ELSEVIER

Information and Software Technology 42 (2000) 171–184

INFORMATION AND SOFTWARE TECHNOLOGY

www.elsevier.nl/locate/infsof

Designing a distributed database on a local area network: a methodology and decision support system

H. Lee[a,*], Y.-K. Park[b], G. Jang[c], S.-Y. Huh[d]

[a]*Corporate Information System Laboratory, Graduate School of Management, KAIST, 207-43, Cheongyangri-Dong, Dongdaemoon-Gu, Seoul 130-012, South Korea*

[b]*LG LCD Ltd. LG Young-Dong Bldg. 891, Daechi-dong, Kangnam-gu, Seoul 135-280, South Korea*

[c]*Oracle Korea Ltd. Jeong Woo Bldg 3F. 1319-15, Dal-dong, Nam-Gu, Ulsan 680-030, South Korea*

[d]*Database Application Laboratory, Graduate School of Management, KAIST, 207-43, Cheongyangri-Dong, Dongdaemoon-Gu, Seoul 130-012, South Korea*

Received 23 March 1998; received in revised form 4 June 1999; accepted 23 June 1999

Abstract

Local area networks (LANs) are important for an enterprise to hold a competitive edge. Many companies have therefore converted terminal-based computing systems to LAN-based distributed data processing systems. This paper proposes a design methodology for distributed databases connected by a LAN. Two primary objectives of the methodology are: (i) to allocate data files and workload among heterogeneous servers; and (ii) to determine the number of servers to satisfy the response time required for processing each transaction. The file and workload allocation decision is formulated as a nonlinear zero–one integer programming problem. This problem is proven to be NP-complete. A heuristic is developed to solve this problem effectively. A decision support system is implemented and an example is solved to illustrate the practical usefulness of the system. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Distributed database; Decision support system; Local area network; File and workload allocation; Nonlinear integer programming; Local processing overhead; Server service capacity

1. Introduction

A local area network (LAN) is an infrastructure which places the company on a secure footing in today's highly competitive business environment. A LAN is a communication network that is confined to a small area, such as a single building or a small cluster of buildings [14,15]. Nowadays most organizations have established LANs and used them in various areas of work. Indeed, LANs play an important role in information sharing and office works in business, academia, and industry. Attached to the LANs would be personal computers, database servers providing various data, and communication servers for interacting with remote systems and downloading data to the local databases [1]. Fig. 1 shows the structure of distributed system on a LAN. An intermediate system (IS) is a device used to connect two subnetworks and permit communication between end systems attached to different subnetworks [14]. Transactions are generated from terminals and routed to DB server having required data by communication server.

In a LAN establishment, the distributed database design and the process capacity design are very important issues. A distributed database is a collection of multiple, logically interrelated databases distributed over a computer network [12]. In a distributed database design, two fundamental allocation design issues are: (i) file allocation, the optimum distribution of fragments; and (ii) workload allocation, the optimum distribution of transactions to be processed in a LAN [11]. Fragments, data files, are database partitions generated from fragmentation design. Allocation strategies affect: (i) the total operating cost which consists of communication overhead, storage cost and local processing cost; and (ii) the response time, which consists of network delay, communication delay, local delay (I/O time, CPU processing time) and local processing delay. Allocation of data files and workload across the nodes of a computer network has been studied extensively. This problem has been proven to be NP-complete [8,9,11].

Wah and Lien [18] studied the file allocation problem (FAP) in a distributed databases design on a LAN. They considered only communication overhead in a homogeneous local broadcast network. In their model, redundant data file allocation policy was adopted, but join problem was not considered. Their model was formulated in the

* Corresponding author. Tel.: + 82-2958-32146; fax: + 82-2958-3220.

0950-5849/00/$ - see front matter © 2000 Elsevier Science B.V. All rights reserved.
PII: S0950-5849(99)00056-7

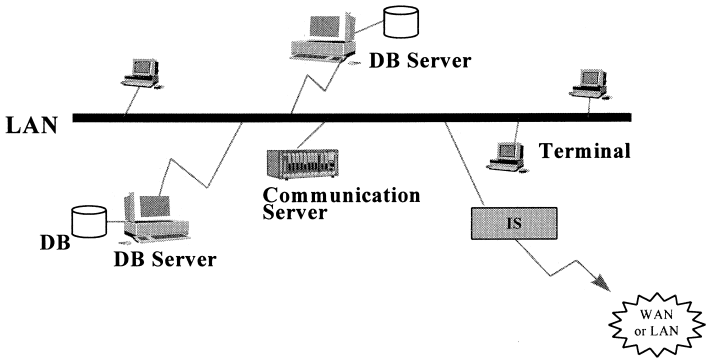

Fig. 1. Structure of distributed system on a LAN.

form of nonlinear zero–one integer programming, and a heuristic for solving the problem was developed. In Ref. [17], the FAP on homogeneous two-level local broadcast network was studied. Yu et al. [19] studied a FAP on a LAN with star topology. They considered only communication overhead and permitted redundant file allocation. They formulated a model as nonlinear zero–one integer programming, and developed an adaptive algorithm to obtain good solutions.

Chen [3,4] originally investigated workload allocation problems (WAPs). He studied the problem that partitions a Poisson input stream of requests into substreams to each device so that the mean system response time is minimized. The model is formulated as a nonlinear programming problem and a closed-form algorithm is developed for the optimal policy. Tantawi and Towsley [16] incorporated communication overhead into the objective function of the mean system response time. The model is developed within the framework of the product-form queuing networks.

Most previous allocation works studied FAP or WAP separately. However, these two problems are interdependent because file allocation has an influence on the workload allocation and vice versa. Thus, it is more realistic to solve these two problems simultaneously.

Lee and Park [11] proposed a rationale for solving the two allocation problems simultaneously and presented an analytical model that would attain effective allocation polices. They considered only local processing in a LAN and ignored communication overhead. Their model was presented in the form of nonlinear zero–one integer programming and developed a heuristic to obtain a good solution. In Lee and Jang's model [8], both local processing and communication overhead were considered. The model adopted the nonredundant file allocation policy. It was formulated as a nonlinear zero–one integer programming, and an effective heuristic for obtaining a good solution was developed. In [9], they extended the model to incorporate redundant file allocation. For a corporate network environment, readers are referred to [10].

In a LAN establishment, it is also important to determine the adequate processing capacity, i.e. the number of servers to satisfy the required response time. Berman and Nigam [1] investigated the determination of the processing capacity for satisfying the required response time of each transaction. They considered only local processing and classified transactions into query, update, and join, but used a full enumeration method to determine the placement of data files.

This paper proposes a methodology for determining file and workload allocation simultaneously on a LAN. This problem will be referred to as "file and workload allocation problem" (FWAP). In addition, the methodology determines the number of processing servers to satisfy the response time. Furthermore, a decision support system (DSS) is implemented to demonstrate the practical usefulness of the methodology.

2. Design methodology

2.1. Methodology architecture

Our methodology consists of three subsystems: a file and

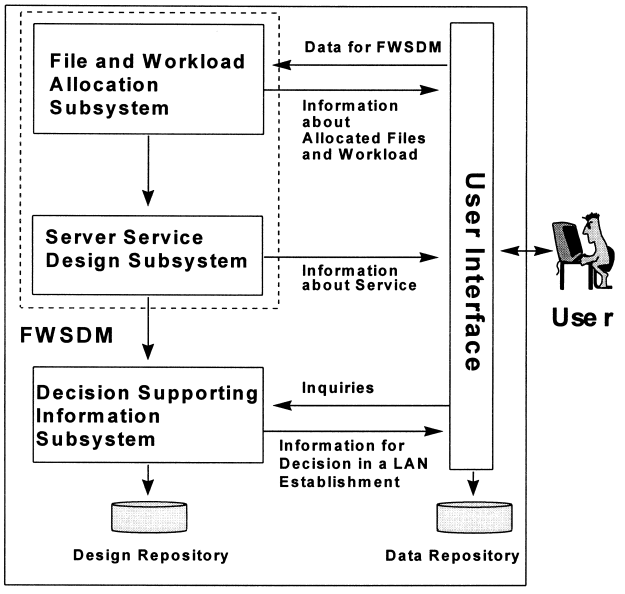

Fig. 2. Methodology architecture.

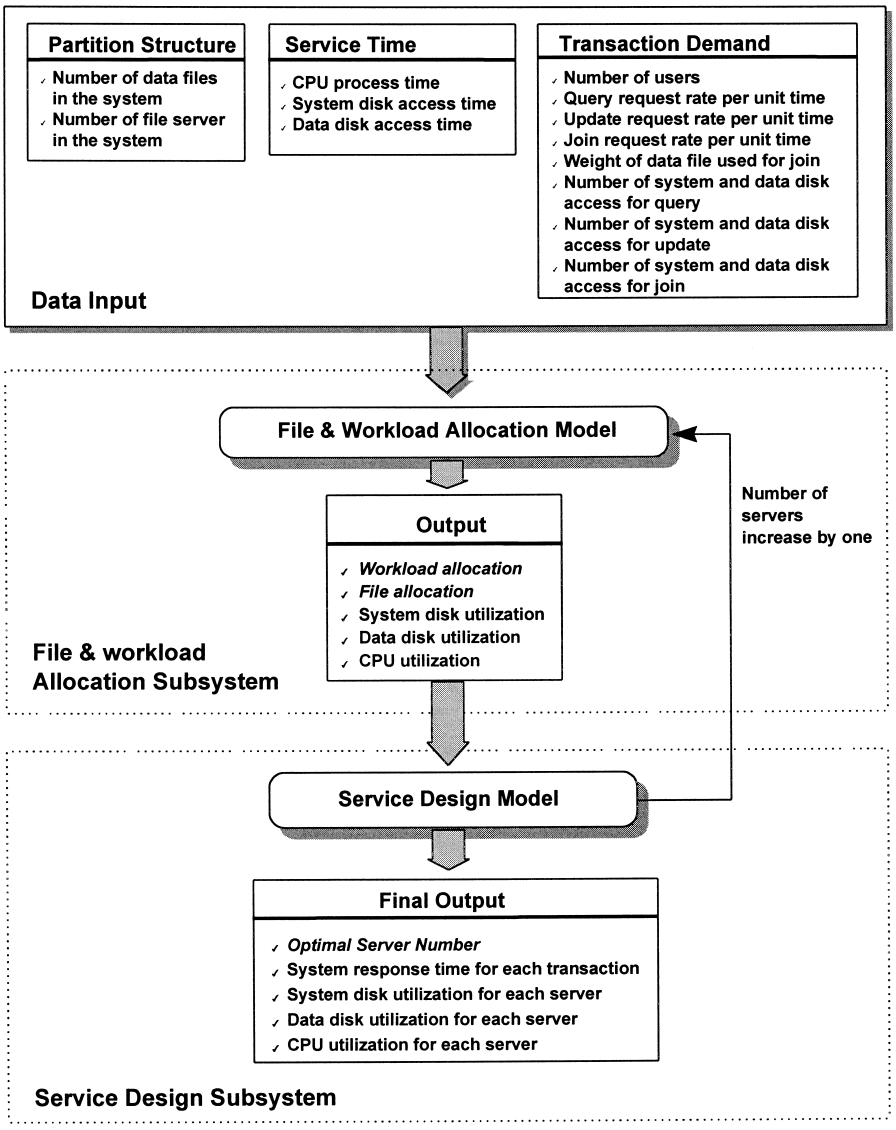

Fig. 3. FWSDM details.

workload allocation subsystem, a server service design subsystem and a decision supporting information subsystem. Fig. 2 shows its architecture.

FWSDM (file, workload and service design model) solves FWAP and determines the optimal number of servers. File and workload allocation subsystem receives data from users about files and workload (to be allocated) and available servers, and then allocates files and workload to servers according to our heuristic. The results of allocation are passed over to the server service design subsystem. Server service design subsystem determines the optimal number of servers. Decision supporting information subsystem provides users with useful information such as the placement of data files, service capacity, server utilization, and what-if analysis information. The architecture requires two repositories, i.e. input data is stored in the data repository and design results are stored in the design repository. Fig. 3 depicts details of FWSDM.

2.2. Development of design model

In this paper, FWAP is modeled to minimize the response time for processing transactions. First, we assume that databases were already fragmented according to an affinity-based fragmentation policy. This policy partitions a set of data with the same properties, i.e. access frequencies [2]. Thus, transactions with the same properties are routed to the same file server, and locality of data reference is achieved. A nonredundant file allocation policy is adopted. The workload allocation algorithm is static. Static policy is based on the average fluctuation in a long run, while dynamic policy is based on the short-term fluctuation of the system workload [13].

Transactions from terminals are routed to the server having the requested data. When another transaction is processing, the transaction is placed in queue and processed as the central processing unit (CPU) becomes available

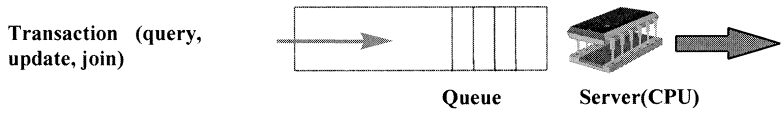

Fig. 4. A queuing model.

according to its order of arrival. Therefore, each server can be considered as a queuing system (Fig. 4). This queuing system is assumed to be an M/M/1 model [6]. That is, (i) a transaction arrives according to the Poisson process, (ii) service time for a transaction has exponential distribution and (iii) service policy is First In First Out (FIFO) [22]. In general, a server has two disk drives, one a system disk for software (for example, database management system (DBMS), operating system, etc.), the other a data disk. Each disk access is preceded by the execution of a sequence of instructions by the CPU to process the I/O [1]. Therefore, each transaction is regarded as a sequence of pairs of CPU and disk accesses. Fig. 5 shows this structure.

Under these assumptions, hardware and software parameters are needed for calculating the system response time. For hardware parameters, CPU processing time is computed as (instructions per access) × (process time per instruction). The disk access time is the sum of the times needed to complete four distinct activities: (a) seek; (b) controller path link; (c) latency; and (d) data transfer. The hardware system vendor will provide these data. For software parameters, there are three types of data: the number of system disk accesses, data disk accesses and CPU accesses for processing each transaction. This information is estimated by the analysis of past data, or DBMS [1].

2.2.1. Notation

The following notations are used in the model to present the problem in the form of nonlinear integer programming.

System parameters

(i) Basic parameters

N_s	Number of file servers in the system
N_f	Number of data files in the system
N_u	Number of users in the system
F_i	Data file i
S_k	File server k
λ	Total transaction request rate to the system
F	Total file service request rate to the system

(ii) Transaction rates

q_i	Query accessing data file I (F_i)
u_i	Update accessing data file i (F_i)
J_S	Join requiring data file set S

(iii) Request rates

f_{q_i}	Query request rate for data file i (F_i) per unit time
f_{u_i}	Update request rate for data file i (F_i) per unit time
f_{J_S}	Join request rate for data file set S per unit time
$w^i_{J_S}$	Weight of data file F_i which is used for join J_S

(iv) Hardware-related parameters

μ_k^S	System disk access time of server k
μ_k^D	Data disk access time of server k
μ_k^C	CPU process time of server k

(v) Disk access rates

S_{q_i}	Number of system disk access for query q_i
S_{u_i}	Number of system disk access for update u_i
S_{J_S}	Number of system disk access for join J_S
D_{q_i}	Number of data disk access for query q_i
D_{u_i}	Number of data disk access for update u_i
D_{J_S}	Number of data disk access for join J_S

(vi) Utilization

ρ_k^S	System disk utilization of sever k
ρ_k^D	Data disk utilization of sever k
ρ_k^C	CPU utilization of sever k
ρ_k	System utilization of sever k
ρ	System utilization

(vii) Response time

$R_k^{q_i}$	Response time for query i of server k
$R_k^{u_i}$	Response time for update i of server k

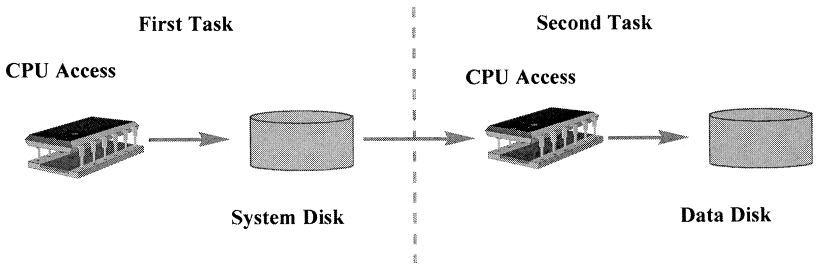

Fig. 5. Transaction processing structure.

Begin { heuristic }

Sort $f_1 \geq f_2 \geq \cdots \geq f_{N_f}$

$E_k \leftarrow \{\}, \forall k$

for i = 1, 2, ... , N_f do

for k = 1, 2, ... , N_s do

$$U_k^S \leftarrow \mu_k^S \sum_{j \in E_k + \{i\}} \{(f_{q_i} S_{q_i} N_u + f_{u_i} S_{u_i} + w_{J_s}^i f_{J_s} S_{J_s} N_u)\}$$

$$U_k^D \leftarrow \mu_k^D \sum_{j \in E_k + \{i\}} \{(f_{q_i} D_{q_i} N_u + f_{u_i} D_{u_i} + w_{J_s}^i f_{J_s} D_{J_s} N_u)\}$$

$$U_k^C \leftarrow \mu_k^C \sum_{j \in E_k + \{i\}} [\{f_{q_i}(S_{q_i} + D_{q_i}) N_u + f_{u_i}(S_{q_i} + D_{u_i}) + w_{J_s}^i f_{J_s}(S_{J_s} + D_{J_s}) N_u\}]$$

$U_k \leftarrow U_k^S + U_k^D + U_k^C$

if $U_k < 1$ then $U_k \leftarrow U_k$ else $U_k \leftarrow \infty$

end for

$k^* \leftarrow \{k | \min_k U_k\}$

$X_{ik^*}^* \leftarrow 1$ {adequate file allocation}

$E_{k^*} \leftarrow E_{k^*} + \{i\}$

end for

$$\lambda_k^* \leftarrow \sum_{i=1}^{N_f} (f_{q_i} N_u + w_{J_s}^i f_{J_s} N_u + f_{u_i}) X_{ik}^* \quad , \forall k \text{ {adequate workload allocation}}$$

end { heuristic }

Fig. 6. A heuristic for the file and workload allocation.

$R_k^{J_S}$ Response time for join J_S of server k
R_s System response time

Decision variables

λ_k is the total transaction rate assigned to S_k (i.e. workload allocation)
X_{ik} is the indicator variable of assignment of F_i to S_k (i.e. file allocation) is given by

$$X_{ik} = \begin{cases} 1, & \text{if } F_i \text{ is assigned to } S_k \\ 0, & \text{otherwise} \end{cases}.$$

2.2.2. Design model

It is proposed to allocate data file to adequate server in such a way as to minimize system response time. We assume that the queuing system satisfies the conditions for M/M/1. In the M/M/1 queuing system, the utilization of server k is calculated as follows.

For the system disk utilization, the access rate per unit time is

$$\sum_{i=1}^{N_f} \{(f_{q_i} S_{q_i} N_u + f_{u_i} S_{u_i} + w_{J_S}^i f_{J_S} S_{J_S} N_u) X_{ik}\}.$$

Therefore, the system disk utilization of server k is

$$\rho_k^S = \mu_k^S \sum_{i=1}^{N_f} \{(f_{q_i} S_{q_i} N_u + f_{u_i} S_{u_i} + w_{J_S}^i f_{J_S} S_{J_S} N_u) X_{ik}\}.$$

In this way, the data disk utilization and CPU utilization are calculated, respectively, as follows:

$$\rho_k^D = \mu_k^D \sum_{i=1}^{N_f} \{(f_{q_i} D_{q_i} N_u + f_{u_i} D_{u_i} + w_{J_S}^i f_{J_S} D_{J_S} N_{uf}) X_{ik}\}$$

$$\rho_k^C = \mu_k^C \sum_{i=1}^{N_f} [\{f_{q_i}(S_{q_i} + D_{q_i}) N_u + f_{u_i}(S_{u_i} + D_{u_i}) + w_{J_S}^i f_{J_S}(S_{J_S} + D_{J_S}) N_u\} X_{ik}].$$

The system utilization is the summation of disk utilization and CPU utilization; i.e.

$$\rho_k = \rho_k^S + \rho_k^D + \rho_k^C.$$

Since in a LAN we can ignore the communication overhead

compared with I/O load, we can consider that all data files are stored at a single server. This idea allows us to regard a join as a simple query. Therefore, the summation of weights for a join is assumed to be one, and the total file request rate is equal to the total transaction rate; i.e.

$$\lambda = \sum_{k=1}^{N_s} \lambda_k = \sum_{i=1}^{N_f} (f_{q_i} N_{uf} + f_{u_i} + w_{J_S}^i f_{J_S} N_u) = f.$$

System utilization is a summation of all server utilization; i.e. $\rho = \sum_{k=1}^{N_S} \rho_k$.Thus, according to queuing theory [6], the system response time is

$$\frac{1}{\lambda} \times \frac{\rho}{1-\rho}.$$

Finally, FWAP is formulated as follows.

Minimize $$\frac{1}{\lambda} \times \frac{\rho}{1-\rho}$$

Subject to

$$\sum_{k=1}^{N_s} X_{ik} = 1, \qquad \forall i = 1, 2, \ldots, N_f$$

$$\lambda_k = \sum_{i=1}^{N_f} (f_{q_i} X_{ik} + w_{J_S}^i f_{J_S} X_{ik}) N_u + \sum_{i=1}^{N_f} f_{u_i} X_{ik}, \qquad \forall k = 1, 2, \ldots, N_s$$

$$\lambda = \sum_{k=1}^{N_s} \lambda_k$$

$$\rho < 1$$

$$\sum_{i=1}^{N_f} w_{J_S}^i = 1$$

$$0 \le w_{J_S}^i \le 1, \qquad \forall i = 1, 2, \ldots, N_f$$

$$X_{ik} \in \{0, 1\}, \qquad \forall i = 1, 2, \ldots, N_f, \ \forall k = 1, 2, \ldots, N_s$$

First constraint implies that a data file is allocated to a single server, i.e. nonredundant allocation. Second constraint implies workload allocation, and third constraint ensures that total transaction rate of system is equal to the summation of all transaction rates of each server. Fourth constraint is stability condition for queuing system, and fifth constraint implies that in a LAN a join can be considered as a simple query. Sixth constraint shows that the weight of data file i for a join J_s can not be greater than one and can not be less than zero. Last constraint implies that each server contains at most one copy of each data file.

Next, we describe a model for designing the appropriate service capacity. The model determines the number of servers that satisfy the required service response time. First, we allocate all data files to a single server having the largest capacity. If any required service response time is not satisfied, allocation is done once again over two servers. Until all of the required service response times are satisfied, server number is increased by one and allocation is done again.

The service response time for each transaction is calculated as follows.

For the simple query routed to server k, the response time is

$$R_k^{q_i} = \frac{\{S_{q_i}\mu_k^S + D_{q_i}\mu_k^D + (S_{q_i} + D_{q_i})\mu_k^C\}X_{ik}}{1-\rho_k},$$

where $S_{q_k}\mu_k^S + D_{q_i}\mu_k^D + (S_{q_i} + D_{q_i})\mu_k^C$ is the processing time for the query.

In this way, the response time for update routed to server k is

$$R_k^{u_i} = \frac{\{S_{u_i}\mu_k^S + D_{u_i}\mu_k^D + (S_{u_i} + D_{u_i})\mu_k^C\}X_{ik}}{1-\rho_k}.$$

For a join, the response time of server k is

$$R_k^{J_S} = \frac{\left\{S_{J_S}\mu_k^S + D_{J_S}\mu_k^D + (S_{J_S} + D_{J_S})\mu_k^C\right\} \times \left(\sum_{i \in S} w_{J_S}^i X_{ik}\right)}{1-\rho_k}.$$

Since the service response time for each transaction varies according to the placement of the data file, all of the response times are calculated again, according to the results of each allocation stage.

To highlight some unique features, our FWSDM is compared with other key models. Table 1 is the summary of this comparison.

2.3. Solution procedure

Solving FWAP is not a trivial task. The complexity of the FWAP is described as follows.

Lemma. FWAP is NP-complete.

Proof. Let Θ, Θ_k be an average response time of the

Table 1
A comparison of design models

Comparison Criteria	Models							
	Wah and Lien [18] (1985)	Yu et al. [19] (1985)	Wah et al. [17] (1988)	Berman and Nigam [1] (1992)	Lee and Park [11] (1995)	Lee and Jang [8] (1996a)	Lee and Jang [9] (1996b)	FWSDM
Consideration factor	Communication overhead	Communication overhead	Communication overhead	Local processing	Local processing	Local processing, Communication overhead	Local processing, Communication overhead	Local processing
File redundancy	Redundant	Redundant	Redundant	Nonredundant	Nonredundant	Nonredundant	Redundant	Nonredundant
System parameters	Detailed	Detailed	Detailed	Subdivided in detail	Simple	Detailed	Detailed	Subdivided in detail
Join	Not considered	Not considered	Not considered	Considered	Not considered	Not considered	Not considered	Considered
Incorporation of service capacity design	No	No	No	Yes	No	No	No	Yes
File and workload allocation policy	Heuristic	Heuristic	Heuristic	Full enumeration	Heuristic	Heuristic	Heuristic	Heuristic
Server capacity	Homogeneous	Homogeneous	Heterogeneous	Homogeneous	Heterogeneous	Heterogeneous	Heterogeneous	Heterogeneous
LAN environment	Simple bus	Star	Two level bus	Simple bus	Simple bus	Simple bus	Simple bus	Simple bus

Table 2
Static information about system load

Frequency		Transaction					
		Join J_S, S = {1,2,4}	Query DB1	Query DB2	Query DB3	Query DB4	Update DB1
Transactions /terminal/hour		5	8	15	12	10	600
I/O counts per transaction	System disk	10	1	8	7	3	0
	Data disk	600	20	22	15	24	10

system, and server k, respectively, then

$$\Theta = \frac{1}{\lambda}\sum_{k=1}^{N_S}\lambda_k\Theta_k = \frac{1}{\lambda}\sum_{k=1}^{N_S}\frac{\rho_k}{1-\rho_k}.$$

Let f_i be a file request rate of file i, then

$$f_i = f_{q_i}N_u + w^i_{J_S}f_{J_S}N_u + f_{u_i}.$$

As a result, the model proposed in this paper is reduced as follows.

$$\text{Minimize} \quad \frac{1}{\lambda}\sum_{k=1}^{Ns}\frac{\rho_k}{1-\rho_k}$$

Subject to

$$\sum_{k=1}^{N_s}X_{ik} = 1, \qquad \forall i = 1,2,\ldots,N_f$$

$$\lambda = \sum_{k=1}^{N_s}\lambda_k$$

$$\lambda_k = \sum_{i=1}^{N_f}f_iX_{ik}, \qquad \forall k = 1,2,\ldots,N_s$$

$$X_{ik} \in \{0,1\}, \quad \forall i = 1,2,\ldots,N_f, \ \forall k = 1,2,\ldots,N_s$$

This problem is the nonlinear zero–one integer programming problem that is proven to be NP-complete [11].

Because of its NP-completeness, a heuristic is developed for solving the problem. We suggest that the heuristic should follow a well known result, "The system time using a multiple processing system is worse than a single centralized one" [7]. The following principles are derived from this result.

Principle 1: The data file having the larger request rate is preferentially assigned. The basic premise was that the file having the larger request rate had a more significant effect upon the system's performance.

Principle 2: A data file is allocated to a server that has the smallest system utilization in order to balance load over the system.

Principle 1 assures that the minimum numbers of file servers are needed to service transactions, because the larger request rate a file has, the more significant effect on the system performance it has. Principle 2 assures that the maximum system response time of transactions can be minimized because of load balancing. Load balancing can improve performance by transferring transactions from heavily loaded servers to lightly loaded servers. Since the system response time of server k is $(1/\lambda_k)\times[\rho_k/(1-\rho_k)]$, it can be minimized by the decrease of the system utilization of server k. When a server is heavily loaded, there is high probability for queuing time to be long, so the maximum system response time for a transaction of that server can increase considerably. Therefore a load balancing can be a good solution for this problem.

According to Principle 1, we arrange data files such a way that the file having the largest request rate is f_1, and the file having the smallest request rate is $f_{Nf.}$ We start the file and workload allocation heuristic by assigning the file f_1 to the server having the largest process capacity, because the file having the largest request rate has the most significant effect on the system performance, and must be assigned to the fastest server. Let E_k be a set that contains the current indices for the files assigned to the server k. Let U_k^S, U_k^D, U_k^C, U_k be a system disk utilization, data disk utilization, CPU utilization, and system utilization of server k for current files assigned to the server k. Then according to Principle 2, the file 2 is allocated to the server k that has the smallest system utilization. This process continues until all of files are assigned. The heuristic is summarized in Fig. 6.

2.4. Numerical example

Table 2 shows the static information about system load per user (terminal), i.e. expected transaction rate per terminal per hour during a period of peak demand. There are six types of transactions and four databases. Query is a simple transaction for requiring one database. Because nonredundant allocation policy is adopted, we update only one database that has the data needing to be updated. There is a join that needs three databases, DB1, DB2, DB4: that is, a join J_S requiring data file set S = {1,2,4}. Simple queries are q_1, q_2, q_3, and q_4. Load on the server disks is given by I/O counts per transaction. For example, Query DB3 accesses the system disk 7 times and accesses the data disk 15 times.

The service time for single I/O of database server is shown in Table 3. Three database servers are available and they have different speed for I/O processing.

Table 4 shows the weights of each data file for transactions. Except for the join, all transactions need only one

Table 3
Service time (sec) for single I/O of database server

	Server number		
	Server 1	Server 2	Server 3
Server component			
System disk	15.0	20.0	25.0
Data disk	15.0	20.0	25.0
CPU	7.0	10.0	15.0

Table 4
Weights for transactions

Transaction	Database			
	DB1	DB2	DB3	DB4
Query DB1	1	0	0	0
Query DB2	0	1	0	0
Query DB3	0	0	1	0
Query DB4	0	0	0	1
Update DB1	1	0	0	0
Join C_{124}	0.5	0.2	0	0.3

database, so their weight is one. The weights of DB1, DB2, and DB4 to join J_S are 0.5, 0.2 and 0.3, respectively.

In this example there are 30 terminals attached to a LAN, i.e. in a LAN thirty users can access DBs ($N_u = 30$). The required service response time for each transaction is presented in Table 5.

The request rates of each file per second are

$$f_1 = 8/3600 \times 30 + 600/3600 + 0.5 \times 5/3600 \times 30 = 0.2542,$$

$$f_2 = 15/3600 \times 30 + 0.2 \times 5/3600 \times 30 = 0.1333,$$

$$f_3 = 12/3600 \times 30 = 0.1000,$$

$$f_4 = 10/3600 \times 30 + 0.3 \times 5/3600 \times 30 = 0.0958.$$

First, we allocate all data files to one server having the largest capacity. Calculation results of system utilization are as follows.

$$U_1^S = 0.0365, \quad U_1^D = 0.5137, \quad U_1^C = 0.2568.$$

$$U_1 = U_1^S + U_1^D + U_1^C = 0.8070.$$

Thus system response time for each transaction is calculated as follows.

$$R_1^{q_1} = \frac{1 \times 0.015 + 20 \times 0.015 + (1 + 20) \times 0.007}{1 - 0.8070} = 2.3942$$

$$R_1^{q_2} = 3.4203, \quad R_1^{q_3} = 2.5082, \quad R_1^{q_4} = 3.0783,$$

$$R_1^{U_1} = 1.1401, \quad R_1^{J_S} = 69.5457.$$

Because any response time for each transaction does not satisfy the required service response time, we increase the server number by 1 and allocate data files over servers 1 and 2. We start the heuristic with allocating the DB1 to server 1. For server 1, the utilization is as follows:

$$U_1^S = 0.0041, \quad U_1^D = 0.22325, \quad U_1^C = 0.1104.$$

Therefore,

$$U_1 = U_1^S + U_1^D + U_1^C = 0.3471.$$

For server 2, the utilization is

$$U_2^S = 0.0055, \quad U_2^D = 0.3100, \quad U_2^C = 0.1578,$$

$$U_2 = U_2^S + U_2^D + U_2^C = 0.4733.$$

Because the utilization of server 1 is smaller than the utilization of server 2, data file 1 is allocated to server 1. That is, $k^{\cdot} = 1, E_1 = \{4\}, E_2 = \{\ \}$, and $X_{11}^* = 1$. In this way, we allocate other data files to servers. The result of file and workload allocation is shown in Table 6.

Consequently, the utilizations of server 1 and server 2 are 0.5643 and 0.3310, respectively. Therefore, the response time for each transaction is calculated as follows.

$$R_1^{q_1} = \frac{1 \times 0.015 + 20 \times 0.015 + (1 + 20) \times 0.007}{1 - 0.5643} = 1.0604$$

$$R_2^{q_2} = 1.3453, \quad R_2^{q_3} = 0.9865, \quad R_1^{q_4} = 1.3633,$$

$$R_1^{U_1} = 0.5049,$$

Table 5
Required service response time for each transaction (sec)

	Transaction					
	Query DB1	Query DB2	Query DB3	Query DB4	Join J_S, S = {1,2,4}	Update DB1
Response time						
Required service response time	1.5	1.5	2.0	2.5	15	0.5

Table 6
The results of heuristic over two servers

Data file	Utilization		File allocation
	Server 1	Server 2	
1	0.3471*	0.4733	$k^{\cdot}=1, X_{11}^{*}=1$
2	0.5413	0.2650*	$k^{\cdot}=2, X_{22}^{*}=1$
3	0.3995	0.3310*	$k^{\cdot}=2, X_{32}^{*}=1$
4	0.5643*	0.6273	$k^{\cdot}=1, X_{41}^{*}=1$

Table 7
The results of heuristic over three servers

Data file	Utilization			File allocation
	Server 1	Server 2	Server 3	
1	0.3471*	0.4733	0.6310	$k^{\cdot}=1, X_{11}^{*}=1$
2	0.5413	0.2650*	0.3534	$k^{\cdot}=2, X_{22}^{*}=1$
3	0.3955	0.3310	0.0880*	$k^{\cdot}=3, X_{33}^{*}=1$
4	0.5643	0.5613	0.4830*	$k^{\cdot}=3, X_{43}^{*}=1$

$$R_1^{J_S} = \frac{(10\times 0.015 + 600\times 0.015 + 610\times 0.007)\times(0.3+0.5)}{1-0.5643} = 21.6408,$$

$$R_2^{J_S} = \frac{(10\times 0.02 + 600\times 0.02 + 610\times 0.01)\times 0.2}{1-0.3310} = 5.4709.$$

The response time for update U_1 is beyond the required service response time, and the response time of server 1 for join J_S also does not satisfy the required response time. Thus we increase the server number by one, and allocate data files and workload. The results of file and workload allocation over three available servers are shown in Table 7.

As a result, utilizations of server 1, 2, 3 are 0.3471, 0.2650, 0.4830, respectively. Thus the response time for each transaction is calculated for each server and shown in Table 8.

As is seen in Table 8, all service response times satisfy the required service response time, thus three available servers are sufficient for serving all transactions. The adequate data file allocation is shown in Table 8. Data file 1 is allocated to server 1, data file 2 to server 2, data file 3 to server 3 and data file 4 to server 3. The adequate workload allocations are as follows.

$$\lambda_1 = 0.5\times\frac{5}{3600}\times 30 + \frac{8}{3600}\times 30 + \frac{600}{3600} = 0.2542,$$

$$\lambda_2 = 0.2\times\frac{5}{3600}\times 30 + \frac{15}{3600}\times 30 = 0.1333,$$

$$\lambda_3 = 0.3\times\frac{5}{3600}\times 30 + \frac{10}{3600}\times 30 + \frac{12}{3600}\times 30 = 0.1958.$$

3. System implementation

To demonstrate the practical usefulness of our methodology, a DSS according to the design procedures is implemented on an IBM PC. Among the components of the DSS [5], the user dialog component and model component are implemented by the use of Microsoft Excel [20] and Visual Basic [21]. This system consists of a data input subsystem, a data file allocation subsystem and a decision-supporting information subsystem. The example in the previous section is used to explain this system.

Fig. 7 shows the main screen that presents three main subsystems.

We start DSS by inputting data that are parameter values for the file allocation model.

For example, Fig. 8 shows the input screen, for hardware parameters that consists of CPU process time, system disk and data disk access time, for each server.

Fig. 9 shows the input step for transaction information. This information is the request rate per hour and the disk access times for processing the transaction.

After the data input stage is completed, data files are allocated over available servers. Fig. 10 shows the results of the file allocation. We can validate that the file allocation results of this system equal to the results in the example of this paper. It shows the placement of each file and the utilization of each server.

Table 8
Service response time for each transaction (sec)

Response time	Transaction							
	Query DB1	Query DB2	Query DB3	Query DB4	Join J_S of Server 1	Join J_S of Server 2	Join J_S of Server 3	Update DB1
Service response time	0.7076	1.2245	1.7021	2.0890	10.2764	4.9796	14.1586	0.3370

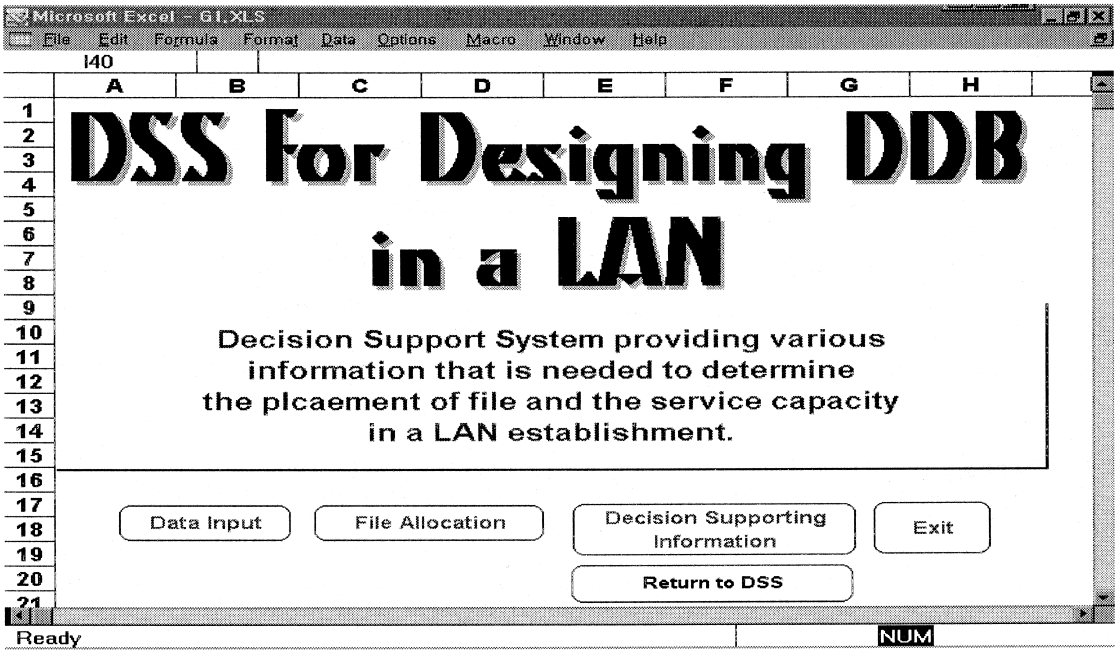

Fig. 7. Main screen.

This system provides many types of information needed for LAN establishment. The system asks the required service response time for each transaction in order to generate that information (Fig. 11).

The system response times, for the allocation results, over three servers can be obtained as shown in Fig. 12. It also provides the chart that compares the actual service response time with the required system response time for each transaction.

The service capacity satisfying the required service response time for each transaction is shown in Fig. 13, which also shows the comparison among server capacities.

The DSS explained here provides very useful information through what-if analysis. When the button of 'What-If Analysis' is pushed, a screen appears and asks the required response time (Fig. 11). This 'What-If Analysis' provides new file allocation, service capacity, and other information according to the new required response time.

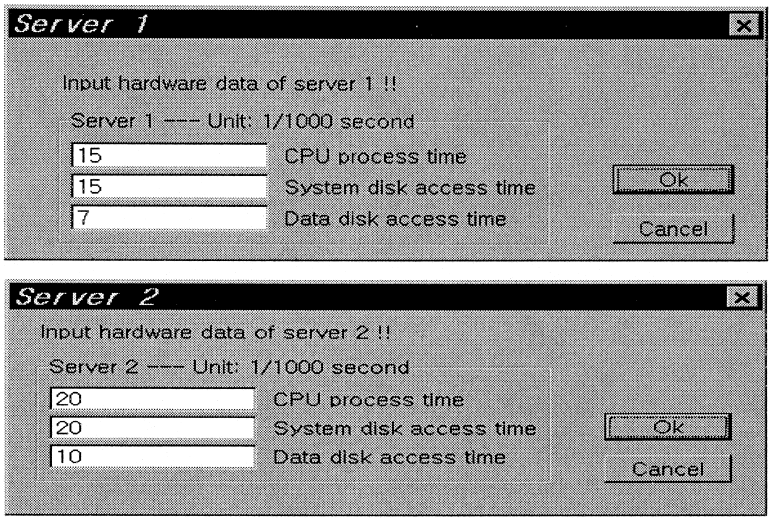

Fig. 8. Hardware data.

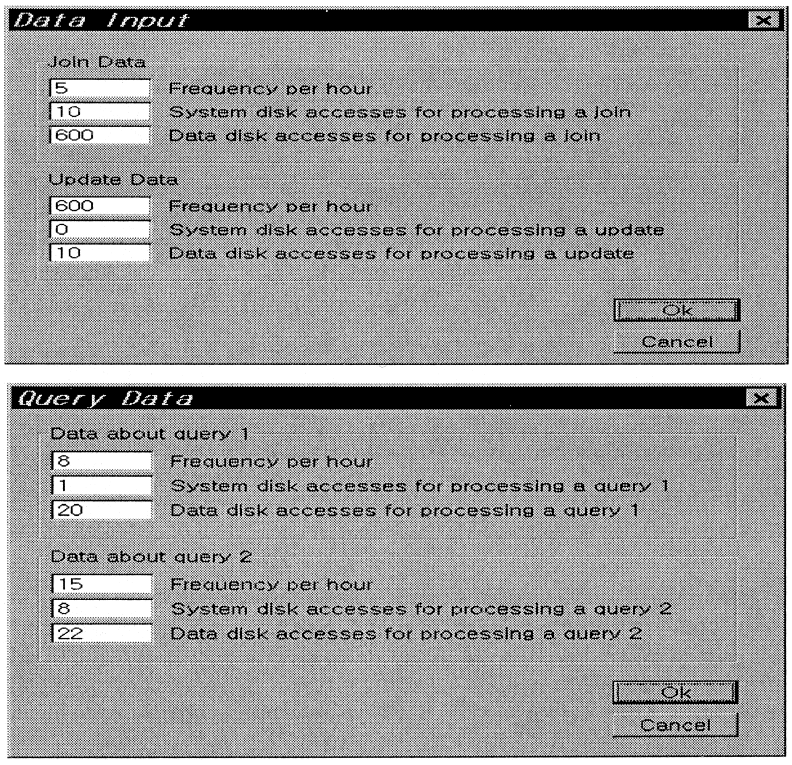

Fig. 9. Transaction data.

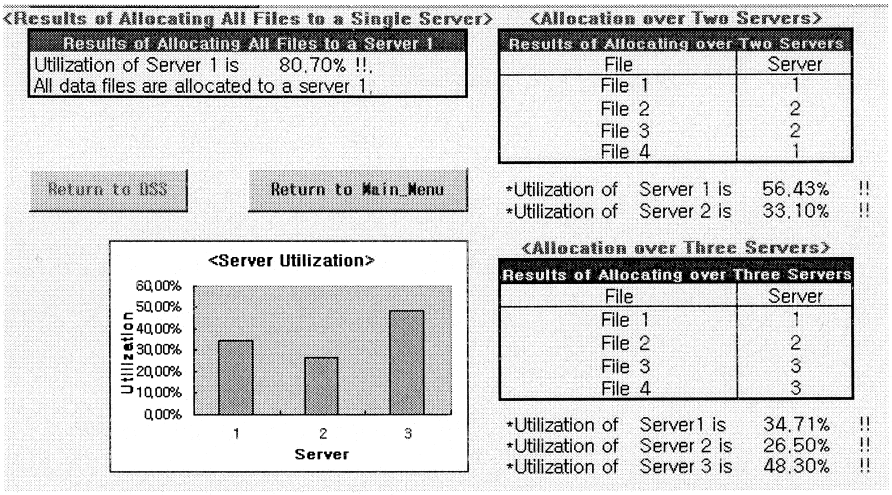

Fig. 10. The results of file allocation.

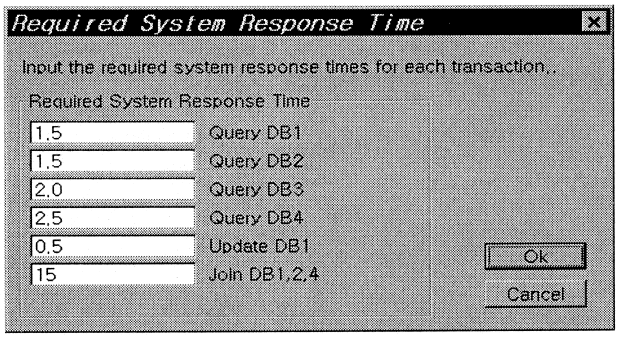

Fig. 11. The required response time for each transaction.

4. Conclusions

In a LAN establishment, the determination of file and workload allocation, and service capacities, is an important problem. This paper proposes a methodology for solving this problem. A heuristic is developed to solve the resulting NP-complete problem. A decision support system is implemented, which provides an interactive design capability. This feature is important because the decision problem is very complex. The system can help the user better design distributed databases in a LAN.

System Response Time

	Query1	Query2	Query3	Query4	Join of Server1	Join of Server2	Join of Server3	Update1
Response Time	0.7076	1.2245	1.7021	2.0890	10.2764	4.9796	14.1586	0.3369

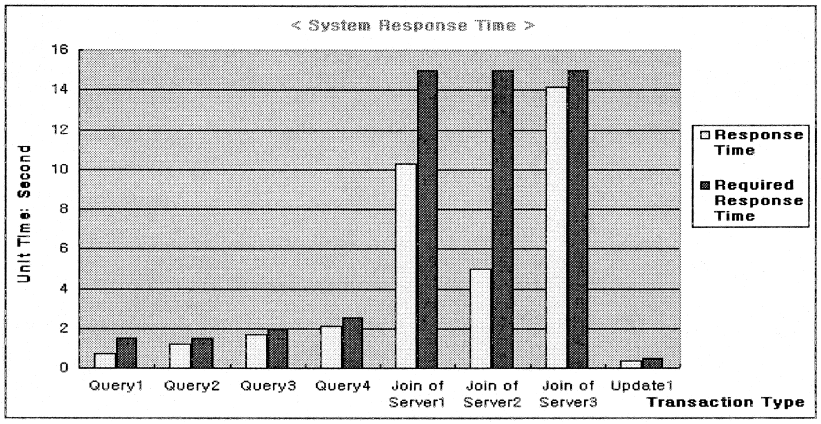

Fig. 12. The system response time.

< Information about Available Servers >

Service Time

	Server1	Server2	Server3
System Disk	15	20	25
Data Disk	15	20	25
CPU	7	10	15

Unit: 1/1000 second

**Number of servers satisfying the required system response time: 3 servers!!

Return to DSS

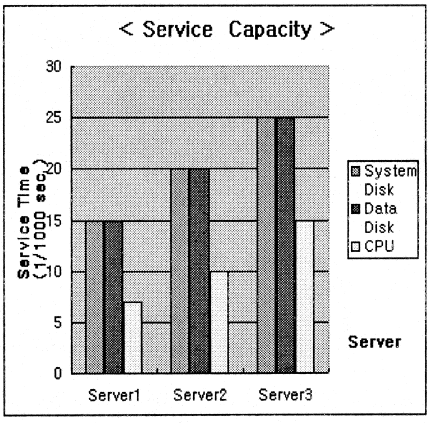

Fig. 13. Service capacity.

References

[1] L. Berman, R. Nigam, Optimal partitioning of data bases across multiple servers in a LAN, Interfaces 22 (1992) 18–27.

[2] S. Ceri, M. Negri, G. Pelagatti, Horizontal data partitioning in database design, Proceedings of the ACM SIGMOD Conference on Management of Data (1982) 128–136.

[3] P.P.S. Chen, Optimal file allocation in multi-level storage systems, Proceedings of the AFIPS National Computer Conference (1973) 277–282.

[4] P.P.S. Chen, Optimal partitioning of input load to parallel exponential servers, Fifth Southeastern Symposium on Systems Theory (1973) 66–69.

[5] S.Y. Huh, Modelbase construction with object-oriented constructs, Decision Science 24 (1993) 409–434.

[6] L. Kleinrock, Queuing Systems: Computer Applications, Wiley, New York, 1976.

[7] H. Lee, Simultaneous determination of capacities and load in parallel M/M/1 queues, European Journal of Operational Research 73 (1994) 95–102.

[8] H. Lee, G. Jang, Data file and workload allocation on a local multiaccess computer networks: incorporating local processing and communication overhead, International Journal of System Science 27 (9) (1996a) 831–837.

[9] H. Lee, G. Jang, File replication and workload balancing for a locally distributed database, Proceedings of the First Asia Pacific DSI Conference (1996b) 1255–1264.

[10] H. Lee, J. Lee, S.M. Nazem, Y. Shi, J. Stolen, M. anaging, user performance for a corporate network, Information and Management 35 (1999) 251–263.

[11] H. Lee, T. Park, Allocating data and workload among multiple servers in a local area network, Information Systems 20 (3) (1995) 261–269.

[12] M.T. Özsu, P. Valduriez, Principles of Distributed Database Systems, Prentice Hall, Englewood Cliffs, NJ, 1991.

[13] E. Rahm, framework for workload allocation in distributed transaction processing systems, Journal of Systems and Software 18 (1992) 171–190.

[14] W. Stallings, Data and Computer Communications, Prentice Hall, New York, 1994.

[15] W. Stallings, Local and Metropolitan Area Networks, Macmillan, New York, 1993.

[16] A.N. Tantawi, D. Towsley, Optimal static load balancing in distributed computer systems, Journal of ACM 32 (1985) 445–465.

[17] B.W. Wah, Y.L. Chang, Y.N. Lien, File allocation problems on homogeneous two-level local broadcast network, Proceedings of the Fourth International Conference on Data in Engineering (1988) 110–117.

[18] B.W. Wah, Y.N. Lien, Design of distributed databases on local computer systems with a multiaccess network, IEEE Transactions on Software Engineering 11 (1985) 606–619.
[19] C.T. Yu, M. Siu, K. Lam, C.H. Chen, Adaptive file allocation in star computer network, IEEE Transactions on Software Engineering (1985) 959–965.
[20] S.L. Nelson, Excel Power Presentations, Addison-Wesley, Reading, MA, 1992.
[21] R. Jennings, Database Developer's Guide with Visual Basic 3, Sams, Indianapolis, 1994.
[22] F.S. Hillier, G.J. Lieberman, Introduction to Operations Research, McGraw-Hill, New York, 1990.