Optimal File Allocation in a Multiple Computer System

WESLEY W. CHU, MEMBER, IEEE

***Abstract*—A model is developed for allocating information files required in common by several computers. The model considers storage cost, transmission cost, file lengths, and request rates, as well as updating rates of files, the maximum allowable expected access times to files at each computer, and the storage capacity of each computer. The criterion of optimality is minimal overall operating costs (storage and transmission). The model is formulated into a nonlinear integer zero-one programming problem, which may be reduced to a linear zero-one programming problem. A simple example is given to illustrate the model.**

***Index Terms*—Computer communication, linear integer programming, multicomputer information system, multiprocessor, nonlinear integer programming, optimal file allocation.**

Introduction

In the automation of large information systems, a major portion of the planning is concerned with storing large quantities of information in a computer system. This requires study of information storage, modification, and transmission. Examples of such efforts are found in business, medical, library, and management information systems. These systems, which may consist of several geographically separated divisions (subsystems), need to process information files in common.

It is apparent that when a given information file is required in common by several computers, it may be stored in at least one of them and accessed by the others when needed. The overall operating cost related to the files is considered to consist of transmission and storage costs. The problem is the following. Given a number of computers that process common information files, how can one allocate the files so that the allocation yields minimum overall operating costs subject to the following constraints: 1) the expected time to access each file is less than a given bound, 2) the amount of storage needed at each computer does not exceed the available storage capacity.

The Model

The file allocation problem can be formulated as an integer (0 or 1) programming model.

Let X_{ij} indicate that the jth file is stored in the ith computer:

$$X_{ij} = \begin{cases} 1 & j\text{th file stored in the } i\text{th computer} \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

where

$i = 1, 2, \cdots, n$
$j = 1, 2, \cdots, m$
n = total number of computers in the multicomputer system
m = total number of distinct files in the multicomputer system.

For storing r_j redundant copies of the jth in the information system, we have

$$\sum_i X_{ij} = r_j \qquad \text{for } 1 \le j \le m. \qquad (2)$$

To assure that the storage capacity of each computer is not exceeded, we have

$$\sum_j X_{ij} L_j \le b_i \qquad \text{for } 1 \le i \le n \qquad (3)$$

where

L_j = length of the jth file,
b_i = available memory size of the ith computer.

The expected time for the ith computer to retrieve the jth file from the kth computer (from initiation of request till start of reception) is denoted as a_{ijk}. The maximum allowable retrieval time of the jth file to the ith computer is T_{ij}. We required that a_{ijk} be less than T_{ij}, i.e.,

$$(1 - X_{ij}) X_{kj} a_{ijk} \le T_{ij} \qquad \text{for } i \ne k, \qquad 1 \le j \le m. \qquad (4)$$

When $r_j = 1$ for all j, then from (2) we know that $X_{ij} X_{kj} = 0$ for $i \ne k$. Thus (4) reduces to

$$X_{kj} a_{ijk} \le T_{ij} \qquad \text{for } i \ne k, \qquad 1 \le j \le m. \qquad (5)$$

Now, a_{ijk} is equal to the sum of the expected queuing delay at the ith computer for the channel to the kth computer[1] $w_{ik}^{(1)}$, the expected queuing delay at the kth computer for the channel to the ith computer $w_{ki}^{(2)}$, and the expected computer access time to the jth file t_{kj}. In most cases, the quantity t_{kj} is much smaller than $w_{ik} = w_{ik}^{(1)} + w_{ki}^{(2)}$ and can be neglected. Hence,

$$a_{ijk} \doteq w_{ik}. \qquad (6)$$

Manuscript received July 12, 1968; revised May 29, 1969. This is a generalized version of the paper entitled "Optimal file allocation in a multicomputer information system," presented at the IFIP Congress 68, Edinburgh, Scotland, August 5–10, 1968.

The author was with Bell Telephone Laboratories, Inc., Holmdel N. J. He is now with the Computer Science Department, University of the City of Los Angeles, Los Angeles, Calif.

[1] The number in the superscripts of $w_{ik}^{(\cdot)}$ and $w_{ki}^{(\cdot)}$ denotes the priority class of the messages that are transmitted between the ith and kth computers, which will be discussed shortly.

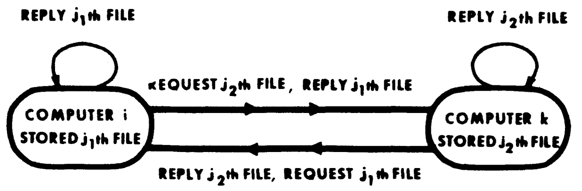
REPLY j_1th FILE
REPLY j_2th FILE
REQUEST j_2th FILE, REPLY j_1th FILE
COMPUTER i STORED j_1th FILE
COMPUTER k STORED j_2th FILE
REPLY j_2th FILE, REQUEST j_1th FILE

Fig. 1. Transmission paths between each pair of computers.

Next we will discuss the mechanism involved in the queuing delay. Each pair of computers is assumed to be able to transmit information in both directions simultaneously. This is known as single-channel full-duplex operation. Further, the files can be accessed by the local and remote computers at the same time. One pair of transmission paths links each pair of computers; one of these paths carries *requests* for the files from the ith computer to the kth computer, and *reply* messages (files) from the ith computer to the kth computer, whereas the other path carries *requests* for files from the kth computer to the ith computer, and *reply* messages from kth to the ith computer (Fig. 1). In most cases, the request message is much shorter than the reply message. Therefore, we shall assign a higher priority to request messages than to reply messages. Messages of the same priority are served in the order of their arrival. Assume, for example, a reply message is being transmitted on a particular transmission path at the time a new request is initiated. The request will interrupt the current reply and the computer will transmit the request first. After finishing transmission of that new request, the computer resumes transmitting the previous reply. Such preemptive-resume priority servicing faciltates optimization, since the queuing delay will be minimum if the shortest messages are serviced first [1], [2]. Recalling that the requested message length is much shorter than that of the reply message, the delay due to the request message can be neglected.[2] Under these conditions, the queuing system can be viewed as a single server queue with constant service time.

In many cases, it is reasonable to model the file accessing process as a Poisson process. Then the rate of requests from the ith computer to the kth computer (arrival rate) λ_{ik} is the sum of the rates of request of those files not stored in the ith computer but stored in the kth computer. The requested file length may be less than the entire storage file length and is defined as the length of each transaction; that is, the length of each transaction of the jth file l_j should be less or equal to the entire jth file length L_j. The average time required to transmit the reply from the kth computer to the jth computer $1/\mu_{ik}$ (i.e., service time) is dependent on l_j and λ_{ik}. Clearly, both λ_{ik} and $1/\mu_{ik}$ depend on the file allocation, yet the allocation is unknown in advance. Hence, we shall express λ_{ik} and $1/\mu_{ik}$ in terms of the X_{ij}'s:

$$\lambda_{ik} = \sum_j u_{ij}(1 - X_{ij})X_{kj} \tag{7}$$

where

u_{ij} = the request rate[3] of the entire or part of the jth file at the ith computer per unit time.

The average time required to transmit a reply message from the kth to the ith computer via a line with transmission rate R is

$$1/\mu_{ik} = \frac{1}{\lambda_{ik}} \sum \frac{1}{\mu_j} u_{ij}(1 - X_{ij})X_{kj} \tag{8}$$

where

$1/\mu_j = l_j/R$ = required time to transmit each transaction of the jth file.

Equation (8) states that $1/\mu_{ik}$ is equal to the time required for the kth computer to reply all the messages requested from the ith computer divided by the total number of requests initiated from the ith computer to the kth computer. Since the l_j's and R are constants, μ_j and μ_{ik} are also constants.

The traffic intensity from the kth to the ith computer ρ_{ik} measures the degree of congestion of the line that provides the transmission path between the kth and ith computer, or the fraction of time that the line is busy. It is defined as

$$\rho_{ik} = \lambda_{ik}/\mu_{ik} = \sum_j \frac{1}{\mu_j} u_{ij}(1 - X_{ij})X_{kj}. \tag{9}$$

Since physically it is impossible for the transmission line to be 100 percent busy, the traffic intensity is less than unity, i.e., $\rho_{ik} < 1$.

When only a single copy of the file is stored in the information system, then $X_{ij}X_{kj} = 0$ for $i \neq k$. Hence

$$\lambda_{ik} = \sum_j u_{ij}X_{kj} \qquad \text{for } i \neq k \tag{10}$$

and

$$1/\mu_{ik} = \frac{1}{\lambda_{ik}} \sum_j u_{ij} \frac{1}{\mu_j} X_{kj} \qquad \text{for } i \neq k \tag{11}$$

when all the $\mu_j = \mu$; then (11) reduces to

$$1/\mu_{ik} = 1/\mu. \tag{12}$$

The average waiting time [1], [2] from the initiation of a request at the ith computer to the receipt of the requested message from the kth computer, with the above assumptions (single server queue with Poisson arrivals and constant service time), is

[2] Preemptive priority servicing permits the assumption that $w_{ik} \doteq w_{ki}^{(2)}$. Note that the file retrieval time constraint as shown in (4) also applies to the nonpremptive priority case, but a more complex expression would be required for w_{ik} since $w_{ik}^{(1)}$ can no longer be neglected.

[3] In general, the request rate may be time dependent. In the analysis here, we are concerned with the request rate of the busy period.

$$w_{ik} \doteq w_{ki}^{(2)} = \frac{1}{\mu_{ik}} \frac{\rho_{ik}}{2(1-\rho_{ik})} \qquad \text{for } i \neq k \tag{13}$$

where $1/\mu_{ik}$ and ρ_{ik} are functions of X_{ij}'s as given by (8) and (9). The variance and probability distribution of w_{ik} for a specific allocation can be computed from its ρ_{ik} and μ_{ik} [1], [2].

Substituting (13) into (6) and (4), we have

$$(1-X_{ij})X_{kj}\frac{1}{\mu_{ik}} \frac{\rho_{ik}}{2(1-\rho_{ik})} \leq T_{ij},$$

which can be rearranged into the form

$$(1-X_{ij})X_{kj}\lambda_{ik} - 2\mu_{ik}(\mu_{ik}-\lambda_{ik})T_{ij} \leq 0. \tag{14}$$

For the special case that $\mu_{ik}=\mu$ and $r_j=1$, (14) reduces to

$$\sum_{\substack{j_1 \\ j\neq j_1}} u_{ij}X_{kj_1}X_{kj} + 2T_{ij}\mu \sum_{j_1} u_{ij_1}X_{kj_1} + u_{ij}X_{kj} - 2\mu^2 T_{ij} \leq 0. \tag{15}$$

Finally, we shall express the operating cost (objective function) in terms of the allocation (X_{ij}'s). Suppose we know the storage cost of the jth file per unit length and unit time at the ith computer C_{ij}, the transmission cost from the kth computer to the ith computer per unit time C_{ik}', the request rate for the entire or part of the jth file at the ith computer per unit time u_{ij}, the frequency of modification of the jth file at the ith computer after each transaction P_{ij}, the length of each transaction for the jth file l_j, and the number of redundant copies of the jth file stored in the system r_j. Then the overall operating cost per unit time C for processing m distinct files required in common by n computers is

$$C = \underbrace{\sum_{i,j} C_{ij}L_jX_{ij}}_{\text{storage cost}} + \underbrace{\sum_{i,j,k} \frac{1}{r_j} C_{ik}'l_ju_{ij}X_{kj}(1-X_{ij}) + \sum_{i,j,k} C_{ik}'l_ju_{ij}X_{kj}P_{ij},}_{\text{transmission cost}}$$

which can be rearranged into the form

$$= \sum_{i,j} D_{ij}X_{ij} - \sum_{i,j,k} E_{ijk}X_{kj}X_{ij} \quad \text{where} \quad D_{ij}>0, \qquad E_{ijk}>0 \tag{16}$$

when $r_j=1$, $1\leq j\leq m$; then $X_{kj}X_{ij}=0$ for $k\neq i$, and $C_{ii}'=0$. Under this case, (16) reduces to

$$C = \sum_{i,j} D_{ij}X_{ij}. \tag{17}$$

We want to minimize (16) subject to storage and access time requirements constraints given in (1), (2), (3), and (14). As X_{ij}'s take on value zero or one, the allocation problem becomes solving a nonlinear zero-one programming problem. In the next section, we shall introduce a technique to reduce the nonlinear zero-one equations to the linear zero-one equations. With this technique, the allocation problem can be then solved by standard linear zero-one programming techniques [3], [4].

Reduction of Zero-One Nonlinear Programming Problems to Zero-One Linear Programming Problems[4]

Because nonlinear programming problems are so complex, we are able to obtain a global optimal solution only for special cases (e.g., convexity). Therefore, it is desirable to reduce the nonlinear zero-one programming problems to the linear zero-one programming problems. We shall now show such a reduction, which is derived from the integer (0 or 1) property.

Suppose we want to minimize an arbitrary cost function (which need not be convex)

$$C = \min_X F(X_1, X_2, \cdots, X_k) \tag{18}$$

subject to a set of nonconvex constraint equations

$$G_i(X_1, X_2, \cdots, X_k) \leq B_i \qquad i=1,2,\cdots,N \tag{19}$$

where the X_i's are zero-one variables, F and G_i are polynomials of the X_i's with constant coefficients, and B_i is a constant. Clearly, $X_i^q = X_i$ (q = positive integer). Let the coefficient of the product terms in (18) or (19), $X_iX_j, \cdots, X_uX_v$, be denoted as $C_{ij\cdots uv}$.

To reduce the above nonlinear zero-one problem to a linear zero-one problem, first we consider the objective function (18). Let us define

$$X_{ij\cdots uv} = \underbrace{X_iX_j\cdots X_uX_v}_{q} \qquad q=2,\cdots,Q \tag{20}$$

which takes value zero or one where Q is the highest degree of nonlinearity. We then represent each nonlinear term in (18) by terms of the form (20) and then examine its coefficient. If the coefficient of the nonlinear term is positive, we introduce the following constraint equation:

$$\underbrace{X_i + X_j \cdots + X_u + X_v}_{q} - q + 1 \leq X_{ij\cdots uv}. \tag{21}$$

[4] A similar but less general result has also been obtained independently by Watters [5].

If the coefficient of the nonlinear term is negative, we introduce the following constraint equation:

$$\underbrace{X_i + X_j + \cdots + X_u + X_v}_{q} \geq qX_{ij \ldots uv}. \tag{22}$$

If all the X's in left side of (21) have value one, then $X_{ij \ldots uv} = 1$. If one or more of the X's have value zero, then $X_{ij \ldots uv}$ may be either zero or one, but the coefficient of $X_{ij \ldots uv}$ in (18), $C_{ij \ldots uv}$, is positive. Thus minimizing (18) under X assures that $X_{ij \ldots uv} = 0$. If we substitute (20) for each nonlinear term in (18) that has positive coefficient and introduce the additional constraint (21), then the X's in the transformed linear equations take on the same values as the original ones. Similarly, if one or more of the X's in (20) have value zero, then $X_{ij \ldots uv} = 0$. If all the X's have value one, then $X_{ij \ldots uv}$ may either be zero or one, but the coefficient of $X_{ij \ldots uv}$ in (18), $C_{ij \ldots uv}$, is negative. Thus minimizing (18) under X assures that $X_{ijkl \ldots uv} = 1$. If we substitute (20) for each nonlinear term in (18) that has negative coefficient and introduce the additional constraint (22), then the X's in the transformed linear equations take on the same values as the original ones. Thus, we have linearized the objective function (18).

To linearize the constraint equations, we represent the nonlinear terms in (19) by (20) and introduce its corresponding *two* additional constraint equations (21) and (22). If one or more of the X's in (20) have value zero, then $X_{ij \ldots uv} = 0$; this condition is also satisfied by (21) and (22). If all the X_{ij}'s have value one in (20), then $X_{ij \ldots uv}$ is one. Similarly, from (21) $X_{ij \ldots uv}$ may be either zero or one, but from (22) $X_{ij \ldots uv}$ is one. Thus (21) and (22) assure that $X_{ij \ldots uv} = 1$. Substituting (20) and introducing the additional constraint (21) and (22) for each nonlinear term in (19) satisfy all the relationships of X's. Thus, we have also linearized the nonlinear constraints equations.

With this reduction technique,[5] nonlinear zero-one programming problems may be transformed into solution of linear zero-one programming problems. Using available linear integer programming techniques [3], [4], we can obtain the global optimal solutions.

Linearization of the Objective Function and Access Time Constants

To apply the above technique to linearize the objective function (16), we let $X_{ij}X_{kj} = X_{ijkj}$. Since the coefficients of X_{ijkj} in (16) are negative, for each X_{ijkj}, we introduce the additional constraint equation

$$X_{ij} + X_{kj} \geq 2X_{ijkj}. \tag{23}$$

Next, we shall linearize the constraint equation (15). We let $X_{kj_1}X_{kj} = X_{kj_1kj}$. For each X_{kj_1kj}, we introduce two additional constraint equations

$$\begin{cases} X_{kj_1} + X_{kj} \geq 2X_{kj_1kj} \\ X_{kj_1} + X_{kj} - 1 \leq X_{kj_1kj} \end{cases} \quad \text{for } j \neq j_1. \tag{24}$$

In the same manner, we can linearize (14). Hence, solution to the optimal file allocation problem is reduced to: minimize (16) subject to (1), (2), (3), (14) or (15), (20), (23), and (24) which is a linear zero–one programming problem.

File Allocation in a Multiprocessor

Let us consider a multiprocessor with virtual memory system that operates in a paging environment. One of the important problems in such a system is how to allocate files to various types of available storage systems such as thin films, cores, disks, drums, data cells, tapes, etc., so that the operating cost is minimum yet the access time requirements are satisfied for each file, and the storage limitation of each storage system is not exceeded. The model developed in this paper can be directly applied to this problem by letting the distances between computers equal to zero. Clearly, under this condition, a multiple computer system becomes a multiprocessor.

Example

Consider a specific computer communication system consisting of three computers that process five information files in common, as shown in Fig. 2. These computers are located about 20 miles from each other. The transmission facility between each pair of computers has a rate of $R = 5 \times 10^3$ char/second. The cost of each such facility is \$1050 per month or $\$1.4 \times 10^{-7}$/char (based on 100 hours per week and 4.2 weeks per month). The first cost of storage is 35 cents per character or $\$5.8 \times 10^{-7}$/char second.[6] Table I gives the lengths of each file, file length for each transactions, the request rate of these files at each computer, the rate of modification of these files at each computer after each transaction, the available storage capacity of each computer, and the maximum allowable retrieval time of each file. Using the linearized model developed above and the Gomory cutting technique [3], [4] for solving the integer linear programming problem, the example was solved on the IBM 360/65. Table II lists the optimal allocation for the case of no redundant files. The computation time required for this example is about 25 seconds.

Some characteristics of the optimum allocation are worthy of note. File 4 is stored in computer 2 as it is only used by that computer. File 1 has highest request

[5] This reduction technique can be easily extended to the case when the X_i's in (18) or (19) are real numbers. In this case, we shall express each X_i in terms of binary variables X_{ij}'s as follows:

$$X_i = \sum_{l=1}^{\alpha_i+\beta_i} (X_{il})2^{\alpha_i - l}$$

where α_i is chosen large enough for $2^{\alpha_i - 1}$ to be an upper bound on the value of X_i, while β_i is chosen large enough for $2^{-\beta_i}$ to be the maximum allowable accuracy tolerance on the value of X_i. Thus (18) or (19) is reduced from a nonlinear equation to a nonlinear zero–one equation.

[6] The calculation is based on 40 months of machine life operating at 100 hours per week and 4.2 weeks per month.

TABLE I
DATA OF EXAMPLE FOR FILE ALLOCATION

				Computer 1		Computer 2		Computer 3	
File j	L_j	l_j	P_j	u_{1j}	T_{1j}	u_{2j}	T_{2j}	u_{3j}	T_{3j}
1	100×10^3	500	0.5	5	30	2	10	0	0
2	10×10^3	500	0.5	0	0	2	30	5	1
3	10×10^3	500	0.5	3	10	0	0	4	1
4	10×10^3	500	0.5	0	0	4	0.1	0	0
5	100×10^3	500	0.5	1	1	1	1	5	1

P_j = the frequency of modification of the jth file after each transaction.
L_j = length of the jth file in characters.
l_j = file length (in characters) of each transaction for the jth file.
u_i = average hourly request rate of the entire or part of the jth file at the ith computer. Request arrivals are assumed to be Poisson distributed.
T_{ij} = maximum allowable average retrieval time in seconds for the jth file to the ith computer.
C_{ij} = (storage cost) = \$0.58 $\times10^{-8}$/char second.
C_{ik}' = (transmission cost) = \$1.4 $\times10^{-7}$/char.
b_i = (available storage capacity of the ith computer) = 110×10^3 char for $i=1, 2, 3$.
$1/\mu = l/R$ (the time required to transmit the reply message) = 0.1 second.

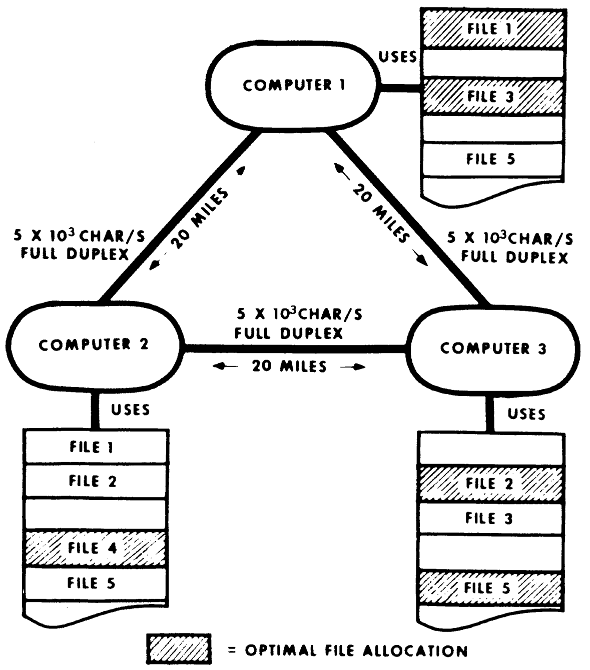

Fig. 2. A specific multiple computer system.

rate at computer 1; to minimize the transmission cost, it is stored in computer 1. For the same reason, Files 2 and 5 are stored in computer 3. Although File 3 has a high request rate at computer 3, the available storage size of computer 3 (110×10^3 char) forced File 3 to be stored in computer 1. The overall operation cost under the optimal allocation is \$3620 per month. If each file is stored at the computer where it is used, the total operating cost under such arrangement is \$6670 per month. The higher operating cost is due to the extra storage cost and the file updating cost.

Conclusion

The file allocation problem can be formulated into a nonlinear zero–one programming problem. By adding additional constraint equations, these nonlinear terms

TABLE II
OPTIMAL FILE ALLOCATION FOR EXAMPLE

		X_{ij}	
File j	Computer 1	Computer 2	Computer 3
1	1	0	0
2	0	0	1
3	1	0	0
4	0	1	0
5	0	0	1

in the objective and constraint equations can be reduced to linear equations. Thus, solution of the optimal allocation requires solving a linear zero–one programming problem. The model introduced in this paper provides a common denominator for analysis and comparison of various proposed information system configurations, a tool to study the sensitivity of various parameters and constraints to the operating cost, and a method for evaluating the growth potential of information systems. However, some related problems still require further studies, such as file reliability, privacy, file partition, etc. All these problems are important considerations for optimal file allocation in multicomputer information systems.

Acknowledgment

The author would like to thank E. Fuchs of Bell Telephone Laboratories for his stimulating discussions.

References

[1] D. R. Cox and W. L. Smith, *Queues, Methnen's Statistical Monographs.* London, England: Spottiswoode, Ballantyne, and Co., 1961, pp. 50–59.
[2] T. L. Saaty, *Elements of Queuing Theory.* New York: McGraw-Hill, 1961, pp. 153–161.
[3] R. Gomory, "All-integer integer programming algorithm," *Industrial Scheduling,* Muth and Thompson, Eds. Englewood Cliffs, N. J.: Prentice-Hall, 1963, pp. 195–206.
[4] J. Haldi and L. M. Isaacson, "Linear integer programming," Graduate School of Business, Stanford University, Stanford, Calif., working paper 45, December 1964.
[5] L. J. Watters, "Reduction of integer polynomial programming problems to zero–one linear programming problems," *Operations Res.*, vol. 15, pp. 1171–1174, November–December 1967.