High Storage Utilization of Hash Memory by Reducing of Information Redundancy for Hashing

E.G.BARDIS*, N.G.BARDIS*, A.P.MARKOVSKI*, A.K.SPYROPOULOS**

* Department of Computer Science
National Technical University of Ukraine
(Kiev Polytechnic Institute)
Glyfada-Athens
Tainarou 66
16561
HELLAS
** Department of Mathematics
University of Athens

Abstract:- The article presents a method of hash memory utilization efficiency increase, which is based on excluding of information redundancy inherent in hash-addresing. It has been shown theoretically that a part of information contained in the key is duplicated in the hash-address code, and the technique that decreases substantially this redundancy is described. This technique utilizes signature storage in the hash-memory, instead of whole keys storage in the auxiliary memory. The main advantage of the method consists in decrease of memory capacity required. The practical application of this technique makes it possible to decrease the hash memory capacity, to increase the searching rate and to provide security of the information stored in the hash memory.

Examples illustrating the method and technique suggested are also presented.

Key words:- information theory, information redundancy, hashing, hash memory, hash signature, auxiliary memory, database management systems. *IMACS/IEEE CSCC'99 Proceedings*, Pages:3101-3105

1 Introduction

Efficiency of searching is important for total system performance. Thus, the choosing of a good searching technique can save a lot of time in the retrieval operations.

Hashing is an efficient and widespread information retrieving technique because its retrieval time is much less comparing to the other methods and this time does not depend on the key numbers among which the searching is being carried out. Hashing ensures effective searching both in static and dynamic key massifs. The idea of hashing consists in the fact that the address at which the key is stored in the memory is a key code function. At key recording into the hash memory, first its hash-address is being computed, and then, the key-code and the information related to this key are being recorded in to this hash-address. At searching, the hash address is computed by the key-code, the code is being read from the hash memory and compared with the received key-code and then, if the codes have coincided, either the information related to the key or the information file address, in case the information is stored on a disk, is being read.

2 Problem Formulation

It is well known that the main disadvantage of the potentially fastest method of searching by key, i.e. hash-addressing is the existence of collisions.

The probability of a collision depends on a degree of hash memory fill, which is called a loading factor. The loading factor is usually given by an α coefficient, which is the ratio of m stored keys and M hash memory addresses, i.e. $\alpha=m/M<1$.

The existence of collisions causes a necessity to store in the hash memory key codes which are being compared in the course of the searching with the given keys.

Thus, besides the collision existence, the substantial disadvantage of the hash addressing is demand for a

considerable memory capacity that is always greater than the key memory capacity. The present work suggests a technique of hash addressing, which eliminates to a great extent this disadvantage of the fastest method of searching by the key.

3 Information Redundancy in the Hash Memory

In general, a keyword code contains I_x bits of an information. If the assumption is valid under which the values (0 or 1) of separate bits of a keyword are independent and the appearance of zeros and ones in those bits is equiprobable, then the amount of the information I_x, contained in the keyword, is numerically equal to its bit number n. At hash addressing the key X is being converted into the $A_x=H_{(x)}$ hash address, where $H(...)$ is the hash conversion function. Besides, the bits number of hash address code A_x is equal to $[\log_2 M]$, where $M=[m/\alpha]$ number of the hash memory address corresponds to the quotient of division of the key number m , stored in the hash memory, by the hash memory loading factor α. Let the amount of the information about the keyword code, which is transmitted in the process of hash conversion into the hash address code, be equal to I_A. The upper bound of the I_A probable values, is defined by the $[\log_2 M]$ number of hash address bits, though in practice in most cases $I_A < [\log_2 M]$. Since the number of hash address bits is always less than the keyword one, $I_A<I_X$. On account of $I_A>0$, it is obvious that storing the whole key in the hash memory is redundant from the information point of view, because a part of the information about the key code, namely I_A, is already contained in the address code A_X at which the key X is stored.

For elimination of this redundancy it is suggested to store in the key memory not the whole n-bit key code, but a certain h-bit code S_X of the whole key hash signature, which is the result of application of functional conversion Ψ(X) over the X code of the keyword, under satisfaction of two following conditions:

•each possible keyword X has the one-to-one correspondence to the pair $<A_X,S_X>$, where $A_X=H(X)$, $S_X=\Psi(X)$ i.e. the hash signature code, must contain the amount of the initial key information I_S that is suffice to hold $I_S+I_A=I_X$;

•the number h of the hash signature bits is always less than the number of keyword bit n, i.e. $h<n$.

The hash number h of signature bits, is defined by the functions H(X) and Ψ(X). The low bound of the possible values of h is defined by the case, when the choice of the hash conversion H(X) and hash signature Ψ(X) functions assures the following conditions to be met:

$$I_A<[\log_2 M], \quad I_S=h \tag{1}$$

In this case the number h of the hash signature code bits, is defined by the equation:

$$h=n-[\log_2 M] \tag{2}$$

For practical implementation of the suggested method for reducing of hash memory information redundancy, the choice of H(X) and Ψ(H) should meet the next conditions:

•H(X) function must generate uniformly distributed hash addresses at any key code distribution law;

•H(X) and Ψ(X) functions choice, must ensure the uniqueness of the solution of the next equation system with respect to X at given A and S:

$$H(X)=A, \quad \Psi(X)=S \tag{3}$$

Both given conditions may be met by making use of selected in a special way linear Boolean functions H(X) and Ψ(X). It means that each bit z_j, j=1,...,d, (d=$[\log_2 M]$) of the hash address and each bit of the hash signature S_i, i=1,...,h are being raised in the following form:

$$\begin{aligned} &z_j=\left(\sum_{k=1}^{n} b_{kj}\cdot x_k\right)\bmod 2,\ j=1,...,d,\\ &b_{kj}\in\{0,1\} \quad k\in Q_j \quad \text{if } b_{kj}=1\\ &s_i=\left(\sum_{k=1}^{n} c_{ki}\cdot x_k\right)\bmod 2,\ i=1,...,h\\ &c_{ki}\in\{0,1\} \quad k\in R_i \quad \text{if } c_{ki}=1 \end{aligned} \tag{4}$$

The choice of the $Q_1,...,Q_d$ and $R_1,...,R_h$ sets determines correspondingly the H(X) and Ψ(X) functions.

For satisfaction of the second condition all the linear Boolean functions defined by (4) must be linear independent.

The next step after forming the functions generating the bits of a hash address is the generation of the functions for forming the h binary bit code of the hash signature $(s_1,s_2,...,s_h)$, and the generation of n functions of initial key bit values recovery by the code of the hash address and signature:

$$x_i=\Phi_i(z_1,...,z_d,s_1,...,s_h)=$$
$$z=(\sum_{t\in\vartheta_i})\bmod 2+(\sum_{q=T_i}s_q)\bmod 2 \qquad (5)$$

To solve the given problems it is necessary to prove the correctness of the following theorem.

Theorem: If d linear Boolean functions, forming each one bit hash address, are given, then, for any n-bit key word $X=(x_1,...,x_n)$, there exist h=n-d functions $\psi_1(X),...,\psi_h(X)$ the hash signature, such that their values in totality with the hash address bits assure one-to-one correspondence to the initial key code. It implies the satisfaction of condition (3), for $\psi_i(X)=s_{1i}$, $s_{1i}\in X$, i=1,...,h, i.e. each i-th bit of the hash signature is equal to one key code. In so doing, the functions $\Phi_i,(A,S)$, i=1,...,n can be always formulated by recovery.

Proof. Let H(X) function of hash address formation be according to (4) given by a set of binary vectors $B_j=(b_{j1},...,b_{jn})$, j=1,...,d. Consider the set Θ of the binary vectors which are the result of all the possible XOR of the vectors $B_1,...,B_d$, in other words, consider the set of the vectors which are linearly dependent on $B_1,...,B_d$. The total number of t vectors making up the set Θ is equal to 2^d-1. Let us demonstrate that a vector whose number of ones does not exceed n-d+1 can always be found among the set Θ. The proof will be carried out from the antithesis. Let us suppose the set Θ comprises only vectors whose number of ones exceeds n-d+1. It means that the following conditions are met

$$\forall A_i,...,A_j\in\Theta,\ i,j\in\{1,...,2^d-1\},\ i\neq j \Rightarrow A_i\oplus A_j\in\Theta$$
$$\forall A_i(\alpha_{i1},\alpha_{i2},...,\alpha_{in}),\alpha_{ik},\in\{0,1\},k=1,...,n$$
$$\sum_{k=1,...,n}\alpha_{ik}>n-d+1 \qquad (6)$$

i.e. the Hamming distance between any two vectors of the set Θ exceeds n-d+1.

For each vector $A_i\in\Theta$, i=1,...,t the set Ω_i of vectors $\Omega_i=\{Q_{i1},Q_{i2},...,Q_{in}\}$ different from the A_i vector only in one component can be determined, i.e. the Hamming distance between the k-th vector from the indicated $Q_{ik}=(q_{ik1},q_{ik2},...,q_{ikn})$ ones and the vector A_i is equal to one:

$$HD(A_i,Q_{ik})=\sum_{l=1,...,n}\alpha_{il}\oplus q_{ikl}=1\ k=1,...,n \qquad (7)$$

It is evident that $Q_{ik}\notin\Theta$ as it follows from $Q_{ik}\in\Omega_i$, because $HD(A_i, Q_{ik})<n-d+1$. It can be shown that $\Omega_i\cap\Omega_j=\varnothing\ \forall$ i,j=1,...,t, i≠j. Indeed, if a binary vector U such that $U\in\Omega_i$ and $U\in\Omega_j$ existed, it would mean that $HD(A_i,U)=1$ and $HD(A_j,U)=1$, and this, in its turn, would mean that $HD(A_i,A_j)=HD(A_i,U)+HD(A_j,U)=2\leq n-d+1$ at the maximal value of d=n-1. This is in contradiction with condition (6) according to which the Hamming distance between any two vectors $A_i,A_j\in\Theta$ strictly exceeds n-d+1 for $n-1\geq d\geq 1$. Consequently, the vector U such that $U\in\Omega_i$, and $U\in\Omega_j$, does not exist and therefore $\Omega_i\cap\Omega_j=\varnothing\forall$i,j=1,...,t, i≠j.Accordingly, the total number of the vectors belonging to $\Omega_1\cup\Omega_2\cup...\cup\Omega_t$, is equal to $n\cdot t=n\cdot(2^d-1)$. The sum of the number of indicated vectors which belong and do not belong the Θ set may not exceed the total number of binary vectors that equals 2^n, i.e.

$$n\cdot(2^d-1)+2^d-1=(2^d-1)\cdot(n+1)\leq 2^n \qquad (8)$$

In practice n≥8 always, therefore condition (8) is held only at d≤n-4. This implies that at $d\in\{n-1,...,n-3\}$ the assumption under which the set Θ comprises only vectors whose number of ones exceeds n-d+2 is not held and, consequently the set Θ contains the vectors with the number of ones less than or equal to n-d+1.

Consider the case when d≤n-4 and reveal that also in this case the set Θ inevitably comprises the vectors whose number of ones is equal to or less than n-d+1. Let us suppose for this purpose that this condition is not held, then the Hamming distance between any two vectors of the set Θ exceeds n-d+1=4 This implies that the set Ω_i comprising the binary vectors can be determined for each vector $A_i\in\Theta$, and the Hamming distance to the A_i vector does not exceed 2: $\forall Q\in\Omega_i$, $HD(A_i,Q)\leq 2$. It is obviously that $\Omega_i\cap\Omega_j=\varnothing$ ∀i,j=1,...,t, i≠j. Then the number of vectors belonging to $\Omega_1\cup\Omega_2\cup...\cup\Omega_t$ is equal to $n\cdot t+(^2_n)\cdot t=n\cdot t+n\cdot(n-1)\cdot t/2$. Expression (8) under condition d≤n-d+1 can be transformed in to the form:

$$n\cdot t+t\cdot n\cdot(n-1)/2+t=$$
$$=(2^d-1)(n+n\cdot(n-1)/2+1)\leq 2^n \qquad (9)$$

Under condition n≥8, that is always valid in practice, inequality (9) is held at d≥n-6, i.e. at $d\in\{n-1,...,n-6\}$ the accepted assumption that the set Θ contains only the vectors with the number of ones more than n-d+1 turns out to be false.

By analogy with specification of inequality (8) given above, specification for d≤n-6 may be carried

out through d≤n-4. In this case each of the sets Ω_i will contain the vectors, the Hamming distance of which from the vector $A_i \in \Theta$ makes 1-3 units. Moreover, in the manner described above the following unequality may be derived:

$$(2^d-1)\cdot(n+n\cdot(n-1)/2+n\cdot(n-1)\cdot(n-2)/6+ +1\le 2^n \quad (10)$$

The analysis of the latter inequality reveals that it is held for d≤n-9 taking account of n≥8 and for d≤n-12 if n≥16. The further proof is performed similarly by decreasing the value d for which the statement being proved is valid, till d takes a value less than n/2. In so doing, it is obviously that the set Θ cannot contain a vector with the number of ones satisfying n-d+1>n/2 to hold condition (6). Thus, it is proved that at any value of d the set Θ obligatory contains the vector whose number of ones is less than or equal to n-d+1. The mentioned vector $B_r=\{b_{r1},b_{r2},...,b_{rn}\}$is the sum of a certain subset $\Delta_r \subseteq \Theta$ modulo 2 and corresponds to the following equation:

$$b_{r1}\cdot x_1 \oplus b_{r2}\cdot x_2 \oplus ... \oplus b_{rn}\cdot x_n =$$
$$= x_{q,1} \oplus x_{q,2} \oplus ... \oplus x_{q,h+1} = \left(\sum_{k\in\Delta_r} z_k\right) \bmod 2 \quad (11)$$
$$q_1,q_2,...,q_{h+1} \in \{1,...,n\};\ q_1<q_2<...<q_{h+1}$$

The variables $x_{q,1},...,x_{q,h}$ of this equation can be determined by corresponding assigning to the h-bit hash signature code values that the are stored in the hash memory. In this case the latters are being arised in the form of $S_u=\Psi_u=x_{qu}$, u=1,...,h, and the variable $x_{q,h+1}$ may be represented as the solution of the following equation:

$$X_{q,h+1}=\left(\sum_{k\in\Delta_r} z_k\right) \bmod 2 \oplus \left(\sum_{j=1,...,h} S_j\right) \bmod 2 \quad (12)$$

In this manner h+1 equations from the total equation number n of system (5) may be obtained. The rest of equations of system (5) may be obtained as the solution of equations that correspond to the other 2^d -2 vectors contained in the set Θ. In this case, if the number ν of ones in the vector B_r is less than h+1, then only ν-1 bit functions of hash signature formation are determined for the solution of equation (6). The remaining h-ν+1 functions of hash signature bit formation can be obtained in the manner described above, making use of the other equations given by the set Θ.

4 Employment of Hash Memory Redundancy Reducing for Memory Capacity Decrease and Searching Acceleration

Storing in the hash memory not the full n bit key codes, but their h bit hash signatures (h<n), may result either in a reducing of the required total key memory capacity with maintaining the hash memory loading factor α or in a reducing of that with maintaining the total key memory capacity.

Decrease of required memory capacity for the hash memory is attained by decrease of the number of the bits stored in the memory from n to h, i.e. only h bits are utilised instead of the whole n bit key. Respectively, the memory capacity is reduced by $n/h \approx n/(n-[\log_2 M])$ times.

Here may arise an effect of compressing, the idea of which is in the fact that memory capacity required for storage of the number m of the n bit keys will come out to be less than m·n i.e. the hash memory capacity will be less than the theoretical minimal memory capacity required for storing of the keys. The following inequality should be satisfied to meet the condition for information compressing effect arising:

$$m/\alpha\cdot(n-\log_2(m/\alpha))<m\cdot n \quad (13)$$

From this inequality the boundary value of the key number m_c can be easily obtained at exceeding of which the effect of key compressing in the hash memory takes place:

$$m>m_c=\alpha\cdot 2^{n\cdot(1-\alpha)} \quad (14)$$

Exclusion of data storage redundancy in the hash memory may be also used in another way. The number M of hash memory cells may be increased from M to M′ and their capacity decreased from n to h with keeping the total memory capacity. Implementation of this approach makes it possible to obtain a set of hash memory structural solutions, where the values of the loading factor α and the number m of the records stored in the memory are varied. The limiting case here is the version of hash memory reconfiguration under which its capacity (n·M=h·M') and the key number m do not vary, and the loading factor α decreases by M'/M=n/h times, that entails collision probability decrease, searching time decrease or decrease of the time for finding the perfect hash algorithm for collision free addressing.

At increasing the hash memory cell number from M to M′ with simultaneous decreasing the capacity from n to h , at the invariable total hash memory capacity and the m numbers of records stored, from the condition of the total memory capacity conservation the next equation which relates the values mentioned above may be obtained:

$$M \cdot n = M' \cdot h \qquad (15)$$

M′ is determined as the solutions of the following equation :

$$F(M') = M' \cdot (n - [\log_2 M']) / M - n = 0 \qquad (16)$$

The memory loading factor α in this case decreases from the initial value of m/M to the value $m \cdot h / n \cdot M$. Thus, in the hash memory with utilization of the exclusion operation the average number T_{0h} of accesses to the memory is defined by the formula:

$$T_{0h} = (1 - \alpha \cdot h/n)^{-1} \qquad (17)$$

If the exclusion operation is not unionized in the hash memory, the average number T_{1h} of accesses to the memory in searching is defined by the formula:

$$T_{1h} = -\ln(1 - \alpha \cdot h/n) \cdot n / \alpha \cdot h \qquad (18)$$

Thus, storage of hash signatures in the key memory instead of the whole keys utilization for the systems of information storage, while the information varies dynamically enables the time of information searching by the key to be deceased substantially, i.e. the main index of hash memory efficiency to be increased at an invariable capacity of the allocated memory.

6 Conclusions

The investigation exposed in the present work proceeds from the fact that there exists information redundancy in the hash memory and at full bit key storage. Duplicating a part of the information about the key, contained in the hash address which is a function of the key code always takes place and it reduces the hash memory efficiency. Information redundancy, inherent for hash addressing may be either utilized for increase of reliability of information storage in the hash memory or eliminated due to special arrangement of key storage in the hash memory.

It has been shown that with the optimal algorithm of hash address formation, i.e. with the algorithm which assures a uniform distribution of hash addresses in the address space of the hash memory at any key massif distribution, in d bits of a hash address d bits of information about the key is stored and consequently there exists a possibility of storage in the hash memory (n-d) bit hash signature instead of n bit keys. The theoretical foundation of the technique of exclusion of information redundancy in the hash memory is presented, the problems of collision settling are considered. It has been proved that combined utilization of a restricted probing in the main memory and in an auxiliary memory of a small capacity is the best way of collision settling in the hash memory without duplication. Analytical models for solution of problems of synthesis and analysis of the hash memory without information duplication has been obtained. All theoretical statements and conclusions have been verified experimentally by statistical simulation.

Practical efficiency of the approach suggested has been investigated in details. The results of investigation proved expediency of practical implementation of the method and procedures suggested as well as good prospects for further study in this direction.

References:

[1] Kohonen T. Content- Addressable Memories. Springer series in information sciences, Vol 311, Spriger-Verlag. 1997.

[2] Jagannathan R. Optimal partial-match hashing design .ORSA Journal on Computing Vol.3, No.2,1991, pp.86-91.

[3]Koushic M.,Diehr G. Linear-density hashing with dynamic overflow sharing. Information Systems, Vol.17, No.5, 1992, pp.359-380.

[4]Larson P.A. Linear Hashing with Separetors - A dynamic hashing scheme achieving one-access retrieval .ASM Trans.of Database Systems, Vol.13 No.3, 1988, pp.366-388.

[5]Ou S.F., Thar A.L. High storage utilisation for single-probe retrieval linear hashing. Computer Journal, Vol. 34, No.5, 1991, pp.455-468.