УДК 004.074.32

МАРКОВСКИЙ А.П.,
ПОРХУН Е.В.
МНАЦАКАНОВ А.В.

ХЕШ-ПАМЯТЬ С ОГРАНИЧЕННЫМ ВРЕМЕНЕМ ПОИСКА ПО КЛЮЧУ

В статье предложена новая организация хеш-поиска, которая предполагает хранение ключа по одному из двух хеш-адресов. Это позволяет ограничить время поиска двумя обращениями к памяти. Предложена процедура рекурсивной записи ключей в память. Получены аналитические оценки вероятности коллизий. Проанализированы возможности использования предложенной организации хеш-поиска для статических и динамических массивов данных.

In article the new hash-searching organization such that keys may storing in one of dual hash-address has been proposed. It is allowed to limit the searching time by dual memory access. The procedure of recursion recording keys into the hash-memory has been developed. The analytical evaluation of collision probability has been obtained. The possibilities of proposed organization for hash-searching in static and dynamic arrays of data has been analyzed

Введение

Поиск по ключу традиционно является одной из базовых операций обработки информации. В последнее десятилетие многократно выросли объемы информационных массивов, в которых выполняется поиск. Расширение использования информационных технологий для управления процессами, протекающими в реальном времени имеет следствием ужесточение требований к времени выполения поиска по ключу.

Указанные факторы снижают эффективность использования в современных условиях традиционных технологий поиска, для которых время поиска зависит от объема поискового массива. В этих условиях создаются предпосылки расширения применения хеш-поиска — потенциально наиболее быстрой технологии поиска, для которой время доступа не зависит от объема данных, в которых выполняется поиск. До последнего времени широкое применение хеш-памяти сдерживалось присущими ей недостатками: наличием коллизий и избыточностью требуемой памяти. Успехи технологии изготовления интегральных микросхем привели к резкому удешевлению памяти и, тем самым, создали предпосылки к снижению значимости второго из указанных недостатков хеш-адресации.

Наличие коллизий, обусловленных совпадением хеш-адресов различных ключей, требует специальных механизмов для их разрешения. Это усложняет работу с хеш-памятью и обуславливает случайный характер времени поиска. Вместе с тем, существует достаточно широкий круг применений, прежде всего, связанных с обработкой информации в реальном времени, для которых важным является гарантированное время поиска.

Исходя из этого, на современном этапе развития информационных технологии, проблема ограничения времени поиска по ключу при использовании хеш-адресации является актуальной и значимой для практики.

Анализ известных технологий разрешения коллизий

Сущность хеш-адресации состоит в том, что адрес, по которому хранится информация в хеш-памяти, функционально зависит от этой информации. Для ключа X его хеш-адрес A определяется в результате выполнения над ключом хеш-преобразования $h(X)$: $A=h(X)$. В большинстве случаев, число ключей, хранимых в хеш-памяти меньше общего числа ее ячеек, так, что хеш-памяти присуща избыточность использования объема накопителя. Основной характеристикой избыточности является коэффициент загрузки – α, определяемый как отношение числа m хранящихся в памяти ключей к общему числу M ячеек хеш-памяти: $\alpha=m/M$ [1]. Поскольку разрядность ключей всегда больше разрядности хещ-адреса, то при их размещении в хеш-памяти возникает явление коллизии – ситуации, при которой несколько различных ключей имеют одинаковый хеш-адрес.

В современной технологии хеш-адресации целесообразно выделять два типа хеш-поиска: в динамическом и статическом массивах клю-

чей. В первом случае коллизии неизбежны и для их разрешения используются специальные средства. Для статических массивов ключей можно путем подбора хеш-функции добиться отсутствие коллизий. Такая хеш-адресация постоянных массивов ключей называется perfect или совершенной [2].

Эффективность динамической хеш-памяти характеризуется зависимостью скорости поиска от коэффициента α ее загрузки [1]. В свою очередь, скорость хеш-поиска характеризуется средним s_{ave} и максимальным s_{max} числом обращений к памяти, необходимыми для выполнения поиска. Необходимость разрешения коллизий увеличивает число обращений к памяти. Наиболее распространенным на практике способом разрешения коллизий является пробинг – то есть запись ключа X в первую свободную ячейку, адресуемую совокупностью кодов, получаемых в результате ряда хеш-преобразований: $h_1(X)$, $h_2(X)$, $h_3(X)$,… В простейших случаях функции ряда зависят от друг друга. Так, при линейном пробинге, каждый следующий адрес на единицу больше предыдущего: $h_i(X) = (h_1(X)+i) \bmod M$. Хотя такие функции просто вычисляются, при их использовании имеет место недостаток - вторичная группировка записей, которая существенно замедляет процесс поиска. При использовании специальных генераторов независимых хеш-функций [3] достигается теоретически наименьшее среднее число s_{ave} обращений к памяти для разрешения коллизий, которое определяется формулой:

$$s_{ave} = \frac{1}{1-\alpha} \qquad (1)$$

Основным недостатком известных технологий пробинга является то, что они не ограничивают максимальное число s_{max} обращений к памяти в процессе разрешения коллизий. Так, с вероятностью $p_s = \alpha^{s+1}$ при поиске может возникнуть необходимость в более, чем s обращениях к памяти.

Для ряда важных практических применений, связанных с обработкой информации в реальном времени, доминирующее значение имеет ограничение значения s_{max}.

Для совершенной хеш-адресации (то есть без коллизий) постоянных массивов ключей, основным критерием эффективности является зависимость времени подбора хеш-преобразования, не образующего коллизий от коэффициента α загрузки памяти.

Известно ряд технологий выполнения такого подбора [1,2,4], наиболее известными из которых являются методы Cichelli R.J. и Czech Z.J. Все они используют перебор функций хеш-преобразования. Для этих методов, среднее число проб T_α, требующееся для поиска perfect хеш-преобразования определяется следующей формулой [4]:

$$T_\alpha = e^m \cdot (1-\alpha)^{m\cdot(\frac{1-\alpha}{\alpha})+\frac{1}{2}} \qquad (2)$$

Как следует из формулы (2), основным недостатком известных методов получения совершенных хеш-преобразований для постоянных массивов ключей является большое число требуемых проб.

Целью исследований является ограничение числа требуемых обращений к хеш-памяти в процессе разрешения коллизий для динамических массивов ключей и сокращение времени получения хеш-преобразования с ограниченной длиной цепочки коллизий для постоянных массивов ключей.

Концепция хеш-памяти с ограничением коллизий

Для решения задачи ограничения времени хеш-поиска предлагается концепция хранения ключей только по одному из двух возможных хеш-адресов. Это позволяет ограничить длину цепочки возникающих коллизий и обеспечить поиск по ключу не более, чем за два обращения к накопителю.

Предлагаемая концепция предполагает организацию хеш-памяти в виде двух банков (или таблиц), обозначенных так T_1 (левый банк памяти) и T_2 (правый банк памяти), каждый из которых содержит $M/2$ ячеек памяти. Для каждого из ключей определяется один возможный хеш-адрес в банке T_1 и один возможный хеш-адрес в банке T_2. Для адресации ключа X в банках памяти используются две хеш-функции $h_1(X)$ для T_1 и $h_2(X)$ для T_2, такие, что пара значений $h_1(X)$ и $h_2(X)$ однозначно определяют ключ X.

Хранение ключа X в левом банке памяти является приоритетным. Соответственно, запись кода ключа X вначале выполняется в ячейку, адресуемую $h_1(X)$ левого банка T_1. Если она уже занята, то запись выполняется в ячейку с адресом $h_2(X)$ правого банка памяти T_2.

Поскольку коды хеш-адресов $h_1(X)$ и $h_2(X)$ ключа X однозначно определяют код самого ключа X, то в рамках предлагаемой концепции можно существенно сократить объем используемой памяти. Для этого, при размещении ключа X в ячейку $h_1(X)$ левого банка памяти T_1, в ней реально записывается код $h_2(X)$, который вместе с адресом $h_1(X)$ однозначно определяет значение ключа X. Аналогично, при размещении ключа X в правом банке T_2 в его ячейку с адресом $h_2(X)$ записывается код $h_1(X)$. Кроме указанных кодов, которые вместе с хеш-адресом однозначно определяют ключ, в ячейках хранится теговый бит занятости.

Поиск ключа в предлагаемой хеш-памяти не более, чем за два обращения к накопителю, достигается за счет того, что часть ключа, которая вместе с адресом её ячейки однозначно определяет ключ X, может храниться только в одной из двух "доступных для хранения" ячеек: по адресу $h_1(X)$ в банке T_1, либо по адресу $h_2(X)$ в банке T_2.

Алгоритм поиска по ключу X состоит в следующем:

1. Для заданного ключа X вычисляются значения хеш-функций $h_1(X)$ и $h_2(X)$

2. Считывание кода из ячейки, адресуемой $h_1(X)$ левого банка памяти. Если теговый бит считанного кода равен нулю (ячейка свободна), то поиск завершен с отрицательным результатом.

3. Сравнение считанного кода с $h_2(X)$. Если сравниваемые коды совпадают, то поиск завершен с положительным результатом.

4. Считывание кода из ячейки, адресуемой $h_2(X)$ правого банка памяти. Если теговый бит считанного кода равен нулю (ячейка свободна), то поиск завершен с отрицательным результатом.

5. Сравнение считанного кода с $h_1(X)$. Если сравниваемые коды совпадают, то поиск завершен с положительным результатом. В противном случае – поиск завершен с отрицательным результатом.

Выполнение условия хранения каждого из ключей в одной из двух ячеек хеш-памяти обеспечивает длину цепочки коллизий не больше двух, Соответственно, состояние хеш-памяти, для которого выполняется это условие, может быть названо состоянием ограниченных коллизий (СОТ).

Запись ключа X в хеш-памяти всегда осуществляется в одну из двух доступных для его хранения ячеек. Однако, может возникнуть ситуация, при которой обе эти ячейки хеш-памяти заняты другими, ранее записанными ключами.

Схематично такая ситуация показана на рис.1. Здесь через a ,b обозначены ячейки левого банка памяти T_1, а через v,z и w – ячейки правого банка T_2. До записи ключа X_4 в хеш-память записаны: ключ X_1, для которого $h_1(X_1)=a$, $h_2(X_1)=z$, записывается в ячейку a левого банка T_1. Ключ X_2, для которого допустимыми являются ячейки b и w ($h_1(X_2)=b$, $h_2(X_2)=w$) помещается в свободную ячейку b левого банка T_1. Для ключа X_3, ($h_1(X_3)=b$, $h_2(X_3)=v$), допустимыми являются ячейки b и v. Однако ячейка b левого банка T_1 уже занята ключом X_2, поэтому ключ X_3 помещается в ячейку v правого банка T_2. Ситуация после записи трех ключей X_1, X_2 и X_3 показана на рис.1.

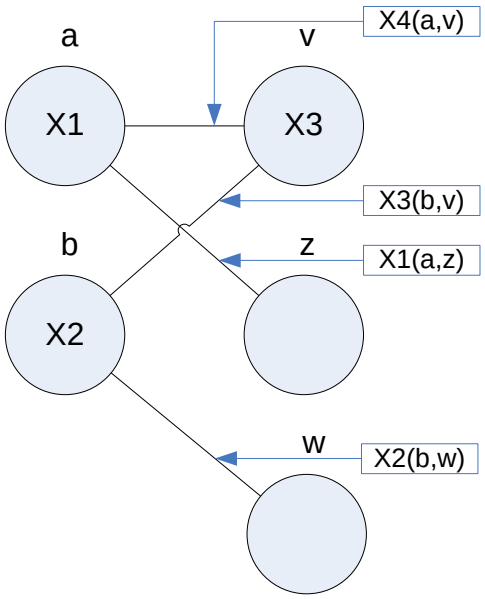

Рис. 1. Запись ключей X_1, X_2, X_3 в память

При записи ключа X_4, для которого $h_1(X_4)=a$ и $h_2(X_4)=v$, возникает ситуация, когда обе допустимых для него ячейки – a в банке T_1 и v в банке T_2 заняты.

Для разрешения такой ситуации предлагается использовать рекурсивное перемещение в памяти ранее записанных ключей с тем, чтобы освободить место для нового ключа.

В рамках рассматриваемого примера, при невозможности записи ключа X_4 производится анализ возможности перемещения ключа X_1, занимающего ячейку a, на которую претендует ключ X_4 в левом банке. Поскольку альтернативная для ключа X_1 ячейка z правого банка T_2

свободна, то ключ X_1 переписывается в нее (фактически из ячейки $a=h_1(X_1)$ считывается код $h_2(X_1)$, адресующий ячейку $z=h_2(X_1)$ правого банка памяти T_2, которую записывается код $a=h_1(X_1)$). После освобождения ячейки a, в нее помещается ключ X_4 (фактически в ячейку a записывается $h_2(X_4)$). Ситуация после записи ключа X_4 схематично показана на рис.2.

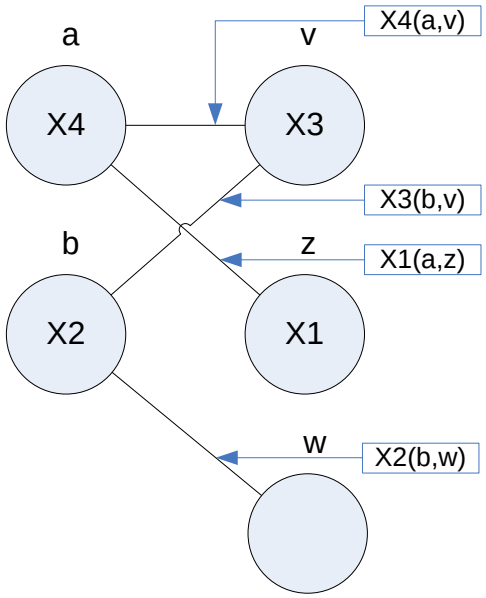

Рис. 2. Запись ключа X_4 в память

Для выполнения рекурсивной записи ключей в хеш-память предлагается формализованный алгоритм. Для его описания используется переменная t ($t \in \{1,2\}$), которая соответствует номеру банка памяти, с которым осуществляется работа в текущий момент времени. Для работы алгоритма необходимо организовать стек, в котором размещаются записываемые и перемещаемые в процессе рекурсии ключи. Каждая из ячеек стека содержит два поля *L, R* для записи адресов в банках хеш-памяти и однобитовое поле *nb* для сохранения номера банка памяти.

Алгоритм записи ключа X сводится к выполнению последовательности действий:

1. Вычисляются значения хеш-функций $h_1(X)$ и $h_2(X)$, устанавливается значение t=1.

2. Из ячейки по адресу $h_1(X)$ в левом банке памяти T_1 считывается код d ссылки и теговый бит β. Если теговый бит β равен нулю: β=0, то есть ячейка не занята, то в нее записывается код $h_2(X)$ и теговый бит β равный единице. Операция записи заканчивается. Иначе переход на пп.4.

3. Из ячейки по адресу $h_2(X)$ правого банка памяти T_1 считывается код g ссылки и теговый бит β. Если теговый бит β равен нулю: β=0, то есть ячейка не занята, то в нее записывается код $h_1(X)$ и теговый бит β равный единице. Операция записи заканчивается.

4. Коды $h_1(X)$, $h_2(X)$ и t помещаются в поля *L,R, nb* стека. Если код d совпадает с одним из кодов, размещенных в поле R стека, то выполнить запись невозможно.

5. Модифицируется код $h_2(X)$: $h_2(X)=d$. Значение переменной t изменяется: $t = 3-t$.

6. Из ячейки по адресу $h_2(X)$ правого банка памяти T_2 считывается код g ссылки и теговый бит β. Если теговый бит β равен нулю: β=0, то есть ячейка не занята, то в нее записывается код $h_1(X)$ и теговый бит β равный единице. Переход на пп.10.

7. Коды $h_1(X)$, $h_2(X)$ и t помещаются в поля *L,R, nb* стека. Если код g совпадает с одним из кодов, размещенных в поле L стека, то выполнить запись невозможно.

8. Модифицируется код $h_1(X)$: $h_1(X)=g$. Значение переменной t изменяется: $t = 3-t$.

9. Из ячейки по адресу $h_1(X)$ считывается код d. Переход на пп.5.

10. Из полей *L,R* и *nb* стека выталкиваются коды соответственно $h_1(X)$, $h_2(X)$ и t.

11. Если t=1, код $h_2(t)$ и единичный бит β записывается по адресу $h_1(t)$ в левый банк памяти T_1. Переход на пп. 13.

12. Если t=2, код $h_1(t)$ и единичный бит β записывается по адресу $h_2(t)$ в правый банк памяти T_2.

13. Если стек не пуст, значение переменной t изменяется: $t = 3-t$ и возврат на повторное выполнение пп.8. Иначе конец операции записи.

Количество обращений к накопителю при выполнении описанной рекурсивной процедуры записи ключа является случайным числом, зависящим от коэффициента α загрузки памяти. Средне значение числа T_W обращений к памяти при записи определяется в виде:

$$T_W = 2\cdot(1-\alpha) + \sum_{j=1}^{\infty} \alpha^j \cdot (1-\alpha)\cdot(2\cdot j+1) = \frac{2}{1-\alpha} - 1 \qquad (3)$$

Анализ формулы (3) показывает, что для используемых на практике значений α: 0.7≤ α

≤ 0.9 время записи по сравнению с обычным пробингом увеличивается практически вдвое.

Важным для исследования эффективности предлагаемой концепции представляется получение оценки вероятности возникновения коллизии, как ситуации, в которой запись нового ключа с сохранением СОК невозможна.

Оценка вероятности возникновения коллизий

Для каждого из ключей формируется два хеш-адреса. Возможна редкая ситуация, когда для ограниченное подмножество Ω из *w* ключей число формируемых хеш-адресов будет меньше *w*. В этой ситуации, размещение ключей подмножества Ω согласно предложенной концепции является принципиально невозможным. То есть, существует возможность возникновения коллизий. Пример такой ситуации схематически показан на рис.3. Кружочками на этом рисунке показаны ячейки левого и правого банков памяти, а дуги – соответствуют ключам подмножества $\Omega=\{X_1,X_2,\ldots,X_6\}$, $h_1(X_1)=a$, $h_2(X_1)=v$, $h_1(X_2)=a$, $h_2(X_1)=z$; $h_1(X_3)=b$, $h_2(X_3)=v$; $h_1(X_4)=b$, $h_2(X_4)=z$; $h_1(X_5)=c$, $h_2(X_5)=v$; $h_1(X_6)=c$, $h_2(X_6)=z$.

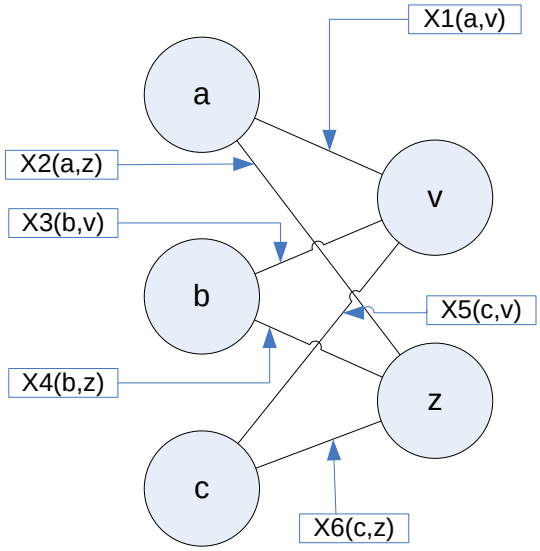

Рис. 3. Пример коллизии при размещении 6-ти ключей в 5-ти ячейках

В ситуации, показанной на рис.3, коллизия получается при размещении 6-ти ключей. При этом, в правом банке занятыми оказываются только две ячейки *v* и **z**, причем существуют ключи, адресующие каждую из упомянутых ячеек и каждую из 3-х ячеек *a,b,c* левого банка памяти. Общее число вариантов ключей при этом равно 6-ти. Очевидно, что для подмножества Ω, содержащего меньше 6-ти ключей коллизия невозможна. Действительно, если предположить, что в левом банке используется только одна ячейка, то общее число ключей, адресующих эту ячейку и *u* ячеек левого банка памяти не превысит *u*, в то время, как суммарное число ячеек составит *u*+1.

Вместе с тем, возможны и другие варианты коллизий, при которых множество Ω содержит более 6-ти ключей. Один из таких вариантов, при котором множество Ω включает 9 ключей, которые должны быть размещены в 6-ти ячейках, показан на рис.4.

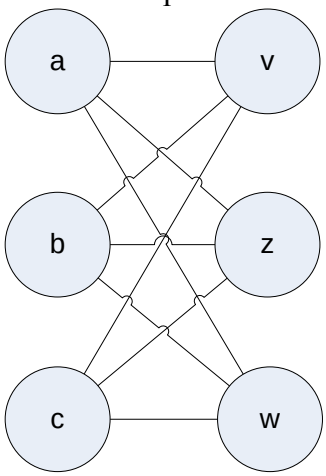

Рис. 4. Пример коллизии при размещении 9-ти ключей в 6-ти ячейках

Важным аспектом исследования эффективности предложенной организации хеш-доступа является получение оценки вероятности возникновения коллизий.

Пусть в хеш-памяти размещается *m* ключей, а общее количество ячеек в обоих банках равно *M*. Аналитическая оценка вероятности p_6 возникновения ситуации, показанной на рис.3, при которой для множества из 6-ти ключей формируется только 5 адресов, может быть выполнена следующим образом.

Пусть фиксируются две ячейки правого банка хеш-памяти, обозначенные как *v* и *z*. Если $\alpha>0.5$, то число адресных ссылок *m* ключей на правый банк памяти превышает количество *M*/2 его ячеек, то есть $m>M/2$. Это означает, что с вероятностью близкой к единице имеется ключ *X*, с адресной ссылкой *v* для правого банка памяти: $h_2(X)=v$. Адресная ссылка этого ключа *X* в левом банке адресует некоторую ячейку a, то есть $h_1(X)=a$. Аналогично, с вероятностью, близкой к единице существует ключ Y, такой, что $h_2(Y)=z$ и $h_1(Y)=b$.

Если выбрать фиксированную ячейку *c* левого банка памяти, то вероятность того, что

существует ключ G, для которого $h_1(G)=c$ и $h_2(G)=v$ составляет M^{-1}. Вероятность того, что фиксированный ключ имеет ссылки на два фиксированных в левом и правом банках хеш-памяти адреса составляет M^{-2}. Поэтому, если фиксировать три ключа Q_1,Q_2 и Q_3, то вероятность того, что все они имеют заранее определенные адресные ссылки (в частности, для первого Q_1 из них: $h_1(Q_1)=a$ и $h_2(Q_1)=z$; для второго Q_2 : $h_1(Q_2)=b$ и $h_2(Q_2)=v$; а для третьего Q_3 из них: $h_1(Q_3)=c$, $h_2(Q_3)=z$) составляет M^{-6}. Таким образом, вероятность того, что возникнет ситуация, показанная на рис.3 для трех фиксированных ключей Q_1,Q_2 и Q_3 и трех фиксированных ячеек: c, v, z составляет M^{-7}. Исходя из этого, ситуация, показанная на рис.3 не возникает, если она не имеет места для всех возможных вариантов выбора упомянутых трех ключей и всех возможных вариантов выбора ячеек c и v, z. Количество вариантов выбора трех ключей Q_1,Q_2 и Q_3 равно:

$$\frac{m!}{3!\cdot(m-3)!}=\frac{m\cdot(m-1)\cdot(m-2)}{6}$$

Для больших m, приближенно равно $m^3/6$.

Число вариантов выбора ячейки c равно M, а количество вариантов выбора пары ячеек v и z для больших M приближенно равно $M^2/2$. С учетом этого, вероятность q_6 того, что ситуация, показанная на рис.3 не возникает определяется выражением:

$$q_6=(1-M^{-7})^{\frac{m^3\cdot M^3}{12}}$$

Вероятность p_6 того, что при хеш-адресации множества из m ключей для 6-ти из них возникнет коллизийная ситуация, показанная на рис.3 определяется в виде:

$$p_6=1-q_6=1-(1-M^{-7})^{\frac{m^3\cdot M^3}{12}} \qquad (4)$$

Для имеющих место на практике больших значений m и M формула (4) может быть упрощена путем раскрытия скобок и учета только первых двух членов биномиального ряда, поскольку значения всех остальных составляют не более 5% второго члена ряда:

$$p_6\approx 1-(1-\frac{m^3\cdot M^3}{12}\cdot M^{-7})=\frac{m^3}{M^4\cdot 12}=\frac{\alpha^3}{12\cdot M} \qquad (5)$$

Проведенные экспериментальные исследования, в целом, подтвердили справедливость теоретической оценки (5) для вероятности p_6 коллизий для подмножества из 6-ти ключей.

Так, проведя аналогичные выкладки для ситуации показанной на рис.4, когда коллизия образуется при адресации 9-ти ключей, можно показать, что вероятность p_9 оценивается следующим выражением:

$$p_9\approx\frac{\alpha^3}{36\cdot M^3} \qquad (6)$$

Сравнение выражений (5) и (6) показывает, что для имеющих место на практике больших значений M: $p_6>>p_9$. Отсюда можно сделать вывод, что вероятность коллизии, возникающей при размещении в хеш-памяти подмножества ключей, быстро падает с увеличением числа ключей этого подмножества. Это означает, что с точностью, достаточных для оценочных расчетов, вполне можно полагать, что коллизии могут быть образованы только подмножеством из 6-ти ключей, вероятность которой оценивается полученной выше формулой (5).

Оценка эффективности применения для динамических и постоянных ключей

Предложенная концепция хеш-адресации ключей с ограничением адресов их размещения может быть использована для повышения эффективности как динамической, так и статической хеш-памяти.

Важнейшей характеристикой эффективности является время поиска, которое определяется средним s_{ave} и максимальным s_{max} числом обращений к накопителю.

Как уже отмечалось, использование предложенного подхода позволяет принципиально ограничить максимальное время поиска двумя обращениями к памяти. То есть, при использовании предложенного подхода $s'_{max}=2$. Как указывалось выше, при использовании пробинга, вероятность p_s того, что потребуется не менее s обращений к памяти определяется как $p_s=\alpha^{s+1}$. Например, при типичном на практике значении коэффициента загрузки $\alpha=0.75$ при использовании пробинга с вероятностью 0.01 потребуется для поиска 28 обращений к памяти. То есть, для такой ситуации, применение предложенного подхода позволяет уменьшить максимальное время поиска в 14 раз.

Среднее число s'_{ave} обращений к накопителю зависит от коэффициентов загружен-

ности каждого из банков хеш-памяти и определяется формулой:

$$s'_{ave} = \frac{\alpha_L + 2 \cdot \alpha_R}{\alpha_L + \alpha_R} \qquad (7)$$

где α_L – коэффициент загрузки левого банка памяти, а α_R – коэффициент загрузки правого банка.

Экспериментально установлено, что при малых значения α ($\alpha < 0.7$) более загруженным оказывается левый банк хеш-памяти, соответственно, среднее число обращений к накопителю при поиске меньше 1.5. Так, при $\alpha=0.6$ коэффициент α_L загрузки левого банка памяти составляет 0.7, в то время как для правого банка $\alpha_R = 0.5$. Соответственно, при этом, среднее число s'_{ave} обращений к накопителю при поиске, в соответствии с формулой (7) составляет 1.42.

При $\alpha > 0.7$ левый и правый банки памяти заполняются практически одинаково и среднее число обращений к памяти при поиске практически равно 1.5.

Таким образом, сопоставление значений среднего числа обращений к памяти при поиске для предлагаемой концепции s'_{ave} и традиционного пробинга s_{ave} позволяет сделать следующий вывод. При $\alpha > 0.4$ (то есть практически всегда) справедливо $s'_{ave} < s_{ave}$, это означает, что применение предложенной концепции позволяет уменьшить среднее число обращений к памяти при поиске, то есть ускорить поиск. Например, при наиболее типичном значении коэффициента загрузки $\alpha=0.75$ использование предложенного подхода позволяет в 2.7 раз уменьшить время поиска по сравнению с пробингом.

Следует указать, что повышение скорости поиска достигается за счет замедления процесса записи ключей. Поэтому, реальная эффективность применения предложенного подхода может быть достигнута только для класса задач, в которых интенсивность операций поиска существенно выше интенсивности операций модификации поискового массива.

При реализации предложенной концепции для хеш-адресации динамических массивов ключей необходимо предусмотреть механизм разрешение коллизий – то есть ситуации, при которой существует подмножество Ω ключей, хеш-адреса которых образуют замкнутый класс и их число меньше, чем количество ключей, составляющих множество Ω.

В качестве такого механизма предлагается использовать дополнительную память переполнения малого объема. В такой памяти предполагается размещать ключи, которые не могут быть размещены в банках хеш-памяти по своим хеш-адресам. Для того, чтобы реализовать переход к дополнительной памяти переполнения необходимо в ячейках основной памяти, помимо тегового бита занятости, выделить второй теговый бит перехода к дополнительной памяти.

Применительно к хеш-поиску в постоянных массивах ключей использование предложенного подхода позволяет значительно сократить время получения хеш-преобразования, допускающего размещение ключа в одной из двух альтернативных ячейках хеш-памяти.

Для получения такого хеш-преобразований случайно выбираются две хеш-функции $h_1(X)$ и $h_2(X)$. Массив ключей в соответствии с описанной выше рекурсивной процедурой записывается в банки хеш-памяти. Если в процессе записи возникнет коллизия – то есть невозможность размещения ключей в соответствии с предложенной концепцией, то хеш-функции $h_1(X)$ и $h_2(X)$ меняются, после чего производится повторная попытка размещения заданного массива постоянных ключей. Среднее число T_p выбора пары хеш-функций $h_1(X)$ и $h_2(X)$ исходя из (5) составит:

$$T_p = \sum_{j=1}^{\infty} j \cdot \frac{\alpha^{3\cdot(j-1)}}{12^{j-1} \cdot M^{j-1}} \cdot \left(1 - \frac{\alpha^3}{12 \cdot M}\right) = \frac{1}{1 - \frac{\alpha^3}{12 \cdot M}} \qquad (8)$$

Сопоставление формул (2) и (8) показывает, что применение предложенного подхода к хеш-адресации постоянных массивов ключей позволяет на несколько порядков ускорить подбор подходящего хеш-преобразований. Однако, необходимо иметь ввиду, что указанный значительный выигрыш во времени формирования хеш-преобразования достигнут за счет того, что при использовании предложенного подхода, в отличие от совершенной хеш-адресации, допускается два (а не одно) обращений к накопителю в процессе поиска, то есть за счет снижения скорости поиска в полтора раза.

Принимая во внимание, что число обращений к памяти, требующее для записи одного ключа определяется формулой (3), общее число N_w обращений к памяти, требующееся для размещения постоянного массива ключей в хеш-памяти определяется следующей формулой:

$$N_w = \frac{m}{1-\frac{\alpha^3}{12 \cdot M}} \cdot \left(\frac{2}{1-\alpha} - 1\right) \qquad (9)$$

Анализ формулы (9) показывает, что использование предложенного подхода для хеш-адресации постоянных массивов ключей эффективнее существующих методов совершенной хеш-адресации при больших значениях коэффициента α загрузки памяти и при большом числе размещаемых в хеш-памяти ключей. Практическая значимость предлагаемого подхода для совершенной хеш-адресации состоит в том, что при его использовании может быть получено решение задачи, недостижимое при применении известных методов, а именно: нахождение хеш-преобразований, которые гарантируют размещение ключей большого постоянного массива ключ по одному из двух альтернативных адресов, с коэффициентом использования памяти, близким к единице.

Выводы

В результате проведенных исследований предложена концепция организации хеш-памяти с ограниченным временем поиска по ключу. Разработаны технологические аспекты этой концепции, проведен теоретический и экспериментальный анализ ее эффективности.

Основной фактор эффективности предложенной концепции состоит в ограничении числа обращений в памяти при поиске за счет усложнения процедуры записи ключей. Доказано, что это позволяет в 2-3 раза уменьшить среднее время поиска по ключу и на порядок – максимальное время в сравнении с использованием пробинга. Это достигается за счет соответствующего увеличения времени записи ключа. Поэтому использование предложенного подхода эффективно при условии когда интенсивность операций поиска по ключу превышает интенсивность операций модификации поискового массива.

Исследованиями показано, что предложенный подход может быть использован не только для повышения скорости поиска в динамических массивах ключей, но и для многократного ускорения в сравнении с известными методами, подбора эффективных хеш-преобразований для постоянных массивов ключей.

Список литературы

1. Кохонен Т. Ассоциативная память.-М.:Мир,1980.- 198 С.
2. Марковский А.П., Гаваагийн Улзисайхан, Бардис Николас. Об одном подходе к повышению эффективности и уровня защищенности систем хранения информации на основе хеш-памяти//Вісник НТУУ “КПІ”. Інформатика, управління та обчислювальна техніка.- 1998,- № 31,- С.14-23.
3. Салех Ибрагим Аль-Омар. Использование генераторов булевых функций для повышения эффективности хеш-памяти. // Вісник Національного технічного університету України ”КПІ”. Інформатика, управління та обчислювальна техніка.- 2003.- № 40.- С.131-140.
4. Czech Z.J., Havas G., Majevski B.S.. An Optimal algorithm for generating minimal perfect hash functions.//Information processing letters. – 1997. – Vol.43.- № 5. - P.257-264.