The Efficient Use of Buffer Storage

Kunio Fukunaga and Tamotsu Kasai

University of Osaka Prefecture

Abstract

The computer storage consists of a large, slow main storage and a small, fast storage called a buffer, which is integrated into the CPU. A proper management of data in the two storages can satisfy the desire for large and fast memory. The performance of this system is significantly influenced by the hit rate, which is defined as the probability of finding the data wanted for a fetch in buffer storage.

This paper discusses an approach of representing the degree of association between variables for a given program and describes an associative assignment of program variables into main storage in order to improve the hit rate. As a result of the simulation, this association mehtod provides a good indication of performance.

1. Introduction

The desire for large memory and for fast memory, to keep pace with CPU speed, are conflicting goals. A properly designed storage hierarchy in the memory system is one method of resolving the conflict between size and speed requirements [1]. The storage hierarchy consists of a large, slow main storage and a small, fast storage called a buffer, which is integrated into the CPU. The buffer is not addressable by a program, but rather is used to hold the contents of those portions of main storage that are currently being used. Most processor fetches can then be handled by referring to the buffer, so that most of the time the processor has a short access time. When the program starts operating on data in a different portion of main storage, the data in that portion must be loaded into the buffer storage and the data from some other portion removed. This system provides effective access time near that of the small buffer memory to an amount of information equal to the capacity of the large main storage. In the hierarchy system, the hit rate plays an important role [2]. This rate is defined as the probability of finding the data wanted for a fetch in buffer storage.

The purpose of this paper is to formulate a model for the degree of association between the variables in the program and to describe the procedure for an associative assignment of the variables to main storage in order to improve the hit rate. The degree of association between two variables is measured on the basis of the frequency of finding these two at the same time in an executable statement of a given program. The variables in a given program are divided into blocks in proportion to the degree of strength of association among them and the variables in a block are assigned into a sector of main storage. If data is requested by the CPU and is unfortunately not found in buffer storage, the sector with this data is transferred from main storage to buffer. In the subsequent request by the CPU, the data can be easily found in buffer storage as a natural course of association among the data in the sector. This results in a higher hit rate, which results in shorter access time.

2. Association among variables

Consider the joint distribution plane between program variables and executable statements of a given program (Fig.1). Let each program variable $v_j \in V$ for $j=1,2,\cdots\cdots,n$ be represented at the position x_j on the X-axis and each executable statement $s_i \in S$ for $i=1,2,\cdots\cdots,m$ be represented at the position y_i on the Y-axis of this plane. Here, an array is considered to be a variable. The sign "•" is marked at the coordinates (x_j, y_i) if the variable v_j is contained in the executable ststement s_i. Fig.2 illustrates this plane of the example program. Using this joint distribution plane, we can define the correlation coefficient between the variables and the executable statements. The value of the correlation coefficient depends on the positions x_j (variable v_j) and y_i (statement s_i) on both axes. We obtain the maximum value of the correlation coefficient by selecting the proper value of x_j and y_i. Under this condition, the mark "•" is distributed along a diagonal line on the joint distribution plane as in Fig.3. According to the arrangement of

```
                                  No. of execu-
                                  table statement.
   READ (5,10) M,K ---------- S1
   N=2*M+K ------------------ S2
   DO 20 I=1,N -------------- S3
   A=FLOAT(I)-2.5 ----------- S4
20 P=P+A--------------------- S5
   R=P*P--------------------- S6
   WRITE (6,30) P,R,A ------- S7
   STOP---------------------- S8
10 FORMAT(2I5)
30 FORMAT(1H1,3E15.7)
   END
```

Fig.1 An example of the program (P).

variables on the X-axis in this figure, the nearer the distance between the two variables, the higher the rate of finding them at the same time in a statement. In contrast to this, two variables cannot be found at the same time in an executable statement if one variable is far from the other on the X-axis. Therefore, the distance between the variables on the X-axis represents the rate of finding the variables at the same time in an executable statement. We call this rate the degree of association between variables for a given program. Consequently, beneath the maximum correlation coefficient, the varables are arranged on the X-axis in proportion to the degree of strength of association among them.

In the following section, we shall discuss the approach for obtaining the maximum value of the correlation coefficient.

3. Association plane

The joint distribution plane can be considered to represent the discrete probability distribution respect to X and Y. This probability function is given as follows,

$$p(X=x_j,\ Y=y_i)=\delta_{ji}/T_o\ , \tag{1}$$

where δ_{ji} and T_o are

$$\delta_{ji}=\begin{cases}1 & ;\ \text{if the variable } v_j \text{ is contained in the statement } s_i,\\ 0 & ;\ \text{otherwise,}\end{cases}$$

and

$$T_o=\sum_{j=1}^{n}\sum_{i=1}^{m}\delta_{ji}\ . \tag{2}$$

We abbreviate the probability function of Eq.(1) as $p(x_j,y_i)$. The probability function $p_1(x_j)$ in the variable X can be obtained by use of this probability function stated above.

$$p_1(x_j)=\sum_{i=1}^{m}p(x_j,y_i)\ . \tag{3}$$

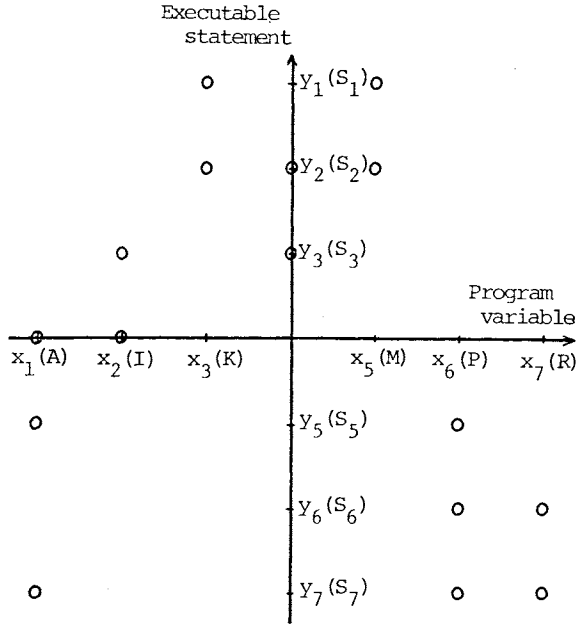

Fig.2 The joint distribution plane of the program (P).

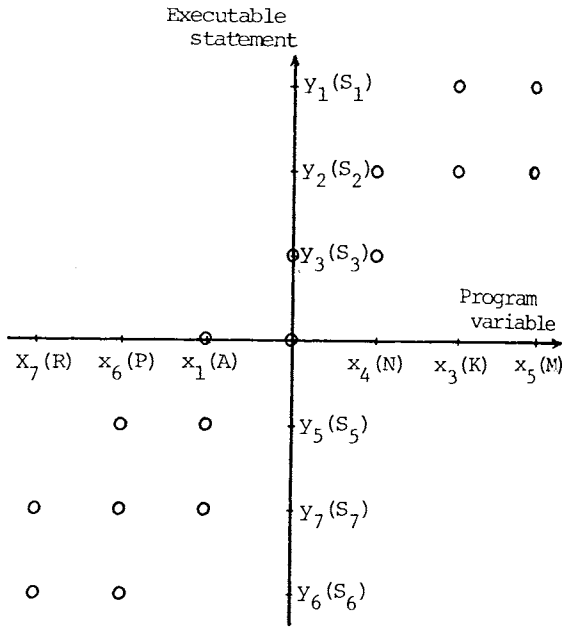

Fig.3 The joint distribution plane of the program (P) under the maximum correlation coefficient.

In the same way, we have

$$p_2(y_i)=\sum_{j=1}^{n} p(x_j,y_i). \quad (4)$$

Now, let us find the values for variables x_j and y_i so that the correlation coefficient may be maximum.

The variance σ_x^2 in X, σ_y^2 in Y and the covariance C_{xy} are given from their definitions.

$$\sigma_x^2=[\sum_{j=1}^{n} x_j^2 p_1(x_j)]-[\sum_{j=1}^{n} x_j p_1(x_j)]^2, \quad (5)$$

$$\sigma_y^2=[\sum_{i=1}^{m} y_i^2 p_2(y_i)]-[\sum_{i=1}^{m} y_i p_2(y_i)]^2, \quad (6)$$

$$C_{xy}=\sum_{j=1}^{n}\sum_{i=1}^{m} x_j y_i p(x_j,y_i) - [\sum_{j=1}^{n} x_j p_1(x_j)][\sum_{i=1}^{m} y_i p_2(y_i)]. \quad (7)$$

Then, the correlation coefficient ρ is given by

$$\rho=\frac{C_{xy}}{\sigma_x\sigma_y}. \quad (8)$$

In order to obtain the maximum correlation coefficient, we differenciate with respect to the variables x_j(j=1,2,------,n), y_i(i=1, 2,------,m), and set the derivatives equal to 0.

$$\frac{\partial\rho}{\partial x_k}=0 \quad , \quad (k=1,2,-------,n), \quad (9)$$

$$\frac{\partial\rho}{\partial y_e}=0 \quad , \quad (e=1,2,-------,m). \quad (10)$$

From these equations, we have [3]

$$A\boldsymbol{x}=\lambda C\boldsymbol{x} \ , \quad (11)$$

where the parameter λ is given by

$$\lambda=\rho^2 \ , \quad (12)$$

and $\boldsymbol{x}$ is represented by the column vector,

$$\boldsymbol{x}=(x_1,x_2,\cdots\cdots\cdots,x_n)^t \ , \quad (13)$$

and each element x_j represents the position of program variable v_j on the X-axis in the joint distribution plane.

The element of the symmetric matrix A (=a_{jk}) is defined by

$$a_{jk}=\sum_{i=1}^{m} \delta_{ji}\delta_{ki}/r_i \ , \quad (14)$$

where $r_i=T_o\, p_2(y_i)$ and the element of the diagonal matrix C (=c_{kk}) is defined by

$$C_{kk}=\sum_{j=1}^{m} \delta_{kj} \quad (15)$$

respectively.

Solving the characteristic equation of Eq.(11), we obtain the eigenvalues $\lambda_1(=\rho_1^2)$ and $\lambda_2(=\rho_2^2)$, which are arranged according to the order of their size. The eigenvector $\boldsymbol{x}^{(k)}$ associated with the eigenvalue λ_k(k=1,2) is represented by

$$\boldsymbol{x}^{(k)}=(x_{1k},x_{2k},\cdots\cdots\cdots,x_{nk})^t, \quad (16)$$

where the maximum element of the vector is normalized to 1. It follows from these vectors that we can define the association vector of each variables as follows:

[Definition]

The association vector $\boldsymbol{x}_j$ of program variable v_j is given by

$$\boldsymbol{x}_j=(x_{j1},x_{j2})^t \ , \quad (17)$$

where x_{jk} (j=1,2,------,n, k=1,2) is defined in Eq.(16).

Next, let us consider the plane, in which each variable v_j is represented by a point at the coordinates (x_{j1},x_{j2}). There is no question that the distance d_{jk} between variables v_j and v_k is given by

$$d_{jk}=\|x_j-x_k\|=[\sum_{p=1}^{2} (x_{jp}-x_{kp})^2]^{1/2} . \quad (18)$$

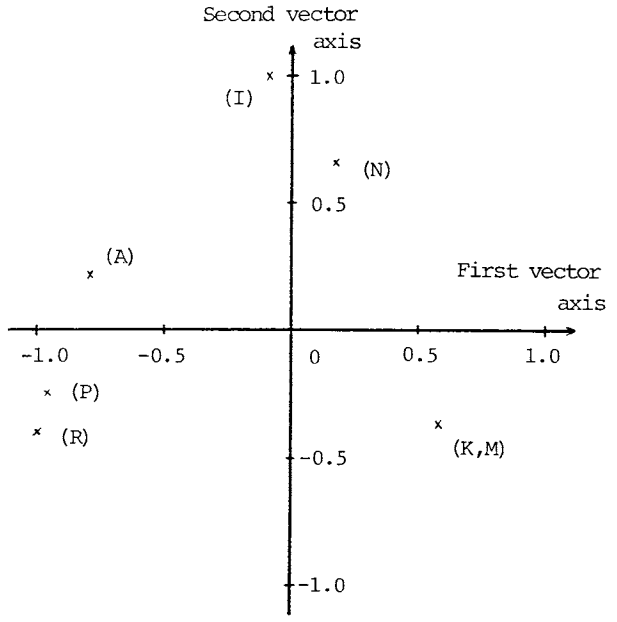

Fig.4 The association plane of the program (P).

According to the discussion about association in Sec.2, this distance represents the degree of association between the two variables. Hereafter, we shall call this plane the association plane. Fig.4 illustrate the association plane of the program (P).

Here, we define the common variables and the assignment variables by using the distance on the association plane.

[Definition]

The variables which satisfy the following two conditions are called the common variables and the rest in the program are called the assignment variables.

(1) The norm (d_j) of the association vector of a variable v_j is lesser than ξ_1.

$$d_j = \|x_j\| < \xi_1. \tag{19}$$

(2) The number (f_j) of executable statements which contain the variable v_j is more than ξ_2.

$$f_j > \xi_2. \tag{20}$$

4. Associative assignment

Fig.5 illustrates the storage system discussed. A small buffer storage with fast access time is packaged into the heart of the CPU. Supposed that the main storage has a true access time and cycle time on the order of more than ten times that of the buffer memory. Data is transferred from main storage to buffer storage in blocks. Main storage is logically divided into sectors, which are subdivided into blocks. Buffer storage is also divided into sectors and into blocks. A sector from main storage can be mapped into any sector in buffer storage [1].

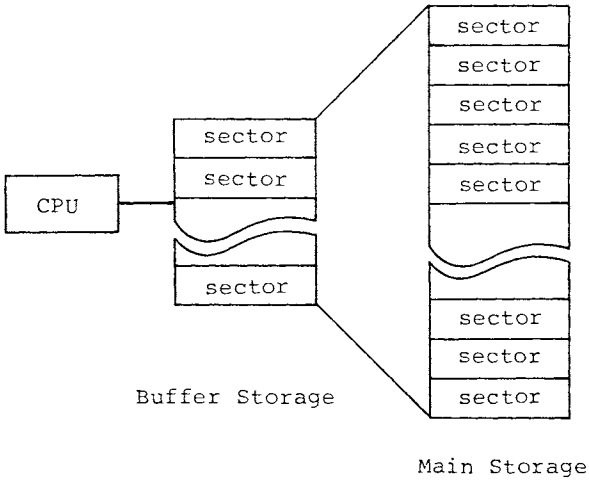

Fig.5 The hierarchical storage system.

Up to now, we have presented the method to represent the program variables on the association plane. As we have described in Sec.3, the distance among the variables represents the degree of association on this plane. Using this degree of association, we now propose the associative assignment of variables as follow:

For the common variables,

(i) The variables are assigned into the sectors of the main storage in order of their appearance in the program.

For the assignment variables,

(ii) We divide this plane into V-shaped blocks (sectors) so that the number of variables may be same as that in a sector of main storage.

(iii) The variables in a sector of this plane are assigned into a sector of the main storage.

Fig.6 illustrates this associative assignment of variables.

In the next, we shall discuss the assigning of the main storage sector to the buffer sector [4]. In this system, during operation a correspondence is set up between buffer sectors and main storage sectors in which each buffer sector is assigned to a single different main storage sector. The assignment of buffer sector is dynamically adjusted during operation, so that they are assigned to the main storage sectors that are currently being used by the program. If the program causes a fetch from a main storage sector that does not have a buffer sector assigned to it, one of the buffer sectors is then reassigned to that main storage sector. To make a good selection of a buffer sector to reassign, enough information is maintained to

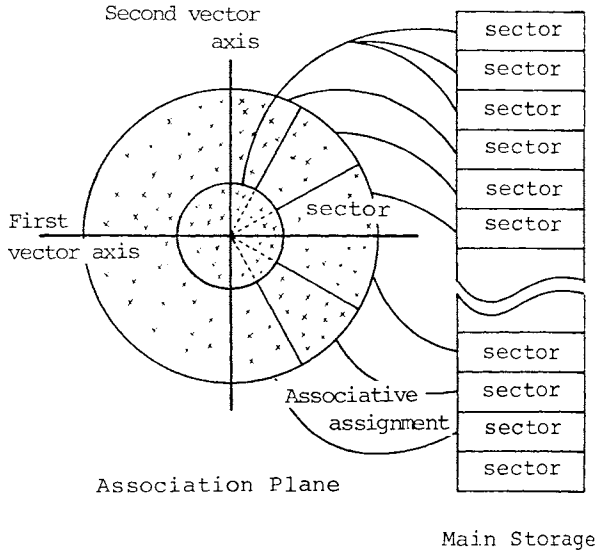

Fig.6 The associative assignment of variables.

order the buffer sectors into an activity list. The sector at the top of the list is the one that was most recently referred to, the second one is the next most recently referred to, and so forth. When a buffer sector is referred to, it is moved to the top of the list, and the intervening ones are moved down one position. This is not meant to imply an actual movement of sectors within the buffer, but rather referred to a logical ordering of the sectors. When it is necessary to reassign a sector, the one selected is the one at the bottom of the activity list.

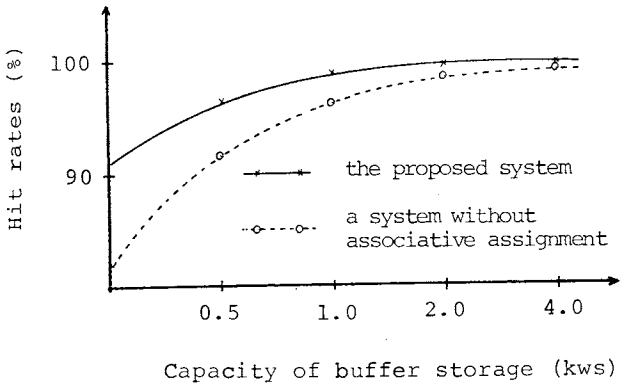

Fig.7 Relations between hit rates and the capacity of buffer storage.

5. Performance studies

In order to measure the effectiveness of the associative assignment, we postulated a system identical to the proposed system, except that variables are assigned into main storage in order of their appearance in the given program. An important statistic related to this comparison is the hit rate. The equivalent access time (t) of the system is represented by the following equation, using the hit rate (h) [2],

$$t=[h+(1-h)t_2/t_1]t_1 \qquad (21)$$

where t_1 and t_2 are access times of buffer and main storage, respectively. Befor discussing the result of the simulation, let us define the storage system in detail. The buffer storage is divided into 10 sectors and each sector subdivided into words, where one word is four byte. In the same way as the buffer storage, the main storage is divided into sectors with the same size as in the buffer. In the simulation, the program contained about 100 variables on the average. Here, an array is considered to be one variable. We obtained the association vectors related to these variables and assigned them into main storage using the associative assignment. The total memory capacity needed to load the program was about 40 kws on the average.

Fig.7 shows the hit rates of the computer simulated results for both the postulated system and the system without the associative assignment. It is apparent that the former is higher hit rate than the later in any capacity, especially in the small capacity of the buffer storage. Note that the access time of buffer storage is more than ten times as fast as that of main storage. Therefore, the equivalent access time of the postulated system is reduced remarkably in comparison with the system without the associative assignment.

The time needed to obtain the association vectors and the associative assignment of variables was several seconds. As the result of numerical studies [5], it is well known that the necessary time to get this association vectors increases in proportion to the square of the number of the variables. This time, however, is relatively small compared to the computing time of the programs.

6. Conclutions

The associative assignment of variables has been introduced to the system with storage hierarchy in order to improve its equivalent access time. This access time is influenced significantly by the hit rate, which is the probability of finding the data wanted for a fetch in buffer storage. It has become clear through the results of the simulation that the associative assignment improve the hit rate and reduces considerably the equivalent access time.

Acknowledgement

We want to show our appreciation to Mr. M. D. Cox for helping us with our English.

References

1. Conti, C. J., "Concepts for buffer storage", Computer Group News, March (1969).

2. Takahashi, S., "Electronic computer", Kyoritsu, Tokyo (1975).

3. Hayashi, C., "Theory and examples of quantification (II)", Jour. Inst. Statis. Math. Japan, Vol.4, No.2 (1956).

4. Conti, C. J., "Structural aspects of the System/360 Model 85", IBM System Jour., Vol.7, No.1 (1968).

5. Amamiya, A., "Numerical analysis and FORTRAN", Maruzen, Tokyo (1970).