected by a high-speed interconnection network. In addition, each computing unit, in turn, is a multiprocessor system with shared memory. If multi-core processors in the system are used then we obtain a third architecture level.

Another source of multiprocessor hierarchies is Grid technology [3] allowing to combine a collection of various supercomputers into a single computing system. A Grid system has a multilevel hierarchical structure. The lower levels of the hierarchy contain the processors of the individual clusters, which are connected by a high-speed internal network. The upper levels of the hierarchy contain computing systems which are connected by a corporate network. The top level of the hierarchy can be represented by the Internet.

Many of studies have been dedicated to the data allocation and load balancing problem in shared nothing parallel database systems (e.g. [1, 2, 7, 12]), however, these problems have not been studied in a hierarchical multiprocessor systems context before.

The remainder if this paper is organized as follows. Section 2 introduces a formal definition of the symmetrical multiprocessor hierarchy. Section 3 describes a fragmentation and load balancing mechanism based on the definition of a segment. Section 4 presents a data replication mechanism based on segmentation, and introduces the definition of replication factor. Section 4.2 describes a partial mirroring technique utilizing a replication function, which compares the replication factor to each hierarchy level. Here, total replica's size estimation theorems are proved. Section 4.3 is dedicated to the problem of replication function choosing, and a regular hierarchy definition is introduced. Theorems for estimating overhead during the formation of replicas without interference are proven. Finally, Section 5 contains the obtained results and discussion for future work.

## 2. Symmetrical hierarchies

Hierarchical multiprocessor configurations are characterized by a wide variety of architectures [11]. Frequently, the multiprocessor system architecture is a determining factor in the development of a data placement strategy. In this section we will consider a hierarchical multiprocessor database systems model which we call a *symmetrical model*. It describes a sufficiently wide class of existing configurations, and is the mathematical base for the data allocation strategy proposed in this paper.

The symmetric model is based on the *DM–tree* concept introduced in the study [5]. A *DM*–tree is a directed tree and is an abstraction of hierarchical multiprocessor database systems.

For any given *DM*–tree $T$ we will denote the set of its nodes as $\mathfrak{M}(T)$, and the set of the edges as $\mathfrak{E}(T)$. The set of nodes $\mathfrak{M}(T)$ of any *DM*–tree $T$ is divided into three disjoint subsets, called classes:

$\mathfrak{P}(T)$ – class of «processor modules»;

♣ This work was supported by the Russian foundation for basic research (project 06-07-89148).

**Proceedings of the Spring Young Researcher's Colloquium On Database and Information Systems SYRCoDIS, St.-Petersburg, Russia, 2006**

$\mathfrak{D}(T)$ – class of «disk modules;

$\mathfrak{N}(T)$ – class of «hub modules».

We assign a *overhead coefficient* $\eta(v)$ to each node $v \in \mathfrak{M}(T)$ in the *DM*–tree *T*. This overhead coefficient is a real number greater or equal to 1 and defines the time required this node to process 1 data unit. For example, the unit of data can be a tuple.

Here we will give a definition for isomorphism of two *DM*–trees.

Let *A* and *B* be *DM*–trees. *A* and *B* are isomorphic if there exists a one-to-one function $f$ from the set $\mathfrak{M}(A)$ to the set $\mathfrak{M}(B)$ and a one-to-one function $g$ from the set $\mathfrak{E}(A)$ to the set $\mathfrak{E}(B)$ and the following conditions hold true:

1) node $v$ is an endpoint of edge $e$ in tree *A* if and only if node $\mathrm{f}(v)$ is an endpoint of edge $\mathrm{g}(e)$ in tree *B*;

2) node $w$ is an initial point of edge $e$ in tree *A* if and only if node $\mathrm{f}(w)$ is an initial point of edge $\mathrm{g}(e)$ in tree *B*;

3) $p \in \mathfrak{P}(A) \quad \Leftrightarrow \quad \mathrm{f}(p) \in \mathfrak{P}(B)$;

4) $d \in \mathfrak{D}(A) \quad \Leftrightarrow \quad \mathrm{f}(d) \in \mathfrak{D}(B)$;

5) $n \in \mathfrak{N}(A) \quad \Leftrightarrow \quad \mathrm{f}(n) \in \mathfrak{N}(B)$;

6) $\eta(\mathrm{f}(v)) = \mathrm{f}(\eta(v))$.

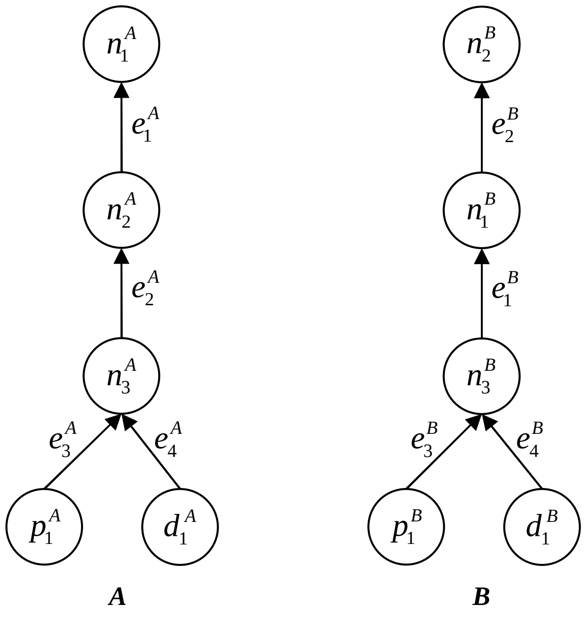

**Figure 1.** Non isomorphic DM-trees (condition 2 not meet).

We will call the pair of maps $\mathrm{q} = (\mathrm{f}, \mathrm{g})$ isomorphism of *DM*–tree *A* to *DM*-tree *B*.

Note that condition 2 from the isomorphism definition is not redundant. This is confirmed by the example expressed in Figure 1. In fact, using the node and edge numeration shown in Figure 1, we define the functions **f** and **g** in the following way:

$$\mathrm{f}(n_i^A) = n_i^B, \mathrm{f}(p_i^A) = p_i^B, \mathrm{f}(d_i^A) = d_i^B,$$
$$\mathrm{g}(e_i^A) = e_i^B.$$

Obviously, the maps **f** and **g** satisfy all of the isomorphism conditions, except the second condition (we are supposing that a overhead coefficient for all nodes is equal to 1). However, we can see that $\mathrm{q} = (\mathrm{f}, \mathrm{g})$ is not isomorphism of *DM*–tree *A* to *DM*-tree *B*, because the owner-member relationship is not satisfied (node *w* is subordinate to node *v* if exists an edge directed from *w* to *v*). In fact, the node $n_2^A$ is governed by $n_1^A$, in *DM*-tree *A* and node $\mathrm{f}(n_2^A) = n_2^B$ is a parent of node $\mathrm{f}(n_1^A) = n_1^B$ in *DM*-tree *B*.

We will define *node level* in a recursive manner [4]. The level of the root of *DM*-tree *T* is equal to zero and the level of any other node by one unit larger than the minimal subtree root level of tree *T*, which contains this node.

We use a *subtree level* definition in the sense of this subtree root level.

Two subtrees are called adjacent if their roots are brothers, i.e. the tree contains a node which is a parent relative to the subtrees root node.

Let the *height* of directed tree *T* be defined as the maximum path length in that tree.

We will call *symmetrical* a *DM*-tree *T* with height H if the following conditions hold:

1) any two adjacent subtrees with level $l < H$ are isomorphic trees;

2) any subtree with level $H-1$ contain exactly one disk and one processor.

In the Symmetry definition condition 2 is a SMP-system abstract model in the sense that in the multiprocessor hierarchies context all the processors of an SMP-system can be considered as a «mega processor» and all the disks as «mega disk». Obviously, load balancing on an SMP-system must be solved by essentially different techniques, since an SMP-system has shared memory and all the disks can be accessed by all processors (see the related studies [6, 9]).

We define *node degree* as the number of edges which end into this node. In a symmetrical tree nodes of the same level have the same degree, called the *level degree.*

## 3. Data fragmentation and segmentation

Each relation is divided into disjoint horizontal fragments [12], which are partitioned into different disk units. We will suppose that a fragment's tuples are in some way, ordered. The order is fixed for each query and defines a tuple reading sequence in the scan fragment operation. We will call this order natural. In practice, a natural order can be defined by an index or a physical sequence order of tuples.

At the logical level each fragment is divided into a sequence of fixed-length *segments*. Segment length is measured in tuples and is a fragment attribute. Partitioning of segments is performed according to natural order and always begins with the first tuple. In accordance with this, the last segment may be incomplete.

The number of segments in fragment *F* is denoted by $\mathrm{S}(F)$ and can be computed by the formula:

$$\mathrm{S}(F) = \left\lceil \frac{\mathrm{T}(F)}{\mathrm{L}(F)} \right\rceil . \qquad (1)$$

Where $\mathrm{T}(F)$ is the number of tuples in fragment *F*, $\mathrm{L}(F)$ - the segment length assigned to fragment *F*.

Consequently, a segment is the smallest data fragmentation unit.

## 4. Data replication

### 4.1 Replica building algorithm

Let us allocate fragment $F_0$ into disk unit $d_0 \in \mathfrak{D}(T)$ of multiprocessor hierarchical system *T*. We suppose that in each disk unit $d_i \in \mathfrak{D}(T) \quad (i > 0)$ a partial replica $F_i$ is allocated containing some subset of tuples (can be empty) of fragment *F*.

Replica segment length is always equal to the replicated fragment segment length.

$$\mathrm{L}(F_i) = \mathrm{L}(F_0), \qquad \forall d_i \in \mathfrak{D}(T) .$$

The size of replica $F_i$ is specified by a *replication factor*

$$\rho(F_i) \in \mathbb{R}, \quad 0 \le \rho(F_i) \le 1 ,$$

which is an attribute of replica $F_i$ and is computed by the formula:

$$\mathrm{T}(F_i) = \mathrm{T}(F_0) - \left\lceil \left(1 - \rho(F_i)\right) \cdot \mathrm{S}(F_0) \right\rceil \cdot \mathrm{L}(F_0) \qquad (2)$$

The natural order of tuples in replica $F_i$ is defined by the tuples natural order in fragment $F_0$. Here, the following formula determines the first tuple number of replica $F_i$:

$$\mathrm{N}(F_i) = \mathrm{T}(F) - \mathrm{T}(F_i) + 1 .$$

For empty $F_i$ we obtain $\mathrm{N}(F_i) = \mathrm{T}(F_0) + 1$, corresponding to the end-of-file indicator.

Consequently, segment is also the smallest data replication unit.

The above data replication technique permits the use of a simplistic and effective load balancing technique in multiprocessor hierarchies. The method description is beyond the scope of this paper; however, simplified version is accessible in study [10].

### 4.2. Partial mirroring technique

Let *T* be a *DM*-tree with height $H = h(T) > 1$. We introduce a replication function $r(l)$ which maps replication factor $\rho_l = r(l)$ to each level $l < H$ of tree *T*.

Let us assume, that $r(H-1) = 1$. This arose from the fact that hierarchy level $H-1$ contains subtrees with height equal 1, which corresponds to an SMP-systems. In SMP-system all the disks can be accessed by all processor, therefore, physical replication is not needed. Load balancing on the logical level is by way of the source fragment segmentation, i.e. the fragment plays its replica role.

Let us calculate the replication function $r(l)$ for $l \le H-2$ values.

Let us allocate fragment $F_0$ to disk unit $d_0 \in \mathfrak{D}(T)$. In order to form replica $F_i$ in disk $d_i \in \mathfrak{D}(T) \quad (i > 0)$ the following technique (called a partial mirroring technique) will be used. We will construct a sequence of subtrees for tree T

$$\{M_0, M_1, \ldots, M_{H-2}\} , \qquad (3)$$

possessing the following properties:

$$\begin{cases} l(M_j) = j \\ d_0 \in \mathfrak{D}(M_j) \end{cases}$$

for each *j*, where $0 \le j \le H-2$. In this formula $l(M_j)$ is a level of subtree $M_j$. Obviously, such a sequence exists for any symmetrical tree *T*.

In order to find the largest index $j \ge 1$ such that

$$\{d_0, d_i\} \subset \mathfrak{D}(M_j) .$$

We assume, that

$$\rho(F_i) = r(j) . \qquad (4)$$

To build replica $F_i$ into disk $d_i$ we use the algorithm described in section 4.1, with a replication factor, defined by formula (4).

The following theorem gives estimation for the replica size.

**Theorem 1.** Let *T* be a symmetrical *DM*-tree having height $H = h(T) > 0$. Let us allocate fragment $F_0$ to disk unit $d_0 \in \mathfrak{D}(T)$. Let M be a subtree of T such that $1 \le l(M) \le H-1$ and $d_0 \in \mathfrak{D}(M)$. Let $M'$ be any subtree adjacent to subtree *M* of tree *T*. Then for any $d_i \in \mathfrak{D}(M')$ for the size of replica $F_i$ of fragment $F_0$ allocated to disk $d_i$ the following estimation is valid:

$$\mathrm{T}(F_i) = r\left(l(M) - 1\right)\mathrm{T}(F_0) + \mathrm{O}\left(\mathrm{L}(F_0)\right)$$

Where $\mathrm{L}(F_0)$ is the segment length of fragment $F_0$.

P r o o f . According to formula (2) we obtain

$$\mathrm{T}(F_i) \quad = \mathrm{T}(F_0) - \left\lceil \left(1 - \rho(F_i)\right) \cdot \mathrm{S}(F_0) \right\rceil \cdot \mathrm{L}(F_0)$$

it follows that

$$\mathrm{T}(F_i) \quad = \mathrm{T}(F_0) - \left(1 - \rho(F_i)\right) \cdot \mathrm{S}(F_0) \cdot \mathrm{L}(F_0) + \mathrm{O}\left(\mathrm{L}(F_0)\right) . \ (5)$$

Since $d_0 \in \mathfrak{D}(M)$, $d_i \in \mathfrak{D}(M')$ and subtrees *M* and $M'$ are adjacent, then the minimal subtree $\tilde{M}$ containing disks $d_0$ and $d_i$ will have a level equal to $l(\tilde{M}) = l(M) - 1$. Then according to (4) we obtain $\rho(F_i) = r(l(M) - 1)$. Substituting this value into (5) we obtain:

$$\mathrm{T}(F_i) = \mathrm{T}(F_0) - \left(1 - r\left(l(M) - 1\right)\right) \cdot \mathrm{S}(F_0) \cdot \mathrm{L}(F_0) + \mathrm{O}\left(\mathrm{L}(F_0)\right) .$$

Substituting for $\mathrm{S}(F_0)$ the value from (1) we obtain

$$\begin{aligned}\mathrm{T}(F_i) &= \mathrm{T}(F_0)-\left(1-r\left(l(M)-1\right)\right)\cdot\left\lceil\frac{\mathrm{T}(F_0)}{\mathrm{L}(F_0)}\right\rceil\cdot\mathrm{L}(F_0)+\mathrm{O}\left(\mathrm{L}(F_0)\right)\\ &= \mathrm{T}(F_0)-\left(1-r\left(l(M)-1\right)\right)\cdot\frac{\mathrm{T}(F_0)}{\mathrm{L}(F_0)}\cdot\mathrm{L}(F_0)+\mathrm{O}\left(\mathrm{L}(F_0)\right)\\ &= \mathrm{T}(F_0)-\left(1-r\left(l(M)-1\right)\right)\mathrm{T}(F_0)+\mathrm{O}\left(\mathrm{L}(F_0)\right)\\ &= r\left(l(M)-1\right)\mathrm{T}(F_0)+\mathrm{O}\left(\mathrm{L}(F_0)\right)\end{aligned}$$

The theorem is proven.

The estimation of the total size of all the fragment replicas can be obtained by the following theorem.

**Theorem 2.** Let *T* be a symmetrical *DM*-tree having height $H \geq 2$. The degree of level *l* of tree *T* we will denote as $\delta_l$. Let $\mathrm{R}(F_0)=\sum_{i=1}^{|\mathfrak{D}(T)|}\mathrm{T}(F_i)$ - total quantity of tuples in all the replicas in fragment $F_0$. Then

$$\mathrm{R}(F_0)=\mathrm{T}(F_0)\sum_{j=0}^{H-2}r(j)(\delta_j-1)\prod_{k=j+1}^{H-2}\delta_k+\mathrm{O}\left(\mathrm{L}(F_0)\right). \quad (6)$$

Proof. This proof is by induction on the height of tree *T*.

Given $H=2$. Then the number of disks in the tree *T* is equal $\delta_0$. In accordance to theorem 1, each disk contains a replica with size $\mathrm{T}_0(F_0)=r(0)\mathrm{T}(F_0)+\mathrm{O}\left(\mathrm{L}(F_0)\right)$. Therefore, the total number of tuples in all the replicas of fragment $F_0$ can be estimated by

$$\begin{aligned}\mathrm{R}(F_0) &= \left(\mathrm{T}\left(F_0\right)r\left(0\right)+\mathrm{O}\left(\mathrm{L}(F_0)\right)\right)\cdot\left(\delta_0-1\right)\\ &= \mathrm{T}\left(F_0\right)r\left(0\right)\left(\delta_0-1\right)+\mathrm{O}\left(\mathrm{L}\left(F_0\right)\right)\end{aligned}, \quad (7)$$

This is confirmed by formula (6) at $H=2$.

Let $H>2$. Then tree *T* contains $\delta_0$ subtrees having height $H-1$:

$$M_0, M_1, \ldots, M_{\delta_0-1}.$$

Let us denote the total number of tuples in all the replicas of fragment $F_0$ allocated to the all disks of tree $M_j$ by $R_j(F_0)$. Obviously that

$$\mathrm{R}(F_0)=\sum_{j=0}^{\delta_0-1}\mathrm{R}_j(F_0). \quad (8)$$

Without loss of generality we may assume that $d_0\in\mathfrak{D}(M_0)$. Since *DM*-tree *T* is symmetrical, from (8) we obtain:

$$\mathrm{R}(F_0)=\mathrm{R}_0(F_0)+(\delta_0-1)\mathrm{R}_1(F_0). \quad (9)$$

In accordance with theorem 1, any replica $F_i$ allocating in subtree $M_1$ has the follow size:

$$\mathrm{T}(F_i)=r\left(0\right)\mathrm{T}\left(F_0\right)+\mathrm{O}\left(\mathrm{L}(F_0)\right). \quad (10)$$

Because of the symmetric property of *DM*-tree *T*, the total number of disks in subtree $M_1$ is equal to $\prod_{k=1}^{H-2}\delta_k$. Taking this fact into account, from (8) we obtain:

$$\mathrm{R}_1(F_0)=r\left(0\right)\mathrm{T}\left(F_0\right)\prod_{k=1}^{H-2}\delta_k+\mathrm{O}\left(\mathrm{L}(F_0)\right). \quad (11)$$

On the other hand, by induction we have:

$$\mathrm{R}_0(F_0)=\mathrm{T}(F_0)\sum_{j=1}^{H-2}r(j)(\delta_j-1)\prod_{k=j+1}^{H-2}\delta_k+\mathrm{O}\left(\mathrm{L}(F_0)\right). \quad (12)$$

Plugging in (9) the right hand side values from (11) and (12) we have

$$\begin{aligned}\mathrm{R}(F_0) &= \mathrm{T}(F_0)\sum_{j=1}^{H-2}r(j)\prod_{k=j+1}^{H-2}\delta_k+(\delta_0-1)r\left(0\right)\mathrm{T}\left(F_0\right)\prod_{k=1}^{H-2}\delta_k+\mathrm{O}\left(\mathrm{L}(F_0)\right)\\ &= \mathrm{T}(F_0)\sum_{j=0}^{H-2}r(j)(\delta_j-1)\prod_{k=j+1}^{H-2}\delta_k+\mathrm{O}\left(\mathrm{L}(F_0)\right)\end{aligned}$$

The theorem is proven.

### 4.3. Choosing the replication function

When choosing the replication function $r(l)$ it is reasonable to take into consideration the overhead coefficients of the *DM*–tree nodes. Obviously, that all the vertices in a symmetrical *DM*-tree have the same overhead coefficient $\eta(l)$, which we will call *overhead of level l.*

We will call symmetrical *DM*-tree *T* regular if for any two levels $l$ and $l'$ of tree *T* the following statement holds true:

$$l<l' \quad\Rightarrow\quad \eta(l)\geq\eta(l'), \quad (13)$$

By increasing the hierarchy level, the overhead coefficient is increased.

The following theorem allows for estimation of a replica tuple-building overhead coefficient in a regular *DM*–tree.

**Theorem 3**. Let *T* be a regular *DM*–tree tree having height $H>0$. Let us allocate fragment $F_0$ to disk unit $d_0\in\mathfrak{D}(T)$. Let *M* be a subtree of tree *T* such that $1\leq l(M)\leq H-2$ and $d_0\in\mathfrak{D}(M)$. Let $M'$ be any subtree of tree *T* adjacent with *M*. $F_i$ is a replica of fragment $F_0$ allocated into $d_i\in\mathfrak{D}(M')$. Let $\tau(F_i)$ be the overhead of tuple-building of replica $F_i$ without interference. Then

$$\tau(F_i)=\eta\left(l(M)-1\right)\cdot r\left(l(M)-1\right)\mathrm{T}\left(F_0\right)+\mathrm{O}\left(\eta_0\right),$$

Here $\eta_0=\eta(0)$ is a tree root overhead coefficient.

Proof. According to the operating system model [5] we will make a pipeline to pass the tuple from disk $d_0\in\mathfrak{D}(M)$ to disk $d_i\in\mathfrak{D}(M')$. Pipeline operating speed is determined by the slowest node. Since *M* and $M'$ are adjacent subtrees their root nodes have a common parent with level $l(M)-1$ which according to (13) is the slowest pipeline chain unit. Consequently, with the pipeline operating at full performance passing one tuple, the overhead coefficient is equal to $\eta\left(l(M)-1\right)$. This implies that

$$\tau(F_i)=\eta\left(l(M)-1\right)\mathrm{T}(F_i)+\mathrm{O}\left(\eta\left(l(M)-1\right)\right). \quad (14)$$

Here $\mathrm{O}\left(\eta\left(l(M)-1\right)\right)$ is denoted as the upper time boundary which is needed for the pipeline to operate at full performance under the assumption that the height of tree *T* is constant.

Let $\eta_0 = \eta(0)$ be the overhead coefficient of the root of tree *T*. Because *T* is regular $\eta(l(M)-1) \le \eta_0$. Thus, from (14) we obtain

$$\tau(F_i) = \eta(l(M)-1)\mathrm{T}(F_i) + \mathrm{O}(\eta_0). \qquad (15)$$

By the theorem 1 from (15) we obtain

$$\begin{aligned}\tau(F_i) &= \eta(l(M)-1)\cdot r(l(M)-1)\mathrm{T}(F_0) \\ &+\mathrm{O}(\eta_0)\mathrm{O}(\mathrm{L}(F_0)) + \mathrm{O}(\eta_0)\end{aligned}. \qquad (16)$$

We can suppose that a segment length does not change in a replica building process i.e. $\mathrm{L}(F_0)$ is constant. Then from (16) we obtain

$$\begin{aligned}\tau(F_i) &= \eta(l(M)-1)\cdot r(l(M)-1)\mathrm{T}(F_0) + \mathrm{O}(\eta_0) + \mathrm{O}(\eta_0) \\ &= \eta(l(M)-1)\cdot r(l(M)-1)\mathrm{T}(F_0) + \mathrm{O}(\eta_0)\end{aligned}.$$

The theorem is proven.

An estimation for overhead coefficients for the formation of all the replicas of a fragment without interference can be obtained by the following theorem.

**Theorem 4**. Let *T* be a regular *DM*-tree having height $H \ge 2$. Let us allocate fragment $F_0$ to disk unit $d_0 \in \mathfrak{D}(T)$. The degree of level *l* of tree *T* we will denote as $\delta_l$. Let $\mathrm{t}(F_0) = \sum_{i=1}^{|\mathfrak{D}(T)|} \mathrm{t}(F_i)$ be the total overhead of tuple-building of all replicas of fragment $F_0$ without interference. Then we obtain

$$\tau(F_0) = \mathrm{T}(F_0)\sum_{j=0}^{H-2}\eta(j)r(j)(\delta_j - 1)\prod_{k=j+1}^{H-2}\delta_k + \mathrm{O}(\eta_0). \quad (17)$$

Proof. The theorem's proof is based on theorem 3 by induction on tree height.

Given $H = 2$, the number of disks in tree *T* is equal to $\delta_0$. In accordance with theorem 3 the overhead of tuple-building any replica of $F_0$ in this case has the follow estimation:

$$\tau(F_i) = \eta(0)\cdot r(0)\mathrm{T}(F_0) + \mathrm{O}(\eta_0).$$

Consequently, the total overhead of tuple-building of all the replicas of fragment $F_0$ without interference can be estimated by:

$$\tau(F_i) = \eta(0)r(0)\mathrm{T}(F_0)(\delta_0 - 1) + \mathrm{O}(\eta_0).$$

This is confirmed by formula (17) at $H = 2$.

Let $H > 2$. Then tree *T* contains $\delta_0$ subtrees having a height $H-1$:

$$M_0, M_1, \ldots, M_{\delta_0 - 1}.$$

Let us to denote the total overhead of building of all the replicas of fragment $F_0$, allocated to the all disks of tree $M_j$ by $\tau_j(F_0)$. We have

$$\tau(F_0) = \sum_{j=0}^{\delta_0-1}\tau_j(F_0). \qquad (18)$$

Without loss of generality we may assume that $d_0 \in \mathfrak{D}(M_0)$. Since *DM*-tree *T* is symmetrical, from (8) we obtain:

$$\tau(F_0) = \tau_0(F_0) + (\delta_0 - 1)\tau_1(F_0). \qquad (19)$$

In accordance with theorem 3, for any replica $F_i$ allocated in subtree $M_1$ we have

$$\tau(F_i) = \eta(0)r(0)\mathrm{T}(F_0) + \mathrm{O}(\eta_0). \qquad (20)$$

Because of the symmetric property of *DM*-tree *T*, the total number of disks in subtree $M_1$ is equal to $\prod_{k=1}^{H-2}\delta_k$. Taking this fact into account, from (20) we obtain:

$$\tau_1(F_0) = \eta(0)r(0)\mathrm{T}(F_0)\prod_{k=1}^{H-2}\delta_k + \mathrm{O}(\eta_0). \quad (21)$$

On the other hand, by induction we have:

$$\tau_0(F_0) = \mathrm{T}(F_0)\sum_{j=1}^{H-2}\eta(j)r(j)(\delta_j - 1)\prod_{k=j+1}^{H-2}\delta_k + \mathrm{O}(\eta_0). (22)$$

Plugging into (19) the right hand side values from (21) and (22) we obtain

$$\begin{aligned}\tau(F_0) &= \mathrm{T}(F_0)\sum_{j=1}^{H-2}\eta(j)r(j)(\delta_j - 1)\prod_{k=j+1}^{H-2}\delta_k \\ &+(\delta_0 - 1)\eta(0)r(0)\mathrm{T}(F_0)\prod_{k=1}^{H-2}\delta_k + \mathrm{O}(\eta_0) \\ &= \mathrm{T}(F_0)\sum_{j=0}^{H-2}\eta(j)r(j)(\delta_j - 1)\prod_{k=j+1}^{H-2}\delta_k + \mathrm{O}(\eta_0)\end{aligned}$$

The theorem is proven.

We will define as $r(l)$ a recursive normal function of replication for values from $0 \le l \le H-2$ in the following way:

1) $r(H-2) = \dfrac{1}{\eta(H-2)(\delta_{H-2} - 1)}$ at $l = H-2$;

2) $r(l) = \dfrac{r(l+1)\eta(l+1)(\delta_{l+1} - 1)}{\eta(l)(\delta_l - 1)\delta_{l+1}}$ at $0 \le l < H-2$.

The following theorem is appropriate:

**Theorem 5.** Let *T* be a regular *DM*-tree having height $H \ge 2$. Let $\mathbb{F}$ be the set of fragments, composing the database. Let $\mathbb{R}$ be the set of all the replicas of all the fragments of $\mathbb{F}$, formed by the normal function of replication use. Let $\mathrm{T}(\mathbb{F})$ be the size of the database in tuples (Here, we assume that all tuples are of same size in bytes). Let $\tau(\mathbb{R})$ be the total overhead of tuple-building of all the replicas without interference. Then

$$\tau(\mathbb{R}) \approx k\,\mathrm{T}(\mathbb{F}),$$

Here *k* is a constant and is independent of $\mathbb{F}$.

Proof. This proof directly allows from theorem 4 and defines the normal function of replication.

The theorem shows, that using a normal function of replication, the overhead of replica renovation in a regular multiprocessor hierarchical system is proportional to the size of the renewable part of the database on the condition, that the interconnection network has sufficient bandwidth.

## 5. Conclusion

In this paper, we introduced symmetrical multiprocessor hierarchical system model. It describes a wide class of existing configurations, and is the mathematical base for the data placement strategy in multiprocessor hierarchies. A replicas building algorithm in the symmetrical hierarchy is proposed. This algorithm is based on logical fragment stripping by the equal size segments. Using this algorithm, the partial mirroring technique is developed. The technique assumes a definition of replication function. Replication function maps the hierarchy level in the replication factor which defines the replica size of the fragment. The theorems allowing to estimate the replica sizes are proven. In addition, we proved the theorems allowing to estimate replicas building overhead without interference. We proposed a form of replication function by using which we have the replicas update overhead to be a factor of the updated data volume, if the interconnect has enough bandwidth.

The main directions of future work are the following. The first is an analytical estimation of replicas update overhead including interference. The second, we plan to develop the program based on *DM*-model for simulating the hierarchical multiprocessor database systems and make experiments by this model for evaluating the data placement and load balancing strategies. The third, we suppose to implement proposed techniques and algorithms in Omega DBMS prototype [8], which designed for the cluster and Grid systems.

## References

1. Bitton D., Gray J. Disk Shadowing // Fourteenth International Conference on Very Large Data Bases, August 29 – September 1, 1988, Los Angeles, California, USA, Proceedings. Morgan Kaufmann. -1988. -P. 331-338.

2. Chen S., Towsley D.F. Performance of a Mirrored Disk in a Real-Time Transaction System // 1991 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, San Diego, California, USA, May 21-24, 1991, Proceedings. Performance Evaluation Review. -May 1991. -Vol. 19, No. 1. -P. 198-207.

3. Foster I.T., Grossman R.L. Blueprint for the future of high-performance networking: Data integration in a bandwidth-rich world // Communications of the ACM. -2003. -Vol. 46, No. 11. -P. 50-57.

4. Knuth D.E. Art of Computer Programming, Volume 1: Fundamental Algorithms (3rd Edition). Addison-Wesley, 1997. 700 c.

5. Kostenetskiy P.S., Sokolinsky L.B., Kulig A. Simulating multiprocessor database system architectures // Proceedings of Spring Young Researcher's Colloquium in Databases and Information Systems (SYRCoDIS'2005). June 1-2, 2005. Research-in-progress reports. -St.-Petersburg, Russia: VVM.com.Ltd. 2005. P. 25-27.

6. Lu H., Tan K.-L. Dynamic and Load-balanced Task-Oriented Datbase Query Processing in Parallel Systems // Advances in Database Technology - EDBT'92, 3rd Int. Conf. on Extending Database Technology, Vienna, Austria, March 23-27, 1992, Proceedings. Lect. Not. in Comp. Sc., Vol. 580. Springer. 1992. P. 357-372.

7. Mehta M., DeWitt D.J. Data Placement in Shared-Nothing Parallel Database Systems // The VLDB Journal. -January 1997. -Vol. 6, No. 1. -P. 53-72.

8. Omega DBMS Prototype Web site http://omega.susu.ru/prototype/.

9. Omiecinski E. Performance Analysis of a Load Balancing Hash-Join Algorithm for a Shared Memory Multiprocessor // 17th International Conference on Very Large Data Bases, September 3-6, 1991, Barcelona, Catalonia, Spain, Proceedings. Morgan Kaufmann. 1991. P. 375-385.

10. Sokolinsky L.B. Organization of Parallel Query Processing in Multiprocessor Database Machines with Hierarchical Architecture // Programming and Computer Software. 2001. Vol. 27. No.6. P. 297-308.

11. Sokolinsky L.B. Survey of Architectures of Parallel Database Systems // Programming and Computer Software. 2004. Vol. 30. No.6. P. 337-346.

12. Williams M.H., Zhou S. Data Placement in Parallel Database Systems // Parallel database techniques / IEEE Computer society. -1998. -P. 203-218.