УДК 681.3.06

ЭФФЕКТИВНЫЙ АЛГОРИТМ ЗАМЕЩЕНИЯ СТРАНИЦ ДЛЯ БУФЕРИЗАЦИИ ОБМЕНОВ С ДИСКАМИ В ПАРАЛЛЕЛЬНОЙ СИСТЕМЕ БАЗ ДАННЫХ БЕЗ СОВМЕСТНОГО ИСПОЛЬЗОВАНИЯ РЕСУРСОВ

Л. Б. Соколинский[1]

В работе предлагается новый алгоритм замещения страниц **LFU–*K*** для буферизации обменов с дисками, ориентированный на использование в параллельных системах баз данных без совместного использования ресурсов. Данный алгоритм является обобщением хорошо известного алгоритма **LFU**. Для предложенного алгоритма **LFU–*K*** вводится формальная теоретико-вероятностная модель, на базе которой получены аналитические оценки параметров данного алгоритма. На базе алгоритма **LFU–2** строится некоторый его модернизированный вариант **LFU–2m**, пригодный для использования в реальных системах баз данных. Приводятся результаты вычислительных экспериментов над искусственными и реальными трассами обращений к диску, подтверждающие высокую эффективность алгоритма **LFU–2m** применительно к параллельным системам баз данных без совместного использования ресурсов.

Ключевые слова: параллельные системы баз данных, управление буферным пулом, алгоритмы замещения страниц, буферизации обменов, анализ эффективности.

1. Введение. В любой системе баз данных при обращении к странице диска образ страницы загружается в буфер, располагающийся в оперативной памяти, и сохраняется там в течение некоторого времени после обращения. Это делается для того, чтобы страницы диска, обращения к которым повторяются очень часто, оставались в оперативной памяти постоянно. Тем самым сокращается количество обменов с дисками. Учитывая, что обращение к диску по крайней мере в 1000 раз медленнее, чем обращение к оперативной памяти [12], мы можем существенным образом повысить производительность системы.

Для эффективной буферизации критическим является вопрос: какую страницу мы должны вытеснить из буфера при нехватке места для "подкачки" новой страницы диска? Правила, которые мы будем использовать для решения этой задачи, называются *стратегией замещения*. Алгоритм, реализующий некоторую стратегию замещения, называется *алгоритмом замещения* [7]. Мерой *эффективности* стратегии замещения является количество возникающих *промахов*, то есть ситуаций, когда выбираемая страница не находится в буфере. Билейди (Bilady) показал в [8], что теоретический максимум эффективности определяется стратегией **OPT**: в соответствии с этой стратегией из буфера вытесняется страница, к которой дольше всего не будет обращений. Однако для реализации стратегии **OPT** нам нужен *оракул*, который мог бы сколь угодно далеко заглядывать в будущее, поэтому на практике реализовать стратегию **OPT**, как правило, не удается.

Вплоть до середины 80-х годов для управления буферным пулом практически во всех случаях использовалась стратегия замещения **LRU** (*Least Recently Used*) [16], вытесняющая из буфера ту страницу, к которой дольше всего не было обращений. В работе [22] Слейтор (Sleator) и Тарьян (Tarjan) показали, что количество промахов F_{LRU} стратегии **LRU** ограничено сверху следующим образом:

$$F_{\mathrm{LRU}} \leqslant kF_{\mathrm{OPT}}. \tag{1}$$

Здесь k — количество страниц, помещающихся в буфер, а F_{OPT} — количество промахов стратегии **OPT**. Однако дальнейшие исследования показали, что стратегия **LRU** далеко не всегда адекватно учитывает специфику систем баз данных [9, 21, 25]. В соответствии с этим были предприняты интенсивные усилия по поиску алгоритмов замещения, которые в большей степени подходили бы для систем баз данных (см., например, работы [9, 14, 15, 17, 20, 21, 23]).

Одним из наиболее впечатляющих достижений в этом направлении является алгоритм **LRU–*K***, предложенный Элизабет О'Нил (Elizabeth O'Neil), Патриком О'Нилом (Patrick O'Neil) и Герхардом Вейкумом (Gerhard Weikum) в работе [17]. Данный алгоритм является обобщением алгоритма **LRU** в том смысле,

[1] Челябинский государственный университет, математический факультет, ул. Бр. Кашириных, д. 129, 454021, г. Челябинск; e-mail: sokolinsky@acm.org

что алгоритм **LRU–1** совпадает с алгоритмом **LRU**. Однако уже при $K = 2$ алгоритм **LRU–*K*** способен показать существенно более высокую эффективность по сравнению с алгоритмом **LRU** для вариантов загрузки, характерных при работе реляционных систем баз данных. Для объяснения принципа работы алгоритма **LRU–*K*** введем понятие трассы. *Трасса* — это последовательность номеров страниц диска, к которым производятся последовательные по времени обращения:

$$r_1, r_2, r_3, \dots \ . \tag{2}$$

При этом меньшее значение индекса соответствует более позднему по времени обращению. Для определения "жертвы" алгоритм **LRU–*K*** для каждой страницы i, хранящейся в буфере, находит K-е по счету вхождение данной страницы в последовательность (2). Везде далее мы будем отождествлять страницу с ее номером (адресом) на диске. Пусть этому вхождению соответствует элемент $r_{b_{iK}}$. Из буфера вытесняется та страница, которая имеет максимальное значение b_{iK}. Если страница с номером i имеет менее K вхождений в последовательность (2), то полагают $b_{iK} = \infty$.

В отличие от алгоритма **LRU**, алгоритм **LRU–*K*** (при $K > 1$) всегда присваивает более высокий приоритет страницам, обращение к которым происходит циклически (по сравнению со страницами, доступ к которым происходит однократно или очень редко). Данная ситуация характерна для выборки записей по случайному ключу из очень большой таблицы с использованием индексов в виде B–дерева [17]. Очевидно, что обращение к узлам верхних уровней B–дерева в данном случае будет происходить циклически. Если длина циклов превышает длину буфера, то алгоритм **LRU** не в состоянии распознать подобные циклы. В соответствии с этим для вариантов загрузки, характерных для OLTP [3], эффективность **LRU–2** может более чем на 40 % превышать эффективность **LRU** для больших реляционных баз данных [17]. В работе [18] доказывается оптимальность алгоритма **LRU–*K*** в том смысле, что не существует более эффективного алгоритма замещения при условии, что для определения "жертвы" используется не больше знаний о трассе (2), чем использует алгоритм **LRU–*K***. В работе [14] был предложен алгоритм замещения **2Q**, использующий те же принципы, что и **LRU–2**, но допускающий реализацию с меньшими накладными расходами.

Алгоритм **LRU–*K*** изначально был разработан для многозадачной среды, предусматривающей одновременное выполнение многих транзакций в режиме разделения времени. В такой среде распределение вероятностей обращений к страницам диска имеет динамический, трудно прогнозируемый характер. Подобный же профиль буферизации имеют и параллельные системы баз данных с разделяемой памятью и дисками. Это объясняется тем, что все процессоры разделяют единый буферный пул (использование схем, предусматривающих дробление буферного пула по числу процессоров, ведет к снижению эффективности использования оперативной памяти [17]). Однако для параллельных систем баз данных без совместного использования ресурсов характерен принципиально иной профиль использования буферного пула. В подобных системах оптимальным является профиль загрузки, при котором на каждом узле системы в каждый момент времени выполняется небольшое количество реляционных операций, являющихся частью одного запроса [19, 26]. При такой загрузке распределение вероятностей обращений к страницам диска будет иметь статический характер на протяжении некоторого ограниченного интервала времени, совпадающего с временем выполнения *фрагмента* запроса, назначенного данному процессорному узлу. Затем узлу будет назначен новый фрагмент другого запроса, что повлечет практически одномоментную смену распределения вероятностей, после чего снова наступит период некоторой стабильности. В данных условиях использование алгоритма **LRU–*K*** оказывается не вполне адекватным. Действительно, после смены текущего распределения в течение некоторого времени алгоритм **LRU–*K*** будет определять рейтинги страниц, по инерции исходя из старого распределения. Если период адаптации алгоритма **LRU–*K*** к новому распределению окажется по длительности близким к периоду выполнения фрагмента запроса, мы будем наблюдать существенное снижение эффективности алгоритма **LRU–*K***.

В данной работе нами предлагается новый алгоритм замещения **LFU–*K***, ориентированный, прежде всего, на использование в параллельных системах баз данных без совместного использования ресурсов. Алгоритм **LFU–*K*** является обобщением известного алгоритма **LFU** (*Least Frequently* Used), вытесняющего из буфера страницу с наименьшим количеством обращений. В том случае, когда вероятность обращения к данной странице остается постоянной во времени, алгоритм **LFU** показывает великолепные результаты, близкие к оптимальным. Однако, в случае изменения данной вероятности, страницы, к которым в текущий период времени не производится обращений, могут по-прежнему удерживаться в буфере за счет большого числа ранее накопленных обращений. В отличие от **LFU** алгоритм **LFU–*K*** подсчитывает количество обращений на отрезке трассы фиксированной длины m (**LFU–0**). Полученное значение может уточняться путем подсчета "*скорости*" (**LFU–1**) и "*ускорения*"(**LFU–2**) роста числа обращений к данной странице. Как будет показано в данной работе, алгоритм **LFU–*K*** во всех случаях не уступает по

эффективности известным алгоритмам, однако для вариантов загрузки, характерных для параллельных систем баз данных без совместного использования ресурсов, алгоритм **LFU–*K*** демонстрирует значительное превосходство в эффективности по сравнению с известными алгоритмами, такими как **LRU–*K*** и **2Q**.

Оставшаяся часть статьи организована следующим образом. В разделе 2 на базе производящих функций строится формальная модель для описания алгоритма **LFU–*K***. В разделе 3 на основе данной модели путем использования теоретико-вероятностных методов строится аналитическая оценка для параметра m в зависимости от характера распределения вероятностей обращений к страницам диска. В разделе 4 рассматриваются практические аспекты реализации алгоритма **LFU–*K*** и предлагается конкретная реализация модернизированной версии алгоритма **LFU–2m** на псевдокоде. В разделе 5 приводятся результаты экспериментов по сравнительной оценке эффективности алгоритма **LFU–2m** для различных искусственных и реальных трасс. Раздел 6 посвящен вопросам настройки параметров алгоритма **LFU–2m**. В заключении суммируются полученные результаты и определяются направления дальнейших исследований.

2. Формальное описание алгоритма LFU–*K*. Пусть N — количество кэшируемых страниц диска. Пусть последовательность

$$r_1, r_2, \ldots, r_M \tag{3}$$

задает некоторую трассу обращений к страницам диска, $M \geqslant 1$. При этом меньшее значение индекса соответствует более позднему по времени обращению. Пусть задано некоторое целое m, $1 \leqslant m \leqslant M$. Для некоторой фиксированной страницы i $(1 \leqslant i \leqslant N)$ рассмотрим последовательность

$$k_{i1}, k_{i2}, \ldots, k_{im}. \tag{4}$$

Здесь $k_{ij} = \delta_{ir_j}$ для всех j, $1 \leqslant j \leqslant m$ (δ_{in_j} — символ Кронекера [1]). Пусть $F_{il}(z)$ — производящая функция для подпоследовательности $k_{il}, \ldots, k_{im}$ последовательности (4), $1 \leqslant l \leqslant m$. По определению производящей функции имеем

$$F_{il}(z) = \sum_{j=l}^{m} k_{ij} z^j. \tag{5}$$

Пусть задано некоторое целое h, $1 \leqslant h \leqslant m$. Определим

$$F_{il}^{[0]}(z) = F_{il}(z), \quad F_{il}^{[n]}(z) = F_{il}^{[n-1]}(z) - F_{i,l+h}^{[n-1]}(z)$$

для произвольного целого $n > 0$. Положим

$$F_i^{[n]}(z) = F_{i1}^{[n]}(z).$$

Обозначим $t = m/h$ (без существенного ограничения общности мы можем полагать, что m всегда кратно h). Определим

$$R_{\mathrm{LFU-K}}(i) = \sum_{n=0}^{K} F_i^{[n]}(1)\,\frac{t^n}{n!} \tag{6}$$

для произвольного целого $K \geqslant 0$. В частности, имеем

$$R_{\mathrm{LFU}-0}(i) = F_i(1) = F_i^{[0]}(1), \tag{7}$$

$$R_{\mathrm{LFU}-1}(i) = F_i^{[0]}(1) + F_i^{[1]}(1)\,t, \tag{8}$$

$$R_{\mathrm{LFU}-2}(i) = F_i^{[0]}(1) + F_i^{[1]}(1)\,t + F_i^{[2]}(1)\,\frac{t^2}{2}. \tag{9}$$

Очевидно, что формула (7) подсчитывает число вхождений страницы i в последовательность

$$r_1, r_2, \ldots, r_m. \tag{10}$$

Если предположить, что вероятность обращения к странице i не меняется со временем [2] , то формула (7) может быть использована для оценки количества вхождений страницы i в последовательность очередных m обращений к страницам диска:

$$r_{(-m+1)}, \ldots, r_{(-1)}, r_0. \tag{11}$$

[2]Все временные интервалы мы здесь измеряем в количестве последовательных обращений к страницам диска (вне зависимости от того, находится искомая страница в буфере или нет).

Если же вероятность обращения к странице i изменяется со временем, то формула (8) может дать более точную оценку количества вхождений страницы i в последовательность (11), так как второе слагаемое в формуле (8) как раз содержит фактор *скорости* изменения частоты обращений к странице i. При этом очевидно, что точность оценки будет зависеть от величины параметров m и h.

Аналогичным образом формула (9) может дать еще более точную оценку за счет фактора *ускорения* в третьем слагаемом, если фактор скорости сам изменяется во времени. Таким образом, в общем случае мы приходим к формуле (6), которая дает нам следующее формальное определение алгоритма **LFU–*K***.

Формальное определение алгоритма* LFU–*K. Для каждой страницы диска, находящейся в буфере, вычисляется значение функции $R_{\mathrm{LFU-K}}$. Замещается страница, имеющая минимальное значение $R_{\mathrm{LFU-K}}$. Если таких страниц несколько, замещается та из них, которая дольше всего находилась в буфере. При этом алгоритм имеет два настраиваемых параметра m и h, удовлетворяющих ограничениям $1 \leqslant h \leqslant m$ и $1 \leqslant m \leqslant M$.

Очевидно, что при $m = M$ алгоритм **LFU–0** совпадает с известным алгоритмом **LFU**. Однако для определенных вариантов загрузки алгоритм **LFU–*K*** при $K \geqslant 1$ может показывать значительно более высокую эффективность по сравнению с **LFU** и другими известными алгоритмами.

Эффективность алгоритма **LFU–*K*** определяющим образом зависит от значений параметров m и h. Получение аналитических оценок для оптимального выбора указанных параметров в общем случае представляется нетривиальной задачей. В следующем разделе мы дадим аналитическую оценку параметра m для различных распределений вероятностей обращений к страницам диска. Проблема подбора оптимальных значений параметра h будет рассмотрена в разделе 6.

3. Аналитическая оценка параметра m. Для получения аналитической оценки параметра m построим вероятностную модель процесса обращения к страницам диска в некоторой абстрактной системе баз данных. Наша модель исходит из предположений, лежащих в основе модели IRM (*Independent Reference Model*) из [10].

3.1. Вероятностная модель. Пусть N — количество кэшируемых страниц. Определим p_i как вероятность обращения к странице i, $1 \leqslant i \leqslant N$. По определению имеем

$$\sum_{i=1}^{N} p_i = 1.$$

Предположим, что вероятности p_i *не меняются во времени* и распределяются по следующему закону [2, с. 434]:

$$p_i = \frac{1}{i^{\Theta} H_N^{(\Theta)}}. \tag{12}$$

Здесь $H_N^{(s)}$ — N-е гармоническое число порядка s, то есть $1^{-s} + 2^{-s} + \cdots + N^{-s}$, а Θ — коэффициент перекоса, $0 \leqslant \Theta \leqslant 1$. При $\Theta = 0$ получаем $p_i = 1/N$, то есть перекос отсутствует, что соответствует случаю равномерного распределения. Значение $\Theta = 1 - \log 0.80 / \log 0.20$ соответствует правилу “80–20” [13]. При $\Theta = 1$ получаем

$$p_i = \frac{1}{i^1 H_N^{(1)}} = \frac{1/H_N}{i},$$

что соответствует закону Зипфа (Zipf) [27].

Для гармонических чисел порядка r известно следующее приближение [5]:

$$H_n^{(r)} = \zeta(r) + \frac{n^{1-r} - 1}{1 - r} + \frac{1}{2n^r} - \sum_{k=1}^{m} \frac{B_{2k}}{2k n^{r+2k-1}} \binom{-r}{2k-1} + \mathrm{O}\left(\frac{1}{n^{r+2m+1}}\right). \tag{13}$$

Здесь $\zeta(r)$ — дзета-функция Римана, а B_k — числа Бернулли [1]. Из (13) мы можем получить упрощенную формулу

$$H_n^{(r)} = \zeta(r) + \frac{n^{1-r} - 1}{1 - r} + \mathrm{O}\left(\frac{1}{n^r}\right). \tag{14}$$

Используя приближение (14), из (12) получаем

$$p_i \approx \frac{1}{i^{\Theta}\left(\zeta(\Theta) + \dfrac{N^{1-\Theta} - 1}{1 - \Theta}\right)}. \tag{15}$$

В частности, вероятность обращения к самой "популярной" в нашей модели странице 1 может быть найдена по следующей приближенной формуле:

$$p_1 \approx \frac{1}{\zeta(\Theta) + \dfrac{N^{1-\Theta} - 1}{1 - \Theta}}. \tag{16}$$

Исследуем влияние параметра m на эффективность алгоритма **LFU–*K*** в контексте описанной модели. Прежде всего заметим, что в данном контексте $\lim\limits_{m\to\infty} F_i^{[n]} = 0$, для $n > 0$ и $h = m$. Поэтому нам достаточно исследовать алгоритм **LFU–0**. Имеем

$$\lim_{m\to\infty} \frac{R_{\mathrm{LFU}-0}(i)}{m} = p_i.$$

Это означает, что при увеличении значения параметра m эффективность алгоритма **LFU–0** будет приближаться к эффективности алгоритма **A0**, вытесняющего из буфера страницу с минимальной вероятностью обращения. В работе [7] было доказано, что алгоритм **A0** является оптимальным для статического распределения вероятностей. Отсюда следует, что увеличение значения m будет вести к повышению эффективности алгоритма **LFU–0**. Однако для практической реализации нам необходима некоторая *мера*, связывающая значение параметра m с эффективностью алгоритма **LFU– 0**.

3.2. Мера для определения параметра m. Построим меру, связывающую значение параметра m с эффективностью алгоритма **LFU–0**. Пусть

$$r_1, r_2, \ldots, r_m \tag{17}$$

представляет собой последовательность обращений к страницам диска. Определим q_{imk} как вероятность того, что i-я страница встречается в данной последовательности k раз. Очевидно, что

$$0 \leqslant k \leqslant m. \tag{18}$$

Пусть $G_{im}(z)$ будет производящей функцией для последовательности $q_{im0}, q_{im1}, \ldots, q_{imm}$. Имеем

$$q_{imk} = p_i q_{i,m-1,k-1} + (1 - p_i) q_{i,m-1,k}. \tag{19}$$

По определению производящей функции запишем

$$G_{im}(z) = \sum_k q_{imk} z^k. \tag{20}$$

Отсюда получаем

$$G_{i1}(z) = q_{i10} + q_{i11} z = 1 - p_i + p_i z = p_i(z - 1) + 1. \tag{21}$$

Из (19) следует, что

$$G_{im}(z) = p_i z G_{i,m-1}(z) + (1 - p_i) G_{i,m-1}(z) = (p_i(z - 1) + 1) G_{i,m-1}(z). \tag{22}$$

Из (21) и (22) находим

$$G_{im}(z) = (p_i(z - 1) + 1)^m. \tag{23}$$

Используя биномиальную теорему, из (23) получаем

$$\begin{aligned} G_{im}(z) &= (p_i(z - 1) + 1)^m = p_i^m (z + (1 - p_i)/p_i)^m = \\ &= p_i^m \sum_k \binom{m}{k} \frac{(1 - p_i)^{m-k}}{p_i^{m-k}} z^k = \sum_k \binom{m}{k} p_i^k (1 - p_i)^{m-k} z^k. \end{aligned} \tag{24}$$

Сопоставляя (24) с (20), имеем

$$q_{imk} = \binom{m}{k} p_i^k (1 - p_i)^{m-k} = m! \frac{p_i^k}{k!\,(m-k)!} (1 - p_i)^{m-k} = m! \frac{p_i^k}{k!} \frac{(1 - p_i)^{m-k}}{(m-k)!}. \tag{25}$$

Используя формулу Стирлинга для приближенного вычисления факториала

$$n! \approx \sqrt{2\pi n}(n/e)^n = \sqrt{2\pi n}(e^{\ln n}/e)^n = \sqrt{2\pi n} e^{(\ln n - 1)n}, \tag{26}$$

из (25) можно получить

$$q_{imk} \approx \tilde{q}_{imk} = \sqrt{\frac{m}{2\pi k(m-k)}} \cdot m^m \left(\frac{p_i}{k}\right)^k \left(\frac{1-p_i}{m-k}\right)^{m-k}. \tag{27}$$

Обозначим $u_{im} = q_{i,m,p_i m}$ как вероятность того, что i-я страница встречается $p_i m$ раз в последовательности (17) (без существенного ограничения общности мы можем считать, что $p_i m$ всегда является целым числом). Из (25) находим

$$u_{im} = m! \frac{p_i^{p_i m}}{(p_i m)!} \frac{(1-p_i)^{(1-p_i)m}}{((1-p_i)\cdot m)!}. \tag{28}$$

С помощью формулы (26) из (28) можно получить

$$u_{im} \approx \tilde{u}_{im} = \frac{1}{\sqrt{2\pi m \cdot p_i(1-p_i)}}. \tag{29}$$

Определим функцию $f_{im}(k) = \tilde{q}_{imk}/\tilde{u}_{im}$. На основании (27) и (29) имеем

$$f_{im}(k) = \frac{\tilde{q}_{imk}}{\tilde{u}_{im}} = \sqrt{\frac{p_i(1-p_i)}{\frac{k}{m}\left(1-\frac{k}{m}\right)}} \cdot m^m \left(\frac{p_i}{k}\right)^k \left(\frac{1-p_i}{m-k}\right)^{m-k}. \tag{30}$$

Введем новую переменную x, связанную с переменной k соотношением $x = \frac{k}{mp_i}$. В силу (18) имеем

$$0 \leqslant x \leqslant 1/p_i. \tag{31}$$

Определим *нормальную функцию* $\nu_{im}(x)$ на интервале $(0; 1/p_i)$ следующим образом:

$$\nu_{im}(x) = f_{im}(mp_i x). \tag{32}$$

Отсюда, используя (30), получаем

$$\nu_{im}(x) = \left(\frac{1}{x}\right)^{mp_i x + 1/2} \left(\frac{1-p}{1-p_i x}\right)^{m(1-p_i x)+1/2}, \tag{33}$$

или в другом виде

$$\nu_{im}(x) = e^{-(mp_i x + 1/2)\ln x + (m(1-p_i x)+1/2)\ln\frac{1-p_i}{1-p_i x}}. \tag{34}$$

Доопределим функцию $\nu_{im}(x)$ на отрезке $[0; 1/p_i]$ следующим образом:

$$\nu_{im}(0) = \nu_{im}(1/p_i) = 0. \tag{35}$$

Положим $p = p_1$ и $v_m(x) = v_{1m}(x)$. Определим функцию

$$\mathbf{M}(m) = \int_0^{1/p} \nu_m(x)dx. \tag{36}$$

Данная функция определяет искомую меру, связывающую значение параметра m с эффективностью алгоритма **LFU–0**. Действительно, чем меньше значение меры $\mathbf{M}$, тем точнее может быть определена величина p_i по формуле

$$p_i = k/m, \tag{37}$$

где k — число вхождений i в последовательность (17) (это следует непосредственно из способа построения меры $\mathbf{M}$).

Из рис. 1 видно, что при фиксированном m бóльшим значениям вероятности p соответствует меньшее значение меры $\mathbf{M}$. Данный результат является закономерным, так как чем больше вероятность обращения к странице i, тем меньший участок трассы нам необходим для определения этой вероятности по формуле (37) с заданной точностью.

В свою очередь, рис. 2 показывает, что при увеличении значения m при фиксированном p значение меры $\mathbf{M}$ уменьшается. Данный результат также представляется закономерным, так как чем больший

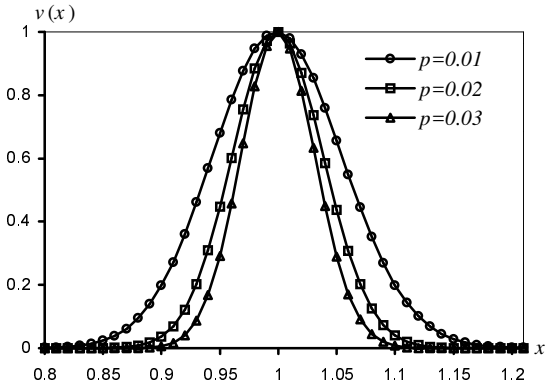

Рис. 1. Вариации $v(x)$ для различных значений p при $m = 32000$

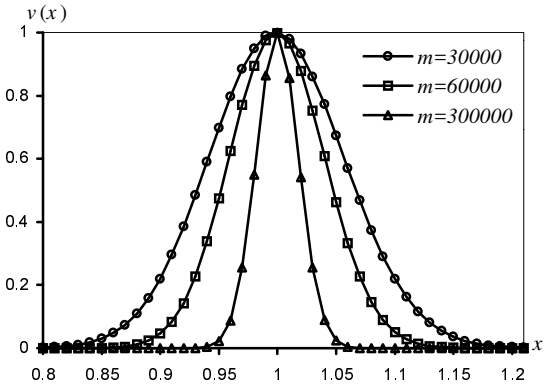

Рис. 2. Вариации $v(x)$ для различных значений m при $p = 0.01$

участок трассы мы анализируем, тем точнее формула (37) будет определять вероятность обращения к странице с номером i.

Однако мера, определяемая формулой (37), не может быть использована для практического определения величины параметра m, так как представляется затруднительным найти обратную функцию $m(\mathbf{M})$. Мы попытаемся решить данную проблему путем разложения нормальной функции $\nu_{im}(x)$ в ряд Тейлора, чтобы затем перейти к некоторой *приближенной мере* $\tilde{\mathbf{M}}(m)$, которая может быть на практике использована для вычисления значения m.

3.3. Разложение нормальной функции в ряд Тейлора. Построим разложение нормальной функции $\nu_{im}(x)$ в ряд Тейлора в точке $x = 1$. Из (34) несложно получить

$$\nu'_{im}(x) = \nu_{im}(x) \cdot \left(mp_i \ln \frac{1 - p_i x}{(1 - p_i)x} + \frac{p_i x - 1/2}{(1 - p_i x)x} \right). \tag{38}$$

Обозначим

$$\varphi_{im}(x) = mp_i \ln \frac{1 - p_i x}{(1 - p_i)x} + \frac{p_i x - 1/2}{(1 - p_i x)x}. \tag{39}$$

Тогда (38) можно переписать в виде:

$$\nu'_{im}(x) = \nu_{im}(x) \cdot \varphi(x). \tag{40}$$

Находим

$$\varphi'_{im}(x) = \frac{-mp_i}{(1 - p_i x)t} + \frac{p_i^2 x^2 - p_i x + 1/2}{(1 - p_i x)^2 x^2}. \tag{41}$$

Используя (40) и (41), получаем

$$\nu''_{im}(x) = \big(\nu_{im}(x) \cdot \varphi(x)\big)' = \nu'_{im}(x) \cdot \varphi(x) + \nu_{im}(x) \cdot \varphi'(x) = \nu_{im}(x) \cdot \varphi^2(x) + \nu_{im}(x) \cdot \varphi'(x). \tag{42}$$

Из (33) находим

$$\nu_{im}(1) = 1. \tag{43}$$

Используя (39), вычисляем

$$\varphi_{im}(1) = \frac{p_i - 1/2}{1 - p_i}. \tag{44}$$

Используя (40), с учетом (43) и (44) имеем

$$\nu'_{im}(1) = \nu_{im}(1) \cdot \varphi_{im}(1) = \frac{p_i - 1/2}{1 - p_i}. \tag{45}$$

Используя (41), вычисляем

$$\varphi'_{im}(1) = \frac{mp_i^2 - p_i^2 - mp_i + p_i + 1/2}{(1 - p_i)^2}. \tag{46}$$

Используя (42), с учетом (43), (44), (45) и (46) находим

$$\nu''_{im}(1) = \varphi^2_{im}(1) + \varphi'_{im}(1) = -\frac{mp_i(1-p_i) - 3/4}{(1-p_i)^2}. \tag{47}$$

Используя (43), (45) и (47), получаем для $\nu_{im}(x)$ разложение в ряд Тейлора в точке $x = 1$:

$$\nu_{im}(x) = 1 - \frac{1/2 - p_i}{1-p_i}(x-1) - \frac{mp_i(1-p_i) - 3/4}{2(1-p_i)^2}(x-1)^2 + R(x) \tag{48}$$

с остаточным членом

$$R(x) = \frac{\nu'''_{im}\big(\theta \cdot (x-1)\big)}{6}(x-1)^3, \quad \theta \in (0,1).$$

3.4. Приближенная мера для параметра m. Определим на основе формулы (48) функцию $\tilde{\nu}_{im}(x)$, являющуюся приближением для $\nu_{im}(x)$:

$$\tilde{\nu}_{im}(x) = -\frac{mp_i(1-p_i) - 3/4}{2(1-p_i)^2}(x-1)^2 - \frac{1/2 - p_i}{1-p_i}(x-1) + 1. \tag{49}$$

При достаточно больших m график функции $\tilde{\nu}_{im}(x)$ представляет собой параболу с вершиной в точке $(1;1)$ и ветвями, направленными вниз. Следовательно, уравнение $\tilde{\nu}_{im}(x) = 0$ имеет два вещественных корня x_1 и x_2. Предположим, что $x_1 < x_2$. Определим функцию

$$\tilde{\mathrm{M}}(m) = \int_{x_1}^{x_2} \tilde{\nu}_{1m}(x)\,dx. \tag{50}$$

Данная функция может служить *приближенной мерой* для определения величины m. Действительно, чем меньше значение интеграла (50), тем точнее будет определяться вероятность выборки страницы по числу ее вхождений в трассу длины m. Таким образом, мы можем задать некоторое фиксированное положительное число $\mathbf{E}$ и выбирать значение m как наименьшее натуральное число, такое, что

$$\tilde{\mathbf{M}}(m) < \mathbf{E}. \tag{51}$$

Для больших m справедливо соотношение

$$\frac{mp_1(1-p_1) - 3/4}{2(1-p_1)^2} \succ \frac{1/2 - p_1}{1-p_1}. \tag{52}$$

Тем самым мы можем считать, что для больших m величина интеграла (50) пропорциональна

$$\frac{mp_1(1-p_1) - 3/4}{2(1-p_1)^2}.$$

Отсюда следует, что соотношение (51) равносильно соотношению

$$\frac{mp_1(1-p_1) - 3/4}{2(1-p_1)^2} > \Delta, \tag{53}$$

где $\Delta = C\mathbf{E}$ для некоторой константы $C > 0$.

Найдем оценку для величины Δ. Для $\Theta = 0.5 \approx 1 - \log 0.45/\log 0.20$ и $N = 32000$, из (12) находим $p_1 \approx 0.0029$. Как показывает рис. 3, приемлемую точность для экспериментальной оценки величины p_1 можно получить при $m = 500\,000$ (на рис. 3 приведены графики для *точной меры*, определяемой формулой (36)). Подставляя указанные значения m и p_1 в (53), получаем $\Delta < 727$.

Положим $\Delta = 700$. Тогда из (53) получаем

$$m > \frac{\Delta 2(1-p_1)^2 + 3/4}{p_1(1-p_1)} > \frac{\Delta 2(1-p_1)^2}{p_1(1-p_1)} = 2\Delta(1/p_1 - 1). \tag{54}$$

Используя (16), отсюда находим окончательную формулу для аналитической оценки параметра m:

$$m > 2\Delta\left(\zeta(\Theta) + \frac{N^{1-\Theta} - 1}{1-\Theta} - 1\right), \quad \Delta = 700. \tag{55}$$

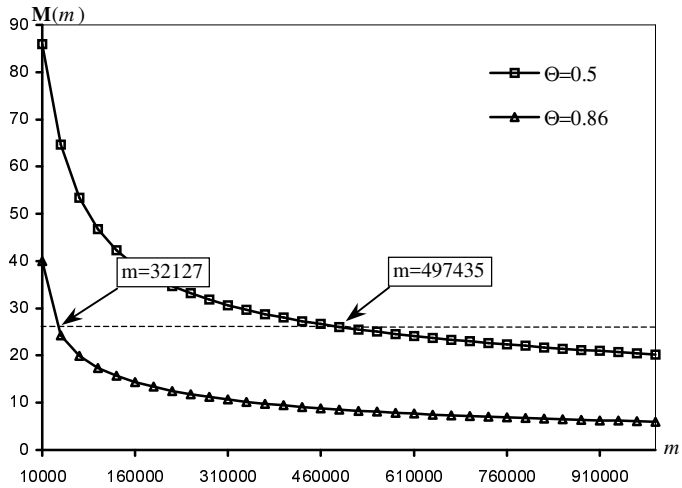

Рис. 3. Вариации M(m) для различных распределений при N=32000

Для проверки оценим по полученной формуле значение m для коэффициента перекоса $\Theta = 0.5$ при $N = 32000$. С помощью формулы (13) находим $\zeta(0.5) \approx 0.54$. Подставляя указанные значения в формулу (55), получаем $m > 497\,435$. Аналогичным образом оценим значение m для коэффициента перекоса $\Theta = 0.86 \approx 1 - \log 0.80/\log 0.20$ при $N = 32000$. Используя формулу (13), находим $\zeta(0.86) \approx 0.57$. Подставляя указанные значения в формулу (55), получаем $m > 32127$. Найденные таким образом границы для m прекрасно согласуются с графиками точной меры $\mathbf{M}(m)$, изображенными на рис. 3. Действительно, на рисунке видно, что обе найденные границы соответствуют приблизительно одному и тому же значению меры $\mathbf{M}(m)$.

4. Реализация алгоритма LFU–*K*. В данном разделе мы сначала опишем реализацию стратегии замещения **LFU–2** в виде алгоритма, названного нами **LFU–2m**. Затем рассмотрим возможность обобщения предложенной реализации для **LFU–*K*** при $K > 2$.

При реализации алгоритма **LFU–2m** мы применили следующие технические приемы:

— использование избыточного индекса буферного пула DIR;

— использование механизма рейтингов для буферизованных страниц.

Индекс буферного пула DIR [6] представляет собой хеш-таблицу в оперативной памяти, элементы которой называются *индексами* и содержат необходимую для работы алгоритма статистическую информацию о буферизуемых страницах. Размер DIR является параметром реализации и может варьироваться от значения, равного длине буфера, до значения, равного количеству всех буферизуемых страниц базы данных. DIR не содержит в себе образы страниц. Вместо этого в DIR помещаются указатели на образы страниц, загруженных в буфер. При обращении к очередной странице диска проверяется наличие ее индекса в DIR. Если индекс указанной страницы в DIR отсутствует, то происходит добавление нового индекса в DIR. Если при этом оказывается, что DIR переполнен, происходит предварительное вытеснение некоторого индекса из DIR. Поиск жертвы осуществляется только среди индексов DIR, соответствующих незагруженным в данный момент в буфер страницам. При этом используется та же стратегия замещения, что и для буферизованных страниц. Как будет показано в разделе 6, использование избыточного индекса буферного пула DIR может существенным образом повысить эффективность алгоритма **LFU–2m**.

Механизм рейтингов страниц используется при реализации многих алгоритмов замещения. Суть данного механизма заключается в том, что в соответствии со стратегией замещения задается *функция вычисления рейтинга*, отображающая множество номеров страниц во множество неотрицательных целых чисел, называемых *рейтингами*. При вытеснении страницы в качестве *жертвы* всегда выбирается страница с наименьшим рейтингом. Если наименьшее значение рейтинга имеют несколько страниц, то среди них выбирается страница с более ранним временем загрузки в буфер. Механизм рейтингов позволяет в известной мере абстрагироваться от особенностей конкретной стратегии замещения, что обеспечивает большую наглядность и гибкость в реализации алгоритма управления буферным пулом.

Схема реализации алгоритма **LFU–2m** на псевдокоде приведена на рис. 4. В данной реализации мы использовали следующие структуры данных:

```
Процедура, выполняемая при обращении к странице r:

/* Вставка в DIR */
if индекс r отсутствует в DIR then
   if DIR переполнен then
      вытеснить из DIR индекс страницы с минимальным рейтингом
   end if
   поместить индекс r в DIR
   запомнить в DIR[r] время помещения индекса r в DIR
end if
/* Вставка в BUF */
if страница DIR[r] отсутствует в BUF then
   if BUF переполнен then
      вытеснить из BUF страницу с минимальным рейтингом
   end if
   поместить r в BUF
   запомнить в DIR[r] время помещения r в BUF
   запомнить в DIR[r], что r находится в BUF
end if
/* Вставка в Squeue */
if Squeue переполнена then
   удалить элемент из начала Squeue
end if
добавить номер r в конец Squeue
/* Вставка в Vqueue */
if Vqueue2 переполнена then
   удалить элемент из начала Vqueue2
end if
if Vqueue1 переполнена then
   удалить элемент из начала Vqueue1 и поместить его в конец
   Vqueue2
end if
добавить номер r в конец Vqueue1
```

Рис. 4. Схема реализации алгоритма **LFU-2m**

- BUF — буферный пул фиксированной длины;
- DIR — избыточный индекс буферного пула. DIR$[r]$ обозначает элемент DIR, соответствующий странице с номером r;
- Squeue — очередь длины m, содержащая m последних элементов трассы;
- Vqueue1 — очередь длины h $(h < m/2)$, содержащая h последних элементов трассы;
- Vqueue2 — очередь длины h, в которую помещаются элементы, удаляемые из Vqueue1.

Схема работы очередей Squeue, Vqueue1 и Vqueue2 приведена на рис. 5. Данные очереди используются для подсчета рейтингов страниц в соответствии со стратегией **LFU–2**.

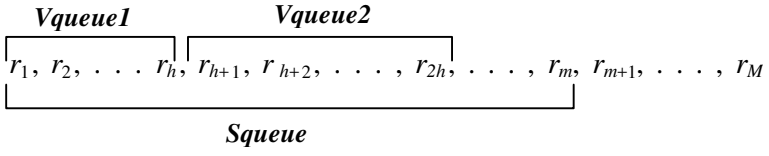

Рис. 5. Схема работы очередей *Squeue*, *Vqueue1* и *Vqueue2* (меньшее значение индекса элемента трассы соответствует более позднему по времени обращению)

В упрощенном виде формула вычисления рейтинга страницы r выглядит следующим образом:

$$\mathsf{рейтинг}(r) = s(r) + v_1(r)\,t + \big(v_1(r) - v_2(r)\big)\,t^2/2.$$

Здесь $t = m/h$, $s(r)$ — количество вхождений r в очередь Squeue, $v_1(r)$ — количество вхождений r в очередь Vqueue1, $\nu_2(r)$ — количество вхождений r в очередь Vqueue2. Очевидно, что данная формула соответствует формуле (9) из раздела 2. Однако для практического использования она не вполне пригодна. Для уяснения этого факта рассмотрим рис. 6, на котором изображены флуктуации *суммы абсолютных ускорений* A_s, вычисляемой по формуле

$$A_s = \sum_r \big|v_1(r) - v_2(r)\big|. \tag{56}$$

Рис. 6. Флуктуации суммы абсолютных ускорений A_s для стационарного распределения по правилу "80-20" при h=2500 для базы данных, состоящей из 32000 страниц

Данный график был нами получен путем компьютерного моделирования трассы со стационарным распределением вероятностей обращений по правилу “80–20” для базы данных, состоящей из 32000 страниц. В нашем эксперименте было выбрано значение параметра h, равное 2500.

Правило “80–20” часто встречается в приложениях баз данных [13]. Это правило гласит, что 80 % транзакций адресовано к 20 % базы данных. Таким образом, наша искусственная трасса достаточно хорошо отражает работу реальной системы баз данных. Поскольку мы использовали стационарное распределение, при котором вероятность обращения к той или иной странице не меняется со временем, мы вправе ожидать, что сумма абсолютных ускорений A_s, вычисляемая по формуле (56), будет всегда равна нулю. Однако наш эксперимент показывает, что это далеко не так. Учитывая, что $\lim\limits_{h\to\infty} A_s = 0$, мы могли бы поправить ситуацию, увеличивая значение параметра h. Но мы не можем произвольным образом увеличивать h, так как это снижает эффективность нашего алгоритма для трасс с динамическим распределением вероятностей. Более детально проблема выбора оптимального значения параметра h будет рассмотрена в разделе 6, где мы покажем, что использованное в данном эксперименте значение $h = 2500$ является на самом деле хорошим выбором.

Для решения проблемы отклонения A_s от нулевого значения в случае стационарного распределения мы ввели дополнительный параметр A_t, задающий *границу для ненулевых ускорений*. Используя этот

параметр, мы модернизировали формулу (56) следующим образом:

$$A_s = \sum_r \left\lfloor \frac{|v_1(r) - v_2(r)|}{A_t} \right\rfloor. \tag{57}$$

В нашем эксперименте уже при $A_t = 100$ формула (57) в подавляющем большинстве случаев выдавала для A_s значение нуль.

Руководствуясь проведенными рассуждениями, мы использовали в **LFU–2m** следующую уточненную *формулу вычисления рейтинга страниц*:

$$\text{рейтинг}(r) = s(r) + \operatorname{sign}(A_s)\, v_1(r)t + \big(v_1(r) - v_2(r)\big)\, t^2/2.$$

В целях уменьшения накладных расходов значения v_1 и v_2 для всех страниц вычислялись инкрементно и сохранялись в DIR. Значение A_s сохранялось в статической переменной и так же вычислялось инкрементно.

Алгоритм **LFU–2m** может быть достаточно легко модифицирован для использования в качестве реализации стратегии **LFU–*K*** при значениях K, больших двух. При этом нам будет необходимо поддерживать K очередей Vqueue1, ..., VqueueK для вычисления всех слагаемых в формуле (6). Однако, как показали наши эксперименты, использование алгоритма **LFU–*K*m** при $K > 2$ на практике нецелесообразно. Ни в одном из проведенных экспериментов алгоритм **LFU–3m** не смог показать заметного приращения эффективности по сравнению с алгоритмом **LFU–2m**.

5. Результаты экспериментов. Мы провели ряд экспериментов на реальных и искусственных трассах, в которых исследовали эффективность алгоритма **LFU–2m** в сравнении с известными алгоритмами замещения страниц. В нашем сравнительном анализе мы использовали алгоритмы **LFU**, **LRU** [11, 16], **LRU–2** [17] и алгоритм **2Q** [14]. Алгоритм **2Q** не представлен на рисунках, так как во всех случаях, которые мы исследовали, его эффективность оказалась очень близкой к эффективности алгоритма **LRU–2**. Кроме этого, в ряде экспериментов для абсолютной оценки качества исследуемых алгоритмов мы использовали также алгоритмы **OPT** [8, 16] и **A0** [10].

Табл. 1. Параметры алгоритма **LFU-2m**

Параметр	Семантика	Значение по умолчанию
m	длина очереди *Squeue*	500000
h	длина очереди *VqueueX*	2500
A_t	граница ненулевых ускорений для правила "80-20"	100
	граница ненулевых ускорений для правила "45-20"	8
L_{DIR}	длина DIR	32000

Во всех наших экспериментах, где не оговорено противное, мы использовали значения параметров алгоритма **LFU–2m**, приведенные в табл. 1. Обсуждение принципов подбора значений параметров и их влияния на эффективность алгоритма **LFU–2m** мы отложим до раздела 6.

5.1. Стационарное распределение вероятностей обращений. В первой серии экспериментов мы использовали искусственные трассы со *стационарным* (не меняющимся во времени) распределением вероятностей обращений к страницам диска, задаваемым формулой (12) с $\Theta = 0.86$ (правило “80–20”) и $\Theta = 0.5$ (правило “45–20”). В обоих случаях мы генерировали трассу, включающую в себя 1 000 000 обращений к базе данных из 32000 страниц. Результаты данных экспериментов представлены на рис. 7 и рис. 8. В качестве ориентира мы включили в эти эксперименты алгоритм **A0** (**A0** всегда замещает страницу с наименьшей вероятностью обращения), являющийся оптимальным для стационарного распределения вероятностей [7]. Мы не стали в данной серии экспериментов тестировать алгоритм **LRU**, так как известно, что в случае стационарного распределения вероятностей по формуле (12) **LRU** всегда проигрывает в эффективности алгоритму **LRU–2** [14].

В первом эксперименте (рис.7) мы взяли $\Theta = 0.86$, что примерно соответствует правилу “80–20” (80 % обращений адресуются к 20 % страниц). Данный эксперимент показывает, что при небольших размерах буферного пула эффективность алгоритма **LFU–2m** близка к оптимальной. Алгоритм **LRU–2** в

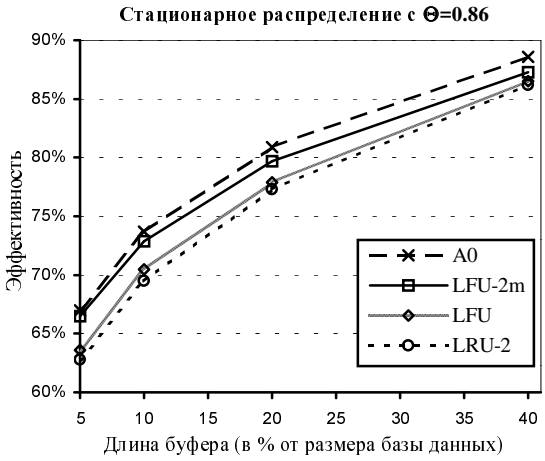

Рис. 7. Сравнение эффективности различных алгоритмов в зависимости от длины буфера для статического распределения по правилу "80-20" (Θ=0.86)

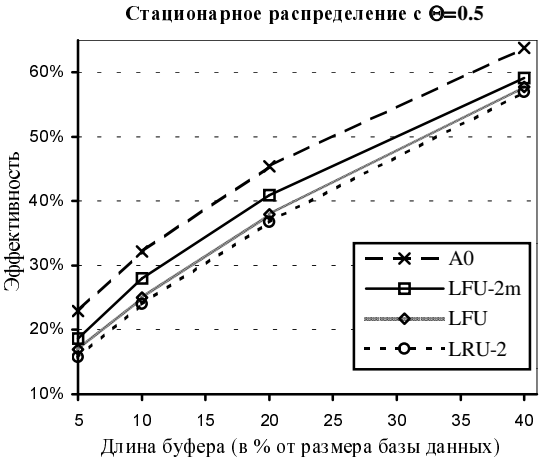

Рис. 8. Сравнение эффективности различных алгоритмов в зависимости от длины буфера для статического распределения по правилу "45-20" (Θ=0.5)

этом случае проигрывает даже алгоритму **LFU**, замещающему страницу, к которой было меньше всего обращений. Преимущество **LFU–2m** над **LFU** в данном эксперименте обусловлено использованием избыточного индекса буферного пула DIR.

Во втором эксперименте (рис.8) мы взяли $\Theta = 0.5$, что примерно соответствует правилу "45–20" (45 % обращений адресуются к 20 % страниц). Данный эксперимент показывает, что относительная эффективность алгоритма **LFU–2m** снижается с уменьшением перекоса в распределении вероятностей обращений, однако **LFU–2m** по-прежнему выигрывает в эффективности по сравнению с **LRU–2**.

5.2. Периодическое распределение вероятностей обращений. Во второй серии экспериментов мы анализировали эффективность алгоритмов замещения на трассах, характерных для параллельных систем баз данных без совместного использования ресурсов. Как уже отмечалось в разделе 1, процессорный узел в такой системе будет генерировать трассу обращений, в которой периоды с относительно стабильным распределением вероятностей обращений будут чередоваться с очень короткими периодами практически мгновенного перераспределения вероятностей обращений между страницами диска.

В соответствии с этим мы построили искусственную трассу с *периодическим* распределением, полученным путем последовательного объединения 1000 отрезков (*периодов*) длиной по 1000 элементов каждый. На каждом отдельном отрезке все страницы имеют стационарное распределение вероятностей обращений, задаваемое формулой (12). Однако на разных отрезках одна и та же страница имеет различные вероятности обращения. Размер базы данных в этой серии экспериментов составлял 32000 страниц. Результаты данных экспериментов представлены на рис. 9 и рис. 10. В качестве ориентира мы включили в наши эксперименты оптимальный алгоритм **OPT**, который всегда вытесняет страницу, к которой дольше всего не будет обращений [8, 16].

В первом эксперименте данной серии для генерации отрезков трассы мы использовали распределение (12) с коэффициентом перекоса $\Theta = 0.86$. Результаты этого эксперимента показывают (см. рис. 9), что на периодических трассах эффективность алгоритма **LFU–2m** может значительно превосходить эффективность алгоритма **LRU–2** (примерно на 17 %), приближаясь при малых размерах буферного пула к оптимальной. Следует также отметить, что алгоритм **LFU**, показывавший неплохие результаты для статического распределения, в случае периодического распределения оказывается неадекватным.

Во втором эксперименте при генерации отрезков трассы мы использовали распределение (12) с коэффициентом перекоса $\Theta = 0.5$. Результаты этого эксперимента показывают (см. рис. 10), что при уменьшении коэффициента перекоса относительная эффективность алгоритма **LFU–2m** снижается, однако и в этом случае **LFU–2m** по-прежнему заметно превосходит по эффективности алгоритм **LRU–2** (примерно на 12 %).

В третьем эксперименте мы исследовали эффективность различных алгоритмов замещения в зависимости от длины периодов стабильности. При генерации отрезков трассы было использовано распределение

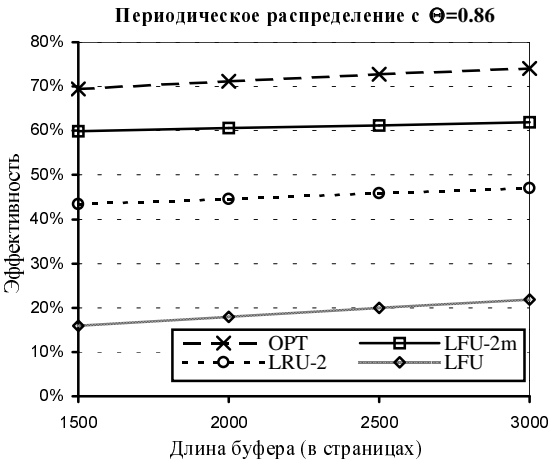

Рис. 9. Сравнение эффективности различных алгоритмов в зависимости от длины буфера для периодического распределения по правилу "80-20" (Θ=0.86)

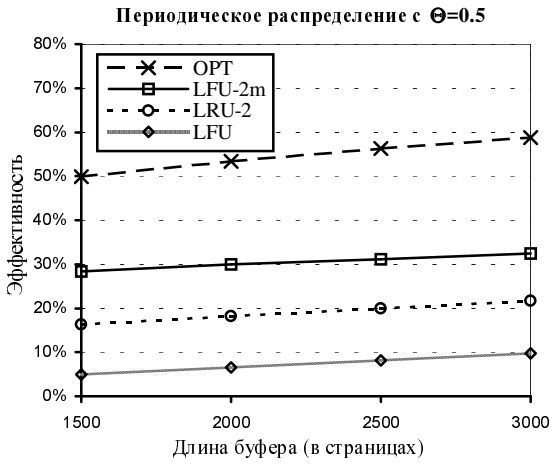

Рис. 10. Сравнение эффективности различных алгоритмов в зависимости от длины буфера для периодического распределения по правилу "45-20" (Θ=0.5)

(12) с коэффициентом перекоса $\Theta = 0.86$. Длина буфера всегда была постоянной и равнялась 1500 страницам. Данный эксперимент показал (см. рис. 11), что при длине периода, большей 2000 и меньшей 20000, алгоритм **LRU** демонстрирует более высокую эффективность по сравнению с **LFU–2m** и **LRU–2**. Однако разница между эффективностью **LRU** и **LFU–2m** не превышает 4 %, тогда как разница между эффективностью **LRU** и **LRU–2** может быть значительной (до 17 %) при малой длине периодов. Учитывая, что **LRU** существенно проигрывает в эффективности алгоритмам **LRU–2** (см. [14]) и **LFU–2m** (см. пункты 5.1 и 5.4) на реальных и искусственных трассах со стационарным распределением вероятностей, **LFU–2m** по-прежнему остается лучшим выбором.

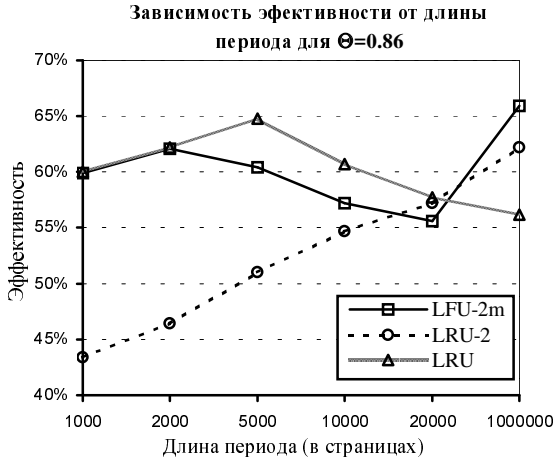

Рис. 11. Сравнение эффективности различных алгоритмов замещения для периодического распределения в зависимости от длины периода при Θ=0.86 и длине буфера, равной 1500

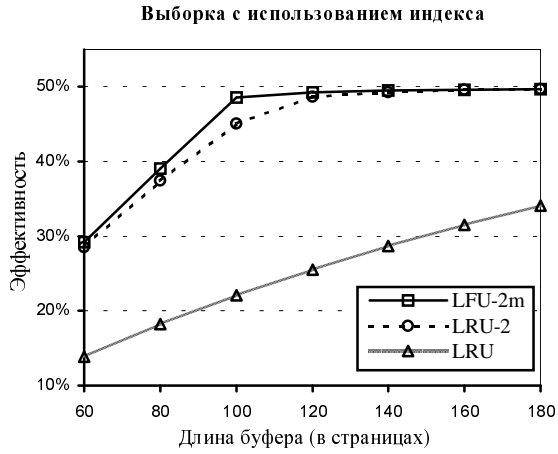

Рис. 12. Сравнение эффективности различных алгоритмов в зависимости от длины буфера для выборки с использованием индекса

5.3. Доступ в режиме OLTP с использованием индексного файла. Одним из важных наблюдений, сделанных в работе [17], является то, что стратегия **LRU** оказывается неадекватной для случайной выборки записей в режиме OLTP (On-Line Transaction Processing) [3] при использовании индексного

файла. Авторы провели эксперимент, в котором осуществлялись поочередные выборки страницы из индексного файла размером в 100 страниц и страницы из файла данных размером в 10000 страниц. Выборки производились случайным образом в соответствии с равномерным законом распределения вероятностей обращений. Данный эксперимент показал, что **LRU–2** присваивает более высокий рейтинг страницам индексного файла, как это и должно быть, так как вероятность обращения к странице индексного файла примерно в 100 раз выше, чем вероятность обращения к странице файла данных. Мы воспроизвели данный эксперимент с идентичными параметрами трассы. Эксперимент показал, что алгоритм **LFU–2m** также присваивает более высокий приоритет страницам индексного файла (см. рис. 12).

5.4. Эксперименты на реальной трассе. Заключительный эксперимент был нами проведен на реальной трассе, полученной при работе коммерческой инсталляции СУБД DB2. Данная трасса содержала 1 000 000 обращений к уникальным страницам, число которых составляло 52701. Результаты эксперимента приведены на рис. 13.

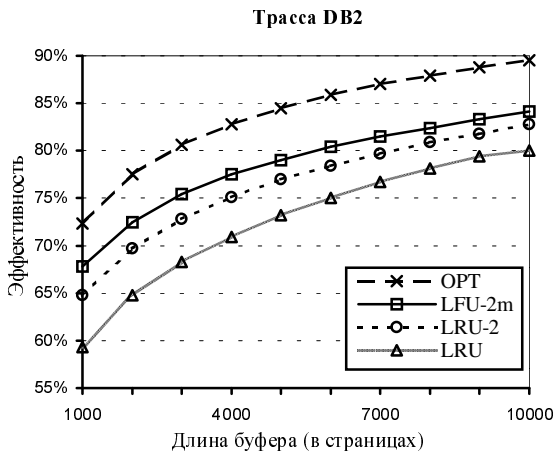

Рис. 13. Сравнение эффективности различных алгоритмов замещения в зависимости от длины буфера для реальной трассы, полученной при работе коммерческой инсталляции СУБД DB2

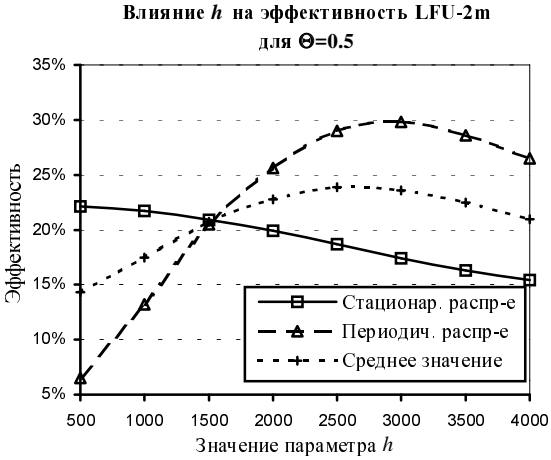

Рис. 14. Зависимость эффективности алгоритма **LFU-2m** от величины параметра h для периодического (с периодом 1000) и стационарного распределений с Θ=0.5 при длине буфера, равной 1600 страниц

Алгоритм **LFU–2m** в данном эксперименте также показал более высокую эффективность по сравнению с алгоритмами **LRU–2** и **LRU**. При уменьшении размеров буферного пула эффективность алгоритма **LFU–2m** приближалась к оптимальной.

6. Подбор параметров алгоритма LFU–2m. В данном разделе мы рассмотрим проблему подбора оптимальных значений параметров алгоритма **LFU–2m**, приведенных в табл. 1. Соответствующие оценки для параметра m были получены в разделе 3, где мы показали, что при распределении вероятностей обращений по правилу "45–20" приемлемым является значение $m = 500\,000$. При увеличении коэффициента перекоса в распределении вероятностей обращений до величины, соответствующей правилу "80–20", значение параметра m может быть уменьшено до 30000 без ощутимой потери эффективности.

Для исследования влияния величины параметра A_t на эффективность алгоритма **LFU–2m** мы провели вычислительные эксперименты на стационарной и периодической искусственных трассах, параметры которых были описаны в пунктах 5.1 и 5.2. Результаты экспериментов показали (см. рис. 15 и рис. 16), что при увеличении значения A_t в случае стационарного распределения эффективность алгоритма увеличивается, а в случае периодического распределения — уменьшается.

Данный факт является вполне закономерным, так как увеличение A_t в случае стационарного распределения нивелирует случайные возмущения (см. рис. 6), отрицательно влияющие на эффективность алгоритма **LFU–2m**. В случае периодического распределения увеличение A_t приводит к тому, что нивелируются уже *неслучайные* возмущения, связанные со сменой распределения вероятностей, и, как следствие, эффективность **LFU–2m** деградирует до уровня **LFU**. В соответствии с этим при выборе значения параметра A_t мы вынуждены искать компромисс. Из анализа графиков на рис. 15 видно, что в случае $\Theta = 0.86$ таким компромиссом является значение $A_t = 100$. В свою очередь, анализ графиков на рис. 16

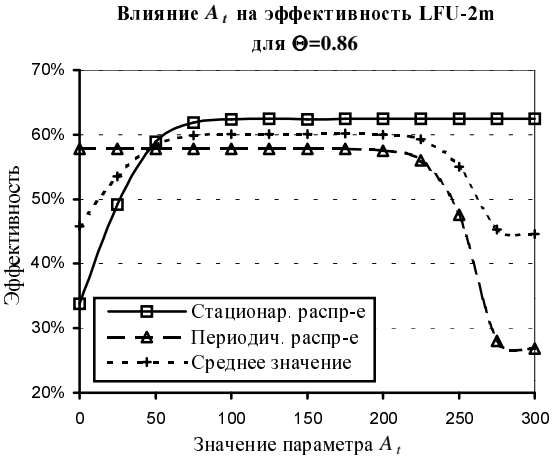

Рис. 15. Зависимость эффективности алгоритма **LFU-2m** от величины параметра A_t для периодического (с периодом 1000) и стационарного распределений с Θ=0.86 при длине буфера, равной 1000 страниц

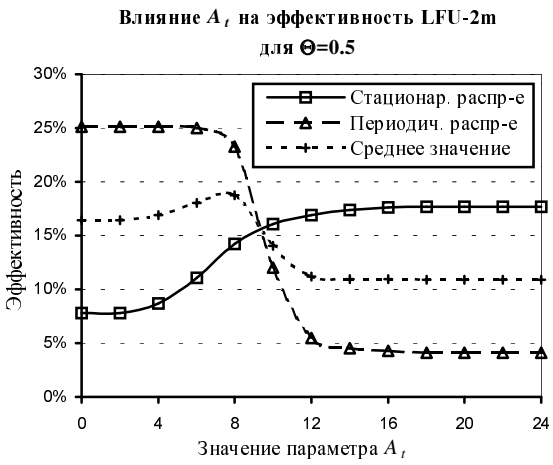

Рис. 16. Зависимость эффективности алгоритма **LFU-2m** от величины параметра A_t для периодического (с периодом 1000) и стационарного распределений с Θ=0.5 при длине буфера, равной 1000 страниц

показывает, что для случая $\Theta = 0.5$ целесообразно взять $A_t = 8$.

В следующем эксперименте мы исследовали зависимость эффективности **LFU–2m** от величины параметра h. Результаты этого эксперимента приведены на рис. 14 . При увеличении значения h на трассе со стационарным распределением эффективность **LFU–2m** падает. Это объясняется тем, что при увеличении h увеличивается и количество случайных возмущений, преодолевающих порог ненулевых ускорений A_t, что отрицательно влияет на эффективность алгоритма. С другой стороны, в случае периодического распределения эффективность **LFU–2m** резко увеличивается с ростом h вплоть до точки $h = 3000$. Затем наступает медленный спад. Анализ графиков показывает, что приемлемый компромисс достигается в точке $h = 2500$.

В заключительной серии экспериментов мы исследовали, как длина DIR влияет на эффективность алгоритма **LFU–2m**. Результаты этих экспериментов, приведенные на рис. 17 и рис. 18, показывают, что в случае $\Theta = 0.86$ (рис. 17) длина DIR, составляющая 25–30 % от количества буферизуемых страниц, является вполне приемлемым выбором. В случае $\Theta = 0.5$ (рис. 18) целесообразно брать максимально возможный размер DIR, хотя и здесь можно ограничиться длиной DIR, составляющей 35–40 % от количества буферизуемых страниц.

7. Заключение. В данной работе был предложен новый алгоритм замещения страниц **LFU–*K***, являющийся обобщением хорошо известного алгоритма **LFU**. Для предложенного алгоритма была построена теоретико-вероятностная модель, на основе которой получены аналитические оценки для параметра m. Оставшиеся параметры алгоритма были исследованы в вычислительных экспериментах, и на основе полученных результатов даны рекомендации по выбору значений данных параметров. На базе алгоритма **LFU–2** был построен его модернизированный вариант **LFU–2m**, пригодный для использования на практике. На языке Си была написана программа, эмулирующая работу алгоритмов **LFU–2m**, **LFU**, **LRU**, **LRU–2**, **2Q**, **OPT** и **A0**. С помощью этой программы были проведены вычислительные эксперименты на искусственных и реальных трассах обращений к страницам диска. Проведенные эксперименты показали высокую эффективность алгоритма **LFU–2m** практически на всех видах трасс. При этом на трассах, моделирующих работу процессорного узла параллельной системы баз данных без совместного использования ресурсов, эффективность алгоритма **LFU–2m** значительно превышала эффективность известных алгоритмов замещения.

Алгоритм **LFU–2m** был нами использован при реализации прототипа параллельной СУБД Омега для многопроцессорной системы МВС–100/1000 [4, 24]. Опытная эксплуатация системы подтвердила высокую эффективность предложенного алгоритма.

В качестве возможных направлений дальнейших исследований можно выделить следующие. Во-первых, было бы интересно получить аналитические оценки для выбора значений параметров h и A_t.

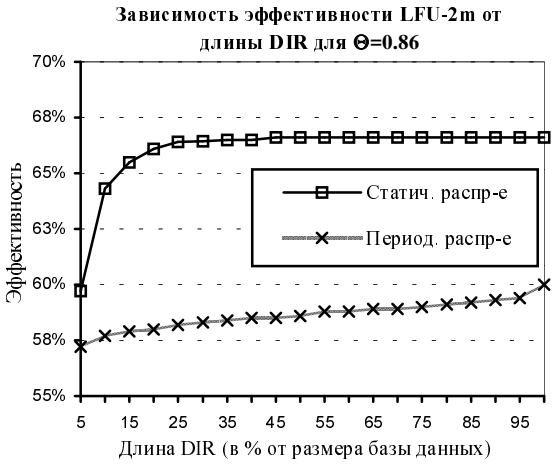

Рис. 17. Зависимость эффективности алгоритма **LFU-2m** от длины DIR для периодического (с периодом 1000) и стационарного распределений с Θ=0.86 при длине буфера, равной 1600 страниц

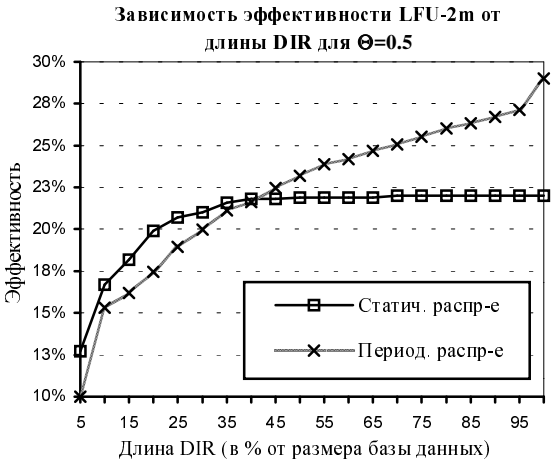

Рис. 18. Зависимость эффективности алгоритма **LFU-2m** от длины DIR для периодического (с периодом 1000) и стационарного распределений с Θ=0.5 при длине буфера, равной 1600 страниц

Во-вторых, определенный интерес представляло бы доказательство или опровержение *гипотезы оптимальности алгоритма* **LFU–*K***. Под оптимальностью мы здесь подразумеваем тот факт, что любой алгоритм вытеснения, использующий не больше информации о трассе, чем использует алгоритм **LFU–*K***, не может показывать эффективность, превышающую эффективность алгоритма **LFU–*K***. В-третьих, мы предполагаем, что алгоритм **LFU–*K*** может быть использован не только в параллельных системах баз данных без совместного использования ресурсов. Одним из таких потенциальных приложений могло бы быть кэширование Web-страниц на proxy-серверах, обслуживающих большое число пользователей. Как известно, proxy-сервер может сохранять на локальном диске Web-страницы, к которым осуществляются частые повторные обращения. Однако при большом количестве пользователей неизбежно возникает ситуация переполнения диска, что приводит к необходимости удаления с диска некоторых страниц. Таким образом, возникает типичная задача оптимального замещения страниц с той лишь разницей, что она выполняется не в оперативной памяти, а на диске. Было бы интересно исследовать применимость алгоритма **LFU–*K*** к этой задаче.

Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проект 00–07–90077).

СПИСОК ЛИТЕРАТУРЫ

1. *Кнут Д.Э.* Искусство программирования. Т. 1. Основные алгоритмы. М.: Издательский дом “Вильямс”, 2000.
2. *Кнут Д.Э.* Искусство программирования. Т. 3. Сортировка и поиск. М.: Издательский дом “Вильямс”, 2000.
3. *Когаловский М.Р.* Энциклопедия технологий баз данных. М.: Финансы и статистика, 2002.
4. *Соколинский Л.Б.* Организация параллельного выполнения запросов в многопроцессорной машине баз данных с иерархической архитектурой // Программирование. 2001. В6. 13–29.
5. *Титчмарш Е.К.* Теория дзета-функции Римана. М.: ИЛ, 1953.
6. *Цымблер М.Л., Соколинский Л.Б.* Организация обработки больших объемов данных в многопроцессорных системах с массовым параллелизмом // Высокопроизводительные вычисления и их приложения. М.: Изд-во Моск. ун-та, 2000. 186–190.
7. *Aho A.V., Denning P.J., Ullman J.D.* Principles of optimal page replacement // J. of the ACM. 1971. **18**, N 1. 80–93.
8. *Belady L.A.* A study of replacement algorithms for virtual-storage computer // IBM Systems Journal. 1966. **5**, N 2. 78–101.
9. *Chou H.-T., DeWitt D.J.* An evaluation of buffer management strategies for relational database systems // Proceedings of the 11th International Conference on Very Large Data Bases. Stockholm: Morgan Kaufmann, 1985. 127–141.
10. *Coffman E.G., Denning P.J.* Operating systems theory. Englewood Cliffs: Prentice-Hall, 1973.

11. *Effelsberg W., Harder T.* Principles of database buffer management // ACM Trans. on Database Systems. 1984. **9**, N 4. 560–595.
12. *Gray J., Graefe G.* The five-minute rule ten years later, and other computer storage rules of thumb // SIGMOD Record. 1997. **26**, N 4. 63–68.
13. *Heising W.P.* Note on random addressing techniques // IBM Systems Journal. 1963. **2**, N 2. 112–116.
14. *Johnson T., Shasha D.* 2Q: a low overhead high performance buffer management replacement algorithm // Proceedings of the 20th International Conference on Very Large Data Bases. 1994. Santiago de Chile: Morgan Kaufmann, 1994. 439–450.
15. *Lee D., Choi J., Kim J.-H., Noh S.H., Min S.L., Cho Y., Kim C.-S.* On the existence of a spectrum of policies that subsumes the least recently used (LRU) and least frequently used (LFU) policies // International Conference on Measurement and Modeling of Computer Systems. Atlanta, 1999. 134–143.
16. *Mattson R.L., et al.* Evaluation techniques for storage hierarchies // IBM Systems Journal. 1970. **9**, N 2. 78–117.
17. *O'Neil E.J., O'Neil P.E., Weikum G.* The **LRU–*K*** page replacement algorithm for database disk buffering // Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data. Washington, D.C.: ACM Press. 1993. 297–306.
18. *O'Neil E.J., O'Neil P.E., Weikum G.* An optimality proof of the **LRU–*K*** page replacement algorithm // J. of the ACM. 1999. **46**, N 1. 92–112.
19. *Rahm E., Marek R.* Analysis of dynamic load balancing strategies for parallel shared nothing database systems // The 19th International Conference on Very Large Data Bases. Dublin: Morgan Kaufmann, 1993. 182–193.
20. *Robinson J.T., Devarakonda M.V.* Data cache management using frequency-based replacement // 1990 ACM Conference on Measurement and Modeling of Computer Systems. University of Colorado. Boulder, 1990. 134–142.
21. *Sacco G. M., Schkolnick M.* Buffer management in relational database systems // ACM Transactions on Database Systems. 1986. **11**, N 4. 473–498.
22. *Sleator D.D., Tarjan R.E.* Amortized efficiency of list update and paging rules // Communications of the ACM. 1985. **28**, N 2. 202–208.
23. *Smaragdakis Y., Kaplan S., Wilson P.R.* EELRU: simple and effective adaptive page replacement // International Conference on Measurement and Modeling of Computer Systems. Atlanta, 1999. 122–133.
24. *Sokolinsky L.B.* Design and evaluation of database multiprocessor architecture with high data availability // The 12th International DEXA Workshop. Munich, 2001. 115–120.
25. *Stonebraker M.* Operating system support for database management // Communications of the ACM. 1981. **24**, N 7. 412–418.
26. *Wilschut A.N., Flokstra J., Apers P.M.G.* Parallel evaluation of multi-join queries // Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data. San Jose: ACM Press, 1995. 115–126.
27. *Zipf G.K.* Human behavior and the principle of least effort: an introduction to human ecology. Reading: Addison-Wesley, 1949.

Поступила в редакцию
25.03.2002