УДК 004.451.31

ОПЕРАЦИИ УПРАВЛЕНИЯ МАССИВАМИ ДАННЫХ В ЛИНЕЙНО АДРЕСУЕМОЙ ПАМЯТИ

В.П. ИВАШЕНКО, С.В. СИНЦОВ

Белорусский государственный университет информатики и радиоэлектроники
П. Бровки, 6, Минск, 220013, Беларусь

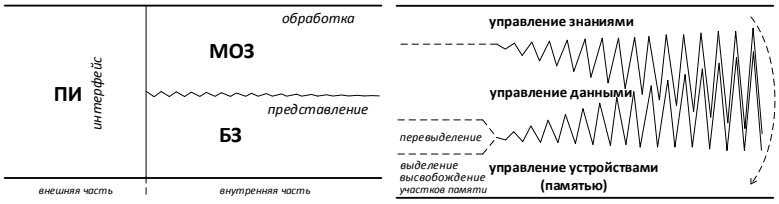
Поступила в редакцию 25 апреля 2016

Рассматривается задача управления массивами данных в конечной линейно адресуемой памяти. Как часть решения этой задачи приводится способ реализации операций управления массивами данных, основанный на использовании стратегий распределения участков памяти на уровне управления устройствами исполнения систем, основанных на знаниях. Приводятся результаты тестирования различных стратегий.

Ключевые слова: линейно адресуемая память, динамическое распределение, участок памяти.

Введение

Система, основанная на знаниях [1], состоит из базы знаний (БЗ), машины обработки знаний (МОЗ) и пользовательского интерфейса (ПИ) [2] (рис. 1, *а*). Компоненты БЗ и МОЗ относятся ко внутренней части системы, основанной на знаниях, которая скрыта от пользователя; ПИ относится ко внешней части, доступной пользователю.



а *б*

Рис. 1. Компоненты (*а*) и уровни внутренней части (*б*) системы, основанной на знаниях

В системе, основанной на знаниях, можно выделить три уровня (рис. 1, *б*): 1) уровень управления устройствами; 2) уровень управления данными; 3) уровень управления знаниями. На уровне 1 рассмотрим задачу управления памятью [3]. Решение этой задачи необходимо для решения задачи представления и обработки множеств, возникающей на уровне 2, что включает в себя реализацию различных структур данных: динамические массивы, списки, деревья и прочее [3–6]. В свою очередь, управление знаниями, представленными, например, в виде унифицированных семантических сетей с теоретико-множественной интерпретацией [7], требует решения задачи представления и обработки множеств.

Можно заключить, что производительность системы, основанной на знаниях, зависит от оптимального решения всех перечисленных выше задач, т.е. решения, дающего оптимальные гарантии по времени и объему памяти. Поэтому целью работы является разработка способов увеличения производительности операций управления памятью и, следовательно, базовых операций обработки множеств. Актуальность работы обусловлена потребностью в обработке больших объемов данных и знаний, возрастающими требованиями ко временным затратам обработки знаний, недостаточной производительностью доступных

распространенных вычислительных архитектур, а также развитием семантических моделей представления знаний.

Известны системы динамического распределения памяти такие, как tlsf [8], ptmalloc2, tcmalloc [9], jemalloc [10], hoard [11], lockless [12] и др. [13]. Также известно доказательство оценки худшего случая внешней фрагментации [14]. Система jemalloc основана на системе двойников, однако не дает гарантий по внешней фрагментации для размеров участка памяти больших, чем 2 килобайта. Системы tlsf и tcmalloc используют стратегию лучшего подходящего, которая имеет тенденцию к увеличению количества свободных участков малого размера [3] и может иметь квадратичную оценку пространственной сложности из-за сильно фрагментированной памяти [14]. Оптимальная (по Робсону [14]) оценка производительности приводится для системы hoard, в которой оценка перерасхода памяти по причине внешней фрагментации не превосходит $O(\log_2(U/u)*S)$, где S – объем занятой памяти, U,u – максимальный и минимальный размеры занятых участков соответственно. Для большинства других систем вовсе не приводится никаких математических оценок ни для времени исполнения операций выделения, высвобождения, перевыделения участков памяти, ни для степени внешней фрагментации. Кроме того, рассмотренные системы не гарантируют сохранение стратегии выбора подходящего участка для операции перевыделения.

Данная работа основывается на системе [13] и развивает полученные в ней результаты путем введения операции перевыделения участков памяти. Новизна работы в том, что в ней 1) предложена и исследована теоретическая модель, позволяющая строить, использовать и сравнивать работоспособность различных стратегий распределения участков памяти; 2) проведен анализ и дано экспериментальное подтверждение характеристик построенной модели; 3) на основе построенной модели реализована операция перевыделения, сохраняющая стратегию выбора первого подходящего свободного участка памяти и дающая близкие к оптимальным гарантии по производительности. Оценка перерасхода памяти по причине внешней фрагментации равна $O(S*\log_2(S))$; среднее амортизационное время [5] оценивается как $O(1)$, если время доступа к ячейке памяти в зависимости от размера памяти равно $O(1)$, а количество записей данных в ячейки памяти зависит от произведения времени одного перевыделения на количество перевыделений как минимум прямо пропорционально.

Управление памятью

Рассмотрим конечную линейно адресуемую битовую память размера m, обычно $m = 2^p$. Пусть время доступа (чтения/записи) к одной ячейке памяти есть значение функции $f(m)$. Для задачи управления памятью можно определить две базовые операции: выделение и высвобождение (рис. 1, *б*). Указанные операции необходимы для управления свободными участками памяти, где не происходит доступ к данным.

Система динамического управления памятью [13] дает следующие гарантии: временна́я и пространственная сложность операции выделения/высвобождения оценивается как $O(\ln^2(m)*f(m))$ и $O(\ln^3(m))$ соответственно; перерасход памяти по причине внешней фрагментации оценивается как $O(S*\log_2(S))$, где S – объем занятой памяти.

Управление массивами данных в памяти

Расширим модель, приведенную в [13], межуровневой операцией перевыделения участков памяти. Эта операция используется для управления участками памяти, в которых содержатся данные (рис. 1, *б*). При перевыделении участка памяти может происходить копирование массива данных из одного участка памяти в другой.

Операция перевыделения участка памяти может быть реализована последовательностью из трех операций: выделение нового участка памяти, копирование данных в новый участок памяти, высвобождение старого участка памяти. Пусть размер участка памяти монотонно возрастает с 1 до $n \le m$ за n операций перевыделения. Для такой цепочки изменений в худшем случае понадобится время $O(n^2*f(m))$, а сложность одной операции перевыделения составит $O(c*f(m))$, $c = 2, \ldots, n$.

Однако указанное время работы не оптимально. Временну́ю сложность можно

понизить, выделяя участок бóльшего, чем требуется, размера. Это позволит избежать копирования данных для некоторого числа последующих операций перевыделения для выделенного участка памяти (такое решение используется, например, в [9] и других системах динамического управления памятью, а также рассматривается в работах [3, 5, 15]). Для этого фактический размер $g(w)(v)$ выделяемого участка памяти поставим в соответствие $g(w)$ каждому затребованному размеру v ($v \le g(w)(v) \le m$) согласно некоторому правилу r, где w – старый размер выделенного участка памяти:

$$g(w)(v) = \begin{cases} r_3 \,|\, (r_1 \le r_2) \wedge (v \ge r_1) \wedge (v \le r_2) \wedge (v > w) \\ r_4 \,|\, (r_1 \le r_2) \wedge (v \ge r_1) \wedge (v \le r_2) \wedge (v < w) \\ r_5 \,|\, (r_1 > r_2) \wedge (v > r_2) \wedge (v \le r_1) \wedge (w \ge r_3) \wedge (w \le r_4) \end{cases} \quad (1)$$

Правило r задается пятеркой $\langle r_1, r_2, r_3, r_4, r_5 \rangle$. При $r_1 \le r_2$: r_1 и r_2 – соответственно минимальное и максимальное значения размера выделяемого участка, r_3 – фактический размер участка для выделения, r_4 – фактический размер участка для выделения при перевыделении участка меньшего размера, r_5 – флаг наличия гистерезиса. При $r_1 > r_2$: r_1 и r_2 – соответственно минимальное и максимальное значения затребованного размера выделяемого участка, r_3 и r_4 – соответственно минимальное и максимальное значение старого размера выделенного участка, r_5 – фактический размер участка для выделения или выделения при перевыделении участка. Стратегия распределения участков памяти есть множество S правил r. Выделяя участок бóльшего, чем требуется размера для случая монотонно возрастающего размера участка можно получить среднюю амортизационную стоимость одной операции перевыделения равную $O(1)$ (к примеру, для $g(w)(v) = \min(2^{\lceil \log_2 v \rceil}, m)$). При этом придется затратить дополнительный объем памяти. Однако вообразим ситуацию немонотонного изменения размера участка памяти, в которой за каждым увеличением следует уменьшение размера участка памяти, и наоборот, а для каждой операции перевыделения происходит копирование данных. Избежать такой худший случай можно, используя стратегию с гистерезисом S_h. Этот тип стратегий, при уменьшении размера участка памяти с y до v, дает значение $g(y)(v) > g(w)(v)$, где $g(w)(v)$ – фактический размер участка памяти при увеличении размера участка памяти с w до v. Значение $g(y)(v)$ определяется в соответствии с шагом гистерезиса h. Единичный шаг означает, что $g(y)(v)$ выбирается минимальным из больших фактических размеров участков.

Пусть правило r для v находится в стратегии S некоторой функцией поиска. Предлагается два способа реализации функции поиска: на основе хранимых правил и на основе метаправил. Первый способ сводится к бинарному поиску хранящихся в памяти правил. При втором способе искомое правило r формируется путем вычисления некоторых его компонент по формуле. Рассмотрим правила $r_1 \le r_2$. Для них применима формула (2), которая в общем виде задает зависимость между исходным фактическим размером x_i участка памяти и j-ым бóльшим фактическим размером x_{i+j} перевыделяемого участка:

$$x_{i+j} = \left(k^j * \sqrt[q]{x_i} + b * \sum_{l=0}^{j-1} k^l \right)^q, \quad (2)$$

где k, b, q – константы – k, $q \in R^+$, $b \in N$. Таким образом, формула (2) позволяет вычислять некоторые последовательности натуральных чисел. Каждый i-ый член последовательности определяет i-ое правило как: $\langle x_{i-1}+1, x_i, x_i, x_{i+h}, \mathrm{sgn}(h) \rangle$.

Из (2) можно получить оценки для затрат времени и дополнительного объема памяти. В стратегиях с гистерезисом в один шаг удельное время добавления одного элемента данных (и последующего его удаления) в связный участок бóльшего размера выражается формулой:

$$\frac{\tau_a + \tau_d + f(m) * (x_i + x_{i+1})}{x_{i+1} - x_i} = \frac{\tau_a + \tau_d + f(m) * \left(x_i + \left(k * \sqrt[q]{x_i} + b \right)^q \right)}{\left(k * \sqrt[q]{x_i} + b \right)^q - x_i},$$

где τ_a и τ_d – времена расширения и сокращения участка памяти соответственно. При $k > 1$ (геометрическая прогрессия), в худшем случае удельное время оценивается как

$$\lim_{i\to\infty}\frac{\tau_a+\tau_d+f(m)*\left(x_i+\left(k*\sqrt[q]{x_i}+b\right)^q\right)}{\left(k*\sqrt[q]{x_i}+b\right)^q-x_i}=f(m)*\left(\frac{k^q+1}{k^q-1}\right)+\frac{1}{k^q-1}*\lim_{i\to\infty}\left(\frac{\tau_a+\tau_d}{x_i}\right).$$

Тогда для добавления n элементов данных с учетом возможных удалений как уже имеющихся, так и добавленных данных, справедлива временна́я сложность $O(f(m)*n+d*\tau)$, где d – количество перевыделений, $\tau = \mathrm{l}\,\mathrm{n}^2(m)$ – время одного перевыделения согласно [13]. Если время доступа к ячейке памяти $O(1)$, то временна́я сложность зависит линейно от n при $n \geq d*\tau$.

Отношение размеров исходного участка и наименьшего из бóльших размеров перевыделяемых участков выражается формулой (3):

$$\frac{x_{i+1}}{x_i}=\frac{\left(k*\sqrt[q]{x_i}+b\right)^q}{x_i}, A=\lim_{i\to\infty}\frac{x_{i+1}}{x_i}=k^q. \qquad (3)$$

Из (3) можно оценить дополнительные затраты запасной памяти как A. Для добавления одного элемента данных в участок размером x_i может потребоваться в k^q раз больше памяти, чем было изначально.

Константы k, b, q могут быть подобраны таким образом, что при сокращении затрат памяти временна́я сложность не будет превосходить линейно-логарифмическую оценку. Преимущество использования хранимых правил в гибкости, так как они позволяют определять стратегии, которые нельзя определить с помощью формирования правил по формуле (2). Однако для хранения правил необходим дополнительный объем памяти. При этом следует брать во внимание зависимость бинарного поиска от числа правил. Метаправила предпочтительны для сохранения объема памяти, а также в случае их аппаратной реализации.

Методика и результаты тестирования операции перевыделения участков памяти

Описанная модель была реализована на языке C++ с использованием программной реализации системы [13] (назовем ее em). В табл. 1 приведено описание четырех стратегий, выбранных для экспериментальной проверки реализации. Компоненты i-го правила каждой из стратегий определяются согласно формуле (2), взятой с уникальным набором констант k, b, q.

Таблица 1. **Описание стратегий для тестирования**

Наименование	Вид зависимости	Константы
геометрическая	$x_i = 2^i$	$k=2, b=0, q=1$
степенная	$x_i = (1+580*i)^2$	$k=1, b=580, q=2$
линейная (a)	$x_i = 17*i+1$	$k=1, b=17, q=1$
линейная ($б$)	$x_i = 4133222*i+1$	$k=1, b=4133222, q=1$

Объем памяти $m = 9*2^{24}$ ячеек. Размер ячейки памяти 32 бита. Пространство признаков для тестирования включает старый размер l выделенного участка памяти, новый размер u выделяемого участка памяти, время работы t операции перевыделения участка размером l на размер u, шаг гистерезиса $h \in \{0,1\}$, способ реализации функции поиска. Пара значений $\langle l, u\rangle \in V \times V$, где $V = \{2097151*i+1 \mid i = 0,\ldots,16\}$. Количество точек с координатами $\langle l, u\rangle$ – 17*17.

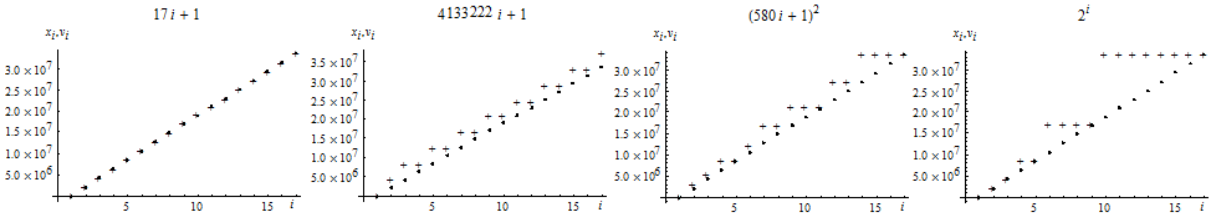


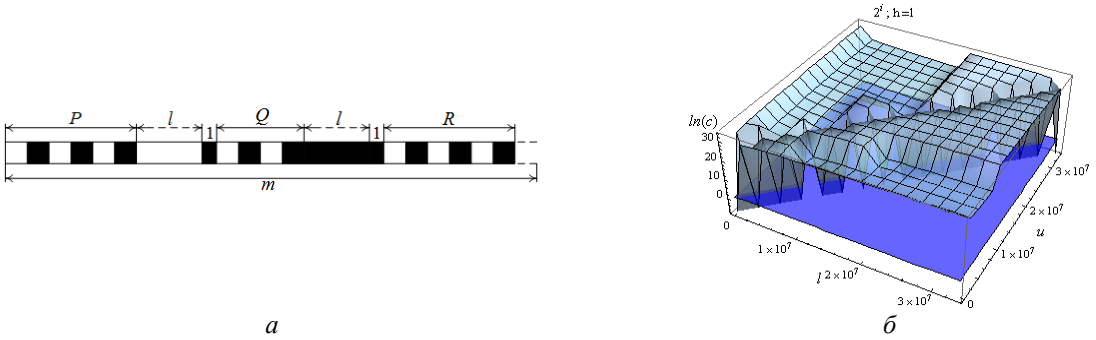
Рис. 2. Зависимость фактического размера x_i выделяемого участка памяти и требуемого размера v_i от номера правила

Использовалась компьютерная конфигурация с ЦП Intel i5-3570, ОЗУ 1x8GB 1333MHz, под управлением ОС *x*64 Windows 8. На рис. 2 для каждой из тестируемых стратегий кружками изображены требуемые размеры v_i, а крестиками – фактические размеры x_i, образующие множество $X = \{x_i \mid (x_i = g(1)(v_i)) \wedge (v_i \in V)\}$.

Система em основана на стратегии выбора первого подходящего незанятого участка. Поэтому для приближения эксперимента к наихудшим условиям память фрагментировалась так, как изображено на рис. 3, *а*. Черные области означают занятые участки памяти. Области *P*, *Q*, *R* равны *p*, фиксированы и вычисляются по формуле:

$$p = \left\lfloor \varphi * \frac{m - 2 * \max(V) - \max(X) - 2 - \rho}{3 * (\delta + 1)} \right\rfloor, \qquad (4)$$

где $\varphi = 1$ – коэффициент степени фрагментации; $\rho = 8$ – число служебных ячеек; $\delta = 1$ – размер участков, которыми области *P*, *Q*, *R* фрагментировались так, что незанятый участок размера δ ячеек чередовался с занятым участком размера 1 ячейка. При этом участки размером $< l$ могут быть помещены в область размером l между *P* и *Q*, а участки размером $> l$ помещаются в незанятую область памяти правее *R*. Формула (4) для $\varphi = 1$ дает близкое к максимальному допустимому значение $p = 7781178$ ячеек, общее для всех тестируемых стратегий. Результаты тестирования стратегий приведены для $\varphi = 1$ и изображены на рис. 3, 4.



а *б*

Рис. 3. Фрагментация памяти (*а*) и график пропускной способности в логарифмическом масштабе (*б*)

Графики на рис. 4, *а*, *в* показывают зависимость времени исполнения (в нс) операции перевыделения от размера выделяемых участков для стратегий без гистерезиса и с гистерезисом соответственно при использовании функции поиска на основе метаправил. Подписи max, mean и min указывают на максимальное, среднее и минимальное значения соответственно, полученные на множестве из 17×17 точек.

Графики на рис. 4, *б* показывают, во сколько раз время исполнения (в нс) t_1 операции перевыделения при использовании стратегий без гистерезиса с функцией поиска на основе метаправил отличается от времени исполнения t_2 операции перевыделения при использовании тех же стратегий, но с функцией поиска на основе хранимых правил. Графики на рис. 4, *г* показывают то же, что и графики на рис. 4, *б*, но для стратегий с гистерезисом. Темные области на графиках с рис. 4, *б*, *г* указывают на области, где отношение времен меньше либо равно 1.

График на рис. 3, *б* в логарифмическом масштабе по оси ln(*c*) для геометрической стратегии с гистерезисом показывает зависимость натурального логарифма пропускной способности (бит/с) от размера, на который изменяется выделенный участок. Темная плоскость на графике отмечает нулевую пропускную способность. Значения *c* на главной диагонали стремятся к минус бесконечности, так как $l = u$.

Сравнительное тестирование различных систем динамического управления памятью дано, например, в [12]. Для сравнения с предлагаемой системой em была взята система tcmalloc. Алгоритм операции перевыделения в системе tcmalloc основан на 1) выделении нового участка памяти с фактическим размером, большим на 25 %, чем исходный, при его увеличении и 2) сокращении фактического размера исходного участка памяти до размера затребованного при уменьшении исходного более, чем на 50 %. Аналогичная стратегия была реализована для системы em. Графики на рис. 5 в логарифмическом масштабе по оси $\log_2(t_{em}/t_{tc})$

показывают отношение времен исполнения (в нс) операций перевыделения в системах em (t_{em}) и tcmalloc (t_{tc}) для размеров перевыделяемого участка 512 ячеек (5, *а*, *г*), 4114 ячеек (5, *б*, *д*) и 2^{25} ячеек (5, *в*, *е*). На рис. 5, *а, б, в* приведены результаты для случая нефрагментированной памяти. На рис. 5, *г, д, е* приведены результаты для случая фрагментированной памяти, когда $\delta = 5$, а $p = 1024$ (рис. 5, *г, д*) и $p = 1$ (рис. 5, *е*).

В системе tcmalloc память фрагментировалась следующим образом: последовательно выделялись 1) участки с размерами 4 байта и $4*\delta$ байт (т.к. размер ячейки в системе em равен 32 бита) для областей P, Q, R и 2) другие участки, согласно схеме фрагментации на рис. 3, *а*. После этого высвобождался каждый нечетный участок с минимальным адресом (всего высвобождалось $3*p$ участков).

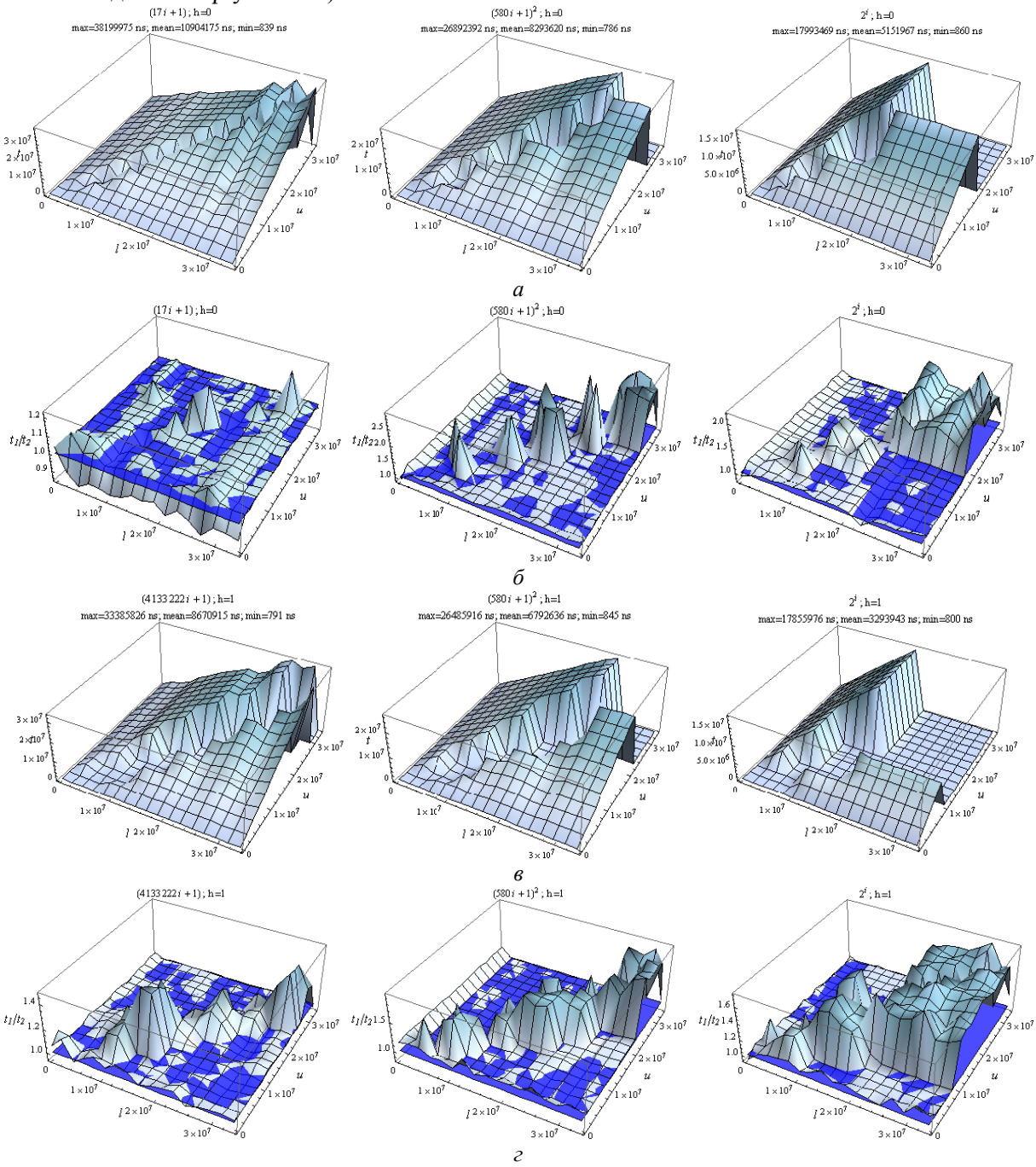


Рис. 4. Фактическое время и отношение времен исполнения операций перевыделения для стратегий без гистерезиса и с гистерезисом при использовании различных функций поиска

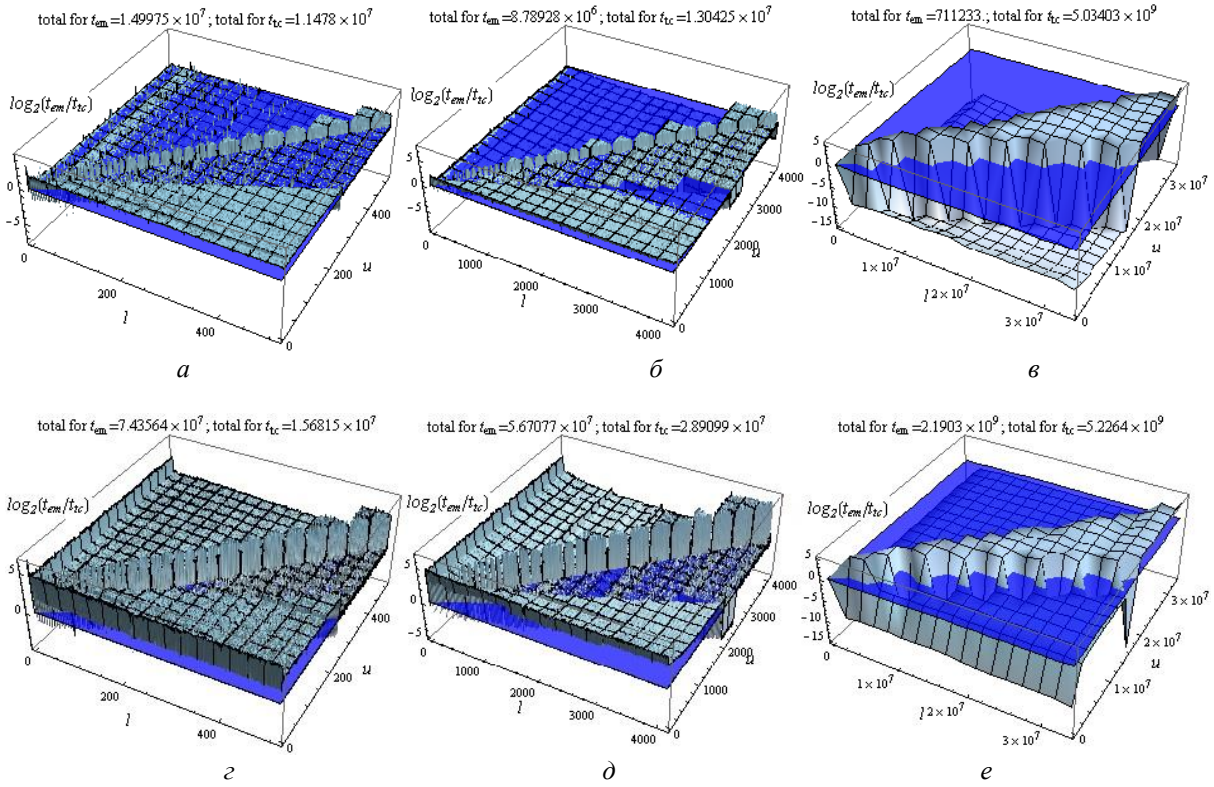


Рис. 5. Отношение времен исполнения операций перевыделения для систем em и tcmalloc

Заключение

Задача динамического распределения участков памяти может быть решена с использованием разработанной модели распределения участков памяти за среднее амортизационное время $O(1)$, если время доступа к ячейке памяти в зависимости от размера памяти равно $O(1)$, а количество записей данных в ячейки зависит от произведения времени одного перевыделения на количество перевыделений как минимум прямо пропорционально. Снижение временно́й сложности повышает расход памяти. Для любой последовательности операций выделения, высвобождения, перевыделения перерасход памяти по причине внешней фрагментации оценивается как $O(S*\log_2(S))$. Для достижения компромисса между затратами дополнительной памяти и временем исполнения интерес могут вызывать степенные стратегии и стратегии на основе геометрической прогрессии.

OPERATIONS ON MANAGEMENT DATA ARRAYS IN LINEAR MEMORY

V.P. IVASHENKO, S.V. SINTSOV

Abstract

This paper considers a task of management data arrays in memory. As a part of a solution of the task authors present a way of implementing operations on management data arrays based on strategies of dynamic memory chunks allocation, which is done at the devices management level where knowledge-based system is executed. Tests results of the implementation of the proposed way are presented.

Keywords: linearly addressable memory, dynamic allocation, memory location.

Список литературы

1. *Гаврилова Т.А., Хорошевский В.Ф.* Базы знаний интеллектуальных систем. СПб., 2000.
2. *Голенков В.В., Гулякина Н.А.* // OSTIS. 2013. С. 55–77.
3. *Кнут Д.* Искусство программирования. Том 1. Основные алгоритмы. М., 2002.
4. *Кнут Д.* Искусство программирования. Том 3. Сортировка и поиск. М., 2014.
5. *Кормен Т.* Алгоритмы: построение и анализ. М., 2013.
6. *Новиков Ф.* Дискретная математика для программистов. СПб., 2009.
7. *Голенков В.В., Гулякина Н.А.* // OSTIS. 2012. С. 23–52.
8. *Masmano M., Ripoll I., Balbastre P. et. al* // Real-Time Systems. 2008. Vol. 40. P. 149–179.
9. TCMalloc: Thread-Caching Malloc. [Электронный ресурс]. – Режим доступа: https://google-perftools.googlecode.com/svn/trunk/doc/tcmalloc.html. Дата доступа: 20.03.2016.
10. *Evans J.* // Proceedings 3rd Technical BSD Conference. Ottawa, May 12–13, 2006.
11. *Berger E., McKinley K., Blumofe R. et. al* // Proc. 9th International Conference on ASPLOS. Cambridge, Nov. 12–15, 2000.
12. Lockless: Benchmarks of the Lockless Memory Allocator. [Электронный ресурс]. – Режим доступа: http://locklessinc.com/benchmarks_allocator.shtml. Дата доступа: 20.03.2016.
13. *Ивашенко В.П.* // Карповские научные чтения : сб. науч. ст. Часть 1. 2012. № 6. С. 196–201.
14. *Robson J. M.* // ACM Computer Journal. 1977. Vol. 20. P. 242–244.
15. *Brodnik A., Carlsson S., Demaine E.D. et. al* // Proc. of 6th International WADS. London, 1999. P. 37–48.