◆ Research Paper

File Replication and Workload Allocation for a Locally Distributed Database

Gil sang Jang*

Abstract

In distributed databases, file replication and workload allocation are important design issues. This paper solves these two issues simultaneously. The primary objective is to minimize the system response time that consists of local processing and communication overhead on a local area network. Workload (query transactions) is assigned among any sites in proportion to the remaining file request service rate of the each server.

The problem is presented in the form of a nonlinear integer programming model. The problem is proved to be NP-complete and thus an efficient heuristic is developed by employing its special structure. To illustrate its effectiveness, it is shown that the proposed heuristic is based on the heuristic of a non-redundant allocation that was provided to be effective. The model and heuristics are likely to provide more effective distributed database designs.

1. Introduction

Recently, distributed database (DDB) technologies have advanced rapidly ·and have attracted extensive investigations. A DDB is a collection of multiple, logically interrelated databases which are geographically distributed over computing sites connected by a computer network [17].

In designing a DDB, one of the major design issues is the location of data file. This problem is referred to as file allocation problem (FAP) [4]. Solving the FAP requires complicated system parameters and networking considerations [8,15,16,19,20]. In addition, balancing workload among computing servers is important. This issue may be referred to as the workload allocation problem (WAP). Workload is defined as the totality of transactions processed in a DDB system. There are two basic principles for workload allocation: load balancing and affinity-based routing. Load balancing is to transfer transactions from heavily loaded servers to lightly loaded servers. Affinity-based routing achieves the locality of data reference in order to reduce I/O delays, lock waits, or paging overhead [2,11,18].

Most past studies of FAP or WAP investigated the two problems separately. However, FAP and WAP are interdependent because the location of data files affects the assignment of workload and vice versa. Therefore, it is more realistic to solve the two problems

* Department of Information Systems, Dongguk University

simultaneously. The integrated problem may be referred to as the file and workload allocation problem (FWAP). Lee and Park [13] originally investigated the rationale of the FWAP. They proposed a model to optimize the system response time that considers the local processing overhead only. Lee and Jang [12] developed the first model to incorporate local processing and communication overhead simultaneously for the FWAP. They proposed a heuristic for solving the FWAP with non-replicated data file.

This paper extends the previous study [12] by considering replication allocation policies to further improve system performance. The extended problem may be referred to as the replicated FWAP. Typically, redundant allocation policies are categorized into (i) all beneficial sites method and (ii) progressive allocation method. The all beneficial sites method assigns data files to any site where the benefit is greater than the cost for one additional file replication. The progressive method allocates the first copy of each file on the basis of maximum value of the benefit minus the cost. The method keeps that copy where it is, and bases the next allocation decision on the location of that first copy and the maximum value of the benefit minus the cost for the remaining sites. This procedure is continued, one allocation at a time, until the benefit no longer exceeds the cost at any of the remaining sites [21]. The all beneficial sites method has been adopted for a number of distributed database designs (e.g., [3, 21]). In contrast, the progressive method has been adopted only recently (e.g., [6]). In this paper, the progressive allocation method is adopted.

2. Design Model Formulation

In this section, we develop a design model for the replicated FWAP on a local area network (LAN). Because of the multi-access/broadcasting characteristic of LANs, the per-unit cost of inter-site communication is independent of source and destination sites. Thus, the communication overhead is assumed to be proportional to the volume of data transmitted [22, 23].

The following are the major characteristics of the design model. (i) Both communication and local processing overhead are considered simultaneously. (ii) Replicated file allocation policy is used. In the case of redundant allocation, transactions are routed to the file server that has the least response time for requested file items. (iii) The workload allocation algorithm is static. Static policies are based on the average fluctuation in a long run while dynamic policies are based on the short-term fluctuations of the system workload [2, 18].
The following notations are adopted to present the problem in the form of non-linear integer programming model.

System parameters :

N Set of sites in the system

F Set of files in the system

$i,j,k,...,n$ Site index ($i,j,k,...,n \in N$)

f,g File index ($f,g \in F$)

q_{fi} Volume of transactions for query originating at the site i for the file f per unit time

u_{fi} Volume of transactions for update originating at the site i for the file f per unit time

c^q Communication cost per transaction for query from any site to any other site

c^u Communication cost per transaction for update from any site to any other site

c^w Queuing (waiting) cost per transaction

s_{fk} Storage cost of the file f at the site k per unit time

L_f Length of the file f

S_k Storage capacity at the site k

λ Total transaction request rate to the system

r Total file service request rate to the system

r_f File request rate of the file f

μ_k File request service rate of the site k

Decision variables :

λ_k Total transaction rate assigned to the site k

X_{fk} $\begin{cases} 1, & \text{if file } f \text{ is assigned to the site } k; \\ 0, & \text{otherwise} \end{cases}$

Y_{fik} The percentage that query transactions originating at the site i for the file f is transferred to site k $(0 \leq Y_{fik} \leq 1)$.

First, we note that the total request rate for the file f is computed as $r_f = \sum_{i \in N} q_{fi} + \sum_{i \in N} u_{fi}$. In the case of a Non-redundant file allocation, the total file request rate is equal to the total transaction rate; i.e.,

$$r = \sum_{f \in F} r_f = \sum_{k \in N} \lambda_k = \lambda .$$

However, in case of the redundant allocation, the total transaction rate is not equal to the total file request rate because of the interactions between query and update transactions; i.e., $r \neq \lambda$.

Workload routing decision variable denotes the volume of file requests that should be allocated to the site k. The volume of query transactions that arrive at the site k is $\sum_{f \in F} \sum_{i \in N} u_{fi} Y_{fik}$. Similarly, the volume of update transactions that arrive at the site k is $\sum_{f \in F} \sum_{i \in N} u_{fi} X_{fk}$. The total volume of transactions that should be processed by the database server at site k is

$$\lambda_k = \sum_{f \in F} \sum_{i \in N} u_{fi} Y_{fik} + \sum_{f \in F} \sum_{i \in N} u_{fi} X_{fk} .$$

The arrival of file requests at any database site is assumed to be a Poisson input process. Then, each database site can be regarded as an M/M/1 queue. According to a well known queuing analysis [7], the average response time of the transactions that receive the file processing service at the site k is computed as

$$\frac{1}{\mu_k - \lambda_k} .$$

Therefore, the average cost of transactions queued or in service in the system is

$$c^w \frac{\lambda_k}{\mu_k - \lambda_k} .$$

The communication costs occur when a file does not exist at the site where the transaction requests the file. Thus, the communication cost by queries is $c^q \sum_{f \in F} \sum_{k \in N} q_{fk}(1 - X_{fk})$, and the communication cost by updates is $c^u \sum_{f \in F} \sum_{k \in N} u_{fk}[1 - X_{fk} \prod_{i \neq k}(1 - X_{fi})]$. Lastly, the file storage cost is $\sum_{f \in F} \sum_{k \in N} s_{fi} X_{fk}$.

Since our objective is to minimize the total cost consisting of the above four cost components, the replicated FWAP is formulated as

Minimize

$$\text{Total Cost} = c^q \sum_{f \in F} \sum_{k \in N} q_{fk}(1 - X_{fk}) + c^u \sum_{f \in F} \sum_{k \in N} u_{fk}[1 - X_{fk} \prod_{i \neq k}(1 - X_{fi})] + \sum_{f \in F} \sum_{k \in N} s_{fi} X_{fk} + c^w \frac{\lambda_k}{\mu_k - \lambda_k}$$

Subject to

$$\sum_{k \in N} X_{fk} \geq 1 , \ \forall f \in F \quad (1)$$

$$Y_{fik} \leq X_{fk} , \ \forall f \in F, \ \forall i,k \in N \quad (2)$$

$$\sum_{k \in N} Y_{fik} = 1, \ \forall f \in F, \ \forall i \in N \quad (3)$$

$$\sum_{f \in F} L_f X_{fk} \leq S_k , \ \forall k \in N \quad (4)$$

$$\lambda_k = \sum_{f \in F} \sum_{i \in N} u_{fi} Y_{fik} + \sum_{f \in F} \sum_{i \in N} u_{fi} X_{fk} , \ \forall k \in N \quad (5)$$

$$\lambda_k < \mu_k , \ \forall k \in N \quad (6)$$

$$X_{fk} \in \{0,1\}, \ \forall f \in F, \ \forall i,k \in N \quad (7)$$

$$0 \leq Y_{fik} \leq 1, \ \forall f \in F, \ \forall i, k \in N \quad (8)$$

The constraint (1) ensures that at least one copy of each file should be allocated. The constraint (2) enforces that a request from the site i can be routed to the site k only if the copy of the requested file exists at that site. The constraint (3) ensures that a query request is served locally only if the local site has the copy of the referenced file, otherwise, the request is to be routed to a site where the file is located. The constraint (4) represents the storage capacity constraints. The constraint (5) implies the total requests of site k. The constraint (6) represents the stability condition of the queuing system. The constraint (7) implies that each site contains at most one copy of each file. The constraint (8) represents that the query routing percentages take values between zero and one.

3. A Heuristic

The replicated FWAP model belongs to a class of nonlinear zero-one integer programming problems. The problem of allocating data files on a computer network is NP-complete [5]. The NP-completeness of the replicated FWAP is proved as follows.

Lemma 1: The replicated FWAP is NP-complete.

Proof: Consider the following instance of the replicated FWAP. By ignoring the local processing overhead ($c^w \frac{\lambda_k}{\mu_k - \lambda_k}$), the replicated FWAP is reduced to the following problem.

Minimize

Total Cost = $c^q \sum_{f\in F}\sum_{k\in N} q_{fk}(1-X_{fk}) + c^u \sum_{f\in F}\sum_{k\in N} u_{fk}[1-X_{fk}\prod_{i\neq k}(1-X_{fi})] + \sum_{f\in F}\sum_{k\in N} s_{fk}X_{fk}$

subject to

$$\sum_{k\in N} X_{fk} \geq 1 \ , \ \forall f \in F$$

$$\sum_{f\in F} L_f X_{fk} \leq S_k \ , \ \forall k \in N$$

$$X_{fk} \in \{0,1\}, \ \forall f \in F, \ \forall i,k \in N$$

The above problem is the file allocation problem which was proved to be NP-complete [23]. ∎

Because of its NP-completeness, solving the replicated FWAP requires a heuristic that employs the special structure of the problem. Our heuristic is based on the following four principles. Principles 1 and 2 were proposed by Lee and Jang [12].

Principle1: The file having the larger request rate is preferentially assigned. The basic premise used is that the file having the larger request rate has a more significant effect upon the system performance.

Principle2: The first file is initially allocated to the site where the largest cost saving is expected.

Principle3: The redundant copy of each data file is additionally assigned to the site where its file is not located and the largest positive cost saving is expected.

Principle4: Query transactions are routed to any of sites having the identical data files in proportion to the remaining file request service rates of the sites. This is due to load balancing. The basic premise is that load balancing can improve performance by transferring transactions from heavily loaded sites to lightly loaded sites.

To formalize the heuristic, we first stipulated that file *1* is the file with the largest request rate and file *F* with the smallest. Since file *1* had the most significant effect upon the system's performance, we first started assigning the file *1* to the site with the largest cost saving.

For non-redundant allocation, the cost saving is estimated as follows. Let be a set that contains the current indices for the files assigned to the site k. The incremental request rate $\overline{\lambda_k}$ by additionally assigning the file f to the site k is computed as

$$\overline{\lambda_k} = \sum_{g\in K_k + f} r_g \ .$$

By additionally assigning the file f to site k , the cost saving , P_{fk}, can be measured

$$P_{fk}= c^q q_{fk}+ c^u u_{fk}- s_{fk}- c^w \frac{\overline{\lambda_k}}{\mu_k - \overline{\lambda_k}} .$$

The file f is allocated to the site k that has the maximum value of P_{fk} . This process continues until all of files are assigned. This file assignment process must satisfy the following two conditions; (i) the processing capacity of the site k, μ_k , must be larger than the incremental request rate $\overline{\lambda_k}$, and (ii) the remaining storage capacity of the site k must be larger than the length of the file f to be assigned to the site k [12].

By allocating the second copy redundantly, the cost saving, P_{fk} , is estimated as follows. Let $\Delta\lambda_k$ be the temporary request rate that is decreased by assigning the second copy of the file f that is initially located to the site k, to another site j. Consider the following,

$$\Delta\lambda_k = \lambda_k^*(1) - q_{fj} - \sum_{i \neq k,j} q_{fi} ,$$

where $\lambda_k^*(1)$ is a heuristic request rate of site k by non-replicated allocation. Let $\Delta\lambda_j$ be the temporary request rate that is increased by assigning the file f that is initially located to the site k, to another site j.

$$\Delta\lambda_j = \lambda_j^*(1) + (r_j - q_{fk} - \sum_{i \neq k,j} q_{fi}) ,$$

where $\lambda_j^*(1)$ is a heuristic request rate of site j by non-replicated allocation.

Using $\Delta\lambda_k$ and $\Delta\lambda_j$, the routing percentage, Y_{fik} and Y_{fij} , are computed as follows;

$$Y_{fik} = \frac{\mu_k - \Delta\lambda_k}{(\mu_k - \Delta\lambda_k) + (\mu_j - \Delta\lambda_j)} ,$$

$$Y_{fik} = \frac{\mu_j - \Delta\lambda_j}{(\mu_k - \Delta\lambda_k) + (\mu_j - \Delta\lambda_j)} ,$$

$$Y_{fik} + Y_{fij} = 1 .$$

And, the request rates for redundant allocation, $\overline{\lambda_k}$ and $\overline{\lambda_j}$, are computed as

$$\overline{\lambda_k} = \Delta\lambda_k + \sum_{i \neq k,j} q_{fi} Y_{fik} ,$$

$$\overline{\lambda_j} = \Delta\lambda_j + \sum_{i \neq k,j} q_{fi} Y_{fij} .$$

The cost saving, P_{fj} is achieved, by replicating the file f to the site j (except for the site k which was assigned the first copy of that file) and can be measured by

$$P_{fj} = c^q q_{fj} - c^u u_{fk} - s_{fj} - c^w \frac{\overline{\lambda_j}}{\mu_j - \overline{\lambda_j}} .$$

Lastly, the redundant file assignment process must satisfy the following three conditions; (i) the processing capacity of the site j, μ_j , must be larger than the incremental request rate $\overline{\lambda_j}$, (ii) the remaining storage capacity of the site j must be larger than the length of the file f to be assigned to the site j, and (iii) P_{fj} should be larger than zero.

The cost saving by allocating the third copy redundantly is estimated as follows. Let

$\Delta\lambda_k$ be the temporary request rate which is decreased by assigning the third copy of the file f, which is located on the site k and site j, to site l.

$$\Delta\lambda_k = \lambda_k^*(2) - \sum_{i \neq k,j} q_{fi} Y_{fik} ,$$

where $\lambda_k^*(2)$ is a heuristic request rate of site k by the second copy allocation. Let $\Delta\lambda_j$ be the temporary request rate which is decreased by assigning the file f which is located on the site k and site j, to site l.

$$\Delta\lambda_j = \lambda_j^*(2) - \sum_{i \neq k,j} q_{fi} Y_{fij} ,$$

where $\lambda_j^*(2)$ is a heuristic request rate of site j by the second copy allocation. Let $\Delta\lambda_l$ be the temporary request rate which is increased by assigning the file f which is located on the site k and site j, to site l.

$$\Delta\lambda_l = \lambda_l^*(2) + (r_f - q_{fk} - q_{fj} - \sum_{i \neq k,j} q_{fi} Y_{fij}) ,$$

where $\lambda_l^*(2)$ is a heuristic request rate of site l by the second copy allocation.

Using $\Delta\lambda_k$, , $\Delta\lambda_j$ and $\Delta\lambda_l$, the routing percentage P_{fik}, P_{fij}, P_{fil}, and are computed as follows;

$$Y_{fik} = \frac{\mu_k - \Delta\lambda_k}{(\mu_k - \Delta\lambda_k) + (\mu_j - \Delta\lambda_j) + (\mu_l - \Delta\lambda_l)} ,$$

$$Y_{fij} = \frac{\mu_j - \Delta\lambda_j}{(\mu_k - \Delta\lambda_k) + (\mu_j - \Delta\lambda_j) + (\mu_l - \Delta\lambda_l)} ,$$

$$Y_{fil} = \frac{\mu_l - \Delta\lambda_l}{(\mu_k - \Delta\lambda_k) + (\mu_j - \Delta\lambda_j) + (\mu_l - \Delta\lambda_l)} ,$$

and $P_{fik} + P_{fij} + P_{fil} = 1$.

The request rates for redundant allocation, $\overline{\lambda_k}$, $\overline{\lambda_j}$, and $\overline{\lambda_l}$ are computed as

$$\overline{\lambda_k} = \Delta\lambda_k + \sum_{i \neq k,j,l} q_{fi} Y_{fik} ,$$

$$\overline{\lambda_j} = \Delta\lambda_j + \sum_{i \neq k,j,l} q_{fi} Y_{fij} ,$$

$$\overline{\lambda_l} = \Delta\lambda_l + \sum_{i \neq k,j,l} q_{fi} Y_{fil} .$$

By replicating the third copy of file f to the site l (except for the site k and site j which were assigned two copies of each file), P_{fj} can be measured by

$$P_{fl} = c^a q_{fl} - s_{fl} - c^w \frac{\overline{\lambda_l}}{\mu_l - \overline{\lambda_l}} .$$

Similarly, the above derivation may be generalized for the nth file copy. Let $\Delta\lambda_k$ be the temporary request rate which is decreased by redundantly assigning the nth copy of the file f which is located on the site k, site j, site l, ..., and site $(n-1)$, up to site n.

$$\Delta\lambda_k = \lambda_k^*(n-1) - \sum_{i \neq k,j,l,\cdots,(n-1)} q_{fi} Y_{fik} ,$$

where $\lambda_k^*(n-1)$ is a heuristic request rate of site k by the $(n-1)$ copy allocation. Let $\Delta\lambda_j$ be the temporary request rate which is decreased by redundantly assigning the file f which is located on the site k, site j, site l,..., and site $(n-1)$, to site n.

$$\Delta\lambda_j = \lambda_j^*(n-1) - \sum_{i \neq k,j,l,\ldots,(n-1)} q_{fi} Y_{fij} ,$$

where $\lambda_j^*(n-1)$ is a heuristic request rate of site j by the $(n-1)$ copy allocation. Let $\Delta\lambda_l$ be the temporary request rate which is decreased by redundantly assigning the file f which is located on the site k, site j, site l,..., and site $(n-1)$, up to site n.

$$\Delta\lambda_l = \lambda_l^*(n-1) - \sum_{i \neq k,j,l,\ldots,(n-1)} q_{fi} Y_{fil} .$$

where $\lambda_l^*(n-1)$ is a heuristic request rate of site l by the $(n-1)$ copy allocation. Finally, let $\Delta\lambda_n$ be the temporary request rate which is increased by assigning the file f which is located on the site k, site j, site l,..., and site $(n-1)$, up to site n.

$$\Delta\lambda_n = \lambda_n^*(n-1) + (r_f - q_{fk} - q_{fj} - \ldots q_{f(n-1)} - \sum_{i \neq k,j,l,\ldots,(n-1)} q_{fi} Y_{fil}) .$$

Using $\Delta\lambda_k$, $\Delta\lambda_j$, $\Delta\lambda_l$ and $\Delta\lambda_n$, the routing percentages, Y_{fik}, Y_{fij}, Y_{fil}, and Y_{fin} are computed as follows;

$$Y_{fik} = \frac{\mu_k - \Delta\lambda_k}{(\mu_k - \Delta\lambda_k) + (\mu_j - \Delta\lambda_j) + (\mu_l - \Delta\lambda_l) + \ldots + (\mu_n - \Delta\lambda_n)} ,$$

$$Y_{fij} = \frac{\mu_j - \Delta\lambda_j}{(\mu_k - \Delta\lambda_k) + (\mu_j - \Delta\lambda_j) + (\mu_l - \Delta\lambda_l) + \ldots + (\mu_n - \Delta\lambda_n)} ,$$

and $$Y_{fil} = \frac{\mu_l - \Delta\lambda_l}{(\mu_k - \Delta\lambda_k) + (\mu_j - \Delta\lambda_j) + (\mu_l - \Delta\lambda_l) + \ldots + (\mu_n - \Delta\lambda_n)} .$$

Finally, $$Y_{fin} = \frac{\mu_n - \Delta\lambda_n}{(\mu_k - \Delta\lambda_k) + (\mu_j - \Delta\lambda_j) + (\mu_l - \Delta\lambda_l) + \ldots + (\mu_n - \Delta\lambda_n)} .$$

Thus, $Y_{fik} + Y_{fij} + Y_{fil} + \ldots + Y_{fin} = 1$.

The request rates for redundant allocation, $\overline{\lambda_k}$, $\overline{\lambda_j}$, $\overline{\lambda_l}$ and $\overline{\lambda_n}$ are computed as

$$\overline{\lambda_k} = \Delta\lambda_k + \sum_{i \neq k,j,l,\ldots,n} q_{fi} Y_{fik} ,$$

$$\overline{\lambda_j} = \Delta\lambda_j + \sum_{i \neq k,j,l,\ldots,n} q_{fi} Y_{fij} ,$$

and $$\overline{\lambda_l} = \Delta\lambda_l + \sum_{i \neq k,j,l,\ldots,n} q_{fi} Y_{fil} .$$

Finally, $$\overline{\lambda_n} = \Delta\lambda_n + \sum_{i \neq k,j,l,\ldots,n} q_{fi} Y_{fin} .$$

Accordingly, P_{fn} can be measured by replicating the nth copy of file f to site n (except for the site k, site j, site l, ..., and site $(n-1)$ which were the $(n-1)$ copy of each file were assigned) . Therefore,

$$P_{fn} = c^q q_{fn} - s_{fn} - c^w \frac{\overline{\lambda_n}}{\mu_n - \overline{\lambda_n}} .$$

In summary, redundant file assignment must satisfy the following three conditions: (i) μ_n^* must be larger than the request rate $\overline{\lambda_n}$; (ii) The remaining storage capacity of site n must be larger than the length of file f to be assigned to site n; (iii) P_{fn} should be larger than zero.

The effectiveness of the heuristic for the non-redundant allocation was proved by Lee and Jang [12]. Because the heuristic for the redundant allocation is based on that of the non-redundant allocation, it is likely to generate effective solutions. Using the above exploration, a heuristic may be proposed as follows:

Step 1. Initialization step

(1) Compute total request rates of each file using query and update request rates.

(2) Sort files by the descending order according to total request rates of each file.

Step 2. Non-replication step (Initial file allocation)

(1) Compute the incremental request rates by additionally assigning each file to each site.

(2) Check the two conditions, i.e., the processing capacity and remaining storage capacity.

(3) Compute the cost saving when each file is located to each site.

(4) Allocate the first copy of each file to the site where the largest cost saving is expected.

(5) Allocate the workload to each site using the file allocation information.

Step 3. Replication step (Progressive allocation method)

(1) Compute the temporary request rates by redundantly assigning each file to the site where the copies of the file are not assigned.

(2) Compute the routing percentages using the above temporary request rates.

(3) Compute the request rates of sites which are considered with the routing percentages.

(4) Check the two conditions, i.e., the processing capacity, the remaining storage capacity.

(5) Compute the cost saving as each file is located to each site.

(6) Allocate the redundant copy of each file to the site where the largest positive cost saving is expected.

(7) Allocate the workload as the latest routing request rates.

The heuristic is summarized in the following Figure1, Figure2, and Figure3. Figure1 depicts a heuristic algorithm for non-redundant allocation. Figure2 depicts a heuristic algorithm for the second copy allocation. Finally, Figure3 shows a heuristic algorithm for the nth copy allocation

begin {* Non-redundant Algorithm *}

$r_f \leftarrow \sum_{i \in N} q_{fi} + \sum_{i \in N} u_{fi}$, $\forall f \in F$

sort $r_1 \geq r_2 \geq \cdots \geq r_F$

$R_k \leftarrow [\ ]$, $\forall k$

for $f \in F$ do

for $k \in N$ do

$\overline{\lambda_k} = \sum_{g \in R_k + [f]} r_g$

if $\mu_k > \overline{\lambda_k}$ and $S_k > L_f$

then $P_{fk} = c^q q_{fk} + c^u u_{fk} - s_{fk} - c^w \dfrac{\overline{\lambda_k}}{\mu_k - \overline{\lambda_k}}$

else $P_{fk} \leftarrow \infty$

end for {* k *}

$k^o \leftarrow [k | \max_k P_{fk}]$

$X^*_{fk^o} \leftarrow 1$ {* heuristic file allocation *}

$R_{k^o} \leftarrow R_{k^o} + [f]$

$S_{k^o} \leftarrow S_{k^o} - L_f$

end for {* f *}

$\lambda_k^* \leftarrow \sum_{f \in F} r_f X^*_{fk^o}$, $\forall k$ {* heuristic request rate allocation *}

end {* Non-redundant Algorithm *}

Figure 1: A heuristic for non-redundant allocation

begin {* The Second Copy Allocation *}

for $f \in F$ do

for j $(\neq k) \in N$ do

$\Delta\lambda_k = \lambda_k^*(1) - q_{fj} - \sum_{i \neq k,j} q_{fi}$

$\Delta\lambda_j = \lambda_j^*(1) + (r_j - q_{fk} - \sum_{i \neq k,j} q_{fi})$

$Y_{fik} = \frac{\mu_k - \Delta\lambda_k}{(\mu_k - \Delta\lambda_k) + (\mu_j - \Delta\lambda_j)}$

$Y_{fik} = \frac{\mu_j - \Delta\lambda_j}{(\mu_k - \Delta\lambda_k) + (\mu_j - \Delta\lambda_j)}$

$\overline{\lambda_k} = \Delta\lambda_k + \sum_{i \neq k,j} q_{fi} Y_{fik}$

$\overline{\lambda_j} = \Delta\lambda_j + \sum_{i \neq k,j} q_{fi} Y_{fij}$

if $\mu_j > \overline{\lambda_j}$ and $S_j > L_f$

then $P_{fj} = c^q q_{fj} - c^u u_{fk} - s_{fj} - c^w \frac{\overline{\lambda_j}}{\mu_j - \overline{\lambda_j}}$

else $P_{fj} \leftarrow \infty$

end for {* j *}

$j^o \leftarrow [j | \max_j P_{fj} > 0]$

if $j^o \in N$ then

$X^*_{fj^o} \leftarrow 1$ {* the second copy file allocation *}

$\lambda_k^*(2) \leftarrow \overline{\lambda_k}$ {* redundant request rate allocation *}

$\lambda_j^*(2) \leftarrow \overline{\lambda_j}$ {* redundant request rate allocation *}

end if

$R_{j^o} \leftarrow R_{j^o} + [f]$

$S_{j^o} \leftarrow S_{j^o} - L_f$

end for {* f *}

end {* The Second Copy Allocation *}

Figure 2 : A heuristic for the second copy allocation

begin {* The nth Copy Allocation *}

for $f \in F$ do

for n $(\neq k,j,l,\ldots,n-1) \in N$ do

$\Delta\lambda_k = \lambda_k^*(n-1) - \sum_{i \neq k,j,l,\ldots,(n-1)} q_{fi} Y_{fik}$

$$\Delta\lambda_j = \lambda_j^*(n-1) - \sum_{i\neq k,j,l,\ldots,(n-1)} q_{fi}Y_{fij}$$

$$\Delta\lambda_l = \lambda_l^*(n-1) - \sum_{i\neq k,j,l,\ldots,(n-1)} q_{fi}Y_{fil}$$

$$\cdots\cdots\cdots$$

$$\Delta\lambda_n = \lambda_n^*(n-1) + (r_f - q_{fk} - q_{fj} - \ldots q_{f(n-1)} - \sum_{i\neq k,j,l,\ldots,(n-1)} q_{fi}Y_{fil})$$

$$Y_{fik} = \frac{\mu_k - \Delta\lambda_k}{(\mu_k - \Delta\lambda_k) + (\mu_j - \Delta\lambda_j) + (\mu_l - \Delta\lambda_l) + \ldots + (\mu_n - \Delta\lambda_n)}$$

$$Y_{fij} = \frac{\mu_j - \Delta\lambda_j}{(\mu_k - \Delta\lambda_k) + (\mu_j - \Delta\lambda_j) + (\mu_l - \Delta\lambda_l) + \ldots + (\mu_n - \Delta\lambda_n)}$$

$$Y_{fil} = \frac{\mu_l - \Delta\lambda_l}{(\mu_k - \Delta\lambda_k) + (\mu_j - \Delta\lambda_j) + (\mu_l - \Delta\lambda_l) + \ldots + (\mu_n - \Delta\lambda_n)}$$

$$\cdots\cdots\cdots$$

$$Y_{fin} = \frac{\mu_n - \Delta\lambda_n}{(\mu_k - \Delta\lambda_k) + (\mu_j - \Delta\lambda_j) + (\mu_l - \Delta\lambda_l) + \ldots + (\mu_n - \Delta\lambda_n)}$$

$$\overline{\lambda_k} = \Delta\lambda_k + \sum_{i\neq k,j,l,\ldots,n} q_{fi}Y_{fik}$$

$$\overline{\lambda_j} = \Delta\lambda_j + \sum_{i\neq k,j,l,\ldots,n} q_{fi}Y_{fij}$$

$$\overline{\lambda_l} = \Delta\lambda_l + \sum_{i\neq k,j,l,\ldots,n} q_{fi}Y_{fil}$$

$$\cdots\cdots\cdots$$

$$\overline{\lambda_n} = \Delta\lambda_n + \sum_{i\neq k,j,l,\ldots,n} q_{fi}Y_{fin}$$

if $\mu_n > \overline{\lambda_n}$ and $S_n > L_f$

then $P_{fn} = c^q q_{fn} - s_{fn} - c^w \dfrac{\overline{\lambda_n}}{\mu_n - \overline{\lambda_n}}$

else $P_{fn} \leftarrow \infty$

end for {* n *}

$n^o \leftarrow [n \mid \max_n P_{fn} > 0]$

if $n^o \in N$ then

$X^*_{fn^o} \leftarrow 1$

$\lambda_k^*(n) \leftarrow \overline{\lambda_k}$ {* redundant request rate allocation *}

$\lambda_j^*(n) \leftarrow \overline{\lambda_j}$ {* redundant request rate allocation *}

$\lambda_l^*(n) \leftarrow \overline{\lambda_l}$ {* redundant request rate allocation *}

$\cdots\cdots\cdots$

$\lambda_n^*(n) \leftarrow \overline{\lambda_n}$ {* redundant request rate allocation *}

end if

$R_{n^o} \leftarrow R_{n^o} + [f]$

$S_{n^o} \leftarrow S_{n^o} - L_f$

end for {* f *}

end {* The nth Copy Allocation *} **Figure 3 : A heuristic for the nth copy allocation**

4. Illustrative Example

In order to illustrate the applicability of the heuristic, a distributed database is considered over the bus LAN system that consists of three sites. The processing rates of each site are 300, 200, and 150, respectively. Request rates of each data file are given in Table1. The request rates of each file are 70, 40, 30, 30, and 20, respectively. The initial total request rate is 190 ($\lambda = r = 190$). The storage capacity of sites and the length of files are given in Table2 and Table3.

Table 1 : Request rates of query, and update

File / Request / Site	file 1			file 2			file 3			file 4			file 5		
	s1	s2	s3	s1	s2	s3	s1	s2	s3	s1	s2	s3	s1	s2	s3
q_{fk}	20	20	14	7	5	12	6	5	7	10	3	6	1	8	3
u_{fk}	6	6	4	5	4	7	1	2	9	5	4	2	2	4	2

Table 2 : Storage capacities of sites (Mbytes)

site 1	site 2	site 3
50	40	35

Table 3 : Lengths of files (Mbytes)

file 1	file 2	file 3	file 4	file 5
12	10	8	8	6

The per-unit query and update costs for communication, queuing delay cost per transaction, and the storage costs of files are given in Table4 and Table5.

Table 4 : Query, update, and queuing delay costs ($)

query cost(c^q)	update cost(c^u)	queuing delay cost(c^w)
1	1	1

Table 5 : Storage costs of files ($)

file1 (s_{1k})	file2 (s_{2k})	file3 (s_{3k})	file4 (s_{4k})	file5 (s_{5k})
4	3	2	2	1

Non-redundant Allocation :

For simplicity, files have already been sorted so that file1 has the largest rate. First, for file1, the cost savings are computed as follows:

$$P_{11} = 1 \times 20 + 1 \times 6 - 4 - 1 \times \frac{60}{300-60} = 21.75;$$

$$P_{12} = 1 \times 20 + 1 \times 6 - 4 - 1 \times \frac{60}{200-60} = 21.57;$$

and $$P_{13} = 1 \times 11 + 1 \times 7 - 4 - 1 \times \frac{60}{150-60} = 13.33.$$

S_1=50-12=38. Thus, file1 is assigned to the site1 ($X_{11}^* = 1$).

For file2,

$$P_{21} = 1 \times 7 + 1 \times 5 - 3 - 1 \times \frac{60+40}{150-(60+40)} = 8.50;$$

$$P_{22} = 1 \times 5 + 1 \times 4 - 3 - 1 \times \frac{40}{200-40} = 5.75;$$

and $$P_{23} = 1 \times 12 + 1 \times 7 - 3 - 1 \times \frac{40}{150-40} = 15.64.$$

S_3 = 20-10 = 10. Hence, $X_{23}^* = 1$. Similarly, $X_{33}^* = 1$, $X_{41}^* = 1$, and $X_{52}^* = 1$. Therefore, the corresponding workload allocation decisions are $\lambda_1^* = 100$, $\lambda_2^* = 20$, and $\lambda_3^* = 70$. The total cost according to this file assignment is \$115.5. The above computational procedure is summarized in Table6.

Table 6: Summary of non-redundant file allocation

Files	Rates	Cost Savings			File Allocation
f	r_f	P_{f1}	P_{f2}	P_{f3}	k^o
1	60	21.75*	21.57	13.33	1
2	40	8.50	5.75	15.64*	3
3	30	4.57	5.82	13.10*	3
4	30	12.57*	4.82	3.94	1
5	20	1.42	10.89*	2.46	2

The Second Copy Allocation :

For file1, when file1 is assigned to site2, the computing procedures of cost saving, Y_{12}, are as follows:

$$\Delta\lambda_1 = 100 - 20 - 14 = 66,$$

$$\Delta\lambda_2 = 20 + (70 - 20 - 14) = 56,$$

$$Y_{131} = \frac{300-66}{(300-66)-(200-56)} = 0.6190,$$

$$Y_{132} = \frac{200-56}{(300-66)-(200-56)} = 0.3810,$$

$$\overline{\lambda_1} = 66 + 14 \times 0.6190 = 75,$$

$$\overline{\lambda_2} = 56 + 14 \times 0.3810 = 61,$$

and $$Y_{12} = 2 \times 10 - 1 \times 6 - 14 - 1 \times \frac{61}{200-61} = 9.56.$$

When file1 is assigned to site3, the computing procedures of cost saving, Y_{13}, are as follows:

$\Delta\lambda_1 = 100 - 14 - 20 = 66,$

$\Delta\lambda_3 = 70 + (70 - 20 - 20) = 100,$

$Y_{121} = \frac{300-66}{(300-66)-(150-100)} = 0.8239,$

$Y_{123} = \frac{150-100}{(300-66)-(150-100)} = 0.1761,$

$\overline{\lambda_1} = 66 + 20 \times 0.8239 = 82,$

$\overline{\lambda_3} = 100 + 20 \times 0.1761 = 104,$

and $Y_{13} = 114 - 1\times6 - 14 - 1\times\frac{104}{150-104} = 1.74.$

When file1 is redundantly allocated to the site2, the remaining storage capacity is 5 Mbytes (S_3= 36-12 = 24).

For file2, when file2 is assigned to the site1, the computing procedures of cost saving, Y_{21}, are as follows:

$\Delta\lambda_3 = 70 - 7 - 5 = 58,$

$\Delta\lambda_1 = 75 + (40 - 12 - 5) = 98,$

$Y_{223} = \frac{150-58}{(150-58)-(300-98)} = 0.3129,$

$Y_{221} = \frac{300-98}{(150-58)-(300-98)} = 0.6871,$

$\overline{\lambda_3} = 58 + 5 \times 0.3129 = 60,$

$\overline{\lambda_1} = 98 + 5 \times 0.6871 = 101,$

and $Y_{21} = 1\times7 - 1\times7 - 1\times3 - 1\times\frac{101}{300-101} = -3.51$

When file2 is assigned to site2, the computing procedures of cost saving, Y_{22}, are as follows:

$\Delta\lambda_3 = 70 - 5 - 7 = 58,$

$\Delta\lambda_2 = 61 + (40 - 12 - 7) = 82,$

$Y_{213} = \frac{150-58}{(150-58)-(200-82)} = 0.4381,$

$Y_{212} = \frac{200-82}{(150-58)-(200-82)} = 0.5619,$

$\overline{\lambda_3} = 58 + 7 \times 0.4381 = 61,$

$\overline{\lambda_2} = 82 + 7 \times 0.5619 = 86,$

and $Y_{22} = 1\times5 - 1\times7 - 1\times3 - 1\times\frac{86}{200-86} = -5.75.$

File2 is not allocated to any site because of the negative cost savings.

For the remaining files (i.e., file3, file4, and file5), the cost savings are calculated by using the same procedures. As a result, the remaining files are not assigned to any site because of the negative cost savings or the storage capacity constraints. The above computational procedure is summarized in Table7.

Table 7: Summary of the second copy allocation

Files	Rates	Cost Savings			File Allocation
f	r_f	P_{f1}	P_{f2}	P_{f3}	j^o
1	60	*	9.56	1.74	2
2	40	-3.51	-5.75	*	-
3	30	-5.47	-6.68	*	-
4	30	*	-4.63	-2.42	-
5	20	-4.40	*	-3.34	-

Hence, as the result of the second copy allocation, file1 is redundantly allocated to site1 and site2. File2 and file3 are assigned to site3. File4 is assigned to the site1. The file5 is only located at site2. Therefore, the corresponding workload allocation decisions are $\lambda_1^* = 75$, $\lambda_2^* = 61$, and $\lambda_3^* = 70$. The total cost, according to the redundant file assignment, is \$105.7.

The Third Copy Allocation :

For file1, when file1 is assigned to the site3 (because of its location at site1 and site2), the computing procedures of cost saving, P_{13}, are as follows:

$\overline{\lambda_1} = 75 - 14 \times 0.6190 = 66,$

$\overline{\lambda_2} = 61 - 14 \times 0.3810 = 56,$

$\overline{\lambda_3} = 70 + (70 - 20 - 20) = 100,$

and $P_{13} = 1 \times 14 - 1 \times 4 - 1 \times \frac{100}{150 - 100} = 8.00.$

The remaining storage capacity is 5 Mbytes ($S_3 = 17 - 12 = 5$).

For file2, when file2 is assigned to the site1, the computing procedures of cost saving, P_{21}, are as follows:

$\Delta \lambda_3 = 100 - 7 - 5 = 88,$

$\Delta \lambda_1 = 66 + (40 - 12 - 5) = 89,$

$Y_{223} = \frac{150 - 88}{(150 - 88) - (300 - 89)} = 0.2271,$

$Y_{221} = \frac{300 - 89}{(150 - 88) - (300 - 89)} = 0.7729,$

$\overline{\lambda_3} = 88 + 5 \times 0.2271 = 89,$

$\overline{\lambda_1} = 89 + 5 \times 0.7729 = 93,$

and $P_{21} = 1 \times 7 - 1 \times 7 - 1 \times 3 - 1 \times = -3.35$

When file2 is assigned to site2, the computing procedures of cost saving, P_{22}, are as follows:

$\Delta \lambda_3 = 100 - 5 - 7 = 88,$

$\Delta \lambda_2 = 56 + (40 - 12 - 7) = 77,$

$$Y_{213} = \frac{150-88}{(150-88)-(200-77)} = 0.3351,$$

$$Y_{212} = \frac{200-77}{(150-88)-(200-77)} = 0.6649,$$

$\overline{\lambda_3} = 88 + 7 \times 0.3351 = 90,$

$\overline{\lambda_2} = 77 + 7 \times 0.6649 = 82,$

and $P_{22} = 1\times5 - 1\times7 - 1\times3 - 1\times = -5.69.$

File2 is not allocated to any site because of the negative cost savings.

The remaining files (i.e., file3, file4 and file5) are not assigned to any site for the same reason. The above computational procedure may be summarized in Table8.

Table 8: Summary of the third copy allocation

Files	Rates	Cost Savings			File Allocation
f	r_f	P_{f1}	P_{f2}	P_{f3}	l^o
1	60	*	8.00	*	2
2	40	-3.45	-5.69	*	-
3	30	-5.42	-6.63	*	-
4	30	*	-4.56	$-\infty$	-
5	20	-4.35	*	$-\infty$	-

Hence, as the result of the third copy allocation, file1 is redundantly allocated to site1, site2, and site3. File2 and file3 are assigned to site3. File4 is assigned to site1. File5 is located at the site2 only. Therefore, the corresponding workload allocation decisions are $\lambda_1^* = 66$, $\lambda_2^* = 56$, and $\lambda_3^* = 100$. The total cost, according to the redundant file assignment, is \$96.4. The above example shows that total costs are decreased by file replication (\$115.5 →\$105→\$96.4).

5. Computational Results

To analyze the effects of file replication , we randomly generated multiple problem instances of various sizes. For $|M| = 10$, 30, and 50, 2 instances for each of the parameter $|F| = 5$ and 10 were generated. For $|M| = 100$, $|F| = 5$, 10, 15, 20, and 30 were generated. Also, we randomly generated 10 instances for different number of files. For each problem, the coefficient q_{fk}, u_{fk}, L_f, s_{fk}, μ_k and S_k were uniformly determined as follows:

$q_{fk} = 40 * U(0, 1) + 100$

$u_{fk} = 19 * U(0, 1) + 1$

$L_f = 10 * U(0, 1) + 6$

$s_{fk} = L_f / 4$

$\mu_k = 70 * U(0,1) + * 40$

$S_k = 40 * U(0,1) + 30$

Since the complexity of our problem is 2^{N*F}, it is NP-Complete. So, it is impossible to obtain optimal solutions in the reasonable time when problem size increase. Nevertheless, we proposed an effective and efficient heuristic with the complexity of $1.5*N*F$. Since our heuristic is a polynomial algorithm, the heuristic solution can be found in a reasonable amount of time. By the property 2, our heuristic is likely to generate good database designs. The results of the experiment is shown in Table9. In Table9, Normalized values mean the normalized average value of each cost on the basis of the cost of nonredundant allocation. Accordingly, all normalized values in the case of nonredundant allocation are 100, and normalized values in the case of replication are cost of replication / cost of nonreplication * 100. Let $FWAP_n$ be the normalized objective value of the FWAP from the nonreplication allocation and $FWAP_r$ be the normalized objective value from the replication allocation. Then the Improvement (%) is defined by

$$\text{Improvement (\%)} = 1 - \frac{FWAP_r}{FWAP_n}$$

Improvement (%) is the percentage of improvement of replication from nonreplication.

Table 9 : Experimental Results

Number of sites	Number of files	Normalized cost for nonreplication	Normalized cost for nonreplication	Improvement(%)
10	5	100	60	40
	10	100	83	17
30	5	100	68	32
	10	100	77	23
50	5	100	84	16
	10	100	82	18
100	5	100	86	14
	10	100	91	9
	15	100	91	9
	20	100	92	8
	30	100	94	6

Hence, if the constraints (4) and (6) are relaxed by increasing the capacities of μ_k and S_k, the effects of replication are increased. The effects of replication are shown in Figure 4. The figure4 is the case of $|N|$ = 100 and $|F|$ = 5, 10, 15, 20, and 30. This shows that as the number of replication is increased, the costs are gradually decreased until profits is generated by file replications.

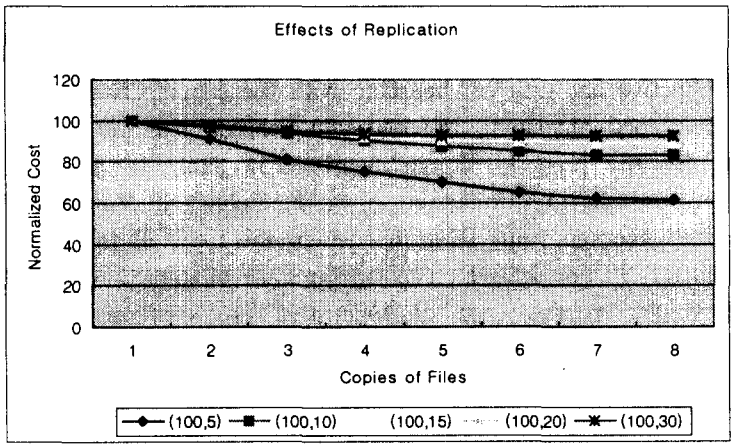

Figure 4 : Effects of Replication

6. Concluding Remarks

Our approach is realistic because the proposed model solves file replication and workload balancing simultaneously in the distributed database design process. Furthermore, our model is the first to consider both local processing and communication overheads. Heuristic is developed to solve the resulting NP-complete problem. The heuristic is likely to provide an effective solution because it is based on the heuristic of a non-redundant allocation which was proved to be effective [12]. Hence, the heuristic can be used as a stepping stone that can help us develop solution procedure for solving similar complex problems. A numerical example is solved to illustrate the suitability of our design approach.

A future research avenue is to expend the model to the case of two-level LAN or corporate network [14, 22]. Other interesting future research areas may include the incorporation of database joins into the model [1] or the determination of processing capacity [10]. We are currently working on some of these problems, and will report any significant results in the future.

Acknowledgement

This study was supported by the research grant for the junior faculties of Dongguk University.

References

[1] Ahn, J.K., and Moon, S.C., 1991, Optimizing joins between two fragmented relations on a broadcast local network, *Information Systems*, 16, 185-198.

[2] Chen, P.P.S., 1973, Optimal partitioning of input load to parallel exponential servers, *Proc. Fifth Southeastern Symp. System Theory*, 66-69.

[3] Ceri, S., Pernica, B., and Wiederhold, B., 1987, Distributed database design methodologies, *Proceeding of the IEEE*, vol. 75, no. 5, pp. 533-546.

[4] Dowdy, L.W., and Foster, D.V., 1982, Comparative models of the file assignment problem, *ACM Computing Surveys*, 14, 287-313.

[5] Eswaran, K.P., 1974, Placement of records in a file and file allocation in a computer network, *Information and Processing '74*, 304-307.

[6] Janakiraman, J., Warack, C., Bhal, G., and Teorey, T.J., 1991, Progressive fragment allocation, *Proc. 10th Int'l Conf. on the entity relationship approach, San Mateo, CA*, October 23-25, pp.543-560.

[7] Kleinrock, L., 1976, *Queuing Systems* (John Wiley and Sons, Inc.).

[8] Kumar, A., Pathak, R.M., and Gupta, Y.P., 1995, Genetic algorithm based approach for file allocation on distributed systems, *Computers Ops Res*, 22, 41-54.

[9] Lee, H., 1993, Modeling and optimization of data assignment in a distributed information system, *Systems Science*, 24, 173-181.

[10] Lee, H., 1994, Simultaneous determination of capacities and load in parallel M/M/1 queues, *European Journal of Operational Research*, 73, 95-102.

[11] Lee, H., 1995, Optimal static distribution of prioritized customers to heterogeneous parallel servers, *Computers Ops Res*, 22, 995-1003.

[12] Lee, H., and Jang, G., 1996, File and workload allocation on a local multi-access computer network: incorporating local processing and communication overhead, *Systems Science*, 27, 831-837.

[13] Lee, H., and Park, T., 1995, Allocating data and workload among multiple servers in a local area network, *Information Systems*, 20, 261-269.

[14] Lien, Y.N., Chang, Y.L., and Wah, B.W., 1992, File allocation on homogeneous local computer systems with two-level multi-access networks, *Working paper* (Department of Electrical and Computer Engineering, University of Illinois).

[15] Liu Sheng, O.R., and Lee, H., 1992, Data allocation design in computer networks: LAN versus MAN versus WAN, *Annals of Operations Research*, 36, 125-150.

[16] March, S.T., and Rho, S., 1995, Allocating data and operations to nodes in distributed database design, *IEEE Transactions on Knowledge and Data Engineering*, 7, 305-317.

[17] Ozsu, M.T., and Valdureiz, P., 1991, *Principles of Distributed Database Systems*, (Prentice Hall).

[18] Rahm, E., 1992, A framework for workload allocation in distributed transaction processing systems, *Journal of Systems and Software*, 18, 171-190.

[19] Rho, S., and March, S.T., 1994, A nested genetic algorithm for distributed database design, *Proceedings of the Twenty-Seventh Annual Hawaii International Conference on System Sciences*, 33-42.

[20] Stallings, W., 1993, *Local and Metropolitan Area Networks* (Macmillan).

[21] Teorey, T.J., 1989, Distributed database design: a practical approach and example, *SIGMOD RECORD*, vol. 18, no. 4, pp.23-39.

[22] Wah, B.W., Chang, Y.L., and Lien, Y.N., 1988, File allocation problems on homogeneous two-level local broadcast network, *Proc. Fourth Int'l Conf. on Data Eng.* 92-99.

[23] Wah, B.W., and Lien, Y.N., 1985, Design of distributed database on local computer systems with a multi-access network, *IEEE Trans. on Software Eng.* SE-11, 606-619.